A DATABASE PROTOTYPE FOR MANAGING COMPUTER SYSTEMS CONFIGURATIONS

Zh. Toteva, Sofia University/CERN, CH-1211 Geneva 23

N. Sinanis, CERN, CH-1211 Geneva 23

{ Zhechka.Toteva | Nick.Sinanis } @cern.ch

Abstract

We describe a database solution in a web application to centrally manage the configuration information of computer systems. It extends the modular cluster management tool Quattor with a user friendly web interface. System configurations managed by Quattor are described with the aid of PAN[1], a declarative language with a command line and a compiler interface. Using a relational schema, we are able to build a database for efficient data storage and configuration data processing. The relational schema ensures the consistency of the described model while the standard database interface ensures the fast retrieval of configuration information and statistic data.

The web interface simplifies the typical administration and routine operations tasks, e.g. definition of new types, configuration comparisons and updates etc. We present a prototype built on the above ideas and used to manage a cluster of developer workstations and specialized services in CMS.

INTRODUCTION

The automated installation and configuration tasks have been an ongoing research matter in order to facilitate the fabric management task. The trend in the fabric configuration description task has turned to be a structured declarative description that offers a high-level of data abstraction [2]. Most of the tools holding on to this concept – EDG [3], SmartFrog [4], LCFG [5] - use declarative languages for describing the configuration information. The configuration information is centrally stored and managed, while its deployment is dynamically handled by procedural logic in the client subsystems. The modular tool Quattor [1] addresses the automated installation and configuration task according to the mentioned concept. Quattor is a part of the WP4 layer of the EUDataGrid project, coordinated by CERN.

Quattor uses a predefined tree-like structure for storing the computer system configuration information. This structure and the initialized configuration information are written in files in the declarative ad hoc language PAN. They are stored and processed in the Configuration Database (CDB) [1]. To reduce the redundancy of common data and to ensure the configuration coherence there are created reusable groups of configuration information. These configuration groups could be transparently inherited and overloaded.

The problem

Although PAN is convenient for describing flexible structures, it decreases the portability of the stored configuration information. A high level of abstraction is achieved only in the configuration information declaration, but not in its managing. The configuration data could be managed only on a file level with the available CLI. The file-level configuration processing appears to be inappropriate in the presence of a complex schema of configurations' hierarchies. Moreover, at the moment the CDB does not offer the possibility of making comparative analysis of the stored configuration data. With the increase of the computer fabric configuration scale, the fast retrieving of varied statistical information for computers' configurations becomes an important necessity.

The goal

The goal of the paper is to propose a solution for storing and managing fabric computer systems' configurations for Quattor with a relational database back-end and a web-client user interface. The logical architecture of the prototype is based on an *Object-Instance Model* (OIM). It is used for the description and the initialization of the semi-defined structure of the computer configuration information. The OIM is physically implemented in a relational data model that guarantees the schema consistency and the structure extension. We try to meet the requirements of Quattor for building groups of configuration information which could be easily reused and transparently inherited. The SQL standard relational database interface offers data portability and a high level of data abstraction in the configuration information management. The flexible web interface presents the concrete and the inherited configuration information and a convenient navigation between schema-connected configuration data. The client interface also offers application wizards that guarantee the coherence of the entered configuration information with the configuration information structure.

OBJECT-INSTANCE MODEL

The computer system configuration information is used by the client deployment subsystems of Quattor. Aiming to replace the CDB and PAN, we are obliged to follow the configuration information structure used by Quattor, called "Global Schema" [5].

"Global Schema"

According to the "Global Schema", the computer configuration information is divided into named units that consist of logically connected computer configuration features (Figure 1 - "software", "hardware", "system"). The features of these units are divided into named subunits according to a stronger logical inter-dependence. The process of the dividing continues according to the desired configuration information structure. The logical structure chosen for describing the configuration information is a property tree, built by tracing the names of the logical units and features. The leaves nodes of the tree are the features, which should be initialized.

