Processing Relational OLAP Queries with UB-Trees and Multidimensional Hierarchical Clustering

Volker Markl
Bayerisches Forschungszentrum
für Wissensbasierte Systeme
(FORWISS)
Orleansstraße 34, D-81667,
München, Germany
volker.markl@forwiss.de

Rudolf Bayer
Institut für Informatik
Technische Universität München
Orleanstraße 34, D-81667
München, Germany
bayer@in.tum.de

Abstract

Multidimensional access methods like the UB-Tree can be used to accelerate almost any query processing operation, if proper query processing algorithms are used: Relational queries or SQL queries consist of restrictions, projections, ordering, grouping and aggregation, and join operations. In the presence of multidimensional restrictions or sorting, multidimensional range query or Tetris algorithms efficiently process these operations. In addition, these algorithms also efficiently support queries that generate some hierarchical restrictions (for instance by following 1:n foreign key relationships). In this paper we investigate the impacts on query processing in RDBMS when using UB-Trees and multidimensional hierarchical clustering for physical data organization. We illustrate the benefits by performance measurements of queries for a star schema from a real world application of a SAP business information warehouse. The performance results reported in this paper were measured with our prototype implementation of UB-Trees on top of Oracle 8. We compare the performance of UB-Trees to native query processing techniques of Oracle, namely access via an index organized table, which essentially stores a relation in a clustered B*-Tree, and access via a full table scan of an entire relation. In addition we measure the performance of the intersection of multiple bitmap indexes to answer multidimensional range queries.

Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'2000)
Stockholm, Sweden, June 5-6, 2000
(M. Jeusfeld, H. Shu, M. Staudt, G. Vossen, eds.)
http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-28/

1 Introduction

The most established relational data models for data warehousing applications are the star schema and the snowflake schema. In both approaches there is a central fact table that contains the measures and the dimension tables are situated around it. The connection between a fact tuple and the corresponding dimension members is realized via foreign key relationships. In the star schema the dimension tables are completely denormalized while in the snowflake schema they may be normalized. Queries usually contain restrictions on multiple dimension tables (e.g., only sales for specific customer group and for a specific time period are asked) that are then used as restrictions on the usually very large fact table.

In this paper we investigate, how UB-Trees and the Tetris algorithm in combination with our technique of multidimensional hierarchical clustering by hierarchy interleaving (MHC/HI) may be used to accelerate relational query processing with a special focus on star-joins, the most frequent operation of query processing for relational data warehouses. With our technique of MHC/HI we specifically cluster the data with respect to the foreign key relationships defined by the star- or snowflake schema of a data warehouse. We also illustrate the physical data modeling with MHC/HI for a SAP business information warehouse and give a performance analysis for this real-world application. Our performance analysis gives insight into the real world data distribution of 6 GB of sales data from a fruit juice company and shows, why MHC/HI in combination with UB-Trees is superior to classical bitmap indexes, clustering B*-Trees and parallel full table scans.

Our paper is organized as follows: Section 2 surveys related work, in Section 3 we present the basic concepts of UB-Trees, their algorithms and MHC/HI. Section 4 presents the SAP business warehouse schema that we use for our analysis. In Section 5 we analyze the data distribution of the real world data. Section 6 gives a performance analysis and performance measurements with our prototype implementation. Section 7 concludes our paper and gives an outlook on future work.

2 Related Work

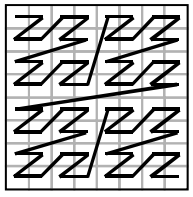
Performance optimization has been well studied in the field of OLTP systems [Gra93]. Due to the completely different query characteristics of OLAP applications in comparison to OLTP new questions have to be addressed here. The performance problem is heavily linked to the physical data model.

The index selection problem for ROLAP application is widely discussed in the research community [GHR+97, Sar97]. Especially bitmap indexes have been proposed to speed up ROLAP applications because of their compactness and support of star joins [CI98]. A common way of performance improvement is the usage of materialized views - often in combination with indexing methods [TS97, Moe98, WB98]. Due to the large number of possible views a selection problem exists besides the maintenance issue [Gup97, SDN+96, SDN+98]. Clustering of OLAP data plays a key role in providing good performance. Clustering has been well researched in the field of access methods. B-Trees, for instance, provide one-dimensional clustering. Multidimensional clustering has been discussed in the field of multidimensional access methods. See [GG97] and [Sam90] for excellent surveys of almost all of these methods. [ZSL98] addresses the issue of hierarchical clustering for the one-dimensional case.

Most work on applying multidimensional indexes to RDBMS discusses restrictions by range queries [SRF97, NHS84, Gut84, OM84, LS90]. [JL98] accelerates range queries with aggregations by storing aggregated data in R-Trees. [MRB99] and [Bay97b] are the basis of our approach, where joins and sorted processing of data organized by a multidimensional index are investigated.

3 The UB-Tree

The basic idea of the UB-Tree [Bay97a, Mar99] is to use a space-filling curve to partition a multidimensional universe. Using the Z-Curve (Figure 1a) the UB-Tree preserves the multidimensional clustering. A *Z-Address* $\alpha = Z(x)$ is the ordinal number of the key attributes of a tuple x on the Z-Curve, which can be efficiently computed by bit-interleaving [OM84]. A standard B-Tree is used to index the tuples taking the linear Z-Address of the tuples as keys.

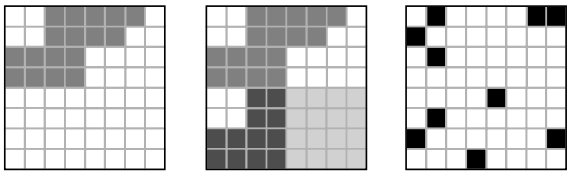


	0	1	2	3	4	5	6	7
0	0	1	4	5	16	17	20	21
1	2	3	6	7	18	19	22	23
2	8	9	12	13	24	25	28	29
3	10	11	14	15	26	27	30	31
4	32	33	36	37	48	49	52	53
5	34	35	38	39	50	51	54	55
6	40	41	44	45	56	57	60	61
7	42	43	46	47	58	59	62	63

(a) (b)

Figure 1: Z-curve and Z-addresses

The fundamental innovation of UB-Trees is the concept of *Z-Regions* to create a disjunctive partitioning of the multidimensional space. This allows for very efficient processing of multidimensional range queries [Mar99]. A Z-Region $[\alpha : \beta]$ is the space covered by an interval on the Z-Curve and is defined by two Z-Addresses α and β. We call β the *region address* of $[\alpha : \beta]$.Each Z-Region maps exactly onto one page on secondary storage, i.e., to one leaf page of the B-Tree.



