

Σημειώσεις στο μάθημα Βάσεις Δεδομένων II

Handson 02

Πίνακας περιεχομένων

1	Γενικά	41.1
	Εισαγωγή	51.2
	ToHuman Resources (HR) Schema	72
	Δημιουργία Όψεων	72.1
	Τι είναι μια Όψη (view);	82.2
	Δημιουργία/Τροποποίηση Όψεων (views)	82.3
	Ανάκτηση Δεδομένων από μια Όψη	102.4
	DMLΕντολές σε μια Όψη	102.5
	Διαγραφή Όψης	112.6
	Inline Όψεις	112.7
	Top-n Ανάλυση	122.8
	Ασκήσεις	133
	Άλλα Αντικείμενα των ΒΔ	133.1
	Αντικείμενα ΒΔ	143.2
	Ακολουθίες (sequences)	143.3
	Ευρετήρια (indexes)	163.4
	Συνώνυμα (synonyms)	173.5
	Ασκήσεις	184
	Συναρτήσεις	184.1
	Γενικά	194.2
	Τύποι Συναρτήσεων	204.3
	Συναρτήσεις Μονής Γραμμής	204.3.1
	<i>Συναρτήσεις Χαρακτήρων</i>	214.3.2
	<i>Αριθμητικές Συναρτήσεις</i>	244.3.3
	<i>Χρήση Ημερομηνιών</i>	264.4
	Ασκήσεις	294.5
	Συναρτήσεις Μετασχηματισμού Τύπων Δεδομένων	314.5.1
	<i>Εσωτερικός Μετασχηματισμός Τύπων Δεδομένων</i>	324.5.2
	<i>Ρητός Μετασχηματισμός Τύπων Δεδομένων</i>	334.6
	Εμφωλευμένες Συναρτήσεις	384.7
	Συναρτήσεις Γενικού Σκοπού	384.8
	Υποθετικές Εκφράσεις στην SQL	414.9
	Ασκήσεις	43

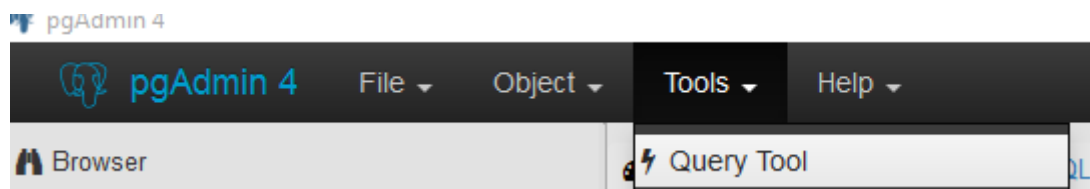
1 Γενικά

1.1 Εισαγωγή

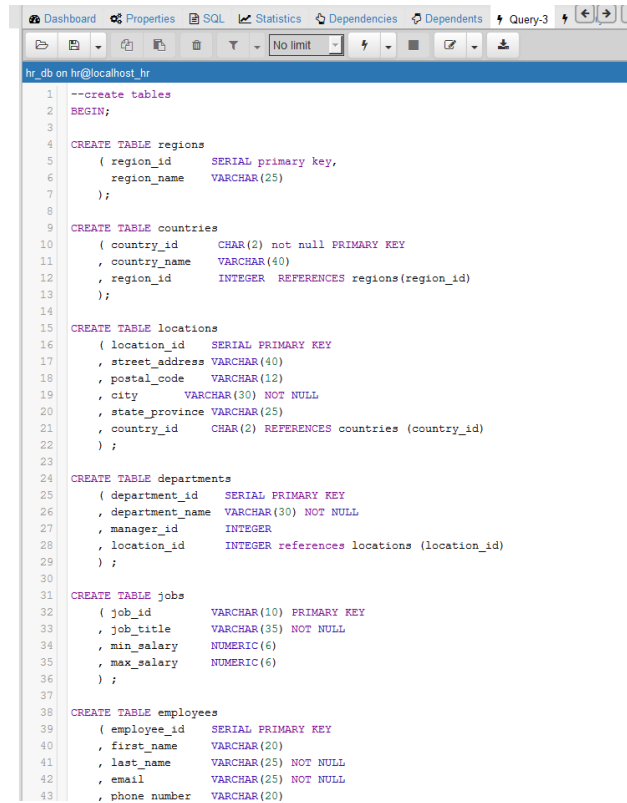
Σε αυτή την πρακτική εκπαίδευση θα συνεχίζουμε να εξετάζουμε πρακτικά θέματα των σχεσιακών βάσεων δεδομένων και διάφορες εντολές της SQL. Σκοπός είναι μετά την ολοκλήρωση της πρακτικής να έχετε μια πληρέστερη εικόνα ενός RDBMS καθώς και των δυνατοτήτων της SQL μέσα σε αυτό.

Στην περίπτωση

α. που έχετε επιλέξει το περιβάλλον του pgAdmin και αφού έχετε ολοκληρώσει τα προηγούμενα βήματα (postgresql_installation.01.oct2016, pgAdmin.01.oct2016), συνδέεστε στον server σαν χρήστης **hr** επιλέγετε την **HRProdDB** και επιλέγετε **Tools>QueryTool** οπότε ανοίγει ένα μικρό καρτέλα για την εκτέλεση εντολών σε SQL.




Στη συνέχεια ανοίγετε το αρχείο **HR_pgsql.sql** (με notepad) από το οποίο επιλέγετε **ΟΛΟ** το κείμενο και κάνετε αντιγραφή και επικόλληση μέσα στην καρτέλα SQL του pgAdmin που ανοίξατε προηγουμένως.



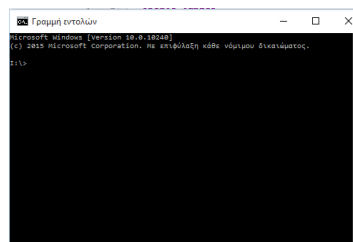
```

1  --create tables
2  BEGIN;
3
4  CREATE TABLE regions
5  (
6    region_id    SERIAL primary key,
7    region_name  VARCHAR(25)
8  );
9
10 CREATE TABLE countries
11 (
12   country_id    CHAR(2) not null PRIMARY KEY
13   , country_name VARCHAR(40)
14   , region_id    INTEGER REFERENCES regions(region_id)
15 );
16
17 CREATE TABLE locations
18 (
19   location_id    SERIAL PRIMARY KEY
20   , street_address VARCHAR(40)
21   , postal_code   VARCHAR(12)
22   , city          VARCHAR(30) NOT NULL
23   , state_province VARCHAR(25)
24   , country_id    CHAR(2) REFERENCES countries (country_id)
25 );
26
27 CREATE TABLE departments
28 (
29   department_id    SERIAL PRIMARY KEY
30   , department_name VARCHAR(30) NOT NULL
31   , manager_id      INTEGER
32   , location_id     INTEGER references locations (location_id)
33 );
34
35 CREATE TABLE jobs
36 (
37   job_id          VARCHAR(10) PRIMARY KEY
38   , job_title      VARCHAR(35) NOT NULL
39   , min_salary     NUMERIC(6)
40   , max_salary     NUMERIC(6)
41 );
42
43 CREATE TABLE employees
44 (
45   employee_id    SERIAL PRIMARY KEY
46   , first_name    VARCHAR(20)
47   , last_name     VARCHAR(25) NOT NULL
48   , email         VARCHAR(25) NOT NULL
49   , phone number  VARCHAR(20)

```

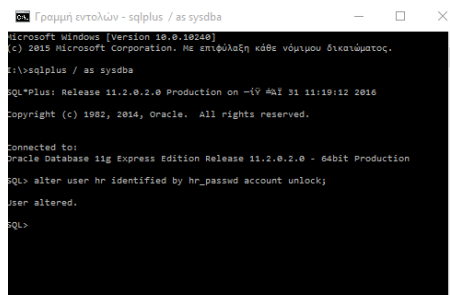
Πατάτε το εικονίδιο  και περιμένετε για να ολοκληρωθεί η εκτέλεση των εντολών. Μετά από αυτό είστε έτοιμοι να ξεκινήσετε.

β. που έχετε επιλέξει το περιβάλλον του SQLDeveloper και αφού έχετε ολοκληρώσει τα προηγούμενα βήματα (oracleexpress_installation, sqldeveloper) , ανοίγετε μια γραμμή εντολών των windows στην οποία δίνετε:



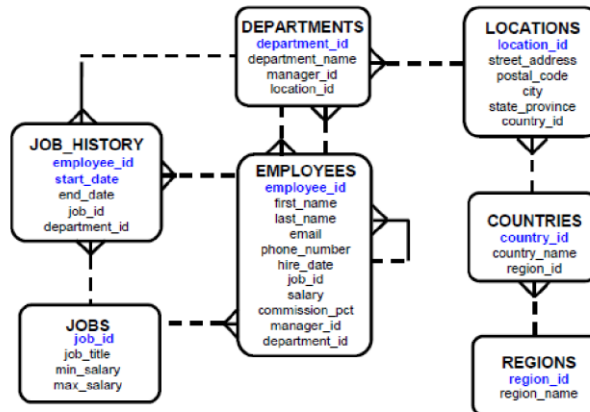
sqlplus / as sysdba

sql>alter user hr identified by hr_passwd account unlock;



Μετά από αυτό είστε έτοιμοι να ξεκινήσετε αφού δημιουργήσετε μια νέα σύνδεση για τον χρήστη hr.

1.2 ToHuman Resources (HR) Schema



Το HumanResources αποτελεί μέρος του sample schema της Oracle και μπορούμε να επιλέξουμε την δημιουργία του κατά την δημιουργία μιας ΒΔ. Οι πίνακες που περιλαμβάνει είναι οι εξής:

REGIONS: περιέχει εγγραφές για την αναπαράσταση μιας περιοχής όπως π.χ. Αμερική, Ασία κλπ

COUNTRIES: περιέχει εγγραφές για χώρες, από τις οποίες κάθε μια δείχνει στην αντίστοιχη περιοχή

LOCATIONS: περιέχει τις διευθύνσεις των γραφείων, αποθηκών της εταιρίας σε μια συγκεκριμένη χώρα

DEPARTMENTS: κρατά τα στοιχεία που αφορούν τα τμήματα της εταιρίας στα οποία εργάζονται οι εργαζόμενοι. Κάθε τμήμα μπορεί να σχετίζεται με τον πίνακα των εργαζομένων **EMPLOYEES** έτσι ώστε να καθορίζεται ο μάνατζερ του

EMPLOYEES: κρατά τις πληροφορίες που αφορούν του εργαζόμενους της εταιρίας οι οποίοι εργάζονται σε κάποιο τμήμα. Υπάρχει βέβαια και η περίπτωση κάποιοι εργαζόμενοι να μην έχουν σχετιστεί με τμήμα

JOBS: περιέχει τις κατηγορίες εργασιών που μπορεί να έχει ένα υπάλληλος στην εταιρία

JOB_HISTORY: περιέχει το ιστορικό των θέσεων-εργασιών των υπαλλήλων μέσα στην εταιρία. Έτσι στην περίπτωση που κάποιος υπάλληλος αλλάξει εργασία μέσα σε ένα τμήμα ή αλλάξει ένα τμήμα όχι όμως και εργασία, τότε δημιουργείται μια νέα εγγραφή στον πίνακα η οποία περιγράφει την προηγούμενη εργασία του

2 Δημιουργία Όψεων

2.1 Τι είναι μια Όψη (view);

Με την χρήση των views (όψεων) μπορούμε να προβάλουμε ένα λογικό υποσύνολο των δεδομένων ενός πίνακα ή ένα συνδυασμό δεδομένων από πίνακες. Μια view είναι ένας λογικός πίνακας βασιζόμενος σε έναν πίνακα ή σε μια άλλη view. Η view δεν περιέχει δικά της δεδομένα αλλά λειτουργεί σαν ένα “παράθυρο” μέσα από το οποίο μπορούμε να δούμε ή να μεταβάλουμε δεδομένα πινάκων. Ο ορισμός της view αποθηκεύεται σαν ένα ερώτημα *SELECT* μέσα στα μετα-δεδομένα του RDBMS

EMPLOYEES Table

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
100	Steven	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	17000
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000
104	Bruce	Skott	BSKOTT	590.423.5555	04-OCT-91	IT_PROG	6000
105	David	Turner	DTURNER	590.423.5555	07-DEC-91	IT_PROG	4200
106	Ellen	Abel	EABEL	590.423.5555	10-AUG-92	IT_PROG	5900
107	Jonathan	Greenberg	JGREEN	590.423.5555	03-JAN-93	IT_PROG	3500
108	Jack	Fisk	JFISK	590.423.5555	04-OCT-93	IT_PROG	3100
109	Greg	Clark	GCLARK	590.423.5555	07-DEC-93	IT_PROG	2500
110	Anna	Hartstein	AHARTSTE	590.423.5555	07-DEC-93	IT_PROG	2500
111	Parto	Serres	PSERRES	590.423.5555	07-DEC-93	IT_PROG	2500
112	Chen	Stein	CSTEIN	590.423.5555	07-DEC-93	IT_PROG	2500
113	Adam	Frey	AFREY	590.423.5555	07-DEC-93	IT_PROG	2500
114	Shelley	Stevens	SSTEVENS	590.423.5555	07-DEC-93	IT_PROG	2500
115	Walter	Tello	WTELLO	590.423.5555	07-DEC-93	IT_PROG	2500
116	Samuel	McCoy	SMCCOY	590.423.5555	07-DEC-93	IT_PROG	2500
117	Michael	Patton	MPATTON	590.423.5555	07-DEC-93	IT_PROG	2500
118	Javier	Renee	JRENEE	590.423.5555	07-DEC-93	IT_PROG	2500
119	Timothy	Gietz	TGIETZ	590.423.5555	07-DEC-93	IT_PROG	2500
120	Pat	Fay	PFAY	603.123.6666	17-AUG-97	MK_REP	6000
121	Shelley	Higgins	SHIGGINS	515.123.6080	07-JUN-94	AC_MGR	12000
122	William	Gietz	WGIETZ	515.123.8181	07-JUN-94	AC_ACCOUNT	8300

Με την χρήση των views μπορούμε:

α. να περιορίσουμε την πρόσβαση σε ευαίσθητα δεδομένα, επιλέγοντας αυτά να μην προβάλλονται

β. να περιορίσουμε την πολυπλοκότητα σύνθετων ερωτημάτων στον χρήστη δίνοντάς του την δυνατότητα να παίρνει τα δεδομένα που θέλει χρησιμοποιώντας τις views που χρειάζεται

γ. να ομαδοποιήσουμε τις απαιτήσεις χρηστών-εφαρμογών και να τις εξυπηρετήσουμε με την χρήση συγκεκριμένων views

Τις views μπορούμε να τις κατηγοριοποιήσουμε σε απλές και σε σύνθετες. Οι απλές λαμβάνουν τα δεδομένα από ένα πίνακα, δεν περιέχουν συναρτήσεις ή ομαδοποιήσεις στον ορισμό τους και υποστηρίζουν DML ενέργειες. Σε αντίθεση οι σύνθετες βασίζονται σε περισσότερους από ένα πίνακες, περιέχουν συναρτήσεις ή ομαδοποιήσεις δεδομένων στον ορισμό τους και συνήθως δεν υποστηρίζουν DML ενέργειες.

