Fundamenta Informaticae 127 (2013) 445–459
DOI 10.3233/FI-2013-920
IOS Press

Two Database Related Interpretations of Rough Approximations: Data Organization and Query Execution

Dominik Ślęzak[*]
Institute of Mathematics, University of Warsaw
ul. Banacha 2, 02-097 Warsaw, Poland
and
Infobright Inc.
ul. Krzywickiego 34 lok. 219, 02-078 Warsaw, Poland
`slezak@{mimuw.edu.pl,infobright.com}`

Piotr Synak, Arkadiusz Wojna, Jakub Wróblewski
Infobright Inc.
ul. Krzywickiego 34 lok. 219, 02-078 Warsaw, Poland
`{synak,arek.wojna,jakubw}@infobright.com`

Dedicated to Professor Andrzej Skowron, our Scientific Father, on His 70th birthday

Abstract. We present analytic data processing technology derived from the principles of rough sets and granular computing. We show how the idea of approximate computations on granulated data has evolved toward complete product supporting standard analytic database operations and their extensions. We refer to our previous works where our query execution algorithms were described in terms of iteratively computed rough approximations. We explain how to interpret our data organization methods in terms of classical rough set notions such as reducts and generalized decisions.

Keywords: Analytic Data Processing Systems, Rough-Granular Computational Models

[*]Address for correspondence: Institute of Mathematics, University of Warsaw, ul. Banacha 2, 02-097 Warsaw, Poland

1. Introduction

The theory of rough sets provides the means for handling incompleteness and uncertainty in large data sets [10]. Out of many rough set inspired approaches it is particularly worth mentioning a variety of algorithms for attribute subset selection [19] and, more generally, construction of possibly simplest models representing meaningful knowledge derived from data [9]. Another aspect of rough set applications refers to introducing more flexibility into standard computational methodologies. One can regard rough clustering [13] and rough database models [2] as being representative for this category.

In this paper, we follow the path related to rough set inspired computational models. We discuss how to utilize rough set principles in order to support scalable analytic data operations. We consider the example of Infobright's data processing system, where rough approximations are calculated on granulated data in order to quickly identify only those portions of information which are truly required to resolve incoming `SQL` statements [18]. We also discuss how rough set techniques can assist in partitioning data into blocks with descriptions providing improved efficiency of approximations [15].

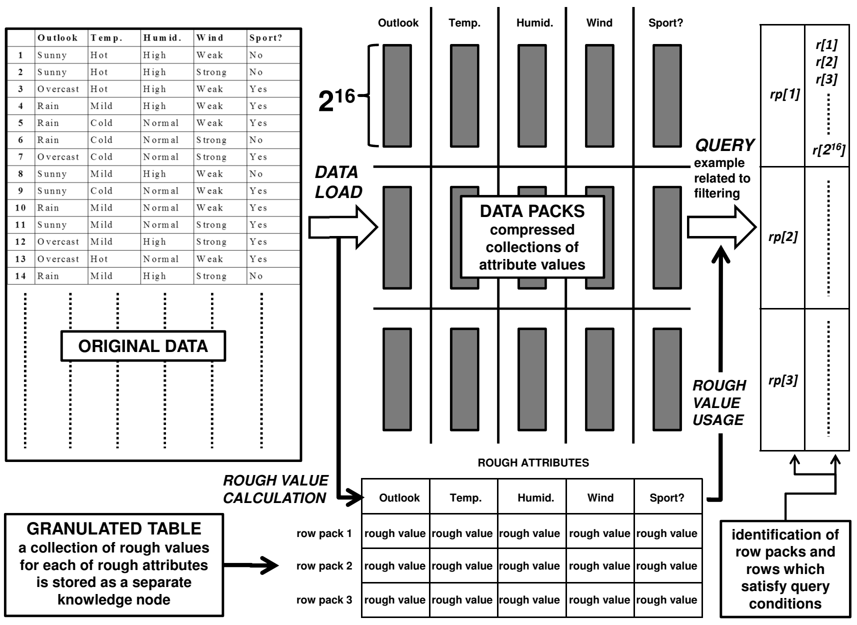
One can treat our technology as industry realization of rough-granular approach to scalable analytics [12]. The idea is to decompose data onto granules, create their higher level summaries, do approximate computations on those summaries and, whenever there is no other way to finish calculations, access detailed content of some of granules. Rough sets are useful at each of those stages. We adopt also some other modern database and data mining techniques such as columnar stores [6], data compression [20], adaptive querying [3] and stream clustering [1]. However, the layer of approximation algorithms using granulated data descriptions remains the most important ingredient of our architecture.

Advantages of rough-granular framework discussed in this paper are particularly significant in applications requiring ad-hoc querying against massive, rapidly growing data sets. Such query workloads are specific for, e.g., web analytics, financial services and telecommunications, where there is a need to handle petabytes of machine-generated data streams produced by various types of mobile devices and online processes.[1] Moreover, those areas of applications will have to rely, to an increasing extent, on non-exact query models. Thus, when referring to scalability of analytic computations one should think about accelerating both standard `SQL` operations and their approximate counterparts [8].

The ideas presented in next sections are partially collected from our previous publications, with some additions related to new rough set interpretations of our data organization process. Indeed, our rough set inspired model of resolving analytic queries is already documented [16]. On the other hand, connections between fundamental rough set notions and Infobright's approach to creation of possibly most efficient data granules and their higher level descriptions is not so well known yet. We regard such connections as extremely important for further improvements of our technology. We also believe that they may help in development of other rough-granular data processing and mining approaches in future.

The paper is organized as follows: In Section 2, we outline correspondence between our mechanisms of identifying data required for query execution and classical model of rough approximations. In Section 3, we provide some case studies illustrating dynamic, iterative and adaptive techniques of creating and utilizing granular data descriptions. In Section 4, we concentrate on the stage of data organization and show some interesting links between the problem of decomposing data onto blocks with possibly most informative descriptions and the problem of selecting subsets of attributes producing the most precise values of generalized decision functions [14]. In Section 5, we briefly conclude the paper.

