Chapter 3

Fragmentation & Access Strategies Using Genetic Algorithms

This chapter is organized as follows: Section 3.1 contains a general introduction to the whole chapter, section 3.2 focuses on introducing the fragmentation process, its types and advantages, Section 3.3 introduces Access Strategies and section 3.4 describes formulation of cost model and objective function. It also describes cost estimation formulation three major local disk access path strategies: Clustered Index Scan, Unclustered Scan and Sequential Scan. In section 3.5 a general outline of the genetic algorithm for fragmentation and access strategies is given along with detailed description of genetic parameter settings and operator’s (Selection, Crossover, Mutation) design. Section 3.6 illustrates functioning of GA_FA’s ‘access cost methods’ by taking an ‘example transaction profile’. Finally Section 3.7 discusses experimental results and analysis. Finally Section 3.8 provides a summary of the chapter.

3.1 Introduction

Distributed Databases Design Process is a very complex process due to the huge size of the problem and interdependency of many complex sub-processes. One of the major objectives of the design is to reduce the movement of irrelevant data across the sites. Moreover data that is accessed together frequently, should be placed together or nearby. This can be achieved by organizing the data at storage sites according to the access pattern behaviour shown by the most active queries. The exact study and forecast of the behaviour of all possible user transactions is an intractable job. This demands application of heuristics and probabilistic solutions. For example, one such interesting heuristic used extensively in query design field is that for studying Query Access Pattern one needs to study, only 20% of the Total Queries on the database. This due to a popular 20-80 theory, established well by the database research community which states that ‘the most active 20% of user queries account for 80% of the data access’,[Ozsu et

---

6 GA_FA: Proposed Genetic Algorithm for Fragmentation & Access Strategies
In this chapter, use of Fragmentation and suitable Access Strategies as heuristic procedures to achieve query optimization is illustrated. This is done by using Genetic Algorithms as an implementation tool, and experimenting with a large number of example transaction profiles [Appendix-B], acting on an experimental database.

In a relational database, data is grouped as records at the physical level, called Relations. For improving the storage and operating efficiency these Relations may be Fragmented and physically distributed over multiple record segments stored as sub files in secondary memory of a local site or on secondary memory of multiple sites in case of a Distributed Database. Access Paths are defined as algorithms and structures that are used to store, retrieve and update the segment instances at a local site. A few examples of Access Paths are like Sequential, Indexed Sequential, Direct Access, Hashed File organizations [March,1983]. A cost based query optimization requires the least possible references to secondary memory or least number multiple sites of a Distributed System. A synergic combination of Fragmentation and Access Strategies can help achieve this goal and is the context and content of the present chapter.

3.2 Fragmentation

Fragmentation is the process of splitting a relation into a number of smaller subrelations called Fragments. There are two main types of fragmentation: Horizontal Fragmentation and Vertical Fragmentation. A third type known as Mixed Fragmentation is also prevalent but it’s just a mix of before said two main strategies. Horizontal Fragmentation splits the table horizontally into subsets of tuples as is done by the Selection Operator (σ), of Relational Algebra. Whereas Vertical Fragmentation splits the table vertically into subsets of attributes as is done by the Projection Operator (π), of Relational Algebra. We need to fragment relations so as to optimize Query Processing. A lot of research has taken place in the database field to answer an important question: Why Fragment? There are following four major reasons to fragment relations for advantage, as listed in database literature [Conolly & Begg 03], [Ozsu et al.2006][Alom et al. 2009].

Usage Pattern:

Database applications work with views rather than entire relations. Views are a logical concept and may contain subsets of different relations. So, data distribution in distributed databases should take subsets of relations (Fragments)
as the units of distribution instead of complete relations. This may reduce unnecessary replication of data at various sites and hence reduce Integrity Control Costs.

- **Efficiency:**
  Data can be stored at various sites in such a manner that a site keeps the copy of a fragment that it frequently queries. This way it will minimize the communication cost component of the total query costs.

- **Parallelism**
  Several subqueries using different fragments can be executed concurrently. Interprocess and Intraprocess Parallelism as well is feasible easily due to fragmented design. This shall result in efficient query response times.

- **Security**
  Data can be stored at various sites in such a manner that a site keeps the copy of a fragment that it frequently queries. A common user’s view is restricted mostly to the data he needs. Other data may reside in different fragments not available to him and thus enhancing the overall security environment of database.

As listed above, fragmentation provides many advantages while designing a distributed database. But it brings along a few disadvantages as well. It may cause the performance degradation of global queries, requiring joining of many data fragments. Integrity Control management becomes more difficult when many replicated fragments are there at different sites and are frequently updated. Advantages offered by good fragmentation strategy have far more positive outcomes than the disadvantage component. It has been applied in Central databases, Multiprocessor and Parallel databases, parallel disk I/O Systems and Distributed DBMS as well. All of these systems use advantages offered by fragmentation [March, 1983], [Dewitt & Gray, 1992], [Ozsu et al.2006].

### 3.3 Access Strategies

From a global view point, Access Strategies refer to procedures applied to minimize the movement of irrelevant data across the network sites and from a local viewpoint it refers to access path selection by minimizing the movement of data from secondary memory to main memory while doing local processing at a particular site. This
can be achieved by organizing the data at storage sites according to the access pattern behaviour shown by the most active queries. To reduce local processing costs data commonly required together should be placed physically together or close on disk. This is done to reduce the number of accesses to secondary memory on a local site. While designing a cost model well established methods and literature by the prominent researchers and authors in the database field has been referred as by [Chu (1992)] [CU90] [Navathe et al. (1984) [MT83] [Cardenas, 75]. Access strategies to reduce CPU and I/O costs at a particular site are no different in a distributed database from a central database, when one is optimizing a local site’s local disk access strategy. To simulate a distributed system’s communication cost component, a penalty constant value is added to local access cost strategy, whenever it is attempting to retrieve data from multiple fragments, which may be stored at different sites.

