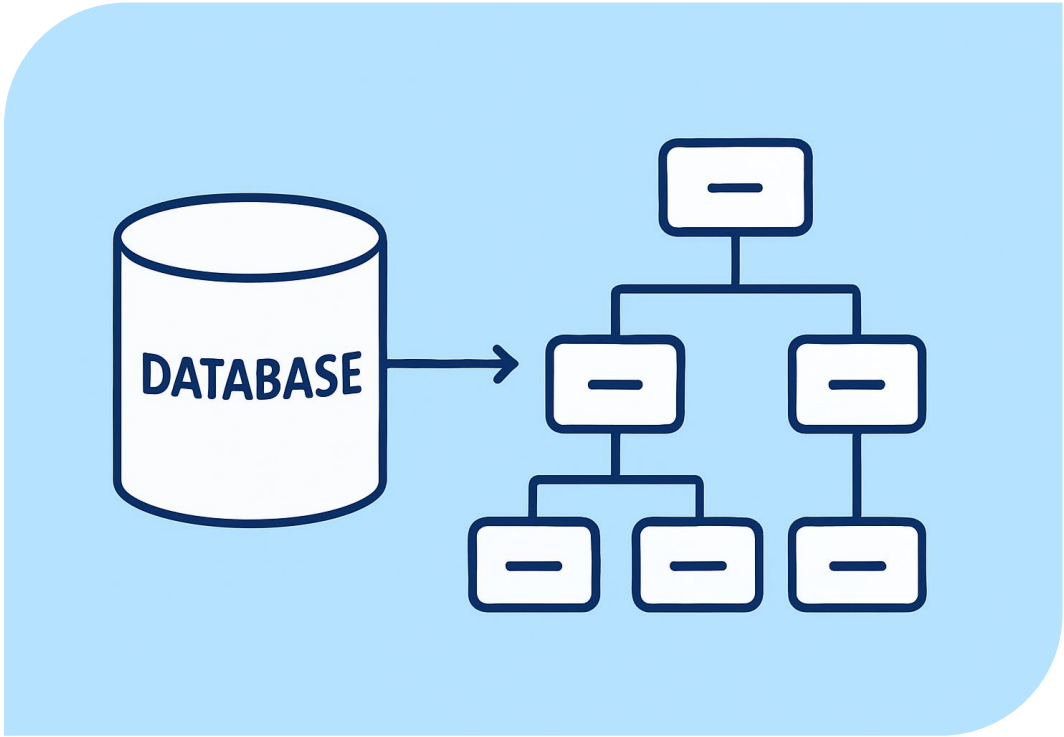
Φυσική Σχεδίαση Βάσεων Δεδομένων (Μέρος 2ο)

Επανάληψη, ασκήσεις και ένα παράδειγμα εφαρμογής Β δέντρων

Φροντιστήριο
Βοηθός Μαθήματος: Μιχαήλ Γαλλιάκης
(email: csdp1413@csd.uoc.gr)
HY-360 Αρχεία και Βάσεις Δεδομένων
12/12/2025



Τι θα δούμε σήμερα

- Λίγη Επανάληψη Θεωρίας
- Ασκήσεις
- Ένα παράδειγμα εφαρμογής Β δέντρων, σε πραγματική βάση δεδομένων
- Q&A για το μάθημα και το Project

Σύγκριση

Μέθοδος Οργάνωσης	Κόστος Εισαγωγής (Insertion)	Κόστος Αναζήτησης (Equality Search)	Αναζήτηση Εύρους (Range Query)	Βασικά Χαρακτηριστικά
Σωρός (Heap File)	Πολύ Χαμηλό (Εγγραφή στο τέλος του αρχείου - 2 block accesses)	Υψηλό (Γραμμική σάρωση όλου του αρχείου - N/2)	Πολύ Υψηλό (Απαιτείται ανάγνωση όλων των blocks)	Η πιο αποδοτική μέθοδος για μαζική φόρτωση δεδομένων, αλλά αναποτελεσματική για ανάκτηση.
Ταξινομημένο (Sorted File)	Πολύ Υψηλό (Απαιτεί μετακίνηση εγγραφών για διατήρηση ταξινόμησης)	Χαμηλό (Δυαδική αναζήτηση - $\log_2 N$)	Χαμηλό (Εύρεση αρχής και διαδοχική ανάγνωση)	Ιδανικό για δεδομένα που δεν αλλάζουν συχνά (read-only) και για ερωτήματα εύρους.
Κατακερματισμός (Hashed File)	Χαμηλό (Υπολογισμός hash function και εγγραφή στο bucket)	Ελάχιστο / Σταθερό (Άμεση πρόσβαση μέσω hash - O(1))	Υψηλό (Απαιτείται γραμμική σάρωση όλου του αρχείου)	Βέλτιστη απόδοση για αναζητήσεις ισότητας (exact match). Ακατάλληλο για αναζητήσεις εύρους.
Δέντρα (B-Trees / B+ Trees)	Χαμηλό / Ισορροπημένο (Κόστος ίσο με το ύψος του δέντρου)	Χαμηλό / Ισορροπημένο (Κόστος ίσο με το ύψος του δέντρου)	Χαμηλό (Ακολουθιακή ανάγνωση των φύλλων)	Η πιο διαδεδομένη δομή. Προσφέρει ισορροπία απόδοσης σε όλες τις λειτουργίες.

