"IBM continues to build the past, Microsoft tries to hold us in present, and Oracle continues to drive us into the future..."

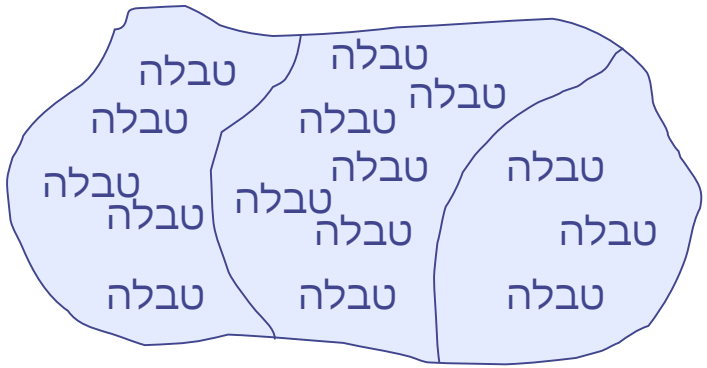
Rich Niemiec,
president of the International Oracle Users Group (IOUG)

'uiaelkeg cipe`l x"c

leokhg@012.net.il
www.geocities.com/leokhg

Oracle basic definitions

Oracle זהו מסד הנתונים טבלאי שמנוהל ע"י שפת SQL ומכיל טכנולוגיה מונחית-עצמים



עצמים של Oracle עיקריים:

מרחב טבלאות	Tablespace
• טבלה	Table
• אינדקס	Index
• תצפית (היבט)	View
• ספרור	Sequence
• הדק (דלגלג)	Trigger
• פרוצדורה מובנת	Stored procedure
• חבילה	Package
• משתמש	User
• תפקיד	Role
• הרשאות	Grants
• מילון נתונים	Data dictionary

column - עמודה
attribute - תכונה

row - שורה

	Column 1	Column 2	...	Column N
	Name1	Name2		NameN
Row 1				
Row 2				
...				
Row M				

Oracle data types

Data type	Description	Examples
CHAR(*size*)	מחרוזת בעלת אורך קבוע. אורך המקסימלי הוגדר ע"י ***size***: 1 ≤ ***size*** ≤ 2000 bytes (Default: 1)	**CREATE TABLE** Person (Id **CHAR**(9), LName **VARCHAR2**(30), FName **VARCHAR2**(30), BDate **DATE**, Photo **BLOB**, Salary **NUMBER**(7, 2)); **INSERT INTO** Person **VALUES**('030424547', 'Parker', 'Anthony', '25/FEB/1972', NULL, 12345.67);
VARCHAR2(*size*)	מחרוזת בעלת אורך משתנה. אורך המקסימלי הוגדר ע"י ***size***: 1 ≤ ***size*** ≤ 4000 bytes (***size*** must be specified)	
NUMBER(*p,s*)	***p*** - אורך הנתון הנומרי ***s*** - מספר הספרות העשרוניות (Default: 38) 1 ≤ ***p*** ≤ 38 (Default: 0) -84 ≤ ***s*** ≤ 127	
DATE	תאריכים החל מ- 01/01/4712 לפני הספירה ועד 31/12/4712 לספירה	
BLOB	עצם בינרי גדול (binary large object) גודל המרבי - 4GB	
LONG	מחרוזת באורך משתנה עד 2GB	
ROWID	Hexadecimal string representing the unique address of a row in its table. This data type is primarily for values returned by the ROWID pseudo column	

Oracle data types

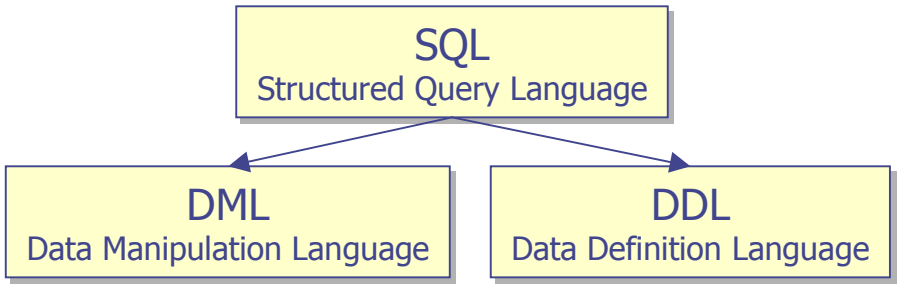
```
SQL> SELECT ROWID, Id, LName, FName, BDate, Salary
  2  FROM Person;

ROWID              ID         LNAME    FNAME      BDATE         SALARY
------------------ ---------- -------- ---------- --------- ----------
AAAFaoAADAAAAHDAAA 030424547 Parker   Anthony    25-FEB-72  12345.67
```

100	NUMBER(3)	100	Fixed point
123.56	NUMBER(4,1)	123.6	Fixed point
123.56	NUMBER(5,2)	123.56	Fixed point
123.56	NUMBER(6,3)	123.560	Fixed point
123.56	NUMBER(6,-1)	120	Fixed point
123.56	NUMBER(4,2)	Error	Fixed point
123.56	NUMBER	123.56	Floating point

INTEGER = NUMBER(38)
FLOAT = NUMBER

FLOAT(***b***) – floating point number with binary precision ***b***
($1 \le b \le 126$)
$\log_2 10 = 3.32193$

יסודות שפת SQL



SELECT <שדות>
FROM <טבלה>
WHERE <מגבלות בחירה>
ORDER BY <תנאי מיון>;

```
SQL> SELECT * FROM Emp;

 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 19-APR-87        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-MAY-87        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

14 rows selected.
```

יסודות שפת SQL (המשך...)

```
SQL> SELECT EmpNo, EName
  2  FROM Emp;

EMPNO  ENAME
------ ----------
  7369 SMITH
  7499 ALLEN
  7521 WARD
  7566 JONES
  7654 MARTIN
  7698 BLAKE
  7782 CLARK
  7788 SCOTT
  7839 KING
  7844 TURNER
  7876 ADAMS
  7900 JAMES
  7902 FORD
  7934 MILLER

14 rows selected.
```

1

```
SQL> SELECT EmpNo, EName FROM Emp
  2  WHERE ( EmpNo > 7500 AND EmpNo <= 7700 )
  3     OR ( EmpNo > 7800 AND EmpNo <= 7900 );

EMPNO  ENAME
------ ----------
  7839 KING
  7844 TURNER
  7876 ADAMS
  7900 JAMES
  7521 WARD
  7566 JONES
  7654 MARTIN
  7698 BLAKE

8 rows selected.
```

3

```
SQL> SELECT EmpNo, EName, HireDate
  2  FROM Emp
  3  WHERE EmpNo > 7500
  4    AND EmpNo <= 7900;

EMPNO  ENAME      HIREDATE
------ ---------- ---------
  7521 WARD       22-FEB-81
  7566 JONES      02-APR-81
  7654 MARTIN     28-SEP-81
  7698 BLAKE      01-MAY-81
  7782 CLARK      09-JUN-81
  7788 SCOTT      19-APR-87
  7839 KING       17-NOV-81
  7844 TURNER     08-SEP-81
  7876 ADAMS      23-MAY-87
  7900 JAMES      03-DEC-81

10 rows selected.
```

2

יסודות שפת SQL (המשך...)

```
SQL> SELECT EmpNo, EName, HireDate
  2  FROM Emp
  3  WHERE EmpNo BETWEEN 7500 AND 7900;

EMPNO  ENAME      HIREDATE
------ ---------- ---------
  7521 WARD       22-FEB-81
  7566 JONES      02-APR-81
  7654 MARTIN     28-SEP-81
  7698 BLAKE      01-MAY-81
  7782 CLARK      09-JUN-81
  7788 SCOTT      19-APR-87
  7839 KING       17-NOV-81
  7844 TURNER     08-SEP-81
  7876 ADAMS      23-MAY-87
  7900 JAMES      03-DEC-81

10 rows selected.
```

2

```
SQL> SELECT EmpNo, EName
  2  FROM Emp
  3  WHERE EName = 'MARTIN';

EMPNO  ENAME
------ ----------
  7654 MARTIN
```

1

```
SQL> SELECT EmpNo, EName FROM Emp
  2  WHERE EmpNo IN (7521,7876,7782);

EMPNO  ENAME
------ ----------
  7782 CLARK
  7876 ADAMS
  7521 WARD
```

3

```
SQL> SELECT EmpNo, EName
  2  FROM Emp
  3  WHERE EName LIKE '%S';

EMPNO  ENAME
------ ----------
  7566 JONES
  7876 ADAMS
  7900 JAMES
```

5

```
SQL> SELECT EmpNo, EName
  2  FROM Emp
  3  WHERE EName LIKE 'M%';

EMPNO  ENAME
------ ----------
  7654 MARTIN
  7934 MILLER
```

4

```
SQL> SELECT EmpNo, EName, Job, Sal
  2  FROM Emp
  3  WHERE EName LIKE '_LA%';

EMPNO  ENAME      JOB          SAL
------ ---------- --------- -----
  7698 BLAKE      MANAGER    2850
  7782 CLARK      MANAGER    2450
```

6

יסודות שפת SQL (המשך...)

```
SQL> SELECT EmpNo, EName,
  2         Job, Sal
  3  FROM Emp
  4  ORDER BY Sal;

EMPNO ENAME   JOB          SAL
----- ------- --------- -----
 7369 SMITH   CLERK        800
 7900 JAMES   CLERK        950
 7876 ADAMS   CLERK       1100
 7521 WARD    SALESMAN    1250
 7654 MARTIN  SALESMAN    1250
 7934 MILLER  CLERK       1300
 7844 TURNER  SALESMAN    1500
 7499 ALLEN   SALESMAN    1600
 7782 CLARK   MANAGER     2450
 7698 BLAKE   MANAGER     2850
 7566 JONES   MANAGER     2975
 7788 SCOTT   ANALYST     3000
 7902 FORD    ANALYST     3000
 7839 KING    PRESIDENT   5000

14 rows selected.
```

3

```
SQL> SELECT EName, Job, Sal
  2  FROM Emp
  3  WHERE EName LIKE '%A%A%';

ENAME      JOB          SAL
---------- --------- -----
ADAMS      CLERK       1100
```

1

```
SQL> SELECT EmpNo, EName, Job FROM Emp
  2  WHERE Sal > 1300;

EMPNO  ENAME      JOB
------ ---------- ---------
  7499 ALLEN      SALESMAN
  7566 JONES      MANAGER
  7698 BLAKE      MANAGER
  7782 CLARK      MANAGER
  7788 SCOTT      ANALYST
  7839 KING       PRESIDENT
  7844 TURNER     SALESMAN
  7902 FORD       ANALYST

8 rows selected.
```

2

שדה שנמצא בתוך מגבלת **WHERE**
לא חייב להופיע בין השדות של **SELECT**

יסודות שפת SQL (המשך...)

```
SQL> SELECT EmpNo, EName, Job, Sal
  2  FROM Emp
  3  ORDER BY Sal DESC;

EMPNO ENAME   JOB         SAL
----- ------- --------- ----
 7839 KING    PRESIDENT 5000
 7788 SCOTT   ANALYST   3000
 7902 FORD    ANALYST   3000
 7566 JONES   MANAGER   2975
 7698 BLAKE   MANAGER   2850
 7782 CLARK   MANAGER   2450
 7499 ALLEN   SALESMAN  1600
 7844 TURNER  SALESMAN  1500
 7934 MILLER  CLERK     1300
 7521 WARD    SALESMAN  1250
 7654 MARTIN  SALESMAN  1250
 7876 ADAMS   CLERK     1100
 7900 JAMES   CLERK      950
 7369 SMITH   CLERK      800

14 rows selected.
```

