

U1M10.LW.Basic Parallel Execution

Shkrabatouskaya Vera

https://github.com/VeraShkrabatouskaya/DataMola_Data-Camping-2022

2. Oracle Architecture - Parallel execution

2.1. Task 01: CREATE Example of Select Parallel execution

Create table Calendar on 1000 rows.

The screenshot shows the Oracle SQL Worksheet interface. The code in the worksheet window is as follows:

```
TO_DATE( '12/31/' || TO_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' )
END ) end_of_cal_quarter,
TO_CHAR( sd + rn, 'Q' ) calendar_quarter_number,
TO_CHAR( sd + rn, 'YYYY' ) calendar_year,
( TO_DATE( '12/31/' || TO_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' )
- TRUNC( sd + rn, 'YEAR' ) ) days_in_cal_year,
TRUNC( sd + rn, 'YEAR' ) beg_of_cal_year,
TO_DATE( '12/31/' || TO_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' ) end_of_cal_year
FROM
(
  SELECT
    TO_DATE( '12/31/2011', 'MM/DD/YYYY' ) sd,
    rownum rn
  FROM dual
  CONNECT BY level <=1000
);
Grant succeeded.
User U_DW_DATA_02 altered.
Session altered.
Table CALENDAR created.
1,000 rows inserted.
```

The output window shows the results of the query:

```
Script Output X | Query Result X
| Task completed in 0.093 seconds
```

Grant succeeded.
User U_DW_DATA_02 altered.
Session altered.
Table CALENDAR created.
1,000 rows inserted.

Insert values into the table.

The screenshot shows the Oracle SQL Worksheet interface. The code in the worksheet window is as follows:

```
TO_CHAR( sd + rn, 'Q' ) calendar_quarter_number,
TO_CHAR( sd + rn, 'YYYY' ) calendar_year,
( TO_DATE( '12/31/' || TO_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' )
- TRUNC( sd + rn, 'YEAR' ) ) days_in_cal_year,
TRUNC( sd + rn, 'YEAR' ) beg_of_cal_year,
TO_DATE( '12/31/' || TO_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' ) end_of_cal_year
FROM
(
  SELECT
    TO_DATE( '12/31/2011', 'MM/DD/YYYY' ) sd,
    rownum rn
  FROM dual
  CONNECT BY level <=1000
);
select * from calendar;
```

The output window shows the results of the query:

```
Script Output X | Query Result X
| Fetched 50 rows in 0.019 seconds
```

TIME_ID	DAY_NAME	DAY_NUMBER_IN_WEEK	DAY_NUMBER_IN_MONTH	DAY_NUMBER_IN_YEAR	CALENDAR_WEEK_NUMBER	WEEK_ENDING_DATE	CALENDAR_MONTH_NUMBER	DAYS_IN_CAL_MONTH	BI
1	03-JUL-13 Wednesday	3	03	184	1	07-JUL-13	07	31	31-
2	04-JUL-13 Thursday	4	04	185	1	07-JUL-13	07	31	31-
3	05-JUL-13 Friday	5	05	186	1	07-JUL-13	07	31	31-
4	06-JUL-13 Saturday	6	06	187	1	07-JUL-13	07	31	31-
5	07-JUL-13 Sunday	7	07	188	1	07-JUL-13	07	31	31-
6	08-JUL-13 Monday	1	08	189	2	14-JUL-13	07	31	31-
7	09-JUL-13 Tuesday	2	09	190	2	14-JUL-13	07	31	31-
8	10-JUL-13 Wednesday	3	10	191	2	14-JUL-13	07	31	31-
9	11-JUL-13 Thursday	4	11	192	2	14-JUL-13	07	31	31-
10	12-JUL-13 Friday	5	12	193	2	14-JUL-13	07	31	31-
11	13-JUL-13 Saturday	6	13	194	2	14-JUL-13	07	31	31-
12	14-JUL-13 Sunday	7	14	195	2	14-JUL-13	07	31	31-

Let's look at the explain plan.

The screenshot shows the Oracle SQL Developer interface. The top navigation bar includes tabs for 'Welcome Page', 'VeraDB-port1521', and 'Lab10.sql'. Below the bar, there's a toolbar with various icons. The main area has a 'Worksheet' tab active, displaying a complex PL/SQL block. This block uses TO_CHAR and TO_DATE functions to calculate calendar quarter numbers and years, and it includes a recursive CTE to generate rows from 1 to 1000. A 'select * from calendar;' statement is also present. Below the worksheet, there are tabs for 'Script Output', 'Query Result', and 'Explain Plan'. The 'Explain Plan' tab is currently selected, showing the execution plan for the query. The plan details the SELECT statement, object names, options, cardinality, and cost. It also shows the execution path through the 'TABLE ACCESS' and 'Other XML' stages, and provides detailed information about the 'info' node, including 'has_user_tab' (yes), 'db_version' (21.0.0.0), 'parse_schema' ('U_DW_DATA_02'), 'dynamic_sampling' (note='y'), and 'plan_hash_full' (3924818931).

```
TO_CHAR( sd + rn, 'Q' ) calendar_quarter_number,
TO_CHAR( sd + rn, 'YYYY' ) calendar_year,
( TO_DATE( '12/31/' || TO_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' )
- TRUNC( sd + rn, 'YEAR' ) ) days_in_cal_year,
TRUNC( sd + rn, 'YEAR' ) beg_of_cal_year,
TO_DATE( '12/31/' || TO_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' ) end_of_cal_year
FROM
(
  SELECT
    TO_DATE( '12/31/2011', 'MM/DD/YYYY' ) sd,
    rownum rn
  FROM dual
  CONNECT BY level <=1000
);
select * from calendar;
```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1000	5
TABLE ACCESS	U_DW_DATA_02.CALENDAR	FULL	1000	5
Other XML				
(info)				
info type='has_user_tab'				
yes				
info type='db_version'				
21.0.0.0				
info type='parse_schema'				
'U_DW_DATA_02'				
info type='dynamic_sampling' note='y'				
2				
info type='plan_hash_full'				
3924818931				

Look at the SQL history.

SQL Worksheet | History

0.058 seconds

Worksheet Query Builder

TO_CHAR(sd + rn, 'Q') calendar_quarter_number,

Script Output x | Query Result x | Explain Plan x

SQL 0.058 seconds

OPERATION	OBJECT_NAME	OPTIONS			
SQL History					
SQL	Connection	TimeStamp	Type	Executed	Duration(seconds)
select * from calendar;	VeraDB-port...	28-JUL-22 1...	SQL	3	0.019
file:/D:/Vera/Datamola_2022/U1M10.Basic Parallel Execution/Lab10.sql	VeraDB-port...	28-JUL-22 1...	Script	20	0.0
SELECT TRUNC(sd + rn) time_id, TO_CHAR(sd + rn, 'fmDay') day_id,	VeraDB-port...	28-JUL-22 1...	SQL	2	0.122
file:/D:/Vera/Datamola_2022/Labs/Lab_7/lreport/scripts_task01/Calen...	VeraDB-port...	28-JUL-22 1...	Script	1	0.0
select * from USER tablespaces;	VeraDB-port...	28-JUL-22 1...	SQL	2	0.01
SELECT * from dba_data_files;	VeraDB-port...	28-JUL-22 1...	SQL	4	0.011
file:/D:/Vera/Datamola_2022/Labs/Lab_8/scripts/TABLES/t_FCT_busin...	VeraDB-port...	27-JUL-22 0...	Script	6	0.0
file:/D:/Vera/Datamola_2022/Labs/Lab_8/scripts/TABLES/t_DIM_prom...	VeraDB-port...	27-JUL-22 0...	Script	1	0.0
file:/D:/Vera/Datamola_2022/Labs/Lab_8/scripts/TABLES/t_DIM_produ...	VeraDB-port...	27-JUL-22 0...	Script	1	0.0
file:/D:/Vera/Datamola_2022/Labs/Lab_8/scripts/TABLES/t_DIM_gen...	VeraDB-port...	27-JUL-22 0...	Script	1	0.0
file:/D:/Vera/Datamola_2022/Labs/Lab_8/scripts/TABLES/t_DIM_agenc...	VeraDB-port...	27-JUL-22 0...	Script	1	0.0
file:/D:/Vera/Datamola_2022/Labs/Lab_8/scripts/TABLES/t_DIM_empl...	VeraDB-port...	27-JUL-22 0...	Script	2	0.0
file:/D:/Vera/Datamola_2022/Labs/Lab_8/scripts/TABLES/t_DIM_custo...	VeraDB-port...	27-JUL-22 0...	Script	2	0.0
file:/D:/Vera/Datamola_2022/Labs/Lab_8/scripts/TABLES/t_customer.sq...	VeraDB-port...	27-JUL-22 0...	Script	2	0.0
file:/D:/Vera/Datamola_2022/Labs/Lab_8/scripts/Storage level SA/dm...	VeraDB-port...	27-JUL-22 0...	Script	1	0.0
file:/D:/Vera/Datamola_2022/Labs/Lab_8/scripts/Storage level SA/dm...	VeraDB-port...	27-JUL-22 0...	Script	1	0.0
file:/D:/Vera/Datamola_2022/Labs/Lab_8/scripts/Storage level SA/dm...	VeraDB-port...	27-JUL-22 0...	Script	4	0.0
file:/D:/Vera/Datamola_2022/Labs/Lab_8/scripts/Storage level SA/dm...	VeraDB-port...	27-JUL-22 0...	Script	1	0.0
file:/D:/Vera/Datamola_2022/Labs/Lab_8/scripts/Storage level SA/dm...	VeraDB-port...	27-JUL-22 0...	Script	2	0.0
file:/D:/Vera/Datamola_2022/Labs/Lab_8/scripts/Storage level SA/dm...	VeraDB-port...	27-JUL-22 0...	Script	5	0.0
file:/D:/Vera/Datamola_2022/Labs/Lab_8/scripts/Storage level SA/ts_s...	VeraDB	27-JUL-22 0...	Script	1	0.0
file:/D:/Vera/Datamola_2022/Labs/Lab_7/lreport/scripts_task01/u_dwh...	VeraDB	26-JUL-22 0...	Script	1	0.0
select * from lc_geo_systems;	VeraDB	26-JUL-22 0...	SQL	2	0.01
select * from lc_geo_regions;	VeraDB	26-JUL-22 0...	SQL	2	0.015
select * from t_geo_parts;	VeraDB	26-JUL-22 0...	SQL	2	0.012
select * from t_neo_object_links;	VeraDB	26-JUL-22 0...	SQL	2	0.014