2.2 Δημιουργία/Τροποποίηση Όψεων (views)

Η σύνταξη για την δημιουργία μιας view φαίνεται στην παρακάτω εικόνα, όπου:


```
CREATE [ OR REPLACE ] VIEW
... [ owner. ]view-name [ ( column-name [ , ... ] ) ]
... AS select-without-order-by
... [ WITH CHECK OPTION ]
```

[ORREPLACE] δηλώνει ότι η view θα ξαναδημιουργηθεί εάν υπάρχει ήδη

VIEW δηλώνει το όνομα της view

column-name δηλώνει τα ονόματα των στηλών όπως θα εμφανίζονται στην view

AS select-without-order-by δηλώνει το ερώτημα που καθορίζει την view

[WITH CHECK OPTION] δηλώνει ότι μόνο εγγραφές που ορίζονται μέσα από την view μπορούν να εισαχθούν ή να τροποποιηθούν

Για παράδειγμα στην εικόνα που φαίνεται δημιουργούμε την view EMPVU80 η οποία περιέχει τους υπαλλήλους μόνο του τμήματος 80

```
CREATE VIEW empvu80
AS SELECT employee_id, last_name, salary
FROM employees
WHERE department_id = 80;
View created.
```

Η view αυτή θα έχει τρεις (3) στήλες: employee_id, last_name, salary. Μπορούμε να δούμε την δομή της δίνοντας την εντολή *DESCRIBE empvu80*; Στη περίπτωση όπου θέλουμε στη view που θα δημιουργήσουμε τα ονόματα των στηλών της να είναι διαφορετικά από αυτά που εμφανίζονται στο SELECT τότε μπορούμε είτε να χρησιμοποιήσουμε ψευδώνυμα μέσα στο *SELECT*, είτε να ορίσουμε τα ονόματα των στηλών της αμέσως μετά την δήλωση *CREATE VIEW* και πριν από το *SELECT*, όπως δείχνεται και στα παραδείγματα που ακολουθούν.

```
CREATE VIEW salvu50
AS SELECT employee_id ID_NUMBER, last_name NAME,
salary*12 ANN_SALARY
FROM employees
WHERE department_id = 50;
View created.
```

```
CREATE VIEW salvu50 (ID_NUMBER, NAME, ANN_SALARY)
AS SELECT employee_id, last_name, salary*12
FROM employees
WHERE department_id = 50;
View created.
```

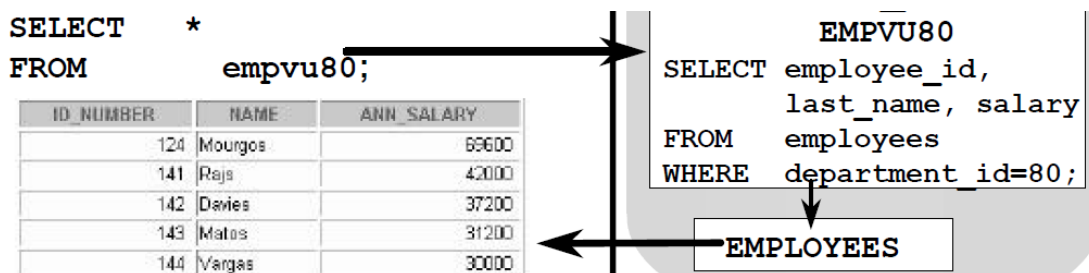
Εκτός όμως από τις απλές views όπως έχουμε πεί υπάρχουν και οι σύνθετες. Μια σύνθετη view δείχνεται παρακάτω, στην οποία υπολογίζονται ο ελάχιστος, ο μέγιστος και ο μέσος μισθός ανά τμήμα. Προσέξτε ότι στην δημιουργία της οι στήλες ορίζονται αμέσως μετά το *CREATE VIEW*.

```
CREATE VIEW dept_sum_vu
(name, minsal, maxsal, avgsal)
AS SELECT      d.department_name, MIN(e.salary) ,
              MAX(e.salary) ,AVG(e.salary)
FROM          employees e, departments d
WHERE         e.department_id = d.department_id
GROUP BY      d.department_name;
View created.
```

Για να τροποποιήσουμε μια όψη μπορούμε να χρησιμοποιήσουμε την επιλογή *ORREPLACE* κατά την οποία δημιουργείται ξανά η όψη ακόμη και αν υπάρχει μια άλλη με το ίδιο όνομα. Με τον τρόπο αυτό μια όψη μπορεί να τροποποιηθεί χωρίς να υπάρχει η ανάγκη να την σβήσουμε, να την δημιουργήσουμε και να δώσουμε ξανά τα κατάλληλα δικαιώματα στους χρήστες (θα δούμε τα δικαιώματα σε άλλη ενότητα).

2.3 Ανάκτηση Δεδομένων από μια Όψη

Μπορούμε να ανακτήσουμε δεδομένα από μια view με τον ίδιο ακριβώς τρόπο όπως και από έναν πίνακα. Μπορούμε να εμφανίσουμε το σύνολο των δεδομένων και στηλών ή να επιλέξουμε κάθε φορά τα επιθυμητά.



Στην πιο πάνω εικόνα φαίνεται η διαδικασία που εκτελείται κατά την χρήση μιας view. Η ΒΔ μεταφράζει το ερώτημα πάνω στην όψη στο αντίστοιχο ερώτημα με το οποίο αυτή έχει οριστεί.

2.4 DML Εντολές σε μια Όψη

Μέσα σε μια όψη μπορούμε να εκτελέσουμε DML εντολές (*INSERT*, *UPDATE*, *DELETE*) όταν ικανοποιούνται συγκεκριμένα κριτήρια. Έτσι μπορούμε να διαγράψουμε εγγραφές στην περίπτωση όπου η όψη δεν περιλαμβάνει εκφράσεις *GROUPBY*, *DISTINCT* και συναρτήσεις συνάθροισης. Μπορούμε να τροποποιήσουμε εγγραφές μόνο εάν δεν περιλαμβάνει συναρτήσεις συνάθροισης, εκφράσεις *GROUPBY*, *DISTINCT* και αν καμία από τις στήλες δεν προκύπτει μέσα από κάποιο υπολογισμό (π.χ. *SALARY*12*), ενώ μπορούμε να εισάγουμε δεδομένα μόνο όταν στον ορισμό της δεν υπάρχουν συναρτήσεις συνάθροισης, εκφράσεις *GROUPBY*, *DISTINCT*, στήλες που προκύπτουν από υπολογισμό ή πεδία *NOTNULL* των πινάκων της που όμως δεν προβάλλονται στην όψη.

Στην περίπτωση που θέλουμε οι εισαγωγές ή οι ενημερώσεις εγγραφών που εκτελούμε να περιορίζονται μόνο σε δεδομένα που προκύπτουν από το πεδίο ορισμού της π.χ. στην

EMPV20 μόνο για υπαλλήλους του τμήματος 20, εισάγουμε στον ορισμό της τον περιορισμό *WITHCHECKOPTION*.

```
CREATE VIEW empvu20
AS SELECT *
FROM employees
WHERE department_id = 20
WITH CHECK OPTION;
```

Έτσι μπορούμε να επεξεργαστούμε μόνο δεδομένα που προκύπτουν από το *SELECT* της όψης και οποιαδήποτε προσπάθεια σε δεδομένα εκτός αυτού του ορισμού αποτυγχάνει.

```
UPDATE empvu20
SET department_id = 10
WHERE employee_id = 201;
UPDATE empvu20
*
```

ERROR at line 1:
ORA-01402: view WITH CHECK OPTION where-clause violation

2.5 Διαγραφή Όψης

Για να διαγράψουμε μια όψη χρησιμοποιούμε την έκφραση *DROP VIEW view_name* και με αυτό τον τρόπο αφαιρούμε τον ορισμό της από την ΒΔ. Η ενέργεια αυτή δεν επηρεάζει τους πίνακες στους οποίους αναφερόταν η όψη, όμως επηρεάζει τυχόν άλλες όψεις που την περιλάμβαναν στους ορισμούς τους.

```
DROP VIEW empvu80;
View dropped.
```

2.6 Inline Όψεις

Μια inline όψη είναι ένα υποερώτημα εκφρασμένο με ψευδώνυμο το οποίο μπορεί να χρησιμοποιηθεί μέσα σε μια έκφραση SQL. Πιο συγκεκριμένα μπορούμε να δημιουργήσουμε μια inline όψη με ένα υποερώτημα στο FROM τμήμα της SQL. Με τον τρόπο αυτό η όψη αυτή μπορεί να χρησιμοποιηθεί να γίνει αναφορά σε αυτή από το κυρίως ερώτημα. Για να γίνει πιο κατανοητός ο ορισμός αυτός ας εξετάσουμε το πιο κάτω παράδειγμα.

```
SELECT a.last_name, a.salary, a.department_id, b.maxsal
FROM employees a, (SELECT department_id, max(salary) maxsal
FROM employees
GROUP BY department_id) b
WHERE a.department_id = b.department_id
AND a.salary < b.maxsal;
```

Εδώ η inlineόψη βεπιστρέφει το αναγνωριστικό των τμημάτων καθώς και το μέγιστο μισθό ανά τμήμα μέσα από τον πίνακα *EMPLOYEES*.

Στη συνέχεια με την χρήση του *WHERE a.department_id =*

b.department_id AND a.salary < b.salary στο κυρίως ερώτημα επιλέγουμε να προβάλλουμε τα ονόματα, τους μισθούς, το τμήμα και τον μέγιστο μισθό εκείνων των υπαλλήλων των οποίων ο μισθός είναι μικρότερος από τον μέγιστο μισθό του τμήματος που ανήκουν.

2.7 Top-n Ανάλυση

```
SELECT ROWNUM as SENIOR, E.last_name, E.hire_date
FROM (SELECT last_name, hire_date FROM employees
      ORDER BY hire_date) E
WHERE rownum <= 4;
```

Με top-πρωτήματα μπορούμε να επιλέξουμε τις n-μεγαλύτερες ή n-μικρότερες εγγραφές ενός πίνακα με βάση μια συνθήκη (στην συνέχεια αυτές να συνδυαστούν με άλλες για περαιτέρω επεξεργασία). Παραδείγματα τέτοιου είδους ερωτημάτων είναι π.χ. Η εύρεση των τριών μεγαλύτερων μισθών ή των τεσσάρων πιο πρόσφατων προσληφθέντων υπαλλήλων. Ένα τέτοιου είδους ερώτημα αποτελείται από τμήματα που δείχνονται πιο κάτω

```
SELECT [column_list], ROWNUM
FROM (SELECT [column_list]
      FROM table
      ORDER BY Top-N_column)
WHERE ROWNUM <= N;
```

- ένα υποερώτημα ή inlineόψη για την δημιουργία των ταξινομημένων αποτελεσμάτων. Το τμήμα αυτό πάντα περιέχει την έκφραση *ORDER BY* έτσι ώστε να διασφαλίζεται πάντα η σωστή βαθμονόμηση των εγγραφών. Για την εύρεση των μεγαλύτερων τιμών απαιτείται η χρήση του *DESC*
- ένα εξωτερικό ερώτημα το οποίο περιορίζει τον αριθμό των εγγραφών. Για να το πετύχουμε αυτό κάνουμε χρήση της ψευδοστήλης που υπάρχει ενσωματωμένη στα περισσότερα συστήματα ΒΔ (*rownum* για την Oracle, *row_number* για SQL Server κ.ο.κ.). Η ψευδοστήλη αυτή αποδίδει ένα σειριακό αριθμό ξεκινώντας από το 1 σε κάθε εγγραφή του υποερωτήματος. Χρησιμοποιώντας την ψευδοστήλη αυτή στο *WHERE* μπορούμε να περιορίσουμε τον αριθμό των εγγραφών σε n.

Με βάση την ανάλυση του ορισμού παρατηρώντας το αρχικό παράδειγμα της παραγράφου βλέπουμε ότι επιστρέφουμε το όνομα και την ημερομηνία πρόσληψης των τεσσάρων πιο πρόσφατα προσληφθέντων υπαλλήλων από τον πίνακα *EMPLOYEES*. Στο παράδειγμα μας επίσης βλέπουμε και την χρήση μιας inlineόψης για την παραγωγή του top-4 αποτελέσματος μας.

