Physical Organization: SQL Server

Leggere Cap 7 Riguzzi et al. Sistemi Informativi

Tables

- Tables and indexes are stored as a collection of 8 KB pages
- A table is divided in one or more **partitions**
- Each partition contains data rows in either a **heap** or a **clustered table**.
- The pages of the heap or clustered index are managed in one or more **allocation units**, depending on the column types in the data rows.

Tables

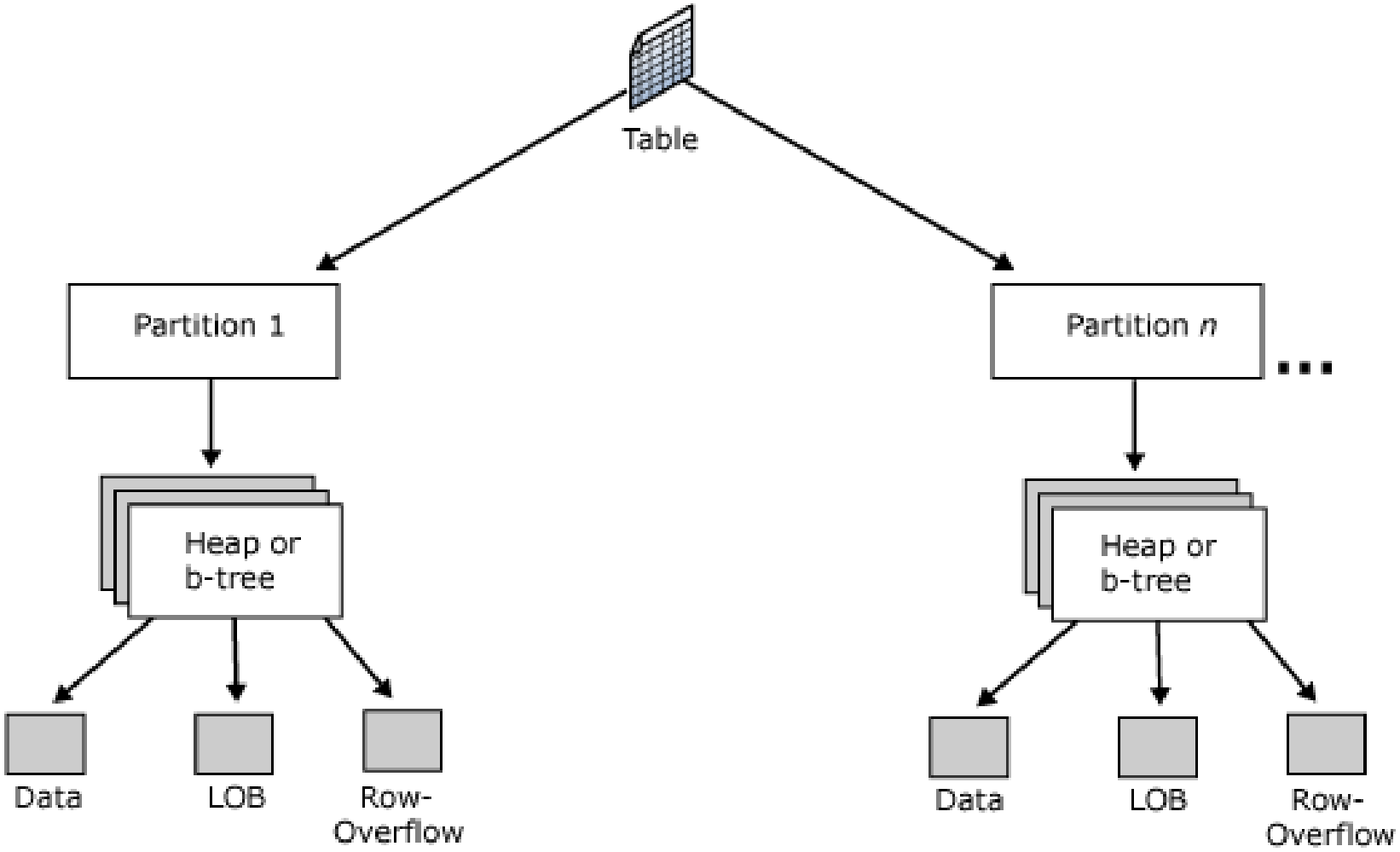


Table
Partition 1
Partition n
Heap or b-tree
Heap or b-tree
Data
LOB
Row-Overflow
Data
LOB
Row-Overflow

