EQUIPMENT DATABASE DATA MODEL*

M. Peryt

Faculty of Physics, Warsaw University of Technology
Koszykowa 75, 00-662 Warszawa, Poland

T. Traczyk

Faculty of Electronics and Information Technology
Warsaw University of Technology
Nowowiejska 15/19, 00-665 Warszawa, Poland

(Received July 23, 2018)

Equipment Database (EqDb) is an information system designed to support construction, assembly, and operation of complex equipment in scientific experiments, *e.g.* detectors in high-energy physics. This paper describes main features of EqDb data model.

DOI:10.5506/APhysPolBSupp.11.685

1. Introduction

Equipment Database (EqDb) [1] is originally intended to be used for construction, assembly and operation of MPD (Multi-Purpose Detector) of NICA [2] at JINR (Dubna, Russia). Thanks to EqDb generic, highly flexible data structure, the system can, however, be quite easily configured to support almost every type of scientific experiment.

As EqDb can store information on all devices used in the experiment, it can become a backbone of the slow control system, and can also be used as a calibration database for the experiment.

2. EqDb data model

The Equipment Database system is based on generic — metadata driven — structures. Though conceptually object-oriented, EqDb is implemented using standard, proven and efficient Oracle relational technology.

Each entity represented in EqDb is treated as an Object. Several Object subclasses, called Object Categories, are predefined, as shown in Fig. 1.

* Presented at the II NICA Days 2017 Conference associated with the II Slow Control Warsaw 2017, Warsaw, Poland, November 6–10, 2017.

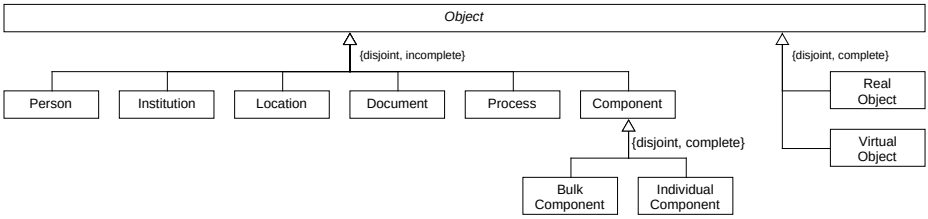

Fig. 1. EqDb objects and categories.

The most important category is Component, which represents equipment parts and more complex components, as well as complete devices. It has two disjoint subclasses. Individual Component represents components, which should be individually identified, *e.g.* some devices. Bulk Component represents components, which are indistinguishable, and we need only information on their sort/brand and quantity, *e.g.* screws or nuts.

EqDb objects are also divided into two other classes. Real Object represents objects which exist in real world. Virtual Object is an abstract representation of some object type, which can be used to represent any object of the type. It can also store common values of object properties, which are the same for all objects of the type. Only one virtual object can represent a given object type.

Objects are described using their properties and associations. Properties are named values which characterize the given object, *e.g.* names, dates, descriptions, quantities, *etc.* Associations are relationships between objects, which also can have some own properties — association attributes.

Some properties are common to all objects, *i.a.* dates when created/changed/canceled, textual description and user notes. Some other properties are common for objects of a given category, *e.g.* first, middle (patronymic) and last name for Person. These common properties are stored in non-generic part of EqDb data structures.

Other properties and associations, which are appropriate only for some kinds of objects, are generic, *i.e.* definable. Their definitions are stored in EqDb metadata and can be created or changed by data administrator. These properties, just like associations, are defined as generic features of a certain generic object type.

Each EqDb object belongs to exactly one category, but it can also belong to several generic, *i.e.* definable object types, which are defined to contain definitions of related object features (see Fig. 2) and/or to group objects that have the same features or other characteristics.

Figure 3 presents an example of type inheritance. Object Type class is a powertype, which instances define generic object types: Component, Cabinet, Container, *etc.* Type CME-2_42U is a component type, *i.e.* a type

that in EqDb represents real make/model of some device/component. It can have some own features, and inherits features from Cabinet type. This type inherits from two types: Component generic type, and Container type, which defines features necessary to express that a given component/device is a container than can contain other components.

