ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΔΙΠΛΩΜΑ ΕΙΔΙΚΕΥΣΗΣ (MSc)
στα ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«QuLMO: Μια γλώσσα ερωτήσεων για κινούμενα αντικείμενα»

Μπρακατσούλας Σωτήρης

M3000017

ΑΘΗΝΑ, Φεβρουάριος 2002

ΜΕΤΑΠΤΥΧΙΑΚΟ ΔΙΠΛΩΜΑ ΕΙΔΙΚΕΥΣΗΣ (MSc)
στα ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«QuLMO: Μια γλώσσα ερωτήσεων για κινούμενα αντικείμενα»

Μπρακατσούλας Σωτήρης

M3000017

Επιβλέπων Καθηγητής: Μ. Βαζιργιάννης
Εξωτερικός Κριτής: Καθηγητής Ε. Γιαννακουδάκης

ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΘΗΝΑ, Φεβρουάριος 2002

QuLMO: Μια γλώσσα ερωτήσεων για κινούμενα αντικείμενα

Εκτενής Περίληψη

Οι βάσεις δεδομένων για κινούμενα αντικείμενα γίνονται όλο και περισσότερο δημοφιλείς μιας και σε πολλά πεδία εφαρμογών είναι αισθητή η ανάγκη της διαχείρισης αντικειμένων με χωροχρονικές ιδιότητες και της δυνατότητας πραγματοποίησης ερωτήσεων που αφορούν αυτές τις ιδιότητες. Έως τώρα, οι υλοποιήσεις αυτού του είδους των συστημάτων είναι λιγότερο ώριμες σε σύγκριση με τα καθιερωμένα και αξιόπιστα παραδοσιακά συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων. Στην παρούσα εργασία παρουσιάζουμε ένα μοντέλο δεδομένων και μια εκφραστική γλώσσα ερωτήσεων για κινούμενα αντικείμενα. Επιπρόσθετα, προτείνουμε ένα σχέδιο υλοποίησής τους ως επέκταση που ενσωματώνεται στα υπάρχοντα σύγχρονα σχεσιακά - αντικειμενοστραφή συστήματα διαχείρισης βάσεων δεδομένων (ΣΑ - ΣΔΒΔ), μέσω κατάλληλων μηχανισμών επέκτασης που προσφέρουν τα τελευταία, διατηρώντας με αυτό τον τρόπο τα πλεονεκτήματα της υψηλής αξιοπιστίας, της εύρωστης απόδοσης, των μηχανισμών ασφαλείας και ανάκαμψης κ.λπ. που χαρακτηρίζουν τα σύγχρονα ΣΑ - ΣΔΒΔ της αγοράς. Η πρότασή μας συμπληρώνεται από την υλοποίηση ενός πρωτοτύπου συστήματος ως επέκταση της Oracle 9i και με μια πειραματική μελέτη της απόδοσής του.

Οι χωροχρονικές εφαρμογές χαρακτηρίζονται από αυξημένες απαιτήσεις διαχείρισης δεδομένων (data intensive systems), δεδομένης της περισσότερο πολύπλοκης φύσης των χωροχρονικών αντικειμένων, των σχέσεων με τις οποίες συνδέονται και των δοσοληψιών (transactions) που τα αφορούν, σε σύγκριση με τις εφαρμογές που διαχειρίζονται παραδοσιακά αλφαριθμητικά δεδομένα.

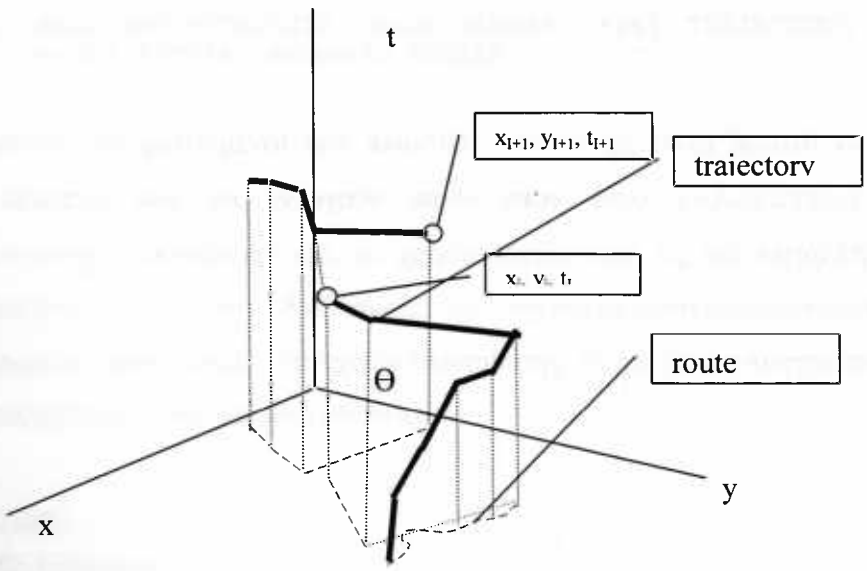
Για παράδειγμα ας θεωρήσουμε μια εταιρία ταξί που επιθυμεί να βελτιστοποιήσει τις παρεχόμενες από αυτή υπηρεσίες μέσω ενός συστήματος που θα της επιτρέπει να γνωρίζει κάθε στιγμή την ακριβή θέση των οχημάτων της. Πιθανές ερωτήσεις σε ένα τέτοιο σύστημα μπορούν να είναι οι εξής:

- Ποιο είναι το κοντινότερο ταξί σε μια δεδομένη διεύθυνση;

- Ποια ταξί θα βρίσκονται ενός απόστασης πέντε χιλιομέτρων από τη διεύθυνση ενός πελάτη στα επόμενα δέκα λεπτά;
- Έχουν οι τροχιές των ταξί Α και Β συναντηθεί τις τελευταίες δύο ώρες;
- Δεδομένου ότι γνωρίζουμε τις προβλεπόμενες τροχιές, θα υπάρξει χρονική στιγμή όπου τα ταξί Α και Β θα απέχουν μεταξύ τους λιγότερο από δύο χιλιόμετρα στα επόμενα τριάντα λεπτά;

Η χρήση των διαθέσιμων τύπων δεδομένων για τη μοντελοποίηση των αντικειμένων της παραπάνω εφαρμογής καθώς και ο τρόπος με τον οποίο μπορούν να εκφρασθούν τα παραπάνω ερωτήματα δεν είναι προφανής στα σημερινά εμπορικά ΣΔΒΔ. Κύριος σκοπός της παρούσας εργασίας αποτελεί η διερεύνηση τρόπων αποδοτικής ενσωμάτωσης ενός μοντέλου δεδομένων καθώς και μιας γλώσσας ερωτήσεων κατάλληλων για κινούμενα αντικείμενα σε περιβάλλον αντικειμενοστραφούς - σχεσιακού συστήματος διαχείρισης βάσεων δεδομένων (ΣΔΒΔ), δηλαδή ο σχεδιασμός μιας επέκτασης ενός ΣΑ-ΣΔΒΔ και συγκεκριμένα ένα "custom data cartridge" που ενσωματώνεται στην Oracle 9i.

Τα δεδομένα που παράγονται από την κίνηση αντικειμένων σχηματίζουν μια καμπύλη σε τρισδιάστατο χώρο, όπου δύο από τις τρεις διαστάσεις αντιστοιχούν στο δυσδιάστατο (x-, y-) επίπεδο και η τρίτη διάσταση αντιστοιχεί στο χρόνο. Το πλαίσιο αυτό μπορεί εύκολα να επεκταθεί στις 4 διαστάσεις για αντικείμενα του πραγματικού κόσμου που κινούνται πάνω από την επιφάνεια της γης. Αντί της τρισδιάστατης καμπύλης, σε μια βάση δεδομένων κινούμενων αντικειμένων αποθηκεύουμε και διαχειριζόμαστε μια τρισδιάστατη ακολουθία γραμμικών τμημάτων, τα οποία αναπαριστούν την *τροχιά* του αντικειμένου (σχήμα 1) και προκύπτουν από τη δειγματοληψία της συνεχούς κίνησής του. Όπως παρατηρείται και από το σχήμα 1, η διαχείριση θέσεων αντικειμένων που μεταβάλλονται με συνεχή τρόπο από τα υπάρχοντα ΣΔΒΔ αντιμετωπίζει δύο σημαντικά προβλήματα: τα υπάρχοντα ΣΔΒΔ δεν μπορούν να διαχειριστούν άπειρα σύνολα και έτσι αναγκαστικά υπάρχει απώλεια πληροφορίας, αλλά και τα υπάρχοντα συστήματα καταγραφής - παρακολούθησης θέσης (GPS και άλλες τεχνολογίες) διαχειρίζονται διακριτή πληροφορία. Ως συνέπεια των παραπάνω, υπάρχουν χρονικά διαστήματα που δεν είναι δυνατό να γνωρίζουμε επακριβώς τη θέση ενός αντικειμένου και για το λόγο αυτό χρησιμοποιούνται



Σχήμα 1

τεχνικές παρεμβολής, είτε γραμμικής είτε περισσότερο πολύπλοκες προσεγγίσεις, για να υπολογίσουμε τη θέση του σε χρονικές στιγμές των συγκεκριμένων διαστημάτων. Αυτή η έλλειψη πληροφορίας εισαγάγει την έννοια της αβεβαιότητας, η οποία πρέπει να λαμβάνεται υπόψη για να αποφεύγονται λανθασμένα αποτελέσματα. Στη συνέχεια παρουσιάζουμε ένα σύνολο τύπων δεδομένων και ερωτήσεων ικανό να εκφράσει τις λειτουργίες που απαιτούν συνήθεις εφαρμογές κινούμενων αντικειμένων.

Τα σχεσιακά - αντικειμενοστραφή ΣΔΒΔ μας επιτρέπουν να ορίσουμε κατάλληλους τύπους δεδομένων για την τροχιά και τα τμήματα της τροχιάς ενός αντικειμένου με την βοήθεια των τύπων δεδομένων για αντικείμενα και σύνολα (collections) αντικειμένων. Με τη χρήση DDL προτάσεων προκύπτουν οι ορισμοί (χρησιμοποιούμε το συντακτικό της Oracle, ανάλογο είναι το συντακτικό σε άλλα ΣΔΒΔ.):

```
CREATE TYPE TRAJECTORYSEGMENT AS OBJECT(t_lower NUMBER, t_upper
NUMBER, x_lower NUMBER, y_lower NUMBER, x_upper NUMBER, y_upper
NUMBER)

CREATE TYPE TRAJECTORY AS TABLE OF TRAJECTORYSEGMENT
```

Χρησιμοποιώντας τους παραπάνω ορισμούς μπορούμε να αποθηκεύσουμε τις τροχιές κινούμενων αντικειμένων σε ένα συνήθη πίνακα:

```
CREATE TABLE MOVINGOBJECTS (moid NUMBER, traj TRAJECTORY, colorid
NUMBER, weight NUMBER, driverid NUMBER)
```

Επιπρόσθετα, επωφελούμενοι των πλαισίων επέκτασης, είναι δυνατό να ορίσουμε νέους τελεστές που θα ενεργούν πάνω στον τύπο TRAJECTORY *για την* πραγματοποίηση ερωτήσεων και να χρησιμοποιήσουμε τις πιο κατάλληλες δομές δεικτοδότησης που είναι διαθέσιμες σε σχεσιακά-αντικειμενοστραφή ΣΔΒΔ. Συγκεκριμένα, προτείνουμε ένα σχέδιο υλοποίησης, το οποίο και πραγματοποιήσαμε, που περιλαμβάνει οκτώ τύπους ερωτήσεων:

1. LOC(time)
2. WHENAT(location)
3. WHITHIN(traveltime from R, along existing path, always between st and et)
4. WHITHIN(traveldistance from R, along existing path, always between st and et)
5. WHITHIN(traveldistance from R, along shortest path, always between st and et)
6. WHITHIN(traveltime from R, along existing path, sometimes between st and et)
7. WHITHIN(traveldistance from R, along existing path, sometimes between st and et)
8. WHITHIN(traveldistance from R, along shortest path, sometimes between st and et)

Εναλλακτικά, σε μια SQL-like μορφή:

```
SELECT LOC(t) | WHEN_AT(location-L) |
       <other attributes of MOVINGOBJECTS >
FROM MOVINGOBJECTS
WHERE WITHIN (DISTANCE s | TRAVELTIME t) FROM R
      [(ALONG EXISTING PATH) | (ALONG SHORTEST PATH) ]
      [(ALWAYS BETWEEN) | (SOMETIMES BETWEEN) st AND et]
```

όπου:

- `LOC(t)`: είναι οι θέσεις όλων των αντικειμένων τη στιγμή t. Ο συγκεκριμένος τελεστής επιστρέφει μια λίστα από ζεύγη σημείων και αναγνωριστικών αντικειμένων.
- `WHEN_AT (location-L)`: οι χρονικές στιγμές κατά τις οποίες κάποιο υποσύνολο κινούμενων αντικειμένων διέρχονται από τη θέση `location-L`. Το αποτέλεσμα της ερώτησης είναι ένα σύνολο από ζεύγη χρονικών στιγμών και αναγνωριστικών των αντικειμένων. Ένα αναγνωριστικό αντικειμένου μπορεί να εμφανίζεται περισσότερες από μία φορές στο αποτέλεσμα, αν το αντίστοιχο αντικείμενο πέρασε περισσότερες από μία φορές από τη θέση `location-L`.

- WITHIN (DISTANCE s | TRAVELTIME t) FROM R: είναι ένας τελεστής που απαιτεί ένα από τα δύο ορίσματα: DISTANCE s ή TRAVELTIME t, όπου s και t είναι πραγματικοί αριθμοί και R ένα σημείο αναφοράς στο δυσδιάστατο χώρο. Ο τελεστής επιστρέφει τη λογική τιμή True αν το αντικείμενο απαιτείται να διανύσει απόσταση μικρότερη ή ίση του s για να διέλθει από το σημείο R, ή αν απαιτείται χρόνος μικρότερος ή ίσος του t για να διέλθει από το σημείο R. Επίσης, προσδιορίζοντας και τα υπόλοιπα μη υποχρεωτικά ορίσματα, μπορούμε να απαιτήσουμε την επαλήθευση ισχυρότερων περιορισμών: α. ALONG EXISTING PATH and ALONG SHORTEST PATH. Οι δύο τιμές αυτού του ορίσματος είναι αμοιβαία αποκλειόμενες. Η πρώτη απαιτεί το κριτήριο WITHIN να είναι αληθές κατά μήκος της τροχιάς του αντικειμένου (πράγμα που σημαίνει ότι το σημείο R πρέπει να βρίσκεται πάνω στην τροχιά του αντικειμένου). Ποιοτικά, αυτό σημαίνει ότι το αντικείμενο μπορεί να διέλθει από το R χωρίς να υπερβεί το μέτρο κόστους που έχει προσδιοριστεί (χρόνος ή απόσταση) καθώς κινείται πάνω στην τροχιά του. Η δεύτερη δυνατή τιμή του ορίσματος απαιτεί το κριτήριο WITHIN να είναι αληθές κατά μήκος του συντομότερου μονοπατιού (ως συντομότερο μονοπάτι θεωρούμε την ευθεία που συνδέει την τρέχουσα θέση του αντικειμένου με το σημείου αναφοράς, R με αρχή την τρέχουσα θέση του αντικειμένου και τέλος τη θέση R). Αν καμιά από τις δύο τιμές δεν προσδιοριστεί, ως προεπιλογή θεωρείται η τιμή ALONG EXISTING PATH. β. ALWAYS BETWEEN and SOMETIMES BETWEEN starttime AND endtime. Οι δύο τιμές αυτού του ορίσματος αναφέρονται στη χρονική διάρκεια του κριτηρίου WITHIN. Η πρώτη απαιτεί το κριτήριο WITHIN να είναι αληθές για όλες τις χρονικές στιγμές μεταξύ starttime και endtime, ενώ η δεύτερη το κριτήριο WITHIN να είναι αληθές για μία τουλάχιστον χρονική στιγμή μεταξύ starttime και endtime. Αν καμιά από τις δύο τιμές δεν προσδιοριστεί, ως προεπιλογή θεωρείται η τιμή ALWAYS BETWEEN και ως starttime και endtime θεωρούνται οι μικρότερη και η μεγαλύτερη τιμή, αντίστοιχα που υπάρχει στη βάση δεδομένων για κάθε αντικείμενο.

Θα ήταν ιδανικό να προσπαθήσουμε να ενσωματώσουμε την προτεινόμενη γλώσσα ερωτήσεων σε όλα τα μεγάλα ΣΔΒΔ που κυκλοφορούν στην αγορά και να

αναφέρουμε τις εμπειρίες μας σχετικά με το πιο θεωρούμε περισσότερο κατάλληλο. Μέχρι στιγμής, επιλέξαμε να πειραματιστούμε με το λογισμικό Oracle 9i.

Χρησιμοποιώντας την ορολογία της Oracle, ο σχεδιασμός του συστήματος αντιστοιχεί στην ανάπτυξη ενός νέου Data Cartridge. Έτσι, απαιτούνται τα παρακάτω βήματα:

- Ορισμός κατάλληλων τύπων δεδομένων με τη βοήθεια των Abstract Data Types.
- Ορισμός και υλοποίηση τελεστών μέσω των οποίων θα πραγματοποιούνται νέοι τύποι ερωτήσεων κατάλληλων για κινούμενα αντικείμενα.
- Επιλογή της καταλληλότερης μεθόδου δεικτοδότησης από αυτές που είναι διαθέσιμες.

Έχουμε ήδη περιγράψει τον τρόπο με τον οποίο πραγματοποιούμε το πρώτο βήμα, ενώ το τρίτο βήμα πραγματοποιείται με την υλοποίηση αποθηκευμένων συναρτήσεων οι οποίες στη συνέχεια συνδέουμε (bind) με τους νέους τελεστές που ορίζουμε.

Σχετικά με τη δεικτοδότηση, η αρχική μας πρόθεση ήταν να αποθηκεύσουμε εγγραφές του τύπου (movingobjectid, segmentid, 3dtrajectorysegment) και να οργανώσουμε αυτόν τον πίνακα - ευρετήριο με ένα "λογικό" R-tree *της* Oracle Spatial. Όμως, αν και η Oracle Spatial υποστηρίζει αντικείμενα έως και 4 διαστάσεων, στη δεικτοδότηση χρησιμοποιούνται μόνο οι δύο πρώτες. Κατά συνέπεια, αποφασίσαμε να δεικτοδοτήσουμε τη χωρική διάσταση ξεχωριστά από την χρονική με ένα "λογικό" R-tree και ένα Relational Interval tree, αντίστοιχα, αποθηκεύοντας εγγραφές της μορφής (moid, segid, t_lower, t_upper, artificial_node, 2dspatialsegment), όπου 2dspatialsegment είναι ένας τύπος δεδομένων (γραμμικό τμήμα) που προσφέρει *η* Oracle Spatial *και* (t_lower, t_upper, artificial_node) *η* σχεσιακή αναπαράσταση ενός χρονικού διαστήματος όπως αυτή χρησιμοποιείται από *το* Relational Interval tree.

Αν και τα πλαίσια επέκτασης αποτελούν ένα πολύτιμο μηχανισμό για το σχεδιαστή *μιας εφαρμογής* για τη διαχείριση μη παραδοσιακών δεδομένων και ταυτόχρονα αποκρύπτουν τις λεπτομέρειες της υλοποίησης από το χρήστη, για τη γρήγορη προτυποποίηση του συστήματος, έχουμε υλοποιήσει μια πιο απλή εκδοχή, στο

πνεύμα, όμως, όλων αυτών που έχουμε περιγράψει μέχρι στιγμής. Αποθηκεύουμε όλα τα δεδομένα για την κίνηση των αντικειμένων σε ένα συνήθη πίνακα:

```
CREATE TABLE TRAJECTORIES(
moid     NUMBER,          -- moving object id
segid    NUMBER,          -- trajectory segment id
node     NUMBER,          -- artificial node for RI-tree
lower    NUMBER,          -- lower time interval bound
upper    NUMBER,          -- upper time interval bound spatseg
MDSYS.SDO_GEOMETRY        -- 2d spatial segment)
```

Για τη χρονική δεικτοδότηση, έχουμε ορίσει δύο σύνθετα B-tree στα ζεύγη στηλών (node, lower) και (node, upper) και έχουμε επίσης υλοποιήσει συναρτήσεις για τον υπολογισμό της τιμής του τεχνητού κόμβου node για την πραγματοποίηση εισαγωγών καθώς της συνάρτησης που προετοιμάζει τις λίστες "left" και "right" για την πραγματοποίηση ερωτήσεων επικάλυψης στο χρόνο. Ακολουθούμε ακριβώς τη μεθοδολογία που προτάθηκε από τους δημιουργούς του Relational Interval Tree και έχουμε επαληθεύσει ότι παράγουμε το ίδιο πλάνο εκτέλεσης. Για τη χωρική δεικτοδότηση δημιουργούμε ένα Relational R-tree πάνω στη στήλη spatseg. Επίσης, υλοποιήσαμε τους νέους τύπους ερωτήσεων ως αποθηκευμένες συναρτήσεις δύο βημάτων. Κατά το πρώτο βήμα επιλέγεται ένα σύνολο εγγραφών με χρήση είτε του χρονικού είτε του χωρικού ευρετηρίου και κατά το δεύτερο εξετάζεται με ακρίβεια η ισχύς των περιορισμών που θέτουν οι τελεστές ερωτήσεων.

Επιπρόσθετα, η υλοποίηση που πραγματοποιήσαμε ελέγχθηκε ως προς την ικανότητα διαχείρισης πολλών αντικειμένων (scalability) με ένα σύνολο ενδεικτικών πειραμάτων σχετικά με το χρόνο απόκρισης των ερωτήσεων για συνθετικά σύνολα δεδομένων που δημιουργήσαμε με τη γεννήτρια του T. Brinkhoff.

Μελλοντικές βελτιώσεις και επεκτάσεις της παρούσας εργασίας μπορούν να αναζητηθούν στις παρακάτω κατευθύνσεις:

- Αναπαράσταση της κίνησης των αντικειμένων με τη βοήθεια προτύπων (patterns) και συναρτήσεων.
- Αναζήτηση κατάλληλων αντιστοιχήσεων δομών δεικτοδότησης σε σχεσιακά σχήματα, όπως το RI-tree, μιας και χωρίς κατάλληλες δομές το εφικτό της αποδοτικής υλοποίησης μοντέλων δεδομένων και γλωσσών ερωτήσεων είναι αμφισβητήσιμο.

- Πειραματισμός και με πλαίσια επέκτασης λειτουργικότητας πέραν του παρεχόμενου από την Oracle.
- Διερεύνηση θεμάτων αβεβαιότητας (uncertainty) που αφορούν τη θέση ενός κινούμενου αντικειμένου μεταξύ δύο δειγματοληπτημένων θέσεων (τα άκρα ενός τμήματος τροχιάς).
- Διερεύνηση θεμάτων εξόρυξης δεδομένων κίνησης (motion mining) και δυνατότητας πραγματοποίησης ερωτήσεων όπως η εύρεση όμοιων συμπεριφορών κίνησης.

QuLMO: A Language for Querying Moving Objects

Executive Summary

Moving object databases are becoming more popular due to the increasing number of application domains that deal with moving entities and need to pose queries. So far implementations of such systems have been rather weak and certainly not at industrial strength level. In this work we present a concise data model and a set of powerful query predicates for moving objects. Exploiting this model we implemented a system based on off-the-shelf industrial solutions enhancing thus the applicability and robustness of our approach. Furthermore, we test the feasibility of the approach by performing a set of indicative experiments concerning the query response time of the proposed query predicates.

Application domains related to moving objects produce massive data and call for data base support. The main entities that characterize this context are the object's location and trajectory and motion features such as speed, acceleration etc. Then inter-object relationships are interesting in the sense of i. objects' location relations and ii. objects' trajectories. Both types of relationships are enriched due to their dependency on time.

As an example, vehicle location involves the management of spatiotemporal objects. Assume a provider's service that needs to know continuously the location of trucks and optimize their services. Potential queries include the following:

- Which is the closest truck to a specific address now?
- Which trucks will be within 5 km from a clients' address in the next 10 minutes for some time.
- Have the trajectories of trucks A and B intersected in the previous 2 hours?
- Assuming we know the anticipated trajectories will trucks A and B be closer than 2 km to each other in the next 30 minutes?

Such queries are quite cumbersome to be expressed and processed by current commercial database products that handle spatial/temporal data. Moreover the co-existence of spatial and temporal attributes in the system as well as its interdependencies make the query processing issue a challenging problem. Thus

requirements for extended functionality able to handle these features rise. We dealt with the issue in terms of designing and developing a Query Language for Moving Objects (QuLMO). The implementation was based on commercial off the shelf database technology proving the feasibility of the design. The contributions of this work are summarized in the following:

- A simple data type system and a small but powerful set of query predicates covering rich requirements regarding collections of moving objects.
- A full implementation of the data types and operations as extensions of a commercial DBMS. More specifically we used the Oracle RDBMS and extension technologies. The result is a custom Data Cartridge consisting of: i. the trajectory segment and trajectory data types ii. the definition and implementation of an appropriate index type for trajectory objects, and iii. definition and implementation of query operators.

Hereafter, we restrict the discussion to moving point objects, i.e. objects with zero extent that change their position continuously over a road network. Such objects are called moving objects for short, they are pervasive, but in contrast to the discretely changing objects, they are much more difficult to accommodate in a database. The motion of such objects can be represented by a 3D «string», but in a database environment what is usually available is a 3D polyline (see Figure 1), resulting from the sampling of an object's motion, that represents the *trajectory* of the object, or a sequence of 3D line segments *(trajectory segments)*, where the first two dimensions correspond to the spatial space and the third to corresponds to time. Note that using such a representation results to information loss, and, in order to calculate the position of an object at a given time or the time that it passes through a given position, along a 3D-trajectory segment, we have to interpolate. In the following we define: a. a concise set of data types to represent moving objects and their structure, b. a set of operations/predicates that can be applied on units/sets of moving objects in order to retrieve desired information on them.

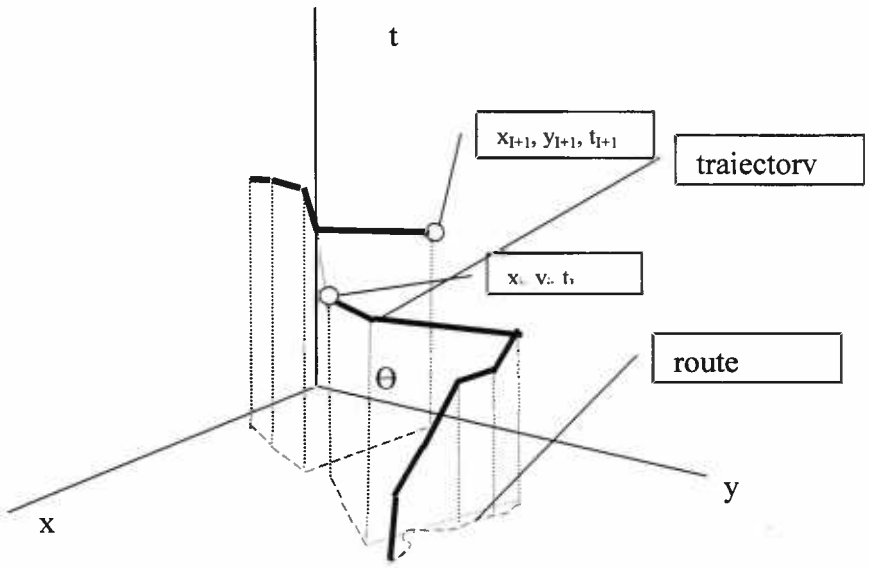


Figure 1

The application context is about the moving objects and their properties. Basically we need to know their identifier, their initial location and their trajectories. Having done the assumptions above we define the trajectory as a set of consecutive line segments. Most modern object-relational DBMS (e.g. Oracle) allow us to define appropriate data types for an object's trajectory segments and trajectory using the notion of object types and object collections respectively. Thus, in terms of DLL statements, we have the following definitions:

```
CREATE TYPE TRAJECTORYSEGMENT AS OBJECT(t_lower NUMBER,
t_upper NUMBER, x_lower NUMBER, y_lower NUMBER, x_upper
NUMBER, y_upper NUMBER)

CREATE TYPE TRAJECTORY AS TABLE OF TRAJECTORYSEGMENT
```

Using the above definitions we can store moving objects in a normal database table

```
CREATE TABLE MOVINGOBJECTS (moid NUMBER, traj TRAJECTORY,
colorid NUMBER, weight NUMBER, driverid NUMBER)
```

Furthermore, by utilizing the extensibility interfaces provided by the above mentioned DBMS we can define suitable operators for the implementation of the proposed spatiotemporal predicates and use the most appropriate indexing methods available in such systems. In particular, we proposed and implemented a design for eight spatiotemporal predicates:

9. LOC(time)
10. WHENAT(location)
11. WHITHIN(traveltime from R, along existing path, always between st and et)
12. WHITHIN(traveldistance from R, along existing path, always between st and et)
13. WHITHIN(traveldistance from R, along shortest path, always between st and et)
14. WHITHIN(traveltime from R, along existing path, sometimes between st and et)
15. WHITHIN(traveldistance from R, along existing path, sometimes between st and et)
16. WHITHIN(traveldistance from R, along shortest path, sometimes between st and et)

Alternatively, embedded in an SQL-like statement:

```
SELECT LOC(t) | WHEN_AT(location-L) |
       <other attributes of MOVINGOBJECTS >
FROM MOVINGOBJECTS
WHERE WITHIN (DISTANCE s | TRAVELTIME t) FROM R
      [(ALONG EXISTING PATH) | (ALONG SHORTEST PATH) ]
      [(ALWAYS BETWEEN) | (SOMETIMES BETWEEN) st AND et]
```

where:

• `LOC(t)` returns the locations of all moving objects at a specific time t. The predicate returns a list of pairs constintuting of a moving object identifier and a 2D point corresponding to the location of the object for all objects whose position is known at the specified time instance.

