ORACLE®

ORACLE®

Oracle Database 12c

Thomas Kyte
http://asktom.oracle.com

12 Things About Oracle Database 12*c*

Thomas Kyte
http://asktom.oracle.com/

#1 Even better PL/SQL from SQL

PL/SQL from SQL

```
c##tkyte%CDB1> create table t ( x varchar2(5) );
Table created.

c##tkyte%CDB1> insert into t values ( 'a' );
c##tkyte%CDB1> insert into t values ( '1' );
c##tkyte%CDB1> insert into t values ( null );
```

PL/SQL from SQL

```
c##tkyte%CDB1> create or replace
  2  function is_number_ool(x in varchar2)
  3  return varchar2
  4  is
  5      Plsql_Num_Error exception;
  6      pragma exception_init(Plsql_Num_Error, -06502);
  7  begin
  8      if (To_Number(x) is NOT null) then
  9        return 'Y';
 10      else
 11        return '';
 12      end if;
 13  exception
 14      when Plsql_Num_Error then
 15        return 'N';
 16  end Is_Number_ool;
 17  /
Function created.
```

PL/SQL from SQL

```
c##tkyte%CDB1> select rownum, x,
  2           is_number_ool(x) is_num
  3     from t;

    ROWNUM X      IS_NUM
---------- ----- ----------
         1 a     N
         2 1     Y
         3
```

PL/SQL from SQL

```
c##tkyte%CDB1> with
  2    function Is_Number
  3    (x in varchar2) return varchar2 is
  4      Plsql_Num_Error exception;
  5      pragma exception_init(Plsql_Num_Error, -06502);
  6    begin
  7      if (To_Number(x) is NOT null) then
  8        return 'Y';
  9      else
 10        return '';
 11      end if;
 12    exception
 13      when Plsql_Num_Error then
 14        return 'N';
 15    end Is_Number;
 16  select rownum, x, is_number(x) is_num from t
 17  /
```

PL/SQL from SQL

```
select is_number_ool( to_char(object_id) ),
       is_number_ool( owner )
  from stage

call     count       cpu    elapsed       rows
------- ------  -------- ---------- ---------
Parse        1      0.00       0.00          0
Execute      1      0.00       0.00          0
Fetch      875      0.93       1.34      87310
------- ------  -------- ---------- ---------
total      877      0.93       1.34      87310
```

PL/SQL from SQL

```
with
  function Is_Number ... end Is_Number;
select is_number( to_char(object_id) ) …,

call     count        cpu    elapsed      rows
------- ------  -------- ---------- ---------
Parse        1       0.00       0.00         0
Execute      1       0.00       0.00         0
Fetch      875       0.29       0.55     87310
------- ------  -------- ---------- ---------
total      877       0.29       0.55     87310
```

#2 Improved Defaults

Improved Defaults

- Default to a sequence
- Default when null inserted
- Identity Type
- Metadata-only Defaults for NULL columns

Improved Defaults - sequences

```
c##tkyte%CDB1> create sequence s;
Sequence created.

c##tkyte%CDB1> create table t
  2   ( x int default s.nextval primary key,
  3     y varchar2(30)
  4   )
  5   /
Table created.
```

Improved Defaults - sequences

```
c##tkyte%CDB1> insert into t(y) values ('hello world');
1 row created.


c##tkyte%CDB1> select * from t;

         X Y
---------- ------------------------------
         1 hello world
```

Improved Defaults – when null

```
c##tkyte%CDB1> create table t
  2  ( x number default s.nextval primary key,
  3    y number,
  4    z number default on null 42
  5  );

Table created.
```

Improved Defaults – when null

```
c##tkyte%CDB1> insert into t (y, z) values ( 55, NULL );
c##tkyte%CDB1> insert into t (y,z) values ( 100, 200 );
c##tkyte%CDB1> insert into t (x,y,z) values (-1,-2,-3);

c##tkyte%CDB1> select * from t;

         X          Y          Z
---------- ---------- ----------
         2         55         42
         3        100        200
        -1         -2         -3
```

Improved Defaults – identities

```
c##tkyte%CDB1> create table t
  2  ( x number generated as identity,
  3    y number
  4  )
  5  /
Table created.
```

Improved Defaults – identities

```
c##tkyte%CDB1> insert into t (x,y) values (1,100);
insert into t (x,y) values (1,100)
            *
ERROR at line 1:
ORA-32795: cannot insert into a generated always identity column

c##tkyte%CDB1> insert into t (y) values (200);
1 row created.

c##tkyte%CDB1> select * from t;
         X          Y
---------- ----------
         1        200
```

Improved Defaults – identities

```
c##tkyte%CDB1> create table t
  2  ( x number generated by default
  3                 as identity
  4                 (start with 42
  5                  increment by 1000),
  6    y number
  7  )
  8  /

Table created.
```

Improved Defaults – identities

```
c##tkyte%CDB1> insert into t (x,y) values (1,100);
1 row created.

c##tkyte%CDB1> insert into t (y) values (200);
1 row created.

c##tkyte%CDB1> select * from t;

         X          Y
---------- ----------
         1        100
        42        200
```

Improved Defaults – metadata only defaults

```
c##tkyte%CDB1> create table t
  2  as
  3  select *
  4    from stage;
Table created.

c##tkyte%CDB1> exec show_space('T')
…
Full Blocks        .....................           1,437
Total Blocks............................           1,536
Total Bytes.............................      12,582,912
Total MBytes............................              12
…

PL/SQL procedure successfully completed.
```

Improved Defaults – metadata only defaults

```
c##tkyte%CDB1> set timing on
c##tkyte%CDB1> alter table t add (data char(2000) default 'x');
Table altered.

Elapsed: 00:00:00.07


ops$tkyte%ORA11GR2> set timing on
ops$tkyte%ORA11GR2> alter table t add (data char(2000) default 'x');
Table altered.

Elapsed: 00:00:28.59
```

Improved Defaults – metadata only defaults

```
c##tkyte%CDB1> exec show_space('T')
…
Full Blocks        .....................            1,437
Total Blocks............................            1,536
Total Bytes.............................       12,582,912
Total MBytes............................               12
…
PL/SQL procedure successfully completed.


ops$tkyte%ORA11GR2> exec show_space('T')
…
Total MBytes............................                9   <<<= before

Total MBytes............................              192   <<<= after
…
PL/SQL procedure successfully completed.
```

