



sones Whitepaper

GraphDB



Die sones GraphDB

Bei der sones GraphDB handelt es sich um ein neuartiges, graphenbasiertes objektorientiertes Datenmanagementsystem. Dieses ermöglicht die effiziente Speicherung, Verwaltung und Auswertung extrem komplexer, hochgradig vernetzter Datenbestände. Die ganzheitliche Lösung verbindet die Vorteile einer Dateiablage mit den Möglichkeiten eines Datenbankmanagementsystems. Unstrukturierte Daten (z.B. Videodateien), semistrukturierte Daten (Metadaten, z.B. Logfiles) und strukturierte Daten können miteinander verknüpft und somit aus einer Hand verwaltet und bei Bedarf ausgewertet werden.

Die Idee für die Neuentwicklung einer auf der Graphentheorie bestehenden Datenbank entstand Anfang 2005 aus Unzufriedenheit über die bestehenden Lösungen am Markt. Diese boten für Abfragen auf stark verknüpfte Daten oft keine oder nur unzureichende Ad-hoc Analyse-Funktionen. Zudem waren diese Systeme den sich mit hoher Dynamik ändernden Daten- und Informationsstrukturen, die Applikationen insbesondere im Web-Umfeld verarbeiten müssen, meist nicht in ausreichendem Maße gewachsen. Fokus der Entwicklung war es daher, diese bisher komplexitätsbedingt kaum oder nur mit höchstem Aufwand lösbaren Aufgabenstellungen einem effizienten Datenhandling und einer Real-time-Analyse zuzuführen. Auf Basis völlig eigenständiger und neuartiger Konzepte wurde eine von Grund auf neue Datenbanktechnologie entwickelt.

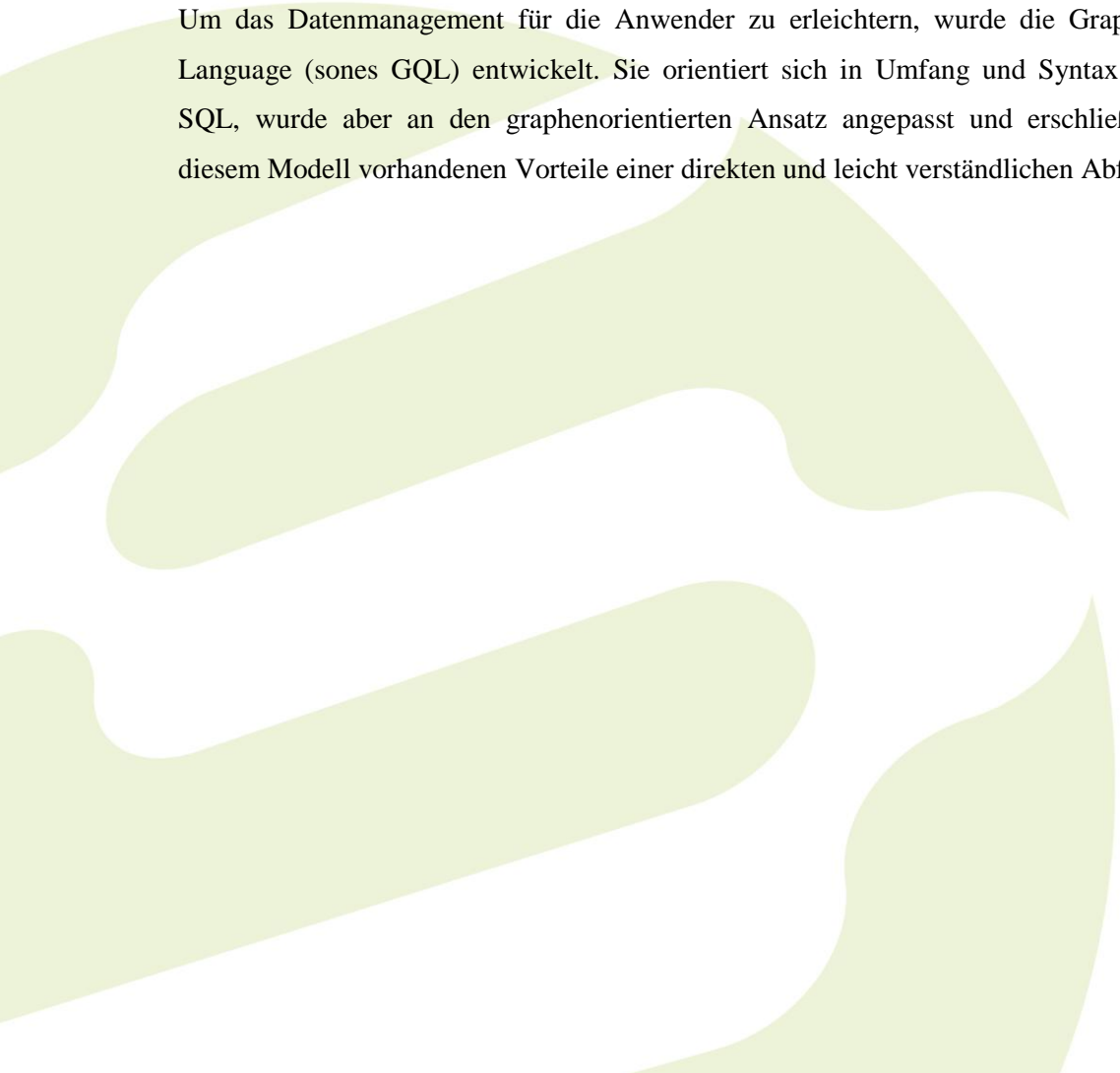
Die Lösung beinhaltet eine objektorientierte Speicherlösung (sones GraphFS) zur performanten Datenspeicherung, sowie das Graphenorientierte Datenbankmanagement-System (sones GraphDB), welches die Daten miteinander verknüpft und deren performante Auswertung ermöglicht. Die Zusammenfassung dieser beiden Teile in eine Software bietet den Vorteil einer hocheffizienten Verarbeitung, da auf jegliche Abstraktionsschicht verzichtet werden kann. Der für den Betrieb einer Datenbanklösung sonst übliche Aufwand wird durch diese Lösung extrem verringert.

