Vendor:home

Exam Code:1Z0-117

Exam Name: Oracle Database 11g Release 2: SQL Tuning Exam

Version: Demo

QUESTION 1

Which three options are true about parallel queries when PARALLEL_DEGREE_POLICY is set to MANUAL and the session is using the default settings for parallel query, DDL, and DML?

A. A subquery in a parallel DML is parallelized only if it includes a parallel hint.

B. The number of parallel execution servers requested for a cursor is based on the greatest degree of parallelism associated with any object accessed by the cursor.

C. A SELECT statement can be executed in parallel only if no scalar subqueries are contained in the SELECT list.

D. In a CREATE TABLE . . . AS SELECT (CTAS) statement, SELECT is parallelized only if create TABLE is parallelized.

E. In an INSERT INTO . . . SELECT FROM statement, INSERT is parallelized if select is parallelized.

F. Single row inserts are never executed is parallel.

Correct Answer: CEF

* Decision to Parallelize

A SELECT statement can be parallelized only if the following conditions are satisfied:

/ The query includes a parallel hint specification (PARALLEL or PARALLEL_INDEX) or the schema objects referred to in the query have a PARALLEL declaration associated with them. / At least one of the tables specified in the query requires one of the following:

A full table scan

An index range scan spanning multiple partitions

/ (C) No scalar subqueries are in the SELECT list.

*

By default, the system only uses parallel execution when a parallel degree has been explicitly set on an object or if a parallel hint is specified in the SQL statement.

*

CREATE TABLE ... AS SELECT in Parallel

Parallel execution lets you parallelize the query and create operations of creating a table as a subquery from another table or set of tables. This can be extremely useful in the creation of summary or rollup tables.

Clustered tables cannot be created and populated in parallel.

* PARALLEL_DEGREE_POLICY specifies whether or not automatic degree of Parallelism, statement queuing, and in-memory parallel execution will be enabled.

MANUAL

Disables automatic degree of parallelism, statement queuing, and in-memory parallel execution. This reverts the behavior of parallel execution to what it was prior to Oracle Database 11g Release 2 (11.2). This is the default.

Incorrect:

A:

*

For parallel DML (INSERT, UPDATE, MERGE, and DELETE), the reference object that determines the DOP (degree of parallelism) is the table being modified by and insert, update, or delete operation. Parallel DML also adds some limits to the DOP to prevent deadlock. If the parallel DML statement includes a subquery, the subquery\\'s DOP is the same as the DML operation.

*

For parallel DDL, the reference object that determines the DOP is the table, index, or partition being created, rebuilt, split, or moved. If the parallel DDL statement includes a subquery, the subquery\\'s DOP is the same as the DDL operation.

D: The CREATE TABLE ... AS SELECT statement contains two parts: a CREATE part (DDL) and a SELECT part (query). Oracle Database can parallelize both parts of the statement. The query part of a CREATE TABLE ... AS SELECT statement can be parallelized only if the following conditions are satisfied:

The query includes a parallel hint specification (PARALLEL or PARALLEL_INDEX) or the CREATE part of the statement has a PARALLEL clause specification or the schema objects referred to in the query have a PARALLEL declaration associated with them.

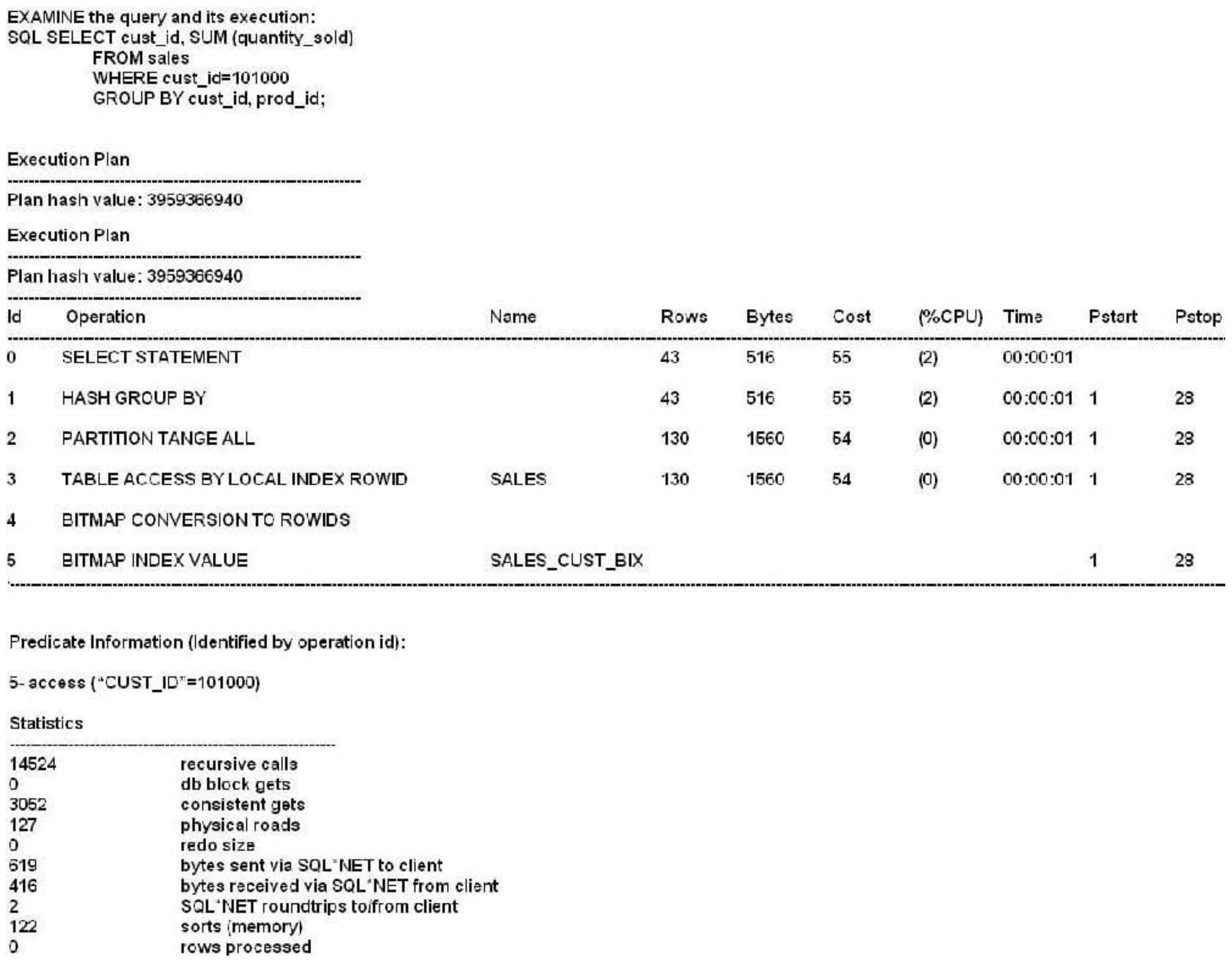
At least one of the tables specified in the query requires one of the following: a full table scan or an index range scan spanning multiple partitions.

