Indexes

Objectives

At the end of this sub-module, you should be able to:

- Define indexes
- Recognize the types of indexes
- Define clustered and non clustered index
- Application of indexes

Introduction to Indexes

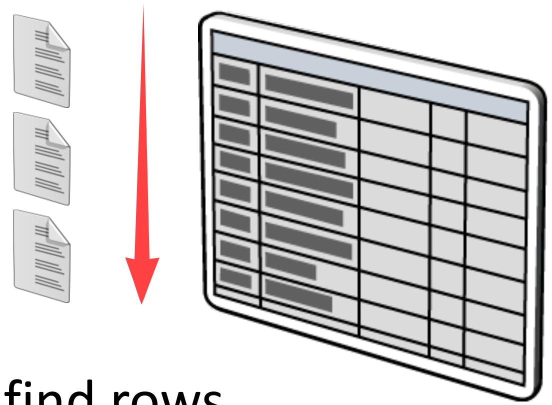
- All SQL server indexes are b-trees. There is a single root page at the top of the tree, branching out into N number of pages at each intermediate level until it reaches the bottom, or leaf level, of the index

- Why to create an index
  - Speeds up data access
  - Enforces uniqueness of rows

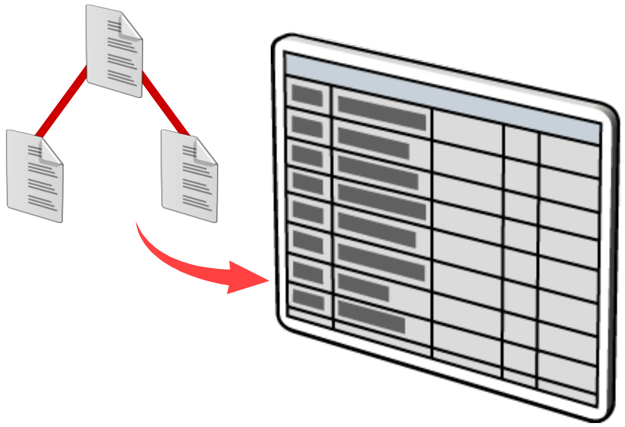
- Why not to create an index
  - Consumes disk space