Die objektorientierte Speicherlösung (sones GraphFS) speichert auch unstrukturierte und semistrukturierte Daten, wie z.B. Dokumentenformate, Bilder, etc., effizienter als Dateisysteme, welche dem Betriebssystem zugrunde liegen. Durch dieses Prinzip wird

eines der wesentlichen Performance-Probleme in bisherigen objektorientierten Datenbanken, Mapping der Objektmodelle in relationale Tabellen, erfolgreich überwunden. Für das Dateisystem ist ein Patent beantragt.

Das Graphenorientierte Datenmanagement (sones GraphDB) bietet hohe Flexibilität bei der Datenstrukturierung und ermöglicht die direkte Verknüpfung der Daten ohne zusätzliche Hilfskonstrukte, wie z.B. Zuordnungstabellen in Relationalen Datenbanken. Da sich das Datenmodell an gängigen Programmiersprachen orientiert, entsteht keine Diskrepanz zwischen Applikationslogik und Datenspeicherung. Die Komplexität ist im Vergleich zu den bestehenden relationalen Lösungen teilweise drastisch geringer. Der sich daraus ergebende Vorteil besteht unter anderem darin, dass bei zunehmender Verknüpfung der Daten die Komplexität nicht exponentiell sondern linear wächst. Ein weiterer entscheidender Vorteil liegt in der Fähigkeit der Datenbank, während der Laufzeit Datenstrukturen ändern oder ergänzen zu können. Neue Anforderungen seitens der Anwendung können daher ohne aufwändige Neuprogrammierung eines relationalen Datenbankschemas realisiert werden.

Um das Datenmanagement für die Anwender zu erleichtern, wurde die Graph Query Language (sones GQL) entwickelt. Sie orientiert sich in Umfang und Syntax zwar an SQL, wurde aber an den graphenorientierten Ansatz angepasst und erschließt die in diesem Modell vorhandenen Vorteile einer direkten und leicht verständlichen Abfrage.

An abstract graphic consisting of several overlapping, organic, light-green shapes that resemble stylized waves or flowing liquid. These shapes are positioned in the lower half of the page, partially overlapping the text area and extending towards the bottom edge.

