Ανάπτυξη Λογισμικού Πληροφορικής
Χειμερινό Εξάμηνο 2015-2016

“Σύστημα ανίχνευσης συγκρούσεων σε επερωτήσεις βάσεων δεδομένων”

Υπεύθυνος καθηγητής

Γιάννης Ιωαννίδης

Συνεργάτες Μαθήματος

Σταμάτης Χριστοφορίδης

Όμηρος Μεταξάς
Αλέξανδρος Παπαδόπουλος
Γιάννης Χρόνης

Πίνακας Περιεχομένων

Γενική Περιγραφή

Το θέμα της φετινής εργασίας βασίζεται στον περσινό προγραμματιστικό διαγωνισμό του Sigmod, όπου καλείστε να αναπτύξετε μια εφαρμογή επεξεργασίας δοσοληψιών (transactions) σε βάσεις δεδομένων.

Ως γνωστόν οι ιδιότητες **A**tomic, **C**onsistent, **I**solated και **D**urable μιας δοσοληψίας αποτελούν θεμελιώδεις λίθοι στις βάσεις δεδομένων. Προκειμένου, να εξασφαλισθεί η 3η ιδιότητα, δηλαδή κάθε δοσοληψία να είναι “απομονώσιμη”, είναι απαραίτητο οι ταυτόχρονες εγγραφές και αναγνώσεις να χειρίζονται προσεκτικά στη βάση.

Είναι σύνηθες, η επεξεργασία δοσοληψιών να πραγματοποιείται με έναν *οπτιμιστικό* τρόπο, κατά τον οποίο, αρχικά εκτελούνται όλες οι επερωτήσεις και μόνο κατά το **commit**, το σύστημα ελέγχει για ενδεχόμενες συγκρούσεις (conflicts), ανάμεσα στους αναγνώστες (readers) και στους συγγραφείς (writers) εγγραφών.

Σκοπός της συγκεκριμένης εργασίας είναι η υλοποίηση μιας εφαρμογής η οποία θα επεξεργάζεται τις ήδη πραγματοποιημένες δοσοληψίες και θα ανιχνεύει τις συγκρούσεις. Δεδομένου ότι το μέγεθος των δεδομένων προς επεξεργασία είναι μεγάλο, είναι αναγκαία η ανάπτυξη κατάλληλων δομών για τον αποδοτικό χειρισμό τους.

Τα δεδομένα που θα χρησιμοποιηθούν στην εργασία θα είναι αντίστοιχα με αυτά του περσινού διαγωνισμού. Σχετικές πληροφορίες αναφορικά με το φετινό θέμα αλλά και τα dataset, μπορείτε να βρείτε στη σελίδα του διαγωνισμού: http://db.in.tum.de/sigmod15contest/task.html .

Περιγραφή παραδοτέων μαθήματος

Η εργασία θα χωριστεί σε τρία επίπεδα. Το κάθε επίπεδο θα υλοποιηθεί σε συγκεκριμένη χρονική περίοδο και θα βασίζεται στα προηγούμενα επίπεδα. Ο διαχωρισμός των επιπέδων στοχεύει: στην ανάπτυξη μιας απλής λύσης του προβλήματος και έπειτα στη σταδιακή βελτιστοποίηση των επιμέρους τμημάτων, βελτιώνοντας με αυτό τρόπο την αποδοτικότητα της εφαρμογής.

Στη συνέχεια παρουσιάζονται οι λειτουργίες που θα αναπτυχθούν στο πρώτο επίπεδο.

1. Υλοποίηση “Parser”
2. Υλοποίηση “Ημερολογίου” (journal)
3. Υλοποίηση βασικής δομής ευρετηρίου και λειτουργιών του
4. Υλοποίηση test main για τον έλεγχο των δομών δεδομένων
5. Υλοποίηση αλγορίθμου επίλυσης του προβλήματος

1. Υλοποίηση “Parser”

Στη πρότυπη είσοδο (stdin) θα σας παρέχεται ένα σύνολο μηνυμάτων σε δυαδική μορφή που θα αποτελείται από την επικεφαλίδα (header) και το σώμα (body). Ο ορισμός της επικεφαλίδας δίνεται στον παρακάτω πίνακα.

```
typedef enum { Done, DefineSchema, Transaction, ValidationQueries, Flush, Forget } Type_t;
typedef struct MessageHead {
   uint32_t messageLen;    /// Total message length, excluding this head
   Type_t type;            /// The message type
} MessageHead_t;
```

Με βάση τον τύπο (type) και το μέγεθος (messageLen) που θα βρίσκεται στην επικεφαλίδα κάθε μηνύματος θα μπορείτε να διαβάζετε (εφόσον χρειάζεται) το σώμα του αντίστοιχου μηνύματος. Παρακάτω, παραθέτουμε ενδεικτικές προδιαγραφές, που μπορείτε να χρησιμοποιήσετε για την ανάγνωση των μηνυμάτων, για το κάθε είδος εντολής που υποστηρίζεται (*Done, DefineSchema, Transaction, ValidationQueries, Flush, Forget*).

