Volume 4, Issue 12, December 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Query Optimization and Execution Plan Generation in the Object-Oriented Databases under Inheritance

Abhijit Banubakode, G. S. Mate

Rajarshi Shahu College of Engineering,

Pune, India

Abstract— In this paper, we present an approach using object-oriented databases under inheritance that permits to enrich technique of query optimization existing in the object-oriented databases. Our experimental study shows performance of query after implementation of inheritance method using relational as well as object oriented database. Looking at the success of query optimization in the relational model, our approach inspires itself of these optimization techniques and enriched it so that they can support the new concepts introduced by the object oriented databases.

Keywords— Query Optimization, Relational Databases, Object-Oriented Databases, Inheritance, Cost, Cardinality and Bytes

I. INTRODUCTION

Query optimization plays an important role in object-oriented database system, without which performance of object oriented database system will not be yield very significant improvement. Conventional optimization techniques used in relational database systems were not design to cope with heterogeneous structures of the database and of particular not suitable to handle collection of objects [1].The object oriented paradigm is based on five fundamental concepts

- **Objec**t is an entity that has a well defined state and behavior each object is associated with a unique identifier called as object identity (OID).
- **Class** is a description of several object that have similar characteristics .Each object is an instance of some classes[2]
- ***Association*** refers to a connection between object instances. Association is basically a reference from one object to another that provides access paths among objects in a system.
- **Inheritance** relationship is known as generalization or specialization relationship in which definition of one class based on other existing classes. The former class is known as subclass, where as latter is the superclass.
- **Aggregation** is a composition or "part-of" relationship, in which a composite object consists of other component objects[3].

In or previous paper [9,10,11,12,13] we described query optimization without inheritance. In this paper we consider object oriented database of a book editing system .We implemented inheritance relationship using traditional rdbms as well as new method of object-oriented database and presented comparative analysis of query performance by constructing various query plans for rdbms and oodbms. We referred the recent paper[8] M. Tamer Ozsu identified a transformation rule that will generate a number of different algebra trees and the plan generation step will produce more than one execution plan for each of these trees. John Grant, Jarek Gryz, Jack Minker,[7] discuss the issues of inheritance with respect to object oriented database.

This paper is organized as follows. Next two sections describe preliminaries notation and implementation of object-oriented schema design. Section IV describe optimization in the object oriented database. Section V presents experimental results it includes the comparative analysis relational database and object oriented database using inheritance finally, Section VI gives conclusion**.**

II. PRELIMINARIES NOTATION

Queries in an inheritance hierarchy involve attributes of the class where the methods reside and attributes of their superclasses. Since a number of classes (at least two) are involved, a join operation to link all of these classes becomes necessary. The general format for the representation of queries in an inheritance hierarchy as shown in Fig.1 the from clause consist of a list of tables. These tables include all intermediate tables between a subclass (table1) and a super-class (table*n*). The inheritance join expression can be a join predicate to join all tables listed if the shared ID technique is used. Alternatively, if the current version of oracle is used, then it can be a *treat* expression to cast the selection from one class type to another within the inheritance hierarchy.


```
SELECT <function or expression>
FROM <table1, table2, …, tablek, …, tablen>
WHERE <inheritance join expression>
AND <tablek.OID = &input_OID>
```

Fig.1: The typical query under consideration

A subclass query is a query that retrieves information from the subclass (es), where the selection predicates are originated at the superclass. Fig.2 depicted the flow of a query in a subclass query. The general syntax and query representation for a subclass query is shown in Fig.3. The subclass-query representation is applicable if we implement the superclass and subclass as two different tables. We used the under features and the treat keyword provided by Oracle™ 9 and above.

