Measuring of Oracle Optimizer Performance for Object-Relational Database Entities

Esam Mahmoud Mohammed  
Foundation of Technical Education  
Mosul Technical Institute, Iraq

Alaa F.S. AL_Mukhtar  
Ministry of Electricity  
Northern Electricity, Iraq

Received on: 06/11/2007  
Accepted on: 04/03/2008

ABSTRACT

SQL Optimization is the process of choosing the most efficient way to execute SQL such as Data manipulation language (Select, Insert, Update, Delete). Relational model represents the object through the use of relations; on the other hand the main aim of ORDBMS model is to represent the special characteristics of OOP. The first model deals with the Relation & Correlated sub queries, while the second one uses collection types (Object type, Varrays, Nested table). Explain plane is used to examine exactly how oracle execute SQL statements “oracle SQL analyze” provides a facility for easily generate any explain plan. DML language (SQL) as tools for any comparison can be achieved by the statistical performance parameter (Elapsed time, CPU time, Logical block read, Physical block read ...). Statistical analysis using ANOVA TABLE is used to judge between comparisons. We found that nested table was the most effective type of data, the object type was the least effective (consume lowest time of execution). We also found that increasing data size will increase any of performance data parameter until we reach size of 40% the time begins to decrease. It is found that SELECT, UPDATE and DELETE have the most influence on the performance parameter, respectively.

Keywords: Oracle, Database.

قياس أداء أوراكل الأمثل لقواعد البيانات الكياتية العلائقية

عاصم محمود محمد  
المعهد التقني/هيئة التعليم التقني/الموصل
تاريخ القبول: 04/03/2008
تاريخ الاستلام: 06/11/2007

إن الأمثلية هي عملية اختيار أكفا الطريقة في تنفيذ ايعازات لغة معالجة البيانات SQL. يتم تمثيل الكائنات في النموذج العلائقي من خلال استعمال العلاقات ، بينما الهدف الرئيسي من استخدام نموذج قواعد البيانات الكياتية العلائقية هو لإظهار الصفات الخاصة ORDBMS.
1. Introduction

Much of what is needed to build large scale distributed information systems is already in existing relational systems and their capabilities continue to grow at an exciting rate. Rather than starting a new and risk never catching the moving relational “train”. On board and extend the technology in a pragmatic way to realize the important benefits of object technology. The following sections will look at how the familiar relational model can be extended in an evolutionary way so that both technologies can work together in the same server. Users of the system can build existing skills and techniques to store, access and update object data together with relational data in the same system using the same schema with new and current tools [2]. Object functionality is one of the major features of Oracle8; however, relatively little has been written about the performance impact of its use [4].

Relational model represents the object by using relations. While the main aim of ORDBMS model is to represent the special characteristics of OOP(Object Oriented Programming) of Database without looking to the Relational characteristics that are used with the present application ORDBMS an more richer than RDBMS model, and this richness leads to consistency and good structure in designing database of complex case like data collection [6].

A comparison of several queries against paired relational tables and nested tables found that data access was almost equal in terms of
performance. The only significant problem was a functional retrieval from the detail table for a given master row. In summary, leveraging the full power of some of these features will only be realized by moving wholly to an OO approach. As for object collections, nested tables will not bring major benefits unless one is using object tables [4].

A case study using typical master-detail type data from a live system was undertaken to investigate two of the most useful features object type and object collections. Nested Table object collections may be useful for implementing master-detail type relationships. A limiting factor in terms of functionality is that access to the detail table is solely through the master table, so look-ups from the detail table are not allowed.

2. Procedure and Methods

The Oracle 9i database was installed with 256MB of RAM running MS Windows XP. In order to study the new object-related features in oracle 9i, case studies were performed against data from a production system. The chosen data comprises a suitable number of relational customer & address records—forming a typical master-detail relationship. Several queries and DML statements were used as shown below.

Data of 600000 records where created for table customer and its related table of address using program. Also data types of the same size with many levels of runs where done to achieve statistical requirements using different kinds of DML statements, with recording performance parameters with different sizes of data.

