

**Τίτλος Επιμορφωτικού Προγράμματος:**

# Η Χρήση της Oracle

**Εκπαιδευτικές Σημειώσεις**

**ΙΝ.ΕΠ. - ΜΑΡΤΙΟΣ 2014**

## **Συγγραφική Ομάδα:**

### **Συντονιστής:**

Πεχλιβανίδης Ηλίας , Προϊστάμενος Δ/σης Προγραμματισμού και Ανάπτυξης Ε.Κ.Δ.Δ.Α.

### **Συντάκτες:**

Ζαφείρης Δημήτριος, τ. Διευθυντής ΚΕ.Π.Υ.Ε.Σ.  
Κιντής Αντώνιος, Στέλεχος Ε.Μ.Π.  
Νταγιόγλου Ηλίας, Στέλεχος Ε.Μ.Π.

Κεφάλαιο 1: Εισαγωγή στις Βάσεις Δεδομένων  
(Ζαφείρης Δημήτριος)

Κεφάλαιο 2: Εισαγωγή στην Αρχιτεκτονική της ORACLE  
(Ζαφείρης Δημήτριος)

Κεφάλαιο 3: Η ORACLE SQL/DML – ΕΝΤΟΛΗ SELECT  
(Κιντής Αντώνιος)

Κεφάλαιο 4: Η ORACLE SQL/DML – INSERT, UPDATE, DELETE  
(Νταγιόγλου Ηλίας)

Κεφάλαιο 5: Η Oracle SQL/DDDL  
(Νταγιόγλου Ηλίας)

Κεφάλαιο 6: Η ORACLE PL/SQL  
(Νταγιόγλου Ηλίας)

Κεφάλαιο 7: ΘΕΜΑΤΑ ΔΙΑΧΕΙΡΙΣΗΣ- ΑΣΦΑΛΕΙΑ  
(Ζαφείρης Δημήτριος)

Παράρτημα Ι – Περιγραφή του σχήματος HR  
(Κιντής Αντώνιος)

Παράρτημα ΙΙ – Εισαγωγή στη χρήση του SQLDeveloper  
(Ζαφείρης Δημήτριος)

### **Αξιολογητές:**

Δρ. Μαραγκός Ηλίας, Αναπληρωτής Διευθυντής ΙΝ.ΕΠ.

Δρ. Μαργαριτόπουλος Μερκούριος, Προϊστάμενος Π.ΙΝ.ΕΠ.

## Περιεχόμενα

1. Εισαγωγή στις Βάσεις Δεδομένων. ....	9
Ορισμός. ....	9
Τύποι Βάσεων Δεδομένων. ....	10
Χαρακτηριστικά των Βάσεων Δεδομένων ....	11
Μέγεθος Βάσεων Δεδομένων. ....	11
Υλικό . ....	12
Λογισμικό Βάσεων Δεδομένων. ....	13
Σύστημα Βάσεων Δεδομένων.....	13
Παράδειγμα Βάσεων Δεδομένων.....	13
Χρήστες Βάσεων Δεδομένων. ....	16
Ιδιότητες Βάσεων Δεδομένων.....	17
Πλεονεκτήματα Βάσεων Δεδομένων. ....	18
Μοντέλα Δεδομένων.....	19
Αρχιτεκτονική Βάσεων Δεδομένων. ....	21
Είδη Μοντέλων Δεδομένων. ....	22
Τα Εννοιολογικά Μοντέλα.....	23
Το Σχεσιακό Μοντέλο (relational model). ....	23
Κανονικοποίηση Βάσεων Δεδομένων. ....	27
2. Εισαγωγή στην Αρχιτεκτονική της ORACLE.....	29
Ορισμός Oracle.....	29
Τα χαρακτηριστικά του Oracle Server. ....	29
Μέθοδοι Πρόσβασης στον Oracle Server.....	30
Αρχιτεκτονική του Oracle Server. ....	32
Δομές Μνήμης.....	34
Αρχιτεκτονική διαδικασιών (Process Architecture). ....	35
Αρχιτεκτονική Διαδικασιών στο παρασκήνιο (Background Processes). ....	36
Αρχιτεκτονική Αποθήκευσης Βάσης Δεδομένων (Database Storage Architecture). ....	39
3. Η ORACLE SQL/DML – ΕΝΤΟΛΗ SELECT .....	43
Σκοπός και στόχοι της ενότητας.....	43
Εισαγωγή .....	43
Προτάσεις SQL (Statements) .....	44
Η χρήση της SQL .....	45

Κατηγορίες εντολών της ORACLE SQL .....	46
Λεξικολογικές Συμβάσεις (Lexical Conventions) .....	47
Εργαλεία σύνταξης και εκτέλεσης εντολών ORACLE SQL .....	48
Δημιουργία απλών ερωτημάτων (πρόταση SELECT) .....	48
Περιγραφή Πίνακα .....	50
Σχόλια σε μια εντολή SQL.....	51
Περιορισμοί και ταξινόμηση δεδομένων.....	51
Ψευδώνυμα Στηλών (column alias).....	53
Επιλογή δεδομένων από πίνακες άλλου σχήματος (χρήση) .....	54
Ο Ψευδοπίνακας DUAL.....	54
SQL Τελεστές (SQL Operators ) .....	56
Αριθμητικοί τελεστές.....	57
Τελεστής συνένωσης (Concatenation Operator) .....	58
Τελεστές συνόλων (Set Operators).....	59
Τελεστές οριζόμενοι από τους χρήστες.....	59
SQL Συνθήκες (Conditions) .....	60
Συνθήκες σύγκρισης (Comparison Conditions) .....	60
Λογικές Συνθήκες (Logical Conditions).....	61
Συνθήκες αναζήτησης μοτίβων (Pattern-matching Conditions) .....	62
Null συνθήκες (Null Conditions) .....	64
BETWEEN συνθήκες (BETWEEN Conditions ) .....	64
IN συνθήκη (IN Condition).....	65
EXISTS συνθήκη (EXISTS Condition).....	67
Η ψευδοστήλη ROWNUM (Pseudocolumn) .....	68
Οι SQL Συναρτήσεις .....	69
Συναρτήσεις μίας γραμμής (Single row functions) .....	70
Συναρτήσεις ομαδοποίησης ή συγκεντρωτικές συναρτήσεις (Aggregate functions) .....	89
Ομαδοποίηση Δεδομένων (GROUP BY clause) .....	92
Η φράση HAVING (HAVING clause) .....	93
Ανάκτηση δεδομένων από περισσότερους πίνακες ( joins) .....	96
Συνενώσεις ισότητας (ισοσυνδέσεις – Equijoins-Inner Joins).....	97
Συνενώσεις ανισότητας και Antijoins.....	99
Ημί-ενώσεις (Semijoins) .....	99
Εξωτερικές συνενώσεις (Outer Joins) .....	101

Αυτό-ενώσεις (Self Joins).....	103
Καρτεσιανά γινόμενα (Cartesian Products) .....	104
Σύνταξη και χρήση υποερωτημάτων (subqueries) στην ORACLE .....	105
Το υποερώτημα σαν πίνακας (inline view) .....	105
Ένθετα υποερωτήματα (nested subqueries).....	106
Πράξεις συνόλων .....	109
Η Πράξη της Ένωσης (UNION ή UNION ALL) .....	109
Η Πράξη της Τομής (INTERSECT).....	112
Η Πράξη της Διαφοράς (MINUS) .....	113
4. Η ORACLE SQL/DML – INSERT, UPDATE, DELETE .....	115
Σκοπός και στόχοι της ενότητας.....	115
Εισαγωγή .....	115
Εισαγωγή εγγραφών – η εντολή INSERT .....	116
Δυο σημαντικότερες εντολές: COMMIT, ROLLBACK .....	117
Μαζική εισαγωγή εγγραφών - η εντολή INSERT INTO ... SELECT.....	118
Ενημέρωση εγγραφών – η εντολή UPDATE .....	118
Διαγραφή εγγραφών – η εντολή DELETE .....	119
Γρήγορη διαγραφή εγγραφών – η εντολή TRUNCATE TABLE .....	120
Αυτόματη δημιουργία εντολών SQL με χρήση εντολών SQL.....	121
Η ψευδοστήλη ROWID .....	122
5. Η ORACLE SQL/DDL .....	124
Σκοπός και στόχοι της ενότητας.....	124
Εισαγωγή .....	124
Διαχείριση πινάκων στην ORACLE με τη χρήση της SQL/DDL.....	125
Δημιουργία πινάκων στην ORACLE .....	125
Συνήθεις τύποι δεδομένων στην ORACLE .....	128
Συνήθεις περιορισμοί.....	129
Δημιουργία πίνακα με τον SQLDeveloper .....	129
Δημιουργία πίνακα από το αποτέλεσμα ενός ερωτήματος.....	131
Τροποποίηση δομής πίνακα.....	132
Μετονομασία, αντιγραφή, διαγραφή πίνακα.....	133
Άλλα αντικείμενα της ORACLE.....	134
Όψεις (VIEWS) στην ORACLE.....	135
Ευρετήρια (INDEXES) στην ORACLE .....	137

Ακολουθίες (SEQUENCES) .....	141
6. Η ORACLE PL/SQL .....	144
Σκοπός και στόχοι της ενότητας.....	144
Τι είναι η ORACLE PL/SQL .....	144
Πλεονεκτήματα της PL/SQL .....	145
Στοιχεία της PL/SQL .....	146
Δομή ενός απλού (ανώνυμου) τμήματος κώδικα PL/SQL.....	147
Ένα πρώτο παράδειγμα κώδικα σε PL/SQL.....	147
Δηλώσεις μεταβλητών και σταθερών .....	148
Σχόλια και κενές γραμμές στην PL/SQL.....	149
Η εντολή SELECT INTO .....	150
Η εντολή εκχώρησης .....	150
Οι εντολές INSERT, UPDATE, DELETE στην PL/SQL.....	150
Εκτέλεση ενός ανώνυμου τμήματος κώδικα PL/SQL .....	151
Η δομή ελέγχου ροής IF...THEN .....	151
Η δομή επανάληψης WHILE...LOOP .....	153
Η δομή επανάληψης FOR...LOOP .....	155
Δρομείς (CURSORS) .....	156
Εξαιρέσεις (exceptions) .....	157
Εξαιρέσεις χρήστη (user exceptions).....	161
Διαδικασίες, συναρτήσεις και πακέτα στην PL/SQL .....	162
Γιατί να χρησιμοποιήσει κανείς αποθηκευμένες διαδικασίες / συναρτήσεις; .....	163
Σύνταξη - δημιουργία Διαδικασιών / Συναρτήσεων .....	163
Κλήση μιας διαδικασίας / συνάρτησης PL/SQL .....	165
Παράδειγμα αποθηκευμένης διαδικασίας .....	166
Παράδειγμα αποθηκευμένης συνάρτησης.....	168
Πακέτα στην PL/SQL.....	170
Σύνταξη και δημιουργία πακέτων στην PL/SQL.....	171
Χρήση διαδικασίας / συνάρτησης ενός πακέτου .....	174
7. ΘΕΜΑΤΑ ΔΙΑΧΕΙΡΙΣΗΣ- ΑΣΦΑΛΕΙΑ .....	175
Ο Oracle Enterprise Manager. ....	175
Ασφάλεια – Διαχείριση χρηστών (Database users).....	180
Δημιουργία / διαγραφή χρηστών .....	180
Δικαιώματα (Privileges).....	183

Ρόλοι (Roles).....	185
Αντίγραφα Ασφαλείας (Oracle Database Backup). ....	186
Απόδοση Β.Δ. (Performance). ....	191
Παράρτημα Ι – Περιγραφή του σχήματος HR. ....	194
Εισαγωγή .....	194
Σύντομη περιγραφή του παραδείγματος.....	194
HR σχήμα .....	195
Περιγραφή των πινάκων του HR .....	196
Παράρτημα ΙΙ – Εισαγωγή στη χρήση του SQLDeveloper .....	199
Ορισμός. ....	199
Το Περιβάλλον Διαχείρισης.....	200
Το menu Connections.....	201
Το menu Edit:.....	202
Το menu View:.....	202
Το menu Navigate:.....	202
Το menu Run:.....	202
Το Tools menu. ....	203
Βιβλιογραφία – Ιστότοποι.....	204





## 1. Εισαγωγή στις Βάσεις Δεδομένων.

### Σκοπός και στόχοι της ενότητας

Σκοπός αυτής ενότητας είναι η κατανόηση των βασικών όρων και εννοιών για την σχεδίαση και υλοποίηση μιας Βάσης Δεδομένων, με βάση του σχεσιακού μοντέλου.

Έτσι, μετά το τέλος της ενότητας, οι επιμορφωμένοι θα είναι σε θέση να γνωρίζουν τα παρακάτω αντικείμενα:

1. Ορισμός των Βάσεων Δεδομένων.
2. Τύποι Βάσεων Δεδομένων.
3. Χαρακτηριστικά Βάσεων Δεδομένων.
4. Μέγεθος Βάσεων Δεδομένων.
5. Υλικό
6. Λογισμικό Βάσεων Δεδομένων.
7. Σύστημα Βάσεων Δεδομένων.
8. Παράδειγμα Βάσεων Δεδομένων.
9. Χρήστες Βάσεων Δεδομένων.
10. Ιδιότητες Βάσεων Δεδομένων.
11. Πλεονεκτήματα Βάσεων Δεδομένων.
12. Μοντέλα Δεδομένων.
13. Αρχιτεκτονική Βάσεων Δεδομένων.
14. Είδη Μοντέλων Δεδομένων.
15. Τα Εννοιολογικά Μοντέλα.
16. Το Σχεσιακό Μοντέλο (relational model).
17. Κανονικοποίηση Βάσεων Δεδομένων.

### Ορισμός.

**Οι Βάσεις Δεδομένων** είναι ένα μηχανισμός αποθήκευσης στοιχείων – Δεδομένων (Data) που έχουν σχέση μεταξύ τους και μπορούν να καταγραφούν. Τα στοιχεία αυτά μπορεί να αντιπροσωπεύουν λογικές τιμές όπως προσωπικά στοιχεία ατόμων (

όνομα, επώνυμο, διεύθυνση, τηλέφωνο) ή στοιχεία προϊόντων (περιγραφή, τιμή, κατηγορία, αριθμός τιμολογίου) όπως επίσης μπορεί να αντιπροσωπεύουν και φυσικά στοιχεία όπως συλλογές αρχείων.

Οι Βάσεις Δεδομένων ευρίσκονται δίπλα μας και διαδραματίζουν σημαντικό ρόλο σε όλους τους τομείς που χρησιμοποιούνται Ηλεκτρονικοί Υπολογιστές, όπως στην μηχανική, στην ιατρική, στην εκπαίδευση, στο ηλεκτρονικό εμπόριο, στα ΜΜΕ και σε όλες γενικά τις επιστήμες. Η εξέλιξη της τεχνολογίας των Βάσεων Δεδομένων είναι άρρηκτα συνδεδεμένη με αυτής των Ηλεκτρονικών Υπολογιστών.

### Τύποι Βάσεων Δεδομένων.

Οι πρώτες μορφές των Βάσεων Δεδομένων χρησίμευαν για να αποθηκεύσουν απλό κείμενο και αριθμούς. Στην συνέχεια όμως με την εξέλιξη της τεχνολογίας ανεπτύχθησαν οι παρακάτω τύποι Βάσεων Δεδομένων:

- Οι Βάσεις Δεδομένων Πολυμέσων για την αποθήκευση και άλλου είδους δεδομένα όπως εικόνες, ήχους, video.
- Τα Πληροφορικά Συστήματα Υγείας για την αποθήκευση ιατρικών δεδομένων όπως αξονικές και μαγνητικές ακτινογραφίες, κάθε είδους εξετάσεις με σκοπό την δημιουργία ιατρικού φακέλου των πολιτών.
- Τα Γεωγραφικά Πληροφορικά Συστήματα (GIS) για αποθήκευση και εκμετάλλευση των γεωγραφικών δεδομένων.
- Τα Συστήματα επεξεργασίας πραγματικού χρόνου (on-line) για εφαρμογές τύπου κρατήσεων θέσεων για μεγάλα ταξιδιωτικά γραφεία και αεροπορικές εταιρείες.
- Τα Συστήματα αναλυτικής επεξεργασίας δεδομένων (OLAP) για επεξεργασία δεδομένων με τεχνικές εξόρυξης δεδομένων (Data mining).
- Οι αποθήκες Δεδομένων (data warehouse) για την αποθήκευση τεραστίου όγκου δεδομένων και τήρηση ιστορικών στοιχείων.

- Οι Βάσεις Δεδομένων για την υποστήριξη εφαρμογών Διαδικτύου (Web Databases) οι οποίες εφαρμόζουν ειδικές τεχνικές αναζήτησης στο διαδίκτυο για την εύρεση και εκμετάλλευση δεδομένων.

## **Χαρακτηριστικά των Βάσεων Δεδομένων .**

Τα Βασικά χαρακτηριστικά των Βάσεων Δεδομένων είναι:

- Αποθήκευση δεδομένων είτε σε φυσική είτε σε λογική μορφή όπως αναφέρθηκε προηγουμένως.
- Αναπαράσταση κάποιας άποψης του πραγματικού κόσμου όπως μιας επιχείρησης, ενός εκπαιδευτικού ιδρύματος ή ενός συστήματος όπως φορολογικού. Τα στοιχεία που αποθηκεύονται έχουν μια λογική σχέση μεταξύ τους όπως τα στοιχεία ενός συστήματος παρακολούθησης προϊόντων. Δεν είναι δηλαδή άσχετα μεταξύ τους όπως οι λέξεις ενός λεξικού.
- Τα στοιχεία αποθηκεύονται σε λογικές μονάδες που σχεδιάζονται για την περαιτέρω επεξεργασία.
- Για την δημιουργία τους ακολουθείται μια συγκεκριμένη διαδικασία η οποία περιλαμβάνει τον σχεδιασμό (Planning), την απεικόνιση στον Ηλεκτρονικό Υπολογιστή (Build), την εισαγωγή των στοιχείων (Store) και την εκμετάλλευση των στοιχείων αυτών με φόρμες (Forms), ερωτήματα (Queries) και αναφορές (Reports).
- Την ύπαρξη ενός λογισμικού για την διαχείριση των στοιχείων αυτών. Το λογισμικό αυτό ονομάζεται Σύστημα Διαχείρισης Βάσεων Δεδομένων (Database Management System - DBMS ) και σκοπό έχει να διαχειρισθεί την εκμετάλλευση των στοιχείων αυτών.

## **Μέγεθος Βάσεων Δεδομένων.**

Μια Βάση Δεδομένων μπορεί να έχει οποιανδήποτε μέγεθος και πολυπλοκότητα. Έτσι μπορεί να συναντήσουμε μικρές Βάσεις Δεδομένων όπως για την τήρηση μιας λίστας ονομάτων τηλεφωνικού καταλόγου, μεγαλύτερες όπως μιας εταιρείας παρακολούθησης προϊόντων, μιας κατασκευαστικής εταιρείας, ενός πανεπιστημίου, μιας τράπεζας για να φθάσουμε στο μέγεθος ενός ασφαλιστικού οργανισμού ή μιας τηλεφωνικής εταιρείας ή της εφορίας όπου στις περιπτώσεις αυτές τηρείται τεράστιος όγκος Δεδομένων.

## Υλικό .

Το υλικό που εγκαθίστανται οι Βάσεις Δεδομένων ανάλογα με το μέγεθος τους διακρίνεται:

- Σε Υπολογιστικά Συστήματα μεγάλων δυνατοτήτων (Servers), που διαθέτουν αριθμό από κατάλληλους επεξεργαστές (CPU), μεγάλη κύρια μνήμη (RAM) πολλών GB, πολλαπλά τροφοδοτικά και γενικά ειδικό εξοπλισμό και εξαρτήματα για την αντιμετώπιση βλαβών και αστοχιών.
- Σε κατάλληλα Συστήματα αποθήκευσης δευτερεύουσας μνήμης με πολλαπλούς δίσκους (Disks Drives) και αντίστοιχες συσκευές μονάδων εισόδου – εξόδου (input – output), ελεγκτές δίσκων (Disk Controllers) και κατάλληλα κανάλια εισόδου – εξόδου. Επίσης περιλαμβάνει ειδικές μονάδες για την λήψη αντιγράφων ασφαλείας. (Back up)
- Το υλικό αυτό περιλαμβάνει και την ύπαρξη εφεδρικών Συστημάτων τα οποία είτε βρίσκονται σε παράλληλη λειτουργία με το κανονικό Σύστημα είτε βρίσκονται σε κατάσταση ετοιμότητας, παρακολουθούν με κατάλληλο λογισμικό το Βασικό Σύστημα και αναλαμβάνουν λειτουργία σε περίπτωση βλάβης αυτού.

Όλο το υλικό βρίσκεται σε κατάλληλα διαμορφωμένους χώρους (Data Centers) που παρέχουν τις απαραίτητες συνθήκες ψύξης και παροχής ενέργειας, όπως και την απαραίτητη ασφάλεια. Η τελευταία εξέλιξη στην φυσική αποθήκευση των δεδομένων, αφορά την μεταφορά και αποθήκευση αυτών σε απομακρυσμένους χώρους με την εφαρμογή της τεχνολογίας cloud computing.

## Λογισμικό Βάσεων Δεδομένων.

Όπως αναφέρθηκε προηγουμένως για την διαχείριση των Βάσεων Δεδομένων απαιτείται ένα σύνολο προγραμμάτων που ονομάζεται Λογισμικό Διαχείρισης Βάσεων Δεδομένων (Database Management System DBMS). Σκοπός του λογισμικού αυτού είναι η δημιουργία (create) και η συντήρηση (maintain) της Βάσης Δεδομένων. Με την χρήση του λογισμικού γίνονται οι παρακάτω λειτουργίες που αφορούν μια Βάση Δεδομένων:

- Ο ορισμός της Βάσης Δεδομένων (definition). Είναι ο καθορισμός των τύπων δεδομένων (data types), οι δομές (structures) και οι περιορισμοί (constraints) των στοιχείων της Βάσης. Ο καθορισμός των πληροφοριών αυτών, ονομάζεται και λεξικό δεδομένων (meta data).
- Η κατασκευή της Βάσης Δεδομένων (construct). Αποτελεί την δημιουργία της Βάσης στο κατάλληλο φυσικό μέσο αποθήκευσης.
- Ο χειρισμός (data manipulation) των στοιχείων της Βάσης Δεδομένων που περιλαμβάνει τις λειτουργίες της αποθήκευσης (store), ερωτήσεων (queries), τροποποιήσεων (update) και διαγραφής (delete) αυτών.

## Σύστημα Βάσεων Δεδομένων.

Σύστημα Βάσεων Δεδομένων, ονομάζεται το Λογισμικό της Βάσης μαζί με την Βάση Δεδομένων που αντιπροσωπεύει την φυσική αποθήκευση των στοιχείων.

## Παράδειγμα Βάσεων Δεδομένων.

Για την κατανόηση των παραπάνω θα παραθέσουμε ένα παράδειγμα Βάσης Δεδομένων που αναφέρεται στην παρακολούθηση προσωπικού. Η Βάση αυτή βρίσκεται στην εγκατάσταση της ORACLE 11g και είναι γνωστή με το όνομα HR.

Η Βάση αποτελείται από 9 αρχεία, τα οποία αποθηκεύουν εγγραφές του ιδίου τύπου. Αυτά είναι:

REGION (Περιοχές)	REGION_ID (ΚΩΔΙΚΟΣ_ΠΕΡΙΟΧΗΣ)	REGION_NAME (ΟΝΟΜΑΣΙΑ_ΠΕΡΙΟΧΗΣ)
	1	EUROPE
	2	AMERICA
	3	ASIA
	4	AFRICA

COUNTRY ΧΩΡΑ	COUNTRY_ID ΚΩΔΙΚΟΣ_ΧΩΡΑΣ	COUNTRY_NAME ΟΝΟΜΑΣΙΑ_ΧΩΡΑΣ	REGION_ID ΚΩΔΙΚΟΣ_ΠΕΡΙΟΧΗΣ
	1	ARGENTINA	2
	2	AUSTRIA	1

LOCATIONS ΤΟΠΟΘΕΣΙΕΣ	LOC_ID ΚΩΔΙΚΟΣ	ADDRESS ΔΙΕΥΘΥΝΣΗ	P_CODE TAX_ΚΩΔ	CITY ΠΟΛΗ	COUNTRY_ID ΚΩΔ_ΧΩΡΑΣ
	1000	1297 Via Cola	00989	ROMA	1
	2000	93091 Calle	11111	VENICE	2

DEPARTMENTS	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID
	10	Administration	200
	20	Marketing	210

JOB ΕΡΓΑΣΙΑ	JOB_ID ΚΩΔΙΚΟΣ	JOB_TITLE ΟΝΟΜΑΣΙΑ	MIN_SALARY ΕΛΑΧΙΣΤΟΣ_ΜΙΣ	MAX_SALARY ΜΕΓΙΣΤΟΣ_ΜΙΣ
	AD_PRES	President	20000	40000
	AD_VP	Administration Vice President	15000	20000

EMPLOYEES ΥΠΑΛΛΗΛΟΙ	EMPLOYEE_ID ΚΩΔΙΚΟΣ	F_NAME ΟΝΟΜΑ	DEPARTMENT ID	MANAGER_ID ΕΠΙΒΛΕΠΩΝ
	198	Donald	10	124
	199	Douglas	20	125

JOB_HISTORY ΙΣΤΟΡΙΚΟ	EMPLOYEE_ID ΚΩΔΙΚΟΣ	START_DATE ΗΜΕΡ_ΕΝΑΡΞ	END_DATE ΗΜΕΡ_ΛΗΞΗΣ	JOB_ID ΚΩΔΙΚΟΣ
	102	13/01/01	24/07/06	IT_PROG

PROJECTS ΕΡΓΑ	PROJECT_ID	PROJECT_NAME
	1	TAX APPLICATION
	2	WEB DEV

WORKS_ON ΕΡΓΑΣΙΑ	PROJECT_ID	EMPLOYEE_ID	START_DATE
	1	198	24/07/06
	2	199	25/07/06

ΣΧΗΜΑ 1- 1 Παράδειγμα Βάσης Δεδομένων

Βλέποντας την αποθήκευση των δεδομένων στους πίνακες παρατηρούμε τα παρακάτω:

- Η Πληροφορία αποθηκεύεται μια φορά δηλ τα προσωπικά στοιχεία του Υπαλλήλου βρίσκονται μόνον στο πίνακα EMPLOYEES. Ομοίως τα στοιχεία του κάθε τμήματος βρίσκονται στο αρχείο DEPARTMENTS.
- Τα αρχεία συσχετίζονται μεταξύ τους με τα πεδία που φέρουν τους κωδικούς. Έτσι βλέπουμε ότι για το DEPARTMENT κάθε υπαλλήλου στο αρχείο EMPLOYEES, αποθηκεύεται μόνον ο κωδικός αυτού και όχι όλη η πληροφορία.

Αφού γίνει η αποθήκευση των δεδομένων σε αρχεία όπως παραπάνω ακολουθεί η εκμετάλλευση αυτής με ερωτήματα (queries) του τύπου:

- Δώστε όλα τα ονόματα των EMPLOYEES που ανήκουν στο DEPARTMENT 10.
- Δώστε τον MIN\_SALARY και MAX\_SALARY όλων των EMPLOYEES που ανήκουν σε DEPARTMENTS που το LOCATIONS είναι 1700.

Οι τροποποιήσεις (update) της Βάσης έχουν την μορφή:

- Αλλαγή του DEPARTMENT του EMPLOYEE Donald σε 20.
- Αλλαγή των μισθών στο DEPARTMENT 20.

### Χρήστες Βάσεων Δεδομένων.

Οι χρήστες των Βάσεων Δεδομένων χωρίζονται σε τρεις (3) κατηγορίες:

- **Οι τελικοί χρήστες:** Είναι αυτοί που μπορούν να προσπελάσουν την Βάση Δεδομένων μέσω τερματικών σταθμών εργασίας που είναι συνδεδεμένοι με την Βάση Δεδομένων. Χρησιμοποιούν κυρίως μια εφαρμογή που παρέχει μια διασύνδεση με την Βάση Δεδομένων (interface) όπως Φόρμες. Επίσης μπορεί να χρησιμοποιήσουν ενσωματωμένη διασύνδεση που παρέχει το DBMS ή εργαλεία και εφαρμογές τρίτων κατασκευαστών.
- **Οι προγραμματιστές εφαρμογών.** Πρόκειται για χρήστες με εξειδικευμένες γνώσεις προγραμματισμού που εκμεταλλεύονται την Βάση Δεδομένων μέσω προγραμμάτων που συντάσσονται σε κάποια γλώσσα προγραμματισμού. (πχ SQL, PL/SQL, C, C++, JAVA, .NET).
- **Ο Διαχειριστής της Βάσης Δεδομένων (Database Administrator –DBA).** Είναι υπεύθυνος για την διαχείριση της Βάσης Δεδομένων και συγκεκριμένα:
  - α Εγκαθιστά το DBMS.
  - β Σχεδιάζει την δομή της Βάσης Δεδομένων.



- γ Δημιουργεί την Βάση Δεδομένων.
- δ Τηρεί τα εφεδρικά αρχεία της Βάσης Δεδομένων.
- ε Δημιουργεί και συντηρεί τους χρήστες της Βάσης Δεδομένων.
- στ Παρακολουθεί την λειτουργία της Βάσης Δεδομένων.
- η Αποκαθιστά την Βάση σε περίπτωση αστοχίας.
- θ Αναπτύσσει τις εφαρμογές εκμετάλλευσης της Βάσης Δεδομένων.

## Ιδιότητες Βάσεων Δεδομένων.

Οι Βάσεις Δεδομένων σε σύγκριση με την παραδοσιακή αποθήκευση στοιχείων με αρχεία (flat files) έχει τα παρακάτω χαρακτηριστικά:

- **Δεν αποθηκεύουν** μόνον τις τιμές (values) των στοιχείων που περιέχουν, αλλά και τον ορισμό αυτών. Αυτό γίνεται στον κατάλογο (catalog) του DBMS με την μορφή των meta-data.
- **Ανεξαρτησία προγραμμάτων και δεδομένων.** Αφού στις Βάσεις Δεδομένων, ο ορισμός των στοιχείων αποθηκεύεται ξεχωριστά από τα προγράμματα που εκμεταλλεύονται τα δεδομένα, προκύπτει ότι υπάρχει ανεξαρτησία και απομόνωση μεταξύ προγραμμάτων και δεδομένων. Αποτέλεσμα του γεγονότος αυτού είναι ότι κάθε μεταβολή στα δεδομένα δεν επηρεάζει τα προγράμματα εκμετάλλευσης και αντιστρόφως. Αυτό δεν συμβαίνει στα παραδοσιακή αποθήκευση αρχείων.
- **Υποστήριξη πολλαπλών όψεων (views) των Δεδομένων.** Το λογισμικό των Βάσεων Δεδομένων (DBMS) έχει την ικανότητα να δημιουργεί εικονικές όψεις των Δεδομένων (views) έτσι ώστε οι τελικοί χρήστες να έχουν προσπέλαση μόνον στα δεδομένα που τους ενδιαφέρουν. Αυτό δεν γίνεται στην παραδοσιακή επεξεργασία στοιχείων.
- **Ταυτόχρονη προσπέλαση των Δεδομένων από πολλούς χρήστες.** Αυτό εξασφαλίζεται με ειδικούς μηχανισμούς που διαθέτει το DBMS.

## Πλεονεκτήματα Βάσεων Δεδομένων.

Οι Βάσεις Δεδομένων παρουσιάζουν τα παρακάτω πλεονεκτήματα σε σύγκριση με την παραδοσιακή αποθήκευση στοιχείων (flat files):

- **Περιορισμός των πλεονασμάτων (Redundancy).** Τα δεδομένα στις Βάσεις Δεδομένων αποθηκεύονται σε ένα σημείο και είναι διαθέσιμα σε όλα τα προγράμματα, σε αντίθεση με την παραδοσιακή αποθήκευση που πρέπει να επαναλαμβάνονται σε κάθε πρόγραμμα επεξεργασίας.
- **Ύπαρξη διαδικασιών για αποθήκευση και επεξεργασία των δεδομένων.** Οι Βάσεις Δεδομένων διαθέτουν ειδικό λογισμικό τόσο για την αποθήκευση των δεδομένων, όσο και για την ταχεία εύρεση αυτών με την χρήση ευρετηρίων (indexes).
- **Ύπαρξη διαδικασιών για τον έλεγχο της χρήσης των Βάσεων Δεδομένων.** Οι Βάσεις Δεδομένων διαθέτουν κατάλληλο λογισμικό για τον έλεγχο της πρόσβασης με την χρήση ονομάτων χρήστη (username) και κωδικών ασφαλείας (password). Επίσης με την ανάθεση ρόλων και δικαιωμάτων στους χρήστες, ελέγχουν το εύρος των στοιχείων που είναι διαθέσιμα σε αυτούς.
- **Επιβολή διαδικασιών ελέγχου ορθότητας και ακριβείας στα δεδομένα.** Το λογισμικό των Βάσεων Δεδομένων ελέγχει την είσοδο των στοιχείων με την επιβολή κανόνων ελέγχου της ακεραιότητας τους. Επίσης μπορεί να αναπαραστήσει πολύπλοκες σχέσεις μεταξύ των δεδομένων, γεγονός που συμβάλει στην καλύτερη αξιοποίηση τους.
- **Υποστήριξη διαδικασιών λήψεως εφεδρικών αντιγράφων.** Το υποσύστημα τήρησης εφεδρικών αντιγράφων και ανάκαμψης (Database backup and recovery system) είναι υπεύθυνο για τη λήψη εφεδρικών αντιγράφων και ανάκαμψης μιας Βάσης Δεδομένων σε περίπτωση αστοχίας του λογισμικού ή του υλικού αυτής.

## Μοντέλα Δεδομένων.

Μοντέλο Δεδομένων είναι μια προσέγγιση περιγραφής της δομής μιας Βάσης Δεδομένων. Περιλαμβάνει ένα σύνολο από έννοιες που σκοπό έχουν να αποκρύψουν τις λεπτομέρειες της φυσικής οργάνωσης και αποθήκευσης των δεδομένων και προσπαθεί να περιγράψει με λογικό τρόπο, με τρόπο δηλαδή που να είναι κατανοητός στον χρήστη, την δομή μιας Βάσης Δεδομένων. Η δομή μιας Βάσης Δεδομένων περιλαμβάνει τις οντότητες των δεδομένων, τους τύπους δεδομένων, τις συσχετίσεις και τους περιορισμούς αυτών, όπως επίσης και τις διαδικασίες (operations) για την καταχώριση και τροποποίηση των δεδομένων.

**Κατηγορίες Μοντέλων Δεδομένων.** Διακρίνουμε τις εξής κατηγορίες μοντέλων δεδομένων:

- **Υψηλού επιπέδου (High Level)** τα οποία παρουσιάζουν την δομή της Βάσης Δεδομένων με μορφή που να είναι αντιληπτή στον χρήστη.
- **Χαμηλού επιπέδου (Low Level)** τα οποία περιγράφουν την φυσική αποθήκευση των δεδομένων και είναι κατανοητά μόνον από ειδικούς της πληροφορικής.
- **Εφαρμογής (Implementation level)** τα οποία αποτελούν το ενδιάμεσο επίπεδο μεταξύ των δύο πρώτων και περιέχουν έννοιες κατανοητές στους χρήστες αλλά δεν αποκρύβουν πλήρως την φυσική αποθήκευση των στοιχείων.

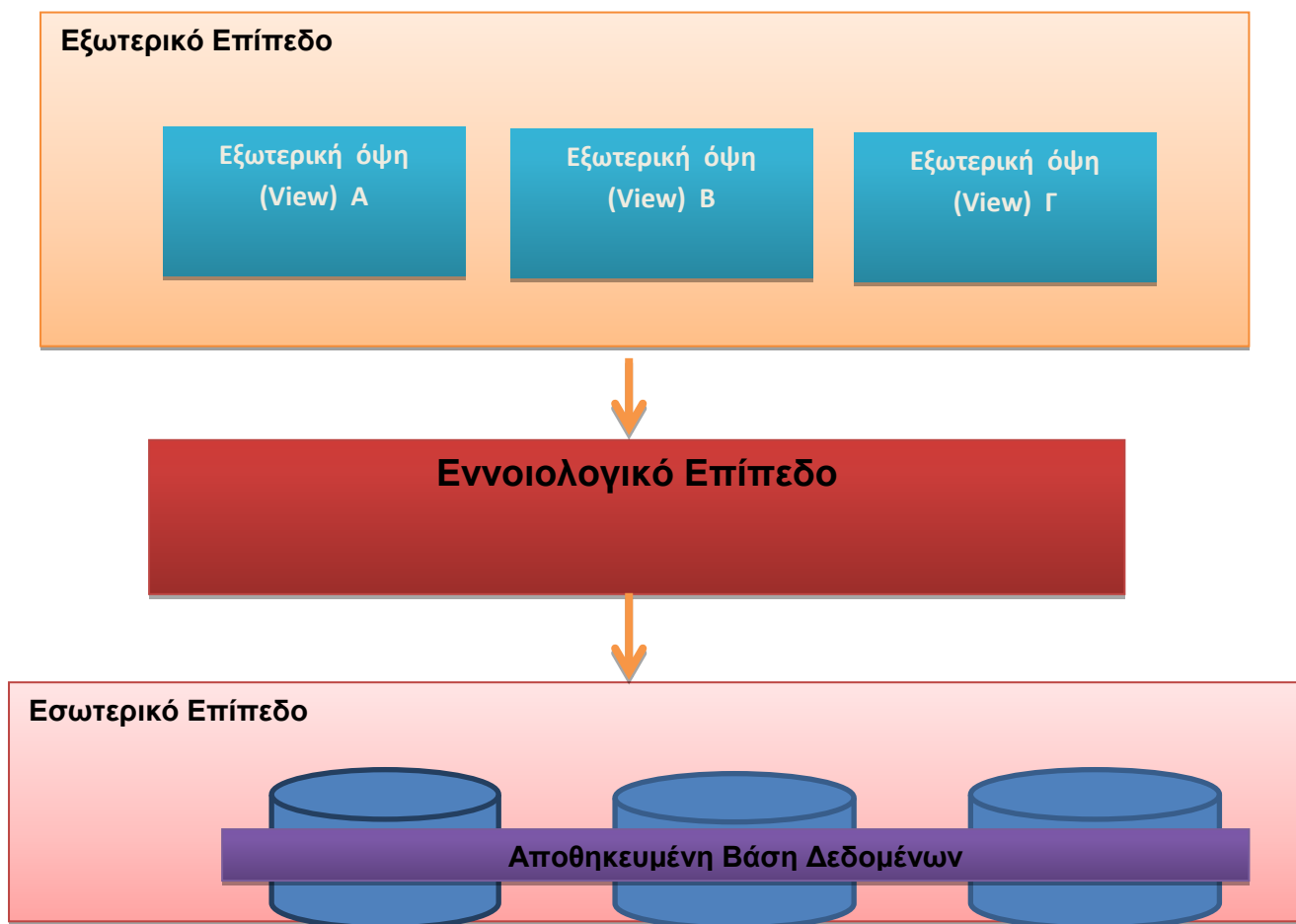


ΣΧΗΜΑ 1- 2 Σύστημα Διαχείρισης Βάσεων Δεδομένων

## Αρχιτεκτονική Βάσεων Δεδομένων.

Μια Βάση Δεδομένων χωρίζεται σε τρία επίπεδα.

- **Το εσωτερικό επίπεδο (internal level)** είναι το επίπεδο της φυσικής αποθήκευσης των στοιχείων. Χρησιμοποιεί ένα φυσικό μοντέλο δεδομένων (Low Level) και περιγράφει με λεπτομέρεια την αποθήκευση των δεδομένων και τους δρόμους (paths) προσπέλασης στην Βάση Δεδομένων.
- **Το εξωτερικό επίπεδο (external level)** είναι το επίπεδο των χρηστών. Είναι δηλαδή ο τρόπος που οι χρήστες έχουν πρόσβαση στα δεδομένα. Υπάρχουν πολλές διαφορετικές εξωτερικές όψεις (views), κάθε μια από τις οποίες αντιστοιχεί σε μια περισσότερο ή λιγότερο αφηρημένη αναπαράσταση κάποιου τμήματος της συνολικής Βάσης Δεδομένων. Στην περίπτωση αυτή χρησιμοποιείται ένα μοντέλο αναπαράστασης βασισμένο σε ένα εξωτερικό σχεδιασμό σχήματος (High Level).
- **Το εννοιολογικό επίπεδο (conceptual level)** είναι το ενδιάμεσο επίπεδο που ευρίσκεται μεταξύ του εσωτερικού και του εξωτερικού επιπέδου. Το εννοιολογικό σχήμα χρησιμοποιεί ένα implementation Level μοντέλο για την αναπαράστασή του.



ΣΧΗΜΑ 1- 3 Αρχιτεκτονική Βάσεων Δεδομένων

### Είδη Μοντέλων Δεδομένων.

Τα είδη των Μοντέλων Δεδομένων είναι:

- Το Ιεραρχικό.
- Το Δικτυωτό.
- Το Σχεσιακό.
- Τα αντικειμενοστραφή (Object – Oriented).

Τα πρώτα δύο χρησιμοποιήθηκαν πολύ στο παρελθόν και βασίσθηκαν στην δομές εγγράφων. Για τον λόγο αυτόν ονομάστηκαν και βασισμένα σε εγγραφές (record

based). Τα αντικειμενοστραφή χρησιμοποιηθήκαν σαν υψηλού επιπέδου εννοιολογικά μοντέλα για να περιγράψουν πολύπλοκες δομές με αμφίβολα αποτελέσματα.

## Τα Εννοιολογικά Μοντέλα.

Είναι υψηλού επιπέδου μοντέλα που εισάγουν τις παρακάτω έννοιες.

- **Οντότητα** (Entity). Παρουσιάζει ένα αντικείμενο ή μια έννοια που υπάρχει στον πραγματικό κόσμο όπως πχ EMPLOYEES (ΥΠΑΛΛΗΛΟΙ) που είδαμε προηγουμένως.
- **Γνώρισμα** (attribute) Είναι ένα χαρακτηριστικό της οντότητας πχ το F\_NAME (Όνομα) του Υπαλλήλου
- **Σχέση** (relationship). Αντιπροσωπεύει την αλληλεπίδραση της μιας οντότητας σε μια άλλη.
- **Σχήμα της Βάσης** (Database Schema). Ονομάζεται η περιγραφή της Βάσης Δεδομένων και συνήθως μετά την δημιουργία του δεν αλλάζει συχνά. Η αναπαράσταση του σχήματος λέγεται διάγραμμα σχήματος.

## Το Σχεσιακό Μοντέλο (relational model).

### Γενικά Στοιχεία.

Προτάθηκε από τον EE Codd, έναν ερευνητή της IBM το 1970. Αποτελεί μια μεγάλη καινοτομία διότι ξεχώρισε την λογική επεξεργασία των στοιχείων μιας Βάσης Δεδομένων από την φυσική αποθήκευση αυτών. Σε αντίθεση με τα άλλα μοντέλα δεδομένων – το ιεραρχικό και το δικτυακό – τα οποία χρησιμοποιούσαν την φυσική τοποθεσία των στοιχείων για να ορίσουν σχέσεις μεταξύ τους, το σχεσιακό μοντέλο ορίζει ότι τα στοιχεία σε ένα πίνακα το μόνον που πρέπει να γνωρίζουν είναι το όνομα του άλλου πίνακα και την τιμή που θα ορίσουν την σύνδεση.

Στο σχεσιακό μοντέλο, οι πληροφορίες διασπώνται σε μικρότερες συλλογές από στοιχεία που ονομάζονται **πίνακες (tables)**. Επίσης εισάγει και διάφορους τελεστές (operators) που ενεργούν πάνω στα συσχετιζόμενα στοιχεία (πχ μια ένωση join) και παράγουν άλλα αντικείμενα. Επιπρόσθετα το μοντέλο ορίζει μια σειρά από ελέγχους για να εξασφαλίσει την ακεραιότητα και την ακρίβεια των στοιχείων. Ο Codd πρότεινε 12 κανόνες που εξασφαλίζουν ότι μια Βάση Δεδομένων ακολουθεί το σχεσιακό μοντέλο. Σήμερα καμία Βάση Δεδομένων που έχει σχεδιασθεί με το σχεσιακό μοντέλο, δεν έχει συμμορφωθεί και με τους 12 κανόνες διότι τότε θα λειτουργούσε πολύ αργά, αλλά καλόν είναι να συμμορφώνεται με τους περισσότερους από αυτούς. Η ουσία του μοντέλου είναι ότι τα δεδομένα οργανώνονται υπό μορφή συνόλων που αναπαριστώνται υπό μορφή πινάκων δύο διαστάσεων με σειρές (rows) και στήλες (columns). Τέλος το λογισμικό της Διαχείρισης Βάσεων Δεδομένων (Database Management System DBMS), στα σχεσιακά μοντέλα ονομάζεται Σχεσιακό Λογισμικό Βάσεων Δεδομένων (RDBMS).

### ΠΙΝΑΚΑΣ REGION

Σειρά (Row)	REGION_ID	REGION_NAME
1	1	EUROPE
2	2	AMERICA
3	3	ASIA
4	4	AFRICA

Στήλη (Column)


ΣΧΗΜΑ 1- 4 Σειρά και Στήλη Πίνακα

Στο παράδειγμα της Βάσης Δεδομένων που δείξαμε προηγουμένως, είδαμε ότι ο πίνακας REGION αποθηκεύει πληροφορίες για τις περιοχές και ο πίνακας COUNTRY για τις χώρες. Και οι δύο πίνακες έχουν ένα κοινό πεδίο το REGION\_ID. Με το πεδίο αυτό ορίζεται μια σχέση (relationship) μεταξύ των πινάκων. Και οι δύο



πίνακες αποθηκεύονται σε μια Βάση Δεδομένων, σε έναν υπολογιστή. Η φυσική θέση των πινάκων είναι άγνωστη, μόνον το όνομα τους είναι γνωστό.

REGION	
REGION_ID	REGION_NAME
1	EUROPE
2	AMERICA
3	ASIA
4	AFRICA




COUNTRY		
COUNTRY_ID	COUNTRY_NAME	REGION_ID
1	ARGENTINA	2
2	AUSTRIA	1

ΣΧΗΜΑ 1- 5 Δευτερεύων Κλειδί

Κάθε σειρά (εγγραφή) από δεδομένα πρέπει να έχει μια στήλη (ή συνδυασμός στηλών) που **μοναδικά** θα ορίζουν την σειρά αυτή ανάμεσα στις υπόλοιπες σειρές του πίνακα. Η στήλη αυτή ονομάζεται κύριο κλειδί (**primary key**). Κάθε πίνακας στην θεωρία του σχεσιακού μοντέλου πρέπει να έχει ένα κύριο κλειδί.

Στον πίνακα REGION, η στήλη που ορίζει μοναδικά κάθε γραμμή του πίνακα, είναι το REGION\_ID αφού παίρνει μοναδικές τιμές. Η στήλη αυτή είναι το κύριο κλειδί του πίνακα.



REGION_ID	REGION_NAME
1	EUROPE
2	AMERICA
3	ASIA
4	AFRICA

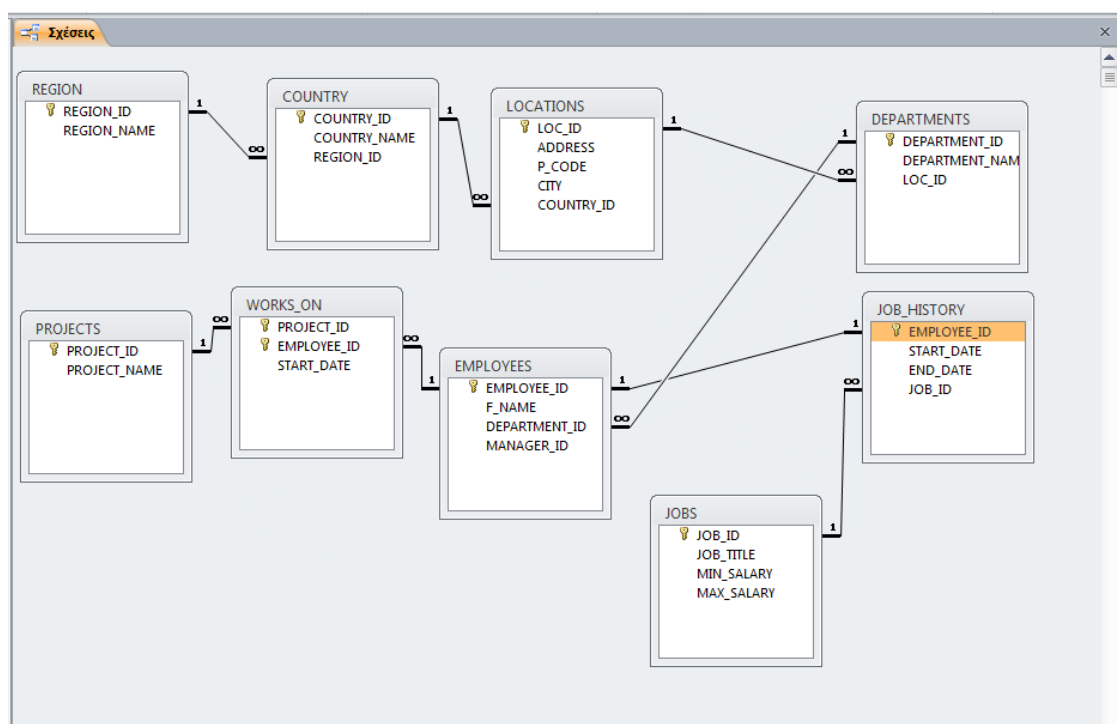
ΣΧΗΜΑ 1- 6 Παράδειγμα Κύριο Κλειδί Πίνακα

Όταν οι πίνακες σχετίζονται μεταξύ τους, γεγονός που αποτελεί τον ακρογωνιαίο λίθο του σχεσιακού μοντέλου, η τιμή του κυρίου κλειδιού ενός πίνακα, τίθεται στην στήλη ενός άλλου πίνακα, πχ η στήλη REGION\_ID που είναι κύριο κλειδί στον πίνακα REGION, τίθεται σαν στήλη και στον πίνακα COUNTRY. Στην περίπτωση αυτή η στήλη REGION\_ID αποτελεί για τον πίνακα COUNTRY **ξένο κλειδί (foreign key)**. Για να υπάρχουν οι τιμές της στήλης REGION\_ID στον πίνακα COUNTRY πρέπει οι αντίστοιχες τιμές να υπάρχουν και στην στήλη REGION\_ID του πίνακα REGION, σε διαφορετική περίπτωση η σχέση των δύο πινάκων δεν μπορεί να υπάρξει. Στο σχεσιακό μοντέλο η γραμμή ονομάζεται εγγραφή (tuple) και η στήλη πεδίο.

**Είδη Σχέσεων.** Οι σχέσεις που αναπτύσσονται μεταξύ των πινάκων είναι:

- **Σχέση 1:1** (ένα προς ένα), μεταξύ δύο πινάκων σημαίνει ότι μια εγγραφή του ενός πίνακα συσχετίζεται με μια εγγραφή του άλλου πίνακα πχ οι πίνακες EMPLOYEES και JOB\_HISTORY έχουν σχέση μια προς μια με κοινό πεδίο το EMPLOYEE\_ID. Συμβολίζεται με 1 -> 1.
- **Σχέση 1:N** (ένα προς πολλά), μεταξύ δύο πινάκων σημαίνει ότι μια εγγραφή του ενός πίνακα, σχετίζεται με πολλές εγγραφές στον άλλο πίνακα πχ οι πίνακες DEPARMENTS και EMPLOYEES έχουν σχέση ένα προς πολλά με κοινό πεδίο το DEPARTMENT\_ID. Εννοιολογικά αυτό σημαίνει ότι ένας EMPLOYEE ανήκει σε ένα DEPARTMENT αλλά ένα DEPARTMENT έχει πολλούς EMPLOYEES. Αποτελεί την πλέον συνήθη σχέση και συμβολίζεται με 1 -> ∞.
- **Σχέση M:N** (πολλά προς πολλά), μεταξύ δύο πινάκων σημαίνει ότι μια εγγραφή του ενός πίνακα σχετίζεται με πολλές εγγραφές του άλλου πίνακα, αλλά και μια εγγραφή του δεύτερου πίνακα σχετίζεται με πολλές εγγραφές του πρώτου πίνακα πχ οι πίνακες EMPLOYEES και PROJECTS σχετίζονται με σχέση πολλά προς πολλά που σημαίνει ότι ένας EMPLOYEE συμμετέχει σε πολλά PROJECTS , αλλά και ένα PROJECT έχει πολλούς EMPLOYEES. Συμβολίζεται με M:N δηλ ∞ -> ∞.

Στα σχεσιακά μοντέλα αυτή η σχέση δεν μπορεί να αναπαρασταθεί. Για τον λόγο αυτό διαχωρίζεται σε δύο σχέσεις 1:N με ένα τρίτο ενδιαμέσο πίνακα που θα έχει ως κύριο κλειδί (primary key) τα δύο κύρια κλειδιά των άλλων πινάκων. Στην περίπτωσή μας δημιουργείται ένας πίνακας WORKS\_ON με σχέση 1:N προς τους πίνακες EMPLOYEES (κοινό πεδίο το EMPLOYEE\_ID) και PROJECTS (κοινό πεδίο το PROJECT\_ID). Σχηματικά οι σχέσεις μεταξύ των πινάκων εμφανίζονται παρακάτω:



ΣΧΗΜΑ 1- 7 Πίνακας Σχέσεων Πινάκων

## Κανονικοποίηση Βάσεων Δεδομένων.

Αναφέρθηκε προηγουμένως ότι για να ακολουθεί μια Βάση Δεδομένων το σχεσιακό μοντέλο πρέπει να συμμορφώνεται με 12 κανόνες. Η διαδικασία αυτή της συμμόρφωσης ονομάζεται κανονικοποίηση. Σήμερα αρκεί ένα σχήμα Βάσεων Δεδομένων να ακολουθεί τουλάχιστον τους τρεις 3 πρώτους κανόνες, για να θεωρείται κανονικοποιημένο.

**Οι στόχοι της κανονικοποίησης είναι:**

- Εξάλειψη διπλών πληροφοριών.
- Να αντιμετωπίζει μελλοντικές αλλαγές στη δομή των Βάσεων.
- Να ελαχιστοποιεί την επίδραση των δομικών αλλαγών της Βάσης Δεδομένων σε εφαρμογές των χρηστών που έχουν πρόσβαση στα δεδομένα.
- **Πρώτη Κανονική Μορφή.** Η Πρώτη Κανονική Μορφή, απαιτεί ότι οι πίνακες της Βάσης Δεδομένων είναι επίπεδοι και δεν περιέχουν επαναλαμβανόμενες ομάδες δεδομένων. Κάθε πεδίο περιέχει μια τιμή (ονομάζεται ατομική τιμή) και οι πίνακες είναι δύο διαστάσεων. Επιπρόσθετα κάθε πίνακα έχει ένα κύριο κλειδί (όπως περιγράφεται προηγουμένως) του οποίου η τιμή δεν μπορεί να είναι Null (απροσδιόριστο).
- **Δεύτερη Κανονική Μορφή.** Η Δεύτερη Κανονική Μορφή απαιτεί ότι ο πίνακας είναι σε Πρώτη Κανονική Μορφή και επιπλέον ότι όλα τα πεδία ενός πίνακα θα εξαρτώνται μόνον από το κύριο κλειδί και όχι από άλλο πεδίο του πίνακα. Παράδειγμα στον πίνακα EMPLOYEES υπάρχει το DEPARTMENT\_ID για να προσδιορίσει το DEPARTMENT κάθε EMPLOYEE. Επομένως δεν μπορεί στον πίνακα EMPLOYEES να περιέχεται το πεδίο DEPARTMENT\_NAME το οποίο εξαρτάται από το DEPARTMENT\_ID και όχι από το EMPLOYEE\_ID που είναι το κύριο κλειδί.
- **Τρίτη Κανονική Μορφή.** Η Τρίτη Κανονική Μορφή απαιτεί ότι ο πίνακας είναι σε Δεύτερη Κανονική Μορφή και επιπλέον ότι κάθε πεδίο του πίνακα που δεν ανήκει στο πρωτεύον κλειδί, εξαρτάται από ολόκληρο το πρωτεύον κλειδί και όχι μέρος αυτού. Παράδειγμα στον πίνακα WORKS\_ON το πεδίο START\_DATE εξαρτάται από ολόκληρο το κύριο κλειδί (PROJECT\_ID, EMPLOYEE\_ID) αφού η START\_DATE είναι συνυφασμένη και με το PROJECT και με τον EMPLOYEE. Αντίθετα δεν μπορεί στον πίνακα αυτό να υπάρχει το πεδίο F\_NAME (όνομα EMPLOYEE), διότι αυτό εξαρτάται μόνον από το EMPLOYEE\_ID που είναι μέρος του κυρίου κλειδιού και όχι από ολόκληρο το κύριο κλειδί (PROJECT\_ID, EMPLOYEE\_ID).

## 2. Εισαγωγή στην Αρχιτεκτονική της ORACLE.

### Σκοπός και στόχοι της ενότητας.

Σκοπός αυτής ενότητας είναι η κατανόηση των βασικών όρων και εννοιών της σχεσιακής Βάσης Δεδομένων ORACLE και στην συνέχεια του λογισμικού διαχείρισης αυτής Oracle Enterprise Manager.

Έτσι, μετά το τέλος της ενότητας, οι επιμορφωμένοι θα είναι σε θέση να γνωρίζουν τα παρακάτω αντικείμενα:

- 1 Ορισμός Oracle.
- 2 Τα χαρακτηριστικά του Oracle Server.
- 3 Μέθοδοι Πρόσβασης στον Oracle Server.
- 4 Αρχιτεκτονική του Oracle Server.
- 5 Δομές Μνήμης.
- 6 Αρχιτεκτονική διαδικασιών (Process Architecture).
- 7 Αρχιτεκτονική Διαδικασιών στο παρασκήνιο (Background Processes).
- 8 Αρχιτεκτονική Αποθήκευσης Βάσης Δεδομένων.
- 9 Ο Oracle Enterprise Manager.

### Ορισμός Oracle.

Η Oracle είναι ένα Σύστημα Διαχείρισης Σχεσιακών Βάσεων Δεδομένων το οποίο είναι γνωστό με το όνομα Oracle Server. Αποτελείται από την Βάση Δεδομένων (Oracle Database) και το περιβάλλον – στιγμιότυπο Διαχείρισης των δομών της Oracle που είναι γνωστό ως Oracle instance.

### Τα χαρακτηριστικά του Oracle Server.

- Διαχειρίζεται μεγάλο όγκο δεδομένων.
- Επιτρέπει την ταυτόχρονη πρόσβαση στα δεδομένα, πολλών χρηστών ταυτοχρόνως.

- Διαχειρίζεται τις διάφορες λειτουργίες της Βάσης Δεδομένων με αξιόπιστο τρόπο, μεγάλη ταχύτητα και επιδόσεις (performance).
- Εμποδίζει την πρόσβαση μη εξουσιοδοτημένων χρηστών, διαθέτοντας τους κατάλληλους μηχανισμούς.
- Διαθέτει ειδικές διαδικασίες για την λήψη αντιγράφων ασφαλείας (Backup) και ανάκαμψης (Recovery), σε περίπτωση αστοχίας υλικού ή λογισμικού.

### Μέθοδοι Πρόσβασης στον Oracle Server.

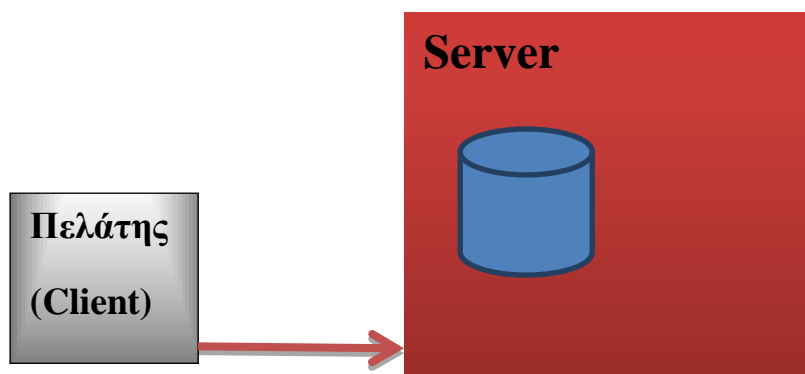
Οι περιπτώσεις πρόσβασης ενός χρήστη στον Oracle Server είναι οι παρακάτω:

- **Ο χρήστης και ο Oracle Server βρίσκονται στον ίδιο υπολογιστή (stand alone).** Στην περίπτωση αυτή ο χρήστης κάνει ασφαλή είσοδο στον τοπικό υπολογιστή χρησιμοποιώντας τις διαδικασίες του Λειτουργικού Συστήματος αυτού και στην συνέχεια χρησιμοποιεί μια εφαρμογή για να αποκτήσει πρόσβαση στον Oracle Server. Η επικοινωνία της τοπικής εφαρμογής με τον Oracle Server, γίνεται με λογισμικό του τελευταίου.
- **Η αρχιτεκτονική πελάτη – διακομιστή (Client – Server).** Η αρχιτεκτονική αυτή έχει δύο μέρη. Το ένα περιλαμβάνει μια εφαρμογή που ευρίσκεται στον τοπικό υπολογιστή του χρήστη (Client - Front-end) και το δεύτερο περιλαμβάνει τον Oracle Server, που βρίσκεται σε άλλο υπολογιστή (Server-Back-end). Η εφαρμογή συνδέεται με τον Oracle Server μέσω ειδικού πρωτοκόλλου δικτύου που διατίθεται από την Oracle. Ο χρήστης μέσω της εφαρμογής, στέλνει μια αίτηση για δεδομένα στον Oracle Server. Αυτός λαμβάνει την αίτηση, την ελέγχει και εκτελεί τις κατάλληλες διαδικασίες προκειμένου να δημιουργήσει μια απάντηση με τα δεδομένα αυτά, τα οποία και τα στέλνει στην εφαρμογή του χρήστη. Αυτή μόλις λάβει την απάντηση παρουσιάζει τα δεδομένα στον χρήστη. Η εφαρμογή αυτή μπορεί να είναι και ένας φυλλομετρητής (web browser) του διαδικτύου. Όπως αναφέρθηκε και

προηγουμένως, ο Oracle Server μπορεί να διαχειρισθεί απρόσκοπτα και με πολύ μεγάλη ταχύτητα, τεράστιο αριθμό παρομοίων αιτήσεων ταυτόχρονα.

- **Η πολυεπίπεδη αρχιτεκτονική. (multitier architecture).** Στην περίπτωση αυτή, υπάρχει και ένα επιπλέον επίπεδο μεταξύ του υπολογιστή του χρήστη και του υπολογιστή που υπάρχει ο Oracle Server. Το επίπεδο αυτό ονομάζεται ενδιάμεσο επίπεδο και εδώ υπάρχει ένα Λογισμικό εφαρμογής (Application Server). Ο Application Server λαμβάνει την αίτηση για δεδομένα από τον χρήστη (Client), ενεργεί μερικώς σε αυτή, έτσι ώστε να απαλλάξει τον Oracle Server από τον κύριο φόρτο της επεξεργασίας των δεδομένων αυτών. Επιπλέον ο Application Server προσθέτει και ένα επιπλέον επίπεδο ασφαλείας για την πρόσβαση στον Oracle Server.

Η επικοινωνία μεταξύ των επιπέδων γίνεται μέσω του ειδικού πρωτοκόλλου που προσφέρεται από την Oracle, όπως αναφέρθηκε προηγουμένως.



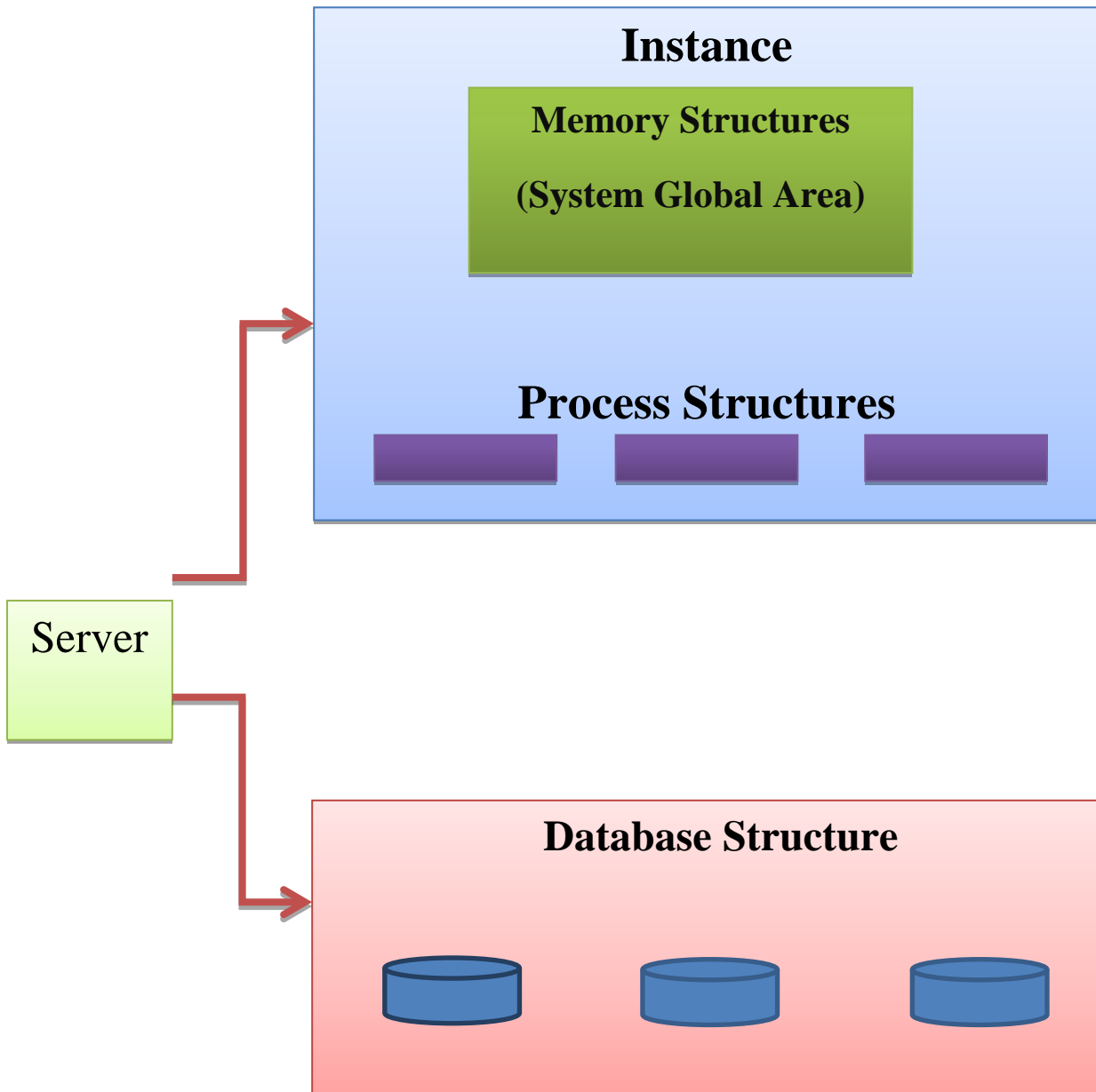
**ΣΧΗΜΑ 2- 1** Μέθοδοι Πρόσβασης στον Oracle Server**Αρχιτεκτονική του Oracle Server.**

Η Oracle Database αποτελείται από τις δομές αποθήκευσης (storage structures) που διακρίνονται σε φυσικές δομές (physical structures) και τις λογικές δομές. (logical structures). Οι φυσικές δομές διαχωρίζονται από τις λογικές και για τον λόγο αυτό μπορεί να γίνεται απρόσκοπτη διαχείριση των φυσικών δομών χωρίς να επηρεάζονται οι λογικές δομές. Το Oracle instance αποτελείται από τις δομές μνήμης (memory Structures) και διεργασιών στο παρασκήνιο (background processes). Οι δομές αυτές αφορούν το συγκεκριμένο instance. Όταν ξεκινά το instance τότε γίνονται οι εξής ενέργειες:

- Δημιουργούνται στην κυρία μνήμη (RAM) του υπολογιστή, όπου βρίσκεται η εγκατάσταση της Oracle, οι δομές μνήμης οι οποίες καταλαμβάνουν ένα χώρο ο οποίος ονομάζεται System Global Area και ένα χώρο μνήμης οποίος ονομάζεται Program Global Area.
- Εκκινούν ταυτόχρονα οι διεργασίες παρασκηνίου background processes.

Μετά τις ενέργειες αυτές, ο Oracle Server, συνδέει το instance με μια συγκεκριμένη Database και αρχίζει το ανέβασμα αυτής. Η διαδικασία αυτή ονομάζεται mounting the Database και ολοκληρώνεται με το άνοιγμα (open) της, έτσι ώστε να αρχίσει η πρόσβαση σε αυτή από τους εξουσιοδοτημένους χρήστες.