Άσκηση 1: Πλατφόρμα "MusicStream"

Η εφαρμογή streaming "MusicStream" αποθηκεύει τραγούδια στον πίνακα:
Songs(SongID, Title, Artist, Genre, Year)

Ο μηχανικός βάσεων εξετάζει 4 δομές αποθήκευσης:

1. Heap File (Σωρός)
2. Sorted File (Ταξινομημένο στο SongID)
3. Hashed File (Hash στο SongID)
4. B+ Tree (Ευρετήριο στο SongID)

Ερωτήματα:

1. Ποια δομή είναι η βέλτιστη αν μας ενδιαφέρει αποκλειστικά η ταχύτητα εισαγωγής νέων τραγουδιών (New Releases);
2. Ποια δομή δίνει τον γρηγορότερο χρόνο για να βρούμε ένα τραγούδι αν ξέρουμε το SongID του;
3. Ποια δομή θα επιλέγατε για να βρείτε όλα τα τραγούδια του είδους "Rock";
4. Ποια δομή έχει τις μεγαλύτερες απαιτήσεις σε "χαμένο" χώρο (waste space);

Άσκηση 1: Ανάλυση & Αιτιολόγηση

1. Εισαγωγή: Heap File.
    - Γιατί: Είναι απλό append στο τέλος (2 I/O). Το B-Tree είναι πιο αργό (θέλει ταξινόμηση/splits), το Hash και το Sorted επίσης.
2. Αναζήτηση (SongID): Hashed File.
    - Γιατί: Άμεση πρόσβαση (O(1)). Το B-Tree είναι γρήγορο (O(log N)) αλλά όχι όσο το Hash.
3. Αναζήτηση (Genre): Heap File.
    - Γιατί: Κανένα ευρετήριο δεν είναι στο Genre. Πρέπει να διαβάσουμε όλο το αρχείο. Το Heap είναι το πιο συμπαγές, άρα διαβάζουμε λιγότερα blocks.
4. Χώρος: Hashed File.
    - Γιατί: Δεσμεύουμε χώρο για buckets που μπορεί να μείνουν άδεια. Το B-Tree έχει επίσης κενά (fill factor ~69%), αλλά το Hash συνήθως σπαταλάει περισσότερο αν δεν ρυθμιστεί τέλεια.

Οι Παράμετροι d και e: Τι σημαίνουν;

Φανταστείτε ότι έχουμε Blocks σταθερού μεγέθους στον δίσκο.

- Το e (για τα Δεδομένα): Μας λέει πόσα Records μπορούμε να χωρέσουμε σε ένα block στα Φύλλα του δέντρου.
- Το d (για το Ευρετήριο): Μας λέει πόσες μικρές "Ταμπελίτσες-Οδηγούς" (Keys + Pointers) χωράνε μέσα στα blocks του Ευρετηρίου για να μας καθοδηγούν.

Ο Κανόνας της Πληρότητας:
Κάθε κουτί πρέπει να είναι τουλάχιστον μισογεμάτο για να θεωρείται έγκυρο (εκτός από τη Ρίζα).

Τύπος Κόμβου	Παράμετρος	Ελάχιστο (Min)	Μέγιστο (Max)
Ευρετήριο (Index Node) (Έχει δείκτες & κλειδιά)	d	d δείκτες (τουλάχιστον d-1 κλειδιά)	2d δείκτες 2d - 1 κλειδιά
Φύλλο (Data Node) (Έχει τις εγγραφές)	e	e εγγραφές	2e - 1 εγγραφές

Άσκηση 2: Υπολογισμός των d και e

Έστω ότι θέλουμε να οργανώσουμε ένα αρχείο φοιτητών χρησιμοποιώντας B+ Δέντρο.
Δίνονται τα εξής τεχνικά χαρακτηριστικά του δίσκου και των δεδομένων:

- Μέγεθος Block (B): 2048 bytes
- Μέγεθος Εγγραφής(Rec): 120 bytes
- Μέγεθος Κλειδιού (Key): 12 bytes
- Μέγεθος Δείκτη (Ptr): 4 bytes

Χρήσιμοι Τύποι (Τυπολόγιο):

- Blocking Factor (R): $\lfloor B / Rec \rfloor$
- Χωρητικότητα Φύλλου (Records): $2e - 1 \leq R$
- Χωρητικότητα Ευρετηρίου (Keys): $2d - 1 \leq \lfloor B/(Key+Ptr) \rfloor$

Ζητείται:
Να υπολογίσετε τις τιμές των παραμέτρων d (για τους κόμβους ευρετηρίου) και e (για τα φύλλα).

Λύση Άσκησης 2

Υπολογισμός e (Για τα Φύλλα - Data Nodes):

- R = ⌊2048 / 120⌋ = ⌊17.07⌋ = 17.
- Άρα Max Εγγραφές= 2e - 1 ≤ 17.
- 2e - 1 ≤ 17 → 2e ≤ 17+1 → 2e ≤ 18 →e ≤ 18/2 → e ≤ 9. Άρα **e = 9**.

Υπολογισμός d (Για το Ευρετήριο - Index Nodes):

- Index Entry Size = Key + Pointer = 12+ 4 = 16 bytes.
- Entries ανά Block = ⌊2048 / 16⌋ = ⌊128⌋ = 128.
- Max κλειδιά = 2d - 1 ≤ 128→ 2d ≤ 129→ d ≤ 64.5. Άρα το **d = 64**.