[1] `www.infobright.com/index.php/customers-partners/customers/`



	Outlook	Temp.	Humid.	Wind	Sport?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cold	Normal	Weak	Yes
6	Rain	Cold	Normal	Strong	No
7	Overcast	Cold	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cold	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

	Outlook	Temp.	Humid.	Wind	Sport?
row pack 1	rough value	rough value	rough value	rough value	rough value
row pack 2	rough value	rough value	rough value	rough value	rough value
row pack 3	rough value	rough value	rough value	rough value	rough value

ORIGINAL DATA
Outlook
Temp.
Humid.
Wind
Sport?
2^{16}
DATA LOAD
DATA PACKS compressed collections of attribute values
QUERY example related to filtering
ROUGH VALUE USAGE
rp[1]
r[1]
r[2]
r[3]
r[2^{16}]
rp[2]
rp[3]
ROUGH ATTRIBUTES
ROUGH VALUE CALCULATION
GRANULATED TABLE a collection of rough values for each of rough attributes is stored as a separate knowledge node
identification of row packs and rows which satisfy query conditions

Figure 1. Loading and querying in Infobright's analytic data processing system.

2. Infobright and Rough Approximations

We combine the principles of columnar databases [20] and information granulation [12] in order to efficiently store and query relational data. During load to a given data table, input rows are organized into row packs. Each row pack corresponds to a collection of so called data packs. Each data pack corresponds to a collection of values of a single attribute. In other words, we partition data into row packs and each row pack consists of data packs – one data pack per attribute.

Data packs are compressed in a lossless way and stored on disk. Prior to compression, various forms of useful mathematical descriptions of data pack contents are calculated. Descriptions may contain information such as minimum and maximum values occurring within a data pack (interpreted in different ways for different types of attributes), sum of values (or, e.g., total length of string values), number of (non-)`null` values, histograms of value distributions (spanned between specific minimum and maximum values for each of data packs), maps of occurrence of characters or tokens in string data packs, etc. [17]. We refer to all such compacted descriptions of data packs as to rough values.

Figure 1 shows how data packs and rough values are assembled. Rough values corresponding to a given data table are stored in its corresponding granulated table. Attributes of granulated tables are called rough attributes. Rows of granulated tables correspond to row packs in original tables. Their values over rough attributes take a form of more or less complex rough values. We refer to metadata layer containing granulated tables as to a kind of knowledge grid describing overall data content. Collections of rough values of particular rough attributes are stored in so called knowledge nodes.

There are also other metadata layers responsible for interpreting the contents of data packs and rough values. They can contain, e.g., dictionaries for attributes with relatively low number of distinct values. The content of each of data packs corresponding to such low cardinality attribute is modified prior to

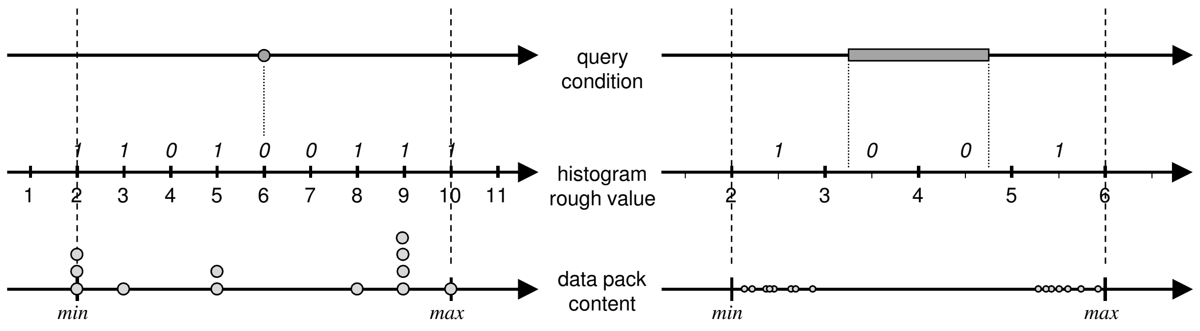


Figure 2. Histogram related rough value components for low cardinality (left) and numeric (right) attributes.

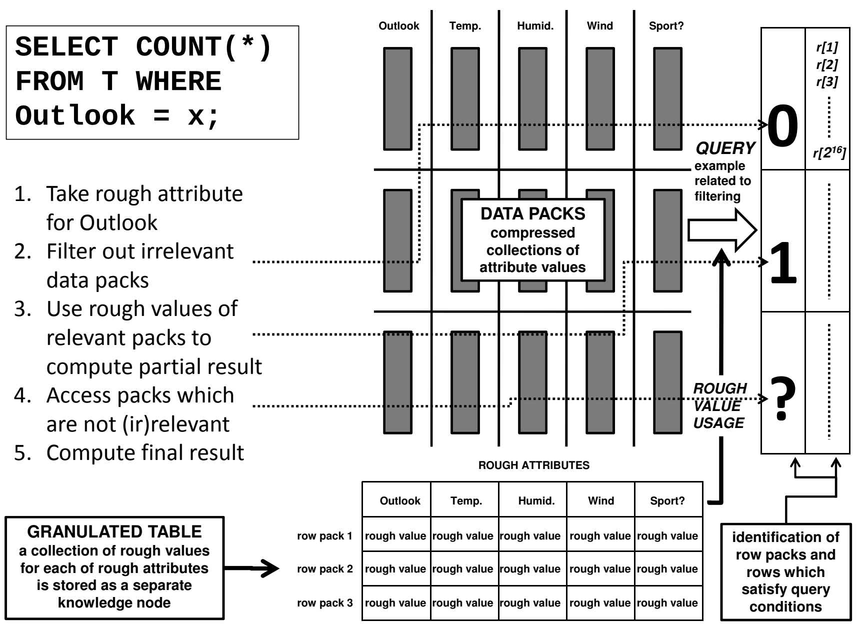
compression by replacing original values with their integer codes. Codes are also utilized when building rough values. For illustration, let us consider an attribute with 1,000 distinct values. Let us take a look at above mentioned histogram components of rough values. In default setting, they represent binary information about occurrence of values within 2^{10} intervals partitioning domains of non-`null` elements of the corresponding data packs. In case of a low cardinality attribute, as shown in Figure 2(left), histograms provide precise information about value codes occurring in particular data packs. Obviously, the dictionary is still necessary to translate those codes into original values.

Rough values may take far more complex forms than in the example above. However, all their types share common functionality – they provide compact data descriptions utilized by our heuristic algorithms to minimize and optimize data pack access. Let us note that typical columnar stores usually focus on accelerating attribute (or attribute index) scanning. In other words, they assume that each data block (page, pack, etc.) corresponding to attributes involved in a given query may need to be accessed. On the other hand, higher level descriptions of data blocks can provide sufficient information to exclude some of them from processing, which leads to a decrease of the scanning effort.

Figures 1 and 2 also illustrate the most basic way of using rough values, i.e., excluding data packs that do not satisfy a given `SQL` condition. Only not excluded packs need to be decompressed and examined value by value. Filtering results obtained for single rows and whole row packs are passed to further stages of query execution in form of boolean filters. In case of `WHERE` clauses including conditions over multiple attributes, the corresponding filters can be merged at both levels.