(a) (b) (c)

Figure 2: Z-regions

For an 8×8 universe, i.e., $s = 3$ and $d = 2$, Figure1b shows the corresponding Z-addresses. Figure2a shows the Z-region [4: 20] and Figure2b shows a partitioning with five Z-regions [0 : 3], [4 : 20], [21: 35], [36 : 47] and [48 : 63]. Assuming a page capacity of 2 points, Figure2c shows ten points, which create the partitioning of Figure2b. The details of the UB-Tree algorithms are described in [Bay97a, Mar99].

3.1 Bit Interleaving

The performance of the UB-Tree crucially relies on an efficient implementation of the Z-address calculation. For tuples of positive integer numbers, bit interleaving immediately yields an algorithm to calculate a Z-address from the binary representation of a tuple.

It is easy to incorporate varying attribute lengths, i.e., attributes with different resolutions, into this algorithm: When the limit of the resolution (i.e., the last bit) in one attribute is reached, this attribute is not used for bit-interleaving any further. The number of bits in each further step is reduced in this case.

For each attribute A_i we denote the number of distinct values of its domain by r_i, i.e., $r_i = |A_i|$

Definition 1: The *number of steps for attribute* A_i of a domain with cardinality[1] r_i is determined by its resolution: $\text{steps}(i) = \log_2 r_i$

Definition 2: The *length of step k* in bits (i.e., the number of dimensions in step k) is:

$$\text{steplength}(k) = |i \mid \text{steps}(i) \geq k \text{ and } i \in D\}|$$

Figure 3 shows this generalized algorithm for bit interleaving.

```
Input:  x : tuple
Output: Z-address α

for step = 1 to max({steps(r_j) | j ∈ D})
   for i = 1 to steplength(step)
      copy bit step of x_i to bit i of α_step
   end for
end for
```

Figure 3: Bit-Interleaving to calculate $\alpha = Z(x)$

With $r = \max(\{\text{steps}(r_j) \mid j \in D\})$ bit interleaving has a CPU-complexity of $O(d \cdot r)$ bit operations (resp. $O\left(\sum_{i=1}^{d} r_i\right)$

[1] Note that we defined r_i to be 2^v for some $v \in \mathbb{N}_0$

for attributes of different length). The same holds for the inverse algorithm Z^{-1} that calculates the Cartesian coordinates of a tuple from its address.

3.2 Range Queries on UB-Trees

To answer a range query, only those Z-regions, which properly intersect the query box, must be fetched from the database and thus from the disk. Initially the range query algorithm calculates and retrieves the first Z-region that is overlapped by the query-box. Then the next intersecting Z-region is calculated and retrieved. This is repeated until a minimal cover for the query box has been constructed, i.e., the region that contains the ending point of the query box has been retrieved.
It is important to note that for any Z-region the calculation of the next point intersecting the query box is performed solely in main memory. [Bay97a] describes an algorithm that for a Z-region $[\alpha : \beta]$ and a query box Q calculates the smallest Z-address $\gamma > \beta$ that intersects Q in time exponential to the number of dimensions. A linear algorithm which is solely based on bit operations is described in [Mar99] and [RMF+00].

3.3 The Tetris Algorithm

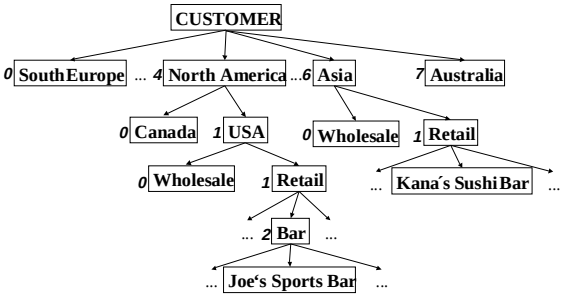
With the Tetris algorithm [MZB99], tables organized by a UB-Tree can be read in any key sort order in O(n) disk accesses where n is the number of pages of the table or the minimal number of regions covering a query box. The Tetris algorithm is a generalization of the multidimensional range query algorithm that efficiently combines sort operations with the evaluation of multi-attribute restrictions. The basic idea is to use the partial sort order imposed by a multidimensional partitioning in order to process a table in some total sort order. Essentially a plane sweep over a query space defined by restrictions on a multidimensionally partitioned table is performed. The direction of the sweep is determined by the sort attribute. Initially the algorithm calculates the first Z-region that is overlapped by the query box, retrieves it and caches it in main memory. Then it continues to read and cache the next Z-regions with respect to the requested sort order, until a complete thinnest possible slice of the query box (in the sorting dimension) has been read. Then the cached tuples of this slice are sorted in main memory, returned in sort order to the caller and removed from cache. The algorithm proceeds reading the next slice, until all Z-regions intersecting the query box have been processed. Only disk pages overlapping the query space are accessed. With sufficient, but modest, cache memory each disk page is accessed only once.

3.4 Multidimensional Hierarchical Clustering

Multidimensional Hierarchical Clustering by Hierarchy Interleaving (MHC/HI, [MRB99]) clusters a multidimensional data set on disk with respect to the hierarchical relationships in multiple dimensions. This is achieved by creating surrogate values that ensure that a hierarchical restriction like REGION = “North America” and NATION = “USA” is mapped to an interval restriction in a linear domain. With this technique, a star join on a schema with d dimensions therefore creates a d-dimensional interval restriction on the fact table which then may efficiently be processed by the UB-Tree.
In general one can imagine any foreign key relationship to be used for such a clustering. In the following we illustrate MHC/HI by hierarchical relationships as they usually occur in data warehousing applications. In ROLAP hierarchies are usually modeled implicitly by a set of attributes A_1, ..., A_n where A_i corresponds to hierarchy level i and level 1 is the root of the hierarchy.
Many attributes in relational DBMS in general and in data warehouses in particular have an actual domain of a very small set of values. A typical example (cf. Figure 4a) is the attribute REGION of the dimension table CUSTOMER, which has an actual domain of 8 values (*Southern Europe*, *Central Europe*, *Northern Europe*, *Western Europe*, *North America*, *Latin America*, *Asia*, *Australia*). By MHC/HI, these long strings are replaced by numbers. In the corresponding tables small numbers or bit strings are stored instead of long, space wasting character strings. If all records must be retrieved that concern Europe, only an interval [0;3] on this number is necessary. Otherwise every member of region must be specified. The mapped numbers are called surrogates because they replace the character strings. This data type is called *enumeration type*.