2.8 Ασκήσεις

1. Δημιουργήστε μια όψη με το όνομα *EMPLOYEES_VU* η οποία θα προβάλλει το αναγνωριστικό του υπαλλήλου, το όνομα του υπαλλήλου και τον αριθμό του τμήματος από τον πίνακα *EMPLOYEES*. Αλλάξτε τον τίτλο της στήλης *employee name* σε *EMPLOYEE*.
2. Επιλέξτε τα περιεχόμενα της όψης *EMPLOYEES_VU*.
3. Από την *EMPLOYEES_VU*, προβάλετε μόνο το όνομα του υπαλλήλου και τον αριθμό του τμήματος.
4. Δημιουργήστε μια όψη με όνομα *DEPT50* η οποία θα περιλαμβάνει το αναγνωριστικό του υπαλλήλου, το επώνυμο του υπαλλήλου και τον αριθμό του τμήματος για όλους τους υπαλλήλους του τμήματος 50. Ονομάστε τα ονόματα των στηλών της όψης *EMPNO*, *EMPLOYEE*, *DEPTNO*. Φροντίστε ώστε να μην δίνεται η δυνατότητα να αλλάξουν τμήμα οι υπάλληλοι μέσω της συγκεκριμένης όψης.
5. Προβάλετε την όψη της όψης *DEPT50*.
6. Προσπαθήστε να αλλάξετε τμήμα μέσω της όψης *DEPT50*.
7. Δημιουργήστε μία όψη *SALARY_VU* η οποία θα περιέχει το επώνυμο του υπαλλήλου, το όνομα του τμήματος, το μισθό και την κατηγορία του μισθού όλων των υπαλλήλων. Χρησιμοποιήστε τους πίνακες *EMPLOYEES*, *DEPARTMENTS* και *JOB_GRADES*. Ονομάστε τις στήλες *EMPLOYEE*, *DEPARTMENT*, *SALARY* και *GRADE*.

3 Άλλα Αντικείμενα των ΒΔ

3.1 Αντικείμενα ΒΔ

Εκτός από τους πίνακες και τις όψεις που έχουμε μελετήσει μέχρι τώρα, μια ΒΔ μπορεί να περιλαμβάνει επιπλέον πολλά περισσότερα αντικείμενα όπως είναι οι sequences (ακολουθίες), τα indexes (ευρετήρια) και τα synonyms (συνώνυμα).

3.2 Ακολουθίες (sequences)

Μια ακολουθία είναι ένα αντικείμενο που δημιουργούν οι χρήστες και το οποίο μπορεί να διαμοιραστεί μεταξύ χρηστών για την παραγωγή μοναδικών ακεραίων. Η συνηθέστερη χρήση των ακολουθιών είναι στην χρήση πρωτευόντων κλειδιών σε πίνακες. Οι αριθμοί των ακολουθιών αποθηκεύονται και δημιουργούνται ανεξάρτητα από τους πίνακες, και για τον λόγο αυτό η ίδια ακολουθία μπορεί να χρησιμοποιηθεί σε πολλαπλούς πίνακες. Η σύνταξη για την δημιουργία μιας ακολουθίας¹ έχει ως εξής:

```
CREATE SEQUENCE sequence
    [INCREMENT BY n]
    [START WITH n]
    [{MAXVALUE n | NOMAXVALUE}]
    [{MINVALUE n | NOMINVALUE}]
    [{CYCLE | NOCYCLE}]
    [{CACHE n | NOCACHE}] ;
```

στην οποία η έκφραση

<i>Sequence</i>	δηλώνει το όνομα της ακολουθίας
<i>INCRIMENT BY n</i>	δηλώνει το βήμα της ακολουθίας μεταξύ των αριθμών που παράγονται
<i>START WITH n</i>	δηλώνει την πρώτη τιμή της ακολουθίας (από εκεί δηλαδή που ξεκινά)
<i>MAXVALUE n</i>	δηλώνει την μέγιστη τιμή της ακολουθίας
<i>NOMAXVALUE</i>	δηλώνει ότι δεν έχει οριστεί μέγιστη τιμή για την ακολουθία
<i>MINVALUE n</i>	δηλώνει την ελάχιστη τιμή της ακολουθίας
<i>NOMINVALUE</i>	δηλώνει ότι δεν έχει οριστεί ελάχιστη τιμή για την ακολουθία
<i>CYCLE NOCYCLE</i>	δηλώνει ότι η ακολουθία θα συνεχίσει να παράγει τιμές (από την αρχή) και μετά την μέγιστη ή την ελάχιστη τιμή
<i>CACHE n NOCACHE</i>	δηλώνει ότι θα δεσμευτούν ή όχι τιμές στην μνήμη της ΒΔ

Στην πιο κάτω εικόνα δημιουργείται η ακολουθία *DEPT_DEPTID_SEQ* η οποία ξεκινά από τον αριθμό 120, έχει βήμα 10 και μέγιστη τιμή 9999.

¹ Εδώ η σύνταξη δημιουργίας της ακολουθίας βασίζεται στο RDBMS της Oracle. Ανάλογη είναι η σύνταξη και στα υπόλοιπα γνωστά συστήματα ΒΔ.

```
CREATE SEQUENCE dept_deptid_seq
    INCREMENT BY 10
    START WITH 120
    MAXVALUE 9999
    NOCACHE
    NOCYCLE;

Sequence created.
```

Για να μπορέσουμε να χρησιμοποιήσουμε την ακολουθία που χρησιμοποιήσαμε είναι απαραίτητο να μπορούμε να καλέσουμε την επόμενη παραγόμενη τιμή. Η σύνταξη της ενέργειας αυτής διαφέρει στα ΣΔΒΔ. Στην Oracle για παράδειγμα χρησιμοποιούμε την ψευδοστήλη *NEXTVAL* ως εξής *sequence.NEXTVAL*. Με τον τρόπο αυτό παράγεται μια νέα τιμή και η τρέχουσα τιμή είναι διαθέσιμη στην ψευδοστήλη *CURRVAL*. Με παρόμοιο τρόπο στον SQLServer² για να πάρουμε την επόμενη τιμή χρησιμοποιούμε την σύνταξη

```
CREATE SEQUENCE Test.CountBy1
    START WITH 1
    INCREMENT BY 1 ;

GO
```

```
SELECT NEXT VALUE FOR Test.CountBy1 AS FirstUse;
SELECT NEXT VALUE FOR Test.CountBy1 AS SecondUse;
```

Ένα ολοκληρωμένο παράδειγμα χρήσης ακολουθίας στην Oracle κατά την εισαγωγή μιας νέας εγγραφής δείχνεται παρακάτω.

```
INSERT INTO departments (department_id,
    department_name, location_id)
VALUES (dept_deptid_seq.NEXTVAL,
    'Support', 2500);

1 row created.
```

Εισάγεται μια νέα εγγραφή στον πίνακα *DEPARTMENTS* και στην θέση του *DEPARTMENT_ID* εισάγεται η τιμή που παράγεται από την ακολουθία *DEPT_DEPTID_SEQ*.

Για την διαγραφή μιας ακολουθίας υπάρχει η σύνταξη *DROP SEQUENCE*.

```
DROP SEQUENCE dept_deptid_seq;

Sequence dropped.
```

²Στην PostgreSQL η σύνταξη είναι *nextval(sequence_name)*, *currval(sequence_name)* κ.ο.κ.
<https://www.postgresql.org/docs/8.1/static/functions-sequence.html>

3.3 Ευρετήρια (indexes)

Όπως έχουμε δει και στην θεωρία τα ευρετήρια μπορούν να επιταχύνουν την ανάκτηση των εγγραφών χρησιμοποιώντας δείκτες. Σκοπός της χρήσης τους είναι να μειώσουν στο δίσκο το I/O που απαιτείται για την παραγωγή του αποτελέσματος. Τα ευρετήρια συντηρούνται από το εκάστοτε ΣΔΒΔ και δεν απαιτείται παρέμβαση από τον χρήστη, ενώ είναι φυσικά και λογικά ανεξάρτητα αντικείμενα με αποτέλεσμα να μπορούν να τροποποιηθούν ή και να διαγραφούν χωρίς να επηρεάζεται ο πίνακας για τον οποίον έχουν δημιουργηθεί.

Η σύνταξη για να δημιουργηθεί ένα ευρετήριο έχει ως εξής:

```
CREATE INDEX index  
ON table (column[, column]...);
```

Η πιο συγκεκριμένα για την βελτιστοποίηση του ερωτήματος στον πεδίο *LAST_NAME* του πίνακα *EMPLOYEES* φτιάχνεται ένα ευρετήριο *EMP_LAST_NAME_IDX*

```
CREATE INDEX emp_last_name_idx  
ON employees(last_name);  
Index created.
```

Η προσθήκη ενός ευρετηρίου βέβαια σε έναν πίνακα δεν σημαίνει αυτόματα και την βελτίωση του χρόνου απόκρισης του. Καθώς και άλλες εντολές εκτελούνται στις εγγραφές του πίνακα, είναι απαραίτητο να γίνονται και οι απαραίτητες ενημερώσεις σε ευρετήρια που τυχόν υπάρχουν. Όσο λοιπόν περισσότερα τα ευρετήρια σε έναν πίνακα τόσο και περισσότερες ενημερώσεις που πρέπει να γίνουν. Καταλήγουμε λοιπόν ότι δεν δημιουργούμε ευρετήρια αυθαίρετα χωρίς κανένα κριτήριο αλλά ακολουθώντας κάποιους πρακτικούς κανόνες, όπως:

- α. όταν μια στήλη περιέχει μεγάλο εύρος τιμών
- β. όταν μια στήλη περιέχει μεγάλο αριθμό *NULLs*
- γ. όταν μια ή περισσότερες στήλες χρησιμοποιούνται στην έκφραση του *WHERE* κατά την σύνδεση πινάκων
- δ. όταν οι πίνακες διαθέτουν μεγάλο αριθμό εγγραφών και περιμένουμε να ανακτούμε περίπου το 2-4% των τιμών

Με παρόμοια λογική εμπειρικοί κανόνες για την ΜΗ δημιουργία ευρετηρίων είναι:

- α. όταν ο πίνακας διαθέτει λίγες εγγραφές
- β. όταν δεν χρησιμοποιούνται οι στήλες συχνά σε κάποια συνθήκη
- γ. όταν ανακτούμε συνήθως μεγάλο αριθμό εγγραφών (>4%)

δ. όταν ο πίνακας ενημερώνεται πολύ συχνά

Στα ευρετήρια δεν δίνεται συνήθως η δυνατότητα να μεταβάλουμε τις παραμέτρους τους. Στην περίπτωση λοιπόν που θέλουμε να κάνουμε κάτι τέτοιο πρέπει να διαγράψουμε το ευρετήριο και να το ξαναδημιουργήσουμε. Αυτό γίνεται με την χρήση της εντολής

```
DROP INDEX index;
```

όπου *index* είναι το όνομα του ευρετηρίου π.χ. για να διαγράψουμε το ευρετήριο *UPPER_LAST_NAME_IDX* δίνουμε:

```
DROP INDEX upper_last_name_idx;  
Index dropped.
```

3.4 Συνώνυμα (synonyms)

Τα συνώνυμα είναι αντικείμενα των ΒΔ με τα οποία μας δίνεται η δυνατότητα να αναφερθούμε σε αρχικά αντικείμενα με ένα άλλο όνομα. Με την χρήση τους λοιπόν μπορούμε να προσθέσουμε ένα λογικό αφαιρετικό επίπεδο για την προστασία των εφαρμογών από τυχόν αλλαγές στο όνομα ή στην θέση των “πηγαίων” αντικείμενων. Επίσης μπορούμε να αναφερθούμε με πιο απλό τρόπο σε αντικείμενα που υπάρχουν σε άλλους χρήστες, χωρίς να είναι απαραίτητη η ύπαρξη προθέματος αλλά και δημιουργήσουμε μικρότερα ονόματα για την απόκρυψη άλλων μεγαλύτερων. Η σύνταξη για την δημιουργία ενός συνώνυμου είναι:

```
CREATE SYNONYM synonym FOR object;
```

όπου *synonym* το όνομα του συνώνυμου και *object* το αντικείμενο για το οποίο δημιουργείται το συνώνυμο. Στην παρακάτω εικόνα δημιουργούμε το συνώνυμο *D_SUMI* για την όψη *DEPT_SUM_VU* και στην συνέχεια το διαγράφουμε.

```
CREATE SYNONYM d_sum  
FROM dept_sum_vu;  
  
DROP SYNONYM d_sum;
```

Κατά την διαγραφή του συνωνύμου δεν επηρεάζεται το “πηγαίο” αντικείμενο. Μπορούμε να δημιουργήσουμε συνώνυμα για πίνακες, όψεις, ακολουθίες, αποθηκευμένες διαδικασίες κ.α.³

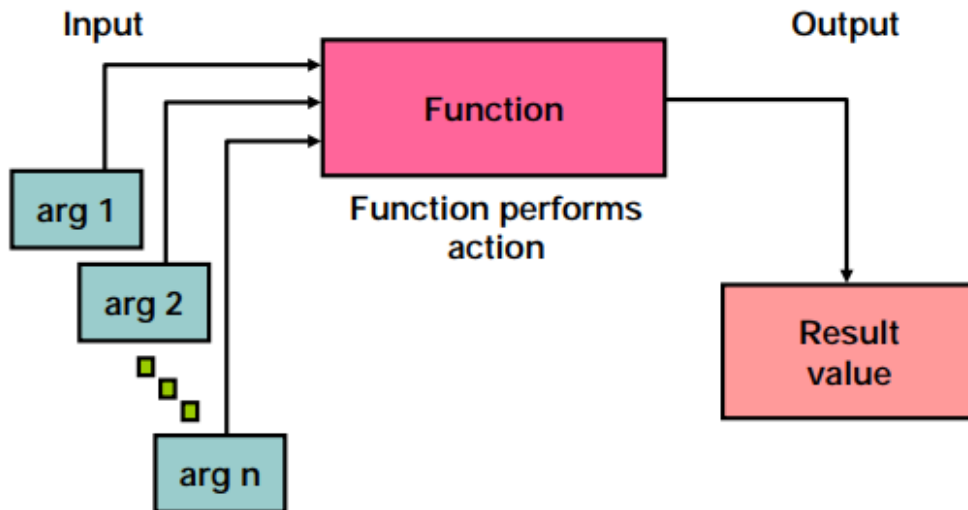
³ Το πλήθος και το είδος των αντικειμένων για τα οποία μπορούμε να δημιουργήσουμε συνώνυμα εξαρτάται κάθε φορά από το ΣΔΒΔ που εργαζόμαστε.

