OPTIMIZATION OF DYNAMICALLY GENERATED SQL QUERIES FOR TINY-HUGE, HUGE-TINY PROBLEM

Arjun K Sirohi[1]

[1]Doctorate Student, Faculty of Technology (Computer Science), CMJ University, Shillong, India
[2]Consulting Member Technical Staff, Oracle, Bellevue, WA, USA
asirohi@yahoo.com

ABSTRACT

In most new commercial business software applications like Customer Relationship Management, the data is stored in the database layer which is usually a Relational Database Management System (RDBMS) like Oracle, DB2 UDB or SQL Server. To access data from these databases, Structured Query Language (SQL) queries are used that are generated dynamically at run time based on defined business models and business rules. One such business rule is visibility- the capability of the application to restrict data access based on the role and responsibility of the user logged in to the application. This is generally achieved by appending security predicates in the form of sub-queries to the main query based on the roles and responsibility of the user. In some cases, the outer query may be more restrictive while in other cases, the security predicates may be more restrictive. This often results in a dilemma for the cost-based optimizer (CBO) of the backend database whether to drive from the outer query or drive from the security predicate sub-queries. This dilemma is sometimes called the "Tiny-Huge, Huge-Tiny" problem and results in serious performance degradation by way of increased response times on the application User Interface (UI). This paper provides a case study of a new approach to vastly reduce this CBO dilemma by a combination of de-normalized columns and re-writing of the security predicates' sub-queries at run-time, thereby levelling the outer and security sub-queries. This approach results in more stable execution plans in the database and much better performance of such SQLs, effectively leading to higher performance and scalability of the application.

KEYWORDS

SQL Performance, RDBMS, Cost Based Optimizer (CBO), SQL, Plan Cost, Execution Plan, Buffer Gets, Query Optimization, Performance, Scalability, CRM Applications

1. INTRODUCTION

Most commercial business applications use relational databases as the back-end to store business data. Such data is accessed using SQL queries that are dynamically generated by the application framework, using a defined business model and business rules. For example, Oracle Application Development Framework uses its SQL generation engine to do this. The performance of the application is generally gauged by the response times in the UI. The UI response times in turn depends in a large part on the query response time in the database. If a user has to wait to get the results in the UI, they complain about it as the application performing badly. The query response time in the database depends on many factors, including the size and complexity of the SQL text as well as how much processing the database engine must do before it can arrive at the result set to be sent to the application requesting for it. A simple SQL with few tables and joins with good, restrictive filter predicates will generally perform better than a SQL with many tables, views and joins and less restrictive filter predicates. One of the key elements of SQL performance is the

DOI: 10.5121/ijdms.2013.5105

decision by the CBO to arrive at an execution plan that it considers as most optimal. In Oracle database, the CBO is a complex engine and it evaluates many different optimization paths and access methods before finalizing an execution plan. In recent releases, the concept of cardinality feedback and other features also often result in second, third or more execution plans for the same SQL. From the application perspective, a common way to restrict access to the data on a need to know basis is to implement security and visibility through a set of roles and responsibilities that each are defined by way of sub-queries that get appended to the main SQL at run time. This method of adding sub-queries cumulatively through a security framework results in complex SQLs, especially for users who are granted many different roles and responsibilities. The main query is usually referred to as the outer query and the appended security predicates' sub-queries are termed as the inner query. When such SQLs arrive at the database, the CBO evaluates them for possible access paths and join optimizations to decide on an execution plan based on available statistics. Very often, the CBO is posed with the dilemma of whether to drive from the outer query or drive from the inner query. For some users, the outer query with its filter predicates can be greatly restrictive while for other users, the inner security predicates' sub-queries can be more restrictive. This if sometimes referred to as the "Tiny-Huge, Huge-Tiny" problem that many application and database designers struggle to manage. Many times, this problem leads to poor choices by the CBO, resulting in sub-optimal execution plans leading to poor query response times and consequently causing performance and scalability issues for the application as well as the database. This paper presents the analysis of SQLs and CBO execution plans from the Opportunity Management module of a CRM application that performed poorly due to the aforesaid "Tiny-Huge, Huge-Tiny" problem. Based on the analysis, the paper then presents a suggested solution incorporating some de-normalized columns and rewrite of the security predicates' sub-queries that result in vastly improved performance and scalability of such queries and consequently of the application.

2. The Existing SQLs, Plans and Issues Affecting Performance

The Opportunity Management Module of any CRM application is a widely used application in any sales department of businesses. Details of sales opportunities are stored in tables of relational databases and accessed through the user interface of the web-based application. The visibility or access-control of which user can access which records of such sales opportunities is controlled through rules defined in terms of SQL sub-queries that get appended to the main SQL at run time based on the roles and responsibilities of the logged in user. For example, roles could be Sales Representative, Sales Manager, Sales Vice President or Sales Administrator. The access could also be defined in terms of teams or sales territories. The access could also come from the user being in a management hierarchy. All these access rules are stored as seeded sub-queries in the application and get appended at run-time to a dynamically generated SQL. As such, the developer does not have much control over how the SQL is written.

2.1. Existing SQLs, Performance and Execution Plans

A sample SQL text, its performance metrics and execution plan are presented below.

```
SELECT
  /*+ FIRST_ROWS(10)  */
  Column1,
  Column2
  .
  .
  .
  .
  .
ColumnN
FROM
  MOO_OPTY OpportunityEO,
  HZ_PARTIES PartyPEO,
  MOO_SALES_METHOD_VL SalesMethodEO,
  MOO_STG_VL SalesStageEO,
  MOO_REVN RevenueEO,
  HZ_PARTY_SITES PartySitePEO,
  HZ_PARTIES PrimaryCompetitorPartyPEO,
  HZ_LOCATIONS LocationPEO,
  HZ_PARTIES OwnerResourcePartyPEO,
  ZBS_LOOKUP_VALUES_VL LookupValuesEO,
  HZ_PARTIES PrimaryContactPartyPEO,
  ZCA_SALES_ACCOUNTS SalesAccountPEO,
  HZ_RELATIONSHIPS RelationshipPEO,
  HZ_CONTACT_POINTS PhonePEO,
  HZ_CODE_ASSIGNMENTS CodeAssignmentPEO
WHERE OpportunityEO.CUST_PARTY_ID         = PartyPEO.PARTY_ID(+)
AND OpportunityEO.SALES_METHOD_ID         = SalesMethodEO.SALES_METHOD_ID(+)
AND OpportunityEO.CURR_STG_ID             = SalesStageEO.STG_ID(+)
AND OpportunityEO.SUM_REVN_ID             = RevenueEO.REVN_ID
AND OpportunityEO.OPTY_PARTY_SITE_ID      = PartySitePEO.PARTY_SITE_ID(+)
AND OpportunityEO.PR_CMPT_PARTY_ID        = PrimaryCompetitorPartyPEO.PARTY_ID(+)
AND PartySitePEO.LOCATION_ID              = LocationPEO.LOCATION_ID(+)
AND OpportunityEO.OWNER_RESOURCE_ID       = OwnerResourcePartyPEO.PARTY_ID(+)
AND OpportunityEO.STATUS_CD               = LookupValuesEO.LOOKUP_CODE(+)
AND ('Y')                                 = LookupValuesEO.ENABLED_FLAG(+)
AND ('OPTY_STATUS')                       = LookupValuesEO.LOOKUP_TYPE(+)
AND OpportunityEO.PR_CON_PARTY_ID         = PrimaryContactPartyPEO.PARTY_ID(+)
AND OpportunityEO.CUST_PARTY_ID           = SalesAccountPEO.PARTY_ID(+)
AND OpportunityEO.PR_CON_RELATIONSHIP_ID = RelationshipPEO.RELATIONSHIP_ID(+)
AND OpportunityEO.PR_CON_PARTY_ID         = RelationshipPEO.SUBJECT_ID(+)
AND RelationshipPEO.RELATIONSHIP_ID       = PhonePEO.RELATIONSHIP_ID(+)
AND RelationshipPEO.SUBJECT_ID            = PhonePEO.OWNER_TABLE_ID(+)
AND RelationshipPEO.SUBJECT_TABLE_NAME    = PhonePEO.OWNER_TABLE_NAME(+)
AND ('PHONE')                             = PhonePEO.CONTACT_POINT_TYPE(+)
AND ('A')                                 = PhonePEO.STATUS(+)
AND OpportunityEO.CUST_PARTY_ID           = CodeAssignmentPEO.OWNER_TABLE_ID(+)
AND ('ORGANIZATION_TYPE')                 = CodeAssignmentPEO.CLASS_CATEGORY(+)
AND ('HZ_PARTIES')                        = CodeAssignmentPEO.OWNER_TABLE_NAME(+)
AND ('Y')                                 = CodeAssignmentPEO.PRIMARY_FLAG(+)
AND ('A')                                 = CodeAssignmentPEO.STATUS(+)
AND ( ( (EXISTS
  (SELECT 1
  FROM MOO_OPTY_RESOURCES OpportunityResourceEONQ1,
    JTF_RS_REP_MANAGERS ReportingManagerPEONQ1
  WHERE (((OpportunityResourceEONQ1.RESOURCE_ID          = ReportingManagerPEONQ1.RESOURCE_ID(+))
  AND (OpportunityResourceEONQ1.RESOURCE_ORG_ID          = ReportingManagerPEONQ1.GROUP_ID(+)))
  AND (OpportunityResourceEONQ1.ORG_TREE_CODE            = ReportingManagerPEONQ1.TREE_CODE(+)))
  AND (OpportunityResourceEONQ1.ORG_TREE_STRUCTURE_CODE = ReportingManagerPEONQ1.TREE_STRUCTURE_CODE(+))
  AND ( ( (OpportunityResourceEONQ1.RESOURCE_ID         <> :BindLoggedInUserResourceId )
  AND (ReportingManagerPEONQ1.PARENT_RESOURCE_ID         = :BindLoggedInUserResourceId ) ) )
  AND (OpportunityEO.OPTY_ID                             = OpportunityResourceEONQ1.OPTY_ID)
  )) ) )
AND ( ( (OpportunityEO.STATUS_CD         = :BindDefaultStatusCd )
AND (RevenueEO.EFFECTIVE_DATE BETWEEN :BindEffectiveBeginDate AND :BindEffectiveEndDate ) ) )
AND ( ( (RelationshipPEO.STATUS(+)   = :vc_temp_4 )
AND (RelationshipPEO.START_DATE(+) <= :BindSysdate )
AND (RelationshipPEO.END_DATE(+)   >= :BindSysdate ) ) )
AND ( ( (PhonePEO.END_DATE(+)      >= :BindSysdate )
AND (PhonePEO.START_DATE(+)        <= :BindSysdate )
AND (PhonePEO.PRIMARY_FLAG(+)       = :vc_temp_5 ) ) ) )
AND ( ((( OpportunityEO.opty_id          IN
```

Figure 1. Existing SQL Outer Query

```
AND ( ((( OpportunityEO.opty_id         IN
  (SELECT opty.opty_id
    FROM FUSION.ZCA_S_ACCT_RESOURCES acntres,
         FUSION.ZCA_SALES_ACCOUNTS salesaccnt,
         FUSION.moo_opty opty,
         FUSION.Jtf_rs_rep_managers mgrchain , -- ADDED
         FUSION.Fnd_tree_version ftv  -- ADDED
    WHERE opty.cust_party_id         = salesaccnt.party_id
    AND salesaccnt.sales_account_id = acntres.sales_account_id
    AND SYSDATE BETWEEN ftv.effective_start_date AND ftv.effective_end_date
    AND ftv.tree_version_id          = mgrchain.tree_version_id
    AND ftv.tree_structure_code      = mgrchain.tree_structure_code
    AND ftv.tree_code                = mgrchain.tree_code
    AND acntres.org_tree_structure_code = mgrchain.tree_structure_code
    AND acntres.org_tree_code           = mgrchain.tree_code
    AND acntres.resource_org_id         = mgrchain.group_id
    AND mgrchain.parent_resource_id  = ( :BindResourceID )
  ) )
OR ( OpportunityEO.opty_id IN
  (SELECT ores.opty_id
    FROM fusion.Moo_opty_resources ores,
           fusion.Jtf_rs_rep_managers mgrchain ,
         fusion.Fnd_tree_version ftv
    WHERE ores.access_level_code IN ('100', '200', '300')
        AND   ores.resource_org_id          = mgrchain.group_id
        AND   ores.org_tree_structure_code = mgrchain.tree_structure_code
    AND   ores.org_tree_code            = mgrchain.tree_code
    AND   SYSDATE BETWEEN ftv.effective_start_date AND ftv.effective_end_date
    AND   ftv.tree_version_id           = mgrchain.tree_version_id
    AND   ftv.tree_structure_code       = mgrchain.tree_structure_code
    AND   ftv.tree_code                 = mgrchain.tree_code
    AND   mgrchain.parent_resource_id  = ( :BindResourceID )
  ) )
OR ( OpportunityEO.opty_id IN
  (SELECT MOO_OPTY.opty_id
  FROM FUSION.MOT_TERR_RESOURCES,
    FUSION.ZCA_S_ACCT_TERRITORIES,
    FUSION.ZCA_SALES_ACCOUNTS,
    FUSION.MOO_OPTY
  WHERE MOT_TERR_RESOURCES.resource_id = (:BindResourceID)
  AND MOT_TERR_RESOURCES.territory_version_id = ZCA_S_ACCT_TERRITORIES.territory_version_id
  AND ZCA_S_ACCT_TERRITORIES.sales_account_id = ZCA_SALES_ACCOUNTS.sales_account_id
  AND ZCA_SALES_ACCOUNTS.party_id             = MOO_OPTY.cust_party_id
  )
OR OpportunityEO.opty_id IN
  (SELECT MOO_OPTY.opty_id
  FROM FUSION.MOT_TERR_RESOURCES,
    FUSION.MOT_TERRITORIES,
    FUSION.MOT_TERR_HIERARCHY_DN,
    FUSION.ZCA_S_ACCT_TERRITORIES,
    FUSION.ZCA_SALES_ACCOUNTS,
    FUSION.MOO_OPTY
  WHERE MOT_TERR_RESOURCES.resource_id =     (:BindResourceID   )
  AND MOT_TERR_RESOURCES.territory_version_id        = MOT_TERRITORIES.territory_version_id
  AND MOT_TERRITORIES.latest_version_flag            = 'Y'
  AND MOT_TERRITORIES.status_code                    = 'FINALIZED'
  AND MOT_TERRITORIES.territory_id                   = MOT_TERR_HIERARCHY_DN.territory_id
  AND MOT_TERR_HIERARCHY_DN.descendant_territory_id != MOT_TERR_HIERARCHY_DN.territory_id
  AND MOT_TERR_HIERARCHY_DN.descendant_territory_id  = ZCA_S_ACCT_TERRITORIES.territory_id
  AND ZCA_S_ACCT_TERRITORIES.sales_account_id        = ZCA_SALES_ACCOUNTS.sales_account_id
  AND ZCA_SALES_ACCOUNTS.party_id                    = MOO_OPTY.cust_party_id
  ) )
OR ( OpportunityEO.opty_id IN
  (SELECT MOO_REVN.opty_id
  FROM FUSION.MOT_TERR_RESOURCES,
    FUSION.MOO_REVN_TERR,
    FUSION.MOO_REVN
  WHERE MOT_TERR_RESOURCES.resource_id =    (:BindResourceID    )
  AND MOT_TERR_RESOURCES.territory_version_id = MOO_REVN_TERR.territory_version_id
  AND MOO_REVN_TERR.SUM_REVN_FLAG             = 'Y'
  AND MOO_REVN_TERR.revn_id                   = MOO_REVN.revn_id
  AND MOO_REVN.PRIMARY_FLAG                   = 'Y'
  )
OR OpportunityEO.opty_id IN
  (SELECT revn.opty_id
  FROM FUSION.MOT_TERR_RESOURCES terr_res,
    FUSION.MOT_TERRITORIES terr,
    FUSION.MOT_TERR_HIERARCHY_DN terrdn,
    FUSION.MOO_REVN_TERR revnterr,
    FUSION.MOO_REVN revn
  WHERE terr_res.resource_id =  (:BindResourceID )
  AND terr_res.territory_version_id    = terr.territory_version_id
  AND terr.latest_version_flag         = 'Y'
  AND terr.status_code                 = 'FINALIZED'
  AND terr.territory_id                = terrdn.territory_id
  AND terrdn.descendant_territory_id != terrdn.territory_id
  AND terrdn.descendant_territory_id  = revnterr.territory_id
  AND revnterr.SUM_REVN_FLAG           = 'Y'
  AND revnterr.revn_id                 = revn.revn_id
  AND revn.primary_flag                = 'Y'
  ) )
OR ( OpportunityEO.opty_id IN
  (SELECT myopres.opty_id
  FROM Moo_opty_resources myopres
  WHERE myopres.resource_id =   (:BindResourceID )
  AND myopres.access_level_code IN ('100', '200', '300')
  ) )))
```

