



sones Whitepaper

GraphDB



The sones GraphDB

The sones GraphDB is a new, object-oriented data management system based on graphs. It enables efficient saving, management and evaluation of extremely complex data bases that are networked to a high degree. The integrated solution combines the advantages of file storage with the possibilities of a Database Management System. Unstructured data (e.g. video files), semi-structured data (Meta data, e.g. log files) and structured data can be linked to each other and thus be managed from one source and be evaluated as needed.

The idea for a new development of a database based on the graph theory arose in the beginning of 2005 due to dissatisfaction with the existing solutions available in the market. Often, those offered no or only insufficient ad-hoc analysis functions for queries into strongly linked data. Besides, these systems were mostly not able to cope with the frequently changing data and information structures applications have to process especially in the web environment. Therefore, the development focused on providing efficient data handling and real time analysis for these tasks which could previously due to their complexity hardly be solved or only with the highest efforts. A fundamentally new database technology was developed based on totally independent and new concepts.


The solution includes an object-oriented storage solution (sones GraphFS) for high-performance data storage as well as the graph-oriented database management system (sones GraphDB), which links the data to each other and makes their high-performance evaluation possible. The summation of these two parts into one software offers the advantage of highly efficient processing since any abstraction layers can be avoided. The otherwise customary efforts for the operation of a database solution will be extremely decreased by this solution.

The object-oriented storage solution (sones GraphFS) also stores unstructured and semi-structured data, like e.g. document formats, photos, more efficiently than file systems, which are the basis for the operation system. This principle successfully overcomes one of the significant performance problems in the previous object-oriented

databases, which is the mapping of object models in relational tables. A patent has been filed for the file system.

The graph-oriented data management (sones GraphDB) offers high flexibility of the data structuring and enables the direct linking of the data without additional auxiliary constructs like, for example, allocation tables in relational data bases. Since the data model is oriented on the established programming languages, there is no discrepancy between application logic and data storage. The complexity is in part drastically smaller compared to the existing relational solutions. This leads amongst others to the following advantage: With increasing linkage of the data, the complexity does not increase exponentially, but in a linear manner. A further decisive advantage is the ability of the data base to change or amend data structures during runtime. New requirements on the part of the application can therefore be realized without extensive re-programming of a relational database schematic.

The Graph Query Language (sones GQL) was developed to facilitate the data management for the user. While its extent and syntax is oriented on SQL, it was adjusted to the graph-oriented approach and makes the advantages which exist in this model accessible to a direct and easily comprehensible query.

An abstract graphic consisting of several overlapping, organic, wavy shapes in a light green color against a white background. The shapes are irregular and fluid, resembling stylized waves or flowing liquid. They are positioned in the lower half of the page, partially overlapping the text area.

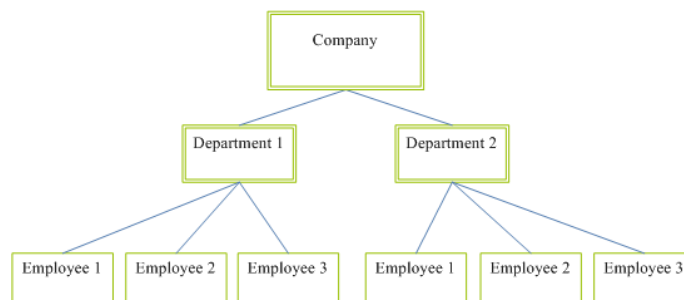
The market

The **Database Management System** is the most important instrument for electronic (mass) data management. It is its job to store data efficiently, unambiguous and durably and to provide it to the user on an as needed basis.

To this end, several concepts have been developed during the software development, which are fundamentally different from each other.

Hierarchical databases

The model of the hierarchical databases is the oldest. Data was stored in a tree structure starting from a “root“ data set. Dependencies between data sets are described as “parent-child“ relations in the hierarchy.



Hierarchical database systems have now been almost completely replaced. These are only still used in banks and insurances for older applications. A representative is IMS/DB by IBM.

