

OpenCyc as a database

Authors: Dmitri Pissarenko and George Herson

WWW: <http://dapissarenko.com>

Table Of Contents

1. Introduction.....	3
2. Data model in human language.....	3
3. Data model in CycL	4
4. Asking OpenCyc questions	6
5. Overview over the application	7
6. Constructor of class UI	8
7. Connecting to the database.....	10
8. Entering data into the OpenCyc database.....	12
9. Asking questions	18
10. CycL vs. SQL.....	23
11. Final words	27
12. Appendix: Related materials.....	29

1. Introduction

In this paper I will explain how *OpenCyc* can be used as a database. We will create an application, which

1. enters data into *OpenCyc*
2. presents them via a Swing GUI.

But first, let's me explain what kind of data we want to store in our database.

2. Data model in human language

Imagine, we want to create an application, which models the government system of a certain country. Let's call this hypothetical country *Ordus'*. For the sake of simplicity we will focus only on a part of the executive branch of Ordussian government.

There are civil servants. There are two types of them:

- those, who have subordinates
- those, who don't have subordinates.

Each civil servant has a field of work, for which he/she is responsible. Each civil servant is a person. Each civil servant has a name. Each civil servant works in a department. A department has also a field of work and a head of department.

There are two departments in *Ordus'*:

- department of internal affairs
- department of ethical supervision.

Field of work of department for internal affairs is protection of citizens against criminals. Head of department is Rededyya Alimagomedov.

There is one subordinate of Rededyya Alimagomedov – civil servant Bagatur Lobo, whose field of work is finding and catching criminals.

Field of work of department of ethical supervision is ensuring that the laws of *Ordus'* are followed by its subjects. In other words department of ethical supervision is Ordussian analogon to the ministry of justice.

Head of department of ethical supervision is Mokii Nilovich Rabinovich. He has a subordinate Bogdan Rukhovich Ouintsev-Su, whose field of work is legal advice to Bagatur Lobo.

3. Data model in CycL

We have now a verbal description of the data, which we have to put into the database. We need somehow to encode them. The first option is to draw an *entity-relationship diagram* and prepare the data for storage in a relational database. We won't use this option.

Instead, we translate verbal description into *OpenCyc* language *CycL*. *CycL* statements can then be entered into *OpenCyc* database either via its web interface or using Java. Look at the table below. You can see on the left hand side a verbal statement and on the right hand side its *CycL* equivalent.

English	CycL
Ordus' is a country.	(#\$isa #\$Ordus #\$Country)
Department is an organization.	(#\$genls #\$Department #\$Organization)
Department of internal affairs is a department.	(#\$isa #\$DepartmentOfInternalAffairs #\$Department)
Department of ethical supervision is a department.	(#\$isa #\$DepartmentOf EthicalSupervision #\$Department)
Field of work of department of internal affairs is protection of citizens against criminals.	(#\$fieldOfWork #\$DepartmentOfInternalAffairs "Protection of citizens against criminals")
Field of work of department of ethical supervision is ensuring that the laws of Ordus' are followed by its subjects.	(#\$fieldOfWork #\$DepartmentOfEthicalSupervision "Ensuring that the laws of Ordus are followed by its subjects")
A civil servant is a person.	(#\$genls #\$CivilServant #\$Person)
Given names of Rededya Alimagomedov are <i>Rededya Peresvetovich</i> .	(#\$givenNames #\$RededyaAlimagomedov "Rededya Peresvetovich")
Family name of Rededya Alimagomedov is <i>Alimagomedov</i> .	(#\$familyName #\$RededyaAlimagomedov "Alimagomedov")
Given name of Bagatur Lobo is <i>Bagatur</i> .	(#\$givenNames #\$BagaturLobo

	"Bagatur")
Family name of Bagatur Lobo is <i>Lobo</i> .	(#\$familyName #\$BagaturLobo "Lobo")
Bagatur Lobo is subordinate of Rededya Alimagomedov.	(#\$superiors #\$RededyaAlimagomedov #\$BagaturLobo)
Field of work of Bagatur Lobo is finding and catching criminals.	(#\$fieldOfWork #\$BagaturLobo "Finding and catching criminals")
Given names of Mokii Nilovich Rabinovich are <i>Mokii Nilovich</i> .	(#\$givenNames #\$MokiiNilovichRabinovich "Mokii Nilovich")
Family name of Mokii Nilovich Rabinovich is <i>Rabinovich</i> .	(#\$familyName #\$MokiiNilovichRabinovich "Rabinovich")
Given names of Bogdan Rukhovich Ouiantsev-Su are <i>Bogdan Rukhovich</i> .	(#\$givenNames #\$BogdanRukhovichOuiantsevSu "Bogdan Rukhovich")
Family name of Bogdan Rukhovich Ouiantsev-Su is <i>Ouiantsev-Su</i> .	(#\$familyName #\$BogdanRukhovichOuiantsevSu "Ouiantsev-Su")
Field of work of Bogdan Rukhovich Ouiantsev-Su is legal advice to Bagatur Lobo.	(#\$fieldOfWork #\$BogdanRukhovichOuiantsevSu "Legal advice to Bagatur Lobo")
Bogdan Rukhovich Ouiantsev-Su is subordinate of Mokii Nilovich Rabinovich.	(#\$superiors #\$MokiiNilovichRabinovich #\$BogdanRukhovichOuiantsevSu)
Head of department of internal affairs is Rededya Alimagomedov.	(#\$headOfDepartment #\$DepartmentOfInternalAffairs #\$RededyaAlimagomedov)
Head of department of ethical supervision is Mokii Nilovich Rabinovich.	(#\$headOfDepartment #\$DepartmentOfEthicalSupervision #\$MokiiNilovichRabinovich)
Mokii Nilovich Rabinovich is a civil servant.	(#\$isa #\$MokiiNilovichRabinovich #\$CivilServant)

