

**Database Principles:
Fundamentals of Design,
Implementation, and
Management
Tenth Edition**

*Chapter 14
Managing Database and SQL
Performance*

Objectives

In this chapter, you will learn:

- Basic database performance-tuning concepts
- How a DBMS processes SQL queries
- About the importance of indexes in query processing

Objectives (cont'd.)

- About the types of decisions the query optimizer has to make
- Some common practices used to write efficient SQL code
- How to formulate queries and tune the DBMS for optimal performance

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
14.1**

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 appli- cation 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

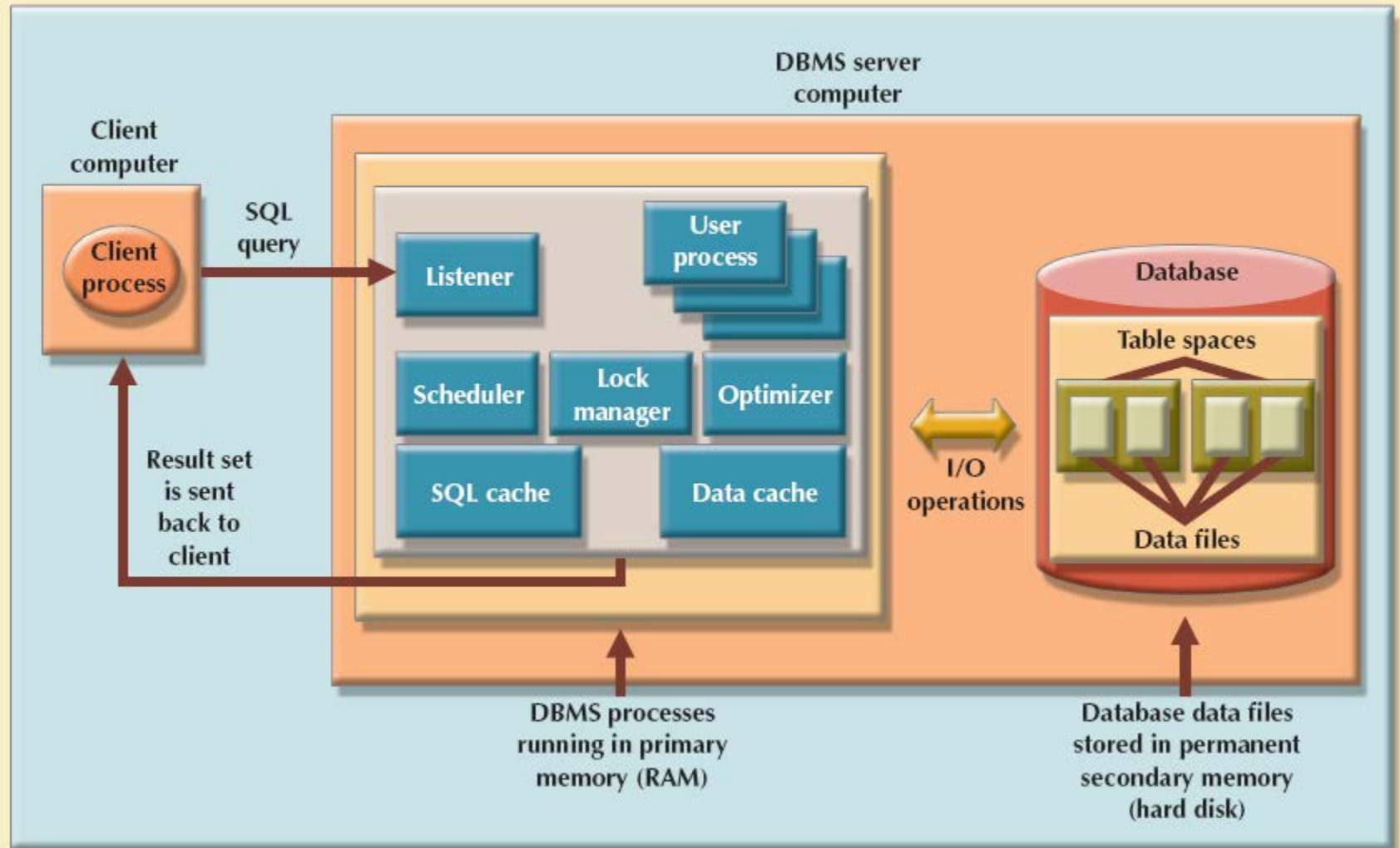
- Client side
 - Generate SQL query that returns correct answer in least amount of time
 - Using minimum amount of resources at server
 - SQL performance tuning
- Server side
 - DBMS environment configured to respond to clients' requests as fast as possible
 - Optimum use of existing resources
 - DBMS 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 14.1

