An In-Database Rough Set Toolkit

Frank Beer and Ulrich Bühler

University of Applied Sciences Fulda
Leipziger Straße 123, 36037 Fulda, Germany
{frank.beer,u.buehler}@informatik.hs-fulda.de

Abstract. The Rough Set Theory is a common methodology to discover hidden patterns in data. Most software systems and libraries using methods of that theory originated in the mid 1990s and suffer from time-consuming operations or high communication costs. Today on the other hand there is a perceptible trend for in-database analytics allowing on-demand decision support. While data processing and predictive models remain in one system, data movement is eliminated and latency is reduced. In this paper we contribute to this trend by computing traditional rough sets solely inside relational databases. As such we leverage the efficient data structures and algorithms provided by that systems. Thereby we introduce a baseline framework for in-database mining supported by Rough Set Theory. Immediately, it can be utilized for common discovery tasks such as feature selection or reasoning under uncertainty and is applicable to most conventional databases as our experiments indicate.

Keywords: concept approximation, in-database analytics, knowledge discovery in databases, relational algebra, relational database systems, rough set theory

1 Introduction

Over the past decades, the huge quantities of data accumulating as a part of business operations or scientific research raised the necessity for managing and analyzing them effectively. As a result, Rough Set Theory (RST) became subject to these interdisciplinary areas as reliable instrument of extracting hidden knowledge from data. That trend is visible in the versatile existence of rough set-based software libraries and tools interfacing data from flat files [1–4]. The design of such libraries and tools, however, suffers when applying them to real-world data sets due to resource and time-consuming file operations. To overcome this technological drawback, researchers have made the effort to build more scalable rough set systems by utilizing relational databases which provide very efficient structures and algorithms designed to handle huge amounts of information [5–9].

 In: R. Bergmann, S. Görg, G. Müller (Eds.): Proceedings of the LWA 2015 Workshops: KDML, FGWM, IR, and FGDB. Trier, Germany, 7.-9. October 2015, published at http://ceur-ws.org

However, the exploitation of database technology can be further extended. One can assess these relational systems to be expandable platforms capable of solving complex mining tasks independently. This design principle has been broadly established under the term *in-database analytics* [10]. It provides essential benefits, because hidden knowledge is stored in relational repositories predominantly either given through transactional data or warehouses. Thus, pattern extraction can be applied in a more data-centric fashion. As such, data transports to external mining frameworks are minimized and processing time can be reduced to a large extend. That given, one can observe database manufacturers continiously expand their engines for analytical models[1] such as association rule mining or data classification.

A full integration of rough sets inside relational systems is most favorable where both processing and data movement is costly. Unfortunately in-database processing and related applications are only covered partially in existing RST literature. Few practical attempts have been made to express the fundamental concept approximation based on existing database operations. In this paper we concentrate on that gap and present a concrete model to calculate rough sets inside relational databases. We redefine the traditional concept approximation and compute it by utilizing extended relational algebra. This model can be translated to various SQL dialects and thus enriches most conventional database systems. In line with ordinary RST our proposed model can be applied to common mining problems such as dimensionality reduction, pattern extraction or classification. Instrumenting SQL and its extensions enable us to cover further steps in the classic knowledge discovery process implicitly including selection and preprocessing. Combined, we obtain a baseline toolkit for in-database mining which relies on rough set methodology and database operations. It is natively applicable without the use of external software logic at low communication costs. Additionally, relational database engines have been significantly improved over the last decades, implementing both a high degree of parallelism for queries and physical operations based on hash algorithms which is a major factor for the efficiency of our model.

The remainder is structured as follows: First we present important aspects of the RST (Section 2). In Section 3 we review ideas and prototypes developed by other authors. Section 4 restructures the concept approximation. The resulting propositions are utilized to build a model based on database operations in Section 5. Then we briefly demonstrate how our model scales (Section 6). Based on that, we present future work (Section 7) and conclude in Section 8.

2 Rough Set Preliminaries

Proposed in the early 1980s by Zdzislaw Pawlak [11, 12], RST is a mathematical framework to analyze data under vagueness and uncertainty. In this section

[1] see Data Mining Extensions for Microsoft SQL Server: `https://msdn.microsoft.com/en-us/library/ms132058.aspx` (June, 2015) or Oracle Advanced Analytics: `http://oracle.com/technetwork/database/options/advanced-analytics` (June, 2015)

we outline principles of that theory: the basic data structures including the indiscernibility relation (Section 2.1) and the illustration of the concept approximation (Section 2.2).

