- **مقدمة من الكتاب :**
  - في هذا الفصل نفترض أن **file** موجود بالفعل في بعض **primary organization** بشكل (unordered, ordered, or hashed organizations) كما ذُكر في (Chapter 17).
  - سنقوم بوصف هياكل وصول إضافية مساعدة تسمى **indexes**، والتي تستخدم لتسريع وتسهيل الوصل للبيانات حسب شروط البحث. إذاً **index structures** هي ملفات إضافية على القرص توفر مسارات وصول ثانوية (**secondary access paths**) لتوفير طرق بديلة للوصول إلى السجلات دون التأثير على المكان الفعلي للسجلات في ملف البيانات الأساسي على القرص. وهي تتيح إمكانية الوصول الفعال إلى السجلات إستناداً إلى حقول الفهرسة (**indexing fields**) المستخدمة في إنشاء الفهرس.
  - أكثر أنواع الفهارس انتشارهي:
    - **based on ordered files (single-level indexes)**
      - **primary**
      - **secondary**
      - **clustering**
    - **tree data structures (multilevel indexes, B+-trees)**
  - هناك أنواع أخرى من الفهارس مثل (ما انذكروا في السلايد لكن جت بواجبات وإختبارات سابقه، مهم نرجع للكتاب)
    - **Hash Indexes**
    - **Bitmap Indexes**

- **Indexes as Access Paths:**
  - **A single-level index is an auxiliary file that makes it more efficient to search for a record in the data file.**
  - **The index is usually specified on one field of the file (although it could be specified on several fields)**
  - **One form of an index is a file of entries <field value, pointer to record>, which is ordered by field value**
  - **The index is called an access path on the field.**
  - **The index file usually occupies considerably less disk blocks than the data file because its entries are much smaller**
  - **A binary search on the index yields a pointer to the file record**
  - **Indexes can also be characterized as dense or sparse**
    - **A dense index has an index entry for every search key value (and hence every record) in the data file.**
    - **A sparse (or nondense) index, on the other hand, has index entries for only some of the search values**

- **الفهارس كمسارات وصول**
  - **single-level index** هو عباره عن ملف مساعد يجعل عملية الوصول للبحث عن **record** في ملف البيانات أكثر كفاءة .
  - يتم تحديد الفهرس عادة على حقل (**field**) واحد من الملف (على الرغم من أنه يمكن تحديده ايضاً في عدة حقول)
  - يكون **index** عادةً ب شكل واحد يحتوي على <قيمة الحقل، مؤشر للسجل في الحقل الأساسي>، والتي يتم ترتيبها حسب قيمة الحقل
  - يسمى **index** مسار وصول على الحقل.
  - ملف الفهرس عادة يأخذ مساحة أقل بكثير من ملف البيانات الأصلي لأن مدخلاتها أصغر بكثير
  - الطريقه المستخدمه للبحث على الفهرس هي البحث الثنائي (**binary search**) ونتيجه البحث تكون ايجاد المؤشر الذي يشير إلى سجل الملف الأساسي
  - ويمكن أيضا وصف **indexes** (بنوعيها) بأنها كثيفة أو متفرقة بناء على حجمها
    - يحتوي **dense index [كثيف]** على مؤشر(**pointer**) لكل حقل (وبالتالي كل **record**) في ملف البيانات.
    - **sparse index [متفرق]** هناك مؤشر(**pointer**) لبعض الحقول فقط

- **Indexes as Access Paths (cont.):**
  - Example: **Given the following data file EMPLOYEE(NAME, SSN, ADDRESS, JOB, SAL, ...)**
  - **Suppose that:**
    - **record size R=150 bytes block size B=512 bytes r=30000 records**
  - **Then, we get:**
    - **blocking factor Bfr= B div R= 512 div 150= 3 records/block**
    - **number of file blocks b= (r/Bfr)= (30000/3)= 10000 blocks**
  - **For an index on the SSN field, assume the field size V_{SSN}=9 bytes, assume the record pointer size P_R=7 bytes. Then:**
    - **index entry size RI=(VSSN+ PR)=(9+7)=16 bytes**
    - **index blocking factor BfrI= B div RI= 512 div 16= 32 entries/block**
    - **number of index blocks b= (r/ BfrI)= (30000/32)= 938 blocks**
    - **binary search needs log2bI= log2938= 10 block accesses**
    - **This is compared to an average linear search cost of:**
      - (b/2)= 30000/2= 15000 block accesses
    - **If the file records are ordered, the binary search cost would be:**
      - $\log_2 b = \log_2 30000$= 15 block accesses

- **مثال يوضح الفرق بين عدد block accesses لكل من الملف الأساسي ،، وملف Index لنفس الجدول (جدول الموظفين)**
  - في الملف الأساسي
    - بعد العمليه الحسابيه عدد ال**Blocks = 10000** بمعدل **3 records** لكل **block**
    - بإستخدام البحث **linear search** يكون متوسط الوصول **15000 block accesses**
    - بإستخدام البحث **binary search** إذا كان الملف **ordered** يكون عدد عمليات الوصول **15 block accesses**
  - في ملف ال **index**
    - عدد ال**Blocks = 938** بمعدل **32 entries** لكل **block**
    - ذكرنا انه الطريقه المستخدمه للبحث على **Indexes** هي **binary search** إذاً يكون عدد عمليات الوصول هو **10 block accesses** **وهو أقل بكثير من عدد ال accesses** على الملف الأساسي .

