1 EINFÜHRUNG

1.1 Zielsetzung

Zielsetzung

- Das Fach Datenbanktheorie soll Sie in die Lage versetzen, mit den Fachspezialisten aus den Bereichen Datenbanken (z.B. DB-Programmierer, -Administrator), Datenanalyse und –Design kommunizieren zu können.
- Dieser Kurs kann auch ein Anreiz sein, selber den Einstieg in die Welt der DB-Programmierung zu suchen.
- Dazu müssen Sie die wichtigsten Konzepte, Methoden und die dazugehörigen Begriffe dieser Fachgebiete kennen und verstehen.

Skript

- Dieses Skript vermittelt die Grundlagen der Datenmodellierung und von SQL, so dass der Leser in der Lage ist, selbständig ein Datenmodell zu entwickeln und mit Hilfe von SQL auch zu implementieren.
- Dieses Skript dient als Arbeitsunterlage und umfasst die wichtigsten Begriffsdefinitionen, Grafiken und Beispiele, die für den Stoff relevant sind.
- Dieses Skript ist **produkteunabhängig**, der Stoff bevorzugt keinen Datenbankhersteller; vielmehr werden Konzepte vermittelt, die für alle Datenbankprodukte gelten.

Ziele

- Denken in Zielen hilft, sich auf das Wesentliche zu konzentrieren.
- Ohne Ziel ist jedes Arbeitsergebnis richtig.
- Die Bestimmung der richtigen Ziele ist wichtiger als die Auswahl der richtigen Lösung.

Begriffe

In der Informatik herrscht ein Wirrwarr von Begriffen, da viele Autoren bestehende Begriffe zwar (meistens) inhaltlich identisch, jedoch anders formuliert wiedergeben. Es ist daher nicht von Bedeutung, ob Sie die Definition eines Begriffs wörtlich wiedergeben können, sondern, ob Sie den Inhalt eines Begriffs verstanden haben und Sie ihn anhand von Beispielen praktisch erläutern und auch anwenden können.

1.2 Definitionen

1.2.a Informationen

Einleitung
Bunt, grafisch und vielfältig präsentiert sich die Datenverarbeitung heute dem Laien wie dem Experten. Ein paar Mausklicks, und schon hat man seine Probleme gelöst – komplizierte Datenreihen werden mehrdimensional dargestellt. Wer Informationen benötigt, browst durchs Internet und holt sich die Multimedia-Show auf den Arbeitsplatz.

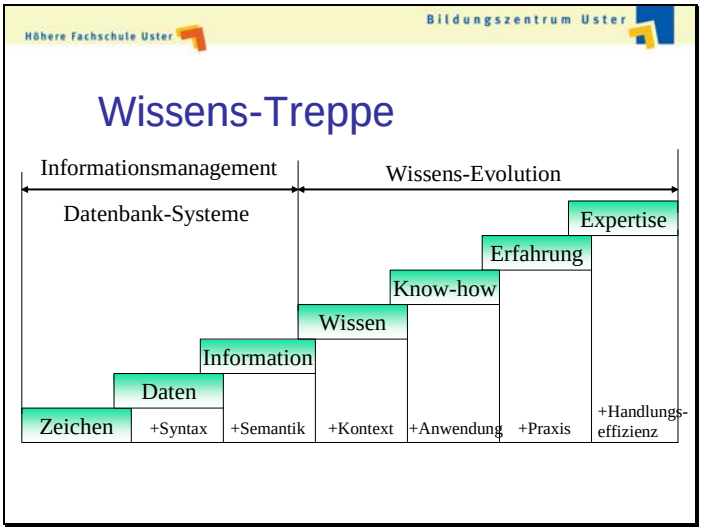
Diese Beispiele stehen für einen im Vergleich zu früher stark vereinfachten, intuitiven Zugriff auf Informationen aller Art. Dass diese verfügbar sind und sich nach verschiedensten Gesichtspunkten auswerten und darstellen lassen, muss allerdings erst einmal gewährleistet werden. Dafür gibt es Datenbanken. Richtig eingesetzt, eröffnen sie alle Möglichkeiten der Informationsverarbeitung. In unverstandener, unvollkommener Weise genutzt, verkommen sie zum Datenfriedhof, aus dem sich nach dem Zufallsprinzip taugliche oder unbrauchbare Informationen ziehen lassen.

Heute ertrinken wir in einem riesigen See von Daten. Viele Daten sind potenziell wertvoll, aber wir sind an einem Punkt angelangt, wo Daten so schnell gesammelt werden, dass sie möglicherweise nie genutzt werden können. Rohe Daten haben nur einen geringen Wert. Ihr Wert steigt, wenn sie in eine nutzbare Form gebracht werden, die aussagekräftige Informationen vermittelt. Datenbanksysteme sind ein leistungsfähiges Werkzeug, um aus Rohdaten und Strukturen wertvolle Informationen zu generieren.

Dieses Skript erklärt Datenbanken aus der Sicht des Anwendungsentwicklers. Es erläutert grundlegende Konzepte der Datenbanktechnologie und ihre Umsetzung bei der Entwicklung von Datenbanksystemen. Im Mittelpunkt stehen dabei die relationalen Datenbanken. Systeme, die auf dem relationalen Modell basieren, haben mittlerweile einen hohen Reifegrad erreicht. Die dabei gesetzten Massstäbe werden von anderen Systemen bisher nicht erreicht.

Andererseits haben manche Forderungen des relationalen Datenmodells noch keine durchgängige Umsetzung in verfügbaren Systemen erfahren, so dass hier noch Entwicklungspotential besteht. Unbestreitbar hat das relationale Datenmodell einige Mängel, die seine Brauchbarkeit speziell für die Verarbeitung komplexer Datenstrukturen einschränken. Hier setzen neue Konzepte, speziell die Objektorientierung an.

1.2.b Wissens-Treppe

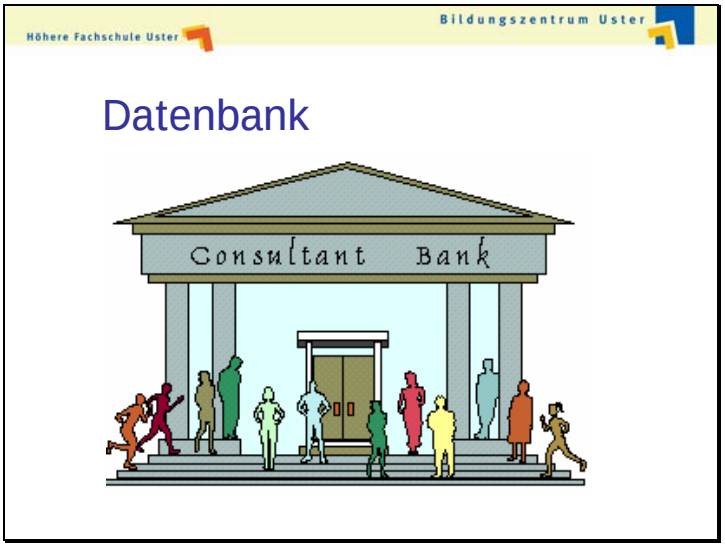


Geschäftswelt
Heutzutage haben die Unternehmen ihre Daten der operativen Geschäftsprozesse wie Finanz- und Rechnungswesen, Vertrieb, Logistik, Produktion usw. allesamt in elektronischen Medien gespeichert. Der Wert der in den Unternehmen vorhandenen Daten ist oft zitiert und häufig beschworen worden, diesen Wert tatsächlich zu nutzen, ist allerdings keine triviale Aufgabe und erfordert weit mehr als die Installation eines Tools.
Information und Wissen als Produktions- und Wettbewerbsfaktor wird zunehmend auf allen Ebenen der Unternehmen wahrgenommen.
Analog der Produktion anderer Güter liegt der Schlüssel in der bedarfsgerechten Bereitstellung der Informationen zum benötigten Zeitpunkt.

