Extensions to the SAM database schema

Summary of all the additional tables and elements

William Ollivier

william.ollivier@cern.ch

This document intends to describe the extensions that have been made to the SAM schema, in order to match a certain number of requirements that the experiments (or VOs) issued during the past year or so.

Table of Contents

1. Overview 1
2. Topology related requirements 1
3. Requirements about test criticality , Availability and Reliability metrics 3
4. Conclusion 5

1. Overview

Most of these requirements were initially issued by members of the CMS experiment during the design of the current CMS SAM test results viewing tool[1] . Later on, during the modification of this interface, more experiments have been involved and have formulated other requirements.

2. Topology related requirements

2.1. VO-specific site naming conventions

One of the requirements of CMS (and later other experiments) was the possibility to have specific names for the sites, which are different from what can be found in the SAM Database.

Example 1.

People from the CMS team at CERN, but also site administrators use the name "T0_CH_CERN" as an alternative to "CERN_PROD" (the official name as it can be found in BDII or GOCDB, and therefore in the SAM database). The same goes for all the other 'Tier' sites (by opposition to opportunistic sites) that provide services to CMS.

2.2. Filtering out sites which do not support a given VO

From an experiment point of view, not all the sites in BDII provide a full support for it. Some sites might only provide services to the experiment as an extra to services already provided for other VOs. In other cases, sites may provide the whole infrastructure for an experiment, but are not used by this experiment. This explains the need for the given experiment to have a way to filter out the sites that do not officially provide support for them.

2.3. Provide site grouping according to the VO-specific hierarchy and topology

Required by CMS, this is quite an important requirement in the sense that CMS doesn't consider the same Tier hierarchy as in the SAM database. This requirement is VO-specific, since each experiment may consider several sites differently.

Example 2.

"UKI-LT2-QMUL" and "UKI-LT2-RHUL" are both considered as Tier3s by CMS, but as Tier2s for ATLAS

This requirement takes also into account the ATLAS topology specificity: the cloud concept. This concept only matters to ATLAS, and the SAM database doesn't take it into account.

2.4. Extension provided

The solution that has been adopted consists in the creation of 4 tables. The first 2 tables (VOSITE and VOSITEMAP) provide the first 2 requirements (specific naming and site filtering), whereas the other 2 (TOPOLOGYGROUP and TOPOLOGYENTRY) provide a solution to the third requirement (VO-specific topology).