2.1 Information Systems and Object Indiscernibility

Information in RST is structured in an Information System (IS) [13], i.e. a data table consisting of objects and attributes. Such an IS can thus be expressed in a tuple $\mathcal{A} = \langle \mathbb{U}, A \rangle$, where the universe of discourse $\mathbb{U} = \{x_1, ..., x_n\}, n \in \mathbb{N}$, is a set of objects characterized by the feature set $A = \{a_1, ..., a_m\}, m \in \mathbb{N}$, such that $a : \mathbb{U} \to V_a, \forall a \in A$, where V_a represents the value range of attribute a. An extension to an IS is the Decision System (DS). A DS even holds a set of attributes where some context-specific decision is represented. It consists of common condition features A and the decision attributes $d_i \in D$ with $d_i : \mathbb{U} \to V_{d_i}, 1 \leq i \leq |D|$ and $A \cap D = \emptyset$. A DS is denoted by $\mathcal{A}_D = \langle \mathbb{U}, A, D \rangle$. If we have for any $a \in A \cup D : a(x) = \perp$, i.e. a missing value, the underlying structure is called incomplete, otherwise we call it complete.

The indiscernibility relation classifies objects based on their characteristics. Formally, it is a parametrizable equivalence relation with respect to a specified attribute set and can be defined as follows: Let be an IS $\mathcal{A} = \langle \mathbb{U}, A \rangle$, $B \subseteq A$, then the indiscernibility relation $IND_{\mathcal{A}}(B) = \{(x, y) \in \mathbb{U}^2 \mid a(x) = a(y), \forall a \in B\}$ induces a partition $\mathbb{U}/IND_{\mathcal{A}}(B) = \{K_1, ..., K_p\}, p \in \mathbb{N}$ of disjoint equivalence classes over $\mathbb{U}$ with respect to B. Out of convenience we write IND_B or $\mathbb{U}/B$ to indicate the resulting partition.

2.2 Concept Approximation

To describe or predict an ordinary set of objects in the universe, RST provides an approximation of that target concept applying the indiscernibility relation. Let be $\mathcal{A} = \langle \mathbb{U}, A \rangle$, $B \subseteq A$ and a concept $X \subseteq \mathbb{U}$. Then, the B-lower approximation of the concept X can be specified through

$$\underline{X}_B = \bigcup \{K \in IND_B \mid K \subseteq X\} \tag{1}$$

while the B-upper approximation of X is defined as

$$\overline{X}_B = \bigcup \{K \in IND_B \mid K \cap X \neq \emptyset\} \, . \tag{2}$$

Traditionally, (1) and (2) can be expressed in a tuple $\langle \underline{X}_B, \overline{X}_B \rangle$, i.e. the rough set approximation of X with respect to the knowledge in B. In a rough set, we can assert objects in $\overline{X}_B$ to be fully or partly contained in X, while objects in $\underline{X}_B$ can be determined to be surely in the concept. Hence, there may be equivalence classes which describe X only in an uncertain fashion. This constitutes the B-boundary $\overline{X}_B = \overline{X}_B - \underline{X}_B$. Depending on the characteristics of $\overline{X}_B$ we get an indication of the roughness of $\langle \underline{X}_B, \overline{X}_B \rangle$. For $\overline{X}_B = \emptyset$, we can classify X decisively, while for $\overline{X}_B \neq \emptyset$, the information in B appears to be insufficient to describe X properly. The latter leads to an inconsistency in the data. The rest of objects not involved in $\langle \underline{X}_B, \overline{X}_B \rangle$ seems to be unimportant and thus can be

disregarded. This set is called B-outside region and is the relative complement of $\overline{X}_B$ with respect to $\mathbb{U}$, i.e. $\mathbb{U} - \overline{X}_B$.

When we are focused in approximating all available concepts induced by the decision attributes, RST provides general notations consequently. Let be $\mathcal{A}_D = \langle \mathbb{U}, A, D \rangle$ and $B \subseteq A, E \subseteq D$, then all decision classes induced by IND_E can be expressed and analyzed by two sets, i.e. the B-positive region denoted as

$$POS_B(E) = \bigcup_{X \in IND_E} \underline{X}_B \tag{3}$$

and the B-boundary region

$$BND_B(E) = \bigcup_{X \in IND_E} \overline{X}_B \, . \tag{4}$$

For $POS_B(E)$ and $BND_B(E)$ we get a complete indication whether the expressiveness of attributes B is sufficient in order to classify objects well in terms of the decisions given in E. Based on that, the concept approximation is suitable for a varity of data mining problems. Among others, it can be applied to quantify imprecision, rule induction or feature dependency analysis including core and reduct computation for dimensionality reduction [12].

3 Related Work

The amount of existing RST literature intersecting with databases theory increased continuously since the beginning. In this section we outline the most relevant concepts and systems introduced by other authors.

