# Mychem Documentation

***Release 1.0.0***

**Jérôme Pansanel**

**Nov 29, 2019**

# CONTENTS

CHAPTER

ONE

# INTRODUCTION

## 1.1 Mychem

Mychem is a chemoinformatics extension for MySQL and MariaDB. This extension provides a set of functions that handles chemical data stored in a database (i.e. file format conversion, chemical properties computation, ...). These functions are provided through the UDF mechanism, making them usable like native MySQL functions.

Mychem is based on Open Babel (version >= 2.3.2). As Open Babel is a well-known software in chemoinformatics, Mychem proposes reliable and fast functions. These functions permit you to search, analyze and convert chemical data.

More informations about Mychem are available on the Mychem website.

## 1.2 MySQL and MariaDB

MySQL is one of the most popular Open Source SQL database servers. It is a very fast, multi-threaded multi-user and robust application. MySQL Server is intended for mission-critical, heavy-load production systems as well as for embedding into mass-deployed software. Further informations about MySQL can be founded on the MySQL website. MariaDB is a fork of MySQL.

## 1.3 Open Babel

Open Babel is a chemical toolbox designed to speak the many languages of chemical data. It is an open, collaborative project allowing anyone to search, convert, analyze, or store data from molecular modeling, chemistry, solid-state materials, biochemistry, or related areas. You can find more informations about Open Babel on the Open Babel website.

CHAPTER

TWO

# INSTALLATION

In this chapter, we will discuss the steps necessary to compile and install Mychem.

- *How to Obtain Mychem* - mainly concentrates on downloading the last stable version of Mychem.
- *Requirements* - lists the programs and libraries which you need installed to successfully compile Mychem.
- *Compilation and Installation* - leads you through all the steps of compilation and installation of the application.
- *Mychem API* - tells how to build the API documentation of Mychem.

## 2.1 How to Obtain Mychem

At this time, Mychem is only available as a source package that you need to compile. You can also get the last snapshot directly from the GIT repository (take care, this version can be unstable). It is planned to release binaries in a future release of Mychem.

### 2.1.1 Source Package

The last source package can be found on the Project Homepage hosted by SourceForge.

### 2.1.2 GIT Repository

The last development version of Mychem can be obtained by cloning the GIT repository. In case you are using a command line interface, follow this step:

```
$ git clone https://github.com/mychem/mychem-code.git
```

## 2.2 Requirements

In order to successfully compile and install Mychem, you need the following programs and libraries.

- C/C++ compiler

  The compilation has been tested successfully with the GNU C Compiler (GCC).

- CMake version 2.6.0 or higher.

  CMake is a multi-platform Makefile generator. It is a Free Software and can be downloaded on the CMake website.

- MySQL version 4.0 or higher, or MariaDB.

  The MySQL database server is available from the MySQL website. The MySQL headers are also required for the compilation. They are included with the standard MS Windows installation of MySQL. For Linux, an additional package has to be installed (libmysqlclient-dev).

- Open Babel version 2.3.2 or any later version. The compilation of Mychem requires also the openbabel header. For MS Windows, they are provided with the source package. For Linux, install the libopenbabel-dev package.

  The software is available from the Open Babel Home Page.

## 2.3 Compilation and Installation

This section describes the compilation of Mychem's source files for GNU/Linux, Mac OS X and Microsoft Windows.

### 2.3.1 Compiling and Installing Mychem on GNU/Linux

This section contains the compilation and installation instructions for Mychem on GNU/Linux.

#### Standard Installation on GNU/Linux

This section describes the standard way to compile and install Mychem on GNU/Linux. You have to check that the header files for the MySQL library and the Open Babel library are installed on your workstation. If they are not installed, the CMake software will raise an error when generating the Makefile. First, extract the appropriate source package. If you are using a command line interface, follow this instructions:

- For the tar gzipped archive:

```
$ tar -xfzv mychem-1.0.0.tar.gz
```

- For the zip archive:

```
$ unzip mychem-1.0.0.zip
```

CMake can build the libraries and executables into any directory. If the directory contains the source, the build is called *in source*. In other cases, it is called *out of source*. CMake strongly recommends and promotes building out of source.

- In source build:

```
$ cd mychem-1.0.0
$ cmake .
$ make
$ sudo make install
```

- Build out-of-source (recommended):

```
$ cd mychem-1.0.0
$ mkdir build
$ cd build
$ cmake ..
$ make
$ sudo make install
$ cd ..
```

**Note**

The default installation directory can be retrieven with the following command:

```
$ mysql_config --plugindir
/usr/lib/x86_64-linux-gnu/mariadb18/plugin
```

Once the library is installed, the SQL functions are created with the following command:

```
$ mysql -u user -p < src/mychemdb.sql
```

**Note**

On Mac OS X and Windows, another SQL file is used instead of the `src/mychemdb.sql` file. The name of this file is detailed in the corresponding OS section.

### Customized Installation

You can customized the build and installation process by modifying CMake arguments. For example, if you want to change the path of the installation directory:

```
$ cd /path/to/mychem/build
$ cmake -DCMAKE_INSTALL_PREFIX=/convenient/path ..
```

If you want to have more details about the compilation process, use the following option for the `make` command:

```
$ make VERBOSE=1
```

### Ubuntu specifities

AppArmor is a Linux Security Module and is installed by default on Ubuntu. It permits to confine individual programs to a set of listed files. To configure correctly AppArmor for Mychem, please follow the instructions detailed in the *AppArmor* section.

### Testing the installation

Since v0.5, Mychem includes a test suite. To build and use these programs, you have to set the MySQL connection settings and run the tests. The following example builds Mychem out-of-source and enables testing.

```
$ cd mychem-1.0.0
$ mkdir build
$ cd build
$ cmake -DMY_HOST=localhost -DMY_USER=user -DMY_PASSWD=passwd ..
$ make
$ make install
$ cd ..
```

When running the command:

```
$ cmake -DMY_HOST=localhost -DMY_USER=user -DMY_PASSWD=passwd ..
```

You will see the following line in the status message:

```
-- Test module enabled
```

The program use the *mysql* database for testing. If the *user* do not have access to this database, it is possible to set the name of the database by using the -DMY_DB option.

**Note**

If the user can access MySQL without a password, then you do not need to set the `MY_PASSWD` parameter. The two other parameters ( `MY_HOST` and `MY_USER`) are mandatory.

**Note**

The password is not stored in a safe location. If you are doing tests on a critical server, please use directly the test executables and do not use the CMake facility to perform the tests ! You can find the test executables in the `/path/to/mychem/build/tests` directory.

To run the tests, use the command `make test`. You should see the following results:

```
Running tests...
Test project mychem-code/build
    Start 1: ConversionTest
1/5 Test #1: ConversionTest ...................   Passed    0.05 sec
    Start 2: HelperTest
2/5 Test #2: HelperTest .......................   Passed    0.00 sec
    Start 3: ModificationTest
3/5 Test #3: ModificationTest .................   Passed    0.01 sec
    Start 4: MolmatchTest
4/5 Test #4: MolmatchTest .....................   Passed    0.04 sec
    Start 5: PropertyTest
5/5 Test #5: PropertyTest .....................   Passed    0.03 sec

100% tests passed, 0 tests failed out of 5

Total Test time (real) =   0.14 sec
```

The `LastTest.log` file contains more details about the test results. It can be found in the `/path/to/mychem/build/Testing/Temporary` directory.

#### Installation Troubleshooting

Building your application can raise some errors:

- If CMake returns the following error:

```
CMake Error: This project requires some variables
to be set, and cmake can not find them. Please set the following
variables: OPENBABEL2_INCLUDE_DIR (ADVANCED) OPENBABEL2_LIBRARIES
(ADVANCED)
```

  It means that CMake did not find Open Babel. If you know where Open Babel is installed on your system, you can tell it to CMake with:

```
$ cd /path/to/mychem-1.0.0/build
$ cmake -DOPENBABEL2_INCLUDE_DIR=/path/to/openbabel/include \
-DOPENBABEL2_LIBRARIES=/path/to/library ..
```

### 2.3.2 Installing Mychem on Mac OS X

The installation on Mac OS X differs slightly from an installation on GNU/linux. First, you have to set some parameters for CMake. The following list shows common values used when compiling Mychem on Mac OS X. Note that

these values may differ on your system.

| CMake Variable | Value |
|---|---|
| OPENBABEL2_INCLUDE_DIR | /usr/local/include/openbabel-2.0 |
| OPENBABEL2_LIBRARIES | /usr/local/lib/libopenbabel.dylib |
| MYSQL_INCLUDE_DIR | /Library/MySQL/include/mysql |
| | /Applications/MAMP/Library/include/mysql |
| MYSQL_LIBRARIES | /Library/MySQL/lib/mysql/libmysqlclient.dylib |
| | /Applications/MAMP/Library/lib/mysql/libmysqlclient.dylib |

Once the library is installed, the SQL functions are created with the following command:

```
$ mysql -u user -p < src/mychemdb_macosx.sql
```

When executing the previous command, you can have the following problem:

```
$ mysql -u root -p < src/mychemdb_macosx.sql
Enter password:
  ERROR 1126 (HY000) at line 10: Cannot open shared library 'libmychem.dylib'
  (errno: 2 dlopen(/usr/local/mysql/lib/plugin/libmychem.dylib, 2): Library
  not loaded: libmysqlclient.18.dylib
  Referenced from: /usr/local/m)
```