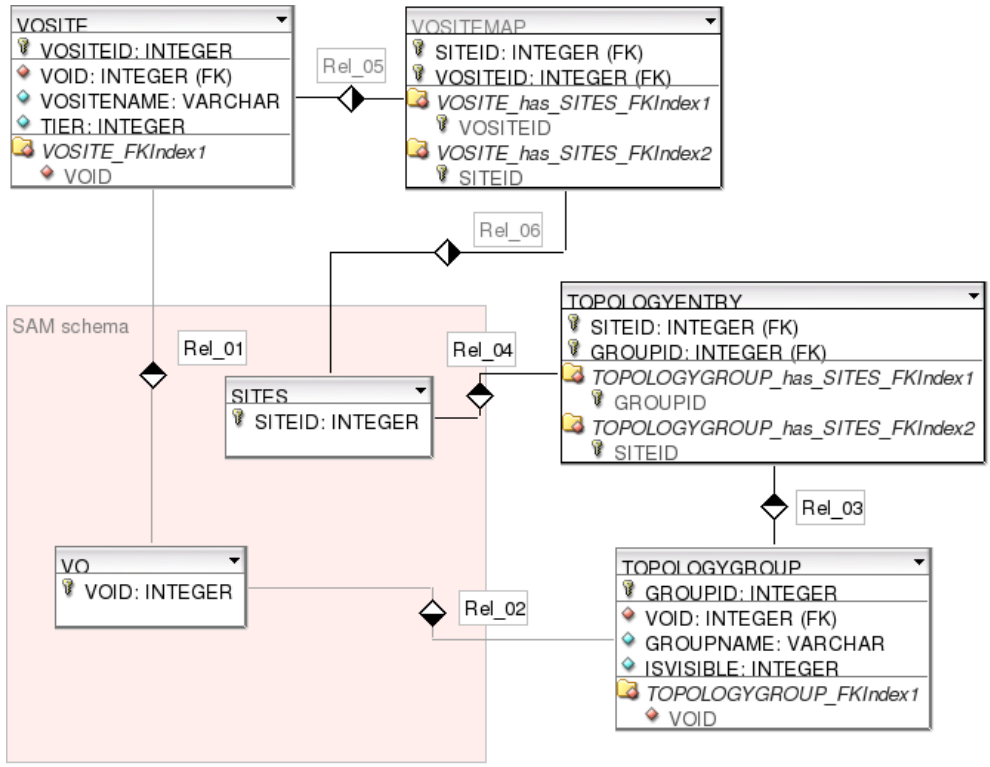

Figure 1. Extension offered for the topology related requirements

3. Requirements about test criticality , Availability and Reliability metrics

3.1. Test criticality

The current way an experiment can define which tests are critical is through the Freedom of Choice for Resources (FCR) portal. All tests that are defined as critical for an experiment via the FCR portal are used by GridView to compute Availability and Reliability metrics. These metrics are then used by the WLCG Management Board to discriminate whether a site behaves according to its contract. For this reason, defining a new test as critical via FCR is quite meaningful and can affect quite dramatically a site's results from the point of view of the Availability and Reliability metrics.

On the contrary, managers of the experiments might say that other tests are critical for a given experiment activity. And yet, they might not want to blame the site because it might not be its fault. A test might also be too generic, testing several functionalities at the same time and therefore often fails, but this doesn't necessarily mean that the tested service is down.

In another use case, an experiment might consider a test to be critical for its Tier1s, but not for the Tier2s or Tier3s. Therefore, to keep the metrics values within a reasonable range, they will avoid setting the considered test as critical, even though it is vital to the experiment.

Example 3.

Let's consider 2 tests defined for the same service. The first one tests a functionality provided by Tier1s only, whereas the second one tests the other by all sites. If the first is marked as critical, availabilities for Tier2s and Tier3s drop without reason.

3.2. Custom Availability and Reliability metrics

This requirement is a consequence of the previous one. Tests are critical for an experiment, in the sense that they guarantee a basic functionality (for example CEs and SEs are working correctly). The fact that these services are running however doesn't guarantee that a task, which might be a composition of these services, is also going to work.

This is why experiments need to be able to define several sets of critical tests, each of which is critical for a specific task, for example "Data re-processing" for ATLAS.

For each of these tests, one can calculate several metrics, in the way that GridView does, showing to the experiment when a site was available for a given task. This gives much more flexibility to the experiments on how they intend to use the resources that are made available to them; and provides a much finer granularity in viewing the capabilities of a site.

Example 4.

If you consider a site that has a CE and a SE. According to GridView, the site is not available as soon as either the CE or the SE is not working properly. However, if the CE only is down, the site is still able to handle data. Therefore, from a "Data Management" point of view, the site is running properly.

Another important point of this requirement was the definition of availability calculation algorithms. Currently, GridView computes the availability and reliability metrics using the same algorithm for all the VOs. However, experiments wanted to be able to have their own algorithm to calculate similar metrics. Providing experiments with the possibility to define their own algorithms is a complex and long task, which involves the use (or the creation) of an algorithm description language that could be translated into the Oracle PL/SQL language. Therefore, all the metrics would not be available to the experiments before months.

After discussions with the experiments, it became clear that this degree of flexibility was not as critical as having the metrics calculated as soon as possible. Therefore, efforts were concentrated on designing a simpler solution that would still suit the experiments' needs. The hybrid solution that was proposed offers a certain degree of customization, but not to the extent of having a completely different algorithm for each of the metrics calculated.