One of the first systems combining RST with database systems was introduced in [5]. The presented approach exploits database potentials only partially, because used SQL commands are embedded inside external programming logic. Porting this sort-based implementation for in-database applications implies the usage of procedural structures such as cursors, which is not favorable in processing enormous data. In [14], the authors modify relational algebra to calculate the concept approximation. Database internals need to be touched and hence a general employment is not given. The approaches in [6, 7] utilize efficient relational algebra for feature selection. The algorithms omit the usage of the concept approximation by other elaborated rough set properties. This factor limits the application to dimension reduction only. Sun et al. calculate rough sets based on extended equivalence matrices inside databases [9]. Once data is transformed into that matrix structure, the proposed methods apply but rely on procedural logic rather than scalable database operations. The work of Nguyen aims for a reduction of huge data loads in the knowledge discovery process [15]. Therefore appropriate methods are introduced using simpler SQL queries to minimize traffic in client-server architectures. The software design follows to the one in [5]. In [8], Chan transforms RST into a multiset decision table which allows to calculate the concept approximation with database queries. The initial construction of such a data table relies on the execution of dynamic queries, helper tables and row-by-row updates as stated in [16] and thus depends on inefficient preprocessing. The work of Naouali et al. implements α-RST in data warehouse environments [17]. The algorithm relies on iterative processing and insert commands to

determine the final classification. Details about its efficiency are not presented. Another model is known as rough relational database [18]. These systems base on multi-valued relations designed to query data under uncertainty. Over the years, specific operations and properties of this theoretic model have been further extended. The authors in [19] try to port the rough relational data model to mature database systems. Details of migrating its algebra are not reported. Infobright is another database system that focuses on fast data processing towards ad-hoc querying [20]. This is achieve by a novel data retrieval strategy based on compression and inspired by RST. Data is organized underneath the knowledge grid. It is used to get estimated query results rather than seeking costly information from disk, which is valid to some domain of interest.

Most discussed approaches utilize inefficient procedural structures, external programs or leverage relational operations for very specific subjects. In contrast, we make use of existing, reliable and highly optimized database operations to compute the concept approximation not employing further procedural mechanisms. With this, we stretch the applicability of independent databases to a broader range of rough set mining problems.

4 Redefining the Concept Approximation

This section points out the formal ideas of transforming Pawlak's concept approximation to relational database systems by introducing a mapping of (1) and (2) to rewritten set-oriented expressions. Those propositions can then be applied to database algebra easily and enable us to transport both, the positive region and the boundary region in addition. We also show that these redefinings are no extensions to the traditional model, but equivalent terms.

Explained in Section 2.2, a rough set $\langle \underline{X}_B, \overline{X}_B \rangle$ can typically be extracted from a concept $X \subseteq \mathbb{U}$ of an IS $\mathcal{A} = \langle \mathbb{U}, A \rangle$ on a specific attribute set $B \subseteq A$, while the classification of each object is based on the induced partition $\mathbb{U}/B$. At this point, we make use of $X/B := X/IND_{\mathcal{A}}(B) = \{H_1, ..., H_q\}, q \in \mathbb{N}$, restructuring the concept approximation of X. Thus, we can deduce two relationships between classes $H \in X/B$ and $K \in \mathbb{U}/B$: $H \cap K \neq \emptyset$, $H \cap K = \emptyset$. This basic idea leads to two propositions, which we discuss in the remainder of this section:

$$\underline{X}_B = \bigcup \{H \in \mathbb{U}/B \mid H \in X/B\} \tag{5}$$

Proof. Considering the classes $H \in X/B$, the following two cases are of interest to form the B-lower approximation: (a) $\exists K \in \mathbb{U}/B : K = H \subseteq X$ and (b) $\exists K \in \mathbb{U}/B : K \neq H$ and $K \cap H \neq \emptyset$. Case (b) implies $\exists z \in K : z \notin X$ and thus $K \not\subseteq X$. As a result, only classes $K = H$ are relevant. Likewise, (1) only contains objects of classes $K \in \mathbb{U}/B$, where $K \subseteq X$. We consider X/B that induces classes $H \in \mathbb{U}/B$ and $H' \notin \mathbb{U}/B$, because $X \subseteq \mathbb{U}$. Combined, we immediately get to (5). □

$$\overline{X}_B = \bigcup \{K \in \mathbb{U}/B \mid \exists H \in X/B : H \subseteq K\} \tag{6}$$

