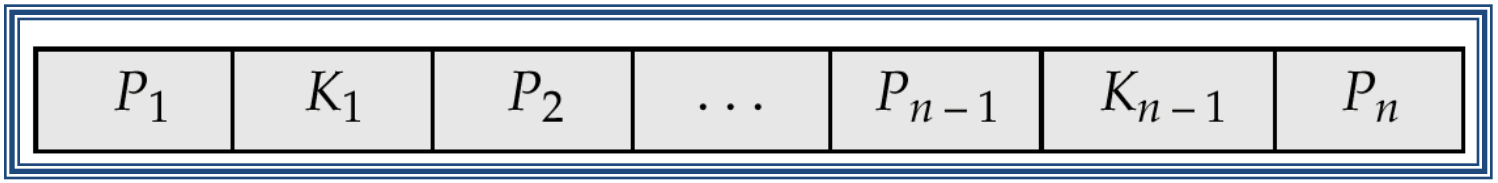
B+-Tree Index Files

تعريف : شجرة B+-tree هي شجرة معممة تحقق مجموعة من الخواص

- تقع الأوراق في مستوى واحد .
- كل عقدة ليست جذر وليست ورقة لها بين $[n/2]$ و n ولد.
- ضمن كل ورقة ما بين $[(n-1)/2]$ و $n-1$ قيمة.
- الحالات الخاصة :
  - إذا لم يكن جذر الشجرة ورقة فله على الأقل ولدان.
  - إذا كان جذر الشجرة ورقة ، فهو يحوي على عناصر ما بين ٠ و n-1 قيمة.

بنية عقدة في B^+-Tree

- لعقدة من الشجرة الشكل التالي

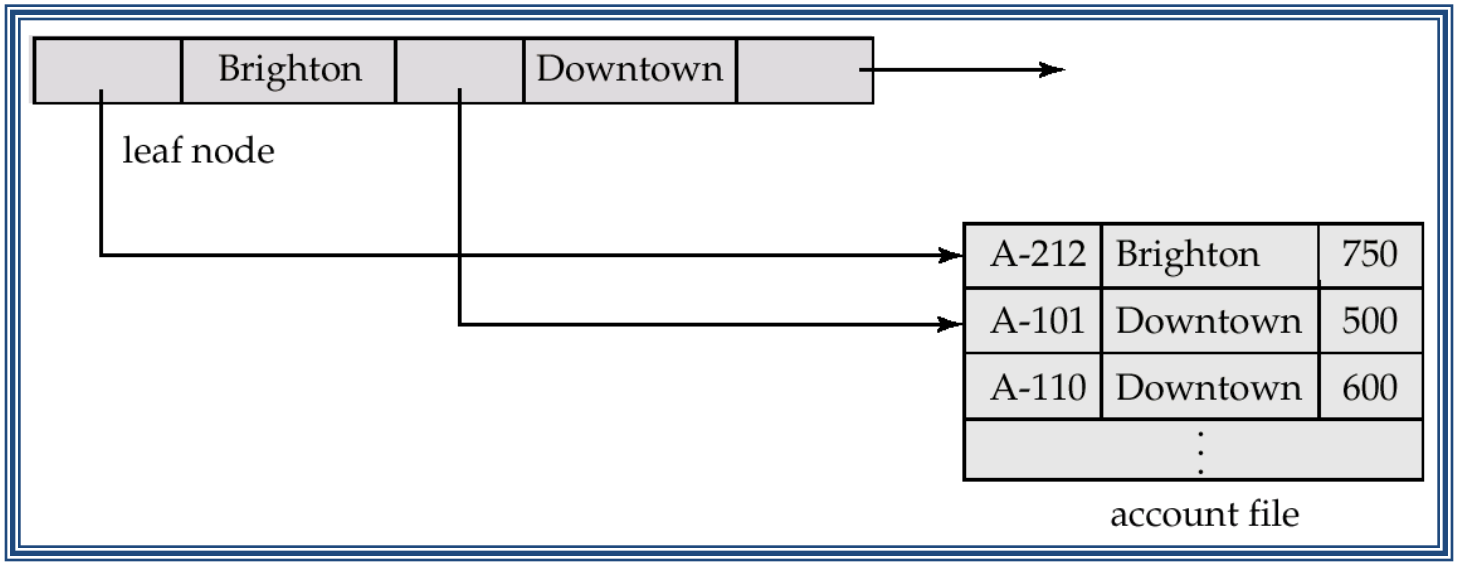


- K_i قيم مفتاح البحث
- P_i مؤشرات للأبناء في حالة عقدة داخل الشجرة ليست ورقة، أو مؤشرات إلى رزمة من التسجيلات في حالة عقدة ورقة.
- تكون قيم مفتاح البحث في العقدة مرتبة فيما بينها