Άσκηση 3: Υπολογισμός των d και e

Έστω ότι θέλουμε να οργανώσουμε ένα αρχείο φοιτητών χρησιμοποιώντας B+ Δέντρο.
Δίνονται τα εξής τεχνικά χαρακτηριστικά του δίσκου και των δεδομένων:

- Μέγεθος Block (B): 4096 bytes
- Μέγεθος Εγγραφής(Rec): 300 bytes
- Μέγεθος Κλειδιού (Key): 36 bytes
- Μέγεθος Δείκτη (Ptr): 4 bytes

Χρήσιμοι Τύποι (Τυπολόγιο):

- Blocking Factor (R): $\lfloor B / Rec \rfloor$
- Χωρητικότητα Φύλλου (Records): $2e - 1 \leq R$
- Χωρητικότητα Ευρετηρίου (Keys): $2d - 1 \leq \lfloor B/(Key+Ptr) \rfloor$

Ζητείται:
Να υπολογίσετε τις τιμές των παραμέτρων d (για τους κόμβους ευρετηρίου) και e (για τα φύλλα).

Λύση Άσκησης 3

Υπολογισμός e (Για τα Φύλλα - Data Nodes):

- R = ⌊4096 / 300⌋ = ⌊13.65⌋ = 13.
- Άρα Max Εγγραφές= 2e - 1 ≤ 13.
- 2e - 1 ≤ 13 → 2e ≤ 13+1 → 2e ≤ 14 →e ≤ 14/2 → e ≤ 7. Άρα **e = 7**.

Υπολογισμός d (Για το Ευρετήριο - Index Nodes):

- Index Entry Size = Key + Pointer = 36 + 4 = 40 bytes.
- Entries ανά Block = ⌊4096 / 40⌋ = ⌊102.4⌋ = 102.
- Max κλειδιά = 2d - 1 ≤ 102 → 2d ≤ 103 → d ≤ 51.5. Άρα το **d = 51**.

Άσκηση 4: Οργάνωση Κατακερματισμού

Δίνονται οι τιμές: {20, 35, 12, 28, 13, 5, 16, 1, 23, 45, 8, 9, 41}
Συνάρτηση Hash: h(x) = x mod 8

Χωρητικότητα Bucket: 3 εγγραφές

Ζητείται: Να σχεδιάσετε τη δομή αποθήκευσης (Buckets).

Διαδικασία: Για κάθε αριθμό, υπολογίζουμε το υπόλοιπο της διαίρεσης με το 8 και τον τοποθετούμε στο αντίστοιχο κουτί (Bucket).

Λύση άσκησης 4 - Περιεχόμενα Buckets

20 mod 8 = 4
35 mod 8 = 3
12 mod 8 = 4
28 mod 8= 4
13 mod 8 = 5
5 mod 8 = 5

16 mod 8 = 0
1 mod 8 = 1
23 mod 8 = 7
45 mod 8 = 5
8 mod 8 = 0
9 mod 8 = 1
41 mod 8 = 1

Διαδικασία: Για κάθε αριθμό, υπολογίζουμε το υπόλοιπο της διαίρεσης με το 8 και τον τοποθετούμε στο αντίστοιχο κουτί (Bucket).

Bucket ID	Περιεχόμενα (Εγγραφές)
0	16, 8
1	1, 9, 41
2	(κενό)
3	35
4	20, 12. 28
5	13, 5, 45
6	(κενό)
7	23

Βλέπουμε ότι τα Buckets 1, 4 και 5 γέμισαν. Αν ερχόταν άλλη μία εγγραφή για αυτά (π.χ. το 60 για το bucket 4), θα είχαμε Overflow και θα χρειαζόμασταν αλυσίδα.

Άσκηση 5: Κατασκευή B+ Δέντρου

Έστω ότι παράμετροι: d=3, e=3.
Και ακολουθία εισαγωγής:
{20, 35, 12, 28, 13, 5, 16, 1, 23, 45, 8, 9, 41}

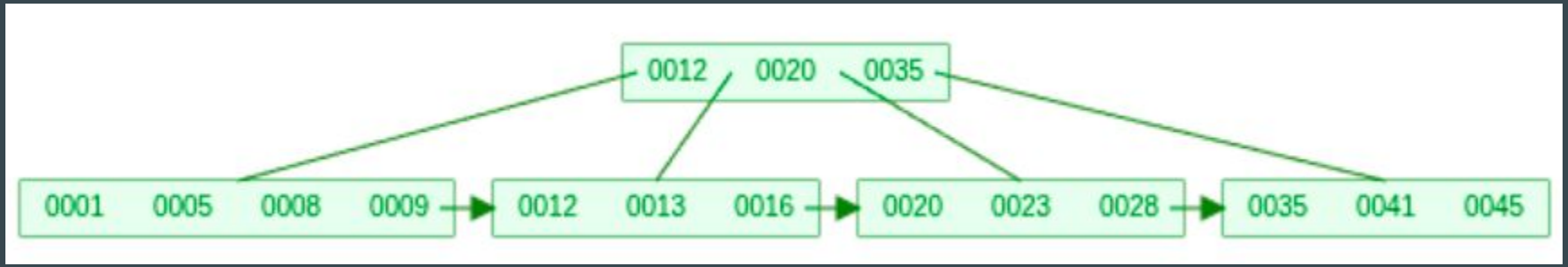
Άρα:
Max χωρητικότητα Φύλλων (Data):
2e - 1 = 5 εγγραφές.
Max χωρητικότητα Ευρετηρίου (Index):
2d - 1 = 5 κλειδιά.
2d = 6 δείκτες

Κανόνες Split (Διάσπασης):

1. Στα Φύλλα: Μοιράζουμε τις εγγραφές. Το πρώτο κλειδί του δεξιού κόμβου Αντιγράφεται (Copy Up) στον πατέρα.
2. Στους Εσωτερικούς Κόμβους: Το μεσαίο κλειδί Ανεβαίνει (Push Up) στον πατέρα και αφαιρείται από τον κόμβο.