1

```
SQL> SELECT EmpNo, EName, Job, DeptNo, Sal
  2  FROM Emp
  3  ORDER BY DeptNo, Sal DESC;

EMPNO  ENAME   JOB       DEPTNO   SAL
-----  ------- --------- ------  ----
 7839  KING    PRESIDENT     10  5000
 7782  CLARK   MANAGER       10  2450
 7934  MILLER  CLERK         10  1300
 7788  SCOTT   ANALYST       20  3000
 7902  FORD    ANALYST       20  3000
 7566  JONES   MANAGER       20  2975
 7876  ADAMS   CLERK         20  1100
 7369  SMITH   CLERK         20   800
 7698  BLAKE   MANAGER       30  2850
 7499  ALLEN   SALESMAN      30  1600
 7844  TURNER  SALESMAN      30  1500
 7521  WARD    SALESMAN      30  1250
 7654  MARTIN  SALESMAN      30  1250
 7900  JAMES   CLERK         30   950

14 rows selected.
```

2

Group-value functions פונקציות קבוצתיות

פונקציה	מחזירה	דוגמה
COUNT	מספר שורות שנבחרו מהטבלה	**SELECT COUNT**(*) **FROM** Emp;
SUM	סה"כ של כל הערכים שנבחרו מן העמודה	**SELECT SUM**(Sal) **FROM** Emp;
MIN	הערך המינימלי המאוחסן בעמודה	**SELECT MIN**((Sal) **FROM** Emp;
MAX	הערך המקסימלי המאוחסן בעמודה	**SELECT MAX**((Sal) **FROM** Emp;
AVG	הערך הממוצע של כל הערכים שנבחרו מן העמודה	**SELECT AVG**((Sal) **FROM** Emp;

```
SQL> column AVG(SAL) format 99999.99
SQL> SELECT SUM(Sal), AVG(Sal), MIN(Sal), MAX(Sal)
  3  FROM Emp;

 SUM(SAL)  AVG(SAL)   MIN(SAL)   MAX(SAL)
---------- --------- ---------- ----------
     29025   2073.21        800       5000
```

1

```
SQL> SELECT COUNT(*)
  2  FROM Emp;

  COUNT(*)
----------
        14

SQL> SELECT COUNT(Comm)
  2  FROM Emp;

COUNT(COMM)
-----------
          4
```

2

What about NULL - ?

- COUNT(*) – with NULLs
- COUNT(*field*) – without NULLs
- SUM, AVG, MIN. MAX – without NULLs

DISTINCT clause

1. forces **SELECT** to select only rows with unique sets of fields

2. forces group function to take in account only unique values of the underlying column

```
SQL> SELECT Job, Sal
  2  FROM Emp
  3  ORDER BY Job, Sal DESC

JOB                 SAL
--------- ----------
ANALYST           3000
ANALYST           3000
CLERK             1300
CLERK             1100
CLERK              950
CLERK              800
MANAGER           2975
MANAGER           2850
MANAGER           2450
PRESIDENT         5000
SALESMAN          1600
SALESMAN          1500
SALESMAN          1250
SALESMAN          1250

14 rows selected.
```

1

```
SQL> SELECT DISTINCT Job, Sal
  2  FROM Emp
  3  ORDER BY Job, Sal DESC;

JOB                 SAL
--------- ----------
ANALYST           3000
CLERK             1300
CLERK             1100
CLERK              950
CLERK              800
MANAGER           2975
MANAGER           2850
MANAGER           2450
PRESIDENT         5000
SALESMAN          1600
SALESMAN          1500
SALESMAN          1250

12 rows selected.
```

2

```
SQL> SELECT COUNT(Job)
  2  FROM Emp;

COUNT(JOB)
----------
        14

SQL> SELECT COUNT( DISTINCT Job )
  2  FROM Emp;

COUNT(DISTINCTJOB)
------------------
                 5
```

3

GROUP BY and HAVING

```
SQL> SELECT Job, AVG(Sal) FROM Emp;
SELECT Job, AVG(Sal) FROM Emp
       *
ERROR at line 1:
ORA-00937: not a single-group group function

SQL> SELECT Job, AVG(Sal) FROM Emp
  2  GROUP BY Job;

JOB         AVG(SAL)
--------- ---------
ANALYST      3000.00
CLERK        1037.50
MANAGER      2758.33
PRESIDENT    5000.00
SALESMAN     1400.00

SQL> SELECT Job, AVG(Sal) FROM Emp
  2  GROUP BY Job
  3  HAVING Job <> 'PRESIDENT';

JOB          AVG(SAL)
--------- ----------
ANALYST       3000.00
CLERK         1037.50
MANAGER       2758.33
SALESMAN      1400.00
```

SELECT <שדות>
FROM <טבלה>
WHERE <מגבלות בחירה>
GROUP BY <הגדרת קבוצות>
HAVING <מגבלות קבוצה>
ORDER BY <תנאיי מיון>;

סדר פעולות

1) It chooses rows matching to **WHERE** clause
2) It groups selected rows together based on the **GROUP BY**
3) It calculates the results of the group functions for each group
4) It chooses and eliminates groups based on the **HAVING** clause
5) It orders the groups in the **ORDER BY**. The order by must use either a group function or a column in the **GROUP BY** clause

איחוד וחיתוך שאילתות SQL

UNION
UNION ALL
INTERSECT
MINUS

<שאילתה 1>
<פעולה>
<שאילתה 2>

```
SQL> DESCRIBE LongTime;
 Name      Null?    Type
 --------- -------- -------------
 NAME      NOT NULL VARCHAR2(25)
 LODGING            VARCHAR2(15)
 AGE                NUMBER

SQL> SELECT Name FROM LongTime;

NAME
-------------------------
ADAH TALBOT
DICK JONES
DONALD ROLLO
ELBERT TALBOT
GEORGE OSCAR
PAT LAVAY
PETER LAWSON
WILFRED LOWELL

8 rows selected.
```

1

```
SQL> DESCRIBE Prospect;
 Name      Null?    Type
 --------- -------- -------------
 NAME      NOT NULL VARCHAR2(25)
 ADDRESS   VARCHAR2(35)

SQL> SELECT Name FROM Prospect;

NAME
-------------------------
ADAH TALBOT
DORY KENSON
ELBERT TALBOT
GEORGE PHEPPS
PAT LAVAY
TED BUTCHER
JED HOPKINS
WILFRED LOWELL

8 rows selected.
```

2

איחוד וחיתוך שאילתות SQL (המשך...)

```
SQL> SELECT Name FROM LongTime
  2  UNION ALL
  3  SELECT Name FROM Prospect;

NAME
------------------------
ADAH TALBOT
DICK JONES
DONALD ROLLO
ELBERT TALBOT
GEORGE OSCAR
PAT LAVAY
PETER LAWSON
WILFRED LOWELL
ADAH TALBOT
DORY KENSON
ELBERT TALBOT
GEORGE PHEPPS
PAT LAVAY
TED BUTCHER
JED HOPKINS
WILFRED LOWELL

16 rows selected.
```

2

```
SQL> SELECT Name FROM LongTime
  2  UNION
  3  SELECT Name FROM Prospect;

NAME
------------------------
ADAH TALBOT
DICK JONES
DONALD ROLLO
DORY KENSON
ELBERT TALBOT
GEORGE OSCAR
GEORGE PHEPPS
JED HOPKINS
PAT LAVAY
PETER LAWSON
TED BUTCHER
WILFRED LOWELL

12 rows selected.
```

1

איחוד וחיתוך שאילתות SQL (המשך...)

```
SQL> SELECT Name FROM LongTime
  2  INTERSECT
  3  SELECT Name FROM Prospect;

NAME
------------------------
ADAH TALBOT
ELBERT TALBOT
PAT LAVAY
WILFRED LOWELL
```

1

```
SQL> SELECT Name FROM LongTime
  2  MINUS
  3  SELECT Name FROM Prospect;

NAME
------------------------
DICK JONES
DONALD ROLLO
GEORGE OSCAR
PETER LAWSON
```

2

- The **select**s must have the same number of columns
- The **select**s might not have individual **order by** clause
- The matching top and bottom columns must be the same data type
- The matching top and bottom columns needn't be the same length
- The **select**s needn't have the same column names
- The **selects** can have common **order by** clause
- In the common **order by** clause instead name of column (columns) may be its number

איחוד וחיתוך שאילתות SQL (המשך...)

```
SQL> SELECT Name, Lodging, Age FROM LongTime
  2  UNION
  3  SELECT Name, Address, 0 FROM Prospect
  4  ORDER BY 3;

NAME                 LODGING                              AGE
-------------------- ---------------------------------- -------
ADAH TALBOT          23 ZWING, EDMESTON                       0
DORY KENSON          GEN. DEL., BAYBAC                        0
ELBERT TALBOT        3 MILE ROAD, WALPOLE                     0
GEORGE PHEPPS        206 POLE, KINGSLEY                       0
JED HOPKINS          GEN. DEL., TURBOW                        0
PAT LAVAY            1 EASY ST, JACKSON                       0
TED BUTCHER          RFD 1, BRIGHTON                          0
WILFRED LOWELL                                                0
DONALD ROLLO         MATTS                                   16
DICK JONES           ROSE HILL                               18
PAT LAVAY            ROSE HILL                               21
ADAH TALBOT          PAPA KING                               23
PETER LAWSON         CRANMER                                 25
GEORGE OSCAR         ROSE HILL                               41
ELBERT TALBOT        WEITBROCHT                              43
WILFRED LOWELL                                               67

16 rows selected.
```

Pseudo-columns. DUAL table

Pseudo-column	Value returned
NULL	Null value
RowID	Returns the unique row identifier for a row
RowNum	Returns the number in which a row was selected from table (1, 2, …)
SysDate	The current date and time
User	Name of the current user
UID	User ID (a unique number assigned to each user)

```
SQL> DESC Dual
 Name    Null?    Type
 ------ -------- ------------
 DUMMY            VARCHAR2(1)
```

```
SQL> column Dummy format a5
SQL> SELECT * FROM DUAL;

DUMMY
-----
X
```

```
SQL> SELECT RowID, RowNum, SysDate, User, UID FROM Dual;

ROWID                  ROWNUM SYSDATE   USER     UID
------------------ -------- --------- -------- ----------
AAAADDAABAAAAHSAAA         1 16-MAR-02 SCOTT            32
```

פונקציות סטנדרטיות לעיבוד מחרוזות

תוצאה	דוגמה	מחזירה	פונקציה
'ABCPQR'	**SELECT** 'ABC' || 'PQR' **FROM** Dual;	שרשור	**||**
'CDEF'	**SELECT SUBSTR**('ABCDEFGHIJ', 3, 4) **FROM** Dual;	חלק ממחרוזת	**SUBSTR**(str, from [,cnt])
3	**SELECT LENGTH**('ABC') **FROM** Dual;	אורך המחרוזת	**LENGTH**(str)
'AABBCC'	**SELECT UPPER**('AaBbCc') **FROM** Dual;	מחרוזת עם אותיות גדולות בלבד	**UPPER**(str)
'aabbcc'	**SELECT LOWER**('AaBbCc') **FROM** Dual;	מחרוזת עם אותיות קטנות בלבד	**LOWER**(str)
'ORGE' 'GEOR'	**SELECT LTRIM**('GEORGE', 'GE') **FROM** Dual; **SELECT RTRIM**('GEORGE', 'GE') **FROM** Dual;	Trims all occurrences of a set characters off on the left(right) side of string	**LTRIM**(str, [,set]) **RTRIM**(str, [,set])
'Bruce Scott'	**SELECT INITCAP**('bruce scott') **FROM** Dual;	First letter capitalization	**INITCAP**(string)

פונקציות סטנדרטיות לעיבוד מחרוזות (המשך...)

