First you have to understand that the database itself is never slow or fast—it has a constant speed. The sessions connected to the database, however, slow down when they hit a bump in the road. To resolve a session performance issue, you need to identify the bump and remove it.

In my experience, the vast majority of poorly performing execution plans can be because of three reasons >

1. Poorly written query
2. Optimizer may not have sufficient information to correctly estimate cardinalities (estimated row counts) -> dbms\_stats.create\_extended\_stats(null,'customers', '(cust\_state\_province,country\_id)');
3. Optimiser does understand the data distribution. ->

By adding the FOR COLUMNS clause, you can have the database create the new column group as well as collect statistics for it, all in one step, as shown here:

exec dbms\_Stats.gather\_table\_stats(

ownname=>null,-

tabname=>'customers',-

method\_opt=>'for all columns size skewonly,-

for columns (cust\_state\_province,country\_id) size skewonly');

exec dbms\_stats.gather\_table\_stats(null,'customers',

method\_opt=>'for all columns size skewonly,

for columns (lower(cust\_state\_province)) size skewonly');

Some other are related to

Table locking because of concurrent access.

Unresonable Undo and Redo requirement

Unresonable Tempspace demand

Monitoring the session in v$session and getting more information about the problem

1. What is the query?

2. What is the execution plan.

3. What is the state of the session during most of the time of execution(idle, processing or waiting)

4. If the session is waiting then how long the session has been waiting(seconds\_in\_wait is curren wait and wait\_time is last wait time, wait\_time is in centi second), A very long wait usually indicates some sort of bottleneck.

5. if it is waiting then retrive more information on the event it is waiting (event column of v$session) "enq: TX - row lock contention" or "db file sequential read" on DML lock identify the object name and row number. if you see event "db file sequential read" then you know that the session is waiting for I/O from the disk to complete. To make the session go faster, you have to reduce that waiting period. There are several ways to reduce the wait one of the simple option is moving the data object to a faster disk or tune the query to do reduce the I/O required. for finding the data object refer the P1 and P2 column which shows the object id and segment id.

If the wait event is "enq: TX - row lock contention" then find out the exact object and row (dbms\_rowid.rowid\_create) which is locked and by which session.

6. if session is idle then it is not a database problem, may be the application server or ETL server is causing the delay.

7. Is the number of block I/O required is higher compared to the number of rows fetched, then may be rows are chained and rebuilding the table or index might be the solution.

8. OS level CPU/Memory spike is observed during the query running? May be because of rapid switching of logs,

9. Read consistency is causing the problem?

10. Is the index fragmented because of vast DML operation? If the value for DEL\_LF\_ROWS/LF\_ROWS is greater than 2, or LF\_ROWS is lower than LF\_BLKS, or HEIGHT is 4 then the index should be rebuilt.

12. Is the problem is with insert query during vast number of record inserttion, use of APEND hint direct path load, nologging might be the solution?

13. is the problem is during deleting major portion of rows from a table, then truncate and insert from tem table can be the solution? or partitioning to enble fast data purgning using partition drop might be the solution.

14. is it causing a full table scan and should we create a new index, what type of index, normal btree, bitmap, bitmap joined, reverse key, function based.

15. Should we use a GTT or IOT

16. Caching Small Tables in Memory

17. Histograms or extended stastics

Various tools like EXPLAIN PLAN, Using Autotrace, TKPROF, TOP, VMSTAT, MEMSTAT, can give fair amount of indication about what could be the problem. Once you understand where the time is being spent then you can take remedy action.

Select best Joining method

**Nested Loops**

If you're joining small subsets of data, the nested loop (NL) method is ideal. If you're returning fewer than, say, 10,000 rows, the NL join may be the right join method. If the optimizer is using hash joins or full table scans, force it to use the NL join method by using the following hint:

SELECT /\*+ USE\_NL (TableA, TableB) \*/

**Hash Join**

If the join will produce large subsets of data or a substantial proportion of a table is going to be joined, use the hash join hint if the optimizer indicates it isnt going to use it:

SELECT /\* USE\_HASH \*/

**Merge Join**

If the tables in the join are being joined with an inequality condition (not an equi join), the merge join method is ideal:

SELECT /\*+ USE\_MERGE (TableA, TableB) \*/

call count cpu elapsed disk query current rows

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

Parse 1 0.00 0.00 0 0 0 0

Execute 1 0.00 0.00 0 0 0 0

Fetch 17322 1.82 1.85 3 136 5 259806

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

total 17324 1.82 1.85 3 136 5 259806

ALTER SESSION SET EVENTS '10046 trace name context off';

EXEC DBMS\_SYSTEM.set\_sql\_trace\_in\_session(sid=>123, serial#=>1234, sql\_trace=>TRUE);

EXEC DBMS\_MONITOR.session\_trace\_enable(session\_id =>1234, serial\_num=>1234, waits=>TRUE, binds=>FALSE);

tkprof dev1\_ora\_367660.trc translated.txt explain=test/test table=sys.plan\_table sys=no waits=yes

function firstName(){ alert('First Name is ' + document.getElementById('P1\_FIRST\_NAME').value );

•How can you clear caches programmaticaly? (APEX\_UTIL.CLEAR\_APP\_CACHE, APEX\_UTIL.CLEAR\_PAGE\_CACHE, APEX\_UTIL.CLEAR\_USER\_CACHE