GIGASCHATTEN IT
Version 2004 (pre-version)

Bearbeiter der Neuauflage:
Anne Tributh
Christian Böhrer
Florian Ganz
Axel Albrecht
Johannes (JoE) Esser
Daniel Sattler
Robert Schweisthal
Manfred Gonschor
Gregor Bender

Respekt an die Autoren der ursprünglichen Fassung!

Gigaschatten IT 2004 (pre-version)

1. Hardware 1
1.1 Hardware-Einteilung der Rechner 1
1.1.1 Klassische Einteilung (nach Leistung und möglicher Zahl der Benutzer): 1
1.1.2 Übliche Einteilung heute – mit Beispielen: 1
1.2 Von-Neumann-Architektur 1
1.3 Fundamentalzyklus der Befehlsverarbeitung 2
1.4 Ablauf eines Programms, Pipeline-Prinzip 2
1.5 RISC <-> CISC 3
1.5.1 CISC (Complex Instruction Set Computer) 3
1.5.2 RISC (Reduced Instruction Set Computer) 3
1.6 1.4 Rechnerarchitekturen 3
1.7 Parallele Prozessoren (symmetrisch <-> asymmetrisch) 3
1.8 Speicher 4
1.8.1 RAM: (Random access memory) 4
1.8.2 ROM: (Read only memory) 4
1.8.3 1.6.3 Datenträger und externe Speicher 5
1.9 Fehlertolerante Systeme 8
1.9.1 RAID 9
1.9.2 NAS und SAN 9
1.10 Synchronisation 10
2. BETRIEBSSYSTEME 10
2.1 Definition 10
2.2 Hauptfunktionen 10
2.3 Aufgaben 11
2.4 Klassifizierung 11
2.5 Betriebsarten 11
2.5.1 Dialogbetrieb 11
2.5.2 Realzeitbetrieb 12
2.5.3 Batchbetrieb 12
2.6 Portabilität: 13
2.7 Interoperabilität: 13
3. DATEN 13
3.1 Darstellungsmöglichkeiten von Zahlen 13
3.1.1 Zahlensysteme 13
3.1.2 Festkommazahlen 14
3.1.3 Gleitkommazahlen 14
3.1.4 Standart-Codierungen 14
3.2 Sortierverfahren 15
3.2.1 Shell sort 15
3.2.2 Quicksort 15
3.2.3 Heap Sort 15
3.2.4 Bubble Sort 15
3.2.5 Selection Sort 16
3.3 Suchverfahren 17
3.3.1 Binäres Suchen 17
3.3.2 Sequentielle Suche 17
3.4 Fehlererkennung bei Übertragung von Daten (Parity-Bit) 17
3.5 Datenschutz - Datensicherheit 18
3.5.1 Datenschutz 18

3.5.2 Datensicherheit ... 19
3.5.3 Überblick ... 22
3.6 Dateiorganisation ... 22
3.6.1 Zugriffsarten (Dateiorganisationsverfahren) ... 22
3.6.2 Verarbeitungs- und Zugriffsformen ... 24
3.6.3 Eigenschaften der Datei ... 24
3.6.4 Aufbau des Ordnungsbegriffs ... 25
3.7 Kryptographie ... 26
3.7.1 symmetrische Verschlüsselung ... 27
3.7.2 asymmetrische Verschlüsselung ... 27
3.7.3 Digitale Unterschrift ... 28
4. DATENBANKEN / DBMS ... 29
4.1 Grundlagen ... 29
4.1.1 Datenmodelle ... 29
4.1.2 Definition Datenbank ... 29
4.1.3 Definition Datenbankmanagementsystem ... 30
4.1.4 ANSI-SPARC-Dreischichtenmodell ... 31
4.2 Konzeptionelle Datenmodelle ... 31
4.2.1 Hierarchisches Datenmodell ... 32
4.2.2 Netzwerkdatenmodell ... 33
4.2.3 Relationales Datenmodell ... 34
4.2.4 OODM ... 34
4.3 Schlüssel ... 34
4.3.1 Definition ... 34
4.3.2 Schlüsselarten ... 35
4.4 Relationales Datenmodell ... 35
4.4.1 Grundregeln zur Darstellung von Relationen in Tabellen: ... 36
4.4.2 Normalisierung ... 36
4.5 Entity-Relationship-Modell ... 39
4.5.1 Generalisierung ... 41
4.5.2 Aggregation ... 42
4.6 SQL ... 42
4.6.1 Datenbankdefinition (DDL – Data Definition Language) ... 43
4.6.2 Datenmanipulation (DML – Data Manipulation Language) ... 44
4.6.3 Zugriffskontrolle und Zugriffsrechte (DCL – Data Control Language) ... 46
4.7 Transaktion ... 46
4.8 Transaktionssynchronisation ... 47
4.9 Eigenschaften von Transaktionen (ACID) ... 47
4.10 Recovery ... 48
4.10.1 Recovery-Maßnahmen ... 49
4.10.2 Rekonstruktion ... 49
4.11 Sperrprotokolle ... 49
4.11.1 Zwei-Phasen-Sperrprotokoll ... 49
4.12 ODBC ... 51
4.12.1 ODBC-Schnittstelle ... 51
4.12.2 Aufbau von ODBC ... 52
4.13 Replikation ... 53
4.14 Synchrone Replikation ... 54
4.14.1 Asynchrone Replikation ... 54
4.15 Data Warehouse – Konzept / OLAP ... 55

4.15.1 Hintergrund 55
4.15.2 Was ist ein Data Warehouse? 55
4.15.3 *Mehrdimensionale Datenmodelle* 56
4.15.4 Star – Modell -> relationaler Ansatz zur Umsetzung von Mehrdimensionalität 56
4.15.5 Elemente von Data Warehouses 56
4.15.6 Informationsdatenbank (Data Dictionary / Repository) 56
4.15.7 Warehouse – Hierarchien: Data Marts 57
4.15.8 FASMI – Definition für Data Warehouses / OLAP-Datenbanken 58
5. PROGRAMMIERSPRACHEN 58
5.1 Generationen von Programmiersprachen 58
5.1.1 Maschinensprachen (1GL) 58
5.2 Assemblersprachen (2GL) 58
5.2.1 Prozedurale Programmiersprachen 59
5.2.2 Nichtprozedurale Programmiersprachen (4GL) 59
5.2.3 Programmiersprachen der künstlichen Intelligenz (5GL) 59
5.2.4 Schematische Darstellung 60
5.3 Unterteilung nach Programmierparadigmen 60
5.3.1 Imperative Sprachen 60
5.3.2 Funktionale oder applikative Sprachen 60
5.3.3 Prädikative Sprachen 60
5.3.4 deskriptive Sprachen 60
5.3.5 Objektorientierte Sprachen 61
5.4 Compiler vs. Interpreter 61
5.4.1 Klassifikation 61
5.4.2 Compiler (Übersetzung erfolgt einmalig) 62
5.4.3 Interpreter (Übersetzung erfolgt schrittweise zur Laufzeit) 62
5.4.4 Assembler 63
5.4.5 Linker 63
5.4.6 Loader 63
5.4.7 Einsatz 63
5.5 Stufen des Compilers: 63
5.6 Call by reference / call by value 64
5.6.1 Call by Value 64
5.6.2 Call by Reference 64
5.7 Objektorientierung 65
5.7.1 Grundgedanke 65
5.7.2 Definition Objekt: 65
5.8 Funktionen des Linker 67
5.8.1 Allgemeinen 67
5.9 JAVA 67
5.9.1 Was ist JAVA ? 67
5.9.2 Was kann man mit JAVA machen ? 68
5.9.3 Woraus besteht ein Java-Programm? 68
5.9.4 Eigenschaften von Java? 68
6. Netzwerke 69
6.1 Kanalprinzip & Bussystem 69
6.1.1 Kanal-Konzept 69
6.1.2 Bus-Konzept 70
6.2 Netzwerktopologien 71

6.2.1 Bustopologie ... 71
6.2.2 Ringtopologie ... 71
6.2.3 Baumtopologie ... 72
6.2.4 Sterntopologie ... 72
6.2.5 Maschennetze ... 73
6.3 Zugriffsverfahren / Kanalzuteilung ... 73
6.3.1 Stochastische/konkurrierendes Verfahren ... 73
Deterministische Verfahren ... 74
6.3.2 ... 74
6.3.3 Drahtlose Netze ... 75
6.4 ISO/OSI Protokolle; TCP/IP ... 75
6.4.1 Protokolle ... 75
6.4.2 TCP/IP ... 80
6.4.3 OSI- Referenzmodell und TCP/IP im Vergleich ... 80
6.5 Client- Server- Architektur ... 81
6.6 Serverarten ... 81
6.6.1 File-Server ... 81
6.6.2 Print-Server ... 81
6.6.3 Fax-Server ... 82
6.7 Internet/Intranet ... 82
6.7.1 WWW (World Wide Web) ... 82
6.7.2 FTP ... 83
6.7.3 E-mail ... 83
6.7.4 TELNET ... 83
6.7.5 Archie ... 83
6.7.6 Gopher ... 83
6.7.7 WAIS ... 83
7. Systementwicklung / Projektmanagment ... 84
7.1 Prinzipien, Methoden, Verfahren/Techniken, Werkzeuge/Tools ... 84
7.1.1 Allgemein Software - Engineering/ Systementwicklung (SE) ... 85
7.1.2 Funktionalität ... 86
7.2 Phasenkonzept ... 86
7.2.1 Initialisierungsphase ... 86
7.2.2 Analysephase ... 86
7.2.3 Entwurfsphase ... 87
7.2.4 Realisierungsphase ... 87
7.3 Methoden der Ist-Analyse ... 87
7.4 Strukturierte Analyse / Systementwurf ... 89
7.5 Prinzipien der Softwareentwicklung ... 91
7.6 Testverfahren für Software ... 92
7.6.1 Der Lebenszyklus einer Programms ... 92
7.6.2 Fehler in der Software ... 92
7.6.3 Testen von Programmen ... 92
7.6.4 Vorgehensweise beim Testen ... 93
7.6.5 Testtechniken ... 93
7.6.6 Stufen des Software-Tests ... 93
7.7 Qualitätsmerkmale für Software ... 94
7.7.1 aus Benutzersicht ... 94
7.7.2 aus Entwicklersicht ... 94
7.7.3 nach DIN 66272 ... 95

7.8 SW-Ergonomie ... 95
7.9 Portabilität / Interoperabilität ... 97
7.10 ISO 9000 ... 97
7.11 Standardsoftware: ... 99
7.11.1 Vorteile von Standartsoftware ... 99
7.11.2 Nachteile ... 100
8. SONSTIGES ... 100
8.1 Wirtschaftsinformatik ... 100
8.2 Stammdaten-Bewegungsdaten ... 100
8.2.1 Stammdaten ... 100
8.2.2 Bewegungsdaten ... 101
8.3 EDI/EDIFACT ... 102
8.3.1 Was ist EDI ? ... 102
8.3.2 EDIFACT ... 105
8.3.3 Zukunft von EDI ... 106
8.4 ActiveX / DCOM ... 107
8.5 Dokumenten-Management-Systeme ... 108
8.5.1 Was ist ein Dokumenten-Management-System (DMS)? ... 108
8.5.2 Sinn und Zweck von Dokumenten-Management-Systeme ... 108
8.5.3 Funktionsweise von DMS ... 108

1. Hardware

1.1 Hardware-Einteilung der Rechner

QUELLE: Hansen s.57f, 688ff

1.1.1 Klassische Einteilung (nach Leistung und möglicher Zahl der Benutzer):

Personalcomputer
Workstation
Minirechner
Großrechner
Superrechner

1.1.2 Übliche Einteilung heute – mit Beispielen:

Persönliche Informationshilfsmittel – PDA, Handy, etc.
Personalcomputer – auch Notebooks, Subnotebooks
Workstations – größere Leistung, hochauflösender Bildschirm, etc.
Server – Hohe Leistung, bedient i.d.R. mehrere User; Abteilungs-/Arbeitsgruppen-/Unternehmensserver
Hochleistungsserver (Superrechner) – Für spezielle, sehr rechenintensive Aufgaben; Wetterzentrum, oder bekanntes Beispiel: Deep Blue als Schach-Supercomputer

1.2 Von-Neumann-Architektur

QUELLE: Hansen, s. 652; Engel-Folien 1.Semester (Rechnerarchitektur)
Im Jahr 1946 von John von Neumann vorgestelltes Konzept zur Gestaltung eines universellen Rechners, der technischen, wissenschaftlichen und kommerziellen Anforderungen gerecht wird. Von wenigen Ausnahmen abgesehen, sind die heutigen Rechenanlagen Weiterentwicklungen dieses klassischen Universalrechners.

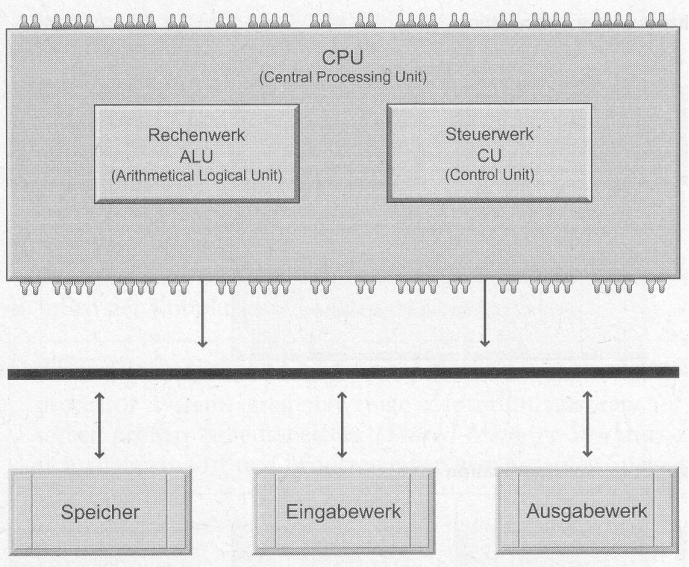


Abb. 7.1.1/1: Grobstruktur einer von-Neumann-Architektur

Eine von-Neumann-Maschine besteht aus fünf Funktionseinheiten: Steuerwerk, Rechenwerk, Speicher (Zentral- und externer Speicher), Eingabewerk und Ausgabewerk.
Die Arbeitsweise folgt dem Prinzip Eingabe, Verarbeitung und Ausgabe (EVA-Prinzip der Datenverarbeitung). Die Zentraleinheit (Steuerwerk und Rechenwerk) interpretiert nacheinander die von außen eingegebenen Befehle und führt sie aus, die Kommunikation der einzelnen Bestandteile erfolgt über den Systembus.
Charakteristika der Von-Neumann-Architektur:
Die Struktur des Rechners ist von der zu bearbeitenden Aufgabenstellung unabhängig
Daten und Befehle sind in einem gemeinsamen Speicher abgelegt
Der Rechner ist mit einem Prozessor ausgestattet, welcher Schritt für Schritt Daten und Befehle abarbeitet

1.3 Fundamentalzyklus der Befehlsverarbeitung

QUELLE: Hansen, s.664, evtl. Preiser-Folien ?
Das Leitwerk (Steuerwerk) übernimmt als Bestandteil der CPU die Koordinationsfunktion für den gesamten Rechner. Es beinhaltet den Befehlszähler, das Befehlsregister und das Statusregister. Zu Beginn der Programmverarbeitung wird der Befehlszähler mit der Adresse des ersten Befehls geladen. Anschließend wird das Programm über folgenden Zyklus abgearbeitet:
1. Befehl aus Hauptspeicher (Arbeits- oder Festwertspeicher) holen;
2. Befehl decodieren, Operanden-Adressen oder Sprungziel errechnen;
3. Operanden holen;
4. Befehl ausführen, d.h. Operanden verarbeiten, ggf. Ergebnis speichern
5. Befehlszähler erhöhen

Das Befehlsregister enthält stets den Befehl, der im Moment ausgeführt wird. Bei Sprungoperationen (z.B. in Schleifen) wird der Befehlszähler nicht auf den nächsten Befehl erhöht, sondern mit der Zieladresse des Sprungbefehls geladen.

1.4 Ablauf eines Programms, Pipeline-Prinzip

QUELLE: Hansen, s.53, 666
Im Fundamentalzyklus übernimmt das Steuerwerk die Phasen 1 bis 3 (Befehlsaufbereitung) und 5, die Phase 4 (Befehlsausführung) übernimmt das Rechenwerk.
Das Pipeline-Prinzip hat zwei Aspekte:
Ein Befehl wird nacheinander (wie in einer Pipeline) zunächst vom Steuerwerk und anschließend vom Rechenwerk abgearbeitet
Während das Rechenwerk einen Befehl ausführt, bereitet das Steuerwerk zeitlich parallel dazu („überlappt“) schon die nächsten Befehle auf. Da auch innerhalb derjenigen Phasen, die dem Steuerwerk übertragen sind, durch Zerlegung der Befehle überlappt gearbeitet wird, ist das Steuerwerk dem Rechenwerk, je nach Rechnertyp, um einen oder mehrere Befehle voraus.
Zur Erhöhung des Durchsatzes erfolgt in modernen Rechnern die Befehlsverarbeitung überlappend in mehreren Pipelines, deren Effizienz jedoch von den Abhängigkeiten der Befehle bestimmt wird.

1.5 RISC <-> CISC

QUELLE: Hansen, s.48, 681ff
Als Befehlsvorrat wird die Anzahl der im Maschinencode verfügbaren Befehle eines Prozessors verstanden, die in der Regel im Festwertspeicher (ROM) dem Rechenwerk (ALU = Aritmethical and Logical Unit) fest zugeordnet sind. Sie beträgt üblicherweise 150 bis 300 Instruktionen.

1.5.1 CISC (Complex Instruction Set Computer)

Beim CISC-Prinzip wird der Befehlsvorrat vergrößert, z.B. durch die Aufnahme von Sprachelementen höherer Programmiersprachen oder von Programmen für komplette Anwendungen (auch ASIC = Application Specific Integrated Circuit). CISC-Architekturen sind besonders für technische Anwendungen geeignet.

1.5.2 RISC (Reduced Instruction Set Computer)

QUELLE in GS angegeben: Aus Stahlknecht, 1997, S. 34 f.
Das RISC-Prinzip geht von Messungen aus, wonach bei vielen Programmabläufen in 80 % der CPU-Zeit nur 20 % des Befehlsvorrates in Form einfacher Lade-, Speicher- und Verzweigungsoperationen genutzt werden. RISC-Prozessoren haben deswegen einen geringeren Befehlsvorrat (50 bis 100, teilweise weniger als 40 Grundbefehle). Dabei handelt es sich um einfache Befehle mit fester (und wegen der geringeren Befehlsanzahl kleinerer) Länge und einheitlichem Format. Während CISC-Prozessoren in der Regel pro Befehl mehrere Takte benötigen, führen RISC-Prozessoren einen oder mehrere Befehle in einem einzigen Takt aus. Im Bereich der Mikrocomputer sind in erster Linie die Arbeitsplatzrechner mit RISC-Prozessoren ausgestattet.

1.6 Rechnerarchitekturen

QUELLE: Hansen, s.82ff, 653ff
SISD (Single Instruction Single Data) Rechner können in einem Bearbeitungsschritt jeweils einen Befehl mit den zugehörigen Operanden ausführen. Bsp.: Standard-PC
SIMD (Single Instruction Multiple Data) Rechner bestehen aus einer Menge von Zentralprozessoren, die jeweils den gleichen Befehl auf einer Vielzahl von Operanden gleichzeitig anwenden. Solche speziellen Vektorrechner ermöglichen zum Beispiel sehr effiziente Berechnungen mit Vektoren und Matrizen.
MISD (Multiple Instruction Single Data) Rechner könnten mehrere Operationen gleichzeitig auf denselben Daten durchführen. Rein konzeptuelle Kategorie!
MIMD (Multiple Instruction Multiple Data) Rechner sind Mehrprozessorsysteme, deren Prozessoren gleichzeitig mehrere Befehlsströme auf verschiedene Daten ausführen. Sie werden weiter unterteilt in eng gekoppelte (Prozessoren greifen auf gemeinsamen Arbeitsspeicher zu) und lose gekoppelte (jeder Prozessor hat seinen eigenen Arbeitsspeicher) Mehrprozessorsysteme. Bsp.: Workstation-Cluster.

1.7 Parallele Prozessoren (symmetrisch <-> asymmetrisch)

QUELLE: Hansen, s. 654, 921

Unter Parallelverarbeitung versteht man generell entweder die simultane Ausführung mehrerer Befehle oder Befehlsteile und / oder die simultane Bearbeitung mehrerer Daten.
Diese Parallelverarbeitung kann durch mehrere, miteinander gekoppelte parallele Prozessoren erreicht werden, sog. Multiprozessor-Systeme.
Bei symmetrischen Systemen verteilt das Betriebssystem die einzelnen Aufträge gleichmäßig auf sämtliche Prozessoren.
Bei asymmetrischen Systemen werden einzelne Prozessoren vom Betriebssystem für die exklusive Abarbeitung einzelner Aufträge reserviert (z.B. für Datenbanksoftware).

1.8 Speicher

1.8.1 RAM: (Random access memory)

QUELE: Hansen, s. 667
Das RAM ist ein Schreib-Lese-Speicher, bei dem jede Speicherzelle einzeln adressierbar und inhaltlich veränderbar ist (wahlfreier Zugriff). RAMs werden als Daten- und als Programmspeicher eingesetzt. RAMs, die durch Halbleiter realisiert sind, verlieren bei Stromausfall ihren Inhalt. Um dem vorzubeugen, rüstet man Computer und auch Taschenrechner mit Schaltungen aus, die bei Stromausfall oder versehentlichem Ausschalten die RAMs über Akkumulatoren oder Batterien mit Strom versorgen, so dass ihr Speicherinhalt erhalten bleibt.

1.8.1.1 CACHE

QUELLE: Hansen, s.54, 659ff
Bezeichnung für die sehr schnellen, kleinen (Größenordnung 20KB) und teuren RAM-Speicher, welche als Pufferspeicher in CPUs eingesetzt werden. Sie sind auf die Geschwindigkeit des Prozessors abgestimmt. Eine ausgeklügelte Steuerungslogik sorgt dafür, dass ein hoher Prozentsatz der Speicherzugriffe der CPU direkt über den Cache abgewickelt werden können, statt auf den langsameren Arbeitsspeicher zugreifen zu müssen. Im Cache werden während der Programmverarbeitung die jeweils aktuellen Befehle und Daten rechtzeitig bereitgestellt.
Der Second Level Cache ist ein zusätzlicher Speicher zwischen (Level-1-) Cache und Arbeitsspeicher. Er ist langsamer, dafür aber größer und kostengünstiger.
Cache kann nicht mit allgemeinem Pufferspeicher gleichgesetzt werden, welcher nur zum zeitlichen Ausgleich zwischen Funktionseinheiten unterschiedlicher Geschwindigkeiten dient.

1.8.2 ROM: (Read only memory)

Festwertspeicher; Speicher, dessen Inhalt bereits bei der Herstellung festgelegt wird und nicht mehr verändert werden kann. ROMs werden vorwiegend als Speicher für feste Programme (z. B. für Betriebssystemkomponenten) und für unveränderbare Daten verwendet. Bei speziellen
Anfertigungen des Festwertspeichers lassen sich die Inhalte wieder löschen:
EPROM (Erasable Programmable Read Only Memory) durch Bestrahlung mit UV-Licht EEPROM (Electrically Erasable Programmable Read Only Memory) durch Stromstöße: Z.B. bei Chipkarten und Flash-Speichern (BIOS, etc.)

1.8.3 Datenträger und externe Speicher

QUELLE: generell Hansen, s. 702ff

1.8.3.1 sequentielle und Direktzugriffsspeicher

QUELLE: KEINE !

Bei sequentiellen Speichern (z.B. Magnetbandspulen, -kassetten) ist im Gegensatz zu den Direktzugriffsspeichern nur ein fortlaufender Zugriff, d.h. ein Zugriff in der gespeicherten Reihenfolge.

Bei Direktzugriffsspeichern besitzen die einzelnen Speicherplätze, -bereiche oder -zellen eine individuelle Adresse, über die sie jeweils direkt angesprochen werden können (schnellerer Zugriff, höhere Verarbeitungsgeschwindigkeit).

1.8.3.2 Aufgaben der externen Speicher:

QUELLE: KEINE !

Zwischenspeicherung der Daten (zwischen Eingabe und der Verarbeitung, bei einer längeren Verarbeitung, vor der endgültigen Ausgabe)
Sammlung von Daten bei einem Datensammelsystem (Mehrplatzsystem)
Bereithaltung von Daten und Programmen
Archivierung von Daten bzw. Dateien
Datenträgeraustausch
Sicherung von Daten und Programmen
Speicherung von Daten und Programmen (zum Verkauf derselben)

1.8.3.3 Speicherungstechniken und Arten von Datenträgern

1.8.3.3.1 Lesbare Schrift bzw. Klarschrift

QUELLE: Hansen, 714

Maschinell lesbare Schrift z.B. OCR A und OCR B oder EUROBANK A und B heute auch
Handschriften (Blockschrift mit bestimmten Grundeigenschaften)

1.8.3.3.2 Strichcodes

QUELLE: Hansen, 709

z.B. EAN-Code (europäische Artikelnummer)

1.8.3.3.3 Markierungen

QUELLE: Hansen, 707

Striche, Kreuze etc. in dafür vorgesehenen, besonderen Feldern (Markt- und Meinungsforschung, Lotto)

1.8.3.3.4 Lochungen

QUELLE: KEINE !

Löcher in Lochkarten oder Lochstreifen (60er Jahre)

1.8.3.3.5 Magnetisierung

QUELLE: Hansen, 718ff

Magnetisierung einer geeigneten Trägerschicht (Ferritoxyd, Chromdioxyd, Reineisen) auf einem Basismaterial (Kunststoff, Aluminium). Die Speicherung erfolgt über die Richtung des Magnetisierungszustandes der Bitpositionen.

Beispiele:
Magnetband: (Ähnlich einer Musikkassette)
Magnetplatte: (z.B. Festplatte, oft mehrere, auf einer Achse übereinander angebrachte Platten)
Diskette
Magnetstreifenkarte (z.B. Euroscheck-Karte oder Kreditkarte)

1.8.3.3.6 Optische Speicherung

QUELLE: Hansen, 768ff
Hierbei werden Daten bzw. Informationen als Bitfolgen in Form von Löchern, Vertiefungen, Strukturveränderungen oder auf andere, durch Laser bewirkte Art auf Trägern gespeichert
(z.B. CD, DVD)

1.8.3.3.7 Mikrofilm

QUELLE: Hansen, 768
Zur Speicherung von Daten, die für das menschliche Auge wahrnehmbar sein sollen (durch Vergrößerungs- bzw. Lesegeräte)

1.8.3.3.8 Elektronische Speicherung

QUELLE: Hansen, 795ff
Speicherung auf Halbleiterelementen (wie in den Prozessoren)
Chipkarten (mit eingebautem Chip mit Mikroprozessor und Speicher)
Halbleiterplatten (aus mehreren Speicherchips; benötigt permanente Stromzufuhr)
Transponder

1.8.3.3.9 Magnetstreifenkarten

QUELLE: Hansen, s. 718ff
Plastikkarte mit standardisierter Größe mit integriertem Magnetstreifen. Auf diesen können in drei parallelen, unabhängigen Spuren Daten aufgezeichnet und gelesen werden. Die Kapazität beträgt in der Regel 1.394 Bit. Anwendung finden die Karten vor allem als Bank- und Identifikationskarten.
Vorteile: Klein und transportabel; weitgehend sichere, kostengünstige Identifikation und Berechtigungsprüfung des Inhabers; Kombination von visuellen und maschinenlesbaren Informationen
Nachteile: Hohe Kosten für Präge- und Codiermaschinen sowie Kartenleser; geringe Datenkapazität; betrügerische Manipulation durch Kopieren oder Fälschen möglich

1.8.3.3.10 Magnetbänder

QUELLE: Hansen, s. 721ff
Dünnes Polyesterband, einseitig magnetisierbar zur Datenaufzeichnung und zum Schutz in Kunststoffgehäuse verbaut. Einsatz hauptsächlich zur Datensicherung und Langzeitarchivierung von großen Informationsbeständen. Bekannte Standards sind QIC, DAT, DLT und LTO. Die Kapazitäten bewegen sich heute in Bereichen bis über 100GB, die Datenübertragungsrate bis über 24 MB/s. Allerdings werden die Daten sequentiell aufgezeichnet und gelesen, was lange Zugriffszeiten zur Folge hat. Neben Einzellaufwerken gibt es sog.

Autoloader und Bibliotheken, welche -teils mit mehreren Laufwerken- den Bandwechsel automatisieren und letztere Gesamtkapazitäten im zweistelligen Terabyte-Bereich bieten.
Vorteile: billige Medien; hohe Schreib-/Leseleistung; hohe Haltbarkeit; große Kapazitäten; transportabel
Nachteile: Lange Zugriffszeiten; Empfindlichkeit gegen Umwelteinflüsse

1.8.3.3.11 Disketten

QUELLE: Hansen, s. 739ff
Disketten sind dünne, runde Magnetscheiben, die sich fest in einer Schutzhülle aus Plastik befinden. Es gibt 8-Zoll-, 5,25-, 3,5- und 2-Zoll-Disketten. Man unterscheidet nach ein- und doppelseitiger Beschreibbarkeit (SS = Single Sided, DS = Double Sided) und zwischen verschiedenen Schreibdichten (SD = Single Double, DD = Double Density, HD = High Density, QD = Quadruple Density). Die Spur- und Sektoreinteilung übernimmt hier das Betriebssystem. Die Kapazität liegt bei 1,44MB für Standarddisketten und bis 250MB für Zip und HiFD Disketten, welche aber recht schnell durch die Möglichkeit des CD-Brennens vom Markt verdrängt wurden.
Vorteile: Versendbarkeit; einfache Handhabung; Austauschbarkeit
Nachteile: heute zu geringe Kapazität; langsam; Empfindlich gegen Umwelteinflüsse

1.8.3.3.12 Magnetplatten

QUELLE: Hansen, s. 747ff
Die Magnetplatte ist eine runde Scheibe Aluminium, Magnesium oder Glassubstrat, bei der die Daten beidseitig auf magnetisierbaren Schichten durch Magnetisierung aufgebracht werden. Im Normalfall sind mehrere Scheiben auf einer rotierenden Achse fest übereinander zu einem Plattenstapel verbunden. Jede Scheibe ist in Spuren (Kreise) und Sektoren aufgeteilt.
(Stahlknecht Abb. 2.19, 2.20 (8. Auflage - S. 64/65)) Der Sektor ist die kleinste adressierbare Einheit auf einer Scheibe und wird immer als Ganzes in den Arbeitsspeicher gelesen oder beschrieben. Die Gesamtheit aller auf den Plattenoberflächen direkt übereinander liegenden Spuren bezeichnet man als Zylinder. Die Kapazität eines Plattenstapels errechnet sich nach der Formel Byte pro Sektor * Sektoren pro Spur * Spuren pro Zylinder * Zylinder pro Stapel. Auf jeder Spur befindet sich die gleiche Anzahl an Byte. Dadurch ist die Schreibdichte in den inneren Spuren höher.
Heute gebräuchliche Festplatten haben standardisierte Gehäusegrößen von 3,5 Zoll (2,5 Zoll speziell für Notebooks) und Kapazitäten zwischen 40 und 300 Gigabyte. Datentransferraten liegen inzwischen über 100 MB/s bis hin zu 320 MB/s bei Ultra320 SCSI Laufwerken.
Vorteile: Hohe Kapazitäten; Hohe Datenübertragungsraten und kurze Zugriffszeiten (im Millisekundenbereich); günstiger und ständig sinkender Preis pro MB
Nachteile: Erschütterungsempfindlich; bedingt zum Transport geeignet

1.8.3.3.13 Mikrofilm

QUELLE: Hansen, s. 768ff
Datenträger aus Filmmaterial, auf dem mittels fotografischer Verfahren schriftliche und bildliche Information stark verkleinert aufgezeichnet wird. Die

analog gespeicherte Information ist visuell lesbar, wozu jedoch eine optische Vergrößerung (spezielles Lesegerät) nötig ist. Haupteinsatzgebiet sind Bankzentralen, die große Belegmengen auf Mikrofilm archivieren.
Vorteile: Kostengünstig bei großen zu archivierenden Datenmengen; hohe Zeichendichte
Nachteile: umständliche Handhabung; lange Zugriffszeit; nur einmal beschreibbar

1.8.3.3.14 Optische Speicherplatten

QUELLE: Hansen, s. 770ff
Feste, runde Kunststoffscheiben, bei denen lasergenerierte Lichtenergie direkt zum Lesen und Schreiben von Informationen auf optisch reaktivem Material verwendet wird oder aber das Lesen und Schreiben unterstützt. Das Auslesen der Daten erfolgt über die Auswertung der Veränderungen des reflektierten Licht.
Standards: CD-ROM; CD-R; CD-RW; DVD-ROM; DVD-R; DVD-RW; DVD-RAM; DVD+R; DVD+RW

1.8.3.3.15 Transponder

QUELLE: Hansen, s. 803ff
Ein Transponder ist ein automatischer Antwortsender, der auf ein eingehendes Signal reagiert. Aktive Transponder beziehen ihre Energie aus einer Batterie, passive werden über ein elektromagnetisches Feld von der Lese-/Schreibstation gespeist. Die Reichweite passiver Transponder ist bis auf wenige Meter begrenzt, die von aktiven Transpondern erreicht je nach Leistung bis über 100 Meter.
Großes Zukunftspotential wird derzeit der Anwendung von RFID (Radio frequence identification) zugesprochen, welche mit passiven Transpondern arbeitet. Denkbar sind beispielsweise „kassenlose“ Supermärkte oder die permanente Standort-Überwachung von Sendungen in Logistikbetrieben.

1.8.3.3.16 Flash-Speicherkarten

QUELLE: Hansen, s. 807ff
Flash-Speicher sind elektrisch wiederbeschreibbare Speicher, die ihren Inhalt ohne Energiezufuhr bewahren. Die Kapazität erreicht etwa bis 2 GB, Größen und Standards variieren Herstellerabhängig: PCMCIA-Format, CompactFlash, SmartMedia, Multimediacard, Memory-Stick
Die Anwendungsgebiete liegen vornehmlich in mobilen Endgeräten, wie PDAs, Handys, MP3 Player, Digitalkameras, etc.

1.9 Fehlertolerante Systeme

QUELLE: Hansen, diverse Kapitel
Fehlertolerante Systeme werden durch Hardware-Redundanzen realisiert. Unter Fehlertoleranz versteht man ganz allgemein die Fähigkeit einer DV-Anlage, trotz einer begrenzten Anzahl fehlerhafter Subsysteme (z.B. bei Ausfall von Prozessoren oder bei Speicherdefekten) die vorgegebene Funktion korrekt zu erfüllen.

In fehlertoleranten Systemen sind die wichtigsten Komponenten mindestens doppelt vorhanden (z.B. Platten mit „gespiegelten“ Dateien oder RAID-Laufwerke). Je nachdem, ob die zusätzlichen Funktionseinheiten ständig oder erst bei einem Ausfall aktiv sind, spricht man von statischer oder dynamischer Redundanz.
In Client/Server-Modellen kann Fehlertoleranz durch einen zusätzlichen Server erzielt werden.

Fehler können in einem Einzelsystem oder in einem verteilten Client – Server System durch eine beliebige Einzelkomponente oder durch schwer nachvollziehbare Kombinationen von Bedingungen entstehen.
Fehler können auf verschiedenen Ebenen innerhalb eines Systems entstehen (Hardware / Software)
Um ein Fehlertolerantes System zu entwickeln, müssen sowohl auf HW – Ebene als auch auf SW – Ebene (z.B. durch Fehlermanagement- und HelpDesk-Werkzeuge) Maßnahmen getroffen werden
Fehlertolerante Systeme werden heute hauptsächlich für zeitkritische Anwendungen im Bankensektor, sowie in großen internationalen Unternehmen eingesetzt, die 24 Stunden am Tag eBusiness betreiben wollen
Hierfür werden auf HW – Ebene Multiprozessor-Server oder bestimmte Serverstrukturen
(z.B. Cluster Server) eingesetzt

1.9.1 RAID

QUELLE: Hansen, s. 752ff
Die RAID (Redundant Array of Inexpensive Disks) strebt durch die Verwendung von vielen kleinen Standardplatten eine Senkung der Kosten bei gleichzeitiger Erhöhung der Datentransferrate und Ausfallsicherheit an.
RAID-0: Die Daten werden parallel in Blöcken auf die vorhandenen Plattenlaufwerke geschrieben (disk striping). Schnellste und effizienteste RAID Stufe, aber keine Fehlertoleranz. Bei Defekt einer Platte sind alle Daten des gesamten Plattenarrays verloren.
RAID-1: Arbeitet mit gespiegelten Platten (disk mirroring; shadowing). Die gleichen Daten werden gleichzeitig auf unterschiedliche Laufwerke geschrieben. Bei Defekt eines Laufwerks stehen die Daten in vollem Umfang auf dem Duplikat zur Verfügung.
RAID-3: Die Daten werden in Blöcken über mehrere Laufwerke hinweg gespeichert und zusätzlich Paritätsinformationen auf einem separaten Laufwerk. Bei Defekt eines Laufwerks, können die Daten anhand der Paritätsinformationen und der verbliebenen Daten auf den anderen Laufwerken wieder hergestellt werden.
RAID-5: Die gebräuchliche RAID Stufe für Fehlertoleranz. Daten und Prüfsummen werden quer über alle Laufwerke hinweg geschrieben, dies ermöglicht einen höheren Datendurchsatz.

