dimensio informatics

White Paper

Technical background information
on the product

DIMENSIO

Ultra high speed for complex database queries

Dramatic performance gain through the indexing and grouping of database content based on context correlation

1. Introduction

Database systems generally store large volumes of data records organised according to unique identifying characteristics on which a search can be based.

Such characteristics may be staff ID numbers, contract numbers or other unique identifiers. They are referred to as primary keys and generally enable the organised storage of data records according to the order criterion of the identifier (i.e. in ascending or descending order). Thus, primary keys span a linear, one-dimensional search space across the stored data records within which various methods can be used to search for data records.

Common methods for the fast retrieval of data records divide the one-dimensional search space into intervals and arrange them sequentially or in tree form in order to speed up the search. Such methods are referred to as database indexes or – if the context is clear - simply as indexes.

This process can also be illustrated using the table of contents and the index of a book as an example: on the one hand, the table of contents depicts text intervals in the form of chapters and sections and, on the other, represents their arrangement/order in the book. In contrast to the content-oriented or semantic formation of intervals in books, database indexes simply divide the search space according to formal mathematical criteria (such as equal intervals, metrics, etc.) with no consideration for the semantics of the data records.

Nonetheless, this type of search optimisation is more than adequate for classic database applications, such as HR management, warehousing, etc., which is why the database systems commonly used for this type of indexing are highly optimised and extremely fast.

However, the emergence of so-called non-standard applications via databases towards the end of the 70s heralded the problem of multi-dimensional search spaces, as linear value ranges for primary keys were often no longer adequate for this type of data. One such example is image data - where the pixels within an image are represented by an X/Y coordinate, i.e. they are no longer uniquely identified by a single value, but by a value pair.

Indexes were also developed and generated for this type of data: so-called multi-dimensional database indexes. Generally speaking, these indexes are also based on the principle of interval formation, just for more than one dimension, with an optimum arrangement of these intervals in data structures, such as in multiway trees or in multi-dimensional grid structures.

While this method of procedure ensures fast data access, the retained criterion of the formal-mathematical formation of intervals ensures that there is no context correlation of data records.

However, in the last 10 - 15 years, the application range of databases has expanded yet again and applications via multi-dimensional search spaces increasingly refer to the contextual/semantic relationships between search space dimensions. Just one example of this type of application can be found in so-called Data Warehouse applications, in which data records

from different databases are merged in order to determine any corporate policy links between data or for the purpose of gleaning strategic information.

Example:

The following simple example from everyday life illustrates the problem of semantic relationships between search space dimensions: all cook books contain recipes for a wide range of dishes, such as desserts, soups, dishes with meat, etc. and each recipe contains a list of ingredients.

However, a person suffering from celiac disease (an intolerance to gluten, frequently accompanied by a lactose intolerance) would be required to view each recipe individually with regard to any incompatible ingredients, thus rendering the usual recipe classification of little use.

However, if these recipes were stored in a database with a multi-dimensional search space based on the ingredients of the recipes, an interval formation based on context would enable the generation of both the usual classification and one that would be suitable for highlighting any ingredients that might trigger an allergic reaction.

In modern database systems, these types of applications are generally handled via additional index structures. A so-called secondary index is created for each dimension of the search space. This index does not form primary key intervals, rather it groups the primary keys of data records according to the occurrence of specific values in the dimension of the respective search space. In this case, it is once again useful to draw the analogy between database indexes and indexes in books that contain page references for specific key words.

However, this method also tends to generate a large number of additional data structures (a complete extra index per dimension) that overlap the primary key index. While this method considerably accelerates the location of data records, it also severely hampers any modification of data or the addition of new data, as each secondary index needs to be updated separately.

Conclusion

It therefore follows that the ideal index is one that enables the compact and efficient organisation of the context dependencies of the dimensions of a multidimensional search space in a single data structure. This type of index can be used to replace multiple secondary and multi-dimensional special indexes.

The **DIMENSIO** database index offered by dimensio informatics GmbH implements just such a complex, semantically classifying data structure. The semantic classification components of **DIMENSIO** distinguish it from conventional database indexes.

2. DIMENSIO

DIMENSIO is a multi-dimensional semantic index for databases with high-dimensional data and comprises one classification and one management component. This method of classifying database content uses artificial neural networks for parameter-free classification and is able to group data records within a database according to their contextual dependencies. This group is then transferred to a highly efficient management structure, which is similar to an R-tree (or so-called V-tree).

How does it work?

Compared to other multi-dimensional index methods, **DIMENSIO** is most efficient when it comes to accelerating queries that contain long predicate lists, i.e. those that have a large number of value or attribute comparisons in the SQL WHERE clause. Our recipe database might handle a query using the following relation:

Recipe (milk, egg, flour, butter, margarine, oil, pork, beef, lamb, beans, peas, carrot, asparagus, potatoes, salt, pepper, paprika and sugar);

whereby, to simplify matters, all attributes will be modelled as Boolean values.

```
select  *
from    Recipes
where   flour = false      and
        milk = false       and
        beef = true        and
        peas = true        and
        carrots = true     and
        asparagus = true;
```

The above would represent a recipe database query for roast beef with mixed vegetables taking into consideration an intolerance to gluten (flour = false) and lactose (milk = false).

In order to understand how **DIMENSIO** operates, we will use this example to demonstrate the individual processes in the following section.

Learn process

In the first step, the user (i.e. the database administrator) specifies the attributes relevant for indexing. In our case, that means all the attributes of the relation **Recipes**. This adequately defines the multi-dimensional search space where each attribute with its value range (the value range actually contained in the database, not the range that is theoretically possible) spans a single dimension and from where the data records are to subsequently retrieved.