Selection of an access path is not independent from fragmentation strategy and vice versa. Fragmentation scheme fixes the size of a fragment which in turn happens to be the main factor to decide an access path. Due to this interdependency of fragmentation and access path selection, one cannot prefix the access strategy and hence the cost functions. Deterministic optimization procedures require cost functions to be pre defined, so these are not easily applicable to vertical fragmentation problem of distributed databases. Vertical fragmentation problem can be expressed as a combinatorial optimization problem but in most cases it is a NP Hard problem and most of earlier proposed heuristic solutions tend to go intractable when number of transactions and attributes are increased ([Chu(1992)],[Chu & Ieong,93]). In this thesis a genetic algorithm solution is proposed (GA_FA) which does not choke on increasing number of transactions or attributes and is computationally very fast as compared to deterministic procedures, which are usually based on exhaustive enumeration and heuristics.

Initial formulation of genetic algorithm for fragmentation and access strategies (GA_FA) concentrates on reducing the local CPU and I/O costs by effecting a binary fragmentation of the input relation and then choosing an access strategy. While choosing an access strategy, Objective function is to minimize the total cost of a transaction. Total Cost of a Transaction from given Transaction Profile may be represented in terms of total number of disk accesses incurred by that transaction [CU90]. These costs are determined by estimating costs of alternative access paths for a fragmentation scheme as in case of a central database, a cost component is added to the cost of a transaction if it accesses data
from more than one fragment, so as to simulate communication cost involved in accessing distributed fragments. In this study three major access strategies are analysed, compared and optimized using genetic methods, for a local site’s query optimization. These are sequential scan, clustered index scan and unclustered index scan on the lines of studies of Barker & Jun [Barker et al.2006], Chu & Ieong [Chu & Ieong,93], Cornell & Yu[CU90], March[March,1983].

3.4 Cost Model and Objective Function Formulation :

3.4.1 Cost Model Definitions & Assumptions :

In this section a cost model is developed for vertical fragmentation and access strategy selection as its synergic ally. In this study only one type of fragmentation i.e. Vertical Fragmentation is assumed while optimizing a query process, as it is the most widely followed approach for distributed databases. A relational database system with structure similar to System R [Astrahan et al.,1976] is considered for illustration. We assume that a set of pre decided retrieval transactions is given on a relation along with the subset of attributes required by that transaction and relation’s attribute sizes are known apriori.

Relation attributes are categorised as key on non-key attributes. If an attribute is a Primary Key then it’s called Clustering Attribute and if it’s a Secondary Key then is referred as Non-Clustering. Selectivity\(^7\) of the attributes and frequency of each transaction is known a priori. Restrict Attribute is an attribute which may be used by the query predicate to limit the selection of rows or columns of a relation. In case of transaction having a single restrict attribute, this attribute is used as a scan attribute. When there are more than one restrict attributes then Scan attribute will be the attribute which will be used to scan the relation. Scan per run for a transaction is a term giving the number of times a relation is scanned to complete that transaction. Scan per run is more than one, in case the relation is in an inner loop of an inner outer loop join.

After fragmenting the relation vertically in two fragments, we need to keep track of parent as well as all child fragments generated, this may be achieved in two ways,

---

\(^7\) Selectivity of an attribute is the ratio of the selected tuples by a subquery predicate to the total number of tuples in a relation.
either by replicating primary key in each fragment or by using concept of Tuple Identifiers (T_id) [MT83] [CU90]. Any tuple can be accessed with direct access methods using T_id, which is a combination of a page number and offset.

Objective function is to minimize the total cost of a transaction, which may be represented in terms of total number of disk accesses incurred by that transaction in one scan run multiplied by the frequency of that transaction [CU90]. After estimating access costs of alternative access strategies, evaluation is done for these three types of access strategies: clustered index scan, unclustered index scan, and the sequential scan, least costing strategy is selected.

3.4.2 Cost Model

The process of Binary Vertical fragmentation splits the relation into two fragments. Which are referred in database literature as Primary Fragment and the Secondary Fragment? Primary Fragment is the one which contains the scan attribute. All the attribute details required by a query result may reside in Primary Fragment or in both Primary as well as Secondary Fragment. Therefore total number of Disk Accesses required to retrieve tuples is sum of disk accesses required from Primary and Secondary Fragments. A set of mathematical formulae to estimate number of disk accesses by various alternative access paths are used as in case of local disk access of central databases, explained extensively in database literature (Cardenas,75) [Yao,1977] [PM79] [March,1983] [CU90].

Case 1)

**Estimating Access Costs to scan Primary Fragment:**

Let A_s be the scan attribute of the transaction and D_1 be the number of disk accesses required to scan the primary fragment. We estimate the disk access costs for three access strategies as following.

**Case 1a) : Clustered Index Scan:**

When clustering attribute A_c is used as a scan attribute A_s, then the number of disk accesses is estimated as

\[ D_{cis} = \rho_{Ac} \ C_s S_{PF} / B \]  \hspace{1cm} (3.1)
Where
\( D_{cis} \) is the estimated number of disk accesses for clustered index scan
\( \rho_{Ac} \) is the selectivity of selection predicate on scan attribute which happens to be clustering attribute
\( C_r \) is the relation cardinality
\( S^{pf} \) is the tuple size of primary fragment in bytes including \( L_{id} \) (length of tuple identifier \( T_{id} \))
\( B \) Block Size (Page Block Size assumed is 4k bytes)

Case 1b) : Unclustered Index Scan:

When the scan attribute is unclustered attribute \( Au \) then Number of Disk Accesses estimated are given as[Cardenas,75].
\[
D_{uis} = P_f (1-(1-1/ P_f)T_n + \rho_{uc} N_{ui})
\]  
\[
T_n = \rho_{Au} C_r
\]
\[
P_f = C_r S^{pf}/ B
\]