Basic DBMS architecture



SOURCE: Course Technology/Cengage Learning

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 Query Optimization Modes

- Automatic query optimization
 - DBMS finds the most cost-effective access path without user intervention
- Manual query optimization
 - Requires that the optimization be selected and scheduled by the end user or programmer
- Static query optimization
 - Takes place at compilation time
- Dynamic query optimization
 - Takes place at execution time

Database Query Optimization Modes (cont'd.)

- Statistically based query optimization algorithm
 - Uses statistical information about the database
 - Dynamic statistical generation mode
 - Manual statistical generation mode
- Rule-based query optimization algorithm
 - Based on a set of user-defined rules to determine the best query access strategy

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
14.2**

Sample Database Statistics Measurements

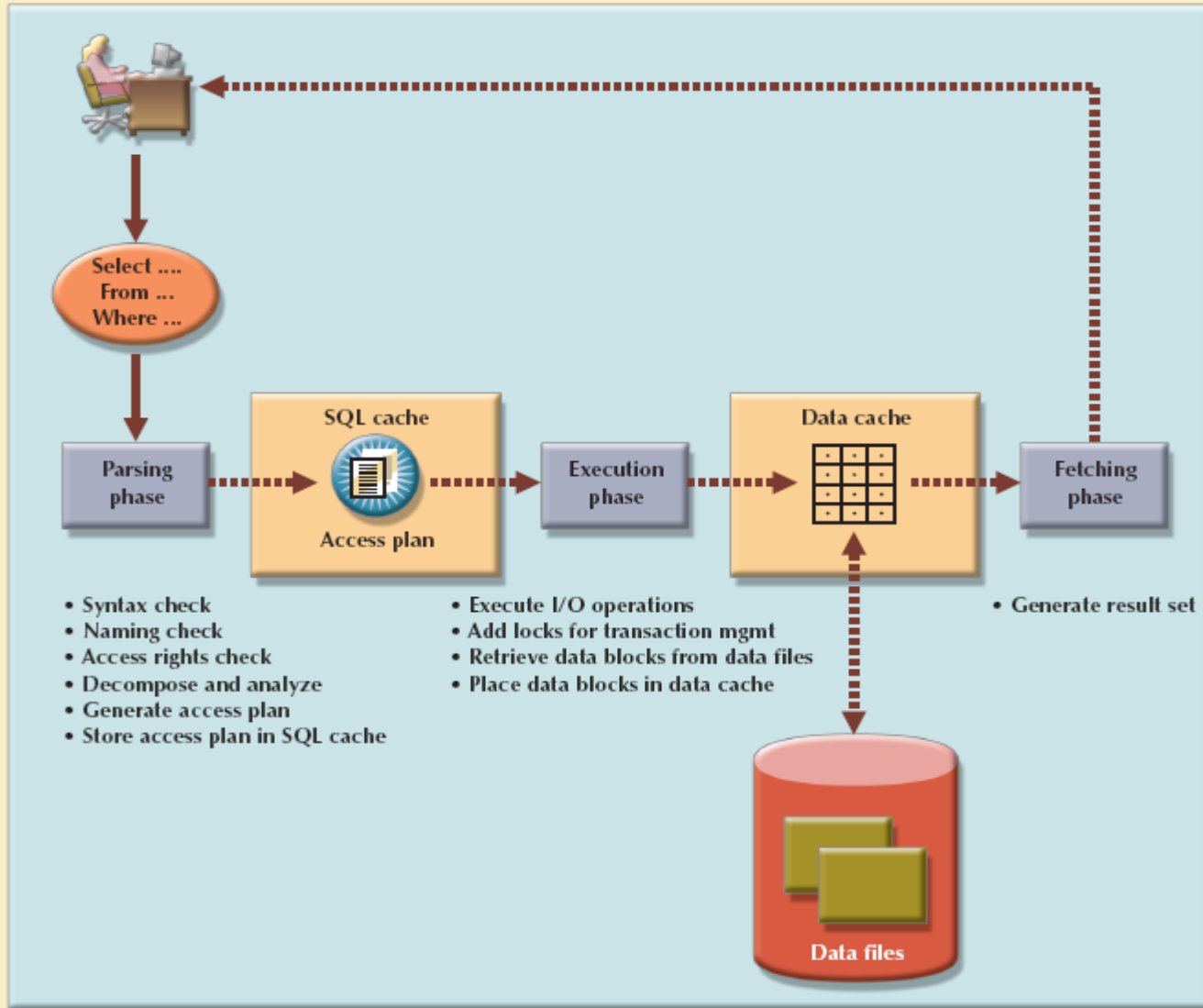
DATABASE OBJECT	SAMPLE MEASUREMENTS
Tables	Number of rows, number of disk blocks used, row length, number of columns in each row, number of distinct values in each column, maximum value in each column, minimum value in each column, and columns that have indexes
Indexes	Number and name of columns in the 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 14.2

