1Z0-117 Q&As

Oracle Database 11g Release 2: SQL Tuning Exam

Pass Oracle 1Z0-117 Exam with 100% Guarantee

QUESTION 1

Which two tasks are performed during the optimization stage of a SQL statement?

A. Evaluating the expressions and conditions in the query

B. Checking the syntax and analyzing the semantics of the statement

C. Separating the clauses of the SQL statement into structures that can be processed

D. Inspecting the integrity constraints and optimizing the query based on this metadata

E. Gathering the statistics before creating the execution plan for the statement

Correct Answer: DE

Note:

Oracle SQL is parsed before execution, and a hard parse includes these steps: *

1.

Loading into shared pool - The SQL source code is loaded into RAM for parsing. (the "hard" parse step)

2.

Syntax parse - Oracle parses the syntax to check for misspelled SQL keywords.

3.

Semantic parse - Oracle verifies all table and column names from the dictionary and checks to see if you are authorized to see the data.

4.

Query Transformation - If enabled (query_rewrite=true), Oracle will transform complex SQL into simpler, equivalent forms and replace aggregations with

materialized views, as appropriate.

5.

Optimization - Oracle then creates an execution plan, based on your schema statistics (or maybe with statistics from dynamic sampling in 10g).

6.

Create executable - Oracle builds an executable file with native file calls to service the SQL query.

*

The parsing process performs two main functions:

o Syntax Check: is the statement a valid one. Does it make sense given the SQL grammar documented in the SQL Reference Manual. Does it follow all of the rules for SQL.

o Semantic Analysis: Going beyond the syntax ? is the statement valid in light of the objects in the database (do the tables and columns referenced exist). Do you

have access to the objects ? are the proper privileges in place? Are there ambiguities in the statement ? for example if there are two tables T1 and T2 and both

have a column X, the query ?select X from T1, T2 where ?? is ambiguous, we don?t know which table to get X from. And so on.

So, you can think of parsing as basically a two step process, that of a syntax check to check the validity of the statement and that of a semantic check ? to ensure

the statement can execute properly.

Reference: Oracle hard-parse vs. soft parse

QUESTION 2

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 3

You need to upgrade you Oracle Database 10g to 11g. You want to ensure that the same SQL plans that are currently in use in the 10g database are used in the upgraded database initially, but new, better plans are allowed subsequently.

Steps to accomplish the task:

1.

Set the OPTIMIZER_USE_SQL_BASELINE and OPTIMIZER_CAPTURE_SQL_PLAN_BASELINE to TRUE.

2.

Bulk load the SQL Management Base as part of an upgrade using an STS containing the plans captured in Oracle Database 10g.

3.

Evolve the plan baseline using the DBMS_SPM.EVOLVE_PLAN_BASELINE procedure.

4.

Fix the plan baseline using the DBMS_SPM.ALTER_SQL_PLANBASELINE procedure.

5.

Accept new, better plans using the DBMS_SPM.ALTER_SQL_PLAN_BASELINE procedure and manually load them to the existing baseline.

6.

Set OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES to FALSE.

Identify the required steps.

A. 1, 3, 4, 5

B. 1, 6, 3, 4, 5

C. 1, 2, 3, 5

D. 1, 2, 3, 4

E. 1, 6, 3

F. 1 and 2

Correct Answer: F

*

(1) OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES In Oracle Database 11g a new feature called SQL Plan Management (SPM) has been introduced to guarantees any plan changes that do occur lead to better performance. When OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES is set to TRUE (default FALSE) Oracle will automatically capture a SQL plan baseline for every repeatable SQL statement on the system. The execution plan found at parse time will be added to the SQL plan baseline as an accepted plan.

*

(2) Once you have completed the software upgrade, but before you restart the applications and allow users back on the system, you should populate SQL Plan Management (SPM) with the 10g execution plans you captured before the upgrade. Seeding SPM with the 10g execution plans ensures that the application will continue to use the same execution plans you had before the upgrade. Any new execution plans found in Oracle Database 11g will be recorded in the plan history for that statement but they will not be used. When you are ready you can evolve or verify the new plans and only implement those that perform better than the 10g plan.

Incorrect:

Not (3): DBMS_SPM.EVOLVE_PLAN_BASELINE is not used to evolve new plans.

DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE should be used:

It is possible to evolve a SQL statement\\'s execution plan using Oracle Enterprise Manager or by running the command-line function

DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE. U

Note:

* SQL plan management (SPM) ensures that runtime performance will never degrade due to the change of an execution plan. To guarantee this, only accepted

(trusted) execution plans will be used; any plan will be tracked and evaluated at a later point in time and only accepted as verified if the new plan performs better

than an accepted plan. SQL Plan Management has three main components:

1.

SQL plan baseline capture:

Create SQL plan baselines that represents accepted execution plans for all relevant SQL statements. The SQL plan baselines are stored in a plan history inside

the SQL Management Base in the SYSAUX tablespace.

2.

SQL plan baseline selection

Ensure that only accepted execution plans are used for statements with a SQL plan baseline and track all new execution plans in the history for a statement as

unaccepted plan. The plan history consists of accepted and unaccepted plans. An unaccepted plan can be unverified (newly found but not verified) or rejected

(verified but not found to performant).

3.