Let us consider `BETWEEN` condition over a numeric attribute. We can utilize minimum/maximum and histogram descriptions to identify data packs with no items satisfying this condition. If minimum-maximum interval of a given data pack does not overlap with the one defined by `BETWEEN` constraints, then we can exclude it immediately. As illustrated by Figure 2(right), sometimes we can do it also for the condition interval dropping between minimum and maximum values.

Similarly, we can handle other filters (e.g.: `LIKE` for `varchar` attributes [17]). Rough values can assist also in excluding data packs while resolving other parts of `SQL`, such as joins, subqueries, etc. [16]. However, data exclusion is not the only way to employ rough values. Let us continue with an example of `BETWEEN` condition. Let us assume that we want to count all rows in table `T` with values between `x` and `y` on attribute `a`. If there is a data pack with minimum greater than `x` and maximum lower than `y`, then we can utilize information about the number of its elements and the fact that all of them meet considered condition. Figure 3 presents this situation for the case of equality condition.



```
SELECT COUNT(*)
FROM T WHERE
Outlook = x;
```

1. Take rough attribute for Outlook
2. Filter out irrelevant data packs
3. Use rough values of relevant packs to compute partial result
4. Access packs which are not (ir)relevant
5. Compute final result

Outlook
Temp.
Humid.
Wind
Sport?
DATA PACKS
compressed collections of attribute values
QUERY
example related to filtering
0
1
?
r[1]
r[2]
r[3]
r[2^16]
ROUGH VALUE USAGE
ROUGH ATTRIBUTES
GRANULATED TABLE
a collection of rough values for each of rough attributes is stored as a separate knowledge node
row pack 1
row pack 2
row pack 3
rough value
identification of row packs and rows which satisfy query conditions

Figure 3. Handling relevant, irrelevant and suspect data packs in query resolving.

One can find many other cases where it is enough to use data descriptions while executing typical SELECT statements. Therefore, let us operate with the following three categories:

- Relevant data packs, where all of data elements are recognized to be relevant for further processing
- Irrelevant data packs, which are recognized to have no data elements relevant for further processing
- Suspect data packs, which are not recognized as either relevant or irrelevant

There is an analogy between the three categories above and the three types of rough set regions [10]. Relevant, irrelevant and suspect data packs clearly correspond to positive, negative and boundary regions, respectively. This analogy is even more visible for row packs, which are comparable to so called indiscernibility classes in approximation spaces [11]. In case of already mentioned queries with multidimensional filters, we can apply simple three-valued logic taking into account both relevant and irrelevant data pack classifications while evaluating boolean combinations of single attribute conditions. For example, if one of disjunction components corresponds to a data pack recognized as relevant, then the corresponding row pack becomes relevant with respect to the whole disjunction.

In order to better understand discussed correspondence, let us recall that rough set regions are usually employed to approximate various concepts of interest (e.g.: predefined decision classes or unions of decision classes) in decision support and knowledge discovery applications [19]. In case of our framework, in their simplest form discussed in this section, the concepts of interest correspond to subsets of rows satisfying some query conditions. Therefore, approximated concepts can be totally different for different SQL statements. Moreover, as one will be able to see in next sections, approximated concepts can dynamically evolve during query execution. Nevertheless, the fundamental idea of constructing rough approximations basing on (currently) available information about data is the same.

3. Query Resolving via Rough Computing

Before we continue with more advanced aspects of our approach, let us emphasize that Infobright is just one of many examples of connections between rough sets and data processing systems. With respect to foundations, it is important to pay attention to the notion of discernibility which is crucial for defining functional dependencies and keys in relational databases, as well as various kinds of reducts in rough set framework [9]. Going further, it is not so well known that subsets of attributes preserving values of generalized decision functions [14] are in strong correspondence with so called multi-valued dependencies introduced in [5] to express more advanced normal forms of relational data models.

In [2], rough approximations of results of relatively basic `SELECT` statements are constructed over data tables with interval values. Operations on such imprecise data sets could be regarded as analogous to our algorithms working with rough values residing in granulated tables. As mentioned later in this section, we also implemented some approximate query methods based on rough values. However, it turns out that for both precise and approximate queries, it is better to rely on computational interactions between rough value and exact data layers rather than using only granulated tables.

As an illustration, let us go back to `WHERE` clauses involving multiple attributes. Consider conjunction of two single attribute conditions. Imagine that for a given row pack its data packs corresponding to both attributes are suspect. Let us note that a suspect data pack may turn out to contain no relevant elements after its decompression. Such a false suspect classification usually results from limited resolution of rough values. For example, for condition `a=x` over numeric attribute `a`, minimum/maximum and histogram descriptions of a given data pack may not let identify it as irrelevant, even if it does not contain any values equal to `x`. In such a case, for considered conjunction, it can happen that rough categorization of one of data packs will change after accessing the other one. Namely, if one of data packs turns out to be false suspect, then the whole row pack becomes irrelevant. Suspect data packs may also turn out to be relevant, leading to re-classification of other data packs in a number of scenarios.

The above considerations show that derivation of rough set regions interpreted in terms of relevant, irrelevant and suspect data packs can dynamically change while resolving a single query. Moreover, it is worth paying more attention to types of information that can influence such derivation. In the previous section, we implicitly assumed that data packs can be recognized as relevant or irrelevant basing on analyzing their corresponding rough values. However, as we can see further in this section, rough values can be assigned also to other structures than data packs corresponding to physical columns. Moreover, available information about data packs can change in time. Actually, one can interpret operation of decompressing and accessing a given data pack as achieving its complete characteristics, which may be used together with rough values of other data pieces in further processing stages.

Another question refers to more thorough specification of concepts that we really want to approximate while resolving queries. In Section 2 we claimed that analogy between our data pack categorizations and classical rough set regions is particularly visible in case of approximating sets of rows satisfying some query conditions. However, next examples illustrate that we can also approximate contents of various intermediate structures and partial query results. Generally, one can say that we approximate information that is sufficient to finalize a given computation. Information is provided at both data pack content and data pack description levels. However, in order to deal with large data volumes, we assume direct access only to that latter level. If we had unlimited access to information at both levels, we would be theoretically able to work with a minimum subset of (meta)data entries required to resolve a query. However, we can only work with iteratively refined approximation of that subset.

