Module 6:
Planning Indexes

Overview

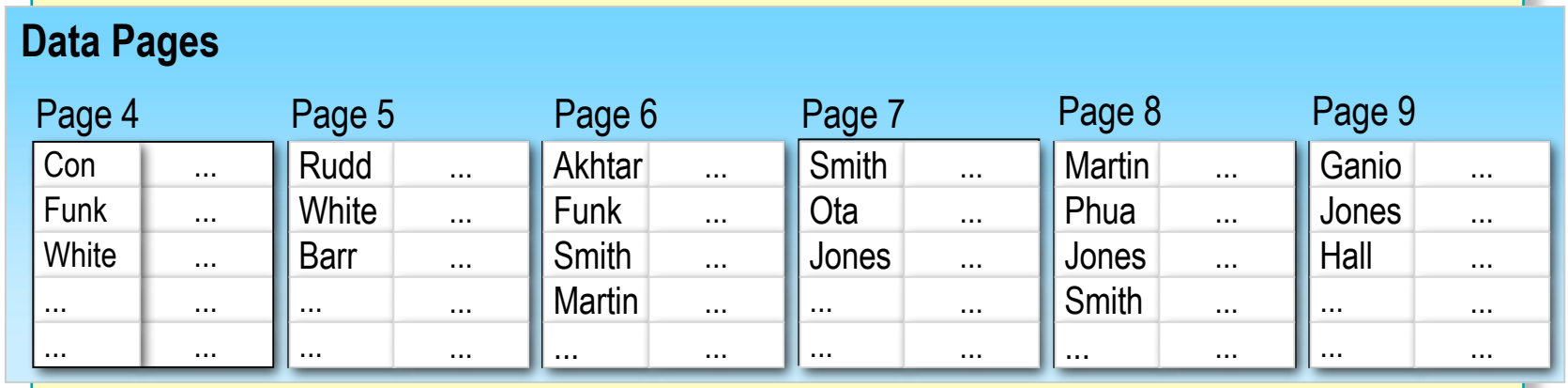
- Introduction to Indexes
- Index Architecture
- How SQL Server Retrieves Stored Data
- How SQL Server Maintains Index and Heap Structures
- Deciding Which Columns to Index

Introduction to Indexes

- How SQL Server Stores and Accesses Data
- Whether to Create Indexes

How SQL Server Stores and Accesses Data

- **How Data Is Stored**
  - Rows are stored in data pages
  - Heaps are a collection of data pages for a table
- **How Data Is Accessed**
  - Scanning all data pages in a table
  - Using an index that points to data on a page



