E10-2014-103

E. P. Akishina, E. I. Alexandrov, I. N. Alexandrov, I. A. Filozova, V. Friese [1], V. V. Ivanov, W. Müller [1], P. V. Zrelov

CONCEPTUAL CONSIDERATIONS FOR **CBM** DATABASES

[1] GSI, Darmstadt, Germany

Акишина Е. П. и др. E10-2014-103
Концепция баз данных эксперимента CBM

Рассматривается концепция баз данных для эксперимента CBM. Для этой цели произведен анализ существующих баз данных для больших экспериментов на LHC в ЦЕРН. Были проанализированы особенности различных СУБД, в том числе реляционных и объектно-ориентированных, которые используются в указанных физических экспериментах. Предложен набор баз данных и сценарий их использования для эксперимента CBM.

Работа выполнена в Лаборатории информационных технологий ОИЯИ.

Сообщение Объединенного института ядерных исследований. Дубна, 2014

Akishina E. P. et al. E10-2014-103
Conceptual Considerations for CBM Databases

We consider a concept of databases for the CBM experiment. For this purpose, an analysis of the databases for large experiments at the LHC at CERN has been performed. Special features of various DBMS utilized in physical experiments, including relational and object-oriented DBMS as the most applicable ones for the tasks of these experiments, were analyzed. A set of databases for the CBM experiment, DBMS for their developments as well as use cases for the considered databases are suggested.

The investigation has been performed at the Laboratory of Information Technologies, JINR.

Communication of the Joint Institute for Nuclear Research. Dubna, 2014

CONTENTS

INTRODUCTION

The CBM (Compressed Baryonic Matter) experimental facility that is being built at GSI (Darmstadt, Germany) at the accelerator complex of antiprotons and heavy ions FAIR (Facility for Antiproton and Ion Research) is intended for studying the properties of superdense baryon matter generating in 8–45 GeV/nucleon proton-nuclear and nucleus–nucleus collisions [1, 2].

Figure 1 presents a dielectron version of the CBM installation intended for research on short-living vector mesons and charmonium decaying into an electron–positron pair. Inside the dipole magnet there is a target and a coordinate tracking system STS (Silicon Tracking System) containing 8 stations of 300 μm two-way strip detectors. Together with the dipole magnet, STS is used to reconstruct charged particle trajectories and to determine their momenta. The Cherenkov detector RICH (Ring Imaging CHerenkov) and the Transition Radiation Detec-

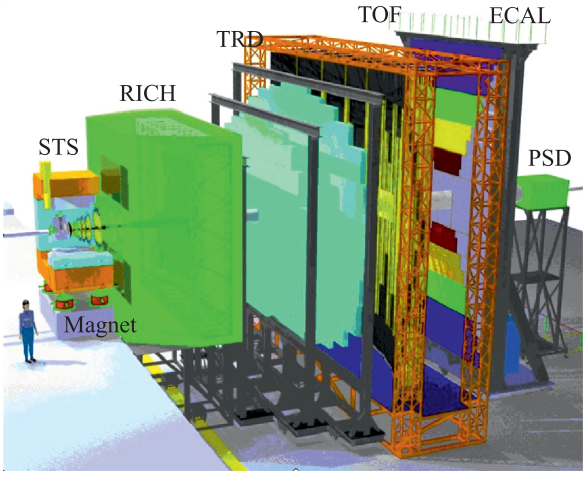

Fig. 1. Dielectron version of the CBM experimental setup

tor (TRD) should provide a reliable registration of electrons/positrons with pulses of more than 1 GeV/s. The time-of-flight detector (TOF) constructed on the basis of Resistive Plate Chambers (RPC) is intended for hadron identification in a wide energy range. The Electromagnetic CALorimeter (ECAL) is used for electron/photon identification. The Projectile Spectator Detector (PSD) is intended for the reaction plane determination.

A di-muon version of the CBM installation is for the research in short-living vector mesons and charmonium decaying into a di-muon pair. In this scheme the RICH detector is replaced by MUCH (MUon CHamber) which is intended for muon identification by means of a maximal suppression of the hadron background (by using a set of ferrous absorbers) and reconstruction of their trajectories with the help of the MUCH coordinate planes.

The CBM detectors should provide a reliable electron/positron or muon identification, a high-level suppression of the hadron background, a charged particle trajectories reconstruction under the conditions of intensive streams (10^7 collisions per second), and high multiplicity of secondary particles (from 100 up to 1000 particles from one nucleus–nucleus collision). Here, as opposed to the traditional approach applied in HEP which is grounded on selecting useful information with the help of complex high-performance trigger systems for its subsequent offline analysis, all this processing should be done in a real time of experiment.

In spite of the fact that maximally filtered information will be selected, however, taking into consideration the high beam intensity and the high multiplicity of secondary particles, enormous data on signal events are expected which will be accumulated on information carriers for their subsequent analysis.

At present, the CBM collaboration is at the stage of transition from research and tests of the prototypes of various detectors and components of the CBM installation to their production. Taking into account a wide spectrum of detectors and components of the setup as well as their structural complexity and high price, a close-up inspection and in-depth control of the manufacturing process are required.

Summing up, there is a need for the development of a database complex for the CBM experiment. This work analyzes the experience gained in the database development for large physical experiments in HEP. On this basis, recommendations were worked out on the compound, intention, and instruments (including recommendations on DBMS) for the development of the database complex for the CBM experiment.

1. DBMS REVIEW

Here we consider typical DBMSs (DataBase Management Systems) that are used in large physical experiments, as well as their advantages and disadvantages.

1.1. Relational and object-oriented DBMSs

Now the most popular approach to the creation of the information systems is building the client-server applications that communicate with a relational DataBase (DB) [3]. Usually, applications providing an interface to a database are implemented within the object-oriented programming (OOP) paradigm. It raises the need for mutual agreement between the database records and application objects. Most frequently RDBMSs (Relational DataBase Management Systems) are used.

Advantages of the relational approach

1. Simplicity and easy to understand by the end-user. Only once used information structure table. Information principle acts as follows: all the content database is represented by one and only one way — explicit assigning values of attributes in tuples of relations. There are no pointers that link one value to another.

2. Strict design rules, based on mathematical apparatus (theory of sets, mathematical logic, relational algebra).

3. Full data independence. When the structure of a relational database is changed, the changes in the applications are usually minimal.

4. Support of the integrity constraints. The presence of relational algebra allows one to realize the declarative programming and declarative integrity constraints, in addition to the navigation (procedural) programming and procedural verification conditions.

