Database Systems: Design, Implementation, and Management

Chapter 11
Database Performance Tuning and Query Optimization
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**
  - No performance tuning will make a poorly designed DB perform well
## General Guidelines for Better System Performance

<table>
<thead>
<tr>
<th>System Resources</th>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hardware</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>The fastest possible&lt;sup&gt;1&lt;/sup&gt; Dual-core CPU or higher</td>
<td>The fastest possible Multiple processors (quad-core technology)</td>
</tr>
<tr>
<td>RAM</td>
<td>The maximum possible</td>
<td>The maximum possible</td>
</tr>
<tr>
<td>Hard disk</td>
<td>Fast SATA/EIDE hard disk with sufficient free hard disk space</td>
<td>Multiple high-speed, high-capacity hard disks (SCSI/SATA/Firewire/Fibre Channel) in RAID configuration</td>
</tr>
<tr>
<td>Network</td>
<td>High-speed connection</td>
<td>High-speed connection</td>
</tr>
<tr>
<td><strong>Software</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operating system</td>
<td>Fine-tuned for best client application performance</td>
<td>Fine-tuned for best server application performance</td>
</tr>
<tr>
<td>Network</td>
<td>Fine-tuned for best throughput</td>
<td>Fine-tuned for best throughput</td>
</tr>
<tr>
<td>Application</td>
<td>Optimize SQL in client application</td>
<td>Optimize DBMS server for best performance</td>
</tr>
</tbody>
</table>

---

<sup>1</sup> The fastest possible refers to the system performance requirement as indicated by the user or application needs.
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
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
## Sample Database Statistics Measurements

<table>
<thead>
<tr>
<th>DATABASE OBJECT</th>
<th>SAMPLE MEASUREMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tables</td>
<td>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</td>
</tr>
<tr>
<td>Indexes</td>
<td>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</td>
</tr>
<tr>
<td>Environment Resources</td>
<td>Logical and physical disk block size, location and size of data files, and number of extends per data file</td>
</tr>
</tbody>
</table>
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
- Break down into smaller unit
- Transform into a more efficient equivalent
- Check if access plan already exists
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>
<thead>
<tr>
<th>OPERATION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table scan (full)</td>
<td>Reads the entire table sequentially, from the first row to the last, one row at a time (slowest)</td>
</tr>
<tr>
<td>Table access (row ID)</td>
<td>Reads a table row directly, using the row ID value (fastest)</td>
</tr>
<tr>
<td>Index scan (range)</td>
<td>Reads the index first to obtain the row IDs and then accesses the table rows directly (faster than a full table scan)</td>
</tr>
<tr>
<td>Index access (unique)</td>
<td>Used when a table has a unique index in a column</td>
</tr>
<tr>
<td>Nested loop</td>
<td>Reads and compares a set of values to another set of values, using a nested loop style (slow)</td>
</tr>
<tr>
<td>Merge</td>
<td>Merges two data sets (slow)</td>
</tr>
<tr>
<td>Sort</td>
<td>Sorts a data set (slow)</td>
</tr>
</tbody>
</table>
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
Database Query Optimization Modes

• Based on who does it:
  – 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

• Based on type of info used:
  – 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

- Can be gathered manually by DBA or automatically by DBMS
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
  – Like reading a book

• DBMSs determine best type of index to use

• If indexes are so cool, why not index all columns?
SELECT CUS_NAME, CUS_STATE
FROM CUSTOMER
WHERE CUS_STATE = 'FL'
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**

#### STATE_NDX INDEX

<table>
<thead>
<tr>
<th>Key</th>
<th>Row</th>
</tr>
</thead>
<tbody>
<tr>
<td>AZ</td>
<td>2</td>
</tr>
<tr>
<td>FL</td>
<td>5</td>
</tr>
<tr>
<td>FL</td>
<td>8</td>
</tr>
<tr>
<td>FL</td>
<td>13245</td>
</tr>
<tr>
<td>FL</td>
<td>14786</td>
</tr>
</tbody>
</table>

#### CUSTOMER TABLE

(14,786 rows)