Definitionen
- Syntax Rechtschreibung, Grammatik
- Semantik Bedeutung, Inhalt
- Kontext Situationszusammenhang

Mensch im Fokus
Der qualitative Unterschied zwischen Information und Wissen besteht darin, dass Information einen punktuellen Charakter hat, während Wissen das Verständnis von Zusammenhängen (Einbindung in Kontexte) benötigt. Die Wissenstreppe (s.Grafik) relativiert denn auch die vermeintliche Informatik-Abhängigkeit für die Wissensentwicklung. Um aktuelle Wissensbestände zu erweitern, zu erneuern oder zu berichtigen, steht der Mensch im Fokus. Nur er kann mit Wahrnehmungen und Fähigkeiten den Entwicklungsprozess von Informationen zu Expertenwissen gestalten: Beobachten, Erkennen, Begreifen, Kombinieren sind Tätigkeiten der Wissensentwicklung und haben mit Managen nichts zu tun.

1.2.c Datenbank



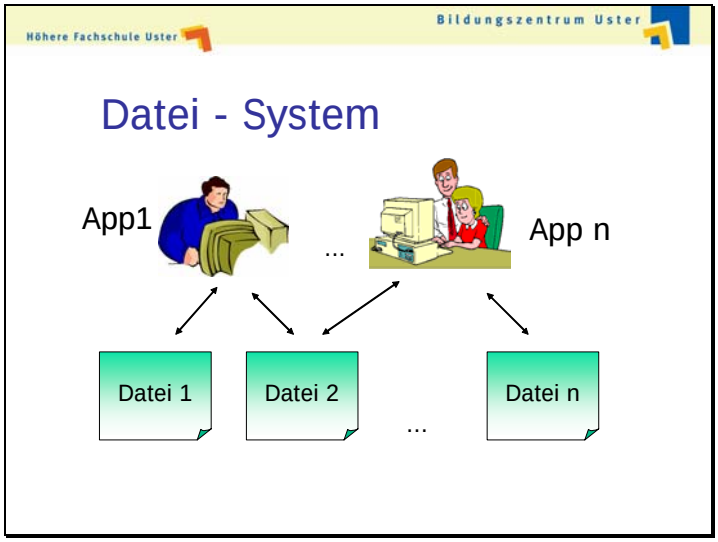
Bank
Daten ist die Pluralform von Datum, lateinisch für das Geschriebene, das Gegebene, Wert, Tatsache. Eine Datenbank ist nun eine Bank für Daten, also im übertragenen Sinne ein sicherer Aufbewahrungsort für (wertvolle) Informationen. Diese übertragene Bedeutung einer Art Institut für Daten gibt uns bereits einige Charakteristika von Datenbanken:

- Eine Datenbank hat die (langfristige) Aufbewahrung von Daten als Aufgabe.
- Die Sicherheit vor Verlusten ist eine Hauptmotivation, '*etwas'* auf die Bank zu bringen.
- Eine Bank bietet Dienstleistungen für mehrere Kunden an.

Eine Bank kostet Gebühren, auch eine Datenbank verursacht Kosten, also bringt man nur Daten auf die Bank, für die sich das tatsächlich lohnt. In einer Datenbank bewahren wir daher operative, für einen Betrieb sehr wichtige Daten, sehr grosse Datenbestände, langfristig vorzuhaltende und von vielen gleichzeitig zu nutzende Daten auf.

Persistenz
- Persistenz ist die Eigenschaft eines Objekts, durch welche es Zeit (Objekt existiert auch nach dem Verschwinden seines Schöpfers) und Raum (Objekt existiert in einer völlig anderen Umgebung) überwindet.
- Sollen Daten nach dem Runterfahren der Applikation immer noch verfügbar sein, so müssen sie persistent gemacht werden.
- Um diese Persistenz zu gewährleisten eignet sich eine Datenbank, aber auch ein Datei-System.
- Datenbank und Datei-System sind konkurrierende Systeme mit unterschiedlichen Eigenschaften, mit ungleichen Vor- und Nachteilen. Man muss sich daher immer zuerst vergewissern, welches von beiden Systemen das Zweckdienlichere ist.

1.2.d Datei - System



Systeme, bei denen die Daten in verschiedenen Dateien gespeichert und nicht miteinander verknüpft werden können, sind Datei-Systeme und gelten in unserem Sinne nicht als Datenbanksysteme.

Ein typisches Szenario gibt die folgende Auflistung wieder:

- Ein Textverarbeitung verwaltet Texte, Artikel und Adressen.
- Die Buchhaltung speichert ebenso Artikel- und Adress-Informationen.
- In der Lagerverwaltung werden Artikel und Aufträge benötigt und verwendet.

Probleme

- In diesem Szenario sind die Daten redundant, also mehrfach gespeichert. Etwa werden Artikel und Adressen von mehreren Anwendungen verwaltet. Die entstehenden Probleme sind die Verschwendung von Speicherplatz aber noch viel mehr das **Vergessen von lokalen Änderungen**.
- Ein Datei-System erlaubt es nicht, dass mehrere Benutzer oder Anwendungen **parallel** auf den gleichen Daten arbeiten können, ohne einander zu stören. Datenverlust durch unkontrolliertes Überschreiben kann entstehen.
- Datei-Systeme können **grosse Mengen von Daten nicht effizient** verarbeiten, da nur ein sequentieller und kein indizierter Zugriff auf Daten besteht.
- Dateien haben **kein Zugriffskontrollsystem** mit dem Benutzer klassifiziert werden können.
- Der grösste Nachteil von Datei-Systemen besteht darin, dass die **Datenstruktur in den Applikationen eincodiert** ist. D.h. sobald etwas an der Datenstruktur geändert wird, laufen sämtliche “alte“ Applikationen nicht mehr.