5. No navigating data manipulation. There is no need of their specific physical organization of databases in external memory for building queries and creation of application.

Disadvantages of the relational approach

1. A relatively low access speed and a large amount of external memory.

2. The difficulty of understanding the data structure because of a large number of tables from the logic design.

3. Not always the subject area can be represented as a set of tables.

Object databases have appeared due to the vital need in solving the problems related to the processing and storage of the complex multiple data and unstructured information (text, images, music). Object-Oriented Databases Management Systems (OODBMS) are well suited for distributed computing [4].

OODBMS advantages

• Support of the OOP approach that removes many restrictions specific to the RDBMS. Depending on the complexity of the data, benefit on the performance may reach tens and thousands of times as compared with the RDBMS.

• To access the data a special query language is not required, because access is directly to the objects.

OODBMS disadvantages

• Weak data manipulation tools. In OODBMS there are declarative languages for the data selection and manipulation, in particular, Object Query Language (OQL). The negative feature of OQL is its excessive complexity. While relational algebra operators deal with only one type of structure — with a relationship, OQL uses many different structures — sets, multisets, and lists. The result of this complexity is a large restriction of the automatic optimization of the expressions.

• Limited support for data integrity. The relational model has a mechanism of automatic integrity support. The object model does not have a similar option: instead of the declarative description of the constraints the developer must provide the relevant checks in the business logic of the application.

• The absence of a theoretical basis and mathematical tools. But the relational model is based on a powerful foundation of relational algebra.

• The absence of a common standard object model.

• Mostly commercial products.

1.2. Relational databases in physical experiments

Among a large number of RDBMS, MySQL and ORACLE are most frequently used in the physical experiments. Despite repeated attempts of the usage of the free DB (mainly MySQL) in physical experiments, a good alternative to ORACLE has not been found. However, it is still advisable to start the development using the free DBMS. And in the case of the inadequacy to the announced requirements, it is necessary to switch to the known reliable variant (for example, ORACLE). Among all free DBMS which could be used as alternative to ORACLE, we should pay attention (except MySQL) to PostgreSQL [5, 6].

PostgreSQL advantages

1. Reliability and stability at very high loads.
2. Platform independence (FreeBSD, Linux, Windows).
3. High level of the correspondence with the standards ISO/ANS.
4. Interfaces for Java, Perl, Python, Ruby, C, C++, PHP, Lisp, Scheme, Tell, Tcl.
5. Scalability (ability of the system to cope with the increased workload when adding resources).
6. High performance.
7. Database support of the almost unlimited size.
8. Advanced fault tolerance.
9. Advanced security model.

PostgreSQL disadvantages

1. Difficulty in maintenance by sysadmin.
2. Poor performance at low load (compared to MySQL).

3. Limitations in the distribution of access rights.
4. Paid development tools.

1.3. OODB in physical experiments

Successful experience with ODB Objectivity/DB is known in the BaBar experiment (SLAC). Each registered physical event in the experiment has the types of objects in the database.

Estimations were carried out for object databases in the LHC experiments, but in the end they were rejected. The Condition DB for LHC Experiments is implemented using the relational model. In the ATLAS experiment the Configuration DB was created on the basis of their own OODB.

1.4. NoSQL Data Warehouse

Huge data amounts force developers and business to seek for alternative relational databases. Together, these technologies are known as "NoSQL database" [7]. The main advantages of NoSQL approach are horizontal scalability, flexible data model, support of a large number of users, management of big data volumes, zero-level administration, and economics (relation between data volume processing and price). The main disadvantages are maturity (now NoSQL is an interesting prospect for developers, but not enterprises), support (most NoSQL systems are open source projects), weak support of the analytics and business intelligence, administration (a lot of skill to install and a lot of effort to maintain), and expertise (now it is far easier to find experienced RDB programmers or administrators than a NoSQL expert).

Horizontal scalability is an automatic distribution of data across multiple servers (distributed databases, support of a few data centers and the ability to add new machines to a running cluster transparently to applications).

Data and queries model defines a logical database structure, transitions between the states of the database (data modification ways), and methods of data extraction from a database, description means of the correct database states.

Storage system provides solutions to the organization of the safe storage of information resources and ensures access to them (how data is stored within the system).

Nowadays the most popular NoSQL systems are Cassandra, CouchDB, MongoDB, Kyoto Cabinet, HBase, Riak, Scalaris, Voldemort, Neo4J.

The CMS collaboration sees a new trend of wrapping up the standard 3-tier architecture based on traditional RDBMS back-end with another software layer that provides additional functionality to data management tools. Three NoSQL solutions — MongoDB, CouchDB, and KyotoCabinet — are successfully deployed into production environment of the CMS experiment. They found a niche in CMS software stack and nicely co-exist with traditional databases. The schema-less feature of MongoDB and CouchDB allows one to abstract existing data-services and

aggregate information from them. The map-reduce operations provide a new view over the data and are used for generating summaries, job progress, and output reports. Key-value file-based store (Kyoto Cabinet) can efficiently substitute static database where data compression is valuable [8].

2. DATABASES IN LHC EXPERIMENTS AND RECOMMENDATIONS

LHC experiments are ones of the last started big experiments, so their solutions can be interesting for CBM. ATLAS and CMS are two great projects in LHC. The analysis of their DBs solution is briefly presented below.

2.1. CMS databases

This subject domain includes the information to physically set up the detector, information about the subdetectors construction, information required to bring the detector in any running mode, and information about detector conditions [9].

All physical parameters and data in the experimental research have a direct relation with the detector and subdetectors construction. A unique device consists of a large number of different integrated devices. In this case, a zero level of databases structure must be the Equipment DB that stores structured data about all detector parts as equipment elements. The next level of the structure is the Construction DB which stores information about relations between different equipment elements. Some equipment elements have special committed information, such as test results, results and conditions of calibration process. The next levels are the Configuration DB and the Conditions DB. Thus, the full databases structure and hierarchy are presented by:

Equipment management database. Holds structured data about all detectors parts as equipment elements.

Construction database. Holds all information about relations between different equipment elements.

Conditions database. Holds all information about detector conditions (data on operating conditions).

Configuration database. Holds all information required to bring the detector in any running mode.

Different types of databases make different demands for interfaces. Thus, the usual request for the Equipment DB interface is worldwide distribution and human oriented (web interface), but the most common interface for the Conditions DB is API (Application Programming Interface).

