Database Systems: Design, Implementation, and Management

Lesson 10

Database Performance-Tuning Concepts

- Goal of database performance is to execute queries as fast as possible
- **Database performance tuning**
  - Set of activities and procedures designed to reduce response time of database system
- All factors must operate at optimum level with minimal bottlenecks
- **Good database performance starts with good database design**

TABLE 11.1 General Guidelines for Better System Performance

	SYSTEM RESOURCES	CLIENT	SERVER
Hardware	CPU	The fastest possible dual-core CPU or higher	The fastest possible Multiple processors (quad-core technology)
	RAM	The maximum possible	The maximum possible
	Hard disk	Fast SATA/EIDE hard disk with sufficient free hard disk space	Multiple high-speed, high-capacity hard disks (SCSI/SATA/Firewire/Fibre Channel) in RAID configuration
	Network	High-speed connection	High-speed connection
Software	Operating system	Fine-tuned for best client application performance	Fine-tuned for best server application performance
	Network	Fine-tuned for best throughput	Fine-tuned for best throughput
	Application	Optimize SQL in client application	Optimize DBMS server for best performance

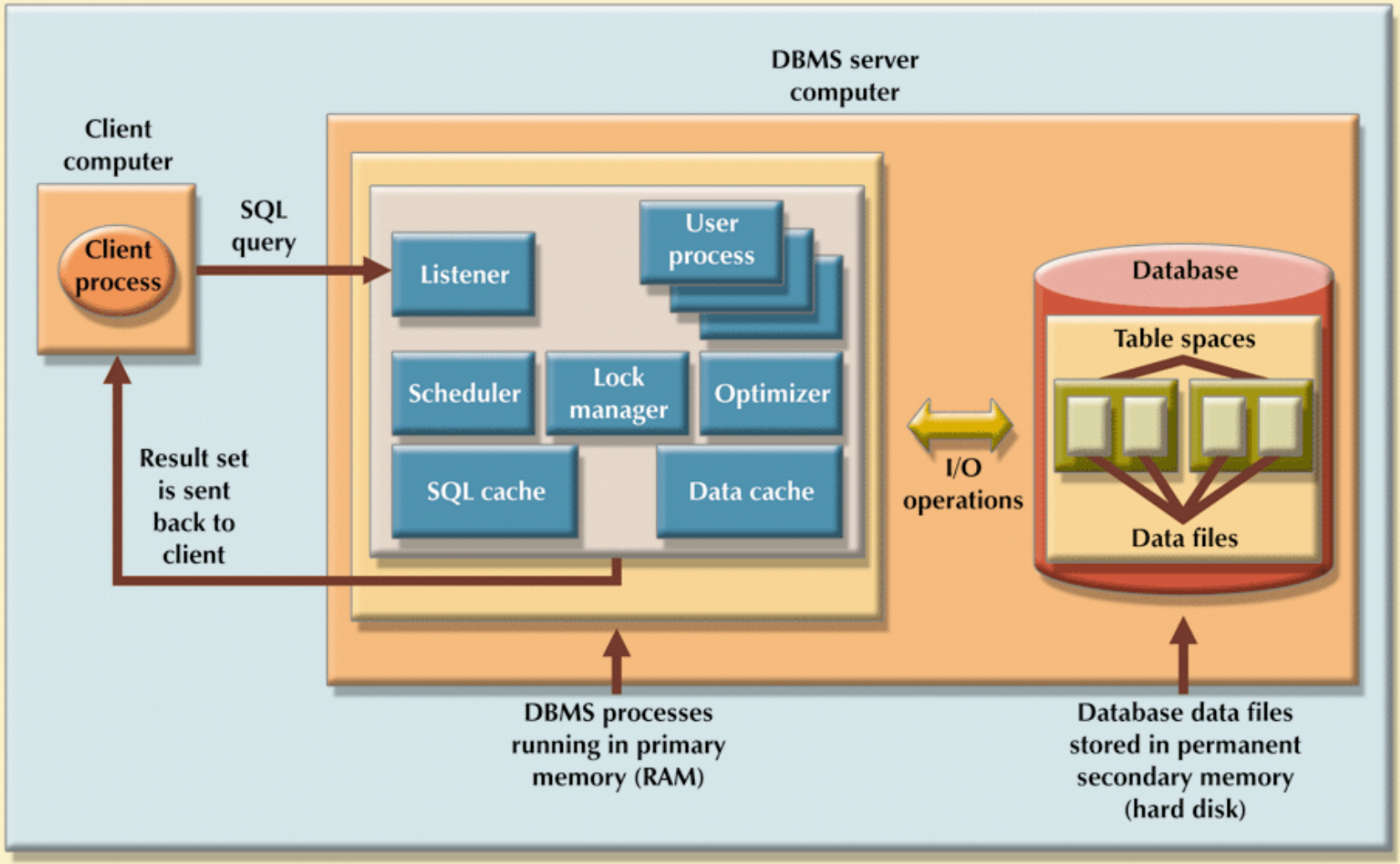
Performance Tuning: Client and Server

- Client side
  - Generate SQL query that returns correct answer in least amount of time
    - Using minimum amount of resources at server
  - **SQL performance tuning**
- Server side
  - DBMS environment configured to respond to clients' requests as fast as possible
    - Optimum use of existing resources