SQL plan baseline evolution

Evaluate all unverified execution plans for a given statement in the plan history to become either accepted or rejected

QUESTION 4

Examine the exhibit.

Id	Operation	Name	Pstsrt	Pstop	IN-OUT	PQ	Distrib
0	SELECT STATEMENT						
1	PX COORDINATOR						
2	PX SEND QC (RANDOM)	:TQ10001			P->S	QC	(RAND)
*3	FILTER				PCWC		
4	HASH GROUP BY				PCWP		
5	PX RECEIVE				PCWP		
6	PX SEND HASH	:TQ10000			P->P	HASH	
7	HASH GRIYP BY		1	16	PCWC		
8	PX PARTITION HASH ALL		1	16	PCWP		
*9	HASH JOIN				PCWP		
10	TABLE ACCESS FULL	CUSTOMERS	1	16	PCWP		
11	PX PARTITION RANGE ITERATOR		8	9	PCWC		
*12	TABLE ACCESS FULL	SALES	113	144	PCWP		

Predicate information (identified by operation id):

3 – filter (COUNT (SYS_OP_CSR(SYS_OP_MSR(COUNT(*)), 0))>100)
9 – access ("S". "CUST_ID"= "C". "CUST_ID")
12 – filter ("S". "TIME_ID"<= TO_DATE ('1999-10-01 00:00:00', 'syyyy-mm-dd hh2:mi:ss') AND
"S", "TIME_ID"> =TO_DATE('1999-07-01
00:00:00', 'syyyy-mm-dd hh24:mi:ss')

Which two are true concerning the execution plan?

A. No partition-wise join is used

B. A full partition-wise join is used

C. A partial partition-wise join is used

D. The SALES table is composite partitioned

Correct Answer: BD

* The following example shows the execution plan for the full partition-wise join with the sales table range partitioned by time_id, and subpartitioned by hash on

cust_id. -- | Id | Operation | Name | Pstart| Pstop |IN-OUT| PQ Distrib |

-- | 0 | SELECT STATEMENT | | | | | |

| 1 | PX COORDINATOR | | | | | |

| 2 | PX SEND QC (RANDOM) | :TQ10001 | | | P->S | QC (RAND) |

|* 3 | FILTER | | | | PCWC | |

| 4 | HASH GROUP BY | | | | PCWP | |

| 5 | PX RECEIVE | | | | PCWP | |

| 6 | PX SEND HASH | :TQ10000 | | | P->P | HASH |

| 7 | HASH GROUP BY | | | | PCWP | |

| 8 | PX PARTITION HASH ALL | | 1 | 16 | PCWC | |

|* 9 | HASH JOIN | | | | PCWP | |

| 10 | TABLE ACCESS FULL | CUSTOMERS | 1 | 16 | PCWP | |

| 11 | PX PARTITION RANGE ITERATOR| | 8 | 9 | PCWC | |

|* 12 | TABLE ACCESS FULL | SALES | 113 | 144 | PCWP | |

Predicate Information (identified by operation id):

3 - filter(COUNT(SYS_OP_CSR(SYS_OP_MSR(COUNT(*)),0))>100)

9 - access("S"."CUST_ID"="C"."CUST_ID")

12 - filter("S"."TIME_ID"=TO_DATE(\\' 1999-07-01

00:00:00\\', \\'syyyy-mm-dd hh24:mi:ss\\'))

* Full partition-wise joins can occur if two tables that are co-partitioned on the same key are joined in a query. The tables can be co-partitioned at the partition level, or at the subpartition level, or at a combination of partition and subpartition levels. Reference partitioning is an easy way to guarantee co-partitioning. Full partition-wise joins can be executed in serial and in parallel.

Reference: Oracle Database VLDB and Partitioning Guide, Full Partition-Wise Joins: Composite

-Single-Level

QUESTION 5

You executed the following statements:

```
SQL> SET AUTOTRACE TRACEONLY EXPLAIN STAT

SQL> SELECT e.name, e.sal, d.name
         DEOM emp, dept d
         WHERE e.dept_id = d.dept_id;
```

Which two statements are true about the query execution?

A. The execution plan is generated and fetched from the library cache.

B. The query executes and displays the execution plan and statistics.

C. The query executes and inserts the execution plan in PLAN_TABLE.

D. The query executes and execution plan is stored in the library cache and can be viewed using v$SQL_PLAN.

E. The query will always use the plan displayed by the AUTOTRACE output.

Correct Answer: BD

B: set autotrace traceonly: Displays the execution plan and the statistics (as set autotrace on does), but doesn\\'t print a query\\'s result.

Note:

/ Autotrace

Autotrace can be configured to run the SQL and gives a plan and statistics afterwards or just give you an explain plan without executing the query. To achieve this

use the following:

*

Explain only

set autotrace traceonly explain

*

Execute with stats and explain plan

set autotrace on explain stat (with data returned by query)

or

autotrace traceo expl stat (without data returned by query)

*

To make the output from an autotrace more readable

col plan_plus_exp format a100

*

Turn off autotrace

set autotrace off

/ V$SQL_PLAN contains the execution plan information for each child cursor loaded in the library cache. http://docs.oracle.com/cd/E11882_01/server.112/e40402/dynviews_3054.htm#REFRN30250