Query processing



SOURCE: Course Technology/Cengage Learning

SQL Parsing Phase

- Break down query into smaller units
- Transform original SQL query into slightly different version of original SQL code
 - Fully equivalent
 - Optimized query results are always the same as original query
 - More efficient
 - Optimized query will almost always execute faster than original query

SQL Parsing Phase (cont'd.)

- Query optimizer analyzes SQL query and finds most efficient way to access data
 - Validated for syntax compliance
 - Validated against data dictionary
 - Tables and column names are correct
 - User has proper access rights
 - Analyzed and decomposed into components
 - Optimized
 - Prepared for execution

SQL Parsing Phase (cont'd.)

- Access plans are DBMS-specific
 - Translate client's SQL query into a series of complex I/O operations
 - Required to read the data from the physical data files and generate result set
- DBMS checks if access plan already exists for query in SQL cache
- DBMS reuses the access plan to save time
- If not, optimizer evaluates various plans
 - Chosen plan placed in SQL cache

**TABLE
14.3**

Sample DBMS Access Plan I/O Operations

OPERATION	DESCRIPTION
Table scan (full)	Reads the entire table sequentially, from the first row to the last, one row at a time (slowest)
Table access (row ID)	Reads a table row directly, using the row ID value (fastest)
Index scan (range)	Reads the index first to obtain the row IDs and then accesses the table rows directly (faster than a full table scan)
Index access (unique)	Used when a table has a unique index in a column
Nested loop	Reads and compares a set of values to another set of values, using a nested loop style (slow)
Merge	Merges two data sets (slow)
Sort	Sorts a data set (slow)

SQL Execution Phase

SQL Fetching Phase

- All I/O operations indicated in access plan are executed
 - Locks acquired
 - Data retrieved and placed in data cache
 - Transaction management commands processed
- Rows of resulting query result set are returned to client
- DBMS may use temporary table space to store temporary data

Query Processing Bottlenecks

- Delay introduced in the processing of an I/O operation that slows the system
 - CPU
 - RAM
 - Hard disk
 - Network
 - Application code