  - **DBMS performance tuning**

DBMS Architecture

- All data in database are stored in data files
- **Data files**
  - Automatically expand in predefined increments known as **extends**
  - Grouped in **file groups** or **table spaces**
- Table space or file group
  - Logical grouping of several data files that store data with similar characteristics

FIGURE 11.1 Basic DBMS architecture



DBMS Architecture (cont'd.)

- **Data cache** or **buffer cache**: shared, reserved memory area
  - Stores most recently accessed data blocks in RAM
- **SQL cache** or **procedure cache**: stores most recently executed SQL statements
  - Also PL/SQL procedures, including triggers and functions
- DBMS retrieves data from permanent storage and places it in RAM

DBMS Architecture (cont'd.)

- **Input/output request**: low-level data access operation to/from computer devices
- Data cache is faster than data in data files
  - DBMS does not wait for hard disk to retrieve data
- Majority of performance-tuning activities focus on minimizing I/O operations
- Typical DBMS processes:
  - Listener, user, scheduler, lock manager, optimizer

Database Statistics

- Measurements about database objects and available resources:
  - Tables
  - Indexes
  - Number of processors used
  - Processor speed
  - Temporary space available

Database Statistics (cont'd.)

- Make critical decisions about improving query processing efficiency
- Can be gathered manually by DBA or automatically by DBMS

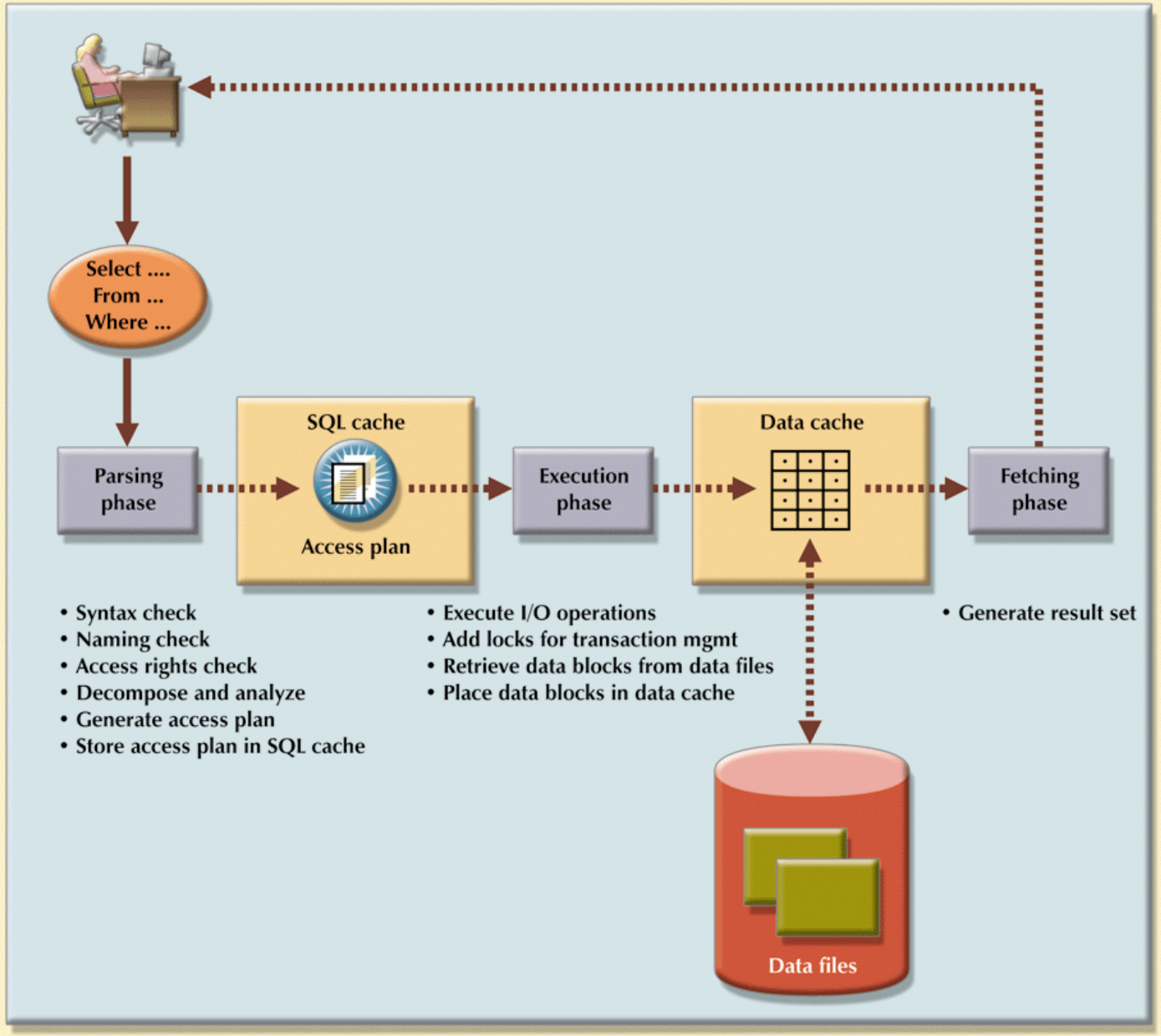
TABLE 11.2 Sample Database Statistics Measurements

