Oracle in memory

Benefit

Faster query performance. Complex queries with group by operations and highly selective query can perform much better with in memory database. This is very useful in online analytics.

How does it work:-

In a database without OIM, when user submits a query which reads from buffer cache. If data is not available in memory, it triggers physical read operations and results are returned to the user.

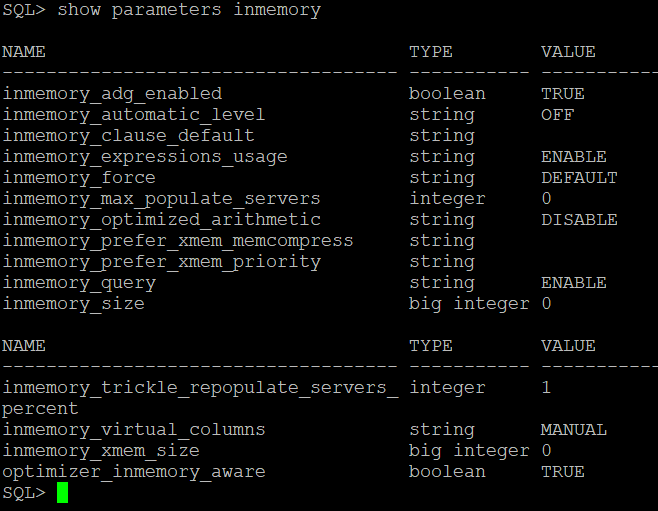
With OIM database, if all the needed data is held in memory, it executes entirely against the in memory cache.

Query can combine data from in-memory and buffer cache.

Getting started with in-memory:-

1. Set up Oracle in memory by changing oracle parameter

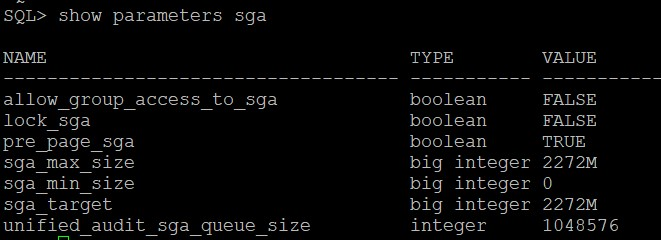
View in memory parameter



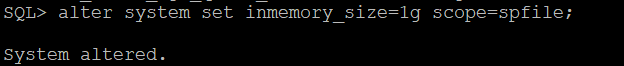
1. Change in memory parameter to enable OIM

Note:- Before changing in memory parameter, check SGA size because oracle in memory columns are part of SGA. It can be verified by checking SGA parameters.

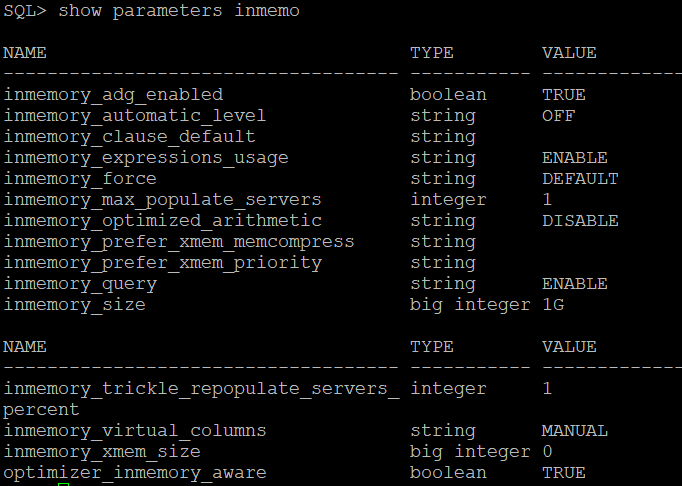
* Show parameters sga



* Update parameter



* Restart the data base and verify OIM(oracle in memory) parameter.



* In memory parameters settings is inherited by all PDB’s, unless it set manually

1. Disable in memory

* Alter system set inmemory\_force=off;

This means objects will not be maintained in memory.

* Alter system set inmemory\_force=default;

This bring OIM to default behaviour

* Alter system set inmemory\_query=disable;

This means optimizer will not consider the column to optimize the query.

* Alter system set inmemory\_query=enable;

This will set behaviour back to default and optimizer will start considering column for query optimization.

* Alter session set inmemory\_query=disable;
* Alter session set inmemory\_query=enable;
* Disable OIM completely and release memory.

Alter system reset inmemory\_size scope=spfile;

Shutdown immediate;

Startup;

1. Create table with in memory option or change in memory for existing table.

* Create table

create table im\_table(id number) inmemory;

create table noim\_table(id number) no inmemory; -- not in memory

create table default\_table(id number); -- not in memory

select table\_name,inmemory, inmemory\_priority,inmemory\_distribute,inmemory\_compression,inmemory\_duplicate from user\_tables

where table\_name like '%TABLE%'; -- verify in memory table

* Change status of existing table

alter table im\_table no inmemory;

alter table noim\_table inmemory memcompress for capacity low;

alter table default\_table inmemory priority high;

1. Create tables with column level in memory

* Create table with in memory setting at column level

create table im\_col\_table(

id number,col1 number,col2 number,col3 number,col4 number

) inmemory

inmemory memcompress for query high (col1,col2)

inmemory memcompress for capacity high(col3)

no inmemory(id,col4);

* Modify in memory at column level

alter table im\_col\_tab

no inmemory (col1, col2)

inmemory memcompress for capacity high (col3)

no inmemory (id, col4);

1. Controlling compression

* No memcompress: Data is held without compression.
* Memcompress for DML: Is mainly for DML performance and minimum compression.
* Memcompress for Query Low: Is optimized for query performance (Default).
* Memcompress for Query High: Is a balance of query performance and space.
* Memcompress for Capacity Low: For additional compression.
* Memcompress for Capacity High: Optimized for space over performance.

1. Load priority

Load priority determines the sequence in which tables are loaded into memory upon start-up or when tables are pinned or removed.

* Critical: Data is populated immediately.
* High: After all critical data is loaded.
* Medium: After high priority data is loaded.
* Low: After medium data is loaded.
* None: Data is only populated when it's read for the first time.

1. Impact on sql query

* Select \* from im\_table;

