DITY™ Weekly Newsletter

itSM Solutions® IT Experience. Practical Solutions

The workable, practical guide to Do IT Yourself Vol. 6.17 • April 30, 2010

Enterprise CMDB

By Hank Marquis
Hank is EVP of Knowledge Management at Universal Solutions Group, and Founder and Director of NABSM.ORG. Contact Hank by email at hank.marquis@usgct.com. View Hank's blog at www.hankmarquis.info.

A CMDB is a nebulous thing, and can exist in the minds of the organization, 3x5 cards, or any other medium. However, an enterprise CMDB is different. For large organizations only software will do, but not traditional relational database software. What we need is a new breed of software products...and they might be coming...

I usually tell smaller and less mature IT organizations that they do not need to invest in expensive and complicated CMDB software. I have used Configuration Management principles of knowledge management to help different IT departments, using basic office tools like Excel.

A CMDB is a system for managing knowledge more than it is a product, and it is more of an index than a database. This means as long as the amount of data is small, you can get the benefits of a CMDB any number of ways—paper, Visio diagrams, Access databases, people's heads, etc. And this is exactly what you see in most organizations.

However, an enterprise IT organization spans large geographies with hundreds or thousands of users and locations, and the sheer number of configuration variations requires a specialized CMDB solution. For the record, I do not think such a product is available yet. The requirements lay just outside of today's technology and protocols.

But this is changing. First to change was the realization that a CMDB is not simply an IT Asset Management (ITAM) system "on steroids," but rather that a true enterprise CMDB requires some unique features. Chief among this is that a true enterprise CMDB solution has to dust off an old database concept that is a bit out of style—dimensional modeling. An enterprise CMDB solution has to be based on a dimensional database; a relational database simply cannot do the job.

Following I explain what a dimensional database is, and what an enterprise-class CMDB software solution requires: Federation, Reconciliation, Synchronization, and Modeling.

Many vendors are starting to talk this talk, but few understand what it really means, and fewer still deliver products that walk the talk.

Relational vs. Dimensional

Data stored in a relational database is easy to lookup when you know in advance what you will want to see. The common "row and column" approach of relational databases is ideal for On Line Transaction Processing (OLTP).

But a CMDB does not store most of its data; it references data stored externally in other, perhaps relational, databases. And a CMDB is used to provide contextual awareness over non-obviously related bits of data. For example, a common inquiry posed to configuration management might be: "How many users, in sales, use SAP during the last week of the month?"

This kind of query is not well suited to a centralized relational database with pre-built SQL queries. There are just too many possible combinations of data. This type of query has to pull information from many systems, and the data it needs is probably not all nicely lined up in rows and columns ready to query. No, an enterprise CMDB has to be dimensional—a technology that represents data as different dimensions or plains.

The dimensions of a CMDB often include locale (e.g., city, state, floor, etc.), work group like sales, marketing and so on, IT service like SAP or Email, date ranges, and others. Instead of an Excel spreadsheet with rows and columns, think about a Rubik's cube and you begin to get the idea. The logic required is not new, it has been around for years in the form of On

Line Analytical Processing (OLAP).

Don't get too excited—simply having OLAP does not give you a CMDB for a couple of very special reasons. First, most CMDB data resides outside of the CMDB system. In order to pull this data from many sources requires federation—a new CMDB buzzword you are hearing about more and more.

By way of an example of federation and what it requires, let us consider an IT service for project management. Composing this service are human resource information residing in SAP, project management data in Microsoft Project Server, IT asset information in CA Unicenter, and networking hardware resource data discovered and stored in CiscoWorks.

Federation

The first major requirement for dimensional modeling is federation, or referencing data from several sources instead of replicating it. The CMDB is a meta-database, that is, it is a database that references other databases. The issue that drove federation the first time around was data validity. If you make a copy of something, then what is definitive? The original or the copy? And how to you know if the copy is the same as the original?

The issues around federation are how to connect to heterogeneous data sources, resolve which bits of data are definitive, and then create and store keys with unique data not found in any external data source but still required. For example, data not found in any of these systems might be the name of the IT service and which workers use it.

There also has to be a method to store awareness of the types of data to be found in each federated data source in order to process ad hoc queries.

