Clustering Related Tuples in Databases

D. A. BELL*, F. J. MCERLEAN, P. M. STEWART AND W. ARBUCKLE

Department of Information Systems, University of Ulster at Jordanstown, Shore Road, Newtownabbey, Co. Antrim BT37 0QB, Northern Ireland

This paper presents empirical results obtained using a heuristic graph-collapsing technique for placing data on storage devices. Previous experiments using this technique worked on small datasets. We have set out to gain further experience with the graph-collapsing algorithm using more realistic data sizes. These experiments have given us promising results and stimulated ideas which should be beneficial to the database designer in the future.

Received November 1986, revised December 1986

1. INTRODUCTION

Physical record clustering is concerned with the storing in close proximity of tuples on storage devices which are referenced frequently together, thereby bringing advantages in reduced operation times on a database system. Optimising the proximity of tuples on a storage device is clearly seen to be a worthwhile procedure where performance is considered.

Early work in this area was done by Flory et al., who used an inverted-file technique in a two-phase placement method. Records were first grouped into clusters in a manner which kept the numbers of inter-cluster accesses needed to service queries to a minimum. Then each cluster was allocated to storage in a way which minimised the length of inter-cluster accesses. Jakobsson1 also did some work on clustering attribute values for inverted files and used it in an evolutionary file re-organisation scheme. Schkolnick devised a clustering algorithm for the clustering of hierarchically related record types rather than that of record occurrences, with significant reduction in the expected number of page faults for data retrieval. Hoffer has presented a good survey of many of the data-clustering methods which have appeared in the literature.

The present study evaluates a proposed solution to the general problem of efficiently clustering tuples which may be of a variety of types and may be related in a variety of ways. In general there can be a many-to-many relationship between the tuple types. It is intended to be part of a database design tool similar to EOS. The method discussed here originated from a technique developed by Malmquist et al. to cluster related tuples on storage devices. This problem was generalised to become the Layered Data Placement problem (LDP), which is NP-complete, i.e. belonging to a class of problems which have resisted persistent efforts of experts at solving them. Therefore heuristic approaches rather than deterministic approaches are appropriate for solving the LDP problem.

Some improvements were made to Malmquist's algorithm in the development of an algorithm which will be called here V1, after which it was suggested that a new placement algorithm should be developed which has two levels of storage hierarchy: pages and cylinders. A page was the only level used in V1.

In the present study this two-level algorithm, called here V2, raises the probability of pages which contain related tuples being on the same cylinder (or some other desirable area which is larger than a page). Super partitioning in this way means that if the pages contain related tuples the probability of their being on the same cylinder is increased. Therefore a buffering policy may be devised to advantageously collect many related tuples at a single read–write head setting.

In the next section we describe the development of algorithms V1 and V2 from Malmquist's algorithm and previous experiments carried out on them. Section 3 presents the results of new experiments carried out on the algorithms using more realistic and more extensive simulated data, and Section 4 looks ahead to further analysis of the algorithms and new applications to which they could be applied.

2. COMPARISON OF DEVELOPED ALGORITHMS

Malmquist used a graph to represent relationships between tuples in a Coadysyl database, the nodes of the graph representing pages on a storage device, and edges between the nodes representing relationships (links) between tuples in the pages. Initially one tuple is assigned per page. The weights (or values) assigned to edges are formed when a directed graph is transformed into an undirected graph.

The V1 algorithm, which was developed for the same purpose, has overcome some of the limitations of Malmquist's algorithm. This means that V1 is not restricted to a special application of the Coadysyl network model, and 'importance values' of accesses are now considered, as we discuss below. The next stage was the development of V2, which deals with both pages and cylinders concurrently rather than sequentially. This produces a higher probability for pages to be close to one another when this is desirable; pages containing related tuples are more likely to be placed in the same cylinder or other larger area of secondary storage when V2 is executed.

Before applying V1 or V2, the load for edges on an undirected graph similar to Malmquist's is considered. The weight of an edge between a pair of nodes is determined from the priority of applications accessing the tuples and the access frequency and responsiveness for transactions traversing it. Fig. 1 shows an example of an undirected graph. Pages are collapsed or clustered in order of priority of edge weights. In the V2 algorithm, when two pages cannot be collapsed the collapsing of their cylinders is attempted.