ΣΧΗΜΑ 2- 2 Αρχιτεκτονική του Oracle Server

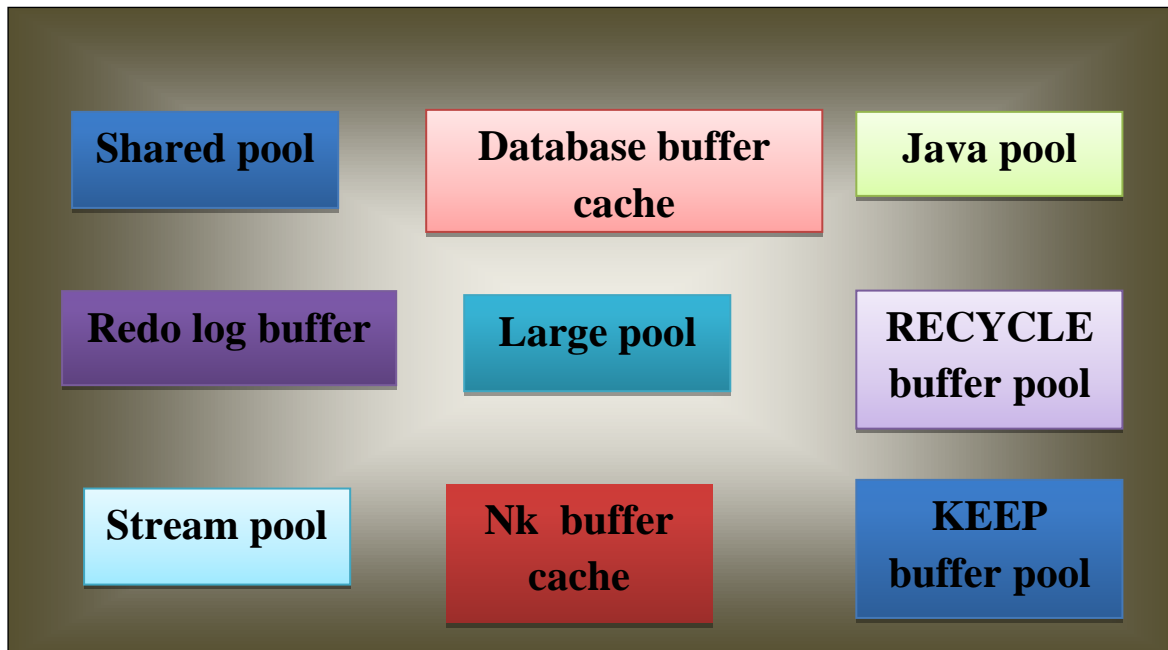
## Δομές Μνήμης.

Οι βασικές μορφές μνήμης που συνδέονται με κάθε instance είναι:

- **System Global Area (SGA).** Είναι ένα σύνολο από κοινές δομές μνήμης (shared memory structures) οι οποίες περιέχουν δεδομένα και πληροφορίες ελέγχου (control information) για ένα Oracle Database Instance. Η SGA είναι κοινή για όλους τους servers και τις background processes.
- **Program Global Area (PGA).** Είναι περιοχή μνήμης που περιέχει δεδομένα και πληροφορίες ελέγχου που αφορά αποκλειστικά έναν server ή μια background processes. Δεν αποτελεί κοινή περιοχή μνήμης όπως η SGA.

Η SGA αποτελείται από τις εξής περιοχές μνήμης:

- **Database buffer cache :** Περιέχει δεδομένα από την Βάση Δεδομένων.
- **Redo log buffer:** Περιέχει πληροφορίες ανάκαμψης της Βάσης Δεδομένων (redo log), μέχρι την εγγραφή τους στα φυσικά αρχεία ανάκαμψης (redo log files), στο δίσκο του υπολογιστή της Oracle
- **Shared pool:** Περιέχει διάφορες πληροφορίες που διαμοιράζονται μεταξύ των χρηστών.
- **KEEP buffer cache:** Αποτελεί έναν ειδικό τύπο Database buffer cache που περιέχει πληροφορίες δεδομένων για μεγάλη χρονική περίοδο.
- **RECYCLE buffer cache:** Αποτελεί έναν ειδικό τύπο Database buffer cache που περιέχει δεδομένα τα οποία πρέπει να απομακρυνθούν άμεσα.
- **nK buffer cache:** Αποτελούν ειδικό τύπο Database buffer cache τα οποία έχουν block size μεγαλύτερο από το προκαθορισμένο block size της Database.
- **Large pool:** Αποτελεί προαιρετικό τύπο μνήμης για την κάλυψη των αναγκών μεγάλων διεργασιών (processes), όπως η λήψη εφεδρικών αρχείων (backup) και ανάκαμψης (recovery).
- **Java pool:** Αποτελεί χώρο μνήμης που χρησιμοποιείται από τον Java κώδικα και για την Java Virtual Machine (JVM).
- **Streams pool:** Αποτελεί χώρο μνήμης που χρησιμοποιείται από τα Oracle Streams.



Σχήμα 2- 3 Oracle System Global Area

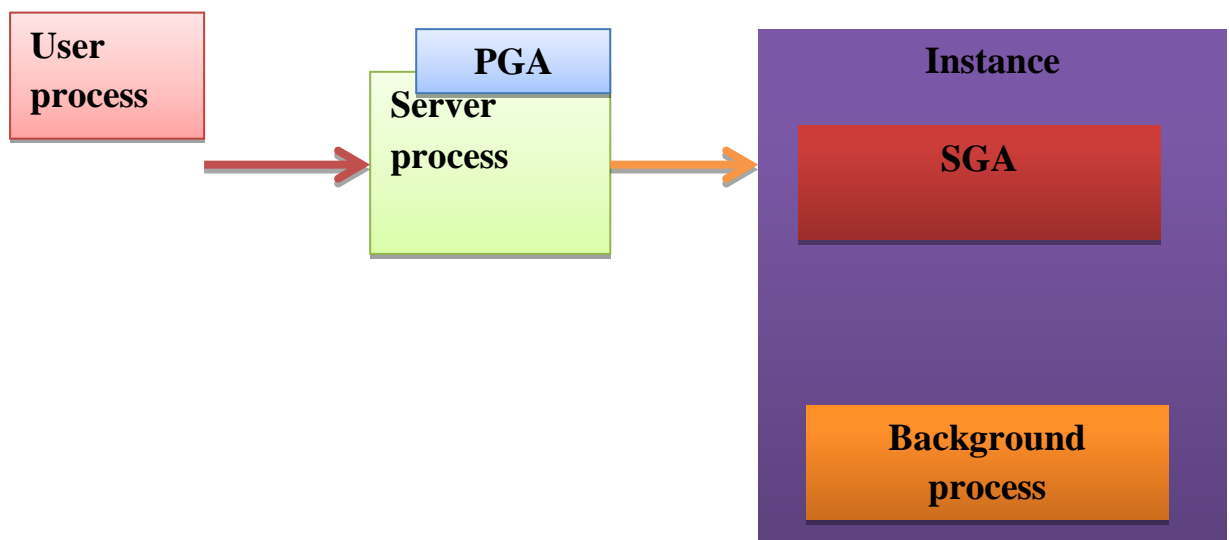
**Program Global Area.** Η PGA είναι μια περιοχή μνήμης η οποία περιέχει δεδομένα και πληροφορίες ελέγχου για κάθε server process (διεργασία). Μια server process εξυπηρετεί μια αίτηση από ένα χρήστη (client). Κάθε server process έχει την δική της αποκλειστική PGA, η οποία δημιουργείται όταν ξεκινά μια server process.

### Αρχιτεκτονική διαδικασιών (Process Architecture).

Η Oracle περιλαμβάνει δύο είδη διεργασιών (processes):

- **User processes (διεργασίες χρηστών).** Αυτές δημιουργούνται όταν οι χρήστες ξεκινούν μια αίτηση σύνδεσης με τον Oracle server. Η αίτηση αυτή μπορεί να είναι μια εφαρμογή ή ένα Oracle tool όπως SQL\*Plus ή ο Enterprise Manager.
- **Database processes (διεργασίες Βάσης Δεδομένων)** Αυτές δημιουργούνται από τον Oracle server στις εξής περιπτώσεις:

- Όταν οι χρήστες ξεκινούν τις διεργασίες (user processes) για σύνδεση προς την Βάση Δεδομένων (database) μέσω των εφαρμογών όπως αναφέρθηκε πριν.
- Όταν ξεκινά το instance και αφορούν μια σειρά από διεργασίες παρασκήνιου (background processes) προκειμένου να δημιουργηθούν οι κατάλληλες δομές μνήμης που με την συνεργασία με το Λειτουργικό Σύστημα, θα διαχειρισθούν την μεταφορά δεδομένων στην Βάση Δεδομένων (I/O write).
- Την εκκίνηση προγραμμάτων (Daemon /Application processes) για την εξυπηρέτηση διαφόρων αναγκών σύνδεσης των χρηστών με την Βάση Δεδομένων (network listeners).

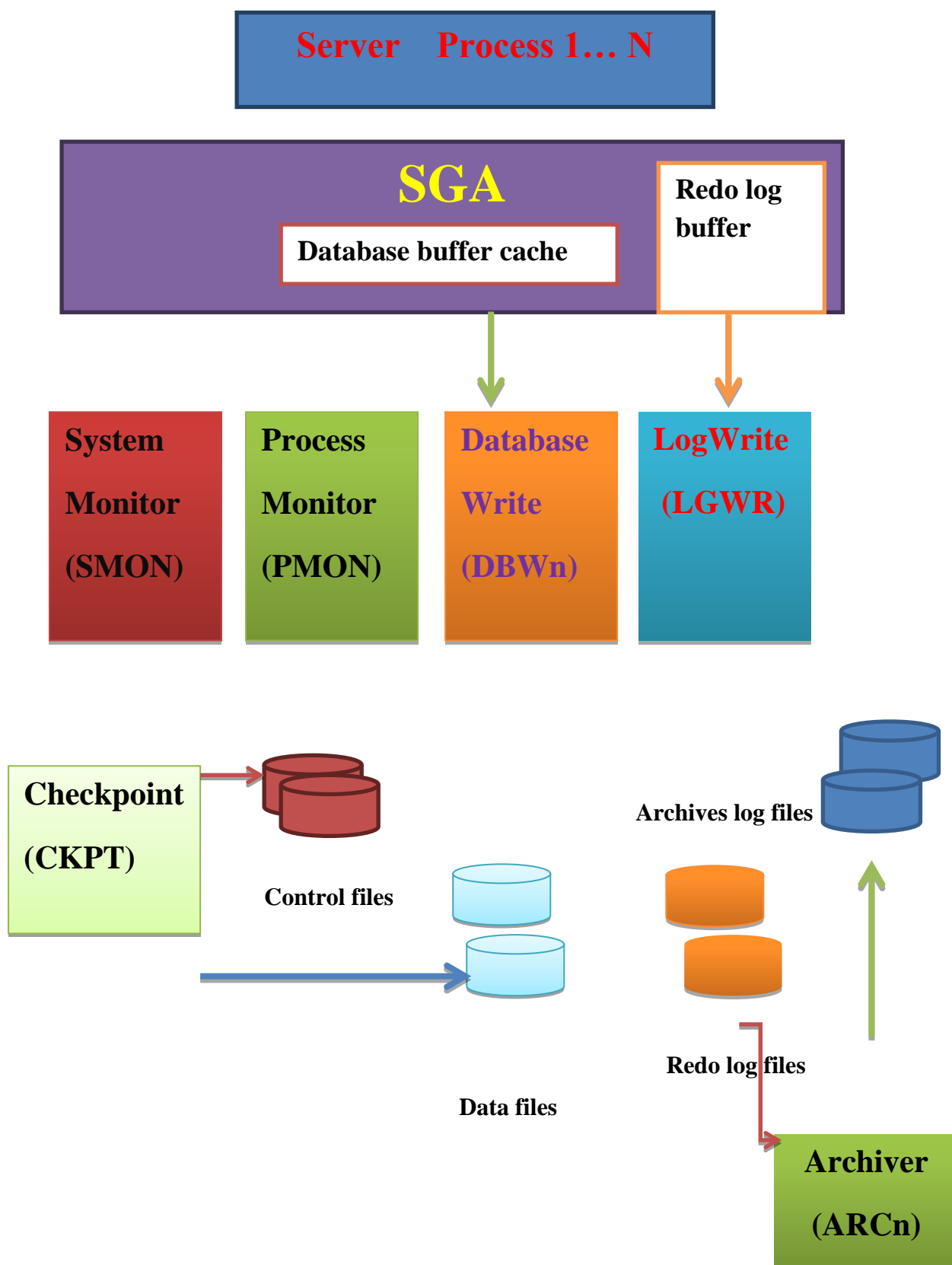


Σχήμα 2- 4 Αρχιτεκτονική διαδικασιών (Process Architecture)

### Αρχιτεκτονική Διαδικασιών στο παρασκήνιο (Background Processes).

Οι κυριότερες background processes είναι:

- **Database Writer (DBWn)** . Μεταφέρει τα blocks δεδομένων που έχουν μεταβληθεί από τον Database buffer cache στα data files (θα περιγραφούν παρακάτω) στην Βάση Δεδομένων.
- **Log Writer (LGWR)**. Μεταφέρει τα redo log στην Βάση Δεδομένων.
- **Checkpoint (CKPT)**. Ενημερώνει τα Data files και τα control files για να δείξει το πιο πρόσφατο Checkpoint (σημείο ελέγχου).
- **System Monitor (SMON)**. Ενεργεί ανάκαμψη της Oracle μετά από αστοχία (crash recovery).
- **Process Monitor (PMON)**. Ενεργεί ανάκαμψη μιας διεργασίας χρήστη (user process) σε περίπτωση αστοχίας.



Σχήμα 2- 5 Background Processes

## Αρχιτεκτονική Αποθήκευσης Βάσης Δεδομένων (Database Storage Architecture).

Η αρχιτεκτονική Αποθήκευσης της Βάση Δεδομένων της Oracle (Database Storage Architecture) , χωρίζεται σε δύο κατηγορίες:

- Η φυσική δομή (Physical Database Structure).
- Η λογική δομή (Logical Database Structure).

Η φυσική δομή της Βάσης Δεδομένων της Oracle περιλαμβάνει:

- **Data files (αρχεία δεδομένων).** Περιλαμβάνουν τα αρχεία δεδομένων των χρηστών, των εφαρμογών όπως επίσης το λεξικό δεδομένων (data dictionary) και τα αρχεία περιγραφής της Βάσης (meta data).
- **Control files (αρχεία ελέγχου).** Περιλαμβάνουν τα αρχεία περιγραφής της Βάσης Δεδομένων (data about the database). Είναι σημαντικά για την Βάση Δεδομένων, διότι χωρίς αυτά δεν μπορεί να υπάρξει πρόσβαση στα Data files. Επίσης περιέχουν πληροφορίες για τα αρχεία αντιγράφων ασφαλείας.
- **Online redo log files (αρχεία καταγραφής των μεταβολών της πρόσβασης στην Βάση Δεδομένων).** Στα αρχεία αυτά καταγράφονται τα απαραίτητα στοιχεία για την ανάκαμψη της Βάσης Δεδομένων σε περίπτωση αστοχίας. Στην περίπτωση αυτή ο database server χάνει την επαφή με τα Data files και το instance χρησιμοποιεί τις πληροφορίες από αυτά τα αρχεία για άμεση ανάκαμψη.

Επιπρόσθετα στην φυσική δομή της Βάσης Δεδομένων ανήκουν και τα εξής αρχεία:

- **Parameter file (αρχεία παραμέτρων).** Αποθηκεύουν πληροφορίες για την παραμετροποίηση του instance κατά την εκκίνηση της Βάσης.

- **Password file (αρχεία κωδικών).** Επιτρέπουν στους χρήστες της Βάσης Δεδομένων που χρησιμοποιούν τους ρόλους SYSDBA, SYSOPER, και SYSASM, να συνδέονται με αυτή και να έχουν δυνατότητες διαχειριστή (administrator).
- **Backup files** (αρχεία αντιγράφων ασφαλείας) . Χρησιμεύουν για την δημιουργία αντιγράφων ασφαλείας των δεδομένων, σε περιπτώσεις είτε αστοχίας του υλικού (Hardware) είτε καταστροφής ή διαγραφής ενός αρχείου από λάθος ενέργειας.
- **Archived redo log files** (αρχεία ιστορικής αποθήκευσης δεδομένων). Χρησιμεύουν για την αποθήκευση των redo log files για ιστορικούς λόγους (μεγάλη χρονική περίοδο).
- **Trace files** (αρχεία καταγραφής ενεργειών). Κάθε διεργασία παρασκηνίου (background process) καταγράφει τις ενέργειες που εκτελεί σε αντίστοιχο αρχείο – trace file. Κατά αυτόν τον τρόπο σε περίπτωση αστοχίας της υπόψη background process, είναι δυνατή η καταγραφή και η ανίχνευση του σφάλματος που την προκάλεσε.
- **Alert log file** (αρχείο καταγραφής ειδοποιήσεων). Στο αρχείο αυτό καταγράφονται με χρονολογική σειρά μηνύματα και αναφορές λαθών και αστοχιών της Βάσης Δεδομένων, τα οποία πρέπει συνεχώς να εξετάζονται.

Η λογική δομή (logical structure) της Βάσης Δεδομένων της Oracle, περιλαμβάνει τα tablespaces.

- **Tablespaces** Η Βάση Δεδομένων χωρίζεται σε λογικές μονάδες αποθήκευσης που ονομάζονται tablespaces. Αυτά μπορεί να είναι ένα ή περισσότερα. Κάθε tablespace περιλαμβάνει ένα ή περισσότερα datafiles. Κάθε Βάση περιλαμβάνει ένα SYSTEM tablespace και ένα SYSAUX tablespace, τα οποία δημιουργούνται αυτόματα με την δημιουργία της Βάσης. Σε ένα tablespace μπορεί να είναι δυνατή η πρόσβαση (online accessible) ή να όχι (offline accessible) Το SYSTEM tablespace είναι πάντα προσβάσιμο, όταν η Βάση είναι ανοικτή (open). Σε αυτό υπάρχουν οι πίνακες που αποθηκεύουν τα στοιχεία για την λειτουργία της Βάσης όπως το λεξικό δεδομένων. Το



SYSAUX tablespace είναι ένα βοηθητικό tablespace στο SYSTEM και χρησιμεύει για την αποθήκευση στοιχείων για την ομαλή λειτουργία της Βάσης Δεδομένων.

- **Segments – Extents – Blocks.** Τα αντικείμενα της Βάσης Δεδομένων όπως οι πίνακες και οι δείκτες, αποθηκεύονται σαν segments στα tablespaces. Κάθε segment περιέχει ένα ή περισσότερα extents. Τέλος κάθε extent αποτελείται από συνεχόμενα data block τα οποία είναι και η μικρότερη μονάδα αποθήκευσης σε μια Oracle Database. Τα extents ανήκουν σε ένα data file.

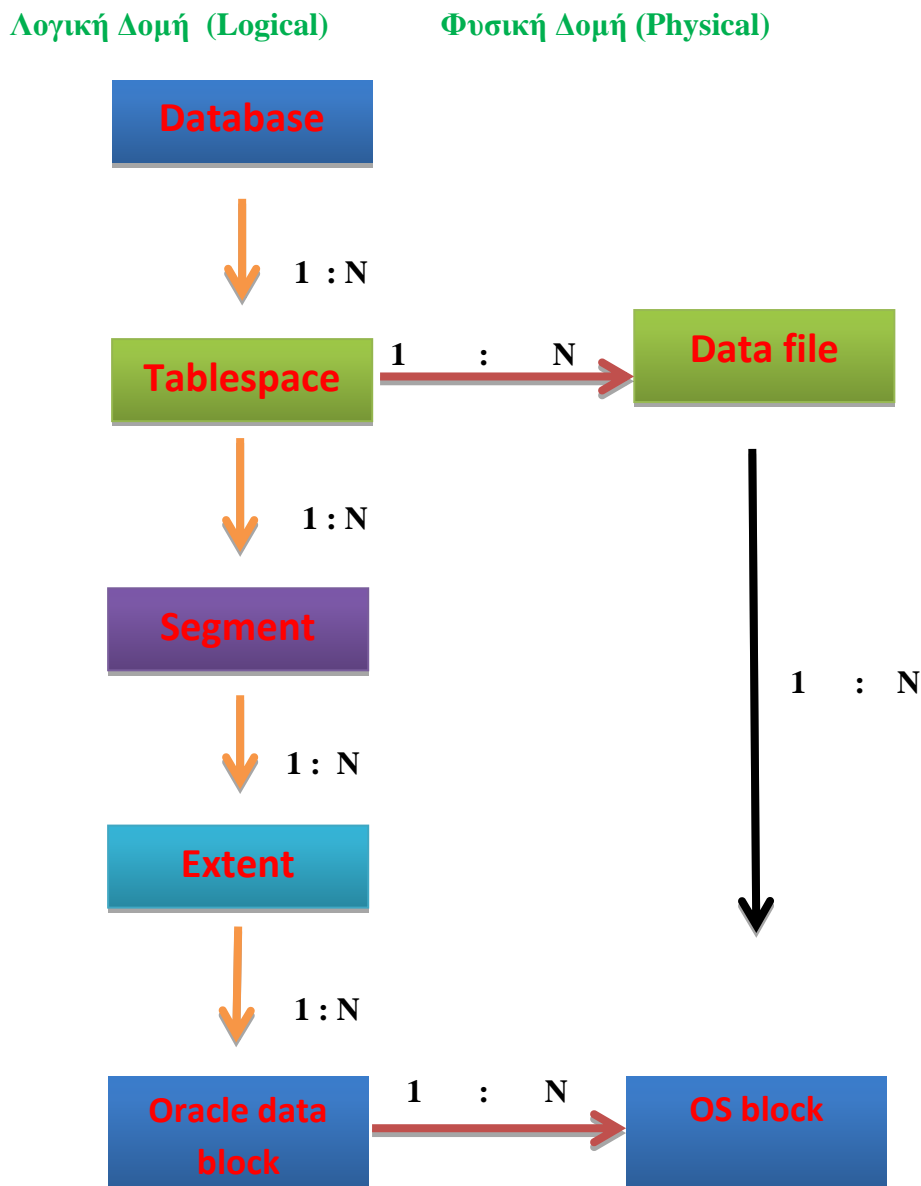
Όταν η Βάση ζητά από το Λειτουργικό Σύστημα, ένα σύνολο από data block για επεξεργασία, το Λειτουργικό Σύστημα φέρνει αυτά τα data block από το φυσικό αρχείο δεδομένων. Κατά τον τρόπο αυτό, ο χρήστης δεν είναι ανάγκη να γνωρίζει που είναι αποθηκευμένο το φυσικό αρχείο στο δίσκο. Είναι δε δυνατό αυτό να είναι διασκορπισμένο (stripped) σε διάφορα σημεία του δίσκου. Το μέγεθος του data block καθορίζεται την στιγμή της δημιουργία της Βάσης με προεπιλεγμένη τιμή τα 8KB. Αυτό μπορεί να αλλάξει αν η Βάση είναι ένα data warehouse, όποτε και θα χρειασθεί μεγάλα αρχεία δεδομένων και δεικτών και κατά συνέπεια μεγαλύτερο data block. Αν η Βάση Δεδομένων υποστηρίζει εφαρμογή με πολλές διεργασίες με τους χρήστες (transactional), τότε απαιτείται μικρότερο μέγεθος block. Το μικρότερο μέγεθος block που υποστηρίζει η Oracle είναι 2 KB.

Τα segments διακρίνονται σε :

- **Data segments** . Είναι τα segments των Data files.
- **Index segments.** Είναι τα segments των Index files.
- **Undo segments.** Σε κάθε Βάση Δεδομένων, δημιουργείται ένα UNDO tablespace το οποίο αποθηκεύει δεδομένα που θα εξασφαλίσουν την συνέπεια (read consistent) της Βάσης Δεδομένων σε περιπτώσεις ανάκαμψης (recovery), καθώς επίσης για την επαναφορά (roll back) των μη επιβεβαιωμένων διεργασιών (uncommitted transactions).
- **Temporary segments.** Δημιουργούνται από την Βάση Δεδομένων για την εξυπηρέτηση των SQL εντολών, όταν αυτές χρειάζονται την διάθεση προσωρινού (temporary) χώρου για την εκτέλεσή τους. Μετά το πέρας των

εντολών SQL, ο χώρος αποδίδεται στο instance της Βάσης για άλλη χρησιμοποίηση.

Ο Oracle server κάνει δυναμική κατανομή του χώρου. Έτσι όταν τα αρχικά διατιθέμενα extents ενός segment, γεμίσουν από δεδομένα, αμέσως διατίθενται νέα και για τον λόγο αυτό τα extents ενός segment δεν είναι συνεχόμενα στο δίσκο.



ΣΧΗΜΑ 2- 6 Λογική και Φυσική δομή μια Βάσης Δεδομένων

### 3. Η ORACLE SQL/DML – ΕΝΤΟΛΗ SELECT

#### Σκοπός και στόχοι της ενότητας

Σκοπός αυτής ενότητας είναι η κατανόηση της χρησιμότητας της γλώσσας SQL, η αναγνώριση των κατηγοριών των εντολών της καθώς και η εξοικείωση με τη διαχείριση της εντολής *Select*, που χρησιμοποιείται για την ανάκτηση δεδομένων από μια Σ.Β.Δ.

Έτσι, μετά το τέλος της ενότητας, οι επιμορφούμενοι θα είναι σε θέση να:

- Αναγνωρίζουν τις βασικές κατηγορίες των SQL εντολών.
- Ανακτούν τα δεδομένα από μια Σ.Β.Δ. με τη χρήση της εντολής *Select*.
- Ταξινομούν τα δεδομένα.
- Επεξεργάζονται τα δεδομένα.
- Ομαδοποιούν τα δεδομένα.
- Ανακτούν δεδομένα από περισσότερους του ενός πίνακες, με τη χρήση της γλώσσας SQL.
- Χρησιμοποιούν τις πράξεις συνόλων για την εκτέλεση σύνθετων ερωτημάτων.

#### Εισαγωγή

Η SQL, Structured English Query Language (γλώσσα δομημένων ερωτημάτων), είναι η πρότυπη μη διαδικαστική γλώσσα που χρησιμοποιείται για την επικοινωνία με μια σχεσιακή βάση δεδομένων.

Ο χρήστης περιγράφει στην SQL τι θέλει κάνει, και ο μεταγλωττιστής (compiler) της γλώσσας, δημιουργεί αυτόματα μια διαδικασία προκειμένου να εκτελεστεί η επιθυμητή εργασία.

Τον Ιούνιο του 1970, ο Dr E.F Codd δημοσίευσε, στο Association of Computer Machinery (ACM), ένα paper με τίτλο "A Relational Model of Data for Large Shared Data Banks". Η IBM Corporation ανέπτυξε μια γλώσσα που υιοθετούσε αυτό το μοντέλο. Η γλώσσα ήταν η Structured English Query Language (SEQUEL). Αργότερα η SEQUEL έγινε SQL (Structured Query Language), που εξακολουθεί όμως να προφέρεται SEQUEL. Το 1979 εκδόθηκε από την Relational Software

Inc.(σήμερα ORACLE corporation) το πρώτο εμπορικό προϊόν SQL. Σήμερα η SQL είναι αποδεκτή ως η πρότυπη γλώσσα για τα (RDBMS).

Το Αμερικάνικο Εθνικό Ινστιτούτο Προτύπων (American National Standards Institute, ANSI) ενέκρινε για πρώτη φορά την SQL ως πρότυπη γλώσσα, για την επικοινωνία σχεσιακών βάσεων δεδομένων, το 1986. Το 1987, το πρότυπο ANSI SQL έγινε δεκτό ως το διεθνές πρότυπο, από το Διεθνή Οργανισμό Προτύπων (International Standards Organization, ISO). Έκτοτε το πρότυπο έχει αναθεωρηθεί αρκετές φορές.

Η Oracle SQL περιλαμβάνει πολλές επεκτάσεις της ANSI / ISO SQL γλώσσας και τα εργαλεία της παρέχουν, επιπλέον συμπληρωματικές εντολές. Κάποιες εφαρμογές και εργαλεία της Oracle απλοποιούν τη σύνταξη των εντολών ή κρύβουν με τη βοήθεια γραφικού περιβάλλοντος την SQL. Πρέπει να τονιστεί ότι το σύνολο των λειτουργιών μιας Oracle ΒΔ πραγματοποιούνται με τη χρήση SQL εντολών.

## Προτάσεις SQL (Statements)

Ο σκοπός της SQL είναι να παρέχει μια διεπαφή για μια σχεσιακή βάση δεδομένων, όπως η Oracle Database. Όλες οι “εντολές” SQL αποτελούν οδηγίες στη βάση δεδομένων. Στο σημείο αυτό, η SQL διαφέρει από τις γλώσσες προγραμματισμού γενικής χρήσης όπως είναι η π.χ. η C και η BASIC.

Μερικά από τα χαρακτηριστικά της SQL είναι τα παρακάτω:

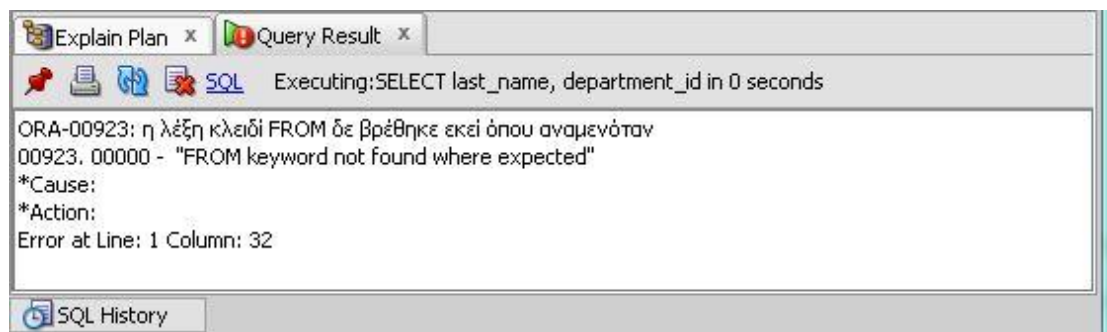
- Επεξεργάζεται σύνολα δεδομένων ως ομάδες και όχι ως μεμονωμένες μονάδες.
- Παρέχει αυτόματη πλοήγηση στα δεδομένα.
- Χρησιμοποιεί τις προτάσεις (statements), που είναι σύνθετες και ισχυρές και αυτόνομες (stand alone). Οι εντολές Ελέγχου ροής, αρχικά δεν αποτελούσαν μέρος της SQL, αλλά ενσωματώθηκαν στο πρόσφατα υιοθετημένο προαιρετικό τμήμα της SQL, ISO / IEC 9075 έως 5: 1996. Είναι γνωστές ως PSM (persistent stored modules). Η επέκταση της Oracle SQL, γνωστή ως PL/SQL, είναι η λύση που προσφέρει η Oracle για να καλύψει την ανάγκη αυτή.

Μια πρόταση (εντολή) SQL, αποτελείται από **αναγνωριστικά** (identifiers), **παραμέτρους** (parameters), **μεταβλητές** (variables) και **δεσμευμένες λέξεις** (reserved words) (π.χ. SELECT, UPDATE).

Μια πρόταση για να τρέξει πρέπει να είναι ολοκληρωμένη.

π.χ `SELECT last_name, department_id FROM employees;`

και όχι `SELECT last_name, department_id`



Εικόνα 3- 1 Μήνυμα σφάλματος, η εντολή δεν είναι ολοκληρωμένη.

Στο εξής συμπληρωματικά με τον όρο **πρόταση SQL** (*SQL statement*), θα χρησιμοποιούμε και τον όρο **εντολή SQL**. Σε κάποιες περιπτώσεις μπορεί να τον συναντήσουμε και ως **δήλωση SQL**.

## Η χρήση της SQL

Η SQL μας παρέχει εντολές για μια ποικιλία εργασιών, όπως:

- Την επερώτηση δεδομένων (Queries).
- Την εισαγωγή, ενημέρωση και διαγραφή γραμμών σ' έναν πίνακα (Insert, Update, Delete).
- Τη δημιουργία, αντικατάσταση, τροποποίηση, και «καταστροφή» αντικειμένων (Create, Replace, Alter, Drop).
- Τον έλεγχο της πρόσβασης στη ΒΔ και τα αντικείμενά της (Control).
- Τη διασφάλιση της συνέπειας και της ακεραιότητας της ΒΔ (database consistency and integrity)

## Κατηγορίες εντολών της ORACLE SQL

Τις Oracle SQL εντολές, μπορούμε να τις κατηγοριοποιήσουμε ως εξής:

- **Data Manipulation Language Statements (Διαχείρισης/Χειρισμού Δεδομένων).** Είναι οι πιο συχνά χρησιμοποιούμενες SQL εντολές. Είναι οι εντολές που χρησιμοποιούμε για να *χειριζόμαστε τα δεδομένα* μιας ΒΔ, δηλαδή για να κάνουμε αναζήτηση, εισαγωγή, ενημέρωση ή διαγραφή εγγραφών. Οι εντολές της DML αφορούν (αναφέρονται ή επηρεάζουν) μόνο τα δεδομένα και όχι τη δομή (*structure*) των πινάκων της ΒΔ. Βασικές **DML** εντολές είναι οι παρακάτω:

- SELECT
- INSERT
- UPDATE
- MERGE
- DELETE
- EXPLAIN PLAN
- LOCK TABLE

- **Data Definition Language Statements (Ορισμού Δεδομένων).** Οι **DDL** εντολές μας επιτρέπουν, να ορίζουμε και να τροποποιούμε τη δομή των αντικειμένων, και όταν δεν τα χρειαζόμαστε πλέον, να τα «καταστρέφουμε». Βασικές εντολές της κατηγορίας είναι:

- CREATE, ALTER, DROP
- RENAME
- TRUNCATE
- GRANT, REVOKE
- AUDIT, NOAUDIT
- COMMENT

- **Transaction Control Statements (Ελέγχου Συναλλαγών).** Διαχειρίζονται τις αλλαγές που γίνονται από τις DML εντολές, και ομαδοποιούν τις DML εντολές σε συναλλαγές (transactions). Βασικές εντολές της κατηγορίας είναι:

- COMMIT
- ROLLBACK
- SAVEPOINT

➤ SET TRANSACTION

- *Session Control Statements (Ελέγχου Συνόδου).*
- *System Control Statements (Ελέγχου Συστήματος).* Η μοναδική εντολή της κατηγορίας είναι η: ALTER SYSTEM, που μας επιτρέπει, εφόσον έχουμε τα κατάλληλα δικαιώματα, να αλλάξουμε κάποια από τα χαρακτηριστικά του INSTANCE της βάσης.
- *Embedded SQL Statements (Ενσωματωμένες).* Μας επιτρέπουν να ενσωματώσουμε εντολές DDL, DML, TCL σε μια διαδικαστική γλώσσα προγραμματισμού. Μερικά παραδείγματα είναι:
  - CONNECT
  - DESCRIBE
  - PREPARE, EXECUTE, EXECUTE IMMEDIATE
  - DECLARE CURSOR, OPEN, CLOSE

## Λεξικολογικές Συμβάσεις (Lexical Conventions)

Μπορούμε να συμπεριλάβουμε ένα ή περισσότερα tabs, Enters (carriage returns), κενά (spaces) ή σχόλια, οπουδήποτε υπάρχει κενό στον ορισμό της εντολής. Οι δύο παρακάτω προτάσεις είναι ισοδύναμες.

```
SELECT last_name, salary*12,MONTHS_BETWEEN(hire_date, SYSDATE)
FROM employees
WHERE department_id = 30
ORDER BY last_name;
```

```
SELECT last_name,
salary * 12,
MONTHS_BETWEEN( hire_date, SYSDATE )
FROM employees WHERE department_id = 30
last_name;                                ORDER BY
```

Τα πεζά-κεφαλαία παίζουν ρόλο μόνο στις αλφαριθμητικές σταθερές και στα ονόματα που έχουμε σε διπλά εισαγωγικά.

Οι εντολές SQL τερματίζονται με διαφορετικό τρόπο στα διάφορα εργαλεία και προγραμματιστικά περιβάλλοντα. Ο χαρακτήρας semicolon (;) χρησιμοποιείται πολύ συχνά για τον σκοπό αυτό και θα τον χρησιμοποιήσουμε στα παραδείγματά μας.

## Εργαλεία σύνταξης και εκτέλεσης εντολών ORACLE SQL

Για να συντάσσουμε και να εκτελούμε SQL εντολές, υπάρχουν διαθέσιμα πάρα πολλά προγράμματα-εργαλεία, τα οποία παρέχονται είτε από την Oracle, είτε από τρίτους κατασκευαστές. Μερικά γνωστά εργαλεία είναι τα εξής:

- Oracle SQL Developer
- SQL\*Plus - iSQL\*Plus
- Oracle JDeveloper
- Oracle Application Express
- Oracle Call Interface and Oracle precompilers
- SQL Navigator της Quest

## Δημιουργία απλών ερωτημάτων (πρόταση SELECT)

Στα παραδείγματα των εκπαιδευτικών σημειώσεων θα χρησιμοποιήσουμε το σχήμα HR, το οποίο περιγράφεται αναλυτικά στο ΠΑΡΑΡΤΗΜΑ Ι.

Ένα ερώτημα είναι μια ερώτηση που κάνουμε στη ΒΔ, χρησιμοποιώντας την εντολή *SELECT*. Τα ερωτήματα χρησιμοποιούνται για να εξάγουν δεδομένα από τη βάση μας, ανάλογα με τις απαιτήσεις του χρήστη. Για παράδειγμα, μπορούμε να δημιουργήσουμε μια κατάσταση με το προσωπικό της πολυεθνικής εταιρείας μας, που εργάζεται στην Ευρώπη και αμείβεται με λιγότερο από 5.000,00€ μηνιαίως.

Η πρόταση *SELECT*, δεν είναι αυτόνομη εντολή και απαιτούνται δύο οι περισσότερες φράσεις (*clauses, τμήματα, στοιχεία*), προκειμένου να γράψουμε μια συντακτικά σωστή εντολή. Σε μια πρόταση *SELECT* θα συναντήσουμε υποχρεωτικά και προαιρετικά τμήματα.

Υπάρχουν τέσσερα τμήματα που είναι πολύτιμα για τη σύνταξη ενός απλού ερωτήματος:

1. *SELECT*
2. *FROM*
3. *WHERE*
4. *ORDER BY*

Η σύνταξη μιας απλής εντολής *SELECT*, είναι η εξής:



```
SELECT      [* | ALL | DISTINCT field1, field 2]
FROM        table1, [table 2]
WHERE       condition1 [condition2]
ORDER BY    filed1|INTEGER [ ASC | DESC ]
```

Οτιδήποτε υπάρχει μεταξύ των **SELECT** και **FROM**, ονομάζεται **select list**. Αν δύο πίνακες ή όψεις που συμμετέχουν σ' ένα ερώτημα, έχουν πεδία με κοινό όνομα, θα πρέπει υποχρεωτικά ν' αναφέρουμε και το όνομα του πίνακα ή της όψης, πριν από το όνομα του πεδίου: π.χ. dept.eno, empl.eno.

Ο αστερίσκος (\*) χρησιμοποιείται, όταν θέλουμε να εμφανιστούν όλες οι στήλες του πίνακα. Η επιλογή **DISTINCT** χρησιμοποιείται για να κρύψει τις διπλοεγγραφές από τ' αποτελέσματα. Τα πεδία στο **select list**, χωρίζονται με κόμματα, ακριβώς όπως και οι πίνακες/όψεις, στο **FROM** τμήμα της εντολής.

Το **FROM** clause είναι υποχρεωτικό στοιχείο σε κάθε ερώτημα και υποδεικνύει σε ποιους πίνακες ή/και όψεις υπάρχουν τα δεδομένα που αναζητούμε. Πρέπει να αναφέρει τουλάχιστον έναν πίνακα ή μία όψη.

Ας δούμε τώρα, δύο παραδείγματα απλών ερωτημάτων στη βάση μας.

```
SELECT * FROM employees;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
80	179 Charles	Johnson	CJOHNSON	011.44.1644...	04/01/00	SA_REP	6200	0,1	149	80
81	180 Winston	Taylor	WTAYLOR	650.507.9876	24/01/98	SH_CLERK	3200	(null)	120	50
82	181 Jean	Fleur	JFLEUR	650.507.9877	23/02/98	SH_CLERK	3100	(null)	120	50
83	182 Martha	Sullivan	MSULLIVA	650.507.9878	21/06/99	SH_CLERK	2500	(null)	120	50
84	183 Girard	Geoni	GGEONI	650.507.9879	03/02/00	SH_CLERK	2800	(null)	120	50
85	184 Nandita	Sarchand	NSARCHAN	650.509.1876	27/01/96	SH_CLERK	4200	(null)	121	50
86	185 Alexis	Bull	ABULL	650.509.2876	20/02/97	SH_CLERK	4100	(null)	121	50
87	186 Julia	Dellinger	JDELLING	650.509.3876	24/06/98	SH_CLERK	3400	(null)	121	50
88	187 Anthony	Cabrio	ACABRIO	650.509.4876	07/02/99	SH_CLERK	3000	(null)	121	50
89	188 Kelly	Chung	KCHUNG	650.505.1876	14/06/97	SH_CLERK	3800	(null)	122	50
90	189 Jennifer	Dilly	JDILLY	650.505.2876	13/08/97	SH_CLERK	3600	(null)	122	50
91	190 Timothy	Gates	TGATES	650.505.3876	11/07/98	SH_CLERK	2900	(null)	122	50
92	191 Randall	Perkins	RPERKINS	650.505.4876	19/12/99	SH_CLERK	2500	(null)	122	50
93	192 Sarah	Bell	SBELL	650.501.1876	04/02/96	SH_CLERK	4000	(null)	123	50
94	193 Britney	Everett	BEVERETT	650.501.2876	03/03/97	SH_CLERK	3900	(null)	123	50
95	194 Samuel	McCain	SMCCAIN	650.501.3876	01/07/98	SH_CLERK	3200	(null)	123	50
96	195 Vance	Jones	VJONES	650.501.4876	17/03/99	SH_CLERK	2800	(null)	123	50
97	196 Alana	Walsh	AWALSH	650.507.9811	24/04/98	SH_CLERK	3100	(null)	124	50
98	197 Kevin	Feeney	KFEENEY	650.507.9822	23/05/98	SH_CLERK	3000	(null)	124	50
99	198 Donald	OConnell	DOCONNEL	650.507.9833	21/06/99	SH_CLERK	2600	(null)	124	50
100	199 Douglas	Grant	DGRANT	650.507.9844	13/01/00	SH_CLERK	2600	(null)	124	50
101	200 Jennifer	Whalen	JWHALEN	515.123.4444	17/09/87	AD_ASST	4400	(null)	101	10
102	201 Michael	Hartstein	MHARTSTE	515.123.5555	17/02/96	MK_MAN	13000	(null)	100	20
103	202 Pat	Fey	PFAY	603.123.6666	17/08/97	MK_REP	6000	(null)	201	20
104	203 Susan	Mavris	SMAVRIS	515.123.7777	07/06/94	HR_REP	6500	(null)	101	40
105	204 Hermann	Baer	HBAER	515.123.8888	07/06/94	PR_REP	10000	(null)	101	70
106	205 Shelley	Higgins	SHIGGINS	515.123.8080	07/06/94	AC_MGR	12000	(null)	101	110
107	206 William	Gietz	WGIEZT	515.123.8181	07/06/94	AC_ACCOUNT	8300	(null)	205	110

Εικόνα 3- 2 Επιστρέφει όλες τις γραμμές και όλες στήλες (\*) του πίνακα employees

```
SELECT last_name, job_id, salary, department_id FROM employees
```

	LAST_NAME	JOB_ID	SALARY	DEPARTMENT_ID
94	Everett	SH_CLERK	3900	50
95	McCain	SH_CLERK	3200	50
96	Jones	SH_CLERK	2800	50
97	Walsh	SH_CLERK	3100	50
98	Feeney	SH_CLERK	3000	50
99	OConnell	SH_CLERK	2600	50
100	Grant	SH_CLERK	2600	50
101	Whalen	AD_ASST	4400	10
102	Hartstein	MK_MAN	13000	20
103	Fay	MK_REP	6000	20
104	Mavris	HR_REP	6500	40
105	Baer	PR_REP	10000	70
106	Higgins	AC_MGR	12000	110
107	Gietz	AC_ACCOUNT	8300	110

Εικόνα 3- 3 Επιστρέφει όλες τις γραμμές και μόνο τέσσερις στήλες του πίνακα employees

## Περιγραφή Πίνακα

Σε περίπτωση που δεν θυμόμαστε τη δομή των πινάκων ή των όψεων, μπορούμε να τη δούμε με την εντολή **Desc** ή **Describe** *<table\_name>*. Όπου *<table\_name>* ο πίνακας ή η όψη που μας ενδιαφέρει.

π.χ. Desc employees

Name	Null	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

Εικόνα 3- 4 Περιγραφή της δομής ενός πίνακα με την DESCribe

## Σχόλια σε μια εντολή SQL

Μπορούμε να ενσωματώσουμε σχόλια σε μια εντολή SQL, με τη χρήση των `/* */`. Ο συνδυασμός των χαρακτήρων `/*` υποδεικνύει την έναρξη ενός σχολίου, ενώ ο συνδυασμός `*/` τον τερματισμό του. Οτιδήποτε υπάρχει μεταξύ αυτών των δύο συνδυασμών θεωρείται σχόλιο και αγνοείται κατά την εκτέλεση της εντολής. Ένα σχόλιο μπορεί να επεκτείνεται σε πολλές γραμμές.

Παράδειγμα:

```
SELECT *  
/* αυτό είναι ένα σχόλιο */  
FROM employees  
WHERE department_id = 30 /* αυτό είναι ένα δεύτερο σχόλιο */  
ORDER BY last_name;
```

Εναλλακτικά μπορούμε να χρησιμοποιήσουμε τους χαρακτήρες `--` για να ξεκινήσουμε ένα σχόλιο. Στην περίπτωση αυτή, το σχόλιο τερματίζεται στο τέλος της ίδιας γραμμής. Π.χ.:

```
SELECT *  
-- αυτό είναι ένα σχόλιο  
FROM employees  
WHERE department_id = 30 -- αυτό είναι ένα δεύτερο σχόλιο  
ORDER BY last_name;
```

## Περιορισμοί και ταξινόμηση δεδομένων

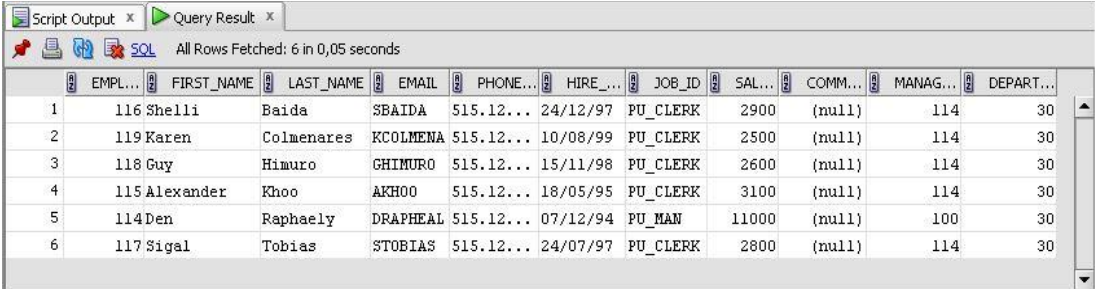
Το τμήμα **WHERE**, είναι προαιρετικό στοιχείο και χρησιμοποιείται για να θέσουμε συνθήκες σ' ένα ερώτημα. Ανάλογα με τις συνθήκες, αποκλείονται από τ' αποτέλεσμα, γραμμές που θα επιστρέφονταν εάν το ερώτημα δεν τις είχε. Η τιμή μιας συνθήκης μπορεί να είναι *TRUE* (αληθής) ή *FALSE* (ψευδής). Στο **WHERE** clause μπορούν να υπάρχουν περισσότερες από μία συνθήκες, οι οποίες συνδέονται με τους κατάλληλους τελεστές.

Το τμήμα **ORDER BY**, είναι προαιρετικό και χρησιμοποιείται για να ταξινομήσουμε τ' αποτέλεσμα ενός ερωτήματος. Μπορούμε να ταξινομήσουμε τα δεδομένα, σε αύξουσα (προεπιλεγμένη ταξινόμηση) ή φθίνουσα σειρά.

Η SQL μας δίνει τη δυνατότητα να ταξινομήσουμε, δηλώνοντας απλά τη θέση της στήλης, με κάποιο ακέραιο αριθμό (μπορεί να θεωρηθεί ως ψευδώνυμο για τη λειτουργία ταξινόμησης) στο *select list*. Στο δεύτερο παράδειγμα γίνεται χρήση αυτής της δυνατότητας.

Ας ξαναδούμε τώρα, τα προηγούμενα παραδείγματα έχοντας προσθέσει τα *WHERE* και *ORDER BY* clauses:

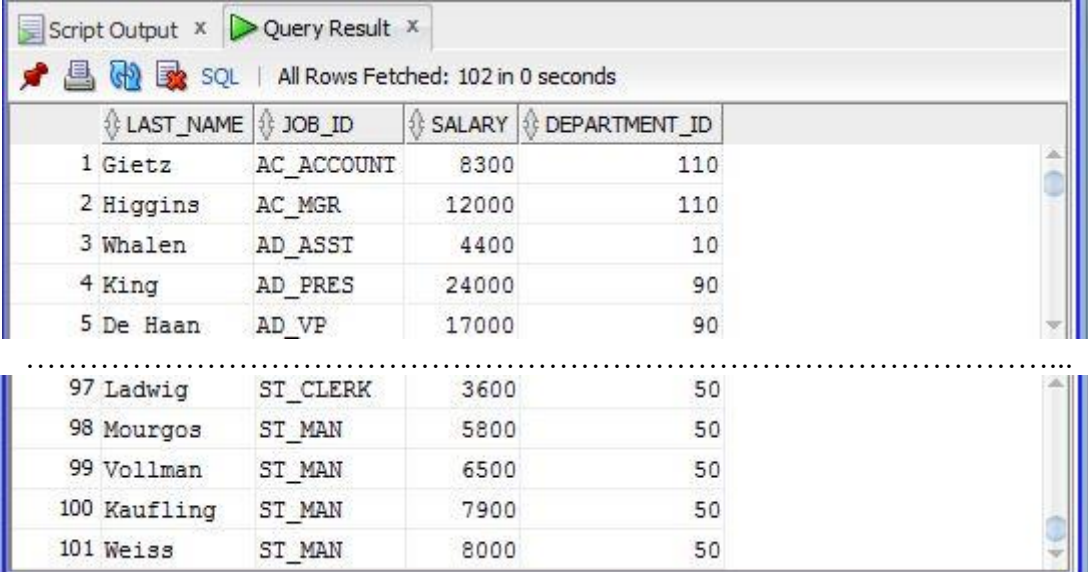
```
SELECT *
FROM employees
WHERE department_id = 30
ORDER BY last_name;
```



	EMPL...	FIRST_NAME	LAST_NAME	EMAIL	PHONE...	HIRE...	JOB_ID	SAL...	COMM...	MANAG...	DEPART...
1	116	Shelli	Baida	SBaida	515.12...	24/12/97	PU_CLERK	2900	(null)	114	30
2	119	Karen	Colmenares	KCOLMENA	515.12...	10/08/99	PU_CLERK	2500	(null)	114	30
3	118	Guy	Himuro	GHIMURO	515.12...	15/11/98	PU_CLERK	2600	(null)	114	30
4	115	Alexander	Khoo	AKH00	515.12...	18/05/95	PU_CLERK	3100	(null)	114	30
5	114	Den	Raphaely	DRAPHEAL	515.12...	07/12/94	PU_MAN	11000	(null)	100	30
6	117	Sigal	Tobias	STOBIAS	515.12...	24/07/97	PU_CLERK	2800	(null)	114	30

Εικόνα 3- 5 Το ερώτημα επιστρέφει 6 γραμμές λόγω της συνθήκης

```
SELECT last_name, job_id, salary, department_id
FROM employees
WHERE NOT (job_id = 'PU_CLERK' AND department_id = 30)
ORDER BY 2, SALARY,1;
```



	LAST_NAME	JOB_ID	SALARY	DEPARTMENT_ID
1	Gietz	AC_ACCOUNT	8300	110
2	Higgins	AC_MGR	12000	110
3	Whalen	AD_ASST	4400	10
4	King	AD_PRES	24000	90
5	De Haan	AD_VP	17000	90
.....				
97	Ladwig	ST_CLERK	3600	50
98	Mourgos	ST_MAN	5800	50
99	Vollman	ST_MAN	6500	50
100	Kaufling	ST_MAN	7900	50
101	Weiss	ST_MAN	8000	50

Εικόνα 3- 6 Ταξινόμηση με βάση τον αριθμό της στήλης στη Select list

Όταν ταξινομούμε με τη χρήση ακεραίων, σαν ταυτότητα (θέση εμφάνισης στο select list) στο ORDER BY clause, πρέπει να θυμόμαστε ότι, σε περίπτωση που αλλάξει η σειρά των πεδίων στο select list, θα αλλάξει και η ταξινόμηση μας. Αντίθετα, αν έχουμε κάνει χρήση των πραγματικών ονομάτων των πεδίων, η ταξινόμησή μας θα παραμείνει ανεπηρέαστη.

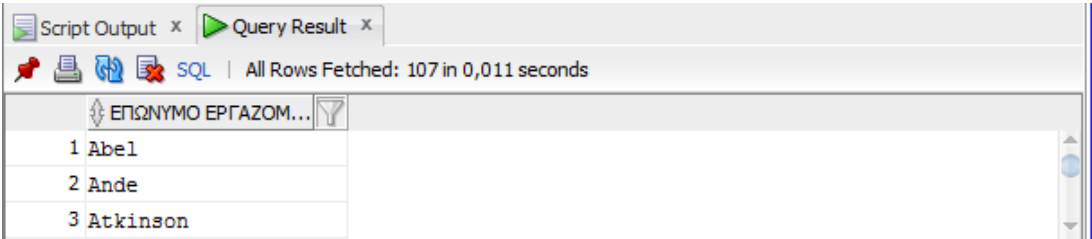
### Ψευδώνυμα Στηλών (column alias)

Έχουμε τη δυνατότητα να μετονομάσουμε τα ονόματα των στηλών των πινάκων σε ένα συγκεκριμένο ερώτημα, με τη χρήση των *ψευδωνύμων* (*alias*). Η μετονομασία αυτή είναι προσωρινή και ισχύει μόνο για την συγκεκριμένη εντολή.

Μπορούμε να δηλώσουμε ψευδώνυμα (alias) για τις στήλες και τις εκφράσεις στο select list, τα οποία μπορούμε να χρησιμοποιήσουμε και στο ORDER BY clause.

Παραδείγματα:

```
SELECT last_name AS "ΕΠΩΝΥΜΟ ΕΡΓΑΖΟΜΕΝΟΥ" FROM employees;
```



The screenshot shows a SQL query result window with the title 'Query Result'. The query is 'SELECT last\_name AS "ΕΠΩΝΥΜΟ ΕΡΓΑΖΟΜΕΝΟΥ" FROM employees;'. The result is displayed in a table with one column, 'ΕΠΩΝΥΜΟ ΕΡΓΑΖΟΜΕΝΟΥ', and three rows of data: '1 Abel', '2 Ande', and '3 Atkinson'.

	ΕΠΩΝΥΜΟ ΕΡΓΑΖΟΜΕΝΟΥ
1	Abel
2	Ande
3	Atkinson

Εικόνα 3- 7 Εκχώρηση ψευδωνύμου στη στήλη last\_name

```
SELECT last_name ΕΠΩΝΥΜΟ FROM employees;
```

```
SELECT department_id did, last_name ΕΠΩΝΥΜΟ  
FROM employees ORDER BY did, 2 DESC;
```



The screenshot shows a SQL query result window with the title 'Query Result'. The query is 'SELECT department\_id did, last\_name ΕΠΩΝΥΜΟ FROM employees ORDER BY did, 2 DESC;'. The result is displayed in a table with two columns, 'DID' and 'ΕΠΩΝΥΜΟ', and four rows of data: '1 10 Whalen', '2 20 Hartstein', '3 20 Fay', and '4 30 Tobias'.

	DID	ΕΠΩΝΥΜΟ
1	10	Whalen
2	20	Hartstein
3	20	Fay
4	30	Tobias

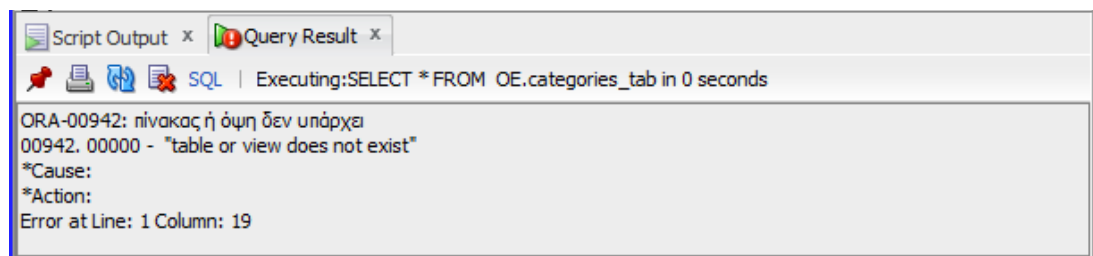
Εικόνα 3- 8 Δημιουργία ψευδωνύμων για δύο στήλες: department\_id και last\_name

## Επιλογή δεδομένων από πίνακες άλλου σχήματος (χρήστη)

Για να ανακτήσουμε και να επεξεργαστούμε δεδομένα που ανήκουν σε άλλο σχήμα (άλλο χρήστη), πρέπει να μας έχουν εκχωρηθεί τα αντίστοιχα δικαιώματα. Σε αντίθετη περίπτωση η προσπέλαση δεν επιτρέπεται. Εφόσον λοιπόν έχουμε το δικαίωμα, να προσπελάσουμε τα δεδομένα ενός πίνακα άλλου χρήστη, πρέπει να βάλουμε πριν από το όνομα του πίνακα ή της όψης, το όνομα του σχήματος. Γενικά η σύνταξη είναι: ΣΧΗΜΑ.ΠΙΝΑΚΑΣ.

Παράδειγμα:

```
SELECT * FROM OE.categories_tab;
```



Εικόνα 3- 9 Δεν έχουμε δικαίωμα ανάγνωσης στον πίνακα OE.categories\_tab

Παίρνουμε ένα μήνυμα λάθους, διότι δεν έχουμε δικαίωμα ανάγνωσης (SELECT) στον πίνακα **categories\_tab** του σχήματος **OE**.

```
SELECT * FROM OE.orders;
```

ORDER_ID	ORDER_DATE	ORDER_MODE	CUSTOMER_ID	ORDER_STATUS
103	2453 04/10/99 20:53:34,362632000	direct	116	0
104	2456 07/11/98 19:53:25,989889000	direct	117	0
105	2457 31/10/99 21:22:16,162632000	direct	118	5

Εικόνα 3- 10 Προσπέλαση του πίνακα OE.orders, για τον οποίο μας έχει εκχωρηθεί το αντίστοιχο δικαίωμα

Αντίθετα με το προηγούμενο παράδειγμα, μπορούμε να διαβάσουμε τον πίνακα **orders** του σχήματος **OE**, διότι μας έχει εκχωρηθεί το κατάλληλο δικαίωμα.

## Ο Ψευδοπίνακας DUAL

Ο DUAL είναι ένας μικρός αλλά χρήσιμος πίνακας της Oracle και χρησιμοποιείται για την εκτέλεση γρήγορων υπολογισμών ή για τον έλεγχο των συναρτήσεων. Ο



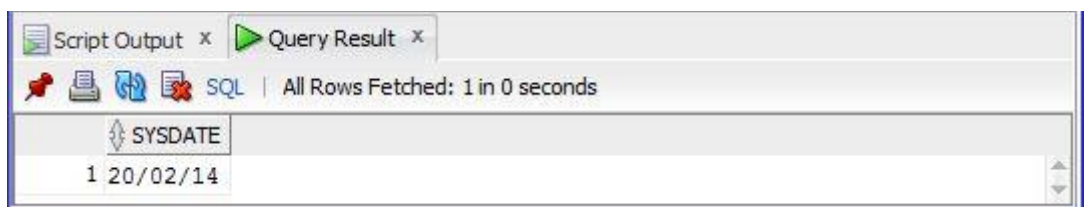
DUAL είναι ένας πίνακας που δημιουργείται αυτόματα από την Oracle, μαζί με το λεξικό δεδομένων και ανήκει στο SYS σχήμα. Διαθέτει μία μόνο στήλη, την DUMMY, τύπου δεδομένων VARCHAR2 (1) και περιέχει μία μόνο εγγραφή με την τιμή 'X'. Όλοι οι χρήστες της βάσης μπορούν να τον προσπελάσουν με το όνομα DUAL.

Πολλές συναρτήσεις λειτουργούν τόσο με σταθερές (constants) όσο και με πεδία πινάκων. Η χρήση του πίνακα DUAL μας επιτρέπει να πειραματιστούμε με τη συμπεριφορά των συναρτήσεων ή τ' αποτελέσματα κάποιων υπολογισμών και στη συνέχεια όταν θα είμαστε σίγουροι, να εφαρμόσουμε τα ίδια με τα πραγματικά πεδία των πινάκων.

Επειδή έχει μόνο μία γραμμή, το αποτέλεσμα επιστρέφεται μία μόνο φορά. Εναλλακτικά, θα μπορούσαμε στη θέση του να χρησιμοποιήσουμε οποιονδήποτε πίνακα, όμως η επιστρεφόμενη τιμή θα εμφανιζόταν τόσες φορές όσες και ο γραμμές του πίνακα.

Ακολουθούν μερικά παραδείγματα:

```
SELECT SYSDATE FROM DUAL;
```



**Εικόνα 3- 11** Χρήση του πίνακα DUAL για να πάρουμε την τρέχουσα ημερομηνία του συστήματος

```
SELECT SYSDATE FROM employees;
```



**Εικόνα 3- 12** Λανθασμένος τρόπος για να πάρουμε την ημερομηνία του συστήματος

## SQL Τελεστές (SQL Operators )

Οι τελεστές χειρίζονται μεμονωμένα στοιχεία δεδομένων και επιστρέφουν ένα αποτέλεσμα. Συντακτικά ένας τελεστής εμφανίζεται πριν ή μετά από έναν τελεστέο ή μεταξύ δύο τελεστέων. Για την παράστασή (απεικόνιση) τους, χρησιμοποιούμε δεσμευμένες λέξεις ή χαρακτήρες. Για παράδειγμα, ο τελεστής για τον πολλαπλασιασμό παριστάνεται με τον αστερίσκο (\*).

Διακρίνουμε τις εξής κατηγορίες τελεστών:

- *Αριθμητικούς τελεστές ( Arithmetic Operators )*
- *Τελεστή συνένωσης (Concatenation Operator )*
- *Τελεστές Συνόλων (Set Operators )*
- *Τελεστές Ιεραρχικών ερωτημάτων (Hierarchical Query Operators )*
- *Multiset Operators*
- *Τελεστές Οριζόμενους από τον χρήστη (User-Defined Operators )*
- *Λογικούς τελεστές*

Ανάλογα με το εάν ένας τελεστής λειτουργεί μ' έναν τελεστέο ή με δύο, διακρίνονται σε Μοναδιαίους και Δυαδικούς (Unary and Binary Operators ). Σε περίπτωση που εφαρμόσουμε έναν τελεστή σε μια NULL τιμή, το επιστρεφόμενο αποτέλεσμα είναι NULL. Ο μοναδικός τελεστής που δεν υπακούει στον παραπάνω κανόνα, είναι ο τελεστής συνένωσης (||) (concatenation).

Σε περίπτωση που μια παράσταση έχει περισσότερους από έναν τελεστές, η σειρά εκτέλεσης τους καθορίζεται ως εξής:

Προτεραιότητα τελεστών SQL:

ΤΕΛΕΣΤΗΣ	ΣΚΟΠΟΣ
+, -	(ως unary τελεστές)
*, /	Πολλαπλασιασμός, Διαίρεση
+, - (ως binary operators),	Πρόσθεση, Αφαίρεση, Συνένωση
=, !=, <>, <, >, <=, >=	Σύγκριση

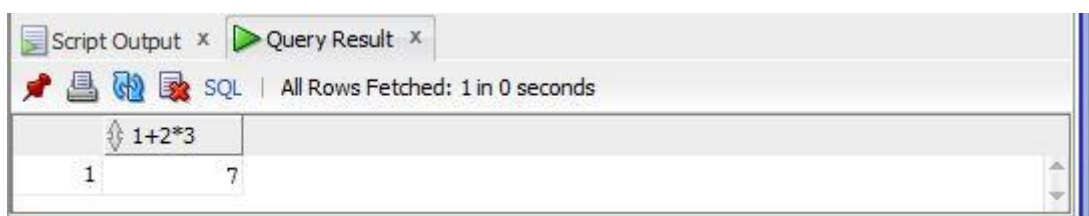


ΤΕΛΕΣΤΗΣ	ΣΚΟΠΟΣ
IS [NOT] NULL, LIKE, [NOT] BETWEEN, [NOT] IN, EXISTS	Σύγκριση
NOT	Λογική άρνηση
AND	Σύζευξη
OR	Διάζευξη
Τελεστές συνόλων	Πράξεις με σύνολα δεδομένων

Πίνακας 3. 1 Προτεραιότητα των SQL τελεστών

Στην παράσταση  $1+2*3$ , θα υπολογιστεί πρώτα το  $2*3$  και μετά η πρόσθεση.