#3 Increased Size Limit for VARCHAR2, NVARCHAR2, and RAW Data Types

32k Strings

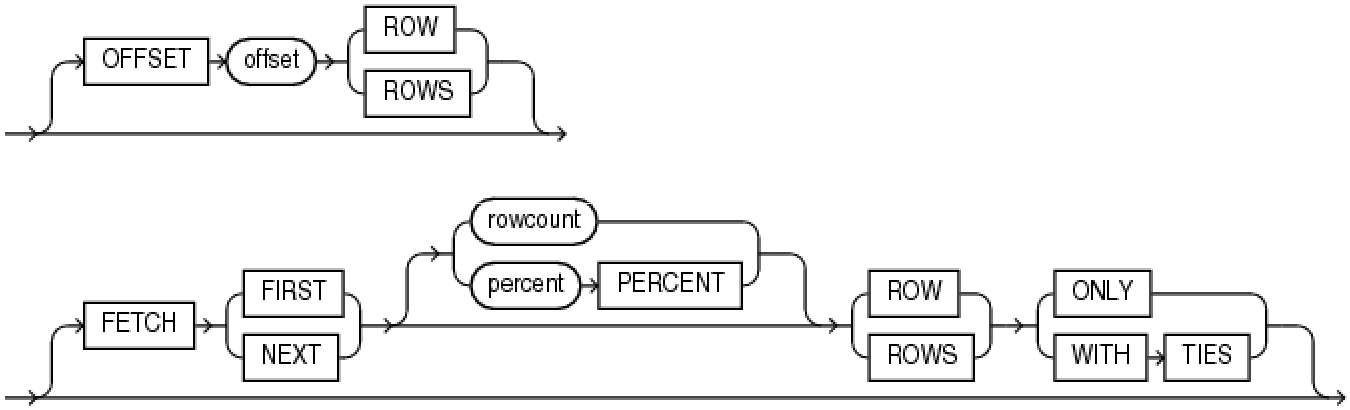
- Varchar2, NVarchar2 and Raw datatypes may be upto 32k in size, like in PL/SQL
- Compatible = 12.0.0.0 or higher
- Max_String_Size init.ora set to EXTENDED (default is not this)
- Not supported in clustered and index organized tables
- Will be stored out of line (LOB) but work just like long strings to your program

32k Strings

```
c##tkyte%CDB1> create table t ( x varchar(32767) );
Table created.

c##tkyte%CDB1> insert into t values ( rpad('*',32000,'*') );
1 row created.

c##tkyte%CDB1> select length(x) from t;

 LENGTH(X)
----------
     32000
```

#4 Easy Top-N and pagination queries

Row Limiting Clause



Row Limiting Clause

```
c##tkyte%CDB1> create table t
  2  as
  3  select * from stage;
Table created.

c##tkyte%CDB1> create index t_idx on t(owner,object_name);
Index created.

c##tkyte%CDB1> set autotrace on explain
```

Row Limiting Clause

```
c##tkyte%CDB1> select /*+ first_rows(5) */ owner, object_name, object_id
  2    from t
  3   order by owner, object_name
  4   FETCH FIRST 5 ROWS ONLY;
…
---------------------------------------------------------------------------------------
| Id  | Operation                     | Name  | Rows  | Bytes | Cost  (%CPU)| Time     |
---------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |       |     5 |  1450 |     7   (0)| 00:00:01 |
|*  1 |  VIEW                         |       |     5 |  1450 |     7   (0)| 00:00:01 |
|*  2 |   WINDOW NOSORT STOPKEY       |       |     5 |   180 |     7   (0)| 00:00:01 |
|   3 |    TABLE ACCESS BY INDEX ROWID| T     | 87310 |  3069K|     7   (0)| 00:00:01 |
|   4 |     INDEX FULL SCAN           | T_IDX |     5 |       |     3   (0)| 00:00:01 |
---------------------------------------------------------------------------------------

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

   1 - filter("from$_subquery$_003"."rowlimit_$$_rownumber"<=5)
   2 - filter(ROW_NUMBER() OVER ( ORDER BY "OWNER","OBJECT_NAME")<=5)
```

Row Limiting Clause

```
c##tkyte%CDB1> select /*+ first_rows(5) */ owner, object_name, object_id
  2     from t
  3    order by owner, object_name
  4   OFFSET 5 ROWS FETCH NEXT 5 ROWS ONLY;
…
-----------------------------------------------------------------------------------------
| Id  | Operation                     | Name  | Rows  | Bytes | Cost  (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |       |     5 |  1450 |     7   (0)| 00:00:01 |
|*  1 |  VIEW                         |       |     5 |  1450 |     7   (0)| 00:00:01 |
|*  2 |   WINDOW NOSORT STOPKEY       |       |     5 |   180 |     7   (0)| 00:00:01 |
|   3 |    TABLE ACCESS BY INDEX ROWID| T     | 87310 |  3069K|     7   (0)| 00:00:01 |
|   4 |     INDEX FULL SCAN           | T_IDX |     5 |       |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("from$_subquery$_003"."rowlimit_$$_rownumber"<=CASE  WHEN (5>=0)
              THEN 5 ELSE 0 END +5 AND "from$_subquery$_003"."rowlimit_$$_rownumber">5)
   2 - filter(ROW_NUMBER() OVER ( ORDER BY "OWNER","OBJECT_NAME")<=CASE  WHEN
              (5>=0) THEN 5 ELSE 0 END +5)
```

Row Limiting Clause

```
c##tkyte%CDB1> select owner, object_name, object_id
  2     from t
  3    order by owner, object_name
  4    FETCH NEXT 0.01 PERCENT ROWS ONLY;
…
9 rows selected.
-------------------------------------------------------------------------------------
| Id  | Operation            | Name | Rows  | Bytes |TempSpc| Cost  (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |      | 87310 |    25M|       |  1230    (1)| 00:00:01 |
|*  1 |  VIEW                |      | 87310 |    25M|       |  1230    (1)| 00:00:01 |
|   2 |   WINDOW SORT        |      | 87310 |  3069K|  4120K|  1230    (1)| 00:00:01 |
|   3 |    TABLE ACCESS FULL| T    | 87310 |  3069K|       |   401    (1)| 00:00:01 |
-------------------------------------------------------------------------------------

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

   1 - filter("from$_subquery$_003"."rowlimit_$$_rownumber"<=CEIL("from$_sub
              query$_003"."rowlimit_$$_total"*0.01/100))
```