Where
\( D_{uis} \) is the estimated number of disk accesses for unclustered index scan
\( T_n \) is the number of tuples satisfying selection condition on unclustered scan attribute \( Au \)
\( P_f \) is the Page size of Primary Fragment including \( L_{id} \)
\( \rho_{Au} \) is the selectivity of selection predicate on a scan attribute which is an unclustered index attribute
\( C_r \) is the relation cardinality
\( S^{pf} \) is the tuple size of primary fragment in bytes including \( L_{id} \) (length of tuple identifier \( T_{id} \))
\( B \) Block Size (Page Block Size assumed is 4k bytes)

Case 1c) : Sequential Scan:

For a sequential scan number of disk accesses are given by
\[
D_{ss} = C_r S^{pf}/ B P_f
\]  
\( P_f \) = Fetch Blocking Factor.
Finally the selection criteria is summarised followed

If scan attribute is the clustering attribute itself then

\[ D_1 = \text{Min}\{D_{cis}, D_{ss}\} \]

If scan attribute is the unclustering attribute but list of restrict attributes include the restrict attribute then

\[ D_1 = \text{Min}\{D_{cis}, D_{uis}, D_{ss}\} \]

If scan attribute is the unclustering attribute but list of restrict attributes don’t include the restrict attribute then

\[ D_1 = \text{Min}\{D_{uis}, D_{ss}\} \]

**Case 2) Estimating Access Costs to scan the Secondary Fragment:**

If all the required attributes by a selection predicate are not found in the Primary fragment, then additional disk accesses are needed to scan the secondary segment to retrieve the attributes present in secondary fragment. Let \( D_2 \) be the number of Disk Accesses required in case of scanning the secondary fragment with scan attribute \( A_s \). If there are ‘k’ restrict attributes in the predicate restriction and \( \rho_{ai} \) is the selectivity of attribute \( a_i \) then, overall selectivity of restrict attributes is the product of ‘k’ selectivities of independent restrict attributes. Total number of tuples selected from the primary fragment are given by the following equation

\[ T^p = \rho_{as} \prod_{i=2}^{k} \rho_{ai} \quad (3.4) \]

Three alternatives to scan the secondary fragment are evaluated individually depending upon the access path used to scan the Primary Fragment as following

Case 2a)

Primary Fragment accessed using a Sequential Scan:

When the primary fragment was accessed using sequential scan, one can use tuple identifiers or sequential scan for the secondary fragment. Use of tuple identifiers (\( T_{id} \)) is beneficial if \( T^p \) is small, and sequential scan is better otherwise. Using \( T_{id} \)
the number of disk accesses can be estimated using Cardenas’s formulation [Cardenas, 75] as

\[ D_{2i} = P_r (1-(1-1/ P))^T_n \]  \hspace{1cm} (3.5)

\[ T_n = T^p C_r \]
\[ P_r = C_r S^{sf} / B \]

Where

- \( D_{2i} \) is the estimated number of disk accesses for secondary fragment scan
- \( T_n \) is the number of tuples satisfying selection condition on primary fragment
- \( P_r \) is the Page size of Primary Fragment including L_id
- \( C_r \) is the relation cardinality
- \( S^{sf} \) is the tuple size of secondary fragment in bytes including L_id (length of tuple identifier T_id)
- \( B \) Block Size ( Page Block Size )

But if sequential scan is used for Primary Fragment then

\[ D_{2s} = C_r S^{sf} / B F_p \]  \hspace{1cm} (3.6)

Where

\[ F_p = \text{Prefetch Blocking Factor.} \]

Finally an access path is chosen by deciding on minimum cost from above two cases for scanning a secondary fragment. The number of disk accesses are given by

\[ D_2 = \text{Min} \{ D_{2i}, D_{2s} \} \]  \hspace{1cm} (3.7)

Case 2b)

When unclustered index scan was used to scan the primary fragment
Similar to case 2a) one may chose tuple identifier method of sequential scan to access attributes from secondary fragment, choice decision is again influenced by the method which results in lesser number of disk accesses

\[ D_2 = \min \{ D_{2i}, D_{2s} \} \quad (3.8) \]

Case 2c)

When Primary Fragment was scanned using clustered index scan, one may go by sequential scan or Tuple Identifier method on secondary fragment. If \( T_{id} \) is used then

\[ D_{2c} = T^p C S^{zf}/B \quad (3.9) \]

\( D_2 \) is the estimated number of disk accesses for secondary fragment scan.

\[ D_2 = \min \{ D_{2c}, D_{2s} \} \quad (3.10) \]

To summarise above cases of Fragmentation & Access Strategies, it can be stated that in case the selection predicate contains multiple restrict attributes, these attributes may split into two fragments. One containing the scan attribute is the Primary Fragment and other the Secondary Fragment. Choosing the scan strategy of secondary fragment is dependent on which way the primary fragment was scanned first. Primary fragment is scanned by choosing a strategy out of assumed three choices of Cluster Index Scan, Unclustered Index Scan and a Sequential Scan. The strategy giving minimum of disk accesses estimated via estimation formulae given by equations (3.x) is chosen.

### 3.5 Design of Proposed Genetic Algorithm : GA_FA

#### 3.5.1 A General Outline of GA_FA

As discussed in detail in chapter 1, Genetic Algorithms (GA’s) are designed to simulate natural biological evolution process. They work on the Darwin’s theory of survival of the fittest among string structures and propagation of fitter solutions from generation to generation and improving them further in a series of random processes. Fitness is evaluated depending upon the objective function and cost model.
GA’s were first introduced by John Holland and his team at University of Michigan and applied to study robustness of natural systems, which were later used in many diverse areas of artificial systems for function optimization. Goldberg in [Goldberg,1999] highlights a Simple Genetic Algorithm SGA, for a general function’s optimization. This has been the motivation and guiding force behind overall modular design of GA_FA.