Άσκηση 5: Τελική Μορφή του B+ Tree

Ακολουθώντας τους κανόνες διάσπασης, φτάνουμε σε αυτό το δέντρο.



Ας το φτιάξουμε βήμα βήμα.
https://www.cs.usfca.edu/~galles/visualization/BPlusTree.html

Άσκηση 6: Κατασκευή B+ Δέντρου

Έστω ότι παράμετροι: d=2, e=2.
Και ακολουθία εισαγωγής:
{8,19,56, 37, 68, 45,11,31}

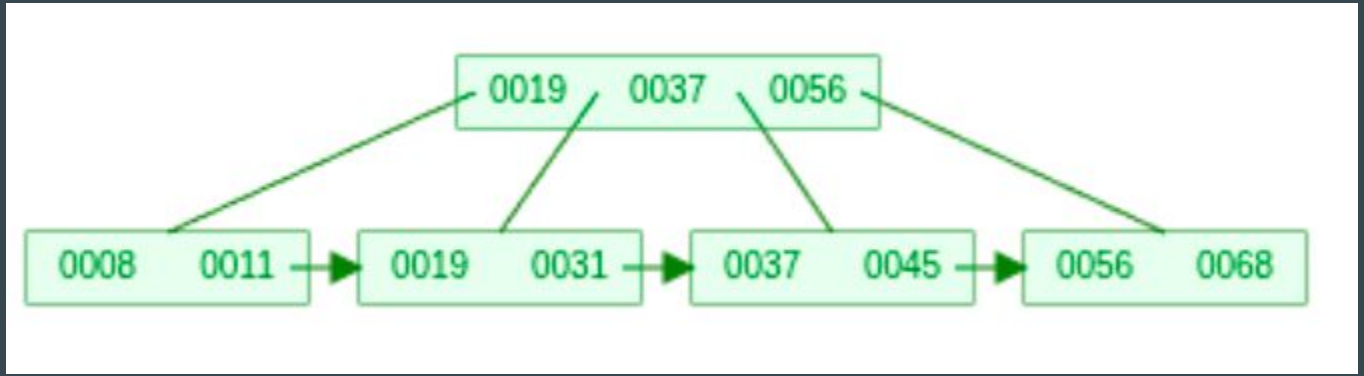
Άρα:
Max χωρητικότητα Φύλλων (Data):
2e - 1 = 3 εγγραφές.
Max χωρητικότητα Ευρετηρίου (Index):
2d - 1 = 3 κλειδιά.
2d = 4 δείκτες

Κανόνες Split (Διάσπασης):

1. Στα Φύλλα: Μοιράζουμε τις εγγραφές. Το πρώτο κλειδί του δεξιού κόμβου Αντιγράφεται (Copy Up) στον πατέρα.
2. Στους Εσωτερικούς Κόμβους: Το μεσαίο κλειδί Ανεβαίνει (Push Up) στον πατέρα και αφαιρείται από τον κόμβο.

Άσκηση 6: Τελική Μορφή του B+ Tree

Ακολουθώντας τους κανόνες διάσπασης, φτάνουμε σε αυτό το δέντρο.



Ας το φτιάξουμε βήμα βήμα.
https://www.cs.usfca.edu/~galles/visualization/BPlusTree.html

Άσκηση 7: Κατασκευή B+ Δέντρου

Έστω ότι παράμετροι: d=2, e=2.
Και ακολουθία εισαγωγής:
{10, 30, 5, 20, 40, 25, 15, 35, 50, 45, 18}

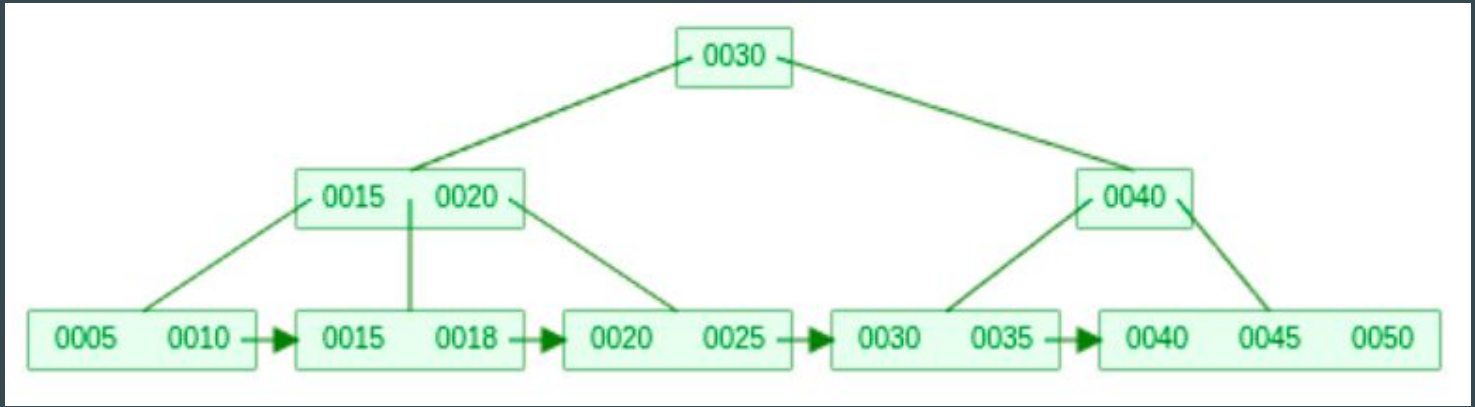
Άρα:
Max χωρητικότητα Φύλλων (Data):
2e - 1 = 3 εγγραφές.
Max χωρητικότητα Ευρετηρίου (Index):
2d - 1 = 3 κλειδιά.
2d = 4 δείκτες