Here we use *Cycl* for the same purpose as *entity-relationship diagram*, i. e. for translating knowledge from human language to machine-readable language. In my opinion, *Cycl* representation is more easier to read and to understand than an entity-relationship representation, particularly for non-IT people.

We enter the data into *OpenCyc* using the class *OrdussianPoliticalSystemMtWriter*. The

application class for writing data into *OpenCyc* database is *Demo3*.

4. Asking OpenCyc questions

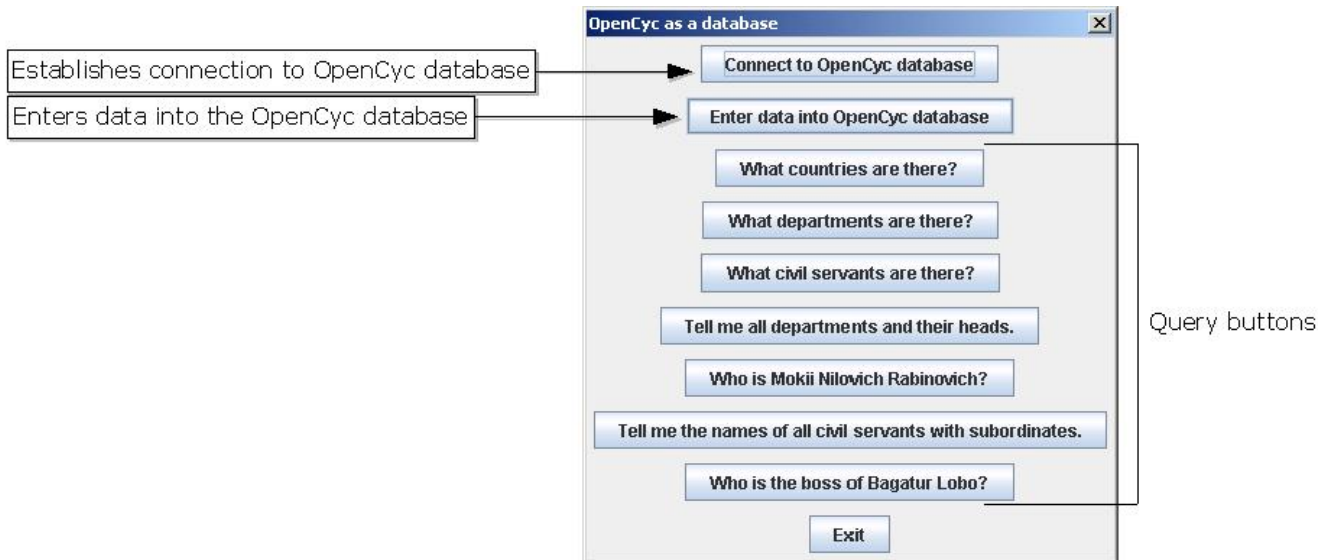
We have now entered data into the *OpenCyc* database. We can now query (ask questions) about the political system of Ordus'. Again, below you will find questions in English and their equivalents in *CycL*.

English	CycL
What countries are there?	(#\$isa ?COUNTRY #\$Country)
What departments are there?	(#\$isa ?DEPARTMENT #\$Department)
What civil servants are there?	(#\$isa ?CIVIL_SERVANT #\$CivilServant)
Tell me all departments and their heads.	(#\$headOfDepartment ?HEAD_OF_DEPARTMENT ?DEPARTMENT)
Who is Mokii Nilovich Rabinovich?	(#\$isa #\$MokiinilovichRabinovich ?PROPERTY)
Tell me the names of all civil servants with subordinates.	(#\$superiors ?BOSS ?SUBORD)
Who is the boss of Bagatur Lobo?	(#\$superiors ?BOSS #\$BagaturLobo)

We will now write a simple Java application which displays the results of these queries.

5. Overview over the application

An overview about how our application works is given in viewlet OpenCyc as a database. Following image shows the structure of the application's main screen.



In the following section, we will look at how this application works.

6. Constructor of class UI

The class containing the main method is *UI*. First, let's look at its constructor, which is defined at line 108:

```
public UI()
{
    try
    {
        this.dialog = (JDialog) (new SwingEngine(this)).render(
            ClassLoader.
            getSystemClassLoader().
            getResource(UI_DEF_FILE_NAME));
    }
    catch (Exception exception)
    {
        this.logger.error("", exception);
    }
    this.dialog.pack();
}
```

When writing this application, I used *SwiXML*, which is a great time-saver. We don't define the user interface in our Java class. Instead, we tell *SwiXML* what the dialog box should look like, it generates that dialog box automatically and the result is assigned to the member variable `this.dialog`, which is an instance of `javax.swing.JDialog`.

