ΕΞΩΦΥΛΛΟ

Περιεχόμενα

Ευρετήριο πινάκων

Ευρετήριο σχημάτων

Κεφάλαιο 1

Εισαγωγή

Η εργασία η οποία εκπονήθηκε για αυτή τη διπλωματική εργασία αφορά στην υλοποίηση ενός γενικευμένου, αντικειμενοστραφούς μεταγλωττιστή για τη γλώσσα ANSI-SQL-2. Στόχος της παρούσας εργασίας είναι η παρουσίαση ενός σχεσιακού σχήματος με τη σύνταξη της SQL, το οποίο θα είναι ανεξάρτητο από πηγές δεδομένων.

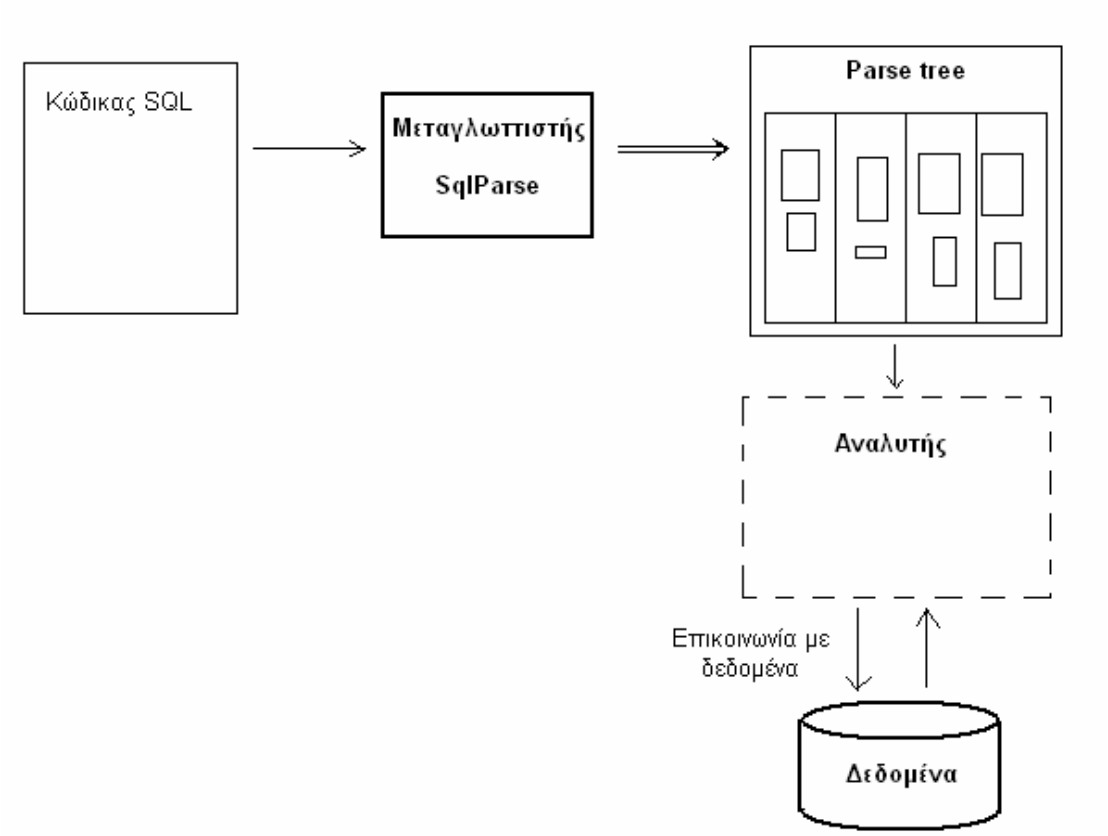
Η χρησιμότητα της εργασίας αυτής απορρέει από την ανάγκη για έναν μεταγλωττιστή ο οποίος θα είναι ανεξάρτητος από το σύστημα της βάσης δεδομένων, αλλά θα παρέχει στον προγραμματιστή τη δυνατότητα να τον προσαρμόσει σε αυτό που επιθυμεί με την κατάλληλη υλοποίηση. Η απευθείας μετάφραση της γλώσσας SQL είναι δύσκολη και τα συστήματα βάσεων δεδομένων είναι πολλά και διαφορετικά μεταξύ τους. Απαιτείται μία κατάλληλη μορφή αναπαράστασης των βασικών χαρακτηριστικών της δομής του κώδικα Ακόμη πολλές φορές μπορεί ο διαχειριστής μιας βάσης να ενδιαφέρεται να κάνει απλό parsing εντολών SQL χωρίς να επενεργήσει στα δεδομένα της βάσης.

Η λύση στα παραπάνω ζητήματα δίνεται στην υλοποίηση με τη χρήση των δέντρων αφηρημένης σύνταξης (Abstract Syntax Trees (AST)).Τα AST είναι δεντροειδείς δομές δεδομένων με ακανόνιστη μορφή τα οποία περιγράφουν τη βασική δομή του κώδικα παραβλέποντας ωστόσο ασήμαντες συντακτικές λεπτομέρειες. Τα δομικά στοιχεία της γλώσσας αναπαρίστανται με διαφόρων τύπων κόμβους του AST ή ίδιου τύπου κόμβους με διαφοροποίηση σε κάποιες τιμές.

Ο μεταγλωττιστής SqlParse που κατασκευάστηκε σε αυτή την εργασία διαβάζει μια σειρά από εντολές και ερωτήματα SQL και κατασκευάζει ένα AST που περιγράφει πλήρως με συγκεκριμένες κλάσεις το συντακτικό περιεχόμενο του κώδικα SQL. Οι δημιουργούμενες κλάσεις περιέχουν μεθόδους που ανασύρουν το συντακτικό περιεχόμενο του αρχικού κώδικα. Κάθε κλάση της δημιουργούμενης δομής δεδομένων διαθέτει μεταβλητές και μεθόδους που καθιστούν τα δεδομένα του συντακτικού του κώδικα εύκολα προσιτά στον προγραμματιστή και τις οποίες μπορεί να χρησιμοποιήσει ώστε να δημιουργήσει τη δική του εφαρμογή αλληλεπίδρασης με δεδομένα

Στη συνέχεια ,με βάση αυτό το AST, γίνεται ο σημασιολογικός έλεγχος. Οι εντολές της SQL χρησιμοποιούνται για την προσθήκη και αφαίρεση στοιχείων σε ένα πίνακα συμβόλων, καθώς και για τον έλεγχο της ορθότητας των εντολών και των ερωτημάτων με βάση αυτόν τον πίνακα συμβόλων.

Στο παρακάτω σχήμα παριστάνεται ο τρόπος με τον οποίο λειτουργεί ο μεταγλωττιστής.



Σχήμα 1.1: Σχηματική λειτουργία του μεταγλωττιστή Sqlparse.*Ο μεταγλωττιστής διαβάζει ένα κομμάτι κώδικα SQL και αφού το αναλύσει και κάνει τους απαραίτητους ελέγχους, παράγει τη δομή δεδομένων parse tree.Η δομή αυτή αποτελείται από αντικείμενα συγκεκριμένων κλάσεων τέτοιων ώστε να περιγράφουν πλήρως τα περιεχόμενα του κώδικα SQL και επιτρέπουν την αποκωδικοποίηση της δομής στον αρχικό κώδικα. Έτσι ένας κατάλληλος αναλυτής μπορεί να χρησιμοποιήσει τη δομή αυτή για να δημιουργήσει την κατάλληλη αλληλεπίδραση του κώδικα με συγκεκριμένης σχεδίασης βάση δεδομένων*.

Ο αναλυτής που περιγράφεται παραπάνω δεν αποτελεί μέρος αυτής της εργασίας, αλλά σκοπός της είναι να επιτρέπει την εύκολη υλοποίηση τέτοιων ώστε να προσαρμόζονται σε διαφορετικούς τύπους βάσεων δεδομένων. Ωστόσο έχει υλοποιηθεί ένα κομμάτι της σημασιολογικής ανάλυσης που αφορά στην προσθήκη και εύρεση δεδομένων στον πίνακα συμβόλων.

Στα πλεονεκτήματα του παραπάνω μεταγλωττιστή είναι και το ότι μπορεί να επεκταθεί και να χρησιμοποιηθεί για διάφορες εφαρμογές. Ο μεταγλωττιστής προσφέρει ευελιξία στη σχεδίαση αφού αποτελεί τη βάση πάνω στη οποία μπορεί να ‘χτιστεί’ μια εφαρμογή ανάσυρσης δεδομένων από συγκεκριμένο σχήμα και επιπρόσθετα παρέχει τη δυνατότητα στο χρήστη να φτιάξει το δικό του πίνακα συμβόλων. Ωστόσο ο συντακτικός αναλυτής είναι προκαθορισμένος και πρέπει να τροποποιηθεί ριζικά ώστε να επεκταθεί και να καλύπτει περισσότερες εντολές SQL.

Η παρούσα υλοποίηση υποστηρίζει ένα μεγάλο μέρος του προτύπου ANSI-SQL-92 και καλύπτει τη λειτουργικότητα για της πιο κοινές εκφράσεις της SQL. Παράλληλα η υλοποίηση του πίνακα συμβόλων και της σημασιολογικής ανάλυσης παρέχει περαιτέρω έλεγχο της ορθότητας του κώδικα. Η σημασιολογική ανάλυση

καθώς και η εκτύπωση σε σχεσιακή άλγεβρα υλοποιείται με βάση τα δεδομένα της δομής Parse tree και επιδεικνύει πώς μπορούν να χρησιμοποιηθούν οι κλάσεις του πακέτου parsetree για την επέκταση του μεταγλωττιστή.

Κεφάλαιο 2

Σχετική εργασία

Στο κεφάλαιο αυτό παρατίθενται μερικά βασικά στοιχεία για τη γλώσσα SQL της, οι πρώτες μορφές της και τα πρότυπά της. Κυρίως όμως γίνεται αναφορά σε εργασίες σχετικές με την παρούσα, οι οποίες είτε χρησιμοποιούν κομμάτια της SQL, είτε λειτουργούν με διάφορους τρόπους ως σύνδεσμοι σχεσιακών μοντέλων δεδομένων με δεδομένα.

2.1 Ιστορία και πρότυπα της SQL

2.1.1 Ιστορία της SQL

Η πρώτη μορφή σχεσιακού μοντέλου εμφανίζεται το 1970.Ο E.F.Codd, μέλος τότε του εργαστηρίου έρευνας της IBM εξέδωσε μια δημοσίευση με θέμα «Ένα σχεσιακό μοντέλο δεδομένων για μεγάλες κοινόχρηστες βάσεις δεδομένων», στο οποίο εξέθεσε της βασικές αρχές για τη διαχείριση βάσεων δεδομένων: το λεγόμενο σχεσιακό μοντέλο. Το αντικείμενο της εργασίας αποτέλεσε έναυσμα για ευρεία έρευνα και πειραματισμό σε πανεπιστήμια και βιομηχανικά εργαστήρια και αποτέλεσε την αρχή για πολλά σημερινά σχεσιακά προϊόντα.

Σύμφωνα με την παραπάνω έρευνα μια σχεσιακή γλώσσα είναι αυτή που αντιλαμβάνεται, με συγκεκριμένη συντακτική μορφή, μερικά ή όλα τα χαρακτηριστικά της αφηρημένου σχεσιακού μοντέλου. Αρκετές τέτοιες γλώσσες δημιουργήθηκαν της αρχές της δεκαετίας του ΄70.Μία από αυτές ήταν η "Structured English Query Language"(SEQUEL), η οποία ορίστηκε από μία ομάδα ανθρώπων που εργάζονταν στο εργαστήριο έρευνας San Jose της IBM και πρωτοϋλοποιήθηκε με ένα πρωτότυπο της IBM λεγόμενο SEQUEL-XRM(1974-75).

Βασισμένη στην εμπειρία με τη SEQUEL-XRM, μια ανανεωμένη έκδοση του προτύπου αυτού, το SEQUEL/2, ορίστηκε το 1976-77.Στη συνέχεια ένα πιο φιλόδοξο πρότυπο της IBM έκανε την εμφάνιση του, το SystemR.Το SystemR, μια υλοποίηση του υπερσυνόλου του SEQUEL/2(ονομαζόμενο και ως SQL), έγινε λειτουργικό το 1977 και εγκαταστάθηκε σε έναν αριθμό θέσεων χρηστών, στο εσωτερικό της IBM και σε επιλεγμένες θέσεις πελατών της. Στη διάρκεια ζωής του SystemR έγιναν πλήθος αλλαγών στη γλώσσα SQL κυρίως λόγω επιθυμιών των χρηστών.

Εξαιτίας του SystemR, άλλες εταιρίες άρχισαν να κατασκευάζουν τα δικά τους προϊόντα βασισμένα στην SQL, ένα από τα οποία ήταν το ORACLE.Αργότερα το 1981 η IBM ανακοίνωσε ένα προϊόν SQL ονόματι SQL/DS για το περιβάλλον VSE.Στα επόμενα χρόνια πολλοί εμπορικοί οργανισμοί ανακοίνωσαν προϊόντα βασισμένα σε SQL.Τα προϊόντα αυτά άλλοτε ήταν εντελώς καινούργια όπως το DG/SQL(1984) ή ήταν διεπαφές σε ήδη υπάρχοντα της το INGRES(1982,1985). Σήμερα υπάρχουν εκατοντάδες διάλεκτοι της SQL και έχει γίνει πλέον το αδιαμφισβήτητο πρότυπο σε ό,τι αφορά τις σχεσιακές βάσεις δεδομένων.

2.1.2 Πρότυπα της SQL

Η γλώσσα SQL έχει πρότυπα από τους διεθνείς οργανισμούς προτύπων ISO και ANSI.Το τρέχον πρότυπο είναι το ISO/IEC 9075:1999, αναφερόμενο και ως SQL-99.

Το 1982 το αμερικάνικο ινστιτούτο διεθνών προτύπων(ANSI) συνέταξε μια επιτροπή(X3H2) για να υλοποιήσει μια πρόταση της προτύπου για μια σχεσιακή γλώσσα. Το 1987 το πρότυπο αυτό έγινε αποδεκτό και από το διεθνή οργανισμό προτυποποίησης (ISO) και είναι γνωστό ως SQL-1 ή SQL/86.Μια βελτίωση αυτού που περιείχε υποστήριξη για αναφορική ακεραιότητα εκδόθηκε το 1989. Επιπρόσθετα, το ίδιο έτος, υιοθετήθηκε ένα παρεμφερές πρότυπο ονομαζόμενο ως *Database Language Embedded SQL*.

Η επόμενη σημαντική έκδοση προτύπου ήταν το 1992 συχνά αναφερόμενη ως SQL-2 ή SQL/92. Η προσπάθεια για τη νέα αυτή έκδοση της SQL ξεκίνησε όταν μια ομάδα ανθρώπων άρχισε να εργάζεται για να δημιουργήσει ένα σύνολο πρόσθετων ιδιοτήτων που θα υποστηρίζουν διαδραστικότητα μεταξύ ανόμοιων συστημάτων. Οι επιτροπές των ANSI και ISO εργάστηκαν για μερικά χρόνια ώστε να ορίσουν την ανανεωμένη έκδοση του αρχικού προτύπου.

Το 1996 εκδόθηκε ένα πρότυπο για stored procedures (SQL/PSM).Το τελευταίο, το οποίο όπως αναφέρθηκε είναι και το τρέχον, είναι το πρότυπο SQL-99 ή SQL-3.

Ο μεταγλωττιστής της εργασίας αυτής είναι βασισμένος στο πρότυπο SQL-92 και καλύπτει ένα σημαντικό μέρος της γραμματικής του. Ακόμη καλύπτει τη δήλωση εναυσμάτων σύμφωνα με το πρότυπο SQL/99.Η πλήρης γραμματική του παρουσιάζεται στο παράρτημα σε μορφή BNF.Στο μεγαλύτερο μέρος της έχει διατηρηθεί η ονοματολογία των μη-τερματικών του προτύπου για συμβατότητα και για να διευκολυνθεί η μελλοντική επέκταση.

2.2 Διάλεκτοι της SQL

Παρακάτω παρουσιάζονται μερικές διάλεκτοι της SQL, γλώσσες δηλαδή βασισμένες στη λογική και τη σύνταξη της SQL που χρησιμοποιούνται για την εύρεση πληροφοριών από ευρύτερους αποθηκευτές δεδομένων.

2.2.1 WebSQL : Υποβολή ερωτημάτων στον παγκόσμιο ιστό

Η WebSQL είναι μια γλώσσα που μοιάζει συντακτικά με την SQL και έχει σκοπό την εύρεση σελίδων στον παγκόσμιο ιστό φιλτράροντας τις με βάση κάποια κριτήρια που επιλέγει ο χρήστης, όπως ακριβώς η SQL αναζητά δεδομένα σε πίνακες.
Η WebSQL κάνει χρήση των μηχανών αναζήτησης και των συνδέσμων μεταξύ σελίδων για να δημιουργήσει έναν ‘εικονικό γράφο’ ιστοσελίδων πάνω στον οποίο ενεργούν τα ερωτήματα της γλώσσας.

Δεδομένου ότι ο παγκόσμιος ιστός δε αποτελεί μέρος μιας βάσης δεδομένων, αλλά αντίθετα περιέχει ετερογενή και σκόρπια δεδομένα, η WebSQL κάνει την εξής πρωταρχική θεώρηση : συσχετίζει κάθε αντικείμενο του ιστού με πλειάδες ενός εικονικού πίνακα με τη μορφή

Document[url,title,text,type,length,modif]

όπου *url* αντιστοιχεί στη διεύθυνση του αντικειμένου στον ιστό,*title* και *text* στον τίτλο και στο κείμενο του αντίστοιχα,*type* στον τύπο του(HTML,Postscript,εικόνα κ.τ.λ),*length* στο μήκος του και *modif* στην ημερομηνία τροποποίησης. Όλα τα παραπάνω χαρακτηριστικά είναι συμβολοσειρές. Το url είναι το πρωτεύον κλειδί. Ακόμη, η δομή των συνδέσμων είναι ένα ζητούμενο που πρέπει να είναι γνωστό. Έτσι, όπως και παραπάνω, θεωρείται ένας πίνακας Anchor με τη μορφή

Anchor[base,href,label]

που περιγράφει τους συνδέσμους μιας σελίδας. Το χαρακτηριστικό *base* αντιστοιχεί στο URL του εγγράφου HTML που περιέχει το σύνδεσμο, *href* είναι το έγγραφο στο οποίο παραπέμπει ο σύνδεσμος και *label* είναι η περιγραφή του.

Με βάση τις παραπάνω θεωρήσεις μπορούν να γραφούν τώρα ερωτήματα για αναζήτηση εγγραφων στον ιστό, με τον τρόπο που φαίνεται στο παρακάτω παράδειγμα.

Παράδειγμα:Εύρεση όλων των εγγράφων HTML που είναι σχετικά με ‘hypertext’.

```
SELECT   d.url,d.title,d.length,d.modif
FROM     Document d
         SUCH THAT d MENTIONS “hypertext”
WHERE    d.type =  “text/html”
```

Γενικότερα μπορούν να θεωρηθούν δύο τύποι πλειάδων, οι Node και Link με την ακόλουθη μορφή:

Node = [id:Oid, ..., a_i : t_i, ...]

