LHSG-CT-2003-503420

BioXHIT

A European integrated project to provide a highly effective technology platform for Structural Genomics.

Life Sciences, Genomics and Biotechnology for Health

WP5: De5.2.5 Specification of version 1 of the Project Tracking Database Design

Due date of deliverable: **31.12.2006**
Actual submission date: **14.02.2007**

Start date of project: **1.1.2004** **Duration: 60 months**

Organisation name of lead contractor for this deliverable: CCP4-CCLRC
Daresbury Laboratory, Warrington, Cheshire WA4 4AD, UK.
Author Peter Briggs and Wanjuan Yang

Specification of Version 1 of the Project Tracking Database Design

1 Introduction

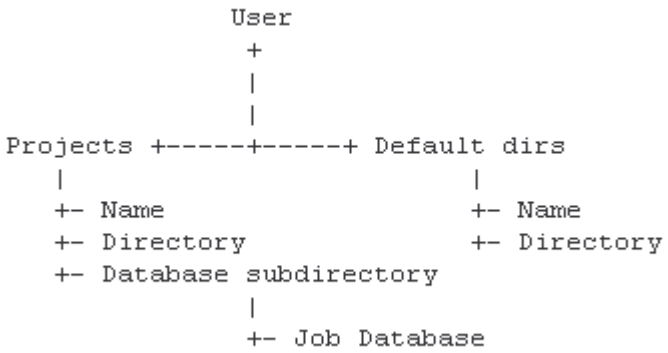
This document outlines the specification of the CCP4i project tracking database. It is based on the project database used in CCP4i and is geared towards tracking runs of software initiated through CCP4i. Later revisions of the schema will be more generic to allow a better description of non-CCP4i initiated software runs.

The sections below contain the following information: section 2 gives an abstract description of the tracking database schema; sections 3 and 4 describe attempts at concrete implementations of the schema in CCP4i def file format and SQL tables respectively.

2 Description of the tracking database schema

2.1 Overview

CCP4i divides the user's work into **projects**. A user can have as many projects as they wish. The schematic diagram below gives a picture of how the data is imagined in a simple hierarchy.



2.2 Projects

A **project** in this model consists very simply of a file directory or folder plus an associated job database which is used to store information about program runs. There is no formal definition of what constitutes a project beyond this, and users are free to define new projects to subdivide their work and organise their data as they wish.

A project has the following attributes:

- A *project name* (also called an alias)

- An associated *project directory*
- A *database subdirectory* with the project directory (conventionally called CCP4_DATABASE)
- A *job database*

The job database contains job records. A job is a run of a software application or task, which can be a single program or a script that runs many programs in some sequence. The data items associated with each each job record are given in the tables below in the section on *Job Records*.

In addition to none or more job records, the job database also has a data item called NJOBS. This is actually the current highest job id number in the database, and not necessarily the total number of jobs.

2.3 Default Directories ("Def Dirs")

A **default directory** (also called a "def dir") has name (also called an "alias") and an associated directory, but no database subdirectory or job database.

A normal use of a def dir is as a convenient way to refer to a directory containing source data.

2.4 Job records

A job represents the run of a software application. Typically this is a task in CCP4i, however in principle it could be a run of any program or script. The following sections describe the data that are with each job in the schema.

2.4.1 Data associated with job records

Each job has a number of associated data items that are described in the following table.

Name	Description	Type
JOB_ID	The job number is a unique integer value assigned to each job in the project's job database, starting from 1 and increasing monotonically. It is used as the primary key for identifying records within the job database.	Positive integer
DATE	The date of the last change to the job record, in "epoch" format i.e. the number of seconds that have elapsed since the system's (arbitrary) start time.	Date (positive integer)
TASKNAME	The name of the CCP4i task or other application which created the job when it ran.	Text (no whitespace)