<table>
<thead>
<tr>
<th>Row ID</th>
<th>CHS_CODE</th>
<th>CHS_NAME</th>
<th>CHS FNAME</th>
<th>CHS_INITIAL</th>
<th>CHS_AREACODE</th>
<th>CHS_PHONE</th>
<th>CHS_STATE</th>
<th>CHS_BALANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10010</td>
<td>Ramas</td>
<td>Alfred</td>
<td>A</td>
<td>815</td>
<td>844-2573</td>
<td>FL</td>
<td>$0.00</td>
</tr>
<tr>
<td>2</td>
<td>10011</td>
<td>Dunne</td>
<td>Leona</td>
<td>K</td>
<td>713</td>
<td>894-1238</td>
<td>AZ</td>
<td>$0.00</td>
</tr>
<tr>
<td>3</td>
<td>10012</td>
<td>Smith</td>
<td>Kathy</td>
<td>W</td>
<td>815</td>
<td>894-2385</td>
<td>TX</td>
<td>$345.86</td>
</tr>
<tr>
<td>4</td>
<td>10013</td>
<td>Olowska</td>
<td>Paul</td>
<td>F</td>
<td>615</td>
<td>894-2180</td>
<td>AZ</td>
<td>$536.75</td>
</tr>
<tr>
<td>5</td>
<td>10014</td>
<td>Orlando</td>
<td>Myron</td>
<td>F</td>
<td>615</td>
<td>222-1672</td>
<td>NY</td>
<td>$0.00</td>
</tr>
<tr>
<td>6</td>
<td>10015</td>
<td>O'Brian</td>
<td>Amy</td>
<td>B</td>
<td>713</td>
<td>442-3381</td>
<td>NY</td>
<td>$0.00</td>
</tr>
<tr>
<td>7</td>
<td>10016</td>
<td>Brown</td>
<td>James</td>
<td>G</td>
<td>615</td>
<td>297-1229</td>
<td>FL</td>
<td>$221.19</td>
</tr>
<tr>
<td>8</td>
<td>10017</td>
<td>Williams</td>
<td>George</td>
<td>G</td>
<td>615</td>
<td>290-2566</td>
<td>FL</td>
<td>$768.93</td>
</tr>
<tr>
<td>9</td>
<td>10018</td>
<td>Farmos</td>
<td>Anne</td>
<td>G</td>
<td>713</td>
<td>382-7185</td>
<td>TX</td>
<td>$216.65</td>
</tr>
<tr>
<td>10</td>
<td>10019</td>
<td>Smith</td>
<td>Olette</td>
<td>K</td>
<td>615</td>
<td>297-3009</td>
<td>AZ</td>
<td>$0.00</td>
</tr>
<tr>
<td>13245</td>
<td>23120</td>
<td>Veron</td>
<td>George</td>
<td>D</td>
<td>415</td>
<td>231-9872</td>
<td>FL</td>
<td>$675.00</td>
</tr>
<tr>
<td>14786</td>
<td>24560</td>
<td>Suarez</td>
<td>Victor</td>
<td>G</td>
<td>435</td>
<td>342-9076</td>
<td>FL</td>
<td>$342.00</td>
</tr>
</tbody>
</table>

**SOURCE:** Course Technology/Cengage Learning
B-tree and bitmap index representation

B-Tree index is used in columns with high data sparsity—that is, columns with many different values relative to the total number of rows.

Bitmap index is used in columns with low data sparsity—that is, columns with few different values relative to the total number of rows.

### CUSTOMER TABLE

<table>
<thead>
<tr>
<th>CUS_ID</th>
<th>CUS_LNAME</th>
<th>CUS_FNAME</th>
<th>CUS_PHONE</th>
<th>REGION_CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Adams</td>
<td>Charlie</td>
<td>4533</td>
<td>NW</td>
</tr>
<tr>
<td>23</td>
<td>Blair</td>
<td>Robert</td>
<td>5426</td>
<td>SE</td>
</tr>
<tr>
<td>37</td>
<td>Corney</td>
<td>Carlos</td>
<td>2359</td>
<td>SW</td>
</tr>
<tr>
<td>43</td>
<td>DePrince</td>
<td>Albert</td>
<td>6543</td>
<td>NE</td>
</tr>
<tr>
<td>56</td>
<td>Green</td>
<td>Tim</td>
<td>2764</td>
<td>SE</td>
</tr>
<tr>
<td>58</td>
<td>Kyle</td>
<td>Ruben</td>
<td>2453</td>
<td>SW</td>
</tr>
<tr>
<td>62</td>
<td>Lee</td>
<td>John</td>
<td>7895</td>
<td>NE</td>
</tr>
<tr>
<td>65</td>
<td>Maior</td>
<td>Jerry</td>
<td>7898</td>
<td>NW</td>
</tr>
<tr>
<td>68</td>
<td>Morris</td>
<td>Steve</td>
<td>4568</td>
<td>SW</td>
</tr>
<tr>
<td>72</td>
<td>Rob</td>
<td>Pete</td>
<td>8123</td>
<td>NE</td>
</tr>
<tr>
<td>75</td>
<td>Server</td>
<td>Lee</td>
<td>8193</td>
<td>SE</td>
</tr>
<tr>
<td>80</td>
<td>Strickland</td>
<td>Thomas</td>
<td>3129</td>
<td>SW</td>
</tr>
<tr>
<td>82</td>
<td>Timmons</td>
<td>Douglas</td>
<td>3499</td>
<td>NE</td>
</tr>
</tbody>
</table>

### B-tree Index

On CUS_LNAME

- Lee
  - DePrince
    - Adams 12
    - Blair 23
    - Corney 37
    - DePrince 43
  - Green 56
  - Lee 62
  - Maior 65
  - Morris 68
  - Rob 72

Leaf objects contain index key and pointers to rows in table. Access to any row using the index will take the same number of I/O accesses. In this example, it would take four I/O accesses to access any given table row using the index: One for each index tree level (root, branch, leaf object) plus access to data row using the pointer.

