

# COMPUTATIONAL MATERIALS REPOSITORY

David Landis

CMR Workshop (CAMD specific)

Practical hints & information for CMR users

# Overview

About db/cmr-files

CMR on Niflheim

Access from outside/wireless

3 ways to create cmr-files

Working with CMR

Examples (Groups)

# A note about db-files and cmr-files

db-files and cmr-files are exactly the same file types. It's just a *different* name. The reason is historic: first we called them db-files (database files), but many others call their file format db-files. Therefore we decided to call them cmr-files.

→ cmr-files is the official names. db-files is the old name. You will find the name *db-files* still in the documentation, but with time it will be completely renamed.

# CMR on Niflheim

To profit from the latest features use the newest ASE3, GPAW and CMR. (CMR runs with previous version, just not all features might be available):

```
[$> module purge]
$> module load ASE3
$> module load GPAW
# Please use the following line until further
# notice:
$> module load CMR/580-1.el5.fys.python2.4
[$> module load CMR]
```

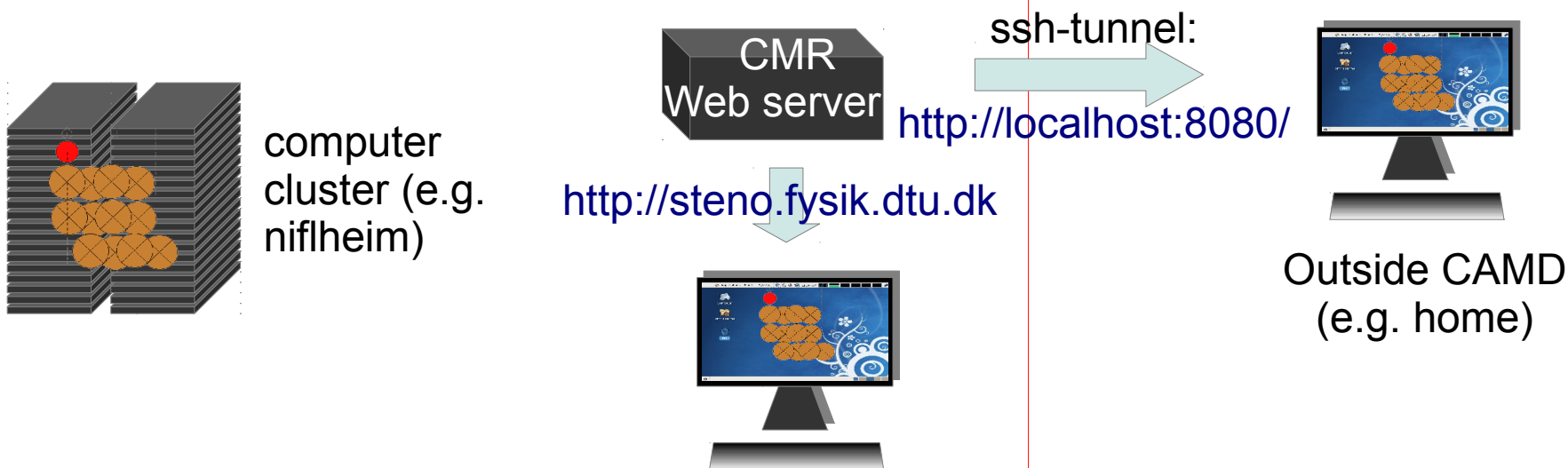
More here:

[https://wiki.fysik.dtu.dk/niflheim/Installed\\_software](https://wiki.fysik.dtu.dk/niflheim/Installed_software)

# Accessing the internal database

If you would like to access the internal CAMD database from **inside** you can just visit <http://steno.fysik.dtu.dk>. If you are outside, e.g. at home or through DTU wireless, you need to open a **ssh-tunnel** (see following slides).  
Please note: this instruction is designed for CAMD members.

## CAMD Intranet



# Accessing the internal database

- Mac/Linux:

```
ssh xxx@demon.fysik.dtu.dk -L 8080:steno.fysik.dtu.dk:80
```

(xxx is your Niflheim username.)

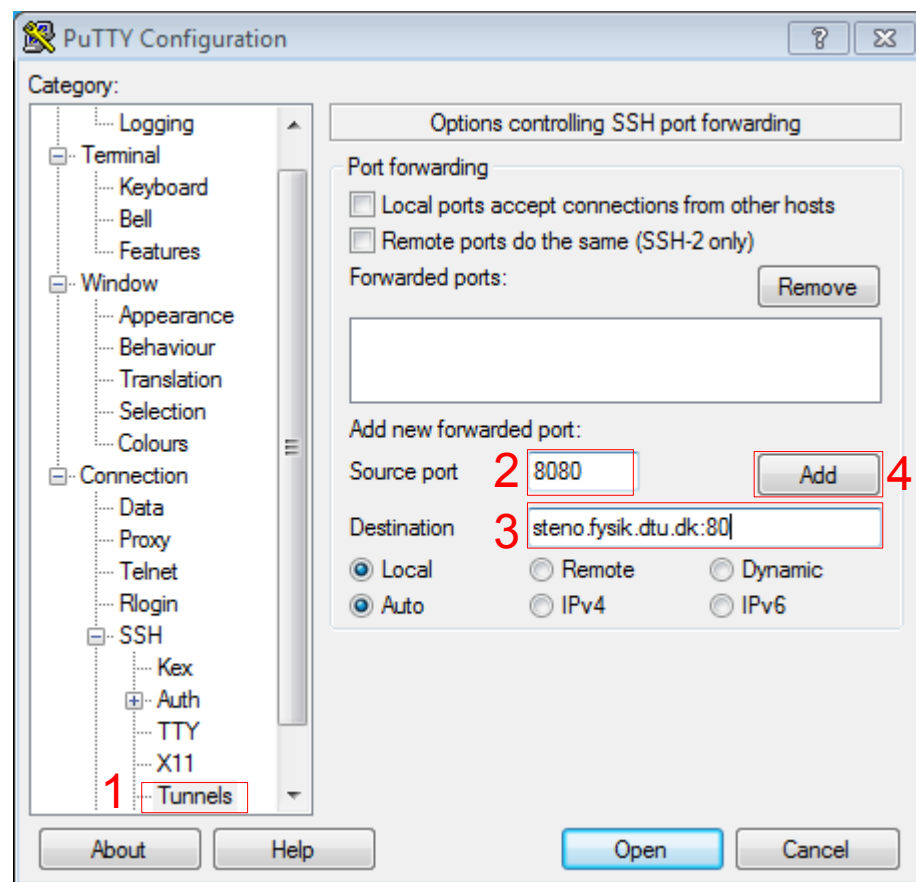
# Accessing the internal database

- Windows:

1. Download and install putty (putty-0.62-installer.exe)

<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>

2. Define a tunnel:  
(follow these instructions)

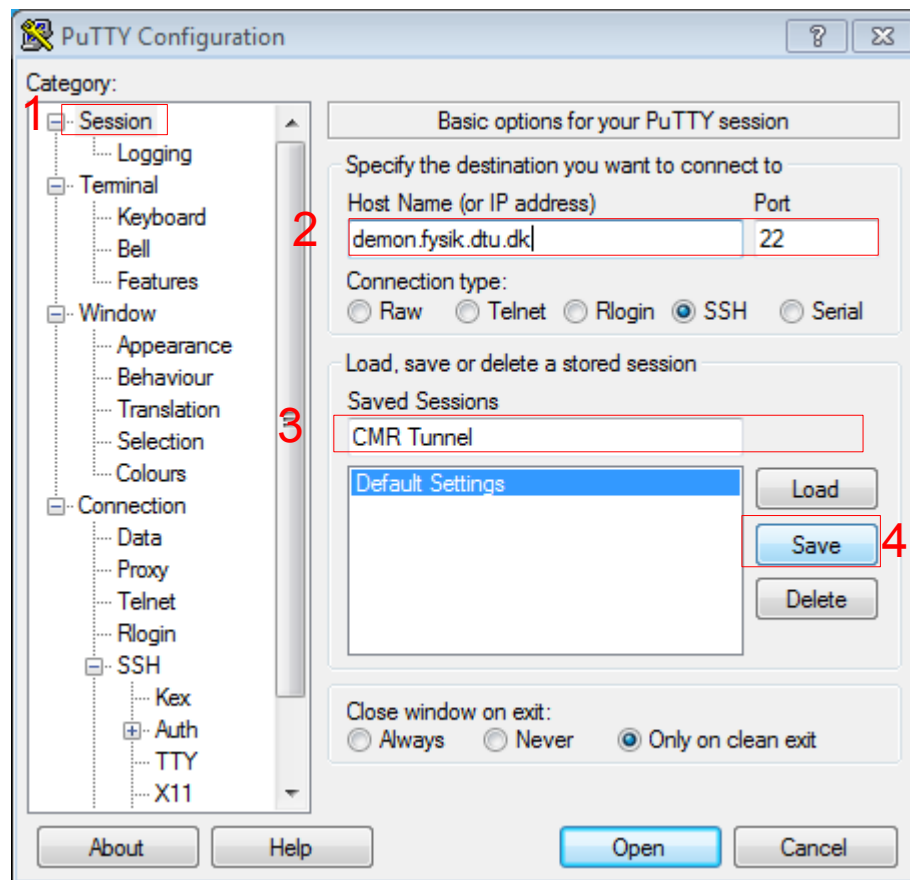


# Accessing the internal database

Windows:

3. Store connection parameter

(After the connection parameters are stored. The next time you only have to perform step 4 and 5.



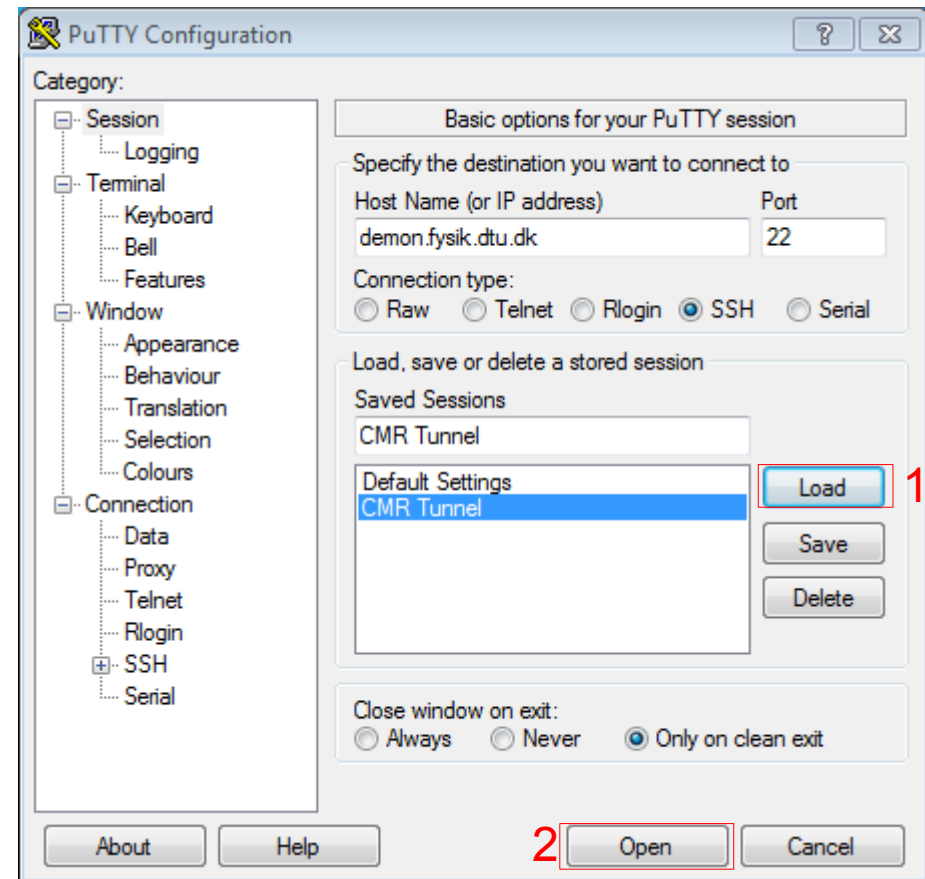


# Accessing the internal database

Windows:

4. Load the connection parameters and connect

5. Open <http://localhost:8080> in your favorite webbrowser



# Working with CMR

You can either work in your home directory (slower, no access from internet but nobody can see your data) or with the internal database (faster, but others can see your data).