• `WHEN_AT (location-L)` returns the time (or times) at which the moving objects passed from the location location-L. For each object an indentifier and a list of time A set of time values is returned.

• `WITHIN (DISTANCE s | TRAVELTIME t) FROM R` is a predicate that requires one of the two operands: `DISTANCE s` or `TRAVELTIME t`, where `s` is a distance value and `t` is a value indicating traveltime. `R` is a point of a polygon. The predicate returns true when either: i. the object needs to travel less than `s` distance units in order to arrive to R_2 or ii. the object needs to travel less than `t` time units in order to arrive to R. This predicate is further refined by the following two groups of quantifiers: a. `ALONG EXISTING PATH` and `ALONG SHORTEST PATH`. They are optional mutually exclusive quantifiers. The former imposes the `WITHIN` criterion to be true along the route of the object (which implies that `R` belongs to the route of the object) while the second requires that the `WITHIN` constraint is valid each time following the shortest path (the straight line connecting the current object position and the point R) between the object and R. If none of them is present in the

query ALONG EXISTING PATH is used as the default value. b. ALWAYS BETWEEN and SOMETIMES BETWEEN st AND et. These optional quantifiers relate to the temporal duration of the WITHIN criterion. The former requires that the WITHIN criterion is true for all times between the st and et values while the latter requires that the WITHIN criterion is true for some times between st and et. If none of them is present in the query ALWAYS BETWEEN is used as the default value and the st and et are assigned the st and et of the object respectively.

It would be ideal to try the embedding of the proposed language into all of the major object-relational DBMS and report our experiences about which one we find more suitable. Currently, we have selected Oracle for our initial experimentation. Using Oracle terminology, the design of our system corresponds to the design of a custom Data Cartridge. Thus, the following steps are required:

- Definition of the trajectory segment and trajectory data types presented in the previous section.
- Definition and implementation of an appropriate index type for trajectory objects with the use of the extensibility interface, or selection of appropriate index types among those offered by the DBMS either as built-in data types either as extensions (e.g. B+-trees and spatial access methods[1] in the latest versions of Oracle).
- Definition and implementation of operators that can be evaluated using the index defined in the previous step. The implementation of these operators corresponds to the implementation of the proposed spatiotemporal predicates.

We have already described how the first step is carried out through the DDL statements of the previous section, while the third step is carried out by implementing user-defined functions that can be bind to the custom operators. In order to carry out the second step, our fist thought was to use an index table and store records of the form (movingobjectid, segmentid, 3dlinesegment) and index the 3dlinesegment column using the functionality of Oracle Spatial. But, although Oracle Spatial supports up to four-dimensional objects, it takes into account only the

[1] Although these are "logical" indexes. For example the R-tree index type provided by Oracle Spatial is really a simulation of a hierarchical search tree, but it provides query functionality, from the user's point of view, similar to the original R-tree structure.

first two dimensions in order to index the data. As a result, we have decided to index the spatial dimension using the data types and indexing functionality provided by Oracle Spatial and the temporal dimension by using the Relational Interval tree structure. Taken into account these two choices we finally store records of the form (moid, segid, t_lower, t_upper, artificial_node, 2dspatialsegment), where 2dspatialsegment is a data type offered by Oracle Spatial and (t_lower, t_upper, artificial_node) is a relational representation of a time interval. Oracle Spatial provides support for two classes of data types: *simple* (point, linear segment, polygon) and *compound* (ordered lists of *simple* type elements). Table columns of such data types can be indexed using "logical" R-trees. The Relational Interval tree structure for time interval management is based on an efficient mapping of a hierarchical search tree to a relational schema, the so-called Relational Interval (RI)-tree. The primary structure of an RI-tree consists of a binary search tree of height h that indexes the values of the range $[0, 2^{h-1}]$ of potential interval bounds. A time interval is "registered" at the first node (found when descending the tree from the root to the leaves) whose value intersects it. Two lists for each node of the binary tree are used to store the lower and upper bounds, respectively, of the time intervals "registered" at this node. The set of all lists of all nodes constitutes the secondary structure. The hierarchical primary structure is not materialized, only the root value, 2^h, is stored as "metadata" and in order to find the intersecting node, when inserting an interval, only arithmetic operations are needed. For the relational storage of the intervals, the nodes of the primary structure are used as artificial key values. Furthermore, the primary and secondary structures are simulated by two relational schemas: *lowerindex(artificialnode, lowerbound, id)* and *upperindex(artificialnode, upperbound, id)*. These two relational schemas are organized by two B-trees, respectively, in order to facilitate efficient intersection queries.

Currently, although we have not fully wrapped our implementation within the extensibility framework, we have implemented the core functionality. Consequently, instead of storing the data into the `MOVINGOBJETS` table, «unnesting» the object types, and creating an index table, we adopt a more straightforward approach in order to rapidly prototype our design. We store the data in a simple table created with the statement:

```
CREATE TABLE TRAJECTORIES(
```

```
   moid     NUMBER,            -- moving object id
   segid    NUMBER,            -- trajectory segment id
   node     NUMBER,            -- artificial node for RI-tree
   lower    NUMBER,            -- lower time interval bound
   upper    NUMBER,            -- upper time interval bound
   spatseg MDSYS.SDO_GEOMETRY-- 2d spatial segment
);
```

For the part of temporal indexing, we defined two composite B-tree indexes on the (node, lower) and (node, upper) column couples and we have also implemented the procedure for calculating the value of the artificial node when inserting, and the procedure for preparing the "left" and "right" node lists used for time interval intersection queries. We followed exactly the methodology of the original RI-tree structure proposal and we have verified that we generate the same execution plan. For the part of spatial indexing we constructed a "logical" (also termed as "relational") R-tree on the `spatseg` column. We have implemented the spatiotemporal query types as stored functions that first perform a selection query and then some refinement steps for each record contained in the result set.

Furthermore, we tested the feasibility of the query languge through a set of indicative experiments concerning the query response time of the proposed query predicates. We used sets of synthetic data generated using Brinkhoff's generator of road network moving objects.

As future work we distinguish the following main directions:

- Addressing the issue of continuous objects' motion representation in terms of patterns/functions. This will apparently increase the accuracy and expressiveness of the modeling. Of course the query operators will have to be designed in a totally different way capitalizing on the continuity features rather than on the discrete positions of the approach as currently adopted.
- Discovering efficient mappings of access methods to relational schemas, since without proper indexing support, the feasibility of new data models and query languages is questionable.
- Experimentation with other commercial DBMS and evaluation of their extensibility interfaces.
- Dealing with uncertainty issues concerning the position of a moving object between two sampled positions (along the 3D trajectory segments).

QuLMO: Μια γλώσσα ερωτήσεων για κινούμενα αντικείμενα

Περίληψη. Οι βάσεις δεδομένων για κινούμενα αντικείμενα γίνονται όλο και περισσότερο δημοφιλείς μιας και σε πολλά πεδία εφαρμογών είναι αισθητή η ανάγκη της διαχείρισης αντικειμένων με χωροχρονικές ιδιότητες καθώς και της δυνατότητας πραγματοποίησης ερωτήσεων που αφορούν αυτές τις ιδιότητες. Έως τώρα, οι υλοποιήσεις αυτού του είδους των συστημάτων είναι λιγότερο ώριμες σε σύγκριση με τα καθιερωμένα και αξιόπιστα παραδοσιακά συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων. Στην παρούσα εργασία παρουσιάζουμε ένα μοντέλο δεδομένων και μια εκφραστική γλώσσα ερωτήσεων για κινούμενα αντικείμενα. Επιπρόσθετα, προτείνουμε ένα σχέδιο υλοποίησής τους ως επέκταση που ενσωματώνεται στα υπάρχοντα σύγχρονα σχεσιακά - αντικειμενοστραφή συστήματα διαχείρισης βάσεων δεδομένων (ΣΑ - ΣΔΒΔ), μέσω κατάλληλων μηχανισμών επέκτασης που προσφέρουν τα τελευταία, διατηρώντας με αυτό τον τρόπο τα πλεονεκτήματα της υψηλής αξιοπιστίας, της εύρωστης απόδοσης, των μηχανισμών ασφαλείας και ανάκαμψης κ.λπ. που χαρακτηρίζουν τα σύγχρονα ΣΑ - ΣΔΒΔ της αγοράς. Η πρότασή μας συμπληρώνεται από την υλοποίηση ενός πρωτοτύπου συστήματος ως επέκταση της Oracle 9i και με μια πειραματική μελέτη της απόδοσής του.

Αθήνα, Φεβρουάριος 2002

Περιεχόμενα

1 Εισαγωγή

Την τελευταία δεκαετία έχει ενταθεί το ερευνητικό ενδιαφέρον για εφαρμογές ικανές να διαχειρίζονται χωροχρονικά δεδομένα και έχουν πραγματοποιηθεί σημαντικά βήματα για την ανάπτυξη κατάλληλων μοντέλων δεδομένων, γλωσσών ερωτήσεων και δομών δεικτοδότησης.

Τυπικές χωροχρονικές εφαρμογές αποτελούν τα Γεωγραφικά Συστήματα Πληροφοριών που αξιοποιούν και χρονική πληροφορία (π.χ. εφαρμογές κτηματολογίου για τη διαχείριση περιοχών που εξελίσσονται στο χρόνο), οι εφαρμογές πραγματικού χρόνου για τη διαχείριση της κυκλοφορίας οχημάτων, οι εφαρμογές εικονικής πραγματικότητας, η μελέτη της κίνησης αντικειμένων με σκοπό την πρόβλεψη πιθανών μελλοντικών διαδρομών, η μελέτη φαινομένων, όπως για παράδειγμα καιρικά φαινόμενα που εξελίσσονται στο χώρο και το χρόνο, κ.λπ. Πρόκειται στην πλειοψηφία τους για εφαρμογές με αυξημένες απαιτήσεις διαχείρισης δεδομένων (data intensive systems), δεδομένης της περισσότερο πολύπλοκης φύσης των χωροχρονικών αντικειμένων, των σχέσεων με τις οποίες συνδέονται και των δοσοληψιών (transactions) που τα αφορούν, σε σύγκριση με τις εφαρμογές που διαχειρίζονται παραδοσιακά αλφαριθμητικά δεδομένα.

Στην παρούσα εργασία το ενδιαφέρον μας στρέφεται στην διαχείριση χωροχρονικών αντικειμένων που αντιστοιχούν σε κινούμενα αντικείμενα. Για παράδειγμα ας θεωρήσουμε μια εταιρία ταξί που επιθυμεί να βελτιστοποιήσει τις παρεχόμενες από αυτή υπηρεσίες μέσω ενός συστήματος που θα της επιτρέπει να γνωρίζει κάθε στιγμή την ακριβή θέση των οχημάτων της. Πιθανές ερωτήσεις σε ένα τέτοιο σύστημα μπορούν να είναι οι εξής:

- Ποιο είναι το κοντινότερο ταξί σε μια δεδομένη διεύθυνση;
- Ποια ταξί θα βρίσκονται ενός απόστασης πέντε χιλιομέτρων από τη διεύθυνση ενός πελάτη στα επόμενα δέκα λεπτά;
- Έχουν οι τροχιές των ταξί Α και Β συναντηθεί τις τελευταίες δύο ώρες;
- Δεδομένου ότι γνωρίζουμε τις προβλεπόμενες τροχιές, θα υπάρξει χρονική στιγμή όπου τα ταξί Α και Β θα απέχουν μεταξύ τους λιγότερο από δύο χιλιόμετρα στα επόμενα τριάντα λεπτά;

Η χρήση των διαθέσιμων τύπων δεδομένων για τη μοντελοποίηση των αντικειμένων της παραπάνω εφαρμογής καθώς και ο τρόπος με τον οποίο μπορούν να

εκφρασθούν τα παραπάνω ερωτήματα δεν είναι προφανής στα σημερινά εμπορικά ΣΔΒΔ.

Κύριος σκοπός της παρούσας εργασίας αποτελεί η διερεύνηση τρόπων αποδοτικής ενσωμάτωσης ενός μοντέλου δεδομένων καθώς και μιας γλώσσας ερωτήσεων κατάλληλα για κινούμενα αντικείμενα σε περιβάλλον αντικειμενοστραφούς - σχεσιακού συστήματος διαχείρισης βάσεων δεδομένων (ΣΔΒΔ). Ακολουθούμε τις προτάσεις που διατυπώθηκαν στο άρθρο [VW01] και πραγματοποιούμε μια πρωτότυπη υλοποίηση χρησιμοποιώντας το λογισμικό Oracle στην έκδοση 9i.

Η δομή της εργασίας έχει ως ακολούθως. Στην επόμενη ενότητα, πραγματοποιούμε μια σύντομη επισκόπηση χωροχρονικών εννοιών, μοντέλων δεδομένων, γλωσσών ερωτήσεων και δομών δεικτοδότησης, όπως εμφανίζονται στην πρόσφατη βιβλιογραφία, με έμφαση σε κινούμενα αντικείμενα. Στην τρίτη ενότητα, καταγράφουμε τις προσπάθειες, σε ερευνητικό αλλά και σε πρακτικό επίπεδο, που έχουν πραγματοποιηθεί προς την κατεύθυνση της χρήσης της ήδη υπάρχουσας αντικειμενοστραφούς-σχεσιακής τεχνολογίας για τη διαχείριση μη παραδοσιακών δεδομένων (χρονικών, χωρικών και χωροχρονικών) και παρουσιάζουμε δύο επεκτάσεις για σχεσιακά - αντικειμενοστραφή ΣΔΒΔ για διαχείριση χωρικών και χρονικών δεδομένων, αντίστοιχα. Τις επεκτάσεις αυτές τις χρησιμοποιούμε για την υλοποίηση - ένταξη σε σχεσιακό - αντικειμενοστραφές περιβάλλον της προτεινόμενης γλώσσας ερωτήσεων. Στην τέταρτη ενότητα, περιγράφουμε το προτεινόμενο προς υλοποίηση μοντέλο δεδομένων και της αντίστοιχης γλώσσας ερωτήσεων. Στην πέμπτη ενότητα, παρουσιάζουμε το σχεδιασμό (σχεδιασμός τύπων δεδομένων, σχήματα επεξεργασίας ερωτήσεων κ.λπ.) του πρωτοτύπου συστήματος για τη διαχείριση κινούμενων αντικειμένων καθώς και τον τρόπο και τα διαθέσιμα μέσα για την επίτευξη της υλοποίησης σε περιβάλλον αντικειμενοστραφούς - σχεσιακού ΣΔΒΔ. Στην έκτη ενότητα, επαληθεύουμε το εφικτό της πρότασής μας με μια σειρά από πειράματα που αφορούν το χρόνο απόκρισης των νέων τύπων ερωτήσεων. Κλείνουμε με τα συμπεράσματα που προέκυψαν και με τις πιθανές μελλοντικές κατευθύνσεις εργασίας.

2 Χωροχρονικές βάσεις δεδομένων

Τα συστήματα διαχείρισης χωροχρονικών βάσεων δεδομένων διαχειρίζονται δεδομένα των οποίων οι χωρικές ιδιότητες μεταβάλλονται στο χρόνο και, όπως αναφέραμε στην εισαγωγή, υπάρχουν αρκετές εφαρμογές που παράγουν αυτού του είδους τα δεδομένα. Συνήθως οι χωρικές ιδιότητες που μας ενδιαφέρουν είναι δύο: η *θέση (position)* ενός αντικειμένου και η *έκταση (extent)* που καταλαμβάνει. Ανάλογα με το ρυθμό μεταβολής των χωρικών ιδιοτήτων στο χρόνο, διακρίνουμε δύο περιπτώσεις χωροχρονικών εφαρμογών. Η πρώτη αναφέρεται σε περιβάλλον διακριτών αλλαγών (μικρός ρυθμός μεταβολής των χωρικών ιδιοτήτων) και η δεύτερη σε περιβάλλον συνεχών αλλαγών (μεγάλος ρυθμός μεταβολής των χωρικών ιδιοτήτων). Στη συνέχεια παραθέτουμε κάποια εισαγωγικά στοιχεία για τα βασικά χαρακτηριστικά των δύο αυτών περιπτώσεων εφαρμογών και μερικά αντιπροσωπευτικά είδη ερωτήσεων, ενώ, στις επόμενες υποενότητες, παρουσιάζουμε αναλυτικότερα τις θεμελιώδεις χωροχρονικές έννοιες που θα μας απασχολήσουν καθώς και μοντέλα δεδομένων, γλώσσες ερωτήσεων και δομές ευρετηρίων που έχουν προταθεί έως τώρα στη βιβλιογραφία.

Τα φυσικά αντικείμενα του πραγματικού κόσμου, χαρακτηρίζονται από μια χωρική *θέση* (π.χ. ένα ζεύγος συντεταγμένων σε δύο διαστάσεων καρτεσιανό επίπεδο) και την έκταση (extent) που καταλαμβάνουν (π.χ. το εμβαδόν ή όγκος που καταλαμβάνει ένα αντικείμενο σε κάποιο χώρο αναφοράς) σε κάθε χρονική στιγμή [EGSV98]. Ένα αεροπλάνο που πετά πάνω από τη γη, ένα αυτοκίνητο που ταξιδεύει σε έναν αυτοκινητόδρομο, η περιοχή που καλύπτει ένα δάσος καθώς αναπτύσσεται ή συρρικνώνεται στο χρόνο και ένα αντικείμενο που ταυτόχρονα κινείται και μεταβάλλει το μέγεθός του σε μια ταινία κινούμενων σχεδίων αποτελούν χαρακτηριστικά παραδείγματα. Στις εφαρμογές που διαχειρίζονται τέτοιου είδους αντικείμενα (*χωροχρονικά* αντικείμενα) είναι συνηθισμένες ερωτήσεις σχετικά με τη θέση και την έκταση των αντικειμένων στο παρελθόν, το παρόν και το μέλλον. Οι χωροχρονικές βάσεις δεδομένων ανταποκρίνονται στη ανάγκη πραγματοποίησης αυτού του είδους των ερωτήσεων, λαμβάνοντας υπόψη τις χωροχρονικές ιδιότητες, σε αντίθεση με τα παραδοσιακά συστήματα που διαχειρίζονται μόνο αλφαριθμητικές ιδιότητες.

Ένα σημαντικό χαρακτηριστικό των χωροχρονικών δεδομένων αποτελεί ο ρυθμός μεταβολής των χωρικών ιδιοτήτων στο χρόνο. Επειδή ο κύριος σκοπός μιας βάσης δεδομένων είναι η καλύτερη δυνατή αναπαράσταση του πραγματικού κόσμου,

ο ρυθμός αυτός επηρεάζει το κατά πόσο αποδοτικά θα αναπαρασταθεί το χωροχρονικό περιβάλλον στη βάση δεδομένων. Όπως ήδη αναφέραμε, διακρίνουμε περιβάλλοντα διακριτών αλλαγών και συνεχών αλλαγών. Οι παραδοσιακές βάσεις δεδομένων θεωρούν ότι τα δεδομένα που αποθηκεύονται σε μια βάση παραμένουν σταθερά ως τη στιγμή που θα πραγματοποιηθεί μια πράξη ενημέρωσης. Για παράδειγμα, αν ένα πεδίο που αναπαριστά την τιμή ενός προϊόντος περιέχει τον αριθμό 10, θα παραμείνει 10 έως να πραγματοποιηθεί μια ενημέρωση από το χρήστη. Αν και αυτό το μοντέλο είναι κατάλληλο για πολλές εφαρμογές διακριτών αλλαγών, δεν είναι κατάλληλο για περιβάλλοντα συνεχών αλλαγών [SWCD97, WXCJ98].

Για παράδειγμα, ας θεωρήσουμε μια βάση δεδομένων που περιέχει τις θέσεις κινούμενων αντικειμένων. Η συνεχής πραγματοποίηση ενημερώσεων για τη θέση ενός αντικειμένου δημιουργεί προβλήματα απόδοσης, ενώ το να επιτρέπουμε την ενημέρωση της βάσης μόνο σε συγκεκριμένες χρονικές στιγμές δημιουργεί προβλήματα εγκυρότητας των απαντήσεων σε ερωτήματα. Τα προβλήματα αυτά γίνονται περισσότερο έντονα όταν στη βάση δεδομένων αποθηκεύονται και θέσεις του παρελθόντος. Μια καλύτερη προσέγγιση θα ήταν να αναπαρασταθεί η θέση ενός αντικειμένου ως συνάρτηση του χρόνου. Έτσι, δεν απαιτούνται συνεχείς ενημερώσεις, παρά μόνο όταν αλλάζει η συνάρτηση που περιγράφει την κίνηση ενός αντικειμένου θα πρέπει να πραγματοποιείται ενημέρωση για αυτή την αλλαγή. Επίσης, στην περίπτωση όπου χρησιμοποιούνται συναρτήσεις που περιγράφουν την κίνηση των αντικειμένων, είναι δυνατή η πραγματοποίηση ερωτήσεων που αφορούν τη μελλοντική συμπεριφορά τους. Αντίθετα, σε ένα διακριτό περιβάλλον οι θέσεις των αντικειμένων θεωρούνται ότι παραμένουν ως έχουν έως να πραγματοποιηθεί μια πράξη ενημέρωσης.

Μια χωροχρονική ερώτηση πραγματοποιείται με τη χρήση χωρικών / χρονικών κατηγορημάτων (predicates) και ανακτώνται όλα τα αντικείμενα που ικανοποιούν τα συγκεκριμένα κατηγορήματα. Ένα χωρικό κατηγόρημα ορίζεται με βάση κάποια θέση / σημείο αναφοράς ή μια έκταση (extent), ενώ ένα χρονικό κατηγόρημα με βάση κάποια χρονική στιγμή ή χρονικό διάστημα. Μερικές τυπικές χωροχρονικές ερωτήσεις είναι:

1. Ερωτήσεις *επιλογής*: "βρες όλα τα αντικείμενα σε μια περιοχή, Q, τη χρονική στιγμή t."

2. Ερωτήσεις *γειτονικότητας*: "βρες ποιο αντικείμενο ήταν το κοντινότερο σε ένα δεδομένο σημείο, s, κατά τη διάρκεια του χρονικού διαστήματος, T," ή

"βρες τα κοντινότερα ασθενοφόρα σε μια θέση ατυχήματος τα επόμενα 10 λεπτά."

3. *Αθροιστικές* ερωτήσεις: "βρες πόσα αντικείμενα πέρασαν από την περιοχή Q κατά το χρονικό διάστημα T," ή "βρες το ταχύτερο αντικείμενο που θα περάσει από την περιοχή Q στα επόμενα πέντε λεπτά."

4. Ερωτήσεις *συνένωσης*: "δεδομένων δύο χωροχρονικών σχέσεων R_1 και R_2, βρες ζεύγη αντικειμένων μεταξύ των *εκτάσεων* των οποίων υπήρξε επικάλυψη κατά το χρονικό διάστημα T," ή "βρες ζεύγη αεροσκαφών των οποίων η μεταξύ τους απόσταση θα γίνει μικρότερη από ένα χιλιόμετρο στα επόμενα πέντε λεπτά."

5. Ερωτήσεις *ομοιότητας*: "δεδομένης μιας περιοχής, Q, βρες τις χρονικές στιγμές όπου υπήρχαν περισσότερα από 10 αντικείμενα εντός της Q," ή "βρες τα αντικείμενα των οποίων η συμπεριφορά κίνησης είναι όμοια με αυτή ενός δεδομένου αντικειμένου, o, για ένα συγκεκριμένο χρονικό διάστημα T."

Στις επόμενες υποενότητες, παρουσιάζουμε αναλυτικότερα τις θεμελιώδεις χωροχρονικές έννοιες που θα μας απασχολήσουν καθώς και μοντέλα δεδομένων, γλώσσες ερωτήσεων και δομές ευρετηρίων που έχουν προταθεί έως τώρα στη βιβλιογραφία.

2.1 Θεμελιώδεις χωροχρονικές έννοιες

Στην ενότητα αυτή παρουσιάζουμε έννοιες που αφορούν χωροχρονικές βάσεις δεδομένων. Μια βάση δεδομένων αποτελεί μια συλλογή αντικειμένων που αναπαριστά μέρος του πραγματικού κόσμου. Κάθε αντικείμενο ανήκει σε κλάση αντικειμένων και χαρακτηρίζεται από ένα σύνολο ιδιοτήτων κάθε μια από τις οποίες σχετίζεται με ένα πεδίο τιμών. Έτσι, κάθε αντικείμενο αναπαρίσταται με ένα σύνολο τιμών, κάθε μια από τις οποίες ανήκει στο πεδίο τιμών της αντίστοιχης ιδιότητας του αντικειμένου. Μια βάση δεδομένων ονομάζεται χρονική, χωρική ή χωροχρονική ανάλογα το είδος των εννοιών που διαχειρίζεται: χρονικές, χωρικές ή χωροχρονικές. Στη συνέχεια παρουσιάζουμε συνοπτικά το σύνολο αυτών των εννοιών. Μια περισσότερο αναλυτική περιγραφή μπορεί να βρεθεί στο [PT98].

Χωρικές έννοιες Ο χώρος είναι ένα σύνολο του οποίου τα στοιχεία ονομάζονται σημεία. Συνήθως, στις εφαρμογές, ο χώρος μοντελοποιείται ως υποσύνολο των

$\mathbb{R}^3, \mathbb{Z}^3, \mathbb{R}^2, \mathbb{Z}^2, \mathbb{R}$ και $\mathbb{Z}$. Αντικείμενα των οποίων η θέση στο χώρο έχει σημασία ονομάζονται χωρικά αντικείμενα και λόγω αυτής της συγκεκριμένης αυτής θέσης τους στο χώρο κληρονομούν μέρος των ιδιοτήτων του χώρου (χωρικές ιδιότητες). Οι τελευταίες αναφέρονται σε ολόκληρο το χώρο και μπορούν να αναπαρασταθούν ως χωρικά "στρώματα" (spatial layers), δηλαδή ως συναρτήσεις που αντιστοιχίζουν χωρικά αντικείμενα σε τιμές ιδιοτήτων. Υπάρχουν δύο βασικοί τύποι χωρικών "στρωμάτων": χωρικά "στρώματα" που αναπαριστούν συνεχείς συναρτήσεις, όπως θερμοκρασία και διάβρωση, και χωρικά "στρώματα" που αναπαριστούν διακριτές συναρτήσεις, όπως για παράδειγμα η βλάστηση.

Χρονικές έννοιες Στη βιβλιογραφία έχουν παρουσιαστεί πολλές μέθοδοι για τη μοντελοποίηση του χρόνου. Συνήθως χρησιμοποιείται η έννοια της "γραμμικά ταξινομημένης χρονικής γραμμής", δηλαδή ο χρόνος μοντελοποιείται ως ένα πεπερασμένο σύνολο φυσικών αριθμών. Σε περιβάλλον βάσης δεδομένων δύο είναι οι όψεις του χρόνου που έχουν αποκτήσει ιδιαίτερη σημασία στις έως τώρα εφαρμογές: αυτή του έγκυρου χρόνου (valid time) και αυτή του χρόνου δοσοληψίας (transaction time). Η πρώτη αναφέρεται στο χρόνο όπου αληθεύει ένα γεγονός του πραγματικού κόσμου που μοντελοποιείται στη βάση δεδομένων (π.χ. σεισμός μεγάλου μεγέθους συνέβη στην Αθήνα το 1999), ενώ η δεύτερη αναφέρεται στο χρόνο όπου ένα γεγονός καταγράφεται στη βάση δεδομένων. Τέλος, για την καταγραφή χρονικών πληροφοριών σε μια βάση δεδομένων χρησιμοποιούνται είτε χρονικά σημεία για γεγονότα χωρίς χρονική διάρκεια (π.χ. μια άφιξη πτήσης), είτε χρονικά διαστήματα για καταστάσεις που χαρακτηρίζονται από κάποια χρονική διάρκεια (π.χ. μια ομιλία), είτε χρονικές περίοδοι, δηλαδή σύνολα χρονικών σημείων, διαστημάτων ή συνδυασμών τους.

Χωροχρονικές έννοιες Ο συνδυασμός των παραπάνω εννοιών παράγουν τις έννοιες που χρησιμοποιούνται για την περιγραφή και μοντελοποίηση των χωροχρονικών εφαρμογών. Αναλυτικότερα, έχουμε τους παρακάτω συνδυασμούς:

1. *Χωρικά αντικείμενα σε χρονικά σημεία.* Καταγράφοντας τη θέση ενός χωρικού αντικειμένου σε διάφορα χρονικά σημεία παράγουμε ένα σύνολο από *στιγμιότυπα*. Για παράδειγμα, για ένα κινούμενο αντικείμενο μπορούμε να παράγουμε στιγμιότυπα της κίνησής του, καταγράφοντας τη νέα του θέση σε χρονικά σημεία $t_1, t_2, \ldots, t_n$.

2. *Χωρικά αντικείμενα σε χρονικά διαστήματα.* Καταγράφοντας τη θέση ενός

χωρικού αντικειμένου σε διάφορα χρονικά διαστήματα μπορούμε να περιγράψουμε την *εξέλιξη* του στο χρόνο, δηλαδή τις πιθανές αλλαγές στη θέση του. Αν οι τελευταίες είναι συνεχείς, τότε περιγράφουμε την κίνηση του αντικειμένου, ενώ αν είναι διακριτές, τότε περιγράφουμε τις αλλαγές που συμβαίνουν σε χρονικά διαστήματα. Η δεύτερη περίπτωση μπορεί να θεωρηθεί ως παραγωγή στιγμιότυπων σε χρονικά σημεία που ανήκουν σε συγκεκριμένο χρονικό διάστημα.

