OpenRules®

Open Source Business Decision Management System

Release 6.2.1

Decision Modeling Tutorial

"Determine Patient Therapy"

Part 2

Getting Data from Database

OpenRules, Inc.

www.openrules.com

June-2012

Table of Contents

INTRODUCTION

This document is the second part of the decision modeling tutorial "Determine Patient Therapy". In the first part we described a business case when a doctor makes a decision about the required therapy for an encounter diagnosis. Here is a simplified scenario when the diagnosis is Acute Sinusitis.

Medication Rules:

If Patient is 18 years old or older, then a therapy choice is Amoxicillin.
If Patient is younger than 18, a therapy choice is Cefuroxime.
If patient Penicillin allergic, therapy of choice is Levofloxacin.
Check if patient on active medication. Coumadin and Levofloxacin can result in reduced effectiveness of Coumadin. Produce the proper warning.

Dosing Rules:
For patients between 15 and 60, the dose is 500mg every 24 hours for 14 days.
If Patient's creatinine level > 1.4, commence creatinine clearance (CCr) calculation according to the formula:

$$\text{CCr, in mL/min} = \frac{(140 - \text{age}) \times \text{lean body weight [kg]}}{\text{PCr [mg/dL]} \times 72}$$

If patient's creatinine clearance < 50 ml/min, then the dose is 250mg every 24 hours for 14 days.
Dosing also depends on renal function, immune state, or liver function but the proper rules will be added later.

We provided a detailed description of how to create the proper executable decision with OpenRules® using only Excel. The patient and visit information was defined in Excel data tables. This decision is available as a part of the standard OpenRules® installation as the project "DecisionPatientTherapy".

In this document we will described a new project "DecisionPatientTherapyDB" that does the same but reads patient data from a relational database.

DATABASE

For demonstration purposes we will use MS Access. However, we will rely on the JDBC interface, so any relational database can be similarly connected with OpenRules® based decisions.

MS Access provides a standard template "Students" with basic information about students. We may consider students as our patients and simply add a few additional fields to the table "Students" to support our business case.

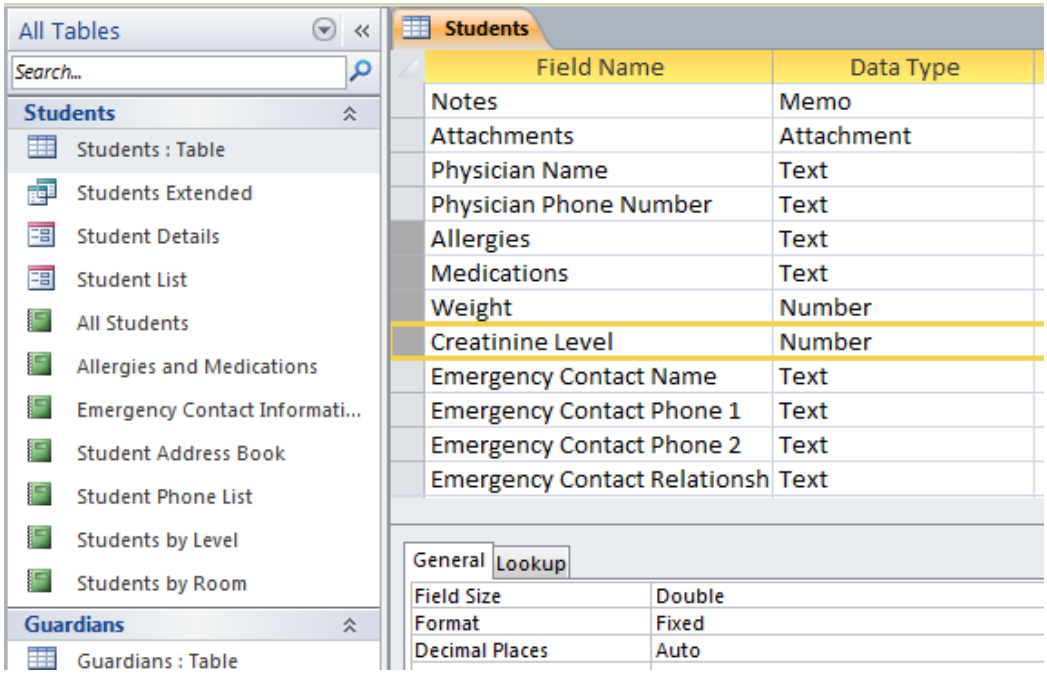
Creating Database "DBPatients"

First, we will create a new project "DecisionPatientTherapyDB" by simply copying our previoud project "DecisionPatientTherapy". We will add a subdirectory "db" and from MS Access create a database "db/DBStudents.accdb" using the template "Students".

