ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Διαδικασίες DataWarehousing
Από την Θεωρία στην Πρακτική Εφαρμογή

Άννα Λόκτεβα (Α.Μ. 35803)

Τμήμα ΗΥΣ

ΤΕΙ Πειραιά

Επιβλέπων: Γ. Ν. Πρεζεράκος

Περιεχόμενα

Περίληψη

Ο σκοπός της διπλωματικής εργασίας είναι η περιγραφή ενός Data Warehouse (DW, Αποθήκης Δεδομένων) και των διαδικασιών της. Η διπλωματική εργασία χωρίζεται σε δύο μέρη, ένα θεωρητικό και ένα πρακτικό. Με λίγα λόγια, στο θεωρητικό μέρος αναλύεται τι είναι ένα DW, ποια η λειτουργία του και από ποια δομικά στοιχεία αποτελείται. Στη συνέχεια ακολουθεί το πρακτικό μέρος όπου έχει γίνει υλοποίηση ενός έργου DW, που δείχνει πως με την βοήθεια ενός «Extract, Transform, and Load» (ETL)εργαλείου τα δεδομένα μεταφέρονται από την πηγή στο τελικό προορισμό, το DW. Και τέλος αναλύονται τα πλεονεκτήματα και μειονεκτήματα ενός ETL εργαλείου σε σχέση με custom ETL διαδικασίες που είναι γραμμένες εξ' ολοκλήρου με κώδικα.

Η πτυχιακή εξετάζει την περίπτωση χρήσης μιας εταιρίας μαρμάρων που ονομάζεται «Κρόνος». Είναι μια από της μεγαλύτερες εταιρίες παραγωγής και εξαγωγής μαρμάρων στην Ελλάδα, και έχει σκοπό την περαιτέρω ανάπτυξη της. Η εταιρία «Κρόνος» αποτελείται από τρία κυρίως εργοστάσια που ασχολούνται με παραγωγή Μαρμάρων, παραγωγή Γρανιτών και Αδρανών Υλικών. Μάλιστα το εργοστάσιο Γρανιτών κάνει εισαγωγές από και εξαγωγές στο εξωτερικό. Επίσης υπάρχουν και διάφορα υποκαταστήματα που πουλάνε τα προϊόντα της και επεκτείνουν την αγορά της. Όπως γίνεται αντιληπτό η εταιρία είναι αρκετά μεγάλη και τα δεδομένα της έρχονται από τελείως διαφορετικές πηγές (τα διαφορετικά καταστήματα και εργοστάσια).

Για να μπορεί η εταιρία να διαχειρίζεται εύκολα και παραγωγικά τα δεδομένα από τα υποκαταστήματα και τα εργοστάσια ήταν απαραίτητο να δημιουργηθεί ένα DW, μια ενιαία αποθήκη όπου θα μαζεύονται όλα τα δεδομένα που σχετίζονται με την εταιρία. Για το λόγο αυτό έχει φτιαχτεί μια ETL διαδικασία η οποία μαζεύει δεδομένα από διαφορετικές πηγές, τα τροποποιεί και τα αποθηκεύει σε μια και μοναδική βάση (DW).

Τα δεδομένα προέρχονται από το παραγωγικό σύστημα της «Κρόνος», τον παραγωγικό Server (AIX Server) όπου τα δεδομένα βρίσκονται σε αρχεία Cobol. Το γεγονός ότι τα δεδομένα βρίσκονται σε Cobol αρχεία τα καθιστά άκρως δύσχρηστα για τον τελικό χρήστη. Το κύριο μειονέκτημα των cobol αρχείων είναι ότι για οποιαδήποτε αναφορά (report) χρειάζεται να γραφτεί ένα καινούριο cobol πρόγραμμα και να τροποποιείται κάθε φορά που χρειάζονται αλλαγές.

Για να λυθεί το πρόβλημα με τη δημιουργία και συντήρηση αναφορών, η εταιρία αποφάσισε να χρησιμοποιήσει ένα Business Intelligence (BI, Επιχειρησιακής Ευφυΐας) εργαλείο. Τα BI εργαλεία δίνουν την δυνατότητα στον τελικό χρήστη να δημιουργεί και να συντηρεί μόνος του τις αναφορές του. Ωστόσο τα BI εργαλεία δεν μπορούν να διαχειρίζονται cobol αρχεία.

Έτσι ένας από τους κύριους λόγους που έπρεπε να χρησιμοποιηθεί μια ETL διαδικασία είναι η μετατροπή των Cobol αρχείων σε μορφή που να είναι κατανοητή και επεξεργάσιμη από το BI εργαλείο. Τελικά δημιουργήθηκε μία Αποθήκη όπου τα δεδομένα είναι δομημένα και τροποποιημένα με τρόπο που να είναι εύχρηστα για τον τελικό χρήστη και συγχρόνως να καταγράφεται ιστορικότητα των κινήσεων της εταιρίας.

Για την δημιουργία μιας Αποθήκης Δεδομένων είναι απαραίτητο να χρησιμοποιηθούν οι λεγόμενες ETL διαδικασίες. Με τον όρο ETL, αναφερόμαστε στην διαδικασία που έχει σαν σκοπό την αντιγραφή, το μετασχηματισμό και την φόρτωση των δεδομένων του παραγωγικού συστήματος στην Αποθήκη, που στην προκειμένη περίπτωση βρίσκεται στον IQ Adaptive Server (IQ).

Η όλη διαδικασία εξαγωγής-μετασχηματισμού-φόρτωσης δεδομένων έχει πραγματοποιηθεί με ένα ETL εργαλείο, το PowerCenter της Informatica.

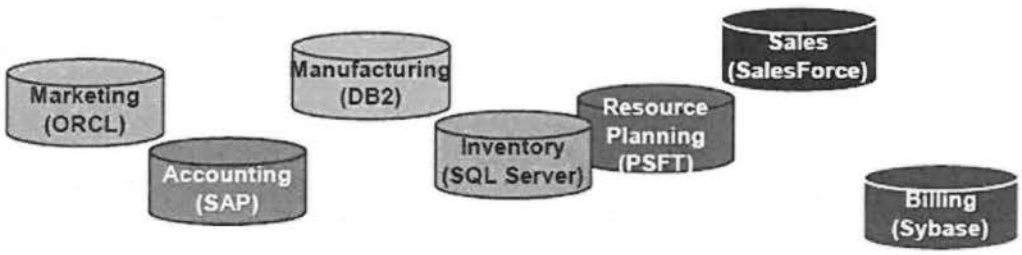
Θεωρητικό Μέρος

1. Εισαγωγή

Ξεκινώντας θα ήθελα να μιλήσω αναφορικά για κάποιες έννοιες του Data Warehousing και να αναφερθώ στους λόγους χρήσεις DW γενικά.

Είναι γνωστό ότι οι μεγάλες εταιρίες και οργανισμοί συγκεντρώνουν τεράστιες ποσότητες δεδομένων και πληροφοριών από τα παραγωγικά τους συστήματα. Τα δεδομένα αυτά μπορεί να αποθηκεύονται σε πολλές και διαφορετικές μορφές (π.χ. flatfiles, txt, excel, cobol αρχεία...), σε αδόμητα αρχεία, ακόμα και σε διαφορετικές βάσης δεδομένων.

Το γεγονός αυτό τα καθιστά δύσχρηστα σε περίπτωση που η εταιρία θελήσει να τα ενοποιήσει και να τα χρησιμοποιήσει όλα μαζί για κάποιο δικό της επιχειρησιακό σκοπό. Επίσης, πρόβλημα παρουσιάζεται όταν δύο εταιρίες αποφασίσουν να συνεργαστούν (ή μια αγοράσει την άλλη) και κριθεί απαραίτητο να συγχωνευτούν τα δεδομένα τους. Έτσι προκύπτει η ανάγκη δύο ή παραπάνω βάσεων να επικοινωνήσουν μεταξύ τους. Σε αυτό το σημείο αναδύεται ένα τεράστιο πρόβλημα, το γεγονός ότι, οι διαφορετικές βάσεις δεν μπορούν να επικοινωνήσουν μεταξύ τους, γιατί απλούστατα δεν μιλάνε όλες την ίδια γλώσσα.



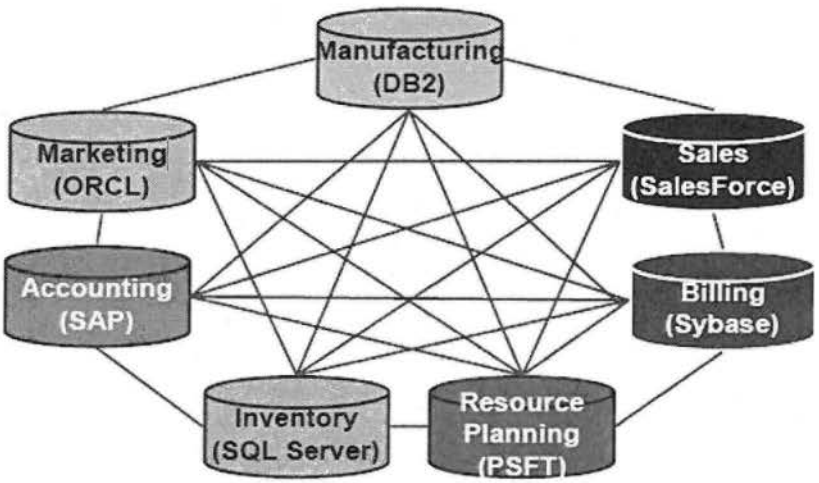
Πηγή: Informatica User Manual

Έστω ότι μια εταιρία βρήκε μια λύση για το παραπάνω πρόβλημα και δημιούργησε διασυνδέσεις μεταξύ των διαφορετικών βάσεων. Στο σημείο αυτό προκύπτει ένα καινούριο πρόβλημα, το πρόβλημα των πολλαπλών διασυνδέσεων.

Συνήθως μια διασύνδεση είναι ένα λογισμικό, πρόγραμμα ειδικά φτιαγμένο για να συνδέει δυο βάσεις. Ένα λογισμικό που μετατρέπει τα δεδομένα της μίας βάσης σε μορφή (format) που θα είναι συμβατά με μια άλλη ώστε να γίνει δυνατή η επικοινωνία και συνεπώς η ανταλλαγή δεδομένων.

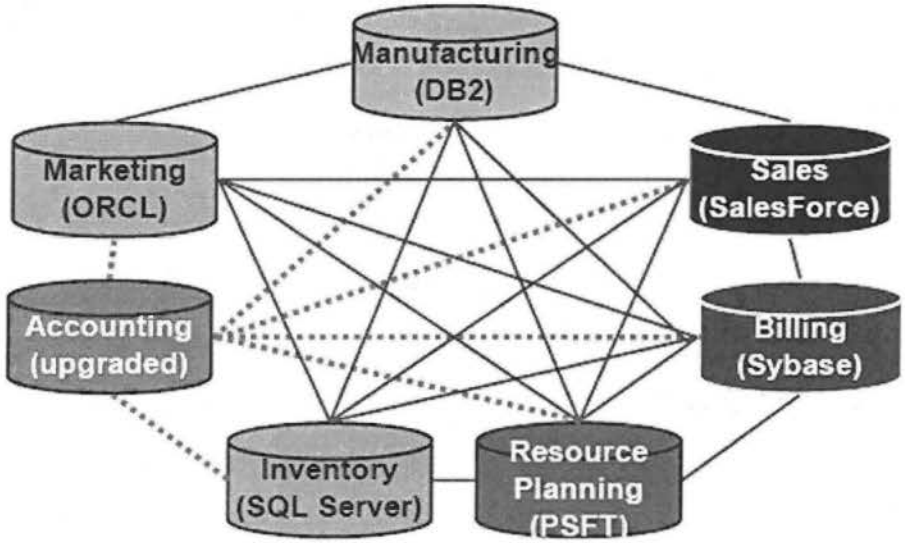
Ας φανταστούμε τώρα, ότι μια εταιρία έχει π.χ. επτά τέτοιες βάσεις οπότε οι διασυνδέσεις που πρέπει να γίνουν είναι 17 (για n βάσεις θέλουμε 1+2+...+(n-1)).

Στο σημείο αυτό τα πράγματα αρχίζουν να γίνονται πολύ περίπλοκα. Παραδείγματος χάρη για να ενώσουμε δεδομένα από marketing, accounting και inventory θα πρέπει να τρέξουν τρία διαφορετικά λογισμικά, ύστερα να τροποποιηθούν τα δεδομένα σε μια ενιαία μορφή και μόνο και μόνο τότε θα είναι έτοιμα για χρήση.



Πηγή: Informatica User Manual

Και σαν να μην έφταναν όλα αυτά, το πρόβλημα των πολλαπλών διασυνδέσεων αναβλύζει ένα άλλο πρόβλημα που προκύπτει αν σε περίπτωση που οι βάσεις για κάποιο λόγο αλλάξουν ή αναβαθμιστούν. Τότε πολλές διασυνδέσεις αυτομάτως θα βγουν εκτός λειτουργίας και θα πρέπει να αναδιαμορφωθούν αναλόγως. Με άλλα λόγια μια εταιρία που έχει πολλές βάσεις και πολλές διασυνδέσεις είτε θα πρέπει να μείνει για πάντα στην παλαιολιθική εποχή είτε να χρυσοπληρώνει συνέχεια για αναβαθμίσεις. Όπου και οι δύο περιπτώσεις είναι ασύμφορες.



Πηγή: Informatica User Manual

Για να αποφευχθούνε όλα τα παραπάνω προβλήματα και να γίνει η διαχείριση πιο εύκολη, η λύση βρίσκεται στις τεχνολογίες Ενοποίησης Δεδομένων (Data Integration). Με άλλα λόγια ένας τρόπος που θα επιτρέπει σε εφαρμογές που χρησιμοποιούνε διαφορετικές βάσεις και διαφορετικά formats να επικοινωνούνε μεταξύ τους μόνο με μια διασύνδεση η καθεμία. Μια τέτοια λύση γίνεται αυτομάτως σωτήρια για πολλές εταιρίες που αντιμετωπίζουν τα πιο πάνω προβλήματα.

Πηγή: Informatica User Manual

Πιο αναλυτικά το Data Integration είναι ο συνδυασμός των τεχνικών και των επιχειρηματικών διαδικασιών που χρησιμοποιούνται για το ενοποίηση δεδομένων από διαφορετικές πηγές με σκοπό την παροχή στους χρήστες μια ενοποιημένη προβολή των στοιχείων αυτών. Υπάρχουν πολλές τεχνολογίες/μεθοδολογίες για Ενοποίηση δεδομένων. Στην εργασία μου θα ασχοληθώ με την λύση που δίνει το ETL εργαλείο Informatica PowerCenter.

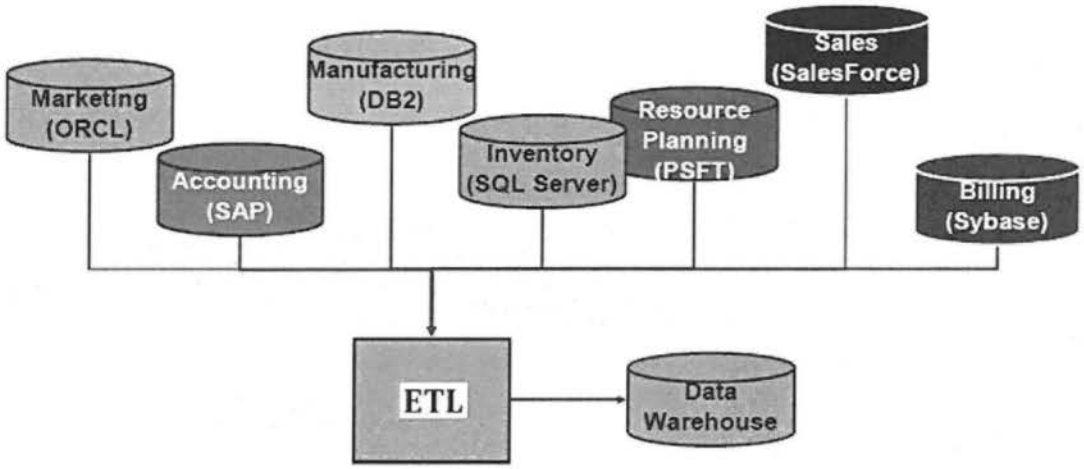
Η μεθοδολογία ETL συνοπτικά αποτελείται από τρείς φάσεις που είναι :

- **Extract** εξαγωγή των δεδομένων από το πηγαίο σύστημα
- **Transform** είναι ο κατάλληλος μετασχηματισμός τους και
- **Load** φόρτωση των τελικών δεδομένων στην αποθήκη.

Σε αυτό το σημείο θα ήρθε η ώρα να μιλήσουμε για της λεγόμενες Αποθήκες Δεδομένων ή αλλιώς Data Warehouses (DW). Από την στιγμή που οι απαιτούμενες βάσεις έχουν ενοποιηθεί και έχει γίνει η κατάλληλη μετατροπή των δεδομένων, στη συνέχεια τα δεδομένα αυτά πρέπει να αποθηκευτούν κάπου.

Το μέρος που αποθηκεύονται τα επεξεργασμένα δεδομένα είναι το Data Warehouse, δηλαδή η αποθήκη μας και όλη η διαδικασία που προηγείται ονομάζεται Data Warehousing.

Τώρα που τα δεδομένα είναι δομημένα σύμφωνα με της ανάγκες της εταιρίας, η εταιρία είναι σε θέση να κάνει μια ανάλυση και να βγάλει πολλές χρήσιμες πληροφορίες και συμπεράσματα μέσα από την αποθήκη αυτή. Από της πληροφορίες αυτές μπορεί να εξαρτηθεί το μέλλον και η πρόσφορη ανάπτυξη της.



Πηγή: Informatica User Manual

Τέλος θα ήθελα να αναφέρω ότι ένα μέρος του Data Warehousing είναι οι DSS τεχνολογίες. Δεν θα επεκταθώ πολύ σε αυτά μιας και το πρακτικό μέρος δεν συμπεριλάμβανε DSS συστήματα.

Με λίγα λόγια για να μπορεί ένας οργανισμός να αξιοποιήσει την αποθήκες του προς όφελος του, δηλαδή να αξιοποιήσει της πληροφορίες (που λέγαμε πιο πάνω) καταλλήλως, γίνεται χρήση DSS (Decision Support System) συστημάτων. Τα συστήματα αυτά βοηθούν της διοικητικές μονάδες, της λειτουργικές μονάδες και της μονάδες σχεδιασμού ενός οργανισμού να πάρουν αποφάσεις, για την ανάπτυξη του οργανισμού, οι οποίες θα ήταν δύσκολο να προσδιοριστούν εξ' αρχής. Με αυτό θέλω να πω ότι, τα DSS συστήματα χρησιμοποιούνται για να κάνουμε το μέλλον λίγο πιο ξεκάθαρο, έτσι οι αποφάσεις που παίρνονται γίνονται λιγότερο επικίνδυνες και περισσότερο κερδοφόρες.

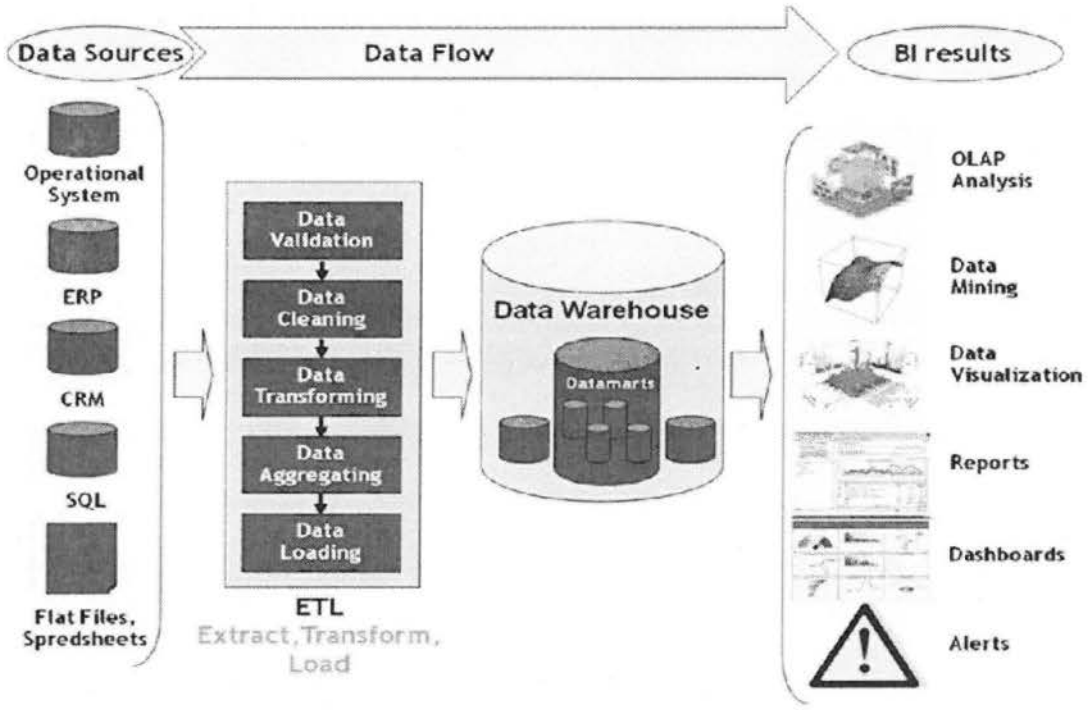
2. Data Warehousing

Γενική Περιγραφή Data Warehousing

Το «Data Warehousing» είναι μια ευρύτερη έννοια η οποία περιλαμβάνει την δημιουργία, την συντήρηση, την χρήση και την συνεχή ανανέωση των δεδομένων μέσα στην Αποθήκη Δεδομένων (Data Warehouse). Ένας άλλος συχνά χρησιμοποιούμενος όρος για το «Data Warehousing» είναι το «Business Intelligence» (BI)

Μια Αποθήκη Δεδομένων (DW) είναι μια σχεσιακή βάση δεδομένων (relational db) που έχει σχεδιαστεί περισσότερο για ερωτήματα (query) και ανάλυση και όχι τόσο για την επεξεργασία συναλλαγών. Ένα DW συνήθως περιέχει ιστορικά στοιχεία που προέρχονται από τα συστήματα καθημερινών συναλλαγών μιας εταιρίας, ωστόσο μπορεί να περιλαμβάνει και δεδομένα από άλλες πηγές όπως διαφορετικές βάσεις δεδομένων ή αρχεία (flat files). Με αυτό τον τρόπο επιτρέπει σε έναν οργανισμό να ενοποιήσει τα δεδομένα προερχόμενα από διαφορετικά παραγωγικά συστήματα που μπορεί να έχει.

Εκτός από μια σχεσιακή βάση δεδομένων, ένα DW περιβάλλον περιλαμβάνει ETL λύσεις, OLAP συστήματα (online analytical processing), εργαλεία ανάλυσης και άλλες εφαρμογές που διαχειρίζονται τη διαδικασία της συλλογής δεδομένων και την παράδοση τους στους εξειδικευμένους χρήστες της εκάστοτε επιχείρησης.



Πηγή: link 17

Στην ιστορία του DW υπήρξαν δύο μεγαλύτερη ιδρυτές περί θεωρίας ανάπτυξης Data Warehousing ο W.Inmon και ο R.Kimball. Ανέπτυξαν τις θεωρίες τους περίπου την ίδια χρονική περίοδο (1996) τότε που οι αποθήκες δεδομένων άρχιζαν να ανθίζουν. Οι δύο θεωρίες διαπραγματεύονται δύο τελείως διαφορετικές (θα έλεγα αντίθετες) μεθοδολογίες σχεδιασμού και ανάπτυξης των Αποθηκών. Ωστόσο οι δυο μεθοδολογίες αυτές είναι τόσο θεμελιώδεις και καθοριστικές που έχουν επικρατήσει μέχρι σήμερα και χωρίζουν τους σχεδιαστές Αποθηκών σε δύο αντίπαλα στρατόπεδα.

Παρακάτω δίνονται οι ορισμοί των δύο μεγάλων ιδρυτών.

- Ένα DataWarehouse είναι ένα αντίγραφο των επιχειρηματικών δεδομένων ειδικά δομημένο για ερωτήματα (queries) και ανάλυση.
  -RalphKimball
- Ένα DataWarehouse αποτελεί μια ολοκληρωμένη (integrated), εξαρτώμενη από το χρόνο (time-variant) και μόνιμη, μη πτητική (non-volatile) συλλογή δεδομένων οργανωμένη κατά υποκειμενοστραφή (subject-oriented) ανάλυση με στόχο τη διαδικασία υποστήριξης λήψης αποφάσεων (DSS).
  -WilliamInmon

Ο R. Kimball υποστήριζε ότι «...μια Αποθήκη Δεδομένων δεν είναι τίποτα άλλο πέρα από ένωση του συνόλου των datamarts». Τα μεμονωμένα datamart συχνά αποτελούν πρότυπο για ένα συγκεκριμένο επιχειρησιακό τομέα, όπως οι "Πωλήσεις" ή η "Παραγωγή". Αυτά τα datamarts μπορούν τελικά να ενσωματωθούν για να δημιουργηθεί μια ολοκληρωμένη αποθήκη δεδομένων. Με άλλα λόγια ο Kimball ανέπτυξε μια από-κάτω-προς-τα-πάνω (button-up) μεθοδολογία σχεδιασμού, με σκοπό αρχικά να δημιουργούνται μικρές αυτοτελής αποθηκούλες και στη συνέχεια να ενώνονται σε ολοκληρωμένο DW.

Από την άλλη πλευρά ο W. Inmon είπε ότι «...έστω και αν συγκεντρώσουμε όλα τα μικρά ψαράκια του ωκεανού μαζί, δεν πρόκειται να δημιουργήσουμε μια φάλαινα». Με αυτά τα λόγια ήθελε να υποστηρίξει την δικιά του, τελείως αντίθετη, μεθοδολογία σχεδιασμού που βασίζεται σε μια από-πάνω-προς-τα-κάτω προσέγγιση (top-down) υλοποίησης. Όπου πρώτα δημιουργείται ολοκληρωμένη η Αποθήκη Δεδομένων από της ήδη υπάρχουσες πηγές και στην πορεία του έργου προστίθενται και δημιουργούνται ξεχωριστά datamarts αν χρειαστούν. Για παράδειγμα, ένα ηλεκτρονικό κατάστημα μπορεί να αποτελέσει πηγή δεδομένων για την ανάπτυξη ενός DW. Ωστόσο στη συνέχεια η διοίκηση μπορεί να αποφασίσει ότι είναι απαραίτητη μια επιπρόσθετη διάσταση πχ Σημείο Πώλησης, τότε προστίθενται επιπλέον datamarts.

Ας αναλύσουμε τον ορισμό που δίνει ο Inmon μιας και αποτελεί πολύ ενδιαφέρον για την κατανόηση ενός DW.

- **Υποκειμενοστραφής (Subject Oriented)**

Τα Datawarehouses έχουν σχεδιαστεί για να βοηθήσουν μια επιχείρηση στην ανάλυση των δεδομένων της. Για παράδειγμα αν θέλουμε να μάθουμε αναλυτικά για τις πωλήσεις της εταιρίας μπορούμε να χτίσουμε μία Αποθήκη που θα εστιάζεται αποκλειστικά στις πωλήσεις. Χρησιμοποιώντας αυτήν την Αποθήκη μπορούμε να κάνουμε ερωτήσεις τύπου «Ποιος ήταν ο καλύτερος πελάτης για κάποιο προϊόν πέρσι;» Αυτή η ικανότητα να ορίζουμε την Αποθήκη Δεδομένων σε σχέση με ένα προσανατολισμένο θέμα, που στην προκειμένη περίπτωση είναι Πωλήσεις, κάνει την Αποθήκη υποκειμενοστραφή.

- **Ολοκληρωμένη (Integrated)**

Ο όρος «ολοκληρωμένη αποθήκη» είναι στενά συνδεδεμένος με το «υποκειμενοστραφής». Οι Αποθήκες Δεδομένων μαζεύουν δεδομένα από ξέχωρες και διαφορετικές πηγές και τα αποθηκεύουν σε μια ομογενοποιημένη μορφή. Θα πρέπει να επιλυθούν προβλήματα όπως η ονοματοδοσία, συγκρούσεις και αντιφάσεις μεταξύ των μετρήσιμων μονάδων, όταν επιτευχθεί κάτι τέτοιο μόνο και μόνο τότε μπορούμε να μιλήσουμε «ολοκλήρωση» ή ομοιογένεια.

- **Μη πτητική (Nonvolatile)**

Ο όρος αυτός υποδηλώνει ότι από την στιγμή που τα δεδομένα μπουν στην Αποθήκη δεν πρέπει να αλλάξουν, αλλά να παραμείνουν σε μόνιμη φάση απαράλλαχτα. Και αυτό είναι πολύ λογικό γιατί μας δίνεται η ικανότητα να αναλύσουμε τη έχει συμβεί στην πραγματικότητα, θέλουμε να έχουμε την πραγματική εικόνα των γεγονότων

- **Εξαρτώμενη από τον χρόνο (TimeVariant)**

Με αυτόν τον όρο εννοείται ότι οι Αποθήκες Δεδομένων μπορούν να δώσουν πληροφορίες για τις αλλαγές που συμβαίνουν στο πέρας του χρόνου. Λογού χάρη, οι αναλυτές μπορούν να παρακολουθήσουν τι τάσεις επικρατούν στην αγορά μιας επιχείρησης, προς τα πού βαδίζει και ίσως που μπορεί να καταλήξει. Αυτό συμβαίνει επειδή τα DataWarehouses αποθηκεύουν τεράστιο όγκο δεδομένων από την αρχή της ίδρυσης μιας εταιρίας έως και όσα χρόνια κρατήσει.

Σύγκριση των OLTP και Data Warehousing

Για καλύτερη κατανόηση του Data Warehousing είναι σκόπιμο να γίνει σύγκριση με κάποιο άλλο σύστημα αποθήκευσης δεδομένων. Τα OLTP συστήματα είναι ένα τέλειο παράδειγμα σύγκρισης, μιας και όλη μας λίγο ή πολύ έχουμε συναναστραφεί με τέτοια συστήματα.

Όπως λέει και η ονομασία του Online Transaction Processing (επεξεργασία ηλεκτρονικών συναλλαγών) είναι μια κατηγορία πληροφοριακών συστημάτων που διαχειρίζονται προγράμματα και λογισμικά που είναι προσανατολισμένα για online συναλλαγές. Η κύρια λειτουργία των προγραμμάτων αυτών είναι εισαγωγή και ανάκτησή των δεδομένων που προέρχονται από καθημερινές συναλλαγές, η ανάκτηση γίνεται με προκαθορισμένες εντολές (με query). Ένα παράδειγμα OLTP συστήματος είναι τα τραπεζικά μηχανήματα αυτόματης ανάληψης (ATM).

Μια κύρια διαφορά των δύο συστημάτων είναι ότι σε μία Αποθήκη Δεδομένων τα δεδομένα δεν βρίσκονται σχεδόν ποτέ σε τρίτη κανονική μορφή (3NF), ενώ για OLTP συστήματα κάτι τέτοιο είναι σύνηθες. Για τις Αποθήκες δεν υπάρχει περιορισμός χώρου, όπως συμβαίνει με τα OLTP συστήματα, και οι ανάγκες για query είναι διαφορετικές και σύνθετες. Για το λόγο αυτό θέλουμε οι δομές των δεδομένων να είναι όσο πιο μη κανονικοποιημένες, πολλές φορές επιθυμητό είναι το τελείως αντίθετο.

Οι Αποθήκες Δεδομένων και τα OLTP περιβάλλοντα έχουν πολλές διαφορές στις απαιτήσεις τους, ας δούμε αναλυτικά ποιες είναι αυτές:

- **Φόρτος Εργασίας:** οι Αποθήκες Δεδομένων είναι σχεδιασμένες ώστε να εξυπηρετούν adhoc queries. Φυσικά είναι δύσκολο να γνωρίζουμε από πριν τι φόρτο εργασίας θα έχει μια Αποθήκη, για το λόγο αυτό κατά το σχεδιασμό γίνεται προσπάθεια να βρεθεί η βέλτιστη λύση ώστε μελλοντικά να είναι εφικτή η εκτέλεση οποιανδήποτε queries στην βάση.

  Από την άλλη τα OLTP είναι σχεδιασμένα να διαχειρίζονται μόνο προκαθορισμένες ερωτήσεις και εντολές. Πχ ένα ATM έχει συγκριμένες λειτουργίες στην οθόνη του, οι ερωτήσεις που μπορούμε να κάνουμε είναι περιορισμένες

- **Μετατροπές Δεδομένων:** μια Αποθήκη Δεδομένων ενημερώνεται σε τακτική βάση από ETL διεργασίες (που τρέχουν κάθε βράδυ ή εβδομαδιαίως) όπου τα δεδομένα περνάνε από δύσκολες και βαριές για το σύστημα τεχνικές μετατροπής και τροποποίησης. Σε αυτή τη περίπτωση ο χρήστης δεν συμμετέχει άμεσα στην ενημέρωση της Αποθήκης.

Ενώ στα OLTP συστήματα, ο τελικός χρήστης έχει άμεση επαφή με την ενημέρωση και την εισαγωγή των δεδομένων στο σύστημα. πχ Το ταμειακό σύστημα σε ένα κατάστημα ρούχων. Η OLTP βάση είναι πάντα ενημερωμένη και απεικονίζει ακριβώς την δεδομένη κατάσταση που βρίσκεται η κάθε επιχειρησιακή συναλλαγή.

- **Σχεδιασμός Σχήματος:** οι Αποθήκες Δεδομένων συχνά χρησιμοποιούν αποκανονικοποιημένα ή μερικώς αποκανονικοποιημένα σχήματα για την βέλτιστη ανταπόκριση σε ερωτήματα.

  Από την άλλη, στα OLTP συστήματα συχνά χρησιμοποιείται πλήρως κανονικοποιημένα σχήματα ώστε η ενημέρωση/εισαγωγή/διαγραφή των δεδομένων να γίνεται με όσο πιο βέλτιστο τρόπο γίνεται και να διατηρείται η συνοχή των δεδομένων.

- **Τυπικές Διεργασίες:** ένα τυπικό query σαρώνει χιλιάδες ή εκατομμύρια γραμμές σε μια Αποθήκη. Για παράδειγμα, «Βρες το σύνολο των πωλήσεων για πελάτες του προηγούμενου μήνα».

  Ενώ μια τυπική OLTP διεργασία έχει πρόσβαση σε μια χούφτα εγγραφών. Για παράδειγμα, «Ανέκτησε την τρέχουσα παραγγελία του πελάτη».

- **Ιστορικά Δεδομένα:** Συνήθως στις Αποθήκες συγκεντρώνονται δεδομένα πολλών μηνών ή χρονών. Αυτό εξυπηρετεί πολύ όσον αφορά την ιστορική ανάλυση.

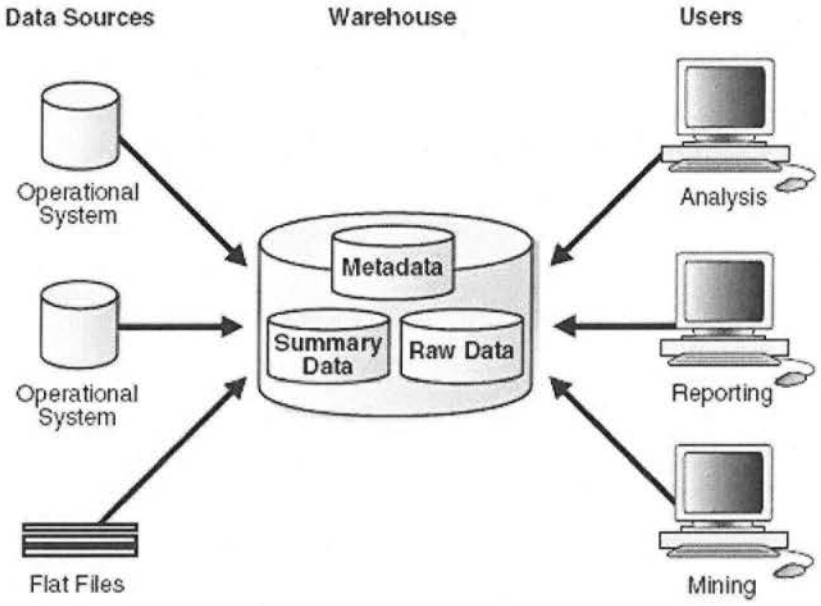
  Ενώ τα OLTP συστήματα δεν περιέχουν δεδομένα που ξεπερνούν μερικές βδομάδες ή μήνες. Μαζεύουν όσες εγγραφές χρειάζονται για τις ανάγκες της τρέχουσας συναλλαγής.

Αρχιτεκτονικές Αποθηκών Δεδομένων

Data Warehouse και οι αρχιτεκτονικές τους ποικίλουν ανάλογα με τις προτεραιότητες και απαιτήσεις της εκάστοτε επιχείρησης. Οι τρείς πιο διαδιδόμενες αρχιτεκτονικές είναι:

- **Data Warehouse ArchitectureBasic:** είναι η πιο απλή αρχιτεκτονική μιας Αποθήκης Δεδομένων.

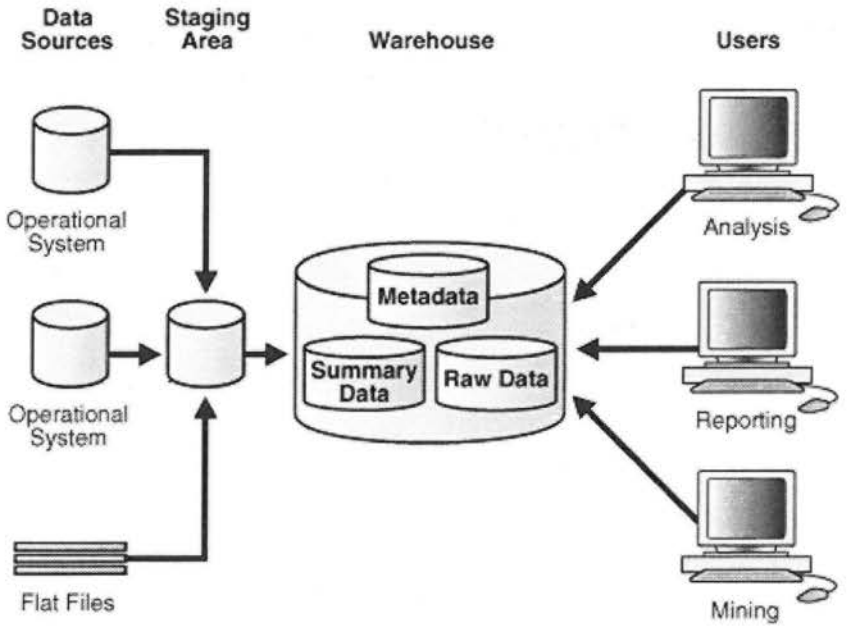
  Σε αυτήν την περίπτωση τα δεδομένα, προερχόμενα από διαφορετικές πηγές, κατευθείαν αποθηκεύονται στην Αποθήκη και από κει τα δουλεύουν οι τελικοί χρήστες.



Πηγή: link 3

- **Data Warehouse Architecture with Staging Area:** σε αυτή την περίπτωση τα επιχειρησιακά δεδομένα δεν αποθηκεύονται απ' ευθείας στην Αποθήκη αλλά περνάνε από ένα ενδιάμεσο στάδιο, stage στάδιο, όπου υπόκεινται σε κάποια επεξεργασία. Με λίγα λόγια τα δεδομένα περνάνε από επεξεργασία και καθαρισμό πριν καταλήξουν στην Αποθήκη.

  Επίσης το ενδιάμεσο στάδιο απλοποιεί την διαδικασία ενσωμάτωσης των δεδομένων στην Αποθήκη και λειτουργεί εποικοδομητικά στην γενική διαχείριση της. Μια καλή τακτική είναι να μαζεύονται όλα τα δεδομένα σε ένα μέρος και ύστερα να ενσωματώνονται στην Αποθήκη, κάτι που θα ήταν αδύνατο χωρίς το ενδιάμεσο στάδιο. Για να το καταλάβουμε καλύτερα ας σκεφτούμε ότι τα δεδομένα έρχονται από διαφορετικές πηγές, σε διαφορετικά χρονικά διαστήματα και ο όγκος τους ποικίλει. Άρα συναντάμε εμπόδια όπως ποικιλία στους κύκλους επεξεργασίας των δεδομένων, περιορισμός στο hardware και στους πόρους δικτύου, τέλος μην ξεχνάμε ότι οι πηγές μπορεί να βρίσκονται σε διαφορετικές περιοχές οπότε έχουμε και γεωγραφικό περιορισμό. Με αποτέλεσμα μην είναι εφικτή η εξαγωγή των δεδομένων από όλες της πηγές ακριβώς την ίδια στιγμή. Για το λόγο αυτό είναι απαραίτητη η ύπαρξη ενός ενδιάμεσου σταδίου που θα συσσωρεύσει όλα τα επιχειρησιακά δεδομένα και ύστερα να ενσωματώνονται στην Αποθήκη.



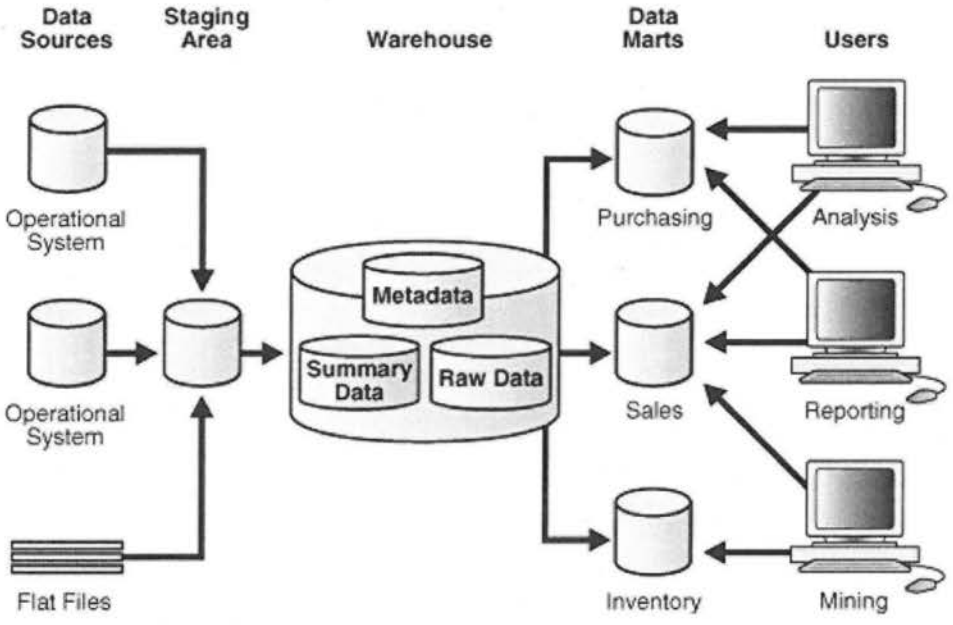
Πηγή: link 3

- **Data Warehouse Architecture with Staging Area and Data Marts:** η συγκεκριμένη αρχιτεκτονική εκτός των staging areas προσθέτει και data marts (αποθήκες συγκεκριμένης χρήσης) στον σχεδιασμό της.

  Παρόλο που η ακριβώς προηγούμενη αρχιτεκτονική είναι η πιο διαδεδομένη, μια επιχείρηση μπορεί να θελήσει να διαμορφώσει τα δεδομένα από την Αποθήκη και να τα συγκεντρώσει σε μικρότερες αποθήκες (datamarts) που θα προορίζονται για την ανάλυση συγκεκριμένου επιχειρησιακού σκοπού.

  Στο συγκεκριμένο παράδειγμα έχουμε τρείς ξεχωριστές αποθήκες: τις αγορές, τις πωλήσεις και το απόθεμα. Ένας οικονομικός αναλυτής μπορεί να θελήσει να κάνει ανάλυση στα ιστορικά στοιχεία των αγορών και πωλήσεων και να μην τον ενδιαφέρουν καθόλου τα αποθέματα, ο διαχωρισμός σε datamarts του προσφέρει αυτήν την δυνατότητα.

  Ένα άλλο παράδειγμα, είναι λογικό για το τμήμα Προσωπικού να έχουν πρόσβαση στις λεπτομέρειες πληροφορίες σχετικά με έναν υπάλληλο, δεν ισχύει το ίδιο για το τμήμα Πωλήσεων. Θα ήταν λάθος να μπορεί το τμήμα Πωλήσεων να έχει την δυνατότητα να μάθει πληροφορίες σχετικά με «μισθούς» ή «διευθύνσεις κατοικίας» των συνεργατών. Με άλλα λόγια ο διαχωρισμός σε datamarts δίνει την δυνατότητα στην επιχείρηση να διαχειριστεί καλύτερα τα τμήματα της.

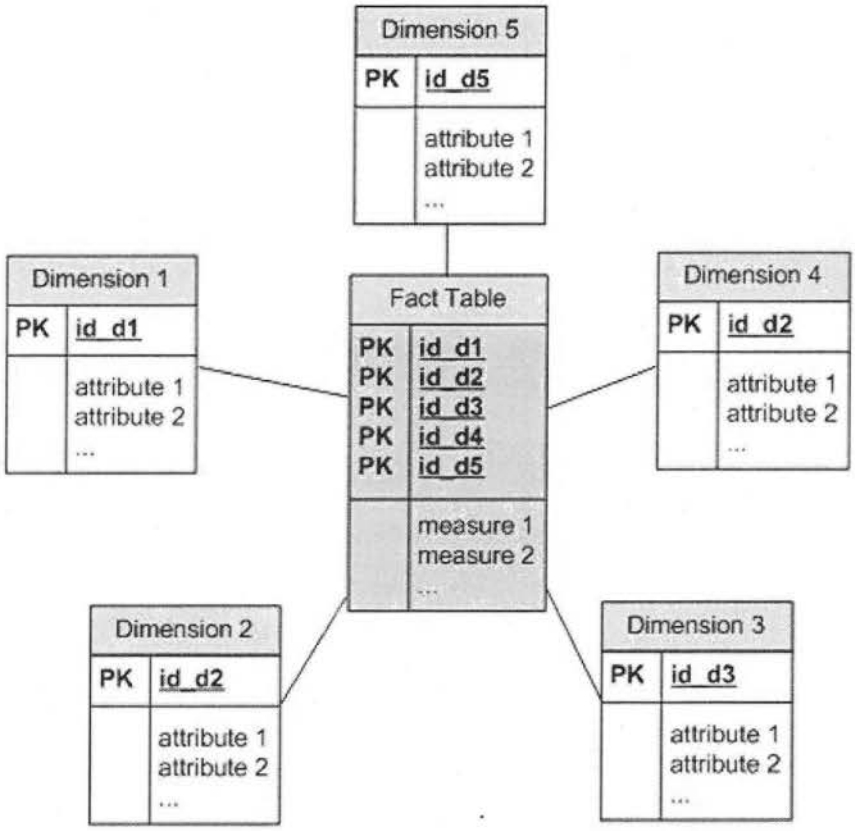


Πηγή: link 3

Σχήματα Αρχιτεκτονικής των Αποθηκών Δεδομένων

Το περιβάλλον ενός DW συνήθως δημιουργεί μετατροπές των σχεσιακών μοντέλων σχεδιασμού σε κάποια ειδικά αρχιτεκτονικά σχήματα. Υπάρχουν πολλά σχήματα στην αγορά που έχουν δημιουργηθεί ειδικά για Data Warehousing, ωστόσο τα κυριότερα που χρησιμοποιούνται είναι τρία : το Star σχήμα, το Snowflake σχήμα και το Fact Constellation σχήμα.

Star Σχήμα:



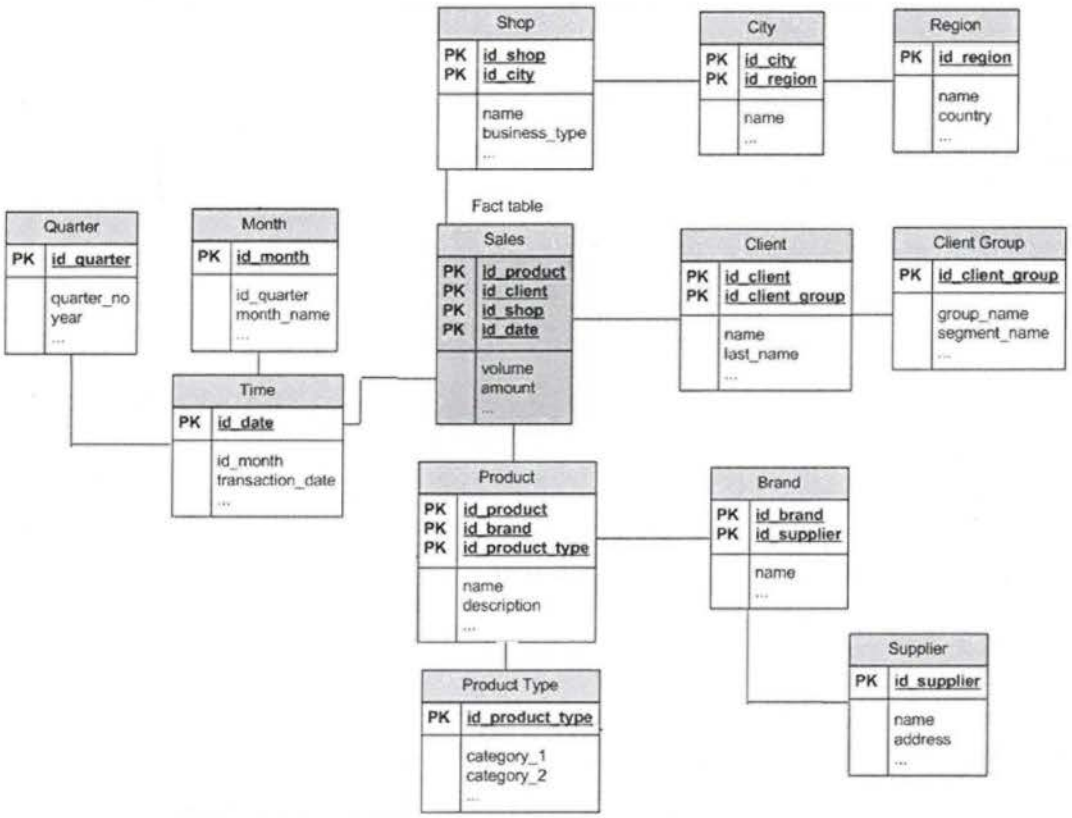
Πηγή: link 1

Ο πιο φυσικός τρόπος σχεδιασμού μιας Αποθήκης είναι με το Star Σχήμα. Υπάρχει μόνο μια ένωση μεταξύ των κυρίων πινάκων και των πινάκων που αποτελούν τις διαστάσεις του. Από εκεί προκύπτει και η ονομασία, είναι σαν να σχηματίζεται ένα αστέρι που έχει ένα ή περισσότερους κύριους πίνακες στο κέντρο και οι ακτίνες του δημιουργούν σχέσεις με τους πίνακες διαστάσεων.

Ένα Star (αστέρα) σχήμα βελτιστοποιεί την απόδοση του συστήματος με το να απλουστεύει τα query και να παρέχει γρήγορη ανταπόκριση του συστήματος.

Snowflake Σχήμα:

Το σχήμα της χιονονιφάδας είναι ένα πιο πολύπλοκο σχήμα αλλά εντάσσεται στο είδος αστέρα. Και σε αυτή τη περίπτωση η ονομασία του προέρχεται από την εικονική του απεικόνιση που θυμίζει μια χιονονιφάδα.

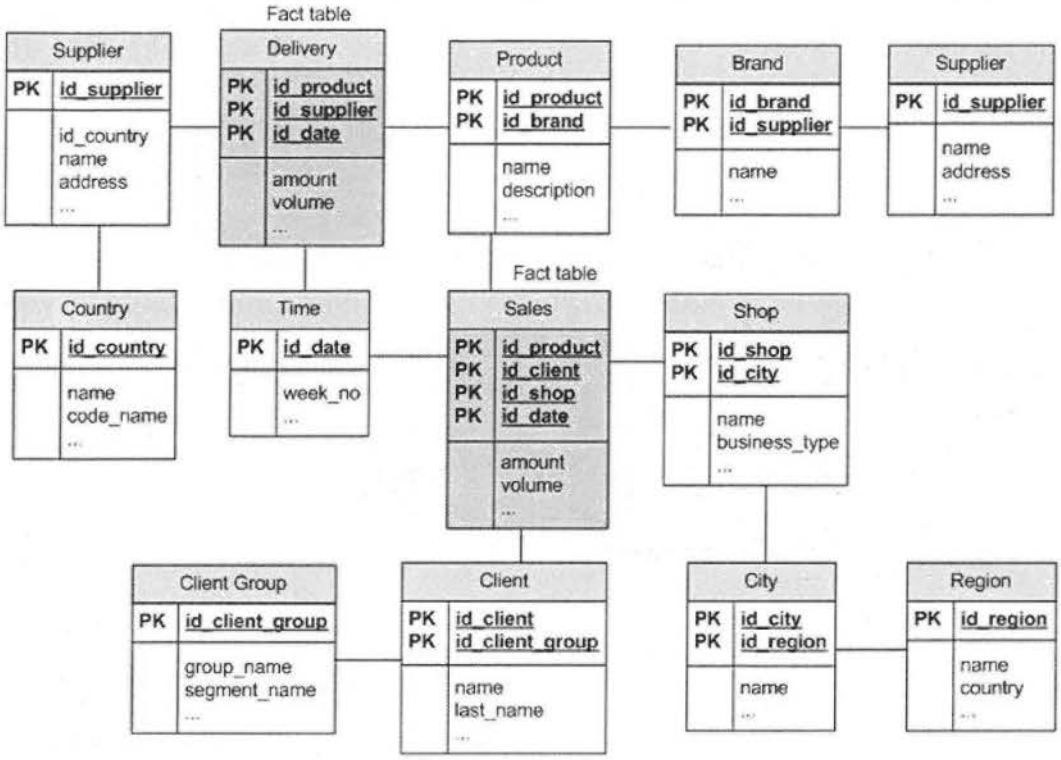


Πηγή: link 1

Η κύρια διαφορά των σχημάτων Snowflake είναι ότι οι διαστάσεις τους είναι σε κανονικοποιημένη μορφή και ο λόγος είναι για να αποφεύγεται ο πλεονασμός. Δηλαδή, ομαδοποιούνται τα δεδομένα διαστάσεων σε πολλαπλούς πίνακες αντί να είναι μαζεμένα σε έναν τεράστιο πίνακα. Για παράδειγμα μια διάσταση «Προϊόντα» σε ένα Star σχήμα όταν κανονικοποιηθεί θα δημιουργηθούν «κατηγορία_προιόντος» και «προέλευση_προιόντος», η ίδια διάσταση θα αποκτήσει περισσότερες διαστάσεις σχηματίζοντας έτσι μια χιονονιφάδα.