```
SELECT 1+2*3 FROM DUAL;
```



Εικόνα 3- 13 Υπολογισμός της έκφρασης (παράστασης)  $1+2*3$  με τη βοήθεια του DUAL

Δύο σημαντικές παρατηρήσεις:

- Μπορούμε να αλλάξουμε τη σειρά προτεραιότητας με τη χρήση παρενθέσεων.
- Οι τελεστές Συνόλων έχουν όλοι την ίδια προτεραιότητα.

Στη συνέχεια θα δούμε μερικά παραδείγματα, από διάφορες κατηγορίες τελεστών.

### Αριθμητικοί τελεστές

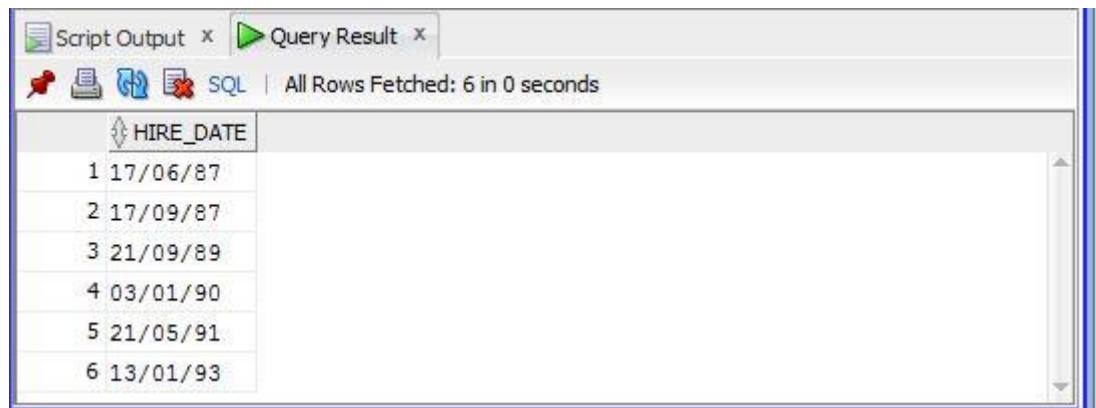
Παραδείγματα:

```
1.SELECT * FROM employees WHERE -salary < 0 ORDER BY 1;
```

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT
94	193	Britney	Everett	BEVERETT	650.501.2876	03/03/97	SH_CLERK	3900	(null)
95	194	Samuel	McCain	SMCCAIN	650.501.3876	01/07/98	SH_CLERK	3200	(null)
96	195	Vance	Jones	VJONES	650.501.4876	17/03/99	SH_CLERK	2800	(null)
97	196	Alana	Walsh	AWALSH	650.507.9811	24/04/98	SH_CLERK	3100	(null)
98	197	Kevin	Feeney	KFEENEY	650.507.9822	23/05/98	SH_CLERK	3000	(null)
99	198	Donald	OConnell	DOCONNEL	650.507.9833	21/06/99	SH_CLERK	2600	(null)
100	199	Douglas	Grant	DGRANT	650.507.9844	13/01/00	SH_CLERK	2600	(null)
101	200	Jennifer	Whalen	JWHALEN	515.123.4444	17/09/87	AD_ASST	4400	(null)
102	201	Michael	Hartstein	MHARTSTE	515.123.5555	17/02/96	MK_MAN	13000	(null)
103	202	Pat	Fay	PFAY	603.123.6666	17/08/97	MK_REP	6000	(null)
104	203	Susan	Mavris	SMAVRIS	515.123.7777	07/06/94	HR_REP	6500	(null)

Εικόνα 3- 14 Χρήση του τελεστή (-) ως μοναδιαίου (unary)

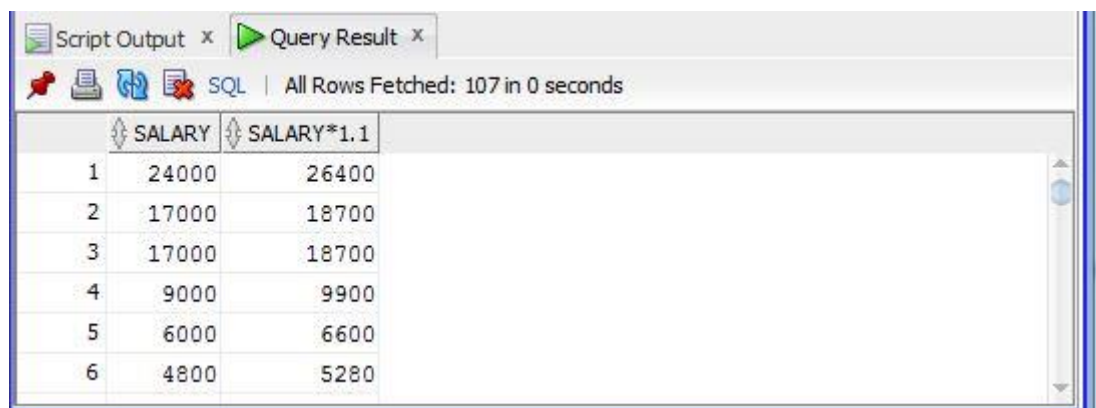
```
2. SELECT hire_date FROM employees
   WHERE SYSDATE - hire_date > 7300 --20 χρόνια
   ORDER BY hire_date;
```



	HIRE_DATE
1	17/06/87
2	17/09/87
3	21/09/89
4	03/01/90
5	21/05/91
6	13/01/93

Εικόνα 3- 15 Χρήση του (-) ως binary τελεστή, για την αφαίρεση δύο ημερομηνιών

```
3. SELECT salary ,salary * 1.1 FROM employees;
```



	SALARY	SALARY*1.1
1	24000	26400
2	17000	18700
3	17000	18700
4	9000	9900
5	6000	6600
6	4800	5280

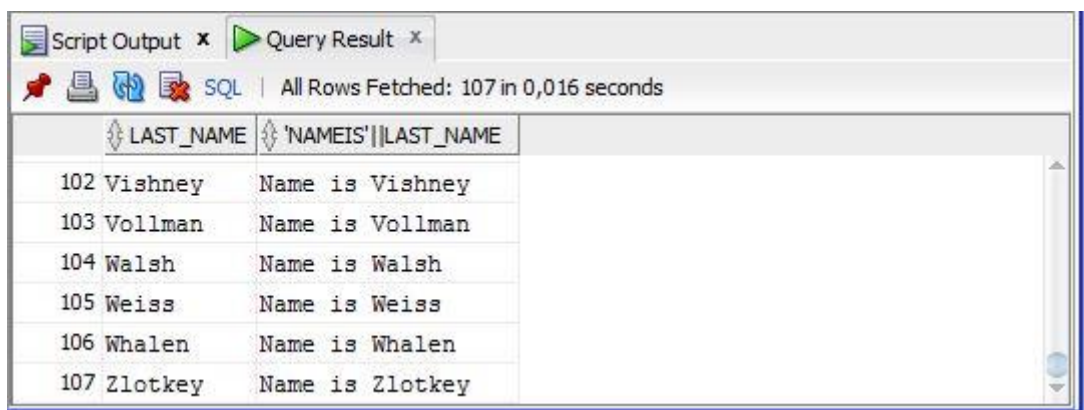
Εικόνα 3- 16 Ο τελεστής (\*)

```
4. UPDATE employees SET salary = salary * 1.1;
```

### Τελεστής συνένωσης (Concatenation Operator)

Παράδειγμα:

```
SELECT last_name , 'Name is ' || last_name
FROM employees
ORDER BY last_name;
```



	LAST_NAME	'NAMEIS'  LAST_NAME
102	Vishney	Name is Vishney
103	Vollman	Name is Vollman
104	Walsh	Name is Walsh
105	Weiss	Name is Weiss
106	Whalen	Name is Whalen
107	Zlotkey	Name is Zlotkey

Εικόνα 3- 17 Ο τελεστής συνένωσης (||)

Η Oracle για τον ίδιο σκοπό, μας παρέχει και τη συνάρτηση CONCAT. Χρησιμοποιείται για λόγους συμβατότητας, σ' εφαρμογές που λειτουργούν σε συστήματα, με διαφορετικά character sets.

### Τελεστές συνόλων (Set Operators)

Συνδυάζουν τα αποτελέσματα δύο ερωτημάτων σ' ένα ενιαίο αποτέλεσμα. Στον παρακάτω πίνακα, βλέπουμε τους τελεστές και τη χρήση τους:

ΤΕΛΕΣΤΗΣ	ΣΚΟΠΟΣ
<b>UNION</b>	Επιστρέφει την ένωση δύο συνόλων. Δεν συμπεριλαμβάνει διπλότυπες εγγραφές.
<b>UNION ALL</b>	Επιστρέφει την ένωση δύο συνόλων. Συμπεριλαμβάνει τις διπλότυπες εγγραφές.
<b>INTERSECT</b>	Επιστρέφει την τομή δύο συνόλων. Δεν Επιστρέφει διπλότυπες εγγραφές
<b>MINUS</b>	Επιστρέφει τις εγγραφές που υπάρχουν στο πρώτο σύνολο και δεν υπάρχουν στο δεύτερο.

Πίνακας 3- 2 Τελεστές συνόλων

Κάθε ένας από τους συγκεκριμένους τελεστές συνόλων, συζητείται λεπτομερώς στο τέλος της ενότητας.

### Τελεστές οριζόμενοι από τους χρήστες

Οι χρήστες έχουν τη δυνατότητα να ορίσουν και δικούς τους τελεστές με την εντολή CREATE OPERATOR και στη συνέχεια να τους χρησιμοποιήσουν, όπως όλους τους

άλλους. Ο μόνος περιορισμός που υπάρχει είναι, να έχει κάποιος το δικαίωμα «εκτέλεσης» (EXECUTE privilege) στον τελεστή.

### SQL Συνθήκες (Conditions)

Μια συνθήκη καθορίζει έναν συνδυασμό από μία ή περισσότερες εκφράσεις και λογικούς τελεστές (Boolean) και επιστρέφει κάποια από τις τιμές, TRUE, FALSE, ή ΑΓΝΩΣΤΗ.

Μπορούμε να χρησιμοποιήσουμε μια συνθήκη στο WHERE τμήμα των εντολών:

- *DELETE*
- *SELECT*
- *UPDATE*

Στην εντολή SELECT, μπορούμε να χρησιμοποιήσουμε συνθήκες στα παρακάτω τμήματά της:

- *WHERE*
- *START WITH*
- *CONNECT BY*
- *HAVING*

Στη συνέχεια παρουσιάζονται συγκεκριμένοι τύποι (μορφές, forms) SQL συνθηκών.

### Συνθήκες σύγκρισης (Comparison Conditions)

Οι συνθήκες σύγκρισης, συγκρίνουν μία ή δύο εκφράσεις και επιστρέφουν μία από τις τιμές TRUE, FALSE, ή NULL. Ακολουθούν μερικά παραδείγματα:

Τελεστής-σκοπός	Παράδειγμα	Περιγραφή αποτελέσματος
<b>= (ισότητα)</b>	<pre>SELECT * FROM employees WHERE salary = 2500 ORDER BY employee_id;</pre>	Επιστρέφει όλες τις στήλες του πίνακα employees, για τους υπαλλήλους, που έχουν μισθό <b>ίσο</b> με 2500. Η ταξινόμηση είναι σε αύξουσα σειρά του κωδικού υπαλλήλου.

Τελεστής-σκοπός	Παράδειγμα	Περιγραφή αποτελέσματος
<b>!= (ανισότητα)</b> <b>^=</b> <b>&lt;&gt;</b>	SELECT * FROM employees WHERE salary <> 2500 ORDER BY employee_id;	Επιστρέφει όλες τις στήλες του πίνακα employees, για τους υπαλλήλους που έχουν μισθό <b>διάφορο</b> του 2500. Η ταξινόμηση είναι σε αύξουσα σειρά του κωδικού υπαλλήλου.
<b>&gt;</b> <b>(μεγαλύτερο)</b> <b>&lt;</b> <b>(μικρότερο)</b>	SELECT * FROM employees WHERE salary < 2500 ORDER BY employee_id Desc;	Επιστρέφει όλες τις στήλες του πίνακα employees, για τους υπαλλήλους που έχουν μισθό <b>μικρότερο</b> από 2500. Η ταξινόμηση είναι σε φθίνουσα σειρά του κωδικού υπαλλήλου.
<b>&gt;=</b> <b>(μεγαλύτερο ή ίσο)</b> <b>&lt;=</b> (μικρότερο ή ίσο)	SELECT * FROM employees WHERE salary >= 2500 ORDER BY employee_id Desc;	Επιστρέφει όλες τις στήλες του πίνακα employees, για τους υπαλλήλους που έχουν μισθό <b>μεγαλύτερο ή ίσο</b> του 2500. Η ταξινόμηση είναι σε φθίνουσα σειρά του κωδικού υπαλλήλου.

Πίνακας 3- 3 Συνθήκες σύγκρισης - Παραδείγματα

### Λογικές Συνθήκες (Logical Conditions)

Μια λογική συνθήκη συνδυάζει τ' αποτελέσματα δύο άλλων συνθηκών, με σκοπό να εξαχθεί ένα αποτέλεσμα ή να αντιστραφεί τ' αποτέλεσμα μιας άλλης συνθήκης.

Ακολουθούν μερικά παραδείγματα:

Τελεστής-σκοπός	Παράδειγμα	Περιγραφή αποτελέσματος
<b>NOT (άρνηση)</b> <b>Επιστρέφει TRUE</b> <b>όταν η έκφραση που</b> <b>ακολουθεί είναι</b> <b>FALSE και</b> <b>αντίστροφα. Όταν η</b> <b>έκφραση είναι</b> <b>NULL επιστρέφει</b> <b>NULL.</b>	SELECT * FROM employees WHERE <b>NOT</b> (job_id IS NULL) ;	Επιστρέφει όλες τις στήλες του πίνακα employees για τους υπαλλήλους που <b>δεν</b> έχουν NULL στο πεδίο job_id.

Τελεστής-σκοπός	Παράδειγμα	Περιγραφή αποτελέσματος
<b>AND (σύζευξη)</b> Επιστρέφει TRUE όταν και οι δύο εκφράσεις είναι TRUE. Όταν μία από τις δύο είναι FALSE επιστρέφει FALSE. Διαφορετικά επιστρέφει NULL.	<pre>SELECT * FROM employees WHERE job_id = 'PU_CLERK' AND department_id = 30;</pre>	Επιστρέφει όλες τις στήλες του πίνακα employees για τους υπαλλήλους που έχουν job_id = 'PU_CLERK' και εργάζονται στο τμήμα με κωδικό 30.
<b>OR (διάζευξη)</b> Επιστρέφει TRUE όταν μία από τις εκφράσεις είναι TRUE. Όταν και οι δύο είναι FALSE επιστρέφει FALSE. Διαφορετικά επιστρέφει NULL.	<pre>SELECT * FROM employees WHERE job_id = 'PU_CLERK' OR department_id = 10;</pre>	Επιστρέφει όλες τις στήλες του πίνακα employees για τους υπαλλήλους που έχουν job_id = 'PU_CLERK' ή εργάζονται στο τμήμα με κωδικό 10.

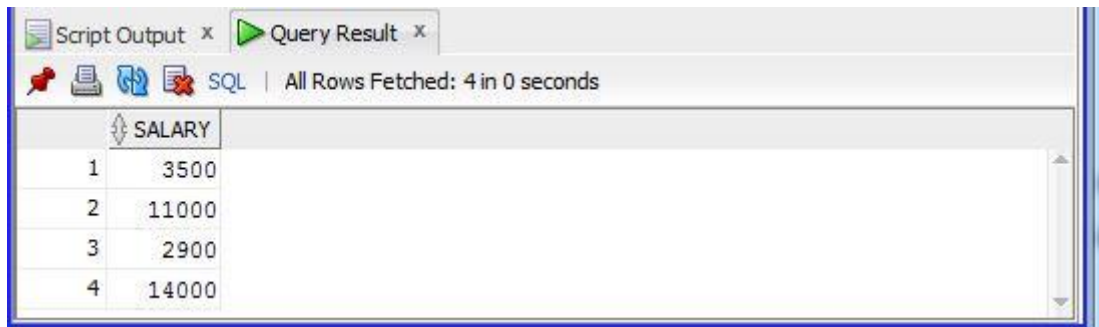
Πίνακας 3- 4 Λογικές συνθήκες, παραδείγματα

### Συνθήκες αναζήτησης μοτίβων (Pattern-matching Conditions)

Οι pattern-matching συνθήκες, χρησιμοποιούνται για τη σύγκριση αλφαριθμητικών. Ο τελεστής LIKE μπορεί να ψάξει σε μια στήλη της βάσης δεδομένων για τιμές, οι οποίες δείχνουν όμοιες με το μοτίβο που περιγράφουμε. Προκειμένου να υποδείξουμε τη μέθοδο ταιριάσματος με το μοτίβο, χρησιμοποιούνται δύο ειδικοί χαρακτήρες (wildcards): το σύμβολο % και η υπογράμμιση (underscore) (\_). Το σύμβολο % υποδηλώνει ότι γίνεται δεκτό οτιδήποτε στη θέση του: κανένας χαρακτήρας, ένας χαρακτήρας ή πολλοί χαρακτήρες. Η υπογράμμιση παριστά έναν μόνο χαρακτήρα.

Το παράδειγμα που ακολουθεί βρίσκει και επιστέφει τον μισθό όλων των υπαλλήλων που το επώνυμο τους ξεκινάει με το γράμμα 'R':

```
SELECT salary
FROM employees
WHERE last_name LIKE 'R%';
```



The screenshot shows a SQL Developer window with a 'Query Result' tab. It displays a table with one column named 'SALARY' and four rows of data. The status bar indicates 'All Rows Fetched: 4 in 0 seconds'.

	SALARY
1	3500
2	11000
3	2900
4	14000

Εικόνα 3- 18 Παράδειγμα Pattern-matching

Το μοτίβο πρέπει να βρίσκεται στα δεξιά του τελεστή LIKE. Η εντολή που ακολουθεί θα φέρει τον μισθό των υπαλλήλων που το επώνυμο τους είναι ίσο με 'SM%' .

```
SELECT salary
FROM employees
WHERE 'SM%' LIKE last_name;
```



The screenshot shows a SQL Developer window with a 'Query Result' tab. The text 'no rows selected' is displayed in the results area. The status bar indicates 'Task completed in 0 seconds'.

Εικόνα 3- 19 Λανθασμένη χρήση του μοτίβου ('SM%')

Πρέπει να τονιστεί ότι γενικά η Oracle, κάνει διάκριση πεζών/κεφαλαίων χαρακτήρων στις τιμές των δεδομένων, οπότε πρέπει να είμαστε προσεκτικοί με το πώς γράφουμε το μοτίβο μας. Εάν θέλουμε ν' αναζητήσουμε κάποιον από τους ειδικούς χαρακτήρες, πρέπει να χρησιμοποιήσουμε τον λεγόμενο ESCAPE χαρακτήρα, σύμφωνα με το παράδειγμα:

```
SELECT last_name
FROM employees
WHERE last_name
LIKE '%A\_B%' ESCAPE '\';
```

Ορίζουμε το '\ ' ως ESCAPE χαρακτήρα, με αποτέλεσμα η Oracle να αντιμετωπίζει τον χαρακτήρα που έπεται αυτού ως κοινό και όχι ως χαρακτήρα μπαλαντέρ. Ως ESCAPE χαρακτήρες μπορούν να οριστούν όλοι εκτός από τους δύο ειδικούς χαρακτήρες '%' και '\_'. Εάν στο ίδιο μοτίβο θέλουμε ταυτόχρονα ν' αναζητήσουμε και τον ESCAPE χαρακτήρα, απλά τον γράφουμε δύο φορές συνεχόμενα.

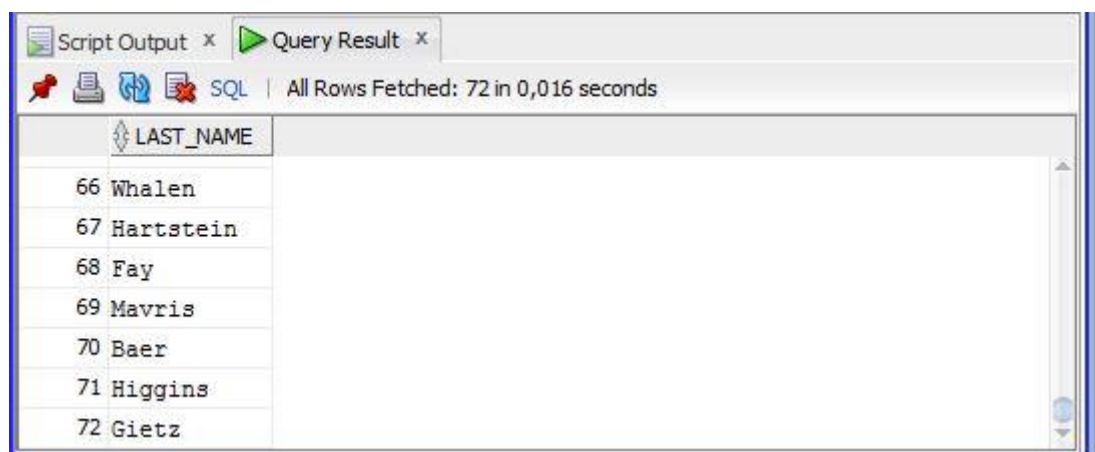


Παράδειγμα, αν έχουμε ορίσει το @ ως ESCAPE χαρακτήρα, τότε μπορούμε να γράψουμε το @@ στο μοτίβο προκειμένου να ψάξουμε για το @.

### Null συνθήκες (Null Conditions)

Ο τελεστής NULL, ελέγχει εάν υπάρχουν δεδομένα σε μια στήλη, μιας γραμμής, ενός πίνακα. Εάν η στήλη δεν έχει δεδομένα (είναι κενή) λέμε ότι είναι NULL . Ο έλεγχος γίνεται με τα IS NULL και IS NOT NULL, που επιστρέφουν TRUE όταν μια στήλη ή έκφραση είναι NULL ή όχι, αντίστοιχα.

```
SELECT last_name
FROM employees
WHERE commission_pct IS NULL;
```



The screenshot shows a 'Query Result' window with a table containing 72 rows. The first column is labeled 'LAST\_NAME'. The rows are numbered 66 through 72. The names listed are Whalen, Hartstein, Fay, Mavris, Baer, Higgins, and Gietz. The window also shows 'All Rows Fetched: 72 in 0,016 seconds'.

	LAST_NAME
66	Whalen
67	Hartstein
68	Fay
69	Mavris
70	Baer
71	Higgins
72	Gietz

Εικόνα 3- 20 Ο τελεστής IS NULL, για τον εντοπισμό κενών τιμών

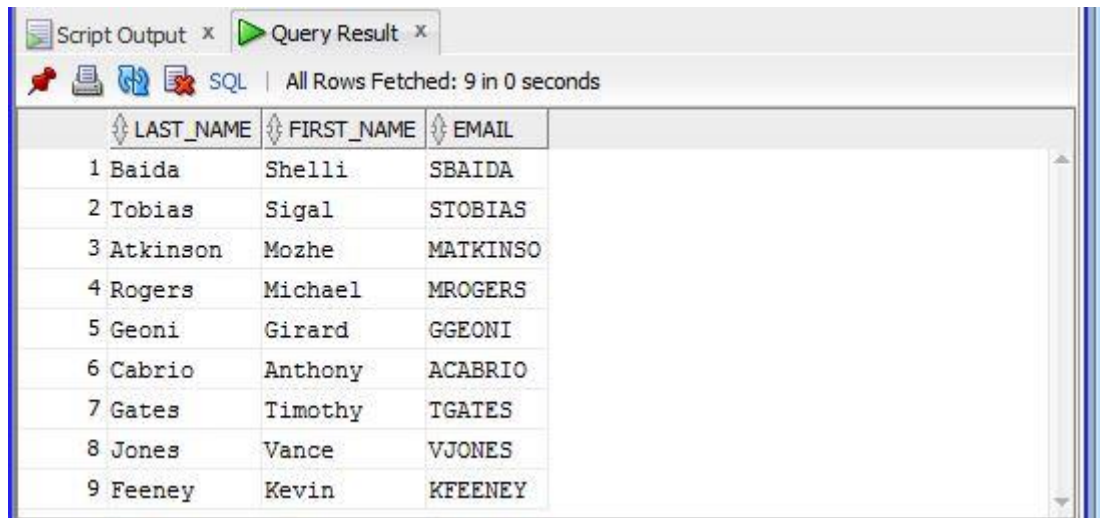
Στο παράδειγμα αυτό, ζητούμε τα επώνυμα των υπαλλήλων που δεν παίρνουν προμήθεια (commission\_pct IS NULL).

### BETWEEN συνθήκες (BETWEEN Conditions )

Ο τελεστής BETWEEN, χρησιμοποιείται για αναζήτηση τιμών που βρίσκονται μέσα σε ένα εύρος τιμών, δεδομένης της ελάχιστης και της μέγιστης τιμής. Τα δύο άκρα περιλαμβάνονται στο σύνολο τιμών. Το παράδειγμα επιστρέφει, το επώνυμο, το όνομα και το email από τον πίνακα employees, μόνο για τους υπαλλήλους που έχουν μισθό μεταξύ 2800 και 3000, συμπεριλαμβανομένων των άκρων.

```
SELECT last_name, first_name, email
FROM employees
WHERE salary
BETWEEN 2800 AND 3000;
```





Script Output x Query Result x

SQL | All Rows Fetched: 9 in 0 seconds

	LAST_NAME	FIRST_NAME	EMAIL
1	Baida	Shelli	SBAIDA
2	Tobias	Sigal	STOBIAS
3	Atkinson	Mozhe	MATKINSO
4	Rogers	Michael	MROGERS
5	Geoni	Girard	GGEONI
6	Cabrio	Anthony	ACABRIO
7	Gates	Timothy	TGATES
8	Jones	Vance	VJONES
9	Feeney	Kevin	KFEENEY

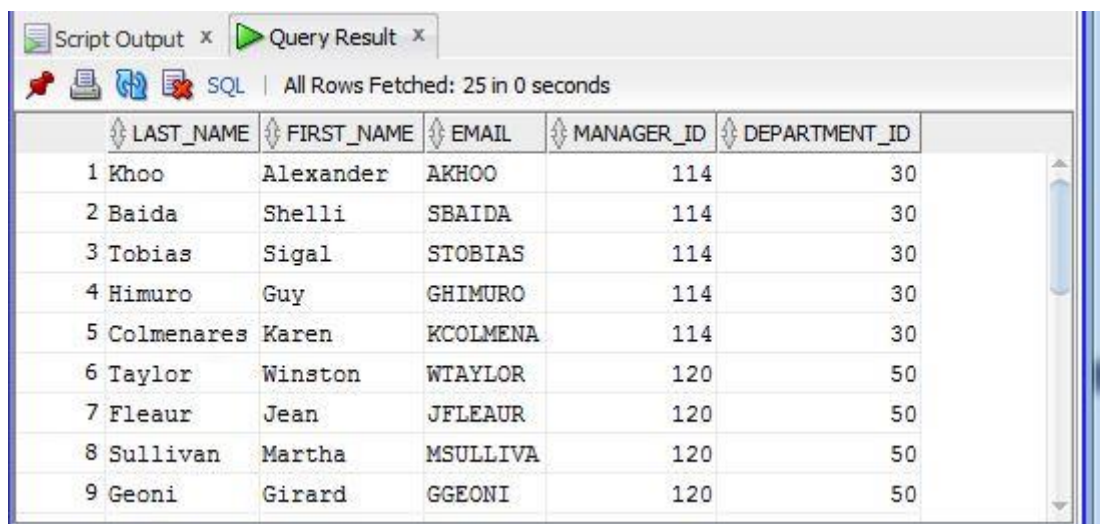
Εικόνα 3- 21 Ο τελεστής BETWEEN, παράδειγμα

### IN συνθήκη (IN Condition)

Ο τελεστής IN, χρησιμοποιείται για να συγκρίνει μια τιμή με μια λίστα τιμών ή ένα άλλο υποερώτημα. Ας εξετάσουμε τα δύο παραδείγματα που ακολουθούν:

```
SELECT last_name, first_name, email, manager_id, department_id  
FROM employees WHERE job_id IN ('PU_CLERK', 'SH_CLERK');
```

Εδώ, ο Database Server θα βρει και θα μας εμφανίσει τα στοιχεία των υπαλλήλων, που εργάζονται σε θέσεις εργασίας με κωδικούς, 'PU\_CLERK' και 'SH\_CLERK'.



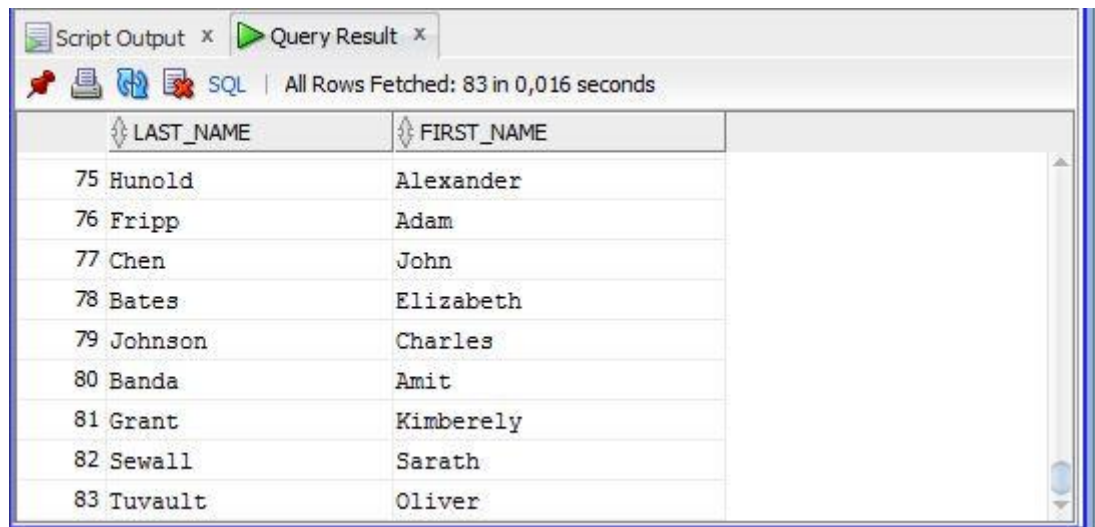
Script Output x Query Result x

SQL | All Rows Fetched: 25 in 0 seconds

	LAST_NAME	FIRST_NAME	EMAIL	MANAGER_ID	DEPARTMENT_ID
1	Khoo	Alexander	AKHOO	114	30
2	Baida	Shelli	SBAIDA	114	30
3	Tobias	Sigal	STOBIAS	114	30
4	Himuro	Guy	GHIMURO	114	30
5	Colmenares	Karen	KCOLMENA	114	30
6	Taylor	Winston	WTAYLOR	120	50
7	Fleaur	Jean	JFLEAUR	120	50
8	Sullivan	Martha	MSULLIVA	120	50
9	Geoni	Girard	GGEONI	120	50

Εικόνα 3- 22 Αναζήτηση σε λίστα τιμών με τον τελεστή IN

```
SELECT last_name, first_name FROM employees
WHERE salary NOT IN (SELECT salary FROM employees
                     WHERE department_id = 30);
```



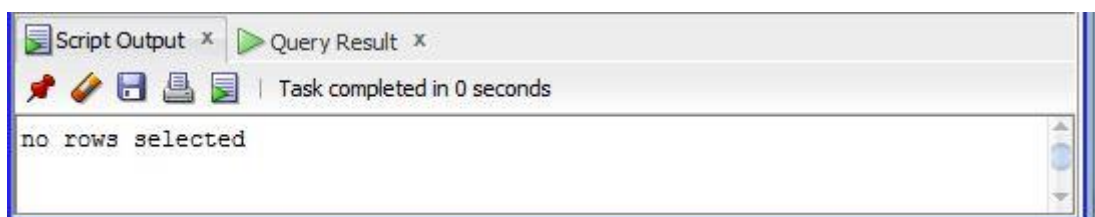
	LAST_NAME	FIRST_NAME
75	Hunold	Alexander
76	Fripp	Adam
77	Chen	John
78	Bates	Elizabeth
79	Johnson	Charles
80	Banda	Amit
81	Grant	Kimberely
82	Sewall	Sarath
83	Tuvault	Oliver

**Εικόνα 3- 23** Αποκλεισμός τιμών που ανήκουν σε κάποιο σύνολο, τελεστής NOT IN

Στην περίπτωση αυτή, θα πάρουμε τα ονοματεπώνυμα των υπαλλήλων, όπου ο μισθός τους δεν ταιριάζει με κανέναν μισθό, οποιουδήποτε υπαλλήλου εργάζεται στο τμήμα με κωδικό 30.

Όταν χρησιμοποιούμε το NOT IN πρέπει να θυμόμαστε, ότι αν κάποια τιμή στη λίστα είναι NULL, τότε η συνθήκη δεν μπορεί να αποτιμηθεί. Το ακόλουθο παράδειγμα δεν θα επιστρέψει καμία γραμμή:

```
SELECT * FROM employees
WHERE department_id NOT IN (10, 20, NULL);
```

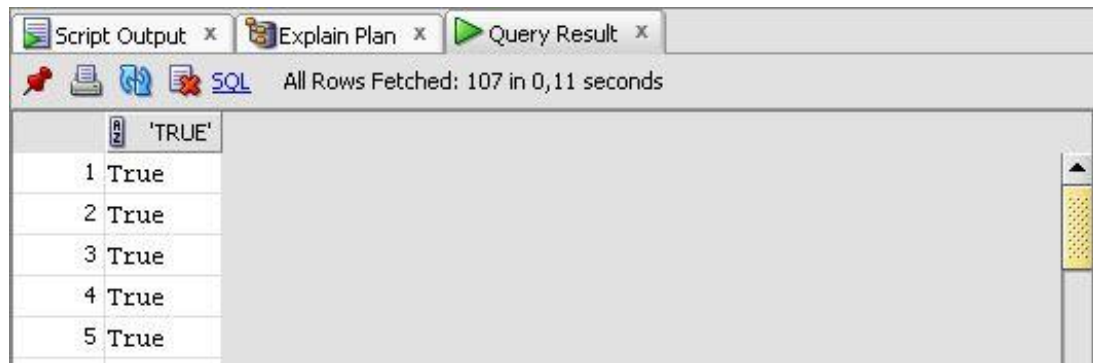


no rows selected

**Εικόνα 3- 24** Δεν επιστρέφει γραμμές, διότι υπάρχει NULL τιμή στη λίστα

Αυτό δεν ισχύει όταν ο τελεστής NOT IN αναφέρεται σε ένα υποερώτημα (subquery). Να σημειωθεί, ότι όταν ο τελεστής NOT IN αναφέρεται σε υποερώτημα που δεν επιστρέφει καμία γραμμή, τότε η συνθήκη αποτιμάται ως αληθής για όλες τις εγγραφές. Ας δούμε το παράδειγμα:

```
SELECT 'True' FROM employees
WHERE department_id NOT IN (SELECT 0 FROM DUAL WHERE 1=2);
```



	'TRUE'
1	True
2	True
3	True
4	True
5	True

**Εικόνα 3- 25** Το NOT IN επιστρέφει τιμές παρόλο που το υποερώτημα δεν επιστρέφει τίποτα.

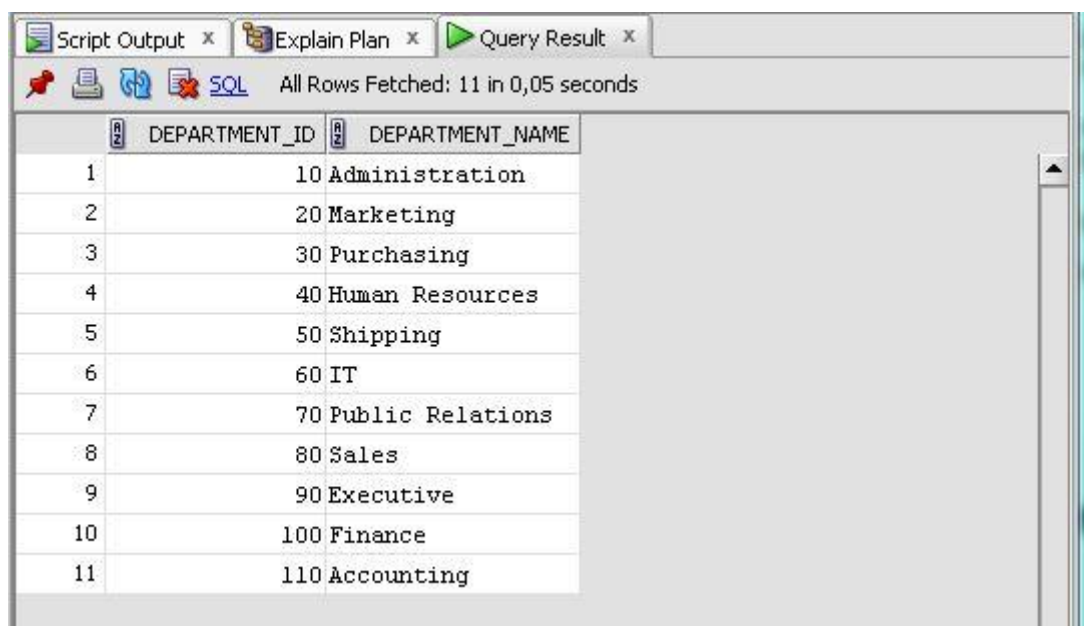
Συγκεκριμένα επιστρέφει 107 γραμμές όσοι και οι υπάλληλοι της εταιρείας.

### EXISTS συνθήκη (EXISTS Condition)

Ο τελεστής EXISTS χρησιμοποιείται για να κάνει αναζήτηση, σχετικά με την παρουσία εγγραφών (τουλάχιστον μία), σε κάποιο υποερώτημα.

```
SELECT department_id
FROM departments d
WHERE EXISTS
(SELECT * FROM employees e
WHERE d.department_id = e.department_id);
```

Το ερώτημα αυτό επιστρέφει τους κωδικούς των τμημάτων, για εκείνα τα τμήματα που έχουν τουλάχιστον έναν εργαζόμενο.



	DEPARTMENT_ID	DEPARTMENT_NAME
1	10	Administration
2	20	Marketing
3	30	Purchasing
4	40	Human Resources
5	50	Shipping
6	60	IT
7	70	Public Relations
8	80	Sales
9	90	Executive
10	100	Finance
11	110	Accounting

**Εικόνα 3- 26** Παράδειγμα χρήσης του τελεστή EXISTS

## Η ψευδοστήλη ROWNUM (Pseudocolumn)

Για κάθε γραμμή που επιστρέφεται από ένα ερώτημα, η ψευδοστήλη ROWNUM, επιστρέφει έναν αριθμό που δείχνει τη σειρά με την οποία η Oracle επέλεξε τη γραμμή. Η πρώτη γραμμή που επιλέγεται έχει ROWNUM 1, η δεύτερη 2 και ούτω καθεξής.

Μπορούμε να χρησιμοποιήσουμε τη ROWNUM, για να περιορίσουμε τον αριθμό των γραμμών που επιστρέφονται από ένα ερώτημα, όπως σε αυτό το παράδειγμα:

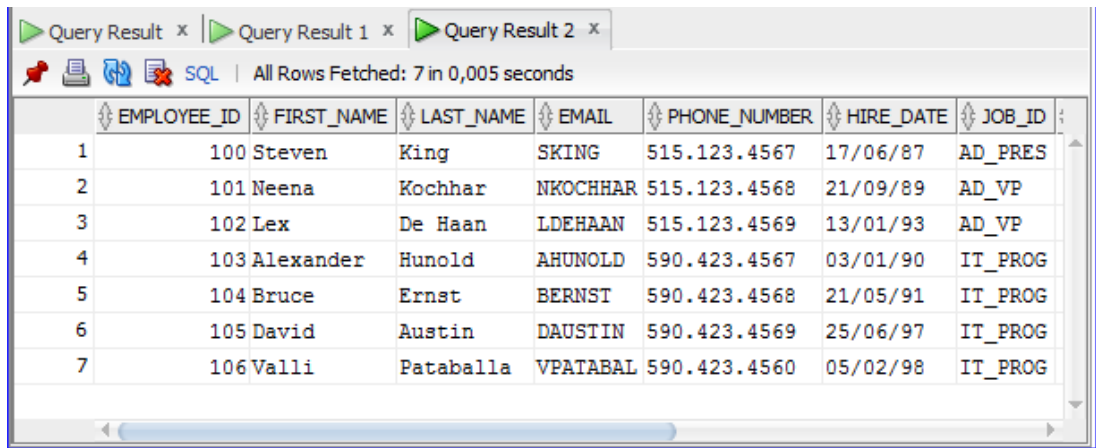
```
SELECT rownum, first_name FROM employees WHERE ROWNUM <=5;
```

Αν στο ίδιο ερώτημα υπάρχει η ROWNUM και στη συνέχεια ακολουθεί το ORDER BY, οι επιστρεφόμενες γραμμές θα αναδιαταχθούν. Τ' αποτελέσματα μπορεί να διαφέρουν ανάλογα με τον τρόπο που προσπελάζονται οι εγγραφές από τον server. Για παράδειγμα, εάν το ORDER BY οδηγήσει την Oracle να χρησιμοποιήσει ένα ευρετήριο (index), τότε η Oracle πιθανόν να ανακτήσει τις γραμμές με διαφορετική σειρά από ό, τι χωρίς το ευρετήριο. Ως εκ τούτου, η ακόλουθη εντολή δεν επιστρέφει κατ' ανάγκη τις ίδιες γραμμές με το προηγούμενο παράδειγμα:

```
SELECT * FROM employees
WHERE ROWNUM < 5
ORDER BY last_name;
```

Εάν ενσωματώσουμε το ORDER BY σε ένα subquery και τοποθετήσουμε τη συνθήκη με τη ROWNUM στο κύριο ερώτημα, τότε η συνθήκη θα ελεγχτεί μετά την ταξινόμηση των εγγραφών. Για παράδειγμα, το ακόλουθο ερώτημα επιστρέφει τους εργαζόμενους με τους 7 μικρότερους κωδικούς. Αυτό αναφέρεται και ως αναφορά «top-N».

```
SELECT * FROM
(SELECT * FROM employees ORDER BY employee_id)
WHERE ROWNUM < 7;
```



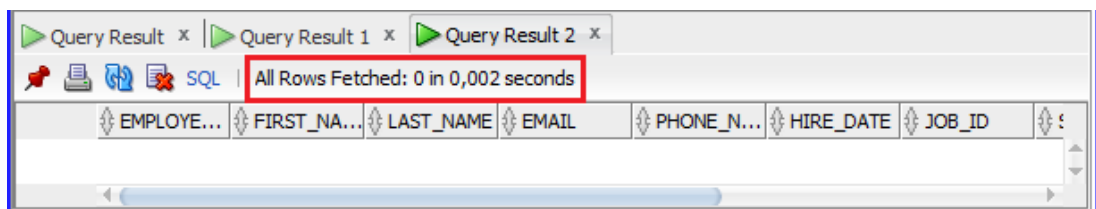
The screenshot shows a SQL Developer window with a query result table. The table has 8 columns: EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER, HIRE\_DATE, and JOB\_ID. The first 7 rows are displayed, corresponding to employees with IDs 100 through 106. The status bar at the top indicates 'All Rows Fetched: 7 in 0,005 seconds'.

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID
1	100	Steven	King	SKING	515.123.4567	17/06/87	AD_PRES
2	101	Neena	Kochhar	NKOCHHAR	515.123.4568	21/09/89	AD_VP
3	102	Lex	De Haan	LDEHAAN	515.123.4569	13/01/93	AD_VP
4	103	Alexander	Hunold	AHUNOLD	590.423.4567	03/01/90	IT_PROG
5	104	Bruce	Ernst	BERNST	590.423.4568	21/05/91	IT_PROG
6	105	David	Austin	DAUSTIN	590.423.4569	25/06/97	IT_PROG
7	106	Valli	Pataballa	VPATABAL	590.423.4560	05/02/98	IT_PROG

Εικόνα 3- 27 Ερώτημα Top-7 με τη βοήθεια της ψευδοστήλης ROWNUM

Οι συνθήκες που ελέγχουν για τιμές της ROWNUM που είναι μεγαλύτερες από έναν θετικό ακέραιο, επιστρέφουν πάντοτε false (ψέμα). Η παρακάτω εντολή δεν επιστρέφει καμία γραμμή:

```
SELECT * FROM employees WHERE ROWNUM > 5;
```



The screenshot shows a SQL Developer window with a query result table. The status bar at the top indicates 'All Rows Fetched: 0 in 0,002 seconds', which is highlighted with a red box. The table header shows columns EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER, HIRE\_DATE, and JOB\_ID, but no data rows are displayed.

Εικόνα 3. 28 Η συνθήκη ROWNUM > 5 είναι πάντα false

## Οι SQL Συναρτήσεις

Οι *SQL Συναρτήσεις*, στο εξής θα αναφέρονται απλά ως συναρτήσεις, είναι παρόμοιες με τους τελεστές, υπό την έννοια ότι χειρίζονται δεδομένα και επιστρέφουν κάποιο αποτέλεσμα. Διαφέρουν όμως από τους τελεστές ως προς τη σύνταξη. Οι συναρτήσεις μπορούν να δέχονται εντός παρενθέσεων, κανένα, ένα, δύο ή περισσότερα ορίσματα. Η γενική τους σύνταξη είναι:

➤ *function(argument1, argument2, ...)*

Είναι ενσωματωμένες στην Oracle Database και είναι διαθέσιμες για χρήση στις διάφορες SQL εντολές. Δεν θα πρέπει να συγχέουμε τις SQL Συναρτήσεις με αυτές που φτιάχνουν οι χρήστες με τη χρήση της PL/SQL.

Σε περίπτωση που καλέσουμε μια συνάρτηση με όρισμα, διαφορετικού τύπου δεδομένων από το προβλεπόμενο, τότε η Oracle θα επιχειρήσει να μετατρέψει τον τύπο δεδομένων του ορίσματος πριν την εκτέλεσή της.

Οι συναρτήσεις που θα εξετάσουμε μπορούν να κατηγοριοποιηθούν ως εξής:

- Single row functions (συναρτήσεις μίας γραμμής)
- Aggregate functions (συναρτήσεις ομαδοποίησης ή συγκεντρωτικές συναρτήσεις)

Υπάρχουν και άλλες συναρτήσεις, οι οποίες όμως δεν θα μας απασχολήσουν στο συγκεκριμένο εγχειρίδιο (Analytic . functions, Object reference functions, Model functions, user defined functions).

### Συναρτήσεις μίας γραμμής (Single row functions)

Οι συναρτήσεις μίας γραμμής, επιστρέφουν τιμή για κάθε γραμμή του πίνακα βάσης και διακρίνονται ανάλογα με τον τύπο δεδομένων των ορισμάτων και των επιστρεφόμενων τιμών τους. Στη συνέχεια θα εξετάσουμε κάποιες βασικές συναρτήσεις και θα δούμε αντίστοιχα παραδείγματα. Το σύνολο των συναρτήσεων που διαθέτει η Oracle είναι πολύ μεγάλο και εφόσον κάποιος επιθυμεί να μελετήσει σε βάθος μπορεί να ανατρέξει στα σχετικά εγχειρίδια της Oracle, τα οποία διατίθενται δωρεάν στον ιστότοπό της (<http://docs.oracle.com/>).

### Αριθμητικές συναρτήσεις (Numeric functions)

Οι συναρτήσεις αυτές δέχονται αριθμητικά ορίσματα και επιστρέφουν αριθμούς. Ο πίνακας παρουσιάζει μερικές βασικές συναρτήσεις χειρισμού αριθμητικών δεδομένων.

Συνάρτηση	Σύντομη Περιγραφή
<b>ABS (τιμή)</b>	Απόλυτη τιμή
<b>CEIL (τιμή)</b>	Ο μικρότερος ακέραιος, που είναι μεγαλύτερος από ή ίσος, με την «τιμή».
<b>EXP(τιμή)</b>	Υψώνει τη σταθερά e στην «τιμή»
<b>FLOOR(τιμή)</b>	Ο μεγαλύτερος ακέραιος, που είναι μικρότερος από ή ίσος, με την «τιμή»
<b>LN(τιμή), LOG(τιμή)</b>	Ο Φυσικός και Δεκαδικός λογάριθμος αντίστοιχα
<b>MOD(τιμή, διαιρέτης)</b>	Επιστρέφει το υπόλοιπο της διαίρεσης
<b>POWER(τιμή, εκθέτης)</b>	Υψώνει την «τιμή» στη δύναμη «εκθέτης»



Συνάρτηση	Σύντομη Περιγραφή
<b>ROUND(τιμή, ακρίβεια)</b>	Στρογγυλοποιεί την «τιμή» σύμφωνα με την «ακρίβεια»
<b>SQRT(τιμή)</b>	Η τετραγωνική ρίζα της τιμής
<b>TRUNC(τιμή, ακρίβεια)</b>	Περικόπτει την «τιμή» σύμφωνα με την «ακρίβεια»
<b>ACOS(), ASIN(), ATAN(), COS(), COSH(), SIN(), SINH(), TAN(), TANH()</b>	Τριγωνομετρικές συναρτήσεις

Πίνακας 3- 5 Χρήσιμες αριθμητικές συναρτήσεις

Παραδείγματα βασικών αριθμητικών συναρτήσεων:

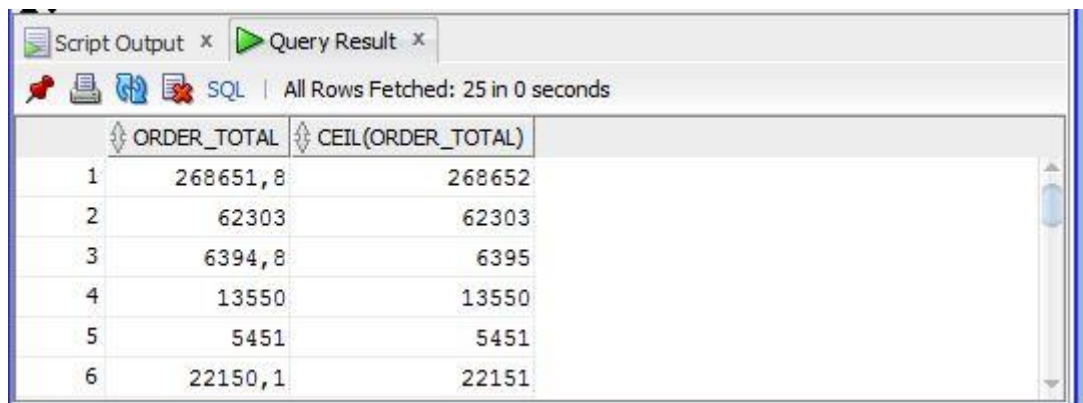
**ABS:** Επιστρέφει την απόλυτη τιμή μιας αριθμητικής τιμής.

```
SELECT ABS(-15) Absolute FROM DUAL;
```

**Αποτέλεσμα: 15**

**CEIL:** Επιστρέφει τον μικρότερο ακέραιο, που είναι μεγαλύτερος από ή ίσος με το όρισμα.

```
SELECT order_total, CEIL(order_total)
FROM OE.orders
WHERE order_id >= 2434;
```



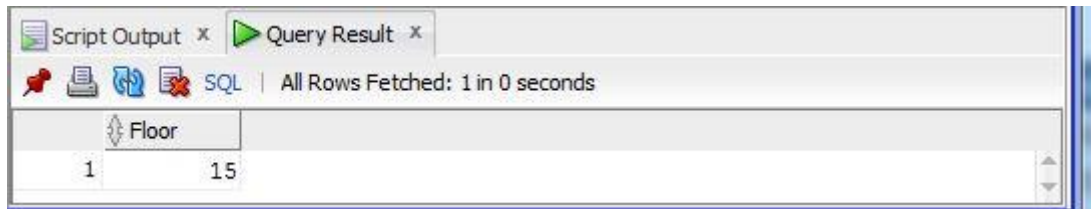
The screenshot shows a SQL query result window with two tabs: 'Script Output' and 'Query Result'. The 'Query Result' tab is active, displaying a table with two columns: 'ORDER\_TOTAL' and 'CEIL(ORDER\_TOTAL)'. The table contains six rows of data, with the first row highlighted. The status bar at the top indicates 'All Rows Fetched: 25 in 0 seconds'.

	ORDER_TOTAL	CEIL(ORDER_TOTAL)
1	268651,8	268652
2	62303	62303
3	6394,8	6395
4	13550	13550
5	5451	5451
6	22150,1	22151

Εικόνα 3- 29 Παράδειγμα κλήσης της συνάρτησης CEIL, με προσπέλαση πίνακα του σχήματος OE

**FLOOR:** Επιστρέφει τον μεγαλύτερο ακέραιο, που είναι μικρότερος από ή ίσος με το όρισμα.

```
SELECT FLOOR(15.7) "Floor" FROM DUAL;
```

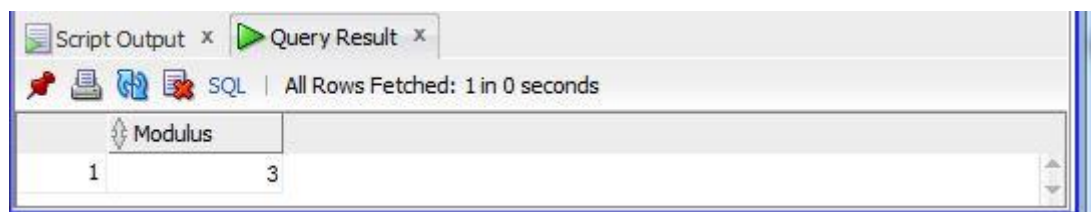


Floor
15

Εικόνα 3- 30 Παράδειγμα της συνάρτησης FLOOR

**MOD:** Επιστρέφει το υπόλοιπο της διαίρεσης του 11 με το 4.

```
SELECT MOD(11,4) "Modulus" FROM DUAL;
```



Modulus
3

Εικόνα 3- 31 Η συνάρτηση MOD, το υπόλοιπο της διαίρεσης του 11 με το 4

**POWER:** Υψώνει μια τιμή σε κάποια δύναμη π.χ. το 3 στο τετράγωνο ( $3^2$ ).

**Σύνταξη:** POWER(*n1* , *n2*), όπου ***n1***, ***n2*** αριθμοί.

Αν το ***n2*** είναι αρνητικός αριθμός, τότε το ***n1*** πρέπει να είναι ακέραιος.

```
SELECT POWER(3,2) "Raised" FROM DUAL;
```



Raised
9

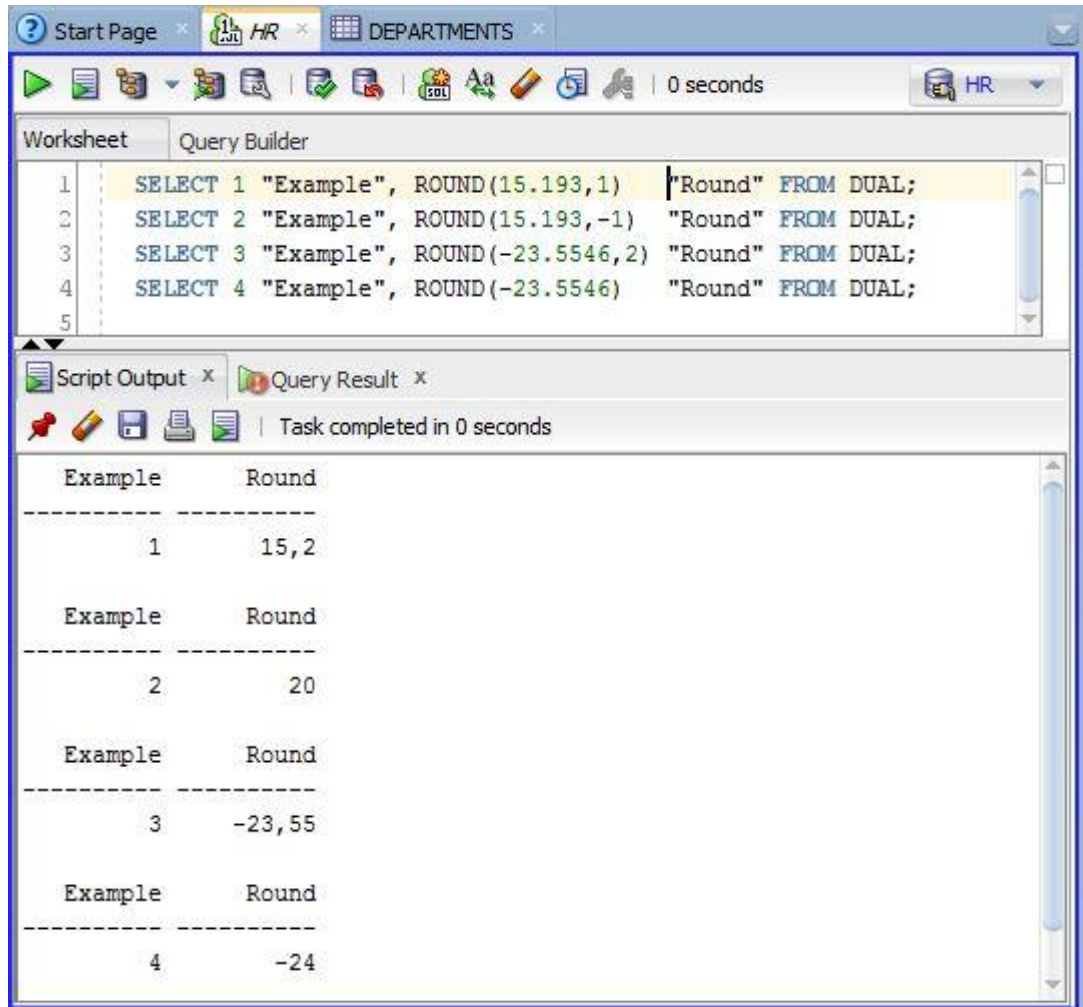
Εικόνα 3- 32 Παράδειγμα της συνάρτησης POWER



ROUND(number): Χρησιμοποιείται για τη στρογγυλοποίηση μιας αριθμητικής τιμής.

Σύνταξη: ROUND(*n* , *int*), όπου *n* αριθμός και *int* ακέραιος.

Αν δεν δοθεί το **int**, εκλαμβάνεται ως μηδέν.



Εικόνα 3- 33 Τέσσερα διαφορετικά παραδείγματα κλήσης της συνάρτησης ROUND

SQRT: επιστρέφει την τετραγωνική ρίζα μιας αριθμητικής τιμής.

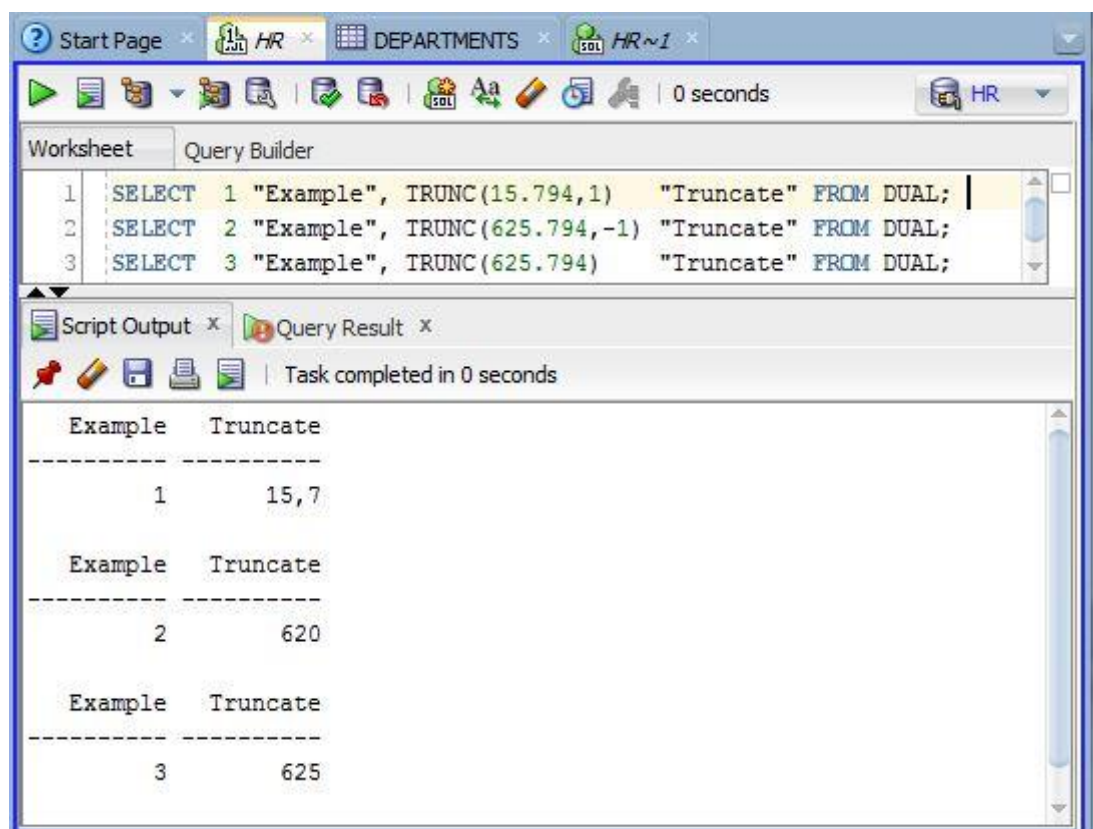
```
SELECT SQRT(26) "Square root" FROM DUAL;
```

**Αποτέλεσμα: 5,09901951359278483002822410902278198956**

TRUNC (number). Περικόπτει μια τιμή σύμφωνα με κάποια ακρίβεια

Σύνταξη: TRUNC (n1,n2).

Αν δεν δοθεί το **n2**, εκλαμβάνεται ως μηδέν.



**Εικόνα 3- 34** Τρία διαφορετικά παραδείγματα κλήσης της συνάρτησης TRUNC

## Αλφαριθμητικές συναρτήσεις που επιστρέφουν αλφαριθμητικά (Character Functions Returning Character Values)

Συνάρτηση	Σύντομη Περιγραφή
<b>CONCAT</b>	Όμοια με τον τελεστή   , συνενώνει δύο αλφαριθμητικά (CONCATenate).
<b>INITCAP</b>	Μετατρέπει το πρώτο γράμμα μιας σειρά λέξεων σε κεφαλαίο.
<b>LOWER</b>	Μετατρέπει όλα τα γράμματα ενός αλφαριθμητικού σε πεζά.
<b>LPAD</b>	Προσθέτει αριστερά του αλφαριθμητικού, έναν αριθμό χαρακτήρων, έτσι ώστε να αποκτήσει ένα συγκεκριμένο μέγεθος.
<b>LTRIM</b>	Αφαιρεί τους προσδιοριζόμενους χαρακτήρες, από το αριστερό άκρο ενός αλφαριθμητικού.
<b>RPAD</b>	Προσθέτει δεξιά του αλφαριθμητικού, έναν αριθμό χαρακτήρων, έτσι ώστε να αποκτήσει ένα συγκεκριμένο μέγεθος.
<b>RTRIM</b>	Αφαιρεί τους προσδιοριζόμενους χαρακτήρες, από το δεξιό άκρο ενός αλφαριθμητικού.
<b>SUBSTR</b>	Αποκόπτει ένα τμήμα από το αλφαριθμητικό.
<b>UPPER</b>	Μετατρέπει όλα τα γράμματα ενός αλφαριθμητικού σε κεφαλαία.

Πίνακας 3- 6 Αλφαριθμητικές συναρτήσεις που επιστρέφουν αλφαριθμητικά

Παραδείγματα:

CONCAT: Χρησιμοποιείται για τη συνένωση δύο αλφαριθμητικών, αντί για τον τελεστή ||

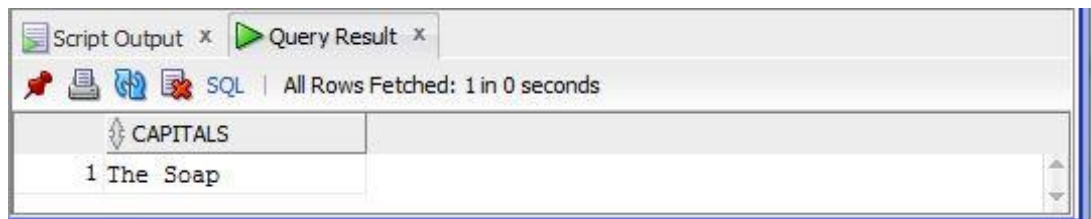
```
SELECT CONCAT(CONCAT(last_name, ''s job category is '),
               job_id) Job
FROM employees
WHERE employee_id = 152;
```



Εικόνα 3- 35 Συνένωση ενός κυριολεκτικού αλφαριθμητικού με ένα πεδίο

**INITCAP:** Μετατρέπει το πρώτο γράμμα κάθε λέξης σε κεφαλαίο.

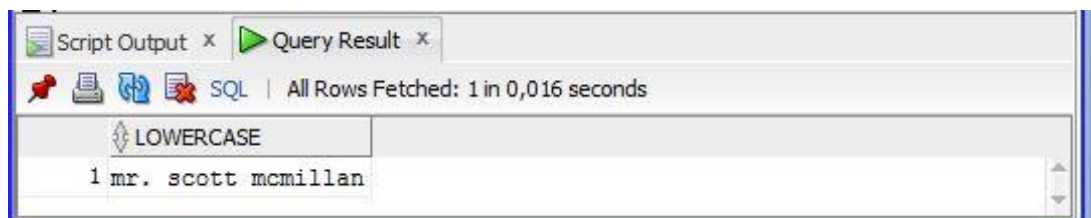
```
SELECT INITCAP('the soap') Capitals FROM DUAL;
```



Εικόνα 3- 36 Παράδειγμα κλήσης της συνάρτησης INITCAP

**LOWER:** Μετατρέπει όλα τα γράμματα ενός αλφαριθμητικού σε πεζά.

```
SELECT LOWER('MR. SCOTT MCMILLAN') Lowercase FROM DUAL;
```



Εικόνα 3- 37 Παράδειγμα κλήσης της συνάρτησης LOWER

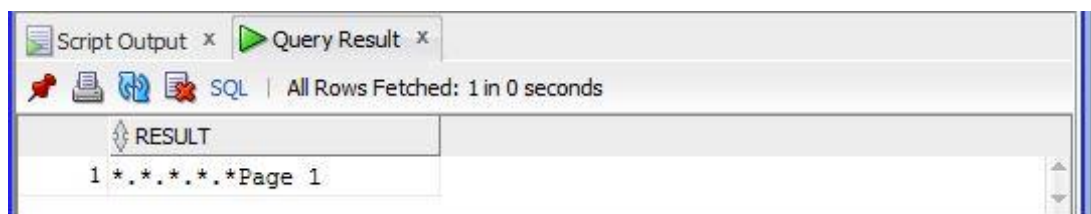
**LPAD:** Προσθέτει αριστερά του αλφαριθμητικού, έναν αριθμό χαρακτήρων, έτσι ώστε να αποκτήσει ένα συγκεκριμένο μέγεθος.

**Σύνταξη:** LPAD(expr1, n, expr2), όπου **expr1** και **expr2** αλφαριθμητικά και το **n** θετικός ακέραιος, που δείχνει το συνολικό μέγεθος της επιστρεφόμενης τιμής.

Αν δεν δοθεί το **expr2**, εκλαμβάνεται ως ένας κενός χαρακτήρας.

Αν το **expr1** είναι μεγαλύτερο από το **n**, επιστρέφεται τ' αντίστοιχο τμήμα του.

```
SELECT LPAD('Page 1',15,'*.') AS Result FROM DUAL;
```



Εικόνα 3- 38 Παράδειγμα κλήσης της συνάρτησης LPAD

**RPAD:** Προσθέτει δεξιά του αλφαριθμητικού, έναν αριθμό χαρακτήρων, έτσι ώστε να αποκτήσει ένα συγκεκριμένο μέγεθος.

**Σύνταξη:** RPAD(expr1, n, expr2), όπου **expr1** και **expr2** αλφαριθμητικά και το **n** θετικός ακέραιος, που δείχνει το συνολικό μέγεθος της επιστρεφόμενης τιμής.

Το **expr1** δεν μπορεί να είναι NULL.

Αν δεν δοθεί το **expr2**, εκλαμβάνεται ως ένας κενός χαρακτήρας.

Αν το **expr1** είναι μεγαλύτερο από το **n**, επιστρέφεται το αντίστοιχο τμήμα του.

```
SELECT last_name, RPAD(' ', salary/1000/1, '*') Salary
FROM employees
WHERE department_id = 60
ORDER BY last_name, Salary;
```



	LAST_NAME	SAL	SALARY
1	Austin	4800	***
2	Ernst	6000	*****
3	Hunold	9000	*****
4	Lorentz	4200	***
5	Pataballa	4800	***

Εικόνα 3- 39 Παράδειγμα κλήσης της συνάρτησης RPAD

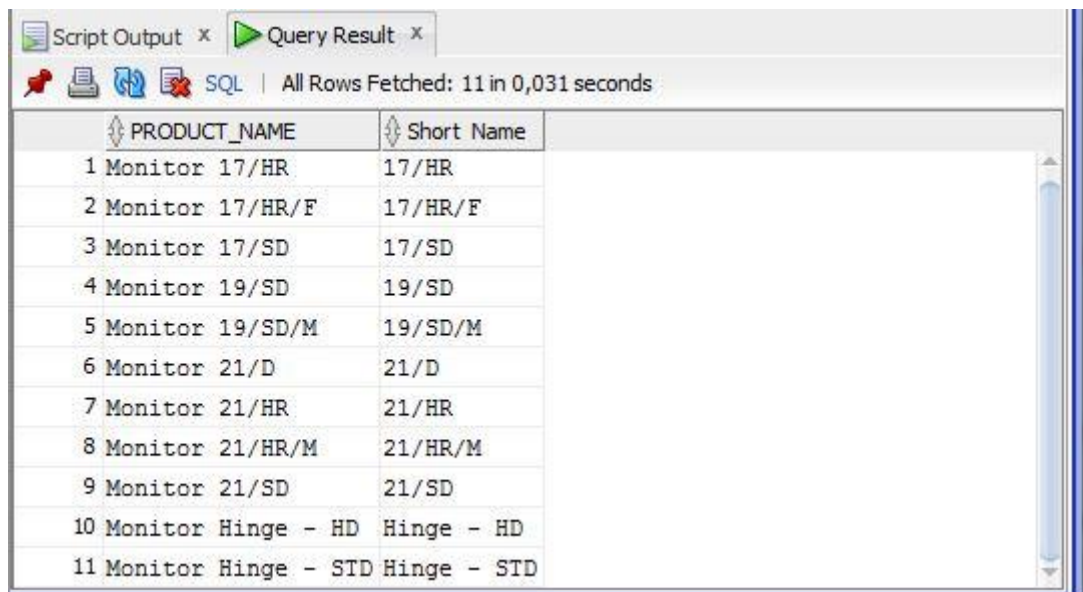
**LTRIM:** Αφαιρεί τους προσδιοριζόμενους χαρακτήρες, από το αριστερό άκρο ενός αλφαριθμητικού.

**Σύνταξη:** LTRIM (char, set), όπου **char** και **set** είναι αλφαριθμητικά.

Αν δεν δοθεί το **set**, εκλαμβάνεται ως ένας κενός χαρακτήρας.

```
SELECT product_name, LTRIM(product_name, 'Monitor ')
                        "Short Name "
FROM OE.product_information
WHERE product_name LIKE 'Monitor%';
```

Τ' αποτέλεσμα φαίνεται στην επόμενη σελίδα.



	PRODUCT_NAME	Short Name
1	Monitor 17/HR	17/HR
2	Monitor 17/HR/F	17/HR/F
3	Monitor 17/SD	17/SD
4	Monitor 19/SD	19/SD
5	Monitor 19/SD/M	19/SD/M
6	Monitor 21/D	21/D
7	Monitor 21/HR	21/HR
8	Monitor 21/HR/M	21/HR/M
9	Monitor 21/SD	21/SD
10	Monitor Hinge - HD	Hinge - HD
11	Monitor Hinge - STD	Hinge - STD

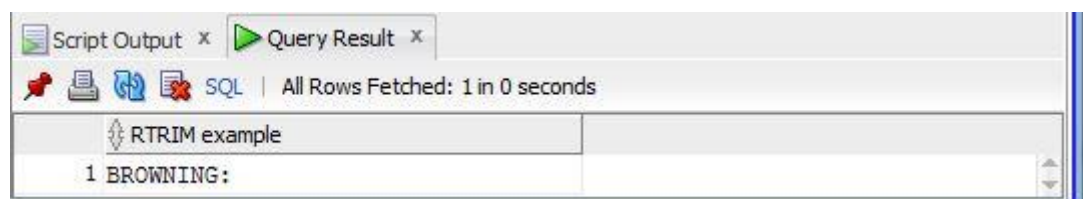
Εικόνα 3- 40 Παράδειγμα κλήσης της συνάρτησης LTRIM

**RTRIM:** Αφαιρεί τους προσδιοριζόμενους χαρακτήρες, από το δεξιό άκρο ενός αλφαριθμητικού.

**Σύνταξη:** RTRIM(char, set), όπου **char** και **set** είναι αλφαριθμητικά.

Αν δεν δοθεί το **set**, εκλαμβάνεται ως ένας κενός χαρακτήρας.