We will open the table "Students" in the design mode and will add necessary attributes such as:

- Weight
- Creatinine Level
- Allergies
- Medications

The modified table "Students" will look like on the snapshot below:



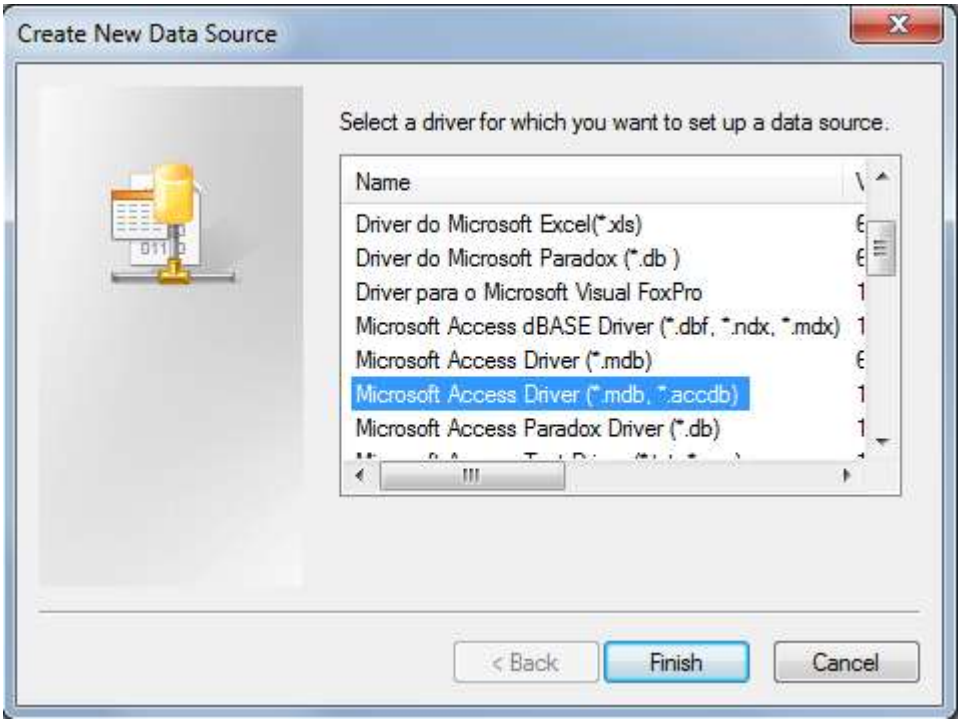
We may add a few test patients (students) to this table directly from the MS Access Datasheet view. We will use the same John Smith and Mary Smith that we used in the Part 1 for testing purposes but defined in the data Excel table "patients".

While this database (as any real database) contains a lot of other information, our objective is simply to read patient data from the table "Students" and pass it through our updated decision "DeterminePatientTherapy" (described in the Part 1):

Decision DeterminePatientTherapy	
Decisions	**Execute Decision Tables**
Define Medication	:= DefineMedication()
Define Creatinine Clearance	:= CalculateCreatinineClearance()
Define Dosing	:= DefineDosing()

Adding Data Source "DBPatients"