STATUS	The current state of the job, indicating that it is either in progress, or the broad condition under which it terminated. The status of a job indicates the state of the job process, rather than indicating the quality of the output. See below for the possible values and their meanings.	Enumerated type
TITLE	A descriptive user-defined title for the job run. This provides simple annotation of the job records to help to distinguish between different runs of the same task.	Text
LOGFILE	The name of the principle logfile associated with the job. Only the filename is specified, the file is assumed to be held in the project directory. Typically CCP4i constructs the logfile names for a particular job using the following convention: [JOB_ID]_[TASKNAME].log However this is a convention and not a general requirement for logfile naming.	Filename
RUNFILE	The name of a "control file" that was used to run the job, for example a shell script. It is possible for a job not to have the RUNFILE set; it currently used when the user adds the report of a jobs run from outside CCP4i.	Filename
SCRATCH	The name of the temporary or "scratch" directory for the job	Directory
INPUT_FILES	A list of the names of files that were used as input to the job. Items in the list are separated by whitespace characters, and can either be filenames without a path, in which case they are assumed to reside in one of the project or def dir directories (specified in the INPUT_FILES_DIR), or alternatively the file may be specified by the full path. See section 2.4.3 for details of how the filenames and directories are stored and recovered.	List of filenames
INPUT_FILES_DIR	A list of the project or def dir names corresponding to the locations of the files in the INPUT_FILES list. Items in the list are separated by	List of directory specifiers

	whitespace characters. See section 2.4.3 for details of how the filenames and directories are stored and recovered.	
INPUT_FILES_STATUS	A list of "file status" signifiers for each file in the INPUT_FILES list. This is not used and the signifiers are currently undefined.	List of text
OUTPUT_FILES	A list of the names of files that were created as output from the job. Items in the list are separated by whitespace characters, and can either be filenames without a path, in which case they are assumed to reside in one of the project or def dir directories (specified in the OUTPUT_FILES_DIR), or alternatively the file may be specified by the full path. See section 2.4.3 for details of how the filenames and directories are stored and recovered.	List of filenames
OUTPUT_FILES_DIR	A list of the project or def dir names corresponding to the locations of the files in the OUTPUT_FILES list. Items in the list are separated by whitespace characters. See section 2.4.3 for details of how the filenames and directories are stored and recovered.	List of directory specifiers
OUTPUT_FILES_STATUS	A list of "file status" signifiers for each file in the OUTPUT_FILES list. This is not used and the signifiers are currently undefined.	List of text
PARAMETER_FILE	The name of the CCP4i def-format file which contains the parameters and values used to run a CCP4i task. It is possible for jobs not to have an associated parameter file, for example reported jobs.	Filename
NOTEBOOK_FILE	The name of an ASCII text file with free text typically used to store annotation or notes on the job. It is possible for jobs not to have an associated notebook file, as these files are only created by an application when adding a new notebook entry.	Filename

Note that the implementations described in the later sections of this document do not necessarily conform to this description at present.

2.4.2 Job Status Values and Meanings

The table gives the possible values for the STATUS data item in a job record:

Value	Description/meaning
STARTING	The job has been registered but the process is not yet running
RUNNING	The job process is currently running on the system
FINISHED	The job process terminated without raising any system errors or non-zero termination status
FAILED	The job process terminated with a system error or with a non-zero termination status
KILLED	The user initiated premature termination of the job process from within CCP4i
ON_HOLD	A special status used in auto-testing; the job is scheduled to begin at a later time, under some condition determined by CCP4i/client application.
REPORTED	The job process ran under some other system and the user has recorded the details retrospectively

2.4.3 Input and output filename storage and recovery

The information about references to input and output files associated with jobs are stored in the INPUT_FILES and INPUT_FILES_DIR data items for input files, and OUTPUT_FILES and OUTPUT_FILES_DIR for output files.

INPUT_FILES is a list of filenames and INPUT_FILES_DIR is a list of corresponding project or def dir names, so that the i'th project name in the INPUT_FILES list is associated with the i'th filename in the INPUT_FILES_DIR. This is the same for the OUTPUT_FILES and OUTPUT_FILES_DIR lists also.