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

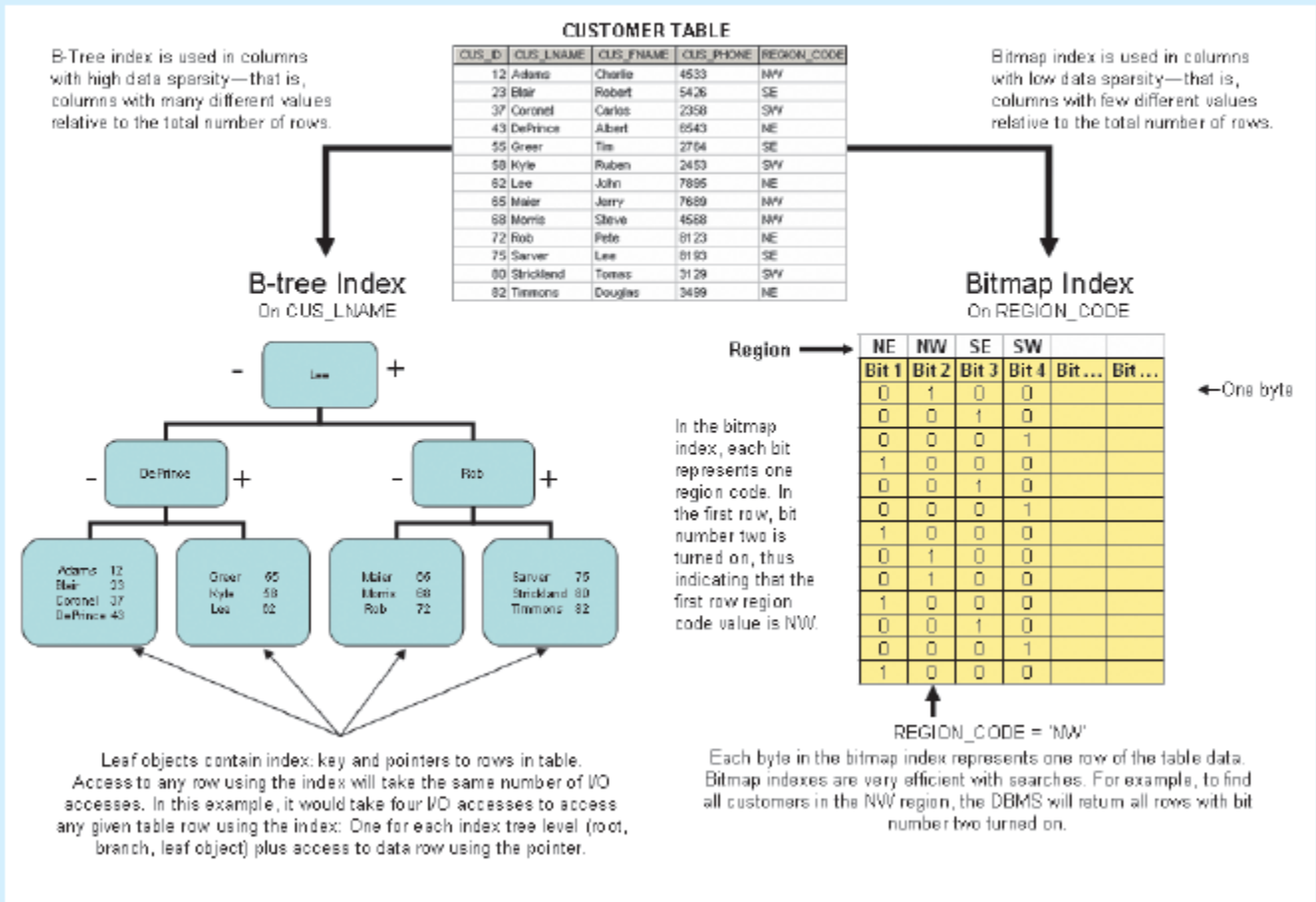
FIGURE 14.3

Index representation for the CUSTOMER table



SOURCE: Course Technology/Cengage Learning

FIGURE 14.4 B-tree and bitmap index representation



SOURCE: Course Technology/Cengage Learning

Optimizer Choices

- Rule-based optimizer
 - Preset rules and points
 - Rules assign a fixed cost to each operation
- Cost-based optimizer
 - Algorithms based on statistics about objects being accessed
 - Adds up processing cost, I/O costs, resource costs to derive total cost