Reference: Oracle Database VLDB and Partitioning Guide, Using Parallel Execution

---

QUESTION 2

Examine the Exhibit to view the structure of an indexes for the SALES table.



```
EXAMINE the query and its execution:
SQL SELECT cust_id, SUM (quantity_sold)
              FROM sales
              WHERE cust_id=101000
              GROUP BY cust_id, prod_id;

Execution Plan
------------------------------------------------------------
Plan hash value: 3959366940

Execution Plan
------------------------------------------------------------
Plan hash value: 3959366940
------------------------------------------------------------
Id   Operation                           Name             Rows   Bytes   Cost   (%CPU)   Time       Pstart   Pstop
-------------------------------------------------------------------------------------------------------------------
0    SELECT STATEMENT                                     43     516     55     (2)      00:00:01

1    HASH GROUP BY                                        43     516     55     (2)      00:00:01   1        28

2    PARTITION TANGE ALL                                  130    1560    54     (0)      00:00:01   1        28

3    TABLE ACCESS BY LOCAL INDEX ROWID   SALES            130    1560    54     (0)      00:00:01   1        28

4    BITMAP CONVERSION TO ROWIDS

5    BITMAP INDEX VALUE                  SALES_CUST_BIX                                             1        28
-------------------------------------------------------------------------------------------------------------------

Predicate Information (Identified by operation id):

5- access ("CUST_ID"=101000)

Statistics
------------------------------------------------------------
14524              recursive calls
0                  db block gets
3052               consistent gets
127                physical roads
0                  redo size
619                bytes sent via SQL*NET to client
416                bytes received via SQL*NET from client
2                  SQL*NET roundtrips to/from client
122                sorts (memory)
0                  rows processed
```

The SALES table has 4594215 rows. The CUST_ID column has 2079 distinct values. What would you do to influence the optimizer for better selectivity?

A. Drop bitmap index and create balanced B*Tree index on the CUST_ID column.

B. Create a height-balanced histogram for the CUST_ID column.

C. Gather statistics for the indexes on the SALES table.

D. Use the ALL_ROWS hint in the query.

Correct Answer: D

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).

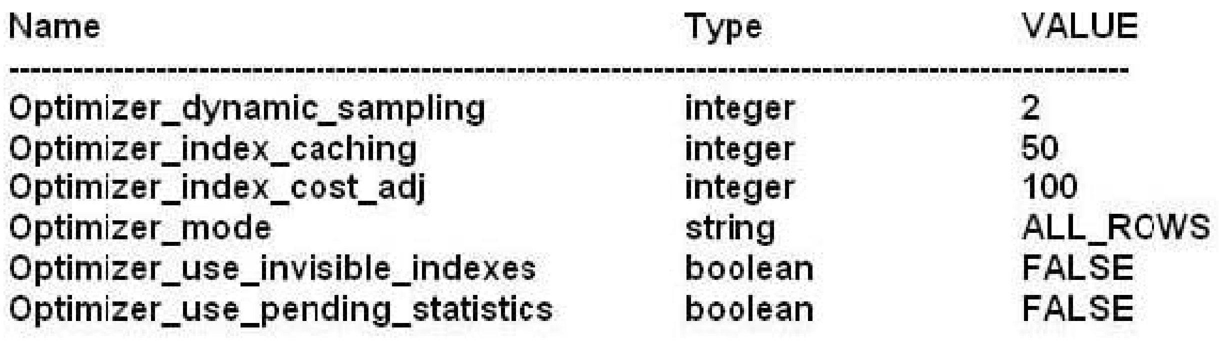
FIRST_ROWS - The optimizer uses a mix of costs and heuristics to find a best plan for fast delivery of the first few rows.