This problem can be fixed by modifying the libmychem.dylib shared library so that all dependent libraries contain the correct path information:

```
$ otool -L libmychem.dylib
libmychem.dylib:
        /usr/local/lib/libmychem.0.dylib (compatibility version 0.0.0, current
        version 1.0.0)
        /usr/local/lib/libopenbabel.4.dylib (compatibility version 4.0.0,
        current version 4.0.1)
        libmysqlclient.18.dylib (compatibility version 18.0.0, current version
        18.0.0)
        /usr/lib/libSystem.B.dylib (compatibility version 1.0.0, current
        version 159.1.0)
        /usr/lib/libstdc++.6.dylib (compatibility version 7.0.0, current
        version 52.0.0)
$ sudo find / -name 'libmysqlclient.18.dylib' -print
Password:
/usr/local/mysql-5.5.24-osx10.6-x86_64/lib/libmysqlclient.18.dylib
$ sudo install_name_tool -change libmysqlclient.18.dylib \
 /usr/local/mysql-5.5.24-osx10.6-x86_64/lib/libmysqlclient.18.dylib \
 libmychem.dylib
```

### 2.3.3 Installing Mychem on Microsoft Windows

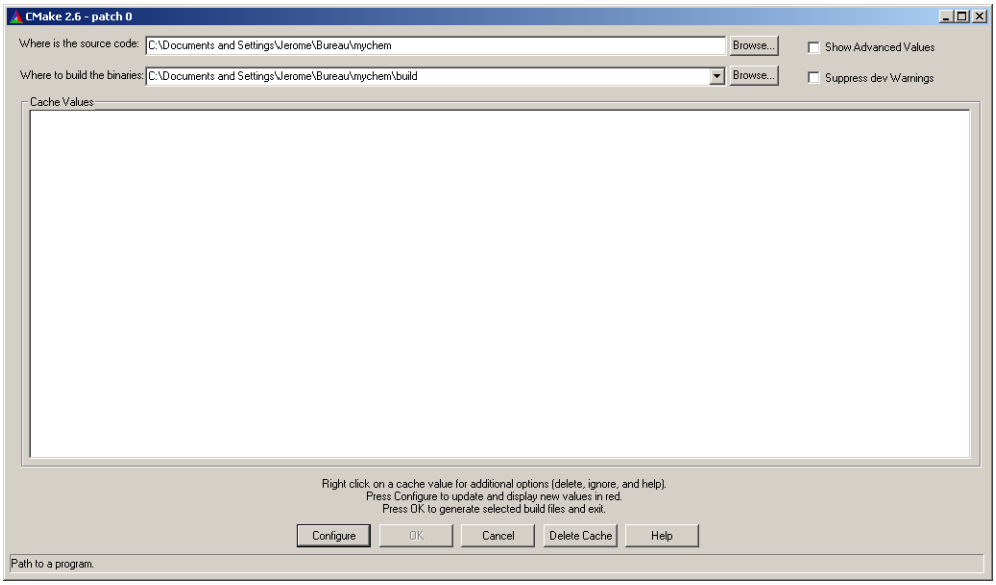
This section describes contains the installation of Mychem on Microsoft Windows.

### Installation using Microsoft Visual Studio Express 2005

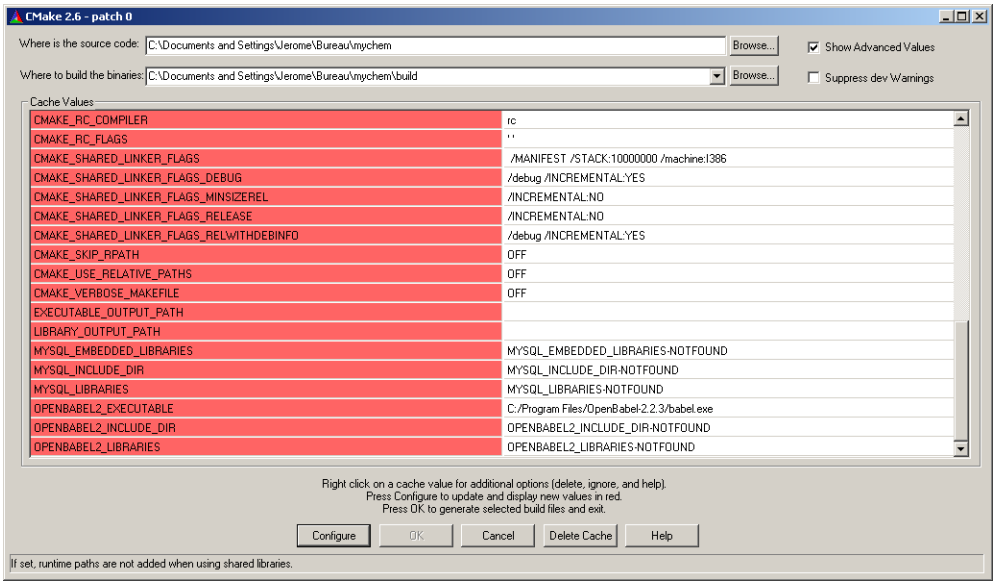
You can compile Mychem with Microsoft Visual C++ 2005 Express Edition ( MSVC++ ). This software can be downloaded from the Microsoft MSDN Website. To complete the Microsoft Visual C++ software, install the SDK Platform. The instruction are given in the following article. The MySQL package for Windows contains all required libraries and include files for building Mychem. However, Open Babel does not provide such a package. You have to compile Open Babel. It can easily be done with MSVC++ and by following the instructions detailed on the Open Babel Website. The following libraries are necessary for the build of Open Babel. They can be downloaded from the Maintainer's Home Page of the libxml2 Windows port.

- iconv
- libxml2
- zlib

Once your compilation environment is ready, you can generate the MSVC++ project file with CMake. Launch the CMake GUI and set up the source code directory and the build directory for the binaries.



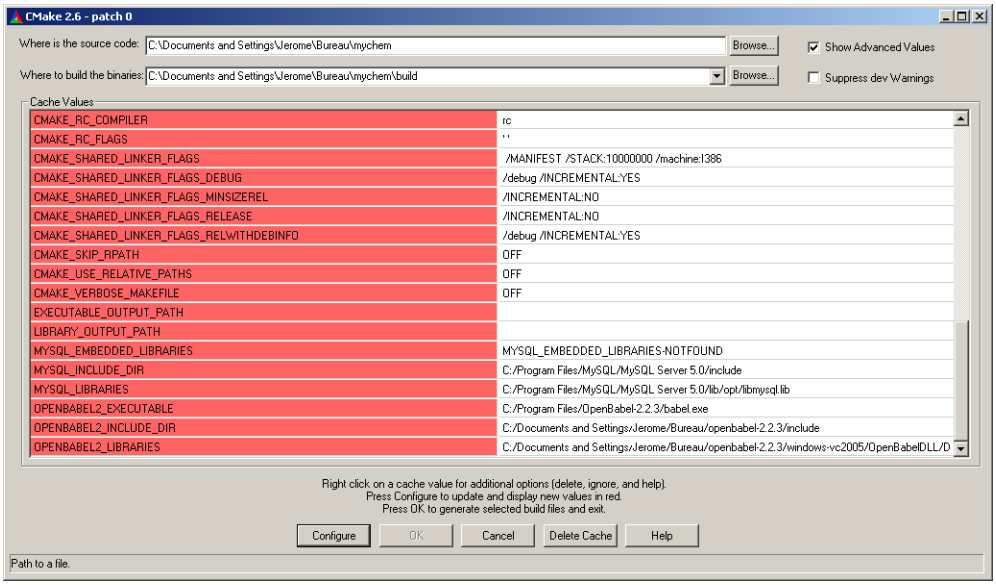

Then, click on Configure. A window will appear to let you select what build system you want CMake to generate files for. Choose Visual Studio 8 2005 and click on Ok. After some processing, CMake will raise an error window, telling you that some variables are not founded. You will have to set it manually. Press twice the Ok button. Click on the Show Advanced Values checkbox to display the full list of parameters. Find the lines with OPENBABEL2_INCLUDE_DIR, OPENBABEL2_LIBRARIES, MYSQL_INCUDE_DIR, MYSQL_LIBRARIES values. These lines are at the end of the list.




You have to set OPENBABEL2_INCLUDE_DIR to the directory `include` contained in the Open Babel source directory (i.e., `C:/path/to/openbabel/include`) and OPENBABEL2_LIBRARIES to the file named `OpenBabelDLLD.lib` (the library should be located in the `C:/path/to/openbabel/windows-vc2005/OpenBabelDLL/Debug` directory).

The MYSQL_INCLUDE_DIR and MYSQL_LIBRARIES parameters should be respectively set to `C:\Program Files\MySQL\MySQL Server X.Y\include` and `C:\Program Files\MySQL\MySQL Server X.Y\opt\libmysql.lib`, where X.Y is the version of MySQL.

Once the values are set, click on Configure and then on Ok. CMake generates the MSVC++ project file (`mychem.sln`) and exit.