SAS computer package were used for statistical Analysis of the performance statistic data cost (millisecond), Elapsed time this data item represents the total amount of time spent parsing SQL statements (millisecond). Besides amount of time spent by CPU is the number of disc accesses affect the query processing efficiency. Logical blocks read is to access and process those data through memory (No.). Physical blocks read is transferring data from disk to memory (No.), UGA (Byte), or user global area, is allocated in the PGA (Program global area in Byte) for each session connected to oracle in a dedicated server environment which is measured in three replications by using statistical page of SQL ANALYZER of ORACLE 9i. ANOVA statistical analysis where used to give Duncan’s multiple range test which used to distinguish between the means of data by giving them letters [11].
2.1 Relation table

The foundation of Relational Database Management Systems (RDBMS) is the relational model first described by E.F. Codd in 1970. The success of this technology is largely due to the elegant simplicity of the relational model. Which nevertheless is capable of supporting large scale systems, which are robust and versatile [3].

Creation of simple Relational table as shown below in figure(1):

```sql
CREATE TABLE customer2
(customer_id varchar2(5),
Last_name varchar2(20),
first_name varchar2(14),
street varchar2(120),
city varchar2(30),
state char(2),
zip number(5));
```

```sql
INSERT INTO customer2
(customer_id, last_name, first_name, street, city, state, zip);
VALUES( '22222', 'Smith', 'Jonny', 'S. Main', 'Moscow', 'ID', 43);
```

2.2 Correlated sub queries

A correlated subquery is one that the subquery refers to values in the parent query. A correlated subquery can return the same result as a join, but can be used where a join cannot, such as in an update, insert and delete statement. In a correlated subquery, the subquery executes repeatedly, once for each value of a candidate row selected by the main query, this is why a correlated update can take more processing time[12].

Creation of one to many Relational table as shown below in figure(2):

```sql
CREATE TABLE new_agent
(customer_id varchar2(5),
last_name varchar2(20),
first_name varchar2(14));
```
CREATE TABLE addressr
(customer_id          varchar2(5),
street                    varchar2(120),
city                       varchar2(30),
state                      char(2),
zip                        number(5));

INSERT INTO new_agentr
(   customer_id    ,
    last_name    ,
    first_name   );
VALUES (    '11111'            ,      'Smith'       ,       'wood'      );

INSERT INTO addressr
( customer_id  ,     street            ,  city        ,       state  ,      zip       );
VALUES (  '11111'
,  '1190 S. Bend'  ,   'Pullman'  ,  'WA'   ,   '98119'  );

2.3 Object Types
Object types are used to extend the modeling capabilities provided
by the native data types. Object types can be used to make better models of
complex entities in the real world by binding data attributes to semantic
behaviors.[5 ]

Abstract Data Types (ADTs) from the SQL3 standards work will be
introduced as the basic representation mechanism for objects. Our ADTs
have the following properties:

One or more attributes. Zero or more methods (methods can be
expressed in PL/SQL or using 3GL call outs). Attribute type can be: scalar (number, char, etc.), an ADT reference or a collection (nested table or array) [9]. ADTs can be used as a column type or a table type. They can be used with relational operators. Constructors can be used to create instances of an ADT and definition would be done with CREATE TYPE. [8]

CREATE TYPE.
Creation of Object type is a simple 2-step process.
First, create the Object Type based on the columns of the relational table;

CREATE TYPE address_typ as OBJECT
( street varchar2(120),
city varchar2(30),
state char(2),
zip number(5));

Second, create the table based on the Type and selecting rows from the Relational Table;
CREATE TABLE customer1
(customer_id varchar2(5),
last_name varchar2(20),
first_name varchar2(14),
address address_typ);

INSERT INTO customer1
( customer_id ,
last_name ,
"ADDR"
VALUES ( '11111' , 'Smith' , ADDRESS_TYP
( '111 S. Main', 'Moscow', 'ID', 83843));

2.4 Collection Types
Collections are SQL data types that contain multiple elements. Each element or value for a collection is the same data type. In Oracle9i there are two collection types VARRAYs and nested tables. A VARRAY contains a variable number of ordered elements. VARRAY data types can be used as a column of a table or as an attribute of an object type. Also, the element type of a VARRAY may be either a native data type such as NUMBER or an object type. Using Oracle9i SQL, a named table type can be created. These can be used as nested tables to provide the semantics of an unordered collection. As with VARRAY, a nested table type can be used as a column of a table or as an attribute of an object type [2] [10].
2.4.1 Collection Types – Varrays

A VARRAY is an ordered collection of data. The number of data elements in the collection is pre-defined by the user. A VARRAY may be defined as a column in a database table, in which case the VARRAY data are stored in line with the ‘main table’. Of particular significance, VARRAY data held in this manner cannot be indexed. Data within the VARRAY are manipulated as a whole [1] [4].