Since query optimization problem of distributed databases involves many complicated and interdependent components, most of the exact or heuristic procedures tend to go intractable as soon as problem domain is increased in terms of number of attributes and number of transactions. The proposed algorithm for Fragmentation and evaluation of different access strategies GA_FA is given above in fig.3.1 as an abstract and general genetic algorithm.

Figure 3.1: Genetic Algorithm GA_FA
3.5.2 Representation & Chromosome Encoding

The Genetic chromosome representation of genetic algorithm for Fragmentation and Access Path Selection (GA_FA) is done by a string of integers whose length is equal to the number of attributes of the relation, for example a relation with 10 attributes may be represented like a binary integer string ‘0111011000’, which shall be interpreted as a Binary Fragment Scheme in which one’s position represent attribute number present in first fragment and zero’s represent the attributes in second fragment. Above given example string represents the scenario where attributes 1,5,8,9,10 are in first fragment and the second fragment contains attributes 2,3,4,6,7.

Initial Population Generation is done by first copying given attribute access pattern strings of and transactions and then to complete the population size, ten bits long strings of random binary numbers are generated and added to the generation pool.

3.5.3 Fitness Evaluation & Cost Function Optimization

Cost function to be optimized i.e. minimized and it is the total number of disk accesses incurring while completing a transaction. Cost of an access strategy is evaluated from given transaction profile by using formulae given earlier in cost model by the equations 3.1 to 3.10. Fitness in GA is maximised to generate a new population, so we take fitness function to be reciprocal of the cost function so that the cost function is minimized when fitness function is maximized.

Fragmentation vertically divides the relation in binary and transaction may be completed by accessing a fragment only or by accessing both. To simulate a distributed environment we add a penalty constant value for a transaction that accesses both fragments by a factor of weight 1.15, because different fragments may reside at different sites, and accessing fragments from different sites will involve an extra component of transmission costs. Value 1.15 was found by experimenting empirically and comparing the solutions found by other studies like [March,1983][CU90]. This small value encourages to look for strings which satisfies the transaction from a single fragment instead from an equally fit string which uses two fragments. This penalty value has to be
very small and devised by testing empirically and carefully so that the benefits offered by fragmentation are not outweighed by this penalty factor. If it is kept too large like (Penalty Factor >= 2.0), then output solution will always be a unpartitioned relation.

3.5.4 Genetic Operators & Parameter Settings - Selection, Crossover & Mutation:

Selection may be performed with standard techniques like ‘Roulette Wheel Selection’ or ‘Stochastic Remainder Selection’ and many more [Goldberg,1999]. In this thesis the technique followed is Stochastic Remainder without selection. The chromosome strings are evaluated for their fitness according the cost functions described in cost model given by section 3.2. Fitness is normalized by dividing it with average fitness of the current generation’s population, to which string belongs. The number of child chromosomes produced is proportional to the nearest integer value of normalised fitness. Elitism procedure automatically takes the best fit solution from current generation to the next generation by eliminating the worst fit solution of the next generation.

Crossover method used is simple one point crossover with percentage of crossover rate kept at (70 %) and mutation is performed at a rate of (0.2 %). Crossover rate should be kept high and mutation rate at low as in case of any general genetic search method [Goldberg,1999]. Population size is kept at 50 and maximum number of generations is kept 50. Genetic Algorithm stops when maximum number of generations is reached or there is no improvement in the solution for 15 consecutive generations. Fragmentation is done recursively by first splitting the relation in two fragments and then again splitting both the fragments in binary till access cost decreases are achieved. Due to the penalty factor added to costs when a transaction refers to more than one fragment, the recursive call is rarely more than thrice. It has been observed that most of solutions are found in two or three iterations of the GA.

3.6 Illustration of GA_FA (Using an example Transaction Profile)

3.6.1 An Example set of Transaction: TP1 (acting on a given set of Attributes)

To illustrate the working of the genetic algorithm (GA_FA) we consider an example scenario consisting of a set of transactions T1-T5 on a relation R with cardinality 100,000. Relation consists of ten attributes as shown in a Transaction Profile in table 3.1
taken on the pattern of [Barker et al.2006],[CU90],and [March,1983]. The transaction profile is put as an input file to genetic algorithm implementation (GA_FA). It also contains data about transaction frequencies, length of attributes, restrict attributes and their selectivities. Columns 3 to 9 of table (3.1) represent whether an attribute of the relation is needed by the transaction, which is represented by 1, and an attribute not needed is represented by 0 in these columns. Attribute 1 is taken as a clustering attribute. Length of tuple identifier is assumed to be 2 bytes, prefetch blocking factor of 4 is assumed for a sequential scan and page size is 4k bytes. The size of index page is 10 pages, it is also assumed that indexes are available on attributes 1,2,3 & 6.

<table>
<thead>
<tr>
<th>Transaction</th>
<th>Freq</th>
<th>Attributes Needed by the Transaction</th>
<th>Restrict Attributes</th>
<th>Selectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1 2 3 4 5 6 7 8 9 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>20</td>
<td>1 1 1 0 0 0 0 1 0</td>
<td>1,2,9</td>
<td>.01,.01,.001</td>
</tr>
<tr>
<td>T2</td>
<td>30</td>
<td>1 0 1 1 0 0 0 0 0</td>
<td>1,3,5</td>
<td>.05,0.1,.05</td>
</tr>
<tr>
<td>T3</td>
<td>40</td>
<td>0 0 1 1 1 1 1 0 0</td>
<td>3,6,10</td>
<td>.001,.05,.001</td>
</tr>
<tr>
<td>T4</td>
<td>10</td>
<td>0 0 1 1 0 0 0 0 1</td>
<td>3,9</td>
<td>.005,.001</td>
</tr>
<tr>
<td>T5</td>
<td>10</td>
<td>0 1 1 0 0 0 0 0 0</td>
<td>2,4</td>
<td>.05,.001</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Length of Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 6 8 20 4 4 7 8 6 5</td>
</tr>
</tbody>
</table>