Figure 2. Existing Data Security Sub-Queries

A sample SQL is provided in two parts above. The first part is the outer SQL that is generated at run time by the application engine. The second part is the data security predicate sub-queries that are appended to the outer SQL at run time based on the identity of the application user logged in.

The performance of such a SQL against an Oracle 11gR2 database was measured using a benchmarking Java tool for fetching the first 25 rows from the database.

```
Bind Variable 1::BindResourceID
Bind Variable 2::BindResourceID
Bind variable 3:OPEN
Bind variable 4:06/01/2012
Bind variable 5:05/31/2013
Bind variable 6:A
Bind variable 7:11/01/2012
Bind variable 8:11/01/2012
Bind variable 9:11/01/2012
Bind variable 10:11/01/2012
Bind variable 11:Y


Iter Prepare(s) Execute(s) Fetch(s)  FetchedRows Total(s)
---- ---------- ---------- --------  ------------ --------
   1      0.032      6.492    3.832            25   10.357
   2      0.000      1.810    3.602            25    5.411
   3      0.000      1.800    3.629            25    5.429

**Buffer_Gets(per exec): 2576102.6
**Shared Mem(MB) : 0.534
**Plan Cost:  145
```

Figure 3. Bind values and SQL benchmark metrics

The execution plan chosen by the optimizer is shown in Figure 4 below.

Id	Operation	Name	Starts	E-Rows	Cost (%CPU)	A-Rows	A-Time	Buffers	Reads
0	SELECT STATEMENT		3		145 (100)	90	00:00:17.03	7728K	76
* 1	COUNT STOPKEY		24			0	00:00:00.01	63	0
* 2	TABLE ACCESS BY INDEX ROWID	HZ_CONTACT_PREFERENCES	24	1	4 (0)	0	00:00:00.01	63	0
* 3	INDEX RANGE SCAN	HZ_CONTACT_PREFERENCES_N1	24	1	3 (0)	0	00:00:00.01	63	0
4	NESTED LOOPS		3	94	145 (0)	90	00:00:17.03	7728K	76
5	NESTED LOOPS SEMI		3	1	49 (0)	90	00:00:00.41	3900	31
6	NESTED LOOPS OUTER		3	1	38 (0)	90	00:00:00.03	3336	0
7	NESTED LOOPS OUTER		3	1	36 (0)	90	00:00:00.03	3120	0
8	NESTED LOOPS OUTER		3	1	34 (0)	90	00:00:00.03	2904	0
9	NESTED LOOPS OUTER		3	1	32 (0)	90	00:00:00.03	2856	0
10	NESTED LOOPS OUTER		3	1	30 (0)	90	00:00:00.03	2556	0
11	NESTED LOOPS OUTER		3	1	28 (0)	90	00:00:00.03	2403	0
12	NESTED LOOPS OUTER		3	1	25 (0)	90	00:00:00.02	1899	0
13	NESTED LOOPS OUTER		3	1	22 (0)	90	00:00:00.02	1638	0
14	NESTED LOOPS OUTER		3	1	20 (0)	90	00:00:00.02	1344	0
15	NESTED LOOPS OUTER		3	1	18 (0)	90	00:00:00.02	1236	0
16	NESTED LOOPS OUTER		3	1	16 (0)	90	00:00:00.02	1083	0
17	NESTED LOOPS OUTER		3	1	14 (0)	90	00:00:00.02	1014	0
18	NESTED LOOPS OUTER		3	1	11 (0)	90	00:00:00.02	921	0
19	NESTED LOOPS		3	1	6 (0)	90	00:00:00.02	750	0
* 20	TABLE ACCESS STORAGE FULL FIRST ROWS	MOO_OPTY	3	2	2 (0)	93	00:00:00.01	459	0
* 21	TABLE ACCESS BY INDEX ROWID	MOO_REVN	93	1	2 (0)	90	00:00:00.01	291	0
* 22	INDEX UNIQUE SCAN	MOO_REVN_PK	93	1	1 (0)	93	00:00:00.01	198	0
* 23	TABLE ACCESS BY INDEX ROWID	HZ_RELATIONSHIPS	90	1	5 (0)	30	00:00:00.01	171	0
* 24	INDEX RANGE SCAN	HZ_RELATIONSHIPS_U1	90	2	3 (0)	66	00:00:00.01	108	0
* 25	TABLE ACCESS BY INDEX ROWID	HZ_CONTACT_POINTS	90	1	3 (0)	21	00:00:00.01	93	0
* 26	INDEX RANGE SCAN	HZ_CONTACT_POINTS_N1	90	1	2 (0)	21	00:00:00.01	72	0
27	TABLE ACCESS BY INDEX ROWID	HZ_PARTIES	90	1	2 (0)	24	00:00:00.01	69	0
* 28	INDEX UNIQUE SCAN	HZ_PARTIES_PK	90	1	1 (0)	24	00:00:00.01	45	0
29	TABLE ACCESS BY INDEX ROWID	HZ_PARTY_SITES	90	1	2 (0)	48	00:00:00.01	153	0
* 30	INDEX UNIQUE SCAN	HZ_PARTY_SITES_PK	90	1	1 (0)	48	00:00:00.01	105	0
31	TABLE ACCESS BY INDEX ROWID	HZ_PARTIES	90	1	2 (0)	33	00:00:00.01	108	0
* 32	INDEX UNIQUE SCAN	HZ_PARTIES_PK	90	1	1 (0)	33	00:00:00.01	75	0
33	TABLE ACCESS BY INDEX ROWID	HZ_PARTIES	90	1	2 (0)	90	00:00:00.01	294	0
* 34	INDEX UNIQUE SCAN	HZ_PARTIES_PK	90	1	1 (0)	90	00:00:00.01	192	0
* 35	TABLE ACCESS BY INDEX ROWID	HZ_CODE_ASSIGNMENTS	90	1	3 (0)	54	00:00:00.01	261	0
* 36	INDEX RANGE SCAN	HZ_CODE_ASSIGNMENTS_U1	90	1	2 (0)	60	00:00:00.01	201	0
37	TABLE ACCESS BY INDEX ROWID	ZCA_SALES_ACCOUNTS	90	1	3 (0)	90	00:00:00.01	504	0
* 38	INDEX RANGE SCAN	ZCA_SALES_ACCOUNTS_N4	90	1	2 (0)	90	00:00:00.01	198	0
39	TABLE ACCESS BY INDEX ROWID	HZ_LOCATIONS	90	1	2 (0)	48	00:00:00.01	153	0
* 40	INDEX UNIQUE SCAN	HZ_LOCATIONS_PK	90	1	1 (0)	48	00:00:00.01	105	0
41	TABLE ACCESS BY INDEX ROWID	HZ_PARTIES	90	1	2 (0)	90	00:00:00.01	300	0
* 42	INDEX UNIQUE SCAN	HZ_PARTIES_PK	90	1	1 (0)	90	00:00:00.01	192	0
43	VIEW PUSHED PREDICATE	ZBS_LOOKUP_VALUES_VL	90	1	2 (0)	90	00:00:00.01	48	0
* 44	FILTER		90			90	00:00:00.01	48	0
45	NESTED LOOPS		90	1	2 (0)	90	00:00:00.01	48	0
* 46	TABLE ACCESS BY INDEX ROWID	ZBS_LOOKUP_VALUES_B	90	1	2 (0)	90	00:00:00.01	30	0
* 47	INDEX RANGE SCAN	ZBS_LOOKUP_VALUES_B_U2	90	1	1 (0)	90	00:00:00.01	21	0
* 48	INDEX UNIQUE SCAN	ZBS_LOOKUP_VALUES_TL_U1	90	1	0 (0)	90	00:00:00.01	18	0
49	VIEW PUSHED PREDICATE	MOO_STG_VL	90	1	2 (0)	90	00:00:00.01	216	0
50	NESTED LOOPS		90	1	2 (0)	90	00:00:00.01	216	0
51	TABLE ACCESS BY INDEX ROWID	MOO_STG_TL	90	1	1 (0)	90	00:00:00.01	108	0
* 52	INDEX UNIQUE SCAN	MOO_STG_TL_U1	90	1	0 (0)	90	00:00:00.01	18	0
53	TABLE ACCESS BY INDEX ROWID	MOO_STG_B	90	1	1 (0)	90	00:00:00.01	108	0
* 54	INDEX UNIQUE SCAN	MOO_STG_B_PK	90	1	0 (0)	90	00:00:00.01	18	0
55	VIEW PUSHED PREDICATE	MOO_SALES_METHOD_VL	90	1	2 (0)	90	00:00:00.01	216	0
56	NESTED LOOPS		90	1	2 (0)	90	00:00:00.01	216	0
57	TABLE ACCESS BY INDEX ROWID	MOO_SALES_METHOD_TL	90	1	1 (0)	90	00:00:00.01	108	0
* 58	INDEX UNIQUE SCAN	MOO_SALES_METHOD_TL_PK1	90	1	0 (0)	90	00:00:00.01	18	0
59	TABLE ACCESS BY INDEX ROWID	MOO_SALES_METHOD_B	90	1	1 (0)	90	00:00:00.01	108	0
* 60	INDEX UNIQUE SCAN	MOO_SALES_METHOD_B_PK	90	1	0 (0)	90	00:00:00.01	18	0
61	VIEW PUSHED PREDICATE	VW_SQ_2	90	1	11 (0)	90	00:00:00.14	564	31
* 62	FILTER		90			90	00:00:00.14	564	31
63	NESTED LOOPS		90			90	00:00:00.14	564	31
64	NESTED LOOPS		90	1	11 (0)	90	00:00:00.14	474	31
* 65	TABLE ACCESS BY INDEX ROWID	MOO_OPTY_RESOURCES	90	23	5 (0)	90	00:00:00.01	282	0
* 66	INDEX RANGE SCAN	MOO_OPTY_RESOURCES_N20	90	23	3 (0)	90	00:00:00.01	192	0
* 67	INDEX RANGE SCAN	JTF_RS_REP_MANAGERS_N4	90	1	2 (0)	90	00:00:00.14	192	31
* 68	TABLE ACCESS BY INDEX ROWID	JTF_RS_REP_MANAGERS	90	1	3 (0)	90	00:00:00.01	90	0
69	VIEW	VW_NSO_1	90	1	96 (0)	90	00:00:15.63	7724K	45
70	SORT UNIQUE		90	105	974 (59)	90	00:00:15.63	7724K	45
71	UNION ALL PUSHED PREDICATE		90			19479	00:00:12.86	7724K	45
* 72	TABLE ACCESS BY INDEX ROWID	MOO_OPTY_RESOURCES	90	1	5 (0)	0	00:00:00.01	360	0
* 73	INDEX RANGE SCAN	MOO_OPTY_RESOURCES_U1	90	1	4 (0)	0	00:00:00.01	360	0
74	NESTED LOOPS		90			102	00:00:00.01	1350	0
75	NESTED LOOPS		90	1	19 (0)	102	00:00:00.01	1341	0
76	NESTED LOOPS		90	1	17 (0)	102	00:00:00.01	1293	0
77	NESTED LOOPS		90	1	9 (0)	15	00:00:00.01	1131	0
78	NESTED LOOPS		90	1	6 (0)	90	00:00:00.01	936	0
79	TABLE ACCESS BY INDEX ROWID	MOO_OPTY	90	1	3 (0)	90	00:00:00.01	360	0
* 80	INDEX UNIQUE SCAN	MOO_OPTY_PK1	90	1	2 (0)	90	00:00:00.01	270	0
81	TABLE ACCESS BY INDEX ROWID	ZCA_SALES_ACCOUNTS	90	1	3 (0)	90	00:00:00.01	576	0
* 82	INDEX RANGE SCAN	ZCA_SALES_ACCOUNTS_N4	90	1	2 (0)	90	00:00:00.01	270	0
* 83	TABLE ACCESS BY INDEX ROWID	ZCA_S_ACCT_RESOURCES	90	1	3 (0)	15	00:00:00.01	195	0
* 84	INDEX RANGE SCAN	ZCA_S_ACCT_RESOURCES_N2	90	1	1 (0)	15	00:00:00.01	180	0
* 85	TABLE ACCESS BY INDEX ROWID	JTF_RS_REP_MANAGERS	15	80	8 (0)	102	00:00:00.01	162	0
* 86	INDEX RANGE SCAN	JTF_RS_REP_MANAGERS_N1	15	38	2 (0)	933	00:00:00.01	60	0
* 87	INDEX UNIQUE SCAN	FND_TREE_VERSION_U1	102	1	1 (0)	102	00:00:00.01	48	0
* 88	TABLE ACCESS BY INDEX ROWID	FND_TREE_VERSION	102	1	2 (0)	102	00:00:00.01	9	0
89	NESTED LOOPS		90	1	203 (0)	1788	00:00:00.06	22650	1
90	NESTED LOOPS		90	59	85 (0)	3282	00:00:00.04	15942	1
91	NESTED LOOPS		90	10	26 (0)	180	00:00:00.04	13623	1
92	MERGE JOIN CARTESIAN		90	10	6 (0)	4320	00:00:00.01	570	1
93	TABLE ACCESS BY INDEX ROWID	MOO_REVN	90	1	4 (0)	90	00:00:00.01	360	0
* 94	INDEX RANGE SCAN	MOO_REVN_PSR1	90	1	3 (0)	90	00:00:00.01	270	0
95	BUFFER SORT		90	10	2 (0)	4320	00:00:00.01	210	1
* 96	INDEX RANGE SCAN	MOT_TERR_RESOURCES_N2	90	10	2 (0)	4320	00:00:00.01	210	1
* 97	TABLE ACCESS BY INDEX ROWID	MOT_TERRITORIES	4320	1	2 (0)	180	00:00:00.03	13053	0
* 98	INDEX UNIQUE SCAN	MOT_TERRITORIES_PK	4320	1	1 (0)	4320	00:00:00.02	8733	0
* 99	TABLE ACCESS BY INDEX ROWID	MOO_REVN_TERR	180	6	12 (0)	3282	00:00:00.01	2319	0
*100	INDEX RANGE SCAN	MOO_REVN_TERR_U1	180	18	3 (0)	3282	00:00:00.01	651	0
*101	INDEX RANGE SCAN	MOT_TERR_HIERARCHY_DN_V_N2	3282	1	2 (0)	1788	00:00:00.01	6708	0
*102	HASH JOIN		90	1	20 (5)	0	00:00:00.02	1779	0
103	NESTED LOOPS		90			1641	00:00:00.01	1587	0
104	NESTED LOOPS		90	6	16 (0)	1641	00:00:00.01	729	0
105	TABLE ACCESS BY INDEX ROWID	MOO_REVN	90	1	4 (0)	90	00:00:00.01	360	0
*106	INDEX RANGE SCAN	MOO_REVN_PSR1	90	1	3 (0)	90	00:00:00.01	270	0
*107	INDEX RANGE SCAN	MOO_REVN_TERR_U1	90	18	3 (0)	1641	00:00:00.01	369	0
*108	TABLE ACCESS BY INDEX ROWID	MOO_REVN_TERR	1641	6	12 (0)	1641	00:00:00.01	858	0
*109	INDEX RANGE SCAN	MOT_TERR_RESOURCES_N2	81	10	3 (0)	3888	00:00:00.01	192	0
110	NESTED LOOPS		90	1	306 (0)	2994	00:00:23.90	7676K	44
111	NESTED LOOPS		90	86	48 (0)	2519K	00:00:01.13	38772	44
112	NESTED LOOPS		90	10	28 (0)	180	00:00:00.02	14199	0
113	MERGE JOIN CARTESIAN		90	10	8 (0)	4320	00:00:00.01	1146	0
114	NESTED LOOPS		90	1	6 (0)	90	00:00:00.01	936	0
115	TABLE ACCESS BY INDEX ROWID	MOO_OPTY	90	1	3 (0)	90	00:00:00.01	360	0
*116	INDEX UNIQUE SCAN	MOO_OPTY_PK1	90	1	2 (0)	90	00:00:00.01	270	0
117	TABLE ACCESS BY INDEX ROWID	ZCA_SALES_ACCOUNTS	90	1	3 (0)	90	00:00:00.01	576	0
*118	INDEX RANGE SCAN	ZCA_SALES_ACCOUNTS_N4	90	1	2 (0)	90	00:00:00.01	270	0
119	BUFFER SORT		90	10	5 (0)	4320	00:00:00.01	210	0
*120	INDEX RANGE SCAN	MOT_TERR_RESOURCES_N2	90	10	2 (0)	4320	00:00:00.01	210	0
*121	TABLE ACCESS BY INDEX ROWID	MOT_TERRITORIES	4320	1	2 (0)	180	00:00:00.01	13053	0
*122	INDEX UNIQUE SCAN	MOT_TERRITORIES_PK	4320	1	1 (0)	4320	00:00:00.01	8733	0
*123	INDEX RANGE SCAN	MOT_TERR_HIERARCHY_DN_V_N1	180	9	2 (0)	2519K	00:00:00.73	24573	44
*124	INDEX RANGE SCAN	ZCA_S_ACCT_TERRITORIES_N3	2519K	1	3 (0)	2994	00:00:14.00	7637K	0
*125	HASH JOIN		90	1	16 (7)	0	00:00:00.04	1827	0
126	NESTED LOOPS		90			2934	00:00:00.02	1617	0
127	NESTED LOOPS		90	22	12 (0)	2934	00:00:00.01	1359	0
128	NESTED LOOPS		90	1	6 (0)	90	00:00:00.01	936	0
129	TABLE ACCESS BY INDEX ROWID	MOO_OPTY	90	1	3 (0)	90	00:00:00.01	360	0
*130	INDEX UNIQUE SCAN	MOO_OPTY_PK1	90	1	2 (0)	90	00:00:00.01	270	0
131	TABLE ACCESS BY INDEX ROWID	ZCA_SALES_ACCOUNTS	90	1	3 (0)	90	00:00:00.01	576	0
*132	INDEX RANGE SCAN	ZCA_SALES_ACCOUNTS_N4	90	1	2 (0)	90	00:00:00.01	270	0
*133	INDEX RANGE SCAN	ZCA_S_ACCT_TERRITORIES_N4	90	22	3 (0)	2934	00:00:00.01	423	0
134	TABLE ACCESS BY INDEX ROWID	ZCA_S_ACCT_TERRITORIES	2934	21	6 (0)	2934	00:00:00.01	258	0
*135	INDEX RANGE SCAN	MOT_TERR_RESOURCES_N2	90	10	3 (0)	4320	00:00:00.01	210	0
136	NESTED LOOPS		90			14595	00:00:00.16	19950	0
137	NESTED LOOPS		90	99	405 (0)	14595	00:00:00.14	19860	0
138	NESTED LOOPS		90	99	207 (0)	14595	00:00:00.12	19731	0
*139	TABLE ACCESS BY INDEX ROWID	MOO_OPTY_RESOURCES	90	23	23 (0)	2025	00:00:00.01	1212	0
*140	INDEX RANGE SCAN	MOO_OPTY_RESOURCES_N20	90	23	3 (0)	2025	00:00:00.01	276	0
*141	TABLE ACCESS BY INDEX ROWID	JTF_RS_REP_MANAGERS	2025	4	8 (0)	14595	00:00:00.08	18519	0
*142	INDEX RANGE SCAN	JTF_RS_REP_MANAGERS_N1	2025	38	2 (0)	102K	00:00:00.04	5553	0
*143	INDEX UNIQUE SCAN	FND_TREE_VERSION_U1	14595	1	1 (0)	14595	00:00:00.01	129	0
*144	TABLE ACCESS BY INDEX ROWID	FND_TREE_VERSION	14595	1	2 (0)	14595	00:00:00.01	90	0