The time it takes to explain plan is 0.058 seconds. The time it takes to execute a request is 0.019 seconds. Cost is 5.

Use Parallel execution:

Screenshot of the Oracle SQL Developer interface showing a query in the Worksheet tab and its results in the Query Result tab.

```

Welcome Page | Lab10.sql | 
SQL Worksheet | History | 
Worksheet | Query Builder | 
TO_CHAR( sd + rn, 'Q' ) calendar_quarter_number,
TO_CHAR( sd + rn, 'YYYY' ) calendar_year,
( TO_DATE( '12/31/' || TO_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' )
- TRUNC( sd + rn, 'YEAR' ) ) days_in_cal_year,
TRUNC( sd + rn, 'YEAR' ) beg_of_cal_year,
TO_DATE( '12/31/' || TO_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' ) end_of_cal_year
FROM
(
  SELECT
    TO_DATE( '12/31/2011', 'MM/DD/YYYY' ) sd,
    rownum rn
  FROM dual
  CONNECT BY level <=1000
);
SELECT /*+ PARALLEL(calendar,4) */ * FROM calendar;

```

Query Result

All Rows Fetched: 1000 in 0.129 seconds

TIME_ID	DAY_NAME	DAY_NUMBER_IN_WEEK	DAY_NUMBER_IN_MONTH	DAY_NUMBER_IN_YEAR	CALENDAR_WEEK_NUMBER	WEEK_ENDING_DATE	CALENDAR_MONTH_NUMBER	DAYS_IN_CAL_MONTH	END_OF_CAL_YEAR
1 01-JAN-12	Sunday	7	01	001	1	01-JAN-12	01	31	31-12
2 02-JAN-12	Monday	1	02	002	1	08-JAN-12	01	31	31-
3 03-JAN-12	Tuesday	2	03	003	1	08-JAN-12	01	31	31-
4 04-JAN-12	Wednesday	3	04	004	1	08-JAN-12	01	31	31-
5 05-JAN-12	Thursday	4	05	005	1	08-JAN-12	01	31	31-
6 06-JAN-12	Friday	5	06	006	1	08-JAN-12	01	31	31-
7 07-JAN-12	Saturday	6	07	007	1	08-JAN-12	01	31	31-
8 08-JAN-12	Sunday	7	08	008	2	08-JAN-12	01	31	31-
9 09-JAN-12	Monday	1	09	009	2	15-JAN-12	01	31	31-
10 10-JAN-12	Tuesday	2	10	010	2	15-JAN-12	01	31	31-
11 11-JAN-12	Wednesday	3	11	011	2	15-JAN-12	01	31	31-
12 12-JAN-12	Thursday	4	12	012	2	15-JAN-12	01	31	31-
13 13-JAN-12	Friday	5	13	013	2	15-JAN-12	01	31	31-

Let's look at the explain plan with Parallel Hint.

Screenshot of the Oracle SQL Developer interface showing the same query in the Worksheet tab and its Explain Plan in the Query Result tab.

```

Welcome Page | Lab10.sql | 
SQL Worksheet | History | 
Worksheet | Query Builder | 
TO_CHAR( sd + rn, 'Q' ) calendar_quarter_number,
TO_CHAR( sd + rn, 'YYYY' ) calendar_year,
( TO_DATE( '12/31/' || TO_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' )
- TRUNC( sd + rn, 'YEAR' ) ) days_in_cal_year,
TRUNC( sd + rn, 'YEAR' ) beg_of_cal_year,
TO_DATE( '12/31/' || TO_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' ) end_of_cal_year
FROM
(
  SELECT
    TO_DATE( '12/31/2011', 'MM/DD/YYYY' ) sd,
    rownum rn
  FROM dual
  CONNECT BY level <=1000
);
SELECT /*+ PARALLEL(calendar,4) */ * FROM calendar;

```

Query Result

0.039 seconds

Explain Plan

0.039 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST	DISTRIBUTION
SELECT STATEMENT			1000	2	
PX COORDINATOR					
PX SEND	SYS_IQ10000	QC (RANDOM)	1000	2	QC (RANDOM)
PX BLOCK		ITERATOR	1000	2	
TABLE ACCESS	U_DW_DATA_02.CALENDAR	FULL	1000	2	
Other XML					
info type="derived_cpu_dop"	0				
info type="derived_io_dop"	0				
info type="dop_reason" note="y"	table property				
info type="dop" note="y"	4				

Look at the SQL history of query with Parallel Hint.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST	DISTRIBUTION
SQL History					
SQL					
<code>SELECT /*+ PARALLEL(calendar_d) */ 'Y' FROM calendar;</code>					
Re[D:/Vera/Database_2022]M10.Basic:Partial Execution[Lab 10.sql]	VeraDB-port...,	28-JUL-22 11:10:00	SQL	1	0.014
select * from calendar;	VeraDB-port...,	28-JUL-22 11:10:00	Script	23	0.0
select * from calendar;	VeraDB-port...,	28-JUL-22 11:10:00	Script	3	0.019
select TRUNC(sj.m) into _me_id, TO_CHAR(sj.m, 'fmDay') day_,	VeraDB-port...,	28-JUL-22 11:10:00	SQL	2	0.122
Re[D:/Vera/Database_2022]Lab[Lab_7/report/scripts/lead1/Calendar]	VeraDB-port...,	28-JUL-22 11:10:00	Script	1	0.0
select * from USR\$Tables;	VeraDB-port...,	28-JUL-22 11:10:00	Script	2	0.01
Re[D:/Vera/Database_2022]Lab[Lab_8/scripts/TABLES/_FCT_bean...	VeraDB-port...,	28-JUL-22 11:10:00	Script	6	0.0
Re[D:/Vera/Database_2022]Lab[Lab_8/scripts/TABLES/_DM_prom...	VeraDB-port...,	27-JUL-22 11:10:00	Script	1	0.0
Re[D:/Vera/Database_2022]Lab[Lab_8/scripts/TABLES/_DM_produ...	VeraDB-port...,	27-JUL-22 11:10:00	Script	1	0.0
Re[D:/Vera/Database_2022]Lab[Lab_8/scripts/TABLES/_DM_gm...	VeraDB-port...,	27-JUL-22 11:10:00	Script	1	0.0
Re[D:/Vera/Database_2022]Lab[Lab_8/scripts/TABLES/_DM_agenc...	VeraDB-port...,	27-JUL-22 11:10:00	Script	1	0.0
Re[D:/Vera/Database_2022]Lab[Lab_8/scripts/TABLES/_DM_custo...	VeraDB-port...,	27-JUL-22 11:10:00	Script	2	0.0
Re[D:/Vera/Database_2022]Lab[Lab_8/scripts/TABLES/_customer_sg	VeraDB-port...,	27-JUL-22 11:10:00	Script	2	0.0
Re[D:/Vera/Database_2022]Lab[Lab_8/scripts/Storage level SA/...	VeraDB-port...,	27-JUL-22 11:10:00	Script	1	0.0
Re[D:/Vera/Database_2022]Lab[Lab_8/scripts/Storage level SA/...	VeraDB-port...,	27-JUL-22 11:10:00	Script	1	0.0
Re[D:/Vera/Database_2022]Lab[Lab_8/scripts/Storage level SA/...	VeraDB-port...,	27-JUL-22 11:10:00	Script	4	0.0
Re[D:/Vera/Database_2022]Lab[Lab_8/scripts/Storage level SA/...	VeraDB-port...,	27-JUL-22 11:10:00	Script	1	0.0
Re[D:/Vera/Database_2022]Lab[Lab_8/scripts/Storage level SA/...	VeraDB-port...,	27-JUL-22 11:10:00	Script	2	0.0
Re[D:/Vera/Database_2022]Lab[Lab_8/scripts/Storage level SA/...	VeraDB-port...,	27-JUL-22 11:10:00	Script	5	0.0
Re[D:/Vera/Database_2022]Lab[Lab_8/scripts/storage_level_sa/_...	VeraDB-port...,	27-JUL-22 11:10:00	Script	1	0.0
select * from l_geo_systems;	VeraDB-port...,	26-JUL-22 11:10:00	Script	1	0.0
select * from l_geo_regions;	VeraDB-port...,	26-JUL-22 11:10:00	Script	2	0.01
select * from l_geo_params;	VeraDB-port...,	26-JUL-22 11:10:00	Script	2	0.015
select * from l_geo_params;	VeraDB-port...,	26-JUL-22 11:10:00	Script	2	0.012
select * from t_mst_object_link;	VeraDB-port...,	26-JUL-22 11:10:00	SQL	7	0.014

