# A Study on Rule Based Reporting Systems

by

**Prodip Kumer Das**

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science & Engineering

Department of Computer Science and Engineering
Khulna University of Engineering & Technology
Khulna-9203, Bangladesh

**December 2018**

## Declaration

This is to certify that the thesis work entitled "A Study on Rule Based Reporting Systems" has been carried out by Prodip Kumer Das in the Department of Computer Science and Engineering, Khulna University of Engineering & Technology, Khulna, Bangladesh. The above thesis work or any part of this work has not been submitted anywhere for the award of any degree or diploma.

| | |
|---|---|
| Supervisor | Candidate |
| [illegible] 05/12/2018 | [illegible] 5/12/18. |
| Prof. Dr. M.M.A. Hashem | Prodip Kumer Das |
| | Roll: 1207508 |

## Approval

This is to certify that the thesis work submitted by Prodip Kumer Das entitled "A Study on Rule Based Reporting Systems" has been approved by the board of examiners for the partial fulfillment of the requirements for the degree of Master of Science in Computer Science & Engineering in the Department of Computer Science and Engineering, Khulna University of Engineering & Technology, Khulna, Bangladesh in December 2018.

**BOARD OF EXAMINERS**

1. [illegible] 05/12/2018

Dr. M.M.A. Hashem
Professor
Department of Computer Science and Engineering
Khulna University of Engineering & Technology, Khulna-9203

Supervisor
(Chairman)

2. Aminul 05/12/18

Dr. Md. Aminul Haque Akhand
Head & Professor
Department of Computer Science and Engineering
Khulna University of Engineering & Technology, Khulna-9203

Member

3. [illegible] 05.12.18

Dr. K.M. Azharul Hasan
Professor
Department of Computer Science and Engineering
Khulna University of Engineering & Technology, Khulna-9203

Member

4. [illegible] 05.12.18

Dr. Kazi Md. Rokibul Alam
Professor
Department of Computer Science and Engineering
Khulna University of Engineering & Technology, Khulna-9203

Member

5. [illegible] 5.12.18

Dr. Md. Anisur Rahman
Professor
Computer Science & Engineering Discipline
Khulna University, Khulna

Member
(External)

## Acknowledgment

First of all, I would like to express my grateful thanks to the almighty to complete my task.

I would like to express my deepest sense of gratitude and sincere thanks to my respected supervisor Dr. M.M.A. Hashem, Professor, Department of Computer Science & Engineering, Khulna University of Engineering & Technology, Khulna, Bangladesh for providing me the opportunity of working under his kind supervision. For his proper guidance, co-operation, invaluable suggestions and constant encouragement throughout this research work it is easy to find the right way to fulfill the desired research. I will remember his inspiring guidance and cordial behavior forever in my future life.

I am pleased to express my gratitude to Dr. Md. Badiul Islam, Research Scientist, Business Processes & Legal Informatics Team, DATA61, CSIRO, Brisbane, Australia for guiding me as my co-supervisor for my research work. I should take this opportunity to express my sincere thanks to all teachers and staffs of this department for their valuable advice and moral support in my research work.

I would like to express my gratitude for Prof. Dr. Bashudeb Chandra Ghosh, Director, Institute of Information & Communication Technology (IICT) for his cordial support for my research work within the job responsibilities.

I wish to convey my hearty thanks to all my friends and class fellows especially Tumpa Rani Roy, Md. Rashedul Islam for helping me according to their ability.

I wish to thank my parents and wife for their great understanding and support.

# Abstract

Government enacts rules & regulations. People are bound to follow those rules and regulations in their everyday life. Human brains do the computation for generating conclusions to take decisions on different circumstances applying those rules. A Human being does not need any external rule reasoner. However, nowadays data is generated by different automated systems, and inserted, updated, deleted in different database design formats. So, the decision-making becomes difficult by generating required conclusions from large repositories of data having multiple formats. Therefore, artificial intelligence in form of the Rule Based System can help to mitigate this problem by analyzing the conclusions from the given facts. But, the existing Rule Based Systems processes the adverse events from only one data sources at a single time and manually generates the facts. In this circumstance, the study proposes an automated Rule Based System, which is capable of generating conclusions as output for the given input queries. The study results in a database independent Rule Based method with dynamic predicate generation & translation for analyzing the stored adverse events from multiple databases by applying the predefined set of rules and regulations. The formal rules are generated using Defeasible Logic (DL) to efficiently handle the logic-based implementation of the proposed methodology. In addition, the proposed system includes SPINdle rule engine to generate required conclusions by loading the formal rules that are generated using the DL. The resulting Rule Based Reporting System (RuleRS) integrates simultaneous reasoning generation from the adverse events along with the existing Rule Based Systems to get the argumentation from multiple relational databases. It also supports the easy integration of additional data sources for generating assumptions. The output is sent to the I/O section for showing the generated reports. The study also conducted an empirical evaluation for the single sources (i.e., FAERS & ChildSafe database separately) and proposed RuleRS in reasoning simultaneously from multiple sources of data (i.e., FAERS & ChildSafe database concurrently). The evaluation result is promising to integrate the developed system with the existing RuleRS.

# Contents

## List of Tables

# List of Figures

# Nomenclature

| | |
|---|---|
| **DBMS** | **D**ataBase **M**anagement **S**ystem |
| **RDBMS** | **R**elational **D**ataBase **M**anagement **S**ystem |
| **SQL** | **S**tructured **Q**uery **L**anguage |
| **DL** | **D**efeasible **L**ogic |
| **RuleRS** | **R**ule **B**ased **R**eporting **S**ystem |
| **DBI-DeLP** | **D**atabase **I**ntegration for **D**efeasible **L**ogic **P**rogramming |
| **DeLP** | **D**efeasible **L**ogic **P**rogramming |
| **PRTDB** | **P**redicate **T**ranslation **D**atabase |
| **DSR** | **D**ata **S**ource **R**etrieval [function] |
| **PRF** | **P**resumption **R**etrieval **F**unction |
| **DS** | **D**ata **S**ource |
| **IDE** | **I**ntegrated **D**evelopment **E**nvironment |
| **JDK** | **J**ava **D**evelopment **K**it |
| **JSP** | **J**ava **S**erver **P**ages |
| **GUI** | **G**raphical **U**ser **I**nterface |

*Dedicated to my sweet daughter, parents, wife and honorable supervisor & co-supervisor sir. . .*

# Chapter I
# Introduction

## 1.1 Background

Government enacts various rules and regulations. Public & private organizations, scientific research institutes, and other related agencies are obliged to follow those rules. Rules and regulations of public & private organizations are used to regulate the activities of organizational behaviors. They are also part of business processes, operations, and practices so that they are obliged to follow a set of rules [4] [5]. So, rules and regulations are enforced to process commercial, non-commercial and scientific data source (FDA adverse events (FAERS) [6] and Childcare database (ChidlSafeDB) [7] for this experiment) and the experimental result can be used to take prospective actions or decisions.

In everyday life, a specific business strategy is obliged by various organizations [8]. Business and government organizations annually spend crores of taka to develop and maintain Rule Based information gathering systems. Rule Based Systems are autonomous because they can understand context changes and respond accordingly without user intervention [9]. The study on Rule Based Systems generates outcome which can be implemented in any organization for improving the effectiveness and efficiency of that organization [10]. For example, jurisdictional Rule Based Systems are used to judge the incidents of individuals and organizations. And, the service Rule Based Systems are used to recruit job seekers to different vacant posts. Also, all the posts must meet different criteria. Most of the rules are written in the paper. So, taking decisions using Rule Based Systems with strict rules and regulations may result in minimizing the time consumption and increase efficiency.

US Food and Drug Administration (FDA) takes initiatives for the protection of peoples health with safety, efficacy, and security of radiation emitted drugs and equipment [6]. FDA always try to minimize the threats to human health by any means. And, the childcare database stores the information regarding Childcare attendance, Childcare facility characteristics, child and family characteristics, and hospitalizations [7]. Also, the Child Care Online Management System preserves data (adverse events) in the relational database (i.e., ChildSafeDB) to detect the Child's abuse information. The study uses data from the U.S. Food and Drug Administration (FDA) and Childcare management System (ChildSafeDB) for designing the proposed Rule Based Reporting System to create reports based on a set of rules. The data from both the databases are selected for the nature of adverse events. So that, we can research various study using compliances to process available data using Rule Based Reporting Systems. Through analysis is completed on any patient using a set of pre-defined and custom conditions. In summary, the proposed Rule Based Reporting System is able to process the adverse events from all the available databases using SPINdle rule reasoner[11][3], defeasible logic and a pre-defined set of compliances.

The idea of PLIS+ applications [8]which is a Rule Based Personalized Location Information System and Personalization of Rule Based Web Services [12] [13] is the main encouragement of this Rule Based study. The development of a Rule Based Expert System model for fraud alert in consumer credit [9] was designed for secure banking transactions. These Rule Based Systems used to monitor and analyze the data over certain rules. And, they suggest the circumstances on the occurred event.

And, in Database Integration for Defeasible Logic Programming (DBI-DeLP), the rule reasoned is a DBI-DeLP server which provides domain logic, strict rules, defeasible rules, and other operators to process argumentation over the data from relational databases. They also have a DBI-DeLP core which is a base for the operational tasks of DBI-DeLP server [14]. Also, this research idea guides to introduce the features of Predicate Translation Database (PRTDB) in the existing Rule Based Reporting Systems. Moreover, the Data Source Retrieval (DSR) function and Presumption Retrieval Function (PRF) is introduced as part of the Domain Data Integrator (DDI) which is already described in [14].

Our study on the proposed Rule Based Reporting System (RuleRS) [1] on adverse events integrates rules and regulations [15] for generating reports and alerts from generated data of this technology-laden world [16]. Most of the organizational data are stored in databases (i.e., relational databases). The study also ensures that the decisions can be made over different formats of data for different organizational data sources. The data from different sources are automatically selected based on the requirements of the users which is then compiled through the provided guidelines for generating Rule Based assumptions.

## 1.2 Motivation

Application data is generated from various types of applications which are used to manage the organizational resources. In most of the cases, the generated data is stored in relational databases. The decision makers need to make decisions analyzing the stored data using defined rules & regulations. Among them, the Rule Based System can provide a specific solution to the problem at hand by using rules & regulations in processing stored adverse events. There are many Rule Based Systems, which can generate conclusions from the stored adverse events. But, most of them do not use SPINdle rule engine and Defeasible Logic (DL) for reasoning from data sources. We have studied that [1][2] uses SPINdle rule engine and Defeasible Logic (DL) for efficient reasoning from a large repository of data. But, none of them supports simultaneous reasoning from multiple data sources using SPINdle rule engine, Defeasible Logic. Also, the relational database schema is different from one organization to another. So, decision making from multiple sources becomes difficult in those cases. The above-illustrated problems guided us to the study to generate simultaneous conclusions from multiple data sources(introducing database independence) using SPINdle rule engine, Defeasible Logic and Database Integration for DeLP (DBI-DeLP) framework which efficiently solves the problem at hand.

## 1.3 Problem Statement

The hard part is to analyze the data from completely different relational databases having different formats. And, there are difficulties in integrating two or more databases together for analyzing the required conclusions simultaneously. In addition, the study requires to reason from a large repository of data. Also, the pre-defined rules & regulations are required to be applied to make decisions from the available data sources. There exists logic based conflict when trying to implement the analysis of data using rules. Moreover, the development of source code requires prerequisite software and the knowledge of JAVA programming language. All the above-mentioned difficulties are taken in action during the implementation of the Rule Based Reporting System to generate simultaneous conclusions from multiple data sources.

## 1.4 Objectives

The growing need for data analysis for conclusions and reporting is a great problem nowadays. There are a huge amount of structured and unstructured data sets. Data are generated by Government organizations, professional organizations, educational institutions, scientific research, sensors and more. The study on Rule Based Reporting System (RuleRS) covers the integration of rules and regulations for making a decision over adverse events data. Therefore, our main objective of this study is to generate simultaneous reports and alerts from stored adverse events in multiple databases using DL and SPINdle rule reasoner applying a set of rules.

However, the existing systems like PLIS+ applications[8], Rule Based Expert System model[13] and others do not focus on using DL and SPINdle rule reasoner for adverse events along with the rules and regulations. But, the Rule Based Online Management System [1]and the Rule Based Reporting Systems [2] provides a better solution for reporting on the adverse events. But the above-mentioned Rule Based Systems do not generate simultaneous conclusions from multiple data sources. So, the objective of this study on Rule Based Reporting System can be stated as follows:

- To achieve the database independent (reasoning from multiple databases) analysis of adverse events using Rule Based Reporting Systems. So that, any type of data can be processed through the proposed system, no matter what is the database.
- To gather the required formal rules and regulations for processing adverse events from multiple data sources (i.e., FAERS & ChildSafe database).
- To use additional data(adverse events) stored in databases of Relational Databases Management Systems (RDBMS) (i.e., PostgreSQL).
- To generate facts which are used as predicates from the available database records that are saved in a file with SQL & JSON file format using Defeasible Logic.
- To apply the pre-defined rules and regulations to generate reports from available data sources.
- To generate conclusions using the SPINdle rule reasoner utilizing pre-defined rules and regulations to send to the I/O interface for visualization.
- To generate Rule Based Report for the time consumption for the data analysis to make best decisions from FAERS & ChildSafe database jointly and separately.
- To minimize the risks involved in this Rule Based implementation of the proposed Rule Based Reporting System.

So, the study on Rule Based Reporting Systems focuses on achieving the aforementioned objectives to satisfy the user requirements using Rule Based Systems.

## 1.5 Scope of the Thesis

Why the study have introduced the database independence(reasoning from multiple data sources) feature extraction method to the existing Rule Based systems? The existing Rule Based Reporting Systems focuses on achieving the goal by formatting only one database information and the rules against that. But, the database independent model generates the rules according to the user requirements and the facts are also defined dynamically to increase the user-friendliness of the system. The study uses two available databases as FAERS & ChildSafe for taking decisions from stored adverse events. So, we designed a system which can accommodate two or more databases at a time to generate assumptions. The study also focuses to use Defeasible Logic (DL) to protect the logic based error propagation and the SPINdle rule engine for generating conclusions according to the provided guidelines as a theory. Finally, the Database Integration for Defeasible Logic Programming (DBI-DeLP) framework is included to introduce the database independence feature with the proposed Rule Based Reporting Systems.

## 1.6 Contributions

The study focuses on the development of a new Rule-Based Reporting System, which efficiently integrates multiple databases for simultaneous reasoning. The study mitigates the problem of generating conclusions from only a single data source at a single time. The proposed method extends the capabilities of the existing Rule Based Systems which is compatible to generate assumptions from the multiple data sources. It can accommodate a large repository of data for reasoning. So, the enterprises are facilitated with decision making instantly from multiple sources using defined rules & regulations. The contributions for the study on Rule Based Reporting Systems is given below.

- The study introduces the integration of multiple data sources (i.e., database independence) in Rule Based Reporting Systems.
- The study integrates multiple relational databases with different formats.
- The study utilizes DBI-DeLP framework's Data Source Retrieval (DSR) function and Presumption Retrieval Function (PRF) for implementing database independent Rule Based Reasoning System.
- The study uses Defeasible Logic (DL) [17] which is a nonmonotonic formalism [18] [19] based on rules and a priority relation on them. So, the DL generates rules with precedence from the stored data.
- The study uses SPINdle as the Rule Based reasoner with the Defeasible Logic (DL) theories to process the conclusions from different sources of relational databases.
- The study uses SPINdle which supports reasoning with ambiguity propagation and well-founded semantics.
- The study uses the I/O section to visualize the generated conclusions from the Rule Based Reporting Systems.

## 1.7 Organization of the Thesis

This thesis is organized into eight chapters and an appendix A (for programming implementation of the functions). The following chapter briefly describes the related works for the study on the Rule

Based Reporting Systems. The third chapter explains the theoretical considerations which include descriptions and usage policy for the defeasible logic and SPINdle rule engine. The following chapter briefly describes the existing RuleRS architecture with I/O interface, predicate generation, and formal rules. The fifth chapter explains the developed methodology for the integration of new features(i.e., simultaneously generating assumptions from multiple databases) with the existing Rule Based Reporting Systems (RuleRS). The next one describes the implementation of the newly developed methodology for the Rule Based Reporting Systems. The seventh chapter describes the experimental setup for running and testing the developed RuleRS application using the computational resources. Finally, evaluation results are illustrated for the developed Rule Based Reporting System with conclusions future extension of the proposed study on the Rule Based Reporting Systems.

# Chapter II

# Literature Review

The study on Rule Based Systems for the analysis of adverse events enables the possibility to enhance the relationship between data and processing systems. Also, rules can be applied simultaneously to different formats of databases to fetch the conclusions for making decisions on adverse events.

## 2.1 Related Works

The Rule Based study started the RuleRS architecture development by describing the analysis of the fact that various business, enterprises, and agencies may require to generate decisions and reports (along with other obligation) based on regulations and policies align with their existing database records. Whereas traditional Database Management Systems (DBMSs) are not active in the sense that their overall focus has been to execute commands (e.g., query, update, delete) as and when requested by the user or application program [20], rule engines cooperate with the rule repository and the underlying database.

### 2.1.1 Rule Engines

Prior approaches for rule engines can be classified into one of five types as Prolog-based, production and reactive rules, deductive databases, triple engines, and knowledge bases [21]. The prolog-based system is defined as a top-down inference engine. While various applications have been used prolog-based system (e.g., XSB [1]; Yap[2]), their common focus has been to deliver a complete environment for building applications [21]. Deductive databases, which are the extension of the Database Management System (DBMS), typically Structure Query Language (SQL) and design storage around a logical model of data [22]. These approaches can be seen in various systems: DLV[3]; IRIS[4]; and Ontobroker[5]. Production and reactive Rule Based Systems are the bottom-up engines where rules have actions in the consequent [21]. All of these systems (e.g., Drools[6]; Jess [7] Prova[8]) are based on (a version of) the Rete algorithm [21]. Triple engines have been used two rule engines: a bottom-up engine and a top-down one for Semantic Web applications [21]. A few prior systems that used rule engines based on triples are Jena[9], SwiftOWLIM[10] and BigOWLIM [11]. Knowledgebase engine such as CYC[12] is an integration of domain

[1]`http://xsb.sourceforge.net` accessed on 14 November 2018
[2]`http://www.dcc.fc.up.pt/~vsc/Yap/` accessed on 14 November 2018
[3]`http://www.dlvsystem.com/` accessed on 15 November 2018
[4]`http://sourceforge.net/projects/iris-reasoner/` accessed on 15 November 2018
[5]`http://www.semafora-systems.com/en/products/ontobroker/` accessed on 16 November 2018
[6]`http://www.drools.org` accessed on 16 November 2018
[7]`http://www.jessrules.com/jess/index.shtml` accessed on 16 November 2018
[8]`https://prova.ws` accessed on 16 November 2018
[9]`http://jena.apache.org` accessed on 16 November 2018
[10]`https://confluence.ontotext.com/display/OWLIMv35/SwiftOWLIM+Introduction` last accessed on 16 November 2018
[11]`https://confluence.ontotext.com/display/OWLIMv35/BigOWLIM+Introduction` accessed on 16 November 2018
[12]`http://www.cyc.com` accessed on 16 November 2018

knowledge for solving classical and common sense reasoning based complex problems. According to Liang et al. [21], although Ontobroker was not the best system in all of the mentioned Rule Based System, Ontobreaker (and partly DLV) was assigned to be the best performing systems.

Moreover, Rule Based Systems have great potential for applied research. One of the first popular computational uses of Rule Based Systems was the work by Newell and Simon on the General Problem Solver [23]. In this work, production rules were used to model human problem-solving behavior. However, the mathematical model of production systems was used much earlier by Post in the domain of symbolic logic [24].

### 2.1.2 Datasets

Data is generated from different sources and they are stored in different formats. A large repository of adverse events requires intelligence in the processing system to generate the required outcome. The data sources of an adverse event for data mining of the public version of the FDA Adverse Event Reporting System (AERS) which has a reproducibility of clinical observations[25]. And, the strengths & limitations for the FDA adverse events are stated in [26]. Also, the childcare database which is a valuable register linkage [7] is a large repository of data used to implement the proposed method.

Integration of knowledge-based systems with the databases [27] results in an expert system for processing adverse events. The integration of the Rule Based Expert System, a case-based reasoner and an ontological knowledge-base [28] can process conclusions from large repositories of data. The Rule Based System follows the rich model of business process compliance [29] to prepare the required output. There has been researching for generating conclusions based on normative requirements for regulatory compliance [30].

### 2.1.3 Rule Based Systems

Rule Based Systems are designed to generate conclusions for enterprise databases. We have studied a practical introduction to Rule Based Expert Systems [31] which provided the idea of the Rule Based System and the required methods for analyzing data. We have learned about Rule Based Systems working procedure from research on Personalized Location Information System (PLIS+) application[8] and Rule Based Web Services [12] [13]. Also, a Rule Based Expert System model for fraud alert in consumer credit [9] helps us to learn secure banking transactions through Rule Based Systems. A Rule Based diagnosis system for diagnosis of diabetes [32] helped to understand the utilization of Rule Based Systems in the area of medical services. And, the validation algorithm [33] guided us to the algorithmic implementation of the proposed Rule Based System and its performance over traditional systems. A thorough research is completed on Rule Based Online Management System (RuleOMS) [1] & Rule Based Reporting Systems (RuleRS) [2] which lacks the feature of integrating multiple sources of data using dynamically generated predicates. But the existing Rule Based Systems only work by accessing a single source of data at a time. To accommodate the extended feature of the Rule Based System, we propose a model which can integrate multiple sources with dynamically generated facts.

### 2.1.4 Defeasible Logic

Defeasible Logic is the base for implementing Rule Based Reporting Systems. It is already proved that reasoning using Defeasible Logic results in better decisions making over large dataset. Defeasible theory can be used as a guide in the construction of a theory for defeasible reasoning, and a computer program implementing that theory [17]. The generated defeasible theory deals with a subset of defeasible reasoning [17]. Most of the reasoning of human actions can be denoted as defeasible reasoning [34]. Since around 1980, considerable research in AI has focused on how to model reasoning of this sort. The defeasible theory can be used for normative reasoning, learning, planning, and other types of automated reasoning. Defeasible theory can be used as argument to embed in different complex systems for real-world applications, and allow more formal work to be done in different circumstances, such as AI and Law, case-based reasoning and negotiation among intelligent agents [35].

### 2.1.5 SPINdle Rule Reasoner

The Rule Based study requires a rule engine because there have been prior studies which have established various advantages of improving systems using rule engines[13]. The key aspects of using Rule Based Systems can be listed as follows: natural knowledge representation, uniform structure, separation of knowledge from its processing and dealing with incomplete and uncertain knowledge [14].