REGION	f(REGION)
South Europe	0
Middle Europe	1
Northern Europe	2
Western Europe	3
North America	4
Latin America	5
Asia	6
Australia	7

(a)



(b)

Figure 4: Surrogates for REGION and the Customer Hierarchy

This surrogate technique is generalized to all levels of the hierarchies. Figure 4b shows a part of the customer hierarchy with the surrogates propagated to all levels of the hierarchy.

To efficiently encode hierarchies, we introduce the concept of *compound surrogates* for hierarchies. Since we require hierarchies to form a disjoint partitioning of a dimension, a uniquely identifying compound surrogate for each child node of a hierarchy member exists and can be recursively calculated by concatenating the compound surrogate of the member with the running number of the child node. Because only efficient bit operations are necessary, the computation is extremely fast. The *ord* function returns the corresponding surrogate of a hierarchy member in the specified level.

The hierarchy path North America → USA → Retail → Bar (cf. Figure 4b) has the compound surrogate: ord_{Customer} (North America) ○ $ord_{\text{North America}}$ (USA) ○ ord_{USA}(Retail) ○ ord_{Retail}(Bar) = 4 ○ 1 ○ 1 ○ 2.

The upper limit of the domain for surrogates of level i is calculated as the maximum fan-out (number of children) of all members of level i–1 of a hierarchy H, i.e., surrogates(H, i) = max {cardinality(children(H, m)) where $m \in$ level(H, i - 1)}

The compound surrogate can be interpreted as a number. In the above example – using the surrogate length of Figure 4a with a length of 10 bits for the customer surrogate (3 for REGION, 3 for NATION, 1 for Trade Type and 3 for Business Type) the hierarchy path results in the compound surrogate $1000011010_2 = 538$.

Usually growth expectations for a hierarchy are known well in advance. Often hierarchy trees are even static. Therefore it is possible to determine a reasonable number of bits for storing each surrogate of the compound surrogate of a hierarchy. The overall number of bits necessary to store a compound surrogate is relatively small. For instance, a hierarchy tree with four branches on 8 levels already represents $4^8 = 65536$ partitions and is stored in only 16 bits. The maximum length of the compound surrogates for the 'Juice & More' schema that we used for our analysis can be computed from the maximum fan-out of the hierarchy levels given in Figure 4a.

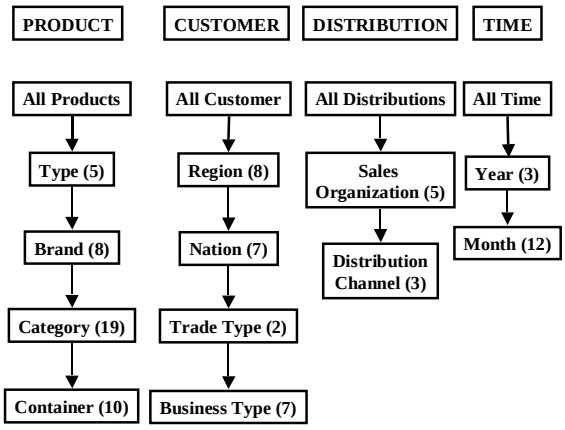
This very compact fixed length encoding preserves the lexicographic order on the hierarchy levels. Thus, point restrictions on upper hierarchy levels result in range restrictions on the finest granularity of a hierarchy. For instance, the point restriction NATION = "USA" on the second level of the CUSTOMER hierarchy with f("North America") = 4 = 100_2 and f("USA") = 1 = 001_2 maps to the range restriction cs_{customer} between 528 = 1000010000_2 and 543 = 1000011111_2. Thus, a star join with this surrogate encoding for the foreign keys of a fact table results in a range restriction on each compound surrogate, if some hierarchy level of each dimension is restricted to a point. In the same way intervals on the children of one hierarchy level result in a range of the corresponding compound surrogates (e.g., YEAR = 1998 and MONTH between April and June). A star join on a schema with d dimensions therefore creates a d-dimensional interval restriction on the fact table.

4 Schema and Queries of 'Juice & More'

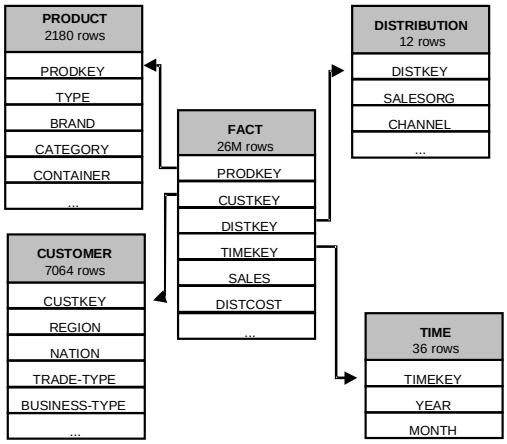
In this section we investigate, how UB-Trees and the Tetris algorithm in combination with MHC/HI may be used to accelerate star-joins, the most frequent operation of query processing for relational data warehouses.

4.1 The Juice & More Schema

We use the schema of the beverages supplier 'Juice & More', a real customer of one of our project partners[2]. In the data warehouse of 'Juice & More' data is organized along the following four dimensions: CUSTOMER, PRODUCT, DISTRIBUTION and TIME. Figure 5a shows the hierarchies over the dimensions (the number in parentheses specifies the maximum number of level members).



(a)



(b)

Figure 5: Hierarchies in the 'Juice & More' schema and the corresponding star schema

The ROLAP data model for the 'Juice & More' schema (Figure 5b) is a typical star schema with one fact table FACT and a table for each of the 4 dimensions. Let 'SALES' and 'DISTCOST' be some of the measures in the fact table. We used the methodologies of surrogates and multidimensional hierarchical clustering as described in Section 3.4 for clustering the fact table of 'Juice & More' with UB-Trees.

