RESEARCH ARTICLE

SAMRIDDHI Volume 14, Special Issue 2, 2022 Print ISSN : 2229-7111 Online ISSN : 2454-5767

Query Optimization in Object Oriented Database Using Cursor with Special Reference to Parallel Processing

Abhijit Banubakode[1*], Mohammed Gadhia[2]

[1*,2] MET Institute of Computer Science Mumbai, India; e-mail*: abhijitsiu@gmail.com

ABSTRACT

Intrusion Detection System is very important tool for network security. However, Intrusion Detection System suffers from the problem of handling large volume of data and produces high false positive rate. In this paper, a novel Grading method of ensemble has proposed to overcome limitation of intrusion detection system. Partial decision tree (PART), RIpple DOwn Rule (RIDOR) learner and J48 decision tree have used as base classifiers of Grading classifier. Optimzed Genetic Search algorithm have used for selection of features.These three base classifiers have graded using RandomForest decision tree as a Meta classifier. Experimental results show that the proposed Grading method of classification offers accuracies of 81.3742%, 99.9159% and 99.8023% on testing, training datasets and cross validation respectively. It is found that the proposed graded classifier outperform its base classifiers and existing hybrid intrusion detection system in term of accuracy, false positive rate and model building time.

Keywords: Grading Ensemble, RIDOR, Meta Classifier, Base Learner, AdaBoost**.**

SAMRIDDHI : A Journal of Physical Sciences, Engineering and Technology, (2022); DOI : 10.18090/samriddhi.v14spli02.18

INTRODUCTION

A cursor is an iterator that scans and processes a set of data (records/keys/tuples that meet specified criteria) one at a time. There are two sorts of cursors in RDBMSs: user cursors and system cursors. A user cursor is a cursor that is defined in a user application by utilising the SQL DCL CURSOR command. System cursors are those that the RDBMS creates and uses internally to access tables whose data is required to respond to user queries. To provide the output equivalent to a single user cursor, one or more system cursors may be employed [14]. The cursor can alternatively be referred to as a handle or a pointer to the context region. The user can enter the context region row by row with the cursor and learn some information about it. Explicit and implicit cursors are the two sorts of cursors. Oracle defines implicit cursor for every SQL query it processes. Explicit cursor is defined explicitly by the user in a PL/SQL block, whereas implicit cursor is defined implicitly by Oracle. [15,16,17] Object-Object-oriented programming is particularly well suited to the development of reusable components and complicated applications.

Corresponding Author : Abhijit Banubakode, MET Institute of Computer Science Mumbai, India; e-mail : abhijitsiu@gmail.com.

How to cite this article : Banubakode, A., Gadhia, M. (2022). Query Optimization in Object Oriented Database Using Cursor with Special Reference to Parallel Processing. *SAMRIDDHI : A Journal of Physical Sciences, Engineering and Technology,* Volume 14, Special Issue (2), 301-306.

Source of support : Nil

Conflict of interest : None

Object types are the foundation of object-oriented programming in PL/SQL. They simulate real- world objects, keep interfaces and implementation details separate, and maintain object-oriented data in the database. When building applications that interact with Java or other object-oriented languages, object types come in handy. An object type is a user-defined composite data type that represents a data structure as well as data manipulation capabilities and processes. Each variable in a scalar data type has a single value. All elements in a collection are of the same kind. Only object types link code to data, and attributes refer to the variables that make up the data structure. Methods refer to the object type's functions and processes. We discussed structured query language optimization in our previous paper [9,10,11,12,13]. In this paper, we look at a Retail Banking System's object-oriented database. We created query performance by generating multiple query plans for OODBMS and implemented cursor using a new approach of object oriented database. The latest paper [8] was referred to us. M. Tamer Ozsu discovered a transformation rule that generates a variety of algebra trees, with the plan generation stage producing many execution plans for each tree. Cursor implementation concerns in relation to object oriented databases are discussed by John Grant, Jarek Gryz, and Jack Minker [7].

The following is a breakdown of the paper's structure. Application and Preliminaries Notation are described in Sections II. The implementation of object-oriented schema design is discussed in Section III. Query optimization in an object- oriented database is discussed in Section IV. The experimental data are presented in Section V, and the conclusion is presented in Section VI.