3. *Χωρικά αντικείμενα σε χρονικές περιόδους.* Καταγράφοντας τη θέση ενός χωρικού αντικειμένου σε διάφορα χρονικά σημεία της ίδιας χρονικής περιόδου παράγουμε διαφορετικές *εκδοχές* του, δηλαδή ένα σύνολο συσχετιζόμενων μεταξύ τους στιγμιότυπων. Ανάλογα, καταγράφοντας τη θέση ενός χωρικού αντικειμένου σε διάφορα χρονικά διαστήματα της ίδιας χρονικής περιόδου περιγράφουμε συσχετιζόμενες αλλαγές.

4. *Χωρικά στρώματα σε χρονικά σημεία.* Ο συνδυασμός αυτός έχει ως αποτέλεσμα την παραγωγή στιγμιότυπων ενός χωρικού στρώματος. Για παράδειγμα, η καταγραφή της βλάστησης την 5/5/2000, δημιουργεί ένα χάρτη της βλάστησης τη συγκεκριμένη ημέρα.

5. *Χωρικά στρώματα σε χρονικά διαστήματα.* Ο συνδυασμός αυτός περιγράφει την *εξέλιξη* ενός χωρικού στρώματος. Αν η εξέλιξη αυτή είναι συνεχής, τότε περιγράφουμε ένα *φαινόμενο* (π.χ. τον κύκλο ζωής ενός καιρικού φαινομένου), διαφορετικά πρόκειται για ένα σύνολο στιγμιότυπων σε χρονικά σημεία ενός διαστήματος.

6. *Χωρικά στρώματα σε χρονικές περιόδους.* Ο τελευταίος συνδυασμός μπορεί να περιγράψει διαφορετικές *εκδοχές* ενός χωρικού στρώματος, δηλαδή, όπως και στην περίπτωση 3, ένα σύνολο συσχετιζόμενων μεταξύ τους στιγμιότυπων ή συσχετιζόμενων αλλαγών σε χρονικά διαστήματα.

Στην παρούσα εργασία, στοχεύοντας στη μοντελοποίηση κινούμενων αντικειμένων, το ενδιαφέρον μας επικεντρώνεται στις χωροχρονικές έννοιες 1 έως 3.

2.2 Χωροχρονικά μοντέλα και γλώσσες ερωτήσεων για κινούμενα αντικείμενα

Δεδομένης της μεγάλης ερευνητικής δραστηριότητας που πραγματοποιήθηκε στις περιοχές της μοντελοποίησης χωρικών και χρονικών εφαρμογών, οι πρώτες προσπάθειες που αφορούσαν χωροχρονικά μοντέλα δεδομένων βασίστηκαν σε επεκτάσεις μοντέλων που είχαν προταθεί είτε για χωρικά αντικείμενα είτε για χρονικά. Για παράδειγμα, η εργασία [Wor94] αποτελεί γενίκευση προηγούμενης δουλειάς για χωρικά αντικείμενα και την πρώτη πρόταση ενός ενιαίου μοντέλου για χωροχρονικά δεδομένα. Στην εργασία [CG94] παρουσιάζεται ένα παράδειγμα μιας άλλης προσέγγισης, δηλαδή η γενίκευση ενός χρονικού μοντέλου σε χωροχρονικό. Δυστυχώς, όμως, οι δύο αυτές προσπάθειες αφορούσαν διακριτές αλλαγές της θέσης των αντικειμένων και όχι συνεχείς. Αν και υπάρχουν εφαρμογές όπου εμφανίζονται διακριτές αλλαγές (π.χ. εφαρμογές κτηματολογίου), οι περισσότερες χωροχρονικές εφαρμογές αφορούν συνεχείς αλλαγές (π.χ. φυσικά φαινόμενα).

Υπό την οπτική των βάσεων δεδομένων με περιορισμούς, ένα αυστηρό θεωρητικό μοντέλο για διακριτές αλλαγές προτάθηκε στην εργασία [GRS98], ενώ στην [CR99] παρουσιάζεται ένα θεωρητικό πλαίσιο για αντικείμενα που υφίστανται συνεχείς αλλαγές του οποίου αποτελεί υποπερίπτωση το μοντέλο των διακριτών αλλαγών.

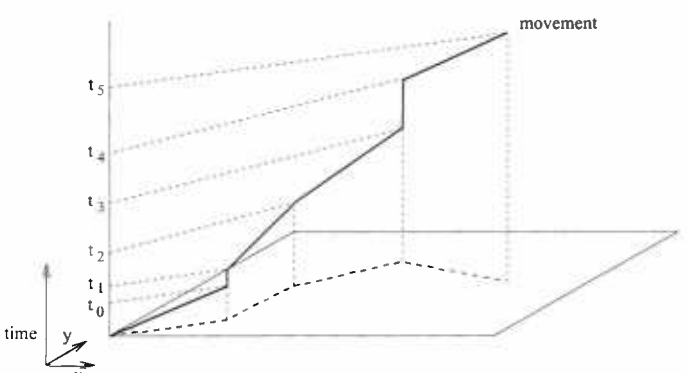
Μια ενοποιημένη θεώρηση των χωρικών και χρονικών ιδιοτήτων των αντικειμένων προς μοντελοποίηση παρουσιάζεται στην εργασία [GBE+00] όπου εισάγεται η έννοια των χωροχρονικών τύπων δεδομένων βασισμένων στους αφαιρετικούς τύπους δεδομένων (abstract data types). Ένα κινούμενο αντικείμενο (σημείο, γραμμή, περιοχή) ορίζεται ως η χρονική εκδοχή ενός στατικού χωρικού αντικειμένου τύπου α και παράγεται από μια συνάρτηση από το πεδίο του χρόνου στο πεδίο τιμών του α.

Στο πεδίο της ανάλυσης απαιτήσεων και της εννοιολογικής μοντελοποίησης, στην εργασία [PT98] παρουσιάζεται ένα σύνολο απαιτήσεων μοντελοποίησης που ως αφετηρία έχει εφαρμογές κτηματολογίου και διαχείρισης δικτύων κοινής ωφέλειας. Στην [PSZ99] προτείνεται το χωροχρονικό μοντέλο MADS (Modeling of Application Data with Spatio-temporal features). Το μοντέλο MADS υποστηρίζει και συνεχείς και διακριτές αλλαγές και επιπλέον επιτρέπει τον ορισμό σχέσεων με με χωροχρονική σημασιολογία, βασιζόμενο στο "9-intersection" μοντέλο [EF91] χωρικών τελεστών καθώς και στο μοντέλο χρονικών τελεστών του Allen [All83].

Στην εργασία [TJ99] επεκτείνεται το μοντέλο οντοτήτων - σχέσεων ER ώστε να περιλαμβάνει χωρικές και χρονικές έννοιες καθώς και συνδυασμούς τους. Το προτεινόμενο μοντέλο ονομάζεται STER (Spatio-Temporal ER).

Επεκτάσεις της SQL έχουν επίσης προταθεί στο παρελθόν, όπως η STSQL [BJS+98] που είναι μια επέκταση της SQL-92 για πολυδιάστατα δεδομένα. Στην πραγματικότητα, όμως, η STSQL δεν αρκεί για να αναπαρασταθούν ερωτήσεις για κινούμενα αντικείμενα, απλά προσθέτει τις έννοιες του χρόνου δοσοληψίας και του έγκυρου χρόνου στην SQL-92. Πρόσφατα οι Erwing και Schneider πρότειναν την STSQL (Spatio-Temporal SQL) που εκμεταλλεύεται την έννοια των αφαιρετικών τύπων δεδομένων (Abstract Data Types).

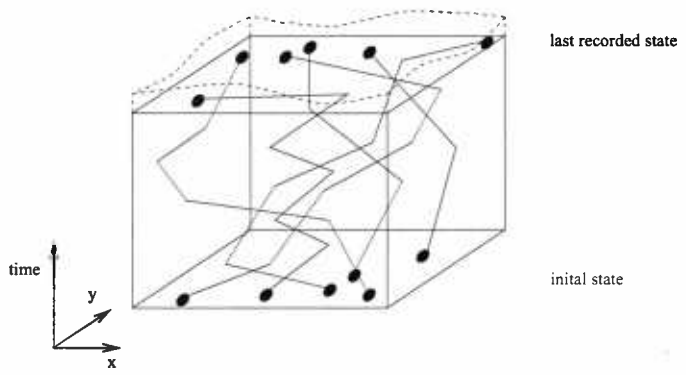
Τα δεδομένα που παράγονται από την κίνηση αντικειμένων σχηματίζουν μια καμπύλη σε τρισδιάστατο χώρο [PT00], όπου δύο από τις τρεις διαστάσεις αντιστοιχούν στο δυσδιάστατο (x-, y-) επίπεδο και η τρίτη διάσταση αντιστοιχεί στο χρόνο. Το πλαίσιο αυτό μπορεί εύκολα να επεκταθεί στις 4 διαστάσεις για αντικείμενα του πραγματικού κόσμου που κινούνται πάνω από την επιφάνεια της γης. Αντί της τρισδιάστατης καμπύλης, σε μια βάση δεδομένων κινούμενων αντικειμένων αποθηκεύουμε και διαχειριζόμαστε μια τρισδιάστατη ακολουθία γραμμικών τμημάτων, τα οποία αναπαριστούν την *τροχιά* του αντικειμένου (σχήματα 1 και 2) και προκύπτουν από τη δειγματοληψία της συνεχούς κίνησής του.



Σχήμα 1: Τροχιά κινούμενου αντικειμένου.

Όπως παρατηρείται και από τα σχήματα 1 και 2, η διαχείριση θέσεων αντικειμένων, που μεταβάλλονται με συνεχή τρόπο, αντιμετωπίζει δύο σημαντικά προβλήματα: τα υπάρχοντα ΣΔΒΔ δεν μπορούν να διαχειριστούν άπειρα σύνολα και έτσι αναγκαστικά υπάρχει απώλεια πληροφορίας, αλλά και τα υπάρχοντα συστήματα

καταγραφής - παρακολούθησης θέσης (GPS και άλλες τεχνολογίες) διαχειρίζονται διακριτή πληροφορία. Ως συνέπεια των παραπάνω, υπάρχουν χρονικά διαστήματα που δεν είναι δυνατό να γνωρίζουμε επακριβώς τη θέση ενός αντικειμένου και για το λόγο αυτό χρησιμοποιούνται τεχνικές παρεμβολής, είτε γραμμικής είτε περισσότερο πολύπλοκες προσεγγίσεις, για να υπολογίσουμε τη θέση του σε χρονικές στιγμές των συγκεκριμένων διαστημάτων. Αυτή η έλλειψη πληροφορίας εισαγάγει την έννοια της αβεβαιότητας, η οποία πρέπει να λαμβάνεται υπόψη για να αποφεύγονται λανθασμένα αποτελέσματα ερωτήσεων. Για παράδειγμα, στην εργασία [PJ99] αναφέρεται η έννοια της περιοχής που περιλαμβάνει όλες τις πιθανές θέσεις ενός αντικειμένου σε μια χρονική στιγμή t μεταξύ συνεχόμενων χρονικών στιγμών δειγματοληψίας t_1 και t_2, $t_1 < t < t_2$.



Σχήμα 2: Σύνολο από τροχιές κινούμενων αντικειμένων.

2.3 Δομές δεικτοδότησης (ευρετήρια) για κινούμενα αντικείμενα

Στη βιβλιογραφία, οι προταθέντες δομές για δεικτοδότηση κινούμενων αντικειμένων ταξινομούνται σε δύο κύριες κατηγορίες. Η πρώτη περιλαμβάνει δομές που μας επιτρέπουν να αποθηκεύουμε τις τρέχουσες θέσεις των αντικειμένων και να πραγματοποιούμε ερωτήσεις που αφορούν μελλοντικές χρονικές στιγμές, ενώ η δεύτερη περιλαμβάνει δομές που μας επιτρέπουν να αποθηκεύουμε τις θέσεις των αντικειμένων στο παρελθόν και να πραγματοποιούμε ερωτήσεις που αφορούν χρονικές στιγμές επίσης του παρελθόντος.

Μια άμεση λύση για τη δεικτοδότηση κινούμενων αντικειμένων αποτελεί ο διαχωρισμός των ιδιοτήτων τους σε αμιγώς χωρικές (σύνολα σημείων ή γραμμικών τμημάτων) και χρονικές (σύνολα χρονικών στιγμών ή διαστημάτων) και η δη-

μιουργία δύο ευρετηρίων, ένα για κάθε σύνολο. Σημειώνουμε ότι αυτή είναι και η μεθοδολογία που ακολουθούμε και για την υλοποίηση της προτεινόμενης γλώσσας ερωτήσεων στην παρούσα εργασία, μιας και είναι μία από της λίγες εφικτές σε περιβάλλον εμπορικού ΣΔΒΔ, σε αντίθεση με την πλειοψηφία των δομών που αναφέρουμε στη συνέχεια, αν και είναι περισσότερο κατάλληλες.

Μια επίσης απλή λύση αποτελεί η θεώρηση του χρόνου ως μιας επιπρόσθετης διάστασης και η αναπαράσταση δυσδιάστατων κινούμενων σημείων ως τρισδιάστατες ακολουθίες γραμμικών τμημάτων (σχήματα 1 και 2). Μια κατάλληλη για το σκοπό αυτό δομή αποτελεί το 3D R-tree [TVS96], το οποίο υποστηρίζει και συνεχείς και διακριτές μεταβολές, αλλά μπορεί να αποθηκεύει μόνο θέσεις παρελθόντος χρόνου. Μια βελτίωση αυτής της δομής αποτελεί το 2+3 R-tree [NST99], μια υβριδική δομή αποτελούμενη από ένα 3D R-tree για τις θέσεις του παρελθόντος και ένα 2D R-tree για τις θέσεις του παρόντος.

Η παρατήρηση ότι ένας χωροχρονικός δείκτης που επιτρέπει μια ιστορική όψη των δεδομένων που δεικτοδοτούνται μπορεί να αναπαρασταθεί ως ένα "δάσος" χωρικών δεικτών (τόσων όσα και τα διαφορετικά στιγμιότυπα που μας ενδιαφέρουν) αποτελεί τη βασική ιδέα του HR-tree (Historical R-tree) [NS98]. Το μειονέκτημα αυτής της δομής αποτελούν οι τεράστιες απαιτήσεις χώρου αποθήκευσης ιδιαίτερα για δεδομένα με μεγάλη συχνότητα μεταβολής στο χρόνο. Πρόταση για τη βελτίωση αυτής της αδυναμίας αποτελεί η δομή HR^+-tree [TP01].

Πρόσφατα έχει προταθεί η δομή TB-tree (Trajectory Bandle - tree) [PJT00] η οποία διαφοροποιείται από το R-tree σε μια βασική του ιδιότητα: αντί της διατήρησης της χωρικής γειτονικότητας των δεδομένων που δεικτοδοτούνται, ελαχιστοποιεί τη διασπορά, σε διαφορετικά φύλλα της δομής, των τμημάτων που ανήκουν στην ίδια τροχιά. Η δομή TB-tree δεικτοδοτεί θέσεις του παρελθόντος και υποστηρίζει συνεχείς μεταβολές.

Στις κυριότερες δομές που έχουν προταθεί για τη δεικτοδότηση παρόντων θέσεων και την υποστήριξη ερωτήσεων που αφορούν μελλοντικές χρονικές στιγμές περιλαμβάνεται το TPR-tree (time parametrized R-tree) [SJLL00]. Η καινοτομία αυτής της δομής συνίσταται στο ότι τα ελάχιστα ορθογώνια[1] (minimum bounding rectangles) είναι συναρτήσεις του χρόνου και όχι στατικά χωρικά αντικείμενα. Επίσης, στην εργασία [SJ02] προτείνεται η δομή R^{EXP}-tree, η οποία υποστηρίζει και την έννοια του *χρόνου λήξης*, δηλαδή δεδομένων που καθίστανται άκυρα μετά το

[1]Τα ελάχιστα ορθογώνια αποτελούν μια βασική έννοια των ιεραρχικών χωρικών δομών αναζήτησης. Για περισσότερες λεπτομέρειες δείτε το άρθρο [Gut84].

πέρας ενός προκαθορισμένου χρόνου.

3 Πλαίσια επέκτασης (extensibility frameworks) σε σύγχρονα αντικειμενοστραφή - σχεσιακά ΣΔΒΔ

Αν και η βασική έρευνα στην περιοχή των χωροχρονικών βάσεων δεδομένων έχει ήδη μια ιστορία μεγαλύτερη της μιας δεκαετίας, οι τρέχουσες εκδόσεις των σύγχρονων εμπορικών ΣΔΒΔ (Oracle, Informix, DB2) δεν υποστηρίζουν αποδοτικά αυτούς τους τύπους εφαρμογών. Ο λόγος μπορεί να είναι η έλλειψη ενός καθολικά αποδεκτού μοντέλου δεδομένων, αλλά και η έλλειψη μιας πολύ σημαντικής χωροχρονικής εφαρμογής (killer application) που θα καθοδηγήσει τις εξελίξεις [SEG+01]. Επιπρόσθετα, δεν έχει υπάρξει μέχρι στιγμής μια ξεκάθαρη εικόνα σχετικά με το τι είδους λειτουργίες διαχείρισης δεδομένων θα πρέπει να προσφέρουν τα συστήματα αυτά, με τις απαιτούμενες λειτουργίες σε επίπεδο διεπαφής χρήστη, με τη γλώσσα ερωτήσεων, καθώς και σχετικά με θέματα όπως επεξεργασία και βελτιστοποίηση ερωτήσεων.

Ωστόσο, στις τελευταίες εκδόσεις ΣΔΒΔ των Oracle, DB2 και Informix παρέχονται *πλαίσια επέκτασης* [SMS+00, CCF+99, BSSJ99], τα οποία επιτρέπουν καλύτερη διαχείριση μη παραδοσιακών δεδομένων, συνδυάζουν τα πλεονεκτήματα ενός μεγάλης κλίμακας εμπορικού συστήματος (backup, recovery, concurrency, multi-user support κ.λπ.), αλλά και υπόκεινται σε κάποιους περιορισμούς που περιγράφουμε στη συνέχεια.

3.1 Περιγραφή, πλεονεκτήματα και μειονεκτήματα των πλαισίων επέκτασης

Αν και τα παραπάνω συστήματα, μέσω του πλαισίου επέκτασης, επιτρέπουν τον ορισμό και κατασκευή μιας νέας δομής δεικτοδότησης και την ενσωμάτωσή της στη γλώσσα ερωτήσεων του συστήματος, δεν παρέχουν πρόσβαση μέσω κάποιας προγραμματιστικής διεπαφής (API, Application Programming Interface) στο φυσικό επίπεδο της βάσης δεδομένων, π.χ. στον διαχειριστή τμημάτων του δίσκου (block manager). Δεδομένης, επίσης, της μη ύπαρξης κάποιου πλαισίου όπως το γενικευμένο δένδρο αναζήτησης[2] (GiST, Generalized Serch Tree [HNP95]) σε αυτά τα συστήματα, οι διαθέσιμες επιλογές για την αποθήκευση της νέας δομής δεικτοδότησης είναι περιορισμένες.

[2]Σημειώνουμε ότι αυτή τη στιγμή το γενικευμένο δένδρο αναζήτησης ενσωματώνεται στο σύστημα POSTGRESQL.

Μια πρώτη επιλογή αποτελεί η αποθήκευση της νέας δομής δεικτοδότησης σε εξωτερικά αρχεία. Αν και η επιλογή αυτή δίνει πολύ ικανοποιητικά αποτελέσματα από άποψη απόδοσης, το μειονέκτημά της συνίσταται στο ότι τα εξωτερικά αρχεία είναι ανεξάρτητα του διαχειριστή δοσοληψιών του ΣΔΒΔ και, επομένως, απαιτείται να υλοποιείται και ένας διαχειριστής τμημάτων δίσκου για τα εξωτερικά αρχεία στην περίπτωση που θα θέλαμε για τη νέα δομή να έχουμε υπηρεσίες, όπως ανάκαμψη, συνδρομικότητα κ.λπ., όμοιες με αυτές που παρέχει το ΣΔΒΔ.

Μια δεύτερη επιλογή αποτελεί η αποθήκευση της δομής δεικτοδότησης ως ένα μεγάλο δυαδικό αντικείμενο (Binary Large Object, BLOB) εντός της βάσης δεδομένων, αλλά και σε αυτή την περίπτωση απαιτούνται εκτενείς προσπάθειες για την υλοποίηση και τη συντήρησή της, κυρίως επειδή σε περιβάλλον πολλαπλών χρηστών τα αντικείμενα BLOB "κλειδώνονται" εξολοκλήρου κατά τη διάρκεια δοσοληψιών.

Για την αποφυγή των μειονεκτημάτων των παραπάνω δύο μεθόδων, τα πλαίσια επέκτασης παρέχουν την επιλογή της αποθήκευσης της δομής δεικτοδότησης σε ένα σχεσιακό πίνακα, πράγμα που αποτελεί και τη συνιστώμενη ενέργεια. Στη συνέχεια, δίνουμε μια σύντομη περιγραφή των πλαισίων επέκτασης και της λειτουργικότητας που προσφέρουν και σκιαγραφούμε τα πλεονεκτήματα και μειονεκτήματα της αποθήκευσης της νέας δομής σε ένα σχεσιακό πίνακα.

Τα πλαίσια επέκτασης, που αρχικά προτάθηκαν από τον Stonebraker [Sto86], όπως έχουν ενταχθεί στα σύγχρονα σχεσιακά - αντικειμενοστραφή ΣΔΒΔ [Ora01a, IBM99, Inf98], επιτρέπουν τη διαχείριση μη παραδοσιακών δεδομένων, με την εφαρμογή της ακόλουθης μεθοδολογίας:

- Ορισμός κατάλληλων τύπων δεδομένων με τη βοήθεια των αφαιρετικών τύπων δεδομένων (Abstract Data Types).
- Ορισμός και υλοποίηση τελεστών μέσω των οποίων θα πραγματοποιούνται νέοι τύποι ερωτήσεων κατάλληλων για το συγκεκριμένο είδος μη παραδοσιακών δεδομένων.
- Ορισμός και υλοποίηση του κατάλληλου τύπου ευρετηρίου για την αποδοτική εκτέλεση των νέων τύπων ερωτήσεων.

Παράδειγμα εφαρμογής της παραπάνω μεθοδολογίας αποτελεί το πακέτο Oracle Spatial [Ora01b], το οποίο υλοποιήθηκε εντός του πλαισίου επέκτασης που προσφέρει το ΣΔΒΔ της Oracle. Το πακέτο αυτό παρέχει νέους τύπους δεδομένων για χωρικά αντικείμενα, κατάλληλους τελεστές για την πραγματοποίηση χωρικών

ερωτήσεων όπως περιοχής, κοντινότερου γείτονα, κ.λπ., καθώς και δύο τύπους χωρικών ευρετηρίων.

Ο ορισμός των νέων τύπων δεδομένων πραγματοποιείται με DDL (Data Definition Language) προτάσεις του τύπου:

```
CREATE TYPE NEWTYPE AS OBJECT (X XTYPE, Y YTYPE)
```

Οι τελεστές που χρησιμοποιούνται για τον ορισμό νέων τύπων ερωτήσεων, ορίζονται επίσης με DLL προτάσεις και η υλοποίησή τους πραγματοποιείται με τη βοήθεια αποθηκευμένων συναρτήσεων (stored functions). Οι τελεστές είναι δυνατό να υπολογίζονται μέσω νέων τύπων ευρετηρίων που ενσωματώνονται στο ΣΔΒΔ.

Η δημιουργία ενός νέου τύπου ευρετηρίου αποτελεί περισσότερο πολύπλοκη διαδικασία. Περιλαμβάνει την υλοποίηση αποθηκευμένων συναρτήσεων για την εκκίνηση και τον τερματισμό της χρήσης του ευρετηρίου, για τη μετακίνηση στις εγγραφές που αποτελούν απάντηση σε ένα ερώτημα, καθώς και για την πραγματοποίηση εισαγωγών, διαγραφών και ενημερώσεων στο ευρετήριο. Οι λειτουργίες αυτές είναι συμπληρωματικές προς τις συναρτήσεις που υλοποιούν τους νέους τελεστές που αναφέραμε προηγουμένως. Επίσης, παρέχεται η δυνατότητα υλοποίησης συναρτήσεων για τον υπολογισμό της επιλεξιμότητας και του κόστους επεξεργασίας μιας ερώτησης. Η δημιουργία και διατήρηση στατιστικών μετα-δεδομένων και ιστογραμμάτων πραγματοποιείται μέσω των καθιερωμένων διαχειριστικών SQL εντολών. Έτσι, ο βελτιστοποιητής ερωτήσεων (query optimizer) κατά την επιλογή πλάνου εκτέλεσης που αφορά πίνακες όπου έχουν οριστεί και νέοι τύποι ευρετηρίων, μπορεί να επιλέξει να προσπελάσει τα δεδομένα του πίνακα, είτε με τη βοήθεια των ευρετηρίων που παρέχει το ίδιο το ΣΔΒΔ είτε με τη βοήθεια των τύπων ευρετηρίων που έχουν οριστεί από το χρήστη του ΣΔΒΔ. Με αυτό τον τρόπο διατηρείται το παραδοσιακό συντακτικό της SQL και, για παράδειγμα, μπορούμε να γράψουμε:

```
SELECT * FROM EARTHQUAKES
WHERE INSIDE(EARTHQUKES.EPICENTER, REGION)
```

Όπως ήδη περιγράψαμε, με τη χρήση των πλαισίων επέκτασης συνιστάται η αποθήκευση της δομής σε ένα σχεσιακό πίνακα. Ειδικότερα, η δομή αποθηκεύεται σε ένα ιδιαίτερο είδος πίνακα που ονομάζεται index-organized table [SDF+00] και οι εγγραφές αυτού του πίνακα οργανώνονται με έναν από τους συνήθεις τύπους

ευρετηρίων διαθέσιμων σε σχεσιακά συστήματα (π.χ. B+-tree). Από τα παραδείγματα χρήσης του πλαισίου επέκτασης που έχουν υπάρξει ως σήμερα, διακρίνονται δύο "σχήματα χρήσης":

"Σχήμα θέσεως" Σε αυτή την περίπτωση, κάθε γραμμή του πίνακα όπου αποθηκεύεται ο δείκτης αντιστοιχίζεται ακριβώς σε μία γραμμή του πίνακα δεδομένων που δεικτοδοτούνται μέσω ενός μετασχηματισμού του συνόλου των γραμμών του πίνακα δεδομένων σε ένα σύνολο του οποίου τα στοιχεία μπορούν να διαταχθούν γραμμικά. Δηλαδή, αν τα δεδομένα προς δεικτοδότηση είναι πολυδιάστατα και δεν είναι δυνατή η διάταξή τους, τα αντιστοιχίζουμε με κάποιο τρόπο σε ένα μονοδιάστατο χώρο όπου μπορεί να οριστεί μια γραμμική διάταξη. Π.χ. τα σημεία σε δυσδιάστατο ευκλείδειο χώρο είναι δυνατό να μετασχηματιστούν σε μονοδιάστατο χώρο με τη βοήθεια μιας καμπύλης Z ή Hilbert και συνεπώς να διαταχθούν γραμμικά. Αφού πραγματοποιηθεί ο κατάλληλος μετασχηματισμός, το σύνολο που προκύπτει, του οποίου τα στοιχεία μπορούν να διαταχθούν γραμμικά, οργανώνεται με ένα δείκτη που παρέχεται από το ΣΔΒΔ (π.χ. ένα B+-tree). Παράδειγμα ενός ευρετηρίου που μπορεί να οριστεί με τη χρήση του "σχήματος θέσεως" αποτελεί το Linear Quadtree [Sam90].

"Σχήμα πλοήγησης" Σε αυτή την περίπτωση, οι εγγραφές του πίνακα όπου αποθηκεύεται ο δείκτης εξομοιώνουν μια ιεραρχική δομή αναζήτησης, δηλαδή ένα σύνολο εγγραφών αντιστοιχούν στα περιεχόμενα της ρίζας της ιεραρχικής δομής, άλλα σύνολα εγγραφών αντιστοιχούν στα περιεχόμενα των παιδιών της ρίζας κ.ο.κ. και επομένως μια εγγραφή του πίνακα δεδομένων σχετίζεται με περισσότερες από μία εγγραφές του πίνακα όπου αποθηκεύεται ο δείκτης. Η πλοήγηση εντός αυτής της ιεραρχικής δομής πραγματοποιείται με τη βοήθεια ενός συνόλου τεχνητών κόμβων που οργανώνονται με ένα τύπο ευρετηρίου που παρέχεται από το ΣΔΒΔ (π.χ. ένα B+-tree). Παράδειγμα ενός ευρετηρίου που μπορεί να οριστεί με τη χρήση του "σχήματος θέσεως" αποτελεί το Relational X-tree [BBKM99].