### 2.1.6 Multiple Database Integration

Moreover, in Database Integration for Defeasible Logic Programming (DBI-DeLP), the rule reasoned is a DBI-DeLP server which provides domain logic, strict rules, defeasible rules and other operators for processing argumentation over the data from relational databases. They also have a DBI-DeLP core which is a base for the operational tasks of DBI-DeLP server [14]. The model of consistent query & answering [36] and defeasible argumentation over relational databases [37] leads us to process relational data. These research outcome guided us to introduce simultaneous multiple data source access in RuleRS [1] [2] which generated the new theme of adverse event processing from multiple data sources through dynamic generation of predicates and translation into PRTDB [14] using Rule Based Systems. Also, we have generated dynamic predicates and automatically converted those predicates into the records of PRTDB using the outcome of RDB to RDF translation approaches and tools [38].

Indeed, the above-illustrated Rule Based Systems [1][2] [8][12][13][9] illustrates how the rule based systems work for retrieving information from single source of data. The [36] illustrates how we can retrieve data from multiple data sources to be processed using Rule Based Systems. The merging of these two different methods generate a new processing engine to retrieve data from multiple sources of data.

[13]`http://java.sys-con.com/node/45082` accessed on 15 November 2018

[14]`http://intelligence.worldofcomputing.net/expert-systems-articles/rule-based-expert-systems.html` accessed on 14 November 2018

# Chapter III

# Theoretical Considerations

The study utilizes the computational features of the Defeasible Logic to minimize the violations and ambiguity among the defined rules by the business organizations. And, the SPINdle rule reasoner inputs the theories generated using the logic based implementations of the Defeasible Logic and generate the conclusions which are visualized using the I/O interface of the proposed architecture for the Rule Based Systems.

## 3.1 Defeasible Logic

The study integrates database independent adverse event analysis mechanism using Rule Based System that uses Defeasible Logic (DL) [17] [16] which is a non-monotonic formalism based on rules and a priority relation on them. Non-monotonic feature of DL can deal with potential conflicts among knowledge items [39]. DL helps to meet the strict requirement of the data analysis. The study uses DL as a formal framework to model protocols and strategies for automated negotiation [40]. It has been significantly used in the legal domain such as modeling regulations [41], e-contracting [42][43], business processes compliance [44] [45] and automatic negotiation system [40]. The DL generates rules with precedence from the stored data. The study uses the SQL syntax to fetch data from the database based on the rules and regulations. Then, formal rules are generated to process the dataset. Data is then converted into an intermediate JSON file.

Legal reasoning can be completed using Defeasible Logic (DL). Because [46] states a strong logic that DL is better than other available formalisms in reasoning for legal environments. It has been proved that Defeasible Deontic Logic is widely used for the developed applications for legal reasoning[47, 42, 45, 48]. Moreover, [49, 48] states that the Defeasible Deontic Logic is free from the logic based error propagation due to action of one logic with another logic for reasoning using legal norms & compliances. Defeasible Deontic Logic prepares a concrete environment for reasoning[50] which empowers the computation of theories in linear time. Also, Defeasible Deontic Logic can be used for representing the obligations in a suitable way. This research work explains a solution to reason from multiple data sources using dynamically generated predicates. Also, the conversion is dynamically performed from SQL type predicates into records for the tables of Predicate Translation Database (PRTDB). The study proposes a feasible solution to evaluate the child's risk prediction process from ChildSafe database[51]. And, also extracted the conclusion from the adverse events of the FAERS database (FDA dataset)[52].

Donald Nute [34] proposed that the Defeasible Logic (DL) is like a “skeptical" nonmonotonic logic. Also the contradictory conclusions are not supported by the DL. Legal reasoning has been more successfully accomplished using the DL. E-contracting [42, 43], modeling regulations [41], automatic negotiation system [40], and business processes compliance [44, 29] are already implemented using the features of the DL. Moreover, the DL covers the features like “Decision support”, “Anomaly detection”, “Explanation”, “Hypothetical reasoning” and “Debugging” tasks for reasoning. So, the “Anomaly detection” is used to detect any kind of unnatural behaviors of the defeasible rules. And, the “Decision support” provides the concrete answer to the given question based on the input rules & regulations. The DL is most likely the solution to the problem of rules conflicting other related rules. [41] also agrees for the conflict resolution feature of the rules using the DL.

A *defeasible theory T* contains facts, defeasible rules, strict rules, defeaters and a superiority relation as the five kinds of knowledge. Defeasible theory is a logic program and works as a knowledge base[53]. T consists of a triple $(F, R, \succ)$. In the triple, the $F$ represents the finite sets of literals or facts, the $R$ represents the associated rules for the computation, and the $\succ$ represents the superiority relation for the associated rules $R$.

A finite set of literals are included in the language that describes the DL. For a given scenario we can represent a literal as an atomic assumption or as the negation of the assumption. Given a input literal $i$, $\sim i$ denotes its complement. That is, if $i = a$ then $\sim i = \neg a$, and if $i = \neg a$ then $\sim i = a$. Here, $i$ denotes the input literal and the $a$ denotes the computed assumption for the input literal.

Facts are like logical statements which describes the indisputable facts. The facts are represented either in the form of states of affairs (literal or modal literal (refer to section 3.1.1)) or actions that have been completed which are considered to be true for every condition. For example, “John is a human” is represented by: *human(John).*

For example, we can consider that a rule $r$ which describes the relations between the defined set of literals. And, $A(r)$ represents the antecedents for the defined rule which may be empty and $C(r)$ represents the consequence for the computation using the rule $r$. The strength associated with the rule *DL* is represented by using any of the following: *strict*, *defeasible*, and *defeater.*

In other words, *Strict rules* are concluded as strict if the premises for the rule is indisputable (e.g. a fact) then so is the conclusion. For instance,

$$\mathit{human}(X) \rightarrow \mathit{mammal}(X)$$

concludes that “All the human are mammal”.

The antecedents for the strict rules can be *empty* which can be treated as the facts. Facts describe contextual information and the rules are used for reasoning from the given input.

*Defeasible rules* are defined rules that can be defeated by opposite evidence. For instance, generally mammal cannot fly, written in formal:

$$\mathit{mammal}(X) \Rightarrow \neg \mathit{flies}(X)$$

The plan behind this is if we can know that $X$ is a mammal, then we can conclude that it is not able to fly *unless there is another rule that is not defeated and the related evidence suggesting that it is able to fly* (for instance the mammal may be a bat). A *presumption* can be fetched for a defeasible rule which contains an empty antecedent.

*Defeaters* are rules that are useless to draw any assumptions. They can be only used to restrict some assumptions, i.e., to oppose some defeasible rules by generating proof in the opposite direction. For instance, it can be represented as the following:

$$heavy(X) \rightsquigarrow \neg flies(X)$$

describes that an animal which is heavy is not enough to conclude that the heavy mammal can not fly. It is the evidence which is against the conclusion that a heavy animal can fly. In contrary, we are not able to conclude that $\neg flies$ if *heavy*, we can prevent an assumption that *flies*.

This study recommeds to briefly study [47, 54] for understanding the full explanation of the proof theory and other related descriptions. In short, suppose the rules with the conclusion $c$ prepares a group that makes competition with another group that consists the rules with conclusion $\neg c$. If the first group wins then $c$ is defeasibly provable. But, if the opposing group wins, $c$ is concluded as non-provable. To make a summary of the discussion, let's consider $T$ is a generated theory in DL (which is explained in the above discussion). An *assumption* of $T$ is defined as a tagged literal and the assumption can contain four forms that is described in the following : $+\Delta r$ concludes that $r$ can be definitely provable in $T$ (i.e. using only facts and strict rules); $-\Delta r$ concludes that it is proved that $r$ can not be definitely provable in $T$; $+\partial r$ concludes that $r$ is defeasible provable in $T$; and $-\partial r$ concludes that it is proved that $r$ can not be defeasible provable in $T$.

Strict rules are used for strict derivations that are acknowledged by using forward chaining. We can conclude that $c$ is defeasibly provable if a rule is responsible to generate $c$ as the conclusion. And, the $\neg c$ can not be proved as the conclusion in-spite of the prerequisites for a case. So, a conclusion is strictly provable if there is no other rules stronger than the defined rule and the $\neg c$ is not provable, or in other words all the other rules are weaker for proving $\neg c$ than the rules accounted for $c$.

### 3.1.1 Defeasible Deontic Logic

There are constitutive rules and prescriptive rules which are the defined type of rules for the legal reasoning. In the legal reasoning, constitutive rules are used to create a model for the terms and specified parameters for the pre-defined legal documents. For instance, defining the terms described in the constitution of a country. In contrary, prescriptive rules guides to the encoding for prohibitions, obligations, permissions etc along with the conditions which can be applied according to the legal document. Deontic operators are used to supplement the prescriptive rules which enable successful modeling of the provisions. In our study, we utilized the classifications which are defined by [55, 56]. Also, we find that [29] describes the full details for logics which have techniques related to removing constraints. The following notations are used for representing constraints and permissions:

- [P]$p$: $p$ is permitted;
- [OM]$p$: there is a maintenance constraint for $p$;[1]
- [OAPP]$p$: there is an achievement preemptive and perdurant obligation for $p$;
- [OAPNP]$p$: there is an achievement preemptive and non-perdurant constraints for $p$;
- [OANPP]$p$: there is an achievement non-preemptive and perdurant constraints for $p$;
- [OANPNP]$p$: there is an achievement non-preemptive and non-perdurant constraints for $p$.

[1]Prohibitions can be expressed as maintenance constraints with a negated content, i.e., [OM]$\neg p$.

The 'reparation chain' assists in building compensations [57]. It can be represented as an expression like the following:

$$[\mathrm{O}_1]c_1 \otimes [\mathrm{O}_2]c_2 \otimes \cdots \otimes [\mathrm{O}_n]c_n,$$

In the above expression, $[\mathrm{O}_i]$ represents an obligation, and all the $c_i$ is the content for the obligation which is modeled by the literal. The 'reparation chain' means as we have that $c_1$ is obligatory, but if the obligation of $c_1$ is violated, i.e., we have $\neg c_1$, then the violation is compensated by $c_2$ (which is then treated as obligatory). But if $[\mathrm{O}_2]c_2$ is prohibited, then the prohibition is compensated by $c_3$ which, after the violation of $c_2$, becomes obligatory, and so on.

Deontic expressions are allowed by the Defeasible Deontic Logic to be included in the content of a rule except for the 'reparation chains'. So, the rules can be represented as follows:

$$drugStore, [\mathrm{P}]sellDrug \Rightarrow [\mathrm{OM}]showLicense \otimes [\mathrm{OAPNP}]payFine$$

The above rule means that if a drug store has a license to sell drug (i.e, it is permitted to sell the drug, $[\mathrm{P}]sellDrug$), then it has a maintenance constraint to expose the license ($[\mathrm{OM}]showLicense$), if it fails to do so then it has to pay a fine ($[\mathrm{OAPNP}]payFine$). The constraint to pay the fine is non-preemptive (meaning that it cannot be paid before the violation). Binary relation over rules is introduced with the logic which is called superiority relation. The superiority relation allows us to handle all the rules having conflicting conclusions. For instance, a rule $r1$ setting a general prohibition. And, suppose $r2$ is another rule which is responsible for derogating the prohibition to permit the conclusions generated by the rule. It is a very common scenario in legal reasoning. It can be modeled by stating that $r2$ is "stronger" than $r1$ using symbols like $s > r$. If both of the rules apply, then we can say that $r2$ defeats $r1$. We have studied [42, 29, 58] for a full description of the logic and its features.

A set of rules and facts are responsible to derive the obligations , permissions and prohibitions by reasoning.

$[\mathrm{O}]p$ is an obligation where $[\mathrm{O}]$, $[\mathrm{O}_x]$ and $[\mathsf{D}_y]$ which are described below and the placeholders for the obligations which are described above can be derivable if:

1. $[\mathrm{O}]p$ is given as one of the facts, or
2. there is a rule

$$r\colon a_1, \ldots a_n \Rightarrow [\mathrm{O}_1]p_1 \otimes [\mathrm{O}_m]p_m \otimes [\mathrm{O}]p \ldots$$

   such that
   (a) for each $1 \leq i \leq n$, $a_i$ is provable, and
   (b) for all $1 \leq j \leq m$, $[\mathrm{O}_j]p_j$ and $\neg p_j$ are provable, and
   (c) for every rules

$$s\colon b_1, \ldots, b_k \Rightarrow [\mathsf{D}_1]q_1 \otimes [\mathsf{D}_l]q_l \otimes [\mathsf{D}]p'$$

   such that $p'$ is the negation of $p$, either
   i. exists $1 \leq i \leq k$ such that $b_i$ is not provable, or
   ii. exists $1 \leq j \leq l$ such that is either $[\mathsf{D}_j]q_j$ or $\neg q_j$ is not provable, or
   iii. $r$ defeats $s$.

One rule must be in action so that all the antecedents elements are provable (a), and in case conclusions are an obligation for a reparation, all the related obligations before it has to be prohibited. On that case, the prohibited obligations were in action (the obligations were provable) to have evidence that it was violated (the negation of the content of each violated obligation is provable) (b). Also, we can ensure that there is no rule for the opposite that fire (c), and if it happens, these rules becomes weaker than the rule which is responsible for the obligation that we want to conclude.

Same conditions apply for the permissions. It is represented as $[\mathrm{P}]p$ instead of $[\mathrm{O}]p$. Moreover, we can make a conclusion as $[\mathrm{P}]p$ if we can make a conclusion as $[\mathrm{O}]p$. [59, 50, 29, 58] explains more about semantics, deontic operator conversions, conflict detection & resolutions, and algorithm implementing this Rule Based System.

## 3.2 SPINdle Rule based Reasoner

SPINdle Rule Based reasoner[2] [11] is a Java-based implementation of DL [47, 34] that computes the extension of a defeasible theory. SPINdle supports Modal DL, all types of DL rule, such as facts, strict rules, defeasible rules, defeaters, and superiority. A full description of SPINdle is out of scope for the current study, we refer the interested readers to [11] for details.

### 3.2.1 SPINdle System Architecture

SPINdle is written in Java which consists of three major components (Figure: 3.1): the Rule Parser, the Theory Normalizer and the Inference Engine.

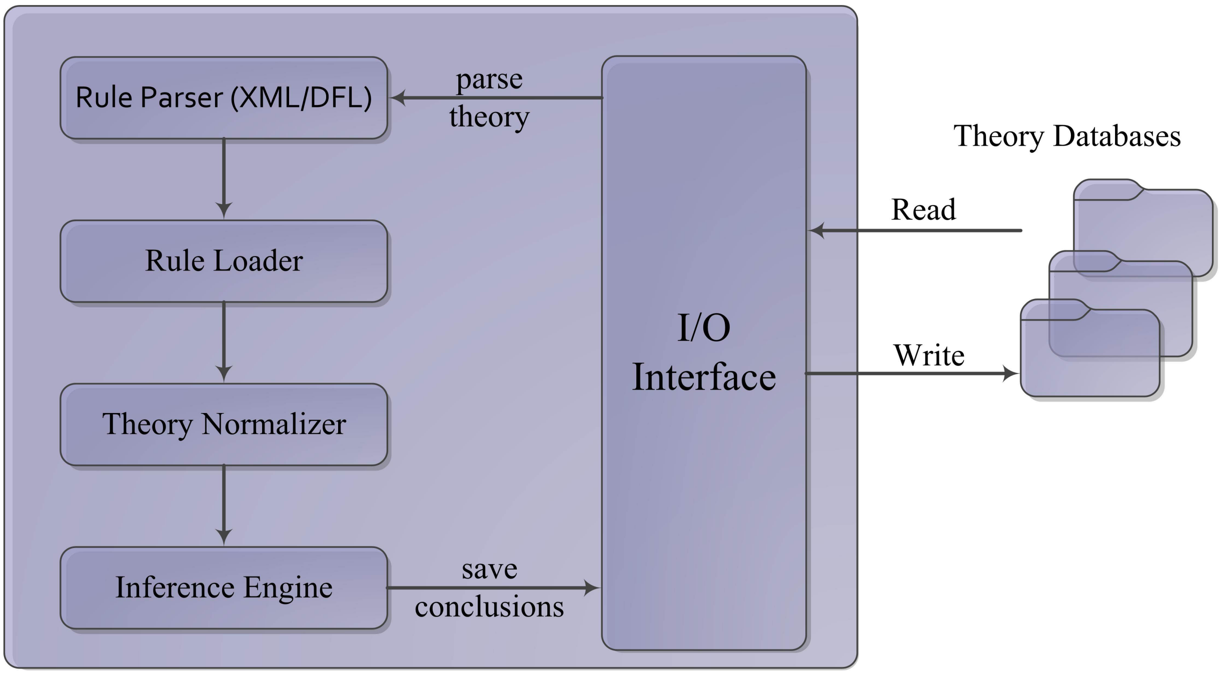


**Figure 3.1:** SPINdle Architecture

[2]SPINdle Reasoner is available to download freely from `http://spindle.data61.csiro.au/spindle/tools.html` accessed on 15 November 2018 under LGPL license agreement

SPINdle takes defeasible logic theories which is denoted using XML or plain text (with pre-defined syntax). The Rule Parser is used to convert the theory from a kept theory document into a data structure that can be handled in the next module. The nature of the rule parser is rather similar to the Logic Program Loader module of the DR-Device family of applications industrialized by [39]. After loading a theory into SPINdle, users are able to modify/manipulate the theory according to their applications requirement. The I/O Interface module in Figure: 3.1 provides useful means for helping users to load and save (altered) theory (also the resulting conclusions) to and from the database. Theories can also be transferred using XML syntaxes for agent interaction, which is a common consequence in the Semantic Web.

### 3.2.2 SPINdle Inference Process

The whole inference procedure comprises of two phases: A pre-processing stage where the study convert the theory using the systems described in [47] into a corresponding theory without superiority relation and defeaters, which later helps to simplify the reasoning procedure in the reasoning engine. The ”Theory Normalizer” module in Figure: 3.1 carry out this procedure by further breaking it down into three linear transformations: one to transform the theory to consistent form, one to void the superiority relation and one to void the defeaters. In addition to this, the theory normalizer also transforms rules with multiple heads into equivalent sets of rules with single head. It is expected that the transformed theory will produce the same sets of conclusions in the language of the theory they transform. Following [53] (a.k.a. the Delores algorithm), the assumptions origination stage (the Inference Engine module) is based on a series of (theory) transformations that allow us in two ways: (i) to assert whether a literal is verifiable or not (and the strength of its derivation) and (ii) to gradually reduce and simplify a theory. The reasoner will, in turn:

- Assert each and every fact (as an atom) as a conclusion and eliminate the atom from the rules where the atom happens positively in the body and “deactivates” (remove) the rule(s) where the atom occurs undesirably in the body. The atom is then detached from the list of atoms.
- Scanning of the list of rules for rules with an empty head. It takes the head element as an atom and searches for the rule(s) with conflicting head. If there are no such rules then the atom is involved to the list of facts and the rule will be detached from the theory. Otherwise, the atom will append to the pending conclusion list until all rule(s) with contradictory head can be verified negatively.
- Execute the first step again.
- The algorithm terminates when one among the two phases fails or when the list of facts is empty. On extinction the reasoner produces the set of assumptions.

Finally, the assumptions are either transferred to the users or saved in the theory database for using in future. Since each atom/literal in a theory is administered exactly once and every time the study scans a set of rules, the complication of the above algorithm is $\mathcal{O}(\ \mathcal{L}| * |\mathcal{R}\ )$, where $\mathcal{L}$ is the size of the distinct modal literals and $\mathcal{R}$ is the number of rules in the theory [50]. This complication result can be improved through the use of proper data structure (Figure: 3.2). For each literal p a linked-list (the dashed arrow) of the incidences of p in rule(s) can be created during the theory paring phase. Each literal occurrence has a link to the record for the rule(s) it occurs in. Using this data structure, whenever a literal update happens, the list of rules related to that literal can be reclaimed easily. Thus in its place of examining through all the rules for empty head (step 2), only rules relevant to that certain literal will be examined. The complication of the inference process will consequently be reduced to $\mathcal{O}( \mathcal{L}| * |\mathcal{M}$ ), where $\mathcal{M}$ is the highest number of rules a literal correlated within the theory.

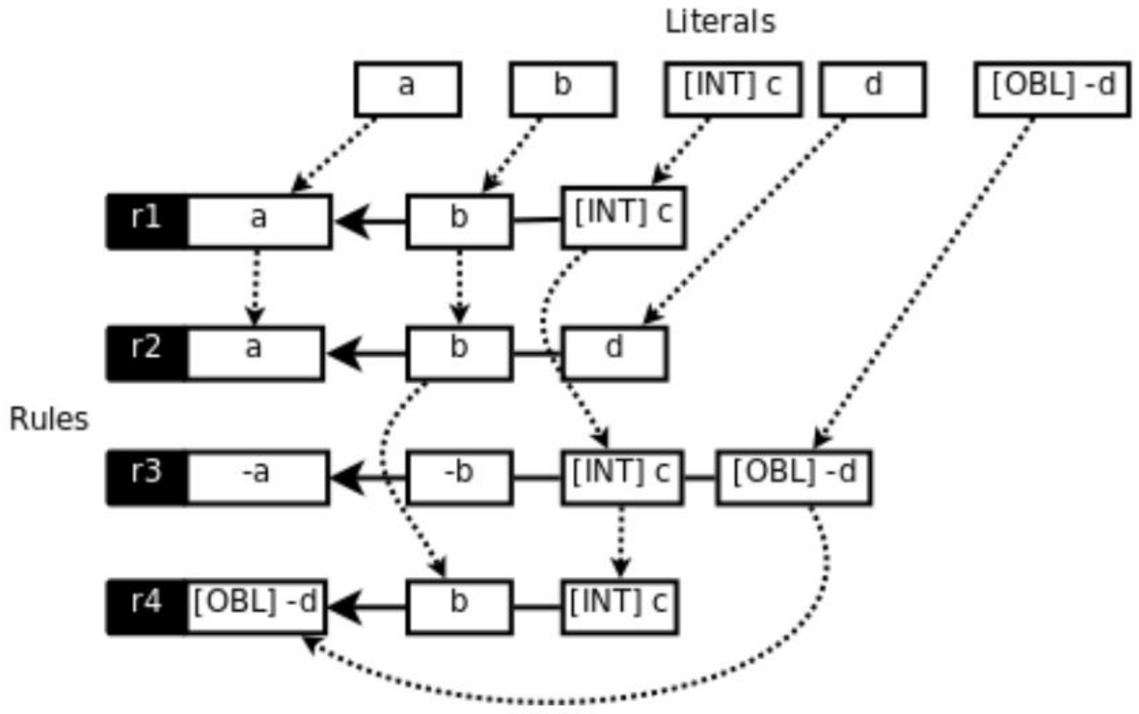


**Figure 3.2:** Data Structure for Literals-Rules Association

It is significant to note that the previously mentioned data structure is a widespread version that is common in inferencing both standard Defeasible Logic and modal Defeasible Logic. In the case of modal Defeasible Logic, an additional process adding extra rules to the theory is needed for the modal operator alterations. In addition, for a literal p with modal operator □i, likewise its complement (i.e., □i ¬p), the similar literal with the modal operator(s) in conflict with □i should also be incorporated in the conflict literal list (step 2) and only literal with the strongest modality is decided.

