# Βάζοντας τα όλα μαζί

## Άλγεβρα, Σχεδίαση Συστημάτων

Βάσεις Δεδομένων

## Στην σημερινή διάλεξη

1. Σχεσιακή Άλγεβρα και Βελτιστοποίηση
2. Παράδειγμα Σχεδίασης Συστημάτων

# Βελτιστοποίηση Ερωτημάτων

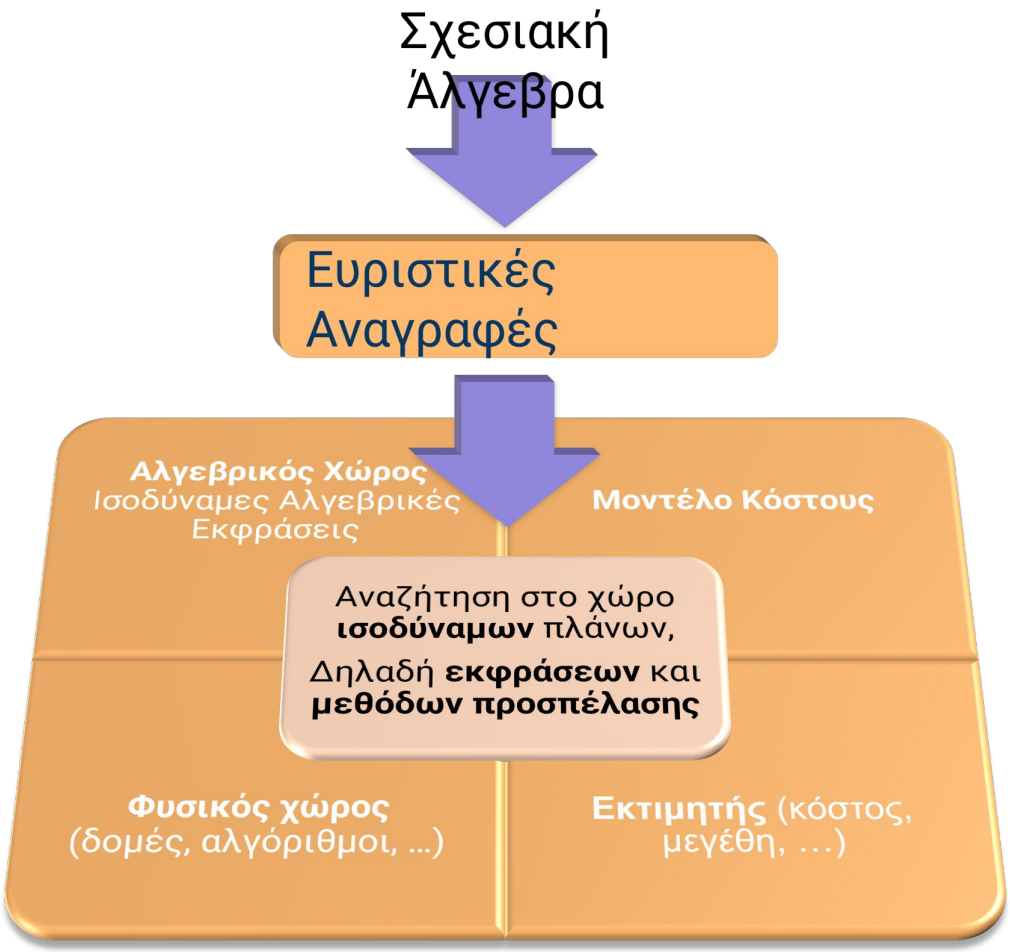
# Βασικοί μετασχηματισμοί ΣΑ(2)

- Αποφεύγω πάντα την εκτέλεση της πράξης του καρτεσιανού γινομένου.
  - Η μόνη περίπτωση που δεν μπορώ να το αποφύγω είναι όταν το απαιτεί το ίδιο το ερώτημα
  - Προσπαθώ να αντικαταστήσω το **καρτεσιανό γινόμενο** με πράξεις **σύζευξης** (⋈)

- Αλγεβρικός & Φυσικός Χώρος
- Πλάνο Εκτέλεσης
- Λογική Βελτιστοποίηση
- Φυσική Βελτιστοποίηση

# Αλγεβρικός & Φυσικός Χώρος

# Βελτιστοποιητής



# Αλγεβρικός χώρος

- Χώρος των ισοδύνα**μων αλγεβρικών εκφράσεων** που προκύπτει από τις **αλγεβρικές ιδιότητες των σχέσεων**

# Φυσικός Χώρος

- **Αλγόριθμοι** που έχουν **υλοποιηθεί** στο σύστημα για κάθε τελεστή και κάθε πιθανή **δομή** από όσες έχουν **υλοποιηθεί** στο σύστημα
- Ευρετήρια που υπάρχουν στο φυσικό σχήμα της βάσης και έχουν **δομές** από όσες έχουν **υλοποιηθεί** στο σύστημα
- Εξετάζουμε τον **Φυσικό Χώρο** για τις πράξεις της:
  - **Επιλογής**
  - **Συνένωσης**
  - **Προβολής**
  - **Συνολοθεωρητικές Πράξεις**
  - **Πράξη συνένωσης**

- Αλγεβρικός & Φυσικός Χώρος
- Πλάνο Εκτέλεσης
- Λογική Βελτιστοποίηση
- Φυσική Βελτιστοποίηση

# Πλάνο Εκτέλεσης

# Πλάνο Εκτέλεσης

- Ένα **πλάνο εκτέλεσης ερωτήματος** είναι η ακολουθία των βημάτων που εκτελούνται από ένα σύστημα διαχείρισης βάσεων δεδομένων προκειμένου να απαντηθεί ένα ερώτημα΄.
- Ένα πλάνο εκτέλεσης περιγράφεται από:
  1. μία ακολουθία τελεστών της σχεσιακής άλγεβρας (**αλγεβρικός χώρος**) μαζί με
  2. τον τρόπο εκτέλεσης του κάθε τελεστή (**φυσικός χώρος**)

# Αρχιτεκτονική ΣΔΣΒΔ (RDBMS)

Πως δουλεύει μια μηχανή SQL?

