1Z0-117 Q&As

Oracle Database 11g Release 2: SQL Tuning Exam

Pass Oracle 1Z0-117 Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

https://www.pass4itsure.com/1z0-117.html

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by Oracle Official Exam Center

QUESTION 1

You are administering a database supporting an OLTP application. The application runs a series of extremely similar queries the MYSALES table where the value of CUST_ID changes.

Examine Exhibit1 to view the query and its execution plan.

```
SELECT prod_id, SUM (quantity_sold)
FROM myscales
WHERE cust_id=25939
GROUP by prod_id

Execution Plan
----------------------------------------------------------
Plan hash value: 1831518131
```

Id	Operation	Name	Rows	Bytes	Cost	(CPU%)	Time
0	SELECT STATEMENT		72	864	527	(1)	00:00:07
1	HASH GROUP BY		72	864	527	(1)	00:00:07
2	TABLE ACCESS BY INDEX ROWID	MYSALES	651	7812	526	(1)	00:00:07
*3	INDEX RANGE SCAN	CUST_ID_IDX	651		4	(0)	00:00:01

```
PREDICATE Information (identified by operation id):
---------------------------------------------------
3- access ("CUST_ID") = 25939

Statistics
----------------------------------------------------
0          recursive calls
0          db block size
1050       consistent gets
0          Physical reads
0          redo size
1348       bytes sent via SQL*NET to client
541        bytes received via SQL*NET to client
4          Bytes received via SQL*NET from client
0          sorts (memory)
0          sorts (disk)
36         rows processed
```

Examine Exhibit 2 to view the structure and indexes for the MYSALES table. The MYSALES table has 4 million records.

```
SQL> DESC mysales

Name                                      Null?            Type
----------------------------------------------------------------
PROD_ID                                   NOT NULL         NUMBER
CUST_ID                                   NOT NULL         NUMBER
TIME_ID                                   NOT NULL         DATE
CHANNEL_ID                                NOT NULL         NUMBER
PROMO_ID                                  NOT NULL         NUMBER
QUANTITY_SOLD                             NOT NULL         NUMBER (10, 2)
AMOUNT_SOLD                               NOT NULL         NUMBER (10, 2)

SQL> SELECT a.index_name, index_type, column_name
          FROM user_ind_columns a, user_indexes b
          WHERE a.index_name=b.index_name AND a.table_name = 'MYSALES'

INDEX_NAME                    INDEX_TYPE          COLUMN_NAME
-----------------------------------------------------------------------------------
PROD_ID_IDX                   NORMAL              PROD_ID
CHANNEL_ID_IDX                NORMAL              CHANNEL_ID
CUST_ID_idx                   NORMAL              CUST_ID
```

Data in the CUST_ID column is highly skewed. Examine the parameters set for the instance:

```
NAME                          TYPE                VALUE
-----------------------------------------------------------------------------------
Cursor_sharing                string              EXACT
Cursor_space_for_time         boolean             FALSE
Open_cursors                  integer             300
Session_cached_cursors        integer             50
Optimizer_dynamic_sampling    integer             2
```

Which action would you like to make the query use the best plan for the selectivity?

A. Decrease the value of the OPTIMIZER_DYNAMIC_SAMPLING parameter to 0.

B. Us the /*+ INDEX(CUST_ID_IDX) */ hint in the query.

C. Drop the existing B* -tree index and re-create it as a bitmapped index on the CUST_ID column.

D. Collect histogram statistics for the CUST_ID column and use a bind variable instead of literal values.

Correct Answer: D

Using Histograms In some cases, the distribution of values within a column of a table will affect the optimizer\\'s decision to use an index vs. perform a full-table scan. This scenario occurs when the value with a where clause has a disproportional amount of values, making a full-table scan cheaper than index access.

A column histogram should only be created when we have data skew exists or is suspected.

QUESTION 2

Examine the following command:

```
SQL > CREATE VIEW all_employees AS
        (SELECT employee_id, last_name, job_id, commission_pct, department_id FROM employees)
UNION
        (SELECT employee_id, last_name, job_id, commission_pct, department_id FROM contract_workers);
```

You query the view as follows:

```
SELECT last_name
FROM all_employee
WHERE department_id = 50;
```

Which query transformation technique is used by the optimizer in this case?

A. View merging

B. Filter push-down

C. Predicate pushing

D. Predicate move-around

Correct Answer: C

In predicate pushing, the optimizer "pushes" the relevant predicates from the containing query block into the view query block. For views that are not merged, this technique improves the subplan of the unmerged view because the database can use the pushed-in predicates to access indexes or to use as filters.