Collapsing two nodes involves breaking the edge

* To whom correspondence should be addressed.
between them and moving the contents of the higher-numbered node on to the lower-numbered node. Common nodes also need to be considered. A common node is connected directly to both nodes on the current edge, suggesting that when that edge is deleted the weights of the edges must be concatenated. We do this simply by adding. It is beyond the scope of this study to see if addition is the most efficient action to take when concatenation of edges occurs. Whether or not there is a common node, edges which were previously connected to the higher-numbered node are now updated so that they become connected to the lower-numbered node. An example of the collapsing procedure is shown in Fig. 2.

Bell has carried out several initial experiments to compare the performance of V1 and V2. Using a simulation in the range 1–4% and a 10% hit-group for his experiments it was noticed that using V2 gave an improvement of up to 39% over V1. (N.B. A hit-group is a set of tuple pairs which account for high-access traffic, and therefore their edges are given a high weighting.)

An important aspect to be considered when using a clustering algorithm is the execution time of the algorithm. Conventional clustering algorithms execute in $O(n^2)$ time [$n = \text{no. of nodes}$] in comparison to $O(n)$ time for the three algorithms mentioned in this paper.

![Figure 1. Initial weighted undirected graph. $e_{ij}$ edge with highest priority; $p$, page; $t$, tuple.](image1)

![Figure 2. Collapsing a graph. (a) Before; (b) after. $p_1 \equiv p_3$ and $p_3$ is the common node for $p_1$ and $p_4$.](image2)
3. EXPERIMENTATION

Several experiments were carried out to compare the performance of V2 with V1. However, up to now only small datasets have been used for the experiments and simulations. Most of our experiments were done using larger datasets, to obtain results for more realistic scenarios.

The datasets used in our experiments represented relationships between two relations called R1 and R2 (modelling Products and Parts entities respectively). The cardinalities of R1 and R2 were in the ratio of 1:3, which was kept constant during all our experiments, but the effect of altering the fanout between relations was investigated. (N.B. Fanout is taken here to be the maximum number of product tuples associated with a part tuple.) The simulation parameters for page and cylinder sizes were based on manufacturers' figures.

The benchmark used for all of the experiments described below, and those of Bell, was to obtain the parts associated with a particular product from the database. The main objective in all cases was to obtain insights into how a single performance index, average response time, varied when the different algorithms were used with a variety of parameters.

(a) Preliminary experimentation with variation of fanout and page size

Initially we carried out some experiments with V1 and V2, in exactly the form that Bell had used in his experiments, using a 10% simulation to see what effect variation in fanout and page size would have on average retrieval times. For fanouts 2–10 the page size ranged from 240 to 2000, and it was noticed that the advantage of V2 over V1 was best at lower fanouts. Both algorithms made an improvement in performance as page size increased. These preliminary experiments used a 3% hit-group whose edges were given a weight of 2, the remaining edges having a weight of 1.

The above experiments were repeated with a 33% hit-group with edges of weight 5 to ascertain the effect on the algorithm of a large hit-group of highly weighted edges. The results achieved were similar to those obtained for the smaller hit-groups.

In the experiments all parts (S nodes) associated with a particular product (F node) were retrieved, and we decided that it would be interesting to see what the effect of reversing the direction of retrieval would be; i.e. all F nodes associated with an S node. After performing an experiment to see what effect this would have, we noticed that retrieval times increased by 29% for both algorithms, leaving the percentage cost differential fixed.

Next the Parts–Products database was partitioned into five equal hit-groups, the edges being given weights 1–5. The purpose of this experiment was to discover the effect of using the algorithms in a more complex situation. The results of these experiments confirm Bell's corresponding results at low fanout, although the times vary greatly with fanout, as discussed below.