1.9.2 NAS und SAN

QUELLE: Hansen, s. 755ff
NAS (Network Attached Storage) ist direkt an ein LAN angeschlossener Speicher. Eine NAS-Einheit ist ein mit geringem Aufwand zu installierendes und einfach zu verwaltendes Gerät, das aus einem oder mehreren internen Servern,

vorkonfigurierter Plattenkapazität und einem speziell auf Dateiverwaltung und Datenübertragung ausgelegten Betriebssystem besteht.
SAN (Storage Area Network) ist ein zentral verwaltetes, speziell auf den blockbasierten Datenaustausch zwischen Servern und Speichergeräten zugeschnittenes Speichernetz für heterogene Umgebungen. Die Speichergeräte sind im Gegensatz zu NAS von den Servern getrennt und über ein unabhängiges, vom LAN völlig getrenntes Netz zugänglich.

1.10 Synchronisation

Abstimmung nebenläufiger Vorgänge aufeinander. Die Synchronisation der Vorgänge kann durch gegenseitige Beobachtung erfolgen oder durch einen speziellen mit der Synchronisation beauftragten Überwacher. Im Mehrprogrammbetrieb einer Rechenanlage kommt der Synchronisation der zahlreichen nebenläufigen arbeitenden Programme und Prozesse eine hohe Bedeutung zu. Oft geschieht eine Synchronisation beim Austausch von Informationen. Man spricht von synchronen Schaltwerken, wenn alle Elemente der Schaltung mit einem einheitlichen Takt versorgt werden. Jede elementare Aktion benötigt in synchronen Schaltwerken die gleiche Zeit. In asynchronen Schaltwerken sind Geschwindigkeit und Arbeitsabläufe verschiedener Komponenten nicht aufeinander abgestimmt, insbesondere muss es keinen einheitlichen Takt geben und nicht jede Aktion unmittelbar nach Beendigung der vorhergehenden beginnen.

Beispiele:
Synchronisation durch gegenseitiges Beobachten (Reisverschlussverfahren bei Verkehrsengpässen)
Synchronisation per Überwacher (Spiel der Musiker eines Orchesters unter Leitung des
Dirigenten)

2. BETRIEBSSYSTEME

2.1 Definition

QUELLE: Hansen, s. 915
Als Betriebssystem bezeichnet man die Programme, welche (zusammen mit den Eigenschaften der Rechnerhardware) die grundlegende Infrastruktur für die Ausführung von Anwendungssoftware bilden. Das Betriebssystem bildet eine Abstraktion von Hardwareeigenschaften und ist für die Steuerung und Überwachung von Anwendungsprogrammen zuständig.

2.2 Hauptfunktionen

QUELLE: Hansen, s.916ff
Unterbrechungsverantwortung (interrupt handling)
Verteilung (Dispatching) Instrument des Time-Sharing
Betreibsmittelverwaltung (resource management)
Programmallokation (Speicherverwaltung)
Dateiverwaltung
Auftragssteuerung (job control)

Zuverlässigkeit (reliability)

2.3 Aufgaben

QUELLE: Hansen, s.916ff
Bereitstellung von Hilfsmitteln (Speicher etc.) zur Bearbeitung von Benutzerprogrammen
Verbergen von Hardwareeigenschaften und spezieller Softwarekomponenten
Koordination und Aufteilung der Betriebsmittel an mehrere, gleichzeitig arbeitende Benutzer

2.4 Klassifizierung

QUELLE: KEINE !
Monolithisch:
eine Sammlung von Prozeduren ohne Strukturen

Schichtenmodell:
Anwendungsprogramm & Betriebssystem (THE Betriebssystem)
Hier kann UNIX angesiedelt werden: Anwender – Shell – Standardprogramme – UNIX Kern – Hardware Schnittstelle – Hardware

Virtuelle Maschinen (Multiusing, Multitasking):
Prinzip des Time-Sharing -> mehrere VM auf der Hardware
Der Anwender glaubt, dass er den Rechner alleine für sich hat, aber er sieht nur einen Pseudo Prozess, der eine komplette Kopie des Systems beinhaltet. Je besser das System arbeitet, desto besser ist die Illusion und auch die Performance.

2.5 Betriebsarten

QUELLE: Hansen, s.916ff

2.5.1 Dialogbetrieb

QUELLE: Hansen, s.923ff
Eine große Anzahl von Benutzern ist direkt mit der Rechenanlage verbunden, gibt mehr oder weniger gleichzeitig in schneller Folge kurze Teilaufträge (Transaktionen) an den Rechner und erwartet deren zügige Bearbeitung. Die Rechenanlage verarbeitet die Teilaufträge mit gewissen Einschränkungen in der Reihenfolge ihres Eingangs. Die Form des Gesamtauftrags ist nicht von vornherein festgelegt, sondern kann vom Benutzer interaktiv, d.h. in unmittelbarer Reaktion auf Ergebnisse der Teilaufträge, ständig verändert werden. Unterschieden werden muss zwischen Teilhaber- und Teilnehmerbetrieb. Im Teilhaberbetrieb benutzen alle Teilhaber dasselbe Programm. Typische Einsatzbereiche für den Teilhaberbetrieb sind z.B. Zentralbuchungen bei Banken und Buchungen von Reisen in einem Reisebüroverbund. Im Teilnehmerbetrieb kann jeder Teilnehmer unabhängig von den anderen Teilnehmern Aufträge an die Rechenanlage senden. Typische Anwendungen des Dialogbetriebs sind Editierarbeiten. Viele Rechenanlagen erlauben gleichzeitig Dialog- und Stapelbetrieb. Das Ziel des Dialogbetriebs ist die Minimierung der Antwortzeit bei Kurzaufträgen.

2.5.1.1 Teilhaberbetrieb

QUELLE: Hansen, s.924ff
Im Teilhaberbetrieb haben alle Teilhaber dasselbe Programm. Es arbeiten mehrere Benutzer gleichzeitig im Dialogbetrieb an Datenstationen (Rechner oder Terminals) am selben Datenbestand. Die Aufgaben sind eng begrenzt und erfordern keine Programmier- und Systemkenntnisse. Jeder Benutzerauftrag wird als Transaktion bezeichnet.

2.5.1.2 Teilnehmerbetrieb

QUELLE: Hansen, s.923ff
Im Teilnehmerbetrieb kann jeder Teilnehmer unabhängig von anderen Teilnehmern Aufträge an die Rechenanlage senden.
Das heißt, jeder hat andere Datenbestände und Programme. Jeder Benutzer kann im Dialog selbst Programme erstellen, testen und ausführen. Entstanden ist der Begriff beim interaktiven Programmieren an Zentralrechnern, das sich inzwischen immer mehr auf dezentrale bzw. vernetzte Rechner verlagert.

2.5.2 Realzeitbetrieb

QUELLE: Hansen, s.926ff, Engel-Folien
Beim Realzeitbetrieb sind mit der Verarbeitung eines Auftrags strenge Zeitbedingungen verbunden, d.h. die Berechnung der Ergebnisse muss sofort, spätestens innerhalb einer vorgegebenen Zeitschranke abgeschlossen sein. Charakteristische Anwendungen des Realzeitbetriebs sind Mess- und Regelungsaufgaben, z.B. automatische Verkehrsregelung durch Fahrzeugzählung und Änderung der Ampelphasen.
Ein Realzeitsystem muss unterbrechbar sein, um auf unvorhergesehene oder katastrophale
Ereignisse unverzüglich reagieren zu können. Es erledigt meist folgende Aufgaben:
Erfassung von Daten aus der Umwelt
Auswertung der Daten
Reaktion durch Ausgabe von Steuerdaten
ständige Überprüfung des Geschehens, um die Sicherheit der zu steuernden Prozesse zu gewährleisten
Protokollierung der Abläufe
Information über den Zustand des Systems

2.5.3 Batchbetrieb

QUELLE: Hansen, s.921ff
Der vollständige definierte Auftrag wird der Rechenanlage zusammenhängend übergeben. Vom Augenblick der Übergabe an besteht für den Benutzer keine Möglichkeit mehr, auf den Ablauf des Auftrags einzuwirken. Eine besondere Form des Stapelbetriebs ist der Stapelfernbetrieb, bei dem die Benutzer ihre Aufträge über eine entfernt gelegene Eingabeeinheit an das Rechensystem übermitteln. Die Rechenanlage sendet die Ergebnisse ebenfalls an diesen Ort zurück. Typische Anwendungen des Stapelbetriebs sind umfangreiche Aufträge, die während der Verarbeitung keinen Eingriff durch den Benutzer erfordern, beispielsweise rechenintensive mathematische Problemstellungen oder Lohn- und Gehaltsabrechnungen. Ziele des Stapelbetriebs sind die

Maximierung des Durchsatzes und die optimale Auslastung aller Betriebsmittel. Diese Betriebsart wird oft in zentralen Rechenzentren (nachts) eingesetzt und war bis etwa 1975 vorherrschend. Mit der Dialogfähigkeit und der Dezentralisierung wird der Stapelbetrieb verstärkt auf wenige rechenintensive Aufgaben beschränkt.
Weitere Begriffe in diesem Kapitel:
Multi Tasking Betrieb
Multi User Betrieb
Time Sharing Betrieb

2.6 Portabilität:

Übertragbarkeit von offenen Betriebssystemen etc. auf eine beliebige andere Systemumgebung (Hardware, Software)

2.7 Interoperabilität:

die Verwendbarkeit von offenen Betriebssystemen in beliebigen, insbesondere heterogenen Rechnernetzen

3. DATEN

3.1 Darstellungsmöglichkeiten von Zahlen

3.1.1 Zahlensysteme

Der Wert einer Zahl bestimmt sich aus dem Wert der einzelnen Ziffer und ihrer Stellung innerhalb der Zahl. Beispiel : Dezimalsystem
Computer arbeiten mit nur zwei Zeichen, weil ihre elektronischen Bauteile nur die Zustände ein und aus bzw. 0 und 1 kennen. Eine Menge, die aus zwei Zeichen besteht heißt binäres System. Beispiele hierfür sind das Morsealphabet und Fußgängerampeln. Sind diese beiden Zeichen 0 und 1, spricht man nach Leibniz vom dualen Zahlensystem oder Dualsystem. Dual ist also ein Spezialfall von binär, und das Dualsystem ist ein Stellenwertsystem auf der Basis 2. Weil Dualzahlen unübersichtlich zu lesen sind, fasst man entweder drei duale Ziffern zu einem Stellenwertsystem auf der Basis 8 zusammen (Oktalsystem) oder vier duale Ziffern zu einem Stellenwertsystem auf der Basis 16 zusammen (Hexadezimalsystem).

Um Buchstaben und Sonderzeichen verarbeiten zu können, müssen diese mittels einem Code umgewandelt werden. Unter einem Code versteht man die Zuordnung der Zeichen eines Zeichenvorrats zu den Zeichen eines anderen Zeichenvorrats. Aufgrund der Datenübertragung ist eine Standardisierung von Codes notwendig.
Die meisten Rechnercodes basieren auf den BCD (binary coded decimals), bei denen
Dezimalziffern, Buchstaben und Sonderzeichen einzeln durch eine Bit – Folge fester Länger
(Bitmuster) dargestellt werden. Reine BCD – Codes basieren auf einer Länger von 4 Bit

(Tetrade). Weil damit nur 2^4 = 16 verschiedene Darstellungen möglich sind, verwendet man hauptsächlich folgende Erweiterungen,

3.1.2 Festkommazahlen

QUELLE: Hansen, s. 1007ff; Engel-Folien
unechte Dualzahlen: jede Dezimalziffer wird genauso wie jeder Buchstabe einzeln als
Dualzahl codiert
echte Dualzahlen: die Summe wird als Dualzahl codiert.

Bei Festkommazahlen ist das Komma eine gedachte Größe, der Computer berücksichtigt das
Komma nicht, es wird somit im Nachhinein wieder an die entsprechende Stelle gesetzt.
Anwendung: im betriebswirtschaftlichen Bereich, weil dort die Anzahl an Dezimalstellen begrenzt ist
Nachteil: vergeudet Speicherplatz und verlängert die Verarbeitungszeit unnötigerweise, da für eine Dezimalziffer bereits 4 Bit ausreichen.

3.1.3 Gleitkommazahlen

QUELLE: Hansen, s. 1015ff; Engel-Folien
bestehen aus Mantisse und Exponent. Zahl = M * B^e
Bsp.: 123,56 -> der Computer speichert 12356 * (10 hoch –2)
in 4Byte kann eine auf Stellen genaue Zahl gespeichert werden.
Anwendung: im mathematisch-technischen Bereich.

3.1.4 Standart-Codierungen

QUELLE: Hansen, s. 1004ff

3.1.4.1 ASCII (American Standard Code for Information Interchange)

QUELLE: Hansen, s. 1004ff
Der ursprüngliche ASCII basierte auf einer Länge von 7 Bit und wurde erweitert, um nationale Sonderzeichen zu berücksichtigen.

3.1.4.2 EBCDI – Code (Extended Binary Coded Decimals Interchange Code)

Der auch als IBM – Code bezeichnete EBCDIC wird ausschließlich bei Großrechnern und mittleren Systemen verwendet. ASCII und EBCDIC basieren auf einer Länge von 8 Bit, was 2hoch8 = 256 verschiedene Kombinationen ermöglicht.

3.1.4.3 Unicode

QUELLE: Hansen, s. 1007ff
UTF-16 (Unicode Transformation Format) ist ein 16-Bit –Code zur Darstellung von schrift- und Steuerzeichen. Er enthält zum Beispiel Zeichen für Arabisch, Hebräisch, Japanisch sowie technische und grafische Symbole. Die Entwicklung von Unicode (ISO 10646) ist noch nicht abgeschlossen,

3.2 Sortierverfahren

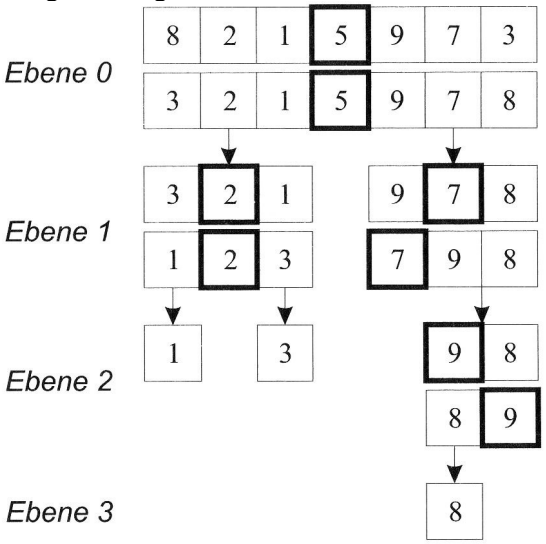
Auswahl anhand des Laufzeitaufwandes, welcher im Wesentlichen aus der Anzahl der Vergleiche und Vertauschungen ergibt.

3.2.1 Shell sort

QUELLE: KEINE !
Sortieren durch Einfügen mit abnehmender Schrittweite

3.2.2 Quicksort

QUELLE: Algorithmen & Datenstrukturen, s.127
Optimiertes Suchen durch Zerlegen
Es wird ein sog. Pivot-Element festgelegt und alle Werte werden mit diesem verglichen. Die größeren und kleineren Werte bilden zwei neue Folgen, über welche dieses Verfahren weiter angewendet wird, bis nur noch einelementige Folgen übrig sind.

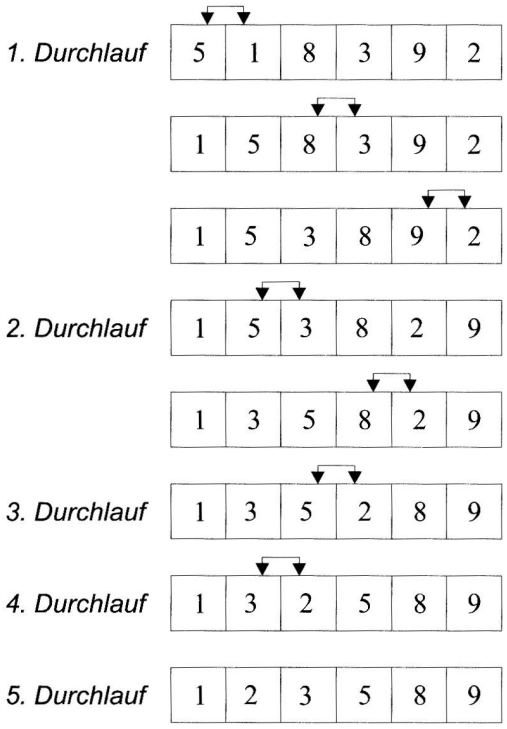


3.2.3 Heap Sort

QUELLE: Algorithmen & Datenstrukturen, s. 369 (aber hardcore Algorithmus)
Sortieren mit Bäumen

3.2.4 Bubble Sort

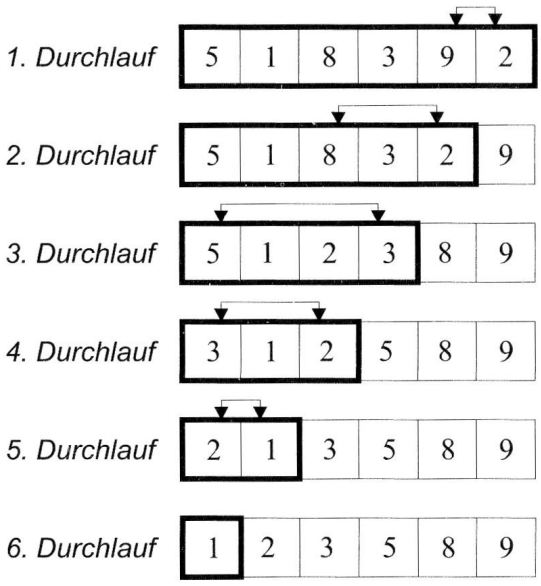
QUELLE: Algorithmen & Datenstrukturen, s.121
Im Datenbestand werden jeweils benachbarte Datensätze miteinander verglichen und paarweise so lange miteinander vertauscht, bis die gewünschte Reihenfolge (auf – oder absteigend) erreicht ist. Elemente die größer sind als ihr Nachfolger überholen diese daher und steigen zum Ende der Folge hin auf.



3.2.5 Selection Sort

QUELLE: Algorithmen & Datenstrukturen

Der Datenbestand wird beginnend mit dem ersten Element durchsucht, das größte gefundene Element wird ans Ende gestellt. Beim den folgenden Durchläufen werden die letzten Elemente nicht mehr angesehen, da diese ja bereits sortiert sind.

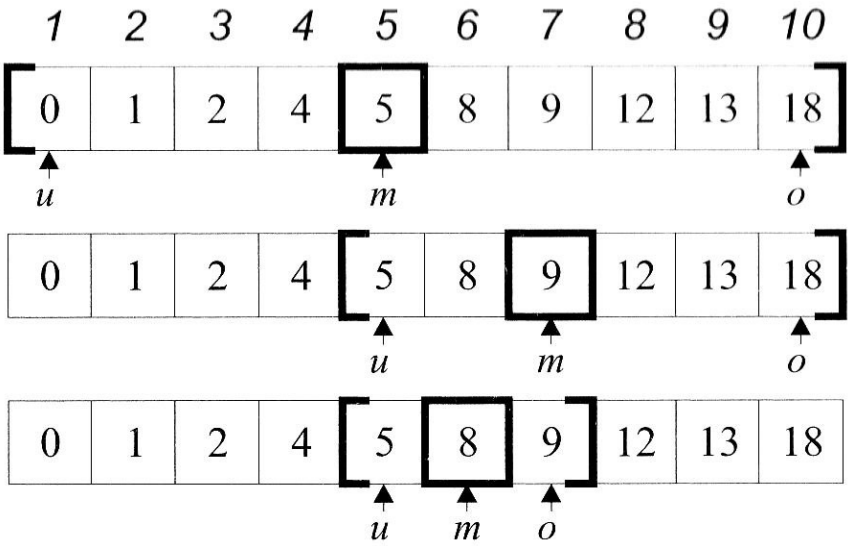


3.3 Suchverfahren

3.3.1 Binäres Suchen

Grundvoraussetzung: die Tabelle muss sortiert sein.
Ausgangspunkt: die Mitte der Tabelle
Verfahren: Es wird überprüft, ob die gesuchte Attributsausprägung kleiner oder größer als das gewählte Tabellenelement ist. Ist es kleiner, wird das mittlere Element der unteren Tabellenhälfte ausgewählt und das Verfahren wiederholt. Handelt es sich um ein größeres Element, wird die obere Tabellenhälfte halbiert und das Verfahren erneut angewendet. Die geschieht so lange, bis die gesuchte Attributsausprägung gefunden ist.

Anzahl der Zugriffe: Die maximale Anzahl der Zugriffe n beim binären Suchen ist in einer sortierten Tabelle mit m Elementen:
n = log2 m



3.3.2 Sequentielle Suche

Einfachstes und ineffizientestes Suchverfahren: Durchlaufen der Liste und Vergleich mit jedem Element, bis das gesuchte gefunden oder das Ende der Liste erreicht wurde.

3.4 Fehlererkennung bei Übertragung von Daten (Parity-Bit)

Die Verfahren zur Fehlererkennung entsprechen den Verfahren zur Wahrung der Datenintegrität, die die Unveränderheit empfangener Daten im Vergleich mit den gesendeten Daten gewährleisten sollen. Das einfachste Verfahren zur Erkennung von Übertragungsfehlern ist das Prüfbit (engl.: parity bit), das häufig zur Prüfung von 7-Bit-ASCII-Zeichen verwendet wird. Bei der Ermittlung des Prüfbits werden die sieben Datenbits des Zeichens addiert. Ergibt die Summe eine gerade Zahl, so ist das Prüfbit 0, bei einer ungeraden Summe hat das Prüfbit den Wert 1. Ist ein übertragenes Bit falsch, ist das Prüfbit ebenfalls falsch.

Zeichen	ASCII-Wert	Bitfolge	Bitfolge mit Prüfbit
A	65	1000001	**0**1000001
B	66	1000010	**0**1000010
C	67	1000011	**1**1000011

Ein Prüfbit ist jedoch kein sehr verlässliches Verfahren zur Erkennung von Übertragungsfehlern. Zu den weit besseren Verfahren, die beispielsweise auch mehrfache Bitfehler in Datenblöcken erkennen können, zählen die CRC-Verfahren (Abk. engl.: cyclical redundancy check).

3.5 Datenschutz - Datensicherheit

Datenschutz und *Datensicherheit* gehören zu den Aufgabenfeldern des IV-Managements. Nach DIN 44300, Teil 1 versteht man unter diesen Begriffen folgendes:
Datenschutz ist die Bewahrung schutzwürdiger Belange von Betroffenen vor Beeinträchtigung durch die Verarbeitung ihrer Daten. Die Betroffenen können sowohl natürliche als auch juristische Personen sein.
Datensicherheit ist die Bewahrung von Daten vor Beeinträchtigung, insbesondere durch Verlust, Zerstörung oder Verfälschung und Missbrauch.

3.5.1 Datenschutz

Datenschutz bezieht sich nicht primär auf den Schutz von Daten, sondern auf den Schutz der Personen und Gegenstände, deren Daten gespeichert werden. So beinhaltet Datenschutz Schutz sowohl von Daten als auch von DV-Anlagen und Programmen vor missbräuchlicher Benutzung. Computermissbrauch kann bestehen aus:
der unberechtigten Benutzung von Hardware und / oder Programmen,
der Einsichtnahme in geschützte Daten und
der Verfälschung oder der Vernichtung von Daten und / oder Programmen.

Hier ist zu unterscheiden, ob die zu schützenden Daten öffentlich, geschäftlich, vertraulich oder geheim sind. Zu dem technisch organisatorischen Datenschutz gehören z.B. Zugangskontrollen, Zugriffsrechte u.ä. Der Schutz personenbezogener Daten natürlicher Personen wird durch das Bundesdatenschutzgesetz, die Datenschutzgesetze der Bundesländer und seit 1995 durch die EU-Datenschutz-Richtlinie geregelt. Das Bundesdatenschutzgesetz muss zukünftig noch an die EU-Richtlinie angepasst werden.
Die drei wichtigsten Stichworte, die im Zusammenhang mit dem Datenschutz immer wieder genannt werden, sind **Vertraulichkeit, Verfügbarkeit und Integrität der Daten**. Um diese Eigenschaften sicherzustellen, werden in verstärktem Maße Methoden der Kryptographie erprobt, d. h. alle Daten werden nicht in lesbarer sondern in verschlüsselter Form gespeichert und übertragen. Am besten wäre es jedoch, wenn Daten gar nicht erst erhoben werden. So lautet die erste Frage aus Sicht des Datenschutzes, ob Daten überhaupt eingeholt werden dürfen, oder sollen. Erst dann ist die Frage zu stellen, wie Daten gegen Missbrauch zu schützen sind.

Besondere Aufmerksamkeit wird dem Datenschutz gewidmet, seit Institutionen Daten untereinander elektronisch austauschen. Hierbei geht in der Regel der Grund, warum die Daten erhoben wurden, und das Umfeld, in dem die Daten zu interpretieren sind, verloren. Auszutauschende Daten müssen auf ein notwendiges Minimum beschränkt werden, und jeder muss die über ihn gespeicherten Daten erhalten und gegebenenfalls korrigieren können.

3.5.1.1 Bundesdatenschutzgesetz

am 1.1.1978 in Kraft getreten
Vorschriften für die Zulässigkeit der Verarbeitung personenbezogener Daten
Unterscheidungen: Datenverarbeitung in behördlichen und sonstigen öffentlichen Instituten
Datenverarbeitung nicht-öffentlicher Stellen für fremde Zwecke

Jeder Bürger hat Recht auf:

- Auskunft
- Berichtigung von Daten
- Sperrung von Daten
- Löschung von Daten (außer Polizei, Bundeswehr, Geheimdienst etc.)

Es müssen entsprechende Vorkehrungen getroffen werden, um die personenbezogenen Daten zu schützen.
Im BDSG werden die Aufgaben des Bundesdatenschutzbeauftragten festgelegt. Er ist die Kontrollinstanz für die öffentliche Datenverarbeitung und kann von jedem Bürger in Anspruch genommen werden.
Die nichtöffentlichen Bereiche der Datenverarbeitung haben unter bestimmten Voraussetzungen sogenannte Betriebliche Datenschutzbeauftragte.
Auf Landesebene können über das BDSG hinaus zusätzliche Landesdatenschutzgesetze die Vorschriften für die öffentliche Datenverarbeitung regeln.

3.5.2 Datensicherheit

Datensicherheit wird allmählich durch den umfassenderen Begriff IT-Sicherheit ersetzt. Datensicherung umfasst alle Maßnahmen und Einrichtungen, die *Datensicherheit* herbeiführen oder aufrechterhalten. Im Einzelnen handelt es sich dabei um die Sicherung ordnungsmäßiger Arbeitsabläufe im gesamten IV-Bereich, von Daten und Programmen vor Verlust, Zerstörung oder Verfälschung und von DV-Anlagen und Nebeneinrichtungen vor Beschädigung und Zerstörung.

Die **zu schützenden Gegenstände** gliedert man im Allgemeinen in:

- **Objektschutz**
  Betriebsgelände, Gebäude, Räume
- **Hardware- und Kommunikationsschutz**
  DV-Anlagen mit gesamter Peripherie und Netze
- **Softwareschutz**
  Programme einschließlich der Dokumentation
- **Datenschutz**
  Dateien und Datenträger.

Die *Datensicherheit* geht bei der Bewertung der Sicherheit von Systemen der Informationstechnik von folgenden Grundgefahren aus:

- *Verlust der Verfügbarkeit der Daten*

→ benötigte Daten sind nicht zugänglich oder zerstört

- *Verlust der Integrität (Unversehrtheit) von Daten*

→ Daten sind bewusst oder unabsichtlich verfälscht

- *Verlust an Vertraulichkeit*

→ Unbefugte erhalten Kenntnis von vertraulichen Daten.

Eine häufig verwendete Realisierung der Datensicherung ist folgende: Jeder Benutzer kann für seine eigenen Dateien entscheiden, ob er sie fremden Benutzern zugänglich macht oder nicht. Die Möglichkeit, bestimmte Zugriffsrechte an verschiedene Benutzergruppen zu vergeben, hat der Benutzer nicht. Macht er eine Datei allen Benutzern zugänglich, so kann er nur noch die Arten des Zugriffs beschränken. Hierbei stehen verschiedene Kombinationen von Lese- bzw. Schreibrechten und Ausführrechten zur Verfügung. Diese Zugriffsbeschränkung gilt dann für alle fremden Benutzer.
Die meisten Rechenanlagen erlauben es jedoch, eine Datei mit einem Passwort zu versehen. Dies ist ein Geheimcode, der regelt, ob ein Benutzer überhaupt Zugang zu einer Datei erhält. Das einfachste und in der Praxis gebräuchlichste Verfahren zur Sicherung von Daten gegen Fehlfunktionen des Systems oder Fehlbedienungen des Benutzers ist die Sicherungskopie. In regelmäßigen Abständen werden alle in dieser Zeit geänderten Dateien gesichert, d. h. auf eine andere Speichereinheit, meist ein Magnetband, kopiert. Wird eine Datei versehentlich zerstört, so steht immer noch eine frühere Version der Datei zur Verfügung.
Gegen Fehlfunktionen des Betriebssystems kann sich ein Benutzer kaum schützen. Eine große Gefahr geht in diesem Zusammenhang von Computerviren aus.
Methoden der Datensicherung sollen zugleich gegen Spionage schützen. Besonders empfindlich gegen unerlaubtes Abhören ist hierbei die Datenübertragung, z. B. zwischen Speicher und Zentraleinheit oder zwischen verschiedenen Rechnern in einem öffentlichen Netz oder einem Rechnernetz. Wirksamen Schutz bieten hier voraussichtlich die Methoden der Kryptographie, d. h. alle Daten werden nicht in lesbarer sondern in verschlüsselter Form gespeichert und übertragen. Dies schützt zugleich gegen die sog. Wiederaufbereitung:
Da Daten in Computern und Speichern faktisch nicht gelöscht, sondern nur der Zugriffspfad beim Löschen zerstört wird, können diejenigen, denen anschließend der spezielle Speicherbereich zugewiesen wird, alle Daten, die vorher dort standen, lesen und so leicht an geheime Informationen gelangen, wenn diese nicht verschlüsselt wurden.

3.5.2.1 § Die 10 Gebote der Datensicherung §

1. Zugangskontrolle

Du sollst Unbefugten den Zugang zu Datenverarbeitungsanlagen, mit denen personenbezogene Daten verarbeitet werden, verwehren.

2. Datenträgerkontrolle
Du sollst verhindern, dass Datenträger unbefugt gelesen, kopiert, verändert oder entfernt werden können.

3. Speicherkontrolle
Du sollst die unbefugte Eingabe in den Speicher sowie die unbefugte Kenntnisnahme, Veränderung oder Löschung gespeicherter, personenbezogener Daten verhindern.

4. Benutzerkontrolle
Du sollst verhindern, dass Datenverarbeitungssysteme mit Hilfe von Einrichtungen zur Datenübertragung von Unbefugten genutzt werden können.

5. Zugriffskontrolle
Du sollst gewährleisten, dass die zur Benutzung eines Datenverarbeitungssystems Berechtigten ausschließlich auf die ihrer Zugriffsberechtigung unterliegenden Daten zugreifen können.

6. Übermittlungskontrolle
Du sollst gewährleisten, dass überprüft und festgestellt werden kann, an welchen Stellen personenbezogene Daten durch Einrichtungen zur Datenübertragung übermittelt werden können.

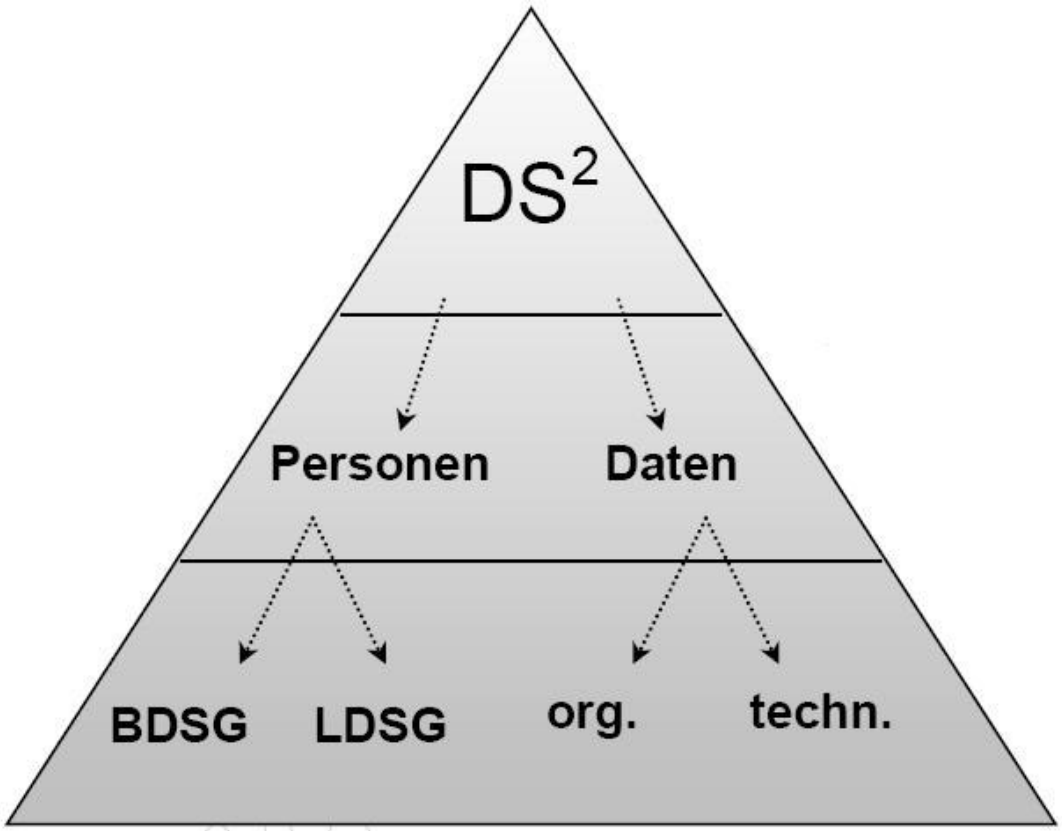
7. Eingabekontrolle
Du sollst gewährleisten, dass nachträglich überprüft werden kann, welche personenbezogenen Daten zu welcher Zeit von wem in Datenverarbeitungssysteme eingegeben worden sind.

8. Auftragskontrolle
Du sollst gewährleisten, dass personenbezogene Daten, die im Auftrag verarbeitet werden, nur entsprechend den Weisungen des Auftraggebers verarbeitet werden können.

9. Transportkontrolle
Du sollst verhindern, dass bei der Übertragung personenbezogener Daten sowie beim Transport von Datenträgern die Daten unbefugt gelesen, kopiert, verändert oder gelöscht werden können.

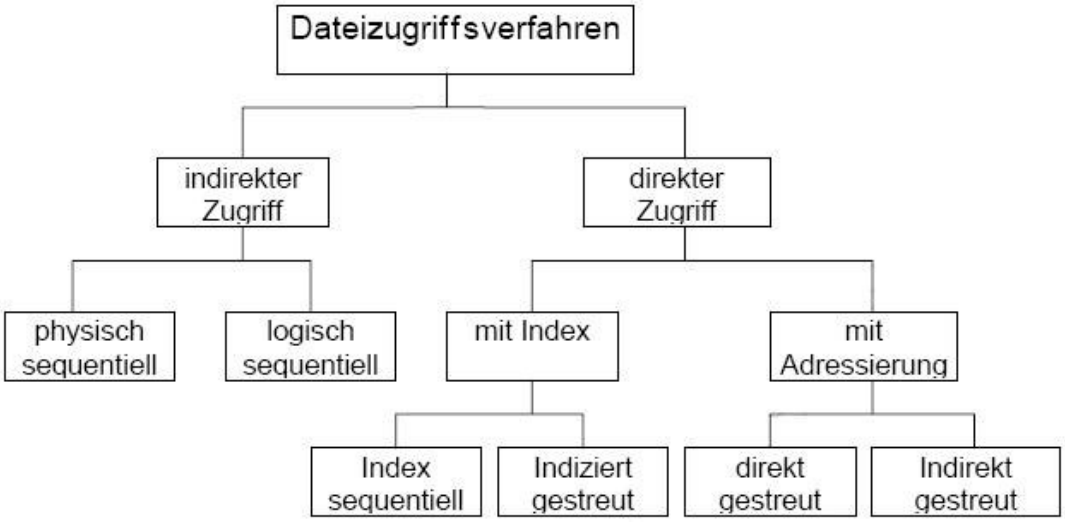
10 Organisationskontrolle
Du sollst die innerbetriebliche Organisation so gestalten, dass sie den besonderen Anforderungen des Datenschutzes gerecht werden.