2.4.1.1 Creating Varrays table collections

Creating a Varray collection included in a database table is achieved in 2 steps. First, create an object type defining the ‘layout’ of the varray. (In this case it is the same as the child relational table except that the Foreign Key column - Order_ID has been removed.)

CREATE TYPE address_t AS VARRAY(10) OF VARCHAR2(30);

Second, create the database table, with the integrated varray as a field in the table. At this stage, the ‘varray’ collection could be manipulated locally within PL/SQL applications. As shown in figure(3).

```
cREATE TYPE address_t AS VARRAY(10) OF VARCHAR2(30);

create table new_agent5(
    agent_id        varchar2(5 ),
    last_name      varchar2(25),
    first_name     varchar2(20),
    actor              address_t );

INSER INTO new_agent5
( agent_id , last_name , first_name , actor )
values('11111' , 'James' , 'Hartman' , address_t
('111 S. Main', 'Moscow 222',));
```
2.4.2 Collection Types - Nested Tables

Nested tables are data collections of unbounded size, allowing for limitless growth. When a table column is defined as a nested table, then logically we have an inner table resident within an outer table. Physically the inner table is a separate out-of-line table having the same general characteristics of relational tables. It is stored in the same table space as the outer table, taking by default the storage parameters of the table space. Data are not ordered in any particular sequence, and indexes may be applied[4]. Relationships between the inner and outer tables are automatically maintained within the database by using pointers. Given the unlimited size of the collection, there is more potential forth using Nested Tables, both inside the database and locally in PL/SQL [13]. Also, in the database there is the advantage of indexing the data and holding the data out of line. Nested tables may be useful for implementing master-detail type relationships [2].

2.4.2.1 Creating Nested table collections

Creating a Nested table collection included in a database table is achieved in 3 steps. First, create an object type defining the ‘layout’ of the nested table. In this case it is the same as the child relational table except that the Foreign Key column - Order_ID has been removed.

```
CREATE TYPE address AS OBJECT
(address_type char(1),
street varchar2(20),
city varchar2(20),
state char(2),
zip varchar2(9));
```

Second, create another type defined as a table of the object type defined in the first statement.

```
CREATE TYPE address_table AS TABLE OF address;
```
At this stage, the ‘nested table’ collection could be manipulated locally within PL/SQL applications. The final step is to create the database table, the outer table, with the integrated nested table, and the inner table as shown in figure(4).

CREATE TABLE new_agent(
    agent_id varchar2(5) not null,
    first_name varchar2(20),
    last_name varchar2(25),
    agent_address address_tab
)

NESTED TABLE agent_address STORE AS address_tab

Retrieval and manipulation of data in the inner table is always achieved through the outer table. The following SQL statements, termed ‘flattened sub-queries’, show how to make use of the new SQL expression, to provide a link between the outer and inner tables;

```
insert into new_agent
    (agent_id, first_name, last_name, address_table)
values ('11111', 'James', 'Hartman', address_table
    (address('H', '1190 S. Bend', 'Pullman', 'WA', '98119'),
    address('W', '500 Main Street N.', 'Moscow', 'ID', '83843')));
```

Here, agent_address is the column name and address_tab is the name of the physical table holding the nested table collection. The inner table is resident in the same table space as the outer table, and has the storage characteristics of the table space. Note that it is also possible to add a nested table column to an existing table similar to adding a new column to a relational table.

At this point, any queries against the inner table will access the data using a full table scan. So, despite the tight integration of the inner table with the outer table, it is still a table and so requires definition of proper access Nested Table Outer Table Inner Table paths [1] [4].