Θα λέγαμε ότι από την μια πλευρά αυτό εξοικονομεί χώρο στην Αποθήκη, από την άλλη όμως αυξάνεται ο αριθμός των πινάκων διαστάσεων και απαιτούνται περισσότερα ξένα κλειδιά. Με αποτέλεσμα να δυσκολεύουν τα query και συνεπώς να καθυστερεί λίγο παραπάνω η απόδοση του συστήματος.

Fact Constellation ή Galaxy Σχήμα:



Πηγή: link 1

Ουσιαστικά ένα Galaxy σχήμα αποτελείται από πολλά Star σχήματα που συνδέονται μεταξύ τους με πίνακες διαστάσεων. Ονομάστηκε γαλαξίας γιατί είναι ένα σύνολο από αστέρια. Αυτό το σχήμα αναπτύχθηκε για περιπτώσεις όπου χρειάζεται πολλαπλοί πίνακες γεγονότων να συνδέονται μεταξύ τους με πίνακες διαστάσεων.

Το κύριο μειονέκτημα αυτού του σχήματος είναι η πολύπλοκη αρχιτεκτονική του, ο designer πρέπει να είναι πολύ έμπειρος γιατί είναι πολλά που πρέπει να προβλεφθούν εκ τον προτέρων ώστε τα query σε αυτές τις αποθήκες να μην αποτελούν χρονοβόρες και μακρόσυρτες εντολές.

Αντικείμενα που χρησιμοποιεί μια Αποθήκη Δεδομένων

Υπάρχουν δύο τύποι πινάκων που χρησιμοποιούνται στην αρχιτεκτονική των Αποθηκών Δεδομένων και είναι οι πίνακες γεγονότων (facttables) και πίνακες διαστάσεων (dimensiontables).

Οι fact πίνακες είναι οι πιο ογκώδες πίνακες που υπάρχουν μέσα στην Αποθήκη Δεδομένων και κρατάνε όλα τα επιχειρησιακά μετρήσιμα (measurements) και τα επιχειρησιακά γεγονότα. Τα μετρήσιμα αποτελούν οτιδήποτε μπορεί να μετρηθεί και να επεξεργαστεί με αριθμητικές πράξεις (πχ πωλήσεις, κόστος και κέρδος).

Τυπικά οι πίνακες γεγονότων αποτελούνται από δεδομένα που είναι αριθμητικού τύπου ή αλλιώς μετρήσιμα και ξένα κλειδιά που συντελούν στη σύνδεση με τους πίνακες διαστάσεων. Οι πίνακες γεγονότων είναι αυτοί που αποτελούν το κέντρο ενός σχήματος αστέρα ή χιονονιφάδας και γύρω-γύρω περιτριγυρίζονται από τους πίνακες διαστάσεων. Σε περίπτωση που έχουμε πολλούς πίνακες γεγονότων μαζεμένους τότε μιλάμε για σχήματα γαλαξία.

Όπως αναφέρθηκε πιο πάνω οι fact πίνακες αποτελούνται από δύο τύπους πεδίων, αυτά που περιέχουν γεγονότα και αυτά που περιέχουν ξένα κλειδιά. Συνήθως το κύριο κλειδί των πινάκων αυτών είναι μια σύνθεση από όλα τα ξένα κλειδιά που περιέχονται σε αυτόν.

Ας περάσουμε στους πίνακες διαστάσεων. Οι πίνακες διαστάσεων όπως περιγράφει και το όνομα τους αποτελούν τις διαστάσεις των fact πινάκων. Οι πίνακες αυτοί υπάρχουν για να περιγράφουν τα δεδομένα των fact πινάκων. Επίσης καμιά φορά αναφέρονται ως lookup ή αναφορικοί πίνακες γιατί τα ερωτήματα που κάνουμε στην βάση θα κάνουν κάποια join και θα κοιτάξουν τους πίνακες διαστάσεων για αναλυτικές και περιγραφικές απαντήσεις. Όταν λέω περιγραφικές απαντήσεις εννοώ ότι πχ τα ονόματα προϊόντων ή πελατών αποτελούν περιγραφές των κωδικών που περιέχονται στους πίνακες γεγονότων.

Ας φανταστούμε μια εταιρία πουλάει κάποια προϊόντα σε πελάτες και κάθε πώληση αποτελεί ένα γεγονός που συμβαίνει. Κάθε fact πίνακας χρησιμοποιείται για να καταγράφει αυτά τα γεγονότα.

Time ID	Product ID	Customer ID	Unit Sold
4	17	2	1
8	21	3	2
8	4	1	1

Όπως βλέπουμε πιο πάνω κάθε γραμμή αποτελεί μία πώληση που πραγματοποιήθηκε. Ωστόσο στον πίνακα αυτόν παρατηρούμε μόνο αριθμούς που ναι μεν καταλαβαίνουμε τι θέλουν να πουν αλλά δεν έχουμε ξεκάθαρη εικόνα. Αυτό που θέλω να πω είναι ότι δεν ξέρουμε ποιο είναι το προϊόν 17, ούτε το όνομα του πελάτη και ούτε τι ώρα το αγόρασε. Το μόνο που είναι ξεκάθαρο είναι ότι αγόρασε ένα τεμάχιο.

Άρα για να μάθουμε αναλυτικά για μια πώληση θα πρέπει μέσω των fact πινάκων να συνδεθούμε με τους πίνακες διαστάσεων, που θα μας δώσουν μια πιο ξεκάθαρη εικόνα.

Ας πούμε ότι προσθέτουμε την διάσταση των πελατών.

Customer ID	Name	Gender	Income	Education	Region
1	Brian Edge	M	2	3	4
2	Fred Smith	M	3	5	1
3	Sally Jones	F	1	7	3

Τώρα με ένα query στην βάση μπορούμε να μάθουμε ότι το όνομα του πελάτη με το κωδικό δύο που αγόρασε ένα τεμάχιο την τάδε ώρα είναι "FredSmith".

Με αυτόν τον τρόπο σιγά-σιγά προσθέτουμε διαστάσεις στους fact πίνακες ώστε να μπορούν να περιγραφούν όλες οι οντότητες και τα μετρήσιμα που υπάρχουν στον πίνακα αυτόν.

Η στήλη «UnitSold» αποτελεί τα μετρήσιμο μέρος του πίνακα. Έχουμε την δυνατότητα να κάνουμε οποιαδήποτε αριθμητική πράξει στην στήλη αυτή για να βγάλουμε συμπεράσματα πχ πόσα συνολικά τεμάχια πουληθήκαν την τάδε εβδομάδα.

Τα μετρήσιμα διαχωρίζονται σε τρείς κύριες κατηγορίες:

- **Προσθετικά (additive):** είναι τα μετρήσιμα που μπορούν να προστεθούν σε οποιαδήποτε διάσταση. Πχ για να μάθουμε πόσα τεμάχια πουλήθηκαν σε χρονικό διάστημα ενός μήνα, θα αθροίσουμε τα τεμάχια στην διάσταση του χρόνου.
- **Μη Προσθετικά (nonadditive):** που δεν μπορούν να προστεθούν σε καμία διάσταση. Ένα παράδειγμα είναι οι διαστάσεις που αποθηκεύονται τα ποσοστά (%), προφανώς θα βγάλουμε λανθασμένο συμπέρασμα αν προσθέσουμε όλα τα ποσοστά πώλησης μέσα στην εβδομάδα. Τα ποσοστά δεν αθροίζονται σε καμία διάσταση.
- **Ημί Προσθετικά (semiadditive):** που μπορούν να προστεθούν σε κάποιες από τις διαστάσεις. Ένα καλό παράδειγμα είναι τα μετρήσιμα που δεν μπορούν να αθροιστούν στην διάσταση του χρόνου. Ας πούμε ότι έχουμε μια αποθήκη με 10 κουτιά, την Δευτέρα πουλιούνται 3 κουτιά, την Τρίτη 2, Τετάρτη 0, Πέμπτη 1 και πάει λέγοντας. Για να μάθουμε το τρέχον σύνολο κουτιών στην αποθήκη δεν έχει νόημα να προσθέσουμε όσα κουτιά πουλήθηκαν.

3. ETLδιαδικασίες

Από την στιγμή που αναλυθήκαν κάποιες βασικές έννοιες περί Data Warehouse ήρθε η ώρα να μιλήσουμε για ένα από τα βασικότερα κομμάτια του Data Warehousing τις περίφημες ETL διαδικασίες. Χωρίς τις διαδικασίες αυτές μια Αποθήκη Δεδομένων δεν θα μπορούσε να υπάρξει είτε θα καταντούσε μια Αποθήκη σκουπιδιών.

Τα δεδομένα που έρχονται από τα παραγωγικά συστήματα μιας επιχείρησης, ανά τακτά διαστήματα πρέπει να φορτώνονται στην Αποθήκη Δεδομένων, ώστε να την διατηρούν ενημερωμένη. Συνήθως η ενημέρωση γίνεται κάθε βράδυ. Για να γίνει αυτό, δεδομένα από ένα ή περισσότερα λειτουργικά συστήματα πρέπει να εξαχθούν από τα πηγαία συστήματα και να αντιγραφτούν στην Αποθήκη. Η διαδικασία της εξαγωγής των δεδομένων από τα παραγωγικά συστήματα και η αποθήκευση/μεταφορά τους στην Αποθήκη είναι διαδεδομένη ως ETL διαδικασία.

Τα πράγματα δεν είναι, όμως, τόσο απλά όσο ακούγονται. Συχνά, ο σχεδιασμός και η συντήρηση μιας ETL διαδικασίας, θεωρείται ότι είναι το πιο δύσκολο, χρονοβόρο και πόρο-καταναλωτικό (που καταναλώνει πολλούς πόρους συστήματος) μέρος ενός έργου. Ουσιαστικά η επιτυχία ή αποτυχία ενός Data Warehouse έργου εξαρτάται από το πόσο καλά είναι σχεδιασμένη και υλοποιημένη η ETL διαδικασία.

Τα αρχικά ETL σημαίνουν Extract, Transform και Load. Με άλλα λόγια, με την βοήθεια των ETL διαδικασιών τα δεδομένα εξάγονται από τα διάφορα συστήματα, μορφοποιούνται και τέλος φορτώνονται στην μεγάλη κεντρική Αποθήκη.

Οι ETL μεθοδολογίες και διαδικασίες είναι γνωστές για πάρα πολλά χρόνια στην επιχειρηματική αγορά. Βεβαίως, εκτός από περιβάλλοντα Data Warehouse οι ETL διαδικασίες χρησιμοποιούνται από μεγάλη ποικιλία εφαρμογών και συστημάτων βάσεων δεδομένων. Είναι η ραχοκοκαλιά κάθε τμήματος IT μίας επιχείρησης. Οπουδήποτε δύο ή παραπάνω διαφορετικά συστήματα (ή εφαρμογές ή βάσεις δεδομένων) χρειάζεται να επικοινωνήσουν μεταξύ τους και να ανταλλάξουν πληροφορίες, τότε αν κοιτάξουμε πιο προσεχτικά, σίγουρα θα δούμε κάποια ETL διαδικασία που θα έχει αναπτυχθεί για τον σκοπό αυτό. Μόνο που ο φόρτος απαιτήσεων που συναντάει μια Αποθήκη δεδομένων είναι μεγαλύτερος σε σχέση με την ανταλλαγή δεδομένων μεταξύ εφαρμογών.

Τα περιβάλλοντα Αποθήκης Δεδομένων πέρα από την απλή ανταλλαγή πληροφοριών, όπως και μεταξύ άλλων συστημάτων, υπάρχει η ανάγκη για ενσωμάτωση, αναδιάταξη και ενοποίηση των δεδομένων. Δηλαδή, τα δεδομένα πρέπει να επεξεργαστούν με τρόπο κατάλληλο ώστε τα δεδομένα που θα συσσωρευτούν στην Αποθήκη να αποτελέσουν πλατφόρμα χρήσιμων πληροφοριών για τους ειδικούς του Business Intelligence.

Extract

Το πρώτο βήμα μιας ETL διαδικασίας είναι η Εξαγωγή των δεδομένων από τα παραγωγικά συστήματα για περαιτέρω χρήση σε ένα περιβάλλον Data Warehouse. Μετά από την εξαγωγή/αντιγραφή, τα δεδομένα είναι έτοιμα για μετασχηματισμό και φόρτωση στην Αποθήκη. Δίπλα στην λέξη εξαγωγή αναφέρω και την λέξη αντιγραφή, με αυτό θέλω να πω ότι ουσιαστικά στο πρώτο βήμα αντιγράφει τα δεδομένα από το πηγαίο σύστημα και δεν τα σβήνει από την πηγή, όπως μπορεί να εννοηθεί με την εξαγωγή.

Τα περισσότερα έργα Data Warehousing έχουν να κάνουν με δεδομένα που προέρχονται από διαφορετικές πηγές (διαφορετικά συστήματα). Το καθένα από τα συστήματα συνήθως έχει διαφορετική δόμηση των δεδομένων (πχ format) γι' αυτό και ο σκοπός του Datawarehousing είναι να τα μαζέψει σε ενιαία δομή και χώρο. Οι πηγές μπορεί να ποικίλουν, ξεκινώντας από συστήματα συναλλαγών (πχ ATM), σχεσιακές βάσεις, flat file (πχ excel), xml αρχεία, Cobol αρχεία, εφαρμογές όπως SAP, Siebel, PeopleSoft, άλλες πηγές μπορεί να είναι Power Exchange, Maiframes, ακόμη και οι μη σχεσιακές βάσεις δεδομένων όπως IMS (Information Management System), VSAM (Virtual Storage Access Method) ή ISAM (Indexed Sequential Access Method), τα δεδομένα μπορούν ακόμα και να έρχονται από webs craping ή screens craping.

Η δημιουργία και ο σχεδιασμός μιας extraction διεργασίας είναι η πιο χρονοβόρα διαδικασία απ' όλες τις υπόλοιπες διεργασίες ETL, μπορεί και όλου του Data Warehousing. Αυτό προκύπτει από το γεγονός ότι το πηγαίο σύστημα μπορεί να είναι πάρα πολύ περίπλοκο και φτωχά τεκμηριωμένο με αποτέλεσμα ο προσδιορισμός των δεδομένων που πρέπει να εξαχθούν να γίνεται φοβερά δύσκολος. Επιπρόσθετα, πρέπει να ληφθεί υπόψη ότι η εξαγωγή των δεδομένων πρέπει να γίνεται περιοδικά, όχι μόνο μια, για να ενημερωθεί η Αποθήκη με όλες τις αλλαγές που έγιναν από την προηγούμενη ενημέρωση. Δηλαδή η διαδικασία της εξαγωγής πρέπει να μελετηθεί και να σχεδιαστεί έτσι ώστε να είναι σε θέση να επαναλαμβάνεται τακτικά χωρίς να επιβαρύνει πολύ το πηγαίο σύστημα και να είναι πλήρως προσαρμοσμένο στην τεχνολογία και το υλικό του πηγαίου συστήματος.

Τέλος, στην επιλογή της μεθοδολογίας για την διαδικασία εξαγωγής είναι απαραίτητο να ληφθούν υπόψη κάποια βασικά πράγματα. Ένα από τα σημαντικότερα κριτήρια επιλογής της μεθοδολογίας είναι το σύστημα προέλευσης των δεδομένων και οι επιχειρησιακές ανάγκες του τελικού προορισμού, της Αποθήκης Δεδομένων. Επίσης, πρέπει να ληφθεί υπόψη το γεγονός ότι ο φόρτος εργασίας της επιχείρησης μπορεί να αυξηθεί. Τότε και η διαδικασία εξαγωγής θα πρέπει να ανταποκρίνεται με βολικούς χρόνους για το πηγαίο σύστημα. Γιατί συχνά συμβαίνει η επιχείρηση να αυξήσει την παραγωγικότητα της και θελήσει να επιταχύνει την διαδικασία εξαγωγής αλλά πλέον να μην είναι δυνατή η προσθήκη επιπλέον λογικής στο σύστημα. Στην ουσία πρέπει να αποφασιστεί πως θα γίνει η διαδικασία εξαγωγής σε λογικό και υλικό επίπεδο.

Transform

Το στάδιο του μετασχηματισμού εφαρμόζει μια σειρά από κανόνες και λειτουργίες προκειμένου τα δεδομένα προερχόμενα από την εξαγωγή να φορτωθούν στο τελικό προορισμό τους. Ο μετασχηματισμός των δεδομένων εξαρτάται κατά κύριο λόγο από το πηγαίο σύστημα και από το business logic που θα έχει η Αποθήκη. Καμιά φορά με μικρές τροποποίησης τα δεδομένα είναι έτοιμα να φορτωθούν στην Αποθήκη Δεδομένων και άλλες φορές χρειάζονται πολύπλοκοι χειρισμοί προτού καταλήξουν εκεί.

Ανάλογα με την εφαρμογή και τα εργαλεία που χρησιμοποιούνται, η ETL διαδικασία μπορεί να περιλαμβάνει μια πληθώρα σειρά από μετασχηματισμούς Παρακάτω θα εξεταστούν τα βασικά από τα είδη μετασχηματισμών που υπάρχουν για την τροποποίηση των δεδομένων προτού αυτά καταλήξουν στην τελική βάση δεδομένων.

- Επιλογή συγκεκριμένων πεδίων για φόρτωση. (πχ να μην επιλέγονται οι κολόνες που έχουν null τιμές)
- Μετάφραση κωδικοποίησης. (πχ η πηγή μπορεί χρησιμοποιεί την κωδικοποίηση 1για άντρες και 2 για γυναίκες, ενώ στην Αποθήκη θέλουμε να έχουμε Α για άντρες και Γ για γυναίκες)
- Κωδικοποίηση τιμών. (πχ αντιστοίχιση «Άντρας» με «Α»)
- Προσθήκη νέων πεδίων που προέρχονται από υπολογισμούς με άλλα πεδία. (πχ overall_price=qty*unit_price)
- Διαλογή (Sorting)
- Ένωση δεδομένων από πολλαπλές πηγές (join, lookup, merge)
- Άθροιση (aggregation). (πχ σύνολο πωλήσεων για κάθε κατάστημα)
- Δημιουργία surrogate-key
- Περιστροφή ή μετατροπή των γραμμών σε στήλες ή το αντίθετο.
- Διάσπαση μιας στήλης σε πολλαπλές στήλες.
- Κανονικοποίηση πινάκων (πχ μετακίνηση μιας σειράς από διευθύνσεις που επαναλαμβάνονται, σε έναν ξεχωριστό πίνακα που θα έχει τις διευθύνσεις από μια φορά και που θα συνδέεται με τον προηγούμενο πίνακα)
- Αναζήτηση (lookup) και επικύρωση (validate) των σχετικών δεδομένων από τους πίνακες ή αρχεία αναφορών για αργά μεταβαλλόμενες διαστάσεις (slowchangingdimensions).
- Εφαρμογή οποιαδήποτε μορφής απλών ή συνθέτων μορφών επικαιροποίησης των δεδομένων. Είναι σαν δικλίδες ασφαλείας, αν κάποια επικαιροποίηση αποτύχει τότε συνεπάγεται μια πλήρη, μερική ή καμία απόρριψη των δεδομένων. Αυτομάτως αναλόγως με τον σχεδιασμό και τους κανόνες λειτουργίας της βάσης η απόρριψη περνάει στο επόμενο βήμα που μπορεί να είναι ειδοποίηση του χρήστη με κάποια exceptions.

Load

Το τελικό στάδιο των δεδομένων μέσα από την ροή μιας ETL διαδικασίας είναι το φόρτωμά τους στην βάση. Τα δεδομένα που ερχόμενα από τα πηγαία συστήματα πλέον είναι μετασχηματισμένα και έτοιμα για την αποθήκευσή στην Αποθήκη Δεδομένων.

Με ποιόν τρόπο θα γίνεται φόρτωμα των δεδομένων εξαρτάται αποκλειστικά από το businesslogic που ακολουθεί μια επιχείρηση. Κάποιες φορές είναι επιθυμητό η ιστορικότητα των δεδομένων να κρατιέται ανά εξάμηνο ή χρόνο. Δηλαδή, το φόρτωμα να γίνεται με update και η Αποθήκη να ενημερώνεται μόνο με καινούριες εγγραφές και όταν αλλάξει ο χρόνος να ξαναφορτώνονται όλες οι πληροφορίες από την αρχή, άσχετα αν το προηγούμενο χρόνο ήταν ίδιες. Άλλες φορές πάλι είναι επιθυμητή η κράτηση δεδομένων για λιγότερο χρονικό διάστημα πχ ενός χρόνου μόνο ή ολική αποθήκευση των δεδομένων για πάντα (πχ οι τραπεζικές Αποθήκες δεν σβήνουν ποτέ δεδομένα ακόμα και για πελάτες που δεν ζουν πια).

ETL Εργαλεία

Οι προγραμματιστές μπορούν να δημιουργήσουν ETL διεργασίες, χρησιμοποιώντας σχεδόν οποιαδήποτε γλώσσα προγραμματισμού. Αλλά η ανάπτυξη μιας τέτοιας διαδικασίας από το μηδέν πολλές φορές αποδεικνύεται περίπλοκη και η συντήρηση δύσκολη. Για το λόγο αυτό, όλο και περισσότερες εταιρίες προτιμούν την αγορά ETL εργαλείων, που είναι έτοιμα πακέτα εργαλείων στημένα με τρόπο να βοηθήσουν τον προγραμματιστή στην εύκολη δημιουργία των ETL διεργασιών.

Όμως τα ETL εργαλεία αποτελούνται από ένα πλαίσιο λειτουργιών, που πολλές φορές λειτουργεί ως περιορισμός και ο προγραμματιστής πρέπει να προσαρμόσει τις ιδέες του στα πλαίσια του εργαλείου. Ενώ η δημιουργία ETL διαδικασίες από κώδικα παρέχει πλήρη ανεξαρτησία και ελευθερία κινήσεων. Επίσης ένας καλογραμμένος κώδικας προσφέρει πιο γρήγορη εκτέλεση της ETL διαδικασίας γιατί είναι πιο ελαφρύς στην εκτέλεση απ' ότι τα βαριά και ογκώδη προγράμματα ETL εργαλείων.

Ωστόσο όταν τίθεται το ερώτημα κώδικας ή εργαλείο τότε πολλές φορές τα ETL εργαλεία είναι η πρώτη επιλογή. Η απόφαση μιας εταιρίας προς το πια κατεύθυνση να κινηθεί, σε γενικές γραμμές μπορεί να προσδιοριστεί από τρία βασικά κριτήρια.

Πρώτο κριτήριο είναι η πολυπλοκότητα των μετασχηματισμών που θα υποστούν τα δεδομένα. Όσο πιο μεγάλος ο δείκτης πολυπλοκότητας μετασχηματισμών τόσο πιο κατάλληλη είναι η αγορά ενός ETL εργαλείου. Δεύτερον, είναι η ανάγκη για καθαρισμό των δεδομένων (data cleansing). Εάν τα δεδομένα πρέπει να περάσουν από ενδελεχή διαδικασία καθαρισμού προτού καταλήξουν στην Αποθήκη Δεδομένων τότε είναι προτιμότερη η κατεύθυνση προς ETL εργαλείο. Διαφορετικά, ίσως είναι επαρκής να κατασκευαστεί μια ETL ρουτίνα από το μηδέν. Και τρίτον, παίζει μεγάλο ρόλο ο όγκος δεδομένων. Στην περίπτωση που ο όγκος των δεδομένων είναι μεγάλος και είναι επιθυμητή η ταχύτητα στην κίνηση των δεδομένων τότε ένα ETL εργαλείο μπορεί να ελαφρύνει κατά πολύ την δουλειά ενός προγραμματιστή. Γιατί σχεδόν όλα τα σύγχρονα εμπορικά ETL εργαλεία έχουν κατασκευαστεί δίνοντας μεγάλη έμφαση στην διαχείριση μεγάλου όγκου δεδομένων σε γρήγορους χρόνους.

Ως τελικό συμπέρασμα, θα έλεγα ότι είναι στο χέρι της κάθε επιχείρησης αν θα επιλέξει την χρήση ενός ETL ή όχι. Την επιλογή την προσδιορίζουν οι ανάγκες και οι προτεραιότητες από έχει ο επιχειρησιακός σκοπός.

Informatica PowerCenter

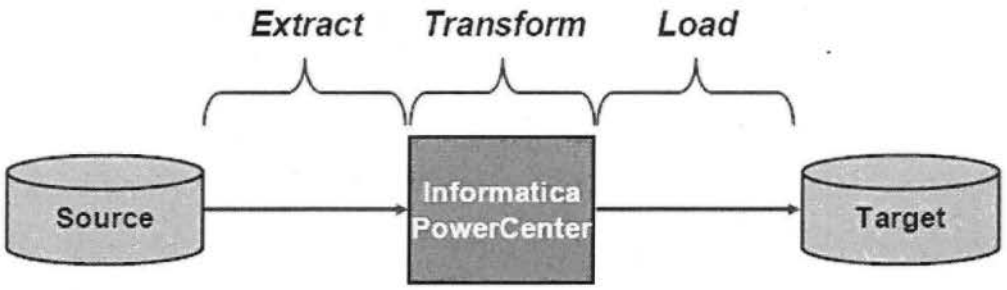
Informatica PowerCenter είναι ένα ευρέως διαδεδομένο ETL εργαλείο που συχνά το χρησιμοποιούν στην δημιουργία Αποθηκών Δεδομένων. Είναι από τα πιο δυνατά και προσεγμένα εργαλεία που υπάρχουν στην αγορά και πολλές εταιρίες επιλέγουν να εμπιστευτούν την αποθήκη τους στο εργαλείο αυτό.

Θα ήθελα να σταθώ λίγο σε αυτό το εργαλείο και εξηγήσω τον τρόπο λειτουργίας του, μιας και είναι το εργαλείο που χρησιμοποιήθηκε κατά την διεκπεραίωση αυτής της εργασίας.

Είναι ένα εργαλείο όπου ο κώδικας του απεικονίζεται γραφικά, πολλές φορές θα δούμε κουτάκια να ενώνονται μεταξύ τους με γραμμούλες. Ωστόσο αυτό δεν σημαίνει ότι πουθενά δεν γράφεται κώδικας. Αντίθετος σε κάθε μετασχηματισμό, ένα κουτάκι, έχουν γραφτεί αρκετές γραμμές sql κώδικα ώστε να επιτευχθεί το ζητούμενο αποτέλεσμα. Έτσι το εργαλείο αυτό συνδυάζει και το εύχρηστο γραφικό περιβάλλον και άφθονο χώρο για τον κώδικα ώστε να μπορεί ο προγραμματιστής να δημιουργήσει ακριβώς αυτό που θέλει.

Ξεκινώντας θα ήθελα να αναφέρω ότι το PowerCenter έχει την ιδιότητα να διαβάζει, να μετασχηματίζει και να γράφει τα δεδομένα γραμμή – γραμμή. Δηλαδή, αν έχουμε πέντε μετασχηματισμούς ώστε τα δεδομένα να φτάσουν από την πηγή ως τον στόχο τότε θα εκτελεστεί η πρώτη γραμμή των δεδομένων που θα περάσει και από τους 5 μετασχηματισμούς, μετά η τρίτη κ.ο.κ . Αυτή η ιδιαιτερότητα του εργαλείου είναι πολύ σημαντική για την κατανόηση του τρόπου λειτουργίας του.

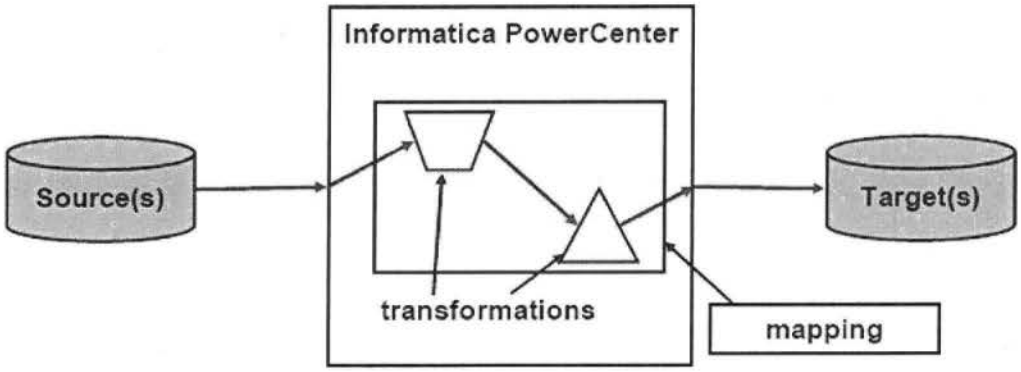
Οι βασικές έννοιες του PowerCenter είναι η πηγή (source), δηλαδή από κει που έρχονται τα δεδομένα, δεύτερον τα αντικείμενα μετασχηματισμού (transformation objects) με τα οποία ορίζεται η λογική των μετατροπών των δεδομένων, και τέλος ο στόχος (target) που είναι ο τελικός πίνακας όπου γράφονται τα δεδομένα.



Πηγή: Informatica User Manual

Mapping

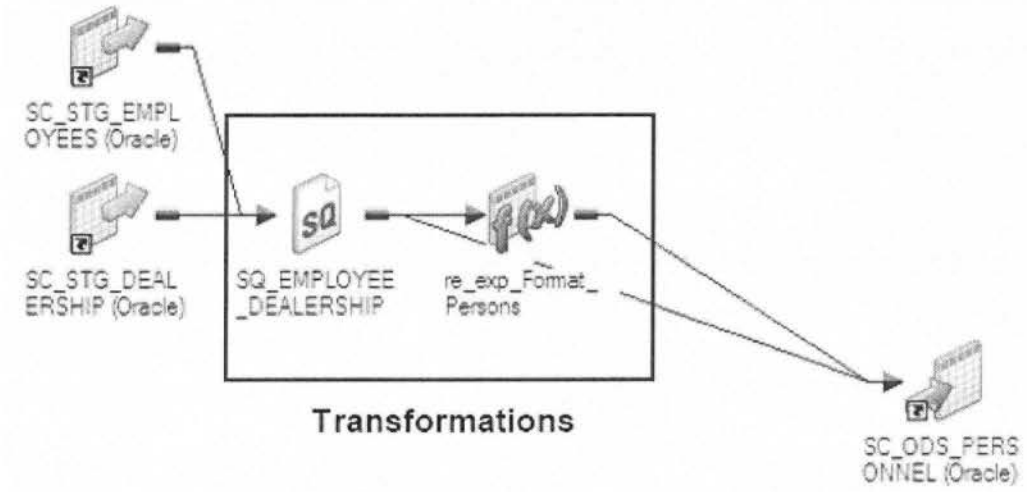
Ένα Mapping είναι ένα σύνολο από μετασχηματισμούς (transformations), που είναι τοποθετημένα είτε παράλληλα είτε σε σειρά, και που έχουν σκοπό τον μετασχηματισμό και την μεταφορά των δεδομένων από μια ή περισσότερες πηγές σε ένα ή περισσότερους στόχους. Ουσιαστικά όλη η λογική της ETL διαδικασίας ορίζεται μέσα σε ένα mapping. Με αυτό εννοώ, ότι ένα mapping διαβάζει δεδομένα από την πηγή, εφαρμόζει τους μετασχηματισμούς στα δεδομένα και τα γράφει στους target πίνακες. Επίσης ένα mapping υπάρχει μόνο μέσα στο PowerCenter, όπως δείχνει η εικόνα πιο κάτω.



Πηγή: Informatica User Manual

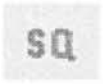
Transformations

Οι μετασχηματισμοί αυτό που κάνουν είναι να λαμβάνουν τα δεδομένα και να τα μετασχηματίζουν. Εδώ ανήκουν οποιαδήποτε μετατροπή μπορούν να υποστούν τα δεδομένα.

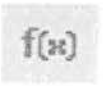


Πηγή: Informatica User Manual

Κάποια από τα βασικότερα transformations είναι:



SourceQualifier: διαβάζει τα δεδομένα από την πηγή



Expression: εκτελεί υπολογισμούς σε επίπεδο γραμμής


Filter: αποκλείει κάποιες γραμμές υπό συνθήκη


Sorter: Ταξινομεί τα δεδομένα


Aggregator: εκτελεί υπολογισμούς αθροίσματος


Joiner: ενώνει ετερογενής πηγές

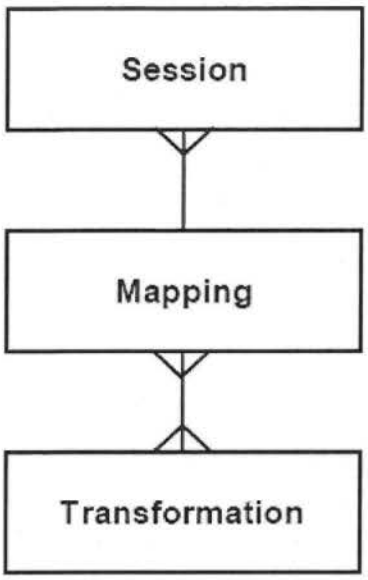

Lookup: ψάχνει για τιμές και τις περνάει σε άλλο αντικείμενο


Router: μαζεύει όλες τις γραμμές και τις περνάει υπό συνθήκη


Union: κάνει ένωση πινάκων

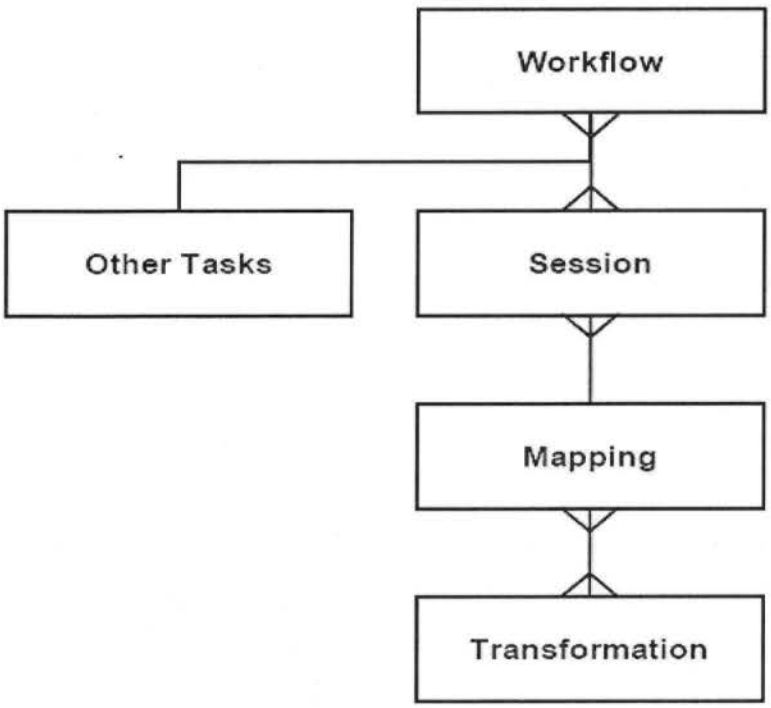
Session

Ένα session είναι το αντικείμενο που εκτελεί το mapping.



Workflow

Το workflow είναι διατεταγμένο σύνολο από ένα ή περισσότερα session και άλλα task, όπου εκτελείται ο απώτερος σκοπός της ETL διαδικασίας. Επίσης θα μπορούσαμε να πούμε ότι είναι ένα σύνολο οδηγιών προς τον διακομιστή (server) της Informatica για το πώς να εκτελέσει τα session και τα task.



Στα task ενός workflow συμπεριλαμβάνονται τα ακόλουθα:

StartTask: είναι η αρχή κάθε workflow


Session: είναι το αντικείμενο που αντιστοιχεί στο mapping

Command: τρέχει εντολές κώδικα

TaskAssignment: κάνει ανάθεση τιμών

Decision: δημιουργεί διακλαδώσεις υπό συνθήκη

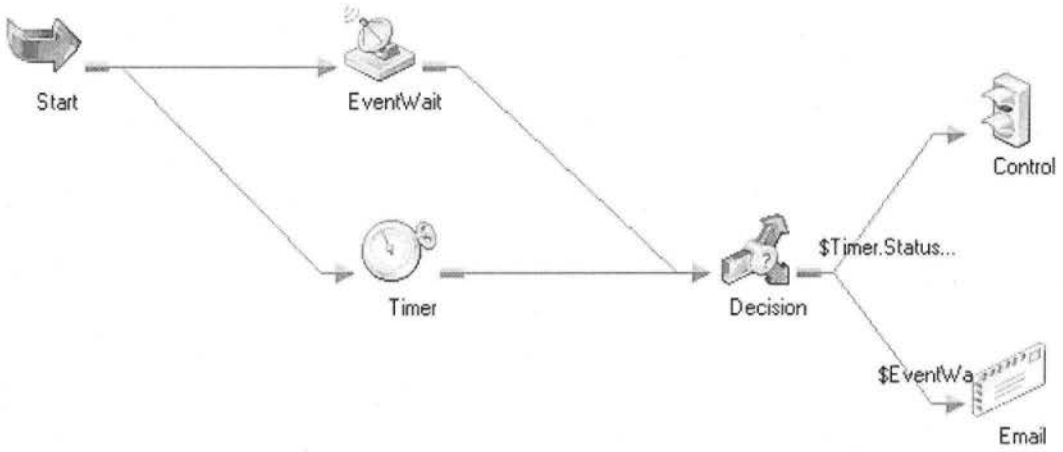
Email: στέλνει email

Timer: περιμένει ένα χρονικό διάστημα που του έχουμε ορίσει πριν εκτελέσει το επόμενο task.

Eventwait: διακόπτει την διεργασία και περιμένει ένα γεγονός

ControlTask: σταματάει, αποτυγχάνει ή ματαιώνει ένα workflow ή worklet

Ένα παράδειγμα χρήσης των task σε ένα workflow. Έχουμε ορίσει ένα EventWait να περιμένει μέχρι να γίνει ένα γεγονός (που μπορεί να είναι η ολοκλήρωση εκτέλεσης ενός session) και παράλληλα έναν Timer να περιμένει για ένα συγκεκριμένο χρονικό διάστημα πχ 10 λεπτά. Σε περίπτωσή που το γεγονός δεν συμβεί εντός δέκα λεπτών τότε το Control σταματάει όλο το workflow, σε αντίθετη περίπτωση αποστέλλεται ένα email πχ ότι η διαδικασία εκτελέστηκε επιτυχώς.



Mapplets

Ένα mapplet είναι ένα σύνολο μετασχηματισμών όπως το mapping με την διαφορά ότι δεν έχει source και target. Η χρήση του είναι εντός ενός mapping, πχ αντί να επιβαρύνουμε το mapping με ένα σύνολο μετασχηματισμών που επαναλαμβάνονται εντός mapping, έχουμε την δυνατότητα να βάλουμε όλους αυτούς τους μετασχηματισμούς σε ένα mapplet και να το χρησιμοποιήσουμε πλέον αυτό μέσα στο mapping. Δηλαδή η βασική του λειτουργία είναι η επαναχρησιμότητα (reusability), το αλλάζουμε μια φορά και αυτομάτως η αλλαγή γίνεται σε όλα τα σημεία όπου το έχουμε χρησιμοποιήσει.

Worklets

Όπως και το mapplet είναι ένα κομμάτι από mapping έτσι και το worklet είναι ένα κομμάτι από το workflow. Είναι ένα σύνολο από sessions και tasks. Χρησιμοποιείται για λόγους καθαρότητας, ευκρίνειας και επαναχρησιμότητας μέσα σε ένα workflow.

Πρακτικό Μέρος

1. Γενικά - Ορισμοί

Στη συνέχεια θα περιγραφεί ένα project το οποίο υλοποιήθηκε ως πρακτική εφαρμογή της θεωρίας που περιγράφηκε πιο πάνω. Πραγματοποιήθηκε δημιουργία μιας ολοκληρωμένης Αποθήκης Δεδομένων (data warehouse) η οποία αποτελείται από πολλάστarσχήματα. Η υλοποίηση έχει γίνει με την βοήθεια του ETLεργαλείου Informatica PowerCenter.

Πρόκειται για επαναπροσέγγιση ενός έργου που για διάφορους λόγους είχε επιλεγεί όλες οι ETL διαδικασίες να γίνουν με χρήση sqlscripts. Στην παρούσα εργασία γίνεται μια διαφορετική προσέγγιση, η υλοποίηση του έργου με χρήση ETL εργαλείου. Ο σκοπός τις πτυχιακής είναι να διερευνηθεί τι αντίκτυπο έχουν οι σύγχρονες τεχνολογίες στην βελτίωση Datawarehousing. Ποιος τρόπος είναι πιο γρήγορος στην επεξεργασία δεδομένων, πιο ευέλικτος, πιο εύχρηστος και αποτελεσματικός για τον τελικό χρήστη.

Ο Server στον οποίο στηρίχτηκε το Datawarehouse είναι ο IQ Server της Sybase.Τα δεδομένα που εισάγονται στον IQ Server (IQ) προέρχονται από το παραγωγικό σύστημα της KronosMarbles (KM).Η διαδικασία εισαγωγής καλείται ETL (Extraction, Transformation and Loading), η οποία κατά την εκτέλεση της περνάει τα δεδομένα από ένα στάδιο μετασχηματισμού, εμπλουτισμού και ομογενοποίησης. Ταυτόχρονα, τα δεδομένα φορτώνονται σε δομές κατάλληλες για γρήγορη απόκριση σε περιβάλλοντα Επιχειρηματικής Ευφυΐας (Business Intelligence).

Όπως αναφέρθηκε λίγο πιο πάνω, στο παρόν DW σύστημα επιλέχθηκε η δομή που είναι γνωστή με το όνομα Star Schema (Star Σχήμα).

Το Star Σχήμα χαρακτηρίζεται από ένα κεντρικό πίνακα που ονομάζεται Fact Table και από διάφορους περιφερειακούς πίνακες που ονομάζονται Πίνακες Διαστάσεων (Dimension Tables). Στον κεντρικό πίνακα καταγράφονται κάποιες κινήσεις συγκεκριμένης ανάλυσης (granularity). Η ανάλυση αυτή αναφέρεται στις Διαστάσεις του Star Σχήματος.

Μια (πολύ σημαντική) Διάσταση είναι η χρονική και στη συγκεκριμένη περίπτωση με τον όρο Ανάλυση αναφερόμαστε στη μικρότερη μονάδα χρόνου που καταγράφεται στον κεντρικό πίνακα (π.χ. Ημερομηνία, Εβδομάδα, Μήνας, ...).

Εκτός από τη χρονική διάσταση υπάρχουν και άλλες Διαστάσεις συγκεκριμένης ανάλυσης (π.χ. Πελάτης που μπορεί να καταγράφεται σε επίπεδο Κωδικού Πελάτη ή Ομάδας Πελατών κλπ).

Στον κεντρικό πίνακα καταγράφονται οι αξίες των κινήσεων (facts) μαζί με τις μονάδες μέτρησης τους (οι οποίες είναι κάποιες από τις διαστάσεις του Star Σχήματος). Οι αξίες αυτές αποτελούν τα Μετρήσιμα (Measures) του Star Σχήματος.

Όσον αφορά τις Διαστάσεις, στον κεντρικό πίνακα καταγράφονται συνήθως οι κωδικοί των διαστάσεων και τα υπόλοιπα στοιχεία τους (περιγραφές κλπ) καταγράφονται στους περιφερειακούς πίνακες διαστάσεων. Με αυτό τον τρόπο ο κεντρικός πίνακας συσχετίζεται (σαν να ενώνεται) με τους περιφερειακούς μέσω των κωδικών (το πολύ με ένα πίνακα κάθε φορά), δίνοντας μια οπτική απεικόνιση ενός άστρου (Star Σχήμα).

2. Σχεδίαση Λύσης

Ο σκοπός της εργασίας είναι υλοποίηση 12 διακριτών Star σχημάτων. Τα σχήματα αυτά θα περιέχουν ως κεντρικούς πίνακες τους πίνακες κινήσεων που θα περιγράφουν τις βασικές λειτουργίες της εταιρίας, πχ. ένα σχήμα θα αναφέρεται στους πελάτες, ένα άλλο στις αγορές, τιμολόγηση, αποθήκες κτλ.

Σύμφωνα με τη δομή του Star σχήματος, οι τελικοί πίνακες που ενημερώνονται από την ETL διαδικασία είναι δύο κύριων κατηγοριών:

- Πίνακες Κινήσεων (Fact Tables)
- Πίνακες Διαστάσεων (Dimension Tables)

Στο παρόν DW σύστημα χρησιμοποιήθηκαν κλειδιά μοναδικότητας μόνο στους πίνακες διαστάσεων οι οποίοι είναι δύο ειδών:

- Πίνακες Κυρίων Διαστάσεων (StrongDimensions). Παράδειγμα: Πελάτες. Συνήθως οι πίνακες αυτοί, εκτός από κάποια σημαντική περιγραφή (στο παράδειγμά μας η Επωνυμία του Πελάτη) περιέχουν διάφορες άλλες πληροφορίες (attributes) σχετικές με την οντότητα που περιγράφουν (στο παράδειγμά μας Διεύθυνση, ΑΦΜ, κωδικός τρέχοντος πωλητού κ.λπ

- Πίνακες Παραμέτρων. Παράδειγμα: Περιγραφές Πωλητών. Συνήθως οι πίνακες αυτοί περιέχουν μία περιγραφή (text) και για το λόγο αυτό πολλές φορές αποκαλούνται TextTables.

Η λύση αποτελείται από 74τελικούς πίνακες (data) από τους οποίους οι 12 είναι πίνακες κινήσεων (fact), 18 πίνακες διαστάσεων (dimension) και 44 παραμετρικούς πίνακες (paramfiles). Όλοι οι πίνακες έχουν δύο πρόσθετα πεδία (Company_Code, Fiscal_Year) για να ενοποιούμε τα στοιχεία των εταιριών και πολλών εταιρικών χρήσεων σε έναν ενιαίο πίνακα, για να μπορούν να δημιουργηθούν reports συγκρίσεων. Τα πρόσθετα πεδία (Company_Code, Fiscal_Year) προέρχονται από το αρχείο FIRM.DAT.

Το αρχείο FIRM.DAT και η σημασία του

Το αρχείο FIRM.DAT φορτώνεται όπως όλα τα υπόλοιπα αρχεία από το παραγωγικό σύστημα. Το αρχείο αυτό περιέχει μόνο μια γραμμή δεδομένων, όπου περιγράφεται ο κωδικός της εταιρίας από την οποία προέρχονται τα δεδομένα και η τρέχουσα χρονιά.

Κωδικός Εταιρίας (CompanyID)

Ο Κωδικός Εταιρίας είναι αριθμητικός και διψήφιος.

Στην παρούσα υλοποίηση υπάρχουν τρείς τιμές (1, 2 και 3 για Μάρμαρα, Γρανίτες και Αδρανή Υλικά αντίστοιχα).

Αποτελεί πεδίο όλων των πινάκων που ενημερώνονται από την ETL διαδικασία.

Ειδικά για τους πίνακες διαστάσεων, ο Κωδικός Εταιρίας αποτελεί μέρος του πρωτεύοντος κλειδιού (αυτού που εξασφαλίζει τη μοναδικότητα).

Κωδικός Εταιρικής Χρήσης (FiscalYear)

Ο Κωδικός Εταιρικής Χρήσης είναι αριθμητικός και τετραψήφιος με περιεχόμενο τον αριθμό του Έτους στο οποίο αναφέρεται.

Λόγω της ανεξαρτησίας των εταιρικών χρήσεων στο Παραγωγικό σύστημα, υπάρχει περίπτωση κάποιοι κωδικοί κάποιων διαστάσεων να χρησιμοποιηθούν για άλλους πίνακες από χρονιά σε χρονιά. Για να αποφύγουμε τα προβλήματα ο κωδικός εταιρικής χρήσης έγινε πεδίο όλων των πινάκων που ενημερώνονται από την ETL διαδικασία (όπως ακριβώς και ο Κωδικός Εταιρίας παραπάνω).

Ειδικά για του πίνακες διαστάσεων, ο Κωδικός Εταιρικής Χρήσης αποτελεί μέρος του πρωτεύοντος κλειδιού (αυτού που εξασφαλίζει τη μοναδικότητα). Αυτό σημαίνει ότι τα στοιχεία των διαστάσεων υπάρχουν πολλές φορές στους πίνακες διαστάσεων (μια φορά για κάθε χρήση).

Ονοματολογία Πινάκων

Για διευκόλυνση έγινε προσπάθεια να διατηρηθούν τα ίδια ονόματα πινάκων και πεδίων (όπου αυτό ήταν δυνατό) αντικαθιστώντας το χαρακτήρα «-» της Cobol με τον χαρακτήρα «_».

Στην ονοματολογία των πεδίων, η τυποποίηση που ακολουθήθηκε γίνεται φανερή από την κατηγορία των χαρακτήρων που χρησιμοποιήθηκαν. Έτσι όταν το όνομα είναι μόνο με ΚΕΦΑΛΑΙΑ γράμματα τότε κατά κανόνα αυτό το πεδίο προέρχεται από κάποιο COBOL αρχείο, ενώ εάν υπάρχουν και μικρά γράμματα τότε πρόκειται για νέο πεδίο.

Η ονοματολογία όλων των mappings αρχίζει με το πρόθεμα MRBL_ και ακολουθεί ένας τετραψήφιος αριθμός που υποδεικνύει σε ποια ομάδα πινάκων ανήκει το εκάστοτε mapping (1=param files, 2=stg, 3=dat) και σε ποια σειρά εκτέλεσης αναφέρεται. Πχ. MRBL_2011_FCU02_STG ανήκει στο δεύτερο worklet και θα είναι το εντέκατο mapping που θα εκτελεστεί στην ομάδα των STG.

Επίσης ακολουθήθηκε η τακτική το τελευταίο συνθετικό στοιχείο του ονόματος των πεδίων να είναι το όνομα του πίνακα που τα περιέχει. Παραδείγματος χάρη, ο πινάκας FCU02 περιέχει πεδία όπως AFM_FCU02,CITY_FCU02.

Στην περίπτωση των προσωρινών πεδίων που διαβάζονται από το source αρχείο και δεν έχουν υποστεί κάποια επεξεργασία το τελευταίο συνθετικό στοιχείο είναι πάντα «TMP». Πχ. ORIO_CR_FCU02_TMP

Σχεδόν σε όλα τα Fact Tables των Star Σχημάτων, κάποια πεδία έχουν σαν τελευταίο συνθετικό το όνομα του πίνακα Διαστάσεων από τον οποίο προέρχονται (αναλλοίωτα) και όχι αυτόν που τα περιέχει.

Στην περίπτωση της Λογιστικής, τα δύο σύνολα αρχείων (Γενική και Αναλυτική Λογιστική) ενοποιήθηκαν σε ένα και έτσι το πρόθεμα του ονόματος των νέων πινάκων είναι FGLAL (ενοποίηση των προθεμάτων FGL και FAL).

Πίνακες Παραμέτρων

Λέγονται παραμετρικοί οι πίνακες γιατί περιέχουν ως επί το πλείστον περιγραφές των κωδικών. Οι παραμετρικοί πίνακες που υλοποιήθηκαν στην παρούσα εργασία είναι οι παρακάτω 44 πίνακες:

01FAX02_DAT	23 FCC01_DAT
02 FCU06_DAT	24 FGL06_DAT
03 FIN05_DAT	25 FLA04_DAT
04 FLAV1_DAT	25 FMA12_DAT
05 FMA13_DAT	27A FMA14_DAT
06 FMA30_DAT	27 FMI50_DAT
07 FMI51_DAT	28 FMI53_DAT
08 FMI54_DAT	29 FMI55_DAT
09 FMI56_DAT	30 FMI65_DAT
10 FMI66_DAT	31 FSM031_DAT
11 FSP04_DAT	32 FST04_DAT
12 FST20_DAT	33 FST20_1_DAT
13 FST20_2_DAT	34 FST21_DAT
14 FST24_DAT	35 FST29_DAT
15 FVA02_DAT	36 FVA04_DAT
16 FVA06_DAT	37 FVA08_DAT
17 FVA10_DAT	38 FVA12_DAT
18 FVA14_DAT	39 FVA16_DAT
19 FVA17_DAT	40 FVA18_DAT
20 FVA20_DAT	41 FVA24_DAT
21 FVA25_DAT	42 FVA28_DAT
22 FVA32_DAT	43 FVA41_DAT

Πίνακες Κυρίων Διαστάσεων (Strong dimensions)

Strong dimensions ονομάζονται οι πίνακες που περιγράφουν τις κύριες οντότητες του DWόπως Πελάτης, Αποθήκη, Εργαζόμενοι κλπ, δηλαδή τις πιο σημαντικές διαστάσεις. Οι σημαντικές διαστάσεις στο σύνολο τους είναι 7, και είναι οι εξής:

1.Σταθερά Στοιχεία Αποθήκης Εμπορευμάτων (FST01_DAT)

Αποτελεί ένα από τους πλέον σημαντικούς Πίνακες Διαστάσεων αφού είναι συνδεδεμένος με διάφορους Πίνακες Κινήσεων (Fact Tables) όπως της Αποθήκης Ετοίμων, Τιμολογίων Πελατών κλπ.