[2] The company and the data presented here have been made anonymous.

In the following we describe some example queries involving star joins for the 'Juice & More' schema. These queries were taken from the decision support system of 'Juice & More'. Thus next to the investigation of multidimensional hierarchical clustering this section is interesting from a second point of view: The highly skewed data distribution of the 'Juice & More' will prove that UB-Trees, the Tetris algorithm and MHC/HI are not only applicable to laboratory environment tests with generated data, but also prove their efficiency in the practical application scenarios. In order to show the highly skewed data distribution we included an entire section displaying the one-dimensional data distribution of 'Juice & More' for every dimension.
We then present measurements performed with our prototype implementation of the UB-Tree on top of Oracle 8. For the evaluation of our clustering technique we defined a benchmark with 36 queries. In comparison we also conducted measurements with native Oracle 8 access methods: parallel full table scan (FTS) and bitmap indexes (BII). For these measurements we used a completely denormalized fact table, that is, no additional joins next to the star join had to be performed to answer the queries. The bitmap indexes were created on each hierarchy level. We did not include secondary indexes in our comparison measurements because earlier experiments showed that they are neither competitive to the UB-Tree nor to FTS or BII [MZB99].

4.2 Queries on the 'Juice & More' Schema

In the following we present typical queries that are taken from the 'Juice & More' SAP business information warehouse. We will use these queries to illustrate our approach and we will present performance measurements for exactly these queries in Section 6.2.
Query 1 (Q1, cf. Figure 6) computes the sales for a given product group (TYPE and BRAND specified as (X1, X2)) and a given customer group (NATION and REGION specified as (Y1, Y2)) for the months from October to December of 1993. This query uses the UB-Tree range query algorithm on MHC/HI surrogates. However, surrogates are only used for clustering and indexing and thus are transparent to the user.

```
SELECT  SUM(Sales)
FROM  Fact F, Customer C, Product P, Time T
WHERE   F.ProdKey = P.ProdKey AND
     F.CustKey = C.CustKey AND
     P.Type = X1 AND P.Brand = X2 AND
     C.Region = Y1 AND C.Nation = Y2 AND
     F.TimeKey = T.TimeKey AND T.Year = 1993 AND
     T.Month >= October AND T.Month <= December
```

Figure 6: Time Interval (Q1)

Query 2 (Q2, cf. Figure 7) calculates the cost of distribution of the products of type X for each distribution channel. This query uses the Tetris algorithm on MHC/HI surrogates in order to efficiently calculate aggregations.

```
SELECT  SALESORG, CHANNEL, SUM(DistCost)
FROM    Fact F, Distribution D, Product P
WHERE   F.DistKey = D.DistKey AND
        F.ProductKey = P.ProductKey AND
        P.Type = X
GROUP BY D.SalesOrg,D.Channel
```

Figure 7: Distribution cost (Q2)

Query 3 (Q3, cf. Figure 8) restricts all dimensions on the first level of the hierarchies. This query also uses the Tetris algorithm on MHC/HI compound surrogates.

```
SELECT  SUM(SALES)
FROM Fact F, Distribution D, Product P,
        Customer C, Time T
WHERE   F.DistKey = D.DistKey AND
        F.TimeKey = T. TimeKey AND
        F.CustKey = C.CustKey AND
        F.ProdKey = P.ProdKey AND
        P.Type = t AND D.SalesOrg = s AND
        T.Year = y AND C.Region = r
```

Figure 8: Partial match query in first hierarchy level (Q3)

5 Data Distributions of the 'Juice & More' Fact Table

The data of 'Juice & More' is real world data; the data distribution of both the fact table and the dimension tables is highly skewed: The dimensions are neither distributed uniformly nor are independent. The original fact table consisted of 823.464 tuples (about 175 MB). To get a realistic large data cube, the fact table was enlarged to 26.350.848 tuples (about 6 GB). Our project partner implemented an augmentation algorithm with minimal impact on the data distribution (see [Pie98]).
In the following we show some charts that describe the one-dimensional data distribution for each dimension. However, we once more stress that the dimensions are not independent (e.g., some customers always order the same subset of products, some customers or products only exist for a certain time, etc.). Thus in general, the overall selectivity of a query restricting several dimensions is not the product of the selectivities of the one-dimensional restrictions (which is shown in the following charts). We will see this deviation in Section 6.
The fact table of 'Juice & More' stores several measures (e.g., distribution cost, sales) aggregated on a daily basis with respect to the dimensions time, customer, product and distribution. Since the data is sensitive real-world business data, it is not possible to show the labels/names of the hierarchy members in the charts.

5.1 The Time Dimension

The time dimension of 'Juice & More' consists of a two-level hierarchy of months and years. The test data stored the years from 1993 to 1995. As shown in Figure 5a, the days of the time dimension are organized by a two level

hierarchy (year and month). Figure 9 shows the cumulated data distribution of the fact table with respect to the time dimension grouped by year and month. The horizontal axis displays the hierarchy members, with "All" at the very bottom (i.e., the lowest level), the years 1993, 1994, and 1995 in the middle and the twelve months for each year above the year. The arrows in the horizontal axis indicate the relationship between the members of neighboring hierarchy levels.

Thus the fact table of 'Juice & More' is almost uniformly distributed with respect to the time dimension. The minimum number of facts for one month is 2,70% of the fact table (in January 1993, December 1993 and January 1995), whereas the maximum number of facts per month is 2,83% (in March of each of the three years). Thus with a multidimensional clustering using 5 bits for time, restrictions to one month in the time dimension (=1/36) can be expected to reduce the amount of data to around 1/25 = 1/32.