The Equipment DB is intended for storage of the information about equipment of various parts of the CMS detector. It should be independent of the concrete equipment as much as possible, available to store tests results and accompanying documentation. It should provide the opportunity of the new data input and information search on many criteria. Information about equipment should be committed with unstructured data like document files, images, drawings, figures, etc. A volume of such kind of data is about dozens of gigabytes. It is necessary to have data storage committed with databases in this case. This data storage must provide the following features: high-speed end easy data access, relations with mass storage system, and reserve backups. For detailed information about the subsystem of Equipment Database, see Appendix A.

2.2. ATLAS databases

The LHC computing models are mostly focused on the problem of managing the petabyte-scale event data that are kept in a file-based data store, with files catalogued in various databases. These event store databases are an integral part of the LHC computing models. In addition to the file-based event data, LHC data processing and analysis require access to large amounts of the non-event data (detector conditions, calibrations, etc.) stored in relational databases.

In ATLAS there are only few database applications that have to be distributed [10, 11]: Trigger DB, Geometry DB, Conditions DB, and Tag DB. ATLAS developed the Trigger DB for central ("online") operations. A small subset of the whole database is distributed on the Grid in SQLite files for use in Monte Carlo simulations. To manage the detector description constants ("primary numbers"), ATLAS developed the Geometry DB with contributions from LHC Computing Grid (LCG). It is the first ATLAS database application that was deployed worldwide. It is distributed on the Grid in SQLite replica files. The Conditions DB was developed by LCG with ATLAS contributions. The LCG technology for Conditions DB is called COOL. The Conditions DB is the most challenging database application. It is a hybrid application that includes data in RDBMS and in files. Conditions data are distributed worldwide via Oracle Streams and via files. The ATLAS Tag DB stores event-level metadata for physics (and detector commissioning). It was developed by LCG with ATLAS contributions. It is distributed worldwide in files and also 4 TB are hosted at select Oracle servers.

ATLAS Geometry DB. Due to the differences in requirements and implementation, ATLAS Geometry DB is separated from the Conditions DB to keep static information, such as nominal geometry. Only the time-dependent alignment corrections to Geometry are stored in the Conditions DB. Such separation of concerns resulted in a moderate data complexity of the Geometry DB. The update frequency of the Geometry DB is "static", i.e. upon request, when the geometry corrections

or updates become necessary. The database is accessed via a low-level common LCG database access interface called Common Object-Relational Access Layer (CORAL). A typical data reconstruction job makes about 3K queries to the Geometry database. The master Geometry DB resides in the "offline" Oracle, where it is not used for production access.

ATLAS Conditions DB. Driven by the complexity of the subdetectors requirements, ATLAS Conditions DB technology is hybrid: it has both database-resident information and external data in separate files, which are referenced by the database-resident data. These external files are in a common LHC format called POOL. ATLAS database-resident information exists in its entirety in Oracle but can be distributed in smaller "slices" of data using SQLite. Since Oracle was chosen as a database technology for the "online" DB, ATLAS benefits from uniform Oracle technology deployment down to the Tier-1 centers. Adoption of Oracle avoids translating from one technology to another and leverages Oracle support from CERN IT and WLCG 3D Services. Historically, ATLAS separated conditions database instances for Monte Carlo simulations and for the real data. The two instances still remain separate to prevent accidental overwrite of the Conditions DB for real data. Both Conditions DB instances are accessed via common LCG interface COOL/CORAL. This approach is similar to the CMS Conditions DB partitioning by usage. The complexity of the ATLAS Conditions DB data for simulations is high. According to a representative snapshot of February, 2010, the instance has 2,893 tables in four schemas. The total number of rows is 842,079 and the data volume of the SQLite replica is 376 MB. There are additionally 247 MB of data in 1049 POOL/ROOT files grouped in 25 datasets. The update frequency is "static"; i.e., the database is updated upon request typically in preparation for major Monte Carlo simulations campaigns. All conditions data for Monte Carlo simulations is replicated on the Grid via files (the full snapshot in SQLite plus the external POOL/ROOT files and their catalogs). The ATLAS Conditions DB for real data has a very high complexity. The schema count is determined by the number of ATLAS detector subsystems: 15 subsystems each having two schemas ("online" and "offline") plus one inactive combined schema (to be decommissioned). The total number of rows is 761,845,364 and the Oracle data volume is 0.5 TB. There are additionally 0.2 TB in POOL/ROOT files grouped in 48 datasets. The Conditions DB for real data is updated continuously. Because of the large volume, use of the full database replica on the Grid is not practical. Only the required "slices" of the ATLAS Conditions DB data are distributed on the Grid. To process a 2 GB file with 1K raw events, a typical reconstruction job makes about 11K queries to read more than 70 MB of database-resident data (with some jobs read tens of MB extra) plus about ten times more volume of data is read from the external POOL files.

Support for on-demand data access — a key feature of the common Gaudi/ Athena framework — emphasizes the importance of database interfaces for LHCb and ATLAS experiments. On-demand data access architecture makes Oracle use straightforward. In the on-demand data access, environment having a redundant infrastructure for database access turns out to be advantageous. The redundancy is achieved through common LHC interfaces for persistent data access, which assure independence from available technologies (Oracle, SQLite, Frontier). No changes in the application code are needed to switch between various database technologies (see Fig. 1). In ATLAS, each major use case is functionally covered by more than one of the available technologies, so that we can achieve a redundant and robust database access system.

In addition, various tools can be built on top of the interfaces. For example, since the large volume of ATLAS Conditions DB prevents use of the full database replica on the Grid, an advanced "db-on-demand" tool was developed to produce "slices" of the required conditions data for the Grid jobs.

ATLAS Configuration DB. It is recommended to use an objective approach for this database. However, the existing OODBMS have a number of serious disadvantages for design database. Therefore, it is suggested to use developments of ATLAS, which has its own confDB [12, 13]. Data from ATLAS confDB stores in files of XML formats and allows different data files to be merged into a single database. It includes Motif based GUIs to design database schema and to manipulate data objects. This database is very simple to install and well scalable. ATLAS Configuration DB has a structure based on partition.
The main functions of the partitioning are to:

- parcel out the system into its components;
- allocate these components to activities executed on several independent systems;
- provide a smooth transition between different combinations.

Details concerning ATLAS online Configuration DB are presented in Appendix B.

2.3. Typical use cases of DBs for physical experiments

The analysis of databases of physical experiments, such as ATLAS and CMS, shows that they have similar use cases and close structure of corresponding datebases. Below we present typical use cases for Configuration, Conditions, Component and Event (tag) databases.

2.3.1. Use case of the Configuration DB

1. Developer, operator, and expert create, manipulate, change, and view data.

2. Start-up system reads Configuration data to activate basic online components.

3. Control System reads configuration data to start by means of process manager those processes that should be started for current run.

