



Data Sheet: Mulgara



INTRODUCTION

Mulgara is a proven data storage solution that utilizes the framework of the Semantic Web to collect and connect information. By using this cutting edge technology, Aperio is able to provide a powerful and flexible method to manage and control test data.

Aperio's use of Mulgara provides a fast, robust and flexible method of managing huge amounts of data. Mulgara is a relational database – it is data about data

KEY FEATURES

RDF Data Store

Data is stored as triples.

Simple groupings of statements about relationships between pieces of data.

Information is organized using an AVL tree.

A World Wide Web Consortium (W3C) standard

64-Bit Architecture

More information can be retained.

Data can be scaled to larger sizes.

Data is processed faster.

Integrity

One writer, multiple readers.

Data store is not corrupted.

Java NIO Libraries

Provides high speed disk read/write operations.

OVERVIEW OF KEY FEATURES

RDF Datastore

Mulgara is an RDF Metastore; it is a repository for Metadata (data about data) that is stored in the Resource Description Format (RDF), a W3C standard. Unlike a SQL (Structured Query Language) database that stores information in tables, Mulgara is comprised of models which are filled with triples. All data is stored as triples, where a triple is simply a statement describing a relationship using a subject, a predicate, and an object. There is no need to create complicated database schema models which are error-prone and difficult to manage. The simple nature of being a group of statements about how pieces of data are related is what allows Aperio to connect your data together.



64-Bit Architecture

Mulgara is optimized for the use of 64-bit architectures, a future requirement for all Microsoft® servers. By having native support for 64-bit data structures, the system is able to take advantage of the benefits that 64-bit processing has over 32-bit processing. Mulgara has a larger memory space, and can scale to larger sizes and process data faster than other solutions.

Mulgara supports multiple readers, but only one writer through its use of the AVL tree's transactional support. When a read connection is opened, a "phase" of the datastore is created for that reader's session. This phase persists until the reader completes its session, releasing the phase. Since only a single writer is allowed, the phased nature of the datastore allows all readers to continue retrieving information from Mulgara, even when a single writer has locked the database. When the writer has completed making modifications to the data, all of the changes will be rolled into the current phase in one operation. With this tactic readers do not have to worry about the data changing while they are working, and the data is prevented from entering an inconsistent state. This is analogous to the transactions and table-locking of a traditional SQL database.

Mulgara's 64-bit architecture and robust transactional support also provides additional data integrity. Even a catastrophic failure such as a hardware failure of the machine running Mulgara or a power outage, will not corrupt the data store.

Integrity

Mulgara is designed to store huge amounts of data. This information is organized internally in the form of an AVL tree. Each piece of information is mapped to a 64-bit integer and added to the tree. Since an AVL tree is merely a modified form of a binary-search tree, read, write and delete operations against the tree occur in $O(\log(n))$ time. The benefit to choosing an AVL tree is that it is auto-balancing. As data is added to the system the tree will adjust itself so that all branches have the same depth. This prevents the tree from becoming lopsided, which would lead to longer query times and degraded performance.

JAVA NIO Libraries

Mulgara utilizes the Java NIO libraries in order to increase the run time of operations. Since Mulgara deals with data sets that can be very large, it needs to be able to perform disk read/write operations very quickly. The Java NIO package provides the mechanism through which Mulgara can approach the speed of a program coded in C, when performing disk operations.

REFERENCES

<http://www.nist.gov/dads/HTML/avltree.html>

<http://www.w3.org/RDF/>