3.5.3 Überblick



3.6 Dateiorganisation

3.6.1 Zugriffsarten (Dateiorganisationsverfahren)



3.6.1.1 physisch sequentiell

- die Datensätze werden nacheinander so gelesen, wie sie physisch abgespeichert sind

3.6.1.2 logisch sequentiell

- die Datensätze werden so gelesen, wie die Verpointerung ist

3.6.1.3 Index-sequentiell

- bei mehreren Feldern und Vorwärts- und Rückwärtsverkettung lohnt sich der Verwaltungsaufwand für Pointerketten bei einer logisch-sequentiellen Verarbeitung nicht mehr
- Lösung: neben der Datendatei wird pro identifizierenden Ordnungsbegriff eine Indexdatei angelegt
- die Indexdatei enthält die Ordnungsbegriffe in sortierter Reihenfolge, mit dem entsprechenden Datensatz
- Voraussetzung: identifizierender Ordnungsbegriff; direkt adressierbarer Datenträger; Betriebssystem muss Index-sequentielles File Management System (ISFMS) beherrschen

3.6.1.4 Indiziert-gestreut

- wie Index-sequentiell, jedoch muss der Ordnungsbegriff nicht identifizierend sein
- eindeutige Schlüssel (Primärschlüssel) werden in Primärindize abgespeichert
- klassifizierende Schlüssel in Sekundärindize, dieser ist für direkten Zugriff nicht geeignet
- d.h.: Sekundärindex kann Duplikate enthalten, Primärindex nicht
- es heißt gestreut, da die Schlüssel der Datendatei in mehreren Indize verstreut sind

3.6.1.5 Direkt-gestreut

- für jeden Ordnungsbegriff der vorkommen kann, muss physischer Speicherplatz reserviert werden
- Voraussetzung: Speichermedium ist direkt adressierbar
- schnellste Zugriffsart, da nur 1 Zugriff
- Voraussetzung: geschlossener Nummernkreis
- durch Schlüsseltransformation (HASH- Algorithmus) wird die 1. Adresse berechnet
- 1:1 Verhältnis zwischen Ordnungsnummer und physischer Adresse

3.6.1.6 Indirekt-gestreut

- verwendet man, wenn direkt gestreut zuviel Speicherplatz verschenkt
- die Ordnungsbegriffe werden z.B. zu Paaren geordnet
- für jedes Paar gibt es genau einen Speicherplatz
- kommen beide Ordnungsbegriffe des Paares vor, kommt der 2. in den Überlauf-Bereich (hier: sequentielles Suchen)

Als Speicherungsformen bezeichnet man die Verfahren, nach denen Dateien auf peripheren Speichern physisch gespeichert werden. Dazu gehört auch Aufbau und Pflege der nach diesen Speicherungsformen angelegten Speicherbereiche und die Organisation des Zugriffs auf die darin enthaltenen Datensätze. Häufig wird auch der Begriff **Zugriffsmethoden** benutzt.

Ausschlaggebend für die Festlegung auf die für eine Datei am besten geeignete Speicherungsform sind:
1. die gewünschte Verarbeitungs- und Zugriffsformen
2. die Eigenschaften der Datei
3. der Aufbau des Ordnungsbegriffs

3.6.2 Verarbeitungs- und Zugriffsformen

Verarbeitungsformen können sortiert oder unsortiert sein.
Zugriffsformen können fortlaufend oder wahlfrei sein. Da meistens Platten als peripherer Speicher eingesetzt werden, ergibt sich keine Einschränkung des Zugriffsverfahrens, da sie beide Verfahren gestattet. Magnetbänder werden nur noch als Datensicherungswerkzeug eingesetzt.

3.6.3 Eigenschaften der Datei

- Benutzungshäufigkeit: Zeitliche Häufigkeit, mit der die Datei benötigt wird (monatlich, tägl. etc.) Sie bestimmt, ob die Datei ständig im Arbeitsspeicher verfügbar zu halten oder nur bei Bedarf zu laden ist.

- Zugriffshäufigkeit: Zeitliche Häufigkeit, mit der auf einzelne Datensätze der Datei zugegriffen wird (z.B. mehrmals täglich, permanent)

- Veränderungshäufigkeit: Zeitliche Häufigkeit, mit der Zu- und Abgänge von Datensätzen vorkommen. Dateien mit häufigen Zu- und Abgängen werden als dynamisch bezeichnet.

- Bewegungshäufigkeit: Zeitliche Häufigkeit, mit der Satzinhalte verändert werden.

- Zugriffs-, Veränderungs-, und Bewegungshäufigkeit ergeben sich daraus, wie oft die Dateioperationen Suchen, Einfügen, Entfernen und Ändern vorkommen. Der Zeitbedarf für jede dieser Operationen hängt von der Speicherungsform ab.

- Umfang: Anzahl der Datensätze und Länge der Datei.

- Umfangsveränderung (Wachstum): Anzahl der Zu- und Abgänge von Datensätzen innerhalb eines festen Zeitraums (pro Monat, Jahr)

- Umfang und Wachstum der Datei bestimmen zunächst den jetzt und zukünftig benötigten peripheren Speicherplatz. In Verbindung mit der Speicherungsform ergibt sich daraus aber auch der Zeitbedarf für das Lesen, das Löschen oder einen Neuaufbau der gesamten Datei.

3.6.4 Aufbau des Ordnungsbegriffs

3.6.4.1 sequentielle Speicherung

Alle Plätze sind lückenlos hintereinander gespeichert und werden im „Normalfall“ nach einem Ordnungsbegriff (Schlüssel) sortiert. Sie kommen für den starr fortlaufenden Zugriff in Frage. Wahlfreier Zugriff ist nicht möglich. Es ist nur ein Stapelbetrieb möglich.

Nachteil: Zeitaufwand ist bei häufigen Zu- und Abgängen sehr hoch, da immer die gesamte Datei gelesen/reorganisiert werden muss.

3.6.4.2 Verkettete Speicherung

Jeder Datensatz beherbergt ein spezielles Kettenfeld (Pointer), der die physische Adresse des nachfolgenden (oder vorangehenden) Datensatzes aufnimmt → Datei kann fortlaufend verarbeitet werden.

Vorteil: einzelne Datensätze können unabhängig von der Reihenfolge gespeichert werden
Nachteil: aufwendig hinsichtlich Speicherplatz, Zugriffszeit und Änderungsdienst.

Wird in dieser Form nicht mehr genutzt, wohl aber in Verbindung mit anderen Speicherungsformen.

3.6.4.3 Index-Verfahren

Zusätzlich zu den abgespeicherten Daten wird eine Index-Tabelle angelegt, die die Schlüssel der jeweiligen Datensätze und ihre korrespondierenden physischen Speicheradressen abspeichert.

Vorteil: die Daten brauchen nur in der Index-Tabelle geordnet werden, nicht auf dem Datenträger → Performance-Vorteil
Nachteil: Fortlaufende Verarbeitung der Datei

3.6.4.4 index-sequentielle Speicherung

Aus Gründen der Datenunabhängigkeit entsprechen die logischen Adressen nicht den physischen. Mehrere Datensätze können in einem Block gespeichert werden → Prinzip des Wörterbuches: innerhalb eines Blocks sind die Daten nach Schlüssel sortiert

Vorteil: sowohl wahlfreier als auch fortlaufender Zugriff
Nachteil: Überlaufblöcke (Organisationsaufwand); Reorganisation der Blöcke, um Effizienz zu erhöhen (gelöschte Datensätze auch physisch löschen)

3.6.4.5 index-verkettete Speicherung

zu jedem Schlüssel in der Index-Tabelle wird noch der Schlüssel des logisch nachfolgenden Datensatzes gespeichert.
Beim Ändern/Hinzufügen der Datensätze sind die Nachfolgeradressen entsprechend zu ändern.

Vorteil: einfache Art, Datensätze hinzuzufügen/zu löschen, da nur die Nachfolgeeinträge geändert werden müssen.
Schnell fortlaufender Zugriff auf die logisch fortlaufenden Datensätze.
Nachteil: evtl. zeitaufwendiger Zugriff, wenn die Datensätze über den ganzen Datenträger verteilt sind.

3.6.4.6 Speicherung durch binäre Bäume

Weiterführung des Prinzips der Verkettung → jedoch 2 Nachfolger
Analogie: binäres Suchen

Vorteile: kurze Suchzeiten bei ausbalanciertem Baum, v.a. bei großen Datenbeständen
Nachteil: häufige Reorganisation der Index-Tabelle

3.6.4.7 Gestreute Speicherung

Es besteht ein rechnerischer Zusammenhang zwischen dem Schlüssel und der physischen Speicheradresse. Die Umrechnungsformel heißt Speicherfunktion (Hash-Funktion)

Direkte Adressierung
Zuordnung ist eindeutig umkehrbar.

Vorteile: kurze Zugriffszeiten und Möglichkeit, Zugänge leicht einzufügen
Nachteil: Reservierung von Speicherplätzen für (noch) nicht existierende Datensätze → wenig geeignet für Schlüssel mit großen Abständen (aber: Faltung etc.)

Indirekte Adressierung
Zuordnung zwischen Nummer und Adresse ist nicht eindeutig umkehrbar →
Methode: Divisionsrestverfahren

3.7 Kryptographie

Kryptographie ist die Lehre, die sich mit der Verschlüsselung von Informationen / Daten beschäftigt.

Das heißt 2 Parteien wollen Daten austauschen, ohne dass diese Daten von Dritten eingesehen werden können. Diese Daten müssen also nach einem bestimmten Verfahren für fremde Augen unkenntlich gemacht werden.

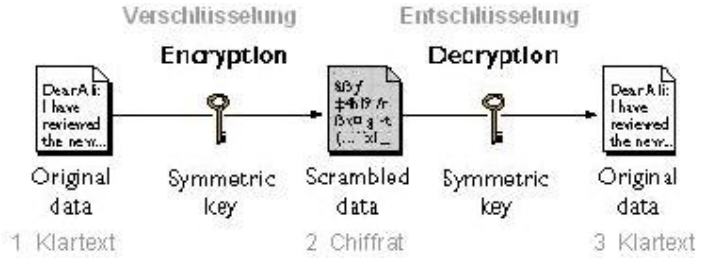
Ihren Ursprung und die längste Tradition hat die Kryptographie im Militär, wo es verständlicherweise wichtig ist, bestimmte Daten vor Augen und Ohren des Feindes geheim zu halten. Auch in der Wirtschaft spielt die Kryptographie eine wichtige Rolle, damit sich ein Unternehmen beim internen Austausch sensibler Informationen vor Industriespionage u.ä. schützen kann.

allgemein:
Klartext = Der zu verschlüsselnde Text
Chiffrat = der verschlüsselte Klartext
IT-Sicht:
Klartext = in der Computerwelt eine Folge von Bits.

Schlüssel = ebenfalls eine Folge von Bits.

Bei der **Verschlüsselung** werden nun der Klartext und der Schlüssel mit Hilfe von mathematischen Funktionen verknüpft. Es entsteht das sogenannte Chiffrat, welches eine komplett neue Folge von Bits ist.

Die **Entschlüsselung** erfolgt genau umgekehrt. Es wird ebenfalls ein Schlüssel benötigt, je nach Verfahren ist dies derselbe oder ein anderer als der Schlüssel zum Verschlüsseln.



Es gibt 2 Verfahren für die Verschlüsselung:

- symmetrisches Verfahren
- asymmetrisches Verfahren

3.7.1 symmetrische Verschlüsselung

→ auch als „klassisches private-key-Verfahren“ bezeichnet.

Es gibt **genau einen Schlüssel**, der sowohl für die Ver-, als auch für die Entschlüsselung benötigt wird. (siehe Grafik oben)

Dieser Schlüssel muss unabhängig von der Nachricht auf einem sicheren Weg an den Empfänger übertragen werden.

Das gängigste Beispiel für symmetrische Verschlüsselung ist der sogenannte

DES (Data Encryption Standard):

Entwickelt wurde dieses Verfahren Anfang der 70er Jahre von der NSA (National Security Agency) und IBM in den USA. Die Nachricht ist hier ein 64-Bit-Block, der Schlüssel ist 56 Bit lang. Es gibt also 2^{56} Möglichkeiten, wie der Schlüssel aussehen kann.

Nachteil: Das Verfahren ist sehr anfällig für sogenannte **„brute force attacks“**, d.h. das Errechnen und Ausprobieren aller Möglichkeiten wie der Schlüssel aussehen kann.

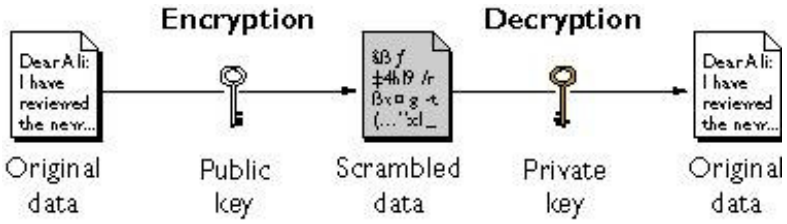
3.7.2 asymmetrische Verschlüsselung

→ sogenanntes „public-key-Verfahren“

Bei diesem Verfahren hat jeder Teilnehmer ein Schlüsselpaar, bestehend aus:

- dem **öffentlichen Schlüssel** (für jeden zugänglich)
- dem **privaten/geheimen Schlüssel** (sollte nur dem Besitzer bekannt sein)

Das Verschlüsseln der Nachrichten erfolgt mit dem öffentlichen Schlüssel (public key) des Empfängers. Entschlüsseln kann man nur mit dem geheimen Schlüssel (private key).



Vorteil: Der geheime Schlüssel muss (im Gegensatz zum symmetrischen Verfahren) nie übertragen werden.

Das gängigste Beispiel für die asymmetrische Verschlüsselung ist:
RSA (Rivest, Shamir, Adelmann)

Dieses Verfahren beruht auf der Zerlegung sehr großer Primfaktoren. Hierbei ist allerdings nur das Produkt 2er Primzahlen bekannt (öffentlich). Die beiden Primzahlen selbst sind nicht bekannt.

Nachteil bei diesem Verfahren ist, dass die Entschlüsselung bei sehr großen Schlüsseln u.U. sehr lange dauern kann.

Eine effektivere Methode ist eine Mischform von symmetrischen und asymmetrischen Verfahren, wie man es beim SSL (Secure Socket Layer) vorfindet:
Man verschlüsselt hier die Nachricht mit dem symmetrischen Verfahren, und zusätzlich den Schlüssel mit dem asymmetrischen Verfahren, damit dieser mit der Nachricht verschickt werden kann.

3.7.3 Digitale Unterschrift

Unter einer digitalen Unterschrift versteht man einen kryptographisch geschützten Nachweis, dass ein eindeutig identifizierter Benutzer einen Datenbereich (ein digitales Dokument) unterzeichnet hat. Durch eine digitale Unterschrift kann man nachweisen, dass ein Dokument wirklich von einer bestimmten Person stammt und bei der Übertragung nicht verändert wurde.

Bei diesem Verfahren wird die Nachricht (bzw. eine bestimmte Passage) mit dem privaten Schlüssel des Senders verschlüsselt.
Der Empfänger kann nun mit dem öffentlichen Schlüssel des Senders diese Passage wieder entschlüsseln und sieht somit, ob die Nachricht noch im Originalzustand ist, oder nicht.

4. DATENBANKEN / DBMS

4.1 Grundlagen

4.1.1 Datenmodelle

Ein Datenmodell besteht aus Beschreibung und Spezifikation von:

- **Datentypen** (data types),
- **Beziehungen und Abhängigkeiten** (relationships),
- **Beschränkungen** (constraints),
- **Grundoperationen** (basic operations) und
- **benutzerdefinierten Operationen** (userdefined operations, behaviour).

In der Regel werden mehrere verschiedene Datenmodelle zur Beschreibung der verschiedenen
Aspekte einer Datenbank benutzt:

1. **konzeptionelles Schema** (conceptional, high-level schema),
2. **Implementationsschema** (implementational schema) und
3. **physikalisches Schema** (physical, low-level schema).

Das ***konzeptionelle Schema*** einer Datenbank kann zum Beispiel mit einem ER-Modell beschrieben werden. Dieses verwendet

- Entitäten (Entities),
- Attribute (repräsentieren Eigenschaften der Entitäten) und
- Relationen (Beziehungen zwischen Entitäten)

um Aspekte der realen Welt abzubilden.

Das ***Implementationsschema*** verwendet die von der konkreten Datenbanksoftware vorgegebenen Möglichkeiten, z.B. das relationale Modell.

Das ***physikalische Schema*** bleibt dem Benutzer verborgen, hier werden spezifische Datenstrukturen des DBMS benutzt, z.B. B-Bäume.

Die Einhaltung des Datenmodells zu überprüfen, ist Aufgabe des DBMS.
Die Mehrzahl der Datenbanken basieren auf einem der drei folgenden Modelle:

- Das *hierarchische Datenmodell* wird durch die Struktur eines Baumes beschrieben. Dadurch kann die „reale Welt“ häufig nicht in der Hierarchie abgebildet werden.
- Dem *Netzwerkdatenmodell* liegt die Struktur eines Graphen zugrunde, wodurch Querverbindungen zwischen verschiedenen Ästen möglich werden.
- Das *relationale Datenmodell* beruht auf der Struktur „Tabelle“, die Relationen beschreibt.

4.1.2 Definition Datenbank

Eine Datenbank ist eine Menge von Daten, die nach einem Ordnungskriterium, das sie als zusammengehörend kennzeichnet, in maschinell lesbaren externen Speichern gespeichert sind (z. B. Kontostände aus der

Kontokorrentbuchhaltung nach Kontenbezeichnung, Lagerbestände nach Artikelbezeichnungen).

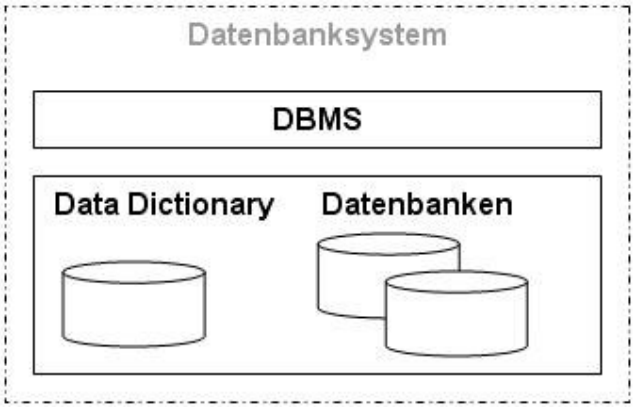
4.1.3 Definition Datenbankmanagementsystem

Das Datenbankverwaltungssystem (DBMS) ist der Teil eines Datenbanksystems, der zur Verwaltung der Daten der Datenbank dient. Es ist zum einen ein Programmsystem, das Methoden und Werkzeuge zur Einrichtung und Pflege der Datenbank beinhaltet und alle von den Anwendungsprogrammen verlangten Zugriffe (Lesen, Ändern, Hinzufügen, Löschen von Daten) auf der Datenbank ausführt, wobei Datensicherheit und Datenschutz gewährleistet sein müssen. Zum anderen stellt es weitere Funktionen, z.B. Abrechnung der Leistungsinanspruchnahme durch die Benutzer, Restauration einer zerstörten Datenbank, zur Verfügung. Das DBMS besteht aus einer Vielzahl von Werkzeugen und Generatoren.

Unter einer Datenbank versteht man einen zentral gespeicherten Datenbestand, der über anwendungsunabhängige Zugriffsverfahren weitgehend zentral verwaltet wird.

Das Datenbankverwaltungssystem (engl.: data base management system, abgekürzt: DBMS) verwaltet diesen Datenbestand und ermöglicht gleichzeitige Zugriffe von mehreren Anwendungsprogrammen und mehreren Benutzern auf den verwalteten Datenbestand. Das Datenbankverwaltungssystem dient auch zur Administration der Daten, wozu beispielsweise die Definition von Datentypen und Attributen, die Definition von Zugriffsrechten usw. gehören.

Ein Datenbanksystem besteht aus einem *Datenbankverwaltungssystem*, einer *Datenbank* sowie aus zusätzlichen Programmen, die die Bearbeitung, Verwaltung und Auswertung der gespeicherten Daten vereinfachen.

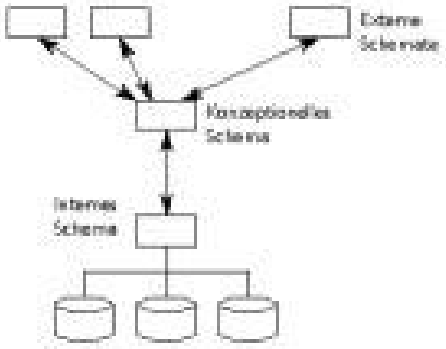


Ein **Data-Dictionary** enthält Metadaten zur Dokumentation der in der Datenbank abgespeicherten Datenobjekte und –attribute. Es werden auch meist Metadaten zur Beschreibung der Datenverwendung durch Anwendungen, Benutzer und Transaktionen, Verantwortlichkeiten, Änderungszeitpunkte usw. im Data-Dictionary abgelegt.

4.1.4 ANSI-SPARC-Dreischichtenmodell

Eine andere Einteilung der Datenmodelle in verschiedene Ebenen nimmt die ANSI/SPARC- Architektur vor. Ziel des Dreischichtenmodells ist es, die Entkopplung von Konzeption, Anwendung und Speicherung zu erreichen.

Verschiedene Benutzergruppen sehen die Datenbank auf unterschiedliche Weise, indem jeder Benutzergruppe ein externes Schema zugeordnet wird. Die Struktur der Gesamtdatenbank wird durch das konzeptionelle Schema beschrieben. Das interne Schema beschreibt die physikalische Speicherung der Daten auf der Basis von Datensätzen und Zugriffspfaden.



Durch die Einteilung in drei Ebenen findet eine zweifache Datenabstraktion statt. Zum einen die Abstraktion zwischen den externen Schemata und dem konzeptionellen Schema. Diese Abstraktion führt zur logischen Datenunabhängigkeit. Eine weitere Abstraktion findet zwischen dem konzeptionellen Schema und dem physikalischen Schema statt. Hieraus resultiert die physische Datenunabhängigkeit.

4.2 Konzeptionelle Datenmodelle

Das konzeptionelle Datenmodell wird oft auch als Informationsmodell bezeichnet. Hier steht die logische Gesamtstruktur der Daten, ihrer Eigenschaften und ihrer Beziehungen untereinander, im Vordergrund. Es wird weder die physikalische implementierungsabhängige Organisation der Datenbank noch die Wünsche von Anwendungsprogrammen berücksichtigt. Das konzeptionelle Schema einer Datenbank kann zum Beispiel mit einem **ER-Modell** beschrieben werden. Dieses verwendet

- Entitäten (Objekte),
- Attribute (repräsentieren Eigenschaften der Entitäten) und
- Relationen (Beziehungen zwischen Entitäten) um Aspekte der realen Welt abzubilden.

Das Konzeptionelle Datenmodell verfolgt dabei die groben **Ziele**:

- modellhafte Abbildung eines anwendungsorientierten Ausschnitts der realen Welt (Miniwelt)
- Nachbildung von Vorgängen durch Transaktionen

Daraus ergeben sich folgende **Nebenbedingungen**:

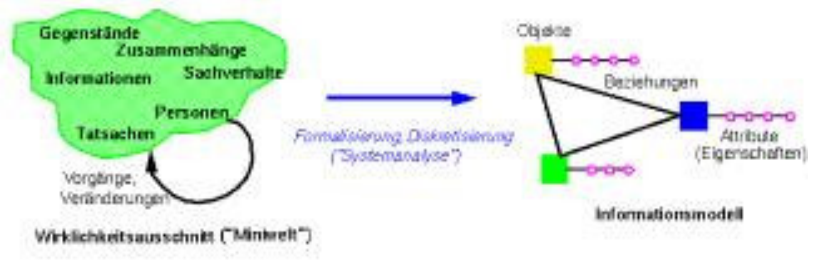
- genaue Abbildung
- hoher Aktualisierungsgrad
- Verständlichkeit
- Natürlichkeit
- Einfachheit, ...

Um den Ausschnitt der realen Welt möglichst gut im Modell abzubilden, leitet man die reale Welt schrittweise aus **verschiedenen Sichten** ab:

1. Information in unserer Vorstellung
2. Informationsstruktur: Organisationsform der Information
3. Logische (zugriffspfadunabhängige) Datenstruktur (Was-Aspekt)
4. Physische Datenstruktur (Was- und Wie-Aspekt)

Daraus ergeben sich folgende **Zwischenziele**:

- Erhebung der Information in der Systemanalyse (Informationsbedarf!)
- Informationsmodell (allgem. Systemmodell)

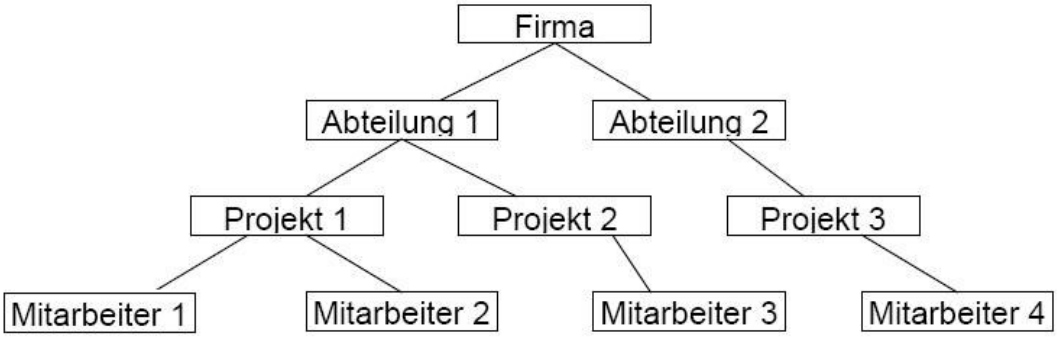


Zur formalen Beschreibung aller in einer Datenbank enthaltenen Daten und ihrer Beziehungen untereinander verwendet man Datenmodelle. Hierbei stellt man Objekte, deren Eigenschaften (Attribute) und ihre Beziehungen untereinander auf. Dies führt zum Entity-Relationship-Modell. Die Mehrzahl der Datenbanken basiert auf einem der drei folgenden Modelle:

- Hierarchisches Datenmodell
- Netzwerkdatenmodell
- Relationales Datenmodell

4.2.1 Hierarchisches Datenmodell

Die hier zugrundeliegende Struktur ist der Baum. Hierarchiemodelle lassen sich einfach und effizient auf physikalische Speicherungsstrukturen (z.B. verkettete Listen) abbilden, haben allerdings den Nachteil, dass sich die „reale Welt“ häufig nicht in einer Hierarchie darstellen lässt. Beispielsweise ist ein Mitarbeiter oft in mehreren Projekten tätig, und ein Projekt wird meistens von mehreren Abteilungen bearbeitet. Dies lässt sich in der Abbildung unten nicht darstellen.



- Konstruktionsgrundlage ist die Baumstruktur

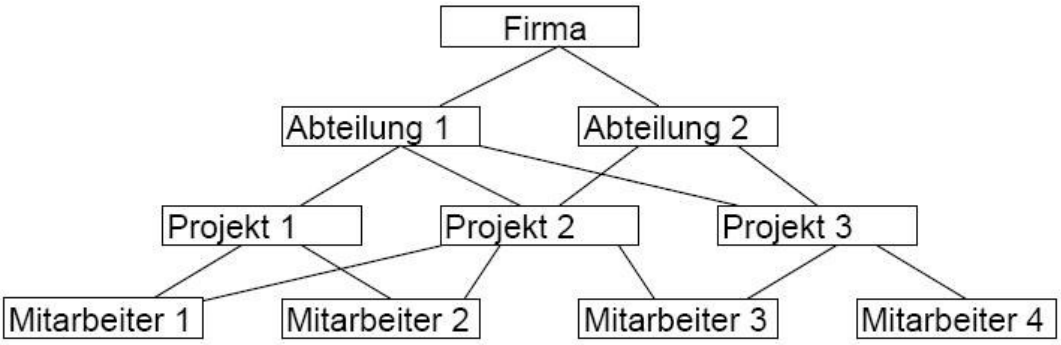
- Datensätze werden durch Knoten dargestellt und über Kanten verknüpft
- es sind nur 1:m-Beziehungen zugelassen
- Jeder Entitytyp hat genau einen Vorgänger (Ausnahme: Wurzel) und kann mehrere Nachfolger besitzen

Nachteile:
- liegt eine m:n Beziehung vor, müssen zwei Hierarchien gebildet werden (Redundanz)
- Vater, Mutter, Kind-Beziehungen können nicht dargestellt werden
- schlechte Strukturflexibilität

Vorteile:
- wenn „Einstiegsknoten“ (Wurzel) bekannt, schneller Zugriff
- liegt dem DB-Verwaltungssystem IMS von IBM zugrunde (Kleine Abänderung: Zeigersegment enthält Adresse, nicht Benutzerinfo bei Redundanzen)

4.2.2 Netzwerkdatenmodell

Wie das hierarchische Datenmodell kann beim Netzwerkdatenmodell jede Entität mehrere Nachfolger haben, es kann aber allerdings auch mehrere Vorgänger besitzen. Es kann mehrere Entitäten geben, die keinen Vorgänger haben (beim hierarchischen Datenmodell nur die Wurzel). Damit können sich also auf der obersten Stufe mehrere Entitäten befinden. Es liegt die Struktur des Graphen zugrunde. Netzwerke lassen sich nicht so einfach wie Bäume auf den physikalischen Speicher (z.B. Magnetplatten) abbilden.



- Darstellung durch ein Netz (zusammenhängender, gerichteter Graph)
- 1:m Beziehungen und m:m Beziehungen zugelassen
- Jeder Entitytyp kann mehrere Vorgänger und mehrere Nachfolger besitzen
- es kann mehr als einen Entitytyp geben, der keinen Vorgänger hat
- Entity-Set = Record-Typ
- eine Beziehung zwischen zwei Record-Typen wird Set-Typ genannt, die beteiligten Record- Typen Owner (1) und Member (m)
- Im Unterschied zum hierarchischen Modell darf ein Record-Typ, der in einem Set-Typ eine Member-Funktion übernimmt, auch in einem anderen Set-Typ ebenfalls Member-Funktion übernehmen.
- „Einstiegsknoten“ ist beliebig: Keytable
- typisch für Erzeugnisstrukturen in Fertigungsbetrieben

- von CODASYL- Gruppe entwickelt (Conference on Data Systems Language)

4.2.3 Relationales Datenmodell

Das relationale Datenmodell beruht auf der Struktur „Tabelle“. Eine Tabelle beschreibt eine Relation. Eine Tabellenzeile nennt man Tupel. Alle Informationen einer Datenbank werden auf die gleiche Art durch Tabellen dargestellt. Beziehungen kommen zustande, wenn ein Wert (z.B. die Personalnummer) in mehreren Relationen vorkommt. Die Darstellung von Objekten im relationalen Datenmodell ist einfach und übersichtlich. Tabellen (Relationen) können leicht in physikalische Speicherstrukturen abgebildet werden (z.B. Relation → Datei, Tupel → Datensatz). Die Durchführung von Abfragen ist jedoch relativ aufwendig, denn die Relationen unterstützen keinen schnellen Suchalgorithmus.

- geht auf Codd zurück und wird häufig aus dem ER-Modell weiterentwickelt
- es werden Objekttypen und Beziehungen sowie deren Attribute mittels Relationen abgebildet
- eine Relation kann in einer Tabelle mit einer festen Anzahl Spalten und beliebig vielen Zeilen dargestellt werden
- Schlüssel = Attribute oder Attributkombinationen, welche die Datensätze identifizieren
- 12 Regeln von Codd über Anforderungen an RDBMS
- Normalisierung
- Armstrong-Axiome

4.2.4 OODM

Objektorientierte Datenbanken sind Bestandteil des Konzepts der objektorientierten Systementwicklung. Der Grundgedanke besteht darin, Daten und Funktionen nicht nacheinander, sondern simultan zu betrachten und bei der Modellierung zu Objekten zusammenzufassen. Dabei bestehen Objekte aus Attributen, die die Eigenschaften und Methoden, die das Verhalten des Objekts beschreiben. Die Objektorientierung beruht auf den drei Grundprinzipien **Datenkapselung, Klassenbildung und Vererbung sowie Nachrichtenkommunikation und Polymorphismus**.
Ein Objekt vereinigt also (statische) Attribute mit (dynamischen) Methoden, die sich auf die Attribute anwenden lassen. Objektorientierte Datenbankverwaltungssysteme zur Speicherung und Verwaltung solcher Objekte sind in ihrer Konzeption entweder voll objektorientiert oder basieren unter der Bezeichnung hybride Datenbanken auf Erweiterungen des relationalen Datenbankmodells.

4.3 Schlüssel

4.3.1 Definition

Ein Schlüssel ist ein Element oder auch eine Kombination von Elementen eines Datensatzes zur **eindeutigen Identifizierung des Datensatzes**. Je zwei Datensätze haben verschiedene Schlüssel. Typische Beispiele für Schlüssel sind die Personalnummern in Personenkarteien oder auch die Kombination von

Name, Vorname und Geburtsdatum, falls hierdurch eindeutig eine Person identifiziert wird. Da beim Verwalten von Daten nur die eindeutige Identifikation von Datensätzen wichtig ist, werden beim Suchen und Sortieren die betrachteten Elemente meist ebenfalls als Schlüssel bezeichnet.
Der **Primärschlüssel** eines Datenmodells ist der **Schlüsselkandidat** (kurz Schlüssel) bestehend aus einem einwertigen Attribut oder Kombination von Attributen, der jede Entität eines Entitätstyps eindeutig identifiziert. Ggf. muss ein Schlüssel durch z.B. eine laufende Nummer künstlich erzwungen werden. Es kann vorkommen, dass ein Entitätstyp mehr als einen Schlüsselkandidaten hat, dann muss ein Primärschlüssel ausgewählt werden. Dieses gewählte Primärschlüsselattribut wird durch Unterstreichung gekennzeichnet. (Schlüsselattributname)

4.3.2 Schlüsselarten

Identifikationsschlüssel: ist eindeutig, d.h. es gibt keine zwei verschiedenen Objekte mit gleichem Schlüssel. Es handelt sich häufig um künstliche Attribute. Er enthält im Allgemeinen keine Informationen über Eigenschaften des betreffenden Objekts (z.B. KH-222)
Klassifikationsschlüssel: enthält Informationen über Eigenschaften der Objekte. Er ordnet die Objekte Klassen mit übereinstimmenden Eigenschaften zu. Sie sind nicht zwingend eindeutig, d.h. zwei verschiedene Objekte können den gleichen Schlüssel haben (z.B. LOS).

Verbundschlüssel: ein klassifizierender Teil eines Schlüssels wird um einen identifizierenden Teil ergänzt.
Parallelschlüssel: ein identifizierender Teil wird um einen klassifizierenden Teil ergänzt. Der identifizierende Teil ist bereits eindeutig. Der klassifizierende Teil dient lediglich als Zusatz (z.B. Zugnummern der Bahn: IC751, RB19828) Verbund- und Parallelschlüssel sind beide sprechend.

Primärschlüssel: ist das Hauptordnungskriterium und ermöglicht eine eindeutige Identifikation des Datensatzes.
Sekundärschlüssel: jede Kombination von Datenelementen, die einen Zugriff auf den zugehörigen Datensatz erlaubt.

4.4 Relationales Datenmodell

Das relationale Datenmodell wurde 1968-73 von dem Amerikaner Codd entwickelt und verwendet statt grafischer Darstellungen (siehe hierarchisches Datenmodell und Netzwerkdatenmodell) eine auf der Mengentheorie basierende mathematische Schreibweise oder eine tabellarische Darstellungsform. Alle Daten werden dabei in zweidimensionalen Tabellen mit einer festen Anzahl von Spalten und einer beliebigen Anzahl von Zeilen dargestellt. Eine Tabelle beschreibt eine Relation. Jede Zeile einer Tabelle nennt man Tupel. Alle Informationen einer Datenbank, also sowohl die Objekte wie auch ihre Beziehungen werden auf die gleiche Art durch Tabellen dargestellt. Beziehungen zwischen Objekten sind vorhanden, wenn ein Wert (z.B. Personalnummer) in mehreren Relationen vorkommt. Erst bei einer Anfrage an die Datenbank werden vorhandene Beziehungen „aufgeweckt“.
Die Darstellung von Objekten im relationalen Datenmodell ist einfach und übersichtlich. Tabellen (Relationen) können leicht in physikalische

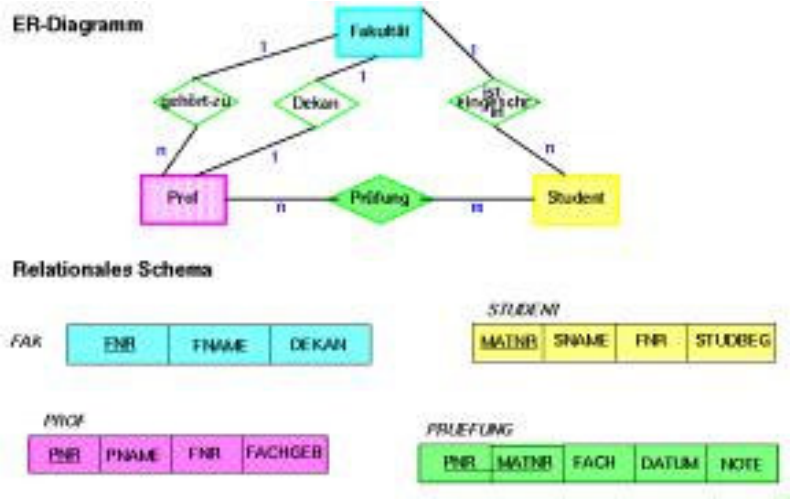
Speicherstrukturen abgebildet werden. Die Durchführung von Abfragen ist jedoch relativ aufwendig, denn die Relationen unterstützen keinen schnellen Suchalgorithmus: Entweder müssen die einzelnen Tupel sequentiell nach einem Merkmal durchsucht werden, oder der Algorithmus muss effizienzsteigernde Hilfsdatenstrukturen selbst erzeugen.