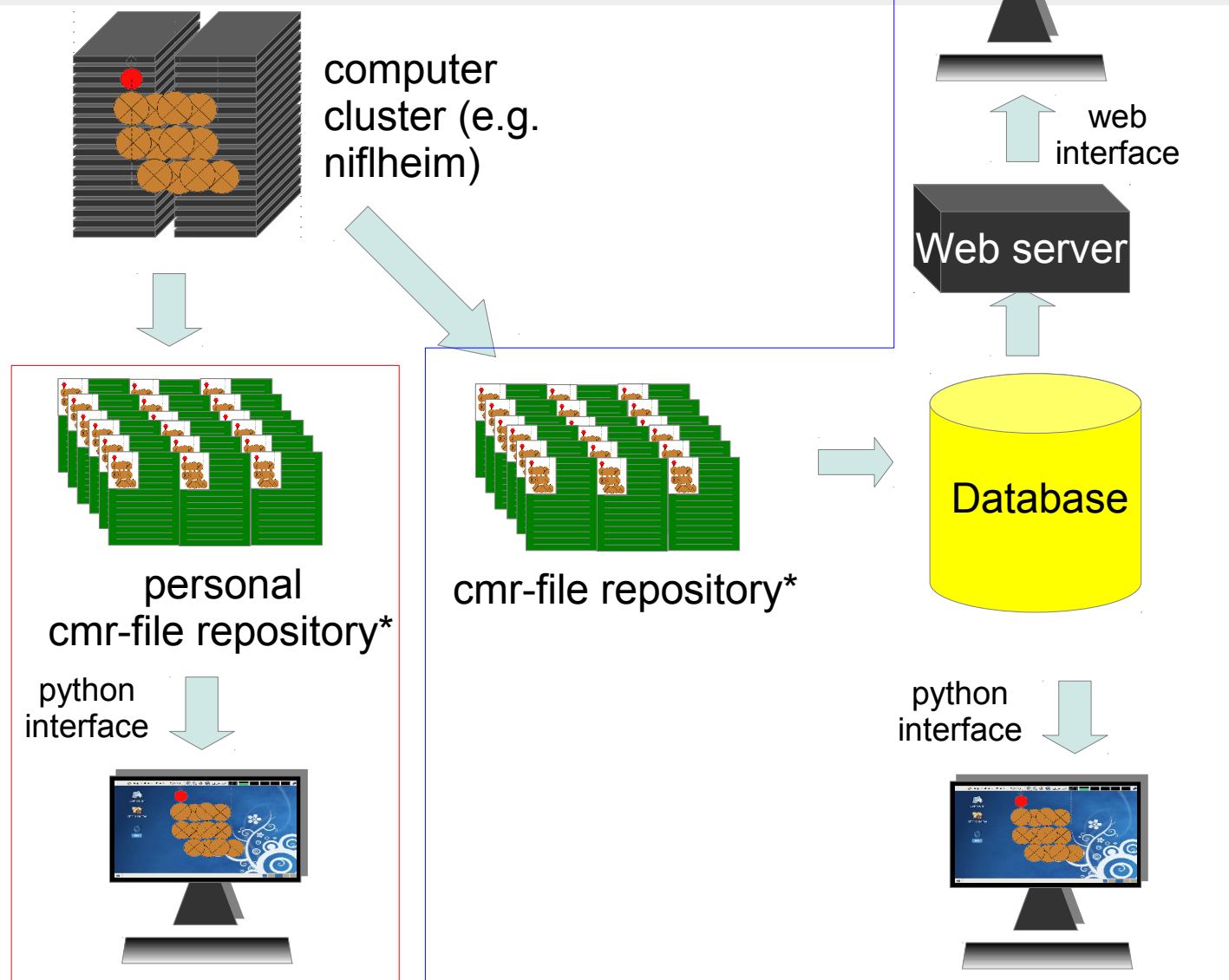
# Working with CMR

2 options:

work in your  
home directory  
(without  
database)

work with  
database

work with web  
interface



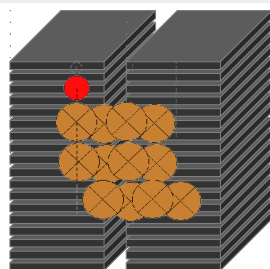
# Working with CMR

2 options:

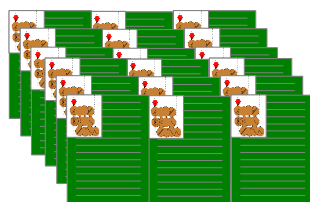
work in your  
home directory  
(without  
database)

Only me can see  
and query the  
data.

work with web  
interface

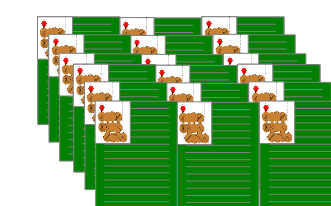
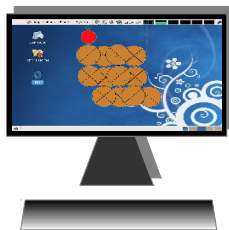


computer  
cluster (e.g.  
niflheim)

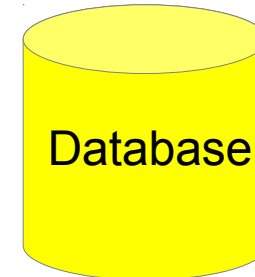


personal  
cmr-file repository\*

python  
interface

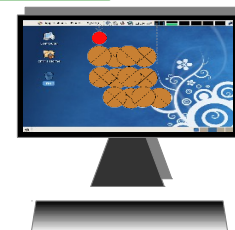


cmr-file repository\*

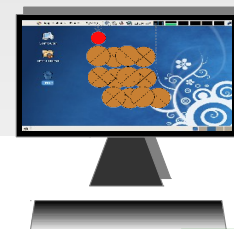


Database

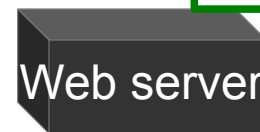
python  
interface



All in my institute  
can see and query my  
data.\*\*



web  
interface



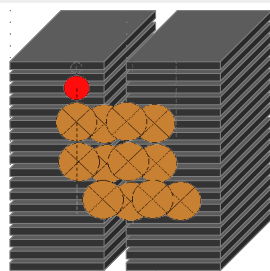
# Working with CMR

2 options:

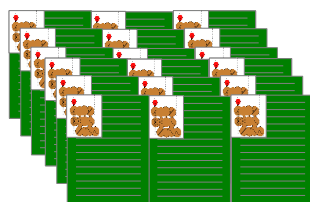
work in your  
home directory  
(without  
database)

