Δυναμικός Κατακερματισμός (Dynamic Hashing)

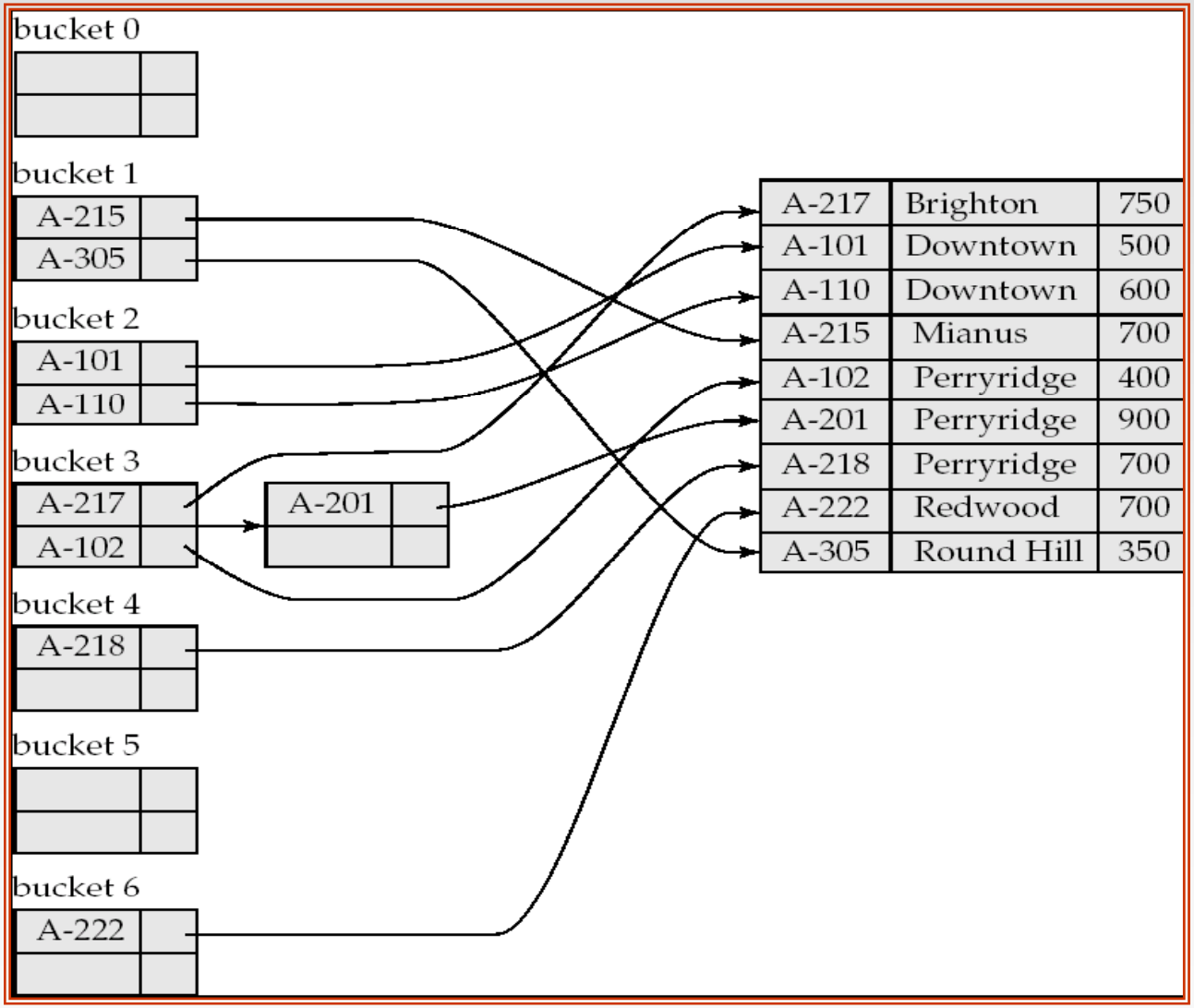
Σπύρος Σιούτας

Καθηγητής CEID@Upatras

Database System Concepts, 5th Ed.

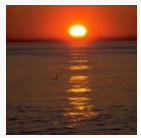
Hash Ευρετήρια

- Το Hashing χρησιμοποιείται όχι μόνο για οργάνωση αρχείων, αλλά και ως τεχνική δόμησης αποδοτικών ευρετηρίων.