Για τους τύπους ευρετηρίων που παρέχει το ίδιο το ΣΔΒΔ τα δομικά δεδομένα (structural data) διαχειρίζονται ξεχωριστά από το περιεχόμενο (content data) του ευρετηρίου. Για παράδειγμα, κατά τη διάσπαση ενός κόμβου ενός B+-tree που υπερχειλίζει (μεταβολή των δομικών δεδομένων) μπορούν να γίνονται παράλληλες

(concurrent) ενημερώσεις των υποδένδρων του κόμβου υπό διάσπαση (μεταβολή περιεχομένου). Αν και το "σχήμα πλοήγησης" αποτελεί ένα άμεσο τρόπο για την εξομοίωση μιας ιεραρχικής δομής αναζήτησης, δεν μπορεί να έχει αυτού του είδους τα πλεονεκτήματα, αφού το σύνολο των τεχνητών κόμβων που χρησιμοποιείται για την πλοήγηση εντός της ιεραρχικής δομής (δηλαδή τα δομικά δεδομένα της δομής που εξομοιώνουμε) διαχειρίζονται από το ΣΔΒΔ ως δεδομένα χρήστη και επομένως όταν "κλειδώνονται" για λόγους συνδρομικότητας δεν είναι δυνατό να πραγματοποιούμε ταυτόχρονες δομικές μεταβολές και μεταβολές περιεχομένου.

Το "σχήμα θέσεως" δεν έχει το παραπάνω μειονέκτημα, αφού κάθε γραμμή του πίνακα όπου αποθηκεύεται ο δείκτης αντιστοιχίζεται ακριβώς σε μία γραμμή του πίνακα δεδομένων και δεν υπάρχει επιπλέον επιβάρυνση της απόδοσης (overhead).

Στη συνέχεια παρουσιάζουμε δύο επεκτάσεις λειτουργικότητας σχεσιακών - αντικειμενοστραφών ΣΔΒΔ που μας ενδιαφέρουν ιδιαίτερα, αφού τις χρησιμοποιούμε ως εργαλεία για την υλοποίηση της προτεινόμενης γλώσσας ερωτήσεων. Η πρώτη αντιστοιχεί στο πακέτο Oracle Spatial για τη διαχείριση χωρικών δεδομένων και η δεύτερη αφορά τη διαχείριση χρονικών διαστημάτων με τη βοήθεια του RI-tree.

3.2 Η επέκταση Oracle Spatial για διαχείριση χωρικών δεδομένων

Η Oracle Spatial παρέχει τον τύπο χωρικών δεδομένων *SDO_GEOMETRY*. Επίσης, έχουν υλοποιηθεί συναρτήσεις και τελεστές που ενεργούν πάνω στον τύπο *SDO_GEOMETRY* και επιτρέπουν την πραγματοποίηση χωρικών ερωτήσεων καθώς και ερωτήσεων που αφορούν και αλφαριθμητικά και χωρικά δεδομένα. Η αποδοτική εκτέλεση των ερωτήσεων υποστηρίζεται από κατάλληλες δομές δεικτοδότησης και επεκτάσεις του βελτιστοποιητή ερωτήσεων που υλοποιούνται με τους μηχανισμούς που περιγράψαμε στην προηγούμενη ενότητα. Στη συνέχεια παραθέτουμε περισσότερες λεπτομέρειες σχετικά με το μοντέλο δεδομένων, τα νέα είδη ερωτήσεων που παρέχονται καθώς και για τη δυνατότητα χωρικής δεικτοδότησης.

3.2.1 Το μοντέλο δεδομένων

Τα δεδομένα τοποθετούνται σε πίνακες με στήλες που αντιστοιχούν στους συνήθεις αλφαριθμητικούς τύπους δεδομένων καθώς και μια στήλη που αντιστοιχεί στον τύπο δεδομένων *SDO_GEOMETRY*. Ένας πίνακας αυτού του είδους αποτελεί ένα *θεματικό χάρτη*, δηλαδή μια συλλογή αντικειμένων του ίδιου τύπου, ένα για κάθε γραμμή του πίνακα. Η προβολή του πίνακα ως προς τη στήλη των χωρικών δε-

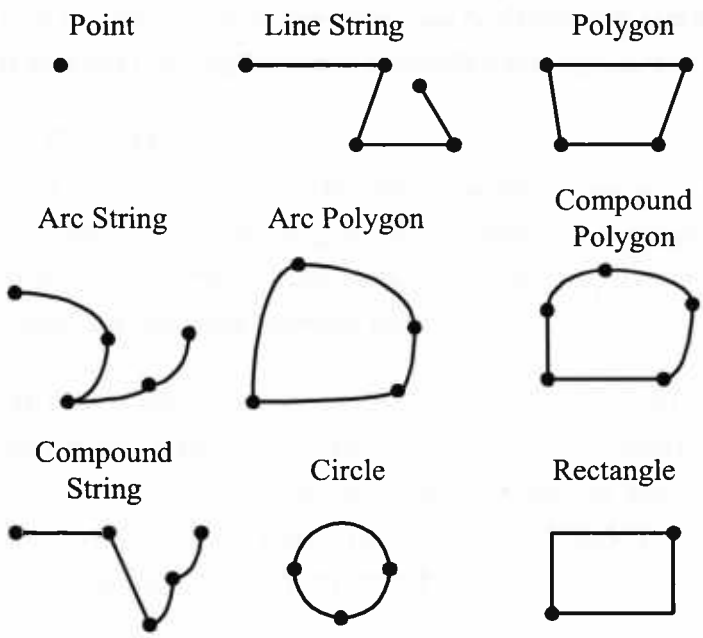
δομένων αποτελεί ένα χωρικό στρώμα (ενότητα 2, σελίδα 8). Για παράδειγμα, ένας πίνακας "σεισμικών δονήσεων" μπορεί να οριστεί ως εξής:

```
CREATE TABLE QUAKES(id NUMBER, magnitude NUMBER,
epicenter MDSYS.SDO_GEOMETRY)
```

Ένα αντικείμενο τύπου *SDO_GEOMETRY* μπορεί να είναι είτε ένα απλό στοιχείο είτε μια διατεταγμένη λίστα απλών στοιχείων, όπου απλά στοιχεία αποτελούν οι τύποι *σημείο, γραμμικό τμήμα, τόξο και πολύγωνο*, ενώ λίστες στοιχείων αυτών των τύπων αποτελούν σύνθετα αντικείμενα. Για παράδειγμα ένα σύμπλεγμα νήσων μοντελοποιείται ως μια διατεταγμένη λίστα πολυγώνων, ενώ ένα νησί όπου υπάρχει μια λίμνη μοντελοποιείται ως μια λίστα δύο πολυγώνων, ένα για το εξωτερικό περίγραμμα και ένα για το εσωτερικό. Τα σύνθετα αντικείμενα είναι δυνατό να αποτελούνται από σύνολο διαφορετικών μεταξύ τους απλών στοιχείων. Συνολικά, παρέχεται υποστήριξη των παρακάτω τύπων δεδομένων (απεικονίζονται στις δύο διαστάσεις στο σχήμα 3):

- Σημεία και ομάδες (clusters) σημείων.
- Γραμμών αποτελούμενες από γραμμικά τμήματα (polylines - line strings).
- Πολύγωνα n-σημείων.
- Γραμμών αποτελούμενες από τόξα (arc line strings).
- Πολύγωνα τόξων (arc polygons).
- Σύνθετα πολύγωνα (compound polygons).
- Σύνθετες γραμμές (compound line strings).
- Κύκλοι.
- Ορθογώνια.

Για να εισάγουμε δεδομένα σε ένα πίνακα, χρησιμοποιούμε την καθιερωμένη εντολή insert. Για παράδειγμα, σε ένα πίνακα, HS όπου αποθηκεύονται τμήματα αυτοκινητοδρόμων με δύο στήλες highwayid αριθμητικού τύπου και highwaysegment χωρικού τύπου, μπορούμε να χρησιμοποιήσουμε την εντολή:



Σχήμα 3: Τύποι δεδομένων διαθέσιμοι από το Oracle Spatial Data Cartridge

```
INSERT INTO HS VALUES (1,
       MDSYS.SDO_GEOMETRY(2002, NULL, NULL,
       MDSYS.SDO_ELEM_INFO_ARRAY(1, 2, 1),
       MDSYS.SDO_ORDINATE_ARRAY(2,2,3,5,6,6,7,8)))
```

Τα ορίσματα του MDSYS.SDO_GEOMETRY αναπαριστούν το χωρικό αντικείμενο που εισάγεται. Το πρώτο όρισμα, 2002, είναι το αναγνωριστικό του τύπου δεδομένων που εισάγεται, δηλαδή ένας σύνθετος τύπος ακολουθίας γραμμικών τμημάτων στις δύο διαστάσεις. Τα ορίσματα δύο και τρία (με τιμή NULL) αντιστοιχούν σε ένα αναγνωριστικό του χωρικού συστήματος αναφοράς που χρησιμοποιείται (π.χ. η τιμή 8037 αντιστοιχεί σε γεωγραφικές συντεταγμένες) και σε ένα σημείο-ετικέτα, αντίστοιχα. Το τέταρτο όρισμα, MDSYS.SDO_ELEM_INFO_ARRAY(1, 2, 1) δηλώνει ότι οι τα ζεύγη συντεταγμένων που ορίζουν την ακολουθία των γραμμικών τμημάτων ξεκινούν από τη θέση 1 του πίνακα MDSYS.SDO_ORDINATE_ARRAY, ότι το σύνθετο στοιχείο που εισάγεται αποτελείται από απλά στοιχεία τύπου 2, δηλαδή από γραμμικά τμήματα, και τέλος ότι τα ζεύγη σημείων που εμφανίζονται στον πίνακα MDSYS.SDO_ORDINATE_ARRAY συνδέονται με ευθείες γραμμές

(κωδικός 1). Το τελευταίο όρισμα αποτελεί μια διατεταγμένη λίστα των κορυφών (ζεύγη συντεταγμένων) που ορίζουν την ακολουθία των γραμμικών τμημάτων.

3.2.2 Χωρικοί τελεστές

Η Oracle Spatial παρέχει 4 τελεστές, οι οποίοι υπολογίζονται με τη βοήθεια χωρικών ευρετηρίων και για τους οποίους έχουν υλοποιηθεί οι κατάλληλες επεκτάσεις ώστε να είναι διαθέσιμες στο βελτιστοποιητή πληροφορίες σχετικά με το κόστος υπολογισμού τους και την επιλεξιμότητά τους:

- SDO_FILTER ο οποίος προσδιορίζει αν ένα χωρικό αντικείμενο σχετίζεται με κάποιο άλλο. Αποτελεί το πρώτο φίλτρο για τον έλεγχο ύπαρξης χωρικών σχέσεων (εντοπίζει ζεύγη χωρικών αντικειμένων που είναι πιθανό να μην έχουν κοινά σημεία), ενώ ο τελεστής SDO_RELATE μπορεί να χρησιμοποιηθεί ως φίλτρο μεγαλύτερης ακρίβειας.

- SDO_NN ο οποίος εντοπίζει τους κοντινότερους γείτονες.

- SDO_RELATE ο οποίος προσδιορίζει με ακρίβεια αν δύο χωρικά αντικείμενα συσχετίζονται με ένα από τους τρόπους που καθορίζονται από το "9-intersection" μοντέλο [EF91].

- SDO_WITHIN_DISTANCE ο οποίος καθορίζει αν δύο χωρικά αντικείμενα απέχουν προκαθορισμένη ευκλείδεια απόσταση.

Για παράδειγμα, για να εντοπίσουμε τους νομούς που συνορεύουν με το νομό Αττικής χρησιμοποιούμε την ερώτηση:

```
SELECT p1.name
FROM   Perfecture p1, Perfecture p2
WHERE  p2.name = 'ATTIKH' AND
       MDSYS.SDO_RELATE(p1.geometry, p2.geometry,
                        'mask = TOUCH') = 'TRUE'
```

Επίσης, παρέχεται και ένα σύνολο βοηθητικών συναρτήσεων οι οποίες δεν χρησιμοποιούνται για να επιλέγουμε γραμμές από πίνακες, αλλά για να πραγματοποιούμε γεωμετρικούς υπολογισμούς και η πλειοψηφία τους δέχεται ένα ή δύο χωρικά αντικείμενα ως ορίσματα.

- SDO_GEOM.RELATE Προσδιορίζει την χωρική σχέση μεταξύ δύο αντικειμένων.
- SDO_GEOM.SDO_AREA Υπολογίζει το εμβαδόν ενός αντικειμένου δύο διαστάσεων.
- SDO_GEOM.SDO_BUFFER Δημιουργεί ένα πολύγωνο - buffer γύρω από ένα αντικείμενο.
- SDO_GEOM.SDO_CENTROID Επιστρέφει το κέντρο ενός πολυγώνου.
- SDO_GEOM.SDO_CONVEXHULL Επιστρέφει το ελάχιστο κυρτό πολύγωνο)convex hull) ενός αντικειμένου.
- SDO_GEOM.SDO_DIFFERENCE Επιστρέφει ένα χωρικό αντικείμενο που αντιστοιχεί στην τοπολογική διαφορά δύο χωρικών αντικειμένων.
- SDO_GEOM.SDO_DISTANCE Υπολογίζει την απόσταση μεταξύ δύο χωρικών αντικειμένων.
- SDO_GEOM.SDO_INTERSECTION Επιστρέφει ένα χωρικό αντικείμενο που αντιστοιχεί στην τοπολογική τομή δύο χωρικών αντικειμένων.
- SDO_GEOM.SDO_LENGTH Υπολογίζει την περίμετρο ενός χωρικού αντικειμένου.
- SDO_GEOM.SDO_POINTONSURFACE Επιστρέφει ένα σημείο που εγγυημένα βρίσκεται πάνω στην επιφάνεια ενός πολυγώνου.
- SDO_GEOM.SDO_UNION Επιστρέφει ένα χωρικό αντικείμενο που αντιστοιχεί στην τοπολογική ένωση δύο χωρικών αντικειμένων.
- SDO_GEOM.SDO_XOR Επιστρέφει ένα χωρικό αντικείμενο που αντιστοιχεί στην τοπολογική συμμετρική διαφορά (XOR) δύο χωρικών αντικειμένων.
- SDO_GEOM.VALIDATE_GEOMETRY Καθορίζει εάν ένα χωρικό αντικείμενο είναι έγκυρο.
- SDO_GEOM.VALIDATE_LAYER Καθορίζει εάν το σύνολο των χωρικών αντικειμένων που είναι αποθηκευμένα σε μια στήλη είναι έγκυρο.

- SDO_GEOM.WITHIN_DISTANCE Καθορίζει αν δύο χωρικά αντικείμενα απέχουν προκαθορισμένη ευκλείδεια απόσταση.

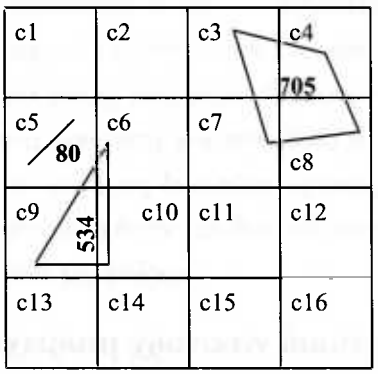
3.2.3 Χωρική δεικτοδότηση και επεξεργασία ερωτήσεων

Η Oracle Spatial δίνει τη δυνατότητα χρήσης "λογικών" (δηλαδή προσομοιώσεων) R-trees και Quad trees ως ευρετήρια. Η εύρεση απαντήσεων στις ερωτήσεις πραγματοποιείται σε δύο βήματα. Κατά το πρώτο βήμα (filter step) ένα υπερσύνολο της απάντησης εντοπίζεται γρήγορα, χρησιμοποιώντας προσεγγίσεις των χωρικών δεδομένων που δεικτοδοτούνται (π.χ. ελάχιστα ορθογώνια, minimum bounding rectangles). Κατά το δεύτερο βήμα (refinment step), εφαρμόζονται τεχνικές που προσδιορίζουν με ακρίβεια τις σχέσεις μεταξύ των χωρικών αντικειμένων.

Για τον τρόπο υλοποίησης των R-trees στην υπάρχουσα βιβλιογραφία δεν υπάρχουν πολλές λεπτομέρειες. Πρόκειται για μια δομή που υλοποιείται με βάση το "σχήμα πλοήγησης" (ενότητα 3.1, σελίδα 18). Αν και μπορούν να οριστούν αντικείμενα έως και διάστασης 4, στην τρέχουσα έκδοση (9i) μόνο οι δύο πρώτες διαστάσεις χρησιμοποιούνται για τη δεικτοδότηση.

Για το "λογικό" Quad-tree, που υλοποιείται με τη βοήθεια του "σχήματος θέσεως" (ενότητα 3.1, σελίδα 18), παραθέτουμε περισσότερες λεπτομέρειες από το [RSV02]. Ο χώρος των συντεταγμένων αναλύεται σε 4^n σταθερού μεγέθους κελιά (cells) τα οποία οργανώνονται σε μια z-order διάταξη. Η ανάλυση αυτή είναι όμοια με αυτή που πραγματοποιεί το Quad-tree, ώστε όλα τα φύλλα να βρίσκονται στο επίπεδο n. Κάθε χωρικό αντικείμενο προσεγγίζεται με το ελάχιστο δυνατό σύνολο κελιών με τα οποία υπάρχει επικάλυψη.

Τα κελιά ταξινομούνται με βάση ένα κλειδί. Η δομή δεικτοδότησης περιλαμβάνει μόνο τα κελιά που επικαλύπτονται με τουλάχιστον ένα από τα αντικείμενα που δεικτοδοτούνται. Το ευρετήριο ενός πίνακα A είναι ένας πίνακας, $A\_SDOINDEX$ με δύο στήλες: SDO_CODE (το κλειδί του κελιού) και SDO_GID (αναγνωριστικό του συσχετιζόμενου χωρικού αντικειμένου). Κάθε εγγραφή στο ευρετήριο αντιστοιχεί σε ένα χωρικό αντικείμενο του οποίου η προσέγγιση περιλαμβάνει το συγκεκριμένο κελί. Ο πίνακας-ευρετήριο οργανώνεται με ένα B+-tree. Η παράμετρος n, που καθορίζει τη λεπτομέρεια της ανάλυσης, δίνεται από το χρήστη. Στο σχήμα 4 παρουσιάζεται ένα παράδειγμα για $n = 2$ με τρία χωρικά αντικείμενα. Το πολύγωνο με αναγνωριστικό 705 προσεγγίζεται με τα κελιά c3, c4, c7 και c8. Το γραμμικό τμήμα 80 προσεγγίζεται με το κελί c5. Το τρίγωνο προσεγγίζεται με τα κελιά c5, c6, c9 και c10.



A_SDOINDEX

SDO_CODE	SDO_GID
c3	705
c4	705
c5	80
c5	534
c6	534
c7	705
c8	705
c9	534
c10	534

Σχήμα 4: Oracle Spatial Quad-tree indexing

Ενδεικτικά, παρουσιάζουμε τα σχήματα επεξεργασίας για τους δύο κυριότερους τύπους ερωτήσεων: ερωτήσεις περιοχής (range-window queries) και συνένωσης (spatial join)

Ερωτήσεις περιοχής Ας υποθέσουμε ότι θέλουμε να εκτελεστεί η ερώτηση "βρες όλα τα αντικείμενα του πίνακα A που επικαλύπτονται με το παράθυρο w". Για τον πίνακα A υπάρχει το ευρετήριο $\_SDOINDEX$ με δύο στήλες: $SDO\_CODE(cellkey)$ και $SDO\_GID(geometryelementkey)$. Πάνω στη στήλη SDO_CODE έχει οριστεί ένα B+-tree. Το παράθυρο w αναλύεται σε ένα σύνολο κελιών σταθερού μεγέθους. Έπειτα, για κάθε ένα από τα κελιά, έστω c, αναζητούμε στον πίνακα $\_SDOINDEX$ τα αντικείμενα των οποίων η προσέγγιση περιλαμβάνει κάποιο από τα c, δηλαδή[3]:

```
SELECT DISTINCT I.SDO_GID
FROM   A_SDO_INDEX I, w.TEMP_INDEX J
WHERE  I.SDO_CODE = J.SDO_CODE
```

Συνένωση Ας υποθέσουμε ότι μια ερώτηση αφορά δύο πίνακες A και B και ότι χρησιμοποιείται ο τελεστής SDO_RELATE. Αν δεν υπάρχει χωρικό ευρετήριο σε κανένα από τους δύο πίνακες, η ερώτηση έχει μεγάλο κόστος, αφού προσπελαύνεται σειριακά ο πίνακας A και για κάθε εγγραφή αυτού

[3]Στην πραγματικότητα η ερώτηση είναι περισσότερο πολύπλοκη, αλλά για λόγους απλότητας στην περιγραφή παρουσιάζουμε μια λιγότερο σύνθετη εκδοχή.

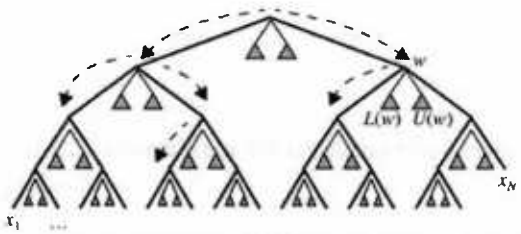
πραγματοποιείται και μια προσπέλαση σε όλες τις εγγραφές του B. Αν υπάρχει χωρικό ευρετήριο σε ένα από τους πίνακες, τότε για κάθε εγγραφή του πίνακα χωρίς ευρετήριο δημιουργείται ένα παράθυρο w και πραγματοποιείται μια ερώτηση για τον άλλο πίνακα όπως στην προηγούμενη περίπτωση. Αν και για τους δύο πίνακες υπάρχει ευρετήριο, τα δύο ευρετήρια μπορούν να συγχωνευθούν αποδοτικά διατηρώντας μόνο τα κελιά που υπάρχουν και στα δύο ευρετήρια.

3.3 Διαχείριση χρονικών διαστημάτων με το RI-tree

Παράδειγμα επέκτασης των δυνατοτήτων ενός σχεσιακού-αντικειμενοστραφούς ΣΔΒΔ αποτελεί και η εργασία [KPS00], όπου προτείνεται μια αντιστοίχηση σε σχεσιακό σχήμα της ιεραρχικής δομής αναζήτησης Interval Tree και κατά συνέπεια αποτελεί μια πρωτότυπη πρόταση για τη διαχείριση χρονικών διαστημάτων. Στη συνέχεια, περιγράφουμε την προτεινόμενη αναπαράσταση - μοντέλο δεδομένων για χρονικά διαστήματα, την υλοποίηση της δομής δεικτοδότησης και τους αλγορίθμους εισαγωγής και αναζήτησης.

Η δομή του Edelsbrunner [Ede80, PS93] είναι μια βέλτιστη δομή για χρονικά διαστήματα και αποτελείται από τρεις επιμέρους δομές (σχήμα 5). Η πρώτη αντιστοιχεί σε ένα δυαδικό δένδρο αναζήτησης που οργανώνει τις τιμές των άκρων των διαστημάτων. Κάθε ένας εσωτερικός κόμβος w σχετίζεται με δύο λίστες $L(w)$ και $U(w)$ οι οποίες αποτελούν τη δεύτερη επιμέρους δομή. Οι $L(w)$ και $U(w)$ αντιστοιχούν σε ταξινομημένες λίστες των άνω και κάτω ορίων των διαστημάτων που σχετίζονται με τον κόμβο w. Ένα διάστημα (l, u) καταχωρείται στον πρώτο κόμβο w με τον οποίο υπάρχει επικάλυψη, δηλαδή στον πρώτο κόμβο w, που συναντούμε καθώς διατρέχουμε το δένδρο από τη ρίζα προς τα φύλλα, για τον οποίο ισχύει $l \leq w \leq u$. Η τρίτη επιμέρους δομή αντιστοιχεί σε ένα επιπλέον δυαδικό δένδρο που αφορά τους κόμβους w των οποίων οι λίστες $L(w)$ και $U(w)$ δεν είναι κενές.

Στο άρθρο [KPS00] προτείνεται ένας αποτελεσματικός τρόπος για την ενσωμάτωση της παραπάνω δομής σε σχεσιακές-αντικειμενοστραφείς βάσεις δεδομένων. Η υλοποίηση της πρώτης επιμέρους δομής πραγματοποιείται εικονικά ενώ η δεύτερη και τρίτη αντιστοιχίζονται σε ένα σχεσιακό σχήμα ως ακολούθως: Έστω $L(w) = \{l_1, \ldots, l_{n_w}\}$ η λίστα των κάτω ορίων των n_w διαστημάτων που είναι καταχωρημένα στον κόμβο w. Η ίδια πληροφορία μπορεί να αναπαρασταθεί από το σύνολο των πλειάδων $\{(w, l_1), \ldots, (w, l_{n_w})\}$, των οποίων η ένωση αποτελεί τη σχέση $(node, lower)$. Με ανάλογο τρόπο προκύπτει και η σχέση $(node, upper)$ η



Σχήμα 5: Η δομή του Edelsbrunner για δεικτοδότηση χρονικών διαστημάτων.

οποία σε συνδυασμό με την $(node, lower)$ αποτελούν τη δεύτερη επιμέρους δομή.

Οι δύο παραπάνω σχέσεις μπορούν να οργανωθούν αποδοτικά με σύνθετους δείκτες σε μια σχεσιακή βάση δεδομένων. Εφόσον οι δομές δεικτοδότησης διαχειρίζονται μόνο τους μη κενούς κόμβους, η τρίτη επιμέρους δομή δεν είναι αναγκαίο να υλοποιηθεί. Έτσι, για τη δεύτερη και τρίτη επιμέρους δομή, αρκεί να υλοποιηθεί το σχήμα $(node, lower, upper, id)$, που αποτελεί τον τρόπο αναπαράστασης των χρονικών διαστημάτων, το οποίο γράφεται ως εξής με τη χρήση DLL προτάσεων:

```
CREATE TABLE Intervals(node int, lower int, upper int, id int);
CREATE INDEX lowerIndex ON Intervals(node, lower);
CREATE INDEX upperIndex ON Intervals(node, upper);
```

Όπως ήδη έχουμε αναφέρει, η πρώτη δομή δεν απαιτείται να υλοποιηθεί. Η εύρεση του κόμβου καταχώρησης για την εισαγωγή ενός διαστήματος καθώς και διαπέραση του δένδρου για την εύρεση της απάντησης σε μια ερώτηση επικάλυψης μπορούν να πραγματοποιούνται μόνο με αριθμητικούς υπολογισμούς.

Η δομή υποστηρίζει τη δεικτοδότηση διαστημάτων στην περιοχή $[1, 2^h - 1]$ για κάποιο $h \geq 0$. Η εύρεση του κόμβου καταχώρησης πραγματοποιείται με τη συνάρτηση:

```
FUNCTION int forkNode(int lower, int upper) {
   int forknode = root;
   for (int step = node / 2; step >= 1; step /= 2)
      if (upper < node) node -= step;
      elsif (node < lower) node += step;
      else break;
```

```
    return node;
}
```

Η μόνη πληροφορία που απαιτείται να αποθηκευθεί είναι η τιμή της ρίζας που αρχικά τίθεται ίση με $2^{(h-1)}$.

Ο αλγόριθμος που πραγματοποιεί αναζήτηση των χρονικών διαστημάτων που επικαλύπτονται με ένα δεδομένο χρονικό διάστημα $(lower, upper)$ αποτελείται από τρεις φάσεις:

1. Διατρέχουμε το δένδρο από τη ρίζα έως τον κόμβο πριν από τον κόμβο καταχώρησης. Κάθε κόμβος w, πάνω σε αυτό το μονοπάτι, βρίσκεται είτε αριστερά είτε δεξιά του διαστήματος αναζήτησης. Αν $w < lower$, τότε τα διαστήματα (l, u) που έχουν καταχωρηθεί στον κόμβο w επικαλύπτονται με το διάστημα της ερώτησης ακριβώς εάν $lower < u$. Για να βρεθούν αυτά τα r_w διαστήματα, διατρέχουμε την ταξινομημένη λίστα $U(w)$ σε χρόνο $O(r_w)$. Ανάλογα διατρέχουμε τη λίστα $L(w)$ στη συμμετρική περίπτωση όπου $upper < w$.

2. Διατρέχουμε το δένδρο από τον κόμβο καταχώρησης έως τον κόμβο με την κοντινότερη τιμή στο άκρο $lower$. Για κάθε κόμβο πάνω σε αυτό το μονοπάτι υπάρχουν δύο περιπτώσεις: αν $w < lower$, διατρέχουμε τη λίστα $U(w)$ για να βρεθούν τα διαστήματα επικάλυψης, διαφορετικά το διάστημα αναζήτησης επικαλύπτεται με όλα τα διαστήματα που έχουν καταχωρηθεί στον κόμβο w. Επιπρόσθετα, όλα τα διαστήματα του δεξιού υποδένδρου του w περιλαμβάνονται στο αποτέλεσμα, εκτός αν ο w είναι ο κόμβος καταχώρησης.

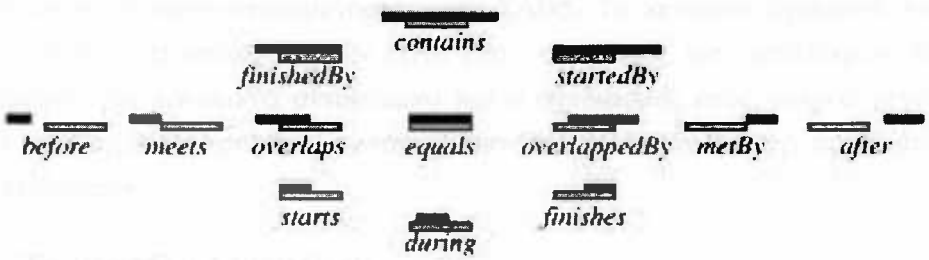
3. Διατρέχουμε το δένδρο από τον κόμβο καταχώρησης έως τον κόμβο με την κοντινότερη τιμή στο άκρο $upper$. Ανάλογα με τη φάση 2 διατρέχουμε τις λίστες $L(w)$.

