

INFOBRIGHT

Open Source
Data Warehousing



INFOBRIGHT

Community Edition

Technology White Paper

August 2009

Table of Contents

	Topic	Page
I.	Introduction	2
II.	Infobright Architecture	4
1.	Column Orientation	4
2.	Data Organization and the Knowledge Grid	5
3.	The Infobright Optimizer	6
4.	How IT Works: Resolving Complex Analytic Queries Without Indexes	7
5.	Data Loading and Compression	12
6.	How Infobright Leverages MySQL	14
III.	Rough Set Mathematics: The Underlying Foundation	15
IV.	Conclusion and the Next Steps	16

Infobright
47 Colborne Street, Suite 403
Toronto, Ontario M5E 1P8
Canada
www.infobright.com
www.infobright.org

I. Introduction

Over the past decade, business intelligence has emerged as one of the highest priority items on CIO agendas. Businesses and government agencies know that mining information from the increasingly large volumes of data they collect is critical to their business or mission.

During this same period, a number of other factors have contributed to the high rate of growth of business intelligence (BI) and data warehousing (DW) technologies including:

- Many more users with diverse needs
- Need for ad hoc queries vs. standard canned reports
- Need for more “real time” information
- Growth of the number of databases within an organization, with need for consolidation of information
- Rapidly growing volumes of data
- Growth of internet and web-based applications, including self-service applications
- Regulatory/legal requirements

Traditional approaches to data warehousing have significant drawbacks in terms of effectively delivering a solution to businesses for such diverse requirements. They were also designed primarily for OLTP, making them unsuited for the growing need for analytics and ad hoc queries. Significant other drawbacks include high licensing and storage cost, slow query performance against large data volumes, and difficulty in providing access to all of the data stored in multiple databases. At the same time, these growing issues are placing a high burden on DBAs and IT organizations to implement, tune, and manage databases that supply BI information.

Terabytes & the Traditional RDBMS

Traditional database technologies were not designed to handle massive amounts of data and hence their performance degrades significantly as volumes increase. Some of the reasons for this are:

- Disk I/O is the primary limiting factor. While the cost of disk has decreased, data transfer rates have not changed. Therefore accessing large structures such as tables or indexes is slow.
- A row oriented design forces the database to retrieve all column data regardless of whether or not is it required to resolve the query.

- As table size increases so do the indexes; ultimately even processing the indexes becomes slow. So as tables grow, the query speed decreases as more and more data must be accessed to resolve the query.
- Load speed degrades since indexes need to be recreated as data is added; this causes huge sorts (another very slow operation).

The ideal architecture to solve this problem must minimize disk I/O, access only the data required for a query, and be able to operate on the data at a much higher level to eliminate as much data as possible, as soon as possible.

Emergence and Growth of Open Source Software

The past several years have also been a time of rapid growth in the availability and deployment of open source software - from operating systems to middleware to applications. Open source delivers lower cost, liberally licensed software that opens up capabilities to even the smallest companies, but also opens up code and functionality to a community process that ensures solutions remained true to prevailing standards.

In the database market, MySQL's fast growth and expanding market share validated the need and suitability of an open source DBMS. MySQL, the world's most popular open source database, expanded the market for database systems, by making it available to a much broader set of users and for many new applications. Companies like Pentaho, JasperSoft, Talend and Actuate/BIRT have shown that there was a corresponding need for open source business intelligence solutions.

Now the open source revolution has reached the data warehousing space. There are not only tools and technologies available for the building blocks of data warehouse implementations but the data warehouse itself has been open sourced. Infobright Community Edition is an open source, column-oriented data warehouse designed for companies of all sizes with a need for analytics across a large data volume.

Infobright Community Edition (ICE)

ICE is an open source software solution designed to deliver a scalable data warehouse optimized for analytic queries. The innovation underlying Infobright technology is the application of Rough Set mathematics to database and data warehousing (for more information on Rough Set mathematics, see section III on page 15).