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

   1 - filter(ROWNUM=1)
   2 - filter(("HCP"."STATUS"='A' AND "HCP"."CONTACT_LEVEL_TABLE"='HZ_CONTACT_POINTS' AND
              TRUNC(INTERNAL_FUNCTION("HCP"."PREFERENCE_START_DATE"))<=TRUNC(SYSDATE@!) AND TRUNC(INTERNAL_FUNCTION("HCP"."PREFERENCE_END_DATE"))>=TRUNC(SYSDATE@!)))
   3 - access("HCP"."CONTACT_LEVEL_TABLE_ID"=:B1)
  20 - storage("OPPORTUNITYEO"."STATUS_CD"=:BINDDEFAULTSTATUSCD)
       filter("OPPORTUNITYEO"."STATUS_CD"=:BINDDEFAULTSTATUSCD)
  21 - filter(("REVENUEEO"."EFFECTIVE_DATE">=:BINDEFFECTIVEBEGINDATE AND "REVENUEEO"."EFFECTIVE_DATE"<=:BINDEFFECTIVEENDDATE))
  22 - access("OPPORTUNITYEO"."SUM_REVN_ID"="REVENUEEO"."REVN_ID")
  23 - filter(("RELATIONSHIPPEO"."STATUS"=:VC_TEMP_4 AND "RELATIONSHIPPEO"."END_DATE">=:BINDSYSDATE AND "RELATIONSHIPPEO"."START_DATE"<=:BINDSYSDATE AND
              "OPPORTUNITYEO"."PR_CON_PARTY_ID"="RELATIONSHIPPEO"."SUBJECT_ID"))
  24 - access("OPPORTUNITYEO"."PR_CON_RELATIONSHIP_ID"="RELATIONSHIPPEO"."RELATIONSHIP_ID")
  25 - filter(("PHONEPEO"."END_DATE">=:BINDSYSDATE AND "PHONEPEO"."START_DATE"<=:BINDSYSDATE))
  26 - access("RELATIONSHIPPEO"."RELATIONSHIP_ID"="PHONEPEO"."RELATIONSHIP_ID" AND "RELATIONSHIPPEO"."SUBJECT_ID"="PHONEPEO"."OWNER_TABLE_ID" AND
              "PHONEPEO"."CONTACT_POINT_TYPE"='PHONE' AND "PHONEPEO"."PRIMARY_FLAG"=:VC_TEMP_5 AND "RELATIONSHIPPEO"."SUBJECT_TABLE_NAME"="PHONEPEO"."OWNER_TABLE_NAME"
              AND "PHONEPEO"."STATUS"='A')
  28 - access("OPPORTUNITYEO"."PR_CMPT_PARTY_ID"="PRIMARYCOMPETITORPARTYPEO"."PARTY_ID")
  30 - access("OPPORTUNITYEO"."OPTY_PARTY_SITE_ID"="PARTYSITEPEO"."PARTY_SITE_ID")
  32 - access("OPPORTUNITYEO"."PR_CON_PARTY_ID"="PRIMARYCONTACTPARTYPEO"."PARTY_ID")
  34 - access("OPPORTUNITYEO"."CUST_PARTY_ID"="PARTYPEO"."PARTY_ID")
  35 - filter(("CODEASSIGNMENTPEO"."STATUS"='A' AND "CODEASSIGNMENTPEO"."PRIMARY_FLAG"='Y'))
  36 - access("OPPORTUNITYEO"."CUST_PARTY_ID"="CODEASSIGNMENTPEO"."OWNER_TABLE_ID" AND "CODEASSIGNMENTPEO"."OWNER_TABLE_NAME"='HZ_PARTIES' AND
              "CODEASSIGNMENTPEO"."CLASS_CATEGORY"='ORGANIZATION_TYPE')
       filter(("CODEASSIGNMENTPEO"."CLASS_CATEGORY"='ORGANIZATION_TYPE' AND "CODEASSIGNMENTPEO"."OWNER_TABLE_NAME"='HZ_PARTIES'))
  38 - access("OPPORTUNITYEO"."CUST_PARTY_ID"="SALESACCOUNTPEO"."PARTY_ID")
  40 - access("PARTYSITEPEO"."LOCATION_ID"="LOCATIONPEO"."LOCATION_ID")
  42 - access("OPPORTUNITYEO"."OWNER_RESOURCE_ID"="OWNERRESOURCEPARTYPEO"."PARTY_ID")
  44 - filter(:BINDDEFAULTSTATUSCD="OPPORTUNITYEO"."STATUS_CD")
  46 - filter("B"."ENABLED_FLAG"='Y')
  47 - access("B"."LOOKUP_TYPE"='OPTY_STATUS' AND "B"."LOOKUP_CODE"="OPPORTUNITYEO"."STATUS_CD")
       filter("B"."LOOKUP_CODE"=:BINDDEFAULTSTATUSCD)
  48 - access("B"."LOOKUP_VALUES_ID"="T"."LOOKUP_VALUES_ID" AND "T"."LANGUAGE"=USERENV('LANG'))
  52 - access("T"."STG_ID"="OPPORTUNITYEO"."CURR_STG_ID" AND "T"."LANGUAGE"=USERENV('LANG'))
  54 - access("B"."STG_ID"="OPPORTUNITYEO"."CURR_STG_ID")
       filter("B"."STG_ID"="T"."STG_ID")
  58 - access("T"."SALES_METHOD_ID"="OPPORTUNITYEO"."SALES_METHOD_ID" AND "T"."LANGUAGE"=USERENV('LANG'))
  60 - access("B"."SALES_METHOD_ID"="OPPORTUNITYEO"."SALES_METHOD_ID")
       filter("B"."SALES_METHOD_ID"="T"."SALES_METHOD_ID")
  62 - filter(TO_DATE(:BINDEFFECTIVEBEGINDATE)<=TO_DATE(:BINDEFFECTIVEENDDATE))
  65 - filter("OPPORTUNITYRESOURCEEONQ1"."RESOURCE_ID"<>TO_NUMBER(:BINDLOGGEDINUSERRESOURCEID))
  66 - access("OPPORTUNITYRESOURCEEONQ1"."OPTY_ID"="OPPORTUNITYEO"."OPTY_ID")
  67 - access("REPORTINGMANAGERPEONQ1"."PARENT_RESOURCE_ID"=TO_NUMBER(:BINDLOGGEDINUSERRESOURCEID) AND
              "OPPORTUNITYRESOURCEEONQ1"."RESOURCE_ID"="REPORTINGMANAGERPEONQ1"."RESOURCE_ID")
       filter("REPORTINGMANAGERPEONQ1"."RESOURCE_ID"<>TO_NUMBER(:BINDLOGGEDINUSERRESOURCEID))
  68 - filter(("OPPORTUNITYRESOURCEEONQ1"."RESOURCE_ORG_ID"="REPORTINGMANAGERPEONQ1"."GROUP_ID" AND
              "OPPORTUNITYRESOURCEEONQ1"."ORG_TREE_CODE"="REPORTINGMANAGERPEONQ1"."TREE_CODE" AND
              "OPPORTUNITYRESOURCEEONQ1"."ORG_TREE_STRUCTURE_CODE"="REPORTINGMANAGERPEONQ1"."TREE_STRUCTURE_CODE"))
  72 - filter(("MYOPRES"."ACCESS_LEVEL_CODE"='100' OR "MYOPRES"."ACCESS_LEVEL_CODE"='200' OR "MYOPRES"."ACCESS_LEVEL_CODE"='300'))
  73 - access("MYOPRES"."OPTY_ID"="OPPORTUNITYEO"."OPTY_ID" AND "MYOPRES"."RESOURCE_ID"=:BindResourceID)
  80 - access("OPTY"."OPTY_ID"="OPPORTUNITYEO"."OPTY_ID")
  82 - access("OPTY"."CUST_PARTY_ID"="SALESACCNT"."PARTY_ID")
  83 - filter(("ACNTRES"."ORG_TREE_STRUCTURE_CODE" IS NOT NULL AND "ACNTRES"."ORG_TREE_CODE" IS NOT NULL AND "ACNTRES"."RESOURCE_ORG_ID" IS NOT NULL))
  84 - access("SALESACCNT"."SALES_ACCOUNT_ID"="ACNTRES"."SALES_ACCOUNT_ID")
  85 - filter(("MGRCHAIN"."PARENT_RESOURCE_ID"=:BindResourceID AND "ACNTRES"."ORG_TREE_STRUCTURE_CODE"="MGRCHAIN"."TREE_STRUCTURE_CODE" AND
              "ACNTRES"."ORG_TREE_CODE"="MGRCHAIN"."TREE_CODE"))
  86 - access("ACNTRES"."RESOURCE_ORG_ID"="MGRCHAIN"."GROUP_ID")
  87 - access("TREE_STRUCTURE_CODE"="MGRCHAIN"."TREE_STRUCTURE_CODE" AND "TREE_CODE"="MGRCHAIN"."TREE_CODE" AND
              "TREE_VERSION_ID"="MGRCHAIN"."TREE_VERSION_ID" AND "ENTERPRISE_ID"=DECODE(SYS_CONTEXT('FND_VPD_CTX','ENTERPRISE_ID'),NULL,1,'0',1,TO_NUMBER(SYS_CONTEXT('FN
              D_VPD_CTX','ENTERPRISE_ID'))))
  88 - filter(("EFFECTIVE_START_DATE"<=SYSDATE@! AND "EFFECTIVE_END_DATE">=SYSDATE@!))
  94 - access("REVN"."OPTY_ID"="OPPORTUNITYEO"."OPTY_ID" AND "REVN"."PRIMARY_FLAG"='Y')
       filter("REVN"."PRIMARY_FLAG"='Y')
  96 - access("TERR_RES"."RESOURCE_ID"=:BindResourceID)
  97 - filter(("TERR"."LATEST_VERSION_FLAG"='Y' AND "TERR"."STATUS_CODE"='FINALIZED'))
  98 - access("TERR_RES"."TERRITORY_VERSION_ID"="TERR"."TERRITORY_VERSION_ID")
  99 - filter("REVNTERR"."SUM_REVN_FLAG"='Y')
 100 - access("REVNTERR"."REVN_ID"="REVN"."REVN_ID")
 101 - access("TERRDN"."DESCENDANT_TERRITORY_ID"="REVNTERR"."TERRITORY_ID" AND "TERR"."TERRITORY_ID"="TERRDN"."TERRITORY_ID")
       filter("TERRDN"."DESCENDANT_TERRITORY_ID"<>"TERRDN"."TERRITORY_ID")
 102 - access("MOT_TERR_RESOURCES"."TERRITORY_VERSION_ID"="MOO_REVN_TERR"."TERRITORY_VERSION_ID")
 106 - access("MOO_REVN"."OPTY_ID"="OPPORTUNITYEO"."OPTY_ID" AND "MOO_REVN"."PRIMARY_FLAG"='Y')
       filter("MOO_REVN"."PRIMARY_FLAG"='Y')
 107 - access("MOO_REVN_TERR"."REVN_ID"="MOO_REVN"."REVN_ID")
```