Table 3.1: TP1 Set consisting of Transactions :[T1,T2,T3,T4,T5]

| 1 | 1100000010 | 0.472 | 21200 |
| 2 | 1011000000 | 0.426 | 23501 |
| 3 | 0010011101 | 0.611 | 16355 |
| 4 | 0011000010 | 0.516 | 19379 |
| 5 | 0111011000 | 0.658 | 15200 |
| 6 | 0001100000 | 0.539 | 18545 |
| 7 | 0001010000 | 0.537 | 18631 |
| 8 | 0110100000 | 0.385 | 25985 |
| 9 | 0000010010 | 0.621 | 16100 |
| 10| 0111010000 | 0.612 | 16345 |
| 1 | 0000100010 | 0.688 | 14545 |
| 2 | 0000011001 | 0.881 | 11345 |
| 3 | 0000011011 | 0.683 | 14631 |
| 4 | 1111000100 | 0.860 | 11631 |
| 5 | 1111101010 | 0.881 | 11345 |
| 6 | 0001001101 | 0.715 | 13985 |
| 7 | 0000100111 | 0.894 | 11185 |
| 8 | 0000011000 | 0.781 | 12810 |
| 9 | 0000010010 | 0.826 | 12110 |
| 10| 0000011000 | 0.844 | 11845 |

Table 3.2:Initial Solution Pool for GA_FA ; Table 3.3 Final Solution Pool for GA_FA.
### 3.6.2 Illustration of finding an Access Strategy By GA_FA

Following section elaborates how the access strategies are evaluated and selected. This illustration is based on applying cost model to solution #5 of the ‘initial population solution pool’ as given in table 3.2, for the Transaction Profile TP1 from Table 3.1.

**Transaction 1 : Method 1:** solution chromosome: 0111011000

Let Primary Fragment be: 1,3,8,9,10 ;

Secondary Fragment: 2,3,4,6,7

Case 1.1a) Restrict Attributes (1,2,9) are split into two fragments, so it has to Accesses both Fragments, clustering attribute (1) belongs to primary fragment, so to access primary fragment, clustered index scan is used:

From equation 3.1 of cost model we have

\[ D_{cis} = \frac{\rho_{Ac} C r S_{pf}}{B} \]

Where

- \( D_{cis} \) is the estimated number of disk accesses for clustered index scan
- \( \rho_{Ac} \) is the selectivity of selection predicate on scan attribute (clustering attribute)
  
  \( \rho_{Ac} = 0.01 \);
- \( C_r \) is the relation cardinality = 100000;
- \( S_{pf} \) is the tuple size of primary fragment in bytes including L_id.
- \( B \) Block Size (Page Block Size assumed is 4k bytes)

\[ S_{pf} = \text{Sum of lengths of primary fragment attributes and } L_{id} = 6 + 4 + 8 + 6 + 5 + 2 = 31 \]

\[ S_{sf} = \text{Sum of lengths of primary fragment attributes and } L_{id} = 6 + 8 + 20 + 4 + 7 + 2 = 47 \]

\[ \therefore D_{cis} = \left[ \rho_{Ac} C_r S_{pf} / B \right] = 0.01 * 100000 * 31 / 4000 = 8 \quad (3.11) \]

Case 1.1b) For Unclustered Index Scan of primary fragment:

Scan Attribute is attribute 9 (of least selectivity .001)

\[ D_{uis} = \left[ P_r (1-(1-1/ P_r) T_n + \rho_{uc} N_{si}) \right] = 775(1-(1/775)^{100}) + .001 * 10 \]
\[ T_n = \rho_{Au} C_r = 100000 \times 0.001 = 100 \]
\[ P_F = C_r S^{pf}/B = 100000 \times 31/4000 = 775 \]

Where
- \( D_{uis} \) is the estimated number of disk accesses for unclustered index scan.
- \( T_n \) is the number of tuples satisfying selection condition on unclustered scan attribute Au.
- \( P_F \) is the Page size of Primary Fragment including \( L_{id} = 31 \).
- \( \rho_{Au} \) is the selectivity of selection predicate on a scan attribute which is an unclustered index attribute \((9) = 0.001\).
- \( C_r \) is the relation cardinality = 100000.

\( S^{pf} \) is the tuple size of primary fragment in bytes including Lid (length of tuple identifier Tid) = 47.

\( B \) Block Size (Page Block Size is assumed of 4k bytes).
\( N_{si} \) Number of pages in scan index = 10.

**Case 1.1c): Sequential Scan:**

For a sequential scan number of disk accesses are given by

\[ D_{1ss} = C_r S^{pf}/B F_p = 100000 \times 31 \times 4000 \times 10 = 78 \]  

(3.13)

Where \( F_p = \) Prefetch Blocking Factor

Least of equations 3.11, 3.12 and 3.13 gives the access strategy as clustered index scan for primary fragment with Number of Disk Accesses = 8.  

(3.14)

\[ \therefore \] Access Path Selected for Primary Fragment is : Clustered Index Scan

For secondary Fragment

Case 1.2a) If Tuple identifier is used with clustered Index then

Number of Disk Accesses are given by
\[ D_{2c} = \left[ T_p \ CrS^{sf}/B \right] = .00001 \times 100000 \times 47/4000 = 1 \quad (3.15) \]

\[ T^p = \rho_{as} \prod_{i=2}^{k} \rho_{ai} = 0.01 \times .001 = .00001 \]

Case 1.2b)

But if sequential scan is used for Primary Fragment then

\[ D_{2s} = C_S^{sf}/B \ F_p = 100000 \times 47/4000 \times 10 = 118 \quad (3.16) \]

Where \( F_p \) = Prefetch Blocking Factor.