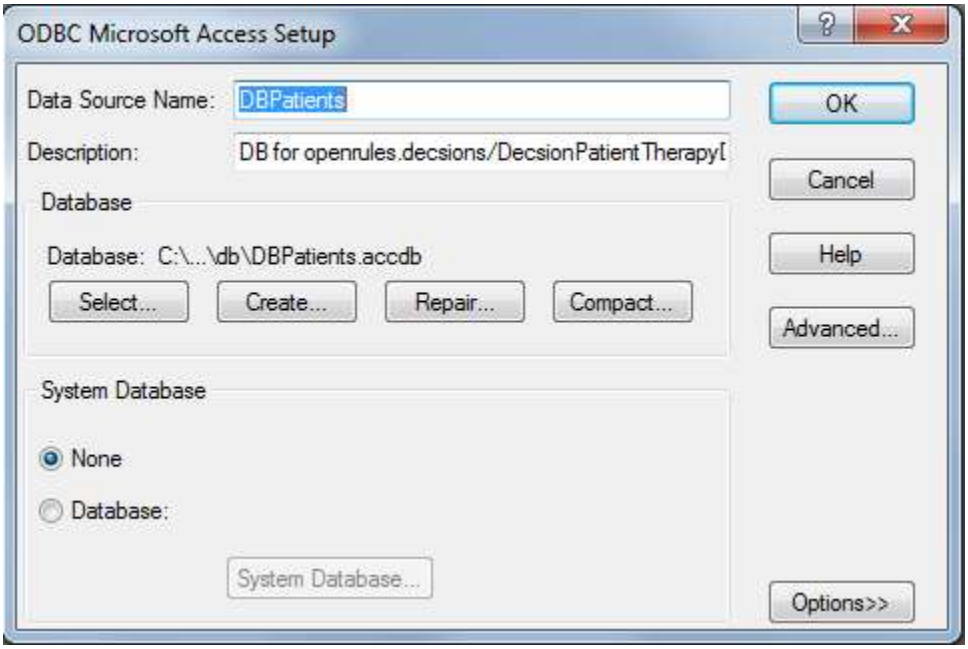
To make our database known to different applications, we need to add the proper data source. To do this in MS Windows we need to open a Control Panel, select Administrative Tools, and then "Set up data sources (ODBC)". It will open "ODBC Data Source Administrator". Select the tab "System DSN" and click on "Add". You will see the following dialog:



Select "Microsoft Access Driver (*mdb,*accdb)" and click on "Finish".

Note. If you have problems to find Microsoft Access Driver, read http://goo.gl/w4vNo or http://msdn.microsoft.com/en-us/library/windows/desktop/ms712362(v=vs.85).aspx.

Then you will see this dialog:



Enter Data Source Name "DBPatients", then any description, and click on "Select" to select already created file "DBPatients.accdb". Then click "OK". When we close other dialogs, a new data source "DBPatients" will become available. Now we should be able to read it from a Java program using a JDBC interface.

SUPPORTING JAVA CLASSES

We want to read patients from the database table "Students" and save them as instances of Java class Patient. We also want to create a test instance of a Java class Visit for different patients and run it against our Excel-based decision model. So, we will need to create the proper Java classes in the package "healthcare" that support these actions.

Java Classes for Patients and Visits

Previously we used the data type "Patient" defined in this Excel table:

Datatype Patient	
String	name
int	age
double	creatinineLevel
double	creatinineClearance
String[]	allergies
double	weight
String	activeMedication

Now we will replace it with the following Java class:

```
public class Patient {

    String       name;
    int          age;
    String       gender;
    double       creatinineLevel;
    double       creatinineClearance;
    String[]     allergies;
    double       weight;
    String[]     activeMedications;
    static int   max = 3;

    public Patient() {
           allergies = new String[max];
           activeMedications = new String[max];
           for (int i = 0; i < max; i++) {
                  allergies[i] = "None";
                  activeMedications[i] = "None";
           }
      // getters and setters
}
```

Note that we replaced a single "activeMedication" with an array of Strings "activeMedications". We will also allow no more than 3 allergies and no more than 3 active medications.

Previously we also used the data type "Visit" defined in this Excel table:

Datatype DoctorVisit	
Date	date
String	encounterDiagnosis
String	recommendedMedication
String	recommendedDose

Now we will replace it with the following Java class:

```
public class Visit {

   Date         date;
   Patient      patient;
   String       encounterDiagnosis;
   String       medication;
   String       dose;
   String       warning;
   String[]     patients;

   public Visit() {
          encounterDiagnosis = "Acute Sinusitis";
          date = Calendar.getInstance().getTime();
          patients = null;

   // getters and setters
}
```

