Σχεδίαση Βάσεων Δεδομένων και Κατανεμημένες Βάσεις Δεδομένων

Πρώτη Εργασία

ΜΕΛΗ ΟΜΑΔΑΣ
ΙΩΑΝΝΗΣ ΣΟΜΟΣ, 21685
ΕΥΑΓΓΕΛΟΣ ΚΟΝΤΟΣ, 21641
ΑΝΔΡΕΑΣ ΜΑΥΡΟΠΟΥΛΟΣ, 217129

Εκφώνηση : 1ο μέρος – Ε*ντολές δημιουργίας ευρετηρίων Βρείτε τη σύνταξη της εντολής CREATE INDEX σε ORACLE, MySQL και PostgreSQL. Εξηγείστε τις παραμέτρους κάθε εντολής σε σχέση με τους διαθέσιμους τύπους ευρετηρίου που έχουμε δει στο μάθημα. Παρουσιάστε συγκριτικά (σε ένα πίνακα) τις δυνατότητες των τριών ΣΔΒΔ, ποια ευρετήρια υποστηρίζουν και ποια όχι η κάθε μια. Αν χρησιμοποιήσατε πηγές από το δίκτυο να τις παραθέσετε (URL διευθύνσεις).*

Σύνταξη

Oracle SQL

```
CREATE [UNIQUE | BITMAP] INDEX [schema.]index
ON {cluster_index_clause | table_index_clause |
    bitmap_join_index_clause}
```

Η Oracle εξ ορισμού δημιουργεί ευρετήρια με τη χρήση B-δένδρων (BTREE), μπορεί όμως να χρησιμοποιήσει και κατακερματισμό (HASH). Υποστηρίζει απλά ευρετήρια, πρωτεύοντα ευρετήρια (UNIQUE), ευρετήρια συστάδων (cluster_index_clause), καθώς και ευρετήρια τύπου bitmap (BITMAP).

MySQL

```
CREATE [UNIQUE | FULLTEXT | SPATIAL] INDEX index_name
[USING {BTREE | HASH}] ON tbl_name (key_part, ...)
[index_option] [algorithm_option | lock_option] ...
```

Η MySQL υποστηρίζει απλά ευρετήρια, πρωτεύοντα ευρετήρια, ευρετήρια κειμένου (FULLTEXT) και χωρικά ευρετήρια (SPATIAL). Χρησιμοποιεί B-δένδρα, αλλά ορισμένες μηχανές αποθήκευσης υποστηρίζουν και κατακερματισμό. Όταν υπάρχει primary key ή κάποιο unique field, δημιουργεί ευρετήριο συστάδων.

PostgreSQL

```
CREATE [UNIQUE] INDEX [CONCURRENTLY]
[[IF NOT EXISTS] name ON table_name
[USING {BTREE | HASH | GIN | [SP-]GIST | BRIN}]
({column_name | (expression)} [COLLATE collation]
  [opclass] [ASC | DESC] [NULLS {FIRST | LAST}][, ...])
[WITH (storage_parameter = value[, ...])]
[TABLESPACE tablespace_name] [WHERE predicate]
```

Η PostgreSQL υποστηρίζει απλά και πρωτεύοντα ευρετήρια. Εξ ορισμού χρησιμοποιεί B-δένδρα και μπορεί επίσης να χρησιμοποιήσει κατακερματισμό, ανεστραμμένα ευρετήρια (GIN), ανεστραμμένα δένδρα αναζήτησης (GIST), χωρικά κατακερματισμένα GIST (SP-GIST) και ευρετήρια εύρους block (BRIN).