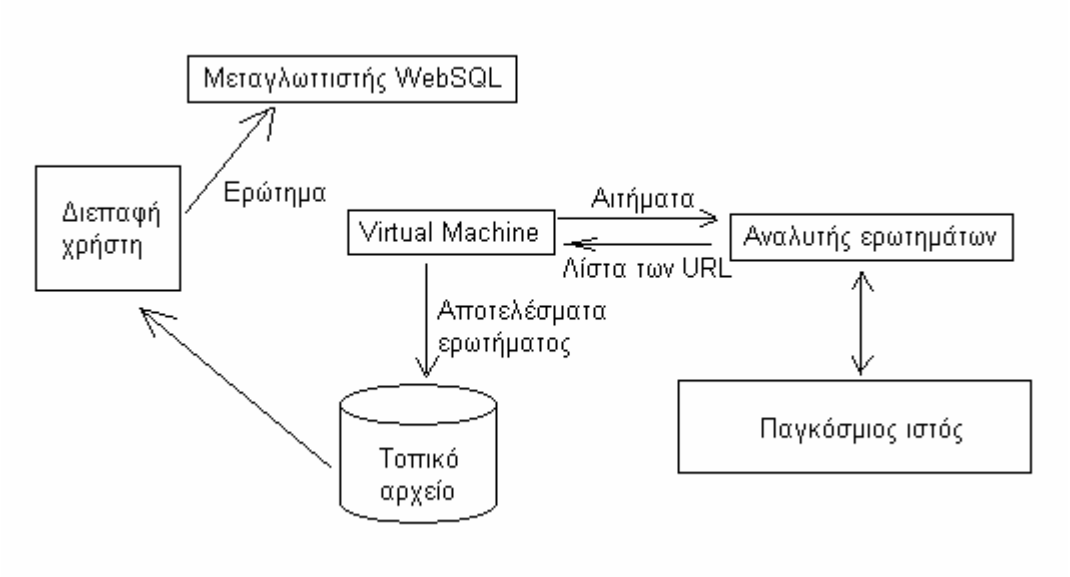
Link = [from:Oid, to :Oid, ...,b_j :t_j, ...]

όπου Oid είναι ένας τύπος προσδιορισμού αντικειμένων. Στο μοντέλο του παγκόσμιου ιστού, τα έγγραφα αντιστοιχίζονται σε αντικείμενα τύπου Node ενώ οι σύνδεσμοι σε αντικείμενα τύπου Link.Οι προσδιοριστές των αντικειμένων είναι τα URL τους.

Τα παραπάνω χρησιμοποιούνται για να εκφραστεί ο παγκόσμιος ιστός ως ένας εικονικός γράφος. Δεδομένου ότι το σύνολο του ιστού είναι απροσδιόριστο(δεν είναι δυνατόν να παραχθεί μια λίστα με όλα τα διαθέσιμα έγγραφα σε μια δεδομένη στιγμή), υπάρχουν δύο τρόποι να βρεθούν έγγραφα. Με μηχανές αναζήτησης και με πλοήγηση ξεκινώντας από ήδη γνωστά έγγραφα.

Επειδή με το κλασικό σχεσιακό σχήμα είναι πρακτικά αδύνατο να γίνουν πράξεις στο σύνολο του παγκόσμιου ιστού, δεδομένου ότι στη χειρότερη περίπτωση θα πρέπει να απαριθμηθούν όλα τα έγγραφα, η WebSQL χρησιμοποιεί τις *συνθήκες εύρους* (range conditions) που επιβάλλουν περιορισμούς στις μεταβλητές ενός ερωτήματος. Έτσι περιορίζεται το εύρος του εξεταζόμενου κάθε φορά χώρου δεδομένων σε ένα πεπερασμένο αλλά εξίσου ουσιώδες σύνολο.

Η αρχιτεκτονική του συστήματος της WebSQL φαίνεται στο παρακάτω σχήμα.



Σχήμα 2.1 : Η αρχιτεκτονική του συστήματος WebSQL

Ο μεταγλωττιστής της WebSQL αναλύει κάθε ερώτημα και το μεταφράζει σε έναν ένθετο βρόχο σε μια ειδικά σχεδιασμένη γλώσσα αντικειμένων. Το παραγόμενο πρόγραμμα αντικειμένων εκτελείται από ένα μεταφραστή ο οποίος υλοποιεί μια ετερογενή στοίβα με αντικείμενα διαφόρων τύπων, από ακέραιους και συμβολοσειρές ως ολόκληρες λίστες αντικειμένων τύπου Node και Link.Στη συνέχεια γίνεται πιστοποίηση των παραπάνω και παράγονται αποκλειστικά vectors αντικειμένων τύπου Node και Link.Ο μεταφραστής διαχωρίζει 3 τύπους στοιχείων και ανάλογα με τον τύπο τους επιστρέφει ως αποτελέσματα της αναζήτησης αντίστοιχα:

- Μονοπάτια του ιδεατού γράφου
- Αποτελέσματα μηχανών αναζήτησης ή
- το σύνολο των συνδέσμων μιας ιστοσελίδας τύπου HTML.

2.2.2 Java EE : EJBQL

Η EJBQL (Enterprise JavaBeans Query Language) είναι μια γλώσσα ερωτημάτων που εφαρμόζεται για την επιλογή αντικειμένων ή τιμών τα οποία βασίζονται σε αφηρημένους τύπους και σχέσεις των οντοτήτων JavaBeans.

Κάθε έκφραση στην EJBQL έχει ένα τύπο.Ο τύπος της έκφρασης εξάγεται από τη δομή της έκφρασης, τον αφηρημένο τύπο των δηλώσεων βοηθητικών μεταβλητών και τους τύπους των πεδίων cmp(container-managed persistence) και cmr(container-managed relationship). Οι επιτρεπόμενοι τύποι στην EJBQL είναι οι αφηρημένοι τύποι των οντοτήτων JavaBeans και των πεδίων cmp.Ο τύπος ενός αφηρημένου σχήματος μιας οντότητας bean εξάγεται από την κλάση της οντότητας.

Τα ερωτήματα της γλώσσας χρησιμοποιούνται για να επιλέξουν αντικείμενα των κλάσεων με συγκεκριμένα κριτήρια που βασίζονται στα χαρακτηριστικά των κλάσεων. Ένα ερώτημα μπορεί να χρησιμοποιηθεί για να ορίσει μια μέθοδο εύρεσης ή μια μέθοδο επιλογής ενός cmp entity bean.

Η EJBQL υποστηρίζει τα παρακάτω στοιχεία της SQL:

- Την πρόταση SELECT η οποία προσδιορίζει τον τύπο των αντικειμένων ή τις τιμές που θα επιλεγούν

- Την πρόταση FROM, η οποία παρέχει δηλώσεις που περιγράφουν την περιοχή επιρροής στην οποία εφαρμόζονται οι εκφράσεις των προτάσεων SELECT και WHERE.
- Προαιρετικά την πρόταση WHERE, η οποία μπορεί να περιορίζει με κριτήρια τα αποτελέσματα που επιστρέφονται από το ερώτημα.
- Προαιρετικά την πρόταση ORDER BY, η οποία ταξινομεί τα αποτελέσματα που επιστρέφονται από το ερώτημα.

Ένα παράδειγμα χρήσης της γλώσσας φαίνεται στη συνέχεια.

SELECT DISTINCT OBJECT(o)
FROM Order AS o, IN(o.lineItems) AS l
WHERE l.shipped = FALSE

Το παραπάνω ερώτημα πλοηγείται στο πεδίο cmr *lineItems* του αφηρημένου τύπου *Order* και χρησιμοποιεί το πεδίο *shipped* του *LineItem* για αν επιλέξει συγκεκριμένα αντικείμενα του *Order*.

Η EJBQL είναι ανεξάρτητη από τον τρόπο που συνδέονται τα beans με έναν καταχωρητή δεδομένων και μπορεί να μεταφέρεται. Μεταγλωττίζεται σε SQL κατά το χρόνο ανάπτυξης με βάση την αντιστοίχιση του σχήματος για το JavaBean.

2.2.3 Η γλώσσα OQL(Object Query Language)

Η γλώσσα OQL χρησιμοποιείται από διεργασίες του λειτουργικού συστήματος διαχείρισης δικτύου Mobile Wireless Fault Mediator(MWFM NMOS) για να μεταφέρει δεδομένα μεταξύ διαφόρων βάσεων δεδομένων. Επιτρέπει στο χρήστη τη σύνδεση με οποιαδήποτε εφαρμογή MWFM ώστε να ανασύρει δεδομένα από την εφαρμογή. Η σύνδεση με τις διάφορες εφαρμογές MWFM επιτυγχάνεται με τον πάροχο υπηρεσιών *riv_oql* ,ο οποίος προσδιορίζει το domain και την υπηρεσία στην οποία θέλει ο χρήστης να υποβάλλει ερωτήματα.

Οι διαφορές της OQL με την SQL είναι :

- Η OQL έχει τη δυνατότητα να υποστηρίζει αναφορές των πινάκων μέσω αντικειμένων. Είναι δυνατόν να υπάρχουν αντικείμενα ένθετα σε αντικείμενα.
- Δεν υποστηρίζονται όλες οι λέξεις κλειδιά της SQL και στην OQL.
- Η OQL παρέχει τη δυνατότητα να διεξάγονται μαθηματικοί υπολογισμοί μέσα από τις εκφράσεις της.

2.2.4 Εφαρμογές CAD

Η σχεδίαση με τη χρήση υπολογιστή (Computer Aided Design, CAD) περιλαμβάνει τη χρήση των υπολογιστών σε διάφορα στάδια σχεδίασης μηχανικής. Η CAD έχει μεγάλα κομμάτια δεδομένων με πολύπλοκες μορφές τα οποία χρειάζεται να αποθηκευτούν και να γίνει σωστή διαχείρισή τους. Δεδομένου ότι τα συστήματα βάσεων δεδομένων παρέχουν ανεξαρτησία μεταξύ του προγράμματος το οποία έχει πρόσβαση στα δεδομένα και τη βάση δεδομένων, είναι σημαντικό να χρησιμοποιούνται συστήματα βάσεων δεδομένων για την αποθήκευση δεδομένων CAD.

Τα γραφικά αντικείμενα μπορούν να δημιουργηθούν στη σχεδίαση με χρήση υπολογιστή με τη χρήση ήδη υπαρχόντων αντικειμένων. Τα δεδομένα αυτών των αντικειμένων έχουν αναφορές στα αντικείμενα τα οποία περιέχουν. Οι συμπερασματικές αντικείμενο-σχεσιακές βάσεις δεδομένων παρέχουν όχι μόνο άμεση υποστήριξη για την αποτελεσματική αποθήκευση, αλλά ακόμη πραγματοποιούν τους υπολογισμούς και την εξαγωγή συμπερασμάτων για να ανακτήσουν τα πλήρη δεδομένα των γραφικών αντικειμένων που χρησιμοποιούν άλλα αντικείμενα.

Ένα από τα συστήματα που κάνουν χρήση σχεσιακών μοντέλων στη σχεδίαση με χρήση υπολογιστή είναι το DrawCAD.Το DrawCAD είναι ένα σύστημα CAD υλοποιημένο σε ένα αντικειμενο-σχεσιακό σύστημα βάσης δεδομένων. Διευκολύνει τη δημιουργία γραφικών αντικειμένων επαναχρησιμοποιώντας ήδη υλοποιημένα αντικείμενα. Ένα άλλο σύστημα που χρησιμοποιεί σχεσιακά μοντέλα για εφαρμογές CAD είναι το RelCAD. Η υλοποίηση του συστήματος έχει γίνει με αντικειμενοστραφή λογική. Το σύστημα χρησιμοποιεί την αρχή ότι οι λειτουργικές απαιτήσεις μπορούν να αναπαρασταθούν ως γεωμετρικές/ αριθμητικές σχέσεις μεταξύ των μερών του σχεδίου και των εξισώσεων της μηχανικής.

2.3 Εργαλεία και μέθοδοι υλοποίησης

2.3.1 Λεκτική και συντακτική ανάλυση με Javacc

Η λεκτική και η συντακτική ανάλυση του μεταγλωττιστή επιτελούνται από το εργαλείο Javacc.Το Javacc είναι ένα εργαλείο κατασκευής μεταγλωττιστών γραμμένοι σε γλώσσα Java.Στο εργαλείο javacc ο καθορισμός των λεξημάτων όπως και του συντακτικού γίνονται στο ίδιο αρχείο. Τα αρχεία που αναλύει ο Javacc έχουν κατάληξη .jj και είναι της παρακάτω μορφής:

```
options{
  //Ρύθμιση διαφόρων επιλογών για τον αναλυτή.
}

PARSER_BEGIN(ονομα_κύριας_κλάσσης)

//Κώδικας java :Δήλωση της κύριας κλάσης και της main.Ο κώδικας που βρίσκεται σε
αυτό το κομμάτι αντιγράφεται ατόφιος στο εξαγόμενο αρχείο του αναλυτή(έχει το
όνομα όνομα_κύριας_κλάσσης.java)

PARSER_END(όνομα_κύριας_κλάσσης)

SKIP:
{
```

Περιλαμβάνει τους χαρακτήρες που θα αγνοούνται από τον αναλυτή(διάστημα, χαρακτήρες αλλαγής γραμμής κ.τ.λ)
}

TOKEN :
{
Περιλαμβάνει τη δήλωση των λέξεων-κλειδιών του μεταγλωττιστή
Ακόμη περιλαμβάνει τη δήλωση της μορφής ορισμένων λεξημάτων όπως μεταβλητές οριζόμενες από το χρήστη και αριθμητικές σταθερές.
}

SPECIAL_TOKEN :
{
Δήλωση ειδικών λεξημάτων όπως τα σχόλια ,που αγνοούνται από τον αναλυτή.
}

Δήλωση μη-τερματικών. Τα μη-τερματικά έχουν τη μορφή δηλώσεων μεθόδων της Java με όνομα μη-τερματικού ,τύπο επιστροφής και ορίσματα(για τη σημασιολογική ανάλυση και περαιτέρω λειτουργικότητα).Μέσα στη δήλωση των μη-τερματικών μπορούν να οριστούν μεταβλητές τοπικής εμβέλειας και να γίνει κλήση άλλων μη-τερματικών. Ακόμη με τη χρήση άγκιστρων ({ }) μπορεί να συμπεριληφθεί κώδικας Java.

Τα κύρια χαρακτηριστικά του Javacc είναι :

- Δημιουργεί Top-down μεταγλωττιστές. Δεν επιτρέπει τη χρήση αριστερής αναδρομής αλλά παρέχει τη δυνατότητα περάσματος ορισμάτων κατά τη συντακτική ανάλυση.
- Όπως προαναφέρθηκε ο καθορισμός της λεκτικής όπως και συντακτικής δομής περιλαμβάνονται στο ίδιο αρχείο.
- Παρέχει τη δυνατότητα ρύθμισης διάφορων επιλογών στην αρχή του αρχείου όπως αν θα υπάρχει αντιστοιχία πεζών-κεφαλαίων ή το πλήθος των συμβόλων που θα ελέγχονται για την άρση των αμφισημιών κ.α.
- Παρέχει τη δυνατότητα επίλυσης των αμφισημιών shift-shift τοπικά. Ο Javacc δημιουργεί από προεπιλογή μεταγλωττιστές LL(1).Ωστόσο για τα σημεία της γραμματικής που δεν είναι LL(1) μπορεί να χρησιμοποιηθεί η λέξη κλειδί LOOKAHEAD με τον αριθμό των συμβόλων που επιθυμεί ο προγραμματιστής να εξεταστούν. Με αυτό τον τρόπο η γραμματική γίνεται LL(k) μόνο στα σημεία όπου είναι απαραίτητο κάνοντας την ανάλυση πολύ ταχύτερη.
- Επιτρέπει τη χρήση χαρακτηριστικών του συμβολισμού BNF,όπως για παράδειγμα (A)* ,(A)+ για την αναπαράσταση αναδρομικών εκφράσεων.
- Τέλος παρέχει τη δυνατότητα δήλωσης ειδικών συμβόλων που αγνοούνται από τον αναλυτή σε συγκεκριμένο κομμάτι του αρχείου .jj.

2.3.2 Δέντρα αφηρημένης σύνταξης (Abstract Syntax Trees (AST))

Τα δέντρα αφηρημένης σύνταξης είναι πεπερασμένα, κατευθυνόμενα δέντρα αφηρημένης σύνταξης. Περιγράφουν με δεντρική μορφή τη βασική δομή του κώδικα εισόδου μιας γλώσσας. Τα AST συνήθως αγνοούν συντακτικές λεπτομέρειες οι οποίες δεν επηρεάζουν τη σημασιολογία του προγράμματος. Τέτοιο παράδειγμα αποτελεί η παράλειψη κόμβων για την αναπαράσταση σημείων στίξης αφού αυτά αναπαρίστανται με τη δομή του δέντρου.

Στις περισσότερες περιπτώσεις, κάθε κομμάτι της γραμματικής αναπαρίσταται με ένα καινούργιο κόμβο συγκεκριμένου τύπου. Ωστόσο μπορεί να επιλεγεί κατά το σχεδιασμό του μεταγλωττιστή να χρησιμοποιούνται και ίδιου τύπου κόμβοι με διαφορετικές μεταβλητές για παρόμοιες γραμματικές εκφράσεις

.

2.3.3 Μετατροπές

Μετά τη δημιουργία των AST οι ακόλουθες φάσεις του μεταγλωττιστή μπορούν να υλοποιηθούν. Στις απλούστερες μεταφράσεις θα μπορούσε να διασχιστεί το AST και να δημιουργηθεί η επιδιωκόμενη γλώσσα. Αν η μετάφραση είναι περίπλοκη και περισσότερες αποφάσεις πρέπει να ληφθούν σχετικά με τι ποιες μετατροπές κώδικα θα πρέπει να εφαρμοστούν στα AST και ποιες σε κάποια ενδιάμεση μορφή. Τέτοιες μετατροπές μπορεί να έχουν σκοπό να βελτιώσουν το χρόνο εκτέλεσης του τελικού προγράμματος ή να διευκολύνουν τη μετάφραση στη γλώσσα στόχο.

2.4 Σχετικό λογισμικό

2.4.1 PostgreSQL

Η PostgreSQL είναι ένα αντικείμενο-σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων(ORDBMS).Ένα από τα χαρακτηριστικά των παραδοσιακών σχεσιακών συστημάτων διαχείρισης δεδομένων (RDBMS) που τα διαφοροποιεί από τα σύγχρονα είναι το ότι οι τύποι των χαρακτηριστικών δεν περιορίζονται σε ένα συγκεκριμένο είδος αλλά μπορούν να ένα από ένα μεγάλο εύρος τύπων, όπως ακέραιοι, δεκαδικοί, συμβολοσειρές, νομισματικές μονάδες και ημερομηνίες. Τα χαρακτηριστικά σχεδίασης που παρέχει η PostgreSQL είναι η κληρονομικότητα, οι διάφοροι τύποι δεδομένων και οι συναρτήσεις που επιτρέπουν την επέκταση της. Ακόμη υποστηρίζονται περιορισμοί, εναύσματα, κανόνες και πληρότητα συνδιαλλαγών. Όλα τα παραπάνω χαρακτηριστικά κατατάσσουν την PostgreSQL στην κατηγορία των αντικείμενο-σχεσιακών. Η PostgreSQL είναι εφαρμογή ανοιχτού κώδικα και η υλοποίηση της περιλαμβάνει όλα τα στάδια από τη συντακτική ανάλυση της SQL ως την αποθήκευση δεδομένων.

Τα στάδια που περνάει ένα ερώτημα για να αποδώσει αποτελέσματα στην PostgreSQL είναι 5 και συνοψίζονται στα εξής :

- Σύνδεση της εφαρμογής με τον server.Το πρόγραμμα της εφαρμογής στέλνει το ερώτημα στον server και λαμβάνει τα αποτελέσματα από αυτόν.
- Το στάδιο της συντακτικής ανάλυσης που ελέγχει τα ερωτήματα για σωστή σύνταξη και δημιουργεί το συντακτικό δέντρο.
- Το σύστημα κανόνων (rule system).
- Το στάδιο της οργάνωσης/ βελτιστοποίησης (planner/ optimizer).
- Το στάδιο της εκτέλεσης.