PRELIMINARIES NOTATION

- **Application**

We're looking at a retail banking system as an example. The bank is divided into several branches, each of which is located in a different city and oversees the assets. Cust-id values are used to identify bank customers. Because the bank offers two types of accounts, a savings account and a checking account with a loan facility, the schema's relationships and characteristics are as follows [14]:

TRANS_DET_A (Trans_no,Inst_no,inst_dept,Inst_clr_dt,….)
FDSLAB_MAST_B (Fdslab_No, Minperiod, Maxperiod…….)
TRANS_MAST_C (Trans_no,Acct_Mo,Dt,Type,Dr_cr,Amt….)
FD_DET_D (Fd_ser_no,Fd_no ,Type,Payto_acctno,Period…)
ACCT_MAST_E (Acct_no, Sf_no,Lf_no,Branch_no,……)
ACCT_FD_CUST_DET_F(Acct_fd_no,Cust_no)
FD_MAST_G (Fd_Ser_No, Sf_No, Branch_No, Intro_Cust_No…**)**
CUST_MAST_H (Cust_No, Fname, Mname, Lname, Dob_Inc…..**)**
NOMINEE_MAST_I (Noinee_No, Acct_Fd_No, Name……**)**
BRANCH_MAST_J (Branch_No, Name……**)**
SPRT_DOC_K (Acct_Code, Type, Docs… .....**)**
ADDR_DET_L (Addr_No, Code_No, Addr_Type, Area_1, …**)**
CNCT_DET_M(Addr_No, Code_No, Cntc_Type, Cntc_Data.....**)**
EMP_MAST_N(Emp_No, Branch_No, Fname, Mname, Dept.....**)**

Figure 1: Object that is being considered for use in the banking system

- **Preparation and Alphabet**

Declaring a cursor gives the cursor a name and connects it to a select statement. A PL/SQL block's declaration section defines an explicit cursor. The syntax is

```
CURSOR1 cursor_name IS
SELECT_statement;
```

Where cursor name is the cursor's name. The select statement is the query that specifies the collection of rows that the cursor will process.

IMPLIMENTATION

We use a typical object-oriented schema of a retail banking system as the database to demonstrate object- oriented query optimizations.

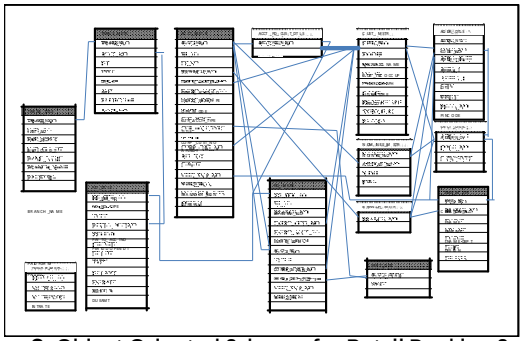


Figure 2: Object Oriented Schema for Retail Banking System

In optimization process we considered empaddress table as object and created modify address procedure and define the various member functions like area info, city info, state info, postalinfo, phone info and setphoneinfo as shown in Figure. 3 our aim is to find out the particular place in a city for that we use empaddress object. Then we created body empaddress as member procedure modify address if area_1 is null then application error exception is to be raised and 'The new address is not the correct address ' is to be print else all assignment operation is to be done. Further we created type address det as object it consist of various field like house name, house address and house name. Finally we implemented concept of cursor as shown in Figure. 4

```
CREATE OR REPLACE TYPE empaddress AS OBJECT
     (Area_11         VARCHAR2(100),
      Area_22         VARCHAR2(100),
      City1           VARCHAR2(100),
      State1          VARCHAR2(10),
      Pin_code1       VARCHAR2(5),
      Phone_number1   VARCHAR2(10),
MEMBER PROCEDURE   ModifyAddress
MEMBER FUNCTION postalinfo1 RETURN VARCHAR2,
MEMBER FUNCTION phoneinfo1 RETURN VARCHAR2,
MEMBER PROCEDURE setphoneinfo2 (newPhone
IN VARCHAR2));
```

```
  ( A_1 IN VARCHAR2, A_2 IN VARCHAR2, cty IN

   VARCHAR2, stat IN VARCHAR2,pin IN VARCHAR
   2),
  DECLARE
         this_society   addressdet;

        CURSOR all_society IS
          SELECT value (b) AS house
             FROM address_dtls b
            ORDER BY b.HouseName;
       BEGIN
       FOR one_society IN all_society LOOP
         this_society := one_society.house;
   dbms_output.put_line(this_society.HouseNa
  me || 'is situated in'
   ||   this_society.HouseAddress.city  || ''
   || this_society.HouseAddress.state);

                   END LOOP;
                    COMMIT;
                     END;
```

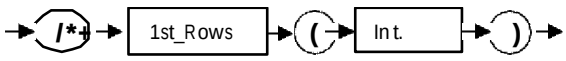
Figure 3: Creation of object and Member Procedure

