Universität Stuttgart

Institut für Parallele und Verteilte Systeme
Anwendersoftware

Einführung in die Datenbanken, Sensornetze und Filesysteme

Seminararbeit

Ausarbeitung im Rahmen des Studienprojekts „SIMPL“
(SS 2009)

Betreuer: Peter Reimann

Firas Zoabi

Stuttgart, 26.06.2009

Einführung in die Datenbanken, Sensornetzwerke und Filesysteme

Firas Zoabi

1 Einleitung

„SIMPL“ steht für *SimTech: Information Management, Processes and Languages.*
In „SIMPL“ geht es darum, ein generisches erweiterbares System zu konstruieren, das den Zugriff auf Datenquellen innerhalb von einem BPEL-Prozess aus ermöglicht.

Diese Ausarbeitung ist eine Einführung in die oben genannte Themenfelder und verschafft uns einen Überblick über diese Themen indem die verschiedenen Grundbegriffe sowie die dazu gehörigen Technologien erläutert und näher betrachtet werden.

In diesem Dokument werden wir die drei Themen, die für die Informationssysteme in unserem Leben eine große Rolle spielen, kennenlernen.

Unter einer Datenbank wird ein logisch zusammengehöriger Datenbestand verstanden, der mithilfe eines Datenbankmanagmentsystems verwaltet wird. Datenbanksysteme sind heutzutage ein wichtiger Bestandteil von Informations- bzw. Softwaresystemen und damit ein kritischer Teil im Unternehmen bzw. im wissenschaftlichen Bereich, der für die Verfügbarkeit, Vollständigkeit und Korrektheit der Daten innerhalb eines Unternehmens bzw. Forschungsinstitut, wie z.B. im SIMPL Projekt, zuständig ist.
Wichtige Daten in mehreren Bereichen, wie in der Wissenschaft oder Informationssystemen können mithilfe von Sensornetzwerken erfasst werden. Ein Sensornetzwerk ist ein verteiltes System, die aus Sensorknoten besteht, die per Funk miteinander kommunizieren.

Ein Dateisystem bildet einen wichtigen Teil eines Betriebsystems. Es ist ein Verwaltungssystem, um Dateien im Computer abzulegen. Dateien werden mit Hilfe des Dateisystems gelesen, gespeichert oder verschoben.

Es werden verschiedene Punkte in folgender Reihenfolge behandelt:
In Kapitel 2 zuerst die grundlegenden Begriffe bzgl. Datenbanken deffiniert.

Danach folgt eine ausführliche Erläuterung der Schemaarchitektur nach ANSI-SPARC und der Systemarchitektur einer Datenbank mit Behandlung des Fünf-

Schichten-Modells. Dann folgen die Auseinandersetzungen bezüglich des Themas Dateisysteme, mit den Teilthemen: Externspeicherverwaltung, Blockzuordnung, die verschiedenen Arten der Blockzuordnungsverfahren und ext3 als Beispiel einer Technologie. Am Ende dieses Kapitels werden kurz verschiedene Datenbanken behandelt.

Als letztes gibt es das Kapitel 3, eine Einführung zum Thema Sensornetzwerke. Unter diesem Thema werden mehrere Aspekte, wie Hardware und Software in Sensornetzwerke und deren Kommunikation, sowie die Synchronisation und Kommunikation der Sensoren behandelt und abschließend zusammengefasst.

2 Datenbanken

In diesem Kapitel wurde das Teilthema Datenbanken grundliegend behandelt, indem zuerst die wichtigen Begriffe in dieses Umfeld definiert wurden.
Sowie auch die verschiedenen Architekturen, wie das Schichten-Modell. Im zweiten Teil von diesem Kapitel erläuterten wir Dateisysteme und weitere wichtige Aspekte der Datenbanken. Dieses Kapitel basiert hauptsächtlich auf [2] und [3].

2.1 Grundlegende Begriffe

Alle wichtigen und fachlichen Grundbegriffe werden erläutert, wie bspw. Datenbanksystem oder Datenmodell um die Inhalte dieses Dokuments besser nachvollziehen zu können.

2.1.1 Datenbanksystem

Ein Datenbanksystem ist ein System, um Daten elektronisch zu verwalten.
Die Aufgabe eines Datenbanksystems ist es, große Datenmengen zu verwalten, indem folgende Aspekte beachtet werden müssen:

- Effizienz beim Lesen und Speichern der Daten und bei der Bereitstellung der Daten für den Benutzer bzw. für Anwendungsprogramme.
- Dauerhaftigkeit der Daten.
- Wiederspruchsfreiheit und Eindeutigkeit der Daten

Ein Datenbanksystem besteht aus zwei Teilen, Einem Datenbankmanagementsystem (DBMS), das die Verwaltungssoftware darstellt, und dem eigentlichen Datenbestand bzw. die Datenbank (DB).

2.1.2 Datenbankmodell

Ein Datenbankmodell ist die Basis eines Datenbanksystems im theoretischen Sinne und legt die Modellierungskonstrukte fest. Ein Datenbankmodell beschreibt die Datenobjekte und die Operatoren, die auf diese Objekte angewand werden können.
Es definiert, auf welche Art die Daten innerhalb der Datenbank gespeichert und bearbeitet werden können. Ein Datenmodell besteht aus folgenden Aspekten:

- Generische Datenstruktur mit deren Hilfe konkrete Datenstrukturen festgelegt werden können. Dazu gibt es die Datendefinitionssprache (DDL) als Werk zeug, um die Datenstrukturen zu definieren.
- Operationen sind das Auslesen, Einfügen, Löschen und das Ändern von Dateninstanzen die zum Datenmanipulation dienen. Die Datenmanipulationssprache (DML) ist ein Werkzeug dafür.
- Bedingungen für die Daten innerhalb der Datenbank womit man den zulässigen Datenbankinhalt einschränken kann. Diese Bedingungen können statisch, d.h einen festen Zustand der DB definieren, oder können auch dynamisch sein, d.h Zustandsübergänge darstellen. Es existiert eine andere Bezeichnung für die Bedingungen und zwar Integritätsbedingungen fach. *Integrity constraints*.

2.1.3 Datenbankschema

Das Datenbankschema ist eine formale Beschreibung sowie eine konkrete Datenstruktur. Sie liegt fest, in welcher Form Daten in einer Datenbank gespeichert werden dürfen. Die Beschreibungssprache ist XML-Schema für XML.
Ein Beispiel für das Datenbankschema ist das relationale Datenbankmodell, das aus mehreren Relationen besteht. Jedes dieser Relationen besteht aus mehreren Tupel bzw. Datenobjekte.

2.1.4 Datenbanksprache

Die Datenbanksprache dient der Kommunikation zwischen dem Benutzer oder Anwendungsprogramm und der Datenbank. Eine Datenbanksprache ermöglicht das Definieren, Verwalten, Analysieren und das Bearbeiten von Daten innerhalb der Datenbank. Die am meisten verwendete Datenbanksprache ist SQL.

2.1.5 DB-Abfragen

Datenbankabfragen sind Ausdrücke einer formalen Sprache, z.B. einer Datenbanksprache, zur Suche, Bereitstellung und Filterung von Daten. DB-Abfragen sind die Befehle innerhalb einer Datenbanksprache und das Ergebnis einer Abfrage ist eine Teilmenge des Datenbestandes. Ein Beispiel für eine Abfragesprache ist XQuery als Abfragesprache für XML-basierte Info. Die Datenbanksprache SQL beinhaltet eine Abfragesprache für relationale Datenbanksysteme.