#5 Row Pattern Matching

Row Pattern Matching

- New pattern matching clause in SQL called *“match_recognize”*
- Match_recognize clause enables users to:
  - Logically partition and order the data used in match_recognize clause
  - define patterns using regular expression syntax over pattern variables
  - the regular expression is matched against a sequence of rows
  - each pattern variable is defined using conditions on individual rows and aggregates

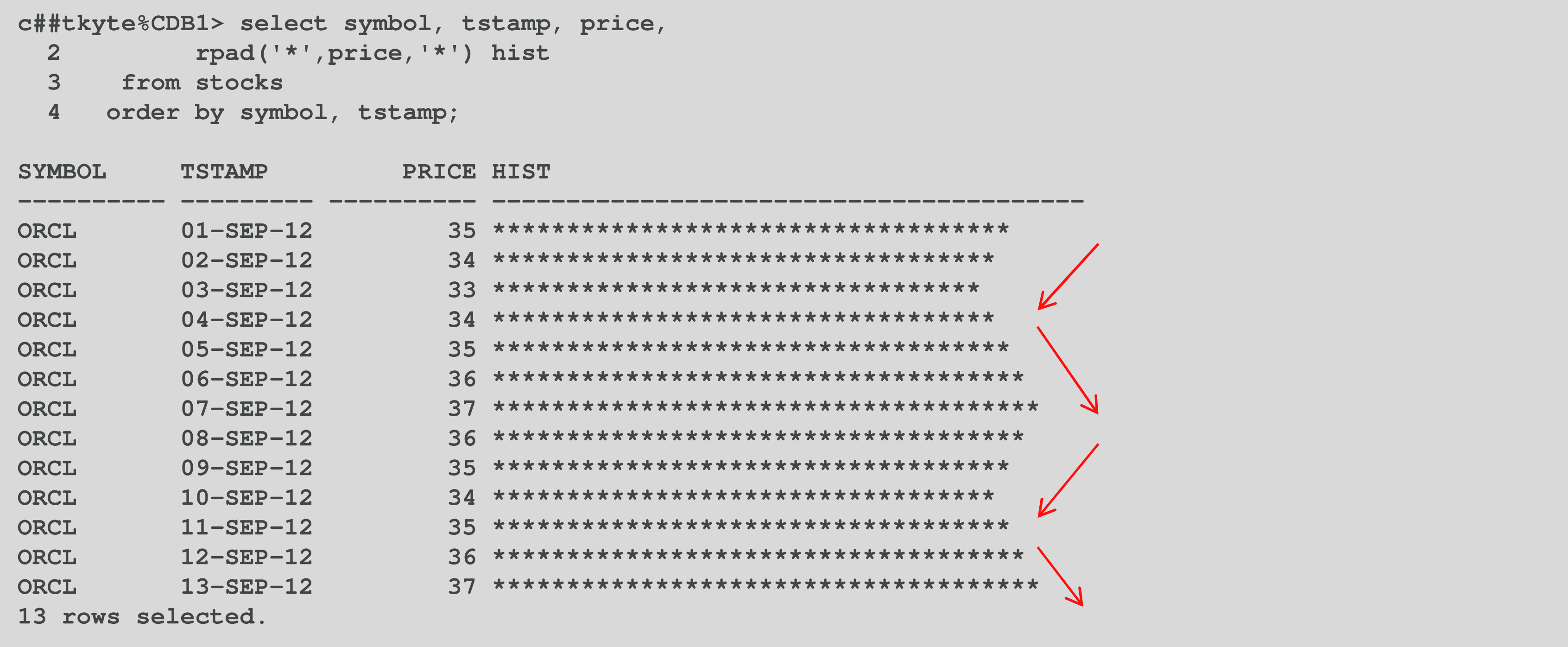
Row Pattern Matching

```
c##tkyte%CDB1> create table stocks
  2  ( symbol   varchar2(10),
  3    tstamp   date,
  4    price    number,
  5    primary key (symbol,tstamp)
  6  )
  7  organization index
  8  /
Table created.
```

Row Pattern Matching

```
c##tkyte%CDB1> declare
  2      l_data sys.odciNumberList :=
  3             sys.odciNumberList
  4             ( 35, 34, 33, 34, 35, 36,
  5             37, 36, 35, 34, 35, 36, 37 );
  6      l_cnt  number := l_data.count;
  7  begin
  8      for i in 1 .. l_cnt
  9      loop
 10          insert into stocks
 11          ( symbol, tstamp, price )
 12          values
 13          ('ORCL', sysdate-l_cnt+i, l_data(i) );
 14      end loop;
 15      commit;
 16  end;
 17  /

PL/SQL procedure successfully completed.
```

Row Pattern Matching



```
c##tkyte%CDB1> select symbol, tstamp, price,
  2            rpad('*',price,'*') hist
  3      from stocks
  4     order by symbol, tstamp;

SYMBOL     TSTAMP         PRICE HIST
---------- --------- ---------- ----------------------------------------
ORCL       01-SEP-12         35 ***********************************
ORCL       02-SEP-12         34 **********************************
ORCL       03-SEP-12         33 *********************************
ORCL       04-SEP-12         34 **********************************
ORCL       05-SEP-12         35 ***********************************
ORCL       06-SEP-12         36 ************************************
ORCL       07-SEP-12         37 *************************************
ORCL       08-SEP-12         36 ************************************
ORCL       09-SEP-12         35 ***********************************
ORCL       10-SEP-12         34 **********************************
ORCL       11-SEP-12         35 ***********************************
ORCL       12-SEP-12         36 ************************************
ORCL       13-SEP-12         37 *************************************
13 rows selected.
```