Κανόνες Split (Διάσπασης):

1. Στα Φύλλα: Μοιράζουμε τις εγγραφές. Το πρώτο κλειδί του δεξιού κόμβου Αντιγράφεται (Copy Up) στον πατέρα.
2. Στους Εσωτερικούς Κόμβους: Το μεσαίο κλειδί Ανεβαίνει (Push Up) στον πατέρα και αφαιρείται από τον κόμβο.

Άσκηση 7: Τελική Μορφή του B+ Tree

Ακολουθώντας τους κανόνες διάσπασης, φτάνουμε σε αυτό το δέντρο.



Ας το φτιάξουμε βήμα βήμα.
https://www.cs.usfca.edu/~galles/visualization/BPlusTree.html

Προσοχή: Θεωρία vs. Online Εργαλεία

Ο Κανόνας (Underflow):

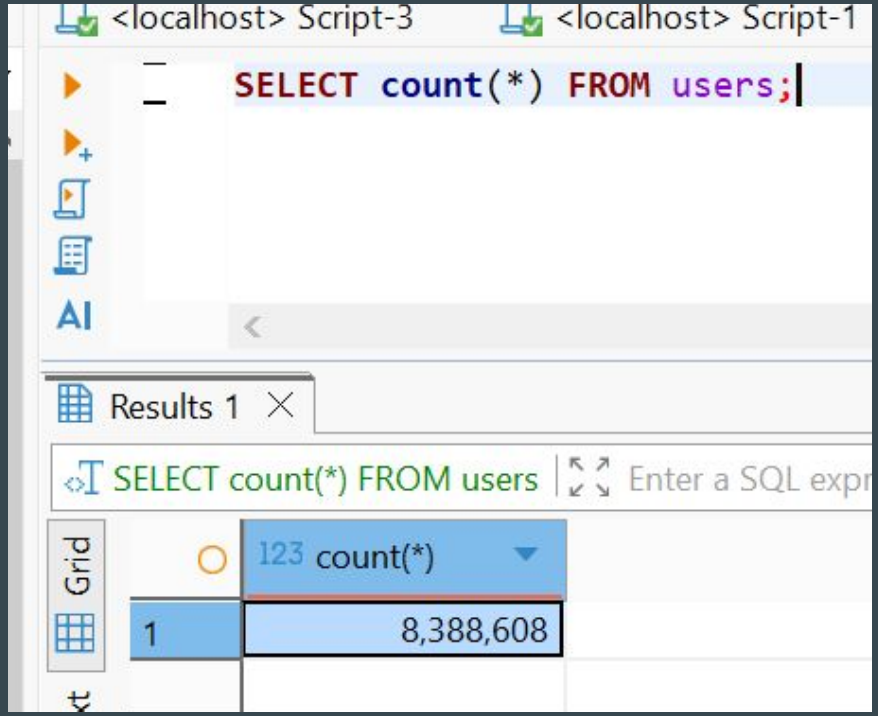
- Για τα Φύλλα (Data): Το όριο είναι το e.
- Για το Ευρετήριο (Index): Το όριο είναι το d (εκτός από τη Ρίζα).
- Αν μετά τη διαγραφή πέσουμε κάτω από το όριο → ΥΠΟΧΡΕΩΤΙΚΑ κάνουμε Δανεισμό (Redistribution) ή Συγχώνευση (Merge).

Η Παγίδα των Online Tools:

- Πολλά εργαλεία (όπως το BPlusTree.html που είδαμε) είναι πιο "χαλαρά" και δεν κάνουν αμέσως Merge.
- Συμπέρασμα: Χρησιμοποιούμε το tool για την Εισαγωγή, αλλά για τη Διαγραφή ακολουθούμε πιστά τις διαφάνειες του μαθήματος.