The architecture delivers the following key benefits:

- Ideal for data volumes up to 30TB
- Market-leading data compression (from 10:1 to over 40:1), which drastically reduces I/O (improving query performance) and results in significantly less storage than alternative solutions
- No licensing fees
- Fast response times for complex analytic queries
- Query and load performance remains constant as the size of the database grows
- No requirement for specific schemas, e.g. Star schema
- No requirement for materialized views, complex data partitioning strategies, or indexing
- Simple to implement and manage, requiring little administration
- Reduction in data warehouse capital and operational expenses by reducing the number of servers, the amount of storage needed and their associated maintenance costs, and a significant reduction in administrative costs
- Runs on low cost, off-the-shelf hardware
- Is compatible with major Business Intelligence tools from companies including Actuate, Pentaho, Jaspersoft, Cognos, Business Objects, and others.

II. Infobright Architecture

Infobright's architecture is based on the following concepts:

- **Column Orientation**
- **Data Packs and Data Pack Nodes**
- **Knowledge Nodes and the Knowledge Grid**
- **The Optimizer**
- **Data Compression**

1. Column Orientation

Infobright at its core is a highly compressed column-oriented data store,

which means that instead of the data being stored row by row, it is stored column by column. There are many advantages to column-orientation, including the ability to do more efficient data compression because each column stores a single data type (as opposed to rows that typically contain several data types), and allows compression to be optimized for each particular data type, significantly reducing disk I/O.

Most analytic queries only involve a subset of the columns of the tables and so a column oriented database focuses on retrieving only the data that is required. The advantage of column oriented databases was recognized by a Bloor research report, which said: "For much of the last decade the use of column-based approaches has been very much a niche activity. However ... we believe that it is time for columns to step out of the shadows to become a major force in the data warehouse and associated markets."¹

2. Data Organization and the Knowledge Grid

Infobright organizes the data into 3 layers:

- [Data Packs](#)

The data itself within the columns is stored in 65,536 item groupings called Data Packs. The use of Data Packs improves data compression since they are smaller subsets of the column data (hence less variability) and the compression algorithm can be applied based on data type.

- [Data Pack Nodes \(DPNs\)](#)

Data Pack Nodes contain a set of statistics about the data that is stored and compressed in each of the Data Packs. There is always a 1 to 1 relationship between Data Packs and DPNs. DPN's always exist, so Infobright has some information about all the data in the database, unlike traditional databases where indexes are created for only a subset of columns.

- [Knowledge Nodes](#)

These are a further set of metadata related to Data Packs or column relationships. They can be more introspective on the data, describing ranges of value occurrences, or can be extrospective, describing how they relate to other data in the database. Most KN's are created at load time,

¹ Philip Howard, "What's Cool About Columns", Bloor Research, March 2008.