Στο άρθρο [KPS00] προτείνεται η διάτρεξη του δένδρου να πραγματοποιείται εικονικά, ενώ οι λίστες $U(w)$ και $L(w)$ να μην διατρέχονται άμεσα κατά την επίσκεψη του αντίστοιχου κόμβου, αλλά να συλλέγονται σε μεταβατικούς πίνακες, `leftNodes` και `rightNodes`, και τελικά να πραγματοποιηθεί η ερώτηση:

```
SELECT id
FROM   Intervals i, leftNodes l, rightNodes r
```

```
WHERE   (i.node = l.node AND i.upper >= :lower)
        OR (i.node = r.node AND i.lower <= upper)
        OR (i.node BETWEEN :lower AND :upper)
```

Στο [KPS01] προτείνονται επεκτάσεις των όσων παρουσιάστηκαν παραπάνω για τις σχέσεις [All83] που απεικονίζονται στο σχήμα 6.



Σχήμα 6: Σχέσεις μεταξύ χρονικών διαστημάτων.

Σε αυτή την ενότητα παρουσιάσαμε τους μηχανισμούς με τους οποίους είναι δυνατό να επεκτείνουμε τη λειτουργικότητα των σχεσιακών-αντικειμενοστραφών ΣΔΒΔ για τη διαχείριση μη παραδοσιακών δεδομένων. Επίσης, παρουσιάσαμε τα εργαλεία που θα μας βοηθήσουν στην υλοποίηση της προτεινόμενης γλώσσας ερωτήσεων. Στη συνέχεια περιγράφουμε το μοντέλο δεδομένων και τη γλώσσα ερωτήσεων που προτείνονται, καθώς και τον τρόπο ενσωμάτωσής τους σε σχεσιακό-αντικειμενοστραφές ΣΔΒΔ. Ο σχεδιασμός του μοντέλου δεδομένων, της γλώσσας ερωτήσεων και της υλοποίησης-ενσωμάτωσης πραγματοποιείται λαμβάνοντας υπόψη όλα τα παραπάνω.

4 Διαχείριση κινούμενων αντικειμένων σε περιβάλλον σχεσιακού - αντικειμενοστραφούς ΣΔΒΔ

Σε αυτή την ενότητα περιγράφουμε το μοντέλο δεδομένων και τη γλώσσα ερωτήσεων που έχουν προταθεί στην εργασία [VW01], με κάποιες, όμως, διαφοροποιήσεις που πραγματοποιούνται ώστε να είναι δυνατό να το εντάξουμε εύκολα σε περιβάλλον σχεσιακού-αντικειμενοστραφούς ΣΔΒΔ. Τα κεντρικά ζητήματα, επομένως, που θα μας απασχολήσουν είναι δύο: ο ορισμός των κατάλληλων τύπων δεδομένων για κινούμενα αντικείμενα και ο σχεδιασμός ενός, μικρού μεγέθους αλλά μεγάλης εκφραστικής ικανότητας, συνόλου τελεστών για την πραγματοποίηση ερωτήσεων.

4.1 Το μοντέλο δεδομένων

Όπως ήδη έχουμε αναφέρει στην ενότητα 2.2, τα δεδομένα που παράγονται από την κίνηση αντικειμένων σχηματίζουν μια καμπύλη σε τρισδιάστατο χώρο [PT00], όπου δύο από τις τρεις διαστάσεις αντιστοιχούν στο δυσδιάστατο (x-, y-) επίπεδο και η τρίτη διάσταση αντιστοιχεί στο χρόνο (σχήμα 1, σελίδα 11). Επίσης, αντί της καμπύλης, σε μια βάση δεδομένων κινούμενων αντικειμένων αποθηκεύουμε και διαχειριζόμαστε μια τρισδιάστατη ακολουθία γραμμικών τμημάτων, τα οποία αναπαριστούν την *τροχιά* του αντικειμένου και προκύπτουν από τη δειγματοληψία της συνεχούς κίνησής του.

Έτσι, μοντελοποιούμε αυτή την κίνηση ενός αντικειμένου ως μια τρισδιάστατη ακολουθία γραμμικών τμημάτων (ή σύνολο από ακολουθίες αν υπάρχουν χρονικές στιγμές όπου δεν έχουμε πληροφορία για τη θέση του αντικειμένου). Τα σχεσιακά-αντικειμενοστραφή ΣΔΒΔ μας επιτρέπουν να ορίσουμε κατάλληλους τύπους δεδομένων για την τροχιά και τα τμήματα της τροχιάς ενός αντικειμένου με την βοήθεια των τύπων δεδομένων για αντικείμενα και σύνολα (collections) αντικειμένων. Με τη χρήση DDL προτάσεων προκύπτουν οι ορισμοί[4]:

```
CREATE TYPE TRAJECTORYSEGMENT AS OBJECT (
    t_lower NUMBER,
    t_upper NUMBER,
    x_lower NUMBER,
    y_lower NUMBER,
```

[4]Χρησιμοποιούμε το συντακτικό της Oracle, ανάλογο είναι το συντακτικό σε άλλα ΣΔΒΔ.

```
    x_upper NUMBER,
    y_upper NUMBER
)

CREATE TYPE TRAJECTORY AS TABLE OF TRAJECTORYSEGMENT
```

Χρησιμοποιώντας τους παραπάνω ορισμούς μπορούμε να αποθηκεύσουμε τις τροχιές κινούμενων αντικειμένων σε ένα συνήθη πίνακα:

```
CREATE TABLE MOVINGOBJECTS (moid NUMBER, traj TRAJECTORY,
colorid NUMBER, weight NUMBER, driverid NUMBER)
```

Επιπρόσθετα, επωφελούμενοι των πλαισίων επέκτασης, είναι δυνατό να ορίσουμε νέους τελεστές που θα ενεργούν πάνω στον τύπο TRAJECTORY για την πραγματοποίηση ερωτήσεων και να χρησιμοποιήσουμε τις πιο κατάλληλες δομές δεικτοδότησης που είναι διαθέσιμες σε σχεσιακά-αντικειμενοστραφή ΣΔΒΔ.

4.2 Η γλώσσα ερωτήσεων

Στην εργασία [VW01] προτείνονται δέκα είδη ερωτήσεων:

1. LOC(time)
2. WHENAT(location)
3. WHITHIN(traveltime from R, along existing path, always between st and et)
4. WHITHIN(traveldistance from R, along existing path, always between st and et)
5. WHITHIN(traveldistance from R, along shortest path, always between st and et)
6. WHITHIN(traveltime from R, along existing path, sometimes between st and et)
7. WHITHIN(traveldistance from R, along existing path, sometimes between st and et)
8. WHITHIN(traveldistance from R, along shortest path, sometimes between st and et)
9. WHITHIN(traveltime from R, along shortest path, always between st and et)
10. WHITHIN(traveltime from R, along shortest path, sometimes between st and et)

ή, σε μια SQL-like μορφή, τα παραπάνω θα μπορούσαν να γραφούν ως εξής:

```
SELECT LOC(t) | WHEN_AT(location-L) |
       <other attributes of moving objects table>
FROM   MOVINGOBJECTS
WHERE  WITHIN (DISTANCE s |TRAVELTIME t) FROM R
       [(ALONG EXISTING PATH) | (ALONG SHORTEST PATH)]
        [(ALWAYS BETWEEN) | (SOMETIMES BETWEEN)
        starttime AND endtime]
```

όπου:

`LOC(t)`: είναι οι θέσεις όλων των αντικειμένων τη στιγμή t. Ο συγκεκριμένος τελεστής επιστρέφει μια λίστα ζευγών σημείων και αναγνωριστικών αντικειμένων (ids).

`WHEN_AT (location-L)`: οι χρονικές στιγμές κατά τις οποίες κάποιο υποσύνολο κινούμενων αντικειμένων διέρχονται από τη θέση *Location-L*. Το αποτέλεσμα της ερώτησης είναι ένα σύνολο των ζευγών χρονικών στιγμών και αναγνωριστικών των αντικειμένων. Ένα αναγνωριστικό αντικειμένου μπορεί να εμφανίζεται περισσότερες από μία φορές στο αποτέλεσμα, αν το αντίστοιχο αντικείμενο πέρασε περισσότερες από μία φορές από τη θέση *Location-L*.

`WITHIN (DISTANCE s | TRAVELTIME t) FROM R`: είναι ένας τελεστής που απαιτεί ένα από τα δύο ορίσματα: `DISTANCE s` ή `TRAVELTIME t`, όπου `s` και `t` είναι πραγματικοί αριθμοί και `R` ένα σημείο αναφοράς στο δυσδιάστατο χώρο. Ο τελεστής επιστρέφει τη λογική τιμή True, αν ένα αντικείμενο απαιτείται να διανύσει απόσταση μικρότερη ή ίση του `s` για να διέλθει από το σημείο `R`, ή αν απαιτείται χρόνος μικρότερος ή ίσος του `t` για να διέλθει από το σημείο `R`. Επίσης, προσδιορίζοντας και τα υπόλοιπα μη υποχρεωτικά ορίσματα, μπορούμε να απαιτήσουμε την επαλήθευση ισχυρότερων περιορισμών:

- `ALONG EXISTING PATH and ALONG SHORTEST PATH`. Οι δύο τιμές αυτού του ορίσματος είναι αμοιβαία αποκλειόμενες. Η πρώτη απαιτεί το κριτήριο WITHIN να είναι αληθές κατά μήκος της τροχιάς του αντικειμένου (πράγμα που σημαίνει ότι το σημείο `R` πρέπει να βρίσκεται πάνω στην τροχιά του αντικειμένου). Ποιοτικά, αυτό σημαίνει ότι

το αντικείμενο μπορεί να διέλθει από το R χωρίς να υπερβεί το μέτρο κόστους που έχει προσδιοριστεί (χρόνος ή απόσταση) καθώς κινείται πάνω στην τροχιά του. Η δεύτερη δυνατή τιμή του ορίσματος απαιτεί το κριτήριο WITHIN να είναι αληθές κατά μήκος του συντομότερου μονοπατιού[5] με αρχή την τρέχουσα θέση του αντικειμένου και τέλος τη θέση R. Αν καμιά από τις δύο τιμές δεν προσδιοριστεί, ως προεπιλογή θεωρείται η τιμή `ALONG EXISTING PATH`.

- `ALWAYS BETWEEN and SOMETIMES BETWEEN starttime AND endtime`. Οι δύο τιμές αυτού του ορίσματος αναφέρονται στη χρονική διάρκεια του κριτηρίου WITHIN. Η πρώτη απαιτεί το κριτήριο WITHIN να είναι αληθές για όλες τις χρονικές στιγμές μεταξύ `starttime` και `endtime`, ενώ η δεύτερη το κριτήριο WITHIN να είναι αληθές για μία τουλάχιστον χρονική στιγμή μεταξύ `starttime` και `endtime`. Αν καμιά από τις δύο τιμές δεν προσδιοριστεί, ως προεπιλογή θεωρείται η τιμή `ALWAYS BETWEEN` και ως `starttime` και `endtime` θεωρούνται οι μικρότερη και η μεγαλύτερη τιμή, αντίστοιχα που υπάρχει στη βάση δεδομένων για κάθε αντικείμενο.

Στην παρούσα προσέγγιση υιοθετούμε επίσης τους τύπους ερωτήσεων 1--8, όχι όμως και τους 9--10. Για αυτούς τους τύπους ερωτήσεων στο [VW01] απαιτείται η ύπαρξη ειδικών χαρτών που μας επιτρέπουν να γνωρίζουμε την τυπική ταχύτητα ενός κινούμενου αντικειμένου δεδομένου ότι κινείται πάνω στη συντομότερη διαδρομή προς ένα καθορισμένο σημείο και δεδομένης της τρέχουσας θέσης του. Μιας και δεν πραγματοποιούμε υποθέσεις σχετικά με την ύπαρξη ειδικών χαρτών, συμπεριλαμβάνουμε στη γλώσσα ερωτήσεων μόνο τους τύπους 1-8.

Στην επόμενη ενότητα, περιγράφουμε το σχεδιασμό του προτεινόμενου συστήματος για διαχείριση κινούμενων αντικειμένων με χρήση των δυνατοτήτων της Oracle.

[5] Ως συντομότερο μονοπάτι θεωρούμε την ευθεία που συνδέει την τρέχουσα θέση του αντικειμένου με το σημείου αναφοράς, R

5 Σχεδιασμός του προτεινόμενου συστήματος

Θα ήταν ιδανικό να προσπαθήσουμε να ενσωματώσουμε την προτεινόμενη γλώσσα ερωτήσεων σε όλα τα μεγάλα ΣΔΒΔ που κυκλοφορούν στην αγορά και να αναφέρουμε τις εμπειρίες μας σχετικά με το πιο θεωρούμε περισσότερο κατάλληλο. Μέχρι στιγμής, επιλέξαμε να πειραματιστούμε με το λογισμικό Oracle 9i.

Χρησιμοποιώντας την ορολογία της Oracle, ο σχεδιασμός του συστήματος αντιστοιχεί στην ανάπτυξη ενός νέου Data Cartridge [Ora01a]. Έτσι, όπως είδαμε και στην ενότητα 3.1, απαιτούνται τα παρακάτω βήματα

- Ορισμός κατάλληλων τύπων δεδομένων με τη βοήθεια των Abstract Data Types.
- Ορισμός και υλοποίηση τελεστών μέσω των οποίων θα πραγματοποιούνται νέοι τύποι ερωτήσεων κατάλληλων για κινούμενα αντικείμενα.
- Επιλογή της καταλληλότερης μεθόδου δεικτοδότησης από αυτές που είναι διαθέσιμες.

Έχουμε ήδη περιγράψει τον τρόπο με τον οποίο πραγματοποιούμε το πρώτο βήμα στην προηγούμενη ενότητα. Το τρίτο βήμα πραγματοποιείται με την υλοποίηση αποθηκευμένων συναρτήσεων οι οποίες στη συνέχεια συνδέουμε (bind) με τους νέους τελεστές που ορίζουμε (π.χ. τον τελεστή WITHIN). Λεπτομερή σχήματα επεξεργασίας των νέων τύπων ερωτήσεων / τελεστών δίνουμε στην ενότητα 5.2.

Σχετικά με τη δεικτοδότηση, επιλέγουμε να δεικτοδοτήσουμε το σύνολο των τμημάτων της τροχιάς όλων των αντικειμένων, πράγμα που απαιτεί την "ανάλυση" (unnesting) των τύπων δεδομένων `TRAJECTORYSEGMENT` και `TRAJECTORY` στα δομικά τους συστατικά[6] και την αποθήκευση τους σε ένα πίνακα-ευρετήριο (όπως καθορίζει το πλαίσιο επεκτασιμότητας) του οποίου όλες οι στήλες αντιστοιχούν σε τύπους δεδομένων που υποστηρίζει το ΣΔΒΔ. Δηλαδή, η αρχική μας πρόθεση ήταν να έχουμε ένα ευρετήριο με εγγραφές του τύπου `(movingobjectid, segmentid, 3dlinesegment)` και να οργανώσουμε αυτόν τον πίνακα-ευρετήριο με ένα "λογικό" R-tree που παρέχεται από την Oracle Spatial. Όμως, αν και η Oracle Spatial υποστηρίζει αντικείμενα έως και 4 διαστάσεων, στη δεικτοδότηση χρησιμοποιούνται μόνο οι δύο πρώτες [Ora01b]. Κατά συνέπεια αποφασίσαμε να δεικτοδοτήσουμε τη χωρική διάσταση ξεχωριστά από την

[6] Λεπτομέρειες για αυτή τη διαδικασία καθώς και ένα παράδειγμα υπάρχουν στο [Ora01a], όπου αναπτύσσεται το Power Demand Cartridge.

χρονική με ένα Relational R-tree και ένα Relational Interval tree, αντίστοιχα, αποθηκεύοντας εγγραφές της μορφής `(moid, segid, t_lower, t_upper, artificial_node, 2dspatialsegment)`.

5.1 Τρέχουσα υλοποίηση

Αν και τα πλαίσια επέκτασης αποτελούν ένα πολύτιμο μηχανισμό για το σχεδιαστή μιας εφαρμογής διαχείρισης μη παραδοσιακών δεδομένων και ταυτόχρονα αποκρύπτουν τις λεπτομέρειες της υλοποίησης από το χρήστη, για τη γρήγορη προτυποποίηση του συστήματος, έχουμε υλοποιήσει μια πιο απλή εκδοχή, στο πνεύμα, όμως, όλων αυτών που έχουμε περιγράψει μέχρι στιγμής. Αποθηκεύουμε όλα τα δεδομένα για την κίνηση των αντικειμένων σε ένα συνήθη πίνακα:

```
CREATE TABLE TRAJECTORIES(
moid     NUMBER,              -- moving object id
segid    NUMBER,              -- trajectory segment id
node     NUMBER,           -- artificial node for RI-tree
lower    NUMBER,            -- lower time interval bound
upper    NUMBER,            -- upper time interval bound
spatseg MDSYS.SDO_GEOMETRY -- 2d spatial segment)
```

Για τη χρονική δεικτοδότηση, έχουμε ορίσει δύο σύνθετα B-tree στα ζεύγη στηλών `(node, lower)` και `(node, upper)` και έχουμε επίσης υλοποιήσει συναρτήσεις για τον υπολογισμό της τιμής του τεχνητού κόμβου `node` για την πραγματοποίηση εισαγωγών, καθώς και της συνάρτησης που προετοιμάζει τις λίστες "left" και "right" για την πραγματοποίηση ερωτήσεων επικάλυψης στο χρόνο (βλέπε και ενότητα 3.3, σελίδα 26). Ακολουθήσαμε ακριβώς τη μεθοδολογία που προτάθηκε στο [KPS00] και έχουμε επαληθεύσει ότι παράγουμε το ίδιο πλάνο εκτέλεσης. Για τη χωρική δεικτοδότηση δημιουργούμε ένα Relational R-tree πάνω στη στήλη `spatseg`.

Επίσης, υλοποιήσαμε τους τύπους ερωτήσεων 1--8 ως αποθηκευμένες συναρτήσεις δύο βημάτων. Κατά το πρώτο βήμα επιλέγεται ένα σύνολο εγγραφών με χρήση είτε του χρονικού είτε του χωρικού ευρετηρίου και κατά το δεύτερο εξετάζεται με ακρίβεια η ισχύς των περιορισμών που θέτουν οι τελεστές ερωτήσεων. Ακολουθούν λεπτομερή σχήματα επεξεργασίας ερωτήσεων.

5.2 Σχήματα επεξεργασίας των προτεινόμενων ερωτήσεων

Για τα παρακάτω σχήματα επεξεργασίας ερωτήσεων ισχύουν οι υποθέσεις:

1. η συνάρτηση intersects που χρησιμοποιείται για να ελέγχουμε αν ένα δυσδιάστατο σημείο βρίσκεται πάνω σε ένα δυσδιάστατο γραμμικό τμήμα παρέχεται από το ΣΔΒΔ.

2. σε ένα τμήμα της τροχιάς ενός αντικειμένου, η χρονική στιγμή κατά την οποία διέρχεται από ένα σημείο του χωρικού τμήματος καθώς και η χωρική θέση του αντικειμένου σε μια χρονική στιγμή του χρονικού διαστήματος, υπολογίζονται με γραμμική παρεμβολή.

3. Η οριζόμενη από το χρήστη συνάρτηση dist(t_1, t_2, trajectorysegment) υπολογίζει την απόσταση που διανύεται από ένα χωρικό αντικείμενο, πάνω στο τμήμα trajectorysegment, μεταξύ των χρονικών στιγμών t_1 και t_2, με τη χρήση γραμμικής παρεμβολής.

4. η συνάρτηση within που χρησιμοποιείται για να ελέγχουμε αν ένα δυσδιάστατο χωρικό τμήμα απέχει όχι περισσότερο από d από ένα δυσδιάστατο σημείο R παρέχεται από το ΣΔΒΔ.

5. Η οριζόμενη από το χρήστη συνάρτηση segmentpart(trajectorysegment, t_1, t_2) επιστρέφει ένα χωρικό τμήμα που αποτελεί το μέρος του χωρικού τμήματος του τρισδιάστατου trajectorysegment και του οποίου η αρχή και το τέλος αντιστοιχούν στις χρονικές στιγμές t_1 και t_2. Για να υπολογιστεί το αποτέλεσμα χρησιμοποιείται γραμμική παρεμβολή.

Οι υποθέσεις 1 και 4 ισχύουν για την περίπτωση της Oracle Spatial, όπως ήδη έχουμε περιγράψει ενότητα 3.2.2 (σελίδα 22).

LOC(t)

Η συνάρτηση αυτή δέχεται ως παράμετρο μια χρονική στιγμή t και επιστρέφει τη θέση όλων των αντικειμένων (αν είναι γνωστή) αυτή τη χρονική στιγμή. Η ερώτηση εκτελείται με τη βοήθεια του χρονικού ευρετηρίου και του παραθύρου ερώτησης $[t, t]$. Για κάθε αντικείμενο, του οποίου κάποιο τμήμα της τροχιάς του αντιστοιχεί σε χρονικό διάστημα που επικαλύπτεται με το παράθυρο της ερώτησης, υπολογίζουμε τη θέση του πάνω στο χωρικό τμήμα του τρισδιάστατου τμήματος της τροχιάς με χρήση γραμμικής παρεμβολής.

WHENAT(pointlocation)

Χρησιμοποιώντας το χωρικό ευρετήριο, ανακτώνται όλα τα τμήματα τροχιάς όλων των αντικειμένων των οποίων το χωρικό τμήμα περιέχει το σημείο pointlocation. Έπειτα, με τη χρήση γραμμικής παρεμβολής, υπολογίζουμε, για κάθε αντικείμενο που ανήκει στο αποτέλεσμα, τις χρονική στιγμές κατά τις οποίες διέρχεται από το σημείο pointlocation.

WITHIN(traveltime t from R, along existing path, always between start time st and end time et)

Χρησιμοποιούμε την προσέγγιση δύο βημάτων, του φιλτραρίσματος και λεπτομερούς επεξεργασίας (filter and refinment) για να επεξεργαστούμε αυτή την ερώτηση (αλγόριθμος 1). Το πρώτο βήμα χρησιμοποιεί τον χρονικό δείκτη ως ακολούθως. Το παράθυρο της ερώτησης είναι το χρονικό διάστημα $[st, et + t]$ και ανακτώνται όλα τα τμήματα της τροχιάς όλων των αντικειμένων των οποίων το αντίστοιχο χρονικό διάστημα επικαλύπτεται με το παράθυρο της ερώτησης.

Το δεύτερο βήμα έχει ως ακολούθως. Αρχικά, ελέγχουμε αν όλα τα τμήματα τροχιάς που ανήκουν στο ίδιο αντικείμενο είναι συνεχόμενα (δηλαδή $s_i.t\_upper = s_{i+1}.t\_lower, i = 1 \ldots N$, N είναι ο αριθμός των τμημάτων τροχιάς, s_i, που ανήκουν σε ένα συγκεκριμένο αντικείμενο, mo_j), πράγμα που απαιτεί την ταξινόμηση των τμημάτων τροχιάς σύμφωνα με το ένα από τα δύο άκρα του χρονικού διαστήματος. Όλα τα αντικείμενα που δεν ικανοποιούν αυτή τη συνθήκη εξαιρούνται από το αποτέλεσμα της ερώτησης. Στη συνέχεια, πρέπει να εντοπίσουμε τα κινούμενα αντικείμενα που διέρχονται από το σημείο αναφοράς, R, και για τα οποία το κριτήριο $within$ αληθεύει. Έστω e_1 η χρονική στιγμή κατά την οποία ένα αντικείμενο διέρχεται για πρώτη φορά από το σημείο R μεταξύ st και $et+t$, e_2 η χρονική στιγμή κατά την οποία διέρχεται για δεύτερη φορά από το σημείο R κ.λπ. Σημειώνουμε ότι $e_1 \geq st$ εξορισμού. Αν $e_1 - st > t$, τότε προφανώς η θέση του αντικειμένου κατά το χρόνο st απέχει περισσότερο από t μονάδες χρόνου από το R κατά μήκος της τροχιάς, και επομένως το αντικείμενο εξαιρείται από το αποτέλεσμα της ερώτησης. Διαφορετικά, αν $t \geq e_1 - st$, τότε εξετάζουμε τη στιγμή et. Αν $e_1 \geq et$, τότε το αντικείμενο δεν έχει φτάσει στο R ακόμη, αλλά προφανώς, από τη θέση του τη στιγμή et, θα φτάσει στο σημείο R σε λιγότερο από t μονάδες χρόνου, αφού τη στιγμή st δεν απέχει περισσότερο από t μονάδες χρόνου. Συμπεραίνουμε ότι

Inputs: t {within time}, st {predicate validity start}, et {predicate validity end}, R {reference point}.
Returns: M {The set of moving objects ids that satisfy the predicate}.
Select, using the time interval index, all trajectory segments of all objects intersecting the time interval $[st, et + t]$. Order them first by moving object id and second by asceding lower time limit.
$M = \varnothing$
for all moving objects, mo_j **do**
  t_{end} = (first segment of mo_j).t_{lower}
  $t_{st} = st$
  for each segment, s_i of mo_j **do**
    if $s_i.t_{lower} \neq t_{end}$ **then**
      continue with the next moving object
    else
      if $s_i.spatseg$ intersects R **then**
        t_R = intersection time
        if $t_R - t_{st} < t$ **then**
          if $t_R > et$ **then**
            $M = M \cup \{mo_j.id\}$
            continue with the next moving object
          else
            $t_{st} = t_R$
          end if
        else
          continue with the next moving object
        end if
      end if
      $t_{end} = s_i.t_{upper}$
    end if
  end for
end for

Algorithm 1: Predicate WITHIN-T-EP-AB, cost in terms of travel time, along existing path, always between start time and end time.

η τροχιά του συγκεκριμένου αντικειμένου πληροί τους περιορισμούς της ερώτησης και συνεχίζουμε για να εξετάσουμε το επόμενο αντικείμενο.

Διαφορετικά, αν $et > e_1$, πρέπει να εξετάσουμε αν το μέρος της τροχιάς μεταξύ e_1 και et ικανοποιεί τους περιορισμούς της ερώτησης ως προς τις στιγμές e_2, e_3, e_4, κ.λπ. Για το λόγο αυτό, θέτουμε $e_1 = st$ και επαναλαμβάνουμε τη διαδικασία αναδρομικά. Ειδικότερα, αν η στιγμή e_2 δεν υπάρχει ή αν $e_2 - st > t$, τότε προφανώς η θέση του αντικειμένου κατά το χρόνο st απέχει περισσότερο από t μονάδες χρόνου από το R κατά μήκος της τροχιάς, και επομένως το αντικείμενο εξαιρείται από το αποτέλεσμα της ερώτησης. Διαφορετικά, αν $t \geq e_2 - st$, τότε εξετάζουμε τη στιγμή et. Αν $e_2 \geq et$, τότε το αντικείμενο δεν έχει φτάσει στο R ακόμη, αλλά προφανώς, από τη θέση του τη στιγμή et, θα φτάσει στο σημείο R σε λιγότερο από t μονάδες χρόνου, αφού τη στιγμή st δεν απέχει περισσότερο από t μονάδες χρόνου. Συμπεραίνουμε ότι η τροχιά του συγκεκριμένου αντικειμένου πληροί τους περιορισμούς της ερώτησης και συνεχίζουμε για να εξετάσουμε το επόμενο αντικείμενο. Διαφορετικά, αν $et > e_2$, πρέπει να εξετάσουμε αν το μέρος της τροχιάς μεταξύ e_2 και et ικανοποιεί τους περιορισμούς της ερώτησης ως προς τις στιγμές e_3, e_4, κ.λπ. Τα αντικείμενα που απομένουν στο σύνολο του αποτελέσματος, μετά από όλα αυτά τα βήματα συνιστούν την απάντηση στην ερώτηση.

WITHIN(traveltime t **from** R**, along existing path, sometimes between start time** st **and end time** et**)**

Τα αντικείμενα που ικανοποιούν τους περιορισμούς αυτού τους ερωτήματος είναι όσα διέρχονται από το σημείο R κάποια στιγμή μεταξύ s και $et + t$ (αλγόριθμος 2). Πρόκειται για μια απλούστερη εκδοχή της περίπτωσης "always between". Το πρώτο βήμα της επεξεργασίας είναι ακριβώς το ίδιο, ενώ το δεύτερο απλοποιείται σημαντικά. Ειδικότερα, το παράθυρο της ερώτησης και το ευρετήριο που χρησιμοποιούμε είναι τα ίδια. Το κριτήριο within, όμως, ικανοποιείται για κάποιο αντικείμενο, αν διέρχεται τουλάχιστον μια φορά από το R κάποια στιγμή μεταξύ st και $et + t$.

WHITHIN(traveldistance d **from** R**, along existing path, always between** st **and** et**)**

Για αυτό τον τύπο ερώτησης, χρησιμοποιούμε, όπως και για τους προηγούμενους, την προσέγγιση δύο βημάτων, του φιλτραρίσματος και λεπτομερούς επεξερ-

Inputs: t {within time}, st {predicate validity start}, et {predicate validity end}, R {reference point}.
Returns: M {The set of moving objects ids that satisfy the predicate}.
Select, using the time interval index, all trajectory segments of all objects intersecting the time interval $[st, et + t]$. Order them first by moving object id and second by asceding lower time limit.

```
M = ∅
for all moving objects, mo_j do
  t_end = (first segment of mo_j).t_lower
  t_st = st
  for each segment, s_i of mo_j do
    if s_i.t_lower ≠ t_end then
      continue with the next moving object
    else
      if s_i.spatseg intersects R then
        t_R = intersection time
        if t_st ≤ t_R ≤ t_end + t then
          M = M ∪ {mo_j.id}
          continue with the next moving object
        end if
      end if
      t_end = s_i.t_upper
    end if
  end for
end for
```

Algorithm 2: Predicate WITHIN-T-EP-SB, cost in terms of travel time, along existing path, sometimes between start time and end time.