Java Iterator for Reading Patients from Database

Now we are ready to create a simple Java class "PatientIterator" that should allow us to iterate through all patients available from the database table "Students". We will define the class PatientIterator as a subclass of the class DatabaseIterator included in the standard OpenRules project "com.openrules.tools". Here is its code:

```
import java.sql.Timestamp;
import org.apache.commons.beanutils.DynaBean;
import com.openrules.tools.DatabaseIterator;

public class PatientIterator extends DatabaseIterator {

   public PatientIterator(String dbName, String tableName) {
          super(dbName, tableName);
   }

   public Patient nextPatient() {
          DynaBean bean = next();

          Patient patient = new Patient();
          String first = (String) bean.get("first name");
          String last = (String) bean.get("last name");
          String gender = (String) bean.get("gender");
          patient.setName(first + " " + last);
          patient.setGender(gender);
          Timestamp dob = (Timestamp) bean.get("date of birth");
          patient.setAge(dob);
          Double weight = (Double) bean.get("weight");
          patient.setWeight(weight.doubleValue());
          Double cl = (Double) bean.get("creatinine level");
          patient.setCreatinineLevel(cl.doubleValue());
          String allergies = (String) bean.get("allergies");
          if (allergies != null) {
```

```
                String[] array = allergies.split("\\r\\n");
                patient.setAllergies(array);
            }

            String medications = (String) bean.get("medications");
            if (medications != null) {
                String[] array = medications.split("\\r\\n");
                patient.setActiveMedication(array);
            }
            return patient;
    }

    public static void main(String[] args) {
            PatientIterator iter =
                new PatientIterator("DBPatients", "Students");
            while(iter.hasNext()) {
                Patient patient = iter.nextPatient();
                System.out.println(patient.toString());
            }
    }
}
```

This class extends DatabaseIterator with only one method "nextPatient" that basically maps fields from a database table "Students" to the Java class Patient.

We may test this class against the actual database "DBPatients" by simply executing it from Eclipse as a Java application. Here are the execution results:

```
Connecting to 'DBPatients' using 'sun.jdbc.odbc.JdbcOdbcDriver' ...
Connected
Reading Students...
Patient [name=John Smith, age=21, gender=Male, creatinineLevel=2.0,
creatinineClearance=0.0, allergies=[Penicillin, Streptomycin, None],
weight=180.0, activeMedications=[Coumadin, None, None]]
Patient [name=Mary Smith, age=19, gender=Female, creatinineLevel=2.35,
creatinineClearance=0.0, allergies=[None, None, None], weight=151.0,
activeMedications=[None, None, None]]
```

Java Decision Launcher

Now we may switch to the actual decision. Previously we had a simple decision's launcher defined in the file "Main.java" of the package "healthcare":

```
import com.openrules.ruleengine.Decision;

public class Main {
    public static void main(String[] args) {
            String fileName = "file:rules/DecisionPatientTherapy.xls";
            Decision decision =
               new Decision("DeterminePatientTherapy",fileName);
            decision.execute();
    }
}
```

A newly created PatientIterator allows us to re-write our launcher as follows:

```
import com.openrules.ruleengine.Decision;

public class Main {
    public static void main(String[] args) {
           String fileName = "file:rules/DecisionPatientTherapy.xls";
           Decision decision =
               new Decision("DeterminePatientTherapy",fileName);
           PatientIterator iter =
               new PatientIterator("DBPatients", "Students");
           while(iter.hasNext()) {
               Patient patient = iter.nextPatient();
               System.out.println("\n" + patient);
               Visit visit = new Visit();
               visit.setPatient(patient);
               decision.put("visit",visit);
               decision.execute();
           }
           iter.close();
    }
}
```

So, for every patient received from a database we will create an object visit and attach this object to the decision using `decision.put("visit",visit);`. We need to remind a reader that the default OpenRules® class Decision is a hash map that support methods "put" and "get".

To run this decision launcher, we still need to make the proper changes in our Excel-based decision model described in the file "`DecisionPatientTherapy.xls`".

UPDATING EXCEL-BASED DECISION MODEL

First of all, we do not need any more Excel tables that defined data types and data instances. So, we will delete tables:

Datatype Patient
Datatype Visit
Data Visit visits
Data Patient patients.