### Bitmap Index

On REGION_CODE

- Region
  - Bit 1 0
  - Bit 2 1
  - Bit 3 0
  - Bit 4 0
  - Bit 5 0
  - Bit 6 0
  - Bit 7 0
  - Bit 8 0

In the bitmap index, each bit represents one region code. In the first row, bit number two is turned on, thus indicating that the first row region code value is NW.

REGION_CODE = 'NW'

Each byte in the bitmap index represents one row of the table data. Bitmap indexes are very efficient with searches. For example, to find all customers in the NW region, the DBMS will return all rows with bit number two turned on.

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
### SELECT

```
P_CODE, P_DESCRIP, P_PRICE, V_NAME, V_STATE
```

### FROM

```
PRODUCT AS P, VENDOR AS V
```

### WHERE

```
P.V_CODE = V.V_CODE AND V_STATE = 'FL'
```

---

**TABLE 11.4**

Comparing Access Plans and I/O Costs

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

**ASSUME:** PRODUCT has 7000 rows, VENDOR has 300 rows, 10 vendors are in FL, 1000 products come from those vendors.
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 11.5

**Optimizer Hints**

<table>
<thead>
<tr>
<th>HINT</th>
<th>USAGE</th>
</tr>
</thead>
</table>
| 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: SELECT  /*+ ALL_ROWS */ *
|                 | FROM PRODUCT                                                                                                                                            |
|                 | WHERE P_QOH < 10;                                                                                                                                                                      |
| 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: SELECT  /*+ FIRST_ROWS */ *
|                 | FROM PRODUCT                                                                                                                                            |
|                 | WHERE P_QOH < 10;                                                                                                                                                                      |
| INDEX(name)     | Forces the optimizer to use the P_QOH_NDX index to process this query. For example: SELECT  /*+ INDEX(P_QOH_NDX) */ *
|                 | FROM PRODUCT                                                                                                                                            |
|                 | WHERE P_QOH < 10;                                                                                                                                                                      |
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

• Measure of how likely an index will be used

• General guidelines for indexes:
  – Create indexes for each attribute in WHERE, HAVING, ORDER BY, or GROUP BY
  – Do not use in small tables or columns with low sparsity (few unique values e.g.: CUS_GENDER)
  – 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

• Common practices for efficient SQL:
  – Use *simple columns* or literals in conditionals
  – Numeric field comparisons are faster
  – Equality comparisons are faster than inequality
  – Write *equality conditions* first
  – **AND**: use condition most likely to be false first
  – **OR**: use condition most likely to be true first
  – Avoid **NOT**: not (EMP_GENDER='M') ➔ EMP_GENDER='F'
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
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 11.5  Initial explain plan

```
SQL> ANALYZE TABLE QUQENDOR COMPUTE STATISTICS;
Table analyzed.
SQL> EXPLAIN PLAN FOR SELECT * FROM QUQENDOR WHERE V_NAME LIKE 'B%' ORDER BY V_AREACODE;
Explained.
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
PLAN_TABLE_OUTPUT
----------------------------------------------
Plan hash value: 1837783589

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>1</td>
<td>38</td>
<td>4 (25)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>1</td>
<td>SORT ORDER BY</td>
<td></td>
<td>1</td>
<td>38</td>
<td>4 (25)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>*</td>
<td>TABLE ACCESS FULL</td>
<td>QUQENDOR</td>
<td>1</td>
<td>38</td>
<td>3 (0)</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):

PLAN_TABLE_OUTPUT

| 2   | filter("V_NAME" LIKE 'B%')

14 rows selected.
```
FIGURE 11.6 Explain plan after index on V_AREACODE

```
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 '8%' ORDER BY V_AREACODE;
Explained.
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
Plan hash value: 2305289760

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>1</td>
<td>38</td>
<td>2 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>*</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>QOvendor</td>
<td>1</td>
<td>38</td>
<td>2 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>2</td>
<td>INDEX FULL SCAN</td>
<td>QOV_NDX1</td>
<td>15</td>
<td>1</td>
<td>1 (0)</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):

1 - Filter("V_NAME" LIKE '8%')
14 rows selected.
```
FIGURE 11.7