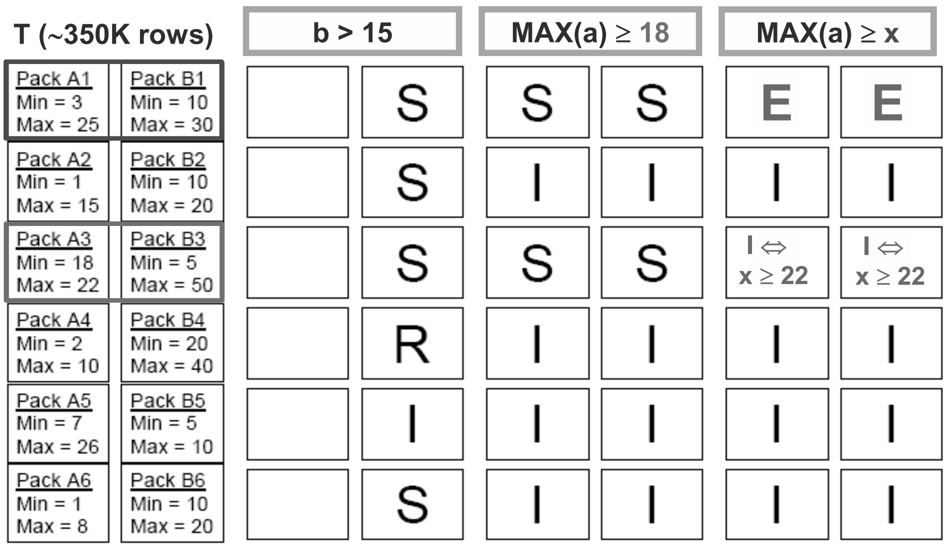


Figure 4. Computation of `SELECT MAX(a) FROM T WHERE b>15;` [18].

The following case study was introduced in [18] as an illustration for adaptive query processing based on iterative work with rough values and exact value-by-value computations. By adaptive query processing one usually means adjusting a plan of execution of a single query according to new information acquired during its resolving [3]. Surely, another important aspect is to adaptively tune the whole system with respect to evolution of data content and expected types of SQL statements [6].

Figure 4 displays a simple query involving two numeric attributes a and b in relatively small data table T. Minimum/maximum descriptions of data packs for a and b are presented at the left side. Histograms are not taken into account because in this case they are useless. For simplicity, we assume no null values in T. Finally, thanks to columnar architecture, we do not need to access rough values and data packs of any other attributes. – We can assume that row packs are limited to a and b.

Data packs are classified into three categories introduced in Section 2, denoted as R (relevant / positive region), I (irrelevant / negative region) and S (suspect / boundary region). In the first execution stage, classification is performed with respect to condition b>15. The second stage employs rough values of row pack [A3,B3] to approximate final result as MAX(a)$\geq$18. As a consequence, all row packs except [A1,B1] and [A3,B3] become irrelevant. At the third stage, approximation is changed to MAX(a)$\geq$x, where x depends on the outcome of exact row-by-row computation (denoted by E) over the content of row pack [A1,B1]. If x$\geq$22, i.e., if row pack [A1,B1] turns out to contain at least one row with values satisfying conditions b>15 and a$\geq$22, then there is no need to access row pack [A3,B3].

The above scenario is actually one more example of iterative refinements of data pack classifications. Moreover, rough values are applied here to optimize decompression ordering, by following a kind of expected information gain related to accessing particular packs. Last but not least, this example shows a natural ability to produce query result approximations, which is now one of our most important research topics [16]. As illustration, let us note that rough values in Figure 4 provide more information than just MAX(a)$\geq$18. Given irrelevance of row pack [A5,B5], we know that MAX(a) must fall into interval [18,25]. This interval can be gradually narrowed down by accessing some of data packs.

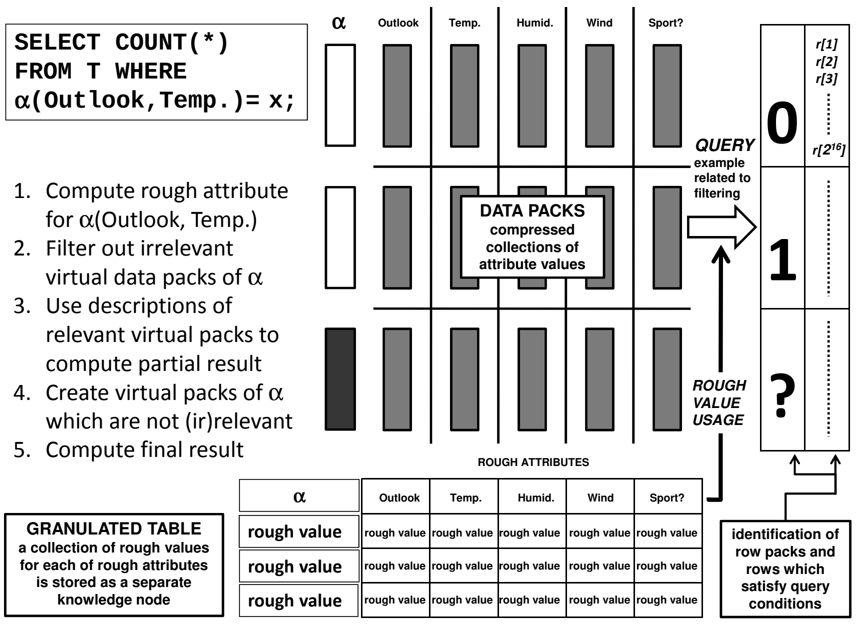

SELECT COUNT(*)
FROM T WHERE
α(Outlook,Temp.)= x;
1. Compute rough attribute for α(Outlook, Temp.)
2. Filter out irrelevant virtual data packs of α
3. Use descriptions of relevant virtual packs to compute partial result
4. Create virtual packs of α which are not (ir)relevant
5. Compute final result
α
Outlook
Temp.
Humid.
Wind
Sport?
DATA PACKS compressed collections of attribute values
QUERY example related to filtering
ROUGH VALUE USAGE
0
1
?
r[1]
r[2]
r[3]
r[2^16]
ROUGH ATTRIBUTES
rough value
GRANULATED TABLE a collection of rough values for each of rough attributes is stored as a separate knowledge node
identification of row packs and rows which satisfy query conditions

Figure 5. Execution of query with complex expression in WHERE clause.

Our next case study relates to query performance stability. Precisely, we are interested in comparable speed of resolving analytic SQL statements differing only in details. Let us consider query in Figure 5, which is almost the same as the one presented in Figure 3. It counts all rows satisfying single equality condition. However, now it includes complex expression α which is actually a new attribute with values dynamically derived from original attributes. We refer to such new attributes as to virtual attributes. They may occur at many stages of query resolving, including attributes to be aggregated, attributes used for grouping, attributes representing results of correlated subqueries and others.