DATABASE OBJECT	SAMPLE MEASUREMENTS
Tables	Number of rows, number of disk blocks used, row length, number of columns in each row, number of distinct values in each column, maximum value in each column, minimum value in each column, and columns that have indexes
Indexes	Number and name of columns in the index key, number of key values in the index, number of distinct key values in the index key, histogram of key values in an index, and number of disk pages used by the index
Environment Resources	Logical and physical disk block size, location and size of data files, and number of extends per data file

Query Processing

- DBMS processes queries in three phases
  - Parsing
    - DBMS parses the query and chooses the most efficient access/execution plan
  - Execution
    - DBMS executes the query using chosen execution plan
  - Fetching
    - DBMS fetches the data and sends the result back to the client

FIGURE 11.2 Query processing



Indexes and Query Optimization

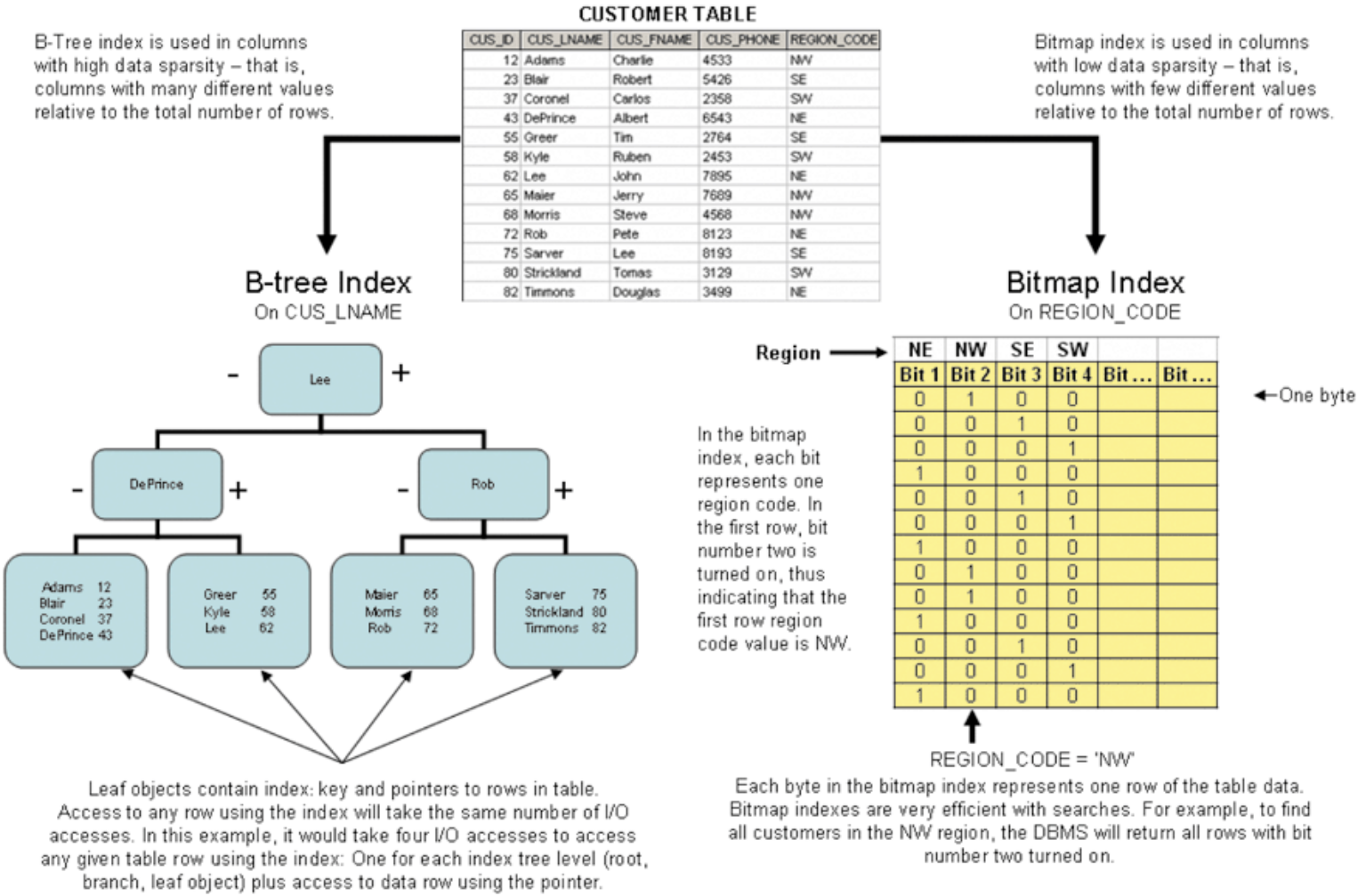
- Indexes
  - Crucial in speeding up data access
  - Facilitate searching, sorting, and using aggregate functions as well as join operations
  - Ordered set of values that contains index key and pointers
- More efficient to use index to access table than to scan all rows in table sequentially