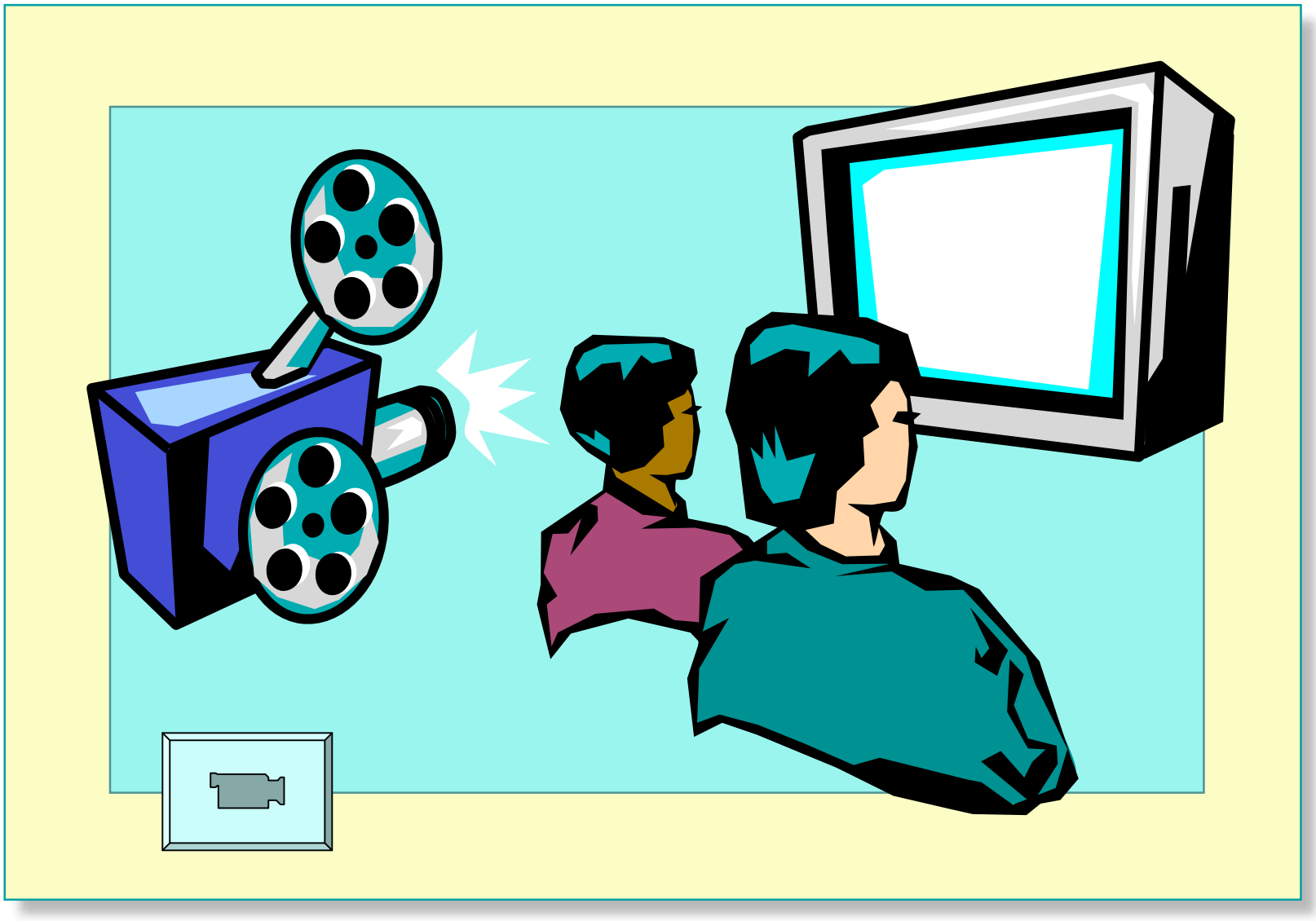
Whether to Create Indexes

- **Why to Create an Index**
  - Speeds up data access
  - Enforces uniqueness of rows
- **Why Not to Create an Index**
  - Consumes disk space
  - Incurs overhead

Index Architecture

- SQL Server Index Architecture
- Using Heaps
- Using Clustered Indexes
- Using Nonclustered Indexes

Multimedia Presentation: SQL Server Index Architecture



Using Heaps

SQL Server:

- **Uses Index Allocation Map Pages That:**
  - Contain information on where the extents of a heap are stored
  - Navigate through the heap and find available space for new rows being inserted
  - Connect data pages
- **Reclaims Space for New Rows in the Heap When a Row Is Deleted**

Using Clustered Indexes

- Each Table Can Have Only One Clustered Index
- The Physical Row Order of the Table and the Order of Rows in the Index Are the Same
- Key Value Uniqueness Is Maintained Explicitly or Implicitly

Using Nonclustered Indexes

- **Nonclustered Indexes Are the SQL Server Default**
- **Existing Nonclustered Indexes Are Automatically Rebuilt When:**
  - An existing clustered index is dropped
  - A new clustered index is created
  - The DROP_EXISTING option is used to change which columns define the clustered index

How SQL Server Retrieves Stored Data

- How SQL Server Uses the sysindexes Table
- Finding Rows Without Indexes
- Finding Rows in a Heap with a Nonclustered Index
- Finding Rows in a Clustered Index
- Finding Rows in a Clustered Index with a Nonclustered Index

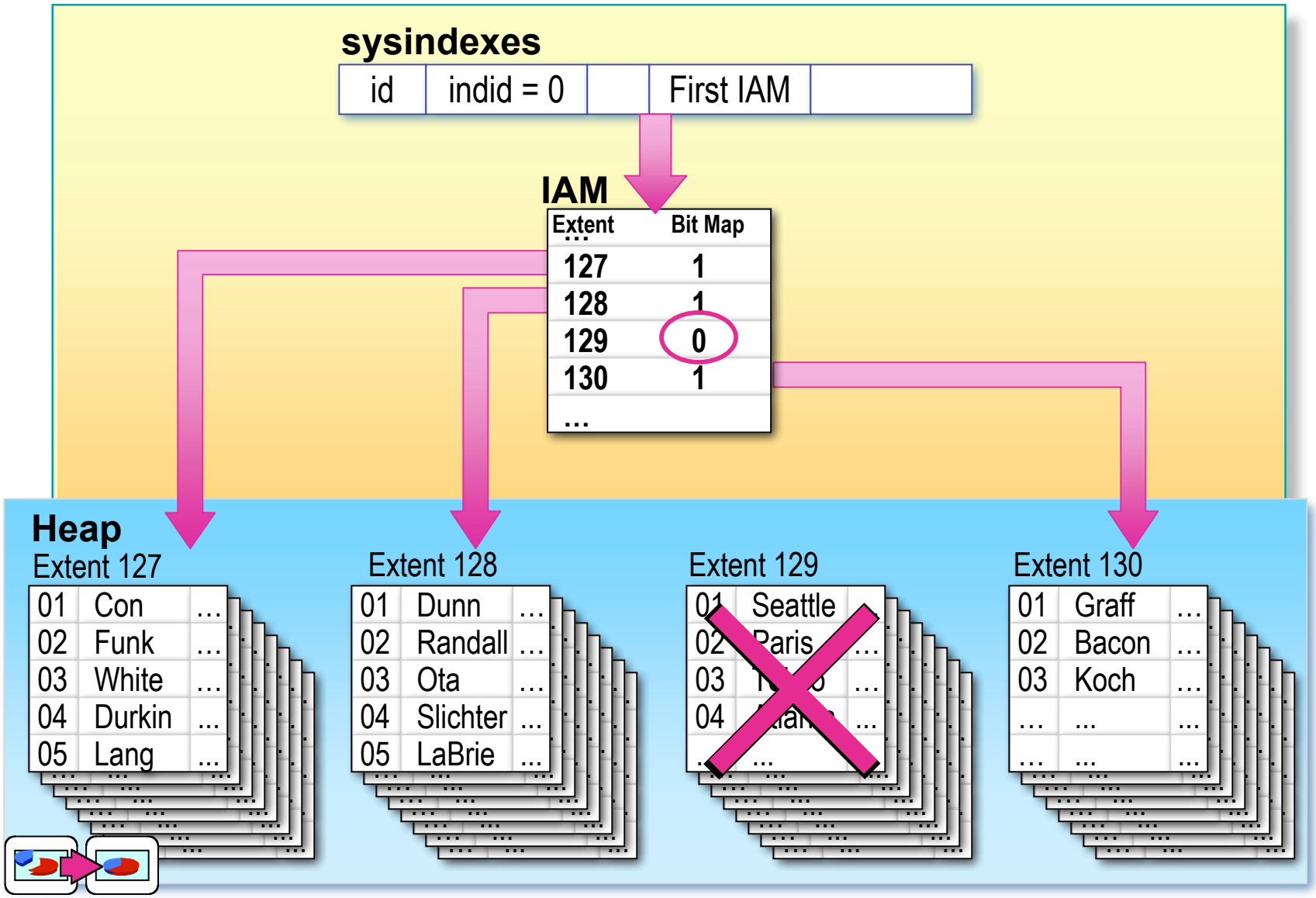
How SQL Server Uses the sysindexes Table