Explain plan after index on V_NAME

```
SQL> CREATE INDEX QOV_NDX2 ON QOVEDOR(V_NAME);
Index created.
SQL> ANALYZE TABLE QOVEDOR COMPUTE STATISTICS;
Table analyzed.
SQL> EXPLAIN PLAN FOR SELECT * FROM QOVEDOR WHERE V_NAME LIKE '%B%' ORDER BY V_AREACODE;
Explained.
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
PLAN_TABLE_OUTPUT
Plan hash value: 2305289760

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>1</td>
<td>38</td>
<td>2 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>*</td>
<td>TABLE ACCESS BY INDEX ROWID QOVEDOR</td>
<td></td>
<td>1</td>
<td>38</td>
<td>2 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>2</td>
<td>INDEX FULL SCAN</td>
<td>QOV_NDX1</td>
<td>15</td>
<td></td>
<td>1 (0)</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):

1 - Filter("V_NAME" LIKE '%B%')
```

14 rows selected.

SOURCE: Course Technology/Cengage Learning
FIGURE 11.8  Access plan using index on V_NAME

```sql
SQL> EXPLAIN PLAN FOR SELECT U_NAME, P_CODE FROM QOVEDOR V, QOPRODUCT P
WHERE U.V_CODE = P.V_CODE AND U_NAME = 'ORDVA, Inc.';

Explained.

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

PLAN_TABLE_OUTPUT

Plan hash value: 3956542569

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>2</td>
<td>74</td>
<td>6 (17)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>* 1</td>
<td>HASH JOIN</td>
<td></td>
<td>2</td>
<td>74</td>
<td>6 (17)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>2</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>QOVEDOR</td>
<td>1</td>
<td>17</td>
<td>2 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>* 3</td>
<td>INDEX RANGE SCAN</td>
<td>QOU_INDEX2</td>
<td>1</td>
<td></td>
<td>1 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>4</td>
<td>TABLE ACCESS FULL</td>
<td>QOPRODUCT</td>
<td>16</td>
<td>320</td>
<td>3 (0)</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):

1 - access("U"."V_CODE"="P"."V_CODE")
3 - access("U_NAME"='ORDVA, Inc.')

Note
---
dynamic sampling used for this statement
21 rows selected.
```

SOURCE: Course Technology/Cengage Learning
Access plan using functions on indexed columns

```sql
SQL> EXPLAIN PLAN FOR SELECT U_NAME, P_CODE FROM QVENDOR U, QOPRODUCT P
    2   WHERE U.V_CODE = P.U_CODE AND UPPER(U_NAME) = 'ORDUA, INC.';
```

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

```
PLAN_TABLE_OUTPUT
-------------------------------------------------------------
Plan hash value: 4061476548

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>1</td>
<td>37</td>
<td>7 (15)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>1</td>
<td>HASH JOIN</td>
<td>Index$<em>.join$</em>.001</td>
<td>1</td>
<td>37</td>
<td>7 (15)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>2</td>
<td>VIEW</td>
<td></td>
<td>1</td>
<td>17</td>
<td>3 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>3</td>
<td>HASH JOIN</td>
<td></td>
<td>1</td>
<td>17</td>
<td>1 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>4</td>
<td>INDEX FAST FULL SCAN</td>
<td>QOU_INDEX2</td>
<td>1</td>
<td>17</td>
<td>1 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>5</td>
<td>INDEX FAST FULL SCAN</td>
<td>SYS_INDEX02</td>
<td>1</td>
<td>17</td>
<td>1 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>6</td>
<td>TABLE ACCESS FULL</td>
<td>QOPRODUCT</td>
<td>16</td>
<td>320</td>
<td>3 (0)</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):  

1 - access("U"."U_CODE"="P"."U_CODE")  
2 - filter(UPPER("U_NAME")='ORDUA, INC.')  
3 - access(ROWID=ROWID)  
4 - filter(UPPER("U_NAME")='ORDUA, INC.')

Note  
- dynamic sampling used for this statement

25 rows selected.
```

SOURCE: Course Technology/Cengage Learning
FIGURE 11.10  First explain plan: aggregate function on a non-indexed column

```
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

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>00:00:01</td>
</tr>
<tr>
<td>1</td>
<td>SORT AGGREGATE</td>
<td></td>
<td>1</td>
<td>4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>TABLE ACCESS FULL</td>
<td>QOPRODUCT</td>
<td>16</td>
<td>64</td>
<td>3</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

9 rows selected.
```

SOURCE: Course Technology/Cengage Learning

Database Systems 49
FIGURE 11.11 Second explain plan: aggregate function on an indexed column

```
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

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>1</td>
<td>4</td>
<td>1 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>1</td>
<td>SORT AGGREGATE</td>
<td></td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>INDEX FULL SCAN (MIN/MAX) QOP_NDX2</td>
<td></td>
<td>16</td>
<td>64</td>
<td>1 (0)</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

9 rows selected.
```

SOURCE: Course Technology/Cengage Learning
Oracle 9i tools for query optimization
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
Database Systems: Design, Implementation, and Management

Chapter 16
Database Administration and Security
Objectives

In this chapter, students will learn:

• That data are a valuable business asset requiring careful management
• How a database plays a critical role in an organization
• That the introduction of a DBMS has important technological, managerial, and cultural consequences for an organization
Objectives (cont’d.)