```
SELECT RTRIM('BROWNING: ./= ./= ./= ./= ./= ./=.', '/')
       "RTRIM example "
FROM DUAL;
```



RTRIM example
1 BROWNING:

Εικόνα 3- 41 Παράδειγμα κλήσης της συνάρτησης RTRIM

**SUBSTR:** Αποκόπτει ένα τμήμα από τ' αλφαριθμητικό.

**Σύνταξη:** SUBSTR (char, position, length), όπου **char** αλφαριθμητικό, και τα **position**, **length** ακέραιοι.

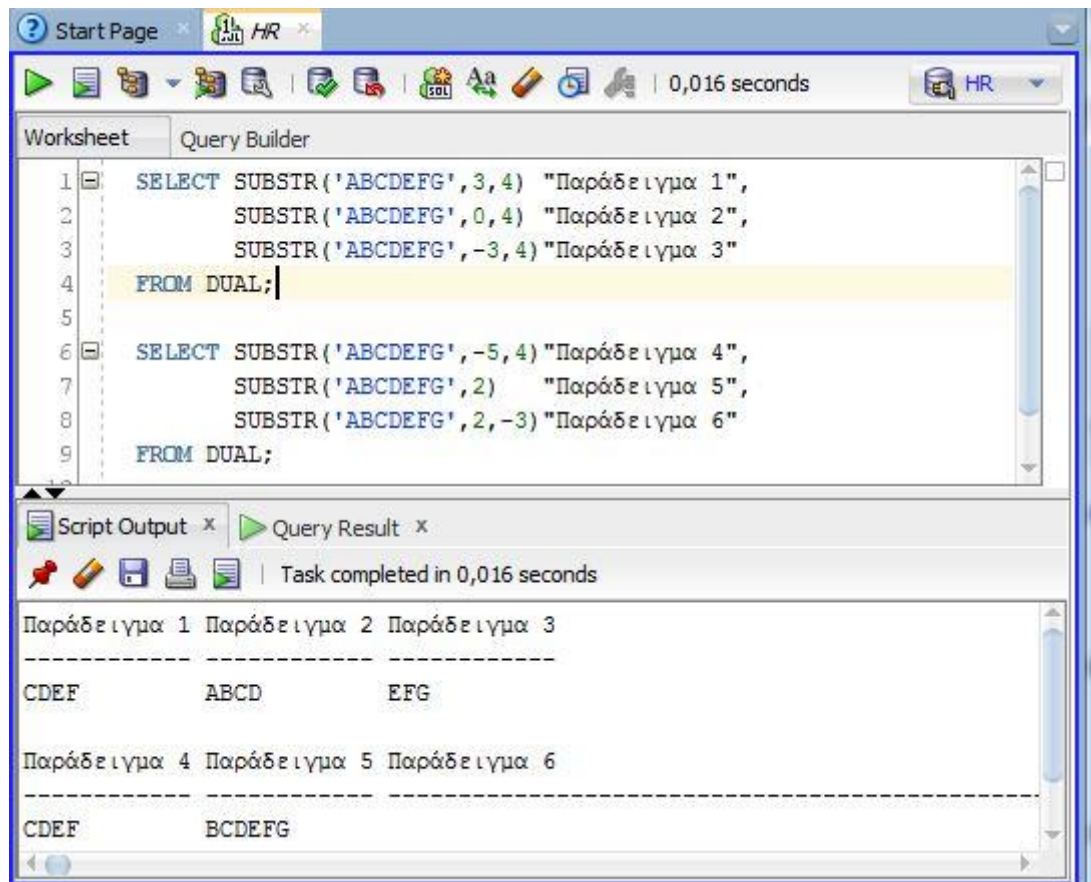
Αν το **position** είναι μηδέν, εκλαμβάνεται ως 1.

Αν το **position** είναι αρνητικό, η Oracle μετράει από το τέλος προς την αρχή του αλφαριθμητικού.

Αν δεν δοθεί το **length**, επιστρέφεται το αλφαριθμητικό από τη θέση **position** μέχρι το τέλος του **char**.



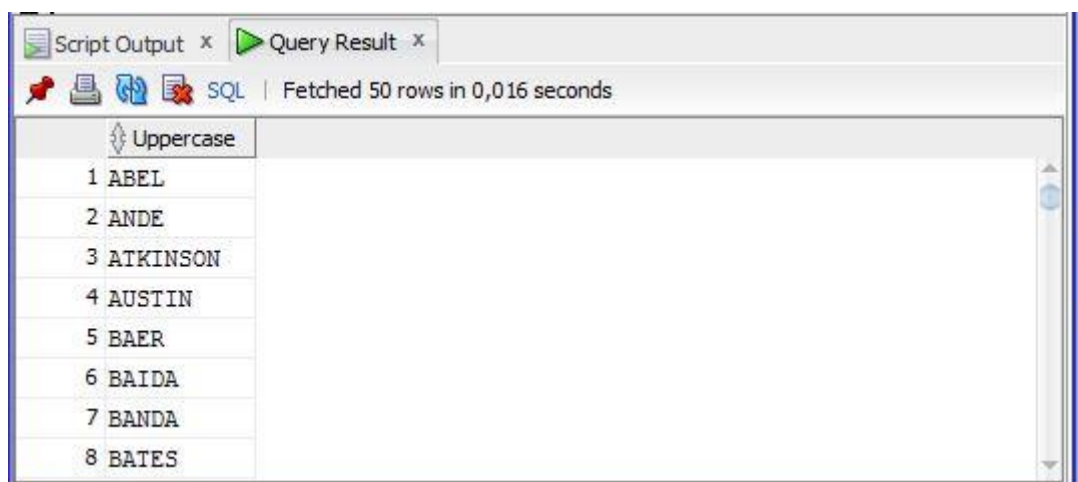
Αν το **length** είναι αρνητικό, επιστρέφει NULL.



Εικόνα 3- 42 Έξι παραδείγματα κλήσης της συνάρτησης SUBSTR

**UPPER:** Μετατρέπει όλα τα γράμματα ενός αλφαριθμητικού σε κεφαλαία.

```
SELECT UPPER(last_name) "Uppercase" FROM employees;
```



Εικόνα 3- 43 Παράδειγμα κλήσης της συνάρτησης UPPER

### Αλφαριθμητικές συναρτήσεις που επιστρέφουν αριθμούς (Character Functions Returning Number Values)

Συνάρτηση	Σύντομη Περιγραφή
<b>INSTR</b>	Εντοπίζει τη θέση ενός αλφαριθμητικού, μέσα σ' ένα άλλο αλφαριθμητικό.
<b>LENGTH</b>	Βρίσκει το μέγεθος ενός αλφαριθμητικού

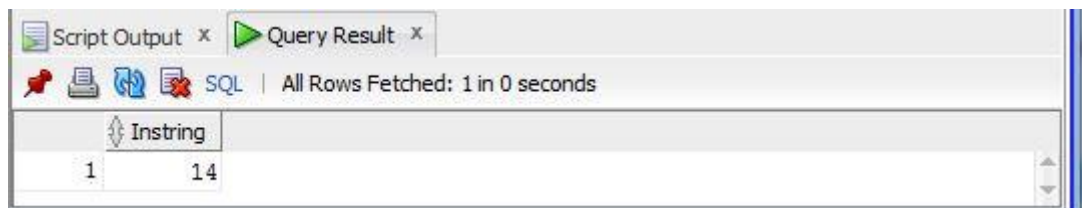
Πίνακας 3- 7 Αλφαριθμητικές συναρτήσεις που επιστρέφουν αριθμούς

Παραδείγματα:

INSTR: Εντοπίζει τη θέση ενός αλφαριθμητικού μέσα σ' ένα άλλο αλφαριθμητικό.

Σύνταξη: INSTR(string, substring, position, occurrence), όπου **string** και **substring** αλφαριθμητικά, το **position** ακέραιος (εκτός από το μηδέν) και το **occurrence** θετικός ακέραιος.

```
SELECT INSTR('CORPORATE FLOOR','OR', 3, 2) "Instring"
FROM DUAL;
```

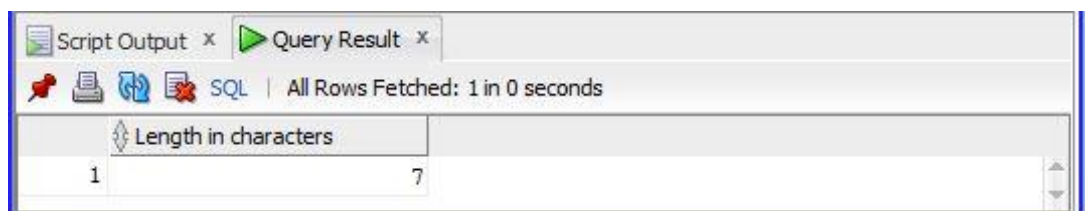


Instring
14

Εικόνα 3- 44 Παράδειγμα κλήσης της συνάρτησης INSTR

LENGTH: Βρίσκει το μέγεθος ενός αλφαριθμητικού

```
SELECT LENGTH('CANDIDE') "Length in characters" FROM DUAL;
```



Length in characters
7

Εικόνα 3- 45 Παράδειγμα κλήσης της συνάρτησης LENGTH



### Συναρτήσεις χειρισμού Ημερομηνιών (Datetime Functions)

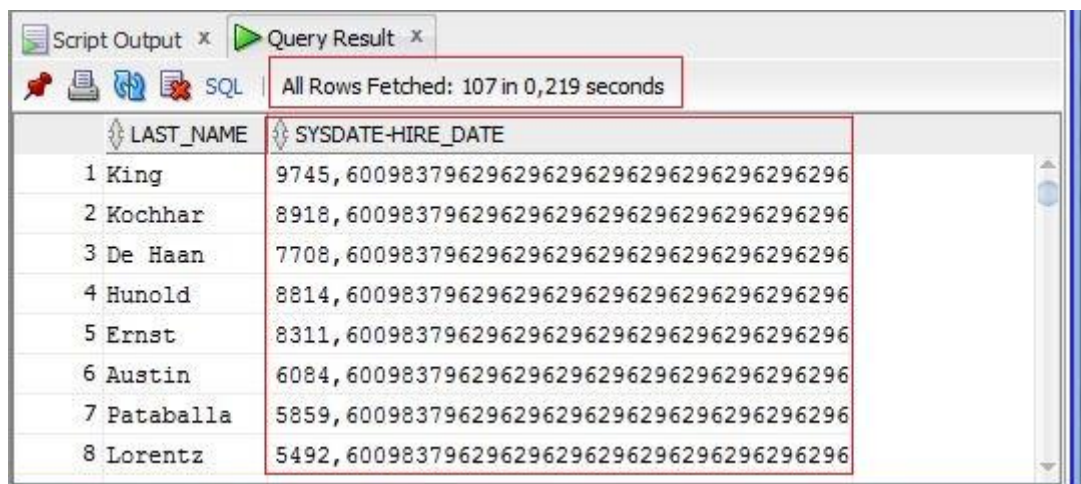
Ένα από τα ισχυρά σημεία της Oracle (φυσικά και άλλων RDBMS) είναι η δυνατότητα που έχει, να αποθηκεύει, να υπολογίζει και να πραγματοποιεί αριθμητική με ημερομηνίες. Επίσης, η Oracle μας επιτρέπει να μορφοποιούμε τις ημερομηνίες, ουσιαστικά με οποιονδήποτε τρόπο μπορούμε να φανταστούμε.

Όπως είπαμε, έχουμε τη δυνατότητα να κάνουμε αριθμητική με ημερομηνίες. Πολύ συχνά χρησιμοποιούμε τις πράξεις της πρόσθεσης και της αφαίρεσης.

Έστω ότι στο δοκιμαστικό μας σχήμα θέλουμε να βρούμε πόσες ημέρες έχουν περάσει από την πρόσληψη ενός υπαλλήλου μέχρι σήμερα. Το μόνο που χρειάζεται να κάνουμε είναι να αφαιρέσουμε από την ημερομηνία πρόσληψης τη σημερινή. Την τρέχουσα ημερομηνία μπορούμε να την πάρουμε με τη χρήση της συνάρτησης SYSDATE. Η πρώτη εντολή που ακολουθεί εμφανίζει την τρέχουσα ημερομηνία ενώ η δεύτερη βρίσκει τη ζητούμενη διαφορά, για όλους τους εργαζομένους της εταιρείας.

```
SELECT SYSDATE FROM DUAL;
```

```
SELECT last_name, SYSDATE - hire_date FROM employees;
```



The screenshot shows a query result window with two tabs: 'Script Output' and 'Query Result'. The 'Query Result' tab is active, displaying the results of the query 'SELECT last\_name, SYSDATE - hire\_date FROM employees;'. The status bar indicates 'All Rows Fetched: 107 in 0,219 seconds'. The results are shown in a table with two columns: 'LAST\_NAME' and 'SYSDATE-HIRE\_DATE'. The data is as follows:

LAST_NAME	SYSDATE-HIRE_DATE
1 King	9745,600983796296296296296296296296296
2 Kochhar	8918,600983796296296296296296296296296
3 De Haan	7708,600983796296296296296296296296296
4 Hunold	8814,600983796296296296296296296296296
5 Ernst	8311,600983796296296296296296296296296
6 Austin	6084,600983796296296296296296296296296
7 Pataballa	5859,600983796296296296296296296296296
8 Lorentz	5492,600983796296296296296296296296296

Εικόνα 3- 46 Αφαίρεση ημερομηνιών

Στον πίνακα παρουσιάζονται συνοπτικά ορισμένες συναρτήσεις χειρισμού ημερομηνιών.

Συνάρτηση	Σύντομη Περιγραφή
<b>ADD_MONTHS</b>	Προσθέτει σε μια ημερομηνία, ένα πλήθος μηνών.
<b>LAST_DAY</b>	Επιστρέφει την ημερομηνία της τελευταίας ημέρας του μήνα, στον οποίο αναφέρεται το όρισμά της.
<b>MONTHS_BETWEEN</b>	Επιστρέφει τη διαφορά μεταξύ δύο ημερομηνιών σε μήνες. Το αποτέλεσμα μπορεί να έχει δεκαδικό μέρος.
<b>NEXT_DAY</b>	Επιστρέφει την ημερομηνία της επόμενης ημέρας από την προσδιοριζόμενη ημερομηνία.
<b>ROUND(ημερομηνία,μορφή)</b>	Σε περίπτωση που δεν καθοριστεί η μορφή, στρογγυλοποιεί την ημερομηνία ως εξής: εάν η ώρα είναι πριν από το μεσημέρι, στις 12 Α.Μ (μεσάνυχτα), αλλιώς στρογγυλοποιεί την ημερομηνία στην επόμενη ημέρα.
<b>TRUNC(ημερομηνία,μορφή)</b>	Σε περίπτωση που δεν καθοριστεί η μορφή, ορίζει την ημερομηνία στα μεσάνυχτα (αρχή της ίδιας ημέρας).
<b>TO_CHAR, TO_DATE</b>	Είναι δύο συναρτήσεις «μετατροπής» που χρησιμοποιούνται πολύ συχνά όταν δουλεύουμε με ημερομηνίες. Θα εξεταστούν αναλυτικότερα στην αντίστοιχη ενότητα.

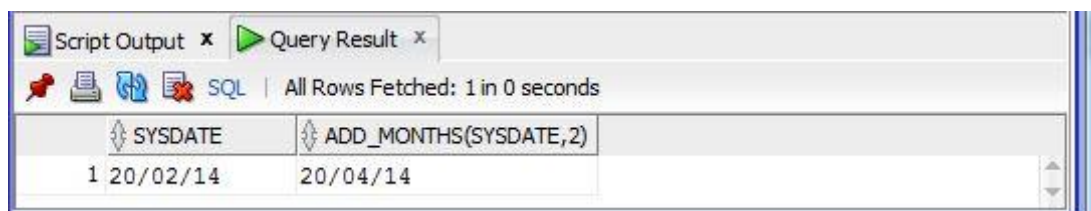
Πίνακας 3- 8 Συναρτήσεις χειρισμού ημερομηνιών

Παραδείγματα συναρτήσεων χειρισμού ημερομηνιών:

**ADD\_MONTHS:** Προσθέτει σε μια ημερομηνία ένα πλήθος μηνών.

**Σύνταξη:** ADD\_MONTHS (date, int), όπου **date** ημερομηνία και **int** ακέραιος.

```
SELECT sysdate, add_months(sysdate,2) FROM DUAL;
```



SYSDATE	ADD_MONTHS(SYSDATE,2)
1 20/02/14	20/04/14

Εικόνα 3- 47 Παράδειγμα κλήσης της συνάρτησης ADD\_MONTHS

### DBTIMEZONE:

```
SELECT DBTIMEZONE FROM DUAL;
```



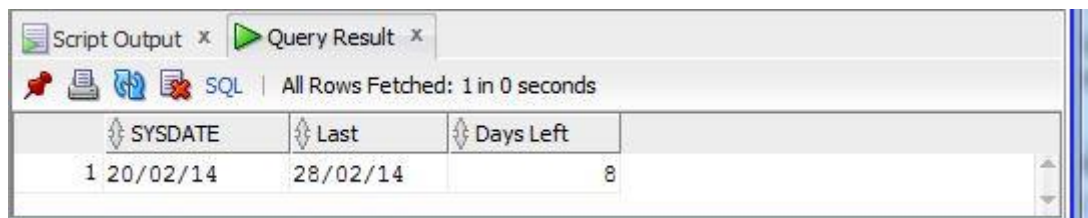
The screenshot shows the 'Query Result' window in SQL Developer. The title bar indicates 'Script Output x' and 'Query Result x'. Below the title bar, there are icons for 'SQL' and 'All Rows Fetched: 1 in 0 seconds'. The main area displays a table with one column named 'DBTIMEZONE' and one row with the value '1 +03:00'.

DBTIMEZONE
1 +03:00

Εικόνα 3- 48 Παράδειγμα κλήσης της συνάρτησης DBTIMEZONE

LAST DAY: Επιστρέφει την ημερομηνία της τελευταίας ημέρας του μήνα, στον οποίο αναφέρεται το όρισμά της.

```
SELECT SYSDATE, LAST_DAY(SYSDATE) "Last",  
       LAST_DAY(SYSDATE) - SYSDATE "Days Left"  
FROM DUAL;
```



The screenshot shows the 'Query Result' window in SQL Developer. The title bar indicates 'Script Output x' and 'Query Result x'. Below the title bar, there are icons for 'SQL' and 'All Rows Fetched: 1 in 0 seconds'. The main area displays a table with three columns: 'SYSDATE', 'Last', and 'Days Left'. The first row contains the values '1 20/02/14', '28/02/14', and '8'.

SYSDATE	Last	Days Left
1 20/02/14	28/02/14	8

Εικόνα 3- 49 Παράδειγμα κλήσης της συνάρτησης LAST\_DAY

### ROUND(Date):

Σύνταξη: ROUND(date, fmt).

```
SELECT ROUND (TO_DATE ('27/10/2000', 'dd/mm/yyyy'), 'YEAR')  
       "New Year"  
FROM DUAL;
```



The screenshot shows the 'Query Result' window in SQL Developer. The title bar indicates 'Script Output x' and 'Query Result x'. Below the title bar, there are icons for 'SQL' and 'All Rows Fetched: 1 in 0 seconds'. The main area displays a table with one column named 'New Year' and one row with the value '1 01/01/01'.

New Year
1 01/01/01

Εικόνα 3- 50 Παράδειγμα κλήσης της συνάρτησης ROUND, με όρισμα ημερομηνία

### Συναρτήσεις μετατροπής (Conversion Functions)

Οι συναρτήσεις μετατροπής, μετατρέπουν τα δεδομένα που χειρίζονται, από έναν τύπο δεδομένων σ' έναν άλλο. Στον πίνακα παρουσιάζονται τρεις απλές και ταυτόχρονα σημαντικές συναρτήσεις μετατροπής.

Συνάρτηση	Σύντομη Περιγραφή
<b>TO_CHAR</b> (ημερομηνία)	Μετατρέπει σε αλφαριθμητικό μια τιμή τύπου DATE.
<b>TO_CHAR</b> (αριθμός)	Μετατρέπει σε αλφαριθμητικό μια τιμή τύπου NUMBER.
<b>TO_DATE</b> (αλφαριθμητικό)	Μετατρέπει σε ημερομηνία (DATE) μια τιμή τύπου NUMBER, VARCHAR2 ή CHAR
<b>TO_NUMBER</b> (αλφαριθμητικό)	Μετατρέπει σε αριθμό μια τιμή τύπου VARCHAR2 ή CHAR.

Πίνακας 3- 9 Συναρτήσεις μετατροπής

Παραδείγματα συναρτήσεων μετατροπής:

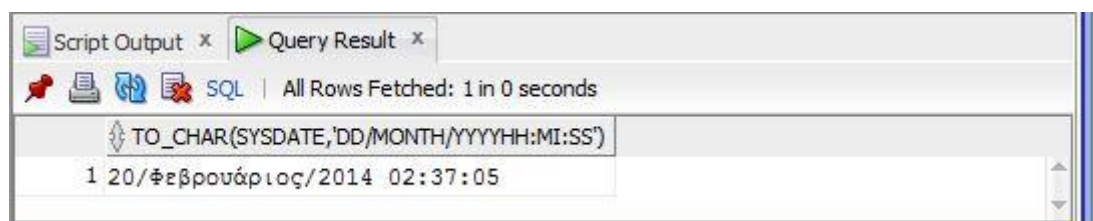
**TO\_CHAR:** Μετατρέπει μια τιμή τύπου DATE ή NUMBER σε αλφαριθμητικό.

```
SELECT TO_CHAR(-10000, 'L99G999D99MI') "Amount"
FROM DUAL;
```



Εικόνα 3- 51 Κλήση της συνάρτησης TO\_CHAR, με όρισμα αριθμό

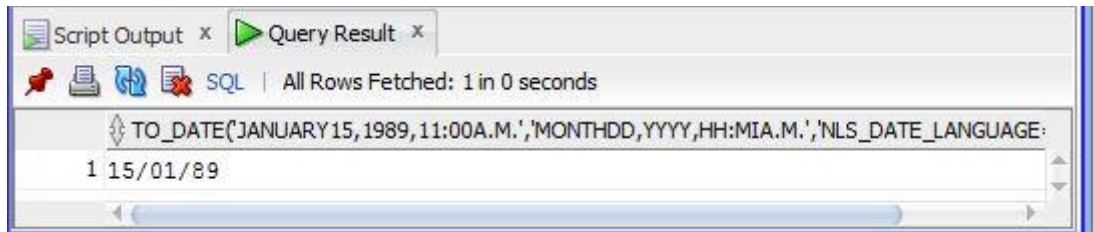
```
SELECT TO_CHAR (sysdate, 'dd/Month/yyyy hh:mi:ss' )
FROM DUAL;
```



Εικόνα 3- 52 Κλήση της συνάρτησης TO\_CHAR, με όρισμα ημερομηνία

**TO\_DATE:** Μετατρέπει σε ημερομηνία (DATE) μια τιμή τύπου NUMBER, VARCHAR2 ή CHAR.

```
SELECT TO_DATE( 'January 15, 1989, 11:00 A.M.', 'Month dd,
              YYYY, HH:MI A.M. ', 'NLS_DATE_LANGUAGE = American')
FROM DUAL;
```



Εικόνα 3- 53 Παράδειγμα κλήσης της συνάρτησης TO\_DATE

### Η συνάρτηση DECODE

Με τη συνάρτηση DECODE μπορούμε να αποκωδικοποιούμε αλφαριθμητικές ή αριθμητικές τιμές, σε διαφορετικά αλφαριθμητικά ή αριθμούς, ανάλογα με τις παραμέτρους που χρησιμοποιούμε. Είναι πολύ ισχυρή συνάρτηση για την εκτέλεση εντολών υπό συνθήκες.

Η σύνταξη της DECODE είναι:

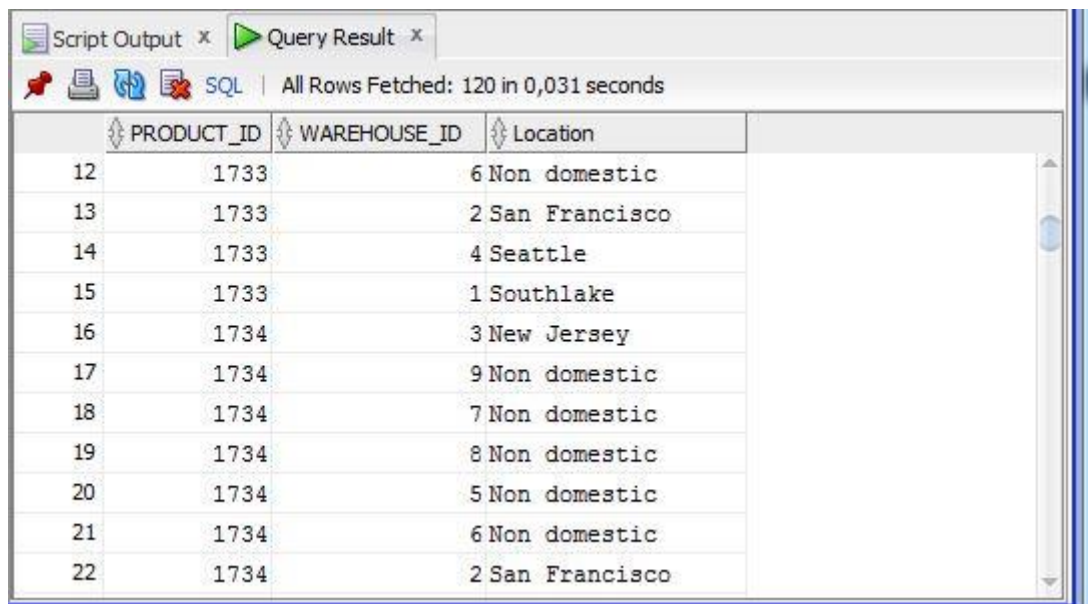
DECODE (expr, search1, result1, search2, result2,.... ,default)

Η παράμετρος **expr** μπορεί να είναι μια στήλη ή μια έκφραση (αποτέλεσμα ενός υπολογισμού). Εάν η τιμή της ισούται με το **search1** τότε η συνάρτηση επιστρέφει το **result1**, εάν ισούται με το **search2** επιστρέφει το **result2** κ.ο.κ. Αν η τιμή δεν ισούται με κανένα από τα **search** τότε επιστρέφει την τιμή **default**.

Το μέγιστο πλήθος των ορισμάτων που μπορεί να δεχθεί η DECODE είναι 255 (έκδοση 11g).

Παράδειγμα (το αποτέλεσμα εμφανίζεται στην επόμενη σελίδα):

```
SELECT product_id, DECODE (warehouse_id,
                          1, 'Southlake',
                          2, 'San Francisco',
                          3, 'New Jersey',
                          4, 'Seattle',
                          'Non domestic') "Location"
FROM OE.inventories
WHERE product_id < 1775
ORDER BY product_id, "Location";
```



PRODUCT_ID	WAREHOUSE_ID	Location
12	1733	6 Non domestic
13	1733	2 San Francisco
14	1733	4 Seattle
15	1733	1 Southlake
16	1734	3 New Jersey
17	1734	9 Non domestic
18	1734	7 Non domestic
19	1734	8 Non domestic
20	1734	5 Non domestic
21	1734	6 Non domestic
22	1734	2 San Francisco

Εικόνα 3- 54 Παράδειγμα κλήσης της συνάρτησης DECODE

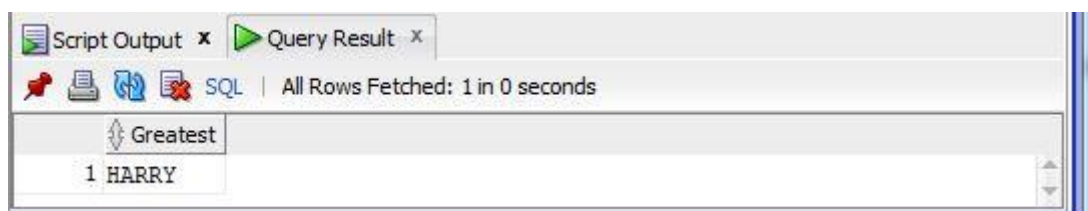
### General Comparison Functions

Πρόκειται για δύο συναρτήσεις, τη GREATEST και τη LEAST, που εφαρμόζονται σε λίστες και βρίσκουν τη μέγιστη και την ελάχιστη τιμή ενός συνόλου τιμών.

Παραδείγματα:

#### GREATEST:

```
SELECT GREATEST ('HARRY', 'HARRIOT', 'HAROLD') "Greatest"
FROM DUAL;
```

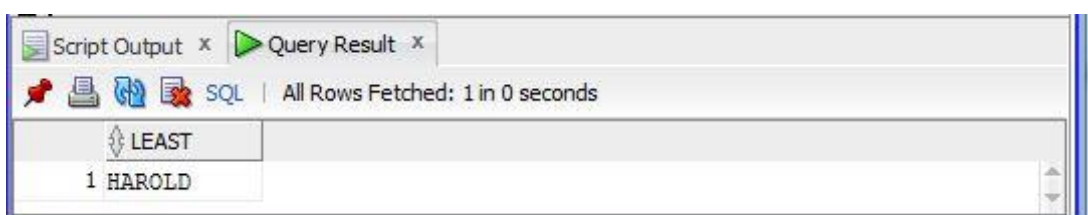


Greatest
1 HARRY

Εικόνα 3- 55 Παράδειγμα κλήσης της συνάρτησης GREATEST

#### LEAST:

```
SELECT LEAST ('HARRY', 'HARRIOT', 'HAROLD') "LEAST"
FROM DUAL;
```



LEAST
1 HAROLD

Εικόνα 3- 56 Παράδειγμα κλήσης της συνάρτησης LEAST



### Συναρτήσεις χειρισμού των NULL (NULL -Related Functions)

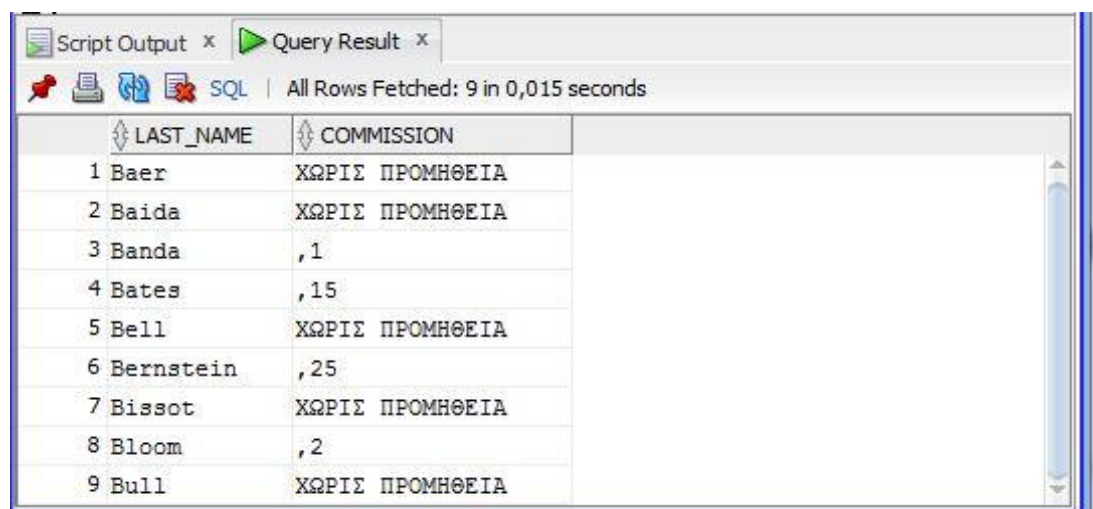
**NVL:** υποκατάσταση NULL τιμών. Έχουμε δει ότι μια τιμή NULL, μπορεί να αναπαριστά μια κενή ή άγνωστη ή άνευ σημασίας τιμή και ότι, όταν εμπλέκεται σε κάποια έκφραση (εκτός από τον || ή τη CONCAT) έχει ως αποτέλεσμα να επιστρέφει όλη η έκφραση μια τιμή NULL. Αυτή η “συμπεριφορά” δεν είναι πάντοτε επιθυμητή. Η Oracle μας παρέχει τη συνάρτηση NVL, με την οποία μπορούμε να χειριστούμε NULL τιμές.

**Σύνταξη:** NVL (τιμή, υποκατάστατο).

Εάν η «τιμή» είναι NULL, τότε η συνάρτηση επιστρέφει το «υποκατάστατο», αλλιώς επιστρέφει την «τιμή». Το «υποκατάστατο» μπορεί να είναι μια σταθερά, μια στήλη ή μια έκφραση. Τα δύο ορίσματα πρέπει να είναι του ίδιου τύπου δεδομένων.

Παράδειγμα:

```
SELECT last_name,  
NVL(TO_CHAR(commission_pct), 'ΧΩΡΙΣ ΠΡΟΜΗΘΕΙΑ') "COMMISSION"  
FROM employees WHERE last_name LIKE 'B%'  
ORDER BY last_name;
```



The screenshot shows a SQL query result window with two tabs: 'Script Output' and 'Query Result'. The 'Query Result' tab is active, displaying a table with two columns: 'LAST\_NAME' and 'COMMISSION'. The table contains 9 rows of data, all filtered by the condition 'last\_name LIKE 'B%'. The 'COMMISSION' column shows the result of the NVL function, where NULL values are replaced by the string 'ΧΩΡΙΣ ΠΡΟΜΗΘΕΙΑ'.

	LAST_NAME	COMMISSION
1	Baer	ΧΩΡΙΣ ΠΡΟΜΗΘΕΙΑ
2	Baida	ΧΩΡΙΣ ΠΡΟΜΗΘΕΙΑ
3	Banda	,1
4	Bates	,15
5	Bell	ΧΩΡΙΣ ΠΡΟΜΗΘΕΙΑ
6	Bernstein	,25
7	Bissot	ΧΩΡΙΣ ΠΡΟΜΗΘΕΙΑ
8	Bloom	,2
9	Bull	ΧΩΡΙΣ ΠΡΟΜΗΘΕΙΑ

Εικόνα 3- 57 Συνδυασμένη κλήση των συναρτήσεων NVL και TO\_CHAR

Μια επέκταση της NVL είναι η NVL2, που μας επιτρέπει να ορίζουμε την επιστρεφόμενη τιμή, ακόμα και αν η τιμή της έκφρασης που ελέγχουμε δεν είναι NULL.

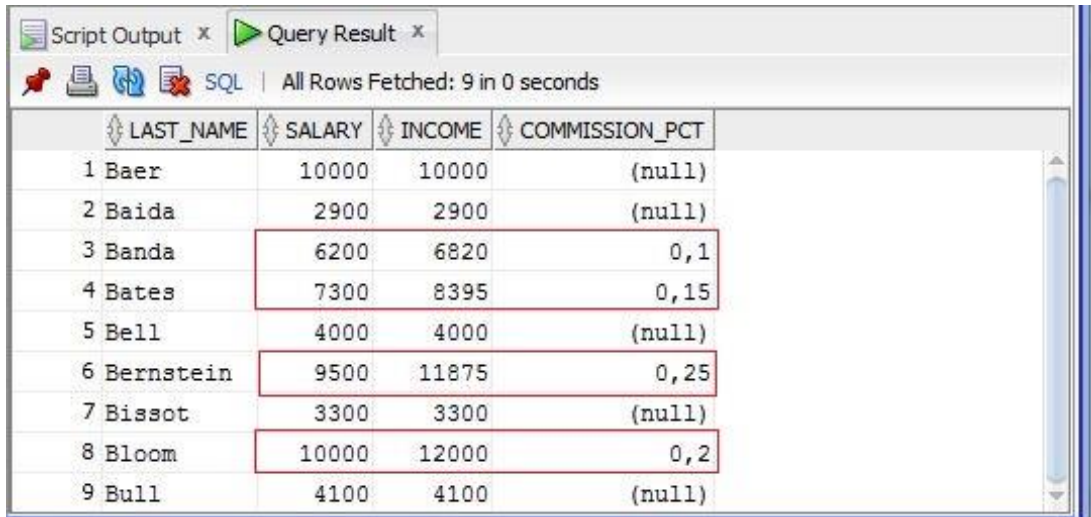
**Σύνταξη:** NVL2 (τιμή, υποκατάστατο1, υποκατάστατο2).

Αν η **τιμή** δεν είναι NULL, επιστρέφει το **υποκατάστατο1**, διαφορετικά επιστρέφει το **υποκατάστατο2**.

```

SELECT last_name, salary
       NVL2(commission_pct, salary + (salary *
commission_pct), salary) income,
       commission_pct
FROM employees
WHERE last_name like 'B%'
ORDER BY last_name;

```



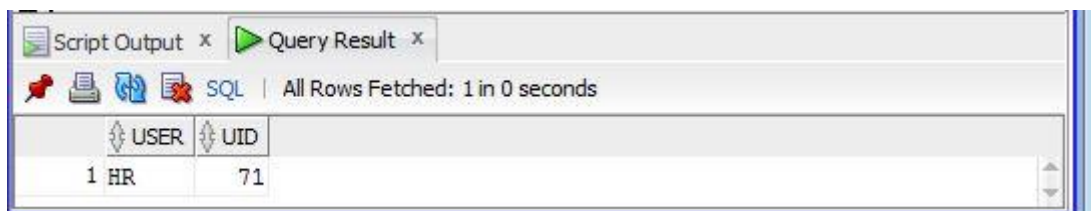
	LAST_NAME	SALARY	INCOME	COMMISSION_PCT
1	Baer	10000	10000	(null)
2	Baida	2900	2900	(null)
3	Banda	6200	6820	0,1
4	Bates	7300	8395	0,15
5	Bell	4000	4000	(null)
6	Bernstein	9500	11875	0,25
7	Bissot	3300	3300	(null)
8	Bloom	10000	12000	0,2
9	Bull	4100	4100	(null)

Εικόνα 3- 58 Παράδειγμα με τη συνάρτηση NVL2

### Οι συναρτήσεις USER και UID (Environment and Identifier Functions)

**USER, UID:** Επιστρέφουν τον χρήστη και έναν μοναδικό κωδικό για τον χρήστη, που είναι συνδεδεμένος σε αυτό το session (σύνοδο), αντίστοιχα.

```
SELECT USER, UID FROM DUAL;
```



	USER	UID
1	HR	71

Εικόνα 3- 59 Οι συναρτήσεις USER και UID



## Συναρτήσεις ομαδοποίησης ή συγκεντρωτικές συναρτήσεις (Aggregate functions)

Μια συνάρτηση ομαδοποίησης ή συγκεντρωτική συνάρτηση, χρησιμοποιείται σε μια εντολή SQL για να παρέχει πληροφορίες σύνοψης, όπως σύνολα, μέσους όρους, τυπικές αποκλίσεις, μέγιστα ή ελάχιστα, και πλήθος. Οι συναρτήσεις αυτές, αντιμετωπίζουν μια ομάδα τιμών σαν μια ολότητα, για παράδειγμα το σύνολο των μισθών όλων των υπαλλήλων της εταιρείας μας.

Οι συναρτήσεις ομαδοποίησης αντιμετωπίζουν τις τιμές NULL διαφορετικά απ' ότι οι συναρτήσεις μίας γραμμής (single row). Υπολογίζουν το αποτέλεσμα που ζητάμε αγνοώντας τις τιμές NULL. Άρα πρέπει να είμαστε προσεκτικοί όταν προσπαθούμε να πάρουμε συγκεντρωτικά αποτελέσματα από ομάδες που περιέχουν NULL τιμές.

Στον παρακάτω πίνακα περιγράφονται περιληπτικά οι βασικές συγκεντρωτικές συναρτήσεις.

Συνάρτηση	Σύντομη Περιγραφή
<b>COUNT</b>	Επιστρέφει το πλήθος των εγγραφών μιας ομάδας.
<b>SUM</b>	Επιστρέφει το άθροισμα των τιμών μιας στήλης ή έκφρασης, μιας ομάδας.
<b>MIN</b>	Επιστρέφει την ελάχιστη τιμή μιας στήλης ή έκφρασης.
<b>MAX</b>	Επιστρέφει τη μέγιστη τιμή μιας στήλης ή έκφρασης.
<b>AVG</b>	Επιστρέφει τον μέσο όρο μιας στήλης ή έκφρασης σε μια ομάδα.
<b>VARIANCE</b>	Η διακύμανση.
<b>STDDEV</b>	Η τυπική απόκλιση.

Πίνακας 3- 10 Συγκεντρωτικές συναρτήσεις

Παραδείγματα:

COUNT: Επιστρέφει το πλήθος των εγγραφών μιας ομάδας.

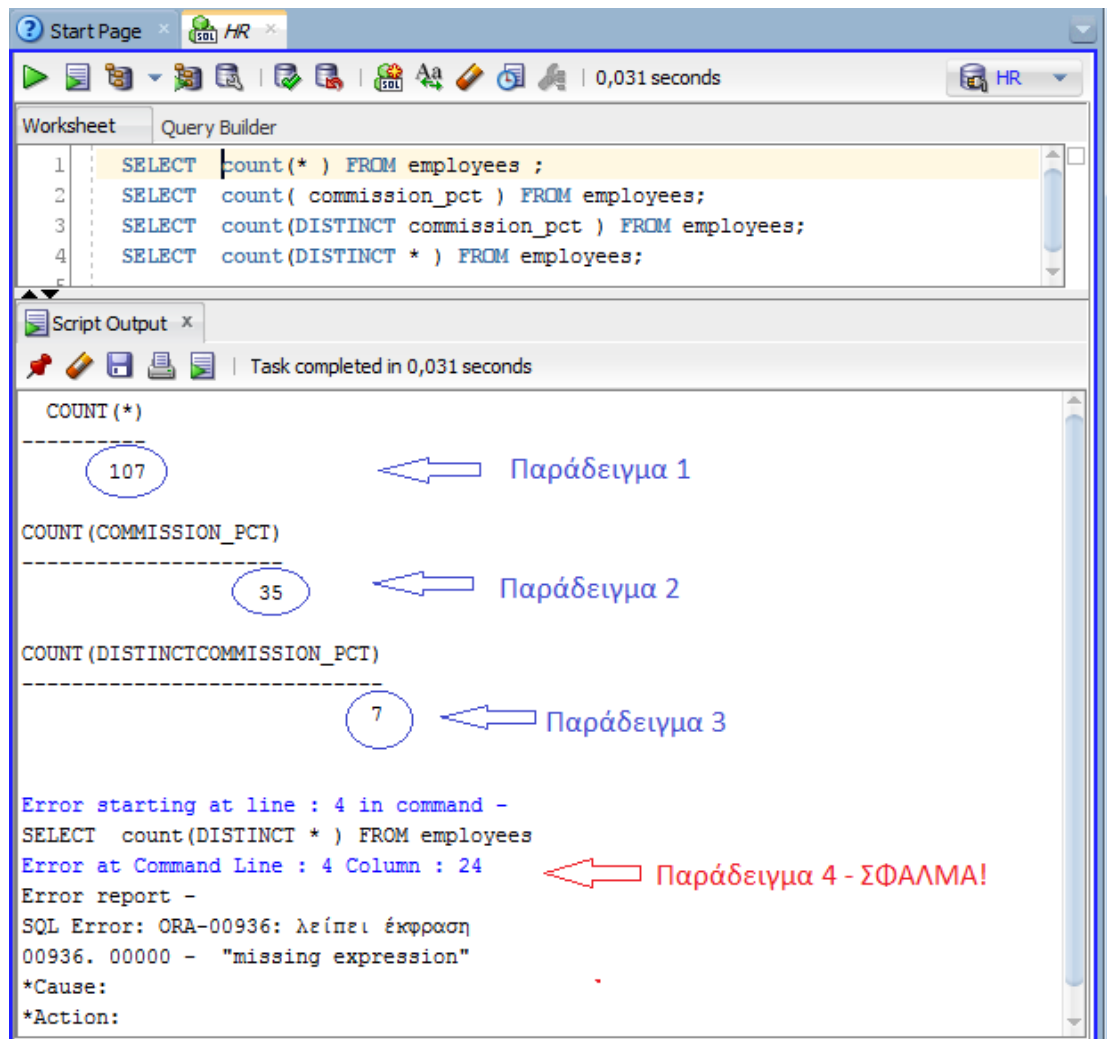
Σύνταξη: COUNT [ (DISTINCT | ALL) ] (FIELD\_NAME) ή COUNT (\*)

Όταν χρησιμοποιείται με μια στήλη ή έκφραση μετρά το πλήθος των γραμμών που δεν περιέχουν τιμή NULL.

Όταν χρησιμοποιείται με τον προσδιοριστή DISTINCT μετράει μόνο τις διακριτές (μοναδικές) τιμές. Το αντίθετο ισχύει με τον προσδιοριστή ALL, που είναι η προεπιλογή (default).

Μπορούμε να χρησιμοποιήσουμε τη COUNT με όρισμα τον αστερίσκο (COUNT(\*)), για να μετρήσουμε όλες τις γραμμές μιας ομάδας είτε περιέχουν στήλες με NULL τιμές είτε όχι.

Ο προσδιοριστής DISTINCT δεν μπορεί να χρησιμοποιηθεί με τη COUNT(\*) .



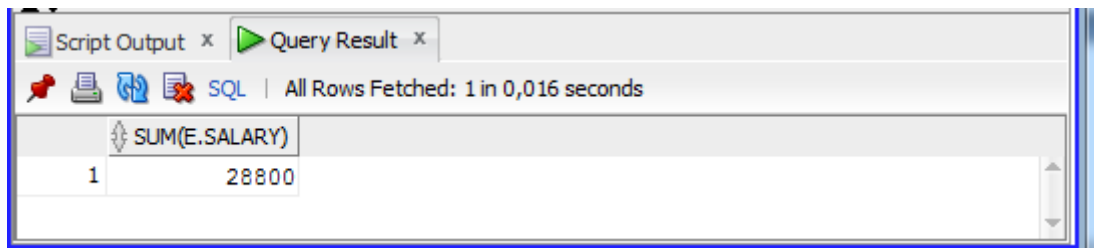
Εικόνα 3- 60 Τέσσερα παραδείγματα της συνάρτησης COUNT

Οι περισσότερες συγκεντρωτικές συναρτήσεις που δέχονται ένα όρισμα, δέχονται τους προσδιοριστές DISTINCT και ALL. Π.χ. ο DISTINCT μέσος όρος των 1,1,1,3 είναι 2 ενώ του ALL 1.5. Η default τιμή είναι ALL.

**SUM:** Επιστρέφει το άθροισμα των τιμών μιας στήλης ή έκφρασης, μιας ομάδας.

Παράδειγμα:

```
SELECT SUM(e.SALARY ) FROM employees e , DEPARTMENTS d
WHERE e.department_id = d.department_id
AND d.DEPARTMENT_NAME LIKE '%IT%';
```



The screenshot shows the 'Query Result' tab in SQL Developer. The query executed is `SUM(E.SALARY)`. The result is displayed in a table with one row and one column, showing the value 28800.

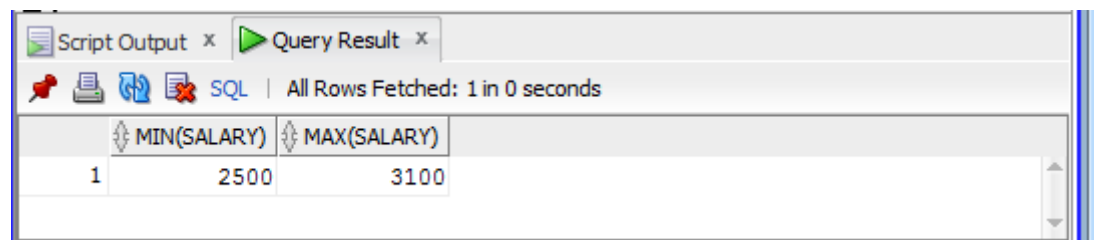
SUM(E.SALARY)
28800

Εικόνα 3- 61 Εύρεση Αθροίσματος με τη συγκεντρωτική συνάρτηση SUM

**MIN, MAX:** Επιστρέφουν τη ελάχιστη και τη μέγιστη τιμή, μιας στήλης ή έκφρασης σε μια ομάδα, αντίστοιχα.

Παράδειγμα:

```
SELECT MIN(salary), MAX (salary) FROM employees
WHERE job_id = 'PU_CLERK' ;
```



The screenshot shows the 'Query Result' tab in SQL Developer. The query executed is `SELECT MIN(salary), MAX (salary) FROM employees WHERE job_id = 'PU_CLERK' ;`. The result is displayed in a table with one row and two columns, showing the values 2500 and 3100.

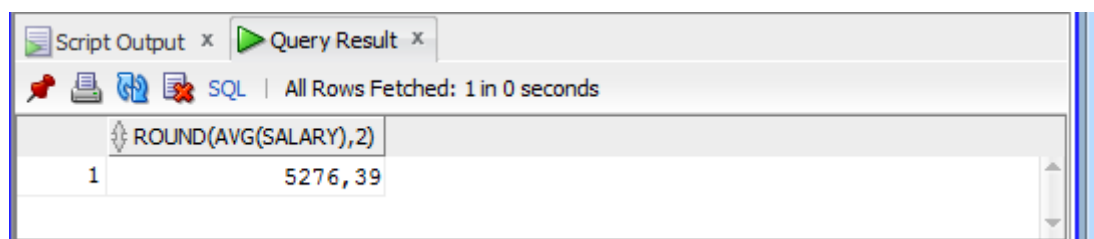
MIN(SALARY)	MAX(SALARY)
2500	3100

Εικόνα 3- 62 Εύρεση ελάχιστου (MIN) και μέγιστου (MAX) σε μια ομάδα

**AVG:** Επιστρέφει τον μέσο όρο μιας στήλης ή έκφρασης σε μια ομάδα.

Παράδειγμα:

```
SELECT ROUND(AVG(salary) ,2) FROM employees
WHERE commission_pct IS NULL;
```



The screenshot shows the 'Query Result' tab in SQL Developer. The query executed is `SELECT ROUND(AVG(salary) ,2) FROM employees WHERE commission_pct IS NULL;`. The result is displayed in a table with one row and one column, showing the value 5276,39.

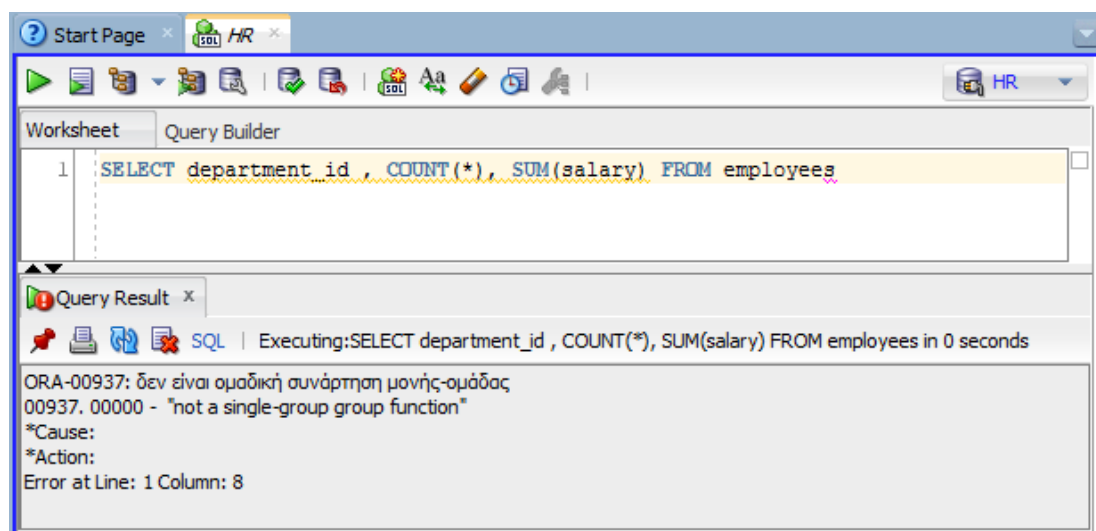
ROUND(AVG(SALARY),2)
5276,39

Εικόνα 3- 63 Εύρεση Μέσου Όρου με τη συγκεντρωτική συνάρτηση AVG

Επιστρέφει τον μέσο όρο των μισθών, των υπαλλήλων που δεν παίρνουν προμήθεια, στρογγυλοποιημένο σε δύο δεκαδικά ψηφία.

## Ομαδοποίηση Δεδομένων (GROUP BY clause)

Όταν λέμε ομαδοποίηση δεδομένων εννοούμε τον συνδυασμό στηλών ή εκφράσεων με τις ίδιες τιμές, σε μια ομάδα. Για παράδειγμα, οι υπάλληλοι της εταιρείας εργάζονται σε διάφορα τμήματα. Σε κάποιες περιπτώσεις μπορεί να χρειαζόμαστε πληροφορίες για κάθε συγκεκριμένο τμήμα, όπως το πλήθος των εργαζομένων ή το συνολικό κόστος μισθοδοσίας. Μια πρώτη σκέψη θα μπορούσε να είναι η εκτέλεση της εντολής: «*SELECT department\_id , COUNT(\*), SUM(salary) FROM employees*». Αν επιχειρήσουμε να τρέξουμε την εντολή, θα λάβουμε ένα μήνυμα λάθους όπως το παρακάτω:

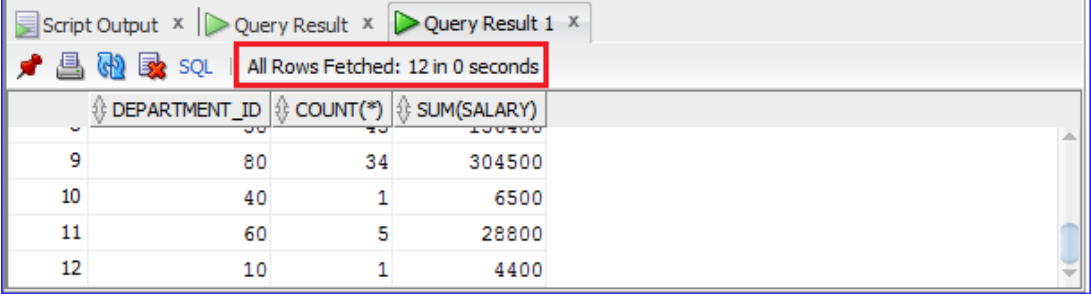


Εικόνα 3- 64 Σφάλμα, γιατί δεν χρησιμοποιήσαμε το GROUP BY clause

Χρειαζόμαστε λοιπόν, έναν τρόπο να ομαδοποιούμε τα δεδομένα μας. Αυτόν τον ρόλο παίζει η φράση **GROUP BY** σε μια πρόταση *SELECT*. Το *GROUP BY* clause ακολουθεί το *WHERE* clause και προηγείται του *ORDER BY*. Η σειρά του *GROUP BY* σε μια εντολή *SELECT* είναι η εξής:

```
SELECT
FROM
WHERE
GROUP BY
ORDER BY
```

Το προηγούμενο πρόβλημα μπορούμε να το λύσουμε με την παρακάτω εντολή: «*SELECT department\_id , COUNT(\*), SUM(salary) FROM employees GROUP BY department\_id*».



DEPARTMENT_ID	COUNT(*)	SUM(SALARY)
9	34	304500
10	1	6500
11	5	28800
12	1	4400

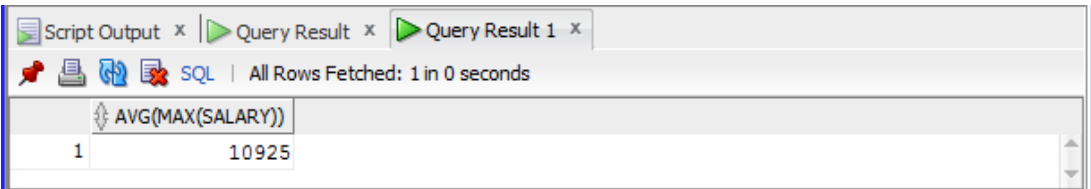
Εικόνα 3- 65 Ενδειγμένη χρήση του GROUP BY clause για την ομαδοποίηση δεδομένων

Το ερώτημα επιστρέφει 12 γραμμές, ενώ ο πίνακας *employees* έχει 107 εγγραφές, διότι ομαδοποιεί τα δεδομένα με βάση τον κωδικό τμήματος.

Οι συγκεντρωτικές συναρτήσεις που χρησιμοποιούμε σε μια εντολή *SELECT*, η οποία έχει *GROUP BY* clause, υπολογίζουν τις τιμές ανά ομάδα και όχι συνολικά για τον πίνακα όπως είδαμε στα παραδείγματα της προηγούμενης ενότητας.

Μπορούμε να έχουμε φωλιασμένες συγκεντρωτικές συναρτήσεις. Έτσι η παρακάτω εντολή θα επιστρέψει μία μόνο γραμμή, διότι υπολογίζει πρώτα τους μέγιστους μισθούς ανά τμήμα και στη συνέχεια τον μέσο όρο αυτών.

```
SELECT AVG(MAX(salary))  
FROM employees  
GROUP BY department_id;
```



AVG(MAX(SALARY))
10925

Εικόνα 3- 66 Φωλιασμένες συγκεντρωτικές συναρτήσεις

## Η φράση HAVING (HAVING clause)

Η πρόταση **HAVING** χρησιμοποιείται σε συνδυασμό με τη *GROUP BY*, προκειμένου να ορίσουμε ποιες ομάδες θα περιλαμβάνονται στ' αποτελέσματα. Η λογική της σχετίζεται μόνο με τ' αποτελέσματα συγκεντρωτικών συναρτήσεων και όχι με στήλες ή με εκφράσεις για μεμονωμένες γραμμές. Το *WHERE* τμήμα θέτει περιορισμούς σε

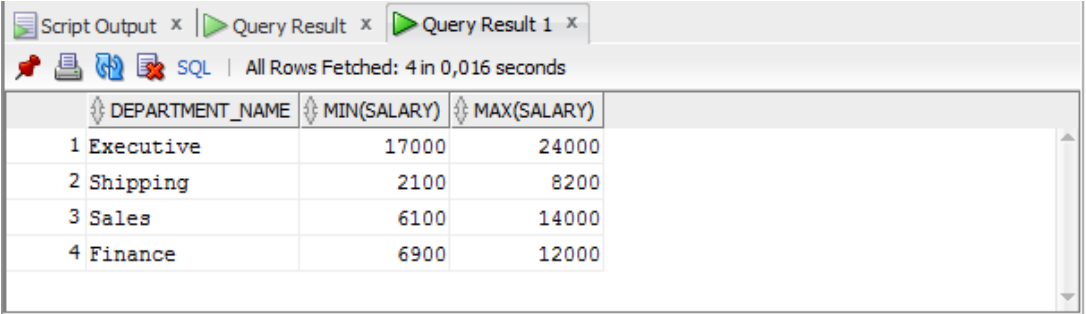
επιλεγμένες στήλες ή εκφράσεις μεμονωμένων γραμμών, ενώ το *HAVING* θέτει περιορισμούς στις ομάδες που δημιουργούνται από το *GROUP BY*.

Η σειρά του *HAVING*, σε μια εντολή *SELECT*, είναι η εξής:

```
SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY
```

Έστω ότι θέλουμε να βρούμε για κάθε τμήμα, τον ελάχιστο και τον μέγιστο μισθό, αλλά ενδιαφερόμαστε μόνο για τα τμήματα που έχουν συνολικό κόστος μισθοδοσίας μεγαλύτερο από 50.000,00€. Η παρακάτω εντολή υλοποιεί αυτό το ερώτημα:

```
SELECT department_name, MIN(salary), MAX (salary)
FROM employees e , departments d
WHERE e.department_id = d.department_id
GROUP BY department_name
HAVING SUM(salary) > 50000;
```



	DEPARTMENT_NAME	MIN(SALARY)	MAX(SALARY)
1	Executive	17000	24000
2	Shipping	2100	8200
3	Sales	6100	14000
4	Finance	6900	12000

Εικόνα 3- 67 Παράδειγμα περιορισμού των ομάδων με το *HAVING* clause

Η σειρά εκτέλεσης που ακολουθεί η Oracle είναι:

1. Επιλογή των εγγραφών με βάση το *WHERE* clause.
2. Ομαδοποίηση των εγγραφών με βάση το *GROUP BY* clause.
3. Υπολογισμός των συγκεντρωτικών συναρτήσεων για κάθε ομάδα.
4. Επιλογή των ομάδων με βάση το *HAVING* clause.
5. Ταξινόμηση των ομάδων με βάση το *ORDER BY* clause. Στο *ORDER BY* τμήμα πρέπει να χρησιμοποιούμε είτε μια στήλη/έκφραση η οποία καθορίζεται στο τμήμα *GROUP BY*, είτε μια συγκεντρωτική συνάρτηση(δεν είναι υποχρεωτικό να υπάρχει και στο *Select list*).





Η σειρά εκτέλεσης είναι σημαντική και επηρεάζει την ταχύτητα εκτέλεσης των ερωτημάτων μας. Πρέπει να εξαλείψουμε όσες περισσότερες εγγραφές μπορούμε, στο *WHERE* τμήμα της εντολής. Έτσι μειώνεται το πλήθος των εγγραφών που θα υποστούν επεξεργασία από το *GROUP BY* τμήμα.

Στη συνέχεια θα δούμε δύο ακόμη παραδείγματα, ώστε να εξοικειωθούμε καλύτερα με τις έννοιες της ενότητας.

#### Παράδειγμα 1:

```
SELECT department_id, MIN(salary), MAX (salary)
FROM employees
WHERE job_id = 'PU_CLERK'
GROUP BY department_id
ORDER BY department_id;
```

Script Output x | Query Result x | Query Result 1 x

    SQL | All Rows Fetched: 1 in 0 seconds

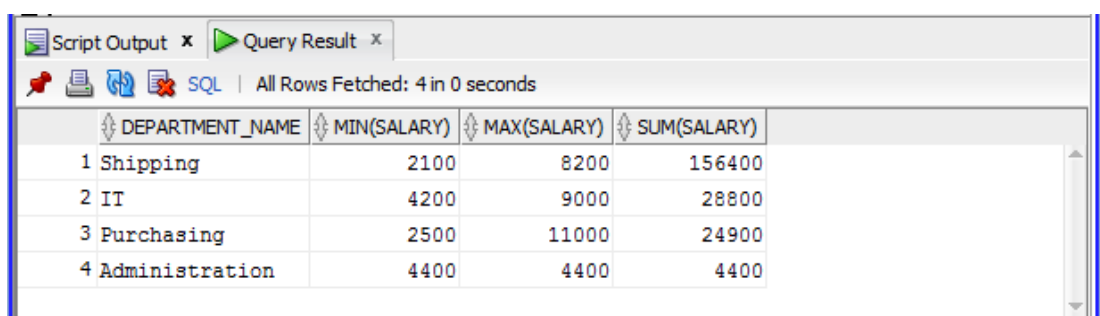
	DEPARTMENT_ID	MIN(SALARY)	MAX(SALARY)
1	30	2500	3100

Εικόνα 3- 68 Παράδειγμα χρήσης συγκεντρωτικών συναρτήσεων

Το ερώτημα αυτό επιστρέφει τον ελάχιστο και τον μέγιστο μισθό ομαδοποιημένα ανά τμήμα, για τους υπαλλήλους που εργάζονται ως 'PU\_CLERK' και ταξινομεί τα αποτελέσματα κατά αύξουσα σειρά του κωδικού τμήματος.

#### Παράδειγμα 2:

```
SELECT department_name, MIN(salary), MAX (salary), SUM(salary)
FROM employees e, departments d
WHERE e.department_id = d.department_id
GROUP BY department_id
HAVING MIN(salary) < 5000
ORDER BY SUM(salary) DESC;
```



DEPARTMENT_NAME	MIN(SALARY)	MAX(SALARY)	SUM(SALARY)
1 Shipping	2100	8200	156400
2 IT	4200	9000	28800
3 Purchasing	2500	11000	24900
4 Administration	4400	4400	4400

Εικόνα 3- 69 Παράδειγμα χρήσης συγκεντρωτικών συναρτήσεων και περιορισμού των ομάδων

Το ερώτημα αυτό επιστρέφει τον ελάχιστο και τον μέγιστο μισθό καθώς και το άθροισμα των μισθών ανά τμήμα, μόνο για τα τμήματα που έχουν εργαζόμενους οι οποίοι αμείβονται με λιγότερο από 5.000,00€ (HAVING MIN(salary) < 5000) και ταξινομεί τ' αποτέλεσμα κατά φθίνουσα σειρά του αθροίσματος των μισθών (ORDER BY SUM(salary) DESC).

### Ανάκτηση δεδομένων από περισσότερους πίνακες ( joins)

Μέχρι τώρα, όλα τα ερωτήματα που έχουμε εξετάσει ανακτούν τα δεδομένα από ένα μόνο πίνακα ή όψη. Ένα από τα δυνατά σημεία της SQL, είναι η δυνατότητα να επιλέγει δεδομένα από πολλούς πίνακες ή/και όψεις ταυτόχρονα. Όπως έχει αναφερθεί, κατά τη διαδικασία της κανονικοποίησης, μια σχεσιακή βάση δεδομένων διαιρείται σε μικρότερους πίνακες που συνδέονται μεταξύ τους με τα κατάλληλα κλειδιά (πρωτεύοντα και ξένα). Αυτό έχει ως αποτέλεσμα τα δεδομένα που χρειαζόμαστε σε κάποιο ερώτημα, να βρίσκονται συνήθως αποθηκευμένα σε πολλούς πίνακες.

Μια *συνένωση (join)*, συνδυάζει δύο ή περισσότερους πίνακες προκειμένου να ανακτηθούν τα ζητούμενα δεδομένα.

Οι τύποι συνενώσεων που θα μας απασχολήσουν είναι οι παρακάτω:

- *Συνενώσεις ισότητας (ισοσυνδέσεις – Equijoins-Inner Joins)*
- *Συνενώσεις ανισότητας και Antijoins*
- *Ημί-ενώσεις (Semijoins)*
- *Εξωτερικές συνενώσεις (Outer Joins)*
- *Αυτό-ενώσεις (Self Joins)*
- *Καρτεσιανά γινόμενα (Cartesian Products)*

Όταν θέλουμε να συνενώσουμε πολλούς πίνακες πρέπει να χρησιμοποιήσουμε υποχρεωτικά το τμήμα WHERE της εντολής SELECT. Οι πίνακες που θα απαιτηθούν για την ανάκτηση των δεδομένων αναφέρονται στο τμήμα FROM. Για τη συνένωση των πινάκων μπορούν να χρησιμοποιηθούν πολλοί τελεστές, με συνηθέστερο αυτόν της ισότητας. Η συνένωση γίνεται στο τμήμα WHERE.

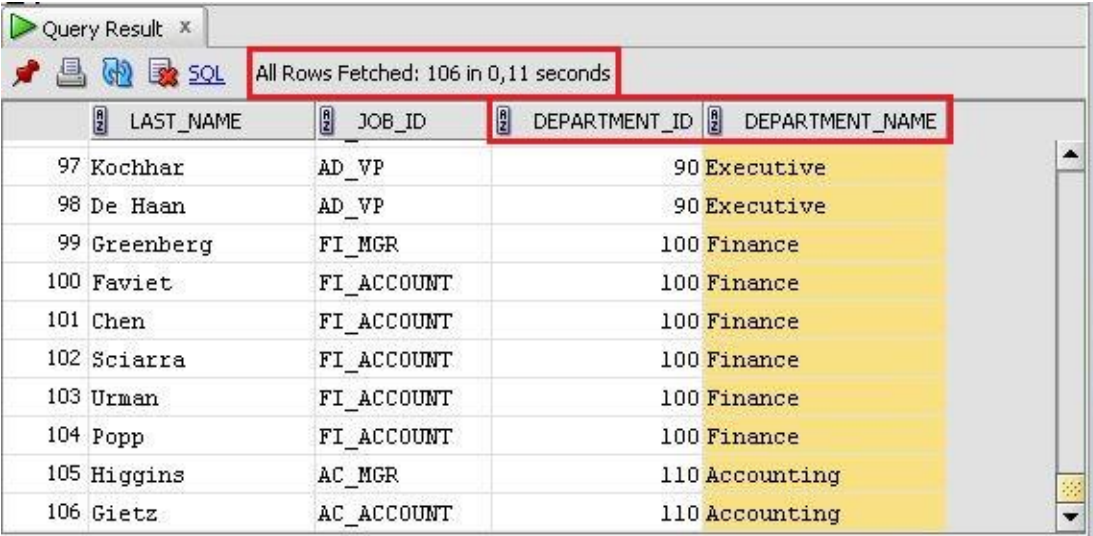


### Συνενώσεις ισότητας (ισοσυνδέσεις - Equijoins-Inner Joins)

Η περισσότερο χρησιμοποιούμενη και ίσως η σημαντικότερη από τις συνενώσεις είναι η ισοσύνδεση ή Equijoin ή Inner Join. Στη συνθήκη μιας Equijoin χρησιμοποιούμε τον τελεστή ισότητας (=). Στην περίπτωση αυτή, επιστρέφονται μόνο οι εγγραφές που έχουν ισοδύναμες τιμές στις στήλες που συμμετέχουν στην ισότητα. Για την καλύτερη κατανόηση αυτού του τύπου συνένωσης θα εξετάσουμε δύο απλά παραδείγματα:

Στο πρώτο παράδειγμα, το ερώτημα θα μας επιστρέψει το επίθετο, την εργασία, τον κωδικό τμήματος και την περιγραφή του τμήματος για όλους τους υπαλλήλους της εταιρείας.

```
SELECT last_name, job_id, departments.department_id,  
       department_name  
FROM employees, departments  
WHERE employees.department_id = departments.department_id;
```



	LAST_NAME	JOB_ID	DEPARTMENT_ID	DEPARTMENT_NAME
97	Kochhar	AD_VP	90	Executive
98	De Haan	AD_VP	90	Executive
99	Greenberg	FI_MGR	100	Finance
100	Faviet	FI_ACCOUNT	100	Finance
101	Chen	FI_ACCOUNT	100	Finance
102	Sciarra	FI_ACCOUNT	100	Finance
103	Urman	FI_ACCOUNT	100	Finance
104	Popp	FI_ACCOUNT	100	Finance
105	Higgins	AC_MGR	110	Accounting
106	Gietz	AC_ACCOUNT	110	Accounting

Εικόνα 3- 70 Ισοσύνδεση των πινάκων employees και departments

Στη συγκεκριμένη περίπτωση απαιτείται η συνένωση (join) των πινάκων employees και departments, διότι η περιγραφή του τμήματος που εργάζεται ο υπάλληλος βρίσκεται αποθηκευμένη στον πίνακα departments. Η Oracle συνδέει τους δύο πίνακες σύμφωνα με την ισότητα:

employees.department\_id = departments.department\_id

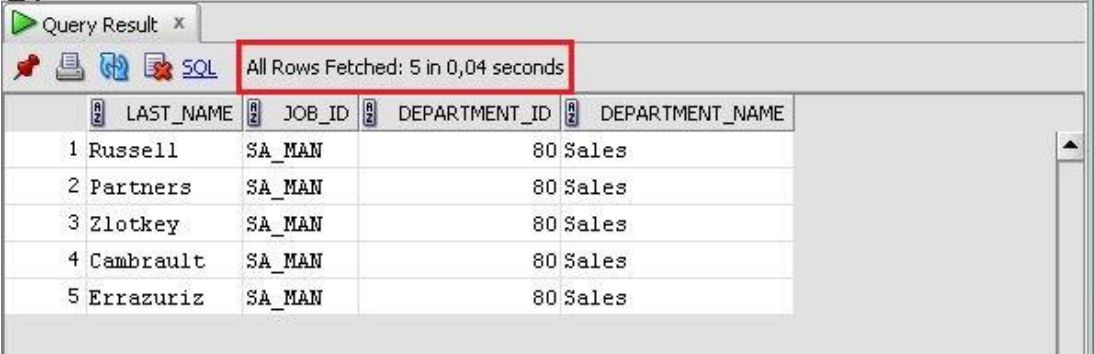
Όπως παρατηρούμε το πεδίο με το οποίο πραγματοποιείται η σύνδεση έχει κοινό όνομα και στους δύο πίνακες και γι' αυτό προηγείται το όνομα του πίνακα στον οποίο ανήκει.

Έχουμε τη δυνατότητα προκειμένου να γλυτώσουμε πληκτρολογήσεις και για να κάνουμε την εντολή μας πιο ευανάγνωστη, να χρησιμοποιήσουμε ψευδώνυμα (alias) για τους πίνακες. Η μετονομασία είναι προσωρινή και ισχύει μόνο για τη συγκεκριμένη εντολή. Στις αυτό-συνδέσεις (Self Joins) οι απόδοση ψευδωνύμων στους πίνακες είναι υποχρεωτική. Θα μπορούσαμε να ξαναγράψουμε το προηγούμενο παράδειγμα ως εξής:

```
SELECT last_name, job_id, d.department_id,
       department_name
FROM employees e, departments d
WHERE e.department_id = d.department_id;
```

Σε μια εντολή SELECT, εκτός από τις συνθήκες που υλοποιούν τη συνένωση των πινάκων, μπορούμε να έχουμε και όποιον άλλο περιορισμό χρειαζόμαστε. Έστω ότι στο προηγούμενο παράδειγμα ενδιαφερόμαστε μόνο για τους υπαλλήλους που εργάζονται ως 'SA\_MAN'. Η παρακάτω εντολή ενσωματώνει τη συγκεκριμένη απαίτηση. Η εντολή είναι ταυτόσημη με την προηγούμενη, απλά έχει προστεθεί στο τέλος της, η συνθήκη **AND job\_id = 'SA\_MAN'**.