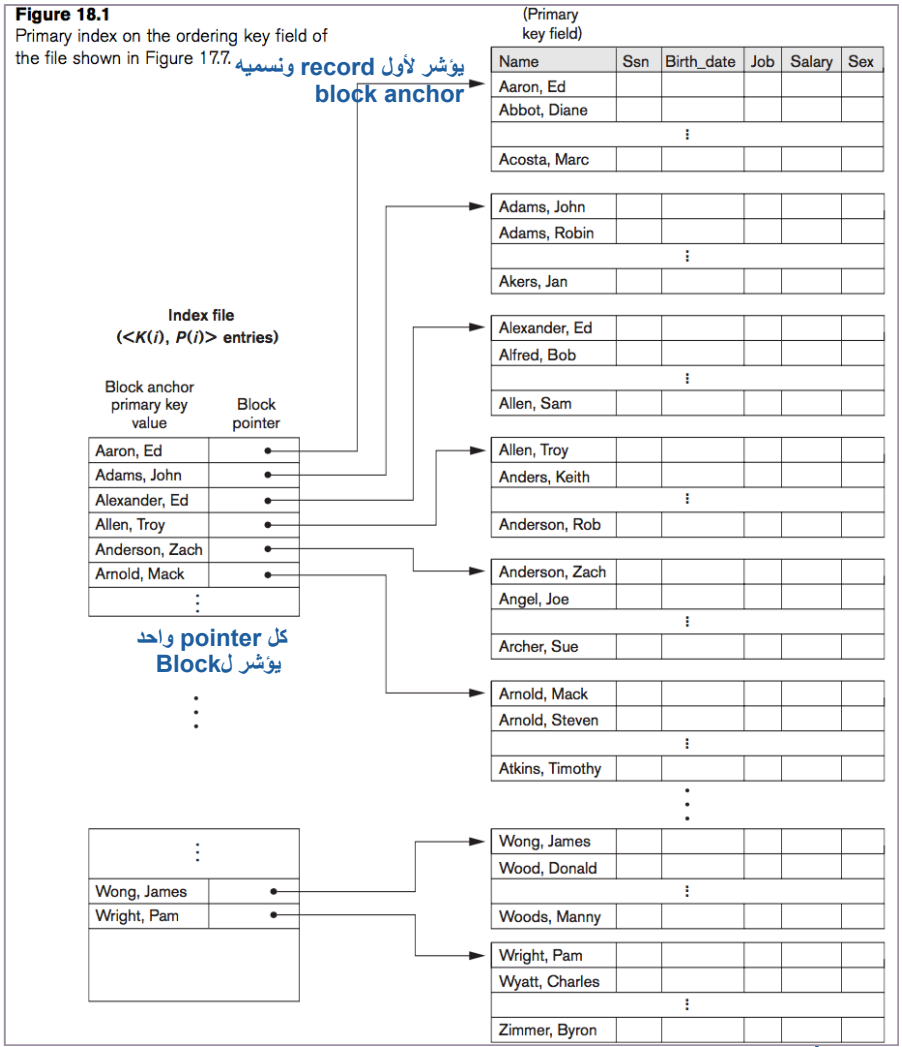
- **Types of Single-Level Indexes**
  - **Primary Index**
    - Defined on an ordered data file
    - The data file is ordered on a key field
    - Includes one index entry for each block in the data file; the index entry has the key field value for the first record in the block, which is called the block anchor
    - A similar scheme can use the last record in a block.
    - A primary index is a nondense (sparse) index, since it includes an entry for each disk block of the data file and the keys of its anchor record rather than for every search value.

- **أنواع Single-Level Indexes**
  - **الفهرس الاساسي Primary Index**

يستخدم هذا النوع عاده للحقول التي تكون (**key field**) لانه أكثر حقل مستخدم في الإستعلامات

- ملف البيانات ضروري يكون مرتب **ordered**
- ترتيب ملف البيانات يكون حسب **key field**
- يكون مُدخل واحد (**pointer**) من ال **index** يشير لكل **block** في ملف البيانات؛ و يكون يشير لأول **record** في كل **block** ويسمى **block anchor**
- بعض الجداول تستخدم آخر **record**
- من تعريف ال **sparse index** قلنا انه المؤشر يشير إلى بعض الحقول وليس جميعها ،، وهنا المؤشر يشير لكل **block** وليس لكل حقل ، ومفتاح البحث هو **block anchor** إذاً **Primary Index** من نوع **sparse** .

Primary Index on the Ordering Key Field



- **Types of Single-Level Indexes**
  - **Clustering Index**
    - Defined on an ordered data file
    - The data file is ordered on a non-key field unlike primary index, which requires that the ordering field of the data file have a distinct value for each record.
    - Includes one index entry for each distinct value of the field; the index entry points to the first data block that contains records with that field value.
    - It is another example of nondense index where Insertion and Deletion is relatively straightforward with a clustering index.

- **أنواع Single-Level Indexes**
  - **Clustering Index**
    - ملف البيانات يكون مرتب **ordered**
    - ترتيب ملف البيانات يكون على حقل غير ال**key field** عكس **Primary Index** ، الأمر الذي يتطلب أن يكون هذا الحقل مرتب في ملف البيانات وله قيمة مميزة لكل **record**.
    - معنى **distinct** هي كل القيم بدون تكرار، كل قيمة عندنا فيها تكرارات يأخذ منها قيمة واحدة فقط ويعمل لها **Index**، المؤشر يشير لأول قيمه في **data block** [لكن ممكن نحصل قيمه آخرى **مو** موجوده في **block** آخر وهنا نشوف مؤشرين لنفس ال**block]**
    - يعتبر نوعه أيضا **nondense index(sparse)** ، لأنه ما أخذ جميع بيانات الحقل وخزنهم في ال **Index** ،، ايضاً الحذف والإدراج يكون بسيط نسبياً في **Clustering Index**