Partitions

- Table and index pages are divided in one or more partitions.
- By default, a table or index has only one partition that contains all the table or index pages. The partition resides in a single filegroup.
- When a table or index uses multiple partitions, the data is partitioned horizontally so that groups of rows are mapped into individual partitions, based on a specified column.

Partitions

- The partitions can be put on one or more filegroups in the database. The table or index is treated as a single logical entity when queries or updates are performed on the data.

Organization of a Partition

- SQL Server tables use one of two methods to organize their data pages within a partition:
- **Clustered tables**: tables that have a clustered index. The data rows are stored in order based on the clustered index key. The clustered index is implemented as a B+tree index.
  - The pages in each level of the index, including the data pages in the leaf level, are linked in a doubly-linked list.
- **Heaps**: tables that have no clustered index. The data rows are not stored in any particular order, and there is no particular order to the sequence of the data pages. The data pages are not linked in a linked list.

Allocation Units

- An allocation unit is a collection of pages within a heap or B+tree used to manage data based on their page type.

Allocation unit type	Is used to manage
IN_ROW_DATA	Data or index rows that contain all data, except large object (LOB) data. Pages are of type Data or Index.
LOB_DATA	Large object data stored in one or more of these data types: **text**, **ntext**, **image**, **xml**, **varchar(max)**, **nvarchar(max)**, **varbinary(max)**, or CLR user-defined types (CLR UDT). Pages are of type Text/Image.
ROW_OVERFLOW_DATA	Variable length data stored in **varchar**, **nvarchar**, **varbinary**, or **sql_variant** columns that exceed the 8,060 byte row size limit. Pages are of type Data.

Heaps

- A heap stores a table without a clustered index.
- When a heap has multiple partitions, each partition has a heap structure that contains the data for that specific partition.
- At a minimum, each heap will have one IN_ROW_DATA allocation unit per partition. The heap may also have one LOB_DATA allocation unit per partition and one ROW_OVERFLOW_DATA allocation unit per partition.