(b) Random assignment of tuples to pages and pages to cylinders

The main experiments reported here were performed to show how useful these clustering algorithms are for database systems. To demonstrate their advantages an algorithm (RAN) was developed to randomly assign tuples to pages and pages to cylinders. The experiments compared algorithms V1, V2 and RAN over a fanout range 2–75 and page sizes from 500 to 2000. Algorithm V1 was actually modified slightly for these experiments. In Bell's experiments the pages left after the clustering process of V1 were packed sequentially into cylinders. This clearly gave V1 an advantage, because the sequential pattern of collapsing nodes combined with the sequential loading of pages on to cylinders caused the V1 algorithm to capitalise on the existing order of the input data. In fact at high fanouts the results we were getting in preliminary experiments occasionally showed V1 as being superior to V2. (This suggests a possible direction for improvement of V1.) The original V1 algorithm was modified, to become V1', so that pages were randomly loaded on to cylinders after the collapsing of tuples on to pages had been completed.

Results of V1' vs V2 vs RAN, which are shown in Table 1, are displayed in Fig. 3. This graph indicates that both V1' and V2 are clearly beneficial to use for the database systems simulated. It is also clear that V2 outperforms V1' consistently, although the percentage advantage is decreased substantially as the fanout is increased to 10 (see Fig. 4). However, further experimentation for fanouts above 75 showed an increase in percentage improvement above that for fanout 10, therefore indicating that percentage improvement does not continuously decrease as fanout increases. We are
Table 1

<table>
<thead>
<tr>
<th>Page size</th>
<th>V1'</th>
<th>V2</th>
<th>RAN</th>
<th>V2 vs V1'</th>
<th>V1' vs RAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>29777</td>
<td>22608</td>
<td>75135</td>
<td>( x = 24.8%; S = 3.12; \text{SE} = 1.80 )</td>
<td>( x = 60.37%; S = 0.58; \text{SE} = 0.34 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>95% CI = 20.55–26.61</td>
<td>95% CI = 59.70–61.04</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( x = 17.5%; S = 2.75; \text{SE} = 1.59 )</td>
<td>( x = 71.31%; S = 0.63; \text{SE} = 0.36 )</td>
</tr>
<tr>
<td>1000</td>
<td>21257</td>
<td>17357</td>
<td>74088</td>
<td>95% CI = 14.38–20.62</td>
<td>95% CI = 70.60–72.02</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( x = 14.83%; S = 2.46; \text{SE} = 1.41 )</td>
<td>( x = 74.90%; S = 0.53; \text{SE} = 0.31 )</td>
</tr>
<tr>
<td>1500</td>
<td>18532</td>
<td>15783</td>
<td>73841</td>
<td>95% CI = 12.05–17.61</td>
<td>95% CI = 74.29–75.51</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( x = 12.20%; S = 4.04; \text{SE} = 2.33 )</td>
<td>( x = 77.34%; S = 0.31; \text{SE} = 0.18 )</td>
</tr>
<tr>
<td>2000</td>
<td>16638</td>
<td>14609</td>
<td>73413</td>
<td>95% CI = 7.63–16.77</td>
<td>95% CI = 76.99–77.69</td>
</tr>
</tbody>
</table>

FANOUT = 4

\[ x = 11.92\%; S = 1.26; \text{SE} = 0.73 \]
95\% CI = 10.49–13.35
95\% CI = 32.83–34.87

\[ x = 9.26\%; S = 1.75; \text{SE} = 1.01 \]
95\% CI = 7.82–11.24
95\% CI = 40.05–40.33

\[ x = 8.46\%; S = 1.87; \text{SE} = 1.08 \]
95\% CI = 6.34–10.58
95\% CI = 41.69–42.31

\[ x = 10.23; S = 1.36; \text{SE} = 0.78 \]
95\% CI = 8.70–11.76
95\% CI = 42.25–43.31

FANOUT = 6

\[ x = 8.04\%; S = 0.45; \text{SE} = 0.26 \]
95\% CI = 7.53–8.55
95\% CI = 22.73–24.09

\[ x = 7.84\%; S = 0.34; \text{SE} = 0.19 \]
95\% CI = 7.46–8.21
95\% CI = 27.04–27.94

\[ x = 7.02\%; S = 0.20; \text{SE} = 0.12 \]
95\% CI = 6.79–7.26
95\% CI = 28.21–29.53

\[ x = 8.30\%; S = 0.67; \text{SE} = 0.39 \]
95\% CI = 7.54–9.06
95\% CI = 28.42–29.86