### 3.2.3 SPINdle Evaluation for Rule Based Systems

So, it is inevitable for implementing a business compliance checker for a business process in an enterprise or company. The engine is supposed to look at the flow of tasks, together with the constraints that each task imposed, and then selects the appropriate task as the next task to work on, or reports the error to users. Imagine further that you have coded this up in a tradition Java class. After a few weeks/months of work, some of the government regulations are changed. It is the responsibilities of your company to be in compliance with the new government regulations which may involve certain modifications to the company's workflow. If you have been using the traditional programming techniques, your code is a gnarled mess as it has to do all sorts of constraints queries testing for each task separately.

However, SPINdle is written in a way that it includes a nice clean code as the government regulations can be encoded using rules. So that later on a new constraint can be added to a task and it would be easier to add/modify the rules. These kinds of developed systems have to take into account for heterogeneous contexts in order to behave properly and effectively, but also dynamically reconfigurable whenever it is necessary. As a consequence, the study utilizes these systems to not only implement complex functionalities, but also have to embed a reasoning engine that can monitor the context in which system runs, take a decision, and report errors.

For this study, SPINdle version 2.2.4 is downloaded and included in the classpath to be recognized by the Eclipse. It grants all the theory description generated by different formalisms. The I/O manager (spindle.io.IOManager) associates different file types using their theory parsers with file name extensions. This study also set the heap size to 1024M as SPINdle requires increased heap size for the Java virtual machine (JVM) with massive number theories. The proposed study on Rule Based Systems generated conclusion of ’D’ which is showed in the following as four forms as a tagged literal.

| Symbol | | Meaning |
|---|---|---|
| +D | q | Means that q is definitely provable in ‘D’. |
| -D | q | Means that q is rejected definitely in ‘D’. |
| +d | q | Means that q is defeasibly provable in ‘D’. |
| -d | q | Means that q is defeasibly rejected in ‘D’. |

**Table 3.1:** Symbol Meaning for the Conclusions Generated by SPINdle

In summary, SPINdle is a tool which accepts rules, facts, monotonic and non-monotonic (modal) rules for reasoning with inconsistent and incomplete information. In RuleRS, SPINdle Reasoner receives the formal facts, contexts as predicates from predicate file generated for data stored in the associated relational databases and computes definite or defeasible inferences which are then displayed by the I/O interface.

So, SPINdle is used as a standalone theory prover and embedded into the Java based applications(i.e., Proposed RuleRS) as a defeasible logic rule engine. It works as a workflow compliance checker which determines whether this current task is compliance with its constraints and report to the user/application if there is any problem for this study. SPINdle Java API makes it easier to understand the features which can be integrated in the proposed Rule Based Systems. So this study embeds the SPINdle defeasible logic engine.

# Chapter IV

# Rule Based Reporting Systems

This chapter contains the referenced RuleRS architecture which had been implemented according to the research in [1, 2]. The basic elements including the I/O interface of the existing architecture are described herein short, and which is described briefly in the next section 5.2.3. Also, the formal rules, predicates, and generation of predicates from the database records are explained briefly which methodologies are also used for the proposed RuleRS. And, the other modules of the existing architecture are described in Chapter V with the explanation of newly proposed architecture for Rule Based System.

## 4.1 RuleRS Architecture

In this section,the existing Rule Based Reporting Systems (*RuleRS*) design architecture[1][2] and technical details are briefly explained (refer to Figure: 4.1) which does not support the simultaneous reasoning from multiple sources(i.e., database independence in reasoning) of data. The existing Rule Based architecture is designed to fetch the conclusions from a single source of adverse events using SPINdle rule reasoner and Defeasible Logic. Overall, the existing RuleRS consists of four main system components. In particular, the key system components of RuleRS are: i) I/O Interface, ii) Predicates, iii) Formal Rules, and iv) SPINdle Reasoner. In other subsections of this section, we give a brief outline of the internal components of existing RuleRS and their functions (Refer to Figure: 4.1).

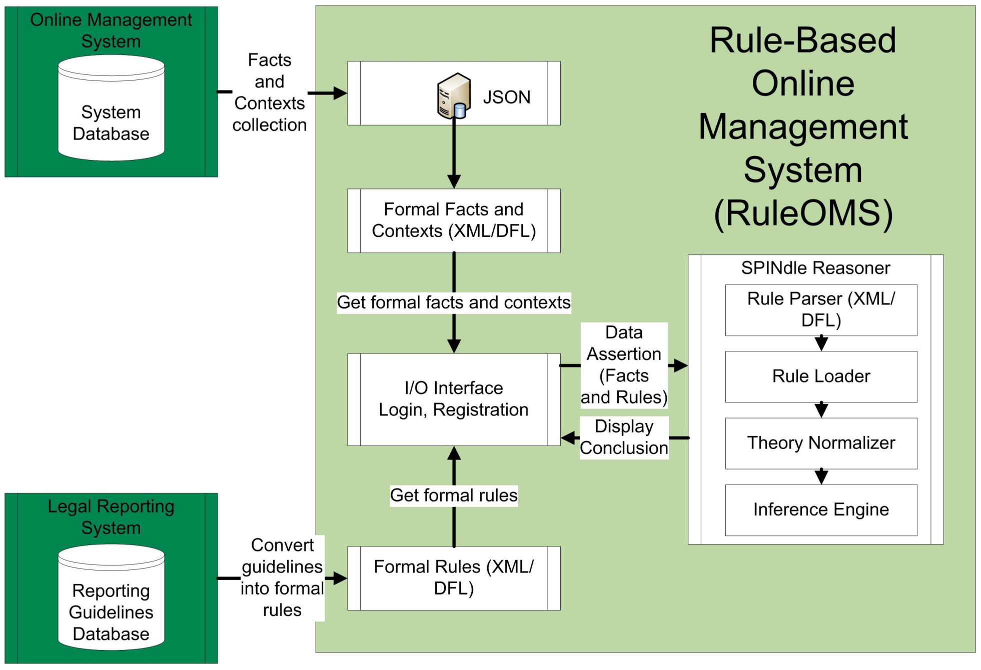

Online Management System
System Database
Facts and Contexts collection
JSON
Rule-Based Online Management System (RuleOMS)
Formal Facts and Contexts (XML/DFL)
Get formal facts and contexts
SPINdle Reasoner
Rule Parser (XML/ DFL)
Rule Loader
Theory Normalizer
Inference Engine
Data Assertion (Facts and Rules)
Display Conclusion
I/O Interface Login, Registration
Get formal rules
Legal Reporting System
Reporting Guidelines Database
Convert guidelines into formal rules
Formal Rules (XML/ DFL)

**Figure 4.1:** Existing Architecture for the Rule Based System [1, 2]

The architecture includes all the above mentioned four components for processing the adverse events to make decisions. The Predicates are generated from single databases using SQL query and the formal rules are generated using the Defeasible Logic (DL). Finally, SPINdle rule reasoner generates conclusions from the input theories and sends to the I/O interface.

### 4.1.1 I/O Interface

The I/O Interface is implemented in JAVA programming language to establish links between RuleRS components and interacting with each other. Predicates contains SQL or JSON file format.The I/O interface is used to compile those predicates and to generate facts and contexts in formal notation using DFL syntax. The SPINdle reasoner takes the generated facts and contexts as a parameter. After that, the I/O interface is used to display the generated remarks or comments for each & every incidents and predicates.

### 4.1.2 Database Records to Predicates

Rule Based Systems encodes legal knowledge to draw inferences from a database. SPINdle rule reasoner requires the predicates to process the data. So,the stored information in databases is queried to provide the facts for the SPINdle reasoner in the form of predicates. Facts are queried from the available databases (i.e., ChildSafeDB). The SPINdle rule reasoner draws a set of assumptions using the provided facts and the defeasible rules encoding the decision trees of the NSW mandatory reporter guidelines and a set of additional rules. The rules are normally provided by the domain experts and give further indicators for analyzing the risks of neglect or abuse.

RuleRS is equipped with SQL/JSON files which contains the definitions of the predicates that are used by the SPINdle component. They are represented using either simple attribute names and values or by SQL queries. For instance, a person is considered a child if his/her age is less than fifteen years according to the provided guidelines. Several steps in the decision trees depend on whether a subject is at risk of abuse is a child or not. In the existing RuleRS, that case is represented by the predicate *beingChild* and contains the following (JSON) definition:

```
{"predicate":"beingChild",
"SQL":"SELECT *
FROM tblchildren
WHERE tblchildren.child_crn = $id
AND tblchildren.child_dob >= ${today - 15y}"}
```

where `$id` is a parameter given as input by the end-user, and `${today - 15y}` is a computation using the system variable `$today`.

The I/O interface takes the user input, examines the definitions, transforms the descriptions into appropriate SQL queries. And then, executes the queries to return facts and context as generated output. For example, for the predicate *beingChild* and `$id=123456789A` ChildSafeDB produces the following JSON snippet, encoding the predicate and the context that it holds.

```
{"beingChild":{
"tblchildren.child_crn":"123456789A",
"tblchildren.child_first_name":"Simone",
"tblchildren.child_middle_name":"Ruby",
"tblchildren.child_last_name":"Shirazi",
"tblchildren.child_dob":"2013-04-02",
"tblchildren.child_gender":"Female"}}
```

The predicate is then used in the successive reasoning phase, while the context is used in case of reasoning phase determines that a report has to be submitted to a relevant authority; the RuleRS is used to populate the required reports.

### 4.1.3 Predicates

The values encoding a regulation and the databases (schema's) used in conjunction with the rules are in general developed independently are likely to have a different vocabulary in general. There is no direct correspondence between the literals used by the rules and the table/attributes of the database schema. Accordingly, we have to establish a mapping among them to enable the integration of rules and instances in the database. The fundamental idea behind predicates is that data stored in the database corresponds to facts in a defeasible theory and these facts can be retrieved from the database using queries. Thus, each and every fact corresponds to a (SQL) query and a predicate is a statement that can be true or false depending on the value of its arguments/variables. A predicate with $n$ arguments is just an $n - ary$ relation.

A predicate in RuleRS corresponds to a database view, i.e.; a named query, where the name is literal to be used by the defeasible rules. The details involves the query to be run to determine if the predicate is true or false for a given set of parameters. In case the output of the query is not empty, the predicate is true and is passed to the defeasible theory as fact.

In the existing RuleRS, predicate comprises of two components: (i) ”predicate name” and (”) ”predicate details”. *Predicate name* represents the action(s), condition(s) or indisputable statement(s), and passed on to the rule engine, SPINdle as defeasible fact (literal and modal literal) [59, 50, 29] or actions that have been performed. For example, the fact “There is a risk for an incident” is represented by “*riskForIncident*” and passed as “$>>$ *riskForIncident*” to SPINdle if it is returned as true from the given relational database. “Predicate details” includes the “incident details” and may be stored as SQL Statement or (after conversion) lightweight data-interchange JSON (JavaScript Object Notation) syntax[1](easy for humans to read and write, easy to generate, and easy to parse for machines) to create a bridge between the data stored in the database and the terms passed as predicates (input case) to the rule engine. The SQL or JSON statements can be created in the initialization of RuleRS with all of the incidents along with all of the predicates for each of the incidents or dynamically add it later.

Incident ID and relevant details of the incidents are also included for each of the predicates and named the predicates with relevant incident information such as “riskForIncident.sql” (for SQL statement) or “riskForIncident.json” (for JSON Statement). The following snippet illustrates the syntax adopted by RuleRS: “riskForIncident” predicate using SQL Statement:

[1]`http://json.org` accessed on 16 November 2018

```
SELECT incidentID, IncidentDetails, IncidentDetails1,
   IncidentDetails2
FROM tblIncident
WHERE incidentID='XXXXXX'
```

In this example, $\boxed{\text{IncidentDetails}}$, $\boxed{\text{IncidentDetails1}}$, $\boxed{\text{IncidentDetails2}}$ are substituted for the placeholders in the “riskForIncident” predicate from relational databases for the $\boxed{\text{incidentID}}$ $\boxed{\text{'XXXXXX}}$.

“riskForIncident” predicate using JSON Statement:

```
{"riskForIncident":
{ "incidentID":"XXXXXX",
"IncidentDetails":"ABC",
"IncidentDetails1":null,
"IncidentDetails2":"XYZ"}}
```

The predicates could be further classified according to the query types as basic (compares two values), quantified (compares a value or values with a collection of values), BETWEEN (compare a value with a range of values), NULL (tests for null values), LIKE (search for strings that have a certain pattern), EXISTS (tests for the existence of certain information), IN (another approach to compare a value with a collection of values [60, 61],the aggregate functions –e.g., AVG, MAX, MIN, SUM, COUNT(*), or COUNT(DISTINCT)– and SQL statement[2]composition to each type of constructs such as WHERE, GROUP BY, HAVING and ORDER BY. Predicates can be further classified according to the query types [3]

In the next step, the records and incidents for which there is a match in the relational database are transformed into predicates to be used by the SPINdle rule engine [11]. And then it is forwarded to SPINdle for further processing using the I/O interface to make the process dynamic.

### 4.1.4 Formal Rules

One of the important features of the RuleRS is that its ability to perform reasoning based on (legal) requirements. As we stated in the other chapters such regulatory requirements are represented as formal rules in Defeasible Logic (Deontic) [47, 34]. It enables their use with the rule engine used by RuleRS (SPINdle, see Section 3.2) the rules are stored using the DFL format [11]. At this stage the rules are created manually and (semi-)automatically by legal knowledge engineers and stored in a knowledge-base.

[2]When “SELECT” statement is used in a predicate, is called a “subquery” [61]

[3]The are various query types as basic (compares two values), quantified (compares a value or values with a collection of values), BETWEEN (compares a value with a range of values), NULL (tests for null values), LIKE (searches for strings that have a certain pattern), EXISTS (tests for the existence of specific information), IN (another approach to compare a value with a collection of values[60, 61]) and different aggregate functions (AVG, MAX, MIN, SUM, COUNT(*), or COUNT(DISTINCT)). and SQL statement[4] composition to each kind of constructs such as WHERE, GROUP BY, HAVING and ORDER BY.

# Chapter V
# Methodology

The proposed methodology for the Rule Based Reporting System (RuleRS) is explained briefly in this chapter. Also, it includes the data collection & processing system for both the databases to ensure the database independence feature integration with the existing RuleRS which is described in the Section:4.1. Moreover, the different essential modules for the proposed RuleRS are described with a brief explanation.

## 5.1 Proposed RuleRS System Architecture

Proposed Rule Based adverse event processing system architecture (Figure: 5.1) includes Domain Data Integrator (DDI), PRTDB, Predicates, Formal facts & contexts, I/O interface, and SPINdle rule reasoner. The concept of DDI for integrating multiple sources of data and PRTDB for dynamic generation and translation of predicates schematic information into database records is introduced with the existing Rule Based Reporting System (RuleRS) [1, 2]. Figure: 5.1 explains the dynamic generation method of SQL

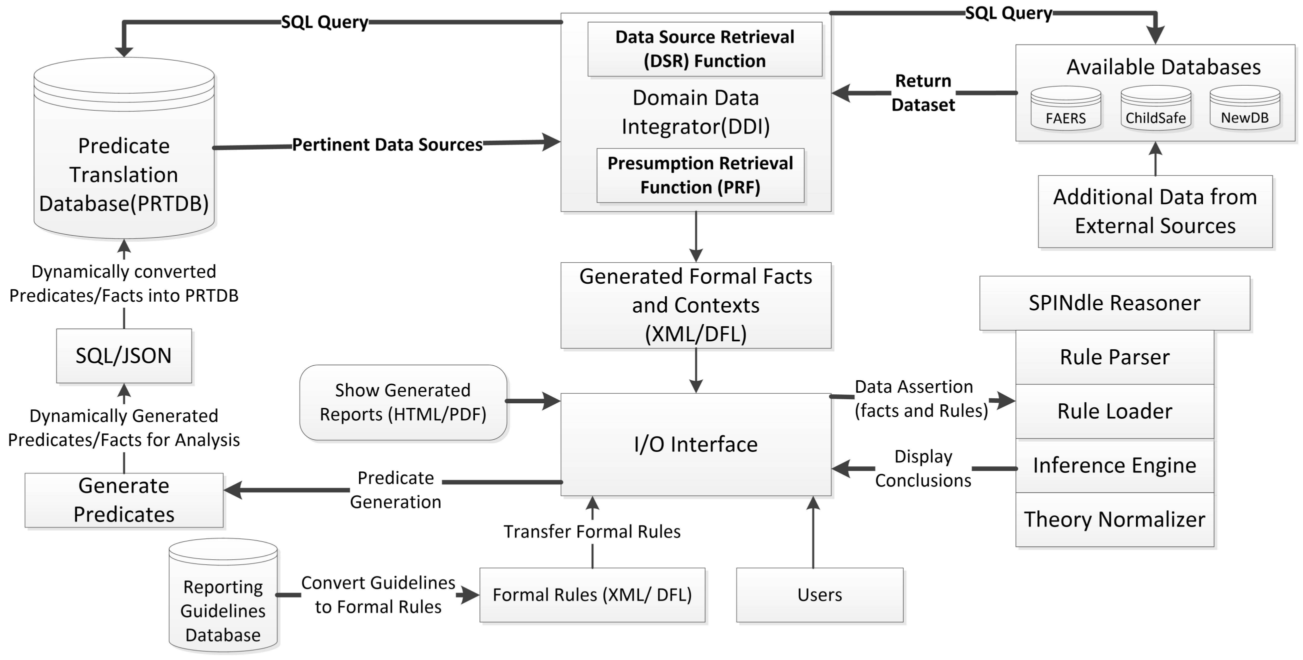

SQL Query
SQL Query
Data Source Retrieval (DSR) Function
Domain Data Integrator(DDI)
Presumption Retrieval Function (PRF)
Return Dataset
Available Databases
FAERS
ChildSafe
NewDB
Additional Data from External Sources
Predicate Translation Database(PRTDB)
Pertinent Data Sources
Dynamically converted Predicates/Facts into PRTDB
SQL/JSON
Dynamically Generated Predicates/Facts for Analysis
Generate Predicates
Generated Formal Facts and Contexts (XML/DFL)
SPINdle Reasoner
Rule Parser
Rule Loader
Inference Engine
Theory Normalizer
Show Generated Reports (HTML/PDF)
I/O Interface
Data Assertion (facts and Rules)
Display Conclusions
Predicate Generation
Transfer Formal Rules
Reporting Guidelines Database
Convert Guidelines to Formal Rules
Formal Rules (XML/ DFL)
Users

**Figure 5.1:** Proposed Architecture for the Rule Based Adverse Event Processing System

predicates and Predicate translation database (PRTDB) which holds the information of predicates that are written using SQL query syntaxes. Domain Data Integrator (DDI) automatically selects the database and generates SQL select query by retrieving schematic information from PRTDB. The SPINdle rule reasoner processes the adverse dataset using the DL generated facts and contexts to generate conclusions from multiple data sources following the reporting guidelines. The generated output is forwarded to the I/O interface to show the result for taking decisions. The proposed method also ensures to adopt additional data sources including the operational data sources for reasoning within a shortest possible time.

## 5.2 Data Collection & Processing

Data is generated from different applications of this technology-laden world. And, the data are stored in different databases which contain different field formats. The difference of database formats can be minimized to draw conclusions by using the proposed RuleRS architecture. We have described in the following two different databases (namely FDA and ChidSafeDB database) for our study.

### 5.2.1 FAERS (FDA) Data

US Food & Drug Administration Adverse Event Reporting Systems (FAERS) is the collection of events for the adverse event reporting system of US Food & Drug Administration (FDA). The events are generated for taking initiatives for the protection of peoples health with safety, efficacy, and security on radiation emitted drugs and equipment [6]. So, the administration stores all the available data on patient demographics (Demographic File) with 260256 counts for total values for the table fields and on an average 92701 counts for missing field values, drug therapies (Drug File) with 902805 counts for total values for the table fields and on an average 548269 counts for missing field values, indications for the prescribed drugs (INDI File) with 910357 counts for total values for the table fields and 327839 counts for missing field values, reaction(s) experienced (Reaction File) with 755410 counts for total values for the table fields and no missing field values, patient outcomes (Outcome File) with on an average 28540 counts for total fields values for the table and on an average 10451 counts for missing field values, the source of the report (Report Source File) with on an average 4854 counts for total values for the table fields and on an average 1048 counts for missing field values, and the drug therapy start and end dates (Therapy File) with 910357 counts for total values for the drugs & 330603 counts for total values for the therapy records table fields and on an average 21054 counts for missing field values. The study validates database independent model which creates reports based on a set of rules & regulations from the FAERS dataset. So, in short, the tables are described in the following. FAERS database contains different tables to store adverse events.

- The DEMO14Q1 table contains patient demographic and administrative information, a single record for each adverse events report.
- The DRUG14Q1 table contains drug/biologic information for as many medications as were reported for the adverse events.
- The REAC14Q1 table contains all "Medical Dictionary for Regulatory Activities" (MedDRA) terms coded for the adverse events.
- The OUTC14Q1 table contains patient outcomes for the adverse events.
- The RPSR14Q1 table contains report sources for adverse events.
- The THER14Q1 table contains drug therapy start dates and end dates for the reported drugs.
- The INDI14Q1 table contains all "Medical Dictionary for Regulatory Activities" (MedDRA) terms coded for the indications for use (diagnoses) for the reported drugs..