Η φάση της συντακτικής ανάλυσης περιλαμβάνει 2 στάδια. Στο πρώτο γίνεται ο συντακτικός έλεγχος και αν δεν υπάρχει κάποιο λάθος τότε δημιουργείται το συντακτικό δέντρο. Στο δεύτερο στάδιο το συντακτικό δέντρο είναι είσοδος και σαρώνεται για την εύρεση κόμβων που αντιπροσωπεύουν ερωτήματα. Με βάση τους κόμβους που βρέθηκαν δημιουργούνται νέες δομές δεδομένων για κάθε ερώτημα και στη συνέχεια διενεργούνται έλεγχοι κατά πόσο οι πίνακες και οι στήλες που αναφέρονται στα ερωτήματα είναι έγκυρα σε σχέση με τη βάση δεδομένων.

Το στάδιο του συστήματος κανόνων περιλαμβάνει κυρίως το σύστημα επανεγγραφής. Το σύστημα επανεγγραφής λαμβάνει το συντακτικό δέντρο που παρήχθη από το προηγούμενο στάδιο και το οποίο αντιπροσωπεύει κάποιο ερώτημα, και ψάχνει για κανόνες. Αν υπάρχει κάποιος που πρέπει να εφαρμοστεί στο ερώτημα, το συντακτικό δέντρο ανακατασκευάζεται για να υλοποιήσει τις κατάλληλες μετατροπές. Οι μετατροπές στο συντακτικό δέντρο κάθε ερωτήματος το κάνει πιο αποτελεσματικό για την επόμενη φάση.

Σκοπός του σταδίου της οργάνωσης και βελτιστοποίησης είναι η δημιουργία του ιδανικού πλάνου εκτέλεσης. Συνδυάζει όλους τους δυνατούς τρόπους αναζήτησης και ένωσης των σχέσεων που αναφέρονται σε ένα ερώτημα. Όλα τα δημιουργούμενα μονοπάτια οδηγούν στο ίδιο αποτέλεσμα και η δουλειά του βελτιστοποιητή είναι να προσδιορίσει το κόστος του κάθε μονοπατιού και να επιλέξει το βέλτιστο. Έτσι κατασκευάζει το πλάνο ερωτημάτων που είναι ένα δέντρο με τα βέλτιστα μονοπάτια και το οποίο γίνεται είσοδος στον εκτελεστή.

Ο εκτελεστής διασχίζει αναδρομικά το δέντρο πλάνου και ανασύρει πλειάδες. Κάνει χρήση του συστήματος αποθήκευσης καθώς ψάχνει τους πίνακες, εκτελεί ταξινομήσεις και ενώσεις πινάκων, εκτιμά τα κριτήρια επιλογής και τελικά επιστρέφει τις σωστές πλειάδες.

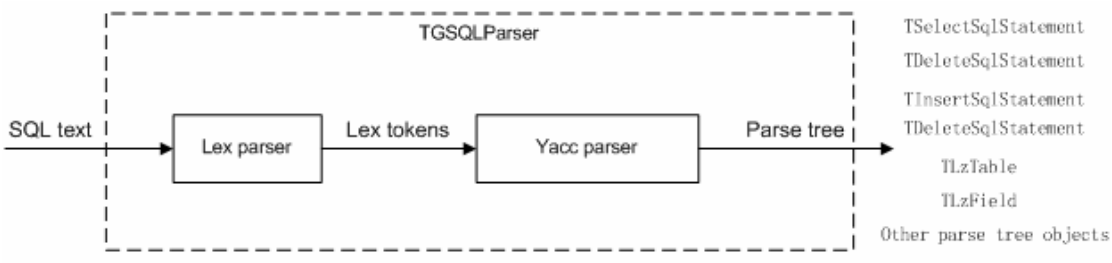
2.4.2 General Sql Parser

Πρόκειται για να ένα εργαλείο παρόμοιο με το παρόν, το οποίο κάνει parsing σε κώδικα SQL και δημιουργεί το κατάλληλο συντακτικό δέντρο. Η σχεδίαση του επιτρέπει τη σύνδεση των εντολών με διάφορες βάσεις δεδομένων όπως οι Oracle,DB2,Informix,Sybase,Postgres και MySQL, ωστόσο οι υλοποιήσεις αυτές δεν παρέχονται.

Ο General SQL Parser λειτουργεί ως εξής. Διαβάζει κώδικα SQL και δημιουργεί σύμβολα με τη λεκτική ανάλυση. Αφού διαβαστεί ολόκληρο το κείμενο SQL τότε δημιουργείται μια λίστα από σύμβολα τα οποία αποτελούν την είσοδο για το συντακτικό αναλυτή. Στη συνέχεια ο συντακτικός αναλυτής διαβάζει τα σύμβολα με βάση τη γραμματική BNF διαφορετικών διαλέκτων βάσεων δεδομένων, και αν δεν υπάρχει σφάλμα τότε δημιουργείται ένα απλό συντακτικό δέντρο. Στη διάρκεια της διαδικασίας αυτής ο χρήστης μπορεί να δημιουργήσει τη δική του μέθοδο χειρισμού λαθών.

Το αρχικό συντακτικό δέντρο στη συνέχεια μετατρέπεται σε μια επίσημη μορφή συντακτικού δέντρου του οποίου οι πρωτεύοντες κόμβοι είναι κόμβοι τύπου που αναπαριστούν εκφράσεις SQL, και περιέχονται και δευτερεύοντες κόμβοι στα

κατώτερα στρώματα του δέντρου. Κατά τη διάρκεια της μετατροπής υπολογίζονται και διαχωρίζονται οι πίνακες και τα πεδία που υπάρχουν στον κώδικα.



Σχήμα 2.2 : Αναπαράσταση του τρόπου λειτουργίας του General SQL Parser

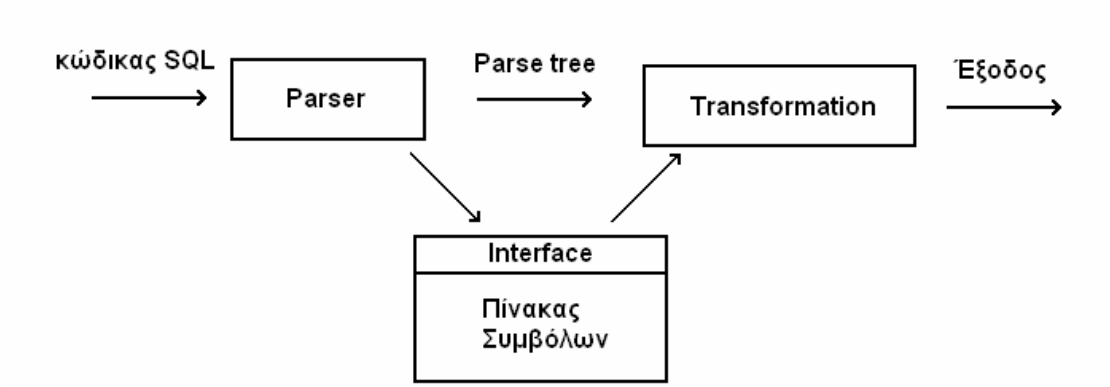
Κεφάλαιο 3

Σχεδίαση

Σε αυτό το κεφάλαιο περιγράφεται ο τρόπος με τον οποίο λειτουργεί ο μεταγλωττιστής σε κάθε στάδιο του και ακόμη με ποιο τρόπο μπορούν να γίνουν οι μετατροπές ώστε να ολοκληρωθεί και να λειτουργήσει πάνω σε βάσεις δεδομένων. Τέλος αναφέρεται το μέγεθος της κάλυψής του σε εντολές της SQL.

3.1 Λειτουργία του SqlParse

Ο μεταγλωττιστής στην τελική του μορφή μπορούμε να θεωρήσουμε ότι αποτελείται από 3 βασικά μέρη. Το πρώτο είναι το κομμάτι του συντακτικού αναλυτή ο οποίος παράγει και το συντακτικό δέντρο. Το δεύτερο είναι το κομμάτι του πίνακα συμβόλων και της συντακτικής ανάλυσης. Σε αυτό υπάρχει η υλοποίηση ενός πίνακα συμβόλων που συνδέεται με τον συντακτικό αναλυτή μέσω μιας διεπαφής(interface). Τέλος υπάρχει το κομμάτι της μετατροπής που χρησιμοποιείται για να μετατρεψει το parse tree σε κατάλληλη μορφή ώστε να λειτουργήσει πάνω σε μια βάση δεδομένων. Τα δύο πρώτα μέρη έχουν υλοποιηθεί. Τα παραπάνω φαίνονται στο σχήμα που ακολουθεί.



Σχήμα 3.1 : Τα μέρη του πλήρους μεταγλωττιστή

3.2 Application Program Interface (API) και υλοποίηση

Ο μεταγλωττιστής έχει σχεδιαστεί με τέτοιο τρόπο ώστε να προσφέρει ένα εύχρηστο API για να μπορεί να γίνει η διασύνδεση του μετατροπέα με το συντακτικό δέντρο. Η σχεδίαση περιλαμβάνει 4 πακέτα, τα 3 εκ των οποίων ανήκουν στο $4^{ο}$.

Το κύριο πακέτο περιλαμβάνει μια κλάση που χρησιμεύει στην εύρεση των τύπων του συντακτικού δέντρου με βάση το όνομα τους.

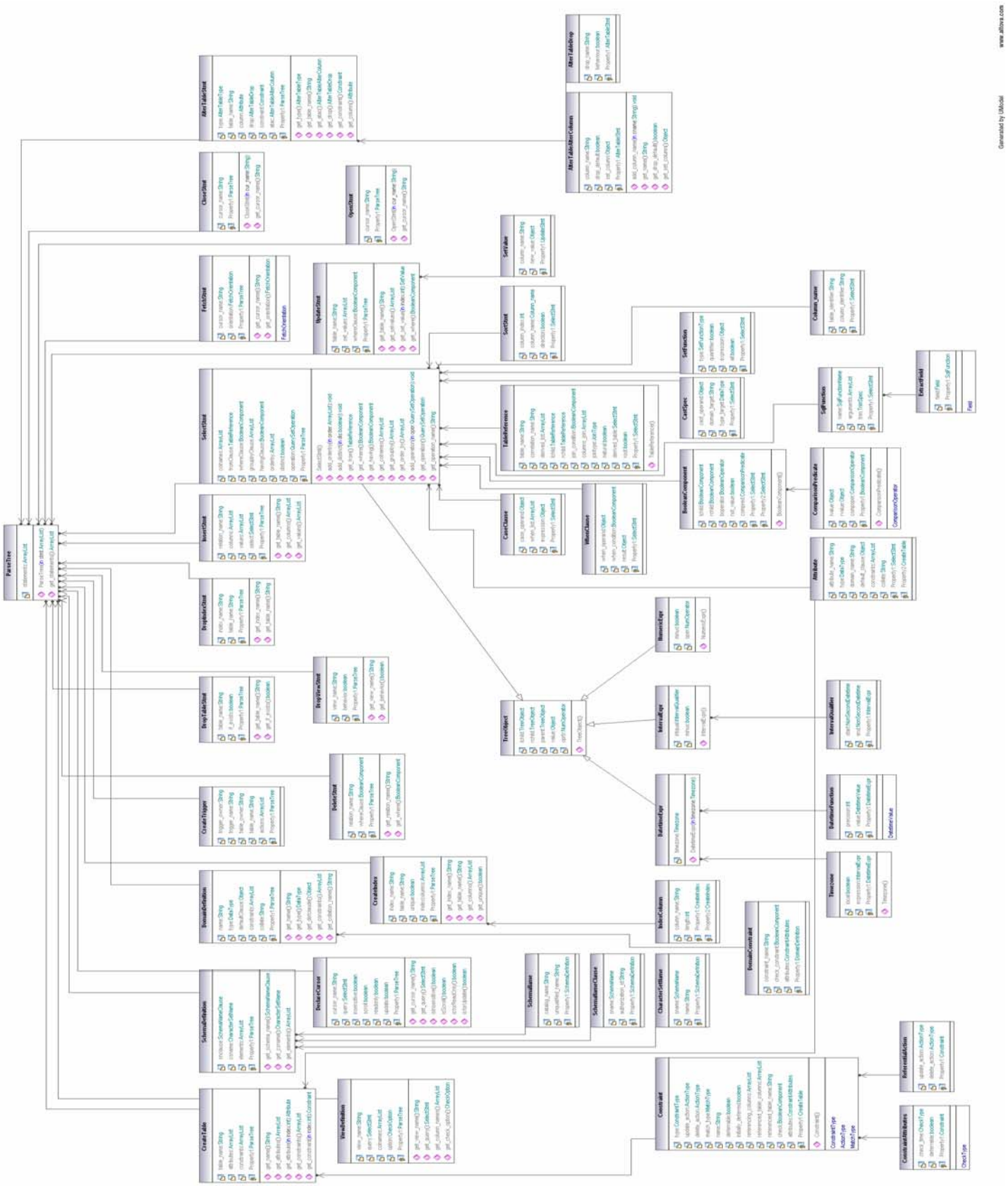
Το πρώτο πακέτο περιέχει όλες τις κλάσεις που εκφράζουν τους τύπους των κόμβων του συντακτικού δέντρου. Οι κλάσεις αυτές περιέχουν κατά κύριο λόγο μία

μέθοδο για κάθε μεταβλητή της κλάσης η οποία επιστρέφει την τιμή της μεταβλητής. Έτσι με τη χρήση του κατάλληλου ονόματος κλάσης και της κατάλληλης μεθόδου ολόκληρα τα δεδομένα του συντακτικού δέντρου μπορούν να χρησιμοποιηθούν με τον τρόπο που επιθυμεί ο προγραμματιστής. Τα δεδομένα αυτά δε μπορούν να τροποποιηθούν μέσω των μεθόδων των κλάσεων στις περισσότερες περιπτώσεις παρά μόνο να διαβαστούν, αφού υπεύθυνος για τη δημιουργία του δέντρου είναι μόνο ο συντακτικός αναλυτής.

Το δεύτερο πακέτο περιέχει τις κλάσεις που διαχειρίζονται τον πίνακα συμβόλων. Περιέχει ακόμη δύο interfaces που για την εισαγωγή και εύρεση στοιχείων από τον πίνακα συμβόλων αντίστοιχα. Η χρήση των interfaces καθιστά δυνατή τη δημιουργία από τον προγραμματιστή ενός άλλου πίνακα συμβόλων αν το επιθυμεί. Οι κλάσεις που περιλαμβάνονται στο πακέτο εκφράζουν τους τύπους των στοιχείων που αποθηκεύονται στον πίνακα συμβόλων, όπως πίνακες, χαρακτηριστικά, εναύσματα, δείκτες κ.α. Αυτές περιλαμβάνουν μεθόδους οι οποίες όπως παραπάνω ανασύρουν τις τιμές των μεταβλητών που αντιστοιχούν στα αποθηκευμένα στοιχεία του πίνακα συμβόλων. Υπάρχει τέλος μια κλάση που διαχειρίζεται τα στοιχεία του πίνακα με μεθόδους εισαγωγής, εύρεσης και διαγραφής.

Το τρίτο πακέτο περιέχει μια κλάση που αφορά στην εκτύπωση των δεδομένων του συντακτικού δέντρου σε σχεσιακή άλγεβρα για να μπορεί να πιστοποιηθεί ότι τα περιεχόμενα του κώδικα SQL μεταφράστηκαν σωστά στο συντακτικό δέντρο.

Παρακάτω φαίνονται σε σχηματικό διάγραμμα UML οι κλάσεις που αποτελούν το αφηρημένο συντακτικό δέντρο καθώς επίσης και οι κλάσεις που αποτελούν την υλοποίηση για τον πίνακα συμβόλων.



Σχήμα 3.2 : Απεικόνιση των κλάσεων του συντακτικού δέντρου σε UML

Σχήμα 3.3 : Απεικόνιση των κλάσεων του πίνακα συμβόλων σε UML.

3.3 Μετατροπές

Το συντακτικό δέντρο που έχει παραχθεί από τη φάση της συντακτικής ανάλυσης περιέχει όλη την πληροφορία για τα περιεχόμενα του κώδικα SQL. Κατόπιν πρέπει να γίνουν οι κατάλληλες ενέργειες ώστε τα περιεχόμενα του να χωριστούν και οι εκφράσεις που περιέχονται να αποκτήσουν πρακτική χρήση.

Το πρώτο βήμα που πρέπει να γίνει, είναι να διαπιστωθεί αν τα περιεχόμενα του κώδικα SQL έχουν σημασιολογική υπόσταση, δηλαδή κατά πόσο οι σχέσεις, οι πίνακες, τα χαρακτηριστικά τους υπάρχουν και συνδέονται μεταξύ τους.

Έτσι το συντακτικό δέντρο διασχίζεται και αποκωδικοποιούνται οι τύποι των κόμβων του πρώτου επιπέδου του που αφορούν τις βασικές εκφράσεις SQL όπως δημιουργία, διαγραφή, μετατροπή στοιχείων(πίνακες, ευρετήρια, εναύσματα κ.τ.λ) καθώς και τα ερωτήματα.

Για τις περιπτώσεις της εισαγωγής και διαγραφής τα στοιχεία με τα χαρακτηριστικά τους προστίθενται και αφαιρούνται αντίστοιχα από τον πίνακα συμβόλων. Ελέγχεται αν τα στοιχεία με το ίδιο όνομα υπάρχουν ήδη κατά την εισαγωγή ή αν δεν υπάρχουν κατά τη διαγραφή.

Για τις περιπτώσεις ερωτημάτων διασχίζεται το δέντρο, αν υπάρχει, με τους κόμβους που αντιπροσωπεύουν εκφράσεις SELECT και για κάθε έναν από αυτούς γίνονται οι απαραίτητοι έλεγχοι(δέντρο με SelectStmts υπάρχει σε περιπτώσεις όπου γίνονται πράξεις συνόλων με ερωτήματα, όπως UNION, INTERSECT και EXCEPT.Στις υπόλοιπες περιπτώσεις μπορούμε να θεωρήσουμε ότι έχουμε ένα δέντρο που αποτελείται μόνο από τη ρίζα).Οι έλεγχοι περιλαμβάνουν πιστοποίηση αν οι πίνακες που αναφέρονται στις προτάσεις FROM υπάρχουν, αν οι στήλες που αναφέρονται στο ερώτημα είναι έγκυρες κ.α.

Σε μελλοντικές υλοποιήσεις μπορεί το στάδιο του σημασιολογικού ελέγχου να κατασκευάζει μετασχηματισμένα συντακτικά δέντρα με περισσότερες πληροφορίες για τα περιεχόμενα των ερωτημάτων και ο πίνακας συμβόλων να συνδέεται με το σύστημα εκτέλεσης εντολών πάνω στη βάση δεδομένων.

3.4 Κάλυψη των εκφράσεων SQL

Σε αυτή την ενότητα παρατίθενται οι εκφράσεις της SQL που καλύπτονται από το συντακτικό αναλυτή.

Καλύπτονται εκφράσεις δημιουργίας πίνακα, όψης, σχήματος, λογικού δρομέα, domain, ευρετηρίου και εναύσματος, δηλαδή οι :

- CREATE TABLE
- CREATE VIEW
- CREATE SCHEMA
- CREATE CURSOR
- CREATE DOMAIN
- CREATE INDEX
- CREATE TRIGGER

Οι πίνακες περιλαμβάνουν δήλωση στηλών με τύπο ή όνομα ενός domain στο οποίο βασίζονται και προαιρετικά περιορισμούς, πρόταση DEFAULT και πρόταση COLLATE.Οι περιορισμοί μπορούν να είναι NOT NULL, μοναδικοί ή πρωτεύοντος κλειδιού, ξένου κλειδιού και check constraints.