OPTIMIZATION OF QUERIES IN OBJECT-ORIENTED DATABASE

If a query is complicated, query optimization is the process of selecting the most efficient query- evaluation approach among a choice of options. One aspect of optimization at the relational algebra level occurs when the system tries to identify an expression that is similar to a given application but more efficient to execute. Another factor to consider is developing a comprehensive strategy for addressing the query, such as the algorithm to use for executing the operation, the indices to use, and so on. To produce alternative query strategies, we employ Optimizer hints. Hints assist the optimizer in making decisions that it would otherwise make. Hints allow the optimizer to select a certain query execution plan based on a set of criteria. [4] The following are the different types of hints: Single-table hints are hints that apply to a single table or view. Single-table hints like INDEX and USE _NL are examples. Multi-table clues are similar to single-table hints, but they can indicate many tables or views. Because USE NL is a shortcut for USE NL and USE NL, it is not considered a multi-table hint. Single query blocks are affected by query block hints. Query block suggestions include STAR TRANSFORMATION and UNNEST. The entire SQL statement is affected by statement hints. A statement hint like ALL ROWS is an example. By putting clues for a SQL statement in a remark within the statement, Hint Syntax can communicate them to the optimizer. Following the SELECT, UPDATE, MERGE, or DELETE keyword, a block in a statement can only include one remark with clues. We employ the following types of tips in our experiments.

FIRST_ROWS(*N*)

The FIRST ROWS(n) hint tells Oracle to optimise a single SQL statement for speed by selecting the best plan for returning the first n rows.

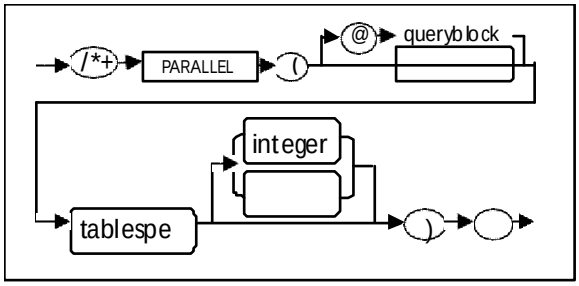


PARALLEL Hint

The PARALLEL hint specifies how many concurrent servers can be utilized for a parallel process. The hint is valid for the SELECT, INSERT, UPDATE, and DELETE parts of a statement, as well as the table scan part.

The hint is disregarded if any parallel limitations are broken.

Parallel hint: =



Query Plan1	**SELECT** /*+ ALL_ROWS1 */ value (b) AS house **FROM** address1_dtls b ORDER BY b.HouseName;
Execution Plan	Execution Plan 0 SELECT STATEMENT Optimizer=HINT: ALL_ROWS 1 (Cost=6 Card=50 Bytes=6615) 1 0 SORT (ORDER BY) (Cost=6 Card=45 Bytes=6615) 2 1 TABLE ACCESS (FULL) OF 'ADDRESS_DTLS' (TABLE) (Cost=4 Card=45 Bytes=6615) **Cost= 16 l Card=145 Bytes=19845 Elapsed: :00:00.35**

Tkprof Output

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.01	0.03	10	52	0	0
Execute	1	0.01	0.00	0	0	0	0
Fetch	4	0.00	0.00	0	4	0	44
total	6	0.02	0.03	10	56	0	44

Figure 4: The result of SQL trace and the tkprof utility

The execution plans generated by SQL trace are based on three different parameters: Cost, Cardinality, and Bytes. Oracle's inbuilt query optimizers estimate these parameters. Cost is the sum of CPU cost and IO Cost, Cardinality is the number of rows accessed, and Byte is the number of bytes accessed [5, 6]. Tkpr of accurately evaluates the SQL statements that an application executes the tkprof application formats the contents of the trace file and outputs it to a readable output file. Optionally, tkprof can calculate SQL statement execution plans and generate a SQL script that keeps the statistics in the database. tkprof displays the resources used by each statement, as well as the number of times it was called and the number of rows it processed. This data makes it simple to find the statements that are consuming the most resources. It has the following features: value of a call Parse: Converts a SQL statement into an execution plan, which includes checks for valid security authorization and the existence of tables, columns, and other referenced objects. Execute: actual Oracle's execution of the statement. This changes the data for the Insert, Update, and Delete commands. This identifies the selected rows in Select statements. Fetch: Retrieves records from a query's results.

