



التخزين وبنى الملفات

Storage and File Structure



تنظيم الملفات

- تُخزن قاعدة المعطيات كتجمع من الملفات
 - يتألف كل ملف من مجموعة من التسجيلات
 - تتكون التسجيلية من مجموعة من الحقول
- يجري تخزين تسجيلات كل علاقة على حدى في ملف خاص بها.
- يحوي الملف تسجيلات من علاقات مختلفة



تنظيم التسجيلات في الملف

- **تنظيم الكومة Heap** الاضافة على بداية الملف وهي الطريقة الافتراضية التي يتبعها DBMS في حال أننا لم نحدد له طريقة
- **تنظيم تسلسلي Sequential** تخزين التسجيلات بشكل متسلسل اعتماداً على قيمة مفتاح البحث search key لكل تسجيلية.
- **تنظيم تقطيع Hashing** باستخدام تابع تقطيع لحساب قيمة تقطيع مجموعة من حقول التسجيلية يجري تحديد مجموعة الكتل من الملف التي يتم وضع التسجيلية فيها.
- ✓ يمكن تخزين كل علاقة في ملف منفصل. في تنظيم الملف المجمع **clustering file organization** يمكن تخزين التسجيلات من علاقات مختلفة في ملف واحد.



استخدام البنى الإضافية المساعدة للتخزين

Indexing and Hashing

- المفاهيم الأساسية
- الأدلة المرتبة
- ملفات الفهرسة من النوع B+Tree
- ملفات الفهرسة من النوع B-Tree
- التقطيع الثابت
- المقارنة بين الفهرسة المرتبة والتقطيع
- تعريف الفهرس في SQL
- الوصول من خلال مفاتيح متعددة



المفاهيم الأساسية

- تُستخدم تقنية الفهرسة لتسريع عملية الوصول إلى المعلومات المرغوبة. مثال : فهرس المؤلفون في المكتبة
- مفتاح البحث **Search Key** : واصفة أو أكثر تُستخدم للبحث عن تسجيلة من الملف
- ملف الفهرس **index file** مؤلف من مجموعة من التسجيلات تُدعى مداخل الفهرس من الشكل

search-key	pointer
------------	---------

- ملف الفهرس أصغر من الملف الأساسي
- هناك نوعان أساسيان من الأدلة
 - الأدلة المرتبة **Ordered indices** : تُخزن مفاتيح البحث بترتيب مفروز
 - الأدلة المقطعة **Hash indices** : تتوزع مفاتيح البحث بشكل منتظم ضمن رزم باستخدام تابع التقطيع.



تقييم فعالية الفهرس المستخدم

• نوع الوصول الذي يقدمه بفعالية

– البحث عن قيمة محددة

– البحث ضمن مجال من القيم

• كلفة العمليات الأساسية

• حجم التخزين الإضافي



الأدلة المرتبة Ordered Indices

✓ في الفهارس المرتبة **ordered index** يتم تخزين مداخل الفهرس مفروزة حسب قيم مفتاح البحث.

• الفهرس الرئيسي **Primary index** هو الفهرس المعرف على ملف تسلسلي وبحيث يحدد مفتاح البحث الترتيب التسلسلي للملف.

– يُدعى أيضاً بـ الفهرس المجمع **clustering index**

– عادة يكون مفتاح البحث في الفهرس الرئيسي هو مفتاح رئيسي (ليس بالضرورة)

• الفهرس الثانوي **Secondary index** هو الفهرس المعرف على مفتاح بحث يختلف عن الترتيب التسلسلي للملف. يُدعى أيضاً