Row Pattern Matching

```
c##tkyte%CDB1> SELECT *
  2  FROM stocks MATCH_RECOGNIZE
  3  ( PARTITION BY symbol
  4    ORDER BY tstamp
  5    MEASURES
  6       STRT.tstamp AS start_tstamp,
  7       LAST(DOWN.tstamp) AS bottom_tstamp,
  8       LAST(UP.tstamp) AS end_tstamp
  9    ONE ROW PER MATCH
 10    AFTER MATCH SKIP TO LAST UP
 11    PATTERN (STRT DOWN+ UP+)
 12    DEFINE
 13      DOWN AS DOWN.price < PREV(DOWN.price),
 14      UP AS UP.price > PREV(UP.price)
 15  ) MR
 16   ORDER BY MR.symbol, MR.start_tstamp;

SYMBOL     START_TST BOTTOM_TS END_TSTAM
---------- --------- --------- ---------
ORCL       01-SEP-12 03-SEP-12 07-SEP-12
ORCL       07-SEP-12 10-SEP-12 13-SEP-12
```

```
SYMBOL     TSTAMP         PRICE HIST
---------- --------- ---------- ----------------------------------------
ORCL       01-SEP-12         35 ***********************************
ORCL       02-SEP-12         34 **********************************
ORCL       03-SEP-12         33 *********************************
ORCL       04-SEP-12         34 **********************************
ORCL       05-SEP-12         35 ***********************************
ORCL       06-SEP-12         36 ************************************
ORCL       07-SEP-12         37 *************************************
ORCL       08-SEP-12         36 ************************************
ORCL       09-SEP-12         35 ***********************************
ORCL       10-SEP-12         34 **********************************
ORCL       11-SEP-12         35 ***********************************
ORCL       12-SEP-12         36 ************************************
ORCL       13-SEP-12         37 *************************************
13 rows selected.
```

#6 Partitioning Improvements

Partitioning Improvements

- *Asynchronous Global Index Maintenance for DROP and TRUNCATE partition
- Cascade Functionality for TRUCATE and EXCHANGE partition
- Multiple partition operations in a single DDL
- Online move of a partition (without DBMS_REDEFINITION)
- *Interval + Reference partitioning

Asynchronous Global Index Maintenance

- DROP and TRUNCATE complete immediately
- We'll maintain a list of invalid data object ids and ignore those entries in the index from then on
- Automatic scheduler job PMO_DEFERRED_GIDX_MAINT_JOB will run to clean up all global indexes
- Can be run manually
- Alter index [partition] CLEANUP is another approach

Interval + Reference Partitioning

```
c##tkyte%CDB1> create table p
  2  (
  3    order#      number primary key,
  4    order_date  date,
  5    data       varchar2(30)
  6  )
  7  enable row movement
  8  partition by range(order_date)
  9  interval (numtodsinterval(1,'day'))
 10  (partition p0 values less than
 11   (to_date('01-jan-2012','dd-mon-yyyy'))
 12  )
 13  /

Table created.
```

Interval + Reference Partitioning

```
c##tkyte%CDB1> create table c1
  2  ( order#   number not null,
  3    line#    number,
  4    data     varchar2(30),
  5    constraint c1_pk primary key(order#,line#),
  6    constraint c1_fk_p foreign key(order#) references p
  7  )
  8  enable row movement
  9  partition by reference(c1_fk_p)
 10  /

Table created.
```

Interval + Reference Partitioning

```
c##tkyte%CDB1> create table c2
  2  ( order#    number not null,
  3    line#     number not null,
  4    subline# number,
  5    data      varchar2(30),
  6    constraint c2_pk primary key(order#,line#,subline#),
  7    constraint c2_fk_c1 foreign key(order#,line#) references c1
  8  )
  9  enable row movement
 10  partition by reference(c2_fk_c1)
 11  /

Table created.
```

Interval + Reference Partitioning

```
c##tkyte%CDB1> select table_name, partition_name
  2    from user_tab_partitions
  3   where table_name in ( 'P', 'C1', 'C2' )
  4   order by table_name, partition_name
  5  /

TABLE_NAME           PARTITION_NAME
-------------------- --------------------
C1                   P0
C2                   P0
P                    P0
```

Interval + Reference Partitioning

```
c##tkyte%CDB1> insert into p (order#, order_date, data)
  2  values ( 1, to_date('15-jan-2012'), 'order data' );
1 row created.

c##tkyte%CDB1> insert into c1 (order#, line#, data)
  2  values ( 1, 1, 'line data 1' );
1 row created.

c##tkyte%CDB1> insert into c1 (order#, line#, data)
  2  values ( 1, 2, 'line data 2' );
1 row created.

c##tkyte%CDB1> insert into c2 (order#, line#, subline#, data)
  2  values ( 1, 1, 1, 'sub-line data' );
1 row created.
```

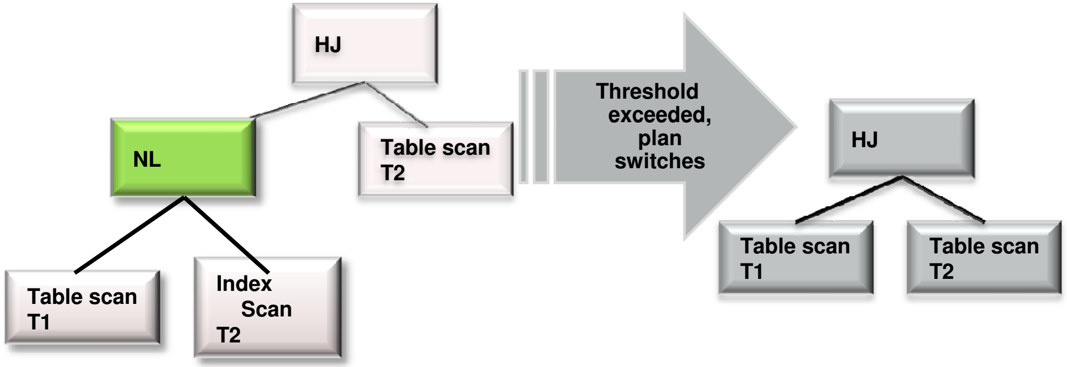
Interval + Reference Partitioning

```
c##tkyte%CDB1> select table_name, partition_name
  2     from user_tab_partitions
  3    where table_name in ( 'P', 'C1', 'C2' )
  4    order by table_name, partition_name
  5   /

TABLE_NAME                     PARTITION_NAME
------------------------------ ------------------------------
C1                             P0
C1                             SYS_P569
C2                             P0
C2                             SYS_P569
P                              P0
P                              SYS_P569

6 rows selected.
```