There remains only one question: what does the cryptic statement

```
ClassLoader.getSystemClassLoader().getResource(UI_DEF_FILE_NAME))
```

mean? This statement simply loads the file *UI.xml*, which is contained in the JAR file of the application. This is why we need this `ClassLoader` stuff.

SwiXML does not only create the dialog box for us, it also establishes connection between our public action members and the related events of the dialog. For instance, in *UI.xml* we wrote

```
<button id="connectButton" text="Connect to OpenCyc database"
Action="connectButtonAction">
```

```
<gridbagconstraints insets="5,5,5,5" gridx="0" gridy="0"/>
</button>
```

In *UI.java*, at line 46 we define an `AbstractAction`, which has the same name as the action in the XML file:

```
public AbstractAction connectButtonAction = new AbstractAction() {
    public void actionPerformed(ActionEvent event) {
        connectButtonAction();
    }
};
```

Whenever the user presses the *Connect to OpenCyc database* button, the `actionPerformed` method of this action is invoked, which invokes the method *connectButtonAction*. In the next section we explore this method.

7. Connecting to the database

Let's look at the *connectButtonAction* method of class *UI*, which is responsible for establishing connection to the *OpenCyc* database.

```
private void connectButtonAction() {
    try {
        this.dialog.setCursor(Cursor
            .getPredefinedCursor(Cursor.WAIT_CURSOR));

        this.cycAccess = new CycAccess("localhost");
        this.cycAccess.traceOn();
        this.dataAccess = new DataAccessLayer(this.cycAccess);
        try {
            this.mt = this.cycAccess
                .getKnownConstantByName(
                    "OrdussianPoliticalSystemMt");
        } catch (Exception exception) {
        }
        this.dialog.setCursor(Cursor
            .getPredefinedCursor(Cursor.DEFAULT_CURSOR));
        JOptionPane
            .showMessageDialog(this.dialog,
                "Connection to OpenCyc database has been
                successfully established.");
    } catch (Exception exception) {
        this.dialog.setCursor(Cursor
            .getPredefinedCursor(Cursor.DEFAULT_CURSOR));
        this.logger.error("", exception);
        JOptionPane
            .showMessageDialog(
                this.dialog,
                "An error occured while trying to connect to
                OpenCyc database. Perhaps OpenCyc server is
                not running.");
    }
}
}
```

`this.cycAccess = new CycAccess("localhost");` establishes connection to *OpenCyc* server, which is running at localhost.

`this.cycAccess.traceOn()`; tells *this.cycAccess*, our object for accessing OpenCyc database, to print log messages.

`this.dataAccess = new DataAccessLayer(this.cycAccess)`; creates an instance of class *DataAccessLayer*, which encapsulates methods for querying the database.

Then, we try to fetch the micro-theory *OrdussianPoliticalSystemMt* using the statement

```
this.mt = this.cycAccess.getKnownConstantByName(  
    "OrdussianPoliticalSystemMt");
```

If there is no such micro-theory in the database, an exception is thrown. We ignore this exception. It is no problem, if there is no such micro-theory - it was simply not created yet (method, which creates this micro-theory, is described in the next section). All other statements require no comments.

8. Entering data into the OpenCyc database

When the user presses the *Enter data into OpenCyc database*, method *UI.enterDataIntoOpenCycButtonAction* is invoked, which is defined in file *UI.java* on line 136. This method creates an instance of class *OrdussianPoliticalSystemMtWriter*, which does the work of creating our micro-theory and filling it with knowledge. Below you find the source code of this class. It demonstrates how to enter information into the *OpenCyc* database. I recommend you to compare this source code with the CycL statements we wrote above in section *Data model in CycL*.

```
public class OrdussianPoliticalSystemMtWriter {
    private static final String COLLECTION = "Collection";
    public OrdussianPoliticalSystemMtWriter()
    {
    }
    public CycConstant writeMicroTheoryToCyc(CycAccess cyc) throws
    IOException, CycApiException
    {
        CycConstant ordus=null;
        CycConstant department=null;
        CycConstant mt=null;
        CycConstant depIntAff=null;
        CycConstant depEthSup=null;
        CycConstant fieldOfWork=null;
        CycConstant headOfDepartment=null;
        CycConstant alimagomedov=null;
        CycConstant civilServant=null;
        CycConstant lobo=null;
        CycConstant ouiantsevSu=null;
        CycConstant rabinovich=null;
        CycFort givenNames=null;
        CycFort familyName=null;
        CycFort superiors=null;

        /**
         * create microtheory
         */
        mt=cyc.createMicrotheory("OrdussianPoliticalSystemMt",
            "This microtheory describes the political system
            of Ordus'",
            "Microtheory",
```

```

        new ArrayList());

/**
 * Enter some countries
 * (#$isa #$Russia #$Country)
 * (#$isa #$Belarus #$Country)
 * (#$isa #$Austria #$Country)
 * (#$isa #$Germany #$Country)
 */
this.createCountry("Russia", "Russian Federation", cyc, mt);
this.createCountry("Belarus", "Belorussia", cyc, mt);
this.createCountry("Austria", "Republic of Austria", cyc, mt);
this.createCountry("Germany", "Federative Republic of
        Germany", cyc, mt);

/**
 * Ordus' is a country
 * (#$isa #$Ordus #$Country)
 */
ordus=cyc.createNewPermanent("Ordus");
cyc.assertComment(ordus, "Hypothetical country", mt);
cyc.assertIsa(ordus, cyc.getKnownConstantByName("Country"), mt);

/**
 * (#$genls #$Department #$Organization)
 */
department=cyc.createNewPermanent("Department");
cyc.assertIsa(department,
cyc.getKnownConstantByName(COLLECTION), mt);
cyc.assertGenls(department,
cyc.getKnownConstantByName("Organization"), mt);

/**
 * (#$isa #$DepartmentOfInternalAffairs #$Department)
 */
depIntAff=cyc.createNewPermanent("DepartmentOfInternalAffairs");
cyc.assertIsa(depIntAff, department, mt);

/**
 * (#$isa #$DepartmentOfEthicalSupervision #$Department)
 */
depEthSup=cyc.createNewPermanent("DepartmentOfEthicalSupervision");

