1Z0-117 Q&As

Oracle Database 11g Release 2: SQL Tuning Exam

QUESTION 1

A database instance is configured in the shared server mode and it supports multiple applications running on a middle tier. These applications connect to the

database by using different services and tracing is enabled for the services. You want to view the detailed tracing setting for particular service.

What would you use to view the tracing information?

A. DBMS_SERVICE package

B. DBMS_MONITOR package

C. DBA_ENABLED_TRACES view

D. Trcsess and tkprof

Correct Answer: C

displays information about enabled SQL traces. DBA_ENABLED_TRACES

Incorrect:

A: The DBMS_SERVICE package lets you create, delete, activate, and deactivate services for a single instance.

B: The DBMS_MONITOR package let you use PL/SQL for controlling additional tracing and statistics gathering.

Reference: Oracle Database Reference, DBA_ENABLED_TRACES

QUESTION 2

You created a SQL Tuning Set (STS) containing resource-intensive SQL statements. You plan to run the SQL Tuning Advisor.

Which two types of recommendations can be provided by the SQL Tuning Advisor?

A. Semantic restructuring for each SQL statement

B. Gathering missing or stale statistics at the schema level for the entire workload

C. Creating a materialized view to benefit from query rewrite for the entire workload

D. Gathering missing or stale statistics for objects used by the statements.

E. Creating a partition table to benefit from partition pruning for each statement

Correct Answer: AD

The output of the SQL Tuning Advisor is in the form of an advice or recommendations, along with a rationale for each recommendation and its expected benefit. The recommendation relates to collection of statistics on objects (D), creation of new indexes, restructuring of the SQL statement (A), or creation of a SQL profile. You can choose to accept the recommendation to complete the tuning of the SQL statements.

Note:

*

A SQL Tuning Set can be used as input to the SQL Tuning Advisor, which performs automatic tuning of the SQL statements based on other input parameters specified by the user.

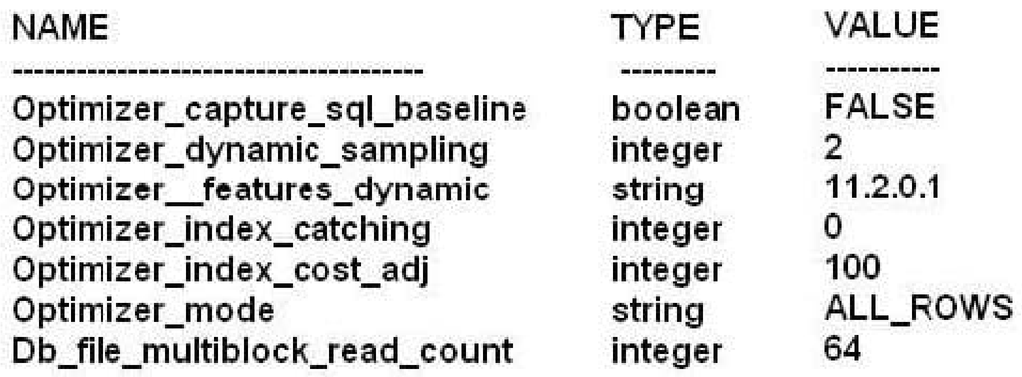
*

A SQL Tuning Set (STS) is a database object that includes one or more SQL statements along with their execution statistics and execution context, and could include a user priority ranking. The SQL statements can be loaded into a SQL Tuning Set from different SQL sources, such as the Automatic Workload Repository, the cursor cache, or custom SQL provided by the user.

Reference: Oracle Database Performance Tuning Guide 11g , SQL Tuning Advisor

QUESTION 3

Examine the utilization parameters for an instance:



```
NAME                                      TYPE        VALUE
----------------------------------------  ---------   -----------
Optimizer_capture_sql_baseline            boolean     FALSE
Optimizer_dynamic_sampling                integer     2
Optimizer__features_dynamic               string      11.2.0.1
Optimizer_index_catching                  integer     0
Optimizer_index_cost_adj                  integer     100
Optimizer_mode                            string      ALL_ROWS
Db_file_multiblock_read_count             integer     64
```

You notice that despite having an index on the column used in the where clause, queries use full table scans with highly selective filters.

What are two possible reasons for the optimizer to use full table scans instead of index unique scans and index range scans?