الفهرس غير المجمع **non-clustering index**

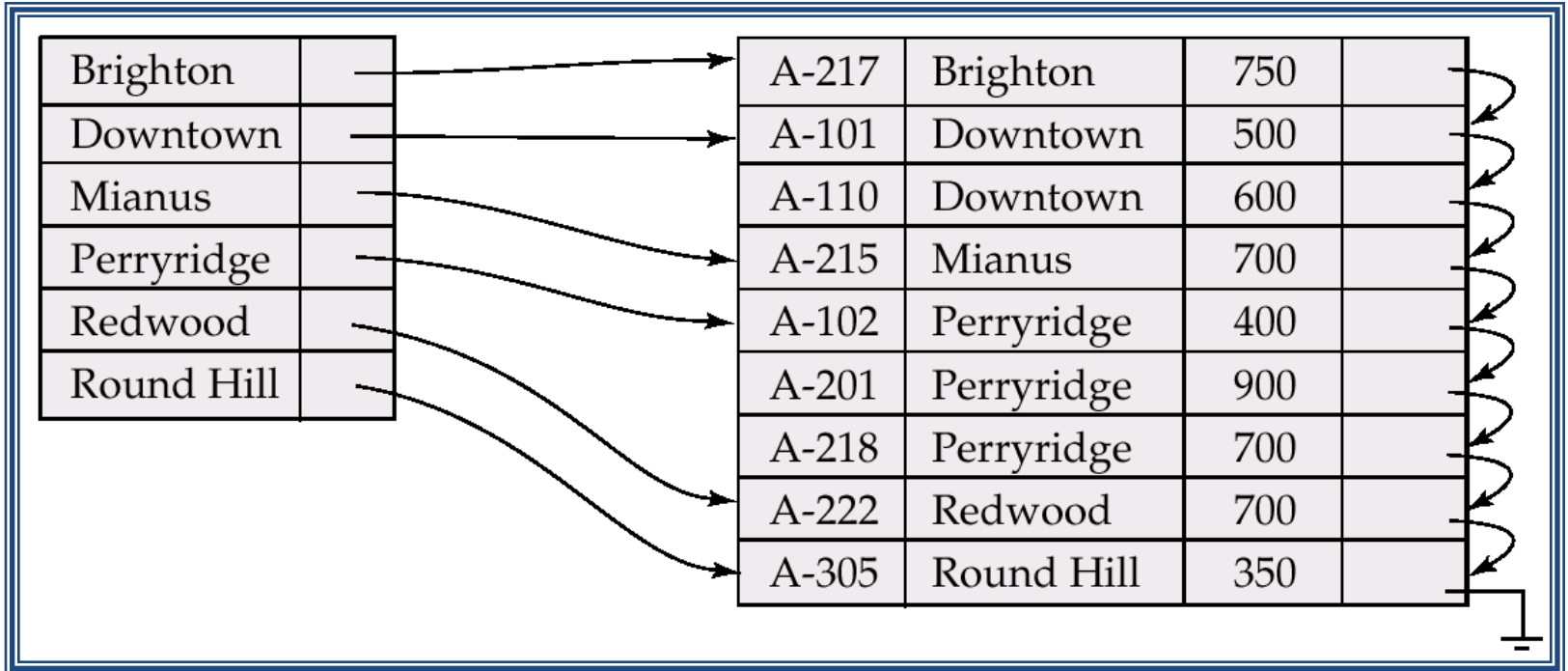
✓ نقول أنه لدينا ملف تسلسلي مفهرس **Index-sequential file** إذا كان الملف مرتب تسلسلياً ومرتبب بفهرس رئيسي.



ملفات الفهرس الكثيف

Dense Index Files

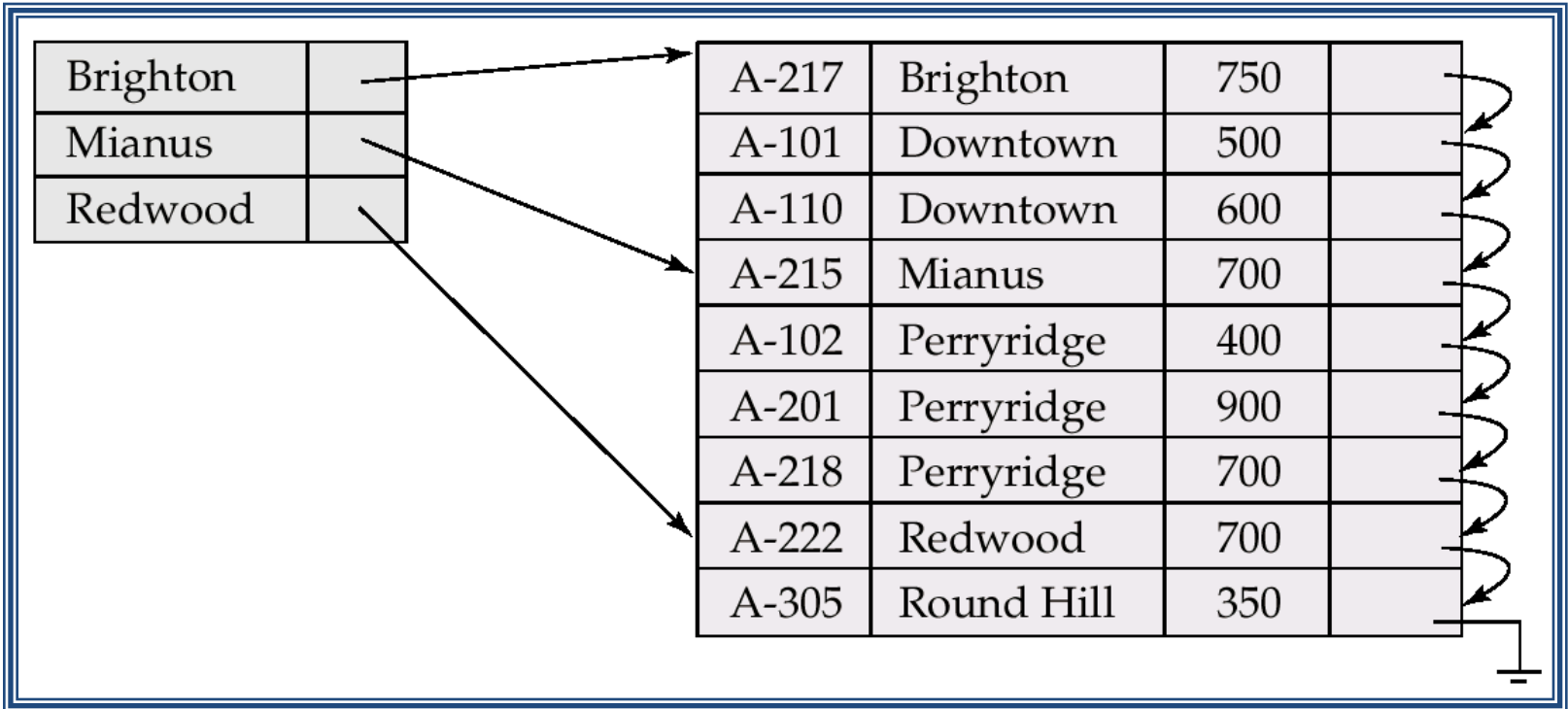
- **Dense index** : يجري تخزين تسجيلة في الفهرس من أجل كل قيمة لمفتاح البحث في الملف.



