Indexes

Rose-Hulman Institute of Technology

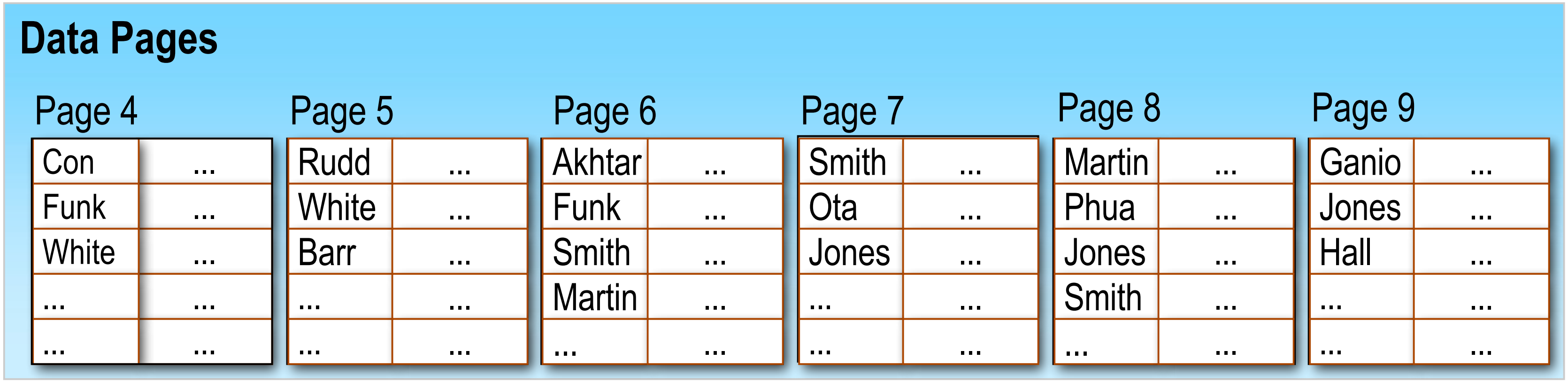
Curt Clifton

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

Storing and Accessing Data

- How Data Is Stored
  - Rows are stored in data pages
  - Heaps are a collection of data pages for a table



Data Pages

Page 4		Page 5		Page 6		Page 7		Page 8		Page 9	
Con	...	Rudd	...	Akhtar	...	Smith	...	Martin	...	Ganio	...
Funk	...	White	...	Funk	...	Ota	...	Phua	...	Jones	...
White	...	Barr	...	Smith	...	Jones	...	Jones	...	Hall	...
...	...	...	...	Martin	...	...	...	Smith	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...