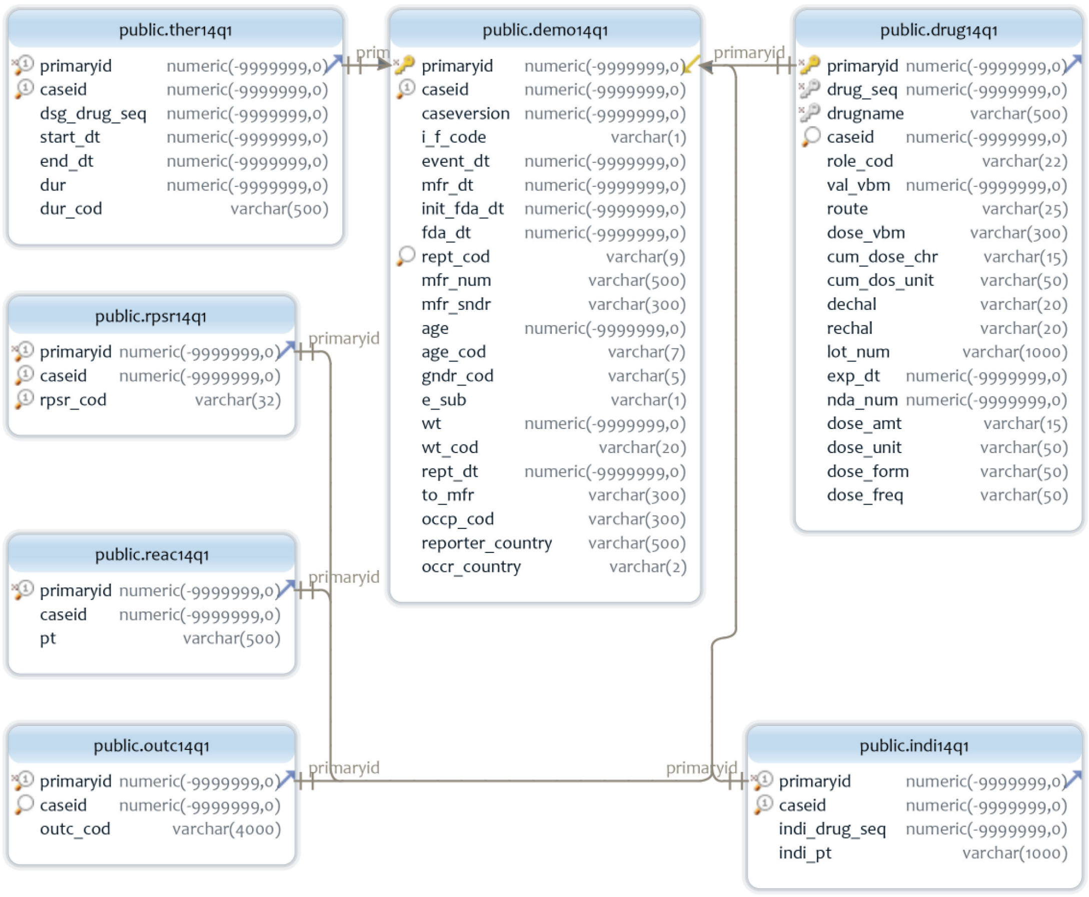

public.ther14q1
primaryid numeric(-9999999,0)
caseid numeric(-9999999,0)
dsg_drug_seq numeric(-9999999,0)
start_dt numeric(-9999999,0)
end_dt numeric(-9999999,0)
dur numeric(-9999999,0)
dur_cod varchar(500)
public.demo14q1
primaryid numeric(-9999999,0)
caseid numeric(-9999999,0)
caseversion numeric(-9999999,0)
i_f_code varchar(1)
event_dt numeric(-9999999,0)
mfr_dt numeric(-9999999,0)
init_fda_dt numeric(-9999999,0)
fda_dt numeric(-9999999,0)
rept_cod varchar(9)
mfr_num varchar(500)
mfr_sndr varchar(300)
age numeric(-9999999,0)
age_cod varchar(7)
gndr_cod varchar(5)
e_sub varchar(1)
wt numeric(-9999999,0)
wt_cod varchar(20)
rept_dt numeric(-9999999,0)
to_mfr varchar(300)
occp_cod varchar(300)
reporter_country varchar(500)
occr_country varchar(2)
public.drug14q1
primaryid numeric(-9999999,0)
drug_seq numeric(-9999999,0)
drugname varchar(500)
caseid numeric(-9999999,0)
role_cod varchar(22)
val_vbm numeric(-9999999,0)
route varchar(25)
dose_vbm varchar(300)
cum_dose_chr varchar(15)
cum_dos_unit varchar(50)
dechal varchar(20)
rechal varchar(20)
lot_num varchar(1000)
exp_dt numeric(-9999999,0)
nda_num numeric(-9999999,0)
dose_amt varchar(15)
dose_unit varchar(50)
dose_form varchar(50)
dose_freq varchar(50)
public.rpsr14q1
primaryid numeric(-9999999,0)
caseid numeric(-9999999,0)
rpsr_cod varchar(32)
public.reac14q1
primaryid numeric(-9999999,0)
caseid numeric(-9999999,0)
pt varchar(500)
public.outc14q1
primaryid numeric(-9999999,0)
caseid numeric(-9999999,0)
outc_cod varchar(4000)
public.indi14q1
primaryid numeric(-9999999,0)
caseid numeric(-9999999,0)
indi_drug_seq numeric(-9999999,0)
indi_pt varchar(1000)
prim
primaryid
primaryid
primaryid
primaryid
primaryid

**Figure 5.2:** Schematic Diagram for the FAERS Database

The above illustrated schematic diagram (Figure: 5.2) describes the field types(such as numeric, varchar etc.) and field lengths (such as varchar(9) for the report code field of demographic table for only storing nine characters) for all the tables of the FAERS database. The diagram (designed using DbSchema [1] software version 8.0.10 build -1 Java 10.0.2 by Oracle Corporation) also explains the primary identification number which is a unique number to search values in different tables and to make relations with other tables of the database.

## 5.2.2 ChildSafe Data

ChildSafe is a collection of generated records based on dataset available from stat-computing. Most of the data come from RITA [2]. These data are divided into yearly chunks after removing the derivable variables. Data is downloaded from 1998 to April 2008. Every table of the database has different types of fields. Data contains the flight arrival and departure information for commercial flights. A database filed 'child_crn' (Child's Child Reference Number ) is created to work as a primary identification (id) number and creating a relationship with other tables. The 'child_crn' identifies that a child's number of a visit to the airport for a year. The proposed database independent Rule Based Reporting System analyzed the children's daily activities to find out child's abuse information along with others. The above illustrated schematic diagram (Figure: 5.3) describes the field types(such as bigint, varchar,text, timestamps etc.) and field lengths (such as varchar(255) for the rule id field of the mandatory report

[1]`https://www.dbschema.com/index.html` accessed on 27 November 2018

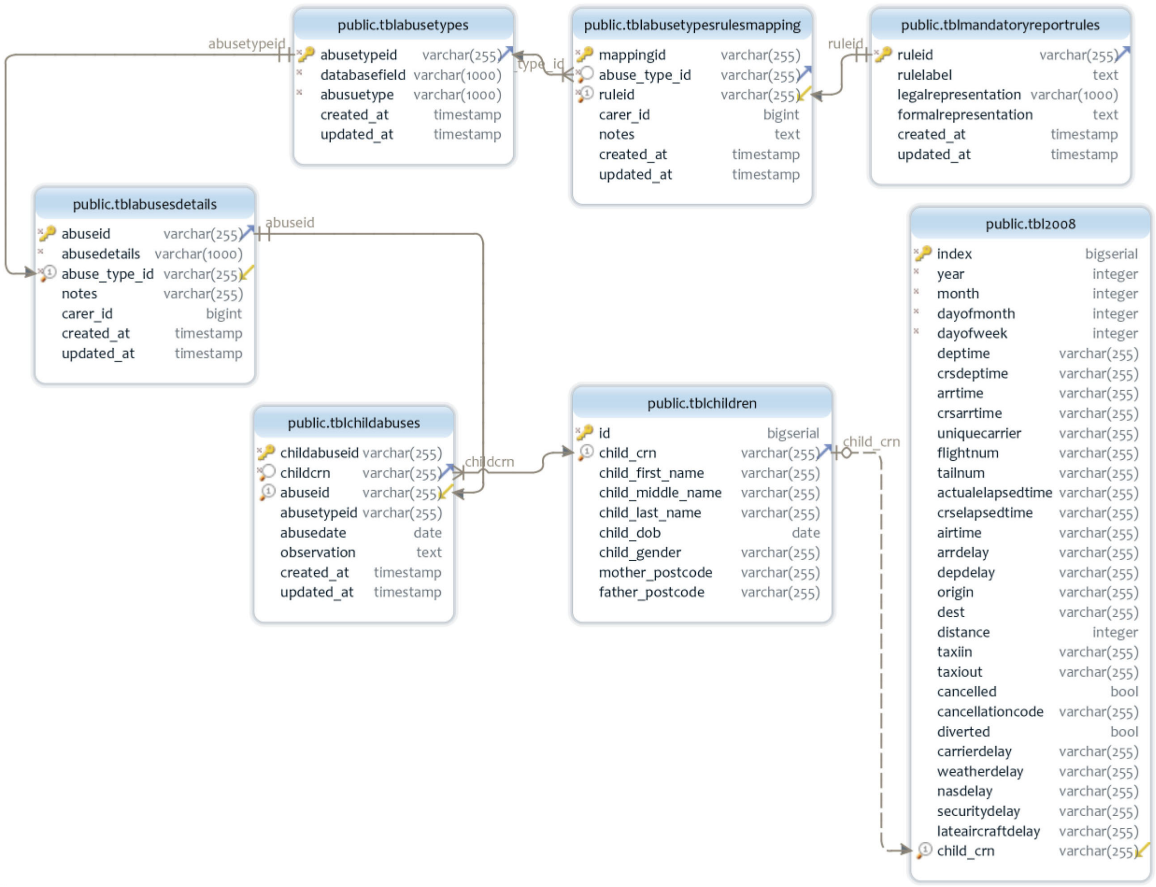


**Figure 5.3:** ChildSafe Database Structure

rules table for only storing 255 characters) for all the tables of the ChildSafe database. The diagram (designed using DbSchema software version 8.0.10 build -1 Java 10.0.2 by Oracle Corporation) also explains the primary identification number as 'child_crn' which is a unique number to search values in different tables and to make relations with other tables of the database.

### 5.2.3 I/O Interface

The I/O interface is used for the visualization of the processed conclusions using the proposed RuleRS architecture. The conclusions are generated from the input multiple data sources. The I/O interface is designed using JAVA which is an object-oriented programming language. Output for the proposed RuleRS can be processed into HTML/ PDF format view which increases the user-friendliness of the proposed Rule Based Reporting System. So, the illustration (figure: 6.11) for the proposed RuleRS architecture introduces I/O system so that it can be used to process the input facts and contexts to show the generated required output from multiple data sources for the end users.

## 5.3 Predicate Translation Database (PRTDB)

The predicate translation database (PRTDB) is designed to introduce the feature of generating conclusions simultaneously from multiple sources of data with the existing RuleRS. For example, we can take a predicate to show how it is converted into records for the fields of different tables of PRTDB. We have selected a sample SQL Query for converting into PRTDB records.

```sql
SELECT primaryid,CASE WHEN age < 20 THEN 'age_below_20_FDA'
ELSE '-age_not_below_20_FDA'
END
FROM demo14q1
```

PREDICATE V.1: report_age_is_below_20_FDA

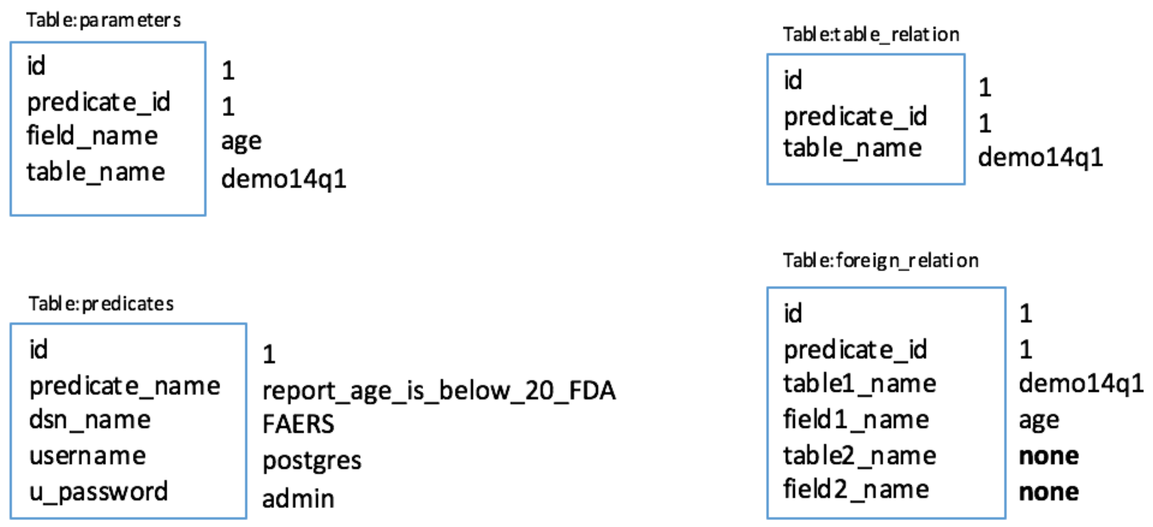


Table:parameters

| | |
|---|---|
| id | 1 |
| predicate_id | 1 |
| field_name | age |
| table_name | demo14q1 |

Table:table_relation

| | |
|---|---|
| id | 1 |
| predicate_id | 1 |
| table_name | demo14q1 |

Table:predicates

| | |
|---|---|
| id | 1 |
| predicate_name | report_age_is_below_20_FDA |
| dsn_name | FAERS |
| username | postgres |
| u_password | admin |

Table:foreign_relation

| | |
|---|---|
| id | 1 |
| predicate_id | 1 |
| table1_name | demo14q1 |
| field1_name | age |
| table2_name | **none** |
| field2_name | **none** |

**Figure 5.4:** Translation of SQL query into Field Records for the Tables of PRTDB

According to the Figure: 5.4, the 'predicates' table stores the predicate name, data source name and username & password for the data source to get the authorizations using a unique identification number for the predicate. The 'parameters' table holds all the column names for the predicate. 'table_relation' keeps records for the required tables for select operations. And, the 'foreign_relation' table stores the join table names and the join column names for the predicate.

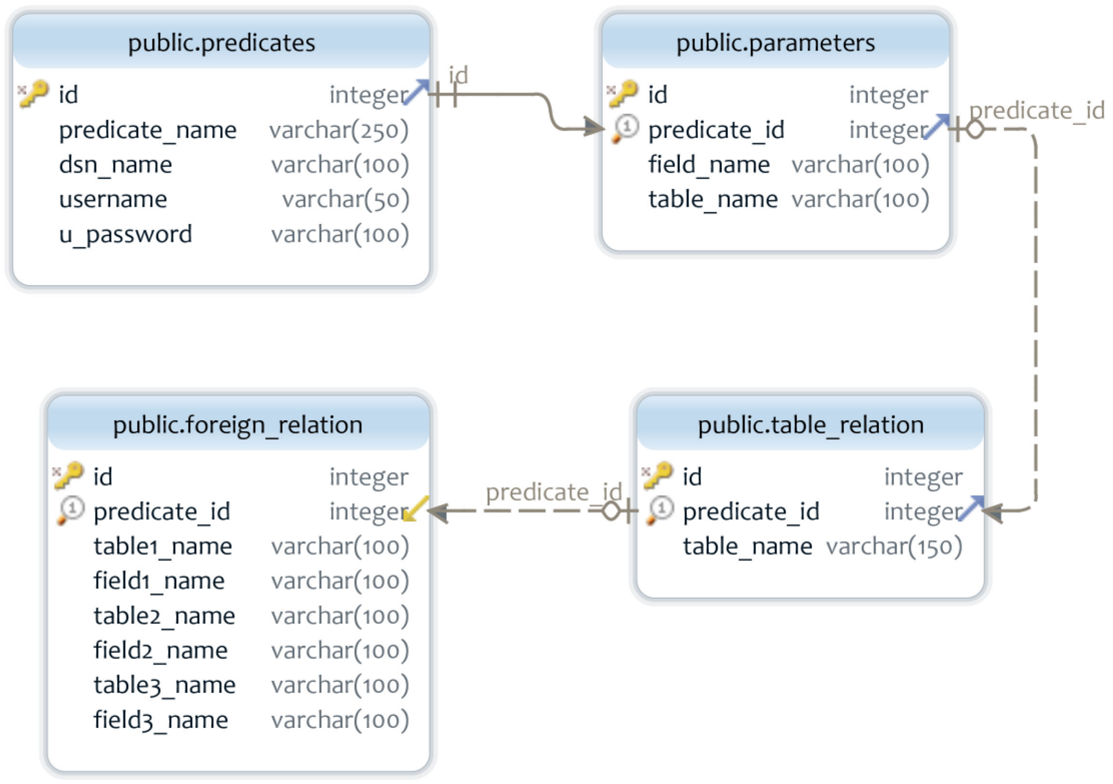


**Figure 5.5:** Schematic Diagram for the PRTDB Database

The above illustrated schematic diagram (Figure: 5.5) describes the field types(such as integer, varchar etc.) and field lengths (such as varchar(50) for the username field of the predicates table for only storing fifty characters) for all the tables of the PRTDB database. The diagram (designed using DbSchema software version 8.0.10 build -1 Java 10.0.2 by Oracle Corporation) also explains the predicate identification number which is a unique number to search values in different tables and to make relations with other tables of the database.

## 5.4 Domain Data Integrator (DDI)

Domain Data Integrator (DDI) generates dynamic Structured Query Language (SQL) query to fetch data from available data sources. DDI integrates the feature of simultaneous access to multiple data sources for generating conclusions with the existing Rule Based Reporting System [1] [2]. DDI includes Data Source Retrieval (DSR) function and Presumption Retrieval Function (PRF). The DSR function generates the column names and joining tables with joining fields. And, the PRF generates a full SQL query with the conditional statement to fetch data from the selected data source.

## 5.5 System Flow Diagram

Generation of reports for conclusions using predefined rules & regulations on data sources of enterprises with Rule Based Reporting Systems can be summarized in several steps as in Figure: 5.6. Database independent Rule Based Reporting System workflow diagram explains a pictorial summary of decomposition for the primary activities into a series of smaller pieces of methods. The first task is to convert the facts in Structured Query Language (SQL) format into records of predicate translation database by inserting the column names, table names and join operators which are defined in the SQL format query predicate. After that, we selected programming methods using Java Object Oriented Programming (OOP) language to design an input form for inputting the unique identification number for the available databases (i.e.FAERS and ChildSafe databases for this experiment). Then, predefined rules & regulations and the DL generated theories are read from the files and databases.

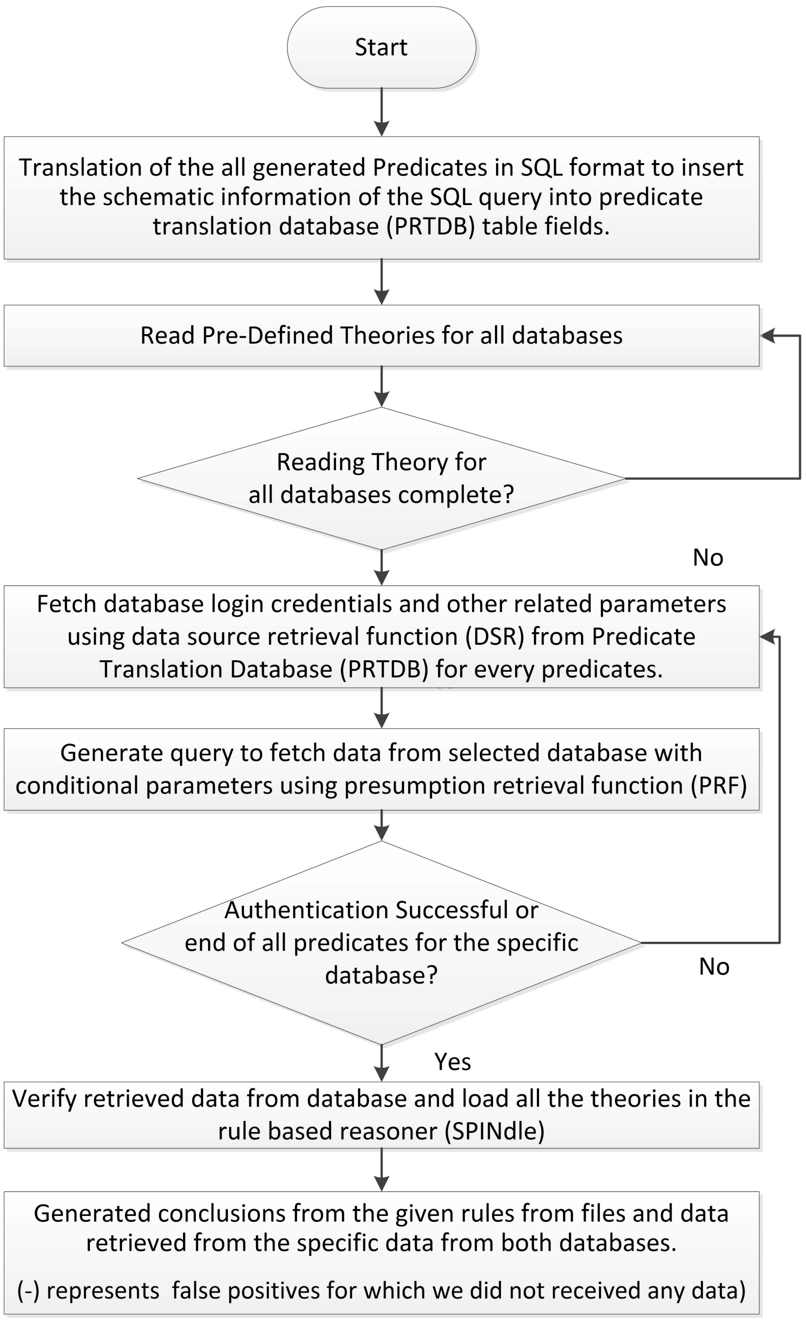

Start
Translation of the all generated Predicates in SQL format to insert the schematic information of the SQL query into predicate translation database (PRTDB) table fields.
Read Pre-Defined Theories for all databases
Reading Theory for all databases complete?
No
Fetch database login credentials and other related parameters using data source retrieval function (DSR) from Predicate Translation Database (PRTDB) for every predicates.
Generate query to fetch data from selected database with conditional parameters using presumption retrieval function (PRF)
Authentication Successful or end of all predicates for the specific database?
No
Yes
Verify retrieved data from database and load all the theories in the rule based reasoner (SPINdle)
Generated conclusions from the given rules from files and data retrieved from the specific data from both databases.
(-) represents false positives for which we did not received any data)

**Figure 5.6:** Workflow Diagram for the Proposed Rule Based Reporting System

After checking the successful completion of the theory loading process, the Data Source Retrieval (DSR) function generates parameters required by Presumption Retrieval Function (PRF) (i.e. Figure: 5.8). And, the PRF generates SQL query to select data from a specific database using the parameters of DSR function. After the successful transaction of SQL select operation for fetching data from the database, the theories and rules are loaded into the SPINdle rule reasoner. Finally, SPINdle generates the required conclusions based on the available rules & regulations. And then, the output for the desired conclusions is sent to the I/O interface.

## 5.6 Algorithm

Before practical implementation of the database independent Rule Based Reporting System, we produced an algorithm for the study rules based system. The algorithm can be easily followed for the logical and programmatic implementation. It also describes the compulsory initialization parameters, database transactions, uses of functions, utilization of loop for repeating steps, logical if-else parameters, and the I/O component.

The algorithm is designed to efficiently handle the input and output for the proposed Rule Based Reporting System. It represents input parameters for predefined rules and regulations, multiple databases and a set of literals generated using DL. And, after that, we have initialized all the related parameters (the value of input & output variables) to null and zero values. Our study generated the predicates based on the user requirements using SQL format. Then, we converted the SQL format predicates to insert the schematic information into PRTDB records. So, when a user requests for information from the stored events, the SQL "select" query is generated using the DSR function and PRF. The DSR function provides the authentication information for the selected database and the PRF method generates a SQL query including all the column names and join operations with conditional parameters. The generated records are then processed using SPINdle Rule Based reasoner for all the available predicates of both the databases. Our proposed Rule Based Reporting System generates output for the required conclusions which are sent to the I/O for display. The whole process combines multiple databases with different formats. The algorithm can accommodate additional databases.