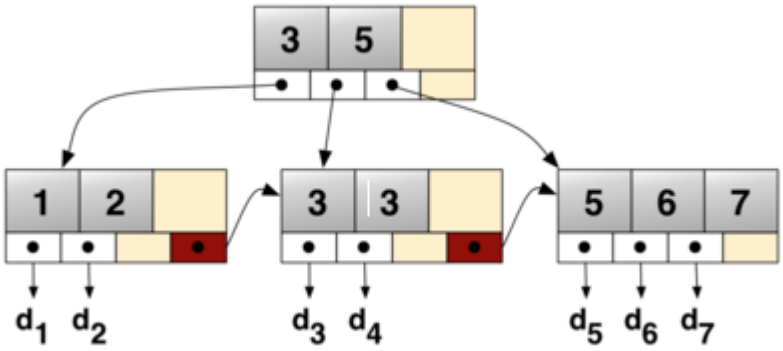
Δυνατότητες

Database / Index Type	**Oracle SQL**	**MySQL**	**PostgreSQL**
Πρωτεύον (Unique)	UNIQUE	UNIQUE	UNIQUE
Συστάδων (Clustered)	CLUSTER	*(primary key/unique field)*	CLUSTER
Β-δένδρων (B-tree)	BTREE	BTREE	BTREE
Κατακερματισμένο (Hashed)	HASH	HASH	HASH,SP-GIST
Ανεστραμμένο (Inverted)	-	FULLTEXT	GIN,GIST,SP-GIST
Χωρικό (Spatial)	-	SPATIAL	SP-GIST
Εύρους block (Block range)	-	-	BRIN
Bitmap	BITMAP	-	-

Πηγές

- Oracle SQL Docs
- MySQL Docs
- PostgreSQL Docs
- A tour of Postgres Index Types

Εκφώνηση: 2ο μέρος – ***Ευρετήριο με B+δένδρο Ένα αρχείο έχει 20.000.000 εγγραφές αταξινόμητες σε σωρό. Οι εγγραφές έχουν κατά μέσο όρο μέγεθος 250 bytes και αποθηκεύονται σε αρχεία σε block μεγέθους 2048 bytes με μη εκτατή καταχώρηση. Πάνω σε ένα αριθμητικό πεδίο (μεγέθους 8 bytes) που μπορεί και να περιέχει διπλότυπα, έχει χτιστεί ευρετήριο με χρήση B+ δένδρου. Κάθε δείκτης προς τις εγγραφές του αρχείου (δείκτες d στην εικόνα) έχει μέγεθος 16 Bytes, κάθε δείκτης προς block του ευρετηρίου μαζί και ο δείκτης προς επόμενο φύλλο έχει μέγεθος 8 Bytes. Το ευρετήριο δημιουργείται μαζικά για όλο το αρχείο και μπορείτε να θεωρήσετε ότι οι κόμβοι στο τελευταίο επίπεδο του δέντρου είναι όσο γίνεται πλήρεις.***



r : συνολικές εγγραφές
R: σταθερό μέγεθος κάθε εγγραφής.
B: μέγεθος block που αποθηκεύονται
SF: Μέγεθος αρχείου δεδομένων
Bfr: blocking factor (Πόσες εγγραφές χωρούν σε ένα block)
V: μέγεθος του πεδίου ευρετηρίασης (δείκτης δεδομένων)
b: Αριθμός blocks για την αποθήκευση ενός αρχείου r εγγραφών:
P: δείκτης block
PR
Pl: P για κάθε leaf
p: classes
Fo-internal node

Τυπολόγιο:
bfr = B/R
SF = Ra/floor
PR = V + P
Pl (PR+V)+P>=B*
(p P) +((p-1)V)B

1) Το μέγεθος του αρχείου αν είναι αρχείο σωρού:

Αρχείο σωρού:

r: 20.000.000
R: 250
B:2048

bfr: B/R = 2048/250 = 8. Άρα χωράνε 8 εγγραφές σε κάθε block
(8,192 με στρογγυλοποίηση προς τα κάτω)

Άρα μέγεθος αρχείου δεδομένων:

b: r/bfr=20.000.000/8=2.5000.000 : 2,5 * 10^6 blocks.

2) Το μέγεθος του αρχείου αν είναι αρχείο κατακερματισμού:

Επειδή στα αρχεία κατακερματισμού θέλουμε να αποφύγουμε πιθανές συγκρούσεις, τα buckets είναι 20% άδεια.
100*(2,5*10^6) /80) = 3,125 * 10^6 blocks.

3) Πόσα επίπεδα έχει το B+ δένδρο συμπεριλαμβανομένου και του τελευταίου επιπέδου; &

4) Πόσους κόμβους θα περιέχει το κάθε επίπεδό του, ποιο το μέγεθος του ευρετηρίου συνολικά;

Γνωρίζουμε από τη θεωρία πως οι κόμβοι στα φύλλα είναι όσο πιο πυκνοί γίνονται
Το V είναι το μήκος πεδίου κλειδιού αναζήτησης και ισούται με 8bytes
Το P είναι το δείκτης block=8 bytes.
PR: P + V = 16 bytes
Pleaves * (PR+V)+P>=B
Pleaves *(16+8) + 8 >= 2048
Pleaves *24>=2040
Pleaves >=2040/24
Pleaves >=85 και επιλέγουμε την μεγαλύτερη τιμή που ικανοποιεί αυτή την σχέση άρα Pl=85bytes