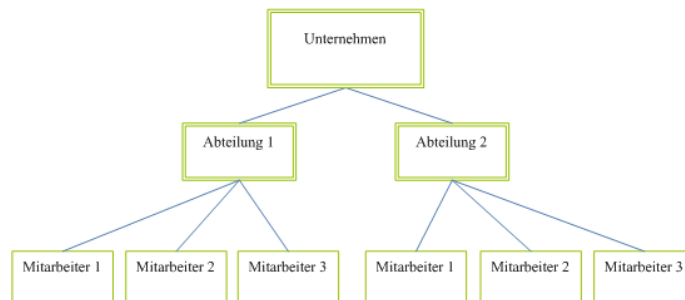
Der Markt

Das bedeutendste Instrument zur elektronischen (Massen-) Datenverwaltung ist das **Datenbankmanagementsystem**. Seine Aufgabe ist es, Daten effizient, widerspruchsfrei und dauerhaft zu speichern, sowie dem Nutzer bedarfsgerecht zur Verfügung zu stellen.

Im Laufe der Software-Entwicklung wurden hierfür mehrere Konzepte entwickelt, welche sich grundlegend voneinander unterscheiden.

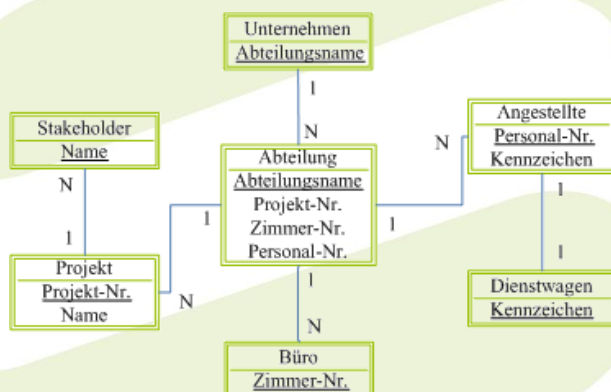
Hierarchische Datenbanken

Das Modell der hierarchischen Datenbanken ist das Älteste. Die Daten werden in einer Baumstruktur, ausgehend von einem „Wurzel“-Datensatz abgelegt. Abhängigkeiten zwischen Datensätzen werden als „Eltern-Kind“-Beziehung in



der Hierarchie abgebildet. Hierarchische Datenbanksysteme sind inzwischen nahezu vollständig verdrängt worden. Nur in Banken und Versicherungen werden diese noch für ältere Anwendungen betrieben. Ein Vertreter ist IMS/DB von IBM.

Das Konzept der hierarchischen Datenbanken wird teilweise in dokumentenorientierten Datenbanken wiederverwendet, wenn z.B. Informationen in XML-Dateien hierarchisch strukturiert gespeichert sind.



Relationale Datenbanken

Das relationale Datenbankmodell ist das derzeit am Weitesten verbreitete.

Die Daten werden zeilenweise in Tabellen gespeichert. Für jeden Datensatz (Zeile) können (in Spalten) beliebig viele Attribute gespeichert werden.

Abhängigkeiten zwischen Datensätzen können mittels gemeinsamer Verwendung von eindeutigen Attributen (Schlüsseln) abgebildet werden. Die Struktur des Datenmodells wird bei Erstellung (oder Modifikation) festgelegt.

Relationale Datenbanken haben sich in den vergangenen 20 Jahren durchgesetzt, da sich mit dem Modell die Anforderungen der Strukturierung (in Tabellen) und Verknüpfung (durch Referenzierung von Datensätzen untereinander) von Daten gut erfüllen lassen. Zudem gibt es eine standardisierte Abfragesprache (SQL), welche sich u.a. gut für Auswertungen nutzen lässt. Das relationale Datenbankmodell erreicht seine Grenzen, wenn Daten nicht direkt, sondern über mehrere Relationen voneinander abhängig sind (Zum Beispiel lässt sich eine mehrstufige Hierarchie nicht gezielt mit einem SELECT Statement auswerten). Zudem müssen oft zusätzliche, künstliche Schlüsselattribute angelegt werden, um die Komplexität von Verknüpfungen zu reduzieren und somit die benötigte Performance sicherzustellen. Dies wiederum erhöht die Komplexität des Modells und benötigt weiteren Speicherplatz.