2.1.6 Metadaten

Metadaten bzw. Metainformationen sind eine Bezeichnung für Daten, die andere Daten nach bestimmten Aspekten beschreiben. Metadaten beinhalten z.B. die Eigenschaften eines Objektes. Metadaten eines Buches können z.B. der Name des Autors, das Erscheinungsjahr, der Verlag oder die ISBN-Nummer sein. Bei einer Computerdatei wären die Metadaten der Dateiname, das Datum oder die Zugriffsrechte.

2.2 Schemaarchitektur

In diesem Kapitel wird die Schemaarchitektur nach ANSI-SPARC grundlegend beschrieben. Die ANSI-SPARC-Architektur wurde von dem *Standards Planning and Requirements Committe* (SPARC) des *American National Standards Institute* (ANSI) im Jahr 1975 entwickelt.

Die ANSI-SPARC Architektur wird auch als Drei-Schema-Architektur bezeichnet. Diese Drei-Schema-Architektur ist eine Art Trennung in drei verschiedene Beschreibungsebenen für das Datenbankschema (siehe Abb.1).

Diese Architektur hat das Ziel, die Datenbank von Fehlern, unerwünschten Änderungen oder nachteiligen Auswirkungen von Seiten des Benutzers oder des Anwendungsprogramms zu schützen. Beispielweise nachteilige Auswirkung durch einen Hardware-Fehler, die zur Datenverlusst führen kann.

Es werden hier Sichten der Daten betrachtet. Erstens die Externe Ebene bzw. benutzerspeziefische Sichten, mit dem der Benutzer bzw. das Anwendungsprogramm interagiert und die Daten bearbeitet.

Die zweite Ebene ist die konzeptionelle Ebene. In dieser Ebene ist beschrieben, wie die Daten in der Datenbank untereinander in Beziehungen stehen. Diese Ebene ist ein Schema, das eine logische Sicht auf die Daten darstellt.

Die dritte und letzte Ebene, die interne Ebene befasst sich mit allen physischen Aspekten der Datenbank. Diese Ebene hat ausserdem direkten Zugriff auf den Datenbestand. Zu jeder Ebene gibt es ein Schema. Die externe Ebene wird mit der Benutzersicht mithilfe von separaten externen Schemata spezifiziert.

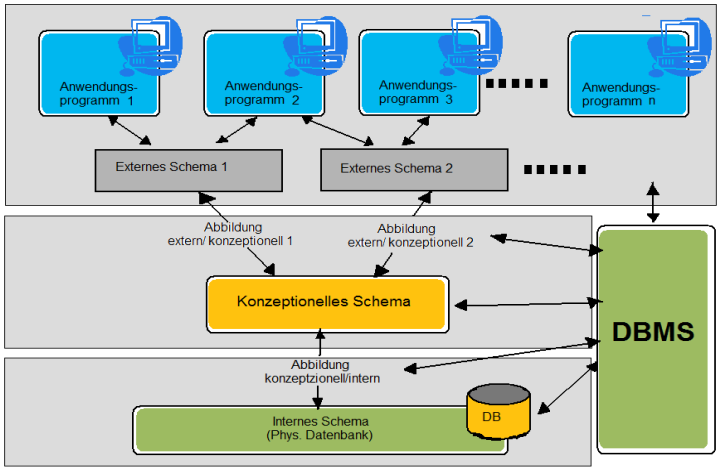

Abbildung 1. vgl. ANSI-SPARC Architektur [2]

Mit dieser benutzerspezifischen Sichtenbildung werden wichtige Ziele, wie die einfache Nutzung und der Zugriffschutz der Daten, erreicht. Mithilfe von Projektionen, Selektionen und Verbünden über das konzeptionelle Schema können sich externe Schemata an die Erfordernisse der Anwendungsprogramme anpassen. Über externe Schemata werden nicht alle Daten für die Anwendung sichtbar bzw. zugreifbar gemacht, damit reduziert sich die Komplexität und es wird eine vereinfachte Verarbeitung erreicht.

Die Funktion der konzeptionellen Ebene ist die Bearbeitung der Daten in der Datenbank anhand der Beschreibung, welche Daten in welcher logischen Form und mit welchen Beziehungen untereinander in der Datenbank gespeichert sind. Am Ende in der externen Ebene werden Daten durch eine Abbildung der Datentypen auf die Datentypen der Anwendungssprache abgebildet (siehe Abbildung 2). Dafür muss es möglich sein, das die Anwendung vom Datenbanksystem mit verschiedenen Anwendungsprogrammen und verschiedenen Programmiersprachen machbar sein.

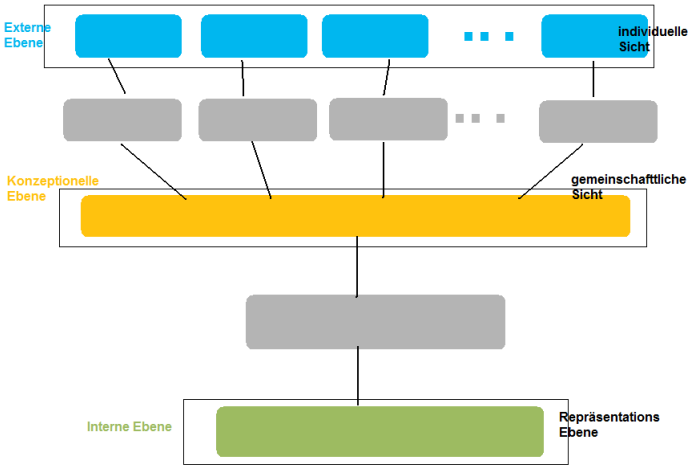

Abbildung 2. Grobe Architektur für Schnittstellen in der ANSI-SPARC Architektur [2]

Ein anderer wichtiger Aspekt, der durch diese Aufteilung erreicht wird ist, dass nicht alle Datenobjekte für den Anwender sichtbar sind. Dadurch wird eine starke Isolation der Daten bewirkt, was den Zugriffschutz verstärkt.

2.3 Systemarchitektur

In diesem Kapitel wird auf eine mögliche Systemarchitektur einer Datenbank in Form eines Schichtenmodells eingegangen. Wir werden das vereinfachte Schichten-Modell sowie das Fünf-Schichten-Modell kennenlernen. Das Schichten-Modell, auch bekannt als Schichtenarchitektur, ist eine Art Strukturierung für Informationssysteme. Innerhalb eines Schichten-Modells werden die einzelnen Teile eines Informationssystems in verschiedene Schichten mit eingeschränkten Abhängigkeitsbeziehungen untereinander zugeordnet.

2.3.1 Vereinfachtes Schichten-Modell

Durch ein solches Schichten-Modell für eine Datenbanksystemarchitektur wird eine hierarchische Systemstrukturierung erreicht. Bei solch einer Architektur sollen verschiedene wichtige Kriterien erfüllt werden:
- einfache Zerlegung des Datenbanksystems in wenig definierte Schichten.
- für jede Ebene eine allgemeine Beschreibung ihrer Funktionen, so dass sie implementierungsunabhängig wird.

In Abbildung 3 sehen wir ein vereinfachtes Schichtenmodell für eine Datenbanksystemarchitektur. Es wird gezeigt wie z.B. durch Separierung des Speichersystems, die Aufgaben der Extern- und Hauptspeicherverwaltung in diesem Modell hervorgebracht werden. Das Speichersystem ermöglicht die Benutzung der Zugriffseinheit „Seite“ im Datenbank-Puffer durch das Speichersystem und bietet einen seitenstrukturierten Datenbankcach für höhere Systemschichten.

Die Benutzung von Sätzen und Zugriffspfaden auf Seiten wird durch das Zugriffsystem ermöglicht.

