The CMS ECAL database services for detector control and monitoring

2010 J. Phys.: Conf. Ser. 219 022016

(http://iopscience.iop.org/1742-6596/219/2/022016)

Journal of Physics: Conference Series **219** (2010) 022016 doi:10.1088/1742-6596/219/2/022016

The CMS ECAL Database Services for Detector Control and Monitoring

Roberta Arcidiacono[9], William Badgett[8], Ursula Berthon[11], Angela Brett[4], Francesca Cavallari[2], Guy Chevenier[5], Andre David[6], Joao De Almeida Simoes[5,6], Giuseppe Della Ricca[10], Emanuele Di Marco[2,5], Ricky Egeland[3], Gautier Hamel De Monchenault[12], Matteo Marone[9], Pasquale Musella[6], Giovanni Organtini[1,2], Pascal Paganini[11], Vladlen Timciuc[7], Zongru Wan[13]

[1] Sapienza, Università di Roma, Roma, Italy
[2] INFN–Sez. di Roma, Roma, Italy
[3] University of Minnesota, Minneapolis, Minnesota, U.S.A.
[4] Institute for Particle Physics, ETH Zurich, Zurich, Switzerland
[5] European Organization for Nuclear Research, Geneva, Switzerland
[6] Laboratório de Instrumentação e Física Experimental de Partículas, Lisboa, Portugal
[7] California Institute of Technology, Pasadena, California, U.S.A.
[8] Fermi National Accelerator Laboratory, Batavia, Illinois, U.S.A.
[9] INFN–Sez. di Torino, Torino, Italy
[10] INFN–Sez. di Trieste and Università degli Studi di Trieste, Trieste, Italy
[11] Laboratoire Leprince-Ringuet, Ecole Polytechnique, IN2P3-CNRS, Palaiseau, France
[12] DSM/DAPNIA, CEA/Saclay, Gif-sur-Yvette, France
[13] Kansas State University, Manhattan, Kansas, U.S.A.

E-mail: `Giovanni.Organtini@roma1.infn.it`

Abstract. In this paper we give a description of the database services for the control and monitoring of the electromagnetic calorimeter of the CMS experiment at LHC. After a general description of the software infrastructure, we present the organization of the tables in the database, that has been designed in order to simplify the development of software interfaces. This feature is achieved including in the database the description of each relevant table. We also give some estimation about the final size and performance of the system.

1. Introduction

CMS [1] is one of the general purpose experiments at the LHC. It is a complex detector subdivided into subdetectors, each specialized in detecting and measuring a given class of particles. In CMS, the silicon tracker, the electromagnetic calorimeter (ECAL) and the hadronic calorimeter are all contained in a superconducting solenoid providing a uniform 3.8 T magnetic field parallel to the beam axis. Four muon stations are interspersed within the return yoke of the magnet.

Due to very different requirements for the various subdetectors, the collaboration agreed that each subdetector group would develop its own database schema and services for their control and monitoring. In this paper we discuss the details of the ECAL database.

ECAL [2] is composed of 75848 lead-tungstate scintillating crystals. The crystals are tapered and arranged to approximately form a cylindrical barrel closed by two end-caps, whose axis coincides with the beam axis. The light produced by the crystals is read out by avalanche photodiodes (APDs) in the barrel and vacuum phototriodes (VPTs) in the endcaps. The photodetector bias is provided by 1276 high–voltage lines, while the power to the front–end boards is fed by 844 different channels. Moreover, there are about 7300 temperature sensors distributed among both the barrel and end-caps, which are regularly monitored.

2. Database project aims and scope

The ECAL database aims at storing data from quality control tests during construction, detector conditions on site, global runs and local runs. Quality control data were collected during the construction phase of the calorimeter and include the associations between crystals, photodetectors, read–out channel and photodetector optimal biases. Moreover, they provide information about the position of every element in the final setup. Some of the quality control data were also used to derive an initial calibration constant [3] for each channel. Detector conditions include data about temperature, humidity, status of the high–voltage and low–voltage distribution, etc., at any time. These data are written to the database each time their variation from previous values exceeds a given threshold, whose exact values depend on the nature of the data. In order to check that the whole monitoring system itself is working, the same data are written to the database every 20 minutes, even if they are stable. Global run conditions include the DAQ and trigger configuration used for each CMS run. Local runs are those runs involving ECAL only; they are special dedicated runs to take data relevant for ECAL control and monitoring.