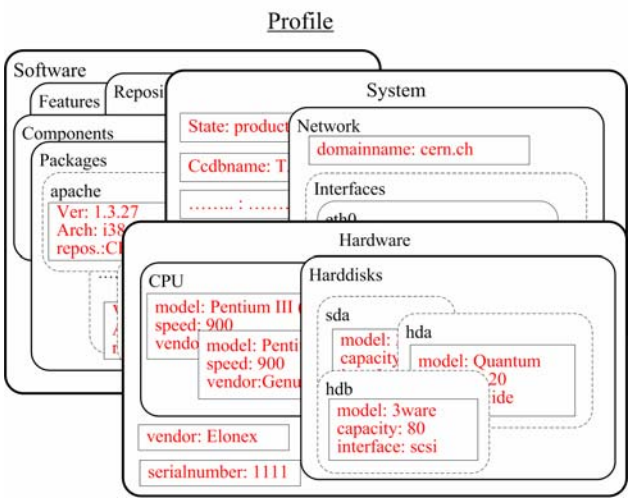

Figure 1: Computer configuration

Some of the configuration features describe common-type logical units ("hda", "sda", "hdb"). The different computers' configurations have a different number of these logically units. The correct description of the configuration structure demands the common-type logical units to be defined as vector elements. The common-type logical units are distinguished in the computer configuration initialization by the vector indexes.

Logical presentation

The prototype uses an object-instance mode (OIM) for the logical presentation of the computer configuration information. The OIM aims to differentiate the declaration of the configuration information structure and its initialization. The object, further called "*Configuration Tree Declaration*" (CTD), presents the construction of the property tree. A computer system configuration initialization is assumed to be an instance of this object. The object will be referred as "*Configuration Tree Initialization*" (CTI).

The CTD is presented as a set, which elements are the CTD nodes. The basic information that should be stored for a CTD node (Figure 2) is the node unique identifier, the node name, the node parent, the domain of the valid values for the node (only for leaf nodes) and a boolean flag stating if the node is a vector. The boolean flag shows whether the node corresponds to a vector element.

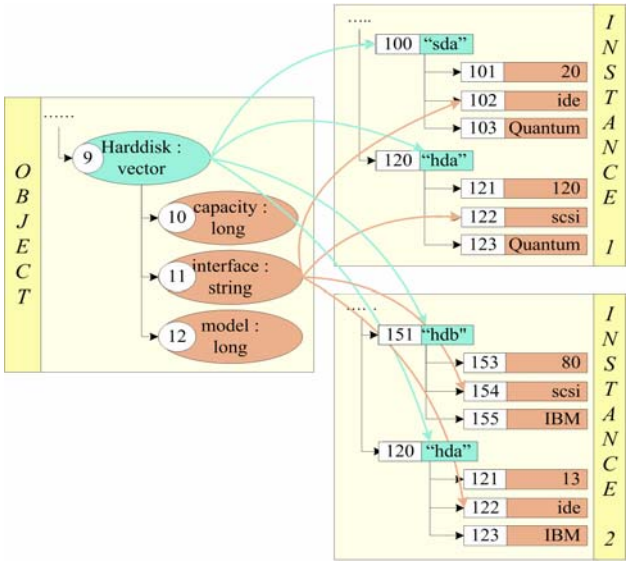

Figure 2: Object-instance model

The CTI is also presented as a set, which elements are the CTI nodes. Unfortunately, the CTD states only the semantics of the CTI, but not its full structure. That fact gives us the reason to call the configuration information structure "semi-defined". The paper addresses the introduction of a unified structure for saving the initialized computers systems configurations. The goal is to create the CTI structure in a way that:

- the same configuration features for each two different common-type logical units could be distinguished in a computer configuration initialization (Figure 2 - "Instance 1" - 101 and 121)
- the same configuration features for each two equal common-type logical units in different computer configurations are identified with the same identifier (Figure 2 - "Instance 1" and "Instance 2" - 121)

The necessary information which should be stored for a CTI node is the node unique identifier, the node parent, the node value, the reference to the CTD node, which the CTI node initializes and the index of the node (only if the corresponding CTD node is a vector).

The unique identifier of the CTI node is the supporting point for solving the inheritance of the reusable configuration groups and for comparing of the computer system configurations.

COMPUTER CONFIGURATION STRUCTURE

The configuration deployment process depends on the computer configuration information structure. Different configuration information structures results in different levels of automatic configuration management.