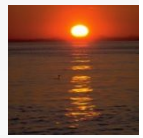
Μειονεκτήματα στατικού κατακ/μού

- Το σύνολο των διευθύνσεων κάδων είναι προκαθορισμένο.
  - Οι βάσεις δεδομένων μεγαλώνουν με το χρόνο. Αν ο αρχικός αριθμός κάδων είναι πολύ μικρός, η απόδοση θα μειωθεί λόγω των αλυσίδων υπερχείλισης.
  - Έστω και εάν είναι προβλέψιμο το μελλοντικό μέγεθος των αρχείων και προσδιοριστεί ανάλογα ο αριθμός των κάδων, στην αρχή θα σπαταλάται άσκοπα σημαντικό ποσό χώρου.
  - Όταν συρρικνώνεται η βάση δεδομένων, πάλι υπάρχει σπατάλη χώρου.
  - Μια λύση είναι η **περιοδική αναδιοργάνωση** του αρχείου με μια νέα συνάρτηση κατακερματισμού, αλλά αυτή η διαδικασία είναι πολύ ακριβή (σε χώρο και χρόνο).
- Αυτά τα προβλήματα αποφεύγονται με τεχνικές που επιτρέπουν να τροποποιείται δυναμικά ο αριθμός των κάδων
  - **Επεκτάσιμος κατακερματισμός** (extensible hashing).