Finally one can choose access path by deciding on minimum of above two cases for scanning secondary fragment and number of disk accesses are given by

From eq’ns 3.15 and 3.16 we get

\[ D_2 = \text{Min} \{ D_{2c}, D_{2s} \} = 1 \quad (3.17) \]

\[ \therefore \text{Access Path Selected for Secondary Fragment: } T_{id} \text{ by Clustering Index} \]

From Eq’ns 3.14 and 3.17,

Total Number of disk Accesses by Method 1 = 8 + 1 = 9 \quad (3.18) \]

**Transaction 1: Method2:**  solution chromosome: 0111011000

Primary Fragment be: [2, 3, 4, 6, 7]

Secondary Fragment: [1, 5, 8, 9, 10]

Attribute 2 is taken as scan attribute due to it being an unclustering attribute with least selectivity, moreover no clustering attribute is present in the primary fragment.

Now for this case, \( S^{pf} = 31 \) & \( S^{sf} = 47 \)

\[ D_{1us} = P_f (1-(1-1/ P_f)^{T_n} + \rho_{uc} N_{ai} = 1175(1-(1-1/1175)10000) + .001 \times 10 = 674 \]

\[ D_{1ss} = C_S^{pf}/B \ F_p = 100000 \times 47/4000 \times 10 = 118 \quad (3.19) \]

\[ \therefore \text{Access Strategy chosen for Primary Fragment is Sequential Scan (} D_{1ss} \text{).} \]

For secondary Fragment:

\[ D_{2i} = P_f (1-(1-1/ P_f)^{T_n} \quad \& \quad D_{2s} = C_S^{sf}/B \ F_p \]

Where \( T_n = T^p C_r \)
\[ P_f = \frac{C_i S^f}{B} \]

\[ \therefore \quad D_{2i} = 775(1-(1-1/775)^{1000}) = 562 \]

\[ D_{2s} = \frac{C_s S^f}{B} F_p = 100000 * 31/4000 * 10 = 78 \quad (3.20) \]

From above equations we conclude \( D_{2i} < D_{2s} \)

\[ \therefore \quad \text{Access Strategy chosen for Secondary Fragment is Sequential Scan ( } D_{2s} \text{).} \]

Total Number of Disk Accesses by Method 2 = 118 + 78 = 196. \quad (3.21)

From equations 3.18, 3.21 we see method 1 takes lesser number of disk accesses to complete transaction.

**Result:** Choose Method1’s Fragmentation Scheme for Transaction1.

**Transaction 2 : Method1**

Let Primary Fragment be :1,5,8,9,10 & Secondary Fragment: 2,3,4,6,7.

It too accesses both fragments. To access primary fragment, clustering attribute is part of fragment and is the scan attribute.

\[ S^p_f = \text{Sum of lengths of primary fragment attributes and } L_{id} = 6+4+8+6+5+2 = 31 \]

\[ S^s_f = \text{Sum of lengths of primary fragment attributes and } L_{id} = 6+8+20+4+7+2 = 47 \]

\[ D_{1_{cis}} = \left[ \rho Ac \frac{C_i S^p_f}{B} \right] = .05 * 100000 * 31/4000 = 40 \quad (3.22) \]

\[ D_{1_{iss}} = \frac{C_s S^p_f}{B} F_p = 100000 * 31/4000*10 = 78 \]

\[ D_{1_{cis}} < D_{1_{iss}} \]

\[ \therefore \quad \text{Access Path Selected for Primary Fragment is : Clustered Index Scan} \]

For secondary Fragment

\[ D_{2c} = \left[ Tp \ CrS^s_f / B \right] = .0025 * 100000 * 47/4000 = 4 \quad (3.23) \]

\[ T^p = \rho_{as} \prod_{i=2}^{k} \rho_{ai} = 0.05 * .05 = .0025 \]

But if sequential scan is used for Primary Fragment then

\[ D_{2s} = \frac{C_s S^s_f}{B} F_p = 100000 * 47/4000 * 10 = 118 \quad (3.24) \]

\[ D_{2c} < D_{2s} \]
Access Path Selected for Secondary Fragment : \( T_{id} \) by Clustering Index (\( D_{2c} \))

From Eq’ns 3.22 and 3.23 ,

Total Number of disk Accesses by Method 1 = 40 + 4 = 44 \( \text{(3.25)} \)

**Transaction 2 : Method2**

Let Primary Fragment be : 2,3,4,6,7 & Secondary Fragment: 1,5,8,9,10

\[ S_{pf} = 47 \quad \text{&} \quad S_{sf} = 31; \]

To access Primary Fragment unclustering attribute 3 is used

\[
D_{1uis} = P_f (1-(1-1/ P_d)^{T_n} + \rho_{uc} N_{si}) = 1175(1-(1-1/1175)^{10000}) + .1 \times 10 = 1176
\]

\[
D_{1ss} = C_r S_{pf} / B F_p = 100000 \times 47 / 4000 \times 10 = 118 \quad \text{(3.26)}
\]

∴ Access Strategy chosen for Primary Fragment is Sequential Scan (\( D_{1ss} \)).

For secondary Fragment:

\[
D_{2i} = P_f (1-(1-1/ P_d)^{T_n} \quad \text{&} \quad D_{2s} = C_r S_{sf} / B F_p
\]

Where \( T_n = T^\circ C_r \)

\[
P_f = C_r S_{sf} / B
\]

∴ \( D_{2i} = 775(1-(1-1/775)^{10000}) = 775 \)

\[
D_{2s} = C_r S_{sf} / B F_p = 100000 \times 31 / 4000 \times 10 = 78 \quad \text{(3.27)}
\]

From above equations we conclude \( D_{2i} < D_{2s} \)

∴ Access Strategy chosen for Secondary Fragment is Sequential Scan (\( D_{2s} \)).

Total Number of Disk Accesses by Method 2 = 118 + 78 = 196. \( \text{(3.28)} \)

From equations 3.25 & 3.28 we see method 1 takes lesser number of disk accesses to complete transaction.