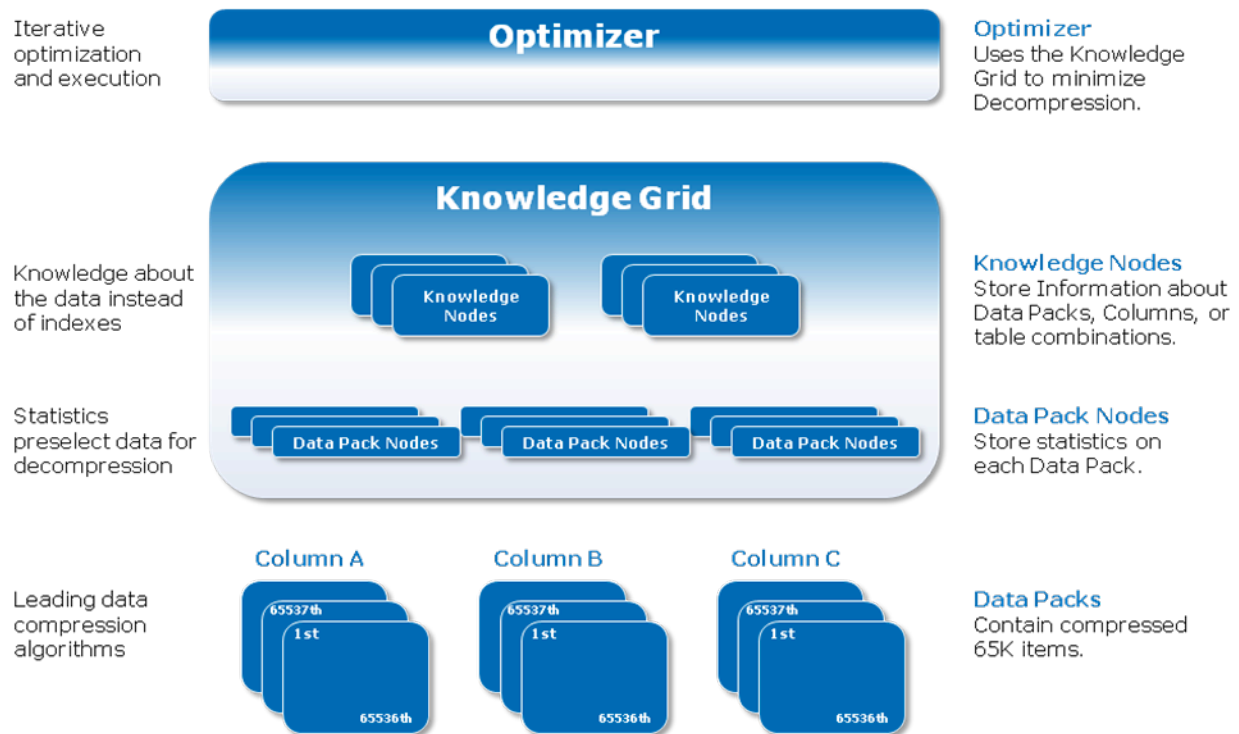
but others are created in response to queries in order to optimize performance. This is a dynamic process, so certain Knowledge Nodes may or may not exist at a particular point in time.

The DPNs and KNs form the **Knowledge Grid**. Unlike traditional database indexes, they are not manually created, and require no ongoing care and feeding. Instead, they are created and managed automatically by the system. In essence, they create a high level view of the entire content of the database.

3. The Infobright Optimizer

The Optimizer is the highest level of intelligence in the architecture. It uses the Knowledge Grid to determine the minimum set of Data Packs, which need to be decompressed in order to satisfy a given query in the fastest possible time.

Figure 1 – Architecture Layers



4. How it Works: Resolving Complex Analytic Queries without Indexes

The Infobright data warehouse resolves complex analytic queries without the need for traditional indexes. The following sections describe the methods used.

Data Packs

As noted, Data Packs consist of groupings of 65,536 items within a given column.

For example, for the table T with columns A, B and 300,000 records, Infobright would have the following Packs:

Pack A1: values of A for rows no. 1-65,536	Pack B1: values of B for rows no. 1-65,536
Pack A2: values of A for 65,537-131,072	Pack B2: values of B for 65,537-131,072
Pack A3: values of A for 131,073-196,608	Pack B3: values of B for 131,073-196,608
Pack A4: values of A for 196,609-262,144	Pack B4: values of B for 196,609-262,144
Pack A5: values of A for 262,145-300,000	Pack B5: values of B for 262,145-300,000

The Knowledge Grid

The Infobright Knowledge Grid includes Data Pack Nodes and Knowledge Nodes.

As an example of a DPN for numeric data types; min value, max value, sum, number of non-null elements and count are stored. Each Data Pack has a corresponding DPN.

For example, for the above table T, assume that both A and B store some numeric values. For simplicity, assume there are no null values in T; hence Infobright can omit information about the number of non-nulls in DPNs.

The following table should be read as follows:

for the first 65,536 rows in T the minimum value of A is 0, maximum is 5, the sum of values on A for the first 65,536 rows is 100,000 (and there are no null values).