The main principle of the algorithm is the same as that of the algorithm GridView uses, but a number of parameters can be tuned. The algorithm is based on a metric called "Status", which characterizes the state of a site at a given date and time. Availability and Reliability metrics are based on the values of the Status metric gathered over a specified amount of time (one hour and one day). During this period of time, the site's status can change. And it happens that sometimes, one can't discriminate the state of a site, for a number of reasons. This indetermination is the point for which the experiments had different opinions.

Some experiments considered that, because one can't tell in which state is the site when it has an unknown status, the amount of time shouldn't be included in the

calculation of Reliability (this method is the one GridView has adopted). Other experiments consider that this amount of time should be included in the calculation, by considering that, even though there is no information to rely on, the site is still running, successfully or not, during this period.

Therefore, the algorithm has been designed so that it performs a slightly different computation, based on 2 parameters that the experiments can specify.

3.3. Solution to this series of requirements

These requirements have been dealt with by adding tables that allow each VO to define several sets of critical tests. These sets of critical tests are called "availabilities" from a database point of view because they define the parameters to calculate the metric. Figure 2 describes these new tables and their interactions with the existing SAM tables.

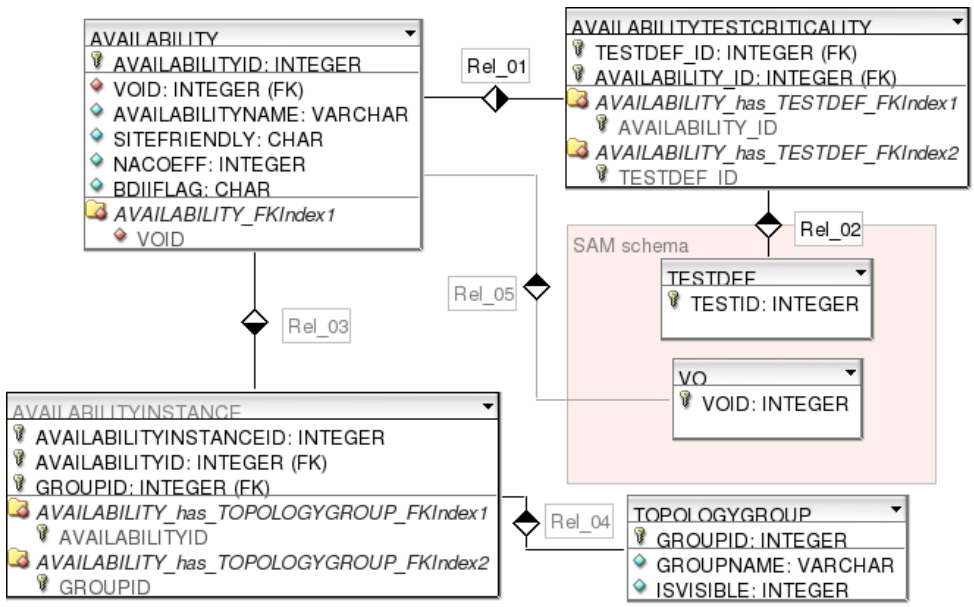

Figure 2. Tables to fulfill the criticality and availability requirements

The columns AVAILABILITY.SITEFRIENDLY and AVAILABILITY.NACOEFF hold the parameters that are used by the algorithm to meet the requirements of the experiments in terms of Reliability computation, as described at the end of Section 3.2. These can eventually be replaced by a column pointing to the description of an algorithm, if the decision of having a different algorithm per Availability metric is made.

4. Conclusion

The extensions to the SAM database that have been described in this document were born from the experiment's need for flexibility, at 3 different levels:

- Flexibility regarding the sites, and the topology: different naming conventions, different topologies, different site grouping mechanisms;

- Flexibility regarding the criticality of tests: tests are not critical for all sites at the same time, need to detach from the metrics that GridView computes and provides;
- Flexibility regarding the criticality of services types: some services are only run in a limited number of sites, and this shouldn't impact the results of other sites. This is achieved through the solution provided for the second point.

Notes

1. http://dashb-cms-sam.cern.ch/dashboard/request.py/samvisualization