Only me can see  
and query the  
data.

work with web  
interface

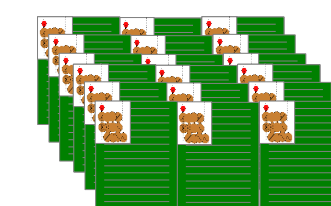
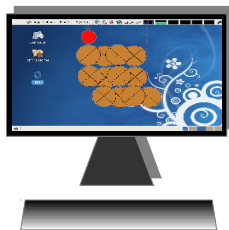


computer  
cluster (e.g.  
niflheim)

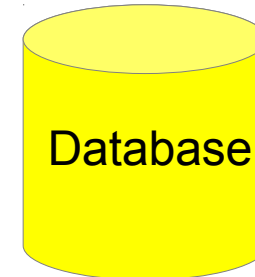


personal  
cmr-file repository\*

python  
interface

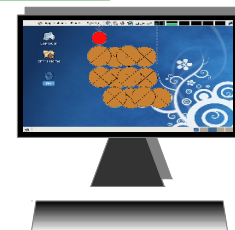


cmr-file repository\*

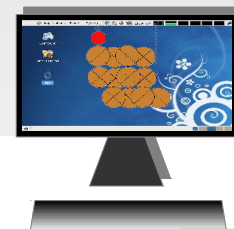


Database

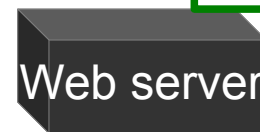
python  
interface



All in my institute  
can see and query my  
data.\*\*



web  
interface



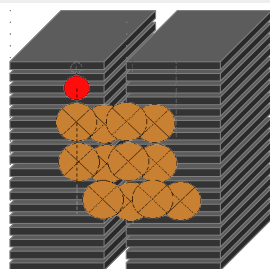
# Working with CMR

2 options:

work in your  
home directory  
(without  
database)

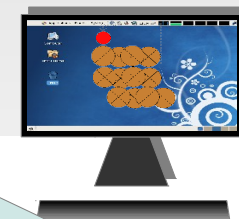
Queries are  
generally slower.

work with web  
interface



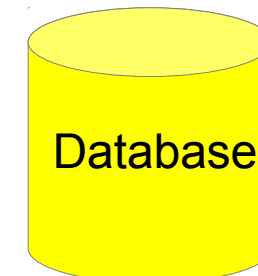
computer  
cluster  
niflheim

Get script template  
for tasks from web  
interface.

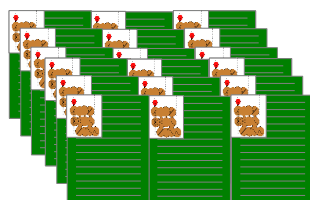


web  
interface

Web server

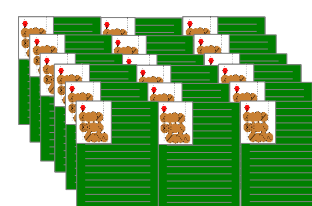
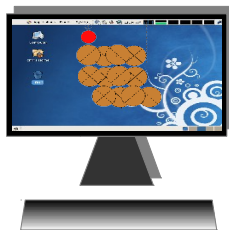


Database



personal  
cmr-file repository\*

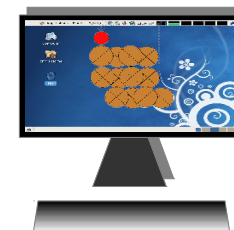
python  
interface



cmr-file repository\*



python  
interface



Queries are generally  
faster.

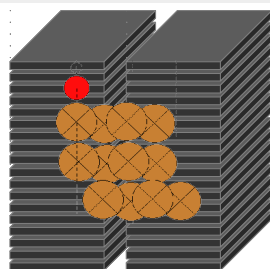
# Working with CMR

How to select the repository

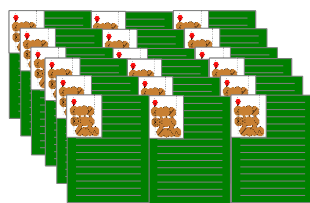
2 options:

work in your  
home directory  
(without  
database)

work with

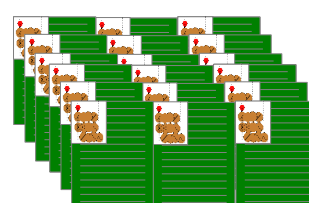


computer  
cluster (e.g.  
niflheim)



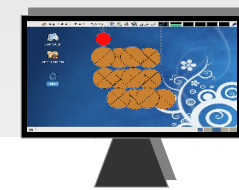
personal  
cmr-file repository

python  
interface

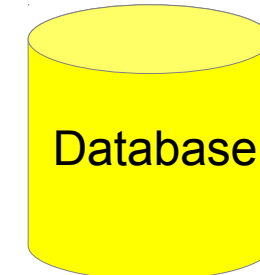
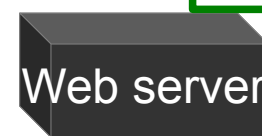


cmr-file repository

python  
interface



web  
interface



environment variable:

```
CMR_REPOSITORY=${HOME}/repository
```

or in python:

```
import cmr  
cmr.set_repository_path("${HOME}/repository")
```

environment variable\*:

```
CMR_REPOSITORY=/home/niflheim/repository/db
```

or in python:

```
import cmr  
cmr.set_repository_path("/home/niflheim/repository/db")
```

## Three ways to create a cmr-file

1. With the command line interface
2. With a python script
3. Direct write to with ASE/GPAW



# Three ways to create a cmr-file

## 1. Command Line Interface (CLI)

(Please note: there is a bug in the current release, so this will not work in all cases yet.)

```
cmr --convert slab.traj
```

will create a db-file called slab.db.

You can add keywords and fields with:

```
cmr --modify slab.db
```

<https://wiki.fysik.dtu.dk/cmr/cmr-tutorial/command-line.html>

# Three ways to create a

## 2a. Create `example.cmr` with a Python script:

```
import cmr
params = { "input": "example.gpw",
           "output": "example.cmr",
           "db_keywords": ["gaussian",
                           "example",
                           "test"],
           "db_description": "Test check-in",
           "scripts": ["Al-fcc.py"],
           "files": ["Al-fcc.txt"],
           "vacuum": 4
        }
```

```
cmr.convert(params)
```

Works also with  
.csv: CSV  
.gpw: GPAW  
.nc: Dacapo  
.out: Gaussian  
.xml: VASP  
.traj: ASE trajectories

output name

keywords

textual description

attach the script

attach a file

user-defined field  
(you can add as many  
as you want!)

