

Oracle→OpenCyc Interface

11th December 2002

Copyright (C)

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Contents

1 Introduction	1
1.1 Overview	1
1.2 See also	1
2 Installation	2
2.1 Prerequisites	2
2.2 Getting the files	3
2.3 Installing it into Oracle	3
2.3.1 Create user cycptest	4
2.3.2 Load the jars into Oracle	4
2.3.3 Load CycJsproc.java	5
2.3.4 Load the CYC package	5
2.3.5 Small test	5
3 Usage	6
3.1 The first query	6
3.2 Oracle puts data in OpenCyc	7
3.3 Oracle gets data from OpenCyc	9
3.4 Type mapping	10
3.5 Method summary	10
3.6 Debugging	10
3.7 Exceptions	11

1 Introduction

This document contains information on how to install and use the Oracle→OpenCyc interface.

1.1 Overview

The two rectangles with dotted lines are the Oracle→OpenCyc interface.

1.2 See also

- [OpenCyc Api Documentation](#)

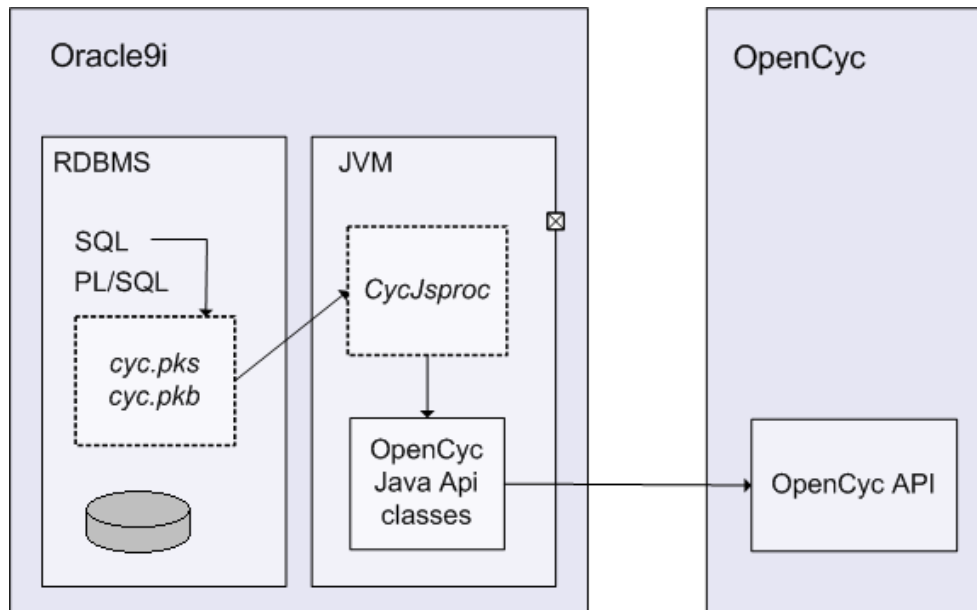


Figure 1: architecture

- [OpenCyc Java Api Documentation](#)
- [Oracle Java Developer's Guide](#)
- [Oracle Java Stored Procedures Developer's Guide](#) (especially part 3)
- [Oracle JDBC Developer's Guide and Reference](#) (part 10 and 18 →connecting to internal driver)

2 Installation

2.1 Prerequisites

You need the following installed on a linux server :

- [OpenCyc 0.7](#)
- [Oracle9i](#) (preferably release 2 for speed)

2.2 Getting the files

Get `ooi-0.7.tgz` from `http://217.117.225.187/~yeb/ooi-0.7.tgz`. This tarball contains the following files:

<code>install.sh</code>	A shell script to install the stuff into Oracle.
<code>oracle-opencyc.jar</code>	This is a modified version of the official <code>opencyc.jar</code> , which has a slightly modified <code>CycAccess.java</code> to remove references to the Fipa Agent classes, and doesn't contain unused (by Oracle) classes.
<code>CycJsprocs.java</code>	The Java Stored Procedures that wrap the methods in the OpenCyc Java Api to Oracle call and data types.
<code>cyc.pks</code>	The PL/SQL CYC package specification.
<code>cyc.pkb</code>	The PL/SQL CYC package body, contains call specifications for the Java Stored Procedures in <code>CycJsprocs.java</code> .

and some sql scripts that are used by the install script.