```
SELECT last_name, job_id, d.department_id,
       department_name
FROM employees e, departments d
WHERE e.department_id = d.department_id
AND job_id = 'SA_MAN';
```



Query Result x

All Rows Fetched: 5 in 0,04 seconds

	LAST_NAME	JOB_ID	DEPARTMENT_ID	DEPARTMENT_NAME
1	Russell	SA_MAN	80	Sales
2	Partners	SA_MAN	80	Sales
3	Zlotkey	SA_MAN	80	Sales
4	Cambrault	SA_MAN	80	Sales
5	Errazuriz	SA_MAN	80	Sales

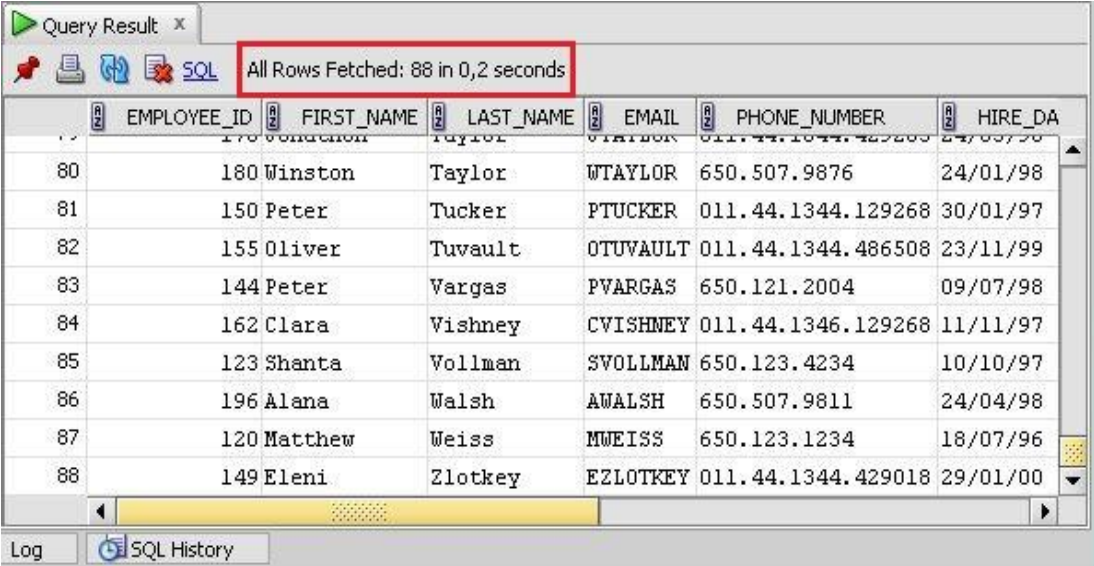
**Εικόνα 3- 71** Επιπλέον συνθήκη περιορισμού των δεδομένων, εκτός της συνθήκης υλοποίησης της σύνδεσης

### Συνενώσεις ανισότητας και Antijoins.

Η συνένωση ανισότητας (non equijoin), συνδέει δύο ή περισσότερους πίνακες με βάση την ανισότητα. Όταν χρησιμοποιούμε **non equijoin** μπορεί να πάρουμε αρκετές γραμμές που δεν μας χρησιμεύουν σε τίποτα. Θα πρέπει να ελέγχουμε πολύ προσεκτικά τα παραγόμενα αποτελέσματα.

Μια antijoin επιστρέφει τις γραμμές που δεν μπορούν να ταιριάξουν με το υποερώτημα που βρίσκεται στα δεξιά μέρος της σύνδεσης. Για να το κατανοήσουμε καλύτερα ας εξετάσουμε τη παρακάτω εντολή.

```
SELECT * FROM employees
WHERE department_id NOT IN
(SELECT department_id FROM departments
WHERE location_id = 1700)
ORDER BY last_name;
```



EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE
80	Winston	Taylor	WTAYLOR	650.507.9876	24/01/98
81	Peter	Tucker	PTUCKER	011.44.1344.129268	30/01/97
82	Oliver	Tuvault	OTUVAULT	011.44.1344.486508	23/11/99
83	Peter	Vargas	PVARGAS	650.121.2004	09/07/98
84	Clara	Vishney	CVISHNEY	011.44.1346.129268	11/11/97
85	Shanta	Vollman	SVOLLMAN	650.123.4234	10/10/97
86	Alana	Walsh	AWALSH	650.507.9811	24/04/98
87	Matthew	Weiss	MWEISS	650.123.1234	18/07/96
88	Eleni	Zlotkey	EZLOTKEY	011.44.1344.429018	29/01/00

Εικόνα 3- 72 Παράδειγμα non equijoin

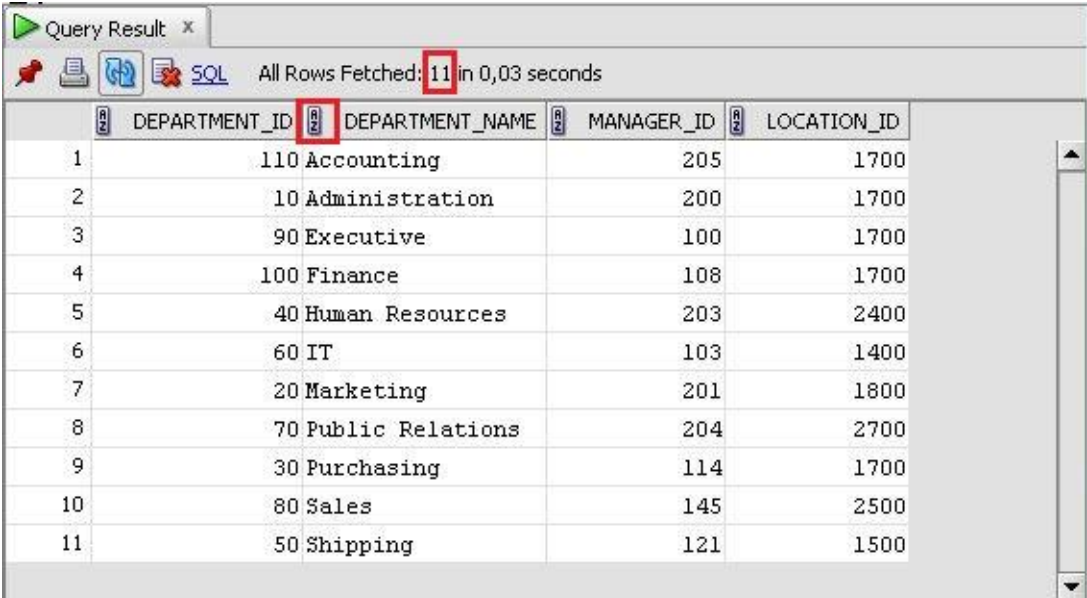
Το ερώτημα αυτό θα εμφανίσει μια λίστα των υπαλλήλων που **δεν ανήκουν** σ' ένα συγκεκριμένο σύνολο τμημάτων (αυτά που έχουν location\_id ίσο με 1700).

### Ημί-ενώσεις (Semijoins)

Για την υλοποίηση μιας Ημί-ένωσης (Semijoin) χρησιμοποιείται ο τελεστής EXISTS. Ένα Semijoin επιστρέφει τις εγγραφές που ταιριάξουν με τ' αποτελέσματα του υποερωτήματος, που βρίσκεται στο δεξί μέρος του τελεστή. Σε περίπτωση που έχουμε διπλότυπες εγγραφές, επιστρέφονται μία φορά.

Στο παρακάτω παράδειγμα επιστρέφεται μία μόνο εγγραφή, από τον πίνακα *departments*, ακόμη και αν υπάρχουν πολλές γραμμές του πίνακα *employees*, που ταιριάζουν με το υποερώτημα που έπεται του τελεστή EXISTS. Το συγκεκριμένο ερώτημα θα επιστρέψει τα τμήματα (μία φορά το καθένα) στα οποία υπάρχουν υπάλληλοι που αμείβονται με περισσότερα από 2500, ταξινομημένα σε αύξουσα σειρά του ονόματος του τμήματος.

```
SELECT * FROM departments d
WHERE EXISTS
  (SELECT * FROM employees e
   WHERE d.department_id = e.department_id AND e.salary > 2500)
ORDER BY department_name;
```



Query Result x

All Rows Fetched: 11 in 0,03 seconds

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	110	Accounting	205	1700
2	10	Administration	200	1700
3	90	Executive	100	1700
4	100	Finance	108	1700
5	40	Human Resources	203	2400
6	60	IT	103	1400
7	20	Marketing	201	1800
8	70	Public Relations	204	2700
9	30	Purchasing	114	1700
10	80	Sales	145	2500
11	50	Shipping	121	1500

Εικόνα 3- 73 Παράδειγμα Semijoin

## Εξωτερικές συνενώσεις (Outer Joins)

Η εξωτερική σύνδεση χρησιμοποιείται όταν θέλουμε το ερώτημα να επιστρέψει όλες τις γραμμές ενός πίνακα, ακόμα και όταν δεν υπάρχουν οι αντίστοιχες γραμμές στον πίνακα συνένωσης. Μια outer join μπορεί να είναι αριστερή (LEFT OUTER JOIN), δεξιά (RIGHT OUTER JOIN), ή πλήρης (FULL OUTER JOIN). Η γενική σύνταξη έχει ως εξής:

```
SELECT ...  
FROM table1 {RIGHT | LEFT | FULL} [OUTER] JOIN ON table2
```

Στην Oracle, παρόλο που δεν συνίσταται, μπορούμε να χρησιμοποιήσουμε και τον Oracle τελεστή εξωτερικής συνένωσης (+). Αν έχουμε δύο πίνακες που θέλουμε να συνενώσουμε, έστω, τον **A** και τον **B**, έχουμε τις παρακάτω επιλογές:

- Για να συντάξουμε ένα ερώτημα που υλοποιεί μια εξωτερική σύνδεση, η οποία επιστρέφει όλες τις εγγραφές από τον πίνακα **A** (left outer join) έχουμε δύο επιλογές:

α) Χρησιμοποιούμε τη *LEFT [OUTER] JOIN* σύνταξη, στο *FROM* τμήμα της εντολής.

β) Εφαρμόζουμε τον τελεστή συνένωσης (+) σε όλες τις στήλες του πίνακα **B**, στη συνθήκη συνένωσης στο *WHERE* τμήμα της εντολής.

Για όλες τις γραμμές του πίνακα **A** που δεν υπάρχουν αντίστοιχες του **B**, η Oracle θα επιστρέψει *NULL*, σε όσες εκφράσεις του *select list* εμπεριέχουν στήλες του πίνακα **B**.

- Για να συντάξουμε ένα ερώτημα που υλοποιεί μια εξωτερική σύνδεση, η οποία επιστρέφει όλες τις εγγραφές από τον πίνακα **B** (right outer join) έχουμε δύο επιλογές:

α) Χρησιμοποιούμε τη *RIGHT [OUTER] JOIN* σύνταξη, στο *FROM* τμήμα της εντολής.

β) Εφαρμόζουμε τον τελεστή συνένωσης (+) σε όλες τις στήλες του πίνακα **A** στη συνθήκη συνένωσης στο *WHERE* τμήμα της εντολής.

Για όλες τις γραμμές του πίνακα **B** που δεν υπάρχουν αντίστοιχες του **A**, η Oracle θα επιστρέψει *NULL*, σε όσες εκφράσεις του *select list* εμπεριέχουν στήλες του πίνακα **A**.

- Για να συντάξουμε ένα ερώτημα που υλοποιεί μια εξωτερική σύνδεση, που επιστρέφει όλες τις εγγραφές και από τους δύο πίνακες (full outer join), χρησιμοποιούμε τη *FULL [OUTER] JOIN* σύνταξη στο *FROM* τμήμα της εντολής.

Η παρακάτω εντολή επιστρέφει όλα τα τμήματα (κωδικό τμήματος και επώνυμο υπαλλήλου), ακόμα και αν σ' αυτά δεν έχει ενταχθεί κανένας υπάλληλος. Σε μια τέτοια περίπτωση, στη θέση του επωνύμου εμφανίζεται το null.

```
SELECT d.department_id, e.last_name
FROM departments d LEFT OUTER JOIN employees e ON
      d.department_id = e.department_id;
```

Query Result x

All Rows Fetched: 122 in 0,11 seconds

	DEPARTMENT_ID	LAST_NAME
101	20	Hartstein
102	20	Fay
103	40	Mavris
104	70	Baer
105	110	Higgins
106	110	Gietz
107	220	(null)
108	170	(null)
109	240	(null)
110	210	(null)
111	160	(null)

Ο πίνακας employees έχει μόλις 107 γραμμές ενώ το ερώτημα επιστρέφει 122

Log SQL History

Εικόνα 3- 74 Αριστερή Εξωτερική Συνένωση (Left Outer Join)

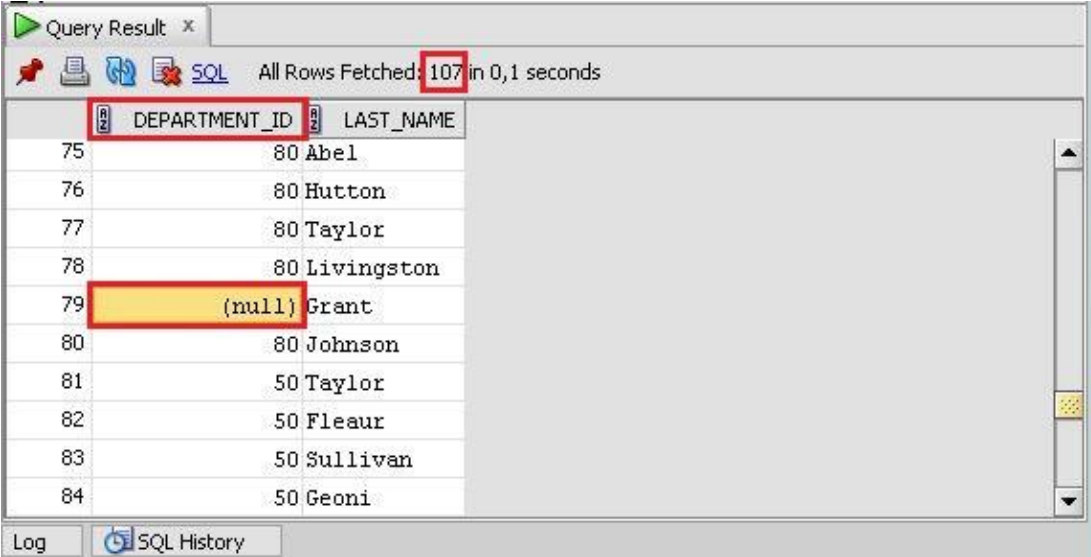
Η ακόλουθη εντολή είναι ταυτόσημη με την προηγούμενη.

```
SELECT d.department_id, e.last_name
FROM departments d, employees e
WHERE d.department_id = e.department_id(+);
```

Η ίδια εντολή με right outer join επιστρέφει όλους τους υπαλλήλους (κωδικό τμήματος και επώνυμο υπαλλήλου) ακόμα και αν κάποιοι δεν έχουν ενταχθεί ακόμα σε τμήμα. Σε μια τέτοια περίπτωση, στον κωδικό του τμήματος εμφανίζεται το null.



```
SELECT d.department_id, e.last_name
FROM departments d RIGHT OUTER JOIN employees e
ON d.department_id = e.department_id;
```



Query Result x

All Rows Fetched: 107 in 0,1 seconds

	DEPARTMENT_ID	LAST_NAME
75	80	Abel
76	80	Hutton
77	80	Taylor
78	80	Livingston
79	(null)	Grant
80	80	Johnson
81	50	Taylor
82	50	Fleaur
83	50	Sullivan
84	50	Geoni

Log SQL History

Εικόνα 3- 75 Δεξιά Εξωτερική Συνένωση (Right Outer Join)

Η τελευταία εντολή υλοποιεί μια full outer join.

```
SELECT department_id AS d_e_dept_id, e.last_name
FROM departments d FULL OUTER JOIN employees e
USING (department_id);
```

### Αυτό-ενώσεις (Self Joins)

Μια Αυτό-ένωση είναι η συνένωση ενός πίνακα με τον εαυτό του. Ο συγκεκριμένος πίνακας εμφανίζεται δύο φορές στο FROM clause. Πρέπει να δώσουμε ψευδώνυμα (aliases) σε κάθε εμφάνιση του πίνακα, ώστε να μπορούμε να αναφερθούμε στα πεδία που επιθυμούμε, στο υπόλοιπο τμήμα της εντολής SELECT. Αν ένας πίνακας εμφανίζεται στο FROM clause N φορές θα πρέπει να ορίσουμε τουλάχιστον N-1 ψευδώνυμα.

Στο παράδειγμα έχουμε ένα ερώτημα που επιστρέφει το επώνυμο κάθε υπαλλήλου ακολουθούμενο από το επώνυμο του προϊσταμένου του, για τους υπαλλήλους που το επίθετο τους ξεκινάει με το γράμμα 'R'.

```
SELECT e.last_name as ΥΠ,
       e.last_name||' εργάζεται για τον '||m.last_name
       "Υπάλληλοι και Προϊστάμενοι", m.last_name ΠΡΟΙΣΤ
FROM employees e, employees m
WHERE e.manager_id = m.employee_id
AND e.last_name LIKE 'R%'
ORDER BY m.last_name
```

	ΥΠ	Υπάλληλοι και Προϊστάμενοι	ΠΡΟΙΣΤ
1	Rogers	Rogers εργάζεται για τον Kaufling	Kaufling
2	Russell	Russell εργάζεται για τον King	King
3	Raphaely	Raphaely εργάζεται για τον King	King
4	Rajs	Rajs εργάζεται για τον Mourgos	Mourgos

Εικόνα 3- 76 Παράδειγμα Self Join

### Καρτεσιανά γινόμενα (Cartesian Products)

Εάν επιλέξουμε στήλες (πεδία) από δύο ή περισσότερους πίνακες και δεν τους συνενώσουμε, θα πάρουμε ως έξοδο το Καρτεσιανό τους γινόμενο. Η Oracle θα συνδυάσει κάθε γραμμή, κάθε πίνακα, με όλες τις γραμμές των υπολοίπων πινάκων. Το Καρτεσιανό γινόμενο παράγει πολλές εγγραφές και σπανίως είναι χρήσιμο. Έστω ότι έχουμε δύο πίνακες με 200 γραμμές ο καθένας, το καρτεσιανό γινόμενο θα επιστρέψει  $200 \times 200 = 40.000$  γραμμές. Το παρακάτω παράδειγμα παράγει το καρτεσιανό γινόμενο των πινάκων *departments* και *employees*, το οποίο δεν έχει καμία πρακτική χρήση.

```
SELECT last_name, job_id, d.department_id, department_name
FROM employees e, departments d;
```

	LAST_NAME	JOB_ID	DEPARTMENT_ID	DEPARTMENT_NAME
2881	OConnell	SH_CLERK	270	Payroll
2882	Grant	SH_CLERK	270	Payroll
2883	Whalen	AD_ASST	270	Payroll
2884	Hartstein	MK_MAN	270	Payroll
2885	Fay	MK_REP	270	Payroll
2886	Mavris	HR_REP	270	Payroll
2887	Baer	PR_REP	270	Payroll
2888	Higgins	AC_MGR	270	Payroll
2889	Gietz	AC_ACCOUNT	270	Payroll

Εικόνα 3- 77 Καρτεσιανό γινόμενο των πινάκων employees και departments



## Σύνταξη και χρήση υποερωτημάτων (subqueries) στην ORACLE

Όπως έχουμε δει μέχρι τώρα, ένα ερώτημα είναι μια λειτουργία που ανακτά δεδομένα από έναν ή περισσότερους πίνακες ή/και όψεις. Ένα ερώτημα που είναι ένθετο μέσα σ' ένα άλλο, θα το ονομάζουμε *δευτερεύον ή υποερώτημα (subquery)*.

Ένα δευτερεύον ερώτημα μπορεί να χρησιμοποιηθεί για να απλοποιήσει άλλα ερωτήματα και μπορούμε να το συναντήσουμε είτε στο FROM, είτε στο WHERE τμήμα, μιας εντολής SELECT. Όταν βρίσκεται στο FROM τμήμα της εντολής ονομάζεται και *inline όψη (inline view)*, ενώ όταν βρίσκεται στο WHERE τμήμα ονομάζεται *ένθετο δευτερεύον ερώτημα (ή φωλιασμένο - nested subquery)*.

Ένα δευτερεύον ερώτημα μπορεί να περιέχει ένα άλλο δευτερεύον ερώτημα. Ο Oracle Database server δεν επιβάλλει κάποιο όριο στον αριθμό των επιπέδων στα *inline views*, ενώ στα *nested subqueries* επιτρέπει μέχρι 255 επίπεδα.

Ένα δευτερεύον ερώτημα μπορεί να χρησιμοποιηθεί σε πολλές εντολές όπως: *SELECT, UPDATE, INSERT, DELETE, CREATE TABLE, CREATE VIEW*.

Στη συνέχεια θα δούμε μερικά παραδείγματα, ώστε αυτά να γίνουν πιο κατανοητά.

### Το υποερώτημα σαν πίνακας (inline view)

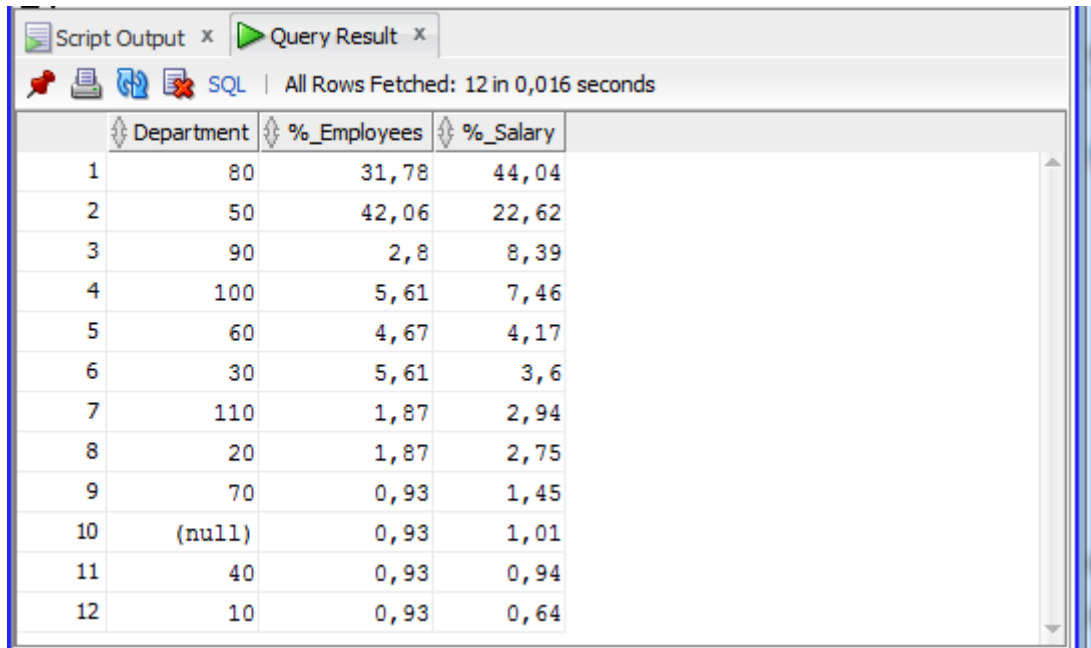
Στο παράδειγμα αυτό έχουμε δύο inline views, το πρώτο έχει επισημανθεί με μπλε χρώμα και το δεύτερο με κόκκινο. Και στα δύο έχουμε αποδώσει ψευδώνυμο (a, b αντίστοιχα) και βρίσκονται εντός παρενθέσεων. Για να αναφερθούμε σε κάποια στήλη τους στο κύριο ερώτημα, αρκεί πριν από το όνομα της στήλης να γράψουμε το κατάλληλο ψευδώνυμο. Το πρώτο (a) υποερώτημα, βρίσκει πόσοι υπάλληλοι υπηρετούν καθώς και το άθροισμα των μισθών τους, ανά τμήμα. Το δεύτερο (b) υποερώτημα, βρίσκει για το σύνολο της εταιρείας, το πλήθος και το άθροισμα των μισθών. Το κύριο SELECT εμφανίζει για κάθε τμήμα (του υποερωτήματος a), τον κωδικό, το ποσοστό του προσωπικού ( $a.num\_emp/b.total\_count$ ) που απασχολείται στο τμήμα, καθώς και το ποσοστό του κόστους μισθοδοσίας που του αναλογεί ( $a.sal\_sum/b.total\_sal$ ). Τ' αποτελέσματα εμφανίζονται ταξινομημένα σε φθίνουσα σειρά του κόστους μισθοδοσίας κάθε τμήματος.

Σ' ένα δευτερεύον ερώτημα, για να κάνουμε τις εντολές πιο ευανάγνωστες, πρέπει να χαρακτηρίζουμε τις στήλες με το όνομα ή το ψευδώνυμο του πίνακα ή της όψης.

```

SELECT a.department_id "Department",
       ROUND (a.num_emp/b.total_count*100 ,2) "%_Employees",
       ROUND(a.sal_sum/b.total_sal * 100,2) "%_Salary"
FROM
(SELECT department_id, COUNT(*) num_emp, SUM(salary) sal_sum
 FROM employees
 GROUP BY department_id) a,
(SELECT COUNT(*) total_count, SUM(salary) total_sal
 FROM employees) b
ORDER BY a.sal_sum DESC;

```



	Department	%_Employees	%_Salary
1	80	31,78	44,04
2	50	42,06	22,62
3	90	2,8	8,39
4	100	5,61	7,46
5	60	4,67	4,17
6	30	5,61	3,6
7	110	1,87	2,94
8	20	1,87	2,75
9	70	0,93	1,45
10	(null)	0,93	1,01
11	40	0,93	0,94
12	10	0,93	0,64

Εικόνα 3- 78 Υποερωτήματα σαν πίνακες (υποερώτημα *a* και *b*)

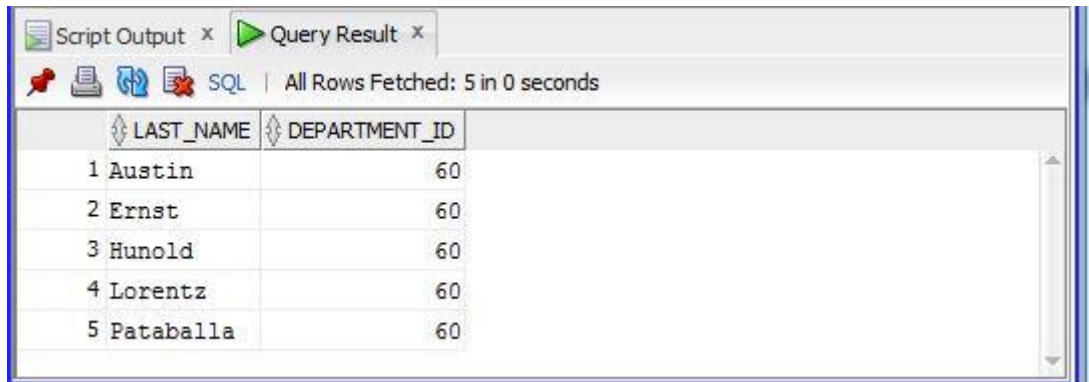
### Ένθετα υποερωτήματα (nested subqueries)

Για την καλύτερη κατανόηση των ένθετων υποερωτημάτων θα δούμε τρία παραδείγματα.

Στο πρώτο παράδειγμα, ένα ένθετο υποερώτημα χρησιμοποιείται προκειμένου να “εντοπίσει” το τμήμα που εργάζεται κάποιος με επίθετο 'Lorentz'. Στη συνέχεια εμφανίζει το επώνυμο και το τμήμα όλων των υπαλλήλων που εργάζονται στο εν λόγω τμήμα και ταξινομεί το αποτέλεσμα με βάση το επώνυμο σε αύξουσα σειρά.

Στις περιπτώσεις που δεν χρησιμοποιούμε τελεστή **Συνόλων**, αλλά σειράς (γραμμής δηλ. =, >, <, <>), πρέπει να προσέχουμε το subquery να επιστρέφει **μία μόνο γραμμή**. Όλοι οι τελεστές σύγκρισης που ελέγχουν μεμονωμένες τιμές, μπορούν να δουλεύουν με υποερωτήματα, υπό την προϋπόθεση ότι, το υποερώτημα με το οποίο δουλεύουν επιστρέφει **μία μόνο γραμμή**.

```
SELECT last_name, department_id
FROM employees
WHERE department_id =
      (SELECT DISTINCT department_id FROM employees
       WHERE last_name = 'Lorentz')
ORDER BY last_name;
```



The screenshot shows a SQL query result window with two tabs: 'Script Output' and 'Query Result'. The 'Query Result' tab is active, displaying a table with two columns: 'LAST\_NAME' and 'DEPARTMENT\_ID'. The table contains five rows of data, all with 'DEPARTMENT\_ID' equal to 60. The status bar indicates 'All Rows Fetched: 5 in 0 seconds'.

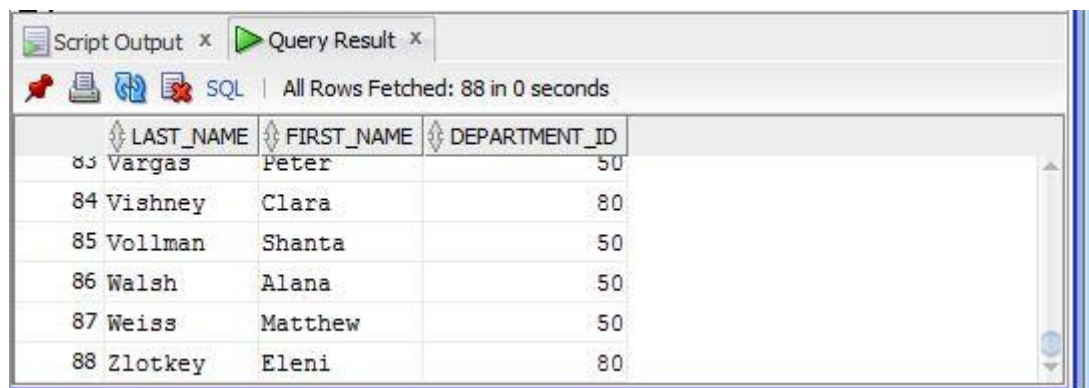
	LAST_NAME	DEPARTMENT_ID
1	Austin	60
2	Ernst	60
3	Hunold	60
4	Lorentz	60
5	Pataballa	60

Εικόνα 3- 79 Ένθετο υποερώτημα, παράδειγμα 1

Όπως μπορούμε να χρησιμοποιήσουμε τελεστές με μεμονωμένες τιμές σ' ένα υποερώτημα, έτσι μπορούμε να χρησιμοποιήσουμε και τελεστές σύγκρισης «πολλαπλών τιμών» (IN, NOT IN, EXISTS)

Το ένθετο ερώτημα του δεύτερου παραδείγματος, θα “δημιουργήσει” μια λίστα με τα τμήματα που υπάρχουν στην περιοχή με κωδικό 1700. Στη συνέχεια θα επιστρέψει τα ζητούμενα δεδομένα (ονοματεπώνυμο και κωδικό του τμήματος), για τους εργαζόμενους που εργάζονται σε τμήματα, που δεν εμπεριέχονται στη λίστα αυτή, ταξινομημένα σε αύξουσα σειρά του επωνύμου.

```
SELECT last_name, first_name, department_id FROM employees
WHERE department_id NOT IN (SELECT department_id FROM
departments WHERE location_id = 1700) ORDER BY last_name;
```



The screenshot shows a SQL query result window with two tabs: 'Script Output' and 'Query Result'. The 'Query Result' tab is active, displaying a table with three columns: 'LAST\_NAME', 'FIRST\_NAME', and 'DEPARTMENT\_ID'. The table contains six rows of data. The status bar indicates 'All Rows Fetched: 88 in 0 seconds'.

	LAST_NAME	FIRST_NAME	DEPARTMENT_ID
83	Vargas	Peter	50
84	Vishney	Clara	80
85	Vollman	Shanta	50
86	Walsh	Alana	50
87	Weiss	Matthew	50
88	Zlotkey	Eleni	80

Εικόνα 3- 80 Ένθετο υποερώτημα, παράδειγμα 2

Στο τρίτο παράδειγμα θα εξετάσουμε ένα *Συσχετισμένο Δευτερεύον Ερώτημα* (*συσχετισμένο υποερώτημα ή correlated subquery*). Ένα *συσχετισμένο δευτερεύον ερώτημα*, είναι ένα υποερώτημα που εξαρτάται από πληροφορίες στο κύριο ερώτημα. Γενικά οι πίνακες σ' ένα υποερώτημα μπορούν να συσχετίζονται με πίνακες του κύριου ερωτήματος. Το κύριο ερώτημα μπορεί να είναι μια πρόταση SELECT, UPDATE ή DELETE.

Η παρακάτω εντολή επιστρέφει δεδομένα για τους εργαζόμενους, των οποίων ο μισθός υπερβαίνει τον μέσο μισθό του τμήματός τους. Δίνουμε ένα ψευδώνυμο (*emp*) στο πίνακα *employees* (περιέχει την πληροφορία για τον μισθό) του κύριου ερωτήματος και το χρησιμοποιούμε στο συσχετισμένο υποερώτημα (*WHERE emp.department\_id = department\_id*). Το ερώτημα *για κάθε γραμμή* του πίνακα *employees*, πραγματοποιεί τα παρακάτω βήματα:

1. Εντοπίζει το *department\_id* που αφορά τη γραμμή.
2. Υπολογίζει τον μέσο μισθό του τμήματος.
3. Ελέγχει εάν ο μισθός του εργαζομένου είναι μεγαλύτερος από τον μ.ο. και στην περίπτωση που είναι, επιστρέφει τη συγκεκριμένη γραμμή στο αποτέλεσμα.

Το υποερώτημα αποτιμάται μία φορά *για κάθε γραμμή* του κύριου πίνακα (στο παράδειγμά μας, του *employees*).

```
SELECT department_id, last_name, salary
FROM employees emp
WHERE salary > (SELECT AVG(salary)
                 FROM employees
                 WHERE emp.department_id = department_id)
ORDER BY department_id;
```

	DEPARTMENT_ID	LAST_NAME	SALARY
33	80	Fox	9600
34	80	Abel	11000
35	90	King	24000
36	100	Greenberg	12000
37	100	Faviet	9000
38	110	Higgins	12000

Εικόνα 3- 81 Συσχετισμένο Δευτερεύον Ερώτημα (correlated subquery)

## Πράξεις συνόλων

Η γλώσσα SQL της Oracle μας επιτρέπει να κάνουμε *πράξεις συνόλων* μεταξύ ερωτημάτων, δηλαδή μεταξύ των αποτελεσμάτων δυο ή περισσότερων ερωτημάτων SELECT. Οι πράξεις αυτές είναι:

- Η ένωση (UNION ή UNION ALL)
- Η διαφορά (MINUS)
- Η τομή (INTERSECT)

Οι πράξεις αυτές έχουν το ανάλογο αποτέλεσμα των αντίστοιχων μαθηματικών πράξεων αλλά αντί μεταξύ συνόλων αριθμών εφαρμόζονται μεταξύ συνόλων γραμμών (εγγραφών). Απαραίτητη προϋπόθεση, σε όλες τις περιπτώσεις, είναι τα επιμέρους ερωτήματα να έχουν την ίδια μορφή, δηλαδή το ίδιο πλήθος και τύπο στηλών, διαφορετικά οι πράξεις συνόλου δεν μπορούν να πραγματοποιηθούν. Στη συνέχεια θα εξετάσουμε αναλυτικότερα κάθε μια από τις παραπάνω πράξεις.

Προκειμένου να υλοποιηθούν οι πράξεις αυτές χρησιμοποιούμε τους αντίστοιχους τελεστές συνόλων.

*Σημείωση: Οι εντολές SQL των παραδειγμάτων που περιγράφονται παρακάτω σαφώς και μπορούν να γραφούν με πιο απλό τρόπο. Η χρήση σύνθετων ερωτημάτων γίνεται καθαρά για τους εκπαιδευτικούς σκοπούς της ενότητας.*

### Η Πράξη της Ένωσης (UNION ή UNION ALL)

Ο τελεστής ένωσης **UNION ALL** ενοποιεί τα αποτελέσματα δυο ή περισσότερων ερωτημάτων σε ένα. Αν τα επιμέρους ερωτήματα επιστρέφουν αντίστοιχα N1, N2, ...Nn αριθμό εγγραφών, το τελικό αποτέλεσμα θα περιέχει N1+N2+...+Nn εγγραφές.

Έστω πως το αποτέλεσμα ενός ερωτήματος E1 είναι οι εγγραφές:

AD_PRES	President	20080	40000
AD_VP	Administration Vice President	15000	30000
AD_ASST	Administration Assistant	3000	6000
FI_MGR	Finance Manager	8200	16000
AD_VP	Administration Vice President	15000	30000

και αντίστοιχα ενός άλλου ερωτήματος E2 είναι οι εγγραφές:

AD_ASST	Administration Assistant	3000	6000
FI_MGR	Finance Manager	8200	16000
ST_MAN	Stock Manager	5500	8500
ST_CLERK	Stock Clerk	2008	5000
SH_CLERK	Shipping Clerk	2500	5500

τότε η ένωση (UNION ALL) των δυο ερωτημάτων θα επιστρέψει ως αποτέλεσμα το εξής σύνολο εγγραφών:

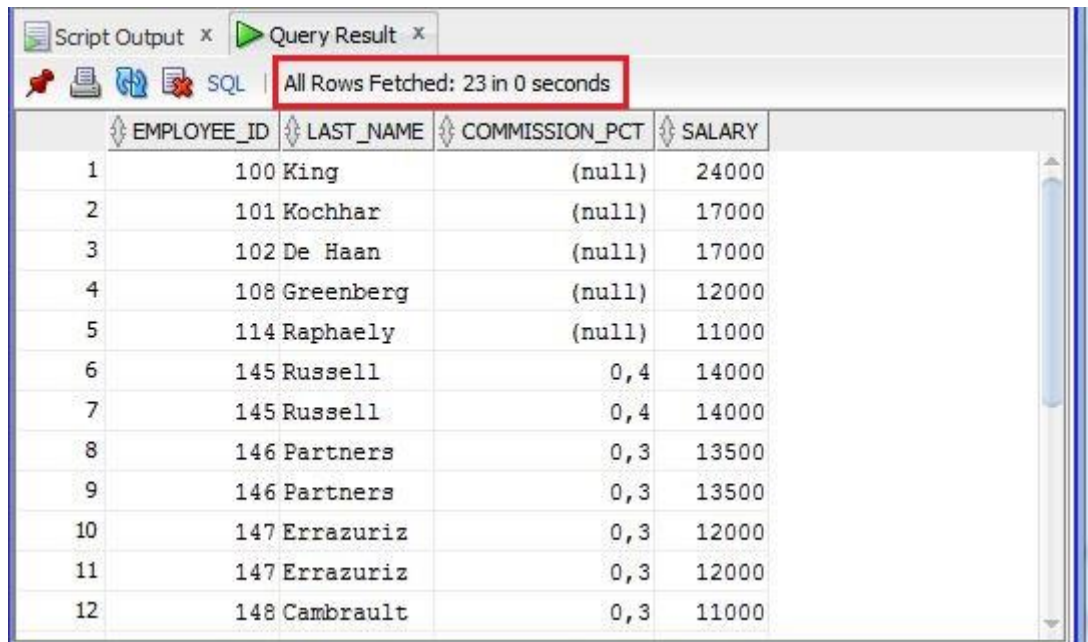
AD_PRES	President	20080	40000
AD_VP	Administration Vice President	15000	30000
AD_ASST	Administration Assistant	3000	6000
FI_MGR	Finance Manager	8200	16000
AD_VP	Administration Vice President	15000	30000
AD_ASST	Administration Assistant	3000	6000
FI_MGR	Finance Manager	8200	16000
ST_MAN	Stock Manager	5500	8500
ST_CLERK	Stock Clerk	2008	5000
SH_CLERK	Shipping Clerk	2500	5500

Για παράδειγμα, έστω πως θέλουμε να εντοπίσουμε με ένα ερώτημα ένωσης τους υπαλλήλους που παίρνουν ποσοστά τουλάχιστον 30% ή ο μέγιστος μισθός που μπορούν να φθάσουν είναι πάνω από 13.000 ευρώ. Επιπλέον θέλουμε να ταξινομήσουμε το τελικό αποτέλεσμα (ένωση) ως προς τον κωδικό υπαλλήλου.

Το διαζευκτικό "ή" στην παραπάνω περιγραφή είναι αυτό που υπονοεί την ένωση δυο συνόλων εγγραφών, αυτών που πληρούν την πρώτη ή τη δεύτερη συνθήκη αντίστοιχα, δηλαδή την ένωση δυο ερωτημάτων. Δεν έχουμε λοιπόν παρά να συντάξουμε ανεξάρτητα τα δυο αυτά ερωτήματα και ακολούθως να προσθέσουμε τον τελεστή ένωσης UNION ALL. Η ακόλουθη εντολή SQL υλοποιεί την ένωση αυτή:

```
SELECT  employee_id, last_name, commission_pct, salary
FROM    employees
WHERE   commission_pct >= 0.3
UNION ALL
SELECT  employee_id, last_name, commission_pct, salary
FROM    employees, jobs
WHERE   employees.job_id = jobs.job_id
AND     jobs.max_salary >= 13000
ORDER  BY 1;
```





The screenshot shows a SQL query result window with the title 'Query Result'. A red box highlights the status bar text 'All Rows Fetched: 23 in 0 seconds'. The table below shows the results of a UNION ALL query, with columns EMPLOYEE\_ID, LAST\_NAME, COMMISSION\_PCT, and SALARY. The first 11 rows are from the first query, and the next 12 rows are from the second query, totaling 23 rows.

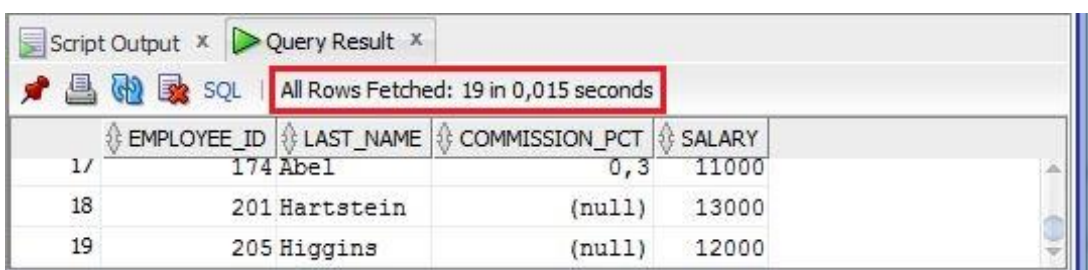
	EMPLOYEE_ID	LAST_NAME	COMMISSION_PCT	SALARY
1	100	King	(null)	24000
2	101	Kochhar	(null)	17000
3	102	De Haan	(null)	17000
4	108	Greenberg	(null)	12000
5	114	Raphaely	(null)	11000
6	145	Russell	0,4	14000
7	145	Russell	0,4	14000
8	146	Partners	0,3	13500
9	146	Partners	0,3	13500
10	147	Errazuriz	0,3	12000
11	147	Errazuriz	0,3	12000
12	148	Cambrault	0,3	11000

Εικόνα 3- 82 Η πράξη της Ένωσης Συνόλων με τον τελεστή UNION ALL

Η πρώτη SELECT επιστρέφει 11 εγγραφές ενώ η δεύτερη 12. Η ένωση τους έχει 23 εγγραφές.

Όπως παρατηρούμε, τα δυο υποερωτήματα έχουν την ίδια δομή (πλήθος και είδος στηλών). Η πρόταση ORDER BY 1 στο τέλος ταξινομεί το τελικό αποτέλεσμα ως προς την πρώτη στήλη, όπως εμφανίζονται στο select list, δηλαδή τον κωδικό υπαλλήλου (EMPLOYEE\_ID).

Η παραλλαγή της ένωσης χωρίς τη λέξη ALL, δηλαδή ο τελεστής **UNION**, απαλείφει από το τελικό αποτέλεσμα τις διπλότυπες εμφανίσεις της ίδιας εγγραφής. Δηλαδή, αν μια εγγραφή εμφανίζεται παραπάνω από μια φορές (στο ίδιο ή σε διαφορετικά υποερωτήματα), τότε στο τελικό αποτέλεσμα UNION αυτή θα εμφανιστεί *μια μόνο* φορά. Εάν στο προηγούμενο παράδειγμα αντικαταστήσουμε τον τελεστή UNION ALL με τον UNION στο τελικό σύνολο θα πάρουμε 19 γραμμές.



The screenshot shows a SQL query result window with the title 'Query Result'. A red box highlights the status bar text 'All Rows Fetched: 19 in 0,015 seconds'. The table below shows the results of a UNION query, with columns EMPLOYEE\_ID, LAST\_NAME, COMMISSION\_PCT, and SALARY. The first row is from the first query, and the next three rows are from the second query, totaling 19 rows.

	EMPLOYEE_ID	LAST_NAME	COMMISSION_PCT	SALARY
1/	174	Abel	0,3	11000
18	201	Hartstein	(null)	13000
19	205	Higgins	(null)	12000

Εικόνα 3- 83 Η πράξη της Ένωσης Συνόλων με το τελεστή UNION

*Σημείωση: Ο έλεγχος που εκτελεί η UNION για τις διπλότυπες τιμές, αγνοεί τις στήλες με τιμές NULL.*

### Η Πράξη της Τομής (INTERSECT)

Ο τελεστής τομής **INTERSECT** υλοποιεί την τομή των αποτελεσμάτων δύο ή περισσότερων ερωτημάτων. Επιστρέφει δηλαδή τις μοναδικές εγγραφές που εμφανίζονται σε όλα τα επιμέρους ερωτήματα, δηλαδή μόνο τις κοινές εγγραφές των υποερωτημάτων.

Έστω πως το αποτέλεσμα ενός ερωτήματος E1 είναι οι εγγραφές:

AD PRES	President	20080	40000
AD VP	Administration Vice President	15000	30000
AD ASST	Administration Assistant	3000	6000
FI MGR	Finance Manager	8200	16000
AD VP	Administration Vice President	15000	30000

και αντίστοιχα ενός άλλου ερωτήματος E2 είναι οι εγγραφές:

AD ASST	Administration Assistant	3000	6000
FI MGR	Finance Manager	8200	16000
ST MAN	Stock Manager	5500	8500
ST CLERK	Stock Clerk	2008	5000
SH CLERK	Shipping Clerk	2500	5500

τότε η τομή (INTERSECT) των δυο ερωτημάτων θα επιστρέψει ως αποτέλεσμα μόνο τις κοινές εγγραφές:

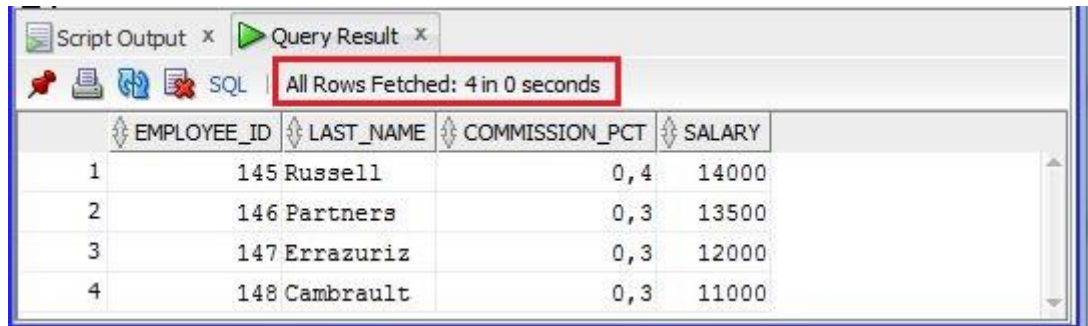
AD ASST	Administration Assistant	3000	6000
FI MGR	Finance Manager	8200	16000

Για παράδειγμα, έστω πως θέλουμε να εντοπίσουμε με ένα ερώτημα τομής τους υπαλλήλους που παίρνουν ποσοστά τουλάχιστον 30% **και** επιπλέον ο μέγιστος μισθός που μπορούν να φθάσουν είναι πάνω από 13.000 ευρώ.

Το συζευκτικό "και" στην παραπάνω περιγραφή είναι αυτό που υπονοεί την τομή δυο συνόλων εγγραφών, αυτών που πληρούν την πρώτη ή τη δεύτερη συνθήκη αντίστοιχα, δηλαδή την τομή δυο ερωτημάτων. Η ακόλουθη εντολή SQL υλοποιεί την τομή αυτή:



```
SELECT employee_id, last_name, commission_pct, salary
FROM employees
WHERE commission_pct >= 0.3
INTERSECT
SELECT employee_id, last_name, commission_pct, salary
FROM employees, jobs
WHERE employees.job_id = jobs.job_id
AND jobs.max_salary >= 13000;
```



EMPLOYEE_ID	LAST_NAME	COMMISSION_PCT	SALARY
1	145 Russell	0,4	14000
2	146 Partners	0,3	13500
3	147 Errazuriz	0,3	12000
4	148 Cambrault	0,3	11000

Εικόνα 3- 84 Η πράξη της Τομής Συνόλων με τον τελεστή INTERSECT

Όπως και ο τελεστής UNION, έτσι και ο τελεστής INTERSECT απαλείφει από το τελικό αποτέλεσμα τις διπλότυπες εμφανίσεις της ίδιας εγγραφής.

### Η Πράξη της Διαφοράς (MINUS)

Ο τελεστής διαφοράς **MINUS** «αφαιρεί» από τα αποτελέσματα ενός ερωτήματος τα αποτελέσματα ενός άλλου ερωτήματος, δηλαδή επιστρέφει τις μοναδικές εγγραφές που εμφανίζονται στο πρώτο ερώτημα αλλά όχι στο δεύτερο.

Έστω πως το αποτέλεσμα ενός ερωτήματος E1 είναι οι εγγραφές:

AD PRES	President	20080	40000
AD VP	Administration Vice President	15000	30000
AD ASST	Administration Assistant	3000	6000
FI MGR	Finance Manager	8200	16000
AD VP	Administration Vice President	15000	30000

και αντίστοιχα ενός άλλου ερωτήματος E2 είναι οι εγγραφές:

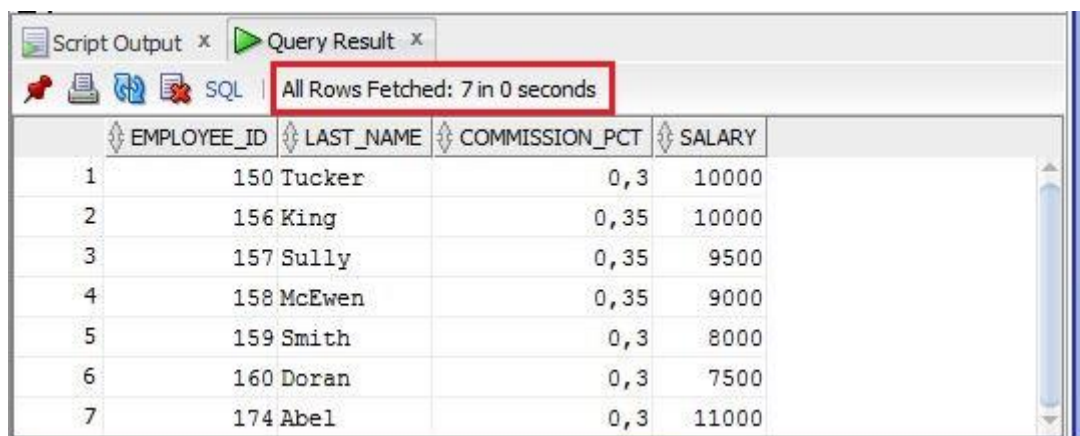
AD ASST	Administration Assistant	3000	6000
FI MGR	Finance Manager	8200	16000
ST MAN	Stock Manager	5500	8500
ST CLERK	Stock Clerk	2008	5000
SH CLERK	Shipping Clerk	2500	5500

τότε η διαφορά (MINUS) των δυο ερωτημάτων θα επιστρέψει ως αποτέλεσμα τις εγγραφές:

AD_PRES	President	20080	40000
AD_VP	Administration Vice President	15000	30000

Για παράδειγμα, έστω πως θέλουμε να εντοπίσουμε με ένα ερώτημα διαφοράς τους υπαλλήλους που παίρνουν ποσοστά τουλάχιστον 30% πλην αυτών που ο μέγιστος μισθός που μπορούν να φθάσουν είναι πάνω από 13.000 ευρώ. Η ακόλουθη εντολή SQL υλοποιεί την τομή αυτή:

```
SELECT employee_id, last_name, commission_pct, salary
FROM employees
WHERE commission_pct >= 0.3
MINUS
SELECT employee_id, last_name, commission_pct, salary
FROM employees, jobs
WHERE employees.job_id = jobs.job_id
AND jobs.max_salary >= 13000;
```



	EMPLOYEE_ID	LAST_NAME	COMMISSION_PCT	SALARY
1	150	Tucker	0,3	10000
2	156	King	0,35	10000
3	157	Sully	0,35	9500
4	158	McEwen	0,35	9000
5	159	Smith	0,3	8000
6	160	Doran	0,3	7500
7	174	Abel	0,3	11000

Εικόνα 3- 85 Η πράξη της Αφαίρεσης Συνόλων με τον τελεστή MINUS

Όπως και οι τελεστές UNION και INTERSECT, ο τελεστής MINUS απαλείφει από το τελικό αποτέλεσμα τις διπλότυπες εμφανίσεις της ίδιας εγγραφής.

## 4. Η ORACLE SQL/DML – INSERT, UPDATE, DELETE

### Σκοπός και στόχοι της ενότητας

Σκοπός της ενότητας είναι οι συμμετέχοντες να κατανοήσουν τη σημασία και τον τρόπο σύνταξης των εντολών τροποποίησης δεδομένων της SQL καθώς και να εξοικειωθούν με τη διαδικασία δημιουργίας εντολών με τη χρήση της ίδιας της SQL.

Ειδικότερα, οι επιμορφούμενοι θα είναι σε θέση να:

- Εισάγουν νέες εγγραφές με την εντολή INSERT
- Τροποποιούν εγγραφές με την εντολή UPDATE
- Διαγράφουν εγγραφές με τις εντολές DELETE και TRUNCATE TABLE
- Συντάσσουν εντολές SQL που δημιουργούν εντολές SQL

### Εισαγωγή

Όπως είδαμε στην προηγούμενη ενότητα, οι βασικές εντολές (ή «πράξεις») της DML είναι 4:

- Αναζήτηση εγγραφών (SELECT)
- Εισαγωγή εγγραφών (INSERT)
- Ενημέρωση εγγραφών (UPDATE)
- Διαγραφή εγγραφών (DELETE)

Εύκολα συνειδητοποιούμε πως με το σετ των 4 αυτών εντολών μπορούμε να εκτελέσουμε σε μια ΒΔ οποιαδήποτε λειτουργία χειρισμού δεδομένων χρειαστούμε.

Γνωρίσαμε ήδη την εντολή αναζήτησης SELECT με την οποία συντάσσουμε ερωτήματα *αναζήτησης* δεδομένων. Στην ενότητα αυτή θα γνωρίσουμε τις εντολές *τροποποίησης*, δηλαδή εισαγωγής, ενημέρωσης και διαγραφής δεδομένων, οι οποίες είναι αντίστοιχα οι εντολές SQL/DML INSERT, UPDATE και DELETE.

## Εισαγωγή εγγραφών – η εντολή INSERT

Η εντολή SQL/DML **INSERT** εισάγει νέες εγγραφές σε έναν πίνακα. Η γενική σύνταξη της εντολής INSERT είναι:

```
INSERT INTO <TABLE> (COL1, COL2, ...) VALUES (VAL1, VAL2, ...);
```

όπου <TABLE> είναι ο πίνακας στον οποίο θέλουμε να εισάγουμε την εγγραφή, COL1, COL2,... τα ονόματα των στηλών (πεδίων) που θέλουμε να εισάγουμε ενώ VAL1, VAL2, ... είναι αντίστοιχα οι τιμές των πεδίων αυτών. Μεταξύ των πεδίων και των τιμών πρέπει να υπάρχει αντιστοιχία ένα προς ένα ενώ παρατηρούμε πως δεν είναι απαραίτητο να εισάγουμε όλα τα πεδία, αρκεί βέβαια όσα πεδία παραλείψουμε να επιτρέπεται να έχουν κενή τιμή (να μην έχουν δηλαδή τον περιορισμό NOT NULL).

Για παράδειγμα, έστω ότι θέλουμε να εισάγουμε στον πίνακα DEPARTMENTS δυο νέα τμήματα με τα παρακάτω στοιχεία:

ID	DESCRIPTION	MNGR_ID	LOC_ID
280	Environmental Issues	200	1700
290	Customer Care	200	1700

Για την εισαγωγή της πρώτης εγγραφής, σύμφωνα με την σύνταξη που περιγράψαμε, δίνουμε την εντολή:

```
INSERT INTO departments (department_id, department_name,
                           manager_id, location_id)
VALUES (280, 'ENVIRONMENTAL ISSUES', 200, 1700);
```

Προς απλοποίηση, η ORACLE μας δίνει τη δυνατότητα να παραλείψουμε τα ονόματα των πεδίων στην περίπτωση που α) εισάγουμε όλα τα πεδία και β) η σειρά τους είναι ακριβώς όπως στην δομή του πίνακα. Έτσι, για την δεύτερη εγγραφή μπορούμε να απλοποιήσουμε την εντολή INSERT ως εξής:

```
INSERT INTO departments
VALUES (290, 'CUSTOMER CARE', 200, 1700);
```

Ερώτηση: έχουν οριστικά εισαχθεί οι νέες αυτές εγγραφές στον πίνακα DEPARTMENTS; Την απάντηση θα δούμε στην επόμενη παράγραφο που αφορά δυο πολύ χρήσιμες εντολές της ORACLE, τις COMMIT και ROLLBACK.

## Δυο σημαντικότερες εντολές: COMMIT, ROLLBACK

Η ORACLE μας παρέχει δυο «μαγικές» εντολές που μπορούμε να χρησιμοποιήσουμε μετά από οποιαδήποτε εντολή ενημέρωσης της SQL/DML (εισαγωγή, ενημέρωση, διαγραφή):

- **COMMIT**: Οριστικοποίηση των αλλαγών στους πίνακες της ΒΔ.
- **ROLLBACK**: Αναίρεση/ακύρωση των αλλαγών από την αρχή της σύνδεσης ή από την προηγούμενη COMMIT ή ROLLBACK.

Ειδικά η εντολή ROLLBACK μπορεί να μας φανεί ιδιαίτερα χρήσιμη όταν οι αλλαγές που κάναμε περιέχουν σφάλμα και πρέπει να αναιρεθούν. Δεν μπορεί όμως να χρησιμοποιηθεί για να αναιρέσει μια SQL/DDDL εντολή, όπως την τροποποίηση της δομής ενός πίνακα ή άλλου αντικειμένου. Η εντολές COMMIT και ROLLBACK αφορούν αποκλειστικά αλλαγές που έγιναν με κάποια εντολή της SQL/DML, δηλαδή INSERT, UPDATE ή DELETE.

**Προσοχή**: ακόμη και αν δεν εκτελέσουμε ρητά την εντολή COMMIT, αυτή θα εκτελεστεί αυτόματα (*implicitly*) στις παρακάτω περιπτώσεις:

- Έξοδος από τη σύνδεση (SQLDeveloper, SQL\*Plus). Κάποια εργαλεία ρωτούν τον χρήστη αν θέλει να οριστικοποιήσει ή ακυρώσει τις αλλαγές, άλλα όμως τις οριστικοποιούν χωρίς προειδοποίηση.
- Εκτέλεση οποιασδήποτε εντολής SQL/DDDL (όπως δημιουργία/διαγραφή πίνακα ή άλλου αντικειμένου): Η ORACLE εκτελεί αυτόματα την εντολή COMMIT πριν και μετά από κάθε εντολή SQL/DDDL!

## Μαζική εισαγωγή εγγραφών - η εντολή INSERT INTO ... SELECT

Μια πολύ χρήσιμη παραλλαγή της εντολής INSERT INTO είναι η εντολή **INSERT INTO ... SELECT** η οποία εισάγει μαζικά εγγραφές από ένα ερώτημα (SELECT QUERY) σε έναν πίνακα. Η σύνταξη της είναι παρόμοια με της εντολής INSERT INTO, τώρα όμως δεν έχουμε μια λίστα εισαγόμενων τιμών (πρόταση VALUES(...)) αλλά το αποτέλεσμα ενός ερωτήματος, δηλαδή πολλές εγγραφές:

```
INSERT INTO <TABLE> (COL1, COL2, ...)
SELECT COL_A, COL_B, ...
FROM ...
WHERE ...
```

Εύκολα συνειδητοποιούμε πως μεταξύ των πεδίων που εισάγουμε (COL1, COL2, ...) και των στηλών του ερωτήματος SELECT (COL\_A, COL\_B, ...) πρέπει να υπάρχει αντιστοιχία ένα προς ένα.

Για παράδειγμα, έστω πως θέλουμε, χρησιμοποιώντας την εντολή INSERT INTO ... SELECT, να εισάγουμε στον υποθετικό πίνακα BONUS με στήλες (EMPLOYEE\_ID, POSO\_BONUS) ένα bonus ίσο με 650€ μόνο για τους Διευθυντές Πωλήσεων (JOB\_ID='SA\_MAN'). Η εντολή SQL που χρειαζόμαστε στην περίπτωση αυτή είναι:

```
INSERT INTO bonus
SELECT e.employee_id, 650
FROM employees e
WHERE e.job_id='SA_MAN';
```

## Ενημέρωση εγγραφών - η εντολή UPDATE

Η εντολή **UPDATE** χρησιμοποιείται για την ενημέρωση εγγραφών σε ένα πίνακα σύμφωνα με συγκεκριμένα κριτήρια. Μπορεί να τροποποιηθεί μια ή πολλές εγγραφές ενώ μπορούν να ενημερωθούν όλα ή μερικά από τα πεδία κάθε εγγραφής. Η γενική σύνταξη της εντολής UPDATE είναι:

```
UPDATE <TABLE>
SET <COL1>= <τιμή1>, <COL2>= <τιμή2>, ...
WHERE <συνθήκη>;
```

όπου <TABLE> είναι ο πίνακας εγγραφές του οποίου θέλουμε να τροποποιήσουμε, <COL1 >, <COL2>... είναι τα ονόματα των στηλών που θέλουμε να ενημερώσουμε με τις τιμές <τιμή1>, <τιμή2>... αντίστοιχα, ενώ <συνθήκη> μια έγκυρη συνθήκη (ή συνθήκες) WHERE η οποία καθορίζει ποιες εγγραφές του πίνακα θα ενημερωθούν.

Για παράδειγμα, έστω πως θέλουμε να δώσουμε αύξηση 5% σε όλους τους υπαλλήλους με μισθό μικρότερο των 3.000 ευρώ ενώ ταυτόχρονα να δώσουμε και ποσοστά προμήθειας 5%. Είναι προφανές πως θέλουμε να τροποποιήσουμε τις στήλες SALARY και COMMISSION\_PCT σε όσες εγγραφές του πίνακα EMPLOYEES πληρούν μια συγκεκριμένη συνθήκη. Έτσι, σύμφωνα με την σύνταξη που περιγράψαμε παραπάνω, η εντολή UPDATE που πρέπει να γράψουμε είναι:

```
UPDATE    employees
SET       salary = salary * 1.05, commission_pct = 0.05
WHERE     salary < 3000;
```

## Διαγραφή εγγραφών - η εντολή DELETE

Η εντολή **DELETE** χρησιμοποιείται για την διαγραφή εγγραφών από ένα πίνακα σύμφωνα με συγκεκριμένα κριτήρια. Η γενική της σύνταξη είναι:

```
DELETE    FROM <TABLE>
WHERE     <συνθήκη>;
```

όπου <TABLE> είναι ο πίνακας από τον οποίο θέλουμε να διαγράψουμε εγγραφές και <συνθήκη> μια έγκυρη συνθήκη (ή συνθήκες) WHERE η οποία καθορίζει ποιες εγγραφές του πίνακα θα διαγραφούν.

**Προσοχή:** Αν δεν προσδιορίσουμε κάποια συνθήκη WHERE τότε θα διαγραφούν όλες οι εγγραφές του πίνακα!

Για την καλύτερη κατανόηση της χρήσης της εντολής DELETE, ας υποθέσουμε πως θέλουμε να διαγράψουμε τα δυο νέα τμήματα που εισάγαμε στον πίνακα DEPARTMENTS σε προηγούμενο παράδειγμα, δηλαδή αυτά με κωδικούς 280 και 290. Η εντολή DELETE που χρειαζόμαστε είναι:

```
DELETE    FROM departments
```

```
WHERE    department_id IN (280,290);
```

Ας δοκιμάσουμε τώρα να διαγράψουμε το τμήμα με κωδικό 100. Σύμφωνα με τα παραπάνω δίνουμε την ακόλουθη εντολή DELETE:

```
DELETE    FROM departments
WHERE     department_id = 100;
```

Όπως ίσως υποψιαζόμαστε, η ORACLE αντί να διαγράψει την εγγραφή, απαντά με το μήνυμα

```
ORA-02292: integrity constraint (HR.EMP_DEPT_FK) violated -
child record found
```

Τι έχει συμβεί; Η ORACLE δεν επιτρέπει τη διαγραφή μιας εγγραφής από ένα πίνακα αν υπάρχουν σε άλλους πίνακες ξένα κλειδιά για αυτή την εγγραφή. Πράγματι, δίνοντας την εντολή

```
SELECT * FROM employees;
```

βλέπουμε πως υπάρχουν εγγραφές υπαλλήλων με DEPARTMENT\_ID = 100 και εφόσον το πεδίο EMPLOYEES.DEPARTMENT\_ID είναι ξένο κλειδί που αναφέρεται στο πεδίο DEPARTMENTS.DEPARTMENT\_ID, η ORACLE δεν επιτρέπει τη διαγραφή του συγκεκριμένου τμήματος.

Τέλος, ας διαγράψουμε όλες τις εγγραφές του πίνακα BONUS. Οι παρακάτω εντολές είναι ισοδύναμες, επομένως αρκεί να εκτελέσουμε μια από τις δυο:

```
DELETE    FROM bonus;
DELETE    bonus;
```

## Γρήγορη διαγραφή εγγράφων - η εντολή TRUNCATE TABLE

Συμβαίνει συχνά να θέλουμε να διαγράψουμε όλες τις εγγραφές ενός πίνακα (παράδειγμα ιστορικό που δεν χρειαζόμαστε πλέον). Είδαμε πως αυτό γίνεται με την εντολή DELETE χωρίς να προσδιορίσουμε κάποια συνθήκη WHERE. Το μειονέκτημα της μεθόδου αυτής είναι πως εάν οι εγγραφές προς διαγραφή είναι πάρα πολλές (ίσως και εκατομμύρια), η διαγραφή ενδέχεται να καθυστερήσει σημαντικά, πιθανά δε και να μην μπορέσει να εκτελεστεί αν δεν υπάρχει ικανός αποθηκευτικός



χώρος<sup>1</sup>. Στην περίπτωση αυτή μπορούμε να χρησιμοποιήσουμε την εντολή **TRUNCATE TABLE**, αρκεί να θυμόμαστε 2 σημαντικά σημεία:

- Δεν έχουμε τη δυνατότητα αναίρεσης (ROLLBACK).
- Δεν μπορούμε να προσδιορίσουμε κριτήρια, επομένως θα διαγραφούν υποχρεωτικά όλες οι εγγραφές του πίνακα.

Για παράδειγμα, για να αδειάσουμε γρήγορα τον πίνακα BONUS δίνουμε την εντολή:

```
TRUNCATE TABLE bonus;
```

### Αυτόματη δημιουργία εντολών SQL με χρήση εντολών SQL

Πολλές φορές προκύπτει η ανάγκη να εκτελέσουμε επαναληπτικά μια εντολή SQL στην οποία κάθε φορά αλλάζουν κάποιες τιμές ή κάποιες παράμετροι ενώ η βασική σύνταξη της εντολής παραμένει η ίδια. Στις περιπτώσεις αυτές μπορούμε να καταφύγουμε στην ίδια την SQL ώστε να δημιουργήσουμε αυτόματα τις εντολές αυτές ως αλφαριθμητικές τιμές.

Για παράδειγμα, για να αλλάξουμε το μισθό του υπαλλήλου με EMPLOYEE\_ID=115 στον ελάχιστο μισθό που μπορεί να πάρει (MIN\_SALARY=2500 ευρώ, από τον πίνακα JOBS με βάση το JOB\_ID του υπαλλήλου), δίνουμε την εντολή:

```
UPDATE employees
SET salary=2500 /*ελάχιστος μισθός από τον πίνακα JOBS*/
WHERE employee_id=115;
```

Χρησιμοποιώντας την γλώσσα SQL μπορούμε να συνθέσουμε την εντολή αυτή αυτοματοποιημένα για όλους τους υπαλλήλους! Η λογική βασίζεται στο "χτίσιμο" μιας εντολής SQL από την συνένωση (με τον τελεστή συνένωσης ||) σταθερών και μεταβλητών αλφαριθμητικών τιμών. Η παρακάτω εντολή SELECT δημιουργεί με τον τρόπο αυτό για κάθε υπάλληλο την αντίστοιχη εντολή UPDATE:

```
SELECT 'UPDATE employees SET salary='
      || jobs.min_salary
      || ' WHERE employee_id='
```

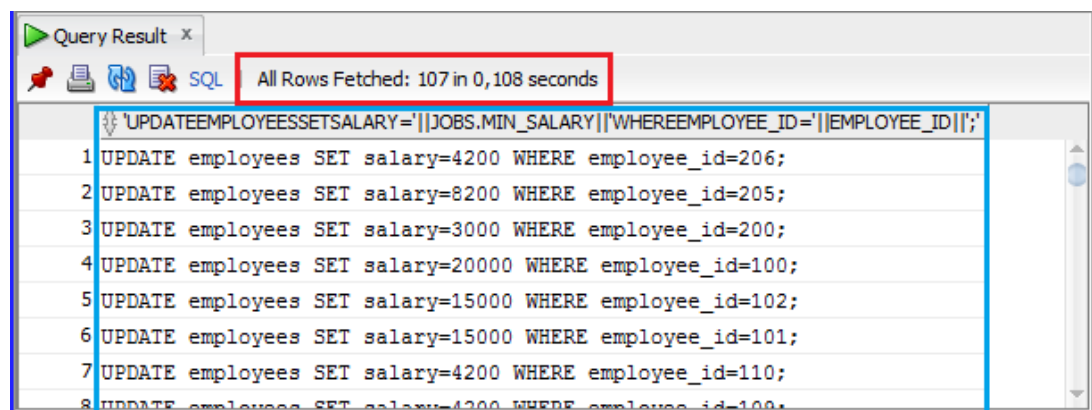
<sup>1</sup> Κατά την εκτέλεση εντολών SQL/DML, η ORACLE κρατά ένα προσωρινό ιστορικό (UNDO DATA) ώστε να είναι δυνατή η αναίρεση των αλλαγών (ROLLBACK).

