

TASK 4 REPORT

1. FULL SCAN, HIGH-WATER MARK AND CONSISTENT GETS

Nº	Count of Blocks	Count of Used Blocks	Count of Rows	Description
1	1664	1536	99999	1664 blocks are allocated to store the table. HWM points to the last block. There are free blocks. Not all blocks are actually filled
2	1664	0	0	After deleting all records, HWM does not change its position, but all blocks contain no data.
3	1664	1	1	After adding one record, one block was taken up. HWM still has not changed its position.
4	8	0	0	Truncation moved HWM to the beginning

2. INDEX CLUSTERING FACTOR

	⚡ IDX_NAME	⚡ CLUSTERING_FACTOR	⚡ BLOCKS	⚡ NUM_ROWS
1	T1.T1_IDX1	99999	1516	99999
2	T2.T2_IDX1	1536	1536	99999

DESCRIPTION OF THE INDEX CLUSTERING FACTOR

The clustering factor is a measure of the ordered-ness of an index in comparison to the table that it is based on. It is used to check the cost of a table lookup following an index access

EXPLANATION: WHY DO WE HAVE DIFFERENT FACTORS FOR T1_IDX1 AND T2_IDX.

A table t2 and an index key are in the same order, the clustering factor equal the number of blocks in the table and the index will be useful for very large index range scans and for retrieving numerous rows from the table. In table t1 the data is randomly scattered, the clustering factor equal the number of rows in the table, and given that the number of rows in a table is usually at least an order of magnitude more than the number of blocks, the index will be less efficient for returning numerous rows.

WHICH INDEX HAS BETTER PERFORMANCE WHILE EXECUTING SELECT CLAUSE FILTERED BY IN (LIST OF VALUES)

Index of t2 table has better performance

3: INDEX UNIQUE SCAN

```
CREATE UNIQUE INDEX idx_t1 ON t1( t_pad );  
SELECT t1.* FROM t1 where t1.t_pad = '1';
```

Script Output x Explain Plan x Query Result x

SQL | All Rows Fetched: 0 in 0.004 seconds

ID	T_PAD
----	-------

Script Output x Query Result x Explain Plan x

SQL | 0.019 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				
TABLE ACCESS	T1	BY INDEX ROWID	1	3
INDEX	IDX_T1	UNIQUE SCAN	1	2

Access Predicates
T1.T_PAD='1'

Other XML
(info)
info type="has_user_tab"
yes
info type="db_version"
21.0.0.0
info type="parse_schema"
"SYSTEM"
info type="plan_hash_full"
979526317
info type="plan_hash"
1698228356

PROCESS DESCRIPTION: HOW DOES ORACLE READ THE BLOCK ON STEP#2?

The query uses UNIQUE SCAN because it contains a condition with a column defined with UNIQUE index. This type of index guarantees that only one row will be returned for the specified value.

4.INDEX RANGE SCAN

```
SELECT t2.* FROM t2 where t2.id = rpad ('1', 100)
```

Script Output x Explain Plan x Query Result x

SQL | All Rows Fetched: 100 in 0.019 seconds

ID	T_PAD
1	1 100
2	1 101
3	1 102
4	1 103
5	1 104
6	1 105
7	1 106
8	1 107
9	1 108
10	1 109
11	1 110
12	1 111
13	1 112
14	1 113
15	1 114
16	1 115
17	1 116

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				100
TABLE ACCESS	T2	BY INDEX ROWID BATCHED		3
INDEX	T2_IDX1	RANGE SCAN	100	3
Access Predicates				1
T2.ID=1				
Other XML				
{info}				
info type="has_user_tab"				
yes				
info type="db_version"				
21.0.0.0				
info type="parse_schema"				
"SYSTEM"				
info type="plan_hash_full"				
3546877230				
info type="plan_hash"				
2614670020				

DESCRIPTION OF PROCESS: HOW DOES ORACLE READ THE BLOCK ON STEP#1?

The query uses RANGE SCAN because the condition returns a range of data.