Local runs are taken between physics fills and are of three types:

(i) DCU (Detector Control Unit) [4]: measure parameters of the front-end electronics such as gains, temperatures, dark currents, voltages, etc. These are used to monitor the stability of the read out and frontend electronics.

(ii) Pedestal: measure the average pedestal level and their width per channel;

(iii) Laser runs: measure the detector response to laser light injected to each single crystal by means of optical fibers, in order to monitor the crystal transparency over time.

Laser runs are also automatically taken during physics runs. The LHC bunch train is such that there is a gap of about 3 μs between the last circulating bunch and the first one. Laser light is injected in crystals during this gap.

Pedestals are also measured during the physics runs. The crystal signal is sampled at 40 MHz and in the event of a positive decision from the level–1 trigger system ten consecutive samples are read out. The first three samples, which contain no signal, are used to determine the pedestal event by event.

There are actually two database instances for CMS: one online and one offline. The online database is physically located close to the experiment and is used to operate the detector and store all the data described above. The portion of these data relevant for physics event reconstruction are automatically copied to the offline database sitting on a machine in the CERN IT division using a dedicated link. We use ORACLE database 10g Enterprise Edition as our Database Management System (DBMS) and its streaming functionalities to copy data from the online to the offline database.

The focus of this paper is on the online database, whose content is accessed via different mechanisms, whose details are given in the following section.

3. Software architecture

We can identify three main use cases for the online database: detector control and configuration, execution of local runs, and browsing of its content.

Detector control and configuration require a Graphical User Interface (GUI) by which a user can load a default configuration from the database, modify it if needed, and store the new configuration on the database, as well as send it to the detector. Run control and remote power control are two examples of this use case. The Detector Control System (DCS) is handled by a PVSS project [5]: a commercial SCADA (Supervisory Control And Data Acquisition) tool for automation produced by the ETM company, as prescribed for LHC experiments. It then uses the PVSS RDB library to communicate with the DBMS. The run control interface adopts the Qt graphic library [6] for the rendering of the interface and C++ to connect to the database.

Local runs do not require a complex user interface and must be fast. For these reasons local run control programs are compiled applications written in C++. They read parameters from the database, acquire data from ECAL, analyze them and store results on the database using the OCCI library provided by ORACLE.

Browsing the database content is achieved using a common web browser. The interface with the database is provided via JDBC calls made by Java servlets. The latter organize the data retrieved from the database in such a way that the user can get all the information in textual or graphical form, rather than as just tables.

Data related to the detector configuration (DAQ and power status) as well as environmental data have a simple, well defined and immutable structure. They are essentially a collection of $(key, value)$ pairs and therefore are stored in well defined, simple tables. Each entry having some relationship with the runs or the timestamp.

On the other hand, monitoring tasks may produce very heterogeneous and complex sets of data and their number is subject to change with time. As the knowledge of the detector increases new tasks may be added to keep under control aspects such as the absolute energy scale or the occurrence of some class of physics events that can be used to monitor the detector behavior.

For this reason the design of the tables containing monitoring data conforms to a simple and general pattern which can use well–understood read/write methods, while the underlying detail tables are specific to the data being stored. The current schema design employs the "Item Description" pattern used in OO programming (illustrated in the next section). This allows data retrieval and interpretation to be made automatic. The description for each table is stored in the database allowing the proper formatting of the data to be delegated to an algorithm.

4. Principles of database tables organization

A simplified version of some of the database tables is shown in Fig. 1. All runs are recorded in a dedicated table. Many different tasks can be associated to a single run (measurement of the transparency with a laser, pedestal monitoring, stability control, etc.). Each task has an entry in its own table and a relationship with the corresponding run, so there is a table for the measurement of crystals' transparency with laser, one for the monitoring task, etc. In the figure, the word `TASK` in the table name represents, in fact, a specific name for the task. Each task can then produce data of very different nature and meaning. For example, the monitoring task may produce data in form of floating point numbers for the pedestal values and in the form of CLOBs (Character Large OBjects: one of the most complex data type supported by the DBMS) for structured data, such as XML configuration strings for the DAQ. Data are contained in separate tables related to the task which produced them by a foreign key. Even in this case the name of the table given in the figure has to be considered as a variable, so that the portion of the name reported as `TASK_DATA` changes according to the content. The content of the data tables is described in a record in the `COND_TABLE_META` and contains, besides other information, a textual description of the data, the number of fields, and the process which fills it.