ملفات الفهرس الأجوف

Sparse Index Files

- **Sparse Index** : يحوي تسجيلات فهرسة من أجل فقط بعض قيم مفتاح البحث – قابل للتطبيق فقط في حال كانت التسجيلات في الملف الأساسي مرتبة بشكل تسلسلي على مفتاح البحث.
- يحتاج إلى مساحة تخزين أقل ويحتاج إلى عمليات إصلاح قليلة عند إجراء عمليات الإضافة والحذف على الملف الأساسي.
- بشكل عام أبطأ في الحصول على تسجيلة محددة من استخدام الفهرس الكثيف.





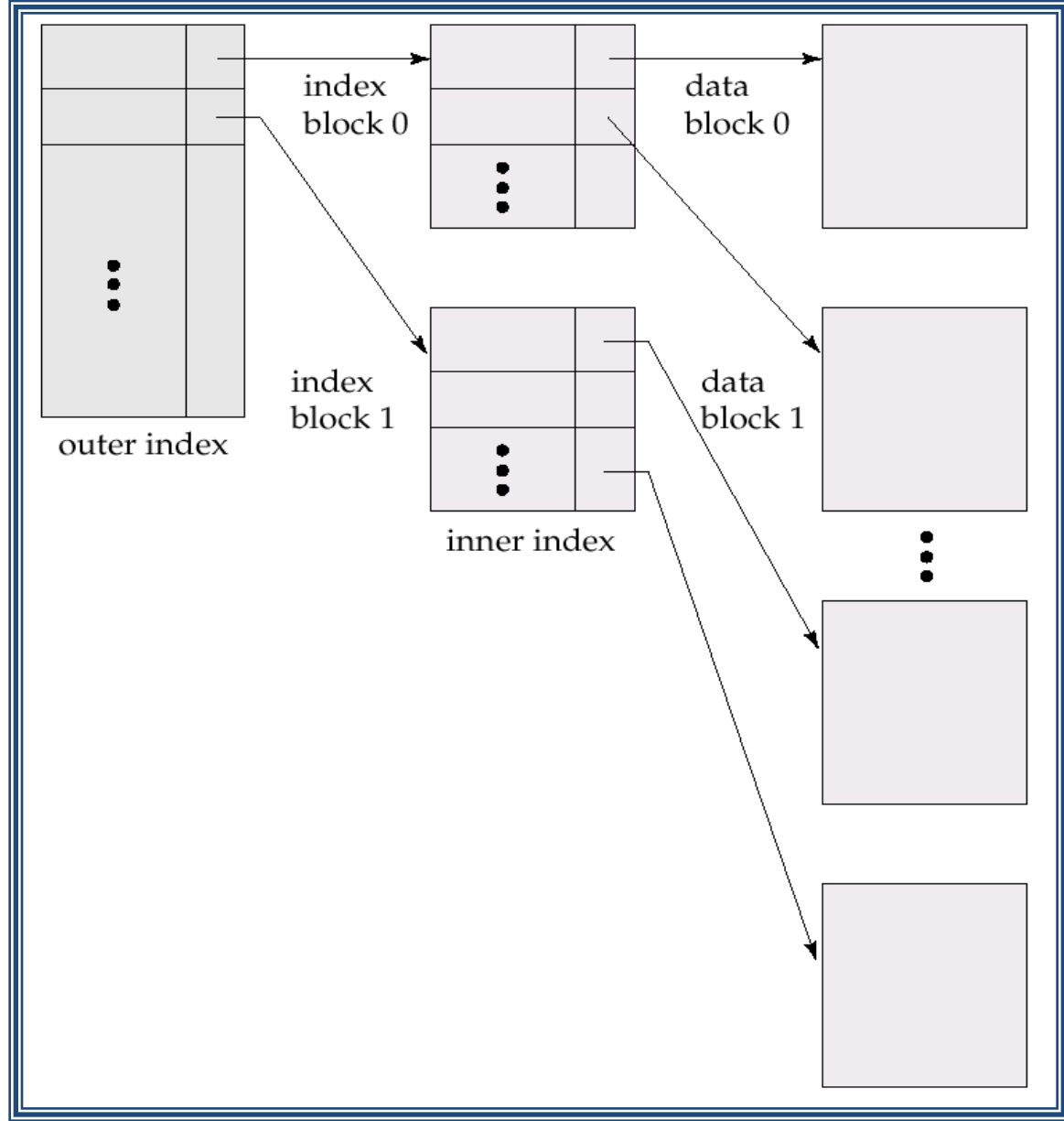
الفهرس متعدد المستويات

Multilevel Index

- في حال لم نستطيع شحن الفهرس الرئيسي بشكل كامل في الذاكرة الرئيسية، يصبح كلفة عملية الوصول إلى المعلومة تحتاج إلى عمليات وصول على مستويين (عملية مكلفة).
- لتقليص عدد الوصول الكتلي إلى القرص التي نحتاجه للوصول إلى التسجيلة المعنية في الفهرس ، يتم تشكيل فهرس أجوف للفهرس الرئيسي المخزن كملف تسلسلي على القرص.
- فهرس خارجي outer index : وهو عبارة عن فهرس أجوف للفهرس الرئيسي
- فهرس داخلي inner index : ملف الفهرس الرئيسي.
- في حال بقي الفهرس الخارجي كبيراً لشحنه في الذاكرة الرئيسية، نقوم بتوليد مستوى ثاني من الفهرس، وهكذا.
- نحتاج إلى إجراء عملية تعديل في جميع المستويات عند إجراء عمليات الإضافة والحذف على الملف الأساسي.



Multilevel Index





الفرق بين الفهارس التسلسلية و الفهارس متعددة المستويات

• الفهارس التسلسلية

- يتناقص الأداء كلما كبر الملف حيث يؤدي ذلك إلى ربط عدد من الكتل الإضافية overflow blocks.
- إعادة تنظيم للملف بأكمله بشكل دوري.

• الفهارس متعدد المستويات

- تجري عملية إعادة تنظيم للفهرس بشكل آلي عند القيام بعمليات الإضافة والحذف، تتطلب هذه العملية تغييرات بسيطة ومحلية.
- لا يوجد حاجة لإعادة تنظيم الملف.
- عمليات إضافة وحذف إضافية،
- تخزين إضافي.