Στη δήλωση των όψεων αναφέρεται το όνομα του πίνακα στον οποίο ανήκουν, μια σειρά από στήλες που θα περιλαμβάνει, το ερώτημα που προσδιορίζει την όψη και προαιρετικά οι επιλογές CASCADED ή LOCAL CHECK OPTION.

Στην πρόταση CREATE SCHEMA τα πιθανά αντικείμενα για τα οποία μπορεί να δημιουργηθούν περιγραφείς(descriptors) είναι τα domains και οι όψεις.

Στους λογικούς δρομείς δηλώνεται το όνομα τους, προαιρετικά οι επιλογές INSENSITIVE και SCROLL, η έκφραση πίνακα που συσχετίζεται με αυτόν και προκύπτει από ένα ερώτημα και τέλος προαιρετικά οι επιλογές FOR READ ONLY ή FOR UPDATE.

Στη δήλωση δημιουργίας ενός domain καθορίζεται το όνομα του, ο τύπος τον οποίο αντικαθιστά, και προαιρετικά οι περιορισμοί του, η πρόταση DEFAULT και η πρόταση COLLATE.

Για τα ευρετήρια δηλώνεται το όνομα τους, το όνομα του πίνακα στον οποίο ανήκουν και οι στήλες που θα περιλαμβάνουν. Προαιρετική είναι η επιλογή UNIQUE.

Η έκφραση δημιουργίας εναύσματος (CREATE TRIGGER) ακολουθεί τη σύνταξη του προτύπου SQL/99.

Σε ό,τι αφορά τα ερωτήματα καλύπτονται οι συναρτήσεις συνάθροισης (aggregate functions) COUNT, SUM, AVG, MAX, και MIN.Στην επιλογή πινάκων του ερωτήματος καλύπτεται η συνένωση πινάκων όπως το καρτεσιανό γινόμενο, το NATURAL JOIN, INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN, LEFT OUTER JOIN, INNER OUTER JOIN, FULL OUTER JOIN και UNION JOIN.Ακόμη αντί για υπάρχοντες πίνακες μπορούν να χρησιμοποιηθούν εξαγόμενοι (derived tables) από ένα υποερώτημα. Υποστηρίζεται ακόμη η ταξινόμηση των αποτελεσμάτων του ερωτήματος με αύξουσα ή φθίνουσα σειρά. Τέλος μπορούν να γίνουν και πράξεις συνόλων μεταξύ ερωτημάτων όπως ένωση, τομή, εξαίρεση(UNION, INTERSECT και EXCEPT αντίστοιχα).

Καλύπτονται οι εκφράσεις μετατροπής πίνακα (ALTER TABLE) και συγκεκριμένα προσθήκη στήλης σε πίνακα, μετατροπή στήλης, διαγραφή στήλης, προσθήκη περιορισμού σε πίνακα και διαγραφή περιορισμού σε πίνακα.

Υποστηρίζονται προτάσεις εισαγωγής, ανανέωσης και διαγραφής(INSERT INTO,UPDATE και DELETE FROM) δεδομένων από πίνακες. Με την πρόταση INSERT INTO δίδονται τιμές στις αντίστοιχες στήλες που αναφέρονται στην πρόταση. Με την πρόταση UPDATE μπορούν να δοθούν νέες τιμές σε στήλες. Οι στήλες μπορούν προαιρετικά να επιλεγούν με μια φράση συνθήκης WHERE.Τέλος με την πρόταση διαγραφής διαγράφονται στήλες από ένα πίνακα που επιλέγονται με τη φράση συνθήκης WHERE.

Με την πρόταση DROP μπορούν να διαγραφούν πίνακες, όψεις και ευρετήρια. Για τους πίνακες αρκεί το όνομα τους για να γίνει η διαγραφή. Προαιρετικά μπορεί να συμπληρωθεί η φράση IF EXISTS για έλεγχο. Στις όψεις αναφέρεται το όνομα του πίνακα στον οποίο ανήκουν και η συμπεριφορά διαγραφής (CASCADE, RESTRICT). Για τα ευρετήρια αναφέρεται το όνομα τους και ο πίνακας στον οποίο ανήκουν.

Τέλος υποστηρίζονται οι προτάσεις FETCH, OPEN και CLOSE για εργασία με λογικούς δρομείς. Η πρόταση FETCH περιλαμβάνει το όνομα του δρομέα και προαιρετικά τις επιλογές NEXT, PRIOR, FIRST, LAST, ABSOLUTE και

RELATIVE.Οι προτάσεις OPEN και CLOSE ανοίγουν και κλείνουν το δρομέα αντίστοιχα και σε αυτές αναφέρεται απλά το όνομα του.

Στο παράρτημα αναγράφεται η πλήρης γραμματική του αναλυτή σε μορφή BNF.

Κεφάλαιο 4

Υλοποίηση

Στο κεφάλαιο αυτό γίνεται περιγραφή της υλοποίησης του μεταγλωττιστή. Περιγράφονται αναλυτικότερα μία-μία οι κλάσεις των πακέτων του και γίνεται αναφορά στον τρόπο με τον οποίο έχει δημιουργηθεί η τεκμηρίωση σε μορφή HTML.

Η υλοποίηση του μεταγλωττιστή έγινε με το εργαλείο δημιουργίας μεταγλωττιστών JAVACC και σε γλώσσα προγραμματισμού JAVA.Η σχεδίαση του κώδικα περιλαμβάνει 2 βασικά πακέτα(packages) :

1. Το gr.tuc.softnet.sqlparse.parsetree.Το πακέτο αυτό περιλαμβάνει τις κλάσεις για τη δημιουργία της δομής δεδομένων που περιγράφει το συντακτικό περιεχόμενο του κώδικα SQL.
2. Το gr.tuc.softnet.sqlparse.symboltable.Το πακέτο αυτό περιλαμβάνει τις κλάσεις που υλοποιούν τον πίνακα συμβόλων του μεταγλωττιστή και επιτελούν σημασιολογική ανάλυση.

Υπάρχει ακόμη το πακέτο gr.tuc.softnet.sqlparse που περιέχει την βοηθητική μεταβλητή τύπου enum ClassName που εξυπηρετεί στην τακτοποίηση των ονομάτων των κλάσεων του πακέτου gr.tuc.softnet.sqlparse.parsetree.
Τέλος υπάρχει το πακέτο gr.tuc.softnet.sqlparse.relelgebra που περιέχει την κλάση RelAlgebra.Η κλάση αυτή έχει κάποιες μεθόδους για την προβολή των ερωτημάτων του κώδικα SQL σε μορφή σχεσιακής άλγεβρας.

4.1 Εργασία με τα εργαλεία της Java

4.1.1 Λεκτική και συντακτική ανάλυση

Μετά την ανάλυση ο Javacc παράγει 7 αρχεία java.Αυτά είναι τα εξής:
ParseException.java
Υλοποιεί την κλάση της εξαίρεσης που ρίχνεται όταν υπάρχει ένα συντακτικό λάθος.
***Όνομα_μεταγλωττιστή*.java**
Περιέχει τον κώδικα που υλοποιεί το μεταγλωττιστή
Παράγονται επίσης τα αρχεία:
SimpleCharStream.java
***Όνομα_μεταγλωττιστή*TokenManager.java**
***Όνομα_μεταγλωττιστή*ParseConstants.java**
TokenMgrError.java
Token.java

4.1.2 Η τεκμηρίωση σε μορφή HTML

Η τεκμηρίωση για το μεταγλωττιστή έγινε με το εργαλείο Javadoc της Java.Με το εργαλείο αυτό καθίσταται δυνατό στον προγραμματιστή να γράψει με τη μορφή ειδικών σχολίων την επεξήγηση του κώδικα στα αρχεία .java και να εξαγάγει την τεκμηρίωση σε μορφή HTML.

Ειδικότερα, με τη χρήση των ειδικών σχολίων /** *σχολια* */ πριν από κάθε κλάση , constructor ή μέθοδο κατασκευάζεται η τεκμηρίωση για κάθε ένα από τα παραπάνω. Το περιεχόμενο των ειδικών σχολίων μπορεί να περιλαμβάνει κείμενο και κώδικα html για την κατάλληλη μορφοποίηση των εξαγόμενων. Το javadoc παρέχει και επιπρόσθετα στοιχεία για κατάλληλη μορφοποίηση ορισμάτων και επιστρεφόμενων στοιχείων των μεθόδων της κλάσης, δημιουργία συνδέσμου για τις παραπομπές της κ.α.

4.2 Πακέτα, κλάσεις και τύποι

Παρακάτω παρουσιάζονται τα πακέτα και οι κλάσεις του κώδικα Java που χρησιμοποιούνται για την υλοποίηση του μεταγλωττιστή.

4.2.1 Πακέτο gr.tuc.softnet.sqlparse.parsetree

Το πακέτο gr.tuc.softnet.sqlparse.parsetree περιέχει τις κλάσεις που χρησιμοποιούνται για τη δημιουργία της δομής δεδομένων που περιγράφει το συντακτικό δέντρο του κώδικα SQL.

Η δημιουργία της δομής δεδομένων γίνεται κατά τη φάση της συντακτικής ανάλυσης. Τα μη-τερματικά του υλοποιημένου μεταγλωττιστή αντιπροσωπεύουν δηλώσεις ή κομμάτια δηλώσεων του κώδικα SQL.Γι΄ αυτό το λόγο τα περισσότερα από αυτά έχουν ως τύπο επιστροφής μία από τις κλάσεις του πακέτου gr.tuc.softnet.sqlparse.parsetree.Όταν συναντάται μια δήλωση κατασκευάζεται ένα νέο αντικείμενο της αντίστοιχης κλάσης με τον τελεστή **new** της Java και προσδίδονται στις μεταβλητές της οι τιμές που αντιστοιχούν στη δήλωση.

Οι κλάσεις και οι enumerated types του πακέτου φαίνονται στον παρακάτω πίνακα και είναι χωρισμένες σε 2 κατηγορίες : στις κλάσεις περιγραφής δεδομένων και στις κλάσεις διαχείρισης δεδομένων. Οι κλάσεις της πρώτης κατηγορίας υλοποιούν εκφράσεις της SQL που δηλώνουν και διαχειρίζονται στοιχεία για μια βάση δεδομένων όπως πίνακες, όψεις, ευρετήρια κ.α. και περιγράφουν τα επιμέρους στοιχεία από τα οποία αποτελούνται. Οι κλάσεις της δεύτερης κατηγορίας υλοποιούν εκφράσεις της SQL που διαχειρίζονται τα δεδομένα μιας βάσης. όπως τα ερωτήματα ή προτάσεις διαγραφής, ανανέωσης και προσθήκης δεδομένων.

Κλάση που περιγράφει ολόκληρο το συντακτικό δέντρο
ParseTree
Κλάσεις περιγραφής δεδομένων

Όνομα κλάσης	**Σχετιζόμενες κλάσεις**	**Σχετιζόμενοι enumerated types**
CreateTable	Attribute	DataType
	Constraint	Constraint.ActionType
		Constraint.ConstraintType
		Constraint.MatchType
	ConstraintAttributes	ConstraintAttributes. CheckType
	ReferentialAction	
CreateTrigger	TriggeredAction	TriggeredAction. TriggeredActionOption
	TriggerEvent	TriggerEvent. TriggerEventOption
	TriggerAlias	TriggerAlias. TriggerAliasValue
SchemaDefinition	SchemaNameClause	
	CharacterSetName	
CreateIndex	IndexColumn	
DeclareCursor		
ViewDefinition		CheckOption
DomainDefinition	DomainConstraint	
FetchStmt		FetchStmt.FetchOrientation
OpenStmt		
CloseStmt		
AlterTableStmt	AlterTableAlterColumn	AlterTableType
	AlterTableDrop	
DropIndexStmt		
DropTableStmt		
DropViewStmt		

Κλάσεις διαχείρισης δεδομένων

Όνομα κλάσης	**Σχετιζόμενες κλάσεις**	**Σχετιζόμενοι enumerated types**
SelectStmt	TreeObject	QuerySetOperation
	TableReference	JoinType
	WhereClause	
	BooleanComponent	BooleanOperator
	SortStmt	
	SqlFunction	SqlFunctionName
		TrimSpec
	ComparisonPredicate	ComparisonPredicate. ComparisonOperator
	DatetimeExpr	GeneralValueSpec

	IntervalExpr	NonSecondDatetime
		GeneralValueSpec
	NumericExpr	NumOperator
	DatetimeFunction	DatetimeFunction. DatetimeValue
	Timezone	
	IntervalQualifier	
	SetFunction	SetFunctionType
	ExtractField	ExtractField.Field
	CastSpec	
	CaseClause	
	Column_name	
	StringExpr	
InsertStmt		
DeleteStmt		
UpdateStmt	SetValue	

Πίνακας 4.1 Οι κλάσεις και οι τύποι του πακέτου gr.tuc.softnet.sqlparse. parsetree. *Ο πίνακας είναι χωρισμένος σε 2 κατηγορίες κλάσεων: Στις κλάσεις περιγραφής δεδομένων και στις κλάσεις διαχείρισης δεδομένων. Στην πρώτη στήλη του πίνακα τοποθετούνται οι κλάσεις που υλοποιούν μια έκφραση της SQL και βρίσκονται στο πρώτο επίπεδο του συντακτικού δέντρου. Στη δεύτερη στήλη τοποθετούνται κλάσεις που περιγράφουν επιπρόσθετα στοιχεία για μία από τις κλάσεις του πρώτου επιπέδου. Τέλος στην τρίτη στήλη τοποθετούνται οι enumerated types που χρησιμοποιούνται από την κλάση της αντίστοιχης γραμμής της δεύτερης ή πρώτης στήλης.*

Για κάθε αρχείο κώδικα SQL που αναλύεται, ο μεταγλωττιστής παράγει ένα αντικείμενο της κλάσης Parsetree.Τα αντικείμενα αυτής της κλάσης περιέχουν μια ArrayList(λίστα της γλώσσας java με αυτόματη παραχώρηση μνήμης και πρόσβαση στα μέλη της με ευρετήριο, όπως ο vector στη C++), η οποία αποθηκεύει αντικείμενα των κλάσεων της πρώτης στήλης, όπως φαίνεται στον πίνακα 4.1. Οι κλάσεις της κατηγορίας αυτής αντιπροσωπεύουν δηλώσεις δημιουργίας και διαγραφής πινάκων, εναυσμάτων, ευρετηρίων, δρομέων κ.τ.λ καθώς και ερωτήματα, όλες δηλαδή τις κύριες εκφράσεις της SQL που υποστηρίζονται. Κάθε αντικείμενο των παραπάνω κλάσεων μπορεί να περιέχει αντικείμενα συγκεκριμένων κλάσεων της δεύτερης στήλης του πίνακα, τα οποία χρησιμοποιούνται για να περιγραφούν επιπρόσθετα χαρακτηριστικά μιας έκφρασης.

ΠΕΡΙΓΡΑΦΗ ΤΩΝ ΚΛΑΣΕΩΝ ΚΑΙ ΤΩΝ ΤΥΠΩΝ ΤΟΥ ΠΑΚΕΤΟΥ

Με έντονο πλαίσιο παρατίθενται οι κλάσεις που ανήκουν στην πρώτη στήλη του πίνακα 4.1.Με το απλό πλαίσιο είναι οι κλάσεις που ανήκουν στη δεύτερη στήλη και με διακεκομμένο πλαίσιο είναι οι enumerated types και περιγράφονται κάτω από την κλάση της πρώτης στήλης με την οποία σχετίζονται.

Class ParseTree

Η ParseTree κρατά όλη τη δομή δεδομένων που περιγράφει το συντακτικό δέντρο του κώδικα SQL.Η κλάση αυτή περιέχει μια λίστα η οποία μετά την ανάλυση περιέχει αντικείμενα των κλάσεων που αφορούν τις κυριότερες δηλώσεις της SQL και τα ερωτήματα.

1. Περιγραφή των κλάσεων περιγραφής δεδομένων

Class CreateTable

Η κλάση αυτή χρησιμοποιείται όταν υπάρχει δήλωση CREATE TABLE στον κώδικα SQL.Η κλάση αποθηκεύει το όνομα του πίνακα ο οποίος δηλώνεται και σε 2 χωριστές λίστες τα στοιχεία και τους περιορισμούς του.

Class Attribute

Χρησιμοποιείται όταν υπάρχει η δήλωση μιας στήλης σε μια εντολή CREATE TABLE και διατηρεί πληροφορίες για το όνομα, τον τύπο, τους περιορισμούς της κ.α.

Enum DataType

Χρησιμοποιείται για να προσδιορίσει τον τύπο μιας στήλης ενός πίνακα ή ενός domain.Οι διαθέσιμοι τύποι στην SQL είναι:

- *BIT*
- *BIT_VARYING*
- *CHAR*
- *CHAR_VARYING*
- *CHARACTER*
- *CHARACTER_VARYING*
- *DATE*
- *DEC*
- *DECIMAL*
- *DOUBLE_PRECISION*
- *FLOAT*
- *INT*
- *INTEGER*
- *INTERVAL*
- *NATIONAL_CHAR*
- *NATIONAL_CHAR_VARYING*
- *NATIONAL_CHARACTER*
- *NATIONAL_CHARACTER_VARYING*
- *NCHAR*
- *NCHAR_VARYING*

- *NUMERIC*
- *REAL*
- *SMALLINT*
- *TIME*
- *TIMESTAMP*
- *VARCHAR*

Class Constraint

Τα αντικείμενα της κλάσης Constraint κατασκευάζονται σε κάθε δήλωση ενός περιορισμού στον κώδικα SQL, είτε αυτός είναι περιορισμός πίνακα είτε είναι περιορισμός στήλης. Η κλάση περιέχει πολλές μεταβλητές τύπου enum που περιγράφουν τα είδη και τις επιλογές ενός περιορισμού καθώς και πολλές άλλες μεταβλητές διαφόρων τύπων που χρειάζονται για κάθε είδος περιορισμού.

Enum Constraint.ActionType

Ο τύπος αυτός είναι μέρος της κλάσης Constraint και χρησιμοποιείται για να προσδιορίσει την ενέργεια που γίνεται κατά την ανανέωση ή διαγραφή ενός αναφορικού περιορισμού. Οι τιμές που παίρνει είναι:

- *CASCADE*
- *NO_ACTION*
- *NULL*
- *SET_DEFAULT*
- *SET_NULL*

Enum Constraint.ConstraintType

Ο τύπος αυτός ανήκει επίσης στην κλάση Constraint και προσδιορίζει τον τύπο ενός περιορισμού. Οι τιμές που παίρνει είναι:

- *CHECK* για περιορισμούς ελέγχου,
- *FOREIGN KEY* για περιορισμούς ξένου κλειδιού,
- *NOT NULL* για περιορισμούς μη μηδενικής τιμής
- *PRIMARY KEY* για περιορισμούς πρωτεύοντος κλειδιού και
- *UNIQUE* για μοναδικούς περιορισμούς.

Enum Constraint.MatchType

Χρησιμοποιείται επίσης από τους περιορισμούς για να εκφράσει τον τύπο ταιριάσματος ενός αναφορικού περιορισμού. Οι τιμές που παίρνει είναι:

- *FULL*
- *NULL*
- *PARTIAL*

Enum ConstraintAttributes.CheckType

Χρησιμοποιείται για να εκφράσει το χρόνο ελέγχου ενός περιορισμού. Οι τιμές που παίρνει είναι:

- *INITIALLY_DEFERRED*
- *INITIALLY_IMMEDIATE*
- *NULL* αν δεν είναι κανένα από τα παραπάνω.