γασίας (filter and refinment) για να επεξεργαστούμε αυτή την ερώτηση (αλγόριθμος 3). Το πρώτο βήμα χρησιμοποιεί τον χωρικό δείκτη ως ακολούθως. Το παράθυρο της ερώτησης είναι ένα τετράγωνο με κέντρο βάρους το σημείο R και μεγέθους πλευράς $2d$. Με αυτό το παράθυρο ανακτώνται όλα τα τμήματα της τροχιάς όλων των αντικειμένων των οποίων το αντίστοιχο χωρικό τμήμα επικαλύπτεται με το παράθυρο της ερώτησης.

Το δεύτερο βήμα έχει ως ακολούθως. Αρχικά, ελέγχουμε αν όλα τα τμήματα τροχιάς που ανήκουν στο ίδιο αντικείμενο είναι συνεχόμενα (δηλαδή $s_i.t_{upper} = s_{i+1}.t_{lower}$, $i = 1 \ldots N$, N είναι ο αριθμός των τμημάτων τροχιάς, s_i, που ανήκουν σε ένα συγκεκριμένο αντικείμενο, mo_j), πράγμα που απαιτεί την ταξινόμηση των τμημάτων τροχιάς σύμφωνα με το ένα από τα δύο άκρα του χρονικού διαστήματος. Όλα τα αντικείμενα που δεν ικανοποιούν αυτή τη συνθήκη εξαιρούνται από το αποτέλεσμα της ερώτησης. Στη συνέχεια, πρέπει να εντοπίσουμε τα κινούμενα αντικείμενα που διέρχονται από το σημείο αναφοράς, R, και για τα οποία το κριτήριο $within$ αληθεύει. Έστω e_1 η θέση όπου ένα αντικείμενο διέρχεται για πρώτη φορά από το σημείο R μεταξύ $loc1$ = LOC(mo_j, st) και $loc2$ = (θέση αφού διανυθούν d μονάδες απόστασης από τη LOC(mo_j, et) κατά μήκος της τροχιάς), e_2 η θέση όπου διέρχεται για δεύτερη φορά από το σημείο R μεταξύ $loc1$ και $loc2$ κ.λπ. Αν η απόσταση που διανύεται πάνω στην τροχιά μεταξύ των θέσεων LOC(mo_j, st) και e_1 είναι μεγαλύτερη του d, τότε προφανώς η θέση του αντικειμένου κατά το χρόνο st απέχει περισσότερο από d μονάδες απόστασης από το R κατά μήκος της τροχιάς, και επομένως το αντικείμενο εξαιρείται από το αποτέλεσμα της ερώτησης. Διαφορετικά, αν αυτή η απόσταση είναι μικρότερη του d, τότε εξετάζουμε τη χρονική στιγμή WHENAT(mo_j, $e1$). Αν WHENAT(mo_j, $e1$)$\geq$ et, τότε το αντικείμενο δεν έχει φτάσει στο R ακόμη, αλλά προφανώς, από τη θέση του τη στιγμή et, θα φτάσει στο σημείο R σε λιγότερο από d μονάδες απόστασης, αφού τη στιγμή st δεν απέχει περισσότερο από d μονάδες απόστασης. Συμπεραίνουμε ότι η τροχιά του συγκεκριμένου αντικειμένου πληροί τους περιορισμούς της ερώτησης και συνεχίζουμε για να εξετάσουμε το επόμενο αντικείμενο.

Διαφορετικά, αν et >WHENAT(mo_j, $e1$), πρέπει να εξετάσουμε αν το μέρος της τροχιάς μεταξύ e_1 και LOC(mo_j, et) ικανοποιεί τους περιορισμούς της ερώτησης ως προς τις θέσεις e_2, e_3, e_4, κ.λπ. Για το λόγο αυτό, θέτουμε st = WHENAT(mo_j, $e1$) και επαναλαμβάνουμε τη διαδικασία αναδρομικά. Ειδικότερα, αν η θέση e_2 δεν υπάρχει ή αν απέχει περισσότερο από d μονάδες απόστασης από

τη θέση LOC(mo_j, st), τότε προφανώς η θέση του αντικειμένου κατά το χρόνο st απέχει περισσότερο από d μονάδες απόστασης από το R κατά μήκος της τροχιάς, και επομένως το αντικείμενο εξαιρείται από το αποτέλεσμα της ερώτησης. Διαφορετικά, αν η θέση e_2 δεν απέχει περισσότερο από d μονάδες απόστασης από τη θέση LOC(mo_j, st), τότε εξετάζουμε τη στιγμή et. Αν $e_2 \geq$WHENAT(mo_j, $e1$), τότε το αντικείμενο δεν έχει φτάσει στο R ακόμη, αλλά προφανώς, από τη θέση του τη στιγμή et, θα φτάσει στο σημείο R σε λιγότερο από d μονάδες απόστασης, αφού τη στιγμή st δεν απέχει περισσότερο από d μονάδες απόστασης. Συμπεραίνουμε ότι η τροχιά του συγκεκριμένου αντικειμένου πληροί τους περιορισμούς της ερώτησης και συνεχίζουμε για να εξετάσουμε το επόμενο αντικείμενο. Διαφορετικά, αν $et >$WHENAT(mo_j, $e1$), πρέπει να εξετάσουμε αν το μέρος της τροχιάς μεταξύ e_2 και LOC(mo_j, et) ικανοποιεί τους περιορισμούς της ερώτησης ως προς τις θέσεις e_3, e_4, κ.λπ. Τα αντικείμενα που απομένουν στο σύνολο του αποτελέσματος, μετά από όλα αυτά τα βήματα συνιστούν την απάντηση στην ερώτηση.

WHITHIN(traveldistance d from R, along existing path, sometimes between st and et)

Πρόκειται για μια απλούστερη εκδοχή της περίπτωσης "always between". Το πρώτο βήμα της επεξεργασίας είναι ακριβώς το ίδιο, ενώ το δεύτερο απλοποιείται σημαντικά. Ειδικότερα, το παράθυρο της ερώτησης και το ευρετήριο που χρησιμοποιούμε είναι τα ίδια. Το κριτήριο within, όμως, ικανοποιείται για κάποιο αντικείμενο, αν διέρχεται τουλάχιστον μια φορά από το R, το σημείο R βρίσκεται πάνω στην τροχιά του αντικειμένου και η απόστασή του, κατά μήκος αυτής της τροχιάς, δεν είναι μεγαλύτερη του d από τη θέση $loc_1 = LOC(mo_j, st)$ ή από τη θέση loc_2 = (θέση αφού διανυθούν d μονάδες απόστασης από τη LOC(mo_j, et) κατά μήκος της τροχιάς) (αλγόριθμος 4).

WHITHIN(traveldistance from R, along shortest path, always between st and et)

Για την εκτέλεση αυτού του τύπου ερωτήματος χρησιμοποιούμε το χωρικό ευρετήριο (Αλγόριθμος 5). Το παράθυρο της ερώτησης του βήματος φιλτραρίσματος είναι ένα δύο διαστάσεων τετράγωνο με κέντρο βάρους το σημείο αναφοράς R και με μέγεθος πλευράς $2d$. Με την εκτέλεση του πρώτου βήματος ανακτώνται

Inputs: d {within distance}, st {predicate validity start}, et {predicate validity end}, R {reference point}.
Returns: M {The set of moving objects ids that satisfy the predicate}.
Select, using the spatial index, all trajectory segments of all objects intersecting the two dimensional square whose center of gravity is R and whose side size is $2d$. Order them first by moving object id and second by asceding lower time limit.
$M = \varnothing$
for each moving object, mo_j **do**
  t_{end} = (first segment of mo_j).t_{lower}
  for each segment, s_i of mo_j **do**
    if $s_i.t_{upper} < st$ **then**
      continue with next segment
    else if $st \geq si.t_{lower}$ **then**
      $t_{st} = st$
      $distfromst = 0$
    else
      $t_{st} = s_i.t_{lower}$
    end if
    if $s_i.t_{lower} \neq t_{end}$ **then**
      continue with the next moving object
    else
      if $s_i.spatseg$ intersects R **then**
        t_R = intersection time
        if $t_R > t_{st} \wedge distfromst +$ dist$(t_{st}, t_R, s_i) < d$ **then**
          if $t_R > et$ **then**
            $M = M \cup \{mo_j.id\}$
            continue with the next moving object
          else
            $distfromst =$ dist$(t_R, s_i.t_{upper}, s_i)$
          end if
        else
          continue with the next moving object
        end if
      else
        $distfromst = distfromst +$ dist$(t_{st}, s_i.t_{upper}, s_i)$
      end if
      $t_{end} = s_i.t_{upper}$
    end if
  end for
end for

Algorithm 3: Predicate WITHIN-D-EP-AB, cost in terms of distance, along existing path, always between startime and end time.

Inputs: d {within distance}, st {predicate validity start}, et {predicate validity end}, R {reference point}.
Returns: M {The set of moving objects ids that satisfy the predicate}.
Select, using the spatial index, all trajectory segments of all objects intersecting the two dimensional square whose center of gravity is R and whose side size is $2d$. Order them first by moving object id and second by asceding lower time limit.
$M = \varnothing$
for each moving object, mo_j **do**
  t_{end} = (first segment of mo_j).t_{lower}
  for each segment, s_i of mo_j **do**
    if $s_i.t_{upper} < st$ **then**
      continue with next segment
    else if $st \geq s_i.t_{lower}$ **then**
      $t_{st} = st$; $distfromst = 0$
    else
      $t_{st} = s_i.t_{lower}$
    end if
    if $s_i.t_{lower} \leq et < s_i.t_{upper}$ **then**
      $t_{et} = et$; $distfromet = 0$
    else if $et < s_i.t_{lower}$ **then**
      $t_{et} = s_i.t_{lower}$
    end if
    if $s_i.t_{lower} \neq t_{end}$ **then**
      continue with the next moving object
    else
      if $s_i.spatseg$ intersects R **then**
        t_R = intersection time
        if $(t_R > st \wedge distfromst+ \text{dist}(t_{st}, t_R, s_i) < d) \vee (t_R > et \wedge distfromet+ \text{dist}(t_{et}, t_R, s_i) < d)$ **then**
          $M = M \cup \{mo_j.id\}$
          continue with the next moving object
        else
          $distfromst = \text{dist}(t_R, s_i.t_{upper}, s_i)$
        end if
      else
        $distfromst = distfromst+ \text{dist}(t_{st}, s_i.t_{upper}, s_i)$
      end if
      if $et < s_i.t_{upper}$ **then**
        $distfromet = distfromet+ \text{dist}(t_{et}, s_i.t_{upper}, s_i)$
      end if
      $t_{end} = s_i.t_{upper}$
    end if
  end for
end for

Algorithm 4: Predicate WITHIN-D-EP-SB, cost in terms of distance, along existing path, sometimes between startime and end time.

όλα τα τμήματα τροχιάς όλων των αντικειμένων των οποίων το χωρικό τμήμα επικαλύπτεται με το παράθυρο της ερώτησης. Το δεύτερο βήμα έχει ως ακολούθως. Αρχικά, ελέγχουμε αν όλα τα τμήματα τροχιάς που ανήκουν στο ίδιο αντικείμενο είναι συνεχόμενα (δηλαδή $s_i.t_{upper} = s_{i+1}.t_{lower}$, $i = 1 \ldots N$, N είναι ο αριθμός των τμημάτων τροχιάς, s_i, που ανήκουν σε ένα συγκεκριμένο αντικείμενο, mo_j), πράγμα που απαιτεί την ταξινόμηση των τμημάτων τροχιάς σύμφωνα με το ένα από τα δύο άκρα του χρονικού διαστήματος. Όλα τα αντικείμενα που δεν ικανοποιούν αυτή τη συνθήκη εξαιρούνται από το αποτέλεσμα της ερώτησης. Στη συνέχεια, ελέγχουμε αν όλα τα τμήματα της τροχιάς (ή υπο-τμήματα μεταξύ των χρονικών στιγμών st και et) ενός αντικειμένου σε απόσταση το πολύ d από το σημείο αναφοράς R. Ο τελευταίος αυτός έλεγχος πραγματοποιείται με τη βοήθεια κατάλληλης συνάρτησης που παρέχεται από την επέκταση Oracle Spatial.

WHITHIN(traveldistance from R, along shortest path, sometimes between st and et)

Σε αυτή την περίπτωση σχεδόν όλα τα βήματα επεξεργασίας είναι τα ίδια όπως στην "always between", εκτός του τελευταίου, όπου αρκεί τουλάχιστον ένα τμήμα τροχιάς (ή υπο-τμήμα μεταξύ των χρονικών στιγμών st και et) να βρίσκεται σε απόσταση το πολύ d από το σημείο αναφοράς R, ώστε να ικανοποιούνται οι συνθήκες του ερωτήματος(Αλγόριθμος 6).

Στην επόμενη ενότητα αξιολογούμε πειραματικά την προτεινόμενη γλώσσα ερωτήσεων με μια σειρά ενδεικτικών πειραμάτων που αφορούν το χρόνο απόκρισης των ερωτήσεων.

Inputs: d {within distance}, st {predicate validity start}, et {predicate validity end}, R {reference point}.
Returns: M {The set of moving objects ids that satisfy the predicate}.
Select, using the spatial index, all trajectory segments of all objects intersecting the two dimensional square whose center of gravity is R and whose side size is $2d$. Order them first by moving object id and second by asceding lower time limit.
M = {every object id belonging to the result set of the select statement}
for each moving object, mo_j **do**
  t_{end} = (first segment of mo_j).t_{lower}
  for each segment, s_i of mo_j **do**
    if $s_i.t_{upper} < st \vee s_i.t_{lower} > et$ **then**
      $t_{end} = s_i.t_{upper}$
      continue with next segment
    end if
    if $s_i.t_{lower} \neq t_{end}$ **then**
      $M = M \backslash \{mo_j.id\}$
      continue with the next moving object
    else
      if $st > s_i.t_{lower}$ **then**
        $s_i.spatseg$ = segmentpart($s_i, st, s_i.t_{upper}$)
      end if
      if $et < s_i.t_{upper}$ **then**
        $s_i.spatseg$ = segmentpart($s_i, si.t_{lower}, et$)
      end if
      if NOT within($s_i.spatseg, d, R$) **then**
        $M = M \backslash \{mo_j.id\}$
        continue with the next moving object
      end if
      $t_{end} = s_i.t_{upper}$
    end if
  end for
end for

Algorithm 5: Predicate WITHIN-D-SP-AB, cost in terms of distance, along shortest path, always between startime and end time.

Inputs: d {within distance}, st {predicate validity start}, et {predicate validity end}, R {reference point}.
Returns: M {The set of moving objects ids that satisfy the predicate}.
Select, using the spatial index, all trajectory segments of all objects intersecting the two dimensional square whose center of gravity is R and whose side size is $2d$. Order them first by moving object id and second by asceding lower time limit.
$M = \varnothing$
for each moving object, mo_j **do**
  t_{end} = (first segment of mo_j).t_{lower}
  for each segment, s_i of mo_j **do**
    if $s_i.t_{upper} < st \vee s_i.t_{lower} > et$ **then**
      $t_{end} = s_i.t_{upper}$
      continue with next segment
    end if
    if $s_i.t_{lower} \neq t_{end}$ **then**
      continue with the next moving object
    else
      if $st > s_i.t_{lower}$ **then**
        $s_i.spatseg$ = segmentpart($s_i, st, s_i.t_{upper}$)
      end if
      if $et < s_i.t_{upper}$ **then**
        $s_i.spatseg$ = segmentpart($s_i, si.t_{lower}, et$)
      end if
      if within($s_i.spatseg, d, R$) **then**
        $M = M \cup \{mo_j.id\}$
        continue with the next moving object
      end if
      $t_{end} = s_i.t_{upper}$
    end if
  end for
end for

Algorithm 6: Predicate WITHIN-D-SP-SB, cost in terms of distance, along shortest path, sometimes between startime and end time.

6 Αξιολόγηση

Σε αυτή την ενότητα εξετάζουμε την πολυπλοκότητα των προτεινόμενων τύπων ερωτήσεων και ελέγχουμε την υλοποίηση που πραγματοποιήσαμε ως προς την ικανότητα διαχείρισης πολλών αντικειμένων (scalability) με ένα σύνολο ενδεικτικών πειραμάτων.

6.1 Πολυπλοκότητα των προτεινόμενων ερωτήσεων

Όλοι οι τύποι ερωτήσεων που προτείνονται αποτελούνται από μια ερώτηση επιλογής, μια φάση ταξινόμησης και μια φάση λεπτομερούς επεξεργασίας. Υποθέτοντας ότι το αποτέλεσμα της ερώτησης επιλογής περιέχει r τμήματα τροχιάς και ότι είναι αρκετά μικρό ώστε να είναι δυνατό να αποθηκευθεί στην κύρια μνήμη, το κόστος της φάσης λεπτομερούς επεξεργασίας είναι $O(r)$ και της φάσης ταξινόμησης $O(r \log r)$. Όμως, ο κυρίαρχος όρος του κόστους είναι το άθροισμα των λειτουργιών εγγραφής / ανάγνωσης εξαιτίας της προσπέλασης του ευρετηρίου και της ανάκτησης των δεδομένων - αποτελεσμάτων από το δίσκο, οι οποίες πραγματοποιούνται κατά τη διάρκεια εκτέλεσης του ερωτήματος επιλογής, είτε με τη βοήθεια του χρονικού είτε με τη βοήθεια του χωρικού ευρετηρίου.

Για τη δομή RI-tree έχει αποδειχθεί [KPS00] ότι αυτό το άθροισμα είναι $O(h \log_b n + r/b)$, όπου n είναι ο συνολικός αριθμός των τμημάτων τροχιάς που δεικτοδοτούνται, b το μέγεθος block του δίσκου, h το ύψος του RI-tree (το οποίο εξαρτάται μόνο από το διάστημα τιμών των άκρων των χωρικών διαστημάτων, π.χ. το ύψος είναι h όταν οι τιμές του χρόνου ανήκουν στο διάστημα $[1, 2^{h-1}]$) και r είναι ο πληθάριθμος του αποτελέσματος της ερώτησης.

Για το "λογικό" R-tree είναι δύσκολο να εκφραστεί το παραπάνω άθροισμα με ένα αριθμητικό τύπο. Για την πραγματοποίηση R-tree δεικτοδότησης η Oracle Spatial χρησιμοποιεί ένα πίνακα μεταδεδομένων για να αποθηκεύσει δομικές πληροφορίες σχετικά με τη ρίζα της δομής, τη διάσταση των δεδομένων και το βαθμό εξόδου των κόμβων (fanout), καθώς και ένα πίνακα - δείκτη που περιέχει τους κόμβους του R-tree [KRSB99]. Δηλαδή υλοποιεί το "σχήμα πλοήγησης" (ενότητα 3.1, σελίδα 18) και ο πίνακας - δείκτης οργανώνεται με ένα B-tree.

6.2 Πειραματική αξιολόγηση

Σε αυτή την υποενότητα ελέγχουμε το εφικτό της υλοποίησης με ένα σύνολο πειραμάτων που αφορούν το χρόνο απόκρισης των δύο περισσότερο πολύπλοκων τύ-

πων ερωτήσεων:

- WHITHIN(traveltime from R, along existing path, always between st and et)
- WHITHIN(traveldistance from R, along existing path, always between st and et)

Για την εκτέλεση της πρώτης ερώτησης χρησιμοποιείται ο χρονικός δείκτης και για την εκτέλεση της δεύτερης ο χωρικός.

Εξαιτίας της έλλειψης δημόσια διαθέσιμων πραγματικών χωροχρονικών δεδομένων για πειραματικούς σκοπούς, χρησιμοποιήσαμε συνθετικά δεδομένα για την πραγματοποίηση της αξιολόγησης. Στη βιβλιογραφία έχουν προταθεί τρεις γεννήτριες συνθετικών χωροχρονικών δεδομένων:

- Ο αλγόριθμος GSTD[7] ("Generating Spatio-Temporal Datasets") προτάθηκε στην εργασία [TN00]. Ο αλγόριθμος αυτός παράγει σημεία και ορθογώνια. Το πρώτο βήμα του αλγορίθμου αποτελεί η τοποθέτηση των κέντρων των αντικειμένων σύμφωνα με προκαθορισμένες κατανομές πιθανότητας. Μετά από αυτή τη φάση αρχικοποίησης, η κίνηση των αντικειμένων ελέγχεται με τη βοήθεια τριών κύριων παραμέτρων: (α) τη διάρκεια ζωής των αντικειμένων, (β) το μέγεθος του βήματος μετακίνησης προς κάποια κατεύθυνση και (γ) τη μεταβολή του μεγέθους των αντικειμένων (μόνο για τα ορθογώνια).

- Η γεννήτρια OPORTO[8] [SM00] παράγει κινούμενα σημεία και κινούμενες περιοχές. Κίνητρο για την υλοποίησή της αποτέλεσε η ιδέα των σκαφών ψαρέματος. Τα σκάφη προσελκύονται από ομάδες ψαριών, ενώ απωθούνται από καταιγίδες και έντονα καιρικά φαινόμενα. Αντίστοιχα, τα ψάρια προσελκύονται από περιοχές όπου υπάρχει τροφή. Τα σκάφη είναι κινούμενα σημεία ενώ τα κοπάδια ψαριών, οι περιοχές όπου υπάρχουν καταιγίδες και οι περιοχές όπου υπάρχει τροφή είναι κινούμενες περιοχές.

- Η γεννήτρια[9] που προτάθηκε στην εργασία [Bri00] παράγει αντικείμενα που κινούνται πάνω σε οδικά δίκτυα. Το κίνητρο για την υλοποίηση αυτής της γεννήτριας αποτέλεσαν οι εφαρμογές διαχείρισης κυκλοφορίας. Η διαδικασία

[7] Διαθέσιμος στο http://www.cti.gr/RD3/GSTD.
[8] Διαθέσιμη στο http://www-inf.enst.fr/ saglio/etudes/oporto/.
[9] Διαθέσιμη στο http://www.fh-oldenburg.de/iapg/personnen/brinkhof/generator.html.

δημιουργίας των αντικειμένων λαμβάνει υπόψη παραμέτρους όπως η μέγιστη ταχύτητα και χωρητικότητα κάθε τμήματος του οδικού δικτύου, η ύπαρξη διαφορετικών "κλάσεων συμπεριφοράς κίνησης", η αλληλεπίδραση μεταξύ διαφορετικών αντικειμένων, η ύπαρξη εξωτερικών αντικειμένων κ.λπ.

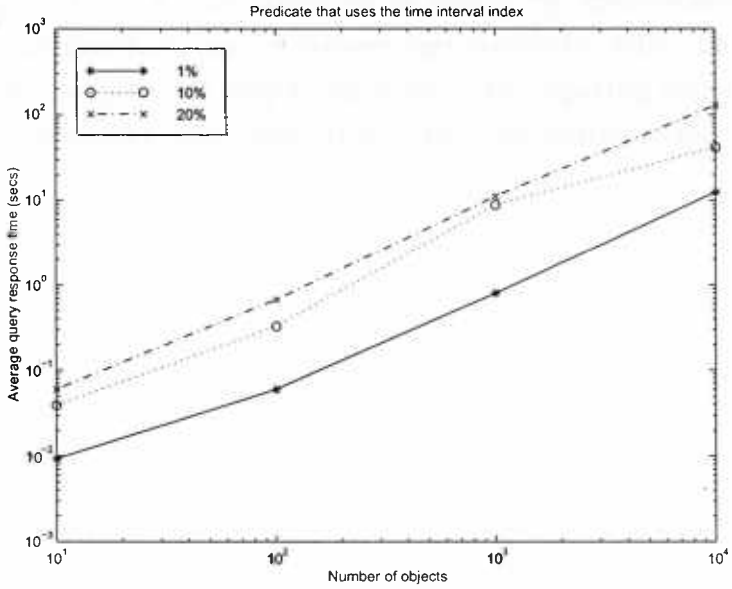
Από τις παραπάνω γεννήτριες αποφασίσαμε να χρησιμοποιήσουμε την τρίτη, μιας και ταιριάζει περισσότερο στο αρχικό κίνητρο της προτεινόμενης γλώσσας ερωτήσεων [VW01], δηλαδή τα αντικείμενα κινούμενα σε οδικά δίκτυα.

Χρησιμοποιώντας την παραπάνω γεννήτρια δημιουργήσαμε 4 σύνολα δεδομένων αποτελούμενα από 10, 100, 1000, και 10000 κινούμενα αντικείμενα, ή 134, 1626, 16305 και 165105 τμήματα τροχιάς, αντίστοιχα. Χρησιμοποιήσαμε το οδικό δίκτυο της πόλης Oldenburg (Γερμανία), που παρέχεται με το λογισμικό της γεννήτριας, και το χρονικό διάστημα [1, 100].

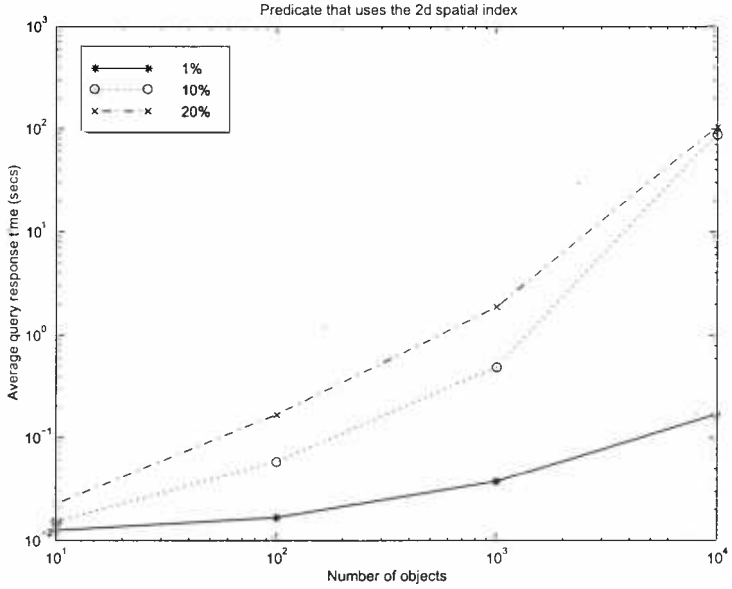
Για τον τύπο ερώτησης που χρησιμοποιεί τον χρονικό δείκτη δημιουργήσαμε τρία φορτία ερωτήσεων κάθε ένα από τα οποία αποτελείται από 100 ερωτήσεις. Για κάθε φορτίο ερωτήσεων δίνουμε τιμές στις παραμέτρους st, et και $traveltime$ με τέτοιο τρόπο ώστε να σχηματίζουν χρονικά παράθυρα μεγέθους 1%, 10%, και 20% του συνολικού μεγέθους του χρονικού χώρου. Η επιλογή του σημείου αναφοράς R δεν επηρεάζει την απόδοση της ερώτησης, αφού η εκτέλεσή της γίνεται μέσω του χρονικού δείκτη. Τα παράθυρα ερωτήσεων κατανέμονται ομοιόμορφα στο χώρο του χρόνου.

Όμοια, για τον τύπο ερώτησης που χρησιμοποιεί τον χωρικό δείκτη δείκτη δημιουργήσαμε τρία φορτία ερωτήσεων κάθε ένα από τα οποία αποτελείται από 100 ερωτήσεις. Για κάθε φορτίο ερωτήσεων δίνουμε τιμές στις παραμέτρους R και $traveldistance$ με τέτοιο τρόπο ώστε να σχηματίζουν χωρικά παράθυρα μεγέθους 1%, 10%, και 20% του συνολικού μεγέθους του χώρου, σε κάθε μια από τις δύο χωρικές διαστάσεις. Η τιμές των παραμέτρων st και et δεν επηρεάζουν την απόδοση της ερώτησης, αφού η εκτέλεσή της γίνεται μέσω του δύο διαστάσεων χωρικού δείκτη. Τα παράθυρα ερωτήσεων κατανέμονται ομοιόμορφα στο χώρο, ο οποίος έχει μέγεθος περίπου 20000 μονάδες στη x διάσταση και 30000 στην y διάσταση.

Τα σχήματα 7 και 8 παρουσιάζουν το μέσο χρόνο απόκρισης των παραπάνω φορτίων ερωτήσεων σε ένα Intel 900MHz - 256MB RAM σύστημα. Σημειώνουμε ότι εξαιτίας της διαφορετικής ανάλυσης του χρονικού και του χωρικού χώρου, οι ερωτήσεις έχουν διαφορετική επιλεξιμότητα και επομένως τα δύο γραφήματα δεν είναι μεταξύ τους συγκρίσιμα.



Σχήμα 7: Μέσος χρόνος απόκρισης για την ερώτηση που χρησιμοποιεί το χρονικό ευρετήριο



Σχήμα 8: Μέσος χρόνος απόκρισης για την ερώτηση που χρησιμοποιεί το χωρικό ευρετήριο

Τα σχήματα 7 και 8 επιβεβαιώνουν το εφικτό της προτεινόμενης γλώσσας ερωτήσεων. Επιπρόσθετα, αφού ή γλώσσα έχει υλοποιηθεί πάνω από ένα ισχυρό ΣΔΒΔ τα πλεονεκτήματα της υψηλής αξιοπιστίας, της εύρωστης απόδοσης και των μηχανισμών ανάκαμψης, αντιγράφων ασφαλείας κ.λπ. διατηρούνται.