More here: [https://wiki.fysik.dtu.dk/cmr/cmr-tutorial/db\\_files.htm](https://wiki.fysik.dtu.dk/cmr/cmr-tutorial/db_files.htm)

# Three ways to create a

## 2b. Write converted file directly to db/cmr-file repository

```
import cmr
params = { "input": "example.gpw",
           "db": True,
           "db_keywords": ["gaussian",
                           "example",
                           "test"],
           "db_description": "Test check-in",
           "vacuum": 4
        }

cmr.convert(params)
```

Works also with  
.csv: CSV  
.gpw: GPAW  
.nc: Dacapo  
.out: Gaussian  
.xml: VASP  
.traj: ASE trajectories

copy to cmr-repository\*

keywords

textual description

user-defined field  
(you can add as many as you want!)

More here: [https://wiki.fysik.dtu.dk/cmr/cmr-tutorial/db\\_files.htm](https://wiki.fysik.dtu.dk/cmr/cmr-tutorial/db_files.htm)

# Three ways to create a cmr-file

## 3a. Direct write with GPAW:

```
calc = GPAW(...)
```

```
params = {"keywords": ["gpaw", "exercise"],  
          "scripts": ["Al-fcc.py"],  
          "files": ["Al-fcc.txt"]}
```

```
calc.write(".cmr", cmr_params=params)
```



write to cmr-repository  
(alternatively put a file-name)

# Three ways to create a cmr-file

## 3b. Direct write with ASE:

```
from ase.structure import molecule
from ase.io import write

system = molecule("H2O")

cmr_params = {
    "db_keywords": ["H2O"],
    "db_description": "Water"
}

write("H2O.cmr" system, cmr_params=cmr_params)
```

More here: <https://wiki.fysik.dtu.dk/cmr/cmr-tutorial/ase.html>

# Working with CMR

## How to query the repositories

### personal repository:

```
import cmr
from cmr.ui import DirectoryReader

user="john" #the user we would like to retrieve data
from
columns = ["db_date", "db_user", "db_keywords"]

reader = DirectoryReader("/home/.../repository")

collection = reader.find(name_value_list=["db_user",
user]), columns=columns)

collection.print_table(20, columns)
```

### database:

```
import cmr
from cmr.ui import DBReader

user="john" #the user we would like to retrieve data
from
columns = ["db_date", "db_user", "db_keywords"]

reader = DBReader("default")

collection = reader.find(name_value_list=["db_user",
user]), columns=columns)

collection.print_table(20, columns)
```

[https://wiki.fysik.dtu.dk/cmr/cmr-tutorial/working\\_db\\_files.html](https://wiki.fysik.dtu.dk/cmr/cmr-tutorial/working_db_files.html)

# Working with CMR

How to query the repositories

## personal repository:

```
import cmr
from cmr.ui import DirectoryReader

user="john" #the user we would like to retrieve data
from
columns = ["db_date", "db_user", "db_keywords"]

reader = DirectoryReader("/home/.../repository")

collection = reader.find(name_value_list=["db_user",
user]), columns=columns)

collection.print_table(20, columns)
```

## database:

Use a different database by choosing a different name. You'll be asked for the credentials 1 time on the command line

```
like to retrieve data
columns = ["db_date", "db_user", "db_keywords"]

reader = DBReader("default")

collection = reader.find(name_value_list=["db_user",
user]), columns=columns)

collection.print_table(20, columns)
```

More on find:

[https://wiki.fysik.dtu.dk/cmr/cmr-tutorial/working\\_db\\_files.html](https://wiki.fysik.dtu.dk/cmr/cmr-tutorial/working_db_files.html)

# Working with CMR

How to query the repositories

Internal CAMD database:  
<http://steno.fysik.dtu.dk>

personal repository:

```
import cmr
from cmr.ui import
```

```
user="john" #the user
from
```

```
columns = ["db_user",
```

```
reader = Director
```

```
collection = reader.read(
user)], columns=
```

```
collection.print_table()
```

M

The screenshot shows the CMR web interface with the following elements:

- Navigation tabs: 1. Data set, 2. DB-Filters, 3. Enhancement, 4. Filters, 5. Visualization.
- Subtabs under '1. Data set': 1a. Default, 1b. LP.
- Instructions: "The data in this database is selected by writing queries into the field below. To make writing queries simpler, you can use the tabs above which are numbered 1-5. The tab row below shows subtabs e.g. '1a Default', '1b. LP'. Click on the individual tabs to get more information about what they do. Generally it's good practise to start by restricting to data of a specific user e. g. db\_user=dlandis. In order to see what users exist switch to tab '2. DB-Filters' to select user and keywords."
- Query input field: Contains the text "db\_user=ivca".
- Query label: "Query:" with hints: "(Hint 1: use \* to search for partial keywords; Hint 2: use " for keyword with spaces)".
- Results per page: A dropdown menu set to "10".
- Buttons: "Execute" and "JSON".

e to retrieve data

```
db_keywords"]
```

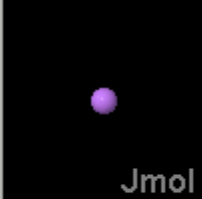









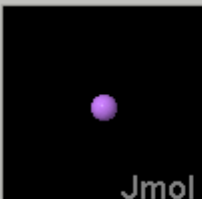









```
e_list=[("db_user",
```

[https://wiki.fysik.dtu.dk/cmr/cmr-tutorial/working\\_db\\_files.html](https://wiki.fysik.dtu.dk/cmr/cmr-tutorial/working_db_files.html)