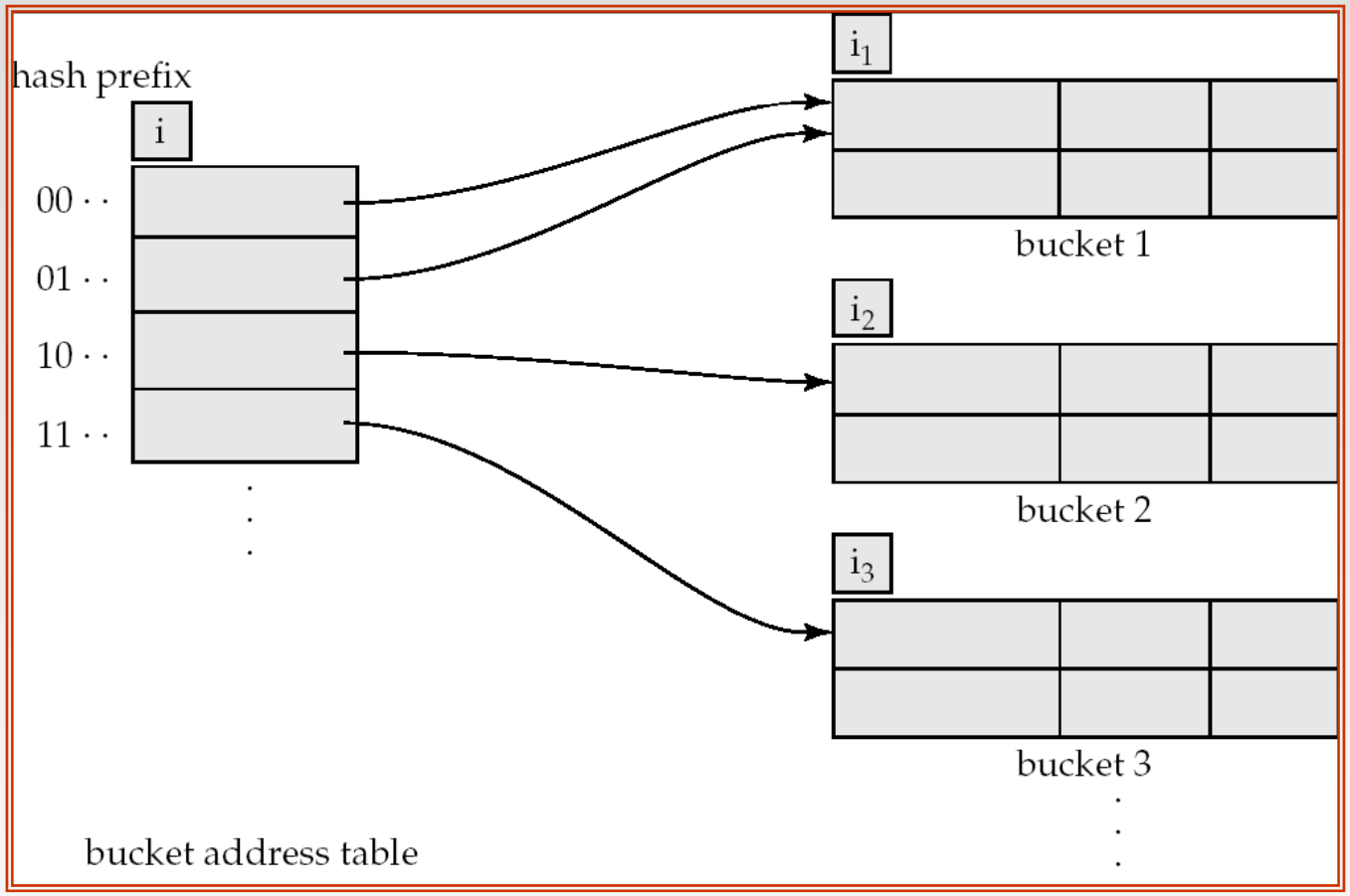


Δυναμικό Hashing

- Βολεύει για ΒΔ το μέγεθος των οποίων αυξομειώνεται
- Επιτρέπει στην hash συνάρτηση να τροποποιείται δυναμικά
- Μια μορφή δυναμικού hashing είναι το **Extendable hashing**
  - Η συνάρτηση Hash παράγει τιμές σε ένα μεγάλο εύρος — τυπικά ακεραίους των *b*-bits, όπου *b* = 32.
  - Ανά πάσα στιγμή μπορούμε να χρησιμοποιήσουμε το πρόθεμα (prefix) της συνάρτησης hash για να δεικτοδοτήσουμε τον πίνακα των διευθύνσεων των κάδων.
  - Έστω ότι το μήκος του προθέματος είναι *i* bits, $0 \leq i \leq 32$.
  - Bucket address table size = 2^i. Αρχικά $i = 0$
  - Το *i* αυξομειώνεται καθώς και η ΒΔ αυξομειώνεται.
  - Πολλαπλές καταχωρήσεις του bucket address table μπορεί να δείχνουν σε ένα bucket.
  - Άρα, ο ακριβής αριθμός των buckets < 2^i
    - Ο αριθμός των buckets επίσης αλλάζει δυναμικά εξαιτίας της συγχώνευσης ή της διάσπασης τους.



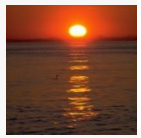
Γενική Extendable Hash Δομή



Σε αυτή τη δομή, $i_2 = i_3 = i$, όπου $i_1 = i - 1$ (για λεπτομέρειες δείτε επόμενη διαφάνεια)

Χρήση της Extendable Hash Δομής

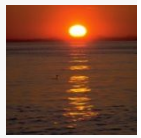
- Κάθε bucket *j* αποθηκεύει μία τιμή i_j . Όλα τα entries που δείχνουν στο ίδιο bucket έχουν τις ίδιες τιμές στα i_j bits.
- Για να βρούμε το bucket που περιέχει το κλειδί K_j:
  1. Υπολογίζουμε τη συνάρτηση $h(K_j) = X$
  2. Χρησιμοποιούμε τα πρώτα *i* bits της δυαδικής τιμής *X* ώστε να μετακινηθούμε στην κατάλληλη θέση του bucket address table, και ακολουθούμε τον κατάλληλο δείκτη προς τον κάδο
- Για να ενθέσουμε μία εγγραφή με τιμή K_j
  - Ακολουθούμε την ίδια διαδικασία εύρεσης με πριν για να εντοπίσουμε το κατάλληλο bucket, έστω *j*.
  - Αν υπάρχει χώρος στο bucket *j* ενθέτουμε εκεί μέσα την εγγραφή.
  - Διαφορετικά το bucket πρέπει να διασπαστεί και η ένθεση ξανά-επιχειρείται (δείτε επόμενη διαφάνεια).
    - Αντί αυτού, σε μερικές περιπτώσεις όπως θα δούμε χρησιμοποιούνται Overflow buckets