```
108 - filter("MOO_REVN_TERR"."SUM_REVN_FLAG"='Y')
109 - access("MOT_TERR_RESOURCES"."RESOURCE_ID"=:BindResourceID)
116 - access("MOO_OPTY"."OPTY_ID"="OPPORTUNITYEO"."OPTY_ID")
118 - access("ZCA_SALES_ACCOUNTS"."PARTY_ID"="MOO_OPTY"."CUST_PARTY_ID")
120 - access("MOT_TERR_RESOURCES"."RESOURCE_ID"=:BindResourceID)
121 - filter(("MOT_TERRITORIES"."LATEST_VERSION_FLAG"='Y' AND "MOT_TERRITORIES"."STATUS_CODE"='FINALIZED'))
122 - access("MOT_TERR_RESOURCES"."TERRITORY_VERSION_ID"="MOT_TERRITORIES"."TERRITORY_VERSION_ID")
123 - access("MOT_TERRITORIES"."TERRITORY_ID"="MOT_TERR_HIERARCHY_DN"."TERRITORY_ID")
      filter("MOT_TERR_HIERARCHY_DN"."DESCENDANT_TERRITORY_ID"<>"MOT_TERR_HIERARCHY_DN"."TERRITORY_ID")
124 - access("MOT_TERR_HIERARCHY_DN"."DESCENDANT_TERRITORY_ID"="ZCA_S_ACCT_TERRITORIES"."TERRITORY_ID" AND
            "ZCA_S_ACCT_TERRITORIES"."SALES_ACCOUNT_ID"="ZCA_SALES_ACCOUNTS"."SALES_ACCOUNT_ID")
125 - access("MOT_TERR_RESOURCES"."TERRITORY_VERSION_ID"="ZCA_S_ACCT_TERRITORIES"."TERRITORY_VERSION_ID")
130 - access("MOO_OPTY"."OPTY_ID"="OPPORTUNITYEO"."OPTY_ID")
132 - access("ZCA_SALES_ACCOUNTS"."PARTY_ID"="MOO_OPTY"."CUST_PARTY_ID")
133 - access("ZCA_S_ACCT_TERRITORIES"."SALES_ACCOUNT_ID"="ZCA_SALES_ACCOUNTS"."SALES_ACCOUNT_ID")
135 - access("MOT_TERR_RESOURCES"."RESOURCE_ID"=:BindResourceID)
139 - filter(("ORES"."ACCESS_LEVEL_CODE"='100' OR "ORES"."ACCESS_LEVEL_CODE"='200' OR "ORES"."ACCESS_LEVEL_CODE"='300'))
140 - access("ORES"."OPTY_ID"="OPPORTUNITYEO"."OPTY_ID")
141 - filter(("MGRCHAIN"."PARENT_RESOURCE_ID"=:BindResourceID AND "ORES"."ORG_TREE_STRUCTURE_CODE"="MGRCHAIN"."TREE_STRUCTURE_CODE" AND
            "ORES"."ORG_TREE_CODE"="MGRCHAIN"."TREE_CODE"))
142 - access("ORES"."RESOURCE_ORG_ID"="MGRCHAIN"."GROUP_ID")
143 - access("TREE_STRUCTURE_CODE"="MGRCHAIN"."TREE_STRUCTURE_CODE" AND "TREE_CODE"="MGRCHAIN"."TREE_CODE" AND
            "TREE_VERSION_ID"="MGRCHAIN"."TREE_VERSION_ID" AND "ENTERPRISE_ID"=DECODE(SYS_CONTEXT('FND_VPD_CTX','ENTERPRISE_ID'),NULL,1,'0',1,TO_NUMBER(SYS_CONTEXT('FN
            D_VPD_CTX','ENTERPRISE_ID'))))
144 - filter(("EFFECTIVE_START_DATE"<=SYSDATE@! AND "EFFECTIVE_END_DATE">=SYSDATE@!))
```

Figure 4. Execution Plan Chosen by Optimizer

2.1.1 Issues Affecting Existing SQLs' Performance

In SQL tuning exercises, the first aim is to identify the issues in the execution plan and the reasons for the optimizer to choose the plan. It is not always an easy task but needs careful understanding of the business functionality that the SQL is trying to achieve as well as a deep understanding of the various operations that the optimizer uses. It also needs constant updating of one's knowledge of the optimizer features and behaviour changes that come with changed and new database parameters. Needless to say, it can become quite complex.

As seen in the performance metrics above in Figure 3, the performance of the SQL is quite poor, both in terms of time taken as well as the number of buffer gets that the optimizer had to process. Even though the plan cost came up low at 145, it is apparent that the optimizer did not do a very good job at estimations. The benchmark metrics show that the database took more than 5 seconds warm time with an extremely high 2.5 million buffer gets to return 25 rows of the result set.

The execution plan shows that even though the outer query processed very few qualifying rows, the optimizer had to evaluate all the security predicates' sub-queries to find out which rows the user was entitled to see, only to eliminate most of them later, based on the outer query. This is a typical "tiny-huge, huge-tiny" problem. It means that the optimizer has to make a quick decision whether to drive from the outer query or drive from the security predicate sub-queries. Which one of these will result in more restrictive row sets is always a difficult question to answer. This decision making is complicated by the fact that the SQL has many tables and joins and it uses bind variables which make estimating the cardinality that much more complicated in the process of deciding on an execution plan.

The analysis of the SQL, execution plan and schema helped in narrowing down to broadly three issues. The three broad areas affecting the existing SQL's performance are the security predicates' sub-queries and the associated "tiny-huge, huge-tiny" problem, the main filter predicate columns coming from different tables and lastly inadequacy of existing indexes.

The two main filtering predicates in the outer SQL are:

```
AND ( ( (OpportunityEO.STATUS_CD        = :BindDefaultStatusCd )

AND (RevenueEO.EFFECTIVE_DATE BETWEEN :BindEffectiveBeginDate AND :BindEffectiveEndDate ) ) )
```

As seen above, the Effective_Date column is from the Revenue table while Status_Cd column is from the Opportunity table. This makes the optimizer's job difficult.

The "tiny-huge, huge-tiny" problem is partly due to the fact that the data security sub-queries do not have any of the two filtering predicates that the outer query has i.e. EFFECTIVE_DATE and STATUS_CD.

Finally, there can be additional indexes that may help performance.

3. SUGGESTED APPROACH TO IMPROVE SQL PERFORMANCE

The resolutions for the three issues identified above can now be easily tried and tested using the same Java benchmark tool.

First, let us try to resolve the issue of filter predicate columns coming from different tables. To make the optimizer's job a little easier, it is sometimes a good idea to de-normalize such filter predicate columns. In the present case, adding EFFECTIVE_DATE column to MOO_OPTY table and adding EFFECTIVE_DATE, STATUS_CD to MOO_OPTY_RESOURCES table will make the optimizer's choices less complex and as a result more stable and predictable. In addition, appropriate indexes will also be needed on these columns. Thus, the outer query would now have the following filter predicates from the same Opportunity table:

```
AND (((OpportunityEO.STATUS_CD = :BindDefaultStatusCd)

AND (OpportunityEO.EFFECTIVE_DATE BETWEEN :BindEffectiveBeginDate AND :BindEffectiveEndDate ) ) )
```

The resolution for the "tiny-huge, huge-tiny" issue is a little trickier. One approach that has worked consistently well is to create a wrapper around the data security predicates' sub-queries, use DISTINCT clause in the wrapper as well as data security sub-queries and also push the outer query's filter predicates into the data security wrapper code. The changed outer SQL would thus be as in Figure 5 below and the changed data security predicates sub-queries would be rewritten as in Figure 6 below.