Pack Numbers Columns A & B	DPNs of A			DPNs of B		
	Min	Max	Sum	Min	Max	Sum
Packs A1 & B1	0	5	10,000	0	5	1,000
Packs A2 & B2	0	2	2,055	0	2	100
Packs A3 & B3	7	8	500,000	0	1	100
Packs A4 & B4	0	5	30,000	0	5	100
Packs A5 & B5	-4	10	12	-15	0	-40

DPNs are accessible without the need to decompress the corresponding Data Packs. Whenever Infobright looks at using data stored in the given Data Pack it first examines its DPN to determine if it really needs to decompress the contents of the Data Pack. In many cases, information contained in a DPN is enough to optimize and execute a query.

DPNs alone can help minimize the need to decompress data when resolving queries, however Infobright’s technology goes beyond DPNs to also include what is called Knowledge Nodes.

Knowledge Nodes (KNs) were developed to efficiently deal with complex, multiple-table queries (joins, sub-queries, etc.). Knowledge Nodes store more advanced information about the data interdependencies found in the database, involving multiple tables, multiple columns, and single columns. It enables precise identification of Data Packs involved and minimizes the need to decompress data.

To process multi-table join queries and sub-queries, the Knowledge Grid uses multi-table Pack-To-Pack KNs that indicate which pairs of Data Packs from different tables should actually be considered while joining the tables. The query optimization module is designed in such a way that Pack-To-Pack Nodes can be applied together with other KNs.

Simple statistics such as min and max within Data Packs can be extended to include more detailed information about occurrence of values within particular value ranges. In these cases a histogram is created that maps

occurrences of particular values in particular Data Packs to determine quickly and precisely the chance of occurrence of a given value, without the need to decompress the Data Pack itself. In the same way, basic information about alphanumeric data is extended by storing information about the occurrence of particular characters in particular positions in the Data Pack.

KN's can be compared to indexes used by traditional databases, however, KN's work on *Packs* instead of *rows*. Therefore KNs are 65,536 times smaller than indexes (or even 65,536 times 65,536 for the Pack-To-Pack Nodes because of the size decrease for each of the two tables involved). In general the overhead is around 1% of the data, compared to classic indexes, which can be 20-50% of the size of the data.

Knowledge Nodes are created on data load and may also be created during query. They are automatically created and maintained by the Knowledge Grid Manager based on the column type and definition, so no intervention by a DBA is necessary.

The Infobright Optimizer

How do Data Packs, DPNs and KN's work together to achieve high query performance?

Decompressing Data Packs is incomparably faster than decompressing larger portions of data. By having a good mechanism for identifying the Packs to be decompressed, Infobright achieves a high query performance over the partially compressed data. The Optimizer uses the Knowledge Grid to determine the minimum set of Data Packs which need to be decompressed in order to satisfy a given query in the fastest possible time. In some cases, the information contained in the Knowledge Grid is sufficient to resolve the query, in which case nothing is decompressed.

The Optimizer applies DPNs and KNs for the purpose of splitting Data Packs among the three following categories for every query coming into the optimizer:

- *Relevant Packs* – in which each element (the record's value for the given column) is identified, based on DPNs and KNs, as applicable to the given query.
- *Irrelevant Packs* – based on DPNs and KNs, the Pack holds no relevant values.
- *Suspect Packs* – some elements may be relevant, but there is no way to claim that the Pack is either fully relevant or fully irrelevant,

based on DPNs and KNs.

While querying, Infobright does not need to decompress either Relevant or Irrelevant Data Packs. Irrelevant Packs are simply not taken into account at all. In case of Relevant Packs, Infobright knows that all elements are relevant, and the required answer is obtainable.