Aber

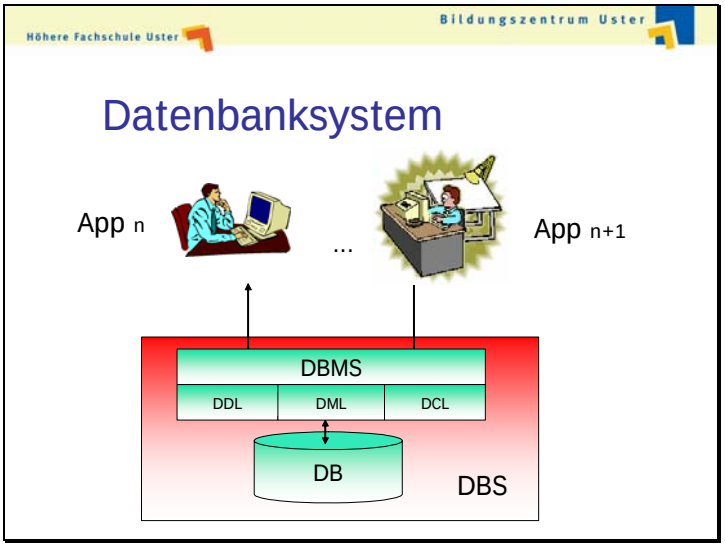
Ein Dateisystem hat seine Berechtigung, nicht in jedem Fall empfiehlt sich der Einsatz eines Datenbanksystems. Ein Filesystem hat gegenüber einem DBS einige Vorteile:

- ist einfach zu verstehen, braucht wenig Know-how
- ist schnell programmiert, oft werden gar keine Programmierkenntnisse gebraucht
- braucht keine Tools

Der Einsatz eines Dateisystems empfiehlt sich umso eher, ...

- je kleiner die Datenmenge
- je einfacher die Datenstruktur
- je weniger Änderungen in der Struktur zu erwarten sind
- je weniger Benutzer gleichzeitig auf Daten zugreifen
- je unwichtiger der Sicherheitsaspekt ist

1.2.e Datenbank- System



Die Probleme der Datei-Systeme können mit Hilfe der Datenbanktechnologie gelöst werden.

Alle Anwendungssoftware arbeitet mit denselben Daten, die in einer zentralen Datenhaltungskomponente verwaltet werden. Der Gesamtbestand der Daten wird nun als Datenbank bezeichnet. Anwendungssoftware kann nur über eine zusätzliche Layer – hier DBMS genannt - auf die Daten zugreifen.

Definition

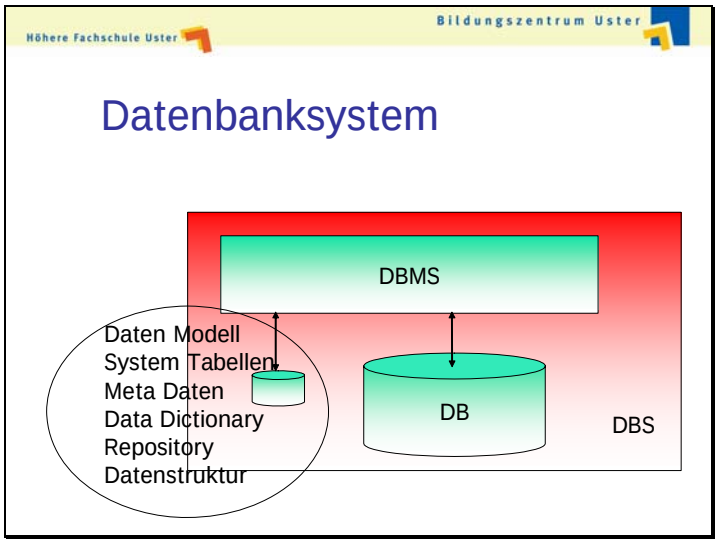
- **DB** (Datenbank) — Strukturierter, von DBMS verwalteter Datenbestand
- **DBMS** (DB-Managementsystem) — Software zur Verwaltung des Datenbestandes
- **DBS** (DB-System) — Ein Datenbank-System ist ein Programm, welches zwei Hauptkomponenten enthält; die erste dient der Speicherung von Daten und die zweite ihrer Verwaltung.

Eigenschaften von DBMS

- DBMS können grosse Datenmengen effizient verwalten.
- Sie bieten benutzergerechte Abfragesprachen an.
- Viele Benutzer können parallel auf Datenbanken arbeiten.
- Die Datenunabhängigkeit der Applikationssoftware wird durch ein Drei-Ebenen-Konzept gewährleistet (siehe 3 Schema Architektur).
- Zugriffskontrolle (kein unbefugter Zugriff) und Datensicherheit (kein ungewollter Datenverlust) werden vom System gewährleistet.

DBMSe können in drei wesentliche Komponenten gegliedert werden. Diese Dreigliederung wird im folgenden immer wieder verwendet.

- **Manipulationskomponente**: Erlaubt die Verarbeitung der Benutzerdaten. Die Verarbeitung kann hierbei mengenorientiert (d.h. mehrere Datensätze pro Befehl) erfolgen, beispielsweise in SQL. Die Sprache dieser Komponente wird DML (Data Manipulation Language) genannt.
- **Strukturierungskomponente**: Mittels dieser Komponente wird die Struktur der Benutzerdaten verwaltet. Diese Strukturinformationen werden in der rel. DB ebenfalls in Relationen abgelegt. Diese Komponente wird DDL (Data Definition Language) genannt.
- **Integritätskomponente**: Stellt die Integrität (Widerspruchsfreiheit) der Daten sicher. Diese Komponente wird DCL (Data Control Language) genannt.



Im Laufe der Jahre hat sich eine Basis-Funktionalität herauskristallisiert, die von einem DBS erwartet wird.

1. Keine Datenredundanz
Jedes Datum ist nur einmal abgespeichert. Redundanzen werden höchstens zur Steigerung der Performance eingeführt.

2. Operationen
Auf der Datenseite müssen Operationen möglich sein, die Speicherung, Suchen und Ändern der Daten ermöglichen.

3. Data Dictionary
System Tabellen, Meta Daten, Data Dictionary sind Synonyme. In ihnen ist die Struktur und Eigenschaften aller Benutzertabellen und sonstiger DB-Objekten abgelegt.

4. Benutzersichten
Für unterschiedliche Anwendungen sind unterschiedliche Sichten (Views) auf den Datenbestand notwendig.

5. Konsistenzüberwachung
Diese Integritätssicherung übernimmt die Gewährleistung der Korrektheit der Daten.

6. Zugriffskontrolle
Aufgabe der Zugriffskontrolle ist der Ausschluss unautorisierter Zugriffe auf die Daten.

7. Transaktionen
Logisch zusammengehörige Datenmanipulationen werden als unteilbare Einheit entweder vollständigen ausgeführt oder gar nicht ausgeführt, auf jeden Fall nichts dazwischen.

8. Synchronisation / Mehrbenutzerbetrieb
DBS vermitteln dem Benutzer den Eindruck, die Daten als einziger zu verwenden. Konkurrierende Transaktionen mehrerer Benutzer müssen synchronisiert werden, um gegenseitige Beeinflussungen, etwa versehentliche Schreibkonflikte auf gemeinsamen Daten, zu vermeiden.

9. Backup, Recorvery
Aufgabe der Datensicherung ist es, die Wiederherstellung von Daten nach Systemfehlern zu garantieren.

10. Logische und physische Datenunabhängigkeit
Strukturänderungen der Daten einerseits und Änderungen an der physischen Organisation der Datenspeicherung andererseits haben, soweit möglich, keinen Einfluss auf bestehende Applikationen.

