Oracle Database 19c New Features Selected Highlights

Richard Foote Consulting

19c

Richard Foote

- Working in IT for 30+ years , 20+ years with Oracle Database
- 19 years employed in Australian Federal Government in various IT roles
- Worked for Oracle Corporation between 1996 and 2002 and between 2011 and 2017
- In September 2017, started my own independent company Richard Foote Consulting
- Been responsible for many large scale, mission critical, “life-dependant” classified Oracle systems, tuned numerous databases often with 10x performance improvements
- Oracle OakTable Member since 2002 and awarded Oracle ACE Director in 2008 & 2018
- Regular speaker at user group meetings and conferences such as Oracle OpenWorld, IOUG Collaborate, Hotsos Symposium, AUSOUG InSync, ODTUG Kscope, UKOUG Tech Conference, E4 Enkitec Extreme Exadata Expo, Trivadis Performance Days …
- UKOUG Lifetime Speaker award 2018 and AUSOUG Oracle Master award 2018
- Richard Foote's Oracle Blog: https://richardfoote.wordpress.com
- Richard Foote Consulting: https//richardfooteconsulting.com
- Spend as much free time as possible listening to the music of David Bowie !!

“Oracle Indexing Internals & Best Practices” 2 Day Seminar

LONDON: 23-24 March 2020

Of benefit to DBAs, Developers, Solution Architects and anyone else interested in designing, developing or maintaining high performance Oracle-based applications/databases.

- Examines most available index structures/options & discusses in considerable detail how indexes function, how/when they should be used & how they should be maintained.
- Details how indexes are costed & evaluated by the Cost Based Optimizer (CBO) & how appropriate data management practices are vital for an effective indexing strategy.
- Covers many useful tips and strategies to maximise the benefits of indexes on application/database performance & scalability.

richardfooteconsulting.com/indexing-webinar/

“Oracle Performance Diagnostics and Tuning” 2 Day Seminar

LONDON: 25-26 March 2020

- A must attend seminar aimed at Oracle professionals (both DBAs and Developers) who are interested in Oracle Performance Tuning.
- Details how to maximise the performance of both Oracle databases and associated applications and how to diagnose and address any performance issues as quickly and effectively as possible.
- The focus is at both the Database and individual SQL layers.
- Bring your own AWR reports and have them reviewed following the Tuning Methodology covered in the seminar to help find and resolve performance issues with your databases/applications !!

richardfooteconsulting.com/performance-tuning-seminar/

NEW IN 19c

19c Database New Features: Selected Highlights

- Partial JSON Update Support
- Distinct Option For LISTAGG Aggregate
- Real-Time Statistics
- High Frequency Statistics Gathering
- Hybrid Partitioned Tables
- Memoptimized Rowstore: Fast Ingest
- SQL Quarantine
- Automatic Indexing

19c

NEW IN 19c

Partial JSON Update Support

{JSON}

```
SQL> CREATE TABLE ziggy_json
  2  (id          number,
  3   ziggy_date date,
  4   ziggy_order CLOB
  5  CONSTRAINT ziggy_json_check CHECK (ziggy_order IS JSON));

Table created.

SQL> insert into ziggy_json values (1, sysdate, '{"This is not legal JSON"}');
insert into ziggy_json values (1, sysdate, '{"This is not legal JSON"}')
*
ERROR at line 1:
ORA-02290: check constraint (BOWIE.ZIGGY_JSON_CHECK) violated
```

NEW IN 19c

Partial JSON Update Support

{JSON}

```
SQL> insert into ziggy_json
  2  select
  3      rownum,
  4      SYSdate,
  5      '{"PONumber"             : ' || rownum || ',
  6          "Reference"          : "2018' || rownum || 'DBOWIE",
  7        "Requestor"            : "David Bowie",
  8        "User"                 : "DBOWIE",
  9        "CostCenter"           : "A42",
 10        "ShippingInstructions" : {"name"   : "David Bowie",
 11                                  "Address": {"street"  : "42 Ziggy Street",
 12                                              "city"    : "Canberra",
 13                                              "state"   : "ACT",
 14                                              "zipCode" : 2601,
 15                                              "country" : "Australia"},
 16                                  "Phone" : [{"type" : "Office", "number" : "417-555-7777"},
 17                                             {"type" : "Mobile", "number" : "417-555-1234"}]},
 18        "Special Instructions" : null,
 19        "AllowPartialShipment" : true,
 20        "LineItems"            : [{"ItemNumber" : 1,
 21                                   "Part"       : {"Description" : "Hunky Dory",
 22                                                   "UnitPrice"   : 10.95},
 23                                   "Quantity"   : 5.0},
 24                                  {"ItemNumber" : 2,
 25                                   "Part"       : {"Description" : "Pin-Ups",
 26                                                   "UnitPrice"   : 10.95},
 27                                   "Quantity"   : 3.0}]}'
 28  from dual connect by level <= 1000000;
1000000 rows created.
SQL> insert into ziggy_json select * from ziggy_json;
1000000 rows created.
SQL> commit;
Commit complete
```

NEW IN 19c

Partial JSON Update Support

{JSON}

```
SQL> CREATE INDEX ziggy_po_num_i ON ziggy_json z (z.ziggy_order.PONumber);

Index created.

SQL> CREATE INDEX ziggy_reference_i ON ziggy_json z (z.ziggy_order.Reference);

Index created.

SQL> select z.ziggy_order.PONumber, z.ziggy_order.Reference from ziggy_json z
     where z.ziggy_order.PONumber='42';

PONUMBER   REFERENCE
---------- ---------------
42         201742DBOWIE
42         201742DBOWIE

SQL> select z.ziggy_order.PONumber, z.ziggy_order.Reference from ziggy_json z
     where z.ziggy_order.Reference='ZIGGY STARDUST';

no rows selected
```

NEW IN 19c

Partial JSON Update Support

{JSON}

```
SQL> UPDATE ziggy_json z SET ziggy_order=json_mergepatch(ziggy_order, '{"Reference":"ZIGGY STARDUST"}') where z.ziggy_order.PONumber='42';

2 rows updated.

Execution Plan
----------------------------------------------------------------------------------------------------------
| Id  | Operation                                | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------------------------------
|   0 | UPDATE STATEMENT                         |                | 20000 |    55M|  8008   (1)| 00:00:01 |
|   1 |  UPDATE                                  | ZIGGY_JSON     |       |       |            |          |
|   2 |   OPTIMIZER STATISTICS GATHERING         |                | 20000 |    55M|  8008   (1)| 00:00:01 |
|   3 |    TABLE ACCESS BY INDEX ROWID BATCHED| ZIGGY_JSON     | 20000 |    55M|  8008   (1)| 00:00:01 |
|*  4 |     INDEX RANGE SCAN                     | ZIGGY_PO_NUM_I |  8000 |       |     3   (0)| 00:00:01 |
----------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - access(JSON_QUERY("ZIGGY_ORDER" FORMAT JSON , '$.PONumber' RETURNING VARCHAR2(4000) ASIS
              WITHOUT ARRAY WRAPPER NULL ON ERROR)='42')

Statistics
----------------------------------------------------------
         46  recursive calls
         32  db block gets
         53  consistent gets
          4  physical reads
      12952  redo size
        195  bytes sent via SQL*Net to client
        483  bytes received via SQL*Net from client
          1  SQL*Net roundtrips to/from client
          2  sorts (memory)
          0  sorts (disk)
          2  rows processed
```

NEW IN 19c

Partial JSON Update Support

{JSON}

```
SQL> SELECT z.ziggy_order.PONumber, z.ziggy_order.Reference FROM ziggy_json z
     where z.ziggy_order.Reference='ZIGGY STARDUST';

PONUMBER   REFERENCE
---------- ---------------
42         ZIGGY STARDUST
42         ZIGGY STARDUST


Execution Plan
----------------------------------------------------------------------------------------------------------
| Id  | Operation                             | Name              | Rows  | Bytes| Cost (%CPU)| Time      |
----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                      |                   |     2 |  8008|     6   (0)| 00:00:01  |
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| ZIGGY_JSON        |     2 |  8008|     6   (0)| 00:00:01  |
|*  2 |   INDEX RANGE SCAN                    | ZIGGY_REFERENCE_I |     2 |      |     3   (0)| 00:00:01  |
----------------------------------------------------------------------------------------------------------
```

NEW IN 19c

Distinct Option For LISTAGG Aggregate

```
SQL> select d.department_name,
  2         listagg (e.job_id,', ' on overflow truncate with count)
  3                  within group (order by e.job_id) jobs
  4  from departments d, employees e
  5  where d.department_id = e.department_id
  6  group by d.department_name;


DEPARTMENT_NAME                JOBS
------------------------------ --------------------------------------------------------------------------------
Accounting                     AC_ACCOUNT, AC_MGR
Administration                 AD_ASST
Executive                      AD_PRES, AD_VP, AD_VP
Finance                        FI_ACCOUNT, FI_ACCOUNT, FI_ACCOUNT, FI_ACCOUNT, FI_ACCOUNT, FI_MGR
Human Resources                HR_REP
IT                             IT_PROG, IT_PROG, IT_PROG, IT_PROG, IT_PROG
Marketing                      MK_MAN, MK_REP
Public Relations               PR_REP
...
```

NEW IN 19c

Distinct Option For LISTAGG Aggregate

```
SQL> select d.department_name,
  2         (select listagg(job_id,', ' on overflow truncate with count)
  3                 within group (order by job_id)
  4          from (select unique job_id
  5                from employees e
  6                where d.department_id = e.department_id)) jobs
  7  from departments d
  8  where exists (select e.department_id from employees e where d.department_id=e.department_id);


DEPARTMENT_NAME                JOBS
------------------------------ --------------------------------------------------------------------------------
Executive                      AD_PRES, AD_VP
IT                             IT_PROG
Finance                        FI_ACCOUNT, FI_MGR
Purchasing                     PU_CLERK, PU_MAN
Shipping                       SH_CLERK, ST_CLERK, ST_MAN
Sales                          SA_MAN, SA_REP
Administration                 AD_ASST
Marketing                      MK_MAN, MK_REP
...
```

NEW IN 19c

Distinct Option For LISTAGG Aggregate

```
SQL> select d.department_name,
  2         listagg (distinct e.job_id,', ' on overflow truncate with count)
  3                  within group (order by e.job_id) jobs
  4  from departments d, employees e
  5  where d.department_id = e.department_id
  6  group by d.department_name;


DEPARTMENT_NAME                JOBS
------------------------------ --------------------------------------------------------------------------------
Accounting                     AC_ACCOUNT, AC_MGR
Administration                 AD_ASST
Executive                      AD_PRES, AD_VP
Finance                        FI_ACCOUNT, FI_MGR
Human Resources                HR_REP
IT                             IT_PROG
Marketing                      MK_MAN, MK_REP
Public Relations               PR_REP
...
```

NEW IN 19c

Real-Time Statistics

- Up-to-date statistics – eliminate lag
- Only for some statistics (NUM_ROWS, LOW_VALUE/HIGH_VALUE)
- Better SQL execution plans
- Negligible overhead
- Automated and transparent
- BUT only available on Exadata Platforms

NEW IN 19c

Real-Time Statistics

```
SQL> create table bowie_stats (id number, code number, name varchar2(42));

Table created.

SQL> insert into bowie_stats select rownum, mod(rownum,100)+1, 'David Bowie'
     from dual connect by level <= 1000000;

1000000 rows created.

SQL> commit;

Commit complete.

SQL> select table_name, num_rows, blocks, sample_size, notes from user_tab_statistics
     where table_name='BOWIE_STATS';

TABLE_NAME        NUM_ROWS     BLOCKS SAMPLE_SIZE NOTES
--------------- ---------- ---------- ----------- ------------------------------
BOWIE_STATS

SQL> select column_name, num_distinct, low_value, high_value, sample_size, notes from user_tab_col_statistics
     where table_name='BOWIE_STATS';

no rows selected
```