4. Component which will start processes reading data concerning computer, binary and its parameters that stored in DB.

5. Verification system reads parameters of tests and hardware where they should be started.

6. Front-end components read parameters concerning hardware and software that are in use for the current run.

2.3.2. Use case of the Conditions DB

1. Updates of Conditions DB tables with the interval-of-validity metadata.

2. First-pass processing: continue using direct Oracle access.

3. Reprocessing: continue using the Conditions DB Release.

4. Monte Carlo simulations: continue using the DB Release.

5. User analysis: Grid jobs with large conditions data need: use the Frontier/Squid servers; local jobs with stable conditions data: use the Conditions DB Release.

2.3.3. Use case of the Geometry DB

1. Users with administrator rights write/update data.

2. Simulation/reconstruction/analyses programs use the Geometric data. Version of geometry can be taken into account if needed.

3. Calibration/alignment software uses geometrical data for calibration and alignment data producing.

4. Graphical navigation in detector geometry browsing.

5. Technical coordination support uses component database to get initial data.

6. Detector production and commission software (databases) use component data as initial data.

2.3.4. Use case of the Event DB

1. Physicist reconstructs event offline.

2. Developer checks work of detector in monitoring tasks.

3. Physicist analyzes event online.

2.4. LHC persistency framework for DBs

LHC developed its own persistency framework to simplify work with databases. It includes three components: CORAL, COOL, and POOL.

2.4.1. CORAL

The primary goal of CORAL (Common Object-Relational Access Layer) is to provide functionality for accessing data in relational databases using a C++ and SQL-free API, shielding the user from the technology-specific APIs and removing at the same time the need to submit directly SQL commands. Therefore, CORAL allows the development of software components that can be used without any code modification or conditional constructs against multiple relational technologies. A user is not expected to be an expert in all possible optimization techniques

relevant to a particular relational technology. In fact, a user is expected to be familiar only with the basic concepts of relational database. On the other hand, the CORAL interfaces have been designed in a way that a user is guided towards standard "best" practices in RDBMS programming. Applications which are based on CORAL are easier to monitor and tune.

• Abstraction layer with an SQL-free API to access data stored using relational databases technologies. Support for Oracle, MySQL, SQLite, and FroNtier.

• Used directly or indirectly via COOL/POOL.

2.4.2. COOL

COOL software has been chosen by ATLAS and LHCb to manage their conditions data. "Conditions data" record the state of the detector at the time when events are collected. Conditions data are extremely important because they are needed for the reconstruction and analysis of the events taken using the detector they describe. The main property of conditions data is that they vary with time. Each value or set of values of conditions data describes the state of the detector during a limited lapse of time, and should only be used for the analysis of the events collected during that "interval of validity" (IOV). In addition to their time variation, certain items of conditions data may also exist in different versions. Examples of these "multi-version" (MV) detector conditions include calibration and alignment data, which are computed by processing large sets of raw event and non-event data using algorithms which may exist in different versions. Only one version of each of these conditions data items must be used for the reconstruction and analysis of event data. Conversely, other items of conditions data, such as detector temperatures and voltages from the detector control systems, only exist in one version because they are raw data produced by direct measurements. These are referred to as "single-version" (SV) conditions data.

• Conditions data management. It provides specific software components and tools for the handling of the time variation and versioning of the experiment conditions data.

— Conditions object metadata (interval of validity, version).

— Conditions object data payload (user-defined attributes).

2.4.3. POOL

POOL is a hybrid technology store for C++ objects and object collections, using a mixture of streaming and relational technologies. POOL provides a set of service APIs and isolates experiment framework user code from details of a particular implementation technology. As a result, the POOL user code is not dependent on implementation API or header files, POOL applications do not directly depend on implementation libraries.

This middleware serves as an interlayer between concrete databases and physical programs. Also it allows one to use various databases similar to those which are already maintained as well as the new ones which are not maintained at the

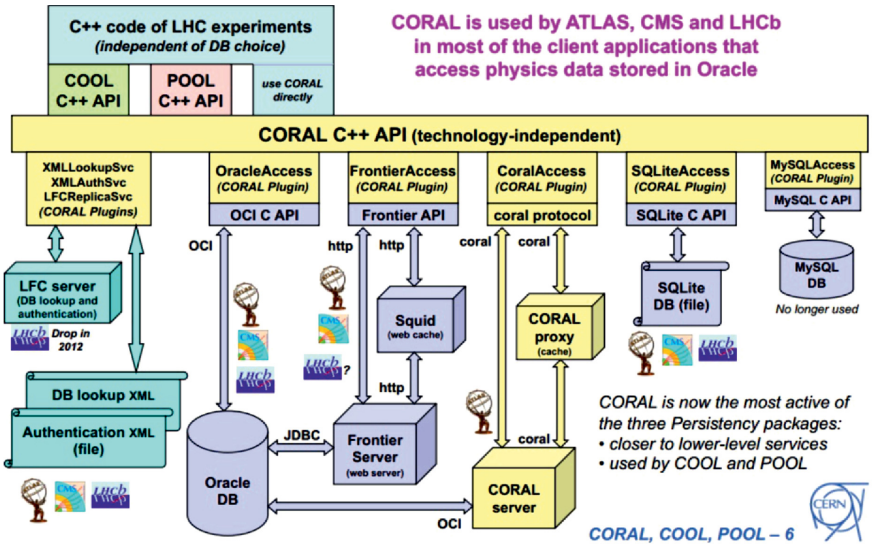

Fig. 2. Scheme of physical programs interaction with databases using middleware

moment. This can be done as the support of the databases is realized in form of plug-ins (see Fig. 2).

• Technologically neutral hybrid data storage for C++ objects, using a mixture of streaming and relational technologies. It provides components that can be used by the experiment to store both their event data and their conditions data.

— Streaming of objects (e.g., to ROOT or relational DBs).

— Object metadata catalogs (e.g., in XML or relational DBs).

Figure 2 shows a scheme of physical programs interaction with databases using middleware.

2.5. Recommendations

In general, the following recommendations can be given. If data consists of short, simple fixed-length fields, then the best approach is to use a relational database. If the data has a nested structure, dynamically resizable, user-defined arbitrary structure (multimedia, for example), their representation in tabular form would be difficult. Thus, the usage of the object model is preferable for distributed databases with a large number of complex relationships: cross-reference links, many-to-many relationships between objects. Alternative to relational databases are also NoSQL warehouses.

