Manual For SNAPPI-DB - Structures, iNterfaces and Alignments for Protein-Protein Interactions - DataBase

Emily Jefferson

July 13, 2006

1 Outline

This manual describes how to create and use SNAPPI-DB, a database of interacting domains observed in structural data, and its Java 5 API. It describes SNAPPI from a implementational point of view but provides a higher level than the API document giving an overview of how to actually use the code and database without having to read all of the source code!

In summary, the structural data for SNAPPI-DB is obtained from the MSD data warehouse as the MSD provides consistant data with links to many types of data about proteins and nucleic acids. In order to increase performance and allow complex analysis with a high degree of abstraction, the relevant (and most frequently queried) data from the MSD is migrated to an object-oriented database developed using the Java Data Objects Technology (JDO). Domain-domain interactions are then determined based on atom-atom distances and the interactions indexed. The domain-domain interactions are then classified by family pair and interaction interface and multiple alignments are then generated for each group

The database can either be directly downloaded from the web and used straight away or if you have a copy of the MSD then you could choose to create the db from scratch. Creating the db from scratch will take at least one week to set up and run if you are doing for the first time. I will describe how to do both methods.

2 Precursors to using the system

There are several things that you need to have or need to install before you can begin. These have been split into fundermental requirements (these are required before you even start to do anything) and possible requirements (these are required for some stages of the building or running of the database and may not be required depending on your needs).

2.1 Fundermentals Requirements

Insure that you enough disk space - 60GB of available space for permanent use and another 40GB for extra temporary files generated if generating the system from scratch. The database itself takes up about 40GB of space. However, there are also some other files which need to be generated to have a working system. These include the orientation information generated by STAMP to classify domains by interface. Therefore, it is recommended that you have available 100GB of space. All of this space is not required once the system is up and working as some of the results are temporary and once the results have been included in the database they can be deleted. When the temporary files can be deleted is described below. **If you are not generating the system from scratch then you will only require about 50GB for the Database and associated files that come with the SNAPPI package**.

Install Java 5 or later - The code is written using Java 5 language and so a version of Java 5 is required to run this system. Java 5 can be down loaded at

`http://java.sun.com/j2se/1.5.0/download.jsp`

Install Fast Objects - In order to create the JDO object orientated database from the MSD database (as summarised above and described in depth in the paper) Fast Objects first needs to be installed. If you want to use another implementation of JDO see the section below. The Fast Objects web site has all the information about installing the program

`http://community.fastobjects.com/default.htm`

You just need to get an academic licence (community edition) and then run the installer. From there all the other setup things are done automatically.

Install Ant - Ant is a platform independent method for making programs. Apache Ant is a Java-based build tool. In theory, it is kind of like Make, but without Make's wrinkles. Instead of a model where it is extended with shell-based commands, Ant is extended using Java classes. Instead of writing shell commands, the configuration files are XML-based, calling out a target tree where various tasks get executed. Each task is run by an object that implements a particular Task interface. Ant is used to compile the code for the system. The compilation has to be done using Ant to make the classes enhanced (turned into code which will fit the JDO interface) so that it will work correctly with the database. Ant can be found at

`http://ant.apache.org/`

Many java editors already come with Ant in built. If you are using such an editor then you do not have to worry about installing Ant separately.

2.2 Possible requirements

Install Blast - Blast was used for some the analysis that I have performed with the database but is not directly required to create the database. However, I have left the Blast calling code in the API for general usage. If you think that you will want to use Blast then install it and my code will link up to it as required. Blast can be downloaded from

`ftp://ftp.ncbi.nih.gov/blast/`

Connection to the MSD Database - A connection to the MSD database is required if you want to either create the database from scratch or link directly back to the MSD using the indexing that is already included in the database. For example if you want some piece of information which is in the MSD but not stored in SNAPPI-DB then the unique id

numbering for most objects is the same as in the MSD and so you can easily link back directly to the MSD from SNAPPI-DB. In order to connect to the MSD from the SNAPPI code all you need to do is put the connection setting in the properties.txt file and it will automatically set up the connection. This is described in the next section.