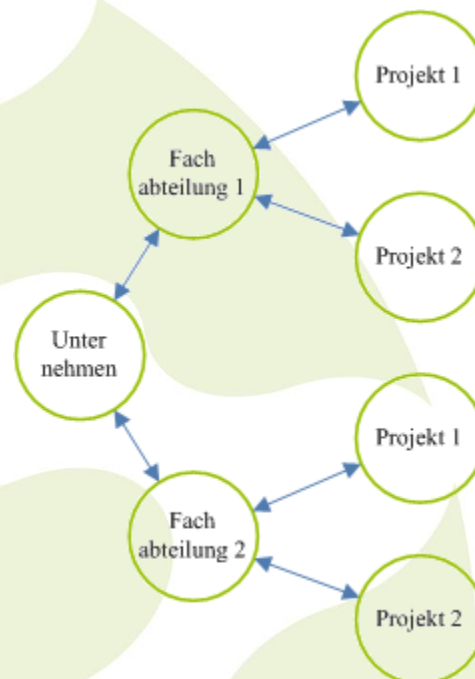
Für relationale Datenbanksysteme gibt es viele standardisierte Schnittstellen und eine große Anzahl nützlicher Tools.

Die bekanntesten kommerziellen Datenbanken sind jene von Oracle (Stand Jan. 2010: Version 11g), Microsoft SQL Server und IBM DB/2. Die bekanntesten Open Source Datenbanken sind MySQL und Postgres.

Objektorientierte Datenbanken

Bei objektorientierten Datenbanken handelt es sich um ein „vergleichsweise“ neues Konzept (Entwicklung seit Ende der 80er Jahre – gemeinsam mit den objektorientierten Programmiersprachen).

Die Struktur des Datenmodells wird bei Erstellung (oder Modifikation) festgelegt. Alle ein Objekt beschreibenden Attribute werden bei der Datenmodellierung zu einer Objektklasse zusammengefasst.



Beziehungen zwischen Objekten werden (wie bei objektorientierten Programmiersprachen) durch Referenzen dargestellt. Der Vorteil gegenüber den relationalen Datenbanken besteht darin, dass für Abhängigkeiten keine Hilfskonstruktionen (künstliche Schlüsselattribute, Zuordnungstabellen) konzipiert werden müssen. Dies verringert den Aufwand der Erstellung, sowie die Komplexität der Datenbank. Da sie nicht die Performance von relationalen Datenbankmanagementsystemen erreichen, haben objektorientierte Datenbanken noch keine große Verbreitung. Kommerzielle Lösungen sind z.B. Versant, InterSystems (Cachè) und Objectivity (Objectivity/DB).

Dokumentenorientierte Datenbanken

Ziel der dokumentenorientierten Datenbanken ist es, Datenstrukturen nicht bei der Konzeption des Datenbankmodells festzulegen. Die Daten werden unstrukturiert gespeichert und bei Bedarf vollständig ausgewertet. Anbieter dokumentenorientierter Datenbanken bieten meist zusätzliche Lösungen, um die Performance der Auswertungen bei großen Datenmengen sicherzustellen.

Vorteil dieser Lösung ist die Flexibilität beim Datenimport, da keine Vorgaben an die Datenstruktur gemacht werden. Dies erhöht allerdings den Aufwand bei der Auswertung der (sogenannten „semi-strukturierten“) Daten.

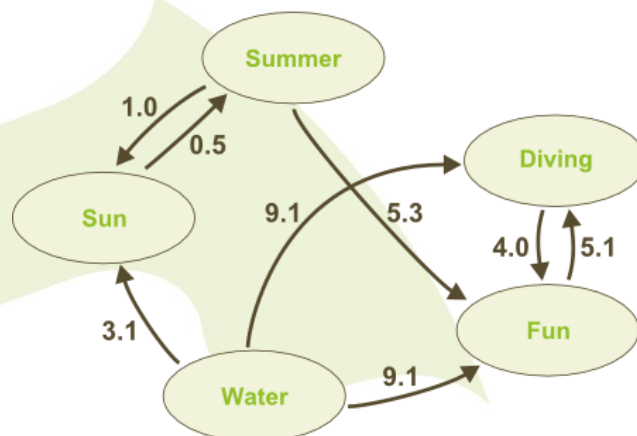
Die Apache CouchDB, MongoDB oder Amazons SimpleDB sind Beispiele für dokumentenorientierte Datenbanken.