A Clustering Index Example

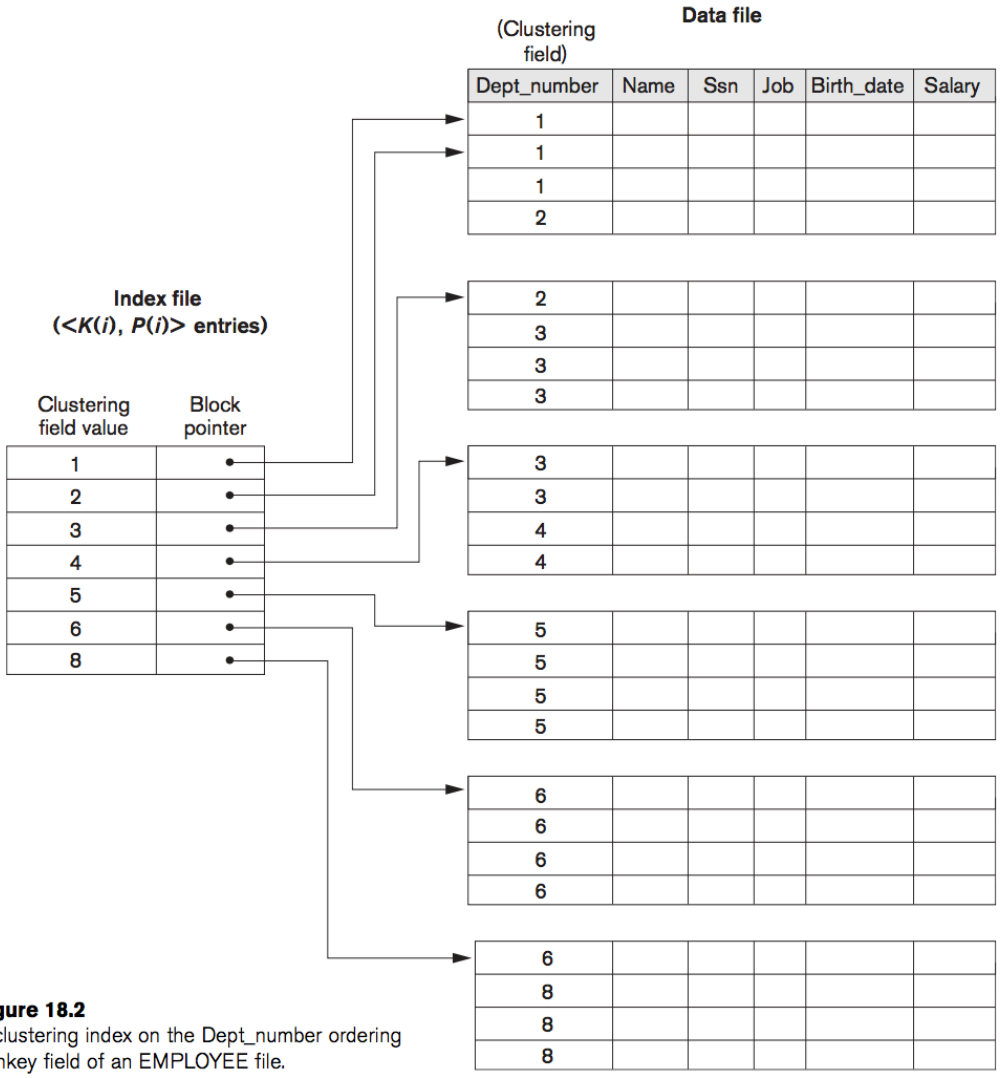
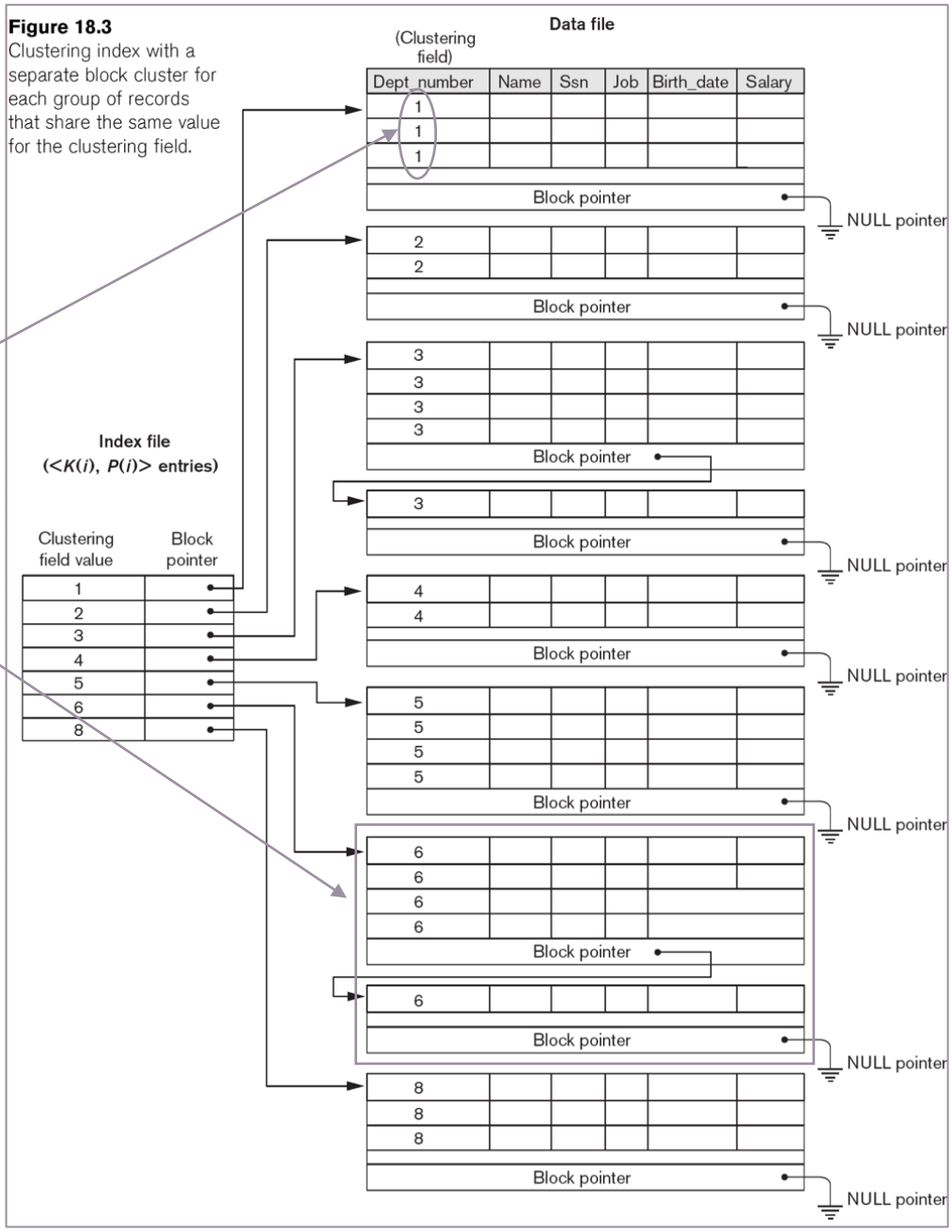


Figure 18.2
A clustering index on the Dept_number ordering nonkey field of an EMPLOYEE file.

Another Clustering Index Example

Figure 18.3
Clustering index with a separate block cluster for each group of records that share the same value for the clustering field.



الفرق عن المثال السابق

- كل القيم المتكرره تكون في **Block** واحد.
- يفضل هذا النوع من التقسيم إذا كان الحذف والإضافه والتعديل أكثر
- هالنوع من التقسيم يكون فيه الحذف والاضافه والتعديل أسهل
- يكون ال**Index** ثابت لايحتاج إلى اعاده تنظيم إذا سوينا عمليات الحذف والاضافه وغيرها
- وجود **NULL pointer** ، عند اكتمال **Block** نضيف **Block** جديد يكون إمتداد لل **Block** السابق ،، فنحل مشكله الإزاحه عند الإضافه.