We also have to make changes in the table "Glossary". Here is our previous table:

Glossary glossary		
Variable	**Business Concept**	**Attribute**
Encounter Diagnosis	Doctor Visit	encounterDiagnosis
Recommended Medication		recommendedMedication
Recommended Dose		recommendedDose
Patient Age	Patient	age
Patient Weight		weight
Patient Allergies		allergies
Patient Creatinine Level		creatinineLevel
Patient Creatinine Clearance		creatinineClearance
Patient Active Medication		activeMedication

What are our changes?

1) Instead of one active medication now we use an array of active medications that is defined in the Java class "Patient" as an array of Strings "medications".
2) Our Java class "Visit" uses "medication" and "dose" instead of "recommendedMedication" and "recommendedDose".

So, here is an update glossary:

Glossary glossary		
Fact Name	**Business Concept**	**Attribute**
Encounter Diagnosis	Doctor Visit	encounterDiagnosis
Recommended Medication		medication
Recommended Dose		dose
Patient Age	Patient	age
Patient Weight		weight
Patient Allergies		allergies
Patient Creatinine Level		creatinineLevel
Patient Creatinine Clearance		creatinineClearance
Patient Active Medications		activeMedications

As we change the name of the decision variable "Patient Active Medication", we also have to change this name in the decision table that uses this variable. Previously this decision table was defined as:

DecisionTable WarnAboutDrugInteraction				
Condition		Condition		Message
Recommended Medication		**Patient Active Medication**		**Warning**
Is	Levofloxacin	Is	Coumadin	Coumadin and Levofloxacin can result in reduced effectiveness of Coumadin.

The update table will look as follows:

DecisionTable WarnAboutDrugInteraction				
Condition		Condition		Message
Recommended Medication		**Patient Active Medications**		**Warning**
Is	Levofloxacin	Include	Coumadin	Coumadin and Levofloxacin can result in reduced effectiveness of Coumadin.

As you may notice, we not only change the variable name (by adding "s") but also replaced the operator "Is" to operator "Include".

And finally, to make our model connected to our new Java objects, we need to replace the old table

DecisionObject decisionObjects	
Business Concept	Business Object
Patient	:= patients[1]
Doctor Visit	:= visits[0]

with this table

DecisionObject decisionObjects	
Business Concept	Business Object
Patient	:= decision.get("patient")
Doctor Visit	:= decision.get("visit")

EXECUTING UPDATED DECISION MODEL

Now we may execute our decision model by running Main.java as a Java application from Eclipse on by simply double-clicking on run.bat. Here are new results:

```
Connecting to 'DBPatients' using 'sun.jdbc.odbc.JdbcOdbcDriver' ...
Connected
Reading Students...

Patient [name=John Smith, age=21, gender=Male, creatinineLevel=2.0,
creatinineClearance=0.0, allergies=[Penicillin, Streptomycin, None], weight=180.0,
activeMedications=[Coumadin, None, None]]
*** Decision DeterminePatientTherapy ***
Decision has been initialized
Decision DeterminePatientTherapy: Define Medication
Conclusion: Recommended Medication Is Levofloxacin
Decision DeterminePatientTherapy: Define Creatinine Clearance
Conclusion: Patient Creatinine Clearance Is 148.75
Decision DeterminePatientTherapy: Define Dosing
Conclusion: Recommended Dose Is 500mg every 24 hours for 14 days
Decision DeterminePatientTherapy: Check Drug Interaction
WarnAboutDrugInteraction: Coumadin and Levofloxacin can result in reduced
effectiveness of Coumadin.
Decision has been finalized
```

```
Patient [name=Mary Smith, age=19, gender=Female, creatinineLevel=2.35,
creatinineClearance=0.0, allergies=[None, None, None], weight=151.0,
activeMedications=[None, None, None]]
Decision DeterminePatientTherapy: Define Medication
Conclusion: Recommended Medication Is Amoxicillin
Decision DeterminePatientTherapy: Define Creatinine Clearance
Conclusion: Patient Creatinine Clearance Is 107.98463356973994
Decision DeterminePatientTherapy: Define Dosing
Conclusion: Recommended Dose Is 500mg every 24 hours for 14 days
Decision DeterminePatientTherapy: Check Drug Interaction
Decision has been finalized
```

You may find these documents at the following locations

Part 1: http://openrules.com/pdf/Tutorial.DecisionPatientTherapy.pdf

Part 2: http://openrules.com/pdf/Tutorial.DecisionPatientTherapyDB.pdf.