3.5 Ασκήσεις

1. Δημιουργήστε την ακολουθία *DEPT_ID_SEQ* η οποία θα ξεκινά από το 200 και θα έχει μέγιστη τιμή 1000. Η ακολουθία θα αυξάνεται κατά 10 αριθμούς κάθε φορά.
2. Πραγματοποιήστε μια εισαγωγή δύο εγγραφών στον πίνακα *DEPT* παράγοντας την τιμή για την στήλη *ID* από την ακολουθία που δημιουργήσατε στην Άσκηση 1. Προσθέστε με αυτό τον τρόπο δύο νέα τμήματα το “*Education*” και το “*Administration*”. Επιβεβαιώστε το αποτέλεσμα.
3. Δημιουργήστε ένα μη-μοναδικό ευρετήριο στη στήλη *DEPT_ID* του πίνακα *EMPLOYEES*.
4. Δημιουργήστε ένα συνώνυμο για τον πίνακα *EMPLOYEES* με το όνομα *EMP_SYN*. Χρησιμοποιήστε το συνώνυμο αυτό για επιλέξετε τον υπάλληλο με *ID*=1.

4 Συναρτήσεις

4.1 Γενικά

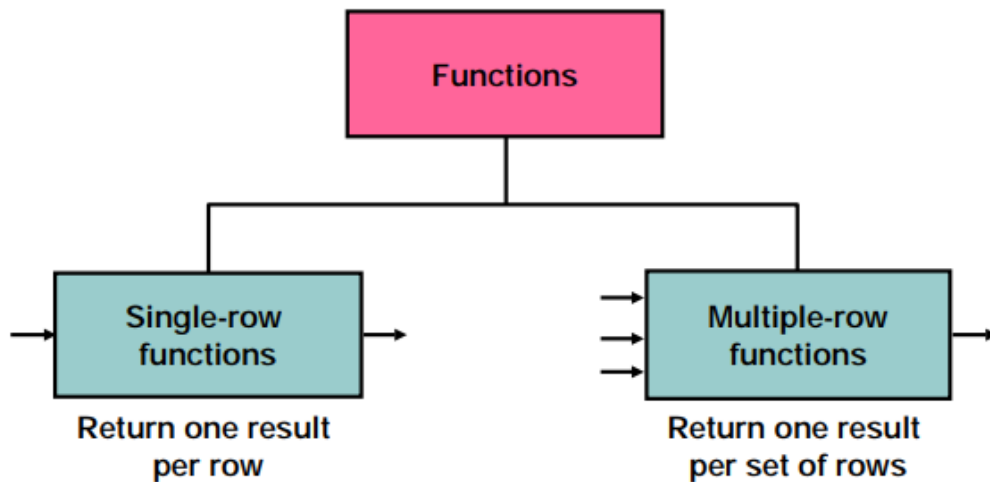


Οι συναρτήσεις είναι δομικά συστατικά της SQL που επεκτείνουν τις λειτουργίες της. Σε κάθε ΣΔΒΔ υπάρχουν ενσωματωμένες συναρτήσεις, ενώ υπάρχει η δυνατότητα να αναπτύξουμε και εμείς τις δικές μας. Με τις συναρτήσεις γενικά μπορούμε να:

- α. Πραγματοποιήσουμε υπολογισμούς πάνω στα δεδομένα
- β. Να μεταβάλουμε συγκεκριμένα τμήματα των δεδομένων
- γ. Να επεξεργαστούμε το αποτέλεσμα από σύνολα δεδομένων
- δ. Να μορφοποιήσουμε τα δεδομένα
- ε. Να μετατρέψουμε δεδομένα από ένα τύπο σε άλλο

κ.ο.κ.. Οι συναρτήσεις συνήθως δέχονται ορίσματα και πάντα επιστρέφουν μια τιμή.

4.2 Τύποι Συναρτήσεων



Υπάρχουν δύο τύποι συναρτήσεων:

- α. συναρτήσεις μονής γραμμής, οι οποίες ενεργούν σε μια μόνο εγγραφή και επιστρέφουν ένα αποτέλεσμα ανά γραμμή.
- β. συναρτήσεις πολλαπλών γραμμών, οι οποίες ενεργούν σε ένα σύνολο εγγραφών για να επιστρέψουν αποτέλεσμα βασιζόμενο στο σύνολο αυτό.

4.3 Συναρτήσεις Μονής Γραμμής

```
function_name [ (arg1, arg2, ...) ]
```

Οι συναρτήσεις μονής γραμμής χρησιμοποιούνται στην επεξεργασία τιμών. Μπορούν να δεχθούν ένα ή περισσότερα ορίσματα και επιστρέφουν μια τιμή για κάθε εγγραφή που επεξεργάζονται. Τα ορίσματα που δέχονται μπορεί να είναι: σταθερές τιμές, τιμές μεταβλητής, όνομα πεδίου ή μια έκφραση.

Συνοπτικά λοιπόν οι συναρτήσεις αυτές:

- α. δέχονται ένα ή περισσότερα ορίσματα
- β. επενεργούν σε μια εγγραφή κάθε φορά
- γ. επιστρέφουν μια τιμή ανά εγγραφή
- δ. το αποτέλεσμα που επιστρέφουν μπορεί να είναι σε διαφορετικό τύπο από αυτόν που επεξεργάζονται
- ε. μπορούν να είναι εμφωλευμένες

ζ. μπορούν να χρησιμοποιηθούν στο *SELECT*, στο *WHERE* ή στο *ORDERBY*

Τις συναρτήσεις μονής γραμμής μπορούμε να τις διακρίνουμε σε:

α. *συναρτήσεις χαρακτήρων*, οι οποίες δέχονται ως είσοδο χαρακτήρες και επιστρέφουν είτε χαρακτήρες είτε αριθμητικές τιμές

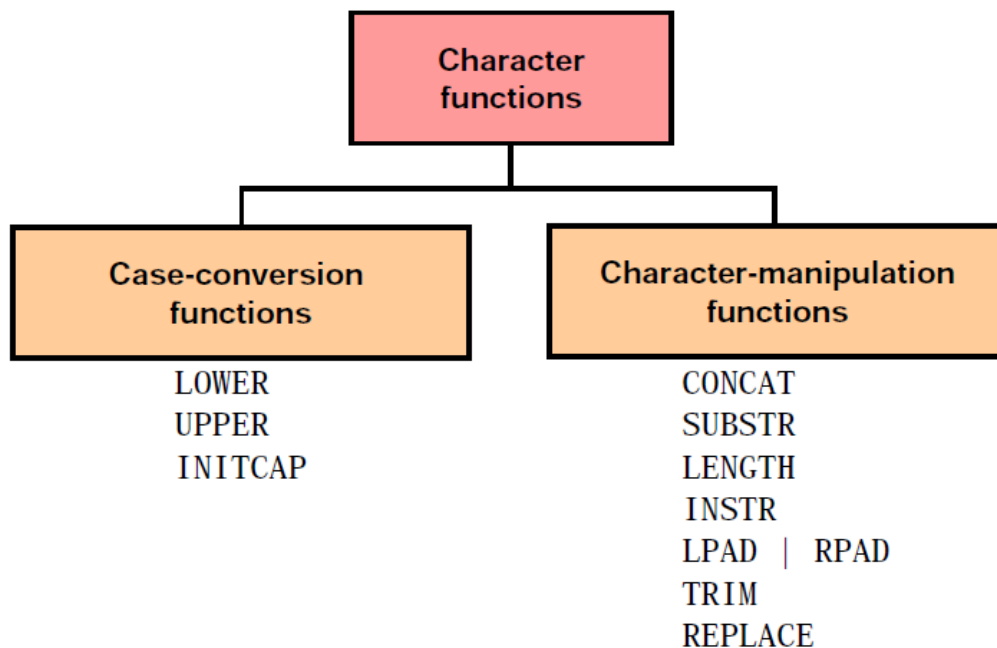
β. *αριθμητικές συναρτήσεις*, οι οποίες δέχονται ως είσοδο αριθμητικές τιμές και επιστρέφουν αριθμητικές τιμές

γ. *χρονικές συναρτήσεις*, οι οποίες επενεργούν σε χρονικά δεδομένα (τύπου DATE) και οι οποίες κυρίως επιστρέφουν χρονικά δεδομένα ως αποτέλεσμα

δ. *συναρτήσεις μετατροπών*, οι οποίες μετατρέπουν την τιμή εισόδου από ένα τύπο σε ένα άλλο τύπο

ε. *γενικές συναρτήσεις*, όπως είναι⁴ οι: *NVL*, *NVL2*, *NULLIF*, *COALESCE*, *CASE*, *DECODE*

4.3.1 Συναρτήσεις Χαρακτήρων



Οι συναρτήσεις χαρακτήρων με την σειρά τους μπορούν να διακριθούν: στις *συναρτήσεις μετατροπής* και στις *συναρτήσεις επεξεργασίας* των χαρακτήρων. Παραδείγματα συναρτήσεων μετατροπής χαρακτήρων αποτελούν η *LOWER*, η οποία μετατρέπει τους χαρακτήρες που δέχεται στην είσοδο της μικρά, η *UPPER*, η οποία μετατρέπει τους χαρακτήρες που δέχεται στην είσοδο της ΚΕΦΑΛΑΙΑ και η *INITCAP*, η οποία μετατρέπει το πρώτο γράμμα από κάθε λέξη σε ΚΕΦΑΛΑΙΟ και τα υπόλοιπα σε μικρά.

⁴ Με τα ονόματα αυτά τις συναντάμε στην Oracle, ενώ σε άλλα RDBMS υπάρχουν ίσως με διαφορετικά ονόματα.

```
SELECT 'The job id for ' || UPPER(last_name) || ' is '
      || LOWER(job_id) AS "EMPLOYEE DETAILS"
FROM   employees;
```

	EMPLOYEE DETAILS
1	The job id for ABEL is sa_rep
2	The job id for DAVIES is st_clerk
3	The job id for DE HAAN is ad_vp

Στις συναρτήσεις επεξεργασίας χαρακτήρων συναντάμε τις *CONCAT*, *SUBSTR*, *LENGTH*, *INSTR*, *RPAD* και *TRIM* κ.α.

α. *CONCAT*: ενώνει τιμές

β. *SUBSTR*: εξάγει ένα κείμενο ορισμένου μεγέθους

γ. *LENGTH*: επιστρέφει με αριθμητική τιμή το μέγεθος ενός κειμένου

δ. *INSTR*: βρίσκει την θέση (με αριθμητική τιμή) ενός χαρακτήρα

ε. *RPAD*, *LPAD*: εισάγουν επιλεγμένους (μέσω ορίσματος) χαρακτήρες έως ότου συμπληρωθεί το επιλεγμένο μέγεθος είτε δεξιά (R) είτε αριστερά (L) ενός κειμένου

ζ. *TRIM*: αφαιρεί επιλεγμένους χαρακτήρες από την αρχή και το τέλος ενός κειμένου

Σαν παράδειγμα χρήσης δείτε την παρακάτω εικόνα όπου ζητείται το αναγνωριστικό του υπαλλήλου και του τμήματος στο οποίο εργάζεται για τον υπάλληλο *Higgins*

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  last_name = 'higgins';
```

0 rows selected

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  LOWER(last_name) = 'higgins';
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	205 Higgins	110

Καθώς τα επίθετα είναι αποθηκευμένα με το πρώτο γράμμα τους σε κεφαλαίο το πρώτο ερώτημα αποτυγχάνει και για τον λόγο αυτό στο δεύτερο ερώτημα εφαρμόζουμε την συνάρτηση *LOWER* στην εφαρμογή του κατηγορήματος.

Ακολουθούν παραδείγματα εφαρμογής και άλλων συναρτήσεων.

Function	Result
CONCAT('Hello', 'World')	HelloWorld
SUBSTR('HelloWorld', 1, 5)	Hello
LENGTH('HelloWorld')	10
INSTR('HelloWorld', 'W')	6
LPAD(salary, 10, '*')	*****24000
RPAD(salary, 10, '*')	24000*****
REPLACE('JACK and JUE', 'J', 'BL')	BLACK and BLUE
TRIM('H' FROM 'HelloWorld')	elloWorld

Μπορείτε να τις δοκιμάσετε στην περίπτωση όπου δεν χρησιμοποιείτε κάποια στήλη ως όρισμα, αλλά σταθερές τιμές γράφοντας:

```
select concat('Hello','World') from dual;
```

Εδώ γίνεται η χρήση του ψευδοπίνακα *DUAL* της Oracle που στην ουσία αναφέρεται στην μνήμη του συστήματος και επιστρέφει πάντα μια στήλη και μια γραμμή. Στην περίπτωση τώρα που το όρισμα της συνάρτησης είναι η στήλη ενός πίνακα π.χ. του *EMPLOYEES* γράφουμε

```
select LPAD(salary, 10, '*') from employees;
```

Ένα άλλο παράδειγμα χρήσης είναι το παρακάτω ερώτημα στο οποίο εμφανίζεται το όνομα, το επίθετο, το αναγνωριστικό της εργασίας του υπαλλήλου καθώς και το μέγεθος σε χαρακτήρες του ονόματος του και η θέση του χαρακτήρα 'α' μέσα στο επίθετο του. Αυτό βέβαια μόνο για εκείνους του υπαλλήλους οι οποίοι περιέχουν το κείμενο 'REP' μέσα στο *JOB_ID* στη θέση 4.