How SQL Server Accesses Data

- Table scan
  - SQL server reads all table pages



- Index
  - SQL server uses index pages to find rows



Types Of Index

- Indexes are of two types:
  - Clustered Index
  - Non Clustered Index

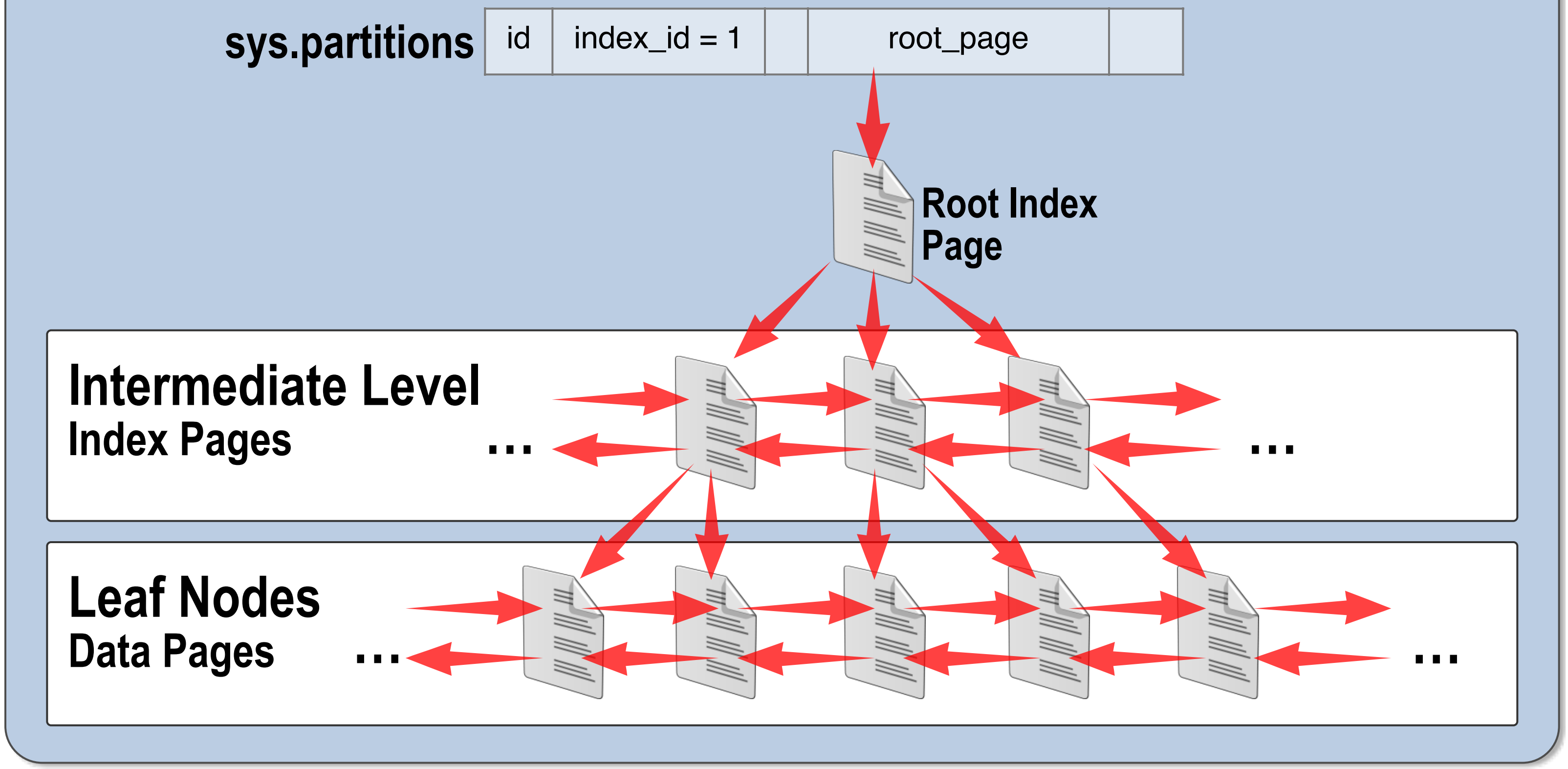
Clustered Index

- A clustered index determines the physical order of data in a table. A clustered index is analogous to a telephone directory, which arranges data by last name .

- Each table can have only one clustered index

Clustered Index (Contd.).

- One clustered index per table
- B-tree stores data pages in order of index key



Guidelines for Clustered Indexes

- Heavily updated tables
  - A clustered index with an identity column keeps updated pages in memory
- Sorting
  - A clustered index keeps the data pre-sorted
- Column length and data type
  - Limit the number of columns
  - Reduce the number of characters
  - Use the smallest data type possible
- Columns that contain a large number of distinct values
- Queries that return a range of values using operators such as between, >, >=, <, and <=

