Θέματα Υλοποίησης Σχεσιακών ΣΔΒΔ

Γιάννης Θεοδωρίδης

InfoLab, Τμήμα Πληροφορικής, Πανεπιστήμιο Πειραιά

http://infolab.cs.unipi.gr

version: Nov.2009

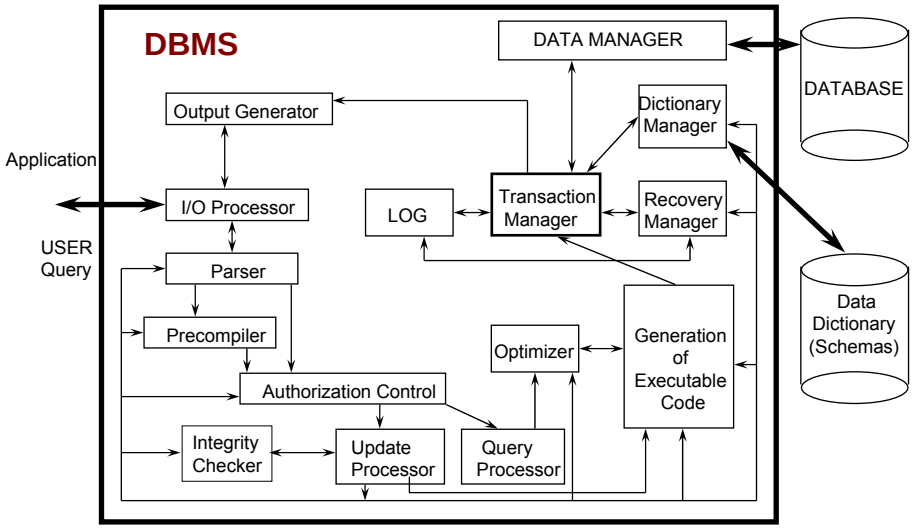
Περιεχόμενα

- Η τυπική αρχιτεκτονική ενός Σχεσιακού ΣΔΒΔ
- Οργάνωση δεδομένων σε φυσικό επίπεδο
  - Αρχεία, Ευρετήρια
- Επεξεργασία ερωτήσεων
  - Ο βελτιστοποιητής ερωτήσεων
- Διαχείριση δοσοληψιών
  - Συγχρονισμός, ανάκαμψη

Περιεχόμενα

- **Η τυπική αρχιτεκτονική ενός Σχεσιακού ΣΔΒΔ**
- Οργάνωση δεδομένων σε φυσικό επίπεδο
  - Αρχεία, Ευρετήρια
- Επεξεργασία ερωτήσεων
  - Ο βελτιστοποιητής ερωτήσεων
- Διαχείριση δοσοληψιών
  - Συγχρονισμός, ανάκαμψη

Αρχιτεκτονική ενός ΣΔΒΔ

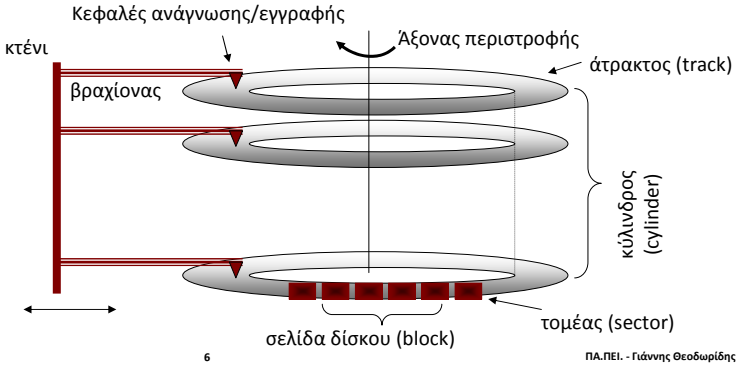


Περιεχόμενα

- Η τυπική αρχιτεκτονική ενός Σχεσιακού ΣΔΒΔ
- **Οργάνωση δεδομένων σε φυσικό επίπεδο**
  - **Αρχεία, Ευρετήρια**
- Επεξεργασία ερωτήσεων
  - Ο βελτιστοποιητής ερωτήσεων
- Διαχείριση δοσοληψιών
  - Συγχρονισμός, ανάκαμψη

Η ΒΔ αποθηκεύεται στο σκληρό δίσκο

- Τα δεδομένα οργανώνονται σε σελίδες δίσκου (blocks)
- Χρόνος προσπέλασης ενός block ~10 msec
  - πολύ μεγαλύτερος από την επεξεργασία των δεδομένων στην κύρια μνήμη



Παραδοχές

- Η μονάδα μεταφοράς μεταξύ δίσκου και μνήμης είναι ένα block δίσκου (με μέγεθος b bytes)
- Κάθε αρχείο έχει εγγραφές ενός τύπου μόνο
- Το μέγεθος των εγγραφών είναι σταθερό (με μέγεθος r bytes)
- Οι εγγραφές δεν επιτρέπεται να διασχίζουν τα όρια ενός block

Παράγοντας ομαδοποίησης (blocking factor):

$$bfr = \lfloor (b / r) \rfloor$$

Αριθμός blocks B για την αποθήκευση ενός αρχείου R εγγραφών:

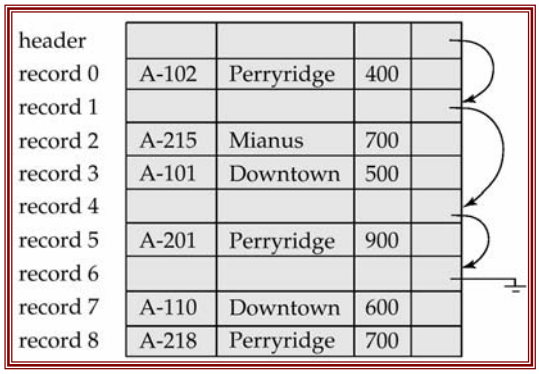
$$B = \lceil (R / bfr) \rceil$$

Οργάνωση των εγγραφών σε αρχεία

- **Αρχεία σωρού** – μια εγγραφή μπορεί να τοποθετηθεί οπουδήποτε υπάρχει χώρος στο αρχείο.
- **Διατεταγμένα αρχεία** – αποθήκευση των εγγραφών βάσει της τιμής του κλειδιού διάταξης κάθε εγγραφής.
- **Αρχεία κατακερματισμού** – μια συνάρτηση κατακερματισμού υπολογίζεται σε κάποιο χαρακτηριστικό κάθε εγγραφής. Το αποτέλεσμα καθορίζει σε ποιο block του αρχείου πρέπει να αποθηκευθεί η εγγραφή.
  - Στατικός vs. Επεκτάσιμος κατακερματισμός

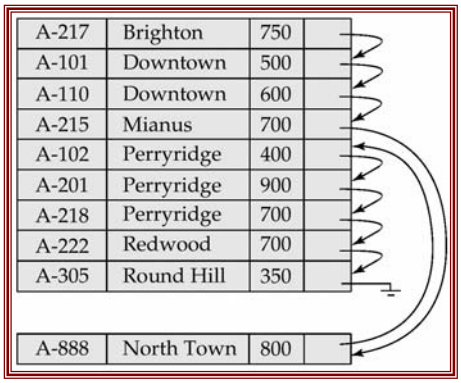
Αρχεία σωρού

- Οι εγγραφές τοποθετούνται στο αρχείο χωρίς κάποια συγκεκριμένη λογική σειρά (**heap file** ή **pile file**)
- Κόστος εισαγωγής:
  - 2 blocks (= 1 read + 1 write)
- (μέσο) κόστος αναζήτησης εγγραφών (ερώτηση ταυτότητας)
  - B/2 blocks (κλειδί)
  - B blocks (όχι κλειδί)



Διατεταγμένα αρχεία

- Οι εγγραφές στο αρχείο είναι διατεταγμένες βάσει ενός κλειδιού διάταξης (**ordered file**)
  - Το ΚΔ δεν ταυτίζεται κατ' ανάγκη με το πρωτεύον κλειδί
  - Υπαρξη (μη διατεταγμένου) αρχείου υπερχείλισης
    - παραδοχή: B' << B
- Κόστος αναζήτησης εγγραφής (ερώτηση ταυτότητας)
  - Δυαδική ($\log_2$B) + σειριακή αναζήτηση (B')
- Κόστος εισαγωγής εγγραφής
  - Κόστος αναζήτησης ($\log_2$B + B') + κόστος εισαγωγής (2)



Αρχεία Κατακερματισμού

- Οι εγγραφές στο αρχείο είναι τακτοποιημένες βάσει της τιμής που επιστρέφει μια συνάρτηση κατακερματισμού (**hash file**)
- Οργάνωση εγγραφών σε κάδους (buckets)
  - Παραδοχή: 1 κάδος = 1 block).
- **Στατικός** vs. **Επεκτάσιμος** κατακερματισμός

bucket 0		

bucket 1		

bucket 2		

bucket 3		
A-217	Brighton	750
A-305	Round Hill	350

bucket 4		
A-222	Redwood	700

bucket 5		
A-102	Perryridge	400
A-201	Perryridge	900
A-218	Perryridge	700

bucket 6		

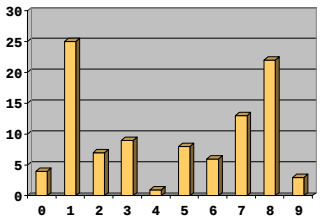
bucket 7		
A-215	Mianus	700

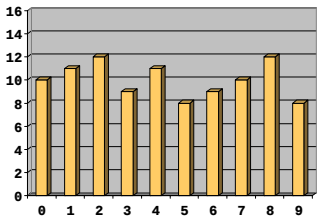
bucket 8		
A-101	Downtown	500
A-110	Downtown	600

bucket 9		

Συναρτήσεις κατακερματισμού

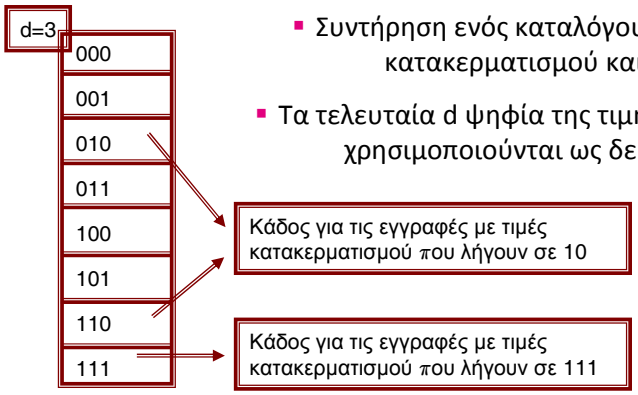
- Μια ιδανική συνάρτηση κατακερματισμού είναι:
  - **ομοιόμορφη**, σε κάθε κάδο ανατίθεται ο ίδιος αριθμός από τιμές του κλειδιού διάταξης από το σύνολο όλων των πιθανών τιμών.
  - **τυχαία**, κάθε κάδος έχει τον ίδιο αριθμό από εγγραφές ανεξάρτητα από την πραγματική κατανομή των τιμών του κλειδιού διάταξης στο αρχείο.