```

SELECT employee_id, CONCAT(first_name, last_name) NAME,
       job_id, LENGTH (last_name),
       INSTR(last_name, 'a') "Contains 'a'?"
FROM   employees
WHERE  SUBSTR(job_id, 4) = 'REP';
    
```

	EMPLOYEE_ID	NAME	JOB_ID	LENGTH(LAST_NAME)	Contains 'a'?
1	174	EllenAbel	SA_REP	4	0
2	176	JonathonTaylor	SA_REP	6	2
3	178	KimberelyGrant	SA_REP	5	3
4	202	PatFay	MK_REP	3	2

Τέλος να επισημάνουμε ότι επειδή είναι πρακτικά αδύνατο να γνωρίζουμε τις παραμέτρους που αναμένει στην είσοδο της η κάθε συνάρτησης, για την πληροφόρησή μας καλό είναι να ανατρέχουμε στο εγχειρίδιο του κατασκευαστή. Αυτό συμβαίνει διότι η υλοποίηση είναι πολύ πιθανό να διαφέρει μεταξύ διαφορετικών κατασκευαστών.

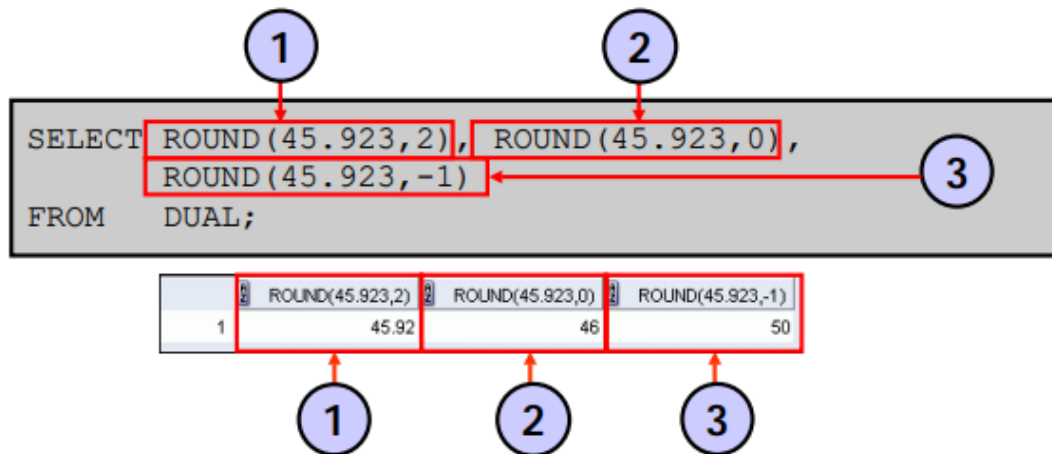
4.3.2 Αριθμητικές Συναρτήσεις

Function	Result
ROUND (45.926, 2)	45.93
TRUNC (45.926, 2)	45.92
MOD (1600, 300)	100

ΠΡΟΣΟΧΗ: στην πιο πάνω εικόνα η απεικόνιση των αριθμών γίνεται σύμφωνα με το αγγλικό σύστημα όπου ο διαχωρισμός των χιλιάδων γίνεται με κόμμα (,).

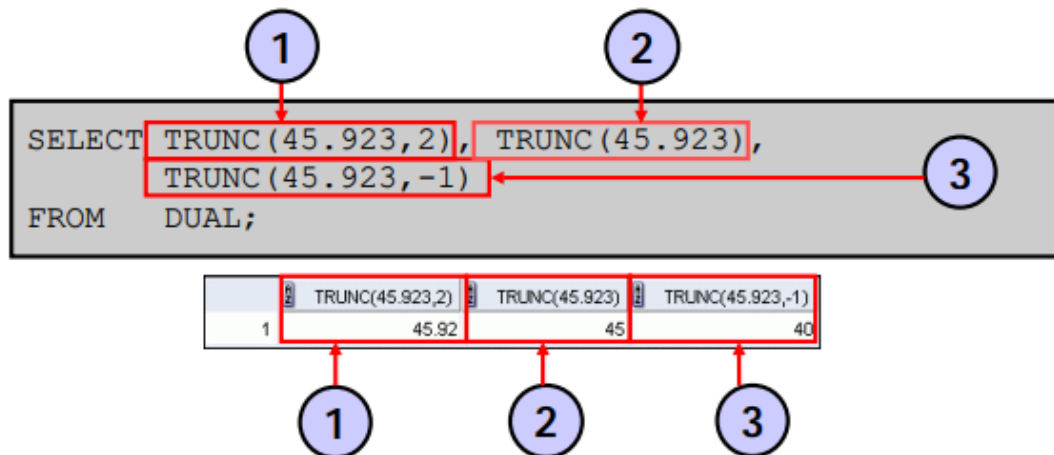
Οι αριθμητικές συναρτήσεις δέχονται στην είσοδο τους αριθμητικές τιμές και επιστρέφουν επίσης αριθμητικές τιμές. Παράδειγμα τέτοιων συναρτήσεων είναι οι *ROUND*, *TRUNC*, *MOD*. Πιο συγκεκριμένα:

Η συνάρτηση *ROUND* στρογγυλοποιεί την έκφραση ή την τιμή σε *n* ψηφία. Στην περίπτωση που το δεύτερο όρισμα, δηλαδή το *n*, της συνάρτησης λείπει ή είναι 0 τότε η τιμή στρογγυλοποιείται με μηδενικό δεκαδικό μέρος. Παρόμοια αν το δεύτερο όρισμα έχει π.χ. τιμή 2 τότε η τιμή στρογγυλοποιείται στα δύο (2) δεκαδικά ψηφία ενώ αν είναι -2 τότε η στρογγυλοποίηση γίνεται πάλι στα δύο (2) ψηφία όμως στο ακέραιο μέρος (βλ. παρακάτω εικόνα).



Η

συνάρτηση *TRUNC* περικόπτει το πεδίο, την έκφραση ή την τιμή σε η δεκαδικά ψηφία και δέχεται παρόμοια ορίσματα όπως η *ROUND*. Η διαφορά στις δύο συναρτήσεις είναι ότι ενώ η *ROUND* βασίζεται στους κανόνες της στρογγυλοποίησης η *TRUNC* απλά περικόπτει την τιμή.



Μια άλλη συνάρτηση είναι η *MOD* η οποία υπολογίζει το υπόλοιπο της διαίρεσης του πρώτου ορίσματος με το δεύτερο. Στο πιο κάτω παράδειγμα υπολογίζεται το υπόλοιπο της διαίρεσης του μισθού με 5.000 για όλους τους υπαλλήλους που έχουν *JOB_ID=SA_REP*. Μια άλλη χρήση της είναι ο έλεγχος εάν ένας αριθμός είναι μονός ή ζυγός.

4.3.3 Χρήση Ημερομηνιών

```
SELECT last_name, hire_date
FROM employees
WHERE hire_date < '01-FEB-88';
```

	LAST_NAME	HIRE_DATE
1	King	17-JUN-87
2	Whalen	17-SEP-87

Η Oracle αποθηκεύει τις ημερομηνίες στην εξήξεσωτερική αριθμητική μορφή: αιώνας, έτος, μήνας, ημέρα, ώρες, λεπτά και δευτερόλεπτα. Η προκαθορισμένη μορφοποίηση είναι DD-MON-RR, η οποία παράγει το αποτέλεσμα όπως δείχνεται στην πιο πάνω εικόνα (χωρίς να εμφανίζει την πληροφορία της ώρας και του αιώνα).

Η Oracle έχει ενσωματωμένη την συνάρτηση `SYSDATE` η οποία επιστρέφει την ημερομηνία και την ώρα στην οποία έχει ρυθμιστεί ο `database server`.

```
1 select sysdate from dual;
```

SYSDATE

18-OCT-18

Καθώς η εσωτερική αναπαράσταση των ημερομηνιών είναι ως αριθμός πάνω σε αυτές μπορούν να γίνουν αριθμητικές πράξεις όπως προσθέσεις, αφαιρέσεις αριθμών ή ημερομηνιών. Ποιο συγκεκριμένα οι πράξεις που μπορούν να εκτελεστούν είναι:

Operation	Result	Description
date + number	Date	Adds a number of days to a date
date - number	Date	Subtracts a number of days from a date
date - date	Number of days	Subtracts one date from another
date + number/24	Date	Adds a number of hours to a date

Ένα παράδειγμα πράξης δείχνεται παρακάτω

```
SELECT last_name, (SYSDATE-hire_date)/7 AS WEEKS
FROM employees
WHERE department_id = 90;
```

	LAST_NAME	WEEKS
1	King	1041.168239087301587301587301587302
2	Kochhar	923.025381944444444444444444444444
3	De Haan	750.168239087301587301587301587302

όπου εμφανίζονται το επώνυμο και αριθμός εβδομάδων πρόσληψης των υπαλλήλων του τμήματος 90. Για να παραχθεί το ζητούμενο αποτέλεσμα αφαιρείται η ημερομηνία πρόσληψης από την τρέχουσα ημερομηνία και στην συνέχεια διαιρείται με το 7 ώστε ο υπολογισμός να γίνει σε εβδομάδες.

Εκτός όμως από τις πράξεις μεταξύ ημερομηνιών και αριθμών, πράξεις μπορούν να γίνουν και με την χρήση ήδη υλοποιημένων συναρτήσεων της Oracle⁵. Όλες οι συναρτήσεις επιστρέφουν ως τύπο ημερομηνία εκτός από την συνάρτηση *MONTHS_BETWEEN* η οποία επιστρέφει αριθμητική τιμή.

Function	Result
MONTHS_BETWEEN	Number of months between two dates
ADD_MONTHS	Add calendar months to date
NEXT_DAY	Next day of the date specified
LAST_DAY	Last day of the month
ROUND	Round date
TRUNC	Truncate date

Ποιο συγκεκριμένα η:

- *MONTHS_BETWEEN(data1, date2)*: βρίσκει τον αριθμό των μηνών μεταξύ της *date1* και της *date2*. Το αποτέλεσμα μπορεί να είναι είτε αρνητικό είτε θετικό. Εάν η *date1* είναι μεταγενέστερη της *date2* το αποτέλεσμα είναι θετικό, στην αντίθετη περίπτωση αρνητικό. Το μη ακέραιο μέρος του αποτελέσματος αναπαριστά τμήμα του μήνα.
- *ADD_MONTHS(date,n)*: προσθέτει αριθμό μηνών στην *date*. Ο αριθμός *n* μπορεί να πρέπει να είναι ακέραιος και μπορεί να είναι και αρνητικός.
- *NEXT_DAY(date, 'char')*: βρίσκει την ημερομηνία μετά από την ημέρα '*char*' που ακολουθεί μετά από την ημερομηνία, *date*, που έχει δοθεί ως όρισμα. Η τιμή του ορίσματος '*char*' μπορεί να είναι αριθμός ο οποίος αναπαριστά μια ημέρα ή το λεκτικό της ημέρας.

⁵ Παρόμοιες συναρτήσεις υπάρχουν και σε άλλα ΣΔΒΔ

- **LAST_DATE(date)**: Βρίσκει την ημερομηνία του μήνα που ορίζεται από το όρισμα *date*. Συνήθως τα ΣΔΒΔ διαθέτουν ένα μεγάλο αριθμό ενσωματωμένων συναρτήσεων και αυτές που δείχνονται εδώ αποτελούν ένα μικρό υποσύνολο τους. Για την επεξεργασία ημερομηνιών μπορούν επίσης να χρησιμοποιηθούν οι συναρτήσεις **ROUND** και **TRUNC**.

- **ROUND(date, 'fmt')**: επιστρέφει την ημερομηνία *date* στρογγυλοποιημένη σύμφωνα με το όρισμα *'fmt'*. Στην περίπτωση που το όρισμα παραληφθεί η ημερομηνία στρογγυλοποιείται στην πλησιέστερη ημέρα.
- **TRUNC(date, 'fmt')**: επιστρέφει την ημερομηνία (συμπεριλαμβανομένης και της ώρας) αποκομμένη σύμφωνα με το όρισμα *'fmt'*. Στην περίπτωση που το όρισμα παραληφθεί η ημερομηνία αποκόπτεται στην πλησιέστερη ημέρα.

Στο παρακάτω παράδειγμα υπολογίζονται το αναγνωριστικό του υπαλλήλου, η ημερομηνία πρόσληψης του, αριθμός μηνών εργασίας του, η ημερομηνία αξιολόγησης μετά από 6 μήνες από την ημερομηνία πρόσληψης, η πρώτη Παρασκευή μετά την ημερομηνία πρόσληψης και η τελευταία ημέρα του μήνα κατά τον οποίο έγινε η πρόσληψη. Αυτό για εκείνους του υπαλλήλους που εργάζονται για λιγότερο από 100 μήνες.

```
SELECT employee_id, hire_date,
       MONTHS_BETWEEN (SYSDATE, hire_date) TENURE,
       ADD_MONTHS (hire_date, 6) REVIEW,
       NEXT_DAY (hire_date, 'FRIDAY'), LAST_DAY(hire_date)
FROM   employees
WHERE  MONTHS_BETWEEN (SYSDATE, hire_date) < 100;
```

R	EMPLOYEE_ID	HIRE_DATE	R	TENURE	REVIEW	NEXT_DAY(HIRE_DATE,'FRIDAY')	LAST_DAY(HIRE_DATE)
1	124	16-NOV-99	91.1099600...	16-MAY-00	19-NOV-99	30-NOV-99	
2	149	29-JAN-00	88.6906052...	29-JUL-00	04-FEB-00	31-JAN-00	
3	178	24-MAY-99	96.8518955...	24-NOV-99	28-MAY-99	31-MAY-99	
4	99999	07-JUN-99	96.4002626...	07-DEC-99	11-JUN-99	30-JUN-99	
5	113	11-JUN-07	0.25824335...	11-DEC-07	15-JUN-07	30-JUN-07	

Ένα ακόμη παράδειγμα δείχνει την χρήση των **ROUND**, **TRUNC** με την χρήση ημερομηνιών. Εάν η στρογγυλοποίηση γίνεται σε μήνα τότε η ημέρες 1-15 έχουν ως αποτέλεσμα την πρώτη ημέρα του ίδιου μήνα, ενώ ημέρες 16-31 έχουν ως αποτέλεσμα την πρώτη ημέρα του επόμενου μήνα. Με παρόμοιο τρόπο αν η στρογγυλοποίηση γίνεται σε έτος, μήνες μεταξύ 1-6 έχουν ως αποτέλεσμα 1 Ιανουαρίου του ίδιου έτους, ενώ 7-12, 1 Ιανουαρίου του επόμενου έτους.