Conditions, Component and Event Data tag databases have accordingly simple, slowly changing structures and RDBMS can be the best choice for their

implementation. Experience of ATLAS shows that RDBMS can be used together with other approaches like storing direct streams or Root files (see POOL description).

An object database can be a better choice for Configuration DB, but, for example, the ATLAS online team evaluated many DBMS and persistent managers as candidates for the service implementation. Unfortunately, no one satisfying all requirements was found [14]. There were not found ODBMSs that are scalable and reliable enough as required for the Configuration DB. Pure relational data model does not support inheritance that is needed and some useful data types, like arrays or generation of data access libraries. This is why the ATLAS team developed their own solution that is flexible, light and seems to be applicable for the CBM experiment too.

3. CBM DATABASES

The following main types of data should be maintained within CBM databases: detector production and installation, survey data, event data and metadata, detector geometry, online configuration, run conditions (DCS (Distributed Control System) and others), online and offline calibrations, and alignments data.

Below we present the list of main databases including short description of their goal, content, parameters degrees of freedom, supposed consumers (actors), operating conditions, data sources, and interfaces.

3.1. Configuration database

The Configuration DB will support only the data that is directly required to start and support the DAQ (Data AcQuisition) system in an efficient operating condition. This database should be elastic because not only data but the structures of data can vary. The changes in the Configuration DB are different at the first stages of design and installation. The support of such structures in general database is labour-consuming. Our proposal is to take into account the way chosen in the ATLAS experiment and, if complexity and needs of flexibility are critical, to develop our own DB using ATLAS Configuration DB as the basis for the CBM Configuration DB. One more feature of the Configuration DB is that it should enter into the kernel of DAQ. This means that the Configuration DB must be developed in close collaboration with the DAQ group.

Purpose:

- to store the parameters describing the topology of the DAQ system, hardware and software components, and run parameters.

Contents:

- nominal configuration of detectors, front-end electronics, and data acquisition,
- hardware and software components,

• run parameters.

Parameter degrees of freedom:

• run, run campaign (e.g., nominal settings).

Consumers:

• setup of the configurations of detectors, front-end electronics and data acquisition at run start through appropriate interfaces,

• online computing farm at start-up and possibly during operation.

Operating conditions:

• the Configuration DB is prerequisite for running the experiment; therefore, high availability and fail safety are indispensable,

• high performance for concurrent access from many clients (order of several 10^4) during run start-up.

Data sources:

• others (nominal configuration).

Interfaces:

• to be specified for configuration start-up,

• Web,

• GUI for creating and updating data.

3.2. Conditions database

Purpose:

• to store, retrieve, and manipulate condition data. Conditions data keeps the state of the detector at the time when events are collected. A subset of the data will be used both for online computing (FLES) and for offline computing.

Contents:

• nominal detector geometry and magnetic field,

• nominal configuration of detectors, front-end electronics, and DAQ system,

• alignment parameters and calibration data,

• control data from online monitoring,

• control system data needed for analysis.

Parameter degrees of freedom:

• run/run campaign (e.g., nominal settings),

• real time (e.g., control data),

• version (e.g., alignment and calibration parameters).

Consumers:

• online computing farm at start-up and possibly during operation,

• offline computing farm.

Operating conditions:

• the Conditions DB is prerequisite for running the experiment and offline computing; therefore, high availability and fail safety are indispensable,

• high performance for concurrent access from many clients,

• good performance for concurrent access from many clients during offline data processing (order of several 10^4).

Data sources:

• offline computing (geometries, field, alignment, calibration),

• online computing (alignment, calibration),

• control.

Interfaces:

• FAIRROOT/TsqlServer (offline computing),

• EPICS (control),

• Web.

The Conditions DB comprises lots of tables not connected to each other and written for relational databases. It is also possible to use several different databases. Having learned the experience of large experiments, we propose to consider the use of middleware designed at LHC for solving similar problems.

3.3. Geometry database

Purpose:

• the detector geometry (or description) database provides primary numbers used in building the offline description and tracking its evolution through time and software version.

Contents:

• characteristics of detector and electronics components,

• software version,

• history of evolution.

Parameter degrees of freedom:

• static.

Consumers:

• physicists for data simulation/reconstruction/analyses,

• calibration/alignment software,

• detector production.

Operating conditions:

• medium availability,

• medium performance,

• high fail safety.

Data sources:

• GEANT3/GEANT4,

• constructor drawings.

Interfaces:

• Labview,

• Web,

• FAIRROOT/C++ interface.

3.4. Tag Event database

Purpose:

- to store, retrieve, and manipulate event data from every stages of data processing (Raw, Event Summary Data, Analysis Object Data, etc.).

Contents:

- tag event data in different formats,
- tag event metadata.

Parameter degrees of freedom:

- version of format,
- real time.

Consumers:

- physicists for offline analysis,
- algorithms.

Operating conditions:

- the Tag Event DB is prerequisite for running the experiment and offline computing; therefore, high availability and fail safety are indispensable,
- high performance for concurrent access from many clients,
- good performance for concurrent access from many clients during offline data processing (order of several 10^4).

Data sources:

- detectors,
- simulate data.

Interfaces:

- C++ and other interfaces,
- ROOT-file-based,
- Web.

3.5. Component database

Purpose:

- manage properties of detector and electronics components obtained in a QA process after and during mass fabrication. This includes both mechanical/engineering production data and electronics data (calibrations, chip thresholds, etc.) which are generally kept completely separately. The format and structure of these tables are entirely up to the subdetectors, the central system simply ensures a uniform access and long-term maintainability. The data should be entered at the same time as the detector hardware is delivered for installation.

Contents:

- characteristics of detector and electronics components.

Parameter degrees of freedom:

- static.

Consumers:

- detector groups.

Operating conditions:

- medium availability,
- medium performance,
- high fail safety.

Data sources:

- detector groups,
- laboratory tests.

Interfaces:

- Labview,
- Web,
- FAIRROOT/C++ interface.

CONCLUSIONS

The concept of database complex for the CBM experiment is presented. For this purpose, the analysis of the databases for large experiments (ATLAS and CMS) on LHC at CERN has been performed. Special features of various DBMS utilized in large physical experiments including relational and object-oriented DBMS as the most applicable ones for the tasks of such experiments were analyzed. A set of databases for ATLAS and CMS experiments, DBMS for their developments as well as use cases for the considered databases are presented. Special persistency framework for databases developed at CERN for large LHC experiments is briefly discussed.

As a result of the performed work, the basic set of databases for the CBM experiment, their purposes and structures are develeped and presented.