Επεκτάσιμος κατακερματισμός

- Δυαδική αναπαράσταση του αποτελέσματος της συνάρτησης κατακερματισμού και κατανομή εγγραφών με βάση την τιμή των λιγότερο σημαντικών ψηφίων (least significant bits - LSB)



- Συντήρηση ενός καταλόγου (directory) με τιμές κατακερματισμού και διευθύνσεις κάδων
- Τα τελευταία d ψηφία της τιμής κατακερματισμού χρησιμοποιούνται ως δείκτης στον κατάλογο

Ευρετήρια - Βασικά θέματα

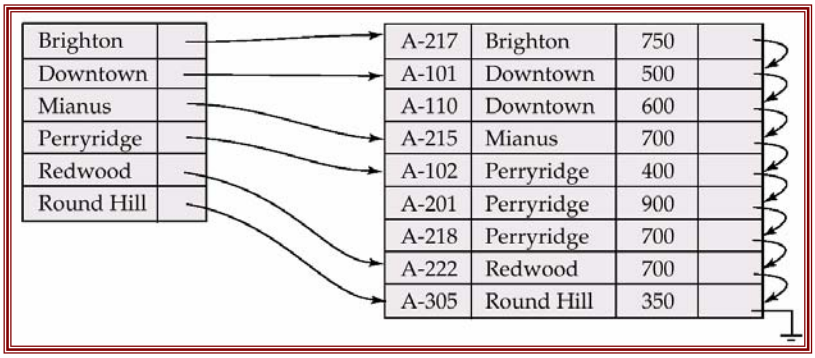
- Τα ευρετήρια λειτουργούν ως **μηχανισμοί δεικτοδότησης** με στόχο την επιτάχυνση της πρόσβασης σε επιθυμητά δεδομένα πέρα από την όποια οργάνωση έχει το αρχείο
- Ένα **ευρετήριο** (**index**) αποτελείται από εγγραφές της μορφής:

τιμή κλειδιού διάταξης	**δείκτης προς το κύριο αρχείο**

- Τα ευρετήρια συνήθως είναι αρκετά μικρότερα από το αρχείο που δεικτοδοτούν.
- **Διατεταγμένα** ευρετήρια vs. ευρετήρια **κατακερματισμού**

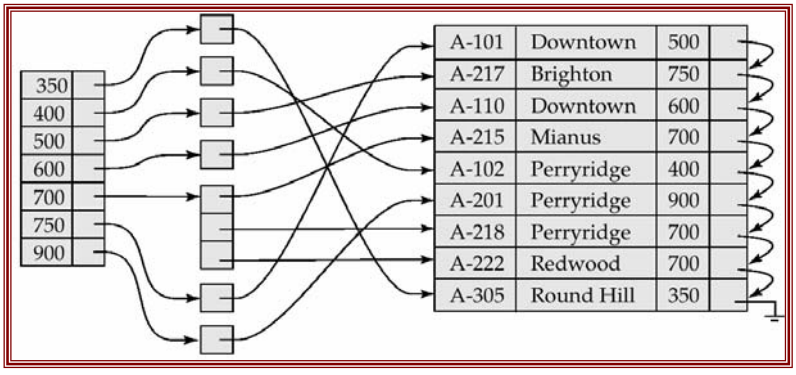
Διατεταγμένα ευρετήρια

- … πάνω στο κλειδί διάταξης του (διατεταγμένου) κύριου αρχείου



Διατεταγμένα ευρετηρία

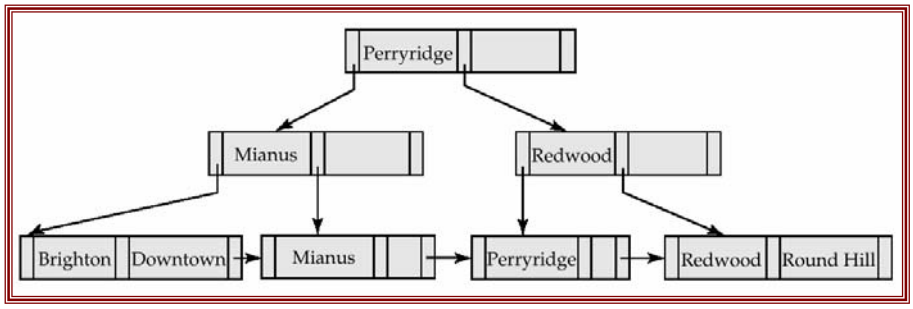
- … πάνω σε πεδίο που δεν είναι κλειδί διάταξης
- Ονομάζονται **δευτερεύοντα ευρετήρια** (secondary indexes)



Βρείτε το λάθος !!

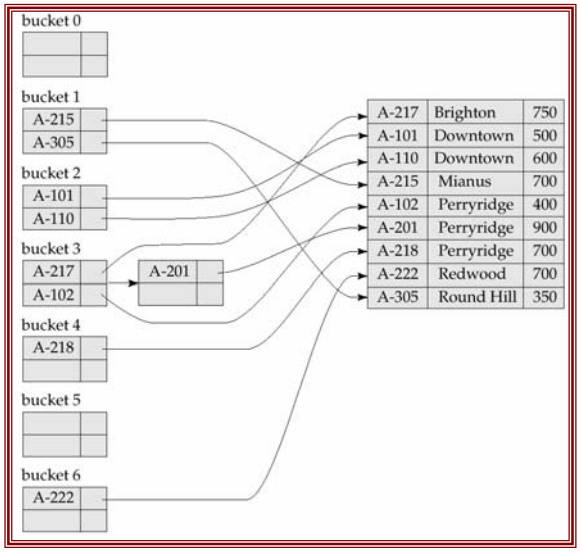
B^+-δένδρα

- Ιεραρχική δομή δένδρου που στοχεύει στην ακόμη ταχύτερη αναζήτηση
  - Σε κάθε κόμβο του δένδρου αποθηκεύονται n τιμές του κλειδιού διάταξης (1 κόμβος = 1 block)
  - κόστος αναζήτησης στο ευρετήριο $\log_n B' << \log_2 B'$ (αν n >> 2)