```

```

cyc.assertIsa(depEthSup, department, mt);

/**
 * create fieldOfWork predicate
 */
fieldOfWork=cyc.createNewPermanent("fieldOfWork");
cyc.assertIsa(fieldOfWork,
cyc.getKnownConstantByName("BinaryPredicate"), mt);
cyc.assertComment(fieldOfWork, "(fieldOfWork a b) means that b
is field of work of a", mt);

/**
 * (#$fieldOfWork #$DepartmentOfInternalAffairs "Protection of
citizens against criminals")
 */
cyc.assertGaf(mt, fieldOfWork, depIntAff, "Protection of
citizens against criminals");

/**
 * (#$fieldOfWork #$DepartmentOfEthicalSupervision "Ensuring
that the laws of Ordus are followed by its subjects")
 */
cyc.assertGaf(mt, fieldOfWork, depEthSup, "Ensuring that the
laws of Ordus are followed by its subjects");

/**
 * create headOfDepartment predicate
 */
headOfDepartment=cyc.createNewPermanent("headOfDepartment");
cyc.assertIsa(headOfDepartment,
cyc.getKnownConstantByName("BinaryPredicate"), mt);
cyc.assertComment(headOfDepartment, "(fieldOfWork a b) means
that b is head of department a", mt);

/**
 * create CivilServant collection
 * (#$genls #$CivilServant #$Person)
 */
civilServant=cyc.createNewPermanent("CivilServant");
cyc.assertIsa(civilServant,
cyc.getKnownConstantByName(COLLECTION), mt);
cyc.assertGenls(civilServant,

```

```

cyc.getKnownConstantByName("Person"), mt);

/**
 * create RededyaAlimagomedov
 * (#$givenNames #$RededyaAlimagomedov "Rededya Peresvetovich")
 * (#$familyName #$RededyaAlimagomedov "Alimagomedov")
 */
givenNames=cyc.getKnownConstantByName("givenNames");
familyName=cyc.getKnownConstantByName("familyName");

alimagomedov=cyc.createNewPermanent("RededyaAlimagomedov");
cyc.assertIsa(alimagomedov, civilServant, mt);
cyc.assertGaf(mt, givenNames, alimagomedov, "Rededya
Peresvetovich");
cyc.assertGaf(mt, familyName, alimagomedov, "Alimagomedov");

/**
 * create BagaturLobo
 * (#$givenNames #$BagaturLobo "Bagatur")
 * (#$familyName #$BagaturLobo "Lobo")
 */
lobo=cyc.createNewPermanent("BagaturLobo");
cyc.assertIsa(lobo, civilServant, mt);
cyc.assertGaf(mt, givenNames, lobo, "Bagatur");
cyc.assertGaf(mt, familyName, lobo, "Lobo");

/**
 * (#$superiors #$RededyaAlimagomedov #$BagaturLobo)
 */
superiors=cyc.getKnownConstantByName("superiors");
cyc.assertGaf(mt, superiors, alimagomedov, lobo);

/**
 * (#$fieldOfWork #$BagaturLobo "Finding and catching
criminals")
 */
cyc.assertGaf(mt, fieldOfWork, lobo, "Finding and catching
criminals");

/**
 * create MokiiNilovichRabinovich
 * (#$givenNames #$MokiiNilovichRabinovich "Mokii Nilovich")