Ενημερώσεις της Extendable Hash Δομής

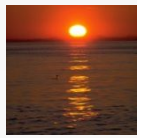
Για να κάνουμε split ένα bucket *j* όταν εντίθεται εγγραφή με κλειδί K_j:

- Αν $i > i_j$ (περισσότεροι από ένας δείκτες προς το bucket *j*)
  - Δεσμεύουμε ένα νέο bucket *z*.
  - Κάνουμε το δεύτερο μισό των entries του bucket address table που δείχνουν στο *j* να δείχνουν στο *z*
  - Διαγράφουμε και επανενθέτουμε κάθε εγγραφή στο bucket *j*.
  - Επαναϋπολογίζουμε ένα νέο bucket για το κλειδί K_j και ενθέτουμε την εγγραφή στο bucket (περαιτέρω splitting απαιτείται αν το bucket παραμένει ακόμη full)
- Αν $i = i_j$ (μόνο ένας δείκτης προς το bucket *j*)
  - Αυξάνουμε το *i* και διπλασιάζουμε το μέγεθος του bucket address table.
  - Αντικαθιστούμε κάθε 1 entry του table με 2 entries που δείχνουν στο ίδιο bucket.
  - Επαναϋπολογίζουμε ένα νέο bucket address table entry για το K_j. Προφανώς τώρα ισχύει $i > i_j$ οπότε σε πιθανό επόμενο splitting του bucket αυτού ισχύει η πρώτη περίπτωση.



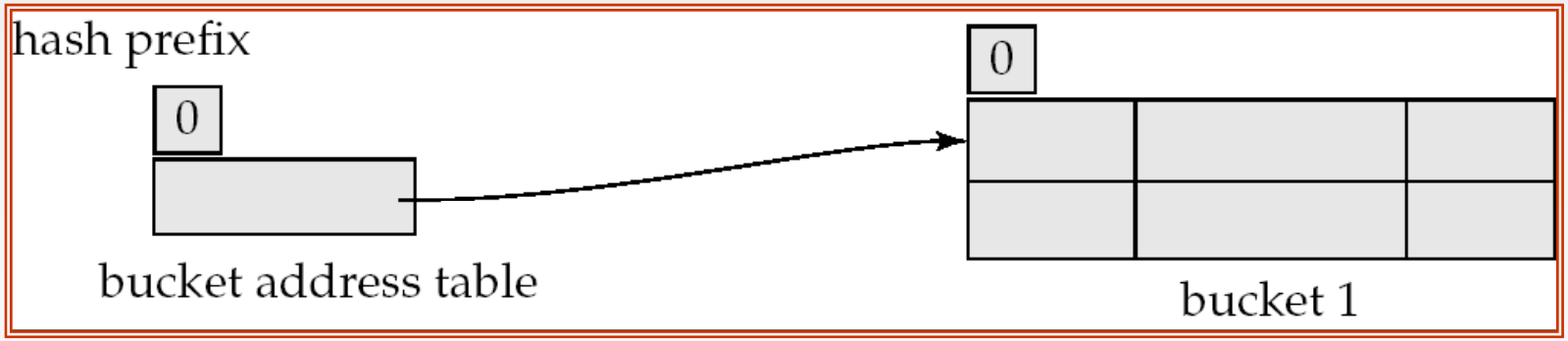
Ενημερώσεις της Extendable Hash Δομής (Συν.)

- Όταν ενθέτουμε μία τιμή, αν το bucket παραμένει full μετά από πολλά splits (που σημαίνει ότι το *i* αγγίζει το όριο *b*) δημιουργούμε ένα overflow bucket αντί να κάνουμε επαναλαμβανόμενα splittings στο bucket entry table.
- Για να διαγράψουμε ένα κλειδί,
  - Το εντοπίζουμε στο κατάλληλο bucket και το διαγράφουμε.
  - Αν το bucket γίνει empty, τότε θα πρέπει να διαγραφτεί (με τις κατάλληλες ενημερώσεις στο bucket address table).
  - Συγχώνευση από buckets μπορεί να γίνει (μπορεί να συγχωνευθεί μόνο με το "συνεταιρακι (buddy)" bucket που έχει την ίδια τιμή του i_j και το ίδιο i_j –1 πρόθεμα)
  - Μείωση στο μέγεθος του bucket address table μπορεί επίσης να γίνει
    - Σημ.: Η μείωση του μεγέθους του bucket address table είναι μία ακριβή πράξη και θα πρέπει να γίνεται μόνο στην περίπτωση που ο αριθμός των buckets γίνει πολύ μικρότερος από το μέγεθος του πίνακα.