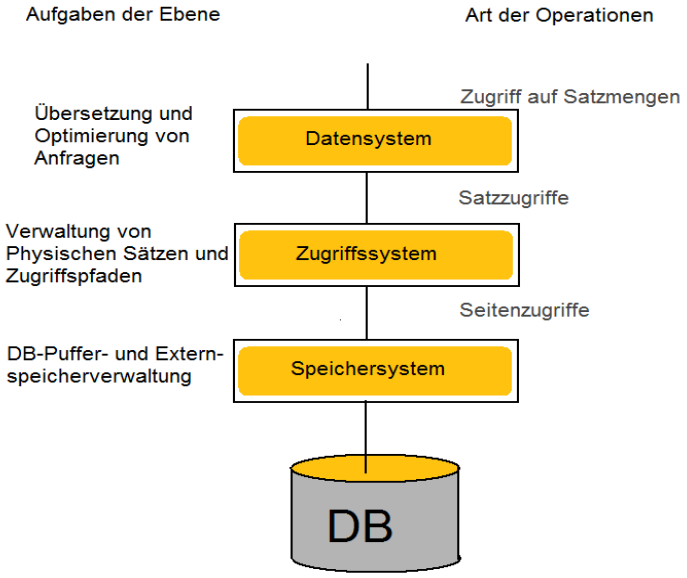

Abbildung 3. Graphische Abbildung für einfaches Schichtenmodell [2]

Das Zugriffssystem ist zuständig für die Abbildung von Sätzen und Zugriffspfaden auf Seiten und dadurch wird das satzweiser Zugriff auf die Objekte für die höhere Ebene bereitgestellt. Dann folgt das Datensystem, das eine Überbrückung zwischen der mengenorientierten Relationen und Sichten und die satzweiser Bearbeitung interner Sätze bildet.

2.3.2 Fünf-Schichten-Modell

Der Unterschied zu dem vereinfachten Schichtenmodell in Abbildung 3 ist, dass im Fünf-Schichten-Modell das Datensystem und das Speichersystem in zwei Schichten aufgespalten werden. Die Schichten bieten über ihre Schnittstellen Operationen an die darüber liegende Schicht an.

Jede Ebene ist aus vier Bausteinen konstruiert (siehe Abbildung 4):

1- Die Schnittstelle zu höheren Schichten, wo die typischen Operationen angeboten werden und als Übergangsstelle zwischen zwei Ebenen steht.

2- Die Addressierungseinheiten (unten) beinhalten die zugehörigen Objekte der betrachteten Schicht.
3- Die Addressierungseinheiten (oben) entsprechen den zugehörigen Objekten in der nächsthöheren Ebene bzw. Schicht.
4- Die Hilfsstrukturen sowie die Beschreibungsdaten, die zur Realisierung der Ebene benutzt werden.

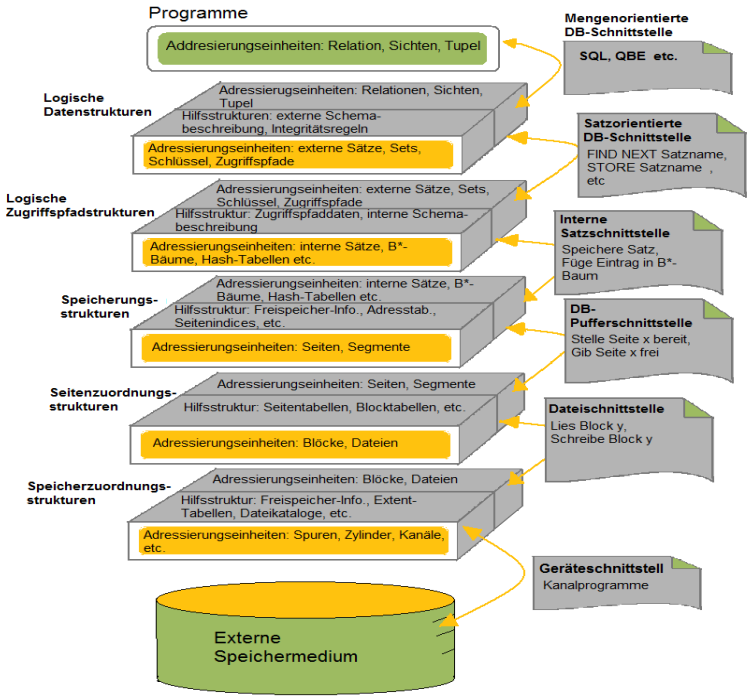

Abbildung 4. Das Fünf-Schichtenmodell [3]

Die Geräteschnittstelle repräsentiert die verwendeten externen Speichermedien und man greift darauf zu, die Schicht an sich repräsentiert das. Über die Dateischnittstelle greift man auf Dateien und Blöcke zu.

Die zweite Ebene, die Seitenzuordnungsstrukturen realisiert die DB-Pufferschnittstelle, die für die Erstellung der Segmente mit Seitengrenzen im DB-Puffer zuständig ist. Als Folge wird die Trennung von Segment und Datei sowie von Seite und Block ermöglicht. Dadurch wird die Abbildung zwischen Segmente bzw. Seiten und Dateien bzw. Blöcken erreicht. Die Ebene der Speicherungsstrukturen stellt Speicherungsstrukturen dar, wie zum Beispiel interne Sätze und physische Zugriffspfade, die von dieser Schicht auf Seiten in Segmenten abgebildet werden. Diese Ebene implementiert die interne Satzschnittstelle, die Sätze und Einträge in den Zugriffspfaden und Seiten bzw. in Segmenten abbildet. Die nächste Ebene, die Ebene der logischen Zugriffspfadstrukturen bietet satzorientierte DB-Schnittstelle an. Diese

Schnittstelle bietet Zugriff auf die externen Sätze. Durch die Schicht Speicherungsstrukturen wird die Unabhängigkeit von den Speicherungsstrukturen erreicht.

Zuletzt Ebene der logischen Datenstrukturen, die die mengenorientierte DB-Schnittstelle implementiert und die Bearbeitung und den Zugriff auf die Daten über die verschiedenen Sprachen behandelt. Bei dieser Ebene nutzt der Benutzer die Daten ohne sich mit den logischen Zugriffspfaden zu beschäftigen. (siehe Abbildung 4)

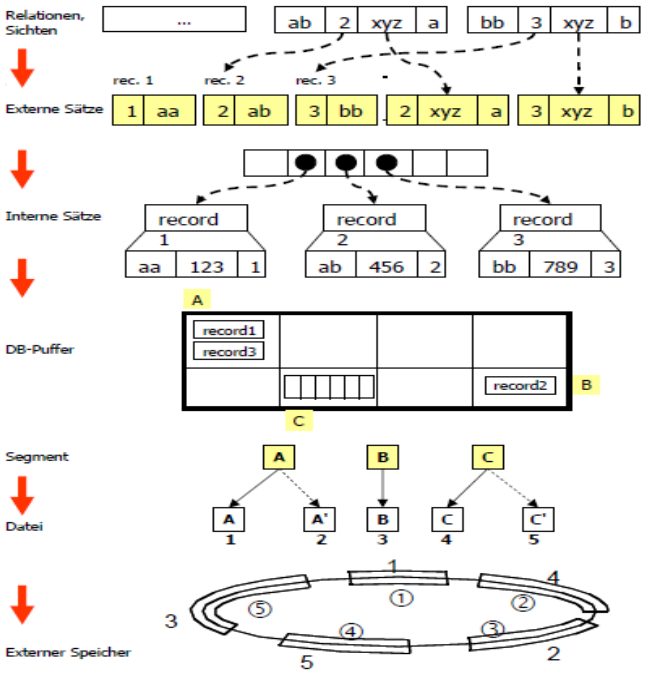

Abbildung 5. Datenvearbeitung innerhalb eines Fünf-Schichtenmodells [1]

In der Abbildung 5 können wir sehen, wie eine Instanz von Daten innerhalb der fünf Schnittstellen in verschiedene Formen abgebildet wird.