Proof. On the one hand, the partition X/B can only produce equivalence classes $H, H' \subseteq X$ which satisfy $H \in \mathbb{U}/B$ and $H' \notin \mathbb{U}/B$. Obviously, those H are members of the B-lower approximation, whereas each class H' has a matching partner class K with $H' \subset K \in \mathbb{U}/B$ which build the B-boundary approximation. With these classes H, K, we directly receive: $\overline{X}_B = \underline{X}_B \cup \overline{\underline{X}}_B$. On the other hand, $\overline{X}_B$ holds objects of classes $K \in \mathbb{U}/B$ with $K \cap X \neq \emptyset$ (see (2)), i.e. each class $K \in X/B$ and $K \supset H \in X/B$. This is proposed by (6). □

Up to this point, the B-boundary approximation and the B-outside region remain untouched for further restructuring since both sets build on the B-lower and B-upper approximation. They have the same validity to the propositions in (5) and (6) as to the classical rough set model.

5 Combining RST and Database Systems

5.1 Information Systems and Database Tables

The IS is a specific way to organize data, similar to a data table in relational database terms. But there are essential differences in their scientific scopes [13]. While an IS is used to discover patterns in a snapshot fashion, the philosophy of databases concerns with long term data storing and retrieval respectively [21].

However, we try to overcome these gaps by simply assembling an IS or DS to the relational database domain considering the following: Let be $\mathcal{A}_D = \langle \mathbb{U}, A, D \rangle$ with the universe $\mathbb{U} = \{x_1, ..., x_n\}, n \in \mathbb{N}$, the features $A = \{a_1, ..., a_m\}, m \in \mathbb{N}$ and the decision $D = \{d_1, ..., d_p\}, p \in \mathbb{N}$, then we use the traditional notation of a $(m+p)$-ary database relation $R \subseteq V_{a_1} \times ... \times V_{a_m} \times V_{d_1} \times ... \times V_{d_p}$, where V_{a_i} and V_{d_j} are the attribute domains of $a_i, d_j, 1 \leq i, j, \leq m, p$.

In database theory, the order of attributes in a relation schema has significance to both semantics and operations. With this we simplify the employment of attributes to finite sets and write $A = \{a_1, ..., a_q\}, q \in \mathbb{N}$ for the ordered appearance in relation R. We notate R_A as shortform or R_{A+D} to identify a decision table. Furthermore modern databases permits duplicated tuples within its relational structure. We adopt this rudiment with practical relevance and designate these types of relations as *database relation* or *data table* respectively.

5.2 Indiscernibility and Relational Operations

Inspired by [5–7], we make use of extended relational algebra to calculate the partition of the indiscernibility relation. Using the *projection* operation $\pi_B(R_A)$ allows to project tuples $t \in R_A$ to a specified feature subset $B \subseteq A$ while eliminating duplicates. Thus, we get each class represented by a proxy tuple with schema B. A column reduction without duplicate elimination is indicated by $\pi^+_B(R_A)$. Additionally, we introduce the *selection* operation $\sigma_\phi(R_A)$ with filter property ϕ and output schema A. Given a geometric repository R_{A+D} (see Figure 1), we may query objects $x \in R_{A+D}$ that are colored *red* by $\sigma_{x.color=red}(R_{A+D})$.

Most relational database systems provide an extension to $\pi_B(R_A)$, i.e. the *grouping* operator γ. It groups tuples of a relation R_A if they share identical values entirely over an specified attribute set $G \subseteq A$, i.e. the grouping attributes.

Each group is only represented once in the resulting relation through a proxy tuple (see π-operator). In addition, γ can be enriched with aggregation functions $f_1, ..., f_n$ that may be applied to each group during the grouping phase. Generally, this operation can be defined by $\gamma_{F;G;B}(R_A)$, where B are the output features with $B \subseteq G \subseteq A$ and $F = \{f_1, ..., f_n\}, n \in \mathbb{N}_0$. For our purpose we simply count the number of members in each single group (class) of R_A, i.e. the cardinality expressed by the aggregate $count(*)$, and include it as new feature. Consolidated, we make use of the following notation

$$\mathcal{I}_B^G(R_A) := \rho_{card \leftarrow count(*)}(\gamma_{\{count(*)\};G;B}(R_A)) \tag{7}$$

where $\rho_{b \leftarrow a}(R_A)$ is the renaming operation of an arbitrary attribute $a \in A$ to its new name b in table R_A. Then $\mathcal{I}_B^B(R_A)$ is supposed to be noted as our compressed multiset representation of a given database table R_A considering feature set $B \subseteq A$. An illustration of this composed operation and its parallels to the RST is depicted in Figure 1 with $A = \{shape, color\}$, $D = \{d\}$ and $B = A$.