Class ConstraintAttributes

Η κλάση αυτή χρησιμοποιείται για να περιγράψει κάποια χαρακτηριστικά των περιορισμών, όπως αν είναι INITIALLY DEFERRED ή INITIALLY IMMEDIATE.

Class ReferentialAction

Βοηθάει κατά τη δήλωση ενός αναφορικού περιορισμού για να εκφράσει την ενέργεια που εκτελείται σε μια ανανέωση ή σε διαγραφή μιας στήλης.

Class CreateTrigger

Η κλάση αυτή χρησιμοποιείται όταν υπάρχει δήλωση CREATE TRIGGER στον κώδικα SQL.Αποθηκεύει το όνομα του εναύσματος, το όνομα του πίνακα στον οποίο ανήκει και άλλα στοιχεία που προσδιορίζουν τα χαρακτηριστικά του εναύσματος. Οι κλάσεις και οι enumerated types που ακολουθούν περιγράφουν αυτά τα χαρακτηριστικά.

Class TriggerAlias

Η κλάση αυτή χρησιμοποιείται για να περιγράψει τις μεταβλητές που ακολουθούν τη λέξη κλειδί REFERENCING σε μία δήλωση εναύσματος. Περιέχει μια μεταβλητή του τύπου TriggerAliasValue και μια μεταβλητή συμβολοσειράς που κρατά το όνομα της μεταβλητής ή το αναγνωριστικό ενός πίνακα.

Enum TriggerAlias.TriggerAliasValue

Ανήκει στην κλάση TriggerAlias και περιέχει τις παρακάτω τιμές :

- *NEW_ROW* .Αυτή η τιμή χρησιμοποιείται για να δηλωθεί μια μεταβλητή η οποία θα διατηρεί τα δεδομένα μιας στήλης που εμπλέκεται στο έναυσμα, μετά τη δράση του.
- *NEW_TABLE* .Αυτή η τιμή χρησιμοποιείται για να δηλωθεί το αναγνωριστικό ενός πίνακα ο οποίος θα διατηρεί τα δεδομένα του πίνακα του εναύσματος, μετά τη δράση του.
- *OLD_ROW* . Αυτή η τιμή χρησιμοποιείται για να δηλωθεί μια μεταβλητή η οποία θα διατηρεί τα δεδομένα μιας στήλης που εμπλέκεται στο έναυσμα, πριν τη δράση του.
- *OLD_TABLE* .Αυτή η τιμή χρησιμοποιείται για να δηλωθεί το αναγνωριστικό ενός πίνακα ο οποίος θα διατηρεί τα δεδομένα του πίνακα του εναύσματος, πριν τη δράση του.

Class TriggeredAction

Χρησιμοποιείται για να περιγράψει την ενέργεια που εκτελεί ένα έναυσμα. Έχει μια μεταβλητή τύπου TriggeredActionOption που περιγράφει πόσες φορές εκτελείται η ενέργεια και μια μεταβλητή BooleanComponent που περιγράφει τη συνθήκη εκτέλεσης.

Enum TriggeredAction.TriggeredActionOption

Ανήκει στην κλάση TriggerAction και περιέχει τις διαφορετικές περιπτώσεις σχετικά με το πόσες φορές εκτελείται το έναυσμα. Οι τιμές του είναι:

- *FOR_EACH_ROW* εκτελείται για κάθε στήλη.
- *FOR_EACH_STATEMENT* εκτελείται για κάθε δήλωση.
- *NULL* Κανένα από τα παραπάνω.

Class TriggerEvent

Χρησιμοποιείται για να προσδιορίσει αν το έναυσμα λειτουργεί κατά την εισαγωγή, διαγραφή ή ενημέρωση. Για την τελευταία περίπτωση διατηρεί και μια λίστα από τις στήλες του πίνακα που επηρεάζονται.

Enum TriggerEvent.TriggerEventOption

Ανήκει στην κλάση TriggerEvent.Περιγράφει τις τρεις περιπτώσεις ενέργειας ενός εναύσματος. Αυτές είναι:

- *INSERT*
- *UPDATE*
- *DELETE*

Class SchemaDefinition

Χρησιμοποιείται όταν υπάρχει δήλωση CREATE SCHEMA διατηρεί στοιχεία για το SCHEMA που δηλώνεται.

Class SchemaNameClause

Χρησιμοποιείται για να διατηρεί στοιχεία για ένα SCHEMA.

Class CharacterSetName

Χρησιμοποιείται όταν υπάρχει μια δήλωση CREATE SCHEMA και κρατάει πληροφορίες για το όνομα του SCHEMA και του character set.

Class CreateIndex

Αντικείμενα της κλάσης αυτής δημιουργούνται όταν υπάρχει μια δήλωση CREATE INDEX στην SQL.Οι μεταβλητές της αποθηκεύουν το όνομα του ευρετηρίου καθώς και το όνομα του πίνακα στον οποίο αναφέρεται. Ακόμα μια δυαδική μεταβλητή προσδιορίζει αν το ευρετήριο είναι UNIQUE ή όχι.

Class IndexColumn

Η κλάση IndexColumn αντιπροσωπεύει μια στήλη ενός πίνακα στην οποία ενεργεί ένα ευρετήριο. Δημιουργείται ένα αντικείμενο για κάθε στήλη που περιέχεται στη δήλωση CREATE INDEX του κώδικα SQL και διατηρεί το όνομα της στήλης και το μήκος (αν χρειάζεται, ανάλογα με τον τύπο της στήλης).

Class DeclareCursor

Χρησιμοποιείται όταν υπάρχει η δήλωση ενός λογικού δρομέα στην SQL.Η κλάση έχει μια μεταβλητή που διατηρεί το όνομα του λογικού δρομέα που δηλώνεται, μια μεταβλητή τύπου SelectStmt που περιγράφει το ερώτημα με βάση το οποίο δημιουργείται ο δρομέας και μερικές δυαδικές μεταβλητές που διατηρούν τις επιλογές για αυτόν.

Class ViewDefinition

Χρησιμοποιείται όταν υπάρχει μια δήλωση Όψης(CREATE VIEW) στην SQL.Διατηρεί πληροφορίες για το όνομα της όψης, τη λίστα με τις στήλες που περιλαμβάνει και μια μεταβλητή SelectStmt που αναφέρεται στο ερώτημα βάσει του οποίου γίνεται η δημιουργία της όψης.

Enum CheckOption

Χρησιμοποιείται για να προσδιορίσει τις επιλογές ελέγχου μιας όψης. Οι επιλογέα αυτές είναι :

- *CHECK_OPTION*
- *CASCADED_CHECK_OPTION* και
- *LOCAL_CHECK_OPTION*.

Class DomainDefinition

Αντικείμενα της κλάσης αυτής δημιουργούνται όταν υπάρχει μια δήλωση ενός domain (CREATE DOMAIN) στην SQL.Οι μεταβλητές της κλάσης κρατούν το όνομα, τον τύπο, τις επιλογές, μια λίστα με τους περιορισμούς και το collation name του domain.

Class DomainConstraint

Περιγράφει τους περιορισμούς των domains.Περιέχει μεταβλητές για τη διατήρηση του ονόματος του domain, την περιγραφή της συνθήκης ελέγχου του περιορισμού και τα χαρακτηριστικά του.

Class FetchStmt

Χρησιμοποιείται όταν υπάρχει μια έκφραση FETCH στον κώδικα SQL.Οι τιμές που διατηρεί είναι το όνομα του δρομέα που γίνεται FETCH καθώς και ο προσανατολισμός(fetch orientation).

Enum FetchStmt.FetchOrientation

Χρησιμοποιείται για να περιγράψει τον προσανατολισμό μιας δήλωσης fetch ενός λογικού δρομέα. Οι τιμές που παίρνει είναι :

- *ABSOLUTE*
- *FIRST*
- *LAST*
- *NEXT*
- *PRIOR*

• *RELATIVE*

Class OpenStmt

Η κλάση αυτή χρησιμοποιείται όταν υπάρχει μια δήλωση OPEN cursor_name στην SQL.Περιέχει μια μεταβλητή που αποθηκεύει το όνομα του δρομέα ο οποίος ανοίγεται.

Class CloseStmt

Αντικείμενα της κλάσης αυτής δημιουργούνται όταν υπάρχει μία δήλωση CLOSE cursor_name και διατηρεί μια μεταβλητή που αποθηκεύει το όνομα του δρομέα.

Class AlterTableStmt

Είναι η γενική κλάση της δήλωσης ALTER TABLE και περιέχει μεταβλητές για την αποθήκευση του υπό μεταβολή πίνακα καθώς και μεταβλητές των 2 παραπάνω τύπων,AlterTableAlterColumn και AlterTableDrop , ανάλογα με το είδος της δήλωσης.

Class AlterTableAlterColumn

Αντικείμενα της κλάσης αυτής δημιουργούνται όταν υπάρχει μια δήλωση ALTER TABLE table_name ALTER COLUMN column_name στον κώδικα SQL.Η κλάση περιέχει μεταβλητές για την αποθήκευση του ονόματος στήλης που μεταβάλλεται καθώς και κάποιες άλλες για την αποθήκευση των επιλογών της μεταβολής.

Class AlterTableDrop

Χρησιμοποιείται όταν υπάρχει δήλωση της μορφής
ALTER TABLE table_name DROP COLUMN ή
ALTER TABLE table_name DROP CONSTRAINT.

Enum AlterTableType

Χρησιμοποιείται για να προσδιορίσει τον τύπο της τροποποίησης πίνακα που εκτελεί μια δήλωση ALTER TABLE.Οι τιμές που έχει είναι :

- *ADD_COLUMN* για προσθήκη στήλης,
- *ADD_CONSTRAINT* για προσθήκη περιορισμού,
- *ALTER_COLUMN* για τροποποίηση στήλης,
- *DROP_COLUMN* για διαγραφή στήλης και
- *DROP_CONSTRAINT* για διαγραφή περιορισμού.

Class DropIndexStmt

Χρησιμοποιείται όταν υπάρχει μια δήλωση διαγραφής ενός ευρετηρίου (DROP INDEX) στον κώδικα SQL.Διατηρούνται τα ονόματα του ευρετηρίου καθώς και του πίνακα στον οποίο ανήκει.

Class DropTableStmt

Χρησιμοποιείται όταν υπάρχει δήλωση διαγραφής ενός πίνακα (DROP TABLE) στον κώδικα SQL.Η κλάση έχει μια μεταβλητή που διατηρεί το όνομα του υπό διαγραφή πίνακα καθώς και μια δυαδική μεταβλητή η οποία υποδηλώνει αν στη δήλωση διαγραφής περιλαμβάνεται η συνθήκη IF EXISTS.

Class DropViewStmt

Χρησιμοποιείται όταν υπάρχει δήλωση διαγραφής μιας όψης (DROP VIEW) στον κώδικα SQL.Η κλάση έχει μια μεταβλητή που διατηρεί το όνομα της υπό διαγραφή όψης καθώς και μια δυαδική μεταβλητή η οποία καθορίζει αν η συμπεριφορά διαγραφής είναι CASCADE ή RESTRICT.

2. Περιγραφή των κλάσεων διαχείρισης δεδομένων

Class SelectStmt

Αντικείμενα της κλάσης αυτής δημιουργούνται κάθε φορά που υπάρχει ερώτημα στον κώδικα SQL είτε μόνο του είτε ως ένθετο σε άλλες εκφράσεις. Διατηρεί πληροφορίες για τις στήλες που θα προβληθούν, τους πίνακες από τους οποίους γίνεται η επιλογή, τα κριτήρια επιλογής και ομαδοποίησης. Όταν υπάρχει ένωση ή τομή ερωτημάτων, για κάθε ερώτημα δημιουργείται ένα αντικείμενο τύπου SelectStmt και όλα μαζί σχηματίζουν ένα δυαδικό δέντρο που διατηρεί την προτεραιότητα των πράξεων συνόλων. Γι' αυτό το λόγο η κλάση αυτή είναι υποκλάση της TreeObject.

Class TableReference

Χρησιμοποιείται για να περιγράψει τις αναφορές σε πίνακες ενός ερωτήματος SQL.Για κάθε πίνακα που χρησιμοποιεί ένα ερώτημα καθώς και για κάθε ένωση μεταξύ των πινάκων δημιουργείται ένα αντικείμενο τύπου TableReference και όλα σχηματίζουν ένα δυαδικό δέντρο. Οι κόμβοι που δεν είναι φύλλα περιέχουν αναφορές για το αριστερό και δεξί υποδέντρο, πληροφορίες για τον τύπο της ένωσης και τη συνθήκη ένωσης. Οι κόμβοι φύλλα κρατούν πληροφορίες για το όνομα του πίνακα ή το όνομα με το οποίο αναφέρεται ή ένα δείκτη σε ένα αντικείμενο τύπου SelectStmt αν χρησιμοποιείται ένα ερώτημα αντί για πίνακα.

Enum JoinType

Κρατά τις τιμές που περιγράφουν τους πιθανούς τρόπους ένωσης πινάκων. Οι τιμές του είναι :

- *CROSS* για cross join

- *FULL* για full join
- *FULL_OUTER* για full outer join
- *INNER* για inner join
- *LEFT* για left join
- *LEFT_OUTER* για left outer join
- *RIGHT* για right
- *RIGHT_OUTER* για right outer
- *UNION* για union

Class TreeObject

Η κλάση αυτή υλοποιεί κόμβους δυαδικών δέντρων. Τα δυαδικά δέντρα όπως αναφέρθηκε παραπάνω χρειάζονται για την αναπαράσταση κάποιων πολύπλοκών εκφράσεων στις οποίες χρειάζεται να διατηρηθεί η προτεραιότητα των τελεστών, όπως οι αριθμητικές και οι λογικές πράξεις. Τα επιμέρους κομμάτια των εκφράσεων αναπαρίστανται από τους κόμβους φύλλα, ενώ τα συνδετικά μέρη των εκφράσεων, όπως οι τελεστές, αναπαρίστανται από τους εσωτερικούς κόμβους. Οι κόμβοι με μεγαλύτερο βάθος έχουν και τη μεγαλύτερη προτεραιότητα.

.

Enum QuerySetOperation

Περιέχει τις τιμές που περιγράφουν τις πράξεις μεταξύ συνόλων. Χρησιμοποιείται για ένωση, τομή ή εξαίρεση ερωτημάτων. Οι τιμές είναι:

- *EXCEPT* για εξαίρεση,
- *INTERSECT* για τομή και
- *UNION* για ένωση.

Class WhereClause

Χρησιμοποιείται όταν υπάρχει μια έκφραση WHEN-THEN στην SQL.Διατηρεί πληροφορίες για τις εκφράσεις οι οποίες αναφέρονται καθώς και τη συνθήκη αν υπάρχει.

Class BooleanComponent

Αντικείμενα της κλάσης αυτής δημιουργούνται όταν υπάρχει συνθήκη αναζήτησης σε ένα ερώτημα ή άλλη έκφραση της SQL.Τα αντικείμενα της κλάσης αυτής σχηματίζουν ένα δέντρο με τέτοιο τρόπο ώστε να διατηρείται η προτεραιότητα των τελεστών της συνθήκης. Στους κόμβους του δέντρου που δεν είναι φύλλα αποθηκεύονται δείκτες στους κόμβους-παιδιά καθώς και το είδος του δυαδικού τελεστή(AND/OR) που συνδέει τα 2 υποδέντρα, ενώ στους κόμβους-φύλλα αποθηκεύονται πληροφορίες για την πρωτεύουσα συνθήκη.

Enum BooleanOperator

Προσδιορίζει τον τύπο μιας έκφρασης Boolean.Οι τιμές που έχει είναι AND και OR για τις αντίστοιχες λογικές πράξεις.

Class SortStmt

Χρησιμοποιείται για να περιγράψει τα χαρακτηριστικά της ταξινόμησης σε ένα ερώτημα. Πιο συγκεκριμένα δημιουργείται όταν υπάρχει η λέξη κλειδί

ORDER BY σε ένα ερώτημα και διατηρεί το όνομα της στήλης καθώς και τη φορά ταξινόμησης(αύξουσα, φθίνουσα).
Χρησιμοποιεί μια μεταβλητή για την αποθήκευση του ονόματος της στήλης ή του περιορισμού που διαγράφεται και μια μεταβλητή για την επιλογή διαγραφής (CASCADE ή RESTRICT).

Class ComparisonPredicate

Χρησιμοποιείται για να περιγράψει μια έκφραση σύγκρισης στην SQL.Περιέχει μια μεταβλητή τύπου enum που απαριθμεί τους διαφορετικούς τελεστές σύγκρισης καθώς και δύο μεταβλητές που αποθηκεύουν το αριστερό και δεξιό μέλος της σύγκρισης.

Enum ComparisonPredicate.ComparisonOperator

Ο τύπος αυτός είναι μέρος της κλάσης ComparisonPredicate και χρησιμοποιείται για να προσδιορίσει τον τελεστή σύγκρισης μιας έκφρασης σύγκρισης. Οι τιμές που παίρνει είναι:

- *EQUAL* για τον τελεστή ισότητας (=),
- *GREATER* για τον τελεστή μεγαλύτερο από(>),
- *GREATEREQ* για τον τελεστή μεγαλύτερο ή ίσο (>=),
- *LESS* για τον τελεστή μικρότερο από (<),
- *LESSEQ* για τον τελεστή μικρότερο ή ίσο (<=),
- *LIKE* για τη λέξη κλειδί LIKE της SQL,
- *NOTEQUAL* για τον τελεστή διαφορετικό από (<>),
- *NOTLIKE* για τη λέξη κλειδί NOT LIKE της SQL.

Class SqlFunction

Χρησιμοποιείται όταν υπάρχει κάποια από τις προκαθορισμένες συναρτήσεις της SQL.Αυτές είναι :POSITION, EXTRACT, CHAR_LENGTH/CHARACTER_LENGTH, SUBSTRING, UPPER/LOWER, CONVERT, TRANSLATE, TRIM, NULLIF/ COALESCE. Κρατά στοιχεία για το όνομα της συνάρτησης και τα ορίσματά της.

Enum SqlFunctionName

Περιέχει τα ονόματα των προκαθορισμένων συναρτήσεων της SQL.Αυτές είναι:

- *CHAR_LENGTH*
- *COALESCE*
- *CONVERT*
- *EXTRACT*
- *LOWER*
- *NULLIF*
- *POSITION*
- *SUBSTRING*
- *TRANSLATE*
- *TRIM*
- *UPPER*

Enum TrimSpec

Περιέχει κάποιες πρόσθετες επιλογές για τη συνάρτηση TRIM της SQL.Αυτές είναι:

- *LEADING*
- *TRAILING*
- *BOTH*

Class DatetimeExpr

Χρησιμοποιείται για να περιγράψει εκφράσεις που περιέχουν ημερομηνία-ώρα στην SQL.Οι εκφράσεις αυτές μπορεί να περιλαμβάνουν αριθμητικές πράξεις μεταξύ αριθμητικών εκφράσεων(class NumericExpr),εκφράσεων ημερομηνίας-ώρας (class DatetimeExpr) και εκφράσεων χρονικών διαστημάτων (class IntervalExpr) της SQL.Για το λόγο αυτό και οι τρεις παραπάνω κατηγορίες περιγράφονται από κλάσεις που κληρονομούν την TreeObject και περιγράφονται από ένα δέντρο που αποτελείται από κόμβους του τύπου TreeObject.

Enum GeneralValueSpec

Κρατά κάποιες σταθερές τιμές που χρησιμοποιούνται στην SQL.Αυτές είναι :

- *CURRENT_USER*
- *SESSION_USER*
- *SYSTEM_USER*
- *USER*
- *VALUE*

Class DatetimeFunction

Περιέχει τον enum τύπο DatetimeValue που περιγράφει κάποιες σταθερές για την ημερομηνία και ώρα.