For files that are stored in a directory that is also a project or def dir, usually only the name of the file is stored without the leading directory path, e.g.:

```
INPUT_FILES:     INPUT_FILES_DIR:
toxd.mtz         TOXD
```

The full path is constructed by acquiring the directory path for the project and then appending the filename, e.g.:

```
/home/pjx/Projects/toxd/toxd.mtz
```

For files that are not in a project or def dir directory, the full filename is recorded in the INPUT_FILES or OUTPUT_FILES list, and the corresponding name in the ..._DIR entry is given as either FULL_PATH or as an "empty" entry {}, e.g.:

```
INPUT_FILES:                    INPUT_FILES_DIR:
```

```
/home/pjx/NotAProject/toxd.mtz   FULL_PATH
```

Additionally, references to directories can also be stored. In this case the filename in the INPUT_FILES or OUTPUT_FILES list is an "empty" entry {} and the corresponding entry in the ..._DIR list is the full directory path, e.g.:

```
INPUT_FILES:     INPUT_FILES_DIR:
{}               /home/pjx/arbitrary/dir/
```

3 Implementation in CCP4i def file format

The above data model has been implemented in CCP4i and in dbCCP4i in the CCP4i def file format.

3.1 Overview of the CCP4i def file format

Def files use a simple flat file data storage format in ASCII text. A line in a def file will be one of:

- A CCP4i header line, beginning with the text `#CCP4I`,
- A parameter-value pair,
- A parameter-type-value triplet (nb in any given file there will either pairs or triplets for all parameters - these two forms are not normally mixed),
- An "include" directive, of the form `@filename` (which instructs the processor to insert the contents of the specified file at this point,
- A comment line, beginning with the hash symbol (#), or
- A blank line, consisting of white space only.

The parameter-value pair lines have the form:

```
PARAMETER        value
```

The parameter-type-value triplets have the form:

```
PARAMETER        type        value
```

The elements on each line are separated by whitespace. Parameter names are conventionally uppercased and consist of alphanumeric characters and underscores only. Values can be arbitrary strings or numbers, however they must be enclosed in double quotes if they contain whitespace.

(The types are defined in the $CCP4/etc/types.def file distributed with the CCP4 suite.)

"Indexed" parameters provide structures similar to one and two-dimensional arrays. An indexed parameter is defined using the "zeroeth" element:

`PARAMETER,0`

The comma (",") indicates that it is an indexed parameter - the index is the part after the the comma. Single indexed parameters have the form:

`PARAMETER,i`

where *i* is an integer and counts from 1 upwards. Double indexed parameters have the form:

`PARAMETER,i_j`

where *i* and *j* are both integers and counting up from 1. Note that although the "zeroeth" element is only singly indexed, it also defines doubly indexed parameters i.e. it is only necessary to use `PARAMETER,0` (not `PARAMETER,0_0`).

3.2 Project and directory definitions: directories.def

The data defining an individual user's projects and def dirs is stored in a file called `directories.def`. The def file "schema" is:

```
N_PROJECTS              _positiveint1          1
PROJECT_ALIAS,0         _text                  ""
PROJECT_PATH,0          _dir                   ""
PROJECT_DB,0            _dir                   ""
PROJECT_ALIAS,1         _text                  PROJECT
PROJECT_PATH,1          _dir                   ""
PROJECT_DB,1            _dir                   ""
N_DEF_DIRS              _positiveint1          1
DEF_DIR_PATH,0          _dir                   ""
DEF_DIR_ALIAS,0         _text                  ""
DEF_DIR_PATH,1          _dir                   \$CCP4_SCR
DEF_DIR_ALIAS,1         _text                  TEMPORARY
```

(See also $CCP4/ccp4i/etc/directories.def.dist. Note that the file contains other parameters used by CCP4i for internal purposes, however they are not part of the database model and so are not described here.)