Ευρετήρια κατακερματισμού

- Ένα **ευρετήριο κατακερματισμού** (**hash index**) οργανώνει τα κλειδιά διάταξης και τους αντίστοιχους δείκτες στις εγγραφές σε μια δομή ενός αρχείου κατακερματισμού.
- Παράδειγμα:
  - 2 entries / bucket
  - H(key) = key % 5

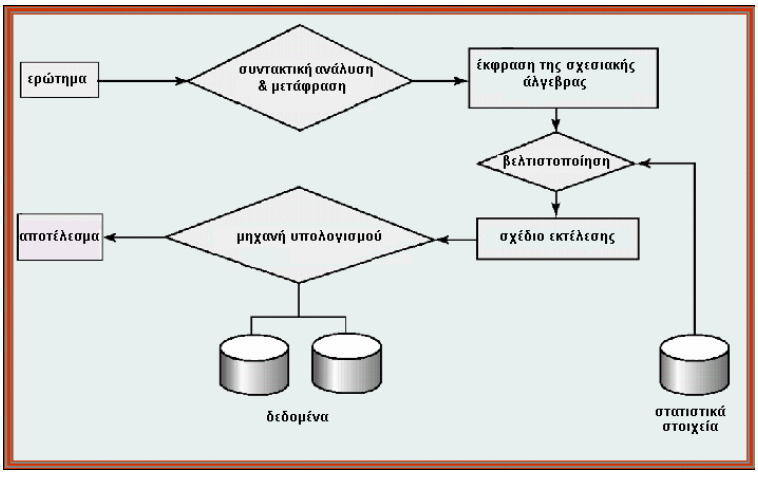


Περιεχόμενα

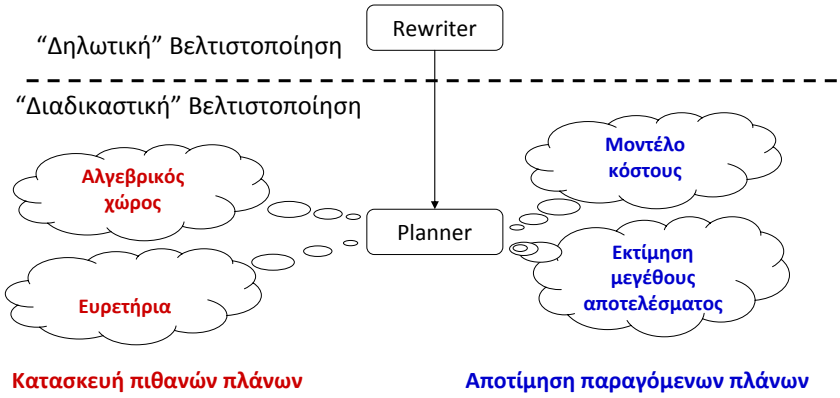
- Η τυπική αρχιτεκτονική ενός Σχεσιακού ΣΔΒΔ
- Οργάνωση δεδομένων σε φυσικό επίπεδο
  - Αρχεία, Ευρετήρια
- **Επεξεργασία ερωτήσεων**
  - **Ο βελτιστοποιητής ερωτήσεων**
- Διαχείριση δοσοληψιών
  - Συγχρονισμός, ανάκαμψη

Επεξεργασία ερωτημάτων στο ΣΔΒΔ

- Από το SQL ερώτημα στο αποτέλεσμα που επιστρέφεται από τη ΒΔ ...



Βελτιστοποίηση ερωτημάτων



Αλγεβρικός χώρος

- Μετασχηματισμός εκφράσεων ΣΑ
  - Δύο εκφράσεις ΣΑ καλούνται **ισοδύναμες** αν, σε οποιοδήποτε έγκυρο στιγμιότυπο της ΒΔ, παράγουν το ίδιο αποτέλεσμα.
  - Ένας **κανόνας ισοδυναμίας** καθορίζει αν δύο εκφράσεις ΣΑ είναι ισοδύναμες
- Ενδεικτικοί κανόνες ισοδυναμίας:
  - επιλογές: $\sigma_{\theta_1 \wedge \theta_2}(R) = \sigma_{\theta_1}(\sigma_{\theta_2}(R))$ $\sigma_{\theta_1}(\sigma_{\theta_2}(R)) = \sigma_{\theta_2}(\sigma_{\theta_1}(R))$
  - συνδέσεις: $\sigma_{\theta}(R \times S) = R \bowtie_{\theta} S$ $\sigma_{\theta_1}(R \bowtie_{\theta_2} S) = R \bowtie_{\theta_1 \wedge \theta_2} S$

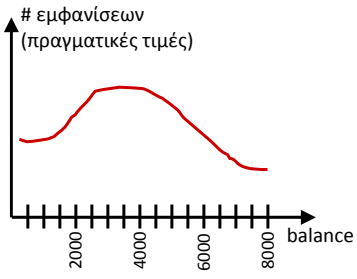
    $R \bowtie_{\theta} S = S \bowtie_{\theta} R$ $\sigma_{\theta_1}(R \bowtie_{\theta} S) = (\sigma_{\theta_1}(R)) \bowtie_{\theta} S$