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

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

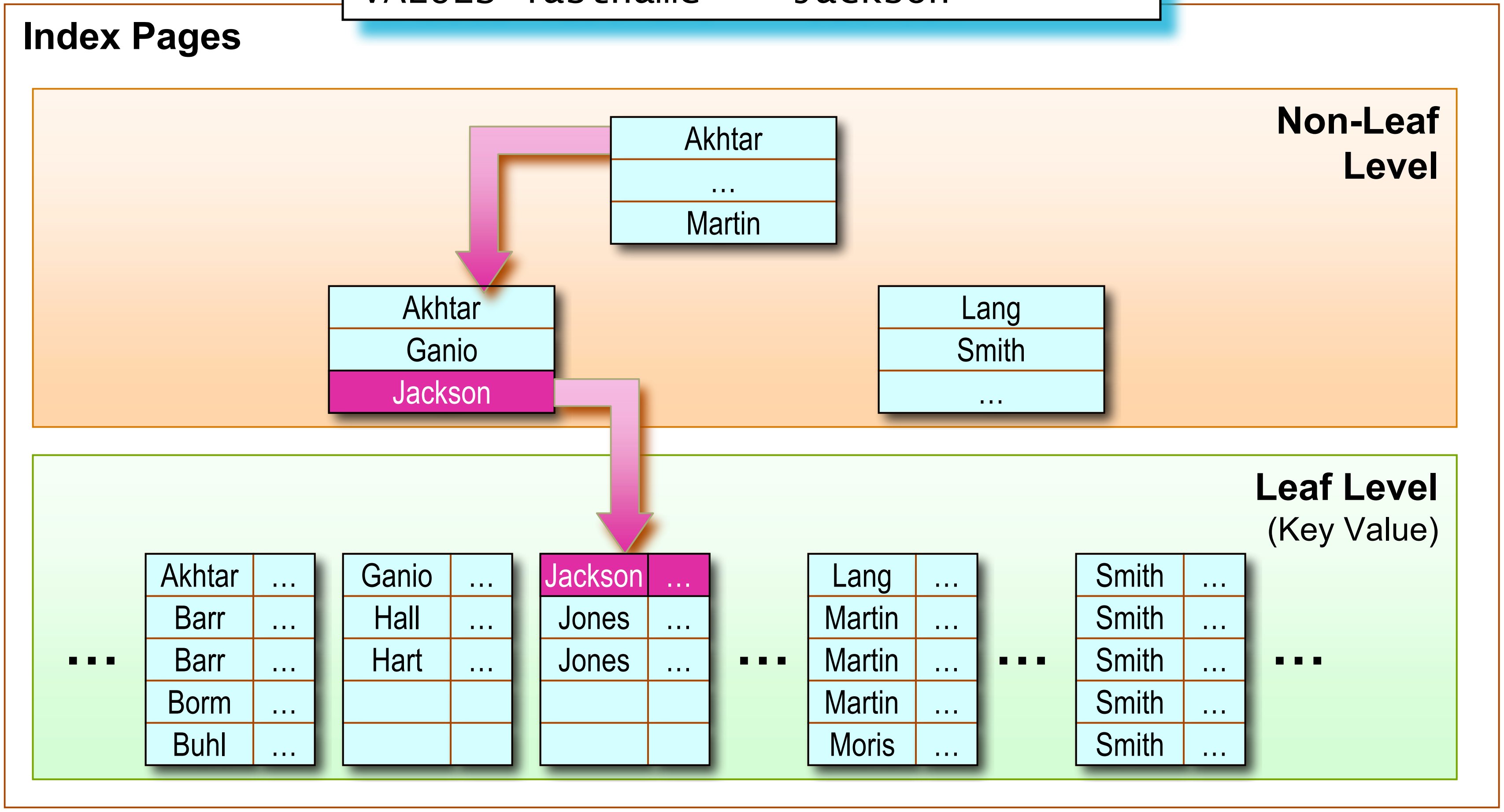
Using Nonclustered Indexes

- 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

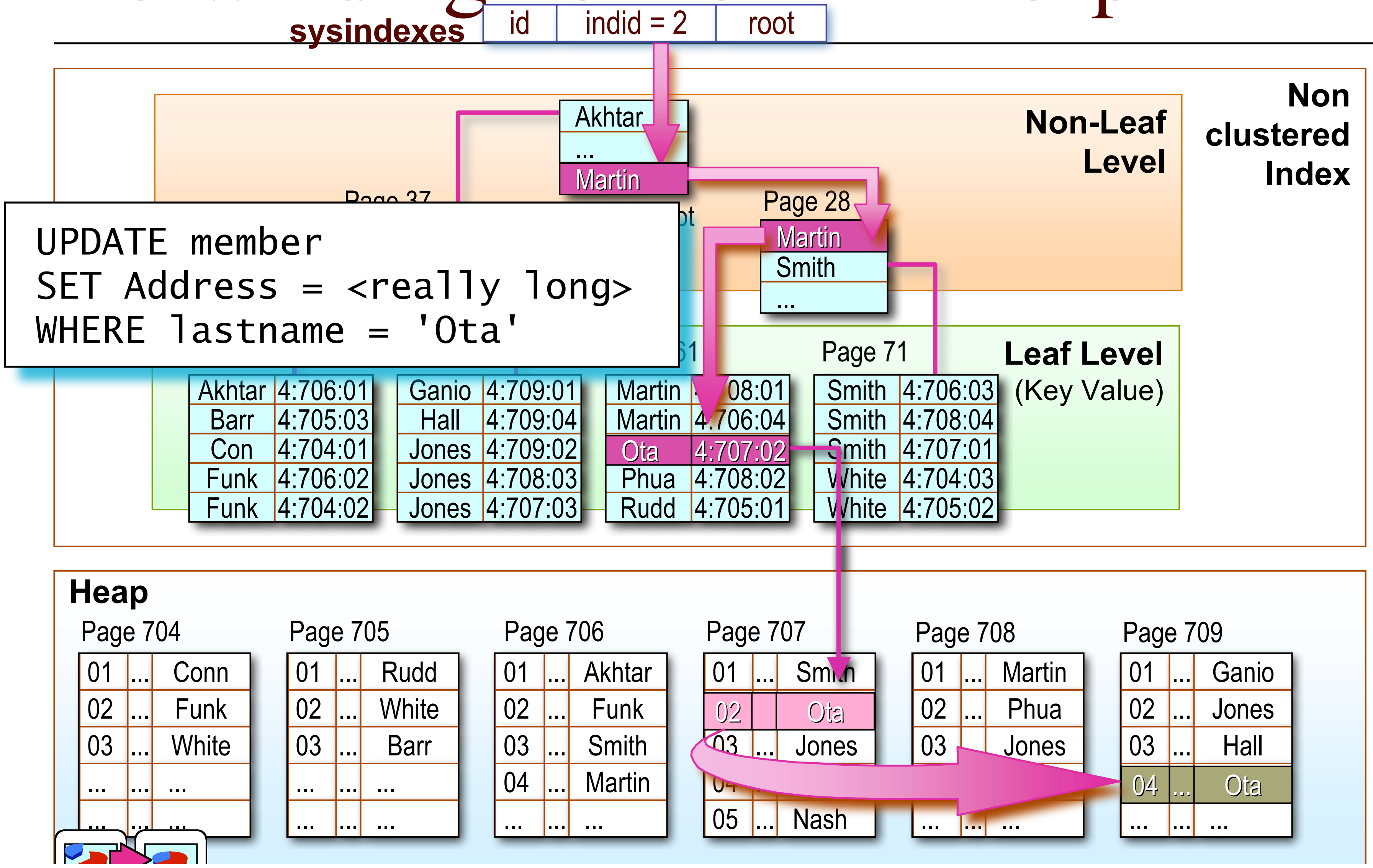
Maintaining Index and Heap Structures

Page Splits in an Index

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



Forwarding Pointer in a Heap



```
UPDATE member
SET Address = <really long>
WHERE lastname = 'Ota'
```

Row Updates

- Generally do not cause rows to move
- Like a delete followed by an update
  - Logically
  - Sometimes practically
- Batch updates touch each index once

Deletion

- Deletion creates “ghost records”
- Reclaiming space
  - Free pages when empty
  - For indexed table:
    - Can overwrite ghost records immediately
  - For non-indexed table:
    - Compact records when more space is needed for insert