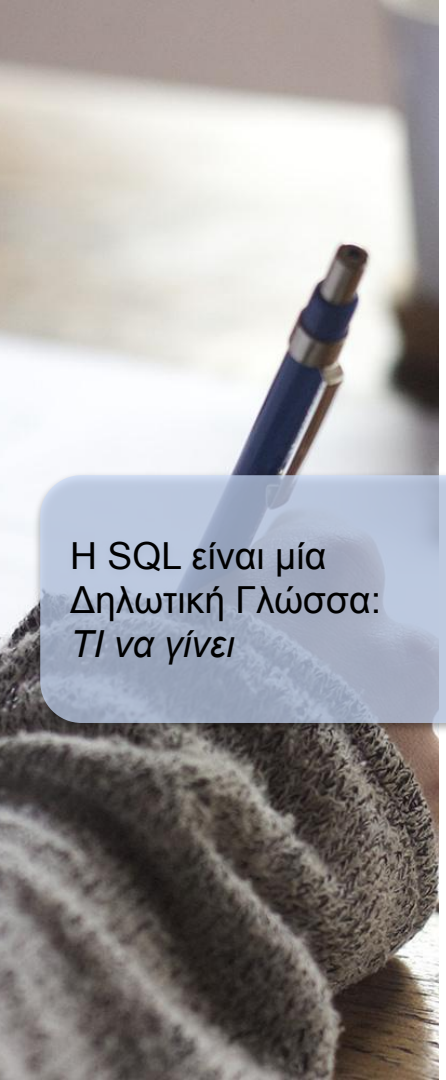
| Ερώτημα SQL | Πλάνο Σχεσιακής Άλγεβρας (ΣΑ) | Βελτιστοποιημένο Πλάνο Εκτέλεσης με Φυσικούς Τελεστές | Εκτέλεση |
|---|---|---|---|
| Δηλωτικό ερώτημα (από τον χρήστη) | Μετάφραση σε έκφραση σχεσιακής άλγεβρας | *Βρίσκουμε λογικά ισοδύναμη- αλλά πιο **αποδοτική ως προς το κόστος**- έκφραση ΣΑ με επιλογές εκτέλεσης φυσικών τελεστών* | Εκτελούμε κάθε τελεστή του βελτιστοποιημένου πλάνου! |

# Αρχιτεκτονική ΣΔΣΒΔ (RDBMS)

Πως δουλεύει μια μηχανή SQL?

| Ερώτημα SQL | Πλάνο Σχεσιακής Άλγεβρας (ΣΑ) | Βελτιστοποιημένο Πλάνο Εκτέλεσης με Φυσικούς Τελεστές | Εκτέλεση |
|---|---|---|---|
| Δηλωτικό ερώτημα (από τον χρήστη) | Μετάφραση σε έκφραση σχεσιακής άλγεβρας | *Βρίσκουμε λογικά ισοδύναμη- αλλά πιο* ***αποδοτική ως προς το κόστος****- έκφραση ΣΑ με επιλογές εκτέλεσης φυσικών τελεστών* | Εκτελούμε κάθε τελεστή του βελτιστοποιημένου πλάνου! |

Η Σχεσιακή Άλγεβρα μας επιτρέπει να μεταφράζουμε δηλωτικά (SQL) ερωτήματα σε ακριβείς και βελτιστοποιημένες εκφράσεις!



# Μετατρέποντας τα ερωτήματα SFW σε ΣΑ

Students(sid,sname,gpa)
People(ssn,sname,address)

Η SQL είναι μία Δηλωτική Γλώσσα:
*ΤΙ να γίνει*

```
SELECT DISTINCT
        gpa,
        address
FROM    Students S,
        People P
WHERE   gpa > 3.5 AND
        sname = pname;
```

$$\Pi_{gpa,address}(\sigma_{gpa>3.5}(S \bowtie P))$$

Η σχεσιακή Άλγεβρα είναι μία Διαδικαστική Γλώσσα:
*ΤΙ και ΠΩΣ να γίνει*

Πως αναπαριστούμε αυτό το ερώτημα σε ΣΑ?

# Λογικά ισοδύναμα των πλάνων ΣΑ

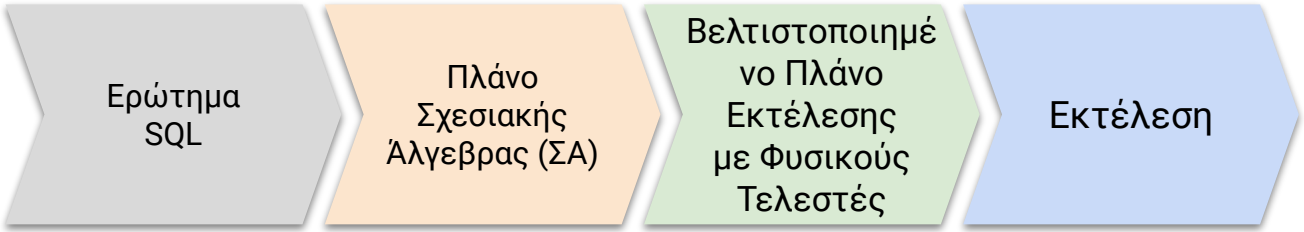
Δεδομένων σχέσεων R(A,B) και S(B,C):

- Εδώ, η προβολή και η επιλογή εναλλάσσονται:
  - $\sigma_{A=5}(\Pi_A(R)) = \Pi_A(\sigma_{A=5}(R))$
- Και εδώ;
  - $\Pi_B(\sigma_{A=5}(R)) \stackrel{?}{=} \sigma_{A=5}(\Pi_B(R))$

Δεν μπορεί εκτελεστεί το δεύτερο πλάνο γιατί πρώτα γίνεται προβολή στο γνώρισμα Β και μετά επιλογή στο Α, το οποίο όμως δεν υπάρχει πλέον λόγω της προβολής.

# Αρχιτεκτονική ΣΔΣΒΔ

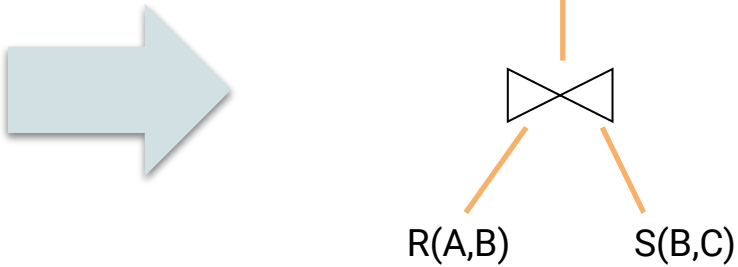
Πως δουλεύει μια μηχανή SQL?




Θα δούμε πως βελτιστοποιούνται αυτά τα πλάνα τώρα!

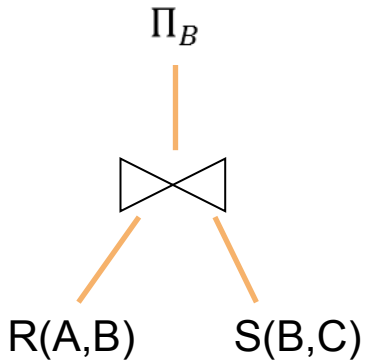
# Απεικονίζουμε το πλάνο σαν δέντρο

$\Pi_B(R(A,B) \bowtie S(B,C))$




Από κάτω προς τα πάνω διάσχιση δέντρου = ταξινόμηση της εκτέλεσης των τελεστών!