פונקציה	מחזירה	דוגמה	תוצאה
INSTR(str, set [,start [,occ]])	Location of *set* in a *string.* Looking for *occ* occurrence from *start* position	**SELECT INSTR**('function has parameter, that has type', 'has', 1, 2) **FROM** Dual;	30
REPLACE(str, if, then)	Replaces all appearance of *if* in *str* with *then*	**SELECT REPLACE**('GEORGE', 'GE', 'K') **FROM** Dual;	'KORK'
TRANSLATE(str, if, then)	Positional substitution in *str* from *then* on base of *if* mask	**SELECT TRANSLATE**('HAL', 'ABCDEHIJKLM', 'BCDEHIJKLMA') **FROM** Dual;	'IBM'

פונקציות סטנדרטיות נומריות

תוצאה	דוגמה	מחזירה	פונקציה
123.46 123.45	**ROUND**(123.458, 2) **TRUNC**(123.458, 2)	Rounds (truncates) value *val to* precision *prcs*	**ROUND**(val, prcs) **TRUNC**(val, prcs)
2 -1 1 -2	**CEIL**(1.3) **CEIL**(-1.3) **FLOOR**(1.3) **FLOOR**(-1.3)	Smallest integer that is >= (CEIL) or <= (FLOOR) to *val*	**CEIL**(val) **FLOOR**(val)
4.1	**ABS**(4.1) **ABS**(-4.1)	Absolute value of *val*	**ABS**(val)
3	**MOD**(23, 5)	*val1* % *val2*	**MOD**(val1, val2)
1 -1	**SIGN**(45) **SIGN**(-45)	Sign of *val*	**SIGN**(val)
2.828427 81	**POWER**(2, 1.5) **POWER**(3, 4)	Raises *val* to *exponent*	**POWER**(val, exponent)

SQRT, EXP, LN, LOG, SIN, SINH, COS, COSH, TAN, TANH...

פונקציות עבור נתונים מטיפוס DATE

פונקציה	מחזירה	דוגמה	תוצאה
LAST_DAY(date)	Date of last day of month that *date* is in	**LAST_DAY**('17-MAR-02')	31-MAR-02
ADD_MONTH(date, cnt)	Adds *cnt* months to *date*	**ADD_MONTH**('31-MAR-02', 3) **ADD_MONTH**('28-FEB-01', -4)	30-JUN-02 31-OCT-00
MONTHS_BETWEEN(date2, date1)	*date2-date1* in month	**MONTH_BETWEEN**('10-AUG-02', '17-MAR-01')	16.7742
NEXT_DAY(date, day)	Date of next *day* after *date*	**NEXT_DAY**('18-MAR-02', 1)	24-MAR-02
ROUND(date) **TRUNC**(date)	Rounds (truncates) *date*	**ROUND**(**TO_DATE**('17-MAR-02 17:38:45', 'dd-MON-yyyy hh24:mi:ss')) **TRUNC**(**TO_DATE**('17-MAR-02 17:38:45', 'dd-MON-yyyy hh24:mi:ss'))	18-MAR-02 17-MAR-02

פונקציות להמרת נתונים

TO_CHAR(date, 'format')

TO_CHAR(**SYSDATE**, 'DD/MM/YYYY')

17/03/2001

TO_CHAR(number, 'format')

TO_CHAR(1234.56, '$9,999.9')

$1,234.6

TO_DATE(string, 'format')

TO_DATE('17/03/2001', ‘DD/MM/YYYY')

17/03/2001

TO_NUMBER(string)

TO_NUMBER('333.46')

333.46

פורמטים

תאריכים

Format	Meaning	Example
DD	Number of day in the month	23
MM	Number of month in the year	03
YYYY	Four-digit year	2002
HH	Hour of day (1 – 12)	05
HH24	Hour of day (1 – 24)	18
MI	Minute of hour	07
SS	Second of minute	47
MON	3-letter abbreviation of month	AUG
MONTH	Month fully spelled out	AUGUST
D	Number of the day in the week	6
DY	3-letter abbreviation of day	FRI
DAY	Day fully spelled out	FRIDAY
/ , . - :	Display punctuation	17-03-2002

מספרים

9 - digit or space
TO_CHAR(123.56, '999.9')
123.6

0 - digit or 0
TO_CHAR(123.56, '0999.9')
0123.6

EEEE – scientific format
TO_CHAR(123.7, '9.99EEEE')
1.24E+02

S – minus or plus
TO_CHAR(123.56, 'S999.9')
+123.6

. - decimal point
, - comma to display

פונקציות שונות

NVL(value, alt) – if *value* is NULL returns *alt*, else returns *value*

```
SQL> SELECT EName, Sal, Comm, Sal+Comm AS Total FROM Emp
  2  WHERE RowNum <= 6 ORDER BY EName;

ENAME              SAL       COMM      TOTAL
---------- ---------- ---------- ----------
ALLEN            1600        300       1900
BLAKE            2850
JONES            2975
MARTIN           1250       1400       2650
SMITH             800
WARD             1250        500       1750

6 rows selected.
```

1

```
SQL> SELECT EName, Sal, Comm, Sal+ NVL(Comm,0) AS Total FROM Emp
  2  WHERE RowNum <= 6 ORDER BY EName;

ENAME              SAL       COMM      TOTAL
---------- ---------- ---------- ----------
ALLEN            1600        300       1900
BLAKE            2850                  2850
JONES            2975                  2975
MARTIN           1250       1400       2650
SMITH             800                   800
WARD             1250        500       1750

6 rows selected.
```

2

פונקציות שונות (המשך...)

DECODE(value, if1, then1, if2, then2,...,else) – if *value* equals *if1*, DECODE returns *then1*, if *value* equals *if2*, DECODE returns *then2*, ... If *value* equals none *ifs*, then result of the DECODE is *else*.

DECODE function supports the **if-then-else** operator logic. It's without doubts one of the most powerful in Oracle's SQL

```
SQL> DESC NewsPaper
 Name      Null?    Type
 --------- -------- -------------
 FEATURE   NOT NULL VARCHAR2(15)
 SECTION            CHAR(1)
 PAGE               NUMBER(3)
```

```
SQL> col Section format a7
SQL> SELECT * FROM NewsPaper;

FEATURE         SECTION        PAGE
--------------- ------- ----------
National News   A                1
Sports          D                1
Editorials      A               12
Business        E                1
Weather         C                2
Television      B                7
Births          F                7
Classified      F                8
Modern Life     B                1
Comics          C                4
Movies          B                4
Bridge          B                2
Obituaries      F                6
Doctor Is In    F                6

14 rows selected.
```

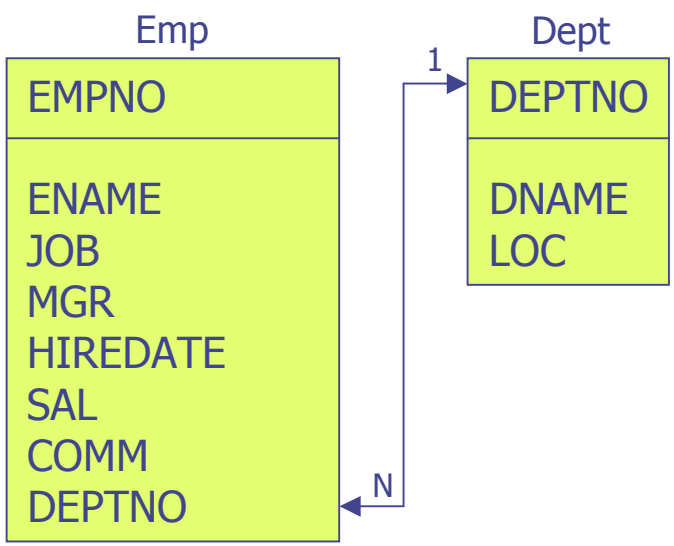
פונקציות שונות (המשך...)

```
SQL> SELECT Feature, Section,
  2         DECODE( Page, '1', 'Front page', 'Turn to ' || Page || ' page' )
  3  FROM NewsPaper;

FEATURE         SECTION DECODE(PAGE,'1','FRONTPAGE','TURNTO'||PAGE||'PAGE')
--------------- ------- ----------------------------------------------------
National News   A       Front page
Sports          D       Front page
Editorials      A       Turn to 12 page
Business        E       Front page
Weather         C       Turn to 2 page
Television      B       Turn to 7 page
Births          F       Turn to 7 page
Classified      F       Turn to 8 page
Modern Life     B       Front page
Comics          C       Turn to 4 page
Movies          B       Turn to 4 page
Bridge          B       Turn to 2 page
Obituaries      F       Turn to 6 page
Doctor Is In    F       Turn to 6 page

14 rows selected.
```

צירוף טבלאות - join

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

```
SQL> SELECT * FROM DEPT;

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




```
SQL> SELECT EMPNO, ENAME, JOB, DNAME
  2  FROM EMP, DEPT

 EMPNO ENAME      JOB       DNAME
---------- ---------- --------- --------------
      7369 SMITH      CLERK     ACCOUNTING
      7499 ALLEN      SALESMAN  ACCOUNTING
      7521 WARD       SALESMAN  ACCOUNTING
      7566 JONES      MANAGER   ACCOUNTING
      7654 MARTIN     SALESMAN  ACCOUNTING
      7698 BLAKE      MANAGER   ACCOUNTING
      7782 CLARK      MANAGER   ACCOUNTING
      7788 SCOTT      ANALYST   ACCOUNTING
      7839 KING       PRESIDENT ACCOUNTING
      7844 TURNER     SALESMAN  ACCOUNTING
      7876 ADAMS      CLERK     ACCOUNTING
      7900 JAMES      CLERK     ACCOUNTING
      7902 FORD       ANALYST   ACCOUNTING
      7934 MILLER     CLERK     ACCOUNTING
      7369 SMITH      CLERK     RESEARCH
      7499 ALLEN      SALESMAN  RESEARCH
      7521 WARD       SALESMAN  RESEARCH
      . . .
      7902 FORD       ANALYST   OPERATIONS
      7934 MILLER     CLERK     OPERATIONS

56 rows selected.
```

צירוף טבלאות - join (המשך...)

```
SQL> SELECT EMPNO, ENAME, JOB, EMP.DEPTNO, DNAME
  2  FROM EMP, DEPT
  3  WHERE EMP.DEPTNO = DEPT.DEPTNO;
  4  ORDER BY ENAME;

     EMPNO ENAME      JOB           DEPTNO DNAME
---------- ---------- --------- ---------- --------------
      7876 ADAMS      CLERK             20 RESEARCH
      7499 ALLEN      SALESMAN          30 SALES
      7698 BLAKE      MANAGER           30 SALES
      7782 CLARK      MANAGER           10 ACCOUNTING
      7902 FORD       ANALYST           20 RESEARCH
      7900 JAMES      CLERK             30 SALES
      7566 JONES      MANAGER           20 RESEARCH
      7839 KING       PRESIDENT         10 ACCOUNTING
      7654 MARTIN     SALESMAN          30 SALES
      7934 MILLER     CLERK             10 ACCOUNTING
      7788 SCOTT      ANALYST           20 RESEARCH
      7369 SMITH      CLERK             20 RESEARCH
      7844 TURNER     SALESMAN          30 SALES
      7521 WARD       SALESMAN          30 SALES

14 rows selected.
```

צירוף טבלאות - join (המשך...)