The time it takes to explain plan with Parallel Hint is 0.039 seconds. The time it takes to execute a request with Parallel Hint is 0.024 seconds. Cost is 2.

Create table Calendar on 10 000 rows and Insert values into the table.

Welcome Page × VeraDB-port1521 × Lab10.sgl ×

SQL Worksheet History

Worksheet Query Builder

```
TO_CHAR( sd + rn, 'YYYY' ) calendar_year,
( TO_DATE( '12/31/' || TO_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' )
- TRUNC( sd + rn, 'YEAR' ) ) days_in_cal_year,
TRUNC( sd + rn, 'YEAR' ) beg_of_cal_year,
TO_DATE( '12/31/' || TO_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' ) end_of_cal_year
FROM
(
  SELECT
    TO_DATE( '12/31/2011', 'MM/DD/YYYY' ) sd,
    rownum rn
  FROM dual
  CONNECT BY level <=10000
);
--SET TIMING ON;
--SET TIMING OFF;

SELECT * FROM calendar;
```

Script Output × Query Result ×

SQL | Fetched 2,250 rows in 0.676 seconds

TIME_ID	DAY_NAME	DAY_NUMBER_IN_WEEK	DAY_NUMBER_IN_MONTH	DAY_NUMBER_IN_YEAR	CALENDAR_WEEK_NUMBER	WEEK_ENDING_DATE	CALENDAR_MONTH_NUMBER	DAYS_IN_CAL_MONTH
1	03-JUL-13 Wednesday	3	03	184	1	07-JUL-13	07	31
2	04-JUL-13 Thursday	4	04	185	1	07-JUL-13	07	31
3	05-JUL-13 Friday	5	05	186	1	07-JUL-13	07	31
4	06-JUL-13 Saturday	6	06	187	1	07-JUL-13	07	31
5	07-JUL-13 Sunday	7	07	188	1	07-JUL-13	07	31
6	08-JUL-13 Monday	1	08	189	2	14-JUL-13	07	31
7	09-JUL-13 Tuesday	2	09	190	2	14-JUL-13	07	31
8	10-JUL-13 Wednesday	3	10	191	2	14-JUL-13	07	31
9	11-JUL-13 Thursday	4	11	192	2	14-JUL-13	07	31
10	12-JUL-13 Friday	5	12	193	2	14-JUL-13	07	31

Let's look at the explain plan.

The screenshot shows the VeraDB SQL Worksheet interface. The top tab bar has tabs for 'Welcome Page', 'VeraDB-port1521', and 'Lab10.sql'. The 'Worksheet' tab is active, displaying the following SQL code:

```

-- TRUNC( sd + rn, 'YEAR' ) ) days_in_cal_year,
TRUNC( sd + rn, 'YEAR' ) beg_of_cal_year,
TO_DATE( '12/31/* || TO_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' ) end_of_cal_year
FROM
(
  SELECT
    TO_DATE( '12/31/2011', 'MM/DD/YYYY' ) sd,
    rownum rn
  FROM dual
  CONNECT BY level <=10000
);

--SET TIMING ON;
--SET TIMING OFF;

SELECT /*+ FROM calendar;

SELECT /*+ PARALLEL(calendar,4) */ * FROM calendar;

```

Below the worksheet, the 'Script Output' tab is selected, showing the execution time: 0.032 seconds. The 'Explain Plan' tab is also visible. The explain plan table is as follows:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			10000	33
TABLE ACCESS	U_DW_DATA_02.CALENDAR	FULL	10000	33
Other XML				
{info}				
info type='has_user_tab'	yes			
info type='db_version'	21.0.0.0			
info type='parse_schema'	U_DW_DATA_02*			
info type='dynamic_sampling' note='y'				

The bottom status bar indicates 'Line 120 Column 24 | Insert | Modified | Windows: C'.

Look at the SQL history.

The screenshot shows the 'SQL History' tab in the VeraDB SQL Worksheet. It lists all the SQL statements executed, including their connection, timestamp, type, and duration. The history includes the execution of Lab10.sql and various other scripts related to the 'calendar' table and other database objects.

SQL	Connection	TimeStamp	Type	Executed	Duration(se...)
SELECT * FROM calendar;	VeraDB-port...	28-JUL-22 1...	SQL	1	0.019
file:D:/Vera/Datamola_2022/UM10.Basic Parallel Execution/Lab10.sql	VeraDB-port...	28-JUL-22 1...	Script	29	0.0
SELECT /*+ PARALLEL(calendar,4) */ * FROM calendar;	VeraDB-port...	28-JUL-22 1...	SQL	3	0.019
select * from calendar;	VeraDB-port...	28-JUL-22 1...	SQL	3	0.019
SELECT TRUNC(sd + rn) time_id, TO_CHAR(sd + rn, 'fmDay') day_...	VeraDB-port...	28-JUL-22 1...	SQL	2	0.122
file:D:/Vera/Datamola_2022/Labs/Lab_7/report/scripts_task01Calen...	VeraDB-port...	28-JUL-22 1...	Script	1	0.0
select * from USER_tablespaces;	VeraDB-port...	28-JUL-22 1...	SQL	2	0.01
file:D:/Vera/Datamola_2022/Labs/Lab_8/sqlscripts/TABLES/t_FCT_busin...	VeraDB-port...	27-JUL-22 1...	Script	6	0.0
file:D:/Vera/Datamola_2022/Labs/Lab_8/sqlscripts/TABLES/t_DIM_promo...	VeraDB-port...	27-JUL-22 1...	Script	1	0.0
file:D:/Vera/Datamola_2022/Labs/Lab_8/sqlscripts/TABLES/t_DIM_produ...	VeraDB-port...	27-JUL-22 1...	Script	1	0.0
file:D:/Vera/Datamola_2022/Labs/Lab_8/sqlscripts/TABLES/t_DIM_gen_...	VeraDB-port...	27-JUL-22 1...	Script	1	0.0
file:D:/Vera/Datamola_2022/Labs/Lab_8/sqlscripts/TABLES/t_DIM_agenc...	VeraDB-port...	27-JUL-22 1...	Script	1	0.0
file:D:/Vera/Datamola_2022/Labs/Lab_8/sqlscripts/TABLES/t_DIM_emplo...	VeraDB-port...	27-JUL-22 1...	Script	2	0.0
file:D:/Vera/Datamola_2022/Labs/Lab_8/sqlscripts/TABLES/t_DIM_custo...	VeraDB-port...	27-JUL-22 1...	Script	2	0.0
file:D:/Vera/Datamola_2022/Labs/Lab_8/sqlscripts/TABLES/t_customer.sq...	VeraDB-port...	27-JUL-22 1...	Script	2	0.0
file:D:/Vera/Datamola_2022/Labs/Lab_8/sqlscripts/Storage level SA/dm...	VeraDB-port...	27-JUL-22 1...	Script	1	0.0
file:D:/Vera/Datamola_2022/Labs/Lab_8/sqlscripts/Storage level SA/dm...	VeraDB-port...	27-JUL-22 1...	Script	1	0.0
file:D:/Vera/Datamola_2022/Labs/Lab_8/sqlscripts/Storage level SA/dm...	VeraDB-port...	27-JUL-22 1...	Script	4	0.0
file:D:/Vera/Datamola_2022/Labs/Lab_8/sqlscripts/Storage level SA/dm...	VeraDB-port...	27-JUL-22 1...	Script	1	0.0
file:D:/Vera/Datamola_2022/Labs/Lab_8/sqlscripts/Storage level SA/dm...	VeraDB-port...	27-JUL-22 1...	Script	5	0.0
file:D:/Vera/Datamola_2022/Labs/Lab_8/sqlscripts/Storage level SA/ts_s...	VeraDB	27-JUL-22 0...	Script	1	0.0
file:D:/Vera/Datamola_2022/Labs/Lab_7/report/scripts_task01_u_dw...	VeraDB	26-JUL-22 1...	Script	1	0.0
select * from lc_geo_systems;	VeraDB	26-JUL-22 1...	SQL	2	0.01
select * from lc_geo_regions;	VeraDB	26-JUL-22 1...	SQL	2	0.015
select * from t_geo_parts;	VeraDB	26-JUL-22 1...	SQL	2	0.012
select * from t_neo_object_links;	VeraDB	26-JUL-22 1...	SQL	2	0.014

The time it takes to explain plan is 0.032 seconds. The time it takes to execute a request is 0.019 seconds. Cost is 33.