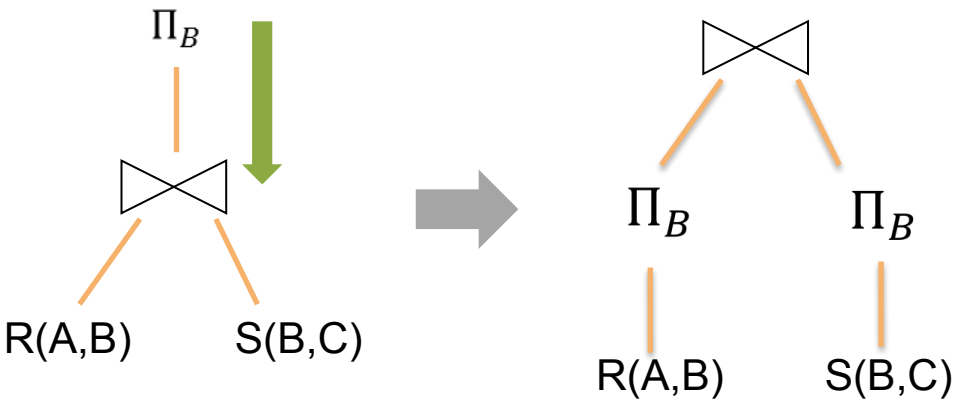
# Ένα απλό πλάνο




Σε ποια SQL ερωτήματα ανταποκρίνεται αυτό?

Υπάρχουν λογικές ισοδύναμες εκφράσεις ΣΑ?

# “Σπρώξιμο προς τα κάτω” προβολή




Γιατί μπορεί να προτιμήσουμε αυτό το πλάνο?

- Αλγεβρικός & Φυσικός Χώρος
- Πλάνο Εκτέλεσης
- Λογική Βελτιστοποίηση
- Φυσική Βελτιστοποίηση

# Λογική Βελτιστοποίηση

# Λογική Βελτιστοποίηση

- Θέλουμε οι επιλογές και οι προβολές να συμβαίνουν όσο το δυνατό νωρίτερα στο πλάνο
  - Ορολογία: “σπρώχνουμε προς τα κάτω **επιλογές** και **προβολές**”
- Διαίσθηση: Συνήθως θα έχουμε λιγότερες πλειάδες στο πλάνο.
- Εξαιρέσεις
  - Μπορεί να αποτύχει αν η **συνθήκη επιλογής** είναι πολύ ακριβή (π.χ, να τρέξουμε αλγορίθμους επεξεργασίας εικόνας)
  - Η **προβολή** μπορεί να είναι σπατάλη προσπάθειας, αλλά πιο σπάνια

# Βήμα 1: Μεταφράζοντας σε ΣΑ

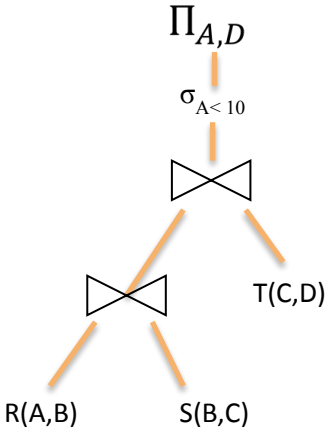
R(A,B) S(B,C) T(C,D)

```
SELECT    R.A,S.D
FROM      R,S,T
WHERE     R.B = S.B
      AND S.C = T.C
      AND R.A < 10;
```

Μετάφραση σε Σχεσιακή Άλγεβρα

$$\Pi_{A,D}(\sigma_{A<10}(T \bowtie (R \bowtie S)))$$

Δενδρικό Πλάνο



# Βήμα 2: Βελτιστοποιώντας το πλάνο ΣΑ

R(A,B) S(B,C) T(C,D)

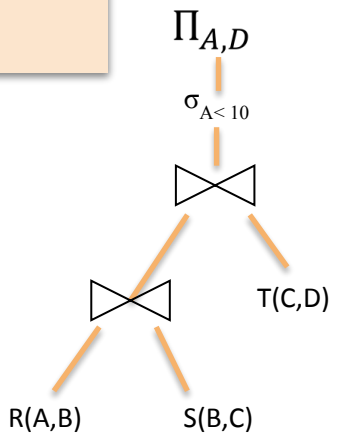
```
SELECT    R.A,S.D
FROM      R,S,T
WHERE     R.B = S.B
      AND S.C = T.C
      AND R.A < 10;
```

Σπρώχνουμε προς τα κάτω την επιλογή στο Α ώστε να συμβεί νωρίτερα

Μετάφραση σε Σχεσιακή Άλγεβρα

$$\Pi_{A,D}(\sigma_{A<10}(T \bowtie (R \bowtie S)))$$

Δενδρικό Πλάνο



# Βήμα 2: Βελτιστοποιώντας το πλάνο ΣΑ

R(A,B) S(B,C) T(C,D)

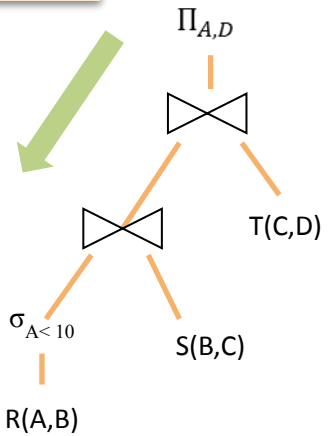
```
SELECT    R.A,S.D
FROM      R,S,T
WHERE     R.B = S.B
      AND S.C = T.C
      AND R.A < 10;
```

Σπρώχνουμε προς τα κάτω την επιλογή στο Α ώστε να συμβεί νωρίτερα

Μετάφραση σε Σχεσιακή Άλγεβρα

$$\Pi_{A,D}(\sigma_{A<10}(T \bowtie (R \bowtie S)))$$

Δενδρικό Πλάνο



# Βήμα 2: Βελτιστοποιώντας το πλάνο ΣΑ

R(A,B) S(B,C) T(C,D)

```
SELECT    R.A,S.D
FROM      R,S,T
WHERE     R.B = S.B
      AND S.C = T.C
      AND R.A < 10;
```

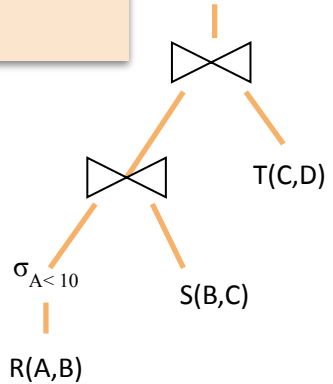
Εξαλείφουμε το Β νωρίτερα!

Γενικά, πότε είναι ένα χαρακτηριστικό μη απαραίτητο…?

Μετάφραση σε Σχεσιακή Άλγεβρα

$$\Pi_{A,D}(\sigma_{A<10}(T \bowtie (R \bowtie S)))$$

Δενδρικό Πλάνο



# Αφαιρέσεις (Takeaways)

- Αυτή η διαδικασία ονομάζεται λογική βελτιστοποίηση
- Πολλά ισοδύναμα πλάνα χρησιμοποιούνται για να ψάξουν για  “καλά πλάνα”
- Η σχεσιακή άλγεβρα είναι μια απλή και κομψή αφαίρεση

# Βασικοί μετασχηματισμοί ΣΑ

- Σπρώχνουμε την **προβολή** μέσω **(1) επιλογής**, **(2) σύζευξης**
- Σπρώχνουμε την **επιλογή** μέσω **(3) επιλογής**, **(4) προβολής**, **(5) σύζευξης**
- Επίσης: Οι συζεύξεις μπορούν να **αναδιαταχθούν**!
- Επίσης: Επιλέγεται ο **κατάλληλος αλγόριθμος για σύζευξη**.