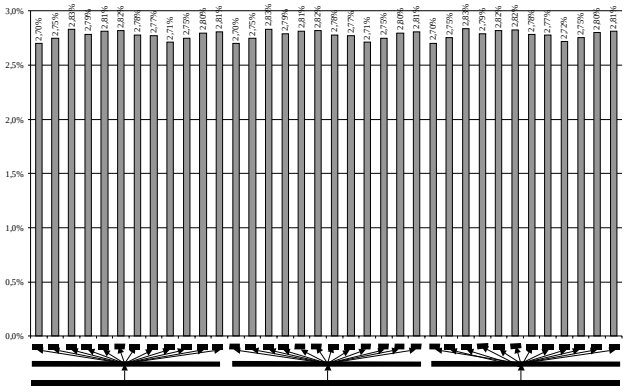


Figure 9: Data Distribution in the Time Dimension

5.2 The Product Dimension

The top level of the hierarchy on product has five entries. The data distribution is quite skewed, there are three product groups to which 93% of all tuples of the fact table belong. 1% of the data is unclassified. The distribution of the first level of the product hierarchy is illustrated in Figure 10.

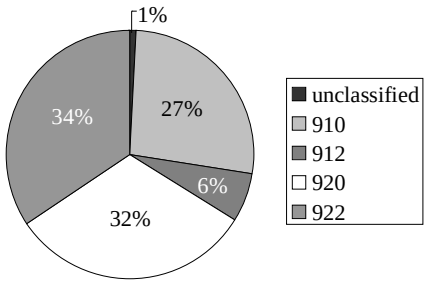


Figure 10: Product – first hierachy level

Multidimensional hierarchical clustering ensures that a restriction in the first hierarchy level will result in a 1%, 27%, 6%, 32% respectively 34% reduction of I/Os which are necessary to retrieve the result set. Without exactly showing the relationship between the hierarchy members, we show the skew of the data distribution over the first four product levels in Figure 11.

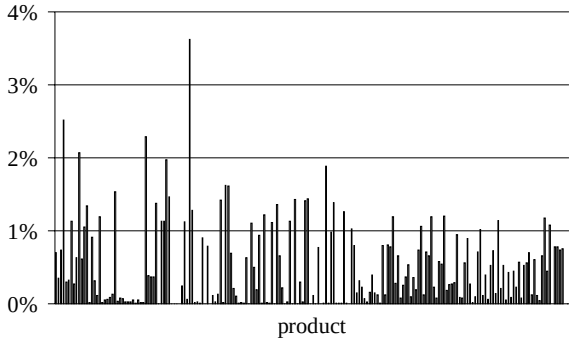


Figure 11: Product – first four hierachy levels

5.2 The Customer Dimension

According to business administration literature 20% of the customers contribute to 80% of the business. The customer dimension of 'Juice & More' is a typical example for a classification of customers in such a company. A high number of customers (in this case 88%, the very left entry of Figure 12) are not classified (maybe they are not interesting for the company because of small turnover or it is not possible to find a classification). The classified customer groups contain 0% to 3% of the tuples stored in the table.[3] The hierarchical relationships on the horizontal axis show that the hierarchy of the customer dimension is not balanced, since several hierarchy members just have one child.

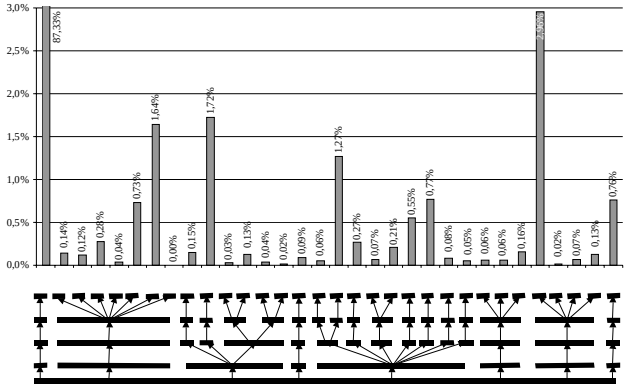


Figure 12: Customer – first four hierachy levels

[3] Note that the data in the 'Juice & More' warehouse is aggregated on a daily basis, thus the amount of data is usually compressed for large customers, thus the proportion of large customers is reduced.

The consequence of the small number of classified customers is that queries the restriction CUSTOMER will result in a selectivity of at most 3% and therefore the overall result set will be small.

5.2 The Distribution Dimension

There are seven entries on the first level of the distribution hierarchy. The data distribution of the fact table with respect to the distribution dimension is highly skewed. Figure 13 shows the distribution of the first hierarchy level (i.e., sales organization), while Figure 14 shows the distribution of facts in the fact table for each distribution channel of each sales organization. Again the arrows indicate the hierarchical relationship of the members of neighboring hierarchy levels with the hierarchy root "All" at the bottom of the Figure.

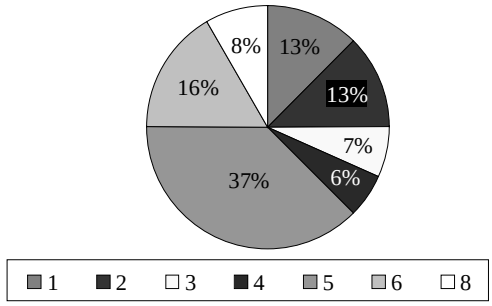


Figure 13: Distribution – first hierachy level

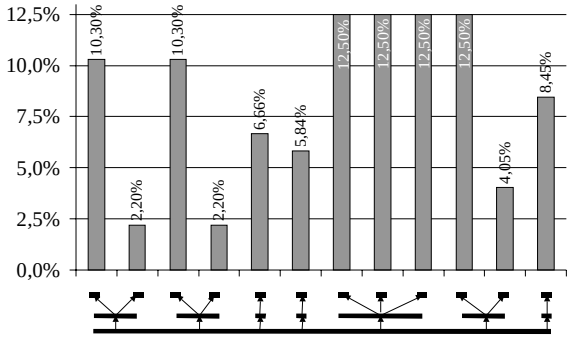


Figure 14: Distribution – first two hierachy levels

6 Performance Analysis of MHC/HI for 'Juice & More'

In this section we first analyze the performance of MHC/HI on UB-Trees analytically and then, taking the data distributions and query selectivities into account, verify our expectations by comparing the number of retrieved pages with the product of the selectivities over all dimensions. We then also give response times which show the superiority of UB-Trees and MHC/HI over traditional indexes even for our prototype implementation.