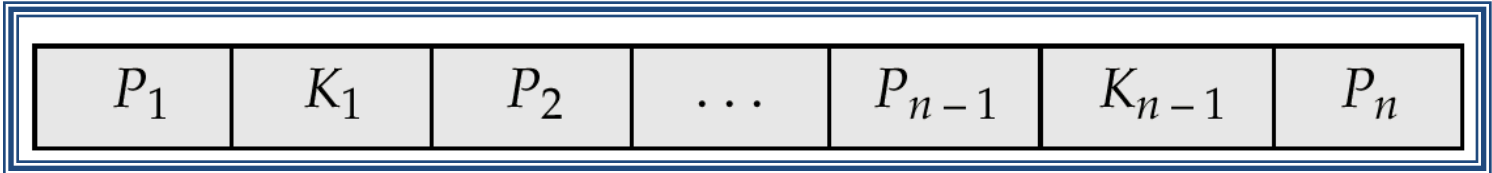
B⁺-Tree Index Files

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

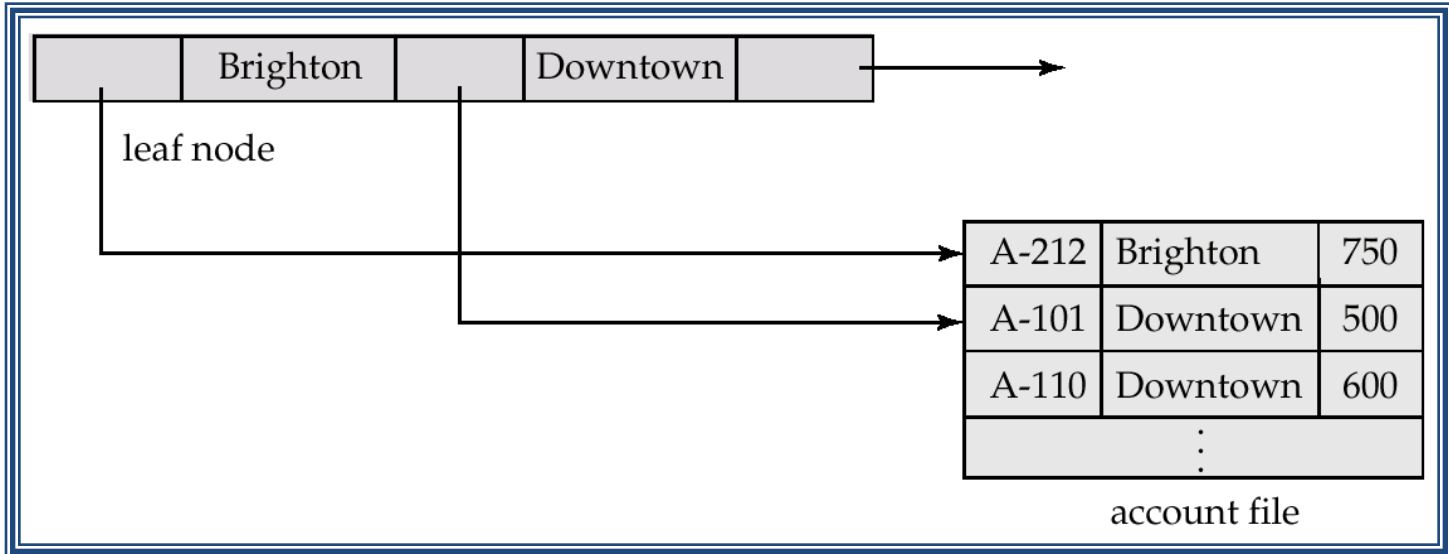


- K_i قيم مفتاح البحث
- P_i مؤشرات للأبناء في حالة عقدة داخل الشجرة ليست ورقة، أو مؤشرات إلى رزمة من التسجيلات في حالة عقدة ورقة.
- تكون قيم مفتاح البحث في العقدة مرتبة فيما بينها

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

الأوراق في شجرة B⁺