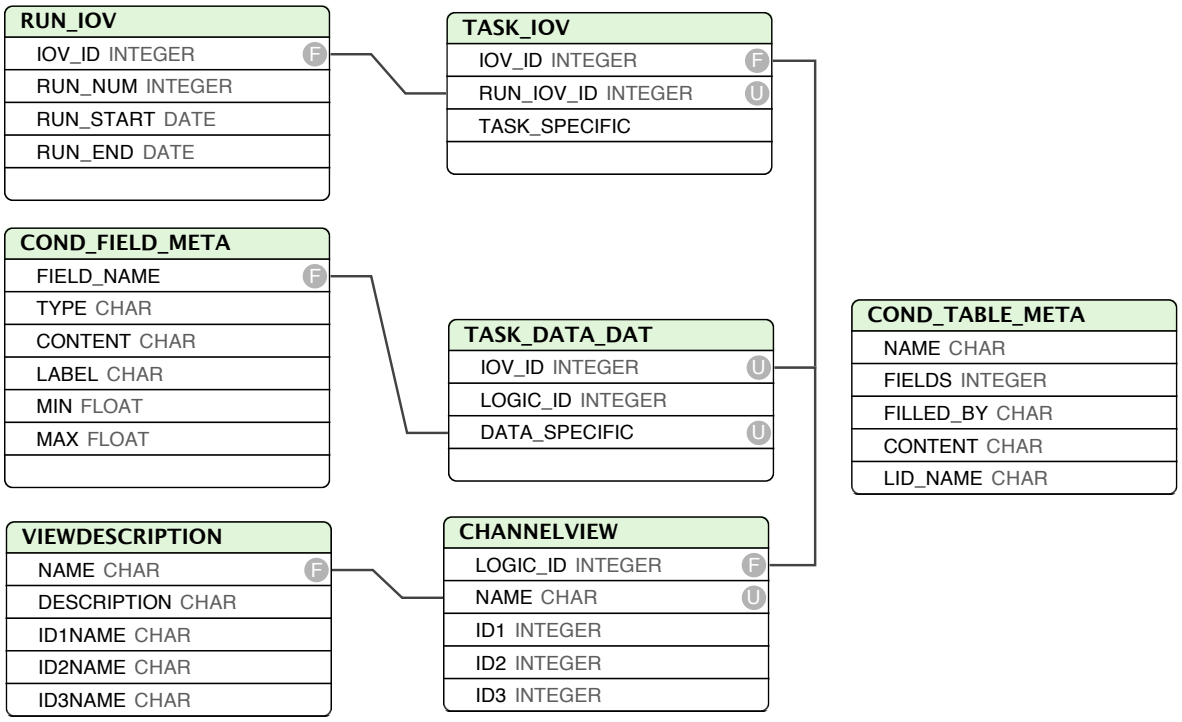

Figure 1. A simplified ER schema for the ECAL online database.

TABLE: RUN_IOV

IOV_ID	RUN_NUM	RUN_START	RUN_END
20424	73030	04-02-2009 12:54:48	04-02-2009 12:55:06
20425	73036	04-02-2009 13:31:25	04-02-2009 13:33:03
20426	73041	04-02-2009 14:35:20	04-02-2009 14:36:08
20427	73072	04-02-2009 17:15:04	04-02-2009 17:15:33
20440	72959	03-02-2009 16:41:29	03-02-2009 16:52:48
20441	72994	04-02-2009 09:17:22	04-02-2009 09:17:41
20442	72997	04-02-2009 09:27:57	04-02-2009 09:28:14
20443	73016	04-02-2009 11:16:37	04-02-2009 11:18:35
20460	72963	03-02-2009 16:52:48	03-02-2009 16:53:22

Table 1. An excerpt of the table containing generic run data.