The prototype offers the possibility to design different CTDs and to support their initialization. In fact, the CTD is assembled from predefined *"Configuration Structure Components"* (CSC). Each CSC contains the declaration of logical connected configuration features ("cpu", "spma"). The CSCs are assembled according to the rule: a CSC could be included by any other CSC or by a CTD. This modeling offers the advantages:

- A separate processing of the development and the production computer configuration structure.
- Different levels of an automatic management of the computer configurations.

The physical implementation of CTDs in terms of the entity-relationship model is expressed by a recursive relationship (Figure 3). A CTD node is an instance of this entity. There is a key attribute that uniquely identifies each instance of the entity. The recursive relationship states the parent node for a CTD node. Descriptive attributes states the name, the data domain of a CTD node and whether it is a vector element.

The CSCs are described in the same tree structure as the CTDs, but they are stored in a different entity. The two entities have common attributes. The same storage of a CSC and a CTD causes same methods for their processing. Because of this functional similarity, we will refer the CSCs and the CTDs as *"Configuration units"* (CU), while exposing the execution of the basic operation.

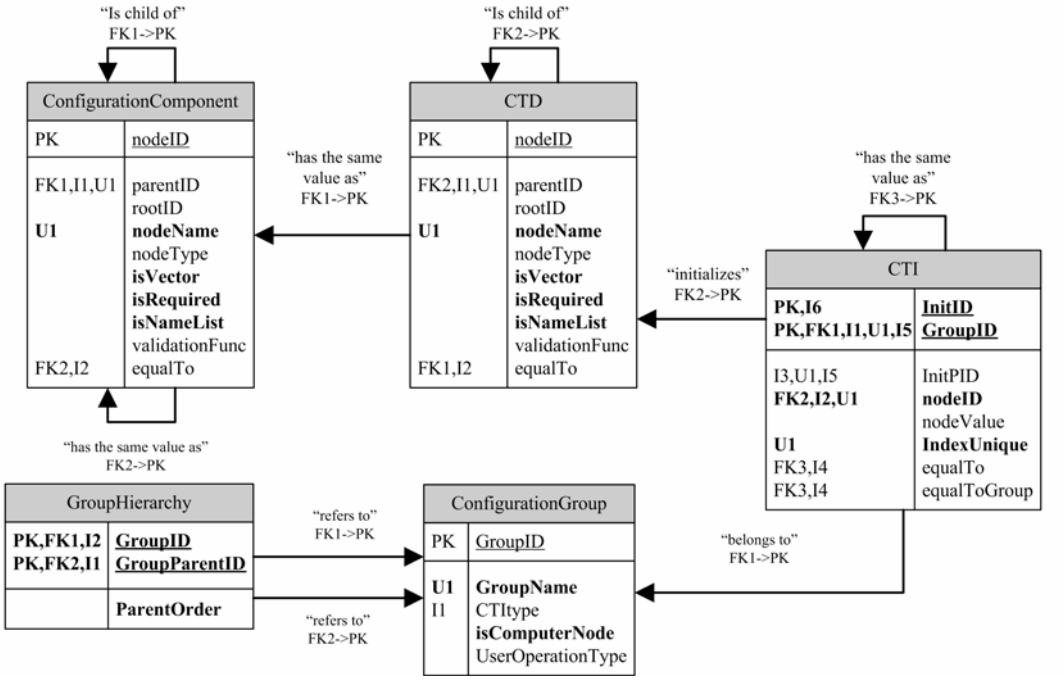

Figure 3: Relational database schema

Operations with a configuration unit

The CU is processed on a node level using the standard SQL interface and procedural logic. The operations which could be executed with the certain CU node are:

- modification
- deletion
- addition of a new node under it
- including of another CU under it

The including of a CU under the node physically results in copying of the to-be-included CU data with recalculated unique and parent identifiers. A relationship states the dependence of a duplicate node on the original one. Procedural logic transparently transfers an operation executed with the original node to the duplicate one.

The web client user interface for managing the computer configurations presents a CU in a directory-like web component. The navigation between the CU nodes resembles a directory browsing. The operations executed with a CU are processed regarding the current chosen CU node. The web controls restricted domains decrease the possibility of random errors and guarantee the consistency of the entered data. The client interface introduces extra functionality for the addition of a CU into another one. The available to-be-included CUs are listed in a dynamically updated list control.