4.4.1 Grundregeln zur Darstellung von Relationen in Tabellen:

- Jede Zeile (Tupel) ist eindeutig und beschreibt ein Objekt (Entität) der Miniwelt.
- Die Ordnung der Zeilen ist ohne Bedeutung; durch ihre Reihenfolge wird keine für den Benutzer relevante Information ausgedrückt.
- Die Ordnung der Spalten ist ohne Bedeutung, da sie einen eindeutigen Namen (Attributnamen) tragen.
- Jeder Datenwert innerhalb einer Relation ist ein atomares Datenelement.
- Alle für den Benutzer bedeutungsvollen Informationen sind ausschließlich durch Datenwerte ausgedrückt.

Darstellung "relationenübergreifender" Information durch Fremdschlüssel (foreign key):

- Der Fremdschlüssel ist ein Attribut, das in Bezug auf den Primärschlüssel einer anderen (oder derselben) Relation definiert ist.
- Beziehungen werden durch Fremdschlüssel und zugehörigen Primärschlüssel dargestellt.



4.4.2 Normalisierung

4.4.2.1 Ziele der Normalisierung

- Vermeidung unerwünschter Abhängigkeiten (Einfüge-, Lösch- oder Änderungsanomalien)
- Vermeidung von Datenredundanzen
- Erzeugung klarer logischer Strukturen

4.4.2.2 Vermeidung von:

- Redundanz

Hierunter ist das mehrmalige Speichern derselben Information zu verstehen. Im Beispiel liegt eine solche bei den Attributen Ort und Entfernung vor.

- Lösch-Anomalie

Das Löschen des letzen Datensatzes eines Lieferanten aus Mannheim – weil er momentan keine Teile liefert – führt zum Verlust der Information, wie weit
- Mannheim entfernt ist.

- Änderungsanomalie

Wenn die redundant gespeicherten Daten geändert werden, ist die Wahrscheinlichkeit hoch, dass nicht alle Datensätze, die die redundante Information enthalten, geändert werden. Gerade bei Mehr-Benutzer-Systemen ist dies kritisch und führt leicht zu Konsistenz-Problemen.

- Einfüge-Anomalie:

Neue Datensätze können erst dann eingefügt werden, wenn die Informationen über alle Attribute vorliegen, auch wenn schon das Speichern von Teilen Sinn macht.

4.4.2.3 Erste Normalform

Definition:
Eine Relation $R \bullet M_1 \times M_2 \times ... \times M_n$ heißt normalisiert in der ersten Normalform (1NF), wenn die Mengen M_i selbst keine Relationen, sondern elementare Mengen sind. Anders ausgedrückt: In 1NF dürfen in jeder Relation nur atomare (nicht weiter zerlegbare) Attribute gespeichert sein.

Beispiel:

Lieferant	Bauteil	Projekt
L_1	A, B, C	P_1
L_2	A	P_1, P_2
L_3	D	P_3

Nicht in 1NF, da Zeile 1 (Bauteil) und Zeile 2 (Projekt) nicht elementar (3 bzw. 2 Elemente) sind.

Lieferant	Bauteil	Projekt
L_1	A	P_1
L_1	B	P_1
L_1	D	P_1
L_2	A	P_1
L_2	A	P_2
L_3	D	P_3

In 1NF, da keine Zeile gleich.

Bemerkung: Bei Tabellen in 1NF steht in jedem Feld höchstens ein Wert.

4.4.2.4 Zweite Normalform

Definition:
Eine Relation R • M_1 x M_2 x ...x M_n heißt normalisiert in der zweiten Normalform (2NF), wenn sie in 1NF ist und zusätzlich die Nicht-Schlüsselattribute voll funktional vom Primärschlüssel und nicht von Teilschlüsseln abhängig sind.

Beispiel:

Abteilungs#	Mitarbeiter#	Name	Standort
01	001	Duck	Entenhausen
02	002	Gaukeley	Blocksberg

In 1NF, aber nicht in 2NF, da Standort nur von Abteilungs# abhängig ist, also von einem Teilschlüssel. Um 2NF zu erreichen, ist folgende Zerlegung notwendig:

Tabelle 1 Abteilung

Abteilungs#	Standort
01	Entenhausen
02	Blocksberg

Tabelle 2 Mitarbeiter

Abteilungs#	Mitarbeiter#	Name
01	001	Duck
02	002	Gaukeley

Beide Tabellen in 2NF. Jedes Nicht-Schlüsselattribut hängt voll funktional vom Schlüssel der jeweiligen Tabelle ab.

Bemerkung: 2NF interessiert nur, wenn Primärschlüssel mehr als 1 Attribut hat.

4.4.2.5 Dritte Normalform

Definition:
Eine Relation R • M_1 x M_2 x ...x M_n heißt normalisiert in der dritten Normalform (3NF), wenn sie in 2NF ist und zusätzlich für alle Nicht-Schlüsselattribute M_a der Relation gilt: es gibt keinen Primärschlüssel von dem M_a transitiv (d.h. indirekt) abhängig ist.

Beispiel:

Vorlesung (Name, Semester, Kurs, RaumNr., Platz)

Nicht in 3NF, da „Platz“ transitiv abhängig von „Name“.

Vorlesung (Name, Semester, Kurs, RaumNr.)

Räume (RaumNr., Platz)

Bemerkung: Alle Relationen, die nur 1 Nicht-Schlüsselattribut haben, sind automatisch in 3NF.

4.4.2.6 Boyce-Codd Normalform

Definition:
Eine Relation R • M_1 x M_2 x ... x M_n heißt normalisiert in der Boyce-Codd Normalform (BCNF), wenn jedes determinierende Attribut zugleich Kandidatenschlüssel ist. Determinierend: Attribute heißen determinierend, wenn andere Attribute von ihnen funktional abhängig sind.
Kandidatenschlüssel: Die Schlüssel, die Primärschlüssel sein könnten.

Beispiel:

4.5 Entity-Relationship-Modell

Das Entity-Relationship-Modell ist ein allgemeines, computerunabhängiges Datenmodell, das vielen Anwendungen zugrunde liegt. Eine zentrale Voraussetzung für den Einsatz von Programmsystemen besteht fast immer darin, dass ein Ausschnitt der realen Welt so genau wie möglich im Rechner repräsentiert wird. Das Entity-Relationship-Modell geht davon aus, dass sich die zu modellierende „Welt“ aus genau abgrenzbaren individuellen Exemplaren von Dingen, Personen, Begriffen usw. (Entitäten, engl.: entities) zusammensetzt und dass zwischen diesen Exemplaren Beziehungen (relationships) bestehen. Heute wird dazu das Entity-Relationship-Modell von CHEN verwendet. Semantische Modelle besitzen generelle Bedeutung für die Beschreibung von Abhängigkeiten. Sie werden beispielsweise in Form semantischer Netze auch für die Wissensrepräsentation bei Expertensystemen verwendet.

Die Begriffe des Entity-Relationship-Modells wurden 1976 von CHEN eingeführt. Im Einzelnen wird beim ER-Modell wie folgt vorgegangen:

1.) Erfassung und Beschreibung der Objekte

- ein zu beschreibendes Objekt wird als Entität bezeichnet
- jede Entität weist Eigenschaften, oder charakteristische Größen (Attribute) auf, die durch Attributwerte oder Attributausprägungen beschrieben werden
- die Gesamtheit aller gleichartigen Entitäten wird unter dem Begriff Entitätstyp (oder Objekttyp) zusammengefasst.

Häufig tritt der Fall auf, dass nicht alle, sondern nur gewisse Attribute benötigt werden, um ein Entität eindeutig festzulegen. Die Kombination dieser Attribute nennt man Schlüssel. Unter allen möglichen Schlüsseln wählt man einen

sogenannten Primärschlüssel aus. Der Zugriff auf Daten z. B. in Datenbanken erfolgt oft über einen Schlüssel.
Demzufolge wird

- zunächst jedem Entitätstyp eine Kombination von Attributen zugeordnet und
- dann jede Entität durch eine Kombination von Attributwerten beschrieben.

Der Begriff Entität entspricht dem Begriff Datensatz, der Begriff Entitätstyp dem Begriff Datensatztyp.

2.) Erfassung und Beschreibung der Beziehungen

zwischen Entitäten oder zwischen Entitätstypen bestehen Beziehungen (Relationen). Gleichartige, d. h. hinsichtlich ihrer Art und der beteiligten Entitätstypen übereinstimmende Beziehungen werden zu Beziehungstypen zusammengefasst:

Beziehungen können vom Typ

- 1:1, 1:n (bzw. n:1) oder m:n sein (sogenannte Komplexität oder Kardinalität der Beziehung) (Zur Vereinheitlichung sollte man n:1-Beziehungen möglichst vermeiden, indem man sie in 1:n- Beziehungen umkehrt.) und
- als Beziehungen zwischen Entitäten verschiedenen Typs oder
- als Beziehungen zwischen Entitäten gleichen Typs, d. h. innerhalb eines Entitätstyps, bestehen.

Beziehungen zwischen Entitäten verschiedenen Typs

	1:1	1:n	m:n
Entitätstyp	Mitarbeiter	Raum	Lieferant
Entität	Bauer	Zimmer 27	Bolle Kaiser
Entität	PC K38	Bauer Müller	Butter Eier Käse
Entitätstyp	Rechner	Mitarbeiter	Ware

Beziehungen zwischen Entitäten gleichen Typs

	1:1	1:n	m:n
Entitätstyp	Mitarbeiter	Mitarbeiter	Mitarbeiter
Entität	Bauer	Boß	Dur Moll
	Urlaubsvertretung	Dienstweg	Arbeitsfluss
Entität	Müller	Groß Klein	Bach Berg Stolz
Entitätstyp	Mitarbeiter	Mitarbeiter	Mitarbeiter

3.) Grafische Darstellung der Objekte und Beziehungen

Das Entity-Relationship-Modell (ER-Modell) beschreibt die Objekte und Beziehungen grafisch im Entity-Relationship-Diagramm (ER-Diagramm). Mit der grafischen Darstellung der Datenstrukturen werden eine stärkere Visualisierung und damit eine übersichtlichere und verständlichere Beschreibung bestehender Zusammenhänge erreicht. Nachteil ist häufig der erhebliche Platzbedarf. Voraussetzung für jede grafische Darstellung ist die eindeutige Festlegung von Symbolen und Regeln für diese.

In der einfachsten Form werden im ER-Diagramm die folgenden Symbole benutzt:
- für Objekttypen: Rechtecke
- für Beziehungen: Rhomben (Rauten)
- für Attribute: Ellipsen oder Kreise an den Rechtecken und an den Rhomben.

Wichtig ist, dass auch zu Beziehungstypen (und nicht nur zu Objekttypen) Attribute gehören. Nach CHEN dürfen Objekttypen jeweils nur mit Beziehungstypen und Beziehungstypen nur mit Objekttypen in Verbindung treten. Die Komplexität wird häufig an den Rechtecken für diejenigen Objekttypen vermerkt, zwischen denen die betreffende Beziehung besteht. Nicht mehr gebräuchlich sind Bachmann-Diagramme, bei denen die Objekttypen ebenfalls durch
Rechtecke dargestellt, die Beziehungstypen aber lediglich an den Verbindungslinien zwischen den Rechtecken notiert werden.

Beim Zeichnen von ER-Diagrammen wird das auch in der Systementwicklung und Programmierung gebräuchliche Prinzip der schrittweisen Verfeinerung verwendet. Das bedeutet konkret, dass
- zuerst ein grobes ER-Diagramm und
- dann schrittweise verfeinerte Diagramme entworfen werden.

Beziehungstypen können zu Objekttypen werden. Diese Umwandlung wird grafisch durch Umranden des Rhombus mit einem Rechteck zum Ausdruck gebracht. Mit der Umwandlung wird allerdings die Forderung von CHEN missachtet, dass sich Objekt- und Beziehungstypen immer abwechseln sollen. Ein Ausweg besteht in darin zwischen dem umgewandelten Beziehungstyp und den benachbarten Objekttypen neue Beziehungstypen einzuführen.

Als **Abstraktionsmechanismen** bezeichnet man die beiden Methoden
- Generalisierung (Gegenteil: Spezialisierung) und
- Aggregation (Gegenteil: Disaggregation).

Dabei bedeutet vereinfacht ausgedrückt

- **Generalisierung**: Zusammenfassung von Objekttypen mit gemeinsamen Merkmalen zu übergeordneten Objekttypen und
- **Spezialisierung**: Zerlegung von Objekttypen in nachgeordnete Objekttypen mit unterschiedlichen Merkmalen.

4.5.1 Generalisierung

Die Generalisierung geht von unten nach oben (bottom up), die Spezialisierung von oben nach unten (top down) vor. Die dabei entstehenden Beziehungen werden grafisch durch Dreiecke mit dem Text „ist ein“ beschrieben. Die Spezialisierung entspricht der Bildung von Klassenhierarchien in der objektorientierten Systementwicklung, bei der Superklassen ihre Eigenschaft an Subklassen „vererben“.

Von Generalisierung spricht man, wenn aus zwei (oder mehr) Klassen eine allgemeinere Klasse herausgehoben wird, die die gemeinsamen Eigenschaften, der Ausgangsklassen enthält.
Von Spezialisierung spricht man, wenn aus einer allgemeinen Klasse eine spezielle Klasse abgeleitet wird, die besondere zusätzliche Eigenschaften besitzt, die die allgemeine Klasse nicht besitzt.

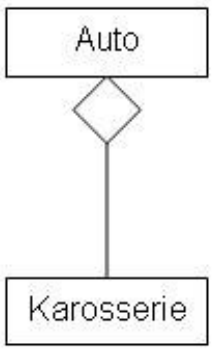
Generalisierung und Spezialisierung

4.5.2 Aggregation

Die Aggregation drückt aus, dass ein Element Bestandteil eines anderen ist (engl.: part-of relationship). Die Objektaggregation drückt demgemäss aus, dass ein Objekt Bestandteil eines anderen Objekts ist.

Aggregation

Auto „besteht aus“ Karosserie



Karosserie „ist Teil von“ Auto

4.6 SQL

In der Praxis existieren Sprachen, mit denen das gesamte Datenbanksystem erstellt und bearbeitet werden kann. ***SQL (structured query language)*** ist eine deskriptive und mengenorientierte Sprache. Alle Ergebnisse einer Abfrage werden gleichzeitig geliefert und nicht nacheinander zur Verfügung gestellt. Sie

vereint in sich Funktionen einer Datenbeschreibungssprache und einer Datenmanipulationssprache. Mittels Listengeneratoren werden die Ein- und Ausgabeprogramme auf relativ einfache Weise erzeugt. Für die Abfrage schafft das SQL-Modul einen direkten Zugriff auf die Daten. Schließlich muss das Programm für die zweckmäßige Speicherung der eingegebenen Daten sorgen. Der Anwender teilt dem Programm die Datenlogik mit, der Anwender formuliert seine Abfragen. Wie die Daten zu speichern sind und welche Suchstrategie am besten anzuwenden ist, das ist Sache des Datenbankprogramms.
SQL unterstützt nur das relationale Datenbankmodell, die Daten werden in Tabellen gespeichert. Die Bearbeitung der Tabelle kann durch eine mathematische Theorie, die Relationenalgebra, beschrieben werden. Beim Arbeiten mit relationalen Sprachen werden Ergebnistabellen erzeugt, indem Auszüge aus bestehenden Tabellen gebildet oder vorhandene Tabellen miteinander verknüpft werden. Relationale Sprachen sind mengenorientiert, das heißt mit einer Anweisung können mehrere Sätze aus einer oder mehreren Tabellen gespeichert, ausgegeben, verändert oder gelöscht werden.

Die Datenbanksprache SQL dient im Wesentlichen der
- Datendefinition (anlegen, ändern, löschen von Tabellen)
- Datenmanipulation (einfügen, ändern, löschen, suchen) und
- Zugriffskontrolle (Datenschutz, Datensicherheit).

4.6.1 Datenbankdefinition (DDL – Data Definition Language)

SQL geht davon aus, dass die Befehle implementiert sind und für den Benutzer das Ergebnis das Wichtigste ist. So wird in den SQL-Anweisungen jeweils nur angegeben, *was* zu tun ist, nicht aber *wie*. Häufig spricht man daher von einer Sprache der 4.Generation (4GL). Bevor in der Tabelle einer Datenbank Daten gespeichert werden können, muss dem System die Struktur der Tabelle mitgeteilt werden. Eine solche Strukturdefinition enthält die Namen der Tabellenspalten sowie die Informationen, ob die Spalte Texte, Zahlen oder Datumswerte speichern soll und wie viel Daten die Tabellenspalte jeweils aufnehmen soll. Bei Schlüsselspalten ist es üblich, mit der Schlüsselbedingung NOT NULL festzulegen, dass in diese Spalte unbedingt Werte eingetragen werden müssen.

Mit der SQL-Anweisung <CREATE TABLE [Benutzername].tabellenname> wird eine neue Tabelle angelegt. Hinter dem Tabellennamen sind in Klammern die Spaltennamen jeweils mit Datentyp und Datenlänge zu definieren. Die einzelnen Spaltenbezeichnungen werden durch Kommata getrennt. Die Liste der Spaltennamen wird mit runden Klammern umschlossen. Zusätzliche Bedingungen, z.B. die Mindestbelegungsdichte (PCTFREE/PCTUSED), der Primär- und Fremdschlüssel (PRIMARY KEY und FOREIGN KEY REFERENCES) und Default-Werte
(DEFAULT) können durch Erweiterung des Befehls festgelegt werden. Die Syntax für das Erstellen einer einfachen Tabelle lautet:

```
CREATE TABLE ueb
(
        uebnr           INTEGER,
```

```
        uebbez      CHAR(20)
);
```

Die Tabellen können im laufenden Betrieb nur vom Besitzer oder vom Systemadministrator gelöscht werden. Mit der Anweisung <DROP TABLE tabellenname> wird eine Tabelle mit den darin gespeicherten Daten aus der Datenbank und in den Systemtabellen gelöscht. Mit <TRUNCATE TABLE tabellenname> wird nur der Tabelleninhalt gelöscht, die Struktur bleibt erhalten. Mit dem Befehl <RENAME tabellenname TO tabellenname_neu> können bestehende Tabellen umbenannt werden. Sämtliche Verweise auf die Tabelle zeigen danach auf den neuen Namen. Mit dem Befehl <ALTER TABLE tabellenname MODIFY/ADD/DROP> kann im laufenden Betrieb die Tabellenstruktur verändert werden.

4.6.2 Datenmanipulation (DML – Data Manipulation Language)

Für die ***Eingabe von Daten*** in die Tabelle gibt es verschiedene Möglichkeiten:

- Mit der Anweisung <INSERT INTO tabellenname VALUES()> kann man die Werte zeilenweise eingeben:

```
INSERT INTO ueb VALUES (10, 'Handbuch');
INSERT INTO ueb VALUES (20, 'Sachbuch');
```

- Mit dem SQL-Befehl <INSERT INTO tabellenname SELECT-Anweisung> kann man Werte aus einer Tabelle in eine andere übertragen:

```
INSERT INTO ueb
        SELECT p.persnr, mgehalt, gehalt * 3.6
        FROM persdat p mgehalt m
        WHERE p.persnr = m.persnr ;
```

Datenabfragen (englisch: queries) werden mit der Anweisung <SELECT spaltennamen/aggregatsfunktion FROM tabellennamen> ausgeführt und in einer Ergebnistabelle zur Ausgabe übergeben. Zusätzlich können wir noch Bedingungen für die Auswahl der Datensätze treffen. Ein Beispiel hierfür ist:

```
SELECT pname
FROM persdat
WHERE pname > 'M' ;
```

Die wichtigsten **Aggregatsfunktionen** sind:

- der Minimalwert (MIN (spaltenname)),
- der Maximalwert (MAX (spaltenname)),
- die Anzahl der Zeilen [mit unterschiedlichem Wert] (COUNT (* / [DISTINCT spaltenname])),
- die Summe (SUM (spaltenname)) und
- der Durchschnitt der Werte (AVG (spaltenname)).

Funktionen für die Verknüpfung verschiedener WHERE-Bedingungen sind: AND (logisches Und), OR (logisches Oder) und NOT (Negation).

Weitere Möglichkeiten für die Strukturierung von Befehlen sind:

- Verknüpfung von mehreren Tabellen über das kartesische Produkt

  ```
  SELECT a.adresse, b.alter
  FROM a, b                    => kartesisches Produkt der Tabellen a und b
  WHERE a.name = b.name ;      => Auswahl der Verknüpfungen, die angezeigt
  werden sollen
  ```

- die Gruppierungen von Daten

  ```
  SELECT abteilung, name, gehalt
  FROM c
  GROUP BY abteilung, name ;        => Einfügen von Gruppenwechseln
  ```

- und das Einfügen von Unterabfragen

  ```
  SELECT name, adresse
  FROM a
  WHERE name IN ( SELECT name
                  FROM b
                  WHERE alter >= 50 ) ;
  ```

Zusätzlich zu diesen Befehlen gibt es eine Reihe von zusätzlichen Anweisungen. Jeder Hersteller, der SQL für sein Datenbankprogramm verwendet, fügt noch zusätzliche Befehle hinzu. Diese Befehle sind in verschiedenen Fachbüchern und den jeweiligen Onlinedokumentationen nachzulesen.
Mit der Anweisung <UPDATE tabellenname SET> werden ***Daten bedingungsabhängig geändert***. Im Anschluss an das Schlüsselwort UPDATE wird die Tabelle genannt, in der die Änderungen durchgeführt werden sollen. Hinter <SET> werden den zu ändernden Spalten die neuen Werte zugewiesen, hierfür können auch Rechenoperationen verwendet werden. Die Bedingung hinter dem Zusatz WHERE definiert die Zeilen, in denen die Änderungen durchgeführt werden sollen. Wird beim UPDATE-Befehl das WHERE weggelassen, werden alle Zeilen der Tabelle geändert.

```
UPDATE ueb
    SET uebnr = uebnr + 10 ;
```

Für das ***Löschen von Datensätzen*** aus einer Tabelle verwenden wir den Befehl <DELETE FROM tabellenname>. Mit dem Zusatz <WHERE bedingung> erfolgt wiederum die Zeilenauswahl. Wird der Befehl ohne WHERE-Zusatz eingegeben, werden alle Zeilen der Tabelle gelöscht.

4.6.3 Zugriffskontrolle und Zugriffsrechte (DCL – Data Control Language)

Wenn die in Datenbanken gespeicherten Informationen mehreren Benutzern zur Verfügung stehen, muss der Datenbankadministrator durch geeignete Maßnahmen den Zugriff von nicht autorisierten Personen verhindern. Um die Gefahr des Datenverlustes, der Fälschung und des Datenmissbrauchs einzuschränken, erteilt der Benutzer Zugangsberechtigungen (subjektbezogener Datenschutz) und Zugriffsberechtigungen (objektbezogener Datenschutz).

Zugriffsberechtigungen werden mit dem Befehl GRANT erteilt:

```
GRANT ALL/ALTER/...
        ON tabellenname/viewname
        TO PUBLIC/Benutzer/...
```

Zugangsberechtigungen werden ebenfalls mit GRANT erteilt, wobei es hier verschiedene Stufen gibt:

- Mit GRANT **CONNECT** TO PUBLIC/Benutzer wird das Recht, bestehende Tabellen abzufragen, zu ändern und zu löschen erteilt, sofern das erforderliche Tabellenzugriffsrecht besteht.

- Durch GRANT **RESSOURCE** TO PUBLIC/Benutzer können Basistabellen erstellt, bearbeitet und gelöscht sowie Tabellenzugriffsrechte vergeben werden

- Der Befehl GRANT **DBA** TO PUBLIC/Benutzer erlaubt Benutzern, Tabellenzugriffsrechte zu vergeben, Datenbanken zu löschen und Datenbankzugriffsrechte zu gewähren.

Durch die Erweiterung des Befehls durch <IDENTIFIED BY passwort> werden Benutzerkennungen ermöglicht.

4.7 Transaktion

Definition: Bündelung mehrerer Datenbankoperationen, die in einem Mehrbenutzesystem ohne unerwünschte Einflüsse durch andere Transaktionen als Einheit fehlerfrei ausgeführt werden.

a) Für den Teilhaberbetrieb ist die Bezeichnung Transaktionsverarbeitung (OLTP = Online Transaction Processing) gebräuchlich. Jeder Benutzerauftrag wird als Transaktion (Der Begriff der Transaktion wird in der Informationsverarbeitung mehrfach verwendet, z.B. auch für Zustandsänderungen in Datenbanken) bezeichnet, die einzelnen Anwendungssysteme heißen Transaktionssysteme.

b) Datensicherheit und Datenintegrität gehen vom Begriff der Transaktion aus. Darunter versteht man den Übergang der Datenbank von einem in einen anderen konsistenten Zustand. Inkonsistenz kann hauptsächlich
- durch den gleichzeitigen Zugriff mehrere Benutzer oder

- durch Störungsfälle wie z.B. Programmfehler oder Systemzusammenbrüche auftreten.

4.8 Transaktionssynchronisation

Grund : Bei parallelen Zugriff zweier Transaktionen auf ein Objekt kann es zu Datenverlust kommen. Man unterschiedet hierbei zwei Probleme.

- ***lost update*** : Lesen des Objekts durch die Transaktionen A und B, A und B verändern den Wert, A speichert den neuen Wert, B speichert den neuen Wert => die Veränderung durch A wird überschrieben und geht somit verloren.
- ***inkonsistent analysis*** : Eine Leseoperation A liest ein Objekt und führt damit eine Auswertung durch. Während der Verarbeitung durch A, wird der Wert durch die Transaktion B verändert. Die Auswertung am Ende der Transaktion A entspricht somit nicht mehr dem aktuellen Stand und ist falsch.

Transaktionsschedule ist der zeitliche Ablauf der Operationen mehrere Transaktionen. Ein Schedule ist serialisierbar, wenn ein paralleler Ablauf der Transaktionen, das gleiche Ergebnis liefert, wie eine serieller Ablauf.

Konfliktsituation : Zwei Operationen aus verschiedenen Transaktionen stehen in Konflikt, wenn sie auf ein Objekt mit mind. einem Schreibbefehl zugreifen. Diese Situation läßt sich durch den Zyklischen Graph darstellen/erkennen.

Lösung des parallelen Zugriffs

Sperrmechanismus : Ein Objekte wird von einer Transaktion reserviert/gesperrt. Zwei Arten : Schreibsperre (exklusives Sperren)

- Lesesperre

4.9 Eigenschaften von Transaktionen (ACID)

Wir haben mit der vorhergehenden Diskussion (hoffentlich) schon ein intuitives Verständnis des Transaktionsbegriffs erzielt. Die Eigenschaften des Transaktionskonzepts werden oft unter der Abkürzung A*CID* zusammengefasst. Das sogenannte *ACID-Paradigma* steht dabei für vier Eigenschaften:

Atomicity (Atomarität) Diese Eigenschaft verlangt, daß eine Transaktion als kleinste, nicht
mehr weiter zerlegbare Einheit behandelt wird, d.h. entweder werden alle Änderungen der
Transaktion in der Datenbasis festgeschrieben oder gar keine. Man kann sich dies auch als
"alles-oder-nichts"-Prinzip merken.

Consistency Eine Transaktion hinterläßt nach Beendigung einen konsistenten Datenbasiszustand. Anderenfalls wird sie komplett (siehe At*omarität)* zurückgesetzt
Zwischenzustände, die während der TA-Bearbeitung entstehen, dürfen inkonsistent sein, aber der resultierende Endzustand muß die im Schema definierten Konsistenzbedingungen (z.B. referentielle Integrität) erfüllen.

Isolation Diese Eigenschaft verlangt, daß nebenläufig (parallel, gleichzeitig) aus geführte Transaktionen sich nicht gegenseitig beeinflussen. Jede Transaktion muß - logisch gesehen - so ausgeführt werden, als wäre

sie die einzige Transaktion, die während ihrer gesamten Ausführungszeit auf dem Datenbanksystem aktiv ist. Mit anderen Worten, alle anderen parallel ausgeführten Transaktionen bzw. deren Effekte dürfen nicht sichtbar sein.
(Mehrbenutzersynchronisation)
Durability (Dauerhaftigkeit): Die Wirkung einer erfolgreich abgeschlossenen Transaktion bleibt dauerhaft in der Datenbank erhalten. Die Transaktionsverwaltung muss sicherstellen, dass dies auch nach einem Systemfehler (Hardware oder Software) gewährleistet ist. Die einzige Möglichkeit, die Wirkungen einer einmal erfolgreich abgeschlossenen Transaktion ganz oder teilweise aufzuheben, besteht darin, eine andere sogenannte kompensierende Transaktion auszuführen. (Recovery)

4.10 Recovery

Aufgabe des Recovery ist die Wiederherstellung eines konsistenten Datenbestandes nach Programmfehler und nach Systemausfällen. Man unterscheidet drei Klassifizierungsarten des Recovery:
- Benutzer- und Programmierfehler
- Systemfehler in Form des verlustes von Arbeitsspeicherinhalten
- Speicherfehler durch Schaden am dauerhaften Speichermedium

Daraus resultieren folgende Fehlerklassen:

Fehlerklassen	Massnahmen
Transaktionsfehler	Rücksetzen von Transaktionen (Rollback)
Systemfehler	Neustart (Instance-Recovery)
Speicherfehler	Rekonstruktion (Media-Recovery)

Zur Wiederherstellung der Datenkonsistenz wird meistens ein Logfile verwendet. Im Logfile werden alle Zugriffe auf die Datenbank mitgeschrieben. Unter anderem werden darin Beginn- und Endmarken einer Transaktion gespeichert, sowie bestimmte Sicherungspunkte
„checkpoints“, an denen sich die Datenbank in einem konstistenten Zustand befunden hat und die Inhalte der bearbeiteten Objekte vor und nach einer Operation („before-Image“ und „after- Image“).

Marke Beginn Transaktion

Before Image	After Image
- Identifikation des Objektes	- Identifikation Objekt
- Alter Wert des Objektes	- Neuer Wert des Objekts
- Identifikation des Objektes	- Identifikation des Objekts

Marke Ende Transaktion

4.10.1 Recovery-Maßnahmen

1. Transaktionsfehler:
Nicht vollendete Transaktionen werden zurückgesetzt (Rollback)
2. Systemfehler
-> Systemneustart. Inhalt des Arbeitsspeichers und der Buffer sind verloren. Abgeschlossene, noch nicht zurückgeschriebene Transaktionen sind zu retten. Dies wird durch Aufsetzen auf dem letzen Checkpoint ermöglicht.
REDO – alle laut Logfile abgeschlossenen Transaktionen werden mit Hilfe der „After-Images“ der Objekte nachgefahren; beim Systemausfall noch nicht beendete Transaktionen werden wie Transaktionen durch „Rollback“ rückgängig gemacht.

4.10.2 Rekonstruktion

Datenbank ist infolge Speicherfehler nicht mehr lesbar (z.B. head-crash)
Aufsetzen auf Komplettsicherung/Backup. Davon ausgehend werden mit Hilfe der im Logfile verzeichneten „after-images“ alle bereits abgeschlossenen Transaktionen wiederholt.
Um nach Störanfällen den korrekten Zustand der Datenbank wieder herzustellen, sind folgende Maßnahmen (recovery) möglich:
before image:
Anfertigung einer Kopie der zu ändernden Daten unmittelbar vor der Änderung
after image:
Anfertigung einer Kopie der geänderten Daten unmittelbar nach der Änderung

Schatten Datenbank:
Kopie der Orginal-Datenbank. Eine zerstörte Orginal-Datenbank kann mit Hilfe der Schatten-
Datenbank und sämtlicher nach dem letzten Anlegen der Schatten-Datenbank angefallenen after image Kopien rekonstruiert werden (roll forward-Technik).

4.11 Sperrprotokolle

Das 2-Phasen-Sperrprotokoll wird in kommerziellen Systemen am häufigsten verwendet und dient der Datensicherheit. Es funktioniert wie folgt: Verschiedene Transaktionen können Schreib- und Lesesperren auf Objekte setzen. Sollte ein Objekt mit einer Lesesperre versehen sein, so sagt man, daß es nicht exklusiv gesperrt ist. Andere Transaktionen können ebenfalls Lesesperren auf dieses Objekt setzen. Es ist im sogenannten „shared mode“. Ist auf ein Objekt eine Schreibsperre gesetzt, so sagt man, daß es exklusiv gesperrt ist, sich im sogenannten
„exclusive mode“ befindet. Andere Transaktionen können jetzt weder Lese- noch
Schreibsperren auf dieses Objekt setzen.

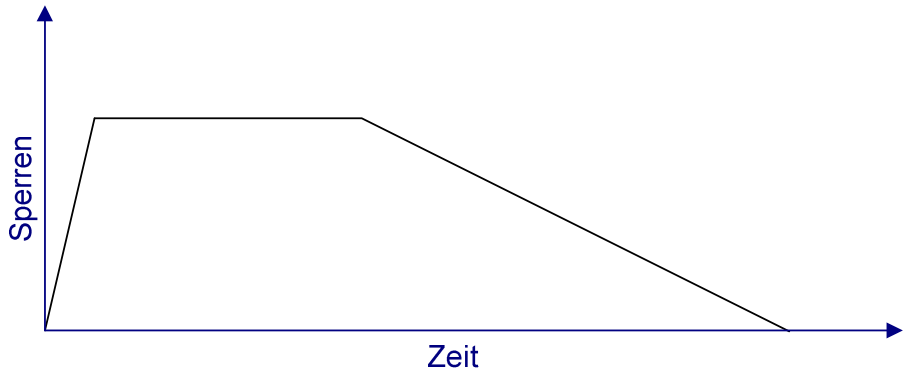
4.11.1 Zwei-Phasen-Sperrprotokoll

Wachstumsphase : Die einzelnen Objekte werden nach und nach unmittelbar vor der Verwendung gesperrt. Während dieser Phase dürfen keine Objekte freigegeben werden. ***Schrumpfungsphase*** : Die Objekte werden nach und nach freigeben. Hier dürfen keine Objekte mehr gesperrt werden.
Nachteil : Deadlock-Situation T1 sperrt A und wartet auf B. T2 sperrt B und

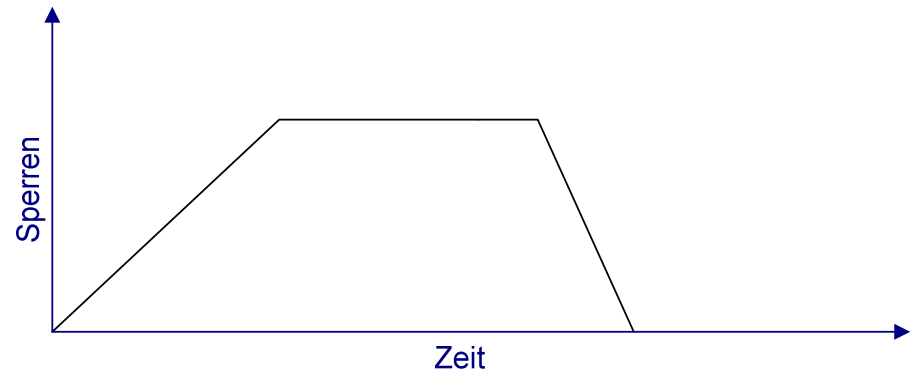
wartet auf A.
Preclaiming : Alle Objekt werden gesperrt und erst anschließend beginnt die Verarbeitung. Freigabe erfolgt nach und nach.
Nachteil : -Objekte werden länger gesperrt als notwendig.
-Fehler bei Fortpflanzung der Transaktion. T1 gibt A frei und läuft weiter, T2 verwendet A, T2 ist fertig, T1 bricht ab. => auch T2 muß zurückgesetzt werden, da die Veränderung an A ungültig ist.
Sperrprotokoll bis EOT (End of Transaction) : Objekte werden nach und nach gesperrt, und auf einmal freigegeben .
Nachteil : Deadlock durch Wachstumsphase
Vorteil : keine Fortpflanzung von Transaktionen.
Optimal : Kombination aus Preclaiming und EOT – aber dadurch langes Sperren => schlechte Performance.