Each field in the data tables needs a description that is given in the `COND_FIELD_META` table. For each field name there is an entry in this table containing a textual description of the represented data, the type, the minimum and maximum value, the names of the axis to be used when plotting these data, and other information.

In Tables 1–6 we give a simplified example of the use of the tables. Table 1 shows an excerpt of the run table. The second line of that table, corresponding to run number 73036, has a relationship to the `MON_PEDESTALS_DAT` table by means of its "interval of validity" identifier `IOV_ID`. An example from `MON_PEDESTALS_DAT` is shown in Table 2. In fact such a relationship is mediated by a table called `MON_RUN_IOV`, represented by the `TASK_IOV` table in Fig. 1. The complexity added by this detail, however, is irrelevant for the current context.

TABLE: MON_PEDESTALS_DAT

IOV_ID	LOGIC_ID	PED_MEAN_G1	PED_RMS_G1	PED_MEAN_G6	PED_RMS_G6
20425	1011060001	1.938E+002	6.238E-001	1.925E+002	7.286E-001
20425	1011060002	1.96E+002	5.906E-001	1.989E+002	8.006E-001
20425	1011060003	2.112E+002	6.053E-001	1.929E+002	6.98E-001
20425	1011060004	1.942E+002	6.394E-001	2.078E+002	7.574E-001
20425	1011060005	1.92E+002	6.657E-001	1.994E+002	7.59E-001
20425	1011060006	2.114E+002	5.531E-001	1.994E+002	7.251E-001
20425	1011060007	2.111E+002	5.792E-001	2.026E+002	7.354E-001
20425	1011060008	1.918E+002	5.66E-001	2.103E+002	7.427E-001
20425	1011060009	1.969E+002	5.137E-001	2.044E+002	7.796E-001

Table 2. Part of the table containing pedestals data.

TABLE: COND_TABLE_META

TABLE_NAME	FIELDS	FILLED_BY	CONTENT	LID_NAME
MON_PEDESTALS_DAT	7	DQM	crystal pedestals	EB_crystal_number

Table 3. Data table explanation.

TABLE: COND_FIELD_META

FIELD_NAME	TYPE	CONTENT	LABEL	MIN	MAX
PED_MEAN_G1	FLOAT	Pedestal gain 1	Ped. G1 (ADC Ch)	180	220
PED_MEAN_G6	FLOAT	Pedestal gain 6	Ped. G6 (ADC Ch)	180	220
PED_RMS_G1	FLOAT	RMS of Pedestal gain 1	Ped. RMS1 (ADC Ch)	.5	.9
PED_RMS_G6	FLOAT	RMS of Pedestal gain 6	Ped. RMS6 (ADC Ch)	.5	.9

Table 4. Fields description.

Looking for the key `MON_PEDESTALS_DAT` in the `COND_TABLE_META`, returns the row shown in Table 3[1]. From this table one can tell that this table has 7 fields (in Table 2 we show only part of them), is filled by a process called DQM (Data Quality Monitor) [7] and contains crystal pedestals. The remaining column is discussed below.

The meaning of each field in the `MON_PEDESTALS_DAT` is contained in the table `COND_FIELD_META`, as shown in Table 4. For example, the field `PED_RMS_G1` contains the RMS of the pedestal for gain 1. Programs aiming to plot these data will label the axis as "Ped. RMS1 (ADC Ch)" and will limit the axis between 0.5 and 0.9 ADC Counts[2].

A very similar concept is used to identify readout channels. Any physical channel in the database has been assigned a unique logic identifier. The same object identifier may correspond to different sets of coordinates, depending on the context. Each set is called a *view*. For example, a given crystal in the barrel can be regarded as an object having two coordinates (η, ϕ) where η represents the pseudo–rapidity and ϕ the azimuthal angle in a physics–related *view*, as well as an object with coordinates (SM, Ch), from the point of view of the read–out electronics, where SM is the identifier of a modular structure called super–module and Ch is the identifier of the readout channel within the super–module. Even if it is a unique object it has more than one

[1] In fact field names are slightly different. We shorten them here for space reasons.
[2] Even in this case, we slightly modify the table for space reasons.