COMPUTER CONFIGURATION INITIALIZATION

A great redundancy of data would have been generated if the configuration data of all the computers have been stored directly. Moreover, some of the configuration features of computers that serve similar tasks are the same. Configuration groups of initialized configuration information are designed. They could be included into each other and inherited according to definite schema rules. Regarding their applicability, the configuration groups could be separated into three categories:

- *Specialized groups* initialize logical connected configuration features (ex. cpu "Intel Pentium 4" at 2.4 GHr).
- *Ordinary groups* initialize random features. They could use a *specialized group* for initialing a given sub-tree of the CTI. Also, they could be inherited by and could inherit other ordinary groups.
- *Computer Configurations* also initialize random features. They differ from the *ordinary groups* only by their extreme (could not be inherited).

A configuration group is presented by a single CTI. A CTI is characterized by 1) the CTD, that the group initializes and 2) the CTD node, which the CTI root node initializes. We will refer the last characteristic as *CTI type*.

The computer configurations and the ordinary groups initialize the whole CTD. Their *CTI type* is "profile". The specialized groups could be of different *CTI type*: "cpu", "hard disk", etc. The CTIs of the same *CTI type* are comparable due to the unique CTI node identifiers. An important preliminary step before the initialization of CTI features is the choice of its characteristics. This choice defines the further applicability of the CTI.

In the terms of the entity-relationship model the CTI is expressed by recursive relationship. The entity is used to describe all the CTIs. An instance of the entity is a CTI node for a CTI. The recursive relationship states the parent node for a CTI node in the certain CTI. A relationship to the "CTD" entity (FK2) shows which CTD node a CTI node initializes. The CTI node identifier is generated by demand. The initialization of a new CTI node for a certain CTI is preceded by a check whether this node has been already initialized by some other CTI. In this case, the identifier is taken from the existing instance of the "CTI" entity. Otherwise, a new identifier is created.

Operations with a CTI

The CTI is processed using the standard SQL interface and procedural logic on two levels:

1. manipulation of the CTI characteristics
2. manipulation of a CTI node

The first level of processing includes the choice of the two mentioned characteristics and the creation of configuration groups' hierarchy. A configuration groups' hierarchy could be created from ordinary groups and computer configurations, which initializes the same CTD. The multi-inheritance is solved by introducing a parents' order, according which the CTI features of the parents groups are overridden (Figure 3 -"GroupHierarchy").

The operations which could be executed with a certain CTI node are:

- modification
- deletion
- addition of a new node under it
- including of a specialized group of a specific *CTI type* under it

The including of a reusable group under the CTI node is processed in the similar way as the including of a CU into another CU. Procedural logic transparently spreads the modifications made in the *specialized group* to the groups which use it.

The operations with CTIs are executed in a user session. The commit (rollback) of the user operations is up to the user decision. The real changes are executed when the user commits the session. Procedural logic resolves the configuration data modifications and the configuration groups hierarchy changes. The affected computers configuration information is recalculated. It is written in files that satisfy the format expected by the Quattor deployment systems. UDP notification packets are sent to the concerned computer systems.

Web user interface

The configuration information is initialized and managed via the web user interface per a CTI (Figure 4). The operations executed with a CTI are processed regarding the current chosen CTI node. The user could review all the initialized children nodes of this CTI node in two grid controls. The first one displays the initialized children nodes as a result of the solved inheritance. The second one reveals only the children nodes initialized for the given CTI taking into account the user modifications.

The user interface offers a wizard for an addition and a modification of a node. The CTI nodes that could exist under a certain node are listed in a dynamically updated list control. The web interface also offers a list of all the initializations of the chosen CTI node. A dynamically generated list box displays the *specialized groups* which could be included under the current node depending on their *CTI type*.

The parents of ordinary groups and computer configurations could be chosen from a dynamically generated list control. In a second list control the user could reorder the chosen parents' configuration groups.