These user data are converted to the internal format used in the index by either dimensio informatics GmbH or an accredited system partner (for further details on the adaptation of the query processor – please refer to the section "Architecture"), whereby the costs for this process are minimal.

In the second step, the index imports the database content in blocks and bases the learn process on the data records already contained in the database. The learn process uses the following method:

With its individual value combination in the attributes to be indexed, each data record marks a point in the multi-dimensional search space (Fig. 1). These points are initially noted by neurons as the centre of potential "point clouds", i.e. it is assumed that further data records exist, whose points in the search space will also be in close proximity to the respective neuron (Fig. 2).

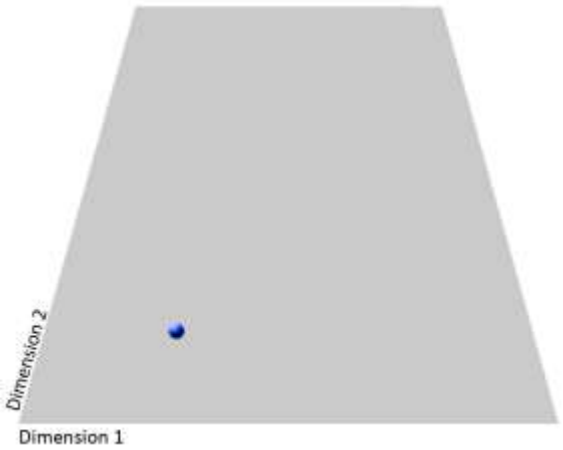


Figure 1

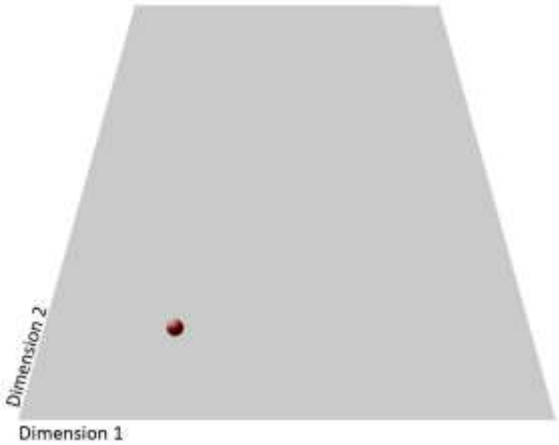


Figure 2

If data records are added with similar content to those already learned, and if their points are close to existing cluster centres (Fig. 3), this causes the neurons – and thus the cluster centres - to be slightly displaced within the search space (Fig. 4).

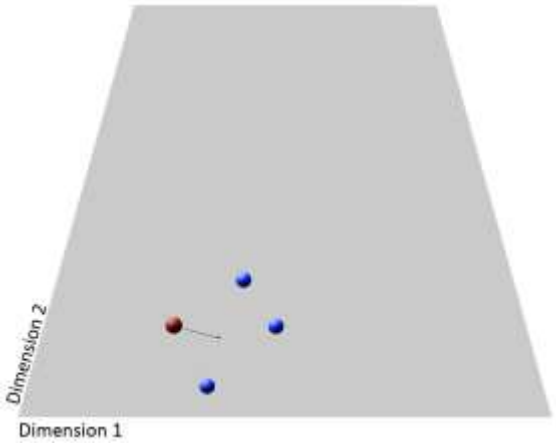


Figure 3

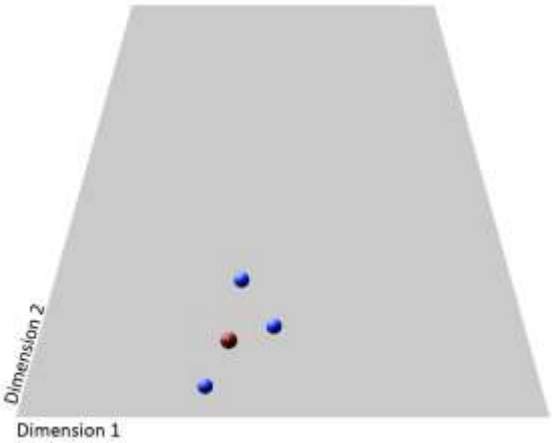


Figure 4

This ultimately produces point clouds around the neurons, whose points represent data records with similar content (Fig.5). In this case *similar content* is the equivalent of a combination of values of the attributes to be indexed and which are located in close proximity to each other on the respective value range scale.

If the diagrams are interpreted such that *Dimension 1* represents the x-coordinate and *Dimension 2* the y-coordinate, then the data records with value combinations (medium x value, small y value) in Figure 5 are located towards the bottom centre, the data records with value

combinations (large x value, medium y value) are to the right in the middle of the space and the data records with value combinations (small x value, large y value) are at the top left. At the same time, you can see that the neurons representing the centre of a cluster have shifted roughly to the middle of the cluster.

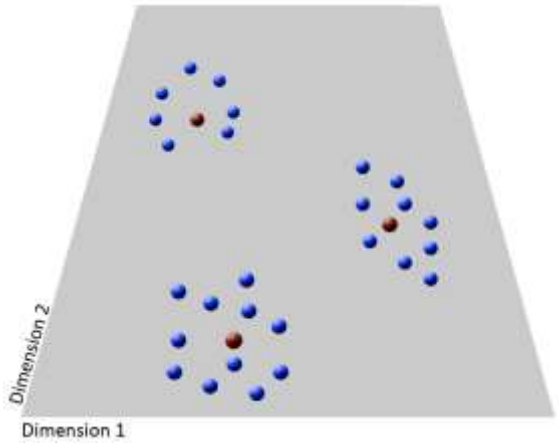


Figure 5

Because our learn process operates as a refinement hierarchy, and when using a tree-like refinement hierarchy we can regard the entire database as a root node, Figure 5 represents the first node level after the root node. Accordingly, large clusters, or clusters containing groups in which additional similarities can be detected, can be further divided to form a number of subgroups (Fig. 6), which leads to further node levels.