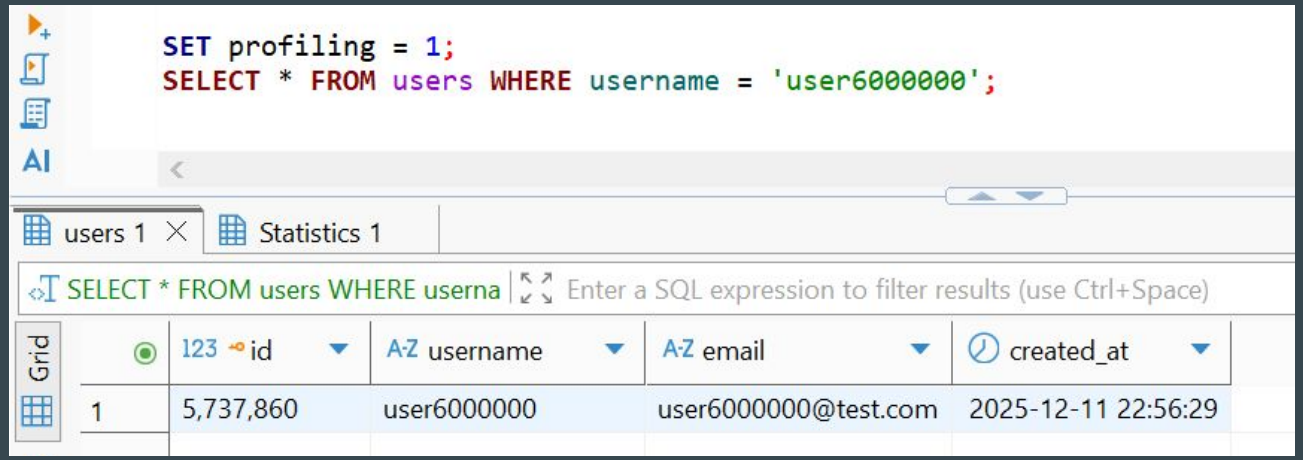
Ας δούμε ένα σωστό παράδειγμα Διαγραφής (Merge) απευθείας από τις διαλέξεις του μαθήματος.
https://www.csd.uoc.gr/~hy360/2025/lectures/Lecture17.pdf

Επισκόπηση του Dataset



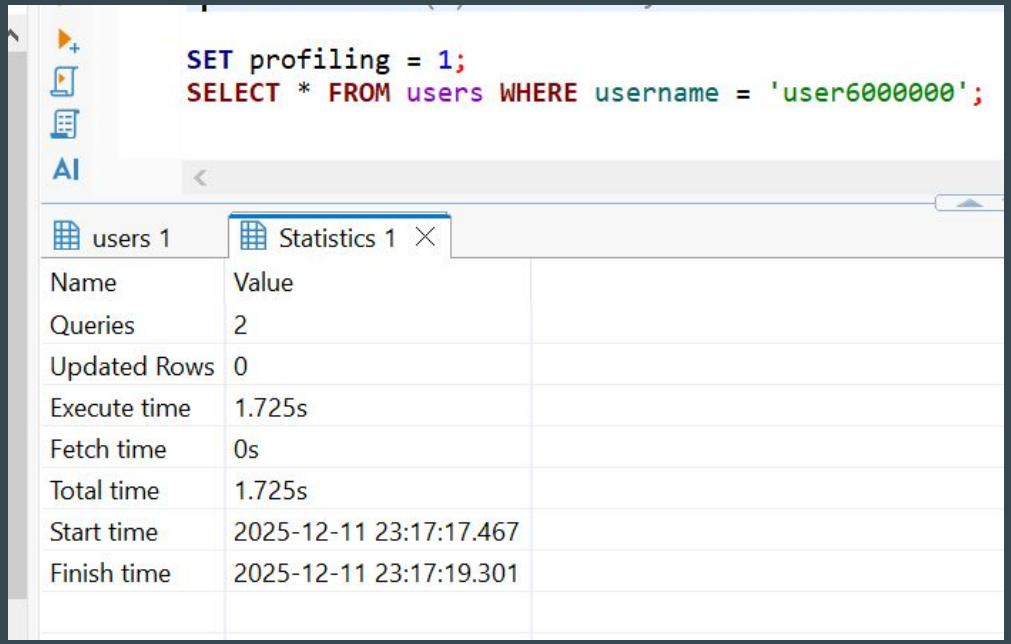
Επιβεβαίωση μεγέθους πίνακα users: Περιέχει 8.388.608 εγγραφές.

Αναζήτηση Χωρίς Ευρετήριο (Full Table Scan)



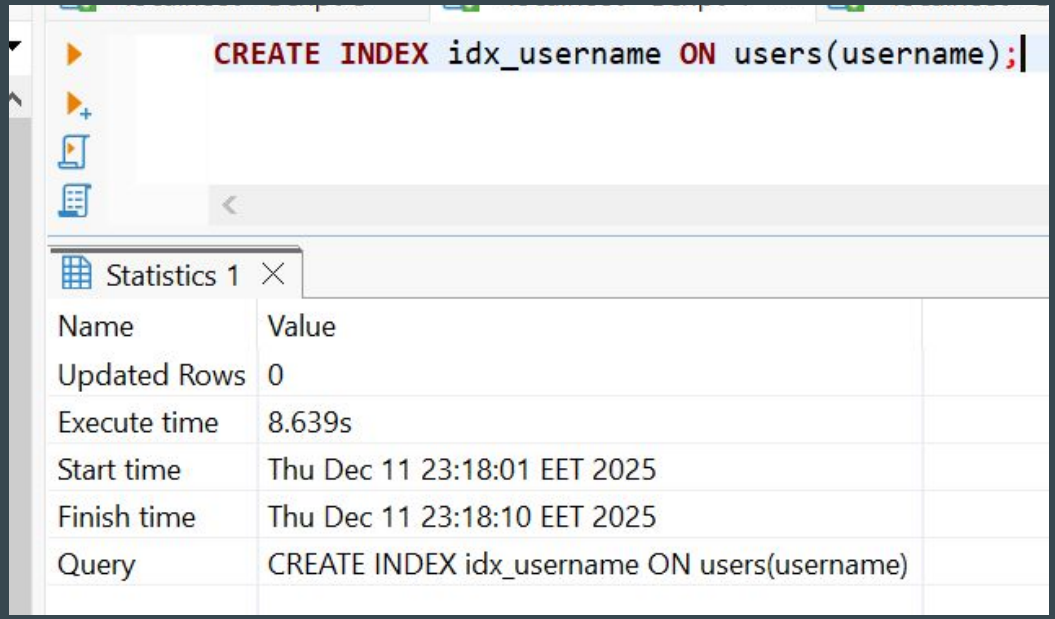
Εκτέλεση ερωτήματος (SELECT) με βάση το username.
Αναγκαστικά διαβάζονται σειριακά όλα τα blocks (Heap Scan).