3 System Configuration

Where each of the programs is installed and were you want to put each of your output files is system dependent therefore these properties need to be set by the user. There should be no hard coded paths in the API.

Each of these properties are set in the properties.txt file. A description of each property is shown below. The properties.txt file contained within the SNAPPI package contains examples of the choices.

- STAMPDIR = this is where you want to put the stamp program
- CLUSTERDIR = this is where you want to run cluster jobs
- ORIENTDIR = this is where you want to store results files of domain interactions classified by orientation
- DATADIR = this is the directory you want to put the data from any analysis you have performed on the database.
- RESSUBDIR = this is the directory where you want to store the Van der Waals radii of each atom. This information is used to determine whether two atoms are interacting so that domain-domain interactions can be determined. A file called VanDerWals.txt is included in the SNAPPI package. These are the default Van der Waals radii used by the system. If you want to use your own then just provide your Van der Waals radii in the same format as the VanDerWals.txt file.
- BLASTDIR = this is the directory which contains the Blast program.

- PDBFILES = sometimes it is beneficial to write out pdb files for a particular Entry, Assembly, Chain or Domain to pass to another program for analysis or calculation. This is the directory where you might want to store all the generated PDB files.
- MSDDRIVER = this is the oracle driver for the MSD e.g. oracle.jdbc.driver.OracleDriver
- MSDURL = this is the jdbc driver e.g. jdbc:oracle:thin:@hornet:1521:MSDSD
- MSDUSER = this is the user name to access the msd e.g fredBloggs
- MSDPASSWD = this is the password to access the msd e.g.a1b2c3
- javax.jdo.PersistenceManagerFactoryClass=
  com.poet.jdo.PersistenceManagerFactories
  (this needs to be set exactly as written here for JDO to work)
- javax.jdo.option.ConnectionURL = this is where the database is stored
  e.g. fastobjects://LOCAL//emily/DATABASE.j1

Another file which needs to be altered depending on your environment is a bash script called RJA which is included in the SNAPPI package. This script is a useful script which calls SNAPPI commands with all of the correct libary paths and java options. All of the paths to libraries need to be changed to fit your system. The RJA script was written for running SNAPPI via the command line. Personally I prefer to use IDEs where I include all of the library paths and runtime settings etc. However, if you prefer the command line option or what to run SNAPPI remotely or using a cluster then this is a useful script. All of the commands below can be run by simply

```
RJA snappi command
e.g. #RJA SNAPPI.IO.ClearLogs
```

You will need to change the paths hard coded in this script.

4 Using the Database

The package SNAPPI.ExampleCode contains several examples of how the code should be used. Hopefully if you read through these examples and the documentation it should be relatively to start using the system straight away. Although I have loads of analysis code which has been developed to investigate a particular aspect of the data (such as the difference between different domain classification systems and the difference between domain-domain interactions observed in PQS biological units in comparison to the PDB asymmetric unit) I have not included this in the SNAPPI-Package as I felt that it may lead to confusion. If you would like and more examples, if anything is unclear or you find a bug please do not hesitate to contact me.

5 Generating the System from Scratch

You do not need to read this section if you are using the prebuilt SNAPPI-DB! The main advantage of rebuilding SNAPPI-DB is that you an over-ride any aspect of the generation of the database e.g. change threshold of the number of residues which need to be interacting for a pair of domains to be considered to be interacting. The generation of the system requires 4 main steps: generation of the database structure, populating the database with the data from the MSD, creation of the domain-domain interactions and classification of the interacting domains by interface.

5.1 Generation of the Database Structure

I would recommend that you read the summary documentation which comes with FastObjects to understand the basics of JDO and persistence of objects. In addition I would recommend that you read the paper describing SNAPPI-DB (on its way) as this summarises JDO and how it works. However if you do not have time to do this then just follow the steps and it should all work without you needing to fully understand what is going on.