#7 Adaptive Execution Plans

Adaptive Execution Plans

Good SQL execution without intervention



- Plan decision deferred until runtime
- Final decision is based on rows seen during execution
- Bad effects of skew eliminated
- Dramatic improvements seen with EBS reports

Adaptive Execution Plans

```
c##tkyte%CDB1> create table t1 as select * from stage;
c##tkyte%CDB1> insert into t1 select * from t1;

c##tkyte%CDB1> create table t2 as select * from stage;
c##tkyte%CDB1> insert into t2 select * from t2;

c##tkyte%CDB1> create table t3 as select * from stage;
c##tkyte%CDB1> insert into t3 select * from t3;

c##tkyte%CDB1> create table t4 as select * from stage;
c##tkyte%CDB1> insert into t4 select * from t4;

c##tkyte%CDB1> create index t1_idx on t1(object_id);
c##tkyte%CDB1> create index t2_idx on t2(object_id);
c##tkyte%CDB1> create index t3_idx on t3(object_id);
c##tkyte%CDB1> create index t4_idx on t4(object_id);
```

Adaptive Execution Plans

```
c##tkyte%CDB1> exec dbms_stats.gather_table_stats( user, 'T1' );
PL/SQL procedure successfully completed.

c##tkyte%CDB1> exec dbms_stats.gather_table_stats( user, 'T2' );
PL/SQL procedure successfully completed.

c##tkyte%CDB1> exec dbms_stats.gather_table_stats( user, 'T3' );
PL/SQL procedure successfully completed.

c##tkyte%CDB1> exec dbms_stats.gather_table_stats( user, 'T4' );
PL/SQL procedure successfully completed.
```

Adaptive Execution Plans

```
c##tkyte%CDB1> update t1 set object_id = 42 where mod(object_id,100) = 0;
1740 rows updated.

c##tkyte%CDB1> update t2 set object_id = 42 where mod(object_id,100) = 0;
1740 rows updated.

c##tkyte%CDB1> update t3 set object_id = 42 where mod(object_id,100) = 0;
1740 rows updated.

c##tkyte%CDB1> update t4 set object_id = 42 where mod(object_id,100) = 0;
1740 rows updated.

c##tkyte%CDB1> commit;
Commit complete.
```

Adaptive Execution Plans

```
c##tkyte%CDB1> set autotrace traceonly explain
c##tkyte%CDB1> select *
  2    from t1, t2, t3, t4
  3   where t1.object_id = t2.object_id+1
  4     and t2.object_id = t3.object_id+1
  5     and t3.object_id = t4.object_id+1
  6     and t4.object_id = 42;

-------------------------------------------------------------------------------------------------
| Id  | Operation                               | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                        |        |    15 |  6840 |   45   (0)| 00:00:01 |
|   1 |  NESTED LOOPS                           |        |       |       |            |          |
|   2 |   NESTED LOOPS                          |        |    15 |  6840 |   45   (0)| 00:00:01 |
|   3 |    NESTED LOOPS                         |        |     8 |  2736 |   21   (0)| 00:00:01 |
|   4 |     NESTED LOOPS                        |        |     4 |   912 |    9   (0)| 00:00:01 |
|   5 |     TABLE ACCESS BY INDEX ROWID BATCHED| T4     |     2 |   228 |    3   (0)| 00:00:01 |
|*  6 |       INDEX RANGE SCAN                  | T4_IDX |     2 |       |    1   (0)| 00:00:01 |
|   7 |     TABLE ACCESS BY INDEX ROWID BATCHED| T3     |     2 |   228 |    3   (0)| 00:00:01 |
|*  8 |       INDEX RANGE SCAN                  | T3_IDX |     2 |       |    1   (0)| 00:00:01 |
|   9 |    TABLE ACCESS BY INDEX ROWID BATCHED  | T2     |     2 |   228 |    3   (0)| 00:00:01 |
|* 10 |     INDEX RANGE SCAN                    | T2_IDX |     2 |       |    1   (0)| 00:00:01 |
|* 11 |   INDEX RANGE SCAN                      | T1_IDX |     2 |       |    1   (0)| 00:00:01 |
|  12 |  TABLE ACCESS BY INDEX ROWID            | T1     |     2 |   228 |    3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------
   6 - access("T4"."OBJECT_ID"=42)
   8 - access("T3"."OBJECT_ID"="T4"."OBJECT_ID"+1)
  10 - access("T2"."OBJECT_ID"="T3"."OBJECT_ID"+1)
  11 - access("T1"."OBJECT_ID"="T2"."OBJECT_ID"+1)
```

Adaptive Execution Plans

```
c##tkyte%CDB1> select * from table(dbms_xplan.display_cursor('ff8rq6av4k4j5',null,'ALLSTATS LAST'));

select /*+ gather_plan_statistics */ *   from t1, t2, t3, t4  where
t1.object_id = t2.object_id+1    and t2.object_id = t3.object_id+1
and t3.object_id = t4.object_id+1    and t4.object_id = 42

Plan hash value: 1662286539

------------------------------------------------------------------------------------
| Id  | Operation                              | Name   | Starts | E-Rows | A-Rows |
------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                       |        |      1 |        |  13936 |
|*  1 |  HASH JOIN                             |        |      1 |     15 |  13936 |
|*  2 |   HASH JOIN                            |        |      1 |      8 |   6968 |
|*  3 |    HASH JOIN                           |        |      1 |      4 |   3484 |
|   4 |     TABLE ACCESS BY INDEX ROWID BATCHED| T4     |      1 |      2 |   1742 |
|*  5 |      INDEX RANGE SCAN                  | T4_IDX |      1 |      2 |   1742 |
|   6 |     TABLE ACCESS FULL                  | T3     |      1 |      2 |    174K|
|   7 |    TABLE ACCESS FULL                   | T2     |      1 |      2 |    174K|
|   8 |   TABLE ACCESS FULL                    | T1     |      1 |      2 |    174K|
------------------------------------------------------------------------------------

   1 - access("T1"."OBJECT_ID"="T2"."OBJECT_ID"+1)
   2 - access("T2"."OBJECT_ID"="T3"."OBJECT_ID"+1)
   3 - access("T3"."OBJECT_ID"="T4"."OBJECT_ID"+1)
   5 - access("T4"."OBJECT_ID"=42)
```

