Table of Contents

LHCb Conditions Database Browser (for COOL/SQLite)

Since end of Run2 LHCb does not support COOL based Conditions Databases. All active branches support GitCondDB, so there's no reason to use this version of CondDBBrowser, but the documentation is kept for reference.

Table of Contents

Introduction

The Conditions Database Browser developed for LHCb is presented here. It was build on top of the CondDBUI (python API to LHCb-COOL) and the PyCool (python API to COOL - the Conditions Database library. This library allows to use the same interface for various database systems, like Oracle, MySQL or SQLite).

The graphical interface of the browser is based on the Qt framework (version 5.x) and on its Python bindings PyQt5.

While mainly targeted to COOL based CondDB, the CondDBBrowser can be used to browse (read-only) Git CondDB databases (by specifying the path to the Git directory as connection string).

Browser Installation For Development

The Conditions Database Browser code is in the subdirectory **Tools/CondDBUI** of the LHCb project. To work on it you can use the following recipe:

```
version=$(lb-dev --list LHCb | awk '{print $1; exit}')
lb-dev --name CondDBBrowserDev LHCb/$version
cd CondDBBrowserDev
git lb-use LHCb
git lb-checkout LHCb/run2-patches Tools/CondDBUI
make CondDBBrowser
./run CondDBBrowser
```

Using the CondDB Browser

This part of the document is mainly a guided tour through which most of the browser's functionalities will be described.

Now, under assumption that you have once build the **CondDBUI** package to run browser you should just type:

```
lb-run LHCb/latest CondDBBrowser
```

This will open the main window of the browser:

CondDB Browser main window view

Navigating the database

Opening an existing database

If you want to open currently released (in SQLDDDB package) database you should go to the menu **Database/Standard** and choose the variant of interest. Among DDDB, LHCBCOND and SIMCOND partitions you will see the list of monthly made snapshots of the Online database.

You may want to open some particular database, then you have to go to **Database/Open** menu:

CondDB Browser open database view

Here you have to locate the .db file and to choose the partition name. Unfortunately, there is no simple way to guess this name and users have to know it to open the database (there is a choice in the expandable list to choose from). Fortunately, for the standard LHCb Conditions database files the partition name is recognizable from the file's names.

By default, the databases are opened in the **Read Only** mode. If you are working on a remote database, it is anyway unlikely that you will have read/write access on it. However, if you are working on your own database, you can open it in Read/Write mode by simply unchecking the **Read Only** check box. When you are done, simply press the Ok button and wait for the browser to establish the connection (for remote databases, this can take some time).

The data tree

CondDB Browser data tree view

The image on the right is an example of the contents of the data tree, located on the left in **Hierarchy browser**. Here the LHCBCOND partition is opened, as an example. You can travel through it like in most file system explorer program. At the top of the tree, you can see a location bar which displays the path to the selected tree item. You can choose the path from the expandable list to go quickly to the item you are interested in.

There are three different type of tree items: *FolderSet*, *Folder* and *Channel*. They are referring to their eponyms in the CondDB.

- The *FolderSets* look like directories and can contain only Folders and other *FolderSets*.
- The *Folders*, which look like a file, can only contain *Channels*.
- Finally, the *Channels*, which also look like files, contain no other tree items.

Each of these items is identified by a name (the *Channels* are identified by a number and the others by a string). In the latest version of CondDB Browser the expandable property of a *Folder* to show the *Channel* is turn off if there is only one *Channel* and it is named "0". In all other cases the *Folder* is expandable to show all the *Channels*.
Folders items also may have a **Version Style** property, telling if the values stored in the CondDB folder can be tagged ("Multi") or not ("Single"). This feature is going to be implemented soon in the **Hierarchy Browser**.

Useful tip:

- If you want to copy the path of selected *folder* you can just right-click on the *folder* of interest to open the pop-up menu, and choose *Copy Path*. There you can find another few useful actions, but they are available only in the write mode.

The data display and the time filter

Selecting a *Folder*, or a *Channel*, item will modify the contents of the central and right sides of the main window: the data display and the time browser. The image below shows what appears when selecting the /Conditions/Ecal/Calibration/Gain.xml *Folder*:

Contents of the data display and of the time browser

- The data display (the central one part) displays the contents (in text) of a condition object stored in the database.
- The **Time Browser** offers the possibility to select the condition object to display. These condition objects are listed in the **IOV Table** area. By default, the list of condition objects is reduced to the one, which is active starting from current time. Full list (or the one for specific "From time and To time") can be seen adjusting the settings in the **Filter** area of the **Time Browser**.