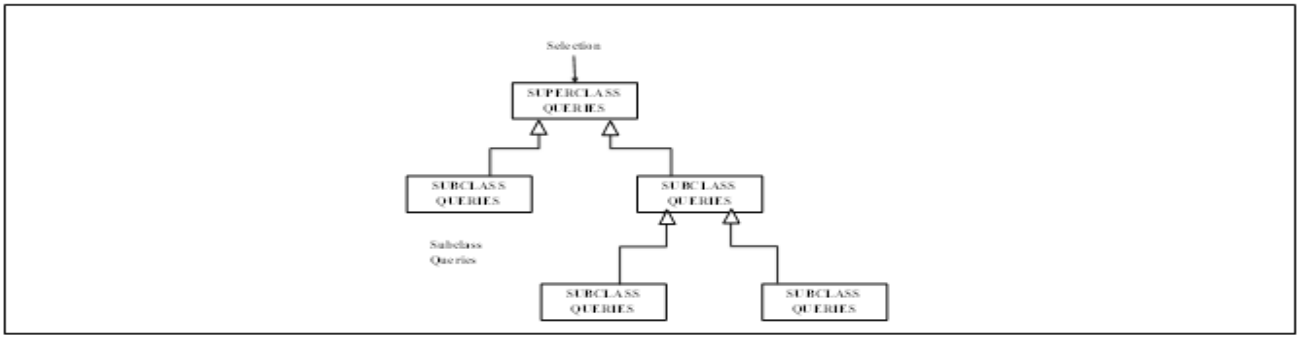


Fig.2: A typical object-oriented query flow

```
SELECT <sub-class attributes>
FROM <table1, table2,…….,tablen>
WHERE <join predicates>
AND <tablen.attr = &input_super-class_selection_predicates>where: Table1, …, table n-1 are subclass tables, and tablen is a superclass table.
```

Fig.3: Subclass query representation

III. IMPLIMENTATION

Author (Name Address)
Industry_Based (Company Name, Company Address, Company Size)
Academic (Institution Name, Institution Address, Number of Students)
Teaching_Staff (Total Contact Hours, ContactNo:<Varray>
Research_Staff (Research Topic, Research Director)
Subject (Subject Code, Subject Name, Venue
Book detail (ISBN, Title, Year)
Chapter (Chapter No, Chapter Title, Page No

Fig.4: Object Considered for book editing system

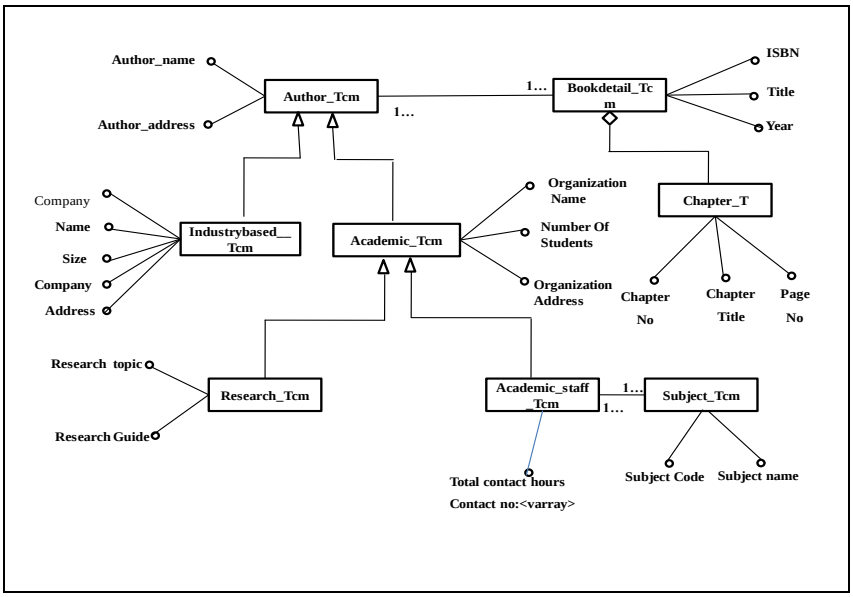


Fig.5: Object-Oriented Schema for Book Editing System

To illustrate the inheritance relationship we considered a typical objected schema of of Book Editing System [6]. Fig.4 shows the object considered for book editing system It consist of various objects like Author, Industry_Based, Academic,Teaching_Staff,Research_Staff, Subject, Book detail ,Chapter Fig.5 shows how we implemented an inheritance hierarchy in the object-oriented database. It consists of two inheritance relationships. First is the union inheritance between an author and an industry-based author and an academic author. Second is the mutual exclusion inheritance between an academic and a research staff and a teaching staff. Schema also shows an association relationships between the author, course manual, and the teaching staff, subject as well as aggregation relationship between the course manual and its chapters. To implement the course-manual authorship object-oriented model into Oracle we applied the following systematic steps: type and table. For this case, we use the types Author_TCM, Industry_TCM, Academic_TCM, Research_Staff_TCM, Teaching_Staff_TCM, Subject_TCM, Course_Manual_TCM and Chapter_TCM for each of them, we created the table respectively. We also use a type Contacts for the multiple-collection varray of the contact_no attribute in Teaching_Staff_TCM. Nested-table technique is used for the implementation of an aggregation relationship there are two inheritance relationships in the model. First is he inheritance between Author_TCM and the subclasses Industry_TCM and Academic_TCM. Second is the inheritance between Academic_TCM and its subclasses Research_Staff_TCM and Teaching_Staff_TCM. There are two association relationships from this model. The first one is between Author_TCM, Bookdetail_TCM and the second Teaching_Staff_TCM and Subject_TCM. Between Bookdetail_TCM and Chapter_TCM .There is one homogeneous aggregation relationship between Book_detail and Chapter_T Aggregation is implemented by using nested table technique.