⇒ Σημειώστε ότι αυτό **δεν είναι ένα εξαντλητικό σύνολο μετασχηματισμών**. Το σύνολο μετασχηματισμών καλύπτει τις τοπικές επανεγγραφές; τις καθολικές επανεγγραφές. Υπάρχουν πολύ πιο δυσκολότεροι μετασχηματισμοί

Αυτό το απλό σύνολο από εργαλεία μας επιτρέπει να βελτιώσουμε σημαντικά τον χρόνο εκτέλεσης των ερωτημάτων βελτιστοποιώντας τα πλάνα ΣΑ!

# Παράδειγμα

Έστω ότι έχω

- $(\mathrm{R} \times \mathrm{T}) \bowtie_{\substack{R.A=S.A \wedge \\ T.B=S.B}} S = R \bowtie_{R.A=S.A} (T \bowtie_{T.B=S.B} S) =$
  $= (R \bowtie_{R.A=S.A} S) \bowtie_{T.B=S.B} T$
- Από τις τρεις εναλλακτικές αποφεύγω αυτήν με το καρτεσιανό γινόμενο

- Αλγεβρικός & Φυσικός Χώρος
- Πλάνο Εκτέλεσης
- Λογική Βελτιστοποίηση
- Φυσική Βελτιστοποίηση

# Φυσική Βελτιστοποίηση

# Φυσική Βελτιστοποίηση

Η φυσική βελτιστοποίηση ενός λογικού πλάνου εκτέλεσης συνίσταται:

- Επιλογή των κατάλληλων αλγορίθμων εκτέλεσης των τελεστών σχεσιακής άλγεβρας
  - Για μία πράξη σύζευξης ⋈ τι αλγόριθμο να χρησιμοποιήσω: Block Nested Loop Join, Index Nested Loop Join, Sort-Merge Join, Hash Partition Join;
  - Αντίστοιχα και για άλλες πράξεις όπως ∪ (ένωση), ∩ (τομή), δ (Distinct), γ (Ομαδοποίηση), τ (ταξινόμηση)

# Φυσική Βελτιστοποίηση (2)

- Σχετικά ερωτήματα:
  - Έχω κάποια **ταξινόμηση** στα δεδομένα μου από προηγούμενες πράξεις;
  - Έχω κάποιο **κατακερματισμό** στα δεδομένα μου;
  - Υπάρχει κάποιο ευρετήριο που μπορεί να χρησιμοποιηθεί για τις πράξεις $\bowtie$, γ, $\cup$, σ (επιλογή)
  - Ποια είναι τα **εκτιμώμενα κόστη** για κάθε διαφορετικό τρόπο εκτέλεσης; (Συμβουλεύομαι κόστη που μελετήσαμε στα προηγούμενα κεφάλαια)

# Φυσική Βελτιστοποίηση (3)

- Μπορώ να έχω χρήση **σωληνώσεων**, δηλαδή παράλληλη συγχρονισμένη εκτέλεση σειράς πολλαπλών τελεστών όπου ο ένας καταναλώνει τα αποτελέσματα του άλλου
  - **Απαλλαγή από ανάγκη γραψίματος/ξαναδιαβάσματος ενδιάμεσων αποτελεσμάτων**
  - **Μικρότερος χρόνος απόκρισης**. Για να εμφανίσουμε το 1ο αποτέλεσμα δεν χρειάζεται να έχουμε τα υπόλοιπα

# Σχεδιασμός Συστημάτων Δεδομένων

# Παράδειγμα Σχεδιασμού Συστημάτων:

# Συνύπαρξη προϊόντων

## Μετρώντας προβολές προϊόντος για δις προϊόντων

Nespresso Vertuo Coffee and Espresso Machine Bundle with Aeroccino Milk Frother by Breville, Red

by Breville

980 customer reviews | 259 answered questions

Amazon's Choice for "nespresso machine red"

List Price: $249.95

Price: $189.96 prime | FREE One-Day

You Save: $59.99 (24%)

Your cost could be $179.96. Eligible customers get a $10 bonus when reloading $100.

Free Amazon product support included

Style Name: Nespresso by Breville

Nespresso | Nespresso by Breville

Color: Red

## Μετρώντας δημοφιλή προϊόντα-ζεύγη

Customers who viewed this item also viewed these products

Dualit Food XL1500 Processor

$560

Add to cart

Kenwood kMix Manual Espresso Machine

$250

Select options

Weber One Touch Gold Premium Charcoal Grill-57cm

$225

Add to cart

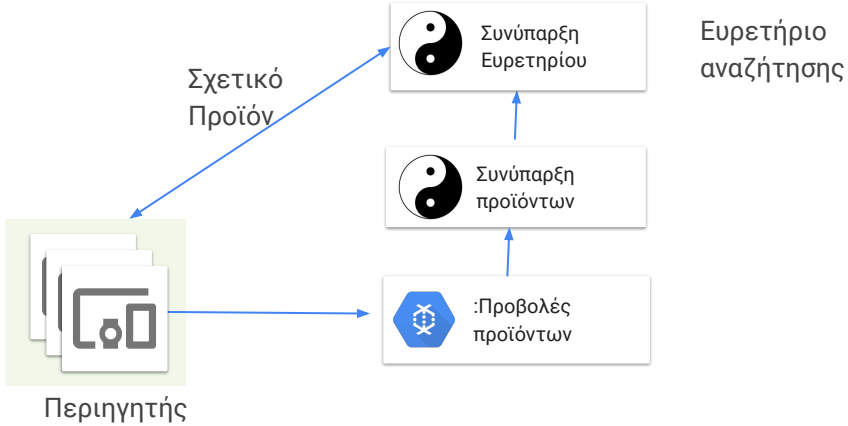
NoMU Salt Pepper and Spice Grinders

$3

View options

# Παράδειγμα Σχεδιασμού Συστημάτων:

# Συνύπαρξη προϊόντων




## Μοτίβο σχεδιασμού Δημοφιλών Συστημάτων

- Αποδοτικά υπολόγισε “παρτίδες” δεδομένων
- Χτίσε Ευρετήριο Αναζήτησης στα αποτελέσματα
- Για ‘streaming’ δεδομένα, ενημέρωσε με ‘μικρο-παρτίδες’

# Βασικά νούμερα

## Cheatsheet

- **Ταχύτητες δίσκου**: 10 msec/αναζήτηση, 100 MBs/sec
- Τυπικές υποθέσεις μηχανήματος (εκτός εάν το πρόβλημα δηλώνει αλλιώς):
  - 64 GB RAM, 64 KB (για block δίσκου, μέγεθος σελίδας RAM)
  - Ακέραιοι 4 byte , 8 byte μεγάλοι ακέραιοι, δείκτες 8 bytes
- 2^10 = 1024, 2^20 ~= 1 Εκατομμύριο, 2^30 ~= 1 Δις (10^9), 2^40 ~= 1 Τρις (10^12)
  - Για αποθήκευση εγγραφών (4 bytes καθεμία): 1 Εκ. εγγραφές= 4MB, 1 Δις εγγραφές = 4 GB