Once the project file is generated, open it with MSVC++ . The `mychem.sln` project file should be located at `C:/path/to/mychem/build/`. Several modules are available for building, however, only *mychem-lib* is required:

- mychem-lib
- conversion_test

- helper_test
- modification_test
- molmatch_test
- property_test

If you build the *debug* version of Mychem, the Mychem DLL file is located in the `C:/path/to/mychem/build/dir/src/debug` directory and is named `mychem.dll`. You have to copy this DLL file into the MySQL `bin` directory. Then copy the `C:/path/to/openbabel/windows-vc2005/OpenBabelDLL/Debug/OpenBabelDLLD.dll` DLL file and all DLL files founded into the `openbabel-2.400/windows-vc2005` directory to the MySQL `bin` directory.

At last, restart MySQL and run the win32 SQL script `/path/to/mychem/src/mychemdb_win32.sql`.

## 2.4 Mychem API

API documentation is available on the Mychem website.

You can also build the documentation yourself, by using the Doxygen software. To generate this documentation, use the following commands:

```
$ cd /path/to/mychem-1.0.0
$ mkdir api
$ doxygen Doxyfile
```

The API documentation can be read using a web browser at the following url: `file:///path/to/mychem-1.0.0/api/html/index.html`

CHAPTER

THREE

# USING MYCHEM

This chapter provides a short tutorial about the usage of Mychem. It will present you a simple way to create a chemical database with Mychem and how to use some functions. More details about each function used in this tutorial are available in the *Command Reference*.

## 3.1 The Database

A chemical database is composed of one or several tables. The examples presented in this chapter are using a set of four tables, with the following structure:

- *compounds* - a table containing an unique id for each molecule and its name.

```
+---------+-----------------+------+-----+---------------------+----------------+
| Field   | Type            | Null | Key | Default             | Extra          |
+---------+-----------------+------+-----+---------------------+----------------+
| id      | int(11) unsigned | NO   | PRI | NULL                | auto_increment |
| name    | varchar(255)    | NO   | MUL | NULL                |                |
| created | timestamp       | NO   |     | 0000-00-00 00:00:00 |                |
+---------+-----------------+------+-----+---------------------+----------------+
```

- *1D_structures* - a table containing an unique reference to the *compounds* table and several types of 1D molecular descriptors (InChI code and SMILES).

```
+-------------+-----------------+------+-----+---------+-------+
| Field       | Type            | Null | Key | Default | Extra |
+-------------+-----------------+------+-----+---------+-------+
| compound_id | int(11) unsigned | NO   | PRI | NULL    |       |
| inchi       | text            | NO   |     | NULL    |       |
| smiles      | text            | NO   |     | NULL    |       |
+-------------+-----------------+------+-----+---------+-------+
```

- *3D_structures* - a table containing an unique reference to the *compounds* table and the 3D structure in MDL Molfile format.

```
+-------------+-----------------+------+-----+---------+-------+
| Field       | Type            | Null | Key | Default | Extra |
+-------------+-----------------+------+-----+---------+-------+
| compound_id | int(11) unsigned | NO   | PRI | NULL    |       |
| molfile     | text            | NO   |     | NULL    |       |
+-------------+-----------------+------+-----+---------+-------+
```

- *bin_structures* - a table containing an unique reference to the *compounds* table and several types of binary descriptors (fingerprints and serialized OBMol object).

```
+--------------+-----------------+------+-----+---------+-------+
| Field        | Type            | Null | Key | Default | Extra |
+--------------+-----------------+------+-----+---------+-------+
| compound_id  | int(11) unsigned | NO   | PRI | NULL    |       |
| fp2          | blob            | YES  |     | NULL    |       |
| obserialized | blob            | YES  |     | NULL    |       |
+--------------+-----------------+------+-----+---------+-------+
```

Such tables can be easily created and populated using mychemdb_manager, a Python script released with Mychem. It can be found in the `scripts` directory. Its usage is detailed in the *Chemical Database Manager* chapter.

## 3.2 Examples

Once the database and the tables are created, you can use many chemical functions. Here is a (very) short overview of the possibilities. The dataset used for this example can be freely obtained from the Chemical Structures Project.

### 3.2.1 Calculate the Molecular Weight

The computation of the molecular weight of a molecule is performed by the `MOLWEIGHT` function. In the following example, the molecular weight of the amino acid *glycine* is calculated.

```
mysql> SELECT MOLWEIGHT(`3D_structures`.`molfile`)
    -> FROM `compounds`,`3D_structures`
    -> WHERE `compounds`.`name`='glycine'
    -> AND `compounds`.`id`=`3D_structures`.`compound_id`;
        -> 75.066600
```

### 3.2.2 Search a Substructure

The `MATCH_SUBSTRUCTURE` function permits to find a substructure and belong to the *Molmatch Commands* function group. In this example, we are looking for compounds containing a phenol group:

```
mysql> SELECT `compounds`.`name`
    -> FROM `compounds`,`bin_structures`
    -> WHERE `compounds`.`id`=`bin_structures`.`compound_id`
    -> AND MATCH_SUBSTRUCT('[OH]c1ccccc1',`obserialized`);
    -> LIMIT 10;
+----------------------------+
| name                       |
+----------------------------+
| 2,3,4,5,6-Pentachlorophenol |
| 2,3-Dimethylphenol         |
| 2,4,6-Trichlorophenol      |
| 2,4-Dimethylphenol         |
| 2,5-Dimethylphenol         |
| 2,6-Dimethylphenol         |
| 2-Bromophenol              |
| 2-Chloro-5-methylphenol    |
| 2-Chlorophenol             |
| 2-Hydroxybenzaldehyde      |
+----------------------------+
10 rows in set (0.02 sec)
```

CHAPTER

FOUR

# COMMAND REFERENCE

This chapter describes all SQL commands provided by Mychem. These commands are classified in five sections:

- *Conversion Commands* - this section details functions that convert chemical files.
- *Helper Commands* - this section details functions that return informations about the Mychem environment.
- *Modification Commands* - this section details functions that modify chemical structures.
- *Molmatch Commands* - this section details functions that compare chemical structures.
- *Property Commands* - this section details functions that compute molecular properties.

## 4.1 The Default Molecule Type

Many functions are using or returning a molecule in *DEFAULT_TYPE* format. Since version 0.6.0 of Mychem, the DEFAULT_TYPE format is *MDL Molfile (V2000)*. In earlier versions, the DEFAULT_TYPE format was *InChI*.

The speed of several functions has been measured when using InChI and MDL Molfile formats. It has been shown that using MDL Molfile is a bit faster than InChI. Using the MDL Molfile format is also interesting, as it permits to store 2D or 3D coordinates. Moreover, the MDL Molfile format is commonly used by many softwares.

## 4.2 Conversion Commands

Conversion commmands permit to convert chemical data from a format to another format. Mychem uses the Open Babel library for the conversion, but does not implement all the 80 chemical file formats supported by Open Babel. Mychem supports only few of them. They are detailed in the *Molecule Formats* appendix.

To avoid having too many functions, a set of two functions is available for each format: FORMAT_TO_MOLECULE and MOLECULE_TO_FORMAT. If you need other formats, please open a ticket on the feature tracker.

**Note**

If a function of this section fails, it returns an empty string.

- `CML_TO_MOLECULE(molecule)`

  `CML_TO_MOLECULE` converts a *molecule* in CML format to a molecule in DEFAULT_TYPE format.

```
mysql> SELECT CML_TO_MOLECULE(cml_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> 2-Aminoacetic acid
 OpenBabel11190809032D
```

(continues on next page)

(continued from previous page)

```
 10  9  0  0  0  0  0  0  0  0999 V2000
   -0.1068   -1.0521    1.1509 H   0  0  0  0  0  0  0  0  0  0  0  0
    0.0877   -0.0798    0.6477 C   0  0  0  0  0  0  0  0  0  0  0  0
   -0.2870    0.7082    1.3331 H   0  0  0  0  0  0  0  0  0  0  0  0
    1.5185    0.1919    0.3951 N   0  0  0  0  0  0  0  0  0  0  0  0
    2.0168    0.1266    1.2568 H   0  0  0  0  0  0  0  0  0  0  0  0
    1.8890   -0.4693   -0.2544 H   0  0  0  0  0  0  0  0  0  0  0  0
   -0.6775   -0.0767   -0.6578 C   0  0  0  0  0  0  0  0  0  0  0  0
   -0.4455   -0.6968   -1.6798 O   0  0  0  0  0  0  0  0  0  0  0  0
   -1.7851    0.6991   -0.6705 O   0  0  0  0  0  0  0  0  0  0  0  0
   -2.2101    0.6489   -1.5212 H   0  0  0  0  0  0  0  0  0  0  0  0
  1  2  1  0  0  0  0
  2  3  1  0  0  0  0
  2  4  1  0  0  0  0
  2  7  1  0  0  0  0
  4  5  1  0  0  0  0
  4  6  1  0  0  0  0
  7  8  2  0  0  0  0
  7  9  1  0  0  0  0
  9 10  1  0  0  0  0
M  END
```