Non Clustered Index

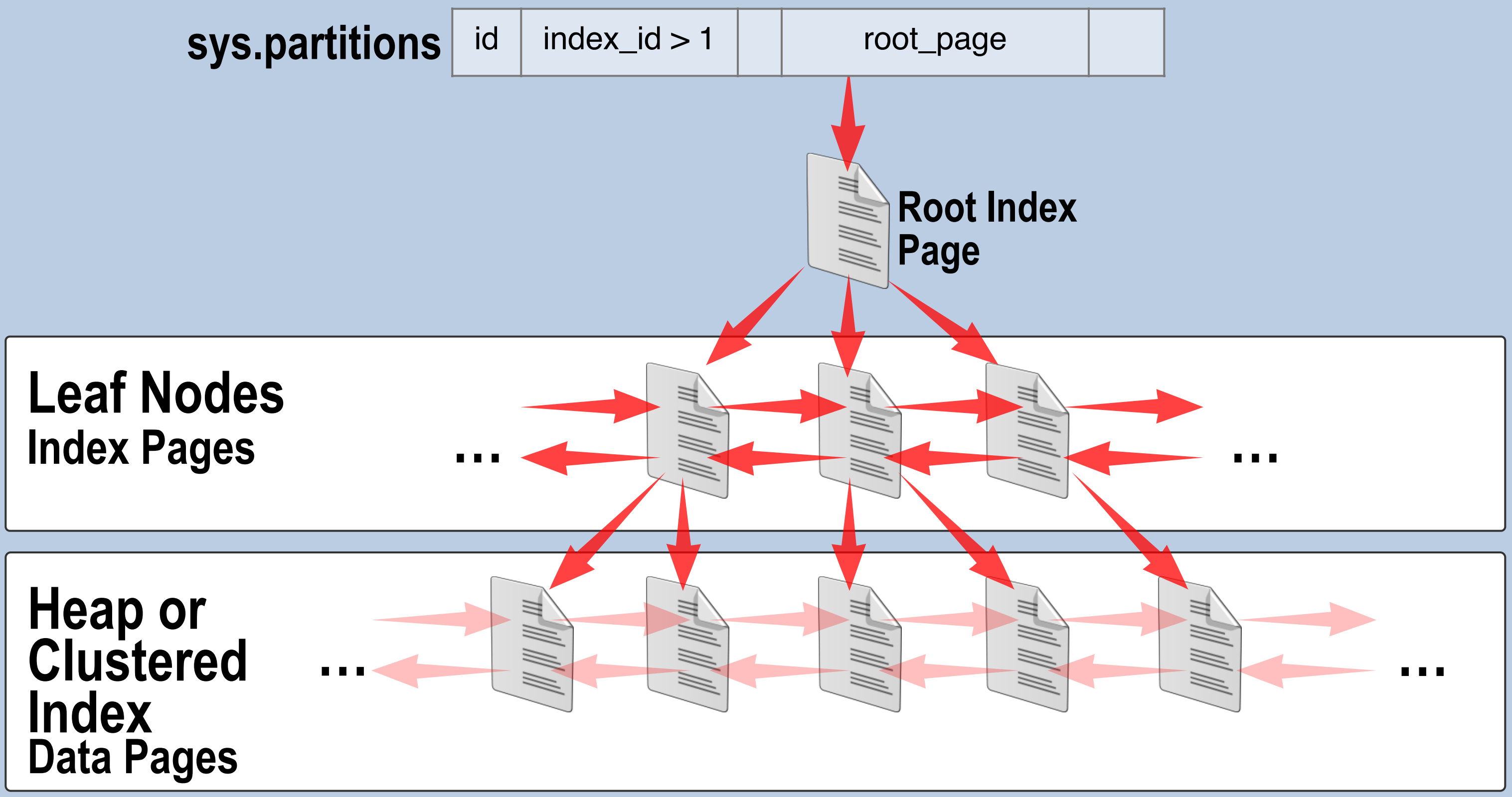
- A nonclustered index is analogous to an index in a textbook. The data is stored in one place, the index in another, with pointers to the storage location of the data

- Nonclustered indexes are the sql server default

- Existing nonclustered indexes are automatically rebuilt when:
  - An existing clustered index is dropped
  - A clustered index is created
  - The DROP_EXISTING option is used to change which columns define the clustered index
  - If no clustered index is created on the table, the rows are not guaranteed to be in any particular order

Non Clustered Index (Contd.).

- B-tree references underlying heap or clustered index
- Up to 1024 nonclustered indexes per table



Creating Unique Indexes

- Duplicate key values are not allowed when a new row is added to the table

```
USE Northwind
CREATE UNIQUE NONCLUSTERED INDEX
ixn_custindex_customers_CustID
    ON  CUSTOMERS(CustomerID)
```

- Composite index can also be created on more than one columns in a table

```
USE Northwind
CREATE UNIQUE NONCLUSTERED INDEX U_OrdID_ProdID
ON [ORDER DETAILS] (OrderID, ProductID)
```

Creating and Dropping Indexes

- Using the CREATE INDEX statement
  - Indexes are created automatically on tables with PRIMARY KEY or UNIQUE constraints
  - Indexes can be created on views if certain requirements are met

```
USE Northwind
CREATE CLUSTERED INDEX
ixc_lnmindex_employees_lastname
ON employees(lastname)
```

- Using the drop index statement

```
USE Northwind
DROP INDEX ixc_lnmindex_employees_lastname
```

Indexing Guidelines

- Columns to index
    - Primary and foreign keys
    - Those frequently searched in ranges
    - Those frequently accessed in sorted order
    - Those frequently grouped together during aggregation

- Columns not to index
    - Those seldom referenced in queries
    - Those that contain few unique values
    - Those defined with text, ntext, or image data types