**TABLE
14.4**

Comparing Access Plans and I/O Costs

PLAN	STEP	OPERATION	I/O OPERATIONS	I/O COST	RESULTING SET ROWS	TOTAL I/O COST
A	A1	Cartesian product (PRODUCT, VENDOR)	7,000 + 300	7,300	2,100,000	7,300
	A2	Select rows in A1 with matching vendor codes	2,100,000	2,100,000	7,000	2,107,300
	A3	Select rows in A2 with V_STATE = 'FL'	7,000	7,000	1,000	2,114,300
B	B1	Select rows in VENDOR with V_STATE = 'FL'	300	300	10	300
	B2	Cartesian Product (PRODUCT, B1)	7,000 + 10	7,010	70,000	7,310
	B3	Select rows in B2 with matching vendor codes	70,000	70,000	1,000	77,310

Using Hints to Affect Optimizer Choices

- Optimizer might not choose best plan
- Makes decisions based on existing statistics
 - Statistics may be old
 - Might choose less-efficient decisions
- Optimizer hints: special instructions for the optimizer embedded in the SQL command text

**TABLE
14.5**

Optimizer Hints

HINT	USAGE
ALL_ROWS	Instructs the optimizer to minimize the overall execution time—that is, to minimize the time needed to return all rows in the query result set. This hint is generally used for batch mode processes. For example: <pre>SELECT /*+ ALL_ROWS */ * FROM PRODUCT WHERE P_QOH < 10;</pre>
FIRST_ROWS	Instructs the optimizer to minimize the time needed to process the first set of rows—that is, to minimize the time needed to return only the first set of rows in the query result set. This hint is generally used for interactive mode processes. For example: <pre>SELECT /*+ FIRST_ROWS */ * FROM PRODUCT WHERE P_QOH < 10;</pre>
INDEX(name)	Forces the optimizer to use the P_QOH_NDX index to process this query. For example: <pre>SELECT /*+ INDEX(P_QOH_NDX) */ * FROM PRODUCT WHERE P_QOH < 10;</pre>

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 the:
 - 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, and high-usage tables

DBMS Performance Tuning (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

Query Optimization Example

- Example illustrates how query optimizer works
- Based on QOVENDOR and QOPRODUCT tables
- Uses Oracle SQL*Plus

FIGURE 14.5

Initial explain plan

```
Oracle SQL*Plus
File Edit Search Options Help
SQL> ANALYZE TABLE QOVENDOR COMPUTE STATISTICS;

Table analyzed.

SQL> EXPLAIN PLAN FOR SELECT * FROM QOVENDOR WHERE U_NAME LIKE 'B%' ORDER BY U_AREACODE;

Explained.

SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
-----
Plan hash value: 1837703589

-----
| Id | Operation          | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-----|-----|-----|-----|-----|-----|-----|
|  0 | SELECT STATEMENT   |           |      1 |    38 |    4 (25) | 00:00:01 |
|  1 |   SORT ORDER BY   |           |      1 |    38 |    4 (25) | 00:00:01 |
|* 2 |    TABLE ACCESS FULL | QOVENDOR |      1 |    38 |    3 (8)  | 00:00:01 |
-----

Predicate Information (identified by operation id):
-----
PLAN_TABLE_OUTPUT
-----

   2 - filter("U_NAME" LIKE 'B%')

14 rows selected.

SQL> |
```

SOURCE: Course Technology/Cengage Learning

**FIGURE
14.6**

Explain plan after index on V_AREACODE

```
Oracle SQL*Plus
File Edit Search Options Help
SQL> CREATE INDEX QOV_NDX1 ON QOVENDOR(V_AREACODE);
Index created.
SQL> ANALYZE TABLE QOVENDOR COMPUTE STATISTICS;
Table analyzed.
SQL> EXPLAIN PLAN FOR SELECT * FROM QOVENDOR WHERE V_NAME LIKE 'B%' ORDER BY V_AREACODE;
Explained.
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
PLAN_TABLE_OUTPUT
-----
Plan hash value: 2985289760
-----
| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |
-----|-----|-----|-----|-----|-----|-----|
| 0 | SELECT STATEMENT | | 1 | 38 | 2 (0)| 00:00:01 |
|* 1 | TABLE ACCESS BY INDEX ROWID | QOVENDOR | 1 | 38 | 2 (0)| 00:00:01 |
| 2 | INDEX FULL SCAN | QOV_NDX1 | 15 | | 1 (0)| 00:00:01 |
-----

Predicate Information (identified by operation id):
-----
   1 - filter(("V_NAME" LIKE 'B%'))

14 rows selected.

SQL> |
```