Πεδία Πίνακα	**Περιγραφή Πεδίου**
CUSTOMER_FST01	Περιγραφή Πελάτη Φασόν
Profit_Center_Code_FST01	Κέντρα Κέρδους
Costing_Sector_Code_FST01	Κοστολογικοί Τομείς
Costing_Division_Code_FST01	Κοστολογικοί Κλάδοι
Costing_Group_Code_FST01	Κοστολογικές Ομάδες
LATOMEIO_FST01	Λατομεία
PROELEYSH_OMADA_FST01	Ομάδες Προελεύσεων
PROELEYSH_FST01	Προελεύσεις
Costing_PROELEYSH_FST01	Κοστολογικές Ομάδες - Προέλευση
POIOTHTA_FST01	Ποιότητες
PAXOS_FST01	Πάχη
Costing_PAXOS_FST01	Κοστολογικές Ομάδες – Πάχος
SUPPL_FST01	Συνήθεις Προμηθευτές
MU_FST01	Μονάδες Μέτρησης Αποθήκης
WH1_FST01	Λογική Αποθήκη
Costing_WH1_FST01	Κοστολογικές Ομάδες – Λογική Αποθήκη
Material_Type_FST01	Τύποι Υλικού
WH2_FST01	Φυσική Αποθήκη
Costing_WH2_FST01	Κοστολογικές Ομάδες – Φυσική Αποθήκη
OMADA_FST01	Ομάδες Είδους
PRODUCT_FST01	Ομάδες Προϊόντων
Product_SubGroup_Code_FST01	Υπο-ομάδες Προϊόντων
Material_Type_FST01	Τύπος Υλικού.
PRODUCT_FST01	Ομάδα Προϊόντων.
Product_SubGroup_Code_FST01	Υπο-ομάδα Προϊόντων.
PROELEYSH_FST01	Προέλευση.
PAXOS_FST01	Πάχος.
Year_Started_FST01	Έτος Δημιουργίας Κωδικού. Αφορά μόνο Όγκους και Πλάκες.

Πεδία Πίνακα	Περιγραφή Πεδίου
Diastasi_FST01	Διαστάσεις Τυποποιημένων Πλακιδίων.
Idos_Hroma_FST01	Είδος και Χρώμα Αρμόστοκου (Marmoline).
Diametros_FST01	Διάμετρος Αρμόστοκου (Marmoline).
Profit_Center_Code_FST01	Κέντρο Κέρδους
Costing_WH1_FST01	Κοστολογική Λογική Αποθήκη
Costing_WH2_FST01	Κοστολογική Φυσική Αποθήκη
Costing_PAXOS_FST01	Κοστολογικό Πάχος.
Costing_PROELEYSH_FST01	Κοστολογική Προέλευση.
Costing_Sector_Code_FST01	Κοστολογικός Τομέας.
Costing_Division_Code_FST01	Κοστολογικός Κλάδος.
Costing_Group_Code_FST01	Κοστολογική Ομάδα.
Συνολική Ποσότητα Παραγωγής	LPP_FST01
Συνολική Ποσότητα Εισαγωγής	LPQ_FST01
Συνολική Διαθέσιμη Ποσότητα	AVAILABLE_QTY_FST01
Συνολική Παραχθείσα Ποσότητα Πλακών	NUM_PLAKES_FST01
Συνολική Διαθέσιμη Ποσότητα Πλακών	NUM_PLAKES_AVAIL_FST01

2.Σταθερά Στοιχεία Λογαριασμών (FGLAL02_DAT)

Στον πίνακα αυτό ενοποιήθηκαν τα δύο Λογιστικά Σχέδια (της Γενικής και της Αναλυτικής Λογιστικής). Υπάρχει ειδικό πεδίο το οποίο μπορεί να διαχωρίσει τα δύο Λογιστικά Σχέδια. Το πεδίο αυτό είναι το Ledger_Section_FGLAL02 και μπορεί να έχει μία από τις παρακάτω δύο τιμές:

- «Γ/Λ» όταν πρόκειται για Λογαριασμό Γενικής Λογιστικής
- «Α/Λ» όταν πρόκειται για Λογαριασμό Αναλυτικής Λογιστικής

Οι τιμές των παραπάνω δύο τιμών είναι με Ελληνικούς Χαρακτήρες.

Πεδία Πίνακα	Περιγραφή Πεδίου
Cost_Center_Code_FGLAL02	Κέντρα Κόστους
Profit_Center_Code_FGLAL02	Κέντρα Κέρδους
LATOMEIO_FGLAL02	Λατομεία
Costing_Sector_Code_FGLAL02	Κοστολογικοί Τομείς
Costing_Division_Code_FGLAL02	Κοστολογικοί Κλάδοι
Costing_Group_Code_FGLAL02	Κοστολογικές Ομάδες
Costing_WH1_FGLAL02	Κοστολογικές Ομάδες - Λογική Αποθήκη
Costing_WH2_FGLAL02	Κοστολογικές Ομάδες - Φυσική Αποθήκη
Costing_PROELEYSH_FGLAL02	Κοστολογικές Ομάδες – Προέλευση
Costing_PAXOS_FGLAL02	Κοστολογικές Ομάδες – Πάχος
MIS_Section_Code_FGLAL02	Τομείς Management Information System
MIS_Group_Code_FGLAL02	Ομάδες Management Information System

Πεδία Πίνακα	Περιγραφή Πεδίου
MIS_RowGroup_Code_FGLAL02	Ομάδες Management Information System
MIS_ColumnGroup_Code_FGLAL02	Ομάδες Management Information System
MIS_Category_Code_FGLAL02	Κατηγορίες Management Information System
MIS_Category_RowGroup_Code_FGLAL02	Κατηγορίες Management Information System
MIS_Category_ColumnGroup_Code_FGLAL02	Κατηγορίες Management Information System
MIS_Subcategory_Code_FGLAL02	Υποκατηγορίες Management Information System
MIS_Detail_Code_FGLAL02	ΑνάλυσηΥποκατηγορίας Management Information System
ACCNO_01_FGLAL02	Σταθερά Στοιχεία Λογαριασμών Γενικής και Αναλυτικής Λογιστικής.
ACCNO_02_FGLAL02	Σταθερά Στοιχεία Λογαριασμών Γενικής και Αναλυτικής Λογιστικής.
ACCNO_03_FGLAL02	Σταθερά Στοιχεία Λογαριασμών Γενικής και Αναλυτικής Λογιστικής.
ACCNO_04_FGLAL02	Σταθερά Στοιχεία Λογαριασμών Γενικής και Αναλυτικής Λογιστικής.
ACCNO_05_FGLAL02	Σταθερά Στοιχεία Λογαριασμών Γενικής και Αναλυτικής Λογιστικής.
ACCNO_06_FGLAL02	Σταθερά Στοιχεία Λογαριασμών Γενικής και Αναλυτικής Λογιστικής.
Cost_Center_Code_FGLAL02	Κέντρο Κόστους.
Profit_Center_Code_FGLAL02	Κέντρο Κέρδους
LATOMEIO_FGLAL02	Λατομείο
Costing_WH1_FGLAL02	Κοστολογική Λογική Αποθήκη
Costing_WH2_FGLAL02	Κοστολογική Φυσική Αποθήκη
Costing_PAXOS_FGLAL02	Κοστολογικό Πάχος.
Costing_PROELEYSH_FGLAL02	Κοστολογική Προέλευση.
Costing_Sector_Code_FGLAL02	Κοστολογικός Τομέας.
Costing_Division_Code_FGLAL02	Κοστολογικός Κλάδος.
Costing_Group_Code_FGLAL02	Κοστολογική Ομάδα.
MIS_Section_Code_FGLAL02	Τομέας MIS
MIS_Group_Code_FGLAL02	Ομάδα MIS
MIS_RowGroup_Code_FGLAL02	Κάθετη Ομάδα MIS
MIS_ColumnGroup_Code_FGLAL02	Οριζόντια Ομάδα MIS
MIS_Category_Code_FGLAL02	Κατηγορία MIS
MIS_Category_RowGroup_Code_FGLAL02	Κάθετη Κατηγορία MIS
MIS_Category_ColumnGroup_Code_FGLAL02	Οριζόντια Κατηγορία MIS
MIS_Subcategory_Code_FGLAL02	Υποκατηγορία MIS
MIS_Detail_Code_FGLAL02	Ανάλυση Υποκατηγορίας MIS
ACCNO_01_FGLAL02	Πρωτοβάθμιος Λογαριασμός

Πεδία Πίνακα	Περιγραφή Πεδίου
ACCNO_02_FGLAL02	Δευτεροβάθμιος Λογαριασμός
ACCNO_03_FGLAL02	Τριτοβάθμιος Λογαριασμός
ACCNO_04_FGLAL02	Τεταρτοβάθμιος Λογαριασμός
ACCNO_05_FGLAL02	Πεμπτοβάθμιος Λογαριασμός
ACCNO_06_FGLAL02	Εκτοβάθμιος Λογαριασμός
Sales_Department_FCU02	Τμήματα Πωλήσεων
COUNTRY_FCU02	Χώρες
AREA_FCU02	Περιοχές
SUBAREA_FCU02	Υποπεριοχές
COUNTRY_DELIVERY_FCU02	Χώρες Παράδοσης
DROMOLOGIO_FCU02	Δρομολόγια
DOY_FCU02	ΔΥΟ
KATEGORY_1_FCU02	Κατηγορία Πελάτη 1
KATEGORY_2_FCU02	Κατηγορία Πελάτη 2
SALESMAN_CODE_FCU02	Πωλητές

3.Σταθερά Στοιχεία Πελατών (FCU02_DAT)

Σε αυτόν τον πίνακα αποθηκεύονται όλες οι πληροφορίες που χρειάζεται να ξέρει η εταιρία για τους πελάτες της.

Πεδία Πίνακα	Περιγραφή Πεδίου
Sales_Department_FCU02	Τμήματα Πωλήσεων
COUNTRY_FCU02	Χώρες
AREA_FCU02	Περιοχές
SUBAREA_FCU02	Υποπεριοχές
COUNTRY_DELIVERY_FCU02	Χώρες Παράδοσης
DROMOLOGIO_FCU02	Δρομολόγια
DOY_FCU02	ΔΥΟ
KATEGORY_1_FCU02	Κατηγορία Πελάτη 1
KATEGORY_2_FCU02	Κατηγορία Πελάτη 2
SALESMAN_CODE_FCU02	Πωλητές
Sales_Department_FCU02	Τμήμα Πωλήσεων.
Market_Segment_FCU02	Τομέας Πωλήσεων. Διακριτικό πελατών Εσωτερικού και Εξωτερικού.
WH1_FSP01	Λογικές Αποθήκες
WH2_FSP01	Φυσικές Αποθήκες
MU_FSP01	Μονάδες Μέτρησης

4.Σταθερά Στοιχεία Αποθήκης Ανταλλακτικών (FSP01_DAT)

Σε αυτόν τον πίνακα κρατιούνται πληροφορίες σχετικές με τις αποθήκες. Υπάρχουν δύο ειδών αποθήκες: λογικές αποθήκες και φυσικές αποθήκες. Οι φυσικές αποθήκες είναι οι αποθήκες με φυσική υπόσταση δηλαδή είναι ένας χώρος που έχει συγκεκριμένη διεύθυνση. Οι λογικές αποθήκες δημιουργηθήκαν για ανάγκη ομαδοποίησης, μέσα στον ίδιο φυσικό χώρο μπορεί να υπάρχουν δυο ή περισσότερες λογικές αποθήκες. Με άλλα λόγια οι αποθήκες αυτές έχουν περισσότερο λογική υπόσταση και βρίσκονται μέσα στις φυσικές αποθήκες.

Πεδία Πίνακα	**Περιγραφή Πεδίου**
WH1_FSP01	Λογικές Αποθήκες
WH2_FSP01	Φυσικές Αποθήκες
MU_FSP01	Μονάδες Μέτρησης

5.Σταθερά Στοιχεία Προμηθευτών (FSU02_DAT)

Σε αυτόν τον πίνακα θα βρούμε όλες τις βασικές πληροφορίες σχετικά με τους προμηθευτές που σχετίζεται η εταιρία.

Πεδία Πίνακα	**Περιγραφή Πεδίου**
COUNTRY_FSU02	Χώρες
AREA_FSU02	Περιοχές
SUBAREA_FSU02	Υποπεριοχές
KATEGORY_FSU02	Επαγγέλματα
CUSTOMER_FSU02	Προμηθευτές - Πελάτες
DOY_FSU02	ΔΟΥ

6.Σταθερά Στοιχεία Εργαζομένων (FMI01_DAT)

Σε αυτόν τον πίνακα κρατιούνται δεδομένα σχετικά με τους εργαζόμενους της εταιρίας.

Πεδία Πίνακα	**Περιγραφή Πεδίου**
KATEGORY_FMI01	Κατηγορίες Εργαζομένων
LOCATION_FMI01	Τόποι Πληρωμής
DIRECTION_FMI01	Διοικητικές Διευθύνσεις
DEPARTMENT_FMI01	Διοικητικά Τμήματα
KENTRO_KOSTOUS_FMI01	Κέντρα Κόστους (Μισθοδοσία)
PROFESSION_FMI01	Ειδικότητες
EDUCATION_FMI01	Σπουδές
CUSTOMER_FMI01	Περιγραφή Εργαζομένου στο Αρχείο Πελατών

7.Σταθερά Στοιχεία Μηχανημάτων (FMA02_DAT)

Σε αυτόν τον πίνακα κρατιούνται δεδομένα σχετικά με τα μηχανήματα που χρησιμοποιούνται στην παραγωγή.

Πεδία Πίνακα	Περιγραφή Πεδίου
MODEL_FMA02	Μοντέλα
FACTORY_FMA02	Εργοστάσια Παραγωγής
SUPPLIER_FMA02	Προμηθευτές
COLOR_FMA02	Χρώματα
OMADA_FMA02	Ομάδες

Άλλοι πίνακες Διαστάσεων

Παρακάτω αναφέρονται οι υπόλοιποι πίνακες διαστάσεων που δεν εντάσσονται όμως στο σύνολο των κυρίων διαστάσεων.

1. Στοιχεία Αξιόγραφων (FAX01_DAT)
2. Στοιχεία Λογαριασμών ανά Λατομείο (FCCST_DAT)
3. Κατηγοριοποίηση Κωδικών Κίνησης Αποθήκης Ανταλλακτικών(FSP05_DAT)
4. Στατιστικά Πωλήσεων – Αρχείο Πωλήσεων ανά Έτος Μηχανί και Πελάτη (FSS80_DAT)
5. Στατιστικά Πωλήσεων – Αρχείο με Υπόλοιπα και Επιταγές (FSS81_DAT)
6. Κατηγοριοποίηση Κωδικών Κίνησης Αποθήκης Ανταλλακτικών (FST05_DAT)
7. Λογιστική Κινήσεων Αποθήκης (FSTAL1_DAT)
8. Λογιστική Κινήσεων Πίστωσης (FSTAL1_CR_DAT)
9. Λογιστική Κινήσεων Χρέωσης (FSTAL1_DR_DAT)
10. Κοστολογικά Στοιχεία Αποθήκης Ετοίμων (FCCAVE_DAT)
11. Κοστολογικά Στοιχεία Ανταλλακτικών (FSPAVE_DAT)

Πίνακες Κινήσεων (Fact Tables)

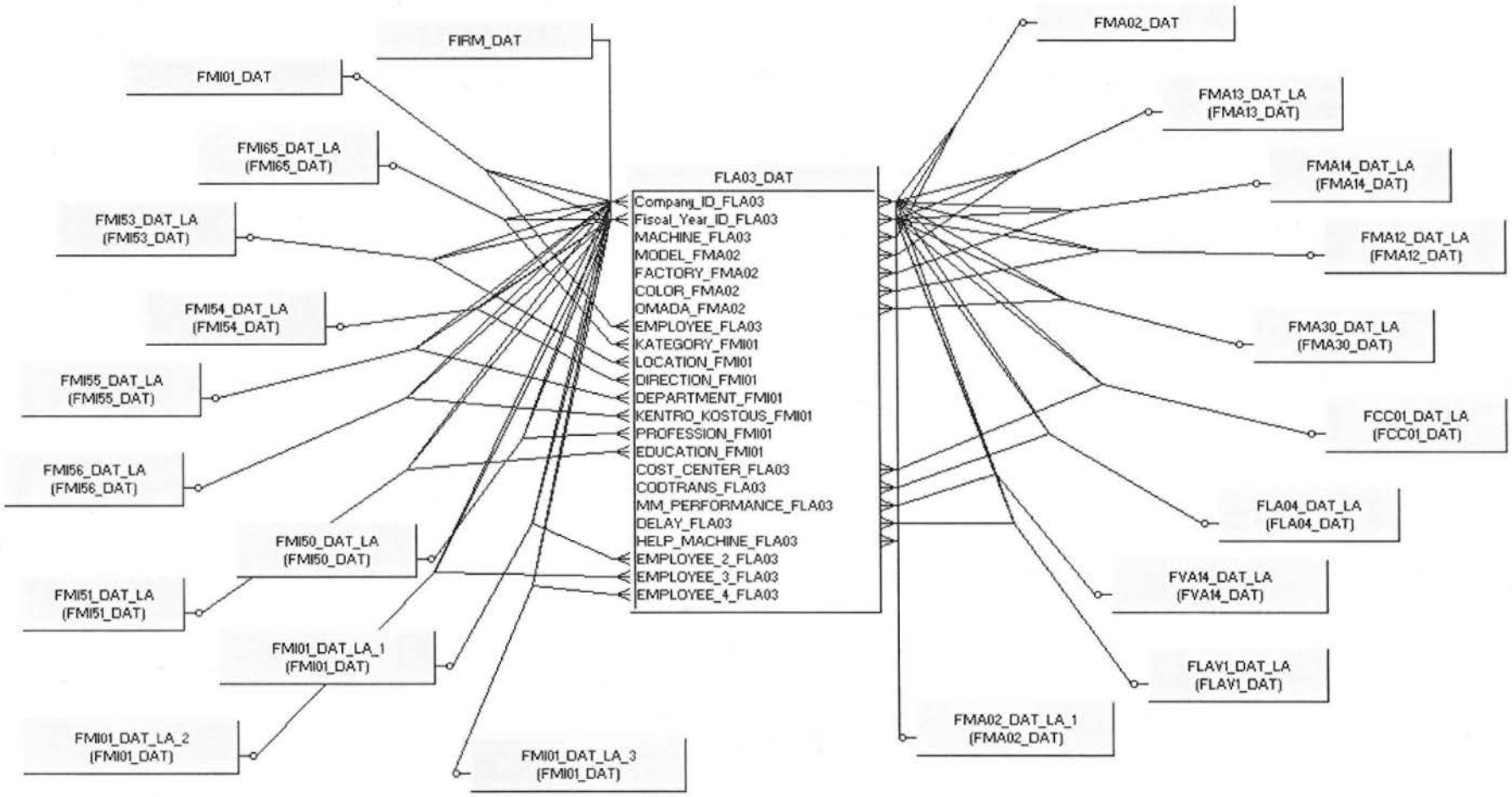
Όλοι οι fact πίνακες είναι οι κεντρικοί πίνακες των star σχημάτων και αναφέρονται στις κινήσεις που έχουν γίνει για την συγκεκριμένη οντότητα. Ακολουθούν αναλυτικά οι περιγραφές των πινάκων και διαγράμματα οντοτήτων που δείχνουν σχεδιαστικά την συσχέτιση των πινάκων μεταξύ τους. Το σύνολο των πινάκων κινήσεων είναι 12.

1.Κινήσεις Παραγωγής (FLA03_DAT)

Στον πίνακα FLA03_DATκαταγράφονται όλες οι κινήσεις της παραγωγής. Οι κύριες διαστάσεις με τις οποίες συνδέεται αυτός ο factπίνακας είναι Employee_fla03 και Machine_fla03 (σταθερά στοιχεία εργαζομένων και σταθερά στοιχεία μηχανημάτων).

Πεδία Πίνακα	**Περιγραφή Πεδίου**
EMPLOYEE_FLA03	Σταθερά Στοιχεία Εργαζομένων
MACHINE_FLA03	Σταθερά Στοιχεία Μηχανημάτων
COST_CENTER_FLA03	Κέντρα Κόστους
CODTRANS_FLA03	Κωδικοί Κίνησης Παραγωγής
MM_PERFORMANCE_FLA03	Μονάδες Μέτρησης Αποθήκης
DELAY_FLA03	Καθυστερήσεις Παραγωγής
HELP_MACHINE_FLA03	Βοηθητικά Μηχανήματα
EMPLOYEE_2_FLA03	Ονοματεπώνυμο Συμπληρωματικών Εργαζομένων
EMPLOYEE_3_FLA03	
EMPLOYEE_4_FLA03	
PROFESSION_FMI01	Ειδικότητες
EDUCATION_FMI01	Σπουδές
LOCATION_FMI01	Τόποι Πληρωμής
DIRECTION_FMI01	Διευθύνσεις
DEPARTMENT_FMI01	Τμήματα
KENTRO_KOSTOUS_FMI01	Κέντρα Κόστους Μισθοδοσίας
KATEGORY_FMI01	Κατηγορίες Εργαζομένων
COLOR_FMA02	Χρώματα Μηχανημάτων
MODEL_FMA02	Μοντέλα Μηχανημάτων
FACTORY_FMA02	Εργοστάσια Κατασκευής Μηχανημάτων
OMADA_FMA02	Ομάδα Σύνθεσης Μηχανημάτων
Minute_Duration_FLA03	Διάρκεια Παραγωγής
PERFORMANCE_FLA03	Απόδοση

Το πεδίο Minute_Duration_FLA03 (Διάρκεια Παραγωγής) είναι η διαφορά σε λεπτά, της Ώρας Έναρξης από την Ώρα Λήξης. Τα πεδία START_TIME_FLA03 και FINISH_TIME_FLA03 είναι τύπου DateTime. Επειδή στα COBOL αρχεία υπήρχε μόνο η πληροφορία της ώρας (ώρες και λεπτά μόνο (χωρίς δευτερόλεπτα)), για την δημιουργία των DateTime πεδίων χρησιμοποιήθηκε η αντίστοιχη ημερομηνία κίνησης.



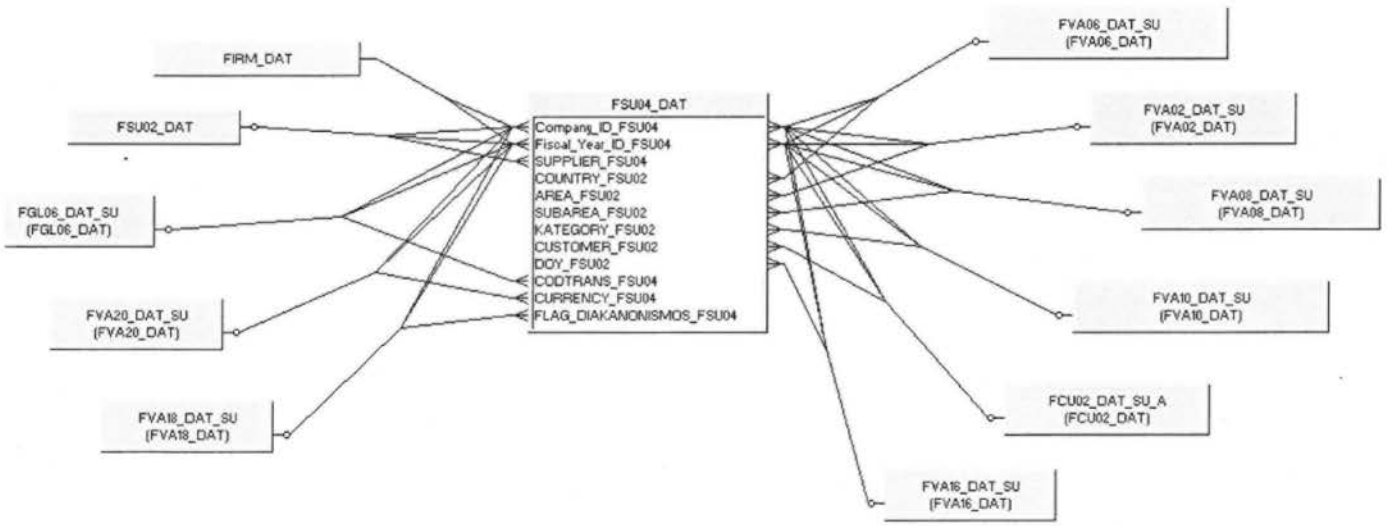
Entity Diagram of FLA03_DAT

2.Κινήσεις Προμηθευτών (FSU04_DAT)

Στον πίνακα FSU04_DAT καταγράφονται όλες οι κινήσεις των παραγωγής. Η κύρια διάσταση με την οποία συνδέεται αυτός ο fact πίνακας είναι Supplier_fsu04 (σταθερά στοιχεία προμηθευτών).

Πεδία Πίνακα	**Περιγραφή Πεδίου**
SUPPLIER_FSU04	Σταθερά Στοιχεία Προμηθευτών
CODTRANS_FSU04	Αιτίες Κίνησης Λογιστικής
FLAG_DIAKANONISMOS_FSU04	Διακανονισμοί
CURRENCY_FSU04	Νομίσματα Συναλλαγής
CUSTOMER_FSU02	Πελάτες (Εάν ο Προμηθευτής είναι και Πελάτης)
AREA_FSU02	Περιοχές Προμηθευτή
COUNTRY_FSU02	Χώρες Προμηθευτή
SUBAREA_FSU02	Υποπεριοχές Προμηθευτή
KATEGORY_FSU02	Επάγγελμα Προμηθευτή
DOY_FSU02	Δ.Ο.Υ. Προμηθευτή
Db_Cr_FC_FSU04	Υπόλοιπο σε Ξένο Νόμισμα

Παρακάτω ακολουθεί το διάγραμμα οντοτήτων για τον πίνακα FSU04_DAT και όσους πίνακες συνδέονται μαζί του σχηματίζοντας ένα StarΣχήμα

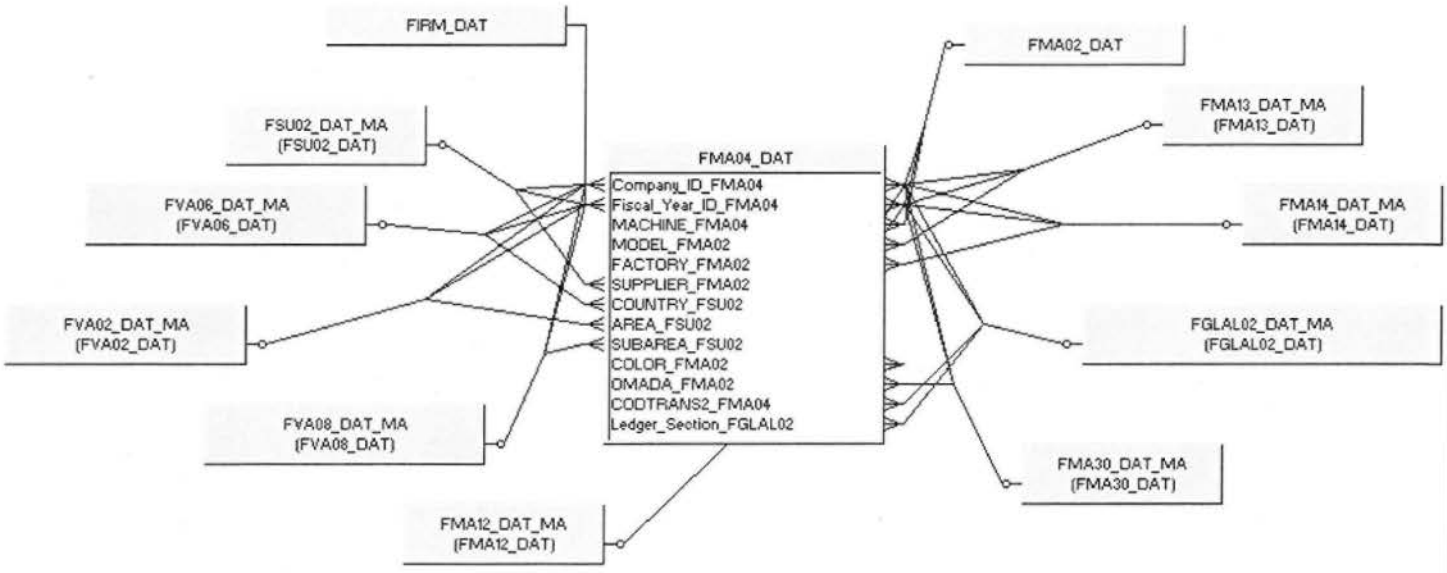


tityDiagramofFSU04_DAT

3.ΚινήσειςΜηχανημάτων (FMA04_DAT)

Στον πίνακα FMA04_DAT καταγράφονται όλες οι κινήσεις των μηχανημάτων που χρησιμοποιούνται στην παραγωγή μαρμάρων. Η κύριες διαστάσεις με την οποίες συνδέεται αυτός ο fact πίνακας είναι Machine_fma04 και Supplier_fma02 (σταθερά στοιχεία μηχανημάτων και σταθερά στοιχεία προμηθευτών).

Πεδία Πίνακα	**Περιγραφή Πεδίου**
MACHINE_FMA04	Σταθερά Στοιχεία Μηχανημάτων
SUPPLIER_FMA02	Σταθερά Στοιχεία Προμηθευτών
CODTRANS2_FMA04	Λογαριασμοί
COUNTRY_FSU02	Χώρες Προμηθευτών
AREA_FSU02	Περιοχές Προμηθευτών
SUBAREA_FSU02	Υποπεριοχές Προμηθευτών
COLOR_FMA02	Χρώματα Μηχανημάτων
MODEL_FMA02	Μοντέλα Μηχανημάτων
FACTORY_FMA02	Εργοστάσια Κατασκευής Μηχανημάτων
OMADA_FMA02	Ομάδα Σύνθεσης Μηχανημάτων
Balance_FMA04	Υπόλοιπο Κινήσεων Μηχανημάτων



EntityDiagramofFMA04_DAT

4.Κινήσεις Αγορών (FSI0102_DAT)

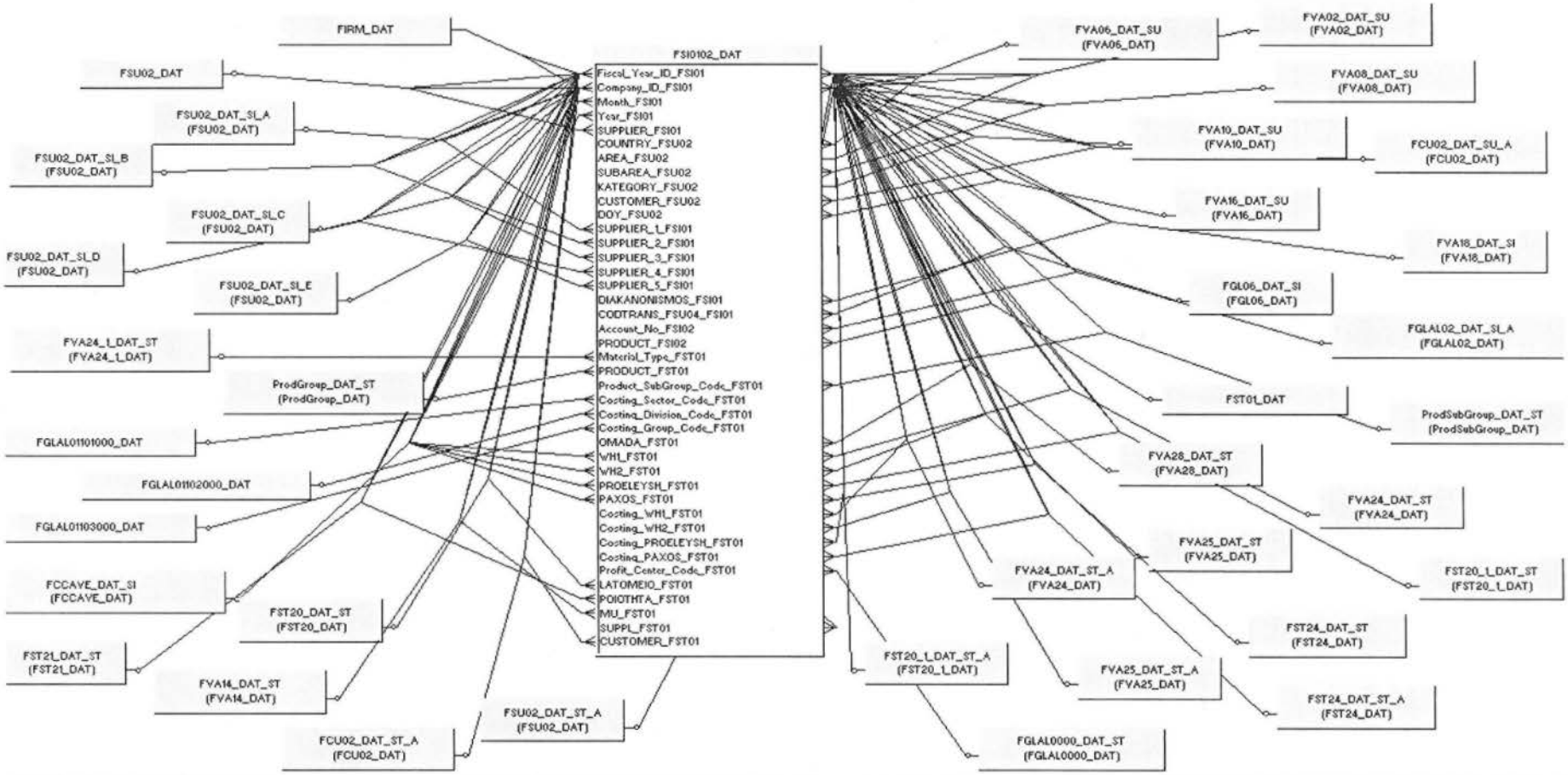
Στον πίνακα FSI0102_DAT καταγράφονται οι Κινήσεις Αγορών, τι εμπορεύματα έχουν αγοραστεί από ποιόν προμηθευτή. Η κύριες διαστάσεις με την οποίες συνδέεται ο fact πίνακας είναι Product_fsi02και Supplier_fsu04 (σταθερά στοιχεία εμπορευμάτων και σταθερά στοιχεία προμηθευτών).

Πεδία Πίνακα	**Περιγραφή Πεδίου**
PRODUCT_FSI02	Σταθερά Στοιχεία Εμπορευμάτων
SUPPLIER_FSU04	Σταθερά Στοιχεία Προμηθευτών
SUPPLIER_1_FSI01	Άλλος Προμηθευτής 1
SUPPLIER_2_FSI01	Άλλος Προμηθευτής 2
SUPPLIER_3_FSI01	Άλλος Προμηθευτής 3
SUPPLIER_4_FSI01	Άλλος Προμηθευτής 4
SUPPLIER_5_FSI01	Άλλος Προμηθευτής 5
DIAKANONISMOS_FSI01	Διακανονισμός
Account_No_FSI02	Λογαριασμός Γενικής Λογιστικής
CODTRANS_FSU04_FSI01	Αιτία Κίνησης Λογιστικής
CURRENCY_FSU04	Νομίσματα Συναλλαγής
Costing_Sector_Code_FST01	Κοστολογικός Τομέας
Costing_Division_Code_FST01	Κοστολογικός Κλάδος
Costing_Group_Code_FST01	Κοστολογική Ομάδα
WH1_FST01, WH2_FST01, PROELEYSH_FST01, PAXOS_FST01	Μέση τιμή κόστους Μηνός
LATOMEIO_FST01	Λατομείο
POIOTHTA_FST01	Ποιότητα
MU_FST01	Μονάδα Μέτρησης Αποθήκης
SUPPL_FST01	Συνήθης Προμηθευτής
CUSTOMER_FST01	Περιγραφή Πελάτη Φασόν
COUNTRY_FSU02	Χώρα
AREA_FSU02	Περιοχή
SUBAREA_FSU02	Υποπεριοχή
KATEGORY_FSU02	Επάγγελμα Προμηθευτή
CUSTOMER_FSU02	Επωνυμία Προμηθευτή – Πελάτη
DOY_FSU02	ΔΥΟ
Material_Type_FST01	Τύπος Υλικού
PRODUCT_FST01	Ομάδα Προϊόντων
OMADA_FST01	Ομάδα Είδους
WH1_FST01	Λογική Αποθήκη
WH2_FST01	Φυσική Αποθήκη
PROELEYSH_OMADA_FST01	Ομάδες Προελεύσεων
PROELEYSH_FST01	Προέλευση
PAXOS_FST01	Πάχος

Costing_WH1_FST01	Κοστολ.Ομάδα - Λογική Αποθήκη
Costing_WH2_FST01	Κοστολ.Ομάδα - Φυσική Αποθήκη
Costing_PROELEYSH_FST01	Κοστολ.Ομάδα - Προέλευση
Costing_PAXOS_FST01	Κοστολ.Ομάδα – Πάχος
Profit_Center_Code_FST01	Κέντρο Κέρδους
Section_Ledger_FSI02	Λογιστικός Τομέας. Διάκριση λογαριασμών Αναλυτικής και Γενικής Λογιστικής.
Account_No_FSI02	Αριθμός Λογαριασμού (όταν δεν αφορά Προϊόν (παρακάτω)).
PRODUCT_FSI02	Κωδικός Προϊόντος (όταν δεν αφορά Λογαριασμό (παραπάνω)).

Στο πεδίο PRODUCT_FSI02 ο διαχωρισμός Λογαριασμός/Προϊόν προκύπτει από συγκεκριμένες τιμές του πεδίου WH_FSI02_TMP το οποίο δεν μεταφέρεται στον τελικό πίνακα.

Παρακάτω ακολουθεί το διάγραμμα οντοτήτων για τον πίνακα FSI0102_DAT και όσους πίνακες συνδέονται μαζί του σχηματίζοντας ένα StarΣχήμα



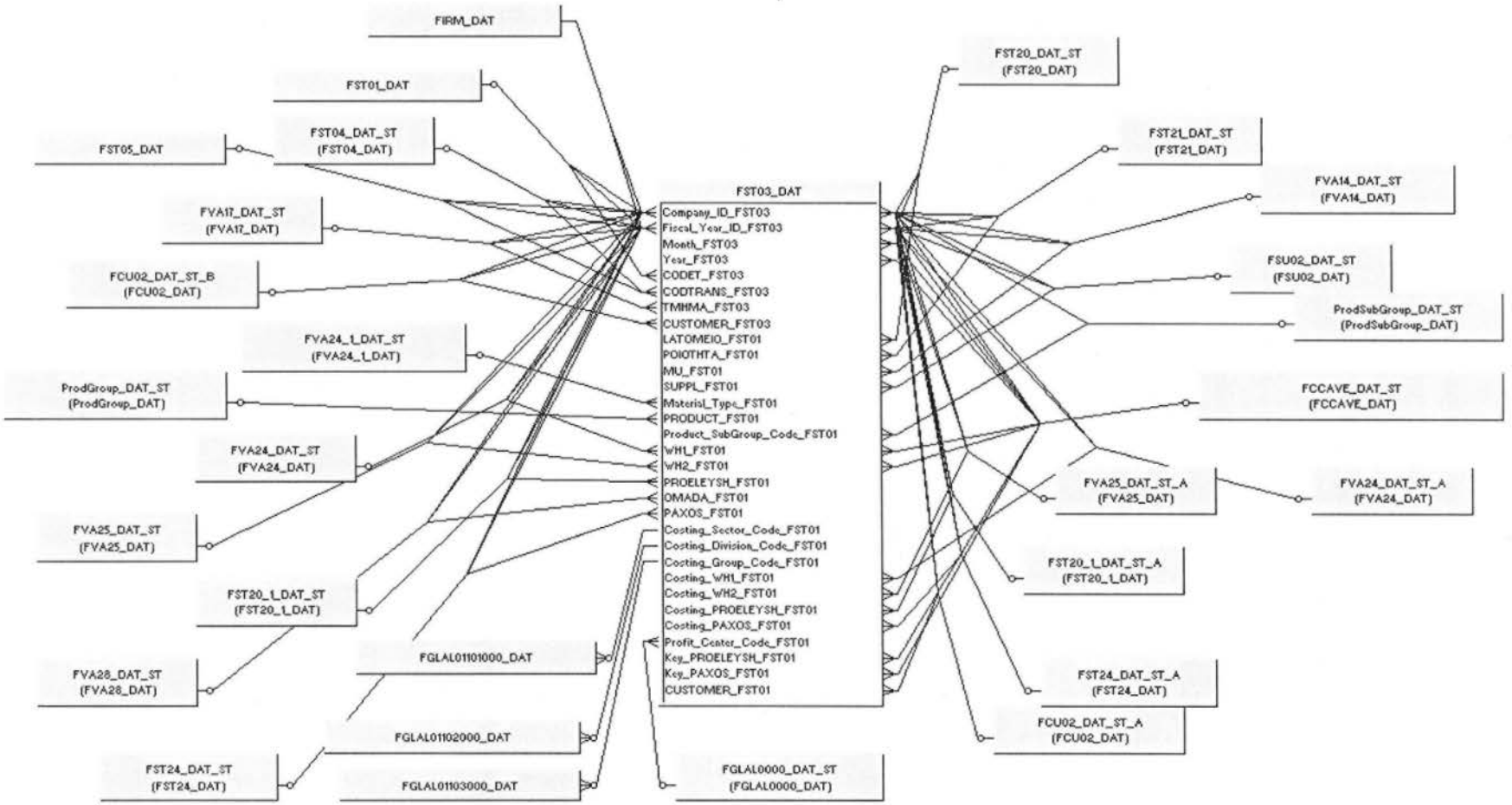
EntityDiagramofFSI0102_DAT

5.Κινήσεις Αποθήκης Εμπορευμάτων (FST03_DAT)

Στον πίνακα FST03_DAT καταγράφονται οι Κινήσεις Αποθήκης Εμπορευμάτων. Η κύριες διαστάσεις με την οποίες συνδέεται ο factπίνακας είναι Customer_fst03 και Codet_fst03 (αρχεία πελατών και master αποθήκης ειδών).

Πεδία Πίνακα	**Περιγραφή Πεδίου**
CUSTOMER_FST03	Αρχείο Πελατών
CODET_FST03	Master Αποθήκης Ειδών
CODTRANS_FST03	Κατηγοριοποίηση Κωδικών Κίνησης Αποθήκης
TMHMA_FST03	Αρχείο Τμημάτων
Costing_Sector_Code_FST01	Κοστολογικός Τομέας
Costing_Division_Code_FST01	Κοστολογικός Κλάδος
Costing_Group_Code_FST01	Κοστολογική Ομάδα
Profit_Center_Code_FST01	Κέντρο Κέρδους
LATOMEIO_FST01	Αρχείο Προελεύσεων και Λατομείων
PROELEYSH_OMADA_FST01	Ομάδες Προελεύσεων
PROELEYSH_FST01	Προελεύσεις
Costing_PROELEYSH_FST01	Κοστολ.Ομάδα – Προέλευση
POIOTHTA_FST01	Αρχείο Ποιοτήτων με Βάση τη Λογική Αποθήκη
PAXOS_FST01	Αρχείο Περιγραφής Πάχους με Βάση τη Λογική Αποθήκη
MU_FST01	Μονάδες Μέτρησης Αποθήκης
WH1_FST01	Λογική Αποθήκη
WH2_FST01	Φυσική Αποθήκη
SUPPL_FST01	Αρχείο Προμηθευτών
PRODUCT_FST01	Ομάδες Προϊόντων
OMADA_FST01	Αρχείο Ομάδων Ειδών Αποθήκης
QTY_FST03	Ποσότητα
AXIA_FST03	Αξία
NUM_PLAKES_FST03	Τεμάχια σε Μέτρα

Παρακάτω ακολουθεί το διάγραμμα οντοτήτων για τον πίνακα FST03_DAT και όσους πίνακες συνδέονται μαζί του σχηματίζοντας ένα StarΣχήμα



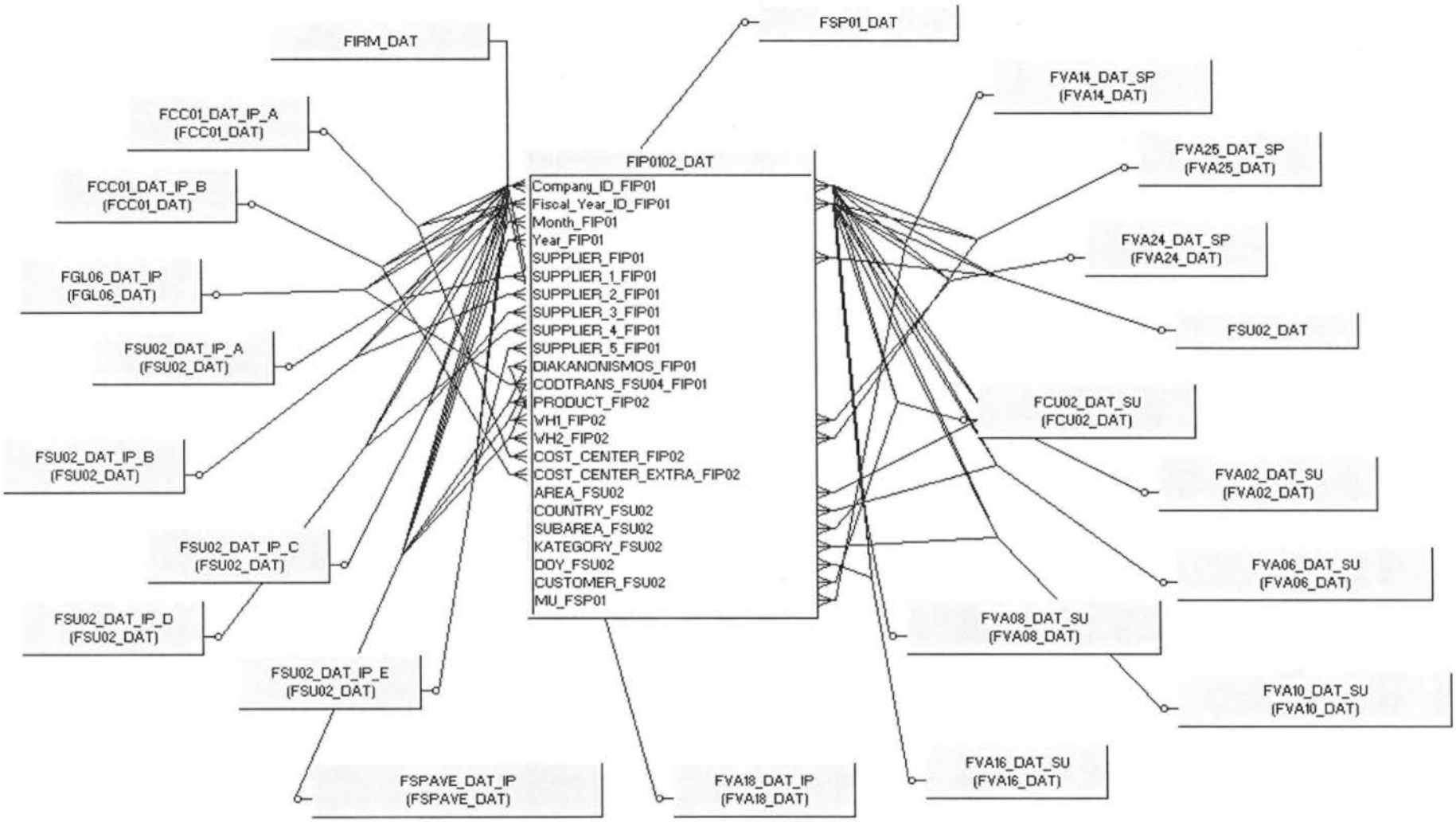
EntityDiagramofFST03_DAT

6.ΤιμολόγιαΑγορών Ανταλλακτικών (FIP0102_DAT)

Στον πίνακα FIP0102_DAT καταγράφονται τα τιμολόγια που έχουν γίνει για την αγορά ανταλλακτικών για τα μηχανήματα που χρησιμοποιούνται στο εργοστάσιο. Η κύριες διαστάσεις με την οποίες συνδέεται ο factπίνακας είναι PROD_FSP01, WH1_FSP01, WH2_FSP01 (masteraποθήκης ανταλλακτικών).

Πεδία Πίνακα	**Περιγραφή Πεδίου**
PROD_FSP01 and WH1_FSP01 and WH2_FSP01	Master Αποθήκης Ανταλλακτικών
SUPPLIER_1_FIP01	Αρχείο Προμηθευτών
DIAKANONISMOS_FIP01	Αρχείο Διακανονισμών
CODTRANS_FSU04_FIP01	Κωδικοί Κίνησης Λογιστικής
WH1_FIP02	Λογική Αποθήκη
WH2_FIP02	Φυσική Αποθήκη
COST_CENTER_FIP02	Κέντρα Κόστους
PRODUCT_FIP02 and WH1_FIP02 and WH2_FIP02 and Year_FIP01 and Month_FIP01	Κοστολόγηση Ανταλλακτικών - Παραμετρικό Αρχείο Μέσων Τιμών
AREA_FSU02	Αρχείο Περιοχών
COUNTRY_FSU02	Αρχείο Χωρών
SUBAREA_FSU02	Αρχείο Υποπεριοχών
KATEGORY_FSU02	Αρχείο Κατηγορίας 1
MU_FSP01	Μονάδες Μέτρησης Αποθήκης
DOY_FSU02	Αρχείο Δ.Ο.Υ.
CUSTOMER_FSU02	Αρχείο Πελατών

Παρακάτω ακολουθεί το διάγραμμα οντοτήτων για τον πίνακα FIP0102_DAT και όσους πίνακες συνδέονται μαζί του σχηματίζοντας ένα StarΣχήμα



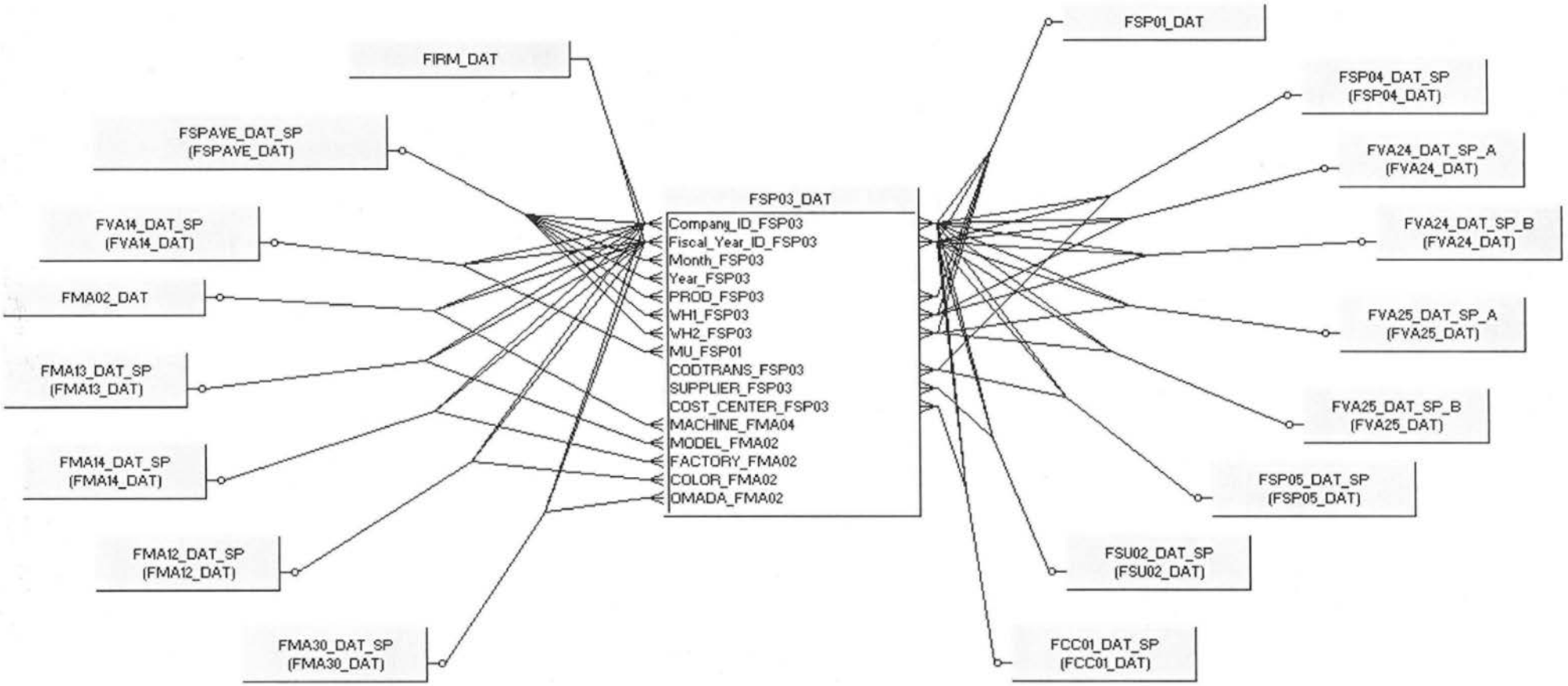
EntityDiagramofFIP0102_DAT

7.Κινήσεις Ανταλλακτικών (FSP03_DAT)

Στον πίνακα FST03_DAT καταγράφονται οι κινήσεις ανταλλακτικών για τα μηχανήματα που χρησιμοποιούνται στο εργοστάσιο. Η κύριες διαστάσεις με την οποίες συνδέεται ο factπίνακας είναι PROD_FSP01, WH1_FSP01, WH2_FSP01 (master αποθήκης ανταλλακτικών).