1.3 Datenbanksysteme

1.3.a Geschichte

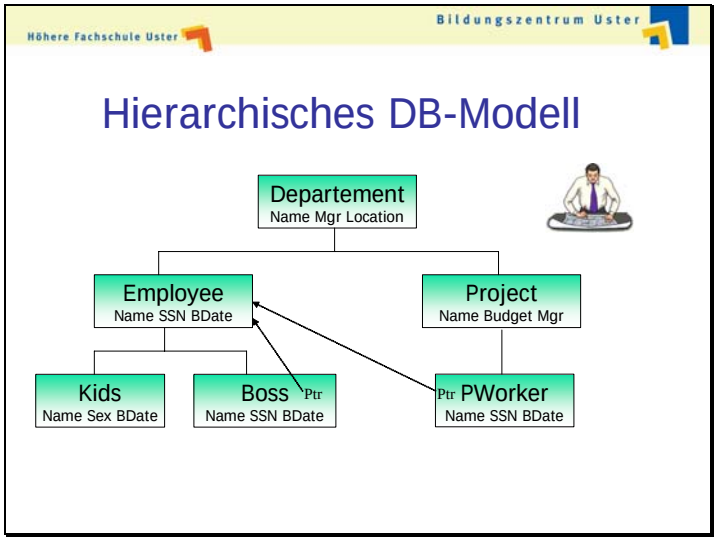
Erste Datenbankanwendungen
Die ersten Datenbanksysteme in den 60er-Jahren wurden auf grosse, unternehmensweite Probleme angewendet, wie beispielsweise für Flugreservierungssysteme oder für die Stücklistenverwaltung für Raumfahrzeuge der NASA. Damals waren Computer grosse Maschinen, deren Anschaffung und Betrieb teuer war. Nur grosse Unternehmen oder Regierungsstellen konnten sich Computer leisten, die für den Betrieb von Datenbanksystemen benötigt wurden.

Beginn der Datenmodellierung
Seit Anfang der sechziger Jahre begann man sich darüber Gedanken zu machen, wie Daten in eine `normale Form` gebracht und entsprechend dargestellt werden könnten. Der erste, der Daten in einer logischen Struktur präsentierte, war C. Bachmann 1969. Er kann durchaus als Vater der Darstellungstechnik von Datenmodellen bezeichnet werden. Etwas später entwickelte E. Codd seine Relationentheorie, mit der er versuchte, Daten mengentheoretisch zu erfassen und darzustellen, die später auch von P. Chen für sein Entity-Relationship-Modell (ERM) übernommen wurde. Codd formulierte damals die sogenannten Normalformen; er kann daher als Vater der Normalisierung betrachtet werden.

- Die zu jener Zeit geprägten Grundsätze und Regeln sind heute noch gültig.
- Seit Ende der siebziger Jahre ist das methodische Modellieren von Daten weit verbreitet.
- In den späten 80er und 90er Jahren kann die relationale Datenbanktechnik als etabliert gelten.

Bedeutung heute
Die Verwaltung von Datenbanksystemen gehört zu den häufigsten Aufgaben, die Computer heute erledigen. Tatsächlich ist nur Software für Textverarbeitung und für Tabellenkalkulation im Geschäftsleben noch stärker verbreitet.
Unternehmen verwalten ihre Kunden, Lagerbestände, Mitarbeiter, Anlagegüter, etc. in Datenbanksystemen. Firmen, Regierungen und Organisationen auf der ganzen Welt würden ohne Datenbanken nicht mehr funktionieren.

1.3.b Hierarchische Datenbank-Systeme



Prinzip
Das hierarchische DB-Modell basiert auf einer Baumstruktur (1 : n Beziehungen). Jeder Datenknoten hat genau einen Vaterknoten (ausser die Wurzel). Jeder Vaterknoten kann beliebig viele Kindknoten enthalten. Alle Daten stehen in einem streng hierarchischen Zusammenhang. Eine Verbindung von Knoten innerhalb einer Stufe ist nicht möglich. Dies ist eine starke Einschränkung. Daher ist dieses Modell nur für Problemstellungen sinnvoll, die bereits natürlicherweise eine Hierarchie beinhalten, wie z.B. Personalstrukturen, Klassifizierungen von Tieren, Pflanzen, Dienstgrade, Buchaufbau, etc.
Die Beziehungen zwischen den Daten werden hierfür mittels physischer Adresszeiger (Pointer) erstellt. Darin spiegelt sich auch die Nähe dieser Systeme zur technischen Lösung wieder. Die Verarbeitung der Daten erfolgt, indem in den Programmen diesen Adresszeigern gefolgt wird. Diese Verarbeitungsart in den Programmen wird Navigation genannt.

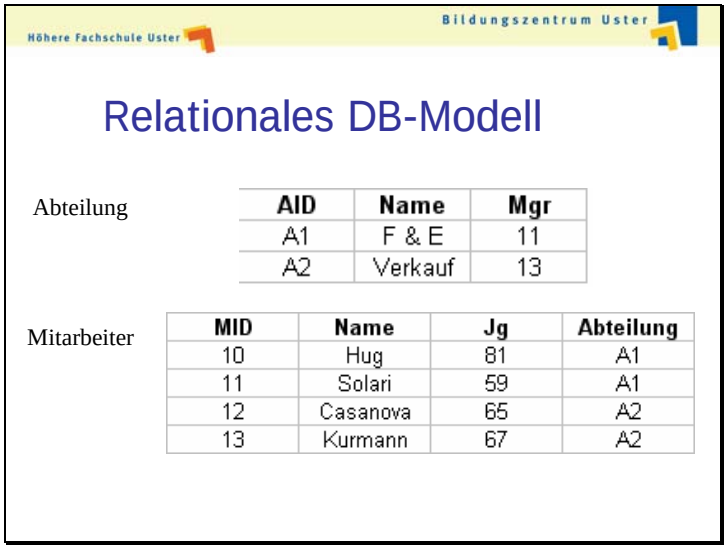
Vorteile
- leicht verständlich, leichte Benutzbarkeit
- schnell, da Zeiger physikalische Adressen sind

Nachteile
- Datenstruktur ist in den Applikationen eincodiert
- Komplexe Einfüge und Löschoperationen (ein Löschen eines Elternteils löscht gleichzeitig alle Kinddatensätze)
- Beziehungen zwischen Daten, die nicht dem strengen hierarchischen Prinzip entsprechen (z.B. ein Projekt Mitarbeiter der gleichzeitig auch Boss und Employee ist) können nur schwer implementiert werden, z.B. mittels Redundanzen oder virtuellen Vater/Kinder Beziehungen.

Bedeutung
- Hierarchische DBS stellen die einfachste, aber auch die realitätsfremdeste Struktur dar.
- Hierarchische DBS in Reinkultur existieren nicht mehr, weil sie zu realitätsfremd sind. Erweiterte hierarchische Systeme können diese Nachteile beseitigen.
- Heute immer noch im Einsatz (besonders in Grossunternehmen wie Banken, Versicherungen). IMS von IBM beherrscht den kommerziellen Markt für hierarchische DBS.
- Gegenwärtig starke Tendenz zur Migration auf relationale DBS.