Varianten des 2-Phasen-Sperrprotokolls (2PL):
Es gibt ein konservatives 2PL, das C2PL, bei dem eine Transaktion zu Beginn ihrer Ausführung alle von ihr benötigten Sperren setzt. Der Vorteil hierbei besteht darin, daß Deadlocks vermieden werden. Das Problem allerdings besteht darin, daß jede Transaktion zu Beginn dem Scheduler ihr komplettes Read-Set und Write-Set deklarieren muß.

Folgendes Schaubild veranschaulicht das C2PL:



Alternativ spricht man von einem strengen 2PL (S2PL), falls alle von einer Transaktion gehaltenen Sperren erst nach deren letzten Schritt aufgehoben werden. Hält ein S2PL alle Sperren bis zum Ende der jeweiligen Transaktion, und nicht nur die Schreibsperren, so spricht man von einem starken S2PL (SS2PL). Dabei ist allerdings der erzielbare „Grad an Parallelität“ zugunsten einer hohen Fehlersicherheit stark eingeschränkt.



4.12 ODBC

ODBC heißt **Open Data Base Connectivity** und beschreibt einen Microsoft QuasiStandard unter Windows. Bei der Verfügbarkeit entsprechender Treiber - ist es damit möglich, auf jede beliebige Grossrechner- und Mikrocomputer-Datenbank zugreifen zu können.

Um Anwendungssysteme, die auf Datenbanken zugreifen, unabhängig von dem jeweils eingesetztem Datenbankverwaltungssystem entwickeln und damit portabel gestalten zu können, wurden bereits in den Mitt70ern die kompatiblen Datenbankschnittstellen (KDBS) definiert. Dabei handelt es sich um systemnahe Software, die als zusätzliche Schicht zwischen Betriebssystem und Anwendungssoftware gelegt wird, der sogenannten Middleware. Diese Middleware fungiert als Übersetzungsprogramm (Treiber) zwischen der jeweiligen Datenbank und dem Anwendungsprogramm. Der Anwender spricht auch von virtuellen Datenbankschnittstellen.

Wie oben schon angesprochen, hat Microsoft hierfür einen eigenen Standard unter Windows, das ODBC.

Die ODBC- Architektur unterscheidet **vier Ebenen:**

1. als oberste Ebene das Anwendungssystem, das SQL- Befehle auslöst,
2. darunter ein Steuerprogramm (driver manager), das die erforderlichen Übersetzungsprogramme (driver) lädt
3. wiederum darunter die Übersetzungsprogramme für alle von dem Anwendungssystem angesprochenen Datenbanksysteme und
4. als unterste Ebene die Datenbanken selbst, die jeweils aus einem Datenbankverwaltungssystem und einem Datenbestand bestehen.

In Zusammenhang mit Microsoft Query können in Microsoft Excel Zugriffe auf Dateien und
Tabellen in externen Datenbanken ermöglicht werden.

Ferner können mittels ODBC-Add-In für Excel Makros geschrieben werden, die einen direkten Zugriff auf Daten in externen Datenquellen ermöglichen, ohne Microsoft Query zu benutzen.

4.12.1 ODBC-Schnittstelle

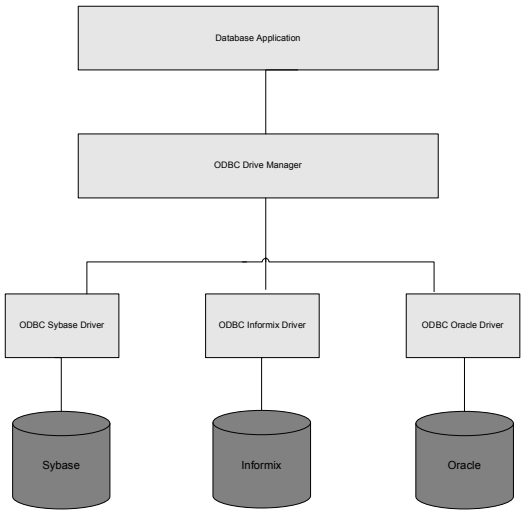
Eine ODBC-Schnittstelle definiert:

- eine Bibliothek mit ODBC-Funktionsaufrufen für
  - die Verbindung mit der Datenquelle
  - die Ausführung von SQL-Anweisungen
  - den Abruf von Ergebnissen
- eine Standardmethode zur Herstellung einer Verbindung und Anmeldung bei der Datenquelle

˜ eine SQL-Syntax
˜ eine standardisierte Darstellung für Datentypen
- eine standardisierte Gruppe von Fehlercodes

4.12.2 Aufbau von ODBC

ODBC ist in vier Schichten aufgebaut.



ODBC-Application:

♦ führt mittels ODBC-Calls SQL-Statments aus und erhält die Ergebnisse der Anfrage

ODBC-Driver Manager

♦ benutzt Datei (odbc.ini), um den erforderlichen Treiber für die Applikation zu aktivieren
♦ verarbeitet ODBC-Calls
♦ bietet Parameterprüfung für ODBC-Calls

ODBC-Driver

♦ stellt Verbindung zur Datenquelle her
♦ schickt Anfragen an die Datenquelle
♦ wandelt Daten in andere Datenformate um
♦ liefert Ergebnisse der Anfrage an die Applikation
♦ formatiert Fehlermeldungen des DBMS in ODBC-Standardfehlermeldungen um
♦ initialisiert und verwaltet Aktionen des Cursors

Data Source (Datenquelle)

♦ spezieller Vertreter eines DBMS in Kombination mit einem Betriebssystem und/oder
Netzwerk, welches notwendig ist um auf das DBMS zuzugreifen

4.13 Replikation

Synchronisation von Daten, die auf mehreren Servern gleichzeitig gelagert sind und dort verändert werden.

Dieser Zustand, der dadurch entsteht wird in der Informatik als "Redundanz" bezeichnet: eine Information ist mehrfach im System vorhanden.
In Zeiten vor der Einführung der Datenbanktechnologie war Redundanz ein weitverbreitetes Problem.
Gleiche Information war in mehreren Dateien vorhanden und bei einem Update (z.B. Namensänderung) mussten alle Dateien geändert werden. Wird eine Änderung vergessen entsteht Inkonsistenz, d.h. das System arbeitet nicht mehr korrekt.

Lösung und Vorteile der redundanzfreien Informationshaltung(zentrale Datenhaltung)

- ♦ redundanzfreie Darstellung der Info (jede Info existiert genau 1mal, alle anderen Stellen erhalten lediglich Verweise).
- ♦ einfache Verwaltung.
- ♦ Verteilung von Daten, Formularen und Programmen vom Server zu den Clients
- ♦ Einsatz beim Rollout von neuen Anwendungen (Betaversionen)
  - Kosteneinsparungen
  - einfach
- ♦ Synchronisation in der Master- DB
- ♦ Update- Problem wird durch "konfigurierbaren Regel" gelöst; Verlierer-Dokument wird als

Konflikt- Dokument gespeichert

Probleme:

- ♦ hohe Performance-Probleme. Greifen viele Anwendungen auf immer gleiche Schlüsselwerte zu, entsteht ein „hot spot“, um dessen Zugriffsrecht sich alle rangeln. Es kommt zu Warteschlangen (=> lange Übertragungszeiten). => Lösung: Daten von der zentralen Datenbank holen und lokal weiter arbeiten
- ♦ Aktualität der Daten (bei der Aktualisierung müssen alle Daten neu runtergeladen werden

  => jegliche Veränderungen in der PC- DB gehen verloren
- ♦ Update- Konflikt bei der Synchronisation (wenn in 2 Repliken das gleiche Dokument zur gleichen Zeit geändert wurde)

Das Prinzip der redundanzfreien Information ist die Kernregel im Design zentraler Datenhaltung. Dies gilt sowohl für die hierachischen als auch für die relationalen Datenbanksysteme. Auch bei verteilter Datenhaltung existiert eine Information nur einmal und nur an einem Ort.
Unter einer verteilten DB versteht man ein DB-system, bei dem ein logisch zusammengehöriger Datenbestand physisch auf mehrere Rechner eines Rechnernetzes verteilt werden. Der Hauptgrund, eine DB verteilt anzulegen, ist der organisatorische Aspekt, die Daten unmittelbar an den Stellen zu speichern, an denen sie am stärksten benutzt werden. Die Aufteilung des Datenbestands wird als Partitionierung bezeichnet. Die herbeigeführte Redundanz durch mehrfache Speicherung derselben Daten an verschiedenen

Stellen heißt Replizierung.

Vorteile sind: schnellerer Zugriff und größere Sicherheit gegenüber Ausfällen. Nachteil:
erhöhter Pflegeaufwand für Aktualisierungen des Datenbestands. Für den Datenabgleich gibt es 2 Möglichkeiten:

- Zweiphasen-Commit-Verfahren: veranlaßt den Zentralrechner (bzw. Datenbankserver), daß jede Änderung sofort in allen Rechnern mit redundanten Daten vorgenommen wird.
- Replikationsverfahren: alle Änderungen werden zunächst über einen festgelegten Zeitraum in einem dafür eingerichteten Rechner des Netzes zwischengespeichert. Dieser Rechner übernimmt später, d.h. zeitversetzt, die Aktualisierung in allen beteiligten Rechnern.

4.14 Synchrone Replikation

die Aktualisierung der Kopien ist direkt an die Transaktion, die mit den Originaldaten ausgeführt wird, gekoppelt, d. h. die Kopien sind immer auf dem aktuellen Stand. Dieses Konzept wird mit
Hilfe von Triggern realisiert. Die Koordination erfolgt nach dem **Zwei-Phasen-Commit- Protokoll**. Ablauf:

- der auslösende Knoten ist Koordinator
- er zerlegt die Transaktion in Sub-Transaktionen und verteilt diese auf die anderen Knoten
- diese erhalten das Signal "prepare to commit"
- wenn alle mit "ready to commit" bestätigt haben, wird systemweit "commit" ausgeführt

Vorteil: immer alle Daten aktuell und konsistent
Nachteile:

- hohe Netzwerkbelastung, wenn viele Transaktionen ausgeführt werden.
- schlechtere Performance des DBMS
- Probleme bei Ausfall eines Knotens

4.14.1 Asynchrone Replikation

(auch "store & forward replication"): Transaktion und Aktualisierung der Kopien sind voneinander getrennt (Separationsprinzip), die Aktualisierung erfolgt "sobald wie möglich". Asynchrone Replikation kann realisiert werden durch

- Auswertung des Logfiles (**logbasierend**): Änderungen an replizierten Objekten werden herausgesucht und an die Knoten der Replikate weitergegeben, die diese dann abarbeiten.
- Vorteil: kein Performanceverlust bei Transaktionen
- Einsatz von Triggern (**triggerbasierend**): bei Änderung eines replizierten Objekts wird ein Trigger ausgelöst, der die Änderung in eine separate Tabelle schreibt, die dann von den Knoten der Replikate abgearbeitet wird.
- Vorteil: die meisten DBMS unterstützen Trigger
- Nachteil: schlechtere Performance
- periodischen Vergleich von Versionsnummern der Kopien anhand einer Liste, Differenzen werden ausgeglichen.

Vorteil: Netzwerkverbindung muß nicht permanent sein (auch für Außendienst einsetzbar)
Nachteile:

- Versionsnummer erforderlich
- Bei vielen Objekten dauert der Vergleich lange.

4.15 Data Warehouse – Konzept / OLAP

4.15.1 Hintergrund

- In den meisten Unternehmen werden heutzutage operationale Systeme zur Automatisierung der Geschäftsprozesse (z.B. Gehalts-, Lohnabrechnung) eingesetzt
- Kennzeichnend für diese Systeme sind folgende Merkmale:
  - Zugriffe auf mittlere bis größere Datenbestände
  - Zugriffe finden vorwiegend als OLTP (Online Transaction Processing) statt
  - In den meisten Fällen finden aktualisierende Zugriffe statt
  - Meist geringe Datenmengen pro Transaktion
  - Komplexe Integritätsregeln müssen eingehalten werden
  - Daten werden i.d.R. überschrieben -> Historie wird nicht gespeichert

OLTP Systeme sind für gleichmäßige Benutzung und Belastung optimiert und somit nicht für die im Rahmen von Analysen auftretenden kurzzeitigen Höchstbelastungen geeignet

- Weiterhin sind keine adhoc – Abfragen möglich (bzw. nur unter sehr großem Aufwand); diese sind jedoch für ein Analysesystem zwingend notwendig
- Aufgrund der zahlreichen Nachteile wurde das **Data Warehouse – Konzept**, das auch **OLAP (Online Analytical Processing)** bezeichnet wird, entwickelt

4.15.2 Was ist ein Data Warehouse?

- Ein Datawarehouse ist ein „Lagerhaus“ für Daten, das vorhandene Datenbestände sammelt, verwaltet und bei Bedarf für analytische Bearbeitung und Auswertung in optimierter Form zur Verfügung stellt
- Nach Immon kann man Datawarehouses folgerndermassen beschreiben:
  - subjektorientiert (Daten werden nicht für einen speziellen Anwender gemacht)
  - ermöglichen die Integration verschiedener Daten (aus unterschiedlichen Benutzersystemen)
  - berücksichtigen das Prinzip der “Nicht – Violaität” (Daten im Data Warehouse werden nicht überschrieben, es sei denn, es sind fehlerhafte Einträge gemacht worden)
  - ermöglichen eine Betrachtung von Daten über einen größeren Zeitraum hinweg (Entwicklungen, Trends etc.)

4.15.3 *Mehrdimensionale Datenmodelle*

♦ Daten, die in eine Hierarchie eingeordnet werden können, werden über **Dimensionen**

beschrieben; Die Dimension ‚Zeit' kann z.B. einen Wert wie ‚Umsatz' in der Hierarchie
„Jahr-Quartal-Monat-Woche-Tag-Stunde" repräsentieren

♦ Durch die Einführung weiterer Dimensionen (z.B. Umsatz – Zeit – Region) wird zuerst ein **dreidimensionales Datengebilde** in Form eines Würfels erzeugt. Werden weitere Dimensionen ergänzt, spricht man von einem **multidimensionalen** oder **n-dimensionalen Datenmodell**

4.15.4 Star – Modell -> relationaler Ansatz zur Umsetzung von Mehrdimensionalität

♦ Im Mittelpunkt steht hierbei eine **Faktentabelle**, die den Dimensionen mit ihren beschreibenden Attributen und Ausprägungen zugeordnet wird

♦ Die Faktentabelle fungiert somit als eine Art „Fremdschlüsseltabelle", durch welche
Beziehungen zwischen Dimensionen beschrieben werden

♦ Aus der sternförmigen Anordnung der Dimensions – Tabellen um die Faktentabelle hat sich der Name des Modells ergeben

♦ ***Aktionen in einem mehrdimensionalen Datenmodell***
Ausgehend von einem imaginären Datenwürfel sollen dem Benutzer in OLAP Systemen folgende „Bewegungen" auf dem Modell möglich sein:

- ♦ **Drill Down**: Konsolidierte Ergebnisse können Schritt für Schritt in eine untergeordnete Dimension zerlegt werden
- ♦ **Roll Up**: Aggregation von Daten auf die nächst höhere Ebene
- ♦ **Slicing**: Isolierung einzelner Schichten oder Scheiben eines Würfels
- ♦ **Dicing**: Rotieren eines Würfels, um verschiedene Seiten sichtbar zu machen
- ♦ **Privoting**: Vertauschen von Zeilen- und Spaltendimensionen

4.15.5 Elemente von Data Warehouses

4.15.5.1 Datenreplikaktionsmanager

♦ verwaltet das Kopieren (/Importieren) und Verteilen der internen und externen Daten auf
Datenbanken

♦ interne Daten: Daten aus den verschiedenen Systemen innerhalb des Unternehmens

♦ externe Daten: öffentlich verfügbare Daten (z.B. aus anderen Firmennetzen, Internet, Fremdsystemen), die für das analytische Auswerten der Gesamt-Unternehmensdaten herangezogen werden können

4.15.6 Informationsdatenbank (Data Dictionary / Repository)

♦ Metadaten

- Dimensionen (Verdichtungsstufen, Hierarchien, etc.) Angaben über die Beziehungen von Daten, sich ändernde Verknüpfung im Roll-Up- bzw. Drill-Down-Fall)
- bestimmte redundant gespeicherte Daten aus Performance – Gründen, um im Fall einer Analyse nicht jeweils auf der absolut untersten Ebene der Datenbanken mit der Analyse zu beginnen redundante Daten müssen aktualisiert werden

4.15.7 Warehouse – Hierarchien: Data Marts

Komplette Warehouse – Lösungen können in den meisten Fällen nicht auf einmal in Unternehmen aufgebaut werden. Deswegen kann eine schrittweise Einführung einer kompletten Warehouse – Infrastruktur vorgenommen werden.

- alle Datenextrakte von Produktionsdatenbanken werden zunächst einem unternehmensweiten Data Warehouse übergeben
- sobald diese Daten dort ankommen, lassen sie sich je nach Bedarf an die Warehouses der Abteilungen, auch als Datamarts bezeichnet, verteilen.
- diese Datamarts sind im Hinblick auf die Aufgaben (Subjects) organisiert, die für die jeweilige Abteilung von Interesse sind, z.B. die Verkaufszahlen und Kunden einer bestimmten Region
- ein wachsendes Interesse besteht auch an mobilen Datamarts -> persönliche Warehouses, die auf ein Notebook geladen und beispielsweise in einem Kundengespräch verwendet werden können

Zusammenfassung von Codd: Coddsche 12 Regeln

- Multidimensionale Sichtweise
- Transparenz -> Daten sollen von mehreren Datenquellen bezogen werden können
- Zugriffsmöglichkeiten -> GUIs für Benutzer sollen angeboten werden
- stabile Antwortzeiten -> Geschwindigkeit soll unabhängig vom Datenumfang sein
- Client / Server-Architektur -> soll zur Lastverteilung unterstützt werden
- Übergreifende Dimensionierung -> Operationen und Berechnungen sollen für alle Dimensionen des Modells gleichermaßen möglich sein
- dynamische Handhabung von Matrixlücken -> Lücken, die entstehen, wenn Tabellen mit unterschiedlicher Spaltenanzahl miteinander in Beziehung gesetzt werden
- Multiuser – Unterstützung -> um einen unternehmensweiten Einsatz zu gewährleisten
- uneingeschränkte Operationen über Dimensionen hinweg
- intuitive Datenmanipulation -> Der Anwender soll ohne Vorkenntnisse im Stande sein, zu verstehen und zu navigieren
- flexibles Berichtswesen
- unbegrenzte Dimensions- und Aggregationsstufen

4.15.8 FASMI – Definition für Data Warehouses / OLAP-Datenbanken

Da Codd mal wieder voll auf die Theoriedrüse gedrückt hat, wurde mit FASMI eine praktischere Definition für OLAP gefunden. Sie beschreibt OLAP in 5 Wörtern:

- **Fast** -> schnelle Antwortzeiten
- **Analysis** -> die Analyse soll intuitiv und Anwenderfreundlich gestaltet sein, adhoc- Analysen sollen möglich sein
- **Shared** -> Datenbestände sollen im Multiuser – Betrieb verfügbar sein -> Sperrmechanismen, Security-Aspekte, etc.
- **Multidimensional** -> Schlüsselkriterium für OLAP – Datenbanken / Data Warehouses; unabhängig davon, ob dies von einem relationalen oder nicht-relationalen DBMS dargestellt wird -> Slicing, Drill-Down, Roll-Up, Dicing, Pivoting
- **Information** -> Gewinnung von konkreten Informationen aus den unterschiedlichsten Daten

5. PROGRAMMIERSPRACHEN

5.1 Generationen von Programmiersprachen

Bei den Generationen von Programmiersprachen handelt es sich um hierarchisches Modell, welches die Einteilung von Programmiersprachen ermöglichen soll. (Vergleichbar mit dem ISO/OSI Modell für die Kommunikation). Da die Einteilung nicht nach der zeitlichen Entstehung der Programmiersprachen, sondern nach dem Abstraktionsgrad durchgeführt wird, ist der Begriff Generation nicht ganz glücklich gewählt.
Man kennt fünf Klassen von Programmiersprachen, welche von eins bis fünf durchnumeriert sind. Je niedriger die Zahl ist, desto niedriger ist der Abstraktionsgrad der Programmiersprache.

5.1.1 Maschinensprachen (1GL)

Die ersten EDV-Anlagen (Ende der 40er Jahre) ließen sich nur maschinennah programmieren. Der Programmcode musste bitweise in den Speicher des Rechners geschrieben werden. Der Vorteil der maschinennahen Programmierung liegt bis heute darin, dass diese Art von Programm direkt von einem Computer ausgeführt werden kann. Allerdings sind sehr genaue Rechnerkenntnisse erforderlich, da alle Anweisungen in Form von elementaren Befehlen sehr kleinschrittig beschrieben werden müssen. Problematisch gestaltet sich die Fehlersuche, wenn ein Programm überhaupt nicht läuft oder falsche Ergebnisse liefert.

5.2 Assemblersprachen (2GL)

Die Assemblersprachen, deren Befehlsvorrat speziell für jeden Rechnertyp zugeschnitten ist, verwenden anstelle des Binärcodes leichter verständliche Symbole, Mnemonics genannt. Ein Assemblerprogramm ist auf einem Computer nicht mehr direkt ablauffähig, sondern muss erst in ein entsprechendes Maschinenprogramm übersetzt werden. Ein Programm,

das dies automatisch durchführt, bezeichnet man als Assembler, den Übersetzungsvorgang als assemblieren. Der Nachteil von Assemblerprogrammen besteht darin, dass sie auf eine ganz bestimmte Hardware zugeschnitten sind und sich nur schwer auf andere Computertypen übertragen lassen. Bei größeren Problemlösungen werden die Programme sehr umfangreich und damit wartungsunfreundlich. Daher werden Assemblersprachen hauptsächlich nur noch da, wo Programme und Programmsysteme schnell reagieren müssen, und für Teile des Betriebssystems eingesetzt.

5.2.1 Prozedurale Programmiersprachen

Diese Sprachgeneration, der die überwiegende Mehrheit der heute gebräuchlichen Programmiersprachen angehört, ist unabhängig von einem Computersystem. Lediglich der Übersetzer (Interpreter oder Compiler) muss an das jeweilige System angepasst sein und den entsprechende Maschinencode erzeugen. Prozedurale Sprachen besitzen einen speziellen, der menschlichen Sprache angenäherten Befehlssatz, um Probleme aus einem bestimmten Anwendungsbereich zu lösen. Sie lehnen sich somit an die Denkweise des Programmierers an. Auch ohne fundamentierte Programmierkenntnisse lassen sich diese Programme leicht nachvollziehen. Die Bezeichnung „prozedural“ kennzeichnet den modularen Aufbau der entsprechenden Programmen in Prozeduren und Funktionen.

5.2.2 Nichtprozedurale Programmiersprachen (4GL)

Bei nichtprozeduralen Programmiersprachen wird nicht mehr festgelegt, wie ein Problem gelöst wird, sondern der Programmierer beschreibt lediglich, was das Programm leisten soll. Danach werden diese Angaben von dem Programmiersystem in ein Programm umgesetzt. Der Vorteil dieser Sprachen besteht darin, dass für diese Art der Programmierung keine umfangreiche Programmierausbildung notwendig ist. Nichtprozedurale Programmiersprachen werden z.B. für Datenbankabfragen oder Tabellenkalkulationen eingesetzt. In Delphi verwendet man z.B. die visuellen Komponenten, um eine Benutzerschnittstelle zu erstellen.

5.2.3 Programmiersprachen der künstlichen Intelligenz (5GL)

Die Programmierung der künstlichen Intelligenz (KI) dient der fortgeschrittenen Programmierung. Es wird versucht, die natürliche Intelligenz des Menschen (z.B. seine Lernfähigkeit) durch entsprechend konstruierte Computer nachzuvollziehen. Hierbei fließt beispielsweise auch die natürliche Sprache in die Programmierung ein. KI-Programme werden überwiegend zu Forschungszwecken eingesetzt und beschreiben Schlussfolgerungen aus Forschungsergebnissen. Erfolgreich werden derartige Systeme zur Spracherkennung eingesetzt.
Programmiersprachen, welche mehreren Kategorien zugeordnet werden können, bezeichnet man auch als *hybride Programmiersprachen*. Ein Beispiel dafür ist die Programmiersprache Delphi. Zur eigentlichen Programmierung wird ein pascalähnlicher Code verwendet, welcher sich in die dritten Generation einordnen lässt. Des weiteren wird zur Definition von

Schnittstellen ein SQL-Code verwendet, der zur vierten Generation gehört.

5.2.4 Schematische Darstellung

Abschliessend sei noch anzumerken das diese Einteilung nicht genormt ist. Es gibt keine Normvorschriften, die verbindliche Kriterien aufstellen, wo und was die exakten Grenzen sind. Aufgrund dieser Tatsache kommt es häufig zum Streit darüber, wo welche, meist neue Sprache, eingeordnet werden soll.

5.3 Unterteilung nach Programmierparadigmen

Eine Programmiersprache kann auch mehrere Konzepte enthalten (z.B. objektorientiert enthält imperative)

5.3.1 Imperative Sprachen

Imperative Sprachen zeichnen sich durch eine Folge von Befehlen aus. Z.B. schreibe in Variable a den Wert 10 oder addiere zu Variable b den Wert 20.Beispiele sind ALGOL, COBOL, FORTRAN, C, Pascal.

5.3.2 Funktionale oder applikative Sprachen

Diese bestehen aus Funktionen, Ein- und Ausgabewerten. Die Funktionen werden auf eine
Liste von Werten angewendet. LISP

5.3.3 Prädikative Sprachen

Sie bestehen aus Tatsachen und Schlußfolgerungen. Tatsachen sind die Vorgaben des
Anwenders.
Schlußfolgerung ist die Aufgabe des Rechners, also die Suche nach einer Lösung. Der Rechner gibt also gemäß den Vorgaben des Anwenders wahr oder falsch bzw. Variablenwerte, die die Vorgaben erfüllen, zurück. PROLOG

5.3.4 deskriptive Sprachen

Z.B. SQL. Hier werden Anfragen, Änderungs- oder Löschoperationen in Datenbanken durchgeführt.

5.3.5 Objektorientierte Sprachen

Bauen sich eine Miniwelt mit Objekten auf. Die Objekte besitzen Zustände, führen Operationen aus, können Nachrichten erhalten oder versenden. C++, Java.

5.4 Compiler vs. Interpreter

Um ein Programm zu testen, es zu verbessern oder es an neue Gegebenheiten anzupassen, muß es zuerst einmal von dem Computer verstanden werden. Dafür muß der Quellcode oder die Elemente des Programms in vom Computer verständliche Symbole umgesetzt werden. Dabei unterscheidet man grundsätzlich zwei Arten:

1. Das zeilenweise Übersetzen und auswerten während des Programmablaufs. (Interpreter)
2. Das Übersetzen des gesamten Programms vor dem ersten Programmablauf. (Compiler)

5.4.1 Klassifikation

Mächtigkeit einer Sprache := Anzahl der Konstrukte (z. B. Schleifen, bedingte Anweisungen), die diese Sprache anbietet

- Übersetzer (Compiler):
- | Quellsprache | > | Zielsprache |
- Umkehrübersetzer (Decompiler, Umkehrcompiler, Aufwärtsübersetzer):
- | Zielsprache | > | Quellsprache |
- 1-1-Übersetzer (Präcompiler, Präprozessor, Vorübersetzer):
- | Quellsprache | ~ | Zielsprache |
- Assemblierer:

  Übersetzer von einer maschinenorientierten Programmiersprache in einer Maschinensprache
- *Compiler-Compiler:*

  Übersetzer, der ausgehend von einer formalen Beschreibung einer Programmiersprache einen Compiler für diese Sprache generiert
- *inkrementeller Compiler:*

  das Zielprogramm wird schrittweise beim Editieren mitaufgebaut und Änderungen ohne globale Auswirkungen im Quellprogramm werden sofort umgesetzt

5.4.1.1 Anforderungen

- semantische Korrektheit:
- Das einem Quellprogramm a der Quellsprache A zugeordnete Zielprogramm b der Zielsprache B muß die gleiche Bedeutung (Semantik) wie a besitzen.
- Effizienz:
- Das Zielprogramm soll möglichst schnell sein und möglichst wenig Speicherplatz belegen.

5.4.1.2 Phasen eines optimierenden Übersetzers

- lexikalische Analyse:

Das Quellprogramm wird mit Hilfe eines Scanners in eine Folge von Token zerlegt. Ein Token ist ein grundlegendes Symbol der Quellsprache, z. B. ein Schlüsselwort oder ein Variablenbezeichner.

- syntaktische Analyse:
  Hauptziel: Das vorliegende Quellprogramm muß den Syntaxregeln der Quellsprache entsprechen !
  Der Parser erzeugt aus der Token-Folge einen Ableitungsbaum, dessen prinzipielle Struktur durch die Syntaxregeln der Quellsprache festgelegt ist.
- *semantische Analyse:*
  Der Ableitungsbaum wird analysiert und es wird beispielsweise kontrolliert, ob alle Variablen deklariert sind und entsprechend ihrem Typ verwendet werden.
- *Codegenerierung:*
  Das Zielprogramm wird mit dem Codegenerator erzeugt.
- *Codeoptimierung:*
  In der Regel ist es möglich, ein Konstrukt der Quellsprache in unterschiedliche Befehlsfolgen der Zielsprache abzubilden. Die Codeoptimierung soll daher eine möglichst optimale Sequenz finden. Meistens finden die Phasen Codeerzeugung und Codegenerierung gleichzeitig statt, so daß nur ein Zielprogramm erzeugt wird.

5.4.1.3 interne Struktur

Übersetzer, bei denen die Zielsprache eine Maschinensprache ist, werden intern oft in einen Front-End-Teil und einen Back-End-Teil zerlegt. Das Front-End übersetzt die Quellsprache in eine Zwischensprache, die noch maschinenunabhängig, aber bereits relativ maschinennah ist. Das Back-End nimmt dann ausgehend von dieser Zwischensprache die Codeerzeugung und –optimierung für die vorliegende Maschinenarchitektur vor.
Der Vorteil dieser Zerlegung ist, daß bei einer Portierung des Übersetzers auf eine andere
Maschinenarchitektur nur der Back-End-Teil neu erstellt werden muß.

5.4.2 Compiler (Übersetzung erfolgt einmalig)

Ein Compiler ist ein Übersetzungsprogramm, das ein in einer höheren Programmiersprache abgefasstes Quellprogramm in eine Maschinensprache übersetzt (kompiliert). Das Ergebnis der Kompilierung ist ein sogenanntes Objektprogramm in Maschinensprache, das in der Regel auf andere Softwarekomponenten (bspw. Programmbibliothek für Ein/Ausgabe) zurückgreift und nur gemeinsam mit diesen ausführbar ist. Diese grundlegenden Softwarekomponenten müssen ebenfalls als Objektprogramme zur Verfügung stehen und sind meist in sog. Programmbibliotheken zusammengefasst.

5.4.3 Interpreter (Übersetzung erfolgt schrittweise zur Laufzeit)

Ein Interpreter übersetzt Zeile für Zeile und moniert Fehler sobald er sie erkennt. Zum ausführen wird der Interpreter und der Quellcode benötigt. Der Interpreter erzeugt den auszuführeden Code zur Laufzeit !

Im Unterschied zum Compiler entsteht beim Interpreter kein dauerhaft gespeichertes Objektprogramm.
Vorteile gegenüber dem Compiler:
- Schnellere Entwicklungszyklen
- Plattformunabhängigkeit des Zwischencodes
- Sichere Programmausführung in einer geschützten Umgebung
- Vereinfachte Fehlerkorrektur

Nachteile gegenüber des Compilers:
- verringerte Laufzeiteffizienz
- Quelltext wird nicht durchgehend überprüft

5.4.4 Assembler

Ein Assembler ist ein Übersetzungsprogramm, das ein in einer Assembliersprache abgefasstes Quellprogramm in eine Maschinensprache umwandelt.

5.4.5 Linker

Um aus mehreren Objektprogrammen ein lauffähiges Programm zu erzeugen benötigt man ein weiteres Werkzeug, den Linker.
Dies ist ein Dienstprogramm, das mehrere Komponenten zu einem gemeinsamen Objektprogramm zusammenfasst und dabei Verweise (wie bspw. Funktionsnamen) auf externe Softwarekomponenten durch Sprungadressen ersetzt.

5.4.6 Loader

Ein Loader ist ein Dienstprogramm, das ein gebundenes Objektprogramm in den Arbeitsspeicher an eine vorgegebene Adresse lädt, dabei relative Adressen durch absolute ersetzt und die Ausführung des Programms anstösst.

5.4.7 Einsatz

Interpreter sind gut für die direkte Verarbeitung von Programmen, ohne das ganze Programm zu übersetzen. In Kauf nehmen muß man die relative Langsamkeit und den jederzeitigen Zugriff auf den Quellcode.
Der Compiler entwickelt aus dem Quellcode ein eigenständiges Programm, das nicht mehr ”lesbar” ist. Nachträgliche Manipulationen sind nur am Quellcode möglich, der aber nicht zusammen mit dem ausführbaren Programm gespeichert werden muß. Eine höhere Sicherheit ist gewährleistet, aber auf Kosten der Handhabung (Problem wird durch objektorientierten Ansatz gelöst!).

5.5 Stufen des Compilers:

- Befehle & Funktionen werden in die Maschinensprache übersetzt
- Speicherbedarf ermitteln und die Namen der verwendeten Variablen in das Adressbuch einzutragen
- Adressbuch und Befehle & Funktionen werden verknüpft

Das fertige Objektmodul liegt dann vollständig im Maschinencode vor.

5.6 Call by reference / call by value

5.6.1 Call by Value

Call By Value ist in vielen Programmiersprachen die Standard-Übergabeart für Parameter einer Funktion, bei der beim Funktionsaufruf nur der Wert des aktuellen Parameters übergeben wird, nicht jedoch der Name oder die Adresse, unter welcher der Ausdruck im Speicher steht. Bei der Parameterübergabe werden folgende Operationen nacheinander durchgeführt:

Innerhalb des Funktionsrumpfes wird eine lokale Variable mit dem Namen und Typ des entsprechenden formalen Parameters vereinbart.
Der aktuelle Parameter wird ausgewertet und das Ergebnis der neuen lokalen Variablen zugewiesen.

Wird der formale Parameter im Funktionsrumpf verwendet, so ist damit immer die lokale Variable gemeint. Veränderungen von Wertparametern haben daher auf die aktuellen Parameter keine Auswirkungen.

Beispiel:

```
void erhoehe (int iParmeter)   // ein Parameter wird als int-Variable vereinbart
{
     iParameter++;   // Aenderung geht verloren
}

void main(void)
{
     int iVariable = 40;
     printf ('\n Varialbenwert vorher : %d' , iVariable);   // iVariable hat den Wert
     40
     erhoehe (iVariable);   // das aufrufende Programm übergibt eine Kopie der
     Variablen
     printf ('\n Variablenwert nachher : %d' , iVariable;     // iVariable bleibt
     unverändert
                                                  (Wert = 40)
}
```

5.6.2 Call by Reference

Bei *Call By Reference* wird nicht mit Kopien von Werten gearbeitet, sondern mit den aktuellen Parametern selbst. Der aufgerufenen Funktion wird dabei die Adresse des aktuellen Parameters, unter welcher der Ausdruck im Speicher steht, übergeben. Man arbeitet also mit den Variablen des aufrufenden Programms über Zeiger.

Innerhalb des Funktionsrumpfes wird eine lokale Variable mit dem Namen des formalen Parameters und vom Typ „Zeiger auf den Parametertyp“ vereinbart. Anschließend wird der aktuelle Parameter ausgewertet, und die Speicheradresse des Ergebnisses wird der neuen Zeigervariablen zugewiesen. Wird der formale Parameter im Funktionsrumpf verwendet (z. B. bei Zuweisungen), so wird dieser immer ersetzt durch die

Speicheradresse bzw. durch den Wert an der Speicheradresse.
Veränderungen von Referenzparametern haben daher Auswirkungen auf die übergebenen aktuellen Parameter.

Beispiel:

```
void erhoehe (int* iParmeter)   // ein Parameter wird als int-Zeiger vereinbart
{
     (*iParameter)++;   // Aenderung geht nicht verloren
}
void main(void)
{
     int iVariable = 40;
     printf ('\n Varialbenwert vorher : %d' , iVariable);   // iVariable hat noch
     den Wert 40 erhoehe (&iVariable);   // aufrufende Programm übergibt
     die Adresse der Variablen printf ('\n Variablenwert nachher : %d' ,
     iVariable;   // jetzt hat iVariable den Wert 41
}
```

5.7 Objektorientierung

5.7.1 Grundgedanke

Der Grundgedanke der objektorientierten Systementwicklung besteht darin, Daten und Funktionen nicht nacheinander, sondern simultan zu betrachten und bei der Modellierung zu Objekten zusammenzufassen

5.7.2 Definition Objekt:

Ein Objekt beschreibt einen Ausschnitt aus der realen Welt oder eines Modells davon. Ihm sind Merkmale, Merkmalswerte und Operationen zugeordnet. Die Merkmalswerte legen einen Zustand des Objekts fest. Durch die Operation kann der Zustand ermittelt beziehungsweise modifiziert werden.