3. Analyzer

Oracle SQL analyzer provides the information about the database structure and modifies some initialization parameters to test the SQL statements against condition and database environment. As a SQL tuner, you need to be able to gather and analyze environmental data and performance statistics to help identify problem areas [7].

The following sections describe the information you can gather and the
methods Oracle SQL Analyze makes available for tuning the following statements.

Relation
Select Relation
SELECT c.street
FROM c.customer2 c WHERE c.customer_id='22222'

Update Relation
UPDATE customer2 c
SET c.zip='22222' WHERE c.customer_id='22222'

Delete relation
DELETE FROM customer2 c WHERE c.customer_id='22222'

Correlated sub query
Select sub query
SELECT c.street FROM addressr c
WHERE c.agent_id=(SELECT a.agent_id FROM new_agentr a
WHERE a.agent_id='22222')

Update sub query
UPDATE addressr SET addressr.zip='11987' WHERE addressr.zip='22222'
AND addressr.agent_id=(SELECT a.agent_id FROM new_agent a
WHERE a.agent_id='22222')

Delete sub query
DELETE FROM addressr WHERE addressr.zip='22222'
AND addressr.agent_id=(SELECT a.agent_id FROM new_agrntr a
WHERE a.agent_id='22222')

Object type (ADT)
Select ADT
SELECT c."ADDRESS".STREET FROM customer1 c WHERE c.customer_id='22222'

Update ADT
UPDATE customer1 c SET c.customer_id='11111' WHERE
"ADDRESS".ZIP=11111

Delete ADT
DELETET FROM customer1 c WHERE c."ADDRESS".ZIP=11111

VARRAY
Update VARRAY
UPDATE new_agent5 SET actor = address_t ('Engle 1111','Wood 222')
WHERE agent_id='22222'

Delete VARRAY
DELETE FROM new_agent5 WHERE agent_id='22222'

Nested table
Update Nested table
UPDATE new_agent SET agent_address = ADDRESS_TABLE(
   ADDRESS('H', '6042 Highland', 'Englewood','CO', '80112'),
   ADDRESS('W', '500 E. Arapahoe', 'Englewood', 'CO','80111'))
WHERE agent_id = '22222'
Delete Nested table
DELETE FROM new_agent WHERE agent_id = '22222'

3.1 Statistics & Discussion
The task of the Oracle optimizer is to find the most efficient method
for executing a SQL statement, but how do you know which optimizer
mode will be most efficient for you statement? Oracle SQL Analyzer lets
you test each of these execution strategies against a SQL statement and
provides performance statistics data that help you to determine which mode
is best [7].
Table(1) : Means of Measured Performance Parameter with the data types

<table>
<thead>
<tr>
<th>Kind</th>
<th>Cost</th>
<th>Elapsed Time</th>
<th>CPU Time</th>
<th>Logical Blocks Read</th>
<th>Physical Blocks Read</th>
<th>UGA Memory</th>
<th>PGA Memory</th>
<th>Table Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nested Table</td>
<td>A 4855</td>
<td>A 1265</td>
<td>A 365037</td>
<td>A 26347936</td>
<td>A 13421</td>
<td>C B 284586</td>
<td>C 1.9</td>
<td></td>
</tr>
<tr>
<td>Varray</td>
<td>B 317</td>
<td>B 2</td>
<td>B 1</td>
<td>B 24139</td>
<td>B 3463</td>
<td>C 63830</td>
<td>A 530689</td>
<td>A 99657</td>
</tr>
<tr>
<td>Relation</td>
<td>C 196</td>
<td>E 1</td>
<td>B 1</td>
<td>D 12532</td>
<td>C 582</td>
<td>D C 55065</td>
<td>B 387502</td>
<td>B 93600</td>
</tr>
<tr>
<td>Object Type</td>
<td>D 93</td>
<td>D 1</td>
<td>B 1</td>
<td>E 11496</td>
<td>D 485</td>
<td>B 191440</td>
<td>C B 335865</td>
<td>C 29</td>
</tr>
<tr>
<td>Sub queries</td>
<td>E 62</td>
<td>C 1</td>
<td>B 1</td>
<td>C 13828</td>
<td>E 243</td>
<td>D 47515</td>
<td>C 260641</td>
<td>B 93601</td>
</tr>
</tbody>
</table>