Netzwerk DB-Modell
Netzwerkartige Systeme eliminieren den entscheidenden Mangel, welche den hierarchischen Systemen eigen war. Sie erlauben beliebige netzwerkartige Datenstrukturen. Dadurch konnten erstmals Datenstrukturen ohne Redundanzen abgebildet werden. Ansonsten sind die netzwerkartigen DBS technologisch noch eng verwandt mit den hierarchischen Systemen, da immer noch navigiert wird. Heute sind die Netzwerk DBS aber nur noch selten im Einsatz.

1.3.c Relationales DB-Modell



Während sich das hierarchische und netzwerkartige DBS an der technischen Lösung orientieren, löst sich das relationale DBS vom technischen Ansatz. Ausgangsidee ist, dass alle (wirklich alle) Daten in Tabellen festgehalten werden (einfache Lösungen sind auch gute Lösungen). Die Darstellung der Daten mittels Relationen ist für Benutzer und Entwickler einfach und klar verständlich. Selbst die Verknüpfung der Daten erfolgt nicht mehr mittels physischer Adresszeiger, sondern mittels identischer Feldeinträge.

Prinzip

- Konzeptionell ist ein rel. DBS eine Ansammlung von 2-dimensionalen Tabellen.
- Diese Tabellen sind untereinander beliebig durch Fremdschlüssel (keine HW-Adresse) kombinierbar.
- Dieses Modell basiert auf der Relationenalgebra. Dieser Zweig der Mathematik ist auch für die etwas gewöhnungsbedürftigen Begriffe verantwortlich, die im folgenden erklärt werden sollen.
- Die Reihenfolge der Spalten und Kolonnen spielt keine Rolle.

Theorie

Dieser Lösungsansatz wurde Anfangs der 70 Jahre von E. Codd ausgearbeitet und mit einer fundierten Theorie ausgestattet: Ein DBS ist, gemäss Codd, minimal rational, wenn dessen Funktionalität mindestens folgende drei Konzepte enthält:

- Die gesamten Informationen sind einheitlich in Relationen abgelegt.
- Der Benutzer sieht keine Verweisstrukturen (keine Pointers) zwischen den Relationen.
- Operationen zur Auswahl von Zeilen (Selektion) und Auswahl von Spalten (Projektion) sowie zur Verbindung (Join) von Relationeneinträgen sind definiert. Solche Operationen machen die DML (Data Manipulation Language) aus.

DBS sind nur dann voll relational, wenn sie über die genannten Konzepte hinaus bestimmte Integritätsbedingungen selbständig überprüfen und wenn zusätzliche, ergänzende Operationen zur Verfügung stehen.

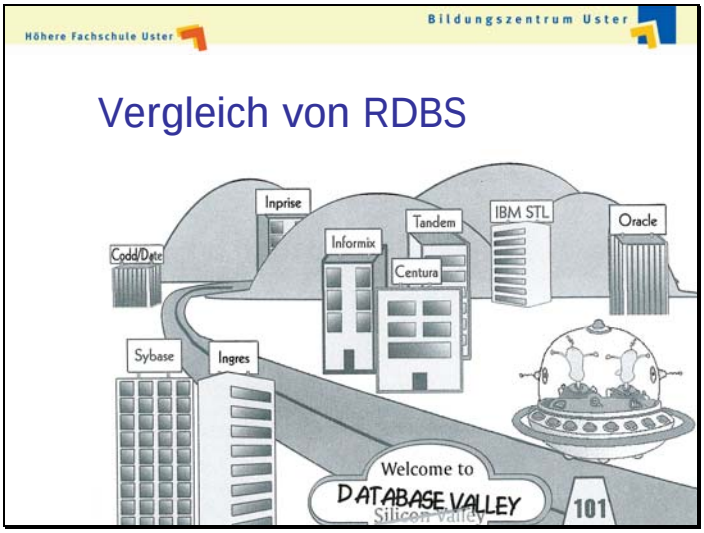
Vorteil

- Der wichtigste Vorteil ist die Tatsache, dass Sie in einem rel. DBS die Datenstruktur ändern können, ohne die Anwendungen ändern zu müssen, die mit der alten Struktur gearbeitet haben.

Nachteil

- langsam, da die Beziehungen zwischen den Tabellen bei jeder Abfrage dynamisch aufgebaut werden müssen

1.3.c.1 Vergleich von RDBS



Verschiedene Systeme
Grosse internationale Unternehmen und nationale Regierungen stellen erheblich andere Anforderungen an die Datenverwaltung als Privatpersonen oder Klein- und Mittelbetriebe. Benutzer grosser Datenbanksysteme benötigen riesige Kapazitäten und stellen hohe Anforderungen an das Leistungsverhalten. Sie sind auch bereit, den entsprechenden Preis zu zahlen, um diese Anforderungen zu erfüllen. Dieses Mass an Leistung übersteigt die Anforderungen einer Einzelperson oder einer KMU bei weitem und wäre ausserdem viel zu teuer. Deswegen gibt es verschiedene Datenbanksysteme für unterschiedliche Marktsegmente.

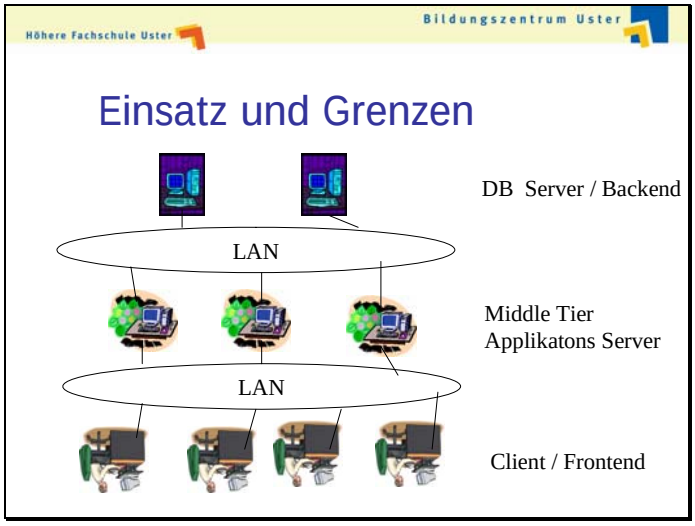
Grosse Systeme
- **Oracle** wird als Marktführer von rel. DBS bezeichnet. Oracle gibt es für fast alle Plattformen vom PC bis zum Grossrechner und für fast alle Betriebssysteme von Windows bis zu herstellerspezifischen Betriebssystemen.
- **MS SQL Server** etabliert sich zunehmend als der grosse Konkurrent von Oracle. Es läuft aber nur in Microsoft-Betriebsystemen.
- **DB2** ist *das* IMB-Datenbanksystem für diverse Plattformen geworden, so für OS/2 (DB2/2) und Unix (DB2/6000).
- **Ingres, Sybase, Informix** sind weitere Beispiele von grossen, multi-user Client-Server-Systemen.