# Ευρετήριο Β+ δέντρου (Επανάληψη)

# Αναζήτηση σε B+ Δέντρο Ευρετήριο [ανακεφαλαίωση]

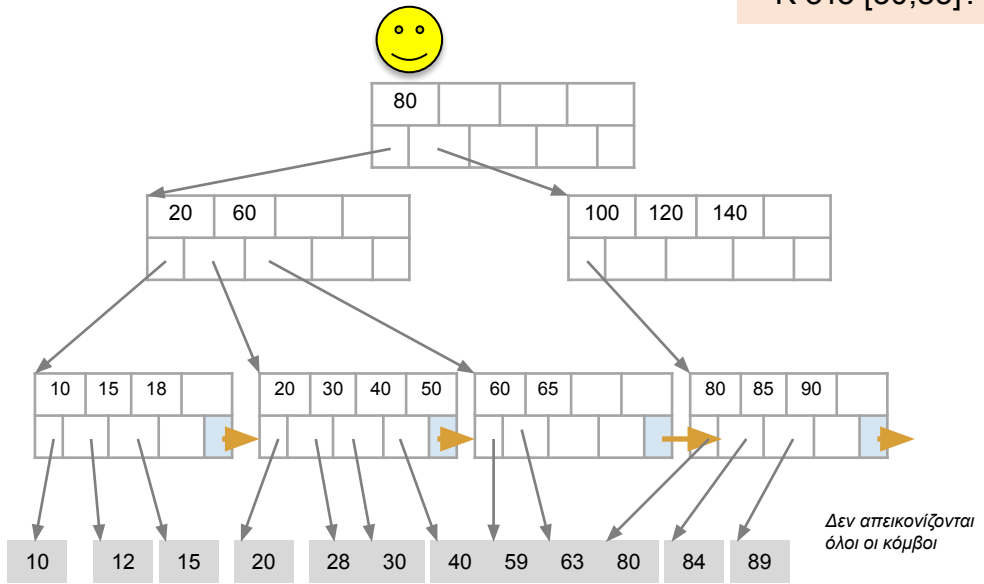
K στο [30,85]?

30 < 80

30 στο [20,60)

30 στο [30,40)

Στα δεδομένα!



# Απλό Μοντέλο Κόστους για Αναζήτηση
## [ανακεφαλαίωση]

- Έστω:
  - ***f*** = διακλάδωση, η οποία είναι **στο [d+1, 2d+1]** ***(υποθέτουμε ότι είναι σταθερό για το δικό μας μοντέλο κόστους…)***
  - ***N*** = συνολικός αριθμός *σελίδων* που χρειάζεται να ευρετηριοποιήσουμε
  - ***F*** = παράγοντας-πληρότητας (συνήθως ~= 2/3)

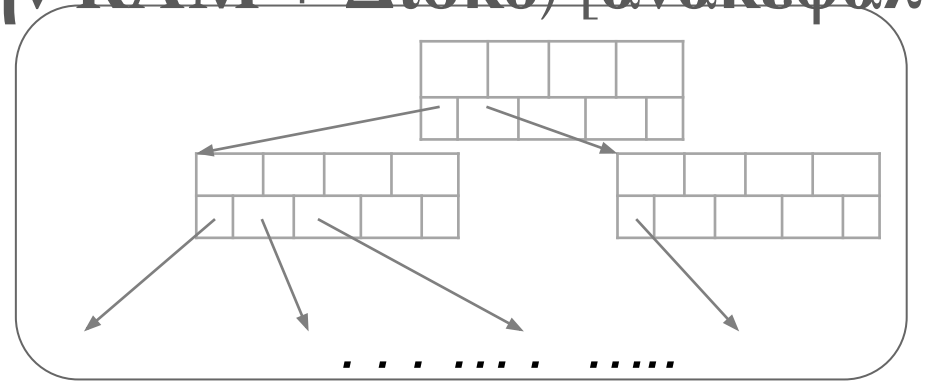
- Το B+ Δέντρο μας χρειάζεται να έχει χώρο για να ευρετηριοποιήσει ***N / F*** σελίδες!
  - Έχουμε τον παράγοντα-πληρότητας ώστε να αφήσουμε μερικές ανοιχτές θέσεις για γρηγορότερες εισαγωγές

- Τι ύψος (*h*) χρειάζεται να έχει το B+ Δέντρο μας;
  - h=1 → Μόνο η ρίζα- χώρος για να ευρετηριοποιηθούν f σελίδες
  - h=2 → f φύλλα- χώρος για να ευρετηριοποιηθούν $f^2$ σελίδες
  - h=3 → $f^2$ φύλλα- χώρος για να ευρετηριοποιηθούν $f^3$ σελίδες
  - …
  - h → $f^{h-1}$ φύλλα- χώρος για να ευρετηριοποιηθούν $f^h$ σελίδες!

→Χρειαζόμαστε ένα B+ Δέντρο ύψους $h = \left\lceil \log_f \frac{N}{F} \right\rceil$!

# Κόστος Αναζήτησης ενός B+ Δέντρου (στην RAM + Δίσκο) [ανακεφαλαίωση]

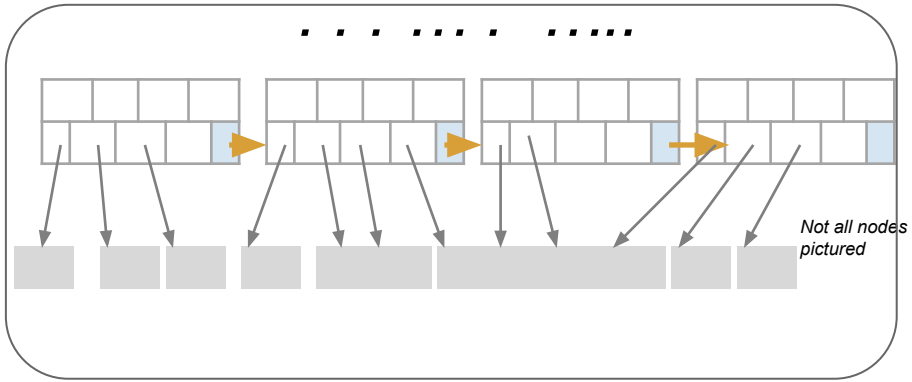
Διαβάζουμε τα πρώτα επίπεδα στον RAM buffer



1+ f + $f^2$ + $f^3$ +... <= B

Κρατάμε τα 1st $L_B$ επίπεδα σε RAM μεγέθους B

Το υπόλοιπο του ευρετηρίου στον δίσκο




Αλγόριθμος: B+ Αναζήτηση
- Διαβάζουμε 1 σελίδα ανά επίπεδο
- Οι σελίδες στην RAM δεν έχουν κόστος
- Διαβάζουμε 1 σελίδα για εγγραφή