If the folder in which the data are stored is a multi-version one, it is possible to get various versions of its contents. This is done using the **Tag Name** selector:

Tag names list area view for the particular _folder_

The selector contains the list of tag names available for the selected Folder (or the Channels inside it). By default, the browser only shows user defined tags, and filters out tags generated automatically (except the HEAD). To display all tags, you can uncheck the **Hide '_auto_' tags** box, just below the tag name selector. Each group of tags (together with 'auto' tags) lying within the area of single color (white or gray one) are referring to the same data. The name, which will pop-up when the cursor is idle on a particular tag name, corresponds to the node (*Folder* or *FolderSet*) under which the tag name is visible. Later in this tutorial, we will see in more details how tagging works.

Useful tip:

- If you want to copy the current contents of the text box you can just select the text and right-click on it to open the pop-up menu, and choose *Copy*. There you can find another few useful actions.

Editing the Database

Creating a new database

There are two ways to create a new database: create it from scratch or create a copy of an existing one. These two possibilities are available respectively in the menu **Database/New** and **Database/Slice**. In both cases, you can only create a SQLite database.

Creation from scratch:

The creation from scratch is strait forward: in the dialog, you simply choose a **File name** and a **Partition** name (in capitals), press OK and start playing:

Creation from a slice:

To create a slice of a database, you first need to open the source database. Then, open the slice dialog window:

In the **Destination** fields, you should put the **File name** and **Partition** name where you want to store the slice of the source database. Decide here whether you are going to create a new database copy or append data to the target database. If the second one, choose the database target which exists and check the **Append** checkbox.

In the **Selection** fields, you will give details of the part of the source database you want to put in the target database. The **Node** selector contains a list of the source's nodes. The copy is recursive, so if you select a *FolderSet*, all its childs will be copied as well. The **Since** and **Until** fields define the time interval you want to copy in the target database. Only objects valid during this period will be copied. Finally, the tags relevant for the selected node are listed in the **Tags** list. If you are doing a copy, you can select as many tags as you want

(by just clicking on their names). However, if you are appending data to the target, you can select only one tag which will become the new HEAD of the target database (i.e. the tag name will *not* be copied !).

When the details of your selection are correct, you have to press the **Add** button. The **Selection set** gives a summary of the selections you want to copy in the target database. You can add as many selection objects as you want, but be careful: some selections may be incompatible and the slicing tool may not accept them.

Be careful: Copying is safe, but appending is a risky operation: selected data are appended to the HEAD of the target database, overriding previous HEAD data. If you are not careful enough you may endup with data loss.

When creating/appending is done, you can open the target database normally (i.e. via **Database/Open** and selecting the last choice of the Schema selector).

Adding a new Condition

If you opened your database in read/write mode (can be turned on from the **Database** menu), a new menu is available: the **Edit** menu. It provides the most common operations for users willing to modify the content of their database.

The most obvious operation is to add a new condition object. For this, we first need to select the *Folder* where we want to store this condition (in the data tree). In our example, it will be /Conditions/TT/ReadoutConf/ReadoutMap.xml. Then, we open the "Add condition" dialog window by selecting **Edit/Add Condition**:

The **Folder** field is read only to prevent mistakes. The other fields allows us to set the **Channel ID** of our new condition object (0 by default). **Since** and **Until** are the lower and upper bounds of the condition object's interval of validity. On the right, the list of available **payload keys** is displayed. To edit the contents of the condition object, you have to select the **Payload Keys** you want to modify and press the **Edit** button:

A text editor will then open the current content of the selected *folder*, or if the *folder* is new - will open with a few XML blocks already written. These blocks are mandatory for all conditions data stored in the LHCb Conditions Database. The five buttons at the top of the text editor are providing some helpful operations:

- **Import**: loads text data from a file. Be aware that this overrides the content of the text editor !
- **Export**: saves the content of the text editor into a file.
- **<condition>**: insert a preformated XML tag for the description of an LHCb condition
- **<param>**: insert a preformated XML tag for the description of a condition parameter, which can be of type "int", "double" or "string". A <param> tag must be children of a tag.
- **<paramvector>**: same as <param>, but instead of storing a single value, it stores a vector of values ("int", "double", or "string").

This editor is very basic and, for example, has no "undo" feature (although if you press **Cancel** the changes will not be saved). Future version of the browser will probably allow to edit conditions in an external text editor (emacs, vim, nedit, etc.). For the moment, the only possibility for you to do so, is to prepare condition data with your favorite editor and to load the contents in the condition editor.

When you have finished the edition of the condition data, you press **OK** and get back to the "Add condition" dialog. Now, to write the condition in the database, you have to add it to the **Condition Objects Stack** by pressing the " **+** " button. A summary of the properties of the condition object will appear.

The stack will contain all the new condition objects you want to add to the selected *Folder*. Objects in the stack are not (yet) editable, but selecting one of them will reload its parameters in the location details and the payload key list. If you want to "modify" an entry, select it to reload its parameters, modify them, add the new object to the stack, and delete the old version using the " **-** " button.