```
SELECT
  /*+  FIRST_ROWS(10)  */
   Column1,
  Column2
  .
  .
  .
  .
  .
ColumnN
FROM MOO_OPTY OpportunityEO,
  HZ_PARTIES PartyPEO,
  MOO_SALES_METHOD_VL SalesMethodEO,
  MOO_STG_VL SalesStageEO,
  MOO_REVN RevenueEO,
  HZ_PARTY_SITES PartySitePEO,
  HZ_PARTIES PrimaryCompetitorPartyPEO,
  HZ_LOCATIONS LocationPEO,
  HZ_PARTIES OwnerResourcePartyPEO,
  ZBS_LOOKUP_VALUES_VL LookupValuesEO,
  HZ_PARTIES PrimaryContactPartyPEO,
  ZCA_SALES_ACCOUNTS SalesAccountPEO,
  HZ_RELATIONSHIPS RelationshipPEO,
  HZ_CONTACT_POINTS PhonePEO,
  HZ_CODE_ASSIGNMENTS CodeAssignmentPEO
WHERE OpportunityEO.CUST_PARTY_ID          = PartyPEO.PARTY_ID(+)
AND OpportunityEO.SALES_METHOD_ID          = SalesMethodEO.SALES_METHOD_ID(+)
AND OpportunityEO.CURR_STG_ID              = SalesStageEO.STG_ID(+)
AND OpportunityEO.SUM_REVN_ID              = RevenueEO.REVN_ID
AND OpportunityEO.OPTY_PARTY_SITE_ID       = PartySitePEO.PARTY_SITE_ID(+)
AND OpportunityEO.PR_CMPT_PARTY_ID         = PrimaryCompetitorPartyPEO.PARTY_ID(+)
AND PartySitePEO.LOCATION_ID               = LocationPEO.LOCATION_ID(+)
AND OpportunityEO.OWNER_RESOURCE_ID        = OwnerResourcePartyPEO.PARTY_ID(+)
AND OpportunityEO.STATUS_CD                = LookupValuesEO.LOOKUP_CODE(+)
AND ('Y')                                  = LookupValuesEO.ENABLED_FLAG(+)
AND ('OPTY_STATUS')                        = LookupValuesEO.LOOKUP_TYPE(+)
AND OpportunityEO.PR_CON_PARTY_ID          = PrimaryContactPartyPEO.PARTY_ID(+)
AND OpportunityEO.CUST_PARTY_ID            = SalesAccountPEO.PARTY_ID(+)
AND OpportunityEO.PR_CON_RELATIONSHIP_ID = RelationshipPEO.RELATIONSHIP_ID(+)
AND OpportunityEO.PR_CON_PARTY_ID          = RelationshipPEO.SUBJECT_ID(+)
AND RelationshipPEO.RELATIONSHIP_ID        = PhonePEO.RELATIONSHIP_ID(+)
AND RelationshipPEO.SUBJECT_ID             = PhonePEO.OWNER_TABLE_ID(+)
AND RelationshipPEO.SUBJECT_TABLE_NAME     = PhonePEO.OWNER_TABLE_NAME(+)
AND ('PHONE')                              = PhonePEO.CONTACT_POINT_TYPE(+)
AND ('A')                                  = PhonePEO.STATUS(+)
AND OpportunityEO.CUST_PARTY_ID            = CodeAssignmentPEO.OWNER_TABLE_ID(+)
AND ('ORGANIZATION_TYPE')                  = CodeAssignmentPEO.CLASS_CATEGORY(+)
AND ('HZ_PARTIES')                         = CodeAssignmentPEO.OWNER_TABLE_NAME(+)
AND ('Y')                                  = CodeAssignmentPEO.PRIMARY_FLAG(+)
AND ('A')                                  = CodeAssignmentPEO.STATUS(+)
AND ( ( (EXISTS
  (SELECT  1
  FROM MOO_OPTY_RESOURCES OpportunityResourceEONQ1,
    JTF_RS_REP_MANAGERS ReportingManagerPEONQ1
  WHERE (((OpportunityResourceEONQ1.RESOURCE_ID        = ReportingManagerPEONQ1.RESOURCE_ID(+))
  AND (OpportunityResourceEONQ1.RESOURCE_ORG_ID        = ReportingManagerPEONQ1.GROUP_ID(+)))
  AND (OpportunityResourceEONQ1.ORG_TREE_CODE          = ReportingManagerPEONQ1.TREE_CODE(+)))
  AND (OpportunityResourceEONQ1.ORG_TREE_STRUCTURE_CODE = ReportingManagerPEONQ1.TREE_STRUCTURE_CODE(+))
  AND ( ( (OpportunityResourceEONQ1.RESOURCE_ID       <> :BindLoggedInUserResourceId )
  AND (ReportingManagerPEONQ1.PARENT_RESOURCE_ID       = :BindLoggedInUserResourceId ) ) )
  AND (OpportunityEO.OPTY_ID                           = OpportunityResourceEONQ1.OPTY_ID)
  AND (OpportunityResourceEONQ1.OPTY_STATUS_CD = :BindDefaultStatusCd)
  AND (OpportunityResourceEONQ1.OPTY_EFFECTIVE_DATE BETWEEN :BindEffectiveBeginDate AND :BindEffectiveEndDate)
  )) ) )
 AND ( ( (OpportunityEO.STATUS_CD        = :BindDefaultStatusCd )
 AND (OpportunityEO.EFFECTIVE_DATE BETWEEN :BindEffectiveBeginDate AND :BindEffectiveEndDate ) ) )
AND ( ( (RelationshipPEO.STATUS(+)    = :vc_temp_4 )
AND (RelationshipPEO.START_DATE(+) <= :BindSysdate )
AND (RelationshipPEO.END_DATE(+)   >= :BindSysdate ) ) )
AND ( ( (PhonePEO.END_DATE(+)      >= :BindSysdate )
AND (PhonePEO.START_DATE(+)        <= :BindSysdate )
AND (PhonePEO.PRIMARY_FLAG(+)       = :vc_temp_5 ) ) ) )
AND (
(
(OpportunityEO.opty_id in
(select  distinct  OpportunityEO1.opty_id from FUSION.MOO_OPTY OpportunityEO1
```

Effective_Date & Status_Cd columns Denormalized to MOO_OPTY_RESOURCES table and added as predicates here

Effective_Date denormalized to MOO_OPTY table and used here instead of from MOO_REVN table

Figure 5. Rewritten Outer Query Using De-normalized Columns