- `MOLECULE_TO_CML(molecule)`

  `MOLECULE_TO_CML` converts a *molecule* in DEFAULT_TYPE format to a molecule in CML format.

```
mysql> SELECT MOLECULE_TO_CML(molecule_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> <molecule id="id2-Aminoacetic acid">
 <name>2-Aminoacetic acid</name>
 <atomArray>
  <atom id="a1" elementType="H" x3="-0.106800" y3="-1.052100" z3="1.150900"/>
  <atom id="a2" elementType="C" x3="0.087700" y3="-0.079800" z3="0.647700"/>
  <atom id="a3" elementType="H" x3="-0.287000" y3="0.708200" z3="1.333100"/>
  <atom id="a4" elementType="N" x3="1.518500" y3="0.191900" z3="0.395100"/>
  <atom id="a5" elementType="H" x3="2.016800" y3="0.126600" z3="1.256800"/>
  <atom id="a6" elementType="H" x3="1.889000" y3="-0.469300" z3="-0.254400"/>
  <atom id="a7" elementType="C" x3="-0.677500" y3="-0.076700" z3="-0.657800"/>
  <atom id="a8" elementType="O" x3="-0.445500" y3="-0.696800" z3="-1.679800"/>
  <atom id="a9" elementType="O" x3="-1.785100" y3="0.699100" z3="-0.670500"/>
  <atom id="a10" elementType="H" x3="-2.210100" y3="0.648900" z3="-1.521200"/>
 </atomArray>
 <bondArray>
  <bond atomRefs2="a1 a2" order="1"/>
  <bond atomRefs2="a2 a3" order="1"/>
  <bond atomRefs2="a2 a4" order="1"/>
  <bond atomRefs2="a2 a7" order="1"/>
  <bond atomRefs2="a4 a5" order="1"/>
  <bond atomRefs2="a4 a6" order="1"/>
  <bond atomRefs2="a7 a8" order="2"/>
  <bond atomRefs2="a7 a9" order="1"/>
  <bond atomRefs2="a9 a10" order="1"/>
 </bondArray>
</molecule>
```

- `FINGERPRINT(molecule, type)`

  `FINGERPRINT` converts a *molecule* in DEFAULT_TYPE format to a fingerprint. The fingerprint type is speci-

fied by the second argument (FP2, FP3 or FP4). The SQL type of the converted molecule is a binary string for all kinds of fingerprint.

```
mysql> SELECT FINGERPRINT(molecule_col, "FP2") FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> binary fingerprint (type FP2)
```

**Note**

Tanimoto scoring can be improved by using a concatenation of different fingerprint types. The concatenation of fingerprints can be performed with the following query:

```
mysql> SELECT CONCAT(FINGERPRINT(molecule_col,"FP2"),
    -> FINGERPRINT(molecule_col,"FP3")) FROM tbl_name
    -> WHERE id=9;
       -> binary fingerprint (type FP2 + type FP3)
```

- `FINGERPRINT2(molecule)`

  `FINGERPRINT2` converts a *molecule* in DEFAULT_TYPE format to a FP2 fingerprint. The SQL type of FP2 fingerprints is a binary string.

```
mysql> SELECT FINGERPRINT2(molecule_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> binary fingerprint (type FP2)
```

- `FINGERPRINT3(molecule)`

  `FINGERPRINT3` converts a *molecule* in DEFAULT_TYPE format to a FP3 fingerprint. The SQL type of FP3 fingerprints is a binary string.

```
mysql> SELECT FINGERPRINT3(molecule_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> binary fingerprint (type FP3)
```

- `FINGERPRINT4(molecule)`

  `FINGERPRINT4` converts a *molecule* in DEFAULT_TYPE format to a FP4 fingerprint. The SQL type of FP4 fingerprints is a binary string.

```
mysql> SELECT FINGERPRINT4(molecule_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> binary fingerprint (type FP4)
```

- `INCHI_TO_MOLECULE(molecule)`

  `INCHI_TO_MOLECULE` converts a *molecule* in InChI format to a molecule in DEFAULT_TYPE format.

```
mysql> SELECT INCHI_TO_MOLECULE(inchi_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        ->
 OpenBabel11190809142D

  5  4  0  0  0  0  0  0  0  0999 V2000
    0.0000    0.0000    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
    0.0000    0.0000    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
    0.0000    0.0000    0.0000 N   0  0  0  0  0  0  0  0  0  0  0  0
    0.0000    0.0000    0.0000 O   0  0  0  0  0  0  0  0  0  0  0  0
    0.0000    0.0000    0.0000 O   0  0  0  0  0  0  0  0  0  0  0  0
```

(continues on next page)

(continued from previous page)

```
  1  2  1  0  0  0  0
  1  3  1  0  0  0  0
  2  4  2  0  0  0  0
  2  5  1  0  0  0  0
M  END
```

- MOLECULE_TO_INCHI(molecule)

  MOLECULE_TO_INCHI converts a *molecule* in DEFAULT_TYPE format to a molecule in InChI format.

```
mysql> SELECT MOLECULE_TO_INCHI(molecule_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> InChI=1S/C2H5NO2/c3-1-2(4)5/h1,3H2,(H,4,5)
```

  **Note**

  The INCHI_VERSION function permits to know which version of the InChI library is used.

- MOL2_TO_MOLECULE(molecule)

  MOL2_TO_MOLECULE converts a *molecule* in Sybyl Mol2 format to a molecule in DEFAULT_TYPE format.

```
mysql> SELECT MOL2_TO_MOLECULE(mol2_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> 2-Aminoacetic acid
 OpenBabel11190809032D

 10  9  0  0  0  0  0  0  0  0999 V2000
   -0.1068   -1.0521    1.1509 H   0  0  0  0  0  0  0  0  0  0  0  0
    0.0877   -0.0798    0.6477 C   0  0  0  0  0  0  0  0  0  0  0  0
   -0.2870    0.7082    1.3331 H   0  0  0  0  0  0  0  0  0  0  0  0
    1.5185    0.1919    0.3951 N   0  0  0  0  0  0  0  0  0  0  0  0
    2.0168    0.1266    1.2568 H   0  0  0  0  0  0  0  0  0  0  0  0
    1.8890   -0.4693   -0.2544 H   0  0  0  0  0  0  0  0  0  0  0  0
   -0.6775   -0.0767   -0.6578 C   0  0  0  0  0  0  0  0  0  0  0  0
   -0.4455   -0.6968   -1.6798 O   0  0  0  0  0  0  0  0  0  0  0  0
   -1.7851    0.6991   -0.6705 O   0  0  0  0  0  0  0  0  0  0  0  0
   -2.2101    0.6489   -1.5212 H   0  0  0  0  0  0  0  0  0  0  0  0
  1  2  1  0  0  0  0
  2  3  1  0  0  0  0
  2  4  1  0  0  0  0
  2  7  1  0  0  0  0
  4  5  1  0  0  0  0
  4  6  1  0  0  0  0
  7  8  2  0  0  0  0
  7  9  1  0  0  0  0
  9 10  1  0  0  0  0
M  END
```

- MOLECULE_TO_MOL2(molecule)

  MOLECULE_TO_MOL2 converts a *molecule* in DEFAULT_TYPE format to a molecule in Sybyl Mol2 format.

```
mysql> SELECT MOLECULE_TO_MOL2(molecule_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        ->@<TRIPOS>MOLECULE
2-Aminoacetic acid
 10 9 0 0 0
```

(continues on next page)

(continued from previous page)

```
SMALL
GASTEIGER

@<TRIPOS>ATOM
      1 HA1         -0.1068   -1.0521    1.1509 H       1  GLY1         0.0537
      2 CA           0.0877   -0.0798    0.6477 C.3     1  GLY1         0.0918
      3 HA2         -0.2870    0.7082    1.3331 H       1  GLY1         0.0537
      4 N            1.5185    0.1919    0.3951 N.3     1  GLY1        -0.3209
      5 H1           2.0168    0.1266    1.2568 H       1  GLY1         0.1187
      6 H2           1.8890   -0.4693   -0.2544 H       1  GLY1         0.1187
      7 C           -0.6775   -0.0767   -0.6578 C.2     1  GLY1         0.3185
      8 O           -0.4455   -0.6968   -1.6798 O.2     1  GLY1        -0.2496
      9 OXT         -1.7851    0.6991   -0.6705 O.3     1  GLY1        -0.4797
     10 HXT         -2.2101    0.6489   -1.5212 H       1  GLY1         0.2951
@<TRIPOS>BOND
     1     1     2    1
     2     2     3    1
     3     2     4    1
     4     2     7    1
     5     4     5    1
     6     4     6    1
     7     7     8    2
     8     7     9    1
     9     9    10    1
```

- `MOLECULE_TO_MOLECULE(molecule)`

  `MOLECULE_TO_MOLECULE` converts a *molecule* in the old DEFAULT_TYPE format (InChI) to a molecule in the current DEFAULT_TYPE format (MDL Molfile). This function is useful when updating a database with a new version of Mychem.

```
mysql> SELECT MOLECULE_TO_MOLECULE(molecule_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
       ->
 OpenBabel11190809092D

  5  4  0  0  0  0  0  0  0  0999 V2000
    0.0000    0.0000    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
    0.0000    0.0000    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
    0.0000    0.0000    0.0000 N   0  0  0  0  0  0  0  0  0  0  0  0
    0.0000    0.0000    0.0000 O   0  0  0  0  0  0  0  0  0  0  0  0
    0.0000    0.0000    0.0000 O   0  0  0  0  0  0  0  0  0  0  0  0
  1  2  1  0  0  0  0
  1  3  1  0  0  0  0
  2  4  2  0  0  0  0
  2  5  1  0  0  0  0
M  END
```