$$K_1 < K_2 < K_3 < \ldots < K_{n-1}$$

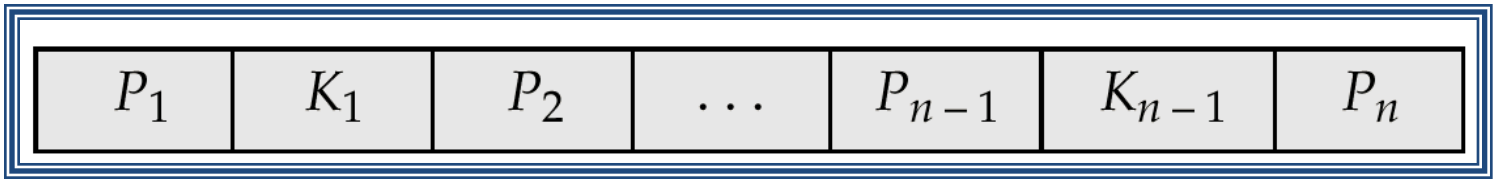
الأوراق في شجرة B^+

- إذا كان لدينا L_i ، L_j ورقتان في الشجرة حيث $i < j$ فجميع قيم مفتاح البحث في i أصغر من القيم في j.
- P_n يؤشر إلى الورقة التالية ضمن ترتيب مفتاح البحث.