By: MAHA OTAIBI

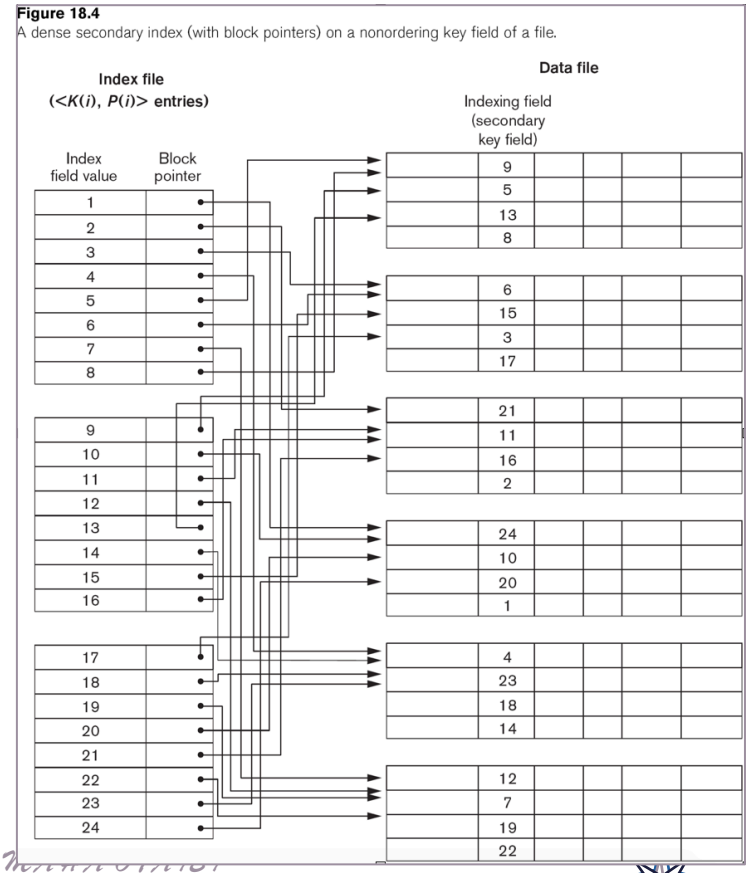
- **Types of Single-Level Indexes**
  - **Secondary Index**
    - **A secondary index provides a secondary means of accessing a file for which some primary access already exists.**
    - **The secondary index may be on a field which is a candidate key and has a unique value in every record, or a non-key with duplicate values.**
    - **The index is an ordered file with two fields.**
      - **The first field is of the same data type as some non-ordering field of the data file that is an indexing field.**
      - **The second field is either a block pointer or a record pointer.**
      - **There can be many secondary indexes (and hence, indexing fields) for the same file.**
    - **Includes one entry for each record in the data file; hence, it is a dense index**

- **أنواع Single-Level Indexes**
  - **الفهارس الثانويه Secondary Index**
    - يوفر ال **Secondary Index** وسيلة ثانوية للوصول إلى ملف الذي يوجد فيه بالفعل **Primary Index**. بمعنى إذا احتجنا نعمل **Index** ثاني راح يكون **Secondary**، لكن الأولويه في البحث دائماً ل**Primary Index** .
    - في ال **Secondary** مايكون عندنا شروط ، ممكن ينعمل على حقل يكون **candidate key** له قيمة فريده لكل **record** أو **non-key** وقيمه مكرره .
    - الحقول تكون غير مرتبه ، لكن ال **Index** يكون مرتب وبه حقلين
      - الحقل الأول(**index field**) يشيرإلى حقل غير مرتب في ملف البيانات من نفس نوع البيانات .
      - الحقل الثاني هو إما يؤشر ل**block** أو يؤشر ل**record**.
      - يمكن أن يكون عندنا أكثر من **Secondary Index** في نفس الملف.
    - يتضمن مُدخل واحد لكل **record** في ملف البيانات؛ وبالتالي، فهو مؤشر كثيف (**dense index**)

Example of a Dense Secondary Index

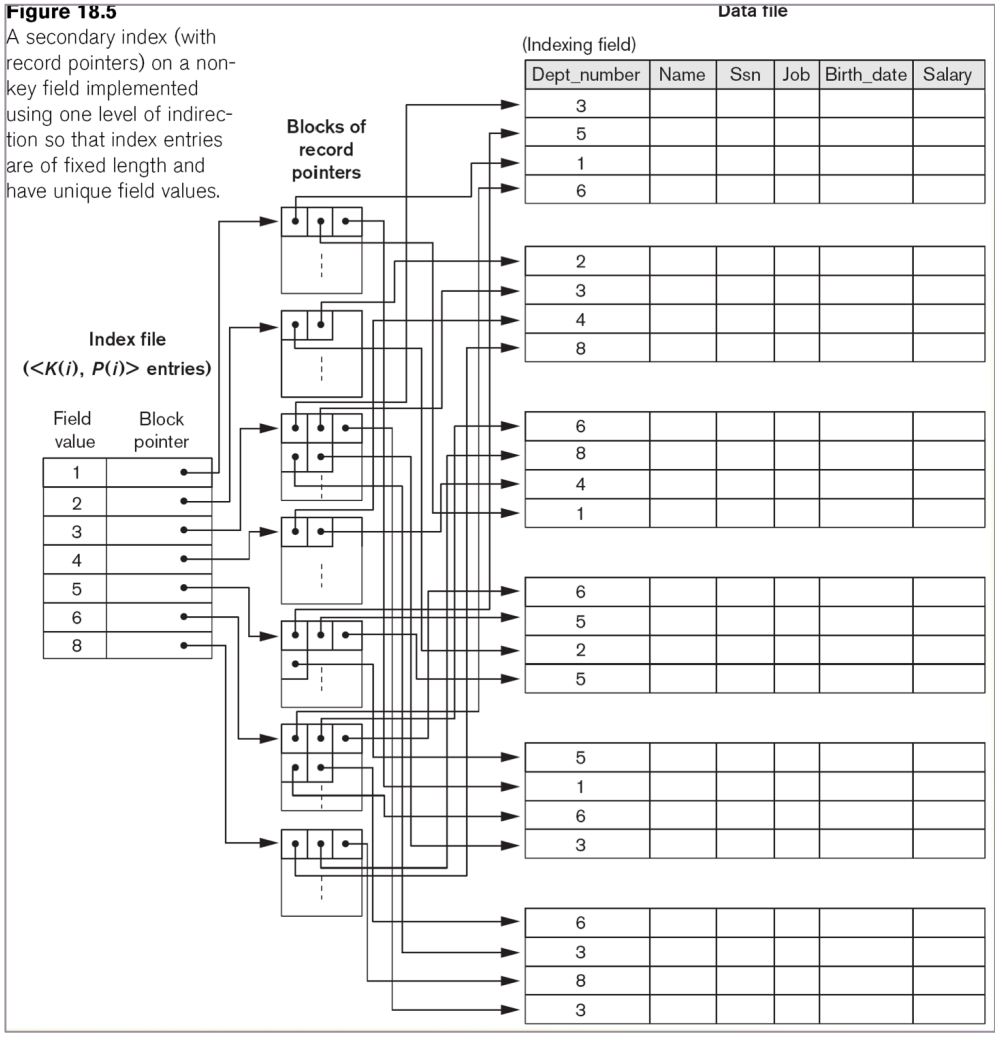
- البيانات في ال **Index**ال مرتبه بينما هي غير مرتبه في ملف البيانات
- كل مؤشر **Index** يشير إلى **record** إذا نوعه **dense**

Figure 18.4
A dense secondary index (with block pointers) on a nonordering key field of a file.