NEW IN 19c

Real-Time Statistics

```
SQL> exec dbms_stats.gather_table_stats(ownname=>null, tabname=>'BOWIE_STATS');

PL/SQL procedure successfully completed.

SQL> select table_name, num_rows, blocks, sample_size, notes from user_tab_statistics
     where table_name='BOWIE_STATS';

TABLE_NAME        NUM_ROWS     BLOCKS SAMPLE_SIZE NOTES
--------------- ---------- ---------- ----------- ------------------------------
BOWIE_STATS        1000000       3520     1000000


SQL> select column_name, num_distinct, low_value, high_value, sample_size, notes from user_tab_col_statistics
      where table_name='BOWIE_STATS';

COLUMN_NAME     NUM_DISTINCT LOW_VALUE              HIGH_VALUE              SAMPLE_SIZE NOTES
--------------- ------------ ---------------------- ---------------------- ----------- -------------------------
ID                   1000000 C102                   C402                        1000000
CODE                     100 C102                   C202                        1000000
NAME                       1 446176696420426F776965 446176696420426F776965      1000000
```

NEW IN 19c

Real-Time Statistics

```
SQL> insert into bowie_stats select rownum+1000000, mod(rownum,110)+1, 'David Bowie'
     from dual connect by level<=1000;

1000 rows created.

Execution Plan
----------------------------------------------------------------------------------------
| Id  | Operation                          | Name        | Rows  | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------------
|   0 | INSERT STATEMENT                   |             |     1 |     2   (0)| 00:00:01 |
|   1 |  LOAD TABLE CONVENTIONAL           | BOWIE_STATS |       |            |          |
|   2 |   OPTIMIZER STATISTICS GATHERING   |             |     1 |     2   (0)| 00:00:01 |
|   3 |    COUNT                           |             |       |            |          |
|*  4 |     CONNECT BY WITHOUT FILTERING   |             |       |            |          |
|   5 |      FAST DUAL                     |             |     1 |     2   (0)| 00:00:01 |
----------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - filter(LEVEL<=1000)

Note
-----
   - dynamic statistics used: statistics for conventional DML
```

NEW IN 19c

Real-Time Statistics

```
SQL> EXEC DBMS_STATS.FLUSH_DATABASE_MONITORING_INFO;

PL/SQL procedure successfully completed.

SQL> select table_name, num_rows, blocks, sample_size, notes from user_tab_statistics
     where table_name='BOWIE_STATS';

TABLE_NAME        NUM_ROWS     BLOCKS SAMPLE_SIZE NOTES
--------------- ---------- ---------- ----------- ------------------------------
BOWIE_STATS        1000000       3520     1000000
BOWIE_STATS        1001000       3520             STATS_ON_CONVENTIONAL_DML

SQL> select column_name, num_distinct, low_value, high_value, sample_size, notes from user_tab_col_statistics
where table_name='BOWIE_STATS';

COLUMN_NAME     NUM_DISTINCT LOW_VALUE              HIGH_VALUE              SAMPLE_SIZE NOTES
--------------- ------------ ---------------------- ---------------------- ----------- -------------------------
ID                   1000000 C102                   C402                        1000000
CODE                     100 C102                   C202                        1000000
NAME                       1 446176696420426F776965 446176696420426F776965      1000000
ID                           C102                   C402010A04                       16 STATS_ON_CONVENTIONAL_DML
CODE                         C102                   C20208                           16 STATS_ON_CONVENTIONAL_DML
NAME                         446176696420426F776965 446176696420426F776965           16 STATS_ON_CONVENTIONAL_DML
```

NEW IN 19c

Real-Time Statistics

```
SQL> select * from bowie_stats;

1001000 rows selected.

Execution Plan
----------------------------------------------------------
Plan hash value: 1291560391

-------------------------------------------------------------------------------
| Id  | Operation          | Name        | Rows  | Bytes | Cost (%CPU)| Time|
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |             |  1001K|    19M|   971   (2)| 00:00:01|
|   1 |  TABLE ACCESS FULL| BOWIE_STATS |  1001K|    19M|   971   (2)| 00:00:01|
-------------------------------------------------------------------------------

Note
-----
   - dynamic statistics used: statistics for conventional DML
```

NEW IN 19c

High-Frequency Automatic Stats Collection

```
SQL> EXEC DBMS_STATS.SET_GLOBAL_PREFS('AUTO_TASK_STATUS','ON'); <= Turn on High-Frequency Auto Stats Collection

PL/SQL procedure successfully completed.

SQL> EXEC DBMS_STATS.SET_GLOBAL_PREFS('AUTO_TASK_MAX_RUN_TIME','900'); <= Run for up to 900 seconds (15 mins)

PL/SQL procedure successfully completed.

SQL> EXEC DBMS_STATS.SET_GLOBAL_PREFS('AUTO_TASK_INTERVAL','900'); <= Run every 900 seconds (15 mins)

PL/SQL procedure successfully completed.
```

NEW IN 19c

High-Frequency Automatic Stats Collection

```
SQL> create table bowie1 (id number, code number, name varchar2(42));  <= Stale with no stats
Table created.
SQL> insert into bowie1 select rownum, mod(rownum, 100)+1, 'David Bowie' from dual connect by level <= 100000;
100000 rows created.
SQL> commit;
Commit complete.
SQL> create table bowie2 (id number, code number, name varchar2(42));
Table created.
SQL> insert into bowie2 select rownum, mod(rownum, 100)+1, 'David Bowie' from dual connect by level <= 100000;
100000 rows created.
SQL> commit;
Commit complete.
SQL> exec dbms_stats.gather_table_stats(ownname=>null, tabname=>'BOWIE2');
PL/SQL procedure successfully completed.
SQL> insert into bowie2 select rownum+100000, mod(rownum, 100)+1, 'Ziggy Stardust' from dual connect by level <= 50000;
50000 rows created.
SQL> commit;
Commit complete.     <= Stale with outdated stats
```

NEW IN 19c

High-Frequency Automatic Stats Collection

```
create table bowie3 (id number, code number, name varchar2(42));

SQL> insert into bowie3 select rownum, 10, 'DAVID BOWIE' from dual connect by level <=1000000;

1000000 rows created.

SQL> update bowie3 set code = 9 where mod(id,3) = 0;

333333 rows updated.

SQL> update bowie3 set code = 1 where mod(id,2) = 0 and id between 1 and 20000;

10000 rows updated.

SQL> update bowie3 set code = 2 where mod(id,2) = 0 and id between 30001 and 40000;

5000 rows updated.

SQL> update bowie3 set code = 3 where mod(id,100) = 0 and id between 300001 and 400000;

1000 rows updated.

SQL> update bowie3 set code = 4 where mod(id,100) = 0 and id between 400001 and 500000;

1000 rows updated.

SQL> update bowie3 set code = 5 where mod(id,100) = 0 and id between 600001 and 700000;

1000 rows updated.
```

NEW IN 19c

High-Frequency Automatic Stats Collection

```
SQL> update bowie3 set code = 6 where mod(id,1000) = 0 and id between 700001 and 800000;

100 rows updated.

SQL> update bowie3 set code = 7 where mod(id,1000) = 0 and id between 800001 and 900000;

100 rows updated.

SQL> update bowie3 set code = 8 where mod(id,1000) = 0 and id between 900001 and 1000000;

100 rows updated.

SQL> commit;

Commit complete.

SQL> exec dbms_stats.gather_table_stats(ownname=>null, tabname=>'bowie3', estimate_percent=>100);

PL/SQL procedure successfully completed.  <= Missing histogram statistics

SQL> select code, count(*) from bowie3 group by code order by code;

      CODE   COUNT(*)
---------- ----------
         1      10000
         2       5000
         3       1000
         4       1000
         5       1000
         6        100
         7        100
         8        100
         9     327235
        10     654465
```

NEW IN 19c

High-Frequency Automatic Stats Collection

```
SQL> select * from bowie3 where code=7;

100 rows selected.

Execution Plan
----------------------------------------------------------
Plan hash value: 2517725203

-------------------------------------------------------------------------------
| Id  | Operation          | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |        |   100K|  1953K|   974   (2)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| BOWIE3 |   100K|  1953K|   974   (2)| 00:00:01 |
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("CODE"=7)
```

NEW IN 19c

High-Frequency Automatic Stats Collection

Current table statistics:

```
SQL> select table_name, num_rows, stale_stats, notes from user_tab_statistics
     where table_name in ('BOWIE1', 'BOWIE2', 'BOWIE3');

TABLE_NAME        NUM_ROWS STALE_S NOTES
--------------- ---------- ------- ------------------------------
BOWIE1
BOWIE2              100000 YES
BOWIE3             1000000 NO
BOWIE2              150000         STATS_ON_CONVENTIONAL_DML


SQL> select column_name, num_buckets, histogram from user_tab_col_statistics where table_name='BOWIE3';

COLUMN_NAME     NUM_BUCKETS HISTOGRAM
--------------- ----------- ---------------
ID                        1 NONE
CODE                      1 NONE
NAME                      1 NONE
```

NEW IN 19c

High-Frequency Automatic Stats Collection

After the High-Frequency Auto Stats task completes (approx. 15 minutes)

```
SQL> select table_name, num_rows, stale_stats from user_tab_statistics where
     table_name in ('BOWIE1', 'BOWIE2', 'BOWIE3');

TABLE_NAME         NUM_ROWS STALE_S
---------------- ---------- -------
BOWIE1               100000 NO
BOWIE2               150000 NO
BOWIE3              1000000 NO

SQL> select column_name, num_buckets, histogram from user_tab_col_statistics where table_name='BOWIE3';

COLUMN_NAME     NUM_BUCKETS HISTOGRAM
--------------- ----------- ---------------
ID                        1 NONE
CODE                     10 FREQUENCY
NAME                      1 NONE
```

NEW IN 19c

High-Frequency Automatic Stats Collection

```
SQL> select * from bowie3 where code=7;

100 rows selected.

Execution Plan
----------------------------------------------------------
Plan hash value: 2517725203

-------------------------------------------------------------------------------
| Id  | Operation          | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |        |   100 |  2000 |   974   (2)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| BOWIE3 |   100 |  2000 |   974   (2)| 00:00:01 |
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("CODE"=7)
```