Deciding What to Index

What You Need to Know

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

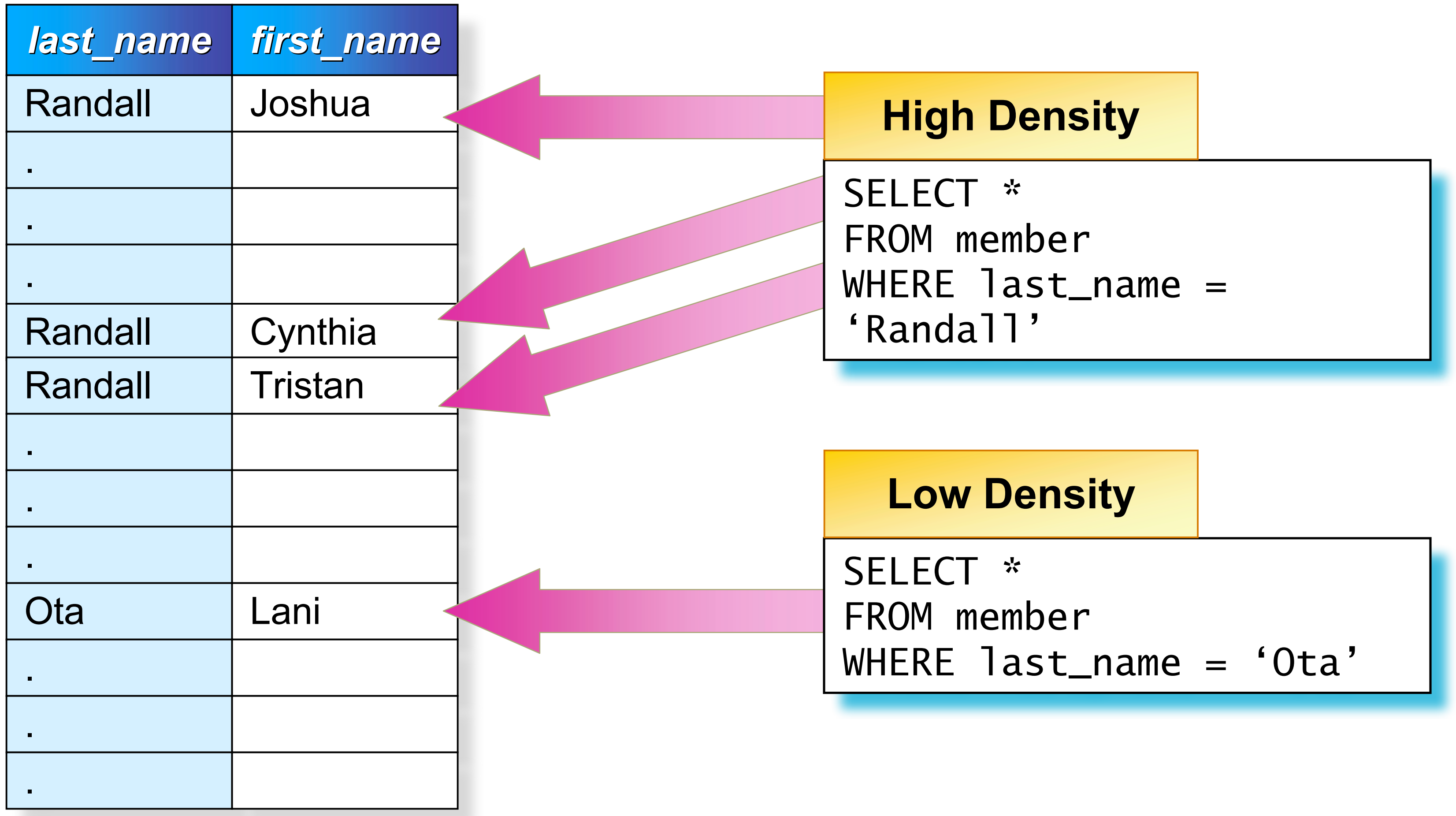
Indexing Guidelines

- 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 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

Data Characteristics – Density



last_name	first_name
Randall	Joshua
.	
.	
.	
Randall	Cynthia
Randall	Tristan
.	
.	
.	
Ota	Lani
.	
.	
.	

High Density

```
SELECT *
FROM member
WHERE last_name =
‘Randall’
```

Low Density

```
SELECT *
FROM member
WHERE last_name = ‘Ota’
```

Data Characteristics – Selectivity

- How effective is a column at selecting a subset of the data
- A property of a given query:
  - Rows matching property / Total number of rows

High selectivity

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

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

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

Low selectivity

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

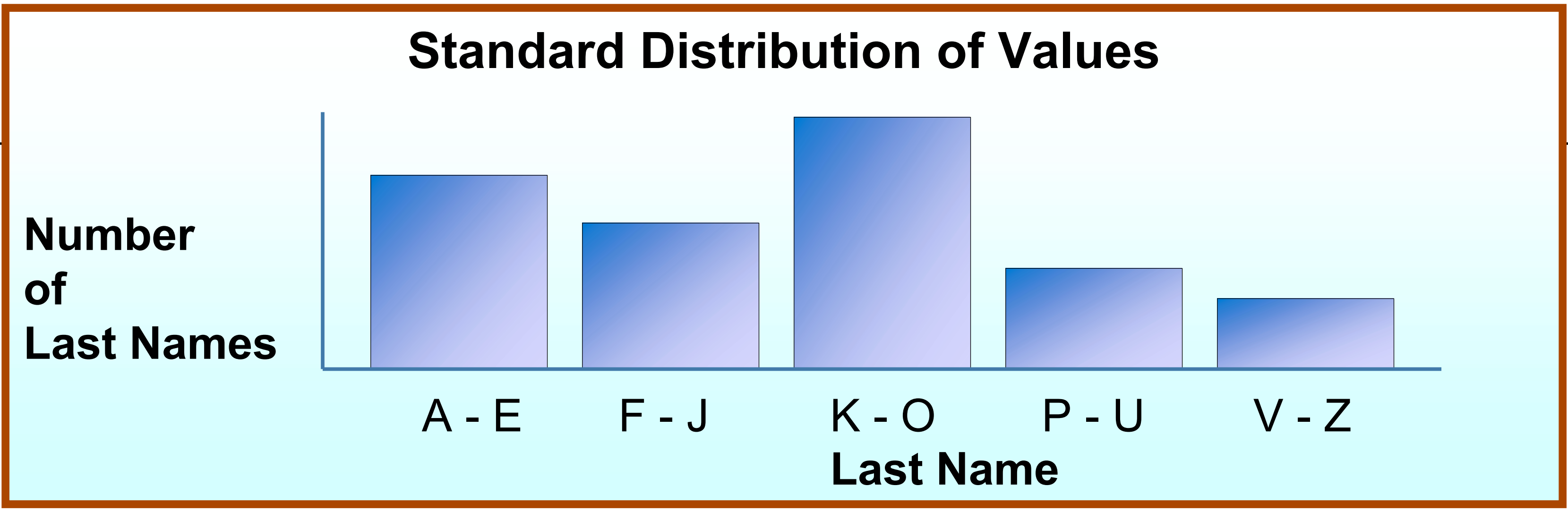
```
SELECT *
FROM member
WHERE member_no < 9001
```