Graphenorientierte Datenbanken

Graphenorientierte Datenbanken speichern Knoten und Beziehungen zwischen Knoten. Eine indirekte (komplexe) Beziehung (z.B. zwischen Objekt 1 und 3) besteht, wenn ein Objekt ein zweites referenziert und dieses wiederum das Dritte.

Mehrstufige Beziehungen werden als Graphen bezeichnet.

Graphenorientierte Datenbanken ermöglichen die Abfrage indirekter Relationen über mehrere Kanten hinweg. Die Kanten können zudem gewichtet werden, um den Grad der Beziehung darzustellen. Somit können komplexe und indirekte Abhängigkeiten in Datenbanken einfach



abgebildet und deren Auswertung performant ermöglicht werden.

Bei der Auswertung der Daten ist die Ergebnisstruktur nicht auf Listen beschränkt, sondern die Daten können als Graph strukturiert aufbereitet werden. Vorteil dieses Ansatzes ist, dass in der datenverarbeitenden Applikation der Arbeitsschritt der Datenstrukturierung gespart wird. Die GraphDB der sones GmbH ist eine Graphenorientierte Datenbank.

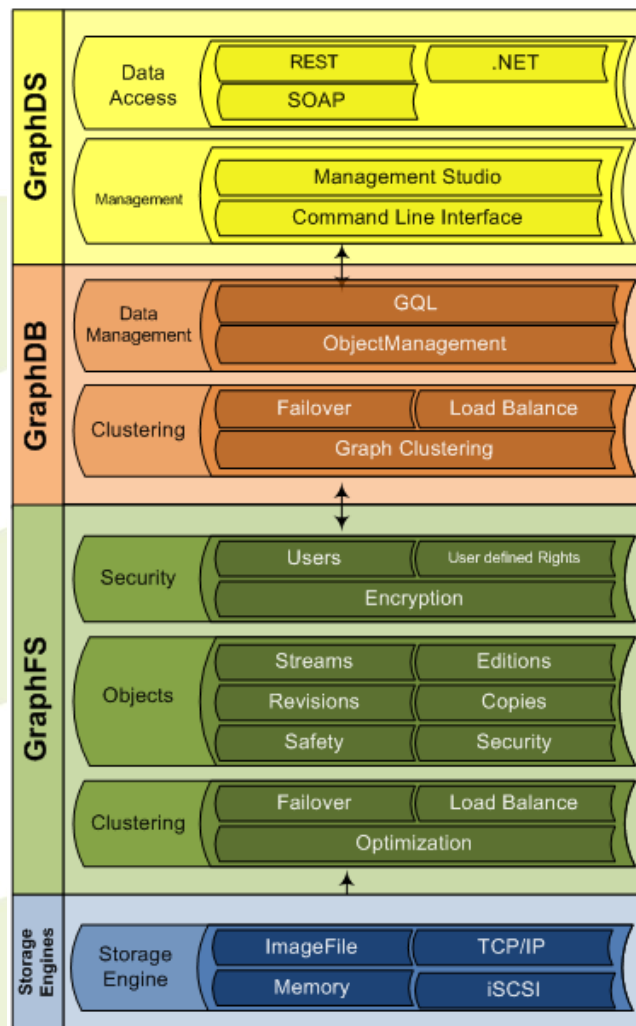


Die Architektur

Bei der sones GraphDB handelt es sich - im Gegensatz zu den meisten objektorientierten Ansätzen - um eine vollständige Eigenentwicklung. Der Ansatz einer kompletten Neuentwicklung wurde vor allem aus dem Grund gewählt, um eventuelle Legacy-Problematiken ausschließen zu können. Der gesamten Entwicklung liegt der Ansatz zugrunde, etablierte Ansätze zu hinterfragen und gegebenenfalls neu zu durchdenken.