APPENDICES

A. Subsystem of the Equipment DB in CMS

ME1/1 Database. ME1/1 Database is a part of the EndCap Database designed by the JINR LIT database group for the CMS experiment. ME1/1 Database requirements:

- Necessity to store large volumes of different types of data concerned with equipment such as specification information, results of tests and calibrations, traveler lists (equipment history), etc.
- Data can have different representations: files (test results, documentation), structured data (equipment specifications, relations between equipment elements), combined data (calibrations results).

Abstract description method of the detector elements was applied for designing of the ME1/1 DB structure. The detector elements were distributed over

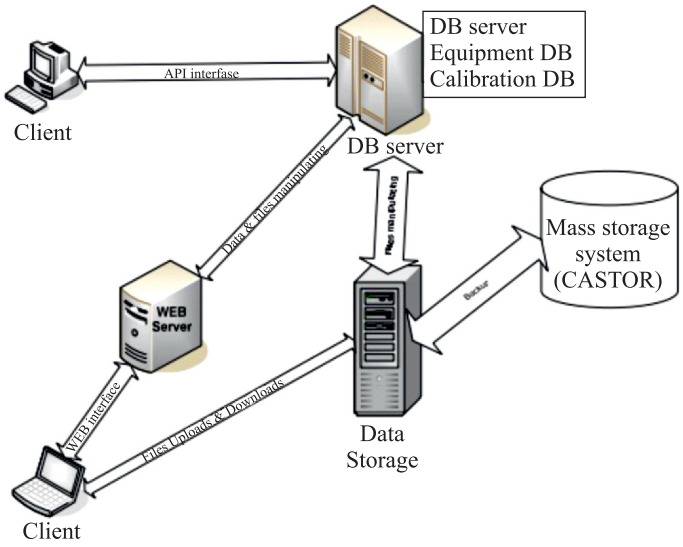

Fig. 3. Databases & storage system structure

groups and subgroups. Common parameters with permanent values were found for each group of the elements. Sets of the specific attributes were defined for each subgroup of the elements. Events, occurring with each equipment element, are fixed at the equipment journals. Equipment journals are defined as especial blocks. The set of the available values is defined for each equipment group. A set of the unstructured data in the form of files can be related with each equipment element, group or subgroup equipment element. Files may be united in the groups (directories). Storage system of the test results is designed as a separate block. The test methods are defined for each equipment group. The sets of the checked parameters are defined for each method. Tests results are grouped by revisions. The files group may be linked with each test from the revision. A common approach to databases & storage system structure is viewed in Fig. 3. This structure is to be realized both at CERN and at JINR in the same way. It will enable the significant part of work on ME1/1 module simulation, testing, and calibration to be carried out at JINR.

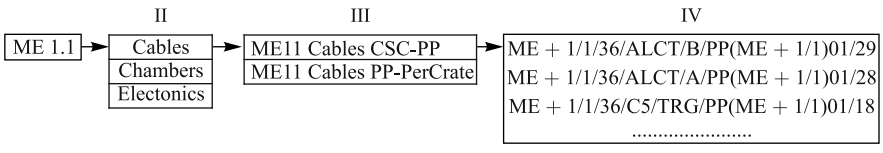

Fig. 4. Equipment hierarchy levels

The DB structure considers the hierarchical relations. It gives the opportunity to store information about relations between the equipment elements (Fig. 4).

At the group level: the general parameters set and values for all equipment of the given group are defined. For example:

Cables: Chamber position [ME 1.1 position], Cable type [ALCTA, ALCTB], Disk [YE+, YE-].

At the subgroup level: the characteristics set specific to the given subgroup is defined. For example:

Cables: ME11 CABLES CSC-PP: Start Point (CSC), End Point (Patch panel), Length.

At the equipment description level: relations to group attributes values are created, characteristics values are defined. For example:

Serial number: ME+1/1/14/LV/PP(ME+1/1)08/2,
Cable type LV,
Chamber ME+1/1/14,
Disk YE+1,
Start point (CSC) ME+1/1/14,
End Point (Patch panel) PP(ME+1/1)08/2,
Length (m) 2.25.

The analysis of the requirements and the objects of the subject domain under consideration allowed one to define a full basic entities set of the designed database. The general parameters with a fixed value set have been revealed for each

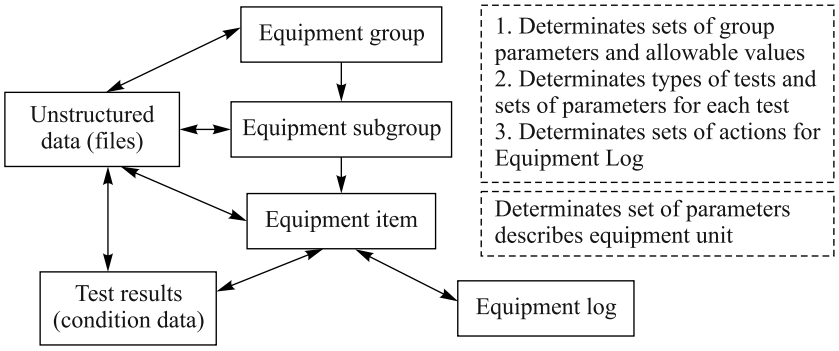

Fig. 5. General logical model of DB

group of elements. Specific attributes sets have been defined for each subgroup. The general logical model of DB is shown in Fig. 5.

In the user's view, all information has a direct link to the detector equipment: unstructured files; group parameters and values; equipment unit properties; equipment logs; tests results.

ME1/1 informational system is realized on the environment:

- Database server: Oracle (provided by CERN IT and JINR LIT);
- WEB Server: Internet Information Server (provided by CERN IT).

ME1/1 User Interface is realized by Web access. Web interface provides the initial filling of database, different access levels for users, information search on different criterions, adding and updating data (Fig. 6).

HE Database. HE Database is intended for the storage of the technical parameters of different detector elements: tubes, index-bars, read-out boxes, megatiles; calibration results of the radioactive wire source, laser, LED and RBX calibration. The following types of data are stored for each tube: length of the plastic tube, length of the metal tube, estimated length of the tube, measured length of the tube, classification tags of the "problem tubes".