Enum DatetimeFunction.DatetimeValue

Χρησιμοποιείται για να κρατά τις σταθερές τιμές για την ημερομηνία-ώρα. Αυτές είναι:

- *CURRENT_DATE*
- *CURRENT_TIME*
- *CURRENT_TIMESTAMP*

Class IntervalExpr

Χρησιμοποιείται για να περιγράψει εκφράσεις που περιέχουν χρονικά διαστήματα στην SQL.Είναι υποκλάση της TreeObject ώστε να επιτρέπει την αναπαράσταση αριθμητικών πράξεων με εκφράσεις του τύπου DatetimeExp και NumericExpr.

Enum NonSecondDatetime

Έχει πιθανές τιμές ημερομηνίας-ώρας εκτός από δεπτερόλεπτα. Χρησιμοποιείται από εκφράσεις χρονικών διαστημάτων(interval expressions).Οι τιμές του είναι:

- *DAY*
- *HOUR*
- *MINUTE*

- *MONTH*
- *NULL*
- *YEAR*

Class NumericExpr

Χρησιμοποιείται για να περιγράψει εκφράσεις που περιέχουν αριθμητικές πράξεις. Κληρονομεί την κλάση TreeObject ώστε να επιτρέπει την αναπαράσταση αριθμητικών πράξεων με εκφράσεις του τύπου DatetimeExpr και IntervalExpr.

Enum NumOperator

Περιέχει τιμές που περιγράφουν τους βασικούς αριθμητικούς τελεστές. Χρησιμοποιείται από εκφράσεις που περιλαμβάνουν αριθμητικές πράξεις για να κρατούν τον τύπο της πράξης. Οι τιμές είναι:

- *DIV* για τον αριθμητικό τελεστή της διαίρεσης (/).
- *MINUS* για τον αριθμητικό τελεστή της αφαίρεσης.
- *MULT* για τον αριθμητικό τελεστή του πολλαπλασιασμού.
- *PLUS* για τον αριθμητικό τελεστή της πρόσθεσης.

Class Timezone

Class ExtractField

Χρησιμοποιείται από τη συνάρτηση EXTRACT της SQL και προσδιορίζει το πεδίο εξαγωγής. Περιέχει ένα τύπο enum με τις δυνατές τιμές του πεδίου εξαγωγής οι οποίες είναι : YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, TIMEZONE_HOUR, TIMEZONE_MINUTE.

Enum ExtractField.Field

Διατηρεί τις δυνατές τιμές πεδίου για τη συνάρτηση EXTRACT της SQL.Ο τύπος αυτός περιέχεται στην κλάση ExtractField.Οι τιμές του είναι:

- *DAY*
- *HOUR*
- *MINUTE*
- *MONTH*
- *SECOND*
- *TIMEZONE_HOUR*
- *TIMEZONE_MINUTE*
- *YEAR*

Class IntervalQualifier

Χρησιμοποιείται για να διατηρεί την αρχή και το τέλος ενός χρονικού διαστήματος (interval) στην SQL.

Class SetFunction

Η κλάση αυτή χρησιμοποιείται για να περιγράψει μια συνάρτηση ομαδοποίησης που περιέχεται σε ένα ερώτημα SQL.Έχει μεταβλητές για να προσδιορίζεται ο τύπος της συνάρτησης(AVG,COUNT,MAX,MIN,SUM), μια δυαδική μεταβλητή για επιλογή DISTINCT ή ALL, και τέλος μια μεταβλητή που διατηρεί την έκφραση την οποία αφορά η ομαδοποίηση.

Enum SetFunctionType

Περιέχει τα ονόματα των συναρτήσεων ομαδοποίησης. Αυτά είναι:

- AVG
- COUNT
- MAX
- MIN
- SUM

Class CaseClause

Χρησιμοποιείται όταν υπάρχει μια έκφραση CASE στον κώδικα SQL.

Class CastSpec

Χρησιμοποιείται όταν υπάρχει μια έκφραση CAST στον κώδικα SQL.

Class Column_name

Η κλάση αυτή χρησιμοποιείται σε πολλά ερωτήματα όπου η στήλη ενός πίνακα αναφέρεται με το προσδιοριστικό όνομα ενός πίνακα ή το όνομα του ίδιου του πίνακα. Η κλάση αποθηκεύει το προσδιοριστικό όνομα καθώς και το όνομα της στήλης και τα στοιχεία αυτά μπορούν να χρησιμοποιηθούν στη συνέχεια για την άρση αμφισημιών.

Class StringExpr

Περιγράφει τα χαρακτηριστικά μιας έκφρασης συμβολοσειρών στην SQL.

Class InsertStmt

Χρησιμοποιείται όταν υπάρχει μια δήλωση εισαγωγής (INSERT INTO) σε ένα πίνακα. Τα αντικείμενα της κλάσης αυτής αποθηκεύουν σε μια λίστα τα ονόματα των στηλών στα οποία θα γίνει η εισαγωγή των νέων δεδομένων και σε μια άλλη λίστα τις τιμές των δεδομένων. Σε μια μεταβλητή αποθηκεύεται και το όνομα του πίνακα στον οποίο γίνεται η εισαγωγή. Για την περίπτωση που η εισαγωγή γίνεται με βάση ένα ερώτημα, μια μεταβλητή τύπου SelectStmt δείχνει στο αντικείμενο που αντιπροσωπεύει το ερώτημα.

Class DeleteStmt

Η κλάση αυτή χρησιμοποιείται όταν υπάρχει δήλωση διαγραφής στοιχείων από ένα πίνακα στην SQL.Οι μεταβλητές της κρατούν το όνομα του πίνακα που μεταβάλλεται καθώς και τη συνθήκη με τον οποία γίνεται η μεταβολή.

Class UpdateStmt

Χρησιμοποιείται όταν υπάρχει μια δήλωση UPDATE στην SQL.Έχει μια μεταβλητή που κρατά το όνομα του πίνακα που θα ανανεωθεί, μια λίστα με τις στήλες που θα αλλαχθούν και τις τιμές που προσδίδονται σε αυτές και μια μεταβλητή τύπου BooleanComponent που αναπαριστά τη συνθήκη της ανανέωσης

Class SetValue

Χρησιμοποιείται από τις εκφράσεις UPDATE για να διατηρεί στοιχεία για κάθε μία από τις στήλες που ανανεώνονται.

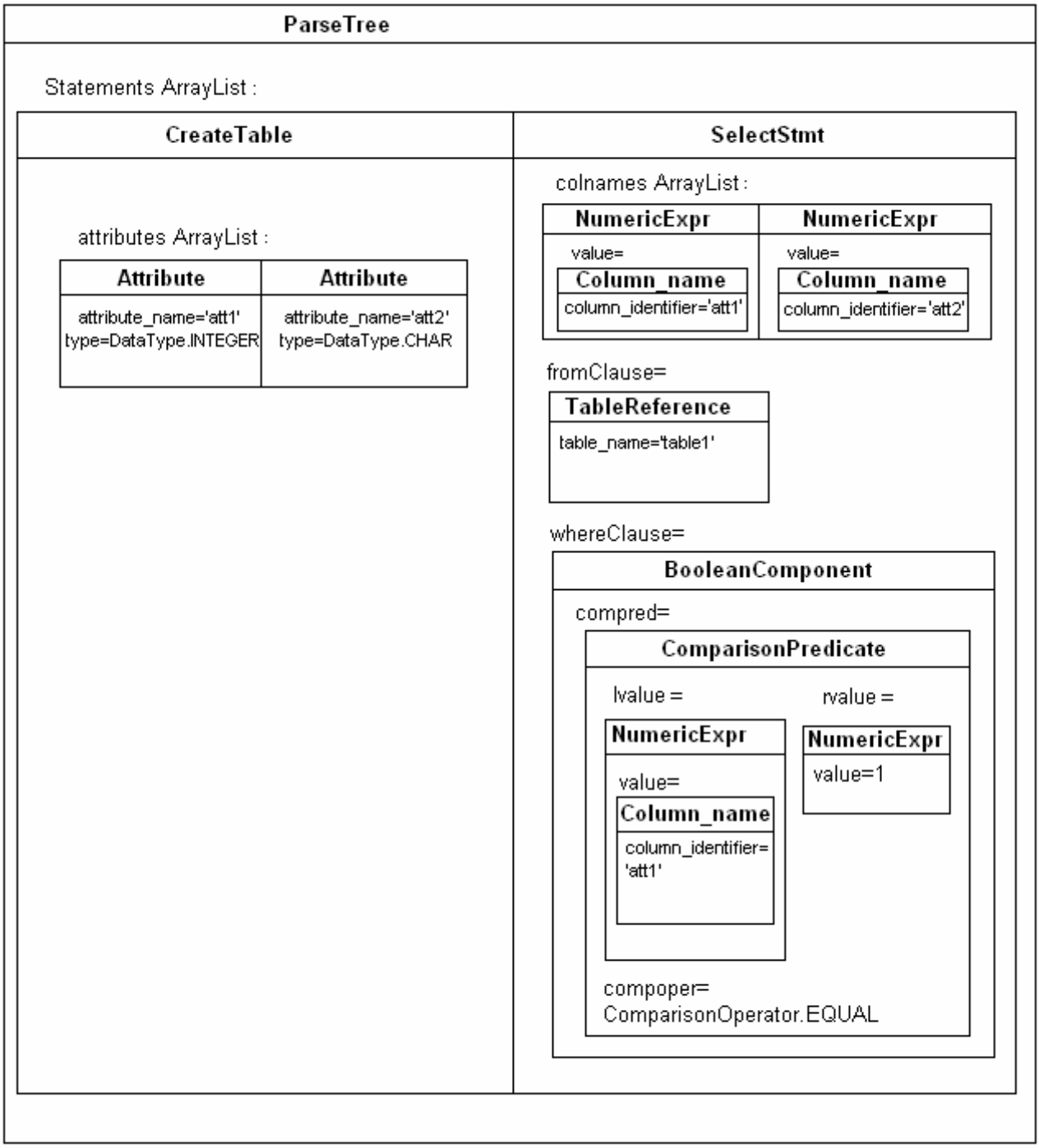
Παρακάτω παρουσιάζεται ένα παράδειγμα ενός μικρού μέρους κώδικα SQL και ένα σχέδιο που παριστάνει σχηματικά τη δημιουργούμενη δομή από την ανάλυση αυτού του κώδικα.

Παράδειγμα κατασκευής της δομής Parse tree.

Παρουσιάζεται ένα παράδειγμα κατασκευής της δομής που περιγράφει το συντακτικό δέντρο με τις παραπάνω κλάσεις.
Ο κώδικας SQL είναι:

```
CREATE TABLE table1(
att1 integer,
att2 char(20)
)

SELECT att1,att2
FROM table1
WHERE att1=1
```



Σχήμα 4.1 Σχηματική αναπαράσταση των δημιουργούμενων αντικειμένων από την ανάλυση του παραπάνω κώδικα SQL.

Όπως φαίνεται από το σχήμα δημιουργείται ένα αντικείμενο της κλάσης ParseTree.Αυτό περιέχει την ArrayList *statements* η οποία γεμίζει με τα αντικείμενα των κλάσεων CreateTable και SelectStmt.Κάθε αντικείμενο των κλάσεων αυτών περιέχει μεταβλητές άλλων κλάσεων ή απλών τύπων που παίρνουν τις κατάλληλες τιμές για να περιγράψουν τα περιεχόμενα του κώδικα SQL.Κάθε κλάση περιέχει μεθόδους που επιστρέφουν την τιμή κάθε μεταβλητής για να μπορεί η παραπάνω δομή δεδομένων να χρησιμοποιηθεί στη συνέχεια.

4.2.2 Πακέτο gr.tuc.softnet.sqlparse.symboltable

Το πακέτο gr.tuc.softnet.sqlparse.symboltable περιέχει τις κλάσεις και τα interfaces που υλοποιούν τον πίνακα συμβόλων του μεταγλωττιστή.

Ο πίνακας συμβόλων αποτελείται από 6 πίνακες κατακερματισμού(hash tables) που αποθηκεύουν αντίστοιχα τους πίνακες ,τα ευρετήρια, τα εναύσματα, τους δρομείς, τις όψεις και τις συναρτήσεις της SQL.

Στην υλοποίηση του πίνακα υποστηρίζεται και η αποθήκευση συναρτήσεων οριζόμενων από το χρήστη. Για τις συναρτήσεις αυτές υπάρχει ο πίνακας κατακερματισμού των συναρτήσεων και οι κλάσεις Function και Argument.

Οι κλάσεις του πακέτου gr.tuc.sqlparse.symboltable φαίνονται στον παρακάτω πίνακα :

1.Κλάση που υλοποιεί τις μεθόδους εισαγωγής εύρεσης και διαγραφής από τους πίνακες κατακερματισμού του πίνακα συμβόλων
SymbolTable
2.Κλάσεις που περιγράφουν τα αντικείμενα που τοποθετούνται στους αντίστοιχους πίνακες κατακερματισμού
Argument
Cursor
Function
Index
Table
Trigger
View
3.Κλάσεις που χρησιμοποιούνται για σημασιολογικό έλεγχο λαθών
STManager
TableRefStack
TableRefAux
TableRefAuxElem
4.Interfaces
Insert
Schema

Πίνακας 4.2 : Οι κλάσεις και τα interfaces του πακέτου gr.tuc.softnet.sqlparse. symboltable.

Το πακέτο gr.tuc.softnet.sqlparse.symboltable περιλαμβάνει 2 interfaces, τα Insert και Schema τα οποία περιέχουν τις δηλώσεις των μεθόδων εισαγωγής και εύρεσης στοιχείων αντίστοιχα στους πίνακες κατακερματισμού. Η υλοποίηση των μεθόδων γίνεται στην κλάση Symboltable.Οι πίνακες κατακερματισμού του πίνακα συμβόλων είναι 7, όσες είναι και οι κλάσεις που περιγράφουν αντικείμενα που εισάγονται σε αυτούς. Οι κλάσεις αυτές ανήκουν στη $2^{η}$ κατηγορία του παραπάνω πίνακα και για κάθε μία υπάρχει και ο αντίστοιχος πίνακας κατακερματισμού. Τέλος οι κλάσεις της $3^{ης}$ κατηγορίας χρησιμοποιούνται για την υλοποίηση σημασιολογικών ελέγχων με βάση τον πίνακα συμβόλων.

Η κλάση SymbolTable κατασκευάζει 7 διαφορετικούς πίνακες κατακερματισμού και υλοποιεί της μεθόδους εισαγωγής εύρεσης και διαγραφής που δηλώνονται στα interfaces Insert και Schema.

Οι κλάσεις που ανήκουν στη δεύτερη κατηγορία, δηλαδή οι Argument, Cursor, Function, Index, Table, Trigger, και View υλοποιούν αντικείμενα των αντίστοιχων στοιχείων της SQL.Κάθε μία από τις παραπάνω κλάσεις περιέχει μεταβλητές οι οποίες περιγράφουν τα χαρακτηριστικά των στοιχείων τα οποία αντιπροσωπεύουν, όπως όνομα ,όνομα των πινάκων στους οποίους ανήκουν κ.α. Για κάθε κλάση από τις παραπάνω υπάρχει και ένας πίνακας κατακερματισμού. Με βάση το όνομα των στοιχείων κατασκευάζεται ένα μοναδικό κλειδί το οποίο καταχωρεί τα αντικείμενα που τα αντιπροσωπεύουν στις κατάλληλες θέσεις των πινάκων κατακερματισμού. Έτσι είναι αρκετά εύκολη και γρήγορη η εύρεση τους όταν χρειαστεί.

Οι κλάσεις της τρίτης κατηγορίας χρησιμοποιούνται για τη διαχείριση του πίνακα συμβόλων και για ελέγχους λαθών. Η κλάση STManager διαβάζει όλες τις δηλώσεις της δομής parsetree και υλοποιεί τις μεθόδους που κάνουν τους απαραίτητους ελέγχους.

Με τη χρήση των interfaces *Insert* και *Schema* μπορεί κάποιος να δημιουργήσει τη δική του υλοποίηση πίνακα συμβόλων και να χρησιμοποιήσει αυτή στην εφαρμογή του. Για να γίνει αυτό πρέπει να γραφεί μια καινούργια κλάση όπως η SymbolTable η οποία θα υλοποιεί τις μεθόδους των interfaces.Η υλοποίηση πρέπει να περιλαμβάνει τα εξής:

```
public class AnotherSymbolTable implements Insert,Schema{

        public void insert_table(String name){
        /*Υλοποίηση της μεθόδου εισαγωγής πίνακα με το αντίστοιχο όνομα*/
           }

        public void insert_attribute(String name,String table_name);
        /*Υλοποίηση της μεθόδου εισαγωγής στήλης σε πίνακα με τα αντίστοιχα
ονόματα στήλης και πίνακα*/
        }
        public void insert_constraint(String name,String table_name);
        /*Υλοποίηση της μεθόδου εισαγωγής περιορισμού πίνακα με τα αντίστοιχα
ονόματα περιορισμού και πίνακα.*/
        }

        public void insert_cursor(String name){
        /*Υλοποίηση της μεθόδου εισαγωγής πίνακα με το αντίστοιχο όνομα*/
        }
```

```
        public void insert_index(String index_name,String table_name,ArrayList
cnames){
        /*Υλοποίηση της μεθόδου εισαγωγής ευρετηρίου με το αντίστοιχο όνομα*/
        }

        public void insert_trigger(String trigger_name,String table_name){
        /*Υλοποίηση της μεθόδου εισαγωγής εναύσματος  με το αντίστοιχο όνομα
εναύσματος και πίνακα*/
        }

        public void insert_view(String view_name,ArrayList list){
        /*Υλοποίηση της μεθόδου εισαγωγής όψεως με το αντίστοιχο όνομα και με τη
λίστα των στηλών που περιλαμβάνει*/
        }


        public void insert_function(String name,String type,ArrayList arguments){
        /*Υλοποίηση της μεθόδου εισαγωγής οριζόμενης από το χρήστη συνάρτησης
με το αντίστοιχο όνομα τύπο επιστροφής και λίστα ορισμάτων*/
        }

        public Table find_table(String name){
        /*Υλοποίηση της μεθόδου εύρεσης πίνακα με το αντίστοιχο όνομα*/
          }

        public STAttribute find_attribute(String table_name,String attribute_name){
        /*Υλοποίηση της μεθόδου εύρεσης στήλης πίνακα με το αντίστοιχο όνομα
πίνακα και στήλης*/
        }

        public STConstraint find_constraint(String table_name,String constraint_name){
        /*Υλοποίηση της μεθόδου εύρεσης περιορισμού πίνακα με το αντίστοιχο
όνομα πίνακα και περιορισμού*/
        }

        public Cursor find_cursor(String name){
        /*Υλοποίηση της μεθόδου εύρεσης δρομέα με το αντίστοιχο όνομα*/
        }

        public Trigger find_trigger(String trigger_name,String table_name){
        /*Υλοποίηση της μεθόδου εύρεσης εναύσματος πίνακα με το αντίστοιχο
όνομα πίνακα και εναύσματος*/
        }

        public View find_view(String view_name){
        /*Υλοποίηση της μεθόδου εύρεσης όψης με το αντίστοιχο όνομα*/
        }

        public Index find_index(String index_name,String table_name){
```

```
        /*Υλοποίηση της μεθόδου εύρεσης ευρετηρίου με το αντίστοιχο όνομα
ευρετηρίου και πίνακα*/
        }

        public Function find_function(String name){
        /*Υλοποίηση της μεθόδου εύρεσης οριζόμενης από το χρήστη συνάρτησης με
το αντίστοιχο όνομα*/


        public boolean delete_attribute(String table_name,String attribute_name){
        /*Υλοποίηση της μεθόδου διαγραφής στήλης πίνακα με το αντίστοιχο
όνομα*/
        }
        public void delete_table(String table_name){
        /*Υλοποίηση της μεθόδου διαγραφής πίνακα με το αντίστοιχο όνομα*/
        }

        public boolean delete_constraint(String table_name,String constraint_name) {
        /*Υλοποίηση της μεθόδου διαγραφής περιορισμού πίνακα με το αντίστοιχο
όνομα πίνακα και περιορισμού*/
        }

        public void delete_cursor(String cursor_name) {
        /*Υλοποίηση της μεθόδου διαγραφής δρομέα με το αντίστοιχο όνομα*/
        }

        public void delete_view(String view_name) {
        /*Υλοποίηση της μεθόδου διαγραφής όψης με το αντίστοιχο όνομα*/
        }

        public void delete_index(String index_name,String table_name) {
        /*Υλοποίηση της μεθόδου διαγραφής ευρετηρίου με το αντίστοιχο όνομα*/
        }
```

Η υλοποίηση, όπως περιγράφεται παραπάνω, κάνει χρήση πινάκων κατακερματισμού(hashtables) για την εύκολη εύρεση των στοιχείων. Με βάση τα ονόματα που παρέχονται από τα ορίσματα των παραπάνω συναρτήσεων κατασκευάζονται κώδικες κατακερματισμού(hashcodes) που βρίσκουν τη θέση των στοιχείων στους πίνακες κατακερματισμού. Ωστόσο αυτή η υλοποίηση δεν είναι δεσμευτική. Μπορεί να κατασκευαστεί πίνακας συμβόλων με διαφορετική δομή δεδομένων, όπως λίστες, αρκεί να εκτελούνται οι εργασίες που περιγράφονται από τις παραπάνω συναρτήσεις.

