

AllegroGraph

AllegroGraph 64-bit RDFStore™

Capable of processing billions of RDF triples, AllegroGraph is a modern, high-performance, persistent, disk-based RDF graph database with support for SPARQL, RDFS++, and Prolog reasoning from Java applications.

AllegroGraph New V2.3 Features

- **Performance!** We are constantly working on the performance of AllegroGraph, and this release is again faster
- Freetext indexing: AllegroGraph can now index text in literals and retrieve triples based on free text
- SPARQL with the RDFS++ Reasoner
- RDFS++ now supports owl:hasValue restriction reasoning
- Prolog with a better environment interface and more elegant "direct reification"
- A remote query interface for SPARQL with the standard SPARQL Protocol Client
- Java API Improvements
- A new Lisp client API that is modeled after the Java API
- The RDF/XML parser is vastly improved
- A TRIX serializer and parser
- An update app for Java Edition users to get the latest updates
- Easier to use documentation

High-performance Storage and Querying

Allegrograph is designed for maximum loading speed and query speed. Loading of triples, through its highly optimized RDF/XML and N-Triples parsers, is best-of-breed, particularly with large files. Using standard x86 64-bit hardware, it can load gigabytes of RDF data in minutes. The following table displays AllegroGraph's performance in loading and indexing a variety of commonly used benchmark RDF files and Ontologies.

AllegroGraph Load Test	# Triples	Size *	Time	Load Rate **
OpenCyc	221 K	56.5 MB	16.5 Seconds	13.4 KTPS
WordNet 2.0	1.81 M	464.5 MB	2.33 Minutes	12.97 KTPS
LUBM50	6.88 M	1.6 GB	6.85 Minutes	16.74 KTPS
Wikipedia	47.1 M	11 GB	42.7 Minutes	18.38 KTPS
LUBM1000	138.3 M	33 GB	2.44 Hours	15.75 KTPS
Uniprot	234 M	28.6 GB	3.35 Hours	19.4 KTPS
LUBM8000	1,106.6 M	264.4 GB	23.5 Hours	13.1 KTPS

* Size = size of triple file on disk, **KTPS = Thousand Triples per second

The system used to generate the numbers was dual CPUs on an AMD Opteron™ (x86-64) 844 running at 1.8 GHz, with 128 KB of L1 and 1MB of L2 cache, and 16 GB of ECC RAM, running Linux 2.6.18.

Querying is both flexible and performant. With multiple indices, it supports very fast access through a simple triple-level API, Allegro Prolog, or SPARQL (the emerging W3C standard RDF query language). When querying for a particular subject with ten triples, AllegroGraph can retrieve about 40,000 triples per second, from disk.

Accessing the AllegroGraph Triple-Store

There are several ways to work with AllegroGraph:

- **Java.** The Java client interface implements most of the Sesame and Jena interfaces for accessing AllegroGraph RDF repositories remotely. When working in Java you will always be in client/server mode. Multiple triple-stores can be accessed from one Java client. Multiple simultaneous Java and/or HTTP connections to the same triple store are supported. Each client has exclusive access during one operation:
 - In Java, an operation is defined by each of the API methods. Thus the array methods offer a primitive form of transaction.
 - In HTTP, each request is one operation.
- **Sesame HTTP client protocol.** It is possible for developers to interact with AllegroGraph using the Sesame 2.0 HTTP protocol to add and delete triples, to query for individual triples and to do SPARQL and Prolog selects. We extended the protocol so that it can do additional database management functions.
- **Python, Ruby, JavaScript, etc.** The HTTP interface can be used from any language that knows how to make HTTP client requests. This way, you can easily use AllegroGraph from Ruby, Python and many other languages.
- **Lisp.** In stand-alone mode, Lisp programmers can open up one or more triple-stores. You can create applications in the same image that the AllegroGraph server is running so you have very fine-grained control over the behavior of the triple-store. You can then use AllegroServe HTTP server to create client/server applications. You can also create your own protocol if required.
- **Lisp client.** The lisp client API allows an application to run in client/server mode. The Lisp client can access a server on the same host or in a separate location. In this mode multiple clients can access the same collection of triple stores and each client can access multiple triple stores. The same server can be used by Java and Lisp clients. The Lisp client API may be used with both local and remote triple stores.
- **TopBraid Composer.** This is a commercially supported tool for modeling and editing ontologies. You can connect TopBraid composer to AllegroGraph and visually inspect your RDF/OWL data. For details see <http://www.topbraidcomposer.com/>

Powerful and Expressive Reasoning and Querying

AllegroGraph provides the broadest array of mechanisms to query and access knowledge in RDF triples:

- **SPARQL Queries on Named Graphs**

SPARQL, the W3C standard RDF query language, returns RDF and XML in responses to queries. AllegroGraph's SPARQL includes a query optimizer, and has full support for named graphs. It can be used with the RDFS++ reasoning turned on (i.e., query over real and inferred triples). SPARQL can be used with every available AllegroGraph interface mentioned in the previous section.