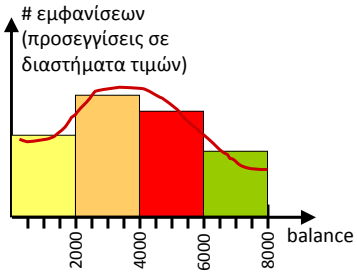
Μοντέλο κόστους

- Βασική παραδοχή: **το κόστος ενός ερωτήματος ισούται με τις προσπελάσεις στο δίσκο**.
  - Δεν λαμβάνουμε υπόψη το κόστος επεξεργασίας στην κύρια μνήμη
  - Δεν λαμβάνουμε υπόψη το κόστος εγγραφής των δεδομένων πίσω στο δίσκο
- Το κόστος επηρεάζεται σημαντικά από το μέγεθος του **buffer** της κύριας μνήμης
  - Η ποσότητα της διαθέσιμης μνήμης στον buffer είναι δύσκολο να υπολογιστεί εκ των προτέρων
- Βασικά εργαλεία υπολογισμού κόστους:
  - **Εναλλακτικοί τρόποι εκτέλεσης μιας πράξης ΣΑ**
  - **Εκτίμηση μεγέθους αποτελέσματος της πράξης**

Εκτίμηση μεγέθους αποτελέσματος

- Πώς θα εκτιμήσουμε το μέγεθος των ενδιάμεσων αποτελεσμάτων (όταν δεν έχουν ακόμη εκτελεστεί) ;;
- Η πιο καλή τεχνική που έχουμε είναι τα **ιστογράμματα**
  - Διαιρούμε το εύρος των τιμών ενός πεδίου σε κάδους
  - Για κάθε διάστημα τιμών, μετράμε τον αριθμό εμφανίσεων





Κόστος της πράξης «Επιλογή»

- **Χωρίς χρήση ευρετηρίου**
  - Αρχείο σωρού ➔ Σειριακό κόστος (γραμμική αναζήτηση)
  - Διατεταγμένο αρχείο ➔ Λογαριθμικό κόστος (δυαδική αναζήτηση)
  - Αρχείο κατακερματισμού ➔ Σταθερό κόστος (απευθείας αναζήτηση)
- **Με χρήση ευρετηρίου**
  - κόστος προσπέλασης στο ευρετήριο (διατεταγμένο ευρετήριο vs. ευρετήριο κατακερματισμού)
    +
  - κόστος προσπέλασης στο κύριο αρχείο

record 0	A-102	Perryridge	400
record 1	A-305	Round Hill	350
record 2	A-215	Mianus	700
record 3	A-101	Downtown	500
record 4	A-222	Redwood	700
record 5	A-201	Perryridge	900
record 6	A-217	Brighton	750
record 7	A-110	Downtown	600
record 8	A-218	Perryridge	700

Κόστος της πράξης «Σύνδεση»

- Ανάλογα με τον αλγόριθμο υλοποίησης της σύνδεσης
  - Σύνδεση με εμφώλευση βρόχων κατά block (Block-nested-loop join)
  - Σύνδεση με εμφώλευση βρόχων μέσω ευρετηρίου (Index-nested-loop join)
  - Σύνδεση με ταξινόμηση και συγχώνευση (Sort-Merge join)
  - Σύνδεση με κατακερματισμό (Hash join)

Block-Nested-Loop Join

- Ψευδοκώδικας για τον υπολογισμό της Θ-σύνδεσης $R \bowtie_\theta S$:

```
Block-Nested-Loop Join
1  FOR each block br in R
2     FOR each block bs in S
3        FOR each tuple r in br
4           FOR each tuple s in bs
5              IF r[A]=s[B] THEN output (r,s)
```

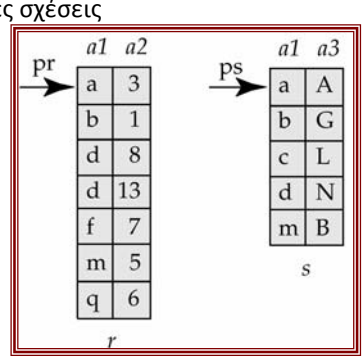
  - Η *R* ονομάζεται εξωτερική σχέση (outer relation) και η *S* εσωτερική σχέση (inner relation) της σύνδεσης.
  - Κόστος: $B_R * B_S + B_R$

Index-Nested-Loop Join

- Αντί να κάνουμε αναζήτηση στο αρχείο μπορούμε να κάνουμε αναζήτηση στο ευρετήριο αν
  - Η σύνδεση είναι ισοσύνδεση (equi-join) ή φυσική σύνδεση (natural join)
  - Υπάρχει ευρετήριο στο χαρακτηριστικό σύνδεσης της εσωτερικής σχέσης
- Για κάθε πλειάδα r της εξωτερικής σχέσης *R*, χρησιμοποιούμε το ευρετήριο (που είναι χτισμένο πάνω στην εσωτερική σχέση *S*) για να βρούμε τις πλειάδες της *S* που ικανοποιούν τη συνθήκη της σύνδεσης.
- Κόστος : $B_R + N_R * C$, όπου
  - *C*, το κόστος για την αναζήτηση στο ευρετήριο και την προσκόμιση όλων των πλειάδων της εσωτερικής σχέσης S που ικανοποιούν τη συνθήκη.