Πεδία Πίνακα	**Περιγραφή Πεδίου**
PROD_FSP01 and WH1_FSP01 and WH2_FSP01	Master Αποθήκης Ανταλλακτικών
WH1_FSP03	Λογικές Αποθήκες
WH2_FSP03	Φυσικές Αποθήκες
CODTRANS_FSP03	Κατηγοριοποίηση Κωδικών Κίνησης Αποθήκης Ανταλλακτικών
SUPPLIER_FSP03	Αρχείο Προμηθευτών
COST_CENTER_FSP03	Κέντρα Κόστους
PROD_FSP03 and WH1_FSP03 and WH2_FSP03 and Year_FSP03 and Month_FSP03	Κοστολόγηση Ανταλλακτικών - Παραμετρικό Αρχείο Μέσων Τιμών
MU_FSP01	Μονάδες Μέτρησης Αποθήκης
MACHINE_FMA04	Αρχείο Μηχανημάτων
COLOR_FMA02	Αρχείο Χρωμάτων Μηχανημάτων
MODEL_FMA02	Αρχείο Μοντέλων Μηχανημάτων
FACTORY_FMA02	Αρχείο Εργοστασίων Κατασκευής Μηχανημάτων
OMADA_FMA02	Ομάδα Σύνθεσης Μηχανημάτων
COST_CENTER_FSP03	Κέντρο Κόστους
MACHINE_FMA04	Κωδικός Μηχανής

Παρακάτω ακολουθεί το διάγραμμα οντοτήτων για τον πίνακα FSP03_DAT και όσους πίνακες συνδέονται μαζί του σχηματίζοντας ένα StarΣχήμα

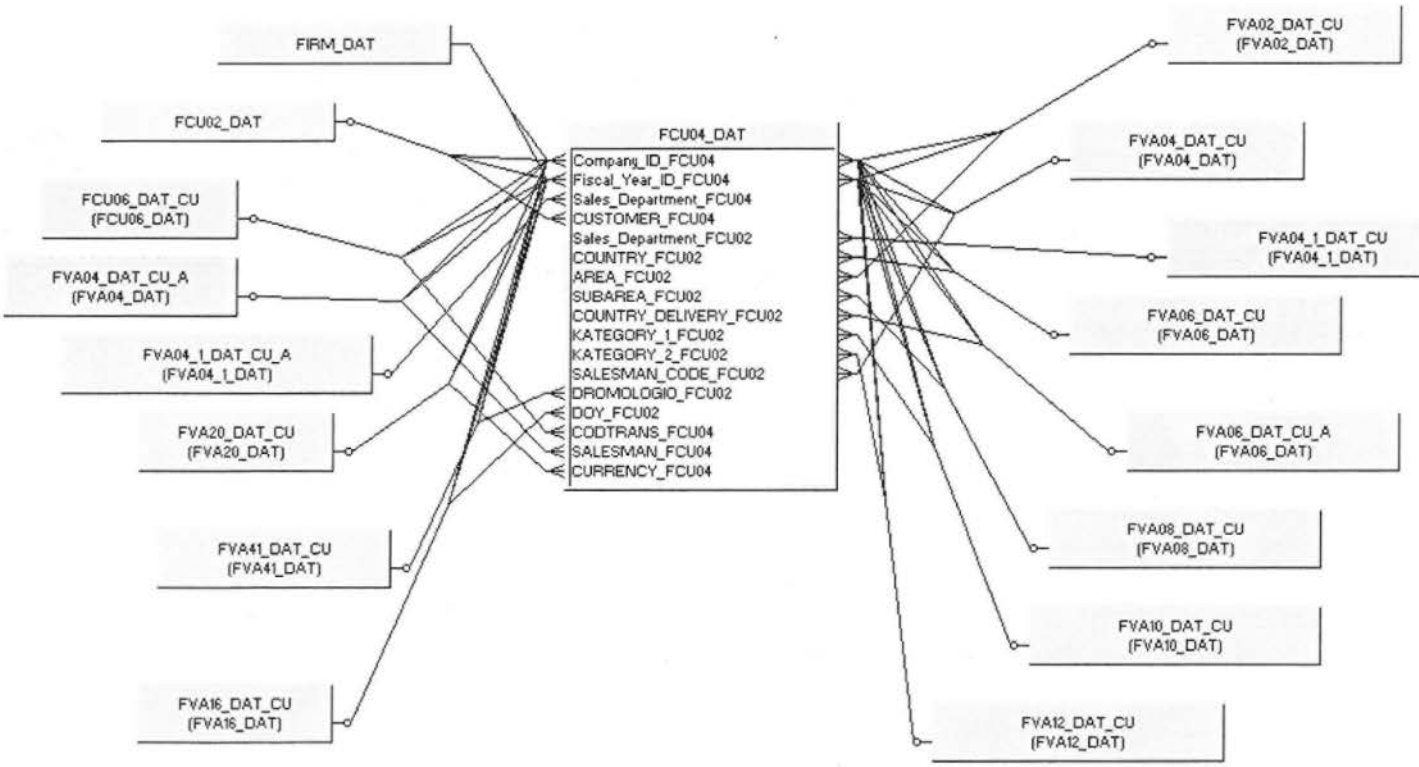


EntityDiagramofFSP03_DAT

8.Κινήσεις Πελατών (FCU04_DAT)

Στον πίνακα FCU04_DAT καταγράφονται όλες οι κινήσεις των πελατών. Η κύρια διάσταση του πίνακα είναι Customer_fcu04 όπου καταγράφονται αναλυτικά όλες οι πληροφορίες σχετικά με τους πελάτες.

Πεδία Πίνακα	**Περιγραφή Πεδίου**
CUSTOMER_FCU04	Αρχείο Πελατών
CODTRANS_FCU04	Κωδικοί Κίνησης Πελατών
SALESMAN_FCU04	Αρχείο Πωλητών
CURRENCY_FCU04	Αρχείο Νομισμάτων
Sales_Department_FCU04	Ιστορικό Τμήμα Πωλήσεων
AREA_FCU02	Αρχείο Περιοχών
Sales_Department_FCU02	Τμήμα Πωλήσεων
COUNTRY_FCU02	Αρχείο Χωρών
SUBAREA_FCU02	Αρχείο Υποπεριοχών
KATEGORY_1_FCU02	Αρχείο Κατηγορίας 1
KATEGORY_2_FCU02	Αρχείο Κατηγορίας 2
DOY_FCU02	Αρχείο Δ.Ο.Υ.
DROMOLOGIO_FCU02	Αρχείο Δρομολογίων
Sales_Department_FCU04	Τμήμα Πωλήσεων



EntityDiagramofFCU04_DAT

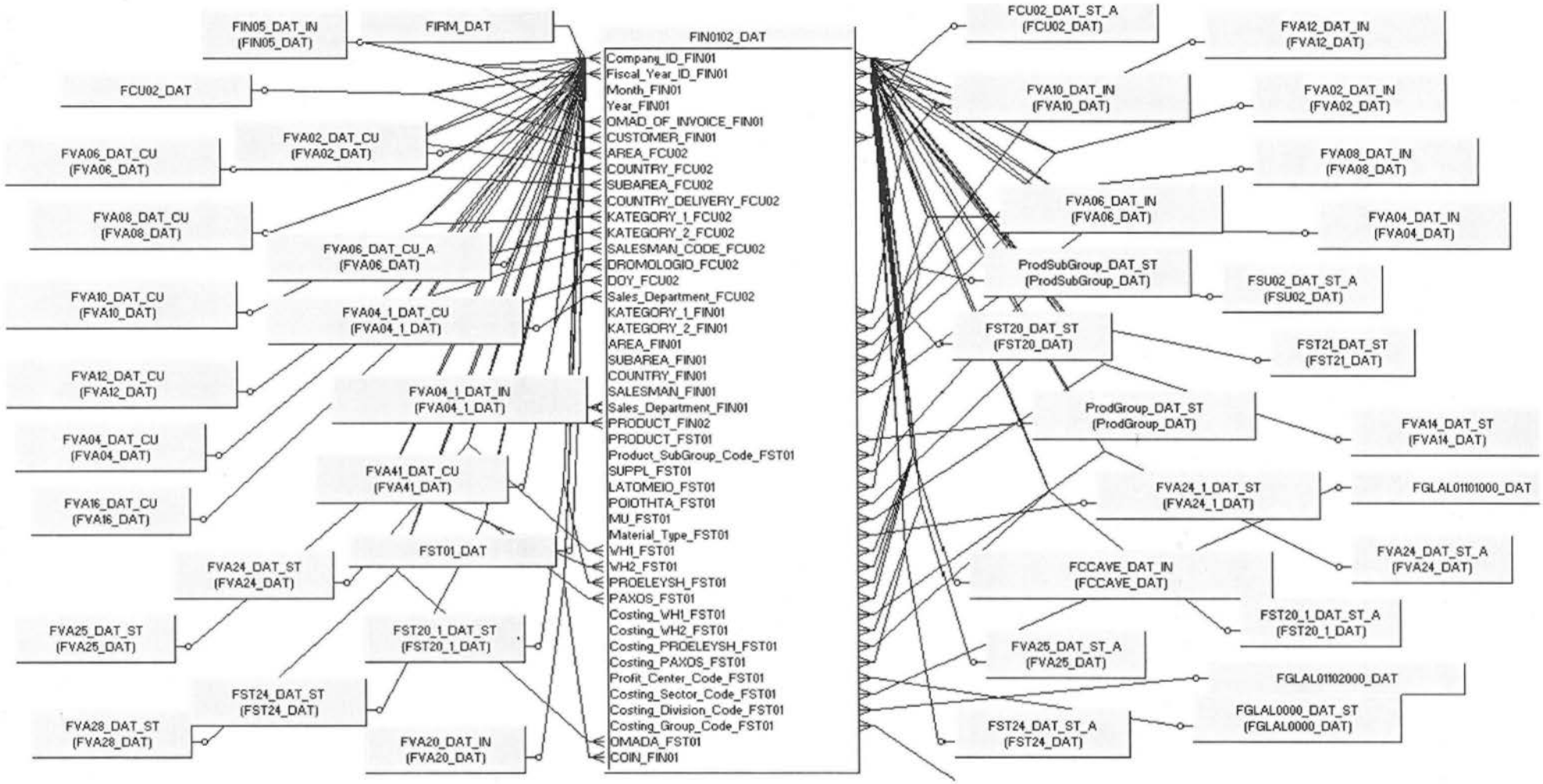
9.Τιμολόγηση Πελατών (FIN0102_DAT)

Στον πίνακα FIN0102_DAT κρατιούνται τα τιμολόγια των πελατών. Εδώ θα βρούμε πληροφορίες για το ποιος πελάτης αγόρασε τι, πότε, από ποια περιοχή και από ποιόν πελάτη. Ουσιαστικά θα μπορούσαμε να πούμε ότι σε αυτόν τον πίνακα καταγράφονται όλες οι κινήσεις πωλήσεων. Οι κύριες διατάσεις που συνδέεται ο factπίνακας για να πάρει αναλυτικές πληροφορίες είναι Customer_fin01 (Αρχείο Πελατών) και Product_fin01 (Αρχείο Αποθήκης ειδών).

Πεδία Πίνακα	**Περιγραφή Πεδίου**
CUSTOMER_FIN01	Αρχείο Πελατών
PRODUCT_FIN02	Αρχείο Αποθήκης ειδών
OMAD_OF_INVOICE_FIN01	Τύποι Τιμολογίου
Sales_Department_FIN01	Ιστορικά Τμήματα Πωλήσεων
COIN_FIN01	Νομίσματα
KATEGORY_1_FIN01	Ιστορικές Κατηγορίες Πελάτη 1
KATEGORY_2_FIN01	Ιστορικές Κατηγορίες Πελάτη 2
AREA_FIN01	Ιστορικές Περιοχές
SUBAREA_FIN01	ΙστορικέςΥποπεριοχές
COUNTRY_FIN01	Ιστορικές Χώρες
SALESMAN_FIN01	Ιστορικοί Πωλητές
AREA_FCU02	Περιοχές
COUNTRY_FCU02	Χώρες
SUBAREA_FCU02	Υποπεριοχές
COUNTRY_DELIVERY_FCU02	Χώρες Παράδοσης
KATEGORY_1_FCU02	Κατηγορίες Πελάτη 1
KATEGORY_2_FCU02	Κατηγορίες Πελάτη 2
SALESMAN_CODE_FCU02	Πωλητές
DROMOLOGIO_FCU02	Δρομολόγια
DOY_FCU02	ΔΥΟ
WH1_FST01	Λογικές Αποθήκες
WH2_FST01	Φυσικές Αποθήκες
PROELEYSH_OMADA_FST01	Ομάδες Προελεύσεων
PROELEYSH_FST01	Προέλευση Προϊόντων
PAXOS_FST01	Πάχος Προϊόντων
OMADA_FST01	Ομάδες Είδους
Sales_Department_FCU02	Τμήματα Πωλήσεων
PRODUCT_FST01	Ομάδες Προϊόντων
Material_Type_FST01	Τύποι Υλικού
CUSTOMER_FST01	Πελάτες Φασόν
Product_SubGroup_Code_FST01	Υπο-ομάδες Προϊόντων
SUPPL_FST01	Συνήθεις Προμηθευτές
LATOMEIO_FST01	Λατομεία

POIOTHTA_FST01	Ποιότητες
MU_FST01	Μονάδες Μέτρησης Αποθήκης
WH1_FST01, WH2_FST01, PROELEYSH_FST01, PAXOS_FST01, Month_FIN01, Year_FIN01	Μέση Τιμή Κόστους Μηνός
Costing_WH1_FST01	Κοστολογικές Ομάδες - Λογικές Αποθήκες
Costing_WH2_FST01	Κοστολογικές Ομάδες - Φυσικές Αποθήκες
Costing_PROELEYSH_FST01	Κοστολογικές Ομάδες - Προελεύσεις
Costing_PAXOS_FST01	Κοστολογικές Ομάδες – Πάχος
Profit_Center_Code_FST01	Κέντρα Κέρδους
Costing_Sector_Code_FST01	Κοστολογικοί Τομείς
Costing_Division_Code_FST01	Κοστολογικοί Κλάδοι
Costing_Group_Code_FST01	Κοστολογικές Ομάδες
COIN_FIN01	Νόμισμα Συναλλαγής.
COIN_ISOTIM_FIN01	Ισοτιμία Νομίσματος σε Euro
DOCUM_FIN01	Αριθμός Παραστατικού. Προκύπτει με συνένωση πεδίων.
Sales_Department_FIN01	Τμήμα Πωλήσεων.

Παρακάτω ακολουθεί το διάγραμμα οντοτήτων για τον πίνακα FIN0102_DAT και όσους πίνακες συνδέονται μαζί του σχηματίζοντας ένα StarΣχήμα



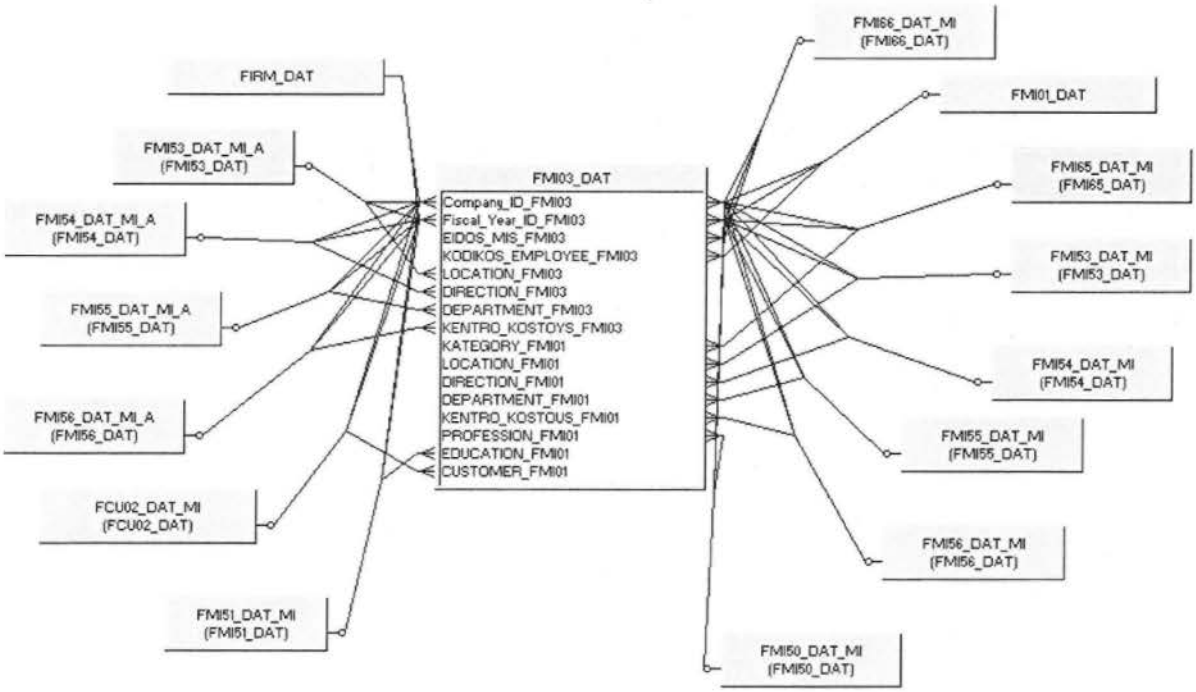
EntityDiagramofFIN0102_DAT

10.Κινήσεις Μισθοδοσίας (FMI03_DAT)

Ο πίνακας FMI03_DAT είναι ένας fact πίνακας και περιέχει πληροφορίες για τις κινήσεις μισθοδοσίας. Από κύριες διαστάσεις περιέχει ο πίνακας είναι ο KODIKOS_EMPLOYEE_FMI03 που συνδέεται με τον πίνακα FMI01_DAT που είναι τα Σταθερά Στοιχεία Εργαζομένων.

Πεδία Πίνακα	Περιγραφή Πεδίου
KODIKOS_EMPLOYEE_FMI03	Σταθερά Στοιχεία Εργαζομένων
LOCATION_FMI03	Ιστορικοί Τόποι Πληρωμής
DIRECTION_FMI03	Ιστορικές Διοικητικές Διευθύνσεις
DEPARTMENT_FMI03	Ιστορικά Διοικητικά Τμήματα
KENTRO_KOSTOYS_FMI03	Ιστορικά Κέντρα Κόστους (Μισθοδοσία)
EIDOS_MIS_FMI03	Τύποι Αποδοχών
PROFESSION_FMI01	Ειδικότητες
EDUCATION_FMI01	Σπουδές
LOCATION_FMI01	Τόποι Πληρωμής
DIRECTION_FMI01	Διοικητικές Διευθύνσεις
DEPARTMENT_FMI01	Διοικητικά Τμήματα
KENTRO_KOSTOUS_FMI01	Κέντρα Κόστους (Μισθοδοσία)
KATEGORY_FMI01	Κατηγορίες Εργαζομένου
CUSTOMER_FMI01	Εργαζόμενοι - Πελάτες

Παρακάτω ακολουθεί το διάγραμμα οντοτήτων για τον πίνακα FMI03_DAT και όσους πίνακες συνδέονται μαζί του σχηματίζοντας ένα Star Σχήμα



EntityDiagramofFMI03_DAT

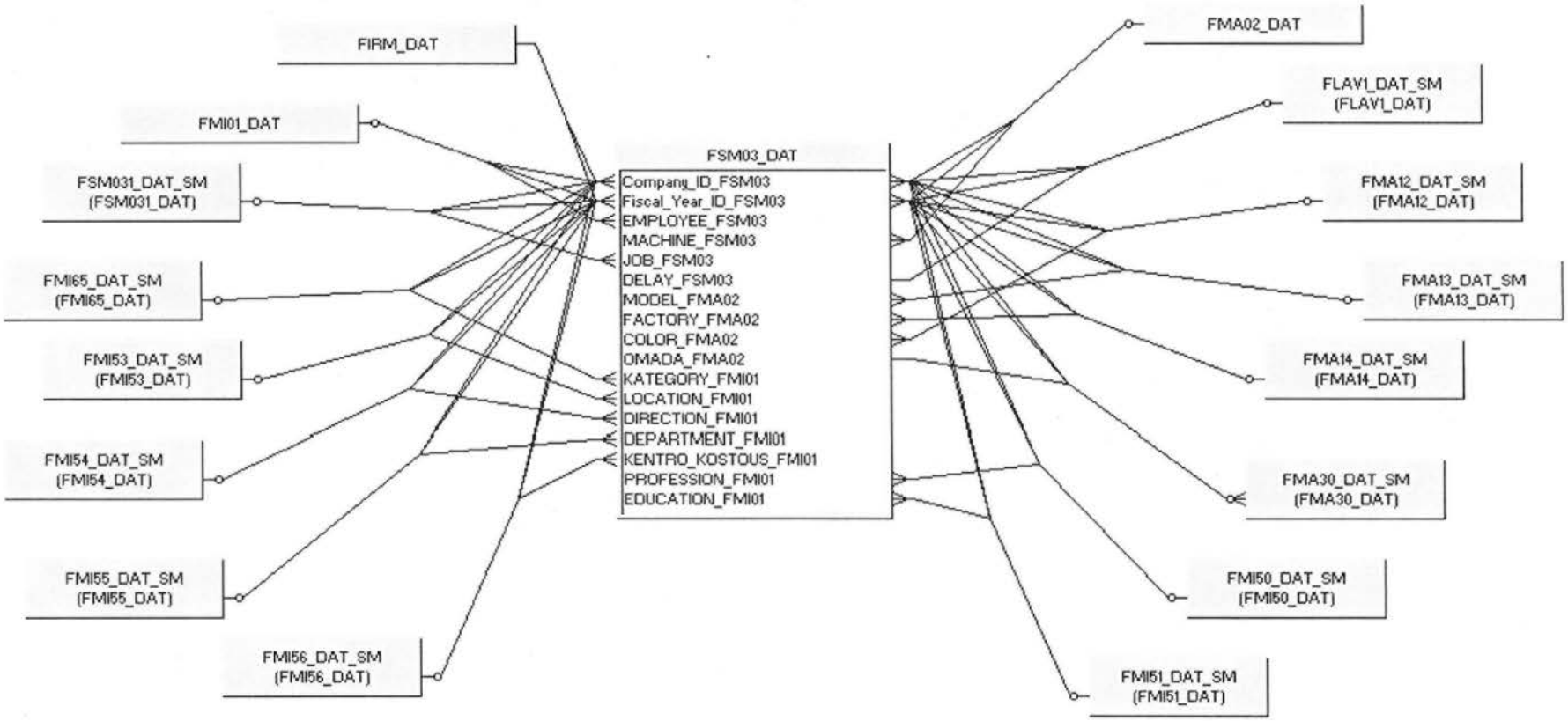
11.Κινήσεις Συνεργείου (FSM03_DAT)

Πίνακας FSM03_DAT περιέχει πληροφορίες σχετικές με το συνεργείο. Εδώ καταγράφονται όλες τις κινήσεις (είτε εργασιακές είτε κοστολογικές) που γίνονται στα πλαίσια του συνεργείου. Οι κύριες διαστάσεις του FSM03_DAT είναι οι πίνακες:

- FMA02_DAT που περιέχει τα Σταθερά Στοιχεία Μηχανημάτων και συνδέεται με τον πίνακα FSM03_DAT μέσω του πεδίου MACHINE_FSM03
- FMI01_DAT που περιέχει τα Σταθερά Στοιχεία Εργαζομένων και συνδέεται με τον πίνακα FSM03_DAT μέσω του πεδίου EMPLOYEE_FSM03

Πεδία Πίνακα	**Περιγραφή Πεδίου**
MACHINE_FSM03	Σταθερά Στοιχεία Μηχανημάτων
EMPLOYEE_FSM03	Σταθερά Στοιχεία Εργαζομένων
JOB_FSM03	Εργασίες Συνεργείου
DELAY_FSM03	Καθυστερήσεις
KATEGORY_FMI01	Κατηγορίες Εργαζομένου
LOCATION_FMI01	Τόποι Πληρωμής
DIRECTION_FMI01	Διοικητικές Διευθύνσεις
DEPARTMENT_FMI01	Διοικητικά Τμήματα
KENTRO_KOSTOUS_FMI01	Κέντρα Κόστους (Μισθοδοσία)
COLOR_FMA02	Χρώματα Μηχανημάτων
MODEL_FMA02	Μοντέλα Μηχανημάτων
FACTORY_FMA02	Εργοστάσια Κατασκευής Μηχανημάτων
OMADA_FMA02	Ομάδες Σύνθεσης Μηχανημάτων
PROFESSION_FMI01	Ειδικότητες
EDUCATION_FMI01	Σπουδές
Minute_Duration_FSM03	Διάρκεια Επισκευής
MACHINE_FSM03	Κωδικός Μηχανήματος
Cost_Center_FSM03	Κέντρο Κόστους

Παρακάτω ακολουθεί το διάγραμμα οντοτήτων για τον πίνακα FSM03_DAT και όσους πίνακες συνδέονται μαζί του σχηματίζοντας ένα StarΣχήμα.



EntityDiagramofFSM03_DAT

12.ΚινήσειςΛογιστικής (FGLAL04_DAT)

Στον πίνακα αυτό ενοποιήθηκαν οι κινήσεις της Γενικής Λογιστικής (ετήσιες και Μηνιαίες) και οι κινήσεις της Αναλυτικής Λογιστικής. Υπάρχει ειδικό πεδίο το οποίο μπορεί να διαχωρίσει τις δύο σύνολα κινήσεων. Το πεδίο αυτό είναι το Ledger_Section_FGLAL04 και μπορεί να έχει μία από τις παρακάτω δύο τιμές:

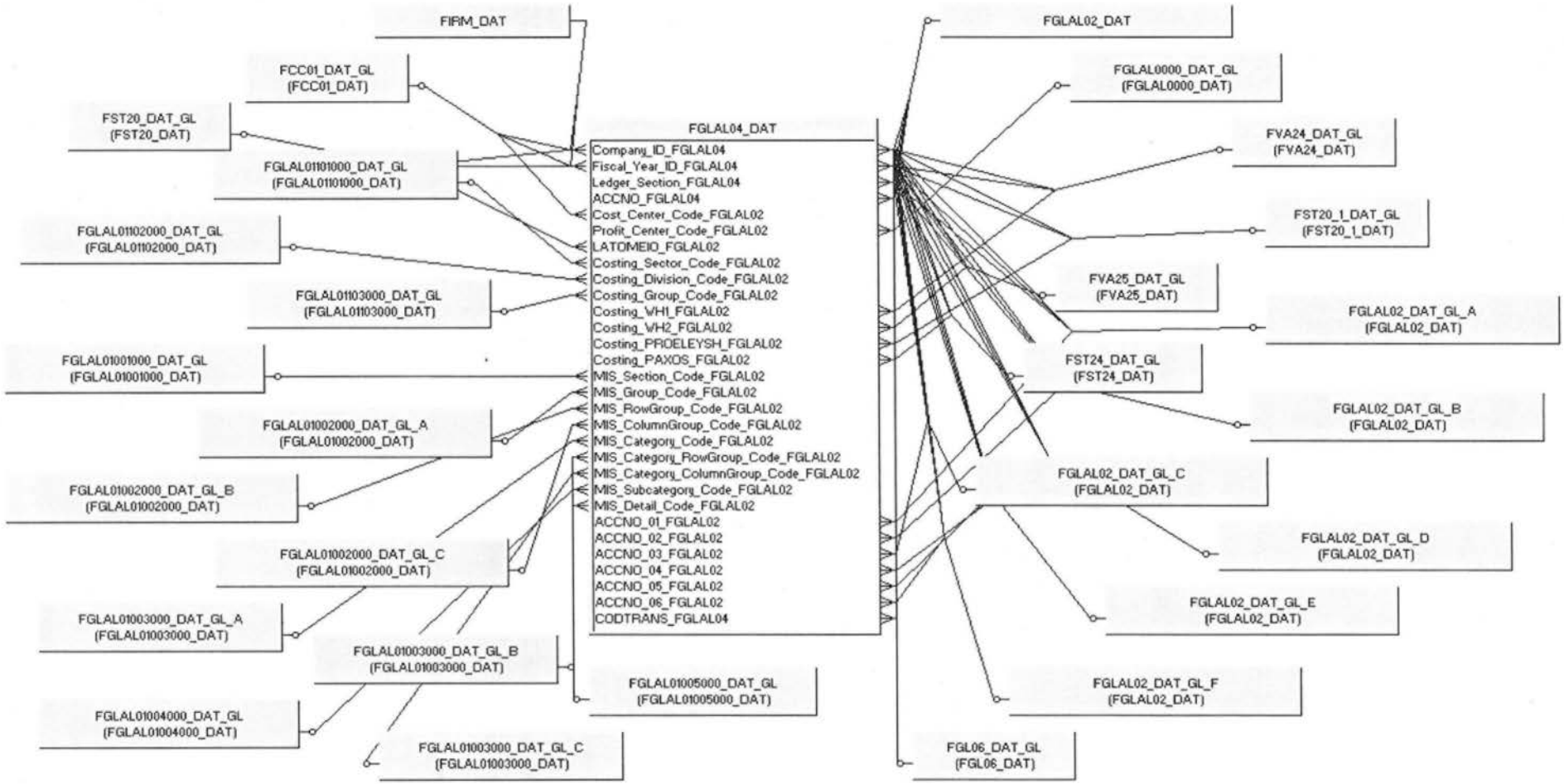
- «Γ/Λ» όταν πρόκειται για Λογαριασμό Γενικής Λογιστικής
- «Α/Λ» όταν πρόκειται για Λογαριασμό Αναλυτικής Λογιστικής

Οι τιμές των παραπάνω δύο τιμών είναι με Ελληνικούς Χαρακτήρες. Επιπρόσθετα υπάρχουν και δύο πεδία συντελεστών (factors) τα οποία μπορούν να διαχωρίσουν τα παραπάνω δύο σύνολα κινήσεων και χρησιμοποιούνται για τη δημιουργία Μετρήσιμων ειδικών για την Γενική και Αναλυτική Λογιστική. Τα πεδία αυτά είναι τα GL_Factor_FGLAL04 και AL_Factor_FGLAL04. Το πρώτο πεδίο έχει τιμή ένα (1) οποτεδήποτε πρόκειται για κίνηση Γενικής Λογιστικής (οποτεδήποτε το πεδίο Ledger_Section_FGLAL04 έχει τιμή «Γ/Λ») και τιμή μηδέν (0) οποτεδήποτε πρόκειται για κίνηση Αναλυτικής Λογιστικής. Το αντίστροφο ακριβώς ισχύει για το δεύτερο πεδίο (AL_Factor_FGLAL04).

Πεδία Πίνακα	**Περιγραφή Πεδίου**
ACCNO_FGLAL04	Σταθερά Στοιχεία Λογαριασμών
ACCNO_01_FGLAL02	Σταθερά Στοιχεία Λογαριασμών Γενικής και Αναλυτικής Λογιστικής.
ACCNO_02_FGLAL02	Σταθερά Στοιχεία Λογαριασμών Γενικής και Αναλυτικής Λογιστικής.
ACCNO_03_FGLAL02	Σταθερά Στοιχεία Λογαριασμών Γενικής και Αναλυτικής Λογιστικής.
ACCNO_04_FGLAL02	Σταθερά Στοιχεία Λογαριασμών Γενικής και Αναλυτικής Λογιστικής.
ACCNO_05_FGLAL02	Σταθερά Στοιχεία Λογαριασμών Γενικής και Αναλυτικής Λογιστικής.
ACCNO_06_FGLAL02	Σταθερά Στοιχεία Λογαριασμών Γενικής και Αναλυτικής Λογιστικής.
Cost_Center_Code_FGLAL02	Κέντρα Κόστους
Profit_Center_Code_FGLAL02	Κέντρα Κέρδους
LATOMEIO_FGLAL02	Λατομεία
Costing_Sector_Code_FGLAL02	Κοστολογικοί Τομείς
Costing_Division_Code_FGLAL02	Κοστολογικοί Κλάδοι
Costing_Group_Code_FGLAL02	Κοστολογικές Ομάδες
Costing_WH1_FGLAL02	Κοστολογικές Ομάδες - Λογική Αποθήκη
Costing_WH2_FGLAL02	Κοστολογικές Ομάδες - Φυσική Αποθήκη
Costing_PROELEYSH_FGLAL02	Κοστολογικές Ομάδες – Προέλευση
Costing_PAXOS_FGLAL02	Κοστολογικές Ομάδες – Πάχος
MIS_Section_Code_FGLAL02	Τομείς Management Information System

MIS_Group_Code_FGLAL02	Ομάδες Management Information System
MIS_RowGroup_Code_FGLAL02	Ομάδες Management Information System
MIS_ColumnGroup_Code_FGLAL02	Ομάδες Management Information System
MIS_Category_Code_FGLAL02	Κατηγορίες Management Information System
MIS_Category_RowGroup_Code_FGLAL02	Κατηγορίες Management Information System
MIS_Category_ColumnGroup_Code_FGLAL02	Κατηγορίες Management Information System
MIS_Subcategory_Code_FGLAL02	Υποκατηγορίες MIS
MIS_Detail_Code_FGLAL02	ΑνάλυσηΥποκατηγορίαςMIS
Balance_FGLAL04	Υπόλοιπο
DB_FGLAL04	Χρέωση
CR_FGLAL04	Πίστωση

Παρακάτω ακολουθεί το διάγραμμα οντοτήτων για τον πίνακα FGLAL04_DAT και όσους πίνακες συνδέονται μαζί του σχηματίζοντας ένα StarΣχήμα.



EntityDiagramofFGLAL04_DAT

3. Οργάνωση της ETL διαδικασίας.

Η οργάνωση της ETL διαδικασίας βασίστηκε στην οργάνωση του Παραγωγικού συστήματος αφού τα αρχικά δεδομένα δημιουργούνται και ενημερώνονται στο Παραγωγικό σύστημα (από εφαρμογές COBOL).

Η συγκεκριμένηETLδιαδικασία αποτελείται από δύο διακριτές φάσεις, η πρώτη αποτελείται από το Extraction-Transformation και η δεύτερη από Loading. Εδώ είναι σημαντικό να επισημάνω την ύπαρξη μιας μικρής φάσης προετοιμασίας των δεδομένων πριν αρχίσει το Extraction-Transformation. Θα αναφερθώ σε αυτή την φάση ως Φάση Προετοιμασίας.

Αξίζει να σημειωθεί ότι συνήθως οι παραπάνω τρεις φάσεις εκτελούνται διαδοχικά βάσει του ορισμού που έχει καταχωρηθεί στο Parameterfile. Το αρχείο αυτό περιέχει όλες τις παραμέτρους που χρειάζεται το workflow για να τρέξει όλες τις φάσης για την κάθε μια εταιρία ξεχωριστά.

Στην συγκεκριμένη εργασία το γενικό workflowέχει προοριστεί να τρέχει όλες τις διαδικασίες για μία εταιρία μονάχα. Επειδή έχουμε συνολικά τρείς εταιρίες (Μάρμαρα, Γρανίτες, Αδρανή Υλικά) το workflow έχει αντιγραφτεί τρείς φορές, όπου και τα τρία μαζί έχουν τοποθετηθεί να εκτελούνται σειριακά. Στη συνέχεια τα workflow αυτά παίρνουν παραμέτρους από το Parameterfile, αυτό σημαίνει ότι όσες εταιρίες έχουμε τόσες φορές θα να ορίσουμε όλες τις παραμέτρους στο παραμετρικό αρχείο.

Το τελικό συμπέρασμα είναι ότι σε περίπτωση που υπάρξει ανάγκη προσθήκης νέας εταιρίας το μόνο που χρειάζεται να κάνει ο τεχνικός είναι να αντιγράψει άλλη μια φορά το workflow και να ορίσει τις παραμέτρους στο αντίστοιχο αρχείο.

Πιο αναλυτικά για το παραμετρικό αρχείο αναφέρεται στο κεφάλαιο «Δήλωση εξωτερικών παραμέτρων του workflow»

Φάση Προετοιμασίας

Η φάση Προετοιμασίας εκτελείται στην αρχή του workflow. Ονομάστηκε έτσι επειδή στην ουσία προετοιμάζει τα δεδομένα του παραγωγικού συστήματος για τη φάση Extraction-Transformation. Τα cobol αρχεία πρέπει να περάσουν από κάποια επεξεργασία προτού είναι έτοιμα για χρήση.

Η φάση Προετοιμασίας θα μπορούσε να παραληφθεί ως αυτόνομο κομμάτι και να ενσωματωθεί στο Extraction-Transformation. Ωστόσο με αυτό τον τρόπο θα γινόταν αρκετά περίπλοκο το κομμάτι του Extraction-Transformation και για λόγους καθαρότητας έγινε μια ξεχωριστή φάση. Η φάση Προετοιμασίας χωρίζεται σε τρία διακριτά στάδια:

- **FTPcopy**

Αναλαμβάνει την αντιγραφή των αρχείων από το AIX (Advanced Interactive eXecutive) σύστημα της KronosMarble στο Windows Server 2003 που είναι εγκατεστημένος ο IQ Server. Το FTP Copy στάδιο αντιγράφει τα προεπιλεγμένα COBOL αρχεία του Legacy συστήματος στο OEM subdirectory. Αυτό επιτυγχάνεται με τη δημιουργία ενός αρχείου κειμένου με εντολές FTP,το οποίο χρησιμοποιείται για να μεταφερθούν σε binary mode τα data files από το AIX.

Το αρχείο πού εκτελείται από το workflow είναι το ftp.dat το οποίο με την σειρά του γράφει σε αρχείο ftp.txt και στην συνέχεια το εκτελεί. Το ftp.dat εκτελεί μια σειρά από εντολές οι οποίες αρχικά διαγράφουν το περιεχόμενο του ftp.txt αρχείου, ύστερα γράφουν τις καινούριες εντολές εκτέλεσης και τέλος γίνεται έλεγχος ορθότητας εγγραφής. Ο λόγος που ύπαρξης όλης αυτής της διαδικασίας είναι επειδή είναι επιθυμητό να συντηρείται μόνο το αρχείο ftp.dat και όχι το ftp.txt.

Ένα δείγμα εντολών από το αρχείο ftp.txt:

```
open  192.9.200.9 21
bo
bo
binary
GET /data/gran/data92/FAL02.DAT   C:\KMarbles\Data\OEM\FAL02.DAT
GET /data/gran/data92/FAL04.DAT   C:\KMarbles\Data\OEM\FAL04.DAT
GET /data/gran/data92/FAX01.DAT   C:\KMarbles\Data\OEM\FAX01.DAT
GET /data/gran/data92/FCCAVE.DAT  C:\KMarbles\Data\OEM\FCCAVE.DAT
```

- **Character set Conversion**

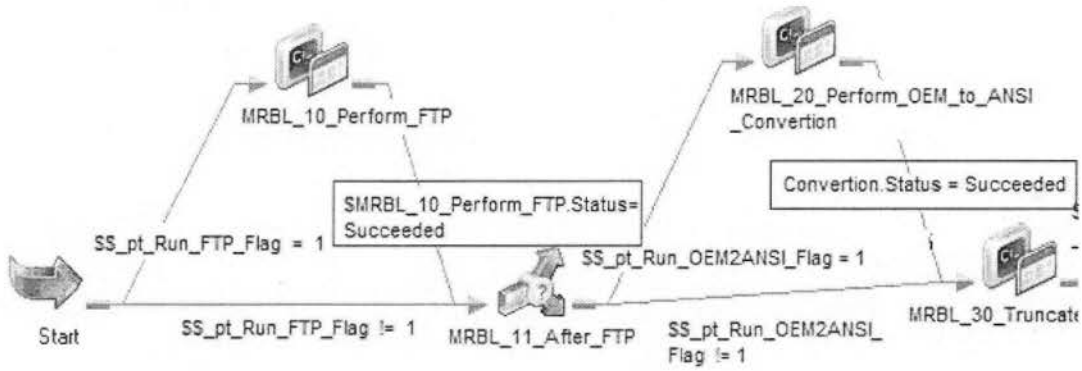
Έχει ως σκοπό την μετατροπή του character set των OEM αρχείων που ήρθαν από το AIX σύστημα σε ANSI αρχεία έτσι ώστε οι Ελληνικοί χαρακτήρες να είναι αναγνωρίσιμοι από πλατφόρμες windows. Η command Perform_OEM2ANSI (φαίνεται στο screenshot του workflow πιο κάτω) εκτελεί

ένα batch file to_ansi που με την βοήθεια του OEM_To_ANSI.exe που αντιγράφει τα προεπιλεγμένα COBOL αρχεία από OEM subdirectory σε ANSI subdirectory ενώ συγχρόνως μεταφράζει σε ansi character set. Η κλήση του OEM_To_ANSI.exe γίνεται πολλές φορές, μία φορά για κάθε προεπιλεγμένο COBOL αρχείο.

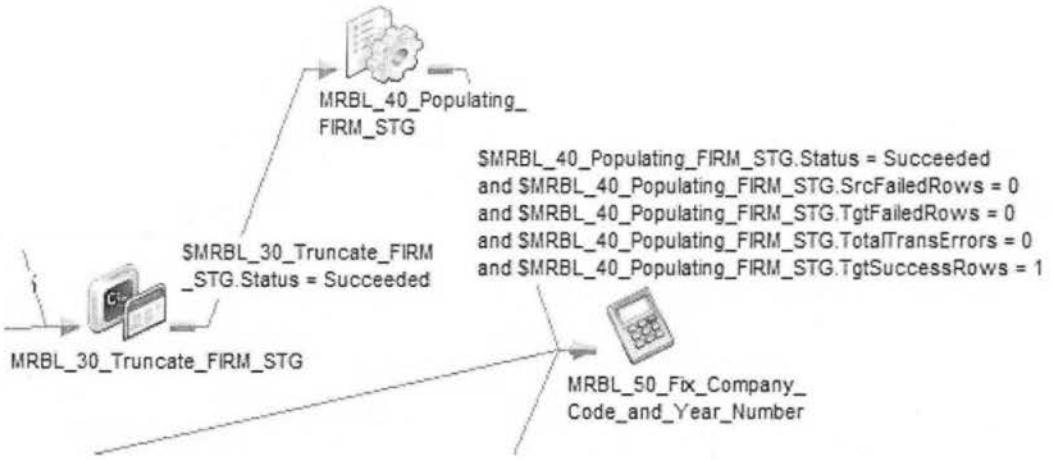
```
e:
cd E:\Informatica_User_Objects\SrcFiles

    OEM_To_ANSI OEM\FAL02.DAT   ANSI\FAL02.DAT
    OEM_To_ANSI OEM\FAL04.DAT   ANSI\FAL04.DAT
    OEM_To_ANSI OEM\FAX01.DAT   ANSI\FAX01.DAT
    OEM_To_ANSI OEM\FAX02.DAT   ANSI\FAX02.DAT
    OEM_To_ANSI OEM\FCC01.DAT   ANSI\FCC01.DAT
    OEM_To_ANSI OEM\FCCAVE.DAT  ANSI\FCCAVE.DAT
    OEM_To_ANSI OEM\FCCST.DAT   ANSI\FCCST.DAT
```

- **Εκκίνηση της φάσης Extraction-Transformation**

Ελέγχει αν τα δύο προηγούμενα στάδια ολοκληρώθηκαν άρτια , δηλαδή εάν όλα τα προεπιλεγμένα Cobol αρχεία έχουν «έρθει» σωστά. Σε αυτό το σημείο γίνονται οι έλεγχοι αν η μεταφορά FTPκαι η μετατροπή OEM_TO_ANSI ολοκληρώθηκαν με status=succeeded.



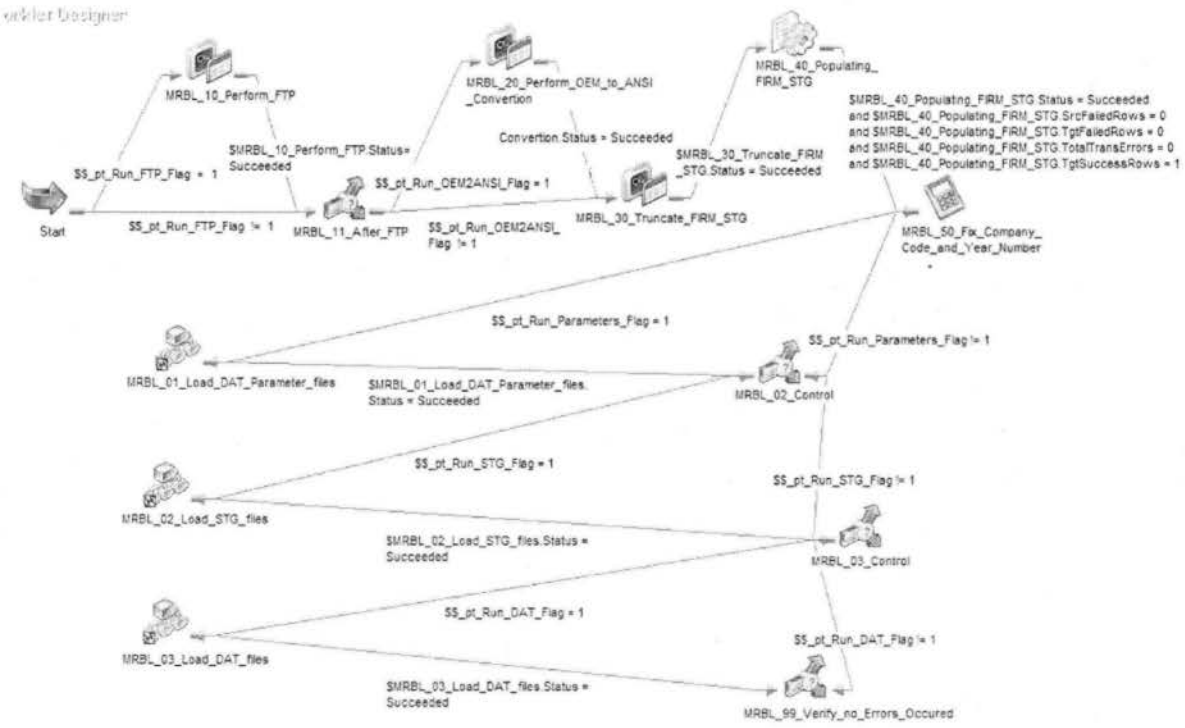
Επίσης σε αυτό το στάδιο προετοιμάζεται το έδαφος για την πρώτη φάση (Extraction-Transformation) που είναι και η πλέον σημαντική της ETL διαδικασίας. Κυρίαρχο ρόλο στην προετοιμασία διαδραματίζει το session Populating_FIRM_STG όπου εκτελείται ο πίνακας FIRM_STG. Ο πίνακας FIRM_STG είναι ό πρώτος πίνακας που τρέχει και περιέχει δύο βασικές πληροφορίες όπως το CompanyID (κωδικός της τρέχουσας εταιρίας) και το FiscalYearID (τρέχουσα λογιστική χρονιά). Ο κωδικός εταιρία και η τρέχουσα λογιστική χρονιά υπάρχουν ως ξένα κλειδιά σχεδόν σε όλους τους πίνακες.



Οποιοδήποτε λάθος ή σφάλμα που προκύπτει κατά την εκτέλεση του FIRM_STG σταματάει την εκτέλεση όλης της ETL διαδικασία και καταγράφεται το σφάλμα στο error log.

Στη συνέχεια ο calculator Fix_CompanyCode_and_YearNumber παίρνει τα πιο πάνω πεδία και τα αναθέτει ως παραμέτρους για να περαστούν στη συνέχεια στους stage πίνακες.
Company Code = CompanyIDκαι Year Number = FiscalYearID.

Εάν δεν εντοπιστεί κάποιο λάθος σηματοδοτείται η έναρξη της επόμενης φάσης.



Για την αποφυγή ιδιαίτερα σημαντικών λαθών κατά την ETL διαδικασία είναι πολύ σημαντικό να εξασφαλιστεί ότι ανά πάσα στιγμή θα εκτελείται μόνο μία

από της φάσεις Extraction-Transformation ή Loading (δηλαδή κάθε φορά θα τρέχει μόνο ένα από τα worklet Load_DAT_Parameter_files,Load_Stg_files και Load_Dat_files).

Αυτό επιτυγχάνεται με το να μπουν στην σειρά τα αντίστοιχα worklets ώστε να τρέξουν το ένα πίσω από το άλλο, τέλος πριν από το καθένα ελέγχεται αν έγινε επιτυχής η εκτέλεση και ύστερα προχωράει στο επόμενο βήμα. Αν για κάποιο λόγο σε κάποιο στάδιο έγινε ένα απρόσμενο σφάλμα, σταματάει η ETL διαδικασία και καταγράφεται το λάθος στο αντίστοιχο Error_Log.

Φάση Extraction-Transformation

Σε αυτή την φάση πραγματοποιείται η εκτέλεση του worklet MRBL_02_Load_STG_filesto οποίο με την σειρά του εκτελεί όλους stage πίνακες. Μέσα από τους stage πίνακες δουλεύεται όλη η λογική Extraction – Transformation.

Σχεδόν όλα τα δεδομένα που έρχονται από το παραγωγικό σύστημα επεξεργάζονται από τους stage πίνακες πριν γραφτούν στην βάση. Είναι το μεταβατικό στάδιο όπου τα δεδομένα εξάγονται από το παραγωγικό σύστημα, περνάνε από εμπλουτισμό και επεξεργασία, γίνεται έλεγχος σφαλμάτων ύστερα αν ολοκληρωθούν όλοι οι πίνακες χωρίς σφάλματα τότε τα δεδομένα περνάνε στην επόμενη φάση της φόρτωσης (loading).

Φάση Loading

Η φάση Loading πραγματοποιείται με την εκτέλεση του worklet MRBL_02_Load_DAT_files το οποίο με την σειρά του εκτελεί όλους dat πίνακες. Οι dat πίνακες αποτελούν τους τελικούς πίνακες όπου καταγράφονται τα επεξεργασμένα δεδομένα στην βάση.

Ο λόγος που η ETL διαδικασία χωρίστηκε σε δύο διακριτές φάσεις, είναι επειδή ήταν απαραίτητη η ύπαρξη πινάκων που θα είναι προσωρινοί - μεταβατικοί (όπου τα δεδομένα μορφοποιούνται – επεξεργάζονται κα πριν από κάθε επεξεργασία διαγράφονται) και πινάκων που είναι σταθεροί – ιστορικοί (όπου τα δεδομένα γράφονται στην τελική τους μορφή).

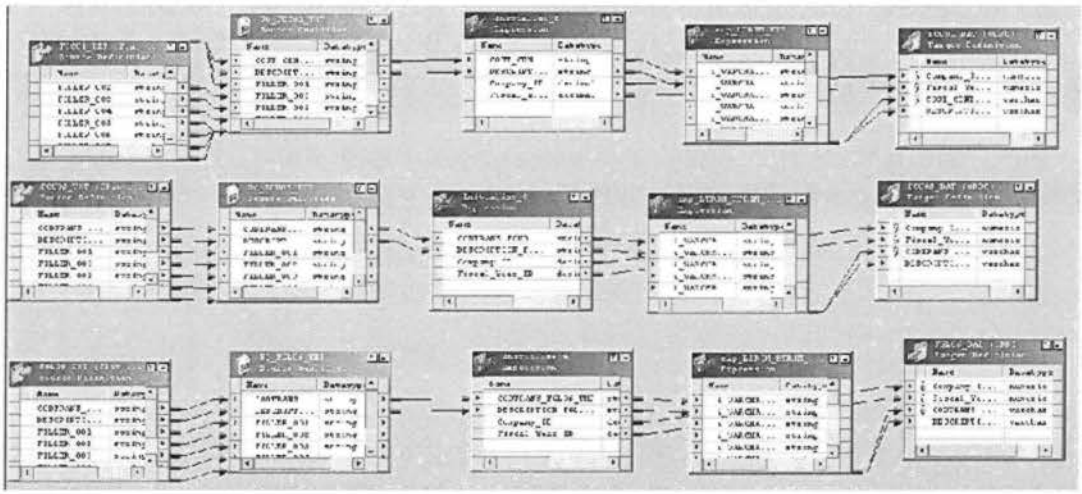
4. Υλοποίηση Λύσης

Η υλοποίηση της ETL διαδικασίας με το εργαλείο Informaticaπεριλαμβάνει τις ακόλουθες ομάδες πινάκων. Για τις συγκεκριμένες ομάδες πινάκων έχω αναφερθεί εκτενέστερα στο κεφάλαιο Σχεδίαση λύσης

Παραμετρικοί Πίνακες (ParamFiles)

Αποτελούν τους πίνακες με κατάληξηParamFiles_DAT. Κατά κανόνα είναι οι πίνακες που περιέχουν περιγραφές για κάποια πεδία. Οι παραμετρικοί πίνακες περιέχουν λίγα πεδία κυρίως την τρέχουσα χρονιά, τον κωδικό της εταιρίας, έναν κωδικό του πίνακα και μία περιγραφή. Ο λόγος που λέγονται παραμετρικοί ή περιγραφικοί πίνακες είναι επειδή τα βασικά πεδία τους είναι ένας κωδικός και η περιγραφή του.