Kleine Systeme, Access, FoxPro, Filemaker
Während die grossen Systeme in die Hände von professionellen Entwicklern gehören, sind Access und Co. darauf konzipiert, dass auch Leute mit keinen oder wenigen DB-Kenntnissen darauf arbeiten können. Die kleinen Systeme haben gegenüber den Grossen vielerlei Einschränkungen:
- viel kleinere Datenmengen (Anzahl Tabellen, Grösse der Tabellen)
- geringere Anzahl Benutzer
- reduzierte Datensicherheit
- keine Recovery - Möglichkeiten
- keine Stored Procedures
- keine Triggers
- keine Skalierbarkeit
- zum Teil kein Transaktions-Konzept

Der Erfolg der sogenannten kleinen Systeme basiert auf einer Reihe von Vorteilen:
- tieferer Preis
- viel einfachere Administrierung
- sehr einfache Gestaltung eines GUI via Drag and Drop
- integrierte Reporting Tools
- einfache Abfragemöglichkeiten ohne SQL - Kenntnisse

Freeware
- MySQL für einfache Internet - Anwendungen
- PostgreSQL dito

1.3.c.2 Skalierbarkeit



Wachstum erfordert skalierbare Systeme
Skalierbarkeit beschreibt folgende Arten von Systemänderungen:

- Vergrösserung des Datenbestandes
- wachsende Anzahl von Benutzern
- zunehmende Transaktionslast
- steigende Anforderungen durch neue Programmiertechniken

Skalierbare Systeme lösen das limitierende Grössenproblem, indem sie die Möglichkeit bieten, das Netzwerk, die Serversysteme, die Datenbank und die Anwendungen durch das Hinzufügen von zusätzlicher Hardware zu erweitern.

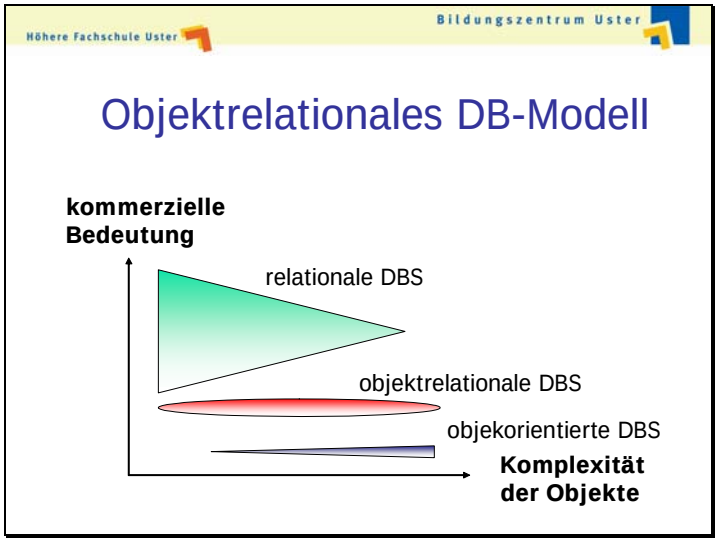
Einsatzgebiete

- Die klassischen Einsatzgebiete der rel. DBS sind Anwendungen im kommerziellen Bereich, die sich aus Buchhaltungs- und Katalogisierungsproblemen entwickelt haben. Ein typisches Beispiel wäre eine Bibliotheksausleihe. Derartige Anwendungen zeichnen sich durch einige Charakteristika aus: Es gibt viele Objekte (15‘000 Bücher, 300 Benutzer, 100 Ausleihvorgänge pro Woche, ...) aber vergleichsweise wenige Objekttypen (Buch, Benutzer, Ausleihungen). Objekte sind einfach strukturiert und verhältnismässig klein. Transaktionen sind kurz und betreffen wenige Objekte (etwa Ausleihen eines Buches), und die ausgeführten Operationen sind relativ unkompliziert, etwa einfache arithmetische Berechnungen.
- Die kommerziellen DBS sind heutzutage auf derartige Problemklassen hin optimiert. Normalerweise sind derartige, herkömmliche DBS überfordert, wenn die Anwendungsgebiete nicht zu den obigen Charakteristika passen.

Grenzen
CAD- oder andere technische Anwendungen betreffen viele Objekte, aber auch viele Objekttypen. Die einzelnen Objekte sind stark strukturierte Objekte, und Transaktionen (etwa das Erzeugen einer Variante eines Bauteils) betreffen viele Objekte und haben eine lange Dauer.
Relationale DBS sind nur bedingt daraufhin konzipiert, derartige Anwendungen effizient zu unterstützen. Aktuelle Entwicklungen wie objektorientierte DBS (OODBS) versuchen darum, besonders diese Art von Anwendungen adäquat zu handhaben.

Ein anderes Beispiel sind Expertensysteme. Expertensysteme verwalten oft wenige Objekte, die aber vielen Objekttypen zugeordnet sind. Die Operationen auf diesen Objekten sind kompliziert und langwierig, und komplexe Integritätsbedingungen sind zu berücksichtigen.
Auch derartige Anwendungen werden von heutigen Systemen nur bedingt unterstützt.

1.3.d Objektrelationales DB-Modell



Objektorientierte DBS
OODBS sind eines der populärsten Schlagworte im Datenbankbereich. Sie ermöglichen die Zusammenfassung von Daten in komplexen Objektstrukturen und bieten so adäquate Strukturierungskonzepte etwa für Ingenieur-Anwendungen. Das Ziel des objektorientierten Modells ist die reale Welt möglichst direkt zu beschreiben. Lösen die OODBS die RDBS ab? Das dachten viele IT-Fachleute vor zehn Jahren. Aber diese Entwicklung ist (bisher) nicht eingetreten.
Beispiele für kommerzielle Produkte sind: Poet, Jasmine, Versam, Objectivity, Matisse, ObjectStore, ...

Objektrelationale DBS
ORDBS haben die Integration der erfolgreichen kommerziellen relationalen DBS mit Techniken der Objektorientierung zum Ziel. Im Gegensatz zu den objektorientierten DBS werden ORDBS-Entwicklungen von den grossen Datenbankherstellern vorangetrieben und haben SQL3 beeinflusst. DB2 UDB von IBM ist ein System dieser Richtung. Oracle 8i und Informix Dynamic Server bieten von der Funktionalität ähnliches.

Das objektrelationale erweitert das relationale Datenmodell unter anderem um benutzerdefinierte Datentypen. Damit können objektorientierte Konstrukte wie **Komposition** und **Vererbung** einfach abgebildet werden: (Nachfolgendes muss zu diesem Zeitpunkt noch nicht verstanden werden)

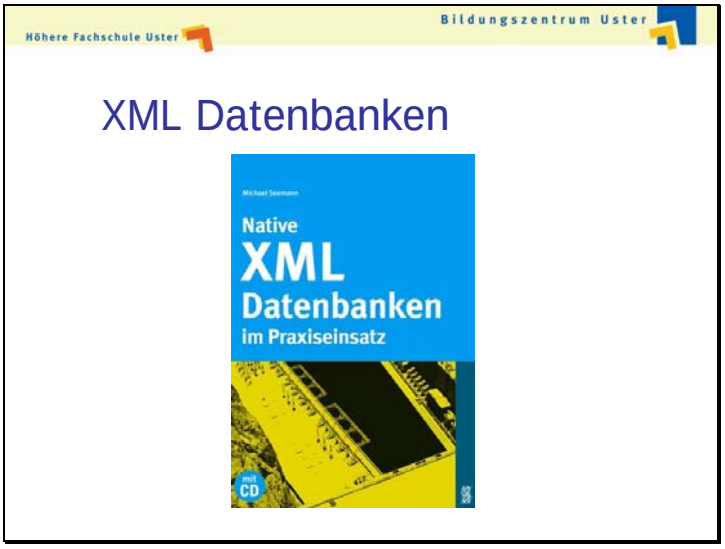
```
Create type ADRESSE as (    strasse     varchar(50),
                            stadt       varchar(25),
                            plz         int );

Create type PERSON as (     name        varchar(25),
                            adressbez   ADRESSE );

Create type ANGESTELLTER under PERSON as (    basisgehalt     decimal(9,2),
                                              method Gehalt() returns decimal(9,2) );
```

