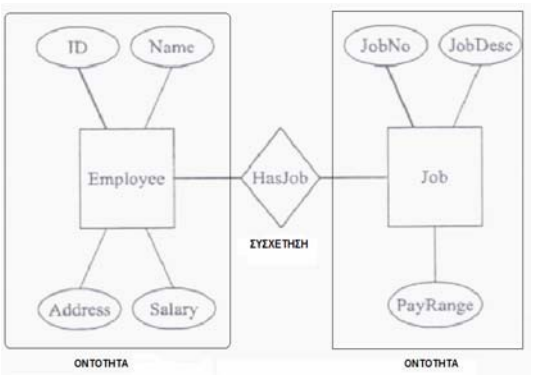
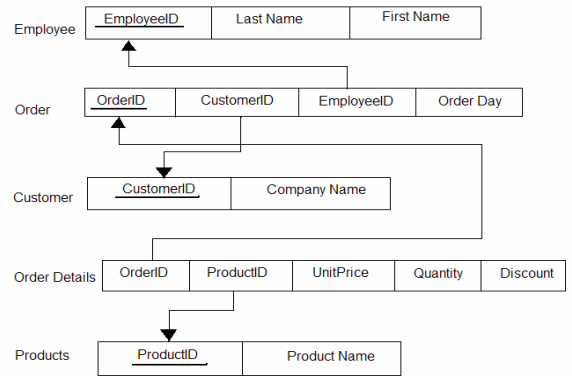
1 ΕΙΣΑΓΩΓΗ

1.1 ΣΥΣΤΗΜΑΤΑ ΔΙΑΧΕΙΡΙΣΗΣ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ

Μια **βάση δεδομένων** είναι μια συλλογή δεδομένων, τα οποία φαίνεται να έχουν μια συγκεκριμένη δομή ή σχήμα με το οποίο σχετίζονται. Για παράδειγμα, (Ταυτότητα, Όνομα, Διεύθυνση, Μισθός, Αρ. Εργασίας) μπορεί να αποτελεί το σχήμα για μια βάση δεδομένων προσωπικού. Με το παράδειγμα αυτό θέλουμε να δηλώσουμε ότι κάθε εγγραφή στη βάση δεδομένων προσωπικού έχει μια τιμή για κάθε ένα από τα πέντε γνωρίσματα. Τα δεδομένα που έχουν αποθηκευτεί σε μια βάση συχνά οπτικοποιούνται σε ένα μοντέλο δεδομένων το οποίο χρησιμοποιείται για να περιγράψει τα δεδομένα, τα γνωρίσματα και τις συσχετίσεις μεταξύ τους. Ένα ευρέως διαδεδομένο μοντέλο είναι το μοντέλο **Οντοτήτων-Συσχετίσεων - ΟΣ** (Entity - Relationship - ER). Τα βασικά συστατικά ενός τέτοιου μοντέλου είναι οι οντότητες και οι συσχετίσεις. Μία **οντότητα** (entity) σχετίζεται με ένα πραγματικό αντικείμενο και έχει ένα κλειδί το οποίο την ταυτοποιεί μοναδικά. Μία **συσχέτιση** (relationship) χρησιμοποιείται για να περιγράψει τη σχέση που υπάρχει μεταξύ των οντοτήτων



Παράδειγμα μοντέλου Οντοτήτων-Συσχετίσεων



Παράδειγμα του σχεσιακού μοντέλου

Ένα Σύστημα διαχείρισης βάσεων δεδομένων (ΣΔΒΔ) (database management system-DBMS) είναι μια συλλογή προγραμμάτων που επιτρέπουν στους χρηστές να δημιουργούν και να συντηρούν μια βάση δεδομένων. Τα ΣΔΒΔ συχνά απεικονίζουν τα δεδομένα σε μια δομή τύπου πίνακα. Αυτή είναι η αφορμή για την εισαγωγή του σχεσιακού μοντέλου (relational model), όπου τα δεδομένα απεικονίζονται να αποτελούνται από σχέσεις (σχήμα(β)). Η πρόσβαση σε μια βάση δεδομένων συνήθως επιτυγχάνεται μέσω μιας γλώσσας ερωτήσεων (query language). Η πιο διαδεδομένη, που χρησιμοποιείται από τα περισσότερα ΣΔΒΔ, είναι η SQL.

```
SELECT Name
FROM  R
WHERE Salary>10000
```

1.2 ΤΥΠΙΚΕΣ ΕΦΑΡΜΟΓΕΣ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ

1.2.1 Μικρές βάσεις δεδομένων

Η SQL και οι σχεσιακές βάσεις δεδομένων αναπτύχθηκαν αρχικά για να χειριστούν μεγάλο όγκο δεδομένων. Ήταν αρχικά υπερβολή να χρησιμοποιηθούν για να χειριστούν μικρό όγκο δεδομένων. Σήμερα όμως, η SQL και οι σχεσιακές βάσεις δεδομένων χρησιμοποιούνται σε χιλιάδες μικρές βάσεις, όπως για τα λογιστικά ή δικηγορικά γραφεία, καταστήματα της αγοράς, ακόμη και για την οργάνωση ενός βιβλίου διευθύνσεών. Κύρια προϊόντα για την ανάπτυξη μικρότερων βάσεων δεδομένων είναι η MS Access, Dbase, Paradox οι οποίες δεν υποστηρίζουν βέβαια συναλλαγές.

1.2.2 Συστήματα αναλυτικής επεξεργασίας συναλλαγών (OLTP)

Τα συστήματα βάσεων δεδομένων όπως ο SQL Server είναι συστήματα που επεξεργάζονται πολλές συναλλαγές ταυτόχρονα (Online transaction processing -OLTP) όπως σε μία τράπεζα ή αεροπορική εταιρία, ή μεγάλο οργανισμό. Εστιάζουν συνήθως σε πολλαπλές εισαγωγές (insert) και ενημερώσεις (update) δεδομένων. Μπορούν επίσης να υπάρχουν και μερικές μικρές αναφορές (εκθέσεις σε καθημερινή βάση, π.χ. τα σύνολα πωλήσεων ημέρας). Συχνά, τα συστήματα αυτά χρησιμοποιούν φόρμες για την καταχώρηση ή αλλαγή δεδομένων και χειρίζονται μόνο μερικές έγγραφες ανά συναλλαγή. Δεν υπάρχει κανένα χαρακτηριστικό σύστημα OLTP. Κάθε εφαρμογή είναι διαφορετική. Αλλά εδώ είναι μια περιγραφή ενός τέτοιου συστήματος.

- 25,000,000 γραμμές από δεδομένα
- 200 χρήστες που μπορούν ταυτόχρονα να μπουν και να ανακτήσουν τα δεδομένα
- 2 λεπτά χρόνος απόκρισης στις περισσότερες συναλλαγές

1.2.3 Αποθήκες δεδομένων (data warehouses)

Οι αποθήκες δεδομένων εστιάζουν στην εισαγωγή και στην ανάλυση μεγάλων όγκων δεδομένων. Αυτά τα συστήματα εστιάζουν σε αναφορές που περιέχουν τα αποτελέσματα λεπτομερούς ανάλυσης των δεδομένων. Για αυτό και έχουν προϋπολογισμένες πολλές τιμές.

Πάλι, δεν υπάρχει κανένα χαρακτηριστικό σύστημα αποθήκης δεδομένων, αλλά εδώ υπάρχει μια περιγραφή ενός τέτοιου συστήματος

- 500,000,000 γραμμές από δεδομένα
- Μια αναφορά εκτελείται ανά χρονικό διάστημα που αναλύει τα δεδομένα
- Τρεις ώρες είναι ένας μέσος χρόνος για να εκτελεστεί μια αναφορά

1.3 ΕΙΣΑΓΩΓΗ ΣΤΙΣ ΑΠΟΘΗΚΕΣ ΔΕΔΟΜΕΝΩΝ

ΟΡΙΣΜΟΣ

Ακριβής ορισμός δεν υπάρχει καθώς συνεχώς εμφανίζονται νέοι. Ωστόσο κάνοντας ορισμένες παραδοχές θα μπορούσαμε να πούμε πως μια Αποθήκη Δεδομένων είναι μια μεγάλη Βάση Δεδομένων που:

- Χρησιμοποιείται συχνά ως βάση σε Συστήματα Στήριξης Αποφάσεων (DSS)
- Χρησιμοποιείται για συλλογή πληροφοριών. Επειδή από μια Α.Δ. δεν διαγράφουμε ποτέ τίποτα, αυτό έχει ως αποτέλεσμα να συγκεντρώνεται ένας μεγάλος όγκος πληροφοριών που αποτελούν ιστορικά στοιχεία του οργανισμού- επιχείρησης που κατέχει την Α.Δ.
- Συνδυάζει όλες τις πληροφορίες που κατέχει από διάφορες πηγές (OLTP, spreadsheets, web, κείμενα κ.λπ.) τα ελέγχει για ακρίβεια και τα οργανώνει έτσι ώστε οι χρήστες να ανακαλύπτουν εύκολα αυτό που επιθυμούν.

Συχνά οι Αποθήκες Δεδομένων τμηματοποιούνται σε άλλες μικρότερες εξειδικευμένες Α.Δ. τις λεγόμενες Data Marts (μικρές αποθήκες). Αυτές είναι συχνά προτιμότερες καθώς έχουν μικρότερο κόστος υλοποίησης και χρειάζονται μικρότερο χρονικό διάστημα για να τις κατασκευάσουμε.

Έτσι μπορεί μια επιχείρηση να έχει Data Marts για το τμήμα Προσωπικού, το τμήμα Μάρκετινγκ κ.τ.λ.

ΤΜΗΜΑΤΑ ΜΙΑΣ ΑΠΟΘΗΚΗΣ ΔΕΔΟΜΕΝΩΝ

Μια αποθήκη δεδομένων αποτελείται από διάφορα τμήματα, κάποια θα τα δούμε με περισσότερες λεπτομέρειες και παρακάτω.

- Data Marts
- Σχεσιακές Βάσεις Δεδομένων
- Τμήμα Προετοιμασίας Δεδομένων (Data Preparation Area)
- Υπηρεσίες Παρουσίασης Δεδομένων
- Εφαρμογές Ανάλυσης για τους τελικούς χρήστες

Συνήθως όλες οι πληροφορίες μιας αποθήκης δεδομένων αποθηκεύονται σε δομές μιας σχεσιακής βάσης δεδομένων

ΥΠΗΡΕΣΙΕΣ ΠΑΡΟΥΣΙΑΣΗΣ ΔΕΔΟΜΕΝΩΝ (PRESENTATION SERVICES)

Μια αποθήκη δεδομένων θα είναι χρήσιμη σε μια επιχείρηση αν έχει κατάλληλα εργαλεία που θα βοηθήσουν το χρήστη στη διαδικασία ανάλυσης και εκτίμησης των δεδομένων.

Ορισμένα από τα εργαλεία είναι:

- Απλές αναφορές: π.χ. αναφορές στο τέλος κάθε μήνα για το σύνολο των πωλήσεων
- Εξειδικευμένες εφαρμογές Data Mining :OLAP-MOLAP-ROLAP
- Κύβοι που περιέχουν πολυδιάστατους πίνακες: Στην ουσία είναι πίνακες που συνδυάζουν δεδομένα από διάφορους πίνακες για την καλύτερη ανάλυση κάποιων αποτελεσμάτων

π.χ. Το τμήμα Πωλήσεων παρατηρεί πως υπήρξε αύξηση των πωλήσεων των παγωτών τον Δεκέμβριο!! Ψάχνοντας μέσα στα δεδομένα του κύβου για το μήνα αυτό ανακαλύπτει πως υπήρχε μια ανεξήγητη (κατ' άλλα) άνοδος της θερμοκρασίας.

ΣΧΕΔΙΑΣΜΟΣ ΜΙΑΣ ΑΠΟΘΗΚΗΣ ΔΕΔΟΜΕΝΩΝ

Πριν προχωρήσει κάποιος στον σχεδιασμό και στην υλοποίηση μιας Α.Δ. οφείλει να λάβει υπόψη του ορισμένα πράγματα:

- Τα δεδομένα πρέπει να οργανωθούν με τέτοιο τρόπο ώστε να υπάρχει άμεση πρόσβαση στην πληροφορία που επιθυμεί ο χρήστης να αναλύσει.
- Τα δεδομένα όταν εισέρχονται στην Α.Δ. έχουν ήδη "καθαριστεί" και επαληθευτεί.

Για να δημιουργήσει κάποιος μια Α.Δ. θα πρέπει να σχεδιάσει πρώτα το σχεσιακό της μοντέλο.

Πρέπει να δημιουργηθούν πίνακες "γεγονότων" και πίνακες "διαστάσεων" και να τεθούν ευρετήρια σε όλα τα πεδία που αποτελούν κλειδιά αυτών των πινάκων.

Ορισμένα από τα σχεσιακά μοντέλα που χρησιμοποιούμε είναι τα παρακάτω:

- Star Schema: Αποτελείται από έναν πίνακα γεγονότων και πολλούς πίνακες διαστάσεων
- Snowflake Schema: Μπορούμε να πάρουμε από πολλούς πίνακες γεγονότων

Σε μερικές πιο πολύπλοκες Αποθήκες Δεδομένων μπορεί να υπάρχουν πολλαπλοί απλοί πίνακες και έναν αριθμό πινάκων διαστάσεων, κάποιοι από τους οποίους είναι τμήματα πολλών απλών και άλλοι τμήμα ενός μόνο. Τι σημαίνει όμως αυτό:

Έστω ότι έχουμε δεδομένα τόσο για τις πωλήσεις προϊόντων όσο και για τα στοιχεία απογραφής τους. Από τη φύση τους τα δεδομένα αυτά είναι διαφορετικά , θα έπρεπε λοιπόν να είναι υποθηκευμένα σε διαφορετικούς πίνακες. Ωστόσο ορισμένοι πίνακες , όπως ο πίνακας διαστάσεων "Προϊόν" θα μπορούσε να περιέχει και τους δύο (πιο πάνω πίνακες) ενώ κάποιοι άλλοι όπως ο πίνακας "Αποθήκη" να περιέχει μόνον τον έναν.

ΚΑΘΑΡΙΣΜΟΣ-ΜΕΤΑΣΧΗΜΑΤΙΣΜΟΣ ΔΕΔΟΜΕΝΩΝ

Πριν να εισαχθούν τα δεδομένα στην Α.Δ. πρέπει να περάσουν από ορισμένες διαδικασίες. Εδώ είναι το Data Preparation Area. Οι διαδικασίες εκτελούνται για να:

- Ελεγχθεί η αξιοπιστία και να διαπιστωθεί η συνέπεια των δεδομένων.
- Μετασχηματισθούν τα δεδομένα σε συνήθη formats.
- Συνδυαστούν πεδία πολλών ονομάτων σε ένα μόνο π.χ έχουμε πεδία(Όνομα, επίθετο) και δημιουργούμε ένα μόνο Ονοματεπώνυμο.
- Διαχωριστούν πεδία ημερομηνιών.

ΜΕΤΑΦΟΡΑ ΔΕΔΟΜΕΝΩΝ ΣΤΗΝ ΑΠΟΘΗΚΗ ΔΕΔΟΜΕΝΩΝ

Τα δεδομένα "φορτώνονται" στην Α.Δ. από πολλές πηγές και αφ' ότου όπως έχει ήδη ειπωθεί "καθαριστούν". Αφού λοιπόν έχουν εισαχθεί τα δεδομένα πρέπει να γίνει επαλήθευση τους ανάμεσα στους πίνακες διαστάσεων και στους πίνακες γεγονότων (referential integrity) δηλ., όλες οι εγγραφές να σχετίζονται με τις κατάλληλες εγγραφές στους κατάλληλους πίνακες.

Επιπλέον θα πρέπει να επαληθευτεί πως κάθε εγγραφή των πινάκων γεγονότων σχετίζεται με μία εγγραφή στον κάθε πίνακα διαστάσεων που περιλαμβάνει ο κύβος.

π.χ ΠροϊόνΠωλήσεις (fact table) συσχετίζεται σε κάποιο κύβο με τις διαστάσεις Πελάτες, Προϊόν, πελάτες, καταστήματα.

Το αντίστροφο δεν είναι ανάγκη να ισχύει.

ΧΡΗΣΙΜΟΠΟΙΩΝΤΑΣ ΜΙΑ ΑΠΟΘΗΚΗ ΔΕΔΟΜΕΝΩΝ

Συνεχώς εξελίσσονται νέοι τρόποι για το πως χρησιμοποιεί και αναλύσει κανείς τα δεδομένα μιας Αποθήκης Δεδομένων. Ακολουθούν μερικοί:

Sql queries
Olap and Data Mining
Ms Office
Offline Olap Cubes
Web agents
Hypertext analysis and transformation
Information visualization
Data marts

ΔΙΑΤΗΡΩΝΤΑΣ ΜΙΑ ΑΠΟΘΗΚΗ ΔΕΔΟΜΕΝΩΝ

ΑΝΑΝΕΩΝΟΝΤΑΣ ΤΑ ΔΕΔΟΜΕΝΑ

Είναι μια διαδικασία σχεδόν εφάμιλλη αυτής της αρχικής εισαγωγής των δεδομένων στη βάση, μόνο που είναι λιγότερο πολύπλοκη. Συνήθως γίνεται μια φορά το μήνα, βδομάδα ή μέρα ανάλογα με τον όγκο των πληροφοριών.

Ανανεώνοντας τα δεδομένα δεν σημαίνει όμως πως αυτόματα ανανεώνονται και οι OLAP cubes (αυτό γίνεται μόνο όταν επεξεργαστούν ξανά (reprocessed)).

1.4 ΠΑΡΟΥΣΙΑΣΗ ΚΑΙ ΣΥΓΚΡΙΣΗ OLTP – OLAP

1.4.1 Συστήματα άμεσης επεξεργασίας συναλλαγών/δοσοληψιών OLTP (on-line transaction processing)

Κύριο έργο των παραδοσιακών σχεσιακών Βάσεων Δεδομένων. Είναι τα συστήματα για προγραμματισμό εφαρμογών, εκτέλεση και διαχείριση των δοσοληψιών. Για παράδειγμα ας εξετάσουμε μια δοσοληψία με μια δανειστική βιβλιοθήκη. Έστω ότι ένας αναγνώστης επιστρέφει ένα αντίτυπο από ένα βιβλίο Α και δανείζεται ένα αντίτυπο από ένα άλλο βιβλίο Β. Εάν θέλουμε να παρουσιάσουμε κωδικοποιημένη την δοσοληψία θα γράψουμε:

Αρχή Συναλλαγής

Διάβασε Αντίτυπα Α

Αντίτυπα Α Αντίτυπα Α+1

Διάβασε Αντίτυπα Β

Αντίτυπα Β Αντίτυπα Β-1

Τέλος Συναλλαγής (επικύρωση/απόρριψη)

Λειτουργικά χαρακτηριστικά επεξεργασίας συναλλαγών

- Κύριο έργο παραδοσιακών σχεσιακών Βάσεων Δεδομένων
- Αφορούν καθημερινές λειτουργίες
- Πρέπει να αποκρίνεται σε ελάχιστο χρόνο, να επανακάμπτει άμεσα και να δουλεύει συνεχώς (Οι OLTP εφαρμογές, σε πολλούς οργανισμούς πρέπει να είναι διαθέσιμες 24 ώρες επί 7 ημέρες την εβδομάδα όπως στις τράπεζες, αεροπορικές εταιρίες)
- Τράπεζες: Πρόσβαση στη Βάση Δεδομένων από ταμεία, δάνεια, κάρτες, ATM's, Κλπ

- • Αεροπορικές εταιρίες: κρατήσεις από διαφορετικά σημεία, πόσοι ταξιδεύουν?
- • Θεμελιώδης για τη λειτουργία ενός οργανισμού
- • Περιορισμένος αριθμός υπολογισμών
- • Ελάχιστος χρόνος διαθέσιμος για την εκτέλεση μιας δοσοληψίας
- • Λιγότερες από 10 προσβάσεις δίσκου.
- • Κατώτατο όριο λειτουργικών απαιτήσεων: 100 on-line Transactions Per Second (TPS) σε μια Βάση Δεδομένων την τάξης του 1 GB
- • Ανώτατο όριο λειτουργικών απαιτήσεων: 50000 TPS σε μια Βάση Δεδομένων μεγαλύτερη του 1 TB

1.4.2 Συστήματα άμεσης αναλυτικής επεξεργασίας δεδομένων OLAP (on-line analytical processing)

Η ΕΦΑΡΜΟΓΗ ΜΑΣ.

Ένα σύστημα Άμεσης Αναλυτικής Επεξεργασίας (OLAP) είναι ένα σύστημα που μας επιτρέπει να βλέπουμε μια διαφορετική σύνοψη πολυδιάστατων δεδομένων. Η λέξη online δηλώνει ότι δεν περιμένουμε για μεγάλο χρονικό διάστημα να δούμε τα αποτελέσματα ενός συγκεντρωτικού ερωτήματος αλλά μας εμφανίζονται άμεσα.

Λειτουργικά χαρακτηριστικά αναλυτικής επεξεργασίας

- Κύριο έργο σε Αποθήκες Δεδομένων
- Ανάλυση δεδομένων και λήψη αποφάσεων
- Πρόσβαση σε μεγάλο όγκο δεδομένων από διαφορετικές πηγές
- Γρήγορη απάντηση σε οποιαδήποτε χρονική στιγμή τεθεί ένα ερώτημα (γι' αυτό και On-Line)
- Μεταβολή της οπτικής γωνιάς παρουσίασης των δεδομένων(π.χ από πωλήσεις ανά περιοχή σε πωλήσεις ανά τμήμα)
- Απαντήσεις σε πολύπλοκες ερωτήσεις όπως:
    1. Ποιος ήταν ο όγκος πωλήσεων ανά περιοχή και κατηγορία προϊόντος την περασμένη χρονιά;
    2. Πόσο σχετίζονται οι αυξήσεις τιμών των υπολογιστών με τα κερδών των πωλήσεων τα 10 τελευταία χρόνια;
    3. Ποια ήταν τα δέκα πρώτα καταστήματα σε πωλήσεις CD;
    4. Πόσους δίσκους πουλήσαμε στην Δυτική Περιφέρεια το τελευταίο τέταρτο της περσινής χρονιάς σε καταστήματα με κατανάλωση μεγαλύτερη από 100 δίσκους μηνιαίως, και ποιο το κέρδος μας από αυτές τις πωλήσεις;
    5. Τι ποσοστό από τους πελάτες που αγοράζουν αναψυκτικά, αγοράζουν και πατατάκια;

Στην εφαρμογή μας δεν κλειδώνονται όλες οι εγγραφές των τεσσάρων βάσεων κατανεμημένων στις τέσσερις τοποθεσίες μέχρι να ολοκληρωθεί το ερώτημα/συναλλαγή για τις συνοπτικές τιμές των πωλήσεων επειδή έχουμε προϋπολογίσει τα μερικά αθροίσματα πωλήσεων ανά παραγγελία, και τα έχουμε αποθηκεύσει ξανά στους τέσσερις πίνακες SALES και στους τέσσερις πίνακες DAYS.

Άρα έχουμε τέσσερα μικρά υβριδικά (HOLAP) συστήματα αποθήκης δεδομένων. Δηλαδή τέσσερις μικρές αποθήκες δεδομένων με σχήμα αστέρα που ενημερώνονται κατά τακτά χρονικά διαστήματα. Αυτό αυξάνει την ταχύτητα και ελαττώνει τον συνολικό αποθηκευτικό χώρο.

1.4.3 Διαφορές OLTP – OLAP

	OLTP	OLAP
Χρήστες	Χειριστής, επαγγελματίας	Αναλυτής δεδομένων/λήπτης αποφάσεων
Λειτουργία	Καθημερινή	Υποστήριξη αποφάσεων
Σχεδιασμός ΒΔ	Προσανατολισμένο στην εφαρμογή	Προσανατολισμένο στο θέμα (π.χ πωλήσεις)
Δεδομένα	Τρέχοντα, λεπτομερώς ενημερωμένα, σχεσιακά, μη-ενοποιημένα	Ιστορικά, περιληπτικά, πολυδιάστατα, ενοποιημένα, συναθροισμένα
Χρήση	Επαναληπτική	Όποτε χρειάζεται
Αριθμός χρηστών	Χιλιάδες	Δεκάδες
Πρόσβαση	Διάβασμα/ εγγραφή	Διάβασμα
Μονάδα εργασίας	Σύντομη, απλή συναλλαγή	Πολύπλοκες ερωτήσεις
Εγγραφές	Δεκάδες	Εκατομμύρια
Πρόσβαση χρηστών	Χιλιάδες	Εκατοντάδες
Μέγεθος ΒΔ	100 MB-GB	100 GB-TB
Φύση δεδομένων	Δυναμικά, Τρέχοντα	Στατιστικά, Ιστορικά
Μέτρηση απόδοσης	Επίπεδο συναλλαγής	Χρόνος απόκρισης

Υπάρχουν ορισμένες διαφορές που θα επισημάνουμε ανάμεσα στα δεδομένα που αποθηκεύονται σε μια βάση δεδομένων και στα δεδομένα που αποθηκεύονται σε μια αποθήκη δεδομένων.

Οι εφαρμογές σε μία αποθήκη δεδομένων σχετίζονται απευθείας με τις επιχειρησιακές αποφάσεις και τις αναλύσεις των δεδομένων, την άμεση αναλυτική επεξεργασία (Online Analytical Processing - OLAP). Μία αποθήκη περιέχει ιστορικά δεδομένα. Τα δεδομένα σε μία αποθήκη ενημερώνονται σε εβδομαδιαία ή μηνιαία βάση και δεν τροποποιούνται τόσο συχνά όπως συμβαίνει σε μία βάση δεδομένων. Παρόλο που αυτό σημαίνει ότι η αποθήκη δεν είναι πλήρως ενημερωμένη, αυτό συνήθως δεν αποτελεί πρόβλημα σε εφαρμογές υποστήριξης αποφάσεων.

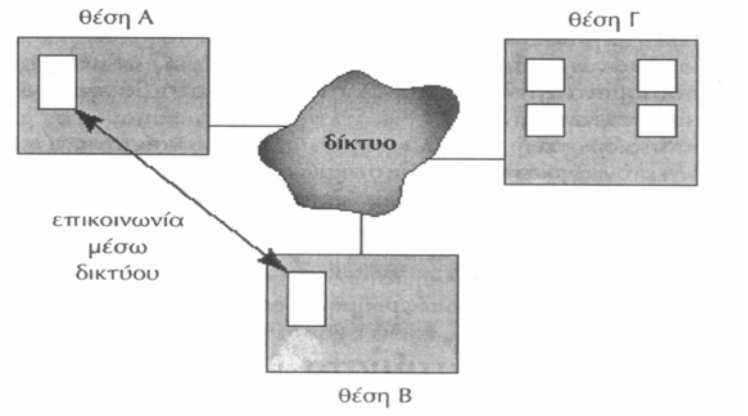
1.4.4 Συνεργασία Αποθήκης δεδομένων και OLAP

Οι πηγές των δεδομένων και των πληροφοριών μιας OLAP εφαρμογής μπορεί να είναι διαχειριστικές βάσεις δεδομένων, αποθήκες δεδομένων καθώς και πηγές έξω από την επιχείρηση. Η ανάγκη όμως για πολυδιάστατη ανάλυση αναδεικνύει τις αποθήκες δεδομένων σαν τη κύρια πηγή άντλησης των πληροφοριών. Οι αποθήκες δεδομένων περιέχουν συγκεντρωτικά και συνοπτικά, όλα τα απαραίτητα δεδομένα για τη δημιουργία των πολυδιάστατων χώρων που χρησιμοποιεί η απευθείας αναλυτική διαδικασία. Οι επιχειρήσεις με την εφαρμογή και εγκατάσταση μιας αποθήκης δεδομένων, ταυτόχρονα εφαρμόζουν τεχνικές OLAP οι οποίες συνεργάζονται άμεσα με αυτές.

1.5 Κατανεμημένα Συστήματα Βάσεων Δεδομένων

Στα κατανεμημένα συστήματα βάσεων δεδομένων, η βάση δεδομένων αποθηκεύεται σε διάφορους υπολογιστές που επικοινωνούν ο ένας με τον άλλον μέσω διαφόρων μέσων επικοινωνίας, όπως δίκτυα υψηλής ταχύτητας, τηλεφωνικές γραμμές ή τοπικά δίκτυα. Δεν μοιράζονται κύρια μνήμη ή δίσκους. Οι υπολογιστές σε ένα κατανεμημένο σύστημα μπορεί να διαφέρουν σε μέγεθος και λειτουργία και μπορεί να είναι από απλοί υπολογιστές (PC) μέχρι μεγάλοι υπολογιστές (Server).

Οι υπολογιστές σε ένα κατανεμημένο σύστημα αναφέρονται με διαφορετικά ονόματα, όπως θέσεις ή κόμβους, ανάλογα με το περιβάλλον στο οποίο αναφέρονται. Κυρίως χρησιμοποιούμε τον όρο θέση για να δώσουμε έμφαση στη φυσική κατανομή αυτών των συστημάτων. Η γενική δομή ενός κατανεμημένου συστήματος φαίνεται στην Εικόνα.



Ένα κατανεμημένο σύστημα.

1.6 Κατανεμημένη Ετερογενής Επεξεργασία Ερωτημάτων

Η κατανεμημένη ετερογενής δυνατότητα ερωτημάτων του SQL Server επιτρέπει ερωτήματα συναλλαγών και ενημερώσεις ως προς μια ποικιλία σχεσιακών και μη σχεσιακών πηγών μέσω OLE-DB, που τρέχουν σε ένα ή περισσότερους υπολογιστές.

Μια πηγή OLE-DB εγγράφεται στον SQL Server ως ένας συνδεδεμένος διακομιστής. Για να δημιουργηθεί ένας συνδεδεμένος διακομιστής, θα πρέπει να οριστούν τέσσερα τμήματα: το όνομα του συνδεδεμένου διακομιστής, κατάλογος, το όνομα του παροχέα, το όνομα του Data Source. Επιπλέον, ο SQL Server υποστηρίζει συναρτήσεις με τιμές πίνακα, που ονομάζονται openquery και επιτρέπουν την αποστολή μη μεταφρασμένων ερωτημάτων σε ένα παροχέα ή συνδεδεμένο διακομιστή, αντίστοιχα, στη διάλεκτο που υποστηρίζεται από τον παροχέα. Το παρακάτω ερώτημα συνδυάζει πληροφορίες αποθηκευμένες σε ένα διακομιστή MySQL και σε έναν της Oracle .

```
SELECT *
 FROM OPENQUERY (MYSQL, 'SELECT * FROM CUSTOMERS') ";
   UNION
SELECT *
 FROM OPENQUERY (ORACLE, 'SELECT * FROM CUSTOMERS') ";
```

1.7 Δημιουργία ενός Partitioned View στον SQL Server

Γενικά η κατάτμηση ενός πίνακα σε πολλούς μικρότερους μπορεί να είναι οριζόντια ή κατακόρυφη. Μια προβολή θα ξαναενώσει όλα τα δεδομένα.

Μια κατατμημένη προβολή (Partitioned View) ενώνει οριζόντιος τα δεδομένα από ένα σύνολο πινάκων διαμέσου ενός ή περισσοτέρων συνδεδεμένων διακομιστών, παρουσιάζοντας τα, σαν να προέρχονται από έναν πίνακα. Ο SQL Server ξεχωρίζει την διαφορά μεταξύ τοπικών και κατανεμημένων κατατμημένων προβολών. Στην τοπική κατατμημένη προβολή (local partitioned view), όλοι οι συμμετέχοντες πίνακες ανήκουν στον ίδιο SQL Server. Στην κατανεμημένη κατατμημένη προβολή (distributed partitioned view), τουλάχιστον ένας εκ των συμμετεχόντων πινάκων βρίσκεται σε διαφορετικό διακομιστή. Επιπλέον ο SQL Server 2000 διακρίνετε μεταξύ των κατατμημένων προβολών που ενημερώνουν τους πίνακες και αυτών που απλώς διαβάζουν τα αντίγραφα των πινάκων.

Η κατανεμημένη κατατμημένη προβολή μπορεί να χρησιμοποιηθεί για να συνδέσει ένα σύνολο από διαφορετικούς διακομιστές. Το σύνολο αυτό είναι ένα γκρουπ από διακομιστές που ο καθένας διαχειρίζονται ανεξάρτητα τα δεδομένα αλλά μπορούν και να συνεργαστούν για να μοιράσουν το επεξεργαστικό φορτίο του συστήματος.

Πριν εφαρμόσουμε μια κατατμημένη προβολή, πρέπει να χωρίσουμε τον πίνακα οριζοντίος. Ο αρχικός πίνακας αντικαθίστατο από ένα σύνολο μικρότερων πινάκων, όπου κάθε ένας από αυτούς έχει τον ίδιο αριθμό στηλών με τον αρχικό και κάθε στήλη έχει τα ίδια χαρακτηριστικά (όπως τύπο δεδομένων, μέγεθος) με την αντίστοιχη στήλη του αρχικού πίνακα. Για να δημιουργήσουμε μια διανεμημένη κατατμημένη προβολή κάθε πίνακας πρέπει να βρίσκεται σε ξεχωριστό διακομιστή. Το όνομα της βάσης δεδομένων μπορεί να είναι ίδιο σε κάθε διακομιστή, ωστόσο αυτό δεν είναι απαραίτητο.

Σχεδιάζουμε τους πίνακες έτσι ώστε κάθε πίνακας να αποθηκεύει μια οριζόντια γραμμή του αρχικού πίνακα βασισμένο στο πρωτεύων κλειδί. Τα πρωτεύων κλειδιά βασίζονται στην αναφορά των δεδομένων (data values) . Ο ορισμός του πρωτεύοντος κλειδιού για κάθε πίνακα είναι επιβεβλημένος από έναν περιορισμό CHECK και δεν μπορούν να επικαλυφθεί. Για παράδειγμα δεν μπορεί να έχετε έναν πίνακα με εγγραφές από 1 έως 200000 και έναν άλλο με εγγραφές από 150000 ως 300000 διότι δεν θα είναι ξεκάθαρο ποιος πίνακας περιέχει τις εγγραφές από 150000 έως 200000. Παράδειγμα:

στον Server1:

```
CREATE TABLE Customer_33
        (CustomerID INTEGER PRIMARY KEY
                CHECK(CustomerID BETWEEN 1 AND 32999) ,……)
```

στον Server2:

```
CREATE TABLE Customer_66
        (CustomerID INTEGER PRIMARY KEY
                CHECK(CustomerID BETWEEN 33000 AND 65999) ,….)
```

Στον Server3:

```
CREATE TABLE Customer_99
        (CustomerID INTEGER PRIMARY KEY
                CHECK(CustomerID BETWEEN 66000 AND 99999) ,…..)
```

Αφού δημιουργηθούν οι πίνακες καθορίζεται μια κατανεμημένη κατατμημένη προβολή για κάθε διακομιστή με κάθε προβολή να έχει το ίδιο όνομα. Αυτό επιτρέπει στα ερωτήματα που αναφέρονται στο όνομα της προβολής να εκτελούνται σε κάθε διακομιστή. Το σύστημα λειτουργεί σαν να υπάρχει ένα αντίγραφο του αρχικού πίνακα σε κάθε διακομιστή, αλλά κάθε διακομιστής έχει μόνο ένα πίνακα . Η τοποθεσία των δεδομένων είναι ορατή στην εφαρμογή.

Για κάθε συνδεδεμένο διακομιστή προσθέτουμε τα ορίσματα που περιέχουν τις πληροφορίες σύνδεσης που χρειάζονται τα διανεμημένα ερωτήματα για να εκτελεστούν. Αυτό δίνει πρόσβαση στις διανεμημένες κατατμημένες προβολείς στα δεδομένα των άλλων πινάκων

Καθορίζουμε τις παραμέτρους για κάθε συνδεδεμένο διακομιστή που χρησιμοποιείτε στη διανεμημένη κατατμημένη προβολή. Αυτό βελτιστοποιεί την απόδοση εξασφαλίζοντας ότι ο επεξεργαστής ερωτημάτων δεν θα ζητήσει άλλα στοιχεία για κάθε ένα από τους συνδεδεμένους πίνακες μέχρι να πάρουμε τα δεδομένα που θέλουμε από τους απομακρυσμένους πίνακες.

Δημιουργώντας διανεμημένη κατατμημένη προβολή για κάθε διακομιστή , οι προβολείς χρησιμοποιούν διανεμημένες καταστάσεις SELECT για να έχουν πρόσβαση στα δεδομένα ενός συνδεδεμένου διακομιστή και συγχωνεύουν τις διανεμημένες γραμμές με γραμμές από τον τοπικό πίνακα.

Έχουμε προσθέσει έναν linked-server με όνομα Server2 με πληροφορίες σύνδεσης για το Server2, και έναν linked-server με όνομα Server3 για πρόσβαση στον Server3.

Δημιουργία μιας διανεμημένης κατατμημένης προβολής για το προαναφερόμενο παράδειγμα:

```
CREATE VIEW Customers AS
    SELECT * FROM CompanyDatabase.TableOwer.Customers_33
UNION ALL
    SELECT * FROM  Server2.CompanyDatabase.TableOwer.Customers_66
UNION ALL
    SELECT * FROM  Server3.CompanyDatabase.TableOwer.Customers_99
```

1.8 ΣΧΕΔΙΑΣΜΟΣ ΣΥΣΤΗΜΑΤΟΣ ΔΙΑΧΕΙΡΙΣΗΣ ΠΑΡΑΓΓΕΛΙΩΝ

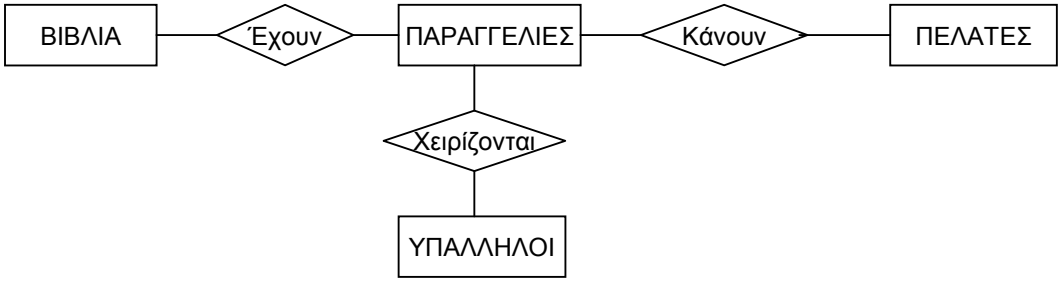
Για να κατανοήσουμε μία βάση διαχείρισης παραγγελιών όπως για παράδειγμα το δείγμα της Northwind που εγκαθίσταται με τον SQL Server το οποίο έχουμε σπάσει σε τέσσερα κομμάτια από τα οποία παίρνουμε τις χιλιάδες καταχωρήσεις παραγγελιών για να τα εισάγουμε στις τέσσερις μικρές αποθήκες δεδομένων κατανεμημένες στις 4 τοποθεσίες και να τα αναλύσουμε με την εφαρμογή μας θα αναφερθούμε πρώτα στον σχεδιασμό της που είναι σχετικά προφανής αν ανατρέξουμε στις σημειώσεις των εργαστηρίων βάσεων δεδομένων.

Ας αναπαράγουμε ακριβώς το μοντέλο ER και τη μετατροπή του σε σχεσιακό μοντέλο από την άσκηση 'Βιβλιοπωλείο' των εργαστηρίων βάσεων δεδομένων.

Το κύριο τμήμα της βάσης Βιβλιοπωλείο απορρέει από την διαπίστωση ότι σε ένα Βιβλιοπωλείο οι **πελάτες** θα κάνουν, απευθυνόμενοι σε **υπαλλήλους** παραγγελίες για **βιβλία**. Η παρακάτω εικόνα δείχνει το διάγραμμα ER.

Εικόνα από την άσκηση Βιβλιοπωλείο. Διάγραμμα οντοτήτων-συσχετίσεων (ορθογώνια οι οντότητες, ρόμβοι οι συσχετίσεις). Έχουμε τρεις οντότητες και μία συσχέτιση. Πως συσχετίζονται οι πελάτες με τα βιβλία και τους υπαλλήλους; με την **τριαδική συσχέτιση παραγγελίες**.

Η αρχική τριαδική συσχέτιση 'Παραγγελίες' θα μετατραπεί σε τύπο οντότητας με δικό της κλειδί και τρεις δυαδικές συσχετίσεις όπως παρακάτω.



Εικόνα. Μετατροπή της τριαδικής συσχέτισης παραγγελίες σε τρεις δυαδικές. Κάθε τύπος οντότητας (τα ορθογώνιο) θα έχει το δικό του κλειδί.

Αν αντικαταστήσουμε τώρα τα βιβλία με προϊόντα γενικού είδους έχουμε τη βάση για κάθε είδους σύστημα διαχείρισης παραγγελιών.

Το μόνο που απομένει είναι να προσδιορίσουμε τον λόγο πληθικότητας των τύπων συσχετίσεων (οι ρόμβοι).

Ο λόγος πληθικότητας (cardinality) ενός δυαδικού τύπου συσχετίσεων ορίζει το πόσες οντότητες από τον πρώτο τύπο οντοτήτων στην συσχέτιση μπορούν να συνδεθούν με τις οντότητες από τον δεύτερο τύπο.

(ένα με ένα) 1 : 1

(ένα με πολλά) 1 : N

(πολλά με ένα) N : 1

(πολλά με πολλά) N : M

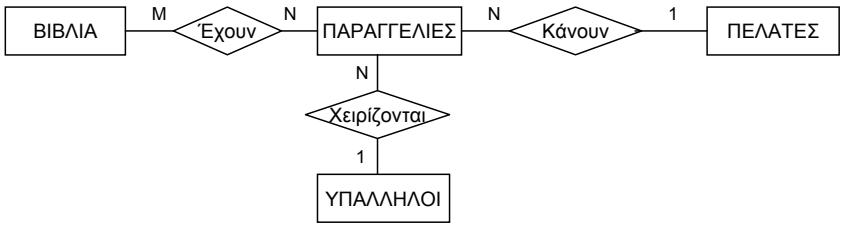
Στην ουσία σημαίνει πόσες εγγραφές από την μία πλευρά της δυαδικής συσχέτισης μπορούν να συνδεθούν με τις εγγραφές τις άλλης πλευράς.

Με απλές ερωταποκρίσεις:

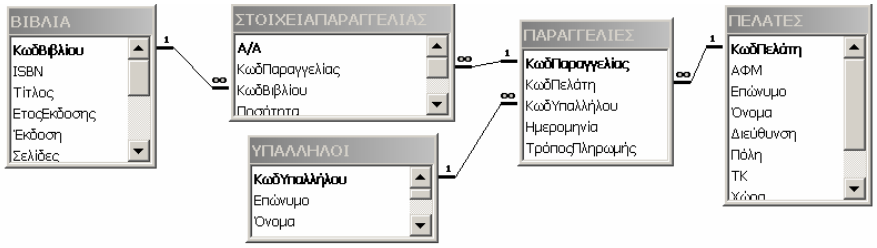
Ένας πελάτης κάνει πολλές παραγγελίες αλλά μία παραγγελία γίνεται από έναν μόνο πελάτη. Άρα η συσχέτιση πελάτες-κάνουν-παραγγελίες είναι ένα προς πολλά.

Ένα βιβλίο μπορεί να περιέχεται σε πολλές παραγγελίες και μία παραγγελία μπορεί να περιέχει πολλά βιβλία. Άρα η συσχέτιση παραγγελίες-έχουν-βιβλία είναι πολλά προς πολλά.

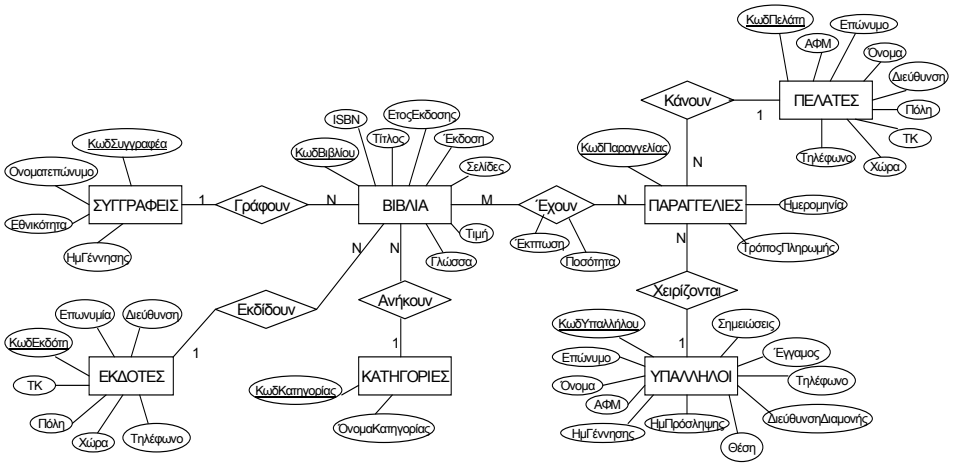
Ένας υπάλληλος μπορεί να χειριστεί πολλές παραγγελίες αλλά μία παραγγελία την χειρίζεται ένας μόνο υπάλληλος. Άρα η συσχέτιση υπάλληλοι-χειρίζονται-παραγγελίες είναι ένα προς πολλά, και έτσι προκύπτει το παρακάτω διάγραμμα.



Αυτό το διάγραμμα μας δείχνει ότι θα πρέπει, μετατρέποντας το στο σχεσιακό μοντέλο, να δημιουργήσουμε 4 πίνακες για τις τέσσερις οντότητες και έναν επιπλέον πίνακα για την συσχέτιση παραγγελίες-έχουν-βιβλία επειδή είναι πολλά προς πολλά, που στα εργαστήρια τον ονομάζουμε Στοιχεία Παραγγελίας.