- What the database administrator’s managerial and technical roles are
- About data security, database security, and the information security framework
- About several database administration tools and strategies
- How various technical tasks of database administration are performed with Oracle
Data as a Corporate Asset

• Data:
  – Valuable asset that requires careful management
  – Valuable resource that translates into information

• Accurate, timely information triggers actions that enhance company’s position and generate wealth
Data as a Corporate Asset (cont’d.)

• Dirty data
  – Data that suffer from inaccuracies and inconsistencies
  – Threat to organizations
Dirty data: Data that suffer from inaccuracies and inconsistencies
Data as a Corporate Asset (cont’d.)

• Data quality
  – Comprehensive approach to ensuring the accuracy, validity, and timeliness of the data

• Data profiling software
  – Consists of programs that gather statistics and analyze existing data sources

• Master data management (MDM) software
  – Helps prevent dirty data by coordinating common data across multiple systems.
The Need for and Role of Databases in an Organization

- Database’s predominant role is to support managerial decision making at all levels
- DBMS facilitates:
  - Interpretation and presentation of data
  - Distribution of data and information
  - Preservation and monitoring of data
  - Control over data duplication and use
- Three levels to organization management:
  - Top, middle, operational
Introduction of a Database: Special Considerations

• Introduction of a DBMS is likely to have a profound impact
  – Might be positive or negative, depending on how it is administered

• Three aspects to DBMS introduction:
  – Technological
  – Managerial
  – Cultural

• One role of DBA department is to educate end users about system uses and benefits
The Evolution of the Database Administration Function

• Data administration has its roots in the old, decentralized world of the file system
• Advent of DBMS produced new level of data management sophistication
  – DP department evolved into information systems (IS) department
• Data management became increasingly complex
  – Development of database administrator (DBA) function
FIGURE 15.3 The placement of the DBA function

Line Authority Position
- Information systems (IS)
  - Application development
  - Database operations
  - Database administration

Staff Consulting Position
- Information systems (IS)
  - Application development
  - Database operations
  - Database administration

SOURCE: Course Technology/Cengage Learning
A DBA functional organization

- Planning
- Design
- Implementation
- Operations
- Training
  - Conceptual
  - Logical
  - Physical
  - Testing

SOURCE: Course Technology/Cengage Learning
The Database Environment’s Human Component

• Even most carefully crafted database system cannot operate without human component
• Effective data administration requires both technical and managerial skills
• DA must set data administration goals
• DBA is focal point for data/user interaction
• Need for diverse mix of skills
The DBA’s Managerial Role

• DBA responsible for:
  – Coordinating, monitoring, allocating resources
    • Resources include people and data
  – Defining goals and formulating strategic plans
• Interacts with end user by providing data and information
• Enforces policies, standards, procedures
The DBA’s Managerial Role (cont’d.)

- Manages security, privacy, integrity
- Ensures data can be fully recovered
  - In large organizations, database security officer (DSO) responsible for disaster management
- Ensures data is distributed appropriately
  - Makes it easy for authorized end users to access the database
<table>
<thead>
<tr>
<th>DBA ACTIVITY</th>
<th>DBA SERVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td>End-user support</td>
</tr>
<tr>
<td>Organizing</td>
<td>Policies, procedures, and standards</td>
</tr>
<tr>
<td>Testing</td>
<td>Data security, privacy, and integrity</td>
</tr>
<tr>
<td>Monitoring</td>
<td>Data backup and recovery</td>
</tr>
<tr>
<td>Delivering</td>
<td>Data distribution and use</td>
</tr>
</tbody>
</table>
The DBA’s Technical Role

• Evaluates, selects, and installs DBMS and related utilities
• Designs and implements databases and applications
• Tests and evaluates databases and applications
The DBA’s Technical Role (cont’d.)

