# THE LHCB CONFIGURATION DATABASE

L. Abadie[1, 2], E. v. Herwijnen[1], C. Gaspar, R. Jacobsson, B. Jost, N. Neufeld

[1]*CERN, Geneva, Switzerland,* [2]*University Pierre et Marie Curie (Paris VI), LIP6*

## ABSTRACT

The Experiment Control System (ECS) [1] will handle the monitoring, configuration and operation of all the LHCb experimental equipment. All parameters required to configure electronics equipment under the control of the ECS will reside in a configuration database. The database will contain two kinds of information:

1. Configuration properties about devices such as hardware addresses, geographical location, and operational parameters associated with particular running modes (dynamic properties).
2. Connectivity between devices: this consists of describing the output and input connections of a device (static properties).

The representation of these data using tables must be complete so that it can provide all the required information to the ECS and must cater for all the subsystems.

The design should also guarantee a fast response time, even if a query results in a large volume of data being loaded from the database into the ECS.

To fulfill these constraints, we apply the following methodology:

- Determine from the dataflow the list of all devices in the subsystem and the connections between devices
- Collect use cases and apply the *Entity Relationship Model* to design the schema of the tables
- Define an API allowing users to interact with the database according to the use cases.
- Implement the API and create tools.

The configuration database is a relational database which has been implemented using Oracle. A part of the configuration database is built using the Database tool developed in common for the four LHC experiments by JCOP (Joints Controls Project) [3].

To test the configuration database design we have integrated it into the LHCb TFC and DAQ systems.

## INTRODUCTION

LHCb is one of the four particle detectors at the CERN LHC (Large Hadron Collider). The monitoring, configuration and operation of experimental equipment will be handled by the ECS (Experiment Control System).Fig. 1 shows the relationship between ECS and other systems.

PVSS [2], a SCADA (Supervisory Control and Data Acquisition) system provides the interface to the experiment's equipment. A run is associated with an activity or mode such as calibrations, physics which corresponds to a particular setting of the detector and with a partition. A partition is a part of the detector which can run independently and concurrently. For each activity, we store the settings of each controllable device which represents huge amount of data in the configuration database.

So all information regarding controllable electronics equipment will be extracted from the configuration database and loaded into PVSS.

In order to define a common architecture between the four experiments, JCOP offers a common framework and tools for PVSS. One of these tools enables users to save and load configurations of devices.

To allow partitioning, we also store information about connectivity between devices in the configuration database using LHCb specific tools.