Diese Darstellung wird von oben nach unten betrachtet, wobei jeder Schritt uns zeigt, wie die Datensatz in die jeweiligen Ebenen von der Ebene der logischen Datenstruktur bis zur Speicherzuordnung-struktur aussieht.

In der oberen Ebene werden Datensätze, die in Form von Tupel und Relationen werden auf Externe-Sätze bzw. Sets abgebildet. Im zwieten Schritt werden die Externen-Sätze auf Interne-Sätze und Tabellen bzw. B*-Bäume abgebildet die wieder im nächsten Schritt als Seiten im DB-Buffer gespeichert werden. Im vierten Schritt wird die seitenformige Ablage in Segmente abgebildet und schliesslich werden die Segmente als Dateien und Blöcke abgebildet. Im Letzten Schritt werden die Dateien und Blöcke als Zylender und Spuren auf dem Exterspeichermedium abgelegt.

2.4 Dateisystem

In diesem Kapitel wurde anfangs eine kurze Einführung in die Dateisystemen erfasst, danach wurden die Themen Externspeicherverwaltung und Blockzuordnungverfahren, mit den verschiedenen Blockzuordnungsverfahren, erläutert. Unter Dateizuordnung bzw. Blockzuordnung versteht man die Zuordnung von Blocken zu einer Datei. Dabei gibt es zwei verschiedene Arten von Dateizuordnung (siehe Abbildung 7).
-Die statische Dateizuordnung, die bei Abspeicherung der physischen Blöcke nach aufsteigenden Blocknummern sich wegen einheitlichen Blockgröße die relative Ad resse eines Blockes zum Dateianfang leicht berechnen läßt.
-Die dynamische Extent-Zuordnung, die bei Erzeugung einer Datei bzw. Erweiterung, Speicherplatz nach Bedarf zuzuordnen.

2.4.1 Einführung in die Dateisysteme

Ein Dateisystem ist ein Verwaltungssystem für Dateien innnerhalb eines Datenträgers, das in Betriebsystemen genutzt wird. Es bietet eine Reihe von Operationen auf Dateien an. Ein Dateisystem ist für das Speichern, Lesen und Schreiben von Dateien zuständig und verwaltet die Dateien durch das sogenannte Zugriffs- und Ordnungssystem anhand ihrer Namen und computerinternen Dateiadressen.

Es gibt innerhalb eines Dateisystems zwei Arten von Dateien:

- Eine Datei, in der sich Daten befinden
- Eine Datei, die als Verzeichnis bezeichnet wird und mehrere Dateien beinhaltet bzw. sie referenziert.

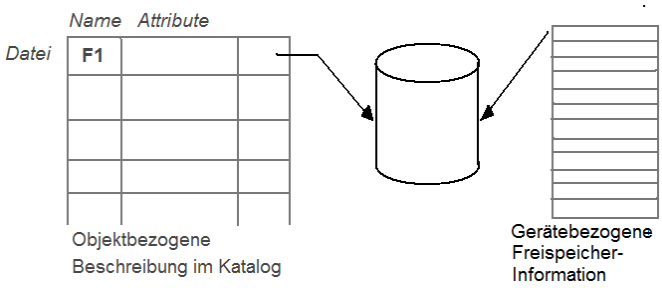

Abbildung 6. Allgemeine Dateiverwaltung [2]

Der Name einer Datei sowie seine Attribute bzw. Informationen werden als Meta-Daten bezeichnet.

Bei einem Dateisystem, das als untere Ebene innerhalb der Ebenen eines Betriebsystems positioniert ist, werden alle Angaben einer Datei, Wie bspw. Blocknummer, Sektor etc., physisch als Adressen auf die verschiedenen Speichermedien gespeichert. (siehe Abbildung 6)

2.4.2 Dateisystem Externspeicherverwaltung

Eine Externspeicherverwaltung hat innerhalb eines DBS folgende Aufgaben:
- Realisierung einer Speicherhierarchie mit einem automatisierten Verfahren zur Speicherverwaltung.
- Sie verschafft eine erhöhte Fehlertoleranz bei Speicherung und bei der Eingabe und Ausgabe von Blöcken.
- Verwaltung der Speichermedien
- Sie bildet die physischen Blöcke auf externe Speichermedien wie bspw. einer CD.

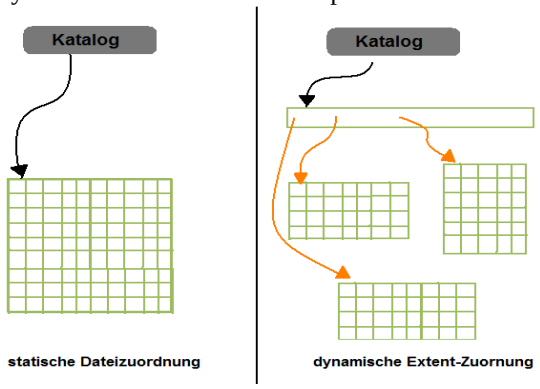

Abbildung 7. Statische Blockzuordnungen und dynamische Extent-Zuordnung [2]

2.4.3 Dynamische Blockzuordnung

Bei diesem Verfahren wird Speicherplatz während dem Erzeugen von Blöcken für eine bestimmte Datei auf dem Externspeicher zugeordnet. Im Fall von Leeren oder undefenierten Blöcken wird kein Speicherplatz dafür zugeordnet.
Man kann dieses Verfahren in folgenden Aspekten zusammenfassen:
- Eine große dynamische Tabelle von Block-Adressen
- Arbeitet mit einem Array von Block-Adressen (siehe Abbildung 8)
- Wahlfreier Zugriff

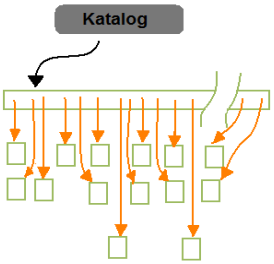

Abbildung 8. Dynamische Blockzuordnung [2]

Die Vorteile bei diesen Verfahren sind eine höhere Flexibilität durch die einfache Erweiterung der Block-Tabelle und ein wahlfreier Zugriff.

2.4.4 Versetzungverfahren

Das Ziel bei diesem Verfahren ist das Vermeiden von unnötigen Leistungsverlusten. Bei der normalen sequentiellen Zuordnung von Blöcken des Slots einer Spur, führt dies bei zyklischen Speichermedien dazu, dass es beim Lesen und Schreiben in der Datei zu Leistungsverlusten kommen kann. Deswegen funktioniert dieses Verfahren folgendermaßen (siehe Abbildung 9.):

Die sequentielle Verarbeitung der Blöcke wird wesentlich schneller, wenn die Blockreihenfolge innerhalb einer Spur nicht aufeinander folgt, sondern geändert wird, sodass die Bearbeitung bzw. Übertragung von mehreren Blöcken in einer Umdrehung erfolgt.