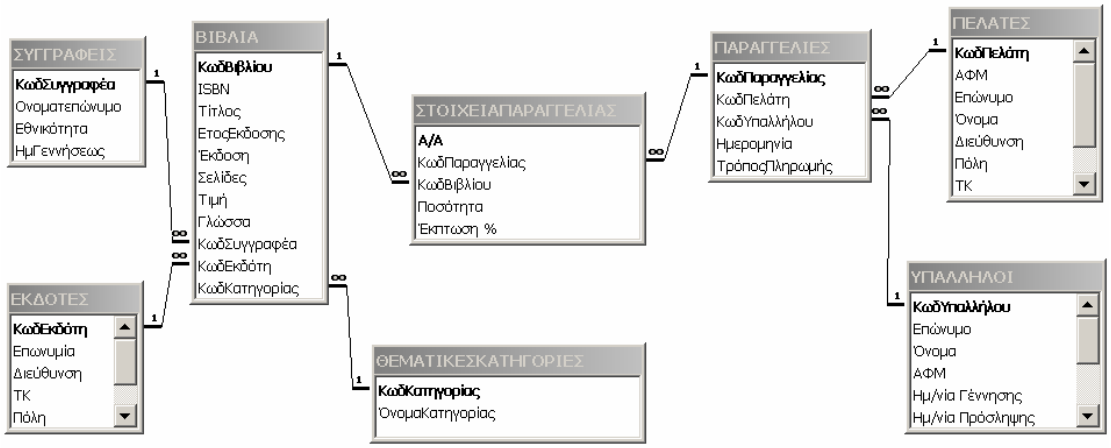
Εικόνα από την άσκηση βιβλιοπωλείο. Κάθε σύστημα διαχείρισης παραγγελιών θα έχει σαν κορμό τους πίνακες Πελάτες, Παραγγελίες, Στοιχεία Παραγγελίας, Προϊόντα.



Εικόνα από την άσκηση βιβλιοπωλείο. Το τελικό διάγραμμα Οντοτήτων-συσχετίσεων-γνωρισμάτων. Εδώ έχουμε καθορίσει επιπλέον ότι σε μία κατηγορία ανήκουν πολλά βιβλία, ότι ένας εκδότης εκδίδει πολλά βιβλία και ότι ένας συγγραφέας γράφει πολλά βιβλία. Η τελευταία συσχέτιση συγγραφέας-γράφει-βιβλία θα μπορούσε να ήταν πολλά-προς-πολλά για

να καλύψει και την περίπτωση των συν-συγγραφέων, αλλά εδώ μας αρκεί η καταγραφή μόνο του πρώτου ονόματος.

Η μετατροπή του παραπάνω διαγράμματος ER στο σχεσιακό μοντέλο είναι εύκολη. Για κάθε οντότητα δημιουργούμε πίνακα, για κάθε δυαδική συσχέτιση ένα προς πολλά προσθέτουμε το πρωτεύον κλειδί της πλευράς του ένα στην πλευρά του πολλά, όπου γίνεται ξένο κλειδί, και για κάθε δυαδική συσχέτιση πολλά-προς-πολλά δημιουργούμε επιπλέον πίνακα με ξένα κλειδιά τα πρωτεύοντα κλειδιά των δύο επιμέρους οντοτήτων.



Εικόνα από την άσκηση βιβλιοπωλείο. Το σχεσιακό μοντέλο με το διάγραμμα περιορισμών αναφορικής ακεραιότητας των ξένων κλειδιών. Η συνολική δομή της βάσης αποτελείται από κανονικοποιημένους πίνακες (σχέσεις). Το διάγραμμα περιορισμών αναφορικής ακεραιότητας δείχνει από που θα παίρνουν τις τιμές τους τα ξένα κλειδιά.

Πως συνδέεται μία εγγραφή από έναν πίνακα με μία εγγραφή από έναν άλλο πίνακα (π.χ. η παραγγελία με ΚωδΠαραγγελίας 1 με τον πελάτη με ΚωδΠελάτη 4); Με ένα **ζεύγος τιμών** των πρωτευόντων κλειδιών τους {1, 4}.

Μία παραγγελία συνδέεται με έναν πελάτη με ένα ζεύγος τιμών
{ΚωδΠαραγγελίας, ΚωδΠελάτη} που αποθηκεύεται στον πίνακα Παραγγελίες.

Μία παραγγελία συνδέεται με έναν υπάλληλο με ένα ζεύγος τιμών
{ΚωδΠαραγγελίας, ΚωδΥπαλλήλου} που αποθηκεύεται στον πίνακα Παραγγελίες.

Μία παραγγελία συνδέεται με ένα βιβλίο με ένα ζεύγος τιμών
{ΚωδΒιβλίου, ΚωδΠαραγγελίας} που αποθηκεύεται στον πίνακα ΣτοιχείαΠαραγγελίας.

Ένα βιβλίο συνδέεται με έναν συγγραφέα με ένα ζεύγος τιμών
{ΚωδΒιβλίου, ΚωδΣυγγραφέα} που αποθηκεύεται στον πίνακα Βιβλία.

Ένα βιβλίο συνδέεται με έναν Εκδότη με ένα ζεύγος τιμών
{ΚωδΒιβλίου, ΚωδΕκδότη} που αποθηκεύεται στον πίνακα Βιβλία.

Ένα βιβλίο συνδέεται με μία Θεματική Κατηγορία με ένα ζεύγος τιμών
{ΚωδΒιβλίου, ΚωδΚατηγορίας} που αποθηκεύεται στον πίνακα Βιβλία.

Αυτά τα **ζεύγη τιμών** {τιμήΑ, τιμήΒ} που στην ουσία το κάθε ένα συνδέει ένα μέλος του συνόλου Α με ένα μέλος του συνόλου Β ονομάζονται **σχέσεις** και οι βάσεις δεδομένων σχεσιακές βάσεις δεδομένων. Αν είχαμε ένα πεδίο ορισμού x (1, 2, 3) και ένα πεδίο τιμών y (3, 6, 9) και θέλαμε να ορίσουμε την σταθερή στο χρόνο σχέση μεταξύ τους (1|3, 2|6, 3|9) τότε αυτή θα ονομάζονταν συνάρτηση (y=f(x)=3*x). Το πρώτο χαρακτηριστικό των σχέσεων είναι ότι μεταβάλλονται με το χρόνο (εισαγωγές, διαγραφές, ενημερώσεις). Το δεύτερο απορρέει από τον ορισμό του C. Codd για μία σχέση. "Σχέση: Ένα υποσύνολο του

καρτεσιανού γινομένου των υποκείμενων πεδίων ορισμού". Άρα δεν είναι δυνατόν να επιτρέπεται να αποθηκευτεί στη σχέση {ΚωδΠαραγγελίας, ΚωδΠελάτη} ένας κωδικός πελάτη που δεν υπάρχει στον πίνακα πελάτες γι' αυτό το λόγο υπάρχουν οι περιορισμοί ακεραιότητας αναφορών των τιμών αυτών σε όποιο πίνακα είναι ξένα κλειδιά.

Πως συνδέονται δύο Πίνακες; Η σύνδεση δύο πινάκων (σχέσεων) επιτυγχάνεται με τα **κοινά πεδία τιμών τους** (πρωτεύον κλειδί και ξένο κλειδί). Μία σύνδεση γίνεται έτσι μέσω των κοινών τιμών (π.χ. ΚωδΠελάτη) που ανήκουν και στους δύο πίνακες. Μπορούμε έτσι να εφαρμόσουμε τη σύζευξη για να ενώσουμε τους δύο πίνακες. Άρα σε τι χρησιμεύουν τα ξένα κλειδιά; "Δείχνουν" τις συσχετιζόμενες εγγραφές.

Ένα ερώτημα σύζευξης (συνθήκη σύζευξης είναι η ξένοκλειδί=πρωτεύονκλειδί) SQL:

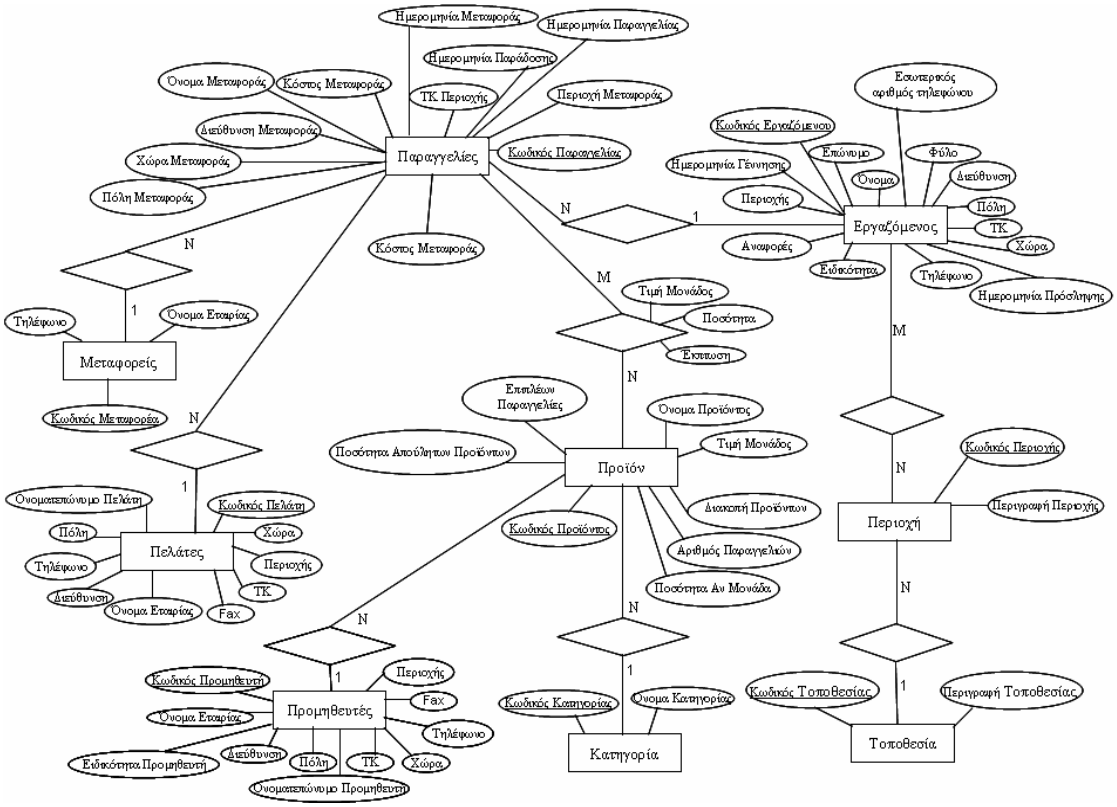
Select Πελάτες.Επώνυμο, Πελάτες.Όνομα, Παραγγελίες.ΚωδΠαραγγελίας

From Πελάτες **inner join** Παραγγελίες

on Πελάτες.ΚωδΠελάτη=Παραγγελίες.ΚωδΠελάτη

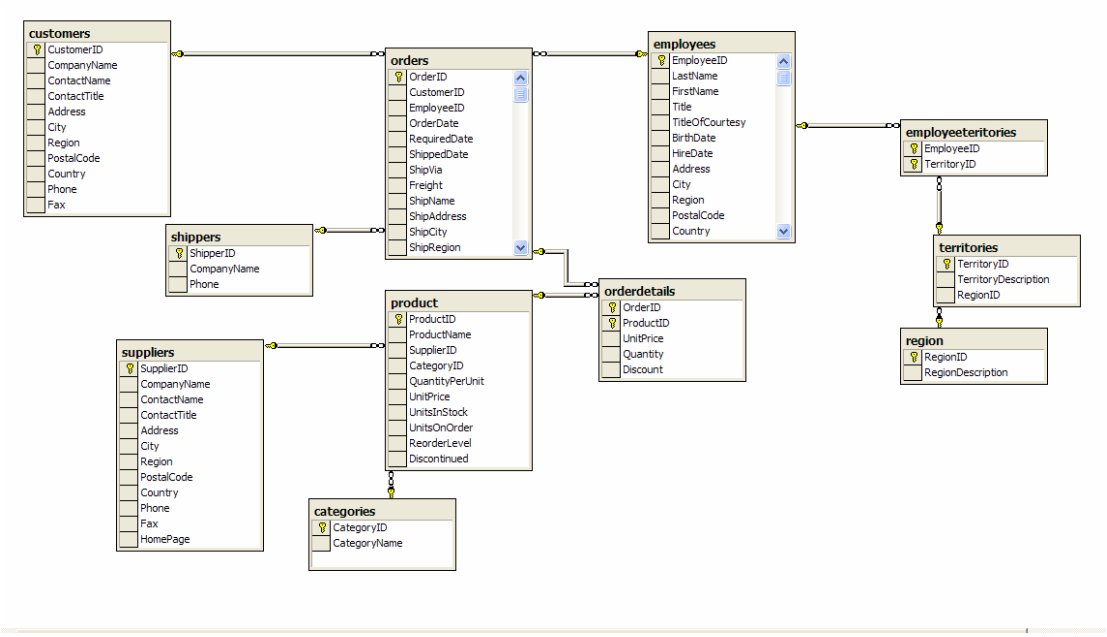
Ανάστροφη Μηχανική της βάσης δεδομένων Northwind

Η διαφορά που έχει η βάση δεδομένων Northwind με το Βιβλιοπωλείο που αναλύσαμε στις προηγούμενες παραγράφους είναι στα προϊόντα. Η ανάστροφη μηχανική μίας σχεσιακής βάσης δεδομένων θα παράγει από το σχεσιακό της μοντέλο (τους πίνακες) το μοντέλο οντοτήτων συσχετίσεων (ER) από το οποίο προήλθε ολόκληρη η δομή της βάσης.



Το διάγραμμα αυτό μετατρέπεται στο σχεσιακό μοντέλο

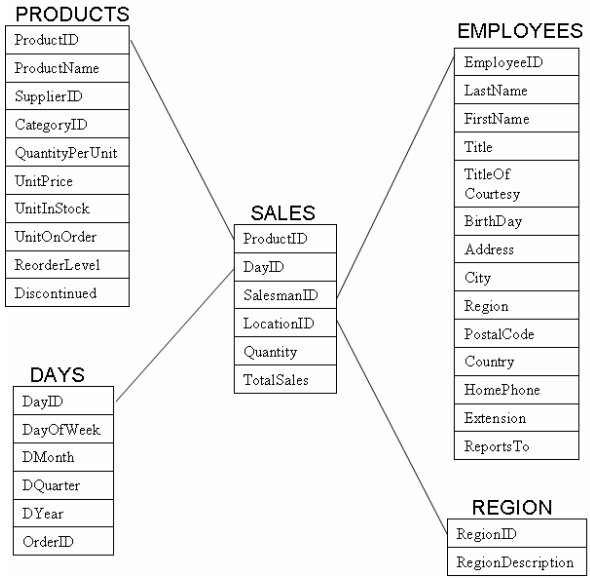
Παρακάτω παρουσιάζονται οι πίνακες από τους οποίους αντλούμε τα δεδομένα στις τέσσερις αποθήκες δεδομένων.



Κάθε σύστημα διαχείρισης παραγγελιών έχει σαν κορμό τους πίνακες Πελάτες, Παραγγελίες, Στοιχεία Παραγγελίας, Προϊόντα. Στον πίνακα ΣτοιχείαΠαραγγελίας (orderdetails) αποθηκεύεται η τιμή πώλησης του κάθε είδους, η ποσότητα και η έκπτωση.

Οι κύβοι στις αποθήκες δεδομένων της εφαρμογής παράγονται από αυτούς τους πίνακες

Στον πίνακα γεγονότων αποθηκεύονται τα αθροίσματα πώλησης κάθε είδους σε κάθε παραγγελία.



Εικόνα. Το σχήμα αστέρα που χρησιμοποιούμε στην πτυχιακή εργασία στις τέσσερις αποθήκες της εφαρμογής μας. Ο πίνακας SALES είναι ο πίνακας γεγονότων και υπάρχουν τέσσερις πίνακες διαστάσεων. Σε μία μεγάλη αποθήκη μπορούν να υπάρχουν πολλοί πίνακες γεγονότων, π.χ. κατά την διαίρεσή της σε μικρότερα τμήματα δημιουργούμε έναν πίνακα γεγονότων για κάθε έτος.

2 ΑΝΑΛΥΣΗ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΚΥΒΟΙ

Ανάλυση δεδομένων και συγκεντρωτικοί πίνακες. Ο παρακάτω είναι παράδειγμα ενός συγκεντρωτικού πίνακας Πωλήσεων με τέσσερις διαστάσεις : **Προϊόν, Έτος, Πωλητής, Περιοχή**

Προϊόν	Έτος	Πωλήσεις	Πωλητής	Περιοχή
Γαλακτοκομικά	1993	750 EUR	Buchanan	Ανατολή
Γαλακτοκομικά	1993	4.448 EUR	Buchanan	Βορράς
Γαλακτοκομικά	1992	4.873 EUR	Buchanan	Βορράς
Γαλακτοκομικά	1993	8.751 EUR	Buchanan	Δύση
Γαλακτοκομικά	1992	7.612 EUR	Buchanan	Νότος
Γαλακτοκομικά	1992	5.575 EUR	Davolio	Ανατολή
Γαλακτοκομικά	1992	7.686 EUR	Davolio	Βορράς
Γαλακτοκομικά	1993	2.741 EUR	Davolio	Βορράς
Γαλακτοκομικά	1993	2.733 EUR	Davolio	Δύση
Γαλακτοκομικά	1992	3.338 EUR	Davolio	Δύση
Γαλακτοκομικά	1993	6.544 EUR	Davolio	Δύση
Γαλακτοκομικά	1993	8.516 EUR	Davolio	Δύση
Γαλακτοκομικά	1993	8.076 EUR	Davolio	Νότος
Δημητριακά	1993	6.955 EUR	Buchanan	Ανατολή
Δημητριακά	1992	8.447 EUR	Buchanan	Ανατολή
Δημητριακά	1992	797 EUR	Buchanan	Βορράς
Δημητριακά	1993	2.956 EUR	Buchanan	Δύση
Δημητριακά	1993	6.930 EUR	Buchanan	Δύση
Δημητριακά	1992	8.165 EUR	Buchanan	Νότος
Δημητριακά	1993	450 EUR	Davolio	Ανατολή
Δημητριακά	1993	4.138 EUR	Davolio	Ανατολή
Δημητριακά	1992	5.720 EUR	Davolio	Ανατολή
Δημητριακά	1992	7.191 EUR	Davolio	Βορράς
Δημητριακά	1993	7.047 EUR	Davolio	Δύση
Δημητριακά	1992	2.686 EUR	Davolio	Δύση
Δημητριακά	1993	4.923 EUR	Davolio	Νότος

Αθρ. από			Περιοχή				
Έτος	Προϊόν	Πωλητής	Ανατολή	Βορράς	Δύση	Νότος	Γενικό Άθροισμα
1992	Γαλακτοκομικά	Buchanan		4873		7612	12485
		Davolio	5575	7686	3338		16599
	Άθροισμα - Γαλακτοκομικά		5575	12559	3338	7612	29084
	Δημητριακά	Buchanan	8447	797		8165	17409
		Davolio	5720	7191	2686		15597
	Άθροισμα - Δημητριακά		14167	7988	2686	8165	33006
Άθροισμα - 1992			19742	20547	6024	15777	62090
1993	Γαλακτοκομικά	Buchanan	75	4448	8751		13274
		Davolio		2741	17793	8076	28610
	Άθροισμα - Γαλακτοκομικά		75	7189	26544	8076	41884
	Δημητριακά	Buchanan	6955		9886		16841
		Davolio	4588		7047	4923	16558
	Άθροισμα - Δημητριακά		11543		16933	4923	33399
Άθροισμα - 1993			11618	7189	43477	12999	75283
Γενικό Άθροισμα			31360	27736	49501	28776	137373

2.1.1 Άμεση Αναλυτική Επεξεργασία- (Online Analytical Processing)

Τα δεδομένα ενός πίνακα ανάλογα με τις ιδιότητες τους μπορούν να χωριστούν σε ιδιότητες διάστασης και ιδιότητες μέτρησης. Για να γίνουν ποιο κατανοητές αυτές οι έννοιες θα αναφερθούμε σε ένα παράδειγμα. Ας υποθέσουμε ότι τα υφάσματα χαρακτηρίζονται από το όνομα , το χρώμα και το μέγεθος και ότι έχουμε έναν πίνακα sales με το σχήμα sales(item-name, color, size, number*)*. Υποθέστε ότι το item-name μπορεί να πάρει τις τιμές (skirt, dress, shirt, pant) (φούστα, φόρεμα, πουκάμισο, παντελόνι, αντίστοιχα), το color μπορεί να πάρει τις τιμές (dark, pastel, white) (σκούρο, παστέλ, άσπρο, αντίστοιχα) και το size μπορεί να πάρει τις τιμές (small, medium, large) (μικρό, μεγάλο, μεσαίο, αντίστοιχα).

Αν δοθεί μια σχέση που χρησιμοποιείται για ανάλυση δεδομένων, μπορούμε να προσδιορίσουμε κάποιες από τις ιδιότητες της ως ιδιότητες μέτρησης, αφού μετρούν κάποια τιμή την οποία συνοψίζουν. Για παράδειγμα, η ιδιότητα number του πίνακα sales είναι μια ιδιότητα μέτρησης, που μετρά τον αριθμό των μονάδων των προϊόντων που έχουν πουληθεί.. Στον πίνακα sales, τα item-name, color και size είναι ιδιότητες διάστασης. Τα δεδομένα μπορούν να μοντελοποιηθούν ως ιδιότητες διάστασης και ιδιότητες μέτρησης ονομάζονται πολυδιάστατα δεδομένα.

Τα αποτελέσματα ενός συγκεντρωτικού ερωτήματος εμφανίζονται σε έναν πίνακα που ονομάζεται πίνακας διασταύρωσης (cross-tabulation) ή συγκεντρωτικός πίνακας (pivot-table). Γενικά, ένας συγκεντρωτικός πίνακας είναι ένας πίνακας όπου οι τιμές ενός πεδίου (ας το πούμε A) αποτελούν τις επικεφαλίδες γραμμών, ενώ οι γραμμές ενός άλλου πεδίου (ας το πούμε B) αποτελούν τις επικεφαλίδες στηλών και οι τιμές ενός κελιού παράγονται ως εξής. Κάθε κελί μπορεί να προσδιοριστεί από τα (a_j, b_j), όπου το a_j, είναι μια τιμή για το *A* και το b_j, είναι μια τιμή για το B. Ένας τέτοιος πίνακας παρουσιάζεται στην εικόνα(α) που ακολουθεί.

size: all

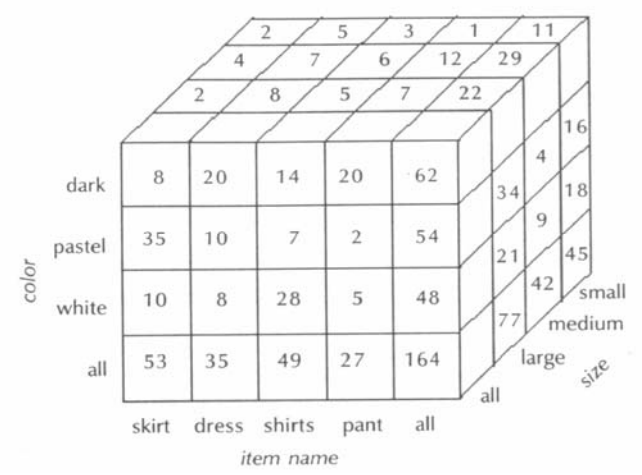
Ιδιότητες διάστασης: color

item-name (Ιδιότητες διάστασης)	dark	pastel	white	Σύνολο
skirt	8	35	10	53
dress	20	10	5	35
shirt	14	7	28	49
pant	20	2	5	27
Σύνολο	62	54	48	164

Ιδιότητες μέτρησης

Εικόνα (α): Πίνακας διασταύρωσης των *sales* κατά *item-name* και *color*.

Η γενίκευση ενός δισδιάστατου συγκεντρωτικού πίνακα, σε *n* διαστάσεις, μπορεί να απεικονισθεί οπτικά ως ένας n-διάστατος κύβος, που ονομάζεται κύβος δεδομένων. Στην εικόνα(β) που ακολουθεί δείχνει ένα κύβο για τις σχέσεις sales. Ο κύβος δεδομένων στο παράδειγμα μας έχει τρεις διαστάσεις, τις item-name, color και size η ιδιότητα μέτρησης είναι το number. Κάθε κελί προσδιορίζεται με τιμές για αυτές τις τρεις διαστάσεις. Κάθε κελί στον κύβο δεδομένων περιέχει μια τιμή, όπως στον συγκεντρωτικό πίνακα.

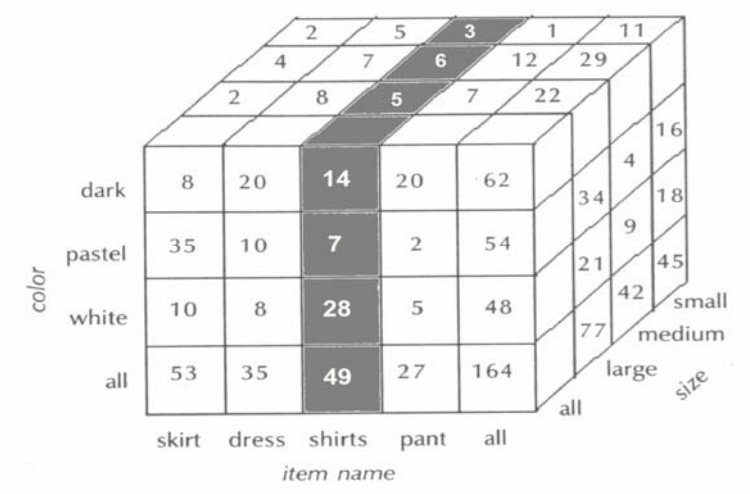


Εικόνα. Τρισδιάστατος κύβος δεδομένων

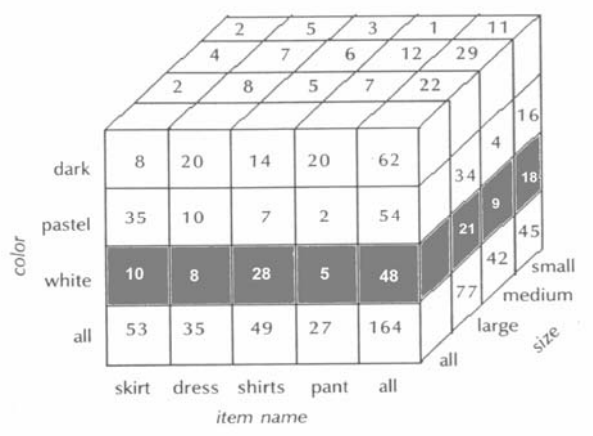
Με ένα OLAP σύστημα, μπορούμε να κοιτάξουμε διαφορετικούς συγκεντρωτικούς πίνακες με τα ίδια δεδομένα, επιλέγοντας τις ιδιότητες στον συγκεντρωτικό πίνακα. Κάθε συγκεντρωτικός πίνακας είναι μια δυσδιάστατη προβολή ενός πολυδιάστατου κύβου. Για παράδειγμα, μπορούμε να επιλέξουμε ένα πίνακα διασταύρωσης μεταξύ item-name και size, ή ένα πίνακα διασταύρωσης μεταξύ color και size. Η λειτουργία της αλλαγής των διαστάσεων που χρησιμοποιείται σε ένα πίνακα διασταύρωσης ονομάζεται pivoting (περιστροφή πίνακα).

Ένα OLAP σύστημα μας παρέχει μια σειρά κι από άλλες λειτουργίες όπως:

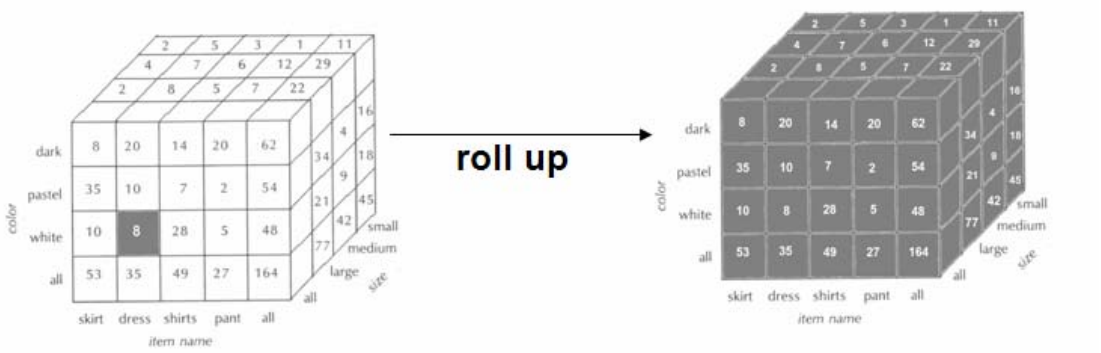
- **Τεμαχισμός (λειτουργία slice):** Βλέπουμε ένα κομμάτι του κύβου (υπο-κύβος) για να πάρουμε κάποιες πιο ειδικές πληροφορίες. Αυτό το επιτυγχάνουμε επιλέγοντας μια μονό διάσταση του κύβου.



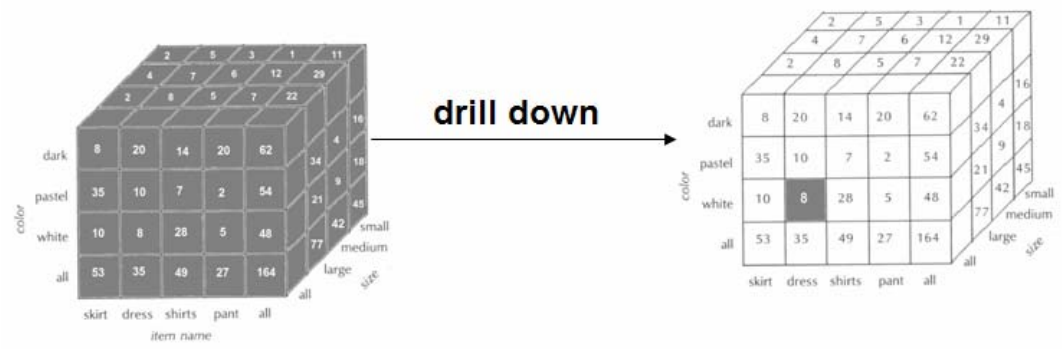
- **Κομμάτιασμα (λειτουργία dice):** Βλέπουμε έναν υπο-κύβο επιλέγοντας δυο ή περισσότερες διαστάσεις. Αυτό μπορεί να εκτελεστεί με ένα τεμαχισμό σε μία διάσταση και κατόπιν με περιστροφή (λειτουργία pivot) του κύβου για να επιλεχθεί και άλλη διάσταση.



- **Συναθροιστική άνοδος (λειτουργία roll up**) (μείωση διαστάσεων, συνάθροιση): Η συναθροιστική άνοδος επιτρέπει στο χρήστη να κάνει ερωτήσεις κινούμενος στην ιεραρχία προς τα πάνω. Η λειτουργία της μετακίνησης από τη λεπτότερη κατάτμηση των δεδομένων σε μια πιο γενική κατάτμηση ονομάζεται **rollup**. Έτσι μπορούμε για παράδειγμα να δούμε τις συνολικές πωλήσεις .



- **Αναλυτική κάθοδος (λειτουργία drill down):** Η αντίστροφη λειτουργία δηλαδή, η μετακίνηση από την πιο γενική κατάτμηση στην πιο λεπτομερή, ονομάζεται **drill down.** Για παράδειγμα πόσα άσπρα φορέματα έχουν πουληθεί.



3 ΟΙ ΑΠΟΘΗΚΕΣ ΔΕΔΟΜΕΝΩΝ ΣΤΗΝ ΕΦΑΡΜΟΓΗ ΜΑΣ

Οι τέσσερις μικρές αποθήκες δεδομένων που κάναμε όπως και η εφαρμογή μας είναι ανεξάρτητες από τις συγκεκριμένες βάσεις δεδομένων (μπορούν να χρησιμοποιηθούν σε οποιοδήποτε σύστημα διαχείρισης παραγγελιών).

Υπάρχουν πολλά πλεονεκτήματα από τη χρήση μίας αποθήκης δεδομένων. Λόγω του ότι παρέχει μία ενοποίηση των δεδομένων από πολλαπλές πηγές, η χρήση της μπορεί να παρέχει πιο αποδοτική προσπέλαση των δεδομένων. Οι αποθήκες δεδομένων παρέχουν στους χρηστές πρόσβαση στα δεδομένα αυτά μέσα από τα οποία μπορούν να γραφούν ερωτήματα για την λήψη διαφόρων αποφάσεων.

Ένα υποσύνολο μιας αποθήκης δεδομένων είναι το data mart το οποίο μπορεί να αποθηκεύεται και να προσπελαύνετε ξεχωριστά.

Υπάρχουν ορισμένοι τρόποι για να βελτιώσει κανείς την απόδοση των αποθηκών δεδομένων:

- **Υλοποίηση συνόψεων:** Επειδή πολλές εφαρμογές απαιτούν συνοπτικές πληροφορίες, τα δεδομένα που είναι γνωστό ότι χρειάζονται για τις ερωτήσεις σύνοψης πρέπει να συνοψίζονται πριν αποθηκευτούν.

- **Αποκανονικοποίηση**: Η κανονικοποίηση μειώνει τα προβλήματα όπως αυτό των περιττών επαναλήψεων κατά τη διάρκεια εισαγωγής, ενημέρωσης και διαγραφής των εγγραφών. Ωστόσο, αυτές οι βελτιώσεις επιτυγχάνονται με κόστος την αύξηση του χρόνου επεξεργασίας. Σε μία αποθήκη δεδομένων, η βελτιωμένη απόδοση επιτυγχάνεται αποθηκεύοντας τα δεδομένα μη κανονικοποιημένα.

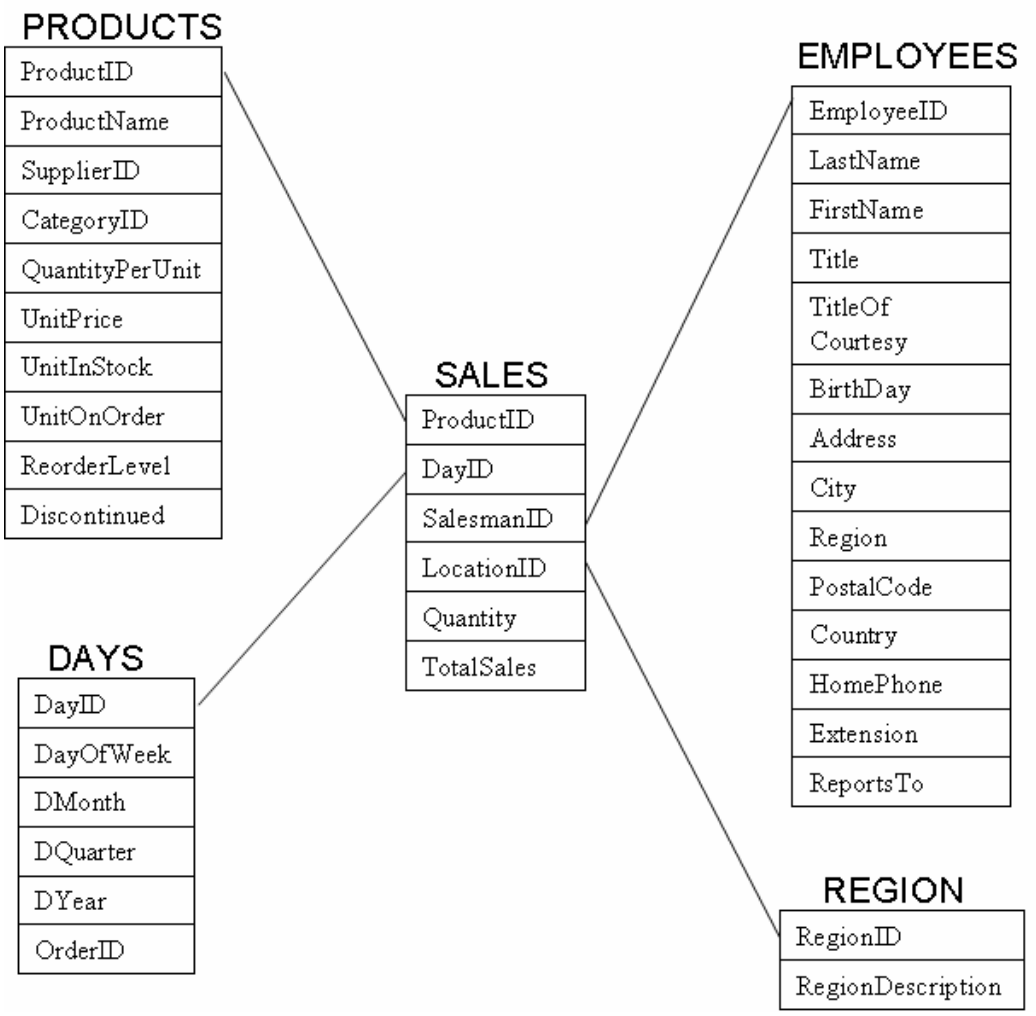
- **Διαμερισμός:** Η διαίρεση της αποθήκης δεδομένων σε μικρότερα τμήματα ελαττώνει το χρόνο επεξεργασίας αφού οι ερωτήσεις προσπελαύνουν μικρότερα σύνολα δεδομένων.

3.1 ΤΟ ΣΧΗΜΑ ΤΩΝ ΑΠΟΘΗΚΩΝ ΔΕΔΟΜΕΝΩΝ

Για την απεικόνιση πολυδιάστατων δεδομένων έχουν αναπτυχθεί ειδικά σχήματα. Μερικά από αυτά είναι τα σχήματα αστέρα και χιονονιφάδας.

Το σχήμα αστέρα (star schema) απεικονίζει τα δεδομένα σαν μία συλλογή δύο τύπων: γεγονότων και διαστάσεων. Σε αντίθεση με το σχεσιακό σχήμα, το οποίο είναι επίπεδο, το σχήμα αστέρα είναι μια γραφική άποψη των δεδομένων. Στο κέντρο του αστέρα υπάρχει ένας πίνακας ο **πίνακας γεγονότων (fact table)** – που μερικές φορές ονομάζετε και πρωτεύον πίνακας (major tables). Εξωτερικά του πίνακα γεγονότων, υπάρχουν άλλοι πίνακες **οι πίνακες διαστάσεων** (dimension tables) -που μερικές φορές ονομάζονται και δευτερεύοντες πίνακες (minor tables). Η πιο απλή μορφή ενός σχήματος αστέρα αποτελείται από έναν πίνακα γεγονότων και πολλούς πίνακες διαστάσεων. Σε αυτή την περίπτωση, ένα πεδίο του πίνακα γεγονότων δείχνει προς ένα πεδίο του πίνακα διαστάσεως.

Τα πραγματικά προσβάσιμα δεδομένα είναι αποθηκεμένα στους πίνακες γεγονότων και για αυτό οι τελευταίοι τείνουν να είναι αρκετά μεγάλοι. Οι πίνακες διαστάσεων, οι οποίοι συνήθως είναι μικρότεροι, αποθηκεύουν αναλυτικές πληροφορίες που σχετίζονται με το πεδίο του πίνακα γεγονότος που συνδέονται. Το Σχήμα που ακολουθεί απεικονίζει το σχήμα αστέρα που χρησιμοποιήσαμε.



Εικόνα. Το σχήμα αστέρα που χρησιμοποιούμε στην πτυχιακή εργασία στις τέσσερις αποθήκες της εφαρμογής μας. Ο πίνακας SALES είναι ο πίνακας γεγονότων και υπάρχουν τέσσερις πίνακες διαστάσεων. Σε μία μεγάλη αποθήκη μπορούν να υπάρχουν πολλοί πίνακες γεγονότων, π.χ. κατά την διαίρεσή της σε μικρότερα τμήματα δημιουργούμε έναν πίνακα γεγονότων για κάθε έτος.

Στην πτυχιακή εργασία ο πίνακας γεγονότων είναι ο SALES (Πωλήσεις) ο οποίος περιέχει άλλα δυο πεδία την ποσότητα (Quantity) και τις συγκεντρωτικές πωλήσεις (Total Sales). Οι πίνακες REGION (Τοποθεσία), DAYS (Μέρες), PRODUCTS (Προϊόντα) και EMPLOYEES (Εργαζόμενοι) αποτελούν τους πίνακες διαστάσεων. Η πρόσβαση στον πίνακα γεγονότων από ένα πίνακα διάστασης μπορεί να επιτευχθεί μέσω της σύνδεσης μεταξύ ενός πεδίου του πίνακα διαστάσεων και ενός πεδίου του πίνακα γεγονότων. Για παράδειγμα, μπορούμε να προσπελάσουμε όλες τις τοποθεσίες στην Αθήνα υποβάλλοντας την ακόλουθη SQL ερώτηση:

Select Quantity, Total Sales

From Sales, Region

Where (Sales.LocationID=Region.RegionID) **AND** (Region.RegionDescription = “Athens”)

Εδώ, το LocationID είναι ένα ξένο κλειδί στον πίνακα γεγονότων Sales και πρωτεύον κλειδί στον πίνακα διάστασης Region (RegionID). Το πρωτεύον κλειδί για τον πίνακα γεγονότων είναι μία συλλογή ξένων κλειδιών που δείχνουν προς τους πίνακες διαστάσεων. Επιπλέον, ένας πίνακας διάστασης μπορεί ο ίδιος να δείχνει προς ένα άλλο πίνακα διάστασης.