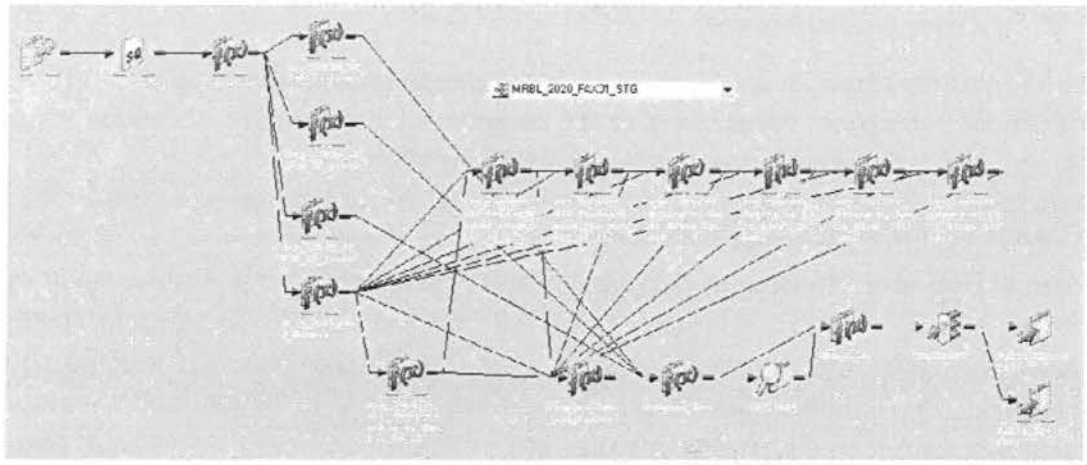
Ο παραμετρικοί πίνακες διαφέρουν από τους υπόλοιπους πίνακες γιατί δεν περνάνε από πίνακες Αναβάθμισης (το Stage (_stg) στάδιο) αλλά οδηγούνται απευθείας σε Τελικούς πίνακες (_dat).



Πίνακες Αναβάθμισης (Stage)

Όπως φαίνεται και από το όνομα τους, χρησιμεύουν στο μετασχηματισμό των δεδομένων με παράλληλο εμπλουτισμό και ομογενοποίηση. Αποτελούν το ενδιάμεσο στάδιο πριν την τελική τους αποθήκευση. Περιέχουν πληθώρα πεδίων τα οποία είναι είτε τελικά είτε βοηθητικά. Διαφοροποιούνται από τους άλλους πίνακες με τη πολύπλοκη ροή και επεξεργασία των δεδομένων. Είναι επίσης το ενδιάμεσο στάδιο πριν τη δημιουργία του Star Σχήματος. Είναι οι πίνακες που χρησιμοποιούνται στη Loading-Transformationφάση.

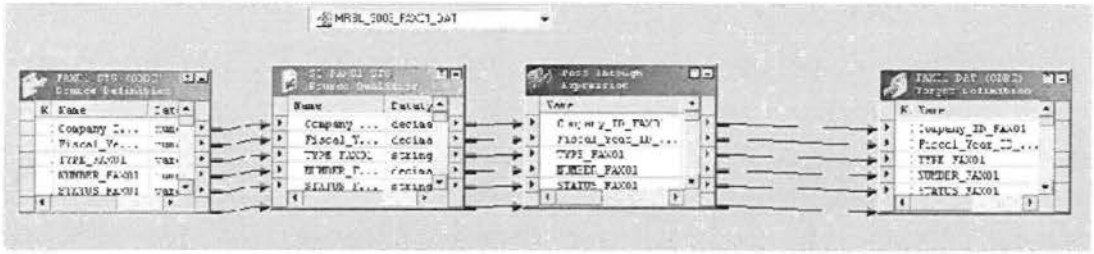
Τα mappings των πινάκων αυτών είναι από τα πιο περίπλοκα στην υλοποίηση μιας και εκεί εφαρμόζεται όλη η λογική της ETL διαδικασίας.



Τελικοί Πίνακες (Dat)

Είναι οι πίνακες όπου αποθηκεύονται τα δεδομένα που περνάνε από πίνακες αναβάθμισης. Ουσιαστικά αποτελούν το τελικό στάδιο της ETL διαδικασίας (Loading), αποθηκεύουν μετασχηματισμένα τα δεδομένα (που έχουν περάσει από stgtables) του παραγωγικού συστήματος. Επίσης θα μπορούσαμε να τους ονομάσουμε και ιστορικούς (σταθερούς) πίνακες γιατί κρατάνε δεδομένα απ' όλες της χρόνιες και εταιρίες

Τέλος, αποτελούν την πηγή δεδομένων των Αναφορών (Reports), καθώς δίνουν τη δυνατότητα σύγκρισης - σύνοψης στοιχείων διαφορετικών Οικονομικών Χρήσεων (ετών) και Μελών ομίλου εταιριών της ΚΜ.

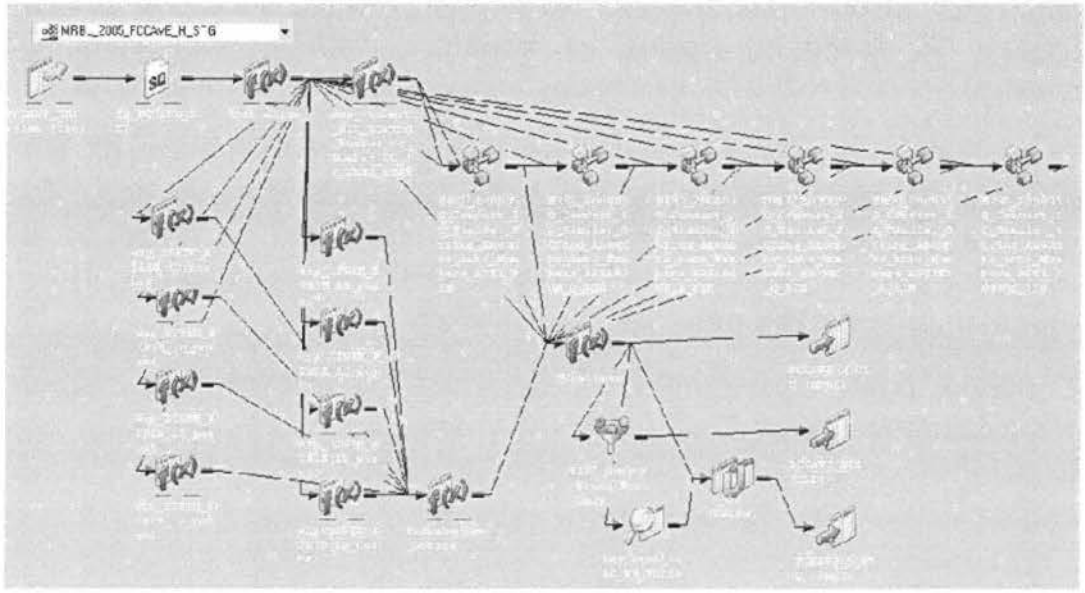


Επίσης δημιουργήθηκαν άλλες δύο κατηγορίες βοηθητικών πινάκων:

Πίνακες Κάθετης Ανάπτυξης (Vertical)

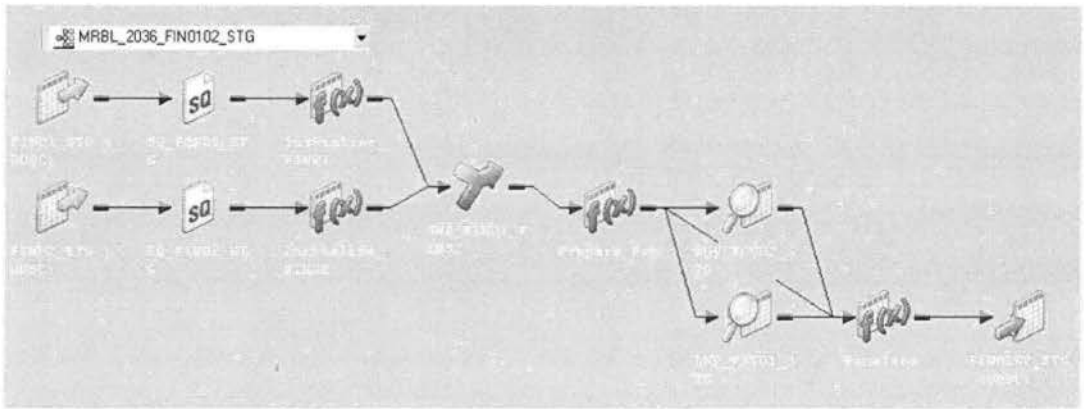
Έχουν ως στόχο να μετατρέψουν την οριζόντια διατύπωση κάποιον πεδίων σε κάθετη διατύπωση. Οι πίνακες οριζόντιας ανάπτυξης είναι FCCAVE_V_STG και FSPAVE_V_STG που προήλθαν από τους αντίστοιχους FCCAVE_H_STG και FSPAVE_H_STG πίνακες κάθετης ανάπτυξης. Αυτό έγινε για χάρη ευκολίας διεξαγωγής reports, όπου ο χρήστης θα έβγαζε χρήσιμες πληροφορίες για την εταιρία.

Πχ. Ο πίνακας κάθετης ανάπτυξης FCCAVE_H_STG έχει το πεδία AGORES_A_PIN_01_FCCAVE,
AGORES_A_PIN_02_FCCAVE,
AGORES_A_PIN_03_FCCAVE ...και έτσι μέχρι
AGORES_A_PIN_12_FCCAVE.
Δηλαδή αγορές που έγιναν για κάθε μήνα ξεχωριστά.
Ο πίνακας που υλοποιήθηκε για την οριζόντια ανάπτυξη FCCAVE_V_STG έχει ενοποιήσει όλα τα πιο πάνω πεδία σε ένα, το Purchased_Value_FCCAVE και οι μήνες γράφονται ο ένας κάτω από τον άλλον (οριζόντια ανάπτυξη) και όχι ξεχωριστά όπως στην κάθετη ανάπτυξη.



Πίνακες Ενοποίησης

Χρησιμεύουν στη δημιουργία Star Σχήματος στην περίπτωση των τριών αρχείων τιμολόγησης (Πελατών, Αγορών και Αγορών Ανταλλακτικών). Συγκεκριμένα, αναλαμβάνουν την ενοποίηση του πίνακα Header (που περιλαμβάνει τα κοινά στοιχεία των τιμολογίων) και του πίνακα Detail (που περιλαμβάνει πιο αναλυτικά της γραμμές των τιμολογίων) σε ένα πίνακα (π.χ. οι πίνακες FIN01 και FIN02 ενοποιούνται στον FIN0102). Ουσιαστικά γίνεται ένα join των δύο πινάκων.



Πιο αναλυτικά, οι πίνακες που δημιουργήθηκαν είναι οι εξής:

FIN0102_STG: Τιμολόγια Πελατών

FIP0102_STG: Τιμολόγια Αγορών Ανταλλακτικών

FSI0102_STG: Τιμολόγια Προμηθευτών

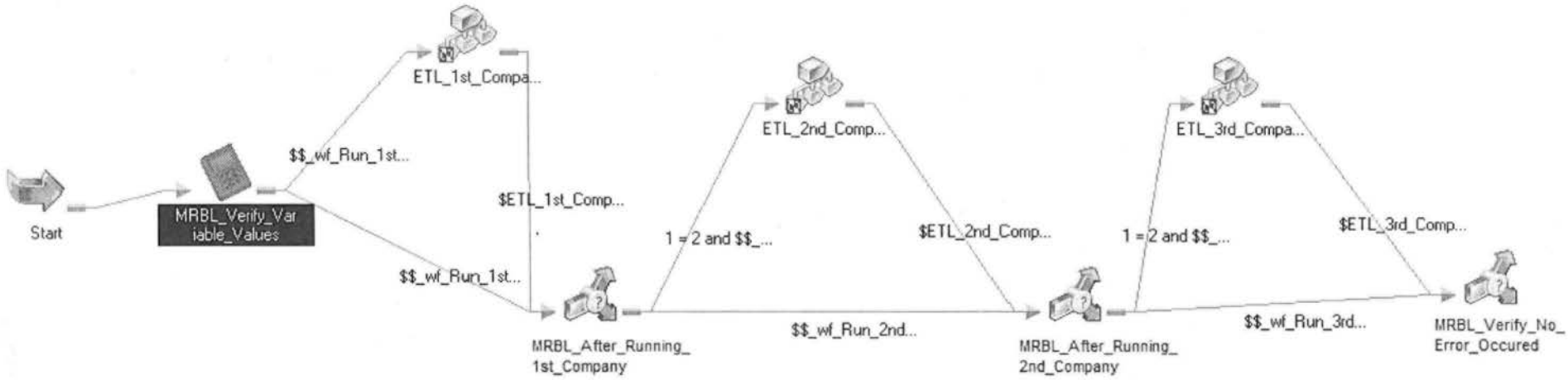
Και στις τρεις περιπτώσεις υπήρχαν δύο αρχεία COBOL, ένα Header με τα στοιχεία που είναι κοινά για όλες τις γραμμές κάθε τιμολογίου και ένα Detail με τα στοιχεία των γραμμών κάθε τιμολογίου. Το αριθμητικό τμήμα του ονόματος αυτών των αρχείων ήταν πάντα 01 για το Header αρχείο και 02 για το Detail αρχείο. Για λόγους συντομίας, οι πρόσθετοι πίνακες Ενοποίησης θα αναφέρονται ως 0102_STG, ενώ οι αρχικοί τους ως 01_STG και 02_STG.

Στην ουσία περιέχουν το σύνολο των πεδίων των 01_STG και 02_STG πινάκων. Σε περιπτώσεις όπου ένα πεδίο περιλαμβάνεται και στους δυο αρχικούς πίνακες, επιλέχθηκε κατά κανόνα το πεδίο από τον πίνακα με την αναλυτικότερη πληροφορία (συνήθως ο 02_STG). Η ονομασία των πεδίων τους διατηρείται ως είχε στους αρχικούς πίνακες.

Επιπλέον, έχουν δημιουργηθεί LookUp πεδία για διασύνδεση με έμμεσα παραμετρικούς πίνακες.

Εκκίνηση διαδικασίας ETL

Η ETL διαδικασία ξεκινάει με το workflow MRBL_0000000_ETL_for_companies_1_to_3 που βλέπουμε στο αμέσως επόμενο screenshot.



MRBL_0000000_ETL_for_companies_1_to_3

Το workflow τρέχει και τις τρείς εταιρίες, την μια μετά την άλλη. Για να την αποφυγή σοβαρών σφαλμάτων είναι σημαντικό να εξασφαλίσουμε ότι οι τρείς εταιρίες αυτές δεν θα τρέξουν ποτέ παράλληλα τα δεδομένα τους. Επειδή και οι τρείς γράφουν στους ίδιους πίνακες δεν γίνεται να γράφουν και οι τρείς ταυτόχρονα. Για το λόγο αυτό τα worklet έχουν μπει στην σειρά, το ένα μετά το άλλο και ενδιάμεσα συνδέονται με Decision (βλ. το screenshot).

Μετά την εκτέλεση της ETL διαδικασίας για τα δεδομένα της πρώτης εταιρίας το decision MRBL_After_Running_1st_company αποφασίζει αν το workflow θα συνεχίσει να τρέχει την επόμενη εταιρία (την 2η) ή όχι, το ίδιο κάνει και το δεύτερο decision MRBL_After_Running_2nd _companyγια την τρίτη εταιρία. Αυτό συμβαίνει γιατί δεν είναι απαραίτητο την ίδια μέρα να τρέξουν και οι τρείς εταιρίες, δίνεται δυνατότητα στον χρήστη να τρέξει ξεχωριστά όποια εταιρία επιθυμεί.

Το τελικό decision MRBL_Verify_No_Error_Occurred ελέγχει αν οι εταιρίες που έτρεξαν προηγουμένως έχουν ολοκληρωθεί με επιτυχία αλλιώς ειδοποιεί με μήνυμα λάθους τον χρήστη. Γενικά σε κάθε worklet/workflow έχει τοποθετηθεί ένα decision για έλεγχο ροής (βλ. Μέθοδος ελέγχου ροής και σφαλμάτων).

Δήλωση εξωτερικών παραμέτρων του workflow

Αναφέρθηκε πιο πάνω ότι τα αρχεία έρχονται από AIX σύστημα με FTP και στη συνέχεια γίνεται η μετατροπή από OEM σε ANSI. Για να γίνει όλη αυτή η διαδικασία πρέπει να υποδείξουμε στην Informatica το μέρος (το directory) που να βρει τα αρχεία, το AIX_path του συστήματος όπου βρίσκονται τα παραγωγικά αρχεία.

Σε μια από τις ιδιότητες της Informatica μπορούμε να ορίσουμε ένα αρχείο που θα έχει κατάληξη .prm να της υποδείξουμε το path που θα βρίσκεται αυτό το αρχείο, και μέσα σε αυτό να δηλώσουμε όσες παραμέτρους θέλουμε να διαβάζει. Με αυτό τον τρόπο μπορούμε ανά πάσα στιγμή να επέμβουμε στο flow εξωτερικά.

Για το λόγο αυτό έχει δημιουργηθεί ένα αρχείο MRBL_WF_PARAMETERS.prm μέσα στο οποίο γίνεται η δήλωση επτά παραμέτρων. Η δήλωση γίνεται για κάθε εταιρία ξεχωριστά, δηλαδή επί τρείς φορές. Το πιο κάτω screenshot δείχνει πως δηλώνουμε στο Parameter Filename το path που βρίσκεται το παραμετρικό αρχείο.

Attribute	Value
Parameter Filename	E:\Informatica_User_Objects\Commands\MRBL_WF_PARAMETERS.PRM
Write Backward Compatible Workflow Log File	☐
Workflow Log File Name	AL_TEST.log
Workflow Log File Directory	$PMWorkflowLogDir\
Save Workflow log by	By runs
Save workflow log for these runs	0
Enable HA recovery	☐

Declare Parameter Filename Path

Οι 7 παράμετροι που δηλώνουμε στο αρχείο είναι οι εξής:

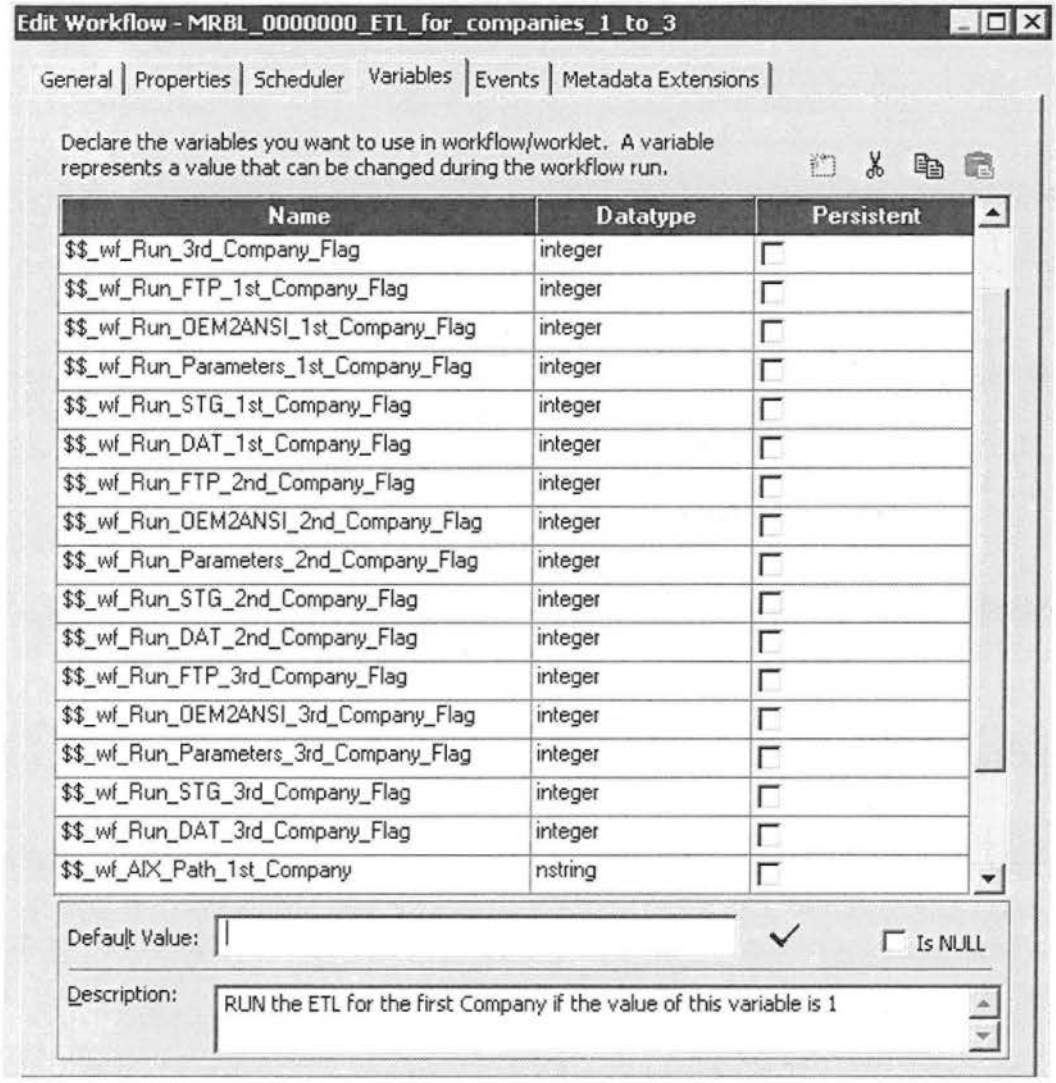
1. $$_wf_Run_1st_Company_Flag=1
2. $$_wf_AIX_Path_1st_Company=/euro/2012/data02
3. $$_wf_Run_FTP_1st_Company_Flag=1
4. $$_wf_Run_OEM2ANSI_1st_Company_Flag=1
5. $$_wf_Run_Parameters_1st_Company_Flag=1
6. $$_wf_Run_STG_1st_Company_Flag=1
7. $$_wf_Run_DAT_1st_Company_Flag=1

(παρόμοια για 2nd Company και 3rd Company)

Όπως ειπώθηκε πιο πάνω στο AIX_Path_1st_Company δηλώνεται το directory όπου βρίσκονται τα αρχεία συστήματος. Οι υπόλοιποι έξη παράμετροι παίζουν ρόλο flag, είναι πάντα είτε ένα είτε μηδέν. Η ύπαρξη τωνflags χρειάζονται για να έχει ο χρήστης την δυνατότητα να ορίζει ανά πάσα στιγμή πιο κομμάτι του κώδικα επιθυμεί να τρέξει.

Αν για παράδειγμα έχει γίνει κάποιο σφάλμα στο κομμάτι του STG worklet τότε όταν στην επόμενη εκκίνηση τουworkflow, τα προηγούμενα στάδια της διαδικασίας που έτρεξαν επιτυχώς και δεν χρειάζεται να τρέξουν πάλι. Δηλαδή το ftp, oem_to_ansi και τα parameters_dat που είναι διαδικασίες πριν από το STGμπορούν να προσπεραστούν (βάζοντας flag=0) και να τρέξει ο κώδικας από εκεί που σταμάτησε.

Τέλος για να χρησιμοποιήσουμε τις παραμέτρους που δηλώθηκαν στο αρχείο MRBL_WF_PARAMETERS.PRM χρειάζεται να τις δηλώσουμε ως μεταβλητές πλέον στο workflow. Με την βοήθεια των μεταβλητών η Informatica πλέον είναι σε θέση να συσχετίσει τις παραμέτρους που δέχεται από το παραμετρικό αρχείο με τις μεταβλητές που θα χρησιμοποιήσει στο workflow. (βλ. πιο κάτω screenshot)

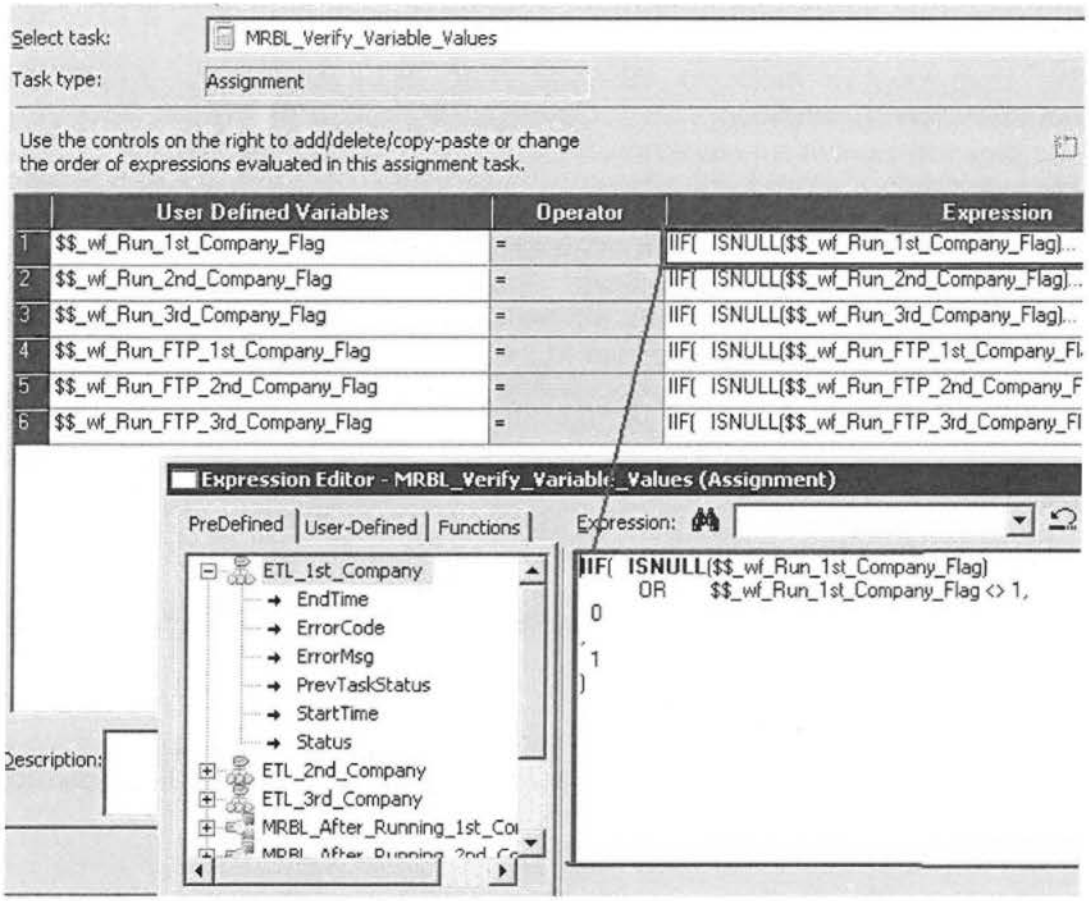


Declare workflow variables

Το πρώτο task που συναντάμε μετά το start είναι ένα assignment (calculator) και συνήθως η δουλεία του είναι να κάνει ανάθεση παραμέτρων ή μεταβλητών που θα χρησιμοποιηθούν στο workflow. Ωστόσο έχουμε την δυνατότητα να κάνουμε εκεί πράξεις και ελέγχους.

Έτσι στο assignment MRBL_Verify_Variable_Values κάνουμε έλεγχο για NULL τιμές. Δεν θέλουμε ποτέ τα flag να είναι NULL, οπότε για αποφυγή λάθους αν κάποια παράμετρος είναι NULLτη μετατρέπουμε σε μηδέν. Η μετατροπή γίνεται στην αρχή του workflow για να μην επιβαρύνεται το worklet με ελέγχους. Ειδάλλως θα χρειαζόταν σε κάθε worklet να γίνεται έλεγχος για Null τιμές πριν από κάθε χρήση των μεταβλητών αυτών. Ο έλεγχος γίνεται ως εξής:

$$_wf_Run_1st_Company_Flag = **IIF**(**ISNULL**($$_wf_Run_1st_Company_Flag)
OR $$_wf_Run_1st_Company_Flag <> 1, 0, 1)



Έλεγχος null τιμών στις παραμέτρους

Εκκίνηση ETL διαδικασίας για την Πρώτη Εταιρία

Στο προηγούμενο κεφάλαιο «Εκκίνηση της ETL διαδικασίας» είδαμε το workflow MRBL_0000000_ETL_for_companies_1_to_3 που αποτελείται από τρία διακριτά worklet που αντιστοιχούν στις τρείς εταιρίες (Μάρμαρα, Γρανίτες, Αδρανή Υλικά).

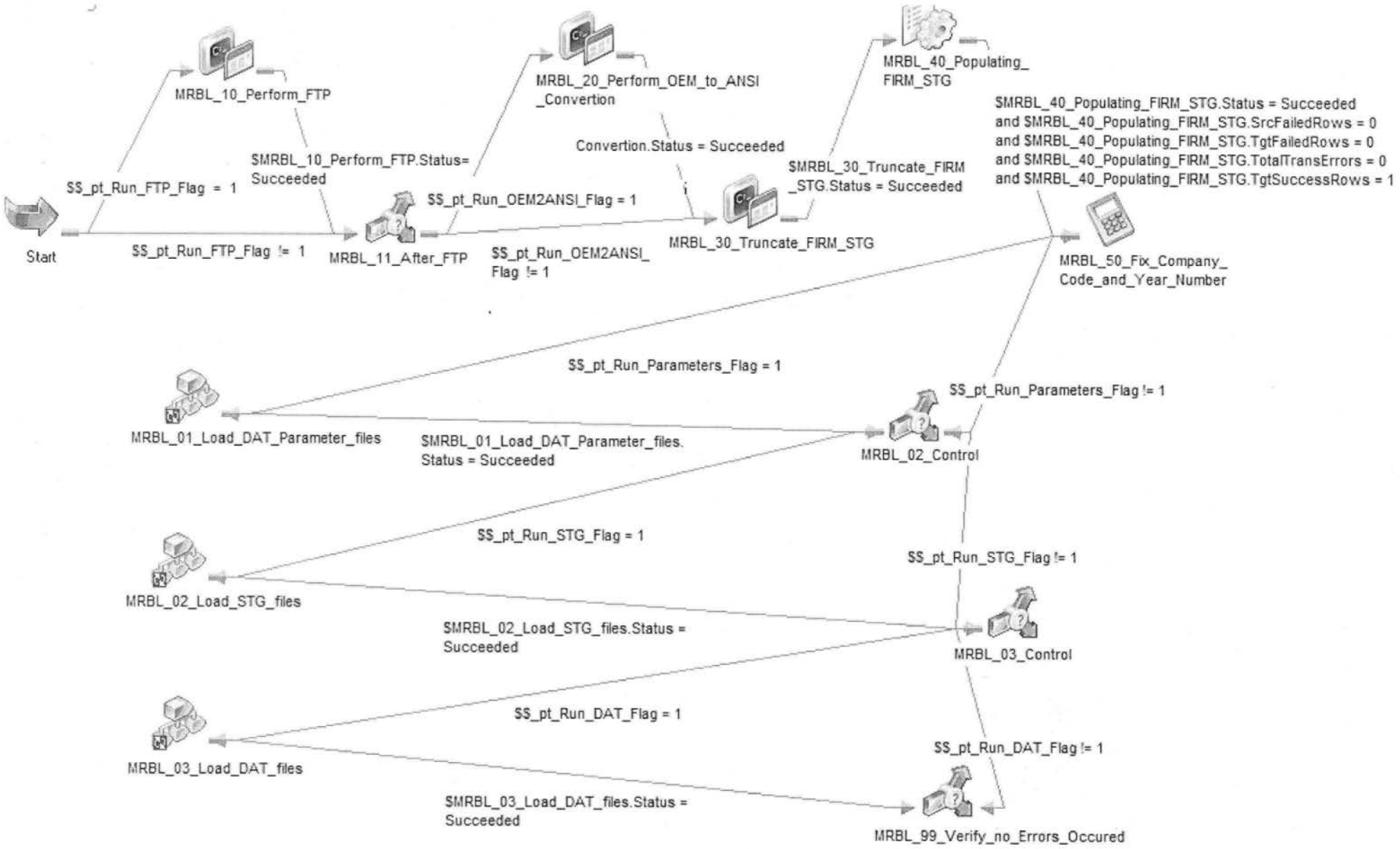
Και τα τρία worklet είναι πανομοιότυπα ο διαχωρισμός γίνεται από το πιο directory χρησιμοποιεί το καθένα. Επειδή η κάθε εταιρία γράφει σε διαφορετικά directory έτσι γίνεται ο διαχωρισμός τους. Συνεπώς ότι ανάλυση γίνει για το ένα worklet ισχύει και για τα υπόλοιπα.

Στη συνέχεια θα ασχοληθώ με την περιγραφή του worklet της Πρώτης Εταιρίας.

Στο γενικό workflow σε συμπυκνωμένη (collapsed) μορφή το workletέχει την παρακάτω μορφή:

ETL_1st_Company

Ενώ σε μορφή πλήρης ανάπτυξης (expanded) το βλέπουμε στο παρακάτω screenshot (Worklet for the First Company).

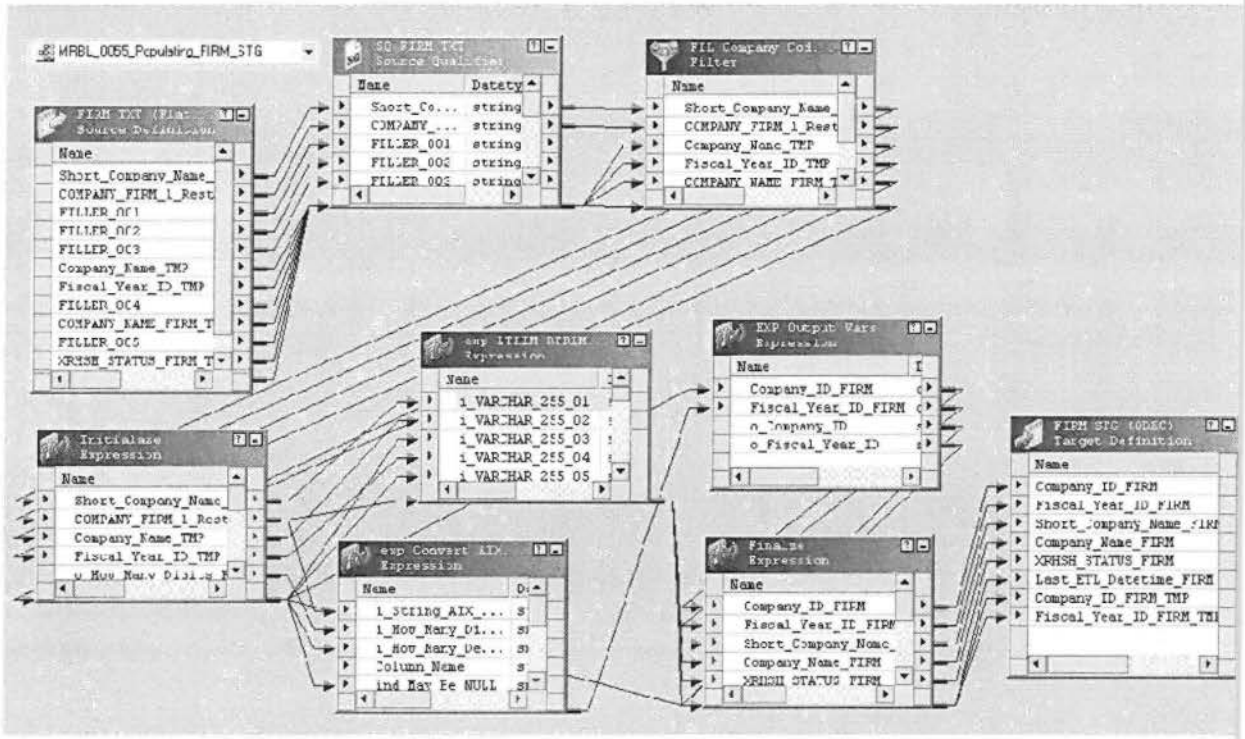


Worklet for the First Company

Μετά από την εκτέλεση των commands FTP και OEM_to_ANSI, που αναλύσαμε σε προηγούμενο κεφάλαιο, εκτελείται το πρώτο session MRBL_40_Populating_FIRM_STG.Το συγκεκριμένο session περιέχει το πρώτο mapping που εκτελείται στην αρχή της ETL διαδικασίας που είναι το MRBL_0055_Populating_FIRM_STG.

MRBL_0055_Populating_FIRM_STG (mapping)

Στη συνέχεια θα περιγράψω την υλοποίηση του mapping Populating_FIRM_STG. Πιο κάτω βλέπουμε πολλά κουτάκια συνδεδεμένα με γραμμούλες, από το πρώτο κουτάκι που λέγεται source definition τα δεδομένα διαβάζονται στη συνέχεια επεξεργάζονται μέσα από επόμενα κουτάκια μέχρι να καταλήξουμε στο τελικό κουτάκι το targetπου γράφει τα δεδομένα σε πίνακα της βάσης.



Πιο συγκεκριμένα η ροή των δεδομένων αρχίζει με το FIRM_TXT,το οποίο με την σειρά του διαβάζει τα δεδομένα από το αρχείο FIRM.DAT το οποίο βρίσκεται στο ANSI directory.

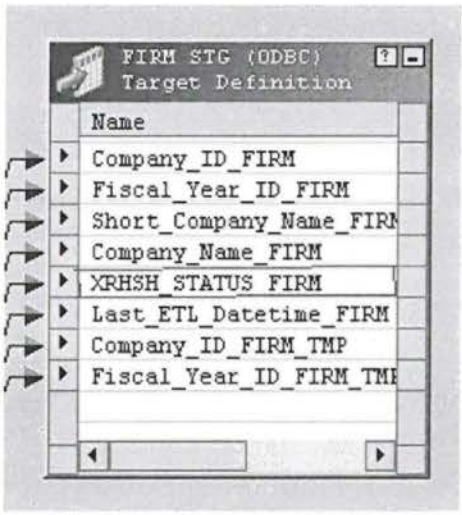
Στη συνέχεια τα data που διαβαστήκαν φιλτράρονται με την χρήση FILT_Company_Code_1_to_9. Στο σημείο αυτό χρησιμοποιείται ένα φίλτρο που απαγορεύει να περάσουν δεδομένα που το COMPANY_NAME τους δεν στα όρια από ένα εώς εννέα :

COMPANY_NAME_FIRM_TMP >= '1' AND COMPANY_NAME_FIRM_TMP <= '9'

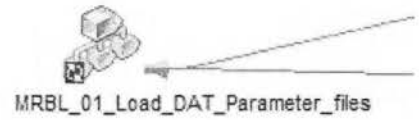
Όσα δεδομένα πληρούν την προϋπόθεση του φίλτρου περνάνε στο επόμενο κουτάκι όπου ακολουθούν μια διαδικασία επεξεργασίας. Στο συγκεκριμένο mappingn επεξεργασία είναι αρκετά εύκολη υπόθεση. Το πρώτο που γίνεται είναι να κοπούν τα κενά που περιέχουν τα αλφαριθμητικά πεδία στην αρχή και το τέλος. Αυτό επιτυγχάνεται με τις εντολές Lefttrim(string) και Righttrim(string).

Τα αλφαριθμητικά πεδία Company_ID_FIRM και Fiscal_Year_ID_FIRM είναι ανάγκη να μετατραπούν σε αριθμητικά πεδία. Αυτό επιτυγχάνεται με την χρήση του reusable Convert_AIX_String_Number_to_Number. Το συγκεκριμένο reusable έχει φτιαχτεί αποκλειστικά για αυτή τη δουλειά, δηλαδή να δέχεται ως είσοδο αλφαριθμητικά πεδία και στην έξοδο να βγάζει αριθμητικά. Όπως λέει και το όνομα του, «reusable transformation» επαναχρησιμοποιούμενος μετασχηματισμός, έχει φτιαχτεί μια φορά αλλά χρησιμοποιείται πολλές φορές και στα επόμενα mapping.

Τέλος το expression Finalize μαζεύει όλα τα πεδία και τα περνάει στον target πίνακα FIRM_STG.Παρακάτω βλέπουμε τα πεδία του FIRM_STG.



MRBL_01_Load_DAT_Parameter_files



Μετά από την ανάλυση του Firm_stg θα περάσω στην ανάλυση του πρώτου worklet που συναντάμε κατά την εκτέλεση τηςETL διαδικασία.

Το worklet MRBL_01_Load_DAT_Parameter_files περιέχει τους όλους τους παραμετρικούς πίνακες και είναι το πρώτο που εκτελείται. Τα παραμετρικά αρχεία περιέχουν δεδομένα αποκλειστικά τύπου varchar (χαρακτήρες κυμαινόμενου μήκους), με μέγιστο μέγεθος το αντίστοιχο Cobol αρχείο.

Λέγονται παραμετρικοί οι πίνακες γιατί περιέχουν ως επί το πλείστον περιγραφές των κωδικών. Πχ. ο πίνακας FMI51 περιέχει το Cod_ FMI51και την περιγραφή του Description_ FMI51.

```
select * from FMI51_DAT
```

	Company_ID_FMI51	Fiscal_Year_ID_FMI51	COD_FMI51	DESCRIPTION_FMI51
1	9	2,002	01	ΑΠΟΦΟΙΤΟΣ ΛΥΚΕΙΟΥ
2	9	2,002	02	ΑΠΟΦΟΙΤΟΣ ΑΝΩΤΕΡΑΣ
3	1	2,002	01	ΑΠΟΦΟΙΤΟΣ ΛΥΚΕΙΟΥ
4	1	2,002	02	ΑΠΟΦΟΙΤΟΣ ΑΝΩΤΕΡΑΣ

Οι παραμετρικοί πίνακες είναι οι πιο απλοί στην υλοποίηση τους. Τα δεδομένα τους δεν τροποποιούνται. Σε σύγκριση με τους πίνακες αναβάθμισης (stg) δεν περιέχουν καμία περίπλοκη λογική στην ροή τους. Ένα άλλο χαρακτηριστικό είναι ότι δεν θα βρούμε ποτέ πίνακα με πολλά πεδία, ο καθένας περιέχει λίγες κολώνες, συνήθως έναν κωδικό και την περιγραφή του. Λόγω του απλοϊκού τους χαρακτήρα οι πίνακες αυτοί δεν περνάνε από το stage στάδιο και έτσι οδηγούνται απευθείας στους τελικούς πίνακες (dat).

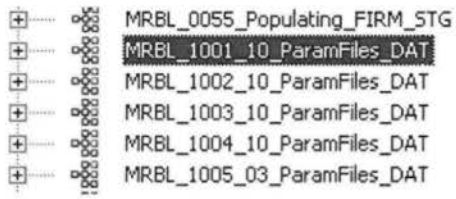
Στο σύνολο τους οι παραμετρικοί πίνακες είναι 44. Στον αριθμό φαίνεται ότι είναι 43 αλλά υπάρχουν δύο πίνακες που είναι 27A και 27B. Θα μπορούσαν να φτιαχτούν 44 mappings, δηλαδή ένα για τον κάθε πίνακα όπως γίνεται για τους stg και dat πίνακες. Για λόγους ευκολίας και συντομίας προτιμήθηκε να μπουν σε δεκάδες οι πίνακες για το κάθε mapping. Έτσι έχουν δημιουργηθεί τέσσερα mapping από δέκα πίνακες και ένα πέμπτο που περιέχει τους τελευταίους τρείς πίνακες.

Σχετικά με την ονοματολογία των παραμετρικών πινάκων: ξεκινάνε πάντα από MRBL_και συνεχίζουν με έναν τετραψήφιο αριθμό που χωρίζεται σε δύο μέρη. Το '1' σημαίνει ότι ανήκει στην ομάδα των Παραμετρικών αρχείων,

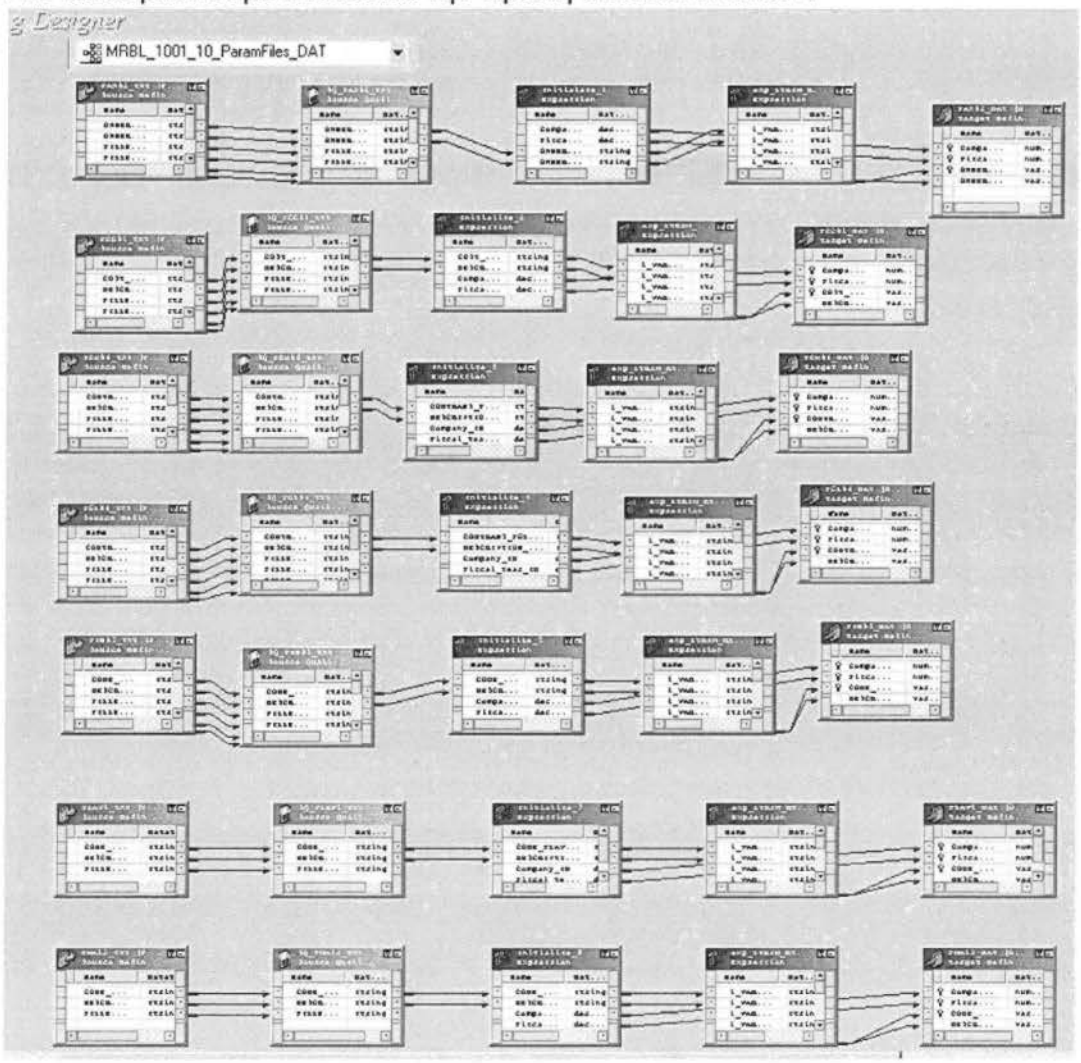
άλλωστε είναι και το πρώτο worklet που εκτελείται κατά την ETL διεργασίας. Ο τριψήφιος αριθμός στη συνέχεια απαριθμεί τα mapping, που είναι πέντε στο σύνολο.

Ο αριθμός 10 στην συνέχεια υποδεικνύει την περιεκτικότητα πινάκων σε ένα mapping. Τέλος έχουμε την γενική ονομασία των πινάκων που υποδηλώνουν και την ιδιότητα τους 'ParamFiles_DAT'.

Πχ. το 'MRBL_1001_10_ ParamFiles_DAT' σημαίνει ότι θα το συγκεκριμένο mapping επεξεργάζεται παραμετρικά αρχεία και περιέχει δέκα παραμετρικούς πίνακες.

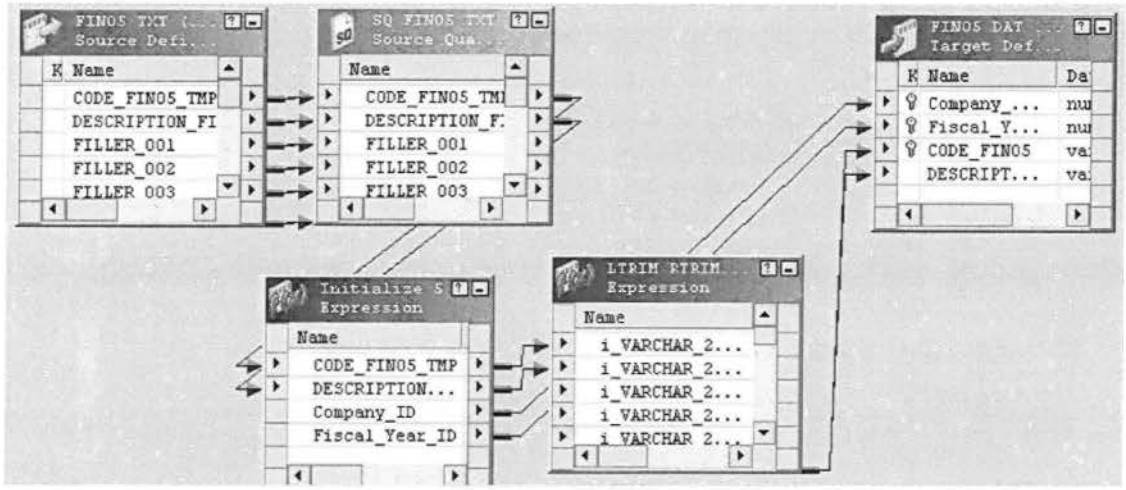


Πιο κάτω βλέπουμε αναλυτικά την πρώτη δεκάδα πινάκων.



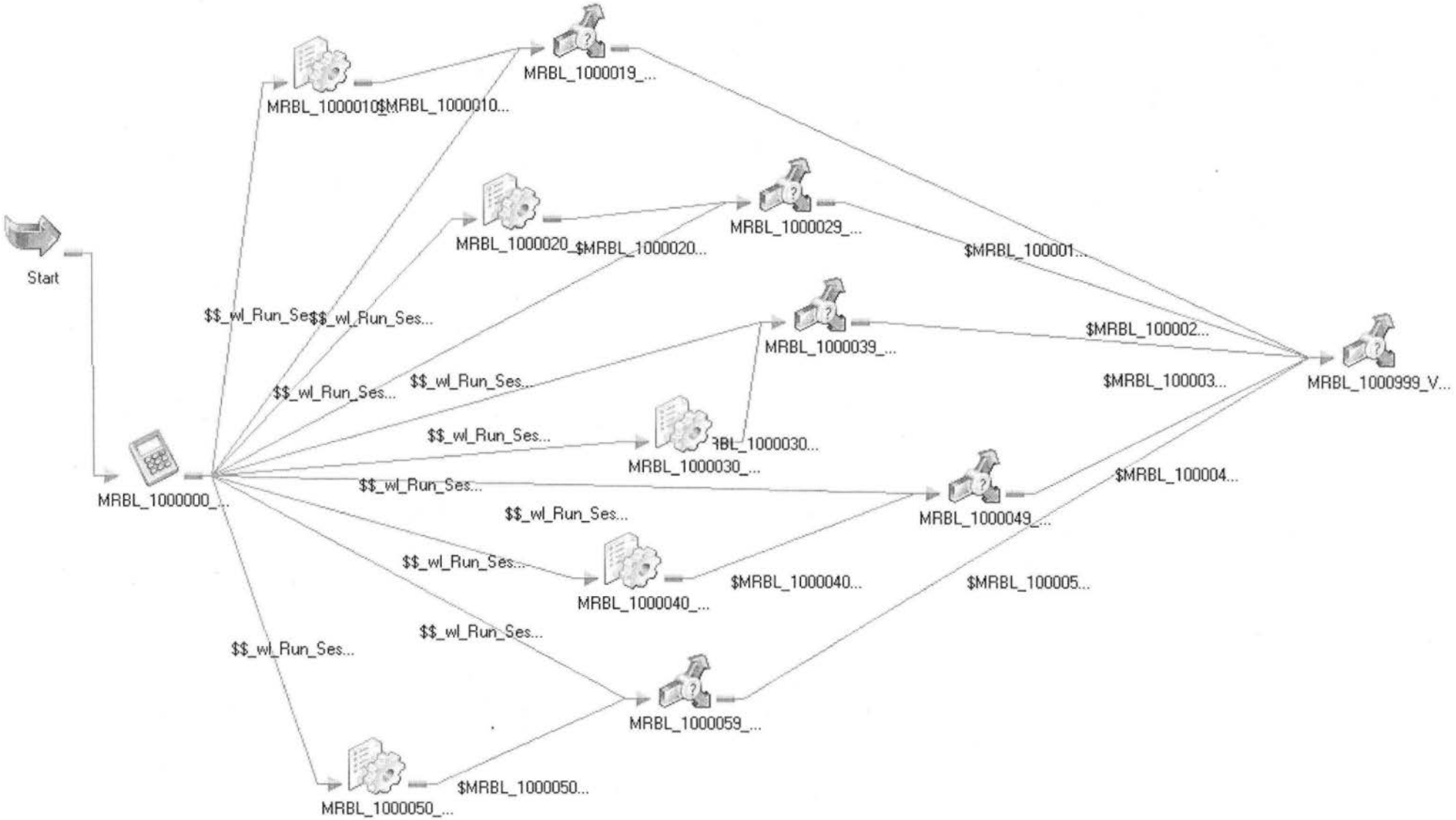
Τα δεδομένα διαβάζονται από το source και το κάθε πεδίο επεξεργάζεται με lefttrim-righttrim, προστίθενται δύο παραπάνω πεδία Fiscal_Year_ID και Company_ID και τέλος καταλήγουν στον target.