NEW IN 19c

Indexing Hybrid Partitioned Tables

```
CREATE DIRECTORY old_bowie_data AS 'c:\app\old_bowie_data';

Directory created.

SQL> CREATE TABLE hybrid_bowie(id number, album_id number, country_id number, release_date date, total_sales number)
       EXTERNAL PARTITION ATTRIBUTES (
         TYPE ORACLE_LOADER
         DEFAULT DIRECTORY old_bowie_data
         ACCESS PARAMETERS (
           FIELDS TERMINATED BY ','
           (id, album_id, country_id, release_date DATE 'dd-mm-yyyy', total_sales))
       REJECT LIMIT UNLIMITED)
       PARTITION BY RANGE (release_date)
         (PARTITION ALBUMS_2010 VALUES LESS THAN (TO_DATE('01-JAN-2011', 'DD-MON-YYYY'))
          EXTERNAL DEFAULT DIRECTORY old_bowie_data LOCATION ('bowie2010_data.txt'),
          PARTITION ALBUMS_2011 VALUES LESS THAN (TO_DATE('01-JAN-2012', 'DD-MON-YYYY'))
          EXTERNAL DEFAULT DIRECTORY old_bowie_data LOCATION ('bowie2011_data.txt'),
          PARTITION ALBUMS_2012 VALUES LESS THAN (TO_DATE('01-JAN-2013', 'DD-MON-YYYY'))
          EXTERNAL DEFAULT DIRECTORY old_bowie_data LOCATION ('bowie2012_data.txt'),
          PARTITION ALBUMS_2013 VALUES LESS THAN (TO_DATE('01-JAN-2014', 'DD-MON-YYYY'))
          EXTERNAL DEFAULT DIRECTORY old_bowie_data LOCATION ('bowie2013_data.txt'),
          PARTITION ALBUMS_2014 VALUES LESS THAN (TO_DATE('01-JAN-2015', 'DD-MON-YYYY'))
          EXTERNAL DEFAULT DIRECTORY old_bowie_data LOCATION ('bowie2014_data.txt'),
          PARTITION ALBUMS_2015 VALUES LESS THAN (TO_DATE('01-JAN-2016', 'DD-MON-YYYY')),
          PARTITION ALBUMS_2016 VALUES LESS THAN (TO_DATE('01-JAN-2017', 'DD-MON-YYYY')),
          PARTITION ALBUMS_2017 VALUES LESS THAN (TO_DATE('01-JAN-2018', 'DD-MON-YYYY')),
          PARTITION ALBUMS_2018 VALUES LESS THAN (TO_DATE('01-JAN-2019', 'DD-MON-YYYY')),
          PARTITION ALBUMS_2019 VALUES LESS THAN (MAXVALUE));

Table created.
```

NEW IN 19c

Hybrid Partitioned Tables

```
SQL> INSERT INTO hybrid_bowie SELECT rownum+1000000, mod(rownum,5000)+1, mod(rownum,100)+1, to_date('31-DEC-2015','dd-mon-
yyyy')-mod(rownum,365), ceil(dbms_random.value(1,500000)) FROM dual CONNECT BY LEVEL <= 200000;

200000 rows created.

SQL> INSERT INTO hybrid_bowie SELECT rownum+1200000, mod(rownum,5000)+1, mod(rownum,100)+1, to_date('31-DEC-2016','dd-mon-
yyyy')-mod(rownum,366), ceil(dbms_random.value(1,500000)) FROM dual CONNECT BY LEVEL <= 200000;

200000 rows created.

SQL> INSERT INTO hybrid_bowie SELECT rownum+1400000, mod(rownum,5000)+1, mod(rownum,100)+1, to_date('31-DEC-2017','dd-mon-
yyyy')-mod(rownum,365), ceil(dbms_random.value(1,500000)) FROM dual CONNECT BY LEVEL <= 200000;

200000 rows created.

SQL> INSERT INTO hybrid_bowie SELECT rownum+1600000, mod(rownum,5000)+1, mod(rownum,100)+1, to_date('31-DEC-2018','dd-mon-
yyyy')-mod(rownum,365), ceil(dbms_random.value(1,500000)) FROM dual CONNECT BY LEVEL <= 200000;

200000 rows created.

SQL> INSERT INTO hybrid_bowie SELECT rownum+1800000, mod(rownum,5000)+1, mod(rownum,100)+1, to_date('31-DEC-2019','dd-mon-
yyyy')-mod(rownum,365), ceil(dbms_random.value(1,500000)) FROM dual CONNECT BY LEVEL <= 200000;

200000 rows created.

SQL> COMMIT;

Commit complete.

SQL> exec dbms_stats.gather_table_stats(ownname=> null, tabname=> 'HYBRID_BOWIE');

PL/SQL procedure successfully completed.
```

NEW IN 19c

Hybrid Partitioned Tables

```
SQL> select hybrid from user_tables where table_name='HYBRID_BOWIE';

HYBRID
------
YES

SQL> select partition_name, num_rows, blocks, segment_created from user_tab_partitions where table_name='HYBRID_BOWIE';

PARTITION_NAME         NUM_ROWS     BLOCKS SEGM
-------------------- ---------- ---------- ----
ALBUMS_2010              200000        611 YES
ALBUMS_2011              200000        611 YES
ALBUMS_2012              200000        611 YES
ALBUMS_2013              200000        611 YES
ALBUMS_2014              200000        611 YES
ALBUMS_2015              200000        635 YES
ALBUMS_2016              200000        635 YES
ALBUMS_2017              200000        635 YES
ALBUMS_2018              200000        635 YES
ALBUMS_2019              200000        635 YES

SQL> select partition_name, header_file, header_block, blocks from dba_segments where segment_name = 'HYBRID_BOWIE';

PARTITION_NAME       HEADER_FILE HEADER_BLOCK     BLOCKS
-------------------- ----------- ------------ ----------
ALBUMS_2015                    7       215185       1024
ALBUMS_2016                    7       216209       1024
ALBUMS_2017                    7       217233       1024
ALBUMS_2018                    7       226577       1024
ALBUMS_2019                    7       227601       1024
```

NEW IN 19c

Hybrid Partitioned Tables

```
SQL> create unique index hybrid_bowie_id_i on hybrid_bowie(id);
create unique index hybrid_bowie_id_i on hybrid_bowie(id)
                                        *
ERROR at line 1:
ORA-14354: operation not supported for a hybrid-partitioned table

SQL> alter table hybrid_bowie add primary key(id);
alter table hybrid_bowie add primary key(id)
                            *
ERROR at line 1:
ORA-14354: operation not supported for a hybrid-partitioned table

SQL> alter table hybrid_bowie add primary key(id) rely disable;

Table altered.

SQL> create index hybrid_bowie_id_i on hybrid_bowie(id);
create index hybrid_bowie_id_i on hybrid_bowie(id)
                                  *
ERROR at line 1:
ORA-14354: operation not supported for a hybrid-partitioned table


SQL> create index hybrid_bowie_id_i on hybrid_bowie(id) local;
create index hybrid_bowie_id_i on hybrid_bowie(id)
                                  *
ERROR at line 1:
ORA-14354: operation not supported for a hybrid-partitioned table
```

NEW IN 19c

Hybrid Partitioned Tables

```
SQL> alter table hybrid_bowie modify default attributes indexing off;
Table altered.
SQL> alter table hybrid_bowie modify partition albums_2015 indexing on;
Table altered.
SQL> alter table hybrid_bowie modify partition albums_2016 indexing on;
Table altered.
SQL> alter table hybrid_bowie modify partition albums_2017 indexing on;
Table altered.
SQL> alter table hybrid_bowie modify partition albums_2018 indexing on;
Table altered.
SQL> alter table hybrid_bowie modify partition albums_2019 indexing on;
Table altered.
SQL> create index hybrid_bowie_id_i on hybrid_bowie(id) indexing partial;
Index created.
SQL> create index hybrid_bowie_total_sales_i on hybrid_bowie(total_sales) local indexing partial;
Index created.
```

NEW IN 19c

Hybrid Partitioned Tables

```
SQL> select * from hybrid_bowie where id=1424242 and release_date between '01-JAN-2017' and '31-DEC-2017';

Elapsed: 00:00:00.00

Execution Plan
----------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                                    | Name              | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
----------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                             |                   |     1 |    26 |     4   (0)| 00:00:01 |       |       |
|*  1 |  TABLE ACCESS BY GLOBAL INDEX ROWID BATCHED| HYBRID_BOWIE      |     1 |    26 |     4   (0)| 00:00:01 |     8 |     8 |
|*  2 |   INDEX RANGE SCAN                           | HYBRID_BOWIE_ID_I |     1 |       |     3   (0)| 00:00:01 |       |       |
----------------------------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("RELEASE_DATE">=TO_DATE('2017-01-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss') AND
              "RELEASE_DATE"<=TO_DATE('2017-12-31 00:00:00', 'syyyy-mm-dd hh24:mi:ss'))
   2 - access("ID"=1424242)

Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
          5  consistent gets
          0  physical reads
          0  redo size
        888  bytes sent via SQL*Net to client
        462  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

NEW IN 19c

Hybrid Partitioned Tables

```
SQL> select * from hybrid_bowie where id=1424242;

Elapsed: 00:00:16.64

Execution Plan
----------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                                | Name               | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
----------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |                    |     1 |    25 |   854   (3)| 00:00:01 |       |       |
|   1 |  VIEW                                    | VW_TE_2            |     2 |   122 |   854   (3)| 00:00:01 |       |       |
|   2 |   UNION-ALL                              |                    |       |       |            |          |       |       |
|*  3 |    TABLE ACCESS BY GLOBAL INDEX ROWID BATCHED| HYBRID_BOWIE       |     1 |    25 |     4   (0)| 00:00:01 | ROWID | ROWID |
|*  4 |     INDEX RANGE SCAN                     | HYBRID_BOWIE_ID_I  |     1 |       |     3   (0)| 00:00:01 |       |       |
|   5 |    PARTITION RANGE ITERATOR              |                    |     1 |    25 |   850   (3)| 00:00:01 |     1 |     5 |
|*  6 |     EXTERNAL TABLE ACCESS FULL           | HYBRID_BOWIE       |     1 |    25 |   850   (3)| 00:00:01 |     1 |     5 |
----------------------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   3 - filter("HYBRID_BOWIE"."RELEASE_DATE">=TO_DATE(' 2015-01-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss') OR
              "HYBRID_BOWIE"."RELEASE_DATE" IS NULL)
   4 - access("ID"=1424242)
   6 - filter("ID"=1424242)

Statistics
----------------------------------------------------------
         56  recursive calls
          4  db block gets
         50  consistent gets
          0  physical reads
        560  redo size
        888  bytes sent via SQL*Net to client
        405  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

NEW IN 19c

Hybrid Partitioned Tables

```
SQL> update hybrid_bowie set total_sales=42 where id=424242;
update hybrid_bowie set total_sales=42 where id=424242
*
ERROR at line 1:
ORA-14466: Data in a read-only partition or subpartition cannot be modified.
```

NEW IN 19c

Hybrid Partitioned Tables

Start with a Non-Hybrid Partitioned Table:

```
SQL> CREATE TABLE hybrid_bowie2 (id number, album_id number, country_id number, release_date date,
                                 total_sales number)
          PARTITION BY RANGE (release_date)
         (PARTITION ALBUMS_OLD  VALUES LESS THAN (TO_DATE('01-JAN-2010', 'DD-MON-YYYY')));

Table created.

SQL> select hybrid from user_tables where table_name='HYBRID_BOWIE2';

HYBRID
------
NO
```

NEW IN 19c

Hybrid Partitioned Tables

```
SQL> ALTER TABLE hybrid_bowie2
     ADD EXTERNAL PARTITION ATTRIBUTES
    (TYPE ORACLE_LOADER
     DEFAULT DIRECTORY old_bowie_data
     ACCESS PARAMETERS (FIELDS TERMINATED BY ',' (id, album_id, country_id, release_date DATE 'dd-mm-yyyy', total_sales)));

Table altered.

SQL> ALTER TABLE hybrid_bowie2
     ADD PARTITION albums_2010 VALUES LESS THAN (TO_DATE('01-JAN-2011','DD-MON-YYYY')) EXTERNAL LOCATION ('bowie2010_data.txt');

Table altered.

SQL> ALTER TABLE hybrid_bowie2
     ADD PARTITION albums_2011 VALUES LESS THAN (TO_DATE('01-JAN-2012','DD-MON-YYYY')) EXTERNAL LOCATION ('bowie2011_data.txt');

Table altered.

SQL> ALTER TABLE hybrid_bowie2
     ADD PARTITION albums_2012 VALUES LESS THAN (TO_DATE('01-JAN-2013','DD-MON-YYYY')) EXTERNAL LOCATION ('bowie2012_data.txt');

Table altered.

SQL> ALTER TABLE hybrid_bowie2
     ADD PARTITION albums_2013 VALUES LESS THAN (TO_DATE('01-JAN-2014','DD-MON-YYYY')) EXTERNAL LOCATION ('bowie2013_data.txt');

Table altered.

SQL> ALTER TABLE hybrid_bowie2
     ADD PARTITION albums_2014 VALUES LESS THAN (TO_DATE('01-JAN-2015','DD-MON-YYYY')) EXTERNAL LOCATION ('bowie2014_data.txt');

Table altered.

SQL> select hybrid from user_tables where table_name='HYBRID_BOWIE2';

HYBRID
------
YES
```