- Describes the Indexes

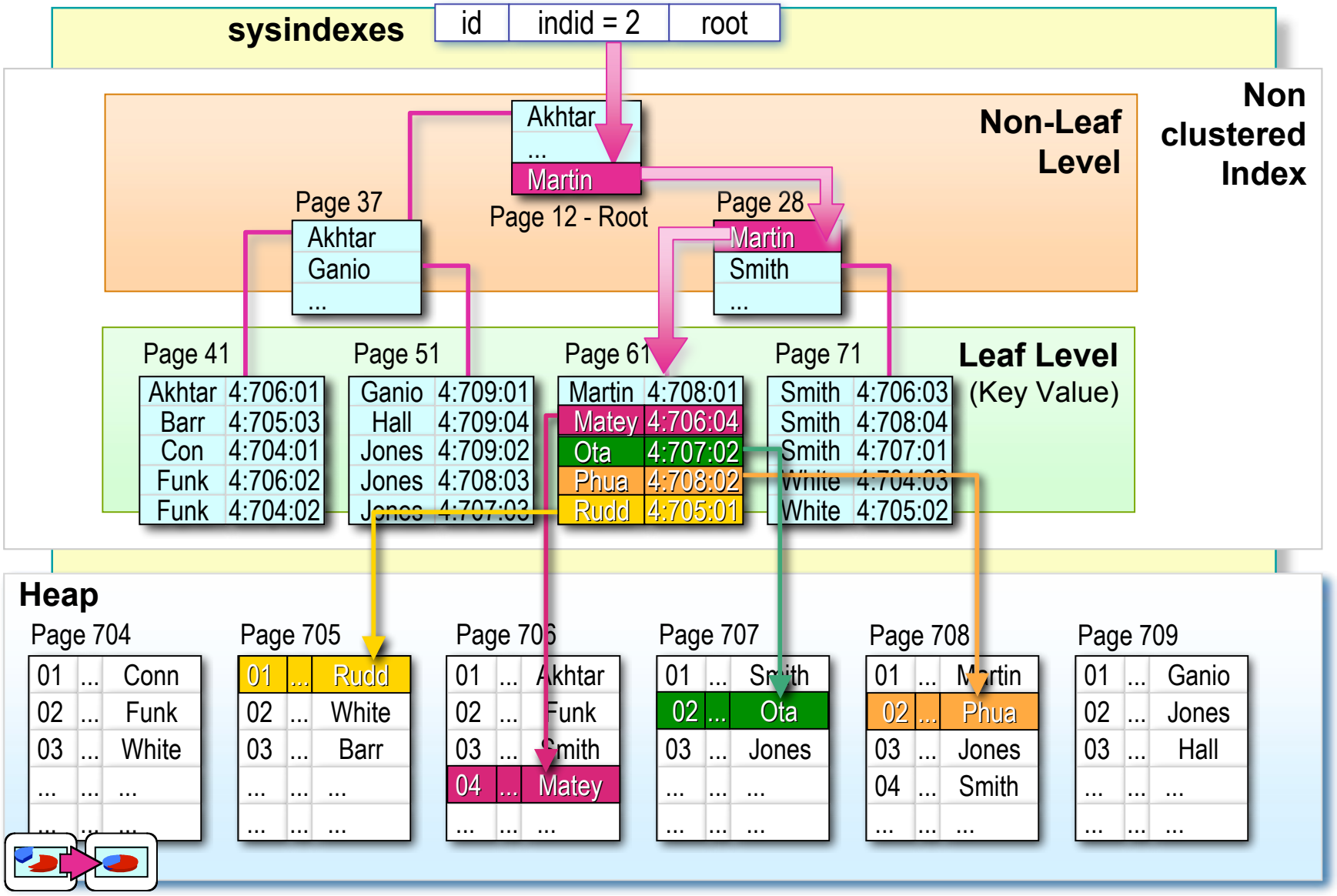
indid	Object Type
0	Heap
1	Clustered Index
2 to 250	Nonclustered Index
255	text, ntext, or image

- Location of IAM, First, and Root of Indexes
- Number of Pages and Rows
- Distribution of Data