The table below describes what each of these parameters represents:

Parameter	Description
N_PROJECTS	Total number of projects defined.
PROJECT_ALIAS,i	The name/alias of the i'th project.
PROJECT_PATH,i	The directory path of the i'th project.
PROJECT_DB,i	The database subdirectory of the i'th project.
N_DEF_DIRS	Total number of def dirs defined.
DEF_DIR_PATH,i	The directory path of the i'th def dir.
DEF_DIR_ALIAS,i	The name/alias of the i'th def dir.

Each user has their own directories.def file, which resides in a subdirectory in their home area. On UNIX or Linux systems this is `$HOME/.CCP4/unix/directories.def`

3.3 Job database: database.def

The `database.def` file in the project database subdirectory holds the job database data for that project. The def file "schema" is:

```
NJOBS                   _positiveint            0
TASKNAME,0              _text                   ""
DATE,0                  _date                   ""
STATUS,0                _db_status              ""
LOGFILE,0               _log_file               ""
RUNFILE,0               _run_file               ""
SCRATCH,0               _dir                    ""
TITLE,0                 _text                   ""
INPUT_FILES,0           _list_of_files          ""
INPUT_FILES_DIR,0       _list_of_files          ""
INPUT_FILES_STATUS,0    _list_of_text           ""
OUTPUT_FILES,0          _list_of_files          ""
OUTPUT_FILES_DIR,0      _list_of_files          ""
OUTPUT_FILES_STATUS,0   _list_of_text           ""
```

(See also $CCP4/ccp4i/etc/database.def.)

Note that the `NJOBS` data item is actually the current highest job id number in the database, and not necessarily the total number of jobs.

Not all data items outlined in the schema are explicitly defined in this implementation (this is for consistency with CCP4i). The following data items are implicitly defined:

Implicit Data Item	Comments
JOB_ID	JOB_ID is not explicitly stored in the database.def file, it is an implicit property of the job record. For example, the date for job number 34 is stored in parameter `DATE,34`, while the title of job 102 is in `TITLE,102`.
PARAMETER_FILE	Parameter filenames are implicitly constructed using the following convention: *[JOB_ID]_[TASKNAME].def* and conventionally these files are placed in the database subdirectory of the project directory.
NOTEBOOK_FILE	Notebook filenames are not explicitly stored and are instead constructed using the following convention: *[JOB_ID]_notebook.def* Conventionally these files are placed in the database subdirectory of the project directory.

4 Implementation in SQL tables

A version of the tracking data model has also been implemented in a set of linked SQL tables.

4.1 Project tracking SQL schema

The SQL commands for setting up the tables for the database are given below:

```
CREATE TABLE Project (ProjectId INTEGER primary key,
                 ProjectName VARCHAR(20),
                 Owner VARCHAR(45),
                 Path VARCHAR(45));

CREATE TABLE Job (jobId INTEGER primary key,
             ProjectId INTEGER,
             Status VARCHAR(45),
             Application VARCHAR(45),
             Taskname VARCHAR(45),
             Title VARCHAR(45),
             LastModified DATETIME,
             LogFile VARCHAR(45),
             ControlFile VARCHAR(45),
             NotebookFile VARCHAR(45));

CREATE TABLE File (FileId INTEGER primary key,
             Name VARCHAR(45),
             Path VARCHAR(45),
             Format VARCHAR(45),
             LastModified DATETIME,
             Note VARCHAR(45));

CREATE TABLE JobFile ( JobId INTEGER,
             FileId INTEGER,
             Type VARCHAR(10));

CREATE TABLE JobLink (JobId INTEGER,
             NextJobId INTEGER,
             Note      VARCHAR(45),
             Type      VARCHAR(45))
```

These tables and their attributes are described in more detail in the following subsections.

4.1.1 "Project" table

This table describes the user's projects.

Attributes	Description	Data Type	Example
ProjectId	A unique positive integer id number for this project.	Integer	1, 2, 3, ...
ProjectName	The name or alias for the project. This must also be unique within the system.	Varchar	RNASE
Owner	The name of the owner of the project.	Varchar	"John Smith"
Path	The path of the project directory.	Varchar	/home/pjx/myProject

4.1.2 "Job" table

This table describes the jobs that have been run within the system. Note that all the user's job records are stored in a single "Job" table, with an attribute to specify which project they belong to.