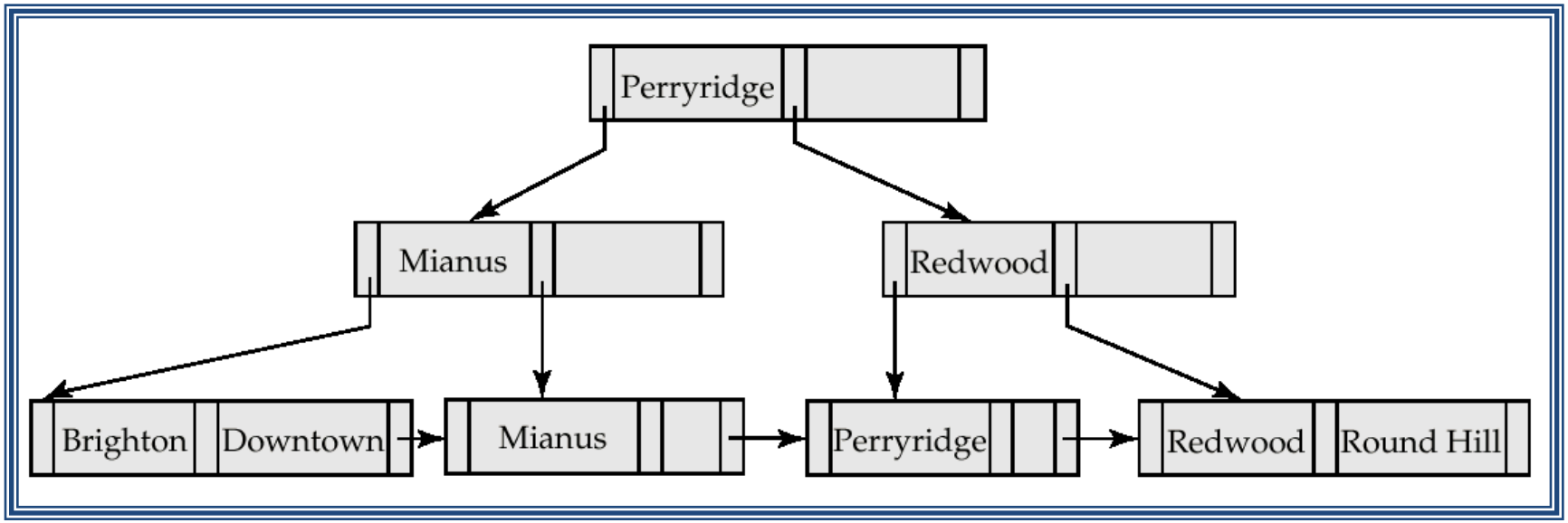


العقد الداخلية في B^+-Trees

- العقد الداخلية تُشكل فهرس أجوف متعدد المستويات لعقد الأوراق في الشجرة. وهي تحقق الخواص التالية :
  - جميع قيم مفتاح البحث في الشجرة الجزئية التي يُشير إليها المؤشر P_1 أصغر من القيمة K_1
  - جميع قيم مفتاح البحث في الشجرة الجزئية التي يؤشر إليها المؤشر P_i هي أكبر أو تساوي K_{i-1} وأصغر تماماً من K_i

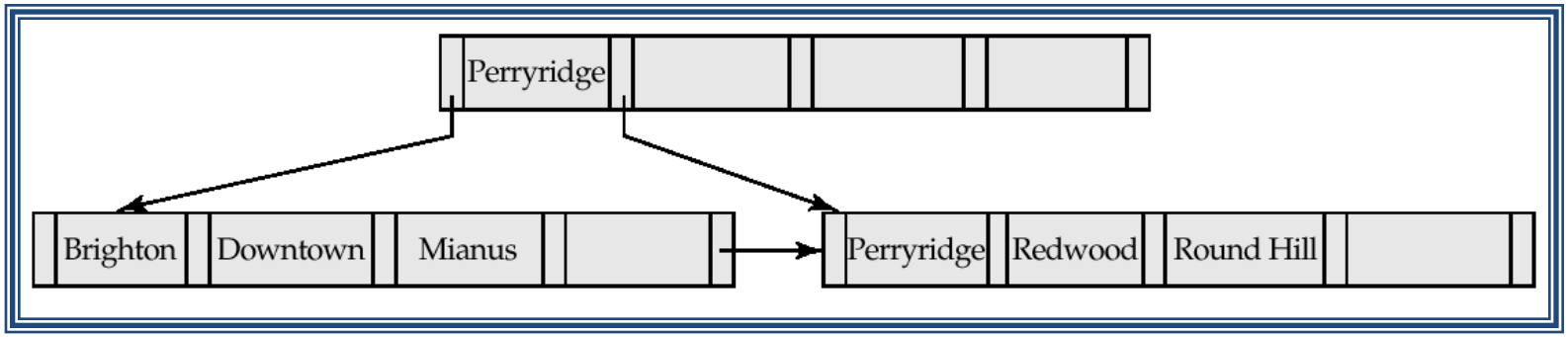


B^+-tree مثال



B^+-tree for *account* file ($n = 3$)

B^+-tree مثال



B^+-tree for *account* file ($n = 5$)

ملاحظات حول B^+-trees

- تحوي الشجرة عدد قليل نسبياً من المستويات (يتعلق بحجم الملف الأساسي)، مما يسمح بإجراء عمليات البحث بفعالية

- عمليات الإضافة والحذف من وإلى الملف الأساسي تتم بفعالية وكذلك إعادة بناء الفهرس تتم بوقت مقبول (لوغاريتمي)