```

```

    * (#$familyName #$MokiiNilovichRabinovich "Rabinovich")
    * (#$isa #$MokiiNilovichRabinovich #$CivilServant)
    */
rabinovich=cyc.createNewPermanent("MokiiNilovichRabinovich");
cyc.assertIsa(rabinovich, civilServant, mt);
cyc.assertGaf(mt, givenNames, rabinovich, "Mokii Nilovich");
cyc.assertGaf(mt, familyName, rabinovich, "Rabinovich");

/**
 * create BogdanRukhovichOuiantsevSu
 *
 * (#$givenNames #$BogdanRukhovichOuiantsevSu "Bogdan
Rukhovich")
 * (#$familyName #$BogdanRukhovichOuiantsevSu "Ouiantsev-Su")
 */
ouiantsevSu=cyc.createNewPermanent("BogdanRukhovichOuiantsevSu");
cyc.assertIsa(ouiantsevSu, civilServant, mt);
cyc.assertGaf(mt, givenNames, ouiantsevSu, "Bogdan Rukhovich");
cyc.assertGaf(mt, familyName, ouiantsevSu, "Ouiantsev-Su");

/**
 * (#$fieldOfWork #$BogdanRukhovichOuiantsevSu "Legal advice to
Bagatur Lobo")
 */
cyc.assertGaf(mt, fieldOfWork, ouiantsevSu, "Legal advice to
Bagatur Lobo");

/**
 * (#$superiors #$MokiiNilovichRabinovich
#$BogdanRukhovichOuiantsevSu)
 */
cyc.assertGaf(mt, superiors, rabinovich, ouiantsevSu);

/**
 * (#$headOfDepartment #$DepartmentOfInternalAffairs
#$RededyAAlimagomedov)
 */
cyc.assertGaf(mt, headOfDepartment, depIntAff, alimagomedov);

/**
 * (#$headOfDepartment #$DepartmentOfEthicalSupervision
#$MokiiNilovichRabinovich)

```

```
        */
        cyc.assertGaf(mt, headOfDepartment, depEthSup, rabinovich);

        return mt;
    }

private void createCountry(String countryName, String comment,
    CycAccess cyc, CycConstant mt)
    throws IOException, CycApiException
{
    CycConstant country=null;

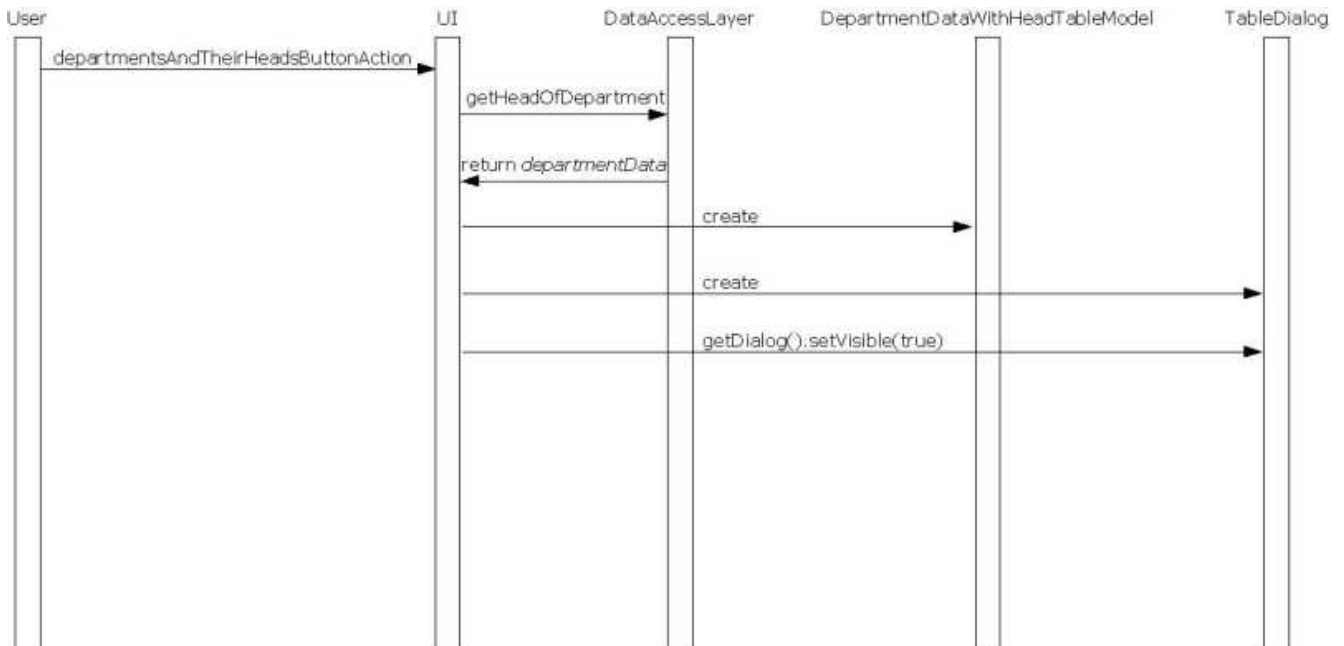
    country=cyc.createNewPermanent(countryName);
    cyc.assertComment(country, comment, mt);
    cyc.assertIsa(country,
        cyc.getKnownConstantByName("Country"),
        mt);
}
}
```

9. Asking questions

Now it is time how ask *OpenCyc* question, which is analogous to issuing SQL `SELECT` queries in relational databases. We have formulated our questions (queries) in section *Asking OpenCyc questions*. Now we have to

- execute these queries,
- fetch the results and
- pack the results into Java objects, which we can display in a table.

The code related to the questions we formulated differs only little from question to question, so we will present only one example. The remaining queries are issued in a similar manner. We will explore how the answer to the question *Tell me all departments and their heads* or (`#$headOfDepartment ?HEAD_OF_DEPARTMENT ?DEPARTMENT`) in CycL is created. For an overview, look at the following sequence diagram.