Adaptive Execution Plans

```
Row Source Operation
---------------------------------------------------
HASH JOIN  (cr=15900 pr=325 pw=0 time=333183 us cost=45 size=6840 card=15)
 NESTED LOOPS  (cr=12763 pr=0 pw=0 time=393875 us)
  NESTED LOOPS  (cr=12763 pr=0 pw=0 time=303025 us cost=45 size=6840 card=15)
   STATISTICS COLLECTOR  (cr=12763 pr=0 pw=0 time=256384 us)
    HASH JOIN  (cr=12763 pr=0 pw=0 time=174171 us cost=21 size=2736 card=8)
     NESTED LOOPS  (cr=8128 pr=0 pw=0 time=109490 us cost=21 size=2736 card=8)
      STATISTICS COLLECTOR  (cr=8128 pr=0 pw=0 time=80742 us)
       HASH JOIN  (cr=8128 pr=0 pw=0 time=46346 us cost=9 size=912 card=4)
        NESTED LOOPS  (cr=3493 pr=0 pw=0 time=209504 us cost=9 size=912 card=4)
         STATISTICS COLLECTOR  (cr=3493 pr=0 pw=0 time=183616 us)
          TABLE ACCESS BY INDEX ROWID BATCHED T4 (cr=3493 pr=0 pw=0 time=157290 us cost=3 size=228 card=2)
           INDEX RANGE SCAN T4_IDX (cr=13 pr=0 pw=0 time=16878 us cost=1 size=0 card=2)(object id 91414)
         TABLE ACCESS BY INDEX ROWID BATCHED T3 (cr=0 pr=0 pw=0 time=0 us cost=3 size=228 card=2)
          INDEX RANGE SCAN T3_IDX (cr=0 pr=0 pw=0 time=0 us cost=1 size=0 card=2)(object id 91413)
        TABLE ACCESS FULL T3 (cr=4635 pr=0 pw=0 time=350678 us cost=3 size=228 card=2)
      TABLE ACCESS BY INDEX ROWID BATCHED T2 (cr=0 pr=0 pw=0 time=0 us cost=3 size=228 card=2)
       INDEX RANGE SCAN T2_IDX (cr=0 pr=0 pw=0 time=0 us cost=1 size=0 card=2)(object id 91412)
     TABLE ACCESS FULL T2 (cr=4635 pr=0 pw=0 time=318272 us cost=3 size=228 card=2)
   INDEX RANGE SCAN T1_IDX (cr=0 pr=0 pw=0 time=0 us cost=1 size=0 card=2)(object id 91411)
  TABLE ACCESS BY INDEX ROWID T1 (cr=0 pr=0 pw=0 time=0 us cost=3 size=228 card=2)
 TABLE ACCESS FULL T1 (cr=3137 pr=325 pw=0 time=362892 us cost=3 size=228 card=2)
```

#8 Enhanced Statistics

ORACLE

Dynamic Sampling

- “These go to eleven”
- When you turn it up to eleven dynamic sampling is
  - Automatic
  - Persistent
- Automatically turned up to eleven for parallel query

Statistics During Loads

```
c##tkyte%CDB1> create table t
  2  as
  3  select *
  4    from stage;

Table created.

c##tkyte%CDB1> select num_rows, last_analyzed
  2    from user_tables
  3   where table_name = 'T'
  4  /

  NUM_ROWS LAST_ANAL
---------- ---------
     87310 14-SEP-12
```

Session Private Statistics for GTT's

```
c##tkyte%CDB1> create table t
  2  as
  3  select *
  4    from stage;
Table created.

c##tkyte%CDB1> create index t_idx on t(object_id);
Index created.

c##tkyte%CDB1> create global temporary table gtt
  2   ( x int )
  3  on commit delete rows
  4  /
Table created.
```

Session Private Statistics for GTT’s

```
c##tkyte%CDB1> insert into gtt
  2  select object_id
  3    from t
  4   where rownum <= 5;
5 rows created.

c##tkyte%CDB1> select num_rows from dba_tab_statistics where owner = user and table_name = 'GTT';
  NUM_ROWS
----------

c##tkyte%CDB1> exec dbms_stats.gather_table_stats( user, 'GTT' );
PL/SQL procedure successfully completed.

c##tkyte%CDB1> select num_rows from dba_tab_statistics where owner = user and table_name = 'GTT';

  NUM_ROWS
----------
         5
```

Session Private Statistics for GTT’s

```
c##tkyte%CDB1> select * from t where object_id in (select x from gtt);

--------------------------------------------------------------------------------------
| Id  | Operation                     | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |       |     5 |   585 |     8   (0)| 00:00:01 |
|   1 |  NESTED LOOPS                 |       |       |       |            |          |
|   2 |   NESTED LOOPS                |       |     5 |   585 |     8   (0)| 00:00:01 |
|   3 |    SORT UNIQUE                |       |     5 |    15 |     2   (0)| 00:00:01 |
|   4 |     TABLE ACCESS FULL         | GTT   |     5 |    15 |     2   (0)| 00:00:01 |
|*  5 |    INDEX RANGE SCAN           | T_IDX |     1 |       |     1   (0)| 00:00:01 |
|   6 |   TABLE ACCESS BY INDEX ROWID| T     |     1 |   114 |     2   (0)| 00:00:01 |
--------------------------------------------------------------------------------------

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

   5 - access("OBJECT_ID"="X")

Note
-----
   - Global temporary table session private statistics used
```

Session Private Statistics for GTT’s

```
c##tkyte%CDB1> connect /
Connected.
c##tkyte%CDB1> set linesize 100
c##tkyte%CDB1> insert into gtt select object_id from stage;

87310 rows created.

c##tkyte%CDB1> select num_rows from dba_tab_statistics where owner = user
and table_name = 'GTT';

  NUM_ROWS
----------
```

Session Private Statistics for GTT’s

```
c##tkyte%CDB1> set autotrace traceonly explain
c##tkyte%CDB1> select * from t where object_id in (select x from gtt);

----------------------------------------------------------------------------
| Id  | Operation            | Name | Rows  | Bytes | Cost  (%CPU)| Time     |
----------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |      | 78938 |  9790K|   817    (1)| 00:00:01 |
|*  1 |  HASH JOIN           |      | 78938 |  9790K|   817    (1)| 00:00:01 |
|   2 |   SORT UNIQUE        |      | 78938 |  1002K|    39    (0)| 00:00:01 |
|   3 |    TABLE ACCESS FULL| GTT  | 78938 |  1002K|    39    (0)| 00:00:01 |
|   4 |   TABLE ACCESS FULL | T    | 87310 |  9720K|   401    (1)| 00:00:01 |
----------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   1 - access("OBJECT_ID"="X")

Note
-----
   - dynamic sampling used for this statement (level=2)
```