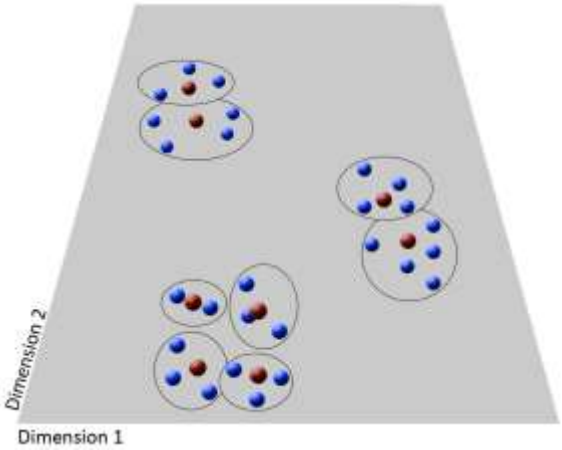


Figure 6

Taken to its extreme, the refinement could be pursued to such a degree that each cluster only contained a single data record – which would rarely be useful as queries are generally performed for value ranges and/or record quantities. For this reason, the learn process is halted in accordance with user specifications (e.g. minimum number of data records per cluster or maximum tree depth).

Overall, this learn process produces a hierarchically refined image of the similarities of all data records in the database in accordance with the combinatorics of their values in the attributes specified for the formation of the index (Fig. 7 shows an inserted fictive root node and Fig. 8 shows the nodes of the first level from Fig. 7 displayed on a small scale).

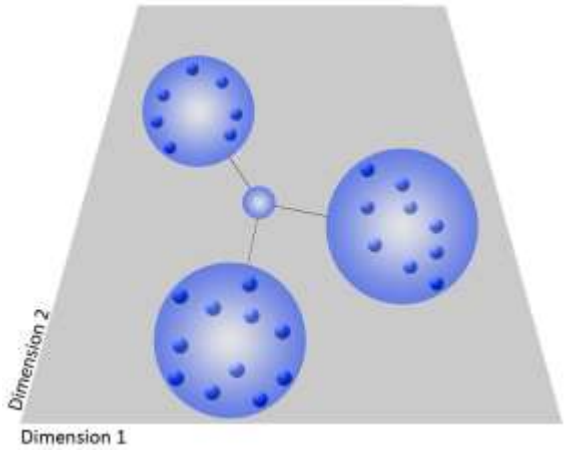


Figure 7

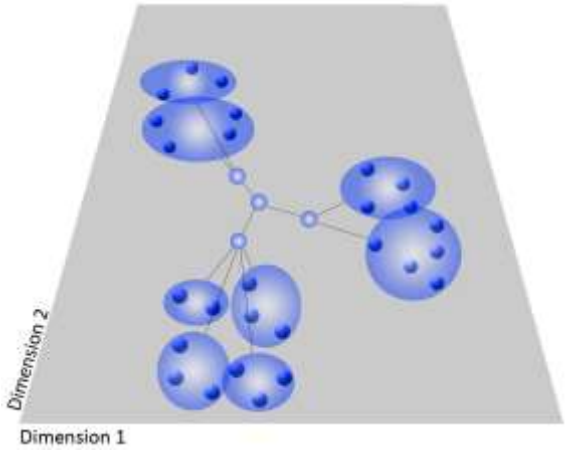


Figure 8

In this case one could use *taxonomy* as an analogy for the hierarchy produced, as it groups objects according to similar characteristics and assigns the groups to categories (e.g. in the field of biology, the taxonomy of zoology or the taxonomy of botany). While our method also groups objects (namely data records) it does not assign them to categories, because in this case we are dealing with a learn process that is not monitored by humans and the classification of objects or object groups represents a linguistic process controlled by man that is completely independent of any recognition of an object or object group.

In order to better understand the process results, the structure produced (Fig. 8) can be represented by a radial tree, where the root of the tree is in the centre and – depending on the number of dimensions used to represent the tree – the nodes are arranged around it, either in the form of balls or spheres. Figure 9 shows one such radial tree in 3D. This represents an actual test database indexed by **DIMENSIO**.

The nodes of the tree represent the clusters formed during the learn process and the tree edges represent the refinement hierarchy. The nodes of the first level have been colour-coded and this colour coding transferred to the subsequent child nodes. In this case therefore , the node colouring indicates the relationship to specific sub-trees.

Staying with our example of the recipe database, one could interpret the green-coded subtree of the radial tree such that these nodes contain recipes with beef, while the blue nodes represent pork recipes - and the magenta nodes lamb recipes.

And because the combinatorics of the attributes are increasingly specialised with each level, we can assume that our query for a roast beef recipe with mixed vegetables that is suitable for celiacs, i.e. with the condition that specific attribute values (flour and milk) may NOT (!) occur, will quickly restrict the search path in the radial tree, thus considerably accelerating the search for the respective cluster.

Please note, however, that in practice, the process does not of course generate a graphical display. This was included in this document for explanatory purposes only.