Attributes	Description	Data Type	Example
JobId	A unique positive integer id number for this job (JOB_ID).	Integer	1, 2, 3, ...
ProjectId	A number corresponding to the ProjectId number in the "Project" table for the project to which this job belongs.	Integer	1, 2, 3, ...
Status	The job status (STATUS)	Varchar	RUNNING, FINISHED, FAILED, ...
LastModified	The last modification time of the job data (DATE)	Timestamp	-
Title	A short description of the job (TITLE)	Varchar	"Refine first mr solution from amore"
Application	The name and version of the application program or pipeline which ran the job	Varchar	"CCP4I", "HAPPy", ...
Taskname	The name of the application-specific task which was run in order to generate the job (TASKNAME)	Varchar	"scala", "refmac5", ...
Logfile	The id number of a file in the "File" table (see below) corresponding to the logfile	Integer	1, 2, 3, ...

	for this job (LOGFILE)		
ControlFile	The id number of a file in the "File" table corresponding to the parameter or control file for the job (RUNFILE/PARAMETER_FILE)	Integer	1, 2, 3, ...
NotebookFile	The id number of a file in the "File" table corresponding to the parameter or control file for the job (NOTEBOOK_FILE)	Integer	1, 2, 3, ...

4.1.3 "File" table

This table holds references to all the files that are associated with the jobs and projects. Files are associated with specific jobs via entries in the "JobFile" table described later on.

Attributes	Description	Data Type	Example
FileId	A unique positive integer id number for this file.	Integer	1, 2, 3, ...
Name	The filename i.e. the trailing part without the directory path.	Varchar	toxd.mtz
Path	The directory path for the file.	Varchar	/some/directory/location
Format	The format of the file.	Varchar	mtz, pdb, ...
LastModified	The last modification time for the file.	Timestamp	-
Note	Annotation attached to the file (similar to the "notebook" text for a job).	Varchar	"Deleted"

4.1.4 "JobFile" table

This table holds the associations of files with jobs. Each row in the table links a file to a particular job, as either an input or output file. While a particular file can only be linked to a single job as an output file, it can be linked to many jobs as a input file.

Attributes	Description	Data Type	Example
JobId	The id number of a job in the "Job" table, which forms the first part of the link.	Integer	1, 2, 3, ...
FileId	The id number of a file in the "File" table, which forms the second part of the link.	Integer	1, 2, 3, ...
Type	The character of the relationship between the job and the file, i.e. either "input" or "output".	Varchar	Either "input" or "output" only

4.1.5 "JobLink" table

This table relates two job records together to record some kind of sequence of jobs. Note that this is not part of the original specification.

Attributes	Description	Data Type	Example
JobId	The id number of the job in the "Job" table which is the first job in the link.	Integer	1, 2, 3, ...
NextJobId	The id number of the job in the "Job" table which is the second job in the link.	Integer	1, 2, 3, ...
Note	Annotation attached to the link (similar to the "notebook" text for a job).	Varchar	"Phasing procedure"
Type	The relationship between the two jobs: either "logical" (the relationship is due to application or procedural logic) or "data" (there is a flow of data, for example a shared file, between the jobs).	Varchar	Either "logic" or "data" only

4.2 Differences between the def file and SQL implementations

The SQL tables describe a centralised database, where all data is held in the same database (for example, all the user's job records are stored in a single "Job" table, with an attribute to specify which project they belong to) whereas the def file implementation describes a distributed database (all job records and data for one project is kept separate from all others).

A distributed database is flexible and relatively robust; the centralised approach offers the option to easily change associations and draw links between data and jobs in different projects.

The SQL tables also define explicit links between jobs which do not appear in the original description, but which are needed for a fuller description of project history.

Authors

Peter Briggs and Wanjuan Yang