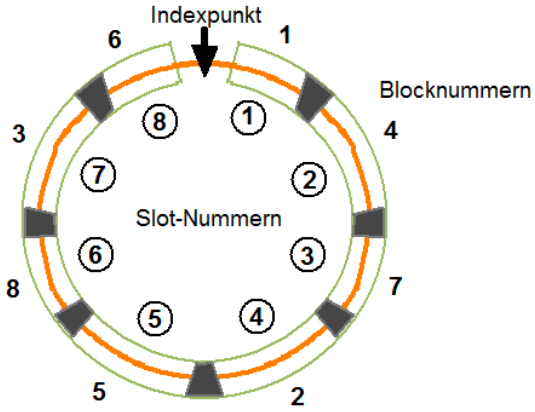

Abbildung 9. Versetzungsverfahren der Blockzuordnung [2]

2.4.5 Log-Strukturierte Dateien

Dieses Verfahren eignet sich nicht für DB-Belange mit großen Dateien und langfristiger Speicherung dieser Dateien, sondern es eignet sich für Anwendungen mit einer großen Zahl an kleinen Dateien, die meistens nur kurze Zeit gespeichert werden. Das Log-Strukturierten-Dateien Verfahren verhindert, dass wahlfreie Lesezugriffe, durch große Hauptspeicherpuffer geschehen.

Es werden alle Blöcke, in denen Änderungen stattgefunden haben, am Ende der Dateibearbeitung sequentiell auf einmal ans Ende der Datei geschrieben und organisiert so wie bei einer Log-Datei, der zyklisch überschrieben wird.

2.4.6 Ext3 als Beispiel-Technologie für ein Dateisystem

In diesem Teilkapitel werden wir das Ext-Dateisystem im Allgemeinen kennenlernen daraufhin wird das ext2-Dateisystem, das als Übergang zu ext3 gilt, mit seinen technischen Eigenschaften erläutert. Am Ende dieses Kapitels wird kurz der Aspekt „Journaling“ kurz erläutert.

Ext ist die Abkürzung für „extended file system “ Linux verwendet dieses Dateisystem. Es wurde im April 1992 entwickelt und hat das „Minix“ Dateisystem ersetzt. Mit diesem Dateisystem gab es mehrere Probleme, weshalb die zweite Version (ext2) entwickelt wurde.

Das Linux Dateisystem implementiert Basisaspekte, die vom Betriebssystem Unix abgeleitet sind. Dateien werden als „inods“ bezeichnet und Verzeichnisse sind einfach Dateien, die Listen mit Adressen für jede Datei beinhalten. Jede Datei wird durch eine Struktur „inod“ dargestellt und jeder dieser „inods“ beinhaltet die Dateibeschreibung, wie Dateityp, Zugriffsrechte, erstellende Personen, Zeitstempel, Größe und Adressen der Blöcke (siehe Abbildung 10)

Die Adressen der Blöcke sind in einer Datei allokiert und sind jeweils in eigenen „inod“ gespeichert.

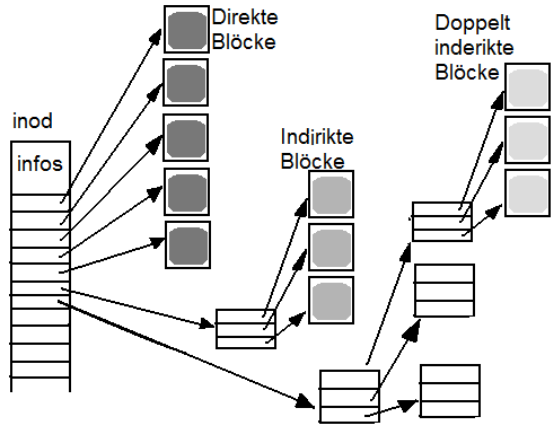

Abbildung 10. Darstellung der Blockverwaltung innerhalb Ext Dateisystem

Von Ext sind später mehrere Versionen entstanden, die solche Funktionalitäten wie Fragmentierung, Wiederherstellung gelöschter Daten und Kompressionsfunktionalität besitzen. ext2 oder „second extended file system“ ist die zweite Version vom Ext Dateisystem und arbeitet mit Blöcken, die verschiedene Größen haben von 1 kB bis zu 8 kB. Ext2 besitzt folgende technische Attribute:

-Die maximale Größe einer Datei ist 2 TiB.
-Die Maximale Länge des Dateinamens sind 255 Byte.
-Die Maximale Größe des Dateisystems sind 16 TiB.

Der Unterschied zwischen ext3, das später entwickelt wurde, und der vorherigen Version ext2 ist erstens, dass ext3 die Journaling-Erweiterung beinhaltet, womit ein Schutz gegen Beschädigungen der Metadaten, durch beispielweise Rechnerabstürze,

geleistet wird. Ein Journal ist eine Datei, in der Metadaten geschrieben werden bevor die eigentliche Datei auf das Dateisystem geschrieben wird, eine Art Zwischenspeicherung. Und der zweite Unterschied ist, dass Veränderungen der Dateisystemgröße online, während dieses Dateisystem im Betrieb ist, durchgeführt werden können.

Um sich einen detaillierteren Überblick zu verschaffen benutzen Sie die Quelle [4].

2.5 Datenintegrität, Transaktionskonzept und ACID-Eigenschaften

Im Bereich von DBSen sind die vier Aspekte, Datenintegrität, Transaktionskonzept und ACID-Egenschaften von großer Wichtigkeit. Dieses Kapitel wird die drei Aspekte Datenintegrität, Transaktionskonzept und ACID-Eigenschaften erläutern und in den Vordergrund stellen.

Datenintegrität:

Teilt sich in physische und logische Datenintegrität.

- Bei der physischen Datenintegrität versteht man, das alle Aspekte zum Sicherstellen gebündelt werden, damit die Daten während der Verarbeitung und Übertragung nicht beschädigt bzw. geändert werden oder verloren gehen. Die Datenintegrität setzt voraus, dass Daten an allen Stellen, wo sie übertragen werden, rekonstruierbar sein müssen.
- Die logische Datenintegrität befasst sich mit Datenkonsistenz und betrifft die Richtigkeit und Wiederspruchsfreiheit der Daten. Konsistenz einer Datenbank erreicht man, wenn sie den Konsistenzregeln genügt.

Transaktionskonzept:

Ein Transaktionskonzept ist ein wichtiges Charakteristikum im Rahmen der Datenbanksysteme. Transaktionen sind Verarbeitungseinheiten in eine DBMS und werden entweder ganz oder gar nicht ausgeführt. Das Transaktionskonzept an sich ist eine Zusicherung, sodass bei Fehlerzustände in eine DBMS, durch Wiederherstellung der konsistente Zustand zu Beginn der Transaktion, behoben werden können.

Ein weiterer Gesichtpunkt innerhalb dieses Aspekts ist, dass bei gleichzeitige Zugriffe von mehreren Benutzern auf den gleichen Datensatz in der Datenbank von dem DBMS verwaltet werden, sodass die Benutzer den Eindruck bekommen, als ob jeder alleine auf den Datensatz zugreift.

ACID-Eigenschaften:

Das sind die Eigenschaften des Transaktionskonzepts, die existieren müssen um ein Transaktionskonzept zu erfüllen. die ACID-Eigenschaften sind:

- **A**tomicity: Es wird verlangt, dass eine Transaktion als eine Einheit, die nicht in kleinere Einheiten zerlegbar ist, behandelt wird. Das heißt, dass entweder alle Änderungen für diese Transaktion durchgeführt werden oder gar keine.
- **C**onsistency: Nach Beendigung einer Transaktion wird ein konsistenter Datenbasiszustand erreicht, der alle definierten Integritätsbedingungen erfüllt. Sonst wird die Transaktion komplett zurückgesetzt.
- **I**solation: Diese Eigenschaft verlangt, dass bei mehreren gleichzeitig ausgeführten Transaktionen keinerlei gegenseitige Beieinflussungen enstehen.

Jede Transaktion muss so ausgeführt werden als wäre sie die einzige Transaktion.