Κόστος IO: $\left\lceil \log_f \frac{N}{F} \right\rceil - L_B + 1$

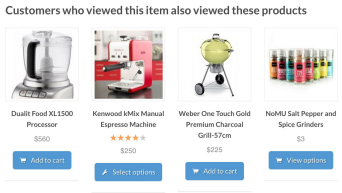
Όπου $B \geq \sum_{l=0}^{L_B - 1} f^l$

# Παράδειγμα Σχεδιασμού Συστημάτων:

# Συνύπαρξη προϊόντων

Μετρώντας δημοφιλή προϊόντα-ζεύγη

(from primer doc)




1. Amazon/Walmart/Alibaba (AWA) need to compute ‘related products’ for all products so their users can explore and buy new products. AWA would like to use collaborative filtering (aka ‘wisdom of crowds’) from their website logs of user views of products. That is, each time a user views a set of products, those products are said to co-occur. By computing product pairs and their co-occurrence frequency (or **co-occur count)** across all users, AWA can compute related products for their catalog.
2. Data input: AWA’s product catalog is 1 billion items. Each product record is ~1MB with a product description and image. AWA has 10 billion product views each week, from 1 billion users. Each log record stores <userID, productID, viewID, viewtime>.
3. Your mission is to design an efficient system to compute co-occur counts on *Sundays* from weekly logs and produce a CoOccurCount table <productID, productID, count>
    - AWA’s data quality magicians recommend (a) retaining only the top billion popular pairs, and (b) dropping product pairs with co-occur counts less than million. Also, assume users view ten products on average (**UserSession** assumption).
    - For simplicity, LogOfViews is stored sorted by <userID, productID>. You can sequentially scan the log and produce co-occurring product pairs for each user. In other words, $(p_i, p_j)$ if a user viewed $p_i$ and $p_j$. This “stream” of tuples (TempCoOccur) may then be (a) stored on disk or (b) discarded after updating any data structures.

# Παράδειγμα Σχεδιασμού Συστημάτων:

# Συνύπαρξη προϊόντων

# Προσχεδιασμός

| | Μέγεθος | Γιατί; |
|---|---|---|
| ProductId | 4 bytes | 1 δις προϊόντα⇒ Χρειάζονται τουλ. 30 bits (2^30 ~= 1 δις) για αναπαράσταση κάθε προϊόντος ξεχωριστά. Οπότε χρηση 4 bytes. |
| UserID | 4 bytes | “ |
| LogOfViewsID | 8 bytes | 10 Δις προβολές προϊόντων. |
| Product | 1 PB | 1 Δις προϊόντα του 1 MB το καθένα |
| Users | Άγνωστο | |
| LogOfViews | 240 GB | Κάθε εγγραφή είναι <userID, productID, viewID, viewtime>. Υπόθεση: χρησιμοποιούμε 8 bytes για viewTime. Οπότε 24 bytes ανά εγγραφή. 10 Δις*24 bytes = 240 GBs. |
| CoOccur | 12 GB | Η έξοδος πρέπει να είναι <productID, productID, count> για την μέτρηση συνύπαρξης. Αυτά είναι, 12 bytes ανά εγγραφή (4 + 4 + 4 για τα 2 productIDs και 4 bytes για μετρητή). Για διατήρηση του κορυφαίου δις προϊόντων-ζευγών (όπως προτείνεται από την AWA ποιότητα δεδομένων), χρειάζεσαι 1 δις * 12 bytes = 12 GBs. |
| TempCoOccur | 10^9 * 12 GB | Για μέτρηση όλων των προϊόντων-ζευγών όπως σκανάρουμε την είσοδο, ίσως χρειαστούμε 1 δις*1 δις (10^18) μετρητές (10^9 * 12 GB αποθηκευτικού χώρου) |
| TempCoOccur (με υπόθεση UserSession, των ~10 προβολών/χρήστη) | 800 GB | # προϊόντων-ζευγών που παρήχθησαν: 1 δις χρήστες * 10^2 = 100 δις<br>Μέγεθος @8 bytes/εγγραφή = 800 GBs. |

# Παράδειγμα Σχεδιασμού Συστημάτων:

## Συνύπαρξη προϊόντων

## Διαχείριση RAM/Δίσκου

| | Κράτησε τον πίνακα στην RAM; Μέγεθος; | Σειριακά σάρωσε από τον δίσκο; Αν ναι, πόσες σελίδες δίσκου; |
|---|---|---|
| Products | | |
| Users | | |
| LogOfViews | | |
| CoOccur | | |
| TempCoOccur | | |
| TempCoOccur (με υπόθεση UserSession) | | |

# Παράδειγμα Σχεδιασμού Συστημάτων:

## Συνύπαρξη προϊόντων

## Διαχείριση RAM/Δίσκου

| | Κράτησε τον πίνακα στην RAM; Μέγεθος; | Σειριακά σάρωσε από τον δίσκο; Αν ναι, πόσες σελίδες δίσκου; |
|---|---|---|
| Products | Δεν χρειάζεται στο πρόβλημα | |
| Users | Δεν χρειάζεται στο πρόβλημα | |
| LogOfViews | Όχι, 240 GB<br><br>Χρειάζεται να σαρώσεις τις ανά-χρήστη εγγραφές μόνο μία φορά. Καμία ανάγκη για τυχαία προσπέλαση. | |
| CoOccur | Ναι, 12 GB. Προτίμησε τυχαία προσπέλαση. | |
| TempCoOccur | Όχι.<br><br>Χειρότερη περίπτωση:<br>1 δις * 1 δις μετρητές.<br>Μέγεθος= 10^18*12 bytes | |
| TempCoOccur (με υπόθεση UserSession) | Όχι.<br><br>800 GBs | |

# Παράδειγμα Σχεδιασμού Συστημάτων:

## Συνύπαρξη προϊόντων

## Διαχείριση RAM/Δίσκου

| | Διατήρηση του πίνακα στην RAM; Μέγεθος; | Σειριακά σάρωσε από τον δίσκο; Αν ναι, πόσες σελίδες δίσκου; |
|---|---|---|
| Products | Δεν χρειάζεται στο πρόβλημα | |
| Users | Δεν χρειάζεται στο πρόβλημα | |
| LogOfViews | Όχι, 240 GB<br><br>Χρειάζεται να σαρώσεις τις ανά-χρήστη εγγραφές μόνο μία φορά. Καμία ανάγκη για τυχαία προσπέλαση. | Ναι.<br><br>~4 εκ. blocks δίσκου (υπενθύμιση: 64KB ανά σελίδα, block δίσκου) |
| CoOccur | Ναι, 12 GB. Προτίμησε τυχαία προσπέλαση. | Όχι, διατήρηση στην RAM. (Γράψε αργότερα στον δίσκο, αν είναι απαραίτητο.) |
| TempCoOccur | Όχι.<br><br>Χειρότερη περίπτωση:<br>1 δις * 1 δις μετρητές.<br>Μέγεθος= 10^18*12 bytes | Ναι, πρέπει να μείνει στον δίσκο.<br><br>Χειρότερη περίπτωση: 12 * 10^18/64KB σελίδες<br>(= 18.75 * 10^13 σελίδες) |
| TempCoOccur (με υπόθεση UserSession) | Όχι.<br><br>800 GBs | Ναι, πρέπει να μείνει στον δίσκο.<br><br># σελίδες: 800 GBs/64 KB ~= 12.5 εκ. |

# Παράδειγμα Σχεδιασμού Συστημάτων:

# Συνύπαρξη προϊόντων

# Σχεδιασμός #2

Σχεδιασμός #2: Με 1 μηχάνημα, Ανάλυσε με υπόθεση για το UserSession.