```
AND (PhonePEO.PRIMARY_FLAG(+)       = :vc_temp_5 ) ) ) )
AND ( ((OpportunityEO.opty_id in
(select  distinct  OpportunityEO1.opty_id from FUSION.MOO_OPTY OpportunityEO1
where  OpportunityEO1.EFFECTIVE_DATE BETWEEN :BindEffectiveBeginDate AND :BindEffectiveEndDate
 and OpportunityEO1.STATUS_CD = :vc_temp_1
 and (( OpportunityEO1.opty_id         IN
  (SELECT distinct opty.opty_id
    FROM FUSION.ZCA_S_ACCT_RESOURCES acntres,
         FUSION.ZCA_SALES_ACCOUNTS salesaccnt,
         FUSION.moo_opty opty,
         FUSION.Jtf_rs_rep_managers mgrchain ,
         FUSION.Fnd_tree_version ftv
    WHERE opty.cust_party_id         = salesaccnt.party_id
    AND salesaccnt.sales_account_id = acntres.sales_account_id
    AND SYSDATE BETWEEN ftv.effective_start_date AND ftv.effective_end_date
    AND ftv.tree_version_id          = mgrchain.tree_version_id
    AND ftv.tree_structure_code      = mgrchain.tree_structure_code
    AND ftv.tree_code                = mgrchain.tree_code
    AND acntres.org_tree_structure_code = mgrchain.tree_structure_code
    AND acntres.org_tree_code           = mgrchain.tree_code
    AND acntres.resource_org_id         = mgrchain.group_id
    AND mgrchain.parent_resource_id  = ( :BindResourceID ) ) )
OR ( OpportunityEO1.opty_id IN
  (SELECT distinct  ores.opty_id
    FROM fusion.Moo_opty_resources ores,
             fusion.Jtf_rs_rep_managers mgrchain ,
         fusion.Fnd_tree_version ftv
    WHERE ores.access_level_code IN ('100', '200', '300')
        AND   ores.resource_org_id         = mgrchain.group_id
        AND   ores.org_tree_structure_code = mgrchain.tree_structure_code
    AND   ores.org_tree_code           = mgrchain.tree_code
    AND   SYSDATE BETWEEN ftv.effective_start_date AND ftv.effective_end_date
    AND   ftv.tree_version_id          = mgrchain.tree_version_id
    AND   ftv.tree_structure_code      = mgrchain.tree_structure_code
    AND   ftv.tree_code                = mgrchain.tree_code
    AND   mgrchain.parent_resource_id  = ( :BindResourceID )
  ) )
OR ( OpportunityEO1.opty_id IN
  (SELECT distinct  MOO_OPTY.opty_id
  FROM FUSION.MOT_TERR_RESOURCES,
    FUSION.ZCA_S_ACCT_TERRITORIES,
    FUSION.ZCA_SALES_ACCOUNTS,
    FUSION.MOO_OPTY
  WHERE MOT_TERR_RESOURCES.resource_id = (:BindResourceID)
  AND MOT_TERR_RESOURCES.territory_version_id = ZCA_S_ACCT_TERRITORIES.territory_version_id
  AND ZCA_S_ACCT_TERRITORIES.sales_account_id = ZCA_SALES_ACCOUNTS.sales_account_id
  AND ZCA_SALES_ACCOUNTS.party_id              = MOO_OPTY.cust_party_id
  )
OR OpportunityEO1.opty_id IN
  (SELECT distinct  MOO_OPTY.opty_id
  FROM FUSION.MOT_TERR_RESOURCES,
    FUSION.MOT_TERRITORIES,
    FUSION.MOT_TERR_HIERARCHY_DN,
    FUSION.ZCA_S_ACCT_TERRITORIES,
    FUSION.ZCA_SALES_ACCOUNTS,
    FUSION.MOO_OPTY
  WHERE MOT_TERR_RESOURCES.resource_id = (:BindResourceID)
  AND MOT_TERR_RESOURCES.territory_version_id       = MOT_TERRITORIES.territory_version_id
  AND MOT_TERRITORIES.latest_version_flag           = 'Y'
  AND MOT_TERRITORIES.status_code                   = 'FINALIZED'
  AND MOT_TERRITORIES.territory_id                  = MOT_TERR_HIERARCHY_DN.territory_id
  AND MOT_TERR_HIERARCHY_DN.descendant_territory_id != MOT_TERR_HIERARCHY_DN.territory_id
  AND MOT_TERR_HIERARCHY_DN.descendant_territory_id  = ZCA_S_ACCT_TERRITORIES.territory_id
  AND ZCA_S_ACCT_TERRITORIES.sales_account_id       = ZCA_SALES_ACCOUNTS.sales_account_id
  AND ZCA_SALES_ACCOUNTS.party_id                   = MOO_OPTY.cust_party_id
  ) )
OR ( OpportunityEO1.opty_id IN
  (SELECT distinct  MOO_REVN.opty_id
  FROM FUSION.MOT_TERR_RESOURCES,
    FUSION.MOO_REVN_TERR,
    FUSION.MOO_REVN
  WHERE MOT_TERR_RESOURCES.resource_id = (:BindResourceID)
  AND MOT_TERR_RESOURCES.territory_version_id = MOO_REVN_TERR.territory_version_id
  AND MOO_REVN_TERR.SUM_REVN_FLAG              = 'Y'
  AND MOO_REVN_TERR.revn_id                    = MOO_REVN.revn_id
  AND MOO_REVN.PRIMARY_FLAG                    = 'Y'
  )
OR OpportunityEO1.opty_id IN
   (SELECT distinct  revn.opty_id
  FROM FUSION.MOT_TERR_RESOURCES terr_res,
    FUSION.MOT_TERRITORIES terr,
    FUSION.MOT_TERR_HIERARCHY_DN terrdn,
    FUSION.MOO_REVN_TERR revnterr,
    FUSION.MOO_REVN revn
  WHERE terr_res.resource_id = (:BindResourceID)
  AND terr_res.territory_version_id   = terr.territory_version_id
  AND terr.latest_version_flag        = 'Y'
  AND terr.status_code                = 'FINALIZED'
  AND terr.territory_id               = terrdn.territory_id
  AND terrdn.descendant_territory_id != terrdn.territory_id
  AND terrdn.descendant_territory_id  = revnterr.territory_id
  AND revnterr.SUM_REVN_FLAG          = 'Y'
  AND revnterr.revn_id                = revn.revn_id
  AND revn.primary_flag               = 'Y'
  ) )
OR ( OpportunityEO1.opty_id IN
  (SELECT distinct  myopres.opty_id
  FROM Moo_opty_resources myopres
  WHERE myopres.resource_id = (:BindResourceID)
  AND myopres.access_level_code IN ('100', '200', '300')))))))
```

Wrapper code using DISTINCT clasue and outer query's filter predicates on EFFECTIVE_DATE and STATUS_CD

Use of DISTINCT clause in data security sub-queries

Figure 6. Rewritten Data Security Sub-Queries with Wrapper Code & DISTINCT Clause

Lastly, to support the above filter predicate changes and the rewritten data security sub-queries, the following indexes were created:

MOO_OPTY (OWNER_RESOURCE_ID, EFFECTIVE_DATE, STATUS_CD, SUM_REVN_ID)

MOO_OPTY (OPTY_ID, SUM_REVN_ID)

MOO_OPTY_RESOURCES (RESOURCE_ID, EFFECTIVE_DATE, STATUS_CD, OPTY_ID)

MOO_REVN (OPTY_ID, OWNER_RESOURCE_ID, EFFECTIVE_DATE, PRIMARY_FLAG, STATUS_CODE)

3.1. Benchmark Metrics and Execution Plan of Rewritten SQL

The three-pronged strategy described above worked very well and when the SQL was benchmarked using the same Java tool against the same database, results were drastically improved both in terms of warm time as well as the buffer gets. As seen in Figure 7 below, the warm time was only 365 milliseconds and buffer gets were down to 9697.

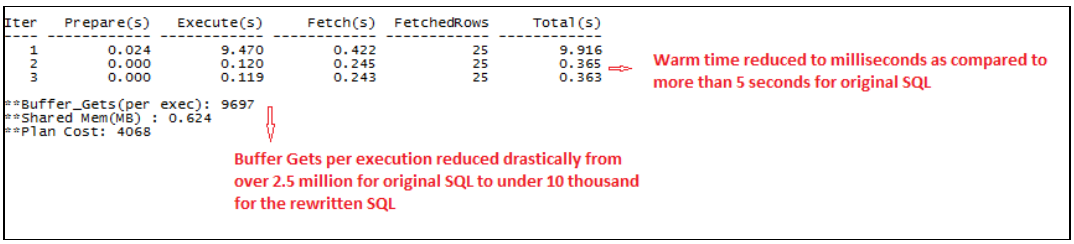

Figure 7. SQL Benchmark Metrics for Rewritten SQL

The execution plan for the rewritten SQL is as seen in Figure 8 below. Notice the much reduced A-Rows column as well as reduced buffers in the plan, which shows the levelling of the outer and data security sub-queries' cardinality achieved using the suggested approach.

Id	Operation	Name	Starts	E-Rows	Cost (%CPU)	A-Rows	A-Time	Buffers	Reads
0	SELECT STATEMENT		3		4068 (100)	90	00:00:00.33	29037	126
* 1	COUNT STOPKEY		18			3	00:00:00.01	48	0
* 2	TABLE ACCESS BY INDEX ROWID	HZ_CONTACT_PREFERENCES	18	1	4 (0)	3	00:00:00.01	48	0
* 3	INDEX RANGE SCAN	HZ_CONTACT_PREFERENCES_N1	18	1	3 (0)	3	00:00:00.01	45	0
4	NESTED LOOPS SEMI		3	1	4068 (1)	90	00:00:00.33	29037	126
5	NESTED LOOPS OUTER		3	1	4042 (1)	90	00:00:00.22	28179	93
6	NESTED LOOPS OUTER		3	1	4040 (1)	90	00:00:00.22	27963	93
7	NESTED LOOPS SEMI		3	1	4038 (1)	90	00:00:00.22	27747	93
8	NESTED LOOPS OUTER		3	4	104 (0)	90	00:00:00.13	2445	93
9	NESTED LOOPS OUTER		3	4	96 (0)	90	00:00:00.13	2397	93
10	NESTED LOOPS OUTER		3	4	84 (0)	90	00:00:00.04	2151	54
11	NESTED LOOPS OUTER		3	4	76 (0)	90	00:00:00.03	1854	53
12	NESTED LOOPS		3	4	68 (0)	90	00:00:00.03	1854	53
13	NESTED LOOPS OUTER		3	4	60 (0)	90	00:00:00.02	1575	53
14	NESTED LOOPS OUTER		3	4	52 (0)	90	00:00:00.01	1281	47
15	NESTED LOOPS OUTER		3	4	40 (0)	90	00:00:00.01	777	47
16	NESTED LOOPS OUTER		3	4	32 (0)	90	00:00:00.01	690	47
17	NESTED LOOPS OUTER		3	4	23 (0)	90	00:00:00.01	627	31
18	NESTED LOOPS OUTER		3	4	19 (0)	90	00:00:00.01	564	24
19	NESTED LOOPS OUTER		3	4	17 (0)	90	00:00:00.01	564	24
20	TABLE ACCESS BY INDEX ROWID	MOO_OPTY	3	5	7 (0)	90	00:00:00.01	462	0
* 21	INDEX RANGE SCAN	MOO_OPTY_PSR1	3		3 (0)	90	00:00:00.01	15	0
* 22	TABLE ACCESS BY INDEX ROWID	HZ_RELATIONSHIPS	90	1	5 (0)	18	00:00:00.02	102	24
* 23	INDEX RANGE SCAN	HZ_RELATIONSHIPS_U1	90	2	3 (0)	36	00:00:00.01	66	12
24	TABLE ACCESS BY INDEX ROWID	HZ_PARTY_SITES	90	1	2 (0)	0	00:00:00.01	0	0
* 25	INDEX UNIQUE SCAN	HZ_PARTY_SITES_PK	90	1	1 (0)	0	00:00:00.01	0	0
26	TABLE ACCESS BY INDEX ROWID	HZ_PARTIES	90	1	2 (0)	18	00:00:00.01	63	7
* 27	INDEX UNIQUE SCAN	HZ_PARTIES_PK	90	1	1 (0)	18	00:00:00.01	45	7
* 28	TABLE ACCESS BY INDEX ROWID	HZ_CONTACT_POINTS	90	1	3 (0)	15	00:00:00.01	63	16
* 29	INDEX RANGE SCAN	HZ_CONTACT_POINTS_N1	90	1	2 (0)	15	00:00:00.01	48	11
30	TABLE ACCESS BY INDEX ROWID	HZ_PARTIES	90	1	2 (0)	42	00:00:00.01	87	0
* 31	INDEX UNIQUE SCAN	HZ_PARTIES_PK	90	1	1 (0)	42	00:00:00.01	45	0
32	TABLE ACCESS BY INDEX ROWID	ZCA_SALES_ACCOUNTS	90	1	3 (0)	90	00:00:00.01	504	0
* 33	INDEX RANGE SCAN	ZCA_SALES_ACCOUNTS_N4	90	1	2 (0)	90	00:00:00.01	198	0
34	TABLE ACCESS BY INDEX ROWID	HZ_PARTIES	90	1	2 (0)	90	00:00:00.02	294	6
* 35	INDEX UNIQUE SCAN	HZ_PARTIES_PK	90	1	1 (0)	90	00:00:00.02	192	6
36	TABLE ACCESS BY INDEX ROWID	MOO_REVN	90	1	2 (0)	90	00:00:00.01	279	0
* 37	INDEX UNIQUE SCAN	MOO_REVN_PK	90	1	1 (0)	90	00:00:00.01	192	0
38	TABLE ACCESS BY INDEX ROWID	HZ_LOCATIONS	90	1	2 (0)	0	00:00:00.01	0	0
* 39	INDEX UNIQUE SCAN	HZ_LOCATIONS_PK	90	1	1 (0)	0	00:00:00.01	0	0
40	TABLE ACCESS BY INDEX ROWID	HZ_PARTIES	90	1	2 (0)	90	00:00:00.01	297	1
* 41	INDEX UNIQUE SCAN	HZ_PARTIES_PK	90	1	1 (0)	90	00:00:00.01	192	1
* 42	TABLE ACCESS BY INDEX ROWID	HZ_CODE_ASSIGNMENTS	90	1	3 (0)	33	00:00:00.04	246	39
* 43	INDEX RANGE SCAN	HZ_CODE_ASSIGNMENTS_U1	90	1	2 (0)	36	00:00:00.04	210	39
44	VIEW PUSHED PREDICATE	ZBS_LOOKUP_VALUES_VL	90	1	2 (0)	90	00:00:00.01	48	0
* 45	FILTER		90			90	00:00:00.01	48	0
46	NESTED LOOPS		90	1	2 (0)	90	00:00:00.01	48	0
* 47	TABLE ACCESS BY INDEX ROWID	ZBS_LOOKUP_VALUES_B	90	1	2 (0)	90	00:00:00.01	30	0
* 48	INDEX RANGE SCAN	ZBS_LOOKUP_VALUES_B_U2	90	1	1 (0)	90	00:00:00.01	21	0
* 49	INDEX UNIQUE SCAN	ZBS_LOOKUP_VALUES_TL_U1	90	1	0 (0)	90	00:00:00.01	18	0
50	VIEW PUSHED PREDICATE	VW_NSO_2	90	1	984 (1)	90	00:00:00.09	25302	0
* 51	FILTER		90			90	00:00:00.09	25302	0
52	NESTED LOOPS		90	1	984 (1)	90	00:00:00.09	25302	0
* 53	TABLE ACCESS BY INDEX ROWID	MOO_OPTY	90	1	3 (0)	90	00:00:00.01	639	0
* 54	INDEX UNIQUE SCAN	MOO_OPTY_PK1	90	1	2 (0)	90	00:00:00.01	192	0
* 55	VIEW	VW_NSO_1	90	1	981 (1)	90	00:00:00.09	24663	0
56	UNION-ALL		90			90	00:00:00.09	24663	0
57	SORT UNIQUE		90	1	6 (17)	0	00:00:00.01	360	0
* 58	TABLE ACCESS BY INDEX ROWID	MOO_OPTY_RESOURCES	90	1	5 (0)	0	00:00:00.01	360	0
* 59	INDEX RANGE SCAN	MOO_OPTY_RESOURCES_U1	90	1	4 (0)	0	00:00:00.01	360	0
60	SORT UNIQUE		90	1	20 (5)	0	00:00:00.01	1116	0
61	NESTED LOOPS		90			0	00:00:00.01	1116	0
62	NESTED LOOPS		90	1	19 (0)	0	00:00:00.01	1116	0
63	NESTED LOOPS		90	1	17 (0)	0	00:00:00.01	1116	0
64	NESTED LOOPS		90	1	9 (0)	0	00:00:00.01	1116	0
65	NESTED LOOPS		90	1	6 (0)	90	00:00:00.01	936	0
66	TABLE ACCESS BY INDEX ROWID	MOO_OPTY	90	1	3 (0)	90	00:00:00.01	360	0
* 67	INDEX UNIQUE SCAN	MOO_OPTY_PK1	90	1	2 (0)	90	00:00:00.01	270	0
68	TABLE ACCESS BY INDEX ROWID	ZCA_SALES_ACCOUNTS	90	1	3 (0)	90	00:00:00.01	576	0
* 69	INDEX RANGE SCAN	ZCA_SALES_ACCOUNTS_N4	90	1	2 (0)	90	00:00:00.01	270	0
* 70	TABLE ACCESS BY INDEX ROWID	ZCA_S_ACCT_RESOURCES	90	1	3 (0)	0	00:00:00.01	180	0
* 71	INDEX RANGE SCAN	ZCA_S_ACCT_RESOURCES_N2	90	1	1 (0)	0	00:00:00.01	180	0
* 72	TABLE ACCESS BY INDEX ROWID	JTF_RS_REP_MANAGERS	0	80	8 (0)	0	00:00:00.01	0	0
* 73	INDEX RANGE SCAN	JTF_RS_REP_MANAGERS_N1	0	38	2 (0)	0	00:00:00.01	0	0
* 74	INDEX UNIQUE SCAN	FND_TREE_VERSION_U1	0	1	1 (0)	0	00:00:00.01	0	0
* 75	TABLE ACCESS BY INDEX ROWID	FND_TREE_VERSION	0	1	2 (0)	0	00:00:00.01	0	0
76	SORT UNIQUE		90	1	204 (1)	90	00:00:00.07	23187	0
77	NESTED LOOPS		90	1	203 (0)	2139	00:00:00.08	23187	0
78	NESTED LOOPS		90	59	85 (0)	3660	00:00:00.12	15759	0
79	NESTED LOOPS		90	10	26 (0)	180	00:00:00.03	13623	0
* 80	INDEX UNIQUE SCAN	MOO_OPTY_PK1	90	1	2 (0)	90	00:00:00.01	216	0
81	TABLE ACCESS BY INDEX ROWID	ZCA_SALES_ACCOUNTS	90	1	3 (0)	90	00:00:00.01	597	0
* 82	INDEX RANGE SCAN	ZCA_SALES_ACCOUNTS_N4	90	1	2 (0)	90	00:00:00.01	270	0
* 83	TABLE ACCESS BY INDEX ROWID	ZCA_S_ACCT_RESOURCES	90	1	3 (0)	8	00:00:00.01	191	0
* 84	INDEX RANGE SCAN	ZCA_S_ACCT_RESOURCES_N2	90	1	1 (0)	11	00:00:00.01	180	0
* 85	TABLE ACCESS BY INDEX ROWID	JTF_RS_REP_MANAGERS	8	80	8 (0)	19	00:00:00.01	80	1
* 86	INDEX RANGE SCAN	JTF_RS_REP_MANAGERS_N1	8	38	2 (0)	197	00:00:00.01	34	0
* 87	INDEX UNIQUE SCAN	FND_TREE_VERSION_U1	19	1	1 (0)	19	00:00:00.01	36	0
* 88	TABLE ACCESS BY INDEX ROWID	FND_TREE_VERSION	19	1	2 (0)	19	00:00:00.01	3	0
89	NESTED LOOPS		90	1	203 (0)	1425	00:00:00.11	21000	66
90	NESTED LOOPS		90	59	85 (0)	2648	00:00:00.11	15565	66
91	NESTED LOOPS		90	10	26 (0)	180	00:00:00.05	13569	23