For example, using our previously discussed DPNs for Table T (below) consider the following SQL query statement:

Query 1: SELECT SUM(B) FROM T WHERE A > 6;

Pack Numbers Columns A & B	DPNs of A			DPNs of B		
	Min	Max	Sum	Min	Max	Sum
Packs A1 & B1	0	5	10,000	0	5	1,000
Packs A2 & B2	0	2	2,055	0	2	100
Packs A3 & B3	7	8	500,000	0	1	100
Packs A4 & B4	0	5	30,000	0	5	100
Packs A5 & B5	-4	10	12	-15	0	-40

It can be seen that:

Packs A1, A2, A4 are Irrelevant – none of the rows can satisfy $A > 6$ because all these packs have maximum values below 6. Consequently, Packs B1, B2, B4 will not be analyzed while calculating SUM(B) – they are Irrelevant too.

Pack A3 is Relevant – all the rows with numbers 131,073-196,608 satisfy $A > 6$. It means Pack B3 is Relevant too. The sum of values on B within Pack B3 is one of the components of the final answer. Based on B3's DPN, Infobright knows that that sum equals to 100. And this is everything Infobright needs to know about this portion of data.

Pack A5 is Suspect – some rows satisfy $A > 6$ but it is not known which ones. As a consequence, Pack B5 is Suspect too. Infobright will need to decompress both A5 and B5 to find, which rows out of 262,145-300,000 satisfy $A > 6$ and sum up together the values over B precisely for those rows. A result will be added to the value of 100 previously obtained for Pack B3, to form the final answer to the query.

The Suspect Packs can change their status during query execution and become Irrelevant or Relevant based on intermediate results obtained from

other Data Packs.

For another example, consider the following, just slightly modified SQL query:

Query 2: SELECT MAX(B) FROM T WHERE A > 6;

Comparing to Query 1, the split among Relevant, Irrelevant, and Suspect Data Packs does not change. From DPN for Pack B3 Infobright knows that the maximum value on B over the rows 131,073-196,608 equals 1. So, it is already sure that at least one row satisfying $A > 6$ has the value equal to 1 on B. From DPN for Pack B5 it knows that the maximum value on B over the rows 262,145-300,000 equals to 0. So, although some of those rows satisfy $A > 6$, none of them can exceed the previously found value of 1 on B. Therefore Infobright knows that the answer to the above Query 2 is equal to 1 without decompressing any Packs for T.

To summarize, the above Data Pack splitting can be modified during the query execution, with the percentage of Suspect Packs decreasing. The actual workload is still limited to the suspect data. This method of optimization and execution is entirely different from other databases as Infobright technology works iteratively with portions of compressed data.

As another example, consider the usage of Pack-To-Pack Nodes. Pack-To-Pack Nodes are one of the most interesting components of the Knowledge Grid.

Query 3: SELECT MAX(X.D) FROM T JOIN X ON T.B = X.C WHERE T.A > 6;

Table X consists of two columns, C and D. The above query asks to find the maximum value on column D in table X, but only for such rows in X, for which there are rows in table T with the same value on B as the value on C in X, and with the value on A greater than 6. Table X consists of 150,000 rows with 3 Data Packs for each column. Its DPNs are presented below, in the same way as for table T. (And no null values assumed.)

Pack Numbers Columns C & D	DPNs of C			DPNs of D		
	Min	Max	Sum	Min	Max	Sum
Packs C1 & D1	-1	5	100	0	5	100
Packs C2 & D2	-2	2	100	0	33	100
Packs C3 & D3	-7	8	1	0	8	100

As in the case of the previous queries, Data Packs to be analyzed from the perspective of table T are A3, A5, B3, B5. Precisely, the join of tables X and T with additional condition $T.A > 6$ may involve only the rows 131,073-196,608 and 262,145-300,000 on the table T's side. Assume that the following Pack-To-Pack Knowledge Node is available for the values of B in table T and C in table X. "T.Bk" refers to Pack Bk in T, where k is the Pack's number. Hence, e.g., it can be seen that there are no pairs of the same values from T.B1 and X.C1 (represented by 0) and there is at least one such pair for T.B1 and X.C2 (represented by 1).