```
SELECT employee_id, hire_date,
       ROUND(hire_date, 'MONTH'), TRUNC(hire_date, 'MONTH')
FROM   employees
WHERE  hire_date LIKE '%97';
```

R	EMPLOYEE_ID	HIRE_DATE	ROUND(HIRE_DATE,'MONTH')	TRUNC(HIRE_DATE,'MONTH')
1	142	29-JAN-97	01-FEB-97	01-JAN-97
2	202	17-AUG-97	01-SEP-97	01-AUG-97

4.4 Ασκήσεις

1. Γράψτε ένα ερώτημα το οποίο θα εμφανίζει την ημερομηνία του συστήματος

SYSDATE
10-OCT-18

2. Το τμήμα HR χρειάζεται μια αναφορά στην οποία θα εμφανίζεται το αναγνωριστικό του υπαλλήλου, το επώνυμο του, ο μισθός του και ο μισθός του αυξημένος κατά 15,5% και στρογγυλοποιημένος σε ακέραιο αριθμό (ονομάστε το πεδίο αυτό New Salary). Σώστε το ερώτημα σας σαν *lab_04_02.sql*.

	EMPLOYEE_ID	LAST_NAME	SALARY	New Salary
1	100	King	24000	27720
2	101	Kochhar	17000	19635
3	102	De Haan	17000	19635
4	103	Hunold	9000	10395
5	104	Ernst	6000	6930
6	107	Lorentz	4200	4851
7	124	Mourgos	5800	6699
8	141	Rajs	3500	4043
9	142	Davies	3100	3581
10	143	Matos	2600	3003

...

19	205	Higgins	12000	13860
20	206	Gietz	8300	9587

3. Τροποποιήστε το ερώτημα *lab_04_02.sql* ώστε να υπολογίσετε μια νέα στήλη η οποία θα αφαιρεί τον αρχικό μισθό από τον νέο. Ονομάστε την νέα αυτή στήλη *Increase*. Σώστε το ερώτημα σας σαν *lab_04_03.sql*.

	EMPLOYEE_ID	LAST_NAME	SALARY	New Salary	Increase
1	100	King	24000	27720	3720
2	101	Kochhar	17000	19635	2635
3	102	De Haan	17000	19635	2635
4	103	Hunold	9000	10395	1395
5	104	Ernst	6000	6930	930

...

20	206	Gietz	8300	9587	1287
----	-----	-------	------	------	------

4. Γράψτε ένα ερώτημα στο οποίο θα εμφανίζεται το επώνυμο (με τον πρώτο γράμμα κεφαλαίο και όλα τα υπόλοιπα μικρά) καθώς και το μήκος του επιθέτου για εκείνους τους υπαλλήλους των οποίων το όνομα αρχίζει από τα γράμματα "J", "A" ή "M". Δώστε κατάλληλες ονομασίες στις στήλες.

	A Z	Name	A Z	Length
1		Abel		4
2		Matos		5
3		Mourgos		7

5. Το τμήμα HR θέλει να γνωρίζει την χρονικό διάστημα το οποίο κάθε εργαζόμενος εργάζεται στην εταιρία. Θα πρέπει να εμφανίζεται το επίθετο του και θα πρέπει να γίνει υπολογισμός των μηνών μεταξύ της σημερινής ημερομηνίας και της ημερομηνίας κατά την οποία έγινε η πρόσληψη του. Ο τίτλος αυτής της στήλης θα είναι *MONTHS_WORKED* και η ταξινόμηση των αποτελεσμάτων θα είναι κατά αύξουσα σειρά ως προς τους υπολογιζόμενους μήνες. Να γίνει στρογγυλοποίηση του *MONTHS_WORKED* στον πλησιέστερο ακέραιο αριθμό.

	A Z	LAST_NAME	A Z	MONTHS_WORKED
1		Zlotkey		88
2		Mourgos		90
3		Grant		96
4		Lorentz		100
5		Vargas		107
6		Taylor		110
7		Matos		111
8		Fay		117
9		Davies		124
10		Abel		133
11		Hartstein		135
12		Rajs		139
13		Higgins		156
14		Gietz		156
15		De Haan		173
16		Ernst		192
17		Hunold		209
18		Kochhar		212
19		Whalen		236
20		King		239

6. Δημιουργήστε ερώτημα στο οποίο θα εμφανίζεται το επώνυμο και ο μισθός των υπαλλήλων. Η μορφοποίηση του μισθού θα πρέπει να είναι τέτοια ώστε να μας δίνεται ένα αλφαριθμητικό μήκους 15 χαρακτήρων. Για τον λόγο αυτό να γίνει συμπλήρωση του πεδίου από αριστερά με το σύμβολο \$. Ονομάστε το πεδίο *SALARY*

	LAST_NAME	SALARY
1	King	\$\$\$\$\$\$\$\$\$24000
2	Kochhar	\$\$\$\$\$\$\$\$\$17000

...

20	Gietz	\$\$\$\$\$\$\$\$\$8300
----	-------	------------------------

7. Δημιουργήστε ένα ερώτημα στο οποίο θα εμφανίζονται οι πρώτοι 8 χαρακτήρες του επιθέτου των υπαλλήλων σε συνδυασμό με το μισθό τους σε αναπαράσταση αστερίσκων. Για την ακρίβεια κάθε αστερίσκος θα αντιστοιχεί και σε ένα ψηφίο του μισθού. Να γίνει αντίστροφη ταξινόμηση ως προς τον μισθό. Το όνομα του πεδίου είναι *EMPLOYEES_AND_THEIR_SALARIES*.

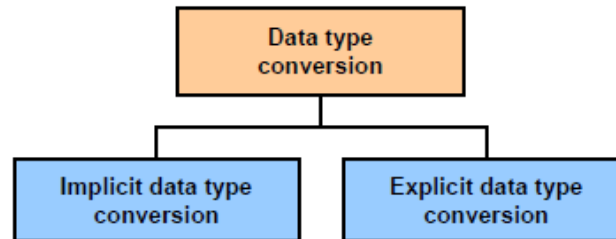
	EMPLOYEES_AND_THEIR_SALARIES
1	King *****
2	Kochhar *****
3	De Haan *****
4	Hartstei *****
5	Higgins *****
...	
19	Matos **
20	Vargas **

4.5 Συναρτήσεις Μετασχηματισμού Τύπων Δεδομένων

Οι συναρτήσεις μετασχηματισμών μπορούν να πραγματοποιήσουν μετατροπή από ένα τύπο δεδομένων σε άλλο χωρίς απώλεια πληροφορίας. Η SQL περιέχει διάφορους τέτοιους τύπους και σε αυτή την ενότητα τα εξετάσουμε αυτούς τους μετασχηματισμούς.⁶

⁶ Όπως σχεδόν όλα τα παραδείγματα έτσι και εδώ ως το περιβάλλον δοκιμών μας ορίζεται αυτό της Oracle.

Conversion Functions



Στην Oracle, οι τύποι στις στήλες των πινάκων μπορούν να οριστούν χρησιμοποιώντας American National Standards Institute (ANSI), DB2⁷, και SQL/DS⁸ τύπους δεδομένων, όμως εσωτερικά αποθηκεύονται μετά από μετατροπή σε έμφυτους τύπους του RDBMS.

Υπάρχουν περιπτώσεις κατά τις οποίες το σύστημα αναμένει δεδομένα συγκεκριμένου τύπου δεδομένων αλλά “δέχεται” άλλα, σε αυτές τις περιπτώσεις μπορεί να γίνει είτε μια αυτόματη εσωτερικά καθοριζόμενη μετατροπή των τύπων είτε μια ρητά καθοριζόμενη μετατροπή από τον χρήστη.

Η εσωτερικά καθοριζόμενη μετατροπές ακολουθούν κανόνες που έχουν οριστεί στην Oracle ενώ οι ρητά καθοριζόμενες μπορούν να γίνουν με την χρήση των συναρτήσεων μετατροπών. Οι συναρτήσεις αυτές μετατρέπουν την τιμή από ένα τύπο δεδομένων σε έναν άλλο. Σαν καλή πρακτική καλό θα είναι να εφαρμόζονται οι ρητές μετατροπές ακόμη και αν είναι δυνατή η αυτόματη εσωτερική μετατροπή των τύπων.

4.5.1 Εσωτερικός Μετασχηματισμός Τύπων Δεδομένων

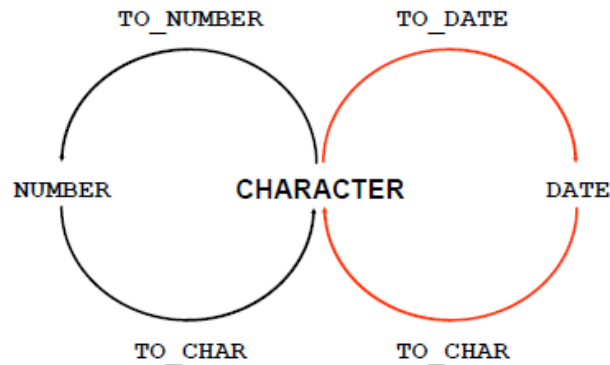
From	To
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE

Η Oracle μπορεί να κάνει αυτόματα μετατροπές τύπων. Για παράδειγμα η έκφραση `hire_date > '01-JAN-90'` έχει ως αποτέλεσμα την αυτόματη μετατροπή του κειμένου '01-JAN-90' σε ημερομηνία. Σε γενικές γραμμές οι μετατροπές αυτές πραγματοποιούνται όταν κάτι τέτοιο απαιτείται με βάση την δοθείσα έκφραση. Έτσι η έκφραση `grade = 20000` έχει ως αποτέλεσμα μετατροπή του αριθμού στο κείμενο "2" εάν το πεδίο `grade` είναι `CHAR(2)`. Να σημειωθεί ότι μετασχηματισμοί από κείμενο σε αριθμό μπορούν να πραγματοποιηθούν όταν το κείμενο αναπαριστά έναν έγκυρο αριθμό.

⁷ https://en.wikipedia.org/wiki/IBM_Db2

⁸ https://en.wikipedia.org/wiki/IBM_SQL/DS

4.5.2 Ρητός Μετασχηματισμός Τύπων Δεδομένων



Η SQL δίνει την δυνατότητα ο μετασχηματισμών μεταξύ τύπων να ορίζεται ρητά στην έκφραση. Για τον λόγο αυτό υπάρχουν υλοποιημένες συναρτήσεις που εξυπηρετούν αυτό το σκοπό. Καθώς το περιβάλλον δοκιμών μας είναι αυτό της Oracle οι διαθέσιμες συναρτήσεις αυτού του είδους είναι οι:

1. *to_char(number|date, [fmt])*: η οποία μετατρέπει αριθμητική τιμή ή τιμή ημερομηνίας σε κείμενο *VARCHAR2* με βάση την δοθείσα μορφοποίηση *fmt*.
2. *to_number(char, [fmt])*: η οποία μετατρέπει ένα κείμενο το οποίο περιέχει ψηφία σε αριθμό με βάση την δοθείσα μορφοποίηση *fmt*.
3. *to_date(char, [fmt])*: η οποία μετατρέπει ένα κείμενο το οποίο αναπαριστά μια ημερομηνία σε ημερομηνία με βάση την δοθείσα μορφοποίηση *fmt*.

Να σημειωθεί ότι αυτές αποτελούν μέρος των συνολικών συναρτήσεων αυτού του σκοπού. Για τον πλήρη κατάλογο θα πρέπει κάθε φορά να συμβουλευόμαστε το εγχειρίδιο του κατασκευαστή⁹.

Με σκοπό μια ποίο ολοκληρωμένη παρουσίαση παρατείνεται παρακάτω εικόνα από τον SQL Server η οποία παρουσιάζει αντίστοιχες δυνατότητες μετατροπών στο σύστημα αυτό.

⁹ Στη συγκεκριμένη περίπτωση το εγχειρίδιο ονομάζεται Oracle Database SQL Language Reference

From \ To	binary	varbinary	char	nchar	nvarchar	datetime	smalldatetime	date	time	datetimeoffset	datetime2	decimal	numeric	float	real	bigint	int(INT4)	smallint(INT2)	tinyint(INT1)	money	smallmoney	bit	timestamp	uniqueidentifier	image	ntext	text	sql_variant	xml	CLR UDT	hierarchyid
binary		●	●	●	●	●	●	■	■	■	■	●	✗	✗	●	●	●	●	●	●	●	●	●	✗	✗	●	●	●	●	●	●
varbinary	●		●	●	●	●	●	■	■	■	■	●	✗	✗	●	●	●	●	●	●	●	●	●	✗	✗	●	●	●	●	●	●
char	■	■		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	■	●	●	●	●	●	●	●	●	●
varchar	■	■	●		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	■	●	●	●	●	●	●	●	●	●
nchar	■	■	●	●		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	■	✗	●	●	●	●	●	●	●	●
nvarchar	■	■	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	■	✗	●	●	●	●	●	●	●	●
datetime	■	■	●	●	●		●	●	●	●	■	■	■	■	■	■	■	■	■	■	■	■	✗	✗	✗	✗	●	✗	✗	✗	✗
smalldatetime	■	■	●	●	●	●		●	●	●	■	■	■	■	■	■	■	■	■	■	■	■	✗	✗	✗	✗	●	✗	✗	✗	✗
date	■	■	●	●	●	●	●		✗	●	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	●	✗	✗	✗	
time	■	■	●	●	●	●	✗		●	●	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	●	✗	✗	✗	
datetimeoffset	■	■	●	●	●	●	●	●		●	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	●	✗	✗	✗	✗	
datetime2	■	■	●	●	●	●	●	●	●		✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	●	✗	✗	✗	✗	
decimal	●	●	●	●	●	●	✗	✗	✗	✗	◆	◆	●	●	●	●	●	●	●	●	●	●	✗	✗	✗	✗	●	✗	✗	✗	✗
numeric	●	●	●	●	●	●	✗	✗	✗	✗	◆	◆	●	●	●	●	●	●	●	●	●	●	✗	✗	✗	✗	●	✗	✗	✗	✗
float	●	●	●	●	●	●	✗	✗	✗	✗	●	●	●		●	●	●	●	●	●	●	●	✗	✗	✗	✗	●	✗	✗	✗	✗
real	●	●	●	●	●	●	✗	✗	✗	✗	●	●	●	●		●	●	●	●	●	●	●	✗	✗	✗	✗	●	✗	✗	✗	✗
bigint	●	●	●	●	●	●	✗	✗	✗	✗	●	●	●	●	●		●	●	●	●	●	●	✗	✗	✗	✗	●	✗	✗	✗	✗
int(INT4)	●	●	●	●	●	●	✗	✗	✗	✗	●	●	●	●	●	●		●	●	●	●	●	✗	✗	✗	✗	●	✗	✗	✗	✗
smallint(INT2)	●	●	●	●	●	●	✗	✗	✗	✗	●	●	●	●	●	●	●		●	●	●	●	✗	✗	✗	✗	●	✗	✗	✗	✗
tinyint(INT1)	●	●	●	●	●	●	✗	✗	✗	✗	●	●	●	●	●	●	●	●		●	●	●	✗	✗	✗	✗	●	✗	✗	✗	✗
money	●	●	●	●	●	●	✗	✗	✗	✗	●	●	●	●	●	●	●	●	●		●	●	✗	✗	✗	✗	●	✗	✗	✗	✗
smallmoney	●	●	●	●	●	●	✗	✗	✗	✗	●	●	●	●	●	●	●	●	●	●		●	✗	✗	✗	✗	●	✗	✗	✗	✗
bit	●	●	●	●	●	●	✗	✗	✗	✗	●	●	●	●	●	●	●	●	●	●	●		✗	✗	✗	✗	●	✗	✗	✗	✗
timestamp	●	●	●	✗	✗	●	✗	✗	✗	✗	●	●	✗	✗	●	●	●	●	●	●	●	✗	●	✗	✗	✗	✗	✗	✗	✗	✗
uniqueidentifier	●	●	●	●	●	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗		✗	✗	✗	●	✗	✗	✗	✗	
image	●	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	●	✗		✗	✗	✗	✗	✗	✗	✗
ntext	✗	✗	●	●	●	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗		●	●	●	✗	✗	✗	✗
text	✗	✗	●	●	●	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	●	●		●	●	●	✗	✗	✗
sql_variant	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	✗				✗	✗	✗	✗
xml	■	■	■	■	■	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	○	○	○	○	○	○
CLR UDT	■	■	■	■	■	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	○	○	○	○	○
hierarchyid	■	■	■	■	■	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗

■

 Explicit conversion

●

 Implicit conversion

✗

 Conversion not allowed

◆

 Requires explicit CAST to prevent the loss of precision or scale that might occur in an implicit conversion.

○

 Implicit conversions between xml data types are supported only if the source or target is untyped xml. Otherwise, the conversion must be explicit.

4.5.2.1 Ρητός Μετασχηματισμός με Χρήση των TO_CHAR, TO_NUMBER, TO_DATE

Η συνάρτηση `to_char` μπορεί να χρησιμοποιηθεί για την μετατροπή ημερομηνίας σε κείμενο τύπου `VARCHAR2` με βάση την δοθείσα μορφοποίηση. Η μορφοποίηση αποτελεί κείμενο το οποίο παρέχει οδηγίες ως προς την μορφή που θα έχει η ημερομηνία ως κείμενο.