5. INDEX SKIP SCAN

<pre>SELECT /*+INDEX_SS(emp idx_emp01)*/ emp.* FROM employees emp where ename = 'SCOTT';</pre>				
Script Output x Query Result x Query Result 1 x Query Result 2 x Explain Plan x				
SQL 0.042 seconds				
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				1
TABLE ACCESS	EMPLOYEES	BY INDEX ROWID BATCHED	1	2
INDEX	IDX_EMP01	SKIP SCAN	1	2
Access Predicates				1
ENAME='SCOTT'				
Filter Predicates				
ENAME='SCOTT'				
Other XML				
{info}				
info type="has_user_tab"				
yes				
info type="db_version"				
21.0.0.0				
info type="parse_schema"				
"SYSTEM"				
info type="dynamic_sampling" note="y"				
2				

<pre>SELECT /*+FULL*/ emp.* FROM employees emp WHERE ename = 'SCOTT';</pre>				
Script Output x Query Result x Query Result 1 x Query Result 2 x Explain Plan x				
SQL 0.048 seconds				
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				1
TABLE ACCESS	EMPLOYEES	FULL	1	2
Filter Predicates				
ENAME='SCOTT'				
Other XML				
{info}				
info type="has_user_tab"				
yes				
info type="db_version"				
21.0.0.0				
info type="parse_schema"				
"SYSTEM"				
info type="dynamic_sampling" note="y"				
2				
info type="plan_hash_full"				

PROCESS DESCRIPTION: HOW DOES ORACLE USES THE INDEX CREATED ON THE STEP#2?;

Index skip scan means, that the first column of the index is ignored. SKIP SCAN is used because the condition uses a non-leading column in the index.

6: EXECUTION PLAN ANALYSIS

1ST QUERY

Select

c3.channel_description, c2.country, sum(s.sale_price-s.sale_cost) as profit

from ce_sales s

inner join ce_stores s2

on s.store_id=s2.store_id

inner join ce_addresses a

on a.address_id=s2.address_id

inner join CE_CITIES c

on c.city_id=a.city_id

inner join ce_states s3

on s3.state_id=c.state_id

inner join ce_countries c2

on c2.country_id=s3.country_id

inner join ce_channels c3

on s.channel_id=c3.channel_id

where c2.country_id in

(Select country_id

From ce_countries

where country in ('Poland','United States of America','Spain'))

Group by c3.channel_description, c2.country

having sum(s.sale_price-s.sale_cost)>10000

order by sum(s.sale_price-s.sale_cost) desc;

PLAN_TABLE_OUTPUT									
1	Plan hash value: 1594611513								
2									
3	-----								
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time		
5	-----								
6	0	SELECT STATEMENT		1	77	7321 (2)	00:00:01		
7	1	SORT ORDER BY		1	77	7321 (2)	00:00:01		
8	* 2	FILTER							
9	3	HASH GROUP BY		1	77	7321 (2)	00:00:01		
10	* 4	HASH JOIN		247K	18M	7292 (1)	00:00:01		
11	5	TABLE ACCESS FULL	CE_CHANNELS	9	117	3 (0)	00:00:01		
12	* 6	HASH JOIN		247K	15M	7287 (1)	00:00:01		
13	* 7	HASH JOIN		11	506	14 (8)	00:00:01		
14	* 8	HASH JOIN		11	418	11 (10)	00:00:01		
15	* 9	HASH JOIN		6	180	8 (13)	00:00:01		
16	10	MERGE JOIN		27	432	6 (17)	00:00:01		
17	11	TABLE ACCESS BY INDEX ROWID	CE_STATES	21	168	2 (0)	00:00:01		
18	12	INDEX FULL SCAN	CE_STATE_PK	21		1 (0)	00:00:01		
19	* 13	SORT JOIN		27	216	4 (25)	00:00:01		
20	14	TABLE ACCESS FULL	CE_CITIES	27	216	3 (0)	00:00:01		
21	15	INLIST ITERATOR							
22	16	TABLE ACCESS BY INDEX ROWID BATCHED	CE_COUNTRIES	3	42	2 (0)	00:00:01		
23	* 17	INDEX RANGE SCAN	IDX_COUNTRY	3		1 (0)	00:00:01		
24	18	TABLE ACCESS FULL	CE_ADDRESSES	47	376	3 (0)	00:00:01		
25	19	TABLE ACCESS FULL	CE_STORES	47	376	3 (0)	00:00:01		
26	20	TABLE ACCESS FULL	CE_SALES	1071K	18M	7265 (1)	00:00:01		
27	-----								
28									
29	Predicate Information (identified by operation id):								
30	-----								
31									
32	2	filter(SUM("S"."SALE_PRICE"-"S"."SALE_COST")>10000)							
33	4	access("S"."CHANNEL_ID"="C3"."CHANNEL_ID")							
34	6	access("S"."STORE_ID"="S2"."STORE_ID")							
35	7	access("A"."ADDRESS_ID"="S2"."ADDRESS_ID")							
36	8	access("C"."CITY_ID"="A"."CITY_ID")							
37	9	access("C2"."COUNTRY_ID"="S3"."COUNTRY_ID")							
38	13	access("S3"."STATE_ID"="C"."STATE_ID")							
39		filter("S3"."STATE_ID"="C"."STATE_ID")							
40	17	access("C2"."COUNTRY"='Poland' OR "C2"."COUNTRY"='Spain' OR "C2"."COUNTRY"='United States of							
41		America')							
42									
43	Note								
44	-----								
45	- this is an adaptive plan								