- Operates DBMS, utilities, and applications
- Trains and supports users
- Maintains DBMS, utilities, and applications
The DBA’s Role in the Cloud

• Cloud services provide:
  – DBMS installation and updates
  – Server/network management
  – Backup and recovery operations

• DBA’s managerial role is largely unchanged
Security

• Securing data entails securing overall information system architecture
  – Confidentiality: data protected against unauthorized access
  – Integrity: keep data consistent and free of errors or anomalies
  – Availability: any-time accessibility of data by authorized users for authorized purposes

• Security policy: collection of standards, policies, procedures to guarantee security
  – Ensures auditing and compliance
Security Vulnerabilities

- **Security vulnerability**: weakness in a system component
  - Could allow unauthorized access or cause service disruptions

- **Security threat**: imminent security violation
  - Could occur at any time

- **Security breach** yields a database whose integrity is either:
  - Preserved
  - Corrupted

### Security Vulnerability

- The user sets a blank password.
- The password is short or includes a birth date.
- The user leaves the office door open all the time.
- The user leaves payroll information on the screen for long periods of time.
<table>
<thead>
<tr>
<th>SYSTEM COMPONENT</th>
<th>SECURITY VULNERABILITY</th>
<th>SECURITY MEASURES</th>
</tr>
</thead>
</table>
| People            | • The user sets a blank password.  
• The password is short or includes a birth date.  
• The user leaves the office door open all the time.  
• The user leaves payroll information on the screen for long periods of time. | • Enforce complex password policies.  
• Use multilevel authentication.  
• Use security screens and screen savers.  
• Educate users about sensitive data.  
• Install security cameras.  
• Use automatic door locks. |
| Workstation and servers | • The user copies data to a flash drive.  
• The workstation is used by multiple users.  
• A power failure crashes the computer.  
• Unauthorized personnel can use the computer.  
• Sensitive data are stored on a laptop computer.  
• Data are lost due to a stolen hard disk or laptop.  
• A natural disaster occurs. | • Use group policies to restrict the use of flash drives.  
• Assign user access rights to workstations.  
• Install uninterruptible power supplies (UPSs).  
• Add security locks to computers.  
• Implement a kill switch for stolen laptops.  
• Create and test data backup and recovery plans.  
• Insure the system against natural disasters—use co-location strategies. |
| Operating system  | • Buffer overflow attacks  
• Virus attacks  
• Root kits and worm attacks  
• Denial-of-service attacks  
• Trojan horses  
• Spyware applications  
• Password crackers | • Apply OS security patches and updates.  
• Apply application server patches.  
• Install antivirus and antispyware software.  
• Enforce audit trails on the computers.  
• Perform periodic system backups.  
• Install only authorized applications.  
• Use group policies to prevent unauthorized installations. |
<table>
<thead>
<tr>
<th>Applications</th>
<th>Network</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Application bugs</td>
<td>- IP spoofing</td>
<td>- Data shares are open to all users.</td>
</tr>
<tr>
<td>- buffer overflow</td>
<td>- Packet sniffers</td>
<td>- Data can be accessed remotely.</td>
</tr>
<tr>
<td>- SQL injection,</td>
<td>- Hacker attacks</td>
<td>- Data can be deleted from a shared</td>
</tr>
<tr>
<td>- session hijacking,</td>
<td>- Clear passwords</td>
<td>resource.</td>
</tr>
<tr>
<td>- etc.</td>
<td>- on network</td>
<td></td>
</tr>
<tr>
<td>- Application</td>
<td></td>
<td>- Implement file system security.</td>
</tr>
<tr>
<td>- vulnerabilities</td>
<td></td>
<td>- Implement share access security.</td>
</tr>
<tr>
<td>- cross-site</td>
<td></td>
<td>- Use access permission.</td>
</tr>
<tr>
<td>- scripting,</td>
<td></td>
<td>- Encrypt data at the file system or</td>
</tr>
<tr>
<td>- nonvalidated</td>
<td></td>
<td>database level.</td>
</tr>
<tr>
<td>- inputs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- E-mail attacks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- spamming,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- phishing,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- etc.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Social engineering</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- e-mails</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Test application</td>
<td>- Install firewalls.</td>
<td>- Implement file system security.</td>
</tr>
<tr>
<td>- programs</td>
<td>- Use virtual</td>
<td>- Implement share access security.</td>
</tr>
<tr>
<td>- extensively</td>
<td>- private networks</td>
<td>- Use access permission.</td>
</tr>
<tr>
<td>- Build safeguards</td>
<td>- (VPNs).</td>
<td>- Encrypt data at the file system or</td>
</tr>
<tr>
<td>- into code.</td>
<td>- Use intrusion</td>
<td>database level.</td>
</tr>
<tr>
<td>- Do extensive</td>
<td>- detection</td>
<td></td>
</tr>
<tr>
<td>- vulnerability</td>
<td>- systems (IDSs).</td>
<td></td>
</tr>
<tr>
<td>- testing in</td>
<td>- Use network</td>
<td></td>
</tr>
<tr>
<td>- applications.</td>
<td>- access control</td>
<td></td>
</tr>
<tr>
<td>- Install spam filters and antivirus software for e-mail systems.</td>
<td>- Use network activity monitoring.</td>
<td></td>
</tr>
<tr>
<td>- Use secure coding</td>
<td>- <a href="http://www.owasp.org">www.owasp.org</a>).</td>
<td></td>
</tr>
<tr>
<td>- techniques (see</td>
<td>- Educate users</td>
<td></td>
</tr>
<tr>
<td>- <a href="http://www.owasp.org">www.owasp.org</a>).</td>
<td>- about social</td>
<td></td>
</tr>
<tr>
<td>- Educate users</td>
<td>- engineering</td>
<td></td>
</tr>
<tr>
<td>- about social</td>
<td>- attacks.</td>
<td></td>
</tr>
<tr>
<td>- engineering</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- attacks.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Database Security