Σχήμα
Μορφή : defineSchema [<#columns>, ...]
Σε κάθε είσοδο αυτό θα είναι το πρώτο μήνυμα που θα συναντήσετε και δηλώνει τον αριθμό των σχέσεων και τον αριθμό των πεδίων για κάθε σχέση.

```
typedef struct DefineSchema {
   /// Number of relations
   uint32_t relationCount;
   /// Column counts per relation, one count per relation.
   /// The first column is always the primary key
   uint32_t columnCounts[];
} DefineSchema_t;
```

Εγγραφή Δοσοληψίας
Μορφή : transaction <tId> [<delOps>, ...] [<insOps>, ...]

delete <relationId> [<keys>, ...]

insert <relationId> [<values>, ...]

```
typedef struct Transaction {
   uint64_t transactionId;                  /// The transaction id. Monotonic increasing

   uint32_t deleteCount,insertCount;    /// The operation counts
   char operations[];                          /// A sequence of transaction operations. Deletes first
} Transaction_t;

typedef struct TransactionOperationDelete {
   uint32_t relationId;     /// The affected relation
   uint32_t rowCount;    /// The row count
   uint64_t keys[];         /// The deleted values, rowCount primary keyss
} TransactionOperationDelete_t;

typedef struct TransactionOperationInsert {
   uint32_t relationId;     /// The affected relation
   uint32_t rowCount;     /// The row count
   uint64_t values[];  /// The inserted values, rowCount*relation[relationId].columnCount values
} TransactionOperationInsert_t;
```

Διαγραφή Δοσοληψιών
Μορφή : forget <transactionId>

```
typedef struct Forget {
   /// Transactions older than that (including) will not be tested for
   uint64_t transactionId;
} Forget_t;
```

Επαλήθευση Δοσοληψιών
Μορφή : validation <validationId> <fromTransactionId> <toTransactionId> [<queries>, ...]

```
typedef struct ValidationQueries {
   uint64_t validationId;    /// The validation id. Monotonic increasing
   uint64_t from,to;          /// The transaction range
   uint32_t queryCount;   /// The query count
   char queries[];            /// The queries
} ValidationQueries_t;

/// Support operations
typedef enum { Equal, NotEqual, Less, LessOrEqual, Greater, GreaterOrEqual } Op_t;

typedef struct Column {
  uint32_t column;    /// The column id
  Op_t op;    /// The operations
  uint64_t value;    /// The constant
} Column_t;

typedef struct Query {
   uint32_t relationId;     /// The relation
   uint32_t columnCount;     /// The number of bound columns
   Column_t columns[];     /// The bindings
} Query_t;
```

Υπολογισμός Συγκρούσεων
Μορφή : flush <validationId>

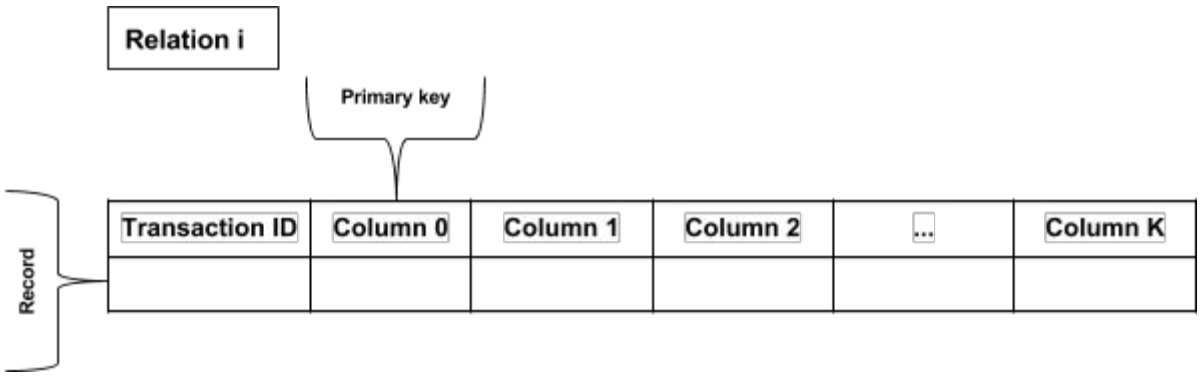
```
typedef struct Flush {
   /// All validations to this id (including) must be answered
   uint64_t validationId;
} Flush_t;
```

Έξοδος
Μορφή : done

Για την υλοποίηση του parser και τη διαδικασία ανάγνωσης της ροής των δεδομένων εισόδου, μπορείτε να ανατρέξετε στην ενδεικτική υλοποίηση στη σελίδα του διαγωνισμού.