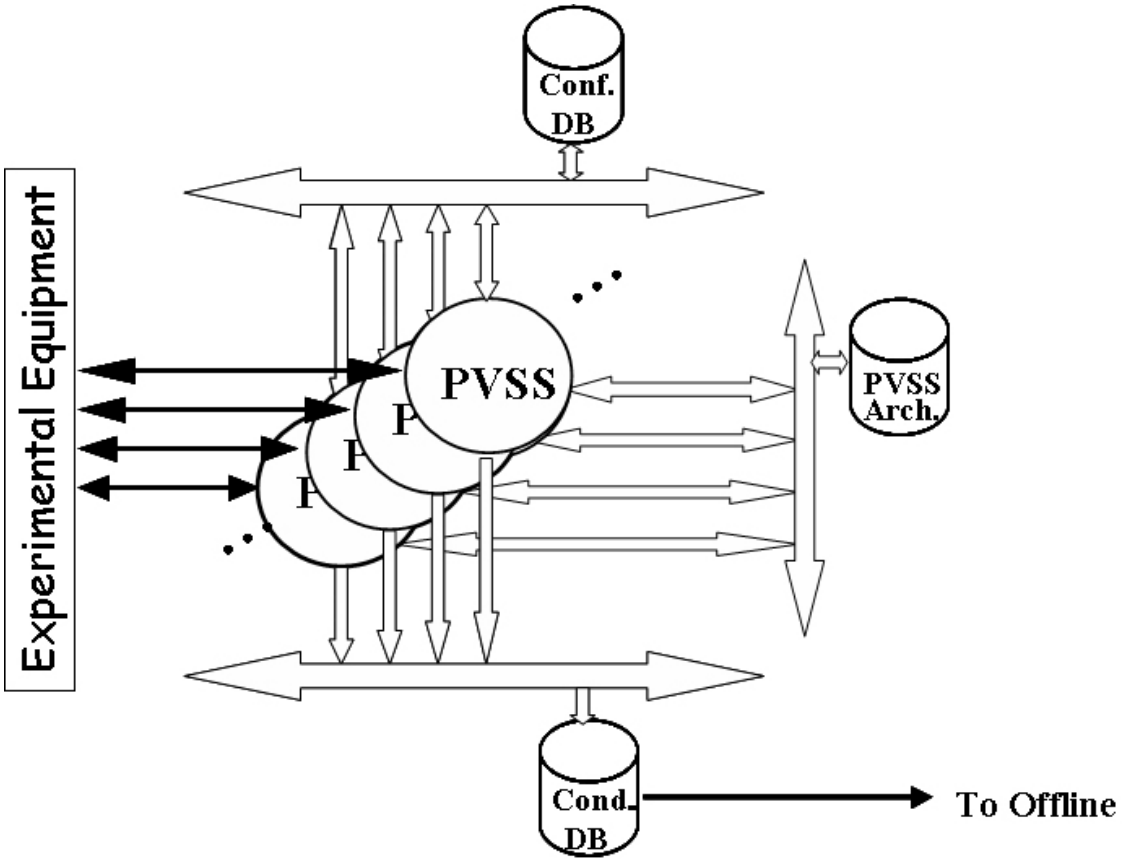


**Figure 1 : External data handling architecture.**

## THE SCHEMA

In this section we describe the objectives and requirements of the LHCb configuration database, and the methodology used to design its schema.

*Objectives*

The database should contain the data that the ECS needs to configure the experiment's equipment:

- **Static properties**. These correspond to properties that don't change or change very infrequently, e.g. the hardware addresses, the geographical location, and device structure. Changes in this data involve a hardware intervention. This information is very useful to keep track of devices especially for debugging.
- **Dynamic properties**. These correspond to properties that change with the activity, e.g. alarm settings, hardware settings such as ramping speed.
- **Connectivity**. This describes how a device can be connected to other devices, e.g. a switch can be connected through one of its input ports to a supervisor and through one of its output ports to a card somewhere in the detector. This allows the creation of routing tables on the fly for dynamic programming of switches or lookup tables for trigger boards. This information is used for partitioning, in other words to determine what devices need to be configured per subsystem and also how to interconnect all the detector subsystems with the TFC, DAQ and ECS.

*Requirements*

The following requirements should be taken into account:

- The schema should be integrated with the one used by the JCOP configuration database tool.
- The schema should be generic enough to cater for all detector subsystems.

- The database should be complete so that the ECS finds all the required information.
- The design should guarantee a reasonable response time when a query resulting in a large volume of data is retrieved from the database into the ECS.
- The tables should be easy to maintain.
- The database will be filled principally from PVSS.
- It should be possible to use the database remotely, possibly without a connection to the database.
- A graphical tool is required to navigate and edit the data.

### *Methodology*

The following methodology has been applied for each subsystem:

- Consider its dataflow. Fig. 2 shows the TFC (Timing and Fast Control) system dataflow [4]. From this we can determine:
  - The list of all devices in the subsystem, e.g. readout supervisor, TFC switch, different fan-out types such as TTCtx, TTCrx belong to the TFC system.
  - The connection between devices, e.g. the TFC switch is connected to the readout supervisor on input and to a TTCtx fan-out on output.