- **D**urability: Es muss die Dauerhaftigkeit der von den Transaktionen gewährleistet durchgeführten Datenänderungen gewährleistet werden. Außerdem muss sicherge stellt werden, dass kein Verlust oder Fehler auf dieser Transaktion durch Soft ware- oder Hardwareproblemen vorkommt.

2.5 logische

2.6 Datenmodelle

Es werden verschiedene Datenbankmodelle vorgestellt, wie das Relationales-, Objekt-orientiertes Datenbankmodell etc.

2.6.1 Relationales Datenbankmodell

Eine wichtige Eigenschaft, was das relationale Datenbankmodell auszeichnet ist, dass dieses Datenbankmodell eine mengenorientierte Verarbeitung seiner Daten ermöglicht. Ein relationales Datenbankmodell unterscheidet sich von den anderen Arten des Datenbankmodells durch seine einfache Strukturierung. Es besteht aus einfachen Tabellen bzw. Relationen, deren Zeilen die Datenobjekte sind. Diese Zeilen nennt man auch Tupel. Diese Daten in den Tabellen werden mit Hilfe von Operatoren mengenorientiert verknüpft und verarbeitet. (siehe Abbildung 11)

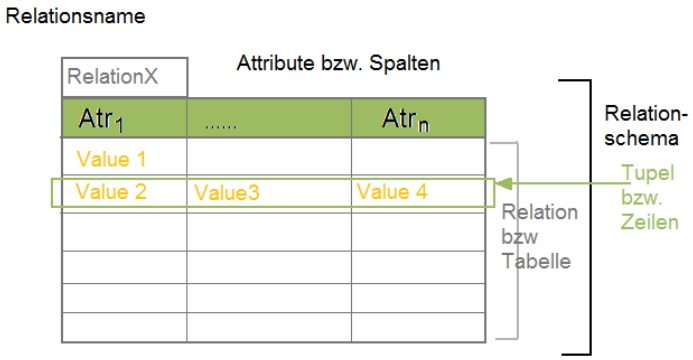

Abbildung 11. Ein Relationenschema

2.6.2 Hierarchisches Datenbankmodell

Ein Hierarchisches Datenbankmodell besteht aus Tabellen, die hierarchisch von einander abhängig sind. (siehe Abbildung 12)
Dieses Datenbankmodell besitzt besondere Eigenschaften:
- Zusammengehörige Daten, sind auf Einmal zugreifbar.
- Dieses Modell ermöglicht kurze Zugriffszeiten.
- Nachteile dabei sind dass der Benutzer den Zugriffspfad genau kennen muss, um auf Daten zuzugreifen. Der zweite Nachteil ist die geringe Flexibilität, weil es nur hiera-

rchisch gut funktioniert.

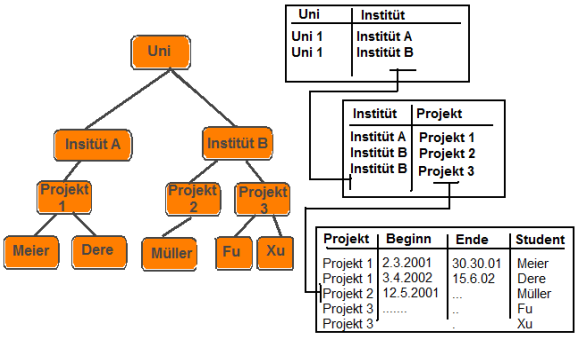

Abbildung 12. Ein Beispiel für hierarchisches Datenbankmodell

2.6.3 Netzwerkdatenbankmodell

Bei Netzwerkdatenbankmodellen, wie der Name schon sagt, handelt es sich um einen Netzwerkstruktur. Die Daten werden in einem Netzwerk angeordnet.

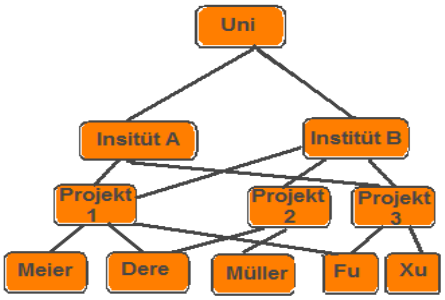

Abbildung 13. Beispiel für Netzwerkdatenbankmodellschema

Es ist ähnlich zum vorherigen Datenbankmodell, dem hierarchischen Datenbankmodell, aber mit dem Unterschied, dass in diesem Modell n:m Beziehungen zwischen Datenobjekte erlaubt sind, wobei die Datenobjekte mit Referenzen identifiziert werden. (siehe Abbildung 13).

2.6.4 Objektorientiertes Datenbankmodell

Dieses Datenbankmodell eignet sich eher als die anderen Datenbankmodelle für komplexere Anwendungsbereiche, wie z.B. Multimedia-Anwendungen, ingenieurwissenschaftliche Entwurfsanwendungen oder Architektur etc. Dieses Modell wird oft als evolutionär bezeichnet, weil es das relationale Datenmodell um weitere komplexe Objekte erweitert.

Objektorientierte Datenbankmodelle werden hauptsächlich im Bereich von objektorientierten Programmiersprachen eingesetzt. In diesen Modellen werden strukturell alle Objekte, die das gleiche oder ähnliche Verhalten und Eigenschaften besitzen in einem Objekttyp bzw. einer Klasse gekapselt. Weitere objektorientierte Abstraktionskonzepte wie Vererbung, Spezialisierung und Generalisierung sind auf die Objekttypen anwendbar. Jedes Objekt besitzt bestimmte Eigenschaften und Operationen, die z.B. zum Auslesen oder Manipulieren der Eigenschaften eines Objektes anbieten.

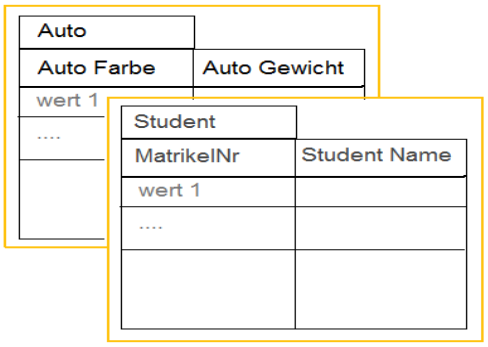

Abbildung 14. Daten als Objekte verwalten

2.6.5 XML – Datenbank

XML-Dokumente sind Textdokumente mit Unicode-Zeichen und sind mit jedem Text-Editor editierbar und lesbar, was XML-Dokumente leicht zu bedienen macht. Ein XML-Dokument besitzt eine definierte baumartige Struktur, die Element-, Attribute-, Kommentar- und Textknoten beinhaltet. Diese Eigenschaft charakterisiert das XML-Datenmodell als eine Baumstruktur.

Datenbanksysteme, die ein XML-Datenbankmodell verwenden, verarbeiten ihre Daten in einem XML-Format.Die Daten sind dann in einer baumartigen Hierarchie angeordnet und sind daher schlecht abbildbar auf relationale Modelle. Aus diesem Grund wird XML eher beim Übertragen von Daten als ein Austauschformat verwendet.

3 Sensornetzwerke

Es wird auf zwei Aspekten, im Bezug auf Sensornetzwerke, eingegangen. Zuerst werden wir die Sensornetzwerke im Allgemeinen, und als nächstes das TinyDB als ein Datenbanksystem für Sensornetzwerke im Spezifischen kennenlernen.
Die Informationen in diesem Kapitel basieren auf [6], [7] und [10].