- Refers to the use of DBMS features and other measures to comply with security requirements

- DBA secures DBMS from installation through operation and maintenance

- Authorization management
  - User access management (create users with passwords)
  - DBMS access control (grant and revoke privileges)
  - View definition
  - DBMS usage monitoring (audit log)
Database Administration Tools

- Data dictionary
- CASE tools
The Data Dictionary

• Two main types of data dictionaries:
  – Integrated
  – Standalone

• Active data dictionary is automatically updated by the DBMS with every database access

• Passive data dictionary requires running a batch process

• Main function: store description of all objects that interact with database
The Data Dictionary (cont’d.)

• Data dictionary that includes data external to DBMS becomes flexible tool
  – Enables use and allocation of all of an organization’s information
• Metadata is often the basis for monitoring database use
  – Also for assigning access rights to users
• DBA uses data dictionary to support data analysis and design
CASE Tools

• Computer-aided systems engineering
  – Automated framework for SDLC
  – Structured methodologies and powerful graphical interfaces
• Front-end CASE tools provide support for planning, analysis, and design phases
• Back-end CASE tools provide support for coding and implementation phases
CASE Tools (cont’d.)

• Typical CASE tool has five components
  – Graphics for diagrams
  – Screen painters and report generators
  – Integrated repository
  – Analysis segment
  – Program documentation generator
FIGURE 15.7
An example of a CASE tool: Visio Professional

Main menu

Modeling options

Completed ERD
Developing a Data Administration Strategy

• Information engineering (IE) translates strategic goals into data and applications

• Information systems architecture (ISA) is the output of IE process

• Implementing IE is a costly process
  – Provides a framework that includes use of computerized, automated, and integrated tools

• Success of information systems strategy depends on critical success factors
  – Managerial, technological, and corporate culture
The DBA at Work: Using Oracle for Database Administration

• Technical tasks handled by the DBA in a specific DBMS:
  – Creating and expanding database storage structures
  – Managing database objects
  – Managing end-user database environment
  – Customizing database initialization parameters

• All DBMS vendors provide programs to perform database administrative tasks
Oracle Database Administration Tools
The Default Login

• Must connect to the database to perform administrative tasks
  – Username with administrative privileges
• Oracle automatically creates SYSTEM and SYS user IDs with administrative privileges
• Define preferred credentials by clicking on Preferences link, then Preferred Credentials
• Username and passwords are database-specific
Ensuring that the RDBMS Starts Automatically

- DBA ensures database access is automatically started when computer turned on
- A service is a Windows system name for a special program that runs automatically
  - Part of the operating system
- Database instance: separate location in memory reserved to run the database
  - May have several databases running in memory at the same time
Creating Tablespaces and Datafiles

- Database composed of one or more tablespaces
- Tablespace is a logical storage space
  - Physically stored in one or more datafiles
- Datafile physically stores the database’s data
  - Each datafile can reside in a different directory on the hard disk
- Database has 1:M relationship with tablespaces
- Tablespace has 1:M relationship with datafiles
Figure 15.12: The Oracle Storage Manager

The image shows a screenshot of the Oracle Enterprise Manager (OEM) interface for managing tablespaces in an Oracle database. The interface displays various tablespace details, including:

- Name of the tablespace
- Allocated Size (MB)
- Space Used (MB)
- Allocated Space Used (%)
- Status
- Type
- Extent Management
- Segment Management

The table also indicates the total allocated size and free space for the database. The interface allows for searching and managing tablespaces within the database environment.
Creating a new tablespace

**Figure 15.13**

[Image of a database management interface for creating a new tablespace.]
Managing the Database Objects: Tables, Views, Triggers, and Procedures

• Database object: any object created by end users
• Schema: logical section of the database that belongs to a given user
  – Schema identified by a username
  – Within the schema, users create their own tables and other objects
• Normally, users are authorized to access only the objects that belong to their own schemas
Managing Users and Establishing Security

• User: uniquely identifiable object
  – Allows a given person to log on to the database

• Role: a named collection of database access privileges
  – Authorizes a user to connect to the database and use system resources

• Profile: named collection of settings
  – Controls how much of a resource a given user can use
Customizing the Database
Initialization Parameters

• Fine-tuning requires modification of database configuration parameters
  – Some are changed in real time using SQL
  – Some affect database instance
  – Others affect entire RDBMS and all instances

• Initialization parameters reserve resources used by the database at run time

• After modifying parameters, may need to restart the database
### The Oracle Enterprise Manager initialization parameters

**Oracle Enterprise Manager (SYS) - Initialization Parameters - Windows Internet Explorer**

The parameter values listed here are currently used by the running instance(s). You can change static parameters in SPFile mode.