92	MERGE JOIN CARTESIAN		90	10	6 (0)	4320	00:00:00.02	516	23
93	TABLE ACCESS BY INDEX ROWID	MOO_REVN	90	1	4 (0)	90	00:00:00.02	306	23
* 94	INDEX RANGE SCAN	MOO_REVN_PSR1	90	1	3 (0)	90	00:00:00.01	216	7
95	BUFFER SORT		90	10	2 (0)	4320	00:00:00.01	210	0
* 96	INDEX RANGE SCAN	MOT_TERR_RESOURCES_N2	90	10	2 (0)	4320	00:00:00.01	210	0
* 97	TABLE ACCESS BY INDEX ROWID	MOT_TERRITORIES	4320	1	2 (0)	180	00:00:00.03	13053	0
* 98	INDEX UNIQUE SCAN	MOT_TERRITORIES_PK	4320	1	1 (0)	4320	00:00:00.01	8733	0
* 99	TABLE ACCESS BY INDEX ROWID	MOO_REVN_TERR	180	6	12 (0)	2648	00:00:00.05	1996	43
*100	INDEX RANGE SCAN	MOO_REVN_TERR_U1	180	18	3 (0)	2648	00:00:00.01	636	15
*101	INDEX RANGE SCAN	MOT_TERR_HIERARCHY_DN_V_N2	2648	1	2 (0)	1425	00:00:00.01	5435	0
*102	HASH JOIN		90	1	20 (5)	0	00:00:00.03	1603	0
103	NESTED LOOPS		90			1324	00:00:00.01	1393	0
104	NESTED LOOPS		90	6	16 (0)	1324	00:00:00.01	681	0
105	TABLE ACCESS BY INDEX ROWID	MOO_REVN	90	1	4 (0)	90	00:00:00.01	306	0
*106	INDEX RANGE SCAN	MOO_REVN_PSR1	90	1	3 (0)	90	00:00:00.01	216	0
*107	INDEX RANGE SCAN	MOO_REVN_TERR_U1	90	18	3 (0)	1324	00:00:00.01	375	0
*108	TABLE ACCESS BY INDEX ROWID	MOO_REVN_TERR	1324	6	12 (0)	1324	00:00:00.01	712	0
*109	INDEX RANGE SCAN	MOT_TERR_RESOURCES_N2	90	10	3 (0)	4320	00:00:00.01	210	0
110	NESTED LOOPS		90	1	306 (0)	2940	00:00:22.34	7671K	10
111	NESTED LOOPS		90	86	48 (0)	2519K	00:00:01.08	38739	0
112	NESTED LOOPS		90	10	28 (0)	180	00:00:00.02	14166	0
113	MERGE JOIN CARTESIAN		90	10	8 (0)	4320	00:00:00.01	1113	0
114	NESTED LOOPS		90	1	6 (0)	90	00:00:00.01	903	0
115	TABLE ACCESS BY INDEX ROWID	MOO_OPTY	90	1	3 (0)	90	00:00:00.01	306	0
*116	INDEX UNIQUE SCAN	MOO_OPTY_PK1	90	1	2 (0)	90	00:00:00.01	216	0
117	TABLE ACCESS BY INDEX ROWID	ZCA_SALES_ACCOUNTS	90	1	3 (0)	90	00:00:00.01	597	0
*118	INDEX RANGE SCAN	ZCA_SALES_ACCOUNTS_N4	90	1	2 (0)	90	00:00:00.01	270	0
119	BUFFER SORT		90	10	5 (0)	4320	00:00:00.01	210	0
*120	INDEX RANGE SCAN	MOT_TERR_RESOURCES_N2	90	10	2 (0)	4320	00:00:00.01	210	0
*121	TABLE ACCESS BY INDEX ROWID	MOT_TERRITORIES	4320	1	2 (0)	180	00:00:00.01	13053	0
*122	INDEX UNIQUE SCAN	MOT_TERRITORIES_PK	4320	1	1 (0)	4320	00:00:00.01	8733	0
*123	INDEX RANGE SCAN	MOT_TERR_HIERARCHY_DN_V_N1	180	9	2 (0)	2519K	00:00:00.68	24573	0
*124	INDEX RANGE SCAN	ZCA_S_ACCT_TERRITORIES_N3	2519K	1	3 (0)	2940	00:00:13.82	7632K	10
*125	HASH JOIN		90	1	16 (7)	0	00:00:00.04	1830	0
126	NESTED LOOPS		90			2871	00:00:00.02	1620	0
127	NESTED LOOPS		90	22	12 (0)	2871	00:00:00.01	1327	0
128	NESTED LOOPS		90	1	6 (0)	90	00:00:00.01	903	0
129	TABLE ACCESS BY INDEX ROWID	MOO_OPTY	90	1	3 (0)	90	00:00:00.01	306	0
*130	INDEX UNIQUE SCAN	MOO_OPTY_PK1	90	1	2 (0)	90	00:00:00.01	216	0
131	TABLE ACCESS BY INDEX ROWID	ZCA_SALES_ACCOUNTS	90	1	3 (0)	90	00:00:00.01	597	0
*132	INDEX RANGE SCAN	ZCA_SALES_ACCOUNTS_N4	90	1	2 (0)	90	00:00:00.01	270	0
*133	INDEX RANGE SCAN	ZCA_S_ACCT_TERRITORIES_N4	90	22	2 (0)	2871	00:00:00.01	424	0
134	TABLE ACCESS BY INDEX ROWID	ZCA_S_ACCT_TERRITORIES	2871	21	6 (0)	2871	00:00:00.01	293	0
*135	INDEX RANGE SCAN	MOT_TERR_RESOURCES_N2	90	10	3 (0)	4320	00:00:00.01	210	0
136	NESTED LOOPS		90			22506	00:00:00.22	29518	212
137	NESTED LOOPS		90	99	405 (0)	22506	00:00:00.20	29428	212
138	NESTED LOOPS		90	99	207 (0)	22506	00:00:00.16	29299	212
*139	TABLE ACCESS BY INDEX ROWID	MOO_OPTY_RESOURCES	90	23	23 (0)	2856	00:00:00.05	2630	15
*140	INDEX RANGE SCAN	MOO_OPTY_RESOURCES_N20	90	23	3 (0)	2856	00:00:00.01	271	1
*141	TABLE ACCESS BY INDEX ROWID	JTF_RS_REP_MANAGERS	2856	4	8 (0)	22506	00:00:00.22	26669	197
*142	INDEX RANGE SCAN	JTF_RS_REP_MANAGERS_N1	2856	38	2 (0)	161K	00:00:00.10	8006	45
*143	INDEX UNIQUE SCAN	FND_TREE_VERSION_U1	22506	1	1 (0)	22506	00:00:00.02	129	0
*144	TABLE ACCESS BY INDEX ROWID	FND_TREE_VERSION	22506	1	2 (0)	22506	00:00:00.01	90	0

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

   1 - filter(ROWNUM=1)
   2 - filter(("HCP"."STATUS"='A' AND "HCP"."CONTACT_LEVEL_TABLE"='HZ_CONTACT_POINTS' AND
              TRUNC(INTERNAL_FUNCTION("HCP"."PREFERENCE_START_DATE"))<=TRUNC(SYSDATE@!) AND TRUNC(INTERNAL_FUNCTION("HCP"."PREFERENCE_END_DATE"))>=TRUNC(SYSDATE@!)))
   3 - access("HCP"."CONTACT_LEVEL_TABLE_ID"=:B1)
  21 - filter(("OPPORTUNITYEO"."STATUS_CD"=:BINDDEFAULTSTATUSCD AND "OPPORTUNITYEO"."EFFECTIVE_DATE">=:BINDEFFECTIVEBEGINDATE AND
              "OPPORTUNITYEO"."EFFECTIVE_DATE"<=:BINDEFFECTIVEENDDATE))
  22 - access("OPPORTUNITYEO"."SUM_REVN_ID"="REVENUEEO"."REVN_ID")
  24 - access("OPPORTUNITYEO"."PR_CON_PARTY_ID"="PRIMARYCONTACTPARTYPEO"."PARTY_ID")
  26 - access("OPPORTUNITYEO"."OWNER_RESOURCE_ID"="OWNERRESOURCEPARTYPEO"."PARTY_ID")
  28 - access("OPPORTUNITYEO"."PR_CMPT_PARTY_ID"="PRIMARYCOMPETITORPARTYPEO"."PARTY_ID")
  30 - access("OPPORTUNITYEO"."OPTY_PARTY_SITE_ID"="PARTYSITEPEO"."PARTY_SITE_ID")
  32 - access("PARTYSITEPEO"."LOCATION_ID"="LOCATIONPEO"."LOCATION_ID")
  34 - access("OPPORTUNITYEO"."CUST_PARTY_ID"="PARTYPEO"."PARTY_ID")
  36 - access("OPPORTUNITYEO"."CUST_PARTY_ID"="SALESACCOUNTPEO"."PARTY_ID")
  37 - filter(("CODEASSIGNMENTPEO"."STATUS"='A' AND "CODEASSIGNMENTPEO"."PRIMARY_FLAG"='Y'))
  38 - access("OPPORTUNITYEO"."CUST_PARTY_ID"="CODEASSIGNMENTPEO"."OWNER_TABLE_ID" AND "CODEASSIGNMENTPEO"."OWNER_TABLE_NAME"='HZ_PARTIES' AND
              "CODEASSIGNMENTPEO"."CLASS_CATEGORY"='ORGANIZATION_TYPE')
       filter(("CODEASSIGNMENTPEO"."CLASS_CATEGORY"='ORGANIZATION_TYPE' AND "CODEASSIGNMENTPEO"."OWNER_TABLE_NAME"='HZ_PARTIES'))
  39 - filter(("RELATIONSHIPPEO"."STATUS"=:VC_TEMP_4 AND "RELATIONSHIPPEO"."END_DATE">=:BINDSYSDATE AND "RELATIONSHIPPEO"."START_DATE"<=:BINDSYSDATE AND
              "OPPORTUNITYEO"."PR_CON_PARTY_ID"="RELATIONSHIPPEO"."SUBJECT_ID"))
  40 - access("OPPORTUNITYEO"."PR_CON_RELATIONSHIP_ID"="RELATIONSHIPPEO"."RELATIONSHIP_ID")
  41 - filter(("PHONEPEO"."END_DATE">=:BINDSYSDATE AND "PHONEPEO"."START_DATE"<=:BINDSYSDATE))
  42 - access("RELATIONSHIPPEO"."RELATIONSHIP_ID"="PHONEPEO"."RELATIONSHIP_ID" AND "RELATIONSHIPPEO"."SUBJECT_ID"="PHONEPEO"."OWNER_TABLE_ID" AND
              "PHONEPEO"."CONTACT_POINT_TYPE"='PHONE' AND "PHONEPEO"."PRIMARY_FLAG"=:VC_TEMP_5 AND "RELATIONSHIPPEO"."SUBJECT_TABLE_NAME"="PHONEPEO"."OWNER_TABLE_NAME"
              AND "PHONEPEO"."STATUS"='A')
  44 - filter(:BINDDEFAULTSTATUSCD="OPPORTUNITYEO"."STATUS_CD")
  46 - filter("B"."ENABLED_FLAG"='Y')
  47 - access("B"."LOOKUP_TYPE"='OPTY_STATUS' AND "B"."LOOKUP_CODE"="OPPORTUNITYEO"."STATUS_CD")
       filter("B"."LOOKUP_CODE"=:BINDDEFAULTSTATUSCD)
  48 - access("B"."LOOKUP_VALUES_ID"="T"."LOOKUP_VALUES_ID" AND "T"."LANGUAGE"=USERENV('LANG'))
  52 - access("T"."STG_ID"="OPPORTUNITYEO"."CURR_STG_ID" AND "T"."LANGUAGE"=USERENV('LANG'))
  54 - access("B"."STG_ID"="OPPORTUNITYEO"."CURR_STG_ID")
       filter("B"."STG_ID"="T"."STG_ID")
  58 - access("T"."SALES_METHOD_ID"="OPPORTUNITYEO"."SALES_METHOD_ID" AND "T"."LANGUAGE"=USERENV('LANG'))
  60 - access("B"."SALES_METHOD_ID"="OPPORTUNITYEO"."SALES_METHOD_ID")
       filter("B"."SALES_METHOD_ID"="T"."SALES_METHOD_ID")
  62 - filter((TO_DATE(:BINDEFFECTIVEBEGINDATE)<=TO_DATE(:BINDEFFECTIVEENDDATE) AND TO_DATE(:BINDEFFECTIVEBEGINDATE)<=TO_DATE(:BINDEFFECTIVEENDDATE)))
  65 - filter(("OPPORTUNITYRESOURCEEONO1"."OPTY_STATUS_CD"=:BINDDEFAULTSTATUSCD AND
              "OPPORTUNITYRESOURCEEONO1"."OPTY_EFFECTIVE_DATE"<=:BINDEFFECTIVEENDDATE AND "OPPORTUNITYRESOURCEEONO1"."OPTY_EFFECTIVE_DATE">=:BINDEFFECTIVEBEGINDATE AND
              "OPPORTUNITYRESOURCEEONO1"."RESOURCE_ID"<>TO_NUMBER(:BINDLOGGEDINUSERRESOURCEID)))
  66 - access("OPPORTUNITYRESOURCEEONO1"."OPTY_ID"="OPPORTUNITYEO"."OPTY_ID")
  67 - access("REPORTINGMANAGERPEONO1"."PARENT_RESOURCE_ID"=TO_NUMBER(:BINDLOGGEDINUSERRESOURCEID) AND
              "OPPORTUNITYRESOURCEEONO1"."RESOURCE_ID"="REPORTINGMANAGERPEONO1"."RESOURCE_ID")
       filter("REPORTINGMANAGERPEONO1"."RESOURCE_ID"<>TO_NUMBER(:BINDLOGGEDINUSERRESOURCEID))
  68 - filter(("OPPORTUNITYRESOURCEEONO1"."RESOURCE_ORG_ID"="REPORTINGMANAGERPEONO1"."GROUP_ID" AND
              "OPPORTUNITYRESOURCEEONO1"."ORG_TREE_CODE"="REPORTINGMANAGERPEONO1"."TREE_CODE" AND
              "OPPORTUNITYRESOURCEEONO1"."ORG_TREE_STRUCTURE_CODE"="REPORTINGMANAGERPEONO1"."TREE_STRUCTURE_CODE"))
  72 - filter(("MYOPRES"."ACCESS_LEVEL_CODE"='100' OR "MYOPRES"."ACCESS_LEVEL_CODE"='200' OR "MYOPRES"."ACCESS_LEVEL_CODE"='300'))
  73 - access("MYOPRES"."OPTY_ID"="OPPORTUNITYEO"."OPTY_ID" AND "MYOPRES"."RESOURCE_ID"=:BindResourceID)
  80 - access("OPTY"."OPTY_ID"="OPPORTUNITYEO"."OPTY_ID")
  82 - access("OPTY"."CUST_PARTY_ID"="SALESACCNT"."PARTY_ID")
  83 - filter(("ACNTRES"."ORG_TREE_STRUCTURE_CODE" IS NOT NULL AND "ACNTRES"."ORG_TREE_CODE" IS NOT NULL AND "ACNTRES"."RESOURCE_ORG_ID" IS NOT NULL))
  84 - access("SALESACCNT"."SALES_ACCOUNT_ID"="ACNTRES"."SALES_ACCOUNT_ID")
  85 - filter(("MGRCHAIN"."PARENT_RESOURCE_ID"=:BindResourceID AND "ACNTRES"."ORG_TREE_STRUCTURE_CODE"="MGRCHAIN"."TREE_STRUCTURE_CODE" AND
              "ACNTRES"."ORG_TREE_CODE"="MGRCHAIN"."TREE_CODE"))
  86 - access("ACNTRES"."RESOURCE_ORG_ID"="MGRCHAIN"."GROUP_ID")
  87 - access("TREE_STRUCTURE_CODE"="MGRCHAIN"."TREE_STRUCTURE_CODE" AND "TREE_CODE"="MGRCHAIN"."TREE_CODE" AND
              "TREE_VERSION_ID"="MGRCHAIN"."TREE_VERSION_ID" AND "ENTERPRISE_ID"=DECODE(SYS_CONTEXT('FND_VPD_CTX','ENTERPRISE_ID'),NULL,1,'0',1,TO_NUMBER(SYS_CONTEXT('FN
             D_VPD_CTX','ENTERPRISE_ID'))))
  88 - filter(("EFFECTIVE_START_DATE"<=SYSDATE@! AND "EFFECTIVE_END_DATE">=SYSDATE@!))
  94 - access("REVN"."OPTY_ID"="OPPORTUNITYEO"."OPTY_ID" AND "REVN"."PRIMARY_FLAG"='Y')
       filter("REVN"."PRIMARY_FLAG"='Y')
  96 - access("TERR_RES"."RESOURCE_ID"=:BindResourceID)
  97 - filter(("TERR"."LATEST_VERSION_FLAG"='Y' AND "TERR"."STATUS_CODE"='FINALIZED'))
  98 - access("TERR_RES"."TERRITORY_VERSION_ID"="TERR"."TERRITORY_VERSION_ID")
  99 - filter("REVNTERR"."SUM_REVN_FLAG"='Y')
 100 - access("REVNTERR"."REVN_ID"="REVN"."REVN_ID")
 101 - access("TERRDN"."DESCENDANT_TERRITORY_ID"="REVNTERR"."TERRITORY_ID" AND "TERR"."TERRITORY_ID"="TERRDN"."TERRITORY_ID")
       filter("TERRDN"."DESCENDANT_TERRITORY_ID"<>"TERRDN"."TERRITORY_ID")
 102 - access("MOT_TERR_RESOURCES"."TERRITORY_VERSION_ID"="MOO_REVN_TERR"."TERRITORY_VERSION_ID")
 106 - access("MOO_REVN"."OPTY_ID"="OPPORTUNITYEO"."OPTY_ID" AND "MOO_REVN"."PRIMARY_FLAG"='Y')
       filter("MOO_REVN"."PRIMARY_FLAG"='Y')
 107 - access("MOO_REVN_TERR"."REVN_ID"="MOO_REVN"."REVN_ID")
 108 - filter("MOO_REVN_TERR"."SUM_REVN_FLAG"='Y')
 109 - access("MOT_TERR_RESOURCES"."RESOURCE_ID"=:BindResourceID)
 116 - access("MOO_OPTY"."OPTY_ID"="OPPORTUNITYEO"."OPTY_ID")
 118 - access("ZCA_SALES_ACCOUNTS"."PARTY_ID"="MOO_OPTY"."CUST_PARTY_ID")
 120 - access("MOT_TERR_RESOURCES"."RESOURCE_ID"=:BindResourceID)
 121 - filter(("MOT_TERRITORIES"."LATEST_VERSION_FLAG"='Y' AND "MOT_TERRITORIES"."STATUS_CODE"='FINALIZED'))
 122 - access("MOT_TERR_RESOURCES"."TERRITORY_VERSION_ID"="MOT_TERRITORIES"."TERRITORY_VERSION_ID")
 123 - access("MOT_TERRITORIES"."TERRITORY_ID"="MOT_TERR_HIERARCHY_DN"."TERRITORY_ID")
       filter("MOT_TERR_HIERARCHY_DN"."DESCENDANT_TERRITORY_ID"<>"MOT_TERR_HIERARCHY_DN"."TERRITORY_ID")
 124 - access("MOT_TERR_HIERARCHY_DN"."DESCENDANT_TERRITORY_ID"="ZCA_S_ACCT_TERRITORIES"."TERRITORY_ID" AND
              "ZCA_S_ACCT_TERRITORIES"."SALES_ACCOUNT_ID"="ZCA_SALES_ACCOUNTS"."SALES_ACCOUNT_ID")
 125 - access("MOT_TERR_RESOURCES"."TERRITORY_VERSION_ID"="ZCA_S_ACCT_TERRITORIES"."TERRITORY_VERSION_ID")
 130 - access("MOO_OPTY"."OPTY_ID"="OPPORTUNITYEO"."OPTY_ID")
 132 - access("ZCA_SALES_ACCOUNTS"."PARTY_ID"="MOO_OPTY"."CUST_PARTY_ID")
 133 - access("ZCA_S_ACCT_TERRITORIES"."SALES_ACCOUNT_ID"="ZCA_SALES_ACCOUNTS"."SALES_ACCOUNT_ID")
 135 - access("MOT_TERR_RESOURCES"."RESOURCE_ID"=:BindResourceID)
 139 - filter(("ORES"."ACCESS_LEVEL_CODE"='100' OR "ORES"."ACCESS_LEVEL_CODE"='200' OR "ORES"."ACCESS_LEVEL_CODE"='300'))
 140 - access("ORES"."OPTY_ID"="OPPORTUNITYEO"."OPTY_ID")
 141 - filter(("MGRCHAIN"."PARENT_RESOURCE_ID"=:BindResourceID AND "ORES"."ORG_TREE_STRUCTURE_CODE"="MGRCHAIN"."TREE_STRUCTURE_CODE" AND
              "ORES"."ORG_TREE_CODE"="MGRCHAIN"."TREE_CODE"))
 142 - access("ORES"."RESOURCE_ORG_ID"="MGRCHAIN"."GROUP_ID")
 143 - access("TREE_STRUCTURE_CODE"="MGRCHAIN"."TREE_STRUCTURE_CODE" AND "TREE_CODE"="MGRCHAIN"."TREE_CODE" AND
              "TREE_VERSION_ID"="MGRCHAIN"."TREE_VERSION_ID" AND "ENTERPRISE_ID"=DECODE(SYS_CONTEXT('FND_VPD_CTX','ENTERPRISE_ID'),NULL,1,'0',1,TO_NUMBER(SYS_CONTEXT('FN
             D_VPD_CTX','ENTERPRISE_ID'))))
```