Example of a Secondary Index

- نستخدم هالنوع إذا كان عندي تكرار في البيانات فيحل لنا مشكله عدم الترتيب
- يصير كل **Block pointer** يشير إلى **Block** ثاني أسمه (**Blocks of record pointer**)



Properties of Index Types

Table 18.2 Properties of Index Types

Type of Index	Number of (First-level) Index Entries (عدد المدخلات في الاندكس)	Dense or Nondense (Sparse)	Block Anchoring on the Data File
Primary	Number of blocks in data file (يساوي عدد البلوك في ملف البيانات)	Nondense	Yes
Clustering	Number of distinct index field values (يساوي عدد القيم الغير مكرره في ملف البيانات)	Nondense	Yes/no[a]
Secondary (key)	Number of records in data file (يساوي عدد الريكورد في ملف البيانات)	Dense	No
Secondary (nonkey)	Number of records[b] or number of distinct index field values[c] (يساوي عدد الريكورد الغير مكرره في ملف البيانات)	Dense or Nondense	No

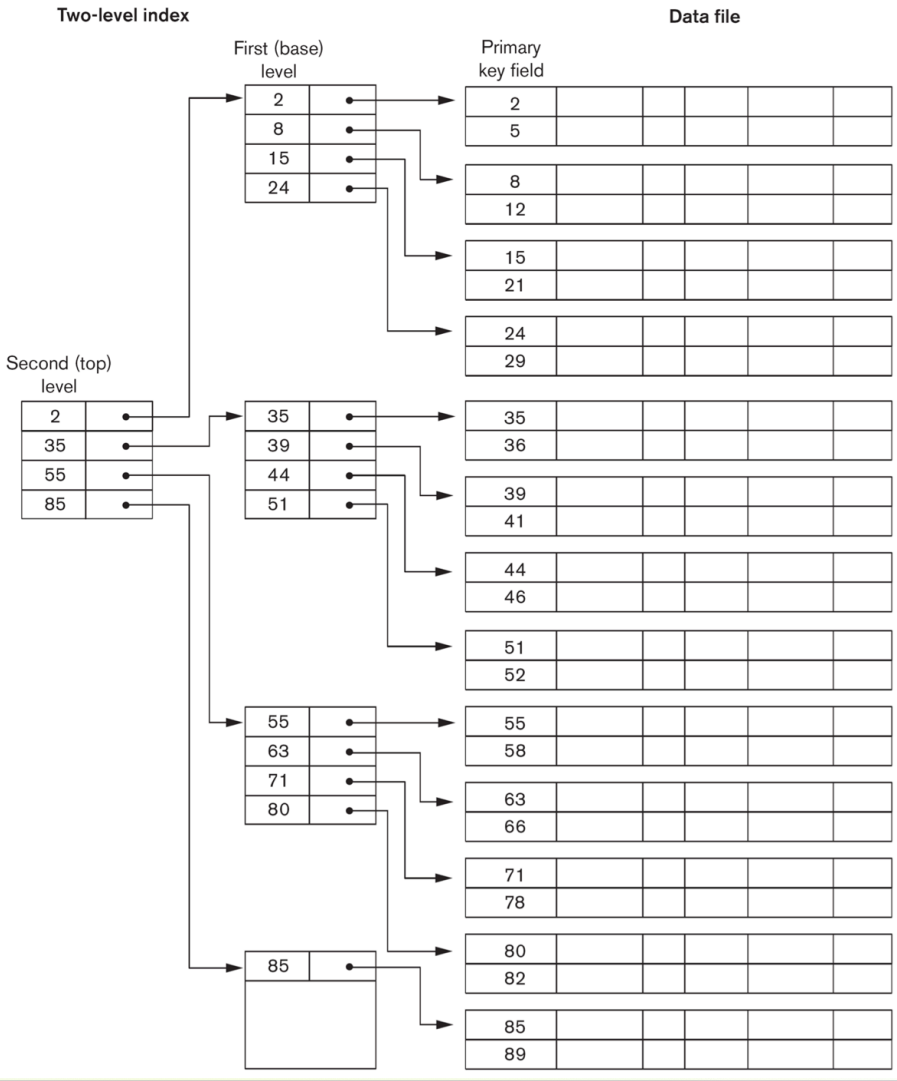
---

- **Multi-Level Indexes**
  - **Because a single-level index is an ordered file, we can create a primary index to the index itself;**
    - **In this case, the original index file is called the first-level index and the index to the index is called the second-level index.**
  - **We can repeat the process, creating a third, fourth, ..., top level until all entries of the top level fit in one disk block**
  - **A multi-level index can be created for any type of first-level index (primary, secondary, clustering) as long as the first-level index consists of more than one disk block**

- **Multi-Level Indexes**
  - لأن **single-level index** هو ملف مرتب ، يمكننا إنشاء **Primary Index** للفهرس نفسه.
  - في هذه الحالة، يسمى ملف الفهرس الأصلي **single-level index** ويسمى فهرس الفهرس **the second-level index**.
  - يمكننا تكرار العملية، وإنشاء الثالث والرابع، ...، حتى يكون أعلى **level** فيه عدد ال**Block** واحد فقط، هنا تتوقف العمليه.
  - يمكن إنشاء فهرس متعدد المستويات لأي نوع من فهرس المستوى الأول بإنواعه(**primary, secondary, clustering**) طالما أن فهرس المستوى الأول يتكون من أكثر من **Block** واحد.

A Two-Level Primary Index

Figure 18.6
A two-level primary index resembling ISAM (Indexed Sequential Access Method) organization.