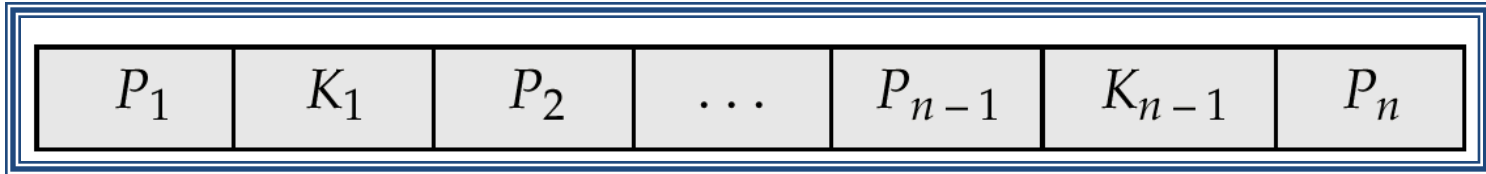
- إذا كان لدينا L_i ، L_z ورقتان في الشجرة حيث $i < z$ فجميع قيم مفتاح البحث في i أصغر من القيم في z .
- P_n يُوْشِر إلى الورقة التالية ضمن ترتيب مفتاح البحث





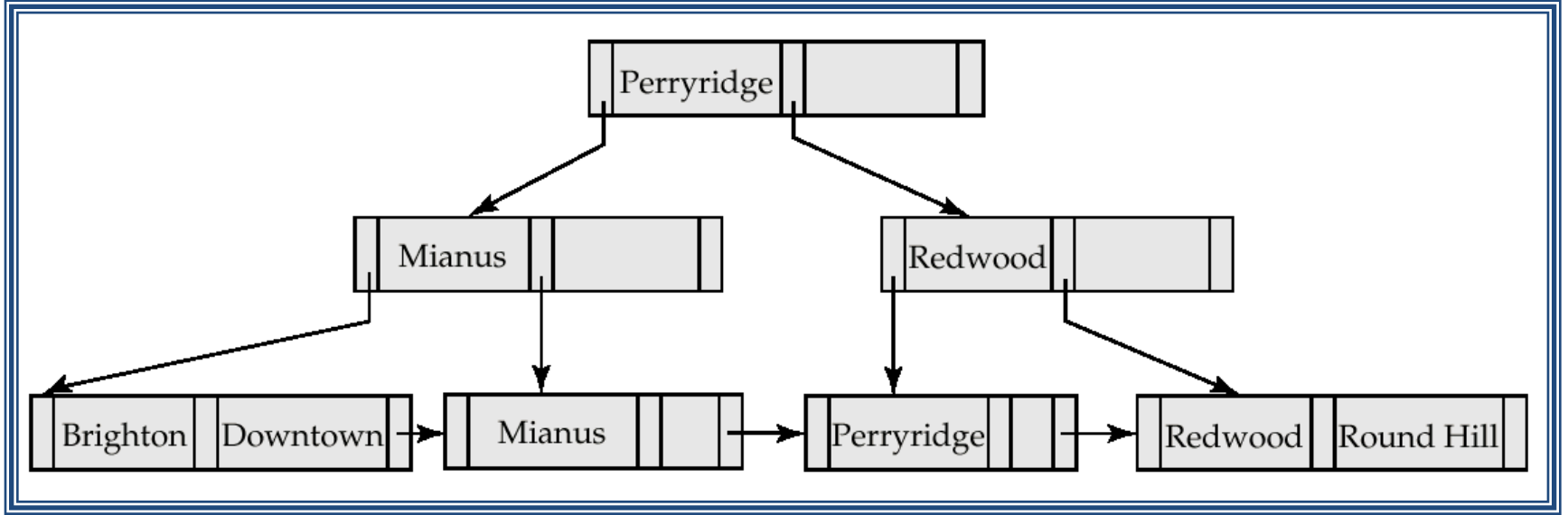
العقد الداخلية في B⁺-Trees

- 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 يُشير إليها المؤشر P_i هي أكبر أو تساوي K_{i-1} وأصغر تماماً من K_i