Objekte besitzen 3 wesentliche Eigenschaften:

- ein Objekt ist ein Exemplar (eine Instanz) einer Klasse, damit besitzt es einen (zeitinvarianten) Typ
- jedes Objekts besitzt einen neuen (zeitvarianten) Zustand, der durch die aktuellen Werte seiner Zustandvariablen festgelegt ist
- jedes Objekt besitzt eine eindeutige (zeitinvariante) Identität

5.7.2.1 Def. Klasse:

Eine Klasse beschreibt Eigenschaften durch Merkmale und Operationen. Aus einer Klasse können Objekte abgeleitet werden, sogenannte Exemplare der Klasse. Sie besitzen die in der Klasse beschriebenen Eigenschaften, das heißt, ihre Merkmale und Operationen.
Durch die Zuordnung eines Klassennamen wird die Identifizierung unterschiedlicher Klassen ermöglicht.

5.7.2.2 Def. Nachricht:

Der Zugriff auf ein Objekt ist nur über eine Nachricht möglich. Eine

Nachricht löst eine Operation des Objektes aus. Operationen können den Zustand eines Objektes ermitteln, den Zustand verändern bzw. eine Nachricht an das Objekt oder an ein anderes Objekt senden.

5.7.2.3 Def. Vererbung:

Die Vererbung ist eine Beziehung zwischen zwei Klassen, von denen die eine Oberklasse und die anderen Unterklasse genannt wird. Die Unterklasse wird aus der Oberklasse abgeleitet; sie erbt alle Eigenschaften (Merkmale und Operationen) der Oberklasse. Die Unterklasse kann um neue Merkmale beziehungsweise neue Operationen erweitert werden. Die aus der Oberklasse geerbten Operationen können bei Bedarf hinsichtlich ihrer Wirkung modifiziert werden.
Eine Oberklasse kann als Verallgemeinerung ihrer Unterklassen betrachtet werden. Werden aus einer Unterklasse weitere Unterklassen abgeleitet, so entsteht eine Klassenhierarchie.
Das Konzept der Vererbung ist die wesentliche Grundlage für die Wiederverwendbarkeit, Erweiterbarkeit bzw. Kompatibilität von Programmbausteinen. Bereits eine einzelne Klasse, wir wollen diese Basisklasse nennen, kann wiederverwendet werden. Dazu werden Objekte dieser Klasse gebildet. Erfordert die Wiederverwendung einer Veränderung der Eigenschaften der Basisklasse, so wird eine Unterklasse aus der Basisklasse abgeleitet, und die entsprechenden Operationen werden modifiziert. Eine Erweiterung der Unterklasse gegenüber der Basisklasse kann durch hinzufügen neuer Merkmale beziehungsweise neue Operationen erreicht werden.

5.7.2.4 Def. Polymorphismus:

Polymorphismus oder "Vielgestaltigkeit" bedeutet, daß eine Methode unter dem gleichen Namen mit unterschiedlichen Implementierungen in verschiedenen Klassen einer Klassenhierarchie enthalten ist.
Bei einer derartigen Methode wird erst zur Laufzeit des Programmes entschieden, welche Implementierung der Methode aufgerufen wird (dynamische Binden). Solche Methoden werden virtuelle Methoden genannt (und durch das Schlüsselwort VIRTUAL hinter dem Prozedurkopf gekennzeichnet).
Virtuelle Methoden gestatten es, Operationen mit dem gleichen Namen auf Objekte aus verschiedenen Klassen anzuwenden.

5.7.2.5 Def. Nebenläufigkeit:

Programme, deren Teile unabhängig voneinander auf parallelen Prozessoren laufen, werden als nebenläufig bezeichnet.

5.7.2.6 Persistenz:

Persistenz ist die Eigenschaft von Objekten, über die Laufzeit des Programmes hinaus in ihrer Identität, ihrem Zustand und Ihrer Beschreibung zu existieren.

5.7.2.7 Datenkapselung:

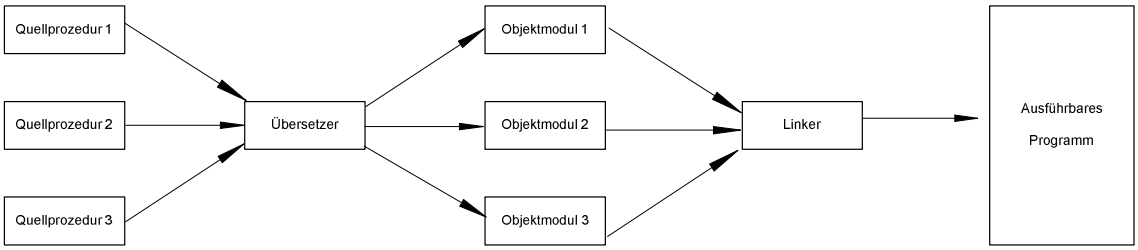
Datenkapselung bedeutet, daß die Attributwerte eines Objektes nur durch die Methoden des Objektes selbst, jedoch nicht von außen verändert

werden können. Damit gilt das Geheimnsprinzip von Parnas, und zwar in der Form, daß Objekte nach außen nur erkennen lassen, WAS sie ausführen, jedoch nicht, WIE die Ausführung erfolgt.

5.8 Funktionen des Linker

Wenn der Programmierer sein Programm fertig geschrieben hat, möchte er es auch ablaufen lassen; nach Möglichkeit auch noch auf verschiedenen Rechnern(die natürlich kompatibel sein müssen). Den Objektcode hat der Compiler erzeugt, und die Bibliotheken mit den Befehlsbedeutungen liegen auch vor. Im Grunde besteht also noch die Hürde mit dem eigenen Programm zu arbeiten.
Es ist jedoch in der PC-Praxis etwas umständlich, immer Objektmodule und die zugehörigen Bibliotheken im Auge und damit ständig einsatzfähig zu behalten. Beim Gebrauch eines Programms an verschiedenen Standorten müßten auch die Bibliotheken installiert um weiterhin für die notwendigen Startprozeduren des Objektmoduls zu sorgen. Was liegt also näher, als die übersetzten Befehle – das Objektmodul – mit ihren Bedeutungen – den Bibliotheken – zu verknüpfen? Hierfür gibt es die Linker

5.8.1 Allgemeinen



5.9 JAVA

5.9.1 Was ist JAVA ?

JAVA ist eine Objektorientierte Programmiersprache, dessen Beliebtheit in den letzten Jahren mit gleicher Stärke gewachsen ist wie die des World Wide Web.

Ein Projekt von Sun, bestehend aus:
- ♦ einem WWW-Browser
- ♦ einer Programmierumgebung

Eine Programmierumgebung, bestehend aus:
- ♦ einem Compiler
- ♦ einem Laufzeitsystem
- ♦ einem Debugger

Eine Programmiersprache
- ♦ objektorientiert
- ♦ statisch getyped
- ♦ ähnlich C++, aber rechnerunabhängig und komfortabler

Warum ist das so: JAVA ist

- einfach zu lernen
- sehr weit verbreitet
- von den führenden Browser Herstellern anerkannt
- „umsonst“ -> kann im WWW heruntergeladen werden

5.9.2 Was kann man mit JAVA machen ?

Abgesehen von den üblichen „Hello World“ Prgrämmchen, kann man mit Java einfach und effektiv zur Verbesserung und Verschönerung seiner Homepages benutzen. Dies hat nicht nur kosmetische, sondern, im Falle einer großen Firma, auch durchaus monetär meßbare Gründe.

5.9.3 Woraus besteht ein Java-Programm?

Objekte – Klassen – Funktionen

5.9.4 Eigenschaften von Java?

5.9.4.1 Portabel

Das Erfolgskonzept von Java beruht auf seiner Plattform-Unabhängigkeit. Wenn Sie heute eine neue Anwendung entwickeln möchten, muß diese für eine bestimmte Kombination von Hard- und Software geschrieben sein - so läuft z.B. eine Textverarbeitung, die für Windows entwickelt wurde, nicht unter UNIX. Wenn daher erreicht werden soll, daß alle Computer diese Anwendung nutzen können, muß eine dedizierte Version für jedes Betriebssystem Entwickelt werden. Das ist kostspielig. Deshalb beschränken sich die meisten Hersteller darauf, eine begrenzte Anzahl von Plattformen zu unterstützen, und ignorieren die anderen vollständig. Java durchbricht die enge Verbindung zwischen der Anwendung, dem Betriebssystem und der Hardware. Das heißt, daß Entwickler Anwendungen erstellen können, die auf jedem Computer ablaufen können so, wie eine Web-Seite von jedem Browser eingesehen werden kann. Programme, die in Java entwickelt wurden, werden nicht, wie üblich, als Maschinen-Code verteilt, sondern basieren auf einem bestimmten Format, das als Byte Code bekannt ist. Erst die ***Java Virtual Machine*** wandelt diesen Byte Code in Maschinen-Code um. Bei der Java Virtual Machine handelt es sich um eine weitere Software-Anwendung, die das Byte-codierte Java Dokument interpretiert und es in Befehle übersetzt, die sowohl der Computer wie das Betriebssystem verstehen. Sie können sich die Virtual Machine als einen herkömmlichen Computer vorstellen, der in Software implementiert wurde. Java Virtual Machines sind derzeit für viele Betriebssysteme und Prozessorplattformen verfügbar - und weitere befinde sich in Entwicklung.

Das einzigartige an Java ist, daß es nicht nur in Bürocomputern eingesetzt werden kann. Da die Virtual Machine für jedes Microprozessor-basierende Gerät entwickelt werden kann, kann Java in allen Arten von Geräten wie Mobiltelefonen, Modems und künftigen TV-Systemen genutzt werden.

5.9.4.2 Objektorientiert

Java ist *vollständig objektorientiert* aufgebaut. Es ähnelt hierbei der Programmiersprache C++

wobei auf unsichere Features (z.B. direkter Zugriff auf Pointer) verzichtet wird.

5.9.4.3 Multithreaded

Ein Applet wird in mehrere einzelne Prozesse bzw. *Threads* unterteilt. Damit kann sich ein Thread z.B. um die Aktualisierung der Bildschirmausgabe kümmern während ein anderer Prozeß mit einem Server kommuniziert. Daher ist eine Parallelisierung möglich.

5.9.4.4 Verteilt

Durch Java stehen dem Entwickler einige Methoden oder auch Klassen zur Verfügung die die Netzwerkprogrammierung unterstützen.
Da Java-Applets selbst meist über das Internet geladen werden, ist das System von vornherein auf eine verteilte kommunizierende Umgebung ausgerichtet.

5.9.4.5 Robust

Da ein sehr großer Anteil von Fehlern auf die Pointerverarbeitung zurück zuführen ist, können diese Fehler in Java erst gar nicht gemacht werden. Das System kümmert sich automatisch um die Anforderung und Freigabe von Speicherplatz.
Auch die strenge Objektorientierung mit gleichzeitiger strenger Typenprüfung leistet ihren
Beitrag zur Robustheit.

5.9.4.6 Sicher

Da Java-Programme keine echte Binärprogramme für Betriebssysteme sind, sondern Bytecode für virtuelle Maschinen kann die Umsetzung des Interpreters einiges an Sicherheit bringen.
So enthalten heutige Java-Bytecode-Interpreter in WWW-Browsern normalerweise keine Möglichkeit, auf das lokale Dateisystem zuzugreifen. Der Interpreter enthält zusätzlich einen sogenannten *Bytecode-Verifier*, der das Programm vor dem Ablauf nach einigen Regeln hin überprüft.

6. Netzwerke

6.1 Kanalprinzip & Bussystem

6.1.1 Kanal-Konzept

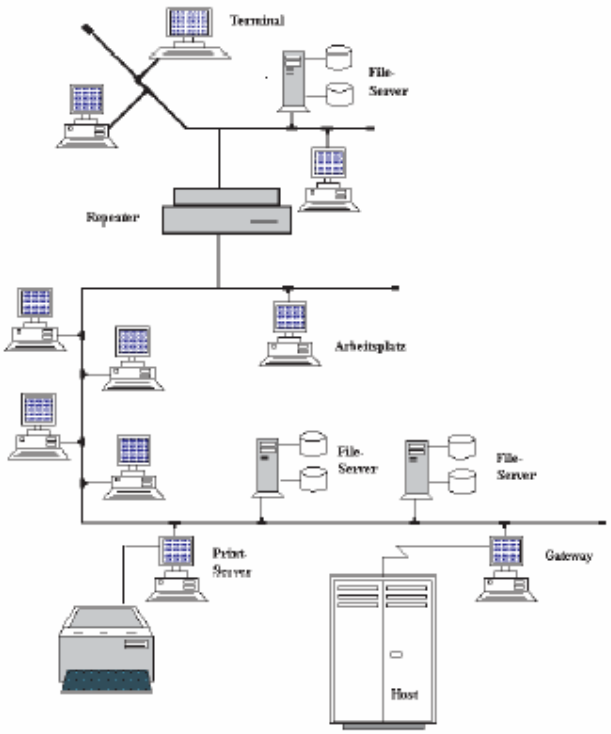
Die peripheren Geräte werden alle miteinander über Kanäle verbunden. Die Kanäle sind selbstständig arbeitende Einheiten mit eigenen Prozessoren (Kanalprozessoren), die neben der CPU parallel arbeiten. Die Kommunikation wird von Steuereinheiten übernommen.
Arten der Kanäle:
Parallel: bitparallele Übertragung über Kupferkabel
Seriell: bitseriell über Glasfaser

Vorteil: es können alle Geräte gleichzeitig miteinander kommunizieren
Nachteil: hoher Kanalaufwand

6.1.2 Bus-Konzept

Alle Peripheriegeräte sind mittels jeweils einem Peripherie-Prozessor an einem Strang, dem E/A-Bus verbunden. Die Prozessoren entsprechen den Steuereinheiten beim Kanalkonzept. Gesteuert wird der Bus vom Betriebssystem in der CPU. Unterteilung des Busses (logisch, aber nicht unbedingt physisch) in folgende Bestandteile:

- Steuerbus
- Adressbus
- Datenbus



Vorteile:

- einfach umsetzbar
- leicht erweiterbare Struktur
- Stationen können bei laufendem Betrieb an- und abgeklemmt werden
- Busverkabelung ist modular
- hohe Übertragungsgeschwindigkeiten
- störunanfällig, da kein Master-Server notwendig

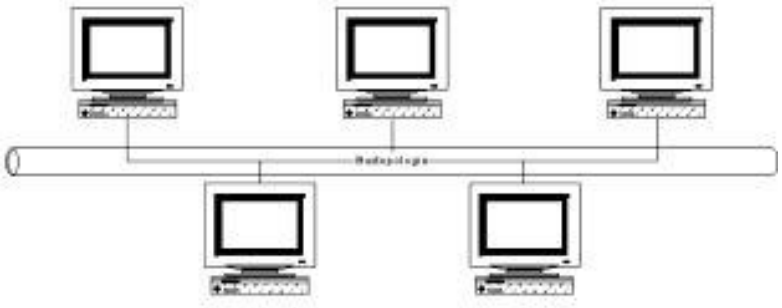
Nachteile:

- keine Abhörsicherheit
- es können immer nur 2 Geräte gleichzeitig miteinander kommunizieren, da dadurch der gesamte Bus belegt ist
- Struktur anfällig gegenüber Ausfall vom Übertragungsmedium
- begrenzte Längenausdehnung

6.2 Netzwerktopologien

6.2.1 Bustopologie

Die Netzwerkteilnehmer sind durch ein gemeinsam genutztes Medium untereinander verbunden. Somit sind kurze Kabelwege möglich und der Ausfall einer Station beeinträchtigt die Übertragung nicht. Bsp.: Ethernet
Bei der Bustopologie sind alle Stationen an einem gemeinsamen passiven Übertragungsmedium angeschlossen. Das Übertragungsmedium muss am Ende durch einen Widerstand terminiert (abgeschlossen) werden. Die Stationen sind nicht Bestandteil des Busses. Das Signal wird damit von den Stationen auch nicht verstärkt. Es ist keine Masterstation nötig, die die Kommunikation im Netz steuert.
Daten, die über den Bus von einer Sendestation zur Empfangsstation übertragen werden, werden von alle Stationen am Bus empfangen, aber nur von der Empfangsstation ausgewertet.



Vorteile:

- leicht erweiterbar
- (störunanfällig) Ausfall einer Station beeinträchtigt die Funktion des Netzes nicht
- geringe Leitungsanzahl
- sehr preiswert, da kein zentraler knoten benötigt wird

Nachteile:

- Die Längenausdehnung ist begrenzt
- Kabelfehler kann das komplette Netz lahm legen
- je nach eingesetztem Protokoll treten unvorhersehbare Wartezeiten bei der Kommunikation auf

6.2.2 Ringtopologie

Funktionsweise:
Token (Bitfolge) liegt sendebereit auf dem Ring. Sendewillige Station nimmt Token vom Netz und hängt Daten an. Es entsteht ein belegtes Token. Das Token kreist nun im Netz und jede Station schaut, ob sie Empfänger ist, wenn nicht, wird das Signal verstärkt und weitergeleitet zur nächsten Station im Ring. Wenn ja, nimmt die Station die Daten vom Token und setzt das Empfangsbit. Wieder bei der Sendestation angekommen entfernt diese das Empfangsbit und erzeugt somit wieder einen Freitoken, der von der nächsten sendewilligen Station benutzt werden kann.

Probleme:
- Ziel-PC fällt aus → Quell-PC erzeugt wieder Freitoken
- Quell-PC fällt aus → Monitorstation (beliebiger Rechner im Netz) erzeugt bei 2x gleichem Paket Freitoken
- max. Kabellänge zwischen Stationen überschritten → MSAU (multiple station access unit → entspricht einem Hub im Ethernet)

Vorteil:
- keine Kollisionen
- weitläufige Netze realisierbar

Nachteil:
- bei Ausfall einer Station Zusammenbruch des Netzes, wenn keine Alternativleitung gelegt wurde
- Abhörgefahr

6.2.3 Baumtopologie

Die Kommunikation zwischen den Knoten A und B erfolgt immer über den Knoten, der beiden übergeordnet ist (Wurzelknoten).
Das Baumnetz ist aus mehreren Sternnetzen zusammengesetzt, die hierarchisch angeordnet sind.
Bsp.: HUB-Prinzip

Vorteil:
- Verkabelung billiger als beim Stern
- Erlaubt gut strukturierte Organisation von großen Datenbeständen

Nachteil:
- Ausfall von kompletten Ästen wenn „hoher" Knoten defekt
- Zentralrechner wichtig und teuer

6.2.4 Sterntopologie

Alle Knoten sind durch physikalische Verbindungen an einen zentralen Knoten angeschlossen.
Unterscheidung in zwei Arten von Sternsystemen:
- Bei passivem Stern verhält sich der Stern neutral (Zugriffssteuerung liegt bei Stationen).
- Bei aktivem Stern bestimmt der Hub, wer senden und empfangen darf.

Vorteil:
- kostengünstig
- leicht erweiterbar
- bei Ausfall einer Station keine Beeinflussung für Netz
- max. 1 Zwischenknotenpunkt für Signal

Nachteil:
- abhängig von Funktionsfähigkeit und Belastbarkeit des Zentralrechners
- hoher Verkabelungsaufwand, wenn Entfernung zwischen Rechnern hoch.

6.2.5 Maschennetze

Jeder Knoten ist mit mind. 2 weiteren Knoten verbunden. Bei vollständig vermaschten Netzen ist jeder Knoten mit jedem vernetzt. Dies ist allerdings nur bei Netzen mit geringem physikalischem Abstand zu realisieren, da der Aufwand in keiner Relation zum Ergebnis steht. Die Anzahl der Verbindungen bei vollst. vermaschten Netzen beträgt n x (n-1) / 2. in der Praxis werden für Weitverkehrsnetze (z.B. Internet) vielfach verschiedene Topologien vermischt. Unterschiedliche Teilnetze werden häufig durch unregelmäßig vermaschte Netzstrukturen verbunden.
Vorteil:

- hohe Ausfallsicherheit

Nachteil:

- hohe Kosten bei starker Vermaschung

6.3 Zugriffsverfahren / Kanalzuteilung

6.3.1 Stochastische/konkurrierendes Verfahren

Alle Stationen können gleichzeitig auf das Medium zugreifen. Gibt es eine Störung durch gleichzeitigen Zugriff ziehen sich die Stationen zurück und versuchen es nach einer zufällig ermittelten Zeit erneut.

6.3.1.1 ALOHA

Zufälliges Senden ohne Busprüfung
→ bei Kollision Beenden der laufenden Übertragung und erneuter Versuch nach beliebiger Zeit

6.3.1.2 Slotted ALOHA

Man legt fest, dass Pakete nur zu bestimmten Slots starten dürfen (bsp. Nur alle 0,5s)

6.3.1.3 CSMA

→ *Carrier Sense:* die sendewillige Station muß erst den Bus abhören, ob er frei ist. Erst dann darf sie senden.
→ *Multiple Access:* Alle Stationen sind gleichberechtigt, es gibt keine Sendeprioritäten. Die erste Station, die sendet, bekommt den Bus (Zufall).
3 Arten:

- 1-persistent: sobald Kanal frei wird, sende!
- Nicht-persistent: wenn Kanal besetzt, warte zufällige Zeit, dann probier es erneut
- p-persistent: wenn Kanal frei, dann sende mit Wahrscheinlichkeit p, sonst nächster Slot

6.3.1.4 CSMA/CD

für Busse:

→ *Carrier Sense:* die sendewillige Station muß erst den Bus abhören, ob er frei ist. Erst dann darf sie senden.
→ *Multiple Access:* Alle Stationen sind gleichberechtigt, es gibt keine Sendeprioritäten. Die erste Station, die sendet, bekommt den Bus (Zufall).

→ *Collision Detection:* Kommt es zu einer Kollision, so werden die Übertragungen abgebrochen und nach einer Zufallszeit wieder gestartet.

6.3.1.5 Adaptive Tree Walk

Wenn bei n Stationen eine Kollision auftritt, dann Teile die Stationen in 2 Teile und lass den ersten Teil wieder senden. Bei erneuter Kollision teile diese Hälfte erneut und lass die erste senden, usw. Bei erfolgreichem Senden lass den nächsten Abschnitt senden

6.3.2 Deterministische Verfahren

Man arbeitet nach dem Token-Prinzip (s. 6.2.2). Es wird sichergestellt, dass zum Sendezeitpunkt jeweils nur 1 Station senden darf. Somit vermeidet man weitestgehend Kollisionen

6.3.2.1 CSMA/CA

Hier werden Prioritäten für die einzelnen Stationen vergeben, die Station mit der höchsten
Priorität darf als erstes senden.
Vorteil:

- kleine Bandbreitenverluste

Nachteil:

- Stationen mit niedriger Priorität kommen selten oder gar nicht zu senden. Hoher Verwaltungsaufwand um dies abzuschaffen

6.3.2.2 Token Ring

(=> Token –Passing): Hier sind Datenkollisionen ausgeschlossen;
Im Netz kreist ein Token, das der jeweiligen Station das Senderecht gibt. Die Stationen sind gleichberechtigt, da das Token ständig von einer Station zur nächsten wandert.
Siehe 6.2.2 Ringtopologie

6.3.2.3 Polling (Sendeaufruf)

Die angeschlossenen Stationen werden in einer festen Reihenfolge periodisch zum senden aufgerufen.
Es werden der Reihe nach alle Stationen durch ENQ (Enquiry → Anforderung) zum Senden aufgerufen. Sendet eine Station nicht, schickt sie ein EOT (End of Transmission) und die nächste wird aufgerufen. Sendet eine Station z.B. ihren Text, wird nach Beendigung ein ACK (Acknowledge → positive Empfangsbestätigung) von der Leitstation geschickt. Nun beendet die Station die Übertragung durch ein EOT. Usw.

6.3.2.4 Selecting (Empfangsaufruf)

Funktioniert wie polling (6.3.2.3), nur werden hier die Stationen zum empfangen aufgefordert.
Der Unterschied zum Polling ist das gezielte Auswählen der Adressen, da sich die Clients vorher schon einmal „anmeldeten“. Beim Selecting werden Stationen durch ein ENQ zum Empfang aufgerufen. Ist eine Station nicht empfangsbereit, so sendet sie ein NAK (negatives Acknowledge) zurück und

die nächste Station wird aufgerufen. Ist eine Station empfangsbereit, sendet sie ein ACK zurück, woraufhin die Leitstation den Text überträgt.

Jede Station erhält vor Beginn eine ID. Wenn mehrere Stationen gleichzeitig senden wollen, werden die ID-Nummern der sendewilligen Stationen verglichen, und nur die höchste darf senden. Da dies auf Dauer die Stationen mit sehr niedrigen IDs benachteiligen würde, entwickelten die Forscher *Mok und Ward* eine Variante, nach der die ID-Nummern rotieren (sendende Station erhält als nächstes die 0 und die anderen rücken auf). Somit kommt jede Station zum Senden.

6.3.3 Drahtlose Netze

6.3.3.1 WDMA

Wavelength Division Multiple Access → bei hoher Bandbreite
Jede Station hat 2 Kanäle (Steuer- und Datenkanal) → Steuerkanal bestimmt, wer über Datenkanal senden darf.

6.3.3.2 MACA

Multiple Access with Collision Avoidance
Durch Ready to send (RTS) und Clear to send (CTS) wird Kommunikation gesteuert. → wenn eine Station etwas empfangen will (CTS), dann sind die umliegenden Stationen still.

6.3.3.3 CDMA

Code Division Multiple Access
Erlaubt verschränkte Übertragungen, die sich gegenseitig nicht stören.
Alle Stationen haben eindeutige Chipsequenzen, die paarweise orthogonal, also entgegengerichtet sind.
Bei Kenntnis der Sequenzen kann Bitstrom durch etwas komplizierte Rechnung wiederhergestellt werden.
Eine gesendete „0“ wird zu „-1“, eine „1“ wird zu „+1“ → nur wenn eine Station nicht sendet steht „0“.
Nachteil: Je länger die Chipsequenz, desto geringer die Bandbreite

6.4 ISO/OSI Protokolle; TCP/IP

6.4.1 Protokolle

Vereinbarung über den geordneten Ablauf einer Kommunikation. Sie enthalten Absprachen über den Aufbau, die Überwachung (anhand von Fehlermeldungen) und den Abbau von Verbindungen.
Entsprechend der aufeinander aufbauenden Schichten bei der Übertragung von Informationen wird für jede Ebene ein eigenes Protokoll vereinbart, dessen Realisierung sich auf das Protokoll der nächst tieferen Ebene abstützt.
Protokolle sind bei der Kopplung von Systemen und in Offenen Systemen (bisher) unverzichtbar.
Ziel aller Standardisierungsversuche sind offene Kommunikationssysteme (OSI = Open systems interconnection), d.h. Kommunikationsmöglichkeiten zwischen unterschiedlichen Rechnern, Betriebssystemen, Netzwerkmodellen usw. ISO

hat 1983 dazu eine Rahmenempfehlung aufgestellt – ISO/OSI Referenzmodell.

6.4.1.1 OSI-Referenzmodell

Beim OSI-Referenzmodell geht es um die **Art und Weise** wie man den **Datenaustausch**
zwischen Rechnern einheitlich beschreibt.
Bereits 1977 gründete die Internationale Standardisierungsorganisation, kurz ISO, einen Ausschuss für **„Offene Systemzusammenschaltung".** In Englisch: „**Open System Interconnection**", kurz OSI.

Ziel des Ausschusses:
Ein Modell zu finden, das sich mit der Verbindung offener Systemen befasst. Ein offenes System kann mit fremden Systemen kommunizieren, ganz im Gegensatz zu einem proprietären System, das seinen eigenen Kommunikationsstandard besitzt.

6.4.1.2 Eigenarten des OSI- Referenzmodells

- macht keinerlei Aussagen über die technische Realisierung und legt keine Spezifikationen bezüglich:
  - der Rechnerarchitektur,
  - Betriebssystem
  - oder Programmiersprache fest
- es handelt sich um ein Modell und nicht um eine Realisierungsvorschrift
- es spezifiziert keine exakten Dienste und Protokolle
- Das OSI- Referenzmodell beschreibt lediglich die **Aufgabe** jeder Kommunikationsschicht

6.4.1.3 angewandte Prinzipien

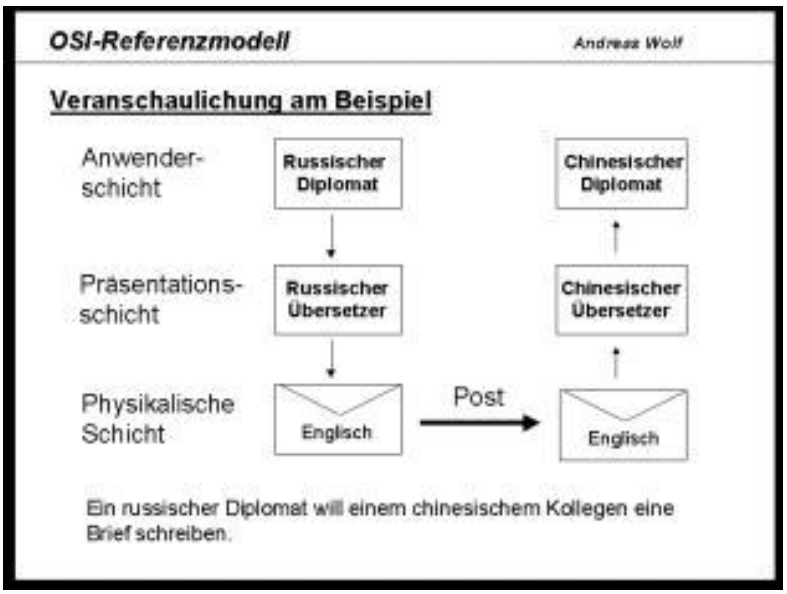
Bei der Erstellung des OSI-Modells wurden folgende **Prinzipien** angewandt:

1. Eine Schicht wird genau dann erzeugt, wenn ein neuer Abstraktionsgrad der Kommunikation benötigt wird.
2. Jede Schicht soll eine genau definierte Funktion ausführen
3. Die Schichten sollen so gewählt werden, dass der Informationsaustausch zwischen den Schichten minimal ist.
4. Anzahl der Schichten soll groß genug gewählt werden, dass die verschiedenen Funktionen sich nicht überschneiden.
5. Anzahl der Schichten klein genug, um Übersichtlichkeit zu wahren.

6.4.1.4 Veranschaulichung an einem Beispiel

Der grundlegende Gedanke lässt sich an einem Beispiel illustrieren:

Ein russischer Diplomat will seinem chinesischen Kollegen einen Brief schreiben.
Problem hierbei: Der Russe beherrscht nur kyrillisch und der Chinese nur die chinesische
Schriftsprache.



In diesem kleinen Modell erkennt man drei Schichten:

Oberste Schicht: Anwenderschicht

Hier sitzt der Diplomat.

→ Hier wird das ausgehende Dokument verfasst

Mittlere Schicht: Präsentationsschicht

Hier sitzt der Übersetzer.

→ Das Dokument wird in eine standardisierte Sprache übersetzt, bzw. zurückübersetzt.

Unterste Schicht: Physikalische Schicht

Hier sitzt das Postsystem.

→ Das Dokument wird befördert.

Damit die Kommunikation zwischen den beiden Diplomaten stattfinden kann, müssen Regeln geschaffen werden, wie der Ablauf der Kommunikation stattfinden soll:

(1) Physikalische Schicht

Festlegung der Übertragungsart

→ Die Info wird mittels Brief übertragen

(2) Präsentationsschicht

Festlegung der Schriftsprache für die beiden Übersetzter

→ Schriftsprache Englisch.

Sog. **Protokolle**, die unabhängig voneinander sind, beschreiben diese Kommunikationsregeln.

6.4.1.5 Aufbau des OSI- Referenzmodells

In ähnlicher Weise wie im obigen Beispiel wird die Kommunikation zwischen Computeranwendungen geregelt.
Das OSI-Modell dient als Rahmen für die **Beschreibung von Protokollen**. Die obere Schicht nimmt dabei Dienste der unteren Schicht in Anspruch, ohne zu wissen, wie diese ihre Aufgaben erfüllt.

Das OSI-Modell basiert auf **7 Schichten**, die voneinander abgegrenzte Aufgaben erledigen:

6.4.1.5.1 Physical Layer (physikalische Schicht)

Beschreibt die Frage in welchen physikalischen Größen die Bits gesendet werden.
„Wie muss ein Bit gesendet werden, dass es auch wirklich ankommt?“
z.B.:

- Signalspannung
- Sendedauer eines Bits
- Pinbelegung des Netzwerksteckers
- Vermittlungsarten (Paket- oder Leitungsorientiert)
- Übertragungsraten (Nyguist, Shannon)

6.4.1.5.2 Data Link Layer (Sicherungsschicht)

Ermöglicht fehlerfreie Verbindungen zwischen Endsystem und Netzzugang.
a) Logical Link Control – Layer

- Übertragungsraten werden vereinbart
- Die von den übergeordneten Schichten kommenden Daten werden in Frames zusammengefasst. (Character-, Bit-Stuffing)
- Fehlererkennung (CRC) und -korrektur (Hamming-Distanz)

b) Medium Access Control - Layer

- Kanalzugriffssteuerung

6.4.1.5.3 Network Layer (Vermittlungsschicht)

- Übernimmt Routing und stellt somit eine End-to-end-Kommunikation bereit.
- Es werden Pakte geroutet
- Statisches Routing:
  - Wege werden über statische Tabellen festgelegt (virtuelle Leitungen)
  - dynamische Routing:
  - jedes Paket wird individuell zum Zielrechner geroutet abhängig von der Netzauslastung (Datagramme)
  - Es sind auch Abrechnungsfunktionen integriert. „Wieviele Bits werden gesendet?“
- wird im Internet von IP realisiert

6.4.1.5.4 Transport Layer (Transportschicht)

- nimmt Daten aus der oberhalb liegenden „Session Layer“ entgegen
- splittet diese falls nötig in kleinere Pakete
- gibt diese an die untergeordnete „Network Layer“ weiter

Die Transportschicht ist eine Art „Verbindungsschicht“.
Sie stellt den oberen Schichten ein netzunabhängigen Transportmechanismus zur Verfügung.

6.4.1.5.5 Session Layer (Kommunikationsschicht)

- errichtet, erhält und beendet die Kommunikationssitzungen
- Zugangskontrolle / Datenschutz

6.4.1.5.6 Presentation Layer (Darstellungsschicht)

→ Obiges Beispiel: Hier sitzen die Übersetzer

- Beschäftigt sich mit Syntax und Semantik
- Beispiel: Codierung der Daten

Verschiedene Computersysteme benutzen unterschiedliche Codierungen
z.B. ASCII oder UNICODE
Diese Codes werden in eine rechnerunabhängige Netzwerk-Standardcodierung umgewandelt.

6.4.1.5.7 Application Layer (Anwendungsschicht)

→ obiges Beispiel: hier sitzen die zwei Diplomaten

Aufgabe:
Enthält Vielzahl von Protokollen für unterschiedlichste Zwecke.

in TCP/ IP Struktur:
- Telnet
- FTP (File Transfer Protocol)
- SMTP (Simple Mail Transfer Protocol)

6.4.2 TCP/IP

- „Transfer Control Protocol / Internet Protocol“
- Basis des Internets
- Protokollfamilie,
- in 4 Schichten aufgeteilt (Verbindungs-, Internet-, Transport- und Anwendungsschicht)
- fand weiter Verbreitung durch Realisierung in UNIX (open source)

Verbindungsschicht:

- enthält Schnittstellen und Hilfsprotokolle für Zusammenspiel mit anderen Schichten

Internetschicht:

- erfüllt Aufgaben der Vermittlungsschicht (OSI)
- enthält zentrales Protokoll: IP → weltweit eindeutige Adressen definiert → IP-Adressen

Transportschicht:

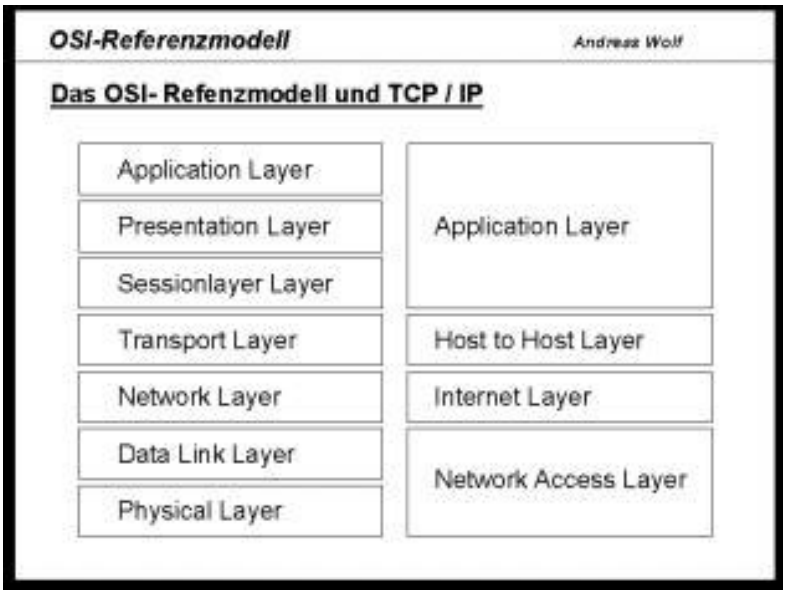
- enthält zweites Kernprotokoll: TCP → stellt Anwendungen sicheres, verbindungsorientiertes Protokoll zur Verfügung
- weiterhin Protokoll UDP → für verbindungslose Anwendungsprotokolle

Anwendungsschicht:

- TCP und UDP sind Grundlage für Anw.-Schicht-

→ Bsp: SMTP (Mail), Telnet, FTP, HTTP
- diese Protokolle definieren Austausch von Steuerinformationen und Nutzdaten zwischen Anwendungen

6.4.3 OSI- Referenzmodell und TCP/IP im Vergleich



TCP/IP ist Standard zur Kommunikation mit der offenen Umgebung (Bsp.