SOURCE: Course Technology/Cengage Learning

**FIGURE
14.7**

Explain plan after index on V_NAME

```
Oracle SQL*Plus
File Edit Search Options Help
SQL> CREATE INDEX QOV_NDX2 ON QOVENDOR(U_NAME);
Index created.
SQL> ANALYZE TABLE QOVENDOR COMPUTE STATISTICS;
Table analyzed.
SQL> EXPLAIN PLAN FOR SELECT * FROM QOVENDOR WHERE U_NAME LIKE 'B%' ORDER BY U_AREACODE;
Explained.
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
PLAN_TABLE_OUTPUT
-----
Plan hash value: 2305289760
-----
| Id | Operation | Name | Rows | Bytes | Cost (%CPU) | Time |
-----|-----|-----|-----|-----|-----|-----|
| 0 | SELECT STATEMENT | | 1 | 38 | 2 (0) | 00:00:01 |
|* 1 | TABLE ACCESS BY INDEX ROWID | QOVENDOR | 1 | 38 | 2 (0) | 00:00:01 |
| 2 | INDEX FULL SCAN | QOV_NDX1 | 15 | | 1 (0) | 00:00:01 |
-----

Predicate Information (identified by operation id):
-----
1 - filter("U_NAME" LIKE 'B%')

14 rows selected.

SQL>
```

SOURCE: Course Technology/Cengage Learning

FIGURE 14.8

Access plan using index on V_NAME

```
Oracle SQL*Plus
File Edit Search Options Help
SQL> EXPLAIN PLAN FOR SELECT V_NAME, P_CODE FROM QOVENDOR V, QOPRODUCT P
2 WHERE U_CODE = P.U_CODE AND U_NAME = 'ORDUA, Inc.';

Explained.

SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
-----
Plan hash value: 3956542569

-----
| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |
-----
| 0 | SELECT STATEMENT | | | | | | |
| * 1 | HASH JOIN | | | | | | |
| 2 | TABLE ACCESS BY INDEX ROWID | QOVENDOR | 1 | 17 | 2 (0) | 00:00:01 |
| * 3 | INDEX RANGE SCAN | QOV_NDX2 | 1 | | 1 (0) | 00:00:01 |
| 4 | TABLE ACCESS FULL | QOPRODUCT | 16 | 320 | 3 (0) | 00:00:01 |
-----

Predicate Information (identified by operation id):
-----
 1 - access("V"."U_CODE"="P"."U_CODE")
 3 - access("V"."NAME"='ORDUA, Inc.')

Note
-----
 - dynamic sampling used for this statement

21 rows selected.

SQL> |
```