Πχ. FIN05_TXT περιέχει τα πεδία: Code_FIN05_TMP, Description_FIN05_TMP, FILLER_001, FILLER_002, FILLER_003. Ο τελικός πίνακας FIN05_DAT θα περιέχει δύο πεδία επιπλέον: Fiscal_Year_ID και Company_ID ενώ τα FILLER κόβονται.



FILLERs είναι κενά πεδία που έχουν προστεθεί στα cobol αρχεία σε περίπτωση που μελλοντικά το Description χρειαστεί να γίνει πιο μεγάλο απ' ότι είναι τώρα. Στους τελικούς πίνακες δεν χρειάζονται τα FILLERs γι' αυτό και παραλείπονται.

MRBL_01_Load_DAT_Parameter_files



Στο πιο πάνω screenshot βλέπουμε το worklet των Parameter Files σε πλήρη ανάπτυξη (expanded). Όπως περιγράφθηκε πιο πάνω από τα πέντε mappings έχουν δημιουργηθεί πέντε session. Επίσης είναι σημαντικό να σημειωθεί ότι τα πέντε session τρέχουν παράλληλα. Οπότε δεν ξέρουμε και δεν μας ενδιαφέρει με πια σειρά θα τρέξουν τα session.

Κάθε φορά που τρέχει το worklet των ParameterFiles είναι επιθυμητό να σβήνονται οι προηγούμενες εγγραφές (της προηγούμενης μέρας, γιατί δεν αλλάζουν). Η μόνη ιστορικότητα που κρατιέται είναι όταν αλλάξει ο χρόνος, δηλαδή να έχουμε διαφορετική λογιστική χρονιά (Fiscal_Year_ID). Συνεπώς εσωτερικά του κάθε sessionπρέπει να οριστεί ο τρόπος που θα γίνεται η διαγραφή των προηγούμενων εγγραφών. Είναι μια διαδικασία που χρήζει ιδιαίτερης προσοχής.

Την πρώτη φόρα που έτρεξαν τα sessions, χρησιμοποιήθηκε ως τρόπος γραφής στους target το external loader. Με την χρήση του external loader φτιάχνονται για κάθε session από ένα control (.ctl) και target (.trg) αρχείο. Το control αρχείο περιέχει πληροφορία για το target αρχείο, όπως τι δομή θα έχουν τα δεδομένα (data format) και οδηγίες για το πώς θα φορτώνονται τα δεδομένα με external loader.

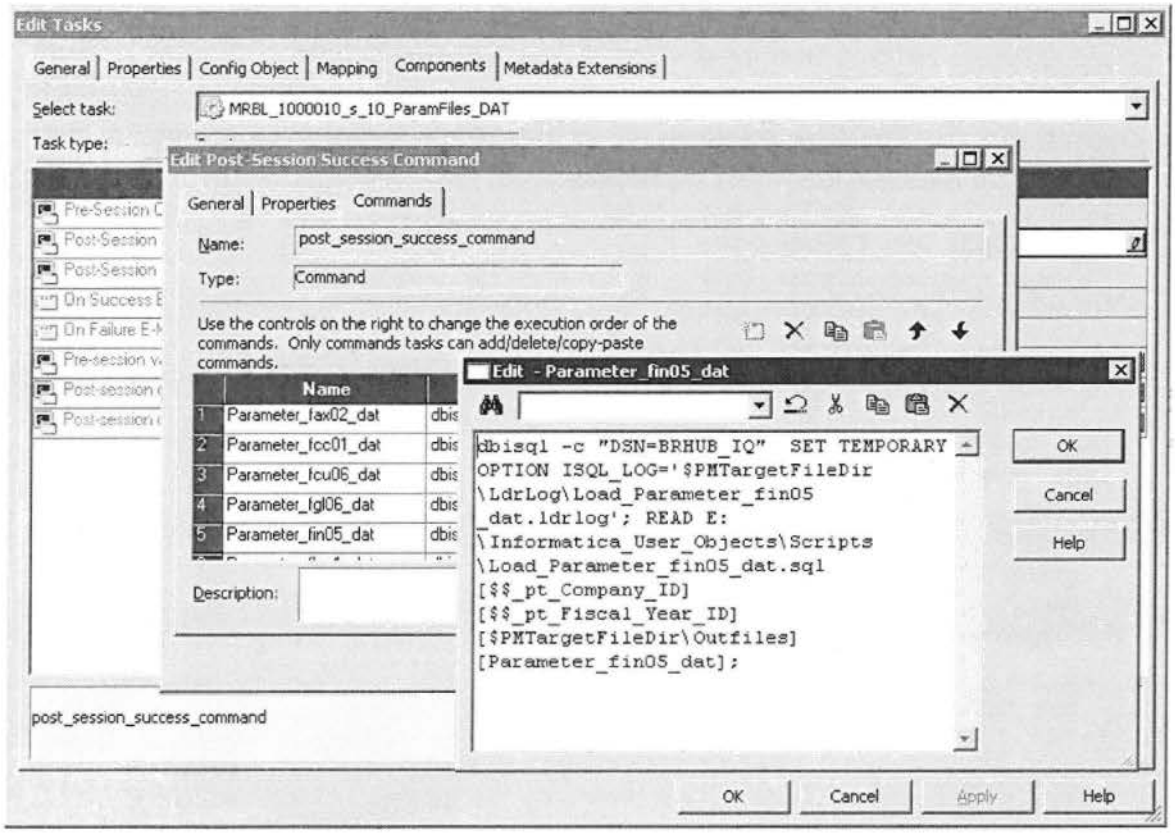
```
SET TEMPORARY OPTION ON_ERROR=EXIT;
SET TEMPORARY OPTION ISQL_LOG='E:\Informatica\9.1.0\server\infa_shared\
                                    TgtFiles\fmi51_dat.out.ldrlog';
SET TEMPORARY OPTION DATE_ORDER='MDY';

LOAD TABLE FMI51_DAT
(
Company_ID_FMI51                        ASCII(    4)  NULL('*   ') ,
Fiscal_Year_ID_FMI51                    ASCII(    6)  NULL('*     ') ,
COD_FMI51                               ASCII(    2)  NULL('* ') ,
DESCRIPTION_FMI51                       ASCII(   35)  NULL('*
                                        FILLER(2)) )
FROM 'E:\Informatica\9.1.0\server\infa_shared\TgtFiles\fmi51_dat.out'
WITH CHECKPOINT OFF
BLOCK FACTOR 10000
NOTIFY 1000
ESCAPES OFF
QUOTES OFF
ON FILE ERROR ROLLBACK
/* End of generated Control File */
```

Εκμεταλλεύτηκα το γεγονός ότι η Informatica δημιουργεί ctl file με δύο σκοπούς: πρώτον για version independence και δεύτερον για την διαγραφή των προηγούμενων εγγραφών. Το κάθε ctl file τροποποιήθηκε και μπήκαν εντολές delete from ...where... Επίσης οι πίνακες φορτώνονται πλέον από path που θα ορίζει ο χρήστης. Με λίγα λόγια το ctl file έχει γίνει όσο πιο ελεγχόμενο γίνεται, τίποτα πλέον δεν αφήνεται στην τύχη της Informatica.

Η τελευταία αλλαγή που έγινε είναι να γίνει το αρχείο read only ώστε να μην αλλάξει από την Informatica στο επόμενο φόρτωμα των πινάκων και άλλαξε η κατάληξη σε .sql. Την επόμενη φορά που θα τρέχουν τα session θα γίνει αλλαγή το external loader σε file reader.

```
PARAMETERS Par_Company_ID
          ,Par_Fiscal_Year
          ,Par_Outfile_Path
          ,Par_Outfile_Name
;

SET TEMPORARY OPTION ON_ERROR=EXIT;
SET TEMPORARY OPTION DATE_ORDER='MDY';

DELETE
  FROM FMI51_DAT
 WHERE Company_ID_FMI51      = {Par_Company_ID}
   AND Fiscal_Year_ID_FMI51 = {Par_Fiscal_Year}
;
LOAD TABLE FMI51_DAT
(
Company_ID_FMI51                          ASCII(    4)  NULL('*   ')
Fiscal_Year_ID_FMI51                      ASCII(    6)  NULL('*
COD_FMI51                                 ASCII(    2)  NULL('* ') ,
DESCRIPTION_FMI51                         ASCII(   35)  NULL('*
                                          FILLER(2))
FROM '{Par_Outfile_Path}\{Par_Outfile_Name}.out'
WITH CHECKPOINT OFF
BLOCK FACTOR 10000
NOTIFY 1000
ESCAPES OFF
QUOTES OFF
ON FILE ERROR ROLLBACK
/* End of generated Control File */
```

Ας επανέλθουμε από τα ctl files στο session. Εσωτερικά του κάθε session στο σημείο όπου ορίζουμε διάφορες εντολές που θέλουμε να εκτελέσει το session, (δηλαδή στο post session command) ορίστηκαν οι παράμετροι που χρειάζονται το ctl file. Αυτός είναι ο τρόπος να συνδέσουμε τα .ctl αρχεία που πλέον έχουν γίνει sql) με το session.



MRBL_02_Load_STG_files

Το επόμενο worklet που εκτελείται κατά σειρά στο γενικό workfloweίναι τοMRBL_02_Load_STG_files και περιέχει όλους τους stage πίνακες.

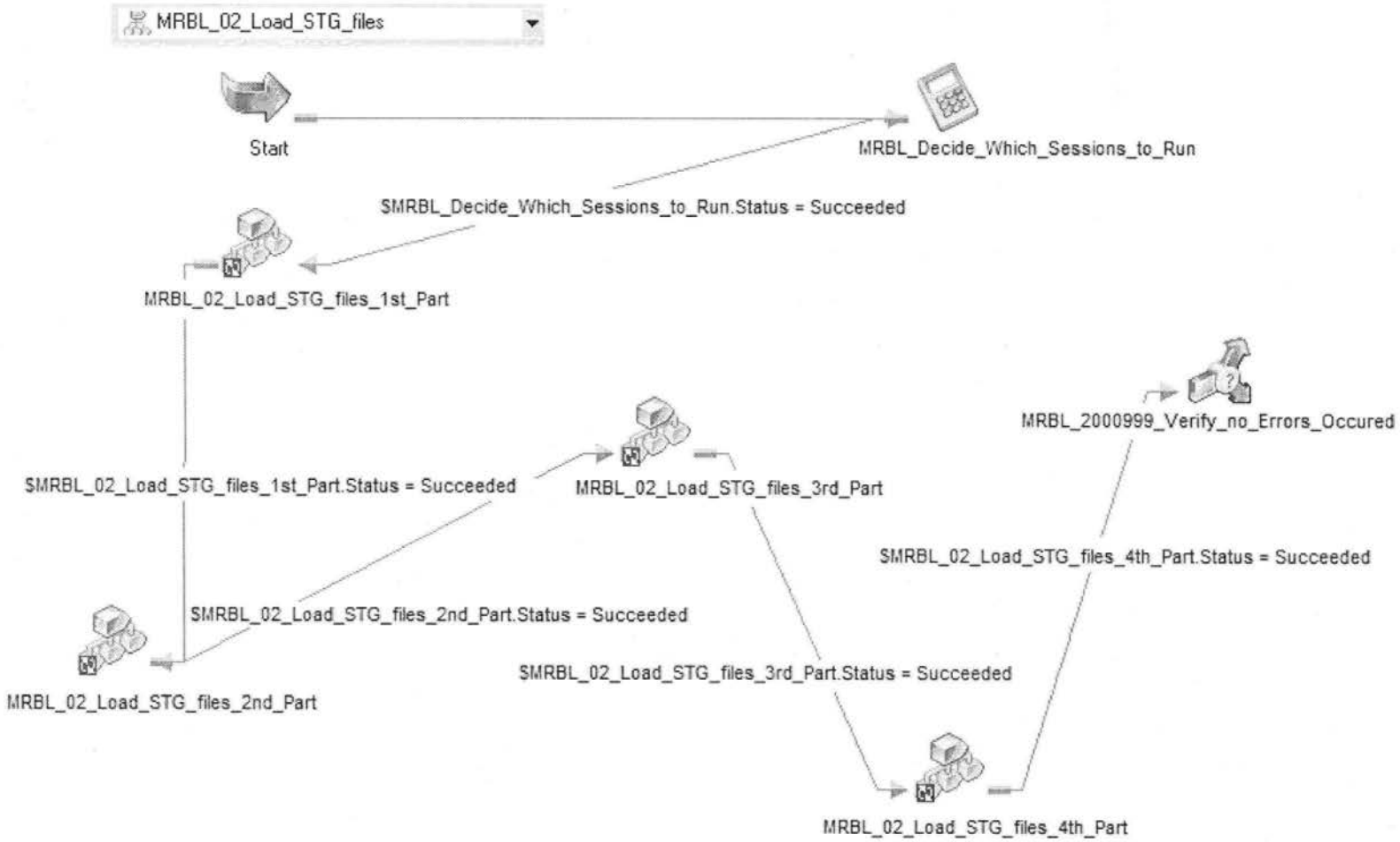
Λέγονται stage (μεταβατικοί) πίνακες γιατί αναφέρονται περισσότερο στο Transformation κομμάτι της ETL διαδικασίας. είναι το βήμα πριν την τελική φάση του φορτώματος. Όλοι οι stageπινάκες αυτοί έχουν κατάληξη _STG.

Στην αρχική υλοποίηση της ETL διαδικασίας με scripts υπήρχαν: οι αρχικοί πίνακες (txt) που έφερναν τα δεδομένα (loading), οι ενδιάμεσοι πίνακες (stg) όπου γινόταν η επεξεργασία και οι τελικοί (ιστορικοί) πίνακες (dat) όπου αποθηκεύονταν δεδομένα.

Λαμβανομένου υπόψη ότι η δική μου εργασία πραγματοποιήθηκε με βοήθεια ενός ETL εργαλείου θεωρήθηκε περιττό βήμα η δημιουργία των αρχικών πινάκων. Σε αντίθεση με τα sql scripts η Informatica κάνει το loading και transformation σε ένα βήμα. Δεν χρειάζεται κάποια ιδική διαδικασία για διαβαστούν τα δεδομένα από flat file, ενώ με τα scripts χρειαστήκαν πολλές σελίδες κώδικα και ξεχωριστοί πίνακες στην βάση. Ενώ η Informatica τα κάνει όλα με ευκολία μέσα σε ένα mapping.

Το αναπόφευκτο ήταν η δημιουργία των dat πινάκων. Στάθηκε αδύνατον να συγχωνευτεί και αυτό το βήμα μαζί με τα άλλα. Θα πρέπει πάντα να υπάρχουν οι μεταβατικοί πίνακες για λόγους ελέγχου σφαλμάτων και για λόγους πρακτικότητας. Είναι σαν πρόχειροι πίνακες τους οποίους χρησιμοποιούν οι τρείς εταιρίες ενώ στους dat πίνακες καθαρογράφουμε τα σωστά αποτελέσματα. Θα ήταν πολύ δύσκολο να γίνεται η επεξεργασία και το φόρτωμα των δεδομένων στους ίδιους πίνακες.

MRBL_02_Load_STG_files



Με το worklet MRBL_02_Load_STG_files στην ουσία τρέχουμε το πιο σημαντικό κομμάτι της ETL διαδικασίας. Βρισκόμαστε στην καρδία της ETL διαδικασίας μιας κι εδώ υλοποιείται η πολύπλοκη λογική της μετατροπής και μετασχηματισμού των δεδομένων.

Ανοίγοντας το worklet MRBL_02_Load_STG_files παρατηρούμε ότι υπάρχουν άλλα τέσσερα worklet μέσα. Το κάθε worklet περιέχει κάμποσα session και το κάθε session αντιστοιχεί σε ένα mapping. Προς το παρόν θα μιλήσω για woklets και sessions και αργότερα θα περιγραφτεί η υλοποίηση των mappings.

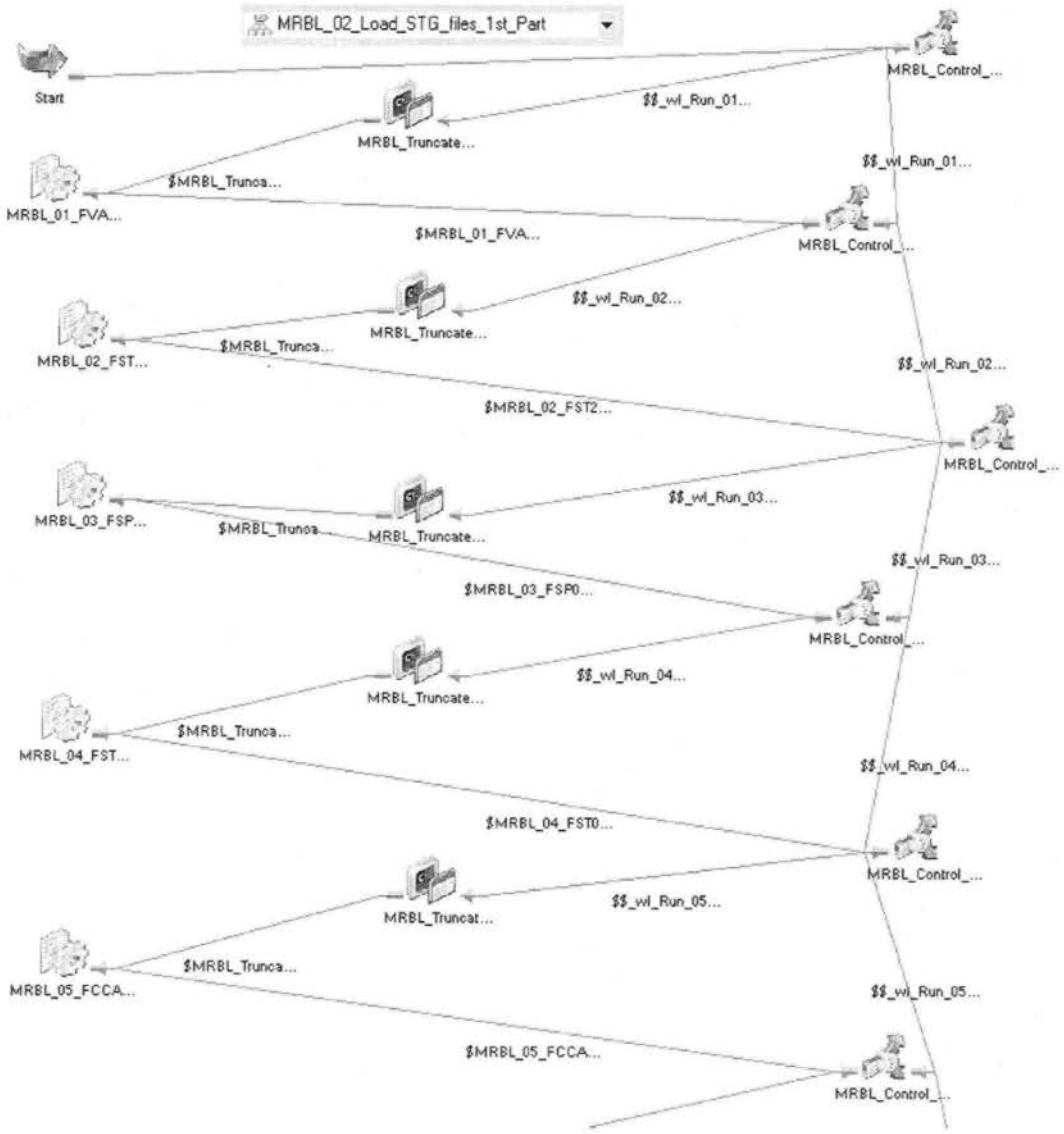
Γενικότερα κατά την διεκπεραίωση της εργασίας ακολουθήθηκε η τακτική worklet σε worklet. Αυτό έγινε για ταξινόμηση των session ανά ομάδες, επιτυγχάνοντας καλύτερη οπτική απεικόνιση και ευκολότερη διαχείριση των ροών. Επειδή έχουμε 38 mappings και για το κάθε πίνακα φτιάχτηκε ένα session, σύνολο έχουμε 38 sessions, οπτικά αυτό θα έδειχνε σαν ένα χάος από sessions, commands και decisions. Έτσι τα 38 sessions χωριστήκαν σε τέσσερα worklet περίπου δέκα το καθένα κάνοντας τον χειρισμό τους ευκολότερο.(βλ. screenshot πιο πάνω)

Επομένως έχουμε τέσσερα worklet, το MRBL_02_Load_STG_files_1st_Part, MRBL_02_Load_STG_files_2nd_Part, MRBL_02_Load_STG_files_3rd_Part, MRBL_02_Load_STG_files_4th_Part

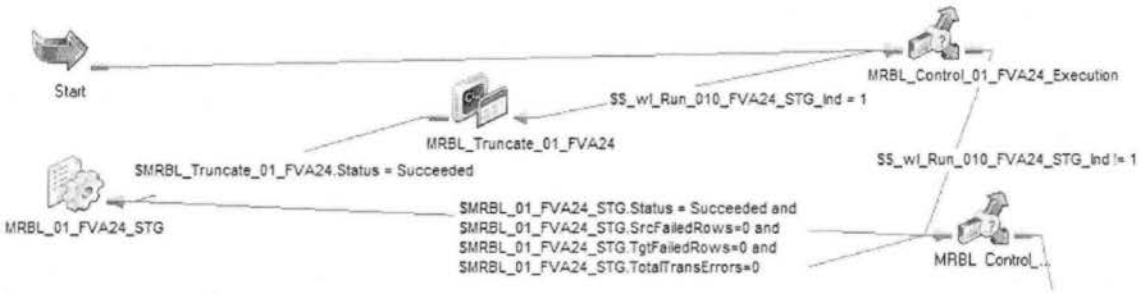
Είναι σημαντικό να επισημάνουμε ότι όλα τα worklet εκτελούνται το ένα μετά το άλλο σειριακά, το ίδιο συμβαίνει και για τα session που αυτά περιέχουν.

Τα worklet έχουν όλα την ίδια δομή, πιο συγκεκριμένα το κάθε worklet περιέχει από δέκα session εκτός από το τέταρτο που έχει τα οχτώ τελευταία., ας αναλύσουμε το πρώτο.

Το πρώτο worklet MRBL_02_Load_STG_files_1st_Part περιέχει τα πρώτα δέκα session. Η δομή είναι αρκετά απλή όπως φαίνεται πιο κάτω.



Με την τριγωνική απεικόνιση των session επιτεύχθηκε καλύτερος έλεγχος εκτέλεσης των ροών. Κάθε decision είναι ένα κομβικό σημείο, εκεί παίρνεται η απόφαση αν θα τρέξει το τρέχον session ή αν θα προχωρήσει στο επόμενο. Κάποιες φορές ήταν αναγκαία η παράκαμψη κάποιον session ώστε να τρέξει κάποιο συγκεκριμένο μόνο του. π.χ. κατά τον έλεγχο το πέμπτο session ήταν προβληματικό, για να μην τρέξουν ξανά τα προηγούμενα που ήταν σωστά υπάρχει αυτή η επιλογή της παράκαμψης. Παράκαμψη γίνεται με δείκτες που ορίζονται στο έξω worklet MRBL_02_Load_STG_files στο calculator Decide_which_session_to_run. Αν wl_Run.._STG_Ind = 1 τότε συνεχίζεται η

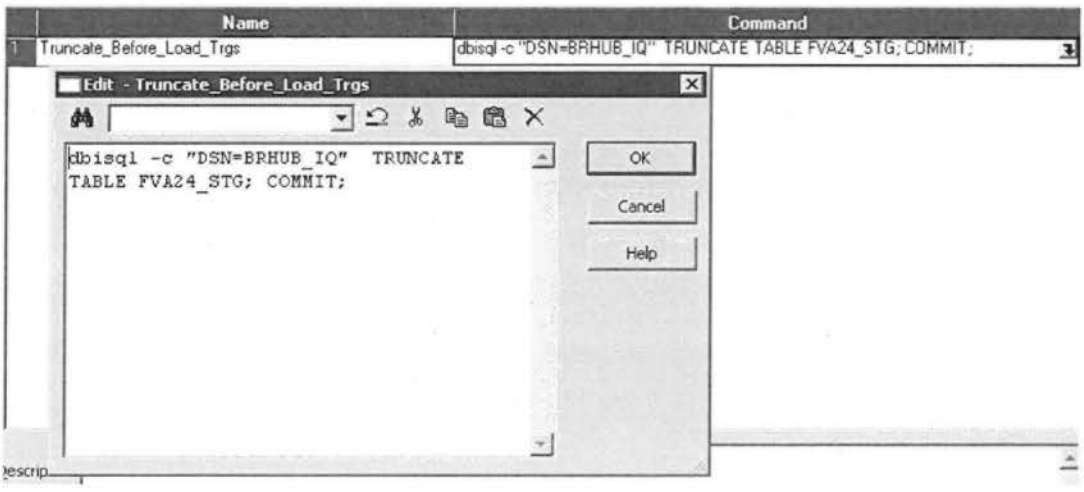


εκτέλεση του session αλλιώς το παρακάμπτει.

Στη συνέχεια αφού αποφασιστεί αν θα εκτελεστεί το session γίνεται εκτέλεση της command truncate. Σε γενικές γραμμές πριν από κάθε εκτέλεση ενός πίνακα εκτελείται η διαγραφή των περιεχομένων του. Με άλλα λόγια πριν από κάθε session έχει τοποθετηθεί ένα command που κάνει truncate των πινάκων.

TruncateCommand

Πιο κάτω βλέπουμε την sql εντολή που εκτελεί το command.

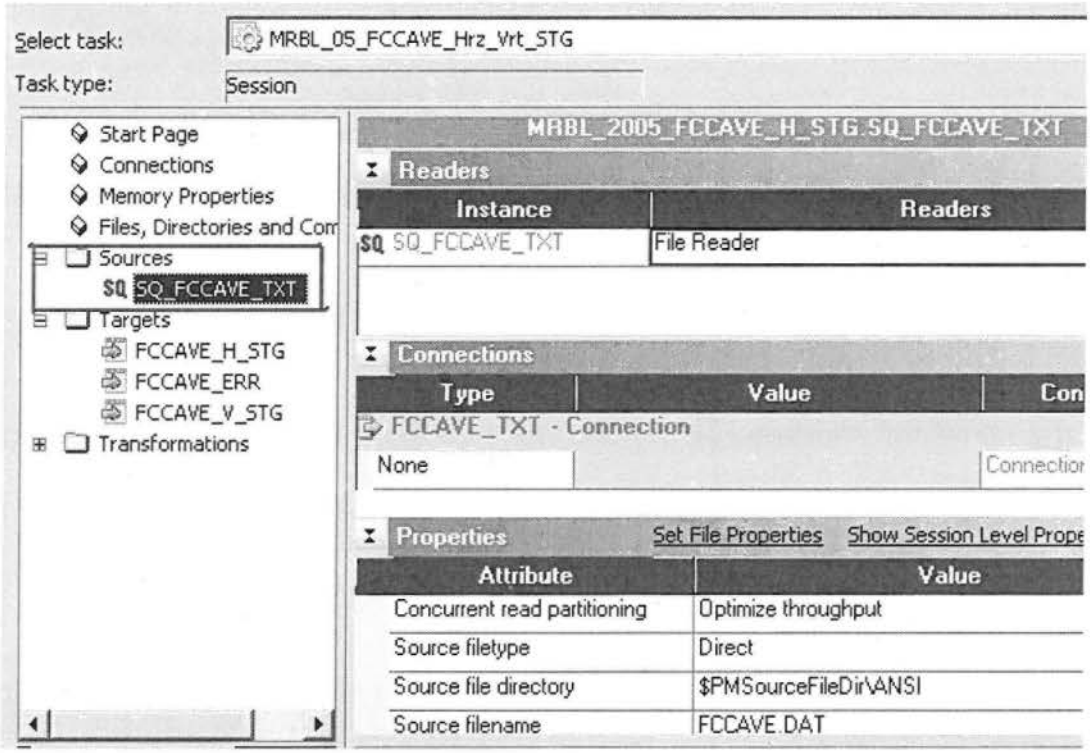


TruncateCommandEdit

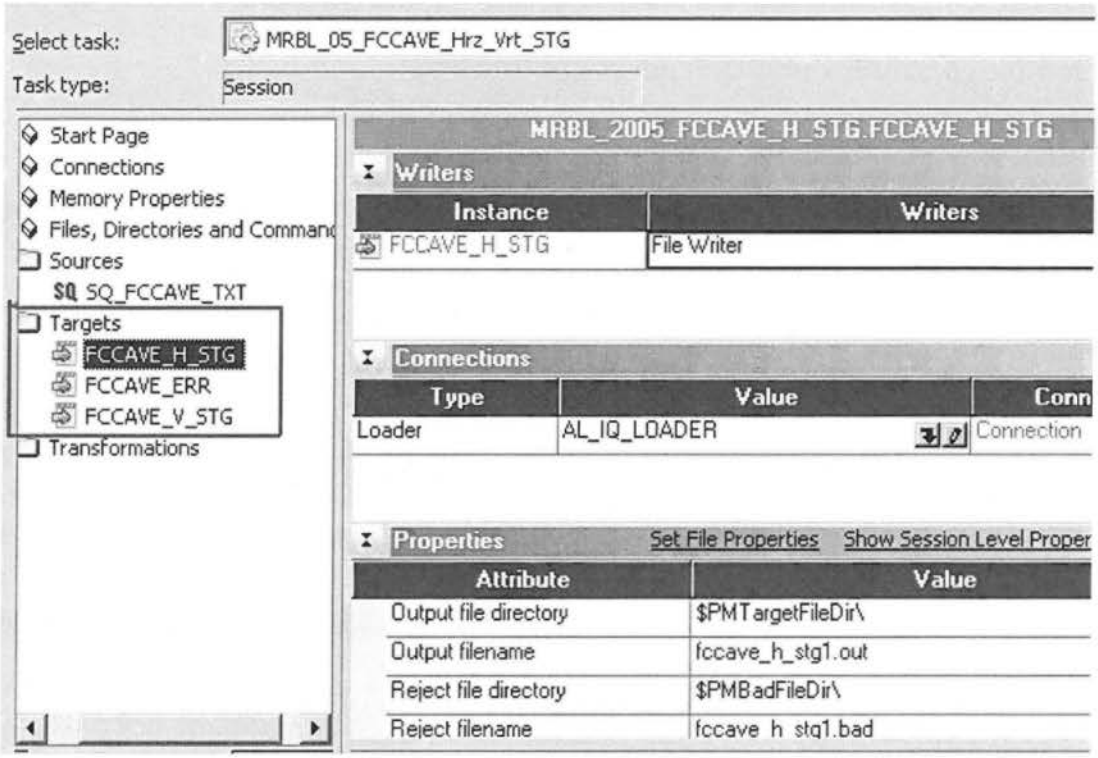
Ορισμός τύπου σύνδεσης στη Βάση Δεδομένων

MRBL_05_FCCAVE_Hrz_Vrt_STG Μέσα σε κάθε session ορίζεται ο τύπος της σύνδεσης που θα έχουν οι source και target πίνακες.

Για source ορίζουμε ότι ο τρόπος που θα διαβάζει τα δεδομένα θα είναι FileReader και σύνδεση στη βάση None, γιατί δεν διαβάζονται τα δεδομένα από τον IQ αλλά από το παραγωγικό σύστημα.



Για target ορίζουμε ότι θα γράφει τα δεδομένα με FileWriter και η σύνδεση που θα χρησιμοποιεί για να συνδέεται με τον IQ να είναι Loader (AL_IQ_LOADER).



Στο συγκεκριμένο παράδειγμα το mapping FCCAVE έχει τρείς target. Εάν για οριζόντια διάταξη, έναν για κάθετη παρουσίαση των δεδομένων και έναν πίνακα όπου γράφονται τα λάθη του mapping. Πρέπει για κάθε target ξεχωριστά να οριστεί ο τρόπος σύνδεσης με την βάση, γιατί αποτελούν ξεχωριστούς πίνακες παρόλο που βρίσκονται στο ίδιο mapping

Τέλος για σε κάθε session γίνεται ανάθεση παραμέτρων.

	Mapping Variables/Parameters	Operator	Parent Workflow/Worklet Variables
1	$$Company_ID	=	$$_pt_Company_ID
2	$$Fiscal_Year_ID	=	$$_pt_Fiscal_Year_ID

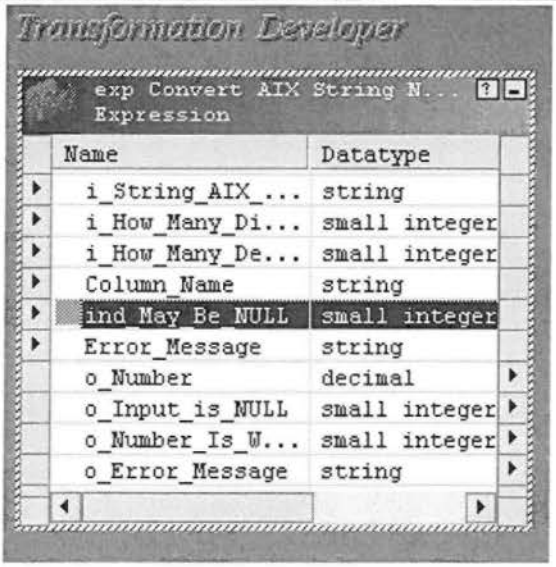
Reusable Transformations
(επαναχρησιμοποιούμενοι μετασχηματισμοί)

Κατά την διάρκεια δημιουργίας των mapping κάποια κομμάτια κώδικα χρειάστηκε να χρησιμοποιηθεί πάνω από μια φορά. Μια λύση θα ήταν αντιγραφή των κομματιών αυτών κάθε φορά που ήταν αναγκαίο. Ωστόσο προέκυπτε το πρόβλημα ότι αν ήθελα να κάνω κάποια μετατροπή στο μέλλον έπρεπε να πάω σε όλα τα mapping και σε όλα τα σημεία όπου χρησιμοποιήθηκε και να κάνω την αλλαγή χειροκίνητα. Συνεπώς αυξάνεται ο κίνδυνος αφενός να χάσω ένα από αυτά τα transformation και αφετέρου να κάνω κάπου λάθος.

Για το λόγο αυτό δημιουργήθηκαν τα λεγόμενα reusable transformation. Δημιουργούμε ένα transformation και το επαναχρησιμοποιούμε όσες φορές χρειάζεται μέσα στον κώδικα. Όταν χρειαστεί να γίνει οποιαδήποτε αλλαγή στο transformation τότε το μόνο που χρειάζεται είναι να πάμε στο Transformation developer και να κάνουμε την τροποποίηση. Αυτομάτως τροποποιούνται και όλα τα σημεία όπου αυτό το transformation χρησιμοποιήθηκε. Είναι σαν να ορίζουμε παραμέτρους στην αρχή του κώδικα.

Στη συγκεκριμένη εργασία δημιουργήθηκαν έξη reusable transformations. Και είναι τα εξής:

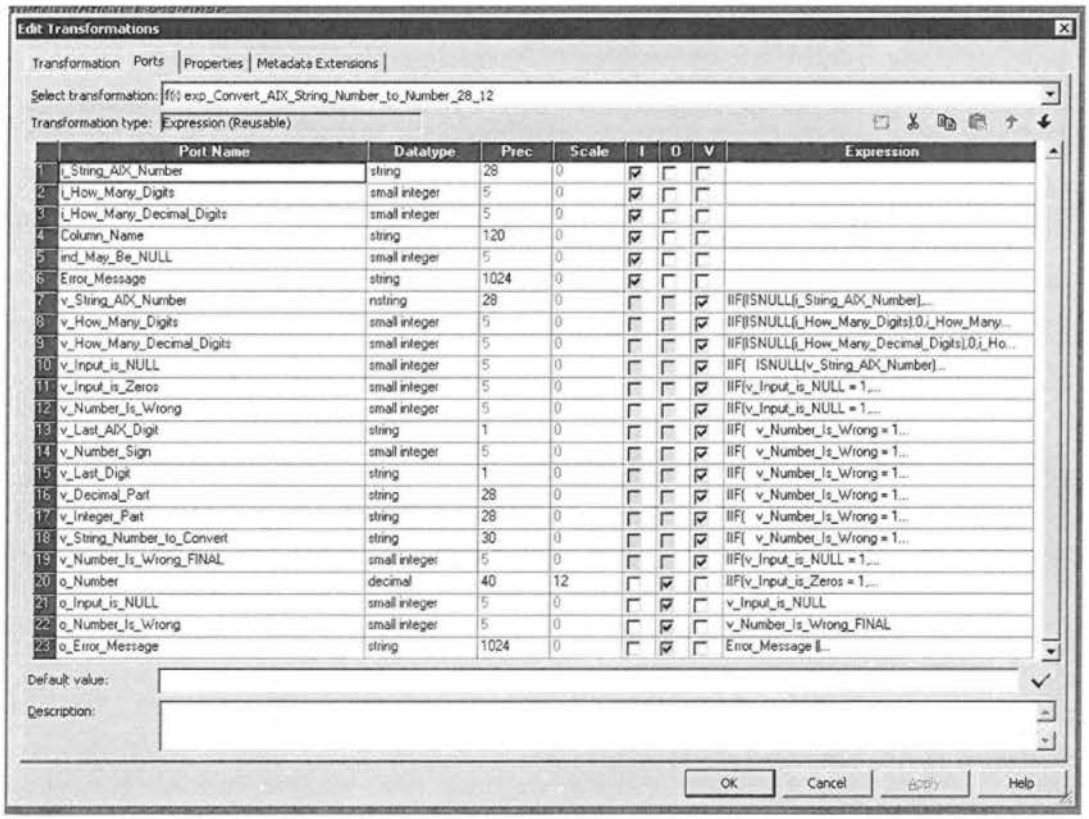
exp_Convert_AIX_String_Number_to_Number_28_12



Μετατρέπει ένα string – numberσενumber. Το reusable παίρνει ως input το string προς επεξεργασία, έναν αριθμό που δείχνει από πόσα ακέραια ψηφία θα αποτελείται, έναν για τα δεκαδικά ψηφία, ένα string "Column_Name", έναν δείκτη αν επιτρέπεται ή όχι να έχουμε null είσοδο και τέλος ένα string "Error_Message".

Στη συνέχεια μέσα από τις μεταβλητές (variables) γίνονται κάποιοι υπολογισμοί και τα αποτελέσματα γράφονται στις εξόδους (outputs). Κατά την επεξεργασία το string διαλύεται σε κομμάτια και ξαναφτιάχνεται με δομή τύπου decimal.

Ως εξόδους έχουμε "Number" ο επιθυμητός αριθμός με ακέραιο μέρος, υποδιαστολή και δεκαδικό μέρος, "Number_is_Wrong" δείχνει αν κατά την μετατροπή προέκυψε κάποιο λάθος, "Error_Message" εδώ γράφονται το όνομα της κολόνας που έγινε το λάθος, "Input_is_Null" ένας δείκτης αν το reusable δέχτηκε Null input.



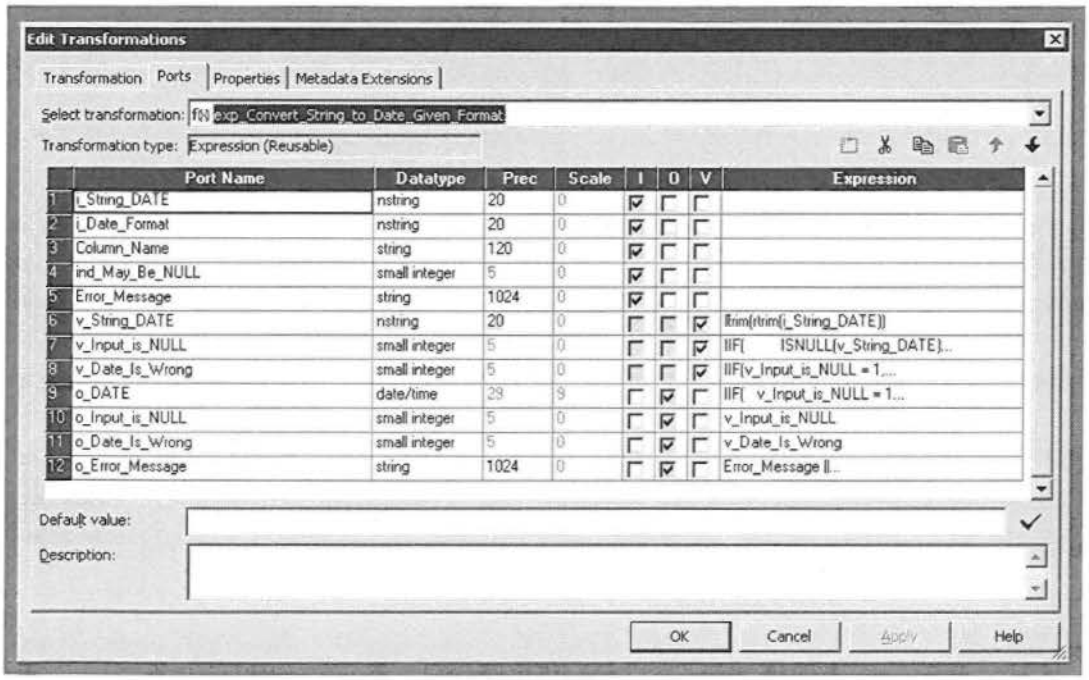
Ο κώδικας αναλυτικά:

i_String_AIX_Number	string(28)	
i_How_Many_Digits	small integer(5)	
i_How_Many_Decimal_Digits	small integer(5)	
Column_Name	string(120)	
ind_May_Be_NULL	small integer(5)	
Error_Message	string(1024)	
v_String_AIX_Number	string(28)	IIF(ISNULL(i_String_AIX_Number), '', ltrim(rtrim(i_String_AIX_Number)))
v_How_Many_Digits	small integer(5)	IIF(ISNULL(i_How_Many_Digits),0,i_How_Many_Digits)
v_How_Many_Decimal_Digits	small integer(5)	IIF(ISNULL(i_How_Many_Decimal_Digits),0,i_How_Many_Decimal_Digits)
v_Input_is_NULL	small integer(5)	IIF(ISNULL(v_String_AIX_Number) OR v_String_AIX_Number = '', 1, 0)
v_Input_is_Zeros	small integer(5)	IIF(v_Input_is_NULL = 1, 0 , IIF(REPLACECHR(0,v_String_AIX_Number,'0','')='',1, 0))
v_Number_Is_Wrong	small integer(5)	IIF(v_Input_is_NULL = 1, IIF(NOT ISNULL(ind_May_Be_NULL) ANDind_May_Be_NULL = 1, 0, 1) , IIF(length(v_String_AIX_Number) <>v_How_Many_Digits OR v_How_Many_Decimal_Digits>v_How_Many_Digits OR v_How_Many_Digits = 0 OR v_How_Many_Digits> 28 OR v_How_Many_Decimal_Digits> 12 OR v_How_Many_Decimal_Digits< 0, 1, 0))
v_Last_AIX_Digit	string(1)	IIF(v_Number_Is_Wrong = 1 OR v_Input_is_Zeros = 1, '' , SUBSTR(v_String_AIX_Number, -1, 1))
v_Number_Sign	small integer(5)	IIF(v_Number_Is_Wrong = 1 OR v_Input_is_Zeros = 1, 1 , IIF(v_Last_AIX_Digit>= 'p' AND v_Last_AIX_Digit<= 'y', -1 , 1))
v_Last_Digit	string(1)	IIF(v_Number_Is_Wrong = 1 OR v_Input_is_Zeros = 1, '' , DECODE(v_Last_AIX_Digit,'p','0','q','1' ,'r','2','s','3','t','4','u','5','v','6' ,'w','7','x','8' ,'y','9', v_Last_AIX_Digit))
v_Decimal_Part	string(28)	IIF(v_Number_Is_Wrong = 1 OR v_Input_is_Zeros = 1, '' , IIF(v_How_Many_Decimal_Digits = 0, '' , SUBSTR(v_String_AIX_Number, (v_How_Many_Decimal_Digits * -1) , (v_How_Many_Decimal_Digits - 1)) || v_Last_Digit))
v_Integer_Part	string(28)	IIF(v_Number_Is_Wrong = 1 OR v_Input_is_Zeros = 1, '' , IIF(v_How_Many_Decimal_Digits> 0, SUBSTR(v_String_AIX_Number, 1, (v_How_Many_Digits - v_How_Many_Decimal_Digits)) , SUBSTR(v_String_AIX_Number, 1, (v_How_Many_Digits - 1)) || v_Last_Digit))
v_String_Number_to_Convert	string(30)	IIF(v_Number_Is_Wrong = 1 OR v_Input_is_Zeros = 1, '' , IIF(v_Number_Sign = -1, '-' , '') || IIF(v_How_Many_Decimal_Digits> 0, v_Integer_Part || '.' || v_Decimal_Part , v_Integer_Part))
v_Number_Is_Wrong_FINAL	small integer(5)	IIF(v_Input_is_NULL = 1, IIF(ind_May_Be_NULL = 1, -- Allowed to be 0 , 1) , IIF(v_Number_Is_Wrong = 1, 1 , IIF(v_Input_is_Zeros = 1, 0 , IIF(IS_NUMBER(v_String_Number_to_Convert), 0 , 1))))
o_Number	decimal(40)	IIF(v_Input_is_Zeros = 1, 0 , IIF(v_Input_is_NULL = 1, IIF(NOT ISNULL(ind_May_Be_NULL) AND ind_May_Be_NULL = 1, 0 , NULL) , IIF(v_Number_Is_Wrong_FINAL = 1, NULL, TO_DECIMAL(v_String_Number_to_Convert))))
o_Input_is_NULL	small integer(5)	v_Input_is_NULL
o_Number_Is_Wrong	small integer(5)	v_Number_Is_Wrong_FINAL
o_Error_Message	string(1024)	Error_Message || IIF(v_Number_Is_Wrong_FINAL = 0, '' , IIF(Error_Message = '','',')|| 'Wrong: ' || Column_Name)

- exp Convert String to Date Given Format

Το συγκεκριμένο reusable μετατρέπει ένα string σε date, με μόνη προϋπόθεση να ξέρουμε από πριν το σχήμα της ημερομηνίας. Εδώ θα είναι πάντα "YYYYMMDD".

Ο κώδικας που υλοποιείται μέσα στο reusable.Πιο πάνω βλέπουμε τις εξόδους και τις εισόδους που έχει. Είναι σημαντικό να σημειώσουμε ότι μετά από κάθε reusable γίνεται έλεγχος σφαλμάτων.

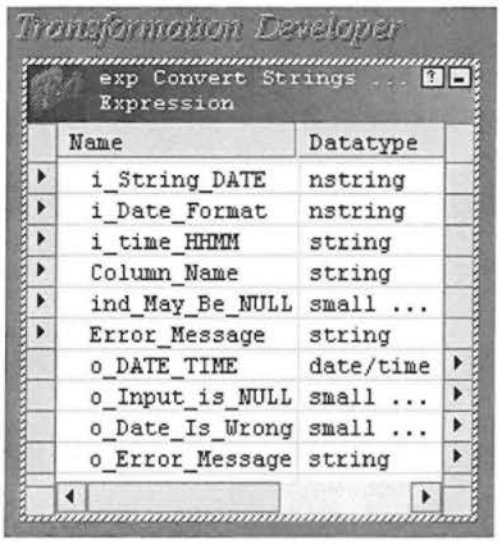


Αναλυτικά ο κώδικας του reusable transformation:

i_String_DATE	nstring(20)	
i_Date_Format	nstring(20)	
Column_Name	string	
ind_May_Be_NULL	small integer(5)	
Error_Message	string(1024)	
v_String_DATE	nstring(20)	ltrim(rtrim(i_String_DATE))
v_Input_is_NULL	small integer(5)	IIF(ISNULL(v_String_DATE) OR v_String_DATE = '' OR REPLACECHR(0,v_String_DATE, '0', '') = '' -- All Zeros or ISNULL(i_Date_Format), 1 , 0)
v_Date_Is_Wrong	small integer(5)	IIF(v_Input_is_NULL = 1, IIF(ind_May_Be_NULL = 1, -- Allowed to be 0, 1) , IIF(IS_DATE(v_String_DATE,i_Date_Format), 0, 1))
o_DATE	date/time(29)	IIF(v_Input_is_NULL = 1 OR v_Date_Is_Wrong = 1, NULL , TO_DATE(v_String_DATE,i_Date_Format))
o_Input_is_NULL	small integer(5)	v_Input_is_NULL
o_Date_Is_Wrong	small integer(5)	v_Date_Is_Wrong
o_Error_Message	string(1024)	Error_Message || IIF(v_Date_Is_Wrong = 0, '' , IIF(Error_Message = '','', ') || 'Wrong: ' || Column_Name || ' (Format: ' || i_Date_Format || ')')

- exp_Convert_String_to_DateTime_Given_Format

Σε κάποιες περιπτώσεις ήταν αναγκαίο να υπολογιστεί η ώρα έναρξης και ώρα λήξεις κάποιων διαδικασιών. Π.χ. το mapping FAL03_STG φτιάχνει τις κολόνες START_TIME_FLA03 και FINISH_TIME_FLA03 με το reusable αυτό για να υπολογίσει στην συνέχεια την κολόνα Minute_Duration_FLA03 (χρονική διάρκεια) με την εντολή
ROUND(DATE_DIFF(FINISH_TIME_FLA03,START_TIME_FLA03,'MI'),0). Για το λόγο αυτό δημιουργήθηκε το συγκεκριμένο reusable που από string υπολογίζει και την ημερομηνία και την ώρα.



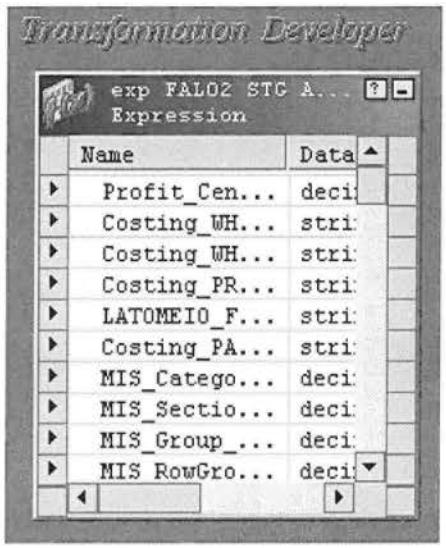
Ο κώδικας διαφέρει ελάχιστα με τον κώδικα του exp_Convert_String_to_Date_Given_Format. Άλλωστε η μόνη διαφορά είναι ο επιπρόσθετος υπολογισμός ώρας.

Αναλυτικά ο κώδικας υλοποίησης:

i_String_DATE	nstring(20)	
i_Date_Format	nstring(20)	
i_time_HHMM	string(4)	
Column_Name	string(120)	
ind_May_Be_NULL	smallinteger (5)	
Error_Message	string(1024)	
v_String_DATE	nstring(20)	ltrim(rtrim(i_String_DATE))
v_time_HHMM	string(4)	ltrim(rtrim(i_time_HHMM))
v_Input_is_NULL	small integer(5)	IIF(ISNULL(v_String_DATE) OR v_String_DATE = '' OR REPLACECHR(0,v_String_DATE, '0', '') = '' -- All Zeros or ISNULL(i_Date_Format)or ISNULL(v_time_HHMM) or v_time_HHMM = '', 1 , 0)
v_Time_is_Wrong	small integer(5)	IIF(ISNULL(v_time_HHMM)OR v_time_HHMM = '' or length(v_time_HHMM) <> 4, 1 , 0)
v_time_HH_MM_SS_Final	string(10)	IIF(v_Time_is_Wrong = 1, NULL , IIF(SUBSTR(v_time_HHMM, 1, 2) = '24', '00', SUBSTR(v_time_HHMM, 1, 2)) || ':' || SUBSTR(v_time_HHMM, 3, 2) || ':00')
v_String_DATE_Final	nstring(30)	v_String_DATE || ' ' || v_time_HH_MM_SS_Final
v_Date_Format_Final	nstring(32)	i_Date_Format || ' HH24:MI:SS'
v_Date_Is_Wrong	small integer(5)	IIF(v_Input_is_NULL = 1, IIF(ind_May_Be_NULL = 1, -- Allowed to be 0 , 1) , IIF(IS_DATE(v_String_DATE_Final,v_Date_Format_Final), 0 , 1))
o_DATE_TIME	date/time(29)	IIF(v_Input_is_NULL = 1OR v_Date_Is_Wrong = 1, NULL , TO_DATE(v_String_DATE_Final,v_Date_Format_Final))
o_Input_is_NULL	small integer(5)	v_Input_is_NULL
o_Date_Is_Wrong	small integer(5)	v_Date_Is_Wrong
o_Error_Message	string(1024)	Error_Message || IIF(v_Date_Is_Wrong = 0, '' , IIF(Error_Message = '','', ') || 'Wrong: ' || Column_Name || ' (Format: ' || v_Date_Format_Final || ')')

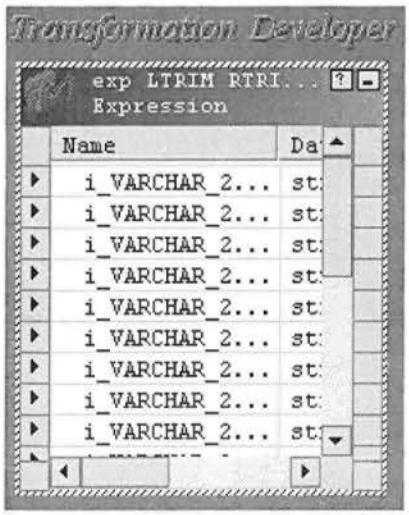
- exp_FAL02_STG_Additional_Dimensions

Το mapping FAL02_STG από τα πιο δύσκολα στην υλοποίηση. Είχε πολλές διαστάσεις που υπολογίζονταν με χρήση lookup transformation. Επειδή γινόταν πολλαπλή επανάληψη μεγάλου μέρους του κώδικα, αποφασίστηκε να γίνει ένα reusable που θα χρησιμοποιηθεί μόνο για το συγκεκριμένο mapping.