Internet)
bei TCP/IP sind die Schichten nicht so klar abgetrennt und daher größer.
Alle versuche TCP/IP OSI- konform zu beschreiben sind nur grobe Annäherung geblieben.

Grund für die Unterschiedlichkeit:

Die Entwicklung der grundlegenden TCP/IP Protokolle fand vor der Veröffentlichung des OSI- Referenzmodells statt.

6.5 Client- Server- Architektur

Das Client/Server-Modell beruht auf dem Prinzip des Downsizing, mit dem Großrechner (Mainframes) schrittweise durch ein System von vernetzten Desktoprechnern abgelöst werden. Vorteile des C/S-Modells liegen in der unternehmensweiten Nutzungsmöglichkeit aller vorhandenen Ressourcen und darüber hinaus in

- einer besseren Zuordnung arbeitsplatzbezogener Aufgaben,
- der Delegierbarkeit von Verantwortlichkeiten an dezentrale Stellen,
- höheren Verfügbarkeiten im Netz bei insgesamt starker Netzauslastung,
- einer größeren Flexibilität als bei zentralen Systemen,
- der Konzentration von Verwaltungsaufgaben im Netz auf ausgewählte Server und
- insgesamt niedrigen Hardware- Kosten.

Das C/S- Modell beschreibt, dass in einem Rechnernetz einzelne Rechner als Server fungieren, die Dienstleistungen zur Verfügung stellen. Diese Dienstleistungen können von anderen Rechnern, den Clients, genutzt werden. Ziel des C/S- Modells ist die gemeinsame Nutzung aller im Unternehmen existierenden Anwendungssysteme, Datenbestände und Rechner- bzw. Geräteleistungen durch alle dazu berechtigten Stellen.

6.6 Serverarten

Generell bezeichnet man eine Netzkomponente, deren Leistung allen Netzteilnehmern zur Verfügung steht als Server.

6.6.1 File-Server

Der Server stellt nur komplette Dateien zur Verfügung. Es ist ein meist umfangreicher Sekundärspeicher mit mehreren Gigabyte Plattenspeicher. File-Server besitzen besonders schnelle Datenverwaltungs- und Datenübertragungssysteme, die die Datenspeicherung und – bereitstellung für verschiedene Rechner aufnehmen. Zur Ausführung von Spezialaufgaben besitzen sie weitere Prozessoren, die für schnelle Ein-/Ausgabe, schnellen externen Datenzugriff, schnellen Bildschirmaufbau usw. sorgen.

6.6.2 Print-Server

Print-Server werden eingesetzt, wenn in einem Rechnerverbund nicht jeder Rechner einen eigenen Drucker hat. Einer der angeschlossenen Rechner fungiert dann als Druckserver.

6.6.3 Fax-Server

Ein Rechner, der die ganze Zeit angeschlossen ist und somit ständig bereit ist Fernbriefe zu empfangen. Auch nur im Netz von Vorteil (gleiches Prinzip wie Print-Server.

6.7 Internet/Intranet

Internet:

- 01.01.1983 Geburtsstunde des Internets → alle Rechner des ARPANets (Netzwerk aus 4 Knoten des amerikanischen Verteidigungsministeriums) von NCP (Network Core Protocol) auf TCP/IP umgestellt
- besteht aus einer Menge von Rechnern, die direkt oder indirekt miteinander verbunden sind und die das Kommunikationsprotokoll (TCP/IP) verwenden.
- Sie sind in verschiedenen Netzen mittels Gateways erreichbar und stellen eine Kommunikationsinfrastruktur zum Datenaustausch zur Verfügung.
- paketorientiert, d.h. die Daten werden beim Absender zerstückelt und die Pakete über verschiedene Wege zum Empfänger geleitet, wo sie in der richtigen Reihenfolge wieder zusammengefügt werden
- Mischung aus allen Topologien.
- Dienste:
  - File Transfer: FTP (File Transfer Protocol)
  - World Wide Web: HTTP (Hypertext Transfer Protocol)
  - Usenet News: NTTP (Network News Transfer Protocol)
  - E-Mail: SMTP (Simple Mail Transfer Protocol)
  - Terminal Emulation: Telnet (Telnet Protocol)

Intranet:

- lokales Netzwerk mit Internet-Technik
- auf die physikalischen und rechtlichen Grenzen einer Unternehmung beschränkt
- Firewall schützt vor feindlichen Angriffen von außen
- Im Intranet können interne Informationen (schwarzes Brett, Organigramme) oder auch Formulare zum Herunterladen (Urlaubsanträge) bereitgestellt werden.

6.7.1 WWW (World Wide Web)

- Heute gebräuchlichste Anwendung zur Darstellung von Informationen im Internet
- Ermöglicht auch Zugriff zu älteren Diensten, wie Archie, Gopher, WAIS (Wide Area Information Systems)
- Arbeitet nach Client-Server-Prinzip: Client fordert an, Server bietet an bzw. führt Anfrage aus
- Benutzte Seitenbeschreibungssprache: HTML (Hypertext Markup Language)

→ermöglicht multimediale Inhalte, Links, Einbettung von Grafiken, Einbettung von Applets, Common Gateway Interface (CGI)= automatische Generierung von Seiten aus externen Datenquellen beim Aufruf, so dass

speziell zugeschnittene Seiten benutzerspezifisch generiert werden)

6.7.2 FTP

- ermöglicht Dateiübertragung von und zu FTP-Servern
- hat an Bedeutung verloren
- Funktionsweise: FTP-Programm aufrufen, Serveradresse eingeben (numerisch oder den Namen), Dateien beim angesprochenen Server suchen und mittels GET oder PUT- Befehl transferieren
- Die Bezeichnung FTP steht dabei sowohl für den Dienst, mit dem auf Remote-Host gesucht und übertragen wird, als auch für das verwendete Übertragungs- und Fehlerprotokoll

FTP-Protokoll: beseitigt Inkompatibilitäten von verschiedenen kommunizierenden Betriebssystemen, z.B. DOS- Befehl des Clients (DIR) in UNIX-Befehl des Servers (LIST) umwandeln

6.7.3 E-mail

Wird realisiert durch das SMTP-Protokoll, welches das Übertragen elektronischer Post regelt.

6.7.4 TELNET

- Stellt Verbindung zu Remote-Server her
- Ermöglicht agieren auf anderen Rechnern
- Aufruf: Telnet –Dienst und Eingabe des Server-Namen (IP oder der Name)
- Auch Telnet setzt auf TCP-IP auf

6.7.5 Archie

- Sucht den Standort von Dateien die auf FTP-Servern liegen, ermöglicht Recherchen von Resourcen vieler FTP-Server
- Funktionsweise: Aufruf von Archie, Eingabe des Namens der gesuchten Datei oder auch nur bekannte Teile des Namens, Suchergebnis ist die Adresse des FTP-Servers und das Verzeichnis, in dem die Datei liegt

6.7.6 Gopher

- Ist eigentlich nur ein Verzeichnis- und Dateisystem
- Hinter dem Verzeichnisnamen kann jedoch ein anderer Server stehen, zu dem dann eine
- Verbindung aufgebaut wird
- Bietet einen besseren Bedienkomfort als die Suche per Telnet

6.7.7 WAIS

(Wide Area Information Systems)

- Liefert Suchmöglichkeiten nach Informations- und Kommunikationsquellen im Internet
- Es wird neben Recherchemöglichkeiten nach Dateien auch deren Einsicht „vor Ort“ ermöglicht
- Die verfügbaren Informationen können auf verschiedenen Servern

liegen, werden aber wie beim WWW als einheitliche Menüoberfläche wiedergegeben
- Ist wohl heute auch kaum noch jemandem ein Begriff

7. Systementwicklung / Projektmanagment

7.1 Prinzipien, Methoden, Verfahren/Techniken, Werkzeuge/Tools

Prinzipien: sind Grundsätze, die man seinem Denken und Handeln zugrunde legt. Sie sind allgemein gültig, abstrakt und allgemeinster Art und bilden eine theoretische Grundlage. Sie sind ein allgemeingültiger Grundsatz des Denkens und Handelns im Sinne einer Norm. Sie sagen nichts darüber aus, wie man sie - bezogen auf ein konkretes Anwendungsgebiet - erreichen kann, z.B.:
- Top down- Prinzip
- Bottom up- Prinzip
- Modularisierung (Basis von Top down und Bottom up; entweder in Teilsysteme zerlegen oder aus Teilsystemen zusammensetzen)

Methoden: sind systematische Handlungsvorschriften oder Vorgehensweisen, um Aufgaben zu lösen oder Ergebnisse darzustellen. Sie sind demnach planmäßig angewandte, begründete und schrittweise Vorgehensarten zur Erreichung von festgelegten Zielen. Einer Methode liegt oftmals ein Prinzip zugrunde.
Verfahren/Techniken: sind Anweisungen zum gezielten Einsatz von Methoden. Sie stellen eine ganz bestimmte Handlungsweise dar. Sie sind demnach die allgemeine Kenntnis und Beherrschung, der Mittel, die zur Ausübung eines Handwerks, einer Kunst notwendig sind, z.B.: Entwurfstechniken für Systemrealisierung, Struktogramme, PAP, Pseudocode, Entwurfstechniken für Systemplanung/-entwurf, Entscheidungstabelle, Hierarchiediagramme, HIPO = Hierarchy Input Process Output, SA = Strukturierte Analyse, Jackson - Diagramm, Petrinetze, SADT = Structured Analysis and Design Technik, Istaufnahmetechniken, Unterlagenstudium, Selbstaufschreibung, Fragebogen, Interview, Dauerbeobachtung,Kreativitätstechniken, Brainstorming/-writing

Verfahren geben an, welche Konzepte wie und wann verwendet werden, um festgelegte Ziele zu erreichen.

Konzipieren bedeutet eine Grundvorstellung von etwas zu entwickeln, zu verfassen und zu entwerfen.

Konzepte erlauben es also, definierte Sachverhalte unter einem oder mehreren Gesichtspunkten zu modellieren.

Werkzeuge/Tools: sind computergestützte Hilfsmittel für die Aufgaben beim Software- Engineering. Sie automatisieren ein Verfahren. Ein Werkzeug implementiert zwangsläufig eine Technik/Verfahren, d.h. es dient der

Unterstützung von Methoden/Techniken/Verfahren. Sie werden eingeführt, um den Einsatz von Verfahren zu erleichtern, zu beschleunigen und abzusichern, z.B.:

- Programmierwerkzeuge
- Datenwerkzeuge, Data Dictionaries
- Testwerkzeuge

Werkzeuge dienen der automatisierten Unterstützung von Methoden, Verfahren, Konzepten und Notationen.

7.1.1 Allgemein Software - Engineering/ Systementwicklung (SE)

Systementwicklung in der Daten- bzw. Informationsverarbeitung befasst sich mit der Entwicklung von Anwendungssystemen.
Vorgehensmodelle in der Softwareerstellung beschreiben die Folge aller Aktivitäten, die zur Durchführung eines Projektes erforderlich sind. Sie geben an, wie die Prinzipien, Methoden, Verfahren und Werkzeuge der Software-Entwicklung eingesetzt werden. Die folgenden sollten man kennen und kurz erläutern können:
Das Softwarelebenszyklus-Modell geht von einer Abfolge einzelner Phasen aus. Das Wiederholen einer Phase ist nur ausnahmsweise gestattet. Nach Beendigung der Phase Betrieb und Wartungliegen möglicherweise neue Anforderungen vor. Dann beginnt der Zyklus der Softwareentwicklung von neuem mit einer neuen Version der Software.
Zyklische Modelle verstehen die Software als Folge von Versionen.

Das Wasserfallmodell geht ebenfalls von einer strikt sequentiellen Abfolge der Projektphasen aus. Im Gegensatz zum Softwarelebenszyklus-Modell sind die einzelnen Schritte weiter verfeinert worden und die Wechselwirkungen zwischen den Phasen werden abgebildet.
Das Wasserfallmodell ist ein Phasenmodell. Kennzeichen dieser Modelle ist, dass sie Herstellung von Software als eine Folge von Abschnitten (Phasen) beschreibt.
Beispielhafter Ablauf:
Planungsphase Definitionsphase Entwurfsphase Implementierungsphase Abnahme- und Einführungsphase

Spiralmodell
Das Spiralmodell stellt eine Verfeinerung des Wasserfallmodells dar. Es trägt zusätzlich der Tatsache Rechnung, das zu beginn einer Software Entwicklung mehrere Lösungsalternativen zur Verfügung stehen können (z.B. Aussehen des GUIs). Durch Prototypen können schon in einer relativ frühen Phase der Entwicklung die späteren Benutzer einbezogen werden. Zusätzlich lassen sich auch die Risiken und der weiteren Entwicklung abschätzen. Der Prototyp wird nun schrittweise verfeinert. Am Ende des Spiral-Modells steht die Installation der Software.
Die folgenden Schritte werden wiederholt durchlaufen, wobei der Prototyp mit jeder Wiederholung weiterentwickelt wird.
(Leistungsbeschreibung, DV - Feinkonzept, Prototypentwicklung, Validierung, Anforderungen)

Das Spiralmodell kann als evolutionäres Modell betrachtet werden. Der Vorteil dieser Modelle ist die Anpassung an geänderte Anforderungen schon während der Laufzeit, sowie die Interaktion zwischen Entwicklern und Benutzern.

7.1.2 Funktionalität

Funktionalität: Fähigkeit von SW- Produkten, ihre Aufgaben exakt zu erfüllen, wie sie durch Anforderungen und Spezifikationen definiert sind.

Qualität: immer in Bezug auf SW- Produkte

Robustheit: Fähigkeit, auch unter außergewöhnlichen Bedingen zu funktionieren

Erweiterbarkeit: Leichtigkeit der Anpassung an Spezifikationsänderungen

Wiederverwendbarkeit: Eigenschaft, ganz oder teilweise wiederverwendet zu können

Portabilität: Maß der Leichtigkeit der Übertragung auf verschiedene Hard- und (System-) Software- Umgebungen

Integrität: Fähigkeit des Schutzes der verschiedenen Komponenten gegen unberechtigte Zugriffe und Veränderungen

7.2 Phasenkonzept

Um in der Systementwicklung die Frage, welche Aktivitäten in welcher Reihenfolge auszuführen sind, haben sich verschiedene Vorgehensmodelle durchgesetzt.
Der Gesamtprozeß besteht aus mehreren zeitlich aufeinander folgenden Stufen mit den 4 Grundphasen:

- Systemanalyse
- Systementwicklung
- Systemeinführung und
- Systempflege

7.2.1 Initialisierungsphase

Es werden ein Grobkonzept und eine Machbarkeitsstudie erstellt. Bei positiver Bewertung wird daraus ein Projektauftrag formuliert.

7.2.2 Analysephase

Zunächst wird das gegebenenfalls bestehende System analysiert. Danach wird der inhaltliche Soll-Zustand (Anforderungsprofil) des neuen Systems erstellt. Aus diesem läßt sich das Pflichtenheft ableiten. Nach dem Vorgaben des Pflichtenhefts kann eine Lösung auf dem Standard-Softwaremarkt gesucht werden.
Am Ende der Phase stehen eine Durchführbarkeitsuntersuchung, sowie Entscheidungen über das weitere Vorgehen.

7.2.3 Entwurfsphase

Im Mittelpunkt dieser Phase stehen der Entwurf des organisatorischen Konzepts, sowie der Softwareentwurf. Im Falle der Einführung von Standardsoftware werden in dieser Phase Angebote eingeholt, analysiert und ausgewählt.
Auch die entgültige Festlegung von Einzelheiten der benötigten Hardware findet in dieser Phase statt.

7.2.4 Realisierungsphase

In dieser Phase liegen die Programmierung der Software bzw. die Einführung der Standardsoftware sowie der Test. Weiter Tätigkeitsfelder sind die Inbetriebnahme des Systems, Änderungen an den betrieblichen Arbeitsabläufen und die Schulungen der beteiligten MA.

7.3 Methoden der Ist-Analyse

Die Methoden zur Erhebung des Ist-Zustands sind:

- Unterlagenstudium
- Fragebogen
- Interview
- Konferenz
- Beobachtung
- Selbstaufschreibung

Gelegentlich wird auch zwischen objektiven und nicht objektiven Techniken unterschieden. Hier kommt es in erster Linie darauf an, ob die gewünschten Informationen direkt zu beschaffen sind oder auf den Aussagen beteiligter Personen beruhen.

Das Unterlagenstudium wird auch als Dokumentanalyse oder Dokumentenstudium bezeichnet. Hier werden vorhandene, aktuelle schriftliche Unterlagen analysiert. Es kann sich um folgende Unterlagen handeln:

- Geschäftsberichte, Bilanzen, Aktionärsmitteilungen;
- Organisationshandbücher und -pläne, Stellenbesetzungspläne, Stellenbeschreibungen;
- Materialflußpläne, Erzeugnisstrukturen, Stücklisten; etc.

Werden vorhandene Programme untersucht, sollten ferner folgende Unterlagen eingesehen werden:

- Programmdokumentationen (Handbücher und / oder Online-Dokumentationen);
- Datenflußpläne, Programmablaufpläne;
- eingesetzte Hardware, Betriebssysteme (Systemplattformen) und sonstige Systemsoftware; etc.

Das Unterlagenstudium muß zu Beginn der Erhebung stattfinden. Es hat das Ziel, eine breite Informationsbasis zu schaffen.Antworten werden auf konkrete projektbezogene Fragen können hier nicht erwartet werden.
Der Fragebogen (schriftliche Befragung) soll strukturiert sein und darf nur präzise und verständlich formulierte Fragen enthalten. Er muss durch unauffällige Kontrollfragen, z.B. bezüglich des Verständnisses; ergänzt

werden.Vor der endgültigen Erhebung sollte der Fragebogen an einzelnen Testpersonen erprobt werden. Die Zustellung sollte mit einem Begleitschreiben des Auftraggebers erfolgen. Der Fragebogen hat nur bedingte Aussagekraft, weil er nicht immer mit Begeisterung und ehrlich ausgefüllt wird. Außerdem besteht die Gefahr, dass Antworten untereinander abgesprochen werden.
Das Interview (mündliche Befragung) ist die vorherrschende und ergiebigste Erhebungstechnik. Es konzentriert sich meist auf einen einzigen Gesprächspartner. Das Interview sollte strukturiert werden und nach einer schriftlichen Vorlage erfolgen. Dazu kann ein Fragebogen dienen. Die Antworten sind zu protokollieren. Es werden alle offenen und versteckten Argumente gegen das Projekt aufgetan.
Bei der Konferenz (Besprechung) werden Fragen, insbesondere auch widersprüchliche Ansichten aus den Interviews, im größeren Kreis diskutiert. Man unterscheidet zwischen Besprechungen, die zur Erörterung einer weitgehend feststehenden Entscheidung und Besprechungen zur Erarbeitung eines Vorschlags durch Diskussion. Auch hier sollte der Themen- und Fragenkatalog gründlich vorbereitet werden. Wichtig für den Erfolg sind eine sorgfältige Auswahl der Teilnehmer und eine straffe Diskussionsleitung. Die Ergebnisse sind zu protokollieren.

Als Beobachtung bezeichnet man die optische Aufnahme und Interpretation von Arbeitsabläufen. Die Beobachtung kann strukturiert (Multimomentaufnahme) oder unstrukturierte (Dauerbeobachtung) sein. Es können Widersprüche aus Interviews aufgeklärt werden. Beobachtungen sind zeitaufwendig. Der Beobachtete kann sich jedoch beeinflußt fühlen, wodurch die Ergebnisse verfälscht werden.
Die Selbstaufschreibung durch Mitarbeiter der untersuchten Stellen dient vorrangig zur Ermittlung von Zeiten und Mengen. Die Aufschreibung erfolgt anhand von gedruckten oder am Bildschirm angezeigten Formularen. Diese müssen verständlich sein und von den Projektbearbeitern vor der Bearbeitung erläutert werden. Voraussetzung für korrekte Angaben ist bei allen Beteiligten die Bereitschaft zu neuen Lösungen.

Vor- und Nachteile

Befragung

- keine objektiven Tatbestände
- Auskunft über Meinungen, Einstellungen
- Fragebögen schrecken oft durch Menge ein

Kundennummer

- Bei Interview nicht unbedingt Vergleich möglich
- Neue Aspekte können im Gespräch erkannt werden
- Es ist teilweise eine Nachfrage möglich

Beobachtung

- tatsächliches Verhalten
- keine Rückschlüsse über Verhalten und Motivation
- größere Objektivität als Selbstaufschreibung

Selbstaufschreibung

- wird viel geschummelt
- Zeitliche Dinge können besser beurteilt werden

7.4 Strukturierte Analyse / Systementwurf

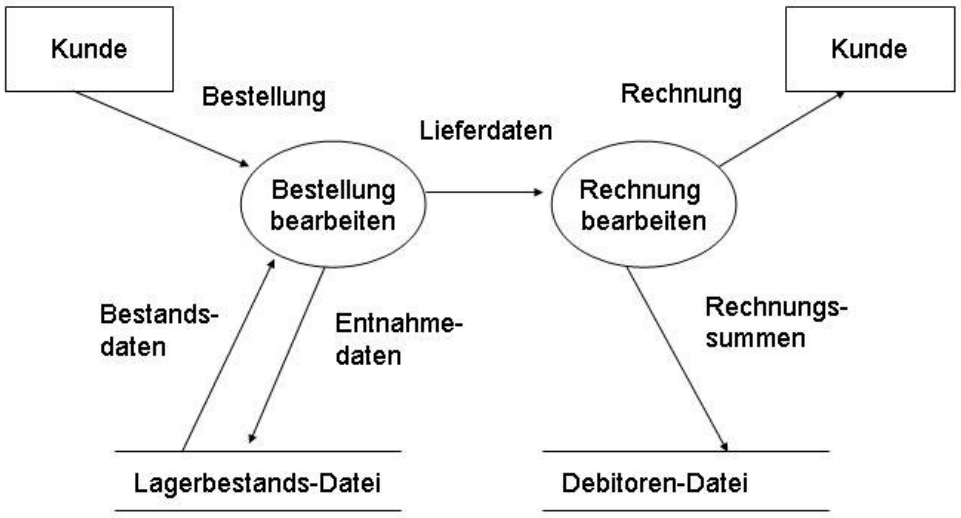
SA stellt heute weltweit einen Industrie Standart dar und ist dementsprechend gut normiert. Es handelt sich um einen Methode zur Darstellung eines Ist- oder Soll-Modells.
Die wichtigsten Konzepte sind:

- Denken in Datenflüßen und Funktionen
- Anwendung des Top-Down Prinzip
- Kontroll und Steuerflüße (Bedinungen und Schleifen) werden nicht dargestellt, folglich handelt es sich um eine Abstraktion
- Es werden grafische Beschreibungsmittel verwendet

Eine vollständige SA einer gegebenen Problemstellung besteht aus:

1. **Datenflußdiagramm**, das die Wege der Daten durch das System darstellt
2. **Data Dictionary**, das die Struktur der Daten angibt
3. **Mini Spezifikationen**, die wichtigsten Funktionen näher beschreiben, z.B. als Pseudo Code oder verbale Beschreibung

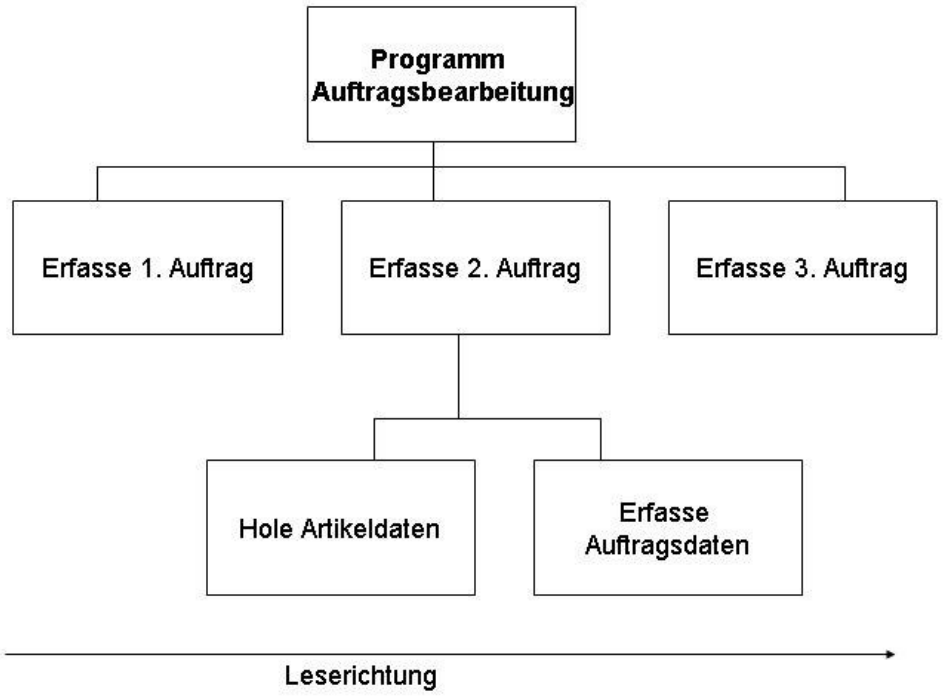
Die Strukturierte Analyse (SA) wurde von DE MARCO und anderen Mitte der 70er Jahre entwickelt und 1977 erstmals veröffentlicht.
Ausgegangen wird vom Datenfluß. Dazu werden im Datenflußdiagramm Datenflüsse durch Pfeile, Arbeitsvorgänge (Prozesse) durch Kreise, Datenspeicher durch zwei parallele Linien und Datenquellen und -ziele durch Quadrate oder Rechtecke dargestellt. Nach dem top-down- Prinzip werden schrittweise verfeinerte Datenflußdiagramme angelegt. Im Datenverzeichnis werden alle Daten, Datenflüsse und Arbeitsvorgänge beschrieben. Das Datenstrukturdiagramm kennzeichnet die Beziehungen zwischen den für das Anwendungssystem angelegten Dateien. Die Prozessspezifikation, d. h. die Beschreibung der den Prozessen gehörenden Operationen, ergibt sich aus der untersten Stufe des Datenflußdiagramms. Sie kann beispielsweise mit Hilfe von Entscheidungstabellen oder in Pseudo-Code vorgenommen werden.
Eine Weiterentwicklung von SA ist die Methode SSAD (Structured System Analysis and
Design), die ebenfalls 1977, und zwar von GANE und SARSON entwickelt worden ist.
In der Praxis der Software-Entwicklung hat die Methode SA die stärkste Verbreitung gefunden. Sie wird häufig mit ER-Diagrammen kombiniert, die den Datenentwurf übernehmen. Die SA beschreibt die Funktion und die Abläufe. Generell bedeutet jedoch jede solche Trennung zwischen Daten und Funktionen einen Bruch in der durchgängigen Systementwicklung. Außerdem erschwert sie die Kommunikation zwischen Anwendern und Entwicklern. Diese Mängel versucht die objektorientierte Vorgehensweise zu beheben.



Bereits in den 60er Jahren setzten in der Programmentwicklung Bemühungen ein, den Prozeß der Programmentwicklung zu systematisieren. Es wurde das Ziel verfolgt die Programme, vor allem in Hinblick auf die spätere Wartung, übersichtlich und auch für andere lesbar zu gestalten. Ein Ergebnis dieser Bemühungen ist die Strukturierte Programmierung. Sie beginnt im Gegensatz zum linearen Programmierstil damit, das gesamte Programm in einer top down-Entwicklung bis auf die Ebene weitgehend voneinander unabhängiger Systembausteine, sogenannter Strukturblöcke, zu zerlegen. Die Strukturierung soll so weit wie möglich von Programmiersprachen unabhängig sein.
Eine grafische Darstellungsform für strukturierte Programmierung wurde 1974 von JACKSON vorgeschlagen (JSP = Jackson Structured Programming). Dabei wird von Datenflüssen ausgegangen, die in einem sogenannten Datenstrukturdiagramm beschrieben werden. Daraus wird unter Verwendung der drei Konstrukte

- Sequenz (oder Strecke) für die Reihung
- Verzweigung (oder Selektion) und
- Schleife (oder Iteration) für die Wiederholung die Programmstruktur abgeleitet.

Hierbei werden für die Darstellung ausschließlich Rechtecke in strikt hierarchischer Anordnung benutzt (Jackson-Bäume). Die unterste Stufe der Zerlegung heißt Element (oder atomare Komponente). Die Abarbeitung erfolgt immer von links nach rechts. Man kann JSP auch so anwenden, daß man damit ausschließlich die Programmstrukturen darstellt und auf das vorangehende Datenstrukturdiagramm ganz verzichtet.



7.5 Prinzipien der Softwareentwicklung

Die wichtigsten Prinzipien sind die:

- top-down Entwicklung
- bottom-up Entwicklung
- Modularisierung

Sowohl die top-down Entwicklung als auch die bottom-up Entwicklung basieren auf dem Prinzip der Modularisierung. Modularisierung bedeutet, daß das Gesamtsystem:

- entweder durch eine top-down-Entwicklung in Teilsysteme zerlegt
- oder durch eine bottom-up Entwicklung aus Teilsystemen zusammengesetzt wird.

Die top-down Entwicklung wird auch als schrittweise Verfeinerung, die bottom-up Entwicklung als schrittweise Verallgemeinerung bezeichnet.

Def. Prinzipien sind Grundsätze, die man seinem Handeln zugrunde legt.

Prinzip der konstruktiven Voraussicht und methodischen Restriktion
= Welche Konsequenzen haben die Produktanforderungen auf die Software

Prinzip der Abstraktion
= Verallgemeinerung (Vom Besonderen zum Einzelnen)

Prinzip der Strukturierung
= Die Struktur ist eine reduzierte Darstellung des gesamten Systems. Sie beinhaltet alle wesentlichen Merkmale der Software

Prinzip der Hierachisierung
= Anordnen aller Strukturbestandteile gemäß einer Rangordnung

Prinzip der Modularisierung
= Aufteilung in funktionale Einheiten mit genau definierten Schnittstellen

Prinzip der Lokalität
= Alle relevanten Informationen sollen gemeinsam an möglichst genau einer Stelle gespeichert

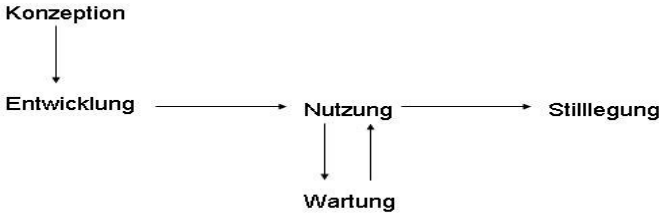
Prinzip der integrierten Dokumentation
= Das Softwareprodukt besteht aus Programmcode und ausführlicher Dokumentation

Prinzip der Mehrfachverwendung
= Zeit und Kostenersparnis durch Wiederverwendbarkeit von Modulen bzw. Klassen

7.6 Testverfahren für Software

7.6.1 Der Lebenszyklus einer Programms

Der Lebenszyklus eines Programms beinhaltet die Phasen vom ersten Konzept bis zur Stillegung des Programms.



7.6.2 Fehler in der Software

Da die Entwicklung eines Programmes meist ein sehr komplexer Prozess ist und dadurch oft unübersichtlich wird, schleichen sich leicht Fehler ein.
Je später diese im Lebenszyklus eines Programms entdeckt werden, desto schwieriger, zeitaufwendiger und teurer wird es, sie wieder auszubügeln.
Deshalb führt man Softwaretests heute nicht erst nach der Programmierung, sondern schon davor und währenddessen durch.
Man verfolgt dabei das Ziel, Entwicklungs- und Wartungszeit insgesamt zu verringern, wobei eine verhältnismäßig längere Entwicklungszeit in Kauf genommen wird, um damit die Wartung zu verkürzen.

7.6.3 Testen von Programmen

Unter Testen in Verbindung mit Software versteht man die Prüfung von codierten Programmen zum einen auf die korrekte Formulierung und zum anderen auf die korrekte Ausführung .
Unter korrekter Formulierung versteht man zum Beispiel die Syntax, die von der verwendeten Programmiersprache abhängt und festlegt, wie ein Befehlswort geschrieben werden muss und ob Argumente übergeben werden müssen.

Wird ein Programm korrekt ausgeführt, so erfüllt es alle Leistungen und Anforderungen, die im Pflichtenheft festgelegt sind.
Das Testergebnis eines korrekt laufenden Programms bezeichnet man als Programm- Verifikation.

7.6.4 Vorgehensweise beim Testen

Zunächst erstellt man die Struktur des Testfalls.
Man legt Randbedingungen und Eingabewerte fest; z.B. Variable a>0, welche Werte sollen als Parameter für eine Operation übergeben werden oder welche Aktionen löst der Benutzer aus usw.
Man ermittelt das zu erwartende Ergebnis, um es später mit dem Testergebnis des Programms zu vergleichen.
Die Testergebnisse müssen wiederholbar sein, d.h. bei einem erneuten Test unter denselben Bedingungen muss dasselbe Ergebnis herauskommen.

Das prinzipielle Problem des Testens besteht darin, dass das Testen nur die Anwesenheit von Fehlern aufzeigen, nie jedoch deren Abwesenheit beweisen kann. Ferner sind erschöpfende Tests wegen der Fälle von möglichen Eingaben und Programmzustände bei größeren Programmen nicht möglich. Da man nicht alle erstellten Testfälle ausführen kann, muss man nun die relevanten auswählen.

7.6.5 Testtechniken

7.6.5.1 Blackbox-Test

Bei diesem Test wird das Programm als Blackbox aufgefasst, d.h. die interne Struktur ist nicht relevant, denn es soll lediglich das für den Anwender sichtbare Verhalten, also die Ausführung, getestet werden.
Die Testfälle können aus den Anforderungen an das Programm (siehe Pflichtenheft) entnommen werden.
Es sind nur die Testfälle relevant, die später auch vom Anwender ausgeführt werden.

7.6.5.2 Whitebox-Test

Ein Whitebox-Test ist der eigentliche Programmtest, denn hier steht die interne Struktur des Programms im Mittelpunkt.
Der Programmtext wird detailliert, Zeile für Zeile, auf die korrekte Logische Formulierung überprüft.
Jeder Programmteil, z.B. Schleifen, If-Bedingungen oder Case-Of-Anwendungen, sollten
jeden für sie möglichen Fall mindestens einmal durchlaufen haben.

7.6.6 Stufen des Software-Tests

7.6.6.1 Einzeltest

= Modultest
Jedes Modul des Programms, z.B. eine Schleife, wird einzeln getestet.

7.6.6.2 Komponententest

Komponententests spielen eine Rolle vor dem Zusammenfügen aller Komponenten eines modular entwickelten Softwaresystems. Jede der Komponenten wird dabei einzeln überprüft.

7.6.6.3 Integrationstest

Die einzelnen Module werden zusammengefügt und als Ganzes getestet.
Da sie die obigen Testarten beide nur auf den Programmcode beziehen, werden sie von den Entwicklern durchgeführt.

7.6.6.4 Systemtest

Weil der Systemtest sich über das komplette Anwendungssystem erstreckt, werden die beteiligten Fachabteilungen hinzugezogen und man verwendet für alle Ein- und Ausgabedaten die endgültigen Bildschirmmasken, die Ein- und Ausgabeformate sowie Vordrucke.

7.6.6.5 Abnahmetest

= Abnahmeverfahren
Der Abnahmetest bildet das Ende des Softwaretests, an dem alle betroffenen Stellen beteiligt sind. Bei diesem sogenannten Probealarm muß die vollständige Erbringung aller Anforderungen verifiziert werden.

7.7 Qualitätsmerkmale für Software

7.7.1 aus Benutzersicht

Funktionserfüllung Maß an Übereinstimmung zwischen geplantem und realisiertem Umfang
Effizienz Ausmaß der Inanspruchnahme von Betriebsmitteln bei gegebener Funktionserfüllung
Zuverlässigkeit Stabilität
Benutzbarkeit umfasst alle SW-Eigenschaften, die dem Anwender oder Bediener ein einfaches, angenehmes und damit gleichzeitig effizientes und fehlerarmes Arbeiten mit dem Softwareprodukt gestatten.
Softwareergonomie ist die Gestaltung von Benutzerschnittstellen wie z.B. die
Dialoggestaltung oder Maskengestaltung
Sicherheit Systemeigenschaft, die folgendes verhindert:

- gefährliche Zustände des technischen Systems
- unbefugte Benutzung
- Zerstörung oder Verfälschung von Programmen und Daten
- Verlust von Daten

7.7.2 aus Entwicklersicht

Erweiterbarkeit Ziel, auch nach Fertigstellung des SW-Produkts den Leistungsumfang zu verändern
Wartbarkeit Zeit, die beansprucht wird, um Fehler zu beheben oder Änderungen durchzuführen, vorzunehmen
Übertragbarkeit, Portabilität Fähigkeit eines SW-Systems leicht in eine andere Hardware bzw. Software-Umgebung zu überführen

Wiederverwendbarkeit Anwendung von bereits entwickelten Software-Komponenten in anderen Umgebungen, für die sie ursprünglich nicht geplant waren

Über die Merkmale für SW-Qualität gibt es verschiedene Ansichten, ich habe hier mal die Hauptkriterien der DIN Norm 66272 zusammengetragen.