2.3 Installing it into Oracle

Untar `ooi-0.7.tgz` :

```
~$ tar zxvf ooi-0.7.tgz
ooi/
ooi/CycJsprocs.java
ooi/cyc.pkb
ooi/cyc.pks
ooi/test.sql
ooi/install.sh
ooi/oracle-opencyc.jar
ooi/resolve-cycaccess.sql
ooi/README
ooi/drop-create-cyctest.sql
ooi/license.txt
~$
```

Change into the directory and edit the install script:

```
~/ooi$ YOURFAVORITEEDITOR install.sh
```

Look for the line containing

```
## directory where opencyc was installed
OPENCYC_DIR=/home/httpd/opencyc-0.7.0
```

change it to the place where your opencyc copy was installed. Now run the install script:

```
~$ cd ooi
~/ooi$ ./install.sh
```

```
THE PASSWORD ASKED FOR IS THE SYSTEM PASSWORD
ITS NEEDED TO DROP AND CREATE USER cyctest
```

```
SQL*Plus: Release 9.2.0.1.0 - Production on Wo Dec 11 19:12:49 2002
```

```
Copyright (c) 1982, 2002, Oracle Corporation. All rights reserved.
```

Enter password:

the password you enter should be the password of the oracle SYSTEM user. Don't worry, it's only used to drop and create the new user called `cyctest`. The next few sections describe the important parts of the script, in case you want to do it manually.

2.3.1 Create user `cyctest`

The java classes of the OpenCyc Java API have to be loaded into the schema of a user. Though it's possible to separate the oracle user who's schema contains the jars from the users that actually use it, the examples in this guide put and use all the stuff into the schema of a single user named `cyctest`. The `install.sh` script doesn't use a TNS connect string, so if you want to install the stuff on a remote Oracle server you need to add the TNS identifier.

The user needs the grant `javauserpriv`.

2.3.2 Load the jars into Oracle

Next in the install script is loading some of the jars supplied in the `opencyc-0.7.0/lib` directory. Loading in Oracle 9i release 1 will take 15 to 30 minutes, in 9iR2 this is

done in max one minute. There should be no errors, but if there were, what might occur if you try this with other jars a while after I write this documentation, you can view the errors with

```
SQL> select * from user_errors
```

This will probably show that there were references to unresolved classes. Find these classes, load them, and try the resolve command again. You can view the status of all loaded java classes with the SQL command

```
SQL> SELECT dbms_java.longname(object_name) as name, status, created
FROM user_objects
WHERE object_type='JAVA CLASS'
```

If the status is `VALID` it means that the class is resolved and can be used (called) by the database. Status `INVALID` means that it hasn't been resolved (yet). *Please note:* The order of loading without resolving doesn't matter. But the order of resolving can be important, if not all necessary classes are loaded at before the first 'resolve' attempt. It can happen that errors disappear after dropping the user and loading all classes from scratch, though this happens very rarely.

2.3.3 Load CycJsproc.java

Once `CycAccess` is resolved, the 'Java Stored Procedure wrapper methods java source file' is loaded by the `install.sh` script. Note that this time a java source instead of class is loaded. Also, the class is now resolved at load time.

2.3.4 Load the CYC package

After `CycJsproc.java` is loaded and resolved, the CYC PL/SQL package is loaded.

2.3.5 Small test

When the CYC packages is loaded, a small test is done by executing `test.sql` which asks `opencyc` for its current time. The following output should be displayed:

```
SQL*Plus: Release 9.2.0.1.0 - Production on Wo Dec 11 19:24:10 2002
```

```
Copyright (c) 1982, 2002, Oracle Corporation. All rights reserved.
```

```
Connected to:
Oracle9i Enterprise Edition Release 9.2.0.1.0 - Production
With the Partitioning, OLAP and Oracle Data Mining options
JServer Release 9.2.0.1.0 - Production
```

```
make connection to opencyc
```

```
PL/SQL procedure successfully completed.
```

```
as for the time
```

```
CYCS_TIME
```

```
-----
(SecondFn 12 (MinuteFn 24 (HourFn 19 (DayFn 11 (MonthFn December (YearFn 2002)))
)))
```

```
end connection to opencyc
```

```
PL/SQL procedure successfully completed.
```

```
Disconnected from Oracle9i Enterprise Edition Release 9.2.0.1.0 - Production
With the Partitioning, OLAP and Oracle Data Mining options
JServer Release 9.2.0.1.0 - Production
```

3 Usage

At this point, the power of OpenCyc is at your Oracle fingertips. Every function in the CYC package can be used in every SQL query (issued from SQLPlus, or for example a PHP script in a web page), and in PL/SQL you can call every function or procedure.

3.1 The first query

All the following commands can be performed in sqlplus in the cyc test schema. Connect to the database, and in the database session, connect to cyc

```
SQL> begin cyc.makeconnection(); end;
2 /
```

Begin and end in SQL? Well, it's actually PL/SQL. Oracle allows you to give 'anonymous' PL/SQL blocks (a block starts with BEGIN and ends with END) where a SQL query could be executed. This is the way a PL/SQL procedure is called from an SQL frontend. Now to the first question *is Dog a collection?*

```
SQL> select cyc.isquerytrue( '($isa #$Dog #$Collection)',
    'InferencePSC') from dual
```

this produces the following output

```
CYC.ISQUERYTRUE('($ISA#$DOG#$COLLECTION)', 'INFERENCEPSC')
-----
1
```

For the type mapping between the different Cyc and Oracle types, see the source `CycJsproc.java` and `cyc.pkb`, or the summary below. Note that a query like this could also be put in e.g. a before trigger, and raise an exception if something is not true.