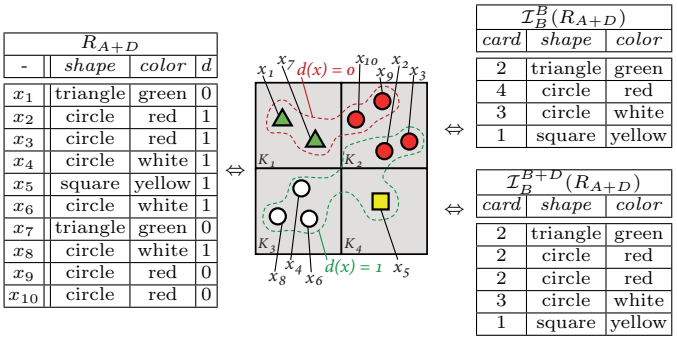

Fig. 1. Mapping the object indiscernibility to relational data tables

5.3 Mapping the Concept Approximation

In practice, the extraction process of a single target concept may vary dependent on domain and underlying data model. In most cases an ordinary target concept can be modelled through decision attributes, i.e. a decision table. However there might be domains of interest, where the concept is located outside the original data table. Especially, this is the case in highly normalized environments. We support both approaches within the boundaries of the relational model. As simplification we assume the target concept C_A and the original data collection R_A to be given through either adequate relational operations or their native existence in a relation where C_A is a subset of R_A.

Taking this and the previous sections into account, we are now able to demonstrate the classical concept approximation in terms of relational algebra and its extensions: Let be R_A representing the universe and C_A our target concept to be examined with the feature subset $B \subseteq A$, then the B-lower approximation of

the concept can be expressed by

$$\mathcal{L}_B(R_A, C_A) := \mathcal{I}_B^B(C_A) \cap \mathcal{I}_B^B(R_A) \ . \tag{8}$$

Initially, (8) establishes the partition for C_A and R_A independently. The intersection then only holds those kinds of equivalence classes included with their full cardinality in both induced partitions, i.e. the B-lower approximation in terms of the RST (see (5)). The B-upper approxiation contains all equivalence classes associated with the target concept. Thus, we simply can extract one representative of these classes from the induced partition of C_A applying the information in B. However, this information is not sufficient to get the correct cardinality of those classes involved. Hence we must consider the data space of R_A in order to find the number of all equivalences. That methodology can be expressed through

$$\mathcal{U}_B(R_A, C_A) := \pi_B(C_A) \bowtie \mathcal{I}_B^B(R_A) \tag{9}$$

whereas $\bowtie$ is the natural join operator, assembling two data tables S_G, T_H to a new relation R such that $s.b = t.b$ for all tuples $s \in S_G, t \in T_H$ and attributes $b \in G \cap H$. Note, R consists of all attributes in G, H, where overlapping attributes are shown only once. As a result, we get all equivalence classes with their cardinality, involved in the B-upper approximation (see (6)). Classically, the B-boundary consists of objects located in the set-difference of B-upper and B-lower approximation. Because of the structural unity of $\mathcal{L}_B(R_A, C_A)$ and $\mathcal{U}_B(R_A, C_A)$, it can be expressed by

$$\mathcal{B}_B(R_A, C_A) := \mathcal{U}_B(R_A, C_A) - \mathcal{L}_B(R_A, C_A) \ . \tag{10}$$

Equivalence classes outside the concept approximation can be found when searching for tuples not included in the B-upper approximation. With the support of both $\mathcal{I}_B^B(R_A)$ and $\mathcal{U}_B(R_A, C_A)$, we therefore get the B-outside region

$$\mathcal{O}_B(R_A, C_A) := \mathcal{I}_B^B(R_A) - \mathcal{U}_B(R_A, C_A) \ . \tag{11}$$

In order to present an equivalent relational mapping of (3) and (4), we first have to look at a methodology that allows us to query each target concept separately. Within a decision table R_{A+D}, let us assume the partition induced by the information in $E \subseteq D$ consists of n decision class. For each of these classes we can find an appropriate condition $\phi_i, 1 \leq i \leq n$ that assists in extracting the associate tuples $t \in R_{A+D}$ belonging to each concept $C_A^{\phi_i}$. One can simply think of a walk through $\pi_E(R_{A+D})$. In the i-th iteration we fetch the decision values, say $v_1, ..., v_m$, for the corresponding features in $E = \{d_1, ...d_m\}, m \in \mathbb{N}$ and build $\phi_i = \bigwedge_{1 \leq j \leq m} t.d_j = v_j$. Thus, we have access to each decision class $C_A^{\phi_i} = \pi_A^+(\sigma_{\phi_i}(R_{A+D}))$ produced by E. With this idea in mind and supported by (8) we are now able to introduce the B-positive region: In a decision table R_{A+D} and $B \subseteq A, E \subseteq D$, the B-positive region is the union of all B-lower approximations induced by the attributes in E. Those concepts can be retrieved by $C_A^{\phi_i}, 1 \leq i \leq n$ where n is the cardinality of $\pi_E(R_{A+D})$. As a consequence we get to