	T.B1	T.B2	T.B3	T.B4	T.B5
X.C1	0	1	0	1	0
X.C2	1	0	0	0	1
X.C3	1	1	0	1	0

It can be seen that while joining tables T and X on the condition $T.B = X.C$ Infobright does not need to consider Pack B3 in T any more because its elements do not match with any of the rows in table X. The only Pack left on T's side is B5. B5 can match only with the rows from Pack C2 in table X. Therefore, Infobright needs to analyze only the values of column D over the rows 65,537-131,072 in table X. It is not known exactly which out of those rows satisfy $T.B = X.C$, with additional filter $T.A > 6$ on T's side, but the final result will be at most 33 based on DPN. In order to calculate the exact answer, Packs B5 and C2, as well as A5 must first be decompressed to get precise information about the rows, which satisfy the conditions. Then, Pack D3 is decompressed in table X to find D's maximum value over those previously extracted row positions.

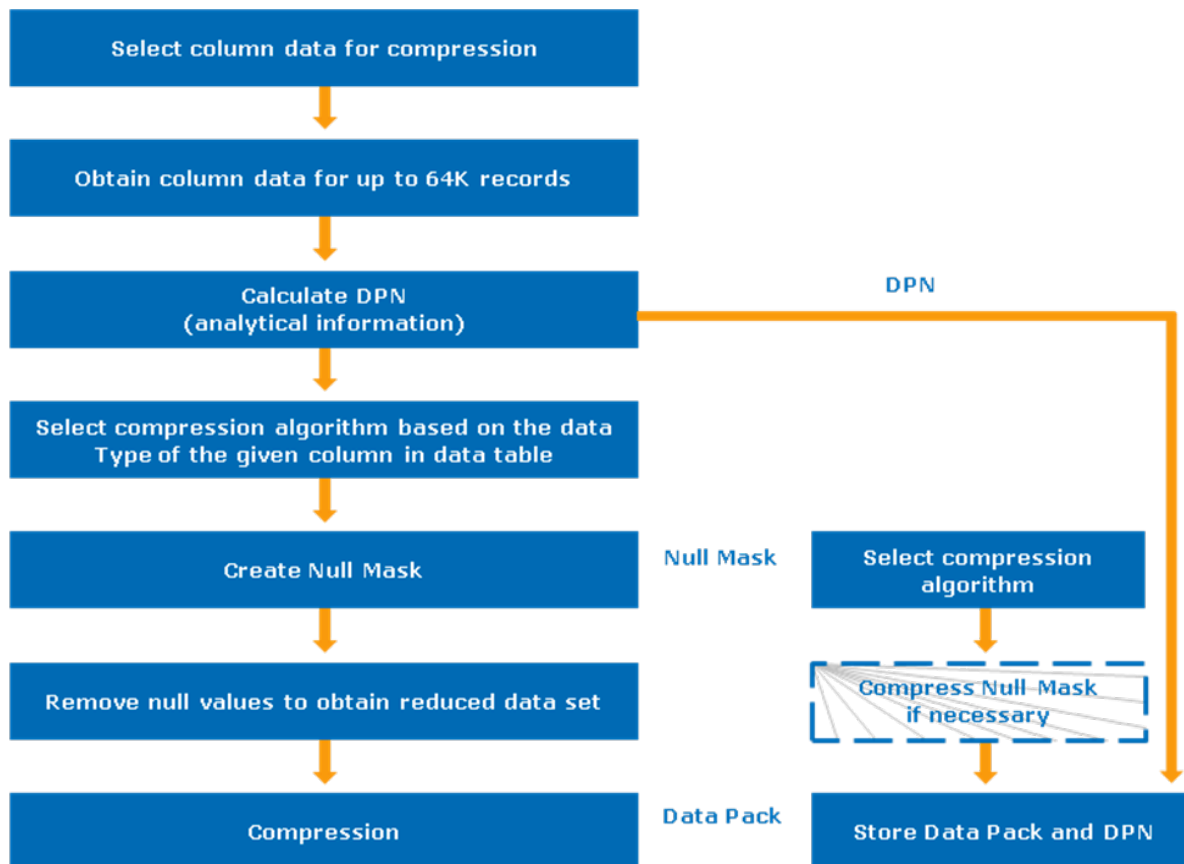
5. Data Loading and Compression

The mechanism of creating and storing Data Packs and their DPNs is illustrated below.