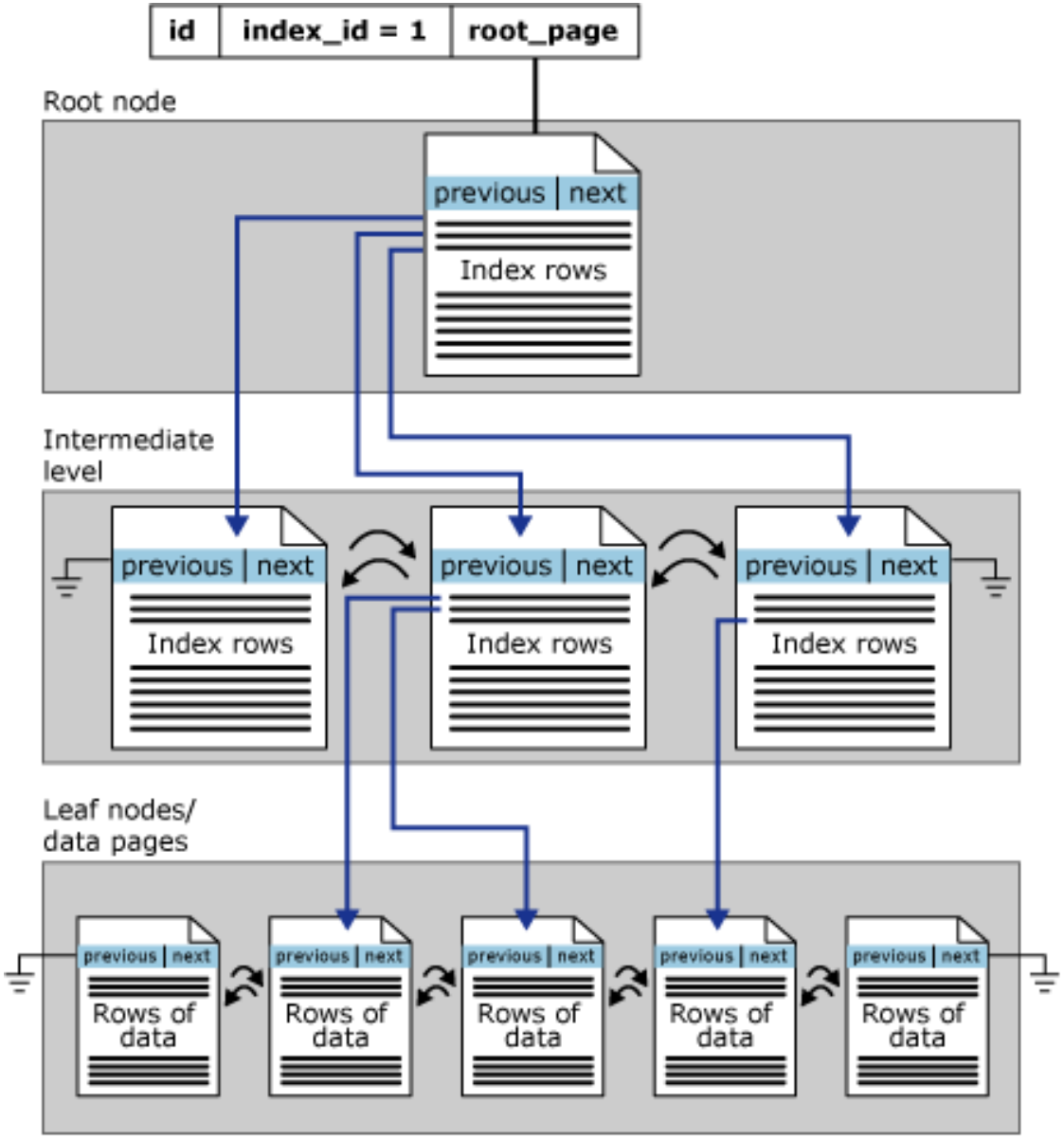
Clustered Tables

- They are organized as B+-trees
- One page per node
- The leaf nodes contain the data pages of the underlying table.
- Thus the data is stored inside the clustered index
- The pages in each level of the index are linked in a doubly-linked list.
- When a clustered index has multiple partitions, each partition has a B+-tree structure that contains the data for that specific partition.

Clustered Tables

- The pages in the data chain and the rows in them are ordered on the value of the clustered index key.
- All inserts are made at the point where the key value in the inserted row fits in the ordering sequence among existing rows.
- At a minimum, each clustered index will have one IN_ROW_DATA allocation unit per partition. The clustered index may also have one LOB_DATA allocation unit per partition and one ROW_OVERFLOW_DATA allocation unit per partition.

A Clustered Table in a Single Partition



Nonclustered Indexes

- Same B+-tree structure as clustered tables but
  - The leaf layer of a nonclustered index is made up of index pages instead of data pages.
  - The data rows of the underlying table are not sorted and stored in order based on their nonclustered keys
- When a nonclustered index has multiple partitions, each partition has a B+-tree structure that contains the data for that specific partition.