- `MOLECULE_TO_SERIALIZEDOBMOL(molecule)`

  `MOLECULE_TO_SERIALIZEDOBMOL` converts a *molecule* in DEFAULT_TYPE format to a serialized OBMol object. The SQL type of the serialized object is a binary string.

```
mysql> SELECT MOLECULE_TO_SERIALIZEDOBMOL(molecule_col)
    -> FROM tbl_name WHERE name='2-Aminoacetic acid';
       -> binary string
```

- `MOLFILE_TO_MOLECULE(molecule)`

MOLFILE_TO_MOLECULE converts a *molecule* in MDL Molfile (V2000) format to a molecule in DEFAULT_TYPE format.

```
mysql> SELECT MOLFILE_TO_MOLECULE(molfile_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> 2-Aminoacetic acid
 OpenBabel12091111263D

 10  9  0  0  0  0  0  0  0  0999 V2000
   -0.1068   -1.0521    1.1509 H   0  0  0  0  0  0  0  0  0  0  0  0
    0.0877   -0.0798    0.6477 C   0  0  0  0  0  0  0  0  0  0  0  0
   -0.2870    0.7082    1.3331 H   0  0  0  0  0  0  0  0  0  0  0  0
    1.5185    0.1919    0.3951 N   0  0  0  0  0  0  0  0  0  0  0  0
    2.0168    0.1266    1.2568 H   0  0  0  0  0  0  0  0  0  0  0  0
    1.8890   -0.4693   -0.2544 H   0  0  0  0  0  0  0  0  0  0  0  0
   -0.6775   -0.0767   -0.6578 C   0  0  0  0  0  0  0  0  0  0  0  0
   -0.4455   -0.6968   -1.6798 O   0  0  0  0  0  0  0  0  0  0  0  0
   -1.7851    0.6991   -0.6705 O   0  0  0  0  0  0  0  0  0  0  0  0
   -2.2101    0.6489   -1.5212 H   0  0  0  0  0  0  0  0  0  0  0  0
  1  2  1  0  0  0  0
  2  3  1  0  0  0  0
  2  4  1  0  0  0  0
  2  7  1  0  0  0  0
  4  5  1  0  0  0  0
  4  6  1  0  0  0  0
  7  8  2  0  0  0  0
  7  9  1  0  0  0  0
  9 10  1  0  0  0  0
M  END
```

- MOLECULE_TO_MOLFILE(molecule)

  MOLECULE_TO_MOLFILE converts a *molecule* in DEFAULT_TYPE format to a molecule in MDL Molfile (V2000) format.

```
mysql> SELECT MOLECULE_TO_MOLFILE(molecule_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> 2-Aminoacetic acid
 OpenBabel11190809062D

 10  9  0  0  0  0  0  0  0  0999 V2000
   -0.1068   -1.0521    1.1509 H   0  0  0  0  0  0  0  0  0  0  0  0
    0.0877   -0.0798    0.6477 C   0  0  0  0  0  0  0  0  0  0  0  0
   -0.2870    0.7082    1.3331 H   0  0  0  0  0  0  0  0  0  0  0  0
    1.5185    0.1919    0.3951 N   0  0  0  0  0  0  0  0  0  0  0  0
    2.0168    0.1266    1.2568 H   0  0  0  0  0  0  0  0  0  0  0  0
    1.8890   -0.4693   -0.2544 H   0  0  0  0  0  0  0  0  0  0  0  0
   -0.6775   -0.0767   -0.6578 C   0  0  0  0  0  0  0  0  0  0  0  0
   -0.4455   -0.6968   -1.6798 O   0  0  0  0  0  0  0  0  0  0  0  0
   -1.7851    0.6991   -0.6705 O   0  0  0  0  0  0  0  0  0  0  0  0
   -2.2101    0.6489   -1.5212 H   0  0  0  0  0  0  0  0  0  0  0  0
  1  2  1  0  0  0  0
  2  3  1  0  0  0  0
  2  4  1  0  0  0  0
  2  7  1  0  0  0  0
  4  5  1  0  0  0  0
  4  6  1  0  0  0  0
  7  8  2  0  0  0  0
```

(continues on next page)

(continued from previous page)

```
  7  9  1  0  0  0  0
  9 10  1  0  0  0  0
  5  4  0  0  0  0  0
M  END
```

- PDB_TO_MOLECULE(molecule)

  PDB_TO_MOLECULE converts a *molecule* in PDB format to a molecule in DEFAULT_TYPE format.

```
mysql> SELECT PDB_TO_MOLECULE(pdb_col) FROM tbl_name
    -> WHERE name='1CRN';
       ->
 OpenBabel08210907243D

327337  0  0  0  0  0  0  0  0999 V2000
   17.0470   14.0990    3.6250 N   0  0  0  0  0
   16.9670   12.7840    4.3380 C   0  0  0  0  0
   15.6850   12.7550    5.1330 C   0  0  0  0  0
   15.2680   13.8250    5.5940 O   0  0  0  0  0
   18.1700   12.7030    5.3370 C   0  0  0  0  0
   19.3340   12.8290    4.4630 O   0  0  0  0  0
   18.1500   11.5460    6.3040 C   0  0  0  0  0
   15.1150   11.5550    5.2650 N   0  0  0  0  0
   13.8560   11.4690    6.0660 C   0  0  0  0  0
   14.1640   10.7850    7.3790 C   0  0  0  0  0
   14.9930    9.8620    7.4430 O   0  0  0  0  0
   12.7320   10.7110    5.2610 C   0  0  0  0  0
   13.3080    9.4390    4.9260 O   0  0  0  0  0
   12.4840   11.4420    3.8950 C   0  0  0  0  0
   13.4880   11.2410    8.4170 N   0  0  0  0  0
   13.6600   10.7070    9.7870 C   0  0  0  0  0
   12.2690   10.4310   10.3230 C   0  0  0  0  0
   11.3930   11.3080   10.1850 O   0  0  0  0  0
   14.3680   11.7480   10.6910 C   0  0  0  0  0
   15.8850   12.4260   10.0160 S   0  0  0  0  0
   12.0190    9.2720   10.9280 N   0  0  0  0  0
   10.6460    8.9910   11.4080 C   0  0  0  0  0
   10.6540    8.7930   12.9190 C   0  0  0  0  0
   11.6590    8.2960   13.4910 O   0  0  0  0  0
   10.0570    7.7520   10.6820 C   0  0  0  0  0
    9.8370    8.0180    8.9040 S   0  0  0  0  0
    9.5610    9.1080   13.5630 N   0  0  0  0  0
    9.4480    9.0340   15.0120 C   0  0  0  0  0
    9.2880    7.6700   15.6060 C   0  0  0  0  0
...
```

- SMILES_TO_MOLECULE(molecule)

  SMILES_TO_MOLECULE converts a *molecule* in SMILES format to a molecule in DEFAULT_TYPE format.

```
mysql> SELECT SMILES_TO_MOLECULE(smiles_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
       ->
 OpenBabel11190809142D

  5  4  0  0  0  0  0  0  0  0999 V2000
    0.0000    0.0000    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
```

(continues on next page)

(continued from previous page)

```
    0.0000    0.0000    0.0000 N   0  0  0  0  0  0  0  0  0  0  0  0
    0.0000    0.0000    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
    0.0000    0.0000    0.0000 O   0  0  0  0  0  0  0  0  0  0  0  0
    0.0000    0.0000    0.0000 O   0  0  0  0  0  0  0  0  0  0  0  0
  1  2  1  0  0  0  0
  1  3  1  0  0  0  0
  3  4  2  0  0  0  0
  3  5  1  0  0  0  0
M  END
```

- `MOLECULE_TO_SMILES(molecule)`

  `MOLECULE_TO_SMILES` converts a *molecule* in DEFAULT_TYPE format to molecule in SMILES format.

```
mysql> SELECT MOLECULE_TO_SMILES(molecule_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> C(C(=O)O)N
```

  **Note**

  If you need a single canonical form for any particular molecule, regardless of atom order, the use of the `MOLECULE_TO_CANONICAL_SMILES` function should be preferred.