When executing queries such as the one in Figure 5, we do not know any higher level descriptions of so called virtual data packs corresponding to α. At exact computation level, there is of course no problem to calculate values of α and create boolean filter representing rows satisfying the condition. However, we also need to be able to operate at rough level, in order to avoid potentially unnecessary accesses to data packs of attributes defining α. In other words, we need to calculate mathematical descriptions of virtual data packs directly from existing rough values, with no need of accessing the corresponding data packs unless they are easily available, e.g., cached in memory in decompressed form.

Infobright's framework includes rough value derivation algorithms for date functions, arithmetic operations, conditional expressions, etc. Our algorithms extend standard calculations onto non-deterministic input values. Most of them are derived from general principles of granular computing, referring e.g. to interval arithmetics [12]. Resulting descriptions of virtual data packs can be utilized exactly like original rough values. Surely, such descriptions may be less accurate than rough values computed directly from data pack contents. Actually, they should be treated as approximations of rough values. Thus, in procedures such as the one outlined in Figure 5, we should expect more false suspects than usually. Nevertheless, such approximate descriptions are usually good enough to provide reasonable comparison in performance of analytic operations with and without complex expressions.

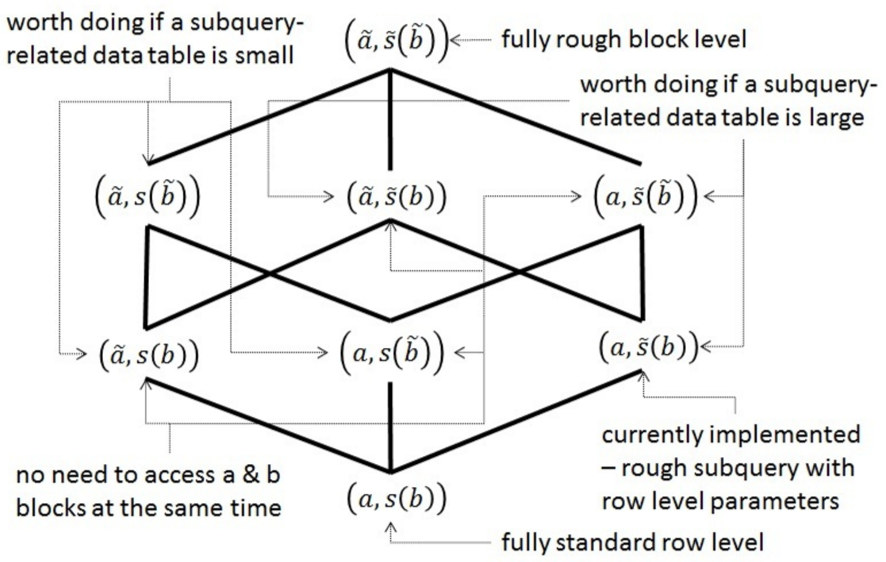

worth doing if a subquery-related data table is small
$(\tilde{a}, \tilde{s}(\tilde{b}))$
fully rough block level
worth doing if a subquery-related data table is large
$(\tilde{a}, s(\tilde{b}))$
$(\tilde{a}, \tilde{s}(b))$
$(a, \tilde{s}(\tilde{b}))$
$(\tilde{a}, s(b))$
$(a, s(\tilde{b}))$
$(a, \tilde{s}(b))$
currently implemented – rough subquery with row level parameters
no need to access a & b blocks at the same time
$(a, s(b))$
fully standard row level

Figure 6. Infobright's roadmap for improving correlated subqueries [16].

We close this section with an example of assigning rough values to intermediate structures produced during query execution. In particular, we show that rough values can result from approximate resolving of subqueries. Let us consider `SQL` statement working on so called outer table `U` with the following condition linking its values to correlated subquery on inner table `T`: `U.a=(SELECT MAX(T.c) FROM T WHERE T.d>U.b);` [16]. Subquery's result needs to be considered for each row of `U`. However, we can attempt to compute approximate results of inner statement parameterized by values of `b` based on rough values of `T`. In many cases, such approximate results can be sufficient to resolve (negatively or positively) equality condition on `a` with no need of accessing any data packs of `T`. Thus, we actually construct a virtual attribute with rough values prior to their (optional) exact derivation.

Let us now briefly discuss how to express query result approximations for arbitrary queries. In the above example, the result of a single subquery takes a form of a single number corresponding to `MAX(T.a)`. In general, one can think about `SELECT` statement results as information systems [10] with attributes and objects specified after `SELECT` and `FROM`, respectively. In Infobright, by rough query we mean computing rough values (or their approximations, just like in case of complex expressions) of such information systems with (almost) no need of decompressing the underlying data packs.

Rough query functionality is a step toward modern scalability of analytic operations which gains an increasing interest in industry.[2] Moreover, we can still search for new ways to utilize rough querying for standard computation purposes. Figure 6 depicts some future possibilities for correlated subqueries in `WHERE` clauses. The role of outer table's attributes `a` and `b` is the same as in the example above. `a` / `b` and $\widetilde{\mathtt{a}}$ / $\widetilde{\mathtt{b}}$ denote their exact (for rows) and rough (for row packs) values. `s` and $\widetilde{\mathtt{s}}$ denote precise and rough types of subquery execution. Pairs in brackets denote which modes of (rough) values are compared to each other while resolving equality condition. Currently, the only implemented strategy is $(\mathtt{a}, \widetilde{\mathtt{s}}(\mathtt{b}))$, followed by $(\mathtt{a}, \mathtt{s}(\mathtt{b}))$ whenever results of $\widetilde{\mathtt{s}}$ are not enough for a given row in outer table.

[2]`www.dbms2.com/2011/06/14/infobright-4-0/`

4. Rough Sets and Data Layout Organization

In the previous sections we focused on adopting rough set principles in order to conduct approximate computations on granulated tables. We mentioned that such computations could be compared with other scenarios of working with imprecisely defined data values [2]. Actually, granulated tables can be considered as an extension of databases with incomplete information [7], with rough values taking a form of sets, intervals, or more complex structures. On the other hand, let us emphasize one more time that the layer of rough values may be not enough to resolve all queries with sufficient accuracy, so interactions between operations on data packs and their descriptions shall be in place.