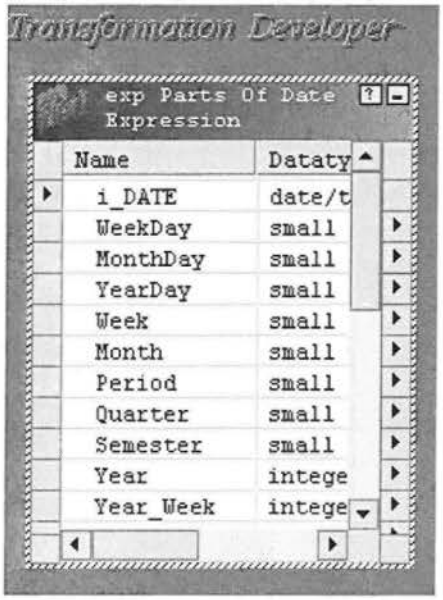
- exp_LTRIM_RTRIM

Δημιουργήθηκε ένα reusable που παίρνει δέκα string ως εισόδους κόβει δεξιά (right trim)και αριστερά (left trim) τα περιττά κενά αν υπάρχουν και καθαρά πλέον τα βγάζει ως εξόδους. Είναι το πιο πολυχρησιμοποιημένο reusable απ' όλα, δεν παραλείπεται από κανέναν mapping. Η ενοτλή που εκτελεί για κάθε γραμμή είναι ltrim(rtrim(string)) .

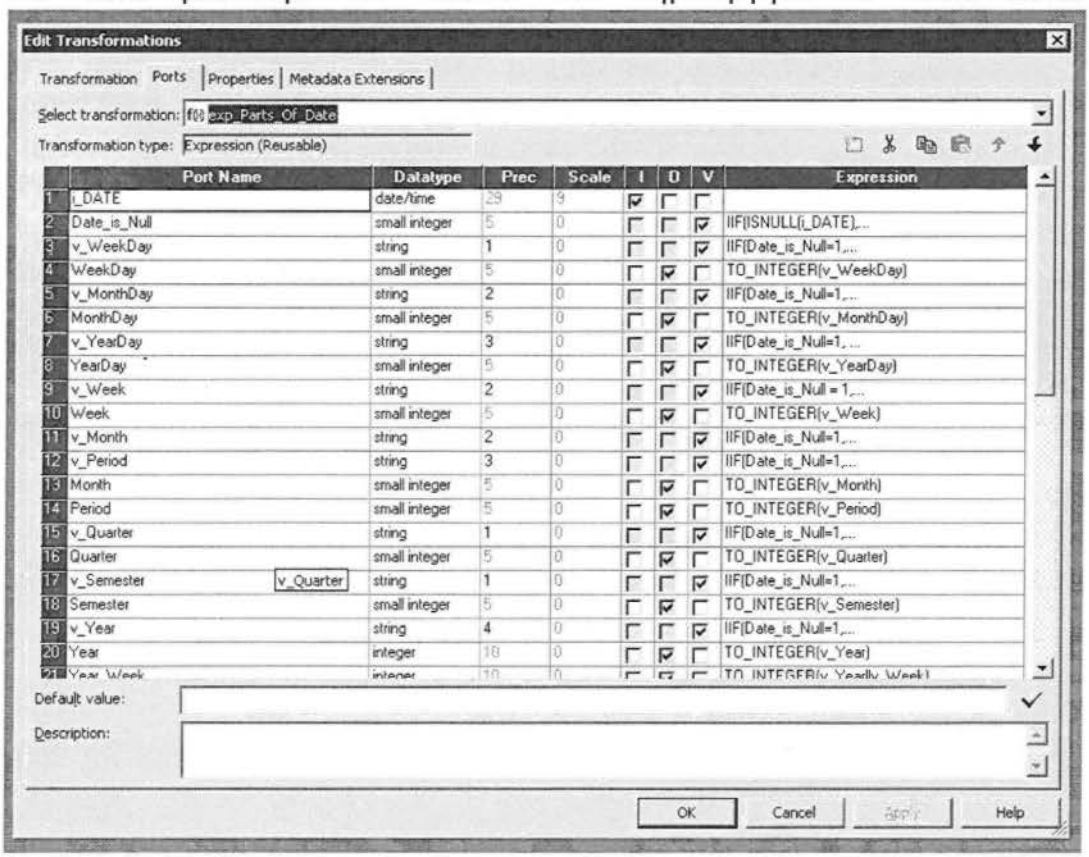


➢ exp Parts Of Date

Κάποιες ημερομηνίες συναλλαγών (transaction date) απαιτούσαν δημιουργία παραπάνω πεδίων που δίνουν παραπάνω λεπτομέρειες για την ημερομηνία αυτή. Τέτοιες λεπτομέρειες είναι εβδομάδα, μήνας, περίοδος του χρόνου κ.α.



Πιο κάτω βλέπουμε τον κώδικα που δημιούργησε τα πεδία αυτά.



Ο κώδικας αναλυτικά:

i_DATE	date/time(29)	
Date_is_Null	small integer(5)	IIF(ISNULL(i_DATE), 1, 0)
v_WeekDay	string(1)	IIF(Date_is_Null=1, NULL, TO_CHAR(i_DATE, 'D'))
WeekDay	small integer(5)	TO_INTEGER(v_WeekDay)
v_MonthDay	string(2)	IIF(Date_is_Null=1, NULL , (TO_CHAR(i_DATE, 'DD')))
MonthDay	small integer(5)	TO_INTEGER(v_MonthDay)
v_YearDay	string(3)	IIF(Date_is_Null=1, NULL , LPAD(TO_CHAR(i_DATE, 'DDD'),3,'0'))
YearDay	small integer(5)	TO_INTEGER(v_YearDay)
v_Week	string(2)	IIF(Date_is_Null = 1, NULL , LPAD(TO_CHAR(i_DATE, 'WW'), 2, '0'))
Week	small integer(5)	TO_INTEGER(v_Week)
v_Month	string(2)	IIF(Date_is_Null=1, NULL , LPAD(TO_CHAR(i_DATE, 'MM'), 2, '0'))
v_Period	string(3)	IIF(Date_is_Null=1, NULL , LPAD(TO_CHAR(i_DATE, 'MM'), 3, '0'))
Month	small integer(5)	TO_INTEGER(v_Month)
Period	small integer(5)	TO_INTEGER(v_Period)
v_Quarter	string(1)	IIF(Date_is_Null=1, NULL , TO_CHAR(i_DATE, 'Q'))
Quarter	small integer(5)	TO_INTEGER(v_Quarter)
v_Semester	string(1)	IIF(Date_is_Null=1, NULL , IIF(v_Quarter='1' ORv_Quarter='2', '1', '2'))
Semester	small integer(5)	TO_INTEGER(v_Semester)
v_Year	string(4)	IIF(Date_is_Null=1, NULL , TO_CHAR(i_DATE, 'YYYY'))
Year	integer(10)	TO_INTEGER(v_Year)
Year_Week	integer(10)	TO_INTEGER(v_Year||v_Week)
Year_Month	integer(10)	TO_INTEGER(v_Year||v_Month)
Year_Period	integer(10)	TO_INTEGER(v_Year||LPAD(v_Month,3,'0'))
Year_Quarter	integer(10)	TO_INTEGER(v_Year || v_Quarter)
Year_Semester	integer(10)	TO_INTEGER(v_Year || v_Semester)
v_Year_NextWeek_Date	date/time(29)	IIF(Date_is_Null = 1, NULL , ADD_TO_DATE(i_DATE, 'DD', 7))
Year_NextWeek	integer(10)	IIF(Date_is_Null=1, NULL , TO_INTEGER(TO_CHAR(v_Year_NextWeek_Date, 'YYYY') || LPAD(TO_CHAR(v_Year_NextWeek_Date, 'WW'), 2, '0')))
v_Year_NextMonth_Date	date/time(29)	IIF(Date_is_Null = 1, NULL , ADD_TO_DATE(i_DATE, 'MM', 1))
Year_NextMonth	integer(10)	IIF(Date_is_Null=1, NULL, TO_INTEGER(TO_CHAR(v_Year_NextMonth_Date, 'YYYY') || LPAD(TO_CHAR(v_Year_NextMonth_Date, 'MM'), 2, '0')))
Year_NextPeriod	integer(10)	IIF(Date_is_Null=1, NULL , TO_INTEGER(TO_CHAR(v_Year_NextMonth_Date, 'YYYY') || LPAD(TO_CHAR(v_Year_NextMonth_Date, 'MM'), 3, '0')))

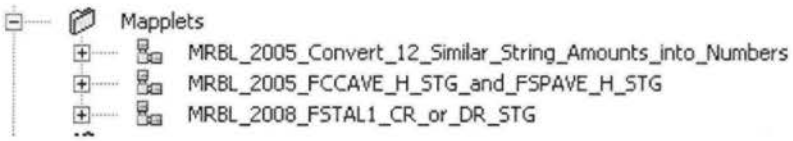
v_Year_NextQuarter_Date	date/time(29)	IIF(Date_is_Null = 1, NULL , ADD_TO_DATE(i_DATE, 'MM', 3))
v_Year_NextQuarter_Quarter	string(1)	IIF(Date_is_Null=1, NULL , TO_CHAR(v_Year_NextQuarter_Date, 'Q'))
Year_NextQuarter	integer(10)	IIF(Date_is_Null=1, NULL , TO_INTEGER(TO_CHAR(v_Year_NextQuarter_Date, 'YYYY') || v_Year_NextQuarter_Quarter))
v_Year_NextSemester_Date	date/time(29)	IIF(Date_is_Null = 1, NULL , ADD_TO_DATE(i_DATE, 'MM', 6))
v_Year_NextSemester_Quarter	string(1)	IIF(Date_is_Null=1, NULL , TO_CHAR(v_Year_NextSemester_Date, 'Q'))
v_Year_NextSemester_Semester	string(1)	IIF(Date_is_Null=1, NULL , IIF(v_Year_NextSemester_Quarter='1' OR v_Year_NextSemester_Quarter='2', '1' , '2'))
Year_NextSemester	integer(10)	IIF(Date_is_Null=1, NULL , TO_INTEGER(TO_CHAR(v_Year_NextSemester_Date, 'YYYY') || v_Year_NextSemester_Semester))
v_Next_Year_Date	date/time(29)	IIF(Date_is_Null = 1, NULL , ADD_TO_DATE(i_DATE, 'YY',1))
v_NextYear	string(4)	IIF(Date_is_Null=1, NULL , TO_CHAR(v_Next_Year_Date, 'YYYY'))
NextYear_Week	integer(10)	IIF(Date_is_Null = 1, NULL ,TO_INTEGER(v_NextYear || LPAD(TO_CHAR(v_Next_Year_Date, 'WW'), 2, '0')))
NextYear_Month	integer(10)	IIF(Date_is_Null = 1, NULL , TO_INTEGER(v_NextYear || LPAD(TO_CHAR(v_Next_Year_Date, 'MM'), 2, '0')))
NextYear_Period	integer(10)	IIF(Date_is_Null = 1, NULL , TO_INTEGER(v_NextYear || LPAD(TO_CHAR(v_Next_Year_Date, 'MM'), 3, '0')))
NextYear_Quarter	integer(10)	IIF(Date_is_Null=1, NULL , TO_INTEGER(v_NextYear||TO_CHAR(v_Next_Year_Date, 'Q')))
NextYear_Semester	integer(10)	IIF(Date_is_Null=1, NULL , TO_INTEGER(v_NextYear||TO_CHAR(v_Semester)))
NextYear	integer(10)	IIF(Date_is_Null=1, NULL , TO_INTEGER(v_NextYear))

Mapplets

Ένα Mapplet είναι ένα επαναχρησιμοποιήσιμο αντικείμενο (reusable object) που αναπαριστά ένα σύνολο μετασχηματισμών (transformations). Μας επιτρέπει να χρησιμοποιήσουμε ξανά τη λογική που συνδέονται τα transformation και μπορεί να περιέχει όσα transformations χρειαστεί.

Κάποιες φορές ήταν απαραίτητο να χρησιμοποιηθεί ένα ολόκληρο κομμάτι του mapping πάνω από μία φορά, στο ίδιο το mapping ή σε άλλο. Για αυτές τις περιπτώσεις φτιαχτήκαν τρία Mapplets. Η κύρια διαφορά maplet με reusable transformation είναι ότι το maplet είναι ένα σύνολο από transformations ενώ το reusable transformation είναι μόνο του.

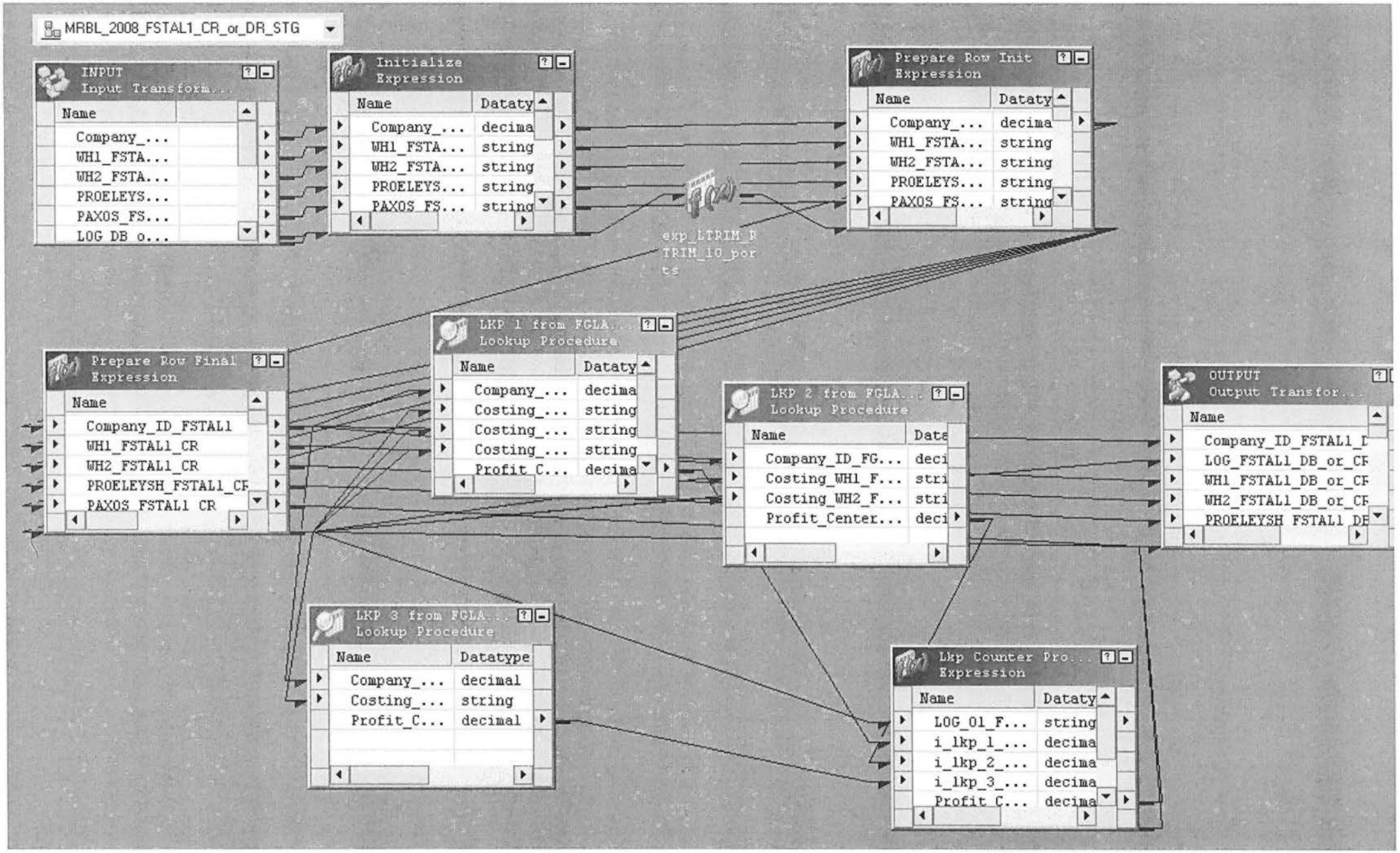
Για τη συγκειμένη εργασία δημιουργηθήκαν τρία Mapplets, που είναι τα πιο κάτω:



Στη συνέχεια θα αναλύσω ένα από αυτά το FSTAL1_CR_or_DR.

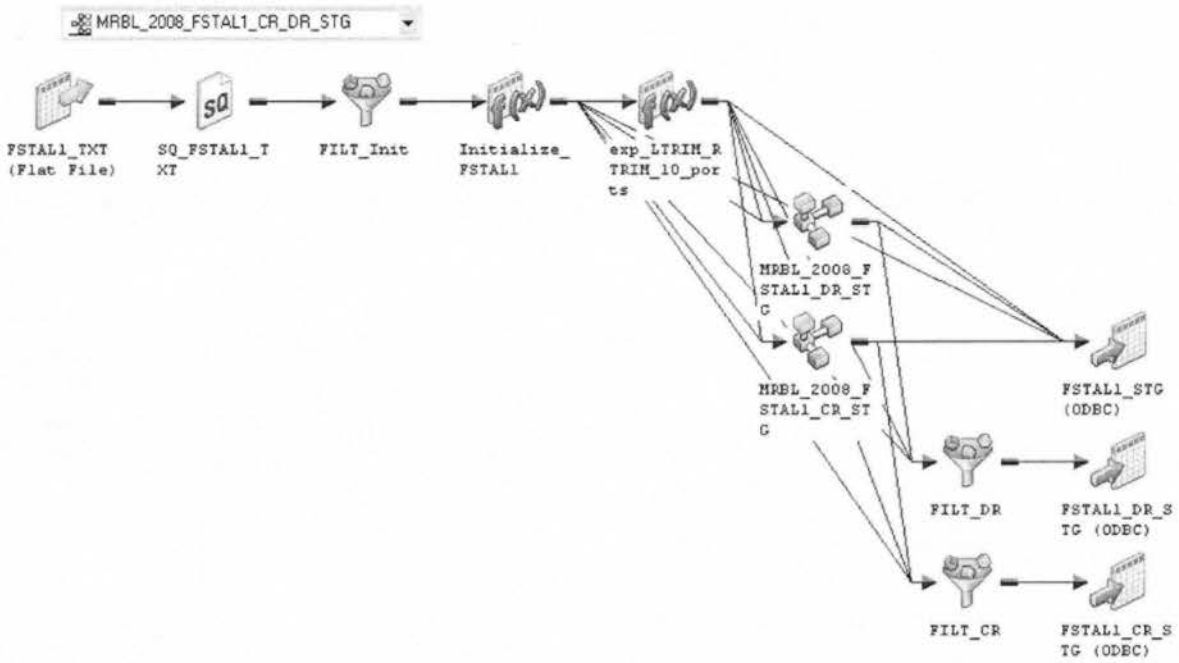
Το mapping FSTAL1_CR_DRασχολείται με την Οριστικοποίηση Κινήσεων Αποθήκης. Έχει ως source πίνακα των FSTAL1_TXT και δύο target το FSTAL1_CR_STG και το FSTAL1_DR_STG. Οι δύο target πίνακες έχουν τις ίδιες κολόνες και τα δεδομένα περνάνε από την ίδια επεξεργασία για να γραφτούν σε αυτούς. Με την μόνη διαφορά ότι ο πρώτος πίνακας διαβάζει της γραμμές που αφορούν τις Πιστώσεις και ο δεύτερος τις Χρεώσεις, εξ' ου και το όνομα CR (credit) και DR (debit).

Πιο κάτω βλέπουμε το screenshot με την υλοποίηση.


MRBL_2008_FSTAL1_CR_or_DR_STG
INPUT
Input Transform...
Name
Company_...
WH1_FSTA...
WH2_FSTA...
PROELEYS...
PAXOS_FS...
LOG_DB_o...
Initialize
Expression
Name
Datatype
Company_... decima
WH1_FSTA... string
WH2_FSTA... string
PROELEYS... string
PAXOS_FS... string
exp_LTRIM_R
TRIM_IO_por
ts
Prepare Row Init
Expression
Name
Datatype
Company_... decima
WH1_FSTA... string
WH2_FSTA... string
PROELEYS... string
PAXOS_FS... string
Prepare Row Final
Expression
Name
Company_ID_FSTAL1
WH1_FSTAL1_CR
WH2_FSTAL1_CR
PROELEYSH_FSTAL1_CR
PAXOS_FSTAL1_CR
LKP 1 from FGLA...
Lookup Procedure
Name
Datatype
Company_... decima
Costing_... string
Costing_... string
Costing_... string
Profit_C... decima
LKP 2 from FGLA...
Lookup Procedure
Name
Company_ID_FG... deci
Costing_WH1_F... stri
Costing_WH2_F... stri
Profit_Center... deci
OUTPUT
Output Transfor...
Name
Company_ID_FSTAL1_D
LOG_FSTAL1_DB_or_CR
WH1_FSTAL1_DB_or_CR
WH2_FSTAL1_DB_or_CR
PROELEYSH_FSTAL1_DB
LKP 3 from FGLA...
Lookup Procedure
Name
Datatype
Company_... decimal
Costing_... string
Profit_C... decimal
Lkp Counter Pro...
Expression
Name
Datatype
LOG_01_F... string
i_lkp_1_... decima
i_lkp_2_... decima
i_lkp_3_... decima
Profit_C... decima

Με λίγα λόγια το συγκεκριμένο mapplet χρησιμοποιείται δύο φορές στο mapping FSTAL1_CR_DR και την μία φορά παίρνει ως input τις γραμμές από Πιστώσεις (cr) και την άλλη από Χρεώσεις (dr).

Επιπρόσθετα η χρήση mapplet απλοποιεί κατά πολύ το διάβασμα ενός mapping, θα μπορούσε να ειπωθεί ότι το mapping γίνεται πιο ευανάγνωστο. Αλλιώς θα παρατηρούσαμε ένα υπερφορτωμένο από transformations mapping.

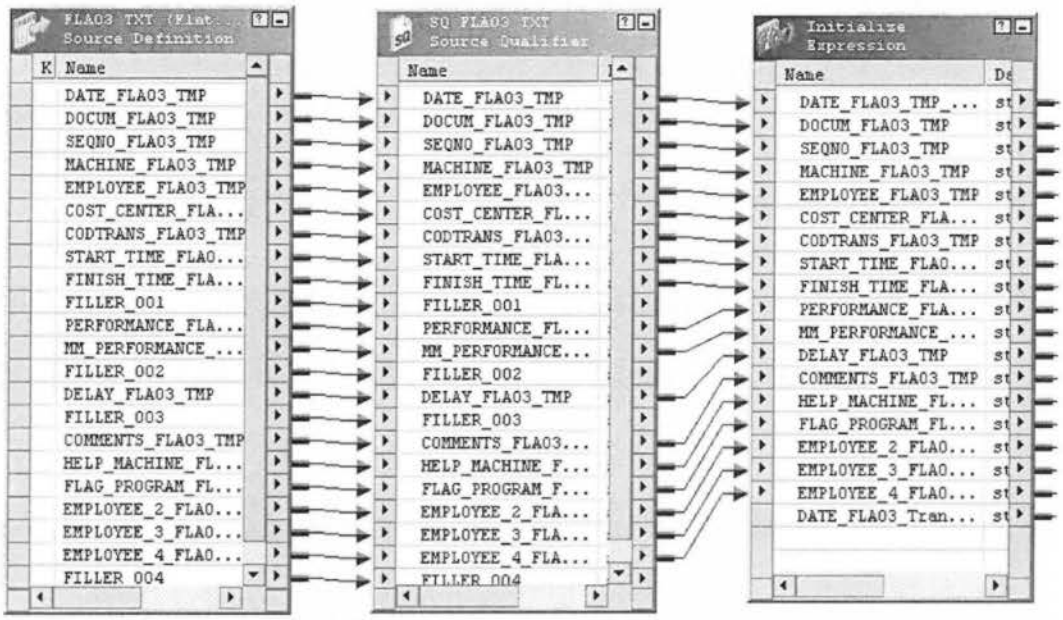


Mappings

Πιο πάνω έγινε αναφορά στα δομικά στοιχεία των mappings που υλοποιηθήκαν. Στη συνέχεια θα παρουσιάσω μερικά από αυτά για να γίνει καλύτερα κατανοητή η εργασία μου.

Στο workflow όλα τα mapping τρέχουν με την σειρά που παρουσιάζονται εδώ:

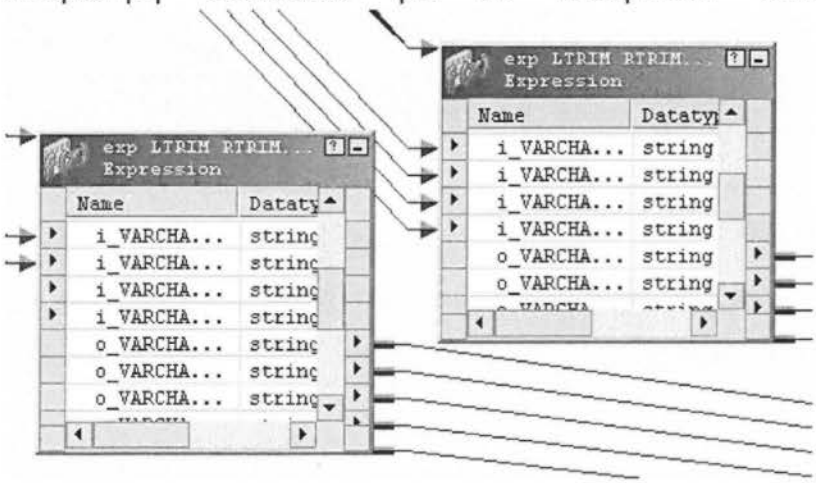
Παράδειγμα mapping FAL03_STG

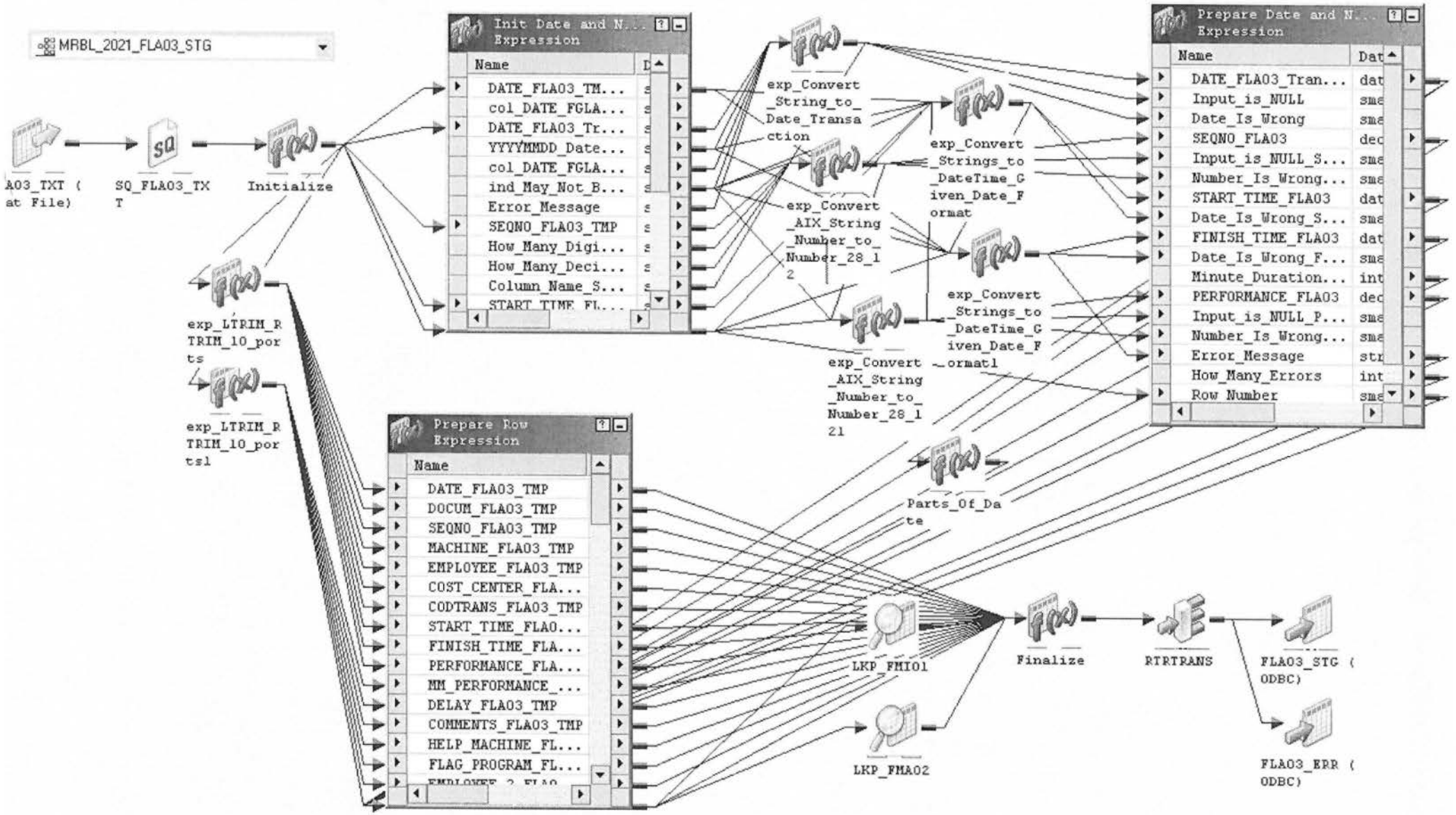


Τα τρία πρώτα transformations όλων των mapping είναι τα : Source Definition, Source Qualifier, Initialize.

Το SourceDefinition(SD) φέρνει τα δεδομένα από τα ansi αρχεία που είναι αποθηκευμένα στο παραγωγικό σύστημα. Σε αυτό το σημείο δεν μπορούμε να κάνουμε καμία αλλαγή στα δεδομένα, η χρήση του SD περιορίζεται στο να περιγράψει τα πηγαία αρχεία που προσφέρουν δεδομένα στην αποθήκη (DW). Για το λόγο αυτό πάντα μετά το SD έπεται ένα SourceQualifier (SQ) και είναι το πρώτο transformation που μας επιτρέπει την επεξεργασία των δεδομένων. Κατά την δικιά μου προτίμηση μεταφέρω όσες κολόνες θέλω να επεξεργαστώ στο transformation Initialize (στο συγκριμένο όλα τα Filler θεωρήθηκαν περιττά) και από κει και πέρα αρχίζει η ουσιαστική τροποποίηση.

Στη συνέχεια όλα τα πεδία περνάνε από exp_Ltrim_Rtrim_10_ports, απαραίτητη διαδικασία για να καθαρίσουν από περιττά κενά.





Το mapping FLA03_STG μεκάποια transformations expanded

Στο transformation Init_Date_and_Number περνάνε μόνο όσα πεδία είναι απαραίτητο να μετατραπούν από αλφαριθμητικό σε αριθμό ή ημερομηνία. Επίσης κάτω από κάθε κολόνα που είναι επεξεργασία ημερομηνία; προστίθεται ένα Column_Name και YYYYMMDD_Date_Format. Αυτές είναι οι επιπρόσθετες πληροφορίες που θέλει το reusable transformation να χρησιμοποιήσει ως εισόδους και να κάνει την μετατροπή ημερομηνίας. Αντίστοιχα το reusable για μετατροπή αριθμών χρειάζεται επιπρόσθετα να ορίσουμε How_Many_Digits, How_Many_Decimals, Column_Name.

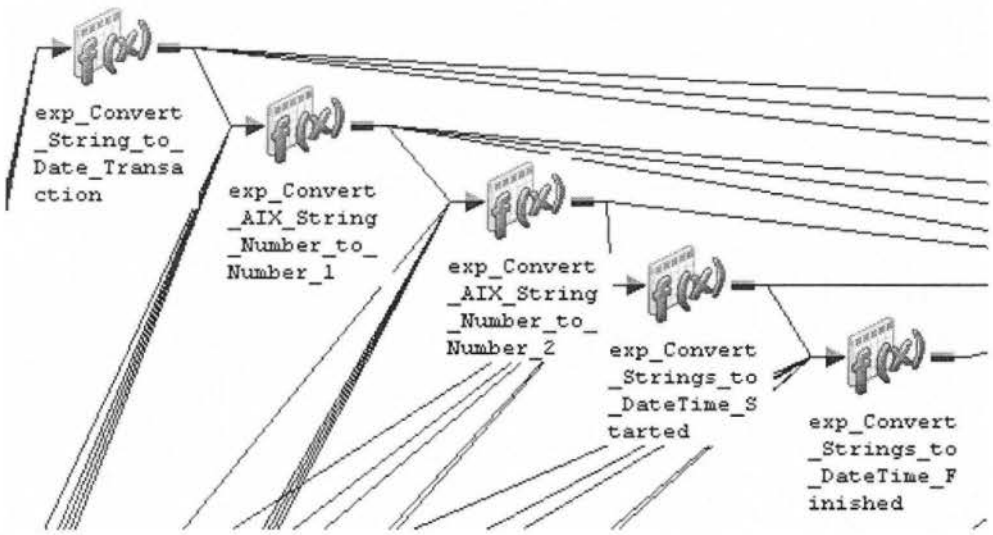
Το Error_Message αρχικοποιείται με κενό ('') και περνάει αλυσιδωτά από το ένα transformation στο άλλο. Με αυτό τον τρόπο μαζεύει όλα πιθανά λάθη που μπορεί να προκύψουν κατά την επεξεργασία.

Το Row_Number μετράει τις γραμμές του πίνακα. Αυτό χρησιμεύει κατά το error handling, μαζί με το μήνυμα λάθους ξέρουμε και σε πια γραμμή έγινε το σφάλμα.

	Port Name	Datatype	Prec	Scale	I	O	V	Expression
1	DATE_FLA03_TMP_Original	string	8	0	☑	☑	☐	DATE_FLA03_TMP_Original
2	col_DATE_FGLAL04_Original	string	120	0	☐	☑	☐	'DATE_FLA03_TMP_Original'
3	DATE_FLA03_Transaction	string	8	0	☑	☑	☐	DATE_FLA03_Transaction
4	YYYYMMDD_Date_Format	string	10	0	☐	☑	☐	'YYYYMMDD'
5	col_DATE_FGLAL04_Transaction	string	120	0	☐	☑	☐	'DATE_FGLAL04_Transaction'
6	ind_May_Not_Be_Nul	small integer	5	0	☐	☑	☐	1
7	Error_Message	string	1024	0	☐	☑	☐	''
8	SEQNO_FLA03_TMP	string	3	0	☑	☑	☐	SEQNO_FLA03_TMP
9	How_Many_Digits_SEQNO	small integer	5	0	☐	☑	☐	3
10	How_Many_Decimals_SEQNO	small integer	5	0	☐	☑	☐	0
11	Column_Name_SEQNO	string	10	0	☐	☑	☐	'SEQNO_FLA03_TMP'
12	START_TIME_FLA03_TMP	string	4	0	☑	☑	☐	START_TIME_FLA03_TMP
13	col_START_TIME	string	120	0	☐	☑	☐	'START_TIME_FLA03_TMP'
14	FINISH_TIME_FLA03_TMP	string	4	0	☑	☑	☐	FINISH_TIME_FLA03_TMP
15	col_Finish_TIME	string	120	0	☐	☑	☐	'FINISH_TIME_FLA03_TMP'
16	PERFORMANCE_FLA03_TMP	string	10	0	☑	☑	☐	PERFORMANCE_FLA03_TMP
17	How_Many_Digits_PERFORMANCE	small integer	5	0	☐	☑	☐	10
18	How_Many_Decimals_PERFORMANCE	small integer	5	0	☐	☑	☐	3
19	Column_Name_PERFORMANCE	string	10	0	☐	☑	☐	'PERFORMANCE_FLA03_TMP'
20	v_Row_Number	small integer	5	0	☐	☐	☑	v_Row_Number +1
21	Row_Number	small integer	5	0	☐	☑	☐	v_Row_Number

Transformation Init_Date_and_Number

Στη συνέχεια πραγματοποιούνται οι μετατροπές σε αριθμούς και ημερομηνίες:



Με τα αποτελέσματα να περνάνε στο επόμενο transformation Prepare_Date_and_Number. Σε αυτό το σημείο θα ήθελα να επισημάνω το How_Many_Errors που κάνει πρόσθεση των μονάδων όπου έχει γίνει λάθος. Γίνεται η πράξη : How_Many_Errors =Date_Is_Wrong + Number_Is_Wrong_SEQNO + Number_Is_Wrong_PERFORMANCE + Date_Is_Wrong_Start_Time + Date_Is_Wrong_Finished_Time

Το κάθε date_is_wrong και number_is_wrong περνώντας γραμμή - γραμμή από τα προηγούμενα transformation, σε περίπτωση λάθους καταγράφουν μονάδα. Οπότε τοHow_Many_Errors προσθέτει αυτές τις μονάδες και μας δείχνει πόσα σφάλματα έχουν γίνει συνολικά.

	Port Name	Datatype	Prec	Scale	I	O	V	Expression
1	DATE_FLA03_Transaction	date/time	29	9	☑	☑	☐	DATE_FLA03_Transaction
2	Input_is_NULL	small integer	5	0	☑	☐	☐	
3	Date_Is_Wrong	small integer	5	0	☑	☐	☐	
4	SEQNO_FLA03	decimal	3	0	☑	☑	☐	SEQNO_FLA03
5	Input_is_NULL_SEQNO	small integer	5	0	☑	☐	☐	
6	Number_Is_Wrong_SEQNO	small integer	5	0	☑	☐	☐	
7	START_TIME_FLA03	date/time	29	9	☑	☑	☐	START_TIME_FLA03
8	Date_Is_Wrong_Start_Time	small integer	5	0	☑	☐	☐	
9	FINISH_TIME_FLA03	date/time	29	9	☑	☑	☐	FINISH_TIME_FLA03
10	Date_Is_Wrong_Finished_Time	small integer	5	0	☑	☐	☐	
11	Minute_Duration_FLA03	integer	10	0	☐	☑	☐	ROUND(DATE_DIFF(FINISH_TIM
12	PERFORMANCE_FLA03	decimal	10	3	☑	☑	☐	PERFORMANCE_FLA03
13	Input_is_NULL_PERFORMANCE	small integer	5	0	☑	☐	☐	
14	Number_Is_Wrong_PERFORMANCE	small integer	5	0	☑	☐	☐	
15	Error_Message	string	1024	0	☑	☑	☐	Error_Message
16	How_Many_Errors	integer	10	0	☐	☑	☐	Date_Is_Wrong + ...
17	Row_Number	small integer	5	0	☑	☑	☐	Row_Number

Transformation Prepare_Date_and_Number

Στο Prepare_Rowμαζεύονται όλες τα πεδία και αυτά που έχουν υποστεί μετατροπή και αυτά που έρχονται μετά το Ltrim-Rtrim transformation.

Κάποια πεδία έχουν περίπλοκους υπολογισμούς και κάποια απλός ελέγχονται

Select transformation: f(x) Prepare_Row

Transformation type: Expression

	Port Name	Datatype	Prec	Scale	I	O	V	Expression
13	COMMENTS_FLA03_TMP	string	20	0	☑	☑	☐	COMMENTS_FLA03_TMP
14	HELP_MACHINE_FLA03_TMP	string	8	0	☑	☑	☐	HELP_MACHINE_FLA03_TMP
15	FLAG_PROGRAM_FLA03_TMP	string	1	0	☑	☑	☐	FLAG_PROGRAM_FLA03_TMP
16	EMPLOYEE_2_FLA03_TMP	string	5	0	☑	☑	☐	EMPLOYEE_2_FLA03_TMP
17	EMPLOYEE_3_FLA03_TMP	string	5	0	☑	☑	☐	EMPLOYEE_3_FLA03_TMP
18	EMPLOYEE_4_FLA03_TMP	string	5	0	☑	☑	☐	EMPLOYEE_4_FLA03_TMP
19	Company_ID_FLA03	decimal	2	0	☐	☑	☐	$$Company_ID
20	Fiscal_Year_ID_FLA03	decimal	4	0	☐	☑	☐	$$Fiscal_Year_ID
21	Period_01_Balance_Factor_FLA03	decimal	1	0	☐	☑	☐	IIF(Period_FLA03 = 01, 1 ,0)
22	Period_02_Balance_Factor_FLA03	decimal	1	0	☐	☑	☐	IIF(Period_FLA03 = 02,1,0)
23	Period_03_Balance_Factor_FLA03	decimal	1	0	☐	☑	☐	IIF(Period_FLA03 = 03,1,0)
24	Period_04_Balance_Factor_FLA03	decimal	1	0	☐	☑	☐	IIF(Period_FLA03 = 04,1,0)
25	Period_05_Balance_Factor_FLA03	decimal	1	0	☐	☑	☐	IIF(Period_FLA03 = 05,1,0)
26	Period_06_Balance_Factor_FLA03	decimal	1	0	☐	☑	☐	IIF(Period_FLA03 = 06,1,0)
27	Period_07_Balance_Factor_FLA03	decimal	1	0	☐	☑	☐	IIF(Period_FLA03 = 07,1,0)
28	Period_08_Balance_Factor_FLA03	decimal	1	0	☐	☑	☐	IIF(Period_FLA03 = 08,1,0)
29	Period_09_Balance_Factor_FLA03	decimal	1	0	☐	☑	☐	IIF(Period_FLA03 = 09,1,0)

να μην έχουν κενό, δηλαδή IIF(MACHINE_FLA03_TMP<>'',MACHINE_FLA03_TMP,NULL). Επίσης εδώ γίνεται ανάθεση των παραμέτρων $$Company_IDκαι $$Fiscal_Year_ID, το σύμβολο $$ δείχνει ότι ακολουθεί παράμετρος ή μεταβλητή. Δήλωση παραμέτρων:

Declare Parameters and Variables

Declare the parameters and variables you want to use in the mapping or mapplet.
A parameter represents a constant value defined before mapping run.
A variable represents a value that can be changed during mapping run.

Name	Type	Datatype	Prec	Scale	Aggregation	IsExprVar
$$Company_ID	Param...	integer	10	0		FALSE
$$Fiscal_Year_ID	Param...	integer	10	0		FALSE

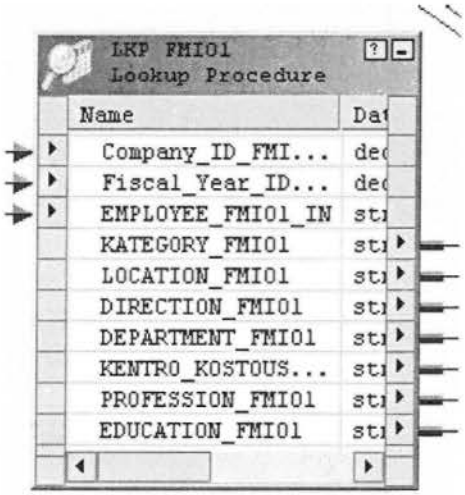
Initial value:

Description:

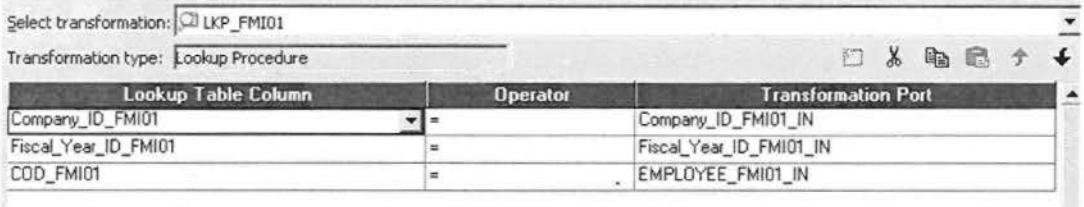
OK Cancel Help

Ο τελικός πίνακας (target) περιέχει πεδία που έρχονται από άλλους πίνακες. Σε αυτή τη περίπτωση με το transformation Lookupφέρνουμε δεδομένα από δύο πίνακες το FMI01_STG και το FMA02. Είναι σημαντικό οι lookup πίνακες να προηγούνται στην ροή από το τρέχον mapping, ώστε τα δεδομένα που κοιτάει να είναι έγκυρα και ενημερωμένα.

Στα επόμενα screenshot βλέπουμε δύο lookup πίνακες που χρειάστηκαν για το συγκεκριμένο mapping. Οι τρείς είσοδοι που παίρνουν τα lookup συμμετέχουν στην συνθήκη με την οποία γίνεται η σύνδεση των πινάκων. Τα πεδία που θέλουμε να πάρουμε από το lookup τα σημειώνουμε ως εξόδους.

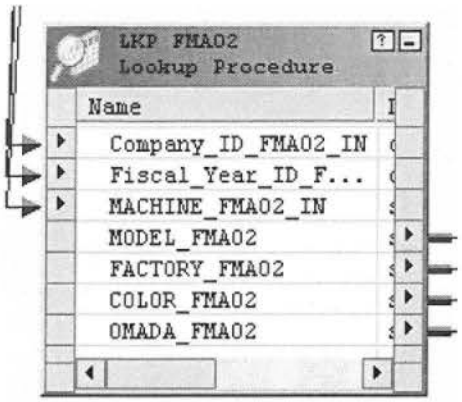


Η συνθήκη (...where πιο κάτω φαίνεται στον sql κώδικα) με την οποία πραγματοποιήται το lookup, όπου κατάληξη είναι _IN συμαίνει ότι είναι πεδία του τρέχοντος mapping (FLA03_STG).

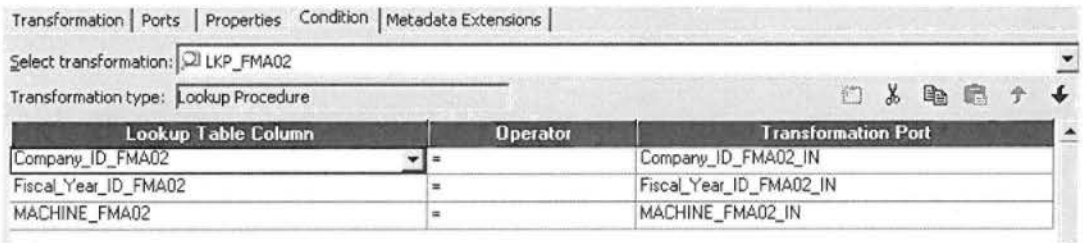


Πιο κάτω βλέπουμε με ποιο τρόπο γίνεται το lookup των πεδίων στον sql κώδικα. Σε αντίθεση με το ETL, το lookup, στον κώδικα γίνεται με update.

```
UPDATE FLA03_STG SET
   FLA03_STG.KATEGORY_FMI01          =   FMI01_STG.KATEGORY_FMI01,
   FLA03_STG.LOCATION_FMI01          =   FMI01_STG.LOCATION_FMI01,
   FLA03_STG.DIRECTION_FMI01         =   FMI01_STG.DIRECTION_FMI01,
   FLA03_STG.DEPARTMENT_FMI01        =   FMI01_STG.DEPARTMENT_FMI01,
   FLA03_STG.KENTRO_KOSTOUS_FMI01    =   FMI01_STG.KENTRO_KOSTOUS_FMI01,
   FLA03_STG.PROFESSION_FMI01        =   FMI01_STG.PROFESSION_FMI01,
   FLA03_STG.EDUCATION_FMI01         =   FMI01_STG.EDUCATION_FMI01
FROM FLA03_STG, FMI01_STG
WHERE FLA03_STG.Company_ID_FLA03     = FMI01_STG.Company_ID_FMI01
AND   FLA03_STG.Fiscal_Year_ID_FLA03 = FMI01_STG.Fiscal_Year_ID_FMI01
AND   FLA03_STG.EMPLOYEE_FLA03       = FMI01_STG.COD_FMI01;
```



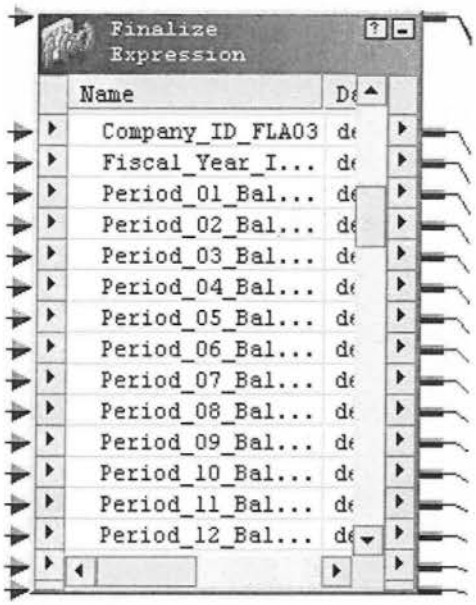
Το δεύτερο lookup γίνεται με τον πίνακα FMA02_STG.



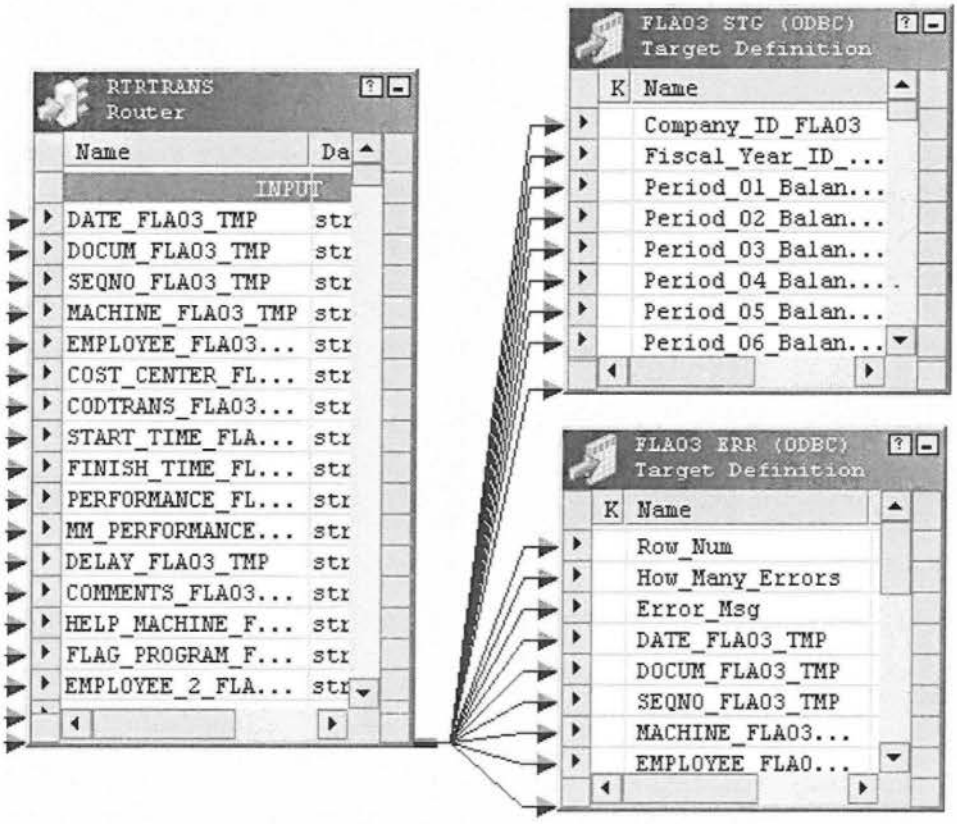
Και ο αντίστοιχα sql κώδικας.

```
UPDATE FLA03_STG SET
   FLA03_STG.MODEL_FMA02              =   FMA02_STG.MODEL_FMA02,
   FLA03_STG.FACTORY_FMA02            =   FMA02_STG.FACTORY_FMA02,
   FLA03_STG.COLOR_FMA02              =   FMA02_STG.COLOR_FMA02,
   FLA03_STG.OMADA_FMA02              =   FMA02_STG.OMADA_FMA02
FROM FLA03_STG, FMA02_STG
WHERE FLA03_STG.Company_ID_FLA03      = FMA02_STG.Company_ID_FMA02
AND   FLA03_STG.Fiscal_Year_ID_FLA03 = FMA02_STG.Fiscal_Year_ID_FMA02
AND   FLA03_STG.MACHINE_FLA03         = FMA02_STG.MACHINE_FMA02;
```

Στο Finalize τα δεδομένα έχουν πλέον υποστεί όλες τις απαραίτητες μετατροπές και είναι ένα βήμα πριν από τους τελικούς (target) πίνακες.