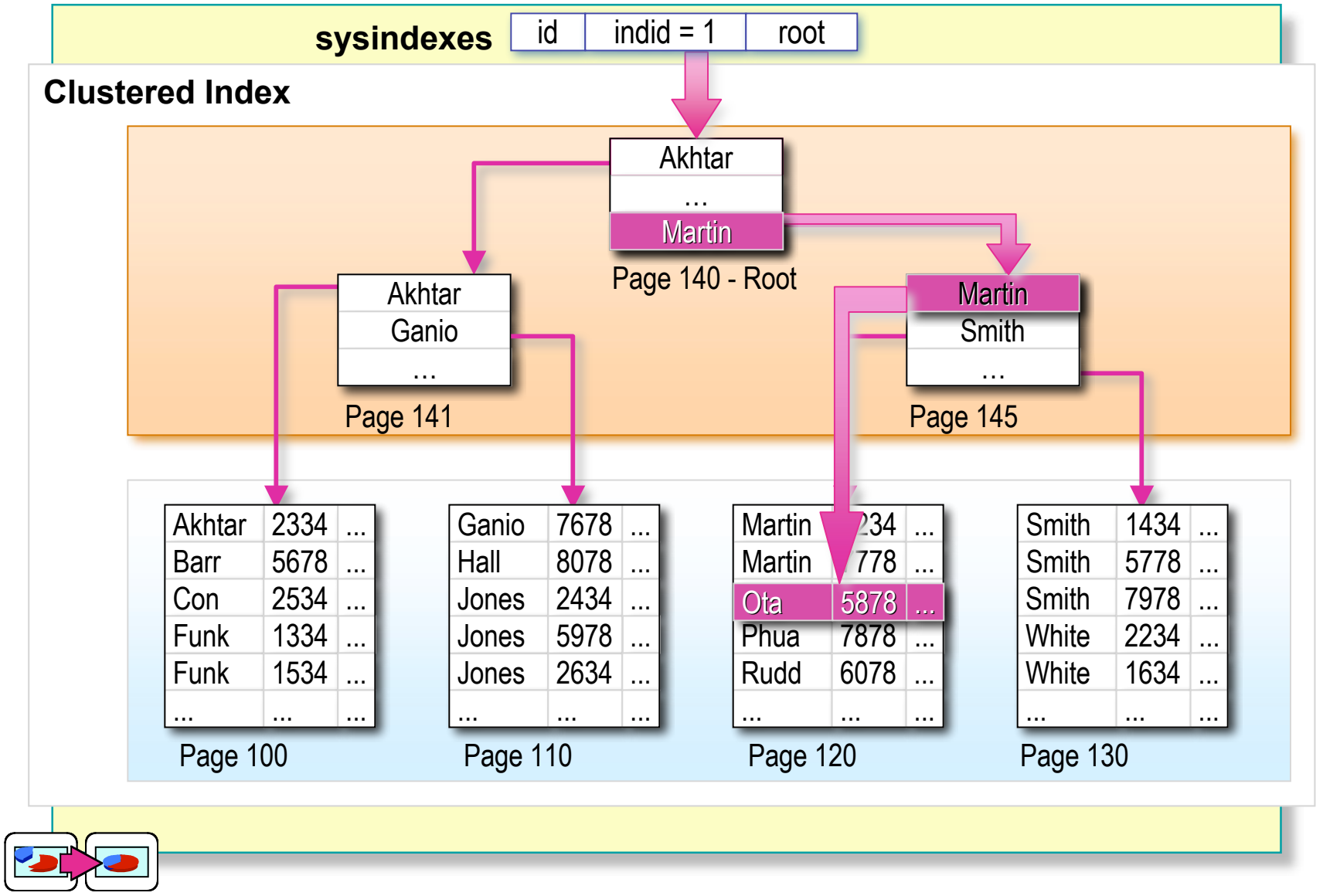
Finding Rows Without Indexes



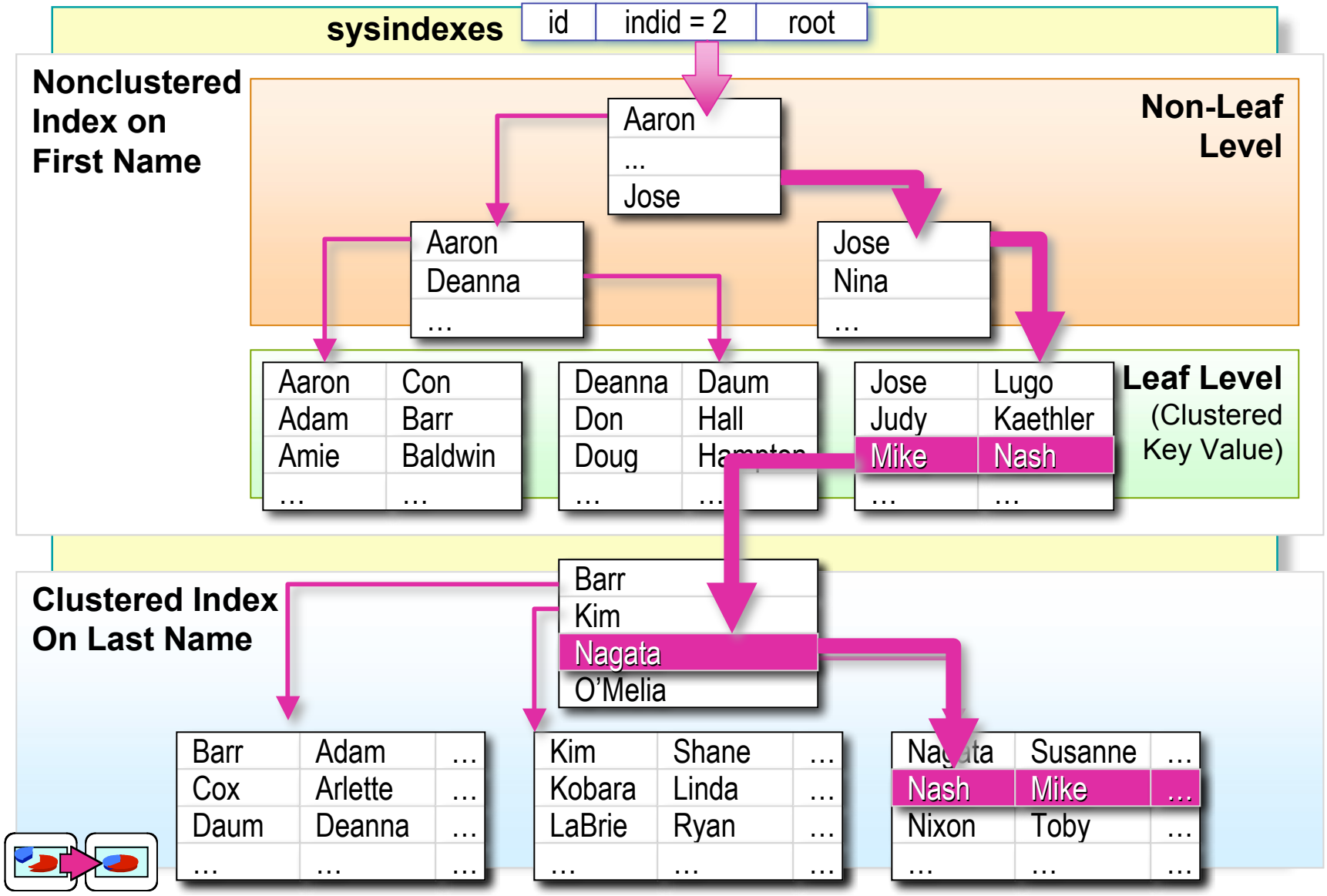
Finding Rows in a Heap with a Nonclustered Index