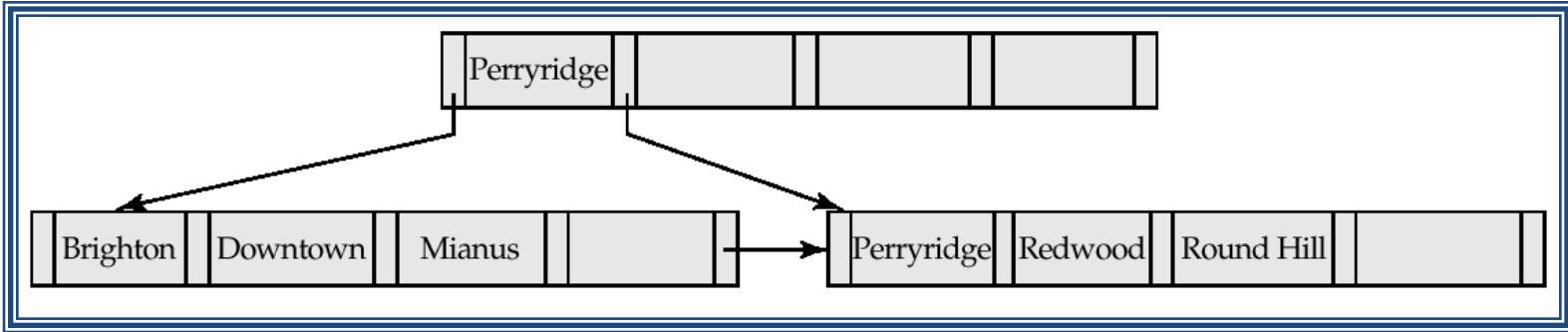


مثال B⁺-tree



B⁺-tree for *account* file ($n = 3$)

مثال B⁺-tree



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 (لو غار يتمي)



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 k_i المتابعة بالبحث في العقدة المشار إليها بـ P_i .
- في حال كانت القيمة k أكبر من آخر قيمة في العقدة المتابعة مع المؤشر P_n

– في حال وصلنا إلى عقدة ليست ورقة، إعادة الخطوات السابقة والمتابعة مع المؤشر الناتج.

– عند وصولنا إلى عقدة ورقة ، إذا وجدت i بحيث $k_i = k$ فإن المؤشر P_i 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.



(a)