*Numbers of same letter mean no significant difference*

Table(1) shows the performances of the data type structure processing parameters. Generally noticed that nested table is the mostly consuming time in case of elapsed time, cpu time, physical and logical read. But it still uses highest area of the memory as found at UGA (USER GLOBAL AREA) and PGA (PROGRAMING GLOBAL AREA). But in reverse it consumes less table space for the other type. Except in the nested table, we found that varray, relational, type, sub queries have different letter from each other although they are statically different from each other (they have different Duncan’s letter) at the parameter from cost until logical read. For table space parameter nested table take the same of type as they have the same Duncan’s letter.

Table(2): Mean of measured performance with DML instructions

<table>
<thead>
<tr>
<th>PROC</th>
<th>Cost</th>
<th>Elapsed Time</th>
<th>CPU Time</th>
<th>Logical Blocks Read</th>
<th>Physical Blocks Read</th>
<th>UGA Memory</th>
<th>PGA Memory</th>
<th>Table Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update</td>
<td>A 771.3</td>
<td>A 158.3</td>
<td>A 51211</td>
<td>A 3204880</td>
<td>A 2892</td>
<td>A 244198</td>
<td>B 409643</td>
<td>B 84240.5</td>
</tr>
<tr>
<td>Delet</td>
<td>B 312.5</td>
<td>B 46.9</td>
<td>B 7088</td>
<td>B 1094594</td>
<td>B 1607</td>
<td>B 127478</td>
<td>C 193795</td>
<td>C 77540.5</td>
</tr>
<tr>
<td>Select</td>
<td>C 150.6</td>
<td>C 0.3</td>
<td>B 0</td>
<td>C 1210</td>
<td>C 470</td>
<td>C 76015</td>
<td>A 552443</td>
<td>A 93600.5</td>
</tr>
</tbody>
</table>

*Numbers of the same letters mean no significant difference*

From table(2) notice that from all DML instructions the UPDATE one has the most effective in consuming time of operations and less effective in table space consuming. SELECT instructions have exactly the reverse effect and have dramatically less effective while DELETE instruction in
Measuring of Oracle Optimizer Performance for Object-Relational Database entities

spite it differs from the other instructions, it nearly has the same effete of UPDATE.

Table(3): Represent the mean of measured performance with the size of processed data

<table>
<thead>
<tr>
<th>Size</th>
<th>Cost</th>
<th>Elapsed Time</th>
<th>CPU Time</th>
<th>Logical Blocks Read</th>
<th>Physical Blocks Read</th>
<th>UGA Memory</th>
<th>PGA Memory</th>
<th>Table Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>1%</td>
<td>E</td>
<td>D 10.7</td>
<td>C 2743</td>
<td>D 591767</td>
<td>F 988</td>
<td>D 119034</td>
<td>B 292800</td>
<td>D 76731.3</td>
</tr>
<tr>
<td>5%</td>
<td>C</td>
<td>C 43.1</td>
<td>C 811</td>
<td>C 791194</td>
<td>C 1521</td>
<td>C 146938</td>
<td>A 420211</td>
<td>C 84404.8</td>
</tr>
<tr>
<td>10%</td>
<td>B</td>
<td>B 134.7</td>
<td>B 27108</td>
<td>B 2394515</td>
<td>A 2938</td>
<td>A 275689</td>
<td>AB 374954</td>
<td>E 75965.2</td>
</tr>
<tr>
<td>20%</td>
<td>A</td>
<td>A 278.3</td>
<td>A 106321</td>
<td>A 5934814</td>
<td>B 2846</td>
<td>B 250893</td>
<td>A 408790</td>
<td>B 87651.6</td>
</tr>
<tr>
<td>40%</td>
<td>F</td>
<td>F 1.6</td>
<td>C 1</td>
<td>F 22453</td>
<td>E 1258</td>
<td>E 81849</td>
<td>AB 382173</td>
<td>B 87655.3</td>
</tr>
<tr>
<td>80%</td>
<td>D</td>
<td>E 3.4</td>
<td>C 2</td>
<td>E 41276</td>
<td>D 1294</td>
<td>F 65801</td>
<td>AB 314901</td>
<td>A 91160</td>
</tr>
</tbody>
</table>