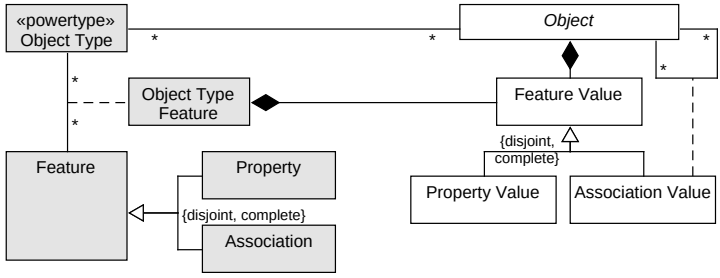

Fig. 2. EqDb object types and features.

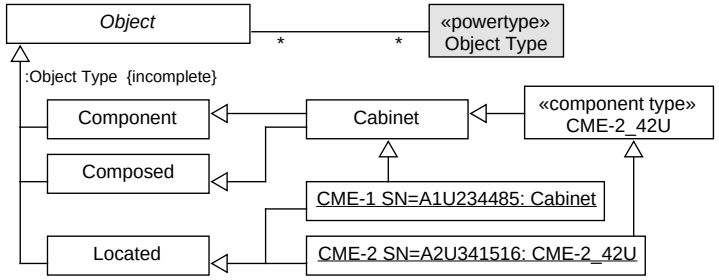

Fig. 3. Inheritance in EqDb.

EqDb data model differs from typical object-oriented models not only in that multiple type inheritance is allowed, but also in that an object instance can inherit features directly from more than one type, as shown in Fig. 3. Object instances inherit not only from their main parent types Cabinet and CME-2_42U, respectively, but also from Located type, which defines an association to some Location, and some additional properties to refine details of the object location.

Figure 3 shows also one more important possibility: an object instance, *e.g.* a component, can be a child of a type at various levels of detail in inheritance hierarchy. One of presented components is related to the real CME-2_42U component type, which represents specific device model. The other component is related to the generic Cabinet type, which represents any cabinet. Therefore, it is possible to register a necessary component although its exact type/model is not registered in the database, using more general

object type, and to refine the data when needed, by registering the device model as a new component type, and adding it as a new parent type for the component.

In Fig. 4, an exemplary use of a virtual object is presented. When an invoice is registered in EqDb, the exact type of a purchased component is specified, but a serial number of the component is not already known. Therefore, no specific real component of the given type can be used at this stage. Fortunately, in EqDb, a virtual component that represents any object of a given type can be created, and such a virtual component can be associated to the invoice. Next, when a delivery note is registered, the device is already physically delivered, so its serial number and other necessary individual properties are known. Therefore, a real component can be created and associated to the delivery note document.

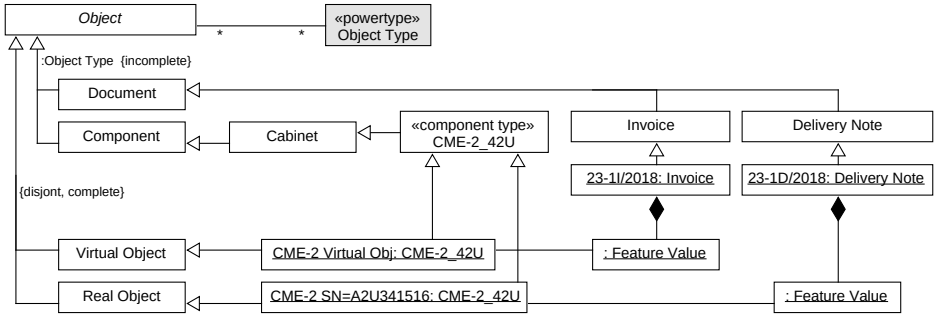

Fig. 4. Virtual and real objects.

3. Conclusion

EqDb Data model is a deliberate combination of a typical rigid data structure with a flexible generic structure driven by metadata. Common basic properties of objects are stored in the non-generic part of the structure, whilst the generic part can be used to store user-definable features, *i.e.* properties and associations of the objects, thus providing system flexibility. Inventive data structure ensures that it is possible to conveniently represent very diverse data in various circumstances, and at varying degrees of accuracy.

Thanks to its highly flexible generic data model, the EqDb system can be adapted to the needs of almost any complex experiment.

REFERENCES

[1] M. Peryt, T. Traczyk, *Acta Phys. Pol. B Proc. Suppl.* **9**, 293 (2016).
[2] V. Kekelidze *et al.*, *EPJ Web Conf.* **71**, 00127 (2014).