```
TO_CHAR(date, 'format_model')
```

Για παράδειγμα για μια ημερομηνία της μορφής '11-Nov-1999' μια έγκυρη μορφοποίηση είναι 'DD-Mon-YYYY'. Η μετατροπή αυτού του είδους μπορεί να γίνει να μετατρέψει μια ημερομηνία από μια μορφοποίηση σε μία άλλη.

```
select sysdate, to_char(sysdate, 'mm/dd') from dual;
```

SYSDATE_	SYSDATE_MM_DD
13/8/2016 1:58:25 μμ	08/13

Η οδηγία για την μορφοποίηση θα πρέπει να περιβάλλεται από μονά εισαγωγικά και είναι ευαίσθητη στην διάκριση πεζών-κεφαλαίων. Μερικές από τις διαθέσιμες μορφοποιήσεις δείχνονται παρακάτω.

Element	Result
YYYY	Full year in numbers
YEAR	Year spelled out (in English)
MM	Two-digit value for the month
MONTH	Full name of the month
MON	Three-letter abbreviation of the month
DY	Three-letter abbreviation of the day of the week
DAY	Full name of the day of the week
DD	Numeric day of the month

Με ανάλογο τρόπο μπορεί να εμφανιστεί και το τμήμα που αφορά την ώρα

HH24:MI:SS AM	15:45:32 PM
---------------	-------------

ενώ η προσθήκη ενδιαμέσου κειμένου μπορεί να γίνει με τη χρήση διπλών εισαγωγικών

DD "of" MONTH	12 of OCTOBER
---------------	---------------

Οπότε για να εμφανίσουμε το όνομα του υπαλλήλου, την ημερομηνία πρόσληψης και την ημερομηνία πρόσληψης μορφοποιημένη ώστε να εμφανίζεται και η ώρα δίνουμε:

```
1 select ename first_name, hiredate, to_char(hiredate, 'dd-mm-yyyy hh24:mi') from employees;
```

FIRST_NAME	HIREDATE	TO_CHAR(HIREDATE, 'DD-MM-YYYYHH24:MI')
KING	17-NOV-81	17-11-1981 00:00
BLAKE	01-MAY-81	01-05-1981 00:00
CLARK	09-JUN-81	09-06-1981 00:00
JONES	02-APR-81	02-04-1981 00:00
SCOTT	19-APR-87	19-04-1987 00:00
FORD	03-DEC-81	03-12-1981 00:00
SMITH	17-DEC-80	17-12-1980 00:00
ALLEN	20-FEB-81	20-02-1981 00:00
WARD	22-FEB-81	22-02-1981 00:00
MARTIN	28-SEP-81	28-09-1981 00:00
TURNER	08-SEP-81	08-09-1981 00:00
ADAMS	23-MAY-87	23-05-1987 00:00
JAMES	03-DEC-81	03-12-1981 00:00
MILLER	23-JAN-82	23-01-1982 00:00

Εκτός από την παραπάνω χρήση με παρόμοιο τρόπο η συνάρτηση *to_char* μπορεί να χρησιμοποιηθεί για την μετατροπή αριθμών. Η μάσκα μορφοποίησης που μπορεί να χρησιμοποιηθεί εδώ δέχεται τις παρακάτω παραμέτρους.

Element	Result
9	Represents a number
0	Forces a zero to be displayed
\$	Places a floating dollar sign
£	Uses the floating local currency symbol
.	Prints a decimal point
,	Prints a comma as a thousands indicator

Element	Description	Example	Result
9	Numeric position (number of 9s determine display width)	999999	1234
0	Display leading zeros	099999	001234
\$	Floating dollar sign	\$999999	\$1234
£	Floating local currency symbol	£999999	£1234
D	Returns the decimal character in the specified position. The default is a period (.).	99D99	99.99
.	Decimal point in position specified	999999.99	1234.00
G	Returns the group separator in the specified position. You can specify multiple group separators in a number format model.	9,999	9G999
,	Comma in position specified	999,999	1,234
MI	Minus signs to right (negative values)	999999MI	1234-
PR	Parenthesize negative numbers	999999PR	<1234>
EEEE	Scientific notation (format must specify four Es)	99.999EEEE	1.234E+03
U	Returns in the specified position the “Euro” (or other) dual currency	U9999	€1234
V	Multiply by 10 <i>n</i> times (<i>n</i> = number of 9s after V)	9999V99	123400
S	Returns the negative or positive value	S9999	-1234 or +1234
B	Display zero values as blank, not 0	B9999.99	1234.00

Ως παράδειγμα θεωρήστε το ερώτημα στο οποίο θέλουμε να εμφανίσουμε το όνομα, τον μισθό καθώς και την τιμή του μισθού μορφοποιημένου ώστε να εμφανίζεται το σύμβολο του νομίσματος καθώς και το διαχωριστικό χιλιάδων για κάθε υπάλληλο.

```
3 select first_name, salary, to_char(sal, 'U9999G999') formatted_salary from emp;
4
```

FIRST_NAME	SALARY	FORMATED_SALARY
KING	5000	\$5.000
BLAKE	2850	\$2.850
CLARK	2450	\$2.450
JONES	2975	\$2.975
SCOTT	3000	\$3.000
FORD	3000	\$3.000
SMITH	800	\$800
ALLEN	1600	\$1.600
WARD	1250	\$1.250
MARTIN	1250	\$1.250
TURNER	1500	\$1.500
ADAMS	1100	\$1.100
JAMES	950	\$950
MILLER	1300	\$1.300

Ιδιαίτερη προσοχή πρέπει να δοθεί στις μορφοποιήσεις αυτού του είδους καθώς στην περίπτωση όπου τα ψηφία του αριθμού είναι περισσότερα από αυτά που έχουν οριστεί στην μάσκα τότε θα εμφανιστεί στο αποτέλεσμα κείμενο συμπληρωμένο με τον χαρακτήρα της δέσης (#).

```
3 select first_name, salary, to_char(sal, 'U9G9') formatted_salary from emp;
4
```

FIRST_NAME	SALARY	FORMATED_SALARY
KING	5000	#####

Εκτός βέβαια από την *to_char* υπάρχουν οι συναρτήσεις *to_number*, *to_date* για τις οποίες ισχύουν οι ίδιοι γενικοί κανόνες μορφοποίησης. Η συνάρτηση *to_number* μπορεί να χρησιμοποιηθεί για την μετατροπή κειμένου σε αριθμό,

TO_NUMBER(char[, 'format_model'])

ενώ η *to_date* για την μετατροπή κειμένου σε ημερομηνία.

TO_DATE(char[, 'format_model'])

Ως παράδειγμα θεωρήστε τα παρακάτω ερωτήματα:

```
SELECT TO_DATE ('08-10-2013', 'DD-MM-YYYY') AS date_test01,
       TO_DATE ('2013/10/08', 'YYYY/MM/DD') AS date_test02,
       TO_DATE ('08-10-2013', 'DD-MM-YYYY') AS date_test03
FROM DUAL;
```

DATE_TEST01	DATE_TEST02	DATE_TEST03
8/10/2013	8/10/2013	8/10/2013

```
SELECT TO_NUMBER ('5467.12'),
       TO_NUMBER ('5467.12', '9999999.99'),
       TO_NUMBER ('4687841', '9999999'),
       TO_NUMBER ('123,456,789', '999,999,999')
FROM DUAL;
```

TO_NUMBER('5467.12')	TO_NUMBER('5467.12','999999.99')	TO_NUMBER('4687841','9999999')	TO_NUMBER('123,456,789','999,999,999')
5467.12	5467.12	4687841	123456789

4.6 Εμφωλευμένες Συναρτήσεις

Οι συναρτήσεις μονής γραμμής μπορούν να εμφωλευτούν σε οποιοδήποτε βάθος, δίνοντας μεγαλύτερη ευελιξία ως προς το παραγόμενο αποτέλεσμα. Υπολογίζονται από την πιο εσωτερικό επίπεδο προς το πιο εξωτερικό. Στο παρακάτω παράδειγμα η εσωτερική συνάρτηση επιστρέφει τους πρώτους 8 χαρακτήρες του επιθέτου, η επόμενη συνενώνει το αποτέλεσμα αυτό με του λεκτικό `_US` και η εξωτερική μετατρέπει το αποτέλεσμα σε κεφαλαία:

```
SELECT last name,
       UPPER(CONCAT(SUBSTR (LAST_NAME, 1, 8), '_US'))
FROM   employees
WHERE  department_id = 60;
```

	LAST_NAME	UPPER(CONCAT(SUBSTR(LAST_NAME,1,8),'_US'))
1	Hunold	HUNOLD_US
2	Ernst	ERNST_US
3	Lorentz	LORENTZ_US

4.7 Συναρτήσεις Γενικού Σκοπού

`NVL (expr1, expr2)`
`NVL2 (expr1, expr2, expr3)`
`NULLIF (expr1, expr2)`
`COALESCE (expr1, expr2, ..., exprn)`

Οι συναρτήσεις που αναφέρονται εδώ αφορούν τον χειρισμό των `NULL` και μπορούν να χρησιμοποιηθούν με οποιοδήποτε τύπο δεδομένων. Πιο συγκεκριμένα η συνάρτηση `NVL` μετατρέπει την τιμή `null` που τυχόν υπάρχει στο πρώτο όρισμα (`expr1`) στην τιμή που εισάγεται ως δεύτερο όρισμα (`expr2`). Ο τύπος δεδομένων που επιστρέφει είναι πάντα ίδιος με τον τύπο του πρώτου ορίσματος.

Data Type	Conversion Example
NUMBER	<code>NVL (number_column, 9)</code>
DATE	<code>NVL (date_column, '01-JAN-95')</code>
CHAR or VARCHAR2	<code>NVL (character_column, 'Unavailable')</code>

Ως παράδειγμα θεωρήστε ότι θέλουμε να υπολογιστούν οι ετήσιες απολαβές των υπαλλήλων. Στην περίπτωση αυτή πρέπει να πολλαπλασιαστεί ο μηνιαίος μισθός τους επί 12 και στην συνέχεια να προστεθούν σε αυτόν, εάν υπάρχουν, τυχόν πρόσθετες απολαβές ως ποσοστό επί του μηνιαίου μισθού.