17	Oracle use index Idx_counrty in range scan
16	Oracle access table ce_countries by index rowid
15	Oracle run inlist iterator
14	Oracle get full access to ce_cities table
12	Oracle use ce_state_pk
11	Oracle get ce_states by index_rowid
13	Oracle join tables

18	Oracle get full access to ce_addresses
10	Oracle join ce_address to the result of 13 step
19	Oracle get full access to ce_stores
9-6	Oracle join tables
5	Oracle get full access to ce_channels
4	Oracle join tables
3	Oracle group the result
2	Oracle filter the result
1	Oracle sort the result

2ND QUERY

Select c.first_name, c.last_name, sum(s.sale_price), sum(s.sale_cost), sum(s.sale_price-s.sale_cost) as profit

from ce_sales s

inner join ce_customers c

on s.customer_id=c.customer_id

inner join ce_countries c2

on c2.country_id=c.country_id

Where c.first_name!='N/A' and c.last_name!='N/A' and c2.country in('Poland', 'Germany','United States of America','Spain')

Group by c.first_name, c.last_name

order by sum(s.sale_price-s.sale_cost) desc;

PLAN_TABLE_OUTPUT									
1	Plan hash value: 1982831152								
2									
3									
4	Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time	
5									
6	0	SELECT STATEMENT		7513	594K		10011 (2)	00:00:01	
7	1	SORT ORDER BY		7513	594K	720K	10011 (2)	00:00:01	
8	2	HASH GROUP BY		7513	594K	720K	10011 (2)	00:00:01	
9	* 3	HASH JOIN		7513	594K		9721 (2)	00:00:01	
10	4	JOIN FILTER CREATE	:BF0000	7513	271K		311 (1)	00:00:01	
11	* 5	HASH JOIN		7513	271K		311 (1)	00:00:01	
12	6	INLIST ITERATOR							
13	7	TABLE ACCESS BY INDEX ROWID BATCHED	CE_COUNTRIES	4	56		2 (0)	00:00:01	
14	* 8	INDEX RANGE SCAN	IDX_COUNTRY	4			1 (0)	00:00:01	
15	* 9	TABLE ACCESS FULL	CE_CUSTOMERS	37565	843K		309 (1)	00:00:01	
16	10	VIEW	VW_GBC_10	46160	1983K		9409 (2)	00:00:01	
17	11	HASH GROUP BY		46160	676K	28M	9409 (2)	00:00:01	
18	12	JOIN FILTER USE	:BF0000	1071K	15M		7265 (1)	00:00:01	
19	* 13	TABLE ACCESS FULL	CE_SALES	1071K	15M		7265 (1)	00:00:01	
20									
21									
22	Predicate Information (identified by operation id):								
23									
24									
25	3 - access("ITEM_1"="C"."CUSTOMER_ID")								
26	5 - access("C2"."COUNTRY_ID"="C"."COUNTRY_ID")								
27	8 - access("C2"."COUNTRY"='Germany' OR "C2"."COUNTRY"='Poland' OR "C2"."COUNTRY"='Spain' OR								
28	"C2"."COUNTRY"='United States of America')								
29	9 - filter("C"."LAST_NAME"<>'N/A' AND "C"."FIRST_NAME"<>'N/A')								
30	13 - filter(SYS_OP_BLOOM_FILTER(:BF0000,"S"."CUSTOMER_ID"))								

8	Oracle use index Idx_counrty in range scan
7	Oracle get full access to ce_countries
6	oracle use inlist iterator
9	Oracle get full access to ce_customers
5	Oracle join ce_customers and ce_countries
4	Oracle create join filter
13	Oracle get full access to ce_sales
12	Oracle use join filter
11	Oracle group sales
10	Oracle get view of sales
3	Oracle join sales view and customer_by_country join
2	Oracle group by result
1	Oracle sort the result