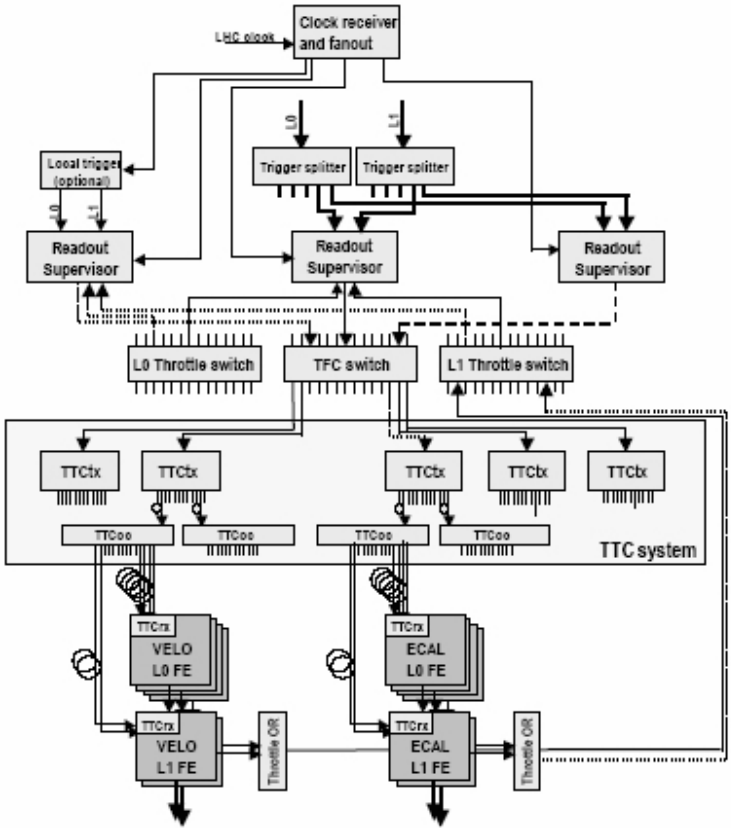


Figure 2: TFC dataflow system

- Collect use cases. The following is an example from the TFC system. For a partition consisting of the VELO and ECAL subdetectors, determine the appropriate readout supervisor and the internal connectivity of the TFC switch. This use case introduces the concepts of links and paths (c.f. Fig. 3) [5].

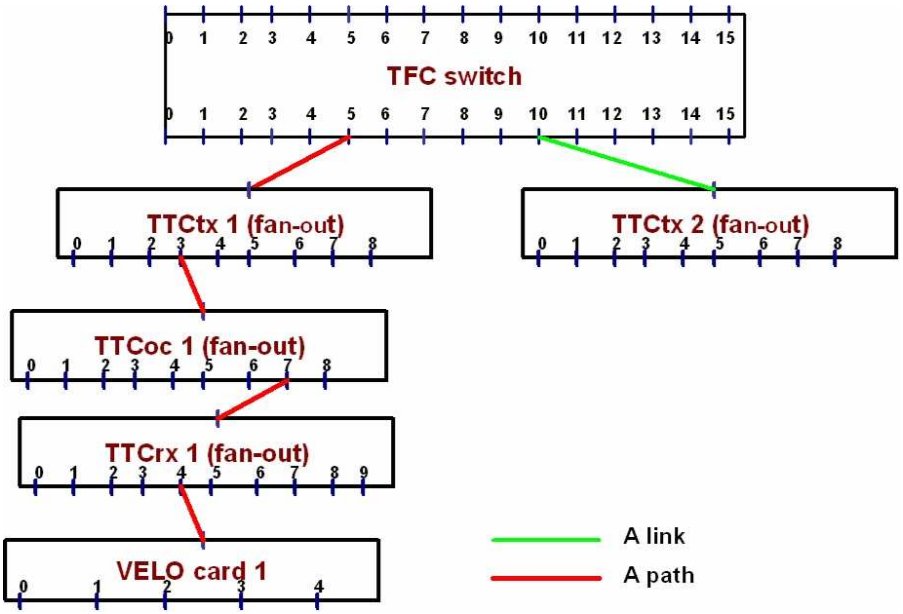


Figure 3 Concept of links and paths

- Use the entity relationship model to design the schema of the tables [6]. Fig. 4 shows the Table design