SOURCE: Course Technology/Cengage Learning

FIGURE 14.9

Access plan using functions on indexed columns

```

Oracle SQL*Plus
File Edit Search Options Help
SQL> EXPLAIN PLAN FOR SELECT U_NAME, P_CODE FROM QOVENDOR U, QOPRODUCT P
2          WHERE U.U_CODE = P.U_CODE AND UPPER(U_NAME) = 'ORDVA, INC.';

Explained.

SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
-----
Plan hash value: 4061476548

-----
| Id | Operation                | Name                | Rows | Bytes | Cost (%CPU)| Time     |
-----|-----|-----|-----|-----|-----|-----|
|  0 | SELECT STATEMENT         |                     |      1 |    37 |       7 (15)| 00:00:01 |
|*  1 |  HASH JOIN               |                     |      1 |    37 |       7 (15)| 00:00:01 |
|  2 |    VIEW                  | index$_join$_001   |      1 |    17 |        3 (8)| 00:00:01 |
|*  3 |      HASH JOIN           |                     |      1 |    17 |       3 (8)| 00:00:01 |
|*  4 |        INDEX FAST FULL SCAN| QOV_NDX2           |      1 |    17 |        1 (8)| 00:00:01 |
|  5 |          INDEX FAST FULL SCAN| SYS_C005802       |      1 |    17 |        1 (8)| 00:00:01 |
|  6 |            TABLE ACCESS FULL| QOPRODUCT          |     16 |   320 |        3 (8)| 00:00:01 |
-----

Predicate Information (identified by operation id):
-----
   1 - access("U"."U_CODE"="P"."U_CODE")
   2 - filter(UPPER("U_NAME")='ORDVA, INC.')
   3 - access(ROWID=ROWID)
   4 - filter(UPPER("U_NAME")='ORDVA, INC.')

Note
-----
   - dynamic sampling used for this statement

25 rows selected.

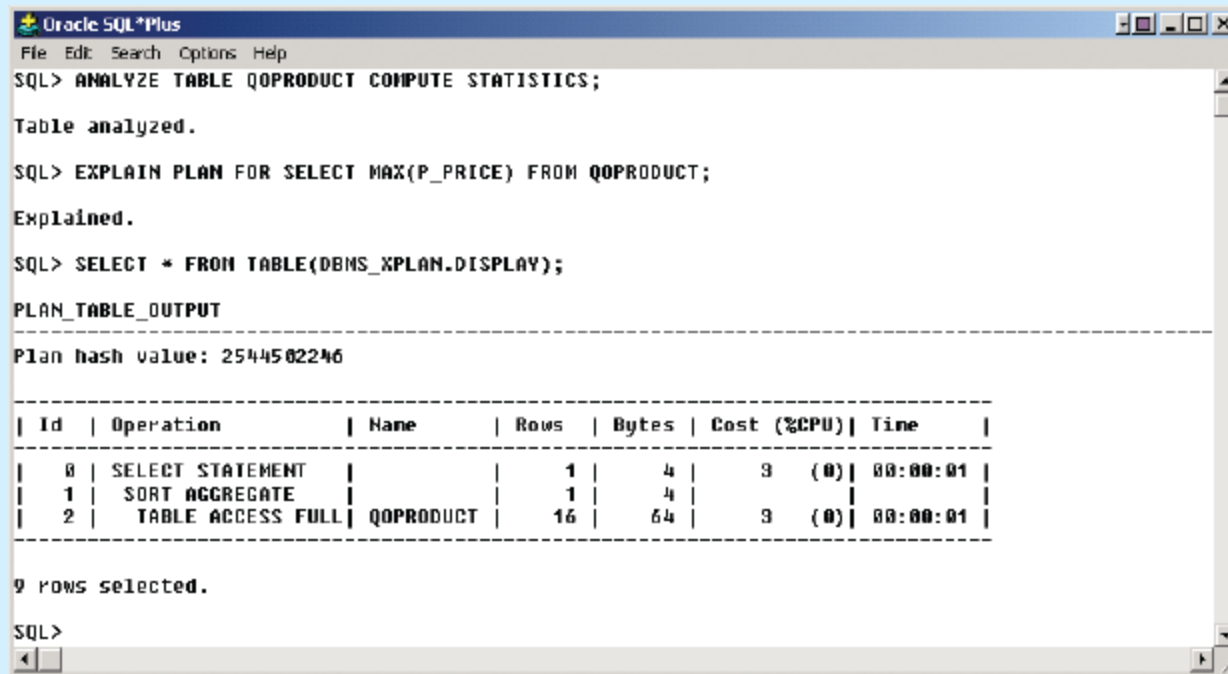
SQL>

```

SOURCE: Course Technology/Cengage Learning

FIGURE 14.10

First explain plan: aggregate function on a non-indexed column



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> ANALYZE TABLE QOPRODUCT COMPUTE STATISTICS;
Table analyzed.
SQL> EXPLAIN PLAN FOR SELECT MAX(P_PRICE) FROM QOPRODUCT;
Explained.
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
PLAN_TABLE_OUTPUT
-----
Plan hash value: 2544502246