3.1 Sensornetzwerke

Sensornetzwerke sind das dritte und damit das letzte Teilthema, das in diesem Dokument behandelt wird. In diesem Kapitel werden Sensornetzwerke in drei Punkte bzw. Teilthemen zergliedert und erläutert. Als erstens wird erläutert, was Sensoren überhaupt sind. Danach behandeln wir unter die Hardwarestruktur eines Sensors sowie die Softwareaspekte, die eine Hauptrolle in Sensornetzwerken spielen.

3.1.1 Sensoren

Ein Sensor ist eine Hardwarekomponente, die mit Batterien funktioniert und per Funk kommuniziert. Sensoren sind sehr klein und haben damit einen minimalen Bedarf an Strom. Außerdem haben sie die Aufgabe Daten zu erfassen.

Der geringe Strombedarf macht die Sensoren einzigartig, da sie mit den gleichen Batterien Monate lang betriebsfähig sein können.

Als Beispiel nehmen wir die *Mica Mote* Sensoren. Diese können bis zu sechs Monaten ohne Batteriewechsel arbiten.

Die *Mica Mote* Sensoren (siehe Abbildung 15) besitzen folgende Komponenten bzw. Eigenschaften:

- einen 8bit Atmel Microprocessor mit einer Leistung von 4Mhz.
- Einen RFM TR1000 Funkchip, der für Funkkommunikation für kleine Entfernungen zuständig ist. Er kommuniziert mit 40 kbits/Sekunde über einen CSMA Kanal.
- Die Größe der Nachrichten kann verschieden sein. (Der Sensor besteht nicht daraus, sie ist nur eine Eigenschaft.)
- Dieser Sensor kann 10 Nachrichten pro Sekunde verschicken, die jeweils 48 Byte groß sind.
- Ein 32 kHz Clock, das für die Synchronisation verwendet wird und auf der äußeren Seite eines Sensor montiert wird

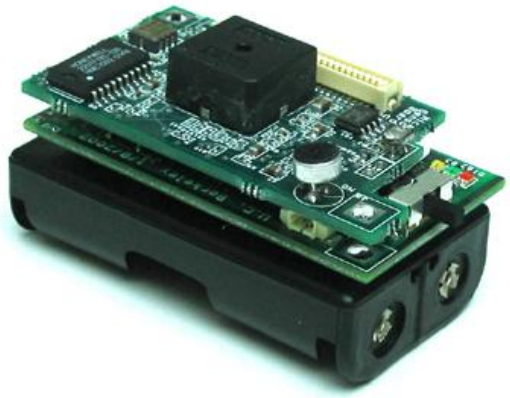

Abbildung 15. Ein *Mica Mote Sensor* [5]

Man kann die Sensoren über spezielle strukturierte Komponenten programmieren und der Code für die Sensoren kann durch spezielle Compiler übersetzt und per Kabel in den Programmspeicher der Sensoren installiert werden.

Es werden spezielle Betriebssysteme für die Sensoren verwendet wie TinyOS. Die Sensoren arbeiten in vier Phasen:

- Die *Snoozing* phase, in der der Sensor im Schlafmodus und nicht bereit für den Empfang von Nachrichten ist. Die Sensoren warten in dieser Phase bis sie ein Ereignis aufweckt.
- Wenn der Sensor aufgeweckt wird, dann übergeht er in die *Processing* Phase. In dieser Phase wird natürlich mehr Strom verbraucht und es findet Datengenerierung für die Daten, die von der Seneoren erfasst werden, in der lokalen Datenbank mithilfe von TinyDB statt (siehe 2.2 TinyDB).
- Die nächste Phase für den Sensor ist *Processing and Receiving*, in dem der Sensor die Daten bzw. Nachrichten von anderen benachbarten Sensoren über Funk empfängt.
- Die letzte Phase nennt man *Transmitting*, in der der Sensor seine Ergebnisdaten an Wurzel Sensor-Knote rekursiv schickt.

Folgende graphische Darstellung zeigt, wie sich der Stromverbrauch während der verschiedenen Phasen verhält.

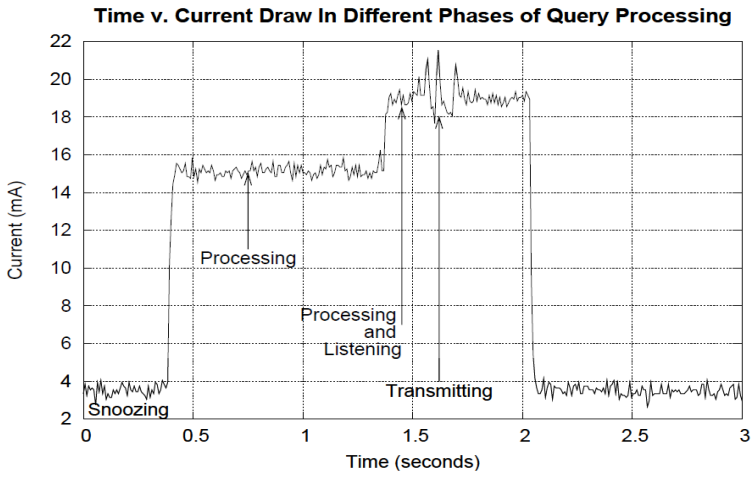

Abbildung 16. Verhältnis des Stromverbrauchs in den vier Sensor Arbeitsphasen eines Sensors [6]

3.1.2 Sensornetzwerke

Sensornetzwerke sind von der Architektur her wie verteilte Systemen aus Sensoren. Sensornetzwerke können aus mehreren hundert Sensorknoten bestehen, die ohne den Einfluss und die Steuerung eines Menschen Wochen oder Monate lang arbeiten.

Die einzige Interaktion, mit der der Mensch die Sensorenknoten beeinflusst, ist bei der Konfiguration der Netzwerkknoten, beim Wiederaufladen der Batterien oder bei der Datenanalyse aus den erfassten Sensordaten.

Sensorennetzwerke werden in mehreren verschiedenen Umgebungen eingesetzt, um Daten auf längere Zeit zu erfassen.Wegen der minimalen Größe der Sensoren und dem geringen Stromverbrauch eignen sich Sensorennetzwerke besonders für den Einsatz in vielen extremen Umgebungen, wo andere mobile Geräte, herkömmliche verteilte Systeme oder Menschen dafür nicht geeignet sind.

3.1.3 Kommunikation der Sensoren

Die Kommunikation zwischen den Sensoren erfolgt über Funkeinheiten oder Bluetooth, ist abhängig von den Umgebungseinflussfaktoren bzw. Ressourcen und hat eine Reichweite von ca. 30 Metern.

Über ein spezielles Betriebssystem, wie TinyOS, wird die Kommunikation bzw. Senden von Nachrichten eines Sensors zu einem anderen Sensor gesteuert. Nach dem Senden einer Nachricht von einem Sensor zu einem anderen Sensor, bekommt der Sender eine Benachrichtigung zurück, ob das Ziel bzw. der Knoten die Nachricht empfangen hat.

Auf jedem der Sensoren-Knoten befindet sich ein Datenbankabfrage-Generator, z.B. TinyDB, der sich für die beschränkten Ressourcen innerhalb der Sensorenknoten eignet, weil TinyDB die Basisaktionen wie normale Datenbanken kann und einen minimalen Ressourcenverbrauch hat. Das gesamte Sensorennetzwerk wird als eine Tabelle betrachtet, die Datenspalten von allen Sensorenknoten beinhaltet (siehe Abbildung 17).

Beim Senden von Nachrichten zwischen den Knoten werden dazu *Routing-Informationen* verschickt, die Aussagen über den Sender und seine genaue Position im Netzwerk im Bezug auf seine Nachbarknoten beinhalten.