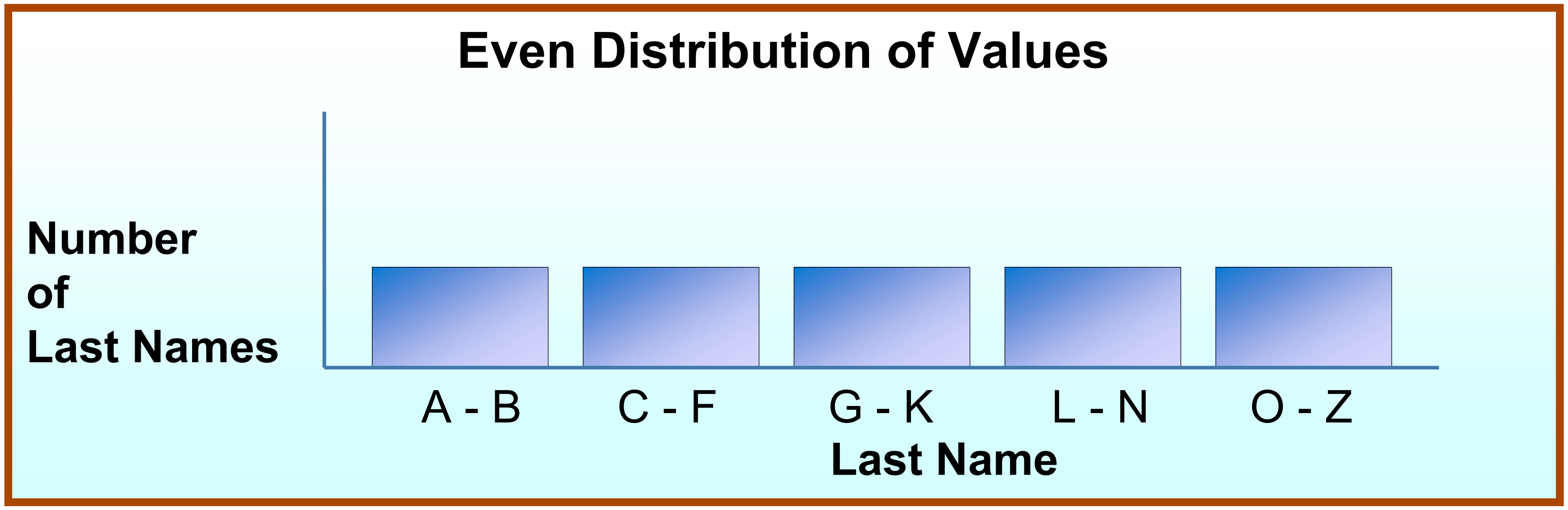
$$\frac{\text{Number of rows meeting criteria}}{\text{Total number of rows in table}} = \frac{9000}{10000} = 90\%$$

Indexing to Support Queries

- 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

Introduction to Statistics


Standard Distribution of Values
Number of Last Names
A - E
F - J
K - O
P - U
V - Z
Last Name


Even Distribution of Values
Number of Last Names
A - B
C - F
G - K
L - N
O - Z
Last Name

How Statistics Are Gathered

- DMBS reads/samples column values
  - Produces an evenly distributed sorted list of values
- Performs a full scan or sampling of rows
  - Depending on size of table and granularity wanted
- Selects samplings if necessary
  - Picks rows to be sampled
  - Includes all rows on the page of selected rows

Creating Statistics

- Automatically Creating Statistics
  - Indexed columns that contain data
  - Non-indexed columns that are used in a join predicate or a WHERE clause
- Manually Creating Statistics
  - Columns that are not indexed
  - All columns other than the first column of a composite index

Viewing Statistics

- The DBCC SHOW_STATISTICS Statement Returns Statistical Information in the Distribution Page for an Index or Column
- Statistical Information Includes:
  - The time when the statistics were last updated
  - The number of rows sampled to produce the histogram
  - Density information
  - Average key length
  - Histogram step information

Performance Considerations

- Create Indexes on Foreign Keys
- Create the Clustered Index Before Nonclustered Indexes
- Consider Creating Composite Indexes
- Create Multiple Indexes for a Table That Is Read Frequently