NEW IN 19c

Memoptimized Rowstore: Fast Ingest

Session One

```
SQL> create table bowie2 (id number constraint bowie2_pk primary key, code number,  name varchar2(42))
     segment creation immediate memoptimize for write;

Table created.

SQL> select * from v$memoptimize_write_area;

TOTAL_SIZE USED_SPACE FREE_SPACE NUM_WRITES NUM_WRITERS     CON_ID
---------- ---------- ---------- ---------- ----------- ----------
         0          0          0          0           0          0

SQL> insert  /*+ MEMOPTIMIZE_WRITE */ into bowie2 values (1, 42, 'DAVID BOWIE');

1 row created.
```

Session Two

```
SQL> select * from bowie2;

no rows selected

SQL> select * from v$memoptimize_write_area;

TOTAL_SIZE USED_SPACE FREE_SPACE NUM_WRITES NUM_WRITERS     CON_ID
---------- ---------- ---------- ---------- ----------- ----------
 943718400    1120496  942597904          0           1          0
```

NEW IN 19c

Memoptimized Rowstore: Fast Ingest

Session One

```
SQL> rollback;

Rollback complete.
```

Session Two

```
SQL> select * from bowie2;

        ID       CODE NAME
---------- ---------- ----------------------------------------
         1         42 DAVID BOWIE
```

Session Two

```
SQL> insert  /*+ MEMOPTIMIZE_WRITE */ into bowie2 values (2, 42, 'DAVID BOWIE');

1 row created.

SQL> commit;

Commit complete.
```

NEW IN 19c

Memoptimized Rowstore: Fast Ingest

```
SQL> select * from bowie2 where id=2;

no rows selected

Execution Plan
----------------------------------------------------------------------------------------------
| Id  | Operation                    | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |           |     1 |    49 |     2   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| BOWIE2    |     1 |    49 |     2   (0)| 00:00:01 |
|*  2 |   INDEX UNIQUE SCAN          | BOWIE2_PK |     1 |       |     1   (0)| 00:00:01 | <= Index updated only when
----------------------------------------------------------------------------------------------    data written to disk

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("ID"=2)

Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
          1  consistent gets
          0  physical reads
          0  redo size
        567  bytes sent via SQL*Net to client
        382  bytes received via SQL*Net from client
          1  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          0  rows processed
```

NEW IN 19c

Memoptimized Rowstore: Fast Ingest

dbms_memoptimize.write_end - flushes all fast ingest data from large pool to disk for current session – data not visible until on disk:

```
SQL> insert  /*+ MEMOPTIMIZE_WRITE */ into bowie2 values (3, 42, 'ZIGGY STARDUST');

1 row created.

SQL> select * from bowie2;

        ID       CODE NAME
---------- ---------- ----------------------------------------
         1         42 DAVID BOWIE
         2         42 DAVID BOWIE

SQL> exec dbms_memoptimize.write_end;

PL/SQL procedure successfully completed.

SQL> select * from bowie2;

        ID       CODE NAME
---------- ---------- ----------------------------------------
         3         42 ZIGGY STARDUST
         1         42 DAVID BOWIE
         2         42 DAVID BOWIE
```

NEW IN 19c

Memoptimized Rowstore: Fast Ingest

```
SQL> create table bowie (id number constraint bowie_pk primary key, code number, name varchar2(42))
     segment creation immediate;

Table created.

SQL> begin
  2    for i in 1..1000000 loop
  3       insert into bowie values (i, 42, 'David Bowie');
  4       commit;
  5    end loop;
  6  end;
  7  /

PL/SQL procedure successfully completed.

Elapsed: 00:02:13.35

SQL> create table bowie2 (id number constraint bowie2_pk primary key, code number, name varchar2(42))
     segment creation immediate memoptimize for write;

Table created.

SQL> begin
  2    for i in 1..1000000 loop
  3       insert /*+ MEMOPTIMIZE_WRITE */ into bowie2 values (i, 42, 'David Bowie');
  4       commit;
  5    end loop;
  6  end;
  7  /

PL/SQL procedure successfully completed.

Elapsed: 00:00:20.57
```

NEW IN 19c

SQL Quarantine

```
SQL> create table problem_bowie (id number, code number, name varchar2(42));

Table created.

SQL> insert into problem_bowie select rownum, mod(rownum,100)+1, 'DAVID BOWIE' from dual connect by level <= 2000000;

2000000 rows created.

SQL> commit;

Commit complete.

SQL> exec dbms_stats.gather_table_stats(ownname=>null, tabname=>'PROBLEM_BOWIE');

PL/SQL procedure successfully completed.


SQL> select a.* from problem_bowie a, problem_bowie b;  <= Takes a long long time !!


SQL> select sql_id from v$sql where sql_text = 'select a.* from problem_bowie a, problem_bowie b';

SQL_ID
-------------
awg3wwn44jf3f
```

NEW IN 19c

SQL Quarantine

```
SQL> begin
   dbms_resource_manager.create_pending_area;
   dbms_resource_manager.create_consumer_group(CONSUMER_GROUP=>'BOWIE_LIMIT_TIME');
   dbms_resource_manager.set_consumer_group_mapping(attribute => 'ORACLE_USER', value => 'BOWIE', consumer_group =>'BOWIE_LIMIT_TIME');
   dbms_resource_manager.create_plan(PLAN=> 'LIMIT_TIME');
   dbms_resource_manager.create_plan_directive(PLAN=>'LIMIT_TIME', GROUP_OR_SUBPLAN=>'BOWIE_LIMIT_TIME', SWITCH_GROUP=>'CANCEL_SQL', SWITCH_TIME=>30);
   dbms_resource_manager.create_plan_directive(PLAN=>'LIMIT_TIME', GROUP_OR_SUBPLAN=>'OTHER_GROUPS', CPU_P1=>100);
   dbms_resource_manager.validate_pending_area;
   dbms_resource_manager.submit_pending_area;
end;
/

PL/SQL procedure successfully completed.

SQL> begin
  dbms_resource_manager.create_pending_area;
  dbms_resource_manager_privs.grant_switch_consumer_group('BOWIE','BOWIE_LIMIT_TIME',false);
  dbms_resource_manager.set_initial_consumer_group('BOWIE','BOWIE_LIMIT_TIME');
  dbms_resource_manager.validate_pending_area;
  dbms_resource_manager.submit_pending_area;
end;
/

PL/SQL procedure successfully completed.

SQL> begin
   dbms_resource_manager.create_pending_area;
   dbms_resource_manager.update_plan_directive(plan=>'LIMIT_TIME', group_or_subplan=>'BOWIE_LIMIT_TIME',new_switch_elapsed_time=>60, new_switch_for_call=>TRUE,
   new_switch_group=>'CANCEL_SQL' );
   dbms_resource_manager.validate_pending_area;
   dbms_resource_manager.submit_pending_area;
END;
/

PL/SQL procedure successfully completed.

SQL> alter system set resource_manager_plan='LIMIT_TIME';

System altered.
```

NEW IN 19c

SQL Quarantine

```
SQL> select DBMS_SQLQ.CREATE_QUARANTINE_BY_SQL_ID(SQL_ID => 'awg3wwn44jf3f', PLAN_HASH_VALUE => null) from dual;

DBMS_SQLQ.CREATE_QUARANTINE_BY_SQL_ID(SQL_ID=>'AWG3WWN44JF3F',PLAN_HASH_VALUE=>N
--------------------------------------------------------------------------------
SQL_QUARANTINE_1tcgjr28h8qd1483de993

SQL> exec DBMS_SQLQ.ALTER_QUARANTINE(QUARANTINE_NAME=> 'SQL_QUARANTINE_1tcgjr28h8qd1483de993', PARAMETER_NAME=> 'CPU_TIME',
PARAMETER_VALUE=> '10');

PL/SQL procedure successfully completed.

SQL> exec DBMS_SQLQ.ALTER_QUARANTINE(QUARANTINE_NAME=> 'SQL_QUARANTINE_1tcgjr28h8qd1483de993', PARAMETER_NAME=>
'ELAPSED_TIME', PARAMETER_VALUE=> '42');

PL/SQL procedure successfully completed.

SQL> select a.* from problem_bowie a, problem_bowie b;
select a.* from problem_bowie a, problem_bowie b
               *
ERROR at line 1:
ORA-56955: quarantined plan used

SQL> select sql_id, sql_quarantine, avoided_executions from v$sql where sql_id='awg3wwn44jf3f';

SQL_ID        SQL_QUARANTINE                          AVOIDED_EXECUTIONS
------------- --------------------------------------- ------------------
awg3wwn44jf3f SQL_QUARANTINE_1tcgjr28h8qd1483de993                     2
```

NEW IN 19c

Automatic Indexing Overview

- THE MOST EXCITING NEW DATABASE FEATURE IN A LONG LONG TIME !!
- An “expert system” that continuously monitors the database for the requirement of any new index
- Implemented as a “black box”, with minimal configurations but its workings auditable as necessary via reports
- The DBA is not required to actively interact with the auto index system once implemented, although some understanding of the auto indexing process is useful
- The aim is for indexes to only be created when they're necessary and validated to improve performance somewhere
- Additionally, aim also that any potential negative impact elsewhere in the database is to be detected and avoided as necessary
- Manually detecting and implementing missing indexes is labour intensive, prone to over indexing and take hours/days/weeks/months/never to implement
- Imagine a scenario where missing indexes are simply automatically created as necessary…

NEW IN 19c

Automatic Indexing Availability

- Available from Oracle Database 19c (so currently new at Version 1 status)
- However, only available:
  - Exadata
  - Oracle Autonomous Database Cloud Environments (now part of “always free tier”)