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

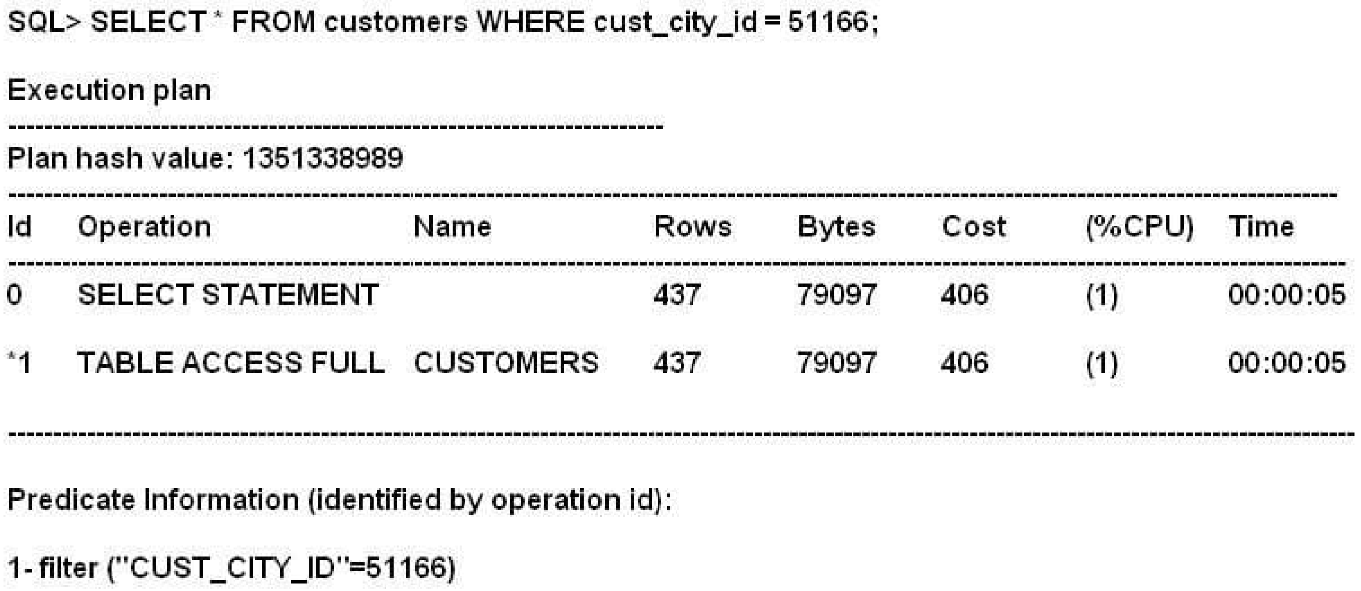
Examine the initializing parameters:



```
Name                                    Type            VALUE
-----------------------------------------------------------------------
Optimizer_dynamic_sampling              integer         2
Optimizer_index_caching                 integer         50
Optimizer_index_cost_adj                integer         100
Optimizer_mode                          string          ALL_ROWS
Optimizer_use_invisible_indexes         boolean         FALSE
Optimizer_use_pending_statistics        boolean         FALSE
```

An index exists on the column used in the WHERE of a query. You execute the query for the first time today and notice that the query is not using the index. The CUSTOMERS table has 55000 rows.

View the exhibit and examine the query and its execution plan.



```
SQL> SELECT * FROM customers WHERE cust_city_id = 51166;

Execution plan
------------------------------------------------
Plan hash value: 1351338989
---------------------------------------------------------------------------------------------
Id   Operation           Name        Rows    Bytes    Cost    (%CPU)   Time
---------------------------------------------------------------------------------------------
0    SELECT STATEMENT                437     79097    406     (1)      00:00:05

*1   TABLE ACCESS FULL   CUSTOMERS   437     79097    406     (1)      00:00:05

---------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):

1- filter ("CUST_CITY_ID"=51166)
```

What can be the two reasons for full table scan?

A. The value of the OPTIMIZER_INDEX_COST_ADJ parameter is set to a low value.

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

C. The statistics for the CUSTOMERS table and the indexes stale.

D. The OPTIMIZER_MODE parameter is set to ALL_ROWS.

E. Histogram statistics for CUST_CITY_ID are missing.

F. Average number of rows per block for the CUSTOMERS table is low.

Correct Answer: CE

C: Old statistics could cause this problem. "Histograms are feature in CBO and it helps to optimizer to determine how data are skewed(distributed) with in the column. Histogram is good to create for the column which

are included in the WHERE clause where the column is highly skewed. Histogram helps to

optimizer to decide whether to use an index or full-table scan or help the optimizer

"

determine the fastest table join order.

QUESTION 4

Which statement is true about an SQL plan baselines that are fixed?

A. New plans are added automatically by the optimizer to the baseline and are automatically evolved.

B. New, better plans are added automatically as a fixed plan baseline.

C. New plan can be manually loaded to the baseline from the cursor cache or a SQL tuning set.

D. New plans can be added as fixed plans to the baseline by using the SQL Tuning Advisor to generate a SQL profile and by accepting the SQL profile.

Correct Answer: D

When a SQL statement with a fixed SQL plan baseline is tuned using the SQL Tuning Advisor, a SQL profile recommendation has special meaning. When the

SQL profile is accepted, the tuned plan is added to the fixed SQL plan baseline as a non-fixed plan. However, as described above, the optimizer will not use the

tuned plan as long as a reproducible fixed plan is present. Therefore, the benefit of SQL tuning may not be realized. To enable the use of the tuned plan, manually

alter the tuned plan to a fixed plan by setting its FIXED attribute to YES.

Note:

It is also possible to influence the optimizer\\'s choice of plan when it is selecting a plan from a SQL plan baseline. SQL plan baselines can be marked as fixed.

Fixed SQL plan baselines indicate to the optimizer that they are preferred. If the optimizer is costing SQL plan baselines and one of the plans is fixed, the optimizer will only cost the fixed plan and go with that if it is reproducible.

If the fixed plan(s) are not reproducible the optimizer will go back and cost the remaining SQL plan baselines and select the one with the lowest cost. Note that