$$\mathcal{L}_B(R_{A+D}, C_A^{\phi_1}) \cup ... \cup \mathcal{L}_B(R_{A+D}, C_A^{\phi_n}) \tag{12}$$

which can be rewritten as

$$\bigcup_{i=1,\ldots,n} \mathcal{I}_B^B(C_A^{\phi_i}) \cap \mathcal{I}_B^B(R_{A+D}) \tag{13}$$

such that we finally have the B-positive region in relational terms defined over a decision table

$$\mathcal{L}_B^E(R_{A+D}) := \pi_{B'}(\mathcal{I}_B^{B+E}(R_{A+D})) \cap \mathcal{I}_B^B(R_{A+D}) \tag{14}$$

with $B' = \{card, b_1, ..., b_k\}, b_j \in B, 1 \leq j \leq k$. Likewise, the B-boundary region consists of tuples in $\mathcal{U}_B(R_{A+D}, C_A^{\phi_1}) \cup ... \cup \mathcal{U}_B(R_{A+D}, C_A^{\phi_n})$ but not in $\mathcal{L}_B^E(R_{A+D})$, where $C_A^{\phi_i}, 1 \leq i \leq n \in \mathbb{N}$ are the separated target concepts induced by E. Hence, we can query these through

$$\bigcup_{i=1,\ldots,n} \mathcal{U}_B(R_{A+D}, C_A^{\phi_i}) - \mathcal{L}_B^E(R_{A+D}) \tag{15}$$

which is equivalent to

$$\mathcal{I}_B^B(R_{A+D}) - (\pi_{B'}(\mathcal{I}_B^{B+E}(R_{A+D})) \cap \mathcal{I}_B^B(R_{A+D})) \tag{16}$$

in a complete decision table and immediately come to our definition of the B-boundary region

$$\mathcal{B}_B^E(R_{A+D}) := \mathcal{I}_B^B(R_{A+D}) - \pi_{B'}(\mathcal{I}_B^{B+E}(R_{A+D})) \tag{17}$$

where $B' = \{card, b_1, ..., b_k\}, b_j \in B, 1 \leq j \leq k$. Denote, we directly deduced $\mathcal{L}_B^E(R_{A+D})$ and $\mathcal{B}_B^E(R_{A+D})$ from (8) and (9). For practical reasons, further simplification can be applied by removing the π-operator. One may verify, this change still preserves the exact same result set, because both expressions rely on $\mathcal{I}_B^B(R_{A+D})$ initially.

6 Experimental Results

In this section, we present the initial experimental results applying the concluded expressions from Section 5.3 to some well-known data sets and two database systems. The objective of this experiment is to demonstrate the performance of our model in a conservative test environment not utilizing major optimization steps such as the application of indices, table partitioning or compression strategies. Thus, we get an impression of how the model behaves natively in different databases. We chose PostgreSQL (PSQL) and Microsoft SQL Server (MSSQL) as two prominent engines providing us with the required relational operations. The hardware profile[2] represents a standalone server environment commonly used in small and medium-sized organizations. Most of our benchmark data sets are extracted from [22] varying in data types and distribution. Table 1 states further details. Both, PSQL and MSSQL provide similar query plans based on hash

[2] OS: Microsoft Windows 2012 R2 (Standard edition x64); DBs: Microsoft SQL Server 2014 (Developer edition 12.0.2, 64-bit), PostgreSQL 9.4 (Compiled by Visual C++ build 1800, 64-bit); Memory: 24 GByte; CPU: 16x2.6 GHz Intel Xeon E312xx (Sandy Bridge); HDD: 500 GByte

Table 1. Summarized characteristics of the assessed data sets

Data set	*Records*	$\|A\|$	$\|D\|$	$\|IND_A\|$	$\|IND_D\|$	$\|C_A\|$
HIGGS [24]	11.000.000	28	1	10.721.302	2	5.829.123
RLCP [25]	5.749.132	11	1	5.749.132	2	5.728.201
SUSY [24]	5.000.000	18	1	5.000.000	2	2.712.173
KDD99	4.898.431	41	1	1.074.974	23	2.807.886
KDD99_M	4.898.431	42	1	1.075.016	23	2.807.886
PAMAP2 [26]	3.850.505	53	1	3.850.505	19	1.125.552
Poker Hand	1.025.010	10	1	1.022.771	10	511.308
Covertype [23]	581.012	54	1	581.012	7	297.711
NSL-KDD [27]	148.517	41	1	147.790	2	71.361
Spambase	4.601	57	1	4.207	2	1.810