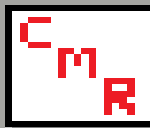



# Exercises

Internal CAMD database:  
<http://steno.fysik.dtu.dk>

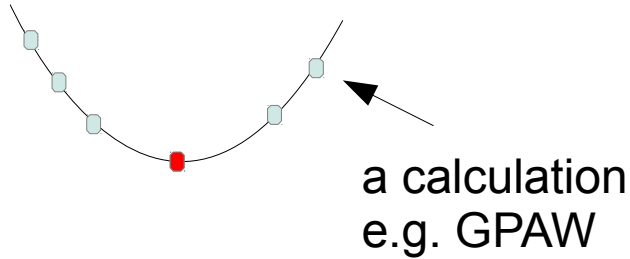
jmol	id_ref	db_date	atoms	db_description	downloads
 Jmol	<a href="#">764824</a>	2012-03-15 20:00:00	Li (2Li)	n/a	   
 Jmol	<a href="#">764825</a>	2012-03-15 20:00:10	Li (1Li)	my first project: Li2 atomize	   
 Jmol	<a href="#">764827</a>	2012-03-16 13:24:42	Li (2Li)		   
 Jmol	<a href="#">764828</a>	2012-03-16 13:24:48	Li (1Li)		   

## downloads



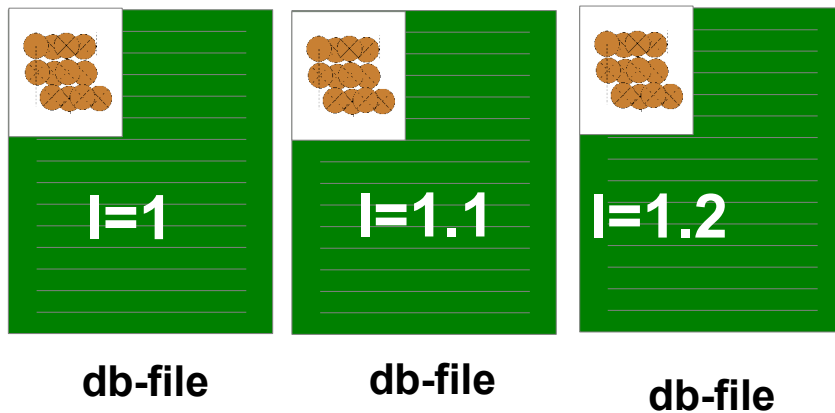
- download ASE-script (to get atom object )
- download CMR-script (to download cmr file) and other modify data

# Optimization Example



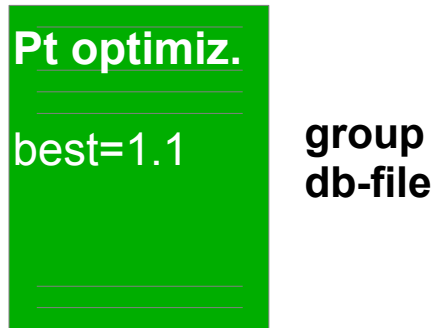
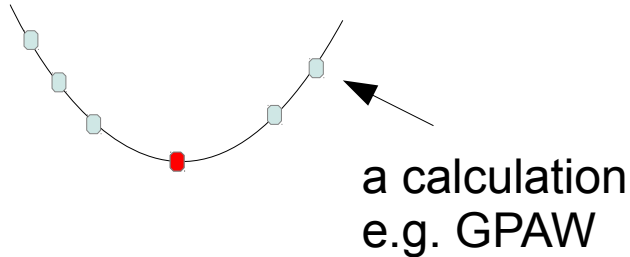
1. Create a group all at once

2. Add items one by one



<https://wiki.fysik.dtu.dk/cmr/cmr-manual/manual/groups.html>

# Optimization Example



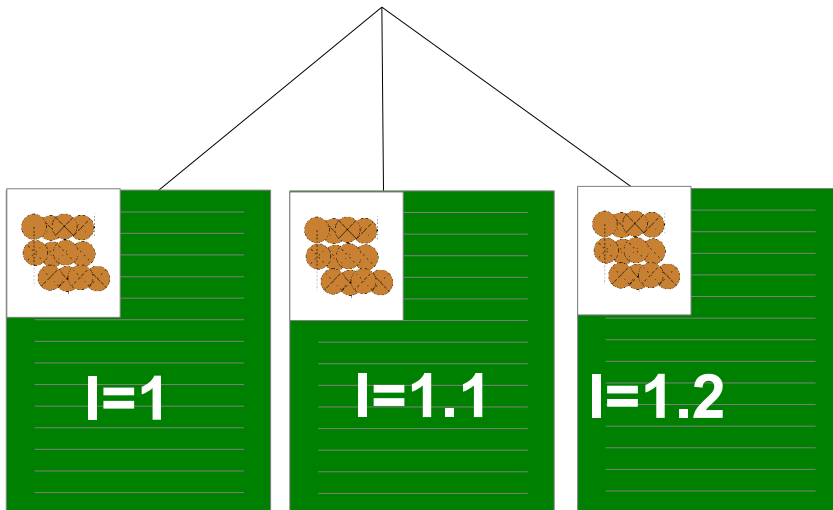
## 1. Create a group all at once

```
import cmr
data = cmr.create_group()

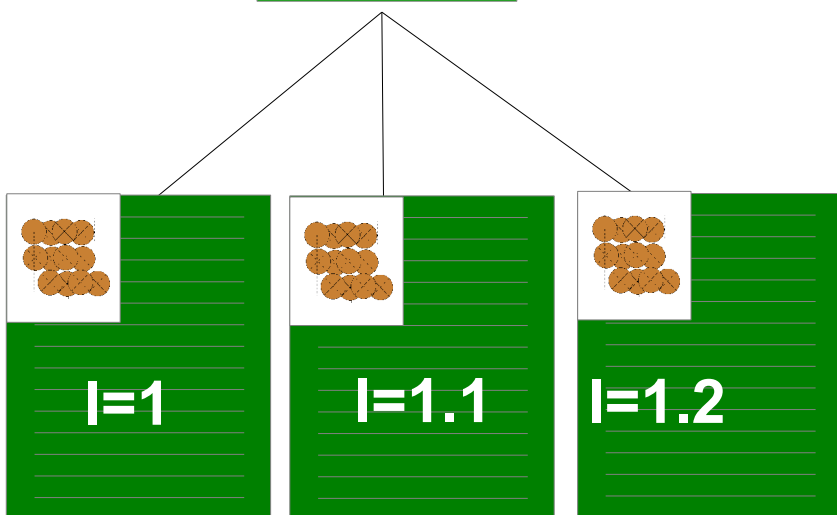
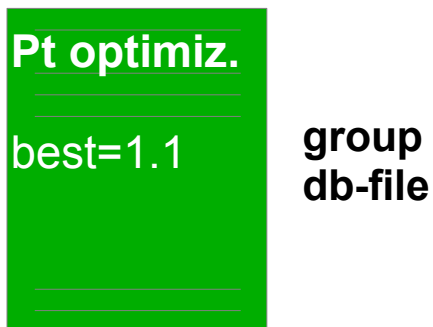
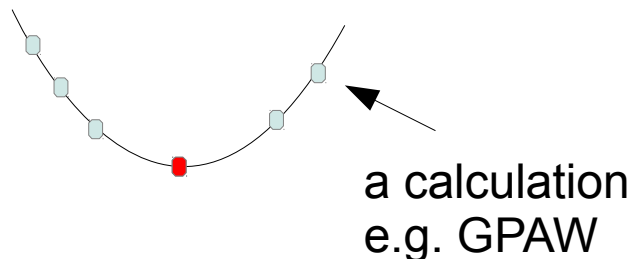
members = [list of file names]
for f in files:
    data.add(cmr.read(f).get_hash())

data.set_user_variable("best", 1.1)

data.write(".db")
```