2. Υλοποίηση “Ημερολογίου”

Τα δεδομένα που αναφέρονται σε δοσοληψίες (εισαγωγές, εγγραφές πλειάδων) και διαβάστηκαν στο πρώτο βήμα, θα αποθηκευτούν σε μια δομή πίνακα με κατάλληλη μορφή. Η λειτουργία του πίνακα αυτού, είναι ανάλογη με μια δομή ημερολογίου (journal) που συναντούμε στις βάσεις δεδομένων και ουσιαστικά καταγράφουν την κάθε δοσοληψία που πραγματοποιείται. Η κάθε σχέση θα έχει έναν πίνακα ως ημερολόγιο με την παρακάτω μορφή:



Σχήμα 1. Περιγραφή δομής ημερολογίου

Εκτός από τις δοσοληψίες εισαγωγής εγγραφών, υπάρχουν και αυτές διαγραφής. Στην περίπτωση αυτή θα αντιμετωπίσουμε τις διαγραφές ως εισαγωγές και θα εισαγάγουμε στο τέλος του ημερολογίου τη δοσοληψία διαγραφής. Ακόμη, πρωτού την εισαγάγουμε θα αναζητούμε την τελευταία εγγραφή με αυτό το κλειδί (εφόσον υπάρχει) και θα αντιγράψουμε τις υπάρχουσες τιμές των υπόλοιπων πεδίων στη νέα εγγραφή.

Καθώς καταφθάνουν ριπές από δοσοληψίες κατά την εκτέλεση του προγράμματος, τα ημερολόγια των σχέσεων θα μεγαλώνουν διαρκώς, αφού οι δοσοληψίες (με εξαίρεση αυτές που διαγράφονται με την εντολή flush) παραμένουν ενεργές καθόλη την εκτέλεση. Στην περίπτωση όπου εξαντλείται ο διαθέσιμος χώρος του πίνακα, τότε το μέγεθος του ημερολογίου πρέπει να μεγαλώνει δυναμικά.

Λειτουργίες Ημερολογίου

Στη συνέχεια παρουσιάζονται τα πρότυπα των συναρτήσεων για τις λειτουργίες που θα υλοποιήσετε. Οι τύποι που δηλώνονται όπως *Journal, JournalRecord,* κλπ είναι είτε κλάσεις είτε δομές (structs) που πρέπει να υπάρχουν ως οντότητες στην εφαρμογή σας.

Journal createJournal();*

OK_SUCCESS insertJournalRecord(Journal, JournalRecord*, …);*

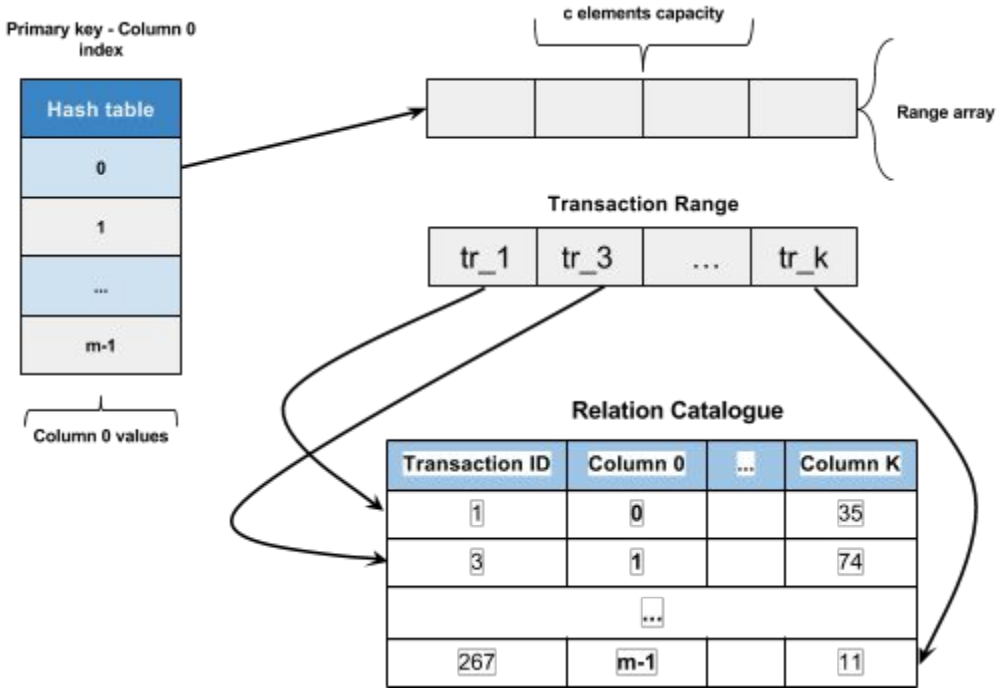
List<Record> getJournalRecords(Journal, JournalRecord*, … , int range_start, int range_end);*