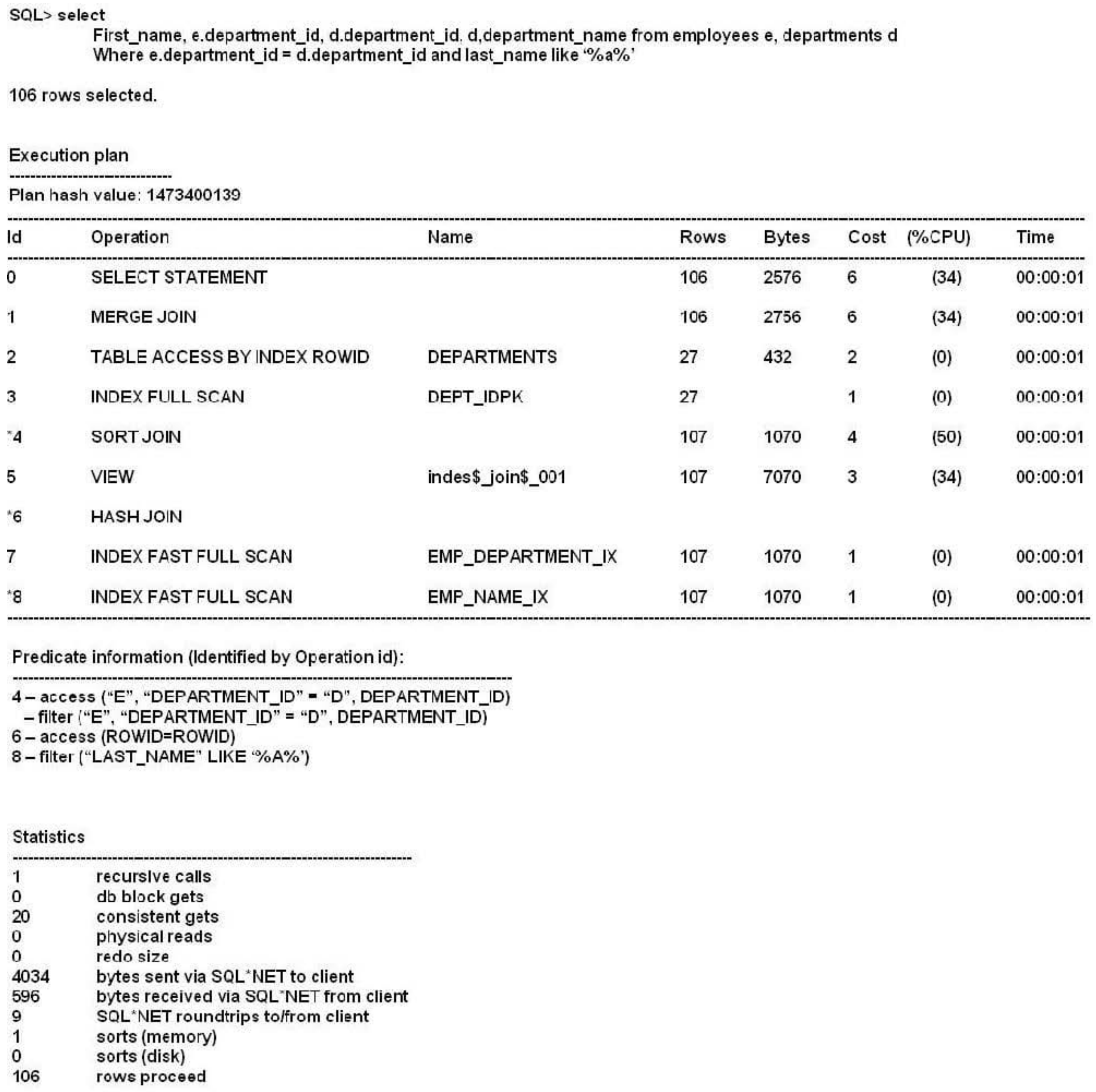
costing a plan is nowhere near as expensive as a hard parse. The optimizer is not looking at all possible access methods but at one specific access path.

Reference: Oracle Database Performance Tuning Guide 11g, Using Fixed SQL Plan Baselines Reference: SQL Plan Management in Oracle Database 11g

---

QUESTION 5

Examine Exhibit 1 to view the query and its execution plan.



```
SQL> select
        First_name, e.department_id, d.department_id. d,department_name from employees e, departments d
        Where e.department_id = d.department_id and last_name like '%a%'

106 rows selected.

Execution plan
------------------------------
Plan hash value: 1473400139
-----------------------------------------------------------------------------------------------------------------------------
Id    Operation                        Name                  Rows    Bytes    Cost   (%CPU)    Time
-----------------------------------------------------------------------------------------------------------------------------
0     SELECT STATEMENT                                       106     2576     6        (34)    00:00:01

1     MERGE JOIN                                             106     2756     6        (34)    00:00:01

2     TABLE ACCESS BY INDEX ROWID      DEPARTMENTS           27      432      2        (0)     00:00:01

3     INDEX FULL SCAN                  DEPT_IDPK             27               1        (0)     00:00:01

*4    SORT JOIN                                              107     1070     4        (50)    00:00:01

5     VIEW                             indes$_join$_001      107     7070     3        (34)    00:00:01

*6    HASH JOIN

7     INDEX FAST FULL SCAN             EMP_DEPARTMENT_IX     107     1070     1        (0)     00:00:01

*8    INDEX FAST FULL SCAN             EMP_NAME_IX           107     1070     1        (0)     00:00:01
-----------------------------------------------------------------------------------------------------------------------------

Predicate information (Identified by Operation id):
-----------------------------------------------------------------------------------------------
4 – access ("E", "DEPARTMENT_ID" = "D", DEPARTMENT_ID)
  – filter ("E", "DEPARTMENT_ID" = "D", DEPARTMENT_ID)
6 – access (ROWID=ROWID)
8 – filter ("LAST_NAME" LIKE '%A%')


Statistics
-----------------------------------------------------------------------------
1         recursive calls
0         db block gets
20        consistent gets
0         physical reads
0         redo size
4034      bytes sent via SQL*NET to client
596       bytes received via SQL*NET from client
9         SQL*NET roundtrips to/from client
1         sorts (memory)
0         sorts (disk)
106       rows proceed
```

Examine Exhibit 2 to view the structure and indexes for the EMPLOYEES and DEPARTMENTS tables. Examine Exhibit 3 to view the initialization parameters for the instance.