6.1 Performance Analysis

Figure 15 shows the compound surrogates for the 'Juice & More' data warehouse, which are calculated as fixed length compound surrogates in [MRB99]. For any of the 4 hierarchies the length of the compound surrogate does not exceed 15 bits and thus can be stored in a single integer value. These compound surrogates are used as attributes for each of the four dimensions of 'Juice & More' to calculate the Z-address for each tuple of the 'Juice & More' fact table.

$$cs_{\text{product}} = \underbrace{p_{15}p_{14}p_{13}}_{\text{level 1}}\underbrace{p_{12}p_{11}p_{10}}_{\text{level 2}}\underbrace{p_9p_8p_7p_6p_5}_{\text{level 3}}\underbrace{p_4p_3p_2p_1}_{\text{level 4}}$$

$$cs_{\text{customer}} = \underbrace{c_{10}c_9c_8}_{\text{level 1}}\underbrace{c_7c_6c_5}_{\text{level 2}}\underbrace{c_4}_{\text{level 3}}\underbrace{c_3c_2c_1}_{\text{level 4}}$$

$$cs_{\text{time}} = \underbrace{t_6t_5}_{\text{level 1}}\underbrace{t_4t_3t_2t_1}_{\text{level 2}}$$

$$cs_{\text{distribution}} = \underbrace{d_5d_4d_3}_{\text{level 1}}\underbrace{d_2d_1}_{\text{level 2}}$$

Figure 15: Compound surrogates for each dimension of 'Juice & More'

The UB-Tree for the 'Juice & More' fact table consists of P = 878362 pages, which corresponds to: $l = \log_2 P = \log_2 878362 = 19{,}7$ bits that are in any case necessary to address the bit-interleaved multidimensional space. With bit interleaving in the order of dimensions PRODUCT, CUSTOMER, TIME, and DISTRIBUTION the Z-address α for a tuple of the 'Juice & More' fact table is calculated as:

$$\alpha = \underbrace{p_{15}c_{10}t_6d_5p_{14}c_9t_5d_4p_{13}c_8t_4d_3p_{12}c_7t_3d_2p_{11}c_6t_2}_{\text{completely partitioned for any data distribution}}\ \underbrace{d_1}_{\text{partly split}}\ \underbrace{p_{10}c_5t_1p_9c_4p_8c_3p_7c_2p_6c_1p_5p_4p_3p_2p_1}_{\text{partitioned depending on the data distribution}}$$

The first 19 bits of the Z-address are guaranteed to be used to partition the four dimensional universe of the 'Juice & More' fact table. This means that the binary strings $p_{15}p_{14}p_{13}p_{12}p_{11}$ of the compound surrogate of product, $c_{10}c_9c_8c_7c_6$ of customer, $t_6t_5t_4t_3t_2$ of time and $d_5d_4d_3d_2$ of distribution are used to partition the universe. For each of the four dimensions the first hierarchy level is completely used for the partitioning. The second hierarchy level is used to a large extent to partition the universe. Therefore a restriction in the first hierarchy level will result in a reduction of the number of pages as determined by the data distribution of Section 5, i.e., a restriction of the product main group to "910" will reduce the number of pages to be retrieved to 27%, a restriction to "912" will result in a reduction to 6,41%. This holds for the restriction of the first hierarchy level in any dimension. If the top hierarchy level is restricted in several dimensions and the independence assumptions holds for these dimensions, the reduction is multiplicative. Figure 16 shows the predicted selectivity calculated as the product of the selectivity in each dimension, the actual selectivity and the loaded pages in percent of the entire pages in the database for two queries, which restrict the first hierarchy level of three out of four dimensions.

$s_{customer}$ in %	$s_{product}$ in $\%_t$	$s_{distribution}$ in %	s_{time} in %	Pages loaded	Predicted selectivity in %	Actual selectivity in %	loaded pages in %
0,78	6,4	37,5	100	178	0,0182	0,0199	0,0207
87,3	6,4	37,5	100	18996	2,0981	2,1618	2,1627

Figure 16: Restrictions in the first hierarchy level in 3 of 4 dimensions

However, the data is not independently distributed in the entire 4-dimensional universe of 'Juice & More'. In this case the predicted selectivity does not describe the actual selectivity anymore. Thus some bits of the first 19 bits are correlated. This means that not all combinations of these bits occur and some partitioning will take places in the bits below bit number 19 of the Z-address. In this case the second level may already be completely partitioned and even a third hierarchy level partitioning may have started for some dimensions. A typical part of the multidimensional space where this will happen is the customer hierarchy "unclassified", which stores 87,34% of the customers. At most the three bits $c_{10}c_9c_8$ of the customer hierarchy are needed to distinguish these customers from all other customers. Thus for the unspecified customers the bits c_7c_6 of the first 19 bits of the Z-address are correlated to $c_{10}c_9c_8$ and two further bits may be used for partitioning. Thus d_1 will be split completely, p_{10} will be used for the partitioning and t_1 will be partly split (c_5 is also correlated to $c_{10}c_9c_8$). Our measurements show that this effect also holds for other dimensions. Figure 17 shows queries where the first two hierarchy levels of CUSTOMER, PRODUCT and TIME are restricted, whereas the DISTRIBUTION dimension is not restricted. The selectivity predicted by the cost functions here differs from the actual selectivity of the query because of dependencies in the data distribution. However, the percentage of pages loaded is similar to the actual selectivity of each query.

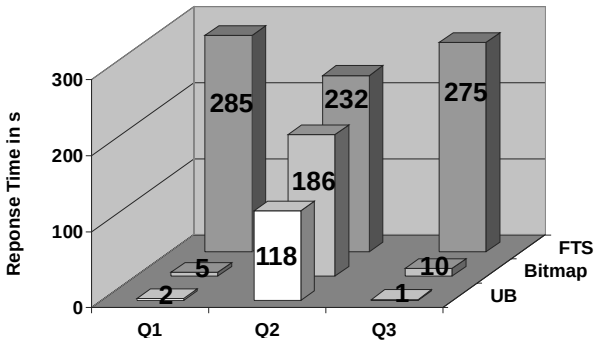
$s_{customer}$ in %	$s_{product}$ in %	$s_{distribution}$ in %	s_{time} in %	Pages loaded	predicted selectivity in %	actual selectivity in %	loaded pages in %
2,95	3,64	100	38,4	312	0,0414	0,0310	0,0355
2,95	27,99	100	38,4	782	0,0318	0,0851	0,0890