Sort-Merge Join

- Διαδικασία σε 2 βήματα:
  - Βήμα 1: Ταξινομούμε και τις δύο σχέσεις ως προς το χαρακτηριστικό της σύνδεσης
  - Βήμα 2: Συγχωνεύουμε τις ταξινομημένες σχέσεις
- Κόστος: $B_R + B_S +$ κόστος ταξινόμησης των 2 σχέσεων

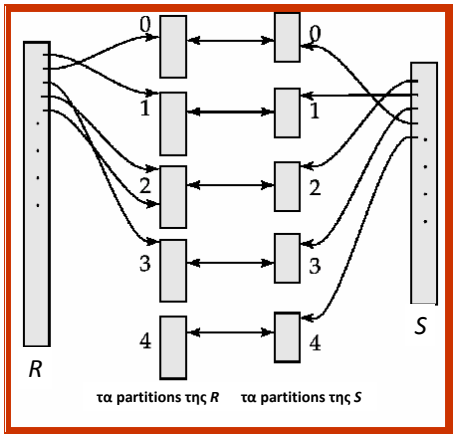


Hash Join

- Εφαρμόζεται μόνο στην περίπτωση ισοσύνδεσης ή φυσικής σύνδεσης.
- **Διαμερισμός** των πλειάδων των δύο σχέσεων σε τμήματα (partitions) βάσει μιας συνάρτησης κατακερματισμού h
- Ψευδοκώδικας (μια απλή εκδοχή):

```
1 Build hash file on S(A) using h
2 FOR each tuple r in R
3     Access relevant bucket b on A
4     FOR each tuple s in S stored in b
5         IF r[A]=s[B] THEN output (r,s)
```

- Η σχέση S ονομάζεται **είσοδος κατασκευής** των τμημάτων (build input) ενώ η σχέση R ονομάζεται **είσοδος διερεύνησης** της σύνδεσης (probe input).

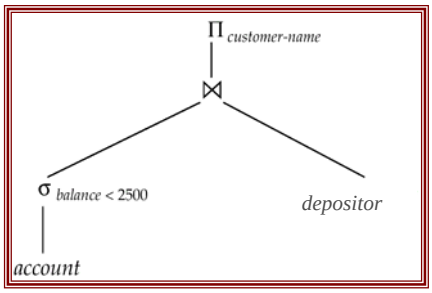


Κόστος άλλων πράξεων: «Προβολή», «Συνάθροιση»

- Η **προβολή** υλοποιείται εφαρμόζοντας προβολή σε κάθε πλειάδα και απαλείφοντας στη συνέχεια τις διπλότυπες.
- Η απαλοιφή των διπλότυπων τιμών μπορεί να γίνει είτε μέσω κατακερματισμού είτε μέσω ταξινόμησης.
  - Στην περίπτωση της ταξινόμησης οι διπλότυπες εγγραφές γειτνιάζουν, και έτσι μπορούν να σβηστούν όλες πλην μίας.
  - Η περίπτωση του κατακερματισμού είναι παρόμοια – οι διπλότυπες εγγραφές θα "μπουν" στο ίδιο τμήμα.
- Η **συνάθροιση** μπορεί να υλοποιηθεί με τρόπο παρόμοιο με αυτόν της απαλοιφής των διπλότυπων πλειάδων.

Κόστος σύνθετων εκφράσεων

- Για την αποτίμηση ενός ολόκληρου δέντρου εκφράσεων (expression tree) υπάρχουν δύο εναλλακτικές μέθοδοι:
  - **Materialization**: "Παραγωγή" των αποτελεσμάτων μιας έκφρασης της οποίας οι είσοδοι είναι είτε απλές σχέσεις είτε έχουν ήδη υπολογιστεί και αποθηκευτεί (materialize) στο δίσκο.
  - **Pipelining**: Υπολογισμός πολλών λειτουργιών ταυτόχρονα, περνώντας τα αποτελέσματα μιας λειτουργίας κατώτερου επιπέδου (child operation) σε μια ανώτερου επιπέδου (parent operation).
    - Ενδέχεται να μην είναι πάντα εφικτό – π.χ., ταξινόμηση, hash-join.

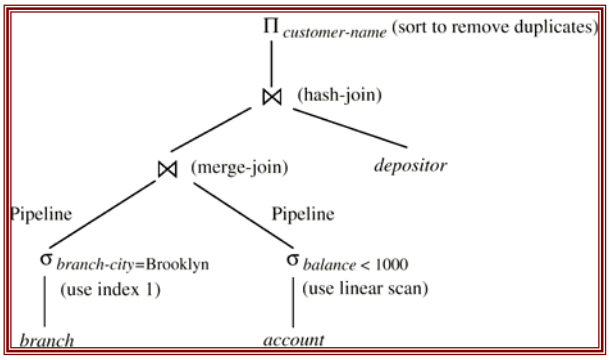


Τελικό αποτέλεσμα: ένα σχέδιο εκτέλεσης

- Η έξοδος του query optimizer είναι ένα σχέδιο εκτέλεσης (**query execution plan – QEP**), το οποίο καθορίζει:

  (α) ποιος αλγόριθμος θα χρησιμοποιηθεί για κάθε επιμέρους λειτουργία, και

  (β) πώς θα συντονιστεί η εκτέλεση των λειτουργιών.



Περιεχόμενα

- Η τυπική αρχιτεκτονική ενός Σχεσιακού ΣΔΒΔ
- Οργάνωση δεδομένων σε φυσικό επίπεδο
  - Αρχεία, Ευρετήρια
- Επεξεργασία ερωτήσεων
  - Ο βελτιστοποιητής ερωτήσεων
- **Διαχείριση δοσοληψιών**
  - **Συγχρονισμός, ανάκαμψη**

Ορισμός της δοσοληψίας

- Μια **δοσοληψία** ή **συναλλαγή** (**transaction**) είναι ένα τμήμα της εκτέλεσης προγράμματος, που προσπελαύνει και πιθανόν ενημερώνει διάφορα αντικείμενα δεδομένων.
- Μια δοσοληψία πρέπει να βλέπει μια συνεπή ΒΔ και όταν επικυρωθεί η δοσοληψία, η ΒΔ πρέπει να είναι πάλι συνεπής.
  - Προσοχή: κατά τη διάρκεια της δοσοληψίας η ΒΔ επιτρέπεται να είναι ασυνεπής.
- Δυο βασικά θέματα πρέπει να αντιμετωπιστούν:
  - Ο **έλεγχος συνδρομικής εκτέλεσης** (concurrency control) πολλαπλών δοσοληψιών.
  - Η **ανάκαμψη συστήματος** (system recovery) μετά από κάποια αποτυχία (αποτυχία υλικού, πτώση συστήματος – crash, κοκ.)

