Conceptual description of the CMS Online Database

Mindaugas Janulis (Vilnius University) on behalf of the CMS collaboration
E-mail: Mindaugas.Janulis@cern.ch

Abstract. The CMS experiment at the LHC at CERN is made of many detectors. The detector properties are controlled and monitored continuously by the Detector Control System (DCS) in order to ensure the stability and precision required by its design. This leads to a large data volume to be accessed continuously by people on shift, experts, automatic monitoring tasks, detector configuration programs for trigger and data acquisition systems and offline data reconstruction programs.

1. Introduction
The Compact Muon Solenoid (CMS[1]) experiment is one of several experiments working on research on fundamental physics at the Large Hadron Collider (LHC[2]) at CERN[3]. The detector properties are controlled and monitored continuously by the Detector Control System (DCS) and stored in the CMS Online Database.
The CMS Online Database is the master database located at the CMS experiment site. It stores the configuration of the detector and the condition data, produced by the systems like DCS, data acquisition (DAQ) and trigger data. It is implemented in the form of a purely relational Oracle database. A subset of the online data is streamed into the offline database for event reconstruction using Oracle Streams technology.
Data received by DCS are stored into the database so they can be accessed by various detector monitoring applications (i.e. Web Based Monitoring). The main challenge for the CMS Online Database is to efficiently archive and provide fast access to large data volumes of parameters (temperatures, currents, voltages, humidity values, software alarms, etc.). While this would normally be considered a classical Data Warehouse solution, in our case only the most recent values are of significant interest, older data (“history data”) are only accessed occasionally. These values are needed to observe the detector state changes.

Since history tables are big, it is difficult to serve the most recent values from these tables without facing performance issues. Therefore, we provide Last Value (LV) tables, which contain only the most recent values. A LV table is created automatically for a particular channel type (class of channel/device) and is updated every time new data are received from DCS. The history tables often contain millions of records whereas the LV tables mostly contain only thousands of rows.
In addition to LV tables a redundant data filtering mechanism has been implemented. When the state of some parameter changes in the detector, DCS passes it to the PL/SQL procedures, which decide if and where the received data should be stored in the CMS DCS Online Database. The received value is compared with the previous one in the LV table. If they are equal (or the difference between old and new value is not greater than a predefined dead band), it is not stored. The LV mechanism prevents the database being polluted by redundant data from misconfigured applications. The data filtering procedures are part of the extended CMS DCS DB schema.

4. Preparation for large data volume
The most requested data are provided by LV tables. However, all the historical data must be kept. The tables which hold these data can easily grow to millions of rows. Despite of that, a good query performance is required. But fast access to such big tables is not trivial. Partitioning the table allows to avoid unnecessary table/index scan operations by performing SQL operations only on a subset of the total data in a table. Thus it increased the query performance by a factor of 15. The queries to access historical data, always include a time range predicate. Therefore monthly time range partitions were created on the most crucial tables.
When the new record is inserted, it is automatically assigned to a particular partition, depending on the partition key value. When the data are accessed, only partitions corresponding to the provided time period are scanned (Figure 3).

5. Original PVSS vs. CMS schema

One of the biggest applications which store values into the database is using the PVSS (Supervisory Control and Data Acquisition software by ETM [4]) framework. In the standard PVSS DB schema data are identified by an internal PVSS ID. In CMS we decided instead to use a schema using a static identifier which is mapped to the PVSS internal one. This way we avoid remapping in case the internal identifier changes.

To make the DB schema self-explaining in CMS we introduced one table per data type, whereas the standard PVSS schema uses mainly two tables to store all information (Figure 4). This also reduces the size of the tables and improves query performance.
6. PVSS intrinsics
The device data in PVSS are structured as so called Data Points (DP) of a predefined Data Point Type (DPT). PVSS allows devices to be modeled using these DPTs/DPs. As such it allows all data associated with a particular device to be grouped together rather than being held in separate variables. A DP contains one or more Data Point Elements (DPEs - items of process information within a device oriented DP). Every DPE corresponds to a value/state.
A DPT also could be imagined like a class in the Object Oriented Programming (OOP). Then DP could be like an instance of the class (object, representing a real device within the control system) and DPE like a member of the class.

![Figure 5. Simplified example of the DPT/DP/DPE representation in the DB.](image)

In the extended CMS DCS DB schema one table per DPT is created and one column per DPE. A history table holds rows with different DPs, but they correspond to the same DPT. The information is stored that each element’s value is in a separate row. This is shown in the Figure 5.

7. Conclusion
The CMS Online DB stores large amounts of data that are received mainly from the DCS. In order to efficiently archive and provide fast access to such data, modifications were introduced to the standard DB schema. This allowed to achieve the defined requirements and prepare for the first data taking.

References
[1] CMS, http://cms.cern.ch
[2] LHC, http://lhc.web.cern.ch
[3] CERN, http://cern.ch
[4] ETM, http://www.etm.at