Figure 17: Restriction in the first two hierarchy levels in 3 of 4 dimensions

Each of the 878362 pages of the 'Juice & More' fact table stores 30 tuples. All of the measurements showed that when restricting the first hierarchy level in each dimension in average 99,99% of the tuples on the pages contributed to the result set. A standard deviation of less than 0,001 for these measurements means that the multidimensional hierarchical clustering is perfect for multidimensional restrictions in the first hierarchy level. When additionally restricting the second hierarchy level in average 557 pages were loaded, where 10,7% of the tuples did not contribute to the result set. The standard deviation here was 0,06. Additionally restricting the third hierarchy level of each dimension usually created result sets with only one page.
Thus multidimensional hierarchical restrictions are very well processed by UB-Trees storing compound surrogates which are created by the multidimensional hierarchical clustering technique. For star schemas as used in present data warehousing applications this approach significantly speeds up query performance and reduce resource requirements in disk space and processing time.

6.2 Performance Measurements

The measurements for 'Juice & More' were performed on a SUN Enterprise with four 300 MHz UltraSPARC processors and 2 GB RAM under Solaris 2.6. As secondary storage a partition on a SPARCstorage array with RAID level 0 (6 disks striping, 5-6 MB/s transfer rate per disk) was used. All measurements were done in a single-user environment.
It is important to note that our implementation still causes significant overhead due to the fact that we have implemented the UB-Tree on top of a DBMS and not in the kernel itself. First, the number of SQL statements that have to be processed (UB: 1 statement for each page in the result set, Oracle 8 methods: 1 statement in total) leads to extensive inter-process communication (about 30% of the total processing time) and DBMS overhead (e.g., parsing of statements). Second, our table is larger than the one for the FTS and the bitmap indexes due to unimplemented compressing techniques in the UB-Tree (for 8 KB pages: UB: 878362 pages, FTS: 723539 pages, BII: FTS+31134 pages).
Figure 18 shows result set sizes and response times of the three example queries (Section 4.2). Q1 shows that the UB-Tree with multidimensional clustering is over 2 times faster than BII even for very small result sets. Q3 which is processed by the unoptimized UB-Tree at least 10 times faster than with any other access method undermines this observation.



Query	Loaded Tuples	Percentage of Database
Q1	8160	0,03%
Q2	1696416	6,52%
Q3	19752	0,08%

Figure 18: Query response times and result set sizes

The result set of query Q2 is quite large but the almost perfect clustering factor of the UB-Tree (in average more than 29 out of 30 tuples/page belong to the result set) still leads to a speed up of more than 30 % in comparison to BII. The time for FTS for Q2 differs from the times for Q1 and Q3 due to the less complex WHERE clause of the statement. The number of comparison operations is therefore much smaller than for the other queries, which causes the faster execution.
All these results on real data show how well the multidimensional hierarchical clustering with UB-Trees works in practice and the accuracy of our theoretical cost model [MZB99]. In total more than 77% of all benchmark queries (28 out of 36) showed a speed up between a factor of 1.3 and 10 over traditional techniques.

	Customer	Product	Distribution	Time	Selectivity Instance 1 in %	Selectivity Instance 2 in %
Q4	2	2	0	1	0,0414	0,3176
Q5	1	1	1	1	0,0070	3,4383
Q6	1	1	1	0	0,0182	2,0981
Q7	1	0	0	1	1,1346	1,1346
Q8	0	1	0	1	6,0000	34,000

Figure 19: Restricted hierarchies and selectivities for five queries against the 'Juice & More' data warehouse

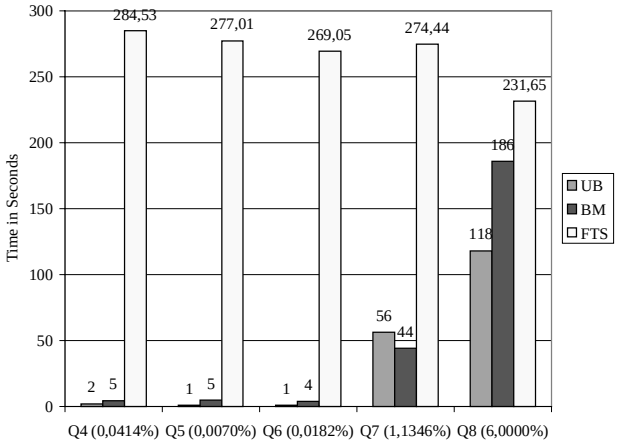


Figure 20: Performance Q4-Q8 (instance 1)

Figures 20 and 21 list two instances for each of five further queries of that benchmark. For each query Figure 19 lists the number of hierarchy levels that by each query are restricted to a point for each dimension. Since the data is non-uniformly distributed, the selectivity of each query depends on the exact point restriction, not only on the number of restricted hierarchy levels. We thus present two instances of the queries, each of which has a different selectivity.

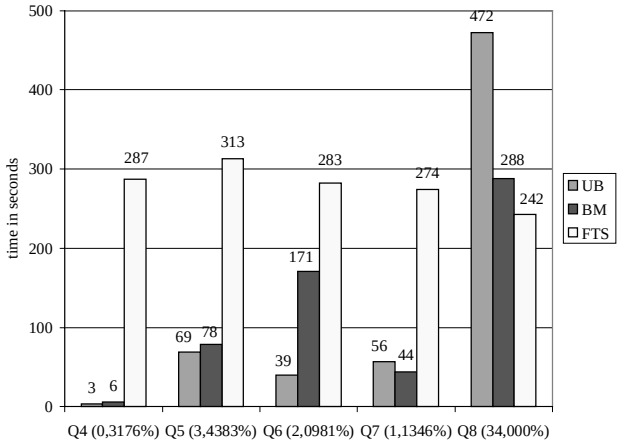


Figure 21: Performance Q4-Q8 (instance 2)