Παράδειγμα

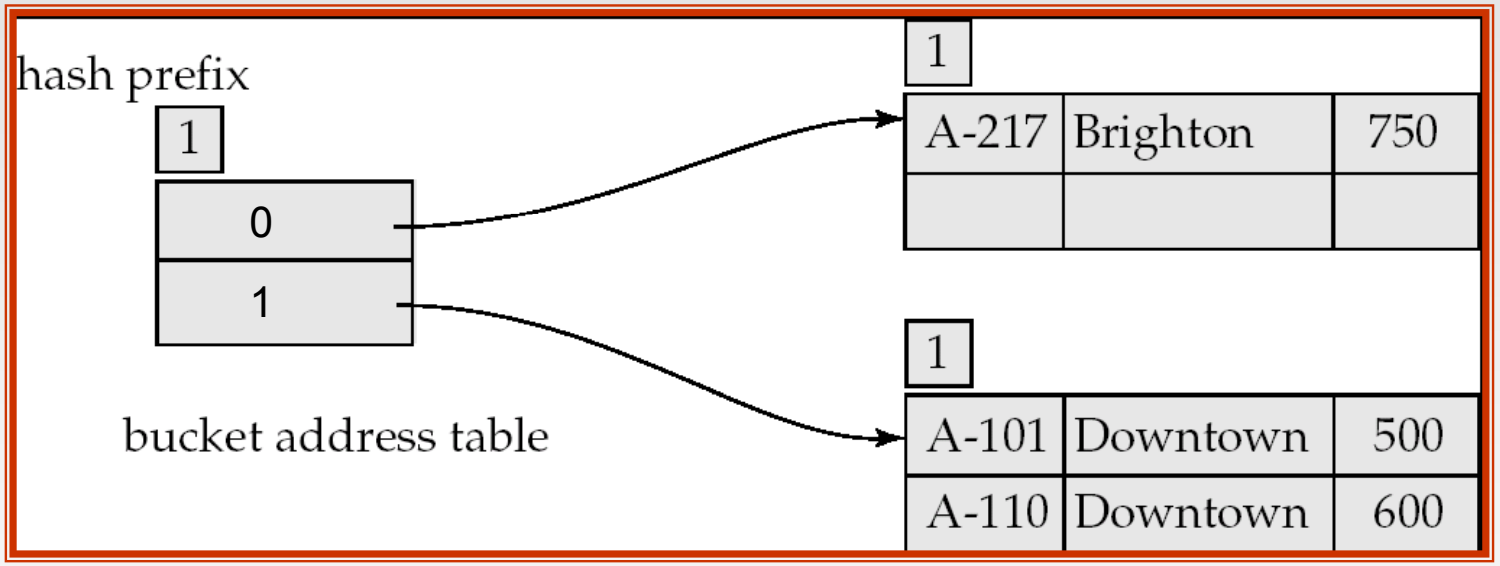
branch_name	h(*branch_name*)
Brighton	0010 1101 1111 1011 0010 1100 0011 0000
Downtown	1010 0011 1010 0000 1100 0110 1001 1111
Mianus	1100 0111 1110 1101 1011 1111 0011 1010
Perryridge	1111 0001 0010 0100 1001 0011 0110 1101
Redwood	0011 0101 1010 0110 1100 1001 1110 1011
Round Hill	1101 1000 0011 1111 1001 1100 0000 0001