```

        || employee_id || ' ';
FROM    employees, jobs
WHERE    employees.job_id = jobs.job_id;

```

Η παραπάνω εντολή SQL επιστρέφει ως αποτέλεσμα μια σειρά από έτοιμες συντακτικά εντολές:



Εικόνα 4-1 Αυτόματη δημιουργία εντολών SQL με χρήση εντολών SQL

Έτσι, το μόνο που έχουμε να κάνουμε είναι να αντιγράψουμε το αποτέλεσμα (με αντιγραφή και επικόλληση) ως εντολές σε ένα άλλο παράθυρο SQL και να το εκτελέσουμε.

## Η ψευδοστήλη ROWID

Σε κάθε πίνακα της ORACLE προϋπάρχει η ψευδοστήλη ROWID. Το πεδίο αυτό αποτελεί την μοναδική «διεύθυνση» της εγγραφής και δημιουργείται αυτόματα από την ORACLE. Δεν φαίνεται στην δομή του πίνακα, στην εντολή DESC, στις εντολές SELECT κλπ. και γι' αυτό συνήθως ο χρήστης αγνοεί την ύπαρξη της. Η στήλη όμως αυτή μπορεί να χρησιμοποιηθεί σε οποιαδήποτε συνθήκη WHERE ώστε να αναφερόμαστε ταχύτατα σε μια συγκεκριμένη εγγραφή διότι η ORACLE μπορεί από το ROWID να εντοπίσει σχεδόν ακαριαία μια εγγραφή.

Για την καλύτερη κατανόηση της χρήσης της ψευδοστήλης ROWID, ας δούμε ένα παράδειγμα ενημέρωσης εγγραφής με αυτό τον τρόπο αναφοράς. Έστω πως θέλουμε να προσθέσουμε το επίθεμα 'RR Tolkien' στο όνομα του υπαλλήλου που ονομάζεται 'John Russell'. Για να εντοπίσουμε την εγγραφή αυτή, δίνουμε αρχικά την ακόλουθη εντολή SELECT:

```

SELECT  ROWID, T.*
FROM    employees T
WHERE    last_name LIKE 'Russell%';

```

Έστω πως το αποτέλεσμα της παραπάνω εντολής είναι η γραμμή:

```
AAASZYAAFAAAAHdAA2, 145, John, Russell, ...
```

Από την εντολή SELECT βλέπουμε πως το ROWID είναι η πρώτη στήλη, δηλαδή η τιμή 'AAASZYAAFAAAAHdAA2'. Για να ενημερώσουμε τώρα την σχετική εγγραφή χρησιμοποιώντας για αναφορά την ψευδοστήλη ROWID, δίνουμε την ακόλουθη εντολή UPDATE:

```
UPDATE employees
SET first_name = first_name || ' RR TOLKIEN'
WHERE ROWID = 'AAASZYAAFAAAAHdAA2';
```

## 5. Η ORACLE SQL/DDL

### Σκοπός και στόχοι της ενότητας

Σκοπός της ενότητας είναι οι συμμετέχοντες να κατανοήσουν την σημασία και τον τρόπο σύνταξης των εντολών διαχείρισης αντικειμένων της SQL καθώς και της σημασίας των ιδιαίτερων χαρακτηριστικών καθενός από αυτά.

Ειδικότερα, οι επιμορφούμενοι θα είναι σε θέση να:

- Δημιουργούν πίνακες, όψεις, ευρετήρια, ακολουθίες.
- Αναγνωρίζουν τους συνήθεις τύπους δεδομένων της ORACLE.
- Δημιουργούν περιορισμούς πρωτεύοντος και ξένου κλειδιού καθώς και περιορισμούς ελέγχων.
- Τροποποιούν πίνακες.
- Διαγράφουν πίνακες, όψεις, ευρετήρια, ακολουθίες.

### Εισαγωγή

Η ORACLE SQL μας παρέχει ένα υποσύνολο εντολών για την διαχείριση των αντικειμένων μιας ΒΔ ORACLE. Όπως είδαμε υπάρχουν πολλές κατηγορίες αντικειμένων, όπως οι πίνακες, οι όψεις, οι ακολουθίες και τα ευρετήρια. Με τον όρο διαχείριση εννοούμε, μεταξύ άλλων την:

- Δημιουργία αντικειμένων
- Τροποποίηση αντικειμένων
- Μετονομασία αντικειμένων
- Διαγραφή αντικειμένων

Η κατηγορία αυτή εντολών της ORACLE SQL ονομάζεται *Γλώσσα Ορισμού Δεδομένων ή SQL/DDL (Data Definition Language)*. Στη συνέχεια θα μελετήσουμε ορισμένες από αυτές τις εντολές.

Φυσικά, η διαχείριση αντικειμένων στην ORACLE είναι δυνατό να γίνει και μέσα από το γραφικό περιβάλλον εργαλείων όπως ο SQLDeveloper, όμως το πλεονέκτημα της χρήσης της SQL/DDl είναι πως αυτοματοποιεί την διαχείριση αντικειμένων (όπως την μαζική δημιουργία αντικειμένων, για παράδειγμα δημιουργία 100 νέων χρηστών για τις ανάγκες ενός εκπαιδευτικού εργαστηρίου). Επιπλέον, δεν εξαρτάται από το εργαλείο που χρησιμοποιούμε αφού η ίδιες εντολές SQL/DDl μπορούν να εκτελεστούν σε οποιοδήποτε εργαλείο.

## Διαχείριση πινάκων στην ORACLE με τη χρήση της SQL/DDl

Η ORACLE SQL μας παρέχει ένα σύνολο εντολών για την διαχείριση των πινάκων μιας ΒΔ. Όπως είδαμε παραπάνω, λέγοντας διαχείριση πινάκων εννοούμε μεταξύ άλλων την:

- Δημιουργία πίνακα
- Τροποποίηση δομής πίνακα
- Μετονομασία πίνακα
- Αντιγραφή πίνακα
- Διαγραφή πίνακα

## Δημιουργία πινάκων στην ORACLE

Η δημιουργία πίνακα με τη χρήση SQL γίνεται με την εντολή CREATE TABLE. Η γενική σύνταξη της εντολής CREATE TABLE είναι η εξής:

```
CREATE TABLE <TABLE_NAME> (  
    <COLUMN_1> <TYPE>    <OPTIONS>,  
    <COLUMN_2> <TYPE>    <OPTIONS>,  
    ...  
    <COLUMN_N> <TYPE>    <OPTIONS>  
) ;
```

όπου <TABLE\_NAME> είναι το όνομα του πίνακα που θα δημιουργηθεί, <COLUMN1>, <COLUMN2>, ... είναι τα ονόματα των πεδίων (ή αλλιώς στηλών) του πίνακα, <TYPE> είναι ο τύπος δεδομένων του κάθε πεδίου, ενώ <OPTIONS> είναι προαιρετικοί περιορισμοί που μπορούμε να προσδιορίσουμε για το εκάστοτε

πεδίο. Η πρόταση <COLUMNxx> <TYPE> <OPTIONS> επαναλαμβάνεται για κάθε πεδίο του πίνακα και διαχωρίζεται από την προηγούμενη της με κόμμα (,).

Τα ονόματα των πεδίων ενός πίνακα πρέπει να είναι μοναδικά στον ίδιο πίνακα αλλά μπορούν να επαναλαμβάνονται σε άλλους πίνακες. Επιπλέον, πρέπει να είναι **έγκυρα αναγνωριστικά** της ORACLE.

Ένα *έγκυρο αναγνωριστικό ORACLE* πρέπει να πληροί τους παρακάτω κανόνες:

- Ξεκινά με ένα γράμμα.
- Μπορεί να περιλαμβάνει γράμματα, ψηφία και τα σύμβολα \$, #, \_ (underscore).
- Δεν είναι ίδιο με κάποια από τις δεσμευμένες λέξεις της ORACLE<sup>2</sup>

Για παράδειγμα, έγκυρα αναγνωριστικά είναι τα MISTHOS1, ONOMA\_ASTHENH και THLEFWNO#1 ενώ το 2NAME και το SELECT δεν είναι (το πρώτο ξεκινά με ψηφίο ενώ το δεύτερο είναι δεσμευμένη λέξη της ORACLE).

Για να κατανοήσουμε καλύτερα την εντολή CREATE TABLE, έστω πως θέλουμε να δημιουργήσουμε έναν νέο πίνακα με όνομα APOUSIES για τις απουσίες των υπαλλήλων με τις εξής στήλες (πεδία):

- EMPLOYEE\_ID (id υπαλλήλου)
- DATE\_FROM (από πότε)
- DATE\_TO (μέχρι πότε).

Επιπλέον, θέλουμε το πεδίο EMPLOYEE\_ID να είναι *ξένο κλειδί (foreign key)* και να συνδέεται με το αντίστοιχο πεδίο στον πίνακα EMPLOYEES (δηλαδή να δημιουργηθεί σχέση μεταξύ των πινάκων APOUSIES και EMPLOYEES με το πεδίο EMPLOYEE\_ID) ενώ κανένα πεδίο δεν επιτρέπεται να είναι κενό (NULL).

Η εντολή SQL/DDDL δημιουργεί τον παραπάνω πίνακα είναι η εξής:

<sup>2</sup> Για το πλήρες ευρετήριο των δεσμευμένων λέξεων της ORACLE μπορείτε να ανατρέξετε στο παράρτημα Oracle SQL Reserved Words and Keywords του εγχειριδίου SQL Language Reference της ORACLE

```
CREATE TABLE APOUSIES (  
    EMPLOYEE_ID NUMBER(6) NOT NULL  
    REFERENCES EMPLOYEES(EMPLOYEE_ID),  
    DATE_FROM DATE NOT NULL,  
    DATE_TO DATE NOT NULL);
```

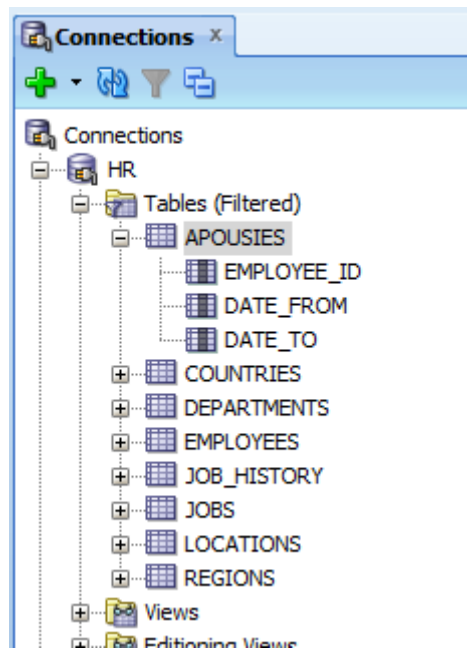
Το πεδίο EMPLOYEE\_ID ορίζεται ως τύπου NUMBER 6 ψηφίων, ίδιο με το αντίστοιχο πεδίο του πίνακα EMPLOYEES, ενώ η πρόταση REFERENCES EMPLOYEES(EMPLOYEE\_ID) δημιουργεί τη σχέση μεταξύ των πινάκων APOUSIES και EMPLOYEES με βάση το πεδίο EMPLOYEE\_ID του κάθε πίνακα. Η σχέση αυτή, όπως έχουμε δει, εξασφαλίζει την ακεραιότητα των δεδομένων των δυο πινάκων, δηλαδή κάθε εγγραφή του πίνακα APOUSIES θα πρέπει να έχει ένα EMPLOYEE\_ID που να υπάρχει στον πίνακα EMPLOYEES και αντίστοιχα δεν μπορεί να διαγραφεί μια εγγραφή από τον πίνακα EMPLOYEES αν υπάρχει σχετική εγγραφή στον πίνακα APOUSIES. Το πεδίο στο οποίο αναφέρεται ένα ξένο κλειδί (στην προκειμένη περίπτωση το πεδίο EMPLOYEES.EMPLOYEE\_ID) πρέπει να έχει τον περιορισμό UNIQUE (ή να είναι πρωτεύον κλειδί (PRIMARY KEY) οπότε είναι υποχρεωτικά και UNIQUE).

Τα πεδία DATE\_FROM και DATE\_TO ορίζονται ως τύπου DATE (ημερομηνίας) ενώ με τον περιορισμό NOT NULL απαγορεύεται να έχουν κενή τιμή.

Η απόκριση της ORACLE μετά την εκτέλεση της παραπάνω εντολής είναι

```
table APOUSIES created.
```

μας ενημερώνει δηλαδή πως ο πίνακας δημιουργήθηκε επιτυχώς. Ο νέος πίνακας APOUSIES φαίνεται τώρα κάτω από την ενότητα TABLES του SQLDeveloper.



Εικόνα 5-1 : Ενότητα TABLES του SQLDeveloper

Μπορούμε επίσης να δούμε την δομή του νέου πίνακα με την εντολή DESC (describe):

```
DESC APOUSIES;
```

Name	Null	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
DATE_FROM	NOT NULL	DATE
DATE_TO	NOT NULL	DATE

### Συνήθεις τύποι δεδομένων στην ORACLE

Η ORACLE παρέχει διάφορους τύπους δεδομένων κατά την δημιουργία ενός πίνακα.

Οι πιο συνήθεις είναι οι εξής:

- **NUMBER**(συνολικά ψηφία, δεκαδικά ψηφία), για παράδειγμα NUMBER(12,3), NUMBER(6).
- **VARCHAR2** (μέγεθος) : κείμενο, για παράδειγμα VARCHAR2(120) είναι αλφαριθμητικό 120 χαρακτήρων.
- **DATE** : ημερομηνία και ώρα, χωρίς άλλο προσδιορισμό.



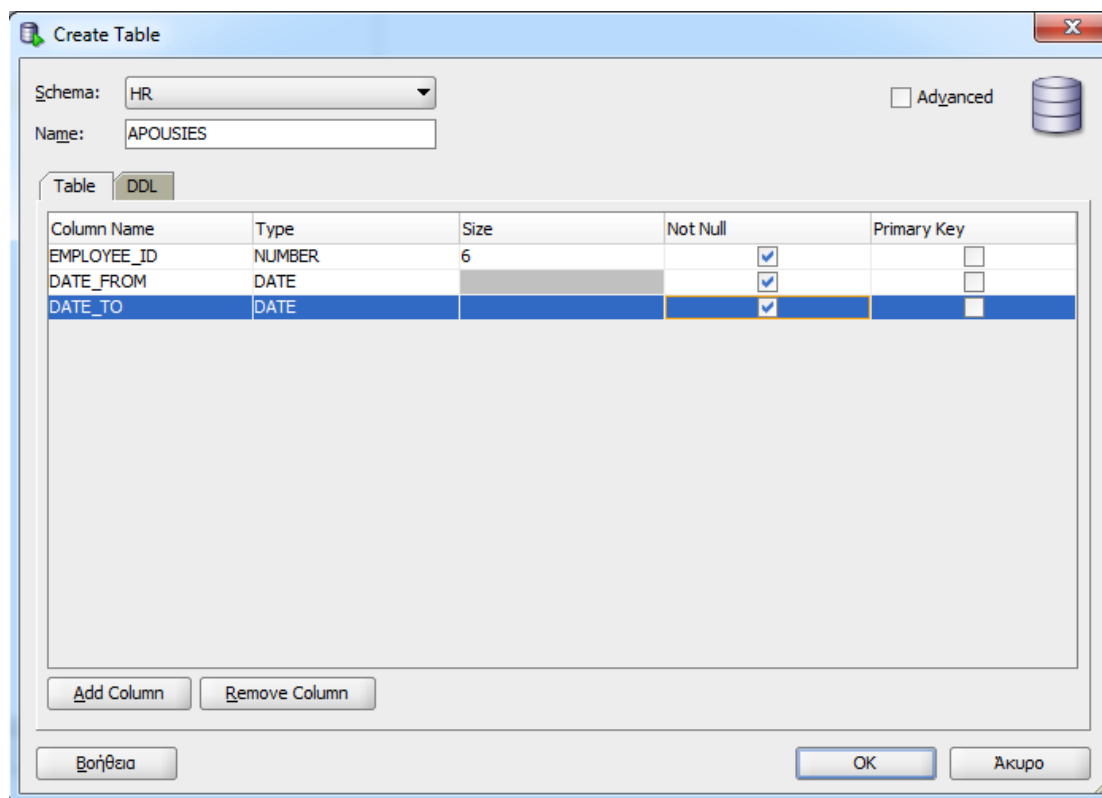
### Συνήθεις περιορισμοί

Κατά τη δημιουργία ενός πίνακα, οι πιο συνήθεις περιορισμοί στα πεδία του είναι οι εξής:

- **NOT NULL** : Απαγορεύονται οι κενές τιμές.
- **UNIQUE** : Απαγορεύονται οι διπλότυπες τιμές.
- **PRIMARY KEY** : Το πεδίο είναι πρωτεύον κλειδί του πίνακα.
- **REFERENCES** : Το πεδίο αναφέρεται σε ένα όμοιου τύπου πεδίο ενός άλλου πίνακα, δηλαδή δημιουργείται σχέση μεταξύ του νέου πίνακα και του πίνακα που προσδιορίζεται στην πρόταση REFERENCES. Με άλλα λόγια, το πεδίο θα αποτελεί ξένο κλειδί (foreign key).

### Δημιουργία πίνακα με τον SQLDeveloper

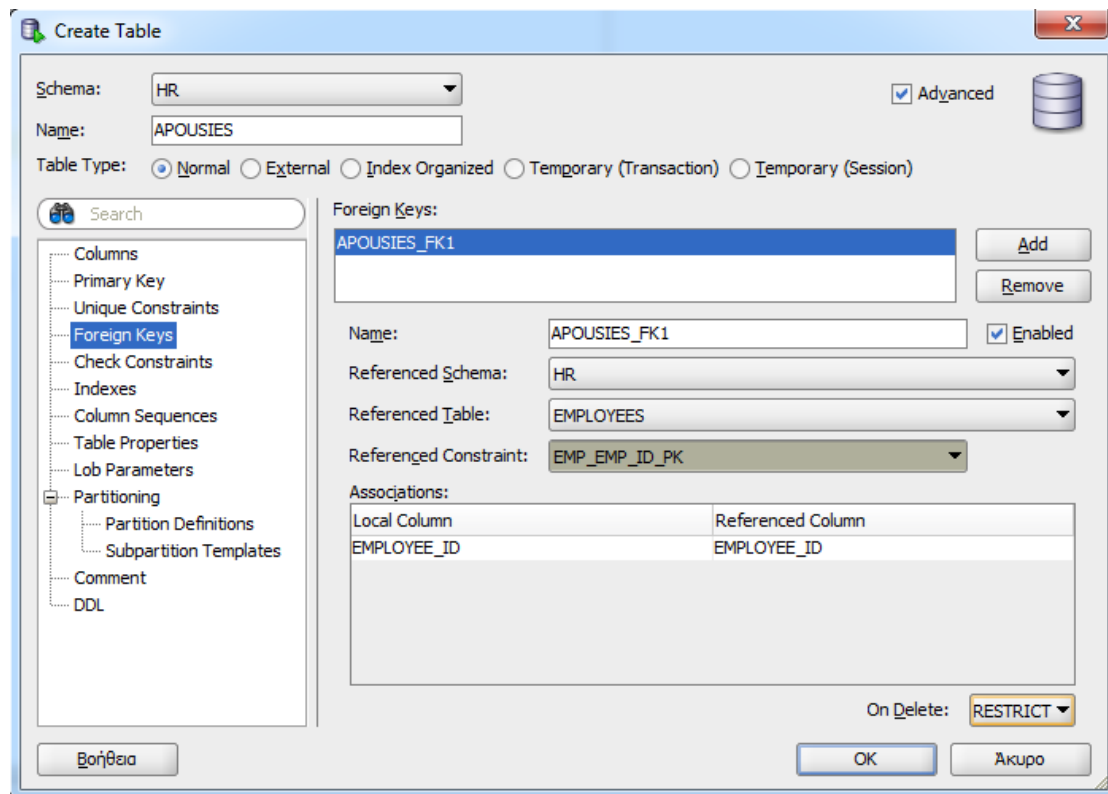
Η δημιουργία πίνακα μπορεί εναλλακτικά να γίνει μέσα από το γραφικό περιβάλλον του SQLDeveloper χρησιμοποιώντας το παράθυρο δημιουργίας πίνακα. Στην ενότητα TABLES κάνουμε δεξί-κλικ και επιλέγουμε New Table. Στο παράθυρο που ακολουθεί συμπληρώνουμε το όνομα του πίνακα (Name) και τα χαρακτηριστικά των στηλών (Column Name, Type, Size, Not Null, Primary Key) όπως φαίνεται στο ακόλουθο σχήμα:



Εικόνα 5-2 : Δημιουργία πίνακα με τον SQLDeveloper

Για τον προσδιορισμού του ξένου κλειδιού, κάνουμε κλικ στην επιλογή Advanced πάνω δεξιά, στο παράθυρο που ανοίγει επιλέγουμε αριστερά Foreign Keys, πατάμε δεξιά το κουμπί Add, επιλέγουμε στο Referenced Schema το σχήμα HR και στο Referenced Table τον πίνακα EMPLOYEES. Τέλος, στο Referenced Constraint επιλέγουμε εκείνο τον περιορισμό του πίνακα EMPLOYEES που αντιστοιχεί στο πεδίο EMPLOYEE\_ID, δηλαδή στην συγκεκριμένη περίπτωση τον EMP\_EMP\_ID\_PK. Αυτόματα στον πίνακα Associations εμφανίζεται το ζεύγος των πεδίων που ορίζουν τη σχέση των πινάκων. Αν στην στήλη Local Column δεν εμφανίζεται το πεδίο που θέλουμε, ανοίγουμε τη λίστα και επιλέγουμε το σωστό (EMPLOYEE\_ID).

Τέλος, μια χρήσιμη επίσης επιλογή είναι η DDL (κάτω αριστερά στο παράθυρο των επιλογών Advanced) η οποία εμφανίζει την ή τις εντολές SQL/DDl οι οποίες αντιστοιχούν στις επιλογές δημιουργίας που έχουμε ορίσει.



Εικόνα 5-3 : Δημιουργία πίνακα με τον SQLDeveloper, επιλογές Advanced

### Δημιουργία πίνακα από το αποτέλεσμα ενός ερωτήματος

Ένας πίνακας μπορεί να δημιουργηθεί και ως το αποτέλεσμα ενός ερωτήματος (μιας εντολής SELECT) με την εντολή **CREATE TABLE ... AS SELECT** της ORACLE. Ο πίνακας που δημιουργείται κληρονομεί αυτόματα τα ονόματα και τους τύπους των πεδίων του αντίστοιχου ερωτήματος, όχι όμως άλλες ιδιότητες (περιορισμούς).

Για παράδειγμα, έστω πως θέλουμε να δημιουργήσουμε έναν νέο πίνακα με όνομα **BONUS** και πεδία **EMPLOYEE\_ID**, **POSO\_BONUS**, ο οποίος θα περιέχει τα μόνους των υπαλλήλων. Το αρχικό **BONUS** θα είναι το 50% του ελάχιστου μισθού της ειδικότητας του κάθε υπαλλήλου (πεδίο **MIN\_SALARY** από τον πίνακα **JOBS**). Αυτό γίνεται με την κατάλληλη εντολή **SELECT** προσθέτοντας μπροστά την εντολή της **SQL/DDL CREATE TABLE AS**:

```
CREATE TABLE BONUS AS
SELECT EMPLOYEES.EMPLOYEE_ID, JOBS.MIN_SALARY/2 AS POSO_BONUS
FROM EMPLOYEES, JOBS
WHERE EMPLOYEES.JOB_ID = JOBS.JOB_ID;
```

### Τροποποίηση δομής πίνακα

Η ORACLE μας δίνει τη δυνατότητα να τροποποιήσουμε την δομή ενός πίνακα μετά τη δημιουργία του. Συγκεκριμένα μπορούμε:

- Να προσθέσουμε στήλη (πεδίο)
- Να τροποποιήσουμε στήλη
- Να διαγράψουμε στήλη

Η τροποποίηση πίνακα γίνεται με την εντολή SQL/DDDL **ALTER TABLE** η οποία έχει την παρακάτω σύνταξη:

```
ALTER TABLE <TABLE_NAME> ADD/MODIFY/DROP (<COL1_DEFINITION>,  
<COL2_DEFINITION>, ...)
```

Οι επιλογές ADD, MODIFY, DROP χρησιμοποιούνται για τις περιπτώσεις προσθήκης, τροποποίησης και διαγραφής στήλης αντίστοιχα ενώ <COL1\_DEFINITION>, <COL2\_DEFINITION>,... είναι οι δηλώσεις των στηλών που προστίθενται, τροποποιούνται ή διαγράφονται.

Επίσης, η ORACLE μας επιτρέπει να προσθέσουμε σε ένα πίνακα περιορισμούς (ελέγχους) που δεν συμπεριλήφθηκαν κατά τη δημιουργία του. Η προσθήκη περιορισμών γίνεται επίσης με την εντολή SQL/DDDL ALTER TABLE με την ακόλουθη σύνταξη:

```
ALTER TABLE <TABLE_NAME> ADD CONSTRAINT <NAME> <TYPE>  
<OPTIONS>
```

όπου <NAME> είναι ένα όνομα για τον περιορισμό, <TYPE> ο τύπος περιορισμού και <OPTIONS> πρόσθετες επιλογές.

Για την καλύτερη κατανόηση των δυνατοτήτων τροποποίησης πίνακα, ας υποθέσουμε πως θέλουμε να κάνουμε τις παρακάτω τροποποιήσεις στους πίνακες APOUSIES και BONUS που δημιουργήσαμε προηγούμενα, και συγκεκριμένα:

1. Να προσθέσουμε στον πίνακα APOUSIES μια νέα στήλη με όνομα REASON (αιτιολογία) τύπου κειμένου με μέγεθος 100 χαρακτήρων.

2. Στη συνέχεια να τροποποιήσουμε το πεδίο αυτό ώστε να χωράει 200 χαρακτήρες.
3. Τώρα να διαγράψουμε εντελώς την στήλη REASON (την δημιουργήσαμε από λάθος).
4. Να προσθέσουμε επιπλέον τον περιορισμό DATE\_FROM <= DATE\_TO στον πίνακα APOUSIES για αποφυγή λαθών στην καταχώρηση.
5. Τέλος, να προσθέσουμε στον νέο πίνακα BONUS τον περιορισμό το πεδίο EMPLOYEE\_ID να αναφέρεται στο πεδίο EMPLOYEES.EMPLOYEE\_ID, δηλαδή να είναι ξένο κλειδί (FOREIGN KEY).

Οι εντολές SQL/DDDL που υλοποιούν τις παραπάνω τροποποιήσεις, σύμφωνα με την γενική σύνταξη που περιγράψαμε, είναι αντίστοιχα οι εξής:

1. ALTER TABLE APOUSIES ADD (REASON VARCHAR2(100));
2. ALTER TABLE APOUSIES MODIFY (REASON VARCHAR2(200));
3. ALTER TABLE APOUSIES DROP (REASON);
4. ALTER TABLE APOUSIES ADD CONSTRAINT CS1 CHECK (DATE\_FROM <= DATE\_TO);
5. ALTER TABLE BONUS ADD CONSTRAINT FK\_BONUS\_EMP\_ID FOREIGN KEY (EMPLOYEE\_ID) REFERENCES EMPLOYEES (EMPLOYEE\_ID);

Στην περίπτωση 4 βλέπουμε ακόμη ένα περιορισμού, τον CHECK (λογικός έλεγχος) ενώ η περίπτωση 5 είναι παρόμοια με τον περιορισμό REFERENCES της εντολής CREATE TABLE.

### Μετονομασία, αντιγραφή, διαγραφή πίνακα

Ολοκληρώνοντας τις εντολές τροποποίησης πίνακα θα δούμε πως μπορούμε να μετονομάσουμε, να αντιγράψουμε ή να διαγράψουμε εντελώς ένα πίνακα.

Η εντολή **RENAME** αλλάζει το όνομα ενός πίνακα σε νέο. Για παράδειγμα, για να μετονομάσουμε τον πίνακα APOUSIES σε ADEIES δίνουμε την εντολή SQL/DDDL:

```
RENAME APOUSIES TO ADEIES;
```

Για τη δημιουργία αντιγράφου ενός πίνακα μπορεί να χρησιμοποιηθεί η εντολή **CREATE TABLE ... AS SELECT** που είδαμε νωρίτερα. Για παράδειγμα, για να δημιουργήσουμε ένα αντίγραφο του πίνακα EMPLOYEES με όνομα EMPLOYEES2, δίνουμε την εντολή SQL/DDl:

```
CREATE TABLE EMPLOYEES2 AS SELECT * FROM EMPLOYEES;
```

Τέλος, η εντολή **DROP TABLE** διαγράφει οριστικά έναν πίνακα. Για παράδειγμα, για να διαγράψουμε τον πίνακα EMPLOYEES2 που δημιουργήσαμε στην προηγούμενη περίπτωση δίνουμε την εντολή SQL/DDl:

```
DROP TABLE EMPLOYEES2;
```

Χρειάζεται ιδιαίτερη προσοχή κατά την διαγραφή πίνακα, και γενικότερα κατά την διαγραφή αντικειμένων, διότι η ενέργεια αυτή δεν είναι δυνατό να αναιρεθεί.<sup>3</sup>

Σημείωση: Η ORACLE δεν επιτρέπει τη διαγραφή ενός πίνακα αν υπάρχουν σε άλλους πίνακες ξένα κλειδιά που αναφέρονται σε αυτόν. Έτσι, για παράδειγμα, δεν μπορούμε να διαγράψουμε τον πίνακα EMPLOYEES διότι ο πίνακας JOB\_HISTORY περιέχει το EMPLOYEE\_ID ως ξένο κλειδί. Αν η ORACLE επέτρεπε κάτι τέτοιο, αυτό θα οδηγούσε σε ασυνέπεια δεδομένων («ορφανές» εγγραφές). Συνειδητοποιούμε λοιπόν τη μεγάλη σημασία της δημιουργίας σχέσεων μεταξύ πινάκων στο στάδιο της σχεδίασης μιας ΒΔ.

## Άλλα αντικείμενα της ORACLE

Όπως έχουμε δει, σε μια ΒΔ ORACLE υπάρχουν πολλοί τύποι αντικειμένων. Ήδη έχουμε γνωρίσει έναν από τους σημαντικότερους τύπους, τους πίνακες. Στην ενότητα αυτή θα γνωρίσουμε τους παρακάτω νέους τύπους αντικειμένων:

- **Όψεις (VIEWS):** Είναι αποθηκευμένα ερωτήματα που μπορούν να επαναχρησιμοποιούνται.

<sup>3</sup> Από την έκδοση 10g και μετά, η ORACLE δίνει, υπό περιορισμούς, τη δυνατότητα επαναφοράς ενός διαγραμμένου πίνακα με την εντολή FLASHBACK TABLE

- **Ευρετήρια (INDEXES):** Είναι δομές που βελτιώνουν την ταχύτητα εύρεσης εγγραφών.
- **Ακολουθίες (SEQUENCES):** Είναι γεννήτριες σειρών αριθμών.

## Όψεις (VIEWS) στην ORACLE

Οι **όψεις (VIEWS)** είναι αποθηκευμένα ερωτήματα, δηλαδή εντολές SELECT από έναν ή περισσότερους πίνακες, με ή χωρίς περιορισμούς, με ή χωρίς ταξινόμηση κοκ. Οποιαδήποτε εντολή SELECT μπορεί να μετατραπεί σε όψη. Οι όψεις αποθηκεύονται ως ονοματισμένα αντικείμενα της ΒΔ και μπορούν να επαναχρησιμοποιούνται, αποφεύγοντας έτσι την ανάγκη να γράφουμε κάθε φορά ξανά την ίδια εντολή SELECT<sup>4</sup>.

Ένα ιδιαίτερα χρήσιμο χαρακτηριστικό των όψεων είναι πως μπορούν να χρησιμοποιούνται μέσα σε άλλες εντολές SELECT (ή ακόμη και μέσα σε άλλες όψεις) ακριβώς σαν να ήταν πίνακες. Με τον τρόπο αυτό μπορούμε να απλουστεύσουμε την σχεδίαση σύνθετων ή πολύπλοκων ερωτημάτων.

Οι όψεις δημιουργούνται με την εντολή SQL/DDDL **CREATE VIEW** της ORACLE η οποία έχει την εξής γενική σύνταξη:

```
CREATE [OR REPLACE] VIEW <VIEW_NAME> AS  
SELECT ...
```

όπου VIEW\_NAME είναι το όνομα της όψης και SELECT ... ένα έγκυρο ερώτημα SELECT.

Για την καλύτερη κατανόηση του τρόπου δημιουργίας και χρήσης μιας όψης, ας υποθέσουμε πως θέλουμε να δημιουργήσουμε μια όψη με όνομα V\_AKRIVOI\_YPALLILOI η οποία να επιστρέφει τους υπαλλήλους εκείνους των οποίων ο μισθός είναι μεγαλύτερος του ελάχιστου της ειδικότητας τους καθώς και να

---

<sup>4</sup> Μια όψη στην ORACLE μπορεί, υπό ορισμένες προϋποθέσεις, να είναι ακόμη και ενημερώσιμη (updatable), δηλαδή να μπορεί να χρησιμοποιηθεί ως πίνακας στις εντολές INSERT, UPDATE και DELETE. Οι συνθήκες αυτές είναι έξω από το πεδίο αυτού του εγχειριδίου. Για περισσότερες πληροφορίες μπορείτε να ανατρέξετε στο εγχειρίδιο SQL Language Reference της ORACLE.

υπολογίζει τη διαφορά αυτή (δηλαδή τη διαφορά Τρέχων\_Μισθός – Ελάχιστος\_Μισθός).

Μια καλή τεχνική για τη δημιουργία όψεων είναι να γράφουμε πρώτα το κατάλληλο ερώτημα με την εντολή SELECT και όταν βεβαιωθούμε πως είναι ορθό να προσθέτουμε την εντολή CREATE VIEW. Στο συγκεκριμένο παράδειγμα, η εντολή SELECT που μας δίνει το επιθυμητό αποτέλεσμα είναι η παρακάτω (φυσικά δεν αποτελεί τον μοναδικό τρόπο υλοποίησης!):

```
SELECT  EMPLOYEES.*, JOBS.MIN_SALARY,
        EMPLOYEES.SALARY - JOBS.MIN_SALARY DIAFORA
FROM    EMPLOYEES, JOBS
WHERE   EMPLOYEES.JOB_ID = JOBS.JOB_ID
AND     EMPLOYEES.SALARY > JOBS.MIN_SALARY;
```

Όπως βλέπουμε, ο περιορισμός EMPLOYEES.SALARY > JOBS.MIN\_SALARY υλοποιεί τη συνθήκη του παραδείγματος (μισθός > ελάχιστου\_μισθού) ενώ η παράσταση EMPLOYEES.SALARY - JOBS.MIN\_SALARY DIAFORA υπολογίζει τη διαφορά και ονομάζει τη στήλη αυτή DIAFORA.

Το μόνο που πρέπει να κάνουμε ώστε να δημιουργήσουμε μια όψη από το παραπάνω ερώτημα είναι να προσθέσουμε την εντολή CREATE VIEW AS:

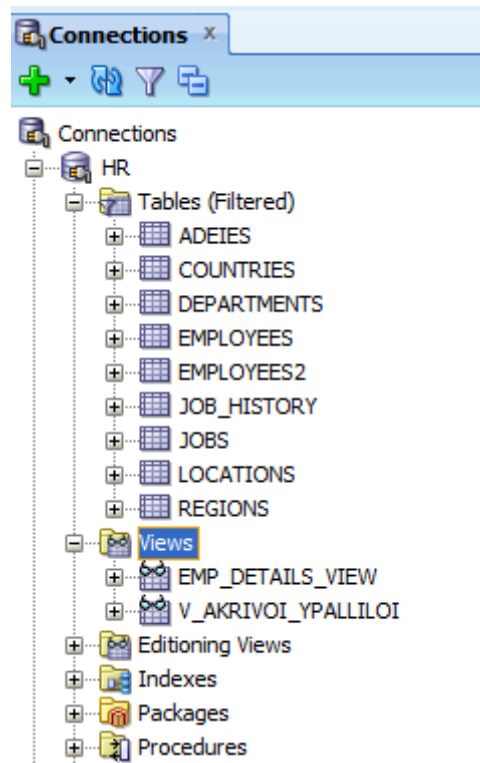
```
CREATE  OR REPLACE VIEW V_AKRIVOI_YPALLILOI AS
SELECT  EMPLOYEES.*, JOBS.MIN_SALARY,
        EMPLOYEES.SALARY - JOBS.MIN_SALARY DIAFORA
FROM    EMPLOYEES, JOBS
WHERE   EMPLOYEES.JOB_ID = JOBS.JOB_ID
AND     EMPLOYEES.SALARY > JOBS.MIN_SALARY;
```

Η πρόταση OR REPLACE αντικαθιστά την όψη, αν υπάρχει, διαφορετικά τη δημιουργεί. Χωρίς την πρόταση OR REPLACE η ORACLE θα σταματήσει με μήνυμα λάθους σε περίπτωση που όψη προϋπάρχει και την ξαναδημιουργούμε, για παράδειγμα μετά από κάποια αλλαγή. Μετά την εκτέλεση της παραπάνω εντολής η ORACLE απαντά

```
view V_AKRIVOI_YPALLILOI created.
```

Η όψη V\_AKRIVOI\_YPALLILOI δημιουργήθηκε και μπορούμε πλέον να τη δούμε κάτω από την ενότητα VIEWS του SQLDeveloper:





Εικόνα 5-4 : Ενότητα VIEWS του SQLDeveloper

Για να χρησιμοποιήσουμε την όψη, δεν έχουμε παρά να γράψουμε μια εντολή SELECT στην οποία η όψη να εμφανίζεται σαν να ήταν ένας πίνακας. Για παράδειγμα:

```
SELECT    *
FROM      V_AKRIVOI_YPALLILOI
ORDER BY  LAST_NAME, FIRST_NAME;
```

Διαγραφή όψης

Η διαγραφή μιας όψης γίνεται με την εντολή DROP VIEW:

```
DROP VIEW V_AKRIVOI_YPALLILOI;
```

## Ευρετήρια (INDEXES) στην ORACLE

Τα **ευρετήρια (INDEXES)** είναι δομές της ORACLE οι οποίες βελτιώνουν σημαντικά την απόδοση κατά την αναζήτηση και γενικότερα κατά την αναφορά σε

εγγραφές στους πίνακες της ΒΔ. Ένα ευρετήριο μπορεί να αφορά ένα ή περισσότερα πεδία ενός πίνακα, που συνήθως είναι εκείνα με τα οποία γίνεται η αναζήτηση των εγγραφών (για παράδειγμα Α.Φ.Μ., Επώνυμο, Κωδικός), ενώ η διαχείριση τους γίνεται αυτόματα από την ORACLE. Ένα ευρετήριο μπορεί να αφορά μόνο ένα πίνακα και συσχετίζεται άμεσα με αυτόν (αν διαγράφει ο πίνακας διαγράφονται αυτόματα και όλα τα ευρετήρια του). Αποτελούν ένα είδος "διευθυνσιολόγιου" των πινάκων, δηλαδή μιας δομής μέσω της οποίας μπορεί να εντοπιστεί γρήγορα η φυσική διεύθυνση μιας εγγραφής από την τιμή ενός ή περισσότερων πεδίων της. Η χρήση ευρετηρίων, αν και είναι προαιρετική, συνίσταται να γίνεται διότι βελτιώνει σε μεγάλο βαθμό την ταχύτητα απόκρισης των εφαρμογών και γενικότερα την απόδοση μιας ΒΔ.

Η δημιουργία ενός INDEX γίνεται με την SQL/DDDL εντολή **CREATE INDEX** με την ακόλουθη γενική σύνταξη:

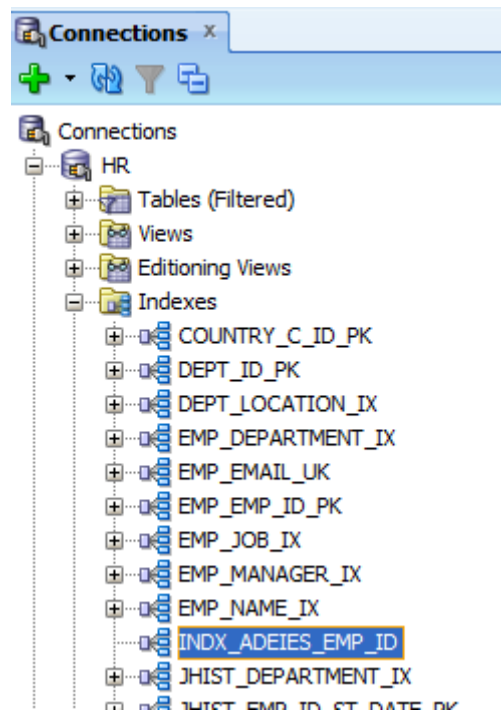
```
CREATE INDEX <INDEX_NAME> ON <TABLE> (<COL1>, <COL2>, ...);
```

όπου <INDEX\_NAME> είναι το όνομα του INDEX, <TABLE> ο πίνακας για τον οποίο δημιουργούμε το INDEX και <COL1>, <COL2>, ... τα πεδία του πίνακα που θα περιέχονται στο INDEX.

Για παράδειγμα, έστω πως θέλουμε να δημιουργήσουμε ένα INDEX με όνομα INDX\_ADEIES\_EMP\_ID για το πεδίο EMPLOYEE\_ID του πίνακα ADEIES. Η εντολή που δημιουργεί το INDEX αυτό είναι:

```
CREATE INDEX INDX_ADEIES_EMP_ID ON ADEIES (EMPLOYEE_ID);
```

Το INDEX που δημιουργήθηκε φαίνεται στον SQLDeveloper κάτω από τη γενική ενότητα INDEXES:



Εικόνα 5-5 : Ενότητα INDEXES του SQLDeveloper

Στο εξής, οποιαδήποτε εντολή SQL, που περιέχει τον πίνακα ADEIES, χρησιμοποιεί το πεδίο `EMPLOYEE_ID` για αναφορά (δηλαδή στην ενότητα `WHERE`) θα εκτελείται πολύ ταχύτερα από πριν. Η χρήση των ευρετηρίων από τις εντολές SQL γίνεται αυτόματα από την ORACLE χωρίς άλλη ενέργεια από τον χρήστη<sup>5</sup>.

Η ORACLE μπορεί να δημιουργήσει αυτόματα ευρετήρια σε κάποια πεδία κατά τη φάση της δημιουργίας ενός πίνακα ακόμη και αν δεν το προσδιορίσει ρητά ο χρήστης. Τέτοιες περιπτώσεις είναι ο ορισμός ενός πεδίου ως πρωτεύον κλειδί ή ως πεδίο με μοναδικές τιμές (`PRIMARY KEY`, `UNIQUE`).

Ειδικότερες επιλογές ευρετηρίων

Ο προσδιορισμός **UNIQUE** κατά τη δημιουργία ενός ευρετηρίου απαγορεύει την εμφάνιση διπλότυπων τιμών στο συγκεκριμένο πεδίο ή ομάδα πεδίων. Για

<sup>5</sup> Παρόλα αυτά, η ORACLE δίνει τη δυνατότητα στο χρήστη να προσδιορίσει σε μια εντολή SQL οδηγίες (optimizer hints) που μεταβάλλουν το *πλάνο εκτέλεσης* (*execution plan*) της εντολής, όπως για παράδειγμα την χρήση ενός εναλλακτικού ευρετηρίου. Οι επιλογές αυτές είναι έξω από το πεδίο αυτού του εγχειριδίου. Για περισσότερες πληροφορίες μπορείτε να ανατρέξετε στο εγχειρίδιο SQL Language Reference της ORACLE.

παράδειγμα, αν υποθέσουμε πως θέλουμε να απαγορεύσουμε να έχει ένας υπάλληλος περισσότερες από μια άδειες οι οποίες να ξεκινούν την ίδια ημέρα, μπορούμε να δημιουργήσουμε το κατάλληλο ευρετήριο με την επιλογή UNIQUE:

```
CREATE UNIQUE INDEX INDX_ADEIES_ID_START ON ADEIES
(EMPLOYEE_ID, DATE_FROM);
```

Με αυτό το ευρετήριο εξασφαλίζουμε πως δεν μπορεί να εμφανιστεί το ίδιο ζεύγος τιμών (EMPLOYEE\_ID, DATE\_FROM) πάνω από μια φορές στον πίνακα ADEIES.

Τέλος, ένα ευρετήριο μπορεί να αφορά ακόμη και *συνάρτηση* ενός πεδίου (function based index), όπως για παράδειγμα ένα ευρετήριο που αφορά μόνο τα 4 πρώτα ψηφία του τηλεφώνου του υπάλληλου και όχι ολόκληρο το τηλέφωνο. Η σύνταξη της εντολής CREATE INDEX δεν αλλάζει, απλά στη θέση του πεδίου προσδιορίζουμε μια *συνάρτηση* αυτού:

```
CREATE INDEX INDX_PHONE_PREFIX ON EMPLOYEES
(SUBSTR(PHONE_NUMBER, 1, 4));
```

Προσοχή όμως: προκειμένου η ORACLE να κάνει χρήση ενός ευρετηρίου συνάρτησης, στην πρόταση WHERE της εντολής SQL θα πρέπει να χρησιμοποιείται η ίδια συνάρτηση πεδίου που έχει οριστεί στο ευρετήριο και όχι ολόκληρο το πεδίο. Έτσι, η εντολή

```
SELECT * FROM EMPLOYEES
WHERE SUBSTR(PHONE_NUMBER, 1, 4) = '0030';
```

θα κάνει χρήση του ευρετηρίου INDX\_PHONE\_PREFIX ενώ η ακόλουθη εντολή όχι:

```
SELECT * FROM EMPLOYEES
WHERE PHONE_NUMBER = '00302108976425';
```

Διαγραφή ευρετηρίου

Η διαγραφή ενός ευρετηρίου γίνεται με την εντολή DROP INDEX, για παράδειγμα:

```
DROP INDEX INDX_PHONE_PREFIX;
```

## Ακολουθίες (SEQUENCES)

Οι ακολουθίες (SEQUENCES) της ORACLE είναι γεννήτριες σειριακών ακολουθιών αριθμών. Μπορούμε να τις παρομοιάσουμε με αυτόματους μετρητές τα χαρακτηριστικά των οποίων ορίζει ο χρήστης. Η αρχική τιμή και το βήμα μπορούν να οριστούν ανάλογα, ενώ μπορεί να επιτρέπεται ή όχι η ανακύκλωση τιμών. Οι ακολουθίες χρησιμοποιούνται στις περιπτώσεις που χρειαζόμαστε μια σειρά αριθμών για τις ανάγκες ενημέρωσης ενός πίνακα, όπως για παράδειγμα ενός Βιβλίου Πρωτοκόλλου.

Οι ακολουθίες δημιουργούνται με την εντολή **CREATE SEQUENCE** με την ακόλουθη γενική σύνταξη:

```
CREATE SEQUENCE <NAME>
[START WITH <NUM>]
[INCREMENT BY <STEP>]
[MAXVALUE/MINVALUE <NUM>] [CYCLE];
```

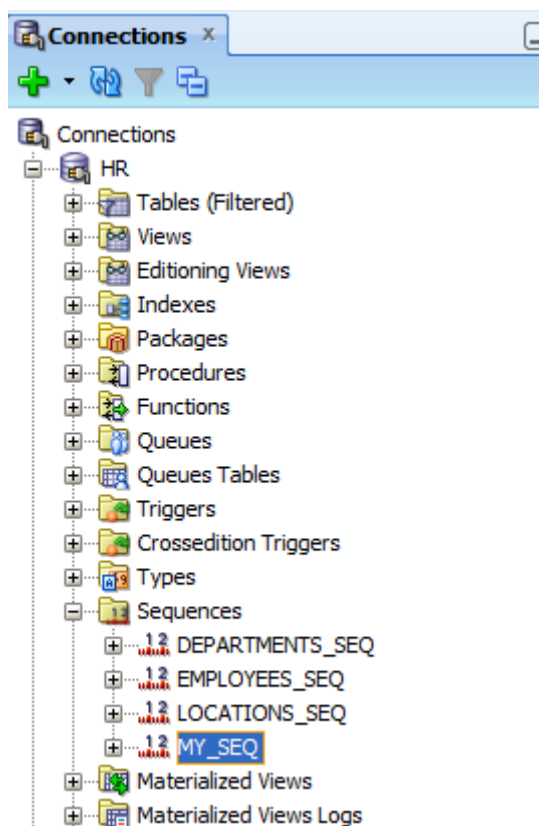
<NAME> είναι το όνομα της ακολουθίας ενώ οι υπόλοιπες παράμετροι είναι προαιρετικές:

- **START WITH <NUM>**: Η αρχική τιμή της ακολουθίας, 1 αν παραληφθεί.
- **INCREMENT BY <STEP>**: Το βήμα της ακολουθίας, 1 αν παραληφθεί. Μπορεί να είναι και αρνητικό, οπότε η ακολουθία είναι φθίνουσα.
- **MAXVALUE/MINVALUE <NUM>**: Η μέγιστη / ελάχιστη τιμή της ακολουθίας.
- **CYCLE**: Η ακολουθία ξεκινά ξανά από την αρχή, αν φθάσει τη μέγιστη τιμή, ή από τέλος, αν είναι φθίνουσα.

Για παράδειγμα, έστω πως θέλουμε να δημιουργήσουμε μια ακολουθία με όνομα MY\_SEQ που να παράγει τους αριθμούς 11, 13, 15, 17, ...99 και μετά ξανά από την αρχή (11). Η ακόλουθη εντολή SQL/DDDL δημιουργεί την ακολουθία αυτή:

```
CREATE SEQUENCE MY_SEQ START WITH 11 INCREMENT BY 2 MINVALUE
11 MAXVALUE 99 CYCLE;
```

Οι ακολουθίες εμφανίζονται κάτω από την ενότητα SEQUENCES του SQLDeveloper:



Εικόνα 5-6 : Ενότητα SEQUENCES του SQLDeveloper

Η λήψη της επόμενης διαθέσιμης τιμής μιας ακολουθίας γίνεται με την ψευδοστήλη **NEXTVAL**. Για παράδειγμα, την πρώτη φορά που θα εκτελεστεί η εντολή

```
SELECT MY_SEQ.NEXTVAL FROM DUAL;
```

επιστρέφει ως αποτέλεσμα το 11. Αν εκτελεστεί ξανά θα επιστρέψει 13 κοκ.

Προσοχή: η χρήση της ψευδοστήλης NEXTVAL προωθεί κάθε φορά την ακολουθία στην επόμενη τιμή της. Αν θέλουμε απλά να δούμε την τρέχουσα τιμή χωρίς να δημιουργήσουμε την επόμενη τιμή, χρησιμοποιούμε την ψευδοστήλη **CURRVAL**. Για παράδειγμα, η εντολή

```
SELECT MY_SEQ.CURRVAL FROM DUAL;
```

επιστρέφει την τρέχουσα τιμή χωρίς να μεταβάλλει την ακολουθία και το αποτέλεσμα της είναι το ίδιο όσες φορές και αν εκτελεστεί.

Όπως αναφέραμε νωρίτερα, συνήθως οι ακολουθίες χρησιμοποιούνται κατά την εισαγωγή εγγραφών όταν χρειαζόμαστε μια σειρά αριθμών για κάποιο πεδίο του πίνακα, δηλαδή σε εντολές INSERT:

```
INSERT INTO <ΚΑΠΟΙΟΣ_ΠΙΝΑΚΑΣ> VALUES (MY_SEQ.NEXTVAL, ...);
```

Διαγραφή ακολουθίας

Η διαγραφή μιας ακολουθίας γίνεται με την εντολή DROP SEQUENCE, για παράδειγμα:

```
DROP SEQUENCE MY_SEQ;
```

## 6. Η ORACLE PL/SQL

### Σκοπός και στόχοι της ενότητας

Σκοπός της ενότητας είναι οι συμμετέχοντες να κατανοήσουν τη σημασία και τον τρόπο σύνταξης και δημιουργίας υποπρογραμμάτων στην ORACLE με τη χρήση της γλώσσας PL/SQL.

Ειδικότερα, οι επιμορφούμενοι θα είναι σε θέση να:

- Κατανοούν τη σημασία, τη δομή και τον τρόπο χρήσης ενός προγράμματος σε ORACLE PL/SQL.
- Να δημιουργούν ένα απλό πρόγραμμα σε ORACLE PL/SQL.
- Να κατανοούν τη σημασία και τον τρόπο διαχείρισης εξαιρέσεων.
- Αναγνωρίζουν τις έννοιες των συναρτήσεων, διαδικασιών και πακέτων.
- Να δημιουργούν απλές συναρτήσεις και διαδικασίες.

### Τι είναι η ORACLE PL/SQL

Η Oracle PL/SQL (Procedural Language / SQL) είναι μια διαδικαστική γλώσσα προγραμματισμού η οποία μας δίνει τη δυνατότητα να συντάξουμε τμήματα κώδικα (προγράμματα) τα οποία υλοποιούν συγκεκριμένες λειτουργίες (συναλλαγές) ή υπολογισμούς. Αποτελεί επέκταση της γλώσσας SQL της ORACLE, με την οποία είναι στενά συνδεδεμένη, και έτσι ενσωματώνει τη λειτουργικότητα της. Δηλαδή, μέσα στον κώδικα PL/SQL μπορούμε να χρησιμοποιήσουμε εντολές της SQL και έτσι έχουμε τη δυνατότητα να υλοποιήσουμε *αλγορίθμους χειρισμού δεδομένων ή υπολογισμών*. Η γλώσσα PL/SQL μοιάζει αρκετά με αντίστοιχες δομημένες γλώσσες προγραμματισμού υψηλού επιπέδου, όπως η Visual Basic και η Java.

Ο κώδικας PL/SQL μπορεί να είναι απλά μια ακολουθία εντολών PL/SQL (*ανώνυμο τμήμα*), να είναι μια *διαδικασία* ή *συνάρτηση* ορισμένη από τον χρήστη (user defined procedure / function) ή να είναι μια συλλογή (βιβλιοθήκη) από δηλώσεις, διαδικασίες ή συναρτήσεις, να είναι δηλαδή ένα *πακέτο*. Οι διαδικασίες, οι συναρτήσεις και τα



πακέτα PL/SQL αποθηκεύονται μέσα στην ΒΔ, αποτελούν δηλαδή ονοματισμένα αντικείμενα της ΒΔ και, γι' αυτόν τον λόγο, ονομάζονται *Αποθηκευμένες Διαδικασίες*, *Αποθηκευμένες Συναρτήσεις* και *Αποθηκευμένα Πακέτα* αντίστοιχα (Stored Procedures, Stored Functions, Stored Packages).

Παραδείγματα προγραμμάτων PL/SQL είναι:

- Μια ρουτίνα ελέγχου ορθότητας του Α.Φ.Μ. ή του IBAN.
- Μια ρουτίνα υπολογισμού μισθοδοσίας.
- Μια ρουτίνα υπολογισμού του φόρου εισοδήματος.

### Πλεονεκτήματα της PL/SQL

Υπάρχουν αρκετοί λόγοι για να χρησιμοποιήσει κανείς τη γλώσσα PL/SQL.: Παρακάτω αναφέρουμε τους κυριότερους

- Ο κώδικας PL/SQL εκτελείται στον database server και όχι στον τοπικό υπολογιστή - πελάτη (client). Έτσι, εκτελείται με ασφάλεια, κάτω από το πλαίσιο δικαιωμάτων του εκάστοτε χρήστη.
- Ο κώδικας PL/SQL αποτελεί αντικείμενο της βάσης δεδομένων, βρίσκεται δηλαδή αποθηκευμένος στην ΒΔ τόσο στην πηγαία μορφή του (source) όσο και σε μεταγλωττισμένη μορφή (compiled). Έτσι, η εκτέλεση γίνεται ταχύτατα διότι δεν απαιτείται ούτε μεταγλώττιση ούτε επαναλαμβανόμενη κίνηση στο δίκτυο επικοινωνίας για κάθε εντολή του τμήματος κώδικα (αρκεί μια κλήση, αυτή για ολόκληρο το τμήμα κώδικα).
- Ένα τμήμα κώδικα PL/SQL αποτελεί αυτόνομη συναλλαγή (transaction), δηλαδή οι επιμέρους εντολές του εκτελούνται είτε σαν ένα σύνολο ή καθόλου. Έτσι, αν μια ενδιάμεση εντολή (για παράδειγμα μια εντολή UPDATE ή INSERT) αποτύχει, τότε ακυρώνονται όλες οι αλλαγές που έγιναν από την αρχή του τμήματος κώδικα. Εξάιρεση αποτελεί η ρητή οδηγία του προγραμματιστή για την οριστικοποίηση των μέχρι ένα σημείο αλλαγών (εντολή COMMIT).

- Χρήση προκαθορισμένων πακέτων (Stored Packages) που δίνει η ORACLE για τη χρήση έτοιμων λειτουργιών ή την υλοποίηση διάφορων διαχειριστικών εργασιών.
- Δυνατότητα προγραμματισμού προσανατολισμένου σε αντικείμενα (object-oriented programming), δηλαδή χρήση του αντικειμενοστραφούς μοντέλου προγραμματισμού για την υλοποίηση εργασιών ή υπολογισμών.
- Χρήση της Δυναμικής SQL (dynamic SQL), εντολών δηλαδή SQL των οποίων η δομή καθορίζεται κατά την εκτέλεση και όχι κατά την σύνταξη του κώδικα.

## Στοιχεία της PL/SQL

Η PL/SQL, ως διαδικαστική γλώσσα, διαθέτει τις περισσότερες από τις δομές των αντίστοιχων δομημένων διαδικαστικών γλωσσών. Οι πιο χρήσιμες από αυτές είναι:

- Η δήλωση μεταβλητών και σταθερών: Η PL/SQL μας δίνει τη δυνατότητα να ορίσουμε τοπικές ή καθολικές μεταβλητές, καθώς και σταθερές, των πιο συνηθισμένων τύπων, προκειμένου να αποθηκεύσουμε τιμές ή να εκτελέσουμε ενδιάμεσους υπολογισμούς.
- Οι δομές ελέγχου ροής: Η PL/SQL διαθέτει, μεταξύ άλλων, τις πιο συνήθεις δομές ελέγχου ροής προγράμματος, όπως:
  - Δομή απόφασης: IF/THEN/ELSE
  - Δομές επανάληψης: FOR-LOOP, WHILE-LOOP
- Οι δρομείς (CURSORS): Δομές που επιτρέπουν την επεξεργασία ενός συνόλου εγγραφών μία προς μία.
- Η σύνταξη διαδικασιών και συναρτήσεων χρήστη (user defined procedures / functions).
- Παγίδευση και χειρισμός λαθών (εξαιρέσεων - exceptions).

## Δομή ενός απλού (ανώνυμου) τμήματος κώδικα PL/SQL

Ένα **ανώνυμο τμήμα** κώδικα, δηλαδή μια ακολουθία δηλώσεων/εντολών σε PL/SQL, ακολουθεί την παρακάτω συντακτική δομή:

```
DECLARE
    <δηλώσεις σταθερών / μεταβλητών>
BEGIN
    <εκτελέσιμες εντολές / εντολές SQL...>
    ...
    <εκτελέσιμες εντολές / εντολές SQL...>
EXCEPTION
    when <exception_name1> then
        <εκτελέσιμες εντολές / εντολές SQL...>
    when <exception_name2> then
        <εκτελέσιμες εντολές / εντολές SQL...>
    ...
END;
```

Οι εκτελέσιμες εντολές / εντολές SQL (προτάσεις μέσα στο τμήμα BEGIN...END) εκτελούνται ακολουθιακά και η εκτέλεση τερματίζεται όταν φτάσει στην τελική εντολή END, εκτός και αν νωρίτερα συμβεί σφάλμα εκτέλεσης.

## Ένα πρώτο παράδειγμα κώδικα σε PL/SQL

Ας δούμε το παρακάτω ανώνυμο τμήμα κώδικα PL/SQL το οποίο αλλάζει τον μισθό ενός συγκεκριμένου υπαλλήλου (έστω EMPLOYEE\_ID=105) και να τον κάνει ίσο με τον μέσο όρο του ελάχιστου και του μέγιστου μισθού της ειδικότητας του, δηλαδή  $(MAX\_SALARY + MIN\_SALARY)/2$ . Αυτό επιτυγχάνεται ανακτώντας από τον πίνακα JOBS τον ελάχιστο και μέγιστο μισθό που αντιστοιχεί στη θέση εργασίας του, υπολογίζοντας τον μέσο όρο και ενημερώνοντας τον μισθό του υπαλλήλου στον πίνακα EMPLOYEES.