During load, 65,536 values of the given column are treated as a sequence with zero or more null values occurring anywhere in the sequence.

Information about the null positions is stored separately (within the *Null*

Mask). Then the remaining stream of the non-null values is compressed, taking full advantage of regularities inside the data.



Managing large volumes of data continue to be a problem for growing organizations. Even though the costs of data storage have declined, data growth continues to increase at a very rapid rate, with TDWI Research stating that the average data growth was 33-50% in 2006 alone.

At the same time disk transfer rates have not increased significantly, with more disks creating additional problems, such as expanding back-up times, and additional management overhead.

The reality is that data compression can significantly reduce the problems of rapidly expanding data. With Infobright, a 1TB database becomes a 100GB database. Since the data is much smaller the disk transfer rate is improved (even with the overhead of compression).

One of Infobright's benefits is its industry-leading compression. Unlike traditional row-based data warehouses, data is stored by column, allowing compression algorithms to be finely tuned to the column data type. Moreover, for each column, the data is split into Data Packs with each

storing up to 65,536 values of a given column. Infobright then applies a set of patent-pending compression algorithms that are optimized by automatically self-adjusting various parameters of the algorithm for each data pack.

Compression Results

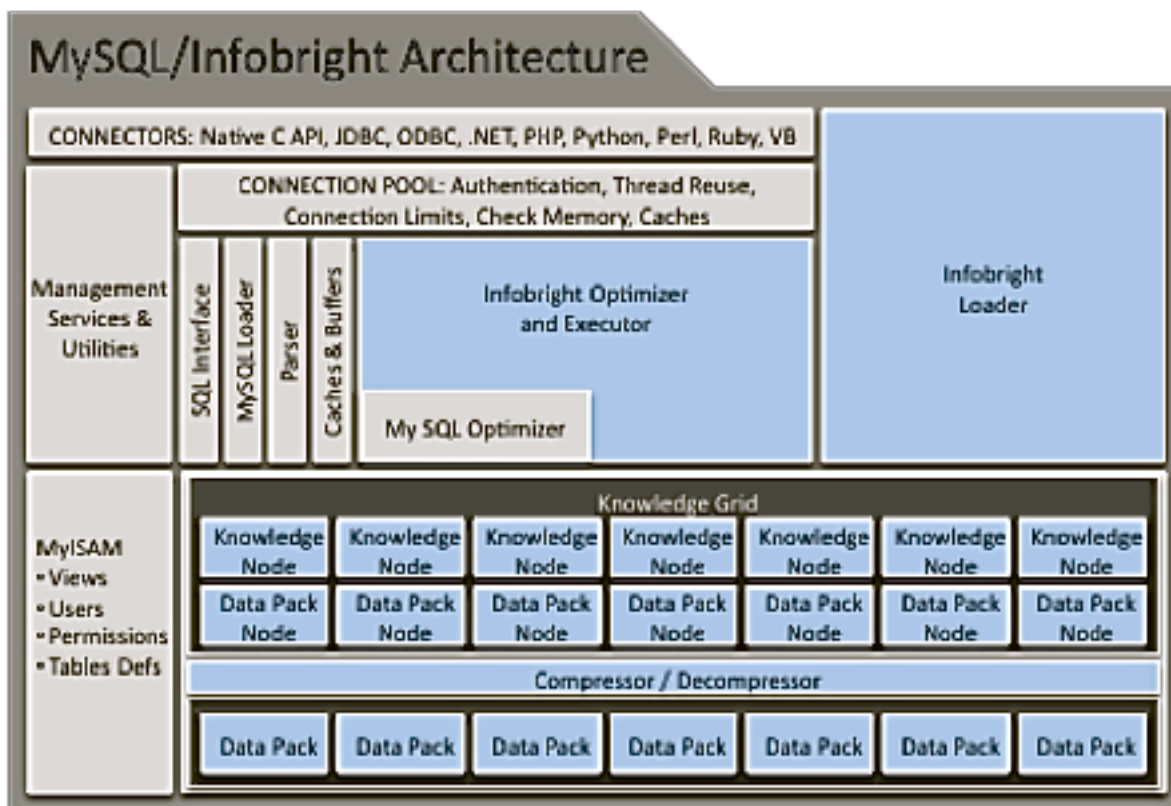
An average compression ratio of 10:1 is achieved in Infobright. For example 10TB of raw data can be stored in about 1TB of space on average (including the overhead associated with Data Pack Nodes and the Knowledge Grid). Other database systems increase the data size because of the additional overhead required to create indexes and other special structures, in some cases by a factor of 2 or more.