Our journey starts at line of class *UI*, where the method *departmentsAndTheirHeadsButtonAction* is defined:

```
private void departmentsAndTheirHeadsButtonAction() {
    TableDialog tableDialog = null;
    DepartmentDataWithHeadTableModel tableModel = null;
    Collection departmentData = null;

```

```

try {
    departmentData = this.dataAccess.getHeadOfDepartmentList(mt);

    tableModel = new DepartmentDataWithHeadTableModel(
        (DepartmentData[]) departmentData
            .toArray(
                new DepartmentData[departmentData.size()]));

    tableDialog = new TableDialog(tableModel);
    tableDialog.getDialog().setTitle(
        "Tell me all departments and their heads.");
    tableDialog.getDialog().setVisible(true);
} catch (Exception exception) {
    this.logger.error("", exception);
    JOptionPane
        .showMessageDialog(
            this.dialog,
            "An error occured while trying to execute
            this action. Perhaps OpenCyc server is
            not running.");
}
}
}

```

The majority of these lines of code is trivial user-interface programming. Just look at the classes `DepartmentDataWithHeadTableModel` and `TableDialog` and you will know that the table model just displays the contents of a collection of Java beans and `TableDialog` simply displays a dialog box with a table therein, model of which the constructor of the dialog takes as a parameter.

The interesting thing is how a collection of instances of *DepartmentData* class is created on line `departmentData = this.dataAccess.getHeadOfDepartmentList(mt);`. *DepartmentData* is a Java bean with three properties: *name*, *fieldOfWork* and *headOfDepartment*. Let's look at the inner workings of the method *DataAccessLayer.getHeadOfDepartmentList*:

```

public Collection getHeadOfDepartmentList(CycConstant mt)
throws CycApiException, IOException
{
    CycList query = null;
    CycList response = null;

```

```
CycVariable departmentVariable = null;
CycVariable headVariable = null;
CycList item=null;
DepartmentData departmentData=null;
Collection collection=null;
ArrayList variables=null;
Iterator iterator=null;
CycConstant department=null;
CycConstant head=null;
String fullHeadName=null;

collection=new Vector();

query = CycAccess.current().makeCycList(
"#$headOfDepartment ?HEAD_OF_DEPARTMENT ?DEPARTMENT");
departmentVariable =
    CycObjectFactory.makeCycVariable("?DEPARTMENT");
headVariable =
    CycObjectFactory.makeCycVariable("?HEAD_OF_DEPARTMENT");

variables=new ArrayList();
variables.add(departmentVariable);
variables.add(headVariable);

response = CycAccess.current().askWithVariables(query,
    variables, mt);

iterator=response.iterator();
while (iterator.hasNext())
{
    item=(CycList)iterator.next();

    head=(CycConstant)item.first();
    department=(CycConstant)item.second();
    fullHeadName=this.getFullHeadName(head, mt);

    departmentData=new DepartmentData();
    departmentData.setHeadOfDepartment(fullHeadName);
    departmentData.setName(department.getName());

    collection.add(departmentData);
}
```

```
    return collection;
}
```

We will now explore this code fragment line by line. First, we create query object using the statement

```
query = CycAccess.current().makeCycList(
    ("#$headOfDepartment ?HEAD_OF_DEPARTMENT ?DEPARTMENT"));
```

For the two placeholders `?HEAD_OF_DEPARTMENT` and `?DEPARTMENT` we create two OpenCyc variables `departmentVariable` and `headVariable`, respectively. This happens on lines

```
departmentVariable = CycObjectFactory.
    makeCycVariable("?DEPARTMENT");
headVariable = CycObjectFactory.
    makeCycVariable("?HEAD_OF_DEPARTMENT");
```

Next, we make a collection of variables. We need this collection in order to issue the OpenCyc query.

```
variables=new ArrayList();
variables.add(departmentVariable);
variables.add(headVariable);
```

Now we are ready to ask OpenCyc our question:

```
response = CycAccess.current().askWithVariables(
    query, variables, mt);
```

`query` is our query object, `variables` is our collection of variables and `mt` is the micro-theory, in which the predicate `#$headOfDepartment` is defined.

Next, we get the results of the query in form of an *Iterator*.

```
iterator=response.iterator();
```

We traverse this iterator in a *while* loop. In each iteration, we first obtain the record of the result set:

```
item=(CycList)iterator.next();
```

As the name suggests, `CycList` represents a list of values. In our case, these are values of `?HEAD_OF_DEPARTMENT` and `?DEPARTMENT`. `?HEAD_OF_DEPARTMENT` is the first element of `item` list, `?DEPARTMENT` the second (because of the order, in which the variables appear in the query!):

```
head=(CycConstant)item.first();
department=(CycConstant)item.second();
```

Next, we want to know not just the name of the constant, which represents head of department (for instance, `#$RededyaAlimagomedov`), but a well-formatted name string (for instance, `Rededya Peresvetovich Alimagomedov`). For this purpose, we call the method `getFullHeadName`, which asks the OpenCyc database for given and family names of a person and returns a string with the names.

```
fullHeadName=this.getFullHeadName(head, mt);
```

Finally, we pack name of department and head of department information into an instance of `DepartmentData` class and add this instance to the collection, which is the return value of this method.