Nonclustered Indexes

- Nonclustered indexes can be defined on a table or view with a clustered index or a heap.
- Each index row in the leaves of the nonclustered index contains the key value and a row locator. This locator points to the data row in the clustered index or heap having the key value.
- At a minimum, each nonclustered index will have one IN_ROW_DATA allocation unit per partition. The unclustered index may also have one LOB_DATA allocation unit per partition and one ROW_OVERFLOW_DATA allocation unit per partition.

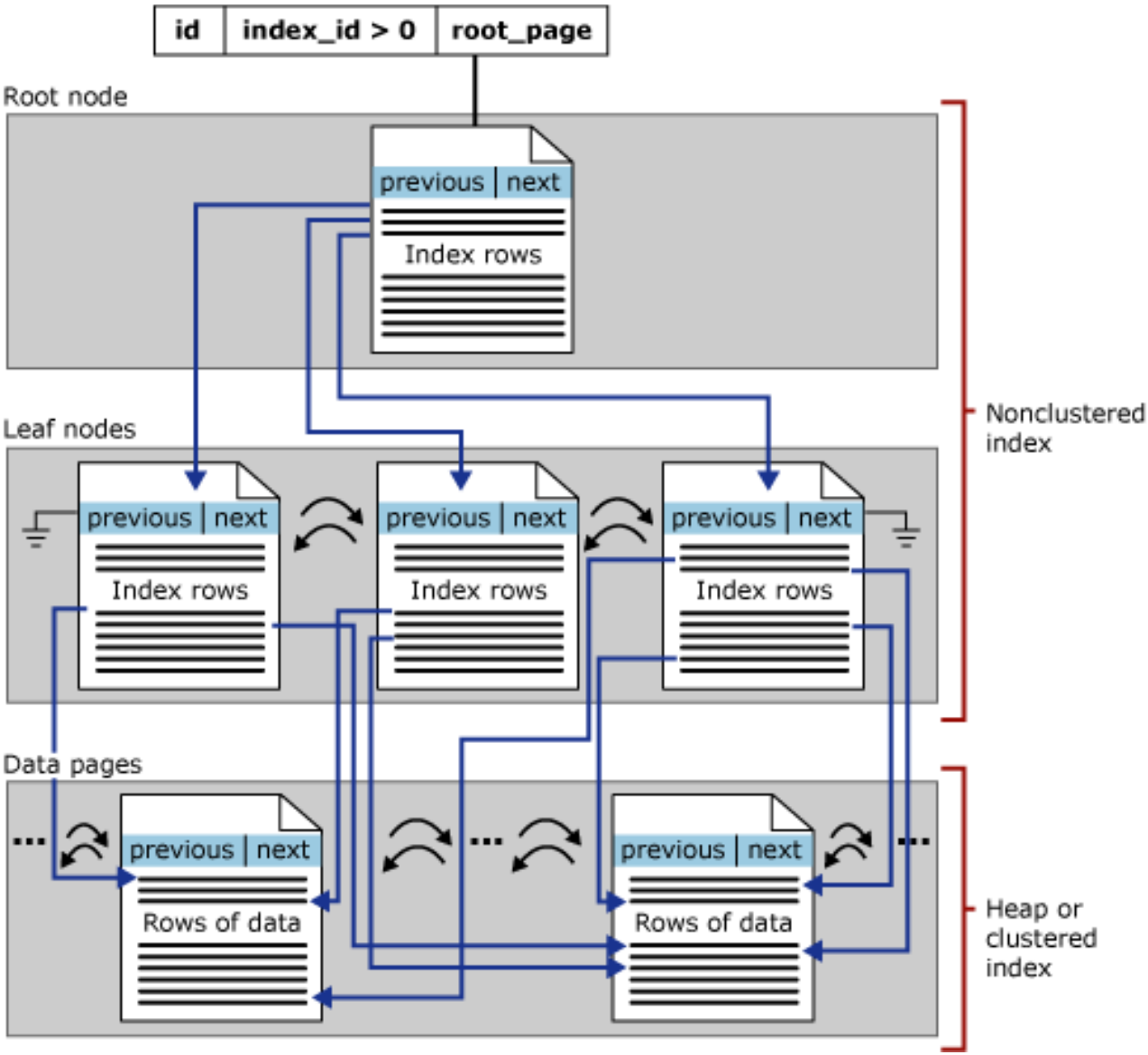
Row Locators

- If the table is a heap the row locator is a pointer to the row. The pointer is built from the file identifier (ID), page number, and number of the row on the page. The whole pointer is known as a Row ID (RID).
- If the table has a clustered index, the row locator is the clustered index key for the row.