```
SQL>SELECT E.EmpNo, E.EName, E.Job, E.Deptno,
  2        D.DName, Z.EName AS Boss
  2  FROM Emp E, Dept D, Emp Z
  3  WHERE E.DeptNo = D.DeptNo
  4    AND E.Mgr = Z.EmpNo
  5* ORDER BY E.EName

 EMPNO ENAME    JOB         DEPTNO DNAME          BOSS
------ -------- --------- ------- ------------ --------
  7876 ADAMS    CLERK           20 RESEARCH     SCOTT
  7499 ALLEN    SALESMAN        30 SALES        BLAKE
  7698 BLAKE    MANAGER         30 SALES        KING
  7782 CLARK    MANAGER         10 ACCOUNTING   KING
  7902 FORD     ANALYST         20 RESEARCH     JONES
  7900 JAMES    CLERK           30 SALES        BLAKE
  7566 JONES    MANAGER         20 RESEARCH     KING
  7654 MARTIN   SALESMAN        30 SALES        BLAKE
  7934 MILLER   CLERK           10 ACCOUNTING   CLARK
  7788 SCOTT    ANALYST         20 RESEARCH     JONES
  7369 SMITH    CLERK           20 RESEARCH     FORD
  7844 TURNER   SALESMAN        30 SALES        BLAKE
  7521 WARD     SALESMAN        30 SALES        BLAKE

13 rows selected.
```

צירוף טבלה עם עצמה

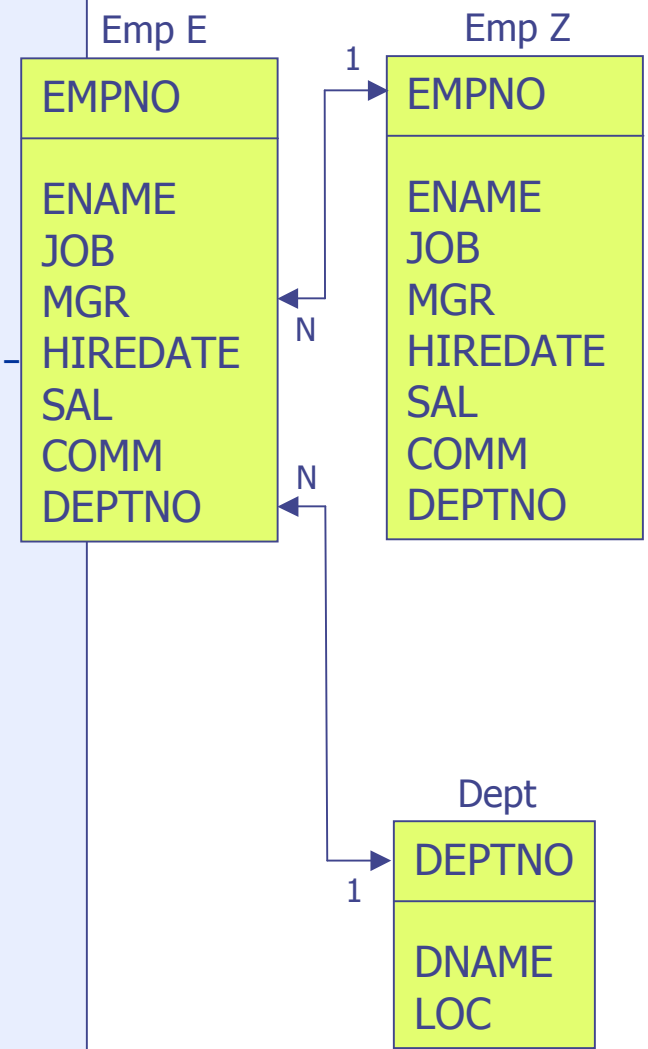


Table creation יצירת טבלאות

```
CREATE TABLE [user_name.] table_name
(
   column_name  data_type  [ DEFAULT value ][ col_constraint ],
   column_name  data_type  [ DEFAULT value ][ col_constraint ],
   . . .
   tbl_constraint,
   tbl_constraint,
   . . .
);
```

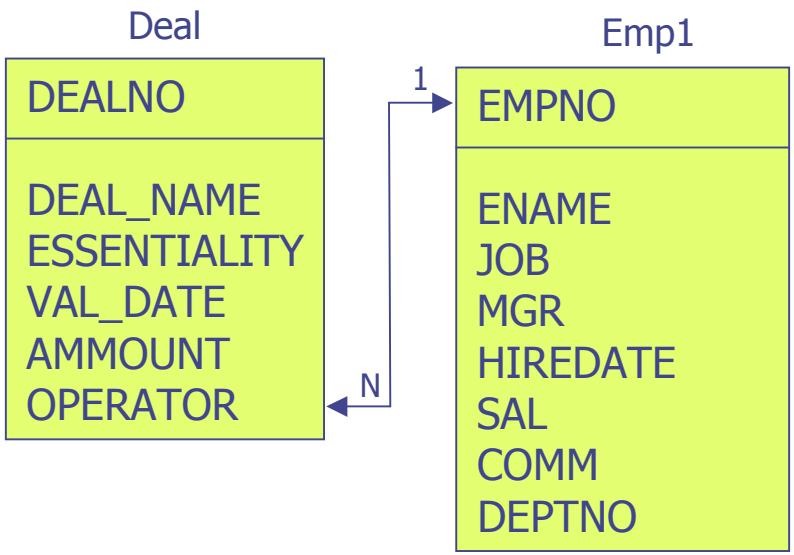
constraint ::= { **NOT NULL** | **PRIMARY KEY** | **CHECK** (condition) | **REFERENCES**... **| ...**}

```
SQL> CREATE TABLE FatClub
 (
  MemberNo    NUMBER(4),
  LastName    VARCHAR2(30),
  FirstName   VARCHAR2(30),
  StartDate   DATE,
  Address     VARCHAR2(30),
  Phone       VARCHAR2(12),
  Height      NUMBER(3),
  Weight      NUMBER(5,2),
  Visits      NUMBER(5)
 );
```

```
SQL> CREATE TABLE FatClub
 (
  MemberNo    NUMBER(4) PRIMARY KEY,
  LastName    VARCHAR2(30) NOT NULL,
  FirstName   VARCHAR2(30),
  StartDate   DATE DEFAULT SYSDATE,
  Address     VARCHAR2(30),
  Phone       VARCHAR2(12) NOT NULL,
  Height      NUMBER(3),
  Weight      NUMBER(5,2) CHECK( Weight > 120 ),
  Visits      NUMBER(5) DEFAULT 0
 );
```

Foreign key מפתח זר

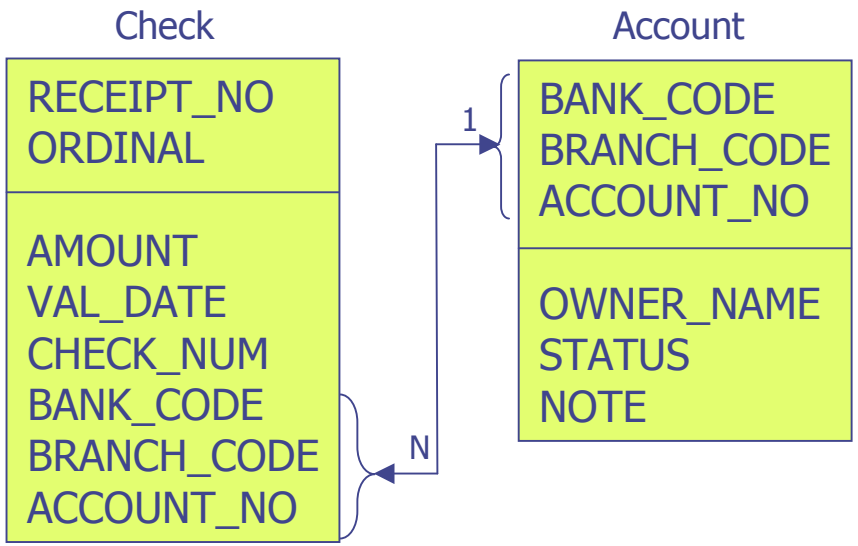
REFERENCES *foreign_table(its_primary_key)* [**ON DELETE CASCADE**]




```
SQL> CREATE TABLE Deal
 (
  DealNo NUMBER(6) PRIMARY KEY,
  Deal_Name VARCHAR2(30) NOT NULL,
  Essentiality VARCHAR2(60),
  Val_Date DATE,
  Amount NUMBER(9,2) CHECK( Amount > 0 ),
  Operator NUMBER(4) REFERENCES Emp1( EmpNo )
 );
```

```
SQL> CREATE TABLE Deal
 (
  DealNo NUMBER(6) PRIMARY KEY,
  Deal_Name VARCHAR2(30) NOT NULL,
  Essentiality VARCHAR2(60),
  Val_Date DATE,
  Amount NUMBER(9,2) CHECK( Amount > 0 ),
  Operator NUMBER(4) REFERENCES Emp1( EmpNo ) ON DELETE CASCADE
 );
```

Multi fields primary and foreign key




```
CREATE TABLE Account
(
   Bank_Code      NUMBER(2) NOT NULL,
   Branch_Code    NUMBER(3) NOT NULL,
   Account_No     VARCHAR2(12) NOT NULL,
   Owner_Name     VARCHAR2(40),
   Status         NUMBER(2) DEFAULT 0
   Note           VARCHAR2(60),
   PRIMARY KEY( Bank_Code,
                Branch_Code,
                Account_No )
);
```

```
CREATE TABLE Checks
(
   Receipt_No     VARCHAR2(6) NOT NULL,
   Ordinal        NUMBER(3) NOT NULL,
   Amount         NUMBER(9,2) CHECK( Amount > 0 ),
   Val_Date       DATE,
   Check_Num      VARCHAR2(8),
   Bank_Code      NUMBER(2) NOT NULL,
   Branch_Code    NUMBER(3) NOT NULL,
   Account_No     VARCHAR2(12) NOT NULL,
   PRIMARY KEY ( Receipt_No, Ordinal ),
   FOREIGN KEY ( Bank_Code, Branch_Code, Account_No )
                        REFERENCES Account( Bank_Code, Branch_Code, Account_No )
);
```

INSERT statement

```
INSERT INTO table [ ( col1 , col2 , …, colN ) ]
VALUES ( val1 , val2 , …, valN  );
```

- רשומה חדשה אחת בלבד נכנסת לטבלה !
- כמות **val_i** == כמות **col_i**
- עמודה שלא ברשימה (***col_1, col_2, …, col_N***) מקבלת **NULL** או **DEFAULT**
- סוג של **val_i** חייב להיות **<u>מתאים</u>** ל-**col_i**
- **val_i** יכול להיות תוצאה של **SELECT** שמחזיר ערך אחד בלבד

```
INSERT INTO table [ ( col1 , col2 , …, colN ) ]
SELECT [ ( cl1 , cl2 , …, clN ) | * ] FROM tb
[ WHERE … ];
```

- כמות הרשומות החדשות שנכנסות ל-*table* תלויה ב- **SELECT** !
- כמות **cl_i** == כמות **col_i**
- עמודה שלא ברשימה (***col_1, col_2, …, col_N***) מקבלת **NULL** או **DEFAULT**
- סוג של **cl_i** חייב להיות **<u>מתאים</u>** ל-**col_i**

INSERT examples

```
INSERT INTO Deal
VALUES( 1, 'Oracle 8i Online Help',
        'Oracle 8i Online Help is sold to DBM Systems Inc',
        TO_DATE( '15/03/2002', 'DD/MM/YYYY' ), 1200.00, 7499 );
INSERT INTO Deal
VALUES( 2, 'Oracle 8.1.6 ODBC drivers',
        'Oracle 8.1.6 ODBC drivers is sold to DBM Systems Inc',
        '21-MAR-02', 3400.00, 7844 );
INSERT INTO Deal ( DealNo, Deal_Name, Operator )
VALUES( 6, 'Oracle 8.1.6 Personal', 7499 );