الاستعلام باستخدام فهرس B^+-trees

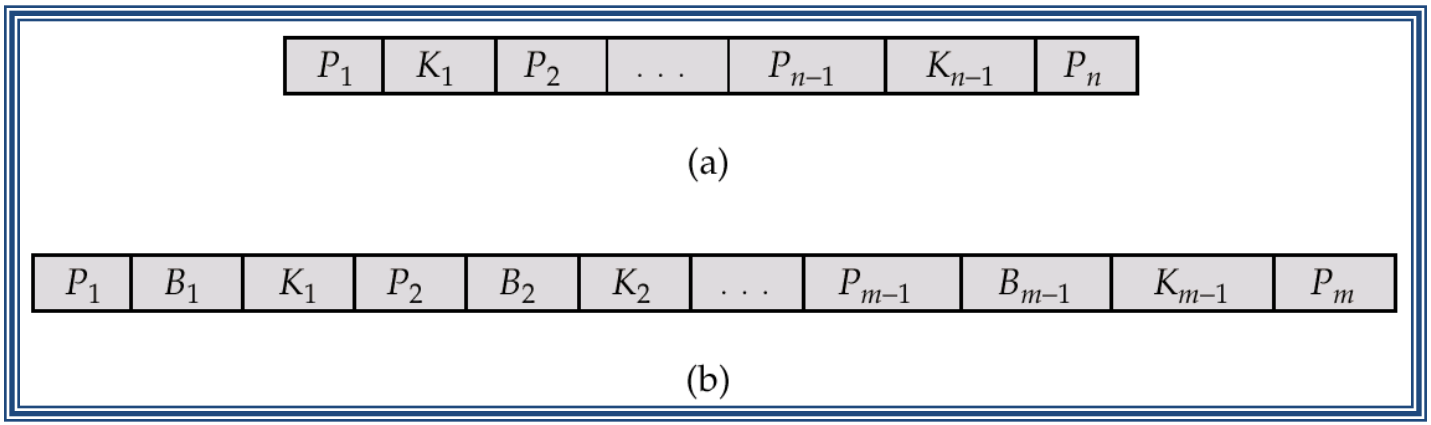
- ايجاد جميع التسجيلات الموافقة لقيمة محددة لمفتاح البحث ولتكن k
  - البدء من عقدة الجذر
    - البحث عن أصغر قيمة أكبر من k ضمن العقدة
    - في حال وجود هذه القيمة ولتكن ki المتابعة بالبحث في العقدة المشار إليها بـ Pi.
    - في حال كانت القيمة k أكبر من أخر قيمة في العقدة المتابعة مع المؤشر Pn
  - في حال وصلنا إلى عقدة ليست ورقة، إعادة الخطوات السابقة والمتابعة مع المؤشر الناتج.
  - عند وصولنا إلى عقدة ورقة ، إذا وجدت i بحيث ki=k فإن المؤشر Pi سوف يقودنا إلى التسجيلة التي نبحث عنها. وإلا فلا وجود للتسجيلة.

تنظيم ملف الفهرس من الشكل B^{+}-Tree

- يجري تخزين التسجيلات مباشرة في عقد الأوراق في شجرة B+ بدلاً من استخدام مؤشرات.
- من المفضل أن تكون عقد الأوراق نصف مملوءة .
- تتم عمليات الإضافة والحذف إلى الملف الأساسي بنفس الوقت التي تجري فيها عمليات الإضافة والحذف إلى شجرة B+-tree