- **RDFS++ Reasoning on Named Graphs**

Description logics or OWL-DL reasoners are good at handling complex ontologies. They tend to be complete (give all the possible answers to a query) but can be totally unpredictable with respect to execution time when the number of individuals increases beyond millions. AllegroGraph offers a less complete but very fast and practical RDFS++ reasoner. We support all the RDF and RDFS predicates and some OWL ones. The supported predicates are `rdf:type`, `rdfs:subClassOf`, `rdfs:range`, `rdfs:domain`, `rdfs:subpropertyof`, `owl:sameAs`, `owl:inverseOf`, `owl:TransitiveProperty`, and `owl:hasValue`.

- **Prolog**

Allegrograph's RDF Prolog provides concise, powerful, industry-standard, domain-specific reasoning to build high-level concepts (that require complex rules or numerical processing) on top of RDF data. Such is difficult (or very cumbersome) to model with only RDF/RDFS and OWL. Prolog can also be used on top of the RDFS++ reasoner as a rule based system.

- **RacerPro and RacerPorter**

The Semantic Web reasoning system, RacerPro, has been integrated with AllegroGraph, exposing RDF data in AllegroGraph to Racer's highly optimized Description Logic (DL) reasoner. It is most suitable for ontology-driven applications or theorem proofing. RacerPro's interfaces also include DIG over HTTP and support for rules (SWRL).

- **Low-level APIs** Allow fast, 'close-to-the-metal' access to triples by subject, predicate, and object.

Additional Features

- **Native Data Types and Efficient Range Queries**

AllegroGraph stores a wide range of data types directly in its triple representation. This allows for very efficient range queries and significant reduction in triple-store data size. With other triple-stores that only store strings, the only way to do a range query is to go through all the values for a particular predicate. This works only if everything fits in memory; but if the predicate has millions of triples, it will need costly machines with huge amounts of RAM. AllegroGraph also supports most XML Schema types (native numeric types, dates, times, longitudes, latitudes, durations and telephone numbers).

- **Named Graphs for Weights, Trust Factors, Provenance**

AllegroGraph, a triple-store, actually stores quintets. A triple in AllegroGraph contains 5 slots, the first three being subject (s), predicate (p), and object (o). The remaining two are a named-graph slot (g) and a unique id assigned by AllegroGraph. The id slot is used for internal administrative purposes, which can also be referred to by other triples directly.

The W3C proposal is to use the 'named-graph' slot for clustering triples. So for example, you load a file with triples into AllegroGraph and you use the filename as the named-graph. This way, if there are changes to the triple file, you just update those triples in the named graph that came from the original file. However, with AllegroGraph, you can also put other attributes such as weights, trust factors, times, latitudes, longitudes, etc, into the named graph slot.

- **Direct Reification**

AllegroGraph allows triple-ids to be the subject or object of another triple. This is beyond the scope of pure RDF. The advantage of this approach is that you can reduce the total number of triples in the store to a more manageable size, and, even more importantly, dramatically reduce query time because a single query can retrieve more data.

- **Geospatial and Temporal Reasoning**

AllegroGraph stores geospatial and temporal data types as native data structures. Combined with its indexing and range querying mechanisms, AllegroGraph lets you do geospatial and temporal reasoning as efficiently as native geospatial databases.

- **Clustering**

When loading a large set of data on a single processor system, roughly 60% of the time is spent in loading triples, 40% is spent in indexing. If you 'bulk load' your data on a multiple processor system or a cluster of independent machines, you can do nearly all indexing parallel to the loading process. And, while running interactively, it can index newly added triples in the background.

- **Distributed Triple Databases**

AllegroGraph supports queries with distributed databases. It allows thread-safe opening of multiple triple-databases from one application (for the read only parts of the database). Queries over multiple databases are easy with direct data access from applications. It also supports physical merging of databases.

System Requirements

Though best suited for 64-bit architectures, AllegroGraph runs on the 32-bit and 64-bit operating systems listed below.

32-Bit

Apple Mac OS X (Intel) 10.4

FreeBSD 6.x

Linux (x86), glibc 2.3

Microsoft Windows 2000/XP/Vista/Server 2003

64-Bit

Linux (x86-64), glibc 2.4

Microsoft Windows XP/Vista/Server 2003

Sun Solaris (x86-64) 2.10

AllegroGraph Professional Services

Social Network Analysis

AllegroGraph is an excellent foundation for Social Network Analysis (SNA) applications. Franz works with customers to develop algorithms for SNA applications using AllegroGraph. Some of the SNA algorithms are trivially implemented; examples are in-degree, out-degree, nodal-degree, density, actor-degree-centrality, and group-degree-centrality. Some of the algorithms are more complex because they require computing shortest paths between actors; examples are actor-closeness-centrality, group-closeness-centrality, actor betweenness-centrality, group-betweenness-centrality, connected-p, and find-clique-around.

Working with RDBMSs

Franz works with customers to migrate data in RDBMS or CVS files into AllegroGraph.



555 12th Street, Suite 1450

Oakland, CA 94607 USA

+1 (510) 452-2000 • FAX +1 (510) 452-0182

info@franz.com • www.franz.com