INSERT INTO Emp1 SELECT * FROM Emp;

CREATE TABLE SalRep
(
        Specif VARCHAR2(15),
        Value NUMBER(9,2)
);

INSERT INTO SalRep
VALUES( 'Minimal salary', ( SELECT MIN( Sal ) FROM Emp ) );

INSERT INTO SalRep
VALUES( 'Average salary', ( SELECT AVG( Sal ) FROM Emp ) );

INSERT INTO SalRep
VALUES( 'Maximal salary', ( SELECT MAX( Sal ) FROM Emp ) );
```

UPDATE statement

```
UPDATE table
SET col1 = val1 , col2 = val2 , …, colN = valN
[WHERE … ];
```

- סוג של val_i חייב להיות **מתאים** ל-col_i
- val_i יכול להיות תוצאה של **SELECT** שמחזיר ערך אחד בלבד
- מתבצע עדכון בכל הרשומות שמספקות את **WHERE**

```
UPDATE table
SET(col1 , col2 , …, colN ) =
   ( SELECT cl1, cl2,…, clN FROM tb [ WHERE … ] )
[WHERE … ];
```

- כמות cl_i == כמות col_i
- סוג של cl_i חייב להיות **מתאים** ל-col_i
- **SELECT** המקונן חייב להחזיר שורה (רשומה) אחד בלבד
- מתבצע עדכון בכל הרשומות שמספקות את **WHERE** של **UPDATE**

UPDATE examples

```
UPDATE Deal
SET Essentiality = 'Oracle 8.1.6 Personal is bought by Engeneering College',
    Amount = 2200.00
WHERE DealNo = 6;

UPDATE SalRep
SET Value = ( SELECT ( MIN( Sal ) + MAX( Sal ) ) / 2 FROM Emp )
WHERE Specif = 'Average salary';

UPDATE Emp1
SET ( Job, Mgr, Sal, Comm, Deptno ) =
    ( SELECT Job, Mgr, Sal, Comm, Deptno
      FROM Emp1
      WHERE EmpNo = 7499 )
WHERE EmpNo = 7369;
```

DELETE statement

```
DELETE FROM table
[ WHERE … ];
```

```
DELETE FROM Emp1
WHERE EmpNo = 7698 OR Mgr = 7698;
```

ALTER TABLE

```
ALTER TABLE [user_name.] tbl_name
ADD
(
   column_name  data_type  [ DEFAULT value ][ col_constraint ],
   column_name  data_type  [ DEFAULT value ][ col_constraint ],
   . . .
   tbl_constraint,
   tbl_constraint
   . . .
);
```

```
ALTER TABLE [user_name.] tbl_name
MODIFY
(
   column_name  data_type  [ DEFAULT value ],
   column_name  data_type  [ DEFAULT value ],
   . . .
);
```

```
ALTER TABLE [user_name.] tbl_name
DROP
(
   column_name,
   column_name,
   . . .
);
```

```
ALTER TABLE [user_name.] tbl_name
DROP tbl_constraint;
```

(המשך) ALTER TABLE

ADD allows to add a new column(s) to the end of existing table or add a constraint to table's definitions

MODIFY changes an existing column with some restrictions:

- you may change the type of column or decrease its size **only** if every row for the column is NULL
- a NOT NULL column may be added **only** to a table with no rows
- an existing column can be modified to NOT NULL **only** if it has a non-NULL value in every row

DROP allows to drop an existing column or constraint

ALTER TABLE examples

```
ALTER TABLE Emp2
ADD
(
 Bonus NUMBER(7,2) CHECK( Bonus <> 666 ),
 BirthDate DATE
);
```

```
ALTER TABLE Emp2
DROP
(
 Bonus;
);
```

```
ALTER TABLE Emp2
MODIFY
(
 Bonus DEFAULT 0,
 Sal NUMBER(8,2)
);
```

DROP TABLE

DROP TABLE *[user_name.] tbl_name*
[**CASCADE CONSTRAINTS**];

Drops
- the table
- constraints, indexes and grants associated with it

The **CASCADE CONSTRAINTS** option drops all referential integrity constraints referring to keys in dropped table

```
SQL> DROP TABLE Emp1;
DROP TABLE Emp1
           *
ERROR at line 1:
ORA-02449: unique/primary keys in table referenced by foreign keys

SQL> DROP TABLE Emp1 CASCADE CONSTRAINTS;

Table dropped.
```

Subqueries תת שאילתות

Subquery is **SELECT** operator nested in other SQL statement

Subquery may be nested in:
- **SELECT, INSERT, UPDATE, DELETE** operators
- **CREATE TABLE** operator

CREATE TABLE *new_tbl* [(col_1, col_2, …, col_N)]
AS
SELECT … FROM *old_tbl* . . .**;**

```
SQL> CREATE TABLE Emp2
  2  AS
  3  SELECT * FROM Emp;

Table created.

SQL> CREATE TABLE Emp4( Tafkid, Kama )
  2  AS
  3  SELECT Job, COUNT( Job )
  4  FROM Emp
  5  GROUP BY Job;

Table created.
```

העתקת טבלה

הקמת טבלה חדשה

Subquery in SELECT statement

May be:

- in **WHERE** clause
- in **HAVING** clause

- Correlative
- Non-correlative

- Single-row (SR)
- Multiple-rows (MR)

```
SQL> SELECT DeptNo FROM Dept
  2  WHERE Loc = 'CHICAGO';

DEPTNO
------
    30

SQL> SELECT EmpNo, EName, Job, Sal
  2  FROM Emp
  3  WHERE DeptNo = 30;

EMPNO ENAME      JOB              SAL
----- ---------- --------- ---------
 7499 ALLEN      SALESMAN    1600.00
 7521 WARD       SALESMAN    1250.00
 7654 MARTIN     SALESMAN    1250.00
 7698 BLAKE      MANAGER     2850.00
 7844 TURNER     SALESMAN    1500.00
 7900 JAMES      CLERK        950.00

6 rows selected
```

Single-row subquery

```
SQL> SELECT EmpNo, EName, Job, Sal
  2  FROM Emp
  3  WHERE DeptNo = (SELECT DeptNo
  4                  FROM Dept
  5                  WHERE Loc = 'CHICAGO');

EMPNO ENAME      JOB              SAL
----- ---------- --------- ---------
 7499 ALLEN      SALESMAN    1600.00
 7521 WARD       SALESMAN    1250.00
 7654 MARTIN     SALESMAN    1250.00
 7698 BLAKE      MANAGER     2850.00
 7844 TURNER     SALESMAN    1500.00
 7900 JAMES      CLERK        950.00

6 rows selected
```

Multi-rows non-corellated subquery

```
SQL> SELECT Country, City
  2  FROM WLocation
  3  WHERE City IN (SELECT City FROM Weather
  4                 WHERE Condition = 'RAIN');

COUNTRY              CITY
-------------------- --------------------
UNITED STATES        CHICAGO
PERU                 LIMA
RUSSIA               MOSCOW
```

```
SQL> DESC Weahter
 Name          Null? Type
 ------------- ----- ------------
 CITY                VARCHAR2(12)
 TEMPERATURE         NUMBER(3)
 HUMIDITY            NUMBER(3)
 CONDITION           VARCHAR2(9)

SQL> DESC WLocation
 Name          Null? Type
 ------------- ----- ------------
 CITY                VARCHAR2(20)
 COUNTRY             VARCHAR2(20)
 CONTINENT           VARCHAR2(10)
 LATITUDE            NUMBER(5,2)
 NORTHSOUTH          CHAR(1)
 LONGITUDE           NUMBER(5,2)
 EASTWEST            CHAR(1)
```

Single-row corellated subquery

```
SQL> SELECT Country, City
  2  FROM WLocation A
  3  WHERE 'RAIN' = ( SELECT Condition
  4                   FROM Weather
  5                   WHERE City = A.City );

COUNTRY              CITY
-------------------- --------------------
UNITED STATES        CHICAGO
PERU                 LIMA
RUSSIA               MOSCOW
```

The same column ->
Correlated subquery

Join instead subquery

```
SQL> SELECT A.Country, A.City
  2  FROM WLocation A, Weather B
  3  WHERE B.City = A.City
  4    AND 'RAIN' = B.Condition

COUNTRY                CITY
---------------------- --------------------
UNITED STATES          CHICAGO
PERU                   LIMA
RUSSIA                 MOSCOW
```

רק תת שאילתה
יכולה לעזור !

```
SQL> SELECT Lodging, MAX( Age )
  2  FROM Worker
  3  GROUP BY Lodging;

LODGING            MAX(AGE)
-------------- ----------
CRANMER                25
MATTS                  35
MULLERS                32
PAPA KING              55
ROSE HILL              41
WEITBROCHT             43
                       67

7 rows selected.
```

```
SQL> SELECT W.Name, W.Age, W.Lodging
  2  FROM Worker W
  3  WHERE Age = ( SELECT MAX( Age )
  4                FROM Worker
  5                WHERE W.Lodging = Lodging );

NAME                          AGE LODGING
-------------------- ---------- --------------
ELBERT TALBOT                  43 WEITBROCHT
GEORGE OSCAR                   41 ROSE HILL
ROLAND BRANDT                  35 MATTS
PETER LAWSON                   25 CRANMER
VICTORIA LYNN                  32 MULLERS
GERHARDT KENTGEN               55 PAPA KING

6 rows selected.
```

```
SQL> SELECT Name, Age, Lodging
  2  FROM Worker
  3  WHERE ( Lodging, Age ) IN ( SELECT Lodging, MAX( Age )
  4                              FROM Worker
  5                              GROUP BY Lodging );

NAME                           AGE LODGING
------------------------- ---------- ---------------
PETER LAWSON                    25 CRANMER
ROLAND BRANDT                   35 MATTS
VICTORIA LYNN                   32 MULLERS
GERHARDT KENTGEN                55 PAPA KING
GEORGE OSCAR                    41 ROSE HILL
ELBERT TALBOT                   43 WEITBROCHT

6 rows selected.
```

תת שאילתה שמחזירה כמה עמודות בכמה שורות

```
SQL> SELECT W.Name, W.Lodging, S.Skill
  2  FROM Worker W, WorkerSkill S
  3  WHERE W.Name = S.Name
  4    AND S.Skill IN ( SELECT Skill FROM WorkerSkill
  5                     WHERE Name = 'JOHN PEARSON' )
  6    AND W.Name != 'JOHN PEARSON'
  7  ORDER BY W.Name;

NAME                      LODGING         SKILL
------------------------- --------------- -------------------------
DICK JONES                ROSE HILL       SMITHY
HELEN BRANDT                              COMBINE DRIVER
VICTORIA LYNN             MULLERS         SMITHY
```

מי יכול להחליף אותו ?

ANY, ALL, EXISTS clauses

Logical operators that test a Single Value

>, <, =, <>, >=, <=

EXISTS *(query)*
NOT EXISTS *(query)*

LIKE *expr*
NOT LIKE *expr*

expr **IS NULL**
expr **IS NOT NULL**

NOT *expr*
expr **AND** *expr*
expr **OR** *expr*

Logical operators that can test more than Single Value

IN (*expr [, expr] ...* | *query* **)**
NOT IN (*expr [, expr] ...* | *query* **)**

BETWEEN *expr* **AND** *expr*
NOT BETWEEN *expr* **AND** *expr*

operator **ANY (** *expr [, expr] ...* | *query* **)**
operator **ALL (** *expr [, expr] ...* | *query* **)**

query **UNION** *query*
query **UNION ALL** *query*
query **INTERSECT** *query*
query **MINUS** *query*

Condition	Takes place if...	Condition	Takes place if...
F = **ANY**(2, 5, 7)	F = 2 **OR** F = 5 **OR** F = 7	F = **ALL**(2, 5, 7)	*Always **FALSE**
F > **ANY**(2, 5, 7)	F > 2	F > **ALL**(2, 5, 7)	F > 7
F < **ANY**(2, 5, 7)	F < 7	F < **ALL**(2, 5, 7)	F < 2
F >= **ANY**(2, 5, 7)	F >= 2	F >= **ALL**(2, 5, 7)	F >= 7
F <= **ANY**(2, 5, 7)	F <= 7	F <= **ALL**(2, 5, 7)	F <= 2
F <> **ANY**(2, 5, 7)	*Always **TRUE**	F <> **ALL**(2, 5, 7)	F <> 2 **AND** F <> 5 **AND** F <> 7