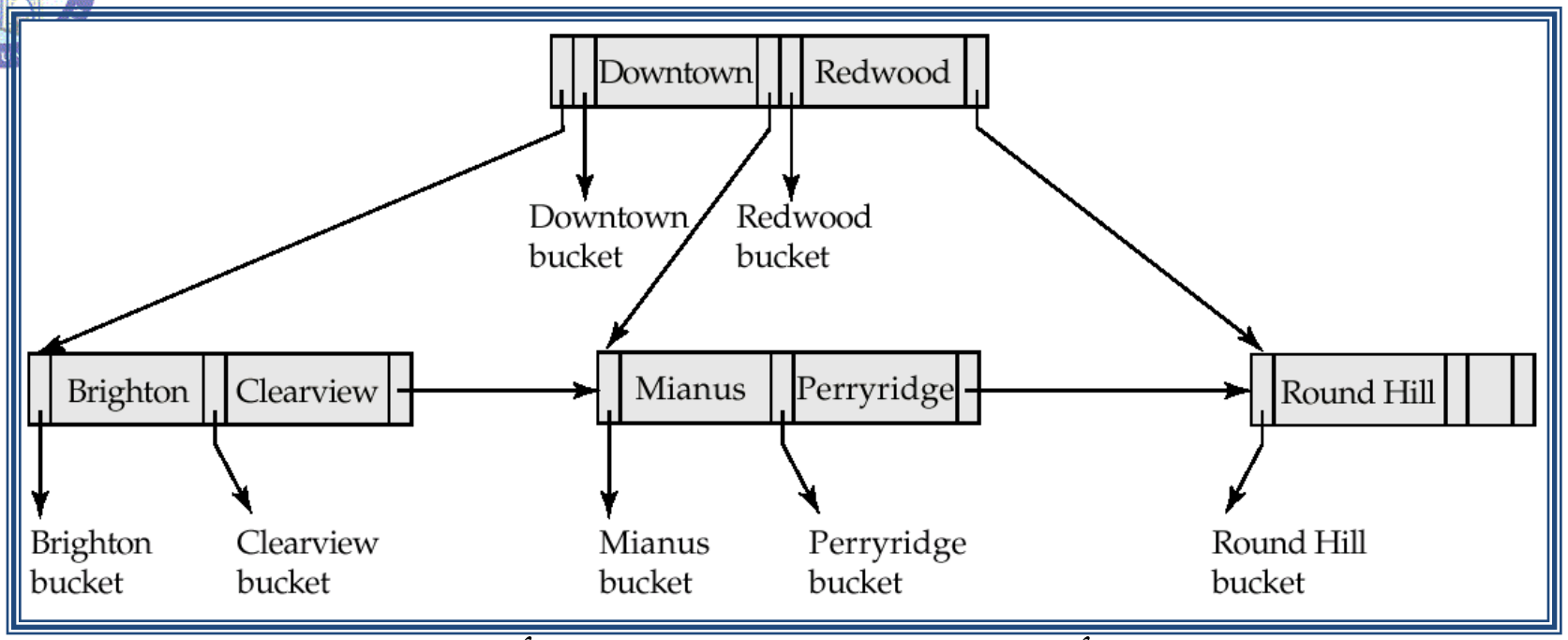
الفهرس B-Tree

- **مشابه لتنظيم الفهرس من نوع B+-tree مع عدم وجود تكرار لقيم مفتاح البحث ضمنها.**
- **قيم مفتاح البحث الذي يظهر في عقدة ليست ورقة لا يظهر في أي مكان آخر، وبالتالي نحتاج لإضافة مؤشر آخر لكل قيمة مفتاح بحث تظهر في العقد الداخلية.**

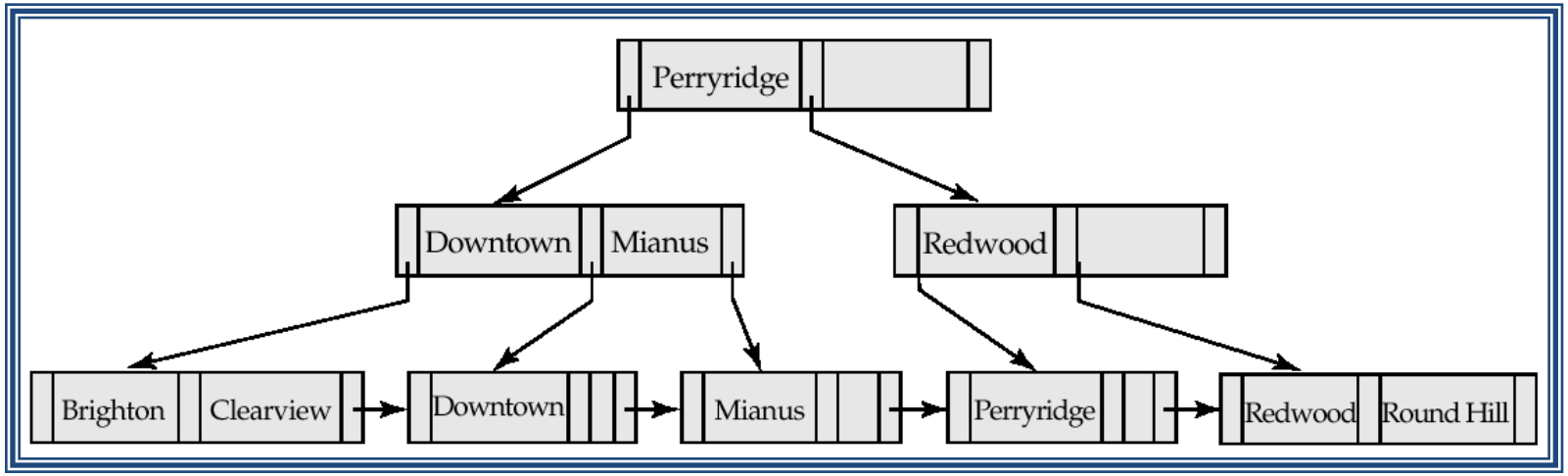


- في العقد الداخلية (غير الأوراق) – المؤشرات B_i تُشير نحو رزمة أو تسجيلة من الملف الأساسي.