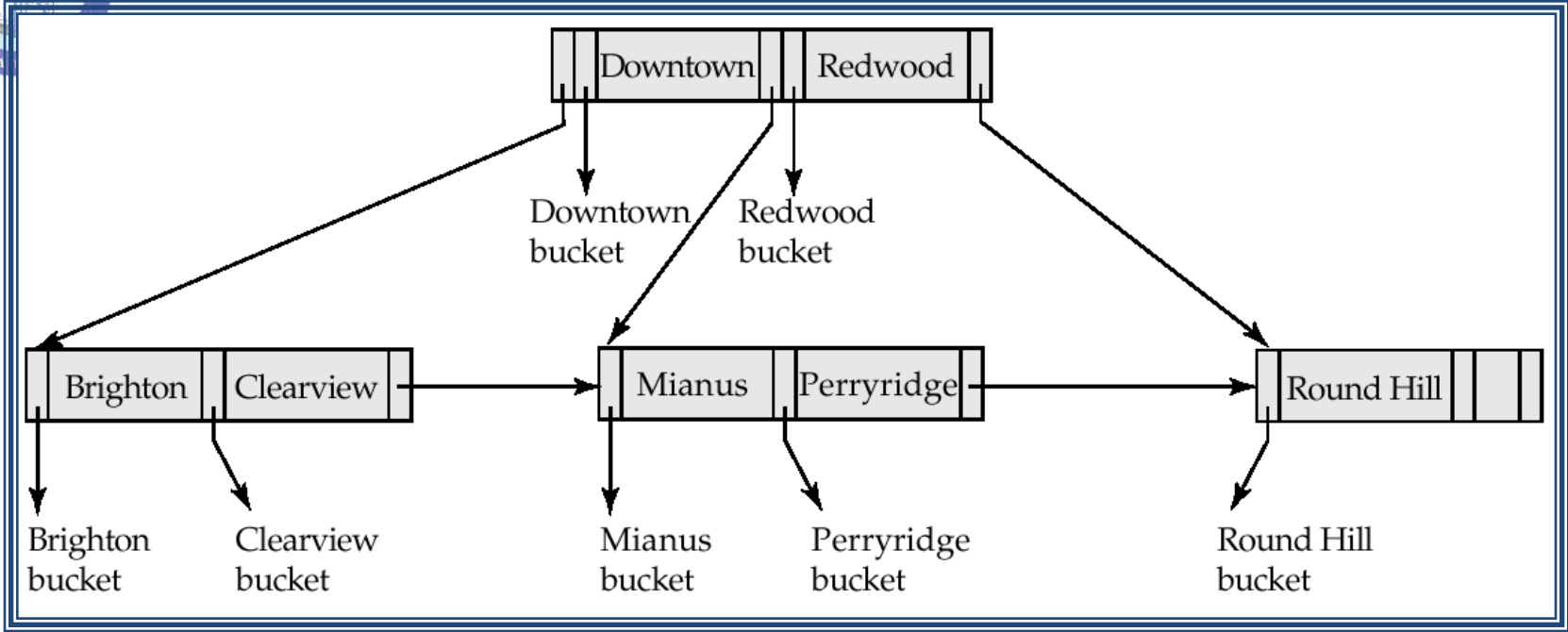


(b)

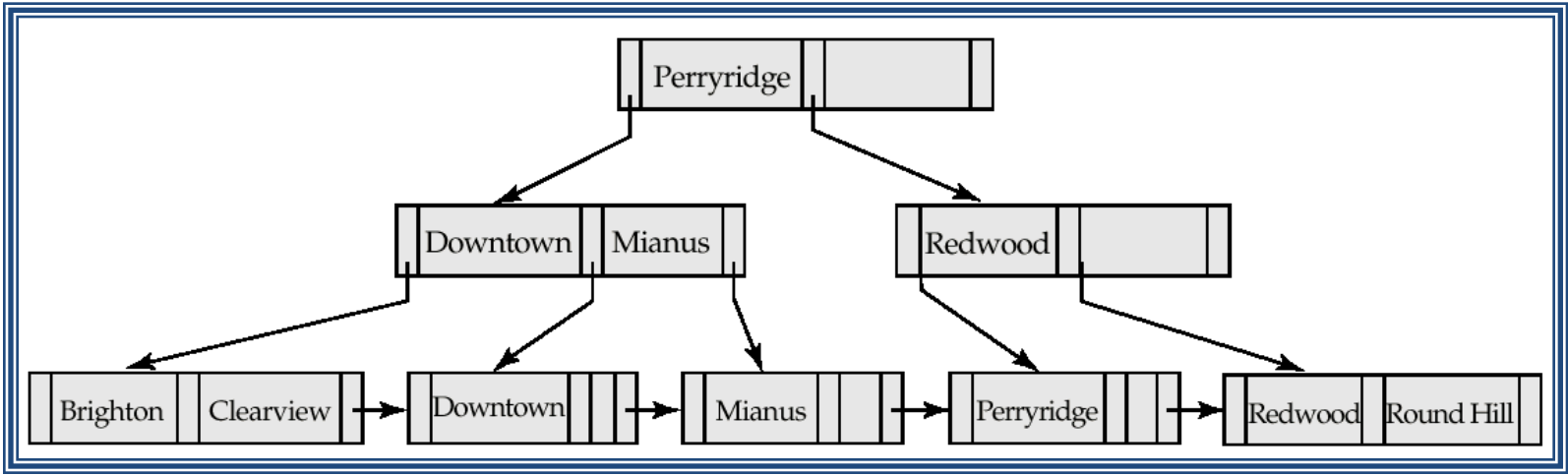
- في العقد الداخلية (غير الأوراق) - المؤشرات B_i تُشير نحو رزمة أو تسجيلة من الملف الأساسي.



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



B-tree في الأعلى و B+-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 تسجيلية أو أكثر (عادة تساوي كتلة disk block أو أكثر).
- يُطبق تابع التقطيع **hash function** على قيمة مفتاح البحث لمعرفة الرزمة الحاوية على التسجيلية.
- مثال : تخزين علاقة account في ملف تقطيع مقسم إلى ١٠ رزم باستخدام مفتاح البحث branch-name وتابع التقطيع h بحيث :

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

وبالتالي:

$$h(\text{Perryridge}) = 5$$

$$h(\text{Round Hill}) = 3$$

$$h(\text{Brighton}) = 3$$

bucket 0			bucket 5		
			A-102	Perryridge	400
			A-201	Perryridge	900
			A-218	Perryridge	700
bucket 1			bucket 6		
bucket 2			bucket 7		
			A-215	Mianus	700
bucket 3			bucket 8		
A-217	Brighton	750	A-101	Downtown	500
A-305	Round Hill	350	A-110	Downtown	600
bucket 4			bucket 9		
A-222	Redwood	700			