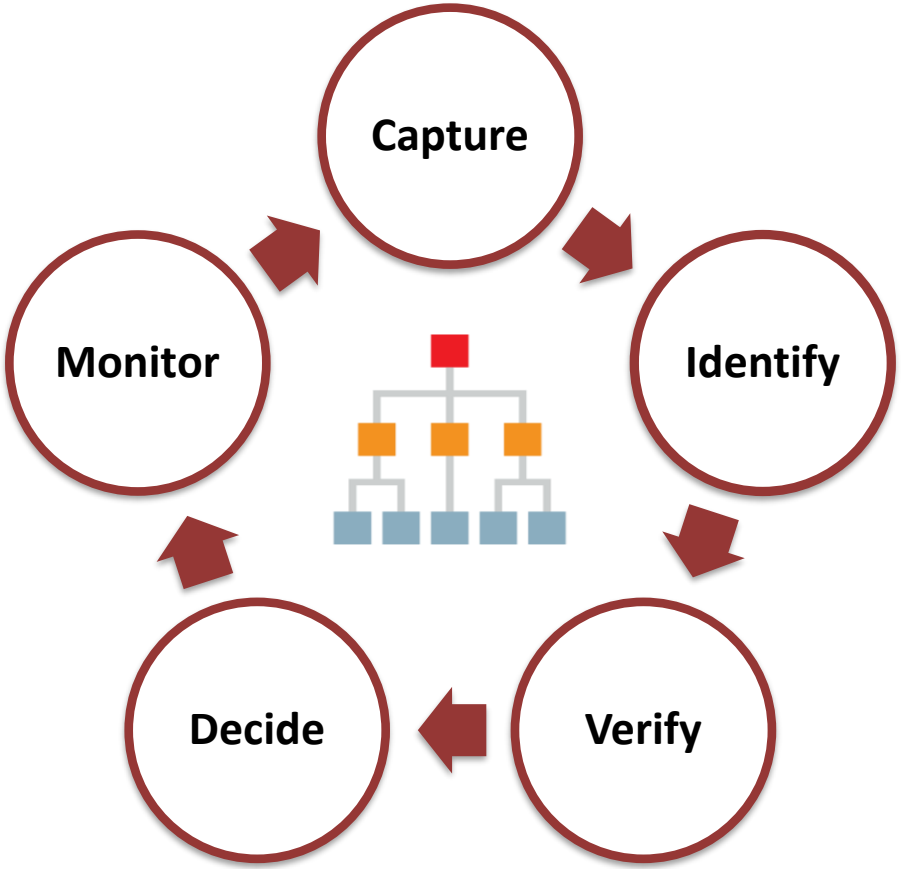
```
SQL> EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_MODE','IMPLEMENT');
BEGIN DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_MODE','IMPLEMENT'); END;

*
ERROR at line 1:
ORA-40216: feature not supported
ORA-06512: at "SYS.DBMS_SYS_ERROR", line 79
ORA-06512: at "SYS.DBMS_AUTO_INDEX_INTERNAL", line 9180
ORA-06512: at "SYS.DBMS_AUTO_INDEX", line 283
ORA-06512: at line 1
```

Note: Following demos run on Oracle Database 19.3 on BOTH Oracle Dedicated ATP Autonomous Database Cloud service AND Windows sandbox environments and subject to change…

NEW IN 19c

Automatic Index Methodology



NEW IN 19c

Capture

- Periodically capture application SQL history into a SQL repository
- Includes SQL, plans, bind values, execution statistics, etc.

NEW IN 19c

Identify

- Identify candidate indexes that may benefit the newly captured SQL statements.
- Creates index candidates as *unusable*, *invisible* indexes, ignored by the CBO and not maintained by DML statements.
- Auto index candidates are identified based on the usage of table columns by SQL statements in equality predicates only.
- Automatic Indexing Task runs every 15 minutes for up to an hour (although this can be longer at times).
- Resource Manager restricts auto indexing task to 1 CPU.

NEW IN 19c

Verify

- The *unusable* auto indexes are validated against the SQL statements by compiling the statements using hard parse to check if the optimizer will consider using them.
- SQL Performance Analyzer (SPA) compiles the SQL Tuning Set (STS) containing the past executions of these statements and checks whether the indexes are used in their execution plans.
- Auto indexes considered by the CBO are rebuilt as *valid*, *invisible* indexes that are then tested to determine if they improve SQL performance.
- Importantly, the verify session uses Dynamic Statistics at Level 11 to detect possible data skew, inaccurate or incomplete statistics.

NEW IN 19c

Decide

- If performance is better for all associated statements, auto indexes marked *visible*/*valid*
- If performance is worse for all associated statements, auto indexes configured as *invisible*/*unusable* and SQL statements can effectively be "blacklisted"
- If performance is better for some and worse for others, intent is for index to be marked as *visible/valid,* with baselines created as necessary to prevent SQL regressions, however:
  - If performance is better for some but overall worse for others, auto indexes can currently be marked as *invisible/unusable* with SQL statements not able to benefit from new auto indexes.
  - If performance is worse for some but overall better for others, auto indexes can currently be created and could potentially be used temporarily by those SQL statements that regress.

NEW IN 19c

Monitor

- Index usage is continuously monitored
- Automatically (and/or manually) created indexes that have not been used in a specified period time will be dropped

Note: The period for retaining unused auto indexes and related auto indexing data in the database can be configured using the CONFIGURE procedure of the DBMS_AUTO_INDEX package.

NEW IN 19c

DBMS_AUTO_INDEX Package

Use the **DBMS_AUTO_INDEX** package subprograms to:

- Specify configuration settings for automatic indexing, such as enabling auto indexing, schemas to exclude from using auto indexes, duration for retaining unused indexes in a database and tablespace to use for storing auto indexes.
- Drop existing (manually created) secondary indexes.
- Report the details of executed automatic indexing tasks.

DBMS_AUTO_INDEX package subprograms:

- CONFIGURE
- DROP_SECONDARY_INDEXES
- REPORT_ACTIVITY
- REPORT_LAST_ACTIVITY

NEW IN 19c

AUTO_INDEX_MODE Parameter

Automatic indexing is disabled by default in an Oracle database.

```
SQL> EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_MODE','IMPLEMENT');

PL/SQL procedure successfully completed. <= visible auto indexes created

SQL> EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_MODE','REPORT ONLY');

PL/SQL procedure successfully completed. <= auto indexes created, tested and left invisible

SQL> EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_MODE','OFF');

PL/SQL procedure successfully completed. <= disables creation of new auto indexes
```

In REPORT ONLY mode, Auto Indexes are created, tested but left in an *INVISIBLE* state, while a report shows the impact without affecting the application.

NEW IN 19c

DBA_AUTO_INDEX_CONFIG

Displays the current configuration parameter settings for automatic indexing:

```
SQL> select * from dba_auto_index_config;

PARAMETER_NAME                          PARAMETER_VALUE      LAST_MODIFIED                              MODIFIED_BY
--------------------------------------- -------------------- ------------------------------------------ ---------------
AUTO_INDEX_COMPRESSION                  OFF
AUTO_INDEX_DEFAULT_TABLESPACE
AUTO_INDEX_MODE                         REPORT ONLY          03-JUL-19 05.43.28.000000 AM               BOWIE
AUTO_INDEX_REPORT_RETENTION             31
AUTO_INDEX_RETENTION_FOR_AUTO           42                   20-JUN-19 06.32.06.000000 AM               BOWIE
AUTO_INDEX_RETENTION_FOR_MANUAL                              02-JUL-19 12.12.21.000000 AM               BOWIE
AUTO_INDEX_SCHEMA                       schema IN (BOWIE)    20-JUN-19 06.27.26.000000 AM               BOWIE
AUTO_INDEX_SPACE_BUDGET                 50
```

Setting index compression for automatic indexing currently undocumented. Discussed later.

NEW IN 19c

REPORT_ACTIVITY Function

Returns a report of the automatic indexing operations executed during a specific period in a database. The following details the last 6 hours of activity, in text format, for all possible report sections for all levels of detail:

```
SQL> select dbms_auto_index.report_activity(sysdate-0.25, sysdate, 'text', 'ALL', 'ALL')
     from dual;
```

Default values:

Last 24 hour period, in text format, including all available reports sections at a typical level of detail.

NEW IN 19c

REPORT_LAST_ACTIVITY Function

Returns a report of the last automatic indexing operation executed in a database.

```
SQL> select dbms_auto_index.report_last_activity() from dual;
```

```
SQL> select dbms_auto_index.report_last_activity(type=> 'html', section=> 'ALL -ERRORS',
level=> 'ALL') from dual;
```

NEW IN 19c

Auto Indexing: Missing Index

```
SQL> create table major_tom (id number, code1 number, code2 number, code3 number, name varchar2(42));

Table created.

SQL> insert into major_tom select rownum, mod(rownum, 10)+1, ceil(dbms_random.value(0, 100)), ceil(dbms_random.value(0, 1000)),
'David Bowie' from dual connect by level <= 10000000;

10000000 rows created.

SQL> commit;

Commit complete.

SQL> exec dbms_stats.gather_table_stats(ownname=>null, tabname=>'MAJOR_TOM');

PL/SQL procedure successfully completed.

SQL> select column_name, num_distinct, density from user_tab_columns where table_name='MAJOR_TOM';

COLUMN_NAME          NUM_DISTINCT    DENSITY
-------------------- ------------ ----------
ID                        9914368 1.0086E-07
CODE1                          10  .00000005
CODE2                         100  .00000005
CODE3                        1000       .001
NAME                            1          1
```

NEW IN 19c

Auto Indexing: Missing Index

```
SQL> select * from major_tom where code3=42 and code2=42 and code1=4;

15 rows selected.

Execution Plan
----------------------------------------------------------------------------------------------
| Id  | Operation                     | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |           |    10 |   280 |  7354   (7)| 00:00:01 |
|   1 |  PX COORDINATOR               |           |       |       |            |          |
|   2 |   PX SEND QC (RANDOM)         | :TQ10000  |    10 |   280 |  7354   (7)| 00:00:01 |
|   3 |    PX BLOCK ITERATOR          |           |    10 |   280 |  7354   (7)| 00:00:01 |
|*  4 |     TABLE ACCESS STORAGE FULL| MAJOR_TOM |    10 |   280 |  7354   (7)| 00:00:01 |
----------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - storage("CODE3"=42 AND "CODE2"=42 AND "CODE1"=4)
       filter("CODE3"=42 AND "CODE2"=42 AND "CODE1"=4)

Statistics
----------------------------------------------------------
         34 recursive calls
          5 db block gets
      45861 consistent gets
          0 physical reads
       1044 redo size
       1087 bytes sent via SQL*Net to client
        588 bytes received via SQL*Net from client
          2 SQL*Net roundtrips to/from client
          0 sorts (memory)
          0 sorts (disk)
         15 rows processed
```

NEW IN 19c

Auto Indexing: Missing Index

```
SQL> select dbms_auto_index.report_last_activity() report from dual;  <= After approx. 15 minutes

REPORT
------------------------------------------------------------------------------

GENERAL INFORMATION
------------------------------------------------------------------------------
 Activity start               : 21-JUN-2019 02:39:32
 Activity end                 : 21-JUN-2019 02:40:18
 Executions completed         : 1
 Executions interrupted       : 0
 Executions with fatal error  : 0
------------------------------------------------------------------------------

SUMMARY (AUTO INDEXES)
------------------------------------------------------------------------------
 Index candidates                                 : 1
 Indexes created (visible / invisible)            : 1 (1 / 0)
 Space used (visible / invisible)                 : 243.27 MB (243.27 MB / 0 B)
 Indexes dropped                                  : 0
 SQL statements verified                          : 2
 SQL statements improved (improvement factor)     : 1 (45853.8x)
 SQL plan baselines created                       : 0
 Overall improvement factor                       : 8.2x
------------------------------------------------------------------------------

SUMMARY (MANUAL INDEXES)
------------------------------------------------------------------------------
 Unused indexes    : 0
 Space used        : 0 B
 Unusable indexes  : 0
------------------------------------------------------------------------------
```

NEW IN 19c

Auto Indexing: Missing Index

```
INDEX DETAILS
-------------------------------------------------------------------------------
1. The following indexes were created:
-------------------------------------------------------------------------------------------
| Owner | Table     | Index                | Key               | Type   | Properties |
-------------------------------------------------------------------------------------------
| BOWIE | MAJOR_TOM | SYS_AI_9mrs058nrg9d5 | CODE1,CODE2,CODE3 | B-TREE | NONE       |
-------------------------------------------------------------------------------------------

VERIFICATION DETAILS
-------------------------------------------------------------------------------
1. The performance of the following statements improved:
-------------------------------------------------------------------------------
 Parsing Schema Name  : BOWIE
 SQL ID               : ayuj12hggwrvc
 SQL Text             : select * from major_tom where code3=42 and code2=42 and code1=4
 Improvement Factor   : 45853.8x  (596102 / 13 = 45854 ??)

Execution Statistics:
-----------------------------
                    Original Plan                Auto Index Plan
                    -----------------------------  -----------------------------
 Elapsed Time (s):  3103394                      946     <= These times are really in Microseconds (1 millionth of a second)
 CPU Time (s):      3092860                      1017    <= These times are really in Microseconds (1 millionth of a second)
 Buffer Gets:       596102                       18
 Optimizer Cost:    7354                         18
 Disk Reads:        0                            2
 Direct Writes:     0                            0
 Rows Processed:    195                          15
 Executions:        13                           1
```