3D QUERY

```

Select b.product_brand,pt.product_type_id, sum(s.sale_price-s.sale_cost) as profit
From ce_sales s
inner join ce_products p
on p.product_id=s.product_id
inner join ce_brands b
on b.product_brand_id=p.product_brand_id
inner join ce_product_types pt
on pt.product_type_id=p.product_type_id
group by b.product_brand,pt.product_type_id
order by sum(s.sale_price-s.sale_cost) desc;

```

PLAN_TABLE_OUTPUT									
1	Plan hash value: 2625066749								
2									
3	-----								
4	Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time	
5	-----								
6	0	SELECT STATEMENT		10665	458K		7651 (2)	00:00:01	
7	1	SORT ORDER BY		10665	458K	640K	7651 (2)	00:00:01	
8	2	HASH GROUP BY		10665	458K	640K	7651 (2)	00:00:01	
9	* 3	HASH JOIN		10665	458K		7404 (2)	00:00:01	
10	4	VIEW	VW_GBC_10	10665	187K		7333 (2)	00:00:01	
11	5	HASH GROUP BY		10665	156K		7333 (2)	00:00:01	
12	6	TABLE ACCESS FULL	CE_SALES	1071K	15M		7265 (1)	00:00:01	
13	7	VIEW	VW_GBF_11	10666	270K		71 (0)	00:00:01	
14	* 8	HASH JOIN		10666	291K		71 (0)	00:00:01	
15	9	TABLE ACCESS FULL	CE_BRANDS	424	6360		3 (0)	00:00:01	
16	10	TABLE ACCESS FULL	CE_PRODUCTS	10666	135K		68 (0)	00:00:01	
17	-----								
18									
19	Predicate Information (identified by operation id):								
20	-----								
21									
22	3 - access("ITEM_1"="ITEM_1")								
23	8 - access("B"."PRODUCT_BRAND_ID"="P"."PRODUCT_BRAND_ID")								

10	Oracle full access to ce_products
9	Oracle full access to ce_brands
8	Oracle join ce_products and ce_brands to ce_products
7	Oracle get view of previous steps

6	<i>Oracle get full access to ce_sales</i>
5	<i>Oracle group ce_sales</i>
4	<i>Oracle get view of ce_sales</i>
3	<i>Oracle join ce_sales view to ce_product by category and brand</i>
2	<i>Oracle group by the result</i>
1	<i>Oracle sort the result</i>

7. ORACLE OPTIMIZATION

SQL QUERY

Select

c3.channel_description, c2.country, sum(s.sale_price-s.sale_cost) as profit

from ce_sales s

inner join ce_stores s2

on s.store_id=s2.store_id

inner join ce_addresses a

on a.address_id=s2.address_id

inner join CE_CITIES c

on c.city_id=a.city_id

inner join ce_states s3

on s3.state_id=c.state_id

inner join ce_countries c2

on c2.country_id=s3.country_id

inner join ce_channels c3

on s.channel_id=c3.channel_id

where c2.country_id in

(Select country_id

From ce_countries

where country in ('Poland','United States of America','Spain'))

Group by c3.channel_description, c2.country

having sum(s.sale_price-s.sale_cost)>10000

order by sum(s.sale_price-s.sale_cost) desc;