# Optimization Example



## 2. Add items one by one

```
import cmr

if os.path.exists("all.db"):
    data = cmr.read("all.db")
else:
    data = cmr.create_group("group_name")

data.add(cmr.read("...").get_hash())

data.set_user_variable("best", 1.1)

data.write("all.db")
```

# Hydrogen Storage with CMR

References for alloy and decomposition energy:

db-file containing a dacapo calculation

$\text{KAg}(\text{BH}_4)_3$   
total energy=-2665.1  
weight %=12.1  
structure=to

db-file containing a **group** calculation

$\text{KAg}(\text{BH}_4)_3$   
total energy=-2665.1  
weight %=12.1  
structure=to  
keywords=complete  
allow\_energy=...  
decomp\_energy=...

1<sup>st</sup> set of  
reference energies

Ag

K

H4BAg

H4K4

H8K2B2

2<sup>nd</sup> set of  
reference energies

Ag

K

H4BAg

H4K4

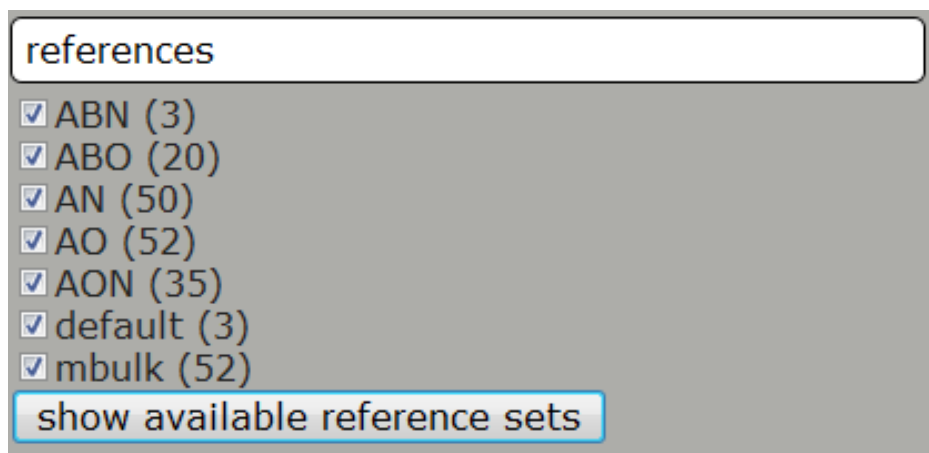
H8K2B2

Always make groups when adding  
*derived* data to a cmr-file. You might  
later use a different set of reference  
energies for example.

# Freezing Results

Research goes on – how can we get the results at the time of publishing?

at time of publishing:

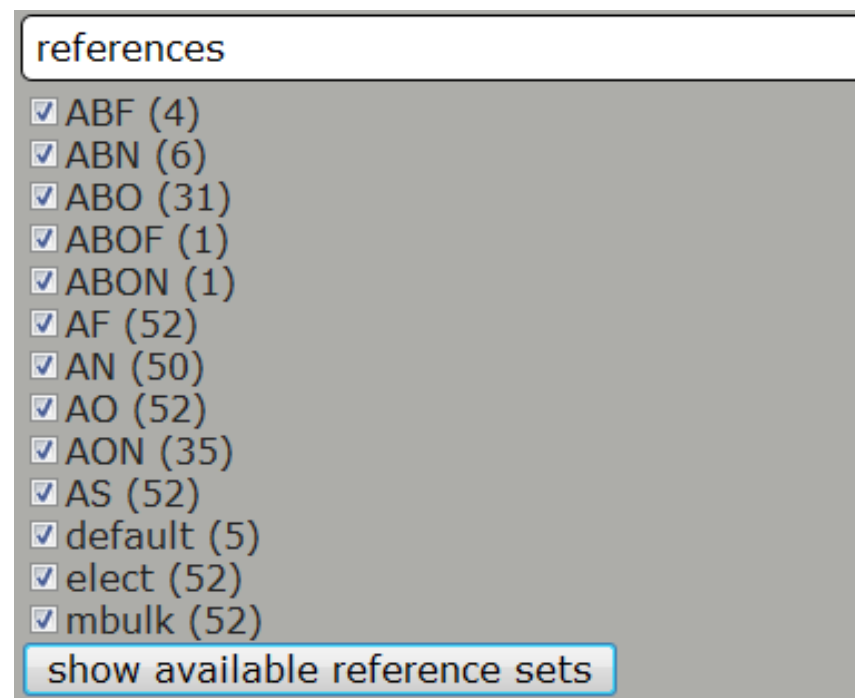


A screenshot of a web interface. At the top, there is a white input field containing the text "references". Below this field is a list of reference sets, each preceded by a checked checkbox. The list includes: ABN (3), ABO (20), AN (50), AO (52), AON (35), default (3), and mbulk (52). At the bottom of the list is a button with the text "show available reference sets".

reference set	count
<input checked="" type="checkbox"/> ABN	3
<input checked="" type="checkbox"/> ABO	20
<input checked="" type="checkbox"/> AN	50
<input checked="" type="checkbox"/> AO	52
<input checked="" type="checkbox"/> AON	35
<input checked="" type="checkbox"/> default	3
<input checked="" type="checkbox"/> mbulk	52

show available reference sets

now



A screenshot of a web interface. At the top, there is a white input field containing the text "references". Below this field is a list of reference sets, each preceded by a checked checkbox. The list includes: ABF (4), ABN (6), ABO (31), ABOF (1), ABON (1), AF (52), AN (50), AO (52), AON (35), AS (52), default (5), elect (52), and mbulk (52). At the bottom of the list is a button with the text "show available reference sets".