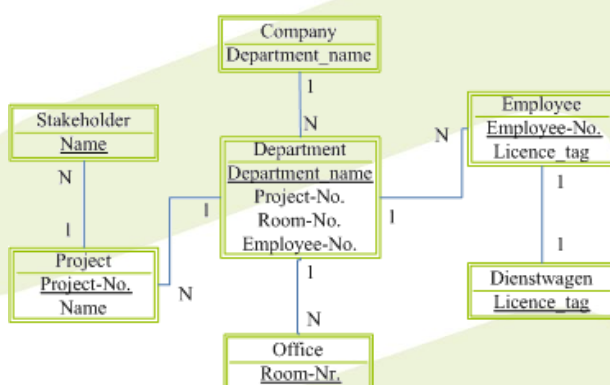
The concept of the hierarchical databases is partially re-used in document-oriented data bases, e.g. when information is stored in a hierarchical structure in XML files.

Relational databases

The relational database model is currently the most prevalent.

Data is stored in tables line by line. For each data set (line) any number of attributes can be stored (in columns). Dependencies between data sets can be described by way of common use of clear attributes

(keys). The structure of the data model will be determined at the time of compilation (or modification). In the past 20 years relational databases have achieved acceptance since



the requirements of structuring (in tables) and linking (by referencing the data sets to each other) can be well fulfilled with the model. Additionally, there is a standardized query language (SQL), which can be easily used for evaluations. The relational database model reaches its limitation when data are not directly dependent, but through several relations (For example, a multi-level hierarchy cannot be specifically evaluated with a SELECT Statement). Moreover, often additional artificial key attributes have to be applied to reduce the complexity of links and thereby ensure the required performance. This in turn increases the complexity of the model and requires additional storage space.

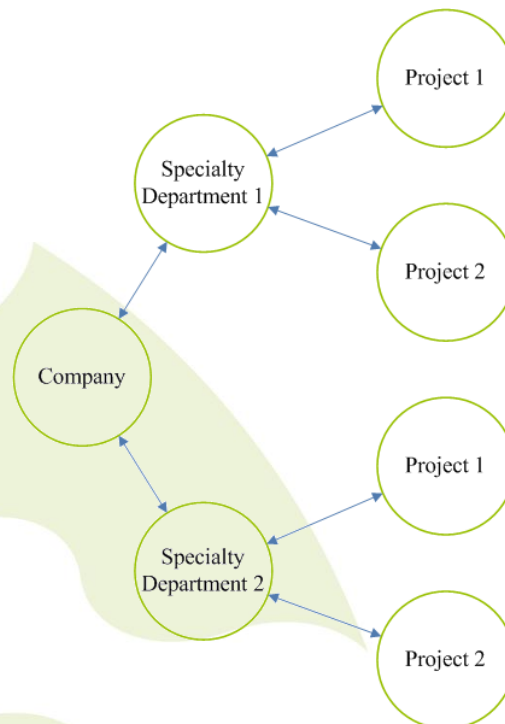
There are many standardized interfaces and a large number of useful tools for relational database systems.

The most well-known commercial databases are those by Oracle (as of Jan. 2010: version 11g), Microsoft SQL Server and IBM DB/2. The most well-known Open Source databases are MySQL and Postgres.

Object-oriented databases

Object-oriented databases are a “comparatively” new concept (Developed since the end of the 80s – together with the object-oriented programming languages).

The structure of the data model is determined at its compilation (or modification). All attributes that describe an object are summarized into one object class by the data modeling. Relations between objects are described through references (as in the object-oriented programming languages). The advantage compared to the relational data bases is that no auxiliary constructs (artificial key attributes, allocation tables) have to be planned for dependencies. This decreases the effort of the compilation as well as the complexity of the data base. Since they do not reach the performance of relational data base management systems, object-oriented data bases do not yet have a large distribution. Commercial solutions are for example Versant, InterSystems (Cachè) and Objectivity (Objectivity/DB).



Document-oriented databases

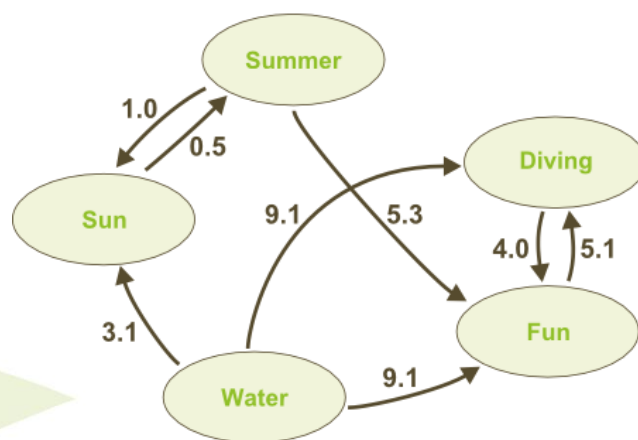
It is the goal of the document-oriented databases not to determine the data structures at the time of the conception of the database models. Data are stored in an unstructured manner and completely evaluated when needed. Providers of document-oriented data bases usually offer other solutions to ensure the performance of the evaluations for large amounts of data.