We already know that approximations of query related concepts are not the only possibility of utilizing rough values. For example, Figure 4 illustrates heuristic prioritization of suspect data pack accesses. This is important because changing an order of such accesses or operations can help to avoid looking through data areas which seem to be required at a current stage of processing but would turn out to be irrelevant eventually. Rough values can be also utilized by many other query optimization techniques, such as result set size estimation or sample generation. However, regardless of how we use rough values representing both original and intermediately constructed data, we need some generic way of evaluating their expected efficiency from overall system performance perspective.

First of all, rough values need to be informative. Informativeness should be considered in relation to what is likely to be most useful for expected data operations. We should also take into account how precisely rough values describe their corresponding data pack contents. Such intuitive criteria of informativeness and precision could be actually considered for any approach based on granular scalability principles introduced in Section 1, regardless of whether it assumes interaction between approximate and exact levels of computing or it rather focuses on the layer of non-exact descriptions. In this section, we follow some rough set notions in order to better formalize those criteria.

Production of efficient rough value layer can be discussed in terms of so called minimum description length principle, also adopted to rough sets [11]. In order to assure fast query performance, physical size of information kept in granulated tables needs to be significantly smaller than that of the underlying data. On the other hand, rough values should prevent accessing data regions that would be unnecessary to resolve a given query when having direct access to detailed data. Richer and more precise descriptions lead to finer approximations, which can limit occurrences of previously mentioned false suspects, i.e., data packs classified as suspects that turn out to be irrelevant after their decompression. However, being highly informative should not mean being overdetailed. Moreover, one should be aware of overhead related to derivations of too sophisticated statistics for new row packs.

Creation of precise and generally efficient rough values depends on the process of organizing rows into row packs. Such process could be compared to data granulation [12] (although one needs to remember that sets of rows dropping into different row packs are pairwise disjoint), data partitioning (although it differs from database partitioning – in particular, we do not assume that minimum/maximum data pack descriptions follow any form of range partitioning) or data clustering (especially generation of micro-clusters and their summaries in data stream clustering [1]). Using rough set terminology, we can say that row packs correspond to indiscernibility classes obtained by using some data organization parameters. Consequently, rough values correspond to compact descriptions of attribute values that we attempt to approximate by means of row packs during query execution. This observation leads us toward a new level of rough set interpretations, which are extremely helpful while discussing how to design data organization procedures aiming at improving Infobright's granulated tables.

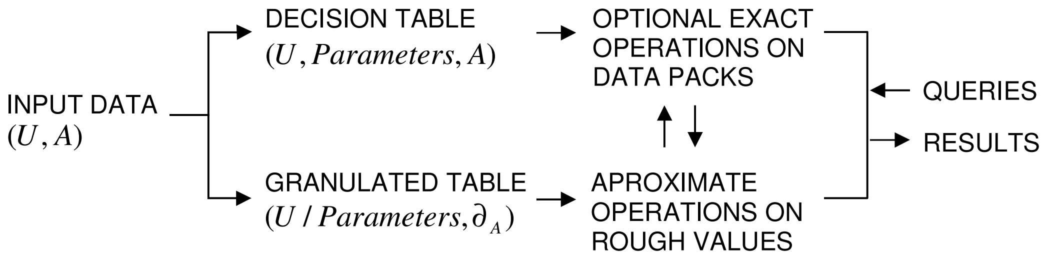


Figure 7. Rough set based interpretation of Infobright's data organization and query processing.

Let us recall some basic notions from rough set theory. Consider a decision table (U, C, D), where U denotes a universe of objects, C is a set of conditional attributes and D is a set of decision attributes. Each $a \in C \cup D$ corresponds to function $a : U \to V_a$, where V_a denotes the set of all values of a. Given indiscernibility relation $IND = \{(x, y) \in U \times U : \forall_{c \in C}\, c(x) = c(y)\}$, for each $d \in D$, we define generalized decision function $\partial_d : U/IND \to 2^{V_d}$ labeling equivalence classes $X \in U/IND$ with subsets $\partial_d(X) = \{d(x) : x \in X\}$. For any $V \subseteq V_d$, we can consider the following:

- $POS_d(V) = \bigcup\{X \in U/IND : \partial_d(X) \subseteq V\}$
- $NEG_d(V) = \bigcup\{X \in U/IND : \partial_d(X) \cap V = \emptyset\}$
- $BND_d(V) = \bigcup\{X \in U/IND : \partial_d(X) \not\subseteq V \wedge \partial_d(X) \cap V \neq \emptyset\}$

As shown in Figure 7, we interpret the set of parameters utilized to create row packs as the set of conditional attributes. Furthermore, we interpret the set `A` of physical data columns as the set of decision attributes. Consequently, our decision table model takes a form of triple $(U, Parameters, A)$. It leads to an interesting analogy between rough values of attributes `a` $\in$ `A` and generalized decisions ∂_a. Such analogy is particularly visible for low cardinality attributes considered in Section 2. In this case, $\partial_a(X)$ corresponds to histogram description of attribute `a` for row pack X. Moreover, if we treat subset $V \subseteq V_a$ as related to a constraint specified by `WHERE` clause on `a`, then, for the corresponding `SQL` statement, the above regions turn out to gather relevant, irrelevant and suspect data packs, respectively.

Treating rough values as generalized decisions enables us to establish even stronger link between Infobright's operations and rough approximations. Moreover, rough set notions can be utilized to describe the process of data organization and rough value evaluation. We can adopt some existing models of analyzing precision of generalized decision functions [4]. We can also think about choosing optimal data organization parameters in terms of searching for rough set reducts [19]. In classical rough set terms, we should actually look for subsets of conditional attributes for which indiscernibility classes and their generalized decisions are good enough to approximate concepts that we are interested in.

On the other hand, the above correspondence needs deeper investigation. For example, in order to deal with arbitrary attribute types, we should extend the meaning of generalized decision as $\widetilde{\partial}_d(X) =$ *description of* $\{d(x) : x \in X\}$. As a result, when considering the already mentioned minimum description length principle for granulated tables represented as tuples $(U/Parameters, \partial_A)$ – where $U/Parameters$ and ∂_A symbolize the sets of indiscernibility classes and vectors of generalized decisions, respectively – we should take into account both complexity of partitioning the universe of rows onto row packs and complexity of descriptions of sets of values stored in single data packs.