```
SELECT last_name, salary, commission_pct,
       (salary*12) + (salary*12*commission_pct) AN_SAL
FROM   employees;
```

	LAST_NAME	SALARY	COMMISSION_PCT	AN_SAL
1	King	24000	(null)	(null)
...				
11	Vargas	2500	(null)	(null)
12	Zlotkey	10500	0.2	151200
13	Abel	11000	0.3	171600
...				

Καταλαβαίνουμε ότι το δεύτερο σκέλος του υπολογισμού πρέπει να γίνει μόνο για εκείνους του υπαλλήλους οι οποίοι λαμβάνουν πρόσθετες απολαβές καθώς σε αντίθετη περίπτωση η πράξη με το *null* θα παράξει *null*. Για να το πετύχουμε αυτό, καθώς θέλουμε να υπολογίσουμε το αποτέλεσμα, για όλους τους υπαλλήλους πρέπει να μετατρέψουμε τυχόν *null* τιμές πριν την πραγματοποίηση των πράξεων υπολογισμού. Αυτό το επιτυγχάνουμε με την χρήση της *NVL*.

```
SELECT last name, salary, NVL(commission_pct, 0)
       (salary*12) + (salary*12*NVL(commission_pct, 0)) AN_SAL
FROM employees;
```

	LAST_NAME	SALARY	NVL(COMMISSION_PCT,0)	AN_SAL
1	King	24000	0	288000
2	Kochhar	17000	0	204000
3	De Haan	17000	0	204000
4	Hunold	9000	0	108000
5	Ernst	6000	0	72000
6	Lorentz	4200	0	50400
7	Mourgos	5800	0	69600
8	Rajs	3500	0	42000
9	Davies	3100	0	37200
10	Matos	2600	0	31200
11	Vargas	2500	0	30000
12	Zlotkey	10500	0.2	151200
...				

Με παρόμοιο τρόπο λειτουργεί και *NVL2* παρέχοντας λίγο μεγαλύτερη ευελιξία ως προς τον υπολογισμό του αποτελέσματος.

NVL2(expr1, expr2, expr3)

Στην σύνταξη της το:

πρώτο όρισμα-*expr1*: αποτελεί την τιμή που ελέγχεται και η οποία μπορεί να περιέχει *null*

δεύτερο όρισμα-*expr2*: αποτελεί την τιμή που επιστρέφεται στην περίπτωση που δεν είναι *null* το *expr1*

τρίτο όρισμα-*expr3*: αποτελεί την τιμή που επιστρέφεται στην περίπτωση που είναι *null* το *expr1*


```
SELECT last_name, salary, commission_pct
      NVL2(commission_pct,
            'SAL+COMM', 'SAL') income
FROM employees WHERE department_id IN (50, 80);
```

	LAST_NAME	SALARY	COMMISSION_PCT	INCOME
1	Mourgos	5800	(null)	SAL
2	Rajs	3500	(null)	SAL
3	Davies	3100	(null)	SAL
4	Matos	2600	(null)	SAL
5	Vargas	2500	(null)	SAL
6	Zlotkey	10500	0.2	SAL+COMM
7	Abel	11000	0.3	SAL+COMM
8	Taylor	8600	0.2	SAL+COMM

Στο παράδειγμα που δείχνεται, εξετάζεται η τιμή του πεδίου *COMMISSION_PCT*. Στην περίπτωση όπου υπάρχει τιμή τότε επιστρέφεται το λεκτικό *SAL+COMM*, ενώ στην περίπτωση που αυτή είναι *null* επιστρέφεται το λεκτικό *SAL*.

Μια άλλη συνάρτηση για τον χειρισμό των *NULLs* είναι και η *NULLIF* η οποία συγκρίνει τις δύο παραμέτρους που δέχεται ως ορίσματα. Στην περίπτωση όπου οι τιμές των παραμέτρων είναι ίσες τότε επιστρέφει *NULL*, ενώ στην περίπτωση που είναι διαφορετικές επιστρέφει την τιμή της πρώτης παραμέτρου.

```
SELECT first_name, LENGTH(first_name) "expr1",
      last_name, LENGTH(last_name) "expr2",
      NULLIF(LENGTH(first_name), LENGTH(last_name)) result
FROM employees;
```

	FIRST_NAME	expr1	LAST_NAME	expr2	RESULT
1	Ellen	5	Abel	4	5
2	Curtis	6	Davies	6	(null)
3	Lex	3	De Haan	7	3
4	Bruce	5	Ernst	5	(null)
5	Pat	3	Fay	3	(null)
6	William	7	Gietz	5	7
7	Kimberely	9	Grant	5	9

...			
19 Jennifer	8 Whalen	6	
20 Eleni	5 Zlotkey	7	

Τέλος υπάρχει και συνάρτηση *COALESCE*, της οποίας το χαρακτηριστικό είναι ότι σε αντίθεση με την *NVL* μπορεί να δεχθεί πολλές εναλλακτικές τιμές.

COALESCE (expr1, expr2, ... exprn)

Πιο συγκεκριμένα επιστρέφει την τιμή του *expr1* στη περίπτωση που αυτό δεν είναι *null* ή επιστρέφει την τιμή του *expr2* στη περίπτωση που το προηγούμενο όρισμα έχει *null* τιμή και το ίδιο δεν είναι *null* ή ... επιστρέφει την τιμή του *exprn* στην περίπτωση που όλα τα προηγούμενα είναι *null* και αυτό δεν είναι. Απαραίτητη προϋπόθεση για την σύγκριση στην *COALESCE* είναι όλα τα ορίσματα να είναι ίδιου τύπου.


```
SELECT last name, employee id,  
COALESCE(TO_CHAR(commission_pct),TO_CHAR(manager_id),  
          'No commission and no manager')  
FROM employees;
```

	1 LAST_NAME	2 EMPLOYEE_ID	3 COALESCE(TO_CHAR(COM
1	King	100	No commission and no manager
2	Kochhar	101	100
3	De Haan	102	100
4	Hunold	103	102
5	Ernst	104	103
6	Lorentz	107	103
7	Mourgos	124	100
8	Rajs	141	124

...

12	Zlotkey	149	.2
13	Abel	174	.3
14	Taylor	176	.2
15	Grant	178	.15
16	Vihalen	200	101

4.8 Υποθετικές Εκφράσεις στην SQL

Οι συναρτήσεις μονής γραμμής μπορούν να εμφωλευτούν σε οποιοδήποτε βάθος, δίνοντας μεγαλύτερη ευελιξία.

Για την εφαρμογή της λογικής *IF-THEN-ELSE* στην SQL υπάρχουν διαθέσιμες οι εκφράσεις *CASE* και *DECODE*. Από αυτές η έκφραση *CASE* είναι συμβατή με το πρότυπο ANSI SQL ενώ η *DECODE* αφορά την Oracle.

Η σύνταξη της *CASE* δείχνεται παρακάτω. Σε αυτή η *expr* ελέγχεται με την σειρά, ως προς την επαλήθευση με τις *comparison_expr1,2,...n*. Στην περίπτωση που είναι αληθής τότε επιστρέφεται η τιμή *return_expr1,2,...n* αντίστοιχα ενώ αν καμία δεν είναι αληθής και υπάρχει η σύνταξη *ELSE* τότε επιστρέφεται η τιμή *else_expr*.

```
CASE expr WHEN comparison_expr1 THEN return_expr1  
          [WHEN comparison_expr2 THEN return_expr2  
          WHEN comparison_exprn THEN return_exprn  
          ELSE else_expr]  
END
```

Ως παράδειγμα θεωρήστε το παρακάτω ερώτημα όπου έλεγχος γίνεται στο πεδίο *job_id* ώστε ανάλογα να υπολογιστεί και το ποσοστό αύξησης (δηλ. ανά κατηγορία εργασίας).

```

SELECT last_name, job_id, salary,
CASE job_id WHEN 'IT_PROG' THEN 1.10*salary
            WHEN 'ST_CLERK' THEN 1.15*salary
            WHEN 'SA_REP' THEN 1.20*salary
ELSE salary END "REVISED_SALARY"
FROM employees;

```

	LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...				
5	Ernst	IT_PROG	6000	6600
6	Lorentz	IT_PROG	4200	4620
7	Mourgos	ST_MAN	5800	5800
8	Rajs	ST_CLERK	3500	4025
9	Davies	ST_CLERK	3100	3565
...				
13	Abel	SA_REP	11000	13200
14	Taylor	SA_REP	8600	10320

Έτσι αν η εργασία είναι *IT_PROG* δίνεται μια αύξηση 10%, αν είναι *ST_CLERK* δίνεται μια αύξηση 20% ενώ αν είναι *SA_REP* δίνεται μια αύξηση 30%. Ένα άλλο παράδειγμα δείχνεται παρακάτω

```

SELECT last_name, salary,
(CASE WHEN salary<5000 THEN 'Low'
      WHEN salary<10000 THEN 'Medium'
      WHEN salary<20000 THEN 'Good'
      ELSE 'Excellent'
END) qualified_salary
FROM employees;

```

Παρόμοια λειτουργία με την παραπάνω προσφέρει και συνάρτηση *DECODE* της Oracle η οποία ενσωματώνει την IF-THEN-ELSE λογική.

```

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

```

Λειτουργεί συγκρίνοντας την έκφραση *col|expression* και στην περίπτωση που ταυτοποιηθεί με κάποιο από τα *search1,2,...,n* επιστρέφει το ανάλογο *result1,2,...,n*. Στην περίπτωση δε που δεν γίνει καμία ταυτοποίηση επιστρέφεται η *default* τιμή. Για να γίνει πιο κατανοητή η χρήση της θεωρήστε την περίπτωση του προηγούμενου ερωτήματος με το ποσοστό αύξησης ανάλογα με το είδος εργασίας το οποίο επιλύθηκε με την *CASE*. Στην περίπτωση της *DECODE* το ερώτημα μετασχηματίζεται όπως δείχνεται παρακάτω.

```

SELECT last name, job id, salary,
DECODE(job_id, 'IT_PROG', 1.10*salary,
        'ST_CLERK', 1.15*salary,
        'SA_REP', 1.20*salary,
        salary)
REVISED_SALARY
FROM employees;

```

	LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...				
6	Lorentz	IT_PROG	4200	4620
7	Mourgos	ST_MAN	5800	5800
8	Rajs	ST_CLERK	3500	4025
...				
13	Abel	SA_REP	11000	13200
14	Taylor	SA_REP	8600	10320
...				

4.9 Ασκήσεις

1. Γράψτε ένα ερώτημα το οποίο θα εμφανίζει για κάθε υπάλληλο μια εγγραφή της μορφής <επώνυμο υπαλλήλου> κερδίζει <μισθός> μηνιαία αλλά θα ήθελε <3 φορές τον μισθό του>. Ονομάστε το πεδίο που προκύπτει Dream Salaries.

	Dream Salaries
1	King earns \$24,000.00 monthly but wants \$72,000.00.
2	Kochhar earns \$17,000.00 monthly but wants \$51,000.00.
3	De Haan earns \$17,000.00 monthly but wants \$51,000.00.
4	Hunold earns \$9,000.00 monthly but wants \$27,000.00.
5	Ernst earns \$6,000.00 monthly but wants \$18,000.00.
...	
19	Higgins earns \$12,000.00 monthly but wants \$36,000.00.
20	Gietz earns \$8,300.00 monthly but wants \$24,900.00.

2. Εμφανίστε για κάθε υπάλληλο το επώνυμο του, την ημερομηνία πρόσληψης και την ημερομηνία αναθεώρησης του μισθού του η οποία ορίζεται ως η πρώτη Δευτέρα μετά από έξι (6) μήνες από την πρόσληψη του. Ονομάστε την στήλη REVIEW. Η μορφοποίηση της παραγόμενης ημερομηνίας να είναι της μορφής "Monday, the Thirty-First of July, 2000". Συμβουλευτείτε για τις δυνατότητες μορφοποίησης της ημερομηνίας σε Oracle με αναζήτηση στο διαδίκτυο.



	LAST_NAME	HIRE_DATE	REVIEW
1	King	17-JUN-87	Monday, the Twenty-First of December, 1987
2	Kochhar	21-SEP-89	Monday, the Twenty-Sixth of March, 1990
3	De Haan	13-JAN-93	Monday, the Nineteenth of July, 1993
4	Hunold	03-JAN-90	Monday, the Ninth of July, 1990
5	Ernst	21-MAY-91	Monday, the Twenty-Fifth of November, 1991
...			
19	Higgins	07-JUN-94	Monday, the Twelfth of December, 1994
20	Gietz	07-JUN-94	Monday, the Twelfth of December, 1994

3. Εμφανίστε το επώνυμο, την ημερομηνία πρόσληψης και την ημέρα πρόσληψης. Ονομάστε την στήλη _DAY. Ταξινομήστε το αποτέλεσμα ανά ημέρα ξεκινώντας από την Δευτέρα.

	LAST_NAME	HIRE_DATE	_DAY
1	Grant	24-MAY-99	MONDAY
2	Gietz	07-JUN-94	TUESDAY
3	Taylor	24-MAR-98	TUESDAY
4	Higgins	07-JUN-94	TUESDAY
5	Rajs	17-OCT-95	TUESDAY
...			
19	Lorentz	07-FEB-99	SUNDAY
20	Fay	17-AUG-97	SUNDAY

4. Χρησιμοποιώντας την DECODE, γράψτε ένα ερώτημα το οποίο θα εμφανίζει τον βαθμό όλων των υπαλλήλων ο οποίος θα παράγεται από το JOB_ID, με βάση τις παρακάτω παραδοχές

<i>Job</i>	<i>Grade</i>
AD_PRES	A
ST_MAN	B
IT_PROG	C
SA_REP	D
ST_CLERK	E
None of the above	0

	 JOB_ID	 GRADE
1	AC_ACCOUNT	0
2	AC_MGR	0
3	AD_ASST	0
4	AD_PRES	A
5	AD_VP	0

...

18	ST_CLERK	E
19	ST_CLERK	E
20	ST_MAN	B