Κόστος Σειριακής Αναζήτησης



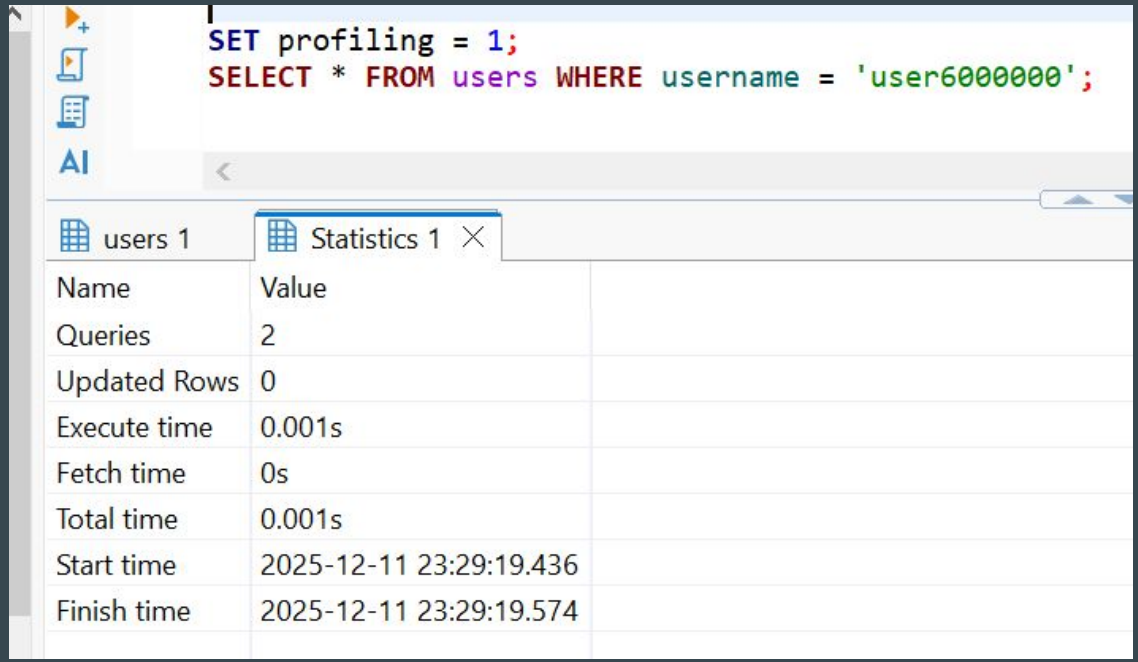
Ο χρόνος εκτέλεσης είναι **1.725 δευτερόλεπτα**. Για μία μόνο αναζήτηση, η καθυστέρηση είναι τεράστια λόγω του όγκου δεδομένων.

Δημιουργία B-Tree Index



Η εντολή CREATE INDEX κατασκευάζει τη δομή του δέντρου.
Χρειάστηκαν **8.6 δευτερόλεπτα**
(εφάπαξ κόστος κατασκευής και ταξινόμησης δεικτών).

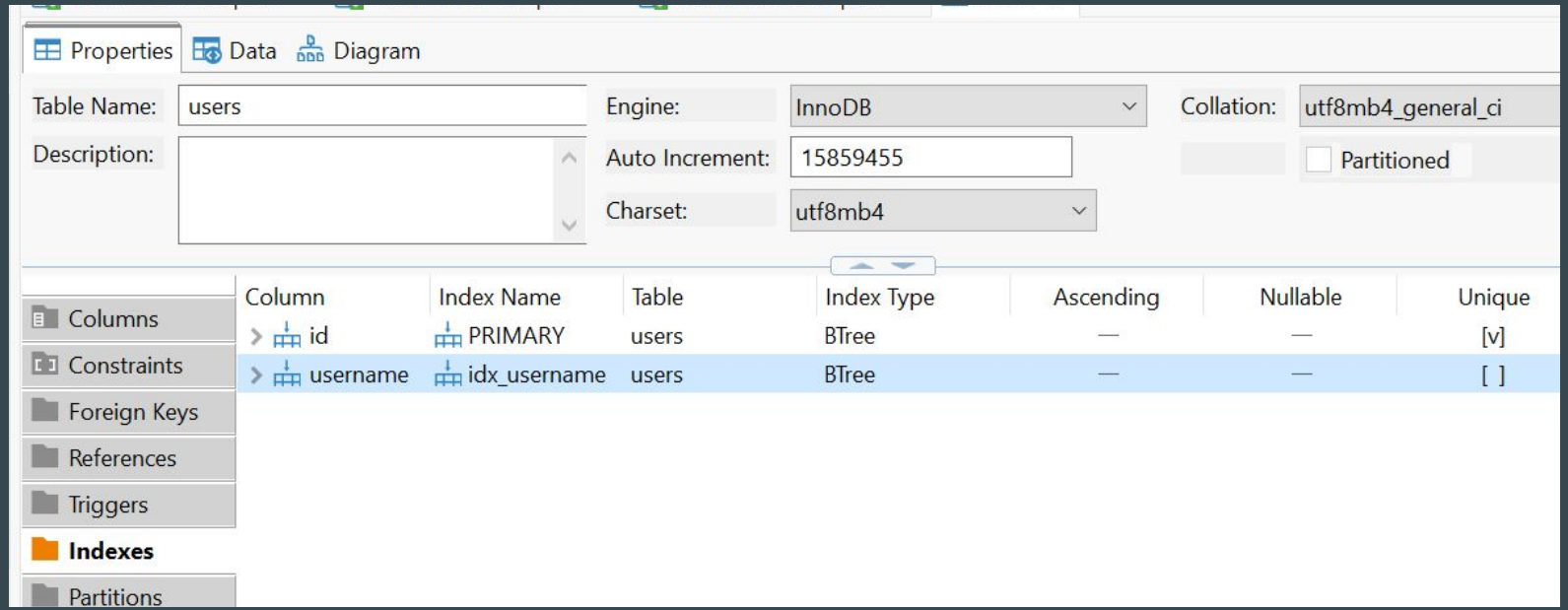
Αναζήτηση Με Ευρετήριο (Index Seek)