All of the generation of the database is done using a writable version of the database. Therefore sometimes the database becomes dirty (there are objects which have not been fully

commited when the program was stopped). These objects are stored in the log file of the database. When the log file is present and contains uncommited objects you may get an error when trying to run a command that returns "'Database dirty"'. In this case run the following class

```
SNAPPI.IO.ClearLogs
```

This will clear all of the objects in the Log file and delete the Log files. This command may need to be run between some of the commands below.

Ant Task Use the "make" ant task to build the code present in the SNAPPI directory. Do not compile the code by the javac command as this does not enhance the classes (required for JDO to work).

Create JDO database instance To create the database from scratch go to the place where you want to store the database and run the following command (using the full path of the ptj command in FastObjects/bin)

```
#ptj -ep classes -register -xc -database DATABASE.j1
```

This creates a database called DATABASE.j1. This is a command specific to the FastObjects JDO version.

Populate JDO database structure To make all the objects which you want to store persistable and create the structure of these objects in the new database the following classes main function must be then called

```
PIP.IO.CreateJDODatabase.PersistDatabase
```

Populate JDO database with peripheral data To store all the possible atom and residue names in the database - to reduce the storage of these properties in each atom/residue the following classes main function must be called

```
PIP.IO.CreateJDODatabase.PersistPeripheral
```

5.2 Populating SNAPPI-DB with data from the MSD

To populate the database with the PDB Entry and Domain Information from the MSD the following classes main function must be then called

```
PIP.IO.CreateJDODatabase.PersistEntries
```

This will take a long time to run (approx 3 days depending on network infrastructure) as it connects to the MSD with one connection and then adds each generated object to the new JDO Database. This could be done in parallel with a different version of FastObjects (one which costs) but the free community edition only allows serial witting. Once the database has been created, the database is mostly accessed in a read only fashion and so usually parallel access is not a problem.

5.3 Creation and Persistance of the Domain-Domain Interactions

To determine all domain-domain interactions, store them in each PDB Entry and then sort and store both domains and domain interactions by domain family and family pair, respectively, run the following classes main function

```
PIP.IO.CreateJDODatabase.PersistDomainInteractions
```

This will take a approx 10 hours to run (depending on hardware) as it works out the distance between all interacting atoms at the surface between domains in order to determine if they are interacting for 3 different domain definitions. This distance based method for determining interactions can easily be overwritten if require e.g. to use solvent accessible area instead. Once the command has been run each Assembly will have a list of domain-domain interactions which are observed in that Assembly. In addition to this the Domains and DomainInteractions classes have been populated in the database. The Domains class contains domains associated with their domain family i.e. $Map < Family, Collection < Domain >>$. The DomainInteractions class contain domain interactions associated with their domain family pair i.e. $Map < Pair < Family >, Collection < DomainInteraction >>$.

5.4 Classification of the Domain-Domain Interactions by Interface

The classification of domain-domain interactions by interface is done by calculating the iRMSD between two domain-domain interactions (see SNAPPI-DB paper on its way). To determine the iRMSD structural alignments between domains need to be performed using STAMP. This step is time consumming and so has been parallelised. The process consists of two steps. Firstly, the running of STAMP and storage of results into file format is performed in a parallel fashion using a read only version of the database. Then the results from the files are read into the database using a writable form of the database in serial fashion. The generating of the results into file format can be done using a cluster if required.

Generating Results In File Format This class is designed to classify domain interactions for each family pair. There are approximately 3000 SCOP family pairs and so the following classes main function has to be called with start ranging from 0 to 2999. This can easily be performed on a cluster.

```
PIP.IO.CreatePeripheralData.GenerateOrientationFiles start
```

Persisting domain interactions by orientation within SNAPPI-DB To persist the classification of domain-domain interactions by interface call the following classes main function

```
PIP.IO.CreateJDODatabase.PersistOritentationInteractions
```

This class reads in all of the results from the files generated in the previous step and stores the results in the database.

The database should now be completely created from scratch and you should be able to go off and play!