For example, suppose you create a view that references two employee tables. The view is defined with a compound query that uses the UNION set operator, as follows:

CREATE VIEW all_employees_vw AS

(SELECT employee_id, last_name, job_id, commission_pct, department_id FROM employees)

UNION

(SELECT employee_id, last_name, job_id, commission_pct, department_id FROM contract_workers);

You then query the view as follows:

SELECT last_name

FROM all_employees_vw

WHERE department_id = 50;

Because the view is a compound query, the optimizer cannot merge the view\\'s query into the accessing query block.

Instead, the optimizer can transform the

accessing statement by pushing its predicate, the WHERE clause condition department_id=50, into the view\\'s compound query.

The equivalent transformed query is as follows:

SELECT last_name

FROM (SELECT employee_id, last_name, job_id, commission_pct, department_id FROM employees

WHERE department_id=50

UNION

SELECT employee_id, last_name, job_id, commission_pct, department_id FROM contract_workers

WHERE department_id=50);

Reference: Oracle Database Performance Tuning Guide, Predicate Pushing

QUESTION 3

When would bind peeking be done for queries that vary only in values used in the WHERE clause?

A. When the column used in the WHERE clause has evenly distributed data and histogram exists on that column.

B. When the column used in the WHERE clause has evenly distributed data and index exists on that column.

C. When the column used in the WHERE clause has non uniform distribution of data, uses a bind variable, and no histogram exists for the column.

D. When the column used in the WHERE clause has non uniform distribution of data and histogram exists for the column.

Correct Answer: B

QUESTION 4

You are working on a database that supports an OLTP workload. You see a large number of hard parses occurring and several almost identical SQL statements in the library cache that vary only in the literal values in the WHERE clause conditions.

Which two methods can you use to reduce hard parsing?

A. Replace literals with bind variables and evolve a baseline for the statement.

B. Use the RESULT_CACHE hint in the queries.

C. Create baselines for the almost identical SQL statement by manually loading them from the cursor cache.

D. Set the CURSOR_SHARING parameter to SIMILAR.

Correct Answer: AD

A: We can reduce this Hard parsing by using bindvariables

D: SIMILAR

Causes statements that may differ in some literals, but are otherwise identical, to share a cursor, unless the literals affect either the meaning of the statement or the degree to which the plan is optimized.

Note:

A hard parse is when your SQL must be re-loaded into the shared pool. A hard parse is worse than a soft parse because of the overhead involved in shared pool RAM allocation and memory management. Once loaded, the SQL must then be completely re-checked for syntax and semantics and an executable generated.

Excessive hard parsing can occur when your shared_pool_size is too small (and reentrant SQL is paged out), or when you have non-reusable SQL statements without host variables.

See the cursor_sharing parameter for a easy way to make SQL reentrant and remember that you should always use host variables in you SQL so that they can be reentrant.

Reference: Oracle Database Reference, CURSOR_SHARING

QUESTION 5

You plan to bulk load data INSERT INTO . . . SELECT FROM statements.

Which two situations benefit from parallel INSERT operations on tables that have no materialized views defined on them?

A. Direct path insert of a million rows into a partitioned, index-organized table containing one million rows and a conventional B*tree secondary index.

B. Direct path insert of a million rows into a partitioned, index-organized table containing 10 rows and a bitmapped secondary index.

C. Direct path insert of 10 rows into a partitioned, index-organized table containing one million rows and conventional B* tree secondary index.

D. Direct path insert of 10 rows into a partitioned, index-organized table containing 10 rows and a bitmapped secondary index

E. Conventional path insert of a million rows into a nonpartitioned, heap-organized containing 10 rows and having a conventional B* tree index.

F. Conventional path insert of 10 rows into a nonpartitioned, heap-organized table one million rows and a bitmapped index.

Correct Answer: AB

Note:

*

A materialized view is a database object that contains the results of a query.

*

You can use the INSERT statement to insert data into a table, partition, or view in two ways: conventional INSERTand direct-path INSERT.

*

With direct-path INSERT, the database appends the inserted data after existing data in the table. Data is written directly into datafiles, bypassing the buffer cache. Free space in the existing data is not reused. This alternative enhances performance during insert operations and is similar to the functionality of the Oracle direct-path loader utility, SQL*Loader. When you insert into a table that has been created in parallel mode, direct-path INSERT is the default.

*

Direct-path INSERT is not supported for an index-organized table (IOT) if it is not partitioned, if it has a mapping table, or if it is reference by a materialized view.

*

When you issue a conventional INSERT statement, Oracle Database reuses free space in the table into which you are inserting and maintains referential integrity constraints

*

Conventional INSERT always generates maximal redo and undo for changes to both data and metadata, regardless of the logging setting of the table and the archivelog and force logging settings of the database