7 Συμπεράσματα και μελλοντικές κατευθύνσεις

Υπάρχει μια μεγάλη ποικιλία εφαρμογών όπου απαιτείται η διαχείριση αντικειμένων με χωρικές ιδιότητες που μεταβάλλονται στο χρόνο. Παραδείγματα αποτελούν οι εφαρμογές διαχείρισης κυκλοφορίας, οι εφαρμογές κτηματολογίου, εφαρμογές εγγράφων πολυμέσων κ.λπ. Η υποστήριξη που παρέχουν τα παραδοσιακά συστήματα διαχείρισης βάσεων δεδομένων για αυτά τα πεδία εφαρμογών είναι περιορισμένη. Αποτελεί ανοικτό θέμα ο καθορισμός του είδους των λειτουργιών διαχείρισης δεδομένων που θα πρέπει να προσφέρουν τα παραπάνω συστήματα για τα νέα αυτά πεδία εφαρμογών, καθώς και ο καθορισμός του μοντέλου δεδομένων, της γλώσσας ερωτήσεων και του μηχανισμού δεικτοδότησης που θα υποστηρίζουν αποδοτικά αυτές τις λειτουργίες. Σε αυτή την κατεύθυνση συμβάλλει και η παρούσα εργασία, όπου:

- Πραγματοποιήσαμε μια περιεκτική επισκόπηση βασικών χωροχρονικών εννοιών καθώς και των μοντέλων δεδομένων, των γλωσσών ερωτήσεων και των δομών δεικτοδότησης για κινούμενα αντικείμενα που έχουν προταθεί στη βιβλιογραφία.

- Διερευνήσαμε τους μηχανισμούς επέκτασης λειτουργικότητας που προσφέρουν τα κυριότερα ΣΔΒΔ της αγοράς.

- Παρουσιάσαμε ένα λιτό και σαφές σύνολο τύπων δεδομένων και μια εκφραστική γλώσσα ερωτήσεων για κινούμενα αντικείμενα.

- Προτείναμε ένα ρεαλιστικό σχέδιο υλοποίησης των τύπων δεδομένων και της γλώσσας ερωτήσεων εντός του πλαισίου επέκτασης του συστήματος Oracle 9i.

- Πραγματοποιήσαμε μια πρωτότυπη υλοποίηση του προτεινόμενου σχεδίου και ελέγξαμε πειραματικά το εφικτό της υλοποίησης.

Μελλοντικές βελτιώσεις και επεκτάσεις της παρούσας εργασίας μπορούν να αναζητηθούν στις παρακάτω κατευθύνσεις:

- Αναπαράσταση της κίνησης των αντικειμένων με τη βοήθεια προτύπων (patterns) και συναρτήσεων.

- Αναζήτηση κατάλληλων αντιστοιχήσεων δομών δεικτοδότησης σε σχεσιακά σχήματα, όπως το RI-tree [KPS00], μιας και χωρίς κατάλληλες δομές το εφικτό της αποδοτικής υλοποίησης μοντέλων δεδομένων και γλωσσών ερωτήσεων είναι αμφισβητήσιμο.
- Πειραματισμός και με πλαίσια επέκτασης λειτουργικότητας πέραν του παρεχόμενου από την Oracle.
- Διερεύνηση θεμάτων αβεβαιότητας (uncertainty) που αφορούν τη θέση ενός κινούμενου αντικειμένου μεταξύ δύο δειγματοληπτημένων θέσεων (τα άκρα ενός τμήματος τροχιάς).
- Διερεύνηση θεμάτων εξόρυξης δεδομένων κίνησης (motion mining) και δυνατότητας πραγματοποίησης ερωτήσεων όπως η εύρεση όμοιων συμπεριφορών κίνησης.

Βιβλιογραφία

[All83] J. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832--843, November 1983.

[BBKM99] S. Berchtold, C. Bohm, H. Kriegel, and U. Michel. Implementation of multidimensional index structures for knowledge discovery in relational databases. In *Proceedings of the first International Conference on Data Warehousing and Knowledge Discovery, DaWaK'99*, pages 261--270, Florence, Italy, August/September 1999.

[BJS+98] M. Bohlen, C. Jensen, B. Skjellaug, D. Pfoser, and N. Tryfona. Spatiotemporal database support for legacy applications. In *Proceedings of ACM Symposium on Applied Computing, ACM-SAC'98*, pages 226--234, Atlanta, GA, USA, February/March 1998.

[Bri00] T. Brinkhoff. Generating network-based moving objects. In *Proceedings of the 12th International Conference on Scientific and Statistical Database Management, SSDBM'00*, pages 253--255, Berlin, Germany, July 2000.

[BSSJ99] R. Bliujute, S. Saltenis, G. Slivinskas, and C. Jensen. Developing a DataBlade for a new index. In *Proceedings of the 15th IEEE International Conference on Data Engineering, ICDE'99*, pages 314--323, Sydney, Australia, March 1999.

[CCF+99] W. Chen, J. Chow, Y. Fuh, J. Grandbois, M. Jou, N. Mattos, B. Tran, and Y. Wang. High level indexing of user-defined types. In *Proceedings of the 25th International Conference on Very Large Data Bases, VLDB'99*, pages 554--564, Edinburgh - Scotland - UK, September 1999.

[CG94] T. Cheng and S. Gadia. A pattern matching language for spatio-temporal databases. In *Proceedings of the third International Conference on Information Knowledge and Management, CIKM'94*, pages 288--295, Gaithersburg, MD, USA, November/December 1994.

[CR99] J. Chomicki and P. Revesz. A geometric framework for specifying spatiotemporal objects. In *Proceedings of the sixth International Workshop on Temporal Representation and Reasoning, TIME 99*, pages 41--46, Orlando, Florida, USA, May 1999.

[Ede80] H. Edelsbrunner. Dynamic rectangle intersection searching. Report 47, Institute for Information Processing, Technical University of Graz, Austria, 1980.

[EF91] M. Egenhofer and R. Franzosa. Point-set topological spatial relations. *International Journal of Geographical Information Systems*, 5(2):161--174, June 1991.

[EGSV98] M. Erwig, R. Guting, M. Schneider, and M. Vazirgiannis. Abstract and discrete modeling of spatio-temporal data types. In *Proceedings of the 6th ACM International Workshop on Advances in Geographic Information Systems, ACM-GIS '98*, pages 131--136, Washington, DC, USA, November 1998.

[GBE+00] R. Guting, M. Bohlen, M. Erwig, C. Jensen, N. Lorentzos, M. Schneider, and M. Vazirgiannis. A foundation for representing and querying moving objects. *ACM Transactions on Database Systems*, 25(1):1--42, March 2000.

[GRS98] S. Grumbach, P. Rigaux, and L. Segoufin. Spatio-temporal data handling with constraints. In *Proceedings of ACM Symposium on Geographic Information Systems, ACM-GIS 98*, pages 106--111, Washington, D.C., USA, November 1998.

[Gut84] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of ACM SIGMOD International Conference on Management of Data, SIGMOD'84*, pages 47--57, Boston, Massachusetts, USA, June 1984.

[HNP95] J. Hellerstein, J. Naughton, and A. Pfeffer. Generalized search tress for database systems. In *Proceedings of the 21st International Conference on Very Large Data Bases, VLDB'95*, pages 562--573, Zurich, Switzerland, Septmber 1995.

[IBM99] *IBM DB2 Universal Database Application Guide, Version 6*. IBM Corporation, 1999.

[Inf98] *DataBlade Developers Kit User's Guide*. Informix Software, Inc., 1998.

[KPS00] H. Kriegel, M. Potke, and T. Seidl. Managing intervals efficiently in object-relational databases. In *Proceedings of the 26th International Conference on Very Large Data Bases, VLDB'00*, pages 407--418, Cairo, Egypt, September 2000.

[KPS01] H. Kriegel, M. Potke, and Thomas Seidl. Object-relational indexing for general interval relationships. In *Proceedings of the 7th International Symposium on Spatial and Temporal Databases, SSTD 2001*, pages 522--542, Los Angeles, CA, USA, July 2001.

[KRSB99] R. Kanth, S. Ravada, J. Sharma, and J. Banerjee. Indexing medium-dimensionality data in oracle. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 521--522, Philadelphia, Pennsylvania, USA, June 1999.

[NS98] M. Nascimento and J. Silva. Towards Historical R-trees. In *Proceedings of ACM Symposium on Applied Computing, ACM-SAC'98*, pages 235--240, Atlanta, GA, USA, February/March 1998.

[NST99] M. Nascimento, J. Silva, and Y. Theodoridis. Evaluation of access structures for discretely moving points. In *Proceedings of International Workshop on Spatio-Temporal Database Management, STDBM'99*, pages 171--188, Edinburgh, Scotland, UK, Septmber 1999.

[Ora01a] *Oracle Data Cartridge User's Guide and Reference, Release 9.0.1.* Oracle Corporation, 2001.

[Ora01b] *Oracle Spatial Cartridge Developer's Guide and Reference, Release 9.0.1.* Oracle Corporation, 2001.

[PJ99] D. Pfoser and C.S. Jensen. Capturing the uncertainty of moving-object representations. In *Proceedings of the sixth International Symposium on Spatial Databases, SSD'99*, pages 111--132, Hong Kong, China, July 1999.

[PJT00] D. Pfoser, C. Jensen, and Y. Theodoridis. Novel approaches to the indexing of moving object trajectories. In *Proceedings of the 26th International Conference on Very Large Data Bases, VLDB'00*, pages 395--406, Cairo, Egypt, September 2000.

[PS93] F. Preparata and M. Shamos. *Computational Geometry: An Introduction*. Springer, 1993.

[PSZ99] C. Parent, S. Spaccapietra, and E. Zimαnyi. Spatio-temporal conceptual models: Data structures + space + time. In *Proceedings of ACM Symposium on Geographic Information Systems, ACM-GIS'99*, pages 26--33, Kansas City - MO, USA, November 1999.

[PT98] D. Pfoser and N. Tryfona. Requirements, Definitions, and Notations for Spatiotemporal Application Environments. In *Proceedings of ACM Symposium on Geographic Information Systems, ACM-GIS'98*, pages 124--130, Washington, D.C., USA, November 1998.

[PT00] D. Pfoser and Y. Theodoridis. Generating semantics-based trajectories of moving objects. In *Proceedings of International Workshop on Emerging Technologies for Geo-Based Applications*, Ascona, Switzerland, May 2000.

[RSV02] P. Rigaux, M. Scholl, and A. Voisard. *Spatial Databases ans Application to GIS*. Morgan Kaufmann, 2002.

[Sam90] H. Samet. *Applications of Spatial Data Structures*. Addison-Wesley, 1990.

[SDF+00] J. Srinivasan, S. Das, C. Freiwald, E. Chong, M. Jagannath, A. Yalamanchi, R. Krishnan, A. Tran, S. DeFazio, and J. Banerjee. Oracle8i index-organized table and its application to new domains. In *Proceedings of the 26th International Conference on Very Large Data Bases, VLDB'00*, pages 285--296, Cairo, Egypt, September 2000.

[SEG+01] B. Seeger, M. Egenhofer, J. Gray, S. Leutenegger, and D. Papadias. Seeking the truth-curses and blessings of experiments. *Panel discussion in the 7th International Symposium on Spatial and Temporal Databases, SSTD'01, Los Angeles, CA, USA*, July 2001.

[SJ02] S. Saltenis and C. Jensen. Indexing of moving objects for location-based services. In *Proceedings of the 18th IEEE Conference on Data Engineering, ICDE'02*, San Jose, CA, USA, February/March 2002.

[SJLL00] S. Saltenis, C. Jensen, S. Leutenegger, and M. Lopez. Indexing the positions of continuously moving objects. In *Proceedings of ACM SIGMOD International Conference on Management of Data, SIGMOD'00*, pages 331--342, Dallas, TX, USA, May 2000.

[SM00] J. Saglio and J. Moreira. Oporto: a realistic scenario generator for moving objects. *Geoinformatica*, 5(1):71--93, March 2000.

[SMS+00] J. Srinivasan, R. Murthy, S. Sundara, N. Agarwal, and S. DeFazio. Extensible indexing: A framework for integrating domain-specific indexing schemes into Oracle8i. In *Proceedings of the 16th IEEE International Conference on Data Engineering, ICDE'00*, pages 91--100, San Diego, California, February/March 2000.

[Sto86] M. Stonebraker. Inclusion of new type in relational database systems. In *Proceedings of the second IEEE Conference on Data Engineering, ICDE'86*, pages 262--269, 1986.

[SWCD97] A. Sistla, O. Wolfson, O. Chamberlain, and S. Dao. Modeling and querying moving objects. In *Proceedings of the 13th IEEE Conference on Data Engineering, ICDE'97*, pages 422--432, Birmingham, UK, April 1997.

[TJ99] N. Tryfona and C. Jensen. Conceptual data modeling for spatiotemporal applications. *Geoinformatica*, 3(3):245--268, September 1999.

[TN00] Y. Theodoridis and M. Nascimento. Generating spatiotemporal datasets on the WWW. *SIGMOD Record*, 29(3):39--43, September 2000.

[TP01] Y. Tao and D. Papadias. Efficient Historical R-trees. In *Proceedings of the 13th IEEE Conference on Scientific and Statistical Database Management, SSDBM'01*, Fairfax, Virginia, USA, July 2001.

[TVS96] Y. Theodoridis, M. Vazirgiannis, and T. Sellis. Spatio-temporal indexing for large multimedia applications. In *Proceedings of the third IEEE Conference on Multimedia Computing and Systems, ICMCS'96*, Hiroshima, Japan, June 1996.

[VW01] M. Vazirgiannis and O. Wolfson. A spatiotemporal query language for moving point objects. In *Proceedings of the 7th International Symposium on Spatial and Temporal Databases, SSTD'01*, pages 20--35, Los Angeles, CA, USA, July 2001.

[Wor94] M. Worboys. A unified model for spatial and temporal information. *The Computer Journal*, 37(1):26--34, 1994.

[WXCJ98] O. Wolfson, B. Xu, O. Chamberlain, and L. Jiang. Moving objects databases: Issues and solutions. In *Proceedings of the 10th IEEE Conference on Scientific and Statistical Database Management, SSDBM'98*, pages 123--132, Capri, Italy, July 1998.