```
SQL> SELECT Title_Id, Title, TO_CHAR( Advance, '$999,999.99' ) AS Advance
  2  FROM Titles
  3  WHERE Advance > ALL( SELECT B.Advance
  4                       FROM Publishers A, Titles B
  5                       WHERE A.Pub_Id = B.Pub_Id
  6                         AND Pub_Name = 'New Age Books' );

TITLE_ TITLE                                    ADVANCE
------ ------------------------------ -----------
MC3021 The Gourmet Microwave                $15,000.00
```

למצוא ספרים שמקדמה ששולמה עבורם גדולה מן המקדמה המקסימלית ששולמה ע"י הוצאה לאור "New Age Book"

Against NULL

```
SQL> SELECT Title_Id, Title, TO_CHAR( Advance, '$99,999.99' ) AS Advance
  2  FROM Titles
  3  WHERE Advance > ALL( SELECT NVL( B.Advance, 0 )
  4                       FROM Publishers A, Titles B
  5                       WHERE A.Pub_Id = B.Pub_Id
  6                         AND Pub_Name = 'Algodata Infosystems' );

TITLE_ TITLE                                    ADVANCE
------ ------------------------------ -----------
BU2075 You Can Combat Computer Stress!     $10,125.00
MC3021 The Gourmet Microwave                $15,000.00
```

```
SQL> SELECT Au_Id, Au_LName || ' ' || Au_FName AS Author, City
  2  FROM Authors
  3  WHERE City = ANY( SELECT City FROM Publishers )
  4  ORDER BY City;

AU_ID       AUTHOR              CITY
----------- ------------------- --------------------
409-56-7008 Bennet Abraham      Berkeley
238-95-7766 Carson Cheryl       Berkeley
```

למצוא את כל המחברים שמתגוררים בעיר שבה נמצאת הוצאה לאור כל שהי

```
SQL> COL Continent HEADING 'Continent'
SQL> COL City HEADING 'City,|Country' JUSTIFY CENTER
SQL> COL Condition HEADING 'Weather|conditions' JUSTIFY CENTER
SQL> SELECT INITCAP( B.Continent ) AS Continent,
  2         INITCAP( A.City || ', ' || B.Country ) AS City,
  3         A.Condition
  4  FROM Weather A, WLocation B
  5  WHERE A.City = B.City
  6    AND A.Temperature < ANY ( SELECT Temperature
  7                              FROM Weather
  8                              WHERE Humidity < 60 );

                         City,                  Weather
Continent              Country                conditions
---------- ---------------------------------- ---------
S.America  Lima, Peru                         RAIN
Australia  Sydney, Australia                  SNOW
```

מהם העובדים שיש להם יותר ממיומנות אחת ?

```
SQL> SELECT Name, Skill
  2  FROM WorkerSkill W
  3  WHERE EXISTS ( SELECT *
  4                 FROM WorkerSkill
  5                 WHERE W.Name = Name
  6                 GROUP BY Name
  7                 HAVING COUNT(Skill) > 1 );

NAME                      SKILL
------------------------- -------------------------
JOHN PEARSON              COMBINE DRIVER
JOHN PEARSON              SMITHY
JOHN PEARSON              WOODCUTTER
WILFRED LOWELL            WORK
WILFRED LOWELL            DISCUS
```

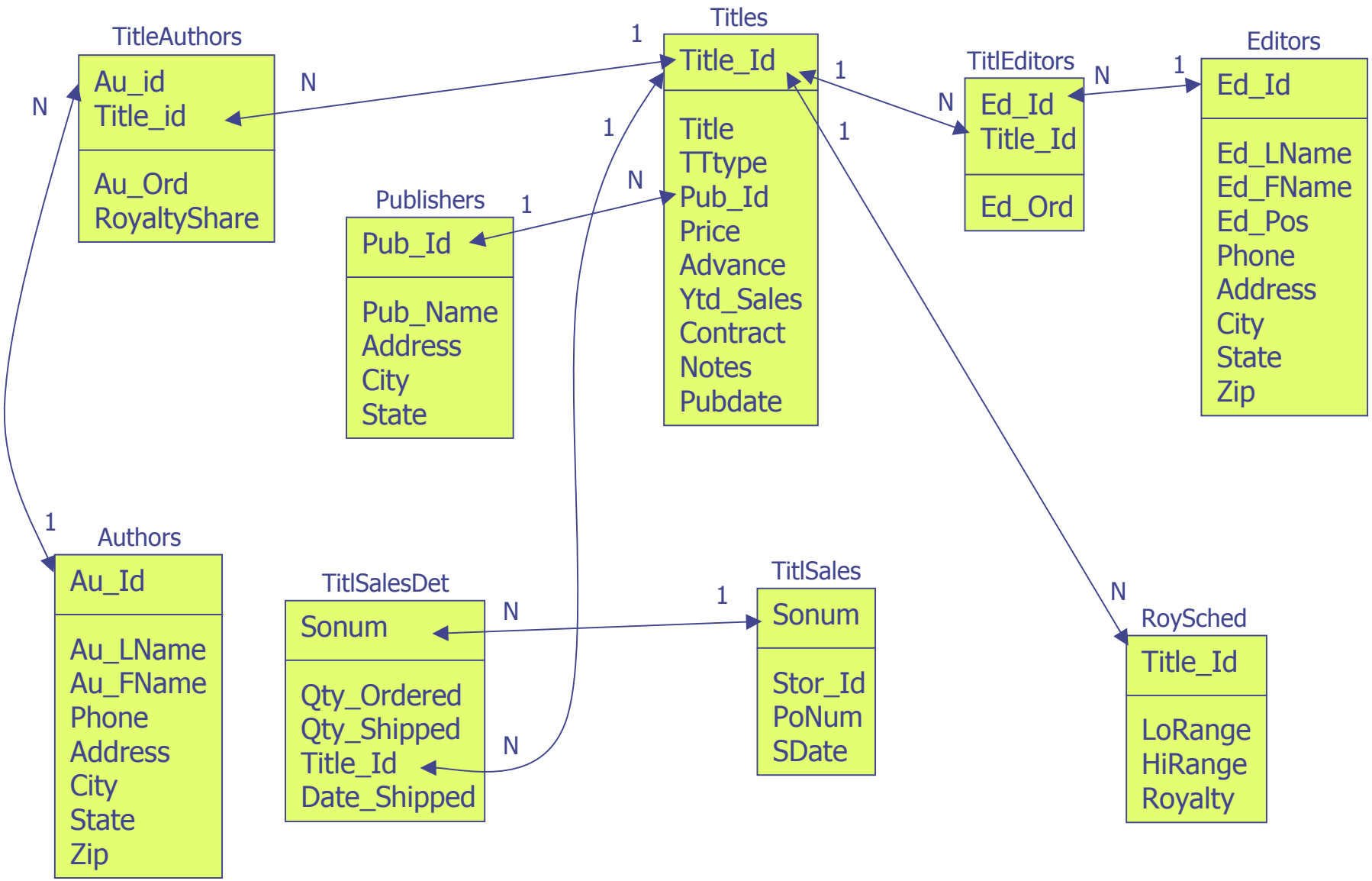
מהי המיומנות שאין עובדים ששולטים בה ?

```
SQL> SELECT S.Skill
  2  FROM Skill S
  3  WHERE NOT EXISTS ( SELECT *
  4                     FROM WorkerSkill W
  5                     WHERE W.Skill = S.Skill )

SKILL
-------------------------
GRAVE DIGGER
```

Subquery rules:

- The subquery must be enclosed in parentheses
- Subquery that produces only one row can be used with either single- or many-value operator
- Subquery that produces more than one row can be used **only** with many-value operator
- **IN** is equal to **=ANY**
- If one of the values of **ALL**-subquery is **NULL**, result of **WHERE** is **FALSE**
- If **ANY**-subquery is empty, result of **WHERE** is **FALSE**
- **EXISTS** test differs from **IN** in two ways:
  - it doesn't match a column or columns
  - it is typically used only with correlated subquery
- **BETWEEN** can't be used with subquery
- **ORDER BY** can't be used with subquery

BookBiz Data Base



BookBiz Data Base (comments)

שם השדה	משמעות
TitleAuthors.Au_Ord	מס' סידורי של המחבר ברשימת המחברים של הספר
TitleAuthors.RoyaltyShare	חלק של המחבר בתמלוגים של הספר (0 < RoyaltyShare <= 1)
Titles.TTtype	סוג הספר - מילת מפתח לחיפוש
Titles.Advance	הוצאות (מקדמה) עבור הספר
Titles.Ytd_Sales	מספר ספרים שנמכרו
Titles.Contract	שדה לוגי שמציג האם הסכם עם מחברי הספר כבר חתום
TitlEditors.Ed_Ord	מס' סידורי של העורך ברשימת העורכים של הספר
RoySched.LoRange	אם מתקיים תנאי: **LoRange** < Titles.Ytd_Sales <= **HiRange** אז חישוב תמלוגים מתבצע לפי נוסחה: Titles.Ytd_Sales * Title.Price * **Royalty**
RoySched.HiRange	
RoySched.Royalty	

VIEW תצפית

CREATE OR REPLACE VIEW *view_name* [(col_1, col_2, ..., col_N)]
AS
subquery
[WITH { READ ONLY | CHECK OPTION }];

```
SQL> CREATE TABLE Emp4( Tafkid, Kama )
  2  AS
  3  SELECT Job, COUNT( Job )
  4  FROM Emp1
  5  GROUP BY Job;

Table created.

SQL> SELECT * FROM Emp4;

TAFKID          KAMA
--------- ----------
ANALYST            2
CLERK              4
MANAGER            3
PRESIDENT          1
SALESMAN           4
```

```
SQL> CREATE VIEW Emp4V( Tafkid, Kama )
  2  AS
  3  SELECT Job, COUNT( Job )
  4  FROM Emp1
  5  GROUP BY Job;

View created.

SQL> SELECT * FROM Emp4V;

TAFKID          KAMA
--------- ----------
ANALYST            2
CLERK              4
MANAGER            3
PRESIDENT          1
SALESMAN           4
```

?

```
SQL> DELETE FROM Emp1
  2  WHERE EName LIKE 'A%';

2 rows deleted.
```

?

שם העמודה בתצפית

```
SQL> CREATE OR REPLACE VIEW Invasion
  2  AS
  3  SELECT L.Continent, L.Country, L.City, W.Condition
  4  FROM WLocation L, Weather W
  5  WHERE L.City = W.City;

View created.