The last detail to keep in mind before writing data in the *Folder* is that conditions objects will be written to the database in the order they appear in the stack. For example, if you have a condition object *A* going in channel 0 and with validity from 500 to 1000, and an object *B*, also in channel 0, with a validity ranging from 700 to 1100, the order in which you insert them will give different results. If *A* is written before *B*, you will get *A's* value valid from 500 to 700, and *B's* value from 700 to 1100. If *B* is written before *A*, *A's* value will be valid from 500 to 1000 and *B's* value from 1000 to 1100.

When your list of condition objects is ready, pressing the **OK** button will update the database. You can now see the result in the browser.

Tagging

To keep track of various possible values of a condition for a given validity range, it is possible to apply a tag to the current state of any node of the database. This is called "tagging". The tagging system of the Conditions Database (called HVS for Hierarchical Versioning System) is very powerful but not easy to describe in a few words.

In the previous section, we have created a new set of condition objects to put inside the /Conditions/TT/ReadoutConf/ReadoutMap.xml *Folder*. To keep track of this modification, we need to apply a tag to it. This is done by first selecting the *Folder* item in the tree, and open the menu **Edit/New Tag**:

You simply have to choose a **New Tag Name** and press the **OK** button. The tag name you choose must be unique in the database. The new tag is created and should appear in the tag list of the *folder*. Now, whatever

changes you can make to the contents of your *folder*, the current status of your data can be recovered by selecting the tag you've just created.

Now, when you want to use these conditions, you don't want to specify a tag for each *Folder* you are going to use. This is why it is possible to create tags for *FolderSets* as well. But as *FolderSets* do not contain data (only the *Folders* contain data), you have to create a "link" between the *FolderSet's* tag and the tags of the child nodes, which eventually link to the tag of a *Folder*. This is the principle of the Hierarchical Versioning System.

To illustrate this, we are going to create a new tag for the *FolderSet* /Conditions/TT/ReadoutConf. To do so, we select it in the tree, and open **Edit/New Tag**:

The dialog is different than the one used to tag a *Folder*. Here, we have a list of the child nodes of the selected *FolderSet*. These nodes can be either *Folders* or *FolderSets*. But for each of them, we can select the tag name that will be linked to the one we are about to create. By default, it is the HEAD. But if you double-click on the selector in the column **Tag** for the node /Conditions/TT/ReadoutConf/ReadoutMap.xml , you can find any tag available for it. But for most cases, it is more convenient to open the **Fill from an existing tag** dialog to choose among an existing tags for the current *FolderSet* the tag of interest, and then modify some of the tags for the child nodes.

When you choose to link a *FolderSet* tag to the HEAD tag of one of its child nodes, the system will automatically create a new tag for this children and create the appropriate link. Tags generated automatically are always starting with "_auto_", so it is a bad idea to use this format for your own tags.

Creating a new node

The creation of a new node in the CondDB is not a common operation. Usually, the structure of the database will be fixed and not supposed to change. For this reason, the **New Node** action is available in the **Advanced**

menu, together with **Delete Node** and **Delete Tag**.

If we select /Conditions/TT *FolderSet* to create a new node in it the dialog window will look like this:

The **Node Name** is the full path to the new node, and the **Description** is a facultative string describing the nature and contents of the node. In the **Type** area, you can decide if you create a *FolderSet* (by default, you are creating a **Multi version** *Folder*), if the *Folder* will be a **Single Version** one (meaning that no tagging is possible), and asking to recursively **create parents** nodes (this action does not override existing nodes). If you open the **Edit Fields** area you will see the list of storage keys. It is meaningless if you are creating a *FolderSet*. However, it is very important if you are creating a *Folder* because all the condition object you will store later will have these keys defined. To add a new one, simply press the " **+** " button, and double click on the new cell to give a name to the new storage key. Finally, pressing the **OK** button will make the new node appear in the tree.

Deleting a node

Node deletion is to be use carefully as it results in data loss. To delete a node, you simply select it in the tree and go to menu **Advanced/Delete Node**. If the node is a **FolderSet**, it must be empty before being able to delete it.

Deleting a Tag

Tag deletion may also imply data loss. To do it, select the node for which you want to delete a tag, then go to the menu **Advanced/Delete Tag**. A list of available tags will be given and you simply have to select the one you want to delete. While HEAD is available in the list, you can't delete it (something to be updated in later versions...).

Further Readings

The Conditions Database Browser is a "never finished" project and contributions are welcome. Documentation for the various tools used to develop the program are available on the web:

- Python2
- PyQt5
- Qt5

As it is now, it should respond to most of the users requests for database manipulation. For more advanced features, you can use:

- the CondDBUI documentation
- the COOL C++ API.

For more details about the Conditions Database framework in LHCb, please refer to the Detector Conditions web page.

-- IllyaShapoval - 31-Jan-2010 -- MarcoClemencic - 2017-06-27

This topic: LHCb > CondDBBrowserSQLite
Topic revision: r1 - 2019-02-05 - MarcoClemencic