1. read (A);
2. A := A – 50;
3. write (A);
4. read (B);
5. B := B + 50;
6. write (B);

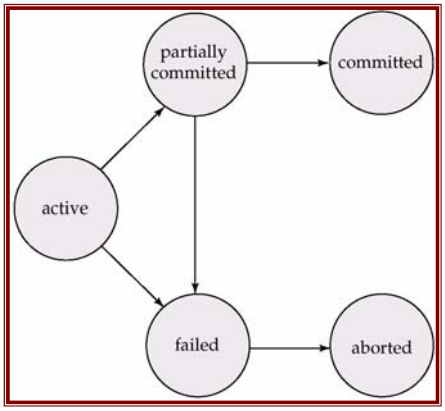
Ιδιότητες ACID

- Για να προστατέψουμε την ακεραιότητα των δεδομένων, το σύστημα της ΒΔ πρέπει να εγγυάται:
  - **Ατομικότητα** (**Atomicity**). Είτε όλες οι πράξεις της δοσοληψίας ανακλώνται στη ΒΔ είτε καμία.
  - **Συνέπεια** (**Consistency**). Η εκτέλεση της δοσοληψίας σε απομόνωση προστατεύει τη συνέπεια της ΒΔ.
  - **Απομόνωση** (**Isolation**). Τα ενδιάμεσα αποτελέ-σματα μιας δοσοληψίας πρέπει να «αποκρύβονται» από άλλες δοσοληψίες που εκτελούνται συνδρομικά.
  - **Διάρκεια** (**Durability**). Αφού μια δοσοληψία ολοκληρωθεί επιτυχώς, οι όποιες τροποποιήσεις έχει επιφέρει στη ΒΔ παραμένουν, ακόμα και αν εκ των υστέρων προκύψουν αποτυχίες του συστήματος.

1. read (A);
2. A := A – 50;
3. write (A);
4. read (B);
5. B := B + 50;
6. write (B);

Καταστάσεις μιας δοσοληψίας

- **Ενεργή** (**Active**), η αρχική κατάσταση. Η δοσοληψία παραμένει σε αυτήν την κατάσταση όσο εκτελείται.
- **Μερικώς επικυρωμένη** (**Partially committed**), μετά την εκτέλεση και της τελευταίας εντολής.
- **Αποτυχημένη** (**Failed**), μετά τη διαπίστωση ότι δε μπορεί να προχωρήσει η εκτέλεση.
- **Ακυρωμένη** (**Aborted**), σε περίπτωση υποχώρησης της δοσοληψίας (rollback), οπότε η ΒΔ έχει επιστρέψει στην κατάσταση που ήταν πριν ξεκινήσει η δοσοληψία.
- **Επικυρωμένη** (**Committed**), μετά την επιτυχή ολοκλήρωση.



Έλεγχος συνδρομικής εκτέλεσης δοσοληψιών

- Το ΣΔΒΔ πρέπει να επιτρέπει τη συνδρομικότητα (concurrency). Τα πλεονεκτήματα είναι:
  - Καλύτερη εκμετάλλευση των πόρων του Η/Υ
  - Μείωση του μέσου χρόνου απόκρισης των δοσοληψιών
- Πρόβλημα: ένα μη σειριακό χρονοπρόγραμμα (schedule) πρέπει να είναι σειριοποιήσιμο, δηλ. ισοδύναμο με ένα σειριακό
  - Το S_1 είναι σειριοποιήσιμο, το S_2 όχι.

S_1

T_1	T_2
read(A)	
$A := A - 50$	
write(A)	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
read(B)	
$B := B + 50$	
write(B)	
	read(B)
	$B := B + temp$
	write(B)

Έλεγχος συνδρομικής εκτέλεσης δοσοληψιών

- Το ΣΔΒΔ πρέπει να επιτρέπει τη συνδρομικότητα (concurrency). Τα πλεονεκτήματα είναι:
  - Καλύτερη εκμετάλλευση των πόρων του Η/Υ
  - Μείωση του μέσου χρόνου απόκρισης των δοσοληψιών
- Πρόβλημα: ένα μη σειριακό χρονοπρόγραμμα (schedule) πρέπει να είναι σειριοποιήσιμο, δηλ. ισοδύναμο με ένα σειριακό
  - Το S_1 είναι σειριοποιήσιμο, το S_2 όχι.

S_2

T_1	T_2
read(A)	
$A := A - 50$	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
write(A)	
read(B)	
$B := B + 50$	
write(B)	
	$B := B + temp$
	write(B)

Συνδρομικότητα βασισμένη σε κλειδιά

- Τα αντικείμενα μιας ΒΔ μπορούν να κλειδωθούν με δυο τρόπους:
  - Αποκλειστικός - exclusive (X) τρόπος. Το αντικείμενο μπορεί και να διαβαστεί και να γραφεί.
  - Διαμοιραζόμενος - shared (S) τρόπος. Το αντικείμενο μπορεί μόνο να διαβαστεί.
- Η διαχείριση των αιτήσεων παραχώρησης κλειδιών γίνονται από το διαχειριστή κλειδωμάτων (lock administrator) .