Die Grafik veranschaulicht die Struktur der sones GraphDB. Es besteht aus vier Anwendungsschichten - dem StorageLayer, dem Dateisystem, dem Datenbankmanagementsystem sowie dem Data Access Layer.

Diese, sowie die zur Verfügung stehenden Schnittstellen werden im Anschluss detailliert erläutert.



Storage Engines

Die Storage Engines kapseln die physikalischen Speichermedien von der Applikationslogik ab. Die Schnittstelle bietet zudem Funktionen zur Performancemessung sowie dem Fehlermanagement.

Dateisystem - GraphFS

Die Entscheidung, das Dateisystem von Grund auf neu zu entwickeln, ist darin begründet, den Paradigmenbruch zwischen graphenorientierter Datenbanklogik und strukturierter Speicherung der Daten in Dateien zu vermeiden.

Stattdessen werden die Objekte als Datenströme blockbasiert auf StorageEngines gespeichert.

Die Speicherung in unstrukturierten Datenströmen ist hochperformant. Die extendbasierte Speicherung stellt definierte Einsprungspunkte sicher.

Die Schnittstelle zwischen Dateisystem und StorageEngines ist auf die Bereitstellung eines Datenstroms reduziert. Somit ist der Zugriff nicht auf lokal angebundene Speichersysteme begrenzt, sondern kann auch per TCP/IP oder iSCSI an externe Systeme angebunden sein.

Um performanten Zugriff auf die Objekte zu gewährleisten, werden verschiedene Metadaten (z.B. Art der Datenobjekte) zusammen mit einer eindeutigen ObjectLocation indexiert. Diese enthält alle Informationen, um gezielt auf ein beliebiges Objekt zuzugreifen. Desweiteren können zu Datenobjekten eigene Metadaten zugeordnet und ausgewertet werden.

Das Dateisystem bietet die folgenden Funktionen:

- Speicherung der Daten in netzartigen Strukturen
- Performante Speicherung sowohl von strukturierten Daten (Objekten) als auch von Binärdaten
- Automatische Datenstrukturierung und -optimierung
- Information LifeCycle Management über Hardwaregrenzen hinweg
- Hohe Performance durch direkte Verbindung mit dem Datenbankmanagementsystem
- Versionierung von Datenobjekten

Datenbankmanagementsystem - GraphDB

Aufgabe des Datenbankmanagementsystems ist die softwareinterne Datenorganisation. Dies umfasst z.B. die strukturierte und performante Objektspeicherung und -bereitstellung. Es stellt zudem die Schnittstellen, die die Interaktion mit dem System ermöglichen, bereit.

Alle Datenzugriffe erfolgen formalisiert in der Abfragesprache Graph Query Language (GQL). Die Syntax orientiert sich an SQL, ist jedoch für die Interaktion mit graphenorientierten Datenstrukturen angepasst worden.

Das Hauptmerkmal der sones GraphDB ist die Datenhaltung von Objekten, welche in Graphenstrukturen miteinander verknüpft werden können. Bei der Abfrage ist es möglich, auch Daten auszuwerten, welche indirekt (über mehrere Objekte hinweg) miteinander verknüpft sind. Die Objekte werden zur Auswertung ebenfalls in der Graphenstruktur bereitgestellt. Ein zusätzliches Mapping (wie z.B. bei relationalen Datenbanken) ist deshalb nicht notwendig.

Die Attribute der Datenobjekte können typischer (C#-Datentypen) definiert werden. Komplexe Strukturen lassen sich über Listen oder Objektabhängigkeiten abbilden.

Die Datenbank bietet die folgenden Funktionen:

- Attribute von Objekten können typischer (C# Datentypen) definiert werden
- Komplexe Typen können z.B. durch Objektreferenzierung abgebildet werden
- Vererbung von Objekten (zur genaueren Typisierung oder zur Erweiterung)
- Editionierung von Objekten
- Versionierung von Objekten
- Datensicherheit (Redundanz, Spiegelung, Striping, Backup) kann auf Objektebene definiert werden
- Datenintegrität wird durch Prüfsummen (CRC32, SHA1, MD5) sichergestellt
- Berechtigungskonzept für die Datenbank kann definiert werden

DataStorage - GraphDS

Das GraphDS stellt die Schnittstellen zum Produkt zur Verfügung. Zur Administration wird eine GQL-Kommandozeile sowie eine Benutzeroberfläche bereitgestellt. Auf das System kann zudem via Webservice, REST oder WebShell zugegriffen werden.