* Numbers of the same letters means no significant difference

Data records size were measured by using 1, 2, .....80 percentage and that as of 60,000 records, the mean of measured performance parameter from the cost to logical read was increased from 1% to 20% begin and begin to drop with 40% till 80% with a significant difference between them and that it may be caused by using different mechanism with size more than 12000 records. But looking to the consumed table space it had been found that it increased according to the increasing of data size and there is a significant difference between them.

Table(4): Means of Measured Performance Parameter with the interaction between DATA Type & DML instruction

<table>
<thead>
<tr>
<th>Kind</th>
<th>Cost</th>
<th>Elapsed Time</th>
<th>CPU Time</th>
<th>Logical Blocks Read</th>
<th>Physical Blocks Read</th>
<th>UGA Memory</th>
<th>PGA Memory</th>
<th>Table Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nested table &amp; Update</td>
<td>A</td>
<td>A 6968</td>
<td>A 1801</td>
<td>A 588914</td>
<td>A 36119433</td>
<td>B 19021</td>
<td>A 2185555</td>
<td>B 455554</td>
</tr>
<tr>
<td>Nested table &amp; Delete</td>
<td>B</td>
<td>B 2319</td>
<td>B 621</td>
<td>B 96383</td>
<td>B 14622139</td>
<td>D 6702</td>
<td>B 563891</td>
<td>G 79425</td>
</tr>
<tr>
<td>Nested table &amp; Select</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Varray &amp; Update</td>
<td>K</td>
<td>C 317</td>
<td>C 3</td>
<td>M 108</td>
<td>A 23225</td>
<td>M 0</td>
<td>G 69919</td>
<td>A 6775880</td>
</tr>
<tr>
<td>Varray &amp; Delete</td>
<td>C</td>
<td>F 2</td>
<td>G 1</td>
<td>C 25053</td>
<td>E 3111</td>
<td>G 57741</td>
<td>C 361638</td>
<td>B 105715</td>
</tr>
<tr>
<td>Varray &amp; Select</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Object Type &amp; Update</td>
<td>G</td>
<td>G 188</td>
<td>F 1</td>
<td>H 16196</td>
<td>I 409</td>
<td>E 85291</td>
<td>D 285125</td>
<td></td>
</tr>
</tbody>
</table>
Table (4) shows that the interaction between data type collect and DML order is noticed. Updating and deleting nested table have the most consuming elapsed time with the most of the measured parameter of performance, but the other data type collection consumes very less time in hundred times with any kind of DML order, but table space gives little difference with any of interaction combination.