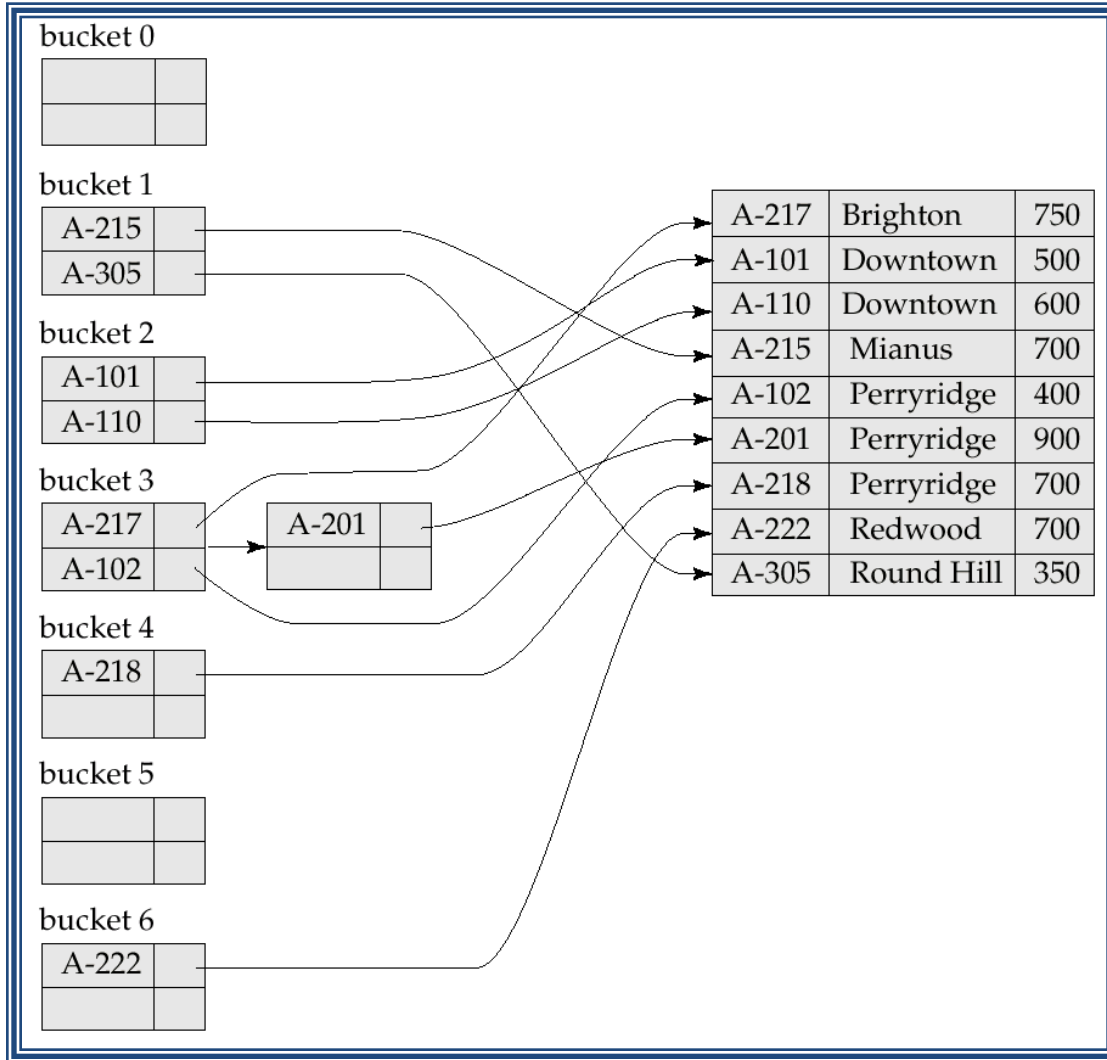


الأدلة المقطعة Hash Indices

- 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





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 .
 - 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 ال- Bitmap

- فهرس ال-bitmap بشكله البسيط لوصفة هو عبارة عن قائمة من ال-bitmap لكل قيمة مختلفة من قيم الوصفة.
- لكل قيمة bitmap عدد من البتات يساوي عدد التسجيلات في العلاقة.
- من أجل القيمة v للوصفة تكون قيمة ال-bitmap الموافقة مؤلفة من مجموعة من البتات بحيث تكون البت الخاصة بالتسجيلة رقم i : تساوي 1 إذا كانت قيمة الوصفة في هذه التسجيلة تساوي v ، وإلا تساوي 0.

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

1	0	0	1	0
---	---	---	---	---

f

0	1	1	0	1
---	---	---	---	---

Bitmaps for income-level

L1

1	0	1	0	0
---	---	---	---	---

L2

0	1	0	0	0
---	---	---	---	---

L3

0	0	0	0	1
---	---	---	---	---

L4

0	0	0	1	0
---	---	---	---	---

L5

0	0	0	0	0
---	---	---	---	---



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

$$100110 \text{ AND } 110011 = 100010$$

$$100110 \text{ OR } 110011 = 110111$$

$$\text{NOT } 100110 = 011001$$