- `MOLECULE_TO_CANONICAL_SMILES(molecule)`

  `MOLECULE_TO_CANONICAL_SMILES` converts a *molecule* in DEFAULT_TYPE format to a molecule in Canonical SMILES format.

```
mysql> SELECT MOLECULE_TO_CANONICAL_SMILES(molecule_col)
    -> FROM tbl_name WHERE name='2-Aminoacetic acid';
        -> NCC(=O)O
```

- `V3000_TO_MOLECULE(molecule)`

  `V3000_TO_MOLECULE` converts a *molecule* in MDL Molfile (V3000) format to a molecule in DEFAULT_TYPE format.

```
mysql> SELECT V3000_TO_MOLECULE(V3000_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> 2-Aminoacetic acid
 OpenBabel11190809082D

 10  9  0  0  0  0  0  0  0  0999 V2000
   -0.1068   -1.0521    1.1509 H   0  0  0  0  0  0  0  0  0  0  0  0
    0.0877   -0.0798    0.6477 C   0  0  0  0  0  0  0  0  0  0  0  0
   -0.2870    0.7082    1.3331 H   0  0  0  0  0  0  0  0  0  0  0  0
    1.5185    0.1919    0.3951 N   0  0  0  0  0  0  0  0  0  0  0  0
    2.0168    0.1266    1.2568 H   0  0  0  0  0  0  0  0  0  0  0  0
    1.8890   -0.4693   -0.2544 H   0  0  0  0  0  0  0  0  0  0  0  0
   -0.6775   -0.0767   -0.6578 C   0  0  0  0  0  0  0  0  0  0  0  0
   -0.4455   -0.6968   -1.6798 O   0  0  0  0  0  0  0  0  0  0  0  0
   -1.7851    0.6991   -0.6705 O   0  0  0  0  0  0  0  0  0  0  0  0
   -2.2101    0.6489   -1.5212 H   0  0  0  0  0  0  0  0  0  0  0  0
  1  2  1  0  0  0  0
  2  3  1  0  0  0  0
  2  4  1  0  0  0  0
  2  7  1  0  0  0  0
  4  5  1  0  0  0  0
```

(continues on next page)

(continued from previous page)

```
  4  6  1  0  0  0  0
  7  8  2  0  0  0  0
  7  9  1  0  0  0  0
  9 10  1  0  0  0  0
M  END
```

- `MOLECULE_TO_V3000(molecule)`

  `MOLECULE_TO_V3000` converts a *molecule* in DEFAULT_TYPE format to a molecule in MDL Molfile (V3000) format.

```
mysql> SELECT MOLECULE_TO_V3000(molecule_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> 2-Aminoacetic acid
 OpenBabel11190809132D
  0  0  0     0  0            999 V3000

M  V30 BEGIN CTAB
M  V30 COUNTS 10 9 0 0 0
M  V30 BEGIN ATOM
M  V30 1 H -0.1068 -1.0521 1.1509 0
M  V30 2 C 0.0877 -0.0798 0.6477 0
M  V30 3 H -0.287 0.7082 1.3331 0
M  V30 4 N 1.5185 0.1919 0.3951 0
M  V30 5 H 2.0168 0.1266 1.2568 0
M  V30 6 H 1.889 -0.4693 -0.2544 0
M  V30 7 C -0.6775 -0.0767 -0.6578 0
M  V30 8 O -0.4455 -0.6968 -1.6798 0
M  V30 9 O -1.7851 0.6991 -0.6705 0
M  V30 10 H -2.2101 0.6489 -1.5212 0
M  V30 END ATOM
M  V30 BEGIN BOND
M  V30 1 1 1 2
M  V30 2 1 2 3
M  V30 3 1 2 4
M  V30 4 1 2 7
M  V30 5 1 4 5
M  V30 6 1 4 6
M  V30 7 2 7 8
M  V30 8 1 7 9
M  V30 9 1 9 10
M  V30 END BOND
M  V30 END CTAB
M  END
```

## 4.3 Helper Commands

This section details *helper* functions. They permit to get informations about the Mychem environment.

- `INCHI_VERSION()`

  `INCHI_VERSION` returns the version of the InChI library.

```
mysql> SELECT INCHI_VERSION();
        -> 1.02
```

- `MYCHEM_VERSION()`

  `MYCHEM_VERSION` returns the Mychem version.

```
mysql> SELECT MYCHEM_VERSION();
        -> 1.0.0
```

- `OPENBABEL_VERSION()`

  `OPENBABEL_VERSION` returns the Open Babel version.

```
mysql> SELECT OPENBABEL_VERSION();
        -> 2.3.2
```

## 4.4 Modification Commands

This section describes *modification* functions that modify chemical structures. The following functions are fully working starting on version 0.6.0 of Mychem.

**Note**

If a function of this section fails, it returns an empty string.

- `ADD_HYDROGENS(molecule)`

  `ADD_HYDROGENS` adds hydrogens to a *molecule* (makes explicit the hydrogen atoms).

```
mysql> SELECT ADD_HYDROGENS(molecule_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> 2-Aminoacetic acid
 OpenBabel11190809242D

 10  9  0  0  0  0  0  0  0  0999 V2000
   -0.1068   -1.0521    1.1509 H   0  0  0  0  0  0  0  0  0  0  0  0
    0.0877   -0.0798    0.6477 C   0  0  0  0  0  0  0  0  0  0  0  0
   -0.2870    0.7082    1.3331 H   0  0  0  0  0  0  0  0  0  0  0  0
    1.5185    0.1919    0.3951 N   0  0  0  0  0  0  0  0  0  0  0  0
    2.0168    0.1266    1.2568 H   0  0  0  0  0  0  0  0  0  0  0  0
    1.8890   -0.4693   -0.2544 H   0  0  0  0  0  0  0  0  0  0  0  0
   -0.6775   -0.0767   -0.6578 C   0  0  0  0  0  0  0  0  0  0  0  0
   -0.4455   -0.6968   -1.6798 O   0  0  0  0  0  0  0  0  0  0  0  0
   -1.7851    0.6991   -0.6705 O   0  0  0  0  0  0  0  0  0  0  0  0
   -2.2101    0.6489   -1.5212 H   0  0  0  0  0  0  0  0  0  0  0  0
  1  2  1  0  0  0  0
  2  3  1  0  0  0  0
  2  4  1  0  0  0  0
  2  7  1  0  0  0  0
  4  5  1  0  0  0  0
  4  6  1  0  0  0  0
  7  8  2  0  0  0  0
  7  9  1  0  0  0  0
  9 10  1  0  0  0  0
M  END
```

- `REMOVE_HYDROGENS(molecule)`

  `REMOVE_HYDROGENS` removes the hydrogens from a *molecule* (makes implicit the hydrogen atoms).

```
mysql> SELECT REMOVE_HYDROGENS(molecule_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        ->  2-Aminoacetic acid
 OpenBabel11190809252D

  5  4  0  0  0  0  0  0  0  0999 V2000
    0.0877   -0.0798    0.6477 C   0  0  0  0  0  0  0  0  0  0  0  0
    1.5185    0.1919    0.3951 N   0  0  0  0  0  0  0  0  0  0  0  0
   -0.6775   -0.0767   -0.6578 C   0  0  0  0  0  0  0  0  0  0  0  0
   -0.4455   -0.6968   -1.6798 O   0  0  0  0  0  0  0  0  0  0  0  0
   -1.7851    0.6991   -0.6705 O   0  0  0  0  0  0  0  0  0  0  0  0
  1  2  1  0  0  0  0
  1  3  1  0  0  0  0
  3  4  2  0  0  0  0
  3  5  1  0  0  0  0
M  END
```

- `STRIP_SALTS(molecule)`

  `STRIP_SALTS` removes all atoms except for the larger contiguous fragment.

```
mysql> SELECT STRIP_SALTS(molecule_col) FROM tbl_name
    -> WHERE name='sodium 2-aminoacetate';
        ->
 OpenBabel11190812342D

  5  4  0  0  0  0  0  0  0  0999 V2000
    0.0000    0.0000    0.0000 N   0  0  0  0  0  0  0  0  0  0  0  0
    0.0000    0.0000    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
    0.0000    0.0000    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
    0.0000    0.0000    0.0000 O   0  0  0  0  0  0  0  0  0  0  0  0
    0.0000    0.0000    0.0000 O   0  0  0  0  0  0  0  0  0  0  0  0
  1  2  1  0  0  0  0
  2  3  1  0  0  0  0
  3  4  2  0  0  0  0
  3  5  1  0  0  0  0
M  CHG  1   5  -1
M  END
```

## 4.5 Molmatch Commands

The *molmatch* functions permit to compare chemical structures.

- `BIT_FP_AND(fingerprint, fingerprint)`

  `BIT_FP_AND` operates on two fingerprints (bit patterns) of equal length and performs the logical AND operation on each pair of corresponding bits. In each pair, the result is 1 if the both bits are 1. Otherwise, the result is 0. If the two fingerprints do not have the same length, the function returns NULL.

```
mysql> SELECT BIT_FP_AND(fingerprint1, fingerprint2);
        -> binary fingerprint
```

  **Note**

  The `BIT_FP_AND` function is very useful when working with structure fingerprints. For example, if a molecule (with a fingerprint fp1) is a substructure of an other molecule (with a fingerprint fp2), the following property is observed:

```
mysql> SELECT TANIMOTO(BIT_FP_AND(fp1, fp2), fp1);
        -> 1
```

- `BIT_FP_COUNT(fingerprint)`

  `BIT_FP_COUNT` returns the number of bits that are set in the *fingerprint* binary representation.

```
mysql> SELECT BIT_FP_COUNT(fp_col) FROM tbl_name
    -> WHERE name='1H-indole';
        -> 23
```

- `BIT_FP_OR(fingerprint, fingerprint)`

  `BIT_FP_OR` operates on two fingerprints (bit patterns) of equal length and performs the logical OR operation on each pair of corresponding bits. In each pair, if the first bit is 1 or the second bit is 1 (or both), the result is 1. Otherwise, the result is 0. If the two fingerprints do not have same length, the function returns NULL.

```
mysql> SELECT BIT_FP_OR(fingerprint1, fingerprint2);
        -> binary fingerprint
```

- `MATCH_SUBSTRUCT(query_smarts, reference_obmol)`

  `MATCH_SUBSTRUCT` checks if a *query_smarts* fragment is a substructure of a *reference_obmol* molecule. The first argument is a SMARTS string, whereas the second argument is a serialized OBMol object. The second argument type is generated by the `MOLECULE_TO_SERIALIZEDOBMOL` function. If the *query_smarts* is a substructure of *reference_obmol*, the function returns 1, otherwise, it returns 0.

```
mysql> SELECT MATCH_SUBSTRUCT('C=O', serializedobmol_col)
    -> FROM tbl_name WHERE name='2-Aminoacetic acid';
        -> 1
```

  **Note**

  If the function encounters an error, it returns NULL.