TABLE: CHANNELVIEW

LOGIC_ID	NAME	ID1	ID2	ID3
1011060006	EB_crystal_angle	1	115	NULL
1011060006	EB_crystal_number	6	6	NULL

Table 5. Two different views of the same channel.

TABLE: VIEWDESCRIPTION

NAME	DESCRIPTION	ID1NAME	ID2NAME
EB_crystal_angle	Crystals in ECAL barrel super–modules by angle index	ieta	iphi
EB_crystal_number	Crystals in ECAL barrel super–modules by number	SM	crystal_number

Table 6. The meaning of the coordinates in two different views.

view. It is then represented always via its logic identifier in the database, but we map each logic identifier to physical views using the `CHANNELVIEW` table.

The `CHANNELVIEW` table has five important fields: the logic identifier `LOGIC_ID`, the name of the view, and three integer coordinates `ID1`, `ID2` and `ID3`. Each object in our detector can be, in fact, represented by at most three integer coordinates. The meaning of these coordinates is given in a separate table called `VIEWDESCRIPTION`. In this table each view is described with a record containing a textual description and the names to be assigned to each coordinate.

As an example, in Table 2, the channel identifier is represented by the `LOGIC_ID`. The channel identified, i.e., by 1011060006 corresponds to a given readout channel, whose coordinates are in the `CHANNELVIEW` table, as shown in Table 5.

In this table there are two possible views. In one of them, called `EB_crystal_number`, the object 1011060006 has coordinates $(1, 115)$, while in the other, called `EB_fe_crystal_number`, the object has two coordinates $(6, 6)$. The meaning of the coordinates is given by `VIEWDESCRIPTION` as shown in Table 6.

From this table one can see that, in the `EB_crystal_angle` view, the object 1011060006 corresponds to a crystal in the ECAL barrel super–modules identified by the angle indexes `ieta` and `iphi`, corresponding to the (η, ϕ) coordinates of the crystal in the CMS reference frame. On the other hand, from the point of view of the electronics, the same object is better identified by the index of the super–module `SM` and the channel number, relative to the latter, `crystal_number`.

Based on what we have just discussed, we can now explain the meaning of column `LID_NAME` in Table 3. That column contains the default view name to the data contained in the corresponding table.

Besides those already discussed, such a pattern has several other advantages: it keeps the formation of the queries simple enough; the queries themselves can be formed algorithmically; the addition of new tasks and new types of data does not imply a schema migration; the size of a single table is shortened.

5. Performance Considerations

During the design of the schema, performance and scale tests were done to ensure that the application would satisfy the long-term requirements of ECAL. For each type of data, growth rates of the data are constant and well-known, enabling us to simulate the growth of the database.

It was assumed that the database would have to store data for one year's worth of CMS running before older data would be archived. Knowing the limits to the database size allowed us to focus our performance testing.

Several different schemas were tested to learn how read and write performance would scale as a function of number of channels or number of IOVs in the database. These schemas differed in how the IOV timestamps were stored and indexed, and how the payload data would be stored. We considered both to include IOV's timestamps inline with the data, or in a separate table with an integer ID acting as a key (the winning method). For payload storage we tested three schemas:

(i) data organized in columns with channels as rows (the chosen one);
(ii) all channels together in CLOB or BLOB columns;
(iii) channels organized in columns with data as rows.

Two access patterns were tested: selecting all channels for a type of data given a timestamp t, and selecting a single channel at time t. The matrix of possibilities above was exercised and guided the design of the current schema.

These performance benchmarks were absolutely crucial to developing a schema that would work for our application. As an example, the first iteration of our schema for storing the pedestals was found to take 20 seconds to query all ECAL barrel channels (61200) at time t with only 10 days worth of ECAL running stored. This result scales linearly with time, leading to the conclusion that if we had one year of data the query would take 12.2 minutes to execute. It may seem surprising that such a schema could even be considered, but at the time we were only working with small portions of the detector (3600 channels) and small databases, so the scaling nightmare was not immediately apparent. With the results of the performance testing, we arrived at a schema which could do the same query 55 times faster when executed on a database with 1,000 IOVs, and more importantly we found that the scaling was constant with more IOVs. Calculations based on the scaling factors obtained in testing showed that the same query on a one-year database could be done in 0.9 seconds, which is nearly 800 times faster than what we started with.