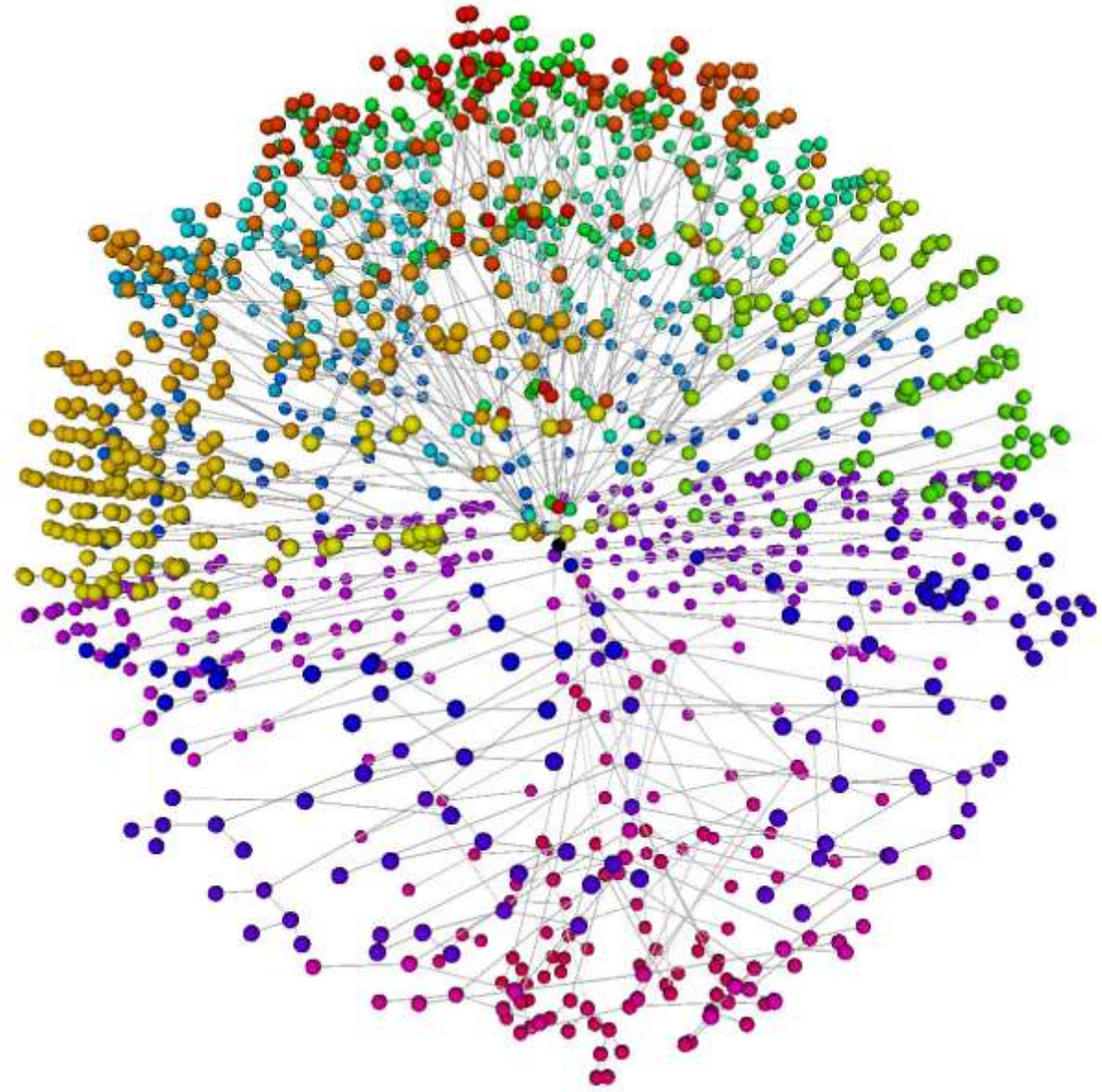


Figure 9

Evaluation

As already mentioned, the diagrams depicting the learn process results in Figures 7 to 9 have been included solely to aid understanding of the method used. Furthermore, it is fairly clear that the tree depicted in Figure 9 shows a rather heterogeneous node distribution and is unbalanced. It follows that this result cannot be used for the purpose of evaluation as an index structure unless converted into more suitable data structures.

On completion of the learn process, the learn results are therefore converted into a so-called management tree, with a node structure that is able to accommodate the elements of the clusters. Because the clusters define multi-dimensional spaces formed on the basis of the point clouds of data records, and because the R-tree is a data structure commonly used in the database world to index spatial structures, the management tree is a special version of the R-tree (Fig.10 and Fig. 11).

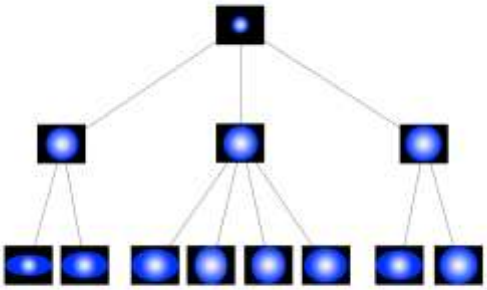


Figure 10

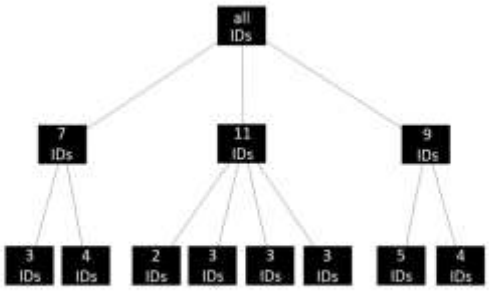


Figure 11

An evaluation process can now be carried out as shown in the following section (for the integration of **DIMENSIO** in a client's existing IT landscape, please refer to chapter 4; the description below merely deals with the **DIMENSIO** architecture components, client and server):

During a normal query without **DIMENSIO,** we assume here that a database application sends your SQL queries directly to the database system. These queries contain predicate lists, which refer, among other things, to the contextual relationships between data records learned by the index, as well as those where this is not the case. If the queries contain predicates that have not been learned, the query is transferred directly to the database system.