Use Parallel execution:

The screenshot shows the Oracle SQL Developer interface. In the Worksheet tab, the following SQL code is executed:

```

FROM
(
  SELECT
    TO_DATE( '12/31/2011', 'MM/DD/YYYY' ) sd,
    rownum rn
  FROM dual
  CONNECT BY level <=10000
);

--SET TIMING ON;
--SET TIMING OFF;

SELECT * FROM calendar;

SELECT /*+ PARALLEL(calendar,4) */ * FROM calendar;

```

In the Query Result tab, the output is a table with 10000 rows, showing various date-related columns like TIME_ID, DAY_NAME, and CALENDAR_WEEK_NUMBER.

Let's look at the explain plan with Parallel Hint.

The screenshot shows the Oracle SQL Developer interface with the Explain Plan tab selected. The explain plan details the execution plan for the parallel query:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST	DISTRIBUTION
SELECT STATEMENT			10000	9	
PX COORDINATOR					
PX SEND	SYS.TQ10000	QC (RANDOM)	10000	9 QC (RANDOM)	
PX BLOCK		ITERATOR	10000	9	
TABLE ACCESS	U_DW_DATA_02.CALENDAR	FULL	10000	9	

Other XML details include derived CPU DOP (0), derived IO DOP (0), and DOP reasons (note=y).

Look at the SQL history of query with Parallel Hint.

The screenshot shows the VeraDB SQL Worksheet interface. The title bar says "Welcome Page" and "VeraDB-port1521". The tab bar has "Lab 10.sql" selected. The main area shows a worksheet with a "Query Builder" tab open. Below it is a "Script Output" tab showing the executed SQL code, which includes a parallel hint (*+ PARALLEL(calendar, 4)). The status bar indicates "0.016 seconds". A toolbar with various icons is visible above the tabs. The bottom part of the window is a grid table titled "SQL History" with columns: OPERATION, OBJECT_NAME, OPTIONS, CARDINALITY, COST, and DISTRIBUTION. The table lists numerous database operations, mostly scripts, with their details like connection, timestamp, type, and duration.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST	DISTRIBUTION
SQL			0.016 seconds		
SQL History					
SQL					
SELECT /*+ PARALLEL(calendar, 4) */ * FROM calendar;	VeraDB-port...	28-JUL-22 1...	SQL	4	0.027
file:/D:/Vera//Datamola_2022/U1M10.Basic Parallel Execution/Lab10.sql	VeraDB-port...	28-JUL-22 1...	Script	34	0.0
SELECT * FROM calendar;	VeraDB-port...	28-JUL-22 1...	SQL	3	0.007
select * from calendar;	VeraDB-port...	28-JUL-22 1...	SQL	3	0.019
SELECT TRUNC(sd + rn) time_id, TO_CHAR(sd + rn, 'fmDay') day_...	VeraDB-port...	28-JUL-22 1...	SQL	2	0.122
file:/D:/Vera//Datamola_2022/Labs/Lab_7/report/scripts_task01/Calen...	VeraDB-port...	28-JUL-22 1...	Script	1	0.0
select * from USER_TABLESPACES;	VeraDB-port...	28-JUL-22 1...	SQL	2	0.01
file:/D:/Vera//Datamola_2022/Labs/Lab_8/scripts/TABLES/t_FCT_busu...	VeraDB-port...	27-JUL-22 1...	Script	6	0.0
file:/D:/Vera//Datamola_2022/Labs/Lab_8/scripts/TABLES/t_DIM_prom...	VeraDB-port...	27-JUL-22 1...	Script	1	0.0
file:/D:/Vera//Datamola_2022/Labs/Lab_8/scripts/TABLES/t_DIM_produ...	VeraDB-port...	27-JUL-22 1...	Script	1	0.0
file:/D:/Vera//Datamola_2022/Labs/Lab_8/scripts/TABLES/t_DIM_gen...	VeraDB-port...	27-JUL-22 1...	Script	1	0.0
file:/D:/Vera//Datamola_2022/Labs/Lab_8/scripts/TABLES/t_DIM_agenc...	VeraDB-port...	27-JUL-22 1...	Script	1	0.0
file:/D:/Vera//Datamola_2022/Labs/Lab_8/scripts/TABLES/t_DIM_empl...	VeraDB-port...	27-JUL-22 1...	Script	2	0.0
file:/D:/Vera//Datamola_2022/Labs/Lab_8/scripts/TABLES/t_DIM_custo...	VeraDB-port...	27-JUL-22 1...	Script	2	0.0
file:/D:/Vera//Datamola_2022/Labs/Lab_8/scripts/Storage_level_SA/dm...	VeraDB-port...	27-JUL-22 1...	Script	1	0.0
file:/D:/Vera//Datamola_2022/Labs/Lab_8/scripts/Storage_level_SA/dm...	VeraDB-port...	27-JUL-22 1...	Script	1	0.0
file:/D:/Vera//Datamola_2022/Labs/Lab_8/scripts/Storage_level_SA/dm...	VeraDB-port...	27-JUL-22 1...	Script	4	0.0
file:/D:/Vera//Datamola_2022/Labs/Lab_8/scripts/Storage_level_SA/dm...	VeraDB-port...	27-JUL-22 1...	Script	1	0.0
file:/D:/Vera//Datamola_2022/Labs/Lab_8/scripts/Storage_level_SA/dm...	VeraDB-port...	27-JUL-22 1...	Script	5	0.0
file:/D:/Vera//Datamola_2022/Labs/Lab_8/scripts/Storage_level_SA/ts_s...	VeraDB	27-JUL-22 0...	Script	1	0.0
file:/D:/Vera//Datamola_2022/Labs/Lab_7/report/scripts_task01/u_dw...	VeraDB	26-JUL-22 1...	Script	1	0.0
select * from t_geo_systems;	VeraDB	26-JUL-22 1...	SQL	2	0.01
select * from t_geo_regions;	VeraDB	26-JUL-22 1...	SQL	2	0.015
select * from t_geo_parts;	VeraDB	26-JUL-22 1...	SQL	2	0.012
select * from t_geo_object_links;	VeraDB	26-JUL-22 1...	SQL	2	0.014

Summary:

	Row Count	Time to Execute Plan	Time to request	Cost
Simple SELECT	1000	0.058 seconds	0.019 seconds	5
SELECT with Parallel Hint	1000	0.039 seconds	0.024 seconds	2
Simple SELECT	10 000	0.032 seconds	0.019 seconds	33
SELECT with Parallel Hint	10 000	0.016 seconds	0.027 seconds	9

As we can see, when using Select Parallel execution, the cost becomes less, but the execution time on the query increases compared to a simple query.

2.2. Task 02: CREATE Example of Parallel DML

Create one table on 1000 rows from another by adding a SELECT statement at the end of the CREATE TABLE statement without parallelization.

The screenshot shows the Oracle SQL Developer interface with the 'Worksheet' tab selected. The code in the worksheet window is as follows:

```
20: --drop table calendar;
21:
22: alter session set nls_territory = "UNITED KINGDOM";
23:
24: CREATE TABLE calendar
25: AS SELECT * FROM (
26:   SELECT
27:     TRUNC( sd + rn ) time_id,
28:     TO_CHAR( sd + rn, 'fmDay' ) day_name,
29:     TO_CHAR( sd + rn, 'D' ) day_number_in_week,
30:     TO_CHAR( sd + rn, 'DD' ) day_number_in_month,
31:     TO_CHAR( sd + rn, 'DDD' ) day_number_in_year,
32:     TO_CHAR( sd + rn, 'W' ) calendar_week_number,
33:     ( CASE
34:       WHEN TO_CHAR( sd + rn, 'D' ) IN ( 1, 2, 3, 4, 5, 6 ) THEN
35:         NEXT_DAY( sd + rn, 'SUNDAY' )
36:       ELSE
37:         ( sd + rn )
38:       END ) week_ending_date,
39:     TO_CHAR( sd + rn, 'MM' ) calendar_month_number,
40:     TO_CHAR( LAST_DAY( sd + rn ), 'DD' ) days_in_cal_month,
41:     LAST_DAY( sd + rn ) end_of_cal_month,
42:     TO_CHAR( sd + rn, 'FMMonth' ) calendar_month_name,
43:     ( ( CASE
44:       WHEN TO_CHAR( sd + rn, 'Q' ) = 1 THEN
45:         TO_DATE( '03/31/' || TO_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' )
```