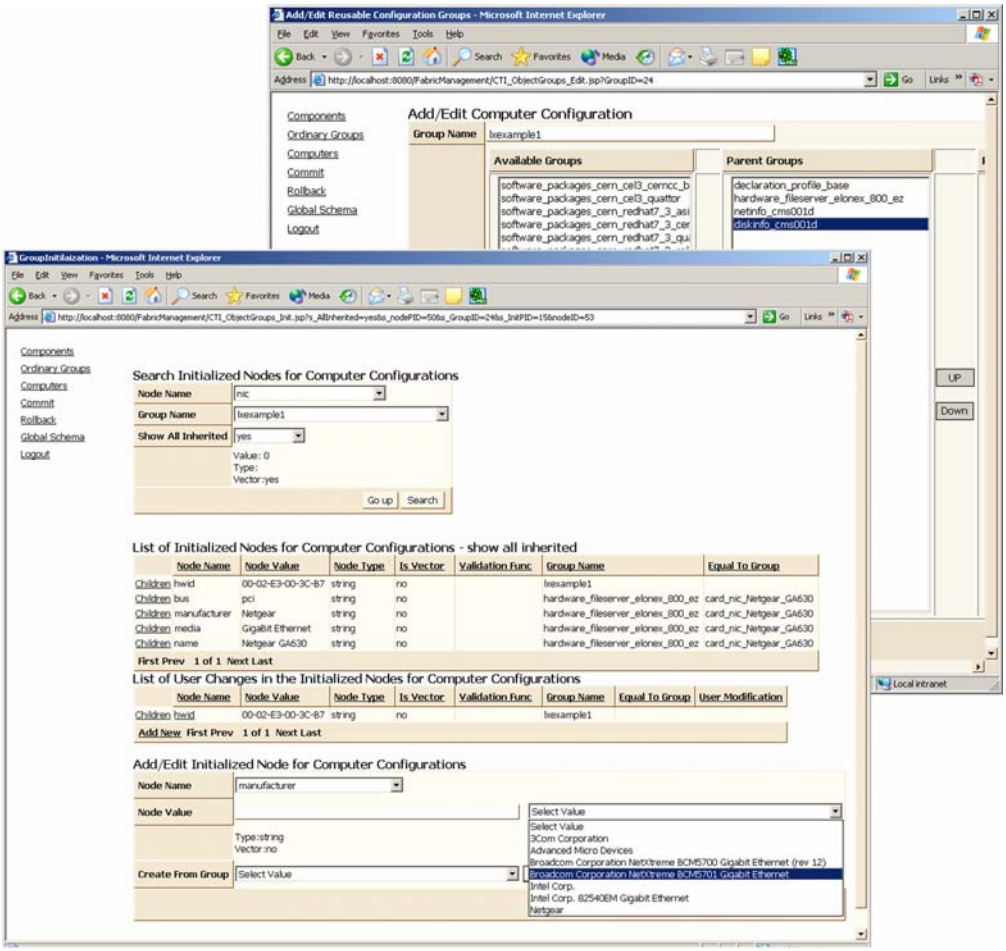

Figure 4: Computer information initialization – user interface

STATISTICS RETRIEVAL

The prototype offers statistical information for the initialized computer configurations information. The RDBMS ensures the fast retrieval of the desired information only by the SQL interface means. At the moment the web client interface gives the following statistical information:

- computer configurations which have a certain feature satisfying a certain criteria
- comparison of two computer configurations
- computer configurations which are using a certain predefined component

The result configuration data information is presented in grid controls. Links offer a flexible navigation between different schema-connected configuration data.

CONCLUSIONS

The prototype addresses the problems of the presentation of structured data with a semi-defined structure in the relational database model and the transparent managing of a high-level data abstraction. The solution to the first one is based on a relational database. The second one has been solved with the aid of SQL and procedural logic. The project has delivered a prototype proving the key concepts of the inheritance validation. The override processing has been already investigated and proved. The prototype is planned to manage the varied configurations information of nearly 300 computers of the CMS cluster at CERN.

As future work it is well worth making an optimization of the database operations. The tuning and the multi-thread processing of the computer configurations information recalculation would result in the minimum response and manipulation time. Another possible task is to improve the web-application wizards and to implement new graphical controls that will increase the data abstraction in the user-interface.

References

[1] http://quattor.web.cern.ch/quattor

[2] "Three languages for fabric configuration", 2003

[3] http://eu-datagrid.web.cern.ch/eu-datagrid/

[4] http://www.hpl.hp.com/research/smartfrog/

[5] http://www.lcfg.org