3.2 Η ΑΠΟΘΗΚΗ ΔΕΔΟΜΕΝΩΝ ΣΤΗΝ ΤΟΠΟΘΕΣΙΑ 3

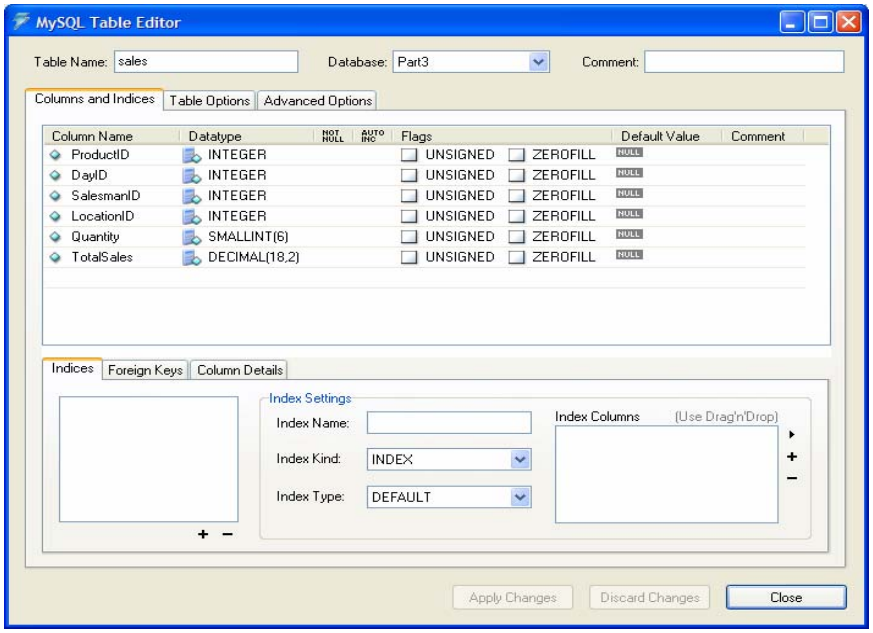
Αυτή η αποθήκη δεδομένων με όνομα Part3, βρίσκεται την τοποθεσία με LocationID=3 σε μια βάση της MySQL

Η Δημιουργία των πινάκων και η εισαγωγή των δεδομένων

- Δημιουργία του πίνακα SALES

```
CREATE TABLE SALES
      (
       ProductID int NULL,
       DayID int NULL,
       SalesmanID int NULL,
       LocationID int NULL,
       Quantity smallint NULL,
       TotalSales numeric(18,2) NULL
      )
```

Ο πίνακας SALES στη βάση δεδομένων στην MySQL

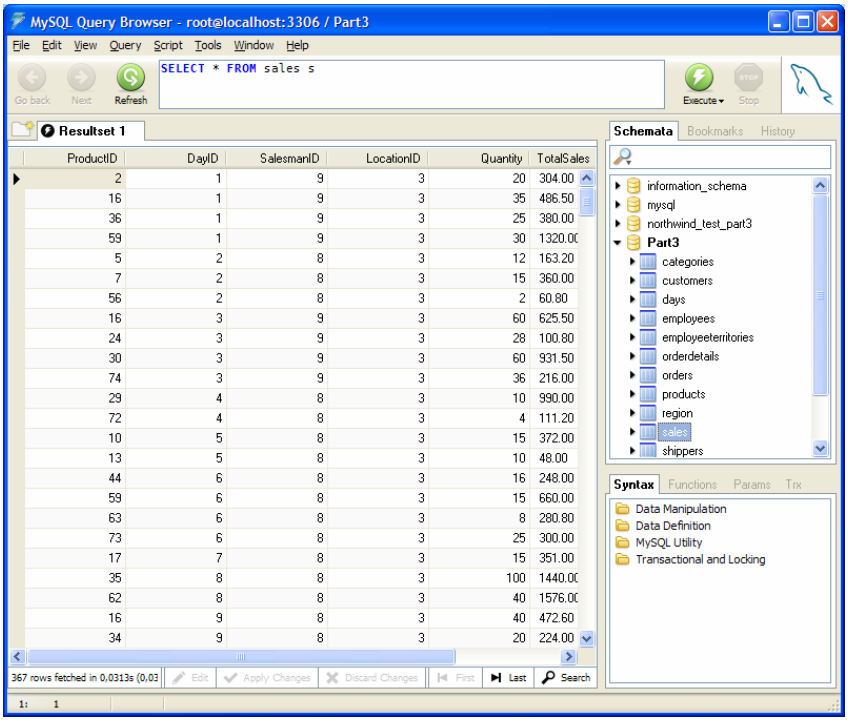


- Εισαγωγή δεδομένων στον πίνακα SALES

```
DELETE FROM Sales;
INSERT INTO Sales (ProductID, DayID, SalesmanID, LocationID,
Quantity, TotalSales)
select p.ProductId,
       d.DayId,
       e.EmployeeID,
       1 as LocationID,
       od.quantity,
      (od.quantity * od.unitprice) * (1 - od.discount) as TotalSales
```

```
from ORDERS o
Inner join Days d on o.OrderId = d.OrderID
inner join ORDERDETAILS od on o.orderID=od.orderID
inner join EMPLOYEES e on e.EmployeeID=o.EmployeeID
inner join PRODUCT p on p.ProductID=od.ProductID
```

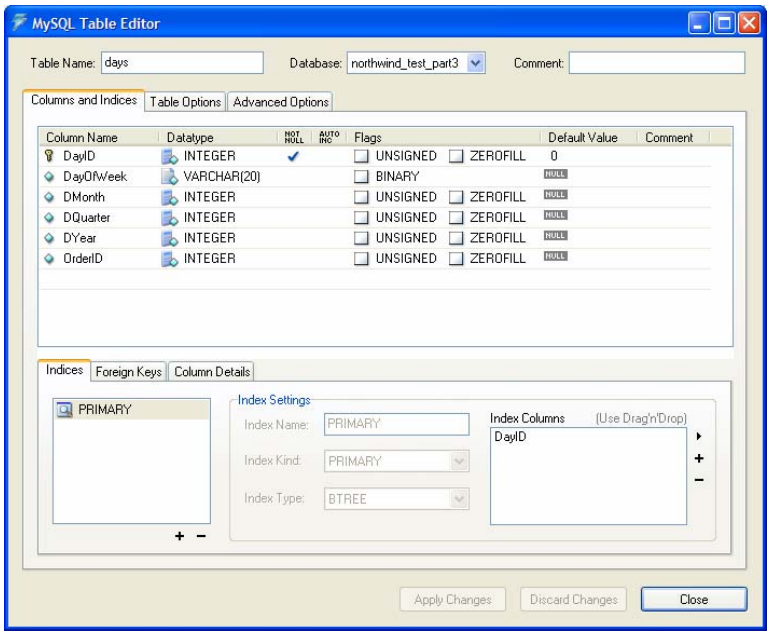
Δείγμα δεδομένων του πίνακα SALES στην MySQL



- Δημιουργία του πίνακα DAYS

```
create table days(DayID int NOT NULL IDENTITY (1, 1),
                  DayOfWeek varchar(20),
                  DMonth int,
                  DQuarter int,
                  DYear int,
                  OrderID int NULL,
                  Primary Key (DayID))
```

Ο πίνακας DAYS στη βάση δεδομένων στην MySQL

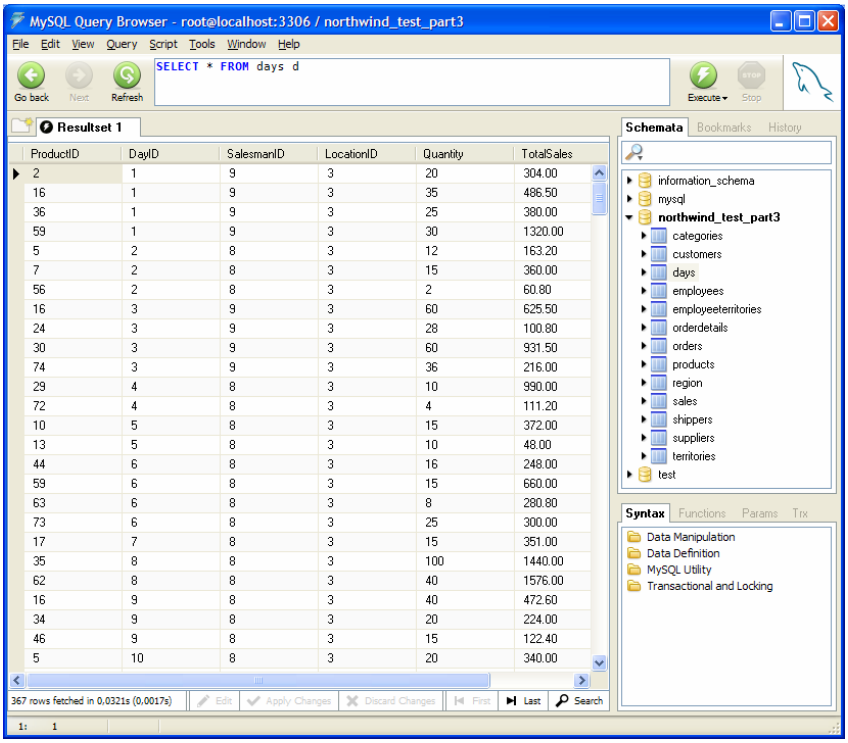


- Εισαγωγή δεδομένων στον πίνακα DAYS

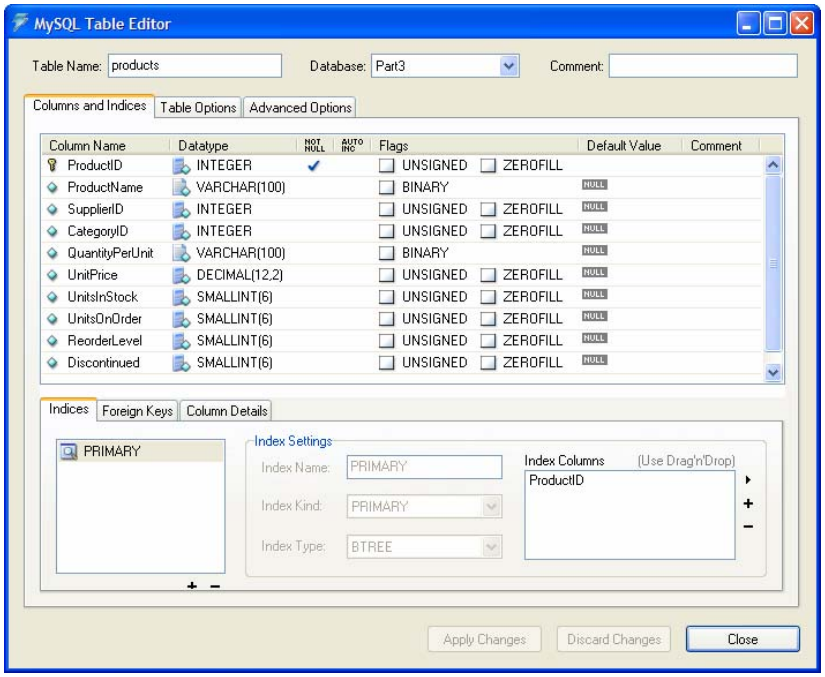
```
insert into Days(DayOfWeek, DMonth, DQuarter, DYear, OrderID)
select DATENAME (dw, OrderDate), Month(orderDate), DATENAME(quarter,
OrderDate), Year(orderDate), OrderID from orders
```

Η πρώτη συνάρτηση στο select εξάγει την ημέρα της εβδομάδος από μια ημερομηνία, η δεύτερη συνάρτηση εξάγει τον μήνα, η τρίτη το τρίμηνο, και η τέταρτη το έτος.

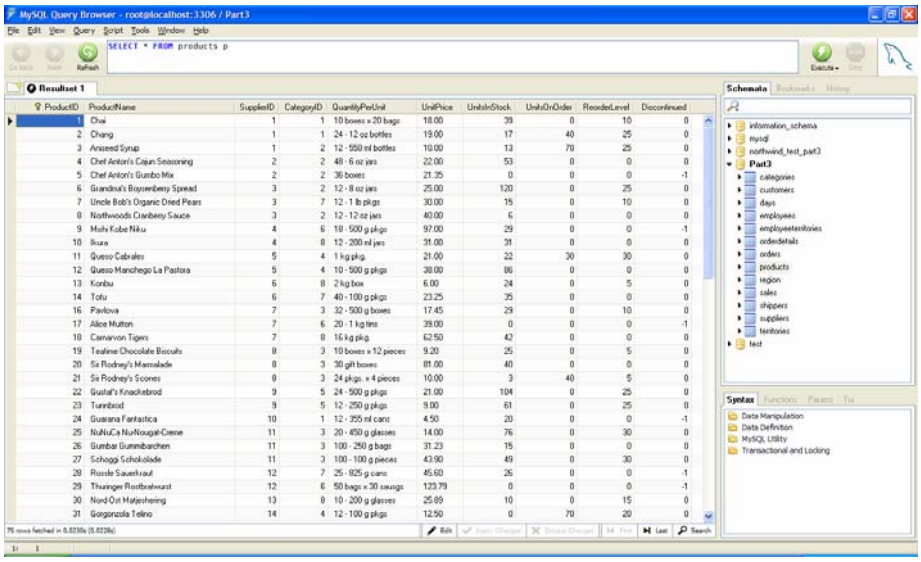
Δείγμα δεδομένων του πίνακα DAYS στην MySQL



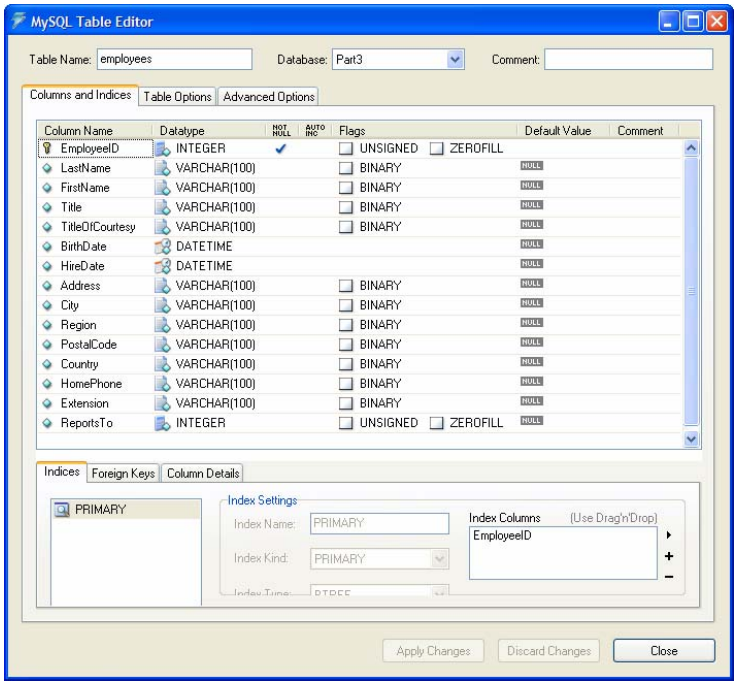
Ο πίνακας products στη βάση δεδομένων στην MySQL



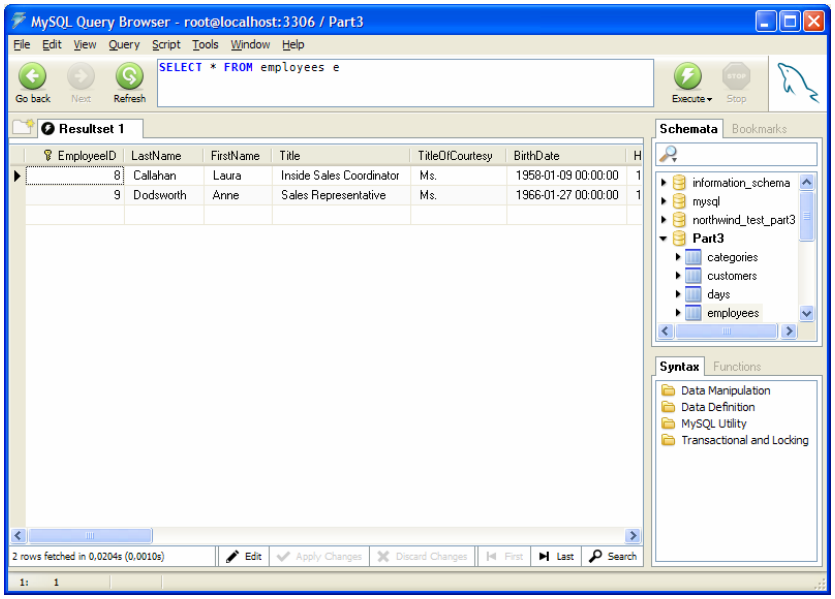
Δείγμα δεδομένων του πίνακα products στην MySQL



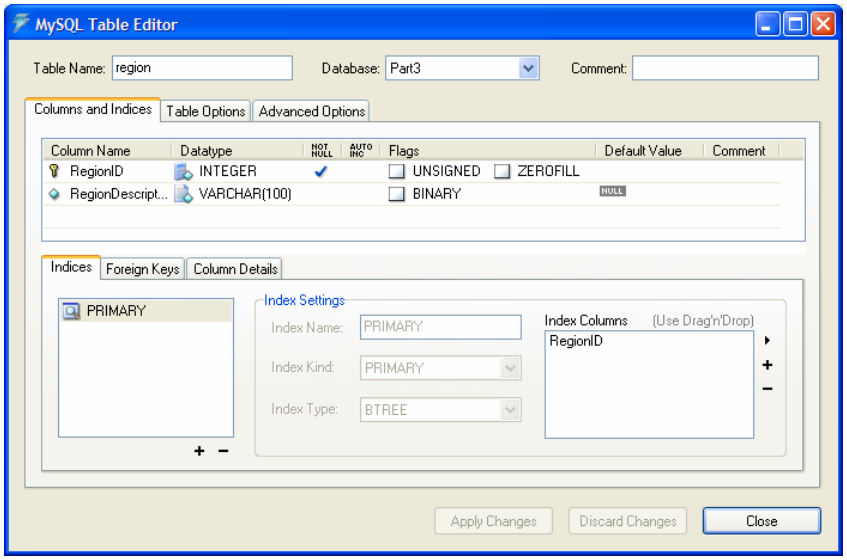
Ο πίνακας employees στη βάση δεδομένων στην MySQL



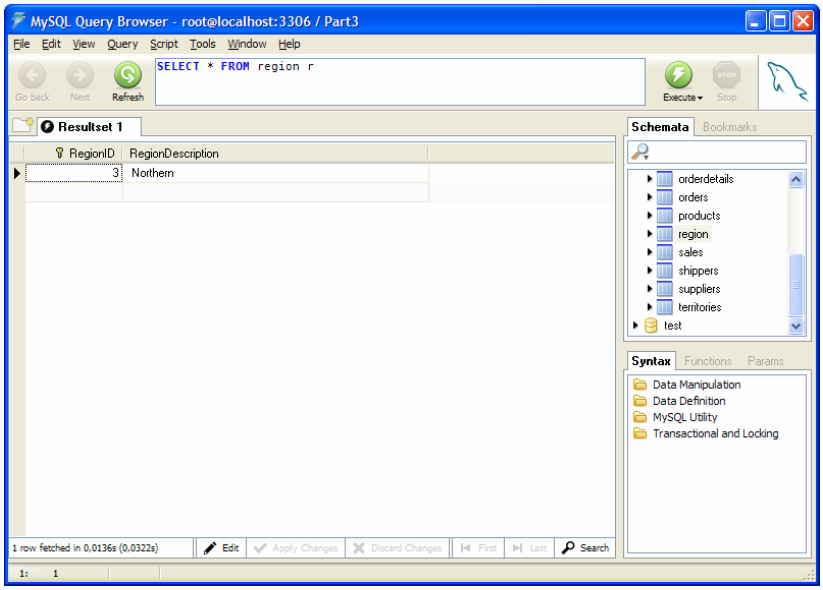
Δείγμα δεδομένων του πίνακα employees στην MySQL

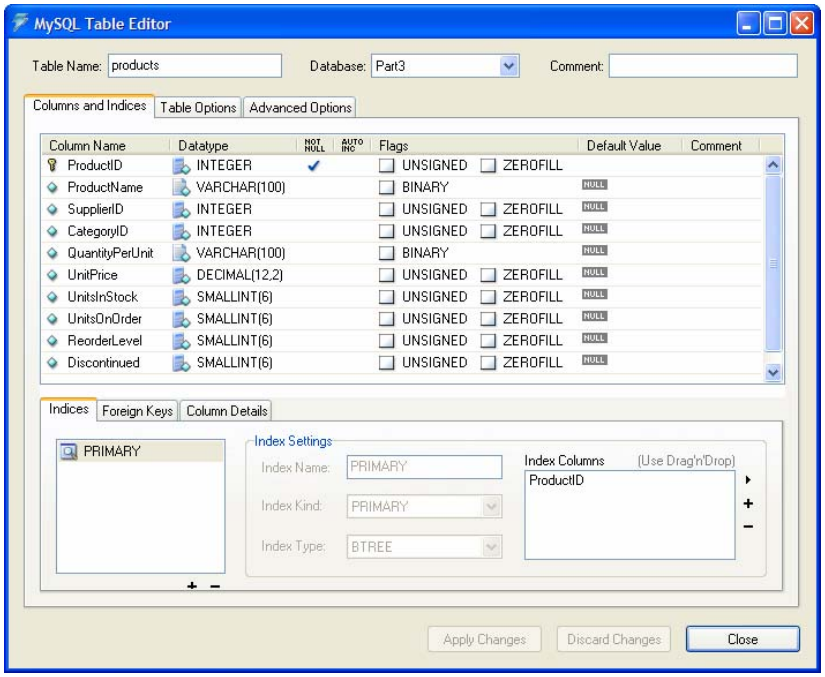


Ο πίνακας region στη βάση δεδομένων στην MySQL



Δείγμα δεδομένων του πίνακα region στην MySQL





3.3 Η ΑΠΟΘΗΚΗ ΔΕΔΟΜΕΝΩΝ ΣΤΗ ΤΟΠΟΘΕΣΙΑ 2

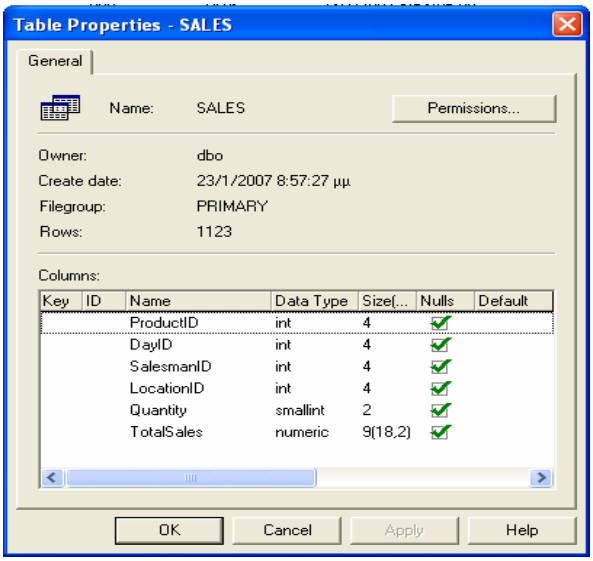
Αυτή η αποθήκη δεδομένων με όνομα Part2, βρίσκεται στην τοποθεσία με LocationID=2 σε μια βάση του SQL Server.

Η Δημιουργία των πινάκων και η εισαγωγή των δεδομένων

- Δημιουργία του πίνακα SALES

```
CREATE TABLE SALES
          (
           ProductID int NULL,
           DayID int NULL,
           SalesmanID int NULL,
           LocationID int NULL,
           Quantity smallint NULL,
           TotalSales numeric(18,2) NULL
          )
```

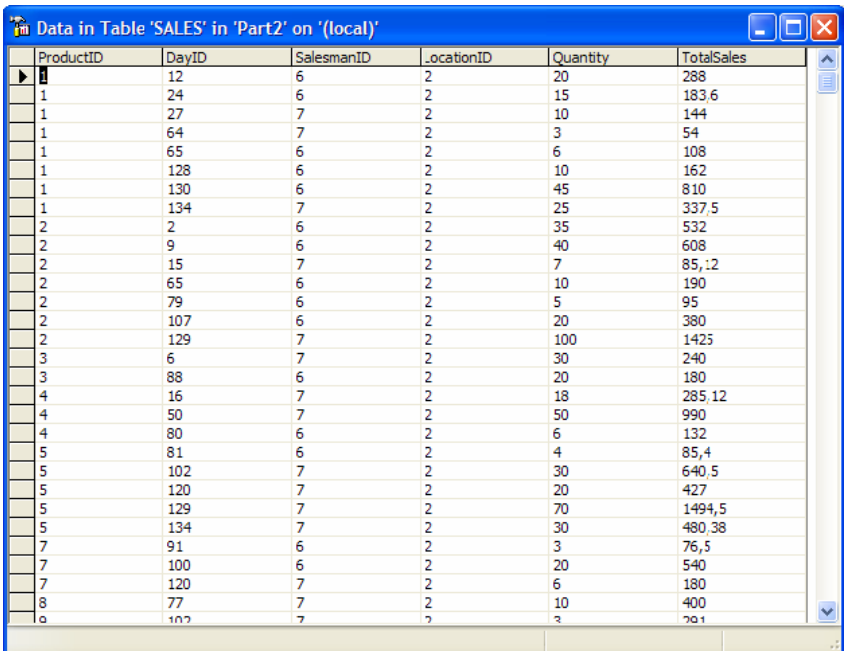
Ο πίνακας SALES στη βάση δεδομένων στον SQL Server



- Εισαγωγή δεδομένων στον πίνακα SALES

```
DELETE FROM Sales;
INSERT INTO Sales (ProductID, DayID, SalesmanID, LocationID,
Quantity, TotalSales)
select p.ProductId,
       d.DayId,
       e.EmployeeID,
       1 as LocationID,
       od.quantity,
       (od.quantity * od.unitprice) * (1 - od.discount) as TotalSales
from ORDERS o
Inner join Days d on o.OrderId = d.OrderID
inner join ORDERDETAILS od on o.orderID=od.orderID
inner join EMPLOYEES e on e.EmployeeID=o.EmployeeID
inner join PRODUCT p on p.ProductID=od.ProductID
```

Δείγμα δεδομένων του πίνακα SALES στον SQL Server

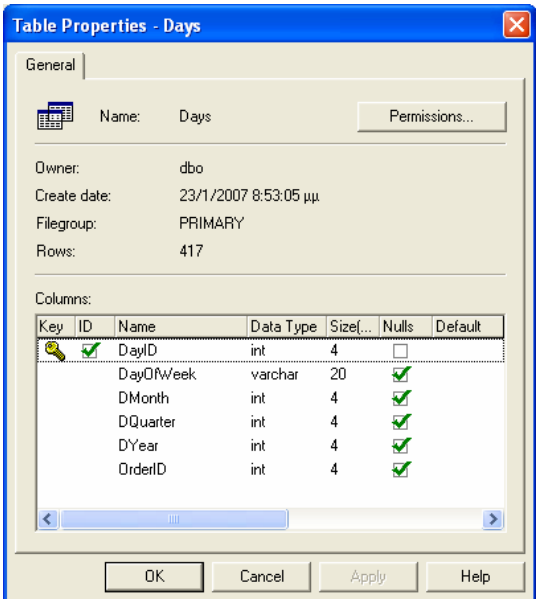


ProductID	DayID	SalesmanID	LocationID	Quantity	TotalSales
1	12	6	2	20	288
1	24	6	2	15	183,6
1	27	7	2	10	144
1	64	7	2	3	54
1	65	6	2	6	108
1	128	6	2	10	162
1	130	6	2	45	810
1	134	7	2	25	337,5
2	2	6	2	35	532
2	9	6	2	40	608
2	15	7	2	7	85,12
2	65	6	2	10	190
2	79	6	2	5	95
2	107	6	2	20	380
2	129	7	2	100	1425
3	6	7	2	30	240
3	88	6	2	20	180
4	16	7	2	18	285,12
4	50	7	2	50	990
4	80	6	2	6	132
5	81	6	2	4	85,4
5	102	7	2	30	640,5
5	120	7	2	20	427
5	129	7	2	70	1494,5
5	134	7	2	30	480,38
7	91	6	2	3	76,5
7	100	6	2	20	540
7	120	7	2	6	180
8	77	7	2	10	400

- Δημιουργία του πίνακα DAYS

```
create table Days(DayID int NOT NULL IDENTITY (1, 1),
                  DayOfWeek varchar(20),
                  DMonth int,
                  DQuarter int,
                  DYear int,
                  OrderID int NULL,
                  Primary Key (DayID))
```

Ο πίνακας DAYS στη βάση δεδομένων στον SQL Server



- Εισαγωγή δεδομένων στον πίνακα DAYS

```
insert into Days(DayOfWeek, DMonth, DQuarter, DYear, OrderID)
```

```
select DATENAME (dw, OrderDate), Month(orderDate), DATENAME(quarter,
OrderDate), Year(orderDate), OrderID from orders
```

Η πρώτη συνάρτηση στο select εξάγει την ημέρα της εβδομάδος από μια ημερομηνία, η δεύτερη συνάρτηση εξάγει τον μήνα, η τρίτη το τρίμηνο, και η τέταρτη το έτος.

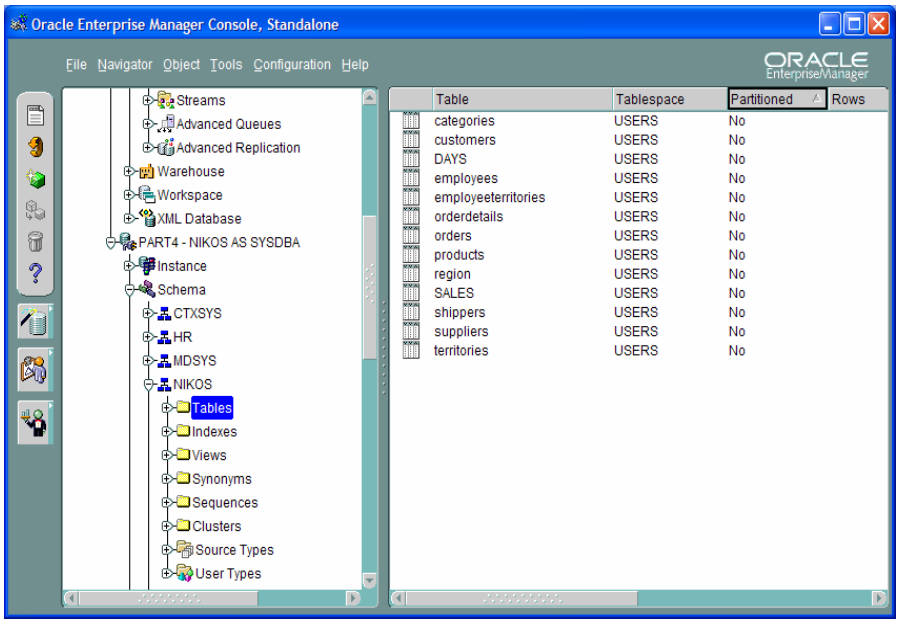
Δείγμα δεδομένων του πίνακα DAYS στον SQL Server

Data in Table 'Days' in 'Part2' on '(local)'

DayID	DayOfWeek	DMonth	DQuarter	DYear	OrderID
1	Friday	7	3	1996	10249
2	Wednesday	7	3	1996	10264
3	Thursday	8	3	1996	10271
4	Friday	8	3	1996	10272
5	Tuesday	8	3	1996	10274
6	Monday	8	3	1996	10289
7	Tuesday	8	3	1996	10291
8	Tuesday	9	3	1996	10296
9	Thursday	9	3	1996	10298
10	Wednesday	9	3	1996	10303
11	Wednesday	9	3	1996	10308
12	Monday	9	3	1996	10317
13	Wednesday	10	4	1996	10319
14	Friday	10	4	1996	10322
15	Tuesday	10	4	1996	10335
16	Wednesday	10	4	1996	10336
17	Tuesday	10	4	1996	10341
18	Friday	11	4	1996	10349
19	Monday	11	4	1996	10350
20	Wednesday	11	4	1996	10353
21	Friday	11	4	1996	10355
22	Monday	11	4	1996	10356
23	Thursday	11	4	1996	10367
24	Tuesday	12	4	1996	10370
25	Monday	12	4	1996	10390
26	Thursday	12	4	1996	10395
27	Tuesday	1	1	1997	10406
28	Thursday	1	1	1997	10423
29	Thursday	1	1	1997	10424

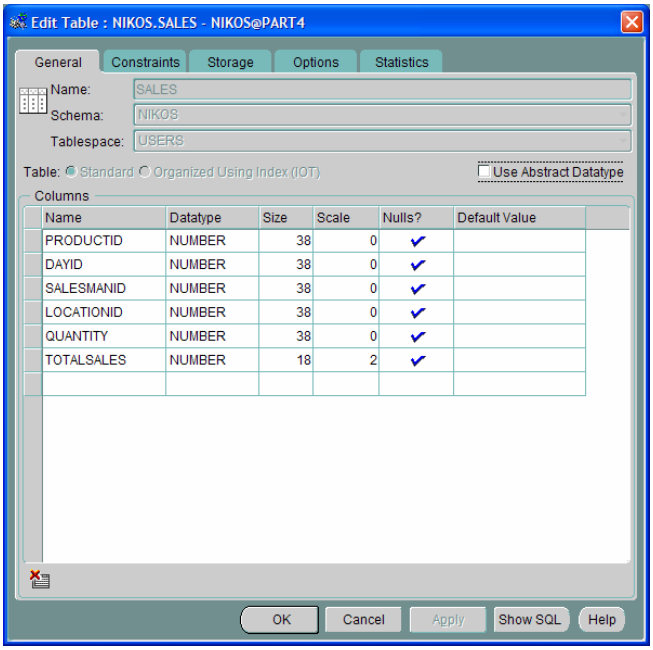
3.4 Η ΑΠΟΘΗΚΗ ΔΕΔΟΜΕΝΩΝ ΣΤΗ ΤΟΠΟΘΕΣΙΑ 4

Αυτή η αποθήκη δεδομένων με όνομα Part4, βρίσκεται στην τοποθεσία με LocationID=4 σε μια βάση της Oracle.

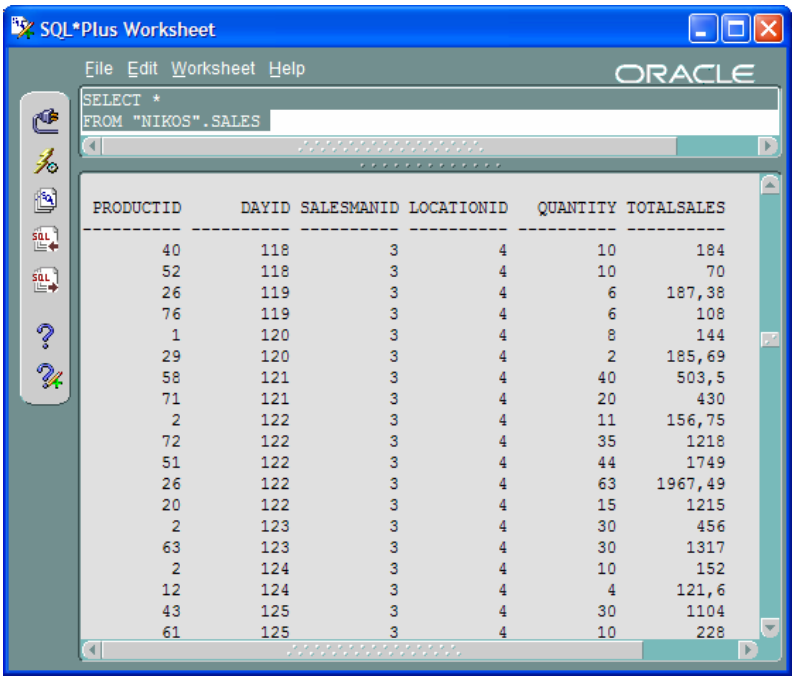


Η Δημιουργία των πινάκων και η εισαγωγή των δεδομένων

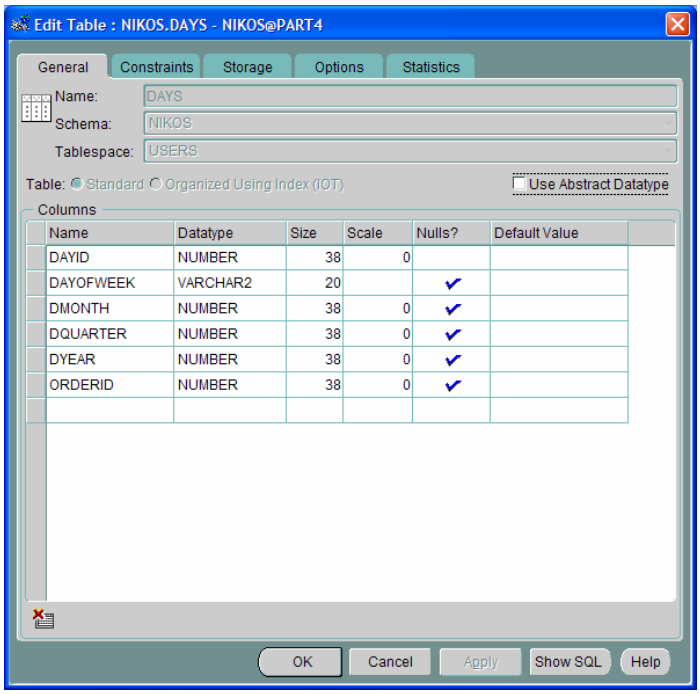
Ο πίνακας SALES στη βάση δεδομένων στης Oracle



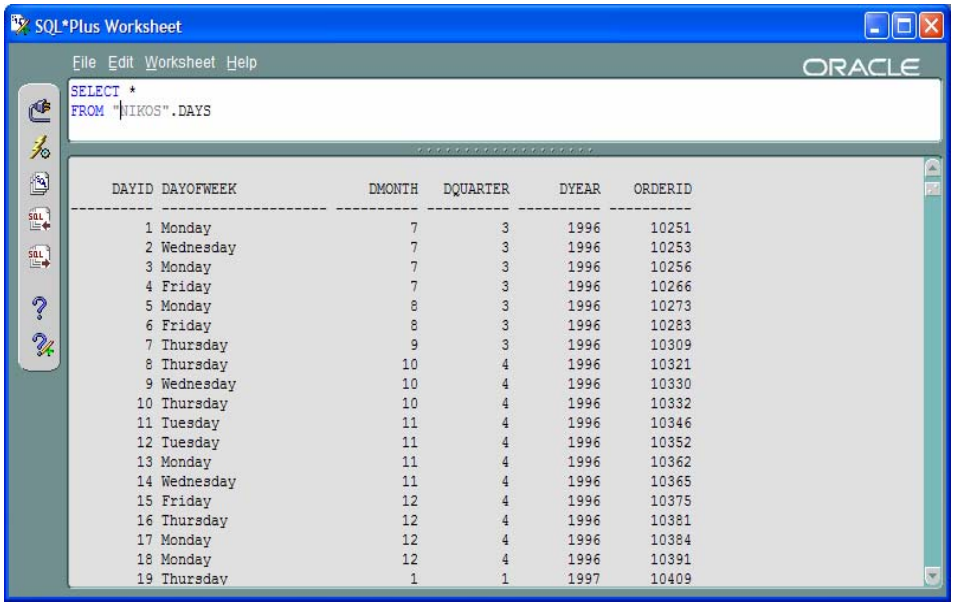
Δείγμα δεδομένων του πίνακα SALES στην Oracle



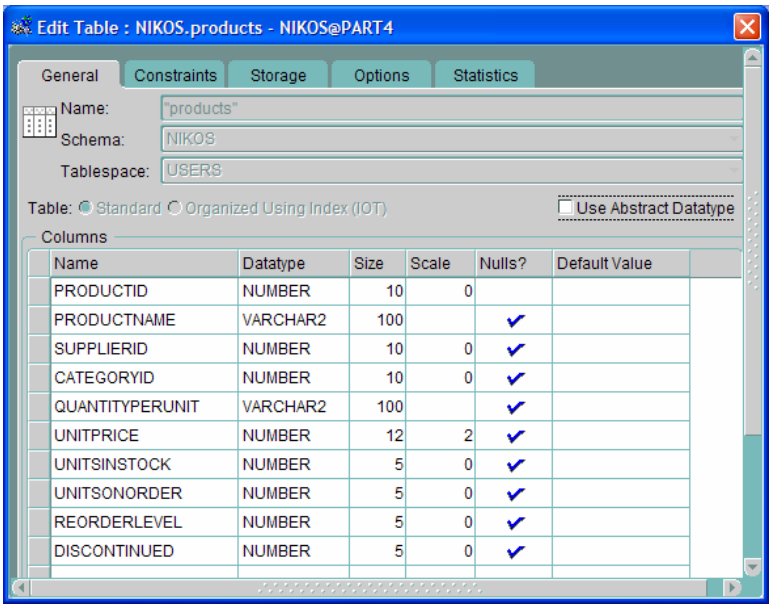
Ο πίνακας DAYS στη βάση δεδομένων στην Oracle



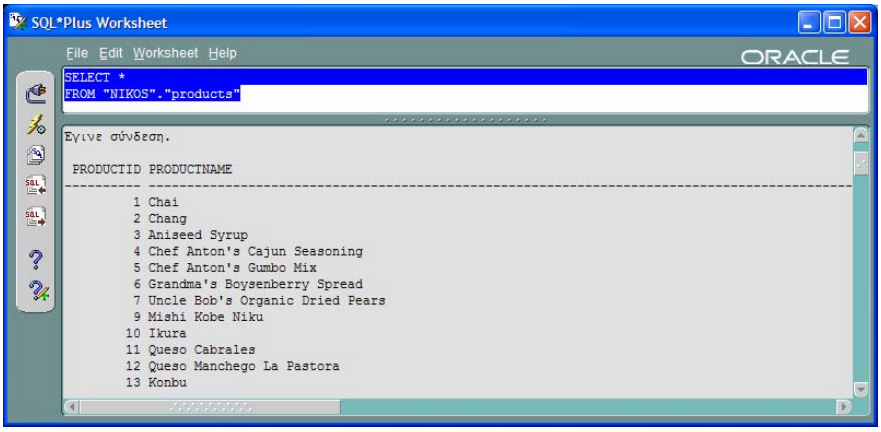
Δείγμα δεδομένων του πίνακα DAYS στην Oracle



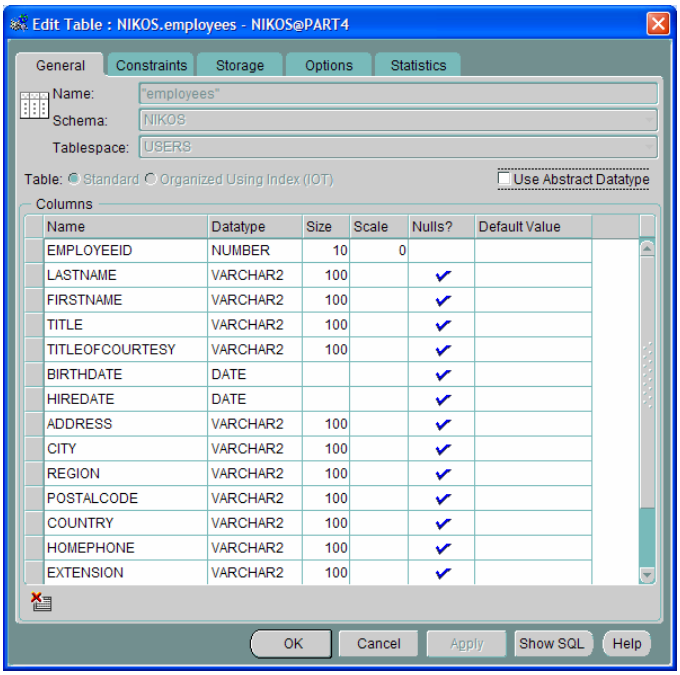
Ο πίνακας products στη βάση δεδομένων στην Oracle



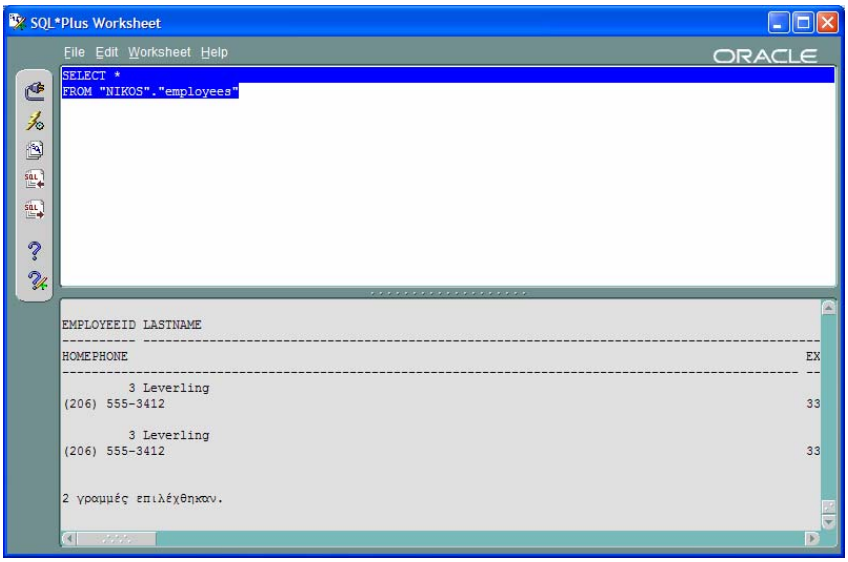
Δείγμα δεδομένων του πίνακα products στην Oracle



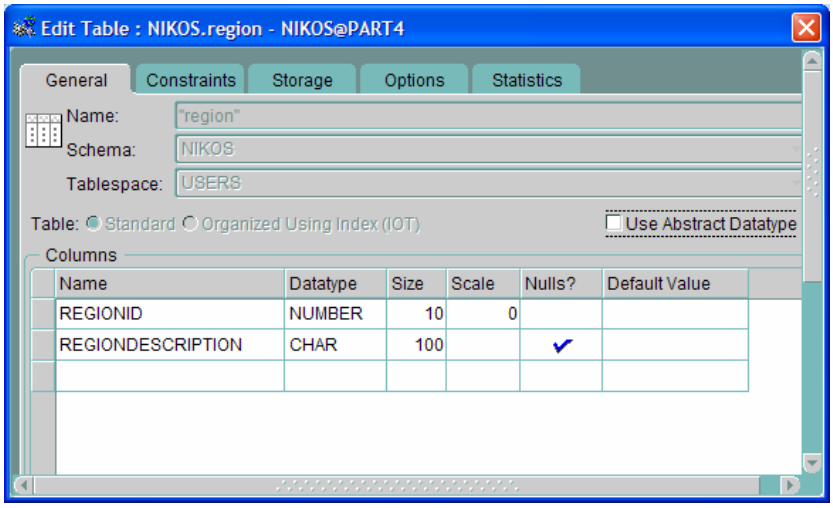
Ο πίνακας employees στη βάση δεδομένων στην Oracle



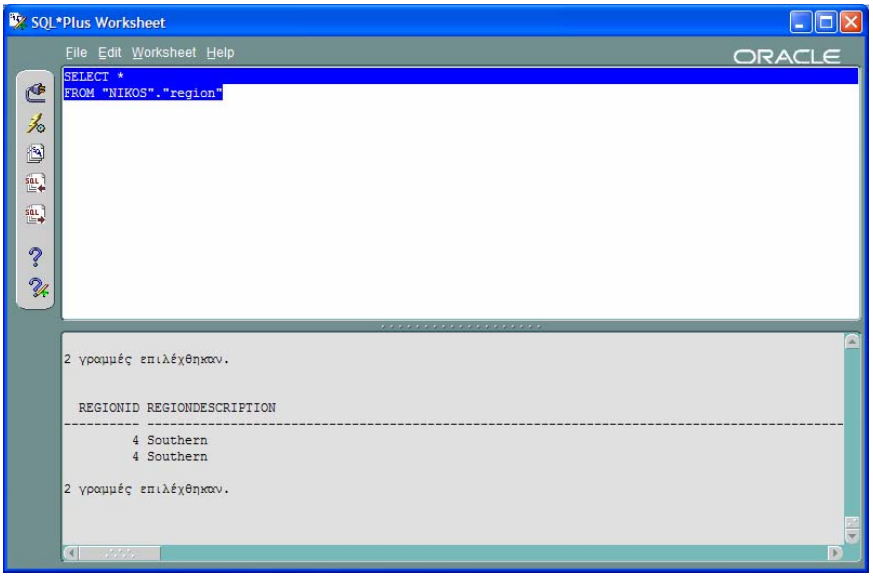
Δείγμα δεδομένων του πίνακα employees στην Oracle



Ο πίνακας region στη βάση δεδομένων στην Oracle



Δείγμα δεδομένων του πίνακα region στην Oracle



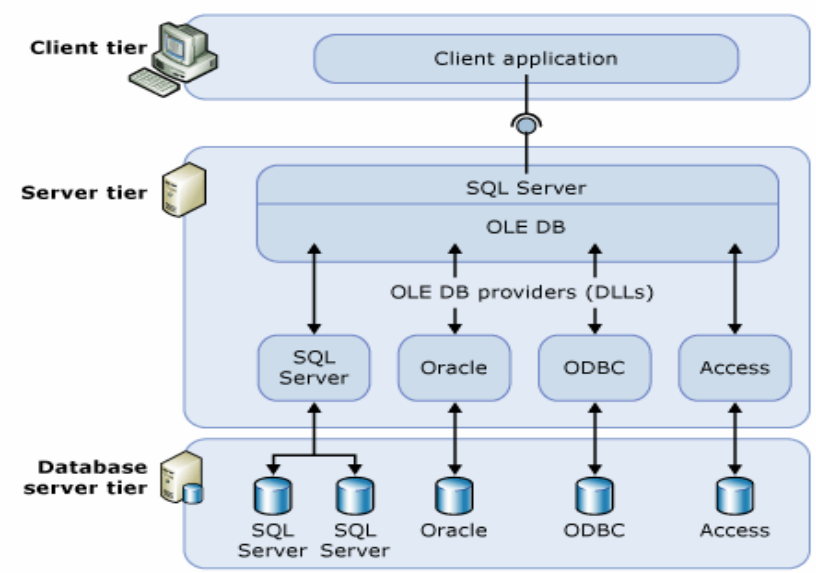
3.5 LINKED SERVERS (ΣΥΝΔΕΔΕΜΕΝΟΙ ΔΙΑΚΟΜΙΣΤΕΣ) ΣΤΟΝ SQL SERVER

Μια εξωτερική πηγή δεδομένων (OLE DB ή ODBC) μπορεί να χρησιμοποιηθεί μέσω του SQL Server αν δηλωθεί σαν συνδεμένος διακομιστής στον **SQL** Server. Ο συνδεδεμένος διακομιστής τότε θα δείχνει στην εξωτερική πηγή δεδομένων. Αυτή η εξωτερική πηγή δεδομένων μπορεί να είναι MySQL, MSAccess, Oracle, Excel ή σχεδόν οποιοδήποτε άλλο σύστημα δεδομένων που είναι προσβάσιμα μέσω OLE DB ή ODBC - συμπεριλαμβανομένων και άλλων διαφορετικών SQL servers.