مثال يبين الفرق بين الفهرسين B-tree وB+-tree



B-tree في الأعلى وB+-tree في الأسفل على المعطيات



الفهرس B-Tree

- الأدلة من النوع B-Tree
  - تحتاج إلى عدد عقد أقل منه في B^+-Tree
  - في بعض الأحيان نصل إلى المفتاح الذي نبحث عنه قبل الوصول إلى الأوراق
- لكن
  - فقط بعض القيم يمكن الوصول إليها أبكر
  - العقد غير الأوراق تحتاج إلى مكان تخزين أكبر، وبالتالي عدد العناصر في العقدة أقل، وبالتالي عمق الشجرة B-Trees أكبر منه في B^+-Tree
  - عملية الإضافة والحذف في B-Trees أعقد.

التقطيع الثابت
Static Hashing

- الرزمة **bucket** هي وحدة تخزين تحوي على تسجيلة أو أكثر (عادة تساوي كتلة disk blockأو أكثر).
- يُطبق تابع التقطيع **hash function** على قيمة مفتاح البحث لمعرفة الرزمة الحاوية على التسجيلة.
- مثال : تخزين علاقة account في ملف تقطيع مقسم إلى ١٠ رزم باستخدام مفتاح البحث branch-name وتابع التقطيع h بحيث :

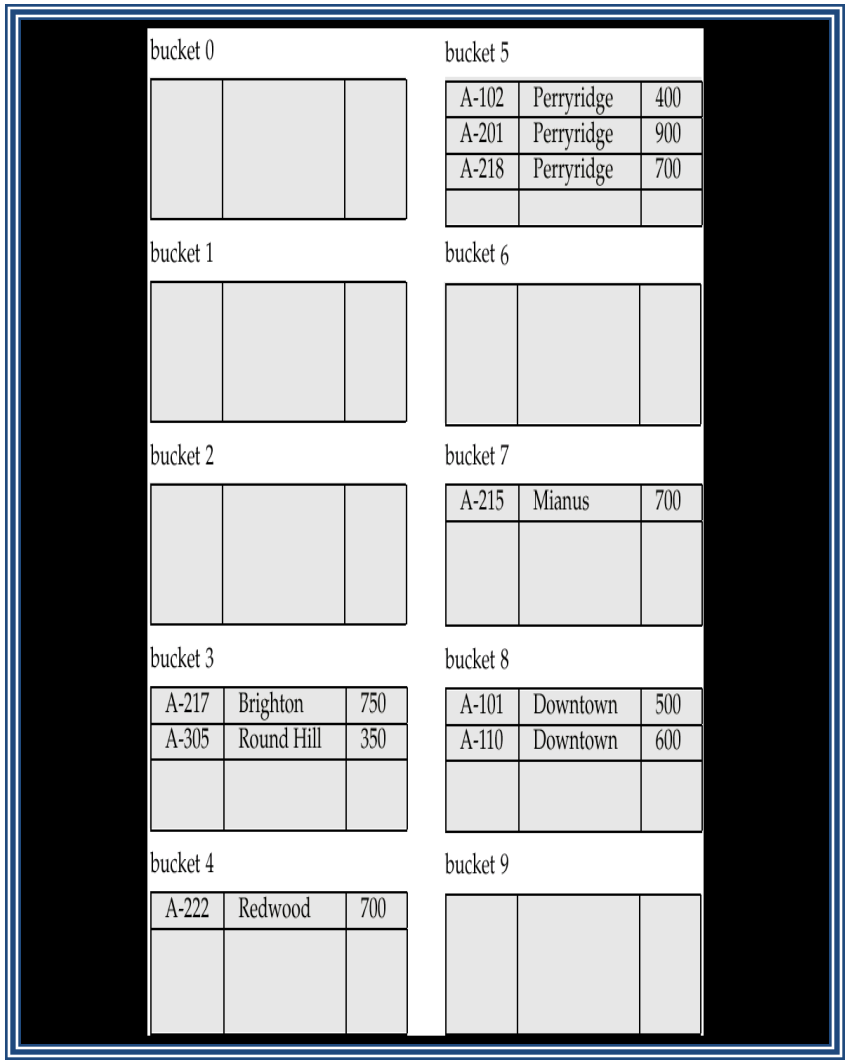
$$h(name) = (\sum name[i]) \bmod 10$$

وبالتالي:

h(Perryridge) = 5

h(Round Hill) = 3

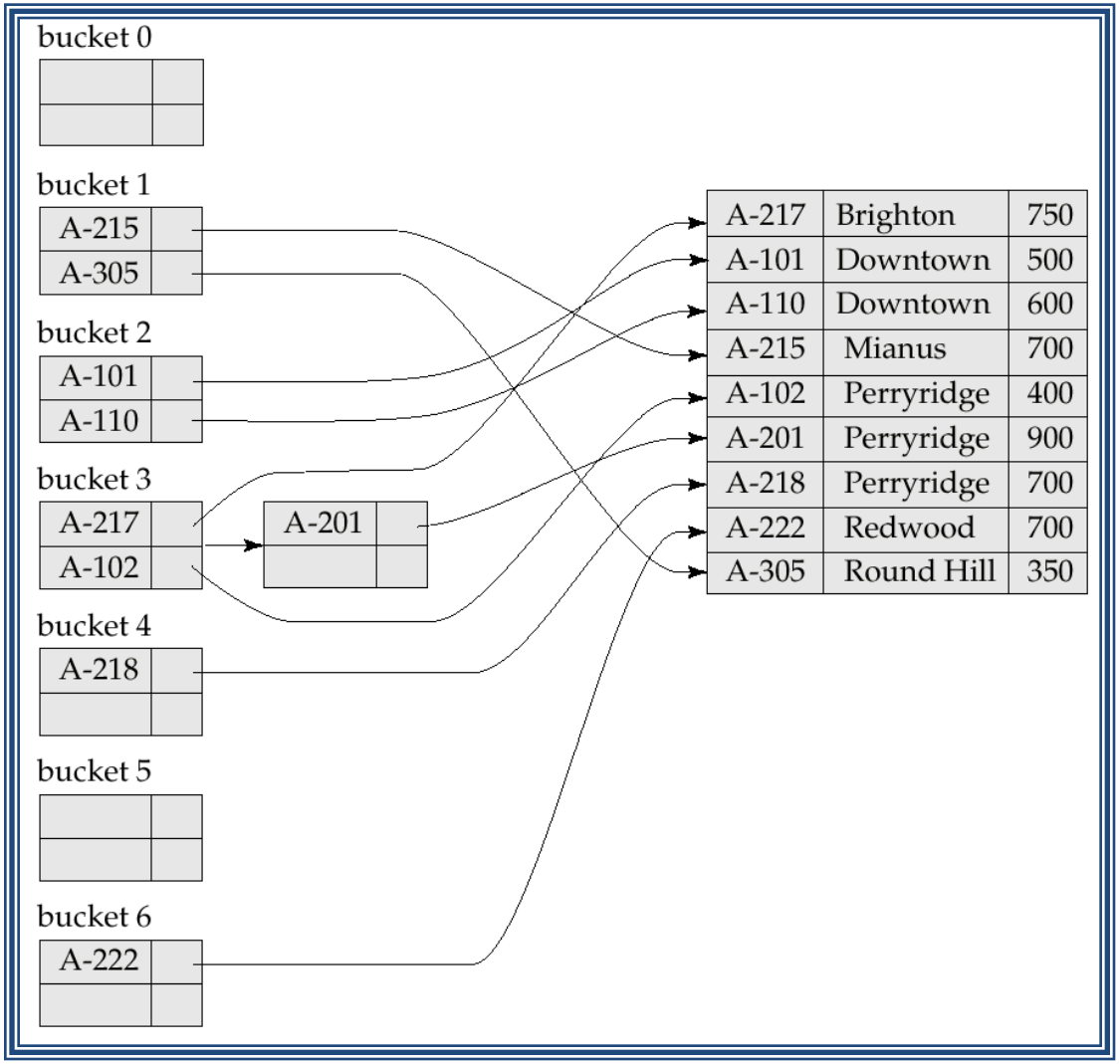
h (Brighton) = 3



الأدلة المقطعة
Hash Indices

- يمكن استخدام التقطيع في تنظيم الملفات وكذلك في بناء بنى الفهارس.
- يُنظم فهرس التقطيع مفاتيح البحث والمؤشرات في تسجيلات موافقة في بنية ملف تقطيع.
- فهرس التقطيع هو دوماً فهرس ثانوي
  - في حال كان الملف منظم بطريقة التقطيع، فإن استخدام فهرس تقطيع رئيسي باستخدام مفتاح البحث نفسه غير ضروري.
  - نستخدم المصطلح فهرس مقطع hash index للدلالة على بنى الفهرس الثانوي secondary index structures والملفات المنظمة كملفات تقطيع.