Session Private Statistics for GTT’s

```
c##tkyte%CDB1> exec dbms_stats.set_table_stats( user, 'GTT', numrows => 300 );
PL/SQL procedure successfully completed.

c##tkyte%CDB1> set autotrace traceonly explain
c##tkyte%CDB1> select * from t where object_id in (select x from gtt);
--------------------------------------------------------------------------------------
| Id  | Operation                     | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |       |     9 |  1143 |    47   (0)| 00:00:01 |
|   1 |  NESTED LOOPS                 |       |       |       |            |          |
|   2 |   NESTED LOOPS                |       |     9 |  1143 |    47   (0)| 00:00:01 |
|   3 |    SORT UNIQUE                |       |   300 |  3900 |    29   (0)| 00:00:01 |
|   4 |     TABLE ACCESS FULL         | GTT   |   300 |  3900 |    29   (0)| 00:00:01 |
|*  5 |    INDEX RANGE SCAN           | T_IDX |     1 |       |     1   (0)| 00:00:01 |
|   6 |   TABLE ACCESS BY INDEX ROWID| T     |     1 |   114 |     2   (0)| 00:00:01 |
--------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   5 - access("OBJECT_ID"="X")

Note
-----
   - Global temporary table session private statistics used
```

#9 Temporary UNDO

Temporary UNDO

- UNDO for temporary tables can now be managed in TEMP
- Reduce the amount of UNDO in the UNDO tablespace
  - Better for retention periods for “real” data
- Reduce the size of the redo generated
- Allows for DML on temporary tables in Active Data Guard

- ALTER SYSTEM/SESSION SET TEMP_UNDO_ENABLED=true|false

Temporary UNDO

```
c##tkyte%CDB1> create global temporary table gtt
  2  on commit delete rows
  3  as
  4  select * from stage where 1=0;

Table created.
```

Temporary UNDO

```
c##tkyte%CDB1> alter session set temp_undo_enabled = false;
Session altered.

c##tkyte%CDB1> insert into gtt
  2  select *
  3    from stage;
87310 rows created.

Statistics
----------------------------------------------------------
…
     566304  redo size
…

c##tkyte%CDB1> update gtt
  2     set object_name = lower(object_name);
87310 rows updated.

Statistics
----------------------------------------------------------
…
    8243680  redo size
…
```

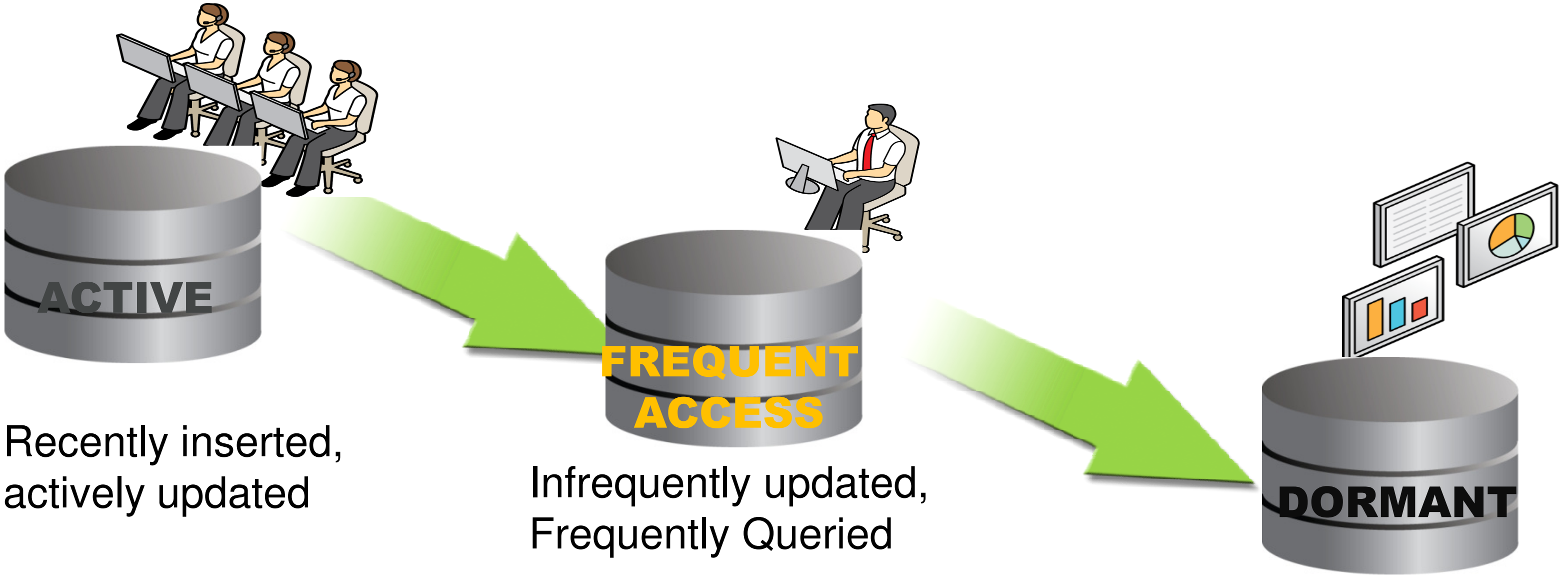
Temporary UNDO

```
c##tkyte%CDB1> alter session set temp_undo_enabled = true;
Session altered.

c##tkyte%CDB1> insert into gtt
  2  select *
  3    from stage;
87310 rows created.

Statistics
----------------------------------------------------------
…
        280  redo size
…

c##tkyte%CDB1> update gtt
  2     set object_name = lower(object_name);
87310 rows updated.

Statistics
----------------------------------------------------------
…
          0  redo size
…
```

#10 Data Optimization

ILM: Hot/Cold Data Classification

Enhanced Insight into Data Usage: “heat map”