OK_SUCCESS destroyJournal(Journal);*

OK_SUCCESS increaseJournal(Journal, …);*

3. Υλοποίηση βασικής δομής ευρετηρίου και λειτουργιών του

Για την αποδοτική υλοποίηση της εφαρμογής είναι απαραίτητη η χρήση ευρετηρίων, για τη γρηγορότερη αναζήτηση εγγραφών στο ημερολόγιο. Η δομή που θα υλοποιήσετε είναι δυναμικός κατακερματισμός και συγκεκριμένα επεκτατός (https://en.wikipedia.org/wiki/Extendible_hashing) με κλειδί ένα μη προσημασμένο 64 bit αριθμό και δεδομένα έναν πίνακα ο οποίος θα εμπεριέχει ως δεδομένα προσαρμοσμένο περιεχόμενο, ανάλογα με τη χρήση του ευρετηρίου. Ο κάθε πίνακας θα έχει αρχικό μέγεθος **c**, ενώ θα μπορεί και να επεκτείνεται άμα χρειαστεί.



Σχήμα 2. Περιγραφή δομής ευρετηρίου κατακερματισμού πάνω στις εγγραφές του ημερολογίου

Το ευρετήριο αυτό θα δημιουργείται για κάθε σχέση που έχει δηλωθεί στο σχήμα και θα κρατάει ως κλειδί τη τιμή της 1ης στήλης που είναι το πρωτεύον κλειδί σε κάθε σχέση. Ως δεδομένα κάθε κλειδιού, θα κρατάτε έναν πίνακα ο οποίος για τις δοσοληψίες που αυτό συμμετέχει θα έχει συνδέσεις προς την αντίστοιχη εγγραφή στον κατάλογο της εν λόγω σχέσης.

Λειτουργίες δομής κατακερματισμού

Παρακάτω παρουσιάζονται τα πρότυπα των συναρτήσεων για τις λειτουργίες της δομής κατακερματισμού που θα υλοποιήσετε. Οι τύποι που δηλώνονται όπως *Hash, RangeArray,* κλπ είναι είτε κλάσεις είτε δομές (structs) που πρέπει να υπάρχουν ως οντότητες στην εφαρμογή σας.

Hash createHash();*

OK_SUCCESS insertHashRecord(Hash, Key, RangeArray*, …);*

OK_SUCCESS deleteHashRecord(Hash, Key, …);*

//marks transaction dirty/inactive
OK_SUCCESS deleteJournalRecord(Hash, Key, int transaction_id, …);*