Initial Hash structure, bucket size = 2

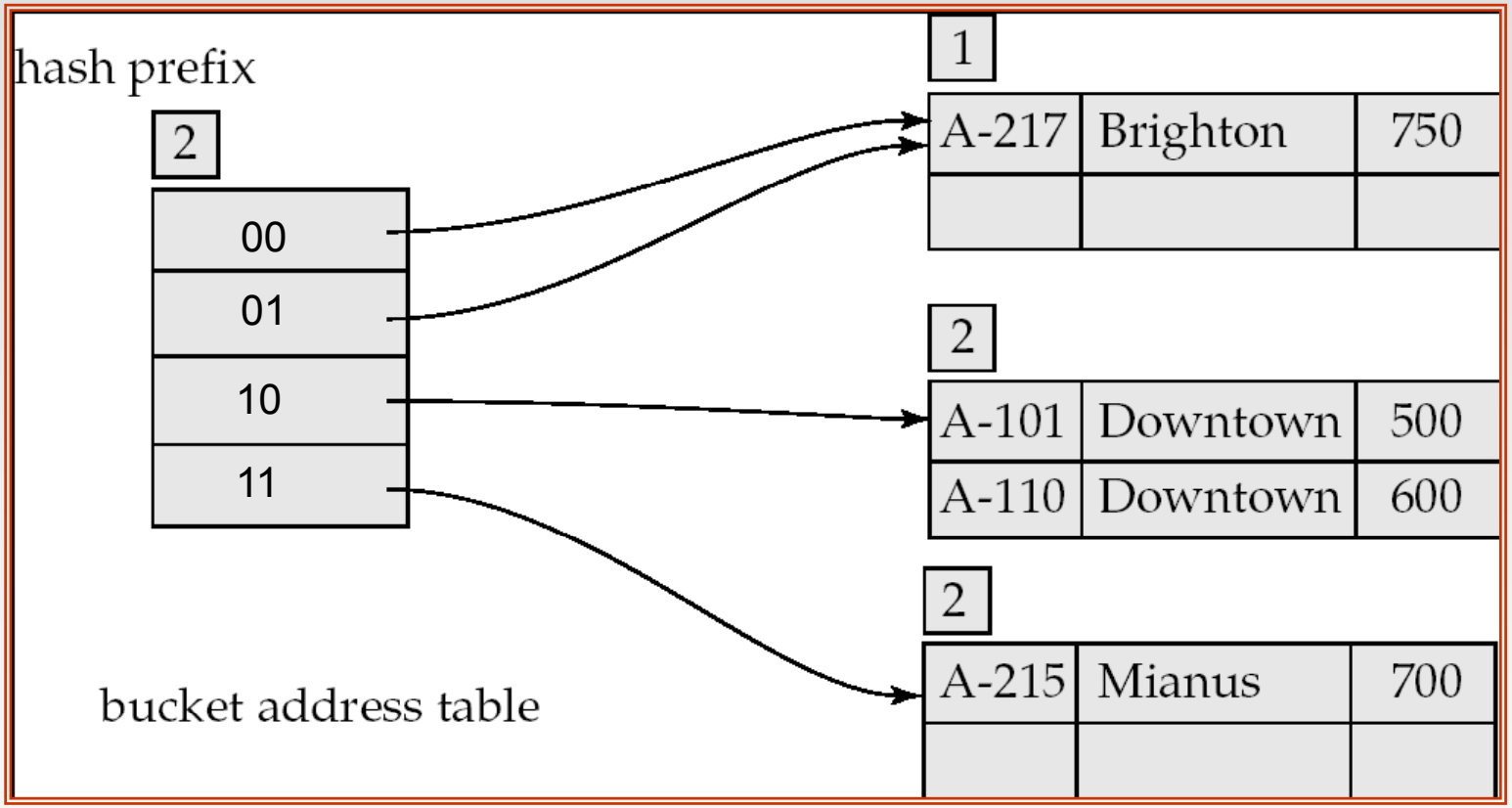
Παράδειγμα (Συν.)