**Algorithm 1:** Proposed Algorithm for the Proposed Rule Based Reporting System

```
1  Inputs:
     List of Predicates & A set of literals generated using Defeasible Logic
     D1: Database 1; D2: Database 2;
     Initialize:
     Initialization of parameters;
     Processing:
     Convert the predicates as SQL file format to predicate translation database(PRTDB) field records;
     insert the schema diagram (column names) of the predicates into PRTDB records;
     for every related predicate from predicate translation database do
2        if predicate name is not null | End of input predicates then
3            select database source and credentials using DSR() function [14]
             if D1 is selected then
4                generate SQL query with conditional parameters using PRF() function [14]
                 generate a resultset from D1 by executing the query
5            else
6                if D2 is selected then
7                    generate SQL query with conditional parameters using PRF() function [14]
                     generate a resultset from D2 by executing the query
8                else
9                    no operation is required for available databases;
10       else
11           Read other input predicates.
12       for every relsultset do
13           if the result is not null then
14               generate facts for every predicate & record predicate name
                 collect the request and response time & compute the average of elapsed time
                 go to the next section;
15           else
16               go back to for loop of this section.
17           go to next resultset;
18       Use the SPINDLE reasoner;
         Forward the theory including facts or contexts;
         for each theory do
19           obtain the resultset from the rule reasoner; obtain the response time for processing;
20       record total response time for processing using SPINDLE rule reasoner;
         go through every resultset from SPINdle
         for every resultset do
21           show output to I/O for generated conclusions.
```

## 5.7 SQL Predicate Generation

We have generated SQL Predicates dynamically for this proposed Rule Based Reporting System. The generation of predicates includes a graphical user interface (GUI) for viewing the databases with their tables. The user can select the relationship from the available columns of the tables and the SQL query is generated based on the selection. The following are some sample predicates which are used to process the data from FAERS and ChildSafe database to output required conclusions. Also, these predicates are generated at the runtime and included in the running Rule Based adverse event processing system to fetch required conclusions from multiple databases.

### 5.7.1 FAERS Database

We generate predicates for FAERS database to fetch information based on user requirements. Some sample scenarios are illustrated in the following.
Check Age is below 20:

```sql
SELECT primaryid,
CASE WHEN age < 20 THEN 'age_below_20_FDA'
ELSE '-age_not_below_20_FDA'
END
FROM demo14q1
```

PREDICATE V.2: Age_is_below_20_FDA

Check whether the patient is Male/Female:

```sql
SELECT primaryid,
CASE WHEN gndr_cod = 'M' THEN 'Gender_is_Male_FDA'
ELSE '-Gender_is_not_Male_FDA'
END
FROM demo14q1
```

PREDICATE V.3: Gender_is_Male_FDA

Check whether the patient has Nasal Congestion disease:

```sql
SELECT primaryid,
CASE WHEN pt = 'Nasal congestion' THEN '
   Patient_has_Nasal_congestion_FDA'
ELSE '-Patient_has_Nasal_congestion_FDA'
END
FROM reac14q1
```

PREDICATE V.4: Patient_has_NasalCongestion_FDA

### 5.7.2 ChildSafe Database

ChildSafe database contains information on the adverse events in different tables. Some sample SQL predicates, which are used to process required conclusions from the data of ChildSafe database illustrated in the following.
Risk factor analysis for a Child 1:

```
SELECT tbl.child_crn, tbl.child_first_name,
tbl.child_middle_name, tbl.child_last_name,
tbl.child_dob, tbl.child_gender
FROM tblchildren as tbl
LEFT JOIN tbl2008 as tbl1 ON tbl1.child_crn=tbl.child_crn
```

Predicate V.5: beingChild

Riskfactor analysis for a Child 2:

```
SELECT tbl.child_crn, tbl.child_first_name,
tbl.child_middle_name, tbl.child_last_name,
tbl.child_dob, tbl.child_gender
FROM tblchildren as tbl
LEFT JOIN tbl2008 as tbl1 ON tbl1.child_crn=tbl.child_crn
WHERE AGE(tblchildren.child_dob)<'15 years 0 mons 0 days'child_crn
```

Predicate V.6: riskForAirportRiskFactor6

## 5.8 Translation of Predicates into Predicate Translation Database (PRTDB)

The column names, table names & join operators of the dynamically generated SQL predicates are input to Predicate Translation Database (PRTDB) through automatic conversion of those predicates into database records. PRTDB consists of four operational tables. The 'predicates' table keeps record of the predicate names with data sources. Parameters table stores field name & table name relationship pair for predicates. The 'table_relation' table keeps information about join operation for all predicates. And, the 'foreign_relation' table stores the primary key& foreign key relationship information for the join operation of the predicates. For example, we take a predicate(i.e. riskForAirportRiskFactor) from the list of generated SQL predicates and convert that into records of PRTDB.

```
SELECT tblchildren.child_crn, tblchildren.child_first_name,
   tblchildren.child_last_name, tbl2008.child_dob FROM tblchildren
LEFT JOIN tbl2008 ON tbl2008.child_crn=tblchildren.child_crn WHERE
   AGE(tblchildren.child_dob)<'5 years 0 mons 0 days'
```

For this SQL query, we run the 'insert' command to add data in the different fields of PRTDB tables. So, 'predicates' table is filled with values (1,riskForAirportRiskFactor,'ChildSafeDB', 'postgres', 'admin'). And, the values (1, 1, child_crn, 'tblchildren'), (2, 1, child_first_name, 'tblchildren'), (3, 1, child_last_name, 'tblchildren') and (4, 1, child_dob, 'tbl2008') are added into 'parameters' table. After that, (1, 1, 'tblchildren') and (2, 1, 'tbl2008') is inserted in table_relation table. Later, we add (1,1,'tblchildren', 'child_crn', 'tbl2008', 'child_crn') values in the foreign_relation table. 'none' value is inserted for predicates without join operations. So, each of the predicates holds a unique identification

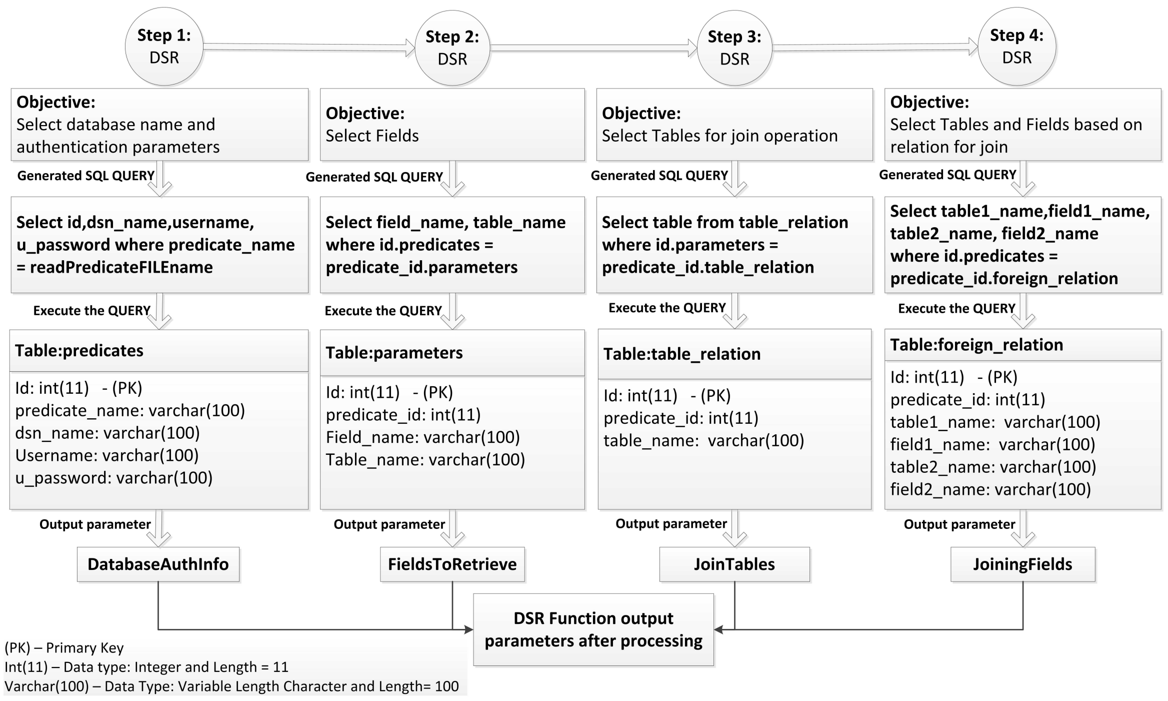

Step 1: DSR
Step 2: DSR
Step 3: DSR
Step 4: DSR
Objective: Select database name and authentication parameters
Objective: Select Fields
Objective: Select Tables for join operation
Objective: Select Tables and Fields based on relation for join
Generated SQL QUERY
Select id,dsn_name,username, u_password where predicate_name = readPredicateFILEname
Select field_name, table_name where id.predicates = predicate_id.parameters
Select table from table_relation where id.parameters = predicate_id.table_relation
Select table1_name,field1_name, table2_name, field2_name where id.predicates = predicate_id.foreign_relation
Execute the QUERY
Table:predicates
Id: int(11) - (PK)
predicate_name: varchar(100)
dsn_name: varchar(100)
Username: varchar(100)
u_password: varchar(100)
Table:parameters
Id: int(11) - (PK)
predicate_id: int(11)
Field_name: varchar(100)
Table_name: varchar(100)
Table:table_relation
Id: int(11) - (PK)
predicate_id: int(11)
table_name: varchar(100)
Table:foreign_relation
Id: int(11) - (PK)
predicate_id: int(11)
table1_name: varchar(100)
field1_name: varchar(100)
table2_name: varchar(100)
field2_name: varchar(100)
Output parameter
DatabaseAuthInfo
FieldsToRetrieve
JoinTables
JoiningFields
DSR Function output parameters after processing
(PK) – Primary Key
Int(11) – Data type: Integer and Length = 11
Varchar(100) – Data Type: Variable Length Character and Length= 100

**Figure 5.7:** Functional Details of DSR function for Predicate Translation Database with Schema Diagram

(id) number against their names. When we search for a predicate, the PRTDB returns with the specific id of that predicate. And, that id number is used to fetch all the related information of the predicate from other tables including database authentication parameters.

The Domain Data Integrator (DDI) is introduced with the proposed RuleRS which integrates all the available databases using DSR function and PRF. DSR function retrieves the database authentication credentials along with the field names, table names for join operation and the conditions required to join the tables from PRTDB. And, the fields for the tables of PRTDB is generated from dynamic conversion of SQL predicates. The DSR function forwards all the retrieved information to the PRF for generating full SQL query to retrieve data from available databases using the conditional parameters.

The functions of Data Source Retrieval (DSR) function with all the database fields is shown in Figure: 5.7. And, the database schema for PRTDB is shown in Figure: 5.7. Domain data integrator (DDI) module sends request query to the predicate table for the database name and authentication parameters for accessing the database information for input predicate. Then, DDI finds the related fields for the predicate from parameters table. After that, table_relation provides the list of tables for the join operation. And, foreign_relation table generates conditions for the join operation between tables. Finally, The DSR function passes all the parameters with value to the Presumption Retrieval Function (PRF).

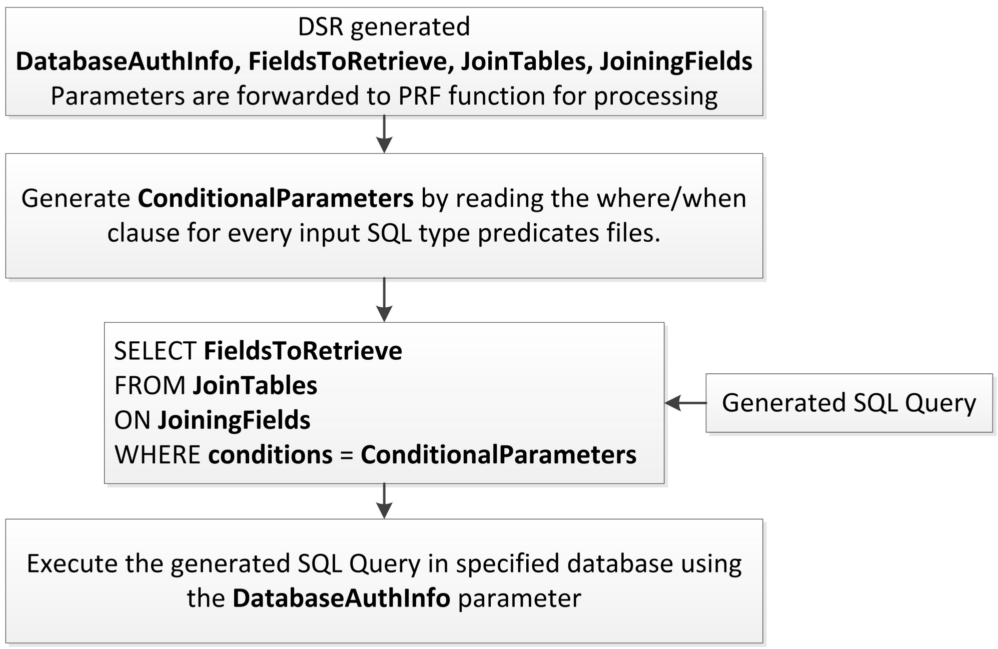

DSR generated
DatabaseAuthInfo, FieldsToRetrieve, JoinTables, JoiningFields
Parameters are forwarded to PRF function for processing
Generate ConditionalParameters by reading the where/when clause for every input SQL type predicates files.
SELECT FieldsToRetrieve
FROM JoinTables
ON JoiningFields
WHERE conditions = ConditionalParameters
Generated SQL Query
Execute the generated SQL Query in specified database using the DatabaseAuthInfo parameter

**Figure 5.8:** Functional Details of PRF Function for Predicate Translation Database

The DDI receives the parameter values passed by the DSR function and guides them to the PRF function as shown in Figure: 5.8. It generates the conditional parameters for the ’where’ clause of the SQL query. And then it generates a SQL query using the conditional parameter, ’FieldsToRetrieve’, ’JoinTables’, and JoiningFields parameters. The SQL query is executed for the selected database using the database authentication credentials.

So, Figure: 5.8, explains the method to fetch predicate as both simple and complex SQL queries from the PRTDB. When we send a query to fetch data according to a predicate name from PRTDB, the database is automatically selected, and tables are co-related using the ‘id’ and ‘predicate_id’ field of the PRTDB tables.

## 5.9 Formal Rules

The proposed Rule Based Reporting System can be used to complete reasoning based on (legal) requirements. We have described regulatory requirements in the introduction chapter (see Chapter I) which are represented as formal rules using Defeasible Logic (Deontic) [47, 34]. Moreover, we have enabled the utilization of those legal requirements with the rule engine used by the proposed RuleRS (SPINdle, See Section: 3.2) and the rules are preserved using the DFL format [11]. In this circumstances, the rules are created (semi-)automatically according to the instructions of the legal knowledge engineers and stored in a knowledge-base. The knowledge-base is then used for generating assumptions for the adverse events from multiple data sources. There are a different type of rules to create formal rules for the proposed Rule Based Reporting System.

### 5.9.1 Defeasible Rules

The main type of rule is a ’defeasible’ rule. These rules are of the form:

$$(condition1),(condition2),... \qquad \Rightarrow \qquad (outcome1),(outcome2),...$$

This reads a ’if (condition 1) and (condition 2) then (outcome 1). If the obligation in (outcome 1) is violated, then is (outcome 2).’ It meets an obligation in force in outcome 2 represents a ’reparation’ or ’compensatory’ action for violating the obligation in outcome 1. Outcome 1, 2 and subsequent outcomes are assumed as a part of an ’obligation chain’. The study recommends that the comma represents ’And’ conditions on the left of the arrow, whereas a comma on the right of the arrow means ’if not’. Conditions and outcomes contain a ’term’ which is important to be used consistently between rules to ensure the correct conclusions are generated (the Vocabulary (/help/rule/vocabulary) functionality assists). These terms are contradicted by placing a negative (-) sign before the term. This means conditions like ’police car’ and ’-police car’ represent ’police car’ and ’NOT police car’, but use the same term. Currently, rules need a terminator of (X) on the end of the last outcome. For an example, consider clause 1 ’Compensatory rule and exception’, rule 1.r1:

vehicle X, vehicle Y, vehicle Y is turning right, X on the right of Y => [OAPP]Y gives way to X, [OP]Y pays fine

The above illustration means if there is a vehicle X, and a vehicle Y, and vehicle Y is turning right, and vehicle X is on the right of Y, then there is an obligation for vehicle Y to give way to vehicle X. If this obligation is violated, then there is an obligation for vehicle Y to pay a fine. The outcomes of a defeasible rule are only ’defeasibly’ concluded, meaning that they can be removed or ’defeated’ by rules that have a higher priority.

### 5.9.2 ChildSafe Database

The database records require appropriate rules to process the required outcome from the stored adverse events. For example, we can explain the rules for processing adverse events from the ChildSafe database (Refer to [29, 55, 56] for other types of obligations). The following rules can be represented as the part of the decision tree which examines that if the child age is between 0 & 5 and the child has coercion behavior then he/she must be reported to being abusive in sexual behavior.

$$r_1: beingChildAged0To5, involveCoercionInBehavior$$
$$\Rightarrow [\mathrm{OANP}]actionBeingAbusiveSexualBehavior.$$

The following is another representation of rules for processing data from the ChildSafe database to generate the report immediately.

$$\begin{aligned} r_2\colon\ & \mathit{PersistentSexualBehaviourVsOther}, \\ & \qquad \mathit{ParentRespondedAppropriately}, \neg[\mathrm{OANP}]\mathit{reportCS}, \\ & \qquad\qquad \neg[\mathrm{OANP}]\mathit{reportCSImmediately} \Rightarrow [\mathrm{OANP}]\mathit{consultWithCWU}. \end{aligned}$$

The rule above describes the decision when the parents or carers are aware of the situation and responded appropriately.

### 5.9.3 FAERS Database

For the FAERS database, formal rules are generated to report electronically to *FDA* as ICSRs include "Patient age" while report to FDA. The rule can be represented as *While an obligation (persistent, non-pre-emptive, achievement) report on ICSRs to FDA, there is an obligation (persistent, non-pre-emptive, achievement) to include "Patient age".* Its formal representation in SPINdle is given in rule $r_3$:

$$r_3\colon [\mathrm{OAPN}]\mathit{report\ on\ ICSRs\ to\ FDA} \Rightarrow [\mathrm{OAPN}]\mathit{report\ Patient\ age\ to\ FDA}$$

Similarly, ICSRs include "Suspect medical product name" while report to FDA. We can represent the rule as *While an obligation (persistent, non-pre-emptive, achievement) to report on ICSRs to FDA, there is an obligation (persistent, non-pre-emptive, achievement) to include "Suspect medical product name".* Its formal representation in SPINdle is as $r_4$:

$$\begin{aligned} r_4\colon\ & [\mathrm{OAPN}]\mathit{report\ on\ ICSRs\ to\ FDA} \\ & \Rightarrow [\mathrm{OAPN}]\mathit{report\ Suspect\ medical\ product\ name\ to\ FDA} \end{aligned}$$

The rules $r_3$ and $r_4$ describe the decision when voluntary (aka direct) and mandatory report to FDA by companies are compliant or non-compliant.

## 5.10 Use case

This section describes the sample use case using two sample databases consisting synthetic data. In this use case, we have used the "MyCSIRO" and "KUETDB" databases which are completely different database. Our primary target is to fetch data from completely different data sources using our proposed Rule Based System.

### 5.10.1 MyCSIRO Database

The "MyCSIRO" database stores the publication titles of all the staffs of CSIRO (a organization). The first table("tblCSIROStaff") contains the identification number and the staff name for the organization. The type for the identification number column is integer. This field("ident") stores all the identification number of the staffs which contains only digits. And, another column("staffname") type is variable

| **Database Name: MyCSIRO** | |
|---|---|
| **Table Name: tblCSIROStaff** | |
| | |
| **Column** | **Column Type** |
| ident | integer |
| staffname | varchar(100) |
| | |
| **Table Name: tblCSIROPublication** | |
| **Column** | **Column Type** |
| ident | integer |
| csiropubtitle | varchar(200) |

| **Database Name: MyCSIRO** | |
|---|---|
| **Table Name: tblCSIROStaff** | |
| **VALUES** | |
| **ident** | **name** |
| abc123 | Guido Governatori |
| def456 | Badiul Islam |
| ghi789 | Nick van Beest |
| | |
| **Table Name: tblCSIROPublication** | |
| **ident** | **csiropubtitle** |
| abc123 | Defeasible Logic |
| def456 | RuleRS |

length character with length 100 which is used to store the names for the CSIRO staffs. The database contains another table (“tblCSIROPublication”) to store the titles for the publications of CSIRO staffs. The table contains an unique identification number for storing the titles according to the publications of CSIRO staffs. The “ident” column relates to the first table to fetch the name of the staff according to the identification number of the staff. Another field(“csiropubtitle”) is for storing the publication titles for the CSIRO staffs.

### 5.10.2 KUETDB Database

The “KUETDB” keeps records for the employees who work at KUET and who have published titles in different research sectors. There is two tables which stores all the related publication titles of the employees of KUET. The first table (“tblKUETstaff”) stores records for the names of the employees(“employeename”) according to the identification number(“kuetid”) provided by the KUET authority. The value of employee name is limited to 100 variable characters. And, another table (tblKUETPublication) keeps the records for the titles of publications(“kuetpubtitle”) according to the identification number(“kuetid”) of the employees. The identification number of second table is matched with the first table to find out the names for the employees whoc have publications.

| **Database Name: KUETDB** | |
|---|---|
| **Table Name: tblKUETstaff** | |
| | |
| **Column** | **Column Type** |
| kuetid | integer |
| employeename | varchar(100) |
| | |
| **Table Name: tblKUETPublication** | |
| **Column** | **Column Type** |
| kuetid | integer |
| kuetpubtitle | varchar(200) |

| **Database Name: KUETDB** | |
|---|---|
| **Table Name: tblKUETstaff** | |
| **VALUES** | |
| **kuetid** | **name** |
| 0307008 | MMA Hashem |
| 0607008 | Prodip Kumer Das |
| 0607009 | Mohammad Azizul Karim |
| | |
| **Table Name: tblKUETPublication** | |
| **kuetid** | **kuetpubtitle** |
| 0307008 | Evolutionary Algorithm |
| 0607008 | RuleRS |

### 5.10.2.1 Case Analysis

The two databases ("MyCSIRO" and "KUETDB") contains separate database structure with separate field names and field types for the tables of databases. Both of the databases contain publication tiles for the registered employees with completely different identification number for the employees. We have also inserted sample data in the databases to illustrate the working procedure for the simultaneous selection of data for generating conclusions. For instance, we show here how to find the paper titles which resides in both of the above-mentioned databases. The total procedure includes different steps to perform the required action and the SQL query for the required action induces PRTDB record for the "MyCSIRO" and "KUETDB" database which is illustrated in the following:

```
SELECT * FROM MyCSIRO.tblCSIROPublication, KUETDB.
   tblKUETPublication WHERE MyCSIRO.tblCSIROPublication= KUETDB.
   tblKUETPublication
```

PREDICATE V.7: MyCSIRO and KUETDB Database

This type of situation is identified and handled when generating dynamic predicates for different databases. And, we convert that into Predicate Translation Database (PRTDB) with the Predicate names. The generated query will work on two steps by executing the generated query in the following:

```sql
SELECT tblCSIROPublication.csiropubtitle FROM MyCSIRO.
   tblCSIROPublication
```

PREDICATE V.8: Query for MyCSIRO Database

For the query in first step, the predicate will use the ”tblCSIROPublication” table for data extraction from ”MyCSIRO” database. The query selects all the publication titles from the ”MyCSIRO” database. And, then the result is passed to the next following query for further processing:

```sql
SELECT tblKUETPublication.kuetpubtitle FROM KUETDB.
   tblKUETPublication
WHERE tblKUETPublication.kuetpubtitle = ('the selected publication
   titles of the first query')
```

PREDICATE V.9: Query for both MyCSIRO and KUETDB Database

The above-mentioned query displays all the publication titles which matches both the ”MyCSIRO” and ”KUETDB” database. If no data is retrieved for the generated query, then the response will be generated with a ”NULL” value.