Με έναν linked server, μπορούμε δεδομένα κατανεμημένα σε απομακρυσμένες τοποθεσίες να ανακτηθούν, να ενωθούν και να συνδυαστούν με τα τοπικά δεδομένα. Ενώ θα ήταν βολικό όλα τα δεδομένα μιας επιχείρησης να βρίσκονται συγκεντρωμένα σε μια τοποθεσία, υπάρχουν πολλά εμπόδια όπως οι εφαρμογές προμηθευτών που χτίζονται για ένα συγκεκριμένο κατάστημα στοιχείων, σύνολα δεδομένων πάρα πολύ μεγάλα για ένα μόνο

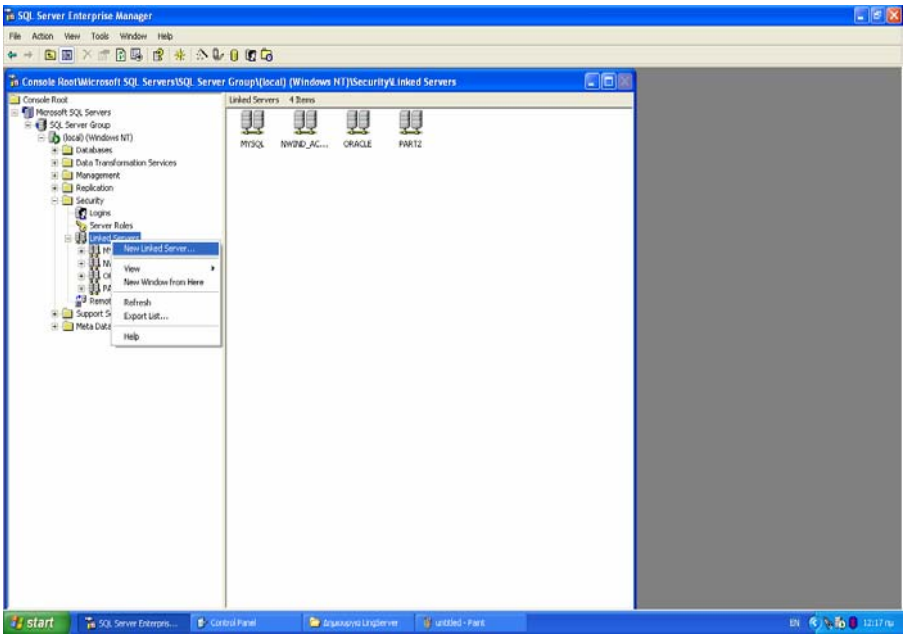
κεντρικό υπολογιστή, απαρχαιωμένες εφαρμογές flat αρχείων που το κόστος απαγορεύει να αναδημιουργηθούν και να αλλάξουν.

Εάν έχουμε ένα πολύ μεγάλο σύνολο δεδομένων, τότε ίσως μπορούν να υπάρξουν οφέλη στην κατάτμηση των δεδομένων, και στη μετακίνηση τους σε διαφορετικούς servers. Στη συνέχεια με τη χρήση διανεμημένων κατατμημένων προβολών (distributed partitioned views) παρουσιάζουμε τα δεδομένα ως μια πηγή. Σ' αυτή τη περίπτωση, οι linked servers είναι η τεχνολογία που το καθιστά ικανό.

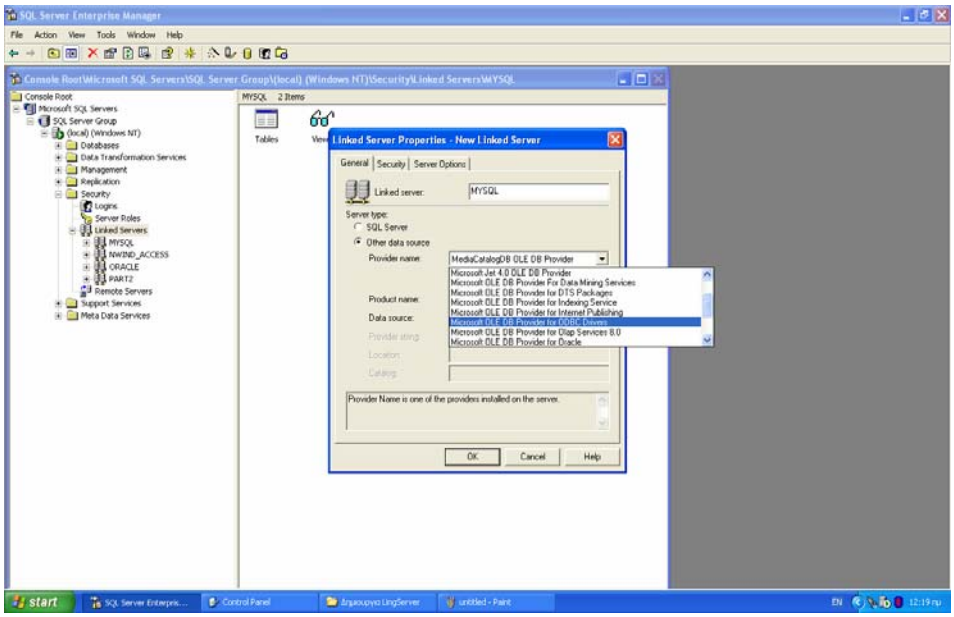


Για να δημιουργήσουμε ένα Linked Server θα πρέπει να φτιάξουμε πρώτα ένα Data Source που να είναι συνδεδεμένος με την βάση.

Μετά πηγαίνουμε στον SQL Server για να δημιουργήσουμε το Linked Server. Κάνουμε δεξί κλικ πάνω στο Linked Server και επιλέγουμε New Linked Server.

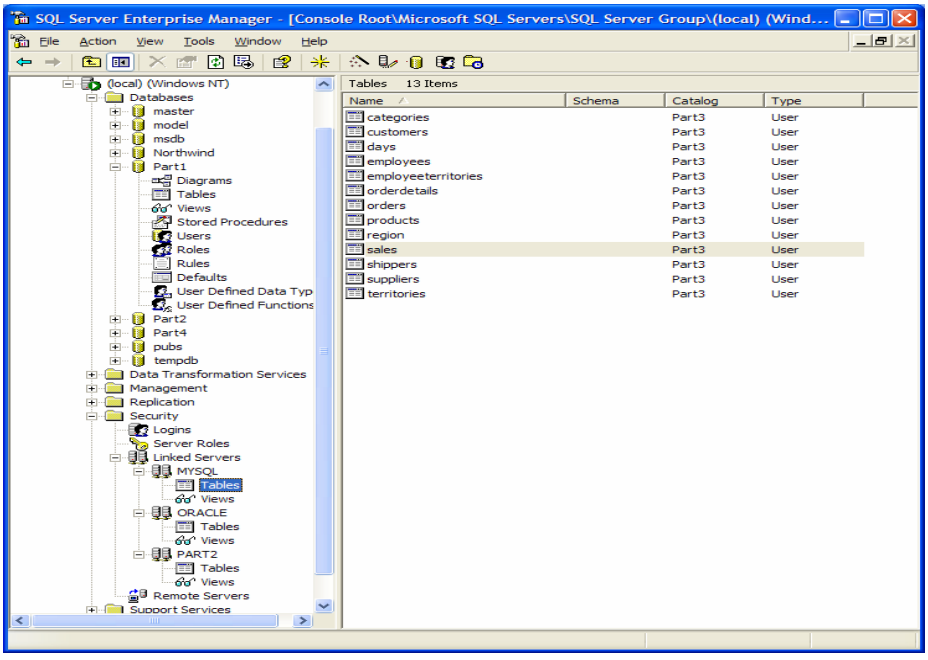


Στη συνέχεια συμπληρώνουμε τα πεδία Linked Server βάζοντας το όνομα που θέλουμε να έχει ο Linked Server και Provider name επιλέγουμε «Microsoft OLE DB Provider for ODBC Drivers» .



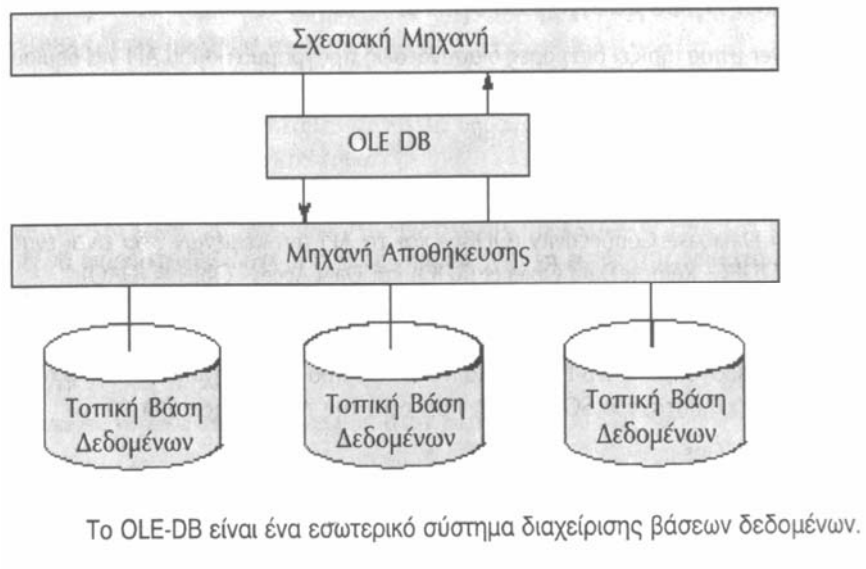
Στο πεδίο Data source γράφουμε το όνομα του Data source που δημιουργήσαμε στην αρχή το οποίο δείχνει στην βάση και πατάμε OK.

Όπως φαίνεται και στην εικόνα την διαδικασία αυτή την επαναλάβαμε τρεις φορές (φτιάξαμε τρεις Linked Server) για να συνδέσουμε και τις τρεις βάσεις μας που βρίσκονται σε διαφορετική τοποθεσία η κάθε μια μέσω του SQL Server.



3.6 Επικοινωνία Μέσα στον SQL Server

Ο διακομιστής βάσης δεδομένων του SQL Server έχει δυο κύρια μέρη: την σχεσιακή μηχανή (relational engine - RE) και τη μηχανή αποθήκευσης (storage engine - SE), όπως φαίνεται στην Εικόνα. Αυτή η επικοινωνία μεταξύ της σχεσιακής μηχανής και της μηχανής αποθήκευσης γίνετε μέσω διασυνδέσεων OLE-DB, που επιτρέπει στη σχεσιακή μηχανή να επεξεργάζεται ερωτήματα από οποιαδήποτε πηγή δεδομένων που έχει τέτοιες διασυνδέσεις.



Το OLE-DB είναι ένα εσωτερικό σύστημα διαχείρισης βάσεων δεδομένων.

3.7 Η ΑΠΟΘΗΚΗ ΔΕΔΟΜΕΝΩΝ ΣΤΗ ΤΟΠΟΘΕΣΙΑ 1

Αυτή η αποθήκη δεδομένων με όνομα Part1, βρίσκεται στην τοποθεσία με LocationID=1 σε μια **ΤΟΠΙΚΗ** βάση του SQL Server. Επίσης σε αυτήν τη βάση δημιουργήσαμε **όλες τις προβολές (views)** που ενώνουν τα συγκεντρωτικά στοιχεία των τεσσάρων αποθηκών σε μία.

Δημιουργία προβολής VW_SalesByEmployee

Η προβολή αυτή μας εμφανίζει τις συγκεντρωτικές πωλήσεις των πωλητών και από τις τέσσερις τοποθεσίες. Τα πεδία που επιλέγουμε είναι ο κωδικός του πελάτη, το ονοματεπώνυμο του εργαζόμενου, ο κωδικός του εργαζόμενου, το όνομα της εταιρίας και οι συνολικές πωλήσεις.

```
--drop view VW_SalesByEmployee
CREATE view VW_SalesByEmployee as
select o.employeeid,
  (e.FirstName + ' ' + e.LastName) as EmployeeName,
   c.Customerid, c.CompanyName,
  (quantity*unitprice) as TotalSales
from
(
  SELECT * FROM ORDERS
  UNION
  SELECT * FROM OPENQUERY(MYSQL, 'SELECT * FROM ORDERS')
  UNION
```

```
  SELECT * FROM OPENQUERY(PART2, 'SELECT * FROM ORDERS')
  UNION
  SELECT * FROM OPENQUERY(ORACLE, 'SELECT * FROM "orders"')
) o
inner join
(
  SELECT * FROM ORDERDETAILS
  UNION
  SELECT * FROM OPENQUERY(MYSQL, 'SELECT * FROM ORDERDETAILS')
  UNION
  SELECT * FROM OPENQUERY(PART2, 'SELECT * FROM ORDERDETAILS')
  UNION
  SELECT * FROM OPENQUERY(ORACLE, 'SELECT * FROM "orderdetails"')
) od on o.orderid=od.orderid
inner join
(
  SELECT * FROM Employees
  UNION
  SELECT * FROM OPENQUERY(MYSQL, 'SELECT * FROM Employees')
  UNION
  SELECT * FROM OPENQUERY(PART2, 'SELECT * FROM Employees')
  UNION
  SELECT * FROM OPENQUERY(ORACLE, 'SELECT * FROM "employees"')
) e on o.employeeid = e.employeeid
inner join
(
  SELECT * FROM Customers
  UNION
  SELECT * FROM OPENQUERY(MYSQL, 'SELECT * FROM Customers')
  UNION
  SELECT * FROM OPENQUERY(PART2, 'SELECT * FROM Customers')
  UNION
  SELECT * FROM OPENQUERY(ORACLE, 'SELECT * FROM "customers"')
) c on o.Customerid = c.Customerid
```

Για να δημιουργήσουμε την προβολή VW_SalesByEmployee επιλέγουμε τα πεδία employeeid, EmployeeName σαν ένωση του ονόματος και του επωνύμου και το TotalSales που είναι το αποτέλεσμα από τον πολλαπλασιασμό των πεδίων ποσότητα και τιμή μονάδος. Επιλέγουμε όλα τα πεδία από τον πίνακα Orders και από τις τέσσερις τοποθεσίες, καθώς επίσης και από τους πίνακες OrderDetails, Employees, Customers όπου το OrderID του πίνακα Order είναι ίδιο με το OrderID του πίνακα Order Details, το EmployeeID του πίνακα Orders είναι ίδιο με το EmployeeID του πίνακα Employees και το CustomerID του πίνακα Orders είναι ίδιο με το CustomersID του πίνακα Customers

Δημιουργία προβολής VW_SalesFromPart1

Η προβολή αυτή μας εμφανίζει τις συγκεντρωτικές πωλήσεις των πωλητών στην πρώτη τοποθεσία. Τα πεδία που επιλέγουμε είναι το επίθετο του εργαζόμενου, το όνομα της περιφέρειας που εργάζεται, το όνομα του προϊόντος, η χρονιά και οι συνολικές πωλήσεις.

```
CREATE view VW_SalesFromPart1 as
select p.ProductName as ProductName, d.DYear as DYear, s.TotalSales
as TotalSales, e.LastName as LastName, r.regionDescription as
RegionDescription
from sales s
  inner join product p on s.ProductID = p.ProductID
  inner join Days d on s.DayID = d.DayID
  inner join Employees e on s.SalesManID = e.EmployeeID
  inner join region r on s.LocationID = r.RegionID
```

Για να δημιουργήσουμε το ερώτημα VW_SalesFromPart1 επιλέγουμε τα πεδία που θέλουμε από τον πίνακα Sales στη βάση που βρίσκεται στην τοποθεσία 1 και μας εμφανίζει τα δεδομένα όπου το ProductID του πίνακα Sales είναι ίδιο με το ProductID του πίνακα Product, το DayID του πίνακα Sales είναι ίδιο με το DayID του πίνακα Days, το SalesManID του πίνακα Sales είναι ίδιο με το EmployeeID του πίνακα Employees και το LocationID του πίνακα Sales είναι ίδιο με το RegionID του πίνακα Region

Δημιουργία προβολής VW_SalesFromPart2

Η προβολή αυτή μας εμφανίζει τις συγκεντρωτικές πωλήσεις των πωλητών στην τοποθεσία 2. Τα πεδία που επιλέγουμε είναι το επίθετο του εργαζόμενου, το όνομα της περιφέρειας που εργάζεται, το όνομα του προϊόντος, η χρονιά και οι συνολικές πωλήσεις

```
CREATE view VW_SalesFromPart2 as
select o.ProductName, o.DYear,  o.TotalSales, o.LastName,
o.RegionDescription
from OPENQUERY(PART2,
'select p.ProductName, d.DYear, s.TotalSales, e.LastName,
r.regionDescription
from ((((sales s
  inner join product p on s.ProductID = p.ProductID)
  inner join Days d on s.DayID = d.DayID)
  inner join Employees e on s.SalesManID = e.EmployeeID)
  inner join region r on s.LocationID = r.RegionID)') o
```

Για να δημιουργήσουμε το ερώτημα VW_SalesFromPart2 επιλέγουμε τα πεδία που θέλουμε να εμφανίσουμε και τα δεδομένα τα παίρνουμε από τον πίνακα Sales στη βάση που βρίσκεται στην τοποθεσία 2 και μας εμφανίζονται τα δεδομένα όπου το ProductID του πίνακα Sales είναι ίδιο με το ProductID του πίνακα Product, το DayID του πίνακα Sales είναι ίδιο με το DayID του πίνακα Days, το SalesManID του πίνακα Sales είναι ίδιο με το EmployeeID του πίνακα Employees και το LocationID του πίνακα Sales είναι ίδιο με το RegionID του πίνακα Region

Δημιουργία προβολής VW_SalesFromPart3

Η προβολή αυτή μας εμφανίζει τις συγκεντρωτικές πωλήσεις των πωλητών στην τρίτη τοποθεσία. Τα πεδία που επιλέγουμε είναι το επίθετο του εργαζόμενου, το όνομα της περιφέρειας που εργάζεται, το όνομα του προϊόντος, η χρονιά και οι συνολικές πωλήσεις

```
CREATE view VW_SalesFromPart3 as
select ProductName, DYear,  TotalSales,  LastName, RegionDescription
from OPENQUERY(MYSQL,
'select p.ProductName, d.DYear, s.TotalSales, e.LastName,
r.regionDescription
from sales s
  inner join products p on s.ProductID = p.ProductID
  inner join Days d on s.DayID = d.DayID
  inner join Employees e on s.SalesManID = e.EmployeeID
  inner join region r on s.LocationID = r.RegionID') o
```

Για να δημιουργήσουμε το ερώτημα VW_SalesFromPart3 επιλέγουμε τα πεδία που θέλουμε να εμφανίσουμε και τα δεδομένα τα παίρνουμε από τον πίνακα Sales στη βάση που βρίσκεται στην τοποθεσία 3 και μας εμφανίζονται τα δεδομένα όπου το ProductID του πίνακα Sales είναι ίδιο με το ProductID του πίνακα Product, το DayID του πίνακα Sales είναι ίδιο με το DayID του πίνακα Days, το SalesManID του πίνακα Sales είναι ίδιο με το EmployeeID του πίνακα Employees και το LocationID του πίνακα Sales είναι ίδιο με το RegionID του πίνακα Region

Δημιουργία προβολής VW_SalesFromPart4

Η προβολή αυτή μας εμφανίζει τις συγκεντρωτικές πωλήσεις των πωλητών στην τέταρτη τοποθεσία. Τα πεδία που επιλέγουμε είναι το επίθετο του εργαζόμενου, το όνομα της περιφέρειας που εργάζεται, το όνομα του προϊόντος, η χρονιά και οι συνολικές πωλήσεις

```
CREATE view VW_SalesFromPart4 as
select ProductName, DYear,  TotalSales,  LastName, RegionDescription
from OPENQUERY(ORACLE,
'select p.ProductName, d.DYear, s.TotalSales, e.LastName,
r.regionDescription
from SALES s
  inner join "products" p on s.ProductID = p.ProductID
  inner join DAYS d on s.DayID = d.DayID
  inner join "employees" e on s.SalesManID = e.EmployeeID
  inner join "region" r on s.LocationID = r.RegionID') o
```

Για να δημιουργήσουμε το ερώτημα VW_SalesFromPart4 επιλέγουμε τα πεδία που θέλουμε να εμφανίσουμε και τα δεδομένα τα παίρνουμε από τον πίνακα Sales στη βάση που βρίσκεται στην τοποθεσία 4 και μας εμφανίζονται τα δεδομένα όπου το ProductID του πίνακα Sales είναι ίδιο με το ProductID του πίνακα Product, το DayID του πίνακα Sales είναι ίδιο με το DayID του πίνακα Days, το SalesManID του πίνακα Sales είναι ίδιο με το EmployeeID του πίνακα Employees και το LocationID του πίνακα Sales είναι ίδιο με το RegionID του πίνακα Region

Δημιουργία προβολής VW_AllSales

Η προβολή αυτή μας εμφανίζει τις συγκεντρωτικές πωλήσεις των πωλητών και από τις

τέσσερις τοποθεσίες μαζί. Τα πεδία που επιλέγουμε είναι το επίθετο του εργαζόμενου, το όνομα της περιφέρειας που εργάζεται, το όνομα του προϊόντος, η χρονιά και οι συνολικές πωλήσεις

```
CREATE VIEW VW_AllSales as
SELECT ProductName, DYear, LastName, RegionDescription, TotalSales
FROM VW_SalesFromPart1
UNION
SELECT ProductName, DYear, LastName, RegionDescription, TotalSales
FROM VW_SalesFromPart2
UNION
SELECT ProductName, DYear, LastName, RegionDescription, TotalSales
FROM VW_SalesFromPart3
UNION
SELECT ProductName, DYear, LastName, RegionDescription, TotalSales
FROM VW_SalesFromPart4
```

Για να δημιουργήσουμε το ερώτημα επιλέγουμε τα πεδία που θέλουμε από τα ερωτήματα VW_SalesFromPart1,VW_SalesFromPart2,VW_SalesFromPart3

VW_SalesFromPart4 και τα ενώνουμε

Δημιουργία προβολής VW_SalesByDay1

Η προβολή αυτή μας εμφανίζει τις συγκεντρωτικές πωλήσεις των προϊόντων στην πρώτη τοποθεσία. Τα πεδία που επιλέγουμε είναι το όνομα του προϊόντος, την χρονιά, τον μήνα, το τρίμηνο, τις συνολικές πωλήσεις, το επίθετο του εργαζόμενου και το όνομα της περιοχής που εργάζεται ο εργαζόμενος

```
CREATE view VW_SalesByDay1 as
select p.ProductName as ProductName, d.DYear as DYear, d.DMonth as
DMonth, d.DQuarter as DQuarter, s.TotalSales as TotalSales,
e.LastName as LastName, r.regionDescription as RegionDescription
from sales s
  inner join product p on s.ProductID = p.ProductID
  inner join Days d on s.DayID = d.DayID
  inner join Employees e on s.SalesManID = e.EmployeeID
  inner join region r on s.LocationID = r.RegionID
```

Για να δημιουργήσουμε το ερώτημα VW_SalesByDay1 επιλέγουμε τα πεδία που θέλουμε από τον πίνακα Sales στη βάση που βρίσκεται στην τοποθεσία 1 και μας εμφανίζει τα δεδομένα όπου το ProductID του πίνακα Sales είναι ίδιο με το ProductID του πίνακα Product, το DayID του πίνακα Sales είναι ίδιο με το DayID του πίνακα Days, το SalesManID του πίνακα Sales είναι ίδιο με το EmployeeID του πίνακα Employees και το LocationID του πίνακα Sales είναι ίδιο με το RegionID του πίνακα Region

Δημιουργία προβολής VW_SalesByDay2

Η προβολή αυτή μας εμφανίζει τις συγκεντρωτικές πωλήσεις των προϊόντων στην δεύτερη τοποθεσία. Τα πεδία που επιλέγουμε είναι το όνομα του προϊόντος, την χρονιά, τον μήνα, το τρίμηνο, τις συνολικές πωλήσεις, το επίθετο του εργαζόμενου και το όνομα της περιοχής που εργάζεται ο εργαζόμενος

```
CREATE view VW_SalesByDay2 as
select o.ProductName, o.DYear, o.DMonth, o.DQuarter, o.TotalSales,
o.LastName, o.RegionDescription
from OPENQUERY(PART2,
'select p.ProductName, d.DYear, d.DQuarter, d.DMonth, s.TotalSales,
e.LastName, r.regionDescription
from ((((sales s
  inner join product p on s.ProductID = p.ProductID)
  inner join Days d on s.DayID = d.DayID)
  inner join Employees e on s.SalesManID = e.EmployeeID)
  inner join region r on s.LocationID = r.RegionID)') o
```

Για να δημιουργησουμε το ερώτημα VW_SalesByDay2 επιλέγουμε τα πεδία που θέλουμε να εμφανίσουμε και τα δεδομένα τα περνούμε από τον πίνακα Sales στη βάση που βρίσκεται στην τοποθεσία 2 και μας εμφανίζονται τα δεδομένα όπου το ProductID του πίνακα Sales είναι ίδιο με το ProductID του πίνακα Product, το DayID του πίνακα Sales είναι ίδιο με το DayID του πίνακα Days, το SalesManID του πίνακα Sales είναι ίδιο με το EmployeeID του πίνακα Employees και το LocationID του πίνακα Sales είναι ίδιο με το RegionID του πίνακα Region

Δημιουργία προβολής VW_SalesByDay3

Η προβολή αυτή μας εμφανίζει τις συγκεντρωτικές πωλήσεις των προϊόντων στην τρίτη τοποθεσία. Τα πεδία που επιλέγουμε είναι το όνομα του προϊόντος, την χρονιά, τον μήνα, το τρίμηνο, τις συνολικές πωλήσεις, το επίθετο του εργαζόμενου και το όνομα της περιοχής που εργάζεται ο εργαζόμενος

```
CREATE view VW_SalesByDay3 as
select ProductName, DYear,  DMonth, DQuarter, TotalSales,  LastName,
RegionDescription
from OPENQUERY(MYSQL,
'select p.ProductName, d.DYear, DMonth, DQuarter, s.TotalSales,
e.LastName, r.regionDescription
from sales s
  inner join products p on s.ProductID = p.ProductID
  inner join Days d on s.DayID = d.DayID
  inner join Employees e on s.SalesManID = e.EmployeeID
  inner join region r on s.LocationID = r.RegionID') o
```

Για να δημιουργήσουμε το ερώτημα VW_SalesByDay3 επιλέγουμε τα πεδία που θέλουμε να εμφανίσουμε και τα δεδομένα τα παίρνουμε από τον πίνακα Sales στη βάση που βρίσκεται στην τοποθεσια 3 και μας εμφανίζονται τα δεδομένα όπου το ProductID του πίνακα Sales είναι ίδιο με το ProductID του πίνακα Product, το DayID του πίνακα Sales είναι ίδιο με το DayID του πίνακα Days, το SalesManID του πίνακα Sales είναι ίδιο με το EmployeeID του

πίνακα Employees και το LocationID του πίνακα Sales είναι ίδιο με το RegionID του πίνακα Region

Δημιουργία προβολής VW_SalesByDay4

Η προβολή αυτή μας εμφανίζει τις συγκεντρωτικές πωλήσεις των προϊόντων στην τέταρτη τοποθεσία. Τα πεδία που επιλέγουμε είναι το όνομα του προϊόντος, την χρονιά, τον μήνα, το τρίμηνο, τις συνολικές πωλήσεις, το επίθετο του εργαζόμενου και το όνομα της περιοχής που εργάζεται ο εργαζόμενος

```
CREATE view VW_SalesByDay4 as
select ProductName, DYear,  DMonth, DQuarter,  TotalSales,  LastName,
RegionDescription  from OPENQUERY(ORACLE,
'select p.ProductName, d.DYear, d.DMonth, d.DQuarter, s.TotalSales,
e.LastName, r.regionDescription
from SALES s
  inner join "products" p on s.ProductID = p.ProductID
  inner join DAYS d on s.DayID = d.DayID
  inner join "employees" e on s.SalesManID = e.EmployeeID
  inner join "region" r on s.LocationID = r.RegionID') o
```

Για να δημιουργησουμε το ερώτημα VW_SalesByDay4 επιλέγουμε τα πεδία που θέλουμε να εμφανίσουμε και τα δεδομένα τα παίρνουμε από τον πίνακα Sales στη βάση που βρίσκεται στην τοποθεσία 4 και μας εμφανίζονται τα δεδομένα όπου το ProductID του πίνακα Sales είναι ίδιο με το ProductID του πίνακα Product, το DayID του πίνακα Sales είναι ίδιο με το DayID του πίνακα Days, το SalesManID του πίνακα Sales είναι ίδιο με το EmployeeID του πίνακα Employees και το LocationID του πίνακα Sales είναι ίδιο με το RegionID του πίνακα Region

Δημιουργία προβολής VW_SalesByDayAll

Η προβολή αυτή μας εμφανίζει τις συγκεντρωτικές πωλήσεις των προϊόντων και στις τέσσερις τοποθεσίες μαζί.

```
CREATE VIEW VW_SalesByDayAll
AS
 SELECT * FROM VW_SalesByDay1
 UNION
 SELECT * FROM VW_SalesByDay2
 UNION
 SELECT * FROM VW_SalesByDay3
 UNION
 SELECT * FROM VW_SalesByDay4
```

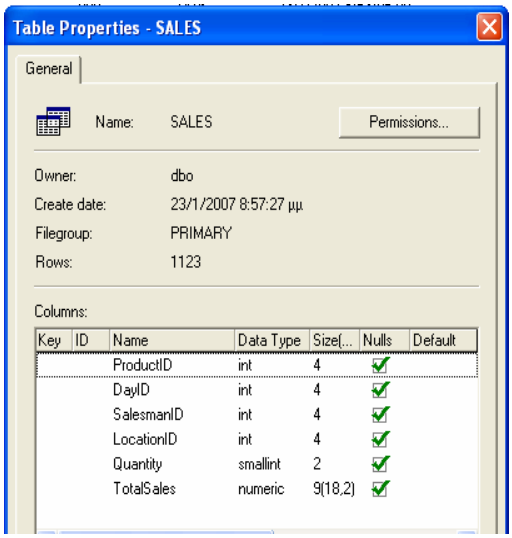
Για να δημιουργήσουμε το ερώτημα VW_SalesByDayAll όπου μας εμφανίζει στις συγκεντρωτικές πωλήσεις και από τις τέσσερις βάσεις επιλέγουμε όλα τα πεδία και από τις τέσσερις βάσεις και τα ενώνουμε μεταξύ τους

3.7.1 Η Δημιουργία των πινάκων και η εισαγωγή των δεδομένων

- Δημιουργία του πίνακα SALES

```
CREATE TABLE SALES
       (
        Product int NULL,
        DayID int NULL,
        SalesmanID int NULL,
        LocationID int NULL,
        Quantity smallint NULL,
        TotalSales numeric(18,2) NULL
       )
```

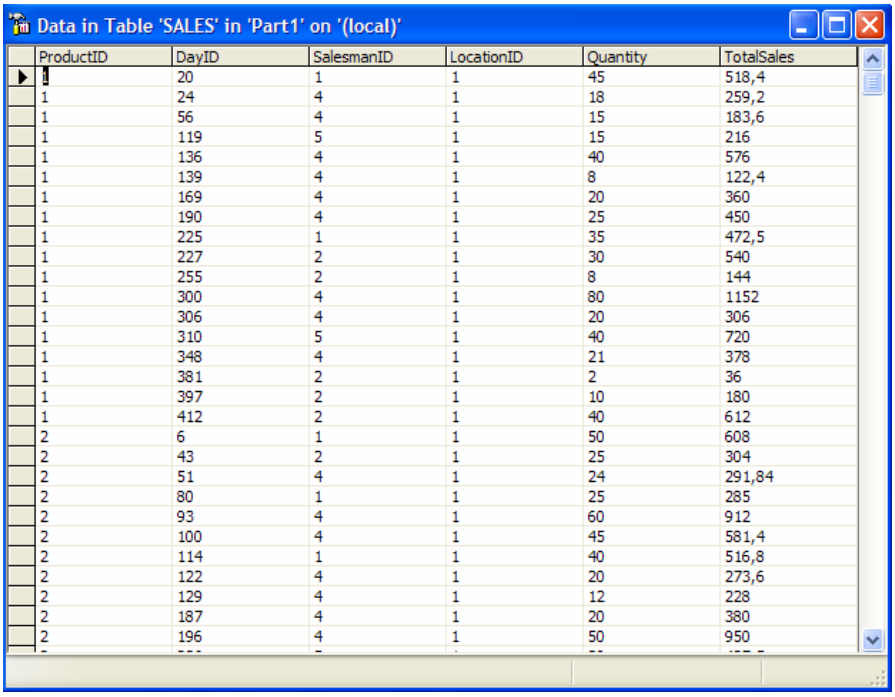
Ο πίνακας SALES στη βάση δεδομένων στον SQL Server



- Εισαγωγή δεδομένων στον πίνακα SALES

```
DELETE FROM Sales;
INSERT INTO Sales (ProductID, DayID, SalesmanID, LocationID,
Quantity, TotalSales)
select p.ProductId,
       d.DayId,
       e.EmployeeID,
       1 as LocationID,
       od.quantity,
       (od.quantity * od.unitprice) * (1 - od.discount) as TotalSales
from ORDERS o
Inner join Days d on o.OrderId = d.OrderID
inner join ORDERDETAILS od on o.orderID=od.orderID
inner join EMPLOYEES e on e.EmployeeID=o.EmployeeID
inner join PRODUCT p on p.ProductID=od.ProductID
```

Δείγμα δεδομένων του πίνακα SALES στον SQL Server

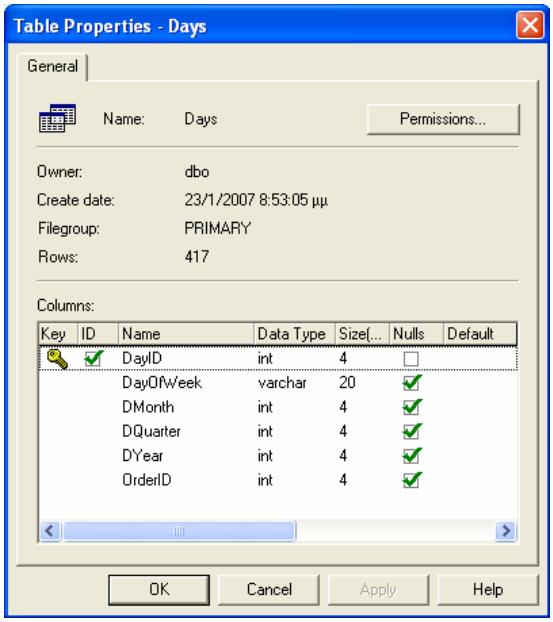

Data in Table 'SALES' in 'Part1' on '(local)'

ProductID	DayID	SalesmanID	LocationID	Quantity	TotalSales
1	20	1	1	45	518,4
1	24	4	1	18	259,2
1	56	4	1	15	183,6
1	119	5	1	15	216
1	136	4	1	40	576
1	139	4	1	8	122,4
1	169	4	1	20	360
1	190	4	1	25	450
1	225	1	1	35	472,5
1	227	2	1	30	540
1	255	2	1	8	144
1	300	4	1	80	1152
1	306	4	1	20	306
1	310	5	1	40	720
1	348	4	1	21	378
1	381	2	1	2	36
1	397	2	1	10	180
1	412	2	1	40	612
2	6	1	1	50	608
2	43	2	1	25	304
2	51	4	1	24	291,84
2	80	1	1	25	285
2	93	4	1	60	912
2	100	4	1	45	581,4
2	114	1	1	40	516,8
2	122	4	1	20	273,6
2	129	4	1	12	228
2	187	4	1	20	380
2	196	4	1	50	950

- Δημιουργία του πίνακα DAYS

```
create table Days(DayID int NOT NULL IDENTITY (1, 1),
                  DayOfWeek varchar(20),
                  DMonth int,
                  DQuarter int,
                  DYear int,
                  OrderID int NULL,
                  Primary Key (DayID))
```

Ο πίνακας DAYS στη βάση δεδομένων στον SQL Server



- Εισαγωγή δεδομένων στον πίνακα DAYS

```
insert into Days(DayOfWeek, DMonth, DQuarter, DYear, OrderID)
select DATENAME (dw, OrderDate), Month(orderDate), DATENAME(quarter,
OrderDate), Year(orderDate), OrderID from orders
```

Η πρώτη συνάρτηση στο select εξάγει την ημέρα της εβδομάδος από μια ημερομηνία, η δεύτερη συνάρτηση εξάγει τον μήνα, η τρίτη το τρίμηνο, και η τέταρτη το έτος.

Δείγμα δεδομένων του πίνακα DAYS στον SQL Server

Data in Table 'Days' in 'Part1' on '(local)'

DayID	DayOfWeek	DMonth	DQuarter	DYear	OrderID
1	Thursday	7	3	1996	10248
2	Monday	7	3	1996	10250
3	Tuesday	7	3	1996	10252
4	Thursday	7	3	1996	10254
5	Tuesday	7	3	1996	10257
6	Wednesday	7	3	1996	10258
7	Thursday	7	3	1996	10259
8	Friday	7	3	1996	10260
9	Friday	7	3	1996	10261
10	Thursday	7	3	1996	10265
11	Monday	7	3	1996	10267
12	Wednesday	7	3	1996	10269
13	Thursday	8	3	1996	10270
14	Wednesday	8	3	1996	10275
15	Friday	8	3	1996	10277
16	Wednesday	8	3	1996	10280

Ο πίνακας Products στη βάση δεδομένων στον SQL Server

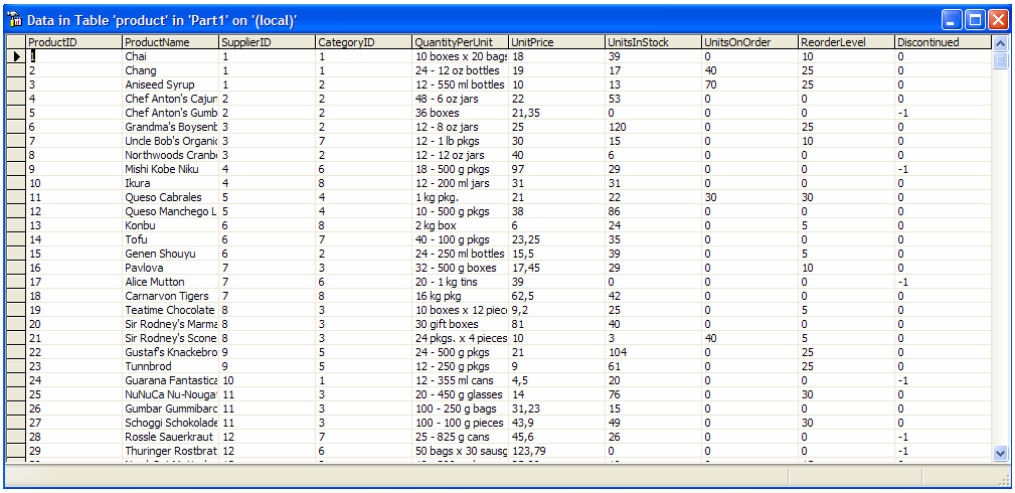
Design Table 'product' in 'Part1' on '(lo...

Column Name	Data Type	Length	Allow Nulls
ProductID	int	4	
ProductName	varchar	100	✓
SupplierID	int	4	✓
CategoryID	int	4	✓
QuantityPerUnit	varchar	100	✓
UnitPrice	numeric	9	✓
UnitsInStock	smallint	2	✓
UnitsOnOrder	smallint	2	✓
ReorderLevel	smallint	2	✓
Discontinued	smallint	2	✓

Columns

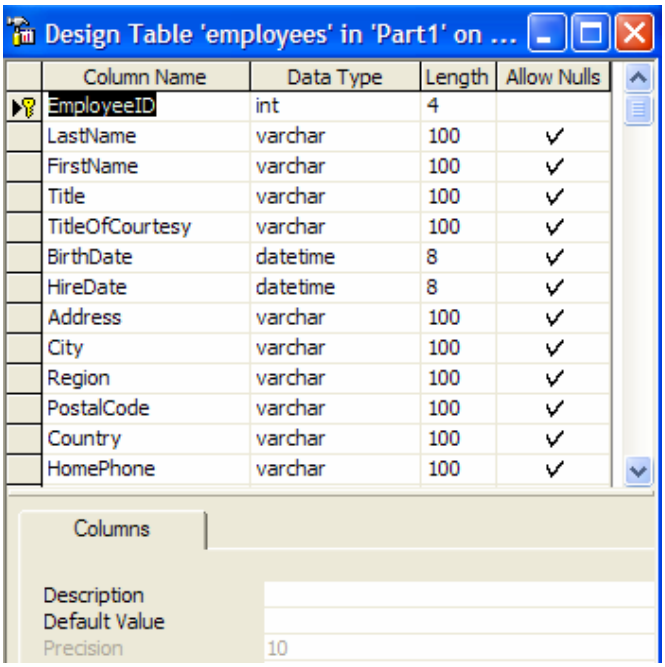
Description	
Default Value	
Precision	10
Scale	0
Identity	No
Identity Seed	

Δείγμα δεδομένων του πίνακα Products στον SQL Server

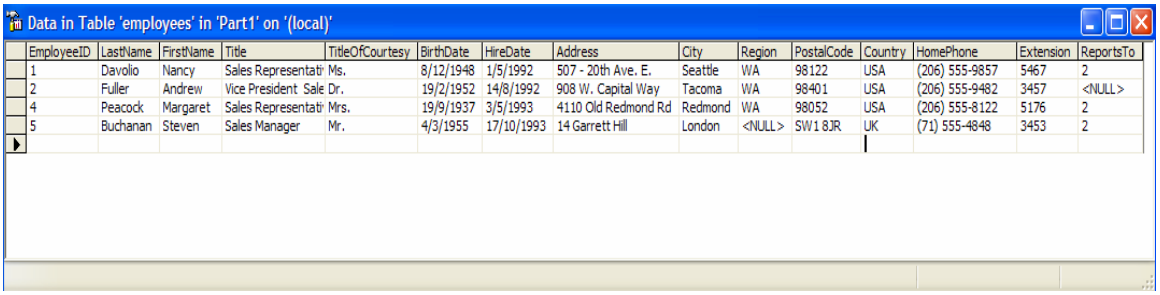


ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
1	Chai	1	1	10 boxes x 20 bag:	18	39	0	10	0
2	Chang	1	1	24 - 12 oz bottles	19	17	40	25	0
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10	13	70	25	0
4	Chef Anton's Cajur	2	2	48 - 6 oz jars	22	53	0	0	0
5	Chef Anton's Gumb	2	2	36 boxes	21,35	0	0	0	-1
6	Grandma's Boysent	3	2	12 - 8 oz jars	25	120	0	25	0
7	Uncle Bob's Organi	3	7	12 - 1 lb pkgs	30	15	0	10	0
8	Northwoods Cranb	3	2	12 - 12 oz jars	40	6	0	0	0
9	Mishi Kobe Niku	4	6	18 - 500 g pkgs	97	29	0	0	-1
10	Ikura	4	8	12 - 200 ml jars	31	31	0	0	0
11	Queso Cabrales	5	4	1 kg pkg.	21	22	30	30	0
12	Queso Manchego L	5	4	10 - 500 g pkgs	38	86	0	0	0
13	Konbu	6	8	2 kg box	6	24	0	5	0
14	Tofu	6	7	40 - 100 g pkgs	23,25	35	0	0	0
15	Genen Shouyu	6	2	24 - 250 ml bottles	15,5	39	0	5	0
16	Pavlova	7	3	32 - 500 g boxes	17,45	29	0	10	0
17	Alice Mutton	7	6	20 - 1 kg tins	39	0	0	0	-1
18	Carnarvon Tigers	7	8	16 kg pkg	62,5	42	0	0	0
19	Teatime Chocolate	8	3	10 boxes x 12 piec	9,2	25	0	5	0
20	Sir Rodney's Marma	8	3	30 gift boxes	81	40	0	0	0
21	Sir Rodney's Scone	8	3	24 pkgs. x 4 pieces	10	3	40	5	0
22	Gustaf's Knackebro	9	5	24 - 500 g pkgs	21	104	0	25	0
23	Tunnbrod	9	5	12 - 250 g pkgs	9	61	0	25	0
24	Guarana Fantastica	10	1	12 - 355 ml cans	4,5	20	0	0	-1
25	NuNuCa Nu-Nougat	11	3	20 - 450 g glasses	14	76	0	30	0
26	Gumbar Gummibarc	11	3	100 - 250 g bags	31,23	15	0	0	0
27	Schoggi Schokolade	11	3	100 - 100 g pieces	43,9	49	0	30	0
28	Rossle Sauerkraut	12	7	25 - 825 g cans	45,6	26	0	0	-1
29	Thuringer Rostbrat	12	6	50 bags x 30 sausg	123,79	0	0	0	-1

Ο πίνακας Employees στη βάση δεδομένων στον SQL Server

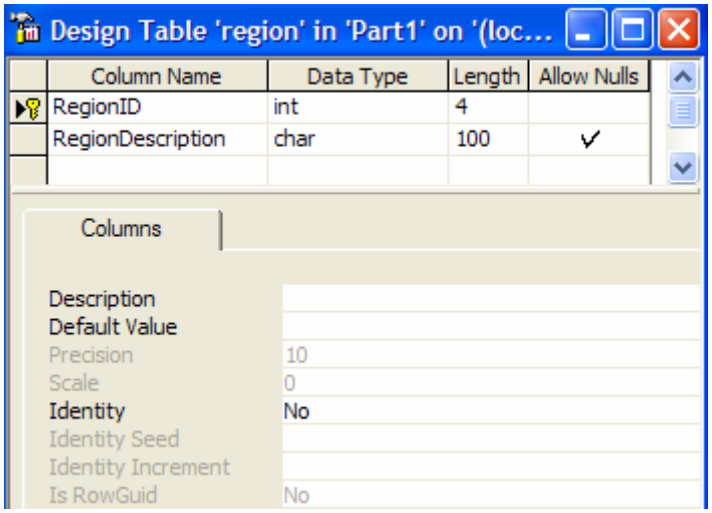


Δείγμα δεδομένων του πίνακα Employees στον SQL Server

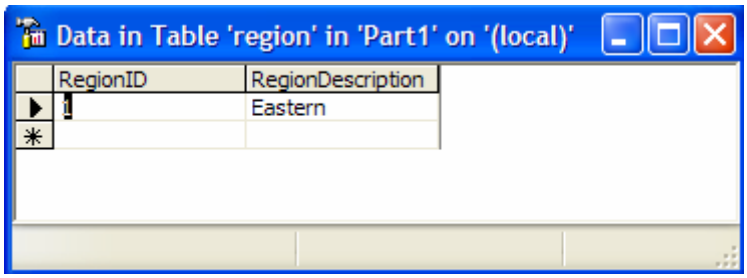


EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate	Address	City	Region	PostalCode	Country	HomePhone	Extension	ReportsTo
1	Davolio	Nancy	Sales Representati	Ms.	8/12/1948	1/5/1992	507 - 20th Ave. E.	Seattle	WA	98122	USA	(206) 555-9857	5467	2
2	Fuller	Andrew	Vice President, Sale	Dr.	19/2/1952	14/8/1992	908 W. Capital Way	Tacoma	WA	98401	USA	(206) 555-9482	3457	<NULL>
4	Peacock	Margaret	Sales Representati	Mrs.	19/9/1937	3/5/1993	4110 Old Redmond Rd	Redmond	WA	98052	USA	(206) 555-8122	5176	2
5	Buchanan	Steven	Sales Manager	Mr.	4/3/1955	17/10/1993	14 Garrett Hill	London	<NULL>	SW1 8JR	UK	(71) 555-4848	3453	2

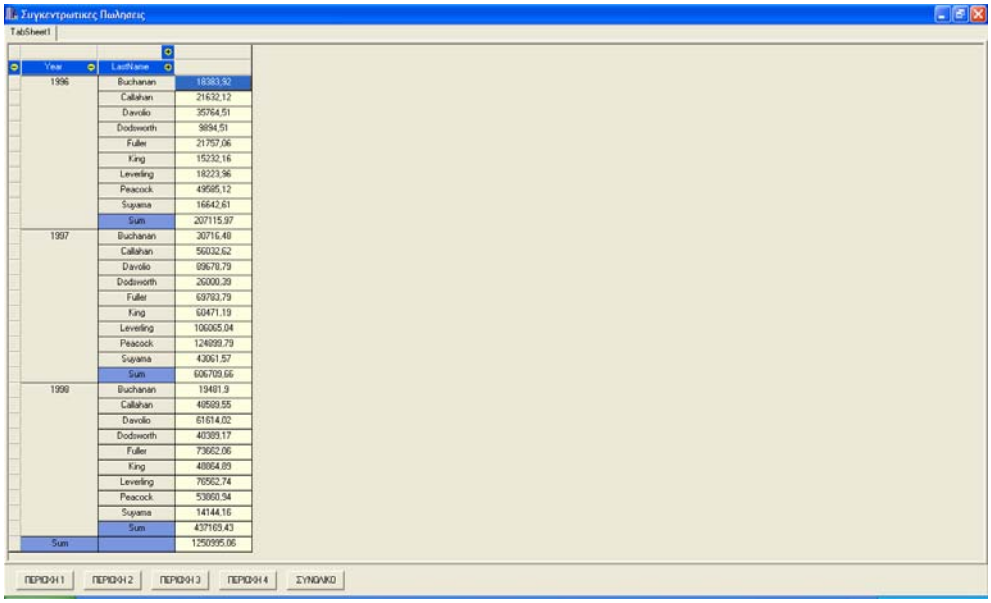
Ο πίνακας Region στη βάση δεδομένων στον SQL Server



Δείγμα δεδομένων του πίνακα Employees στον SQL Server

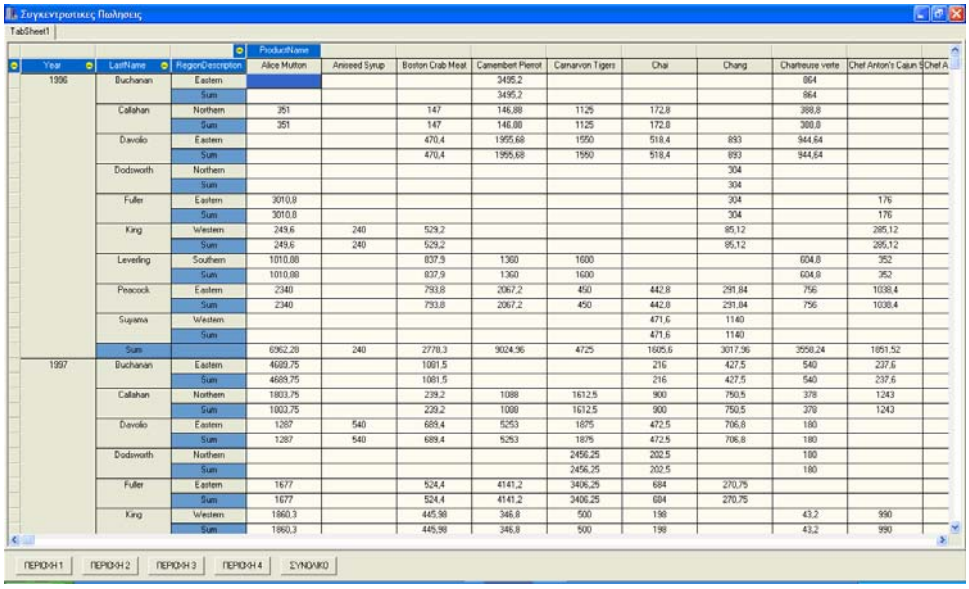


- **Συναθροιστική άνοδος (λειτουργία roll up)** (μείωση διαστάσεων, συνάθροιση): Η συναθροιστική άνοδος επιτρέπει στο χρήστη να κάνει ερωτήσεις κινούμενος στην ιεραρχία προς τα πάνω. Η λειτουργία της μετακίνησης από τη λεπτότερη κατάτμηση των δεδομένων σε μια πιο γενική κατάτμηση ονομάζεται **rollup**. Έτσι μπορούμε για παράδειγμα να δούμε τις συνολικές πωλήσεις .