IV. QUERY OPTIMIZATION IN OBJECT ORIENTED DATABASE

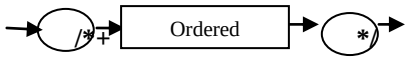
Query optimization is the process of selecting the most efficient query-evaluation plan from many strategies usually possible for processing a given query if the query is complex. One aspect of optimization occurs at the relational-algebra level, where the system attempts to-find an expression that is equivalent to given application, but more efficient to execute. Another aspect is selecting a detailed strategy for processing the query, such as choosing the algorithm to use for executing the operation, choosing the specific indices to use, and so on.

While generating different query plans we use Optimizer hints. Hints make decisions usually made by the optimizer. Hints provide a mechanism to the optimizer to choose a certain query execution plan based on the specific criteria. [4]

Hints falls into the following general classifications: Single-table hints are specified on one table or view. INDEX and USE_NL are examples of single-table hints. Multi-table hints are like single-table hints, except that the hint can specify one or more tables or views. The USE_NL is not considered a multi-table hint because it is actually a shortcut for USE_NL and USE_NL Query block hints operate on single query blocks. STAR_TRANSFORMATION and UNNEST are examples of query block hints. Statement hints apply to the entire SQL statement. ALL_ROWS is an example of a statement hint .Hint Syntax can send hints for a SQL statement to the optimizer by enclosing them in a comment within the statement. A block in a statement can have only one comment containing hints following the SELECT, UPDATE, MERGE, or DELETE keyword. Following types of hints we use in our experimentation.

ORDERED

The ORDERED hint causes Oracle to join tables in the order in which they appear in the FROM clause. If the ORDERED hint from a SQL statement performing a join omit then the optimizer chooses the order in which to join the tables.

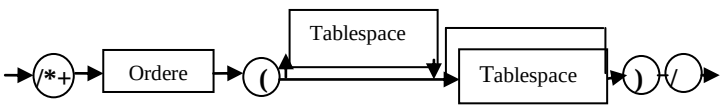


NO_CPU_COSTING

This hint turns CPU costing off for the SQL statement. The optimizer uses the I/O cost model which measures everything in single block reads and ignores CPU cost.

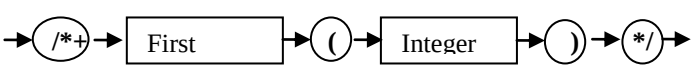
USE_MERGE

The USE_MERGE hint causes Oracle to join each specified table with another row source using a sort-merge join.



FIRST_ROWS (n)

The FIRST_ROWS (*n*) hint instructs Oracle to optimize an individual SQL statement for fast response, choosing the plan that returns the first *n* rows most efficiently.



As an example : **Suppose we want to find the details of the authors whose institution name is IIT, Delhi.**

The Superclass Query

Select A. Name, A.Address, C.Topic
From Authorcm A, Academiccm B, Reasearch_staffcmc
Where A.Ao_Id = B.Ao_Id
And B.I_Name = 'IIT, Delhi'

The Subclass Query

Select Contact_No
From Authorcm A, Teaching_Staffcm B
Where A.Ao_Id = B.Ao_Id
And A. Name = 'Abhijit'

as the optimization occurs at the object-algebra level so we formulated the object algebra for the super class and sub class query here we explain the algebraic implementation and the transformation for superclass query. In order to achieve transformation query rewrite rules are used

Query rewrites rules:

The overall goal of expression transformation is to reduce the cost of query evaluation by recasting the original algebraic expression as a more desirable one. This is accomplished by means of two sets of rules, algebraic and semantic, for the algebra operations defined in the language. Algebraic rules create equivalent expressions based upon pattern matching and textual substitution. Semantic rules are similar, but they are additionally dependent on the semantics of the database schema as defined by the type definitions and behavioral inheritance let us consider some example rules that one may develop for the object algebra operators we use μ_1, μ_2 μ_3 to denote the predicate, O1, O2, O3 denotes the object. In defining rules, we use the notation $E_1 \Leftrightarrow E_2$ which specifies that expression E_1 is equivalent to expression E_2[5]. Rule (1) captures commutatively of select. Rules (2) depicted join operations and natural joins are commutative. Rule (3) depicted conjunctive selection operations can be deconstructed into a sequence of individual selections.

$$\sigma_{\mu 1}(\sigma_{\mu 2}(O)) \Leftrightarrow \sigma_{\mu 2}(\sigma_{\mu 1}(O)) \quad (1)$$

$$O_1 \bowtie_{\mu 2} \Leftrightarrow O_2 \bowtie_{\mu} O_1 \quad (2)$$

$$\sigma_{\mu 1 \wedge \mu 2}(O) \Leftrightarrow \sigma_{\mu 1}(\sigma_{\mu 2}(O)) \quad (3)$$

$$(O_1 \bowtie O_2) \bowtie O_3 \Leftrightarrow O_1 \bowtie (O_2 \bowtie O_3) \quad (4)$$

$$(O_1 \bowtie_{\mu 1} O_2) \bowtie_{\mu 2 \wedge \mu 3} O_3 \Leftrightarrow O_1 \bowtie_{\mu 1 \wedge \mu 3} (2 \bowtie_{\mu 2} O_3) \quad (5)$$

$$\sigma_{\mu 0}(O_1 \bowtie_{\mu} O_2) \Leftrightarrow (\sigma_{\mu 0}(O_1)) \bowtie O_2 \quad (6)$$

$$\sigma_{\mu 1 \wedge \mu 2}(1 \bowtie_{\mu} O_2) \Leftrightarrow (\sigma_{\mu 1}(O_1)) \bowtie_{\mu} (\sigma_{\mu 2}(O_2)) \quad (7)$$

Fig.5: Transformation Rules

Rule(4) describe about the Association of natural join Rule(5) also depicted Association of join where μ_2 involves attributes from only O_2 and O_3. Rule (6)-(7) Explain the selection operation distributes over the mue join operation under the following two conditions first when all the attributes in μ_0 involve only the attributes of one of the expressions (O_1) being joined and second When θ_1 involves only the attributes of O_1 and μ_2 involves only the attributes of O_2. Further we consider the algebraic transformation for the mention superclass query

Initial Query

$\Pi_{A.name,A.address,C.topic}(\sigma_{A.ao\_id = B.ao\_id \wedge B.Iname = \text{'IIT,Delhi'}}$
(AuthorcmA $\bowtie$ (Academiccm B $\bowtie$ Research_staffcmC)))

Transformation 1

$\Pi_{A.name, A.address, C.topic}(\sigma_{A.ao\_id = B.ao\_id \wedge B.Iname = \text{'IIT, Delhi'}}$
((Authorcm A$\bowtie$AcademiccmB)$\bowtie$ Research_staffcmc))

Transformation 2

$\Pi_{A.name, A.address, C.topic}((\sigma_{A.ao\_id = B.ao\_id \wedge B.Iname = \text{'IIT, Delhi'}}$
(AuthorcmA$\bowtie$Academiccm B))$\bowtie$ Research_staffcmC)

Transformation 3

$\Pi_{A.name, A.address, C.topic}(\sigma_{A.ao\_id = B.ao\_id}(\sigma_{B.Iname = \text{'IIT, Delhi'}}$
(Authorcm A$\bowtie$ AcademiccmB))$\bowtie$ Research_staffcmC)

Final Query

$\Pi_{A.name, A.address, C.topic}(\sigma_{A.ao\_id = B.ao\_id}$ (Authorcm A $\bowtie$ Academiccm B) $\bowtie \sigma_{B.Iname = \text{'IIT, Delhi'}}$ (Academiccm B)) $\bowtie$ Research_stafcmC