ACTIVE

Recently inserted, actively updated

FREQUENT ACCESS

Infrequently updated, Frequently Queried

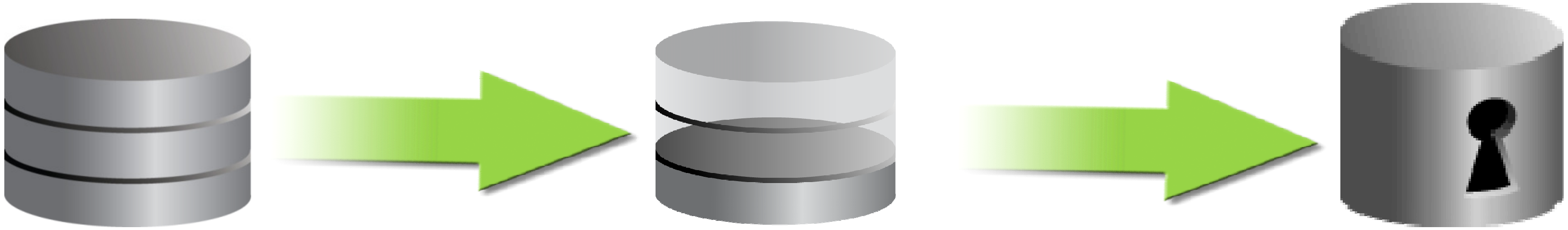
DORMANT

Retained for long term analytics and compliance with corporate policies and regulations

- Block and Segment level statistics on last Read and last Update

ILM: Automatic Compression & Tiering

Usage based and custom compression and tiering



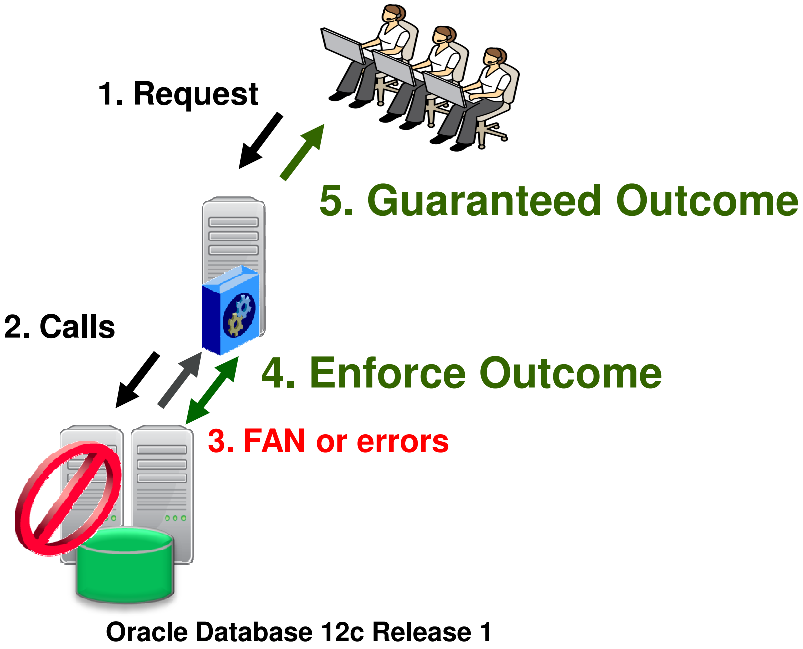
```
ALTER TABLE orders
ILM ADD CompressionPolicy
COMPRESS Partitions for Query
AFTER 90 days from creation;
```

```
ALTER TABLE sales
ILM ADD MovePolicy
TIER Partitions TO ‘Archive_TBS’
ON OrdersClosedPolicy;
```

#11 Application Continuity

Transaction Guard

FIRST RDBMS to preserve COMMIT outcome

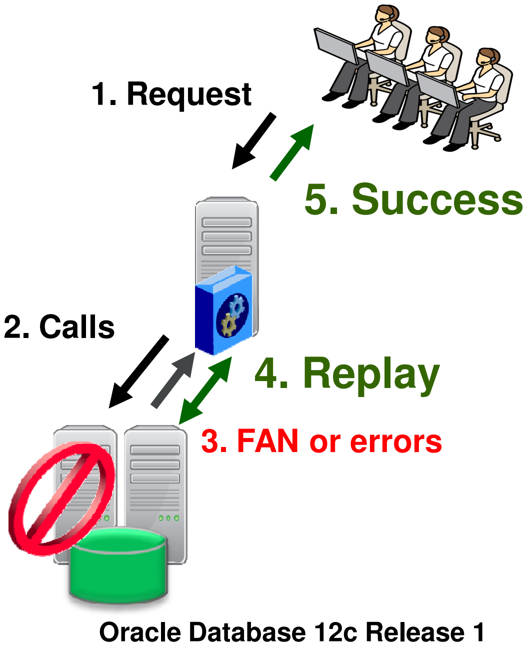


- Known outcome for every transaction
- At-most-once transaction execution
- Used by Application Continuity
- Available for JDBC-thin, OCI, OCCI, ODP.net

Without Transaction Guard, retries can cause logical corruption

Application Continuity

First RDBMS to mask planned/unplanned outages



- Improves end user experience
- Improves developer productivity
- Application transparent when using Oracle stack
- Enabled with WebLogic Server, Peoplesoft, Fusion Apps, Siebel(possibly)

#12 SQL ‘Esperanto’

SQL Translation Framework

- Migrations – without changing SQL
- Translations for
  - Sybase
  - MS SQLServer
  - Partial DB2
- Or, do it yourself

SQL Translation Framework

```
ops$tkyte%ORA12CR1> begin
  2      dbms_sql_translator.create_profile( 'MY_PROFILE' );
  3      dbms_sql_translator.register_sql_translation
  4      ( 'MY_PROFILE',
  5        'select * from scott.emp',
  6        'select * from scott.dept' );
  7  end;
  8  /

PL/SQL procedure successfully completed.
```

SQL Translation Framework

```
ops$tkyte%ORA12CR1> alter session set
                         sql_translation_profile = MY_PROFILE;
Session altered.

ops$tkyte%ORA12CR1> alter session set events =
                          '10601 trace name context forever, level 32';
Session altered.
```

SQL Translation Framework

```
ops$tkyte%ORA12CR1> select * from scott.emp;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

Hardware and Software

ORACLE®

Engineered to Work Together

ORACLE®