Only SELECT statements with the following are subject to fetches. Count of SQL trace statistics: number of times the OCI procedure was executed, CPU: CPU time in seconds executing, elapsed: elapsed time in seconds executing, disc: number of physical reads of buffers from disc, query: number of buffers gotten for consistent read, current= number of buffers gotten in current mode (usually for update), rows: number of rows processed by the fetch or execute call.

INVESTIGATIONAL RESULTS

Table-1: Query Performance of SQL trace, tkprof and time statistics

Query Plans	Optimizer Hint	Cost	Card	Bytes	Count	CPU	Elapsed	Disk	Query	Throughput	Rows	Elapsed Time
Plan1	H1	8	135	19845	6	0.00	0.04	8	55	2.8571429	45	00:00:00.35
Plan2	H2	8	135	19845	6	0.01	0.02	0	52	2.5	45	00:00:00.40
Plan3	H3	10	315	46305	6	0.00	0.35	0	8	1.8867925	45	00:00:00.53
Plan4	H4	8	135	19845	6	0.00	0.03	0	51	2.7027027	45	00:00:00.37
Plan5	H5	8	135	19845	6	0.01	0.00	0	51	2.8571429	45	00:00:00.35
Plan6	H6	8	135	19845	6	0.00	0.00	0	51	2	45	00:00:00.50
Plan7	H7	8	135	19845	6	0.01	0.00	0	51	2.7027027	45	00:00:00.37
Plan8	H8	8	135	19845	6	0.01	0.00	0	51	2.5641026	45	00:00:00.39

Table-2: Optimizer hint for Query plan1 to Queryplan8

Sr. No	Optimizer Hint	Description
1	H1	/*+ ALL_ROWS */
2	H2	/*+ PARALLEL(L, 2) */
3	H3	/*+ PARALLEL(B, 2) */
4	H4	/*+ PARALLEL_INDEX(B, 2) */
5	H5	/*+ FULL(ADDRESS_DTLS) PARALLEL(ADDRESS_DTLS, 5) */
6	H6	/*+ FULL(ADDRESS_DTLS) PARALLEL(ADDRESS_DTLS, DEFAULT) */
7	H7	/*+ NO_ PARALLEL(ADDRESS_DTLS) */
8	H8	/*+ PARALLEL_INDEX(ADDRESS_DTLS,INDEX1 , 3) */

The output of tkprof is shown in Table 1. We built eight distinct query plants using algebraic transformations, and for each plan, we used the proper optimizer suggestion. After that, we used the SQL trace and tkprof utilities to get SQL trace statistics. Our statistical results reveal that as the optimizer suggestions change, query cost, cardiacity, and number of bytes do not change in seven of the eight plans, but they do in Plan 3. The PARALLEL hint provides the desired number of concurrent servers that can be employed for a parallel operation, hence the cardinality and amount of bytes are increased in plan3 /*+ PARALLEL (B, 2) */. The count indicates how many times the Oracle Call Interface procedure has been executed, parsed, or fetched. We discovered that the count is the same for all of the plans. All parse, execute, and fetch calls for the statement are measured in seconds. If TIMED STATISTICS is not enabled, this value is zero (0). Plan 5, 7, and 8 require 0.01 CPU time, whereas the rest of the plan requires 0.00. The total elapsed time for all parse, execute, and fetch calls for the statement is measured in seconds. If TIMED STATISTICS is not enabled, this value is zero (0). Because plan 3 employs parallel hint, the elapsed time for plan 3 is longer than the elapsed time for the other plans. The overall disc size is the disc size.

The total number of buffers retrieved in consistent mode for all parse, execute, and fetch calls is referred to as the query. For queries, buffers are usually retrieved in consistent mode. Buffers recovered for plans 1, 2, and 3 have a higher value than buffers retrieved for other plans. The total number of buffers retrieved in current mode is called current. For statements like Insert, Update, and Delete, buffers are retrieved in current mode.