For initial query we never applied selection predicate directly to the AuthorcmA relation, since predicate involves attributes of both the AuthorcmA and AuthorcmB Relation .However we applied Rule 4 associativity of natural join to transform the join AuthorcmA ⋈ (Academiccm B ⋈ Research_staffcmC) to ((Authorcm A ⋈ AcademiccmB) ⋈ Research_staffcmC) as shown in transformation 1. On applying rule no 6 we rewrite the query as shown in transformation 2 then using Rule 1 we break the selection into two selection as shown in transformation 3 both of the expression select tuples with A.ao_id = B.ao_id and B.Iname ='IIT,Delhi' the latter form of the expression provides a new opportunity to apply Rule no 6 i.e. perform selection early ,resulting the final query similar transformation we have made for the mention query and obtained various query plans in order to alter the query execution plans we use different optimizer hints like ordered,no_cpu_costing,merge and first rows after that we executed query on oracle 10g and obtained the query execution plans based on three different parameter i.e. cost, cardinality and bytes. These parameters are estimated by inbuilt query optimizer's of oracle. Cost is the combination of cpu_cost and io_cost ,cardinality is the number of rows accessed by the operation and byte is the number of bytes accessed by the operation.[5,6]The execution plans obtained using oracle 10g for above cases are:

Execution Plan for the superclass query

```
---------------------------------------------------------------------------
 0 SELECT STATEMENT Optimizer=ALL_ROWS
      (Cost=5 Card=47 Bytes=1457)
 1   0   HASH JOIN (Cost=5 Card=47 Bytes=1457)
 2   1 TABLE ACCESS (FULL) OF 'ACADEMICCM'
       (TABLE) (Cost=2 Card =47 Bytes=423)
 3   1 TABLE ACCESS (FULL) OF 'AUTHORCM'
     (TABLE)   (Cost=2 Card=50 Bytes=1100)
```

Execution Plan for the subclass query

```
---------------------------------------------------------------------------
 0     SELECT STATEMENT Optimizer=ALL_ROWS
        (Cost=5 Card=13 Bytes=1027)
 1     0   HASH JOIN (Cost=5 Card=13 Bytes=1027)
 1     1 TABLE ACCESS (FULL) OF 'AUTHORCM'
          (TABLE)  (Cost=2 Card=13 Bytes=143)
 2     1TABLE ACCESS (FULL) OF 'TEACHING_
        STAFFCM'(TABLE) (Cost= 2 Card=50 Bytes=3400)
```

For rdbms we implemented inheritance relationship using primary key and foreign key relationship in order to simulate relationship between superclass and its subclass. For oodbms we implemented inheritance relationship using "under" keyword. Table1 depicted the query performance of oodbms based on query performance measuring parameters. We have constructed the five different plans of queries using algebraic transformation for every plan we use appropriate optimizer hints then query plans are tested under oracle 10g environment. Table shows the output of Sql trace and tkprof utility .The output of sql trace is the execution plans based on three different parameter i.e. cost, cardinality and bytes. These parameters are estimated by inbuilt query optimizer's of oracle. Cost is the combination of cpu_cost and io_cost, cardinality is the number of rows accessed by the operation and byte is the number of bytes accessed by the operation.[5,6].

.

V. EXPERIMENTAL RESULTS

Table1**:** Query Performance of Sqltrace ,Tkprof and Time statistics for OODB

Query Plans	**Optimizer Hint**	**Cost**	**Card**	**Bytes**	**Count**	**CPU**	**Elapsed**	**Disk**	**Query**	**Throughput**	**Rows**	**Elapsed Time**
Plan0	-	1 4	89	1910	5	0.00	0 .00	0	8	33.3333 333	28	00:00:00.03
Plan1	H1	1 4	89	1910	5	0.00	0 .00	0	8	33.3333 333	18	00:00:00.03
Plan2	H2	1 4	89	1910	6	0.00	0 .00	0	8	25	18	00:00:00.04
Plan3	H3	22	152	3040	5	0.00	0 .00	1	6	33.3333 333	18	00:00:00.03
Plan4	H4	33	41	930	5	0.01	0 .02	1	26	25	18	00:00:00.04
Plan5	H5	10 7	78	1790	5	0.01	0 .02	1	26	25	18	00:00:00.04