Figure 8. Execution Plan for Rewritten SQL

3.2. Confirmation of Performance Improvements Using Above Approach

The approach described and detailed above was tried on many different variations of SQLs for different roles, responsibilities and users. The performance metrics and execution plans for the rewritten SQL were always better than the original SQLs.

A summary of the benchmark results for different scenarios is provided below in Figures 9 and 10.

		Original SQL	Rewritten SQL
		Query Statistics	Query Statistics
	Resource-id	JDBCBench results for 25 rows fetch (EFFECTIVE_DATE range 1 year - 06/01/2012 to 05/31/2013)	JDBCBench results for 25 rows fetch (EFFECTIVE_DATE range 1 year - 06/01/2012 to 05/31/2013)
SALES_MGR - My Salesteam			
	User1	Iter Prepare(s) Execute(s) Fetch(s) FetchedRows Total(s) 1 0.039 21.816 17.704 25 39.559 2 0.000 1.991 4.226 25 6.218 3 0.000 2.003 4.208 25 6.211 **Buffer_Gets(per exec): 2945477 **Shared Mem(MB) : 0.506 **Plan Cost: 18109	Iter Prepare(s) Execute(s) Fetch(s) FetchedRows Total(s) 1 0.025 4.402 0.319 25 4.747 2 0.000 0.149 0.326 25 0.475 3 0.000 0.142 0.251 25 0.394 **Buffer_Gets(per exec): 1410 **Shared Mem(MB) : 0.561 **Plan Cost: 1042
	User2	Iter Prepare(s) Execute(s) Fetch(s) FetchedRows Total(s) 1 0.023 4.974 0.001 1 4.999 2 0.000 0.439 0.001 1 0.440 3 0.000 0.445 0.000 1 0.445 **Buffer_Gets(per exec): 171845 **Shared Mem(MB) : 0.506 **Plan Cost: 2727	Iter Prepare(s) Execute(s) Fetch(s) FetchedRows Total(s) 1 0.025 4.203 0.001 1 4.229 2 0.000 0.118 0.001 1 0.119 3 0.000 0.113 0.001 1 0.114 **Buffer_Gets(per exec): 61 **Shared Mem(MB) : 0.565 **Plan Cost: 1022
	User3	Iter Prepare(s) Execute(s) Fetch(s) FetchedRows Total(s) 1 0.040 4.220 0.886 25 5.147 2 0.000 0.130 0.285 25 0.415 3 0.000 0.130 0.281 25 0.411 **Buffer_Gets(per exec): 19642.3 **Shared Mem(MB) : 0.510 **Plan Cost: 17400	Iter Prepare(s) Execute(s) Fetch(s) FetchedRows Total(s) 1 0.043 4.408 0.296 25 4.748 2 0.000 0.118 0.252 25 0.370 3 0.000 0.115 0.236 25 0.351 **Buffer_Gets(per exec): 1343 **Shared Mem(MB) : 0.561 **Plan Cost: 657

Figure 9. Benchmark Results for Rewritten SQL for “MySalesTeam” Scenario

		Original SQL	Rewritten SQL
		Query Statistics	Query Statistics
	Resource-id	JDBCBench results for 25 rows fetch (EFFECTIVE_DATE range 1 year - 06/01/2012 to 05/31/2013)	JDBCBench results for 25 rows fetch (EFFECTIVE_DATE range 1 year - 06/01/2012 to 05/31/2013)
SALES_MGR - My_Subordinates			
	User1	Iter Prepare(s) Execute(s) Fetch(s) FetchedRows Total(s) 1 0.035 7.009 4.663 25 11.707 2 0.000 1.825 3.632 25 5.457 3 0.000 1.815 3.643 25 5.457 **Buffer_Gets(per exec): 2600794.6 **Shared Mem(MB) : 0.534 **Plan Cost: 141	Iter Prepare(s) Execute(s) Fetch(s) FetchedRows Total(s) 1 0.037 9.520 0.261 25 9.818 2 0.000 0.119 0.252 25 0.372 3 0.000 0.119 0.248 25 0.367 **Buffer_Gets(per exec): 8469 **Shared Mem(MB) : 0.624 **Plan Cost: 4148
	User2	Iter Prepare(s) Execute(s) Fetch(s) FetchedRows Total(s) 1 0.032 6.492 3.832 25 10.357 2 0.000 1.810 3.602 25 5.411 3 0.000 1.800 3.629 25 5.429 **Buffer_Gets(per exec): 2576102.6 **Shared Mem(MB) : 0.534 **Plan Cost: 145	Iter Prepare(s) Execute(s) Fetch(s) FetchedRows Total(s) 1 0.024 9.470 0.422 25 9.916 2 0.000 0.120 0.245 25 0.365 3 0.000 0.119 0.243 25 0.363 **Buffer_Gets(per exec): 9697 **Shared Mem(MB) : 0.624
	User3	Iter Prepare(s) Execute(s) Fetch(s) FetchedRows Total(s) 1 0.025 14.058 18.060 25 32.143 2 0.000 3.995 5.581 25 9.577 3 0.000 3.923 5.488 25 9.411 **Buffer_Gets(per exec): 2519724.6 **Shared Mem(MB) : 0.538 **Plan Cost: 13457	Iter Prepare(s) Execute(s) Fetch(s) FetchedRows Total(s) 1 0.038 7.047 0.366 25 7.450 2 0.000 0.673 0.354 25 1.028 3 0.000 0.678 0.348 25 1.027 **Buffer_Gets(per exec): 202920 **Shared Mem(MB) : 0.636 **Plan Cost: 1956

Figure 10. Benchmark Results for Rewritten SQL for “MySubordinates” Scenario

4. CONCLUSION

Most commercial business applications today like Sales Automation are built around technology layers that generate physical SQLs at run time. Access to data in many such applications is restricted on a need to know basis usually by implementing security and visibility through a set of roles and responsibilities that each are defined by way of sub-queries that get appended to the main SQL at run time. This method of adding sub-queries cumulatively through a security framework results in complex SQLs, especially for users who are granted many different roles and responsibilities. The main query is usually referred to as the outer query and the appended security predicates' sub-queries are termed as the inner query. When such SQLs arrive at the database, the CBO evaluates them for possible access paths and join optimizations to decide on an execution plan based on available statistics. Very often, the CBO is posed with the dilemma of whether to drive from the outer query or drive from the inner query. For some users, the outer query with its filter predicates can be greatly restrictive while for other users, the inner security predicates' sub-queries can be more restrictive. This if sometimes referred to as the “Tiny-Huge, Huge-Tiny” problem that many application and database designers struggle to manage. Many times, this problem leads to poor choices by the CBO, resulting in sub-optimal execution plans leading to poor query response times and consequently causing performance and scalability issues for the application as well as the database.

This paper has presented the analysis of such SQLs and CBO execution plans from the Opportunity Management module of a CRM application that performed poorly due to the aforesaid “Tiny-Huge, Huge-Tiny” problem. Based on the analysis, the paper then presented a suggested solution incorporating some de-normalized columns and rewrite of the security predicates’ sub-queries that result in vastly improved performance and scalability of such queries and consequently of the application.

This approach can be suitably amended and applied to different applications based on the specifics of any such similar SQL performance issues.

REFERENCES

[1] “Oracle® Database Performance Tuning Guide 11g Release 2 (11.2) E16638-07” [Online]. Available: http://docs.oracle.com/cd/E11882_01/server.112/e16638.pdf. [Accessed Dec-Jan 2012-2013]

[2] “Oracle® Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework 11g Release 1 (11.1.1.5.0)” [Online] Available: http://docs.oracle.com/cd/E21764_01/web.1111/b31974/bcintro.htm[Accessed Jan 2013]

[3] Abraham Silberschatz, Hank Korth and S. Sudarshan. Database system concepts, 5th Edition. McGraw-Hill, 2006

[4] Ramez Elmasri and Shamkant B. Navathe. Fundamentals of database systems, Fifth edition. Pearson Education ,2009.

[5] Won Kim,David S. Reiner,Don s. Batory. Query processing in database systems. Springer Verlag, Berlin Heidelberg New York Tokyo.

[6] Aper PMG, Hevner AR,Yao SB, “Optimization algorithms for distributed queries”, IEEE Transactions on software Engineering, SE-9.1, January 1983, 57-68.

[7] Chokri Ben Necib and Johann-Christoph Freytag, “Using Ontologies for Database Query Reformulation, Advances in Databases and Information Systems” (ADBIS), Hungary, 2004.

[8] William miles , “Optimizing SQL Query Processing” , InterviewInfo.net 2005.

[9] Williams, and Martha E., “Query Expansion, Annual Review of Information Systems and Technology" (ARIST), http://faculty.washington.edu/efthimis/pubs/Pubs/qe-arist/QE-arist.html, pp 121-187, 1996.

[10] Steve Renals, “Query Expansion”,http://homepages.inf.ed.ac.uk/srenals/ pubs/1999/esca99-thisl/node6.html, 1999.

[11] F.A. Grootjen and Th. P. van der Weide, “Conceptual Query Expansion”, Journal of Data Knowledge and Engineering, pp. 174-193, 2004.

[12] Abdelkrim Bouramoul, Mohamed-Khireddine Kholladi and Bich-Lien Doan, “Using Context to Improve the Evaluation of Information Retrieval Systems”,International Journal of Database Management Systems (IJDMS), Vol.3, No.2, May 2011.

[13] Query Result Size Estimation Techniques in Database Systems – Banchong Harangsri - 1998

[14] The Effect of Cost Distributions on Evolutionary Optimization Algorithms – F. Waas, C. Galindo-Legaria, Florian Waas - 2000

[15] Selinger, P.G., Astrahan, M.M., Chamberlin, D.D., Lorie, R.A., Price, T.G.: Access path selection in a relational database management system. In: Proceedings of the 1979 ACM SIGMOD international conference on Management of data, pp. 23–34. ACM Press, New York (1979)

[16] Exploiting Functional Dependence in Query Optimization – Glenn Norman Paulley - 2000

[17] Optimizing Join Orders – Michael Steinbrunn, Guido Moerkotte, Alfons Kemper - 1993

[18] Query Optimization – Yannis E. Ioannidis - 1996

[19] Exploiting Cost Distributions for Query Optimization – F. Waas, J. Pellenkoft, Florian Waas, Arjan Pellenkoft - 1998

[20] Steinbrunn, M., Moerkotte, G., Kemper, A.: Heuristic and randomized optimization for the join ordering problem. VLDB Journal: Very Large Data Bases 6(3), 191–208 (1997)

[21] Swami, A.: Optimization of large join queries: combining heuristics and combinatorial techniques. In: SIGMOD 1989: Proceedings of the 1989 ACM SIGMOD international conference on Management of data, pp. 367–376. ACM Press, New York (1989)

Authors

1. Mr. Arjun Sirohi is currently a Consulting Member of Technical Staff in the PSR Engineering Division (Performance Scalability and Reliability) at Oracle USA located in Bellevue, WA, USA. He holds a MS CS degree from City University of Seattle, from where he graduated with President's Honours with 3.93 GPA.

2. He has 21 years of work experience in software design, development and implementation in various roles as Developer, Database Consultant, Architecture Specialist and Senior Systems Manager. He has extensive experience in application, SQL, database and system performance tuning, optimizations and testing. Apart from databases, he has deep expertise in data warehousing and business intelligence applications' performance and scalability.