A. The OPTIMIZER_MODE parameter is set to ALL_ROWS.

B. The clustering factor for the indexes is high.

C. The number of leaf blocks for the indexes is high.

D. The OPTIMIZER_INDEX_COST_ADJ initialization parameter is set to 100.

E. The blocks fetched by the query are greater than the value specified by the DB_FILE_MULTIBLOCK_READ_COUNT parameter.

Correct Answer: DE

D: OPTIMIZER_INDEX_COST_ADJ lets you tune optimizer behavior for access path selection to be more or less index friendly--that is, to make the optimizer more or less prone to selecting an index access path over a full table scan.

The default for this parameter is 100 percent, at which the optimizer evaluates index access paths at the regular cost. Any other value makes the optimizer evaluate the access path at that percentage of the regular cost. For example, a setting of 50 makes the index access path look half as expensive as normal.

E: DB_FILE_MULTIBLOCK_READ_COUNT is one of the parameters you can use to minimize I/O during table scans. It specifies the maximum number of blocks read in one I/O operation during a sequential scan. The total number of I/Os needed to perform a full table scan depends on such factors as the size of the table, the multiblock read count, and whether parallel execution is being utilized for the operation. g release 2, the default value of this parameter is a value that As of Oracle Database 10 corresponds to the maximum I/O size that can be performed efficiently. This value is platform-dependent and is 1MB for most platforms.Because the parameter is expressed in blocks, it will be set to a value that is equal to the maximum I/O size that can be performed efficiently divided by the standard block size. Note that if the number of sessions is extremely large the multiblock read count value is decreased to avoid the buffer cache getting flooded with too many table scan buffers.

Even though the default value may be a large value, the optimizer will not favor large plans if you do not set this parameter. It would do so only if you explicitly set this parameter to a large value.

Online transaction processing (OLTP) and batch environments typically have values in the range of 4 to 16 for this parameter. DSS and data warehouse environments tend to benefit most from maximizing the value of this parameter. The optimizer is more likely to choose a full table scan over an index if the value of this parameter is high.

Note:

* OPTIMIZER_MODE establishes the default behavior for choosing an optimization approach for the instance.

Values:

first_rows_n

The optimizer uses a cost-based approach and optimizes with a goal of best response time to return the first n rows (where n = 1, 10, 100, 1000).

first_rows

The optimizer uses a mix of costs and heuristics to find a best plan for fast delivery of the first few rows.

all_rows

The optimizer uses a cost-based approach for all SQL statements in the session and optimizes with a goal of best throughput (minimum resource use to complete

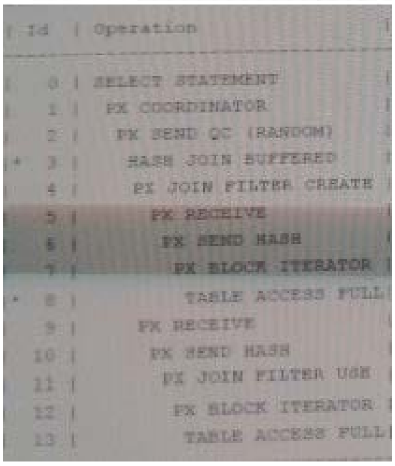
the entire statement).

QUESTION 4

Examine the Exhibit.

```
-------------------------------------------------------------------------------------------
Id    Operation                  Name        TQ       IN-OUT   PQ Distrib
-------------------------------------------------------------------------------------------
0     SELECT STATEMENT
1     PX COORDINATOR
2     PX SEND QC (RANDOM)        :TQ10002    Q1, 02   P->S     QC (RAND)
*3    HASH JOIN BUFFERED                     Q1, 02   PCWP
4     PX JOIN FILTER CREATE      :BF0000     Q1, 02   PCWP
5     PX RECEIVE                             Q1, 02   PCWP
6     PX SEND HASH               :TQ10000    Q1, 00   P->P     HASH
7     PX BLOCK ITERATOR                      Q1, 00   PCWP
*8    TABLE ACCESS FULL          T1          Q1, 00   PCWP
9     PX RECEIVE                             Q1, 02   PCWP
10    PX SEND HASH               :TQ10001    Q1, 01   P->P     HASH
11    PX JOIN FILTER USE         :BF0000     Q1, 01   PCWP
12    PX BLOCK ITERATOR                      Q1, 01   PCWC
13    TABLE ACCESS FULL          T2          Q1, 01   PCWP
-------------------------------------------------------------------------------------------
```