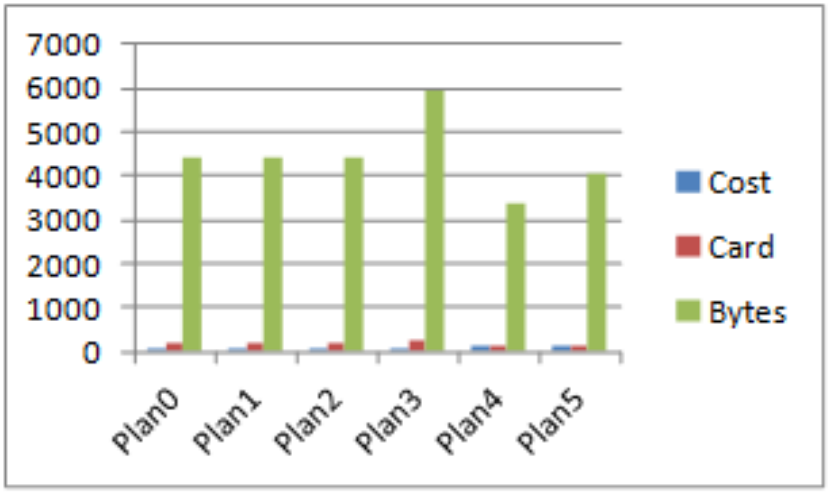


Fig.6: Query performance Histogram for RDB

Plan1 uses ordered hint with this cost is less but cardinality and bytes are varied in oodbms. plan2 uses no_cpu_costing hint with this cost is less but cardinality and bytes are varied in oodbms. Plan3 uses use_merge hint with this cost is less but cardinality and bytes are varied in oodbms. Plan4 uses first_row hint with this cost, cardinality and bytes are varied in oodbms. Plan5 uses first_row hint with different transformation with this cost is less but bytes and cardinality are varied in oodbms

VI. CONCLUSIONS

One of the biggest problems in Object Oriented Database is the optimization of queries. Due to these problems optimization of object-oriented queries is extremely hard to solve and is still in the research stage. This work is expected to be a significant contribution to the Object Oriented Database Management. From above results we could conclude that cost of query is not affected while cardinality and no of bytes are affected after implementation of inheritance method using under keyword in object-oriented database.

REFERENCES

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

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

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

[4] "Understanding Optimizer hints", Oracle database Performance Tuning Guide

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

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

[7] John Grant, Jarek Gryz, Jack Minker, Fellow, IEEE, And Louiqa Raschid, Member, IEEE "Logic-Based Query Optimization for Object Databases " IEEE transactions on knowledge and data engineering, VOL. 12, NO. 4, JULY/AUGUST 2000

[8] M. Tamer Ozsu," Query Processing Issues in Ob ject-Oriented Database Systems Preliminary Ideas "Laboratory for Database Systems Research Department of Computing Science University of Alberta Edmonton, Alberta, Canada.

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

[11] Abhijit Banubakode and Haridasa Acharya "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] Abhijit Banubakode and Haridasa Acharya "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] Abhijit Banubakode and Seema Kedar "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-443201, Maharashtra, India

ABOUT AUTHOR

Dr. Abhijit Banubakode received Ph.D. degree in Computer Studies from Symbiosis Institute of Research and Innovation (SIRI), a constituent of Symbiosis International University (SIU), Pune, India in April 2014 and ME degree in Computer Engineering from Pune Institute of Computer Technology(PICT), University of Pune ,India in 2005 and BE degree in Computer Science and Engineering from Amravati University, India, in 1997. His current research area is Query Optimization in Compressed Object-Oriented Database Management Systems (OODBMS). Currently he is working as Professor and Head of Department (HOD) in Department of Information Technology, Rajarshi Shahu College of Engineering,Pune, India. He is having 16 years of teaching experience. He is a member of International Association of Computer Science and Information Technology (IACSIT), ISTE, CSI and presented 12 papers in International journal and conference.

Gitanjali S. Mate received ME degree in Computer Engineering from D.Y. PAtil College of Engineering Akurdi Savitribai Phule Pune Vidyapeeth, India in 2007 and BE degree in Computer Science and Engineering from Pune Institute of Computer Technology, India, in 1999. Currently working as Assistant Professor in Department of Information Technology, Rajarshi Shahu College of Engineering, and Pune, India having 14 years of teaching experience. Also she is a member of , Indian Society for Technical Education (ISTE), Computer Society of India (CSI) and presented eight papers in International and National conference and four paper in International Journal