# Chapter VI
# Experimental Setup and Results

The proposed methodology is implemented using the required software and computing devices which are described in Chapter: V. The following sections illustrate the experimental setup & results for the proposed Rule Based System after generating assumptions from FAERS and ChildSafe database. And, then the study represents the experimental results for the simultaneous reasoning for the adverse events from multiple data sources (both FAERS and ChildSafe database) which introduces database independence for generating conclusions.

## 6.1 Prerequisites

An Integrated Development Environment (IDE) is created to begin the implementation of the proposed Rule Based architecture. In the IDE, we have developed the source code using JAVA programming language and run the proposed Rule Based System to view the generated conclusions from the FAERS & ChildSafe databases. The implementation steps consist the following software and programming language prerequisites including a source code compiler, a web service provider software and a web browser. And, finally, the experimental setup process combines all the prerequisites to run the proposed methodology successfully.

### 6.1.1 Eclipse Software

Eclipse software (ver. neon.3) works as an Integrated Development Environment (IDE) which is used in writing our source code, compiling the source code and executing the source code to view the outcome of the developed Rule Based System. And, it is the most widely used Java IDE. The development environments can be easily customized using the base workspace and an extensible plug-in system. The Eclipse Foundation has written the source code for Eclipse software using Java and defined its primary use is for developing Java applications. But, the software is also available to develop applications in other programming languages via plug-ins, including Ada, ABAP, C, C#, JavaScript, Perl, PHP, Python,Ruby (including Ruby on Rails framework), Rust, Scala, and many others.

Documents with LaTeX (via a TeXlipse plug-in) and packages for the software Mathematica is also developed using Eclipse software. The generated development environments include the Eclipse Java Development Tools (JDT) for Java and Scala, Eclipse CDT for C/C++, and Eclipse PDT for PHP, among others. Eclipse Software Development Kit (SDK) is a free and open source software which is released under the terms of the Eclipse Public License [1].

Moreover, the Eclipse Foundation provides a global community of individuals and organizations with a mature, scalable, and business-friendly environment for open source software collaboration and innovation [2].

[1]https://en.wikipedia.org/wiki/Eclipse_(software) accessed on 15 November 2018
[2]https://www.eclipse.org/org/ accessed on 15 November 2018

This is why we have downloaded Eclipse IDE for Java EE Developers as the package of Eclipse (ver. Neon 3) and installed on a MAC OS based computer to develop and run the source code for the proposed Rule Based Systems to generate conclusions from multiple sources.

### 6.1.2 Glassfish Software

GlassFish is an open-source application server project which was started by Sun Microsystems for the Java EE platform. Oracle Corporation is a sponsor for the Glassfish software. Oracle GlassFish Server has different supported versions available for download. GlassFish is a free software and dual-licensed under two free software licenses: the Common Development and Distribution License (CDDL) and the GNU General Public License (GPL) with the classpath exception. GlassFish is the reference implementation of Java EE and it also supports Enterprise JavaBeans, JPA, JavaServer Faces, JMS, RMI, JavaServer Pages, servlets, etc. This facilitates the developers to create enterprise level applications that are portable and scalable, and that integrate with legacy technologies. Optional components can also be installed for additional services. Glassfish is built on a modular kernel. The kernel is powered by OSGi, Glassfish which runs straight on top of the Apache Felix implementation. The Glassfish also runs with Equinox OSGi or Knopflerfish OSGi runtime. Such services are discovered and injected at runtime for the proposed study on Rule Based Reporting Systems. The source code for the GlassFish was released by Sun and Oracle Corporation. Glassfish uses a derivative of Apache Tomcat as the servlet container for serving Web content using an additional component called Grizzly which uses Java New I/O (NIO) for scalability and speed [3].

Glassfish (version 4.1)[4] is downloaded from the official website of the sun micro systems [5] and installed in the MAC OS based computer for working as the web service provider to run and test the developed source code. The Glassfish software is integrated and configured to use with the Eclipse software so that we can run the source code and view the output in a web browser (i.e., http://localhost:8080/Rulers-FAERS).

### 6.1.3 PostgreSQL Software

PostgreSQL is often called Postgres. This software is a object-relational database management system (ORDBMS) with an emphasis on extensibility and standards compliance. This software is able to handle workloads ranging from small computing device based applications to large Internet-facing applications (or for data warehousing) with huge concurrent users. MAC OS based Server uses PostgreSQL as the default database (i.e., The study uses MAC OS based Server). It is platform independent and also available for Microsoft Windows and Linux (supplied in most distributions) [6].

PostgreSQL has many features including ACID-compliant and transactional. PostgreSQL has dynamically updatable views and materialized views, triggers, foreign keys. It supports functions and stored procedures, and other expandability.

[3]`https://en.wikipedia.org/wiki/GlassFish` accessed on 15 November 2018

[4]`https://download.oracle.com/glassfish/4.1` accessed on 15 November 2018

[5]`https://www.oracle.com/technetwork/middleware/glassfish/overview/index.html` accessed on 15 November 2018

[6]`https://en.wikipedia.org/wiki/PostgreSQL` accessed on 15 November 2018

PostgreSQL software is developed and customized by the PostgreSQL Global Development Group who are continuously working on the feature development. Also, Many companies and individual contributors are in a group for the development and maintenance of the PostgreSQl software. It is completely free and open-source which is released under the terms of the PostgreSQL License that is a permissive software license.

For this Rule Based study, we have downloaded PostgreSQL(version 11.0) [7]. All the databases (FAERS, ChildSafe and Predicate_translation) are loaded in the admin panel of the PostgreSQL. Also, it show a statistics for the records of FAERS database including the total size on disk (299 MB). The variations in the features of the PostgreSQL, helped us a lot to organize and retrieve data for generating conclusions using the proposed database independent analysis of the stored adverse events from multiple databases.

### 6.1.4 JAVA Programming Language

The study requires the knowledge in JAVA programming language also to develop the source code for the proposed Rule Based Systems. JAVA is a general purpose computer programming language that is concurrent, class-based, object-oriented, and it has few implementation dependencies. It is intended to let JAVA application developers providing the facility to compile JAVA code that can run on all platforms and support JAVA without the need for recompilation. In this study, the developed source code is a JAVA application which is typically compiled to byte code that can run on any Java Virtual Machine (JVM) regardless of computer architecture(no matter what the OS is). In 2016, JAVA was one of the most popular programming languages in use. The programming language is very popular for developing the client-server web applications. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them [8].

Primarily, Sun launched the implementation of Java compilers, virtual machines, and class libraries with the original and reference were under proprietary licenses. But, after that GNU General Public License was relicensed for most of the Java technologies according to the compliance with the specifications of the Java Community Process.

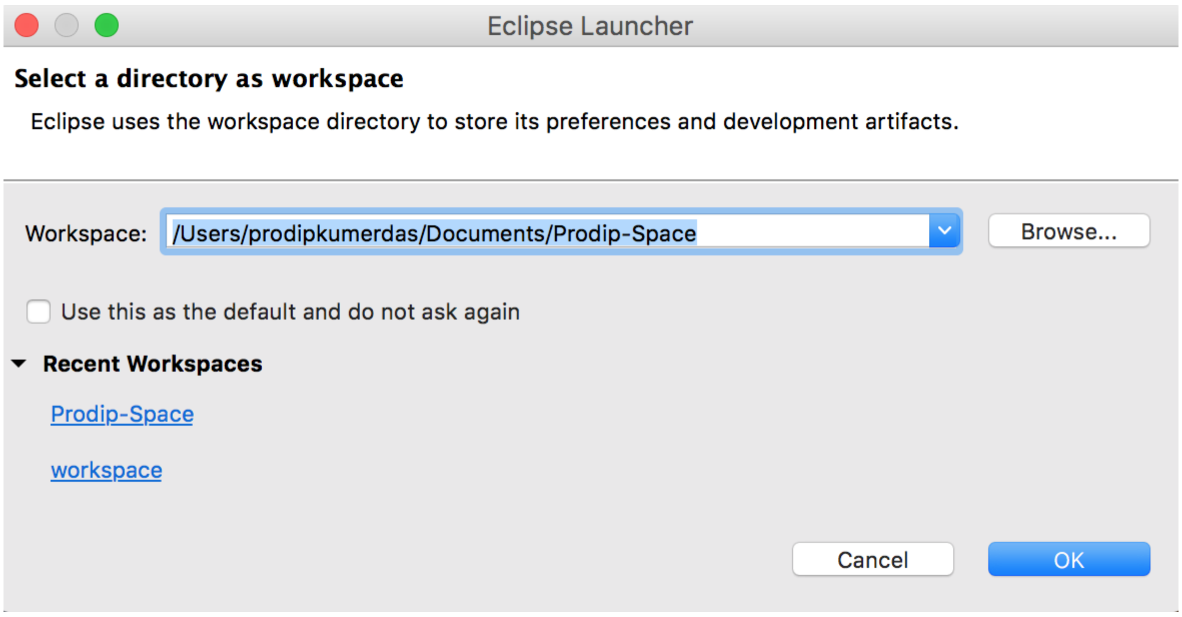
The latest version of Java is version 11 which is released recently. The release follows Java 10 after only a short interval in line with the new release schedule. But, in our study we have used Java Server pages (JSP) to design the graphical user interfaces for the proposed Rule Based Systems. The designed system includes a menu-bar to link the functions in different JSP files in the working directory.

## 6.2 Experimental Setup

The above-explained prerequisites are combined together to build a development environment for the proposed methodology. So, we explain about setting up the development environment, defining the knowledge base, database, inference engine, formal rules, and a user interface. The proposed rule-based system for adverse event processing from multiple data sources is implemented on a computer having a MAC operating system (OS). And, the proposed system is evaluated using two relational databases

[7]`https://wiki.postgresql.org/wiki/PostgreSQL_11_Open_Items` accessed on 15 November 2018

[8]`https://en.wikipedia.org/wiki/Java_(programming_language)` accessed on 15 November 2018


Eclipse Launcher

Select a directory as workspace

Eclipse uses the workspace directory to store its preferences and development artifacts.

Workspace: /Users/prodipkumerdas/Documents/Prodip-Space

Use this as the default and do not ask again

Recent Workspaces

Prodip-Space

workspace

**Figure 6.1:** Proposed Rule Based Adverse Event Processing System Workspace in Eclipse Software

(FAERS and ChildSafe database). The Figure: 6.1 shows the workspace for developing the source code for the proposed Rule Based System. In this workspace all the class files and functions files work together to build a Rule Based System. The configuration of the computer is as follows:
Model Name: MacBook Pro
Model Identifier: MacBookPro 12. 1
Processor Name: Intel Core i5
Processor Speed: 2.7 GHz
Number of Processors: 1
Total Number of Cores: 2
L2 Cache (per core): 256 KB
L3 Cache: 3 MB
Physical Memory: 8 GB

The DL is used to create rules to process data. And, we have developed programming method to convert the SQL query format predicates to records of PRTDB. Our proposed system is developed using Java server pages (JSP) which is a complementary technology to JAVA servlet. Eclipse software is installed as an Integrated Development Environment (IDE) and Java Development Kit (JDK) is used to execute the developed source code for our proposed rule-based system. We also installed PostgreSQL (version 11.0) software to act as a Relational Database Management System (RDBMS) to run queries in the PRTDB, FAERS and ChildSafe database. Glassfish software provides the web services to host the web pages in our local web server. After executing the system, we can view the Graphical User Interface (GUI) to enter the identification number for available databases to process conclusions from the stored adverse events. When we press "Generate Conclusions" button in the GUI, the SPINdle rule-reasoner loads all the generated theories for the available database records through DSR and PRF functions. The inference process of a defeasible theory is divided into two phases. First one is "theory transformations phase" and the second one is "conclusions generations phase". Various transformation algorithms are

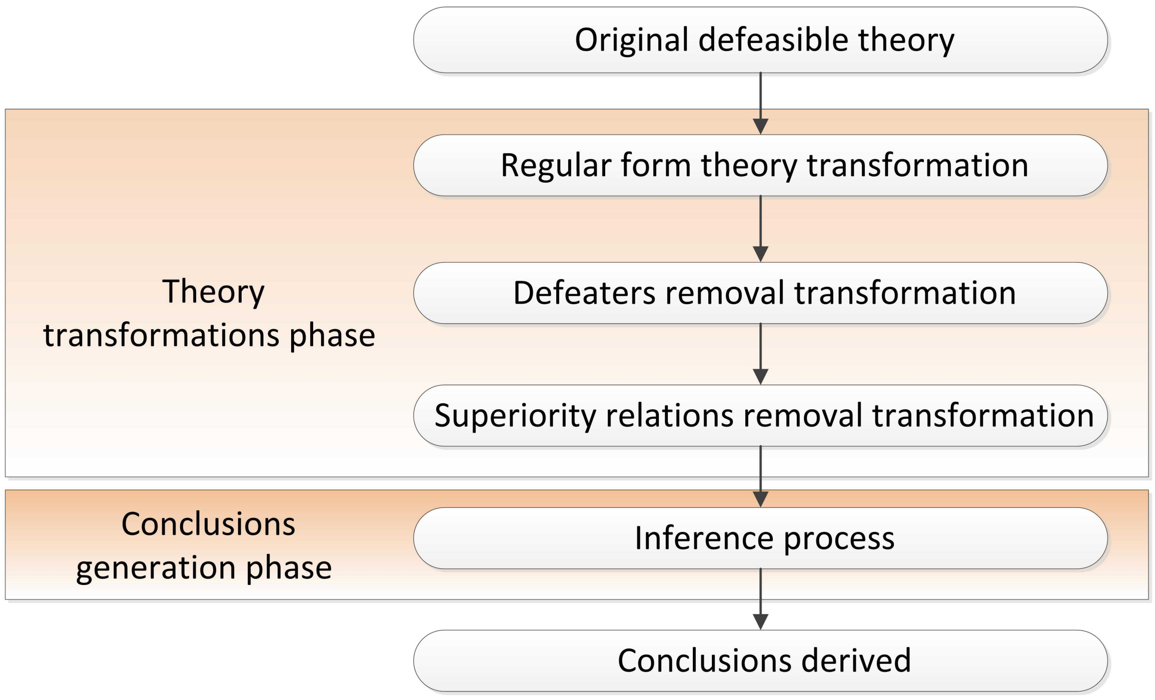

Original defeasible theory
Theory transformations phase
Regular form theory transformation
Defeaters removal transformation
Superiority relations removal transformation
Conclusions generation phase
Inference process
Conclusions derived

**Figure 6.2:** SPINdle - Defeasible Theory Inference Process [3]

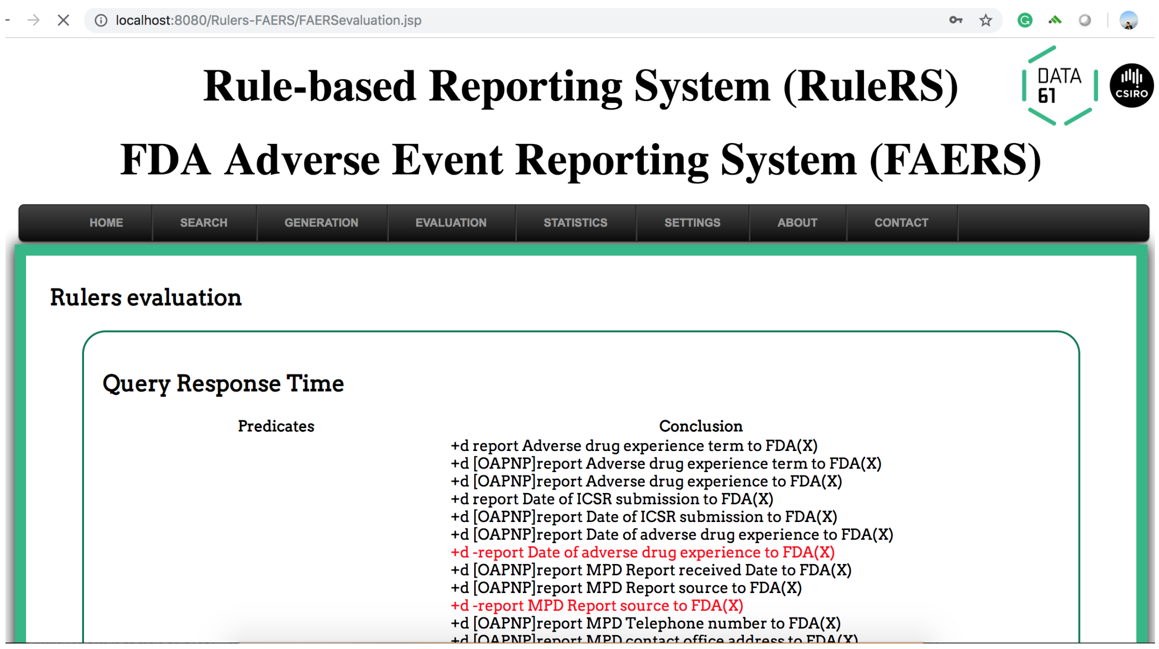
implemented by the theory normalizers that transform a defeasible theory into a form that is processed by the associated inference engines. And, different reasoning approaches[3] helps the inference engines to implement the defeasible reasoning mechanisms. Then, the constraints are identified in the theory by SPINdle that cannot be honored by the inferencing engines by checking the elements in the theory (i.e., assuming that the theory contains any facts, defeaters, etc.) as shown in the Figure: 6.2. Finally, SPINdle inferences on the theory and computes the conclusions and send it back to the users through the I/O interface.

The SPINdle classpath which acts as the rule engine to generate conclusions from the given data sources. Also, it has been proved that the SPINdle is used to take decisions on a large dataset within a short response time. All the modules of SPINdle with functions are explained in section: 3.2.

## 6.3 Experimental Results

Eclipse software is used to develop the source code for the proposed methodology of Rule Based System. After compiling the source code using eclipse, Graphical User Interface (GUI) is generated for the programmatic implementation of the proposed Rule Based study. The database independent analysis of adverse events using the proposed Rule Based System is briefly explained with the generated output. We have experimented our developed system with both of the databases (i.e., FAERS & ChildSafeDB) separately and simultaneously.

Figure: 6.3 shows the GUI for the adverse events analysis system only from the FAERS database. It represents the analyzed conclusions from the stored adverse events from the FAERS database. Figure: 6.4 illustrates the conclusions (+d means the conclusion is defeasibly provable and others are explained in Section: 3.2) and the rules which are in action for the generated output. Finally, the study represents the time consumption with predicate response time, response time per predicate, SPINdle response time and total response time in Figure: 6.5. The number of iterations is represented in the x-axis of the illustration and response time in milliseconds (msec) is represented in the y-axis of the illustration. The time consumption for the FAERS database depends on the Database Structure of FAERS, Number of predicates to be processed during the execution, Nature of data(whether indexed or not indexed structured/unstructured), the method following which the data is being stored continuously, SPINdle theory parsing using theory analyzer and SPINdle processing time.