However, if the queries contain predicates from the learned index range, they are not transferred directly to the database system, but are initially sent to the **DIMENSIO** client (which can be addressed at various points, depending on the method of integration - see chapter 4). In each case, the client then forwards the query to the **DIMENSIO** server for index evaluation.

In the server, the predicates of the predicate list are evaluated such that the specified attribute values determine in which clusters – or in which tree node of the index management tree - the search for the data records is to be initialised. However, because the tree node does not contain any data records, but merely its representatives (primary key, see Fig. 11), the server does not search the database directly for the data records, rather it replaces the predicate list of the original query with a specific number of primary keys. This number corresponds directly to the number of data records originally sought. Thus amended, the query is forwarded to the database system where it is executed.

Let's take another look at the *where* clause of our sample query,

```
select  *
from    Recipes
where   flour = false     and
        milk = false      and
        beef = true       and
        peas = true       and
        carrots = true    and
        asparagus = true,
```

We can now see that the first two predicates represent a very extreme exclusion criterion, as the vast majority of recipes will contain these ingredients - certainly flour. We can therefore assume that in the first step of the tree search, well more than half of the nodes in the first level that serve as the stepping stone to any further search of specific sub-trees can be excluded and that the index will quickly find a cluster that contains the predicate combination being sought.

However, even if the query does not contain such a strong exclusion criterion – which will generally be the case – the index will find the specified cluster faster than other indexes because the special combinatorics of the sought attribute values lead directly to the search result. There is no longer any need to evaluate the predicate lists by directly searching attributes or indexed attributes.

In summary, we can say: because **DIMENSIO**'s classification method learns all the combinatorics of the specified attributes (based on the data records actually stored in the database), **DIMENSIO** knows what your query is before you do!

Differences - different from the rest?

The process: **DIMENSIO** is so much faster than other methods because the query is dramatically simplified. Even in the case of long predicate lists, the database system is now only required to access a single primary key. i.e. **DIMENSIO** relieves the database system of the task of evaluating long predicate lists through the evaluation of the learned clusters! All that is left is a concrete search of the data records for their primary keys – and all modern database systems are optimally designed for just this task.

The system: **DIMENSIO** is a standalone product that can be deployed in any data management system and any IT landscape! While not an integral part of an application, it can be integrated in applications via API. And while not an integral part of a database system, plug-in technology enables simple integration as a third-party system. Because **DIMENSIO** is geared towards the actual content of data record attributes, it is not even necessary for the data records to be in a database - whatever form that may take.

Technology: **DIMENSIO** requires no in-memory technology, but operates extremely well with systems equipped with this technology. Together, both systems achieve considerable performance gain due to the fact that the associated data records of the primary keys grouped in the index are not on the hard drive itself, but are stored in compressed, or maybe column-based, form in main memory. Thus, the latter disk access operations are also no longer required.

DIMENSIO itself does not use any special form of primary organisation (such as column-based), but can operate with systems that are organised in this way. The same is true of compression technology: **DIMENSIO** does not use data compression because no data records are stored in the index (except in the cache), but operates with systems that use data compression. And it goes without saying that both statements also apply in combination.

Summing up: as an independent solution, **DIMENSIO** is designed to be minimally invasive, making it absolutely ideal for seamless integration in existing IT landscapes (computing centres, clouds, etc.)! Even if client-owned applications and data management systems do not permit even minimal intervention - for whatever reasons - **DIMENSIO-Appliance** offers a solution (please refer to section 3 for further details). Furthermore, the loose coupling of the index to the data management system means that it will operate with any model. In other words: **DIMENSIO** also operates without a DBMS, e.g. with so-called flat files.

Architecture – Familiar with Lego®?

DIMENSIO is a client/server system. The client merely accepts the queries and forwards them to the server. This enables maximum independence from the operational environment, which means **DIMENSIO** can be integrated in a wide range of IT landscapes while still ensuring high system stability. The following section describes the architecture of the server.

The focus was on a consistent modular architecture, which enabled us to achieve dramatic performance gain using standardised functional units. Dividing the architecture into *module level* and *base level* (Fig. 12) enabled the actual functionality to be decoupled from the required infrastructure. The module level enables compilation of the required functional requirements and performance.

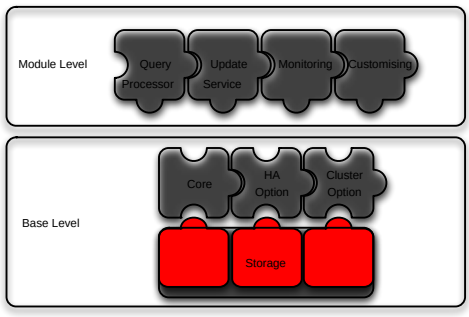


Figure 12

From the client's standpoint, the *query processor* is the most important component because it is geared towards the client's area of application (e.g.: BI, full-text search or post-processing) and is adapted to the attributes that need to be learned (see section "How does it work"). If a client intends to use **DIMENSIO** for a range of applications simultaneously, these components can be created several times over in a single system.

However, the u*pdate service* is equally important because it ensures that **DIMENSIO** is not merely an index for *information retrieval* purposes – i.e. merely an evaluation tool – but can also be deployed securely and reliably in real database operations with update and delete functionality. A coordinated procedure between learn and management components ensures that the index is constantly up-to-date and effects a balanced lookup behaviour if the tree is unbalanced.

Technicians and administrators probably see the *monitoring component* as a key component, as it enables them to monitor the load distribution in the system and any performance gain.