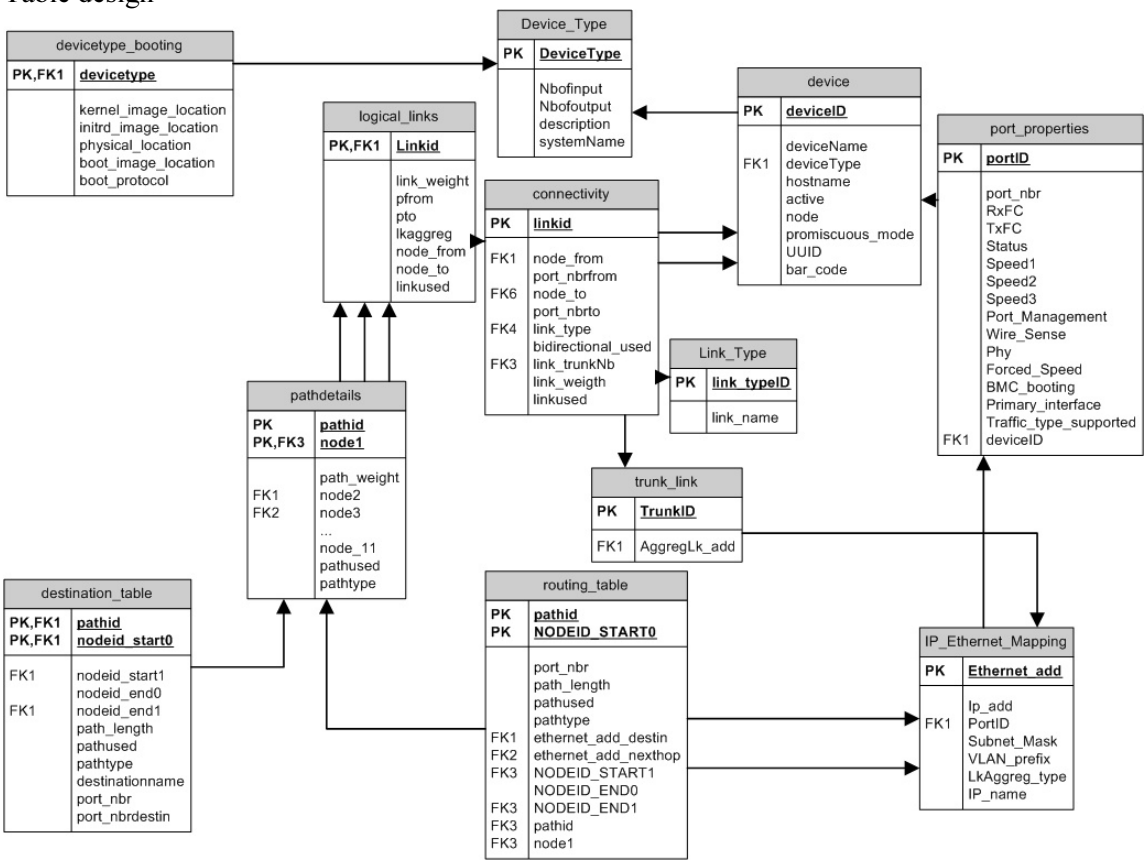


Figure 4 Table design for the LHCb specific part

## IMPLEMENTATION

This section describes the components and tools of the configuration database.

### *Integration of the JCOP configuration database tool*

An important requirement was to fully exploit the work done by JCOP. This tool provides tables that contain dynamic device properties. They are stored in the configuration database from PVSS panels. It does not store information about links or hierarchy which are stored in tables specific to LHCb. Some work had to be done to provide a unified interface to both sets of tables; the LHCb tables point to information stored in the JCOP part to avoid duplication of data.
Conventions regarding the name of devices and activities ensure the coherence between the JCOP tables and the LHCb tables. We have suggested that all devices and activities should be prefixed by the subsystem name when saving them in the configuration database. For instance, a beetle chip from the VELO subdetector will be called VELO_*beetlename.*
Fig.5 shows the two different parts of the database.

### *The confDB library*

The LHCb specific part of the configuration database consists of storing the connectivity of the devices in each subsystem. We have designed an API to allow developers and users to populate and query against the LHCb specific part.
The API has been implemented as a C library of functions [7] using OCI [8] called confDB library.
This library is used to:

- Populate the database with links between devices.
- Generate routing and destination tables
- Get the path between 2 devices.

A PL/SQL [9] package provides functions to generate routing and destination tables. The PL/SQL functions are part of the library.