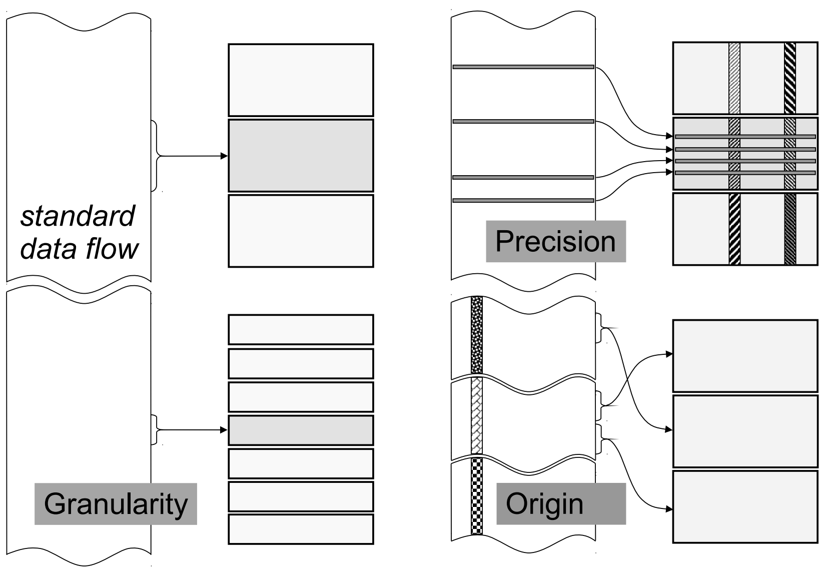
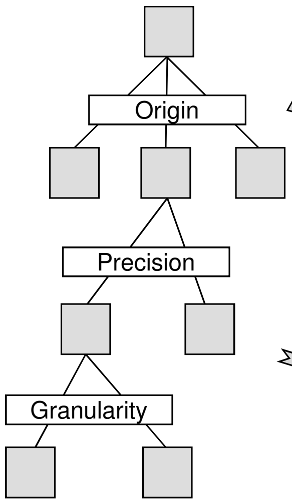


Figure 8. Parameters controlling organization of incoming rows into row packs.

Let us now take a look at some specific parameters influencing precision of rough values and the whole layout of Infobright's (meta)data. We focus on examples in Figure 8. We start with `Granularity` which has straightforward influence on the size of granulated tables. Granularity refers to the amount of rows in each row pack (2^{16} by default). Smaller data packs would be labeled with potentially more precise rough values. Additionally, finer granularity enables to access data more selectively, once we know which data portions are required in a given query. However, it also results in larger number of entries in granulated tables, which would decrease efficiency of their usage because of the size. This tradeoff can be compared with complexity of rough set decision models with respect to the number of their indiscernibility classes and average precision of their generalized decision values [4].

The next parameter – `Precision` – refers to reorganization of ordering of rows prior to producing row packs [15]. Our goal is to group together rows in such a way that precision of their rough value descriptions is improved comparing to initial ordering. As already mentioned, this idea is close to clustering of data streams, i.e., creating maximally homogeneous blocks of rows using low memory footprint algorithms that can digest loaded rows with (almost) no time overhead [1]. One of possible row pack homogeneity criteria is to attempt to improve rough values over arbitrary attributes, but with additional penalty whenever precision for any other attributes drops down too significantly. Such approach has quite a lot in common with rough set criteria for discerning pairs of rows with too different attribute values [9]. Moreover, in case of occurrence of outlying values, gathering such values within minimum amount of data packs turns out to improve average rough value precision in the whole granulated table.

The last considered parameter refers to a way of data acquisition. In many cases, machine-generated data sets originate from different sources such as geographically distributed application servers or sensor networks. In such scenarios, streams of incoming rows can be preprocessed remotely and then they are sent to the main analytic system in form of already assembled data packs. Rows coming from different sources cannot coexist in the same row packs. It implies high precision of rough values for attributes corresponding or even partially correlated to source identifiers (e.g. coordinates or addresses).



Origin	Precision	Timestamp	Outlook	Temp.	...
1	0	1	Sunny	Hot	...
1	0	2	Sunny	Hot	...
1	1	3	Overcast	Hot	...
1	0	4	Rain	Mild	...
1	0	5	Rain	Cold	...

Origin	Precision	Granularity	$\partial_{\text{Outlook}}$	$\partial_{\text{Temp.}}$	...
1	0	0	{Sunny}	{Hot}	...
1	0	0			
1	0	1	{Rain}	{M,C}	...
1	0	1			
1	1	0	{Overcast}	{Hot}	...

Figure 9. Data loading as construction of indiscernibility classes and generalized decisions.

Finally, let us discuss how to compare procedures of row pack and indiscernibility class generation. According to Figure 9, the first step refers to `Origin`. Sources of data coming into our system can be described by hierarchies of attributes corresponding, e.g., to server locations and their clusters. One can load data streams coming from different places in a distributed way or use some intermediate servers to merge streams prior to row pack creation. Streams can be merged at various levels of cluster hierarchy, reflecting tradeoff between a number of parallel load processes and intensity of each of them on average. Optimization of such tradeoff relates to expected overall load speed, data compression ratios and data layout characteristics. From the perspective of decision table $(U, Parameters, A)$, it corresponds to choosing the most appropriate conditional attributes related to data origin.

The next step corresponds to improving precision of rough values. Our framework presented with this respect in [15] can be interpreted as the process of splitting incoming data onto several sub-streams containing rows that are relatively more similar to each other. Amounts of maintained sub-streams and coefficients of row similarity definitions may vary for different data load instances. Moreover, they can evolve in time in order to better adjust to new types of analytic operations performed by end users. Thus, the whole mechanism of partitioning loaded data onto row packs should be considered in terms of adaptive hierarchical attribute construction rather than attribute selection [9].

As already mentioned, our methods for improving rough values may modify initial row ordering. However, sub-streams produced by those algorithms are ordered with respect to time of row acquisition, as symbolized by `Timestamp` in Figure 9. Our study can be completed by comparing `Granularity` to `Timestamp` discretization. From the perspective of decision table $(U, Parameters, A)$, once conditional attribute values related to `Origin` and `Precision` are assigned, the corresponding indiscernibility classes should be split onto pieces containing some predefined amounts of consecutive rows. It also reflects implementation reported in [15], where each sub-stream is buffered until certain amount of rows is gathered and then its content is passed further as a new row pack. It is equivalent to a discretization cut which separates rows belonging to consecutive row packs. As before, parameters responsible for such cuts can vary for particular sub-streams and, moreover, they can change in time.

5. Conclusion

We discussed rough set foundations of Infobright's analytic system aiming at querying against massive amounts of data. We concentrated on two specific interpretations of rough approximations: construction of data granules described by rough values which extend classical generalized decision functions and utilization of those rough values to accelerate execution of ad-hoc `SELECT` statements. We presented a number of case studies of computing query related concept approximations, beginning from relatively simple examples introduced in [18] and following with more advanced mechanisms reported, e.g., in [16]. We also described a new framework for interpreting our data load algorithms in terms of generating indiscernibility classes induced by conditional attributes in rough set decision tables.