- **Multi-Level Indexes**
  - **Such a multi-level index is a form of search tree**
    - **However, insertion and deletion of new index entries is a severe problem because every level of the index is an ordered file.**

- **Multi-Level Indexes**
  - ال **multi-level index** هو شكل من أشكال ال **search tree** .
  - ومع ذلك، إدراج وحذف إدخالات الفهرس الجديد مشكلة تواجهنا (حيث يتم التعديل على اكثر من **Level**) وكل مستوى من الفهرس هو ملف **ordered**.

A Node in a Search Tree with Pointers to Subtrees Below It

Figure 18.8
A node in a search tree with pointers to subtrees below it.

P=pointer
K=value
X=value for Subtree

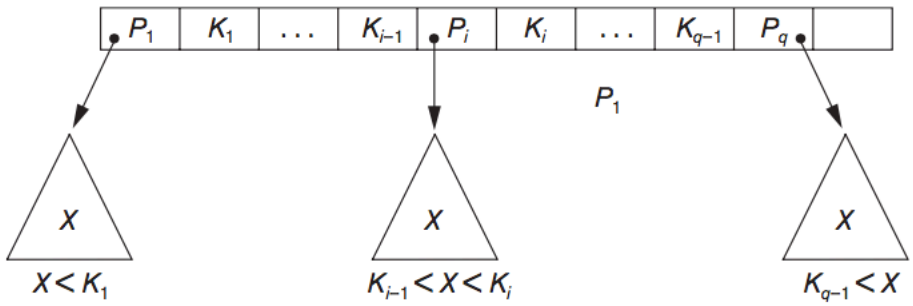
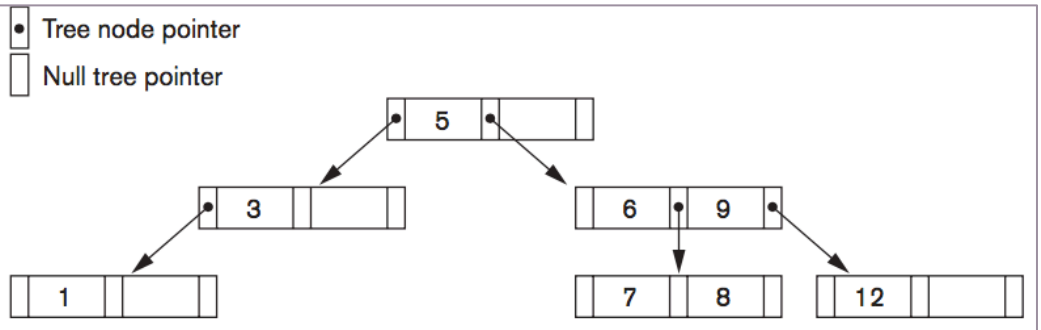


Figure 18.9
A search tree of order $p = 3$.



- **Dynamic Multilevel Indexes Using B-Trees and B+-Trees**
  - Most multi-level indexes use B-tree or B+-tree data structures because of the insertion and deletion problem
    - This leaves space in each tree node (disk block) to allow for new index entries
  - These data structures are variations of search trees that allow efficient insertion and deletion of new search values.
  - In B-Tree and B+-Tree data structures, each node corresponds to a disk block
  - Each node is kept between half-full and completely full
  - An insertion into a node that is not full is quite efficient
    - If a node is full the insertion causes a split into two nodes
  - Splitting may propagate to other tree levels
  - A deletion is quite efficient if a node does not become less than half full
  - If a deletion causes a node to become less than half full, it must be merged with neighboring nodes

- **Dynamic Multilevel Indexes Using B-Trees and B+-Trees**
  - معظم الفهارس المتعددة المستويات تستخدم هياكل بيانات من نوع **B-tree** أو **B+-tree data** بسبب مشكلة الإدراج والحذف
  - تسمح بإضافه **new index entries** من خلال إضافه **node** جديدة لـ**tree**
  - هياكل البيانات **(B-tree)search trees** تسمح الإدراج الفعال وحذف قيم البحث الجديدة.
  - في **B-Tree** و **B+-Tree**، كل **node** يشير إلى **block**.
  - كل **node** تكون بين نصف ممتلئ ممتلئ تماما ، حتي نقدر نتوسع ونضيف قيم جديدة.
  -
  - الإدراج في **node** غير ممتلئه يعتبر فعال جدا، لإنه ماراح يعمل إزاحه.
  - إذا كانت **node** ممتلئة يؤدي الإدراج إلى الانقسام لـ**node 2.**
  - أحياناً الإ نقسام يؤدي إلى **Level** أخرى من **tree**
  - الحذف يكون فعال جدا إذا لم تصبح ال **node** أقل من نصف ممتلئه. لأنه إذا كانت أقل راح يكون في دمج مع **node** مجاوره وهذا يأخذ وقت أطول.
  - إذا تسبب الحذف في أن تصبح ال **node** أقل من نصف ممتلئه، يجب أن يتم دمجها مع **node** المجاورة.

- **Difference between B-tree and B+-tree**
  - In a B-tree, pointers to data records exist at all levels of the tree
  - In a B+-tree, all pointers to data records exists at the leaf-level nodes
  - A B+-tree can have less levels (or higher capacity of search values) than the corresponding B-tree

- **الفرق بين B-tree وَ B+-tree**
  - في **B-tree** كل **level** يحتوي **على pointers** تشير إلى ال**records**
  - في **B+-tree** كل ال**pointers** موجوده في **leaf-level**
  - في **B+-tree** عدد مستوياتها **level** أقل (وبالتالي قدرة البحث عن القيم أعلى) من **B-tree.**