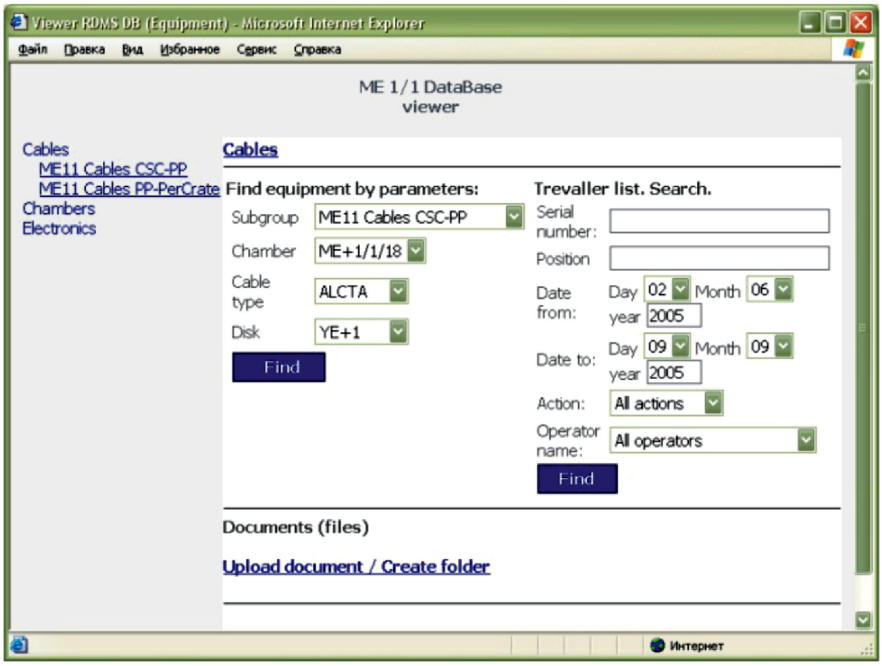

Fig. 6. Web-interface of the Equipment DB

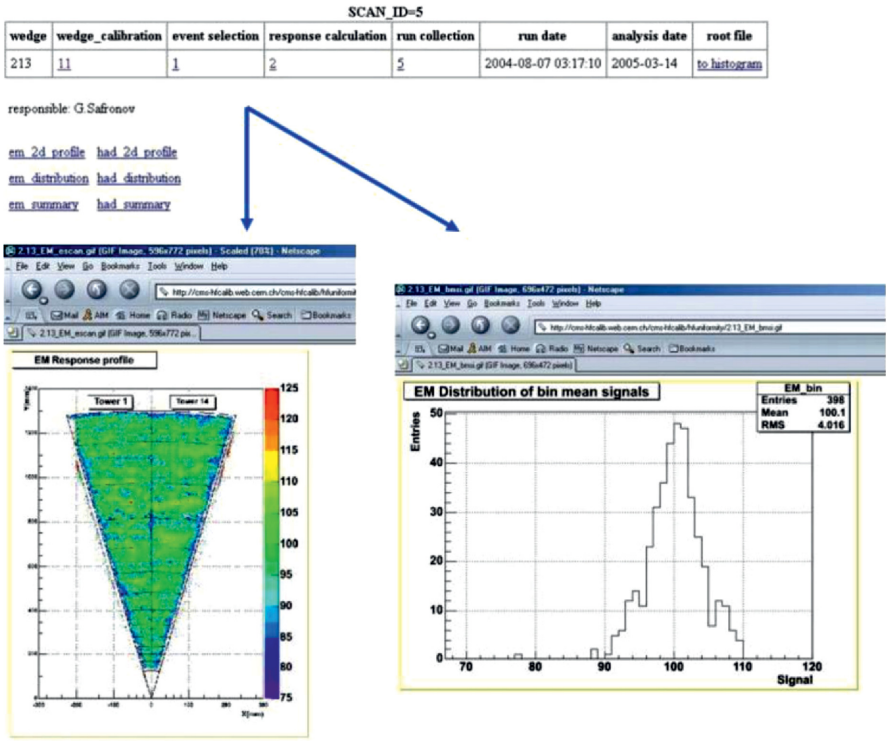

Fig. 7. User interface of the HF DB

HF Database. HF Database contains data for each detector wedge. HF data are wedge calibration, beam wedge calibration, and channel calibration. Database is placed at DEVDB CERN Oracle server. Various stages of the navigation on HF Database by Web are viewed in Fig. 7. Access is realized in two modes: wedge calibration and scan surface.

B. Organization of the Configuration DB in ATLAS

A brief description of the main configuration classes in the Configuration DB of the ATLAS experiment is presented in the table. A partition should contain a set of segments to perform a physics or calibration run. A segment describes a group of applications which are associated with a system or a hardware object. A segment is always controlled by a run control application. When a segment is disabled, all associated applications, hardware objects and included segments are also disabled. The Segment class should be used as a base class to describe the TDAQ systems and detectors (see Fig. 8). Figure 9 shows full scheme of the Configuration DB used in the ATLAS experiment.

Main configuration classes

Class	Description	Instances
BaseSegment (abstract)	Abstract class describing common properties of partitions and segments	
Partition	The Partition object is the configuration description entry point. It contains list of top-level segment, root controller and defines environment variables and default tags for all applications	One object per configuration
Segment	It is the container class including references to objects used to describe configuration. A segment can be enabled to be used, or disabled to be ignored	One per a controlled group of config. objects
RunControl	Describes a run-control application. An RC application always controls a segment	One object per an RC process
Application	Describes a process to be started in certain moment with its parameters. The dependencies between applications can be used to define the order of their initialisation and shutdown	One object per process to be started