Εικόνα από την εφαρμογή μας

- **Αναλυτική κάθοδος (λειτουργία drill down):** Η αντίστροφη λειτουργία δηλαδή, η μετακίνηση από την πιο γενική κατάτμηση στην πιο λεπτομερή, ονομάζεται **drill down**.



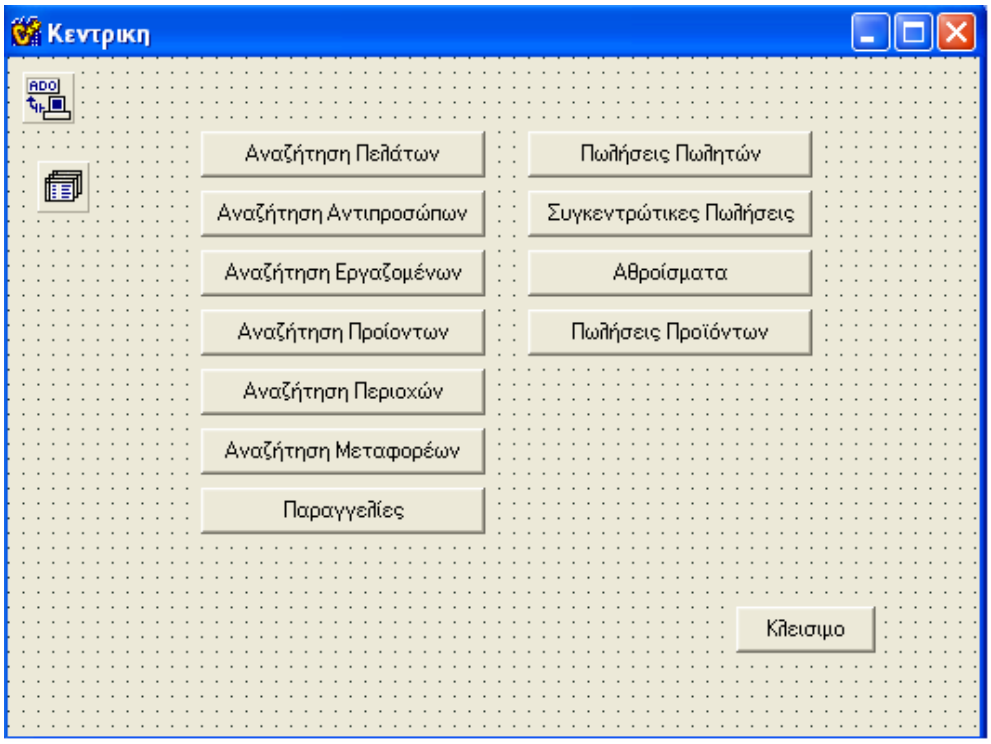
Εικόνα από την εφαρμογή μας

4 ΥΛΟΠΟΙΗΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

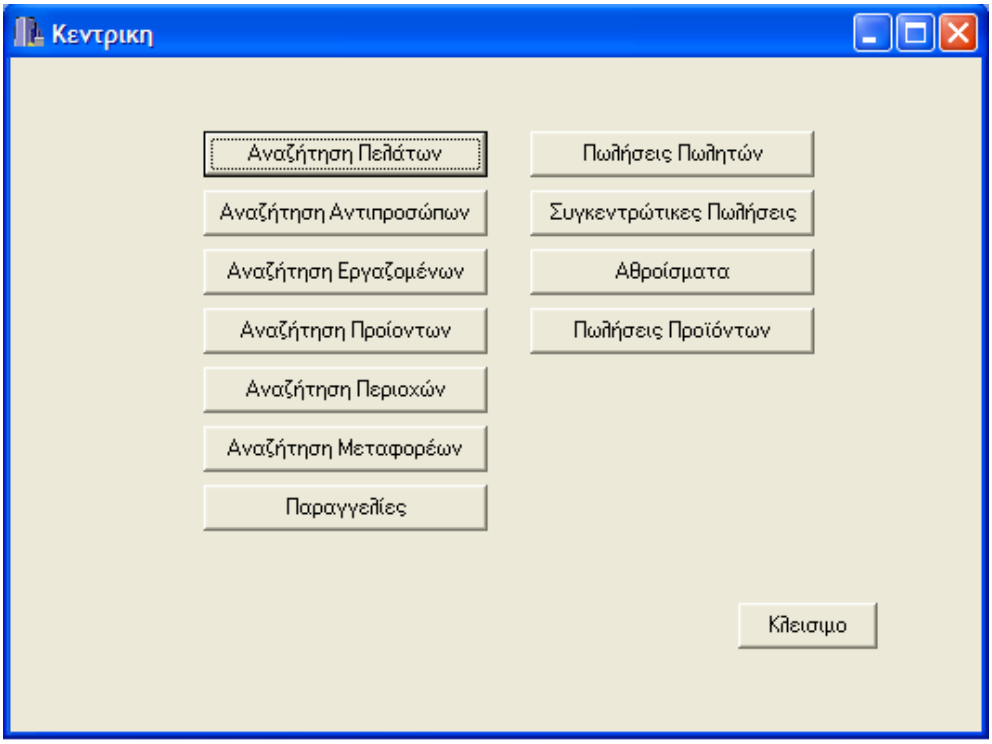
Στην παράγραφο αυτή θα περιγράψουμε τις φόρμες από τις οποίες αποτελείται η εφαρμογή μας. Για κάθε φόρμα δίνεται μια περιγραφή για το τι κανει, στη συνεχεία παρουσιάζεται η φόρμα (σε χρόνο σχεδίασης και σε χρόνο εκτέλεσης) και ο κώδικας της κάθε φόρμας με αναλυτικά σχόλια για το πώς δημιουργήθηκε. Τέλος παρουσιάζονται αναλυτικά σε πίνακα τα αντικείμενα που χρησιμοποιήσαμε για την δημιουργία κάθε φόρμας

4.1.1 Κεντρική Φόρμα

Είναι η πρώτη φόρμα που εμφανίζεται όταν ανοίξουμε το πρόγραμμα Μέσω αυτής μπορούμε να περιηγηθούμε σε όλες τις φόρμες του προγράμματος. Για την δημιουργία αυτής της φόρμας χρησιμοποιήθηκαν τα παρακάτω αντικείμενα: **ADOConnection, Database, Button.** Το αντικείμενο ADOConnection το χρησιμοποιούμε για να δημιουργήσουμε μία σύνδεση που θα χρησιμοποιήσουν όλα τα ADOQueries στις υπόλοιπες φόρμες. Το αντικείμενο Database το χρησιμοποιούν για να μπορούμε να έχουμε πρόσβαση μέσα στην βάση μέσω των αντικειμένων DecisionCube. Τα Button τα χρησιμοποιούμε για να μεταφερόμαστε στις υπόλοιπες φόρμες, όπου κάθε Button έχει το όνομα της αντίστοιχης φόρμας και ένα Button που βγαίνουμε από την εφαρμογή.



Εικόνα. Η φόρμα σε χρόνο σχεδίασης (Disign Time)



Εικόνα. Η φόρμα σε χρόνο εκτέλεσης (Run Time)

```
//---------------------------------------------------------------------
-
#include <vcl.h>
#pragma hdrstop

#include "MainForm.h"
#include "SearchCustForm.h"
#include "SearchSuplForm.h"
#include "SearchEmpForm.h"
#include "ProductsForm.h"
#include "RegionForm.h"
#include "ShippersForm.h"
#include "OrdersForm.h"
#include "CubeForm1.h"
#include "CubeForm2.h"
#include "frmSums.h"
#include "DaysCubeForm.h"
//---------------------------------------------------------------------
-
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmMain *frmMain;
//---------------------------------------------------------------------
-
void __fastcall TfrmMain::btnSearchCustomerClick(TObject *Sender)
{
  TfrmSearchCustomer *frm = new TfrmSearchCustomer(this);
```

```
  this->Hide();
  frm->ADOQuery1->Connection = ADOConnection1;
  frm->ADOQuery2->Connection = ADOConnection1;
  frm->Show();
}
```

Κρύβουμε την κεντρική φόρμα, συνδέουμε τα Query1 & Query2 με το ADOConection1, εμφανίζουμε την φόρμα Αναζήτηση Πελατών

```
//---------------------------------------------------------------------
-
void __fastcall TfrmMain::btnSearchSuppierClick(TObject *Sender)
{
  TfrmSearchSupplier *frm = new TfrmSearchSupplier(this);
  frm->ADOQuery1->Connection = ADOConnection1;
  frm->ADOQuery2->Connection = ADOConnection1;
  this->Hide();
  frm->Show();
}
```

Κρύβουμε την κεντρική φόρμα, συνδέουμε τα Query1 & Query2 με το ADOConection1, εμφανίζουμε την φόρμα Αναζήτηση Προμηθευτών

```
//---------------------------------------------------------------------
-
void __fastcall TfrmMain::Button1Click(TObject *Sender)
{
  TfrmSearchEmp *frm = new TfrmSearchEmp(this);
  frm->ADOQuery1->Connection = ADOConnection1;
  frm->ADOQuery2->Connection = ADOConnection1;
  this->Hide();
  frm->Show();
}
```

Κρύβουμε την κεντρική φόρμα, συνδέουμε τα Query1 & Query2 με το ADOConection1, εμφανίζουμε την φόρμα Αναζήτηση Εργαζομένων

```
//---------------------------------------------------------------------
-
void __fastcall TfrmMain::Button2Click(TObject *Sender)
{
  TfrmProducts *frm = new TfrmProducts(this);
  frm->ADOQuery1->Connection = ADOConnection1;
  frm->ADOQuery2->Connection = ADOConnection1;
  this->Hide();
  frm->Show();
}
```

Κρύβουμε την κεντρική φόρμα, συνδέουμε τα Query1 & Query2 με το ADOConection1, εμφανίζουμε την φόρμα Αναζήτηση Προϊόντων

```
//---------------------------------------------------------------------
-
void __fastcall TfrmMain::Button3Click(TObject *Sender)
{
  TfrmRegion *frm = new TfrmRegion(this);
  frm->ADOQuery1->Connection = ADOConnection1;
  frm->ADOQuery2->Connection = ADOConnection1;
  this->Hide();
  frm->Show();
}
```

Κρύβουμε την κεντρική φόρμα, συνδέουμε τα Query1 & Query2 με το ADOConection1, εμφανίζουμε την φόρμα Αναζήτηση Τοποθεσίας

```
//---------------------------------------------------------------------
-
void __fastcall TfrmMain::Button4Click(TObject *Sender)
{
  TfrmShippers *frm = new TfrmShippers(this);
  frm->ADOQuery1->Connection = ADOConnection1;
  frm->ADOQuery2->Connection = ADOConnection1;
  this->Hide();
  frm->Show();
}
```

Κρύβουμε την κεντρική φόρμα, συνδέουμε τα Query1 & Query2 με το ADOConection1, εμφανίζουμε την φόρμα Αναζήτηση Μεταφορέων

```
//---------------------------------------------------------------------
-
void __fastcall TfrmMain::Button5Click(TObject *Sender)
{
  TfrmOrders *frm = new TfrmOrders(this);
  frm->ADOQuery1->Connection = ADOConnection1;
  this->Hide();
  frm->Show();
}
```

Κρύβουμε την κεντρική φόρμα, συνδέουμε τα Query1 & Query2 με το ADOConection1, εμφανίζουμε την φόρμα Παραγγελίες

```
//---------------------------------------------------------------------
-
void __fastcall TfrmMain::FormCreate(TObject *Sender)
{
   // open  database connections
   if(!Database1->Connected)
     Database1->Connected = true;

     ADOConnection1->Connected =false;
```

```
    ADOConnection1->ConnectionString = "FILE NAME=.\\New.udl";
    ADOConnection1->Connected =true;
}
```

Με το που ανοίγει η κεντρική φόρμα γίνεται η σύνδεση με την βάση, συνδέουμε το ADOConnection1 με το udl αρχείο το οποίο είναι συνδεδεμένο με τη βάση Part1 που βρίσκεται στον SQL Server

```
//---------------------------------------------------------------------
-
void __fastcall TfrmMain::Button8Click(TObject *Sender)
{
  TfrmSalesBySalesmanCube *frm = new TfrmSalesBySalesmanCube(this);
  this->Hide();
  frm->Show();
}
```

Κρύβουμε την κεντρική φόρμα, εμφανίζουμε την φόρμα Πωλήσεις Πωλητών

```
//---------------------------------------------------------------------
-
void __fastcall TfrmMain::Button6Click(TObject *Sender)
{
  TfrmTotalSalesCube *frm = new TfrmTotalSalesCube(this);
  this->Hide();
  frm->Show();
}
```

Κρύβουμε την κεντρική φόρμα, εμφανίζουμε την φόρμα Συγκεντρωτικές Πωλήσεις

```
//---------------------------------------------------------------------
-
void __fastcall TfrmMain::Button9Click(TObject *Sender)
{
  TfrmTotalSums *frm = new TfrmTotalSums(this);
  this->Hide();
  frm->Show();
}
```

Κρύβουμε την κεντρική φόρμα, εμφανίζουμε την φόρμα Αθροίσματα

```
//---------------------------------------------------------------------
-
void __fastcall TfrmMain::Button10Click(TObject *Sender)
{
  TfrmDaysCube *frm = new TfrmDaysCube(this);
  this->Hide();
  frm->Show();
}
```

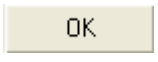
Κρύβουμε την κεντρική φόρμα, εμφανίζουμε την φόρμα Πωλήσεις Προϊόντων ανά περιοχή

```
//---------------------------------------------------------------------
void __fastcall TfrmMain::FormClose(TObject *Sender, TCloseAction
&Action)
{
 // close database connections
   if(Database1->Connected)
     Database1->Connected = false;

   if(ADOConnection1->Connected)
     ADOConnection1->Connected =false;
}
```

Με το κλείσιμο την κεντρικής φόρμας απενεργοποιούμε και την σύνδεση με την βάση

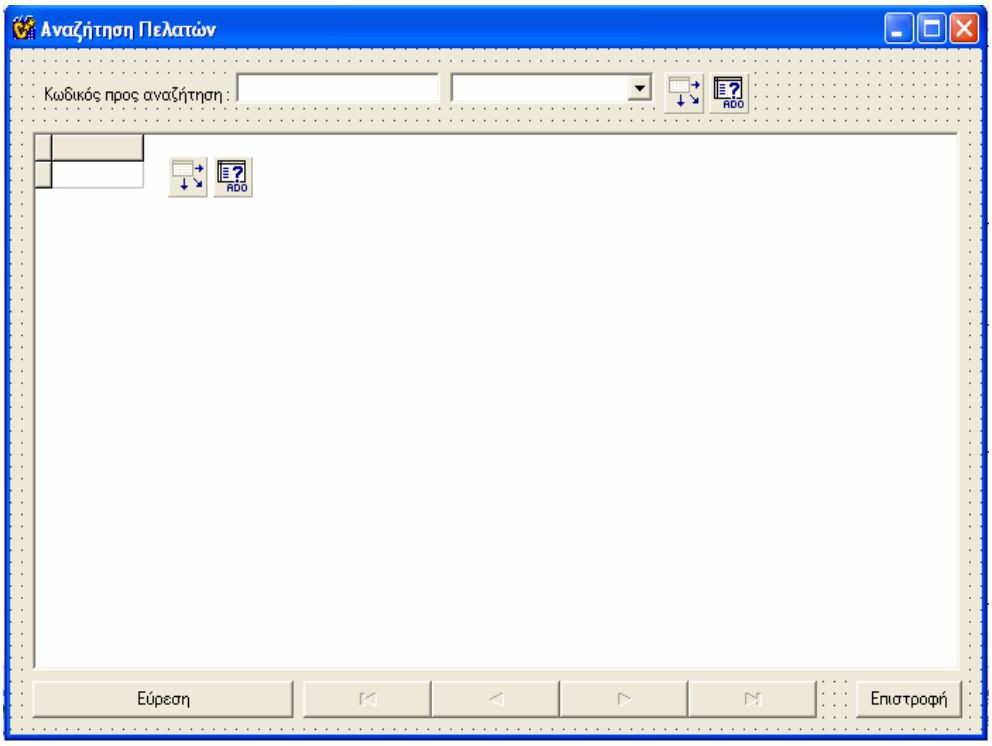
```
//---------------------------------------------------------------------
```

Αντικείμενα	**Όνομα**	**Κλάση**	**Δεδομένα**	**Σχόλια**
	**ADOConnection1**	TADOConnection	Περιέχει το connection string	Συνδέεσαι με το ADOConnection που έχουμε δημιουργήσει, υλοποιεί την σύνδεση με την βάση δεδομένων και διαχειρίζεται τις συναλλαγές
	**Database1**	TDatabase	Περιέχει το όνομα της βάσης δεδομένων	Μπορούμε να έχουμε πρόσβαση μέσα στην βάση δεδομένων
	Button1	TButton	-	Με το κουμπί αυτό μας εμφανίζεται η φόρμα Αναζήτηση Πελατών
	Button2	TButton	-	Με το κουμπί αυτό μας εμφανίζεται η φόρμα Αναζήτηση Αντιπροσώπων
	**Button3**	TButton	-	Με το κουμπί αυτό μας εμφανίζεται η φόρμα Αναζήτηση Εργαζομένων
	Button4	TButton	-	Με το κουμπί αυτό μας εμφανίζεται η φόρμα Αναζήτηση Προϊόντων

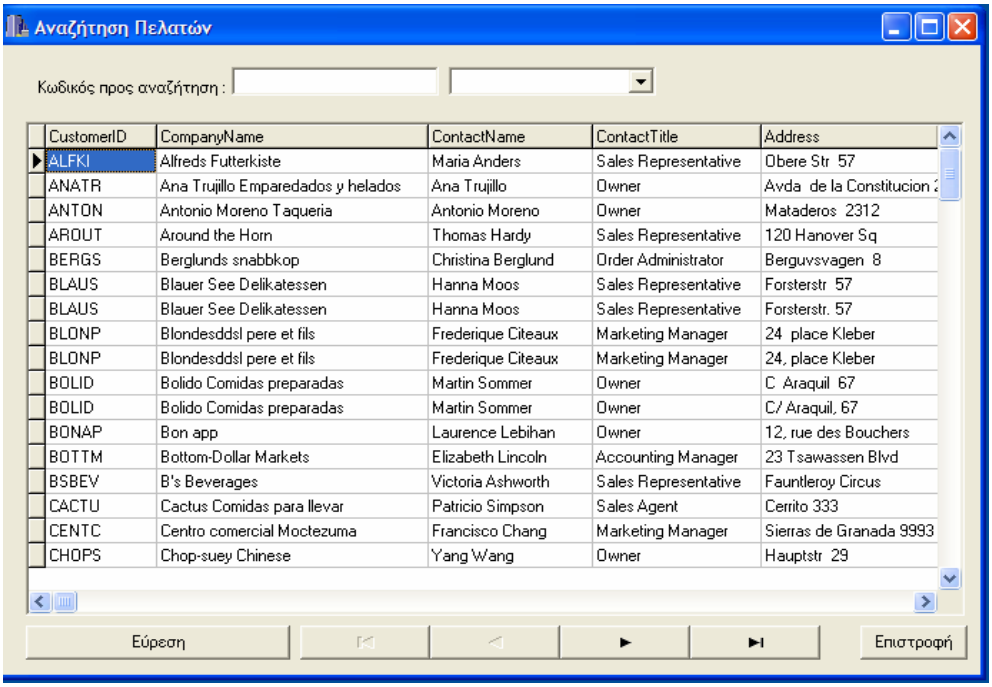
	Button5	TButton	-	Με το κουμπί αυτό μας εμφανίζεται η φόρμα Αναζήτηση Περιοχής
	Button6	TButton	-	Με το κουμπί αυτό μας εμφανίζεται η φόρμα Μεταφορέων
	Button7	TButton	-	Με το κουμπί αυτό μας εμφανίζεται η φόρμα Παραγγελίες
	Button8	TButton	-	Με το κουμπί αυτό μας εμφανίζεται η φόρμα Πωλήσεις Πωλητών
	Button9	TButton	-	Με το κουμπί αυτό μας εμφανίζεται η φόρμα Συγκεντρωτικές Πωλήσεις
	Button10	TButton	-	Με το κουμπί αυτό μας εμφανίζεται η φόρμα Αθροίσματα
	Button11	TButton	-	Με το κουμπί αυτό μας εμφανίζεται η φόρμα Πωλήσεις Προϊόντων
	Button12	TButton	-	Με το κουμπί αυτό βγαίνουμε από το πρόγραμμα

4.1.2 Φόρμα Αναζήτηση Πελατών

Στην φόρμα αυτή κάνουμε αναζήτηση των πελατών που είναι καταχωρημένοι στις βάσεις μας. Για τη δημιουργία της φόρμας χρησιμοποιούμε τα αντικείμενα: **Edit, ComboBox, DataSource, ADOQuery, DBGrid, Button, DBNavigator.** Στο Edit μπορούμε να πληκτρολογήσουμε τον κωδικό του πελάτη και με το Button Εύρεση μας εμφανίζονται τα στοιχεία του πελάτη στο DBGrid. Στο ComboBox είναι καταχωρημένοι όλοι οι κωδικοί πελατών και επιλέγοντας τον κωδικό του πελάτη μας εμφανίζονται όλα στοιχεία του. Το DBGrid είναι συνδεδεμένο με τα DataSource και χρησιμοποιείται για να μας εμφανίζει τα αποτελέσματα της αναζήτησης. Τα ADOQuery χρησιμοποιουνται για να δημιουργήσουμε τα ερωτήματα και να μας εμφανιστούν τα αποτελέσματα στο DBGrid. Τα DataSource τα χρησιμοποιούμε για την διαχείριση των εγγραφών που επιστρέφονται στο ADOQuery με την εκτέλεση του ερωτήματος. Τέλος το αντικείμενο DBNavigator μπορούμε να μετακινιόμαστε πάνω-κάτω μέσα στις εγγραφές.



Εικόνα. Η φόρμα σε χρόνο σχεδίασης (Disign Time)



Εικόνα. Η φόρμα σε χρόνο εκτέλεσης (Run Time)

```
//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "SearchCustForm.h"
#include "MainForm.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmSearchCustomer *frmSearchCustomer;
//---------------------------------------------------------------------
void __fastcall TfrmSearchCustomer::FormClose(TObject *Sender,
      TCloseAction &Action)
{
  ((TForm *)this->Owner)->Show();
}
```

Εμφανίζουμε την κεντρική φόρμα

```
//---------------------------------------------------------------------
void __fastcall TfrmSearchCustomer::Button3Click(TObject *Sender)
{
 Close();
}
```

Κλείνουμε την φόρμα Αναζήτηση Πελατών

```
//---------------------------------------------------------------------
void __fastcall TfrmSearchCustomer::Button1Click(TObject *Sender)
{
  AnsiString c1 = Edit1->Text + "%";
  AnsiString sqlStr = "SELECT ALLCUSTOMERS.* FROM ";
  sqlStr += "( ";
  sqlStr += "  SELECT * FROM CUSTOMERS ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(MYSQL, 'SELECT * FROM
CUSTOMERS') ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(ORACLE, 'SELECT * FROM
CUSTOMERS') ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(PART2, 'SELECT * FROM
CUSTOMERS') ";
  sqlStr += ") ALLCUSTOMERS ";
  sqlStr += "WHERE ALLCUSTOMERS.CUSTOMERID like '" + c1 + "'";
  if(ADOQuery1->Active)
    ADOQuery1->Close();
    ADOQuery1->SQL->Clear();
    ADOQuery1->SQL->Add(sqlStr);
    ADOQuery1->Open();
}
```

Στην μεταβλητή c1 βάζουμε το περιεχόμενο του Edit συν ότι το ακολουθεί, στην μεταβλητή SqlStr βάζουμε το ερώτημα όπου θέλουμε να εμφανίσουμε όλα τα πεδία από τον πίνακα Customers και από τις τέσσερις τοποθεσίες, αφού ενώσουμε τα δεδομένα, εμφανίζουμε αυτά οπού το CustomerID να είναι ίδιο με το περιεχόμενο της μεταβλητής, κλείνουμε το ADOQuery1, το καθαρίζουμε, φορτώνουμε το ερώτημα και ανοίγουμε το ADOQuery1

```
//---------------------------------------------------------------------
void __fastcall TfrmSearchCustomer::FormShow(TObject *Sender)
{
  AnsiString sqlStr = "SELECT ALLCUSTOMERS.* FROM ";
  sqlStr += "( ";
  sqlStr += "  SELECT * FROM CUSTOMERS ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(MYSQL, 'SELECT * FROM
CUSTOMERS') ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(ORACLE, 'SELECT * FROM
CUSTOMERS') ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(PART2, 'SELECT * FROM
CUSTOMERS') ";
  sqlStr += ") ALLCUSTOMERS ";

  if(ADOQuery2->Active)
    ADOQuery2->Close();
    ADOQuery2->SQL->Clear();
    ADOQuery2->SQL->Add(sqlStr);
    ADOQuery2->Open();

  ComboBox1->Items->Clear();
  for(register int i = 0; i < ADOQuery2->RecordCount; i++)
  {
    ComboBox1->Items->Add(ADOQuery2->FieldByName("CustomerID")-
>AsString);
    if (i < ADOQuery2->RecordCount -1 )
      ADOQuery2->MoveBy(1);
  }
  ADOQuery2->Close();
  ComboBox1->Text = "";

  if(ADOQuery1->Active)
    ADOQuery1->Close();
    ADOQuery1->SQL->Clear();
    ADOQuery1->SQL->Add(sqlStr);
    ADOQuery1->Open();
}
```

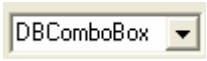
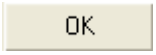
Με το που ανοίγει η φόρμα θα μας εμφανίζονται όλα τα πεδία μαζί με όλες τις εγγραφές τους από τον πίνακα Customers και από τις τέσσερις τοποθεσίες, κλείνουμε το ADOQuery2, το καθαρίζουμε, φορτώνουμε το ερώτημα και το ανοίγουμε, καθαρίζουμε το ComboBox, όσο το

i είναι μικρότερο από τον αριθμό των καταχωρήσεων το αυξάνουμε κατά ένα ώστε να δείχνει στην επόμενη εγγραφή, βάζουμε στο ComboBox το περιεχόμενο του CustomerID σαν χαρακτήρες, αν το i γίνει μικρότερο από την πρώτη εγγραφή τότε το μετακινούμε να δείχνει την πρώτη εγγραφή ώστε να μην μας πετάξει σφάλμα, κλείνουμε το ADOQuery2, κλείνουμε το ADOQuery1, το καθαρίζουμε, του φορτώνουμε το ερώτημα και το ανοίγουμε

```
//---------------------------------------------------------------------
void __fastcall TfrmSearchCustomer::ComboBox1Change(TObject *Sender)
{
  AnsiString c1 = ComboBox1->Text;
  AnsiString sqlStr = "SELECT ALLCUSTOMERS.* FROM ";
  sqlStr += "( ";
  sqlStr += "  SELECT * FROM CUSTOMERS ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(MYSQL, 'SELECT * FROM
CUSTOMERS') ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(ORACLE, 'SELECT * FROM
CUSTOMERS') ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(PART2, 'SELECT * FROM
CUSTOMERS') ";
  sqlStr += ") ALLCUSTOMERS ";
  if (c1 != "")
    sqlStr += "WHERE ALLCUSTOMERS.CUSTOMERID like '" + c1 + "'";
  if(ADOQuery1->Active)
    ADOQuery1->Close();
    ADOQuery1->SQL->Clear();
    ADOQuery1->SQL->Add(sqlStr);
    ADOQuery1->Open();
}
```

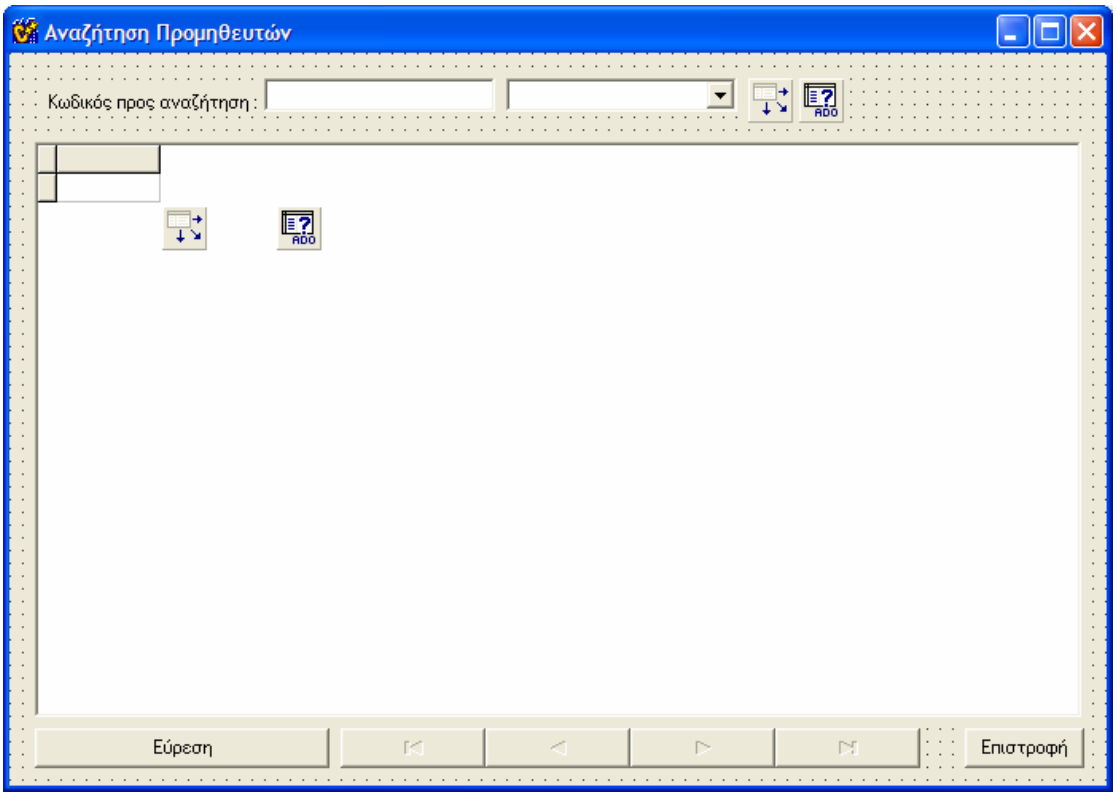
Στην μεταβλητή c1 βάζουμε το περιεχόμενο του ComboBox ,στην μεταβλητή SqlStr βάζουμε το ερώτημα επιλέγοντας όλα τα πεδία από τον πίνακα Customers και από τις τέσσερις τοποθεσίες, θα εμφανίσουμε τα δεδομένα όπου το CustomerID είναι ίδιο με το περιεχόμενο της μεταβλητής c1, κλείνουμε το ADOQuery1, το καθαρίζουμε, του φορτώνουμε το ερώτημα και το ανοίγουμε

```
//---------------------------------------------------------------------
```

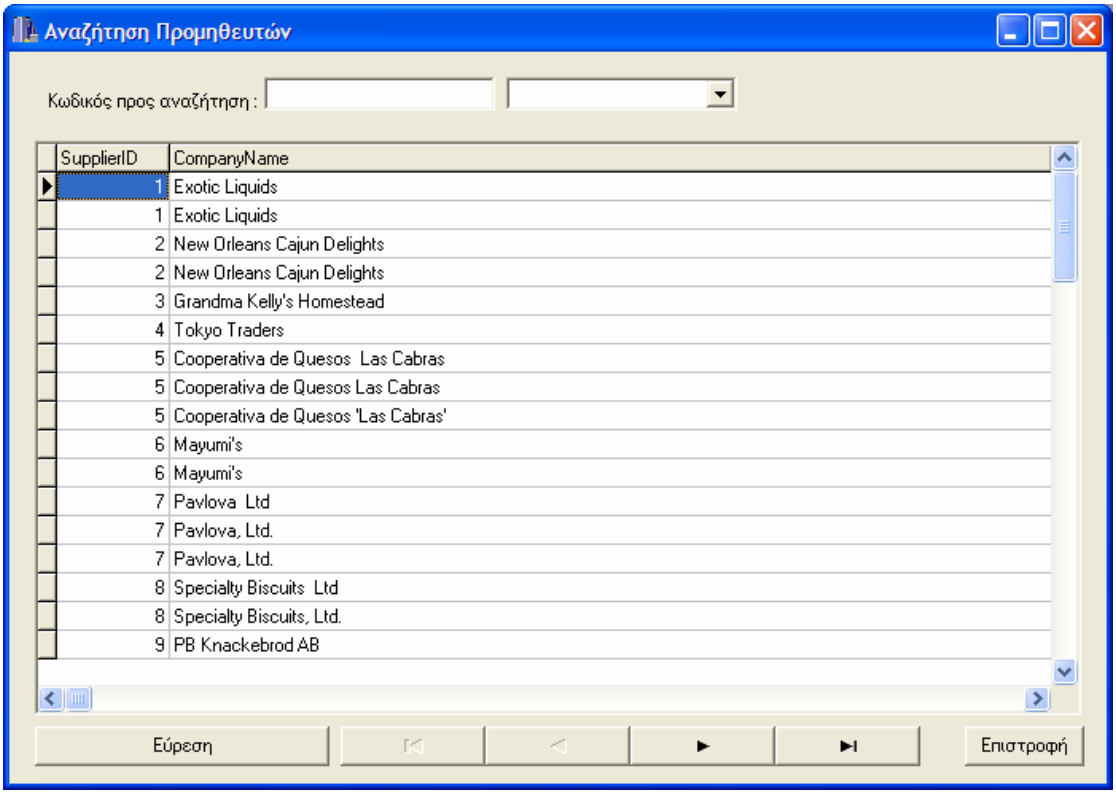
Αντικείμενα	**Όνομα**	**Κλάση**	**Δεδομένα**	**Σχόλια**
	**Edit1**	TEdit	Τα Δεδομένα δίνονται από τον χρήστη	Εισαγωγή κωδικού πελάτη προς αναζήτηση
	**Combo Box1**	TCombo Box	Περιλαμβάνει όλους τους κωδικούς των πελατών	Επιλέγουμε έναν κωδικό πελάτη κάθε φορά μέσα από το σύνολο των κωδικών
	**Data Source1**	TData Source	-	Διαχείριση των εγγραφών που επιστρέφονται από τους πίνακες Customers και από τις τέσσερις βάσεις μέσω του ADOQuery1
	Data Source2	TData Source	-	Διαχείριση των εγγραφών που επιστρέφονται από τους πίνακες Customers και από τις τέσσερις βάσεις μέσω του ADOQuery2
	**ADO Query1**	TADO Query	Περιέχει το αποτέλεσμα που προέκυψε από την εκτέλεση του ερωτήματος	Μέσω του ερωτήματος περνούμε τα Δεδομένα που ζητάμε από τον πίνακα Customers και από τις τέσσερις βάσεις
	ADO Query2	TADO Query		
	**DBGrid1**	TDBGrid	Περιέχει τα αποτελέσματα της αναζήτησης	Εμφανίζει όλα τα στοιχειά του πελάτη που ψάχνουμε
	**Button1**	TButton	-	Με το κουμπί αυτό μας φανίζονται όλες οι εγγραφές που έχουν σχέση με αυτό που γράψαμε στο Edit
	Button3	TButton	-	Με το κουμπί αυτό κλείνουμε την φόρμα
	**DBNavigator**	TDBNavigator	-	Μετακινούμαστε μέσα στις εγγραφές

4.1.3 Φόρμα Αναζήτηση Προμηθευτών

Στην φόρμα αυτή κάνουμε αναζήτηση των προμηθευτών που είναι καταχωρημένοι στις βάσεις μας. Για τη δημιουργία της φόρμας χρησιμοποιούμε τα αντικείμενα: **Edit, ComboBox, DataSource, ADOQuery, DBGrid, Button, DBNavigator.** Στο Edit μπορούμε να πληκτρολογήσουμε τον κωδικό του αντιπροσώπου και με το Button Εύρεση μας εμφανίζονται τα στοιχεία του. Στο ComboBox είναι καταχωρημένοι όλοι οι κωδικοί αντιπροσώπων και επιλέγοντας τον κωδικό μας εμφανίζονται όλα στοιχεία του αντιπροσώπου. Το DBGrid είναι συνδεδεμένο με τα DataSource χρησιμοποιείται για να μας εμφανίζει τα αποτελέσματα της αναζήτησης. Τα ADOQuery χρησιμοποιείται για να δημιουργήσουμε τα ερωτήματα και να μας εμφανιστούν τα αποτελέσματα στο DBGrid. Τα DataSource τα χρησιμοποιούμε για την διαχείριση των εγγραφών που επιστρέφονται στο ADOQuery με την εκτέλεση του ερωτήματος. Τέλος το αντικείμενο DBNavigator μπορούμε να μετακινιόμαστε πάνω-κάτω μέσα στις εγγραφές.