B-tree Structures

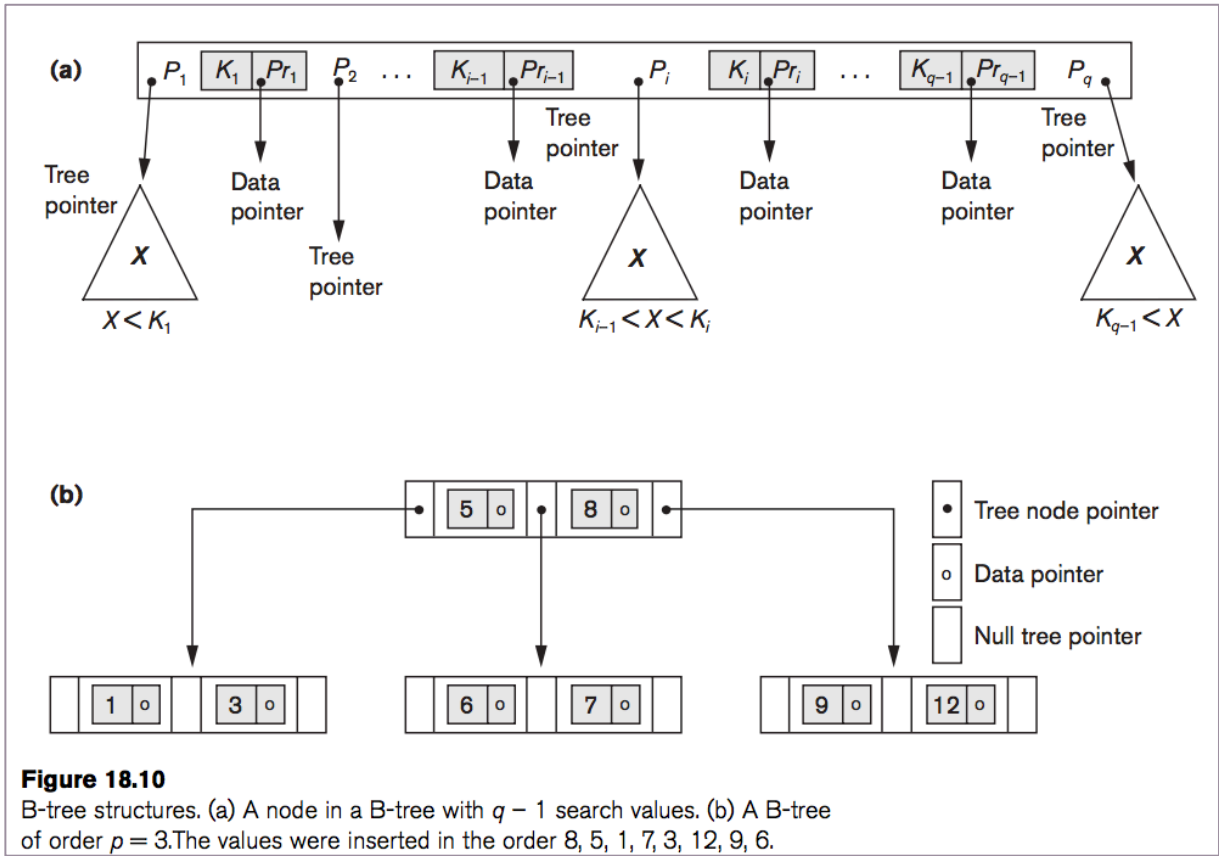
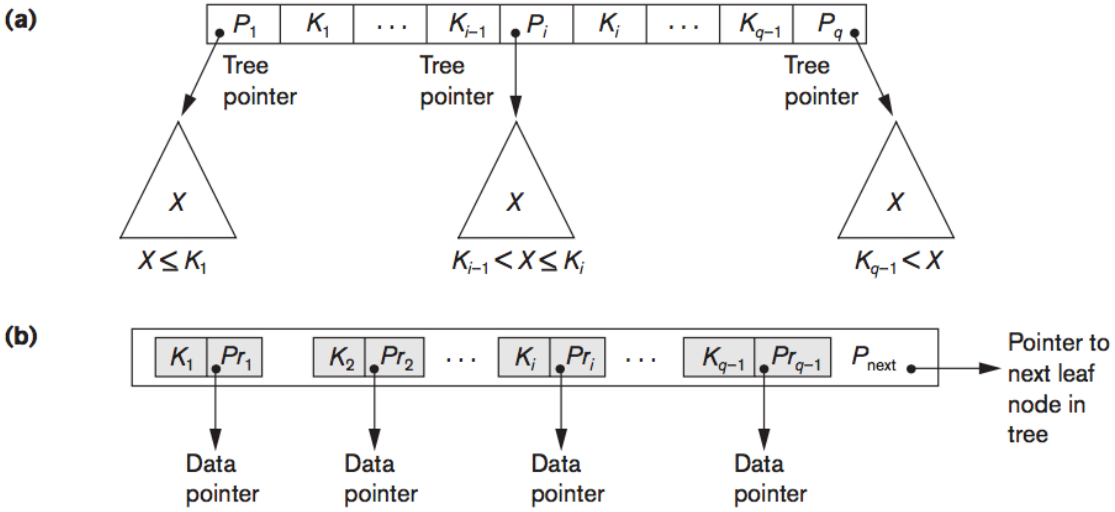


Figure 18.10
B-tree structures. (a) A node in a B-tree with $q - 1$ search values. (b) A B-tree of order $p = 3$.The values were inserted in the order 8, 5, 1, 7, 3, 12, 9, 6.

The Nodes of

Figure 18.11
The nodes of a B+-tree. (a) Internal node of a B+-tree with $q - 1$ search values. (b) Leaf node of a B+-tree with $q - 1$ search values and $q - 1$ data pointers.