```
DECLARE
    ELAXISTOS  NUMBER(6);
    MEGISTOS   JOBS.MAX_SALARY%TYPE; /*ίδιος τύπος με το
περίο JOBS.MAX_SALARY*/
    NEOS_MISTHOS EMPLOYEES.SALARY%TYPE;
BEGIN
    -- Είμαι ένα σχόλιο, η ORACLE με αγνοεί
    /* Είμαι ένα άλλο σχόλιο, τελειώνω με */
    SELECT      JOBS.MIN_SALARY, JOBS.MAX_SALARY
    INTO        ELAXISTOS, MEGISTOS
```

```

FROM      EMPLOYEES, JOBS
WHERE     EMPLOYEES.JOB_ID = JOBS.JOB_ID
AND       EMPLOYEES.EMPLOYEE_ID = 105;

NEOS_MISTHOS := (ELAXISTOS+MEGISTOS) /2;

UPDATE    EMPLOYEES
SET       SALARY = NEOS_MISTHOS
WHERE     EMPLOYEE_ID = 105;
END; /*Η επόμενη γραμμή (/) χρειάζεται στην SQL*Plus μόνο */
/

```

### Παράδειγμα 6- 1

Μέσα από το παράδειγμα αυτό έχουμε την πρώτη επαφή με μια σειρά από στοιχεία της PL/SQL τα οποία θα συζητήσουμε παρακάτω.

## Δηλώσεις μεταβλητών και σταθερών

Στο παραπάνω παράδειγμα, το τμήμα μεταξύ της λέξης DECLARE και της λέξης BEGIN ονομάζεται *τμήμα δηλώσεων* και σε αυτό δηλώνουμε τις μεταβλητές και σταθερές που θα χρησιμοποιήσουμε στο τμήμα κώδικα που ακολουθεί. Μια μεταβλητή ή σταθερά δεν μπορεί να χρησιμοποιηθεί αν δεν δηλωθεί. Αν δεν απαιτούνται μεταβλητές ή σταθερές για το τμήμα κώδικα που θα γράψουμε τότε παραλείπουμε τη λέξη DECLARE και το τμήμα δηλώσεων και ξεκινάμε κατευθείαν με τη λέξη BEGIN.

Η δήλωση μεταβλητών και σταθερών υπακούει στους ίδιους κανόνες που ισχύουν για τη δήλωση πεδίων κατά τη δημιουργία ενός πίνακα και η γενική μορφή είναι:

<Όνομα μεταβλητής> <τύπος> [προαιρετική αρχική τιμή]

Έτσι η πρόταση

```
ELAXISTOS      NUMBER (6) ;
```

δημιουργεί τη μεταβλητή ELAXISTOS η οποία είναι αριθμητικού τύπου με μήκος 6 ψηφία χωρίς δεκαδικά ψηφία. Εναλλακτικά, ο τύπος μιας μεταβλητής μπορεί να δηλωθεί έμμεσα ίδιος με τον τύπο ενός πεδίου πίνακα γράφοντας αντί για τον τύπο την πρόταση <πίνακας>.<πεδίο>%TYPE. Έτσι η πρόταση

```
MEGISTOS      JOBS.MAX_SALARY%TYPE;
```

δημιουργεί τη μεταβλητή MEGISTOS η οποία έχει τον ίδιο τύπο με το πεδίο MAX\_SALARY του πίνακα JOBS, δηλαδή NUMBER(6) επίσης. Ομοίως, η πρόταση

```
NEOS_MISTHOS EMPLOYEES.SALARY%TYPE;
```

δηλώνει τη μεταβλητή NEOS\_MISTHOS η οποία έχει τον ίδιο τύπο με το πεδίο SALARY του πίνακα EMPLOYEES, δηλαδή NUMBER(8,2).

Οι μεταβλητές αρχικά περιέχουν την τιμή NULL και είναι ευθύνη του προγραμματιστή να τους αποδώσει τιμές. Μπορούμε προαιρετικά να δώσουμε μια αρχική τιμή σε μια μεταβλητή, ταυτόχρονα με τη δήλωση της, προσθέτοντας μετά τον τύπο την πρόταση := <αρχική τιμή>. Έτσι, η πρόταση

```
NEOS_MISTHOS EMPLOYEES.SALARY%TYPE := 1200;
```

δηλώνει τη μεταβλητή NEOS\_MISTHOS με αρχική τιμή 1200.

Τέλος, για να δηλώσουμε μια σταθερά (δηλαδή μια ονοματισμένη σταθερή τιμή η οποία δεν μπορεί να μεταβληθεί) ακολουθούμε τη σύνταξη της τελευταίας περίπτωσης προσθέτοντας πριν τον τύπο τη λέξη CONSTANT:

```
CREDIT_LIMIT CONSTANT NUMBER(8,2) := 5000.00;
```

## Σχόλια και κενές γραμμές στην PL/SQL

Μια γραμμή που ξεκινά με -- θεωρείται σχόλιο και δεν λαμβάνεται υπόψη από την ORACLE. Εναλλακτικά, μπορούμε να περικλείσουμε περισσότερες γραμμές μεταξύ των συμβόλων /\* και \*/ και τότε αυτές ομοίως δεν λαμβάνονται υπόψη από την ORACLE:

```
-- Είμαι ένα σχόλιο, η Oracle με αγνοεί  
/*Είμαι ένα άλλο σχόλιο,  
τελειώνω με */
```

Οι κενές γραμμές δεν λαμβάνονται υπόψη από την ORACLE και μπορούν να παρεμβάλλονται όπου κρίνουμε χρήσιμο για την καλύτερη αναγνωσιμότητα του κώδικα.

## Η εντολή SELECT INTO

Η εντολή **SELECT INTO** λειτουργεί όπως ακριβώς η εντολή **SELECT** της SQL με τη διαφορά ότι οι τιμές των πεδίων που επιστρέφει αποθηκεύονται αντίστοιχα (μια προς μια) στις μεταβλητές που ακολουθούν τη λέξη **INTO**. Γι' αυτόν τον λόγο το πλήθος των στηλών της εντολής **SELECT** πρέπει να είναι ίδιο με το πλήθος των μεταβλητών μετά τη λέξη **INTO**. Ένας σημαντικός όμως περιορισμός της εντολής **SELECT INTO** είναι πως πρέπει να επιστρέφει σαν αποτέλεσμα ακριβώς μια εγγραφή, διαφορετικά η εντολή αποτυγχάνει με μήνυμα λάθους. Έτσι η εντολή

```
SELECT      JOBS.MIN_SALARY, JOBS.MAX_SALARY
INTO        ELAXISTOS, MEGISTOS
FROM        EMPLOYEES, JOBS
WHERE       EMPLOYEES.JOB_ID = JOBS.JOB_ID
AND         EMPLOYEES.EMPLOYEE_ID = 105;
```

αποθηκεύει τις τιμές **JOBS.MIN\_SALARY**, **JOBS.MAX\_SALARY** που αφορούν τον υπάλληλο με κωδικό 105 στις μεταβλητές **ELAXISTOS**, **MEGISTOS** αντίστοιχα.

## Η εντολή εκχώρησης

Η εντολή εκχώρησης **:=** αποδίδει μια τιμή σε μια μεταβλητή. Η τιμή μπορεί να είναι μια σταθερά ή μια παράσταση. Έτσι, η εντολή

```
NEOS_MISTHOS := (ELAXISTOS+MEGISTOS) /2;
```

εκχωρεί στην μεταβλητή **NEOS\_MISTHOS** τον μέσο όρο του ελάχιστου και μέγιστου μισθού που αντιστοιχεί στην θέση εργασίας του υπαλλήλου.

## Οι εντολές INSERT, UPDATE, DELETE στην PL/SQL

Οι εντολές **INSERT**, **UPDATE**, **DELETE** μέσα σε ένα τμήμα κώδικα **PL/SQL** χρησιμοποιούνται ακριβώς όπως και στην απλή **SQL** με τη δυνατότητα όμως της χρήσης μεταβλητών στην θέση σταθερών τιμών. Έτσι, η εντολή

```
UPDATE      EMPLOYEES
SET         SALARY = NEOS_MISTHOS
WHERE       EMPLOYEE_ID = 105;
```

ενημερώνει τον μισθό του υπαλλήλου με κωδικό 105 με την τιμή της μεταβλητής  
NEOS\_MISTHOS.

## Εκτέλεση ενός ανώνυμου τμήματος κώδικα PL/SQL

Η εκτέλεση ενός ανώνυμου τμήματος κώδικα PL/SQL στον SQLDeveloper γίνεται  
όπως ακριβώς και η εκτέλεση μιας εντολής της απλής SQL (στην SQL\*Plus  
χρειάζεται επιπλέον να πληκτρολογήσουμε κάτω από την τελευταία γραμμή τον  
χαρακτήρα / (forward slash) και να πατήσουμε <ENTER>). Εάν δεν υπάρχουν  
σφάλματα, η ORACLE επιστρέφει το μήνυμα

```
anonymous block completed (SQLDeveloper) ή
```

```
PL/SQL procedure successfully completed. (SQL*Plus)
```

διαφορετικά εμφανίζει το μήνυμα σφάλματος και τον αριθμό γραμμής στον οποίο  
συνέβη.

## Η δομή ελέγχου ροής IF...THEN

Η δομή ελέγχου ροής **IF...THEN** χρησιμοποιείται για την εκτέλεση εντολών υπό  
συνθήκη. Οι εντολές που περικλείονται στην δομή IF...THEN εκτελούνται μόνο αν η  
σχετική συνθήκη ελέγχου είναι αληθής.

Η σύνταξη της δομής IF...THEN είναι:

```
IF <συνθήκη> THEN  
    <εντολές>  
ELSIF <συνθήκη> THEN  
    <εντολές>  
ELSE  
    <εντολές>  
END IF;
```

Η ORACLE εξετάζει τις συνθήκες από πάνω προς τα κάτω και εκτελεί τις εντολές  
που περικλείονται στην πρώτη συνθήκη που βρίσκει να είναι αληθής. Αν μια  
συνθήκη δίπλα από τις λέξεις IF ή ELSIF αληθεύει, οι υπόλοιπες συνθήκες δεν  
εξετάζονται και ο έλεγχος μεταφέρεται μετά την εντολή END IF. Αν καμία συνθήκη  
δεν είναι αληθής τότε εκτελείται το τμήμα ELSE, αν υπάρχει.

## Το τμήμα

```
ELSIF <συνθήκη> THEN
    <εντολές>
```

μπορεί να επαναληφθεί όσες φορές απαιτείται ενώ είναι προαιρετικό, όπως και το τελικό τμήμα ELSE.

Για παράδειγμα, ας υποθέσουμε πως θέλουμε να τροποποιήσουμε το προηγούμενο τμήμα κώδικα PL/SQL ώστε να ελέγχει πρώτα εάν ο τρέχων μισθός του συγκεκριμένου υπαλλήλου είναι μικρότερος από τον μέσο όρο του ελάχιστου και του μέγιστου μισθού της ειδικότητας του, δηλαδή  $(MAX\_SALARY + MIN\_SALARY) / 2$  και, εάν είναι, τότε και μόνο να τον κάνει ίσο με τον μέσο όρο. Έτσι, μετά τις απαραίτητες αλλαγές, ο κώδικας διαμορφώνεται ως εξής:

```
SET SERVEROUTPUT ON

DECLARE
    MISTHOS          EMPLOYEES.SALARY%TYPE;
    ELAXISTOS        JOBS.MIN_SALARY%TYPE;
    MEGISTOS         JOBS.MAX_SALARY%TYPE;
    NEOS_MISTHOS     EMPLOYEES.SALARY%TYPE;
BEGIN
    SELECT      EMPLOYEES.SALARY, JOBS.MIN_SALARY,
                JOBS.MAX_SALARY
    INTO        MISTHOS, ELAXISTOS, MEGISTOS
    FROM        EMPLOYEES, JOBS
    WHERE       EMPLOYEES.JOB_ID = JOBS.JOB_ID
    AND         EMPLOYEES.EMPLOYEE_ID = 105;

    NEOS_MISTHOS := (ELAXISTOS + MEGISTOS) / 2;

    IF (MISTHOS < NEOS_MISTHOS ) THEN
        UPDATE   EMPLOYEES
        SET       SALARY = NEOS_MISTHOS
        WHERE     EMPLOYEE_ID = 105;

        DBMS_OUTPUT.PUT_LINE('NEOS MISTHOS = ' ||
                               NEOS_MISTHOS);
    END IF;
END;
```

Όπως παρατηρούμε, η εντολή IF (MISTHOS < NEOS\_MISTHOS ) THEN καθορίζει πως η αλλαγή του μισθού του υπαλλήλου (εντολή UPDATE) θα γίνει μόνο αν ο τρέχων μισθός είναι μικρότερος από τον νέο υπολογισμένο μισθό.



Παράλληλα, βλέπουμε μια νέα εντολή, την

```
DBMS_OUTPUT.PUT_LINE('NEOS MISTHOS = ' || NEOS_MISTHOS);
```

Η εντολή αυτή είναι μια προκαθορισμένη διαδικασία της ORACLE και χρησιμοποιείται για την εμφάνιση μηνυμάτων χρήστη στην οθόνη του τερματικού. Το μήνυμα μπορεί να είναι μια σταθερή αλφαριθμητική τιμή ή, όπως βλέπουμε, μια παράσταση, όπως η συνένωση τιμών με τον τελεστή συνένωσης || (concatenation operator). Για να ενεργοποιηθεί η εμφάνιση μηνυμάτων χρήστη στην οθόνη του τερματικού απαιτείται να εκτελεστεί μια μόνο φορά στην αρχή η εντολή

```
SET SERVEROUTPUT ON
```

## Η δομή επανάληψης WHILE...LOOP

Η δομή επανάληψης **WHILE...LOOP** χρησιμοποιείται για την επανάληψη μιας ομάδας εντολών PL/SQL σύμφωνα με μια λογική συνθήκη ελέγχου. Ακολουθεί την εξής σύνταξη:

```
WHILE <συνθήκη> LOOP
    <εντολές PL/SQL>
    ...
END LOOP;
```

Όσο η λογική συνθήκη ελέγχου είναι αληθής η ομάδα εντολών επαναλαμβάνεται, δηλαδή η ροή του κώδικα επανέρχεται στην εντολή που ακολουθεί την πρόταση WHILE...LOOP. Η συνθήκη ελέγχεται πριν από κάθε επανάληψη. Αν η συνθήκη δεν ισχύει η ροή του κώδικα μεταφέρεται στην εντολή που ακολουθεί την πρόταση END LOOP. Ο αριθμός των επαναλήψεων, όπως προκύπτει από τον τρόπο λειτουργίας της δομής WHILE...LOOP, δεν είναι γνωστός εκ των προτέρων αφού εξαρτάται από την τιμή της λογικής συνθήκης ελέγχου.

Για παράδειγμα, ας υποθέσουμε πως πρέπει να δώσουμε αύξηση στον υπάλληλο της προηγούμενης άσκησης με την ακόλουθη μέθοδο:

1. Εάν ο μισθός είναι μικρότερος του μέσου όρου ελάχιστου – μέγιστου, να δώσουμε αύξηση 2%.
2. Να επαναλάβουμε το βήμα 1 μέχρι ο μισθός να ξεπεράσει τον μέσο όρο ελάχιστου – μέγιστου.

Είναι προφανές πως η υλοποίηση της μεθόδου αυτής απαιτεί τη χρήση μιας επαναληπτικής δομής. Μετά τις απαραίτητες τροποποιήσεις ο προηγούμενος κώδικας διαμορφώνεται ως εξής (φυσικά δεν είναι ο μόνος τρόπος επίλυσης):

```

DECLARE
    MISTHOS          EMPLOYEES.SALARY%TYPE;
    ELAXISTOS        JOBS.MIN_SALARY%TYPE;
    MEGISTOS         JOBS.MAX_SALARY%TYPE;
    NEOS_MISTHOS     EMPLOYEES.SALARY%TYPE;
    SYNT_AYXISIS     CONSTANT NUMBER := 1.02;
BEGIN
    SELECT      EMPLOYEES.SALARY, JOBS.MIN_SALARY,
                JOBS.MAX_SALARY
    INTO        MISTHOS, ELAXISTOS, MEGISTOS
    FROM        EMPLOYEES, JOBS
    WHERE       EMPLOYEES.JOB_ID = JOBS.JOB_ID
    AND         EMPLOYEES.EMPLOYEE_ID = 105;

    NEOS_MISTHOS := MISTHOS;

    WHILE NEOS_MISTHOS < (ELAXISTOS+MEGISTOS) /2 LOOP
        NEOS_MISTHOS := NEOS_MISTHOS * SYNT_AYXISIS;
    END LOOP;

    IF (MISTHOS <> NEOS_MISTHOS ) THEN
        UPDATE      EMPLOYEES
        SET          SALARY = NEOS_MISTHOS
        WHERE        EMPLOYEE_ID = 105;

        DBMS_OUTPUT.PUT_LINE('NEOS MISTHOS = ' ||
                               NEOS_MISTHOS);
    END IF;
END;
```

Η δομή επανάληψης WHILE...LOOP ελέγχει κάθε φορά εάν ο νέος μισθός είναι μικρότερος του μισθού-στόχου πριν κάνει την αντίστοιχη αναπροσαρμογή, δηλαδή τον πολλαπλασιασμό με τον συντελεστή αύξησης SYNT\_AYXISIS. Για τον υπολογισμό της αύξησης 2% ορίζεται στο τμήμα δηλώσεων η σταθερά SYNT\_AYXISIS ώστε να είναι εύκολη η αλλαγή του ποσοστού χωρίς άλλες τροποποιήσεις. Η εντολή IF (MISTHOS <> NEOS\_MISTHOS) THEN...

εξασφαλίζει την αποφυγή της περιττής ενημέρωσης του μισθού εάν αυτός δεν μεταβλήθηκε.

## Η δομή επανάληψης FOR...LOOP

Η δομή επανάληψης **FOR...LOOP** χρησιμοποιείται για την επανάληψη μιας ομάδας εντολών για έναν προκαθορισμένο αριθμό επαναλήψεων. Συντάσσεται ως εξής:

```
FOR <μετρητής> IN [REVERSE] <αρχ_τιμή>..<τελ_τιμή> LOOP
    <εντολές PL/SQL>
...
END LOOP;
```

Ο <μετρητής> πρέπει να είναι ένα έγκυρο αναγνωριστικό ORACLE και δεν χρειάζεται να δηλωθεί στο τμήμα δηλώσεων. Η δομή επαναλαμβάνεται για όλες τις διακριτές τιμές από <αρχ\_τιμή>..<τελ\_τιμή>, ο <μετρητής> περιέχει σε κάθε επανάληψη την τρέχουσα τιμή ενώ μπορεί να χρησιμοποιηθεί σε οποιαδήποτε παράσταση εντός της δομής. Η προαιρετική λέξη REVERSE μετά την IN έχει ως αποτέλεσμα οι τιμές να διατρέχονται με αντίστροφη σειρά, δηλαδή από <τελ\_τιμή>..<αρχ\_τιμή>.

Για παράδειγμα, το παρακάτω τμήμα κώδικα PL/SQL υπολογίζει το άθροισμα των αριθμών 1...25 χρησιμοποιώντας τη δομή FOR...LOOP.

```
DECLARE
    ATHROISMA  NUMBER(3) := 0;
BEGIN
    FOR i in 1..25 LOOP
        ATHROISMA := ATHROISMA + i;
    END LOOP;

    dbms_output.put_line('Άθροισμα 1..25 = ' || ATHROISMA);
END;
```

Η δομή FOR...LOOP μπορεί επίσης να χρησιμοποιηθεί για την υλοποίηση ενός δρομέα (cursor) στην ORACLE, όπως θα δούμε στην επόμενη ενότητα.

## Δρομείς (CURSORS)

Ο δρομέας (CURSOR) είναι μια επαναληπτική δομή της PL/SQL η οποία επιτρέπει την ανάκτηση ή/και την επεξεργασία των εγγραφών ενός ερωτήματος μια προς μια. Οι δρομείς χρησιμοποιούνται για να χειριστούμε (για παράδειγμα να ενημερώσουμε) ένα σύνολο εγγραφών όταν αυτό δεν είναι δυνατό ή εύκολο να γίνει με απλές εντολές SQL. Για κάθε εγγραφή ο δρομέας μας δίνει τη δυνατότητα να εκτελέσουμε όσες εντολές χρειαζόμαστε για τη συγκεκριμένη επεξεργασία.

Υπάρχουν διάφοροι τρόποι δημιουργίας δρομέα στην PL/SQL. Ο πιο απλός είναι ο **FOR...LOOP CURSOR** ο οποίος έχει την ακόλουθη σύνταξη:

```
FOR <CURSOR_NAME> IN (<select query>) LOOP
    <εντολές SQL - PL/SQL>
END LOOP;
```

Το <CURSOR\_NAME > πρέπει να είναι ένα έγκυρο αναγνωριστικό ORACLE και δεν χρειάζεται να δηλωθεί στο τμήμα δηλώσεων του κώδικα, ενώ το <select query> πρέπει να είναι έγκυρο ερώτημα SELECT.

Έστω, για παράδειγμα, πως πρέπει να δώσουμε την αύξηση με τον αλγόριθμο που υλοποιήσαμε στην προηγούμενη περίπτωση (προοδευτικά 2%), αυτή τη φορά όμως όχι σε ένα συγκεκριμένο υπάλληλο αλλά σε ένα υποσύνολο ή σε όλους τους υπαλλήλους. Πρέπει λοιπόν να τροποποιήσουμε τον κώδικα ώστε να επεξεργαστεί μια-μια τις εγγραφές των αντίστοιχων υπαλλήλων, να «περάσει» δηλαδή από έναν-έναν υπάλληλο και να κάνει τις ανάλογες πράξεις. Ο κώδικας που υλοποιεί τη λύση αυτή για όλους τους υπαλλήλους φαίνεται παρακάτω:

```
DECLARE
    NEOS_MISTHOS      EMPLOYEES.SALARY%TYPE;
BEGIN
    FOR X IN (
        SELECT      EMPLOYEES.SALARY, JOBS.MIN_SALARY,
                    JOBS.MAX_SALARY, EMPLOYEES.ROWID
        FROM        EMPLOYEES, JOBS
        WHERE       EMPLOYEES.JOB_ID = JOBS.JOB_ID) LOOP

        NEOS_MISTHOS := X.SALARY;

        WHILE NEOS_MISTHOS < (X.MIN_SALARY+X.MAX_SALARY) /2
        LOOP
            NEOS_MISTHOS := NEOS_MISTHOS * 1.02;
        END LOOP;
    END LOOP;
```

```
IF (X.SALARY <> NEOS_MISTHOS) THEN
    UPDATE      EMPLOYEES
    SET         SALARY = NEOS_MISTHOS
    WHERE       ROWID = X.ROWID;

    DBMS_OUTPUT.PUT_LINE('NEOS MISTHOS = ' ||
        NEOS_MISTHOS);
END IF;
END LOOP;
END;
```

Η πρόταση FOR X IN (...) δημιουργεί τον δρομέα X ο οποίος διατρέχει μια προς μια όλες τις εγγραφές του ερωτήματος που περικλείεται στην παρένθεση. Μεταξύ των λέξεων LOOP και END LOOP περιέχονται οι εντολές που θέλουμε να εκτελεστούν για κάθε εγγραφή, δηλαδή οι ίδιες εντολές που χρησιμοποιήσαμε στο τελευταίο παράδειγμα για την προοδευτική αύξηση 2%. Η αναφορά όμως τώρα στα πεδία της τρέχουσας κάθε φορά εγγραφής γίνεται βάζοντας σε αυτά το πρόθεμα "<όνομα\_δρομέα>.", δηλαδή στην περίπτωση αυτή "X.". Έτσι η αναφορά στο πεδίο SALARY γίνεται με την πρόταση "X.SALARY". Τέλος, παρατηρούμε πως για την ενημέρωση της εγγραφής του υπαλλήλου (δηλαδή στην εντολή UPDATE EMPLOYEES...) χρησιμοποιείται στην συνθήκη WHERE η ψευδοστήλη ROWID ώστε να είναι γρηγορότερη η αναφορά στην εκάστοτε εγγραφή. Φυσικά αυτό δεν είναι απαραίτητο και η συνθήκη WHERE θα μπορούσε να περιέχει στη θέση του ROWID κάποιο άλλο πεδίο αναφοράς, όπως το EMPLOYEE\_ID, δηλαδή WHERE EMPLOYEE\_ID = X. EMPLOYEE\_ID.

## Εξαιρέσεις (exceptions)

Ένα τμήμα κώδικα PL/SQL δεν μπορεί να εκτελεστεί αν δεν είναι συντακτικά έγκυρο. Δηλαδή, η ORACLE δεν θα επιτρέψει την εκτέλεση του μέχρι να διορθωθούν όλα τα συντακτικά σφάλματα. Τα συντακτικά σφάλματα εντοπίζονται κατά την απόπειρα εκτέλεσης του κώδικα και, εάν υπάρχουν, τυπώνονται στην οθόνη του τερματικού χωρίς να εκτελεστεί ο κώδικας ώστε ο προγραμματιστής να προβεί στην διόρθωση τους.

Κατά την εκτέλεση ενός τμήματος κώδικα PL/SQL είναι όμως επίσης πιθανό να συμβούν *λογικά σφάλματα εκτέλεσης (runtime errors)*. Τα σφάλματα αυτά δεν

μπορούν να εντοπιστούν όλα εξ' αρχής διότι μπορεί να συμβαίνουν κάτω από συγκεκριμένες συνθήκες (συγκεκριμένες τιμές μεταβλητών ή δεδομένων πινάκων) οι οποίες πιθανά και να μην ισχύσουν ποτέ. Ένα οικείο παράδειγμα είναι η διαίρεση με μηδέν ή η ανάθεση μιας μεγάλης τιμής σε μια μεταβλητή στην οποία δεν χωρά. Σφάλματα εκτέλεσης μπορούν να συμβούν ακόμη και χωρίς υπαιτιότητα του προγραμματιστή, όπως στην περίπτωση αποτυχίας μέσου αποθήκευσης (disk failure) ή σφάλματος δικτύου (network error). Αν συμβεί σφάλμα εκτέλεσης, η εκτέλεση του κώδικα διακόπτεται και, όπως αναφέραμε νωρίτερα, το πλεονέκτημα της PL/SQL είναι πως αναιρούνται όλες οι αλλαγές που έγιναν στην βάση δεδομένων από την αρχή του κώδικα.

Ο προγραμματιστής μπορεί να προβλέψει και να ενσωματώσει τους κατάλληλους ελέγχους (IF...THEN) στα σημεία του κώδικα που αναμένει να συμβεί σφάλμα, παρόλα αυτά είναι δύσκολο να προβλέψει όλα τα πιθανά σφάλματα. Για τον σκοπό αυτό, η PL/SQL δίνει τη δυνατότητα *χειρισμού εξαιρέσεων*.

Μια εξαίρεση είναι μια κατάσταση λάθους που παράγεται από την ORACLE αν συμβεί σφάλμα κατά την εκτέλεση κώδικα PL/SQL. Η εκτέλεση τότε σταματά και «μεταπηδά» στο τμήμα διαχείρισης της εξαίρεσης. Το τμήμα αυτό, όπως είδαμε νωρίτερα, γράφεται στο τέλος του εκάστοτε τμήματος κώδικα. Μπορούμε να αφήσουμε την ORACLE να διαχειριστεί με προκαθορισμένο τρόπο τις εξαιρέσεις (να τυπώνει τα προκαθορισμένα μηνύματα λάθους), μπορούμε να ορίσουμε δικές μας ενέργειες ή να κάνουμε συνδυασμό των δυο.

Οι εντολές διαχείρισης εξαιρέσεων γράφονται στην ενότητα EXCEPTION με την ακόλουθη σύνταξη:

```
EXCEPTION
    WHEN <EXCEPTION_NAME> THEN
        <ΕΝΕΡΓΕΙΕΣ>
    WHEN <EXCEPTION_NAME> THEN
        < ΕΝΕΡΓΕΙΕΣ >
    . . . .
    . . . .
```

Όπως βλέπουμε, μπορούμε να συμπεριλάβουμε εντολές διαχείρισης για όσες διαφορετικές εξαιρέσεις μας ενδιαφέρουν.

Μερικές προκαθορισμένες εξαιρέσεις (σφάλματα) στην PL/SQL είναι οι παρακάτω:

ΟΝΟΜΑΣΙΑ ΕΞΑΙΡΕΣΗΣ	ΠΕΡΙΓΡΑΦΗ / ΑΙΤΙΑ
<b>ZERO_DIVIDE</b>	Έγινε απόπειρα διαίρεσης με το μηδέν.
<b>VALUE_ERROR</b>	Για παράδειγμα μια αλφαριθμητική τιμή δεν «χωρά» σε μια μεταβλητή.
<b>NO_DATA_FOUND</b>	Δεν βρέθηκε καμιά εγγραφή κατά την εντολή SELECT...INTO.
<b>TOO_MANY_ROWS</b>	Βρέθηκαν παραπάνω από μια εγγραφές κατά την εντολή SELECT...INTO. Η εντολή SELECT...INTO πρέπει να επιστρέφει ακριβώς μια εγγραφή
<b>INVALID_NUMBER</b>	Η μετατροπή ενός αλφαριθμητικού σε αριθμό απέτυχε
<b>LOGIN_DENIED</b>	Απόπειρα σύνδεσης με λανθασμένο username ή password.
<b>OTHERS</b>	Όλες οι εξαιρέσεις.

**Πίνακας 6- 1** Προκαθορισμένες εξαιρέσεις στην PL/SQL

Σαν παράδειγμα χειρισμού εξαίρεσης, ας υποθέσουμε πως θέλουμε να προσθέσουμε στον κώδικα του **παραδείγματος 6-1** ένα τμήμα διαχείρισης εξαιρέσεων που, σε περίπτωση εξαίρεσης (σφάλματος), να τυπώνει ένα μήνυμα λάθους, το μήνυμα της εξαίρεσης και να σταματά την εκτέλεση του κώδικα. Οι εντολές που προσθέτουμε φαίνονται στο τμήμα EXCEPTION. Έτσι, ο κώδικας διαμορφώνεται ως εξής:

```

DECLARE
    ELAXISTOS          JOBS.MIN_SALARY%TYPE;
    MEGISTOS           JOBS.MAX_SALARY%TYPE;
    NEOS_MISTHOS       EMPLOYEES.SALARY%TYPE;
BEGIN
    SELECT  JOBS.MIN_SALARY, JOBS.MAX_SALARY
    INTO    ELAXISTOS, MEGISTOS
    FROM    EMPLOYEES, JOBS
    WHERE   EMPLOYEES.JOB_ID = JOBS.JOB_ID
    AND     EMPLOYEES.EMPLOYEE_ID = 1105; /* &EMP_ID */

    NEOS_MISTHOS := (ELAXISTOS+MEGISTOS) /2;

    UPDATE  EMPLOYEES
    SET     SALARY = NEOS_MISTHOS
    WHERE   EMPLOYEE_ID = 1105; /* &EMP_ID */

EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Σφάλμα στον κώδικα: ' ||
                               SQLERRM(SQLCODE) );
        RAISE; /*Η εξαίρεση θα μεταδοθεί έξω από το τρέχον
        τμήμα κώδικα, στον χρηστή η στο σημείο κλήσης*/
END;
```

Η πρόταση WHEN OTHERS THEN υποδηλώνει πως οι ακόλουθες εντολές θα εκτελεστούν σε περίπτωση οποιασδήποτε εξαίρεσης, δηλαδή σε κάθε περίπτωση σφάλματος εκτέλεσης. Αν θέλαμε να εκτελεστούν μόνο για μια συγκεκριμένη εξαίρεση, για παράδειγμα την VALUE\_ERROR, θα έπρεπε να γράψουμε WHEN VALUE\_ERROR THEN. Συνιστάται να συμπεριλαμβάνουμε πάντα τουλάχιστον την περίπτωση WHEN OTHERS THEN διότι έτσι καλύπτουμε όλες τις πιθανές περιπτώσεις σφάλματος.

Η πρόταση DBMS\_OUTPUT.PUT\_LINE('Σφάλμα στον κώδικα: ' || SQLERRM(SQLCODE)) τυπώνει στην οθόνη του τερματικού το μήνυμα "Σφάλμα στον κώδικα: " ακολουθούμενο από το προκαθορισμένο μήνυμα της ORACLE για τη συγκεκριμένη εξαίρεση. Η μεταβλητή SQLCODE και η συνάρτηση SQLERRM είναι προκαθορισμένες, η πρώτη περιέχει τον κωδικό αριθμό του σφάλματος ενώ η δεύτερη επιστρέφει το λεκτικό μήνυμα της ORACLE που αντιστοιχεί σε αυτό. Εναλλακτικά, ο κώδικας θα μπορούσε να καταχωρεί με την εντολή INSERT...INTO, το μήνυμα σφάλματος και την ημερομηνία/ώρα που αυτό συνέβη σε έναν πίνακα (έστω ERROR\_LOG) για λόγους ιστορικού (application error log), για παράδειγμα:

```
EXCEPTION
    WHEN OTHERS THEN
        INSERT INTO ERROR_LOG
        VALUES (SYSDATE, SQLERRM(SQLCODE)) ;
```

Στο παράδειγμα μας, για να ενεργοποιηθεί το τμήμα διαχείρισης εξαιρέσεων πρέπει προκαλέσουμε σκόπιμα ένα λογικό σφάλμα. Αυτό μπορεί να γίνει αλλάζοντας την εντολή

```
SET SALARY = NEOS_MISTHOS
```

σε

```
SET SALARY = NEOS_MISTHOS * 1000000000 /*Η τιμή δεν χωρά*/
```

Αν εκτελέσουμε τώρα τον κώδικα, κατά την ενημέρωση του πεδίου SALARY θα συμβεί σφάλμα εκτέλεσης VALUE\_ERROR, διότι η νέα τιμή είναι μεγάλη και δεν χωρά στο πεδίο. Κατά συνέπεια, θα ενεργοποιηθεί το τμήμα διαχείρισης εξαιρέσεων.

Η εντολή RAISE είναι προαιρετική και αυτό που κάνει είναι να "σημάνει" εκ' νέου το σφάλμα εκτέλεσης ώστε αυτό να μεταδοθεί (propagate) έξω από το τρέχον τμήμα



κώδικα, στον χρήστη ή στο σημείο από το οποίο αυτό κλήθηκε (όπως στην περίπτωση κλήσης μιας διαδικασίας (PROCEDURE) μέσα από μια άλλη διαδικασία). Εάν δεν συμπεριληφθεί η εντολή RAISE, τότε το σφάλμα θεωρείται πως αντιμετωπίστηκε εντός του τρέχοντος τμήματος κώδικα και η εκτέλεση θα συνεχιστεί κανονικά έξω από το τρέχον τμήμα, δηλαδή στο σημείο από το οποίο αυτό κλήθηκε.

Η διαφορά φαίνεται αν εκτελέσουμε το παραπάνω τμήμα κώδικα πρώτα με την εντολή RAISE και μετά χωρίς την εντολή RAISE. Στην πρώτη περίπτωση η ORACLE απαντά με μήνυμα σφάλματος VALUE\_ERROR ενώ στην δεύτερη απαντά "PL/SQL procedure successfully completed" διότι θεωρεί πως το σφάλμα αντιμετωπίστηκε (δεν έγινε εκ' νέου RAISE).

### **Εξαιρέσεις χρήστη (user exceptions)**

Εκτός από τις εξαιρέσεις συστήματος, δηλαδή τις εξαιρέσεις που παράγονται αυτόματα από την ORACLE ως αποτέλεσμα κάποιου σφάλματος εκτέλεσης μέσα σε ένα τμήμα κώδικα, η PL/SQL μας δίνει τη δυνατότητα να δημιουργήσουμε δικές μας εξαιρέσεις, δηλαδή εξαιρέσεις χρήστη (user exceptions).

Οι εξαιρέσεις χρήστη δημιουργούνται σκόπιμα από τον προγραμματιστή με την προκαθορισμένη διαδικασία RAISE\_APPLICATION\_ERROR της ORACLE. Για παράδειγμα, ο προγραμματιστής μπορεί να δημιουργήσει μια τεχνητή εξαίρεση σε κάποιο σημείο του κώδικα αν ανιχνεύσει κάποιον συνδυασμό συνθηκών κάτω από τις οποίες γνωρίζει εκ των προτέρων πως ο κώδικας δεν θα λειτουργήσει σωστά, όπως για παράδειγμα έναν συνδυασμό τιμών πεδίων τις οποίες ο αλγόριθμος δεν μπορεί να χειριστεί. Με τον τρόπο αυτό εξασφαλίζει πως, σε περίπτωση που αυτές οι συνθήκες ισχύσουν, ο κώδικας δεν θα εκτελεστεί, ενώ παράλληλα θα παραχθεί ένα σφάλμα εκτέλεσης με συγκεκριμένη περιγραφή.

Η σύνταξη και χρήση της διαδικασίας RAISE\_APPLICATION\_ERROR σε κάποιο σημείο του κώδικα είναι ως εξής:

```
RAISE_APPLICATION_ERROR (<Κωδικός Σφάλματος>, '<μήνυμα  
λάθους>');
```

όπου <ΚωδικόςΣφάλματος> είναι ένας οποιοσδήποτε αριθμός μεταξύ -20000..-20999 και <μήνυμα λάθους> ένα κείμενο που θα εμφανιστεί ως μήνυμα σφάλματος.

Για παράδειγμα, αν εκτελεστεί η εντολή:

```
RAISE_APPLICATION_ERROR(-20001, 'Συνδυασμός τιμών που δεν υποστηρίζεται. Η εκτέλεση της διαδικασίας θα τερματιστεί.');
```

η ORACLE θα σταματήσει με το μήνυμα σφάλματος

ORA-20001: Συνδυασμός τιμών που δεν υποστηρίζεται. Η εκτέλεση της διαδικασίας θα τερματιστεί.

## Διαδικασίες, συναρτήσεις και πακέτα στην PL/SQL

Τα τμήματα κώδικα που είδαμε σε όλες τις προηγούμενες περιπτώσεις είναι ανώνυμα, δηλαδή δεν εντάσσονται σε κάποια ευρύτερη δομημένη ενότητα. Για τον λόγο αυτόν ονομάζονται *ανώνυμα τμήματα κώδικα PL/SQL (anonymous PL/SQL blocks)*. Τα αποθηκεύουμε σε κάποιο αρχείο κειμένου (συνήθως με κατάληξη .SQL) και τα εκτελούμε με αποκοπή και επικόλληση (copy/paste) μέσα από κάποιο client εργαλείο, όπως η SQL\*Plus ή ο SQLDeveloper.

Η PL/SQL όμως μας δίνει επιπλέον τη δυνατότητα να γράψουμε, όπως και σε άλλες γλώσσες προγραμματισμού, δικά μας επώνυμα τμήματα κώδικα, δηλαδή Διαδικασίες και Συναρτήσεις. Οι διαδικασίες και οι συναρτήσεις στην PL/SQL δημιουργούνται και αποθηκεύονται στον database server και αποτελούν έτσι ονοματισμένα αντικείμενα της ΒΔ, όπως οι πίνακες ή οι όψεις. Για τον λόγο αυτόν ονομάζονται *Αποθηκευμένες Διαδικασίες / Συναρτήσεις (Stored Procedures / Functions)*. Επίσης, πολλές φορές χρησιμοποιείται ο όρος *υποπρογράμματα* για να αναφερθούμε από κοινού στις διαδικασίες και συναρτήσεις (*PL/SQL subprograms*).

Τόσο οι διαδικασίες όσο και οι συναρτήσεις μπορούν να δέχονται ορίσματα (παραμέτρους) ενώ η διαφορά τους είναι πως οι διαδικασίες εκτελούν μια λειτουργία ενώ οι συναρτήσεις συνήθως κάνουν έναν υπολογισμό και επιστρέφουν ένα αποτέλεσμα (μια τιμή).

### Γιατί να χρησιμοποιήσει κανείς αποθηκευμένες διαδικασίες / συναρτήσεις;

- Επέκταση της γλώσσας PL/SQL για τις ανάγκες μας: Οι διαδικασίες συμπεριφέρονται σαν νέες εντολές και οι συναρτήσεις διευκολύνουν τους υπολογισμούς μας.
- Επαναχρησιμοποίηση: Οι αποθηκευμένες διαδικασίες και συναρτήσεις μπορούν να κληθούν από οποιοδήποτε περιβάλλον προγραμματισμού το οποίο υποστηρίζει τη χρήση της ORACLE καθώς και από διαφορετικές εφαρμογές.
- Ευκολία συντήρησης του κώδικα: Οι αλλαγές ή βελτιώσεις σε μια διαδικασία ή συνάρτηση δεν επηρεάζουν τις εφαρμογές που την χρησιμοποιούν, εφόσον δεν αλλάζει ο τρόπος κλήσης (επικεφαλίδα της διαδικασίας / συνάρτησης).
- Απόκρυψη λεπτομερών υλοποίησης των εργασιών: Ένας προγραμματιστής μπορεί να χρησιμοποιήσει μια υπάρχουσα αποθηκευμένη διαδικασία / συνάρτηση για να εκτελέσει μια εργασία ή έναν υπολογισμό χωρίς να τον ενδιαφέρει ο τρόπος υλοποίησης της. Το μόνο που χρειάζεται να γνωρίζει είναι τι κάνει και ποιες παραμέτρους δέχεται (αν υπάρχουν).
- Καλύτερη οργάνωση του κώδικα: Όπως θα δούμε στη συνέχεια, οι διαδικασίες και συναρτήσεις, μαζί με άλλες πιθανές απαραίτητες δηλώσεις, μπορούν να οργανωθούν σε ευρύτερες ενότητες (βιβλιοθήκες), τα Πακέτα PL/SQL (PL/SQL Packages).

### Σύνταξη - δημιουργία Διαδικασιών / Συναρτήσεων

Η δημιουργία μιας αποθηκευμένης διαδικασίας PL/SQL ακολουθεί την εξής γενική σύνταξη:

```
CREATE OR REPLACE PROCEDURE <όνομα_διαδικασίας> [<παραμέτροι>]  
<Δηλώσεις μεταβλητών/σταθερών...>  
IS  
BEGIN  
    <εντολές>  
    ...
```

```

        [COMMIT;]
    EXCEPTION
        when <exception_name> then
            <εντολές>
    END <όνομα_διαδικασίας>;

```

Μια αποθηκευμένη συνάρτηση συντάσσεται με τον ίδιο ακριβώς τρόπο, εκτός από την επικεφαλίδα στην οποία μπορούμε επιπλέον να δηλώσουμε μια επιστρεφόμενη τιμή, ενώ συνήθως στο τέλος συμπεριλαμβάνουμε την εντολή RETURN <τιμή> ώστε να επιστρέψουμε το αποτέλεσμα της συνάρτησης:

```

CREATE OR REPLACE FUNCTION <όνομα_συνάρτησης> [<παράμετροι>]
RETURN <τύπος_επιστρεφόμενης_τιμής>
<Δηλώσεις μεταβλητών/σταθερών...>
IS
BEGIN
    <εντολές>
    ...
    RETURN <τιμή_αποτελέσματος>
EXCEPTION
    when <exception_name> then
        <εντολές>
END < όνομα_συνάρτησης >;

```

Το τμήμα δηλώσεων (πριν την πρόταση IS BEGIN...), το τμήμα εκτελέσιμων εντολών (BEGIN...END), το τμήμα διαχείρισης εξαιρέσεων και γενικά όλη η δομή μιας διαδικασίας ή συνάρτησης, πλην της επικεφαλίδας, είναι ίδια με τη δομή των ανώνυμων τμημάτων κώδικα PL/SQL που είδαμε νωρίτερα.

Η εντολή RETURN <τιμή\_αποτελέσματος> σε μια συνάρτηση PL/SQL τερματίζει την εκτέλεση της συνάρτησης και επιστρέφει ως αποτέλεσμα την τιμή που ακολουθεί.

Η δήλωση των παραμέτρων γίνεται στην επικεφαλίδα της διαδικασίας / συνάρτησης και ακολουθεί τον τρόπο σύνταξης που χρησιμοποιούμε και για τη δήλωση μεταβλητών. Επιπλέον, μια παράμετρος μπορεί να δηλωθεί ως τύπου IN, OUT ή IN OUT.

- **IN** : Η παράμετρος δέχεται μια τιμή η οποία δεν μπορεί να αλλάξει, δηλαδή εντός της διαδικασίας / συνάρτησης η παράμετρος συμπεριφέρεται σαν μια σταθερά. Μια παράμετρος, εάν δεν προσδιοριστεί διαφορετικά, είναι εξ'

ορισμού τύπου IN. Χρησιμοποιείται για τη μεταβίβαση μιας τιμής από τον καλούντα προς το υποπρόγραμμα.

- **OUT** : Η παράμετρος χρησιμοποιείται για τη μεταβίβαση μιας τιμής από τη διαδικασία / συνάρτηση προς τον καλούντα, δηλαδή για την επιστροφή μιας τιμής. Στην θέση αυτής της παραμέτρου πρέπει να περάσουμε μια μεταβλητή, δηλαδή δεν μπορεί να κληθεί το υποπρόγραμμα με μια σταθερή τιμή ή μια παράσταση στη θέση της παραμέτρου αυτής. Η παράμετρος αυτή εντός της διαδικασίας / συνάρτησης συμπεριφέρεται σαν μια μεταβλητή η οποία αρχικά δεν έχει τιμή αλλά παίρνει μια τιμή από τη διαδικασία / συνάρτηση η οποία μεταβιβάζεται εκτός προς τον καλούντα.
- **IN OUT** : Η παράμετρος χρησιμοποιείται για τη μεταβίβαση μιας τιμής από τον καλούντα προς τη διαδικασία / συνάρτηση και στην συνέχεια για την επιστροφή μιας τιμής από τη διαδικασία / συνάρτηση στον καλούντα. Όπως και στην προηγούμενη περίπτωση, στην θέση αυτής της παραμέτρου πρέπει να περάσουμε μια μεταβλητή.

### Κλήση μιας διαδικασίας / συνάρτησης PL/SQL

Μια αποθηκευμένη διαδικασία / συνάρτηση μπορεί να κληθεί με τη χρήση των κατάλληλων εντολών SQL. Ο τρόπος κλήσης είναι διαφορετικός για διαδικασία ή συνάρτηση:

- Αν είναι διαδικασία, την καλούμε με μια από τις εντολές:
  - EXEC <όνομα\_διαδικασίας> (<παράμετροι>); ή
  - BEGIN <όνομα\_διαδικασίας> (<παράμετροι>); END;
- Αν είναι συνάρτηση, δηλαδή επιστρέφει μια τιμή, πρέπει να χρησιμοποιήσουμε τη συνάρτηση ως τμήμα μιας παράστασης. Ο πιο απλός τρόπος για να τη δοκιμάσουμε είναι η εντολή SELECT...FROM DUAL:
  - SELECT <όνομα\_συνάρτησης> (<παράμετροι>) FROM DUAL;

### Παράδειγμα αποθηκευμένης διαδικασίας

Για να κατανοήσουμε καλύτερα τον τρόπο δημιουργίας και χρήσης μιας διαδικασίας, ας δούμε την ακόλουθη άσκηση:

Έστω πως θέλουμε να υλοποιήσουμε την αποθηκευμένη διαδικασία ReplaceBonus με τις εξής παραμέτρους:

- EMP\_ID : ο κωδικός ενός υπαλλήλου
- NEW\_BONUS : το νέο bonus του υπαλλήλου

η οποία θα πρέπει να εκτελεί τις εξής λειτουργίες και μόνο:

- Να διαγράφει από τον πίνακα BONUS όσες εγγραφές αφορούν τον συγκεκριμένο υπάλληλο (αν υπάρχουν).
- Να εισάγει μια νέα εγγραφή στον πίνακα BONUS για τον υπάλληλο με τιμές (EMP\_ID, NEW\_BONUS).
- Να οριστικοποιεί τις παραπάνω μεταβολές.

Η πλήρης δήλωση και υλοποίηση της διαδικασίας φαίνεται παρακάτω:

```
CREATE OR REPLACE PROCEDURE REPLACE_BONUS (EMP_ID IN NUMBER,
NEW_BONUS BONUS.POSO_BONUS%TYPE )
IS
BEGIN
    DELETE FROM BONUS
    WHERE EMPLOYEE_ID = EMP_ID;

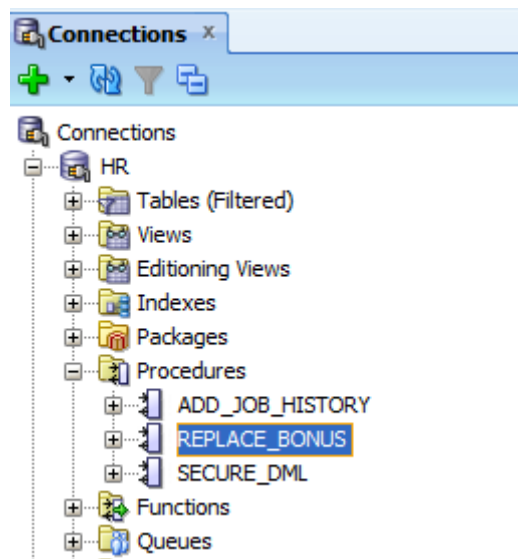
    INSERT INTO BONUS
    VALUES (EMP_ID, NEW_BONUS);

    COMMIT;
END REPLACE_BONUS;
```

Όπως φαίνεται στην επικεφαλίδα της διαδικασίας, η παράμετρος EMP\_ID είναι τύπου IN, δηλαδή μεταβιβάζει μια τιμή από τον καλούντα προς τη διαδικασία. Ομοίως η παράμετρος NEW\_BONUS, εφόσον δεν προσδιορίζεται διαφορετικά, είναι τύπου IN εξ' ορισμού. Μάλιστα, η παράμετρος αυτή δηλώνεται ως τύπου ίδιου με το πεδίο POSO\_BONUS του πίνακα BONUS (BONUS.POSO\_BONUS%TYPE). Οι δυο αυτές παράμετροι χρησιμοποιούνται ακολούθως εντός της διαδικασίας στις

εντολές PL/SQL DELETE και INSERT σαν να ήταν τοπικές τιμές. Τελευταία είναι η εντολή COMMIT η οποία οριστικοποιεί στη ΒΔ τις αλλαγές.

Αν εκτελέσουμε το παραπάνω τμήμα κώδικα στο παράθυρο της SQL η ORACLE αποκρίνεται με το μήνυμα πως η διαδικασία δημιουργήθηκε επιτυχώς. Η διαδικασία αυτή αποτελεί πλέον αντικείμενο της ΒΔ και μπορούμε να τη δούμε κάτω από την ενότητα PROCEDURES του SQLDeveloper:



Εικόνα 6-1 : Ενότητα PROCEDURES του SQLDeveloper

Όπως είδαμε στην προηγούμενη παράγραφο, για την κλήση της διαδικασίας μπορούμε να χρησιμοποιήσουμε την εντολή EXEC: Για παράδειγμα, η κλήση της διαδικασίας για τον υπάλληλο με id=201 και ποσό bonus 780 γίνεται με την εντολή

```
EXEC REPLACE_BONUS (201,780) ;
```

και η απόκριση της ORACLE είναι:

```
anonymous block completed (SQLDeveloper) ή  
PL/SQL procedure successfully completed. (SQL*Plus)
```

### Παράδειγμα αποθηκευμένης συνάρτησης

Ας δούμε τώρα και μια περίπτωση αποθηκευμένης συνάρτησης. Έστω πως θέλουμε να υλοποιήσουμε την αποθηκευμένη συνάρτηση με όνομα CHECK\_AFM η οποία να δέχεται ως παράμετρο έναν Α.Φ.Μ. και να επιστρέφει 0 αν είναι ο Α.Φ.Μ. είναι έγκυρος, διαφορετικά να επιστρέφει 1.

Δίδεται ο αλγόριθμος ελέγχου ορθότητας Α.Φ.Μ.

1. Υπολογισμός αθροίσματος γινομένων των 8 πρώτων ψηφίων του ΑΦΜ με 256, 128, 64, 32, 16, 8, 4 και 2 αντίστοιχα.
2. Το υπόλοιπο της διαίρεσης του αθροίσματος με το 11 πρέπει να είναι ίσο με το τελευταίο ψηφίο του ΑΦΜ (εάν το υπόλοιπο είναι 10 τότε το θεωρούμε ίσο με 0).

Για παράδειγμα, είναι ο Α.Φ.Μ. 094222211 έγκυρος;

Έχουμε  $\Sigma = 0 \times 256 + 9 \times 128 + 4 \times 64 + 2 \times 32 + 2 \times 16 + 2 \times 8 + 2 \times 4 + 1 \times 2 = 1530$ . Το υπόλοιπο της διαίρεσης του 1530 με το 11 είναι ίσο με 1 το οποίο είναι ίσο με το τελευταίο (9ο) ψηφίο του Α.Φ.Μ., επομένως ο παραπάνω Α.Φ.Μ. είναι έγκυρος.

Η υλοποίηση της παραπάνω συνάρτησης φαίνεται στο ακόλουθο τμήμα κώδικα:

```
CREATE OR REPLACE FUNCTION CHECK_AFM (AFM_VAL IN VARCHAR2)
RETURN NUMBER
IS
    S NUMBER;
    K NUMBER;
BEGIN
    IF (LENGTH (AFM_VAL) <> 9) THEN
        RETURN 1;
    ELSE
        S := 0;
        K := 256;

        FOR N IN 1..8 LOOP
            IF (SUBSTR (AFM_VAL, N, 1) < '0'
                OR SUBSTR (AFM_VAL, N, 1) > '9') THEN
                RETURN 1;
            ELSE
                S := S + K * TO_NUMBER (SUBSTR (AFM_VAL, N, 1));
            END IF;

            K := K / 2;
        END LOOP;

        RETURN (S % 11 = SUBSTR (AFM_VAL, 9, 1) - '0') ? 0 : 1;
    END IF;
END;
```



```
END LOOP;

S := MOD(S,11);

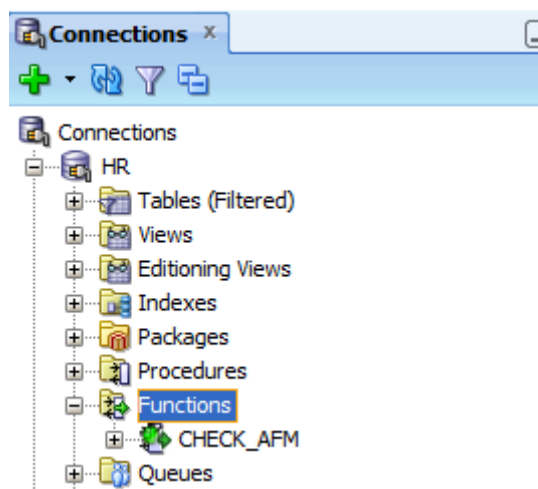
IF (S=10) THEN
    S := 0;
END IF;

IF (S <> TO_NUMBER(SUBSTR(AFM_VAL,9,1))) THEN
    RETURN 1;
ELSE
    RETURN 0;
END IF;

END IF;
END CHECK_AFM;
```

Αρχικά, η πρόταση IF (LENGTH(AFM\_VAL) <> 9) ελέγχει εάν το μήκος του Α.Φ.Μ. είναι διαφορετικό από 9 χαρακτήρες, στην οποία περίπτωση είναι προφανώς λάθος, και, εάν είναι, επιστρέφει την τιμή 1, δηλαδή λάθος Α.Φ.Μ. (RETURN 1). Διαφορετικά, αρχικοποιούνται οι μεταβλητές S, K οι οποίες περιέχουν το άθροισμα των γινομένων και τον τρέχοντα πολλαπλασιαστή αντίστοιχα, σύμφωνα με τον παραπάνω αλγόριθμο. Η πρόταση FOR N IN 1..8 LOOP διατρέχει ένα προς ένα τα ψηφία του Α.Φ.Μ. από αριστερά προς δεξιά. Αν ο ν-οστός χαρακτήρας (SUBSTR(AFM\_VAL,N,1)) δεν είναι ψηφίο (είναι <'0' ή >'9') τότε προφανώς ο Α.Φ.Μ. είναι λάθος (RETURN 1), διαφορετικά ενημερώνονται το μεν άθροισμα με την πρόσθεση του γινομένου του ν-οστού ψηφίου με τον τρέχοντα πολλαπλασιαστή (S:= S + K\*TO\_NUMBER(SUBSTR(AFM\_VAL,N,1))), ο δε πολλαπλασιαστής διαιρείται με το 2 για το επόμενο βήμα (K:= K/2, από 256 γίνεται 128, μετά 64 κοκ). Όταν ολοκληρωθεί η επανάληψη FOR...LOOP, η μεταβλητή S παίρνει την τιμή του υπολοίπου της διαίρεσης του αθροίσματος με το 11 (S:= MOD(S,11)) και, εάν είναι 10, γίνεται ίσο με το 0. Τέλος, η τιμή αυτή συγκρίνεται με το τελευταίο (9ο) ψηφίο του Α.Φ.Μ. ώστε να ελεγχθεί η εγκυρότητα του (S<> TO\_NUMBER(SUBSTR(AFM\_VAL,9,1))).

Αν εκτελέσουμε το παραπάνω τμήμα κώδικα στο παράθυρο της SQL η ORACLE αποκρίνεται με το μήνυμα πως η συνάρτηση δημιουργήθηκε επιτυχώς. Η συνάρτηση αυτή αποτελεί πλέον αντικείμενο της ΒΔ και μπορούμε να τη δούμε κάτω από την ενότητα FUNCTIONS του SQLDeveloper:



Εικόνα 6-2 : Ενότητα FUNCTIONS του SQLDeveloper

Όπως είδαμε προηγούμενα, η αποθηκευμένη συνάρτηση που δημιουργήσαμε μπορεί να χρησιμοποιηθεί πλέον ως τμήμα μιας παράστασης, όπως μιας εντολής SELECT:

```
SELECT CHECK_AFM('099793476') FROM DUAL;
```

```
CHECK_AFM('099793476')
```

```
-----
```

```
1
```

### Πακέτα στην PL/SQL

Οι αποθηκευμένες διαδικασίες και συναρτήσεις μπορούν να ομαδοποιηθούν (ενταχθούν) σε ευρύτερες δομικές μονάδες, τα *πακέτα PL/SQL (PL/SQL Packages)*. Ένα πακέτο PL/SQL είναι μια συλλογή από, συνήθως λογικά συσχετιζόμενες, διαδικασίες, συναρτήσεις ή/και άλλες δηλώσεις (μεταβλητών, σταθερών), είναι δηλαδή κάτι ανάλογο με τις *βιβλιοθήκες (libraries)* που συναντούμε σε άλλα προγραμματιστικά περιβάλλοντα.

Ένα πακέτο μπορεί να περιέχει πολλές διαδικασίες / συναρτήσεις, καθώς και άλλες δηλώσεις, δημόσιες (προσπελάσιμες από τον χρήστη του πακέτου) ή ιδιωτικές (προς χρήση μόνο εντός του συγκεκριμένου πακέτου). Τα πακέτα αποτελούν και αυτά με τη σειρά τους ονοματισμένα αντικείμενα που αποθηκεύονται ως μέρος της ΒΔ, όπως οι διαδικασίες και συναρτήσεις, και για τον λόγο αυτόν ονομάζονται *Αποθηκευμένα Πακέτα PL/SQL (PL/SQL Stored Packages)*. Όταν μια διαδικασία ή συνάρτηση βρίσκεται εντός κάποιου πακέτου, η ταυτότητα της αποτελείται από το όνομα του

πακέτου και το όνομα της διαδικασίας / συνάρτησης. Για παράδειγμα, αν δημιουργήσουμε το πακέτο MY\_PACKAGE και μέσα σε αυτό την συνάρτηση CHECK\_AFM, τότε αναφερόμαστε στην συνάρτηση με τον συνδυασμό MY\_PACKAGE.CHECK\_AFM.

Παραδείγματα πακέτων είναι μια συλλογή από χρηματοοικονομικές συναρτήσεις και σταθερές, μια συλλογή διαδικασιών για τον υπολογισμό μισθοδοσίας, μια συλλογή διαδικασιών για μαθηματικές πράξεις μεταξύ πινάκων κλπ.

### Σύνταξη και δημιουργία πακέτων στην PL/SQL

Για συντακτικούς λόγους, η δομή ενός πακέτου στην PL/SQL αποτελείται από δυο τμήματα, το τμήμα δηλώσεων (PACKAGE) και το σώμα (PACKAGE BODY). Στο πρώτο (PACKAGE) δηλώνουμε τις επικεφαλίδες των διαδικασιών / συναρτήσεων (όχι το σώμα τους), τις δημόσιες σταθερές και γενικά οτιδήποτε θέλουμε να είναι ορατό (προσπελάσιμο) από τον χρήστη του πακέτου. Στο σώμα του πακέτου (PACKAGE BODY) γράφουμε τις πλήρεις υλοποιήσεις των διαδικασιών / συναρτήσεων και πιθανά πρόσθετες ιδιωτικές διαδικασίες / συναρτήσεις ή άλλες δηλώσεις που χρησιμοποιούνται μόνο για τις ανάγκες του πακέτου και στις οποίες ο χρήστης του πακέτου δεν έχει πρόσβαση ούτε γνωρίζει την ύπαρξη τους.

Για να χρησιμοποιήσει ένας χρήστης ένα πακέτο αρκεί να γνωρίζει μόνο το τμήμα δηλώσεων (PACKAGE), δηλαδή τις επικεφαλίδες των διαδικασιών / συναρτήσεων ή άλλες δηλώσεις που περιέχει, ενώ δεν τον ενδιαφέρει καθόλου το σώμα (PACKAGE BODY), δηλαδή ο τρόπος υλοποίησης (αποτελεί για αυτόν ένα "μαύρο κουτί"). Για τον λόγο αυτόν, το τμήμα δηλώσεων ονομάζεται και *Τμήμα Προδιαγραφών του πακέτου* (PACKAGE SPECIFICATION).

Η γενική σύνταξη για τη δημιουργία ενός πακέτου PL/SQL φαίνεται παρακάτω:

#### **ΤΜΗΜΑ ΔΗΛΩΣΕΩΝ**

```
CREATE OR REPLACE PACKAGE <ΟΝΟΜΑ_ΠΑΚΕΤΟΥ> AS
```

```
    <ΔΗΛΩΣΕΙΣ ΔΗΜΟΣΙΩΝ ΣΤΑΘΕΡΩΝ / ΜΕΤΑΒΛΗΤΩΝ>  
    <ΔΗΛΩΣΕΙΣ ΔΗΜΟΣΙΩΝ ΔΙΑΔΙΚΑΣΙΩΝ / ΣΥΝΑΡΤΗΣΕΩΝ, ΜΟΝΟ Η  
    ΕΠΙΚΕΦΑΛΙΔΑ>
```

```
END <ΟΝΟΜΑ_ΠΑΚΕΤΟΥ>;
```

/

**ΣΩΜΑ ΠΑΚΕΤΟΥ**

```

CREATE OR REPLACE PACKAGE BODY <ΟΝΟΜΑ_ΠΑΚΕΤΟΥ> AS

    <ΔΗΛΩΣΕΙΣ ΙΔΙΩΤΙΚΩΝ ΣΤΑΘΕΡΩΝ / ΜΕΤΑΒΛΗΤΩΝ>
    <ΥΛΟΠΟΙΗΣΗ ΔΗΜΟΣΙΩΝ ΚΑΙ ΙΔΙΩΤΙΚΩΝ ΔΙΑΔΙΚΑΣΙΩΝ/ΣΥΝΑΡΤΗΣΕΩΝ >

END <ΟΝΟΜΑ_ΠΑΚΕΤΟΥ> ;
/

```

Για την καλύτερη κατανόηση του τρόπου δημιουργίας και χρήσης ενός πακέτου PL/SQL στην ORACLE, ας υποθέσουμε πως θέλουμε δημιουργήσουμε ένα πακέτο με όνομα MY\_PACKAGE το οποίο να περιέχει τη συνάρτηση CHECK\_AFM που είδαμε στο προηγούμενο παράδειγμα. Προφανώς, η αποθηκευμένη συνάρτηση του προηγούμενου παραδείγματος είναι τώρα περιττή εφόσον πλέον θα υπάρχει σαν μέρος ενός πακέτου. Ο πλήρης κώδικας για τη δημιουργία του πακέτου φαίνεται παρακάτω:

```

CREATE OR REPLACE PACKAGE MY_PACKAGE AS

    FUNCTION CHECK_AFM(AFM_VAL IN VARCHAR2) RETURN NUMBER;

END MY_PACKAGE;
/

CREATE OR REPLACE PACKAGE BODY MY_PACKAGE AS

FUNCTION CHECK_AFM(AFM_VAL IN VARCHAR2) RETURN NUMBER
IS
    S NUMBER;
    K NUMBER;
BEGIN
    IF (LENGTH(AFM_VAL) <> 9) THEN
        RETURN 1;
    ELSE
        S := 0;
        K := 256;

        FOR N IN 1..8 LOOP
            IF (SUBSTR(AFM_VAL,N,1) <'0'
                OR SUBSTR(AFM_VAL,N,1) >'9') THEN
                RETURN 1;
            ELSE
                S := S + K*TO_NUMBER(SUBSTR(AFM_VAL,N,1));
            END IF;
        END LOOP;
    END IF;
END CHECK_AFM;

```

```
K := K/2;

END LOOP;

S := MOD(S,11);

IF (S=10) THEN
    S := 0;
END IF;

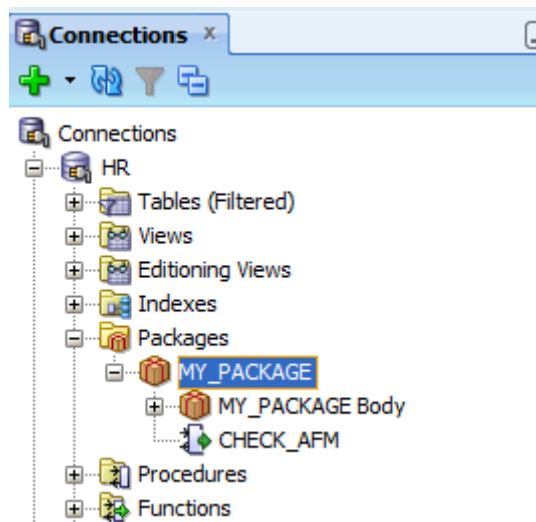
IF (S <> TO_NUMBER(SUBSTR(AFM_VAL,9,1))) THEN
    RETURN 1;
ELSE
    RETURN 0;
END IF;

END IF;
END CHECK_AFM;

END MY_PACKAGE;
/
```

Το πρώτο τμήμα κώδικα δημιουργεί το τμήμα δηλώσεων του πακέτου MY\_PACKAGE το οποίο βλέπουμε πως περιέχει μόνο τη δήλωση (δηλαδή την επικεφαλίδα) της συνάρτησης CHECK\_AFM. Το δεύτερο τμήμα δημιουργεί το σώμα του πακέτου (PACKAGE BODY) το οποίο περιέχει την πλήρη υλοποίηση της συνάρτησης CHECK\_AFM.

Μετά την εκτέλεση των παραπάνω τμημάτων κώδικα στο παράθυρο της SQL, η ORACLE αποκρίνεται με τα μηνύματα "Package created" και "Package body created" αντίστοιχα, μας ενημερώνει δηλαδή πως η δημιουργία του πακέτου ολοκληρώθηκε επιτυχώς. Το πακέτο αποτελεί πλέον αντικείμενο της ΒΔ και μπορούμε να το δούμε κάτω από την ενότητα PACKAGES του SQLDeveloper:



Εικόνα 6-3 : Ενότητα PACKAGES του SQLDeveloper

### Χρήση διαδικασίας / συνάρτησης ενός πακέτου

Ο τρόπος χρήσης μιας διαδικασίας ή συνάρτησης ενός αποθηκευμένου πακέτου είναι ο ίδιος με μιας κανονικής αποθηκευμένης διαδικασίας / συνάρτησης, απλά προσθέτουμε μπροστά από το όνομα της το όνομα του πακέτου και μια "." (τελεία). Έτσι, για να ελέγξουμε την ορθότητα του Α.Φ.Μ. "099793475" δίνουμε την εντολή:

```
SELECT MY_PACKAGE.CHECK_AFM('099793475') FROM DUAL;
```

```
MY_PACKAGE.CHECK_AFM('099793475')
```

0

## 7. ΘΕΜΑΤΑ ΔΙΑΧΕΙΡΙΣΗΣ- ΑΣΦΑΛΕΙΑ

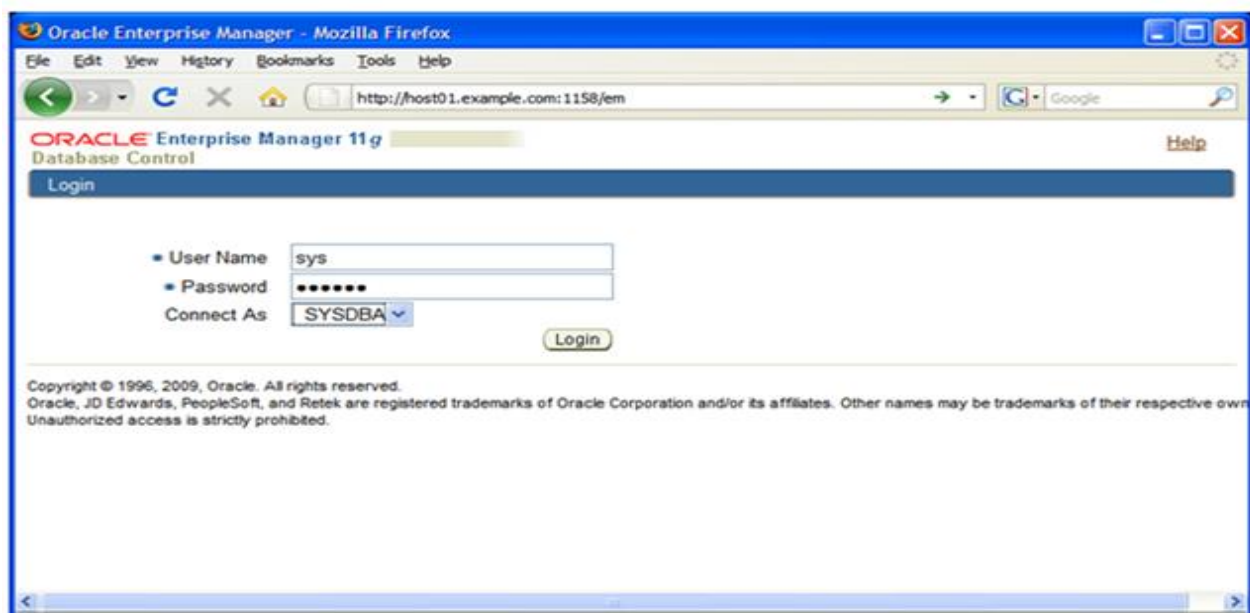
### Ο Oracle Enterprise Manager.

**Ο Oracle Enterprise Manager (EM).** Είναι μια γραφική εφαρμογή διαδικτύου (web-based) με την οποία γίνεται η διαχείριση μιας Oracle Database. Εγκαθίσταται με το λογισμικό της Oracle, και δίδει την δυνατότητα στο Διαχειριστή μιας Oracle Database (DBA), να έχει πρόσβαση σε όλες τις λειτουργίες αυτής. Η διαχείριση της Oracle Database μπορεί να γίνει και με εντολές (commands) όπως και παλαιότερα στις εκδόσεις μέχρι και την 9i. Όμως από την έκδοση 10g και μετά είναι προτιμότερο και πιο εύκολο η διαχείριση να γίνεται με τον Oracle Enterprise Manager.

Οι λειτουργίες του Oracle Enterprise Manager είναι:

- Δημιουργία και διαχείριση όλων των αντικειμένων (objects) της Βάσης Δεδομένων.
- Πλήθος αναφορών (summaries) για όλες τις λειτουργίες της Βάσης Δεδομένων.
- Προειδοποιητικά μηνύματα (alerts) σε περίπτωση κακής λειτουργίας μια διεργασίας.
- Γραφική παρακολούθηση της καλής λειτουργίας και απόδοσης (performance) όλων των αντικειμένων (objects) και διεργασιών της Βάσης Δεδομένων.
- Λήψη αντιγράφων ασφαλείας (back up).
- Αποκατάσταση (recovery) της Βάσης Δεδομένων σε περίπτωση κάποιας αστοχίας.

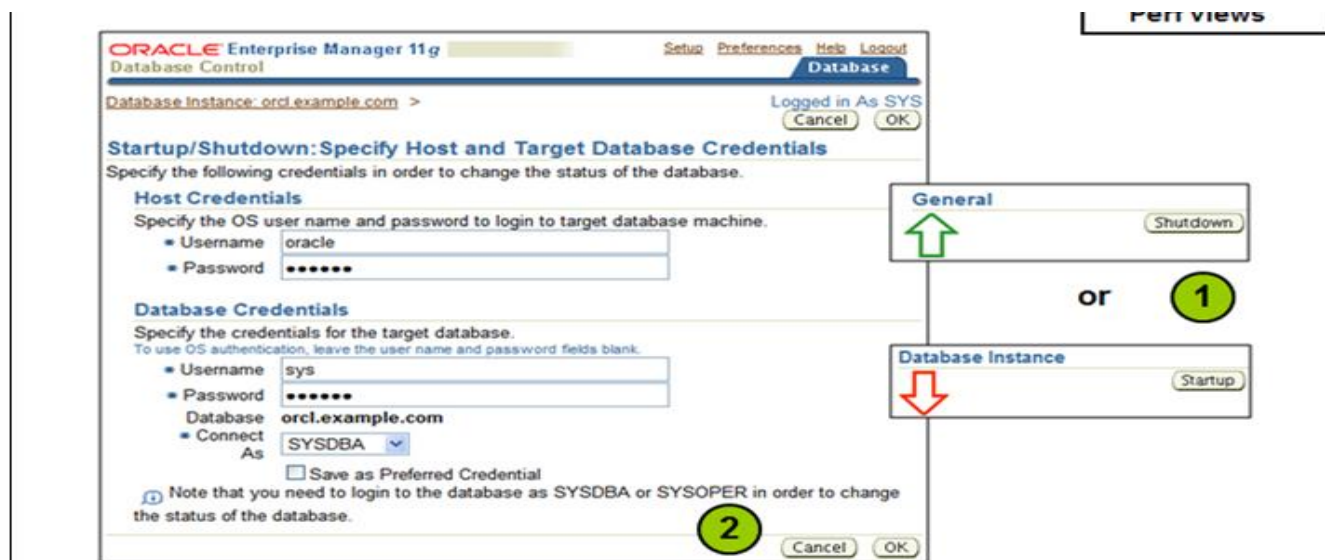
Η εκκίνηση του Oracle Enterprise Manager γίνεται δίνοντας σε ένα φυλλομετρητή (Web browser) την διεύθυνση <http://host name:port number /em> και εμφανίζεται η παρακάτω οθόνη:



ΣΧΗΜΑ 7- 1 Εισαγωγή στον Enterprise Manager

Για να εισέλθουμε στην Βάση Δεδομένων (Login) δίνουμε αρχικά έναν από τους χρήστες (SYS, SYSMAN ή SYSTEM) και τον κωδικό ασφαλείας που ορίσαμε στην εγκατάσταση της Oracle.

### Εκκίνηση και Τερματισμός της Oracle Database



ΣΧΗΜΑ 7- 2 Οθόνη εκκίνησης και τερματισμού της Oracle Database

Η εκκίνηση (startup) και ο τερματισμός (shutdown) της Oracle Database, γίνεται από τον EM από την οθόνη της εικόνας, που φαίνεται στο Σχήμα 7-2.



Και στις δύο περιπτώσεις θα ζητηθούν δικαιώματα (privileges) του χρήστη SYSDBA. Επίσης η οθόνη αυτή παρέχει τη δυνατότητα παρακολούθησης των SQL εντολών της εκκίνησης ή του τερματισμού της Oracle Database.

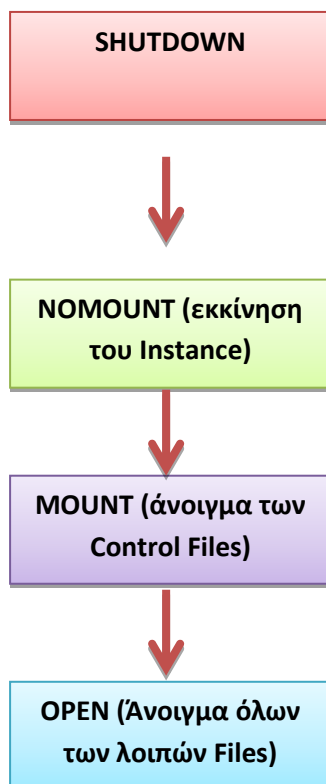
### Στάδια εκκίνησης της Oracle Database.

Η Oracle Database λειτουργεί κανονικά και επιτρέπει στους εξουσιοδοτημένους χρήστες να έχουν πρόσβαση σε αυτή, όταν:

- Το instance έχει εκκινήσει κανονικά.
- Η Βάση έχει φορτωθεί (mount) και έχει ανοίξει (open) κανονικά.

Αποτέλεσμα των παραπάνω, είναι τα datafiles και τα redo log files να λειτουργούν κανονικά.

Τα στάδια της κανονικής εκκίνησης είναι:



ΣΧΗΜΑ 7- 3 Στάδια Εκκίνησης της Oracle Database

### Τύποι (modes) τερματισμού της Oracle Database.

- **Abort:** Εκτελεί τις ελάχιστες λειτουργίες τερματισμού της Βάσης Δεδομένων. Χρησιμοποιείται μόνον σε έκτακτες ανάγκες, όταν κανένας άλλος τύπος τερματισμού δεν μπορεί να εφαρμοστεί.
- **Immediate:** Αποτελεί μια τυπική διαδικασία τερματισμού της Βάσης Δεδομένων. Διεργασίες που δεν έχουν καταγραφεί στην Βάση (Uncommitted transactions) απορρίπτονται (rollback).
- **Transactional:** Επιτρέπει τις διεργασίες (transactions) να τερματίσουν κανονικά.
- **Normal:** Επιτρέπει όλες λειτουργίες της Βάσης να τερματίσουν κανονικά.

### Διαχείριση των Δομών Δεδομένων (Storage Structures).



ΣΧΗΜΑ 7- 4 Διαχείριση των Δομών Δεδομένων (Storage Structures)

Η διαχείριση των Δομών Δεδομένων μιας Oracle Database, γίνεται από τον EM από την οθόνη του **Σχήματος 7-4**.

Η οθόνη περιέχει τους απαραίτητους υπερσυνδέσμους (links) προς όλες τις δομές αποθήκευσης όπως Control files, Datafiles, Tablespaces, Redo Log files κλπ.

### Δημιουργία των Tablespaces.

The screenshot shows the 'Create Tablespace' dialog box with the following settings:

- Name:** INVENTORY
- Extent Management:** Locally Managed (selected), Dictionary Managed (unselected)
- Type:** Permanent (selected), Set as default permanent tablespace (unchecked), Encryption (unchecked), Temporary (unselected), Set as default temporary tablespace (unchecked), Undo (unselected), Undo Retention Guarantee: No (selected)
- Status:** Read Write (selected), Read Only (unselected), Offline (unselected)
- Datafiles:** Use bigfile tablespace (unchecked)
- Datafiles Table:** A table with columns 'Select Name', 'Directory', and 'Size (MB)' is shown at the bottom, currently empty with the text 'No items found'.

ΣΧΗΜΑ 7-5 Δημιουργία των Tablespaces

Η δημιουργία των Tablespaces γίνεται από την οθόνη του **Σχήματος 7-5**.

Εισάγονται οι παράμετροι ενός Tablespace:

**Name:**

**Extent Management:** Locally Managed,

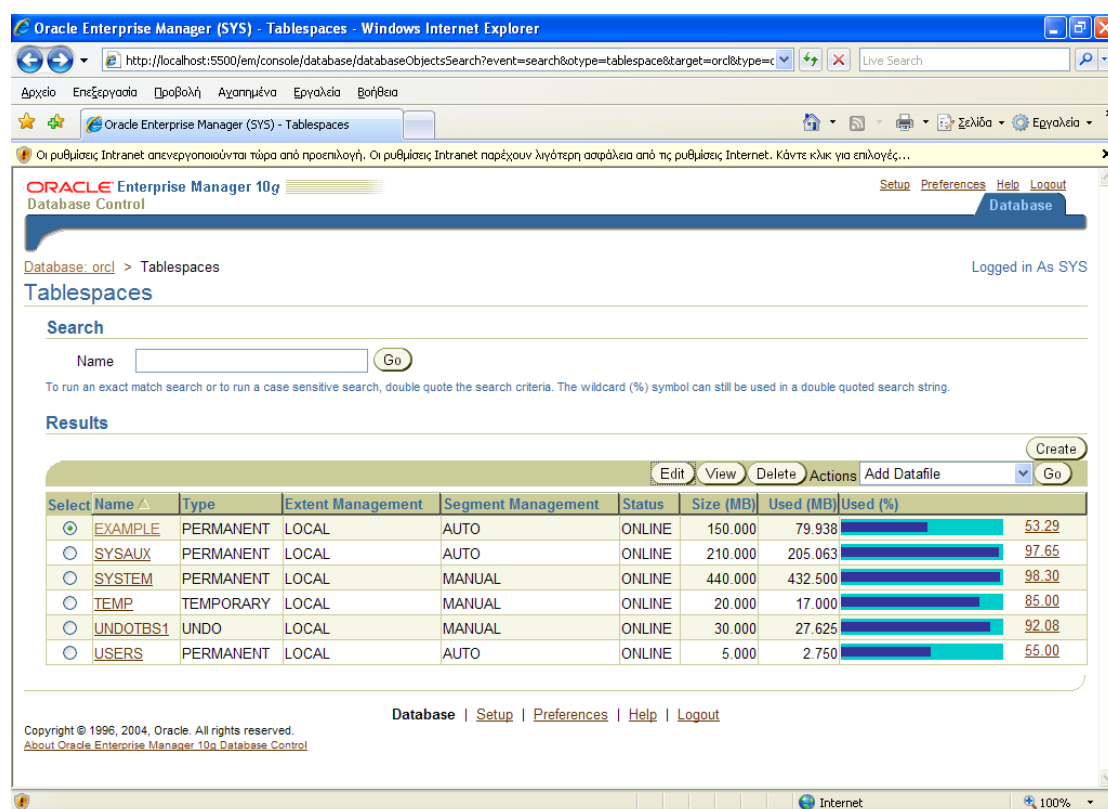
**Type heading:** Permanent.

**Status heading:** Read Write.

Στην περιοχή **Datafiles:** Συμπληρώνονται τα σχετικά στοιχεία

**Size:**

**Storage region:** AUTOEXTEND

**Extent Allocation** (Automatic, Uniform)Επιπρόσθετα και οι **Compression Options, Logging, Block Information****Διαχείριση των Tablespaces****ΣΧΗΜΑ 7-6** Διαχείριση των Tablespaces

Η Διαχείριση των Tablespaces γίνεται από τις οθόνες του **Σχήματος 7-6**.

Οι ενέργειες που μπορεί να γίνουν σε ένα Tablespace είναι:

Πρόσθεση data file, Δημιουργία από πρότυπο (Create Like), Δημιουργία DDL, Make Locally Managed, Make Readonly, Make Writable, Place Online, Reorganize, Run Segment Advisor, Show Dependencies, Show Tablespace Contents, Take Offline.

**Ασφάλεια – Διαχείριση χρηστών (Database users).****Δημιουργία / διαγραφή χρηστών**

Η πρόσβαση στην Oracle Database γίνεται από εξουσιοδοτημένους χρήστες.

Τα στοιχεία τα οποία απαιτούνται για τον ορισμό ενός χρήστη είναι:

Όνομα χρήστη,  
Κωδικός χρήστη,  
Tablespace χρήστη (default),  
Προσωρινό tablespace (temporary),  
Χαρακτηριστικά χρήστη (user profile),  
Ομάδα χρήστη

**Κατάσταση χρήστη (status)** όπως δικαιώματα ανοίγματος (open), απενεργοποιημένος (locked) ή χρήστης που η χρονική περίοδο της πρόσβασης του στην Database έχει λήξει.

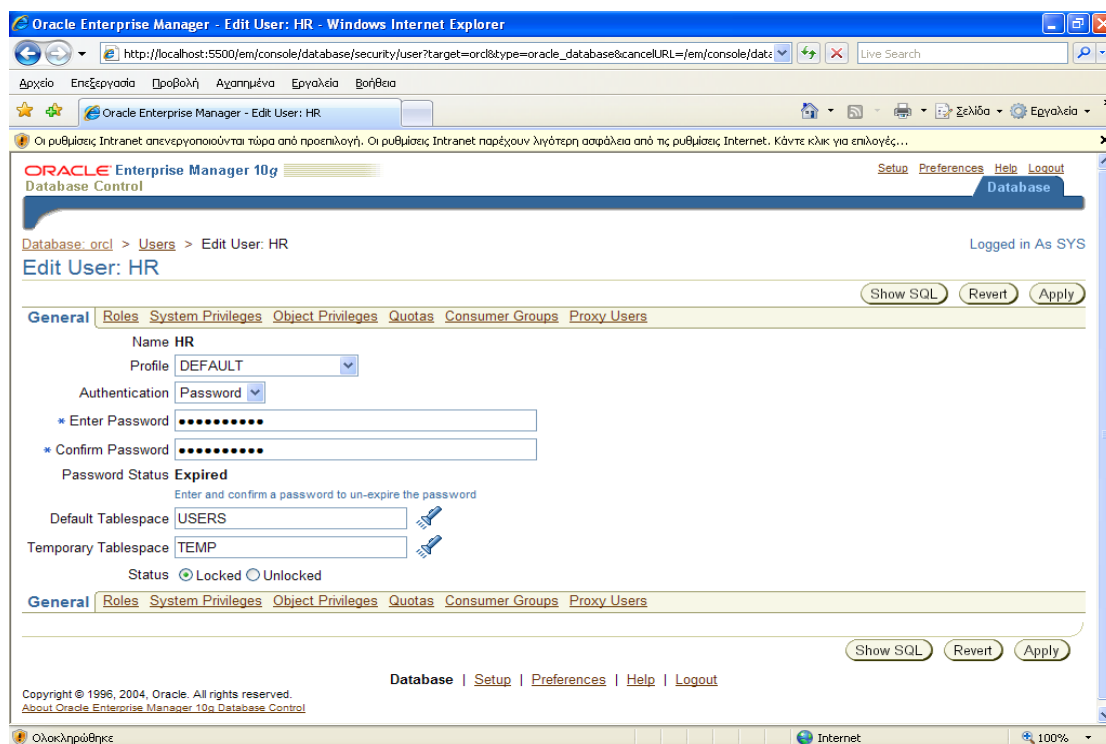
Σε κάθε χρήστη αντιστοιχεί μια συλλογή από όλα τα αντικείμενα που δημιουργούνται στην Database και ονομάζεται schema. Τα αντικείμενα αυτά είναι: πίνακες (tables), όψεις (views), δείκτες (indexes), διαδικασίες (procedures), sequences, συνώνυμα (synonyms) κλπ.

### **Προκαθορισμένοι Χρήστες (Predefined users),**

Με την εγκατάσταση του λογισμικού της Oracle και την δημιουργία των Δομών της Database, δημιουργούνται οι παρακάτω χρήστες με ιδιαίτερα δικαιώματα και ρόλους για την διαχείριση της Oracle Database:

- **SYS** Έχει τα δικαιώματα διαχειριστή της Βάσης Δεδομένων (DBA) και όλα τα δικαιώματα με την επιλογή ADMIN OPTION. Έχει πρόσβαση στο Λεξικό Δεδομένων και χρησιμοποιείται για την εκκίνηση και τον τερματισμό της Oracle Database.
- **SYSTEM:** Έχει τα δικαιώματα του DBA, MGMT\_USER και AD\_ADMINISTRATOR
- **SYSMAN:** Έχει τα δικαιώματα του MGMT\_USER, RESOURCE και SELECT\_CATALOG\_ROLE.

Σημειώνεται ότι οι χρήστες αυτοί δεν ενδείκνυται να χρησιμοποιούνται για την διαχείριση δεδομένων χρηστών.



ΣΧΗΜΑ 7-7 Διαχείριση των Χρηστών

Η Διαχείριση των χρηστών (users) γίνεται από τις οθόνες του **Σχήματος 7-7**. Για την δημιουργία των χρηστών, δίδονται οι πληροφορίες που αναφέρθηκαν προηγουμένως.

### Πιστοποίηση Χρηστών (user authentication).

Πιστοποίηση χρηστών είναι μια διαδικασία κατά την οποία επιβεβαιώνεται η ταυτότητα (identity) ενός χρήστη, προκειμένου αυτός να μπορεί έχει πρόσβαση με ασφάλεια στα δεδομένα, στους πόρους και στις εφαρμογές της Oracle Database.

### Εξουσιοδότηση Χρηστών (user authorization).

Εξουσιοδότηση χρηστών είναι μια διαδικασία που ακολουθεί την επιτυχή πιστοποίηση του χρήστη στην Oracle Database, με σκοπό να του καθορίσει το επίπεδο πρόσβασης και ενεργειών στα διάφορα αντικείμενα (objects) και εφαρμογές (applications) αυτής.

**Οι μέθοδοι πιστοποίησης χρηστών είναι:**

- **Oracle Password (Κωδικός):** Δημιουργείται από την Oracle Database κατά την δημιουργία του χρήστη και ακολουθεί όλους τους διεθνείς κανόνες ασφαλείας δημιουργίας κωδικών. (Κρυπτογράφηση AES, άμεση αλλαγή με την είσοδο στο σύστημα, case-sensitive κλπ).
- **Με εξωτερική διαδικασία (External):** Στην περίπτωση αυτή η διαδικασία της πιστοποίησης γίνεται από το Λειτουργικό Σύστημα ή από Συστήματα όπως το Kerberos ή το Radius, με το απαραίτητο όνομα χρήστη και κωδικός ασφαλείας. Δεν απαιτείται περαιτέρω πιστοποίηση από την Oracle Database.

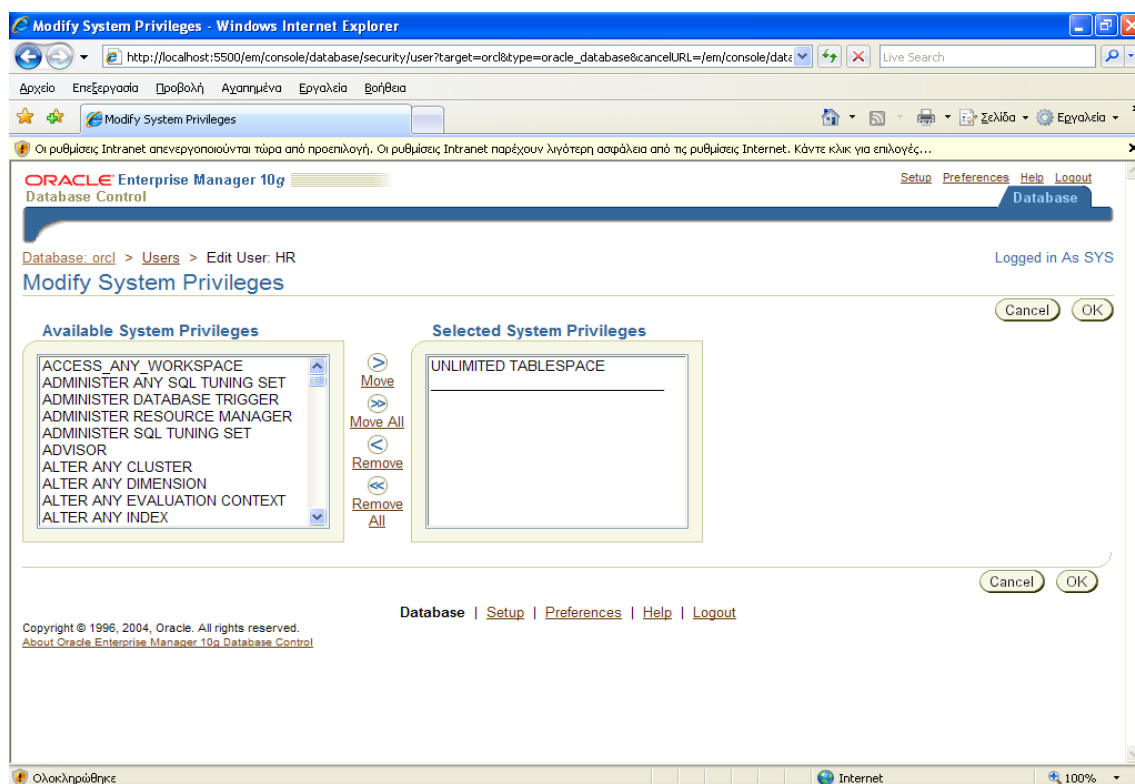
### **Δικαιώματα (Privileges).**

Δικαίωμα είναι η δυνατότητα που εκχωρείται σε ένα χρήστη για να εκτελεί SQL εντολές και να έχει πρόσβαση σε αντικείμενα άλλων χρηστών.

### **Κατηγορίες Δικαιωμάτων.**

**Υπάρχουν δύο είδη δικαιωμάτων:**

- **Τα δικαιώματα συστήματος (System privileges).** Είναι τα δικαιώματα τα οποία επιτρέπουν στους χρήστες να ενεργούν σε αντικείμενα της Oracle Database, όπως να δημιουργούν tablespaces. Τα δικαιώματα αυτά εκχωρούνται (granted) από τον Διαχειριστή της Βάσης (DBA). Υπάρχουν 170 δικαιώματα συστήματος. Τα δικαιώματα αυτά περιέχουν την λέξη any.
- **Τα δικαιώματα αντικειμένων.** Είναι δικαιώματα τα οποία επιτρέπουν σε ένα χρήστη να ενεργεί σε ένα πίνακα, όψη (view), συνάρτηση (function), διαδικασία (procedure) κλπ. Τα δικαιώματα αυτά εκχωρούνται είτε από τον DBA είτε από άλλο χρήστη που είναι και ο δημιουργός του αντικειμένου.



ΣΧΗΜΑ 7-8 Διαχείριση των Δικαιωμάτων Χρηστών

Τα δικαιώματα του συστήματος εκχωρούνται από την οθόνη του **Σχήματος 7-8**.

Μερικές Ειδικές κατηγορίες δικαιωμάτων συστήματος είναι:

- **SYSDBA και SYSOPER:** Επιτρέπουν την εκκίνηση και τερματισμό της Oracle Database.
- **CREATE, MANAGE, DROP και ALTER TABLESPACE:** Επιτρέπουν την διαχείριση των tablespaces.
- **CREATE ANY DIRECTORY:** Επιτρέπει την διαχείριση του κατλόγου της εγκατάστασης της ORACLE.
- **GRANT ANY OBJECT PRIVILEGE:** Επιτρέπει την εκχώρηση δικαιωμάτων σε αντικείμενα που δεν ανήκουν στο συγκεκριμένο χρήστη.
- **ALTER DATABASE και ALTER SYSTEM :** Επιτρέπουν την διαχείριση της Oracle Database.

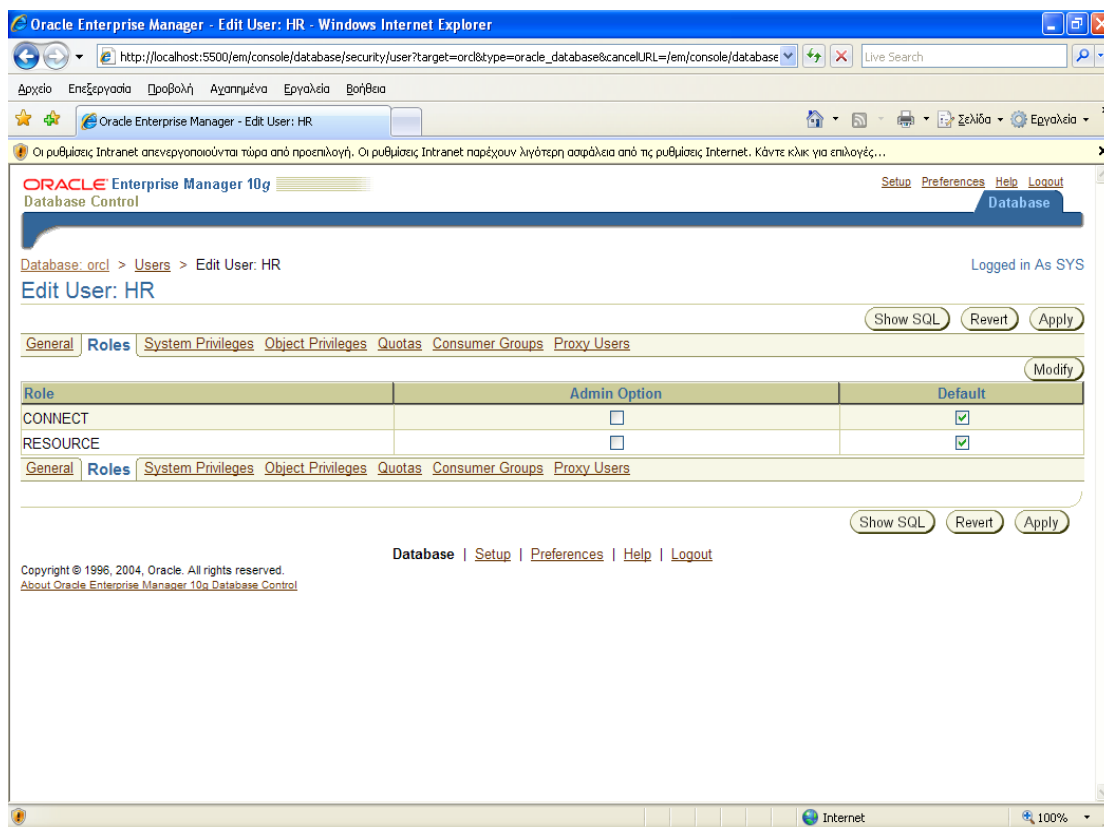


## Ρόλοι (Roles).

Οι ρόλοι χρησιμοποιούνται για την μαζική ανάθεση δικαιωμάτων σε χρήστες που έχουν τα ίδια δικαιώματα. Στους ρόλους ανατίθεται σύνολο δικαιωμάτων και στην συνέχεια στους χρήστες ανατίθενται οι ρόλοι. Αυτό έχει σαν αποτέλεσμα αν μεταβληθεί ένα δικαίωμα, όλοι οι χρήστες που τους έχει ανατεθεί το συγκεκριμένο δικαίωμα να ενημερώνονται αυτόματα. Οι ρόλοι μπορεί να ενεργοποιηθούν ή να απενεργοποιηθούν από τον DBA με αντίστοιχη επίπτωση και στα δικαιώματα που εξαρτώνται από αυτά. Αυτό έχει επίπτωση στην επιβολή και τον έλεγχο των ενεργειών των χρηστών της Oracle Database και την προσαρμογή σε κάθε παρουσιαζόμενη κατάσταση.

### Χαρακτηριστικά των ρόλων.

- Οι ρόλοι περιέχουν τόσο δικαιώματα συστήματος όσο και δικαιώματα χρηστών. Οι ρόλοι απαιτούν κωδικό ασφαλείας για να ενεργοποιηθούν.
- Οι ρόλοι ελέγχονται από τους χρήστες που τους δημιουργούν.
- Τα δικαιώματα προστίθενται και αφαιρούνται στους ρόλους από τους χρήστες που τους δημιουργούν.



ΣΧΗΜΑ 7-9 Διαχείριση των Ρόλων

Η διαχείριση των ρόλων γίνεται από την οθόνη του **Σχήματος 7-9**.

### Αντίγραφο Ασφαλείας (Oracle Database Backup).

Τα αντίγραφα ασφαλείας υλοποιούνται με τους παρακάτω τρόπους:

- Με το λογισμικό Recovery Manager (RMAN).
- Με την διαδικασία Oracle Secure Backup η οποία δίδει την δυνατότητα λήψεως των Backup και των αρχείων του λειτουργικού συστήματος και της λήψεως Backup σε ταινία.
- Με διαδικασίες του Διαχειριστή της Βάσης (DBA) (User-Managed Backup). Οι διαδικασίες αυτές αφορούν την σύνταξη αρχείων κώδικα (scripts) και απαιτούν μεγαλύτερη προσπάθεια και έλεγχο. Η πλέον ενδεδειγμένη διαδικασία είναι με τον RMAN.

### Ορισμοί Λήψεως Αντιγράφων Ασφαλείας.

- **Αντίγραφο Ασφαλείας όλης της Βάσης (Full backup ).** Περιλαμβάνει την λήψη αντιγράφου ασφαλείας, όλων των data files και ενός τουλάχιστον control file.
- **Μερικό Αντίγραφο Ασφαλείας (Partial database backup).** Περιλαμβάνει την λήψη αντιγράφου ασφαλείας μέρους των data files.
- **Κλιμακούμενο Αντίγραφο Ασφαλείας (Incremental Backup)** Περιλαμβάνει την λήψη αντιγράφου ασφαλείας μόνον των data files που μεταβλήθηκαν από το τελευταίο αντίγραφο ασφαλείας (backup).
- **Offline backup.** Περιλαμβάνει αντίγραφο ασφαλείας το οποίο λαμβάνεται όταν η Βάση δεν έχει ανοίξει.
- **Online backup.** Περιλαμβάνει αντίγραφο ασφαλείας το οποίο λαμβάνεται όταν η Βάση έχει ανοίξει.

Το offline backup είναι ασφαλέστερο διότι δεν υπάρχουν μεταβολές στην Βάση κατά την διάρκεια που λαμβάνεται σε αντίθεση με το online το οποίο λαμβάνεται την στιγμή που τα data files μεταβάλλονται αφού η Βάση είναι σε λειτουργία.

### **User – Managed Backup.**

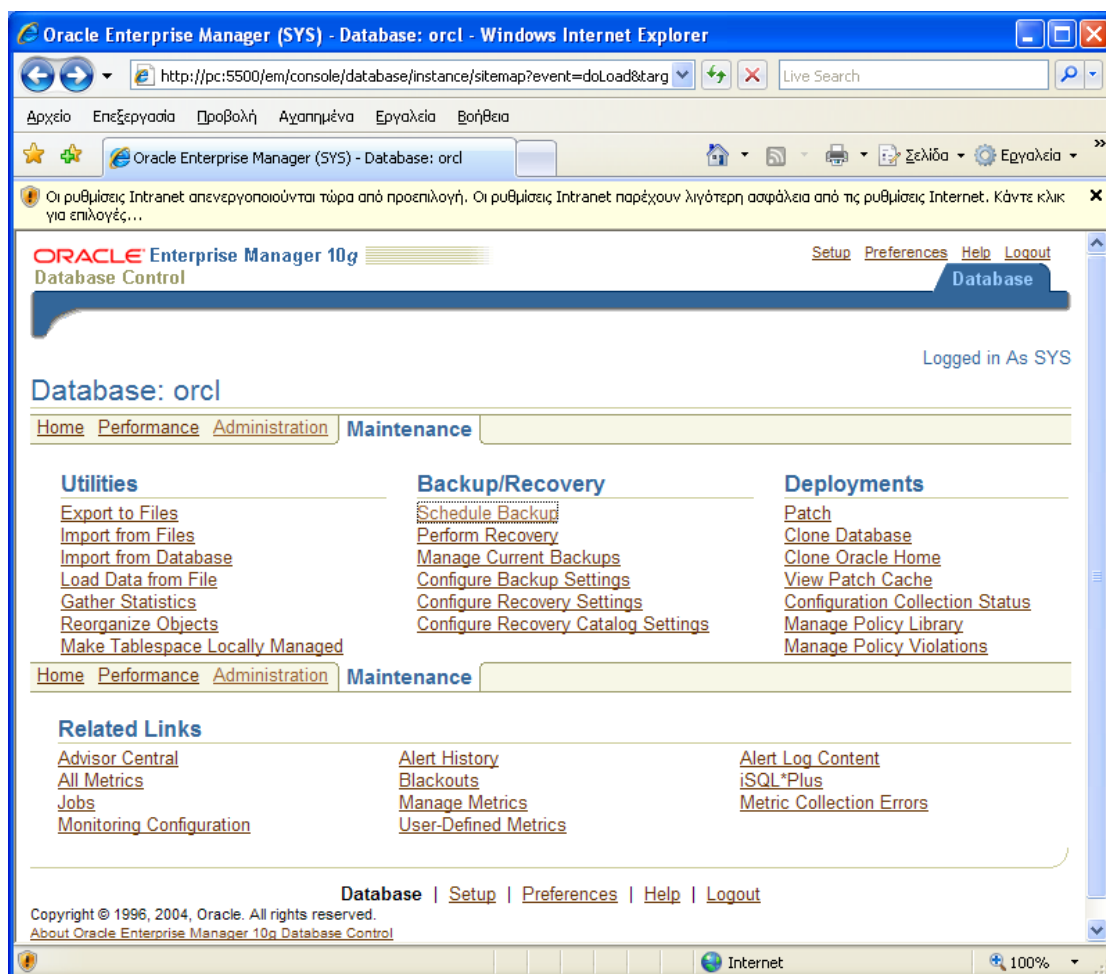
Η διαδικασία αυτή όπως αναφέρθηκε χρησιμοποιεί εντολές απευθείας πρόσβασης στην Oracle Database (command – line) και την δημιουργία προσωπικών σεναρίων του DBA. Συνηθισμένες εντολές που χρησιμοποιούνται στα scripts είναι:

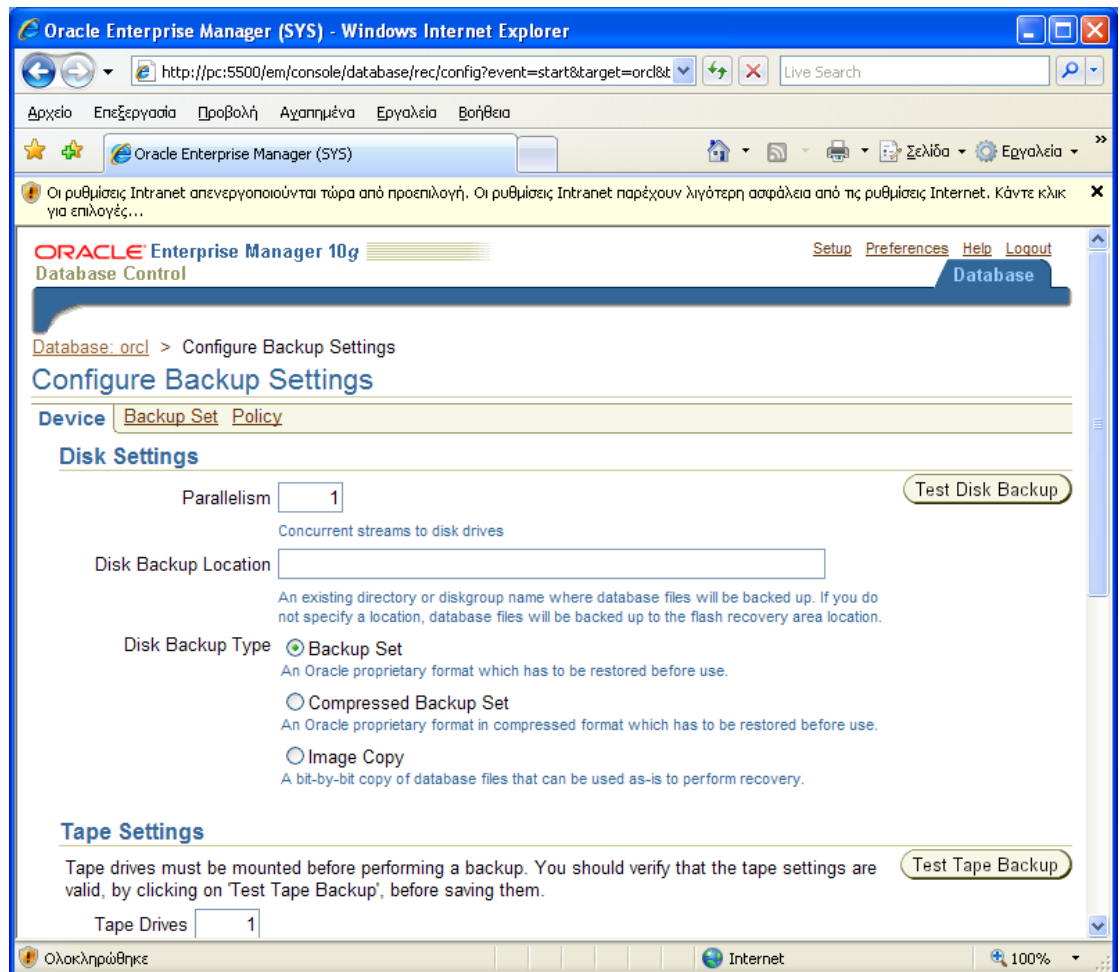
- **V\$DATAFILE** για την εξέταση των data files που θα ληφθούν backup
- **V\$LOGFILE** για την εξέταση των online log files.
- **V\$BACKUP** για την εξέταση των data files που αποτελούν μέρος των tablespace που έχουν τεθεί σε κατάσταση Backup (backup mode).
- Κατάλληλες εντολές του λειτουργικού συστήματος (αντιγραφή – copy).

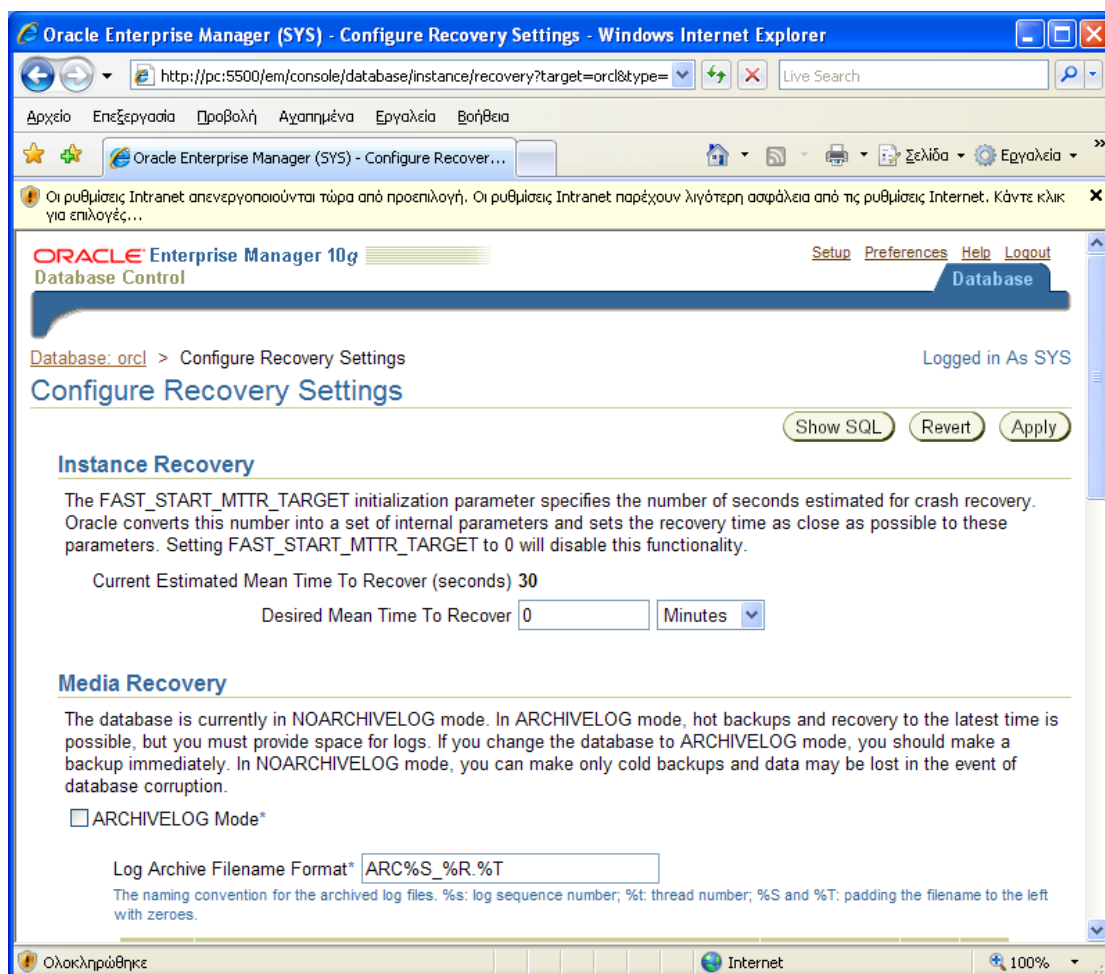
### **Recovery Manager (RMAN).**

Ο RMAN αποτελεί λογισμικό του Oracle Server και χρησιμοποιείται για την λήψη αντιγράφων ασφαλείας (backup) αλλά και επαναφορά του συστήματος (recovery) σε περίπτωση αστοχίας. Λαμβάνει όλα τα είδη των backups που περιγράφηκαν παραπάνω. Έχει δικό του περιβάλλον χειρισμού και γλώσσα (scripting) αλλά υπάρχει και σαν API. Έχει την δυνατότητα να αποθηκεύει αντίγραφα ασφαλείας σε ταινίες για μεγάλο χρονικό διάστημα. Ο EM παρέχει γραφικό περιβάλλον στο RMAN, αλλά όταν απαιτούνται προχωρημένες τεχνικές λήψεως αντιγράφων ασφαλείας, χρησιμοποιούνται εντολές άμεσης πρόσβασης στο σύστημα (command – line).

Στα σχήματα παρακάτω εμφανίζονται οι οθόνες του EM, που χρησιμοποιούνται για την λήψη αντιγράφων ασφαλείας.







Options Settings Schedule Review

Schedule Customized Backup: Schedule

Database orcl.oracle.com

Backup Strategy Customized Backup

Object Type Whole Database

Job

Job Name BACKUP\_ORCL\_ORACLE.COM\_001

Job Description Whole Database Backup

Schedule

Type ☐ One Time (Immediately) ☐ One Time (Later) ☒ Repeating

Frequency Type By Minutes

Repeat Every 1 Minutes

Time Zone (UTC-08:00) US Pacific Time (PST)

Start Date Jun 18, 2009

Start Time 1:00 AM

Repeat Until ☒ Indefinite ☐ Specified Date

Date (example: Jun 18, 2009)

Time AM PM

ΣΧΗΜΑ 7-10 Διαχείριση των Αντιγράφων Ασφαλείας

**Οι παράμετροι των αντιγράφων ασφαλείας είναι:**

- **Parallelism**
- Καθορίζει τον αριθμό των streams που χρησιμοποιούνται κατά την λήψη των backups
- **Disk backup location**
- Καθορίζει την τοποθεσία που λαμβάνονται τα backups
- **Disk backup type**
- Καθορίζει το είδος των backup set.

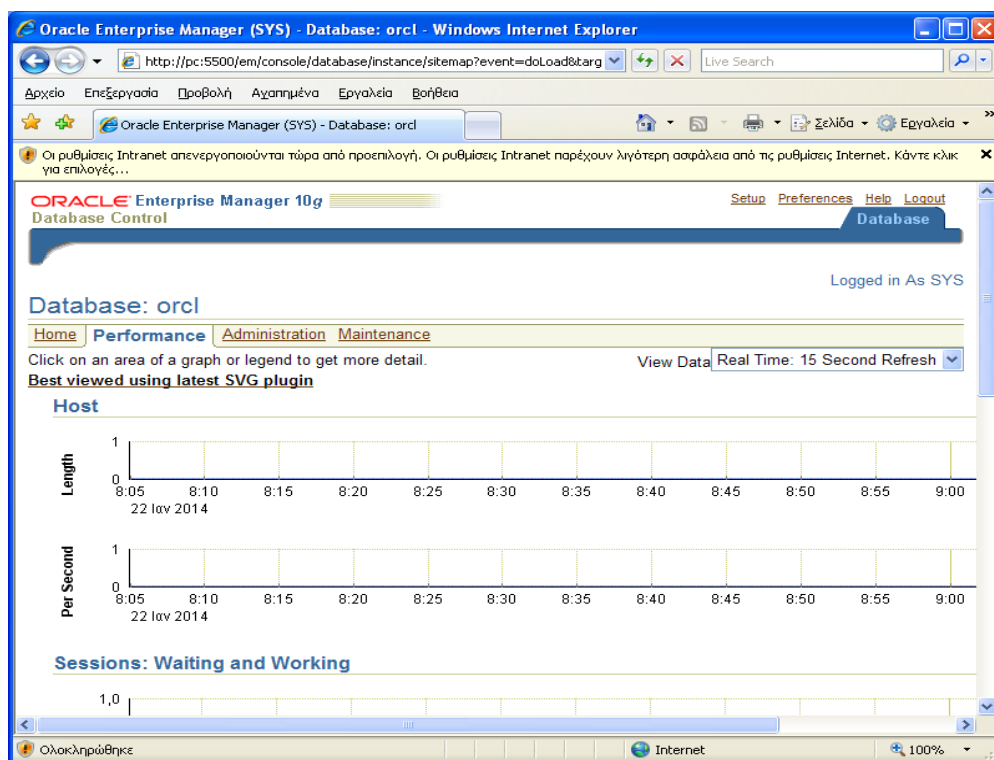
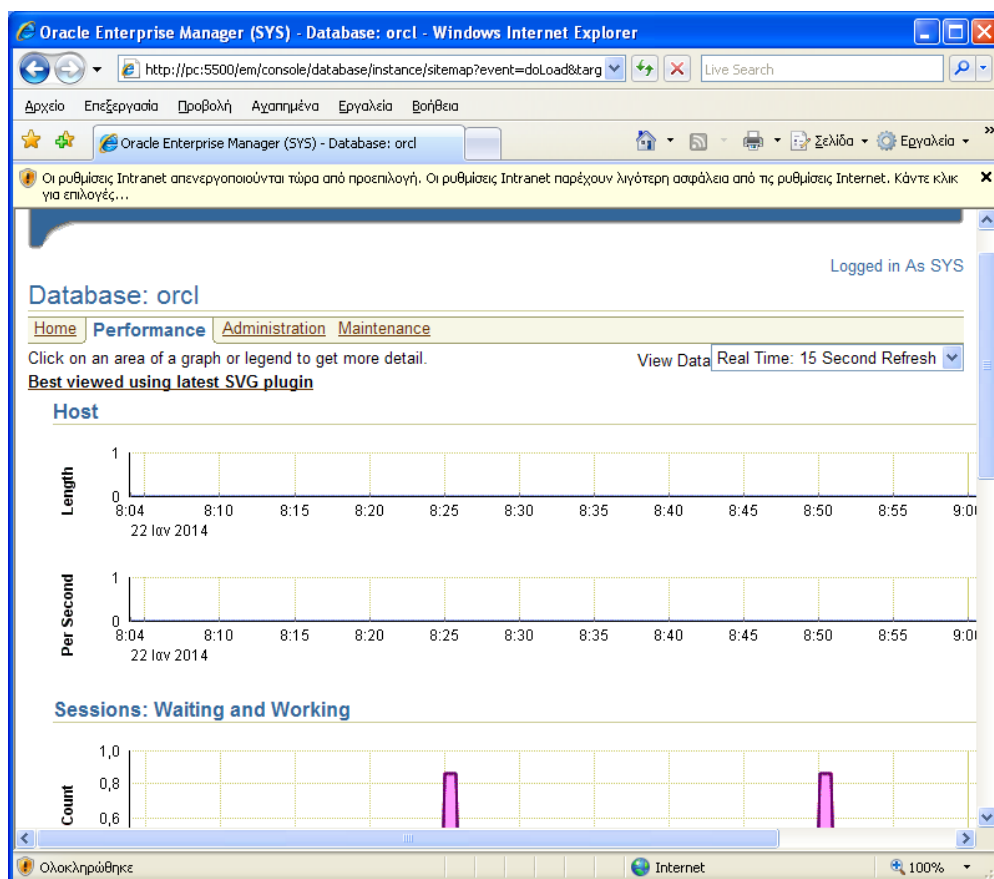
### **Απόδοση Β.Δ. (Performance).**

Ο Διαχειριστής της Oracle Database (DBA) πρέπει σε τακτά χρονικά διαστήματα να ελέγχει την απόδοση της λειτουργίας της Βάσης προκειμένου να προλαβαίνει τις καθυστερήσεις (bottlenecks) και να διορθώνει τα προβλήματα.

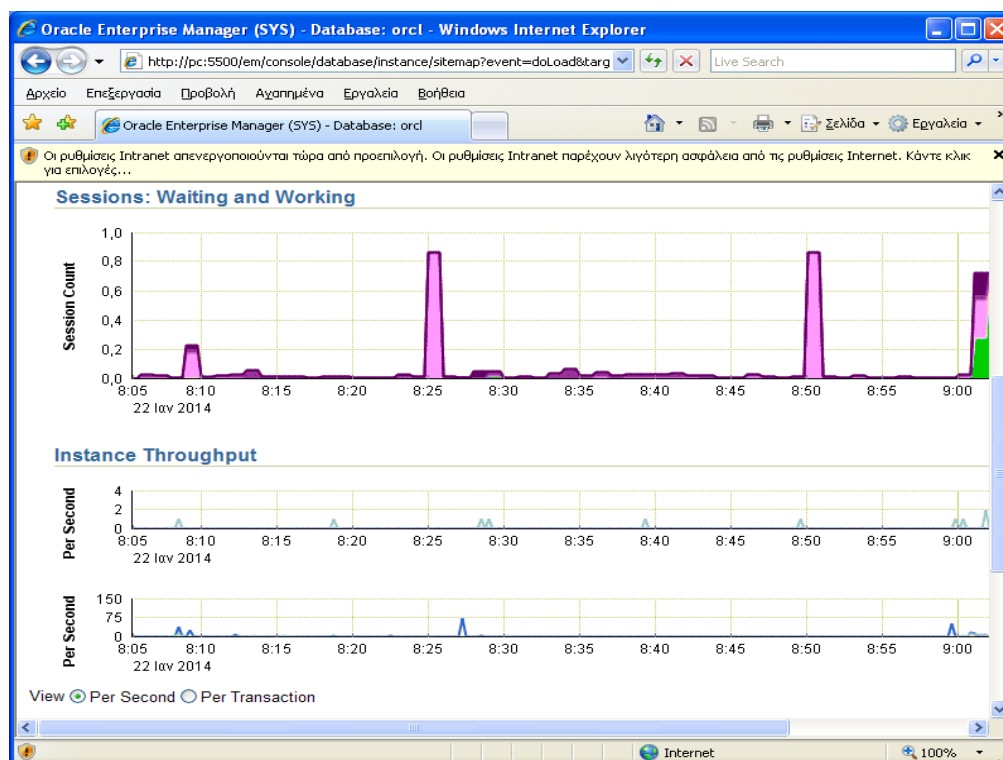
**Τα προβλήματα αυτά αναφέρονται:**

- Στο Μέγεθος μνήμης που εκμεταλλεύεται η Oracle Database.
- Στις καθυστερήσεις των μονάδων εισόδου – εξόδου (disk I/O).
- Σε σφάλματα των εφαρμογών που εκμεταλλεύονται την Βάση Δεδομένων.
- Σε καθυστερήσεις του Δικτύου (network).
- Σε καθυστερήσεις των πόρων του συστήματος .

Για την αντιμετώπιση όλων αυτών των προβλημάτων ο DBA λαμβάνει μετρήσεις για την απόδοση των διαφόρων τμημάτων της Βάσης. Η παρακολούθηση της απόδοσης της Βάσης, γίνεται από οθόνες του EM όπως παρακάτω:







ΣΧΗΜΑ 7-11 Παρακολούθηση των Μετρήσεων μιας Oracle Database

## Παράρτημα Ι – Περιγραφή του σχήματος HR.

### Εισαγωγή

Κάθε έκδοση της Oracle Database συνοδεύεται από κάποια έτοιμα σχήματα, που μπορούν να εγκατασταθούν και στη συνέχεια να χρησιμοποιηθούν ως μία κοινή πλατφόρμα, για εξάσκηση στις δυνατότητες του Oracle RDBMS.

Τα Oracle Database Sample Schemas είναι μια σειρά αλληλένδετων σχημάτων. Το σύνολο αυτών των σχημάτων, παρέχει μία στρωματοποιημένη προσέγγιση όσον αφορά την πολυπλοκότητα των παραδειγμάτων.

- Ένα απλό σχήμα διαχείρισης Ανθρώπινων Πόρων (HR), είναι χρήσιμο για τις εισαγωγικές έννοιες. Αυτό είναι το σχήμα με το οποίο θα εργαστούμε στα πλαίσια αυτού του υλικού.
- Το δεύτερο σχήμα, Order Entry (OE), είναι χρήσιμο για την αντιμετώπιση θεμάτων ενδιάμεσης πολυπλοκότητας.
- Το Online Catalog (OC) υποσχήμα είναι μια συλλογή από αντικειμενοστραφή αντικείμενα, που έχουν δημιουργηθεί μέσα στο OE σχήμα.
- Το Product Media (PM) σχήμα είναι αφιερωμένο σε τύπους δεδομένων κατάλληλους για πολυμέσα.
- Παρέχονται επίσης δύο σχήματα το Information Exchange (IX) και το Sales History (SH), για ακόμα πιο προχωρημένα θέματα.

### Σύντομη περιγραφή του παραδείγματος.

Η εταιρεία του παραδείγματος λειτουργεί σε όλο τον κόσμο (πολυεθνική), προκειμένου να καλύψει παραγγελίες για διάφορα προϊόντα. Η εταιρεία έχει πολλά τμήματα:

- Το τμήμα Ανθρώπινου Δυναμικού (Human Resources) παρακολουθεί πληροφορίες σχετικά με τους εργαζόμενους και τις εγκαταστάσεις.
- Το τμήμα παραγγελιών (Order Entry) παρακολουθεί τα αποθέματα των προϊόντων και των πωλήσεων της εταιρείας.

- Το τμήμα Product Media διαχειρίζεται τις περιγραφές και τις λεπτομερείς πληροφορίες, σχετικά με κάθε προϊόν που πωλείται από την εταιρεία.
- Το τμήμα Ανταλλαγής Πληροφοριών (The Information Exchange division) διαχειρίζεται την ανταλλαγή πληροφοριών μέσω B2B εφαρμογών.
- Το τμήμα πωλήσεων (Sales division ) πραγματοποιεί στατιστικές μελέτες προς διευκόλυνση λήψης επιχειρηματικών αποφάσεων.

Κάθε ένα από αυτά τα τμήματα αντιπροσωπεύεται από ένα σχήμα.

Στο (HR) σχήμα για κάθε υπάλληλο αποθηκεύεται, ένας κωδικός, το ονοματεπώνυμο, το e-mail, το τηλέφωνο, η ημερομηνία πρόσληψης, ο κωδικός εργασίας (job identification code), ο μισθός, ο κωδικός του προϊσταμένου, ο κωδικός του τμήματος και η προμήθεια που έχει κερδίσει.

Για κάθε «θέση εργασίας» (job identification code) αποθηκεύεται ο κωδικός, η περιγραφή, ο ελάχιστος και ο μέγιστος προβλεπόμενος μισθός.

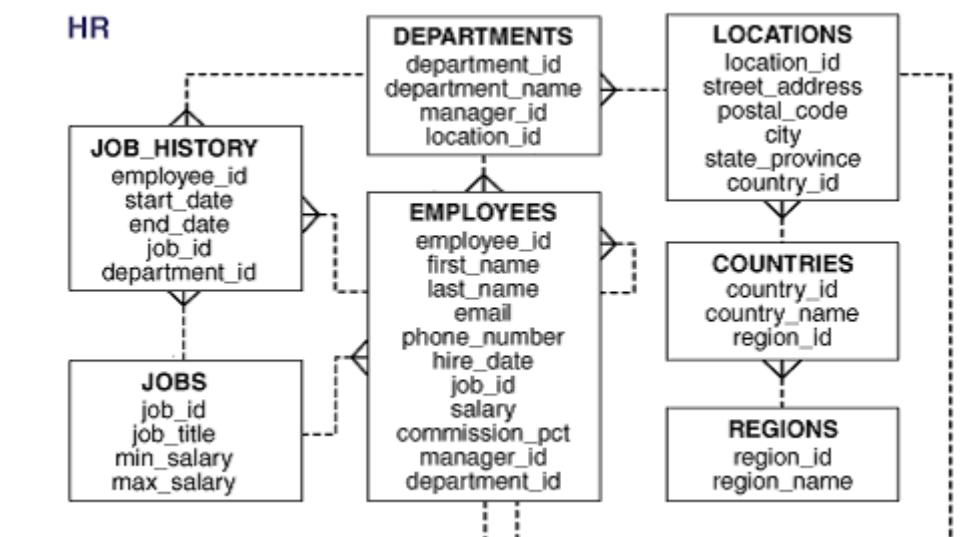
Σε περίπτωση που κάποιος αλλάξει θέση στην εταιρεία αυτό καταγράφεται σε έναν βοηθητικό πίνακα.

Η εταιρεία του παραδείγματος είναι πολυεθνική, και χρειάζεται να παρακολουθεί τις θέσεις των εγκαταστάσεων και των υπηρεσιών της.

Κάθε υπάλληλος είναι ενταγμένος σε κάποιο τμήμα και κάθε τμήμα είναι σε κάποια «τοποθεσία», η οποία με τη σειρά της έχει καταχωρισμένα τα πλήρη στοιχεία π.χ. διεύθυνση, Τ.Κ., πόλη, πολιτεία ή επαρχία και τον κωδικό της Χώρας που βρίσκεται. Στις τοποθεσίες που η εταιρεία έχει εγκαταστάσεις αποθηκεύονται και κάποια στοιχεία για το κράτος και τη γεωγραφική περιοχή που βρίσκεται.

## **HR σχήμα**

Παρακάτω παρουσιάζεται το (HR) σχήμα καθώς και οι πίνακες από τους οποίους αποτελείται.



Εικόνα 1: Το σχήμα HR, πηγή: Oracle Database Sample Schemas 11g Release 1 (11.1)

## Περιγραφή των πινάκων του HR

### HR.COUNTRIES

Όνομα στήλης	Null?	Τύπος Δεδομένων (Type)
COUNTRY_ID	NOT NULL	CHAR(2)
COUNTRY_NAME		VARCHAR2(40)
REGION_ID		NUMBER

### HR.DEPARTMENTS

Όνομα στήλης	Null?	Τύπος Δεδομένων (Type)
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

### HR.EMPLOYEES

Όνομα στήλης	Null?	Τύπος Δεδομένων (Type)
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

#### HR.JOBS

Όνομα στήλης	Null?	Τύπος Δεδομένων (Type)
JOB_ID	NOT NULL	VARCHAR2(10)
JOB_TITLE	NOT NULL	VARCHAR2(35)
MIN_SALARY		NUMBER(6)
MAX_SALARY		NUMBER(6)

#### HR.JOB\_HISTORY

Όνομα στήλης	Null?	Τύπος Δεδομένων (Type)
EMPLOYEE_ID	NOT NULL	NUMBER(6)
START_DATE	NOT NULL	DATE
END_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
DEPARTMENT_ID		NUMBER(4)

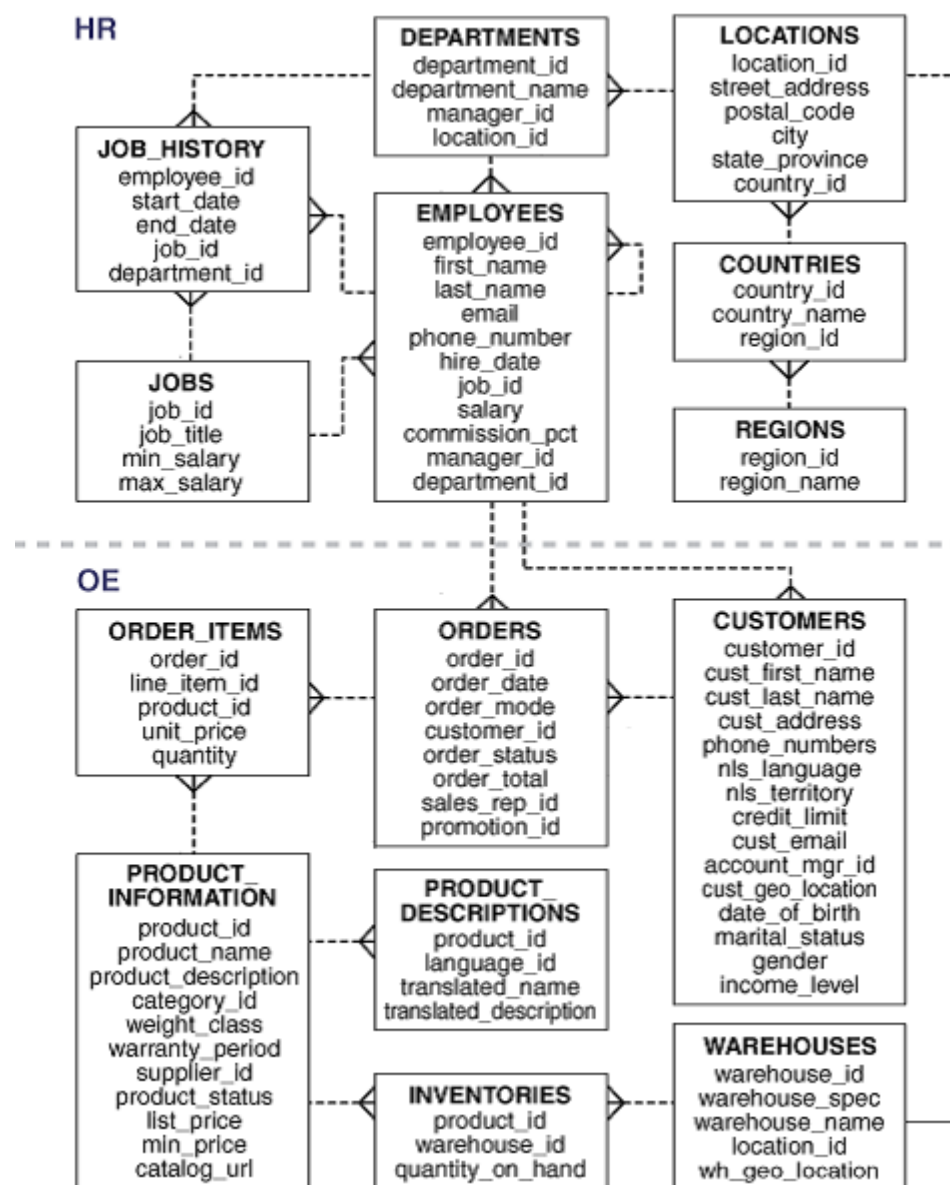
#### HR.LOCATIONS

Όνομα στήλης	Null?	Τύπος Δεδομένων (Type)
LOCATION_ID	NOT NULL	NUMBER(4)
STREET_ADDRESS		VARCHAR2(40)
POSTAL_CODE		VARCHAR2(12)
CITY	NOT NULL	VARCHAR2(30)
STATE_PROVINCE		VARCHAR2(25)
COUNTRY_ID		CHAR(2)

#### HR.REGIONS

Όνομα στήλης	Null?	Τύπος Δεδομένων (Type)
REGION_ID	NOT NULL	NUMBER
REGION_NAME		VARCHAR2(25)

Στο τελευταίο διάγραμμα παρουσιάζεται η συσχέτιση του (HR ) σχήματος με το (OE) σχήμα.



Εικόνα 2: Συσχέτιση των σχημάτων HR και OE, πηγή: Oracle Database Sample Schemas 11g Release 1 (11.1)

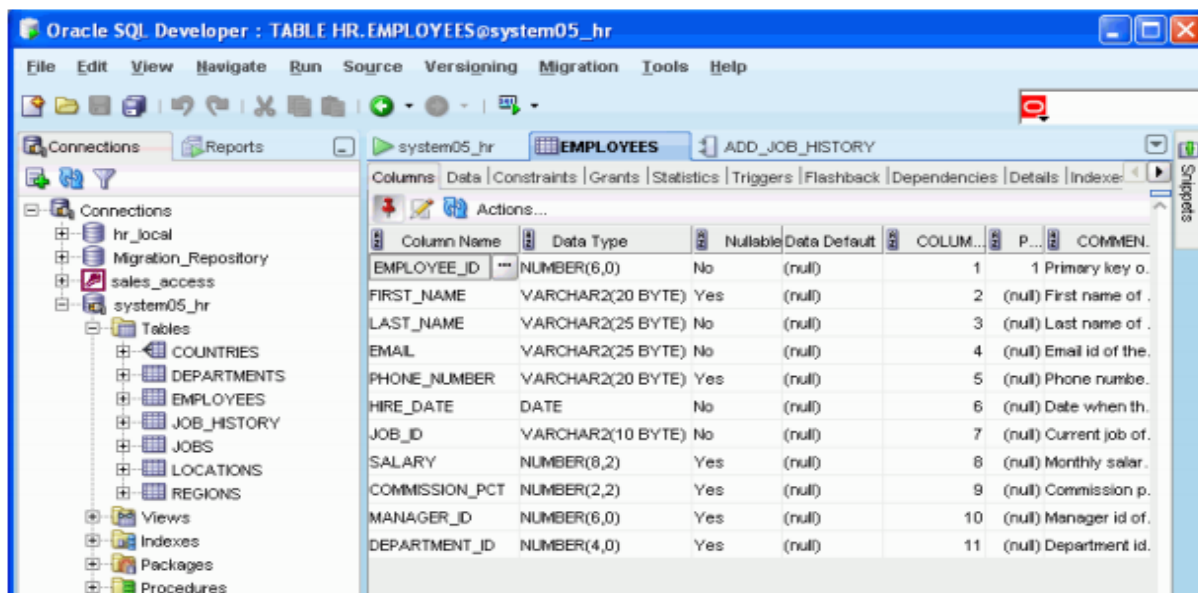
## Παράρτημα II – Εισαγωγή στη χρήση του SQLDeveloper

### Ορισμός.

Ο Oracle SQL Developer είναι μια γραφική εφαρμογή, η οποία παρέχει τη δυνατότητα πρόσβασης σε μια Oracle Database, για την εκτέλεση απλών αλλά και πολύπλοκων διεργασιών. Αποτελεί την εξέλιξη της SQL\*Plus. Συνολικά με τον Oracle SQL Developer μπορούμε:

- Να συνδεθούμε σε οποιανδήποτε Oracle database schema, χρησιμοποιώντας διαδικασίες πιστοποίησης χρηστών.
- Να συνδεθούμε με άλλες Βάσεις Δεδομένων, όπως MySQL, Microsoft SQL Server, Microsoft Access κ.α.
- Να δημιουργήσουμε και να επεξεργαστούμε αντικείμενα (objects) της Oracle Database.
- Να εκτελέσουμε (run) εντολές (statements) και αρχεία (scripts) εντολών SQL και PL/SQL.
- Να διαχειριστούμε και να επεξεργαστούμε δεδομένα (data manipulation).
- Να εισάγουμε (import) και να εξάγουμε (export) δεδομένα.
- Να δημιουργήσουμε όψεις (views) δεδομένων.
- Να παράγουν εκτυπωτικές καταστάσεις με τα επεξεργασμένα δεδομένα.

## Το Περιβάλλον Διαχείρισης.



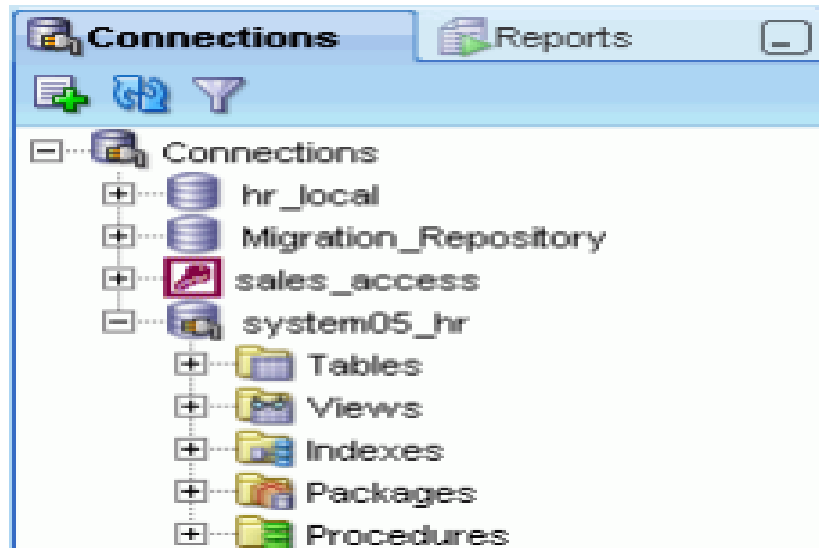
Σχήμα 1 Το περιβάλλον του SQL Developer

Οι επιλογές των menu στην κορυφή του παραθύρου του SQL Developer είναι οι παρακάτω:

- **New:** Δημιουργεί νέα αντικείμενα στην Βάση Δεδομένων. (new database objects).
- **Open:** Ανοίγει ένα αρχείο.
- **Save:** Αποθηκεύει ένα επιλεγμένο αντικείμενο της Βάσης Δεδομένων.
- **Save All:** Αποθηκεύει όλα τα ανοικτά αντικείμενα της Βάσης Δεδομένων.
- **Back:** Επιστρέφει στην προηγούμενη οθόνη.
- **Forward:** Μεταφέρει την οθόνη στην προηγούμενη από αυτή που πρόσφατα είχε επιλεγθεί.
- **Open SQL Worksheet:** Ανοίγει ένα περιβάλλον εργασίας SQL.



## To menu Connections.



Σχήμα 2 To menu Connections

Βρίσκεται στο αριστερό τμήμα του παραθύρου του SQL Developer και περιέχει τις εξής επιλογές:

- **Connections navigator:** Περιλαμβάνει όλες τις συνδέσεις με την Βάση Δεδομένων που έχουν δημιουργηθεί.
- **Files navigator:** Περιλαμβάνει σε ιεραρχική δομή, όλα τα αρχεία και τους φακέλους που έχουν δημιουργηθεί .
- **Reports navigator:** Περιλαμβάνει τις αναφορές που παρέχονται από τον SQL Developer.

### To menu Edit:

Περιλαμβάνει τις παρακάτω επιλογές:

- **Extended Paste:** Απεικονίζει το εκτεταμένο menu επικόλλησης (paste).
- **Duplicate Selection:** Δημιουργεί αντίγραφα των επιλεγμένων αντικειμένων.
- **Wrap Selection:** Δημιουργεί αναδίπλωση του επιλεγμένου αρχείου.

### To menu View:

Περιλαμβάνει τις παρακάτω επιλογές:

- **Captured Models, Converted Models, Versioning Navigator:** Απεικονίζει τους αντίστοιχους επιλογείς. (navigators).
- **Find DB Object:** Χρησιμοποιείται για την εύρεση αντικειμένων της Βάσης Δεδομένων.
- **Log:** Απεικονίζει τα μηνύματα λειτουργίας της Βάσης Δεδομένων.
- **Debugger:** Απεικονίζει την διαδικασία αποσφαλμάτωση.
- **Run Manager:** Απεικονίζει διάφορες διαδικασίες απασφαλμάτωσης.

### To menu Navigate:

Περιλαμβάνει τις παρακάτω επιλογές:

- **Toggle Bookmark, Remove Bookmarks from File, Remove All Bookmarks, Go to Bookmark, Go to Previous Bookmark:** Απεικονίζει τις επιλογές των Bookmark.

### To menu Run:

Περιλαμβάνει τις επιλογές εκτέλεσης των διαφόρων εντολών του SQL Developer.

## To Tools menu.

Περιλαμβάνει τα παρακάτω αντικείμενα:

- **Database Copy:** Χρησιμοποιείται για την δημιουργία αντιγράφων αντικειμένων της Βάσης Δεδομένων.
- **Database Export:** Χρησιμοποιείται για την εξαγωγή αντικειμένων της Βάσης Δεδομένων.
- **Database Diff:** Χρησιμοποιείται για την σύγκριση δύο σχημάτων της Βάσης Δεδομένων.
- **Monitor SQL:** Χρησιμοποιείται για την απεικόνιση πληροφοριών για τις εντολές που εκτελούνται.
- **Help menu:** Χρησιμοποιείται για την παροχή βοήθειας για τις δυνατότητες του SQL Developer.
- **Search:** Χρησιμοποιείται για την εύρεση αντικειμένων της Βάσης Δεδομένων.
- **Check for Updates:** Χρησιμοποιείται για τον έλεγχο των ενημερώσεων του SQL Developer.

## Βιβλιογραφία – Ιστότοποι

### Βιβλιογραφία:

1. *Εισαγωγή στα Συστήματα Βάσεων Δεδομένων, C J DATE*
2. *Fundamentals of DATABASE SYSTEMS, Elmasri & Navathe.*
3. *OCA ORACLE DATABASE 11g ADMINISTRATOR CERTIFIED STUDY GUIDE, B Thomas SYBEX*
4. *ORACLE 9i The Complete Reference (Kevin Loney, George Koch And the Experts at TUSC Oracle Press)*
5. *Oracle® Database SQL Language Reference 11g Release 2 (11.2), ORACLE*
6. *Oracle® Database PL/SQL Language Reference 11g Release 1 (11.1) , ORACLE*
7. *Oracle® Database Administrator's Guide 11g Release 1 (11.1), ORACLE.*
8. *Oracle® Database 2 Day DBA 11gRelease 2 (11.2), ORACLE.*
9. *Oracle® Database 11g Administrator Workshop I, ORACLE*
10. *Oracle® Database Sample Schemas 11g Release 1 (11.1) ORACLE*
11. *Oracle® Database Concepts 11g Release 1 (11.1) ORACLE*
12. *Oracle Database SQL Developer User's Guide*
13. *Μάθετε τηνSQL σε 24 Ώρες (Stephens Plew)*

### Ιστότοποι:

1. *The PL/SQL Tutorial, <http://plsql-tutorial.com/>*
2. <http://www.oracle.com/technetwork/index.html>