RangeArray getHashRecord(Hash*, Key, …);*

List<Record> getHashRecords(Hash, Key, … , int range_start, int range_end);*

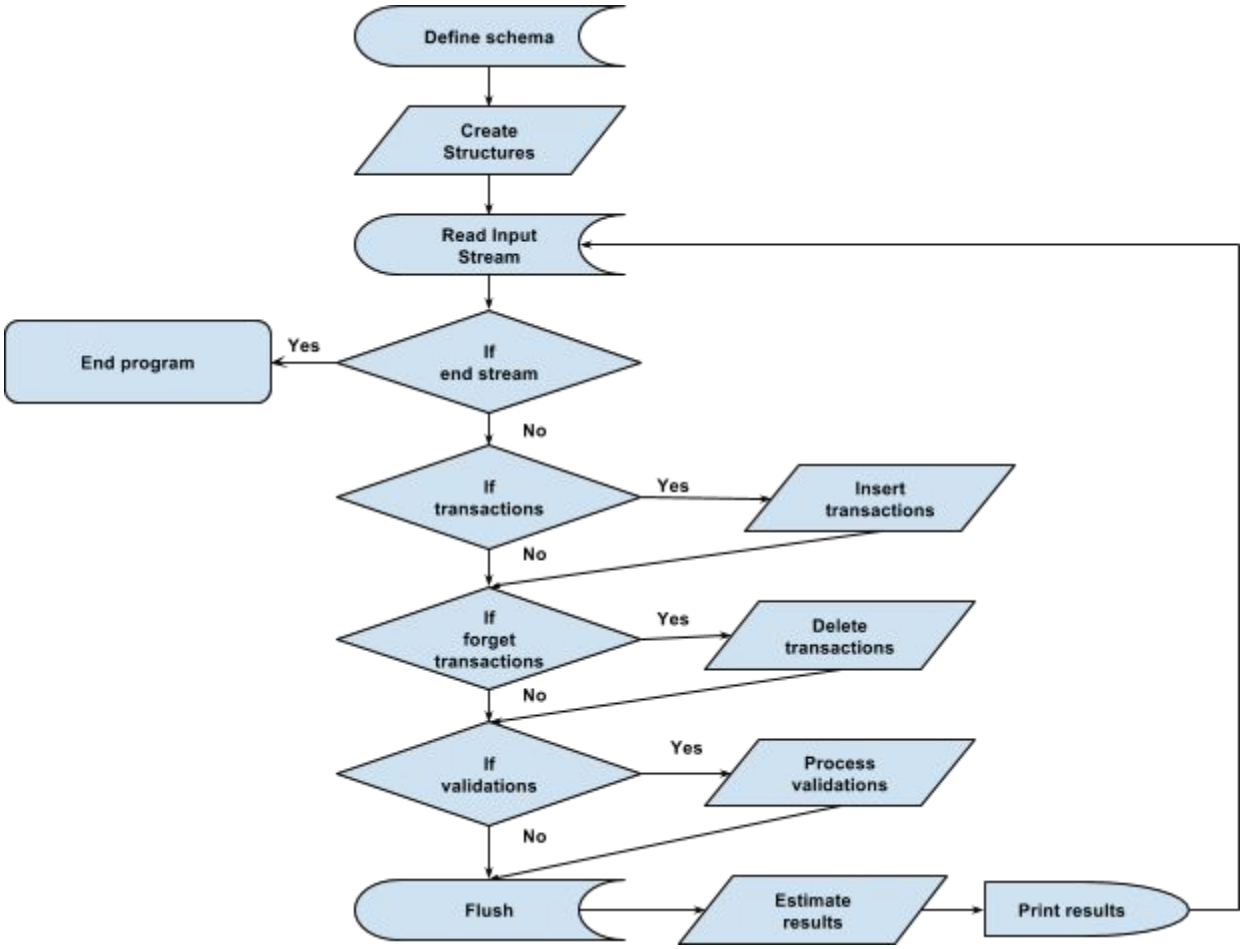
OK_SUCCESS destroyHash(Hash);*

4. Υλοποίηση test main για τον έλεγχο των δομών δεδομένων

Θα πρέπει να φτιάξετε δοκιμαστικό πρόγραμμα, το οποίο θα χρησιμοποιεί τις βασικές λειτουργίες που αναπτύξατε τόσο για τη δομή του ημερολογίου όσο και για αυτή του κατακερματισμού. Ουσιαστικά, θα πρέπει για συγκεκριμένη είσοδο το πρόγραμμα να παράγει την αναμενόμενη έξοδο. Καλό είναι το πρόγραμμα σας να προκαλεί δυναμικές μεταβολές στις δομές σας (π.χ. διπλασιασμός hash, καταλόγου κλπ) προκειμένου να εξασφαλίσετε ότι λειτουργούν σωστά. Ένας προτεινόμενος τρόπος για να ελέγχετε τις δομές σας είναι με τη χρήση unit tests (https://en.wikipedia.org/wiki/Unit_testing).

5. Υλοποίηση αλγορίθμου επίλυσης του προβλήματος

Σε αυτό το σημείο θα χρησιμοποιήσετε τα modules που αναπτύξατε και θα τα συνδυάσετε ώστε να λειτουργήσει συνολικά η εφαρμογή σας και να παράγει αποτελέσματα. Η ροή εκτέλεσης της εφαρμογής παρουσιάζεται στο παρακάτω σχήμα.