Σχεδιασμός 2

1. Σάρωσε τα LogOfviews. Για κάθε χρήστη, πρόσθεσε το <p_i, p_j> σε ένα log TempCoOccurLog αν ο χρήστης έχει δει το προϊόν p_i και p_j. (δηλαδή, δημιούργησε ανά χρήστη ζεύγη συνυπάρχοντων προϊόντων)
2. Ταξινόμησε εξωτερικά το TempCoOccurLog στον δίσκο, οπότε πανομοιότυπα ζεύγη προϊόντων είναι διπλανά το ένα στο άλλο στο ταξινομημένο αρχείο
3. Σάρωσε το ταξινομημένο TempCoOccurLog. Αυτό με μονό πέρασμα, μπορείς να μετρήσεις ζεύγη συνύπαρξης. Άφησε ζεύγη συνύπαρξης με < 1 εκ.

# Παράδειγμα Σχεδιασμού Συστημάτων:

# Συνύπαρξη προϊόντων

# Σχεδιασμός #2

Σχεδιασμός #2: Με 1 μηχάνημα, Ανάλυσε με υπόθεση για το UserSession.

| Βήματα | Κόστος (χρόνος) | Γιατί; |
|---|---|---|
| Σάρωσε το LogOfViews<br><br>Πρόσθεσε το <p_i, p_j> στο TempCoOccurLog | | |
| Ταξινόμησε εξωτερικά το TempCoOccurLog στον δίσκο<br><br>(Υποθέτουμε κόστος ταξινόμησης ~2N, όπου N ο αριθμός σελίδων για τον πίνακα και B ο αριθμός των buffers, και B ~~ N) | | |
| Σάρωσε το TempCoOccurLog (ταξινομημένο) και διατήρησε μετρητές στο CoOccur | | |

# Παράδειγμα Σχεδιασμού Συστημάτων:

# Συνύπαρξη προϊόντων

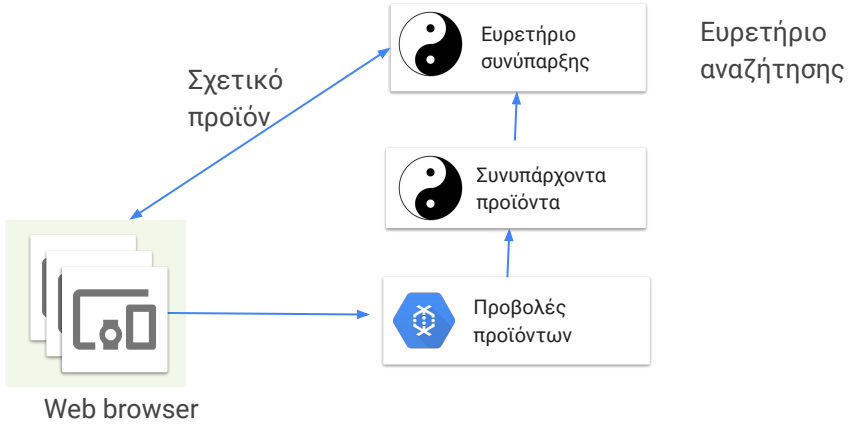
# Σχεδιασμός #2

Σχεδιασμός #2: Με 1 μηχάνημα, Ανάλυσε με υπόθεση για το UserSession.

| Βήματα | Κόστος (χρόνος) | Γιατί; |
|---|---|---|
| Σάρωσε το LogOfViews<br><br>Πρόσθεσε το <p_i, p_j> στο TempCoOccurLog | ~2400 secs<br><br>~8000 secs | 240GB @100 MB/sec<br><br>800 GB @100 MB/sec |
| Ταξινόμησε εξωτερικά το TempCoOccurLog στον δίσκο<br><br>(Υποθέτουμε κόστος ταξινόμησης ~2N, όπου N ο αριθμός σελίδων για τον πίνακα και B ο αριθμός των buffers, και B ~~ N) | ~16,000 secs | Το κόστος του IO είναι (περίπου) 2 * (1 αναζήτηση + κόστος σάρωσης για 12.5 εκ. Σελίδες * 64 KB/ανά σελίδα) = 2* κόστος σαρωσης 800 GBs. Αυτό είναι, 16000 δευτ. (2*800 GB @100 MB/sec).<br><br>Υποθέτουμε ότι τα TempCoOccurLog (και οι εκτελέσεις) αποθηκεύονται σειριακά. |
| Σάρωσε το TempCoOccurLog (ταξινομημένο) και διατήρησε μετρητές στο CoOccur | ~8000 δευτ. | 800 GB @100 MB/sec |

# Παράδειγμα Σχεδιασμού Συστημάτων:

# Συνύπαρξη προϊόντων



# Παράδειγμα Σχεδιασμού Συστημάτων:

## Συνύπαρξη προϊόντων

## Ευρετήριο $B^+$ δέντρου

Εκτίμησε τα κόστη των αναζητήσεων σε ένα ομαδοποιημένο B+ δέντρο, ομαδοποιημένο στο productId που ψάχνουμε. Πόσες IO αναζητήσεις μπορούμε να αναμένουμε αν είχαμε 1 GB RAM για το ευρετήριο;

Υπενθύμιση: Έστω ‘β’ ο βαθμός των B+ κόμβων του δέντρου, και ‘Π’ ο συντελεστής πληρότητας. Οι κόμβοι-φύλλα έχουν μεταξύ β και 2β κλειδιά.