+d report Adverse drug experience term to FDA(X)
+d [OAPNP]report Adverse drug experience term to FDA(X)
+d [OAPNP]report Adverse drug experience to FDA(X)
+d report Date of ICSR submission to FDA(X)
+d [OAPNP]report Date of ICSR submission to FDA(X)
+d [OAPNP]report Date of adverse drug experience to FDA(X)
+d -report Date of adverse drug experience to FDA(X)
+d [OAPNP]report MPD Report received Date to FDA(X)
+d [OAPNP]report MPD Report source to FDA(X)
+d -report MPD Report source to FDA(X)
+d [OAPNP]report MPD Telephone number to FDA(X)
+d [OAPNP]report MPD contact office address to FDA(X)

**Figure 6.3:** Graphical User Interface (GUI) for only FAERS Database

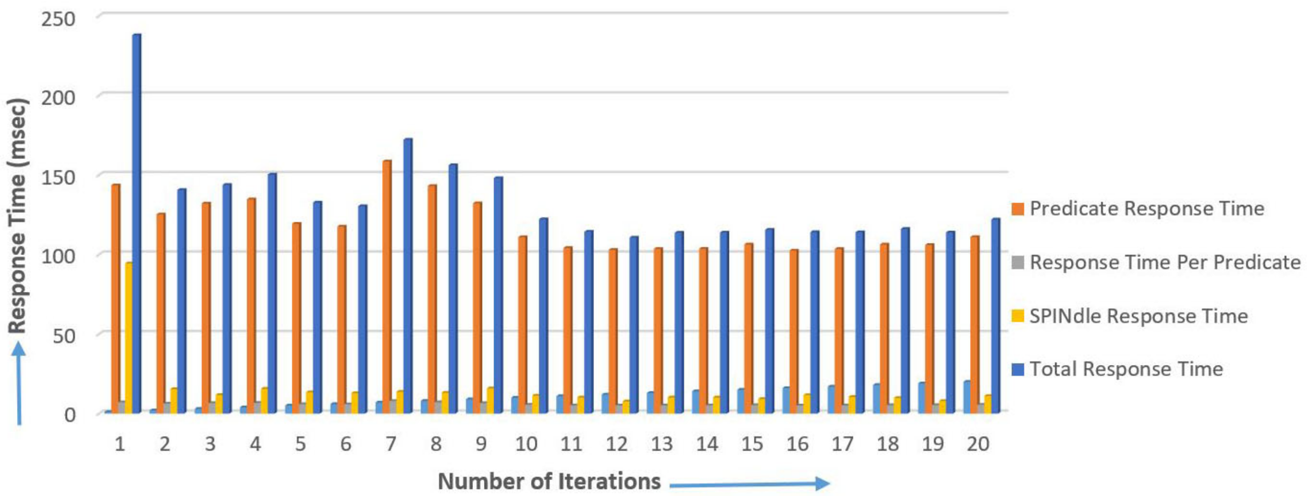
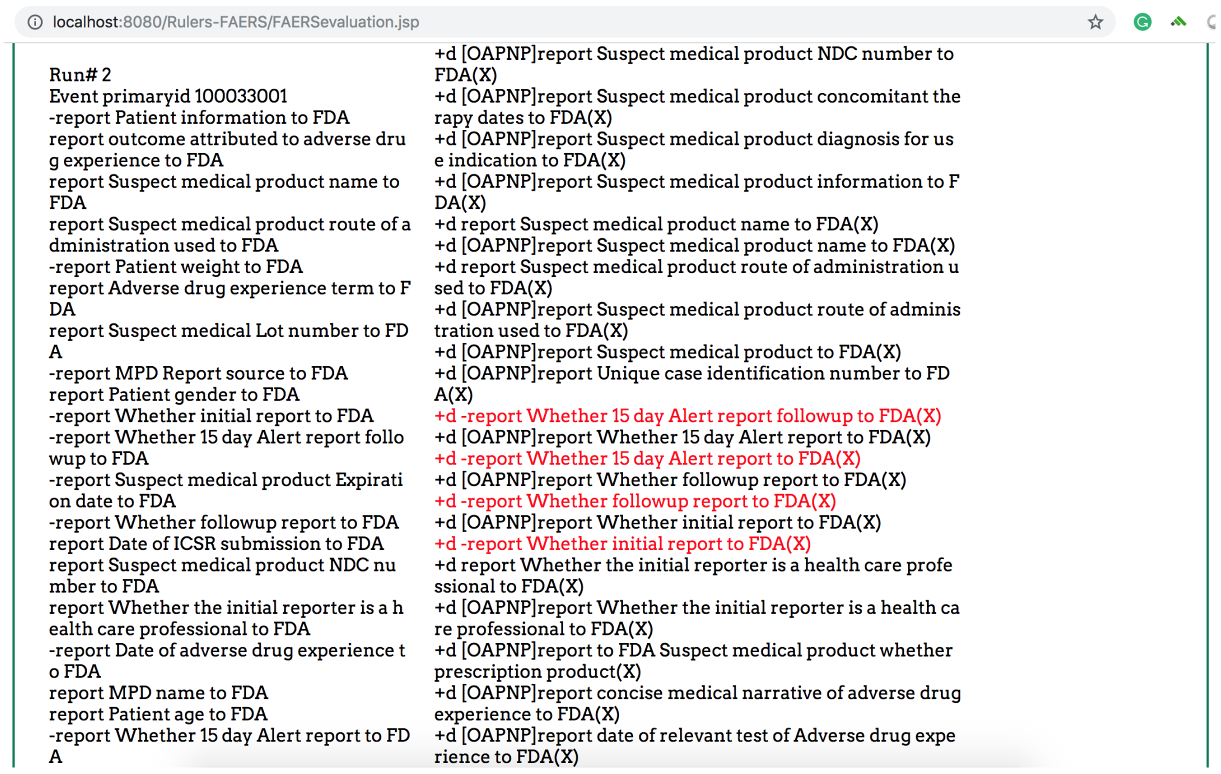


**Figure 6.5:** Report for Time Consumption to Process the Data from FAERS Database

Figure: 6.6 shows the GUI for the adverse events analysis system only from the ChildSafe database. It represents the analyzed conclusions from the stored adverse events from the ChildSafe database. Figure: 6.7 illustrates the conclusions (+D means the conclusion is definitely provable and others are


localhost:8080/Rulers-FAERS/FAERSevaluation.jsp

```
Run# 2                                         +d [OAPNP]report Suspect medical product NDC number to
Event primaryid 100033001                      FDA(X)
-report Patient information to FDA             +d [OAPNP]report Suspect medical product concomitant the
report outcome attributed to adverse dru       rapy dates to FDA(X)
g experience to FDA                            +d [OAPNP]report Suspect medical product diagnosis for us
report Suspect medical product name to         e indication to FDA(X)
FDA                                            +d [OAPNP]report Suspect medical product information to F
report Suspect medical product route of a      DA(X)
dministration used to FDA                      +d report Suspect medical product name to FDA(X)
-report Patient weight to FDA                  +d [OAPNP]report Suspect medical product name to FDA(X)
report Adverse drug experience term to F       +d report Suspect medical product route of administration u
DA                                             sed to FDA(X)
report Suspect medical Lot number to FD        +d [OAPNP]report Suspect medical product route of adminis
A                                              tration used to FDA(X)
-report MPD Report source to FDA               +d [OAPNP]report Suspect medical product to FDA(X)
report Patient gender to FDA                   +d [OAPNP]report Unique case identification number to FD
-report Whether initial report to FDA          A(X)
-report Whether 15 day Alert report follo      +d -report Whether 15 day Alert report followup to FDA(X)
wup to FDA                                     +d [OAPNP]report Whether 15 day Alert report to FDA(X)
-report Suspect medical product Expirati       +d -report Whether 15 day Alert report to FDA(X)
on date to FDA                                 +d [OAPNP]report Whether followup report to FDA(X)
-report Whether followup report to FDA         +d -report Whether followup report to FDA(X)
report Date of ICSR submission to FDA          +d [OAPNP]report Whether initial report to FDA(X)
report Suspect medical product NDC nu          +d -report Whether initial report to FDA(X)
mber to FDA                                    +d report Whether the initial reporter is a health care profe
report Whether the initial reporter is a h     ssional to FDA(X)
ealth care professional to FDA                 +d [OAPNP]report Whether the initial reporter is a health ca
-report Date of adverse drug experience t      re professional to FDA(X)
o FDA                                          +d [OAPNP]report to FDA Suspect medical product whether
report MPD name to FDA                         prescription product(X)
report Patient age to FDA                      +d [OAPNP]report concise medical narrative of adverse drug
-report Whether 15 day Alert report to FD      experience to FDA(X)
A                                              +d [OAPNP]report date of relevant test of Adverse drug expe
                                               rience to FDA(X)
```

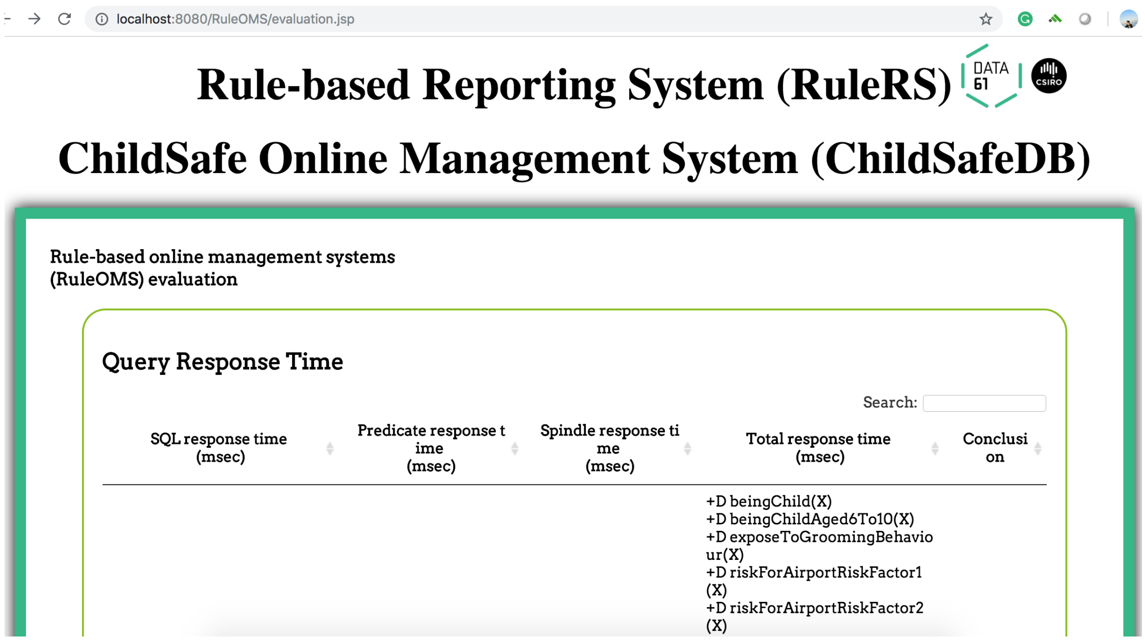
**Figure 6.4:** Graphical User Interface (GUI) for Evaluation only from FAERS Database

explained in Section: 3.2) and the rules which are in action for the generated output. Finally, the study represents the time consumption with predicate response time, response time per predicate, SPINdle response time and total response time in Figure: 6.8. The number of iterations is represented in the x-axis of the illustration and response time in milliseconds (msec) is represented in the y-axis of the illustration. The time consumption for the ChildSafe database depends on the Database Structure of ChildSafe, Number of predicates to be processed during the execution, Nature of data(whether indexed or not indexed structured/unstructured), the method following which the data is being stored continuously, SPINdle theory parsing using theory analyzer and SPINdle processing time.

The end users need to set up the database authentication parameters for the Predicate Translation Database (PRTDB) to fetch data from all the available databases (in our proposed study i.e., FAERS & ChildSafe database) as shown in Figure: 6.9. Here, the username and password for the PRTDB are entered and save to a session variable to be used in the runtime of the developed Rule Based System.

After saving the database authentication credentials for the Predicate Translation Database (PRTDB). The users are able to choose a unique identification number (as illustrated in Figure: 6.10) from the available databases for processing the stored adverse events. Here, unique identification numbers for both of the databases are randomly fetched from the database for providing flexibility to the end users. After, entering the unique identification number for any of the databases, click the submit button from the GUI. Then, data is generated by DSR and PRF functions according to supplied facts and formal rules which is shown in Figure: 6.11. The database name is automatically selected with the authentication parameters and other constraints generated by DSR function. It also shows the number of rows returned by the PRF function.

The study on Rule Based System load the theories are generated using DL into through SPINdle Rule



Rule-based Reporting System (RuleRS)

ChildSafe Online Management System (ChildSafeDB)

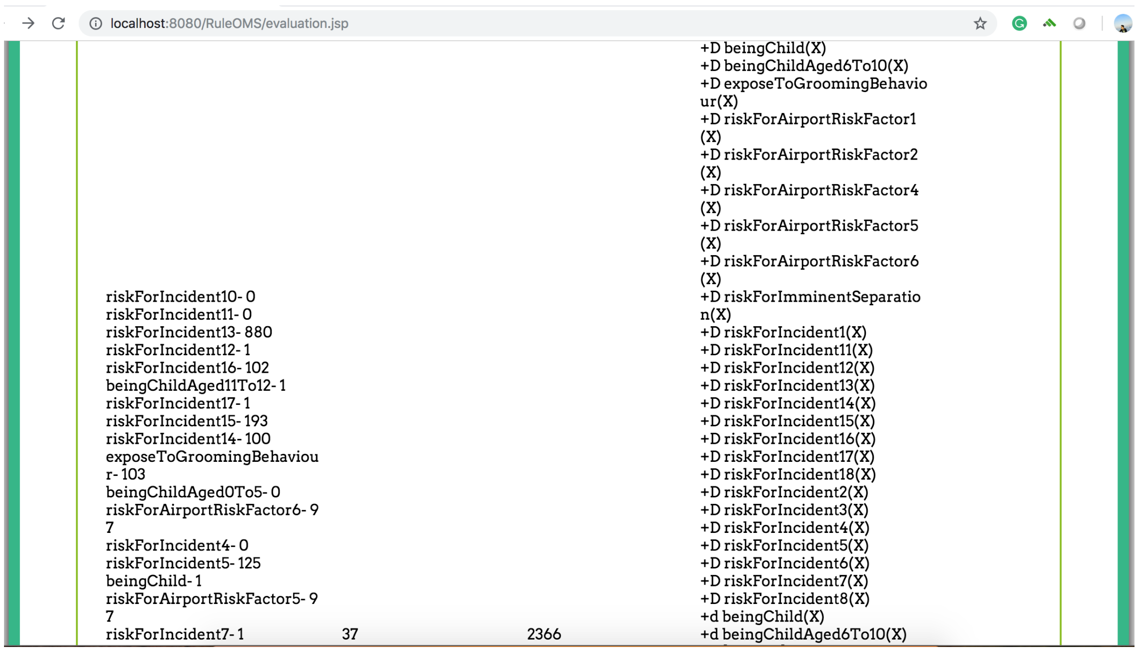
Rule-based online management systems (RuleOMS) evaluation

Query Response Time

Search:

| SQL response time (msec) | Predicate response time (msec) | Spindle response time (msec) | Total response time (msec) | Conclusion |
|---|---|---|---|---|
| | | | | +D beingChild(X)<br>+D beingChildAged6To10(X)<br>+D exposeToGroomingBehaviour(X)<br>+D riskForAirportRiskFactor1(X)<br>+D riskForAirportRiskFactor2(X) |

Figure 6.6: Graphical User Interface (GUI) for only ChildSafe Database



```
                                                      +D beingChild(X)
                                                      +D beingChildAged6To10(X)
                                                      +D exposeToGroomingBehavio
                                                      ur(X)
                                                      +D riskForAirportRiskFactor1
                                                      (X)
                                                      +D riskForAirportRiskFactor2
                                                      (X)
                                                      +D riskForAirportRiskFactor4
                                                      (X)
                                                      +D riskForAirportRiskFactor5
                                                      (X)
                                                      +D riskForAirportRiskFactor6
                                                      (X)
riskForIncident10- 0                                  +D riskForImminentSeparatio
riskForIncident11- 0                                  n(X)
riskForIncident13- 880                                +D riskForIncident1(X)
riskForIncident12- 1                                  +D riskForIncident11(X)
riskForIncident16- 102                                +D riskForIncident12(X)
beingChildAged11To12- 1                               +D riskForIncident13(X)
riskForIncident17- 1                                  +D riskForIncident14(X)
riskForIncident15- 193                                +D riskForIncident15(X)
riskForIncident14- 100                                +D riskForIncident16(X)
exposeToGroomingBehaviou                              +D riskForIncident17(X)
r- 103                                                +D riskForIncident18(X)
beingChildAged0To5- 0                                 +D riskForIncident2(X)
riskForAirportRiskFactor6- 9                          +D riskForIncident3(X)
7                                                     +D riskForIncident4(X)
riskForIncident4- 0                                   +D riskForIncident5(X)
riskForIncident5- 125                                 +D riskForIncident6(X)
beingChild- 1                                         +D riskForIncident7(X)
riskForAirportRiskFactor5- 9                          +D riskForIncident8(X)
7                                                     +d beingChild(X)
riskForIncident7- 1          37           2366        +d beingChildAged6To10(X)
```

Figure 6.7: Graphical User Interface (GUI) for Evaluation only from ChildSafe Database

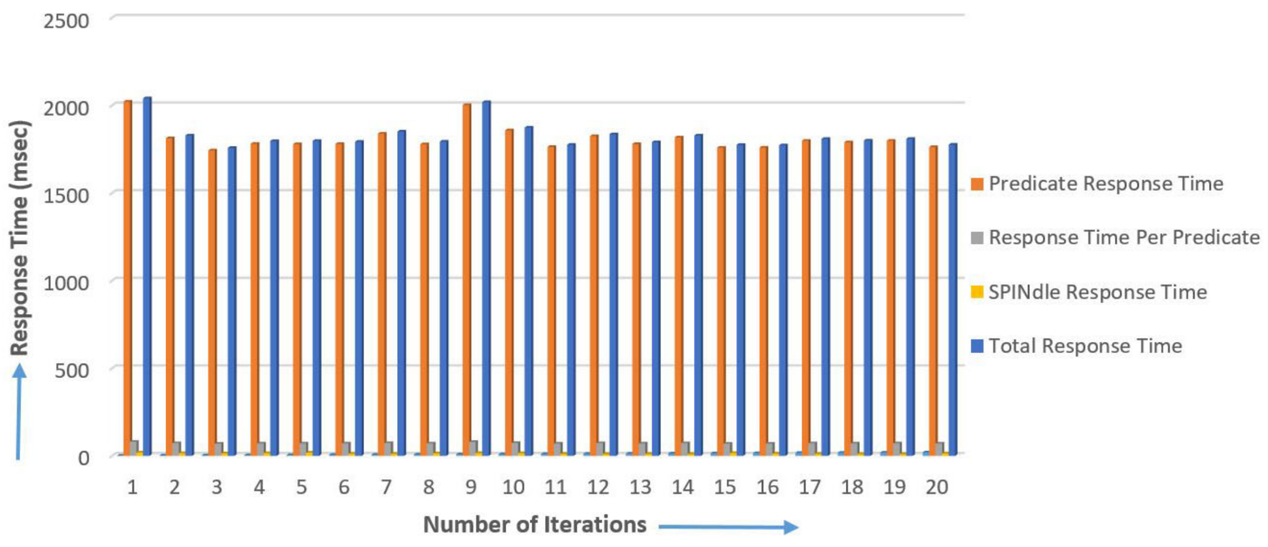
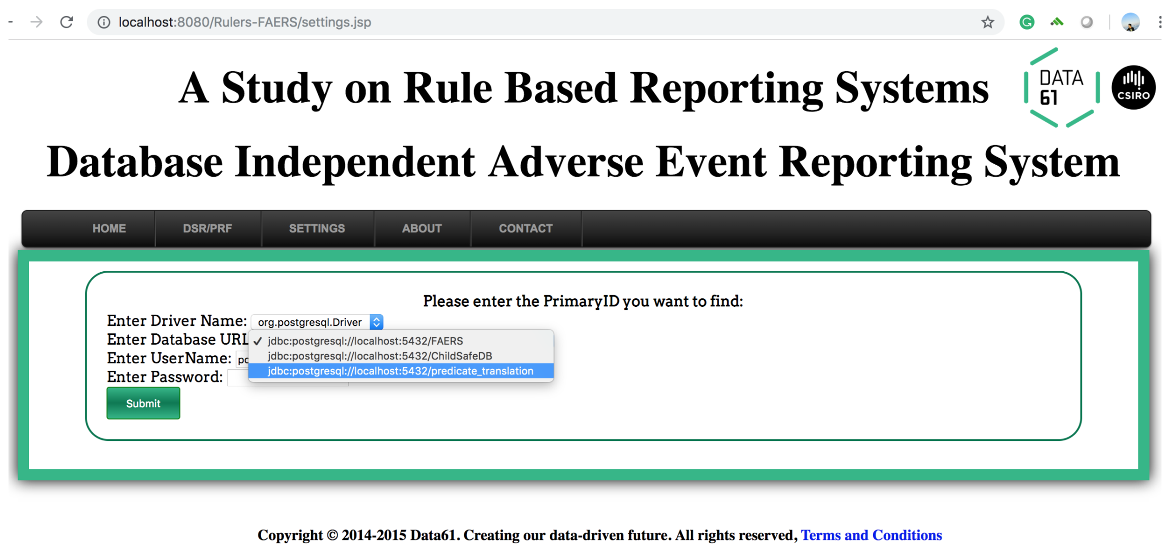


**Figure 6.8:** Report for Time Consumption to Process the Data from ChildSafe Database



A Study on Rule Based Reporting Systems

Database Independent Adverse Event Reporting System

Please enter the PrimaryID you want to find:

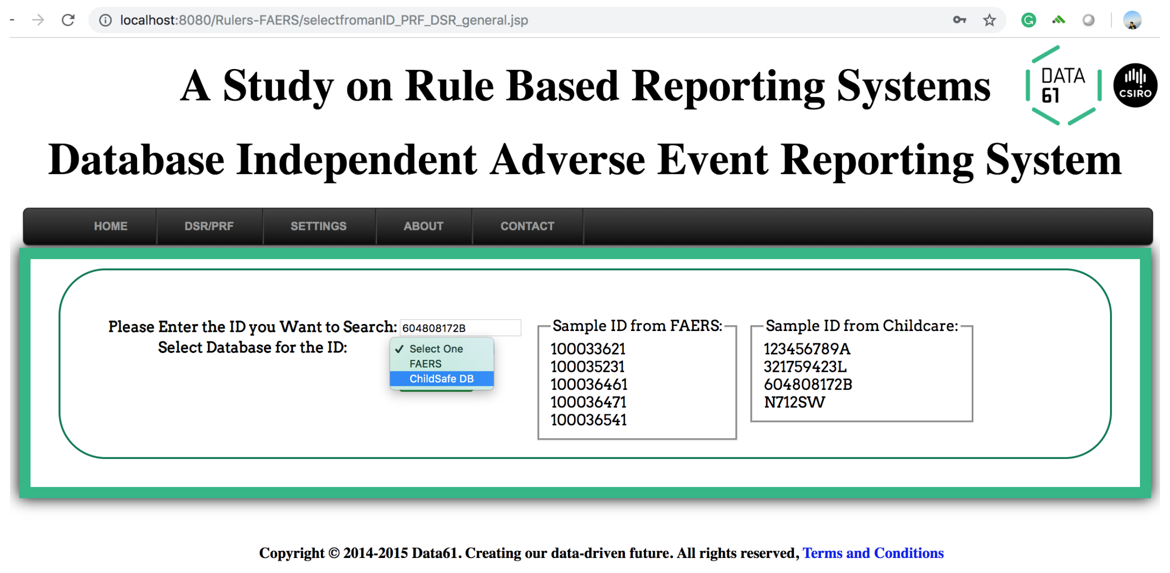
Enter Driver Name: org.postgresql.Driver

Enter Database URL: ✓ jdbc:postgresql://localhost:5432/FAERS
jdbc:postgresql://localhost:5432/ChildSafeDB
jdbc:postgresql://localhost:5432/predicate_translation

Enter UserName:

Enter Password:

**Figure 6.9:** Graphical User Interface (GUI) for Input Unique ID Number for Both the Databases



localhost:8080/Rulers-FAERS/selectfromanID_PRF_DSR_general.jsp
DATA 61
CSIRO
HOME
DSR/PRF
SETTINGS
ABOUT
CONTACT