This collection is then displayed in the table dialog box.

10. CycL vs. SQL

In this section, we will compare our implementation of this application with an alternative one. The alternative to *OpenCyc* is a normal, relational database, in which data is entered and queried using *SQL*. George Herson provided the following comparison between *CycL* and *SQL* formulation of assertions and questions, presented in sections *Data model in CycL* and *Asking OpenCyc questions*, respectively.

Assertions

English	CycL	SQL
Ordus' is a country.	(#\$isa #Ordus #\$Country)	CREATE TABLE country (countryid INT, name TEXT); INSERT INTO country VALUES (1,'Ordus')
Department is an organization.	(#\$genls #Department #\$Organization)	CycL only
Department of internal affairs is a department.	(#\$isa #\$DepartmentOfInternalAffairs #\$Department)	CREATE TABLE dept (deptid INT, name TEXT, branch TEXT, field_of_work TEXT); INSERT INTO dept (deptid, name) VALUES (1,'Department of Internal Affairs')
Department of ethical supervision is a department.	(#\$isa #\$DepartmentOfEthicalSupervision #\$Department)	INSERT INTO dept (deptid,name) VALUES (2,'Department of Ethical Supervision')
Field of work of department of internal affairs is protection of citizens against criminals.	(#\$fieldOfWork #\$DepartmentOfInternalAffairs "Protection of citizens against criminals")	UPDATE dept SET field_of_work='Protection of citizens against criminals' WHERE deptid=1
Field of work of department of ethical supervision is ensuring that the laws of Ordus' are followed by its subjects.	(#\$fieldOfWork #\$DepartmentOfEthicalSupervision "Ensuring that the laws of Ordus are followed by its subjects")	UPDATE dept SET field_of_work='Ensuring that the laws of Ordus are followed by its subjects' WHERE deptid=2
A civil servant is a person.	(#\$genls	CycL only

	#\$CivilServant #\$Person)	
Given names of Rededya Alimagomedov are <i>Rededya Peresvetovich</i> .	(#\$givenNames #\$RededyaAlimagomedov "Rededya Peresvetovich")	CREATE TABLE emp (empid INT, fname TEXT, mname TEXT, lname TEXT, deptid INT, boss INT NULL, field_of_work TEXT); INSERT INTO emp (empid,fname,mname) VALUES (1,'Rededya','Peresvetovich')
Family name of Rededya Alimagomedov is <i>Alimagomedov</i> .	(#\$familyName #\$RededyaAlimagomedov "Alimagomedov")	UPDATE emp SET lname='Alimagomedov' WHERE empid=1
Given name of Bagatur Lobo is <i>Bagatur</i> .	(#\$givenNames #\$BagaturLobo "Bagatur")	INSERT INTO emp (empid,fname) VALUES (2,'Bagatur')
Family name of Bagatur Lobo is <i>Lobo</i> .	(#\$familyName #\$BagaturLobo "Lobo")	UPDATE emp SET lname='Lobo' WHERE empid=2
Bagatur Lobo is subordinate of Rededya Alimagomedov.	(#\$superiors #\$RededyaAlimagomedov #\$BagaturLobo)	UPDATE emp SET boss=1 WHERE empid=2
Field of work of Bagatur Lobo is finding and catching criminals.	(#\$fieldOfWork #\$BagaturLobo "Finding and catching criminals")	UPDATE emp SET field_of_work='Finding and catching criminals' WHERE empid=2
Given names of Mokii Nilovich Rabinovich are <i>Mokii Nilovich</i> .	(#\$givenNames #\$MokiiNilovichRabinovich "Mokii Nilovich")	INSERT INTO emp (empid,fname,mname) VALUES (3,'Mokii','Nilovich')
Family name of Mokii Nilovich Rabinovich is <i>Rabinovich</i> .	(#\$familyName #\$MokiiNilovichRabinovich "Rabinovich")	UPDATE emp SET lname='Rabinovich' WHERE empid=3
Given names of Bogdan Rukhovich Ouintsev-Su are <i>Bogdan Rukhovich</i> .	(#\$givenNames #\$BogdanRukhovichOuintsevSu "Bogdan Rukhovich")	INSERT INTO emp (empid,fname,mname) VALUES (4,'Bogdan','Rukhovich')
Family name of Bogdan Rukhovich Ouintsev-Su is <i>Ouintsev-Su</i> .	(#\$familyName #\$BogdanRukhovichOuintsevSu "Ouintsev-Su")	UPDATE emp SET lname='Ouintsev-Su' WHERE empid=4
Field of work of Bogdan	(#\$fieldOfWork	UPDATE emp SET