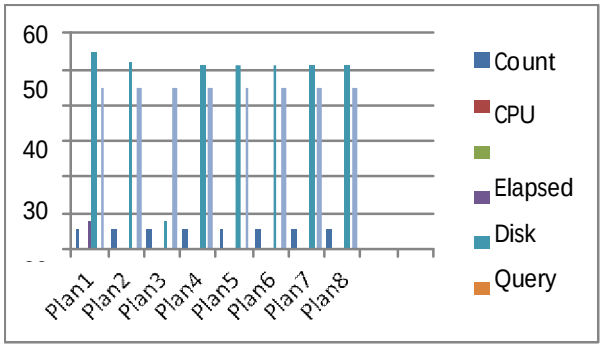


Figure 5: Histogram of Tkprf output

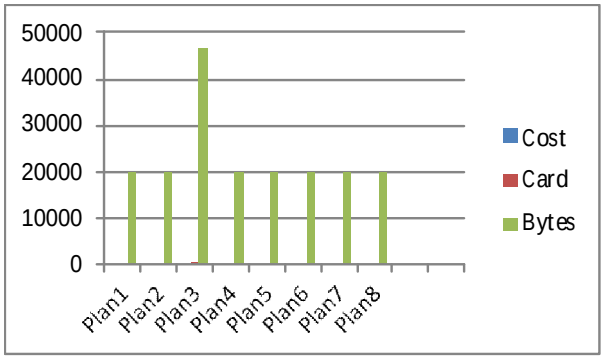


Figure 7: Query performance histogram for time statistics

CONCLUSIONS

The optimization of queries is one of the most difficult topics in Object-Oriented Databases. As a result of this issue, object-oriented query optimization is exceedingly difficult to achieve and is still in the research stage. This research should make a substantial addition to the field of Object Oriented Database Management. We can deduce from the foregoing findings that query cost, cardinality, and number of bytes remain unaffected, however elapsed time and other tkprof factors change once the cursor technique is implemented in an object-oriented database.

REFERENCES

[1] Delobel,C and Cluet,S., " A General Framework for the optimization of Object-Oriented Queries". Proceedings of the ACM SIGMOD Conference, pp. 383-392,1992.

[2] A.Quarati and E.Bertino arid "An Approach to Support Method Invocations in Object-Oriented Queries "dipartimento di Matematica - Univeresita di Genova.

[3] Amel Grissa-Touzi and Minyar Sassi, "Contribution to the Query Optimization in the Object-Oriented Databases" World Academy of Science, Engineering and Technology 11 2005.

[4] B. Ding, S. Das, W. Wu, S. Chaudhuri, and V. Narasayya. Plan stitch: Harnessing the best of many plans. Proceedings of the VLDB Endowment, 11(10): 1123–1136, June 2018.

[5] Silberschatz A and Korth H. F. , "Database System Concepts", 5th edition, McGraw-Hill.

[6] David Taniar,Eric Pardede and Wenny Rahayu,," Object Oriented Oracle",IRM press.

[7] A. Kipf, T. Kipf, B. Radke, V. Leis, P. Boncz, and A. Kemper. Learned Cardinalities: Estimating Correlated Joins with Deep Learning. In 9th Biennial Conference on Innovative Data Systems Research, CIDR '19, 2019.

[8] MySQL hints, https://dev.mysql.com/doc/refman/8.0/en/serversystem- variables.html#sysvar optimizer switch.

[9] Haridasa Acharya and Abhijit Banubakode "Query Optimization In The Object Oriented Database Using Nested Query", Proc. 3rd International Conference on Computer Modeling and Simulation ICCMS 2011 January 7 - 9, 2011, Mumbai, India.

[10] Haridasa Acharya and Abhijit Banubakode "Query Optimization in the Object-Oriented Database Using Views", Proc. International Conference On Computing ICC 2010,NewDelhi 27-28 December 2010.

[11] Haridasa Acharya,and Abhijit Banubakode "Query Optimization In The Object-oriented Database Using Equi-join" *Advances in Computational Sciences and Technology Print: ISSN 0973-6107, Online ISSN 0974-4738 Volume 4 Number 1 (2011) pp. 83–94 © Research India Publications.*

[12] Haridasa Acharya and Abhijit Banubakode "Query Optimization on Compressed and Decompressed Object- Oriented Database Using Operators" International Journal on Computer Science and Engineering (IJCSE) e-ISSN: 0975–3397 (online version); Print ISSN:2229-5631 (Print version);

[13] Seema Kedar and Abhijit Banubakode "Query Optimization in Compressed Database System" International Conference on Advance Computing (ICAC- 2008)" ACM Students Chapter Department of Computer Science and Engineering Anuradha Engineering College Chikhli-443 201, Maharashtra, India [14] Ivan Bayross,SQL,PL/SQL The programming language of oracle,BPB Publication.

[15] T. Neumann and G. Moerkotte, "An efficient framework for order optimization," in Proceedings of the International Conference on Data Engineering (ICDE), 2004.

[16] Antoshenkov, G. *Dynamic Query Optimization in Rdb/VMS*, Proc. International Conference on Data Engineering, Vienna, April 1993.

[17] Antoshenkov, G. *Dynamic Optimization of Index Scans Restricted by Booleans*, Proc. International Conference on Data Engineering, New Orleans, February 1996.