NEW IN 19c

Auto Indexing: Missing Index

```
SQL> select * from major_tom where code3=42 and code2=42 and code1=4;

Execution Plan
----------------------------------------------------------------------------------------------------------
| Id  | Operation                             | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                      |                     |    10 |   280 |    13   (0)| 00:00:01 |
|   1 |  PX COORDINATOR                       |                     |       |       |            |          |
|   2 |   PX SEND QC (RANDOM)                 | :TQ10001            |    10 |   280 |    13   (0)| 00:00:01 |
|   3 |    TABLE ACCESS BY INDEX ROWID BATCHED| MAJOR_TOM           |    10 |   280 |    13   (0)| 00:00:01 |
|   4 |     BUFFER SORT                       |                     |       |       |            |          |
|   5 |      PX RECEIVE                       |                     |    10 |       |     3   (0)| 00:00:01 |
|   6 |       PX SEND HASH (BLOCK ADDRESS)    | :TQ10000            |    10 |       |     3   (0)| 00:00:01 |
|   7 |        PX SELECTOR                    |                     |       |       |            |          |
|*  8 |         INDEX RANGE SCAN              | SYS_AI_9mrs058nrg9d5|    10 |       |     3   (0)| 00:00:01 |
----------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   8 - access("CODE1"=4 AND "CODE2"=42 AND "CODE3"=42)
```

NEW IN 19c

Auto Indexing: Column Re-Order

```
SQL> select * from major_tom where code3=42;  <= Can't use Auto ordered indexed based on (CODE1, CODE2, CODE3)

9961 rows selected.

Execution Plan
----------------------------------------------------------------------------------------------
| Id  | Operation                     | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |            | 10000 |   273K|  7354   (7)| 00:00:01 |
|   1 |  PX COORDINATOR               |            |       |       |            |          |
|   2 |   PX SEND QC (RANDOM)         | :TQ10000   | 10000 |   273K|  7354   (7)| 00:00:01 |
|   3 |    PX BLOCK ITERATOR          |            | 10000 |   273K|  7354   (7)| 00:00:01 |
|*  4 |     TABLE ACCESS STORAGE FULL | MAJOR_TOM  | 10000 |   273K|  7354   (7)| 00:00:01 |
----------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - storage("CODE3"=42)
       filter("CODE3"=42)

Statistics
----------------------------------------------------------
          8 recursive calls
          4 db block gets
      45853 consistent gets
          0 physical reads
          0 redo size
     166137 bytes sent via SQL*Net to client
        599 bytes received via SQL*Net from client
          3 SQL*Net roundtrips to/from client
          0 sorts (memory)
          0 sorts (disk)
       9961 rows processed
```

NEW IN 19c

Auto Indexing: Column Re-Order

```
SQL> select * from major_tom where code3=42 and code2=42; <= Again, not using auto ordered indexed based on (CODE1, CODE2, CODE3)

101 rows selected.

Execution Plan
----------------------------------------------------------------------------------------------
| Id  | Operation                     | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |            |   100 |  2800 |  7354   (7)| 00:00:01 |
|   1 |  PX COORDINATOR               |            |       |       |            |          |
|   2 |   PX SEND QC (RANDOM)         | :TQ10000   |   100 |  2800 |  7354   (7)| 00:00:01 |
|   3 |    PX BLOCK ITERATOR          |            |   100 |  2800 |  7354   (7)| 00:00:01 |
|*  4 |     TABLE ACCESS STORAGE FULL | MAJOR_TOM  |   100 |  2800 |  7354   (7)| 00:00:01 |
----------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - storage("CODE3"=42 AND "CODE2"=42)
       filter("CODE3"=42 AND "CODE2"=42)

Statistics
----------------------------------------------------------
          6 recursive calls
          0 db block gets
      45853 consistent gets
          0 physical reads
          0 redo size
       2281 bytes sent via SQL*Net to client
        588 bytes received via SQL*Net from client
          2 SQL*Net roundtrips to/from client
          0 sorts (memory)
          0 sorts (disk)
        101 rows processed
```

NEW IN 19c

Auto Indexing: Column Re-Order

```
SQL> select dbms_auto_index.report_last_activity() report from dual;

REPORT
------------------------------------------------------------------------------
GENERAL INFORMATION
------------------------------------------------------------------------------
 Activity start               : 21-JUN-2019 02:39:32
 Activity end                 : 21-JUN-2019 02:40:18
 Executions completed         : 1
 Executions interrupted       : 0
 Executions with fatal error  : 0
------------------------------------------------------------------------------

SUMMARY (AUTO INDEXES)
------------------------------------------------------------------------------
 Index candidates                          : 1
 Indexes created (visible / invisible)     : 1 (1 / 0)
 Space used (visible / invisible)          : 243.27 MB (243.27 MB / 0 B)
 Indexes dropped (space reclaimed)         : 1 (243.27 MB)
 SQL statements verified                   : 1
 SQL statements improved                   : 0
 SQL plan baselines created                : 0
 Overall improvement factor                : 0x
------------------------------------------------------------------------------

SUMMARY (MANUAL INDEXES)
------------------------------------------------------------------------------
 Unused indexes    : 0
 Space used        : 0 B
 Unusable indexes  : 0
------------------------------------------------------------------------------
```

NEW IN 19c

Auto Indexing: Column Re-Order

```
INDEX DETAILS
-------------------------------------------------------------------------------
1. The following indexes were created:
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------
| Owner | Table     | Index                | Key               | Type   | Properties |
-------------------------------------------------------------------------------------------------
| BOWIE | MAJOR_TOM | SYS_AI_00hxxxkgb821n | CODE3,CODE2,CODE1 | B-TREE | NONE       |  <= This index can now service all known SQL
-------------------------------------------------------------------------------------------------

2. The following indexes were dropped:
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------
| Owner | Table     | Index                | Key               | Type   | Properties |
-------------------------------------------------------------------------------------------------
| BOWIE | MAJOR_TOM | SYS_AI_9mrs058nrg9d5 | CODE1,CODE2,CODE3 | B-TREE | NONE       |  <= Meanwhile the previous index is dropped
-------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------


SQL> select index_name, column_name, column_position from user_ind_columns where table_name='MAJOR_TOM' order by column_position;

INDEX_NAME           COLUMN_NAME     COLUMN_POSITION
-------------------- --------------- ---------------
SYS_AI_00hxxxkgb821n CODE3                         1
SYS_AI_00hxxxkgb821n CODE2                         2
SYS_AI_00hxxxkgb821n CODE1                         3
```

NEW IN 19c

Auto Indexing: Second Index Requirement

```
SQL> select * from major_tom where code2=42 and code1=42; <= This new combination can not use index based on (CODE3, CODE2, CODE1)

no rows selected

Execution Plan
----------------------------------------------------------------------------------------------
| Id  | Operation                    | Name       | Rows  | Bytes | Cost (%CPU)|Time     |
----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |            |     1 |    28 |  7318   (6)|00:00:01 |
|   1 |  PX COORDINATOR              |            |       |       |            |         |
|   2 |   PX SEND QC (RANDOM)        | :TQ10000   |     1 |    28 |  7318   (6)|00:00:01 |
|   3 |    PX BLOCK ITERATOR         |            |     1 |    28 |  7318   (6)|00:00:01 |
|*  4 |     TABLE ACCESS STORAGE FULL| MAJOR_TOM  |     1 |    28 |  7318   (6)|00:00:01 |
----------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - storage("CODE1"=42 AND "CODE2"=42)
       filter("CODE1"=42 AND "CODE2"=42)

Statistics
----------------------------------------------------------
          8 recursive calls
          4 db block gets
      45853 consistent gets
          0 physical reads
          0 redo size
        645 bytes sent via SQL*Net to client
        577 bytes received via SQL*Net from client
          1 SQL*Net roundtrips to/from client
          0 sorts (memory)
          0 sorts (disk)
          0 rows processed
```

NEW IN 19c

Auto Indexing: Second Index Requirement

```
SQL> select dbms_auto_index.report_last_activity() report from dual;

REPORT
------------------------------------------------------------------------------
GENERAL INFORMATION
------------------------------------------------------------------------------
 Activity start               : 21-JUN-2019 07:41:55
 Activity end                 : 21-JUN-2019 07:42:49
 Executions completed         : 1
 Executions interrupted       : 0
 Executions with fatal error  : 0
------------------------------------------------------------------------------

SUMMARY (AUTO INDEXES)
------------------------------------------------------------------------------
 Index candidates                                : 1
 Indexes created (visible / invisible)           : 1 (1 / 0)
 Space used (visible / invisible)                : 201.33 MB (201.33 MB / 0 B)
 Indexes dropped                                 : 0
 SQL statements verified                         : 1
 SQL statements improved (improvement factor)    : 1 (5.1x)
 SQL plan baselines created                      : 0
 Overall improvement factor                      : 5.1x
------------------------------------------------------------------------------

SUMMARY (MANUAL INDEXES)
------------------------------------------------------------------------------
 Unused indexes    : 0
 Space used        : 0 B
 Unusable indexes  : 0
------------------------------------------------------------------------------
```

NEW IN 19c

Auto Indexing: Second Index Requirement

```
INDEX DETAILS
-------------------------------------------------------------------------------
1. The following indexes were created:
-------------------------------------------------------------------------------
| Owner | Table     | Index                | Key         | Type   | Properties |
-------------------------------------------------------------------------------
| BOWIE | MAJOR_TOM | SYS_AI_gmhdbjx21zcr1 | CODE1,CODE2 | B-TREE | NONE       |
-------------------------------------------------------------------------------

VERIFICATION DETAILS
-------------------------------------------------------------------------------
1. The performance of the following statements improved:
-------------------------------------------------------------------------------
 Parsing Schema Name  : BOWIE
 SQL ID               : 3y246nzwdpybv
 SQL Text             : select * from major_tom where code2=42 and code1=4
 Improvement Factor   : 5.1x


SQL> select index_name, column_name, column_position from user_ind_columns where table_name='MAJOR_TOM'
     order by index_name, column_position;

INDEX_NAME                COLUMN_NAME          COLUMN_POSITION
------------------------- -------------------- ---------------
SYS_AI_00hxxxkgb821n      CODE3                              1
SYS_AI_00hxxxkgb821n      CODE2                              2
SYS_AI_00hxxxkgb821n      CODE1                              3
SYS_AI_gmhdbjx21zcr1      CODE1                              1    <= Need a second index to cater for all the combination of
SYS_AI_gmhdbjx21zcr1      CODE2                              2       SQL predicates
```

NEW IN 19c

Auto Indexing: Index Created, But…

```
SQL> create table major_tom3 (id number, code1 number, code2 number, code3 number, name varchar2(42));

Table created.

SQL> insert into major_tom3 select rownum, mod(rownum, 1000)+1, ceil(dbms_random.value(0, 100)), ceil(dbms_random.value(0, 10)),
'David Bowie' from dual connect by level <= 10000000;

10000000 rows created.

SQL> commit;

Commit complete.

SQL> exec dbms_stats.gather_table_stats(ownname=>null, tabname=>'MAJOR_TOM3');

PL/SQL procedure successfully completed.


SQL> select * from major_tom3 where code3=4 and code2=42;

9968 rows selected.
```