-----
| Id | Operation          | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-----|-----|-----|-----|-----|-----|-----|
| 0  | SELECT STATEMENT   |            |      1 |      4 |    3 (0)| 00:00:01 |
| 1  |  SORT AGGREGATE    |            |      1 |      4 |    1 (0)| 00:00:01 |
| 2  |    TABLE ACCESS FULL| QOPRODUCT |     16 |    64 |    3 (0)| 00:00:01 |
-----

9 rows selected.

SQL>
```

SOURCE: Course Technology/Cengage Learning

**FIGURE
14.11**

Second explain plan: aggregate function on an indexed column

```
Oracle SQL*Plus
File Edit Search Options Help
SQL> CREATE INDEX QOP_NDX2 ON QOPRODUCT(P_PRICE);
Index created.
SQL> ANALYZE TABLE QOPRODUCT COMPUTE STATISTICS;
Table analyzed.
SQL> EXPLAIN PLAN FOR SELECT MAX(P_PRICE) FROM QOPRODUCT;
Explained.
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
PLAN_TABLE_OUTPUT
-----
Plan hash value: 3423609809
-----
| Id | Operation                    | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-----|-----|-----|-----|-----|-----|-----|
|  0 | SELECT STATEMENT              |           |      1 |      4 |    1 (0)| 00:00:01 |
|  1 |   SORT AGGREGATE              |           |      1 |      4 |           |         |
|  2 |    INDEX FULL SCAN (MIN/MAX) | QOP_NDX2  |     16 |     64 |    1 (0)| 00:00:01 |
-----
9 rows selected.
SQL>
```

SOURCE: Course Technology/Cengage Learning

FIGURE 14.12

Oracle 9i tools for query optimization

Oracle Enterprise Manager Console, Standalone

File Navigator Object Tools Configuration Help

ORACLE Enterprise Manager

General

Name: ORALAB.BIZLAB.MTSU.EDU

SQL Explain Plan TEACHER@ORALAB.BIZLAB.MTSU.EDU

File Edit View Drilldown Help

```

SELECT v_name, p_code
FROM qovendor v, qoproduct p
WHERE v.v_code = p.v_code
AND v_name = 'ORDVA, Inc.'
    
```

Plan COST ALL ROWS (Cost: 5)

TEACHER

```

SQL Cost: 5
SELECT STATEMENT
  Rows: 2
  Bytes: 0.057
  Hash Join Cost: 5
    Rows: 1
    Bytes: 0.017
    Table Access (BY INDEX ROWID) Cost: 2
      Rows: 1
      Index (RANGE SCAN) Cost: 1
        TEACHER.QOV_IDX2 INDEX
    Rows: 14
    Bytes: 0.164
    Table Access (FULL) Cost: 2
      TEACHER.QOPRODUCT TABLE ACCESS (FULL)
    
```

This plan step designates this statement as a SELECT statement.

Press F1 for Help

Commit is ... Execute time (s): 0.031 Rows returned: 3 Execute Close Help

SOURCE: Course Technology/Cengage Learning

Summary

- Database performance tuning
 - Refers to activities to ensure query is processed in minimum amount of time
- SQL performance tuning
 - Refers to activities on client side to generate SQL code
 - Returns correct answer in least amount of time
 - Uses minimum amount of resources at server end
- DBMS architecture is represented by processes and structures used to manage a database

Summary (cont'd.)

- Database statistics refers to measurements gathered by the DBMS
 - Describe snapshot of database objects' characteristics
- DBMS processes queries in three phases: parsing, execution, and fetching
- Indexes are crucial in process that speeds up data access

Summary (cont'd.)

- During query optimization, DBMS chooses:
 - Indexes to use, how to perform join operations, table to use first, etc.
- Hints change optimizer mode for current SQL statement
- SQL performance tuning deals with writing queries that make good use of statistics
- Query formulation deals with translating business questions into specific SQL code