**Result:** Choose Method1’s Fragmentation Scheme for Transaction2.

**Transaction 3 : Method1**

Let Primary Fragment be : [ 2,3,4,6,7 ] & Secondary Fragment: [1,5,8,9,10]
It too accesses both fragments. To access Primary Fragment, clustering attribute is not part of restrict attribute of fragment.

\[SP^f = \text{Sum of lengths of primary fragment attributes and } L_{id} = 6+8+20+4+7+2 = 47\]

\[S^s = \text{Sum of lengths of secondary fragment attributes and } L_{id} = 6+4+8+6+5+2 = 31\]

To access Primary Fragment unclustering attribute 3 is used

\[D_{\text{uis}} = P_f (1-(1-1/ P_f)T_n + \rho_{ui} N_{si} = 1175(1-(1-1/1175)^{100}) + .001*10 = 96\]

(3.29)

\[D_{\text{iss}} = C_r S^f / B F_p = 100000 * 47 / 4000*10 = 118\]

\[D_{\text{uis}} < D_{\text{iss}}\]

∴ Access Path Selected for Primary Fragment is: Unclustered Index Scan

For secondary Fragment

\[D_{2i} = P_f (1-(1-1/ P_f)T_n\]

Where

\[T^p = \rho_{as} \Pi_{i=2}^c \rho_{ai} = 0.05 * .001 = .00005\]

\[T_n = T^p C_f = .0005 * 100000 = 5\]

\[P_f = C_r S^s / B =100000 * 31 /4000 = 775\]

\[D_{2i} = 775(1-(1-1/775)^5) = 5\]

(3.30)

\[D_{2s} = C_r S^s / B F_p = 100000 * 31/ 4000 * 10 = 78\]

\[D_{2i} < D_{2s}\]

∴ Access Path Selected for Secondary Fragment: \(T_{ad}\) by Unclustering Index (\(D_{2i}\))

From Eq’ns 3.29 and 3.30,

Total Number of disk Accesses by Method 1 = 96 + 5 = 101

(3.31)

**Transaction 3 : Method2**

Let Primary Fragment be: [1,5,8,9,10] & Secondary Fragment: [2,3,4,6,7]
\[ S^{pf} = 31 \quad \& \quad S^{sf} = 47; \]

To access Primary Fragment nonindexed attribute 10 is used

\[ D_{1ss} = \frac{C_s S^{pf} / B F_p}{100000 * 31/ 4000 * 10} = 78 \quad (3.32) \]

\( \therefore \) Access Strategy chosen for Primary Fragment is Sequential Scan (D_{1ss}).

For secondary Fragment:

\[ D_{2i} = P_f (1-(1-1/ P_f)^T_n) \quad \& \quad D_{2s} = C_s S^{sf} / B F_p \]

Where \( T_n = T^p C_r \)

\[ P_f = \frac{C_s S^{sf} / B}{1175 (1-(1/1175))^{100}} = 96 \]

\[ D_{2s} = \frac{C_s S^{sf} / B F_p}{100000 * 47/ 4000 * 10} = 118 \quad (3.33) \]

From above equations we conclude \( D_{2i} < D_{2s} \)

\( \therefore \) Access Path Selected for Secondary Fragment : T_id by Unclustering Index (D_{2i}).

Total Number of Disk Accesses by Method 2 = 118 + 96 = 214. \( (3.34) \)

From equations 3.31 & 3.34 we see method 1 takes lesser number of disk accesses to complete transaction.

**Result:** Choose Method1’s Fragmentation Scheme for Transaction3.

**Transaction 4 : Method1**

Let Primary Fragment be : [2,3,4,6,7] \quad \& \quad Secondary Fragment: [1,5,8,9,10]

It too accesses both fragments. To access Primary Fragment, clustering attribute is not part of restrict attribute of fragment.

\[ S^{pf} = \text{Sum of lengths of primary fragment attributes and } L_{id} = 6+8+20+4+7+2 = 47 \]

\[ S^{sf} = \text{Sum of lengths of secondary fragment attributes and } L_{id} = 6+4+8+6+5+2 = 31 \]

To access Primary Fragment unclustering attribute 3 is used
\[ D_{\text{luis}} = \text{Pr} \left( 1 - \frac{1}{\text{Pr}^n} \right) + \rho_{\text{as}} \text{N}_{\text{si}} = 1175 \left( 1 - \frac{1}{1175} \right)^{500} + 0.005 \times 10 = 408 \]
\[ D_{\text{ss}} = \text{C}_s \text{S}^f / B \text{ F}_\text{p} = 100000 \times \frac{47}{4000} = 118 \quad (3.35) \]

\[ D_{\text{ss}} < D_{\text{luis}} \]

\[ \therefore \text{Access Path Selected for Primary Fragment is: Sequential Scan} \]

For secondary Fragment

\[ D_{2i} = \text{Pr} \left( 1 - \frac{1}{\text{Pr}^n} \right) \]

Where
\[
T^p = \rho_{\text{as}} \prod_{i=2}^{k} \rho_{ai} = 0.005
\]
\[ T_n = T^p \text{ C}_r = 0.005 \times 100000 = 500 \]
\[ \text{Pr} = \text{C}_s \text{S}^f / B = 100000 \times \frac{31}{4000} = 775 \]

\[ D_{2i} = 775 \left( 1 - \frac{1}{1175} \right)^{500} = 369 \]

\[ D_{2s} = \text{C}_s \text{S}^f / B \text{ F}_\text{p} = 100000 \times \frac{31}{4000} = 78 \quad (3.36) \]

\[ D_{2s} < D_{2i} \]

\[ \therefore \text{Access Path Selected for Secondary Fragment: Sequential Scan} \]

From Eq’ns 3.35 and 3.36,

Total Number of disk Accesses by Method 1 = 118 + 78 = 196 \quad (3.37) \]

**Transaction 4 : Method2**

Primary Fragment be: [1,5,8,9,10] \& Secondary Fragment: [2,3,4,6,7]

\[ S^f = 31 \quad \& \quad S^{sf} = 47; \]

To access Primary Fragment nonindexed attribute 9 is used

\[ D_{\text{ss}} = \text{C}_s \text{S}^f / B \text{ F}_\text{p} = 100000 \times \frac{31}{4000} = 78 \quad (3.38) \]

\[ \therefore \text{Access Strategy chosen for Primary Fragment is Sequential Scan (} D_{\text{ss}} \).