The combination of CODE2 and CODE3 predicates results in a relatively large number of rows being returned on poorly clustered data…

NEW IN 19c

Auto Indexing: Index Created, But…

```
INDEX DETAILS
-------------------------------------------------------------------------------
1. The following indexes were created:
-------------------------------------------------------------------------------
| Owner | Table      | Index                 | Key         | Type   | Properties|
-------------------------------------------------------------------------------
| BOWIE | MAJOR_TOM3 | SYS_AI_bnyacywycxx8b  | CODE2,CODE3 | B-TREE | NONE      |
-------------------------------------------------------------------------------

VERIFICATION DETAILS
-------------------------------------------------------------------------------
1. The performance of the following statements improved:
-------------------------------------------------------------------------------
 Parsing Schema Name  : BOWIE
 SQL ID               : 1h4j53jruuzht
 SQL Text             : select * from major_tom3 where code3=4 and code2=42
 Improvement Factor   : 5.1x   <= 183493/4 = 45873    45873/8954 = 5.1x (based on Buffer Gets)

Execution Statistics:
-----------------------
                     Original Plan              Auto Index Plan
                     --------------------------  --------------------------
 Elapsed Time (s):   1276903                     29310
 CPU Time (s):       1226240                     25905
 Buffer Gets:        183493                      8954
 Optimizer Cost:     7355                        8996
 Disk Reads:         0                           26
 Direct Writes:      0                           0
 Rows Processed:     39872                       9968
 Executions:         4                           1
```

NEW IN 19c

Auto Indexing: Index Created, But…

```
PLANS SECTION
-------------------------------------------------------------------------------------------

- Original
------------------------------
 Plan Hash Value  : 2354969370

-----------------------------------------------------------------------------------------
| Id | Operation                     | Name       | Rows  | Bytes  | Cost | Time       |
-----------------------------------------------------------------------------------------
|  0 | SELECT STATEMENT              |            |       |        | 7355 |            |
|  1 | PX COORDINATOR                |            |       |        |      |            |
|  2 |  PX SEND QC (RANDOM)          | :TQ10000   | 10000 | 280000 | 7355 | 00:00:01   |
|  3 |   PX BLOCK ITERATOR           |            | 10000 | 280000 | 7355 | 00:00:01   |
|  4 |    TABLE ACCESS STORAGE FULL  | MAJOR_TOM3 | 10000 | 280000 | 7355 | 00:00:01   | <= Multiblock reads makes FTS relatively cheap
-----------------------------------------------------------------------------------------

- With Auto Indexes
------------------------------
 Plan Hash Value  : 1676847804

-----------------------------------------------------------------------------------------------------------
| Id  | Operation                             | Name                | Rows  | Bytes  | Cost | Time     |
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                      |                     | 10820 | 302960 | 8996 | 00:00:01 | <= CBO Cost is greater with
|   1 |   TABLE ACCESS BY INDEX ROWID BATCHED | MAJOR_TOM3          | 10820 | 302960 | 8996 | 00:00:01 |    auto index
| * 2 |    INDEX RANGE SCAN                   | SYS_AI_bnyacywycxx8b |  9968 |        |   27 | 00:00:01 |
-----------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------
* 2 - access("CODE2"=42 AND "CODE3"=4)
```

NEW IN 19c

Auto Indexing: Index Created, But…

```
SQL> select index_name, auto, constraint_index, visibility, compression, status from user_indexes
     where table_name='MAJOR_TOM3';

INDEX_NAME           AUT CON VISIBILIT COMPRESSION   STATUS
-------------------- --- --- --------- ------------- --------
SYS_AI_bnyacywycxx8b YES NO  VISIBLE   DISABLED      VALID


SQL> select index_name, column_name, column_position from user_ind_columns where table_name='MAJOR_TOM3'
     order by index_name, column_position;

INDEX_NAME           COLUMN_NAME          COLUMN_POSITION
-------------------- -------------------- ---------------
SYS_AI_bnyacywycxx8b CODE2                              1
SYS_AI_bnyacywycxx8b CODE3                              2
```

NEW IN 19c

Auto Indexing: Index Created But Not Used

```
SQL> select * from major_tom3 where code3=4 and code2=42;

9968 rows selected.

Execution Plan
----------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |            | 10045 |   274K|  7355   (7)| 00:00:01 |
|   1 |  PX COORDINATOR               |            |       |       |            |          |
|   2 |   PX SEND QC (RANDOM)         | :TQ10000   | 10045 |   274K|  7355   (7)| 00:00:01 |
|   3 |    PX BLOCK ITERATOR          |            | 10045 |   274K|  7355   (7)| 00:00:01 |
|*  4 |     TABLE ACCESS STORAGE FULL| MAJOR_TOM3 | 10045 |   274K|  7355   (7)| 00:00:01 | <= Auto Index not used despite being created for this SQL
----------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - storage("CODE2"=42 AND "CODE3"=4)
       filter("CODE2"=42 AND "CODE3"=4)

Statistics
----------------------------------------------------------
         11 recursive calls
          4 db block gets
      45859 consistent gets
          0 physical reads
          0 redo size
     275084 bytes sent via SQL*Net to client
       7892 bytes received via SQL*Net from client
        666 SQL*Net roundtrips to/from client
          0 sorts (memory)
          0 sorts (disk)
       9968 rows processed
```

NEW IN 19c

Auto Indexing: Index Created But Not Used

```
SQL> select /*+ index(major_tom3) */ * from major_tom3 where code3=4 and code2=42;

9968 rows selected.

Execution Plan
----------------------------------------------------------------------------------------------------------------------
| Id  | Operation                             | Name                 | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                      |                      | 10045 |  274K |  9065   (1)| 00:00:01 | <= CBO still costs index more than FTS
|   1 |  PX COORDINATOR                       |                      |       |       |            |          |
|   2 |   PX SEND QC (RANDOM)                 | :TQ10001             | 10045 |  274K |  9065   (1)| 00:00:01 |
|   3 |    TABLE ACCESS BY INDEX ROWID BATCHED| MAJOR_TOM3           | 10045 |  274K |  9065   (1)| 00:00:01 |
|   4 |     BUFFER SORT                       |                      |       |       |            |          |
|   5 |      PX RECEIVE                       |                      | 10045 |       |    27   (4)| 00:00:01 |
|   6 |       PX SEND HASH (BLOCK ADDRESS)    | :TQ10000             | 10045 |       |    27   (4)| 00:00:01 |
|   7 |        PX SELECTOR                    |                      |       |       |            |          |
|*  8 |         INDEX RANGE SCAN              | SYS_AI_bnyacywycxx8b | 10045 |       |    27   (4)| 00:00:01 |
----------------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   8 - access("CODE2"=42 AND "CODE3"=4)

Statistics
----------------------------------------------------------
         16 recursive calls
          4 db block gets
       8958 consistent gets   <= Fewer consistent gets, down from 45859
          0 physical reads
          0 redo size
     275084 bytes sent via SQL*Net to client
       7892 bytes received via SQL*Net from client
        666 SQL*Net roundtrips to/from client
          2 sorts (memory)
          0 sorts (disk)
       9968 rows processed
```

Auto Indexing: Add Column(s) to Existing Index

```
SQL> select * from major_tom3 where code1=42 and code3=4 and code2=42; <= New predicate makes the SQL much more selective

10 rows selected.

Execution Plan
----------------------------------------------------------------------------------------------
| Id  | Operation                    | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |            |    10 |   280 |  7354   (7)| 00:00:01 |
|   1 |  PX COORDINATOR              |            |       |       |            |          |
|   2 |   PX SEND QC (RANDOM)        | :TQ10000   |    10 |   280 |  7354   (7)| 00:00:01 |
|   3 |    PX BLOCK ITERATOR         |            |    10 |   280 |  7354   (7)| 00:00:01 |
|*  4 |     TABLE ACCESS STORAGE FULL| MAJOR_TOM3 |    10 |   280 |  7354   (7)| 00:00:01 | <= FTS as existing auto index is deemed too expensive
----------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - storage("CODE1"=42 AND "CODE2"=42 AND "CODE3"=4)
       filter("CODE1"=42 AND "CODE2"=42 AND "CODE3"=4)

Statistics
----------------------------------------------------------
          6 recursive calls
          0 db block gets
      45853 consistent gets
          0 physical reads
          0 redo size
       1032 bytes sent via SQL*Net to client
        588 bytes received via SQL*Net from client
          2 SQL*Net roundtrips to/from client
          0 sorts (memory)
          0 sorts (disk)
         10 rows processed
```

Auto Indexing: Add Column(s) to Existing Index

```
INDEX DETAILS
-------------------------------------------------------------------------------
1. The following indexes were created:
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------
| Owner | Table      | Index                | Key               | Type   | Properties |
-------------------------------------------------------------------------------------------------
| BOWIE | MAJOR_TOM3 | SYS_AI_cy8rs2dqb0nrp | CODE2,CODE3,CODE1 | B-TREE | NONE       |  <= New index created with the additional column CODE1
-------------------------------------------------------------------------------------------------

2. The following indexes were dropped:
-------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
| Owner | Table      | Index                | Key         | Type   | Properties|
-----------------------------------------------------------------------------------
| BOWIE | MAJOR_TOM3 | SYS_AI_bnyacywycxx8b | CODE2,CODE3 | B-TREE | NONE      |        <= Previous index is now redundant and dropped
-----------------------------------------------------------------------------------
-------------------------------------------------------------------------------

VERIFICATION DETAILS
-------------------------------------------------------------------------------
1. The performance of the following statements improved:
-------------------------------------------------------------------------------
 Parsing Schema Name  : BOWIE

 SQL ID               : 1hv3d685x2cy4

 SQL Text             : select * from major_tom3 where code1=43 and code3=4 and code2=42

 Improvement Factor   : 45853.6x
```

Auto Indexing: Add Column(s) to Existing Index

```
SQL> select index_name, auto, constraint_index, visibility, status compression from user_indexes where table_name='MAJOR_TOM3';

INDEX_NAME           AUT CON VISIBILIT COMPRESSION   STATUS
-------------------- --- --- --------- ------------- --------
SYS_AI_cy8rs2dqb0nrp YES NO  VISIBLE   DISABLED      VALID


SQL> select index_name, column_name, column_position from user_ind_columns where table_name='MAJOR_TOM3' order by index_name,
column_position;

INDEX_NAME           COLUMN_NAME          COLUMN_POSITION
-------------------- -------------------- ---------------
SYS_AI_cy8rs2dqb0nrp CODE2                              1
SYS_AI_cy8rs2dqb0nrp CODE3                              2
SYS_AI_cy8rs2dqb0nrp CODE1                              3
```

Previous Auto Index has been “replaced” by a new index that can cater for both types of known SQL predicate combinations, by adding CODE1 column to end of index list

Auto Indexing: Multiple Indexing Combinations

```
SQL> create table thin_white_duke (id number, code1 number, code2 number, code3 number, code4 number, name varchar2(42));

Table created.

SQL> insert into thin_white_duke select rownum, ceil(dbms_random.value(0, 100)), ceil(dbms_random.value(0, 1000)),
ceil(dbms_random.value(0, 10000)), ceil(dbms_random.value(0, 100000)), 'David Bowie' from dual connect by level <= 10000000;

10000000 rows created.

SQL> commit;

Commit complete.

SQL> exec dbms_stats.gather_table_stats(ownname=>null, tabname=>'THIN_WHITE_DUKE');

PL/SQL procedure successfully completed.

SQL> select column_name, num_distinct, density from user_tab_columns where table_name='THIN_WHITE_DUKE';

COLUMN_NAME     NUM_DISTINCT    DENSITY
--------------- ------------ ----------
ID                   9914368 1.0086E-07
CODE1                    100        .01
CODE2                   1000       .001
CODE3                  10000      .0001
CODE4                 100824 9.9183E-06
NAME                       1          1
```