Die mit create type erzeugten Typen können jetzt weiterverwendet werden und zwar als Attribut einer Tabelle als auch als neuer Tabellentyp. Create table MITARBEITER of ANGESTELLTER

Abfragen gestalten sich immer noch intuitiv einfach:

```
Select   basisgehalt
From     MITARBEITER m
Where    m.ADRESSE.stadt = `Bern`
And      m.PERSON.name = `Müller`
```

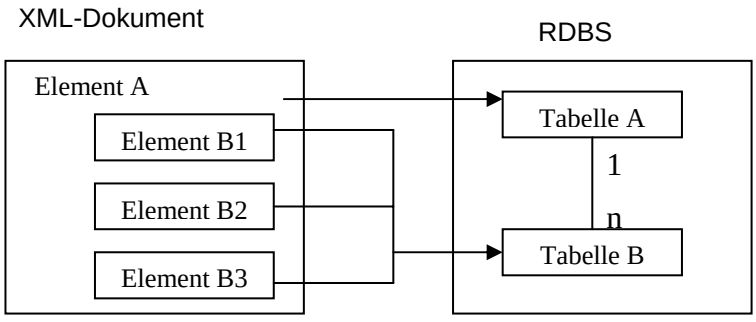
1.3.e XML Datenbanken



XML
Mit etwas Goodwill kann man ein XML-Dokument schon als eine Datenbank betrachten. So ein Dokument enthält Daten in einer selbstbeschreibenden und portierbaren Form und hat, mit den entsprechenden Erweiterungen, weitere Eigenschaften von Relationalen Datenbanken wie Schemas (DTD, Xml Schema), Abfragesprachen (XQuery, XPath) und Programmierschnittstellen (DOM).
Aber der Datenzugriff ist aufgrund des Parsens des Dokuments langsam, Daten sind nicht indexierbar und es existieren keine Sicherheits- und Transaktionsmechanismen. Hier aber liegen die Stärken von relationalen DBS. Ein logischer Schritt ist daher die XML-Dokumente in die bestehenden relationalen DB-Systeme zu integrieren.

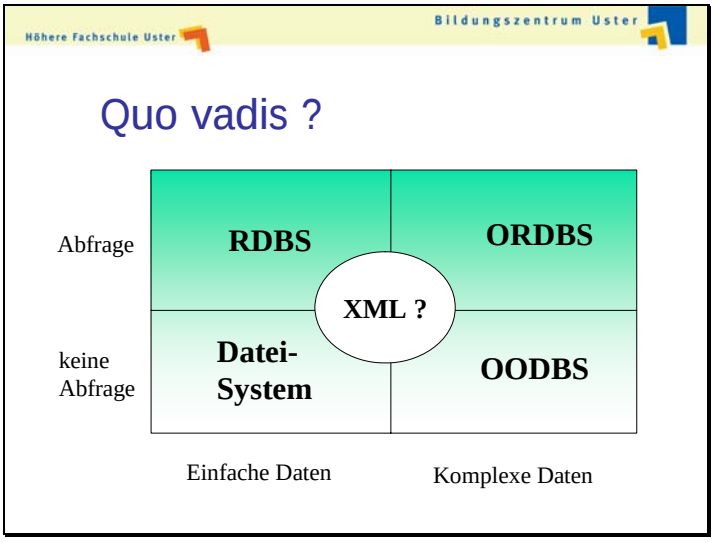
XML-Schnittstellen von RDBS

- Am einfachsten ist es, die XML-Dokumente als BLOB-Datentypen abzulegen. Damit können aber serverseitig keine Abfragen innerhalb des Xml-Blobs gemacht werden.
- Etliche DBS bieten Xml als eigenständigen Datentyp an. Mit entsprechenden Abfragesprachen können einzelne Xml-Elemente abgefragt werden.
- Bei der tabellenbasierten Abbildung von XML-Dokumenten (Shredding) werden die Daten jeder Elementebene in einer eigenen Tabelle abgelegt und die Tabellen untereinander mit Foreign Keys referenziert, siehe Abbildung.



Native XML-Datenbanken
Ein Beispiel für eine vollständig native XML-Datenbank ist Tamino von Software AG. Welche Prinzipien zur Datenverwaltung zu Grunde liegen, behält die Firma für sich. Man kann aber wohl davon ausgehen, dass ein erheblicher Teil aus dem Adabas-Know-how (hierarchisches DBS) in die Entwicklung eingeflossen ist. Abfragen können sowohl dokumentübergreifend als auch auf Dokumentebene durchgeführt werden.

1.3.f Quo vadis



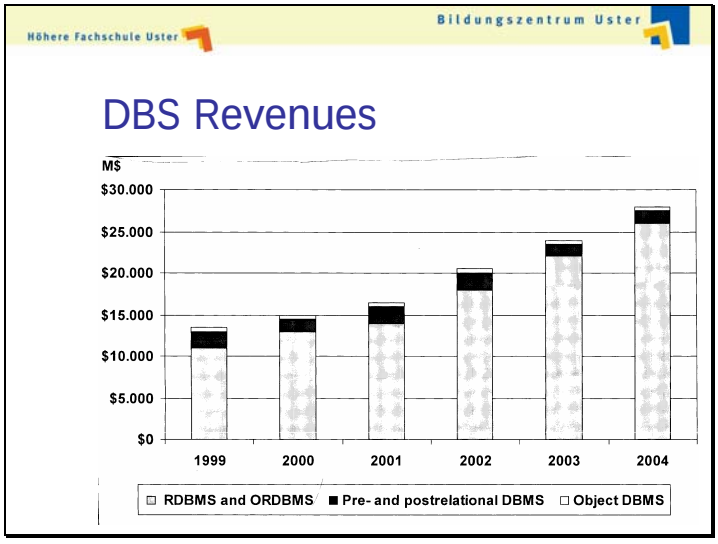
Marktentwicklung ?
Relationale DB-Systeme stellen heute die **aktuelle Datenbanktechnologie** dar und werden es auch noch eine ganze Weile bleiben. Dennoch ist zu bedenken, dass ein grosser Teil der heute gespeicherten Daten sich immer noch in hierarchischen DBS befindet.

Der Markt relationaler DBS hat ein Volumen von etwa 18 Milliarden US Dollar Umsatz pro Jahr. Der Markt **objektorientierter** DBS ist um den Faktor 100 kleiner, also ein Nischenmarkt. Die OODBS sind in der Praxis für hohen Durchsatz und grosse Datenmengen nicht brauchbar. Sie konnten sich deshalb nur in Spezialbereichen bewähren, wie z.B. bei ingenieurmässigen Entwurfsaufgaben, etwa im Maschinenbau.