Indexes and Query Optimization (cont'd.)

- **Data sparsity**: number of different values a column could possibly have
- Indexes implemented using:
  - **Hash indexes**
  - **B-tree indexes**
  - **Bitmap indexes**
- DBMSs determine best type of index to use

FIGURE 11.3 Index representation for the CUSTOMER table

FIGURE 11.4 B-tree and bitmap index representation



SQL Performance Tuning

- Evaluated from client perspective
  - Most current relational DBMSs perform automatic query optimization at the server end
  - Most SQL performance optimization techniques are DBMS specific
    - Rarely portable
- Majority of performance problems are related to poorly written SQL code
- Carefully written query usually outperforms a poorly written query

Index Selectivity

- Indexes are used when:
  - Indexed column appears by itself in search criteria of WHERE or HAVING clause
  - Indexed column appears by itself in GROUP BY or ORDER BY clause
  - MAX or MIN function is applied to indexed column
  - Data sparsity on indexed column is high
- Measure of how likely an index will be used

Index Selectivity (cont'd.)

- General guidelines for indexes:
  - Create indexes for each attribute in WHERE, HAVING, ORDER BY, or GROUP BY clause
  - Do not use in small tables or tables with low sparsity
  - Declare primary and foreign keys so optimizer can use indexes in join operations
  - Declare indexes in join columns other than PK/FK

Conditional Expressions

- Normally expressed within WHERE or HAVING clauses of SQL statement
- Restricts output of query to only rows matching conditional criteria

Conditional Expressions (cont'd.)

- Common practices for efficient SQL:
  - Use simple columns or literals in conditionals
  - Numeric field comparisons are faster
  - Equality comparisons are faster than inequality
  - Transform conditional expressions to use literals
  - Write equality conditions first
  - AND: use condition most likely to be false first
  - OR: use condition most likely to be true first
  - Avoid NOT

Query Formulation

- Identify what columns and computations are required
- Identify source tables
- Determine how to join tables
- Determine what selection criteria is needed
- Determine in what order to display output

DBMS Performance Tuning

- Includes managing DBMS processes in primary memory and structures in physical storage
- DBMS performance tuning at server end focuses on setting parameters used for:
  - Data cache
  - SQL cache
  - Sort cache
  - Optimizer mode

DBMS Performance Tuning (cont'd.)

- Some general recommendations for creation of databases:
  - Use **RAID** (Redundant Array of Independent Disks) to provide balance between performance and fault tolerance
  - Minimize disk contention
  - Put high-usage tables in their own table spaces
  - Assign separate data files in separate storage volumes for indexes, system, high-usage tables

DBMS Performance Tuning (cont'd.)

- Some general recommendations for creation of databases (cont'd):
  - Take advantage of table storage organizations in database
  - Partition tables based on usage
  - Use denormalized tables where appropriate
  - Store computed and aggregate attributes in tables