Die Graph Query Language (GQL)

Eine Herausforderung der Graphenorientierten Datenbanken besteht darin, die Vorteile dieses Ansatzes auch für Anwender und Applikationen einfach nutzbar zu machen.


Derzeit gibt es zwei verfolgte Lösungsansätze:

Es gibt die Abfragesprache SPARQL. Sie orientiert sich am RDF, was eine Adaption für Datenbankmanagementsysteme erschwert.

Der zweite Ansatz besteht darin, die Kanten gezielt nachzuverfolgen bzw. auszuwerten. Diese Lösung ist flexibel und performant, allerdings ist diese Lösung nur schwer ohne Programmierkenntnisse umsetzbar und erfordert Nähe zum Datenmodell (z.B. API).

Die sones GmbH hat sich deshalb dazu entschieden, eine eigene an SQL angelehnte Abfragesprache (Graph Query Language - GQL) zu entwickeln. Somit sind die Lernaufwände mit SQL-Kenntnissen sehr gering. Auf die GQL kann über mehrere Schnittstellen zugegriffen werden. Die Unterstützung der SPARQL ist geplant.

Die Graph Query Language bietet die folgenden Funktionen:

- Funktionalitäten analog SQL (CREATE, INSERT, UPDATE, ALTER, SELECT, etc.)
 - Graphentheoretische Funktionen, z.B. Wegfindung oder Gewichtung von Kanten
- 

Auf einen Blick (USP's)

! **Universeller Zugriff**

Die sones GmbH bietet mit der GraphDB eine Datenbank-Lösung für Probleme, welche sich nur aufwändig in den relationalen Modellen abbilden lassen, ohne jedoch deren Vorteile (standardisierte Abfragesprache, Strukturierung der Daten) aufzugeben. Das Modell empfiehlt sich für die Abbildung netzartiger (z.B. SocialNetworks) oder hierarchischer Strukturen.

! **Mehr Geschwindigkeit**

Durch die Verwendung graphentheoretischer Algorithmen wird eine höhere Performance bei Abfragen erreicht, als dies mit bestehenden Lösungen bisher möglich ist.

Das System ist für die Verwendung von Multi- / Many-Core Prozessoren optimiert. Komplexe Abfragen werden parallel in mehreren Threads abgearbeitet.

! **Skalierbarkeit**

Die Performance des Datenbankmanagementsystems ist unabhängig von der Datenmenge.

Die GraphDB kann in bestehende Strukturen integriert werden, da es sich um eine transparente Lösung handelt, welche standardisierte Schnittstellen unterstützt und eine leicht zu erlernende Abfragesprache anbietet.

! **Flexibilität**

Durch das flexible Speichersystem sind Objekt- und Attributänderungen jederzeit zur Laufzeit möglich.

! **Geringere Kosten**

Deutliche Verringerung der Aufwände bei der Implementierung sowie Betriebskosten.

Die sones GraphDB verwendet konsequent das .NET-Framework. Das System wird deshalb ausschließlich auf von .NET unterstützten Betriebssystemen angeboten. Die Kompatibilität zum Mono-Projekt wird im Produkt-Test sichergestellt.



sones GmbH
Februar 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. Alle Rechte vorbehalten. sones sowie die entsprechenden Logos sind eingetragene Marken der sones GmbH. Alle anderen Namen von Produkten und Dienstleistungen sind Marken der jeweiligen Firmen. Die Angaben im Text sind unverbindlich und dienen lediglich zu Informationszwecken. Produkte können länderspezifische Unterschiede aufweisen.

In dieser Publikation enthaltene Informationen können ohne vorherige Ankündigung geändert werden. Die vorliegenden Angaben werden von sones bereitgestellt und dienen ausschließlich Informationszwecken. sones übernimmt keinerlei Haftung oder Garantie für Fehler oder Unvollständigkeiten in dieser Publikation. Aus den in dieser Publikation enthalten Informationen ergibt sich keine weiterführende Haftung.