مثال


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

استخدامات فهرس التقطيع

- تابع التقطيع h يقابل قيم مفاتيح البحث بـ B رزمة من العناوين التي تُشير إلى الملف الأساسي
  - قاعدة المعطيات تنمو مع الوقت. فإذا كان عدد الرزم البدائي صغير، فإن الأداء سيتناقص نتيجة لكثرة الطوافان overflows
  - نحتاج إلى إعادة تنظيم الملف بشكل دوري واستخدام تابع تقطيع جديد لتقليل التصادم قدر المستطاع (حل غالي الثمن).
- يمكن تجنب الكثير من المشاكل باستخدام تقنية تسمح تعديل عدد الرزم بشكل ديناميكي

مقارنة بين الفهرسة المرتبة والتقطيع
Comparison of Ordered Indexing and Hashing

- كلفة إعادة التنظيم الدورية : عامل من العوامل الهامة التي تدخل في تحديد نوع تنظيم الفهرس
  - التردد النسبي لعمليات الإضافة والحذف
- نوع الاستعلامات المتوقعة على القاعدة :
  - عادة التقطيع يكون أفضل في حال استعلامات عن تسجيلات لها قيمة محددة للمفتاح.
  - في حال الاستعلام عن مجال قيم للمفتاح، تكون الفهرسة المرتبة أفضل.

تعريف الفهرس بلغة SQL

- توليد فهرس

create index <index-name> **on** <relation-name>
(<attribute-list>)

مثال

create index *b-index* **on** *branch (branch-name)*

- لحذف فهرس

drop index <index-name>

الفهارس من نوع bitmap

- هو نوع خاص من الفهارس، مصمم لزيادة فعالية الاستعلامات المرتكزة على مفاتيح متعددة الواصفات.
- من المفترض أن تكون تسجيلات العلاقة مرقمة بشكل تسلسلي، لنقل ابتداءً من الصفر ٠
  - من المفروض أن يكون الحصول على تسجيلة بعد إعطاء رقمها سهل
- قابل للتطبيق على الواصفات التي تأخذ عدداً من القيم المتمايزة القليلة نسبياً.
  - مثال : نوع، البلاد، الحالة العائلية،...
  - مثال : مستوى الدخل (يجري تجزئة الدخل إلى عدد قليل من المستويات مثل 0-9999, 10000-19999, 20000-50000, 50000- infinity

- القيم المخزنة في فهرس الـ bitmap هي عبارة عن شعاع من البتات array of bits

فهارس الـ Bitmap

- فهرس الـbitmap بشكله البسيط لواصفة هو عبارة عن قائمة من الـbitmap لكل قيمة مختلفة من قيم الواصفة.
  - لكل قيمة bitmap عدد من البتات يساوي عدد التسجيلات في العلاقة.
  - من أجل القيمة v للواصفة تكون قيمة الـbitmap الموافقة مؤلفة من مجموعة من البتات بحيث تكون البت الخاصة بالتسجيلة رقم i : تساوي ١ إذا كانت قيمة الواصفة في هذه التسجيلة تساوي v، وإلا تساوي ٠.

record number	*name*	*gender*	*address*	*income-level*
0	John	m	Perryridge	L1
1	Diana	f	Brooklyn	L2
2	Mary	f	Jonestown	L1
3	Peter	m	Brooklyn	L4
4	Kathy	f	Perryridge	L3

Bitmaps for *gender*

m	10010
f	01101

Bitmaps for *income-level*

L1	10100
L2	01000
L3	00001
L4	00010
L5	00000

فهارس الـ Bitmap

- مفيدة في الاستعلامات على عدة واصفات
- يتم الإجابة على الاستعلامات باستخدام عمليات الـ bitmap :
  - التقاطع (and)
  - الاجتماع or
  - الإتمام not
- تأخذ كل عملية قيمتين bitmap لهما نفس الطول وتطبق العملية على الخانات الموافقة للحصول على قيمة bitmap التي تشكل النتيجة.
  - مثال

100110 AND 110011 = 100010

100110 OR 110011 = 110111

NOT 100110 = 011001