Σχήμα 3. Διάγραμμα ροής - εκτέλεσης της εφαρμογής

Στην αρχή της εισόδου με την εντολή “define schema”, δηλώνεται το σχήμα της βάσης, δηλαδή πόσες σχέσεις υπάρχουν συνολικά, καθώς και το πλήθος των στηλών κάθε σχέσης. Έπειτα, ακολουθούν transactions ακολουθούμενα από validations. Καθώς, έρχονται τα transactions πρέπει να ενημερώνονται οι δομές του ημερολογίου και του κατακερματισμού για τις νέες εγγραφές. Αφότου, ολοκληρωθεί η επεξεργασία των transactions, θα καθοριστεί αν το κάθε validation είναι έγκυρο. Για να είναι έγκυρο το κάθε validation (δηλαδή να μην υπάρχει σύγκρουση) πρέπει να μην

υπάρχει κανένας όρος του που να δημιουργεί σύγκρουση. Επομένως, για να απαντηθεί η τιμή ενός validation, είναι απαραίτητος ο έλεγχος των επιμέρους συνθηκών.

Το ευρετήριο κατακερματισμού θα χρησιμοποιείται για να απαντάει σε επερωτήσεις ισότητας στο πρωτεύον κλειδί μιας σχέσης. Για όλους τους υπόλοιπους όρους των validations θα χρησιμοποιείται το ημερολόγιο για την απάντηση τους στο διάστημα των transactions που αναφέρονται.

Στην εντολή “forget transaction”, δεν είναι ανάγκη να διαγραφεί πραγματικά η δοσοληψία από τις δομές, αλλά αρκεί να “μαρκαριστεί” ως διαγραμμένο το συγκεκριμένο εύρος δοσοληψιών στο ευρετήριο κατακερματισμού, ώστε να αγνοείται κατά τις αναζητήσεις.

Τέλος, στην εντολή “flush” θα τυπώνονται τα αποτελέσματα (true / false) για το κάθε validation της τρέχουσας ριπής.

Παράδειγμα Εκτέλεσης

Στην αριστερή στήλη του πιο κάτω πίνακα φαίνεται ένα απλό παράδειγμα εισόδου και στη δεξιά στήλη οι διάφορες φάσεις επεξεργασίας που θα βρίσκεται το πρόγραμμα που θα αναπτύξετε. Στη συνέχεια περιγράφουμε αναλυτικότερα την κάθε φάση επεξεργασίας και την κατάσταση των δομών δεδομένων σε κάθε μία από αυτές.

Πρότυπη Είσοδος	**Φάση Επεξεργασίας**	**Πρότυπη Έξοδος**
defineschema [3 4] transaction 0 [] [0 [1 1 2 2 1 2 3 4 5 7 7 7] 1 [1 0 0 0 3 0 0 1 4 1 1 1]]	Φάση Αρχικοποίησης	
forget 0 transaction 1 [] [0 [6 5 4]] transaction 2 [1 [4]] transaction 3 [0 [3]] [0 [3 5 6]]	Φάση Δοσοληψιών	
validation 0 1 2 [0 c0=4] [1 c1>8] validation 1 1 2 [1 c2=1] validation 2 1 3 [0 c0=3 c1=2] [0 c2=4]	Φάση Επαλήθευσης	
flush 2	Φάση Υπολογισμού	0 1 1
done	Φάση Εξόδου	

Φάση Αρχικοποίησης

Το πρώτο μήνυμα που θα λαμβάνεται στη πρότυπη είσοδο (stdin) θα ορίζει τις σχέσεις που θα αναφέρονται στη συνέχεια τα μηνύματα δοσοληψιών και επαλήθευσης. Αμέσως μετά θα ακολουθεί ένα μήνυμα δοσοληψίας που θα αρχικοποιεί τις αντίστοιχες σχέσεις όπως φαίνεται στο παρακάτω παράδειγμα.

```
defineschema [3 4]
transaction 0 [] [
  0 [1 1 2  2 1 2  3 4 5  7 7 7]
  1 [1 0 0 0  3 0 0 1  4 1 1 1]
]
```

Σε αυτή την περίπτωση έχουμε 2 σχέσεις (έστω Σ0 και Σ1) όπου έχουμε 3 και 4 πεδία (στήλες) αντίστοιχα. Έπειτα ακολουθεί η δοσοληψία με κλειδί 0 όπου αρχικοποιεί τις δύο σχέσεις με κάποια αρχικά δεδομένα. Μετά την επεξεργασία των δύο μηνυμάτων στον πιο κάτω πίνακα φαίνεται η κατάσταση των ημερολογίων και των ευρετηρίων.

Κατάσταση Ημερολογίων κ Ευρετηρίων.