Α Παράρτημα - Υλοποίηση

```
CREATE TABLE TRAJECTORIES(
   moid    NUMBER(38) NOT NULL, -- moving object id
   segid   NUMBER(38) NOT NULL, -- trajectory segment id
   node    NUMBER(38),          -- artificial node for RI-tree
   lower   NUMBER(38),          -- lower time interval bound
   upper   NUMBER(38),          -- upper time interval bound
   spatseg MDSYS.SDO_GEOMETRY   -- spatial segment
);

-- An index on moving objects ids
CREATE INDEX mo_idx ON TRAJECTORIES(moid);

-- Two composite indexes for time interval indexing
CREATE INDEX lower_idx ON TRAJECTORIES(node, lower, segid, moid);
CREATE INDEX upper_idx ON TRAJECTORIES(node, upper, segid, moid);

-- Update spatial meta data view
-- This is required before the spatial index is created
INSERT INTO USER_SDO_GEOM_METADATA
  VALUES('TRAJECTORIES',
         'spatseg',
         MDSYS.SDO_DIM_ARRAY( -- Workspace size and tolerance
            MDSYS.SDO_DIM_ELEMENT('X', 0, 30000, 0.5),
            MDSYS.SDO_DIM_ELEMENT('Y', 0, 30000, 0.5)),
         NULL -- SRID spatial reference system id
);

-- An index for the spatial segments
CREATE INDEX spatseg_idx ON TRAJECTORIES(spatseg)
   INDEXTYPE IS MDSYS.SPATIAL_INDEX;

-- A table for storing parameters of RI-tree virtual backbone
CREATE TABLE RITPARAMS(root NUMBER(38));
-- supported range (0, 101)
INSERT INTO RITPARAMS VALUES(101);

-- These type definitions are useful for creating
-- in memory lists which are used for time
-- time interval intersection queries
CREATE TYPE NODETYPE AS OBJECT(NODE NUMBER);
```

```
/
CREATE TYPE NODELIST AS TABLE OF NODETYPE;
/

-- INSERT PROCEDURE
CREATE OR REPLACE PROCEDURE
   INSERTTRAJSEG(moid    NUMBER,  -- moving object id
                 segid   NUMBER,  -- trajectory segment id
                 lower   NUMBER,  -- lower time interval bound
                 upper   NUMBER,  -- upper time interval bound
                 x1      REAL,    -- x coor of start point
                 y1      REAL,    -- y coor of start point
                 x2      REAL,    -- x coor of end point
                 y2      REAL)    -- y coor of end point
IS

node NUMBER;
step NUMBER;

BEGIN

   SELECT RITPARAMS.root INTO node FROM RITPARAMS;
   step := node / 2;

   -- Compute the fork (artificial) node of RI-tree
   LOOP
      EXIT WHEN step < 1;
      IF upper < node THEN
         node := node - step;
      ELSIF node < lower THEN
         node := node + step;
      ELSE
         EXIT;
      END IF;
      step := step / 2;
   END LOOP;

   -- ... AND INSERT!
   INSERT INTO TRAJECTORIES VALUES(moid, segid, node, lower, upper,
      MDSYS.SDO_GEOMETRY(2002,    -- 2 dimesional line string
                         NULL,    -- No Spatial Reference System associated
```

```
                    NULL,  -- Ignored
                    MDSYS.SDO_ELEM_INFO_ARRAY(1,2,1), -- line string
                    MDSYS.SDO_ORDINATE_ARRAY(x1, y1, x2, y2)));

END;
/

-- PREPARATION PROCEDURE FOR TIME INTERVAL INTERSECTION QUERY
-- (FIRST PROCEDURAL STEP)
CREATE OR REPLACE PROCEDURE
   PREPTIMEINTERSECT(lower NUMBER, -- lower limit of query interval
                     upper NUMBER, -- upper limit of query interval
                     leftNodes IN OUT NODELIST, -- list to store leftnodes
                     rightNodes IN OUT NODELIST -- list to store rightnodes
                      )
IS

node NUMBER;
step NUMBER;
lstep NUMBER;
rstep NUMBER;
fork NUMBER;
lcount INTEGER;
rcount INTEGER;

BEGIN

   SELECT RITPARAMS.root INTO node FROM RITPARAMS;
   lcount := 1;
   rcount := 1;

   step := node / 2;
   LOOP
      EXIT WHEN step < 1;
      IF upper < node THEN
         rightNodes.EXTEND;
         rightNodes(rcount) := NODETYPE(node);
         rcount := rcount + 1;
         node := node - step;
      ELSIF node < lower THEN
         leftNodes.EXTEND;
```

```
      leftNodes(lcount) := NODETYPE(node);
      lcount := lcount + 1;
      node := node + step;
   ELSE
      fork := node;
      EXIT;
   END IF;
   step := step / 2;
END LOOP;

IF lower < fork THEN
   node := fork - step;
   lstep := step / 2;
   LOOP
      EXIT WHEN step < 1;
      IF node < lower THEN
         leftNodes.EXTEND;
         leftNodes(lcount) := NODETYPE(node);
         lcount := lcount + 1;
         node := node + lstep;
      ELSIF lower < node THEN
         leftNodes.EXTEND;
         leftNodes(lcount) := NODETYPE(node);
         lcount := lcount + 1;
         node := node -lstep;
      ELSE
         EXIT;
      END IF;
      lstep := lstep / 2;
   END LOOP;
END IF;

IF fork < upper THEN
   node := fork + step;
   rstep := step / 2;
   LOOP
      EXIT WHEN step < 1;
      IF node < upper THEN
         rightNodes.EXTEND;
         rightNodes(rcount) := NODETYPE(node);
         rcount := rcount + 1;
```

```
            node := node + rstep;
         ELSIF upper < node THEN
            rightNodes.EXTEND;
            rightNodes(rcount) := NODETYPE(node);
            rcount := rcount + 1;
            node := node - rstep;
         ELSE
            EXIT;
         END IF;
         rstep := rstep / 2;
      END LOOP;
   END IF;

END;
/

-- TYPICAL TIME INTERVAL INTERSECTION QUERY
CREATE OR REPLACE PROCEDURE TESTINTQUER(lin IN NUMBER, uin IN NUMBER, moidin NUM
IS

ln NODELIST := NODELIST();
rn NODELIST := NODELIST();

CURSOR c1 IS
SELECT node from TABLE(CAST(ln AS NODELIST)) l;

CURSOR cr IS
SELECT node from TABLE(CAST(rn AS NODELIST)) r;

CURSOR c1 IS
SELECT /*+ ORDERED INDEX (t upper_idx) USE_NL (l t) */
segid FROM TABLE(CAST(ln AS NODELIST)) l, TRAJECTORIES t
WHERE t.node = l.node AND t.upper >= lin AND t.moid = moidin
UNION ALL
SELECT /*+ ORDERED INDEX (t lower_idx) USE_NL (r t) */
segid FROM TABLE(CAST(rn AS NODELIST)) r, TRAJECTORIES t
WHERE t.node = r.node AND t.lower <= uin AND t.moid = moidin
UNION ALL
SELECT segid FROM TRAJECTORIES t
WHERE t.node BETWEEN lin AND uin AND t.moid = moidin;
```

```
BEGIN
   PREPTIMEINTERSECT(lin, uin, ln, rn);

   dbms_output.put_line('LEFTS');
   FOR cl_rec IN cl LOOP
      dbms_output.put_line(cl_rec.node);
   END LOOP;
   dbms_output.put_line('RIGHTS');
   FOR cr_rec IN cr LOOP
      dbms_output.put_line(cr_rec.node);
   END LOOP;
   dbms_output.put_line('RES');
   FOR c1_rec IN c1 LOOP
      dbms_output.put_line(c1_rec.segid);
   END LOOP;
END;
/

-- LOC PREDICATE
CREATE OR REPLACE FUNCTION
   LOCATEMO(lmoid NUMBER, time NUMBER)
   RETURN MDSYS.SDO_GEOMETRY
IS

x1 REAL;
x2 REAL;
y1 REAL;
y2 REAL;
t1 NUMBER;
t2 NUMBER;
x REAL;
y REAL;
seg MDSYS.SDO_GEOMETRY;
ln NODELIST := NODELIST();
rn NODELIST := NODELIST();

CURSOR c1 IS
SELECT /*+ ORDERED INDEX (t upper_idx) USE_NL (l t) */
t.spatseg, t.upper, t.lower
FROM TABLE(CAST(ln AS NODELIST)) l, TRAJECTORIES t
```

```
WHERE t.node = l.node AND t.upper >= time AND t.moid = lmoid
UNION ALL
SELECT /*+ ORDERED INDEX (t lower_idx) USE_NL (r t) */
t.spatseg, t.upper, t.lower
FROM TABLE(CAST(rn AS NODELIST)) r, TRAJECTORIES t
WHERE t.node = r.node AND t.lower <= time AND t.moid = lmoid
UNION ALL
SELECT
t.spatseg, t.upper, t.lower
FROM TRAJECTORIES t
WHERE t.node BETWEEN time AND time AND t.moid = lmoid;

BEGIN

   PREPTIMEINTERSECT(time,time, ln, rn);
   FOR c1_rec IN c1 LOOP
      x1 := c1_rec.spatseg.SDO_ORDINATES(1);
      y1 := c1_rec.spatseg.SDO_ORDINATES(2);
      x2 := c1_rec.spatseg.SDO_ORDINATES(3);
      y2 := c1_rec.spatseg.SDO_ORDINATES(4);
      t2 := c1_rec.upper;
      t1 := c1_rec.lower;
      EXIT;
   END LOOP;

   x := x1 + ((x2 - x1) / (t2 - t1)) * (time - t1);
   y := y1 + ((y2 - y1) / (t2 - t1)) * (time - t1);


   RETURN MDSYS.SDO_GEOMETRY(2001,    -- 2 dimesional point
                        NULL,   -- No Spatial Reference System associated
                            MDSYS.SDO_POINT_TYPE(x,y,NULL),
                            NULL,
                            NULL);

END;
/

--SOMEWHERE STO STORE THE RESULTS
CREATE TABLE WHENATRES(time NUMBER);
```

```
-- WHENAT PREDICATE
CREATE OR REPLACE PROCEDURE
   WHENAT(lmoid NUMBER, xcoor REAL, ycoor REAL)

IS

x1 REAL;
x2 REAL;
y1 REAL;
y2 REAL;
t1 NUMBER;
t2 NUMBER;
t NUMBER;
seg MDSYS.SDO_GEOMETRY;

CURSOR c1 IS
      SELECT t.spatseg, t.upper, t.lower
      FROM TRAJECTORIES t
      where SDO_RELATE(t.spatseg, MDSYS.SDO_GEOMETRY(2001, NULL,
                              MDSYS.SDO_POINT_TYPE(xcoor,ycoor,NULL),NULL,
                              NULL), 'mask = ANYINTERACT') = 'TRUE'
      AND t.moid = lmoid;

BEGIN

   DELETE FROM WHENATRES;
   FOR c1_rec IN c1 LOOP
      x1 := c1_rec.spatseg.SDO_ORDINATES(1);
      y1 := c1_rec.spatseg.SDO_ORDINATES(2);
      x2 := c1_rec.spatseg.SDO_ORDINATES(3);
      y2 := c1_rec.spatseg.SDO_ORDINATES(4);
      t2 := c1_rec.upper;
      t1 := c1_rec.lower;
      t := t1 + (xcoor - x1) / (x2 - x1) * (t2 - t1);
      -- INSERT INTO WHENATRES VALUES(t);
   END LOOP;

END;
/

-- PREDICATE WITHIN_T_EP_AB
```

```
-- T  > TRAVEL TIME
-- EP > ALONG EXISTING PATH
-- AB > ALWAYS BETWEEN starttime AND endtime

CREATE OR REPLACE PROCEDURE WITHIN_T_EP_AB(time      NUMBER,
                                           starttime NUMBER,
                                           endtime   NUMBER,
                                           xcoor     REAL,
                                           ycoor     REAL)
IS


x1 REAL;
x2 REAL;
y1 REAL;
y2 REAL;
t1 NUMBER;
t2 NUMBER;
t REAL;
uplimit NUMBER;
lolimit NUMBER;
TYPE NUMARRAY IS TABLE OF NUMBER;
candidates NUMARRAY := NUMARRAY();
candindex INTEGER;
currentmoid NUMBER;
currentsegid NUMBER;
ln NODELIST := NODELIST();
rn NODELIST := NODELIST();
consecutive BOOLEAN;
intersectsR BOOLEAN;
within BOOLEAN;
valid BOOLEAN;
interdim MDSYS.SDO_DIM_ARRAY;
pointR MDSYS.SDO_GEOMETRY;
lastupper NUMBER;
localstart NUMBER;
i INTEGER;

CURSOR c1 IS
   SELECT /*+ ORDERED INDEX (t upper_idx) USE_NL (l t) */
   t.moid, t.segid, t.spatseg, t.upper, t.lower
```

```
    FROM TABLE(CAST(ln AS NODELIST)) l, TRAJECTORIES t
    WHERE t.node = l.node AND t.upper >= lolimit
    UNION ALL
    SELECT /*+ ORDERED INDEX (t lower_idx) USE_NL (r t) */
    t.moid, t.segid, t.spatseg, t.upper, t.lower
    FROM TABLE(CAST(rn AS NODELIST)) r, TRAJECTORIES t
    WHERE t.node = r.node AND t.lower <= uplimit
    UNION ALL
    SELECT
    t.moid, t.segid, t.spatseg, t.upper, t.lower
    FROM TRAJECTORIES t
    WHERE t.node BETWEEN lolimit AND uplimit
    ORDER BY moid ASC, lower ASC;

BEGIN

    lolimit := starttime;
    uplimit := endtime + time;
    PREPTIMEINTERSECT(lolimit, uplimit, ln, rn);
    pointR := MDSYS.SDO_GEOMETRY(2001, NULL,
                              MDSYS.SDO_POINT_TYPE(xcoor,ycoor,NULL),
                                   NULL, NULL);
    currentmoid := -1;
    currentsegid := -1;
    SELECT DIMINFO INTO interdim FROM USER_SDO_GEOM_METADATA
    WHERE TABLE_NAME = 'TRAJECTORIES';
    consecutive := false;
    intersectsR := false;
    within := false;
    candindex := 0;

    FOR c1_rec IN c1 LOOP
      IF c1_rec.segid = currentsegid AND c1_rec.moid = currentmoid THEN
        GOTO end_of_loop; -- There isn't a CONTINUE statement in PL/SQL ?
       ELSE
          currentsegid := c1_rec.segid;
       END IF;
       IF (c1_rec.moid != currentmoid) THEN
          -- DO SOMETHING WITH THE FINISHED OBJECT
        IF (currentmoid != -1) AND (consecutive AND intersectsR AND within) THEN
              candidates.EXTEND;
```

```
            candindex := candindex + 1;
            candidates(candindex) := currentmoid;
      END IF;
      currentmoid := c1_rec.moid;
      localstart := starttime;
      consecutive := TRUE;
      intersectsR := FALSE;
      within := FALSE;
   ELSE
      IF NOT (c1_rec.lower = lastupper) THEN
         consecutive := FALSE;
      END IF;
   END IF;
   lastupper := c1_rec.upper;
IF SDO_GEOM.RELATE(c1_rec.spatseg, interdim, 'ANYINTERACT',
   pointR, interdim) = 'TRUE' THEN
      intersectsR := true;
      x1 := c1_rec.spatseg.SDO_ORDINATES(1);
      y1 := c1_rec.spatseg.SDO_ORDINATES(2);
      x2 := c1_rec.spatseg.SDO_ORDINATES(3);
      y2 := c1_rec.spatseg.SDO_ORDINATES(4);
      t2 := c1_rec.upper;
      t1 := c1_rec.lower;
      t := t1 + (xcoor - x1) / (x2 - x1) * (t2 - t1);
      valid := true;
      IF (t < localstart) OR (t > endtime + time) THEN
         valid := FALSE;
      ELSIF (t - localstart > time) THEN
         valid := FALSE;
      ELSIF (t < endtime) THEN
         valid := false;
         localstart := t;
      END IF;
      within := within OR valid;
   END IF;
   -- dbms_output.put_line(currentmoid);
   -- dbms_output.put_line(currentsegid);
-- if consecutive THEN dbms_output.put_line('consecutive TRUE');
   -- ELSE dbms_output.put_line('consecutive FALSE');END IF;
-- if intersectsR THEN dbms_output.put_line('intersectsR TRUE');
   -- ELSE dbms_output.put_line('intersectsR FALSE');END IF;
```

```
      -- if within THEN dbms_output.put_line('within TRUE');
      -- ELSE dbms_output.put_line('within FALSE');END IF;
      <<end_of_loop>>
      NULL;
   END LOOP;

   -- Some processing for the last moving object
   IF (consecutive AND intersectsR AND within) THEN
      candidates.EXTEND;
      candindex := candindex + 1;
      candidates(candindex) := currentmoid;
   END IF;

   --i := 1;
   --LOOP
      --EXIT WHEN i > candindex;
      --dbms_output.put_line(candidates(i));
      --i := i + 1;
   --END LOOP;

END;
/

-- PREDICATE WITHIN_D_EP_AB
-- D  > DISTANCE
-- EP > ALONG EXISTING PATH
-- AB > ALWAYS BETWEEN starttime AND endtime

CREATE OR REPLACE PROCEDURE WITHIN_D_EP_AB(dist      REAL,
                                           starttime NUMBER,
                                           endtime   NUMBER,
                                           xcoor     REAL,
                                           ycoor     REAL)
IS


x1 REAL;
x2 REAL;
y1 REAL;
y2 REAL;
t1 NUMBER;
```

```
t2 NUMBER;
t REAL;
TYPE NUMARRAY IS TABLE OF NUMBER;
candidates NUMARRAY := NUMARRAY();
candindex INTEGER;
currentmoid NUMBER;
currentsegid NUMBER;
consecutive BOOLEAN;
intersectsR BOOLEAN;
within BOOLEAN;
valid BOOLEAN;
interdim MDSYS.SDO_DIM_ARRAY;
pointR MDSYS.SDO_GEOMETRY;
lastupper NUMBER;
locallowerx REAL;
locallowery REAL;
distfromst REAL;
i INTEGER;


CURSOR c1 IS
   SELECT /*+ INDEX (t spatseg_idx) */
   t.moid, t.segid, t.spatseg, t.upper, t.lower
   FROM TRAJECTORIES t
   WHERE SDO_RELATE(t.spatseg,
      MDSYS.SDO_GEOMETRY(2003, NULL, NULL,
      MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,3),
     MDSYS.SDO_ORDINATE_ARRAY(xcoor - dist, ycoor - dist, xcoor + dist, ycc
      'mask = ANYINTERACT') = 'TRUE'
   ORDER BY t.moid ASC, t.lower ASC;


BEGIN

   pointR := MDSYS.SDO_GEOMETRY(2001, NULL,
                          MDSYS.SDO_POINT_TYPE(xcoor,ycoor,NULL),
                               NULL, NULL);
   currentmoid := -1;
   currentsegid := -1;
   SELECT DIMINFO INTO interdim FROM USER_SDO_GEOM_METADATA
   WHERE TABLE_NAME = 'TRAJECTORIES';
```

```
candindex := 0;

FOR c1_rec IN c1 LOOP
  IF c1_rec.segid = currentsegid AND c1_rec.moid = currentmoid THEN
    GOTO end_of_loop; -- There isn't a CONTINUE statement in PL/SQL ?
  ELSE
     currentsegid := c1_rec.segid;
  END IF;
  IF (c1_rec.moid != currentmoid) THEN
     -- DO SOMETHING WITH THE FINISHED OBJECT
    IF (currentmoid != -1) AND (consecutive AND intersectsR AND within) THEN
           candidates.EXTEND;
           candindex := candindex + 1;
           candidates(candindex) := currentmoid;
     END IF;
     currentmoid := c1_rec.moid;
     consecutive := TRUE;
     intersectsR := FALSE;
     within := FALSE;
  ELSE
     IF NOT (c1_rec.lower = lastupper) THEN
        consecutive := FALSE;
     END IF;
  END IF;
  lastupper := c1_rec.upper;
  x1 := c1_rec.spatseg.SDO_ORDINATES(1);
  y1 := c1_rec.spatseg.SDO_ORDINATES(2);
  x2 := c1_rec.spatseg.SDO_ORDINATES(3);
  y2 := c1_rec.spatseg.SDO_ORDINATES(4);
  t1 := c1_rec.lower;
  t2 := c1_rec.upper;
  IF starttime >= c1_rec.upper THEN
     GOTO end_of_loop;
  ELSIF starttime >= c1_rec.lower THEN
    locallowerx := x1 + (starttime - t1) / (t2 - t1) * (x2 - x1);
    locallowery := y1 + (starttime - t1) / (t2 - t1) * (y2 - y1);
     distfromst := 0;
  ELSE
     locallowerx := x1;
     locallowery := y1;
  END IF;
```

```
   IF SDO_GEOM.RELATE(c1_rec.spatseg, interdim, 'ANYINTERACT',
    pointR, interdim) = 'TRUE' THEN
       intersectsR := true;
       t := t1 + (xcoor - x1) / (x2 - x1) * (t2 - t1);
       valid := true;
       IF (t < starttime) THEN
          valid := FALSE;
      ELSIF distfromst + SQRT((xcoor - locallowerx) * (xcoor - locallowerx) +
      (ycoor - locallowery) * (ycoor - locallowery)) >dist  THEN
          valid := FALSE;
       ELSIF (t < endtime) THEN
          valid := false;
        distfromst := SQRT((x2 - xcoor)*(x2 - xcoor)+(y2-ycoor)*(y2-ycoor));
       END IF;
       within := within OR valid;
    ELSE
      distfromst := distfromst + SQRT((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1));
    END IF;
    -- dbms_output.put_line(currentmoid);
    -- dbms_output.put_line(currentsegid);
   -- if consecutive THEN dbms_output.put_line('consecutive TRUE');
    -- ELSE dbms_output.put_line('consecutive FALSE');END IF;
   -- if intersectsR THEN dbms_output.put_line('intersectsR TRUE');
    -- ELSE dbms_output.put_line('intersectsR FALSE');END IF;
    -- if within THEN dbms_output.put_line('within TRUE');
    -- ELSE dbms_output.put_line('within FALSE');END IF;
    <<end_of_loop>>
    NULL;
END LOOP;

-- Some processing for the last moving object
IF (consecutive AND intersectsR AND within) THEN
    candidates.EXTEND;
    candindex := candindex + 1;
    candidates(candindex) := currentmoid;
END IF;

--i := 1;
--LOOP
    --EXIT WHEN i > candindex;
    --dbms_output.put_line(candidates(i));
```

```
      --i := i + 1;
   --END LOOP;

END;
/

-- PREDICATE WITHIN_D_SP_AB
-- D  > DISTANCE
-- SP > ALONG SHORTEST PATH
-- AB > ALWAYS BETWEEN starttime AND endtime

CREATE OR REPLACE PROCEDURE WITHIN_D_SP_AB(dist      REAL,
                                           starttime NUMBER,
                                           endtime   NUMBER,
                                           xcoor     REAL,
                                           ycoor     REAL)
IS


x1 REAL; x2 REAL; xx REAL; xx1 REAL; xx2 REAL;
y1 REAL; y2 REAL; yy REAL; yy1 REAL; yy2 REAL;
t1 NUMBER;
t2 NUMBER;
t REAL;
TYPE NUMARRAY IS TABLE OF NUMBER;
candidates NUMARRAY := NUMARRAY();
candindex INTEGER;
currentmoid NUMBER;
currentsegid NUMBER;
consecutive BOOLEAN;
within BOOLEAN;
valid BOOLEAN;
interdim MDSYS.SDO_DIM_ARRAY;
pointR MDSYS.SDO_GEOMETRY;
segment MDSYS.SDO_GEOMETRY;
lastupper NUMBER;
i INTEGER;


CURSOR c1 IS
   SELECT /*+ INDEX (t spatseg_idx) */
```

```
    t.moid, t.segid, t.spatseg, t.upper, t.lower
    FROM TRAJECTORIES t
    WHERE SDO_RELATE(t.spatseg,
       MDSYS.SDO_GEOMETRY(2003, NULL, NULL,
       MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,3),
       MDSYS.SDO_ORDINATE_ARRAY(xx1, yy1, xx2, yy2)),
       'mask = ANYINTERACT') = 'TRUE'
    ORDER BY t.moid ASC, t.lower ASC;


BEGIN

    pointR := MDSYS.SDO_GEOMETRY(2001, NULL,
                                 MDSYS.SDO_POINT_TYPE(xcoor,ycoor,NULL),
                                    NULL, NULL);
    currentmoid := -1;
    currentsegid := -1;
    SELECT DIMINFO INTO interdim FROM USER_SDO_GEOM_METADATA
    WHERE TABLE_NAME = 'TRAJECTORIES';
    candindex := 0;
    xx1 := xcoor - dist;
    xx2 := xcoor + dist;
    yy1 := ycoor - dist;
    yy2 := ycoor + dist;

    FOR c1_rec IN c1 LOOP
      IF c1_rec.segid = currentsegid AND c1_rec.moid = currentmoid THEN
        GOTO end_of_loop; -- There isn't a CONTINUE statement in PL/SQL ?
       ELSE
          currentsegid := c1_rec.segid;
       END IF;
       IF (c1_rec.moid != currentmoid) THEN
          -- DO SOMETHING WITH THE FINISHED OBJECT
         IF (currentmoid != -1) AND (consecutive AND within) THEN
               candidates.EXTEND;
               candindex := candindex + 1;
               candidates(candindex) := currentmoid;
          END IF;
          currentmoid := c1_rec.moid;
          consecutive := TRUE;
          within := FALSE;
```

```
ELSE
   IF NOT (c1_rec.lower = lastupper) THEN
      consecutive := FALSE;
   END IF;
END IF;
lastupper := c1_rec.upper;
x1 := c1_rec.spatseg.SDO_ORDINATES(1);
y1 := c1_rec.spatseg.SDO_ORDINATES(2);
x2 := c1_rec.spatseg.SDO_ORDINATES(3);
y2 := c1_rec.spatseg.SDO_ORDINATES(4);
t1 := c1_rec.lower;
t2 := c1_rec.upper;
segment := MDSYS.SDO_GEOMETRY(2002, NULL, NULL,
MDSYS.SDO_ELEM_INFO_ARRAY(1, 2, 1),
MDSYS.SDO_ORDINATE_ARRAY(x1, y1, x2, y2));
IF (t2 < starttime) OR (t1 > endtime) THEN
   GOTO end_of_loop;
END IF;
IF starttime > t1 THEN
   xx := x1 + (starttime - t1) / (t2 - t1) * (x2 - x1);
   yy := y1 + (starttime - t1) / (t2 - t1) * (y2 - y1);
   segment := MDSYS.SDO_GEOMETRY(2002, NULL, NULL,
   MDSYS.SDO_ELEM_INFO_ARRAY(1, 2, 1),
   MDSYS.SDO_ORDINATE_ARRAY(xx, yy,
   segment.SDO_ORDINATES(3), segment.SDO_ORDINATES(4)));
END IF;
IF endtime < t2 THEN
   xx := x1 + (endtime - t1) / (t2 - t1) * (x2 - x1);
   yy := y1 + (endtime - t1) / (t2 - t1) * (y2 - y1);
   segment := MDSYS.SDO_GEOMETRY(2002, NULL, NULL,
   MDSYS.SDO_ELEM_INFO_ARRAY(1, 2, 1),
   MDSYS.SDO_ORDINATE_ARRAY(segment.SDO_ORDINATES(1),
   segment.SDO_ORDINATES(2), xx, yy));
END IF;
IF SDO_GEOM.WITHIN_DISTANCE(segment, interdim, dist,
pointR, interdim) = 'TRUE' THEN
   valid := true;
ELSE
   valid := false;
END IF;
within := within AND valid;
```

```
      <<end_of_loop>>
      NULL;
   END LOOP;

   -- Some processing for the last moving object
   IF (consecutive AND within) THEN
      candidates.EXTEND;
      candindex := candindex + 1;
      candidates(candindex) := currentmoid;
   END IF;

   i := 1;
   LOOP
      EXIT WHEN i > candindex;
      dbms_output.put_line(candidates(i));
      i := i + 1;
   END LOOP;

END;
/

-- PREDICATE WITHIN_T_EP_SB
-- T  > TRAVEL TIME
-- EP > ALONG EXISTING PATH
-- SB > SOMETIMES BETWEEN starttime AND endtime

CREATE OR REPLACE PROCEDURE WITHIN_T_EP_SB(time      NUMBER,
                                           starttime NUMBER,
                                           endtime   NUMBER,
                                           xcoor     REAL,
                                           ycoor     REAL)
IS

x1 REAL;
x2 REAL;
y1 REAL;
y2 REAL;
t1 NUMBER;
t2 NUMBER;
t REAL;
uplimit NUMBER;
```

```
lolimit NUMBER;
TYPE NUMARRAY IS TABLE OF NUMBER;
candidates NUMARRAY := NUMARRAY();
candindex INTEGER;
currentmoid NUMBER;
currentsegid NUMBER;
ln NODELIST := NODELIST();
rn NODELIST := NODELIST();
consecutive BOOLEAN;
intersectsR BOOLEAN;
within BOOLEAN;
valid BOOLEAN;
interdim MDSYS.SDO_DIM_ARRAY;
pointR MDSYS.SDO_GEOMETRY;
lastupper NUMBER;
localstart NUMBER;
i INTEGER;

CURSOR c1 IS
   SELECT /*+ ORDERED INDEX (t upper_idx) USE_NL (l t) */
   t.moid, t.segid, t.spatseg, t.upper, t.lower
   FROM TABLE(CAST(ln AS NODELIST)) l, TRAJECTORIES t
   WHERE t.node = l.node AND t.upper >= lolimit
   UNION ALL
   SELECT /*+ ORDERED INDEX (t lower_idx) USE_NL (r t) */
   t.moid, t.segid, t.spatseg, t.upper, t.lower
   FROM TABLE(CAST(rn AS NODELIST)) r, TRAJECTORIES t
   WHERE t.node = r.node AND t.lower <= uplimit
   UNION ALL
   SELECT
   t.moid, t.segid, t.spatseg, t.upper, t.lower
   FROM TRAJECTORIES t
   WHERE t.node BETWEEN lolimit AND uplimit
   ORDER BY moid ASC, lower ASC;

BEGIN

   lolimit := starttime;
   uplimit := endtime + time;
   PREPTIMEINTERSECT(lolimit, uplimit, ln, rn);
   pointR := MDSYS.SDO_GEOMETRY(2001, NULL,
```

```
                        MDSYS.SDO_POINT_TYPE(xcoor,ycoor,NULL),
                            NULL, NULL);
currentmoid := -1;
currentsegid := -1;
SELECT DIMINFO INTO interdim FROM USER_SDO_GEOM_METADATA
WHERE TABLE_NAME = 'TRAJECTORIES';
consecutive := false;
intersectsR := false;
within := false;
candindex := 0;

FOR c1_rec IN c1 LOOP
  IF c1_rec.segid = currentsegid AND c1_rec.moid = currentmoid THEN
    GOTO end_of_loop; -- There isn't a CONTINUE statement in PL/SQL ?
   ELSE
      currentsegid := c1_rec.segid;
   END IF;
   IF (c1_rec.moid != currentmoid) THEN
      -- DO SOMETHING WITH THE FINISHED OBJECT
     IF (currentmoid != -1) AND (consecutive AND intersectsR AND within) THEN
            candidates.EXTEND;
            candindex := candindex + 1;
            candidates(candindex) := currentmoid;
      END IF;
      currentmoid := c1_rec.moid;
      localstart := starttime;
      consecutive := TRUE;
      intersectsR := FALSE;
      within := FALSE;
   ELSE
      IF NOT (c1_rec.lower = lastupper) THEN
         consecutive := FALSE;
      END IF;
   END IF;
   lastupper := c1_rec.upper;
  IF SDO_GEOM.RELATE(c1_rec.spatseg, interdim, 'ANYINTERACT',
   pointR, interdim) = 'TRUE' THEN
      intersectsR := true;
      x1 := c1_rec.spatseg.SDO_ORDINATES(1);
      y1 := c1_rec.spatseg.SDO_ORDINATES(2);
      x2 := c1_rec.spatseg.SDO_ORDINATES(3);
```

```
            y2 := c1_rec.spatseg.SDO_ORDINATES(4);
            t2 := c1_rec.upper;
            t1 := c1_rec.lower;
            t := t1 + (xcoor - x1) / (x2 - x1) * (t2 - t1);
          within := within OR (t > starttime AND t < endtime + time);
        END IF;
        <<end_of_loop>>
        NULL;
    END LOOP;

    -- Some processing for the last moving object
    IF (consecutive AND intersectsR AND within) THEN
        candidates.EXTEND;
        candindex := candindex + 1;
        candidates(candindex) := currentmoid;
    END IF;

    i := 1;
    LOOP
        EXIT WHEN i > candindex;
        dbms_output.put_line(candidates(i));
        i := i + 1;
    END LOOP;

END;
/

-- PREDICATE WITHIN_D_EP_SB
-- D  > DISTANCE
-- EP > ALONG EXISTING PATH
-- SB > SOMETIMES BETWEEN starttime AND endtime


CREATE OR REPLACE PROCEDURE WITHIN_D_EP_SB(dist      REAL,
                                           starttime NUMBER,
                                           endtime   NUMBER,
                                           xcoor     REAL,
                                           ycoor     REAL)
IS
```

```
x1 REAL; x2 REAL; xx REAL; xx1 REAL; xx2 REAL;
y1 REAL; y2 REAL; yy REAL; yy1 REAL; yy2 REAL;
t1 NUMBER; t2 NUMBER; t REAL;
totallength REAL; currentlength REAL; locallength REAL;
TYPE NUMARRAY IS TABLE OF NUMBER;
candidates NUMARRAY := NUMARRAY();
candindex INTEGER;
currentmoid NUMBER;
currentsegid NUMBER;
consecutive BOOLEAN;
intersectsR BOOLEAN;
within BOOLEAN;
valid BOOLEAN;
interdim MDSYS.SDO_DIM_ARRAY;
pointR MDSYS.SDO_GEOMETRY;
segment MDSYS.SDO_GEOMETRY;
lastupper NUMBER;
i INTEGER;


CURSOR c1 IS
   SELECT /*+ INDEX (t spatseg_idx) */
   t.moid, t.segid, t.spatseg, t.upper, t.lower
   FROM TRAJECTORIES t
   WHERE SDO_RELATE(t.spatseg,
      MDSYS.SDO_GEOMETRY(2003, NULL, NULL,
      MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,3),
      MDSYS.SDO_ORDINATE_ARRAY(xx1, yy1, xx2, yy2)),
      'mask = ANYINTERACT') = 'TRUE'
   ORDER BY t.moid ASC, t.lower ASC;


BEGIN

   pointR := MDSYS.SDO_GEOMETRY(2001, NULL,
                              MDSYS.SDO_POINT_TYPE(xcoor,ycoor,NULL),
                                   NULL, NULL);
   currentmoid := -1;
   currentsegid := -1;
   SELECT DIMINFO INTO interdim FROM USER_SDO_GEOM_METADATA
   WHERE TABLE_NAME = 'TRAJECTORIES';
```

```
candindex := 0;
xx1 := xcoor - dist;
xx2 := xcoor + dist;
yy1 := ycoor - dist;
yy2 := ycoor + dist;

FOR c1_rec IN c1 LOOP
  IF c1_rec.segid = currentsegid AND c1_rec.moid = currentmoid THEN
    GOTO end_of_loop; -- There isn't a CONTINUE statement in PL/SQL ?
   ELSE
      currentsegid := c1_rec.segid;
   END IF;
   IF (c1_rec.moid != currentmoid) THEN
      -- DO SOMETHING WITH THE FINISHED OBJECT
     IF (currentmoid != -1) AND (consecutive AND intersectsR AND within) THEN
            candidates.EXTEND;
            candindex := candindex + 1;
            candidates(candindex) := currentmoid;
      END IF;
      currentmoid := c1_rec.moid;
      consecutive := TRUE;
      within := FALSE;
      intersectsR := TRUE;
      currentlength := 0;
   ELSE
      IF NOT (c1_rec.lower = lastupper) THEN
         consecutive := FALSE;
      END IF;
   END IF;
   lastupper := c1_rec.upper;
   x1 := c1_rec.spatseg.SDO_ORDINATES(1);
   y1 := c1_rec.spatseg.SDO_ORDINATES(2);
   x2 := c1_rec.spatseg.SDO_ORDINATES(3);
   y2 := c1_rec.spatseg.SDO_ORDINATES(4);
   t1 := c1_rec.lower;
   t2 := c1_rec.upper;
   segment := MDSYS.SDO_GEOMETRY(2002, NULL, NULL,
   MDSYS.SDO_ELEM_INFO_ARRAY(1, 2, 1),
   MDSYS.SDO_ORDINATE_ARRAY(x1, y1, x2, y2));
   IF (t2 < starttime) THEN
      GOTO end_of_loop;
```

```
    END IF;
    IF starttime > t1 THEN
       x1 := x1 + (starttime - t1) / (t2 - t1) * (x2 - x1);
       y1 := y1 + (starttime - t1) / (t2 - t1) * (y2 - y1);
       segment := MDSYS.SDO_GEOMETRY(2002, NULL, NULL,
       MDSYS.SDO_ELEM_INFO_ARRAY(1, 2, 1),
       MDSYS.SDO_ORDINATE_ARRAY(x1, y1, x2, y2));
    END IF;
    IF endtime < t2 THEN
       xx := x1 + (endtime - t1) / (t2 - t1) * (x2 - x1);
       yy := y1 + (endtime - t1) / (t2 - t1) * (y2 - y1);
       totallength := currentlength +
       SQRT((xx - x1) * (xx - x1) + (yy - y1) * (yy - y1));
    END IF;
    IF SDO_GEOM.RELATE(segment, interdim, 'ANYINTERACT',
    pointR, interdim) = 'TRUE' THEN
       intersectsR := TRUE;
       t := t1 + (xcoor - x1) / (x2 - x1) * (t2 - t1);
       valid := true;
       locallength := SQRT((xcoor - x1) * (xcoor - x1) +
       (ycoor - y1) * (ycoor - y1));
       IF (t >= starttime AND t <= endtime) OR
       (t > endtime AND
       currentlength + locallength < totallength + dist) THEN
          valid := true;
       ELSE
          valid := false;
       END IF;
       within := within OR valid;
    END IF;
    currentlength := currentlength +
    SQRT((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1));
    -- dbms_output.put_line(currentmoid);
    -- dbms_output.put_line(currentsegid);
   -- if consecutive THEN dbms_output.put_line('consecutive TRUE');
    -- ELSE dbms_output.put_line('consecutive FALSE');END IF;
    -- if within THEN dbms_output.put_line('within TRUE');
    -- ELSE dbms_output.put_line('within FALSE');END IF;
    <<end_of_loop>>
    NULL;
END LOOP;
```

```
   -- Some processing for the last moving object
   IF (consecutive AND intersectsR AND within) THEN
      candidates.EXTEND;
      candindex := candindex + 1;
      candidates(candindex) := currentmoid;
   END IF;

   i := 1;
   LOOP
      EXIT WHEN i > candindex;
      dbms_output.put_line(candidates(i));
      i := i + 1;
   END LOOP;

END;
/

-- PREDICATE WITHIN_D_SP_SB
-- D  > DISTANCE
-- SP > ALONG SHORTEST PATH
-- SB > SOMETIMES BETWEEN starttime AND endtime

CREATE OR REPLACE PROCEDURE WITHIN_D_SP_SB(dist      REAL,
                                           starttime NUMBER,
                                           endtime   NUMBER,
                                           xcoor     REAL,
                                           ycoor     REAL)
IS


x1 REAL; x2 REAL; xx REAL; xx1 REAL; xx2 REAL;
y1 REAL; y2 REAL; yy REAL; yy1 REAL; yy2 REAL;
t1 NUMBER;
t2 NUMBER;
t REAL;
TYPE NUMARRAY IS TABLE OF NUMBER;
candidates NUMARRAY := NUMARRAY();
candindex INTEGER;
currentmoid NUMBER;
currentsegid NUMBER;
```

```
consecutive BOOLEAN;
within BOOLEAN;
valid BOOLEAN;
interdim MDSYS.SDO_DIM_ARRAY;
pointR MDSYS.SDO_GEOMETRY;
segment MDSYS.SDO_GEOMETRY;
lastupper NUMBER;
i INTEGER;


CURSOR c1 IS
   SELECT /*+ INDEX (t spatseg_idx) */
   t.moid, t.segid, t.spatseg, t.upper, t.lower
   FROM TRAJECTORIES t
   WHERE SDO_RELATE(t.spatseg,
      MDSYS.SDO_GEOMETRY(2003, NULL, NULL,
      MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,3),
      MDSYS.SDO_ORDINATE_ARRAY(xx1, yy1, xx2, yy2)),
      'mask = ANYINTERACT') = 'TRUE'
   ORDER BY t.moid ASC, t.lower ASC;

BEGIN

   pointR := MDSYS.SDO_GEOMETRY(2001, NULL,
                        MDSYS.SDO_POINT_TYPE(xcoor,ycoor,NULL),
                           NULL, NULL);
   currentmoid := -1;
   currentsegid := -1;
   SELECT DIMINFO INTO interdim FROM USER_SDO_GEOM_METADATA
   WHERE TABLE_NAME = 'TRAJECTORIES';
   candindex := 0;
   xx1 := xcoor - dist;
   xx2 := xcoor + dist;
   yy1 := ycoor - dist;
   yy2 := ycoor + dist;

   FOR c1_rec IN c1 LOOP
     IF c1_rec.segid = currentsegid AND c1_rec.moid = currentmoid THEN
       GOTO end_of_loop; -- There isn't a CONTINUE statement in PL/SQL ?
      ELSE
         currentsegid := c1_rec.segid;
```

```
END IF;
IF (c1_rec.moid != currentmoid) THEN
   -- DO SOMETHING WITH THE FINISHED OBJECT
  IF (currentmoid != -1) AND (consecutive AND within) THEN
         candidates.EXTEND;
         candindex := candindex + 1;
         candidates(candindex) := currentmoid;
   END IF;
   currentmoid := c1_rec.moid;
   consecutive := TRUE;
   within := FALSE;
ELSE
   IF NOT (c1_rec.lower = lastupper) THEN
      consecutive := FALSE;
   END IF;
END IF;
lastupper := c1_rec.upper;
x1 := c1_rec.spatseg.SDO_ORDINATES(1);
y1 := c1_rec.spatseg.SDO_ORDINATES(2);
x2 := c1_rec.spatseg.SDO_ORDINATES(3);
y2 := c1_rec.spatseg.SDO_ORDINATES(4);
t1 := c1_rec.lower;
t2 := c1_rec.upper;
segment := MDSYS.SDO_GEOMETRY(2002, NULL, NULL,
MDSYS.SDO_ELEM_INFO_ARRAY(1, 2, 1),
MDSYS.SDO_ORDINATE_ARRAY(x1, y1, x2, y2));
IF (t2 < starttime) OR (t1 > endtime) THEN
   GOTO end_of_loop;
END IF;
IF starttime > t1 THEN
   xx := x1 + (starttime - t1) / (t2 - t1) * (x2 - x1);
   yy := y1 + (starttime - t1) / (t2 - t1) * (y2 - y1);
   segment := MDSYS.SDO_GEOMETRY(2002, NULL, NULL,
   MDSYS.SDO_ELEM_INFO_ARRAY(1, 2, 1),
   MDSYS.SDO_ORDINATE_ARRAY(xx, yy,
   segment.SDO_ORDINATES(3), segment.SDO_ORDINATES(4)));
END IF;
IF endtime < t2 THEN
   xx := x1 + (endtime - t1) / (t2 - t1) * (x2 - x1);
   yy := y1 + (endtime - t1) / (t2 - t1) * (y2 - y1);
   segment := MDSYS.SDO_GEOMETRY(2002, NULL, NULL,
```

```
            MDSYS.SDO_ELEM_INFO_ARRAY(1, 2, 1),
            MDSYS.SDO_ORDINATE_ARRAY(segment.SDO_ORDINATES(1),
            segment.SDO_ORDINATES(2), xx, yy));
         END IF;
         IF SDO_GEOM.WITHIN_DISTANCE(segment, interdim, dist,
         pointR, interdim) = 'TRUE' THEN
            valid := true;
         ELSE
            valid := false;
         END IF;
         within := within OR valid;
         <<end_of_loop>>
         NULL;
      END LOOP;

      -- Some processing for the last moving object
      IF (consecutive AND within) THEN
         candidates.EXTEND;
         candindex := candindex + 1;
         candidates(candindex) := currentmoid;
      END IF;

      i := 1;
      LOOP
         EXIT WHEN i > candindex;
         dbms_output.put_line(candidates(i));
         i := i + 1;
      END LOOP;

   END;
   /
```