Within Infobright, the compression ratio may differ depending on data types and content. Additionally, some data may turn out to be more *repetitive* than others and hence compression ratios can be as high as 40:1.

6. How Infobright Leverages MySQL

In the data warehouse marketplace, the database must integrate with a variety of drivers and tools. By integrating with MySQL, Infobright leverages the extensive driver connectivity provided by MySQL connectors (C, JDBC, ODBC, .NET, Perl, etc.). MySQL also provide cataloging functions, such as table definitions, views, users, permissions, etc. These are stored in a MyISAM database.

Although MySQL provides a storage engine interface which is implemented in Infobright, the Infobright technology has its own advanced Optimizer. This is required because of the column orientation, but also because of the unique Knowledge Node technology described above.



III. Rough Set Mathematics: the Underlying Foundation

The innovation underlying Infobright technology, the secret sauce of the optimizer, is the application of Rough Set mathematics to database and data warehousing. The theory of rough sets has been studied and in use since the 1980s. Based on a pair of sets which give the *lower* and *upper* approximation of standard sets, it was used as a formal methodology to deal with vague concepts.

For example, rough set theory would be useful for determining a banking transaction's risk of being a fraud, or an image's likelihood to match a certain pattern. In these applications, the approximations resulting from the underlying data analysis were implicitly assumed to be put together with the domain experts' knowledge, in order to result in precise decisions.

In the early 2000s, a team of Polish mathematicians, graduates of Warsaw University, began to make important new advances by going beyond the original applications of the theory of rough sets. Their work sought to replace an individual's domain expertise with domain-specific massive computational routines developed from the knowledge of the data itself. The result is the ability to programmatically deliver precise results much faster by following rough guidelines pre-learned from the data as an initial step.

Consequently, a *new hybrid* approach to interpreting and applying the original foundations of rough sets is taking hold, one that blends the power of precision with the flexibility of approximations.

The Warsaw team then considered the above idea as the means for a new database engine, particularly targeted at applications where precision is important, but the datasets are too large to get carefully reviewed in their entirety. Using the pre-learned rough guidelines as a means to quickly approximate the sets of records matching database queries allows computational power to focus on precise analysis of only relatively small portions of data. It led to the creation of a new company in 2005, Infobright, and the development of a markedly new technology to address the growing need for business analytics across large volumes of data.

IV. Conclusion and Next Steps

Infobright has developed an architecture designed for analytic data warehousing, which also requires a lot less work by IT organizations, delivering faster time-to-market for analytic applications. The open source Infobright Community Edition makes it affordable for companies of all sizes. With an InformationWeek article finding that almost half of IT directors are prevented from rolling out the BI initiatives they desire because of expensive licensing and specialized hardware costs, Infobright can help by supplying the low TCO that IT executives are looking for.

For more information on Infobright Community Edition go to Infobright.org or read the Forums: www.infobright.org/forums or download the software at www.infobright.org/download

Infobright is a trademark of Infobright Inc. All names referred to are trademarks or registered trademarks of their respective owners.