Finding Rows in a Clustered Index



Finding Rows in a Clustered Index with a Nonclustered Index

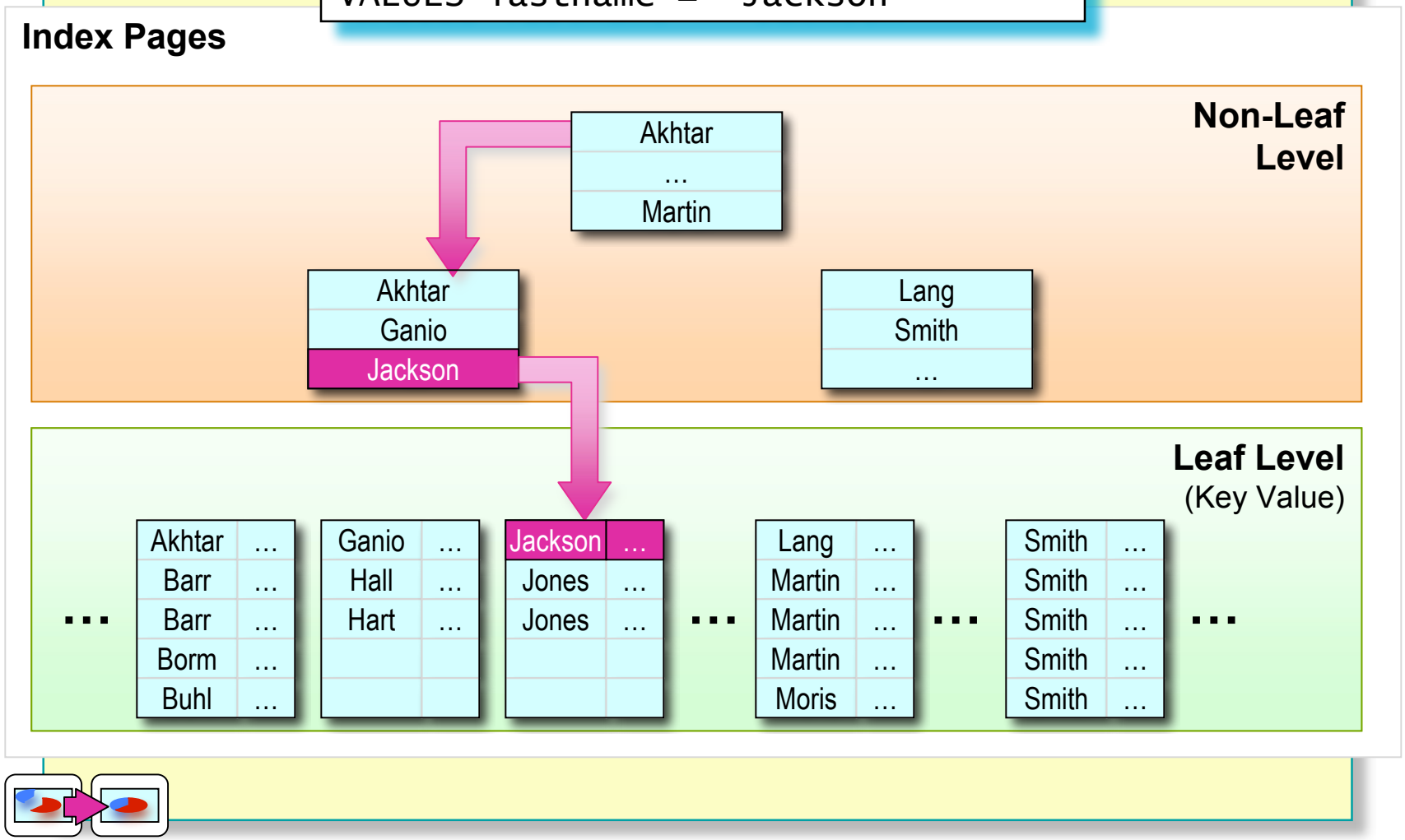


How SQL Server Maintains Index and Heap Structures

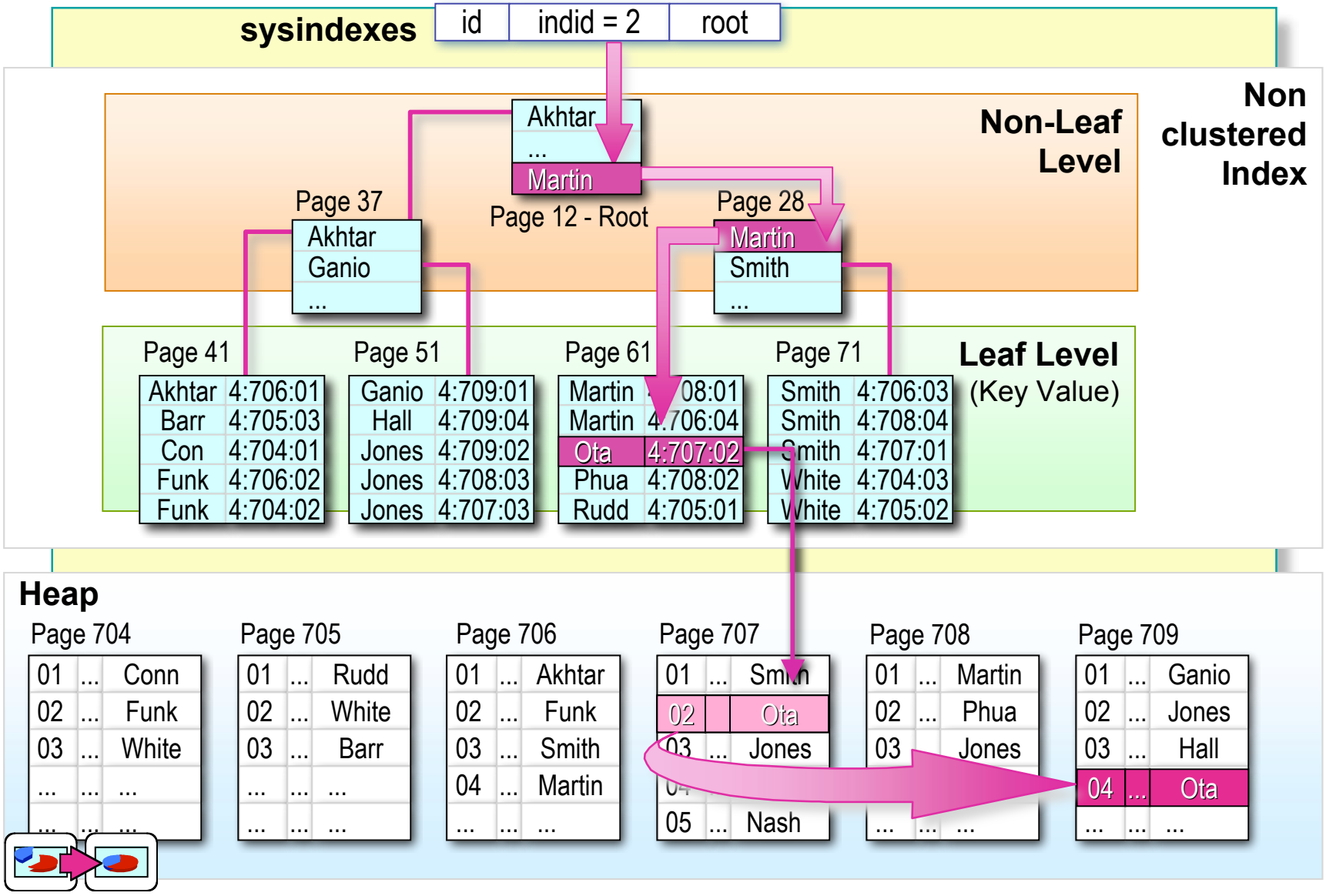
- Page Splits in an Index
- Forwarding Pointer in a Heap
- How SQL Server Updates Rows
- How SQL Server Deletes Rows