Über diese Kommunikation, die wie ein Baum (*Routing Tree*) funktioniert (sie wird in einer Baumstruktur abgearbeitet), kann der Wurzel-Knoten eine Abfrage generieren und von allen Sensorknoten Daten sammeln. In diesem Verfahren läuft die Abfrage automatisch zu allen Sensorknoten im Netzwerk wiederholend bis es alle Knoten erreicht hat (siehe Abbildung 17). Und damit werden die Daten rekursiv bis zum Wurzel-Knoten gliefert.

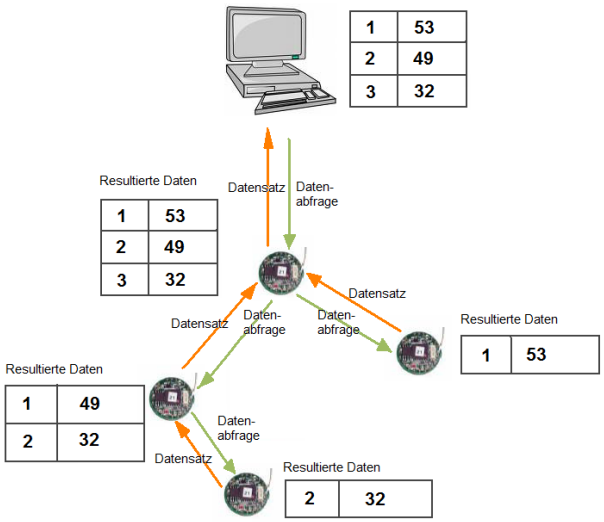

Abbildung 17. vgl. Bearbeitung der Anfrage an einem Sensornetzwerke [6]

3.2 TinyDB

In diesem Kapitel wird TinyDB als ein Datenbanksystem für Sensornetzwerke sowie die Grundstruktur und allgemeine Erläuterung der Anfragesprache von TinyDB Sensoren näher betrachtet. Wie vorher im Unterkapitel 3.1.3 „Kommunikation der Sensoren“ erklärt wurde, stellt das gesamte Netzwerk eine Datentabelle dar, und jeder Sensor liefert Datensatz, das eine Spalte dieser Tabelle darstellt.

TinyDB bietet Anfragen wie in anderen SQL-Datenbanken, die aus SELECT, FROM, WHERE bestehen und Selektion, Join und Aggregation unterstützen (siehe Abbildung 18).

```
ON EVENT <event>:
      SELECT <aggregates>, <attributes>
      [FROM {sensors | <buffer>}]
      [WHERE <predicates>]
      [GROUP BY <exprs>]
      [SAMPLE PERIOD <const> FOR <const> |
                ONCE |
                LIFETIME <const> ]
      [INTO <buffer>]
      [TRIGGER ACTION <command>]
```

Abbildung 18. TinyDB Anfrage [7]

TinyDB unterscheidet sich von normalem SQL in folgenden Punkten:
- In der optionalen Abfrage-Prädikat kann nur auf eine Tabelle Bezug genommen

werden.
- TinyDB unterstützt keine Subquerys.
- Boolesche Operatoren und String-Vergleiche werden nicht unterstützt.
- Die arithmetischen Ausdrücke sind auf die Operatoren +,-,*,/ begrenzt.
- TinyDB kann die Zeit für die Anfragen durch die Klausel *SAMPLE PERIOD* festlegen, die die normale SQL Sprache nicht kann.
- TinyDB hat ein besonderes Feature, und zwar die *TRIGGER ACTION* Klausel, die ein Trigger spezifiziert, der nach Erhalt eines Query-Resultats ausge führt wird.

Nehmen wir folgende Anfrage (siehe Abbildung 19) als Beispiel:

```
SELECT nodeid, light, temp
      FROM sensors
      SAMPLE INTERVAL 1s FOR 10s
```

Abbildung 19. TinyDB Anfrage [7]

Diese Anfrage besagt, dass jeder Sensor seine eigene ID, Licht und Temperaturmessungen jede Sekunde für die nächsten zehn Sekunden liefern soll. Die gelieferten Resultate dieser Anfrage fließen zum Anfrage-Erstellenden bzw. zum Haupt-Knoten (Wurzel) zurück.

Eine weitere Interessante Anfrageart bei TinyDB, ist die ereignisbasierende Anfrage. In diesen Anfragen können zu bestimmten Zeitpunkten und Ereignissen Anfragen ausgeführt werden.

```
ON EVENT e(nodeid)
SELECT sensors.a1
FROM sensors, events
WHERE sensors.nodeid = events.nodeid
      AND events.type = e
      AND sensors.time-events.time <=k
      AND sensors.time > events.time
SAMPLE PERIOD d
```

Abbildung 20. TinyDB Anfrage [7]

In dem Beispiel Anfrage (siehe Abbildung 20) heißt das, wenn ein Ereignis „*e*“ in Sensor mit der ID = *nodeid* auftritt, dann führe die Anfragesequenz aus.

4. Zusammenfassung

In dieser Arbeit wurden die drei Themen, Datenbanken, Sensornetzwerke und Dateisysteme, grundliegend behandelt, indem jeweils über die wichtigen Aspekten eine Einführung gegeben wurde. Bei Datenbanken sind dies solche Themenpunkte wie die

Systemarchitektur einer Datenbank, die verschiedene Datenbankmodelle sowie die drei Eigenschaften, die für die optimale Arbeit eines Datenbanksystems sehr wichtig sind. Weiterhin ist die große Rolle der Datenintegrität, des Transaktionskonzepts und der ACID-Eigenschaften, die bei Transktionen vorausgesetzt werden, nicht zu vergessen. Man muss die wichtige Rolle eines Dateisystems im Rahmen eines Betriebssystems bedenken und wie ist damit die Dateien in unseren Informationssystemen gespeichert, gelesen und geschrieben werden, bedenken.

Ein Dateisystem kann mit verschiedenen Arten von Blockzuordnungsverfahren, wie statische Dateizuordnung, Dynamische Extent-Zuordnung, Dynamische Blockzuordnung oder dem Versetzungsverfahren, arbeiten. Dazu wird ext3 als Beispiel beschrieben.

Zuletzt wurden Sensoren sowie Sensorennetzwerke vorgestellt. Dabei besteht ein Sensor aus Hardwarekomponenten mit dem Zweck bei verschiedenen Umständen bzgl. Der Umgebung des Sensors, Daten zu erfassen. Ein Sensornetzwerke besteht aus mehreren miteinander kommunizierenden Sensoren, die auch Daten gemeinsam erfassen, die vom Anwender über Abfragen mithilfe eines Gateways, z.B. der Datenbank TinyDB, abgefragt werden können.

5. Abbildungsverzeichnis

Literaturverzeichnis

1. Mitschang B: Grundlagen von Datenbanken und Informationssystemen

2. Theo Härder, Erhard Rahm, Datenbanksysteme, Konzepte und Techniken der Implementierung

3. A.Kemper, A.Eickler, Datenbanksysteme, Eine Einführung

4. Remy Card: Design and Implementation of the Second Extended Filesystem http://web.mit.edu/tytso/www/linux/ext2intro.html

5. IBR GruppeVS (Prof. Fischer): http://www.ibr.cs.tu-bs.de/theses/cbuschma/cfb-mica-zip.html

6. Samuel Madden, Michael J. Franklin, and Joseph M. Hellerstein: The Design of an Acquisitional Query Processor For Sensor Networks

7. Martin Schnyder: TinyDB An Acquisitional Query Processing System for Sensor Networks

8. First Dutch International Symposium on Linux, http://web.mit.edu/tytso/www/linux/ext2intro.htm

9. pagina GmbH: XML-Datenbank http://www.pagina-online.de

10. Yong Yao, Johannes Gehrke: Query Processing for Sensor Networks