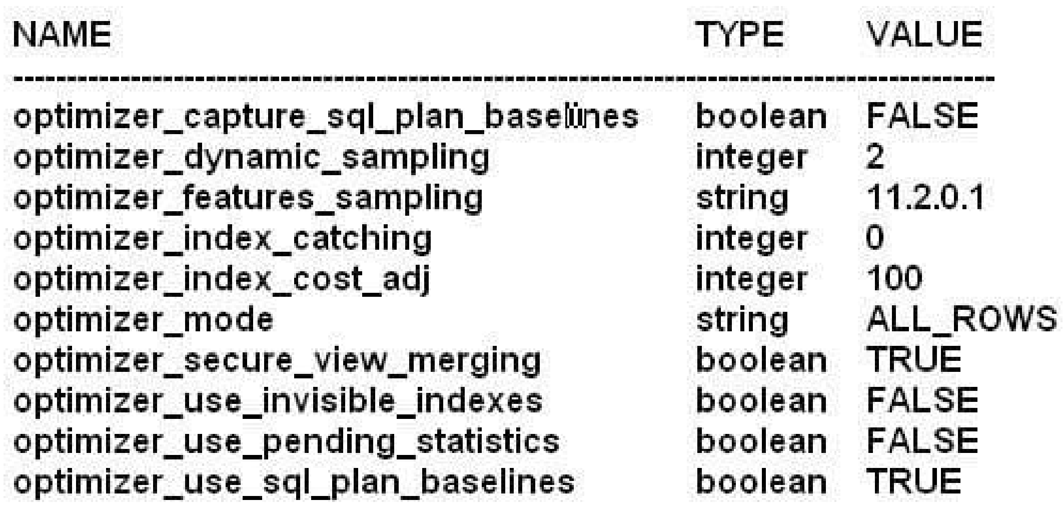
```
Name                          Null?           Type
----------------------------------------------------------------
EMPLOYEE_ID                   NOT NULL        NUMBER(6)
FIRST_NAME                                    VARCHAR(20)
LAST_NAME                     NOT NULL        VARCHAR(25)
EMAIL                         NOT NULL        VARCHAR (25)
PHONE_NUMBER                                  VARCHAR(20)
HIRE_DATE                     NOT NULL        DATE
JOB_ID                        NOT NULL        VARCHAR (10)
SALARY                                        NUMBER (8, 2)
COMISSION_PCT                                 NUMBER (2, 2)
MANAGER_ID                                    NUMBER (6)
DEPARTMENT_ID                                 NUMBER (4)
```

```
INDEX_NAME                    INDEX_TYPE      COLUMN_NAME
-----------------------------------------------------------------
EMP_NAME_IX                   NORMAL          LAST_NAME
EMP_MANAGER_IX                NORMAL          MANAGER_ID
EMP_JOB_IX                    NORMAL          JOB_ID
EMP_DEPARTMENT_IX             NORMAL          DEPARTMENT_ID
EMP_EMP_ID_PK                 NORMAL          EMPLOYEE_ID
EMP_EMAIL_UK                  NORMAL          EMAIL
```

Departments

```
Name                          Null?           Type
--------------------------------------------------
DEPARTMENT                    NOT NULL        NUMBER (4)
DEPARTMENT_NAME               NOT NULL        VARCHAR (30)
MANAGER_ID                                    NUMBER (6)
LOCATION_ID                                   NUMBER (4)
```

```
INDEX_NAME                    INDEX_TYPE      COLUMN_NAME
------------------------------------------------------------
DEPT_LOCATION_IX              NORMAL          LOCATION_ID
DEPT_ID_PK                    NORMAL          DEPARTMENT_ID
```



```
NAME                                   TYPE      VALUE
------------------------------------------------------------------------------------------
optimizer_capture_sql_plan_baselines   boolean   FALSE
optimizer_dynamic_sampling             integer   2
optimizer_features_sampling            string    11.2.0.1
optimizer_index_catching               integer   0
optimizer_index_cost_adj               integer   100
optimizer_mode                         string    ALL_ROWS
optimizer_secure_view_merging          boolean   TRUE
optimizer_use_invisible_indexes        boolean   FALSE
optimizer_use_pending_statistics       boolean   FALSE
optimizer_use_sql_plan_baselines       boolean   TRUE
```

Why is sort-merge join chosen as the access method?

A. Because the OPTIMIZER_MODE parameter is set to ALL_ROWS.

B. Because of an inequality condition.

C. Because the data is not sorted in the LAST_NAME column of the EMPLOYEES table

D. Because of the LIKE operator used in the query to filter out records

Correct Answer: A

Incorrect:

B: There is not an inequality condition in the statement.

C: Merge joins are beneficial if the columns are sorted.

D: All regular joins should be able to use Hash or Sort Merge, except LIKE, !=, and NOT ... joins.

Note:

*

A sort merge join is a join optimization method where two tables are sorted and then joined.

*

A "sort merge" join is performed by sorting the two data sets to be joined according to the join keys and then merging them together. The merge is very cheap, but the sort can be prohibitively expensive especially if the sort spills to disk. The cost of the sort can be lowered if one of the data sets can be accessed in sorted order via an index, although accessing a high proportion of blocks of a table via an index scan can also be very expensive in comparison to a full table scan.

*

Sort merge joins are useful when the join condition between two tables is an inequality condition (but not a nonequality) like =. Sort merge joins

perform better than nested loop joins for large data sets. You cannot use hash joins unless there is an equality condition.