3 – access (“T1”. “ID” = “T2”. “ID”)
8 – filter (“T1”. “MOD” = 42)



```
| Id  | Operation                      |
---------------------------------------
|   0 | SELECT STATEMENT               |
|   1 |  PX COORDINATOR                |
|   2 |   PX SEND QC (RANDOM)          |
|*  3 |    HASH JOIN BUFFERED          |
|   4 |     PX JOIN FILTER CREATE      |
|   5 |      PX RECEIVE                |
|   6 |       PX SEND HASH             |
|   7 |        PX BLOCK ITERATOR       |
|*  8 |         TABLE ACCESS FULL      |
|   9 |     PX RECEIVE                 |
|  10 |      PX SEND HASH              |
|  11 |       PX JOIN FILTER USE       |
|  12 |        PX BLOCK ITERATOR       |
|  13 |         TABLE ACCESS FULL      |
```

Which two statements are true about the bloom filter in the execution plan?

A. The bloom filter prevents all rows from table T1 that do not join T2 from being needlessly distributed.

B. The bloom filter prevents all rows from table T2 that do not join table T1 from being needlessly distributed.

C. The bloom filter prevents some rows from table T2 that do not join table T1 from being needlessly distributed.

D. The bloom filter is created in parallel by the set of parallel execution processes that scanned table T2.

E. The bloom filter is created in parallel by the set of parallel execution processes that later perform join.

F. The bloom filter is created in parallel by the set of parallel execution processes that scanned table T1.

Correct Answer: BF

*

PX JOIN FILTER CREATE The bloom filter is created in line 4.

*

PX JOIN FILTER USE The bloom filter is used in line 11.

Note:

*

You can identify a bloom pruning in a plan when you see :BF0000 in the Pstart and Pstop columns of the execution plan and PART JOIN FILTER CREATE in the operations column.

*

A Bloom filter is a probabilistic algorithm for doing existence tests in less memory than a full list of keys would require. In other words, a Bloom filter is a method for representing a set of n elements (also called keys) to support membership queries.

*

The Oracle database makes use of Bloom filters in the following 4 situations:

-To reduce data communication between slave processes in parallel joins: mostly in RAC

-

To implement join-filter pruning: in partition pruning, the optimizer analyzes FROM and WHERE clauses in SQL statements to eliminate unneeded partitions when building the partition access list

-

To support result caches: when you run a query, Oracle will first see if the results of that query have already been computed and cached by some session or

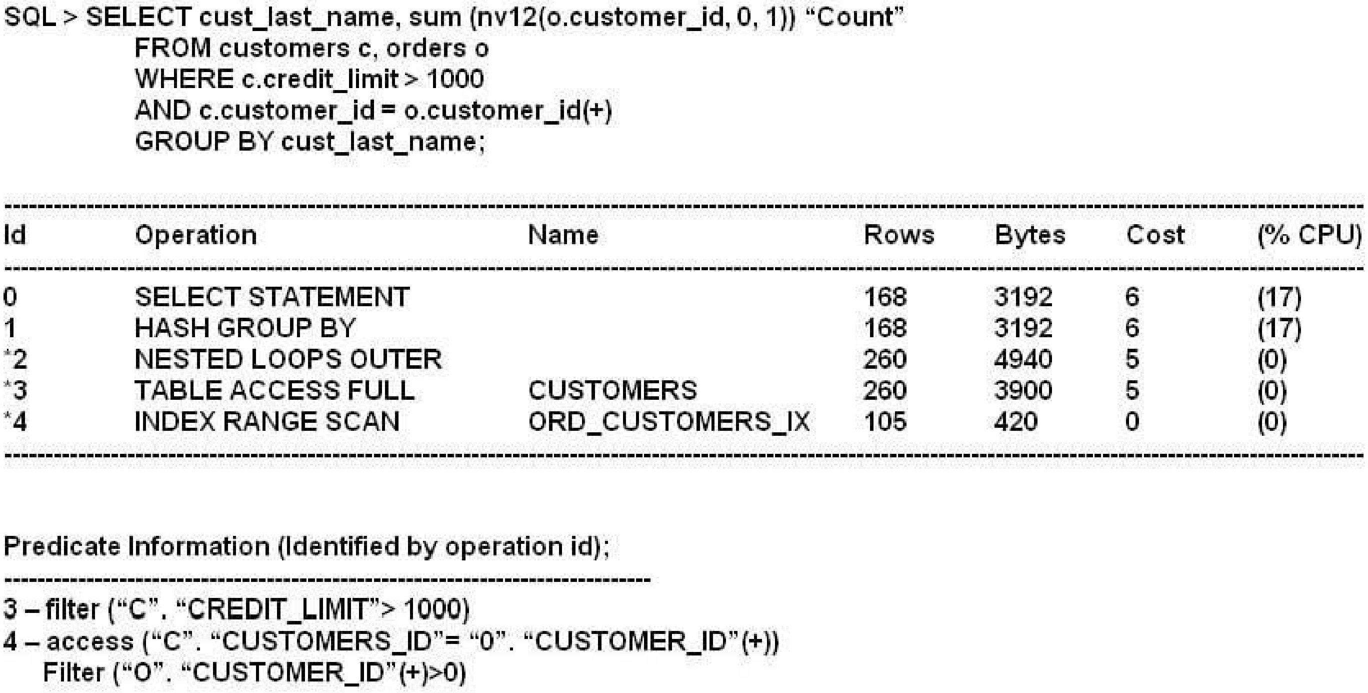
user, and if so, it will retrieve the answer from the server result cache instead of gathering all of the database blocks