The *customising component* represents the active interface between system administrator(s) and **DIMENSIO**. It allows adjustment of the learn behaviour, management of database connectivity and adaptation of data modelling.

Compared to the module level, the base level represents the actual system environment in which the configured modules of the module level are executed, whereby active load balancing improves availability of the components.

The standard version of the base level contains the memory for storing the index and for buffering frequently requested data records and the *core unit*, which is required to process the index.

This core unit can also be optionally supplemented with an *HA option* (HA = High Availability), which seamlessly takes over the tasks of the core unit in the event of a failure, thus ensuring failsafe operation.

Another optional extra is the *cluster option*, which takes over the load balancing of additional core units, thus further enhancing the performance of the overall **DIMENSIO** system. Deployed with this component, **DIMENSIO** is also cloud ready (see version "**DIMENSIO Power Cloud**" in the next section).

3. Integration

DIMENSIO is an extremely low-cost, minimally invasive tuning product that is easy to integrate in existing IT landscapes. The server (see section "Architecture") is available in four versions:

- **DIMENSIO Appliance**: Pre-configured, optimised complete system.
- **DIMENSIO Virtual Appliance**: Uses existing client resources.
- **DIMENSIO Software Service**: Can be integrated in existing software.
- **DIMENSIO Power Cloud**: Cloud computing solution

In order to meet varying customer requirements, there are three technical options for the integration of **DIMENSIO** in an existing IT infrastructure/software application.

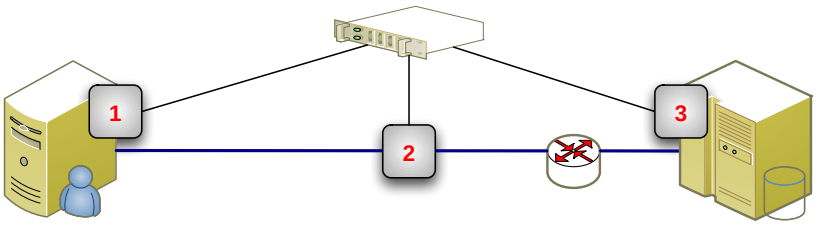


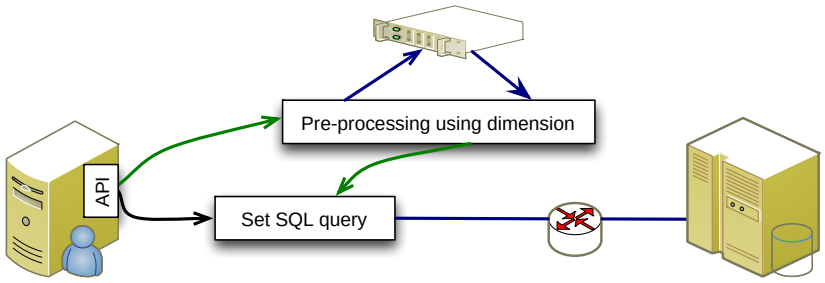
Figure 13

Figure 13 shows the available integration points (1-3). These are described in further detail in the following section.

Integration point 1 - the application

This approach is geared towards software providers (software companies, ISVs) or companies that have access to the source code of their application, but may also be suitable for so-called end customers. Only minimum changes to the code are required, as only the **DIMENSIO** client is addressed by the application and a separate program library (API) is provided for this purpose. This enables smooth and cost-efficient integration. However, the requirements of software companies/ISVs and end customers often differ widely:

Our **DIMENSIO Software Service** is the best way for software companies and ISVs to implement the **DIMENSIO** server, as it can be fully integrated in client applications and is invisible to end customers. This option enables software companies and/or ISVs to dramatically enhance the performance of proprietary products and benefit from this performance gain under their own name.



Pre-processing using dimension
Set SQL query

Figure 14

If the client wishes to benefit from the product without integrating it in the application, which may be the case with end-customer projects, we recommend the solution **DIMENSIO Virtual Appliance** and, in some cases, **DIMENSIO Appliance** (Fig. 14).

DIMENSIO Virtual Appliance runs on customer-provided IT, i.e., the customer provides a virtual machine with sufficient performance for the **DIMENSIO** server and adequate storage capacity. If that is either not possible or not acceptable, **DIMENSIO Appliance** may be a better option, as it is completely pre-configured by dimensio informatics GmbH and comes with its own enclosure pre-fitted with all the necessary connections.

In the two latter cases it is only necessary to set up the addressing of the **DIMENSIO** client in the application, which can generally be achieved with just a few lines of code.

Integration point 2 - the network

For IT service providers, such as computing centres and/or cloud operators, an extremely elegant way to integrate **DIMENSIO** in an existing IT infrastructure is integration in the network (Fig. 15). This integration option requires absolutely no modifications to the existing IT infrastructure. Neither application nor database require intervention or modification.

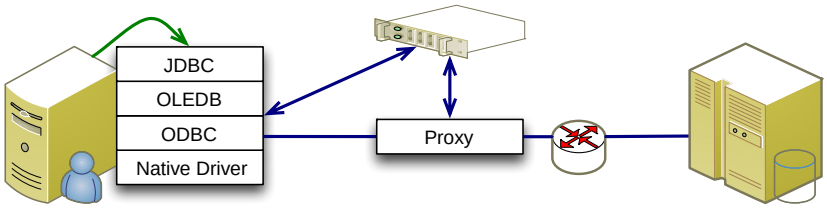


Figure 15: (Native driver)

With this solution, the **DIMENSIO** client is integrated in the database driver. The application is prevented from directly accessing the database and a bypass is installed that checks all database requests. A proxy filters the queries, and tables indexed by **DIMENSIO** are forwarded to the **DIMENSIO** server. The **DIMENSIO** server optimises queries to indexed tables, supplements them with information to accelerate database access, and forwards these queries to the database. Responses from the database are then forwarded directly to the requesting application.