3.2 Oracle puts data in OpenCyc

Before OpenCyc can say anything interesting about your data, you have to put some information in your database into OpenCyc. This is easy. In the Oracle demo user SCOTT's schema is a table EMP. This table contains 14 employees, with the following names

```
SQL> select ename from scott.emp;
```

```
ENAME
-----
SMITH
ALLEN
WARD
JONES
MARTIN
BLAKE
CLARK
```

SCOTT
KING
TURNER
ADAMS
JAMES
FORD
MILLER

14 rows selected.

Assert that these people are Employees in OpenCyc's HumanSocialLifeMt

```
SQL> select cyc.assertgaf( '($isa #$$' || ename || ' #$$Employee)',  
 2 'HumanSocialLifeMt' )  
 3 from scott.emp;
```

```
CYC.ASSERTGAF('($ISA#$$' || ENAME || '#$EMPLOYEE)', 'HUMANSOCIALLIFEMT')
```

```
-----  
(#$$isa #$$SMITH #$$Employee)  
(#$$isa #$$ALLEN #$$Employee)  
(#$$isa #$$WARD #$$Employee)  
(#$$isa #$$JONES #$$Employee)  
(#$$isa #$$MARTIN #$$Employee)  
(#$$isa #$$BLAKE #$$Employee)  
(#$$isa #$$CLARK #$$Employee)  
(#$$isa #$$SCOTT #$$Employee)  
(#$$isa #$$KING #$$Employee)  
(#$$isa #$$TURNER #$$Employee)  
(#$$isa #$$ADAMS #$$Employee)  
(#$$isa #$$JAMES #$$Employee)  
(#$$isa #$$FORD #$$Employee)  
(#$$isa #$$MILLER #$$Employee)
```

14 rows selected.

GAF is short for 'Ground Atomic Formula'. Though `CYC.ASSERTGAF` is a function, and shouldn't change anything in the database, it's very handy that Oracle allows it to change things outside the database, in this case. It's very fundamental in Oracle that functions do not change the database state. And I think if you know a bit

PL/SQL, calling `cyc.assertGaf` from a procedure in PL/SQL LOOP might be bit more proper. `Assertgaf` returns it's input, because functions are supposed to return something.

3.3 Oracle gets data from OpenCyc

Now OpenCYC knows about the employees, we can ask stuff... *Is Scott a person?*

```
SQL> SELECT cyc.isquerytrue(
  2  '($isa $$SCOTT $Employee)', 'InferencePSC'
  3  ) AS isaemp FROM DUAL
  4  /
```

```
CYC.ISQUERYTRUE('($ISA$$SCOTT$EMPLOYEE)', 'INFERENCEPSC')
```

1

Who are all the employees?

```
SQL> select column_value from
  2  the( select cyc.askwithvariable(
  3         '($isa ?X $Employee)',
  4         '?X',
  5         'InferencePSC' )
  6        from dual );
```

```
COLUMN_VALUE
```

SMITH
ALLEN
WARD
JONES
MARTIN
BLAKE
CLARK
SCOTT
KING
TURNER
ADAMS

JAMES
FORD
MILLER

14 rows selected.

Because this is a REAL oracle resultset, it can be used in all wild Oracle SQL constructs. Union, order, group by etc etc. Don't forget to end the connection at the end of the Oracle session

```
SQL> begin cyc.endconnection(); end;
```

3.4 Type mapping

OpenCyc	Java	Oracle
-	java.lang.?	DATE
-	boolean	NUMBER in [0,1]

In this case, 1 means true. (yes, Oracle SQL doesn't know booleans. PL/SQL does however.)

3.5 Method summary

At this point, you really must check the CycJsproc source. There are a LOT of methods in the OpenCyc CycAccess.java that still need to be written. Basic stuff works, more is to follow..

To get all hypothesized constants.

```
select column_value from the  
(select cyc.converselist( '(constant-complete "HYP-")' ) from dual)
```

```
select cyc.converselist( '(cyc-kill #$$' || column_value || ')' ) from the  
(select cyc.converselist( '(constant-complete "HYP-")' ) from dual)
```

3.6 Debugging

In the beginning of the CycJsprocs.java source you'll find the method `makeConnection()`. At the end of this method is the call to `CycAccess.traceOn`. The trace of CycAccess is default off, so if you comment this call Standard output (`System.out.println`) is

dumped by oracle in trace files in the directory `$ORACLE_BASE/admin/<instancename>/udump`. Find the last trace file with `ls -l --sort=time -r` and then monitor the contents with `less` or `tail -f`.

3.7 Exceptions

All java exceptions are caught and thrown to the caller. In Oracle, all java exceptions appear as ORA-29532 errors. At the end of the text of the error message you should see the java error. So, if there is an error in your CYC query, look good at the ORA-29532 errors!