Rukhovich Ouiantsev-Su is legal advice to Bagatur Lobo.	#\$BogdanRukhovichOuiantsevSu "Legal advice to Bagatur Lobo")	field_of_work='Legal advice to Bagatur Lobo' WHERE empid=4
Bogdan Rukhovich Ouiantsev-Su is subordinate of Mokii Nilovich Rabinovich.	(#\$superiors #\$MokiiNilovichRabinovich #\$BogdanRukhovichOuiantsevSu)	UPDATE emp SET boss=3 WHERE empid=4
Head of department of internal affairs is Rededya Alimagomedov.	(#\$headOfDepartment #\$DepartmentOfInternalAffairs #\$RededyaAlimagomedov)	UPDATE dept SET head=1 WHERE deptid=1
Head of department of ethical supervision is Mokii Nilovich Rabinovich.	(#\$headOfDepartment #\$DepartmentOfEthicalSupervision #\$MokiiNilovichRabinovich)	UPDATE dept SET head=3 WHERE deptid=2
Mokii Nilovich Rabinovich is a civil servant.	(#\$isa #\$MokiiNilovichRabinovich #\$CivilServant)	UPDATE emp SET deptid=2 WHERE empid=3; UPDATE dept SET branch='civil'

Asking Questions

English	Cycl	SQL
What countries are there?	(#\$isa ?COUNTRY #\$Country)	SELECT * FROM country
What departments are there?	(#\$isa ?DEPARTMENT #\$Department)	SELECT * FROM dept
What civil servants are there?	(#\$isa ?CIVIL_SERVANT #\$CivilServant)	SELECT e.* FROM emp e, dept d WHERE e.deptid=d.deptid AND d.branch='civil'
Tell me all departments and their heads.	(#\$headOfDepartment ?HEAD_OF_DEPARTMENT ?DEPARTMENT)	SELECT d.name, e.* FROM dept d, emp e WHERE d.head=e.empid
Who is Mokii Nilovich Rabinovich?	(#\$isa #\$MokiiNilovichRabinovich ?PROPERTY)	SELECT e.*,d.* FROM emp e, dept d WHERE e.deptid=d.deptid AND e.empid=3
Tell me the names of all civil servants with subordinates.	(#\$superiors ?BOSS ?SUBORD)	SELECT e1.fname, e1.mname, e1.lname FROM emp e1, emp e2, dept

		d WHERE e1.deptid=d.deptid AND d.branch='civil' AND e1.empid=e2.boss
Who is the boss of Bagatur Lobo?	(#\$superiors ?BOSS #\$BagaturLobo)	SELECT e2.fname, e2.mname, e2.lname FROM emp e1, emp e2 WHERE e1.empid=2 AND e1.boss=e2.empid

You can use this table to determine, which of the representations is better, easier to understand. In my and George Herson's opinion, the *CycL* version is more natural than *SQL* version.

11. Final words

This article has shown you how one could use OpenCyc as a replacement for a relational database. When thinking about use of OpenCyc in real-life applications, I asked me whether it is realistic to think that in future applications will use OpenCyc extensively. Use of OpenCyc imposes several constraints on the application, for instance, by its size. OpenCyc 0.7.0b, for instance, takes 126 MB on my hard drive. This is OK for a workstation, but we want to develop applications with OpenCyc for all platforms, from mainframes to handhelds. How can we use the advantages of OpenCyc and, at the same time, keep our applications small?

There are at least three potential answers to this question:

1. The application connects to OpenCyc via network. If OpenCyc is located on a server and the application has access to a network, it can be used even on "small" devices. Then, the size of OpenCyc does not matter.
2. If the application does not have a connection to a network, we can still use OpenCyc, if we create a program, let's call it *metaprogrammer*, which converts the things, which we normally do in OpenCyc to the language of the application.
3. We create a version of OpenCyc, which contains *only* those data, which are actually used by the application in question.

The latter two options are based on following assumption: *An application, which uses OpenCyc, always uses only a fraction of its knowledge.* You can compare OpenCyc with the brain, and the applications with programs in the brain, which are responsible for specific tasks (typing, moving the hands etc). Most scientists agree that an average human being uses only a small fraction of the brain resources.

The same is with OpenCyc. A particular application will probably use only very few predicates. Look at the application, which was described in this article. We have 7 questions (queries), i. e. the program can ask OpenCyc only these 7 questions, not more and not less.

We could

- delete from OpenCyc database everything, which is not needed for answering these questions or
- create an OpenCyc emulator, which can answer *only* these 7 questions.

In my opinion, this is a realistic way to use advantages of OpenCyc and have good performance.

I conclude this article with some questions, which might interest you.

- Is there a way to enter CycL data into the OpenCyc database using some convenient (human-readable) format, for instance, XML?
- Is there a way to view the CycL definitions of predicates and functions defined in an OpenCyc database?
- How can we visualize the structure of a CycL based database? What is the analogon to the entity-relationship diagram in OpenCyc world?
- Does it make sense to visualize the structure of CycL data using UML class diagrams?

It would be great if you shared your opinion on these subjects with other people, for instance, on the forum on <http://www.opencyc.org>.

12. Appendix: Related materials

Here is a list of materials related to this article.

- [hierarchy.zip](#) - Source code of the application described in this article
- [article.pdf](#) - PDF version of this article
- [2005_09_29_gherson_Ordus_Data_Model.pdf](#) - comparison of *CycL* version with *SQL* version of knowledge representation in nicely formatted PDF file
- http://dapissarenko.com/resources/2005_09_16_fpOrdus - second part of this article, which is devoted to improving readability of the code using functional programming techniques