*

When the Optimizer Uses Sort Merge Joins

The optimizer can choose a sort merge join over a hash join for joining large amounts of data if any of the following conditions are true:

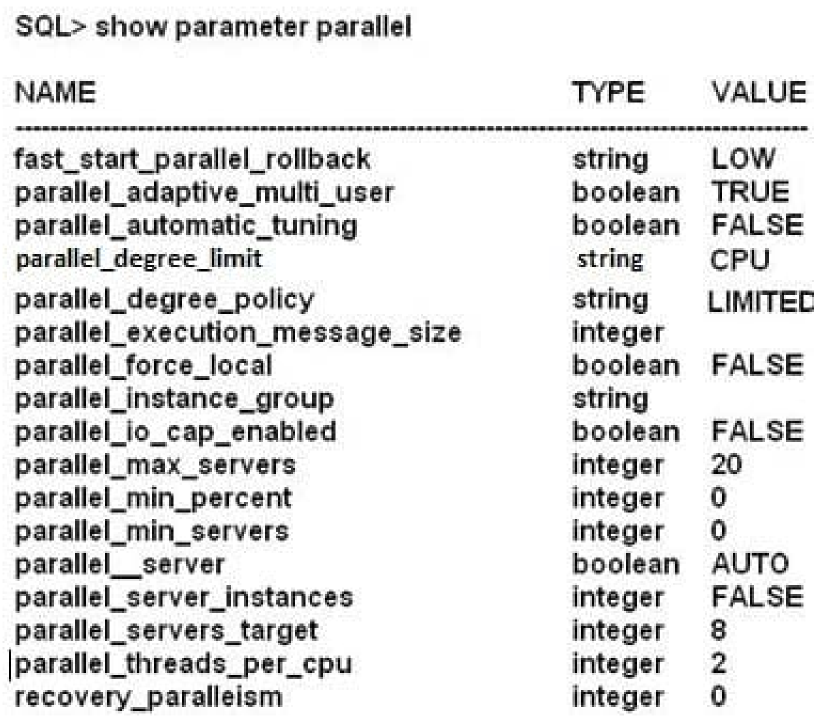
/ The join condition between two tables is not an equi-join.

/ Because of sorts already required by other operations, the optimizer finds it is cheaper to use a sort merge than a hash join. Reference: Oracle Database Performance Tuning Guide , Sort Merge Joins

---

QUESTION 6

Examine the parallel parameters for your instance.



```
SQL> show parameter parallel

NAME                                 TYPE      VALUE
------------------------------------------------------------
fast_start_parallel_rollback         string    LOW
parallel_adaptive_multi_user         boolean   TRUE
parallel_automatic_tuning            boolean   FALSE
parallel_degree_limit                string    CPU
parallel_degree_policy               string    LIMITED
parallel_execution_message_size      integer
parallel_force_local                 boolean   FALSE
parallel_instance_group              string
parallel_io_cap_enabled              boolean   FALSE
parallel_max_servers                 integer   20
parallel_min_percent                 integer   0
parallel_min_servers                 integer   0
parallel__server                     boolean   AUTO
parallel_server_instances            integer   FALSE
parallel_servers_target              integer   8
parallel_threads_per_cpu             integer   2
recovery_paralleism                  integer   0
```

All parallel execution servers are available and the session use default parallelism settings. In which two cases will the degree of parallelism be automatically calculated?

A. Statements accessing tables whom dictionary DOP is 2 or more

B. Statements accessing tables whose dictionary DOP is DEFAULT

C. Statements that are estimated to execute for more than 10 seconds serially

D. Statements accessing tables with any setting for Dictionary DOP

E. Statements with parallel hints

Correct Answer: BE

PARALLEL_DEGREE_POLICY specifies whether or not automatic degree of Parallelism, statement queuing, and in-memory parallel execution will be enabled.

Syntax PARALLEL_DEGREE_POLICY = { MANUAL | LIMITED | AUTO }

LIMITED is used here:

Enables automatic degree of parallelism for some statements but statement queuing and in-memory Parallel Execution are disabled. Automatic degree of parallelism is only applied to those statements that access tables or indexes decorated explicitly with the DEFAULT degree of parallelism using the PARALLEL clause. Statements that do not access any tables or indexes decorated with the

DEFAULT degree of parallelism will retain the MANUAL behavior.

Note:

*

In earlier versions of the Oracle Database, we had to determine the DOP more or less manually, either with a parallel hint or by setting a parallel degree with alter table. There was an automatic computation of the DOP available for the objects with dictionary DOP of default, derived from the simple formula CPU_COUNT

*

PARALLEL_THREADS_PER_CPU.

QUESTION 7

Which three statements are true about the usage of optimizer hints?

A. Whenever a query uses table aliases, the hints in the query must use the aliases.

B. The OPTIMIZER_FEATURES_ENABLE parameter must be set to a version supports the hints used.

C. The optimizer uses the execution plan with lower cost even if a hint is specified.

D. A schema name for the table must be used in the hint if the table us qualified in the FROM clause.

E. Hints can be used to override the optimization approach specified with the OPTIMIZER_MODE parameter.

F. A statement block can have only one hint, and that hint must be immediately after SELECT, UPDATE, INSERT, MERGE, or DELETE keyword.

Correct Answer: ABE

*

You must specify the table to be accessed exactly as it appears in the statement. If the statement uses an alias for the table, then use the alias rather than the table name in the hint.

*

OPTIMIZER_FEATURES_ENABLE acts as an umbrella parameter for enabling a series of optimizer features based on an Oracle release number.

For example, if you upgrade your database from release 10.1 to release 11.1, but you want to keep the release 10.1 optimizer behavior, you can do so by setting this parameter to 10.1.0. At a later time, you can try the enhancements introduced in releases up to and including release 11.1 by setting the parameter to

11.1.0.6.

* If a SQL statement has a hint specifying an optimization approach and goal, then the optimizer uses the specified approach regardless of the presence or absence of statistics, the value of the OPTIMIZER_MODE initialization parameter, and the OPTIMIZER_MODE parameter of the ALTER SESSION statement.

QUESTION 8

View the exhibit and examine the plans in the SQL baseline for a given statement. Which interpretation is correct?

A. A new plan cannot be evolved because SYS_SQL_bbedc41f554c408 is accepted.

B. Plan SYS_SQL_PLAN_bbdc741f554c408 will always be used by the optimizer for the query.

C. A new plan must be evolved using the DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE function before it can be used.

D. Plan SYS_SQL_bbedc741a57b5fc2 can be used by the optimizer if the cost of the query is less than plan SYS_SQL_PLAN_bbedc741f554c408.