SQL> SELECT * FROM Invasion;

CONTINENT  COUNTRY              CITY                 CONDITION
---------- -------------------- -------------------- ---------
EUROPE     GREECE               ATHENS               SUNNY
N.AMERICA  UNITED STATES        CHICAGO              RAIN
S.AMERICA  PERU                 LIMA                 RAIN
ASIA       INDIA                MADRAS               SUNNY
EUROPE     SPAIN                MADRID               SUNNY
EUROPE     ENGLAND              MANCHESTER           FOG
EUROPE     RUSSIA               MOSCOW               RAIN
EUROPE     FRANCE               PARIS                CLOUDY
EUROPE     GREECE               SPARTA               CLOUDY
AUSTRALIA  AUSTRALIA            SYDNEY               SNOW

10 rows selected.
```

, W.City ?

תצפית על תצפית


```
SQL> CREATE OR REPLACE VIEW InvEur
  2  AS
  3  SELECT Country, City, Condition
  4  FROM Invasion
  4  WHERE Continent = 'EUROPE'
  5  ORDER BY Country

View created.

SQL> SELECT * FROM InvEur;

COUNTRY              CITY                 CONDITION
-------------------- -------------------- ---------
ENGLAND              MANCHESTER           FOG
FRANCE               PARIS                CLOUDY
GREECE               ATHENS               SUNNY
GREECE               SPARTA               CLOUDY
RUSSIA               MOSCOW               RAIN
SPAIN                MADRID               SUNNY

6 rows selected.
```

צירוף תצפית עם טבלה

```
SQL> CREATE OR REPLACE VIEW DeptCount( DeptNo, EmpCount )
  2  AS
  3  SELECT DeptNo, COUNT( DeptNo )
  4  FROM Emp
  5  GROUP BY DeptNo

View created.

SQL> SELECT * FROM DeptCount;

    DEPTNO   EMPCOUNT
---------- ----------
        10          3
        20          5
        30          6

SQL> SELECT V.DeptNo, T.Loc, V.EmpCount
  2  FROM DeptCount V, Dept T
  3  WHERE V.DeptNo = T.DeptNo;

    DEPTNO LOC             EMPCOUNT
---------- ------------- ----------
        10 NEW YORK               3
        20 DALLAS                 5
        30 CHICAGO                6
```

צירוף תצפיות

```
SQL> CREATE OR REPLACE VIEW MonthTotal
  2  AS
  3  SELECT LAST_DAY( ActionDate ) AS Month,
  4         SUM( Amount ) AS Total
  5  FROM Ledger
  6  WHERE Action = 'PAID'
  7  GROUP BY LAST_DAY( ActionDate );

View created.

SQL> SELECT * FROM MonthTotal;

MONTH          TOTAL
--------- ----------
31-JAN-01         3.5
28-FEB-01         1.5
31-MAR-01           4
30-APR-01          10
31-MAY-01         3.4
30-JUN-01        4.25
31-JUL-01         5.2
31-AUG-01        11.7
30-SEP-01        4.25
31-OCT-01        4.93
30-NOV-01        4.25
31-DEC-01        2.25

12 rows selected.
```

```
SQL> CREATE OR REPLACE VIEW YearTotal
  2  AS
  3  SELECT SUM( Amount ) AS YTotal
  4  FROM Ledger
  5  WHERE Action = 'PAID';

View created.
```

```
SQL> COL Percent FORMAT 999.9
SQL> SELECT Month, Total,
  2         ( Total/YTotal ) *100 AS Prcnt
  3  FROM MonthTotal, YearTotal
  4  ORDER BY Month;

MONTH          TOTAL   PRCNT
--------- ---------- -------
31-JAN-01        3.5     5.9
28-FEB-01        1.5     2.5
31-MAR-01          4     6.8
30-APR-01         10    16.9
31-MAY-01        3.4     5.7
30-JUN-01       4.25     7.2
31-JUL-01        5.2     8.8
31-AUG-01       11.7    19.8
30-SEP-01       4.25     7.2
31-OCT-01       4.93     8.3
30-NOV-01       4.25     7.2
31-DEC-01       2.25     3.8
```

Rules of Views

It's possible to **INSERT** rows through view **only** if the view is based on a single table
It's impossible to **INSERT** rows through view if:

- its subquery contains:
  - ✓ The **GROUP BY** clause
  - ✓ The **DISTINCT** clause
  - ✓ Group functions
  - ✓ Any expressions
  - ✓ References to any pseudo-columns
- The underlying table has any **NOT NULL** column that doesn't appear in the view

Single table - no join
- no **UNION, MINUS, INTERSECT**

It's possible to **UPDATE** or **DELETE** rows in view **only** if the view is based on a single table
It's possible to **UPDATE** views containing pseudo-columns (other than **RowNum**) or expression, as long as they aren't referenced in the **UPDATE**
It's impossible to **UPDATE** or **DELETE** rows in view if:

- Its subquery contains:
  - ✓ The **GROUP BY** clause
  - ✓ The **DISTINCT** clause
  - ✓ Group functions
  - ✓ References to pseudo-column **RowNum**

It's impossible to **INSERT, UPDATE** or **DELETE** rows in view if the view is created **WITH READ ONLY**
WITH CHECK OPTION restricts **INSERT**s and **UPDATE**s performed through view to prevent them from creating rows that the view cannot itself select, based on its **WHERE** clause

שימוש בתצפיות

- אם לבטל טבלה שמתחת לתצפית, התצפית לא יכולה לענות לשאילתה.
- אם אחרי-כן לשחזר הטבלה אז התצפית חוזרת לתפקוד

- תצפית מאפשרת ללקוח להתרכז בנתונים הנדרשים ולהתעלם מסיבוכי השאילתה

```
SQL> CREATE OR REPLACE VIEW ForAdvrs
  2  AS
  3  SELECT T.Au_Id AS ID, A.Au_LName || ' ' || A.Au_FName AS Author,
  4         COUNT( T.Title_Id ) AS BookCnt
  5  FROM TitleAuthors T, Authors A
  6  WHERE T.Au_Id = A.Au_Id
  7    AND T.Au_Ord = 1
  8  GROUP BY T.Au_Id, A.Au_LName || ' ' || A.Au_FName
  9  HAVING Count( T.Title_Id ) > 1;

View created.

SQL> SELECT * FROM ForAdvrs;

ID          AUTHOR                                              BOOKCNT
----------- -------------------------------------------------- ----------
486-29-1786 Locksley Chastity                                           2
998-72-3567 Ringer Albert                                               2
```

שימוש בתצפיות (המשך)

- יותר פשוט להוסיף תנאים חדשים כי הרבה פעולות כמו צירופים, איחודים, חיתוכים כבר הוגדרו ולא נמצאים בזווית המבט (encapsulation)
- התאמה ללקוח ספציפי
- בטיחות
- אי תלות בשינויים בטבלאות

```
SQL> CREATE OR REPLACE VIEW RoyaltyChecks( Author, Total_Income )
  2  AS
  3  SELECT A.Au_LName || ' ' || A.Au_FName,
  4         SUM( T.Price * T.Ytd_Sales * R.Royalty * TA.RoyaltyShare )
  5  FROM Authors A, Titles T, TitleAuthors TA, RoySched R
  6  WHERE A.Au_Id = TA.Au_Id
  7    AND T.Title_Id = TA.Title_Id
  8    AND T.Title_Id = R.Title_Id
  9    AND T.Ytd_Sales BETWEEN R.LoRange AND R.HiRange
 10  GROUP BY A.Au_LName || ' ' || A.Au_FName

View created.
```

שימוש בתצפיות (המשך)

```
SQL> COL Total_Income FORMAT 9999999.99
SQL> SELECT * FROM RoyaltyChecks;

AUTHOR                                                TOTAL_INCOME
----------------------------------------------------- ------------
Bennet Abraham                                            4911.54
Blotchet-Halls Reginald                                  25255.61
Carson Cheryl                                            32240.16
DeFrance Michel                                           9977.33
Dull Ann                                                  4095.00
Green Marjorie                                           13350.54
Gringlesby Burt                                           1841.52
Hunter Sheryl                                             4095.00
Karsen Livia                                               607.22
Locksley Chastity                                         2665.46
MacFeather Stearns                                        2981.50
O'Leary Michael                                           3694.25
Panteley Sylvia                                            785.63
Ringer Albert                                             1421.27
Ringer Anne                                               4669.34
Straight Dick                                             8185.91
White Johnson                                             8139.93
Yokomoto Akiko                                            2455.36
del Castillo Innes                                        4874.36

19 rows selected.
```

שימוש בתצפיות (המשך)

```
SQL> CREATE OR REPLACE VIEW YellowPages( Name, Spec, City, State, Phone )
  2  AS
  3  SELECT Au_LName || ' ' || Au_FName, 'A', City, State, Phone
  4  FROM Authors
  5  UNION
  6  SELECT Ed_LName || ' ' || Ed_FName, 'E', City, State, Phone
  7  FROM Editors
  8  UNION
  9  SELECT Pub_Name, 'P', City, State, NULL
 10  FROM Publishers
 11  ORDER BY 1

View created.

SQL> COL Name FORMAT A30
SQL> COL State FORMAT A5
SQL> COL Spec FORMAT A4
SQL> SELECT * FROM YellowPages