PLAN_TABLE_OUTPUT									
1	Plan hash value: 1594611513								
2									
3	-----								
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time		
5	-----								
6	0	SELECT STATEMENT		1	77	7321 (2)	00:00:01		
7	1	SORT ORDER BY		1	77	7321 (2)	00:00:01		
8	* 2	FILTER							
9	3	HASH GROUP BY		1	77	7321 (2)	00:00:01		
10	* 4	HASH JOIN		247K	18M	7292 (1)	00:00:01		
11	5	TABLE ACCESS FULL	CE_CHANNELS	9	117	3 (0)	00:00:01		
12	* 6	HASH JOIN		247K	15M	7287 (1)	00:00:01		
13	* 7	HASH JOIN		11	506	14 (8)	00:00:01		
14	* 8	HASH JOIN		11	418	11 (10)	00:00:01		
15	* 9	HASH JOIN		6	180	8 (13)	00:00:01		
16	10	MERGE JOIN		27	432	6 (17)	00:00:01		
17	11	TABLE ACCESS BY INDEX ROWID	CE_STATES	21	168	2 (0)	00:00:01		
18	12	INDEX FULL SCAN	CE_STATE_PK	21		1 (0)	00:00:01		
19	* 13	SORT JOIN		27	216	4 (25)	00:00:01		
20	14	TABLE ACCESS FULL	CE_CITIES	27	216	3 (0)	00:00:01		
21	15	INLIST ITERATOR							
22	16	TABLE ACCESS BY INDEX ROWID BATCHED	CE_COUNTRIES	3	42	2 (0)	00:00:01		
23	* 17	INDEX RANGE SCAN	IDX_COUNTRY	3		1 (0)	00:00:01		
24	18	TABLE ACCESS FULL	CE_ADDRESSES	47	376	3 (0)	00:00:01		
25	19	TABLE ACCESS FULL	CE_STORES	47	376	3 (0)	00:00:01		
26	20	TABLE ACCESS FULL	CE_SALES	1071K	18M	7265 (1)	00:00:01		
27	-----								
28									
29	Predicate Information (identified by operation id):								
30	-----								
31									
32	2	filter(SUM("S"."SALE_PRICE"- "S"."SALE_COST")>10000)							
33	4	access("S"."CHANNEL_ID"="C3"."CHANNEL_ID")							
34	6	access("S"."STORE_ID"="S2"."STORE_ID")							
35	7	access("A"."ADDRESS_ID"="S2"."ADDRESS_ID")							
36	8	access("C"."CITY_ID"="A"."CITY_ID")							
37	9	access("C2"."COUNTRY_ID"="S3"."COUNTRY_ID")							
38	13	access("S3"."STATE_ID"="C"."STATE_ID")							
39		filter("S3"."STATE_ID"="C"."STATE_ID")							
40	17	access("C2"."COUNTRY"='Poland' OR "C2"."COUNTRY"='Spain' OR "C2"."COUNTRY"='United States of							
41		America')							
42									
43	Note								
44	-----								
45	- this is an adaptive plan								

17	Oracle use index Idx_counrty in range scan
16	Oracle access table ce_countries by index rowid
15	Oracle run inlist iterator
14	Oracle get full access to ce_cities table
12	Oracle use ce_state_pk
11	Oracle get ce_states by index_rowid
13	Oracle join tables

18	<i>Oracle get full access to ce_addresses</i>
10	<i>Oracle join ce_address to the result of 13 step</i>
19	<i>Oracle get full access to ce_stores</i>
9-6	<i>Oracle join tables</i>
5	<i>Oracle get full access to ce_channels</i>
4	<i>Oracle join tables</i>
3	<i>Oracle group the result</i>
2	<i>Oracle filter the result</i>
1	<i>Oracle sort the result</i>

UPDATED SQL QUERIES

```

create MATERIALIZED VIEW sum_by_countries_ch
as
Select
c3.channel_description, c2.country, sum(s.sale_price-s.sale_cost) as profit
from ce_sales s
inner join ce_stores s2
on s.store_id=s2.store_id
inner join ce_addresses a
on a.address_id=s2.address_id
inner join CE_CITIES c
on c.city_id=a.city_id
inner join ce_states s3
on s3.state_id=c.state_id
inner join ce_countries c2
on c2.country_id=s3.country_id
inner join ce_channels c3
on s.channel_id=c3.channel_id

```

Group by c3.channel_description, c2.country;

Select channel_description, country, profit

from sum_by_countries_ch

where country in ('Poland','United States of America','Spain') and profit >10000

order by profit desc;

PLAN_TABLE_OUTPUT							
1	Plan hash value: 409451967						
2							
3	-----						
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
5	-----						
6	0	SELECT STATEMENT		11	2387	3 (34)	00:00:01
7	1	SORT ORDER BY		11	2387	3 (34)	00:00:01
8	* 2	MAT_VIEW ACCESS FULL	SUM_BY_COUNTRIES_CH	11	2387	2 (0)	00:00:01
9	-----						
10							
11	Predicate Information (identified by operation id):						
12	-----						
13							
14	2 - filter(("COUNTRY"='Poland' OR "COUNTRY"='Spain' OR "COUNTRY"='United States						
15	of America') AND "PROFIT">10000)						
16							
17	Note						
18	-----						
19	- dynamic statistics used: dynamic sampling (level=2)						

2	Oracle get full access to sum_by_countries_ch
1	Oracle sort order the result

COMPARISON OF THE EXECUTION PLANS WITH CONCLUSION

The new execution plan is less expensive:the query cost dropped from 7321 to 3 (more than 2000 times).