Δραματική βελτίωση απόδοσης. Ο χρόνος έπεσε στα 0.001 δευτερόλεπτα.

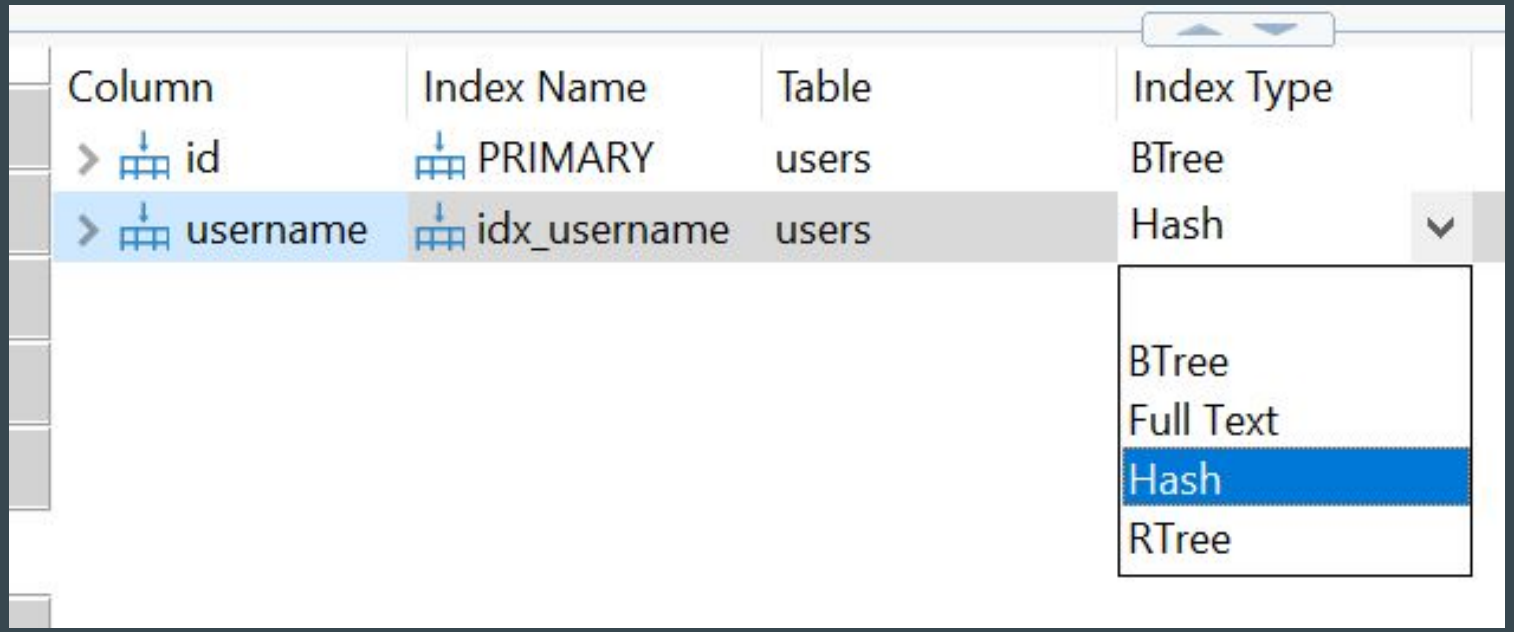
Χρησιμοποιήθηκε το δέντρο για να βρούμε άμεσα την εγγραφή.

Τα Indexes του Πίνακα Users



Βλέπουμε τα ενεργά ευρετήρια: Το Primary Key (φτιάχνεται αυτόματα) και το idx_username που δημιουργήσαμε. Και τα δύο είναι δομής **B-Tree** .

Εναλλακτικοί Τύποι Ευρετηρίων



Επιλογές αλγορίθμων: Το **Hash** είναι ιδανικό μόνο για ισοτιμίες (=), ενώ το **B-Tree** είναι η βέλτιστη επιλογή για γενική χρήση και εύρος τιμών (>, <).

Τέλος φροντιστηρίου

Ερωτήσεις;

ΗΥ - 360 Αρχεία και Βάσεις Δεδομένων
Φυσική Οργάνωση Δεδομένων & B-Trees (Μέρος 2ο)
Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης
(12/12/25)