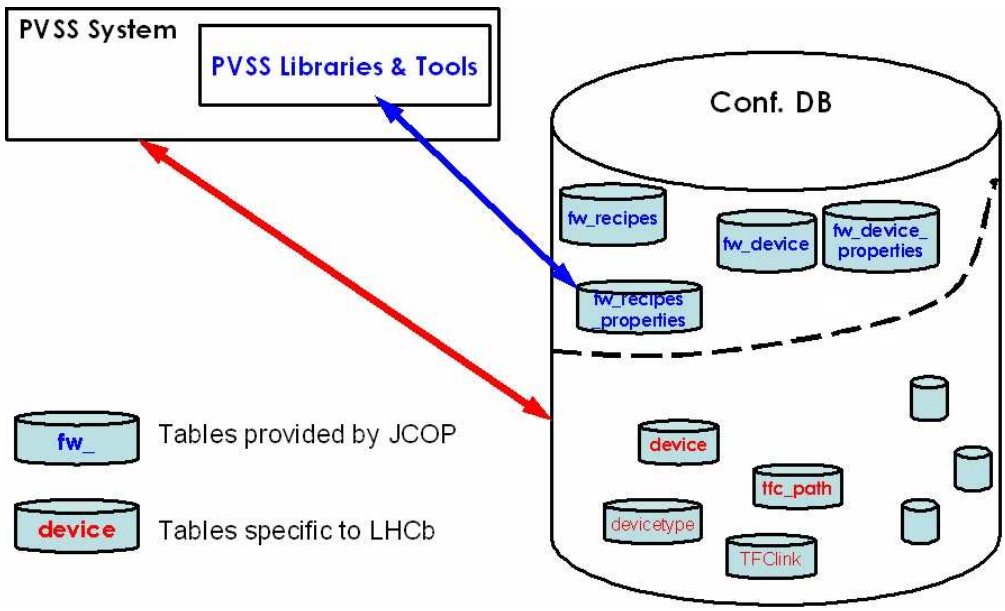


Figure 5 : the two parts of the configuration database: LHCb specific part and tables provided by the JCOP tool

### *Implementation details*

The configuration database has been implemented using Oracle technology. Oracle was chosen because of the CERN Central support and the wide range of tools. For future extensions of the database to other subsystems, we have tested the initial prototype and provided with some feedback. A new version of the JCOP tool is expected by end September 2005. We shall use this version to store the dynamic subdetector data related to an activity.
To interface the confDB library with PVSS, we use the General External Handler, module provided by JCOP
Standard database editors are not good at making the table structure explicit. We required a tool to inspect the links and hierarchy of the devices in the database graphically
We have implemented in Python a tool to edit and navigate through the database (see next subsection for details). This python tool also uses the confDB library. To make it work with Python, we have first to develop a C++ interface and then to use BOOST [11] integrated with Gaudi to interface with Python. A Python prototype (cdbVIS [12]) allows viewing of the connectivity.

We keep the different versions of PVSS projects and other software in CVS [13].

## CONCLUSION

The design schema has been done for the DAQ and TFC system. We will apply it for the other LHCb subdetectors.
We have also integrated PVSS and the configuration database (JCOP and LHCb parts). We have developed a C library of functions and integrate it in PVSS and also in Python.
We also plan to add some functions such deleting and updating attributes and also to extend the features of the cdbVis.
The initial JCOP looks promising and we are waiting for the new version.
We can store the connectivity but we need to define the granularity: it will be a case by case based on subsystems.
We need to write some user guidelines and to start thinking about storing the history.

**ACKNOWLEDGEMENT**
We would like to thank the JCOP group especially Piotr Golonka and Manuel Gonzales for their help with the JCOP confDB tool.

## REFERENCES

[1] http://lhcb-comp.web.cern.ch/lhcb-comp/ECS/default.html
[2] http://itcobe.web.cern.ch/itcobe/Services/Pvss/welcome.html
[3] http://itcobe.web.cern.ch/itcobe/Projects/Framework/
[4] http://lhcb-comp.web.cern.ch/lhcb-comp/TFC/default.html
[5] http://lhcb-online.web.cern.ch/lhcb-online/configurationdb/PLSQLpck_doc.htm
[6] http://lhcb-online.web.cern.ch/lhcb-online/configurationdb/default.htm
[7] http://www.w3schools.com/ado/ado_connect.asp
[8] http://lhcb-online.web.cern.ch/lhcb-online/configurationdb/APIusage.htm
[9] http://www.oracle.com/technology/tech/oci/index.html
[10] http://www.oracle.com/technology/tech/pl_sql/index.html
[11] http://www.boost.org/libs/python/doc/index.html
[12] http://lhcb-comp.web.cern.ch/lhcbcomp/ECS/configurationdb/cdbVis.htm
[13] http://isscvs.cern.ch/cgi-bin/cvsweb.cgi/TFC/?cvsroot=lhcb