Ημερολόγιο Σχέσης Σ0.

tid	c0	c1	c2
0	1	1	2
0	2	1	2
0	3	4	5
0	7	7	7

Ημερολόγιο Σχέσης Σ1.

tid	c0	c1	c2	c3
0	1	0	0	0
0	3	0	0	1
0	4	1	1	1

Ευρετήριο Σχέσης Σ0.	
key	**value**
1	[0]
2	[0]
3	[0]
7	[0]

Ευρετήριο Σχέσης Σ1.	
key	**value**
1	[0]
3	[0]
4	[0]

Φάση Δοσοληψιών

Στη συνέχεια θα έχουμε ένα σύνολο μηνυμάτων από διαγραφή ή εγγραφή δοσοληψιών όπως φαίνεται για εξής παράδειγμα.

```
forget 0
transaction 1 [] [0 [6 5 4]]
transaction 2 [1 [4]]
transaction 3 [0 [3]] [0 [3 5 6]]
```

Σε αυτή την περίπτωση έχουμε μια διαγραφή δοσοληψιών μέχρι και την δοσοληψία με κλειδί 0 και 3 δοσοληψίες με κάποιες λειτουργίες εισαγωγής και διαγραφής εγγραφών (πλειάδων). Η λειτουργία διαγραφής δοσοληψιών δεν επηρεάζει την κατάσταση ημερολογίων, αφορά μόνο την ενημέρωση τυχόν ευρετηρίων. Σε αυτή τη φάση επίσης μπορεί να έχουμε δοσοληψίες εισαγωγής ή διαγραφής εγγραφών (πλειάδων) για μια σχέση. Στην δοσοληψία με κλειδί 1 έχουμε μόνο μία εισαγωγή στην σχέση 0 της εγγραφής με κλειδί 6. Η δοσοληψία με κλειδί 2 έχει μόνο μια διαγραφή στο σχέση 1 της εγγραφής με κλειδί 4. Η δοσοληψία με κλειδί 3 έχει μία διαγραφή της εγγραφής με κλειδί 3 στη σχέση 0 και την εισαγωγή της εγγραφής (γραμμής) με κλειδί 3 στη σχέση 0. Μετά την επεξεργασία των μηνυμάτων έχουμε την ακόλουθη κατάσταση των ημερολογίων. Στη λειτουργία της διαγραφής για μία δοσοληψία αρκεί να προσθέσουμε μια καινούργια εγγραφή στο αντίστοιχο ημερολόγιο με τις ίδιες τιμές στα υπόλοιπα πεδία της εγγραφής για να δηλώσουμε την πρόσβαση μας στην αντίστοιχη εγγραφή.

Κατάσταση Ημερολογίων κ Ευρετηρίων.

Κατάσταση Ημερολογίου Σχέσης Σ0.

tid	c0	c1	c2
0	1	1	2
0	2	1	2
0	3	4	5
0	7	7	7
1	6	5	4
3	3	4	5
3	3	5	6

Κατάσταση Ημερολογίου Σχέσης Σ1.

tid	c0	c1	c2	c3
0	1	0	0	0
0	3	0	0	1
0	4	1	1	1
2	4	1	1	1

Ευρετήριο Σχέσης Σ0.

key	value
1	[0]
2	[0]
3	[0, 3]
6	[1]
7	[0]

Ευρετήριο Σχέσης Σ1.

key	value
1	[0]
3	[0]
4	[0,2]

Φάση Επαλήθευσης

Μετά τα μηνύματα εγγραφής ή διαγραφής δοσοληψιών ακολουθούν τα μηνύματα επαλήθευσης δοσοληψιών όπως φαίνονται στο παράδειγμα του παρακάτω πίνακα.

```
validation 0 1 2 [0 c0=4] [1 c1>8]
validation 1 1 2 [1 c2=1]
validation 2 1 3 [0 c0=3 c1=2] [0 c2=4]
```

Σε αυτή την περίπτωση έχουμε τα πρώτα δύο μηνύματα επαλήθευσης (με κλειδί 0,1) τα οποία αφορούν τις δοσοληψίες [1,2] και το τρίτο μήνυμα επαλήθευσης που αφορά τις δοσοληψίες [1,3]. Σε αυτή τη φάση η κατάσταση ημερολογίων και των ευρετηρίων δεν αλλάζει. Το μόνο που χρειάζεται να κρατήσετε είναι μία λίστα με τα τρέχοντα μηνύματα επαλήθευσης.