Εικόνα. Η φόρμα σε χρόνο σχεδίασης (Disign Time)



Εικόνα. Η φόρμα σε χρόνο εκτέλεσης (Run Time)

```
//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "SearchSuplForm.h"
#include "MainForm.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmSearchSupplier *frmSearchSupplier;
//---------------------------------------------------------------------
void __fastcall TfrmSearchSupplier::FormClose(TObject *Sender,
      TCloseAction &Action)
{
  ((TForm *)this->Owner)->Show();
}
```

Εμφανίζουμε την φόρμα

```
//---------------------------------------------------------------------
void __fastcall TfrmSearchSupplier::Button3Click(TObject *Sender)
{
  Close();
}
```

Κλείνουμε την φόρμα Αναζήτηση Προμηθευτών

```
//---------------------------------------------------------------------
void __fastcall TfrmSearchSupplier::Button1Click(TObject *Sender)
{
  AnsiString c1 = Edit1->Text + "%";
  AnsiString sqlStr = "SELECT ALLSUPPLIERS.* FROM ";
  sqlStr += "( ";
  sqlStr += "  SELECT * FROM SUPPLIERS ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(MYSQL, 'SELECT * FROM
SUPPLIERS') ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(PART2, 'SELECT * FROM
SUPPLIERS') ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(ORACLE, 'SELECT * FROM
SUPPLIERS') ";
  sqlStr += ") ALLSUPPLIERS ";
  sqlStr += "WHERE SupplierID like '" + c1 + "'";
  if(ADOQuery1->Active)
    ADOQuery1->Close();
    ADOQuery1->SQL->Clear();
    ADOQuery1->SQL->Add(sqlStr);
    ADOQuery1->Open();
}
```

Στην μεταβλητή c1 τύπου AnsiString βάζουμε το περιεχόμενο του Edit συν ότι το ακολουθεί, στην μεταβλητή SqlStr βάζουμε το ερώτημα οπού θέλουμε να εμφανίσουμε όλα τα πεδία από τον πίνακα Suppliers και από τις τέσσερις τοποθεσίες, αφού ενώσουμε τα δεδομένα εμφανιζουμε αυτά οπού το SupplierID είναι ίδιο με το περιεχόμενο της μεταβλητής c1. Κλείνουμε το ADOQuery1, το καθαρίζουμε, του φορτώνουμε το ερώτημα και το ανοίγουμε

```
//---------------------------------------------------------------------
void __fastcall TfrmSearchSupplier::FormShow(TObject *Sender)
{
  AnsiString sqlStr = "SELECT ALLSUPPLIERS.* FROM ";
  sqlStr += "( ";
  sqlStr += "  SELECT * FROM SUPPLIERS ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(MYSQL, 'SELECT * FROM
SUPPLIERS') ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(PART2, 'SELECT * FROM
SUPPLIERS') ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(ORACLE, 'SELECT * FROM
SUPPLIERS') ";
  sqlStr += ") ALLSUPPLIERS ";

  if(ADOQuery2->Active)
    ADOQuery2->Close();
    ADOQuery2->SQL->Clear();
```

```
    ADOQuery2->SQL->Add(sqlStr);
    ADOQuery2->Open();
    ComboBox1->Items->Clear();
    ComboBox1->Text = "";
    ADOQuery2->First();
  for(register int i = 0; i < ADOQuery2->RecordCount; i++)
  {
    ComboBox1->Items->Add(ADOQuery2->FieldByName("SupplierID")-
>AsInteger);
    if (i < ADOQuery2->RecordCount -1 )
      ADOQuery2->MoveBy(1);
  }
  ADOQuery2->Close();
    if(ADOQuery1->Active)
      ADOQuery1->Close();
      ADOQuery1->SQL->Clear();
      ADOQuery1->SQL->Add(sqlStr);
      ADOQuery1->Open();
}
```

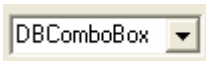
Με το που εμφανίζεται η φόρμα εμφανίζονται όλα τα πεδία μαζί με όλες τις εγγραφές τους από τον πίνακα Suppliers και από τις τέσσερις τοποθεσίες, κλείνουμε το ADOQuery2, το καθαρίζουμε, του φορτώνουμε το ερώτημα και το ανοίγουμε, καθαρίζουμε το ComboBox, όσο το i είναι μικρότερο από τον αριθμό των καταχωρήσεων το αυξάνουμε κατά ένα ώστε να δείχνει στην επόμενη εγγραφή, βάζουμε στο ComboBox το περιεχόμενο του SupplierID σαν αριθμό, αν το i γίνει μικρότερο από την πρώτη εγγραφή τότε το μετακινούμε να δείχνει την πρώτη εγγραφή ώστε να μην μας πετάξει σφάλμα, κλείνουμε το ADOQuery2, κλείνουμε το ADOQuery1, το καθαρίζουμε, του φορτώνουμε το ερώτημα και το ανοίγουμε

```
//---------------------------------------------------------------------
-
void __fastcall TfrmSearchSupplier::ComboBox1Change(TObject *Sender)
{
  AnsiString c1 = ComboBox1->Text;
  AnsiString sqlStr = "SELECT ALLSUPPLIERS.* FROM ";
  sqlStr += "( ";
  sqlStr += "  SELECT * FROM SUPPLIERS ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(MYSQL, 'SELECT * FROM
SUPPLIERS') ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(PART2, 'SELECT * FROM
SUPPLIERS') ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(ORACLE, 'SELECT * FROM
SUPPLIERS') ";
  sqlStr += ") ALLSUPPLIERS ";
  if (c1 != "")
    sqlStr += "WHERE SUPPLIERID = '" + c1 + "'";
  if(ADOQuery1->Active)
```

```
    ADOQuery1->Close();
    ADOQuery1->SQL->Clear();
    ADOQuery1->SQL->Add(sqlStr);
    ADOQuery1->Open();
}
```

Δημιουργούμε δυο μεταβλητές c1 και sqlStr τύπου AnsiString, στην μεταβλητή c1 βάζουμε το περιεχόμενο του ComboBox σαν κείμενο , στην μεταβλητή SqlStr βάζουμε το ερώτημα επιλέγοντας όλα τα πεδία από τον πίνακα Suppliers και από τις τέσσερις τοποθεσίες, θα εμφανίσουμε τα δεδομένα όπου το SuppliersID είναι ίδιο με το περιεχόμενο της μεταβλητής c1, κλείνουμε το ADOQuery1, το καθαρίζουμε, του φορτώνουμε το ερώτημα και το ανοίγουμε

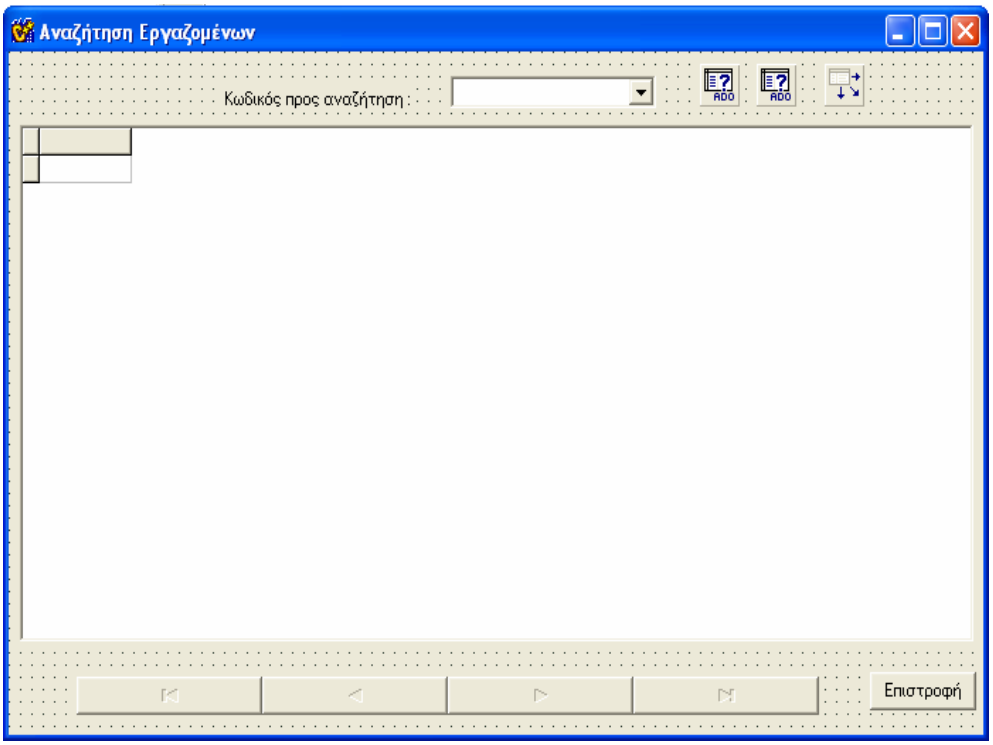
```
//---------------------------------------------------------------------------
```

Αντικείμενα	**Όνομα**	**Κλάση**	**Δεδομένα**	**Σχόλια**
	**Edit1**	TEdit	Τα δεδομένα δίνονται από τον χρηστή	Εισαγωγή κωδικού αντιπροσώπου προς αναζήτηση
	**Combo Box1**	TComboBox	Περιλαμβάνει όλους τους κωδικούς των αντιπροσώπων	Επιλέγουμε έναν κωδικό αντιπροσώπου κάθε φορά μέσα από το σύνολο των κωδικών
	**Data Source1**	TDataSource	-	Διαχείριση των εγγραφών που επιστρέφονται από τους πίνακες Supplier και από τις τέσσερις βάσεις μέσω του ADOQuery1
	Data Source2	TDataSource	-	Διαχείριση των εγγραφών που επιστρέφονται από τους πίνακες Supplier και από τις τέσσερις βάσεις μέσω του ADOQuery2
	**ADO Query1**	TADOQuery	Περιέχει το αποτέλεσμα που προέκυψε από την εκτέλεση του ερωτήματος	Μέσω του ερωτήματος παίρνουμε τα δεδομένα που ζητάμε από τον πίνακα Supplier και από τις τέσσερις βάσεις
	ADO Query2	TADOQuery		
	**DBGrid1**	TDBGrid	Περιέχει τα αποτελέσματα της αναζήτησης	Εμφανίζει όλα τα στοιχεία του αντιπροσώπου που ψάχνουμε
	Button1	TButton	-	Με το κουμπί αυτό μας

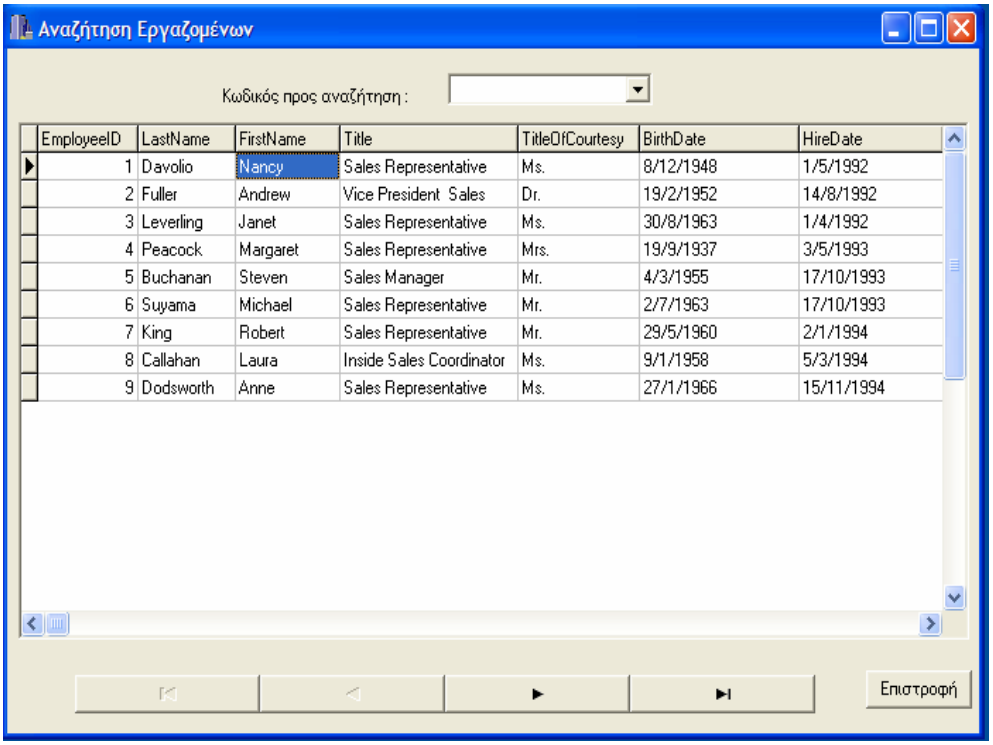
OK				φανίζονται όλες οι εγγραφές που έχουν σχέση με αυτό που γράψαμε στο Edit
	Button3	TButton	-	Με το κουμπί αυτό κλείνουμε την φόρμα
	**DBNavigator**	TDBNavigator	-	Μετακινούμαστε μέσα στις εγγραφές

4.1.4 Φόρμα Αναζήτηση Εργαζομένων

Στην φόρμα αυτή κάνουμε αναζήτηση των εργαζομένων που είναι καταχωρημένοι στις βάσεις μας. Για τη δημιουργία της φόρμας χρησιμοποιούμε τα αντικείμενα: **ComboBox, DataSource, ADOQuery, DBGrid, Button, DBNavigator.** Στο ComboBox είναι καταχωρημένοι όλοι οι κωδικοί και επιλέγοντας τον κωδικό μας εμφανίζονται όλα στοιχεία του εργαζομένου. Το DBGrid είναι συνδεδεμένο με το DataSource χρησιμοποιείται για να μας εμφανίζει τα αποτελέσματα της αναζήτησης. Τα ADOQuery χρησιμοποιούνται για να δημιουργήσουμε τα ερωτήματα και να μας εμφανιστούν τα αποτελέσματα στο DBGrid. Το DataSource το χρησιμοποιούμε για την διαχείριση των εγγραφών που επιστρέφονται στο ADOQuery με την εκτέλεση του ερωτήματος. Το αντικείμενο DBNavigator μπορούμε να μετακινιόμαστε πάνω-κάτω μέσα στις εγγραφές. Τέλος με το Button Επιστροφή μεταφερόμαστε στην κεντρική φόρμα



Εικόνα. Η φόρμα σε χρόνο σχεδίασης (Disign Time)



EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate
1	Davolio	Nancy	Sales Representative	Ms.	8/12/1948	1/5/1992
2	Fuller	Andrew	Vice President Sales	Dr.	19/2/1952	14/8/1992
3	Leverling	Janet	Sales Representative	Ms.	30/8/1963	1/4/1992
4	Peacock	Margaret	Sales Representative	Mrs.	19/9/1937	3/5/1993
5	Buchanan	Steven	Sales Manager	Mr.	4/3/1955	17/10/1993
6	Suyama	Michael	Sales Representative	Mr.	2/7/1963	17/10/1993
7	King	Robert	Sales Representative	Mr.	29/5/1960	2/1/1994
8	Callahan	Laura	Inside Sales Coordinator	Ms.	9/1/1958	5/3/1994
9	Dodsworth	Anne	Sales Representative	Ms.	27/1/1966	15/11/1994

Εικόνα. Η φόρμα σε χρόνο εκτέλεσης (Run Time)

```
//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "SearchEmpForm.h"
#include "MainForm.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmSearchEmp *frmSearchEmp;
//---------------------------------------------------------------------
void __fastcall TfrmSearchEmp::FormClose(TObject *Sender,
      TCloseAction &Action)
{
((TForm *)this->Owner)->Show();
}
```

Εμφανίζουμε την κεντρική φόρμα

```
//---------------------------------------------------------------------
void __fastcall TfrmSearchEmp::Button3Click(TObject *Sender)
{
Close();
}
```

Κλείνουμε την φόρμα Αναζήτηση Εργαζομένων

```
//---------------------------------------------------------------------
void __fastcall TfrmSearchEmp::FormShow(TObject *Sender)
```

```
{
  AnsiString sqlStr = "";
  sqlStr += "SELECT ALLEMPLOYEES.* FROM ";
  sqlStr += "( ";
  sqlStr += "  SELECT * FROM EMPLOYEES ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(MYSQL, 'SELECT * FROM
EMPLOYEES') ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(PART2, 'SELECT * FROM
EMPLOYEES') ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(ORACLE, 'SELECT * FROM
EMPLOYEES') ";
  sqlStr += ") ALLEMPLOYEES ";
  sqlStr += "order by EmployeeID ";

  if(ADOQuery2->Active)
    ADOQuery2->Close();
    ADOQuery2->SQL->Clear();
    ADOQuery2->SQL->Add(sqlStr);
    ADOQuery2->Open();

  ComboBox1->Items->Clear();
  ComboBox1->Text = "";
  ADOQuery2->First();
  for(register int i = 0; i < ADOQuery2->RecordCount; i++)
  {
    ComboBox1->Items->Add(ADOQuery2->FieldByName("EmployeeID")-
>AsInteger);
    if (i < ADOQuery2->RecordCount -1 )
      ADOQuery2->MoveBy(1);
  }
  ADOQuery2->Close();

  if(ADOQuery1->Active)
    ADOQuery1->Close();
    ADOQuery1->SQL->Clear();
    ADOQuery1->SQL->Add(sqlStr);
    ADOQuery1->Open();
}
```

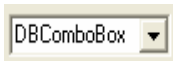
Δημιουργούμε μια μεταβλητή sqlStr τύπου AnsiString όπου εκεί θα βάλουμε το ερώτημα, επιλέγουμε όλα τα πεδία από τον πίνακα Employees και από τις τέσσερις τοποθεσίες και αφού ενώσουμε τα δεδομένα τα ταξινομούμε ως προς το EmployeeID, κλείνουμε το ADOQuery2, το καθαρίζουμε, του φορτώνουμε το ερώτημα και το ανοίγουμε, καθαρίζουμε το ComboBox και εκτελούμε πρώτα το ADOQuery2, οσο το i είναι μικρότερο από το αριθμό τον εγγραφών το μετακινούμε κατά μια θέση ώστε μα δείχνει την επόμενη εγγραφή, καταχωρούμε στο ComboBox το περιεχόμενο του πεδίου EmployeeID σαν ακέραιο αριθμό, αν το i γίνει μικρότερο από την πρώτη εγγραφή τότε το μετακινούμε να δείχνει την πρώτη

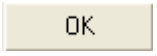
εγγραφή ώστε να μην μας πετάξει σφάλμα, κλείνουμε το ADOQuery2, Αν το ADOQuery1 είναι ενεργό το κλείνουμε, το καθαρίζουμε του φορτώνουμε το ερώτημα και το ανοίγουμε

```
//---------------------------------------------------------------------
void __fastcall TfrmSearchEmp::ComboBox1Change(TObject *Sender)
{
AnsiString c1 = ComboBox1->Text;
  AnsiString sqlStr = "";
  sqlStr += "SELECT ALLEMPLOYEES.* FROM ";
  sqlStr += "( ";
  sqlStr += "  SELECT * FROM EMPLOYEES ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(MYSQL, 'SELECT * FROM
EMPLOYEES') ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(PART2, 'SELECT * FROM
EMPLOYEES') ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(ORACLE, 'SELECT * FROM
EMPLOYEES') ";
  sqlStr += ") ALLEMPLOYEES ";
  if (c1 != "")
    sqlStr += "WHERE EmployeeID = '" + c1 + "'";
    sqlStr += "order by EmployeeID ";
  if(ADOQuery1->Active)
    ADOQuery1->Close();
    ADOQuery1->SQL->Clear();
    ADOQuery1->SQL->Add(sqlStr);
    ADOQuery1->Open();
}
```

Βάζουμε το κείμενο του ComboBox σε μια μεταβλητή c1 τύπου AnsiString, δημιουργούμε άλλη μια μεταβλητή sqlStr τύπου AnsiString για να βάλουμε το ερώτημα, στο ερώτημα επιλέγουμε όλα τα πεδία από τον πίνακα Employees και από τις τέσσερις βάσεις και αφού τα ενώσουμε μας εμφανίζει τα δεδομένα όπου το EmployeeID είναι ίδιο με το περιεχόμενο της μεταβλητής c1 και τα ταξινομούμε ως προς το EmployeeID, κλείνουμε το ADOQuery1 αν είναι ενεργό, το καθαρίζουμε, του φορτώνουμε το ερώτημα και το ανοίγουμε

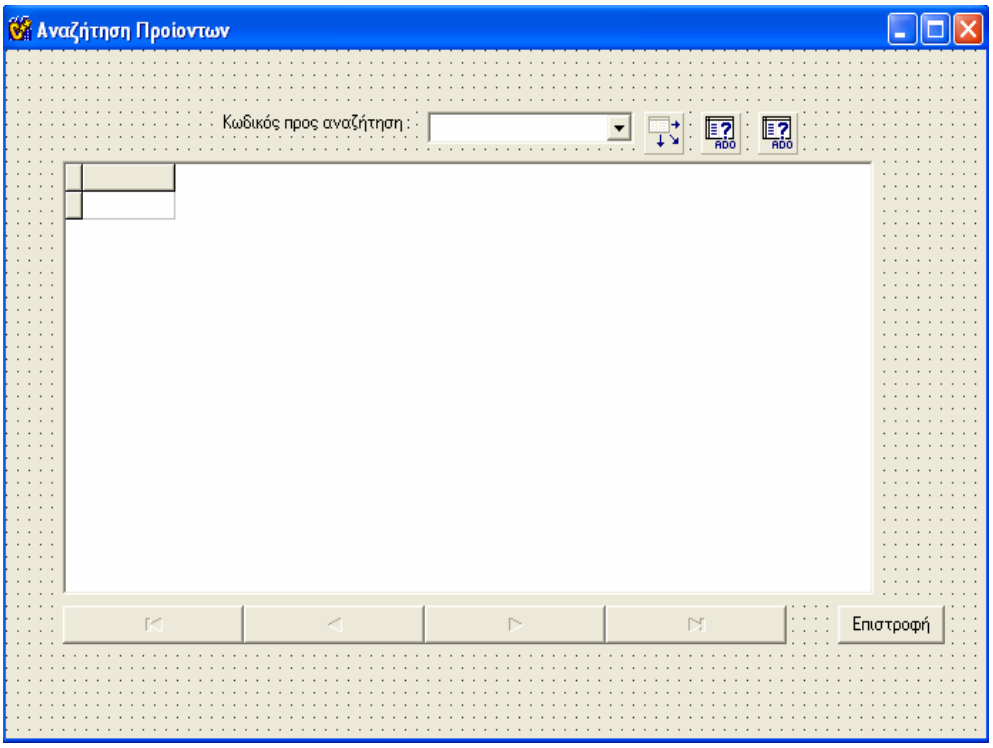
```
//---------------------------------------------------------------------
-
```

Αντικείμενα	**Όνομα**	**Κλάση**	**Δεδομένα**	**Σχόλια**
	**ComboBox1**	TComboBox	Περιλαμβάνει όλους τους κωδικούς των εργαζομένων	Επιλέγουμε έναν κωδικό εργαζομένου κάθε φορά μέσα από το σύνολο των κωδικών
	DataSource1	TDataSource	-	Διαχείριση των εγγραφών που

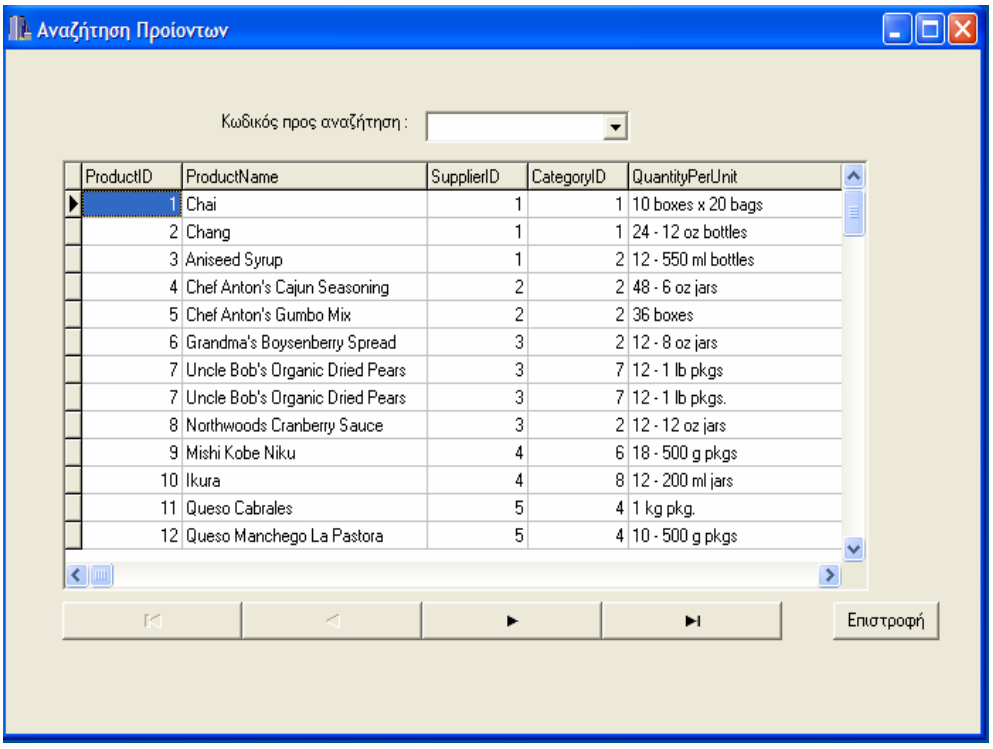
				επιστρέφονται από τους πίνακες Employees και από τις τέσσερις βάσεις μέσω του ADOQuery1
	**ADOQuery1**	TADOQuery	Περιέχει το αποτέλεσμα που προέκυψε από την εκτέλεση του ερωτήματος	Μέσω του ερωτήματος παίρνουμε τα δεδομένα που ζητάμε από τον πίνακα Employees και από τις τέσσερις βάσεις
	ADOQuery2	TADOQuery		
	**DBGrid1**	TDBGrid	Περιέχει τα αποτελέσματα της αναζήτησης	Εμφανίζει όλα τα στοιχεία του εργαζομένου που ψάχνουμε
	**Button3**	TButton	-	Με το κουμπί αυτό κλείνουμε την φόρμα
	**DBNavigator**	TDBNavigator	-	Μετακινούμαστε μέσα στις εγγραφές

4.1.5 Φόρμα Αναζήτηση Προϊόντων

Στην φόρμα αυτή κάνουμε αναζήτηση των προϊόντων που είναι καταχωρημένες στις βάσεις μας. Για τη δημιουργία της φόρμας χρησιμοποιούμε τα αντικείμενα**: ComboBox, DataSource, ADOQuery, DBGrid, Button, DBNavigator.** Στο ComboBox είναι καταχωρημένοι όλοι οι κωδικοί και επιλέγοντας τον κωδικό μας εμφανίζονται όλα στοιχεία του προϊόντος. Το DBGrid είναι συνδεδεμένο με το DataSource χρησιμοποιείται για να μας εμφανίζει τα αποτελέσματα της αναζήτησης. Τα ADOQuery χρησιμοποιούνται για να δημιουργήσουμε τα ερωτήματα και να μας εμφανιστούν τα αποτελέσματα στο DBGrid. Το DataSource το χρησιμοποιούμε για την διαχείριση των εγγραφών που επιστρέφονται στο ADOQuery με την εκτέλεση του ερωτήματος. Το αντικείμενο DBNavigator μπορούμε να μετακινιόμαστε πάνω-κάτω μέσα στις εγγραφές. Τέλος με το Button Επιστροφή μεταφερόμαστε στην κεντρική φόρμα



Εικόνα. Η φόρμα σε χρόνο σχεδίασης (Disign Time)



Εικόνα. Η φόρμα σε χρόνο εκτέλεσης (Run Time)

```
//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "ProductsForm.h"
#include "MainForm.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmProducts *frmProducts;
//---------------------------------------------------------------------
void __fastcall TfrmProducts::Button1Click(TObject *Sender)
{
  if(ADOQuery1->Active)
    ADOQuery1->Close();
  Close();
}
```

Κλείνουμε το ADOQuery1 και την φόρμα

```
//---------------------------------------------------------------------
void __fastcall TfrmProducts::FormClose(TObject *Sender,
      TCloseAction &Action)
{
((TForm *)this->Owner)->Show();
}
```

Εμφανιζουμε την κεντρική φόρμα

```
//---------------------------------------------------------------------
void __fastcall TfrmProducts::ComboBox1Change(TObject *Sender)
{
  AnsiString c1 = ComboBox1->Text;
  AnsiString sqlStr = "";
  sqlStr += "SELECT ALLPRODUCT.* FROM ";
  sqlStr += "( ";
  sqlStr += "  SELECT * FROM PRODUCT ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(MYSQL, 'SELECT * FROM
PRODUCTS') ";
```

Όνομα Linked Server για την τοποθεσία 3, επιλέγουμε όλα τα πεδία από τον πίνακα Products

```
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(PART2, 'SELECT * FROM
PRODUCT') ";
```

Όνομα Linked Server για την τοποθεσία 1, επιλέγουμε όλα τα πεδία από τον πίνακα Products

```
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(ORACLE, 'SELECT * FROM
PRODUCTS') ";
```

Όνομα Linked Server για την τοποθεσία 4, επιλέγουμε όλα τα πεδία από τον πίνακα Products

```
  sqlStr += ") ALLPRODUCT ";
  if (c1 != "")
    sqlStr += "WHERE ProductID = '" + c1 + "'";
    sqlStr += "order by ProductID ";
  if(ADOQuery1->Active)
    ADOQuery1->Close();
    ADOQuery1->SQL->Clear();
    ADOQuery1->SQL->Add(sqlStr);
    ADOQuery1->Open();
}
```

Το περιεχόμενο του ComboBox το βάζουμε σε μια μεταβλητής1 τύπου AnsiString και εκτελούμε το ερώτημα επιλέγουμε όλα τα πεδία από τον πίνακα Product και από τις τέσσερις τοποθεσίες τα ενώνουμε, αν η μεταβλητή c1 που έχει το περιεχόμενο του ComboBox δεν είναι άδεια την προσθέτουμε στο ερώτημα και μας εμφανίζει τις εγγραφές, ταξινομημένες ως προς το ProductID, κλείνουμε το ερώτημα, καθαρίζουμε το ADOQuery1, φορτώνουμε το ερώτημα στο ADOQuery1 και το ανοίγουμε

```
//---------------------------------------------------------------------
void __fastcall TfrmProducts::FormShow(TObject *Sender)
{
  AnsiString sqlStr = "";
  sqlStr += "SELECT ALLPRODUCT.* FROM ";
  sqlStr += "( ";
  sqlStr += "  SELECT * FROM PRODUCT ";
  sqlStr += "  UNION ";
  sqlStr += "  SELECT * FROM OPENQUERY(MYSQL, 'SELECT * FROM
PRODUCTS') ";
```

Όνομα Linked Server για την τοποθεσία 3, επιλέγουμε όλα τα πεδία από τον πίνακα Products

```
  sqlStr += "  UNION ";
  sqlStr += "  SELECT * FROM OPENQUERY(PART2, 'SELECT * FROM
PRODUCT') ";
```

Όνομα Linked Server για τον την τοποθεσία 1, επιλέγουμε όλα τα πεδία από τον πίνακα Products

```
  sqlStr += "  UNION ";
  sqlStr += "  SELECT * FROM OPENQUERY(ORACLE, 'SELECT * FROM
PRODUCTS') ";
```

Όνομα Linked Server για την τοποθεσία 4, επιλέγουμε όλα τα πεδία από τον πίνακα Products

```
  sqlStr += ") ALLPRODUCT ";
  sqlStr += "order by ProductID ";
```

Με το που ανοίγει η φόρμα εκτελούμε το ερώτημα, επιλέγουμε όλα τα πεδία από τον πίνακα Product και από τις τέσσερις τοποθεσίες, τα ενώνουμε, και ταξινομούμε τα αποτελέσματα ως προς το ProductID

```
    if(ADOQuery2->Active)
      ADOQuery2->Close();
```

```
      ADOQuery2->SQL->Clear();
      ADOQuery2->SQL->Add(sqlStr);
      ADOQuery2->Open();
```

Κλείνουμε το ADOQuery2, το καθαρίζουμε, του φορτώνουμε το ερώτημα και το ανοίγουμε

```
  ComboBox1->Items->Clear();
  ComboBox1->Text = "";
  ADOQuery2->First();
  for(register int i = 0; i < ADOQuery2->RecordCount; i++)
  {
    ComboBox1->Items->Add(ADOQuery2->FieldByName("ProductID")-
>AsInteger);
    if (i < ADOQuery2->RecordCount -1 )
      ADOQuery2->MoveBy(1);
  }
  ADOQuery2->Close();
```

Καθαρίζουμε το ComboBox ώστε να μην δείχνει σε καμια εγγραφή, εκτελείται πρώτα το ADOQuery2. Όσο το i είναι μικρότερο από το πλήθος των εγγραφών αύξησε το κατά ένα πέρνα στο ComboBox τα περιεχόμενα του πεδίου ProductId σαν ακέραιους αριθμούς , αν ο μετρητής είναι μηδέν το πηγαίνουμε να δείχνει την πρώτη εγγραφή κλείνουμε το ADOQuery2

```
  sqlStr = "";
  sqlStr += "SELECT ALLPRODUCT.* FROM ";
  sqlStr += "( ";
  sqlStr += "  SELECT * FROM PRODUCT ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(MYSQL, 'SELECT * FROM
PRODUCTS') ";
```

Όνομα Linked Server για την MySQL,επιλέγουμε όλα τα πεδία από τον πίνακα Products

```
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(PART2, 'SELECT * FROM
PRODUCT') ";
```

Όνομα Linked Server για τ,επιλέγουμε όλα τα πεδία από τον πίνακα Products

```
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(ORACLE, 'SELECT * FROM
PRODUCTS') ";
```

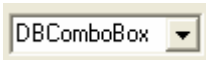
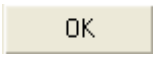
Όνομα Linked Server για την Oracle, επιλέγουμε όλα τα πεδία από τον πίνακα Products

```
  sqlStr += ") ALLPRODUCT ";
  sqlStr += "order by ProductID ";
  if(ADOQuery1->Active)
    ADOQuery1->Close();
    ADOQuery1->SQL->Clear();
    ADOQuery1->SQL->Add(sqlStr);
    ADOQuery1->Open();
```

```
}
```

Στην αρχή καθαρίζουμε το AnsiString sqlStr, επιλέγουμε όλα τα πεδία από τον πίνακα Product και από τις τέσσερις τοποθεσίες, τα ενώνουμε , ταξινομούμε τα αποτελέσματα ως προς το ProductID, κλείνουμε το ADOQuery1 αν είναι ανοιχτό, το καθαρίζουμε, του φορτώνουμε το ερώτημα και το ανοίγουμε

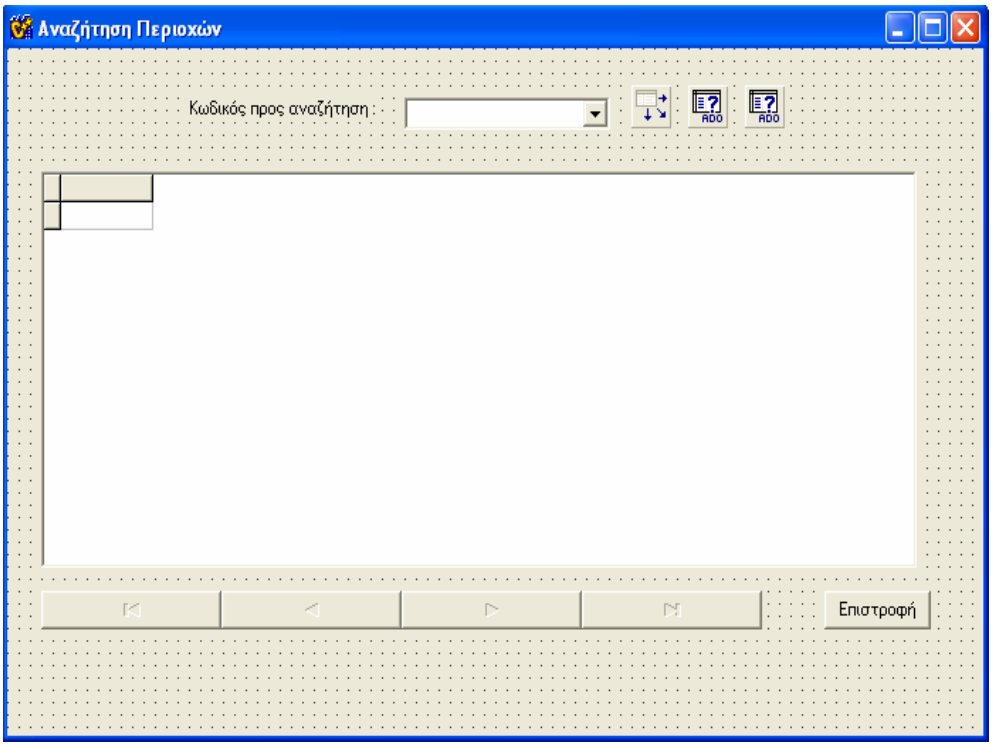
```
//---------------------------------------------------------------------------
```

Αντικείμενα	**Όνομα**	**Κλάση**	**Δεδομένα**	**Σχόλια**
	**ComboBox1**	TComboBox	Περιλαμβάνει όλους τους κωδικούς των προϊόντων	Επιλέγουμε έναν κωδικό προϊόντος κάθε φορά μέσα από το σύνολο των κωδικών
	**DataSource1**	TDataSource	-	Διαχείριση των εγγραφών που επιστρέφονται από τους πίνακες Products και από τις τέσσερις βάσεις μέσω του ADOQuery1
	**ADOQuery1**	TADOQuery	Περιέχει το αποτέλεσμα που προέκυψε από την εκτέλεση του ερωτήματος	Μέσω του ερωτήματος περνούμε τα δεδομένα που ζητάμε από τον πίνακα Products και από τις τέσσερις βάσεις
	ADOQuery2	TADOQuery		
	**DBGrid1**	TDBGrid	Περιέχει τα αποτελέσματα της αναζήτησης	Εμφανίζει όλα τα στοιχεία του προϊόντος που ψάχνουμε
	**Button3**	TButton	-	Με το κουμπί αυτό κλείνουμε την φόρμα
	**DBNavigator**	TDBNavigator	-	Μετακινούμαστε μέσα στις εγγραφές

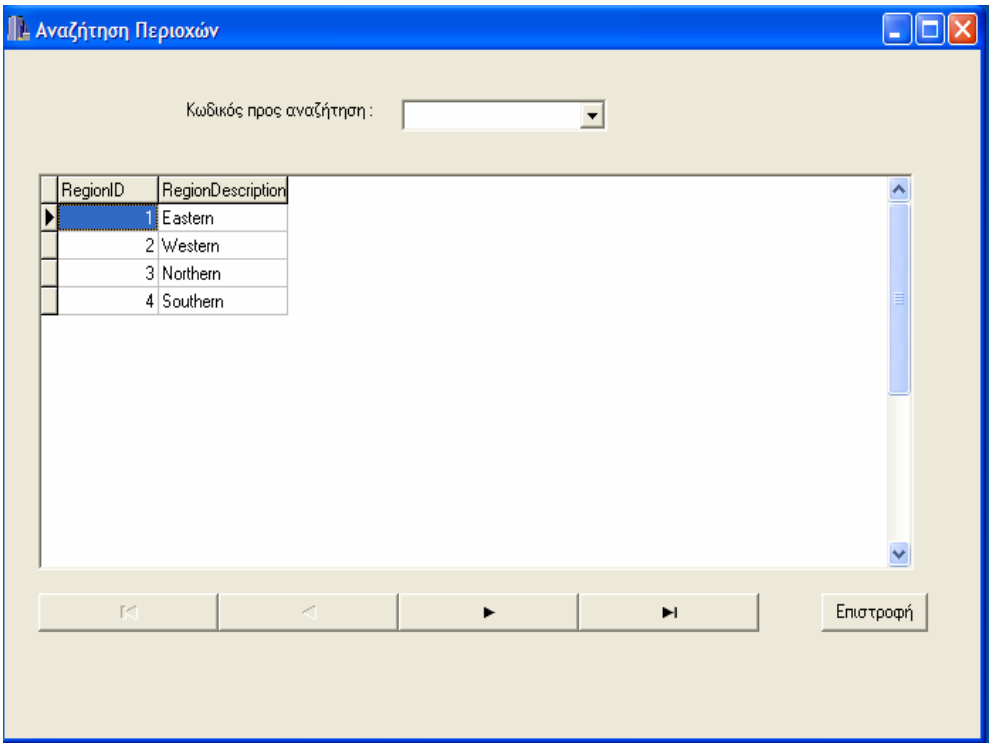
4.1.6 Φόρμα Αναζήτηση Περιοχών

Στην φόρμα αυτή κάνουμε αναζήτηση των περιοχών που είναι καταχωρημένες στις βάσεις μας. Για τη δημιουργία της φόρμας χρησιμοποιούμε τα αντικείμενα: **ComboBox, DataSource, ADOQuery, DBGrid, Button, DBNavigator.** Στο ComboBox είναι καταχωρημένοι όλοι οι κωδικοί και επιλέγοντας τον κωδικό μας εμφανίζονται η αντίστοιχη περιοχή. Το DBGrid είναι συνδεδεμένο με το DataSource χρησιμοποιείται για να μας

εμφανίζει τα αποτελέσματα της αναζήτησης. Τα ADOQuery χρησιμοποιούνται για να δημιουργήσουμε τα ερωτήματα και να μας εμφανιστούν τα αποτελέσματα στο DBGrid. Το DataSource το χρησιμοποιούμε για την διαχείριση των εγγραφών που επιστρέφονται στο ADOQuery με την εκτέλεση του ερωτήματος. Το αντικείμενο DBNavigator μπορούμε να μετακινιόμαστε πάνω-κάτω μέσα στις εγγραφές. Τέλος με το Button Επιστροφή μεταφερόμαστε στην κεντρική φόρμα



Εικόνα. Η φόρμα σε χρόνο σχεδίασης (Disign Time)



Εικόνα. Η φόρμα σε χρόνο εκτέλεσης (Run Time)

```
//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "RegionForm.h"
#include "MainForm.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmRegion *frmRegion;
//---------------------------------------------------------------------
void __fastcall TfrmRegion::Button1Click(TObject *Sender)
{
Close();
}
```

Κλείνουμε την φόρμα Αναζήτηση Περιοχής

```
//---------------------------------------------------------------------
void __fastcall TfrmRegion::FormShow(TObject *Sender)
{
  AnsiString sqlStr = "";
  sqlStr += "SELECT ALLREGION.* FROM ";
  sqlStr += "( ";
  sqlStr += "  SELECT * FROM REGION ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(MYSQL, 'SELECT * FROM REGION')
";
```

```
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(PART2, 'SELECT * FROM REGION')
";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(ORACLE, 'SELECT * FROM
REGION') ";
  sqlStr += ") ALLREGION ";
  sqlStr += "order by RegionID ";

  if(ADOQuery2->Active)
    ADOQuery2->Close();
    ADOQuery2->SQL->Clear();
    ADOQuery2->SQL->Add(sqlStr);
    ADOQuery2->Open();

  ComboBox1->Items->Clear();
  ComboBox1->Text = "";
  ADOQuery2->First();
  for(register int i = 0; i < ADOQuery2->RecordCount; i++)
  {
    ComboBox1->Items->Add(ADOQuery2->FieldByName("RegionID")-
>AsInteger);
    if (i < ADOQuery2->RecordCount -1 )
      ADOQuery2->MoveBy(1);
  }
  ADOQuery2->Close();

  if(ADOQuery1->Active)
    ADOQuery1->Close();
    ADOQuery1->SQL->Clear();
    ADOQuery1->SQL->Add(sqlStr);
    ADOQuery1->Open();
}
```

Δημιουργούμε μια μεταβλητή sqlStr τύπου AnsiString όπου εκεί θα βάλουμε το ερώτημα, επιλέγουμε όλα τα πεδία από τον πίνακα Region και από τις τέσσερις τοποθεσίες και αφού ενώσουμε τα δεδομένα τα ταξινομούμε ως προς το RegionID, κλείνουμε το ADOQuery2, το καθαρίζουμε, του φορτώνουμε το ερώτημα και το ανοίγουμε, καθαρίζουμε το ComboBox και εκτελούμε πρώτα το ADOQuery2, όσο το i είναι μικρότερο από το αριθμό τον εγγραφών το μετακινούμε κατά μια θέση ώστε μα δείχνει την επόμενη εγγραφή, καταχωρούμε στο ComboBox το περιεχόμενο του πεδίου RegionID σαν αριθμό ακέραιο, αν το i γίνει μικρότερο από την πρώτη εγγραφή τότε το μετακινούμε να δείχνει την πρώτη εγγραφή ώστε να μην μας πετάξει σφάλμα, κλείνουμε το ADOQuery2, αν το ADOQuery1 είναι ενεργό το κλείνουμε, το καθαρίζουμε του φορτώνουμε το ερώτημα και το ανοίγουμε

```
//---------------------------------------------------------------------
void __fastcall TfrmRegion::FormClose(TObject *Sender,
      TCloseAction &Action)
{
((TForm *)this->Owner)->Show();
```

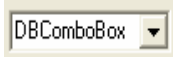
```
}
```

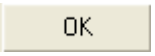
Εμφανίζουμε την κεντρική φόρμα

```
//---------------------------------------------------------------------
void __fastcall TfrmRegion::ComboBox1Change(TObject *Sender)
{
  AnsiString c1 = ComboBox1->Text;
  AnsiString sqlStr = "";
  sqlStr += "SELECT ALLREGION.* FROM ";
  sqlStr += "( ";
  sqlStr += "  SELECT * FROM REGION ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(MYSQL, 'SELECT * FROM REGION')
";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(PART2, 'SELECT * FROM REGION')
";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(ORACLE, 'SELECT * FROM
REGION') ";
  sqlStr += ") ALLREGION ";
  if (c1 != "")
    sqlStr += "WHERE RegionID = '" + c1 + "'";
  sqlStr += "order by RegionID ";
  if(ADOQuery1->Active)
    ADOQuery1->Close();
    ADOQuery1->SQL->Clear();
    ADOQuery1->SQL->Add(sqlStr);
    ADOQuery1->Open();
}
```

Βάζουμε το κείμενο του ComboBox σε μια μεταβλητή c1 τύπου AnsiString, δημιουργούμε άλλη μια μεταβλητή sqlStr τύπου AnsiString για να βάλουμε το ερώτημα, στο ερώτημα επιλέγουμε όλα τα πεδία από τον πίνακα Region και από τις τέσσερις τοποθεσίες και αφού τα ενώσουμε μας εμφανίζει τα δεδομένα όπου το RegionID είναι ίδιο με το περιεχόμενο της μεταβλητής c1 και τα ταξινομούμε ως προς το RegionID, κλείνουμε το ADOQuery1 αν είναι ενεργό, το καθαρίζουμε, του φορτώνουμε το ερώτημα και το ανοίγουμε

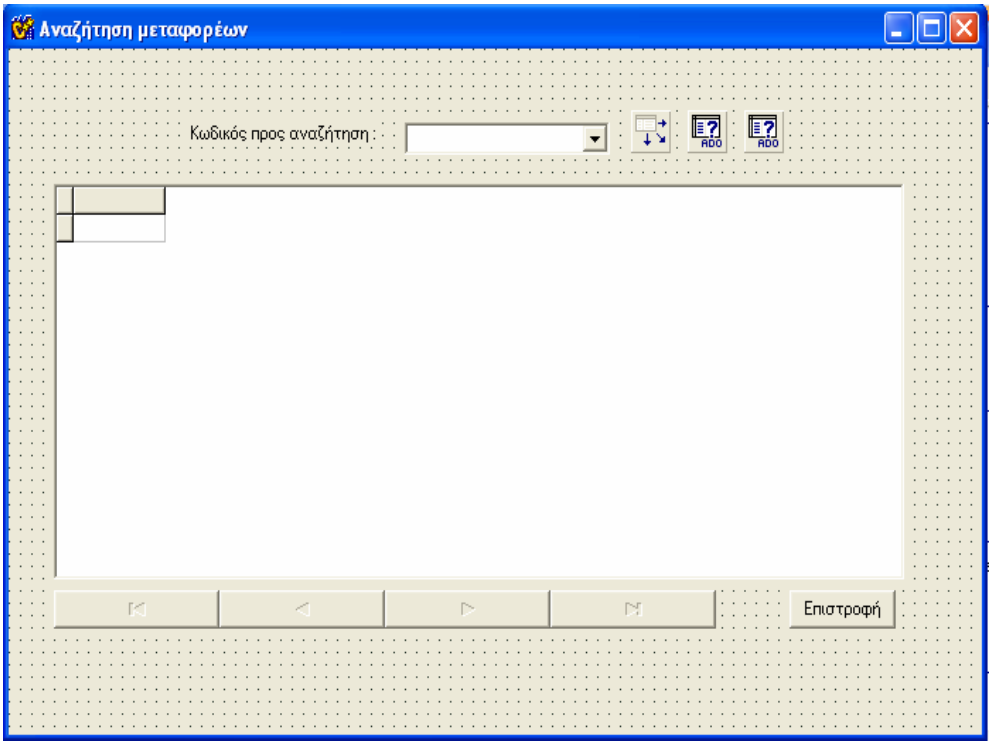
```
//---------------------------------------------------------------------
-
```

Αντικείμενα	**Όνομα**	**Κλάση**	**Δεδομένα**	**Σχόλια**
	**ComboBox1**	TComboBox	Περιλαμβάνει όλους τους κωδικούς των περιοχών	Επιλέγουμε έναν κωδικό περιοχής κάθε φορά μέσα από το σύνολο των κωδικών
	**DataSource1**	TDataSource	-	Διαχείριση των εγγραφών που επιστρέφονται από τους πίνακες Region και από