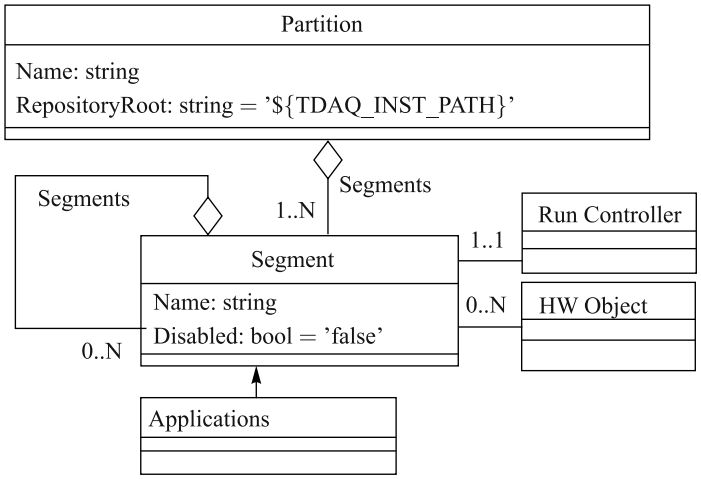

Fig. 8. Segmentation used in the Configuration DB of ATLAS

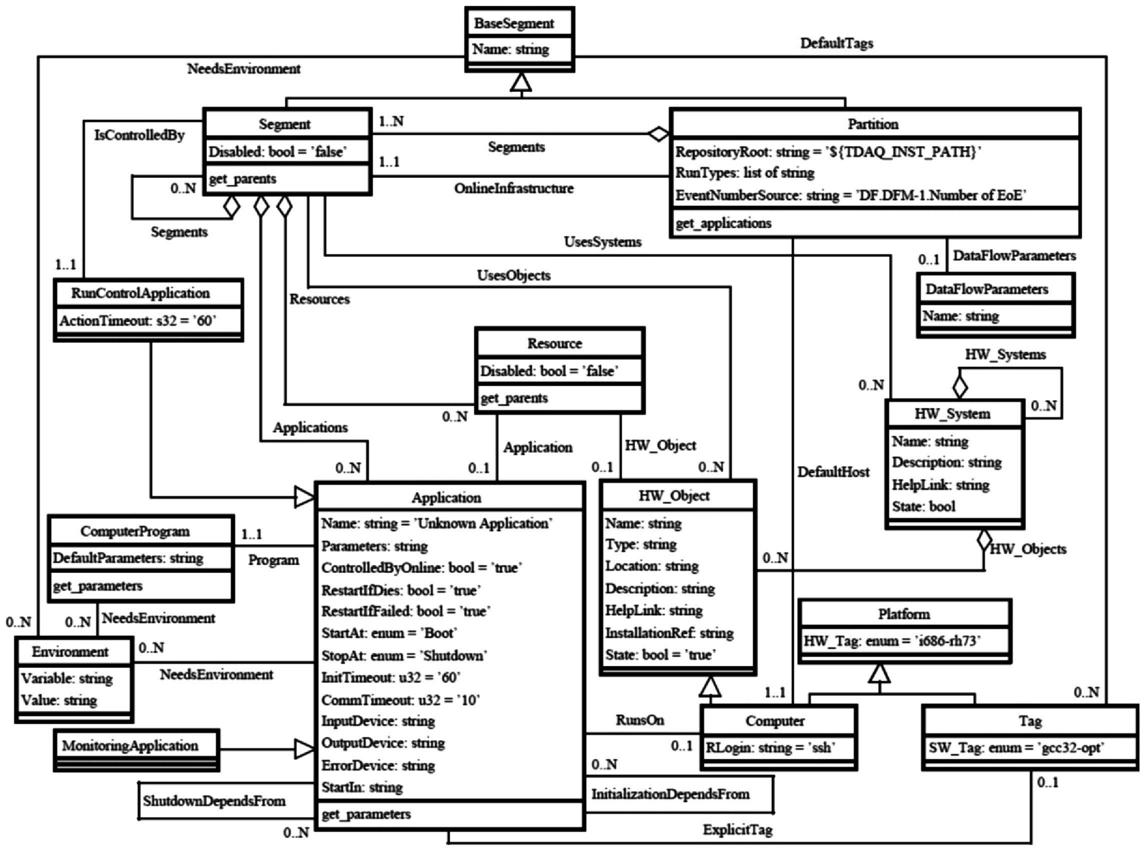

Fig. 9. Full scheme of the Configuration DB in the ATLAS experiment

REFERENCES

1. Compressed Baryonic Matter in Laboratory Experiments. The CBM Physics Book, 2011 (http://www.gsi.de/forschung/fair_experiments/CBM/PhysicsBook.html).
2. Compressed Baryonic Matter Experiment. Technical Status Report, GSI, Darmstadt, 2005 (http://www.gsi.de/onTEAM/dokumente/public/DOC-2005-Feb-447 e.html).
3. C. J. Date: Introduction to Database Systems, 8th edition, 2008.
4. S. V. Tyapkin: *Preimuschestva i nedostatki obektno-jrientirovannix baz dannix.* Jurnal nauchnix publikacii aspirantov i doktorantov, 2007, N4.- ISSN 1991-3087. http://jurnal.org/articles/2007/inf3.html (in Russian).
5. Kratkii obzor vozmozhnostei PostgreSQL. http://postgresql.ru.net/docs/overview.html (in Russian).
6. Desyat' glavnix nedostatkov PostgreSQL. http://vb-net.ru/PostgreSQL_defect/index.htm (in Russian).
7. Obzor NoSQL sistem. http://habrahabr.ru/post/77909/ (in Russian).
8. V. Kuznetsov et al.: *Life in extra dimensions of database world or penetration of NoSQL in HEP community.* The Compact Muon Solenoid Experiment Conference Report. 2012, CMS CR 2012/052.
9. D. A. Oleinik, A. Sh. Petrosyan, R. N. Semenov, E. G. Nikonov, I. A. Filozova, V. V. Korenkov, E. A. Tikhonenko: *Development of the CMS data bases and interfaces for CMS experiment: Current status and future trends.* XXI International Symposium on Nuclear Electronics & Computing (NEC'2007), Varna, Bulgaria, p. 376–381; http://nec2011.jinr.ru/pdf/nec2007.pdf.
10. A. Vaniachine: *Scalable Database Access Technologies for ATLAS Distributed Computing.* Proceedings of the DPF-2009 Conference, Detroit, MI, July 27–31, 2009.
11. E. J. Gallas, D. Malon, R. J. Hawkings, S. Albrand, E. Torrence: *An integrated overview of metadata in ATLAS.* CHEP 2009 Conference, Prague, Czech Republic, March 23, 2009.
12. G. L. Miotto, I. Aleksandrov, A. Amorim, G. Avolio, et. al: *Configuration and control of the ATLAS trigger and data acquisition*. Nuclear Instruments and Methods in Physics Research Section A, Volume 623, Issue 1, p. 549–551., 11/2010.
13. J. Almeida, M. Dobson, A. Kazarov, G. L. Miotto, J. E. Sloper, I. Soloviev, R. Torres: *The ATLAS DAQ system online configurations database service challenge.* Journal of Physics: Conference Series, Volume 119, Issue 2, p. 022004 (2008).
14. A. V. Vaniachine and J. G. von der Schmitt: *Development, deployment and operations of ATLAS databases.* J. Phys. Conf. Ser., Volume 119, p. 072031 (2008).

Received on December 12, 2014.

Редактор *Е. И. Кравченко*

Подписано в печать 9.04.2015.
Формат 60 × 90/16. Бумага офсетная. Печать офсетная.
Усл. печ. л. 1,69. Уч.-изд. л. 2,29. Тираж 225 экз. Заказ № 58519.

Издательский отдел Объединенного института ядерных исследований
141980, г. Дубна, Московская обл., ул. Жолио-Кюри, 6.
E-mail: publish@jinr.ru
www.jinr.ru/publish/