Φάση Υπολογισμού

Σε αυτή τη φάση πρέπει να γίνουν οι απαραίτητοι υπολογισμοί έτσι ώστε να αποφανθούμε για τα αντίστοιχα μηνύματα επαλήθευσης. Όπως φαίνεται στο πιο κάτω μήνυμα πρέπει να υπολογίσουμε για όλα τα μηνύματα επαλήθευσης που έχουν έρθει μέχρι και το κλειδί 2 για το αν προκύπτουν συγκρούσεις ή όχι.

```
flush 2
```

Για κάθε στοιχείο της λίστας με τα τρέχοντα μηνύματα επαλήθευσης θα υπολογίσουμε αν υπάρχουν συγκρούσεις και θα ενημερώσουμε την πρότυπη έξοδο (stdout) με 1 αν έχουμε συγκρούσεις, αλλιώς με 0.

Για το μήνυμα επαλήθευσης με κλειδί 0 έχουμε τα κατηγορήματα c0=4 για τη σχέση Σ0 και c1>8 για τη σχέση Σ1. Για κατηγορήματα ισότητας (c0=<x>) πάνω στο πρωτεύον κλειδί όπως έχουμε σε αυτή την περίπτωση αρκεί να χρησιμοποιήσετε το ευρετήριο. Για c0=4 στη σχέση Σ0 δεν υπάρχουν δοσοληψίες στο ευρετήριο της Σ0, έτσι δεν υπάρχει σύγκρουση. Για τα υπόλοιπα κατηγορήματα θα χρησιμοποιείται το ημερολόγιο δοσοληψιών. Οπότε για το κατηγόρημα c1 > 8 της σχέσης Σ1 θα κοιτάξουμε το ημερολόγιο της σχέσης Σ1 για το διάστημα [1,2] αν υπάρχουν πλειάδες που ικανοποιούν το συγκεκριμένο κατηγόρημα. Τελικά, το μήνυμα με κλειδί 0 δεν προκαλεί συγκρούσεις για τα κατηγορήματα [0 c0=4] [1 c1>8] στο διάστημα [1,2].

Για το μήνυμα επαλήθευσης με κλειδί 1 έχουμε το κατηγόρημα c2=1 για τη σχέση Σ1. Παρόλο που είναι κατηγόρημα ισότητας λόγω του ότι δεν αναφέρεται στο πρωτεύον κλειδί θα χρησιμοποιήσετε το ημερολόγιο της σχέσης Σ1 για υπολογίσετε αν έχει συγκρούσεις. Αρκεί να ελέγξετε τις δοσοληψίες στο διάστημα [1,2] αν έχουν πρόσβαση στο πεδίο c2 με τιμή 1. Εδώ έχουμε σύγκρουση μιας και υπάρχει η δοσοληψία με κλειδί 2 που διαγράφει τη συγκεκριμένη πλειάδα.

Για το μήνυμα επαλήθευσης με κλειδί 2 έχουμε τα κατηγορήματα c0=3 και c1=2 για τη σχέση Σ0 ή c2=4 για τη σχέση Σ0. Για το πρώτο κατηγόρημα c0=3 θα χρησιμοποιήσετε το ευρετήριο της σχέσης Σ0 για να βρείτε αν υπάρχουν δοσοληψίες για το συγκεκριμένο διάστημα που προκαλούν σύγκρουση. Για το c0=3 υπάρχουν οι δοσοληψίες με κλειδί 0, 3 αλλά επειδή μας ενδιαφέρει το διάστημα [1,3] θα πρέπει να ελεγχθεί μόνο η δοσοληψία 3 όπου δεν δημιουργεί σύγκρουση με τα κατηγορήματα c0=3 και c1=2. Για το κατηγόρημα c1=2 θα χρησιμοποιήσουμε το ημερολόγιο της σχέσης Σ0 για να δούμε αν υπάρχουν δοσοληψίες στο διάστημα [1,3] που προκαλούν σύγκρουση στο πεδίο c1 με τιμή 2. Όπως φαίνεται από το αντίστοιχο ημερολόγιο δεν

υπάρχει σύγκρουση ούτε γι αυτό το κατηγόρημα. Απομένει ο έλεγχος για το κατηγόρημα c2=4 για τη σχέση 0 στο διάστημα [1,3] όπου φαίνεται από το ημερολόγιο της σχέσης Σ0 ότι προκαλεί σύγκρουση με τη δοσοληψία με κλειδί 1.

Τέλος με βάση τους πιο πάνω υπολογισμούς συγκρούσεων η πρότυπη έξοδος θα πρέπει να έχει την ακόλουθη μορφή.

```
0
1
1
```

Φάση Εξόδου

```
done
```

Σε αυτή τη φάση γίνεται αποδέσμευση των πόρων έχουν δεσμευτεί (μνήμη ημερολογίων, ευρετηρίων) και έξοδος του προγράμματος.