-

To filter members in different cells in Exadata: Exadata performs joins between large tables and small lookup tables, a very common scenario for data warehouses with star schemas. This is implemented using Bloom filters as to determine whether a row is a member of the desired result set.

QUESTION 5

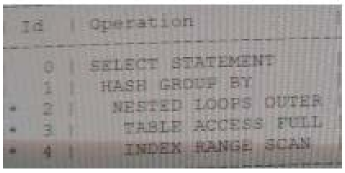
Examine the query and its execution plan: Which two statements are true regarding the execution plan?



```
SQL > SELECT cust_last_name, sum (nv12(o.customer_id, 0, 1)) "Count"
            FROM customers c, orders o
            WHERE c.credit_limit > 1000
            AND c.customer_id = o.customer_id(+)
            GROUP BY cust_last_name;
```

```
---------------------------------------------------------------------------------------------------
Id     Operation              Name                 Rows    Bytes    Cost    (% CPU)
---------------------------------------------------------------------------------------------------
0      SELECT STATEMENT                            168     3192     6       (17)
1      HASH GROUP BY                               168     3192     6       (17)
*2     NESTED LOOPS OUTER                          260     4940     5       (0)
*3     TABLE ACCESS FULL      CUSTOMERS            260     3900     5       (0)
*4     INDEX RANGE SCAN       ORD_CUSTOMERS_IX     105     420      0       (0)
---------------------------------------------------------------------------------------------------
```

```
Predicate Information (Identified by operation id);
-----------------------------------------------------
3 – filter ("C". "CREDIT_LIMIT"> 1000)
4 – access ("C". "CUSTOMERS_ID"= "0". "CUSTOMER_ID"(+))
   Filter ("O". "CUSTOMER_ID"(+)>0)
```



```
 Id  | Operation
--------------------------------
   0 | SELECT STATEMENT
   1 |  HASH GROUP BY
*  2 |   NESTED LOOPS OUTER
*  3 |    TABLE ACCESS FULL
*  4 |    INDEX RANGE SCAN
```

A. For every row of CUSTOMERS table, the row matching the join predicate from the ORDERS table are returned.

B. An outer join returns NULL for the ORDERS table columns along with the CUSTOMERS table rows when it does not find any corresponding rows in the ORDER table.

C. The data is aggregated from the ORDERS table before joining to CUSTOMERS.

D. The NESTED LOOP OUTER join is performed because the OPTIMZER_MODE parameter is set to ALL_ROWS.

Correct Answer: BD

B: An outer join extends the result of a simple join. An outer join returns all rows that satisfy the join condition and also returns some or all of those rows from one table for which no rows from the other satisfy the join condition.

Note:

*

All_rows attempts to optimize the query to get the very last row as fast as possible. This makes sense in a stored procedure for example where the client does not regain control until the stored procedure completes. You don\\'t care if you have to wait to get the first row if the last row gets back to you twice as fast. In a

client server/interactive application you may well care about that.

*

The optimizer uses nested loop joins to process an outer join in the following circumstances:

/ It is possible to drive from the outer table to inner table.

/ Data volume is low enough to make the nested loop method efficient.

*

First_rows attempts to optimize the query to get the very first row back to the client as fast as possible. This is good for an interactive client server environment

where the client runs a query and shows the user the first 10 rows or so and waits for them to page down to get more.