7 Conclusions and Future Work

We have presented multidimensional hierarchical clustering by hierarchy interleaving (MHC/HI) as a physical data modeling technique for OLAP applications in relational databases. Our performance measurements have shown that MHC/HI in combination with UB-Trees and the Tetris algorithm significantly reduces the number of random accesses to the fact table for star joins and other queries with restrictions in multiple hierarchies. This results in considerably lower response times for typical data warehousing queries with star joins. For a 6 GB retail data SAP business information warehouse, our prototype implementation of MHC/HI accelerates the processing of star-join queries more than a factor of two compared to bitmap indexes, clustering B*-Trees or parallel full table scans. For dimensionalities typical for data warehousing, only I/O-time linear in size of the result set prior to aggregation and sublinear temporary storage are necessary to aggregate parts of a fact table of a star or snowflake schema. Thus secondary storage space and pre-computation time for many aggregates and bitmap indexes can be avoided. In addition the widely discussed view maintenance problem is minimized. Depending on the query, temporary storage requirements for sorting are reduced by several orders of magnitude. Our clustering approach also holds not only for ROLAP but also for MOLAP implementations of a data warehouse since both ROLAP fact tables and MOLAP data cubes can be clustered in this way.
Our future work includes deriving a set of cost based decision rules for how to – possibly multidimensionally - index a relation for a given set of queries. We also intend to apply our model to cost based query optimization and combine the model with multidimensional histograms in

order to take dependencies and correlations between the attributes into account.

8 References

[Bay97a] R. Bayer. The universal B-Tree for multidimensional Indexing: General Concepts. WWCA '97, Tsukuba, Japan, p- 10-11, Springer Verlag, March, 1997.

[Bay97b] R. Bayer. UB-Trees and UB-Cache – A new Processing Paradigm for Database Systems. Technical Report TUM-I9722, TU München, 1997.

[CD97] S. Chaudhuri and U. Dayal. An Overview of Data Warehousing and OLAP Technologies. SIGMOD Rec. 26(1), 1997.

[CI98] C. Chan and Y. Ioannidis. Bitmap Index Design and Evaluation. SIGMOD, 1998.

[Dat88] C.J. Date. A Guide to the SQL Standard,2nd edition. Addison-Wesley, 1988.

[GG97] V. Gaede and O. Günther. Multidimensional Access Methods. ACM Computing Surveys 30(2), 1997.

[GHR+97] H. Gupta, V. Harinarayan, A. Rajaraman, and D. Ullman. Index Selection for OLAP. ICDE, 1997.

[GR97] J. Gray and A. Reuter. Transaction Processing: Concepts and Techniques. Morgan Kaufmann Publishers, 1997.

[Gra93] G. Graefe. Query Evaluation Techniques for Large Databases. ACM Computing Surveys 25, 1993, pp. 73-170.

[Gup97] H. Gupta. Selection of Views to Materialize in a Data Warehouse. Proc. of the Intl. Conference on Database Theory, Athens, Greece, January 1997.

[Gut84] A. Guttman. R-Trees: A dynamic Index Structure for spatial Searching. Proc. of ACM SIGMOD Conf., 1984, pp. 47-57.

[HAM+97] C.T. Ho, R. Agrawal, N. Megiddo, and R. Srikant. Range Queries in OLAP Data Cubes. SIGMOD Conf., 1997, pp. 73-88.

[JL98] Jürgens, M.; Lenz, H.J.: The R*a-Tree: An improved R*-Tree with Materialized Data for Supporting Range Queries on OLAP-Data, DWDOT Workshop, Vienna, 1998.

[Kim96] R. Kimball. The Data Warehouse Toolkit. John Wiley & Sons, New York. 1996.

[LS90] Lomet, D.; Salzberg, B.: The hB-Tree: A Multiattribute Indexing Method with good guaranteed Performance. ACM TODS, 15(4), 1990, pp. 625 – 658.

[Mar99] V. Markl. MISTRAL: Processing Relational Queries using a Multidimensional Access Method. Ph.D. Thesis, TU München, 1999.

[Moe98] G. Moerkotte. Small Materialized Aggregates: A Light Weight Index Structure for Data Warehousing. VLDB Conference. New York, USA, 1998.

[MRB99] V. Markl, F. Ramsak, and R. Bayer. Improving OLAP Performance by Multidimensional Hierarchical Clustering. Proc. of IDEAS'99, 1999.

[MZB99] V. Markl, M. Zirkel, and R. Bayer. Processing Operations with Restrictions in Relational Database Management Systems without external Sorting. ICDE, 1999.

[NHS84] J. Nievergelt, H. Hinterberger, and K. C. Sevcik. The Grid-File. ACM TODS, 9(1), March 1984, pp. 38-71.

[OM84] J. A. Orenstein and T.H. Merret. A Class of Data Structures for Associate Searching. SIGMOD-PODS Conf., Portland, Oregon, 1984, pp. 294-305.

[OQ97] P. O´Neill and D. Quass. Improved Query Performance with Variant Indexes. ACM SIGMOD Intl. Conf. On Management of Data, Tucson, Arizona, pp. 38-49,1997.

[Pie98] R. Pieringer. Evaluation of the UB-Tree in the SAP Environment. Master Thesis, TU München, 1998.

[RMF+00] F. Ramsak, V. Markl, R. Fenk et al. Integrating the UB-Tree into a data-basesystem kernel. VLDB2000, Cairo, 2000.

[Sar97] S. Sarawagi. Indexing OLAP data. Data Eng. Bulletin 20 (1), pp. 36-43, 1997.

[Sam90] H. Samet. The Design and Analysis of Spatial Data Structures. Addison Wesley, 1990

[SDN+96] A. Shukla, P. Deshpande, J. Naughton, and K. Ramasamy. Storage Estimation for Multidimensional Aggregates in the Presence of Hierarchies. VLDB Conference. 1996.

[SDN+98] A. Shukla, P. Deshpande, and J. Naughton. Materialized View Selection for Multidimensional Datasets. SIGMOD Conf., 1998.

[TS97] D. Theodoratos and T. K. Sellis. Data Warehouse Configuration. VLDB, 1997, p. 126-135.

[SRF97] T. K. Sellis, N. Roussopoulos, C. Faloutsos: Multidimensional Access Methods: Trees Have Grown Everywhere. VLDB, 1997, P. 13-14.

[WB98] M.C.Wu and A.P. Buchmann. Encoded Bitmap Indexing for Data Warehouses. ICDE, Orlando, 1998.