7.7.3 nach DIN 66272

Die Kriterien sind als Fragen formuliert, durch deren Beantwortung sich die Qualität feststellen lässt.

Funktionalität

Sind alle im Pflichtenheft geforderten Funktionen vorhanden und ausführbar?

=>Zuverlässigkeit

Zu welchem Grad erfüllt die SW dauerhaft die geforderten Funktionen?

=> **Verfügbarkeit**

Werden alle Funktionen richtig ausgeführt?

=> **Korrektheit**

Benutzbarkeit

(Benutzerfreundlichkeit): Wie schnell lässt sich der Umgang mit der SW vom Benutzer erlernen?

=> **Erlernbarkeit**

Wie einfach lässt sich die SW durch den Benutzer handhaben?

=> **Bedienbarkeit**

Effizienz:

Welches zeitliche Verhalten (Antwortzeiten im Dialogbetrieb, Laufzeiten Stapelbetrieb) und welchen Ressourcenverbrauch zeigt die SW unter gegebenen Systemvoraussetzungen

(Hardware, Betriebssystem Kommunikationseinrichtungen)?

Änderbarkeit:

Mit welchem Aufwand bzw. in welcher Zeit lassen sich ausführen Wie lässt sich der Aufwand für die Fehlererkennung und –behebung minimieren?

=> **Wartbarkeit**

Übertragbarkeit:

Mit welchem Aufwand lässt sich die SW an individuelle Anforderungen anpassen?

=> **Anpassbarkeit**

Lässt sich die SW ohne großen Aufwand in anderen Systemumgebungen zum Einsatz bringen?

=> **Portabilität**

Kann die SW bei Austausch des Rechners unverändert eingesetzt werden? => Skalierbarkeit

7.8 SW-Ergonomie

Zu den wichtigsten Forderungen der SW-Ergonomie gehört die beutzerfreundliche Gestaltung der Benutzerschnittstelle. Die Benutzerschnittstelle ermöglicht dem Benutzer, Dateiverzeichnisse einzusehen, Daten einzutragen, zu verändern, löschen oder sichern, Programmfunktionen zu starten, Hardware-Komponenten zu aktivieren und Transaktionen auszulösen.

Die SW-Ergonomie besitzt eine große Bedeutung für die SW-Qualität. Ihre Ziele bestehen in erster Linie darin, beim Arbeiten im Dialogbetrieb

1. die Arbeit nicht monoton oder ermüdend werden zu lassen
2. den Benutzer im richtigen Maß geistig zu fordern

Weiterhin ist die SW-Ergonomie weitgehend mit dem Qualitätskriterium Benutzbarkeit identisch, wozu Angaben gehören über:

- die Art der Benutzerschnittstelle
- die beim Benutzer vorausgesetzten Kenntnisse
- vorgesehene Anpassungsmöglichkeiten an die Bedürfnisse des Benutzers
- vorhandene Schutzmaßnahmen gegen Verletzungen des Urheberrechts

In DIN 66234, Teil 8 (für wen´s interessiert) sind die Anforderungen an ergonomische
Dialoggestaltung definiert:

Aufgabenangemessen: Unterstützt den Benutzer bei der Erledigung seiner Aufgaben effektiv und beansprucht ihn nicht zusätzlich durch ausschließlich systembedingten Zusatzaufwand (erreichbar durch: anschauliche Symbole und Fenster, Menüs, Masken, Funktionstasten und Makros für Routineaufgaben)

Selbstbeschreibend: Jeder Schritt ist dem Benutzer durch Rückmeldung unmittelbar verständlich oder wird ihm auf Anfrage erklärt erreichbar durch: Statusmeldungen, automatische Erklärungen, deutsche Hilfetexte)

Steuerbar: Benutzer kann Ablauf bis zu seinem Ziel gut beeinflussen (erreichbar durch: Löschbestätigung, UNDO-Funktion, Abbruchmöglichkeit, individuell festlegbare Bearbeitungs- und Antwortzeiten)

Erwartungskonform: Entspricht den Kenntnissen aus bisherigen Arbeitsabläufen und der Erfahrung des Benutzers sowie bzgl. Der verwendeten Begriffe allgemein anerkannten Übereinkünften. (erreichbar durch: Quittierung von Eingaben, Rückfragen des Systems, Zustandsanzeigen, Erläuterung von Unterbrechungen)

Fehlerrobust: Wenn trotz fehlerhafter Eingabe das beabsichtigte Arbeitsergebnis mit minimalem oder keinem Korrekturaufwand erreichbar ist. (erreichbar durch: Kein Systemabsturz bei Eingabefehlern, verständliche Fehlermeldungen, Hinweise auf Korrekturmaßnahmen, automatische Korrekturen bei geringfügigen Fehlern)

Seit Mai 1996 sind noch (in wer´s mag: DIN EN ISO 9241, Teil 10) die beiden Merkmale
Individualisierbarkeit und Lernförderlichkeit enthalten.
Weitere Empfehlungen sagen einiges aus über die Gestaltung der Benutzeroberfläche, wie z.B.:

- Zusammenfassung von verwandten Informationen zu Gruppen und Zuordnung zu festen Bildschirmbereichen (Status-, Arbeits-, Steuerungs-, Meldungs-) bzw. entsprechenden Fenstern.
- Sparsame Verwendung von Hervorhebungen (Blinken, Invertierung, Kontrastverstärkung, Farben)
- Abschaltbarkeit von optischen und akustischen Signalen.

7.9 Portabilität / Interoperabilität

Es gibt verschiedene Anforderungen an ein Informationssystem bei der Softwareentwicklung. Der Grad der Erfüllung der einzelnen Kriterien ist Ausdruck für die Softwarequalität. Eine der Anforderungen (neben: Fehlerfreiheit, Zuverlässigkeit, Benutzerfreundlichkeit, Effizienz, Modularität, Parametrisierung, Änderungsfreundlichkeit, Wartungsfreundlichkeit, Kompatibilität, Robustheit) ist die Portabilität. Portabilität (Übertragbarkeit) bedeutet, dass ein Informationssystem ohne besonderen Aufwand auch auf andere als die ursprünglich vorgesehene Hardware übernommen werden kann. Man spricht auch von Geräteunabhängigkeit.
Auch in Bezug auf Betriebssysteme wird der Begriff der Portabilität verwendet. Und zwar bei der Unterscheidung in proprietäre (nur auf der Hardware eines bestimmten Herstellers einsetzbare) und offene Systeme.
Die Hauptanforderungen an ein offenes Betriebssystem und die darunter laufende
Anwendungssoftware sind:

- **Portabilität:** Übertragbarkeit auf eine beliebige andere Systemumgebung (Hardware, Betriebssystem)
- **Interoperabilität:** Verwendbarkeit in beliebigen, insbesondere heterogenen Rechnernetzen
- **Skalierbarkeit:** Einsatzfähigkeit auf Rechnern aller Größenklassen

Dazu braucht man einen Source-Code, der von der Hardware verstanden werden muss:

- **Portabilität:** in Bezug auf ein System
- **Interoperabilität:** in Bezug auf verschiedene Systeme

7.10 ISO 9000

ISO ist nicht die Abkürzung für “International Organization for Standardization”. ISO ist der Name der Organisation, der sich ableitet vom griechischen Wort “isos”, der “gleich” bedeutet.

Alle ISO-Standards sind freiwillig. Die ISO hat nicht das Recht, ihre Normungen durchzusetzen, aber viele Länder nehmen die Standards als Grundlage für gesetzliche Regelungen.
Die ISO 9000-Serie ist eine Qualitätsnorm. Die ISO unterhält kein System um Unternehmen die Zertifizierung zu geben oder nachzuweisen. Dies wird von unabhängigen Organen in den jeweiligen Ländern gemacht, die aber durchaus Teil einer nationalen Organisation wie der DIN (Deutsches Institut für Normung) sein können, die der ISO angehört.
Die ISO 9000-Familie repräsentiert den internationalen Konsens einer guten Management- Praxis. Ihr Hauptziel ist es, Organisationen Richtlinien zu geben, aus was ein effektives QM- System bestehen sollte.
Die Familie beinhaltet ebenfalls drei Qualitätsversicherungsmodelle, die ISO 9001, ISO 9002 und die ISO 9003, mit denen das Qualitätssystem verglichen werden kann um zu sehen, ob es den Anforderungen der ISO 9000 genügt.

Unternehmen und Organisationen sollten diesen Vergleich selbst durchführen, um sicherzustellen, daß sie ihre Prozesse effektiv managen.

Die ISO 9004 beinhaltet letztlich den Aufbau, die Pflege und Weiterentwicklung von QM- Systemen. Sie enthält keine Forderungen, sondern nur Empfehlungen. Die Auswahl geeigneter Elemente und der Umfang, in dem sie übernommen werden, hängt vom zu bedienenden Markt ab, der Art der Produkte, den Produktionsprozessen sowie den Erfordernissen der Kunden. In der Praxis ist es aber sehr populär geworden, sich die Erfüllung der Normvorschriften von den unabhängigen Stellen zertifizieren zu lassen.
Die ISO 9000 ist kein Label oder Garantie für Produktqualität, sondern eine Bewertung der Prozesse, die die Qualität beeinflussen können. Sie bezieht sich auf die Herstellung des Produkts.

Die ISO 9000-Familie findet Anwendung im Bereich:

- Hardware
- Software
- Verfahrenstechnische Produkte
- Dienstleistungen

Sie ist für 4 Situationen vorgesehen:

- Anleitung zum Qualitätsmanagement
- vertraglich Regelungen zwischen Lieferanten und Kunden
- Genehmigung oder Registrierung durch einen Kunden
- Zertifizierung oder Registrierung durch eine unabhängige Stelle

Grundsätzlich sind in der ISO 9000-Norm die folgenden Punkte Richtlinien, die branchenspezifisch und betriebsabhängig beschrieben werden:
Verantwortung der Leitung

- Qualitätsmanagement
- Vertragsprüfung
- Designlenkung
- Lenkung der Dokumente und Daten
- Beschaffung
- Lenkung der vom Kunden beigestellten Produkte
- Kennzeichnung und Rückverfolgbarkeit von Produkten
- Prüfungen
- Prüfmittelüberwachung
- Prüfstatus
- Lenkung fehlerhafter Produkte
- Korrektur- und Vorbeugungsmaßnahmen
- Handhabung, Lagerung, Verpackung, Konservierung und Versand
- Lenkung von Qualitätsaufzeichnungen
- Interne Qualitätsaudits
- Schulung
- Wartung

- Statistische Methoden

Diese einzelnen Punkte werden jeweils genau beschrieben, nach dieser Beschreibung wird eine Prüfung vorgenommen und man bekommt das entsprechende Zertifikat.

7.11 Standardsoftware:

Standardsoftware ist SW, die nicht im Unternehmen oder speziell für das Unternehmen entwickelt wird, sondern komplett fertig oder im Baukastenprinzip vom Hardware-Hersteller oder von Software-Firmen gemietet bzw. gekauft wird. Im allgemeinen handelt es sich um komplette Programmpakete (wie z.B. MS-Office), die zusammen ein bestimmtet betriebliches Anwendungsgebiet wie die Finanzbuchhaltung abdecken. Die Programme werden i.d.R. noch den betrieblichen Bedürfnissen angepasst (z.B. SAP/R3). Kleinere Unternehmen passen normalerweise ihre Ablauforganisation der Standardsoftware an.
Große Unternehmen stellen sich, wenn es um die Einführung neuer SW geht, die Frage, ob sie lieber Individual-SW entwickeln (lassen) oder Standard-SW kaufen sollten. Für kleinere Unternehmen kommt es meist sowieso nur in Betracht, Standard-SW zu kaufen, da diese billiger zu haben ist als Individual-SW.
Standard-SW wird angeboten als:

- integriertes Programmpaket (Kompaktpaket), das alle Vorgänge einer Prozesskette unterstützt, z.B. Angebotsbearbeitung, Auftragsbearbeitung, Lagerverwaltung, Fakturierung, FiBu.
- Programmpaket für ein abgegrenztes betriebliches Arbeitsgebiet wie FiBu oder Personalwirtschaft.

Häufig gibt es von demselben Programm mehrere Versionen für unterschiedliche Rechnertypen oder Betriebssysteme und weitere Versionen mit verschiedenen Ausbaustufen. Weitere Standard-SW gibt es in den Gebieten Betriebssysteme, Übersetzungsprogramme, Office-Pakete.
Standard-SW ist definitionsgemäß weitgehend unabhängig von einer Branche, deshalb nennt man SW, die speziell für eine Branche zugeschnitten ist, auch Branchen-SW.
Wenn keine Qualitätsansprüche gestellt werden, dann lassen sich Public-Domain-Programme
(frei verfügbar, bis auf Versandkosten und CD) auch zur Standard-SW zählen, ebenso wie Shareware-Versionen, denen wichtige Programmteile fehlen.

7.11.1 Vorteile von Standartsoftware

- i.d.R. billiger als Eigenentwicklungen
- schnell einsetzbar, weil i.a. sofort verfügbar
- bei Eigenentwicklung anfallende Probleme entfallen weitgehend (wie Personalausfall, Terminüberschreitungen, Abstimmungsprobleme)
- Releasepflege und Updateerstellung durch den Anbieter der Standartsoftware (Euroeinführung)

- oft von besserer Qualität wegen größerer Erfahrungen der Entwickler des Anbieters

7.11.2 Nachteile

- Eigenschaften des Programmpakets stimmen oft nicht mit den Anforderungen des UN überein, so dass Anpassungen notwendig werden
- Häufig so allgemein entwickelt, dass schlechtere Performance oder Effizienz als bei Eigenentwicklungen gegeben ist
- Innerbetriebliche Ablauforganisation muss der Standard-SW angepasst werden (kann aber auch von Vorteil sein, wenn dadurch rationalisiert werden kann)
- Kann zu Schnittstellenproblemen mit anderen Programmen führen
- Überstürzte Einführung führt zu Fehlern, die später noch im laufenden Betrieb korrigiert werden müssen
- Bringt Anwender häufig in starke Abhängigkeit vom Anbieter
- Für die Anpassung (Adaption) von Standard-SW (Customizing) an die individuellen betrieblichen Anforderungen kommen im wesentlichen drei Möglichkeiten in Betracht:
    - Parametrisierung:
      gewünschte Programmfunktionen werden durch das Setzen von Parametern initialisiert. Voraussetzung ist, daß alle benötigten Programmfunktionen in der SW enthalten sind. Nachteil: SW von anfang an unhandlich und langsam.
    - Konfigurierung: (auch Modularisierung) nur die gewünschten Programmbausteine werden in das SW-Paket aufgenommen, dieses wird dann computergestützt generiert.
    - Individualprogrammierung: für die erforderlichen Anpassungen bzw. Ergänzungen wird individuell SW erstellt. Die so erstellte SW wird den Anforderungen des Kunden am besten gerecht, ist aber auch am teuersten.

8. SONSTIGES

8.1 Wirtschaftsinformatik

Die Wissenschaft, die sich mit der Konzeption, Entwicklung, Einführung, Wartung und Nutzung von Systemen beschäftigt, in denen die computergestützte Informationsverarbeitung im Unternehmen angewandt wird.

8.2 Stammdaten-Bewegungsdaten

8.2.1 Stammdaten

Stammdaten (engl. master data) sind zustandsorientierte Daten, die der Identifizierung, Klassifizierung und Charakterisierung von Sachverhalten dienen und die unverändert über einen längeren Zeitraum hinweg zur Verfügung stehen.

Änderungsdaten (engl. change data) sind abwicklungsorientierte Daten, die fallweise eine Änderung von Stammdaten auslösen. Das Berichtigen, Ergänzen und Löschen von Stammdaten in Datenbeständen wird als Änderungsdienst (engl. updating) bezeichnet.

8.2.2 Bewegungsdaten

Daten in der betrieblichen Datenverarbeitung, die Veränderungen von Zuständen beschreiben und dazu herangezogen werden, Stammdaten zu aktualisieren. Sie werden oft über einen gewissen Zeitraum gesammelt und dann verarbeitet
Bsp: Daten über Änderungen von Kundenadressen, mit denen die Kundenstammdaten aktualisiert werden

Bewegungsdaten stellen die Veränderung der Bestandsdaten dar, Darstellung in mengen-/wertmäßigen Zu-/Abgängen. Großteil der Daten, daher ist eine schnelle und wirtschaftliche Verarbeitung notwendig.
(außerdem gibt es: Bestandsdaten: Darstellung des sich ständig ändernden Bestands, immer aktualisiert, entweder permanent (Dialog) oder durch Fortschreibung (Stapelverarbeitung)

8.2.2.1 Anwendung (Beispiel aus der Materialwirtschaft)

Um ermitteln zu können, welche Materialien für die Leistungserstellung eines Unternehmens nach Art, Zeit, Menge bereitzustellen sind, muß zunächst der Bedarf an Materialien festgestellt werden. Mengen und Zeitpunkte der Beschaffung von Materialien hängen wesentlich davon ab, welche Höhe die Bestände an Materialien im Unternehmen aufweisen. Die Lagerbestände werden „naturellement“ EDV-mäßig geführt: die Stammdaten, die für eine überschaubare Zeit keiner Änderung unterliegen, sind im Materialstammsatz enthalten, der auf den Datenträgern gespeichert ist. Die Bewegungsdaten ergeben sich aus den Zugängen und Abgängen der vergangen Fertigungsperioden. Findet ein Materialabgang statt, wird die Menge vom Lagerbestand abgebucht und der Fertigungsdisposition per Programm zur Verfügung gestellt.

8.2.2.2 Problem der Unterscheidung

Oftmals läßt sich aber eine solch genaue Trennung nicht vornehmen, z.B. beim Gehalt. daher unterscheidet man hier häufig auch zwischen

> Zustands(orientierte)daten: beschreiben Zustände der realen Welt mit zeitlicher Gültigkeit (z.B. Materialbeschreibungen, werden sie selten geändert nennt man sie Stammdaten (Name) , sonst Bestandsdaten (Preis)
> Aktionsdaten: beschreiben den Übergang eines Zustand der realen Welt in einen anderen (Begleichen einer Rechnung), manche werden zeitpunktbezogen mit der Aktion, die die Änderung bewirkt, gespeichert, um sie später zu Analysezwecken heranzuziehen.

8.2.2.3 Ergebnis

Zustands- und Aktionsdaten dienen zur Abbildung der realen Welt und zur Verarbeitung in IV- Systemen (Informationsverarbeitungssystemen).
Man nennt sie daher auch Nutzdaten.

8.3 EDI/EDIFACT

8.3.1 Was ist EDI ?

Electronic Data Interchange (EDI) ist der elektronische Austausch strukturierter Daten von
Applikation zu Applikation basierend auf einem akzeptierten EDI-Standard.
Ein EDI-System übernimmt die automatische Steuerung der Ablaufprozesse für EDI-Geschäftsvorfälle.

8.3.1.1 Ablauf von EDI

Tätigkeiten, die ein Unternehmen A (Sender) ohne EDI heute machen muß:
ein Dokument erstellen und speichern, das Dokument ausdrucken und per Post versenden.

Unternehmen B (Empfänger) erhält dieses Dokument und muß ebenfalls eine manuelle Bearbeitung vornehmen:
das Dokument öffnen und erfassen, es auswerten und verteilen,
die Daten erfassen und verarbeiten (hier schleichen sich erfahrungsgemäß Fehler ein, etwa durch die Eingabe falscher Artikel- und/oder Rechnungsnummern).
Im Vergleich zu der aufwendigen Abwicklung ohne EDI sind die Abläufe der Geschäftsvorfälle auf der Basis des elektronischen Datenaustausches auf ein Minimum reduziert. Seitens des Senders ist notwendig:
das Dokument zu erstellen und die Datei zu senden.

Der Empfänger dagegen muß die Daten empfangen und automatisch verarbeiten.

Beispiel:
Unternehmen A nimmt mit Unternehmen B einen Austausch von Rechnungsdaten vor. Dazu muß A eine “Inhouse-Datei” erstellen, d. h. eine interne Dateibeschreibung, die das Fakturierungs- oder Warenwirtschaftssystem erkennt. Dieser individuelle Satzaufbau wird über einen Konverter in eine Datei nach der Norm umgesetzt. Diese normierte Datei erfordert wiederum ein normiertes Übertragungsprotokoll (FTP), das auf beiden Seiten (A und B) identisch sein sollte. Entweder durch eine point-to-point-Verbindung oder durch ein Mailbox-System tritt das Unternehmen A mit dem Unternehmen B in Kontakt. B hat nun die Datei erhalten und muss über einen Konverter aus dieser Normdatei wieder eine Datei kreieren, die es für seine Anwendung benötigt.

8.3.1.2 Verlauf der Übertragung:

Erstellung der Datei im eigenen Inhouse-System
Umformung in ein EDI-Format

Senden der Datei
Konvertierung der Datei seitens des Empfängers ins eigene Inhouse-Format
Bearbeitung der empfangenen Daten

Dabei ist es von Vorteil, wenn in den beteiligten Betrieben ein HOST- bzw. LAN-System angebunden ist, damit die Daten automatisch weiterverteilt werden können. Erst dadurch wird der volle Nutzen von EDI realisiert, das manuelle Aufbereiten und Erfassen entfällt und die Daten sind sofort und fehlerfrei im Anwendungsprogramm verfügbar.

8.3.1.3 Einsatzgebiete von EDI

Der Einsatz von EDI bietet sich für folgende Firmenbereiche und Geschäftsvorfälle an:

- Beschaffung/Vertrieb (Anfrage, Angebot/Produktdaten, Bestellung, Bestätigung, Preisdaten, Rechnung)
- Lager/Versand (Lieferschein, Frachtbrief, Feinabrufe)
- Finanzbereich (Zahlungsverkehr, Rechnungen, Valutierungen, Gutschriften)
- Konstruktion (CAD-Daten, Artikeldaten)
- Produktion (Lieferplan, produktionssynchrone Abrufe/Impulse
- Qualitätssicherung(Prüfberichte)

8.3.1.4 Welche Vorteile / Effekte bringt EDI für einen Betrieb ?

Kosteneffekte:

- Verringerung des Erfassungsaufwandes
- Reduktion der Übermittlungskosten
- Reduktion der administrativen Kosten für das Sammeln, Verteilen und Archivieren von Papierdokumenten
- Senkung von Bestellabwicklungskosten
- weniger Personalkosten

Zeiteffekte:

- deutliche Reduzierung von Rückfrageaufwand
- unmittelbare Auftragsbestätigung (Lieferant an Kunde)
- Verkürzung der Lieferzeiten (durch Beschleunigung der Abwicklung und kürzere Durchlaufzeiten)
- schnellere reaktionsmöglichkeiten auf Kundenwünsche

Qualitätseffekte:

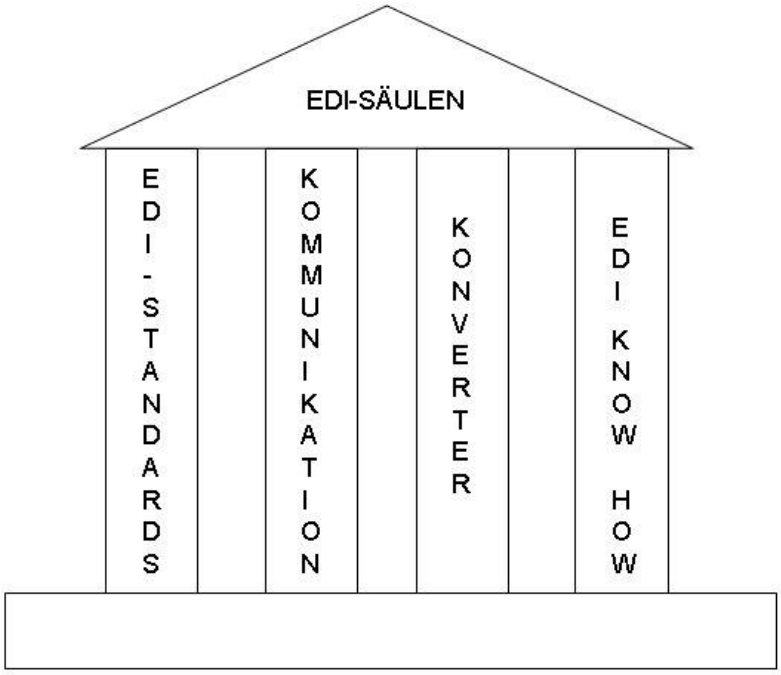
- keine Fehler durch manuelle Datenerfassung
- aktuellere Daten (durch automatische Stammdatenpflege)
- Überwindung von Sprachbarrieren und Vermeidung von Mißverständnissen

Strategische Effekte:

- Reduzierung der Lagerbestände (Just-in-Time)
- bessere Kontrolle der Materialbewegungen
- Steigerung der Planungs- und Dispositionssicherheit

- Beschleunigung des Zahlungsverkehrs
- Verbesserung von Lieferservice/Kundendienst
- erhöhte Flexibilität gegenüber Kunden und Zulieferern
- Erhöhung der internen Transparenz
- Optimierung des internen Workflow durch EDI-Integration (HOST/LAN)
- Straffung der Ablauf- und Aufbauorganisation
- Beseitigung von Medienbrüchen
- Umweltschutz durch verringertes Papieraufkommen

8.3.1.5 Voraussetzungen für EDI



Ein EDI-System unterstützt verschiedene EDI-Standards (siehe nächstes Kapitel), sorgt für die Konvertierungen der Dateien und stellt die Kommunikationsverbindungen sicher. Um EDI sinnvoll zu betreiben, ist zusätzlich ein fundiertes Know How nötig.

Als Kommunikationswege kommen in Frage:
point-to-point-Verbindungen
Übertragungsprotokolle: OFTP / X.400 / FTAM Übertragungswege: X.25 / ISDN / Wählleitung / Standleitung

indirekt via Mail / VAN-Anbieter
Übertragungsprotokolle: OFTP / X.400
Übertragungswege: Telebox 400 / IBM/IE / GEIS

Verschlüsselter Versand über das Internet

Bei der Einführung von EDI ist folgendes zu beachten:

- Analyse der Ist-Situation der Datenkommunikation (z.B. Datenstrukturen und Ablauforganisation)
- Feststellung von EDI-relevanten Geschäftsvorfällen und -partnern
- Betrachtung der EDV-Hard- und Softwareausstattung unter Einbeziehung der Datenhaltung Branchenanalyse über verfügbare Standards (z.B. EDIFACT, SEDAS, ODETTE)
- Auswahl der Technik
- Installation der Technik, Anpassen der Datenstrukturen und Programmierung der Schnittstellen
- Testbetrieb / Parallelbetrieb / Echtbetrieb (produktiv)
- evtl. Anbindung an HOST/LAN

8.3.1.6 EDI-Standards

Wie schon erwähnt gibt es für EDI verschiedene Standards. Welcher Standard eingesetzt wird, ist je nach Firmenbranche verschieden. Es gibt jedoch eine Entwicklung zum branchenübergreifenden, weltweiten Standard EDIFACT.

Die verschiedenen Standards unterscheiden sich in Inhalts- und Strukturfestlegung elektronischer Geschäftsvorfälle. Neben den unten aufgeführten, gibt es noch zahlreiche eigene Standards, die nur für den Einsatz zwischen zwei bestimmten Firmen entwickelt wurden.

Branchenstandards (Beispiele):

- VDA (Automobilbranche)
- EANCOM (Verbrauchermärkte)
- EDIFICE (Elektro)
- CEFIC (chemische Industrie)
- SWIFT (Banken)

Internationale Standards (Beispiele):

- Odette (Automobil, Frankreich)
- Tradacom ()
- ANSI X.12 (Amerika)

Weltweiter EDI-Standard:

- EDIFACT

8.3.2 EDIFACT

(Electronic Data Interchange For Administration, Commerce and Transport)
EDIFACT ist eine Universalsprache. Sie beinhaltet die Syntaxregeln auf Anwendungsebene für die Strukturierung von Benutzerdaten sowie die verbundenen Servicedaten für den elektronischen Austausch von Daten in einem offenen System. Die Regeln finden Anwendung im elektronischen Datenaustausch in Verwaltung, Wirtschaft und Transport.
In diesem Bereich ist eine Normierung insbesondere vorteilhaft, da in Handelsdokumenten meist identische Angaben bzw. Formularfelder vorhanden sind.

EDIFACT wurde 1988 als ISO-Norm 9735 verabschiedet. Die Dienst- und Protokollbeschreibungen für Kommunikation offener Systeme die verwendet werden, sind nicht Bestandteil dieser Norm.
Durch die Universalität sind hohe Anforderungen an die Sprache gestellt, die in aufwendigen Syntaxregeln definiert sind. Die Syntax ermöglicht die Datenelemente zu Segmenten und diese zu Nachrichten zu verbinden. Der "Wortschatz" der Sprache ist im Verzeichnis der Datenelemente, im Verzeichnis der Handelsdatenelemente und im Verzeichnis der Segmente festgehalten.

Eine Datei besteht aus einer oder mehreren Nachrichtengruppen bzw. Nachrichten. Diese Nachrichten sind jeweils in Segmente aufgegliedert, in diesen wiederum sind Datenelemente vorhanden, die jeweils nur einen einzigen Datenwert beinhalten.

Die exakte Aufschlüsselung der Segmente befindet sich im "Segment Directory", darin ist außerdem beschrieben, welche Datenelemente sich in welchem Segment und in welcher Reihenfolge unbedingt vorkommen müssen.
Durch Verwendung von Seperatoren (Trennzeichen) wird ein flexibler Aufbau ermöglicht.

Eine Besonderheit von EDIFACT ist, dass jedem Datenelement ein Qualifier vorangestellt ist und so eine variable Nutzung der Datenelemente möglich ist. Ein Datenelement kann also z.B. sowohl numerisch, als auch alphabetisch genutzt werden, nur durch Wahl des entsprechenden Qualifiers.

8.3.3 Zukunft von EDI

Zahlreiche neue Entwicklungen gab es in den letzten Jahren im Bereich XML/EDI. Ein erstes umfangreiches Konzept zur Nutzung von XML für den elektronischen Datenaustausch wurde mit dem „E-Business Framework" von der XML/EDI Group vorgeschlagen. Die im Jahr 1997 gegründete Gruppe ist ein Zusammenschluss von vielen Unternehmen und Einzelpersonen zur Förderung und Entwicklung von XML/EDI-Standards und –Produkten. Das E-Business Framework besteht aus einer Kombination von 5 Techniken:
XML, EDI, Templates, Agenten und Repository. XML stellt die Basistechnik dar, mit der die Nachrichteninhalte über das Internet übertragen werden. Templates beinhalten die Regeln für die Definition und Verarbeitung von XML-Dokumenten. Agenten interpretieren diese Templates und führen die notwendigen Verarbeitungsprozesse durch. Ein Repository bietet die semantische Grundlage für die Geschäftstransaktionen. Die Kombination der genannten Komponenten stellt ein System zur Verfügung, das nicht nur die reinen Daten der Geschäftsprozesse übertragen kann, sondern auch die zur Verarbeitung notwendige Information.
Die Projekte zum Aufbau von XML/EDI-Repositories lassen sich in zwei Gruppen unterscheiden: Zum einen wird versucht, bestehende EDI-Standarts nach XML zu überführen, zum anderen sollen unabhängig von vorhandenen Austauchsformaten standardisierte XML-Spezifikationen geschaffen werden.

8.4 ActiveX / DCOM

Kurzbeschreibung:
COM ist das Komponentenmodell von Microsoft. Es stellt ein objektbasiertes Programmiermodell dar, welches zusammen mit dem Netzwerkprotokoll DCOM die sprachenunabhängige Entwicklung verteilter Applikationen erlaubt. Populäre Techniken wie beispielsweise ActiveX-Steuerelemente oder -Dokumente basieren auf COM.

Längere Form:
Die von Microsoft entwickelten Standards für Componentware sind Component Object Model (COM) und Distributed Component Object Model (DCOM). DCOM ist ein Netzwerkprotokoll und erweitert COM um die Fernverarbeitungsarchitektur. Diese Standards werden vielfach verwendet. Heute wird jedes neue Subsystem von Microsoft als COM-Objekt implementiert. DCOM unterstützt TCP/IP, IPX/SPX, Apple Talk, HTTP und Named Pipes. Für die Plattformen Sun Solaris, Linux, Digital Unix, HP-UX, IBM OS/390 und AIX stehen bereits Komponenten und Entwicklungstools bereit, an Lösungen für weitere Plattformen wird gearbeitet. In Microsoft Windows 98 und Windows NT 4.0 ist DCOM nahtlos integriert und wird somit in kurzer Zeit weiter verbreitet sein als andere Standards.
Erst durch DCOM können die COM-Ziele Interoperabilität und Ortsunabhängigkeit vollständig realisiert werden.

DCOM ist von Microsoft als Protokoll der Anwendungsschicht für objektorientierte Remote Procedure Calls in verteilten, komponentenbasierten und heterogenen Systemen definiert. Während COM eine Spezifikation für den Aufbau von zur Zusammenarbeit fähiger Komponenten ist, ist Distributed COM einfach ein Netzwerkprotokoll höherer Ebene, das es COM-basierten Komponenten ermöglichen soll, über ein Netzwerk zusammenzuarbeiten.
Die Stellung von DCOM in einem Protokollstapel wird anhand eines Beispiels im Vergleich mit dem OSI-Schichtenmodell in Abbildung 5.1 deutlich.
DCOM ist kein eigenständiges Protokoll, das im Protokollstapel über RPC steht. Es benutzt vielmehr die Protocol Data Unit (PDU) des RPC-Protokolls für seine Zwecke, indem es Header und Nutzdaten von RPC durch Hinzufügen eigener Daten verändert. Die enge Beziehung zwischen RPC und DCOM lässt sich daran erkennen, dass das DCOM-Protokoll auch Object RPC (ORPC) genannt wird
DCOM wählt automatisch, abhängig von der Konfiguration der Client- und Server-Rechner, das am besten geeignete Netzwerkprotokoll niedrigerer Ebene; standardmäßig wird das verbindungslose UDP-Protokoll verwendet, da RPC selbst eine Empfangsbestätigung vorsieht. Außerdem sorgt RPC für die Reihenfolgegarantie der übermittelten Daten
EntireX DCOM ist eine Middleware, d.h. eine Menge von Diensten zwischen System- und Anwendungsplattform, die von der Software AG in Zusammenarbeit mit Microsoft entwickelt wurde. Sie erlaubt, DCOM auf verschiedene Mainframe- und UNIX-Plattformen zu portieren

ActiveX
ActiveX ist eine Komponente, die in eine HTML-Seite eingefügt wird, um eine Funktion zur Verfügung zu stellen, die nicht direkt in HTML enthalten ist.

Hierzu gehören Animationen, Transaktionen mit Kreditkarten oder steuerbare Simulationen und Berechnungen. ActiveX- Steuerelemente können in verschiedenen Programmiersprachen, z.B. Visual Basic, C++ etc. erstellt werden.
Nur für Microsoft Windows !!!

8.5 Dokumenten-Management-Systeme

8.5.1 Was ist ein Dokumenten-Management-System (DMS)?

DMS befassen sich ganz allgemein mit der Verwaltung nicht-codierter Informationen, wobei das elektronische Erfassen, Speichern, Wieder finden, Verarbeiten, Verteilen und Drucken als Image Processing (Imaging) bezeichnet wird. Da DMS die Vorgangssteuerung erheblich unterstützt können, werden sie zu den Groupware-Produkten gezählt.

8.5.2 Sinn und Zweck von Dokumenten-Management-Systeme

- dauerhafte Aufbewahrung von Schriftstücken und anderen Informationen
- mangelnde Verfügbarkeit von Originalen bei traditioneller Büroarbeit
- Redundanz
- Kopierkosten
- Raumbedarf
- lange Zugriffszeiten
- Kostenersparnis
- Raum- und Papierkosten
- schnellere Ablage
- Telefonkosten
- organisatorische Aspekte
- Produktivitätszuwachs
- schneller und paralleler Zugriff möglich
- verkürzte Durchlaufzeiten
- Kundenzufriedenheit

8.5.3 Funktionsweise von DMS

Zur Erfassung werden die Dokumente mit Scannern im Original gelesen und am Bildschirm angezeigt. Der Sachbearbeiter kann daraus diejenigen Abschnitte (Attributieren) anwählen, die gespeichert (“elektronisch archiviert”) werden sollen. Generell handelt es sich um bildpunkteweise Eingabe und Speicherung nicht codierter Daten. Das Wiederauffinden (retrieval) erfolgt über Verzeichnisse, die nach der traditionellen Hierarchie (Aktenschrank, Ordner, Dokument) aufgebaut sind. Nach der Bearbeitung (z.B. der Beantwortung der Anfrage) können diese Informationen durch codierte Informationen (z.B. den Text des Antwortschreibens) ergänzt werden.