Πίνακας συμβατότητας κλειδωμάτων

	S	X
S	true	false
X	false	false

- Εάν δεν μπορεί να παραχωρηθεί κλειδί σε μια δοσοληψία, αυτή περιμένει έως ότου ελευθερωθούν όλα τα μη συμβατά κλειδιά που διατηρούνται από άλλες δοσοληψίες. Στη συνέχεια τής παραχωρείται το κλειδί που έχει ζητήσει.
- Τα κλειδώματα / ξεκλειδώματα πρέπει να γίνουν με κάποιο έξυπνο τρόπο
  - **Πρωτόκολλο κλειδώματος 2 φάσεων** (2 phase locking – 2PL): φάση εξάπλωσης – φάση συρρίκνωσης

Προβλήματα λόγω των κλειδωμάτων

- Θεωρήστε το ακόλουθο «παράθυρο» ενός χρονοπρογράμματος
  - Για να προχωρήσει η μία δοσοληψία, περιμένει την άλλη να ελευθερώσει κάποιο κλειδί (που δεν μπορεί να γίνει)
- Μια τέτοια κατάσταση ονομάζεται **αδιέξοδο** (**deadlock**).
  - Για την αντιμετώπιση του αδιεξόδου, μια από τις δύο δοσοληψίες πρέπει να υποχωρήσει (rollback), ώστε τα κλειδιά που κατέχει να ελευθερωθούν.

S_3

T_3	T_4
lock-X(B)	
read(B)	
$B := B - 50$	
write(B)	
	lock-S(A)
	read(A)
	lock-S(B)
lock-X(A)	

  - Εάν μια δοσοληψία υποχωρεί κατ' εξακολούθηση τότε έχουμε το φαινόμενο της **λιμοκτονίας** (**starvation**)

Ανάκαμψη δοσοληψιών

- Ανακάμψιμο χρονοπρόγραμμα — εάν μια δοσοληψία T_j διαβάζει ένα αντικείμενο που έχει προηγουμένως γραφτεί από μια δοσοληψία T_i , η λειτουργία επικύρωσης (commit) της T_i πρέπει να προηγείται της λειτουργίας επικύρωσης της T_j.
  - Το διπλανό χρονοπρόγραμμα είναι μη ανακάμψιμο εάν η T_9 επικυρωθεί (commit) αμέσως μετά την εντολή read(A).
  - Εάν στη συνέχεια ακυρωθεί η T_8, η T_9 θα έχει διαβάσει μια ασυνεπή κατάσταση της ΒΔ.

S_4

T_8	T_9
read(A)	
write(A)	
	read(A)
read(B)	

- Ο διαχειριστής δοσοληψιών του ΣΔΒΔ πρέπει να εξασφαλίζει ότι τα χρονοπρογράμματα είναι ανακάμψιμα.
  - Αλγόριθμοι ανάκαμψης

Ανάκαμψη βασισμένη στο ημερολόγιο συστήματος

- Ένα **ημερολόγιο συστήματος** (**log**) φυλάσσεται στην ακλόνητη αποθήκευση (θεωρητικά, δεν καταστρέφεται ποτέ!)
- Δυο προσεγγίσεις με χρήση ημερολογίου:
  - Ετεροχρονισμένη τροποποίηση της ΒΔ
    - οι τροποποιήσεις της ΒΔ καταγράφονται στο ημερολόγιο
    - η (πραγματική) γραφή στη ΒΔ καθυστερεί έως ότου η δοσοληψία επικυρωθεί μερικώς.
  - Άμεση τροποποίηση της ΒΔ
    - οι τροποποιήσεις της ΒΔ καταγράφονται στο ημερολόγιο
    - η (πραγματική) γραφή στη ΒΔ μπορεί να γίνει οποιαδήποτε στιγμή μετά

Ετεροχρονισμένη vs. άμεση τροποποίηση ΒΔ

- Παράδειγμα log για ετεροχρονισμένη τροποποίηση της ΒΔ:

(a)	(b)	(c)
<T_0 start>	<T_0 start>	<T_0 start>
<T_0, A, 950>	<T_0, A, 950>	<T_0, A, 950>
<T_0, B, 2050>	<T_0, B, 2050>	<T_0, B, 2050>
	<T_0 commit>	<T_0 commit>
	<T_1 start>	<T_1 start>
	<T_1, C, 600>	<T_1, C, 600>
		<T_1 commit>

- Παράδειγμα log για άμεση τροποποίηση της ΒΔ:

(a)	(b)	(c)
<T_0 start>	<T_0 start>	<T_0 start>
<T_0, A, 1000, 950>	<T_0, A, 1000, 950>	<T_0, A, 1000, 950>
<T_0, B, 2000, 2050>	<T_0, B, 2000, 2050>	<T_0, B, 2000, 2050>
	<T_0 commit>	<T_0 commit>
	<T_1 start>	<T_1 start>
	<T_1, C, 700, 600>	<T_1, C, 700, 600>
		<T_1 commit>

Περαιτέρω μελέτη

- Hellerstein & Stonebraker: Anatomy of a Database System (1998)
  - μια ενδοσκόπηση των σύγχρονων ΣΔΒΔ

Donald Knuth and the B$^+$-tree

- About the naming of B+-trees:
  - A variation of B-trees, B*-Trees, was named by Donald Knuth in The Art of Computer Programing, Vol. 3, Searching and Sorting. Knuth proposed that B*-Trees had nodes that were at least 2/3 full rather than 1/2 full. Knuth also proposed another B-Tree variation having all keys in the leaves of the tree and the leaves are linked from left-to-right (making B$^+$-Tree ideal for sequential search) Comer says that Knuth did not name this variation, but Comer chooses to call it B$^+$-Tree. (source: http://mactech.com/articles/mactech/Vol.06/06.11/Nov90Letters/index.html)

Donald Knuth