6. The Web Based Monitoring.

Database content browsing is achieved by means of the so–called Web Based Monitoring (WBM) by which a user can search for data using a common web browser. The system makes use of the Apache Tomcat application server [8], developed by the Apache Software Foundation, that executes Java servlets and renders HTML pages. Servlets interact with the database via JDBC calls and renders web pages on the client side.

For ECAL, WBM allows users to search for runs with a simple form. Runs can be searched by means of their number or by date. Users can specify the run type or the detector components to be included in the DAQ in the resulting run list.

Selecting a run, a summary page shows all the information stored in the monitoring task tables, as well as in the configuration tables, in a suitable way. Software is organized in such a way that the main page is common to all views and is composed of a collection of *PageComponent* objects. A *PageComponent* is a part of the page, self–contained in a box frame, whose content is usually an HTML table constructed with the information found in the main table of each monitoring task associated to the given run. *PageComponents* can be freely combined to produce the wanted page layout. Data provided by the task are shown to the user by a simple click on the corresponding button(s) for each entry in the *PageComponent* table. According to the data type, plots can be produced as graphs, histograms or 2–D maps.

The data needed to build the right plot are retrieved from the database using always the same tool that obtains the necessary information from their description, avoiding duplication of

software and greatly reducing time needed for maintenance. In fact, the addition of a new task or the introduction of new page layouts does not change the way in which plots are produced, thanks to the "Item Description" pattern. Plots are produced on the fly, but a caching mechanism avoids unnecessary queries if data have already been extracted within the last few hours. Beside each plot we provide a link with the data used to produce it, in text form as well as a ROOT [9] Tree.

The WBM also provides a tool to make history plots of any value in the data tables. The list, the type and the meaning of each value is again obtained by the description table.

Moreover, the last obtained environmental data are shown as color coded maps, so that shifters can easily identify channels with a bad behavior looking for red spots in the map or in pale color. A pale color indicates that the corresponding value is not current. Details about each channel are always available as properly formatted text as well.

7. Current experience.

The current implementation of services for database access is very satisfactory, both for users and developers. Due to the pattern employed in the definition of the tables developers can develop generic applications that automatically retrieve all the information needed to provide the user data in the proper format.

The services have been exercised during 2008 and are currently under test while the CMS experiment has been undergoing commissioning. The frequency of updates, readings and the size of the transactions are almost independent on the type of activity, so we can reasonably extrapolate the behavior of the system at the time when there will be p–p collisions from LHC.

From summer to autumn 2008 the database size grew about 3.3 GB/month. Taking into account that the duty cycle during this time was about 1/3 of the one foreseen during LHC running, we can estimate that the database size will grow about 10 GB/month, i.e. 66 GB/year, assuming 200 days of operation of LHC per year.

8. Conclusion

The database services for the control and monitoring of the electromagnetic calorimeter of the CMS experiment are already well established and ready for p–p collisions. Users interact with the database by means of three classes of applications that, in turn, make use of the "Item Description" pattern employed for the table definition. The estimated rate of growth of the database, obtained measuring the size of database after real data taking during commissioning, is comfortable, being less than 100 GB/year.

References

[1] S. Chatrchyan et al., "The CMS experiment at the CERN LHC", (2008) JINST 3 S08004.
[2] G.L. Bayatian et al, "CMS ECAL Technical Design Report", (1997) CERN/LHCC 97-33
[3] P. Adzic et al. "Intercalibration of the barrel electromagnetic calorimeter of the CMS experiment at start-up", JINST 3 P10007 (2008)
[4] R. Arcidiacono et al, "ECAL front–end monitoring in the CMS experiment", these proceedings
[5] `http://itcobe.web.cern.ch/itcobe/Services/Pvss`
[6] `http://www.qtsoftware.com/products`
[7] G. Della Ricca et al., "Data Quality Monitoring for the CMS Electromagnetic Calorimeter", Nuclear Physics B - Proceedings Supplements, (2007) 172, pagg. 253-256
[8] `http://tomcat.apache.org`
[9] `http://root.cern.ch`