Page Splits in an Index

```
INSERT member (last name)
VALUES lastname = 'Jackson'
```



Forwarding Pointer in a Heap

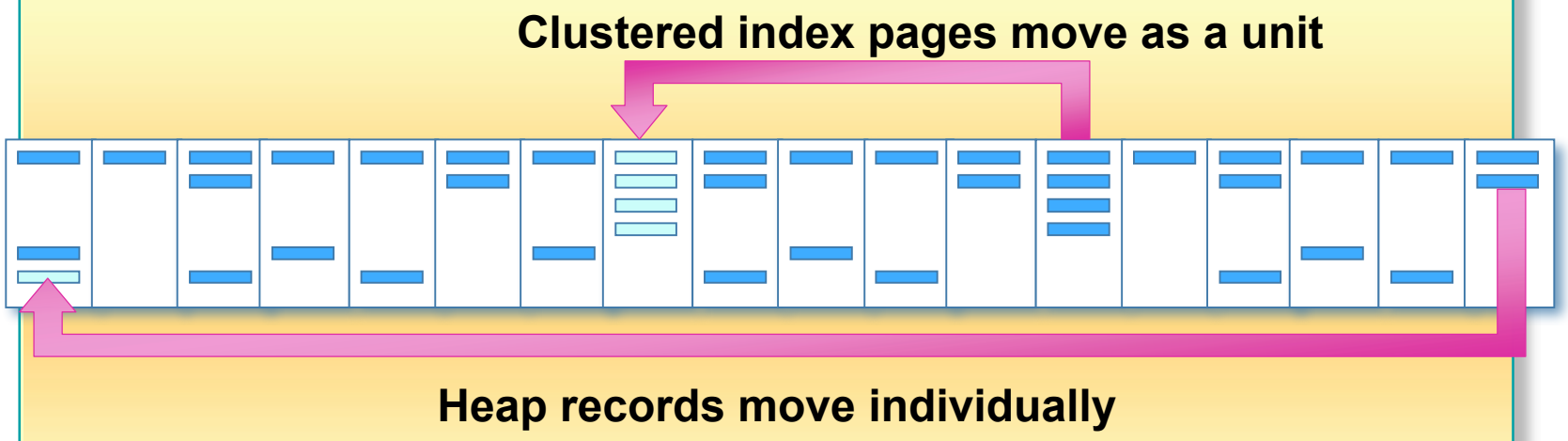


How SQL Server Updates Rows

- An Update Generally Does Not Cause a Row to Move
- An Update Can Be a Delete Followed by an Insert
- Batch Updates Touch Each Index Only Once