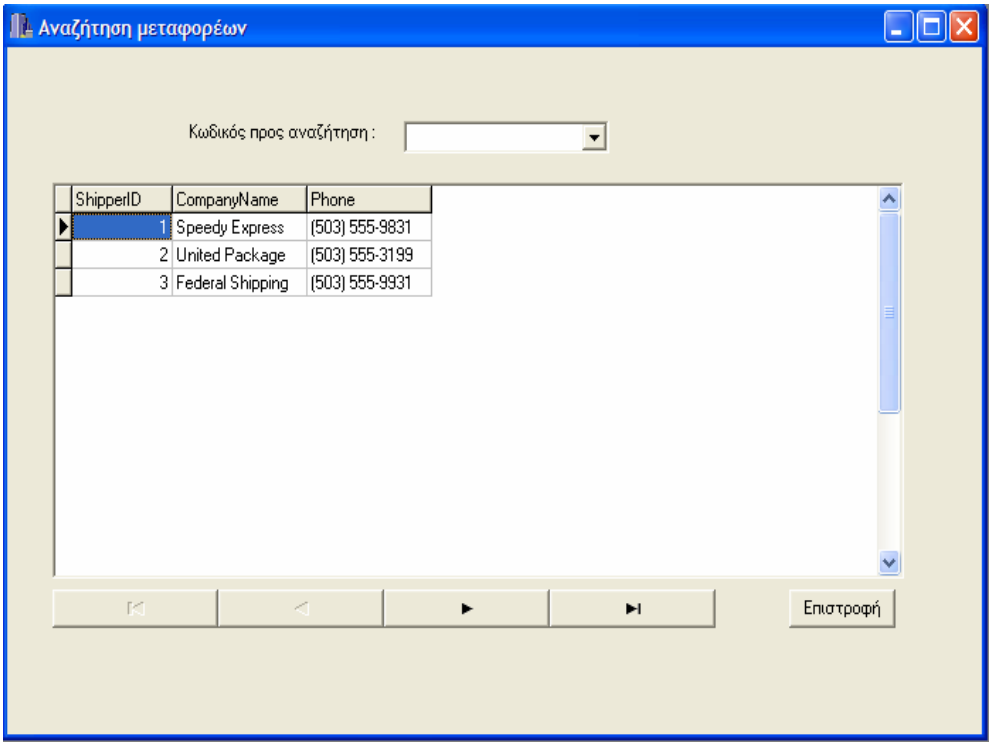
				τις τέσσερις βάσεις μέσω του ADOQuery1
	**ADOQuery1**	TADOQuery	Περιέχει το αποτέλεσμα που προέκυψε από την εκτέλεση του ερωτήματος	Μέσω του ερωτήματος περνούμε τα δεδομένα που ζητάμε από τον πίνακα Region και από τις τέσσερις βάσεις
	ADOQuery2	TADOQuery		
	**DBGrid1**	TDBGrid	Περιέχει τα αποτελέσματα της αναζήτησης	Εμφανίζει την περιοχή που ψάχνουμε
	**Button3**	TButton	-	Με το κουμπί αυτό κλείνουμε την φόρμα
	**DBNavigator**	TDBNavigator	-	Μετακινούμαστε μέσα στις εγγραφές

4.1.7 Φόρμα Αναζήτηση Μεταφορέων

Στην φόρμα αυτή κάνουμε αναζήτηση των μεταφορέων που είναι καταχωρημένες στις βάσεις μας. Για τη δημιουργία της φόρμας χρησιμοποιούμε τα αντικείμενα: **ComboBox, DataSource, ADOQuery, DBGrid, Button, DBNavigator.** Στο ComboBox είναι καταχωρημένοι όλοι οι κωδικοί και επιλέγοντας τον κωδικό μας εμφανίζονται ο αντίστοιχος μεταφορέας. Το DBGrid είναι συνδεδεμένο με το DataSource χρησιμοποιείται για να μας εμφανίζει τα αποτελέσματα της αναζήτησης. Τα ADOQuery χρησιμοποιούνται για να δημιουργήσουμε τα ερωτήματα και να μας εμφανιστούν τα αποτελέσματα στο DBGrid. Το DataSource το χρησιμοποιούμε για την διαχείριση των εγγραφών που επιστρέφονται στο ADOQuery με την εκτέλεση του ερωτήματος. Το αντικείμενο DBNavigator μπορούμε να μετακινιόμαστε πάνω-κάτω μέσα στις εγγραφές. Τέλος με το Button Επιστροφή μεταφερόμαστε στην κεντρική φόρμα



Εικόνα. Η φόρμα σε χρόνο σχεδίασης (Disign Time)



Εικόνα. Η φόρμα σε χρόνο εκτέλεσης (Run Time)

```
//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "ShippersForm.h"
```

```
#include "MainForm.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmShippers *frmShippers;
//---------------------------------------------------------------------
void __fastcall TfrmShippers::FormClose(TObject *Sender, TCloseAction
&Action)
{
((TForm *)this->Owner)->Show();
}
```

Εμφανίζουμε την κεντρική φόρμα

```
//---------------------------------------------------------------------
-
void __fastcall TfrmShippers::FormShow(TObject *Sender)
{
  AnsiString sqlStr = "";
  sqlStr += "SELECT ALLSHIPPERS.* FROM ";
  sqlStr += "( ";
  sqlStr += "  SELECT * FROM SHIPPERS ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(MYSQL, 'SELECT * FROM
SHIPPERS') ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(PART2, 'SELECT * FROM
SHIPPERS') ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(ORACLE, 'SELECT * FROM
SHIPPERS') ";
  sqlStr += ") ALLSHIPPERS ";
  sqlStr += "order by ShipperID ";
  if(ADOQuery2->Active)
    ADOQuery2->Close();
    ADOQuery2->SQL->Clear();
    ADOQuery2->SQL->Add(sqlStr);
    ADOQuery2->Open();

  ComboBox1->Items->Clear();
  ComboBox1->Text = "";
  ADOQuery2->First();
  for(register int i = 0; i < ADOQuery2->RecordCount; i++)
  {
    ComboBox1->Items->Add(ADOQuery2->FieldByName("ShipperID")-
>AsInteger);
    if (i < ADOQuery2->RecordCount -1 )
      ADOQuery2->MoveBy(1);
  }
  ADOQuery2->Close();
```

```
  if(ADOQuery1->Active)
    ADOQuery1->Close();
    ADOQuery1->SQL->Clear();
    ADOQuery1->SQL->Add(sqlStr);
    ADOQuery1->Open();
}
```

Δημιουργούμε μια μεταβλήτη sqlStr τύπου AnsiString όπου εκεί θα δημιουργήσουμε το ερώτημα, επιλέγουμε όλα τα πεδία από τον πίνακα Shippers και από τις τέσσερις τοποθεσίες και αφού τα ενώσουμε τα ταξινομούμε ως προς το ShipperID, κλείνουμε το ADOQuery2, το καθαρίζουμε, του φορτώνουμε το ερώτημα και το ανοίγουμε, καθαρίζουμε το ComboBox και εκτελούμε πρώτα το ADOQuery2, όσο το i είναι μικρότερο από το αριθμό τον εγγραφών το μετακινούμε κατά μια θέση ώστε μα δείχνει την επόμενη εγγραφή, καταχωρούμε στο ComboBox το περιεχόμενο του πεδίου ShipperID σαν ακέραιο αριθμό, αν το i γίνει μικρότερο από την πρώτη εγγραφή τότε το μετακινούμε να δείχνει την πρώτη εγγραφή ώστε να μην μας πετάξει σφάλμα, κλείνουμε το ADOQuery2, αν το ADOQuery1 είναι ενεργό το κλείνουμε, το καθαρίζουμε του φορτώνουμε το ερώτημα και το ανοίγουμε

```
//---------------------------------------------------------------------
void __fastcall TfrmShippers::ComboBox1Change(TObject *Sender)
{
AnsiString c1 = ComboBox1->Text;
  AnsiString sqlStr = "";
  sqlStr += "SELECT ALLSHIPPERS.* FROM ";
  sqlStr += "( ";
  sqlStr += "  SELECT * FROM SHIPPERS ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(MYSQL, 'SELECT * FROM
SHIPPERS') ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(PART2, 'SELECT * FROM
SHIPPERS') ";
  sqlStr += "  UNION  ";
  sqlStr += "  SELECT * FROM OPENQUERY(ORACLE, 'SELECT * FROM
SHIPPERS') ";
  sqlStr += ") ALLSHIPPERS ";
  if (c1 != "")
    sqlStr += "WHERE ShipperID = '" + c1 + "'";
  sqlStr += "order by ShipperID ";
  if(ADOQuery1->Active)
    ADOQuery1->Close();
    ADOQuery1->SQL->Clear();
    ADOQuery1->SQL->Add(sqlStr);
    ADOQuery1->Open();
}
```

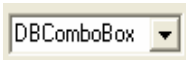
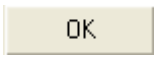
Βάζουμε το κείμενο του ComboBox σε μια μεταβλητή c1 τύπου AnsiString, δημιουργούμε άλλη μια μεταβλητή sqlStr τύπου AnsiString για να βάλουμε το ερώτημα, στο ερώτημα επιλέγουμε όλα τα πεδία από τον πίνακα Shippers και από τις τέσσερις τοποθεσίες και αφού τα ενώσουμε μας εμφανίζει τα δεδομένα όπου το ShipperID είναι ίδιο με το περιεχόμενο της

μεταβλητής c1 και τα ταξινομούμε ως προς το ShipperID, κλείνουμε το ADOQuery1 αν είναι ενεργό, το καθαρίζουμε, του φορτώνουμε το ερώτημα και το ανοίγουμε

```
//---------------------------------------------------------------------
void __fastcall TfrmShippers::Button1Click(TObject *Sender)
{
Close();
}
```

κλείνουμε την φόρμα Αναζήτηση Μεταφορέων

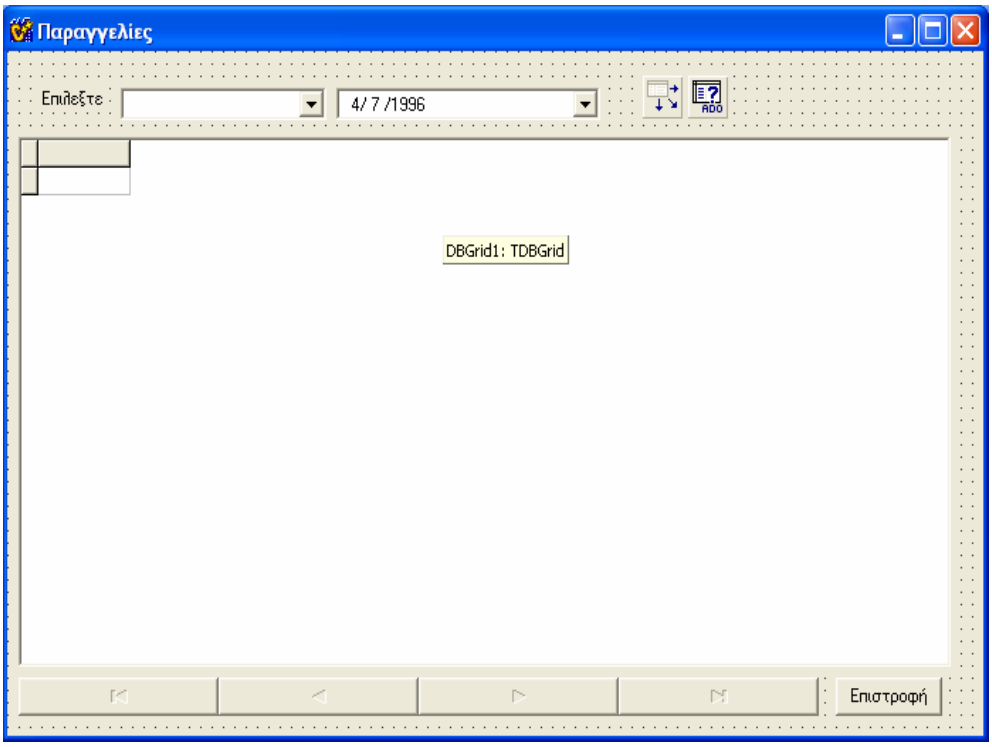
```
//---------------------------------------------------------------------
```

Αντικείμενα	**Όνομα**	**Κλάση**	**Δεδομένα**	**Σχόλια**
	**ComboBox1**	TComboBox	Περιλαμβάνει όλους τους κωδικούς των μεταφορέων	Επιλέγουμε έναν κωδικό μεταφορέα κάθε φορά μέσα από το σύνολο των κωδικών
	**DataSource1**	TDataSource	-	Διαχείριση των εγγραφών που επιστρέφονται από τους πίνακες Shippers και από τις τέσσερις βάσεις μέσω του ADOQuery1
	**ADOQuery1**	TADOQuery	Περιέχει το αποτέλεσμα που προέκυψε από την εκτέλεση του ερωτήματος	Μέσω του ερωτήματος παίρνουμε τα δεδομένα που ζητάμε από τον πίνακα Shippers και από τις τέσσερις βάσεις
	ADOQuery2	TADOQuery		
	**DBGrid1**	TDBGrid	Περιέχει τα αποτελέσματα της αναζήτησης	Εμφανίζει τα στοιχεία του μεταφορέα που ψάχνουμε
	**Button3**	TButton	-	Με το κουμπί αυτό κλείνουμε την φόρμα
	**DBNavigator**	TDBNavigator	-	Μετακινούμαστε μέσα στις εγγραφές

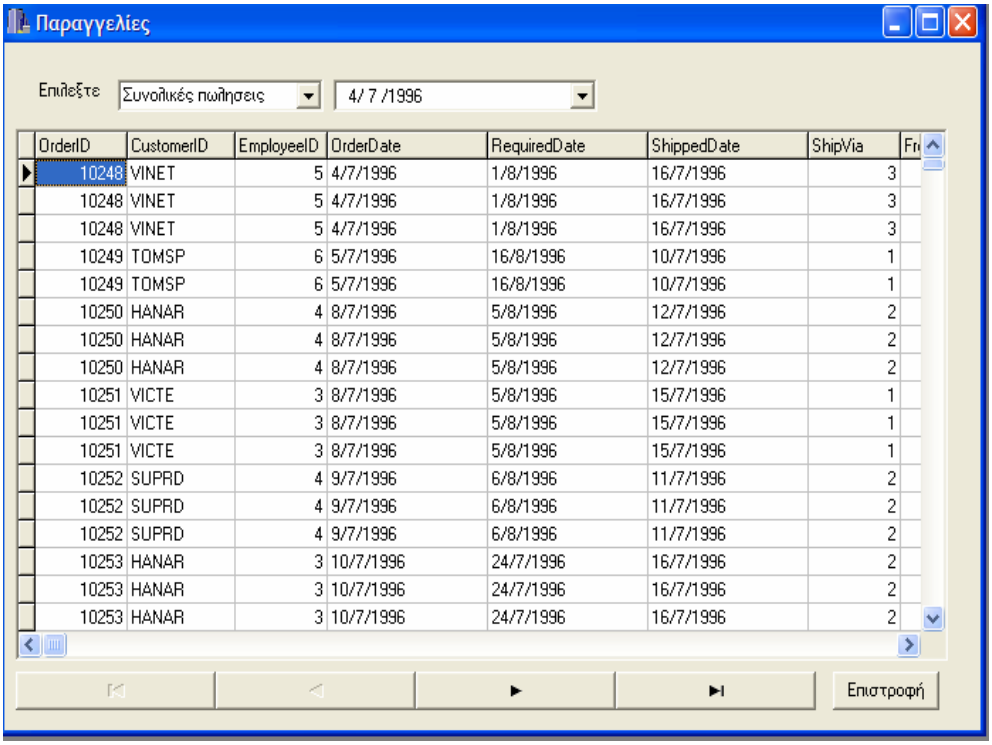
4.1.8 Φόρμα Παραγγελίες

Στην φόρμα αυτή μπορούμε να δούμε διάφορα συγκεντρωτικά στοιχεία για τις πωλήσεις . Για τη δημιουργία της φόρμας χρησιμοποιούμε τα αντικείμενα: **ComboBox, DataSource, ADOQuery, DBGrid, Button, DBNavigator, DateTimePicker.** Στο ComboBox μπορείς να επιλέξεις ένα από τα συγκεντρωτικά πεδία .Στο DateTimePicker επιλέγεις την ημερομηνία , Το DBGrid είναι συνδεδεμένο με το DataSource χρησιμοποιείται για να μας εμφανίζει τα

αποτελέσματα της αναζήτησης. Τα ADOQuery χρησιμοποιούνται για να δημιουργήσουμε τα ερωτήματα και να μας εμφανιστούν τα αποτελέσματα στο DBGrid. Το DataSource το χρησιμοποιούμε για την διαχείριση των εγγραφών που επιστρέφονται στο ADOQuery με την εκτέλεση του ερωτήματος. Το αντικείμενο DBNavigator μπορούμε να μετακινιόμαστε πάνω-κάτω μέσα στις εγγραφές. Τέλος με το Button Επιστροφή μεταφερόμαστε στην κεντρική φόρμα



Εικόνα. Η φόρμα σε χρόνο σχεδίασης (Disign Time)



OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia
10248	VINET	5	4/7/1996	1/8/1996	16/7/1996	3
10248	VINET	5	4/7/1996	1/8/1996	16/7/1996	3
10248	VINET	5	4/7/1996	1/8/1996	16/7/1996	3
10249	TOMSP	6	5/7/1996	16/8/1996	10/7/1996	1
10249	TOMSP	6	5/7/1996	16/8/1996	10/7/1996	1
10250	HANAR	4	8/7/1996	5/8/1996	12/7/1996	2
10250	HANAR	4	8/7/1996	5/8/1996	12/7/1996	2
10250	HANAR	4	8/7/1996	5/8/1996	12/7/1996	2
10251	VICTE	3	8/7/1996	5/8/1996	15/7/1996	1
10251	VICTE	3	8/7/1996	5/8/1996	15/7/1996	1
10251	VICTE	3	8/7/1996	5/8/1996	15/7/1996	1
10252	SUPRD	4	9/7/1996	6/8/1996	11/7/1996	2
10252	SUPRD	4	9/7/1996	6/8/1996	11/7/1996	2
10252	SUPRD	4	9/7/1996	6/8/1996	11/7/1996	2
10253	HANAR	3	10/7/1996	24/7/1996	16/7/1996	2
10253	HANAR	3	10/7/1996	24/7/1996	16/7/1996	2
10253	HANAR	3	10/7/1996	24/7/1996	16/7/1996	2

Εικόνα. Η φόρμα σε χρόνο εκτέλεσης (Run Time)

```
//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "OrdersForm.h"
#include "MainForm.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmOrders *frmOrders;
//---------------------------------------------------------------------
void __fastcall TfrmOrders::Button1Click(TObject *Sender)
{
  if(ADOQuery1->Active)
    ADOQuery1->Close();
    Close();
}
```

Αν το ADOQuery1 είναι ενεργό το κλείνουμε

```
//---------------------------------------------------------------------
void __fastcall TfrmOrders::FormShow(TObject *Sender)
{
  ComboBox1->ItemIndex = 0;
  ComboBox1Change(NULL);
}
//---------------------------------------------------------------------
void __fastcall TfrmOrders::FormClose(TObject *Sender,
      TCloseAction &Action)
{
((TForm *)this->Owner)->Show();
}
```

Εμφανιζουμε την κεντρική φόρμα

```
//---------------------------------------------------------------------
void __fastcall TfrmOrders::ComboBox1Change(TObject *Sender)
{
   if(ComboBox1->ItemIndex == 0)
   {
      try
      {
        if(ADOQuery1->Active)
          ADOQuery1->Active = false;
      }
      catch(...)
      {
      }
```

Αν στο ComboBox βρίσκεται η πρώτη επιλογή κλείνουμε το ADOQuery1 αν είναι ανοιχτό και το Βάζουμε μέσα σε try – catch για να μην μας πετάξει σφάλμα το πρόγραμμα

```
      ADOQuery1->SQL->Clear();
      ADOQuery1->SQL->Add("select o.*, od.* ");
      ADOQuery1->SQL->Add("from ");
      ADOQuery1->SQL->Add("( ");
      ADOQuery1->SQL->Add("  SELECT * FROM ORDERS ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(MYSQL, 'SELECT *
FROM ORDERS') ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(PART2, 'SELECT *
FROM ORDERS') ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(ORACLE, 'SELECT
* FROM ORDERS') ");
      ADOQuery1->SQL->Add(") o ");
      ADOQuery1->SQL->Add("inner join ");
      ADOQuery1->SQL->Add("( ");
      ADOQuery1->SQL->Add("  SELECT * FROM ORDERDETAILS ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(MYSQL, 'SELECT *
FROM ORDERDETAILS') ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(PART2, 'SELECT *
FROM ORDERDETAILS') ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(ORACLE, 'SELECT
* FROM ORDERDETAILS') ");
      ADOQuery1->SQL->Add(") od on o.orderID=od.orderID ");
      ADOQuery1->Open();
   }
```

Αρχικά καθαρίζουμε το ADOQuery1, φορτώνουμε το ερώτημα στο ADOQuery1 επιλέγουμε όλα τα πεδία από τον πίνακα Orders και Orderdetails και από τις τέσσερις τοποθεσίες, τα ενώνουμε , επιλέγουμε να εμφανιστούν τα δεδομένα όπου το OrdeID είναι ίδιο και στους δυο πίνακες και ανοίγουμε το ADOQuery1

```
   else if(ComboBox1->ItemIndex == 1)
   {
      try
      {
        if(ADOQuery1->Active)
          ADOQuery1->Active = false;
      }
      catch(...)
      {
      }
```

Αν στο ComboBox βρίσκεται στη δεύτερη επιλογή κλείνουμε το ADOQuery1 άμα είναι ανοιχτό και το Βάζουμε μέσα σε try – catch για να μην μας πετάξει σφάλμα το πρόγραμμα

```
      ADOQuery1->SQL->Clear();
```

```
      ADOQuery1->SQL->Add("select od.ProductID, ");
      ADOQuery1->SQL->Add("       p.ProductName, ");
      ADOQuery1->SQL->Add("       SUM(od.quantity) as Quantity, ");
      ADOQuery1->SQL->Add("       SUM(od.quantity * od.unitprice) as
[Total Price] ");
      ADOQuery1->SQL->Add("from ");
      ADOQuery1->SQL->Add("( ");
      ADOQuery1->SQL->Add("  SELECT * FROM ORDERS ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(MYSQL, 'SELECT *
FROM ORDERS') ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(PART2, 'SELECT *
FROM ORDERS') ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(ORACLE, 'SELECT
* FROM ORDERS') ");
      ADOQuery1->SQL->Add(") o ");
      ADOQuery1->SQL->Add("inner join ");
      ADOQuery1->SQL->Add("( ");
      ADOQuery1->SQL->Add("  SELECT * FROM ORDERDETAILS ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(MYSQL, 'SELECT *
FROM ORDERDETAILS') ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(PART2, 'SELECT *
FROM ORDERDETAILS') ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(ORACLE, 'SELECT
* FROM ORDERDETAILS') ");
      ADOQuery1->SQL->Add(") od on o.orderID=od.orderID ");
      ADOQuery1->SQL->Add(" inner join ");
      ADOQuery1->SQL->Add("( ");
      ADOQuery1->SQL->Add("  SELECT * FROM PRODUCT ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(MYSQL, 'SELECT *
FROM PRODUCTS') ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(PART2, 'SELECT *
FROM PRODUCT') ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(ORACLE, 'SELECT
* FROM PRODUCTS') ");
      ADOQuery1->SQL->Add(") p on p.ProductID=od.ProductID ");
      ADOQuery1->SQL->Add("group by od.ProductID, p.ProductName ");
      ADOQuery1->SQL->Add("order by od.ProductID ");
      ADOQuery1->Open();
   }
```

Καθαρίζουμε το ADOQuery1, επιλέγουμε ποια πεδία θέλουμε να εμφανιστούν, επιλέγουμε όλα τα πεδία από τους πίνακες Orders, Orderdetails και Product και από τις τέσσερις τοποθεσίες, τα ενώνουμε, εμφανίζει τα πεδία όπου το OrderID είναι ίδιο στον πίνακα Orders και Order details καθώς και το ProductID είναι ίδιο στον πίνακα Products και Orderdetails, τα ομαδοποιούμε, τα ταξινομούμε και ανοίγουμε το ADOQuery1.

```
   else if(ComboBox1->ItemIndex == 2)
   {
      try
      {
        if(ADOQuery1->Active)
          ADOQuery1->Active = false;
      }
      catch(...)
      {
      }
```

Αν στο ComboBox βρίσκεται στη τρίτη επιλογή κλείνουμε το ADOQuery1 άμα είναι ανοιχτό και το Βάζουμε μέσα σε try - catch για να μην μας πετάξει σφάλμα το πρόγραμμα

```
      ADOQuery1->SQL->Clear();
      ADOQuery1->SQL->Add("select o.EmployeeID, ");
      ADOQuery1->SQL->Add("       e.LastName, ");
      ADOQuery1->SQL->Add("       e.FirstName, ");
      ADOQuery1->SQL->Add("       od.ProductID, ");
      ADOQuery1->SQL->Add("       p.ProductName, ");
      ADOQuery1->SQL->Add("       SUM(od.quantity) as Quantity, ");
      ADOQuery1->SQL->Add("       SUM(od.quantity * od.unitprice) as
[Total Price] ");
      ADOQuery1->SQL->Add("from ");
      ADOQuery1->SQL->Add("( ");
      ADOQuery1->SQL->Add("  SELECT * FROM ORDERS ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(MYSQL, 'SELECT *
FROM ORDERS') ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(PART2, 'SELECT *
FROM ORDERS') ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(ORACLE, 'SELECT
* FROM ORDERS') ");
      ADOQuery1->SQL->Add(") o ");
      ADOQuery1->SQL->Add("inner join ");
      ADOQuery1->SQL->Add("( ");
      ADOQuery1->SQL->Add("  SELECT * FROM ORDERDETAILS ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(MYSQL, 'SELECT *
FROM ORDERDETAILS') ");
      ADOQuery1->SQL->Add("  UNION  ");
```

```
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(PART2, 'SELECT *
FROM ORDERDETAILS') ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(ORACLE, 'SELECT
* FROM ORDERDETAILS') ");
      ADOQuery1->SQL->Add(") od on o.orderID=od.orderID ");
      ADOQuery1->SQL->Add("inner join ");
      ADOQuery1->SQL->Add("( ");
      ADOQuery1->SQL->Add("  SELECT * FROM EMPLOYEES ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(MYSQL, 'SELECT *
FROM EMPLOYEES') ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(PART2, 'SELECT *
FROM EMPLOYEES') ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(ORACLE, 'SELECT
* FROM EMPLOYEES') ");
      ADOQuery1->SQL->Add(") e on e.EmployeeID=o.EmployeeID ");
      ADOQuery1->SQL->Add("inner join ");
      ADOQuery1->SQL->Add("( ");
      ADOQuery1->SQL->Add("  SELECT * FROM PRODUCT ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(MYSQL, 'SELECT *
FROM PRODUCTS') ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(PART2, 'SELECT *
FROM PRODUCT') ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(ORACLE, 'SELECT
* FROM PRODUCTS') ");
      ADOQuery1->SQL->Add(") p on p.ProductID=od.ProductID ");
      ADOQuery1->SQL->Add("group by o.EmployeeID, e.LastName,
e.FirstName, od.ProductID, p.ProductName ");
      ADOQuery1->SQL->Add("order by o.EmployeeID, od.ProductID ");
      ADOQuery1->Open();
   }
```

Καθαρίζουμε το ADOQuery1, επιλέγουμε ποια πεδία θέλουμε να εμφανιστούν, επιλέγουμε όλα τα πεδία από τους πίνακες Orders, OrderDetails, Emloyees και Product και από τις τέσσερις τοποθεσίες, τα ενώνουμε, μας εμφανίζει τα πεδία όπου το OrderID είναι ίδιο στον πίνακα Orders και Order Details, το ProductID είναι ίδιο στον πίνακα Products και Order Details καθώς και το EmloyeeID είναι ίδιο στον πίνακα Employees και Orders, τα ομαδοποιούμε, τα ταξινομούμε και ανοίγουμε το ADOQuery1.

```
   else if(ComboBox1->ItemIndex == 3)
   {
      try
      {
        if(ADOQuery1->Active)
```

```
        ADOQuery1->Active = false;
    }
    catch(...)
    {
    }
```

Αν στο ComboBox βρίσκεται στη τέταρτη επιλογή κλείνουμε το ADOQuery1 άμα είναι ανοιχτό και το Βάζουμε μέσα σε try – catch για να μην μας πετάξει σφάλμα το πρόγραμμα

```
    ADOQuery1->SQL->Clear();
    ADOQuery1->SQL->Add("select o.OrderDate, ");
    ADOQuery1->SQL->Add("       od.ProductID, ");
    ADOQuery1->SQL->Add("       p.ProductName, ");
    ADOQuery1->SQL->Add("       SUM(od.quantity) as Quantity, ");
    ADOQuery1->SQL->Add("       SUM(od.quantity * od.unitprice) as
[Total Price] ");
    ADOQuery1->SQL->Add("from ");
    ADOQuery1->SQL->Add("( ");
    ADOQuery1->SQL->Add("  SELECT * FROM ORDERS ");
    ADOQuery1->SQL->Add("  UNION  ");
    ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(MYSQL, 'SELECT *
FROM ORDERS') ");
    ADOQuery1->SQL->Add("  UNION  ");
    ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(PART2, 'SELECT *
FROM ORDERS') ");
    ADOQuery1->SQL->Add("  UNION  ");
    ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(ORACLE, 'SELECT
* FROM ORDERS') ");
    ADOQuery1->SQL->Add(") o ");
    ADOQuery1->SQL->Add("inner join ");
    ADOQuery1->SQL->Add("( ");
    ADOQuery1->SQL->Add("  SELECT * FROM ORDERDETAILS ");
    ADOQuery1->SQL->Add("  UNION  ");
    ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(MYSQL, 'SELECT *
FROM ORDERDETAILS') ");
    ADOQuery1->SQL->Add("  UNION  ");
    ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(PART2, 'SELECT *
FROM ORDERDETAILS') ");
    ADOQuery1->SQL->Add("  UNION  ");
    ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(ORACLE, 'SELECT
* FROM ORDERDETAILS') ");
    ADOQuery1->SQL->Add(") od on o.orderID=od.orderID ");
    ADOQuery1->SQL->Add(" inner join ");
    ADOQuery1->SQL->Add("( ");
    ADOQuery1->SQL->Add("  SELECT * FROM PRODUCT ");
    ADOQuery1->SQL->Add("  UNION  ");
    ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(MYSQL, 'SELECT *
FROM PRODUCTS') ");
    ADOQuery1->SQL->Add("  UNION  ");
```

```
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(PART2, 'SELECT *
FROM PRODUCT') ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(ORACLE, 'SELECT
* FROM PRODUCTS') ");
      ADOQuery1->SQL->Add(") p on p.ProductID=od.ProductID ");
      ADOQuery1->SQL->Add("group by o.OrderDate, od.ProductID,
p.ProductName ");
      ADOQuery1->SQL->Add("order by o.OrderDate, od.ProductID ");
      ADOQuery1->Open();
   }
```

Καθαρίζουμε το ADOQuery1, επιλέγουμε ποια πεδία θέλουμε να εμφανιστούν, επιλέγουμε όλα τα πεδία από τους πίνακες Orders, Orderdetails και Product και από τις τέσσερις τοποθεσίες, τα ενώνουμε, μας εμφανίζει τις εγγραφές των πεδίων που θέλουμε όπου το OrderID είναι ίδιο στον πίνακα Orders και Order Details, το ProductID είναι ίδιο στον πίνακα Product και Order Details , τα ομαδοποιούμε, τα ταξινομούμε και ανοίγουμε το ADOQuery1.

```
   else if(ComboBox1->ItemIndex == 4)
   {
      try
      {
        if(ADOQuery1->Active)
          ADOQuery1->Active = false;
      }
      catch(...)
      {
      }
```

Αν στο ComboBox βρίσκεται στη πέμπτη επιλογή κλείνουμε το ADOQuery1 άμα είναι ανοιχτό και το Βάζουμε μέσα σε try – catch για να μην μας πετάξει σφάλμα το πρόγραμμα

```
      ADOQuery1->SQL->Clear();
      ADOQuery1->SQL->Add("select YEAR(o.OrderDate) as [Sales Year],
");
      ADOQuery1->SQL->Add("       SUM(od.quantity * od.unitprice) as
[Total Sales] ");
      ADOQuery1->SQL->Add("from ");
      ADOQuery1->SQL->Add("( ");
      ADOQuery1->SQL->Add("  SELECT * FROM ORDERS ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(MYSQL, 'SELECT *
FROM ORDERS') ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(PART2, 'SELECT *
FROM ORDERS') ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(ORACLE, 'SELECT
* FROM ORDERS') ");
      ADOQuery1->SQL->Add(") o ");
      ADOQuery1->SQL->Add("inner join ");
```

```
      ADOQuery1->SQL->Add("( ");
      ADOQuery1->SQL->Add("  SELECT * FROM ORDERDETAILS ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(MYSQL, 'SELECT *
FROM ORDERDETAILS') ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(PART2, 'SELECT *
FROM ORDERDETAILS') ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(ORACLE, 'SELECT
* FROM ORDERDETAILS') ");
      ADOQuery1->SQL->Add(") od on o.orderID=od.orderID ");
      ADOQuery1->SQL->Add("group by YEAR(o.OrderDate) ");
      ADOQuery1->SQL->Add("order by YEAR(o.OrderDate) ");
      ADOQuery1->Open();
   }
```

Καθαρίζουμε το ADOQuery1, επιλέγουμε ποια πεδία θέλουμε να εμφανιστούν, επιλέγουμε όλα τα πεδία από τους πίνακες Orders, Orderdetails και από τις τέσσερις τοποθεσίες, τα ενώνουμε, μας εμφανίζει τις εγγραφές όπου το OrderID είναι ίδιο στον πίνακα Orders και Order Details, τα ομαδοποιούμε, τα ταξινομούμε και ανοίγουμε το ADOQuery1.

```
   else if(ComboBox1->ItemIndex == 5)
   {
      try
      {
        if(ADOQuery1->Active)
          ADOQuery1->Active = false;
      }
      catch(...)
      {
      }
```

Αν στο ComboBox βρίσκεται στη έκτη επιλογή κλείνουμε το ADOQuery1 άμα είναι ανοιχτό και το Βάζουμε μέσα σε try – catch για να μην μας πετάξει σφάλμα το πρόγραμμα

```
      ADOQuery1->SQL->Clear();
      ADOQuery1->SQL->Add("select o.ShipRegion, ");
      ADOQuery1->SQL->Add("       SUM(od.quantity * od.unitprice) as
[Total Sales] ");
      ADOQuery1->SQL->Add("from ");
      ADOQuery1->SQL->Add("( ");
      ADOQuery1->SQL->Add("  SELECT * FROM ORDERS ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(MYSQL, 'SELECT *
FROM ORDERS') ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(PART2, 'SELECT *
FROM ORDERS') ");
      ADOQuery1->SQL->Add("  UNION  ");
```

```
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(ORACLE, 'SELECT
* FROM ORDERS') ");
      ADOQuery1->SQL->Add(") o ");
      ADOQuery1->SQL->Add("inner join ");
      ADOQuery1->SQL->Add("( ");
      ADOQuery1->SQL->Add("  SELECT * FROM ORDERDETAILS ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(MYSQL, 'SELECT *
FROM ORDERDETAILS') ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(PART2, 'SELECT *
FROM ORDERDETAILS') ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(ORACLE, 'SELECT
* FROM ORDERDETAILS') ");
      ADOQuery1->SQL->Add(") od on o.orderID=od.orderID ");
      ADOQuery1->SQL->Add("group by o.ShipRegion ");
      ADOQuery1->SQL->Add("order by o.ShipRegion ");
      ADOQuery1->Open();
   }
```

Καθαρίζουμε το ADOQuery1, επιλέγουμε ποια πεδία θέλουμε να εμφανιστούν, επιλέγουμε όλα τα πεδία από τους πίνακες Orders, Orderdetails και από τις τέσσερις τοποθεσίες, τα ενώνουμε, μας εμφανίζει τις εγγραφές όπου το OrderID είναι ίδιο στον πίνακα Orders και Order Details, τα ομαδοποιούμε, τα ταξινομούμε και ανοίγουμε το ADOQuery1.

```
   else if(ComboBox1->ItemIndex == 6)
   {
      try
      {
        if(ADOQuery1->Active)
          ADOQuery1->Active = false;
      }
      catch(...)
      {
      }
```

Αν στο ComboBox βρίσκεται στη έβδομη επιλογή κλείνουμε το ADOQuery1 άμα είναι ανοιχτό και το Βάζουμε μέσα σε try – catch για να μην μας πετάξει σφάλμα το πρόγραμμα

```
      AnsiString c = DateTimePicker1->Date.FormatString("yyyymmdd");

      ADOQuery1->SQL->Clear();
      ADOQuery1->SQL->Add("select Count(*) as cnt ");
      ADOQuery1->SQL->Add("from ");
      ADOQuery1->SQL->Add("( ");
      ADOQuery1->SQL->Add("  SELECT * FROM ORDERS ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(MYSQL, 'SELECT *
FROM ORDERS') ");
      ADOQuery1->SQL->Add("  UNION  ");
```

```
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(PART2, 'SELECT *
FROM ORDERS') ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(ORACLE, 'SELECT
* FROM ORDERS') ");
      ADOQuery1->SQL->Add(") o ");
      ADOQuery1->SQL->Add("inner join ");
      ADOQuery1->SQL->Add("( ");
      ADOQuery1->SQL->Add("  SELECT * FROM ORDERDETAILS ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(MYSQL, 'SELECT *
FROM ORDERDETAILS') ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(PART2, 'SELECT *
FROM ORDERDETAILS') ");
      ADOQuery1->SQL->Add("  UNION  ");
      ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(ORACLE, 'SELECT
* FROM ORDERDETAILS') ");
      ADOQuery1->SQL->Add(") od on o.orderID=od.orderID ");
      ADOQuery1->SQL->Add("Where o.OrderDate = '" + c + "' ");
      ADOQuery1->Open();
      int cnt = ADOQuery1->FieldByName("cnt")->AsInteger;
      ADOQuery1->Close();
```

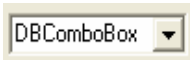
Αρχικά τοποθετούμε το περιεχόμενο του DateTimePicker σε μια μεταβλητή c τύπου AnsiString, καθαρίζουμε το ADOQuery1, επιλέγουμε να εμφανιστούν όλα τα πεδία από τους πίνακες Orders και Orderdetails και από τις τέσσερις τοποθεσίες, τα ενώνουμε, και εμφανίζοντε οι εγγραφές όπου το OrderID είναι ίδιο στον πίνακα Orders και Orderdetails καθώς και το OrderDate είναι ίδιο με τη μεταβλητή c

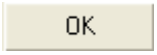
```
if (cnt > 0)
      {
        ADOQuery1->SQL->Clear();
        ADOQuery1->SQL->Add("select * ");
        ADOQuery1->SQL->Add("from ");
        ADOQuery1->SQL->Add("( ");
        ADOQuery1->SQL->Add("  SELECT * FROM ORDERS ");
        ADOQuery1->SQL->Add("  UNION  ");
        ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(MYSQL, 'SELECT
* FROM ORDERS') ");
        ADOQuery1->SQL->Add("  UNION  ");
        ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(PART2, 'SELECT
* FROM ORDERS') ");
        ADOQuery1->SQL->Add("  UNION  ");
        ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(ORACLE,
'SELECT * FROM ORDERS') ");
        ADOQuery1->SQL->Add(") o ");
        ADOQuery1->SQL->Add("inner join ");
        ADOQuery1->SQL->Add("( ");
        ADOQuery1->SQL->Add("  SELECT * FROM ORDERDETAILS ");
        ADOQuery1->SQL->Add("  UNION  ");
```

```
        ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(MYSQL, 'SELECT
* FROM ORDERDETAILS') ");
        ADOQuery1->SQL->Add("  UNION  ");
        ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(PART2, 'SELECT
* FROM ORDERDETAILS') ");
        ADOQuery1->SQL->Add("  UNION  ");
        ADOQuery1->SQL->Add("  SELECT * FROM OPENQUERY(ORACLE,
'SELECT * FROM ORDERDETAILS') ");
        ADOQuery1->SQL->Add(") od on o.orderID=od.orderID ");
        ADOQuery1->SQL->Add("Where o.OrderDate = '" + c + "' ");
        ADOQuery1->SQL->Add("order by o.OrderDate, o.CustomerId ");
        ADOQuery1->Open();
      }
   }
}
```

Αν η μεταβλητή cnt είναι μεγαλύτερη του μηδέν τότε καθαρίζουμε το ADOQuery1, επιλέγουμε να εμφανίσουν όλα τα πεδία από τους δυο πίνακες και από τις τέσσερις τοποθεσίες, τα ενώνουμε και μας εμφανίζει τις εγγραφές των πεδίων που έχουν κοινό OrderID και το OrderDate είναι ίδιο με το περιεχόμενο της μεταβλητής και ανοίγουμε το ADOQuery1

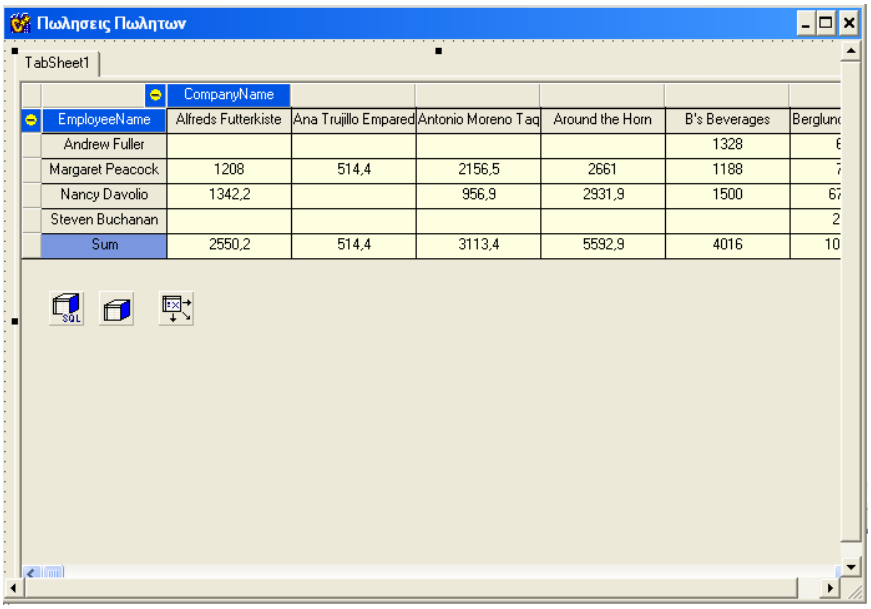
```
//---------------------------------------------------------------------
```

Αντικείμενα	**Όνομα**	**Κλάση**	**Δεδομένα**	**Σχόλια**
	**ComboBox1**	TComboBox	Περιλαμβάνει ορισμένα συγκεντρωτικά πεδία	Επιλέγουμε ένα συγκεντρωτικό πεδίο κάθε φορά
	**DateTimePicker1**	TDateTimePicker	-	Μας Εμφανίζει ένα ημερολόγιο και επιλέγουμε την ημερομηνία που θέλουμε να δούμε τα συγκεντρωτικά αποτελέσματα
	**DataSource1**	TDataSource	-	Διαχείριση των εγγραφών που επιστρέφονται από τον πίνακα Orders και από τις τέσσερις βάσεις μέσω του ADOQuery1
	**ADOQuery1**	TADOQuery	Περιέχει το αποτέλεσμα που προέκυψε από την εκτέλεση του	Μέσω του ερωτήματος παίρνουμε τα δεδομένα που ζητάμε από τον

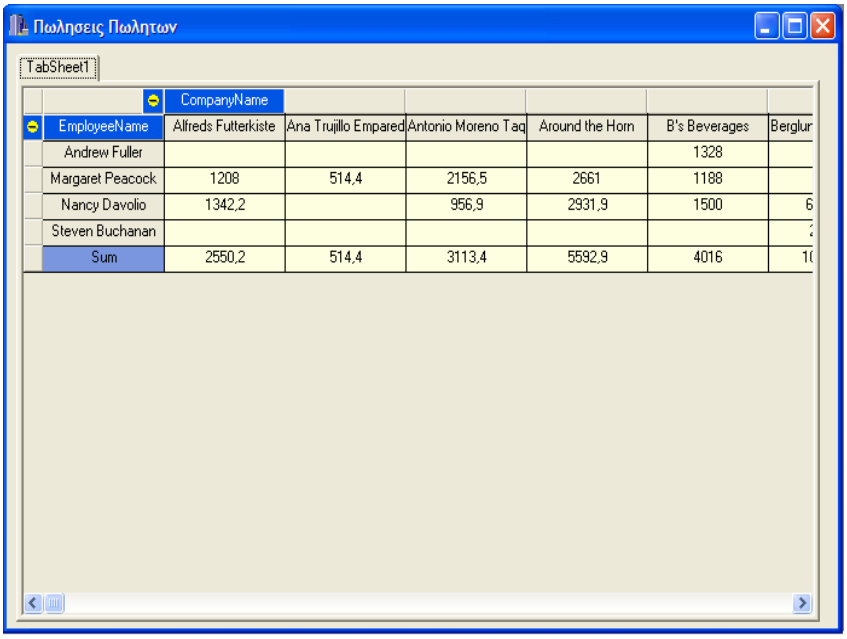
			ερωτήματος	πίνακα Orders και από τις τέσσερις βάσεις
	**DBGrid1**	TDBGrid	Περιέχει τα αποτελέσματα της αναζήτησης	Εμφανίζει τα στοιχεία του μεταφορέα που ψάχνουμε
	**Button3**	TButton	-	Με το κουμπί αυτό κλείνουμε την φόρμα
	**DBNavigator**	TDBNavigator	-	Μετακινούμαστε μέσα στις εγγραφές

4.1.9 Φόρμα Πωλήσεις Πωλητών

Στην φόρμα αυτή μπορούμε να δούμε τις πωλήσεις των πωλητών υπό μορφή κύβου . Για τη δημιουργία της φόρμας χρησιμοποιούμε τα αντικείμενα: **PageControl, DecisionGrid, DecisionSource, DecisionCube, Decision Query.** Το Decision Query περιέχει το ερώτημα που θα εκτελεστή και είναι συνδεδεμένο με το DecisionCube, Το DecisionCube παίρνει τα αποτεσματα από την εκτέλεση του ερωτήματος και τα μεταφέρει στο DecisionSource σε μορφή κύβου . Το DecisionGrid είναι συνδεδεμένο με το DecisionSource και χρησιμοποιείται για να μας εμφανίζει τα αποτελέσματα του ερωτήματος το οποίο έχουμε τοποθετήσει πάνω σε ένα Page Control . Το DecisionSource το χρησιμοποιούμε για να μπορούμε να αλάζουμε τις διαστάσεις του κύβου και να βλέπουμε τα αποτελέσματα από διαφορετική οπτική γωνία κάθε φορά.