- Η Hash δομή μετά την ένθεση ενός Brighton και δύο Downtown εγγραφών

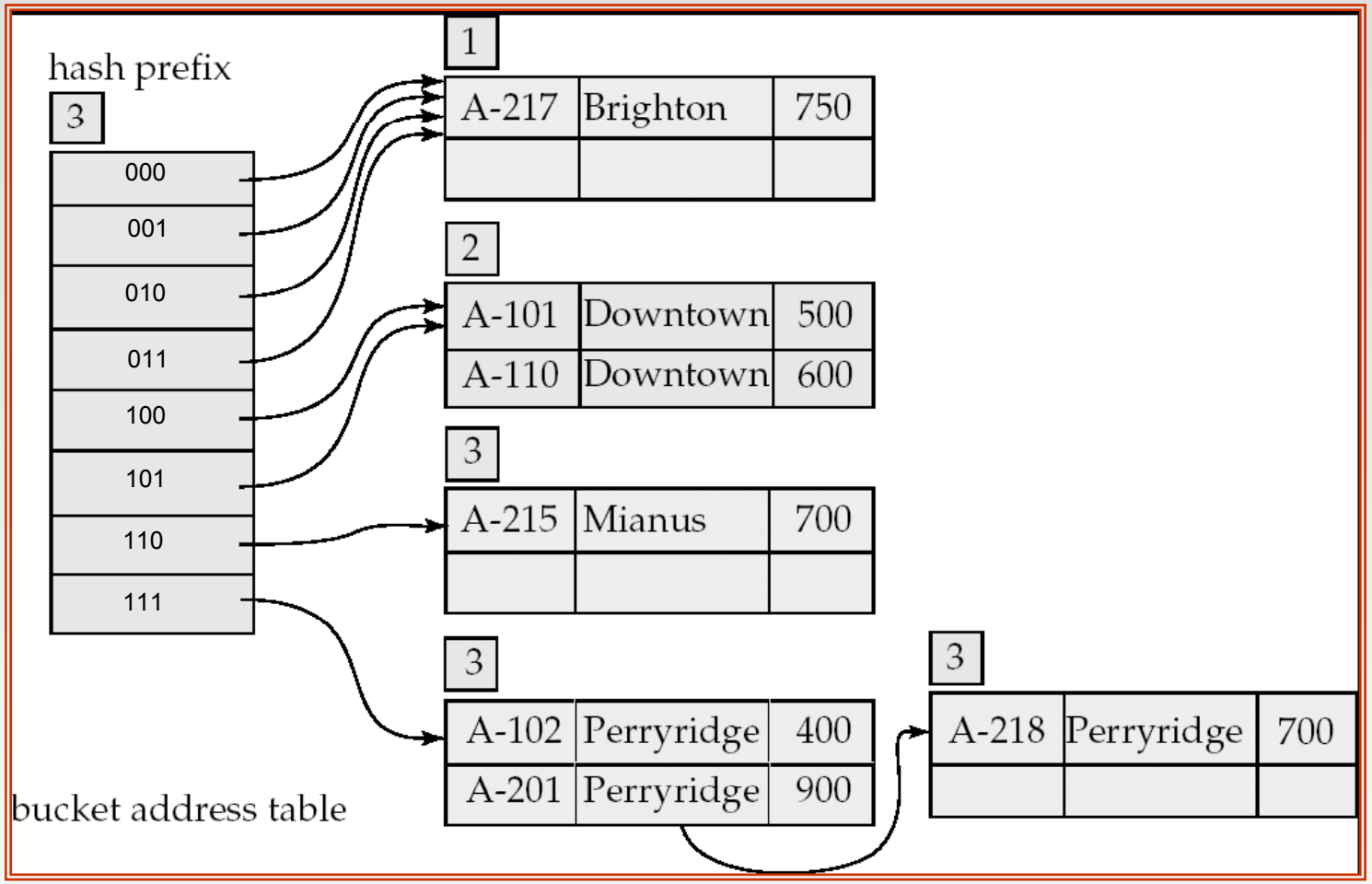


Παράδειγμα (Συν.)