The script output window below shows the results of the execution:

```
Session altered.

Table CALENDAR created.
```

The screenshot shows the Oracle SQL Developer interface with the 'Worksheet' tab selected. The code in the worksheet window is as follows:

```
TO_DATE( '12/31/' || TO_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' )
END ) end_of_cal_quarter,
TO_CHAR( sd + rn, 'Q' ) calendar_quarter_number,
TO_CHAR( sd + rn, 'YYYY' ) calendar_year,
( TO_DATE( '12/31/' || TO_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' )
- TRUNC( sd + rn, 'YEAR' ) ) days_in_cal_year,
TRUNC( sd + rn, 'YEAR' ) beg_of_cal_year,
TO_DATE( '12/31/' || TO_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' ) end_of_cal_year
FROM
(
  SELECT
    TO_DATE( '12/31/2011', 'MM/DD/YYYY' ) sd,
    rownum rn
  FROM dual
  CONNECT BY level <=1000
);
```

The script output window below shows the results of the execution:

```
Session altered.

Table CALENDAR created.

1,000 rows inserted.
```

Let's delete the data without parallelization:

The screenshot shows a script execution window in Oracle SQL Developer. The script contains three commands: `SET TIMING ON;`, `DELETE FROM calendar WHERE CALENDAR_MONTH_NAME IN ('June','July','August');`, and `SET TIMING OFF;`. The output panel below shows "Task completed in 0.028 seconds", "276 rows deleted.", and "Elapsed: 00:00:00.016".

```
SET TIMING ON;
DELETE FROM calendar WHERE CALENDAR_MONTH_NAME IN ('June','July','August');
SET TIMING OFF;

Script Output x
Task completed in 0.028 seconds

276 rows deleted.

Elapsed: 00:00:00.016
```

Let's look at the explain plan.

The screenshot shows two windows. The top window displays the execution of an explain plan for the same delete statement. It shows the creation and alteration of a temporary table, and the execution of the explain plan command. The bottom window shows the resulting explain plan output in a tabular format, detailing the execution plan steps and their characteristics.

```
explain plan for
DELETE FROM calendar WHERE CALENDAR_MONTH_NAME IN ('June','July','August');

select * from table (dbms_xplan.display);

Script Output x
Task completed in 0.025 seconds

Table CALENDAR dropped.

Session altered.

Table CALENDAR created.

1,000 rows inserted.

Explained.

explain plan for
DELETE FROM calendar WHERE CALENDAR_MONTH_NAME IN ('June','July','August');

select * from table (dbms_xplan.display);

Script Output x | Query Result x
SQL | All Rows Fetched: 19 in 0.104 seconds
PLAN_TABLE_OUTPUT
1 Plan hash value: 2886354060
2
3 -----
4 | Id | Operation           | Name      | Rows   | Bytes | Cost (%CPU)| Time     |
5 -----
6 | 0 | DELETE STATEMENT    |          | 276    | 4968  |       5  (0) | 00:00:01  |
7 | 1 |  DELETE              | CALENDAR  |        |        |       5  (0) | 00:00:01  |
8 |* 2 |  TABLE ACCESS FULL  | CALENDAR  | 276    | 4968  |       5  (0) | 00:00:01  |
9 -----
10
11 Predicate Information (identified by operation id):
12 -----
13
14   2 - filter("CALENDAR_MONTH_NAME"='August' OR
15                 "CALENDAR_MONTH_NAME"='July' OR "CALENDAR_MONTH_NAME"='June')
16
17 Note
```

```

--SET TIMING OFF;
--SET autotrace ON;
--DELETE FROM calendar WHERE CALENDAR_MONTH_NAME IN ('June','July','August');

explain plan for
DELETE FROM calendar WHERE CALENDAR_MONTH_NAME IN ('June','July','August');

select * from table (dbms_xplan.display);

```

The Explain Plan details the execution plan for the DELETE statement. It shows a parallel delete operation with four parallel query blocks (PQ). The plan includes a full table access on the CALENDAR table. The predicates indicate that the rows being deleted have CALENDAR_MONTH_NAME values of 'August', 'July', or 'June'.

Consider table Calendar on 1000 rows.

Let's delete the data with parallelization:

```

SET TIMING ON;
DELETE /*+ PARALLEL(calendar,4) */ FROM calendar WHERE CALENDAR_MONTH_NAME IN ('June','July','August');
SET TIMING OFF;

```

Script Output: Task completed in 0.038 seconds

276 rows deleted.

Elapsed: 00:00:00.030

Let's look at the explain plan.

```

explain plan for
DELETE /*+ PARALLEL(calendar,4) */ FROM calendar WHERE CALENDAR_MONTH_NAME IN ('June','July','August');

select * from table (dbms_xplan.display);

```

The Explain Plan details the execution plan for the parallel delete statement. It shows a parallel delete operation with four parallel query blocks (PQ) and a full table access on the CALENDAR table. The predicates indicate that the rows being deleted have CALENDAR_MONTH_NAME values of 'August', 'July', or 'June'.

```

explain plan for
SELECT /*+ PARALLEL(calendar,4) */ FROM calendar WHERE CALENDAR_MONTH_NAME IN ('June','July','August');
select * from table (dbms_xplan.display);

```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST	DISTRIBUTION
DELETE STATEMENT				276	S
DELETE				276	S
PX COORDINATOR				276	SQC (RANDOM)
PX SEND	SYS_IQTQ10000	QC (RANDOM)		276	SQC (RANDOM)
PX BLOCK		ITERATOR		276	S
TABLE ACCESS	U_DW_DATA_02_CALENDAR	FULL		276	S
Filter Predicates					
OR					
CALENDAR_MONTH_NAME = 'August'					
CALENDAR_MONTH_NAME = 'July'					
CALENDAR_MONTH_NAME = 'June'					
Other XML					
info type='derived_cpu_dop'					
0					
info type='derived_io_dop'					
0					
info type='dop_reason' note='y'					
table prop					
info type='stop' note='y'					
4					
info type='px_in_memory_mc' note='y'					
no					
info type='px_in_memory' note='y'					
no					

Create table Calendar on 10 000 rows and Insert values into the table.

```

TO_DATE( '12/31/2011', 'MM/DD/YYYY' ) sd,
rownum rn
FROM dual
CONNECT BY level <=10000
));

```

Session altered.

Table CALENDAR created.

10,000 rows inserted.

Let's delete the data without parallelization:

The screenshot shows the SQL Developer interface. In the top script editor window, the following SQL code is run:

```
SET TIMING ON;
DELETE FROM calendar WHERE CALENDAR_MONTH_NAME IN ('June','July','August');
SET TIMING OFF;
```

The output window below shows the results of the execution:

```
Table CALENDAR dropped.

Session altered.

Table CALENDAR created.

10,000 rows inserted.

2,484 rows deleted.

Elapsed: 00:00:00.021
```

Let's look at the explain plan.

The screenshot shows the SQL Developer interface. In the top script editor window, the following EXPLAIN PLAN command is run:

```
explain plan for
DELETE FROM calendar WHERE CALENDAR_MONTH_NAME IN ('June','July','August');

select * from table (dbms_xplan.display);
```

The output window below displays the explain plan results:

PLAN_TABLE_OUTPUT
3 -----
4 Id Operation Name Rows Bytes Cost (%CPU) Time
5 -----
6 0 DELETE STATEMENT 2484 44712 33 (0) 00:00:01
7 1 DELETE CALENDAR
8 * 2 TABLE ACCESS FULL CALENDAR 2484 44712 33 (0) 00:00:01
9 -----
10
11 Predicate Information (identified by operation id):
12 -----
13
14 2 - filter("CALENDAR_MONTH_NAME"='August' OR
15 "CALENDAR_MONTH_NAME"='July' OR "CALENDAR_MONTH_NAME"='June')
16
17 Note
18 -----
19 - dynamic statistics used: dynamic sampling (level=2)

```

SET TIMING ON;
DELETE FROM calendar WHERE CALENDAR_MONTH_NAME IN ('June','July','August');

explain plan for
DELETE FROM calendar WHERE CALENDAR_MONTH_NAME IN ('June','July','August');

select * from table (dbms_xplan.display);

```

Script Output X | Query Result X | Explain Plan X | SQL | 0.015 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
0 1 DELETE STATEMENT			2484	33
0 2 TABLE ACCESS	U_DW_DATA_02_CALENDAR	FULL	2484	33
0 3 Filter Predicates				
0 4 OR				
0 4 1 CALENDAR_MONTH_NAME='August'				
0 4 2 CALENDAR_MONTH_NAME='July'				
0 4 3 CALENDAR_MONTH_NAME='June'				
0 Other XML (info)				
0 5 1 info type='has_user_tab'				
0 5 2 yes				
0 5 3 info type='db_version'				
0 5 4 21.0.0.0				
0 5 5 info type='parse_schema'				
0 5 6 'U_DW_DATA_02'				
0 5 7 info type='dynamic_sampling' note='y'				
0 5 8 2				
0 5 9 info type='plan_hash_full'				

Consider table Calendar on 10 000 rows with parallelization.