<table>
<thead>
<tr>
<th>Name</th>
<th>Path/Value</th>
<th>Comments</th>
<th>Type</th>
<th>Basic</th>
<th>Modified</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>audit_file_dest</td>
<td>C:\ORACLE\SMORRIS\ADMBF</td>
<td></td>
<td>String</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>audit_trail</td>
<td>08</td>
<td></td>
<td>String</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>diagnostic_dest</td>
<td>C:\ORACLE\SMORRIS</td>
<td></td>
<td>String</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>dispatchers</td>
<td>(PROTOCOL=TCP) SERVICE=</td>
<td></td>
<td>String</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>log_archive_format</td>
<td>ARCS_64R.6T</td>
<td></td>
<td>String</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>compatible</td>
<td>11.1.0.0.0</td>
<td></td>
<td>String</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>control_files</td>
<td>C:\ORACLE\SMORRIS\ORADATA\GRADORA\CONTROL02.CTL</td>
<td></td>
<td>String</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>db_block_size</td>
<td>4096</td>
<td></td>
<td>Integer</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>db_domain</td>
<td>cisorsa1.biolab.mitsu.edu</td>
<td></td>
<td>String</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>db_name</td>
<td>gradora</td>
<td></td>
<td>String</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>db_recovery_file_dest</td>
<td>C:\oracle\smorris\itrch_recon</td>
<td></td>
<td>String</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>db_recovery_file_dest_size</td>
<td>1G</td>
<td></td>
<td>Big</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>open_cursors</td>
<td>300</td>
<td></td>
<td>Integer</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

**SOURCE:** Course Technology/Cengage Learning
Performing DBA tasks

- creating ROLES, USERS, GRANTING privileges
  - Individual-based:
    - `GRANT privileges ON someTable TO userName;`
    - PS: new users must be create before giving them privileges:
      - `CREATE USER userName IDENTIFIED BY userPassword;`
      - `GRANT CREATE SESSION TO userName;` (otherwise they can’t login to the oracle server)
  - Role-based:
    - `GRANT privileges ON someTable TO newRole;`
    - `GRANT newRole TO userName;`
    - PS: new roles must be created before assigning privileges:
      - `CREATE ROLE newRole;`
  - PS: privileges can be SELECT, DELETE, UPDATE, etc.
Performing DBA tasks

• To add privileges:
  - `GRANT privilege ON someTable TO roleName;`
  - `GRANT privilege ON someTable TO userName;`

• To remove privileges:
  - `REVOKE privilege ON someTable FROM roleName;`
  - `REVOKE privilege ON someTable FROM userName;`
Performing DBA tasks

• To create a USER
  - `CREATE USER userName IDENTIFIED BY userPassword;`

• To change a user’s password, the command is
  - `ALTER USER userName IDENTIFIED BY userPassword;`

• To remove a USER
  - `DROP USER userName;`

• To create a ROLE
  - `CREATE ROLE roleName;`

• To remove a ROLE
  - `DROP ROLE roleName;`
Summary

• Data management is a critical activity for any organization
  – Data should be treated as a corporate asset
• DBMS is the most commonly used electronic tool for corporate data management
• DBMS has impact on organization’s managerial, technological, and cultural framework
• Data administration function evolved from centralized electronic data processing
Summary (cont’d.)

• Database administrator (DBA) is responsible for managing corporate database
• Broader data management activity is handled by data administrator (DA)
• DA is more managerially oriented than more technically oriented DBA
  – DA function is DBMS-independent
  – DBA function is more DBMS-dependent
• When there is no DA, DBA executes all DA functions
Summary (cont’d.)

• Managerial services of DBA function:
  – Supporting end-user community
  – Defining and enforcing policies, procedures, and standards for database function
  – Ensuring data security, privacy, and integrity
  – Providing data backup and recovery services
  – Monitoring distribution and use of data in database
Summary (cont’d.)

• Technical role of DBA:
  – Evaluating, selecting, and installing DBMS
  – Designing and implementing databases and applications
  – Testing and evaluating databases and applications
  – Operating DBMS, utilities, and applications
  – Training and supporting users
  – Maintaining DBMS, utilities, and applications
Summary (cont’d.)

• Security: ensures confidentiality, integrity, availability of information system and data
• Security policy: collection of standards, policies, and practices
• Security vulnerability: weakness in system component
• Information engineering guides development of data administration strategy
• CASE tools and data dictionaries translate strategic plans to operational plans