Έπειτα, μέσω του τύπου υπολογίζουμε το p που έπειτα θα χρησιμοποιήσουμε για την διακλάδωση.
(p P) +((p-1)V)B
(p 8) +((p-1)8)2048
8p +8p-82048
16p2056
p128.5
(p P) +((p-1)V)<=B
(p 8) +((p-1)8)<=2048
8p +8p-8<=2048
16p<=2056 p<=128.5 και επιλέγουμε την μεγαλύτερη τιμή που ικανοποιεί αυτή την σχέση άρα p = 128
Στα B+ δένδρα θέλουμε τα επίπεδα πλην των φύλλων να είναι γεμάτα ώστε να μην χρειαζόμαστε αναδιοργάνωση.
Ο παράγοντας διακλάδωσης θα είναι :
(Από Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom, Συστήματα Βάσεων Δεδομένων)
Fo-internal node = p * 0,5 = 64.
bleaves = r / Fo-leaves = r / Pleaves = 20.000.000 / 85 = 235.294 blocks πυκνού ευρετηρίου.
Bl1 = bleaves / Fo-internal node = 235.294,118 / 64 = 3.676 blocks
Bl2 = Bl1 / Fo-internal node = 58 blocks
Broot = Bl2 / Fo-internal node = 1 block άρα είναι η ρίζα , άρα έχει 4 επίπεδα
Επίπεδο 0 : **Bleaves** με 235.294 κόμβους
Επίπεδο 1 : **Bl1** με 3676 κόμβους
Επίπεδο 2: **Bl2** με 58 κόμβους
Επίπεδο 3 : **Broot** με 1 κόμβο.

Άρα το μέγεθος του ευρετηρίου συνολικά είναι:
235294+3676+58+1 = 23929 blocks.

5) Αν το δέντρο σας γίνει B ποια η απάντηση στο 4;*

(Θεωρία: Οι ενδιάμεσοι κόμβοι πρέπει να είναι γεμάτοι γεμάτοι κατά τα 2/3 και όχι κατά το 1/2 όπως στα B+ δένδρα.)
Για ένα δέντρο B* θα εφαρμόσουμε την ίδια διαδικασία με πριν αλλά γνωρίζοντας ότι το δέντρο γεμίζει κατά 2/3 και όχι στο μισό έχουμε ότι:

Fo-internal node = p * 0,66 = 84.

bleaves = r / Fo-leaves = r / Pleaves = 20.000.000 / 85 = 235.294 blocks πυκνού ευρετηρίου.

Bl1 = bleaves / Fo-internal node = 235.294,118 / 84 = 2802 blocks

Bl2 = Bl1 / Fo-internal node = 34 blocks

Broot = Bl2 / Fo-internal node = 1 block άρα είναι η ρίζα , άρα έχει 4 επίπεδα

Επίπεδο 0 : **Bleaves** με 235.294 κόμβους

Επίπεδο 1 : **Bl1** με 2802 κόμβους

Επίπεδο 2: **Bl2** με 34 κόμβους

Επίπεδο 3 : **Broot** με 1 κόμβο.

Άρα το μέγεθος του ευρετηρίου συνολικά είναι:

235294+2802+34+1 = 238132 blocks.

6) Ποιο θα είναι το κόστος αναζήτησης ισότητας για μια συγκεκριμένη τιμή που γνωρίζετε ότι εμφανίζεται 5 φορές σε όλο το αρχείο;

Κάθε φορά που μέσω αναζήτησης βρίσκουμε μία ζητούμενη τιμή για να γίνει output κοστίζει 1 μονάδα I/O. Επίσης κάθε φορά που πηγαίνουμε από ένα επίπεδο σε ένα άλλο προστίθεται 1 ακόμα μονάδα στο συνολικό κόστος αναζήτησης. Άρα αν υποθέσουμε ότι όλα τα διπλότυπα είναι στο πρώτο επίπεδο τότε θα ήταν το λιγότερο δυνατό κόστος: 5 I/O μονάδες για τα 5 output που χρειάστηκαν να γίνουν. Ενώ το μέγιστο κόστος θα ήταν έστω και ένα διπλότυπο να ήταν στο τελευταίο επίπεδο άρα 5 μονάδες I/O για το output και 4 ακόμα για την μετάβαση από κάθε επίπεδο με σύνολο 9 I/O.

Άρα ο μέσος όρος θα είναι: **(5+9) / 2 = 7**.