**Table(5) : Means of Measured Performance Parameter with the interaction between DATA size & DML instruction**

<table>
<thead>
<tr>
<th>Kind</th>
<th>Cost</th>
<th>Elapsed Time</th>
<th>CPU Time</th>
<th>Logical Blocks Read</th>
<th>Physical Blocks Read</th>
<th>UGA Memory</th>
<th>PGA Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nested table 1</td>
<td>D 329</td>
<td>J 88</td>
<td>C 23315</td>
<td>D 5001349</td>
<td>N 1874</td>
<td>D 356692</td>
<td>M 20741</td>
</tr>
<tr>
<td>Nested table 5</td>
<td>C 2245</td>
<td>C 617</td>
<td>D 11751</td>
<td>C 11403483</td>
<td>H 9207</td>
<td>C 1200766</td>
<td>A 774417</td>
</tr>
<tr>
<td>Nested table 10</td>
<td>B 4457</td>
<td>B 1140</td>
<td>B 230415</td>
<td>B 20295908</td>
<td>D 18353</td>
<td>B 1713398</td>
<td>H K G J I 224733</td>
</tr>
<tr>
<td>Nested table 20</td>
<td>A 29770</td>
<td>A 7768</td>
<td>A 2976980</td>
<td>A 165831301</td>
<td>A 48310</td>
<td>A 5250888</td>
<td>B 599720</td>
</tr>
<tr>
<td>Varray 1</td>
<td>P 159</td>
<td>F 159</td>
<td>P 1</td>
<td>O 9004</td>
<td>L 4609</td>
<td>Y 30580</td>
<td>K L 163097</td>
</tr>
<tr>
<td>Varray 5</td>
<td>O 161</td>
<td>I 159</td>
<td>R 1</td>
<td>W 4021</td>
<td>J 6258</td>
<td>X 33589</td>
<td>H D F G J E I 292209</td>
</tr>
<tr>
<td>Varray 10</td>
<td>N 664</td>
<td>H 159</td>
<td>O 1</td>
<td>R 5761</td>
<td>I 7950</td>
<td>B 23548</td>
<td>H K F G J I L 253574</td>
</tr>
<tr>
<td>Varray 20</td>
<td>M 169</td>
<td>G 159</td>
<td>K 1</td>
<td>L 11250</td>
<td>G 11058</td>
<td>A 23548</td>
<td>H K F G J I L 253574</td>
</tr>
<tr>
<td>Varray 40</td>
<td>L 179</td>
<td>E 159</td>
<td>G 2</td>
<td>K 16209</td>
<td>E 17866</td>
<td>Z 23648</td>
<td>K J L 190848</td>
</tr>
<tr>
<td>Varray 80</td>
<td>E 199</td>
<td>D 160</td>
<td>E 6</td>
<td>G 29239</td>
<td>B 31268</td>
<td>W 38308</td>
<td>L 141368</td>
</tr>
<tr>
<td>Object Type 1</td>
<td>W 78</td>
<td>P 15</td>
<td>B 0</td>
<td>B 832</td>
<td>P 871</td>
<td>E 184272</td>
<td>B 655160</td>
</tr>
<tr>
<td>Object Type 5</td>
<td>V 79</td>
<td>N 15</td>
<td>V 0</td>
<td>Y 2916</td>
<td>O 1851</td>
<td>H 83157</td>
<td>D F C E 358791</td>
</tr>
<tr>
<td>Object Type 10</td>
<td>U 81</td>
<td>O 15</td>
<td>T 0</td>
<td>V 4058</td>
<td>M 2861</td>
<td>J 18485</td>
<td>D C E 388323</td>
</tr>
<tr>
<td>Object Type 20</td>
<td>T 84</td>
<td>M 16</td>
<td>N 1</td>
<td>T 5506</td>
<td>K 5004</td>
<td>I 82296</td>
<td>H D F G E 332162</td>
</tr>
<tr>
<td>Object Type 40</td>
<td>S 91</td>
<td>L 16</td>
<td>I 2</td>
<td>Q 6763</td>
<td>F 11318</td>
<td>F 84503</td>
<td>C 472582</td>
</tr>
<tr>
<td>Object Type 80</td>
<td>R 104</td>
<td>K 17</td>
<td>F 2</td>
<td>J 17903</td>
<td>C 18380</td>
<td>G 84420</td>
<td>H D F G J E I 289916</td>
</tr>
</tbody>
</table>
Table (5) gives the of mean values of measured parameters data type collection with data size interactions. We can see in table (1) that nested table is the most effective in consuming processing time with every size of data, this effect increased as the size increased. Nested table with the size of 40% and 80% their measured parameter could not be done because of huge time of execution. And this is true from interaction of any of data type collection with the data size for most of the measured parameters.

Conclusion:
From this paper we conclude that:
- SQL analyzer is a good tool for studying optimum performance of SQL statements using statistics.
- Using statistical analyze represented by ANOVA table were very helpful in judgment between compromises, it enables to group the effects of data type, data size & DML language individually. Without any interference between each other, also we can get the interaction rule between data type and data size and DML. We found that nested table was the most effective from other types of data type, the object type was the least effective (consume lowest time of execution). We also found that increasing data size will increase any of performance data parameter until we reach size of 40% the time begin to degrees.
- Update statements show the most effective in consuming execution time & table space then Delete & Select statements, respectively.
REFERENCES


[5] Nori A., Bringing Object_Relational Technology to the Main Stream Vertis Corporation, University Avenue, Palo Alto, CA.