ΠΕΡΙΓΡΑΦΗ ΤΩΝ ΚΛΑΣΕΩΝ ΤΟΥ ΠΑΚΕΤΟΥ

Class SymbolTable

Η κλάση αυτή υλοποιεί της μεθόδους εισαγωγής , εύρεσης και διαγραφής που δηλώνονται στα Interfaces Insert και Schema.Υλοποιούνται μέθοδοι εύρεσης, εισαγωγής και διαγραφής για πίνακες, στήλες, όψεις, δρομείς, ευρετήρια, εναύσματα οριζόμενες από το χρήστη συναρτήσεις και περιορισμούς στους και από τους αντίστοιχους πίνακες κατακερματισμού.

1. Κλάσεις που περιγράφουν τα αντικείμενα που τοποθετούνται στους αντίστοιχους πίνακες κατακερματισμού

Class Argument

Η κλάση αυτή χρησιμοποιείται για να περιγράψει κάθε ένα όρισμα μιας οριζόμενης από το χρήστη συνάρτησης στην SQL.Έχει μεταβλητές για να διατηρεί το όνομα και τον τύπο του ορίσματος.

Class Cursor

Κάθε αντικείμενο της κλάσης αυτής αποτελεί μια εγγραφή στον πίνακα κατακερματισμού των λογικών δρομέων της SQL.Έχει μια μεταβλητή για να διατηρεί το όνομα του δρομέα.

Class Function

Κάθε αντικείμενο της κλάσης αυτής αποτελεί μια εγγραφή στον πίνακα κατακερματισμού των οριζόμενων από το χρήστη συναρτήσεων της SQL.Οι μεταβλητές της κρατούν το όνομα, τον τύπο επιστροφής και τη λίστα με τα ορίσματα της.

Class Index

Κάθε αντικείμενο της κλάσης αυτής αποτελεί μια εγγραφή στον πίνακα κατακερματισμού των ευρετηρίων της SQL.Έχει μεταβλητές για την αποθήκευση του ονόματος του ευρετηρίου και του πίνακα στον οποίο ανήκει, καθώς και μια ArrayList με τα ονόματα των στηλών που περιλαμβάνει.

Class Table

Κάθε αντικείμενο της κλάσης αυτής αποτελεί μια εγγραφή στον πίνακα κατακερματισμού των πινάκων της SQL.Έχει μια μεταβλητή για την αποθήκευση του ονόματος του πίνακα, μια λίστα για την αποθήκευση των στηλών και μία για την αποθήκευση των περιορισμών.

Class Trigger

Κάθε αντικείμενο της κλάσης αυτής αποτελεί μια εγγραφή στον πίνακα κατακερματισμού των εναυσμάτων της SQL.Έχει μια μεταβλητή που διατηρεί το όνομα του εναύσματος και μία που διατηρεί το όνομα του πίνακα στον οποίο ανήκει.

Class View

Κάθε αντικείμενο της κλάσης αυτής αποτελεί μια εγγραφή στον πίνακα κατακερματισμού των όψεων της SQL.Έχει μια μεταβλητή για την αποθήκευση του ονόματος της όψης και μια λίστα με τις στήλες τις οποίες περιλαμβάνει η όψη.

2.Κλάσεις που χρησιμοποιούνται για σημασιολογικό έλεγχο λαθών

Class STManager

Η κλάση αυτή περιλαμβάνει τις μεθόδους που διαχειρίζονται τις πληροφορίες από το συντακτικό δέντρο για να κατασκευάσει και να τροποποιεί τον πίνακα συμβόλων και να κάνει έλεγχο λαθών με βάση τα δεδομένα που περιέχει. Έτσι η STManager είναι υπεύθυνη για την προσθήκη πινάκων, στηλών, ευρετηρίων, δεικτών, συναρτήσεων, εναυσμάτων και όψεων στους αντίστοιχους πίνακες κατακερματισμού σε κάθε δήλωση τους από την SQL, αλλά και για τον έλεγχο κάθε φορά που υπάρχει αναφορά σε ένα από τα παραπάνω. Η κλάση αυτή διαβάζει τα περιεχόμενα του parse tree και όχι τον αρχικό κώδικα SQL.

Class TableRefStack

Η κλάση αυτή χρησιμοποιείται στον έλεγχο των πινάκων και των στηλών που αναφέρονται σε ένα ερώτημα της SQL.Υλοποιεί μια δομή που λειτουργεί σαν στοίβα και αποτελείται από αντικείμενα της κλάσης TableRefAux που περιγράφεται στη συνέχεια. Η λειτουργία της στοίβας βοηθά στην εύρεση της εμβέλειας ενός πίνακα σε μια ένωση πινάκων που υπάρχει σε ένα ερώτημα. Όταν κλείνει μια εμβέλεια το αντικείμενο τύπου TableRefAux που την περιγράφει αφαιρείται από τη στοίβα. Η κορυφή της στοίβας περιέχει σε κάθε φάση του ελέγχου τη λίστα με τους πίνακες ενεργής εμβέλειας. Ο απαραίτητος έλεγχος γίνεται κάθε φορά με βάση το αντικείμενο που βρίσκεται στην κορυφή της στοίβας. Η στοίβα υλοποιείται με μια ArrayList στην οποία προστίθενται και αφαιρούνται αντικείμενα στο τέλος.

Class TableRefAux

Η κλάση αυτή χρησιμοποιείται για την υλοποίηση των αντικειμένων που υπάρχουν μέσα στην στοίβα TableRefStack.Η κλάση έχει μια λίστα στην οποία αποθηκεύονται αντικείμενα τύπου TableRefAuxElem κάθε ένα από τα οποία περιγράφει έναν πίνακα. Οι πίνακες που βρίσκονται σε αυτή τη λίστα κάθε φορά είναι και αυτοί που έχουν ενεργή εμβέλεια.

Class TableRefAuxElem

Κάθε αντικείμενο της κλάσης αυτής περιγράφει έναν πίνακα που αναφέρεται σε ένα ερώτημα SQL.Επειδή η χρήση της κλάσης αυτής είναι για σημασιολογικό έλεγχο τα στοιχεία του πίνακα που χρειάζεται να διατηρεί είναι το όνομα του πίνακα, το βοηθητικό όνομα με το οποίο πιθανόν αναφέρεται στο ερώτημα και αν ο πίνακας είναι εξαγόμενος από ερώτημα ή μια αναφορά στο ερώτημα αυτό.

4.2.3 Σχεσιακή άλγεβρα

Για την πιστοποίηση από το χρήστη ότι τα περιεχόμενα των ερωτημάτων του κώδικα SQL έχουν εκφραστεί σωστά από τον αναλυτή, έχει υλοποιηθεί η κλάση RelAlgebra η οποία αναλαμβάνει την εκτύπωση στην οθόνη των περιεχομένων των ερωτημάτων της SQL σε σχεσιακή άλγεβρα. Η κλάση RelAlgebra περιέχεται στο πακέτο gr.tuc.softnet.sqlparse.relalgebra και διαβάζει αποκλειστικά το parse tree και όχι τον αρχικό κώδικα SQL.

Στον κώδικα που ακολουθεί φαίνεται η δομή της κλάσης RelAlgebra και με ποιο τρόπο διαβάζει το συντακτικό δέντρο.

```
public class RelAlgebra{

  ParseTree parsetree;
  SelectStmt select;


  public RelAlgebra(ParseTree parsetree){
    this.parsetree=parsetree;
    this.select=null;
  }

/*Εδώ βρίσκονται οι δηλώσεις των μεθόδων της κλάσης */
}
```

Όπως φαίνεται στον κώδικα ο constructor της κλάσης παίρνει ως όρισμα μια μεταβλητή τύπου ParseTree. Έτσι κάθε φορά που κατασκευάζεται ένα αντικείμενο της κλάσης δηλώνεται πιο συντακτικό δέντρο θα διαβαστεί για να προβάλει τα περιεχόμενα των ερωτημάτων.

Από τις δηλώσεις που περιέχει η δομή parse tree αποθηκεύονται σε μια λίστα μόνο τα αντικείμενα SelectStmt που αφορούν τα ερωτήματα του κώδικα. Η μεταβλητή *select* τύπου SelectStmt της κλάσης RelAlgebra που αρχικοποιείται σε *null* από τον constructor χρησιμοποιείται για να αποθηκεύονται προσωρινά αυτά τα ερωτήματα. Στη συνέχεια καλείται η μέθοδος print_select_statements η οποία διαχωρίζει ένα ερώτημα στα παρακάτω μέρη:

- Προβολή
- Επιλογή
- Ομαδοποίηση
- Αναφορά πινάκων

Για κάθε ένα από τα παραπάνω μέρη καλείται η αντίστοιχη μέθοδος που αναλαμβάνει την εκτύπωση των στοιχείων που το συνθέτουν με βάση το parse tree.

Η προβολή αντιστοιχεί στο κομμάτι που ακολουθεί τη λέξη κλειδί SELECT της SQL και συμβολίζεται με (π) στη σχεσιακή άλγεβρα. Στην οθόνη τυπώνεται η λέξη PROJECT για το κομμάτι που αφορά την προβολή. Για κάθε αναφορά σε στήλη η άλλη έκφραση που περιλαμβάνεται στο κομμάτι της προβολής ενός ερωτήματος

καλείται η μέθοδος print_value_expression η οποία ξεχωρίζει το είδος της έκφρασης και καλεί αντίστοιχα τις μεθόδους print_numeric_expression, print_interval_expression ή print_string_expression.Οι δύο πρώτες μέθοδοι είναι αναδρομικές. Διασχίζουν αναδρομικά το δυαδικό δέντρο που περιγράφει την έκφραση και με βάση αυτό την ανασυνθέτουν και την εκτυπώνουν στην οθόνη. Η μέθοδοι έχουν παρόμοια δομή. Η τρίτη μέθοδος τυπώνει εκφράσεις συμβολοσειρών που αποτελούνται από επιμέρους εκφράσεις ενωμένες με το σύμβολο συνένωσης (| |).Οι εκφράσεις σε αυτή την περίπτωση δε σχηματίζουν δυαδικό δέντρο αλλά μια σειριακή δομή. Οι προσπέλαση τους γίνεται σειριακά και κάθε μία τυπώνεται στην οθόνη.

Η επιλογή αντιστοιχεί στο κομμάτι που ακολουθεί τη λέξη κλειδί WHERE της SQL και συμβολίζεται με (σ) στη σχεσιακή άλγεβρα. Στην οθόνη τυπώνεται η λέξη κλειδί SELECT.Η μέθοδος που αναλαμβάνει την ανασύνθεση του κριτηρίου επιλογής από τη δομή parse tree και να το τυπώσει στην οθόνη είναι η print_search_condition. Η μέθοδος αυτή παίρνει ως όρισμα το αντικείμενο τύπου BooleanComponent που αντιστοιχεί στη ρίζα του δυαδικού δέντρου που περιγράφει τη συνθήκη. Όπως έχει αναφερθεί παραπάνω μια συνθήκη παριστάνεται στο parse tree με ένα δυαδικό δέντρο που αποτελείται από αντικείμενα του τύπου BooleanComponent.Η print_search_condition λοιπόν είναι μια αναδρομική μέθοδος που ανασυνθέτει το δυαδικό δέντρο ώστε να εκτυπώσει σωστά τα περιεχόμενά του.

Η ομαδοποίηση αντιστοιχεί στο κομμάτι που ακολουθεί τη λέξη κλειδί GROYP BY στην SQL και συμβολίζεται με (G) στη σχεσιακή άλγεβρα. Στην οθόνη εκτυπώνεται η λέξη GROUP.Η μέθοδος που εκτυπώνει τα δεδομένα μιας έκφρασης ομαδοποίησης είναι η print_groupby.Η μέθοδος αυτή παίρνει ως όρισμα τη λίστα με τις στήλες βάσει των οποίων γίνεται η ομαδοποίηση και τις τυπώνει μία μία.
Το τελευταίο μέρος ενός ερωτήματος που τυπώνεται είναι οι αναφορές στους πίνακες. Αυτή περιλαμβάνει τα ονόματα των πινάκων καθώς και τον τρόπο με τον οποίο συνδέονται στο ερώτημα. Η αναφορά στους πίνακες είναι το κομμάτι που ακολουθεί τη λέξη κλειδί FROM σε ένα ερώτημα SQL.Η μέθοδος που αναλαμβάνει την εκτύπωση στην οθόνη των πινάκων και τη διασύνδεση μεταξύ τους είναι η print_table_reference.Η μέθοδος αυτή είναι αναδρομική και παίρνει ως όρισμα το αντικείμενο τύπου TableReference που αντιστοιχεί στη ρίζα του δυαδικού δέντρου που περιγράφει την αναφορά πινάκων του ερωτήματος. Η μέθοδος διασχίζει αναδρομικά το δέντρο με τέτοιο τρόπο ώστε οι μεγαλύτεροι σε βάθος κόμβοι τυπώνονται πρώτοι. Για κάθε πίνακα τυπώνεται το όνομα του και για κάθε σύνδεσμο η συνθήκη διασύνδεσης αν υπάρχει. Στην εκτύπωση τοποθετούνται παρενθέσεις ώστε να υποδηλώνεται η προτεραιότητα.

Κεφάλαιο 5

Σύνοψη

Όπως φάνηκε στα παραπάνω κεφάλαια, η χρήση σχεσιακών σχημάτων ανεξάρτητα από πηγές δεδομένων είναι αρκετά χρήσιμη και προσφέρει μεγάλη ευελιξία ανάμεσα στα διάφορα συστήματα βάσεων δεδομένων. Επιπρόσθετα, πέρα από την εφαρμογή τους σε βάσεις δεδομένων μπορούν να βρουν και άλλες εφαρμογές, όπως να χρησιμοποιηθούν για υποβολή ερωτημάτων στον παγκόσμιο ιστό, για προγράμματα που χρησιμοποιούν αντικειμενοστραφή προγραμματισμό, για εφαρμογές CAD κ.α.

Ο παρών μεταγλωττιστής της SQL αποτελεί ένα τέτοιο σχεσιακό σχήμα . Η υλοποίηση του καλύπτει τις πιο κοινές εκφράσεις της SQL όπως ερωτήματα, δημιουργία και τροποποίηση πινάκων, δημιουργία εναυσμάτων, όψεων, λογικών δρομέων, ευρετηρίων κ.α. Ακόμη περιλαμβάνει σημασιολογικό έλεγχο για την ορθότητα των εκφράσεων όσων αφορά τη χρήση των παραπάνω. Συνεπώς καλύπτει ένα ευρύ φάσμα των δυνατοτήτων της SQL σχετικά με την περαιτέρω χρήση του και ταυτόχρονα η συμβατότητα με το πρότυπο διευκολύνει την επέκταση του ώστε να καλύπτει περισσότερες εκφράσεις της γλώσσας.

Ο σχεδιασμός του μεταγλωττιστή του δίνει το προνόμιο να μπορεί να επεκταθεί για πολλές διαφορετικές βάσεις δεδομένων. Η χρήση της αντικειμενοστραφούς γλώσσας Java και των εργαλείων της διευκολύνει τη δημιουργία των απαιτούμενων δομών δεδομένων, όπως είναι τα αφηρημένα συντακτικά δέντρα, την ανάγνωση και κατανόηση του κώδικα, καθώς και τη συγγραφή ευανάγνωστης τεκμηρίωσης. Ωστόσο η επέκταση του απαιτεί προσεκτική εργασία και καθορισμένους στόχους και δεν ενδείκνυται η τροποποίηση του συντακτικού αναλυτή.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] *A guide to the SQL standard(third edition)* 1993, C.J Date and Hugh Darwen.

[2] *An Overview of the SQL standards*, Nelson M. Mattos,Hugh Darwen
Paul Cotton, Peter Pistor, Krishna Kulkarni, Stefan Dessloch, Kathryn Zeidenstein

[3] *ISO/IEC 9075:1992, Database Language SQL ,* Digital Equipment Corporation Maynard

[4] *EJB Core Contracts and Requirements* : Sun Microsystems

[5] *WebSQL : Querying the worldwide web*: http://www.cs.toronto.edu/~websql/

[6] *The OQL language.*

[7] Joel Jones : *Abstract Syntax Tree Implementation Idioms.* Department of Computer Science, University of Alabama.

[8] *Java Compiler Compiler (Javacc)* , Sun Microsystems.