The advantage of this solution is the flexibility at the time of the data import, since no specifications are made to the data structure. This, however, increases the effort when evaluating the (so-called “semi-structured“) data.

Apache CouchDB, MongoDB or Amazons SimpleDB are examples for document-oriented data bases.

Graph-oriented databases

Graph-oriented data bases store nodes and relations between nodes. An indirect (complex) relation (e.g. between object 1 and 3) exist if an object references a second and this in turn a third.



Multilevel relations are called graphs.

Graph-oriented databases allow a query of indirect relations over multiple edges. The edges can also be weighed to describe the degree of the relation. Thus, complex and indirect dependencies in data bases can be easily described and their high-performance evaluation can be enabled.

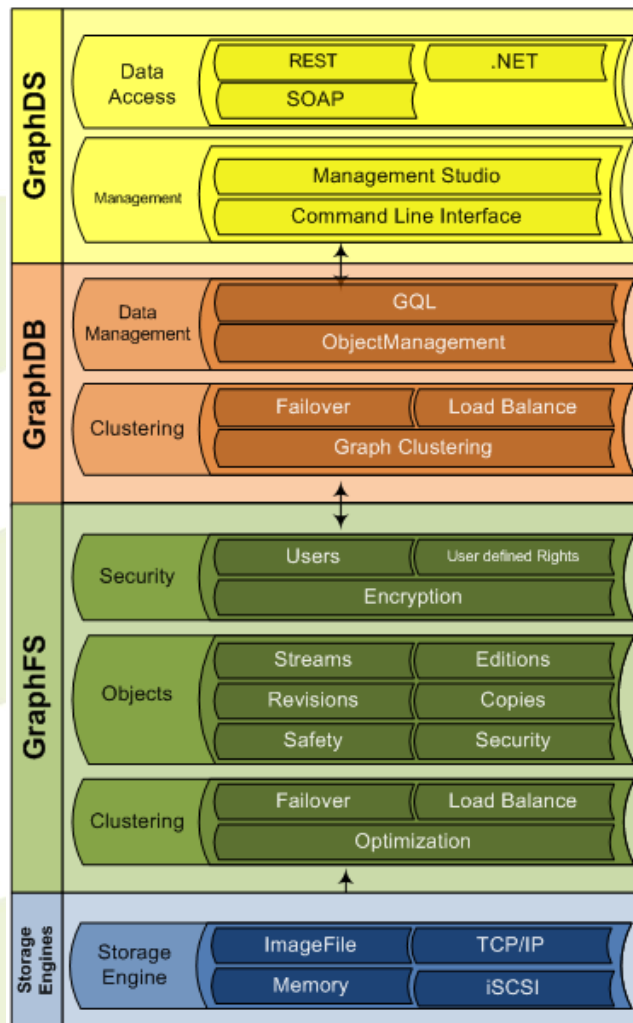
The evaluation of the data is not limited to the lists, but the data can also be processed structured as graphs. The advantage of this approach is that the work step of data structuring will be saved in the data processing application. The GraphDB of sones is a graph-oriented database.

The architecture

Contrary to most object-oriented approaches the sones GraphDB is a complete internal development. The approach of a complete re-development was in particular chosen to preclude potential legacy problems. The whole development is based on the approach to challenge established approaches and to re-think them if necessary.

The graphic illustrates the structure of the sones GraphDB. It consists of four application layers – the Storage Layer, the File System, the Database Management System as well as the Data Access Layer.

These as well as the available interfaces are explained in detail below.



Storage Engines

The Storage Engines separate the physical storage media from the application logic. The interface furthermore offers functions to measure performance and for fault management.

File System – GraphFS

The decision to re-develop the File System from the ground up is based on avoiding the paradigm shift between graph-oriented data base logic and structural storage of the data in files.

Instead the objects are stored as data streams based on blocks in the Storage Engines.

The storage in unstructured data streams is high-performing. The extend based storage ensures defined entry points.