Auto Indexing: Multiple Indexing Combinations

```
SQL> select * from thin_white_duke where id=42;

SQL> select * from thin_white_duke where code1=42;
SQL> select * from thin_white_duke where code2=42;
SQL> select * from thin_white_duke where code3=42;
SQL> select * from thin_white_duke where code4=42;

SQL> select * from thin_white_duke where code1=42 and code2=42;
SQL> select * from thin_white_duke where code1=42 and code3=42;
SQL> select * from thin_white_duke where code1=42 and code4=42;

SQL> select * from thin_white_duke where code2=42 and code1=42;
SQL> select * from thin_white_duke where code2=42 and code3=42;
SQL> select * from thin_white_duke where code2=42 and code4=42;

SQL> select * from thin_white_duke where code3=42 and code1=42;
SQL> select * from thin_white_duke where code3=42 and code2=42;
SQL> select * from thin_white_duke where code3=42 and code4=42;

SQL> select * from thin_white_duke where code4=42 and code1=42;
SQL> select * from thin_white_duke where code4=42 and code2=42;
SQL> select * from thin_white_duke where code4=42 and code3=42;

SQL> select * from thin_white_duke where code1=42 and code2=42 and code3=42;
SQL> select * from thin_white_duke where code1=42 and code2=42 and code4=42;
SQL> select * from thin_white_duke where code1=42 and code3=42 and code4=42;

SQL> select * from thin_white_duke where code2=42 and code1=42 and code3=42;
SQL> select * from thin_white_duke where code2=42 and code1=42 and code4=42;
SQL> select * from thin_white_duke where code2=42 and code3=42 and code4=42;

SQL> select * from thin_white_duke where code3=42 and code1=42 and code2=42;
SQL> select * from thin_white_duke where code3=42 and code1=42 and code4=42;
SQL> select * from thin_white_duke where code3=42 and code2=42 and code4=42;

SQL> select * from thin_white_duke where code4=42 and code1=42 and code2=42;
SQL> select * from thin_white_duke where code4=42 and code1=42 and code3=42;
SQL> select * from thin_white_duke where code4=42 and code2=42 and code3=42;

SQL> select * from thin_white_duke where code1=42 and code2=42 and code3=42 and code4=42;  <= 30 different column predicate combinations
```

Auto Indexing: Multiple Indexing Combinations

```
SQL> select dbms_auto_index.report_last_activity() report from dual;

REPORT
------------------------------------------------------------------------------
GENERAL INFORMATION
------------------------------------------------------------------------------
 Activity start               : 15-JUL-2019 07:46:25
 Activity end                 : 15-JUL-2019 07:48:56
 Executions completed         : 1
 Executions interrupted       : 0
 Executions with fatal error  : 0
------------------------------------------------------------------------------

SUMMARY (AUTO INDEXES)
------------------------------------------------------------------------------
 Index candidates                                : 7
 Indexes created (visible / invisible)           : 7 (7 / 0)
 Space used (visible / invisible)                : 1.8 GB (1.8 GB / 0 B)
 Indexes dropped                                 : 0
 SQL statements verified                         : 29
 SQL statements improved (improvement factor)    : 29 (147.2x)
 SQL plan baselines created                      : 0
 Overall improvement factor                      : 147.2x
------------------------------------------------------------------------------

SUMMARY (MANUAL INDEXES)
------------------------------------------------------------------------------
 Unused indexes    : 0
 Space used        : 0 B
 Unusable indexes  : 0
------------------------------------------------------------------------------
```

Auto Indexing: Multiple Indexing Combinations

```
INDEX DETAILS
-------------------------------------------------------------------------------
1. The following indexes were created:
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
| Owner | Table           | Index                | Key                       | Type   | Properties |
-------------------------------------------------------------------------------------------------------------
| BOWIE | THIN_WHITE_DUKE | SYS_AI_0u1qx8vgtstkb | CODE4,CODE1,CODE2         | B-TREE | NONE       |
| BOWIE | THIN_WHITE_DUKE | SYS_AI_2j5g09nzrhqsw | CODE2,CODE3,CODE4         | B-TREE | NONE       |
| BOWIE | THIN_WHITE_DUKE | SYS_AI_4y26dtkybxq6k | CODE3,CODE4               | B-TREE | NONE       |
| BOWIE | THIN_WHITE_DUKE | SYS_AI_5pmdyk5pjay8a | CODE3,CODE1,CODE4         | B-TREE | NONE       |
| BOWIE | THIN_WHITE_DUKE | SYS_AI_6uqhvvzabg5n8 | ID                        | B-TREE | NONE       |
| BOWIE | THIN_WHITE_DUKE | SYS_AI_bwfbc6nah6uga | CODE2,CODE4               | B-TREE | NONE       |
| BOWIE | THIN_WHITE_DUKE | SYS_AI_fftcb8q17yy6g | CODE1,CODE2,CODE3,CODE4   | B-TREE | NONE       |
-------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------
```

Only minimum of 7 new index created to cater for all 29 different SQL predicate column combinations

Many DBAs would manually create many more than 7 indexes in this scenario…

NEW IN 19c

Auto Indexing: Supported Index Types

According to the documentation, Auto Indexes have the following restrictions:

- B-tree Indexes only
- Created for non-partitioned as well as partitioned tables (but only as Local Indexes)
- Cannot be created for temporary tables

However, the are currently several other restrictions that are not documented, including:

- Auto Indexes ONLY created based on *equality* predicates (no range predicates, LIKE, etc.)
- Auto Indexes NOT created based on Min/Max scan operations
- Auto Indexes NOT created based on Foreign Key locking scenarios
- Only columns in predicates considered (no Index Overloading supported)
- Function-Based Indexes not currently supported
- JSON Indexes not currently supported

Note: Currently at V1 status, these limitations will no doubt be addressed in later versions…

NEW IN 19c

Auto Indexing: Non-Equality Predicates

```
SQL> create table ziggy (id number, code number, name varchar2(42));

Table created.

SQL> insert into ziggy select rownum, mod(rownum, 1000000)+1, 'David Bowie' from dual connect by level <= 10000000;

10000000 rows created.

SQL> commit;

Commit complete.

SQL> exec dbms_stats.gather_table_stats(ownname=>null, tabname=>'ZIGGY');

PL/SQL procedure successfully completed.

SQL> select * from ziggy where id between 42 and 50;

9 rows selected.  <= Very selective SQL, only 9 rows from a 10M row table selected...

Execution Plan
----------------------------------------------------------------------------------------
| Id  | Operation                     | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |          |    10 |   230 |  6173   (6)| 00:00:01 | <= Very accurate cardinality estimate...
|   1 |  PX COORDINATOR               |          |       |       |            |          |
|   2 |   PX SEND QC (RANDOM)         | :TQ10000 |    10 |   230 |  6173   (6)| 00:00:01 |
|   3 |    PX BLOCK ITERATOR          |          |    10 |   230 |  6173   (6)| 00:00:01 |
|*  4 |     TABLE ACCESS STORAGE FULL| ZIGGY    |    10 |   230 |  6173   (6)| 00:00:01 |
----------------------------------------------------------------------------------------
```

NEW IN 19c

Auto Indexing: Non-Equality Predicates

```
SQL> select dbms_auto_index.report_last_activity() report from dual;

REPORT
------------------------------------------------------------------------------
GENERAL INFORMATION
------------------------------------------------------------------------------
 Activity start                      : 20-JUL-2019 13:55:37
 Activity end                        : 20-JUL-2019 13:56:20
 Executions completed                : 1
 Executions interrupted              : 0
 Executions with fatal error         : 0
------------------------------------------------------------------------------

SUMMARY (AUTO INDEXES)
------------------------------------------------------------------------------
 Index candidates                    : 0
 Indexes created                     : 0
 Space used                          : 0 B
 Indexes dropped                     : 0
 SQL statements verified             : 2
 SQL statements improved             : 0
 SQL plan baselines created          : 0
 Overall improvement factor          : 0x
------------------------------------------------------------------------------

SUMMARY (MANUAL INDEXES)
------------------------------------------------------------------------------
 Unused indexes    : 0
 Space used        : 0 B
 Unusable indexes  : 0
------------------------------------------------------------------------------
```

NEW IN 19c

Auto Indexing: Non-Equality Predicates

```
SQL> select index_name, auto, constraint_index, visibility from user_indexes where table_name='ZIGGY';

no rows selected

SQL> select * from ziggy where id between 42 and 50;

9 rows selected.

Execution Plan
----------------------------------------------------------------------------------------
| Id  | Operation                     | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |          |    10 |   230 |  6173   (6)| 00:00:01 |
|   1 |  PX COORDINATOR               |          |       |       |            |          |
|   2 |   PX SEND QC (RANDOM)         | :TQ10000 |    10 |   230 |  6173   (6)| 00:00:01 |
|   3 |    PX BLOCK ITERATOR          |          |    10 |   230 |  6173   (6)| 00:00:01 |
|*  4 |     TABLE ACCESS STORAGE FULL| ZIGGY    |    10 |   230 |  6173   (6)| 00:00:01 |
----------------------------------------------------------------------------------------
```

An Auto Index is never created. Currently, Auto Indexes are only created based on *Equality* SQL predicates.

NEW IN 19c

Auto Indexing: Index DDL Operations

Some DDL operations are not permitted with Auto Indexes:

```
SQL> alter index "SYS_AI_ah0dmxua0ub7q" invisible;
alter index "SYS_AI_ah0dmxua0ub7q" invisible
*
ERROR at line 1:
ORA-65532: cannot alter or drop automatically created indexes


SQL> alter index "SYS_AI_ah0dmxua0ub7q" unusable;
alter index "SYS_AI_ah0dmxua0ub7q" unusable
*
ERROR at line 1:
ORA-65532: cannot alter or drop automatically created indexes


SQL> drop index "SYS_AI_ah0dmxua0ub7q";
drop index "SYS_AI_ah0dmxua0ub7q"
           *
ERROR at line 1:
ORA-65532: cannot alter or drop automatically created indexes
```

NEW IN 19c

Auto Indexing: Index DDL Operations

But all index DDL operations designed to improve index structure are permitted:

```
SQL> alter index "SYS_AI_ah0dmxua0ub7q" rebuild online;

Index altered.

SQL> alter index "SYS_AI_ah0dmxua0ub7q" coalesce;

Index altered.

SQL> alter index "SYS_AI_ah0dmxua0ub7q" shrink space;

Index altered.
```

Currently, Automatic Indexing does not automatically rebuild indexes, including detected UNUSABLE indexes…

“Oracle Indexing Internals & Best Practices” 2 Day Seminar

LONDON: 23-24 March 2020

Of benefit to DBAs, Developers, Solution Architects and anyone else interested in designing, developing or maintaining high performance Oracle-based applications/databases.

- Examines most available index structures/options & discusses in considerable detail how indexes function, how/when they should be used & how they should be maintained.
- Details how indexes are costed & evaluated by the Cost Based Optimizer (CBO) & how appropriate data management practices are vital for an effective indexing strategy.
- Covers many useful tips and strategies to maximise the benefits of indexes on application/database performance & scalability.

Thank You !!

richardfooteconsulting.com/indexing-webinar/