How SQL Server Deletes Rows

- How Deletes Cause Ghost Records
- How SQL Server Reclaims Space
- How Files Can Shrink



Deciding Which Columns to Index

- Understanding the Data
- Indexing Guidelines
- Choosing the Appropriate Clustered Index
- Indexing to Support Queries
- Determining Selectivity
- Determining Density
- Determining Distribution of Data

Understanding the Data

- **Logical and Physical Design**
- **Data Characteristics**
- **How Data Is Used**
  - The types of queries performed
  - The frequency of queries that are typically performed

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

Choosing the Appropriate Clustered Index

- **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

Indexing to Support Queries

- **Using Search Arguments**
- **Writing Good Search Arguments**
  - Specify a WHERE clause in the query
  - Verify that the WHERE clause limits the number of rows
  - Verify that an expression exists for every table referenced in the query
  - Avoid using leading wildcards

Determining Selectivity

High selectivity

member_no	last_name	first_name
1	Randall	Joshua
2	Flood	Kathie
.		
.		
.		
10000	Anderson	Bill

$$\frac{\text{Number of rows meeting criteria}}{\text{Total number of rows in table}} = \frac{1000}{10000} = 10\%$$

```
SELECT *
FROM member
WHERE member_no > 8999
```

Good candidate for Index?

Low selectivity

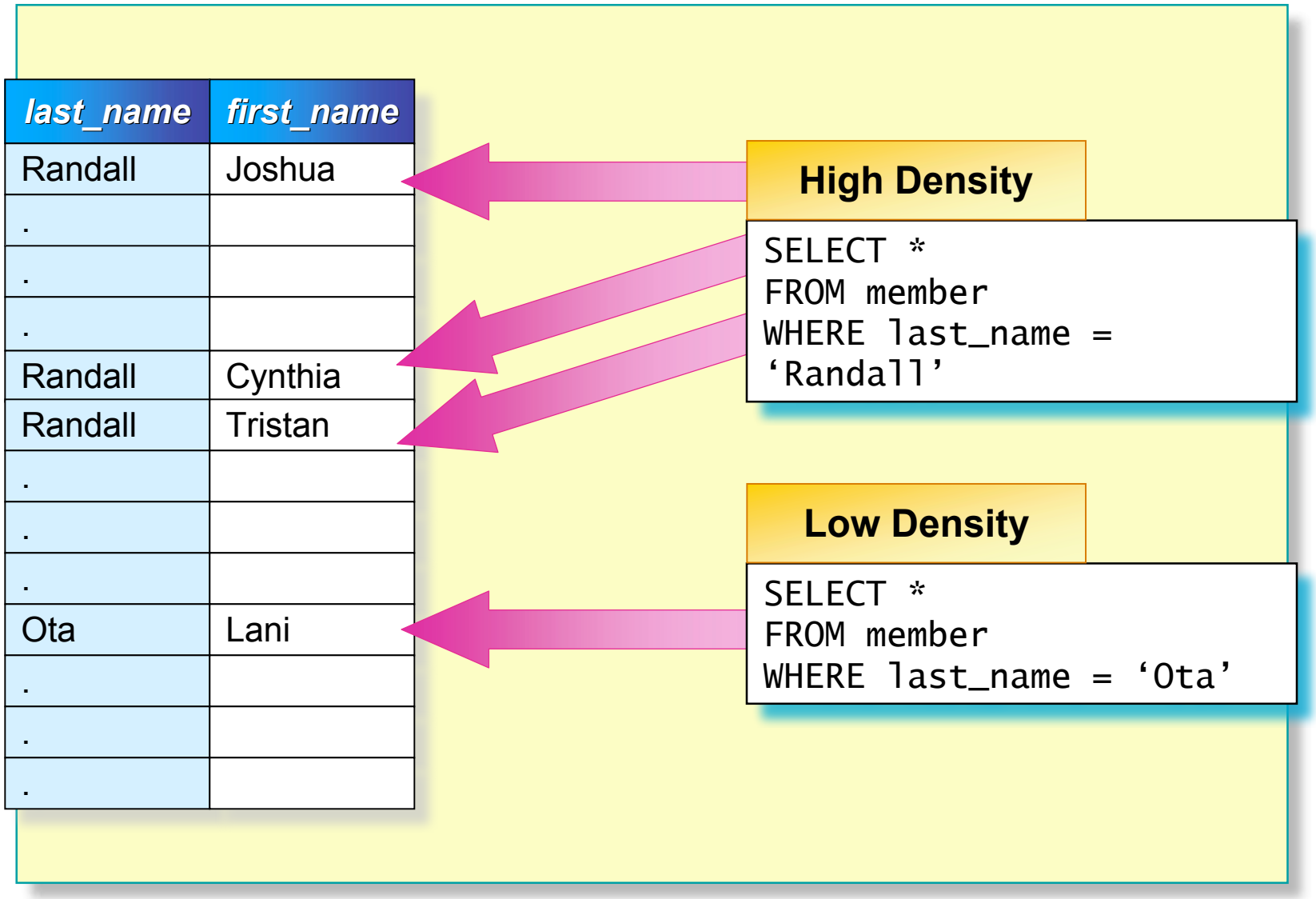
member_no	last_name	first_name
1	Randall	Joshua
2	Flood	Kathie
.		
.		
.		
10000	Anderson	Bill

$$\frac{\text{Number of rows meeting criteria}}{\text{Total number of rows in table}} = \frac{9000}{10000} = 90\%$$