In this case, **DIMENSIO Appliance** is the best option for normally structured IT service providers – such as computing centres - or **DIMENSIO Power Cloud** for cloud operators. Each solution comprises a pre-configured, completely autonomous product that is easy to integrate in the client network. These solutions most aptly represent the term "minimally invasive".

Integration point 3 - the database

A quick and simple integration option for users with access to the database system, or for service providers who adapt database systems, is direct connection of the **DIMENSIO** server to the DBMS (Fig. 16).

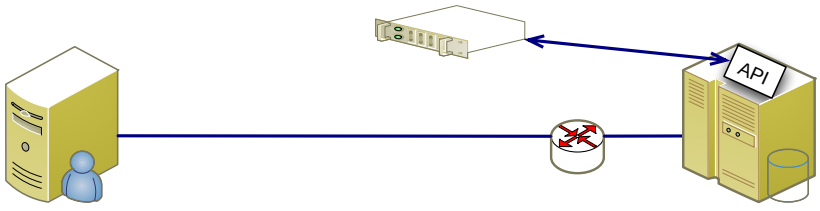


Figure 16

In this case, the **DIMENSIO** client is added to the DBMS as a plug-in, whereby the DBMS must provide a suitable interface. Oracle and DB2 are currently the only providers that offer

interfaces with sufficient functional scope for **DIMENSIO**. However, in the case of open source database systems, integration can be enabled directly in the system code.

If the application accesses tables indexed by **DIMENSIO,** the DBMS forwards the query to the interface, and thus to the **DIMENSIO** client. When requested by the client, the **DIMENSIO** server then delivers the indexing information directly to the DBMS via plug-in.

This version offers the greatest potential for performance gain as, in addition to integrating the **DIMENSIO** system, it offers considerable scope for collaboration between database manufacturers and dimensio informatics GmbH.

4. Applications

As already mentioned, conventional database indexing methods divide the search space into mathematically or technically defined intervals, without taking into account the *context correlation* of data records. Conversely, it is generally the case that programmes from the AI sector that analyse data content do not have the capability to subsequently use these analysis results to facilitate access to large amounts of data. However, by integrating technologies used in both areas - artificial intelligence and databases - **DIMENSIO** has made it possible.

The successful implementation of **DIMENSIO** is basically dependent on whether, and to what extent, the classification component provides added value for the indexing of data volumes (in contrast to mathematical/technical, content-independent index formation). This has definitely proven to be the case when dealing with complex queries that require the correlation of a large number of attribute values.

Business Intelligence - Data Warehouse applications

Data Warehouses are used to store heterogeneous data at a single site where they are evaluated according to general criteria (physical data integration). These evaluations are generally based on corporate policy objectives, pending management decisions or for the analysis of business processes over a prolonged period. In such cases - compared to cluster solutions with in-memory technology, **DIMENSIO** increased speeds by up to a factor of 10^3 = 1,000.

Full-text search

Full-text searches in large data volumes have become daily routine, both in the private sector (e.g. Internet search engines) and in business environments (document management systems, DMS). Despite the fact that these search engines operate very efficiently, and thus very quickly, **DIMENSIO** was able to demonstrate that with an orthogonal search approach (no indexing, just a window-based search in a 26^3 dimensional search space), it was possible to achieve a performance gain of a factor of 15 or higher.

Telecommunications

Due to intense competition telecommunications companies are constantly in a position of having to analyse and optimise their network infrastructures. This includes the regular analysis of the sort of technical data that commonly occur when using mobile phone services, such as voice calls, text messages, and data usage. The number of data records to be evaluated is generally around 10^9 to 10^{10}. The response times are somewhere in the range of 20 to 60 minutes. **DIMENSIO** was able to reduce these times to between just a few seconds and approx. two minutes, depending on the number of measurement data records.

Image data evaluation

These days, a wide range of applications and problems require databases to be searched for the purpose of finding identical or similar images or patterns. One such example would be the detection of biometric identifiers in the security sector. Images taken with a camera must be immediately compared with thousands, if not millions, of data records in a database, a task that is expected to take seconds. **DIMENSIO** reduces lookup times in this field of application by a factor of up to 10^4 = 10,000.

Post-processing, material certification and handling warranty claims

Material certification and the processing of warranty claims often means that products that left the production process a long time previously, need to be virtually analysed again in order to be able to recognise the individual components and to identify supplier/manufacturer data (post-processing). Such processes tend to be extremely time-consuming, often taking hours, as the normal production systems cannot be used for this purpose. In such cases, **DIMENSIO** increased speeds by a factor of approx. 10^3 = 1,000 - in many cases at greatly reduced cost in terms of the hardware required.

Sequential analysis in gene databases

The protein and gene sequences of thousands of different organisms are held in databases around the world. A comparison of these sequences enables, among other things, certain conclusions to be drawn concerning the function of human genes. Likewise, these data can be used for the purpose of basic research and applied development, i.e. of cancer medication. Even simple alignments in pairs tend to be extremely slow. However, multiple alignments, i.e. the simultaneous analysis of several sequences, take even longer. Once again, implementing **DIMENSIO** considerably enhanced performance. **DIMENSIO** can be used in this field by analogy with the full-text search and substantially increase performance speeds.

5. Economies of speed

First performance measurements with a university prototype and a market leading database system (source: Dissertation TU Chemnitz, O. Görlitz, 2004) showed that in a 10-dimension search space, more than **99%** of the otherwise necessary I/O operations could be dispensed with when compared to technology using secondary indexes. This figure was still around **85%** when compared to a system-specific composite key index[1].