# Παράδειγμα Σχεδιασμού Συστημάτων:

# Συνύπαρξη προϊόντων

# Ευρετήριο B+ δέντρου

| # εγγραφές Συνύπαρξης | 1 δις | Δίνεται |
|---|---|---|
| Μέγεθος σελίδας ευρετηρίου | 64KB | Δίνεται |
| Αριθμος σελίδων που χωράνε σε 1 GB RAM | | |
| Πόσο μεγάλο είναι το β; | | |
| Μέσος όρος εγγραφών ευρετηρίου ανά σελίδα, με Π=3/4 | | |
| Αριθμός σελίδων ευρετηρίου για ευρετηρίαση (N) | | |
| Ριζικός κόμβος (επίπεδο 0) | | |
| # σελίδων στα επίπεδα 1, 2 | | |
| # επιπέδων στο B+ δέντρο για ευρετηρίαση 1 δις εγγραφών | | |
| Αριθμός IOs για να πάρει την εγγραφή για το ερώτημα=urlID | | |

# Παράδειγμα Σχεδιασμού Συστημάτων:

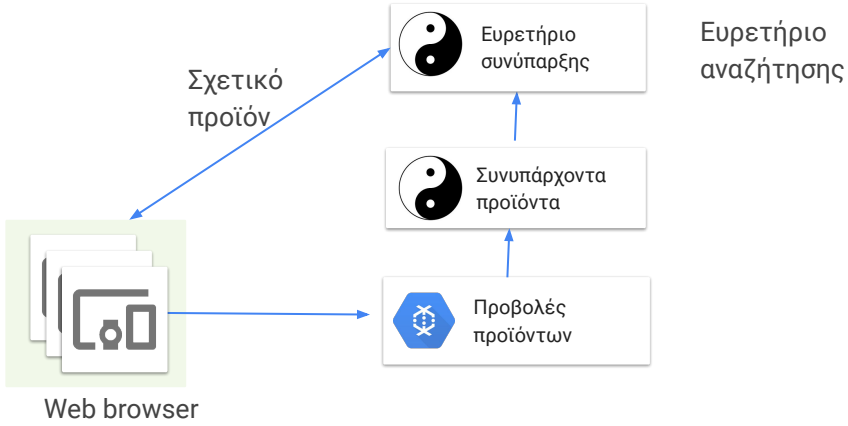
# Συνύπαρξη προϊόντων

# Ευρετήριο B+ δέντρου

| | | |
|---|---|---|
| # εγγραφές Συνύπαρξης | 1 δις | Δίνεται |
| Μέγεθος σελίδας ευρετηρίου | 64KB | Δίνεται |
| Αριθμος σελιδών που χωράνε σε 1 GB RAM | ~16k σελίδες | 1 GB/64KB |
| Πόσο μεγάλο είναι το β; | ~2730 | 4 bytes για productId + 8 bytes για δείκτες @ 64KB/σελίδα<br>2β * 4 + (2β+1)*8 <= 64k ⇒ β ~= 2730 |
| Μέσος όρος εγγραφών ευρετηρίου ανά σελίδα, με Π=3/4 | 4000 | ¾ του 2β ~= 4000 |
| Αριθμός σελιδών ευρετηρίου για ευρετηρίαση (N) | 250,000 | 1 δις εγγραφές/4000 |
| Ριζικός κόμβος (επίπεδο 0) | 1 σελίδα | Έχει ~4000 δείκτες |
| # σελίδων στα επίπεδα 1, 2 | 4000, 4000^2 | Στο επίπεδο n, έχει 4000^n σελίδες |
| # επιπέδων στο B+ δέντρο για ευρετηρίαση 1 δις εγγραφών | Ρίζα + 2 | 4000 <= 250,000<br>4000 + 4000^2 >= 250,000 |
| Αριθμός IOs για να πάρει την εγγραφή για το ερώτημα=urlID | 2 | Υπόθεσε ότι ρίζα και επίπεδο 1 μπορούν να είναι στην RAM (Το επίπεδο 2 χρειάζεται 4000^2 σελίδες >> 16k σελίδες στην RAM)<br># IOs: 1 για επίπεδο 2, 1 για εγγραφή |

# Παράδειγμα Σχεδιασμού Συστημάτων:

# Συνύπαρξη προϊόντων



# Παράδειγμα Σχεδιασμού Συστημάτων:

# Μεγαλύτερη συχνότητα συνύπαρξης δεδομένων

**Το πρόβλημα μέχρι στιγμής**

- Ο κατάλογος προϊόντων του AWA's έχει 1 δισεκατομμύριο αντικείμενα. Η AWA έχει 10 δισεκατομμύρια προβολές προϊόντων κάθε εβδομάδα, από 1 δισεκατομμύρια χρήστες. Κάθε καταγραφή log αποθηκεύει <userID, productID, viewID, χρόνο προβολής>

**Θεωρείστε 1000x Μεγαλύτερο πρόβλημα!**

- Ο κατάλογος των προϊόντων είναι **1 τρισεκατομμύριο αντικείμενα**. Η AWA έχει 10 δις προβολές προϊόντων. Τα υπόλοιπα παραμένουν τα ίδια

⇒ Τι αλλάζει;

# Παράδειγμα Σχεδιασμού Συστημάτων:

## Συνύπαρξη δεδομένων

## Προσχέδιο

| | Μέγεθος | Γιατί |
|---|---|---|
| ProductId | 8 bytes | 1 τρις προϊόντα ⇒ Χρειάζονται τουλ.40 bits (2^40 ~= 1 Δις) για αναπαράσταση κάθε προϊόντος ξεχωριστά. Οπότε χρηση 8 bytes. |
| UserID | 4 bytes | " |
| LogOfViewsID | 8 bytes | 10 Δις προβολές προϊόντων.. |
| Product | 1000 PB | 1 Τρις προϊόντα του 1 MB το καθένα |
| Users | Unknown | |
| LogOfViews | 280 GB | Κάθε εγγραφή είναι <userID, productID, viewID, viewtime>. Υπόθεση: χρησιμοποιούμε 8 bytes για viewTime. Οπότε 24 bytes ανά εγγραφή. 10 Δις*28 bytes = 280 GBs. |
| CoOccur | 20 GBs | Η έξοδος πρέπει να είναι <productID, productID, count> για την μέτρηση συνύπαρξης. Αυτά είναι, 20 bytes ανά εγγραφή (8 + 8 + 4 για τα 2 productIDs και 4 bytes για μετρητή). Για διατήρηση του κορυφαίου δις προϊόντων-ζευγών (όπως προτείνεται από την AWA ποιότητα δεδομένων), χρειάζεσαι 1 δις * 20 bytes = 20 GBs. |
| TempCoOccur | 10^24 counters | Για μέτρηση όλων των προϊόντων-ζευγών όπως σκανάρουμε την είσοδο, ίσως χρειαστούμε 1 τρις*1 τρις (10^24) μετρητές |
| TempCoOccur (με υπόθεση UserSession, των ~10 προβολών/χρήστη) | 1600 GB | # προϊόντων-ζευγών που παρήχθησαν:<br>1 δις χρήστες * 10^2 = 100 δις<br><br>Μέγεθος @ 16 bytes/εγγραφή = 1600 GBs. |

# Σχεδιασμός Συστήματος Δεδομένων

**Δημοφιλές πρότυπο σύστημα σχεδίασης**

1. Επαρκής υπολογισμός ομάδας (‘batch’) δεδομένων
2. (ταξινόμηση, κατακερματισμός, καταμέτρηση)
3. Κατασκευή Lookup ευρετηρίου στα αποτελέσματα (b+ Δέντρο, πίνακας κατακερματισμού)
4. Για ‘Ροή’ δεδομένων, ανανέωση με μικρές ομάδες (‘micro batches’)

**Δημοφιλή προβλήματα**

1. Σχετικά βίντεο (youtube), άτομα (Facebook), σελίδες (web)
2. Κίνδυνοι ασφάλειας, malware (ασφάλεια), Ανάλυση Συσχέτισης

# Ευχαριστούμε!