Let's delete the data with parallelization:

```

SET TIMING ON;
DELETE /*+ PARALLEL(calendar,4) */ FROM calendar WHERE CALENDAR_MONTH_NAME IN ('June','July','August');
SET TIMING OFF;

```

Script Output X | Task completed in 0.066 seconds

Session altered.

Table CALENDAR created.

10,000 rows inserted.

2,484 rows deleted.

Elapsed: 00:00:00.052

Let's look at the explain plan.

```

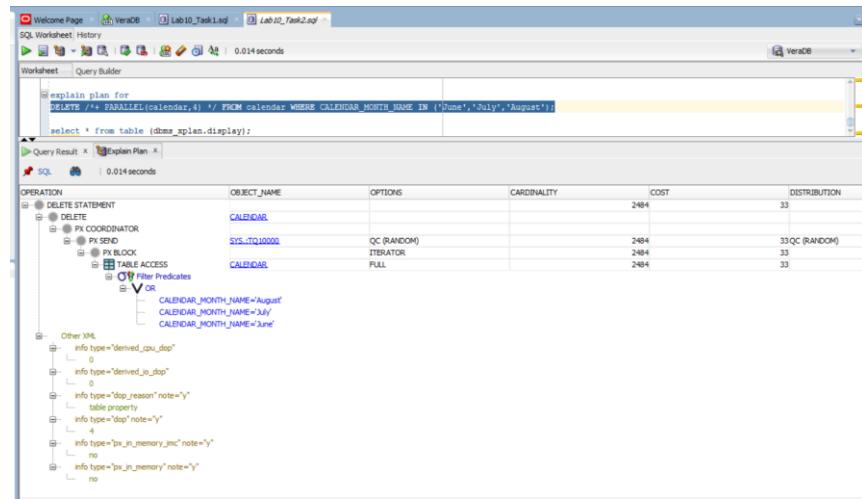
explain plan for
DELETE /*+ PARALLEL(calendar,4) */ FROM calendar WHERE CALENDAR_MONTH_NAME IN ('June','July','August');

select * from table (dbms_xplan.display);

```

Query Result X | SQL | All Rows Fetched: 24 in 0.039 seconds

PLAN_TABLE_OUTPUT								
3	-----	4 Id Operation	Name	Rows	Bytes	Cost (%CPU)	Time	TQ IN-OUT PQ Distrib
5	-----	6 0 DELETE STATEMENT		2484	44712	33 (0)	00:00:01	
7 1 DELETE	CALENDAR							
8 2 PX COORDINATOR								
9 3 PX SEND QC (RANDOM)	:TQ10000	2484	44712	33 (0)	00:00:01	Q1,00 P->S QC (RAND)		
10 4 PX BLOCK ITERATOR		2484	44712	33 (0)	00:00:01	Q1,00 PCWP		
11 * 5 TABLE ACCESS FULL	CALENDAR	2484	44712	33 (0)	00:00:01	Q1,00 PCWP		
12	-----	13						
14	Predicate Information (identified by operation id):							
15	-----							
16								
17	5 - filter("CALENDAR_MONTH_NAME"='August' OR "CALENDAR_MONTH_NAME"='July' OR							
18	"CALENDAR_MONTH_NAME"='June')							
19								
20	Note							
21	-----							
22	- dynamic statistics used: dynamic sampling (level=2)							
23	- Degree of Parallelism is 4 because of table property							
24	- FDMT is disabled in current session							



Summary:

	Number of rows deleted	Time to Execute Elapsed Time	Elapsed Time	Time to DELETE of Explain plane	Cost
Simple DELETE	276	0.028 seconds	00:00:00.016	00:00:01	5
DELETE with Parallel Hint	276	0.038 seconds	00:00:00.030	00:00:01	5
Simple DELETE	2484	0.041 seconds	00:00:00.021	00:00:01	33
DELETE with Parallel Hint	2484	0.066 seconds	00:00:00.052	00:00:01	33

Data Manipulation Language (DML) operations such as INSERT, UPDATE, and DELETE can be parallelized by Oracle. Parallel execution can speed up large DML operations and is particularly advantageous in data warehousing environments where it's necessary to maintain large summary or historical tables. In OLTP systems, parallel DML sometimes can be used to improve the performance of long-running batch jobs.

As you can see from the different options of job 02, when performing operations (DML) with Parallel Hint, the cost does not change, but it takes more time compared to normal operations without Parallel Hint.

It can be supposed that paralleling does not always improve results for small amounts of data.

2.3. Task 03: CREATE Example of Parallel DDL

Create one table on 1000 rows from another by adding a SELECT statement at the end of the CREATE TABLE statement without Parallel DDL.

The screenshot shows the Oracle SQL Worksheet interface. The top menu bar has tabs for 'Welcome Page', 'VeraDB', 'Lab 10_Task2.sql', and 'Lab 10_Task3.sql'. The main area is titled 'Worksheet' and contains the following SQL code:

```
15: --drop table calendar;
16:
17: alter session set nls_territory = "UNITED KINGDOM";
18:
19: CREATE TABLE calendar
20: AS SELECT * FROM (
21:   SELECT
22:     TRUNC( sd + rn ) time_id,
23:     TO_CHAR( sd + rn, 'fmDay' ) day_name,
24:     TO_CHAR( sd + rn, 'D' ) day_number_in_week,
25:     TO_CHAR( sd + rn, 'DD' ) day_number_in_month,
26:     TO_CHAR( sd + rn, 'DDD' ) day_number_in_year,
27:     TO_CHAR( sd + rn, 'W' ) calendar_week_number,
28:     ( CASE
29:         WHEN TO_CHAR( sd + rn, 'D' ) IN ( 1, 2, 3, 4, 5, 6 ) THEN
30:           NEXT_DAY( sd + rn, 'SUNDAY' )
31:         ELSE
32:           ( sd + rn )
33:         END ) week_ending_date,
34:     TO_CHAR( sd + rn, 'MM' ) calendar_month_number,
35:     TO_CHAR( LAST_DAY( sd + rn ), 'DD' ) days_in_cal_month,
36:     LAST_DAY( sd + rn ) end_of_cal_month,
37:     TO_CHAR( sd + rn, 'FMMonth' ) calendar_month_name,
38:     ( ( CASE
39:         WHEN TO_CHAR( sd + rn, 'Q' ) = 1 THEN
40:           TO_DATE( '03/31/' || TO_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' )
41:         ELSE
42:           TO_DATE( '12/31/' || TO_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' )
43:         END ) month_name
44:       )
45:   );
46:
47: 
```

The 'Script Output' pane below shows the results of the execution:

```
Session altered.

Table CALENDAR created.
```

Consider Explain plan.

The screenshot shows the Oracle SQL Worksheet interface with the 'Explain Plan' tab selected. The top menu bar has tabs for 'Welcome Page', 'VeraDB', 'Lab 10_Task2.sql', and 'Lab 10_Task3.sql'. The main area is titled 'Worksheet' and contains the same SQL code as the previous screenshot. The 'Explain Plan' pane below shows the execution plan for the CREATE TABLE statement:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
CREATE TABLE STATEMENT	CALENDAR		1	4
LOAD AS SELECT			1	3
OPTIMIZER STATISTICS GATHERING			1	3
VIEW			1	3
COUNT				
CONNECT BY		WITHOUT FILTERING		
Filter Predicates				
LEVEL <=1000				
FAST DUAL			1	3
Other XML				
{info}				

```
97 select * from table (dbms_xplan.display);

Script Output X Query Result X
SQL | All Rows Fetched: 18 in 0.043 seconds

PLAN_TABLE_OUTPUT
1 Plan hash value: 1220224350
2
3 -
4 | Id  | Operation          | Name    | Rows  | Bytes | Cost (%CPU) | Time      |
5 -
6 | 0  | CREATE TABLE STATEMENT |         |       |       |        | 00:00:01 |
7 | 1  | LOAD AS SELECT     | CALENDAR|       |       |        |           |
8 | 2  | OPTIMIZER STATISTICS GATHERING |         |       |       |        | 00:00:01 |
9 | 3  | VIEW                |         |       |       |        | 00:00:01 |
10 | 4  | COUNT               |         |       |       |        |           |
11 |* 5  | CONNECT BY WITHOUT FILTERING|         |       |       |        |           |
12 | 6  | FAST DUAL            |         |       |       |        | 00:00:01 |
13 -
14
15 Predicate Information (identified by operation id):
16 -
17
18   5 - filter(T.LEVEL<=1000)
```

Look on the Elapsed time.

```
SQL Worksheet History
Welcome Page VeraDB Lab10_Task2.sql Lab10_Task3.sql

Worksheet Query Builder
25 SET TIMING ON;
26 CREATE TABLE calendar
27 AS SELECT * FROM (
28   SELECT
29     TRUNC( sd + rn ) time_id,
30     TO_CHAR( sd + rn, 'fmDay' ) day_name,
31     TO_CHAR( sd + rn, 'D' ) day_number_in_week,
32     TO_CHAR( sd + rn, 'DD' ) day_number_in_month,
33     TO_CHAR( sd + rn, 'DDD' ) day_number_in_year,
34     TO_CHAR( sd + rn, 'W' ) calendar_week_number,
35   FROM
36     dual
37   ) t;
```

Script Output X

SQL | Task completed in 0.102 seconds

Table CALENDAR created.