Example of an Insertion in a B+-tree

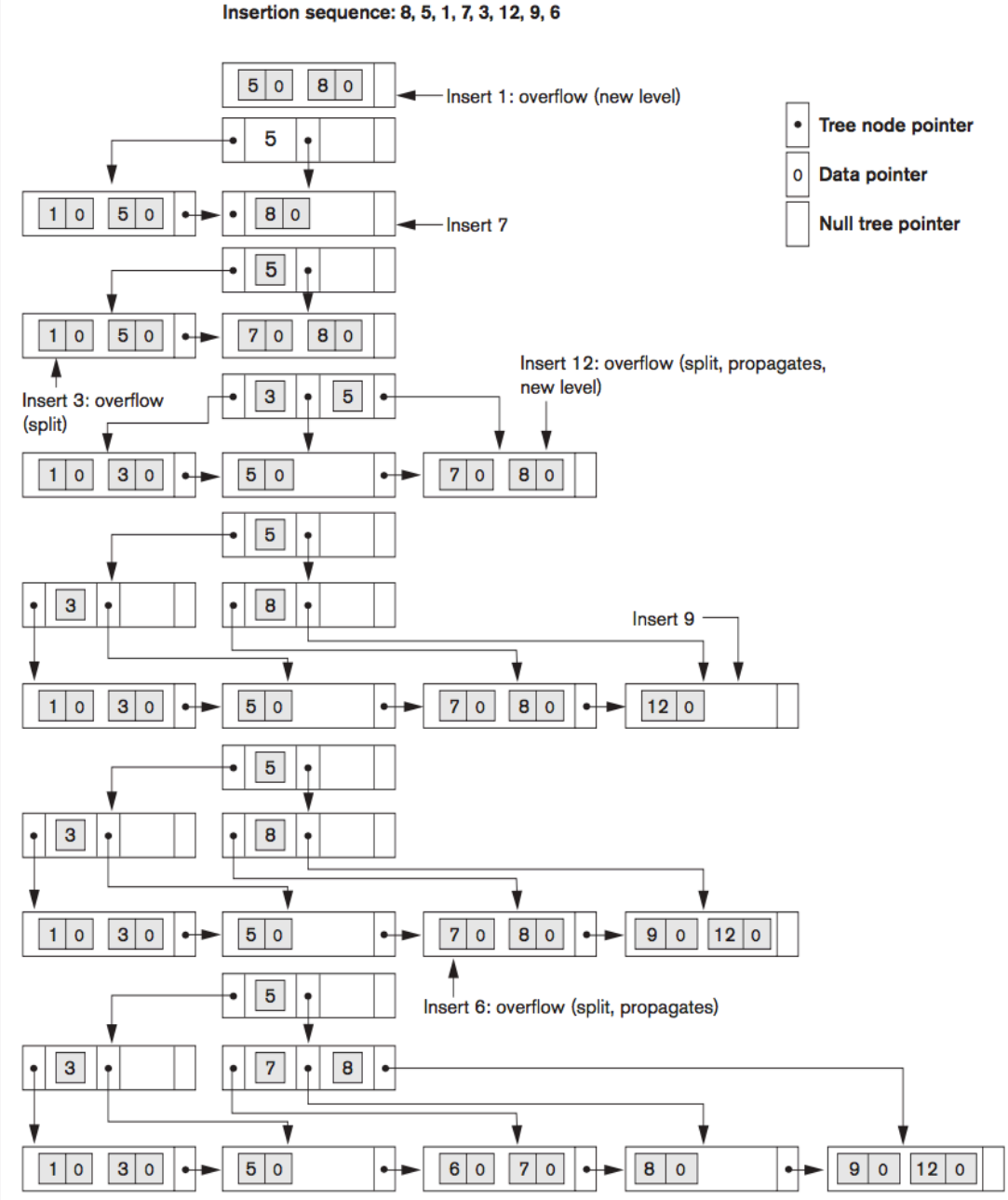


Figure 18.12
An example of insertion in a B+-tree with $p = 3$ and $p_{leaf} = 2$.

Example of a Deletion in a B+-tree

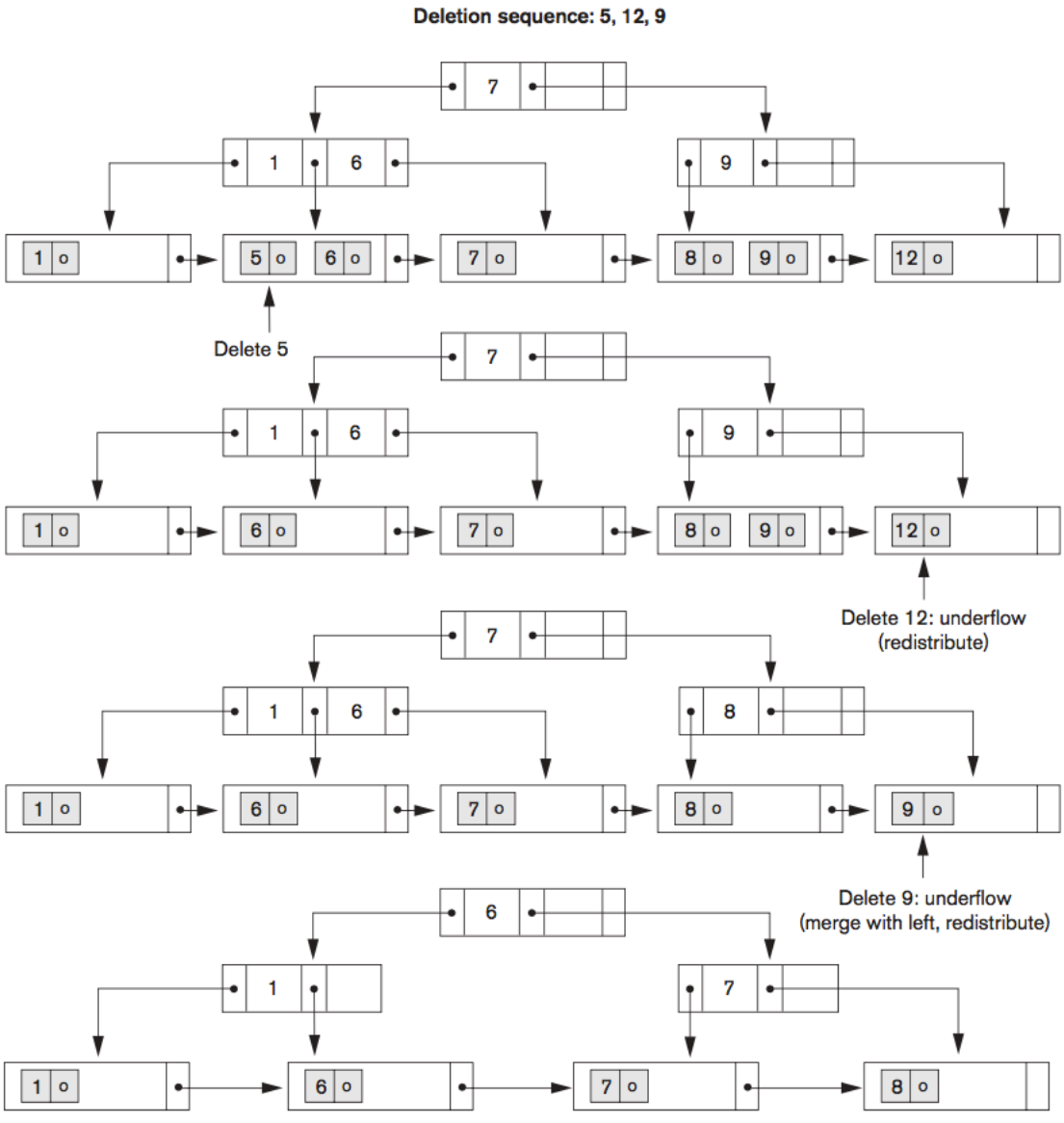


Figure 18.13
An example of deletion from a B+-tree.

- **Summary**
  - Types of Single-level Ordered Indexes
    - Primary Indexes
    - Clustering Indexes
    - Secondary Indexes
  - Multilevel Indexes
  - Dynamic Multilevel Indexes Using B-Trees and B+-Trees
  - Indexes on Multiple Keys