Between 2009 and 2011 further tests (PoCs) were performed with improved prototypes, a few of which are listed below:

Within the framework of the statewide mapping of Saxony, the aim was to determine how well **DIMENSIO** performed in comparison to established multi-dimensional database extensions (in this case, the spatial extension of a large database manufacturer). Using a 50-metre grid of Saxony, each with 20 values per grid point (= 20 dimensions), the PoC performed involved some **300 million data records**. The objective was to achieve a minimum of 1,000 queries per second. The run times per query **without DIMENSIO** were approx. **0.15** seconds, which corresponds to 6 queries per second, and **with DIMENSIO** around **5 * 10^{-4}** seconds, which corresponds to approx. 2,000 queries per second, i.e. an increase factor of **300**!

A further test was performed using real industry data from the MES sector[2] (Manufacturing Execution System) via a table with **6.6 million data records** and the following SQL statement:

[1] The tests were performed on a database (on a Sun II Ultra Sparc Server) with 2.1 million 10-dimensional, randomly generated data records. The value range of each dimension spanned the interval [-10^4, 10^4]. The classification component produced 10^5 hyperspherical clusters/groups with a radius of 250. In order to enable a comparison with conventional methods of multi-dimensional indexing, the table was used to create two types of system-integrated secondary key indexes in classic primary key organisation: a so-called cascaded key for 10 dimensions (composite key index) and 10 individual secondary indexes. Seven rows were added to this configuration, each row containing 100 test queries with increasing range sizes (between 100 and 3000 response data records), whereby each query affected at least one group. The time-relevant number of I/O operations were measured (rounded).

[2] Further details can be found at: http://www.dimensio-informatics.com/html/di-ais.html

```
select count(*) from (
   select Attr-1,  count(*) Attr-2 , NVL(Attr-3, 0),NVL(Attr-4, 0)
   from Relation
   where Attr-5 = 0 and Attr-6 is not null
      and Datetime >= to_date('10.08.2009','dd.mm.yyyy')
      and Datetime <  to_date('11.08.2009' ,'dd.mm.yyyy')
      and Attr-7 is not null
   group by Attr-1, NVL(Attr-3, 0), NVL(Attr-4, 0))
```

Across all test runs, we were able to improve speeds by an average factor of **0.97 * 10^3** – i.e. a factor of almost **1,000**.

One of the companies that dimensio informatics GmbH serves as development partner is IBM. In this context, the IBM Innovation Center in Ehningen extended an invitation to test the **DIMENSIO** prototype in-house on an actual DB2 installation. The tests were designed to verify the speed advantage gained using **DIMENSIO** compared to normal access using DB2[3]. At the end of the test day, the average time for a query in DB2 for ad-hoc queries **with DIMENSIO** was **3%** of direct DB2 access (100%) and **68%** for statically bound queries (i.e. without prepare time).

6. Unique selling points

- High-dimensional index for any number of dimensions
- Interval definition through semantic data analysis and cluster formation using artificial neural networks
- Minimally-invasive integration in existing IT landscapes (integration in application, network or DBMS)
- Compatible with any data management system (not just databases, but also customised storage solutions and/or flat files)

7. Client benefits

- Data analysis up to 10,000 times faster compared to conventional solutions
- Enormous time savings enable optimised business processes
- Unrivalled cost savings, due to lower hardware costs compared to standard solutions and lower investments in additional CPU licenses for databases.
- Ongoing cost savings, starting with support costs, as the hardware and software are less complex, including maintenance fees, through to lower energy costs due to reduction in hardware and computing requirements

[3] The test database abstracted from the BI sector contained a relation with 998 columns (value range 0 or 1 respectively, i.e. property available or not) and approx. 12,800 data records. The test was performed on a system x 3755 with 4 x AMD Opteron 64 Dual Core 3.2 GHz and 12 GB RAM running Ubuntu 10.04 (64-bit). Version 9.7 FP2 was used for DB2. The test was performed using 50 point queries, which returned between one and 42 tuples.

- New options available for data evaluation, previously unacceptable to users due to long waiting times

8. dimensio informatics GmbH – the performers

dimensio informatics GmbH – the spirit of speed - is an independent, technology provider focused on solving performance problems. Together with our partners (ISVs, OEM partners, DBMS manufacturers, etc.) we offer products and IT services designed to analyse and accelerate business processes. We analyse performance bottlenecks and develop individual, minimally invasive solutions that build on proprietary products. dimensio informatics GmbH software products use innovative technologies, enable simple and cost-efficient integration in existing IT infrastructures and can be deployed with all standard business applications and all database management systems.

Retrospect – How it all began

DIMENSIO began in a university environment towards the end of the 80's when first attempts to realise a hardware-based method of indexing ended in failure. In the mid-90's, a new approach was taken, this time as a software-based solution.

In 1999, a diploma thesis based on this subject was awarded the university prize by Chemnitz University of Technology and in 2000 a first prototype of the indexing method, then named ICIx (Intelligent Cluster Index), was unveiled at CeBIT in Hanover. The continuation of this work over a number of years produced a stable university prototype.

In July 2010, after successful participation of the university team in various business plan competitions, dimensio informatics GmbH was founded as a spin-off of the Chemnitz University of Technology. Since October 2010, the company has enjoyed financial security thanks to funding from the early stage Venture Capital Fund "Technologiegründerfonds Sachsen" (TGFS) and since July 2013 it is part of the micData AG, based in Munich.

dimensio informatics GmbH
Brückenstr. 4
09111 Chemnitz
Germany

Telefon +49 371 26 20 19 0
Telefax +49 371 26 20 19 10
info@dimensio-informatics.com
www.dimensio-informatics.com