The interface between File System and Storage Engines is reduced to the provision of a data stream. Thus, the access is not limited to locally tied storage systems, but can also be tied via TCP/IP or iSCSI to external systems.

To guarantee high-performance access to the objects, different Meta data (e.g. type of data objects) are indexed together with a clear Object Location. This contains all information to specifically access any object. Moreover, one's own Meta data can be allocated to the data objects and can be evaluated.

The File System offers the following functions:

- Storage of the data in netlike structures
- High-performance storage of both structured data (objects) as well as binary data
- Automatic data structuring and optimization
- Information LifeCycle Management beyond hardware limitations
- High performance through direct connection with the Data Base Management System
- Versioning of data objects

Database Management System - GraphDB

Software internal data organization is the job of the Database Management System. This includes for example the structured and high-performance object storage and provisioning. It also provides the interfaces, which enable the interaction with the system.

All data access takes place formalized in the query language Graph Query Language (GQL). The syntax is oriented on SQL, but has been adapted for the interaction with graph-oriented data structures.

The main feature of sones GraphDB is the data management of objects, which can be linked to each other in graph structures. When making a query it is possible to also evaluate data, which is indirectly linked (through several objects). The objects are also provided in the graph structure for evaluation. An additional mapping (like e.g. in the case of relational data bases) is therefore not necessary.

The attributes of the data objects can be defined free of type errors (C# data types). Complex structures can be shown through lists and object dependencies.

The database offers the following functions:

- Attributes of objects can be defined free of type errors (C# data types)
- Complex types can for example be shown through object referencing
- Inheritance of objects (for more specific standardization of types or for expansion)
- Editing of objects
- Versioning of objects
- Data safety (redundancy, mirroring, striping, backup) can be defined on object level
- Data integrity is ensured via checksums (CRC32, SHA1, MD5)
- Authorization concept for the data base can be defined

DataStorage – GraphDS

GraphDS provides the interface to the product. A GQL command line as well as a user interface is provided for administration. The system can also be accessed via Webservice, REST or WebShell.

The Graph Query Language (GQL)

The challenge of the graph-oriented data bases is to make the advantages of this approach also easy to use for the users and applications.


Currently 2 approaches are being pursued:

There is the query language SPARQL. It is oriented on RDF, which makes an adaption for Database Management Systems more difficult.

The second approach is to specifically track resp. evaluate the edges. This solution is flexible and high-performing, however, this solution is difficult to implement without knowledge of programming and requires proximity to the data model (e.g. API).

sones therefore decided to develop its own query language modeled on SQL (Graph Query Language - GQL). Thus, the learning requirements are very small for persons with SQL knowledge. GQL can be accessed through several interfaces. Support of SPARQL is planned.

The Graph Query Language offers the following functions:

- Functionalities analog SQL (CREATE, INSERT, UPDATE, ALTER, SELECT, etc.)
 - Graph theoretical functions, e.g. pathfinding or weighing of edges
- 

At a glance (USP's)

! **Universal access**

With Graph DS ones offers a data base solution for problems which can only be shown in the relational models with big efforts without giving up their advantages (standardized query language, structuring of the data). The model recommends itself for the illustration of netlike (e.g. social networks) or hierarchical structures.

! **More speed**

A higher performance for queries than what was possible with the previously existing solutions is reached by using graph theoretical algorithms.

The system is optimized for the use of multi / many core processors. Complex queries are processed simultaneously in several threads.

! **Scalability**

The performance of the Database Management System is independent from the data amount.

GraphDS can be integrated in existing structures, since it is a transparent solution, which supports standardized interfaces and offers an easy to learn query language.

! **Flexibility**

The flexible storage system makes object and attribute changes possible at any time during runtime.

! **Lower costs**

Significant decrease in expenses for implementation as well as operating costs.

Sones GraphDB uses the .NET framework consistently. Therefore, the system is only offered on .NET supported operating systems. The compatibility with the mono project will be ensured in the product test.



sones GmbH
February 2010

sones GmbH
Eugen-Richter-Str. 44
99085 Erfurt
Germany

Tel.: +49(0) 361 - 30 26 25 0
Fax: +49(0) 361 - 244 500 8

© 2010 sones GmbH. All rights reserved. sones as well as the according logos are registered trademarks of the sones. All other names of products and services are trademarks of their respective owners. This document is provided for information purposes only and the contents hereof are subject to change without notice.

This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronical or mechanical, for any purpose, without our prior written permission.