E. Plan SYS_SQL_PLAN_bbedc741f554c408 will not be used until it is fixed by using the DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE function.

Correct Answer: C

Note:

*

Evolving a SQL plan baseline is the process by which the optimizer determines if non-accepted plans in the baseline should be accepted. As mentioned

previously, manually loaded plans are automatically marked as accepted, so manual loading forces the evolving process. When plans are loaded automatically,

the baselines are evolved using the

EVOLVE_SQL_PLAN_BASELINE function, which returns a CLOB reporting its results.

SET LONG 10000

SELECT DBMS_SPM.evolve_sql_plan_baseline(sql_handle =>

\\'SYS_SQL_7b76323ad90440b9\\')

FROM dual;

*

Manual plan loading can be used in conjunction with, or as an alternative to automatic plan capture. The load operations are performed using the DBMS_SPM package, which allows SQL plan baselines to be loaded from SQL tuning sets or from specific SQL statements in the cursor cache. Manually loaded statements

are flagged as accepted by default. If a SQL plan baseline is present for a SQL statement, the plan is added to the baseline, otherwise a new baseline is created.

*

fixed (YES/NO) : If YES, the SQL plan baseline will not evolve over time. Fixed plans are used in preference to non-fixed plans.

QUESTION 9

You are administering a database supporting a DDS workload in which some tables are updated frequently but not queried often. You have SQL plan baseline for these tables and you do not want the automatic maintenance task to gather statistics for these tables regularly.

Which task would you perform to achieve this?

A. Set the INCREMENTAL statistic preference FALSE for these tables.

B. Set the STALE_PERCENT static preference to a higher value for these tables.

C. Set the GRANULARITY statistic preference to AUTO for these tables.

D. Set the PUBLISH statistic preference to TRUE for these tables.

Correct Answer: B

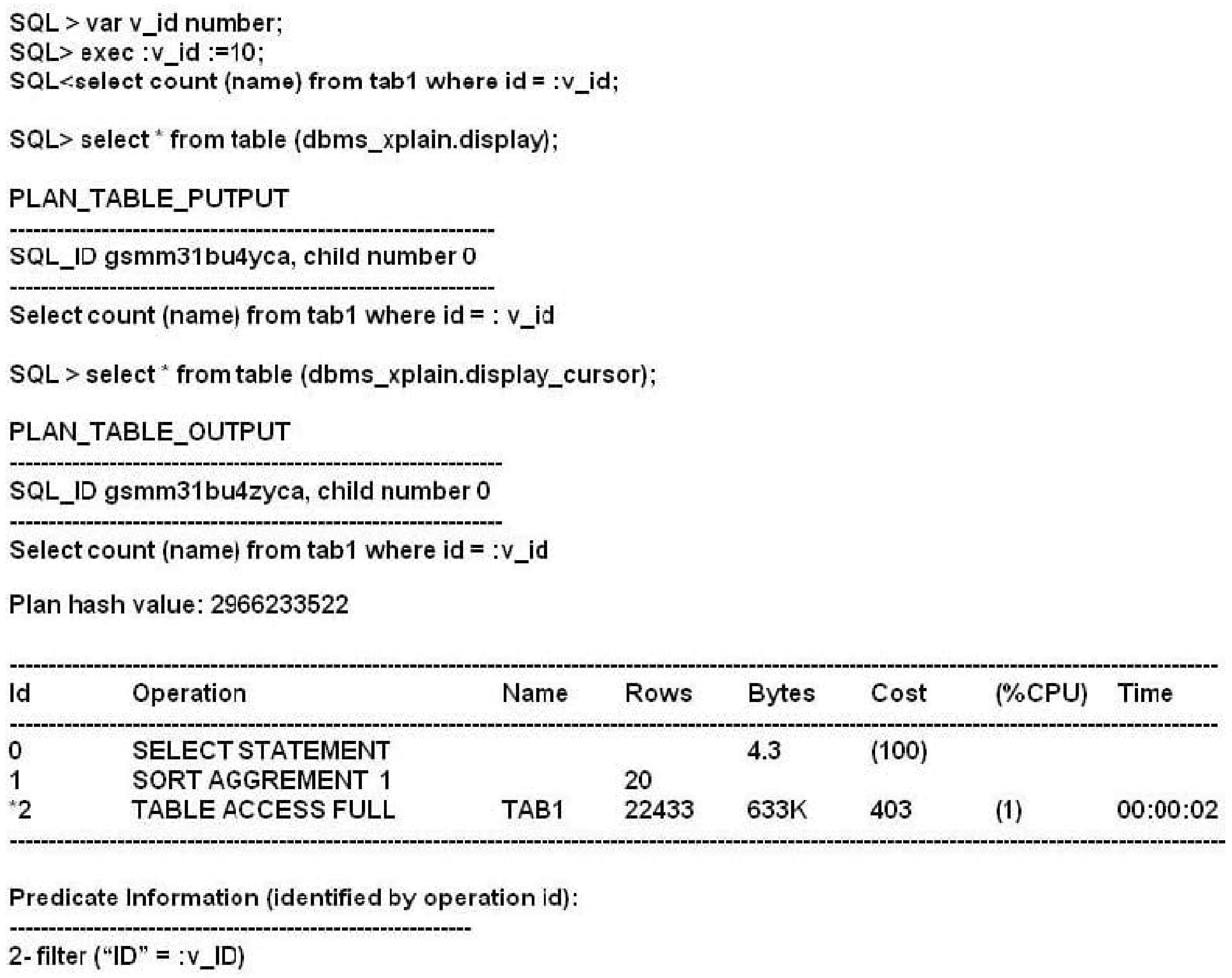
With the DBMS_STATS package you can view and modify optimizer statistics gathered for database objects.