A Study on Rule Based Reporting Systems

Database Independent Adverse Event Reporting System

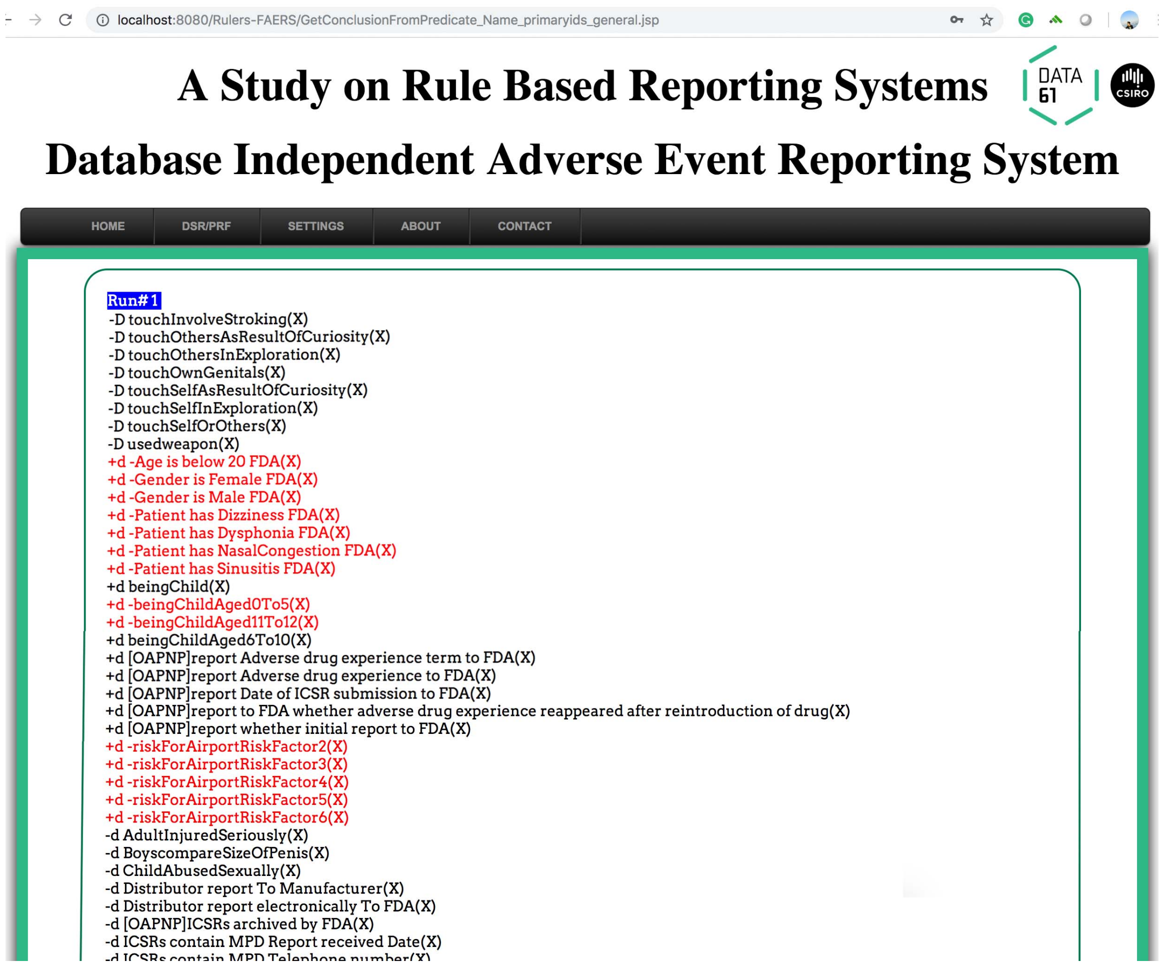
Please Enter the ID you Want to Search: 604808172B

Select Database for the ID: ✓ Select One
FAERS
ChildSafe DB

Sample ID from FAERS:
100033621
100035231
100036461
100036471
100036541

Sample ID from Childcare:
123456789A
321759423L
604808172B
N712SW

**Figure 6.10:** Graphical User Interface (GUI) for Input Unique ID Number for Both the Databases


localhost:8080/Rulers-FAERS/GetConclusionFromPredicate_Name_primaryids_general.jsp
DATA 61
CSIRO
HOME
DSR/PRF
SETTINGS
ABOUT
CONTACT

A Study on Rule Based Reporting Systems

Database Independent Adverse Event Reporting System

```
Run#1
-D touchInvolveStroking(X)
-D touchOthersAsResultOfCuriosity(X)
-D touchOthersInExploration(X)
-D touchOwnGenitals(X)
-D touchSelfAsResultOfCuriosity(X)
-D touchSelfInExploration(X)
-D touchSelfOrOthers(X)
-D usedweapon(X)
+d -Age is below 20 FDA(X)
+d -Gender is Female FDA(X)
+d -Gender is Male FDA(X)
+d -Patient has Dizziness FDA(X)
+d -Patient has Dysphonia FDA(X)
+d -Patient has NasalCongestion FDA(X)
+d -Patient has Sinusitis FDA(X)
+d beingChild(X)
+d -beingChildAged0To5(X)
+d -beingChildAged11To12(X)
+d beingChildAged6To10(X)
+d [OAPNP]report Adverse drug experience term to FDA(X)
+d [OAPNP]report Adverse drug experience to FDA(X)
+d [OAPNP]report Date of ICSR submission to FDA(X)
+d [OAPNP]report to FDA whether adverse drug experience reappeared after reintroduction of drug(X)
+d [OAPNP]report whether initial report to FDA(X)
+d -riskForAirportRiskFactor2(X)
+d -riskForAirportRiskFactor3(X)
+d -riskForAirportRiskFactor4(X)
+d -riskForAirportRiskFactor5(X)
+d -riskForAirportRiskFactor6(X)
-d AdultInjuredSeriously(X)
-d BoyscompareSizeOfPenis(X)
-d ChildAbusedSexually(X)
-d Distributor report To Manufacturer(X)
-d Distributor report electronically To FDA(X)
-d [OAPNP]ICSRs archived by FDA(X)
-d ICSRs contain MPD Report received Date(X)
-d ICSRs contain MPD Telephone number(X)
```

**Figure 6.11:** Output After Processing with DSR() and PRF() Function and SPINDLE Rule Reasoner

Based reasoner. And, the rule reasoner process all the rules and facts to generate conclusions as shown in Figure: 6.11. After all, we can evaluate the generated system with a good Rule Based reasoning system through the outcome of our study. And, the generated methods can be easily customized to fit for big data analysis using this Rule Based Systems. The time required for processing all the input predicates

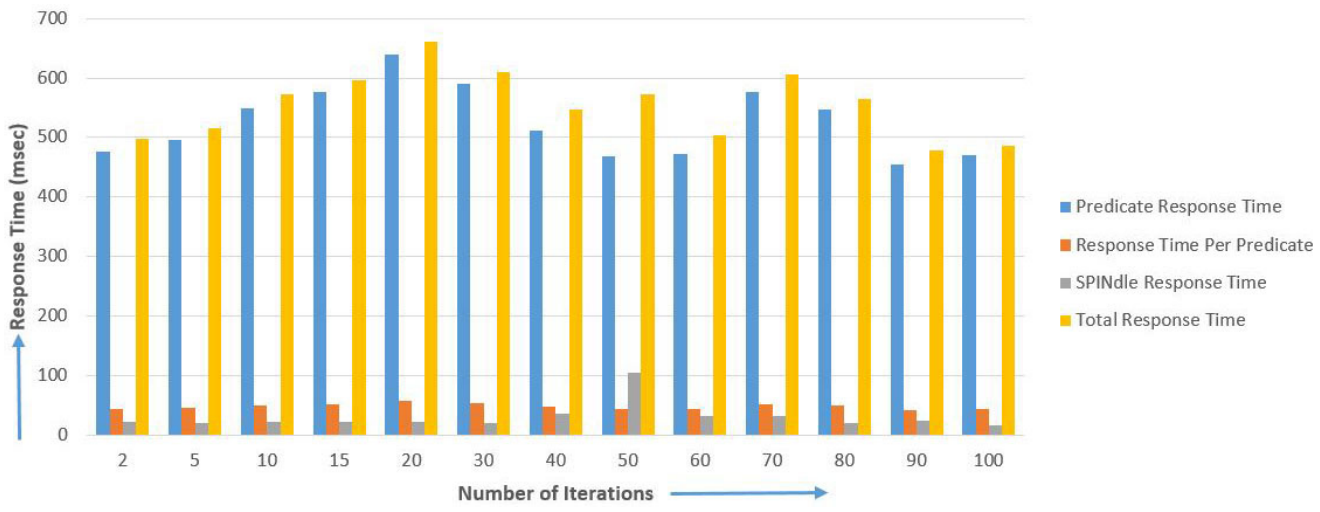


**Figure 6.12:** Report for Time Consumption to Process the Data from Both Databases

based on the provided rules & regulations is visualized in Figure: 6.12. The assumption retrieval time for SPINdle is also shown in the figure. From the graph in the figure, we can see that after some iterations the predicate response time decreases with the SPINdle's response time. The study represents the time consumption with predicate response time, response time per predicate, SPINdle response time and total response time in Figure: 6.12 for both the databases. The number of iterations is represented in the x-axis of the illustration and response time in milliseconds (msec) is represented in the y-axis of the illustration. The time consumption for both the databases depends on the Database Structure of available databases (i.e., FAERS & ChildSafe database), Number of predicates to be processed during the execution, Nature of data(whether indexed or not indexed structured/unstructured), the method following which the data is being stored continuously, SPINdle theory parsing using theory analyzer and SPINdle processing time. In this circumstances, the study generates database independent Rule Based System to process the adverse events does not require plenty time for generating conclusions using given facts (with simple SQL queries and complex SQL queries with join operations) and formal rules.

# Chapter VII
# Conclusions and Future Works

## 7.1 Conclusions

Research is growing exponentially on Rule Based Systems for generating decisions by analyzing data from databases. This is why we have completed our study on database independent model to analyze the adverse events over the methodologies of Rule Based Online Management System (RuleOMS). We have focused on the simultaneous access to a set of databases. Our system design required advanced mathematical and analytical abilities to succeed in drawing database independence(i.e., retrieving data on demand from multiple databases) to analyze the data. After removing the involved risks, we started to plan the implementation of a study on Rule Based System over the existing Rule Based method. At first, we analyzed the drawbacks for existing Rule Based methods to analyze adverse events. We found that all the Rule Based Systems works with a single source of data. So, we designed a Rule Based System which does not depend on the data source. For database independence in Rule Based System design, we introduced the predicate translation database (PRTDB) which translates all the predicates in SQL/JSON format to a database record with all the parameters. Data source retrieval (DSR) function was used to fetch the database authentication information with field data from the database and transfer to the presumption retrieval function (PRF). And, PRF was used to generate SQL query to fetch data from the database with and without join operation and conditional statements.

Defeasible Logic (DL) was used to generate rules for improved business analytic abilities to analyze adverse events. SPINdle rule reasoner was responsible for generating conclusions by loading theories generated by defeasible logic. The graphical user interface (GUI) was designed using *JAVA* programming language. We installed the software (i.e., Eclipse, Glassfish/ Tomcat) on the local computers with running Windows and MAC operating systems to generate show the web interface for our developed program. End users are able to input data into the system and our database independent Rule Based System is generating conclusions based on the predefined rules & regulations. Also, the study on Rule Based System produces a facility for both the end users and conclusions processing systems by removing the strain of database selection. The study outputs database independent Rule Based System which can adopt databases automatically for generating conclusions from the given facts on adverse events. Moreover, the generated Rule Based System is effective for analyzing large datasets.

Our study on database independent analysis of adverse events using Rule Based Systems has a huge impact on the technological world. In the study, we have only experimented with the adverse events from two sample databases which can be featured with extra features for analyzing adverse events to extract conclusions from multiple databases.

## 7.2 Recommendations for Future Works

The growing need for data analysis for conclusions and reporting is a great problem now a day. There are a huge amount of structured and unstructured data-sets. The Government & professional organizations, educational institutions, scientific research, sensors and more generate datum. Rule Based Reporting System covers the integration of rules and regulations for making a decision over adverse events stored in the FDA & ChildSafe database. So, our main objective of this study on Rule Based System is to generate assumptions and alerts from data stored in multiple databases using DL and SPINdle rule reasoner based on a set of guidelines. We also studied about the extension of the proposed Rule Based Systems for simultaneous reasoning of adverse events from multiple data sources and listed the following recommendations.

- The proposed research method can be extended to generate conclusions from unstructured data like generated log files(i.e., stored as text files).
- The data cleaning (removing/ ignoring *NULL* values) may be introduced with the proposed Rule Based System.
- The grammatical/ syntax errors can be minimized to detect the repetitions of the words in the available dataset.
- The method for generating the rules can be changed to evaluate the impact for the processing of adverse events.
- The proposed methodology can be introduced with more features of artificial intelligence for generating conclusions
- Generating conclusions from terabytes of data to make decisions.
- High computation power may be used to generate conclusions quickly.

Finally, we can say that the study on Rule Based Systems is very useful for making decisions instantly by computation from different types of data sources. The Rule Based System eases the hard works introducing artificial intelligence in decision making.

## References

1. Mohammad Badiul Islam and Guido Governatori. Ruleoms: A rule-based online management system. In *Proceedings of the ICAIL 2015*, pages 187–191, New York, NY, USA, 2015. ACM.

2. Mohammad Badiul Islam and Guido Governatori. Rulers: a rule-based architecture for decision support systems. *Artificial Intelligence and Law*, 2018.

3. Brian Lam. Spindle-user guide. 2010.

4. Guido Governatori and Antonino Rotolo. A conceptually rich model of business process compliance. In *Proceedings of the Seventh Asia-Pacific Conference on Conceptual Modelling-Volume 110*, pages 3–12. Australian Computer Society, Inc., 2010.

5. Mustafa Hashmi, Guido Governatori, and Moe Thandar Wynn. Business process data compliance. In *International Workshop on Rules and Rule Markup Languages for the Semantic Web*, pages 32–46. Springer, 2012.

6. US Food, Drug Administration, et al. About fda, 2009. *Preliminary public health notification: serious complications associated with negative pressure wound therapy systems*, 2013.

7. Mads Kamper-Jørgensen, Jan Wohlfahrt, Jacob Simonsen, and Christine S Benn. The childcare database: a valuable register linkage. *Scandinavian Journal of Social Medicine*, 35(3):323–329, 2007.

8. Iosif Viktoratos, Athanasios Tsadiras, and Nick Bassiliades. Plis+: A rule-based personalized location information system. In *Proceedings of the RuleML 2012 @ ECAI Challenge*, number 874 in CEUR Workshop Proceedings, 2012.

9. Kevin J Leonard. The development of a rule based expert system model for fraud alert in consumer credit. *European journal of operational research*, 80(2):350–356, 1995.

10. R Hevner von Alan, Salvatore T March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS quarterly*, 28(1):75–105, 2004.

11. Ho-Pun Lam and Guido Governatori. The making of spindle. In *Proceedings of the RuleML 2009*, number 5858 in LNCS, pages 315–322. Springer, 2009.

12. Okkyung Choi and Sang Yong Han. Personalization of rule-based web services. *Sensors*, 8(4):2424–2435, 2008.

13. Raihan Ul Islam, Karl Andersson, and Mohammad Shahadat Hossain. A web based belief rule based expert system to predict flood. In *Proceedings of the 17th International conference on information integration and web-based applications & services*, page 3. ACM, 2015.

14. Cristhian AD Deagustini, Santiago E Fulladoza Dalibón, Sebastián Gottifredi, Marcelo A Falappa, Carlos I Chesñevar, and Guillermo R Simari. Relational databases as a massive information source for defeasible argumentation. *Knowledge-Based Systems*, 51:93–109, 2013.

15. Shazia Sadiq and Guido Governatori. *Managing Regulatory Compliance in Business Processes*, pages 159–175. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

16. F Burstein and CW Holsapple. Handbook on decision support systems 1: Basic themes. 2008.

17. Robert Koons. Defeasible reasoning. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Spring 2014 edition, 2014.

18. Donald Nute. Defeasible logic. In *International Conference on Applications of Prolog*, pages 151–169. Springer, 2001.

19. Donald Nute. *Defeasible deontic logic*, volume 263. Springer Science & Business Media, 2012.

20. Norman W Paton and Oscar Díaz. Active database systems. *ACM Computing Surveys (CSUR)*, 31(1):63–103, 1999.

21. Senlin Liang, Paul Fodor, Hui Wan, and Michael Kifer. Openrulebench: an analysis of the performance of rule engines. In *Proceedings of the 18th international conference on World wide web*, pages 601–610. ACM, 2009.

22. J Kozák. Rules in database systems. *Proceedings of Contributed Papers WDS'11*, pages 131–136, 2011.

23. Allen Newell, Herbert Alexander Simon, et al. *Human problem solving*, volume 104. Prentice-Hall Englewood Cliffs, NJ, 1972.

24. Emil L Post. Formal reductions of the general combinatorial decision problem. *American journal of mathematics*, 65(2):197–215, 1943.

25. Toshiyuki Sakaeda, Kaori Kadoyama, and Yasushi Okuno. Adverse event profiles of platinum agents: data mining of the public version of the fda adverse event reporting system, aers, and reproducibility of clinical observations. *International journal of medical sciences*, 8(6):487, 2011.

26. Mohamed A Omar and James P Wilson. Fda adverse event reports on statin-associated rhabdomyolysis. *Annals of Pharmacotherapy*, 36(2):288–295, 2002.

27. Jane B Grimson. Integrating knowledge-based systems and databases. *Clinica chimica acta*, 222(1-2):101–115, 1993.

28. Luigi Ceccaroni. Integration of a rule-based expert system, a case-based reasoner and an ontological knowledge-base in the wastewater domain. 2000.

29. Guido Governatori and Antonino Rotolo. A conceptually rich model of business process compliance. In *Proceedings of the APCCM 2010*, number 110 in CRPIT, pages 3–12. ACS, 2010.

30. Mustafa Hashmi, Guido Governatori, and Moe Thandar Wynn. Normative requirements for regulatory compliance: An abstract formal framework. *Information Systems Frontiers*, 18(3):429–455, 2016.

31. M Sasikumar, S Ramani, S Muthu Raman, KSR Anjaneyulu, and R Chandrasekar. A practical introduction to rule based expert systems. *New Delhi: Narosa Publishing House*, 2007.

32. Dilip Kumar Choubey, Sanchita Paul, Vinay Kumar Dh, et al. Rule based diagnosis system for diabetes. *Biomedical Research*, 28(12):5196–5209, 2017.

33. László Lengyel. Validating rule-based algorithms. *Acta Polytechnica Hungarica*, 12(4), 2015.

34. Donald Nute. Defeasible logic. In Dov M. Gabbay, Chris H. Hogger, and J.A. Robinson, editors, *Handbook of logic in artificial intelligence and logic programming*, volume 3, pages 353–395. Oxford University Press, 1994.

35. Carlos Iván Chesñevar, Ana Gabriela Maguitman, and Ronald Prescott Loui. Logical models of argument. *ACM Comput. Surv.*, 32(4):337–383, December 2000.

36. Cristhian AD Deagustini, Santiago E Fulladoza Dalibón, Sebastián Gottifredi, Marcelo A Falappa, and Guillermo R Simari. Consistent query answering using relational databases through argumentation. In *International Conference on Database and Expert Systems Applications*, pages 1–15. Springer, 2012.

37. Cristhian Ariel David Deagustini, Fulladoza Dalibón, Santiago Emanuel, Sebastián Gottifredi, Marcelo Alejandro Falappa, Carlos Iván Chesñevar, and Guillermo Ricardo Simari. Defeasible argumentation over relational databases. *Argument & Computation*, 8(1):35–59, 2017.

38. Franck Michel, Johan Montagnat, and Catherine Faron-Zucker. *A survey of RDB to RDF translation approaches and tools*. PhD thesis, I3S, 2014.

39. Nick Bassiliades, Grigoris Antoniou, and Ioannis Vlahavas. A defeasible logic reasoner for the semantic web. 2(1):1–41, 2006.

40. Thomas Skylogiannis, Grigoris Antoniou, Nick Bassiliades, Guido Governatori, and Antonis Bikakis. Dr-negotiate— a system for automated agent negotiation with defeasible logic-based strategies. *Data & Knowledge Engineering*, 63:362–380, 2007.

41. Grigoris Antoniou, David Billington, Guido Governatori, and Michael J Maher. On the modeling and analysis of regulations. In *Australian Conference on Information Systems*, 1999.

42. Guido Governatori. Representing business contracts in RuleML. *International Journal of Cooperative Information Systems*, 14(2-3):181–216, June-September 2005.

43. Benjamin N. Grosof. Representing e-commerce rules via situated courteous logic programs in RuleML. *Electronic Commerce Research and Applications*, 3(1):2–20, 2004.

44. Shazia Sadiq and Guido Governatori. Managing regulatory compliance in business processes. In Jan vom Brocke and Michael Rosemann, editors, *Handbook of Business Process Management 2nd edition*, volume 2, pages 265–288. Springer, 2 edition, 2015.

45. Guido Governatori and Sidney Shek. Regorous: A business process compliance checker. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Law*, pages 245–246, 2013.

46. Guido Governatori. On the relationship between Carneades and defeasible logic. In *Proceedings of the ICAIL 2011*, pages 31–40. ACM, 2011.

47. Grigoris Antoniou, David Billington, Guido Governatori, and Michael J Maher. Representation results for defeasible logic. 2(2):255–287, 2001.

48. Guido Governatori. The Regorous approach to process compliance. In *2015 IEEE 19th International Enterprise Distributed Object Computing Workshop*, pages 33–40. IEEE Press, 2015.

49. Guido Governatori and Mustafa Hashmi. No time for compliance. In *Enterprise Distributed Object Computing Conference (EDOC), 2015 IEEE 19th International*, pages 9–18. IEEE, 2015.

50. Guido Governatori and Antonino Rotolo. Bio logical agents: Norms, beliefs, intentions in defeasible logic. 17(1):36–69, 2008.

51. NSW Government. The nsw mandatory reporter guide, 2016. Last accessed on 4 November 2018.

52. U.S. Food and Drug Administration. Fda adverse event reporting system (faers): Latest quarterly data files, December 2015. Accessed on 15 March 2016.

53. Michael J Maher. Propositional defeasible logic has linear complexity. *Theory and Practice of Logic Programming*, 1(06):691–711, 2001.

54. David Billington, Grigoris Antoniou, Guido Governatori, and Michael J. Maher. An Inclusion Theorem for Defeasible Logics. *ACM Transactions in Computational Logic*, 12(1):1–27, 2010.

55. Guido Governatori. Business process compliance: An abstract normative framework. *IT – Information Technology*, 55(6):231–238, 2013.

56. Mustafa Hashmi, Guido Governatori, and Moe Thandar Wynn. Normative requirements for regulatory compliance: An abstract formal framework. 2015.

57. Guido Governatori and Antonino Rotolo. Logic of violations: A Gentzen system for reasoning with contrary-to-duty obligations. *Australasian Journal of Logic*, 4:193–215, 2006.

58. Guido Governatori, Francesco Olivieri, Antonino Rotolo, and Simone Scannapieco. Computing strong and weak permission in defeasible logic. *Journal of Philosophical Logic*, 42(6):799–829, nov 2013.

59. Guido Governatori and Antonino Rotolo. Defeasible logic: Agency, intention and obligation. In *Proceedings of the DEON 2004*, number 3065 in LNCS, pages 114–128. Springer, 2004.

60. PS Yu, MS Chen, and HU Heiss. On workload characterization of relational database environments. 1992.

61. International Business Machines Corporation. Ibm db2 universal database sql reference version 8, 1993, 2001. Last accessed on 4 November 2018.