NAME                           SPEC CITY                 STATE PHONE
------------------------------ ---- -------------------- ----- ------------
Algodata Infosystems           P    Berkeley             CA
Almond Alfred                  E    Chicago              IL    312 699-4177
Bennet Abraham                 A    Berkeley             CA    415 658-9932
Binnet and Hardley             P    Washington           DC
Blotchet-Halls Reginald        A    Corvallis            OR    503 745-6402
Carson Cheryl                  A    Berkeley             CA    415 548-7723
DeFrance Michel                A    Gary                 IN    219 547-9982
```

DeLongue Martinella	E	Berkeley	CA	415 843-2222
Dull Ann	A	Palo Alto	CA	415 836-7128
Green Marjorie	A	Oakland	CA	415 986-7020
Greene Morningstar	A	Nashville	TN	615 297-2723
Gringlesby Burt	A	Covelo	CA	707 938-6445
Himmel Eleanore	E	Boston	MA	617 423-0552
Hunter Amanda	E	Boston	MA	617 432-5586
Hunter Sheryl	A	Palo Alto	CA	415 836-7128
Karsen Livia	A	Oakland	CA	415 534-9219
Kaspchek Christof	E	Berkeley	CA	415 549-3909
Locksley Chastity	A	San Francisco	CA	415 585-4620
MacFeather Stearns	A	Oakland	CA	415 354-7128
McBadden Heather	A	Vacaville	CA	707 448-4982
McCann Dennis	E	Rockbill	MD	301 468-3909
New Age Books	P	Boston	MA	
O'Leary Michael	A	San Jose	CA	408 286-2428
Panteley Sylvia	A	Rockville	MD	301 946-8853
Ringer Albert	A	Salt Lake City	UT	801 826-0752
Ringer Anne	A	Salt Lake City	UT	801 826-0752
Rutherford-Hayes Hannah	E	Rockbill	MD	301 468-3909
Samuelson Bernard	E	Oakland	CA	415 843-6990
Smith Meander	A	Lawrence	KS	913 843-0462
Sparks Manfred	E	Denver	CO	303 721-3388
Straight Dick	A	Oakland	CA	415 834-2919
Stringer Dirk	A	Oakland	CA	415 843-2991
White Johnson	A	Menlo Park	CA	408 496-7223
Yokomoto Akiko	A	Walnut Creek	CA	415 935-4228
del Castillo Innes	A	Ann Arbor	MI	615 996-8275

35 rows selected.

INDEXES

CREATE [UNIQUE] INDEX *index_name* **ON** *table* (col_1, col_2, ..., col_N) **;**

DROP INDEX *index_name* **;**

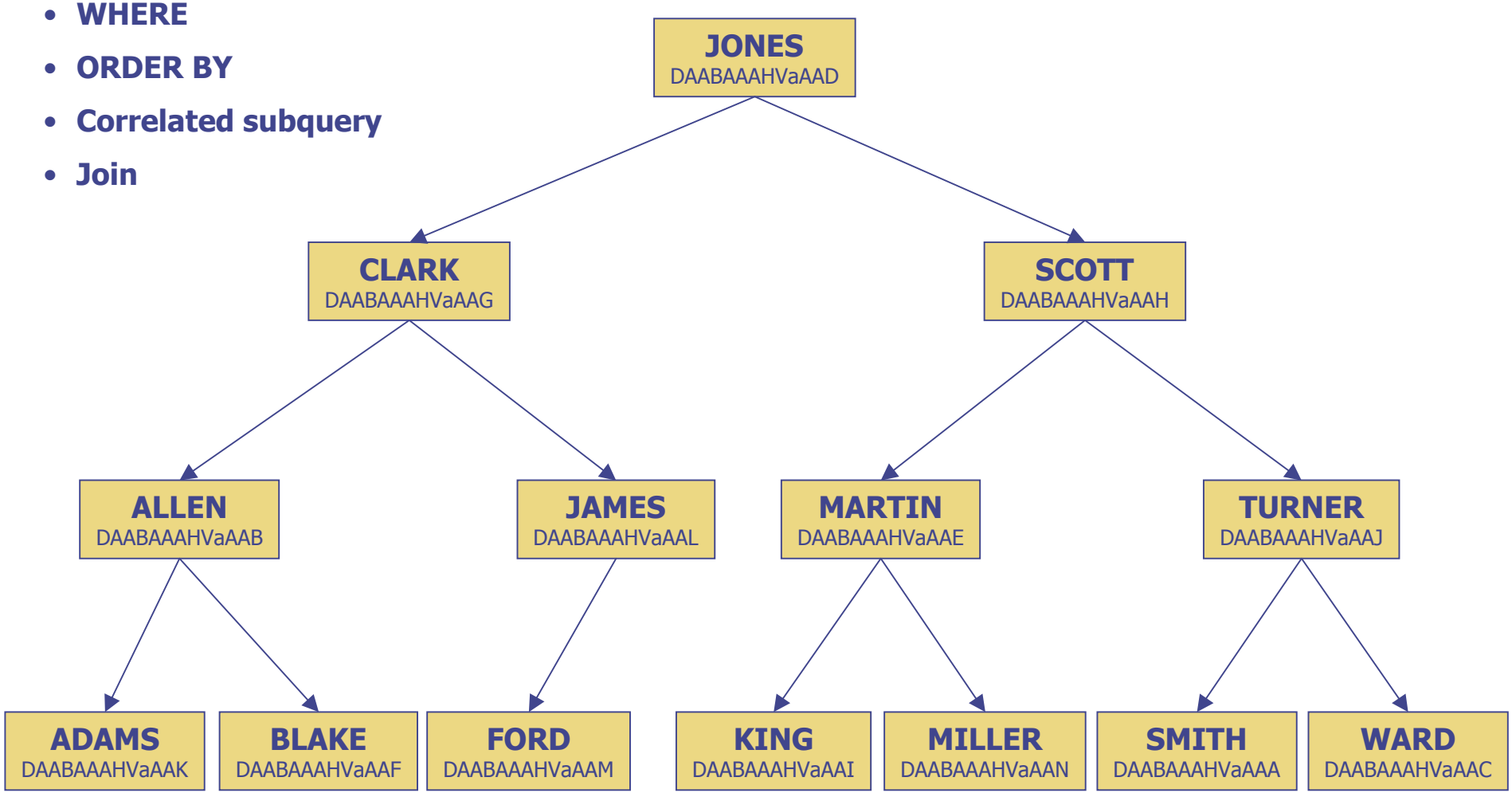
N <= 16

```
SQL> SELECT RowID, EmpNo, EName, Job, Sal
  2  FROM Emp;

ROWID                  EMPNO ENAME      JOB              SAL
------------------ ---------- ---------- --------- ----------
AAAFXDAABAAAHVaAAA       7369 SMITH      CLERK            800
AAAFXDAABAAAHVaAAB       7499 ALLEN      SALESMAN        1600
AAAFXDAABAAAHVaAAC       7521 WARD       SALESMAN        1250
AAAFXDAABAAAHVaAAD       7566 JONES      MANAGER         2975
AAAFXDAABAAAHVaAAE       7654 MARTIN     SALESMAN        1250
AAAFXDAABAAAHVaAAF       7698 BLAKE      MANAGER         2850
AAAFXDAABAAAHVaAAG       7782 CLARK      MANAGER         2450
AAAFXDAABAAAHVaAAH       7788 SCOTT      ANALYST         3000
AAAFXDAABAAAHVaAAI       7839 KING       PRESIDENT       5000
AAAFXDAABAAAHVaAAJ       7844 TURNER     SALESMAN        1500
AAAFXDAABAAAHVaAAK       7876 ADAMS      CLERK           1100
AAAFXDAABAAAHVaAAL       7900 JAMES      CLERK            950
AAAFXDAABAAAHVaAAM       7902 FORD       ANALYST         3000
AAAFXDAABAAAHVaAAN       7934 MILLER     CLERK           1300

14 rows selected.
```

(המשך) INDEXES

```
SQL> CREATE INDEX Emp_EName ON Emp( EName );

Index created.
```

- **WHERE**
- **ORDER BY**
- **Correlated subquery**
- **Join**



Data Dictionary מילון נתונים

Data dictionary stores all of the information that is used to manage the object s in the database

DICT	Data dictionary full content
ALL_USERS	Information about all users of the database
USER_ALL_TABLES	Description of all object and relational tables owned by the user's
USER_CATALOG	Tables, Views, Synonyms and Sequences owned by the user
USER_COL_COMMENTS	Comments on columns of user's tables and views
USER_CONSTRAINTS	Constraint definitions on user's own tables
USER_CONS_COLUMNS	Information about accessible columns in constraint definitions
USER_ERRORS	Current errors on stored objects owned by the user
USER_INDEXES	Description of the user's own indexes
USER_IND_COLUMNS	COLUMNs comprising user's INDEXes and INDEXes on user's TABLES
USER_SYS_PRIVS	System privileges granted to current user
USER_TABLES	Description of the user's own relational tables
USER_TAB_COLUMNS	Columns of user's tables, views and clusters
USER_TAB_COMMENTS	Comments on the tables and views owned by the user
USER_VIEWS	Description of the user's own views

Authority סמכויות

```
CREATE USER  user_name  IDENTIFIED { BY password  | EXTERNALLY } ;

ALTER USER  user_name  IDENTIFIED { BY password  | EXTERNALLY } ;
DROP USER  user_name [ CASCADE ];
```

Privilege is a permission granted to an Oracle user to execute some action

סמכויות לקוח

Privileges זכויות (הרשאות)

Roles תפקידים

System privileges

Object privileges

CREATE [**ANY**]...
ALTER [**ANY**]...
DROP [**ANY**]...
SELECT ANY TABLE
UPDATE ANY TABLE
INSERT ANY TABLE
GRANT ANY PRIVILEGE
GRANT ANY ROLE

Tables and views (user's own only)
- **INSERT**
- **UPDATE** (all or specific columns)
- **DELETE**

On tables only
- **ALTER** (all or specific columns)
- **REFERENCES**
- **INDEX**
- **ALL** (of the above)

On tables and views
- **SELECT**

תפקיד זהו חבילת הרשאות עם שם נקוב

3 תפקידים סטנדרטים:
CONNECT
RESOURCE
DBA

System privileges

GRANT {*system_privilege* | *role* } [, {*system_privilege* | *role* }, . . .]
TO { *user* | *role* } [, { *user* | *role* }, . . .] [**WITH ADMIN OPTION**];

REVOKE {*system_privilege* | *role* } [, {*system_privilege* | *role* }, . . .]
FROM { *user* | *role* } [, { *user* | *role* }, . . .];

```
SQL> CONNECT SYSTEM/MANAGER;
Connected.

SQL> CREATE USER BOB IDENTIFIED BY CRAZY;

User created.

SQL> CREATE USER BILL IDENTIFIED BY POLITE;

User created.

SQL> GRANT CONNECT, RESOURCE TO BOB, BILL;

Grant succeeded.

SQL> GRANT CREATE ANY TABLE,
  2        CREATE ANY INDEX,
  3        CREATE ANY VIEW TO BOB
  4  WITH ADMIN OPTION;

Grant succeeded.
```

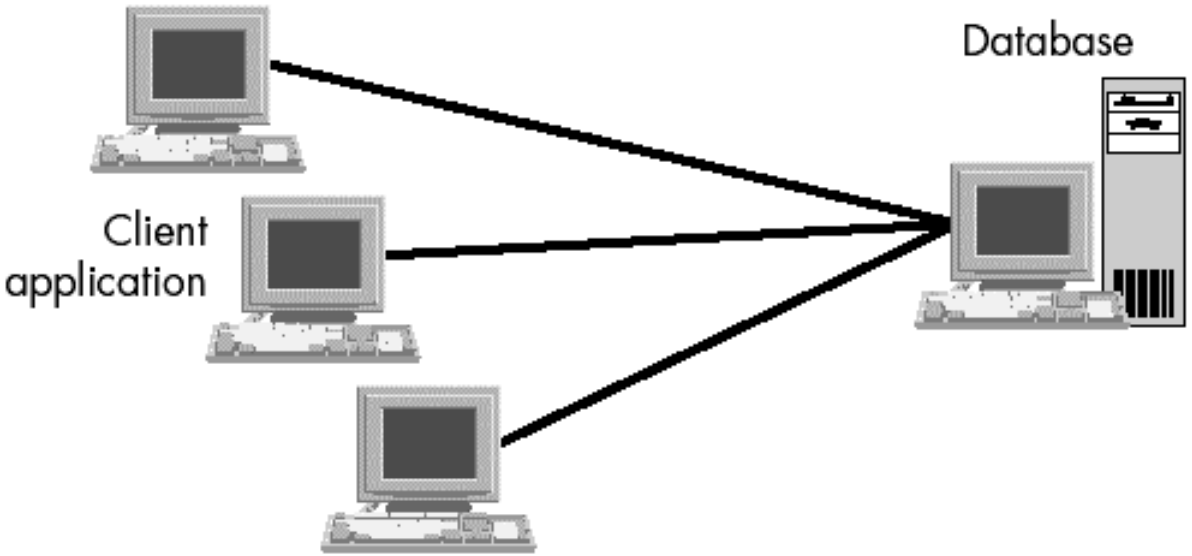
Object privileges

GRANT *object_privilege* [(*column* [, *column*, . . .])]
ON *object* **TO** { *user* | *role* } [**WITH GRANT OPTION**];

REVOKE *object_privilege* [, *object_privilege*, . . .]
ON *object* **FROM** { *user* | *role* } [, { *user* | *role* }, . . .];

```
SQL> CONNECT SCOTT/TIGER;
Connected.
SQL> GRANT SELECT ON EMP TO BILL;

Grant succeeded.

SQL> CONNECT BILL/POLITE;
Connected.

SQL> SELECT EMPNO, ENAME, HIREDATE, SAL, DEPTNO
  2  FROM SCOTT.EMP
  3  WHERE JOB = 'CLERK';

     EMPNO ENAME      HIREDATE          SAL     DEPTNO
---------- ---------- --------- ---------- ----------
      7369 SMITH      17-DEC-80        800         20
      7876 ADAMS      23-MAY-87       1100         20
      7900 JAMES      03-DEC-81        950         30
      7934 MILLER     23-JAN-82       1300         10
```

Client-server application



Multitier applications