Let us outline several directions for future improvements of our technology. First of all, as emphasized in the previous sections, there is a broad discussion whether analytic querying needs to be perfectly precise. From the perspective of industry applications, it refers to requirements of predictive and investigative analytics over dynamically growing data sources. With respect to academic research, one may think about scalable machine learning algorithms which rely on ad-hoc query workloads [9]. Certainly, rough sets can be a source of valuable inspirations in this area. Our current rough query functionality can be further extended, e.g., toward clustered representations of large size `GROUP BY` results, analogously to our already existing framework for data layout organization [15]. Enhanced approximate query capabilities can be especially useful for analysis of naturally distributed machine-generated data sets, where information synchronization may be often possible only at the level of rough values.

Query result clustering deserves additional discussion with respect to delivering summarized information about complex values, often represented as long strings. Such information needs to be meaningful to end users and useful for third party applications, which is in analogy to informativeness criteria formulated for rough values corresponding to `varchar` attributes [17]. It refers to far broader topic of adaptive tuning of analytic data processing systems with respect to user expectations. In case of Infobright's granulated tables, such tuning can be considered in terms of precision and richness of data pack descriptions, as well as parameters responsible for granularity of row pack partitions. It is also worth thinking about granulated tables as interactive information systems with values generated according to some adaptively controlled parameters.[3] On the other hand, one should remember that rough value layer is just one of components of our data processing system requiring a systematic tuning framework.

We would also like to emphasize that, in our opinion, rough-granular approach described in this article could be adopted to some other categories of systems, not so strongly related to mainstream database and data analytics functionalities. Moreover, some of discussed research aspects seem to be important not only for Infobright's technology. For example, in order to better prepare rough set based knowledge discovery and decision support systems for real-world data challenges, one may consider similar extensions of generalized decision functions as those proposed in Section 4. Last but not least, let us note that some other rough set models of approximate computation and knowledge representation are extendable toward dealing with complex, distributed and dynamic data sets as well. In particular, we believe that one could achieve highly efficient and powerful data mining framework by putting together some elements of stream data clustering [1] and rough clustering [13] methodologies.

[3]Based on personal discussion between Professor Andrzej Skowron and the first author.

References

[1] Aggarwal, C. C., Han, J., Wang, J., Yu, P. S.: On Clustering Massive Data Streams: A Summarization Paradigm, in: *Data Streams: Models and Algorithms* (C. C. Aggarwal, Ed.), Springer, 2007, 9–38.

[2] Beaubouef, T., Petry, F. E.: Incorporating Rough Data in Database Design for Imprecise Information Representation, in: *Rough Sets and Intelligent Systems – Professor Zdzisław Pawlak in Memoriam, Volume 2* (A. Skowron, Z. Suraj, Eds.), vol. 42 of *Intelligent Systems Reference Library*, Springer, 2013, 137–155.

[3] Deshpande, A., Ives, Z. G., Raman, V.: Adaptive Query Processing, *Foundations and Trends in Databases*, **1**(1), 2007, 1–140.

[4] Düntsch, I., Gediga, G.: Uncertainty Measures of Rough Set Prediction, *Artificial Intelligence*, **106**(1), 1998, 109–137.

[5] Fagin, R.: Multivalued Dependencies and a New Normal Form for Relational Databases, *ACM Transactions on Database Systems*, **2**(3), 1977, 262–278.

[6] Idreos, S., Groffen, F., Nes, N., Manegold, S., Mullender, K. S., Kersten, M. L.: MonetDB: Two Decades of Research in Column-oriented Database Architectures, *IEEE Data Engineering Bulletin*, **35**(1), 2012, 40–45.

[7] Lipski Jr., W.: On Databases with Incomplete Information, *Journal of the ACM*, **28**(1), 1981, 41–70.

[8] Liu, Q.: Approximate Query Processing, in: *Encyclopedia of Database Systems* (L. Liu, M. T. Özsu, Eds.), Springer, 2009, 113–119.

[9] Nguyen, H. S.: Approximate Boolean Reasoning: Foundations and Applications in Data Mining, *LNCS Transactions on Rough Sets*, **5**, 2006, 334–506.

[10] Pawlak, Z.: Rough Sets, *International Journal of Parallel Programming*, **11**(5), 1982, 341–356.

[11] Pawlak, Z., Skowron, A.: Rough Sets: Some Extensions, *Information Sciences*, **177**(1), 2007, 28–40.

[12] Pedrycz, W.: *Granular Computing – Analysis and Design of Intelligent Systems*, CRC Press, 2013.

[13] Peters, G., Crespo, F., Lingras, P., Weber, R.: Soft Clustering – Fuzzy and Rough Approaches and Their Extensions and Derivatives, *Internation Journal of Approximate Reasoning*, **54**(2), 2013, 307–322.

[14] Skowron, A., Grzymała-Busse, J.: From Rough Set Theory to Evidence Theory, in: *Advances in the Dempster Shafer Theory of Evidence* (R. R. Yaeger, M. Fedrizzi, J. Kacprzyk, Eds.), Wiley, 1994, 193–236.

[15] Ślęzak, D., Kowalski, M., Eastwood, V., Wróblewski, J.: Methods and Systems for Database Organization, US Patent 8,266,147 B2, 2012.

[16] Ślęzak, D., Synak, P., Borkowski, J., Wróblewski, J., Toppin, G.: A Rough-columnar RDBMS Engine – A Case Study of Correlated Subqueries, *IEEE Data Engineering Bulletin*, **35**(1), 2012, 34–39.

[17] Ślęzak, D., Toppin, G., Kowalski, M., Wojna, A.: System and Method for Managing Metadata in a Relational Database, US Patent 8,521,748, 2013.

[18] Ślęzak, D., Wróblewski, J., Eastwood, V., Synak, P.: Brighthouse: An Analytic Data Warehouse for Ad-hoc Queries, *Proceedings of the VLDB Endowment*, **1**(2), 2008, 1337–1345.

[19] Świniarski, R. W., Skowron, A.: Rough Set Methods in Feature Selection and Recognition, *Pattern Recognition Letters*, **24**(6), 2003, 833–849.

[20] White, P. W., French, C. D.: Database System with Methodology for Storing a Database Table by Vertically Partitioning all Columns of the Table, US Patent 5,794,229, 1998.