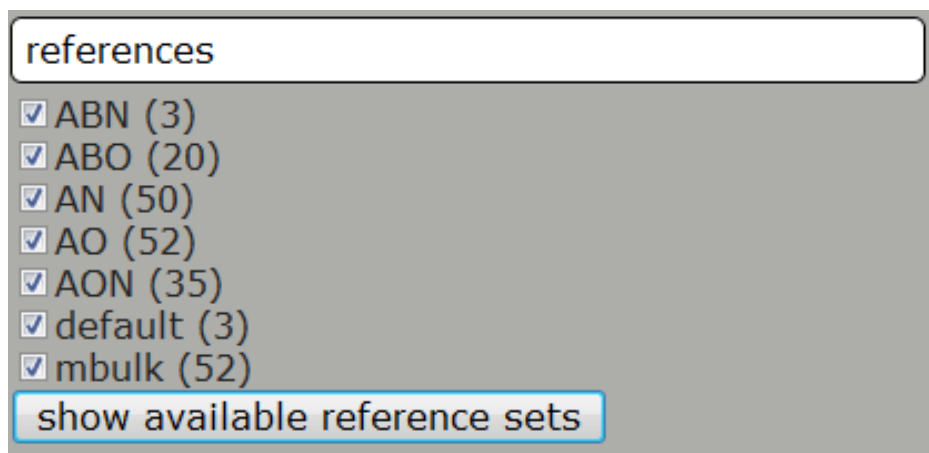
reference set	count
<input checked="" type="checkbox"/> ABF	4
<input checked="" type="checkbox"/> ABN	6
<input checked="" type="checkbox"/> ABO	31
<input checked="" type="checkbox"/> ABOF	1
<input checked="" type="checkbox"/> ABON	1
<input checked="" type="checkbox"/> AF	52
<input checked="" type="checkbox"/> AN	50
<input checked="" type="checkbox"/> AO	52
<input checked="" type="checkbox"/> AON	35
<input checked="" type="checkbox"/> AS	52
<input checked="" type="checkbox"/> default	5
<input checked="" type="checkbox"/> elect	52
<input checked="" type="checkbox"/> mbulk	52

show available reference sets

# Freezing Results

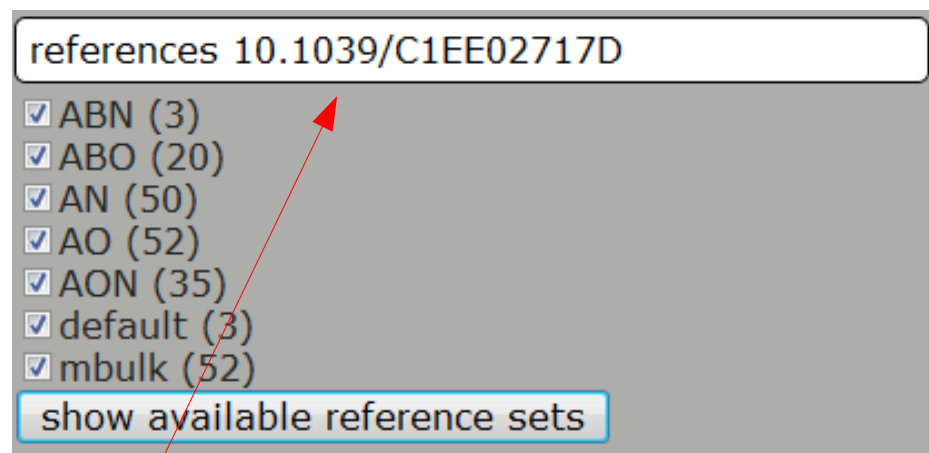
Research goes on – how can we get the results at the time of publishing?

at time of publishing:



A screenshot of a web interface for searching references. At the top is a search bar containing the word "references". Below the search bar is a list of reference sets, each with a checked checkbox and a count in parentheses: ABN (3), ABO (20), AN (50), AO (52), AON (35), default (3), and mbulk (52). At the bottom of the list is a button labeled "show available reference sets".

now



A screenshot of the same web interface as before, but the search bar now contains "references 10.1039/C1EE02717D". A red arrow points from a text box below to the DOI in the search bar. The list of reference sets and the "show available reference sets" button are the same as in the previous screenshot.

add the DOI as keyword to all data at the time of publishing and use it in your queries!

# Querying

## Querying with the (cmr) python interface

1) connect



2) data selection  
with keywords & restrictions



3) local queries  
(in your computers memory)

**database:**

```
import cmr
from cmr.ui import DBReader
from cmr.ui.operators import *
```

```
user="john"
```

```
1 { reader = DBReader("default")
2 { collection = reader.find(name_value_list=[("db_user", user)])

# in your computer's memory:
3 { col = collection.select("delta_E_decomp", 0,
                           Operator("<"), True)
    col.sort("delta_E_decomp")
    col.print_table(20, columns)
```



# Querying

## Querying with the (cmr) python interface

1) connect



2) data selection  
with keywords & restrictions



3) local queries  
(in your computers memory)

if the columns are not  
specified the whole db-file  
is downloaded (slow).  
→ only request the data  
that you would like to use  
in the future

database:

```
import cmr
from cmr.ui import DBReader
from cmr.ui.operators import *
```

```
user="john"
columns = ["db_date", "db_user", "db_keywords"]
```

```
1 { reader = DBReader("default")
2 { collection = reader.find(name_value_list=["db_user", user],
                                columns=columns)

    # in your computer's memory:
3 { col = collection.select("delta_E_decomp", 0,
                            Operator("<"), True)
    col.sort("delta_E_decomp")
    col.print_table(20, columns)
```

# Extra Slides

# Browse multiple db-files (1)

```
from cmr.ui import DirectoryReader
from cmr.io import READ_XML

collection = DirectoryReader(directory, READ_XML)
#or
collection = DBReader.find(...)

print "Available keys: ",
print collection.keys()

columns = ["e_pot", "keywords", "user", "date"]

# get only calculations done by dlandis
collection = collection.select("user" ,"dlandis")

collection.print_table(20, columns)
```