algorithms which we review briefly to understand the priciples: The initial stage consists of scanning two input sources from disk followed by *hash aggregations*. Finally, both aggregated inputs are fused using the *hash join* operator. Denote, a hash aggregation only requires one single scan of the given input to build the resulting hash table. The hash join relies on a build and probe phase where essentially each of the two incoming inputs is scanned only once. In comparison to other alternatives, these query plans perform without sorting, but require memory to build up the hash tables. Once a query runs out of memory, additional buckets are spilled to disk, which was not the case throughout the series of experiments. Even though both engines share similar algorithms, MSSQL is capable of running the queries in parallel while PSQL covers single core processing only. In general, we realized a very high CPU usage which is characteristic for the performance of our model. However we further observed that MSSQL does not scale well processing KDD99, because it is unable to distribute the workload evenly to all threads. We relate this issue to the lack of appropriate statistics in the given raw environment including its data distribution, where three equivalence classes represent 51% of all records. Therefore, we introduce a revised version called KDD99_M. In contrast, it holds an additional condition attribute splitting huge classes into chunks of 50K records. Note, this change does not influence the approximation, but results in a speed up of 76%. Further details of the runtime comparison are given in Figure 2. Summarized, we could achieve reasonable responses without major optimization steps. In particular, our model scales well appending additional cores in 9 out of 10 tests. Supported by this characteristic, MSSQL computes most queries within few seconds.

7 Future Work

The evaluation of the previous section shows how our RST model behaves in a native relational environment. However, further practical experiments are required, which we will address in the near future. In our domain of interest, i.e. network and data security, we will study classical as well as modern cyber attack scenarios in order to extract significant features of each single attack in both IPv4 and IPv6 environments. Our model is most suited for that subject, because it is designed to process huge amounts of data efficiently and can han-

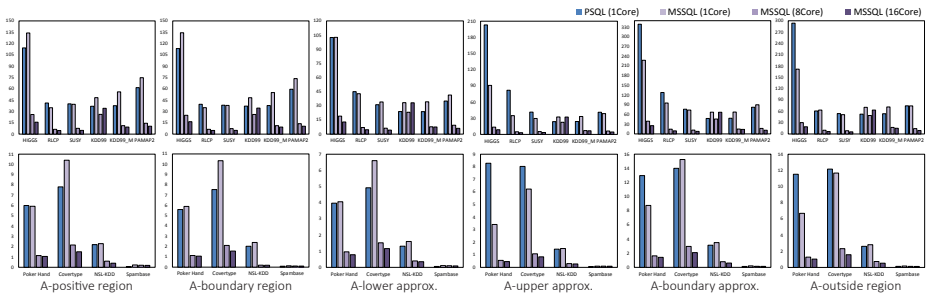

Fig. 2. Runtime comparison of the proposed rough set model in seconds

dle uncertainty which is required for proper intrusion detection. Additionally, we will use the outcome of our model to generate precise and characteristic attack signatures from incoming traffic and construct a rule-based classifier. Enabled by in-database capabilities, we can compute the resulting decision rules in parallel and integrate that approach into our existing data store. Hence, we can avoid huge data transports which is crucial for our near real time system.

8 Conclusion

In the past, the traditional Rough Set Theory has become a very popular framework to analyze and classify data based on equivalence relations. In this work we presented an approach to transport the concept approximation of that theory to the domain of relational databases in order to make use of well-established and efficient algorithms supported by these systems. Our model is defined on complete data tables and compatible with data inconsistencies. The evaluation on various prominent data sets showed promising results. The queries achieved low latency along with minor optimization and preprocessing effort. Therefore, we assume our model is suitable for a wide range of disciplines analyzing data within its relational sources. That given, we introduced a compact mining toolkit which is based on rough set methodology and enabled for in-database analytics. Immediately, it can be utilized to efficiently explore data sets, expose decision rules, identify significant features or data inconsistencies that are common challenges in the process of knowledge discovery in databases.

Acknowledgments. The authors deeply thank Maren and Martin who provided expertise and excellent support in the initial phase of this work.

References

1. M. Gawrys, J. Sienkiewicz: RSL - The Rough Set Library - Version 2.0. Technical report, Warsaw University of Technology (1994).
2. I. Düntsch, G. Gediga: The Rough Set Engine GROBIAN. In: Proc. of the 15th IMACS World Congress, pp. 613–618 (1997).
3. A. Ohrn, J. Komorowski: ROSETTA - A Rough Set Toolkit for Analysis of Data. In: Proc. of the 3rd Int. Joint Conf. on Information Sciences, pp. 403–407 (1997).