As you can see, the idea of federation is easy to state as "connecting to multiple data sources", but having a CMDB system that can actually federate is a very tall order indeed. And federation is just one of four equally complicated CMDB technical requirements. Consider this: what if two data stores reference the same data? Which data store is definitive then, and more importantly, how would you determine which is definitive? This is the issue of data confrontation and reconciliation.

Reconciliation

Aside from the issues of simply connecting to heterogeneous and possibly competitive data sources, the big problem with federation for a CMDB is data confrontation. During creation and maintenance of the contextual information in the CMDB metadatabase, key bits of data transfer from federated data sources to the CMDB data store. Since it is common to have multiple applications and systems that overlap and monitor the same IT assets or store similar data, possible data inconsistencies and redundancies arise: this is data confrontation.

For example, in our sample project management service, perhaps CA and CiscoWorks both store hardware data. Unicenter may refer to a router by name, perhaps “CISCO01.” CiscoWorks may be aware of the same router, not by the name “CISCO01,” but rather by its IP address "128.10.0.1"— this is a real problem since there are not two routers but one. How does the CMDB system discover that "CISCO01" and "128.10.0.1" refer to the same single router? Further, how would the CMDB system know it is a router at all? This is the domain of reconciliation.

Reconciliation implies adjusting data derived from more than one source to eliminate duplicates and maintain consistency of data. Federation is useless with reconciliation.

Reconciliation however is not the end of the requirements for an enterprise CMDB. Adding even more complexity to a CMDB system is the need to handle any changes arising from successful reconciliation, and this leads to synchronization.

Synchronization

Most data stored in federated CMDB data stores changes—sometimes slowly, sometimes swiftly. For example, considering our example project management IT service, the name of the project manager (e.g., the "user") might change, or the type of router hardware might change from a Cisco 2504 to a Cisco 2801. Reconciliation has to be able to resolve these differences to maintain the integrity of the CMDB. But this is IT, not "simple" data warehousing. No Configuration Item (CI) represented in an ITIL-aligned CMDB should change without a Request for Change (RFC). Thus, a CMDB system that can successfully federate and reconcile data must also be able to alert when it detects unauthorized/unplanned changes. Failure to synchronize reconciled changes will quickly result in an out-of-control CMDB—a recipe for disaster.

This makes the CMDB system require an awareness of approved changes. Then, when the reconciliation engine detects and resolves a change in infrastructure or data, it has to compare this change to an expected list of approved changes and generate an alert if the change is unapproved (e.g., not planned). This alert brings CMDB data to the attention of its administrators, who need help visualizing, mapping, and displaying data—the next major technical requirement of modeling.

Modeling

Modeling is mapping and visualizing synthesized relationships that are IT service definitions and CI interconnections. Modeling is more than reporting or displaying lists of resource trees and forks. The CMDB has to be able to visibly display its data in ways that let humans use the information in impact assessments for Change Management, privilege determination at the Service Desk, troubleshooting by Incident or Problem Management, and dozens of other ad hoc inquiries from all over IT.

This requirement goes way beyond the simple "directory tree" listings so commonly found in most alleged CMDB products today. Federation, reconciliation, and synchronization are worthless if a user in IT cannot get a definitive, understandable answer to their complex questions quickly. This requires representing complex relationships between CIs graphically, on demand.

Summary

An enterprise CMDB is not a singular database, it is a complex system that has to federate other data stores, reconcile alternate views of the same data, detect unauthorized changes, synchronize approved changes with its own metadata store, and be able to dynamically represent configurations graphically on demand. This is no small task, and also the reason there are so few true CMDB solutions available today.

As you go forward with your own CMDB plans, keep the concepts of multidimensional modeling and the issues of federation, reconciliation, synchronization, and modeling in clear view. Dig into these core issues to understand them. If you are in the throes of purchasing a CMDB product, ask your vendors how they handle these issues. If you are building your own CMDB solution, ask your developers how they plan to accomplish these tasks. If you are working with a consultant ask them what they know about these issues.

In all cases, make sure you create processes to monitor and ensure that federation, reconciliation, synchronization, and modeling occur. Failure to manage these critical issues can quickly convert your CMDB project from an asset to a liability.

Forewarned is forearmed! Now you can at least "talk the talk as you walk the walk!"