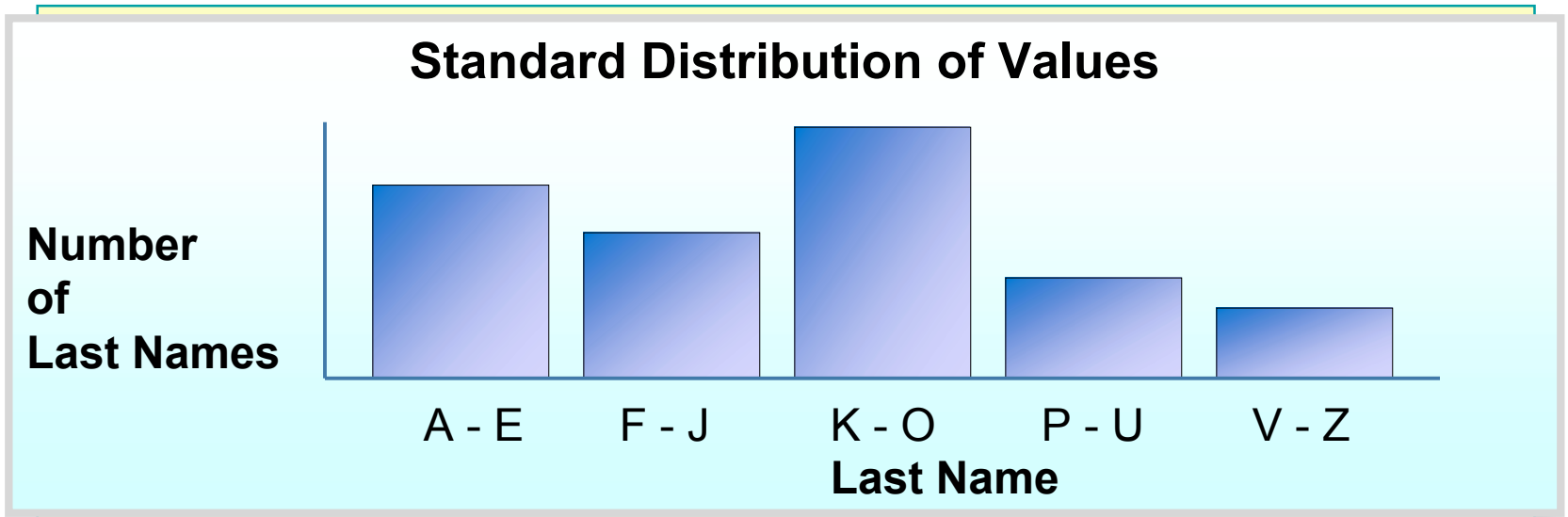
```
SELECT *
FROM member
WHERE member_no < 9001
```

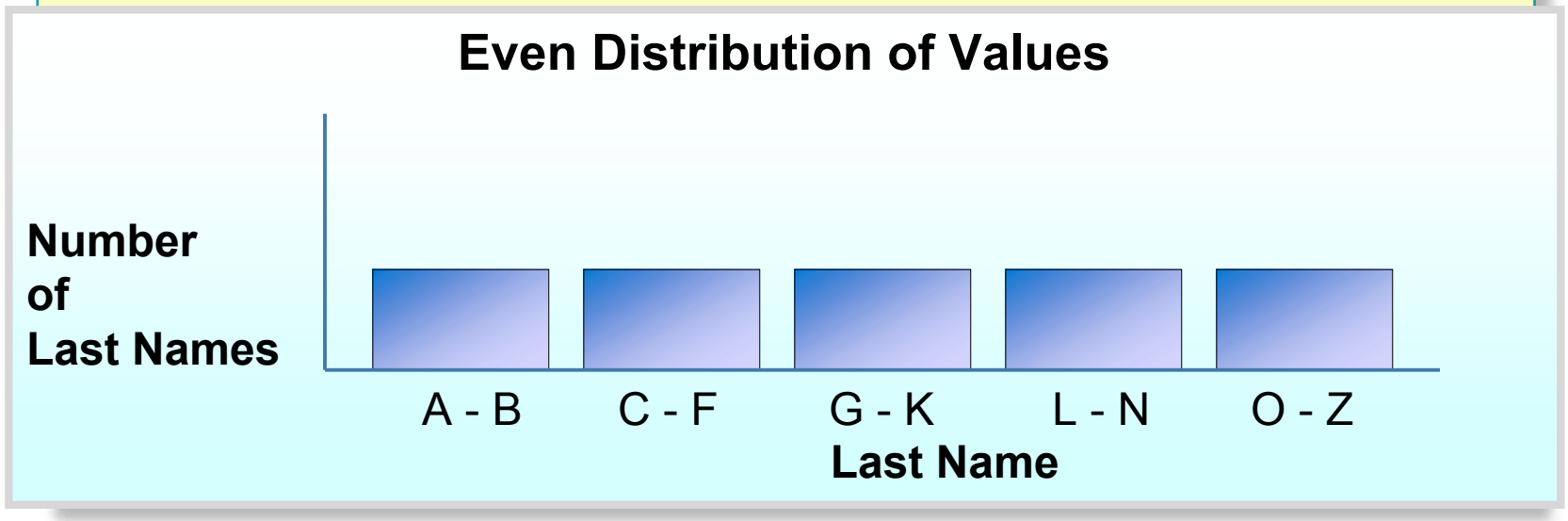
Good candidate for Index?

Determining Density



Determining Distribution of Data





Recommended Practices

Review

- Introduction to Indexes
- Index Architecture
- How SQL Server Retrieves Stored Data
- How SQL Server Maintains Index and Heap Structures
- Deciding Which Columns to Index