- `SUBSTRUCT_ATOM_IDS(query_smarts, reference_obmol)`

  `SUBSTRUCT_ATOM_IDS` returns the atom ids of a *reference_obmol* molecule that are contained in substructures matching a *query_smarts* fragment. The first argument is a SMARTS string, whereas the second argument is a serialized OBMol object. The second argument type is generated by the `MOLECULE_TO_SERIALIZEDOBMOL` function. If a *reference_obmol* molecule contains several fragments matching a *query_smarts* fragment, a list of items is returned. Each item contains a fragment's atom ids and is separated from the next item by a semicolon character.

```
mysql> SELECT SUBSTRUCT_ATOM_IDS('C(=O)', serializedobmol_col)
    -> FROM tbl_name WHERE name='2-Aminoacetic acid';
        -> 2 3 ;
```

  **Note**

  If the function encounter an error, it returns NULL.

- `SUBSTRUCT_COUNT(query_smarts, reference_obmol)`

  `SUBSTRUCT_COUNT` returns the number of *query_smarts* fragments founded in a *reference_obmol* molecule. The first argument is a SMARTS string, whereas the second argument is a serialized OBMol object. The second argument type is generated by the `MOLECULE_TO_SERIALIZEDOBMOL` function.

```
mysql> SELECT SUBSTRUCT_COUNT('C(=O)', serializedobmol_col)
    -> FROM tbl_name WHERE name='2-Aminoacetic acid';
        -> 2
```

**Note**

If the function encounter an error, it returns NULL.

- `TANIMOTO(first_fingerprint, second_fingerprint)`

  `TANIMOTO` returns the tanimoto coefficient between two fingerprints. Fingerprints are bit patterns and can be generated with the `FINGERPRINT` function. The returned value is comprised between 0 and 1. The higher the tanimoto coefficient is, the more the molecules are similar.

```
mysql> SELECT TANIMOTO(molecule_fp, fp_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> 0.8934
```

**Note**

The use of another Mychem functions (like `FINGERPRINT` or `FINGERPRINT2`) within the `TANIMOTO` function makes the query slower. In order to get the best performance, you should use the `SET` function of MySQL:

```
mysql> SET @fp = (SELECT FINGERPRINT2(
    -> SMILES_TO_MOLECULE('C(C(=O)O)N')));
mysql> SELECT id FROM tbl_name WHERE TANIMOTO(@fp, fp_col)
    -> FROM tbl_name > 0.7;
        -> list of id
```

## 4.6 Property Commands

This section describes several functions that calculate molecular properties.

- `EXACTMASS(molecule)`

  `EXACTMASS` returns the monoisotopic molecular weight of a *molecule*. The monoisotopic molecular weight is defined as the molecular weight calculated using the mass of the most abundant isotope for each element of a molecule. The unit of the returned value is $g.mol^{-1}$.

```
mysql> SELECT EXACTMASS(molecule_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> 75.032028
```

- `IS_2D(molecule)`

  `IS_2D` returns 1 if a molecule has 2D coordinates.

```
mysql> SELECT IS_2D(molecule_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> 1
```

- `IS_3D(molecule)`

  `IS_3D` returns 1 if a molecule has 3D coordinates.

```
mysql> SELECT IS_3D(molecule_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> 1
```

- `IS_CHIRAL(molecule)`

  `IS_CHIRAL` returns 1 if a molecule is chiral.

```
mysql> SELECT IS_CHIRAL(molecule_col) FROM tbl_name
    -> WHERE name='2S-Butan-2-ol';
        -> 1
```

- `MOLFORMULA(molecule)`

  `MOLFORMULA` returns the molecular formula of a *molecule*.

```
mysql> SELECT MOLFORMULA(molecule_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> C2H5NO2
```

- `MOLLOGP(molecule)`

  `MOLLOGP` returns the LogP of a *molecule*.

```
mysql> SELECT MOLLOGP(molecule_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> -0.27
```

  **Note**

  Note that the result of this function is depending on the hydrogen atoms. If there is any doubt on the presence of hydrogen atoms in the molecule, it is recommended to use the `ADD_HYDROGENS` function.

- `MOLMR(molecule)`

  `MOLMR` returns the molar refractivity of a *molecule*. The unit of the returned value is $J.mol^{-1}.K^{-1}$.

```
mysql> SELECT MOLMR(molecule_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> 16.2072
```

  **Note**

  Note that the result of this function is depending on the hydrogen atoms. If there is any doubt on the presence of hydrogen atoms in the molecule, it is recommended to use the `ADD_HYDROGENS` function.

- `MOLPSA(molecule)`

  `MOLPSA` returns the topological polar surface area of a molecule. The unit of the returned value is $Å^2$.

```
mysql> SELECT MOLPSA(molecule_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> 63.32
```

  **Note**

  Note that the result of this function is depending on the hydrogen atoms. If there is any doubt on the presence of hydrogen atoms in the molecule, it is recommended to use the `ADD_HYDROGENS` function.

- `MOLWEIGHT(molecule)`

  `MOLWEIGHT` returns the molecular weight of a *molecule*. The unit of the returned value is $g.mol^{-1}$.

```
mysql> SELECT MOLWEIGHT(molecule_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> 75.066600
```

- NUMBER_OF_ACCEPTORS(molecule)

  NUMBER_OF_ACCEPTORS returns the number of hydrogen-bond acceptors in a *molecule*.

```
mysql> SELECT NUMBER_OF_ACCEPTORS(molecule_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> 3
```

  **Note**

  Note that the result of this function is depending on the hydrogen atoms. If there is any doubt on the presence of hydrogen atoms in the molecule, it is recommended to use the ADD_HYDROGENS function.

- NUMBER_OF_ATOMS(molecule)

  NUMBER_OF_ATOMS returns the number of atoms in a *molecule*.

```
mysql> SELECT NUMBER_OF_ATOMS(molecule_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> 10
```

- NUMBER_OF_BONDS(molecule)

  NUMBER_OF_BONDS returns the number of bonds in a *molecule*.

```
mysql> SELECT NUMBER_OF_BONDS(molecule_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> 9
```

- NUMBER_OF_DONORS(molecule)

  NUMBER_OF_DONORS returns the numbers of hydrogen-bond donors in a *molecule*.

```
mysql> SELECT NUMBER_OF_DONORS(molecule_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> 2
```

  **Note**

  Note that the result of this function is depending on the presence of hydrogen atoms in the molecule. If the hydrogen atoms are not described by the molecule, the ADD_HYDROGENS function must be used. The two following examples described the effect of the ADD_HYDROGENS function:

```
mysql> SELECT NUMBER_OF_DONORS(
    -> SMILES_TO_MOLECULE('O=C(O)CCN'));
        -> 0
```

```
mysql> SELECT NUMBER_OF_DONORS(ADD_HYDROGENS(
    -> SMILES_TO_MOLECULE('O=C(O)CCN')));
        -> 2
```

- NUMBER_OF_HEAVY_ATOMS(molecule)

  NUMBER_OF_HEAVY_ATOMS returns the number of heavy atoms in a *molecule* (all atoms except hydrogen).

```
mysql> SELECT NUMBER_OF_HEAVY_ATOMS(molecule_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> 5
```

- NUMBER_OF_RINGS(molecule)

NUMBER_OF_RINGS returns the number of rings in a *molecule*.

```
mysql> SELECT NUMBER_OF_RINGS(molecule_col) FROM tbl_name
    -> WHERE name='adenine';
        -> 2
```

- NUMBER_OF_ROTABLE_BONDS(molecule)

  NUMBER_OF_ROTABLE_BONDS returns the number of rotable bonds in a *molecule*.

```
mysql> SELECT NUMBER_OF_ROTABLE_BONDS(molecule_col) FROM tbl_name
    -> WHERE name='2-Aminoacetic acid';
        -> 1
```

- TOTAL_CHARGE(molecule)

  TOTAL_CHARGE returns the total charge of a *molecule*.

```
mysql> SELECT TOTAL_CHARGE(SMILES_TO_MOLECULE('NCC(=O)[O-]'));
        -> -1
```

CHAPTER

FIVE

# TROUBLESHOOTING

When running Mychem, you may encounter certain errors that prevent the Mychem software to run perfectly. The purpose of this chapter is to help you to diagnose and correct some of these errors.

## 5.1 MySQL-Related Errors

This section describes the errors encounter with MySQL when running Mychem.

### 5.1.1 ERROR 2013 (HY000): Lost connection to MySQL server during query

The following examples show an error messages you may encounter when using the SMILES_TO_MOLECULE function.

```
mysql> SELECT SMILES_TO_MOLECULE('CCOCC');
ERROR 2013 (HY000): Lost connection to MySQL server during query
```

This problem has been observed with version of Mychem earlier to 0.6.0. To avoid this error, set the thread_stack parameter of MySQL to 192K. The thread_stack parameter is defined in the MySQL server configuration file.