  - If the clustered index is not a unique index, SQL Server 2005 makes any duplicate keys unique by adding an internally generated value called a **uniqueifier**. This four-byte value is not visible to users.
  - SQL Server retrieves the data row by searching the clustered index using the clustered index key stored in the leaf row of the nonclustered index.

Nonclustered Indexes

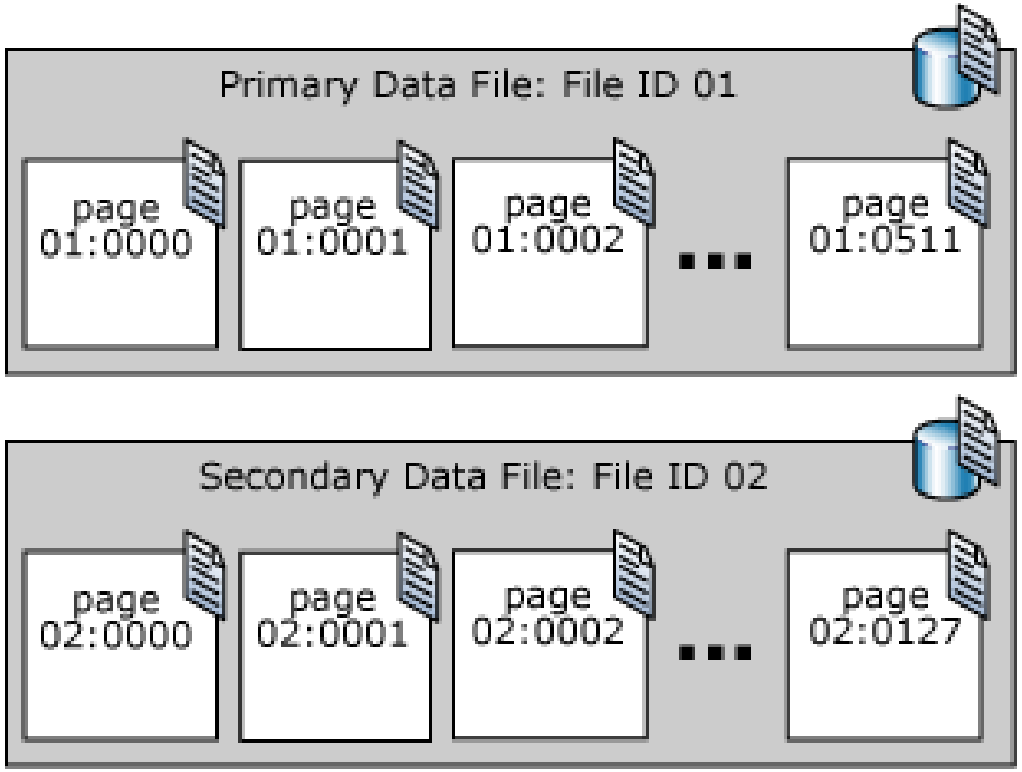
- When a nonclustered index has multiple partitions, each partition has a B+tree structure that contains the index rows for that specific partition.

A Nonclustered Index in a Single Partition



Pages

•Pages in file are numbered sequentially, starting from 0



Pages

- The first page of a file contains information about the attributes of the file
- Other pages at the beginning of the file can be used for containing system information, such as allocation maps