Database performance tuning refers to a set of activities and procedures designed to reduce the response time of the database system—that is, to ensure that an end-user query is processed by the DBMS in the minimum amount of time.
- Good database performance starts with good database design.

PERFORMANCE TUNING: CLIENT AND SERVER

In general, database performance-tuning activities can be divided into those taking place on the client side and those taking place on the server side.

• On the client side, the objective is to generate a SQL query that returns the correct answer in the least amount of time, using the minimum amount of resources at the server end. The activities required to achieve that goal are commonly referred to as SQL performance tuning.
• On the server side, the DBMS environment must be properly configured to respond to clients’ requests in the fastest way possible, while making optimum use of existing resources. The activities required to achieve that goal are commonly referred to as DBMS performance tuning.

DBMS ARCHITECTURE

The architecture of a DBMS is represented by the processes and structures (in memory and in permanent storage) used to manage a database. Such processes collaborate with one another to perform specific functions. Figure below illustrates the basic DBMS architecture.
• All data in a database are stored in **data files**. A data file can contain rows from one single table, or it can contain rows from many different tables.

• A **table space** or **file group** is a logical grouping of several data files that store data with similar characteristics.

• The **data cache** or **buffer cache** is a shared, reserved memory area that stores the most recently accessed data blocks in RAM.

• The **SQL cache** or **procedure cache** is a shared, reserved memory area that stores the most recently executed SQL statements or PL/SQL procedures.

• To work with the data, the DBMS must retrieve the data from permanent storage (data files in which the data are stored) and place it in RAM (data cache).

• **An Input/output (I/O) request** is a low-level (read or write) data access operation to and from computer devices, such as memory, hard disks, video, and printers. The purpose of the I/O operation is to move data to and from various computer components and devices.

• The majority of performance-tuning activities focus on minimizing the number of I/O operations because user I/O operations is many times slower than reading data from the data cache.

Also illustrated in the same Figure are some typical DBMS processes. Although the number of processes and their names vary from vendor to vendor, the functionality is similar. The following processes are:

• **Listener.** The listener process listens for clients’ requests and handles the processing of the SQL requests to other DBMS processes. Once a request is received, the listener passes the request to the appropriate process.

• **User.** The DBMS creates a user process to manage each client session. Therefore, when you log on to DBMS, you are assigned a user process. This process handles all requests you submit to the server. There many user processes—at least one per each logged-in client.

• **Scheduler.** The scheduler process organizes the concurrent execution of SQL requests.

• **Lock manager.** This process manages all locks placed on database objects, including disk pages.

• **Optimizer.** The optimizer process analyzes SQL queries and finds the most efficient way to access the data.

**QUERY PROCESSING**

What happens at the DBMS server end when the client’s SQL statement is received? In simple terms, the DBMS processes a query in three phases:

1. Parsing. The DBMS parses the SQL query and chooses the most efficient access/execution plan.

2. Execution. The DBMS executes the SQL query using the chosen execution
3. Fetching. The DBMS fetches the data and sends the result set back to the client.

**SQL PARSING PHASE**

The SQL parsing activities are performed by the *query optimizer*, which analyzes the SQL query and finds the most efficient way to access the data. This process is the most time-consuming phase in query processing. Parsing a SQL query requires several steps, in which the SQL query is:

- Validated for syntax compliance.
- Validated against the data dictionary to ensure that tables and column names are correct.
- Validated against the data dictionary to ensure that the user has proper access rights.
- Analyzed and decomposed into more atomic components.
- Optimized through transformation into a fully equivalent but more efficient SQL query.
- Prepared for execution by determining the most efficient execution or access plan.

An *access plan* is the result of parsing an SQL statement; it contains the series of steps a DBMS will use to execute the query and to return the result set in the most efficient way.

**SQL EXECUTION PHASE**

In this phase, all I/O operations indicated in the access plan are executed. When the execution plan is run, the proper locks — if needed — are acquired for the data to be accessed, and the data are retrieved from the data files and placed in the
DBMSs data cache. All transaction management commands are processed during the parsing and execution phases of query processing.

**SQL FETCHING PHASE**

During the fetching phase, the rows of the resulting query result set are returned to the client. The DBMS might use temporary table space to store temporary data. In this stage, the database server coordinates the movement of the result set rows from the server cache to the client cache. For example, a given query result set might contain 9,000 rows; the server would send the first 100 rows to the client and then wait for the client to request the next set of rows, until the entire result set is sent to the client.

**QUERY PROCESSING BOTTLENECKS**

A query processing bottleneck is a delay introduced in the processing of an I/O operation that causes the overall system to slow down. There are five components that typically cause bottlenecks:

- CPU.
- RAM.
- Hard disk.
- Network.
- Application code.

**OPTIMIZER CHOICES**

Query optimization is the central activity during the parsing phase in query processing. In this phase, the DBMS must choose what indexes to use, how to perform join operations, what table to use first, and so on. Each DBMS has its own algorithms for determining the most efficient way to access the data. The query optimizer can operate in one of two modes:

- A rule-based optimizer uses preset rules and points to determine the best approach to execute a query.
- A cost-based optimizer uses sophisticated algorithms based on the statistics about the objects being accessed to determine the best approach to execute a query.