69
For secondary Fragment:

\[ D_{2i} = P_f \left( 1 - \left( \frac{1}{P_f} \right)^{T_n} \right) \quad \& \quad D_{2s} = C_s S^{sf} / B F_p \]

Where

\[ T_n = T^P C_r \]

\[ P_f = C_s S^{sf} / B \]

\[ \therefore D_{2i} = 1175 (1 - (1 - 1/1175)^{100}) = 96 \quad (3.39) \]

\[ D_{2s} = C_s S^{sf} / B F_p = 100000 \times 47 / 4000 \times 10 = 118 \]

From above equations we conclude \( D_{2i} < D_{2s} \)

\[ \therefore \text{Access Path Selected for Secondary Fragment} \]

\[ i.e., \text{T_id by Unclustering Index (D}_{2i} \]

Total Number of Disk Accesses by Method 2 = 78 + 96 = 174. \quad (3.40)

From equations 3.37 & 3.40 we see method 2 takes lesser number of disk accesses to complete transaction.

**Result:** Choose Method2’s Fragmentation Scheme for Transaction4.

**Transaction 5 : Note:** It accesses only one fragment

Primary Fragment : [2,3,4,6,7] ; \( S^{pf} = 47; \)

To access Primary Fragment unclustering attribute 4 is used

\[ D_{1uis} = P_f \left( 1 - \left( \frac{1}{P_f} \right)^{T_n} + \rho_{uc} N_{ui} \right) = 1175 (1 - (1 - 1/1175)^{100}) + .001 \times 10 = 96 \quad (3.41) \]

\[ D_{1ss} = C_s S^{pf} / B F_p = 100000 \times 47 / 4000 \times 10 = 118 \quad (3.42) \]

\[ \therefore \text{Access Path Selected for Primary Fragment : Unclustering Index Scan (D}_{1uis} \]

Total Number of Disk Accesses = 96. \quad (3.43)
### Summary of Results for Solution # 5 of Set of Transactions TP1 (from GA_FA Run)

<table>
<thead>
<tr>
<th>Transaction</th>
<th>Access Strategy For</th>
<th>Number of Disk Accesses</th>
<th>Frequency</th>
<th>Total_No_of Disk_Access</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Primary_Frag</td>
<td>Second_Frag</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Clustered_Scan</td>
<td>Clustered T_id</td>
<td>09</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>Clustered_Scan</td>
<td>Clustered T_id</td>
<td>44</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>Unclustered_scan</td>
<td>Un_Clustered T_id</td>
<td>101</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>Sequential Scan</td>
<td>Un_Clustered T_id</td>
<td>174</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>Unclustered Scan</td>
<td>Not Present</td>
<td>96</td>
<td>10</td>
</tr>
</tbody>
</table>

Total Number of Disk Accesses for solution # 5: 9332

Table - 3.4: GA_FA Run for TP1 and details of Access Strategies for a Solution

### 3.7 Experimental Methodology & Analysis of Results

Experiments were conducted after coding the Genetic Algorithm for fragmentation and access strategies (GA_FA) in PASCAL programming language. It was run on an Intel® core2™ 6420 @ 2.13 GHz machine, with 1.00 GB RAM, running on a WINDOWS-XP platform. Experiments were conducted by varying number of attributes and the number of transactions on the given relations in sets of ten transaction profiles TP1 to TP10. Set of experimental Transaction profiles vary from small to medium and very large problems.

Small problems consisted of 10 attributes with transactions running on them up to the tune of five e.g. TP1, taken & illustrated in full details in section 3.4.1. Large problems similar to TP6 [March,1983] etc, widely used in distributed database research field were taken, which contained more than 15 transactions acting on more than 30 attributes. One such example transaction profile is given below in table 3.
One such set of experiments was conducted on TP$_6$, by calculating the number of disk accesses incurred to run all the 18 transactions on a set of 30 attributes, by an unpartitioned scheme. Then same set of transactions are used to compare GA_FA with March’s Scheme, Cornell and Yu’s scheme and Barker & Jun’s methodology, on the basis of disk access costs. The number of disk accesses and hence query costs showed a reduction of 61 percent as compared to unpartitioned one, and disk access costs decreased to the 33 percent to that of deterministic techniques of cornell & Yu [CU90], and by 23% of the genetic solution by March [March,1983].GA_FA’s disk access cost showed a 10% decrease in access costs as compared to Barker & Jun’s GA approach [BJ2006], as highlighted by output solution data and graphical analysis in fig3.2 below, for the set of transactions given by TP$_6$ Transaction profile.

Table 3.5: A Transaction profile for a Large Problem TP$_6$
GA_FA was able to find optimal solution in all the cases for transaction profiles, ranging from small group of attributes and transactions like TP1 to very complex ones like TP8. Ranging from medium sized Transaction Profiles like TP5 to very complex ones like TP8, Genetic Algorithm was faster to find a reasonably optimal solution by 70% of time taken to find a solution by exhaustive methods.
3.8 Summary

Chapter 3 started with a general introduction to the fragmentation process, its types and advantages. Next sections introduced Access Strategies and described a formulation of the cost model and an objective function. It also described the cost estimation formulation of the three major local disk access path strategies: Clustered Index Scan, Unclustered Scan and Sequential Scan. Then a general outline of the genetic algorithm for fragmentation and access strategies was given along with detailed description of genetic parameter settings and operator’s (Selection, Crossover, Mutation) design. At fag end it illustrated functioning of GA_FA’s access cost methods, by taking an example transaction profile. Finally it discussed experimental results and analysis.