Στο τελικό βήμα υπάρχουν δύο target πίνακες. Ένας κανονικός ο FLA03_STG και ένας FLA03_ERR όπου γράφονται οι λάθος γραμμές αν προέκυψαν. Ουσιαστικά ο πίνακας FLA03_ERR χρησιμεύει στο error control.

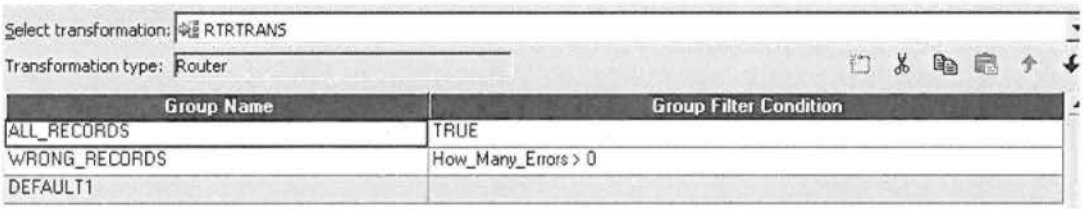


Ο FLA03_ERR έχει όλα τα πεδία _TMP που έρχονται αυτούσια από τον source πίνακα, δηλαδή όλα τα πεδία έχουν τύπο string Τα πεδία αυτά δεν θέλουμε να είναι επεξεργασμένα για να βλέπουμε τα λάθος δεδομένα όπως έρχονται από το παραγωγικό σύστημα. Π.χ. από το σύστημα έρχεται μία ημερομηνία 11110104, αμέσως - αμέσως καταλαβαίνουμε ότι έχει γίνει κάποιο λάθος στην χρονολογία, το λάθος αυτό θα γραφτεί στον πίνακα _err όπου θα αναφέρεται σε πια γραμμή βρίσκεται, πόσα λάθη περιλαμβάνει αυτή η γραμμή και ποια είναι. Αν εκτός ημερομηνίας δεν είχαμε άλλο λάθος θα βλέπαμε στο Error_Msg "Wrong: Date_FLA03_TMP (Format:YYYYMMDD)".

Προφανώς τα _TMP πεδία που πηγαίνουν στον error table δεν πρέπει να περάνανε στον target FLA03_STG όπου γράφονται όσα δεδομένα έχουν υποστεί σωστή επεξεργασία. Τον διαχωρισμό των πεδίων καταφέρνουμε με το Router transformation, που έχει την δυνατότητα να φτιάχνει ομάδες ανάλογα με την συνθήκη που του επιβάλουμε.

Στην συγκεκριμένη περίπτωση έχουν δημιουργηθεί δύο ομάδες :

- All_Records: όσες γραμμές πάνε στο FLA03_STG
- Wrong_Records: φιλτράρονται όσες γραμμές έχουν έστω και ένα λάθος, How_Many_Errors>0.

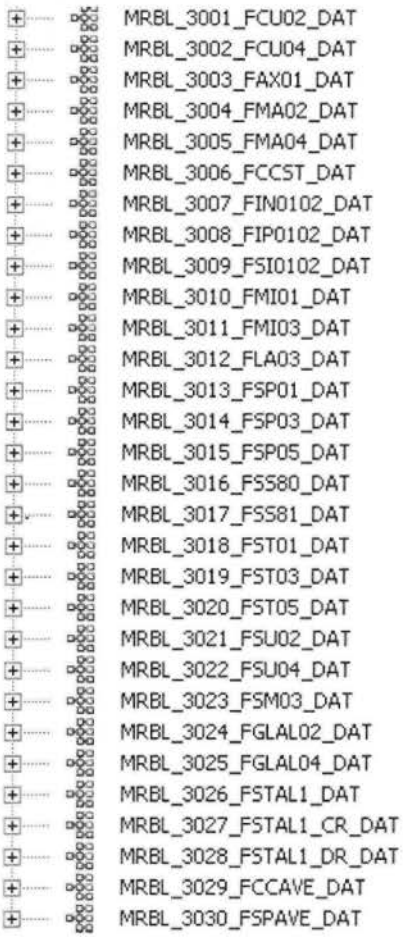


MRBL_03_Load_DAT_files

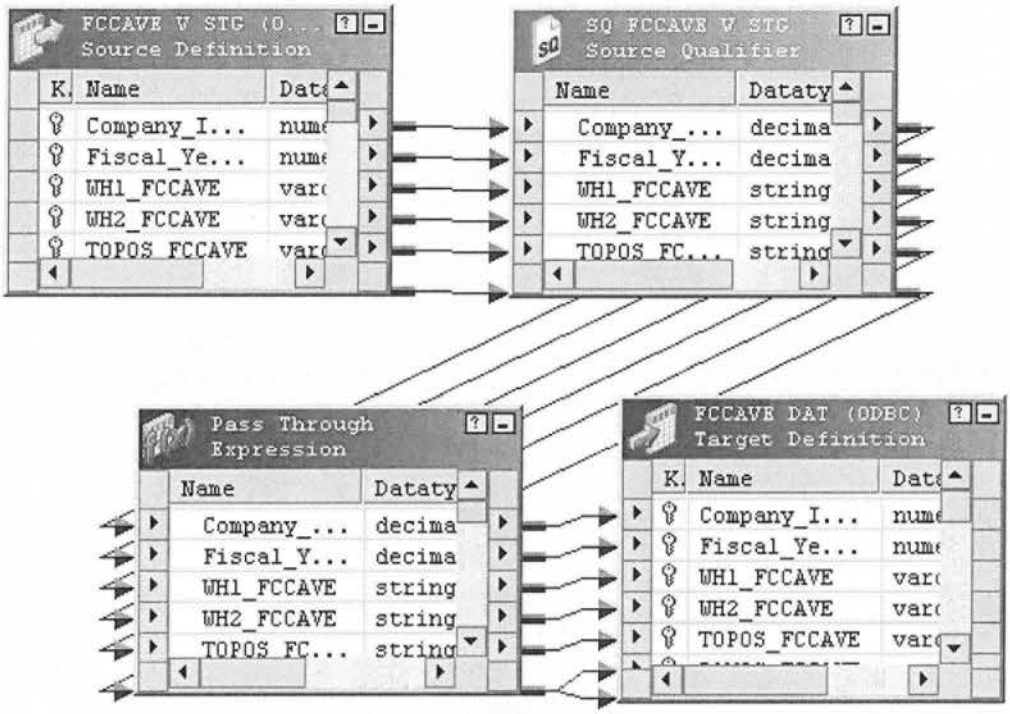
Υλοποίηση DAT Mappings

Σε αυτό το σημείο περνάμε στην δημιουργία των τελικών πινάκων που θα κρατάνε την ιστορικότητα. Φτάσαμε στην τελευταία φάση της ETL διαδικασίας το Loading. Τα δεδομένα έχουν περάσει από extraction και transformation και είναι έτοιμα να φορτωθούν στους dat πίνακες.

Συνολικά έχουμε 30 dat πίνακες. Είναι κατά οχτώ λιγότεροι από τους stg. Αυτό συμβαίνει γιατί κάποιαmappings συγχωνευτήκαν σε ένα (FGLAL02,FGLAL04). Άλλοι πίνακες ήταν βοηθητικοί και υπήρχαν μόνο στο stg σταδιο (π.χ. FIN01_STG, FIN02_STG ήταν βοηθητικοί για να προκύψει το FIN0102_DAT). Υπάρχει μια περίπτωση όπου ένα mapping stg (FSTAL1_CR_DR_STG) σε dat επίπεδο έχει χωριστεί σε τρία mapping (FSTAL1_DAT, FSTAL1_CR_DAT, FSTAL1_DR_DAT).

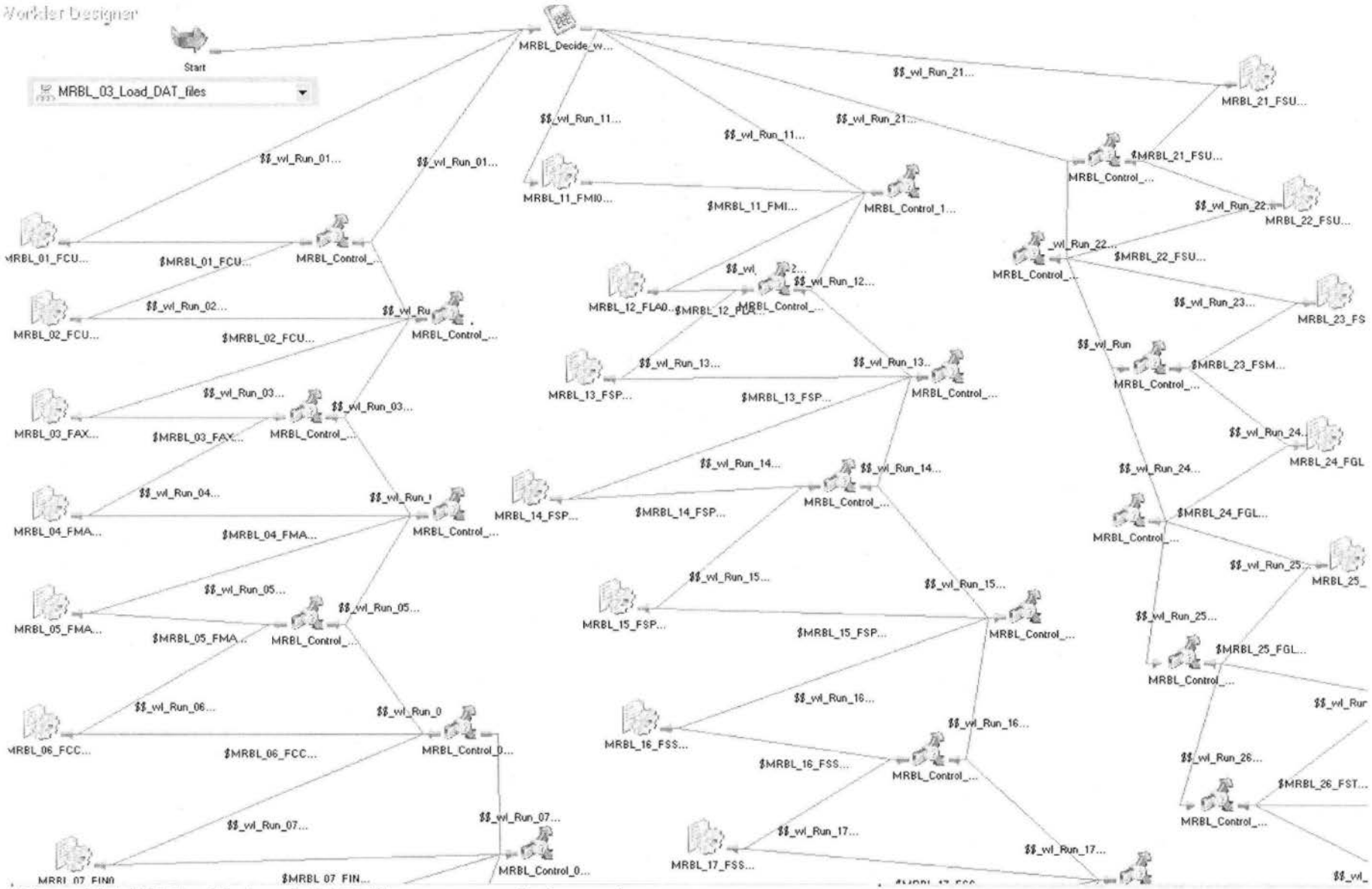


Η υλοποίηση των dat πινάκων γίνεται ως εξής: ως source πίνακες έχουμε τους target των stg πινάκων και τα δεδομένα περνάνε σχεδόν κατευθείαν στους target dat πίνακες χωρίς καμία επεξεργασία. Βασικά στους dat δεν θέλουμε να γίνεται καμία επεξεργασία γιατί ότι χρειαζόταν να γίνει έγινε στους stage tables.



Λίγο πιο πάνω βλέπουμε ένα χαρακτηριστικό dat mapping. Τα δεδομένα του FCCAVE_V_STG γράφονται κατευθείαν στο FCCAVE_DAT, περνάνε μεν από το transformation Pass_Throughαλλά όπως εισέρχονται έτσι απαράλλακτα και εξέρχονται.

Όλα τα dat mappings γίνονται sessions στο worklet
MRBL_03_Load_DAT_files που σε πλήρη μορφή φαίνεται πιο κάτω:



Toworklet MRBL_03_Load_DAT_files σε expanded μορφή.

Στο worklet MRBL_03_Load_DAT_files τα sessions τοποθετούνται σε τρία threads που τρέχουν παράλληλα. Στο συγκεκριμένο σημείο δεν είναι απαραίτητο οι πίνακες να εκτελεστούν σειριακά όπως στα stg. Στους stage υπάρχουν dependencies, ο επόμενος πίνακας εξαρτάται από τον προηγούμενο, ενώ οι dat εξαρτώνται μόνο από τους stage που έχει εκτελεστεί λίγο πιο πριν.

Όπως και σε προηγούμενα worklet ακολουθήθηκε η τριγωνική απεικόνιση για καλύτερο έλεγχο των ροών. Στο MRBL_Decide_Which_Sessions_to_Run έχουμε την επιλογή να επιλέξουμε ποια session θέλουμε να τρέξουν.

Select task: MRBL_Decide_which_Sessions_to_Run

Task type: Assignment

Use the controls on the right to add/delete/copy-paste or change the order of expressions evaluated in this assignment task.

	User Defined Variables	Operator	Expression
1	$$_wl_Run_01_FCU02_DAT_Ind	=	1
2	$$_wl_Run_02_FCU04_DAT_Ind	=	1
3	$$_wl_Run_03_FAX01_DAT_Ind	=	1
4	$$_wl_Run_04_FMA02_DAT_Ind	=	1
5	$$_wl_Run_05_FMA04_DAT_Ind	=	1
6	$$_wl_Run_06_FCCST_DAT_Ind	=	1
7	$$_wl_Run_07_FIN0102_DAT_Ind	=	1
8	$$_wl_Run_08_FIP0102_DAT_Ind	=	1
9	$$_wl_Run_09_FSI0102_DAT_Ind	=	1
10	$$_wl_Run_10_FMI01_DAT_Ind	=	1
11	$$_wl_Run_11_FMI03_DAT_Ind	=	1
12	$$_wl_Run_12_FLA03_DAT_Ind	=	1
13	$$_wl_Run_13_FSP01_DAT_Ind	=	1
14	$$_wl_Run_14_FSP03_DAT_Ind	=	1
15	$$_wl_Run_15_FSP05_DAT_Ind	=	1
16	$$_wl_Run_16_FSS80_DAT_Ind	=	1
17	$$_wl_Run_17_FSS81_DAT_Ind	=	1
18	$$_wl_Run_18_FST01_DAT_Ind	=	1

Βάζοντας μια από τις μεταβλητές διάφορο του ενός αποκλείουμε το συγκεκριμένο session από την εκτέλεση.

Όταν εκτελεστούν όλα τα νήματα με επιτυχία σηματοδοτείται σε όλο το workflow ότι η διαδικασία ολοκληρώθηκε με επιτυχία. Με αυτό τον τρόπο ολοκληρώνεται η εκτέλεση της ETLδιαδικασίας.

Ολοκλήρωση ETL διαδικασίας

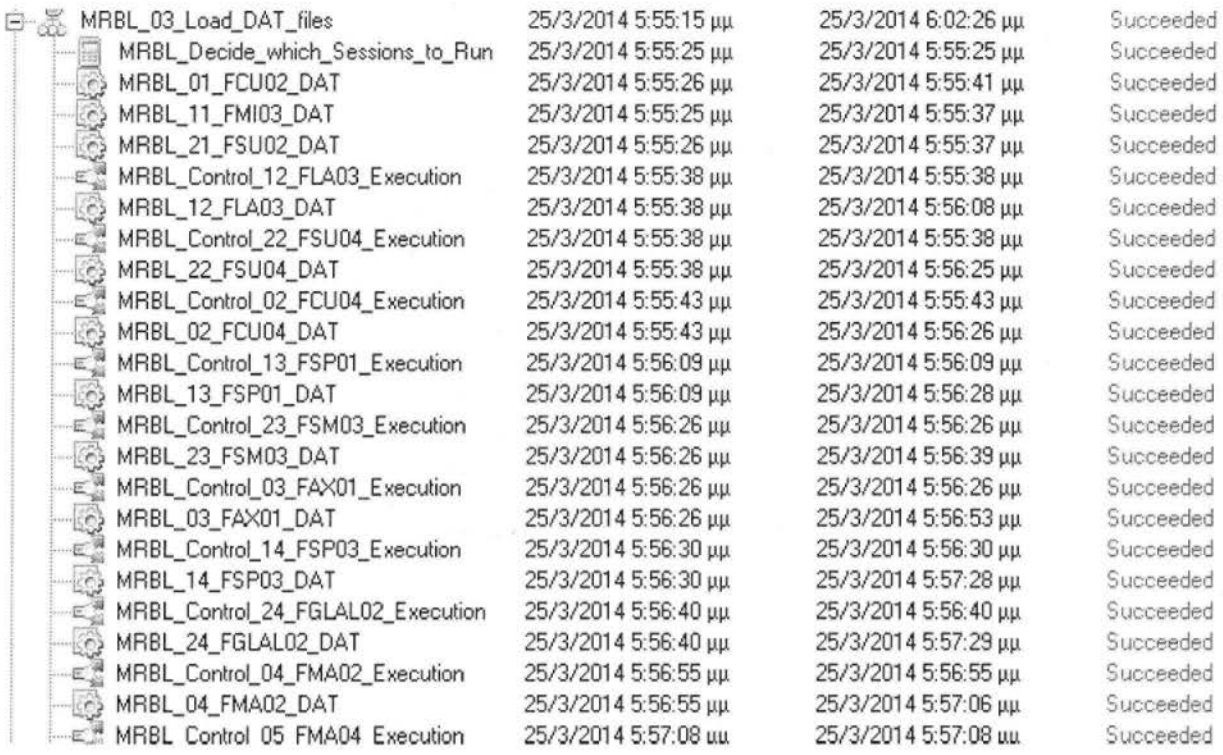
Η παρακολούθηση της εκτέλεσης της ETL διαδικασίας γίνεται από το Workflow Monitor. Από εκεί μπορούμε να δούμε όλα τα κομβικά σημεία των ροών, σε τι στάδιο εκτέλεσης βρίσκονται και αν ολοκληρώθηκαν με επιτυχία ή όχι. Παρακάτω απεικονίζεται η αναλυτικά η λίστα με όλα τα κομβικά σημεία που περνάει η ροή του workflow. Η λίστα βρίσκεται σε συμπυκνωμένη (collapsed) μορφή, βλέπουμε ότι τρία worlet έχουν (+) από δίπλα.

Workflow Run	Start Time	Completion Time	Status
MRBL_0000000_ETL_for_companies_1_to_3			
MRBL_0000000_ETL_for_companies_1_to_3	25/3/2014 5:43:38 μμ	25/3/2014 6:02:26 μμ	Succeeded
ETL_1st_Company	25/3/2014 5:43:38 μμ	25/3/2014 6:02:26 μμ	Succeeded
MRBL_30_Truncate_FIRM_STG	25/3/2014 5:43:38 μμ	25/3/2014 5:43:42 μμ	Succeeded
MRBL_11_After_FTP	25/3/2014 5:43:38 μμ	25/3/2014 5:43:38 μμ	Succeeded
MRBL_40_Populating_FIRM_STG	25/3/2014 5:43:42 μμ	25/3/2014 5:43:45 μμ	Succeeded
MRBL_50_Fix_Company_Code_and_Year_...	25/3/2014 5:43:47 μμ	25/3/2014 5:43:47 μμ	Succeeded
MRBL_01_Load_DAT_Parameter_files	25/3/2014 5:43:47 μμ	25/3/2014 5:44:26 μμ	Succeeded
MRBL_02_Control	25/3/2014 5:44:26 μμ	25/3/2014 5:44:26 μμ	Succeeded
MRBL_02_Load_STG_files	25/3/2014 5:44:26 μμ	25/3/2014 5:55:15 μμ	Succeeded
MRBL_03_Control	25/3/2014 5:55:15 μμ	25/3/2014 5:55:15 μμ	Succeeded
MRBL_03_Load_DAT_files	25/3/2014 5:55:15 μμ	25/3/2014 6:02:26 μμ	Succeeded
MRBL_99_Verify_no_Errors_Occured	25/3/2014 6:02:26 μμ	25/3/2014 6:02:26 μμ	Succeeded
MRBL_Verify_Variable_Values	25/3/2014 5:43:38 μμ	25/3/2014 5:43:38 μμ	Succeeded
MRBL_After_Running_1st_Company	25/3/2014 6:02:26 μμ	25/3/2014 6:02:26 μμ	Succeeded
MRBL_After_Running_2nd_Company	25/3/2014 6:02:26 μμ	25/3/2014 6:02:26 μμ	Succeeded
MRBL_Verify_No_Errors_Occured	25/3/2014 6:02:26 μμ	25/3/2014 6:02:26 μμ	Succeeded

Αν ανοίξουμε τα (+) των worlet MRBL_01_Load_DAT_Paramiter_files, MRBL_02_Load_STG_files και MRBL_03_Load_DAT_files θα δούμε sessions και decisions που τα απαρτίζουν.

MRBL_01_Load_DAT_Parameter_files	25/3/2014 5:43:47 μμ	25/3/2014 5:44:26 μμ	Succeeded
MRBL_1000000_a_Decide_which_Ses...	25/3/2014 5:43:47 μμ	25/3/2014 5:43:47 μμ	Succeeded
MRBL_1000030_s_10_ParamFiles_DAT	25/3/2014 5:43:48 μμ	25/3/2014 5:44:24 μμ	Succeeded
MRBL_1000020_s_10_ParamFiles_DAT	25/3/2014 5:43:47 μμ	25/3/2014 5:44:20 μμ	Succeeded
MRBL_1000010_s_10_ParamFiles_DAT	25/3/2014 5:43:48 μμ	25/3/2014 5:44:21 μμ	Succeeded
MRBL_1000040_s_10_ParamFiles_DAT	25/3/2014 5:43:48 μμ	25/3/2014 5:44:22 μμ	Succeeded
MRBL_1000050_s_03_ParamFiles_DAT	25/3/2014 5:43:48 μμ	25/3/2014 5:44:04 μμ	Succeeded
MRBL_1000059_d_After_03_ParamFile...	25/3/2014 5:44:05 μμ	25/3/2014 5:44:05 μμ	Succeeded
MRBL_1000019_d_After_10_ParamFile...	25/3/2014 5:44:22 μμ	25/3/2014 5:44:22 μμ	Succeeded
MRBL_1000029_d_After_10_ParamFile...	25/3/2014 5:44:22 μμ	25/3/2014 5:44:22 μμ	Succeeded
MRBL_1000049_d_After_10_ParamFile...	25/3/2014 5:44:24 μμ	25/3/2014 5:44:24 μμ	Succeeded
MRBL_1000039_d_After_11_ParamFile...	25/3/2014 5:44:26 μμ	25/3/2014 5:44:26 μμ	Succeeded
MRBL_1000999_Verify_no_Errors_Occ...	25/3/2014 5:44:26 μμ	25/3/2014 5:44:26 μμ	Succeeded
MRBL_02_Control	25/3/2014 5:44:26 μμ	25/3/2014 5:44:26 μμ	Succeeded
MRBL_02_Load_STG_files	25/3/2014 5:44:26 μμ	25/3/2014 5:55:15 μμ	Succeeded
MRBL_Decide_Which_Sessions_to_Run	25/3/2014 5:44:27 μμ	25/3/2014 5:44:27 μμ	Succeeded
MRBL_02_Load_STG_files_1st_Part	25/3/2014 5:44:27 μμ	25/3/2014 5:46:25 μμ	Succeeded
MRBL_02_Load_STG_files_2nd_Part	25/3/2014 5:46:25 μμ	25/3/2014 5:48:57 μμ	Succeeded
MRBL_02_Load_STG_files_3rd_Part	25/3/2014 5:48:57 μμ	25/3/2014 5:52:00 μμ	Succeeded
MRBL_02_Load_STG_files_4th_Part	25/3/2014 5:52:00 μμ	25/3/2014 5:55:15 μμ	Succeeded
MRBL_2000999_Verify_no_Errors_Occ...	25/3/2014 5:55:15 μμ	25/3/2014 5:55:15 μμ	Succeeded

Επειδή το MRBL_03_Load_DAT_files ήταν πιο μεγάλο σε expanded μορφή από τα δύο προηγούμενα, του αφιέρωσα ξεχωριστό screenshot.



MRBL_03_Load_DAT_files	25/3/2014 5:55:15 μμ	25/3/2014 6:02:26 μμ	Succeeded
MRBL_Decide_which_Sessions_to_Run	25/3/2014 5:55:25 μμ	25/3/2014 5:55:25 μμ	Succeeded
MRBL_01_FCU02_DAT	25/3/2014 5:55:26 μμ	25/3/2014 5:55:41 μμ	Succeeded
MRBL_11_FMI03_DAT	25/3/2014 5:55:25 μμ	25/3/2014 5:55:37 μμ	Succeeded
MRBL_21_FSU02_DAT	25/3/2014 5:55:26 μμ	25/3/2014 5:55:37 μμ	Succeeded
MRBL_Control_12_FLA03_Execution	25/3/2014 5:55:38 μμ	25/3/2014 5:55:38 μμ	Succeeded
MRBL_12_FLA03_DAT	25/3/2014 5:55:38 μμ	25/3/2014 5:56:08 μμ	Succeeded
MRBL_Control_22_FSU04_Execution	25/3/2014 5:55:38 μμ	25/3/2014 5:55:38 μμ	Succeeded
MRBL_22_FSU04_DAT	25/3/2014 5:55:38 μμ	25/3/2014 5:56:25 μμ	Succeeded
MRBL_Control_02_FCU04_Execution	25/3/2014 5:55:43 μμ	25/3/2014 5:55:43 μμ	Succeeded
MRBL_02_FCU04_DAT	25/3/2014 5:55:43 μμ	25/3/2014 5:56:26 μμ	Succeeded
MRBL_Control_13_FSP01_Execution	25/3/2014 5:56:09 μμ	25/3/2014 5:56:09 μμ	Succeeded
MRBL_13_FSP01_DAT	25/3/2014 5:56:09 μμ	25/3/2014 5:56:28 μμ	Succeeded
MRBL_Control_23_FSM03_Execution	25/3/2014 5:56:26 μμ	25/3/2014 5:56:26 μμ	Succeeded
MRBL_23_FSM03_DAT	25/3/2014 5:56:26 μμ	25/3/2014 5:56:39 μμ	Succeeded
MRBL_Control_03_FAX01_Execution	25/3/2014 5:56:26 μμ	25/3/2014 5:56:26 μμ	Succeeded
MRBL_03_FAX01_DAT	25/3/2014 5:56:26 μμ	25/3/2014 5:56:53 μμ	Succeeded
MRBL_Control_14_FSP03_Execution	25/3/2014 5:56:30 μμ	25/3/2014 5:56:30 μμ	Succeeded
MRBL_14_FSP03_DAT	25/3/2014 5:56:30 μμ	25/3/2014 5:57:28 μμ	Succeeded
MRBL_Control_24_FGLAL02_Execution	25/3/2014 5:56:40 μμ	25/3/2014 5:56:40 μμ	Succeeded
MRBL_24_FGLAL02_DAT	25/3/2014 5:56:40 μμ	25/3/2014 5:57:29 μμ	Succeeded
MRBL_Control_04_FMA02_Execution	25/3/2014 5:56:55 μμ	25/3/2014 5:56:55 μμ	Succeeded
MRBL_04_FMA02_DAT	25/3/2014 5:56:55 μμ	25/3/2014 5:57:06 μμ	Succeeded
MRBL_Control_05_FMA04_Execution	25/3/2014 5:57:08 μμ	25/3/2014 5:57:08 μμ	Succeeded

Ως τελικό αποτέλεσμα βλέπουμε ότι όλες οι ροές του workflow έχουν τρέξει με επιτυχία. Δίπλα σε κάθε γραμμή βλέπουμε succeeded που σημαίνει ότι η εκτέλεση διεκπεραιώθηκε χωρίς λάθη.

Παρόλο που όλες οι ροές έχουν τρέξει με επιτυχία θα πρέπει να εξετάσουμε κατά πόσο τα αποτελέσματα συμφωνούν με την πραγματικότητα. Για το λόγο αυτό έχει γίνει η σύγκριση των δικών μου αποτελεσμάτων με τα αποτελέσματα που δημιουργήθηκαν κατά την εκτέλεση των sql scripts. Τα sql scripts θεωρούνται έγκυρη πυγή αποτελεσμάτων.

Η ETL διαδικασία έτρεξε δύο φορές, μία με sql scripts και δεύτερη με Informatica για δύο διαφορετικές ημερομηνίες για να μην μπερδεύονται τα αποτελέσματα μεταξύ τους. Έπειτα δημιουργήθηκαν sql scripts που κάνουν σύγκριση μια - μια τις γραμμές των δύο πινάκων, από τις δύο διαφορετικές εκτελέσεις. Η σύγκριση δείχνει πόσες γραμμές είναι ίδιες, και στις δύο περιπτώσεις, και πόσες διαφορετικές. Επίσης δίπλα στο Comparison_Token γράφονται τα πεδία στα οποία εντοπίστηκε η διαφορά.

Στη συνέχεια βλέπουμε την σύγκριση που έχει γίνει με τον πίνακα FCU02_DAT. Στην πρώτη γραμμή το Comparison_Token είναι κενό, που σημαίνει ότι 6,226 εγγραφές δεν είχαν απολύτως καμία διαφορά μεταξύ τους. Στην δεύτερη και στην τρίτη γραμμή βλέπουμε ότι οι πίνακες είχαν 8 διαφορετικές γραμμές στο πεδίο AFM_FCU02 και 2,435 διαφορές στο πεδίο NUMBER_FCU02.

Οι διαφορές που ανακαλύφθηκαν οφείλονται στο γεγονός ότι με την Informaticata κενά που υπήρχαν στην αρχή του string αφαιρέθηκαν ενώ με τα scripts παρέμειναν. Στην ουσία αυτή η διαφορά δεν υποδεικνύει λάθη απλά διαφορά στην προσέγγιση που είχαν οι δύο μέθοδοι.

SQL Statements

```
+cast(case when i.ORIO_CR_FCU02 is null then case when s.ORIO_CR_FCU02 is null then ''
+cast(case when i.MERES_CR_FCU02 is null then case when s.MERES_CR_FCU02 is null then '
+cast(case when i.PRICE_LIST_FCU02 is null then case when s.PRICE_LIST_FCU02 is null th
+cast(case when i.DISCOUNT_FPA_FCU02 is null then case when s.DISCOUNT_FPA_FCU02 is nul
+cast(case when i.DROMOLOGIO_FCU02 is null then case when s.DROMOLOGIO_FCU02 is null th
+cast(case when i.SXOLIA_FCU02 is null then case when s.SXOLIA_FCU02 is null then '' el
+cast(case when i.DOY_FCU02 is null then case when s.DOY_FCU02 is null then '' else ',D
+cast(case when i.OFFICE_FCU02 is null then case when s.OFFICE_FCU02 is null then '' el
+cast(case when i.PHONE2_FCU02 is null then case when s.PHONE2_FCU02 is null then '' el
Comparison_Token
from (select * from FCU02_DAT where Company_ID_FCU02 = 1) i /* Informatica */
full
join (select * from FCU02_DAT where Company_ID_FCU02 = 9) s /* Script */
on s.Fiscal_Year_ID_FCU02 = i.Fiscal_Year_ID_FCU02
and s.CUSTOMER_FCU02 = i.CUSTOMER_FCU02
group by Comparison_Token
```

Results

	How_Many_from_Infa	How_Many_from_Script	Comparison_Token
1	6,226	6,226	
2	8	8	,AFM_FCU02
3	2,435	2,435	,NUMBER_FCU02

Σε γενικές γραμμές η σύγκριση έφερε θετικά αποτελέσματα. Οι τελικοί πίνακες δεν είχαν διαφορές μεταξύ τους. Οπότε το χτίσιμο της διαδικασίας με την βοήθεια ενός ETL εργαλείου θεωρείται επιτυχημένη.

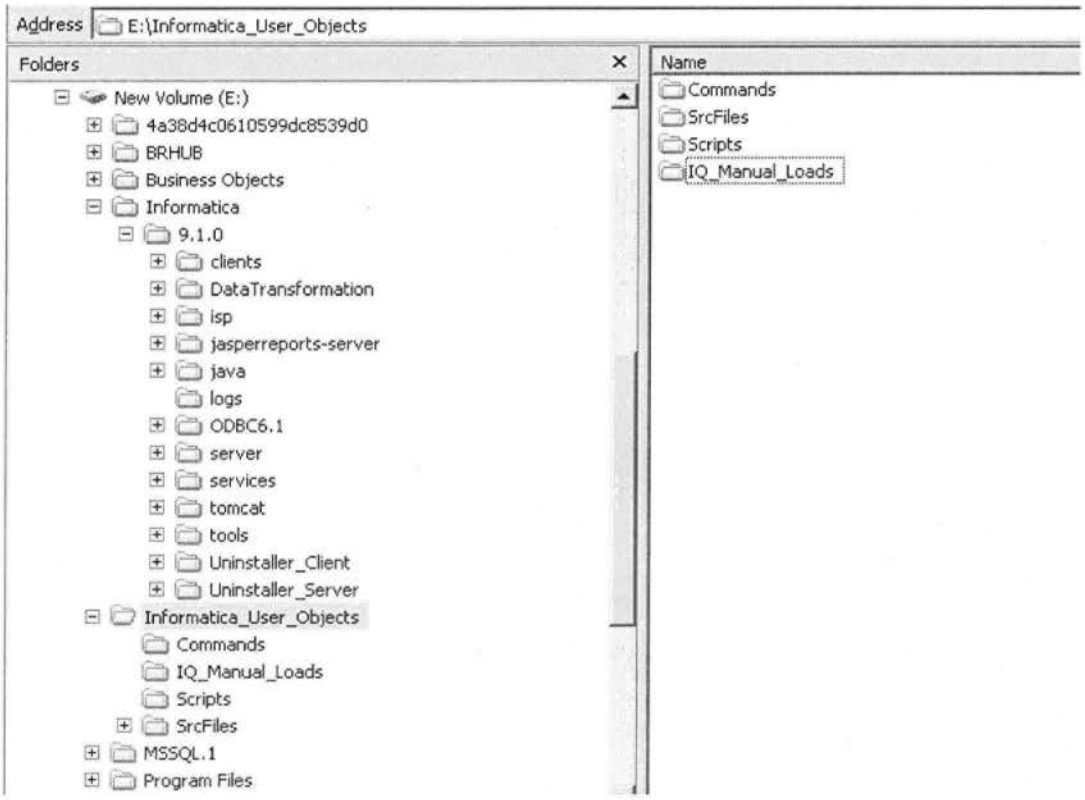
Version Independence (Ανεξαρτησία από την έκδοση του προγράμματος)

Informatica PowerCenter όπως όλα προγράμματα, εργαλεία και πλατφόρμες αλλάζει και βελτιώνεται συνεχόμενα με αποτέλεσμα να εκδίδονται καινούριες εκδόσεις κάθε χρόνο. Η έκδοση που δουλεύω το project είναι Informatica PowerCenter Designer Version 9.1 υπάρχει και πιο καινούρια version 9.5.

Υποθετικά μιλώντας, αν στο μέλλον η εταιρία αποφασίσει να αναβαθμίσει την έκδοση σε τελευταία που υπάρχει τότε θα συναντήσει ένα αρκετά σοβαρό πρόβλημα. Τα session δεν θα ξέρουν σε ποια directories να γράψουν τα target logs, session και workflow logs, loader logs και ctl files γιατί η νέα έκδοση θα έχει νέο directory. Με άλλα λόγια θα έπρεπε manually να αλλάξουμε πολλά σημεία μέσα στο workflow, worklets και sessions. Όσα πιο πολλά πράγματα θα πρέπει να αλλαχτούν τόσο μεγαλώνει και η πιθανότητα λάθους.

Ως παράδειγμα θα πάρουμε το αρχείο Load_Parameter_fma30_dat.sql που στην νέα έκδοση θα έπρεπε να βρίσκεται στο
E:\Informatica\9.5.0\server\infa_shared\Scripts\
Load_Parameter_fma30_dat.sql

Ενώ έχει δηλωθεί στο directory
E:\Informatica\9.1.0\server\infa_shared\Scripts\
Load_Parameter_fma30_dat.sql στην παρούσα υλοποίηση.

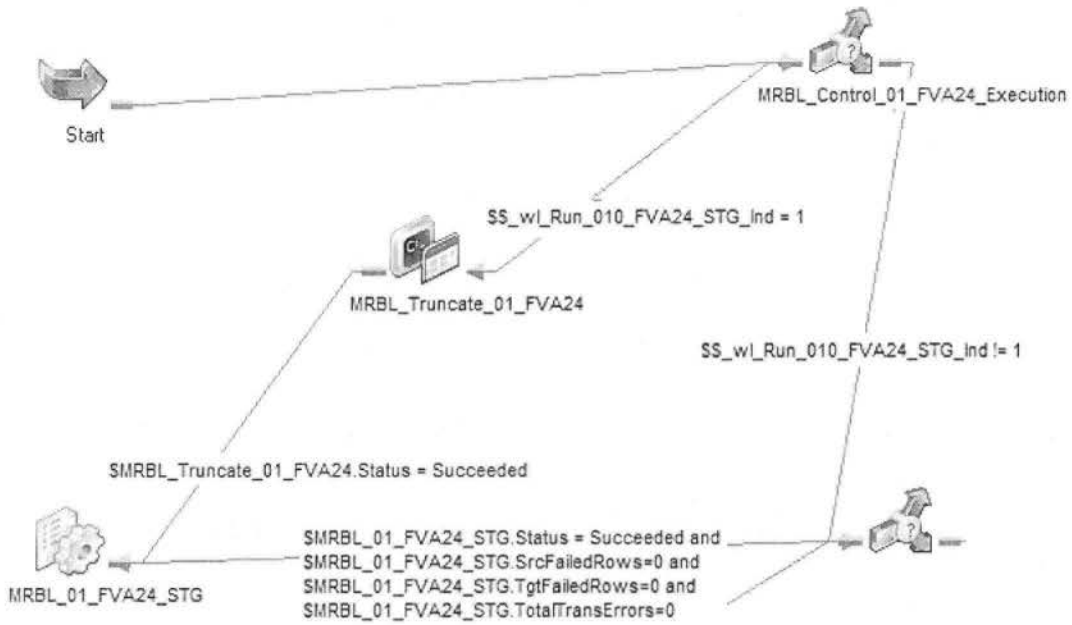


Version Independent Directory : E:\ Informatica_User_Objects

Για να προσπεραστεί αυτό το εμπόδιο δημιουργήθηκε ένα αρχείο Informatica_User_Objects έξω από το directory της Informatica όπου θα γράφονται τα πιο σημαντικά αρχεία του project που δεν πρέπει να χαθούν για κανένα λόγο. Όπως Commands, Src Files, Scripts και IQ Manual Loads.

Με αυτό τον τρόπο και να αλλάξει πλέον η έκδοση του προγράμματος το project δεν επηρεάζεται από αυτό. Μιας και τα πιο σημαντικά αρχεία βρίσκονται σε ανεξάρτητο directory από το directory το Informatica PowerCenter.

Μέθοδος Ελέγχου Ροής Σφαλμάτων (Error Control)

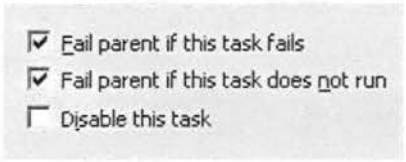


Το ενδεχόμενο εμφάνισης σημαντικών προβλημάτων κατά τη ροή της ETL διαδικασίας, οδήγησε στη δημιουργία μιας Μεθόδου Ελέγχου Ροής. Η μέθοδος αυτή, στην περίπτωση ουσιαστικού σφάλματος, διατηρεί τα δεδομένα των Τελικών Πινάκων όπως ήταν (nodeletion), αποκλείει την ενημέρωση τους με λανθασμένα δεδομένα (noinsertion – update). Έτσι, τα αποτελέσματα των Αναφορών θα αποτυπώνουν ορθές πληροφορίες, σύμφωνα με την τελευταία επιτυχή ολοκλήρωση της ETL διαδικασίας.

Έλεγχος σφαλμάτων πραγματοποιείται μετά από κάθε command-Truncate και κάθε session, δηλαδή σχεδόν σε κάθε βήμα της ροής. Το connection(το μπλε βελάκι) μετά το κάθε truncateτων πινάκων γίνεται έλεγχος για την ορθή εκτέλεση του, δηλαδή truncate.status=succeeded για να προχωρήσει στην εκτέλεση του session, αλλιώς πετάει μήνυμα λάθους και σταματάει η εκτέλεση της ροής.

Το ίδιο συμβαίνει όταν προκληθεί σφάλμα κατά την εκτέλεση ενός session. Μόνο που σε αυτήν την περίπτωση υπάρχουν περισσότερα κριτήρια ελέγχου. Το connection κοιτάει αν: status=succeeded and SrcFailedRows=0 and TrgFailedRows=0 and TotalTransErrors=0 και μόνο τότε περνάει στο επόμενο βήμα. Με άλλα λόγια το επιθυμητό είναι να έχουμε: καμία λάθος εγγραφή στο source και target πίνακα, και σύνολο των λαθών της ροής να είναι μηδέν.

Επίσης στο τέλος κάθε ροής workflow ή worklet υπάρχει ένα decision που έχει επιλεγμένα τα checkboxes: Fail parent if this task fails, Fail parent if this task does not run. Ενώ στα προηγούμενα decision γίνεται έλεγχος μόνο για sessions, στο τελευταίο γίνεται έλεγχος για όλη τη ροή. Αν για κάποιο λόγο δεν τρέξει το τελευταίο decision σημαίνει ότι δεν εκτελέστηκε ολόκληρη η ροή του worklet άρα πρέπει να σηματοδοτηθεί λάθος για όλο το workflow.



Συνοπτικά, έλεγχος σφαλμάτων γίνεται σε κάθε βήμα της ροής. Μετά από κάθε session, στο τέλος κάθε worklet και workflow.Τα λάθη καταγράφονται σε directory της Informatica ...\BadFiles. Απ' όπου μπορούμε να μάθουμε λεπτομέρειες από ταerror_log του κάθε session.

Ωστόσο υπάρχει περίπτωση να τρέξει το flow χωρίς κανένα error και ελέγχοντας τα δεδομένα στη βάση να είναι λάθος. Που σημαίνει ότι κάποια μετατροπή πήγε στραβά.

Συνήθως προβλήματα με λάθος δεδομένα προκύπτουν σε mappings όπου γίνεται μετατροπή αλφαριθμητικού σε αριθμό ή ημερομηνία. Για το λόγο αυτό έχει δημιουργηθεί ένας ξεχωριστός target πίνακας όπου καταγράφονται τα λάθη. Με άλλα λόγια όταν θα υπάρχουν προβλήματα με τις μετατροπές, τα λάθη αυτά θα καταγράφονται σε έναν πίνακα error.

Οι πίνακες αυτοί έχουν ως όνομα το όνομα του πίνακα στον οποίο προκύπτουν τα λάθη και κατάληξη _ERR, πχ FCCAVE_ERR. Επίσης εκεί καταγράφονται σε ποια γραμμή και στήλη έγινε το λάθος και πόσα λάθη έγιναν σε κάθε γραμμή. Αυτός ο έλεγχος είναι σημαντικός για την διόρθωση των μη προβλεπόμενων λαθών, ειδικά στο αρχικό στάδιο υλοποίησης της εργασίας.

Αξιολόγηση διαδικασίας / Σύγκριση με την παλιά μέθοδο

Ο στόχος της εργασίας ήταν δημιουργία μιας ολοκληρωμένης Αποθήκης Δεδομένων (data warehouse) με ETL εργαλείο (Informatica PowerCenter). Η υλοποίηση αυτή είχε πλεονεκτήματα και μειονεκτήματα σε σχέση με τα sqlscripts (που ήταν αρχικά υλοποιημένο το project).

Ένας από τους μεγαλύτερους παράγοντες που καθορίζει την αποτελεσματικότητα της εφαρμογής είναι ο χρόνος εκτέλεσης της. Το μεγαλύτερο πλεονέκτημα του κώδικα είναι η ταχύτητα στην εκτέλεση, μιας και τα sql scripts είναι κατά πολύ ελαφρύτερα από το συγκριτικά ογκώδες ETL. Στην πράξη όμως η ταχύτητα κατά κύριο λόγο εξαρτάται από το πόσο καλά έχει σχεδιαστεί το DW και στις δύο περιπτώσεις. Ένας όχι και τόσο καλά δομημένος κώδικας σίγουρα θα υστερεί στην ταχύτητα σε σύγκριση με ένα ETL.

Στην παρούσα εργασία ο χρόνος εκτέλεσης των scripts είναι γύρω στα 5 λεπτά και ο χρόνος εκτέλεσης με Informatica είναι γύρω στα 19 λεπτά. Παρατηρούμε ότι ο χρόνος εκτέλεσης με ETL εργαλείο είναι σχεδόν τριπλάσιος από αυτόν που τρέχουν τα scripts. Ήταν αναμενόμενη αυτή η διαφορά στον χρόνο παρόλο που κατά την υλοποίηση με Informatica έγιναν μεγάλες προσπάθειες να ελαφρύνει όσο γίνεται περισσότερο ο κώδικας.

Πιο συγκεκριμένα, στην υλοποίηση με scripts έχουμε τρία διακριτά βήματα που εκτελούνται αυστηρά το ένα μετά το άλλο, που είναι το TMP (extract), STG (transform) και DAT (load) στάδια. Ενώ με την Informatica τα στάδια TMP και STG έχουν συγχωνευτεί σε ένα, το STG. Ωστόσο ένα σημεία που η Informatica σίγουρα υστερούσε είναι ότι σχεδόν κάθε διαδικασία (task) γινόταν πιο περίπλοκη απ' ότι χρειαζόταν να είναι, εκεί που με κώδικα θα ξεμπερδεύαμε με ένα απλό update με την Informatica χρειάζεται να γίνει lookup σε άλλους πίνακες. Και σίγουρα ένα lookup είναι πιο βαριά διαδικασία από το update.

Σε γενικές γραμμές η υλοποίηση και ο σχεδιασμός που έχει γίνει με τα sql scripts αποδείχτηκε γρηγορότερος από τοETL εργαλείο.

Ας περάσουμε στην περεταίρω σύγκρισή των δύο μεθόδων. Ένα από τα μεγαλύτερα πλεονεκτήματα της Informatica σε σχέση με τον κώδικα sql είναι ευκολία στην συντήρηση. Ο κώδικας από την μια μπορεί να είναι πολύ βολικός στην ταχύτητα και στις άπειρες δυνατότητες που δίνει στην υλοποίηση ενός έργου ωστόσο η συντήρηση του κώδικα είναι πολύ δύσκολη.

Σε τέτοιες περιπτώσεις όλο το έργο και η συντήρηση του κρέμεται κυριολεκτικά από ένα άτομο, κάτι που είναι τελείως αντιπαραγωγικό και μη ωφέλιμο για τον πελάτη. Ας υποθέσουμε ότι παρουσιάζεται ένα πρόβλημα στην παραγωγή και ο δημιουργός του έργου λείπει σε διακοπές ή είναι

απασχολημένος με άλλα έργα, τότε ο θέση του πελάτη είναι πολύ δυσμενής. Δεν έχει ούτε την ευχέρεια χρόνου να περιμένει ούτε μπορεί εύκολα να προσλάβει κάποιον άλλο προγραμματιστή να λύσει το πρόβλημα, χάνοντας πάλι χρόνο μέχρι να προσαρμοστεί ο συγκεκριμένος. Ενώ με ένα ETL εργαλείο η μετάδοση γνώσης είναι πολύ ευκολότερη.

Τα ETL εργαλεία, όπως η Informatica, έχουν ωραίο γραφικό περιβάλλον που είναι πολύ φιλικό προς τον χρήστη. Με αυτά τα χαρακτηριστικά η συντήρηση γίνεται πολύ εύκολη. Δεν απαιτείται εξειδικευμένο προσωπικό για την συντήρηση της αποθήκης (DW). Ακόμα και οι χρήστες που δεν ξέρουν τίποτα από προγραμματισμό με ευκολία μπορούν να μάθουν να χειρίζονται, να παρακολουθούν (monitoring) και να ξανατρέχουν τις βασικές ροές στην Informatica. Με άλλα λόγια πολύ πιο εύκολο θα μάθει κάποιος να χειρίζεται ένα ETL εργαλείο πάρα να τροποποιεί ή να συμπληρώνει περίπλοκο κώδικα.

Ένα άλλο μεγάλο πλεονέκτημα ενός ETL εργαλείου είναι ότι μέσα σε μια οθόνη εύκολα και γρήγορα μπορούμε να δούμε όλες τις ροές και τις διαδικασίες. Όπως αναφέρθηκε λίγο πιο πάνω, η Informatica έχει γραφικό περιβάλλον προγραμματισμού, αποτελείται από πινακάκια που συνδέονται μεταξύ τους με γραμμές. Η οπτική απεικόνιση του κώδικα έχει μεγάλα πλεονεκτήματα όταν θέλουμε με μια ματιά να εντοπίσουμε ποιες είναι οι συνδέσεις και εξαρτήσεις (dependencies) που έχουν οι πίνακες μεταξύ τους. Είναι μεγάλο πλεονέκτημα να φαίνεται γρήγορα – γρήγορα τι θα επηρεαστεί σε περίπτωση αλλαγής στη δομή ή αν γίνει κάπου σφάλμα από πού έχει προέλθει. Είναι φανερό ότι σε αυτό τον τομέα τα sql scripts υστερούν κατά πολύ.

Επίλογος

Το Data Warehousing έχει μπει εδώ και χρόνια δυναμικά στην αγορά και συνεχίζει να αναπτύσσεται και προσαρμόζεται στα καινούρια δεδομένα. Ζούμε σε μια εποχή που η πληροφορία είναι σε βάρος χρυσού και αυτό συνεπάγεται ότι οι τομείς που ασχολούνται με την επεξεργασία δεδομένων θα συνεχίσουν την κερδοφόρα εξέλιξη τους.

Εν κατακλείδι, παρόλο το μεγάλο όγκο και βάρος των ETL εργαλείων υπάρχουν πολλά πλεονεκτήματα που σε τελική ανάλυση υπερκαλύπτουν τα μειονεκτήματα. Με την διεκπεραίωση της συγκεκριμένης πτυχιακής εργασίας αποδείχτηκε ότι τα ETL εργαλεία έχουν μεγάλη συνεισφορά στην ανάπτυξη των Data Warehouse. Κάνουν τις αποθήκες ευέλικτες και εύκολα συντηρήσιμες. Πλέον η επικοινωνία της αποθήκης με διαφορετικές βάσεις δεδομένων δεν είναι εμπόδιο με την βοήθεια ενός ETL εργαλείου καθώς και η ανάπτυξη αναφορών (reports) που αποτελεί από τα σημαντικότερα σημεία σχεδόν για όλες τις εταιρίες.

Βιβλιογραφία

Ralph Kimball (1998)*The Data Warehouse Lifecycle Toolkit*. Wiley

Ralph Kimball (2004) The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data

Bill Inmon (1981) Effective Data Base Design. Prentice Hall

Bill Inmon (2000) *Exploration Warehousing: Turning Business Information into Business Opportunity*. With R. H. Terdeman, John Wiley and Sons

Informatica PowerCenter Level 1 Developer Student Guide(pdf)

Informatica PowerCenter User Manual (pdf)

Links

1. http://datawarehouse4u.info/Data-Warehouse-Schema-Architecture.html
2. http://docs.oracle.com/cd/B10501_01/server.920/a96520/logical.htm#745
3. http://docs.oracle.com/cd/B10501_01/server.920/a96520/concept.htm
4. http://www.1keydata.com/datawarehousing/datawarehouse.html
5. http://www.tutorialspoint.com/dwh/dwh_schemas.htm
6. http://en.wikipedia.org/wiki/Fact_table
7. http://en.wikipedia.org/wiki/Data_warehouse
8. http://en.wikipedia.org/wiki/Business_intelligence
9. http://informaticareference.wordpress.com/2011/12/07/what-are-the-output-files-that-the-informatica-server-creates-during-the-session-running/
10. http://informaticatutorials-naveen.blogspot.gr/#axzz30x5cWsxL
11. https://community.informatica.com/message/43758
12. http://www.1keydata.com/datawarehousing/inmon-kimball.html
13. http://www.investopedia.com/terms/d/data-warehousing.asp
14. http://en.wikipedia.org/wiki/Extract,_transform,_load
15. http://www.etltools.net/etl-tools-comparison.html
16. http://www.kimballgroup.com/2008/04/06/should-you-use-an-etl-tool/
17. http://www.intechopen.com/books/supply-chain-management-new-perspectives/intelligent-value-chain-networks-business-intelligence-and-other-ict-tools-and-technologies-in-suppl