4. J.G. Bazan, M. Szczuka: The Rough Set Exploration System. TRS III, LNCS, vol. 3400, pp. 37–56 (2005).
5. M.C. Fernandez-Baizán, E. Menasalvas Ruiz, J.M. Peña Sánchez: Integrating RDMS and Data Mining Capabilities using Rough Sets. In: Proc. of the 6th Int. Conf. on IPMU, pp. 1439–1445 (1996).
6. A. Kumar: New Techniques for Data Reduction in a Database System for Knowledge Discovery Applications. JIIS, vol. 10(1), pp. 31–48 (1998).
7. X. Hu, T.Y. Lin, J. Han: A new Rough Set Model based on Database Systems. In: Proc. of the 9th Int. Conf. on RSFDGrC, LNCS, vol. 2639, pp. 114–121 (2003).
8. C.-C. Chan: Learning Rules from Very Large Databases using Rough Multisets. TRS I, LNCS, vol. 3100, pp. 59-77 (2004).
9. H. Sun, Z. Xiong, Y. Wang: Research on Integrating Ordbms and Rough Set Theory. In: Proc. of the 4th Int. Conf. on RSCTC, LNCS, vol. 3066, pp. 169-175 (2004).
10. T. Tileston: Have Your Cake & Eat It Too! Accelerate Data Mining Combining SAS & Teradata. In: Teradata Partners 2005 "Experience the Possibilities" (2005).
11. Z. Pawlak: Rough Sets. Int. Journal of Computer and Information Science, vol. 11(5), pp. 341–356 (1982).
12. Z. Pawlak: Rough Sets - Theoretical Aspects of Reasoning about Data (1991).
13. Z. Pawlak: Information Systems - Theoretical Foundations. Inform. Systems, vol. 6(3), pp. 205–218 (1981).
14. F. Machuca, M. Millan: Enhancing Query Processing in Extended Relational Database Systems via Rough Set Theory to Exploit Data Mining Potentials. Knowledge Management in Fuzzy Databases, vol. 39, pp. 349–370 (2000).
15. H.S. Nguyen: Approximate Boolean Reasoning: Foundations and Applications in Data Mining. TRS V, LNCS, vol. 4100, pp. 334–506 (2006).
16. U. Seelam, C.-C. Chan: A Study of Data Reduction Using Multiset Decision Tables. In: Proc. of the Int. Conf. on GRC, IEEE, pp. 362–367 (2007).
17. S. Naouali, R. Missaoui: Flexible Query Answering in Data Cubes. In: Proc. of the 7th Int. Conf. of DaWaK, LNCS, vol. 3589, pp. 221–232 (2005).
18. T. Beaubouef, F.E. Petry: A Rough Set Model for Relational Databases. In: Proc. of the Int. Workshop on RSKD, pp. 100–107 (1993).
19. L.-L. Wei, W. Zhang: A Method for Rough Relational Database Transformed into Relational Database. In: Proc. of the Int. Conf. on SSME, IEEE, pp. 50–52 (2009).
20. D. Slezak, J. Wroblewski, V. Eastwood, P. Synak: Brighthouse: An Analytic Data Warehouse for Ad-hoc Queries. In: Proc. of the VLDB Endowment, vol. 1, pp. 1337–1345 (2008).
21. T.Y. Lin: An Overview of Rough Set Theory from the Point of View of Relational Databases. Bulletin of IRSS, vol. 1(1), pp. 30–34 (1997).
22. K. Bache and M. Lichman: UCI Machine Learning Repository. University of California, Irvine, http://archive.ics.uci.edu/ml (June, 2015).
23. J.A. Blackard, D.J. Dean: Comparative Accuracies of Neural Networks and Discriminant Analysis in Predicting Forest Cover Types from Cartographic Variables. In: Second Southern Forestry GIS Conf., pp. 189–199 (1998).
24. P. Baldi, P. Sadowski, D. Whiteson. Searching for Exotic Particles in High-energy Physics with Deep Learning. Nature Communications 5 (2014).
25. I. Schmidtmann, G. Hammer, M. Sariyar, A. Gerhold-Ay: Evaluation des Krebsregisters NRW Schwerpunkt Record Linkage. Technical report, IMBEI (2009).
26. A. Reiss, D. Stricker: Introducing a New Benchmarked Dataset for Activity Monitoring. In: Proc. of the 16th ISWC, IEEE, pp. 108–109 (2012).
27. NSL-KDD: Data Set for Network-based Intrusion Detection Systems. http://nsl.cs.unb.ca/NSL-KDD (June, 2015).