## 5.2 AppArmor

AppArmor is a Linux Security Module implementation of name-based access controls. AppArmor confines individual programs to a set of listed files. The default configuration of AppArmor does not permit the use of Mychem. In fact, MySQL is not allowed to access the Open Babel library. To fix this problem, the following lines must be added to the `/etc/apparmor.d/local/usr.sbin.mysqld` file:

```
/usr/lib/openbabel/* m,
/usr/lib/openbabel/2.3.2/* m,
/usr/share/openbabel/* r,
/usr/share/openbabel/2.3.2/* r,

**Note**

Replace 2.3.2 by the version of OpenBabel on your system (2.3.0,
2.3.1, ...).
```

## 5.3 Other Errors

If you encounter an error not listed here, please report it on our bug tracking system.

CHAPTER

SIX

# CREDITS AND LICENSE

## 6.1 Contributions

This section lists the developers and contributors to Mychem.

### 6.1.1 Developers

These are developers that are involved in Mychem:

- Jérôme Pansanel jerome.pansanel@iphc.cnrs.fr

  Project founder and developer

- Aurélie De Luca aureliedeluca@gmail.com

  Developer

- Bjoern Gruening bjoern@gruenings.eu

  Developer and tester

- Fredrik Wallner

  For comments and bug fixes

### 6.1.2 Contributors

Contributors to Mychem are listed here:

- Ernst-Georg Schmid - Pgchem lead developer

  Pgchem is a chemoinformatics extension for PostgreSQL developed by Ernst-Georg Schmid. We have decided that Mychem Project should have the same functions as Pgchem. Of course, we are sharing our knowledge on Open Babel, Relational Database and extension programing.

- Chris Morley - Open Babel lead developer

  For comments and help on OpenBabel-Discuss mailing list.

- Noel O'Boyle - Open Babel developer

  For comments and help on OpenBabel-Discuss mailing list.

- Hans de Winter

  For debugging and tests on Mac OS X.

## 6.2 Copyright



## 6.3 License

CHAPTER

SEVEN

# CHEMICAL DATABASE MANAGER

This chapter presents the **mychemdb_manager** tool, a chemical database manager for handling chemical databases with MySQL and Mychem.

## 7.1 mychemdb_manager

The Mychem software is useful when working with chemical databases. In order to facilitate the creation and the management of such databases, `mychemdb_manager`, a Python program, is distributed with the Mychem code. This script is a command line interface that permits to create or update a chemical database.

The `mychemdb_manager` script can be found in the `scripts` directory from Mychem. It is a Python program released under the new BSD license. It requires the **pymysql** Python module. This module is provided by most GNU/Linux distributions. It can also be installed using `pip`.

The usage of the script is simple. A help can be displayed by using the `-h` option:

```
usage: mychem_manager [-h] [-v] -H HOST -U USER -D DATABASE [-P] [-n NAME_TAG]
                      [-l LOG_FILE] [-p PREFIX] [-a | -r] [-V]
                      sdfile

mychem_manager load a file in MDL SDF format into a MySQL database and creates
a chemical cartridge with Mychem.

positional arguments:
  sdfile                Name of the MDL SDF file containing the chemical-data
                        to load into the MySQL database.

optional arguments:
  -h, --help            show this help message and exit
  -v, --version         show program's version number and exit
  -H HOST, --host HOST  Name of the MySQL host.
  -U USER, --user USER  User for login to the MySQL server.
  -D DATABASE, --db DATABASE
                        Name of the MySQL database to use
  -P, --password        Specify if a password is required to connect to MySQL.
  -n NAME_TAG, --nametag NAME_TAG
                        Name of the tag used in the MDL SDF file to define the
                        name of the chemical compound.
  -l LOG_FILE, --logfile LOG_FILE
                        Name of the log file to send logging output to.
  -p PREFIX, --prefix PREFIX
                        Prefix added to the default table names.
  -a, --append          Specify if the data should be added to the existing
```

(continues on next page)

(continued from previous page)

```
                        mychem tables.
  -r, --replace         Specify if the new data should replace existing data.
  -V, --verbose         Enable verbose debug messages.
```

The first step is to create a database for storing the chemical tables. In this documentation, the database will be named *mychem*, but any other name can be used (the `-D` option permit to set the database name).

```
--
-- Database creation
--

CREATE DATABASE `mychem`;
```

Once the database is created, it is possible to load the MDL SDF file with the `mychemdb_manager` script:

```
$ python mychemdb_manager -D mychem -H localhost -U user -P database.sdf
Enter MySQL password:
The MDL SDFFile has been successfully loaded.
```

The previous command will create the following tables:

- *mychem_compounds* - It contains the compound's name and two timestamps (when the entry is created and when the entry is updated).
- *mychem_1D_structures* - It contains the 1D representation of the compounds (SMILES and InChI code).
- *mychem_3D_structures* - It contains the 3D structure of the compounds in MDL Molfile format.
- *mychem_bin_structures* - It contains the binary representation (fp2 and obserialized object) of the compounds.

  **Note**

  It is possible to use another prefix than *mychem* by using the `-t` option.

If all these tables are already existing, you have to choose either to append data (`-a` option) or to replace existing data (`-r` option).

CHAPTER

EIGHT

# MOLECULE FORMATS

The *Molecule Formats* appendix describes the file formats supported by the Mychem software. Further informations about chemical file formats are available on Wikipedia.

## 8.1 Serialized OBMol

The serialized OBMol format is a bit string obtained by serializing an OBMol object. This type of string stores most of the data contained in a OBMol object, with exception to 2D and 3D coordinates. The binary structure of this string is described on the ChemiSQL Website.

## 8.2 CML

The Chemical Markup Language (CML) is an open standard for representing molecular structures, as well as many other types of chemical data. It is based on the XML language and can be simply processed by any XML parser. The CML Project Website, hosts the XML Schema and source codes for parsing and working with CML data.

## 8.3 Fingerprints

This section presents fingerprint types used by the Mychem software. Further details about fingerprints are given in an article written by Andrew Dalke. In short, fingerprints are a bit string representation of a molecule. Most of them can be classified in two categories:

- Structural fingerprint - This fingerprint type is based on substructure features.
- Hash fingerprints - This fingerprint type is a hash of the representation of a molecule. It is used most often in similarity searching, with the hypothesis that two similar compounds create similar fingerprints, and that two similar fingerprints means the compounds are similar.

### 8.3.1 FP2

FP2 fingerprints index small molecule fragments based on linear segments of up to 7 atoms in length. They are hash fingerprints. The specification of the FP2 fingerprints is available on the Open Babel Website.

### 8.3.2 FP3

FP3 fingerprints index small molecule fragments based on a list of SMART patterns. They are hash fingerprints. The SMART patterns are listed in the file named `patterns.txt`, that is distributed with the Open Babel software. The specification of the FP3 fingerprints is available on the Open Babel Website.

### 8.3.3 FP4

FP4 fingerprints index small molecule fragments based on a list of SMART patterns. They are hash fingerprints. The SMART patterns are listed in the file named `SMARTS_InteLigand.txt`, that is distributed with the Open Babel software. The specification of the FP4 fingerprints is available on the Open Babel Wiki.

## 8.4 InChI

The IUPAC International Chemical Identifier (InChI) is an identifier for chemical substances that can be used in printed and electronic data sources. It was developed under IUPAC Project 2000-025-1-800. Details of the project are available from the IUPAC Website.

## 8.5 Sybyl Mol2

The Sybyl Mol2 format is a complete, portable representation of a molecule. It is an ASCII file that contains structural data as well as Sybyl related data (Sybyl is a chemoinformatics software released by Tripos). The file format is described on the Tripos Website.

## 8.6 MDL Molfile

A MDL Molfile is a file format created by MDL for holding data about the atoms, bonds, connectivity and coordinates of a molecule. This file format consists of some header information, the Connection Table (CT) containing atom info, then bond connections and types, followed by sections for more complex information. The format is described on the MDLI Website. There is two versions of MDL Molfile:

- V2000
- V3000

### 8.6.1 V2000

It is the current standard. However, it presents a limitation, as it support up to 999 atoms or bonds.

### 8.6.2 V3000

The V3000 format has been created to support more than 999 atoms or bonds. It can be useful for describing proteins and polymers.

## 8.7 PDB

The Protein Data Bank Format is commonly used for proteins and biological macromolecules. It was originally designed as a fixed-column-width format and thus officially has a built-in maximum number of atoms; however, many tools can read files that exceed the limit. Some PDB files contain an optional section describing atom connectivity as well as position. Further informations about this format can be retrieved from the PDB Website.

## 8.8 SMILES

The Simplified Molecular Input Line Entry Specification (SMILES) format is a linear text format which can describe the connectivity and chirality of a molecule. It does not include 2D or 3D coordinates and hydrogens atoms are not represented. The SMILES are described on the Daylight Website. However, a complete SMILES standard does not exist. Craig James is leading a campaign to develop an Open Standard for SMILES. The discussion is taking place under the umbrella of the Blue Obelisk Group.

### 8.8.1 Canonical SMILES

Canonical SMILES is a particular type of SMILES, that has a single *canonical* form for any particular molecule, regardless of atom order.