Εικόνα. Η φόρμα σε χρόνο σχεδίασης (Disign Time)



Εικόνα. Η φόρμα σε χρόνο εκτέλεσης (Run Time)

```
//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "CubeForm1.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmSalesBySalesmanCube *frmSalesBySalesmanCube;
//---------------------------------------------------------------------
void __fastcall TfrmSalesBySalesmanCube::FormClose(TObject *Sender,
TCloseAction &Action)
{
 ((TForm *)this->Owner)->Show();
}
```

Εμφανίζουμε την κεντρική φόρμα

```
//---------------------------------------------------------------------
void __fastcall TfrmSalesBySalesmanCube::FormShow(TObject *Sender)
{
  if (DecisionQuery1->Active)
    DecisionQuery1->Active = false;
    DecisionQuery1->SQL->Clear();
    DecisionQuery1->SQL->Add("SELECT CompanyName, Customerid,
EmployeeName, SUM( TotalSales ) ");
    DecisionQuery1->SQL->Add("FROM dbo.VW_SalesByEmployee ");
    DecisionQuery1->SQL->Add("GROUP BY CompanyName, Customerid,
EmployeeName");

  DecisionQuery1->Active = true;
}
```

Απενεργοποιούμε το DecisionQuery1 αν αυτό είναι ενεργό, το καθαρίζουμε και του

φορτώνουμε το ερώτημα, επιλέγουμε τα πεδία που θέλουμε να εμφανίσουμε που τα παίρνουμε από το ερώτημα VW_SalesByDayAll, τα ομαδοποιούμε και τρέχουμε το DecisionQuery1

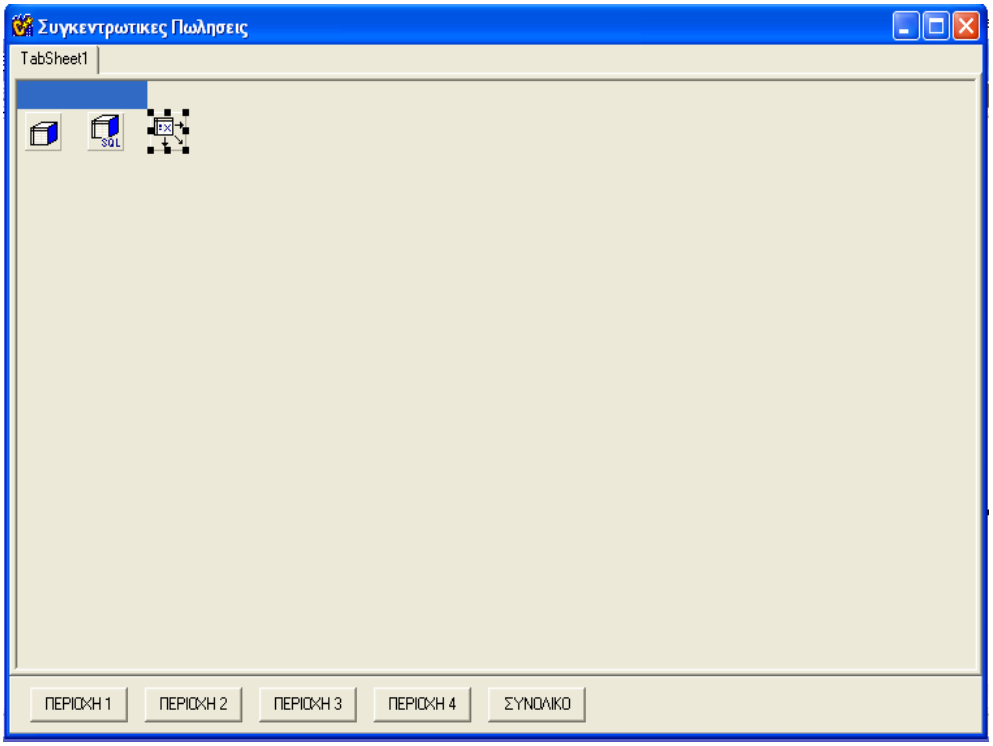
//----------

Αντικείμενα	Όνομα	Κλάση	Δεδομένα	Σχόλια
	**Page Control**	T PageContro		
	**Decision Query**	T Decision Query	Περιέχει τα αποτελέσματα που προέκυψαν από την εκτέλεση του ερωτήματος	
	**Decision Cube**	T Decision Cube	Περνει τα δεδομένα από το Decision Query και τα μεταφέρει στο Decision Source σε μορφή κύβου	
	**Decision Source**	T Decision Source	-	Διαχειρίζεται τις διαστάσεις του κύβου
	**Decision Grid**	T Decision Grid	Περιέχει τα αποτελέσματα του ερωτήματος	Εμφανίζει τα αποτελέσματα σε μορφή κύβου και είναι συνδεδεμένο με το Decision Source

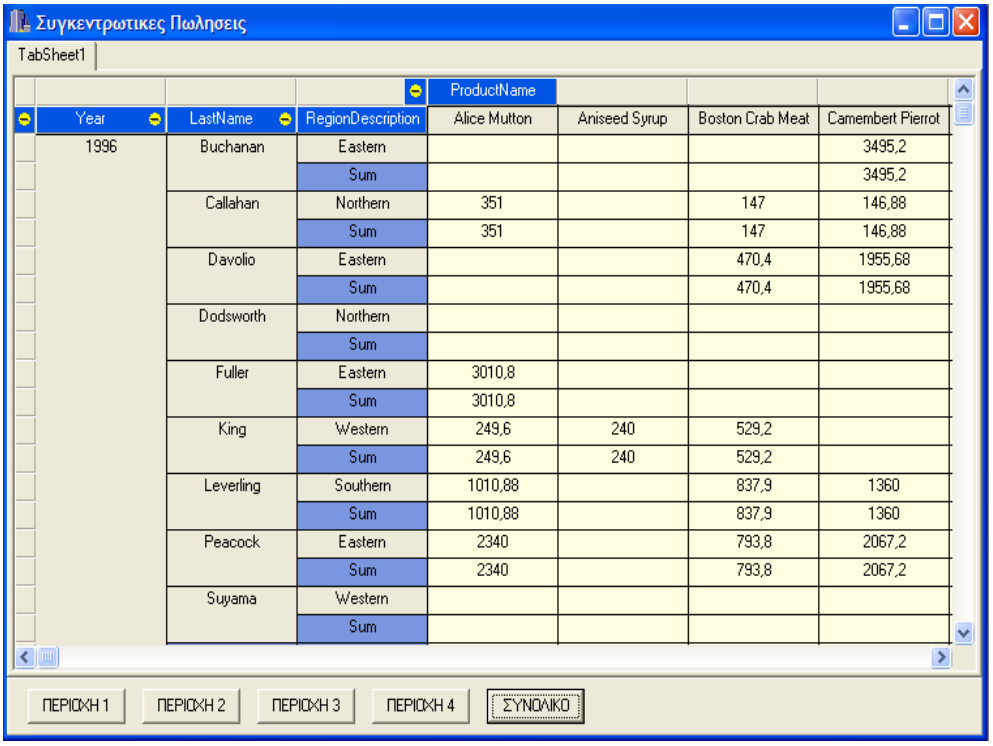
4.1.10Φόρμα Συγκεντρωτικές πωλήσεις

Στην φόρμα αυτή μπορούμε να δούμε τις συγκεντρωτικές πωλήσεις υπό μορφή κύβου . Για τη δημιουργία της φόρμας χρησιμοποιούμε τα αντικείμενα: **PageControl,DecisionGrid, DecisionSource, DecisionCube, Decision Query, Edit.** Το Decision Query Περιέχει το ερώτημα που θα εκτελεστή και είναι συνδεδεμένο με το DecisionCube, Το DecisionCube παίρνει τα αποτελέσματα από την εκτέλεση του ερωτήματος και τα μεταφέρει στο DecisionSource σε μορφή κύβου . Το DecisionGrid είναι συνδεδεμένο με το DecisionSource και χρησιμοποιείται για να μας εμφανίζει τα αποτελέσματα της αναζήτησης το οποίο το έχουμε τοποθετήσει πάνω σε ένα Page Control . Το Decision Source το χρησιμοποιούμε για να μπορούμε να αλάζουμε τις διαστάσεις του κύβου και να βλέπουμε τα αποτελέσματα από διαφορετική οπτική γωνία κάθε φορά. Στα τέσσερα πρώτα Edit (Περιοχη1, Περιοχη2, Περιοχη3, Περιοχη4)μας εμφανίζουν τις συγκεντρωτικές πωλήσεις σε κάθε μια βάση

ξεχωριστά ενώ το τελευταίο Edit (Συνολικό) μας εμφανίζονται οι συγκεντρωτικές πωλήσεις και από τις τέσσερις βάσεις μαζί.



Εικόνα. Η φόρμα σε χρόνο σχεδίασης (Disign Time)



			ProductName			
Year	LastName	RegionDescription	Alice Mutton	Aniseed Syrup	Boston Crab Meat	Camembert Pierrot
1996	Buchanan	Eastern				3495,2
		Sum				3495,2
	Callahan	Northern	351		147	146,88
		Sum	351		147	146,88
	Davolio	Eastern			470,4	1955,68
		Sum			470,4	1955,68
	Dodsworth	Northern				
		Sum				
	Fuller	Eastern	3010,8			
		Sum	3010,8			
	King	Western	249,6	240	529,2	
		Sum	249,6	240	529,2	
	Leverling	Southern	1010,88		837,9	1360
		Sum	1010,88		837,9	1360
	Peacock	Eastern	2340		793,8	2067,2
		Sum	2340		793,8	2067,2
	Suyama	Western				
		Sum				

Εικόνα. Η φόρμα σε χρόνο εκτέλεσης (Run Time)

```
//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "CubeForm2.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmTotalSalesCube *frmTotalSalesCube;
//---------------------------------------------------------------------
void __fastcall TfrmTotalSalesCube::FormClose(TObject *Sender,
TCloseAction &Action)
{
 ((TForm *)this->Owner)->Show();
}
```

Εμφανίζουμε την κεντρική φόρμα

```
//---------------------------------------------------------------------
void __fastcall TfrmTotalSalesCube::Button1Click(TObject *Sender)
{
  if (DecisionQuery1->Active)
    DecisionQuery1->Active = false;

  DecisionQuery1->SQL->Clear();
  DecisionQuery1->SQL->Add("SELECT ProductName, DYear, LastName,
RegionDescription, SUM( TotalSales ) ");
  DecisionQuery1->SQL->Add("FROM VW_SalesFromPart1 ");
  DecisionQuery1->SQL->Add("GROUP BY ProductName, DYear, LastName,
regionDescription ");
  DecisionQuery1->Active = true;
}
```

Κλείνουμε το DecisionQuery1 άμα αυτό είναι ενεργό, το καθαρίζουμε και επιλέγουμε τα πεδία που θέλουμε να εμφανιστούν από το ερώτημα VW_SalesFromPart1, τα αποτελέσματα τα ομαδοποιούμε και ενεργοποιούμε το DecisionQuery1

```
//---------------------------------------------------------------------
void __fastcall TfrmTotalSalesCube::Button2Click(TObject *Sender)
{
  if(DecisionQuery1->Active)
    DecisionQuery1->Active = false;
    DecisionQuery1->SQL->Clear();
    DecisionQuery1->SQL->Add("SELECT ProductName, DYear, LastName,
RegionDescription, SUM( TotalSales ) ");
    DecisionQuery1->SQL->Add("FROM VW_SalesFromPart2 ");
    DecisionQuery1->SQL->Add("GROUP BY ProductName, DYear, LastName,
regionDescription ");
    DecisionQuery1->Active = true;
}
```

Κλείνουμε το DecisionQuery1 άμα αυτό είναι ενεργό, το καθαρίζουμε και επιλέγουμε τα πεδία που θέλουμε να εμφανιστούν από το ερώτημα VW_SalesFromPart2, τα αποτελέσματα

τα ομαδοποιούμε και ενεργοποιούμε το DecisionQuery1

```
//---------------------------------------------------------------------
void __fastcall TfrmTotalSalesCube::Button3Click(TObject *Sender)
{
  if (DecisionQuery1->Active)
    DecisionQuery1->Active = false;
    DecisionQuery1->SQL->Clear();
    DecisionQuery1->SQL->Add("SELECT ProductName, DYear, LastName,
RegionDescription, SUM( TotalSales ) ");
    DecisionQuery1->SQL->Add("FROM VW_SalesFromPart3 ");
    DecisionQuery1->SQL->Add("GROUP BY ProductName, DYear, LastName,
regionDescription ");
    DecisionQuery1->Active = true;
}
```

Κλείνουμε το DecisionQuery1 άμα αυτό είναι ενεργό, το καθαρίζουμε και επιλέγουμε τα πεδία που θέλουμε να εμφανιστούν από το ερώτημα VW_SalesFromPart3, τα αποτελέσματα τα ομαδοποιούμε και ενεργοποιούμε το DecisionQuery1

```
//---------------------------------------------------------------------
-
void __fastcall TfrmTotalSalesCube::Button4Click(TObject *Sender)
{
  if (DecisionQuery1->Active)
    DecisionQuery1->Active = false;
    DecisionQuery1->SQL->Clear();
    DecisionQuery1->SQL->Add("SELECT ProductName, DYear, LastName,
RegionDescription, SUM( TotalSales ) ");
    DecisionQuery1->SQL->Add("FROM VW_SalesFromPart4 ");
    DecisionQuery1->SQL->Add("GROUP BY ProductName, DYear, LastName,
regionDescription ");
    DecisionQuery1->Active = true;
}
```

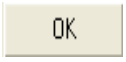
Κλείνουμε το DecisionQuery1 άμα αυτό είναι ενεργό, το καθαρίζουμε και επιλέγουμε τα πεδία που θέλουμε να εμφανιστούν από το ερώτημα VW_SalesFromPart4, τα αποτελέσματα τα ομαδοποιούμε και ενεργοποιούμε το DecisionQuery1

```
//---------------------------------------------------------------------
-void __fastcall TfrmTotalSalesCube::Button5Click(TObject *Sender)
{
  if (DecisionQuery1->Active)
    DecisionQuery1->Active = false;
    DecisionQuery1->SQL->Clear();
    DecisionQuery1->SQL->Add("SELECT ProductName, DYear, LastName,
RegionDescription, SUM( TotalSales ) ");
    DecisionQuery1->SQL->Add("FROM VW_AllSales ");
    DecisionQuery1->SQL->Add("GROUP BY ProductName, DYear, LastName,
regionDescription ");
    DecisionQuery1->Active = true;
}
```

Κλείνουμε το DecisionQuery1 άμα αυτό είναι ενεργό, το καθαρίζουμε και επιλέγουμε τα πεδία που θέλουμε να εμφανιστούν από το ερώτημα VW_AllSales, τα αποτελέσματα τα

ομαδοποιούμε και ενεργοποιούμε το DecisionQuery1

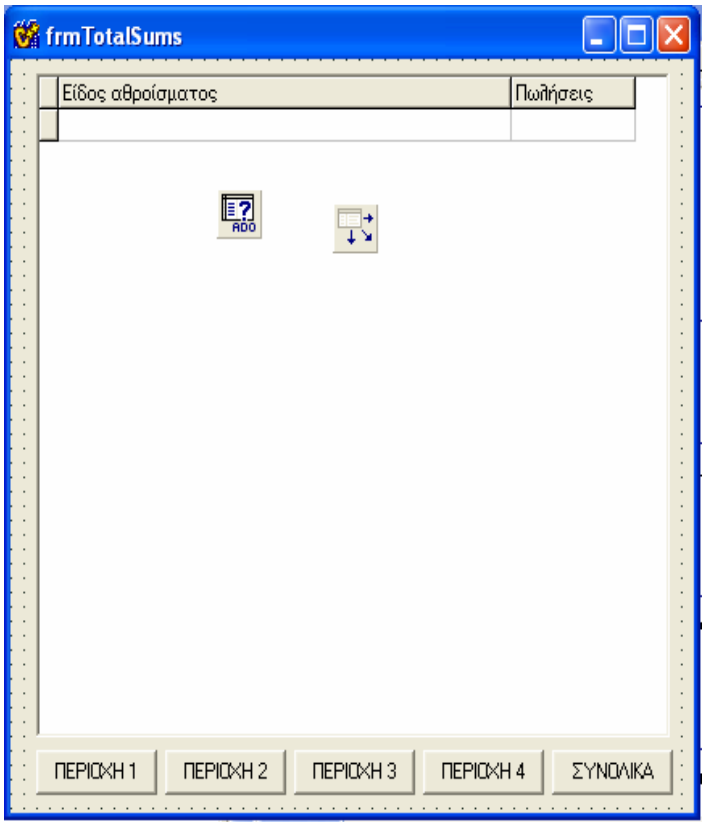
//----------

Αντικείμενα	Όνομα	Κλάση	Δεδομένα	Σχόλια
	**Page Control**	T PageControl	-	
	**Decision Query**	T Decision Query	Περιέχει τα αποτελέσματα που προέκυψαν από την εκτέλεση του ερωτήματος	
	**Decision Cube**	T DecisionCube	Πέρνει τα δεδομένα από το Decision Query και τα μεταφέρει στο Decision Source σε μορφή κύβου	
	**Decision Source**	T Decision Source	-	Διαχειρίζεται τις διαστάσεις του κύβου
	**DecisionGrid**	T DecisionGrid	Περιέχει τα αποτελέσματα του ερωτήματος	Εμφανίζει τα αποτελέσματα σε μορφή κύβου και είναι συνδεδεμένο με το Decision Source
	Button1	TButton	-	Με το κουμπί αυτό εμφανίζονται οι συγκεντρωτικές πωλήσεις από την πρώτη βάση
	Button2	TButton	-	Με το κουμπί αυτό εμφανίζονται οι συγκεντρωτικές πωλήσεις από την δεύτερη βάση
	**Button3**	TButton	-	Με το κουμπί αυτό εμφανίζονται οι συγκεντρωτικές πωλήσεις από την τρίτη βάση
	Button4	TButton	-	Με το κουμπί αυτό εμφανίζονται οι συγκεντρωτικές πωλήσεις από την τέταρτη βάση

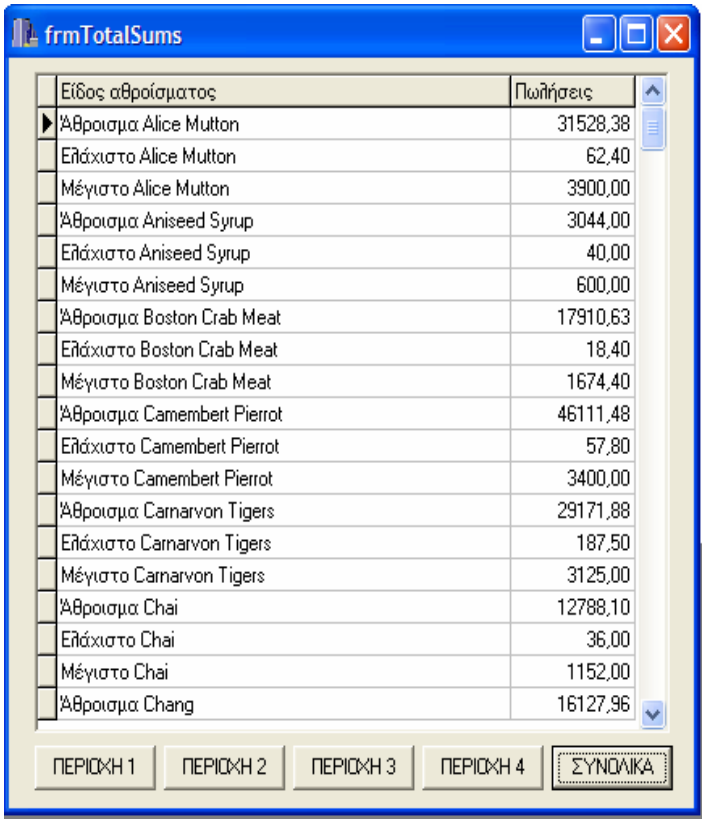
	Button5	TButton	-	Με το κουμπί αυτό εμφανίζονται οι συγκεντρωτικές πωλήσεις όλων των βάσεων

4.1.11 Φόρμα Αθροίσματα

Στη φόρμα αυτή μπορούμε να δούμε τις μέγιστες, τις ελάχιστες αλλά και τις συνολικές πωλήσεις ανά προϊόν. Για την δημιουργία της φόρμας χρησιμοποιούμε τα αντικείμενα :**DataSource, ADOQuery, DBGrid, Button.** Το ADOQuery χρησιμοποιείται για να δημιουργήσουμε το ερώτημα και να μας εμφανιστούν τα αποτελέσματα στο DBGrid. Το DBGrid είναι συνδεδεμένο με το DataSource και χρησιμοποιείται για να μας εμφανίζει τα αποτελέσματα της αναζήτησης. Το DataSource το χρησιμοποιούμε για την διαχείριση των εγγραφών που επιστρέφονται στο ADOQuery με την εκτέλεση του ερωτήματος.



Εικόνα. Η φόρμα σε χρόνο σχεδίασης (Disign Time)



Εικόνα. Η φόρμα σε χρόνο εκτέλεσης (Run Time)

```
//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "frmSums.h"
#include "MainForm.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmTotalSums *frmTotalSums;
//---------------------------------------------------------------------
void __fastcall TfrmTotalSums::FormClose(TObject *Sender,
TCloseAction &Action)
{
  ((TForm *)this->Owner)->Show();
}
```

Εμφανίζουμε την φόρμα

```
//---------------------------------------------------------------------
void __fastcall TfrmTotalSums::Button1Click(TObject *Sender)
{
  if (ADOQuery1->Active)
    ADOQuery1->Active = false;
    ADOQuery1->SQL->Clear();
    ADOQuery1->SQL->Add("Select 'Μέγιστο ' + ProductName as Descr,
ProductName, Max(TotalSales) as Sales from VW_SalesFromPart1 ");
```

```
    ADOQuery1->SQL->Add("group By ProductName ");
    ADOQuery1->SQL->Add("union ");
    ADOQuery1->SQL->Add("Select 'Ελάχιστο '+  ProductName,
ProductName, Min(TotalSales) from VW_SalesFromPart1 ");
    ADOQuery1->SQL->Add("group By ProductName ");
    ADOQuery1->SQL->Add("union ");
    ADOQuery1->SQL->Add("Select 'Άθροισμα ' + ProductName,
ProductName, sum(TotalSales) from VW_SalesFromPart1 ");
    ADOQuery1->SQL->Add("group By ProductName ");
    ADOQuery1->SQL->Add("order by ProductName ");
    ADOQuery1->Active = true;
}
```

Κλείνουμε το ADOQuery1 αν αυτό είναι ενεργό, το καθαρίζουμε, του φορτώνουμε το ερώτημα όπου θέλουμε να εμφανίζει στη μια στήλη το Μέγιστο, το Ελάχιστο και το Άθροισμα, μαζί με το Όνομα του κάθε προϊόντος και στην άλλη στήλη τα αποτελέσματα, τα ομαδοποιούμε και τα ταξινομούμε κατά το ProductName, Τα αποτελέσματα τα παίρνουμε μέσα από το ερώτημα VW_SalesFromPart1 και ενεργοποιουμε το ADOQuery1

```
//---------------------------------------------------------------------
void __fastcall TfrmTotalSums::Button2Click(TObject *Sender)
{
  if (ADOQuery1->Active)
    ADOQuery1->Active = false;
    ADOQuery1->SQL->Clear();
    ADOQuery1->SQL->Add("Select 'Μέγιστο  ' + ProductName as Descr,
ProductName, Max(TotalSales) as Sales from VW_SalesFromPart2 ");
    ADOQuery1->SQL->Add("group By ProductName ");
    ADOQuery1->SQL->Add("union ");
    ADOQuery1->SQL->Add("Select 'Ελάχιστο '+  ProductName,
ProductName, Min(TotalSales) from VW_SalesFromPart2 ");
    ADOQuery1->SQL->Add("group By ProductName ");
    ADOQuery1->SQL->Add("union ");
    ADOQuery1->SQL->Add("Select 'Άθροισμα ' + ProductName,
ProductName, sum(TotalSales) from VW_SalesFromPart2 ");
    ADOQuery1->SQL->Add("group By ProductName ");
    ADOQuery1->SQL->Add("order by ProductName ");
    ADOQuery1->Active = true;
}
```

Κλείνουμε το ADOQuery1 αν αυτό είναι ενεργό, το καθαρίζουμε, του φορτώνουμε το ερώτημα όπου θέλουμε να εμφανίζει στη μια στήλη το Μέγιστο, το Ελάχιστο και το Άθροισμα, μαζί με το Όνομα του κάθε προϊόντος και στην άλλη στήλη τα αποτελέσματα, τα ομαδοποιούμε και τα ταξινομούμε κατά το ProductName, Τα αποτελέσματα τα παίρνουμε μέσα από το ερώτημα VW_SalesFromPart2, και ενεργοποιουμε το ADOQuery1

```
//---------------------------------------------------------------------
void __fastcall TfrmTotalSums::Button3Click(TObject *Sender)
{
  if (ADOQuery1->Active)
    ADOQuery1->Active = false;
```

```
    ADOQuery1->SQL->Clear();
    ADOQuery1->SQL->Add("Select 'Μέγιστο ' + ProductName as Descr,
ProductName, Max(TotalSales) as Sales from VW_SalesFromPart3 ");
    ADOQuery1->SQL->Add("group By ProductName ");
    ADOQuery1->SQL->Add("union ");
    ADOQuery1->SQL->Add("Select 'Ελάχιστο '+  ProductName,
ProductName, Min(TotalSales) from VW_SalesFromPart3 ");
    ADOQuery1->SQL->Add("group By ProductName ");
    ADOQuery1->SQL->Add("union ");
    ADOQuery1->SQL->Add("Select 'Άθροισμα ' + ProductName,
ProductName, sum(TotalSales) from VW_SalesFromPart3 ");
    ADOQuery1->SQL->Add("group By ProductName ");
    ADOQuery1->SQL->Add("order by ProductName ");
    ADOQuery1->Active = true;
}
```

Κλείνουμε το ADOQuery1 αν αυτό είναι ενεργό, το καθαρίζουμε, του φορτώνουμε το ερώτημα όπου θέλουμε να εμφανίζει στη μια στήλη το Μέγιστο, το Ελάχιστο και το Άθροισμα, μαζί με το Όνομα του κάθε προϊόντος και στην άλλη στήλη τα αποτελέσματα, τα ομαδοποιούμε και τα ταξινομούμε κατά το ProductName, τα αποτελέσματα τα παίρνουμε μέσα από το ερώτημα VW_SalesFromPart3, και ενεργοποιουμε το ADOQuery1

```
//---------------------------------------------------------------------
void __fastcall TfrmTotalSums::Button4Click(TObject *Sender)
{
  if (ADOQuery1->Active)
    ADOQuery1->Active = false;
    ADOQuery1->SQL->Clear();
    ADOQuery1->SQL->Add("Select 'Μέγιστο ' + ProductName as Descr,
ProductName, Max(TotalSales) as Sales from VW_SalesFromPart4 ");
    ADOQuery1->SQL->Add("group By ProductName ");
    ADOQuery1->SQL->Add("union ");
    ADOQuery1->SQL->Add("Select 'Ελάχιστο '+  ProductName,
ProductName, Min(TotalSales) from VW_SalesFromPart4 ");
    ADOQuery1->SQL->Add("group By ProductName ");
    ADOQuery1->SQL->Add("union ");
    ADOQuery1->SQL->Add("Select 'Άθροισμα ' + ProductName,
ProductName, sum(TotalSales) from VW_SalesFromPart4 ");
    ADOQuery1->SQL->Add("group By ProductName ");
    ADOQuery1->SQL->Add("order by ProductName ");
    ADOQuery1->Active = true;
}
```

Κλείνουμε το ADOQuery1 αν αυτό είναι ενεργό, το καθαρίζουμε, του φορτώνουμε το ερώτημα όπου θέλουμε να μας εμφανίζει στη μια στήλη το Μέγιστο, το Ελάχιστο και το Άθροισμα, μαζί με το Όνομα του κάθε προϊόντος και στην άλλη στήλη τα αποτελέσματα, τα ομαδοποιούμε και τα ταξινομούμε κατά το ProductName, Τα αποτελέσματα τα παίρνουμε μέσα από το ερώτημα VW_SalesFromPart4, ενεργοποιουμε το ADOQuery1

```
//---------------------------------------------------------------------
void __fastcall TfrmTotalSums::Button5Click(TObject *Sender)
{
  if (ADOQuery1->Active)
    ADOQuery1->Active = false;
    ADOQuery1->SQL->Clear();
    ADOQuery1->SQL->Add("Select 'Μέγιστο ' + ProductName as Descr,
ProductName, Max(TotalSales) as Sales from VW_AllSales ");
    ADOQuery1->SQL->Add("group By ProductName ");
    ADOQuery1->SQL->Add("union ");
    ADOQuery1->SQL->Add("Select 'Ελάχιστο '+  ProductName,
ProductName, Min(TotalSales) from VW_AllSales ");
    ADOQuery1->SQL->Add("group By ProductName ");
    ADOQuery1->SQL->Add("union ");
    ADOQuery1->SQL->Add("Select 'Άθροισμα ' + ProductName,
ProductName, sum(TotalSales) from VW_AllSales ");
    ADOQuery1->SQL->Add("group By ProductName ");
    ADOQuery1->SQL->Add("order by ProductName ");
    ADOQuery1->Active = true;
}
```

Κλείνουμε το ADOQuery1 αν αυτό είναι ενεργό, το καθαρίζουμε, του φορτώνουμε το ερώτημα όπου θέλουμε να εμφανίζει στη μια στήλη το Μέγιστο, το Ελάχιστο και το Άθροισμα, μαζί με το Όνομα του κάθε προϊόντος και στην άλλη στήλη τα αποτελέσματα, τα ομαδοποιούμε και τα ταξινομούμε κατά το ProductName, Τα αποτελέσματα τα παίρνουμε μέσα από το ερώτημα VW_AllSales, ενεργοποιούμε το ADOQuery1

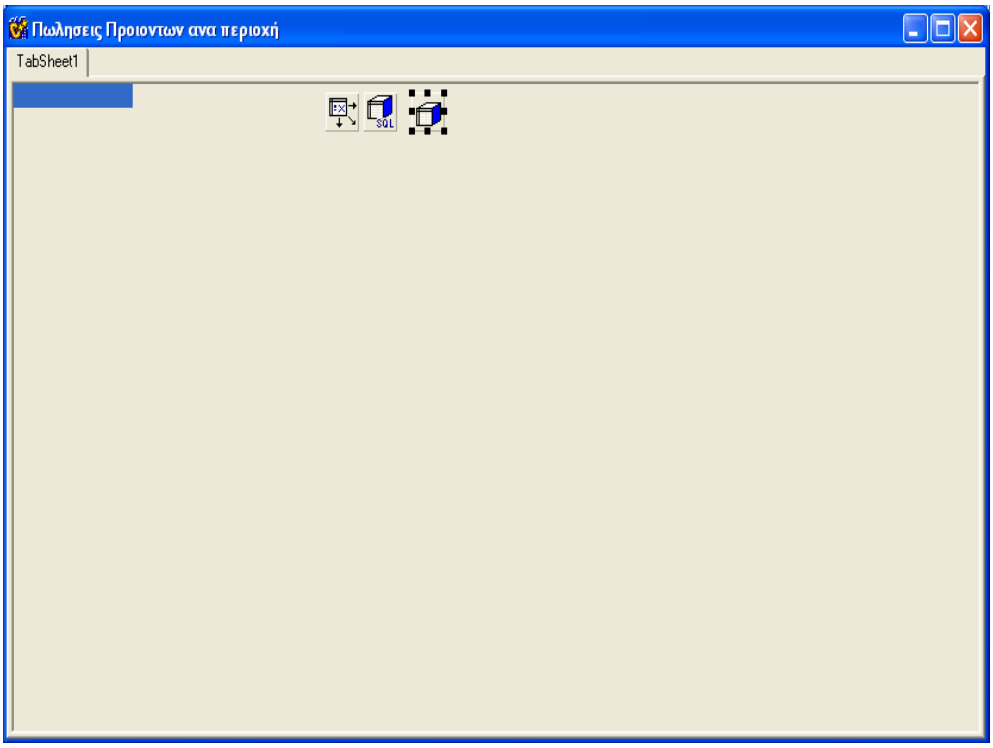
```
//---------------------------------------------------------------------
```

Αντικείμενα	**Όνομα**	**Κλάση**	**Δεδομένα**	**Σχόλια**
	**DataSource1**	TDataSource	-	Διαχείριση των εγγραφών που επιστρέφονται από το ερώτημα VW_AllSales μέσω του ADOQuery1
	**ADOQuery1**	TADOQuery	Περιέχει το αποτέλεσμα που προέκυψε από την εκτέλεση του ερωτήματος	Μέσω του ερωτήματος παίρνουμε τα δεδομένα που ζητάμε το ερώτημα VW_AllSales
	**DBGrid1**	TDBGrid	Περιέχει τα αποτελέσματα της αναζήτησης	Εμφανίζει τα αποτελέσματα του ερωτήματος

OK	**Button1**	TButton	-	Με το κουμπί αυτό εμφανιζουμε τα συγκεντρωτικά αποτελέσματα από την πρώτη βάση
	Button2	TButton	-	Με το κουμπί αυτό εμφανιζουμε τα συγκεντρωτικά αποτελέσματα από την δεύτερη βάση
	Button3	TButton	-	Με το κουμπί αυτό εμφανιζουμε τα συγκεντρωτικά αποτελέσματα από την τρίτη βάση
	Button4	TButton	-	Με το κουμπί αυτό εμφανιζουμε τα συγκεντρωτικά αποτελέσματα από την τέταρτη βάση
	Button5	TButton	-	Με το κουμπί αυτό εμφανιζουμε τα συγκεντρωτικά αποτελέσματα όλων των βάσεων

4.1.12 Φόρμα Πωλήσεις Προϊόντων ανά Περιοχή

Στην φόρμα αυτή μπορούμε να δούμε τις πωλήσεις προϊόντων ανά περιοχή υπό μορφή κύβου .Ποιο συγκεκριμένα θα δούμε τις πωλήσεις κάθε προϊόντος για κάθε βάση καθώς και το Άθροισμα τους για κάθε χρόνο αλλά και τα συνολικά Αθροίσματα. Για τη δημιουργία της φόρμας χρησιμοποιούμε τα αντικείμενα: **PageControl,DecisionGrid, DecisionSource, DecisionCube, Decision Query, Edit.** Το Decision Query Περιέχει το ερώτημα που θα εκτελεστή και είναι συνδεδεμένο με το DecisionCube, Το DecisionCube παίρνει τα αποτελέσματα από την εκτέλεση του ερωτήματος και τα μεταφέρει στο DecisionSource σε μορφή κύβου . Το DecisionGrid είναι συνδεδεμένο με το DecisionSource και χρησιμοποιείται για να μας εμφανίζει τα αποτελέσματα της αναζήτησης το οποίο το έχουμε τοποθετήσει πάνω σε ένα Page Control . Το Decision Source το χρησιμοποιούμε για να μπορούμε να αλάζουμε τις διαστάσεις του κύβου και να βλέπουμε τα αποτελέσματα από διαφορετική οπτική γωνία κάθε φορά.



Εικόνα. Η φόρμα σε χρόνο σχεδίασης (Disign Time)

Πωλησεις Προιοντων ανα περιοχή

TabSheet1

			ProductName				
DYear	DQuarter	DMonth	Alice Mutton	Aniseed Syrup	Boston Crab Meat	Camembert Pierrot	Carnarvon Tig
1996	3	7	936		735	1088	
		8	819	240	1308,3	1931,2	600
		9	1248		676,2		1125
		Sum	3003	240	2719,5	3019,2	1725
	4	10	2948,4				2550
		11	1010,88		58,8	3587,68	
		12				2418,08	450
		Sum	3959,28		58,8	6005,76	3000
	Sum		6962,28	240	2778,3	9024,96	4725
1997	1	1	2355,6	400	577,71	1550,4	
		2	312		294		
		3		144	896,7	1632	1500
		Sum	2667,6	544	1768,41	3182,4	1500
	2	4			460	3097,4	1406,25
		5	2718,3	600	561,2	892,5	
		6	1294,8		956,8	693,6	956,25
		Sum	4013,1	600	1978	4683,5	2362,5
	3	7	3900	140	772,8	4131	2000
		8			1104	1020	5100
		9	936		2535,52	4428,5	
		Sum	4836	140	4412,32	9579,5	7100
	4	10	1866,15	60		340	562,5
		11	741	200	1637,6	2210	2550
		12	3480,75	180	18,4	510	1875

Εικόνα. Η φόρμα σε χρόνο εκτέλεσης (Run Time)

```
//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
```

```
#include "DaysCubeForm.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmDaysCube *frmDaysCube;
//---------------------------------------------------------------------
void __fastcall TfrmDaysCube::FormClose(TObject *Sender, TCloseAction
&Action)
{
  if (DecisionQuery1->Active)
    DecisionQuery1->Active = false;
    ((TForm *)this->Owner)->Show();
}
```

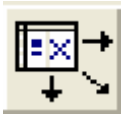
Απενεργοποιούμε το DecisionQuery1 αν αυτό είναι ενεργό και εμφανιζουμε την κεντρική φόρμα

```
//---------------------------------------------------------------------
void __fastcall TfrmDaysCube::FormShow(TObject *Sender)
{
  if (DecisionQuery1->Active)
    DecisionQuery1->Active = false;
    DecisionQuery1->SQL->Clear();
    DecisionQuery1->SQL->Add("SELECT ProductName, DYear, DQuarter,
DMonth, SUM( TotalSales ) ");
    DecisionQuery1->SQL->Add("FROM VW_SalesByDayAll ");
    DecisionQuery1->SQL->Add("GROUP BY ProductName, DYear, DQuarter,
DMonth ");
    DecisionQuery1->Active = true;
}
```

Απενεργοποιούμε το DecisionQuery1 αν αυτό είναι ενεργό, το καθαρίζουμε και του φορτώνουμε το ερώτημα όπου επιλέγουμε τα πεδία που θέλουμε να εμφανίσουμε, τα παίρνουμε από το ερώτημα VW_SalesByDayAll, τα ομαδοποιούμε και τρέχουμε το DecisionQuery1

```
//---------------------------------------------------------------------
```

Αντικείμενα	**Όνομα**	**Κλάση**	**Δεδομένα**	**Σχόλια**
	**PageControl**	TPageControl	-	
	**Decision Query**	TDecision Query	Περιέχει τα αποτελέσματα που προέκυψαν από την εκτέλεση του ερωτήματος	

	**DecisionCube**	TDecisionCube	Παίρνει τα δεδομένα από το Decision Query και τα μεταφέρει στο Decision Source σε μορφή κύβου	
	**Decision Source**	TDecision Source	-	Διαχειρίζεται τις διαστάσεις του κύβου
	**DecisionGrid**	TDecisionGrid	Περιέχει τα αποτελέσματα του ερωτήματος	Εμφανίζει τα αποτελέσματα σε μορφή κύβου και είναι συνδεδεμένο με το Decision Source

ΠΑΡΑΡΤΗΜΑ

Παραδείγματα δημιουργίας Linked Server

Στην παράγραφο αυτή θα δούμε μερικά παραδείγματα δημιουργίας Linked Server για διαφορές Βάσεις Δεδομένων

Χρήση του OLE DB Provider για Jet

Στο παράδειγμα αυτό δημιουργούμε ένα linked server με όνομα SEATTLE Mktg για να τον συνδέσουμε με μια βάση με όνομα Northwind η οποία βρίσκεται στην Access στην θέση C:\Msoffice\Access\Samples

Με την χρήση παραμέτρων:

```
USE master
GO
EXEC
   @server = 'SEATTLE Mktg',
   @provider = 'Microsoft.Jet.OLEDB.4.0',
   @srvproduct = 'OLE DB Provider for Jet',
   @datasrc = 'C:\MSOffice\Access\Samples\Northwind.mdb'
GO
```

Ή χωρίς την χρήση παραμέτρων:

```
USE master
GO
EXEC sp_addlinkedserver
   'SEATTLE Mktg',
   'OLE DB Provider for Jet',
   'Microsoft.Jet.OLEDB.4.0',
   'C:\MSOffice\Access\Samples\Northwind.mdb'
GO
```

Χρήση του OLE DB Provider για την Oracle

Στο παράδειγμα αυτό δημιουργούμε έναν linked server με όνομα LONDON Mktg με την χρήση του OLE DB provider για την Oracle και το όνομα του Data Sources (ODBC) είναι MyServer.

Με την χρήση παραμέτρων:

```
USE master
GO
EXEC sp_addlinkedserver
   @server = 'LONDON Mktg',
   @srvproduct = 'Oracle',
   @provider = 'MSDAORA',
   @datasrc = 'MyServer'
GO
```

Ή χωρίς την χρήση παραμέτρων:

```
USE master
GO
EXEC sp_addlinkedserver
   'LONDON Mktg',
   'Oracle',
   'MSDAORA',
   'MyServer'
GO
```

Χρήση του OLE DB Provider για τον ODBC με τις παραμέτρους του data_source (ODBC)

Στο παράδειγμα αυτό δημιουργούμε έναν linked server με όνομα SEATTLE Payroll που χρησιμοποιεί OLE DB Provider για ODBC και τις παραμέτρους του data_source.

Με την χρήση παραμέτρων:

```
USE master
GO
EXEC sp_addlinkedserver
   @server = 'SEATTLE Payroll',
   @provider = 'MSDASQL',
   @datasrc = 'LocalServer'
GO
```

Ή χωρίς την χρήση παραμέτρων:

```
USE master
GO
EXEC sp_addlinkedserver
   'SEATTLE Payroll',
   '',
   'MSDASQL',
   'LocalServer'
GO
```

Χρήση του OLE DB Provider για τον ODBC με τις παραμέτρους του provider_string

Στο παράδειγμα αυτό δημιουργούμε έναν linked server με όνομα LONDON Payroll που χρησιμοποιεί OLE DB Provider για ODBC και τις παραμέτρους του provider_string.

Με την χρήση παραμέτρων:

```
USE master
GO
```

```
EXEC sp_addlinkedserver
   @server = 'LONDON Payroll',
   @provider = 'MSDASQL',
   @provstr = 'DRIVER={SQL Server};SERVER=MyServer;UID=sa;PWD=;'
GO
```

Ή χωρίς την χρήση παραμέτρων:

```
USE master
GO
EXEC sp_addlinkedserver
   'LONDON Payroll',
   '',
   'MSDASQL',
   NULL,
   NULL,
   'DRIVER={SQL Server};SERVER=MyServer;UID=sa;PWD=;'
GO
```

OLE DB data source	OLE DB provider	Product_ name	Provider_ name	Data_source	Provider_string	catalog
SQL Server	Microsoft OLE DB Provider for SQL Server	SQL Server (1)-(default)	-	-	-	-
SQL Server	Microsoft OLE DB Provider for SQL Server	SQL Server	SQLOLEDB	Το όνομα του SQL Server (for default instance)	-	Το όνομα της βάσης
SQL Server	Microsoft OLE DB Provider for SQL Server	-	SQLOLEDB	Servername\instancename (for specific instance)	-	Το όνομα της βάσης
Oracle	Microsoft OLE DB Provider for Oracle	Any (2)	MSDAORA	SQL *Net alias για Oracle database	-	-
Access/Jet	Microsoft	Any	Microsoft.Jet.	Το πλήρες	-	-

	OLE DB Provider for Jet		OLEDB.4.0	όνομα του μονοπατιού που βρίσκεται αποθηκευμένο το αρχείο		
ODBC data source	Microsoft OLE DB Provider for ODBC	Any	MSDASQL	System DSN of ODBC Source	-	-
ODBC data source	Microsoft OLE DB Provider for ODBC	Any	MSDASQL	-	ODBC connection string	-
File system	Microsoft OLE DB Provider for Indexing Server	Any	MSIDXS	Indexing Service catalog name (Να δείχνει στο όνομα του καταλόγου του Server)	-	-
Microsoft Excel Spreadsheet	Microsoft OLE DB Provider for Jet	Any	Microsoft.Jet. OLEDB.4.0	Το πλήρες όνομα του μονοπατιού που βρίσκεται αποθηκευμένο το αρχείο Excel	Excel 5.0	-
IBM DB2 Database	Microsoft OLE DB Provider for DB2	Any	DB2OLEDB	-	Microsoft OLE DB Provider for DB2	Το όνομα του καταλόγου για την βάση DB2

ΠΙΝΑΚΑΣ: Συγκεντρωτικός πίνακας δημιουργίας Linked Server

ΒΙΒΛΙΟΓΡΑΦΙΑ

Silberscatz, Korth and Sudarshan. "Συστήματα Βάσεων Δεδομένων, 4η Έκδοση", Εκδόσεις Μ. Γκιούρδας 2002, Κεφάλαια 2, 3, 19, 25, 27,

M.H. Dunham, "Data Mining introductory and advanced topics" Prentice Hall 2004.

A. Silberschatz, H. F. Korth and S. Sudrshan, "Database System Concepts" 4th ed., McGraw Hill, 2002.

D. Petkovic, "Οδηγός του SQL Server 2000", Εκδόσεις Μ. Γκιούρδας.

J. Han and M. Kamber, "Data Mining: Concepts and Techniques"

D. Agrawal, A. E. Abbadi, A. Singh, and T. Yurek. Efficient view maintenance in data warehouses. SIGMOD'97.

K. Beyer and R. Ramakrishnan. Bottom-Up Computation of Sparse and Iceberg CUBEs.. SIGMOD'99.

S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. ACM SIGMOD Record, 26:65-74, 1997.

OLAP council. MDAPI specification version 2.0. In http://www.olapcouncil.org/research/apily.htm, 1998.

Microsoft. OLEDB for OLAP programmer's reference version 1.0. In http://www.microsoft.com/data/oledb/olap, 1998.

R. Elmasri and S.B Navathe. "Θεμελιώδεις αρχές συστημάτων Βάσεων Δεδομένων, Τόμος Α', 3η Έκδοση Αναθεωρημένη". Εκδόσεις Δίαυλος

C.J. Date. "Εισαγωγή στα συστήματα Βάσεων Δεδομένων, Τόμος Α' ". Εκδόσεις Κλειδάριθμος

Jorrod Hollingworth, Bob Swart, Mark Cashman and Paul Gustavson. "Borland C++ Builder 6". Εκδόσεις Μ. Γκιούρδας