[9] *General SQL Parser* : http://www.sqlparser.com/index.php

[10] *PostgreSQL Developer's Guide* : http://www.postgresql.org/

[11] *Wikipedia, the free encyclopedia*, http://www.wikipedia.org

ΠΑΡΑΡΤΗΜΑ

BNF for sqlparse.jj
ΜΗ-ΤΕΡΜΑΤΙΚΑ

mult_statement ::= statement (mult_statement)? <EOF>
statement ::= create_statement
| delete_statement_searched
| drop_statement
| insert_statement
| select_statement
| update_statement_searched
| fetch_statement
| open_statement
| close_statement
| alter_table_statement
alter_table_statement ::= <K_ALTER> <K_TABLE> table_name add_column_definition
| <K_ALTER> <K_TABLE> table_name alter_column_definition
| <K_ALTER> <K_TABLE> table_name drop_column_definition
| <K_ALTER> <K_TABLE> table_name add_table_constraint_definition
| <K_ALTER> <K_TABLE> table_name drop_table_constraint_definition
add_column_definition ::= <K_ADD> (<K_COLUMN>)? column_definition
alter_column_definition ::= <K_ALTER> (<K_COLUMN>)? column_identifier alter_column_action
alter_column_action ::= set_column_default_clause
| drop_column_default_clause
set_column_default_clause ::= <K_SET> default_clause
drop_column_default_clause ::= <K_DROP> <K_DEFAULT>
drop_column_definition ::= <K_DROP> (<K_COLUMN>)?

column_identifier drop_behavior

drop_table_constraint_definition ::= <K_DROP> <K_CONSTRAINT> constraint_name drop_behavior

add_table_constraint_definition ::= <K_ADD> table_constraint_definition

fetch_statement ::= <K_FETCH> ((fetch_orientation)? <K_FROM>)? cursor_name

fetch_orientation ::= <K_NEXT>
| <K_PRIOR>
| <K_FIRST>
| <K_LAST>
| <K_ABSOLUTE> integer
| <K_RELATIVE> integer

open_statement ::= <K_OPEN> cursor_name

close_statement ::= <K_CLOSE> cursor_name

drop_statement ::= drop_table_statement
| drop_view_statement
| drop_index_statement

drop_index_statement ::= <K_DROP> <K_INDEX> index_name <K_ON> table_name

create_statement ::= create_table_statement
| view_definition
| schema_definition
| declare_cursor
| domain_definition
| create_index
| create_trigger_statement

create_trigger_statement ::= <K_CREATE> <K_TRIGGER> trigger_name trigger_action_time trigger_event <K_ON> table_name (<K_REFERENCING> old_or_new_values_alias_list)? triggered_action

trigger_action_time ::= <K_BEFORE>
| <K_AFTER>

trigger_event ::= <K_INSERT>
| <K_DELETE>
| <K_UPDATE> (<K_OF> column_name_list)?

old_or_new_values_alias_list ::= (old_or_new_values_alias)*

old_or_new_values_alias ::= <K_OLD> (<K_ROW>)? (<K_AS>)? actual_identifier
| <K_NEW> (<K_ROW>)? (<K_AS>)? actual_identifier

| <K_OLD> <K_TABLE> (<K_AS>)? actual_identifier
| <K_NEW> <K_TABLE> (<K_AS>)? actual_identifier

triggered_action ::= (<K_FOR> <K_EACH> (<K_ROW> | <K_STATEMENT>))? (<K_WHEN> "(" search_condition ")")? triggered_SQL_statement

triggered_SQL_statement ::= SQL_procedure_statement

SQL_procedure_statement ::= schema_definition
| create_table_statement
| view_definition
| domain_definition
| create_trigger_statement

owner_name ::= <S_IDENTIFIER>

trigger_name ::= <S_IDENTIFIER>

action ::= <K_INSERT>
| <K_UPDATE>
| <K_DELETE>

create_index ::= <K_CREATE> (<K_UNIQUE>)? <K_INDEX> index_name <K_ON> table_name "(" column_identifier ("(" length ")")? ("," column_identifier ("(" length ")")?)* ")"

index_name ::= <S_IDENTIFIER>

declare_cursor ::= <K_DECLARE> cursor_name (<K_INSENSITIVE>)? (<K_SCROLL>)? <K_CURSOR> <K_FOR> query_expression (order_by_clause)? (<K_FOR> (<K_READ> <K_ONLY> | <K_UPDATE>))?

cursor_name ::= <S_IDENTIFIER>

schema_definition ::= <K_CREATE> <K_SCHEMA> schema_name_clause (schema_character_set_specification)? (schema_element)*

schema_name_clause ::= schema_name
| <K_AUTHORIZATION> schema_authorization_identifier
| schema_name <K_AUTHORIZATION> schema_authorization_identifier

schema_authorization_identifier ::= authorization_identifier

authorization_identifier ::= actual_identifier

schema_character_set_specification ::= <K_DEFAULT> <K_CHARACTER> <K_SET> character_set_name

character_set_name ::= (schema_name ".")? <S_IDENTIFIER>

schema_element ::= domain_definition create_table_statement
| view_definition

domain_definition ::= <K_CREATE> <K_DOMAIN>
domain_name (<K_AS>)? data_type (
default_clause)? (domain_constraint)* (
collate_clause)?

domain_constraint ::= (constraint_name_definition)?
check_constraint_definition (
constraint_attributes)?

drop_view_statement ::= <K_DROP> <K_VIEW> table_name
drop_behavior

drop_behavior ::= <K_CASCADE>
| <K_RESTRICT>

view_definition ::= <K_CREATE> <K_VIEW> table_name ("("
view_column_list ")")? <K_AS>
query_expression (<K_WITH> (
<K_CASCADED> | <K_LOCAL>)?
<K_CHECK> <K_OPTION>)?

view_column_list ::= column_name_list

create_table_statement ::= <K_CREATE> <K_TABLE>
base_table_name "(" column_definition (","
table_element)* ")"

table_element ::= column_definition
| table_constraint_definition

column_definition ::= column_identifier (data_type | domain_name
) (default_clause)? (
column_constraint_definition)* (
collate_clause)?

column_constraint_definition ::= (constraint_name_definition)?
column_constraint (constraint_attributes)?

constraint_name_definition ::= <K_CONSTRAINT> constraint_name

column_constraint ::= <K_NOT> <K_NULL>
| unique_specification
| references_specification
| check_constraint_definition

references_specification ::= <K_REFERENCES>
referenced_table_and_columns (
<K_MATCH> match_type)? (
referential_triggered_action)?

referenced_table_and_columns ::= table_name ("(" referencing_columns ")")?

referencing_columns ::= <S_IDENTIFIER> ("," <S_IDENTIFIER>
)*

match_type ::= <K_FULL>
| <K_PARTIAL>

referential_triggered_action ::= update_rule (delete_rule)?
| delete_rule (update_rule)?
delete_rule ::= <K_ON> <K_DELETE> referential_action
update_rule ::= <K_ON> <K_UPDATE> referential_action
default_option ::= literal
| datetime_value_function
| <K_USER>
| <K_CURRENT_USER>
| <K_SESSION_USER>
| <K_SYSTEM_USER>
| <K_NULL>
referential_action ::= <K_CASCADE>
| <K_SET> <K_NULL>
| <K_SET> <K_DEFAULT>
| <K_NO> <K_ACTION>
unique_specification ::= <K_UNIQUE>
| <K_PRIMARY> <K_KEY>
constraint_name ::= qualified_name
default_clause ::= <K_DEFAULT> default_option
table_constraint_definition ::= (constraint_name_definition)?
table_constraint (constraint_attributes)?
table_constraint ::= unique_constraint_definition
| referential_constraint_definition
| check_constraint_definition
referential_constraint_definition ::= <K_FOREIGN> <K_KEY> "("
referencing_columns ")"
references_specification
check_constraint_definition ::= <K_CHECK> "(" search_condition ")"
constraint_attributes ::= constraint_check_time ((<K_NOT>)?
<K_DEFERRABLE>)?
| (<K_NOT>)? <K_DEFERRABLE> (
constraint_check_time)?
constraint_check_time ::= <K_INITIALLY> (<K_DEFERRED> |
<K_IMMEDIATE>)
unique_constraint_definition ::= unique_specification "(" unique_column_list
")"
unique_column_list ::= column_name_list
domain_name ::= qualified_name
delete_statement_searched ::= <K_DELETE> <K_FROM> table_identifier
(<K_WHERE> search_condition)?
drop_table_statement ::= <K_DROP> <K_TABLE> base_table_name
(<K_IF> <K_EXISTS>)?
insert_statement ::= <K_INSERT> <K_INTO> table_identifier (

"(" column_identifier ("," column_identifier)* ")")? <K_VALUES> "(" insert_value ("," insert_value)* ")"

| <K_INSERT> <K_INTO> table_identifier ("(" column_identifier ("," column_identifier)* ")")? query_expression

select_statement ::= query_expression (order_by_clause)?

query_expression ::= query_term ((<K_UNION> | <K_EXCEPT>) query_expression)?

query_term ::= query_primary (<K_INTERSECT> query_term)?

query_primary ::= <K_SELECT> (set_quantifier)? select_list <K_FROM> table_reference_list (<K_WHERE> search_condition)? (group_by_clause)? (having_clause)?

| row_subquery

table_reference_list ::= table_reference ("," table_reference_list)?

set_quantifier ::= <K_DISTINCT>

| <K_ALL>

having_clause ::= <K_HAVING> search_condition

group_by_clause ::= <K_GROUP> <K_BY> grouping_column_reference_list

grouping_column_reference_list ::= grouping_column_reference ("," grouping_column_reference)*

grouping_column_reference ::= column_reference

column_reference ::= column_name

qualifier ::= correlation_name

correlation_name ::= actual_identifier

collate_clause ::= <K_COLLATE> collation_name

collation_name ::= qualified_name

update_statement_searched ::= <K_UPDATE> table_identifier <K_SET> column_identifier "=" (value_expression | <K_NULL> | <K_DEFAULT>) ("," column_identifier "=" (value_expression | <K_NULL> | <K_DEFAULT>))* (<K_WHERE> search_condition)?

base_table_name ::= <S_IDENTIFIER>

search_condition ::= boolean_term (<K_OR> search_condition)?

boolean_term ::= boolean_factor (<K_AND> boolean_term)?

boolean_factor ::= (<K_NOT>)? boolean_primary

boolean_primary ::= comparison_predicate

| "(" search_condition ")"

character_string_literal ::= <S_CHAR_LITERAL>

column_identifier ::= <S_IDENTIFIER>

column_name ::= (table_identifier ".")? column_identifier

comparison_operator ::= "<"
| ">"
| "<="
| ">="
| "="
| "<>"
| <K_LIKE>
| <K_NOT> <K_LIKE>

comparison_predicate ::= row_value_constructor comparison_operator row_value_constructor

row_value_constructor ::= row_value_constructor_element
| row_subquery

value_expression_primary ::= column_reference
| set_function_specification
| unsigned_value_specification
| subquery
| case_expression
| cast_specification
| "(" value_expression ")"

case_expression ::= case_abbreviation
| case_specification

case_abbreviation ::= <K_NULLIF> "(" value_expression "," value_expression ")"
| <K_COALESCE> "(" value_expression ("," value_expression)* ")"

case_specification ::= simple_case
| searched_case

simple_case ::= <K_CASE> case_operand (simple_when_clause)* (else_clause)? <K_END>

case_operand ::= value_expression

simple_when_clause ::= <K_WHEN> when_operand <K_THEN> result

when_operand ::= value_expression

result ::= result_expression
| <K_NULL>

result_expression ::= value_expression

else_clause ::= <K_ELSE> result

searched_case ::= <K_CASE> (searched_when_clause)* (else_clause)? <K_END>

searched_when_clause ::= <K_WHEN> search_condition <K_THEN>

result

unsigned_value_specification ::= unsigned_literal
| general_value_specification

general_value_specification ::= <K_USER>
| <K_CURRENT_USER>
| <K_SESSION_USER>
| <K_SYSTEM_USER>
| <K_VALUE>

cast_specification ::= <K_CAST> "(" cast_operand <K_AS> (domain_name | data_type) ")"

cast_operand ::= value_expression
| <K_NULL>

set_function_specification ::= <K_COUNT> "(" "*" ")"
| general_set_function

general_set_function ::= set_function_type "(" (set_quantifier)? value_expression ")"

set_function_type ::= <K_AVG>
| <K_MAX>
| <K_MIN>
| <K_SUM>
| <K_COUNT>

unsigned_literal ::= unsigned_numeric_literal

unsigned_numeric_literal ::= <S_NUMBER>

signed_integer ::= "+" <S_INTEGER>
| "-" <S_INTEGER>

row_value_constructor_element ::= value_expression
| <K_NULL>
| <K_DEFAULT>

row_subquery ::= subquery

subquery ::= "(" query_expression ")"

data_type ::= character_string_type
| national_character_string_type
| bit_string_type
| numeric_type
| datetime_type
| interval_type

character_string_type ::= <K_CHARACTER> ("(" length ")")?
| <K_CHAR> ("(" length ")")?
| <K_CHARACTER> <K_VARYING> "(" length ")"
| <K_CHAR> <K_VARYING> "(" length ")"
| <K_VARCHAR> "(" length ")"

national_character_string_type ::= <K_NATIONAL> <K_CHARACTER> ("(" length ")")?
| <K_NATIONAL> <K_CHAR> ("(" length ")")?
| <K_NCHAR> ("(" length ")")?
| <K_NATIONAL> <K_CHARACTER> <K_VARYING> "(" length ")"
| <K_NATIONAL> <K_CHAR> <K_VARYING> "(" length ")"
| <K_NCHAR> <K_VARYING> "(" length ")"

bit_string_type ::= <K_BIT> ("(" length ")")?
| <K_BIT> <K_VARYING> "(" length ")"

numeric_type ::= exact_numeric_type
| approximate_numeric_type

exact_numeric_type ::= <K_NUMERIC> ("(" precision ("," scale)? ")")?
| <K_DECIMAL> ("(" precision ("," scale)? ")")?
| <K_DEC> ("(" precision ("," scale)? ")")?
| <K_INTEGER>
| <K_INT>
| <K_SMALLINT>

precision ::= <S_INTEGER>

scale ::= <S_INTEGER>

approximate_numeric_type ::= <K_FLOAT> ("(" precision ")")?
| <K_REAL>
| <K_DOUBLE> <K_PRECISION>

datetime_type ::= <K_DATE>
| <K_TIME> ("(" time_precision ")")? (<K_WITH> <K_TIME> <K_ZONE>)?
| <K_TIMESTAMP> ("(" timestamp_precision ")")? (<K_WITH> <K_TIME> <K_ZONE>)?

time_precision ::= <S_INTEGER>

timestamp_precision ::= <S_INTEGER>

interval_type ::= <K_INTERVAL> interval_qualifier

interval_qualifier ::= start_field <K_TO> end_field
| single_datetime_field

start_field ::= non_second_datetime_field ("(" interval_leading_field_precision ")")?

non_second_datetime_field ::= <K_YEAR>
| <K_MONTH>

| <K_DAY>
| <K_HOUR>
| <K_MINUTE>

interval_leading_field_precision ::= <S_INTEGER>

end_field ::= non_second_datetime_field
| <K_SECOND> ("(" interval_double_seconds_precision ")")?

interval_double_seconds_precision ::= <S_NUMBER>

single_datetime_field ::= non_second_datetime_field ("(" interval_leading_field_precision ")")?
| <K_SECOND> ("(" interval_double_seconds_precision ")")?

numeric_value_expression ::= term (("-" numeric_value_expression | "+" numeric_value_expression))?

term ::= factor (("*" term | "/" term))?

factor ::= ("+")? numeric_primary
| ("-")? numeric_primary

numeric_primary ::= column_name
| literal
| integer
| number
| numeric_value_function
| "(" numeric_value_expression ")"

numeric_value_function ::= position_expression
| extract_expression
| char_length_expression

position_expression ::= <K_POSITION> "(" character_value_expression <K_IN> character_value_expression ")"

extract_expression ::= <K_EXTRACT> "(" extract_field <K_FROM> extract_source ")"

extract_field ::= <K_YEAR>
| <K_MONTH>
| <K_DAY>
| <K_HOUR>
| <K_MINUTE>
| <K_SECOND>
| <K_TIMEZONE_HOUR>
| <K_TIMEZONE_MINUTE>

extract_source ::= datetime_value_expression
| interval_value_expression

char_length_expression ::= (<K_CHAR_LENGTH> |

<K_CHARACTER_LENGTH>) "("
string_value_expression ")"

insert_value ::= literal
| <K_NULL>
| <K_USER>
| integer
| number

literal ::= signed_numeric_literal
| general_literal

signed_numeric_literal ::= "+" unsigned_numeric_literal
| "-" unsigned_numeric_literal

general_literal ::= character_string_literal

order_by_clause ::= <K_ORDER> <K_BY> sort_specification (
"," sort_specification)*

integer ::= <S_INTEGER>

number ::= <S_NUMBER>

length ::= <S_INTEGER>

select_list ::= "*"
| select_sublist ("," select_sublist)*

select_sublist ::= value_expression (<K_AS>
<S_IDENTIFIER>)?
| qualifier "." "*"

value_expression ::= numeric_value_expression
| string_value_expression
| datetime_value_expression
| interval_value_expression

interval_value_expression ::= interval_term ("+" interval_value_expression
)?
| interval_term ("-" interval_value_expression
)?
| "(" datetime_value_expression ")"
interval_qualifier

interval_term ::= interval_factor
| factor "*" interval_term
| factor "/" interval_term
| term "*" interval_factor

interval_factor ::= "+" interval_primary
| "-" interval_primary

interval_primary ::= value_expression_primary (interval_qualifier
)?

character_value_expression ::= character_factor ("||"
character_value_expression)?

datetime_value_expression ::= datetime_term
| interval_value_expression "+" datetime_term
| interval_term "+" datetime_value_expression
| interval_term "-" datetime_value_expression

datetime_term ::= datetime_factor

datetime_factor ::= datetime_primary (time_zone)?

datetime_primary ::= value_expression_primary
| datetime_value_function

datetime_value_function ::= <K_CURRENT_DATE>
| <K_CURRENT_TIME> ("(" precision ")")?
| <K_CURRENT_TIMESTAMP> ("(" precision ")")?

time_zone ::= <K_AT> time_zone_specifier

time_zone_specifier ::= <K_LOCAL>
| <K_TIME> <K_ZONE> interval_value_expression

string_value_expression ::= character_value_expression

character_factor ::= character_primary (collate_clause)?

character_primary ::= value_expression_primary
| character_value_function

character_value_function ::= character_substring_function
| fold
| form_of_use_conversion
| character_translation
| trim_function

character_substring_function ::= <K_SUBSTRING> "(" character_value_expression <K_FROM> start_position (<K_FOR> string_length)? ")"

start_position ::= numeric_value_expression

string_length ::= numeric_value_expression

fold ::= (<K_UPPER> | <K_LOWER>) "(" character_value_expression ")"

form_of_use_conversion ::= <K_CONVERT> "(" character_value_expression <K_USING> form_of_use_conversion_name ")"

form_of_use_conversion_name ::= qualified_name

character_translation ::= <K_TRANSLATE> "(" character_value_expression <K_USING> translation_name ")"

translation_name ::= qualified_name

trim_function ::= <K_TRIM> "(" ((trim_specification)? (trim_character)? <K_FROM>)? trim_source

")"

trim_specification ::= <K_LEADING>
| <K_TRAILING>
| <K_BOTH>

trim_character ::= character_value_expression

trim_source ::= character_value_expression

sort_specification ::= (<S_INTEGER> | column_name) (
<K_ASC> | <K_DESC>)?

table_identifier ::= <S_IDENTIFIER>

table_reference ::= "(" table_reference ")"
| table_name ((<K_AS>)? correlation_name
("(" derived_column_list ")")?)? (joined)*
| derived_table (<K_AS>)? correlation_name
("(" derived_column_list ")")? (joined)?

joined ::= <K_CROSS> <K_JOIN> table_reference
| (<K_NATURAL>)? (join_type)?
<K_JOIN> table_reference (join_condition |
named_columns_join)

derived_table ::= subquery

table_name ::= qualified_name
| qualified_local_table_name

qualified_name ::= qualified_identifier

schema_name ::= (catalog_name ".")?
unqualified_schema_name

catalog_name ::= actual_identifier

unqualified_schema_name ::= actual_identifier

qualified_local_table_name ::= <K_MODULE> "." local_table_name

local_table_name ::= qualified_identifier

qualified_identifier ::= actual_identifier

derived_column_list ::= column_name_list

column_name_list ::= actual_identifier ("," actual_identifier)*

join_type ::= <K_INNER>
| <K_LEFT>
| <K_RIGHT>
| <K_FULL>
| <K_LEFT> <K_OUTER>
| <K_RIGHT> <K_OUTER>
| <K_FULL> <K_OUTER>
| <K_UNION>

join_condition ::= <K_ON> search_condition

named_columns_join ::= <K_USING> "(" join_column_list ")"

join_column_list ::= column_name_list

```
actual_identifier ::= <S_IDENTIFIER>
                    |  <S_QUOTED_IDENTIFIER>
```