Elapsed: 00:00:00.088

With DDL statements, we do not specify the degree of parallelism using hints. Instead, we use the PARALLEL clause.

Create one table on 1000 rows from another by adding a SELECT statement at the end of the CREATE TABLE statement with Parallel DDL.

Consider Explain plan.

The screenshot shows the Oracle SQL Developer interface with the Explain Plan tab selected. The Explain Plan details the execution plan for a Parallel DDL statement:

```

CREATE TABLE calendar PARALLEL 4
AS SELECT * FROM (
  SELECT
    TRUNC( sd + rn ) time_id,
    TO_CHAR( sd + rn, 'fmDay' ) day_name,
    TO_CHAR( sd + rn, 'D' ) day_number_in_week,
    TO_CHAR( sd + rn, 'DD' ) day_number_in_month,
    TO_CHAR( sd + rn, 'DDD' ) day_number_in_year,
    TO_CHAR( sd + rn, 'W' ) calendar_week_number,
    ( CASE
        WHEN TO_CHAR( sd + rn, 'D' ) IN ( 1, 2, 3, 4, 5, 6 ) THEN
          NEXT_DAY( sd + rn, 'SUNDAY' )
        ELSE
          NULL
      END )
  FROM
    ( SELECT
        TRUNC( sysdate - 1000 ) + rownum - 1 sd,
        mod( rownum, 7 ) rn
      FROM
        dual
      CONNECT BY
        LEVEL <= 1000
    )
)
  
```

The Explain Plan table shows the following operations and their properties:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST	DISTRIBUTION
CREATE TABLE STATEMENT			1	2	
PX COORDINATOR		SYS.:TQ10001	QC (RANDOM)	1	2QC (RANDOM)
PX SEND			(HYBRID TSM/HWMB)	1	2
LOAD AS SELECT	CALENDAR			1	2
PX RECEIVE				1	2
PX SEND	SYS.:TQ10000	ROUND-ROBIN		1	2ROUND-ROBIN
VIEW				1	2
COUNT				1	2
CONNECT BY		WITHOUT FILTERING		1	2
Filter Predicates	LEVEL<=1000			1	2
FAST DUAL				1	2

Other XML information includes info type="derived_cpu_dop" with value 0.

The screenshot shows the Oracle SQL Developer interface with the Explain Plan tab selected. The Explain Plan displays the execution plan for a Parallel DDL statement using the dbms_xplan.display procedure:

```

select * from table (dbms_xplan.display);
  
```

The Explain Plan table shows the following operations and their properties:

PLAN_TABLE_OUTPUT
3 --
4 Id Operation Name Rows Bytes Cost (%CPU) Time TQ IN-OUT PQ Distrib
5 -----
6 0 CREATE TABLE STATEMENT 1 19 2 (0) 00:00:01
7 1 PX COORDINATOR
8 2 PX SEND QC (RANDOM) :TQ10001 1 19 2 (0) 00:00:01 Q1,01 P->S QC (RAND)
9 3 LOAD AS SELECT (HYBRID TSM/HWMB) CALENDAR
10 4 OPTIMIZER STATISTICS GATHERING 1 19 2 (0) 00:00:01 Q1,01 PCWP
11 5 PX RECEIVE
12 6 PX SEND ROUND-ROBIN :TQ10000 1 19 2 (0) 00:00:01 S->P RND-ROBIN
13 7 VIEW 1 19 2 (0) 00:00:01
14 8 COUNT
15 * 9 CONNECT BY WITHOUT FILTERING
16 10 FAST DUAL 1 1 2 (0) 00:00:01
17 -----
18
19 Predicate Information (identified by operation id):
20 -----
21
22 9 - filter(LEVEL<=1000)
23
24 Note
25 -----
26 - Degree of Parallelism is 4 because of table property

Look on the Elapsed time with Parallel DDL.

The screenshot shows the VeraDB SQL Worksheet interface. The title bar has tabs for "Welcome Page", "VeraDB", "Lab10_Task2.sql", and "Lab10_Task3.sql". The main area is titled "Worksheet" and contains the following SQL code:

```
SET TIMING ON;
CREATE TABLE calendar PARALLEL 4
AS SELECT * FROM (
SELECT
    TRUNC( sd + rn ) time_id,
    TO_CHAR( sd + rn, 'fmDay' ) day_name,
    TO_CHAR( sd + rn, 'D' ) day_number_in_week,
    TO_CHAR( sd + rn, 'DD' ) day_number_in_month,
    TO_CHAR( sd + rn, 'DDD' ) day_number_in_year,
    TO_CHAR( sd + rn, 'W' ) calendar_week_number,
    ( CASE
        WHEN TO_CHAR( sd + rn, 'D' ) IN ( 1, 2, 3, 4, 5, 6 ) THEN
            NEXT DAY( sd + rn, 'SUNDAY' )
    END ) day_number_in_year,
    TO_CHAR( sd + rn, 'fmMonth fmYear' ) date_time
FROM
    ( SELECT
        TRUNC( sysdate ) - MOD( TRUNC( sysdate ), 7 ) + 1 + rn
        AS sd
        , rn
        , mod( rn, 7 ) + 1
        AS rn
    FROM
        dual
    CONNECT BY
        rn <= 7 * 52
    ) t1
) t2;
```

The "Script Output" tab is selected, showing the message "Table CALENDAR created." and an elapsed time of "Elapsed: 00:00:00.079".

Let's check the same operations for a table of 10,000 rows.

The screenshot shows the VeraDB SQL Worksheet interface. The top menu bar includes tabs for 'Welcome Page', 'VeraDB', 'Lab 10_Task2.sql', and 'Lab 10_Task3.sql'. The main window has a toolbar with icons for file operations, search, and help. A status bar at the bottom indicates '0.053 seconds'.

Worksheet | **Query Builder**

```
25 --SET TIMING ON;
26 CREATE TABLE calendar
27 AS SELECT * FROM (
28 SELECT
29   TRUNC( sd + rn ) time_id,
30   TO_CHAR( sd + rn, 'fmDay' ) day_name,
31   TO_CHAR( sd + rn, 'D' ) day_number_in_week,
32   TO_CHAR( sd + rn, 'DD' ) day_number_in_month,
33   TO_CHAR( sd + rn, 'DDD' ) day_number_in_year,
34   TO_CHAR( sd + rn, 'W' ) calendar_week_number,
35   ( CASE
36     WHEN TO_CHAR( sd + rn, 'D' ) IN ( 1, 2, 3, 4, 5, 6 ) THEN
37       NEXT_DAY( sd + rn, 'SUNDAY' )
38     ELSE
39   
```

Query Result | **Script Output** | **Explain Plan**

SQL | **0.053 seconds**

Explain Plan

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
CREATE TABLE STATEMENT				
LOAD AS SELECT			1	4
OPTIMIZER STATISTICS GATHERING	CALENDAR		1	3
VIEW			1	3
COUNT				
CONNECT BY		WITHOUT FILTERING		
Filter Predicates				
LEVEL <= 10000				
FAST DUAL			1	3
Other XML	{info}			
info type="db_version"	21.0.0.0			
info type="parse_schema"	VSHKRABATOVSKAYA			
info type="plan_hash_full"	1987848866			

```

81 | select * from table (dbms_xplan.display);
82 |
Script Output x Query Result x
SQL | All Rows Fetched: 18 in 0.036 seconds
PLAN_TABLE_OUTPUT
1 Plan hash value: 1220224350
2
3 -
4 | Id  | Operation          | Name   | Rows  | Bytes | Cost (%CPU) | Time   |
5 -
6 | 0  | CREATE TABLE STATEMENT |        | 1    | 19   |    4  (0) | 00:00:01 |
7 | 1  | LOAD AS SELECT      | CALENDAR | 1    | 19   |    3  (0) | 00:00:01 |
8 | 2  | OPTIMIZER STATISTICS GATHERING |        | 1    | 19   |    3  (0) | 00:00:01 |
9 | 3  | VIEW                |        | 1    | 19   |    3  (0) | 00:00:01 |
10 | 4  | COUNT               |        | 1    |       |        |           |
11 |* 5  | CONNECT BY WITHOUT FILTERING |        | 1    |       |        |           |
12 | 6  | FAST DUAL            |        | 1    |       |    3  (0) | 00:00:01 |
13 -
14
15 Predicate Information (identified by operation id):
16 -
17
18   5 - filter(LEVEL<=10000)

```

Look on the Elapsed time.