Η Hash δομή μετά την ένθεση του Mianus record



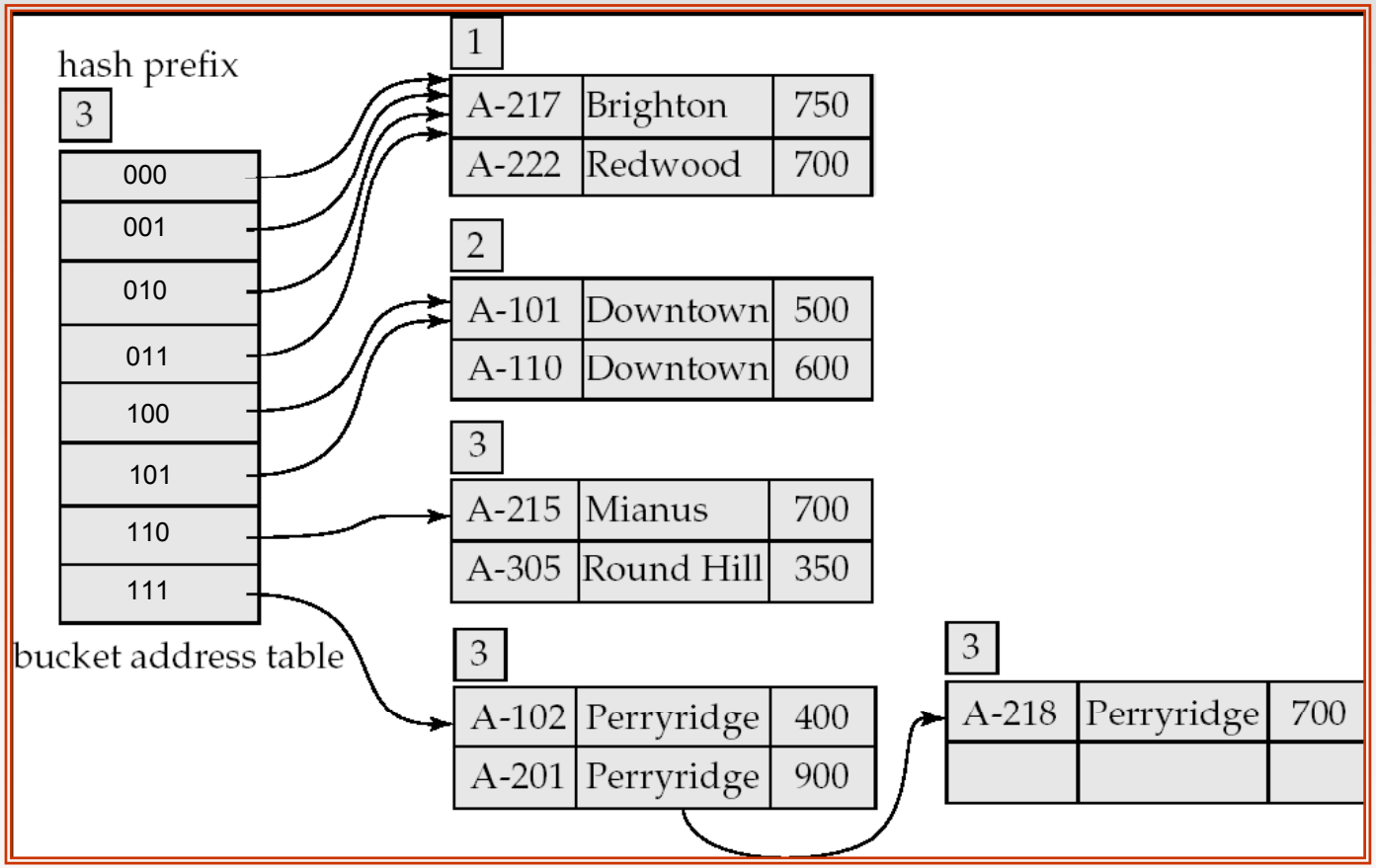
Παράδειγμα (Συν.)



Η Hash δομή μετά την ένθεση τριών Perryridge records

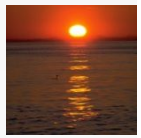
Παράδειγμα (Συν.)

- Η Hash δομή μετά την ένθεση των Redwood και Round Hill records



Πότε Προτιμάμε Δυναμικό Κατακερματισμό? Από τι εξαρτάται?

- Κόστος περιοδικής αναδιοργάνωσης.
- Σχετική συχνότητα εισαγωγών και διαγραφών.
- Αναμενόμενοι τύποι ερωτήσεων:
  - Ο κατακερματισμός είναι προτιμότερος για **ερωτήσεις ταυτότητας** πάνω στο κλειδί διάταξης.
  - Για **ερωτήσεις διαστήματος** πάνω στο κλειδί διάταξης επιλέγονται τα ταξινομημένα ευρετήρια.
    - Ο κατακερματισμός **δεν** μπορεί να υποστηρίξει ερωτήσεις διαστήματος



ΤΕΛΟΣ

Database System Concepts, 5th Ed.