FANOUT = 8

\[ x = 4.57\%; S = 0.26; \text{SE} = 0.15 \]
95\% CI = 4.28–4.86
95\% CI = 17.36–18.58

\[ x = 5.07\%; S = 0.69; \text{SE} = 0.40 \]
95\% CI = 4.29–5.85
95\% CI = 21.53–21.79

\[ x = 5.38\%; S = 0.66; \text{SE} = 0.38 \]
95\% CI = 4.64–6.12
95\% CI = 21.60–22.78

\[ x = 5.92\%; S = 0.63; \text{SE} = 0.36 \]
95\% CI = 5.21–6.63
95\% CI = 22.16–23.58

FANOUT = 10

\[ x = 2.64\%; S = 0.54; \text{SE} = 0.31 \]
95\% CI = 2.03–3.25
95\% CI = 15.46–15.86

\[ x = 3.24\%; S = 0.14; \text{SE} = 0.08 \]
95\% CI = 3.08–3.40
95\% CI = 17.86–18.26

\[ x = 2.96\%; S = 0.34; \text{SE} = 0.19 \]
95\% CI = 2.59–3.33
95\% CI = 18.43–18.97

\[ x = 4.60\%; S = 0.05; \text{SE} = 0.03 \]
95\% CI = 4.54–4.66
95\% CI = 18.93–19.23

x = Mean percentage improvement; S = Standard deviation; SE = Standard error of mean

Currently investigating the reason(s) for this phenomenon. An important result of this work is therefore that the excellent improvements reported by Bell are not steady for high fanouts. This implies that the choice of V2 for all practical design situations is not automatic.

4. FUTURE WORK

Up to now a two-level partitioning has been considered for the solution of the LDP problem. Our experiments have shown the algorithm developed for this to be satisfactory. We intend to develop a multi-level algorithm to take into consideration such storage units as disc packs and distributed databases.

So far we have dealt with two relations in our algorithms. It is important to extend the algorithms so as to deal with many relations and various patterns of inter-relationship linking which exist in real databases. An
of a physical database design and reorganisation system which we will develop. As well as improving the graph-collapsing algorithm we will be developing and modifying other heuristic clustering techniques. Two methods of interest to us are the Metropolis algorithm and the solution to the multidimensional zero-one knapsack problem\textsuperscript{10,11}

5. SUMMARY AND CONCLUSION

This paper describes work on improving and further understanding physical record clustering, making use of a graph-collapsing technique developed previously.

Experiments were performed using datasets representing relationships between a relation pair, to compare the efficiency of the two methods, \( V1' \) for a single-level partitioning algorithm and \( V2 \) for a two-level partition. A number of experiments were carried out using a variation of page size and fanout to observe when \( V2 \) would outperform \( V1' \). The difference in retrieval times between \( V2 \) and \( V1' \) as fanout increased remained constant, and therefore the percentage improvement of \( V2 \) over \( V1' \) decreased as fanout increased, for the fanout range considered here (at higher fanouts the improvement was more dramatic).

Summarising our results we can say that, although the original \( V1 \) performed better than \( V2 \) with some particularly suitable data distributions, it was found that using realistic simulations \( V2 \) consistently and often significantly outperformed \( V1' \). The results of these experiments have given us many ideas for the future.

In the area of physical record clustering there is a lot of scope for improvement of clustering techniques. The graph-collapsing algorithm described here is a useful tool for optimising placement of interrelated tuples.

We were encouraged by the results of our experiments to focus our attention upon the two-level algorithm. In general we believe that the graph-collapsing techniques have promising prospects for the future and will benefit the database designer. But at the same time we shall not ignore other possible solutions to this interesting problem.

Acknowledgements

This work was carried out with the financial support of SERC.

REFERENCES


THE COMPUTER JOURNAL, VOL. 31, NO. 3, 1988 257