The screenshot shows the Oracle SQL Developer interface. The top navigation bar has tabs for 'Welcome Page', 'VeraDB', 'Lab10_Task2.sql', and 'Lab10_Task3.sql'. The active tab is 'Worksheet'.

The code editor window displays the following SQL script:

```

25 SET TIMING ON;
26 CREATE TABLE calendar
27 AS SELECT * FROM (
28   SELECT
29     TRUNC( sd + rn ) time_id,
30     TO_CHAR( sd + rn, 'fmDay' ) day_name,
31     TO_CHAR( sd + rn, 'D' ) day_number_in_week,
32     TO_CHAR( sd + rn, 'DD' ) day_number_in_month,
33     TO_CHAR( sd + rn, 'DDD' ) day_number_in_year,

```

The 'Script Output' window at the bottom shows the results of the execution:

```

Table CALENDAR created.

Elapsed: 00:00:00.330

```

Create one table on 10 000 rows from another by adding a SELECT statement at the end of the CREATE TABLE statement with Parallel DDL.

The screenshot shows the SQL Worksheet interface with several tabs at the top: Welcome Page, VeraDB, Lab10_Task2.sql, and Lab10_Task3.sql. The Lab10_Task3.sql tab is active. The main area displays the following SQL code:

```
91 CREATE TABLE calendar PARALLEL 4
92 AS SELECT * FROM (
93   SELECT
94     TRUNC( sd + rn ) time_id,
95     TO_CHAR( sd + rn, 'fmDay' ) day_name,
96     TO_CHAR( sd + rn, 'D' ) day_number_in_week,
97     TO_CHAR( sd + rn, 'DD' ) day_number_in_month,
98     TO_CHAR( sd + rn, 'DDD' ) day_number_in_year,
99     TO_CHAR( sd + rn, 'W' ) calendar_week_number,
```

Below the code, the Script Output pane shows the message "Table CALENDAR created." and a completion time of "Task completed in 0.159 seconds".

Consider Explain plan.

The screenshot shows the SQL Worksheet interface with the Explain Plan tab selected. The Explain Plan details the execution steps for the parallel table creation:

- CREATE TABLE STATEMENT
- PX COORDINATOR
 - PX SEND
 - LOAD AS SELECT
 - OPTIMIZER STATISTICS GATHERING
 - PX RECEIVE
 - PX SEND
 - VIEW
 - COUNT
 - CONNECT BY
 - Filter Predicates
 - LEVEL <= 10000
- Other XML (info)
 - info type="derived_cpu_dop" 0
 - info type="derived_io_dop" 0
 - info type="dop_reason" note="y"

Look on the Elapsed time.

The screenshot shows the Oracle SQL Developer interface. The top tab bar has tabs for 'Welcome Page', 'VeraDB', 'Lab10_Task2.sql', and 'Lab10_Task3.sql'. The 'Worksheet' tab is selected. The code in the worksheet window is:

```
142 FROM dual
143   CONNECT BY level <=10000
144 );
145 SET TIMING OFF;
146
147 select * from table (dbms_xplan.display);
```

Below the code, the 'Explain Plan' tab is selected. The output shows the execution plan for the query:

| Plan hash value: 2348843576 |
|---|
| 2 |
| 3 ----- |
| 4 Id Operation Name Rows Bytes Cost (%CPU) Time TQ IN-OUT PQ Distrib |
| 5 ----- |
| 6 0 CREATE TABLE STATEMENT 1 19 2 (0) 00:00:01 |
| 7 1 PX COORDINATOR |
| 8 2 PX SEND QC (RANDOM) :TQ10001 1 19 2 (0) 00:00:01 Q1,01 P->S QC (RAND) |
| 9 3 LOAD AS SELECT (HYBRID TSM/HWMB) CALENDAR |
| 10 4 OPTIMIZER STATISTICS GATHERING 1 19 2 (0) 00:00:01 Q1,01 PCWP |
| 11 5 PX RECEIVE |
| 12 6 PX SEND ROUND-ROBIN :TQ10000 1 19 2 (0) 00:00:01 S->P RND-ROBIN |
| 13 7 VIEW |
| 14 8 COUNT |
| 15 /* 9 CONNECT BY WITHOUT FILTERING |
| 16 10 FAST DUAL 1 2 (0) 00:00:01 |
| 17 ----- |
| 18 |
| 19 Predicate Information (identified by operation id): |
| 20 ----- |
| 21 |

The 'Script Output' tab is also visible at the bottom of the window.

The screenshot shows the Oracle SQL Developer interface. The top tab bar has tabs for 'Welcome Page', 'VeraDB', 'Lab10_Task2.sql', and 'Lab10_Task3.sql'. The 'Worksheet' tab is selected. The code in the worksheet window is:

```
90 SET TIMING ON;
91 CREATE TABLE calendar PARALLEL 4
92 AS SELECT * FROM (
93   SELECT
94     TRUNC( sd + rn ) time_id,
95     TO_CHAR( sd + rn, 'fmDay' ) day_name,
96     TO_CHAR( sd + rn, 'D' ) day_number_in_week,
97     TO_CHAR( sd + rn, 'DD' ) day_number_in_month,
98     TO_CHAR( sd + rn, 'DDD' ) day_number_in_year,
99     TO_CHAR( sd + rn, 'W' ) calendar_week_number,
100  ( CASE
101    WHEN TO_CHAR( sd + rn, 'D' ) IN ( 1, 2, 3, 4, 5, 6 ) THEN
102      NEXT_DAY( sd + rn, 'SUNDAY' )
103    ELSE
104      ( sd + rn )
```

The 'Script Output' tab is selected in the bottom panel. The output shows the results of the command:

```
Table CALENDAR created.

Elapsed: 00:00:00.124
```

Summary:

| | Number of rows deleted | Time to Execute Elapsed Time | Elapsed Time | Time to Simple CREATE TABLE ...AS SELECT of Explain plane | Cost |
|--|------------------------|------------------------------|--------------|---|------|
| Simple CREATE TABLE ...AS SELECT | 1000 | 0.102 seconds | 00:00:00.088 | 00:00:01 | 4 |
| Simple CREATE TABLE ...AS SELECT with Parallel DDL | 1000 | 0.092 seconds | 00:00:00.079 | 00:00:01 | 2 |
| Simple CREATE TABLE ...AS SELECT | 10 000 | 0.345 seconds | 00:00:00.330 | 00:00:01 | 4 |
| Simple CREATE TABLE ...AS SELECT with Parallel DDL | 10 000 | 0.136 seconds | 00:00:00.124 | 00:00:01 | 2 |

Parallel DDL works for both tables and indexes, whether partitioned or nonpartitioned. For nonpartitioned tables and indexes, only the following types of DDL statements can be parallelized:

CREATE TABLE...AS SELECT

CREATE INDEX

ALTER INDEX...REBUILD

When working with partitioned tables and indexes, the scope of Oracle's parallel DDL support broadens. The following statements can be parallelized for partitioned tables and indexes:

CREATE TABLE...AS SELECT

ALTER TABLE...MOVE PARTITION

ALTER TABLE...SPLIT PARTITION

CREATE INDEX

ALTER INDEX...REBUILD PARTITION

ALTER INDEX...SPLIT PARTITION

Not all tables allow these operations to be executed in parallel. Tables with object columns or LOB columns don't allow parallel DDL.

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

With DDL statements, you do not specify the degree of parallelism using hints. Instead, you use the PARALLEL clause.

In our case, when using parallel DDL, the cost and execution time of the operation decreased, indicating the optimization of the process.

3. Business Task - Parallel execution

3.1. Task 03: CREATE Strategy of Parallel execution

Strategy of Parallel refreshing Fact Tables and Dimension Tables.

Databases today contain a large amount of information. However, finding, updating and presenting the right information in a timely fashion can be a challenge because of the vast quantity of data involved.

Parallel execution is the capability that addresses this challenge. Using parallel execution (also called parallelism), terabytes of data can be processed in minutes, not hours or days, simply by using multiple processes to accomplish a single task. This dramatically reduces response time for data-intensive operations on large databases typically associated with decision support systems (DSS) and data warehouses.

Parallelism is the idea of breaking down a task so that, instead of one process doing all of the work in a query, many processes do part of the work at the same time.

When it becomes necessary to make changes to Fact Tables and Dimension Tables in the data warehouse, we can use parallel queries and parallel subqueries in SELECT statements. We can also parallelize the query portions of DDL statements and DML statements (INSERT, UPDATE, and DELETE). Parallel execution can accelerate large DML operations and is especially useful in data warehouse environments where large summary or history tables must be maintained.