STALE_PERCENT - This value determines the percentage of rows in a table that have to change before the statistics on that table are deemed stale and should be regathered. The default value is 10%. Reference: Oracle Database PL/SQL Packages and Types Reference

---

QUESTION 10

Exhibit



```
SQL > var v_id number;
SQL> exec :v_id :=10;
SQL<select count (name) from tab1 where id = :v_id;

SQL> select * from table (dbms_xplain.display);

PLAN_TABLE_PUTPUT
-------------------------------------------------------------
SQL_ID gsmm31bu4yca, child number 0
-------------------------------------------------------------
Select count (name) from tab1 where id = : v_id

SQL > select * from table (dbms_xplain.display_cursor);

PLAN_TABLE_OUTPUT
-------------------------------------------------------------
SQL_ID gsmm31bu4zyca, child number 0
-------------------------------------------------------------
Select count (name) from tab1 where id = :v_id

Plan hash value: 2966233522

------------------------------------------------------------------------------------------------------------------------
Id        Operation                     Name     Rows     Bytes     Cost      (%CPU)    Time
------------------------------------------------------------------------------------------------------------------------
0         SELECT STATEMENT                                4.3       (100)
1         SORT AGGREMENT  1                      20
*2        TABLE ACCESS FULL             TAB1     22433    633K      403       (1)       00:00:02
------------------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
-----------------------------------------------------------
2- filter ("ID" = :v_ID)
```

A table has three distinct values in its ID column. In the ID column, values 10 and 20 have more than 20000 rows each and value 30 has only five rows. The

statistics for the schema have been updated recently.

The CURSOR_SHARING parameter is set to EXACT.

The query was executed recently and the cursor for the query is bind aware. Examine the exhibits to view the commands executed.

You plan to execute the same query with a value of 30 for the bind variable V_ID.

Which statement is true in this scenario?

A. The same execution plan will always be used irrespective of the value in the bind variable.

B. A new execution plan will be generated depending on the access pattern and the bind value.

C. Adaptive cursor sharing will ensure that a new cursor is generated for each distinct value in the bind variable.

D. Adaptive cursor sharing will happen only if you use the literal values instead of bind variables in the query.

Correct Answer: C

Note:

*

 CURSOR_SHARING determines what kind of SQL statements can share the same cursors.

*

 Setting CURSOR_SHARING to EXACT allows SQL statements to share the SQL area only when their texts match exactly. This is the default behavior. Using

this setting, similar statements cannot shared; only textually exact statements can be shared.

*

 Values:

*

 FORCE

Forces statements that may differ in some literals, but are otherwise identical, to share a cursor, unless the literals affect the meaning of the statement.

*

 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.

*

 EXACT

Only allows statements with identical text to share the same cursor.

QUESTION 11

You plan to bulk load data using INSERT /*+PARALLEL*/ INTO . . . . SELECT FROM statements.

Which four types of operations can execute in parallel on tables that have no bitmapped indexes or materialized views defined on term?

A. Direct path insert of a million rows into a partitioned, index-organized table containing one million rows.

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

C. Direct path insert of a million rows into a nonpartitioned, index-organized table containing one million rows.

D. Direct path insert of a million rows into a nonpartitioned, heap-organized table containing 10 million rows.

E. Direct path insert of a million rows into a nonpartitioned, heap-organized table containing one million rows.

Correct Answer: ABDE

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.

QUESTION 12

You are administering a database that supports an OLTP workload in which one of the applications inserts rows in a table until 12 noon every, after which multiple years perform frequent queries on the table. You want the statistics to be more representative of the table population.

What must be done to ensure that an optimizer uses the latest statistics of the table?

A. Set the STALE_PERCENT preference to 0.

B. Set the OPTIMIZER_MODE parameter to ALL_ROWS.

C. Set the OPTIMIZER_DYNAMIC_SAMPLING parameter to 0.

D. Use the FIRST_ROWS_n hint in the queries.

E. Unlock and gather statistics for the table after inserts are done and lock them again.

Correct Answer: E

*

For tables that are substantially modified in batch operations, such as with bulk loads, gather statistics on these tables as part of the batch operation. Call the DBMS_STATS procedure as soon as the load operation completes.

* Statistics for a table or schema can be locked. After statistics are locked, you can make no modifications to the statistics until the statistics have been unlocked. Locking procedures are useful in a static environment in which you want to guarantee that the statistics never change.

The DBMS_STATS package provides two procedures for locking (LOCK_SCHEMA_STATS and LOCK_TABLE_STATS) and two procedures for unlocking statistics (UNLOCK_SCHEMA_STATS andUNLOCK_TABLE_STATS).

Incorrect:

A: STALE_PERCENT cannot be set to 0.

* With the DBMS_STATS package you can view and modify optimizer statistics gathered for database objects.

STALE_PERCENT - This value determines the percentage of rows in a table that have to change before the statistics on that table are deemed stale and should be regathered. The default value is 10%.

B: Optimizer_mode applies to the database, not to a specific table.

*

Possible values for optimizer_mode = choose/ all_rows/ first_rows/ first_rows[n]

Important facts about ALL_ROWS

1.

ALL_ROWS considers both index scan and full scan and based on their contribution to the overall query, it uses them. If Selectivity of a column is low, optimizer may use index to fetch the data (for example `where employee_code=7712\\'), but if selectivity of column is quite high (\\'where deptno=10\\'), optimizer may consider doing Full table scan. With ALL_ROWS, optimizer has more freedom to its job at its best.

2.

Good for OLAP system, where work happens in batches/procedures. (While some of the report may still use FIRST_ROWS depending upon the anxiety level of report reviewers)

3.

Likes hash joins over nested loop for larger data sets.

C: Optimizer dynamic sampling refers to the ability of the SQL optimizer to take a sample of rows from a table to calculate missing statistics. Dynamic sampling can be controlled with the OPTIMIZER_DYNAMIC_SAMPLING parameter or the DYNAMIC_SAMPLING hint.

level 0 - do not use dynamic sampling

D: First_row_n cannot be used in this way.