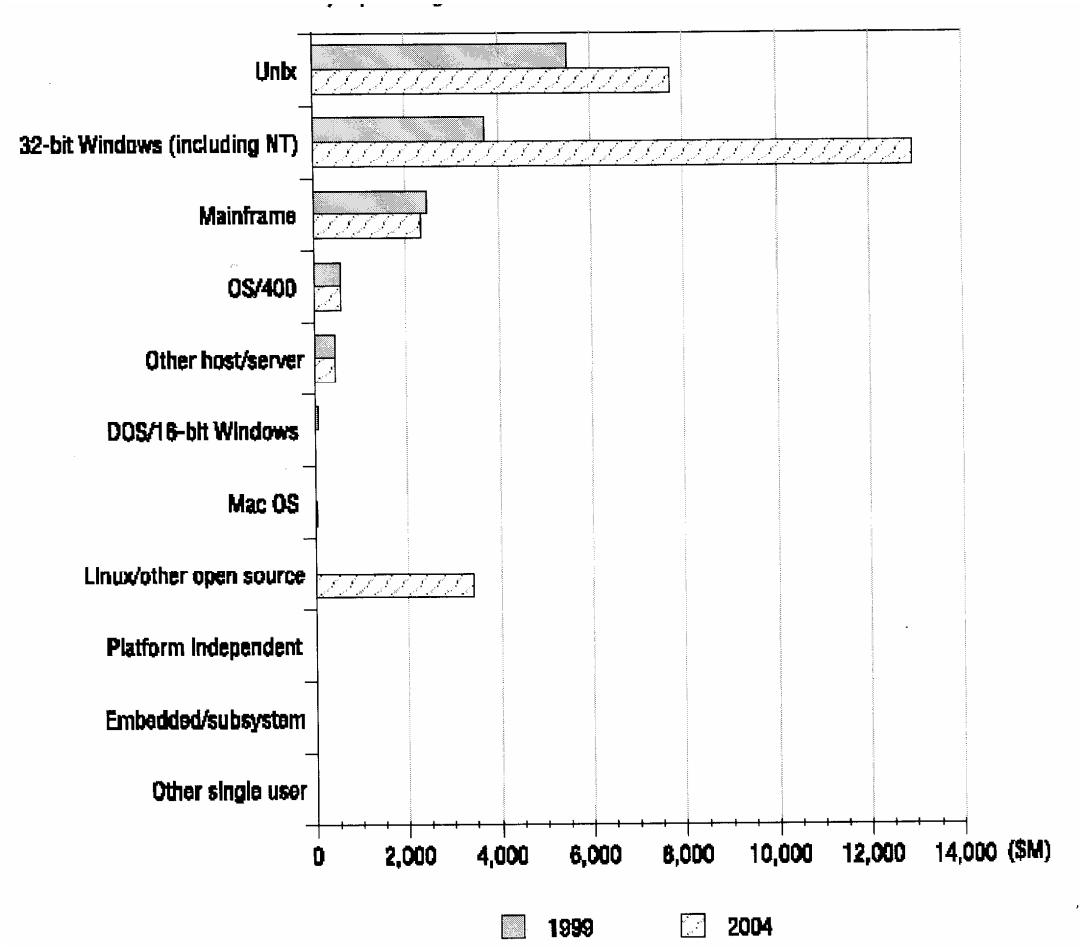
Für **XML Datenbanken** sind noch keine Prognosen bekannt. Entscheidend ist, dass es sich um ein Konzept handelt, das so einfach und universell einsetzbar ist, wie etwa das relationale Datenmodell vor 30 Jahren. Dazu kommt das riesige Anwendungsgebiet Internet, das die XML-Technologie voranbringt. Einiges spricht dafür, dass XML den nächsten Evolutionsschritt für Datenbanken auslösen könnte.

In der Breite der Anwendung wirklich tragfähig sind die **objektrelationalen** Systeme. Sie verbinden die Fähigkeit, komplexe Objekte zu verwalten, mit den bewährten Errungenschaften relationaler Systeme. Es ist nicht unwahrscheinlich, dass die objektrelationalen Systeme mit der Zeit marktbeherrschend werden.

1.3.g DBS, Revenues



Worldwide Enterprise Database Systems Revenue by Operating Environment, 1999 and 2004
Gartner Group 2001



1.4 Übungen

1.4.a Datenbanksysteme

A) Für hierarchische und netzwerkartige Datenbanksysteme gilt:

1. Daten lassen sich immer redundanzfrei ablegen
2. Sämtliche in der Realität auftretenden Datenstrukturen lassen sich 1:1 abbilden
3. Die Daten werden immer baumstrukturiert abgelegt
4. Strukturänderungen der Daten können nur mit grossem Aufwand vollzogen werden
5. Benutzer können Daten einfach abfragen

B) Für das relationale DBS gilt:

1. Die DML wird zur Definition der Datenstrukturen rel. DBS verwendet
2. Im relationalen DBS werden alle Daten in Relationen festgehalten
3. Die Daten werden baumstrukturiert abgelegt
4. Der Fremdschlüssel identifiziert die Zeilen der Tabellen eindeutig
5. Rel. DBS können die physische Datenunabhängigkeit nicht gewährleisten

C) Bei welchen DBS wird in Programmen nicht navigiert ?

1. hierarchische DBS
2. netzwerkartige DBS
3. relationale DBS
4. integrative DBS

D) In welchen DBS werden Verknüpfungen zwischen Daten nicht durch Adresszeiger dargestellt ?

1. hierarchisches DBS
2. netzwerkartiges DBS
3. relationale DBS
4. in keinem
5. in allen

E) Eine rel. Datenbanksystem ist ...

1. ein beliebiges Datenbankverwaltungssystem
2. ein relationales DBMS
3. ein relationales DBMS und deren Datenbasis
4. ein beliebiges DBMS und die verwendete Hardware

F) Eine DML ist ...

1. eine 3. Generationssprache
2. eine Menge von Datensätzen
3. eine Sprache zur Manipulation von Datenbeständen
4. eine Datei zum Protokollieren von Zugriffen
5. eine Konsistenzbedingung

G) Welche Eigenschaft weist ein rel. DBS nicht auf ?

1. Trennung der Daten von den Anwendungen
2. physische Datenabhängigkeit der Anwendungsprogramme

3. dauerhafte Nutzung der Daten
4. spezifische Datensichten für verschiedene Benutzer

H) Welche Aussagen sind für die Dateiverwaltung im Vergleich zum DBS korrekt ?

ja	nein	
☐	☐	Die Dateiverwaltung erlaubt ein einfacheres, besseres Sicherstellen der Integritätsanforderungen.
☐	☐	Die mit Dateiverwaltungssystemen entwickelten Anwendungen sind flexibler gegenüber zukünftigen Entwicklungen.
☐	☐	Je einfacher die Datenstruktur und je kleiner die Datenmenge umso eher sollte ein Dateisystem verwendet werden.
☐	☐	Je mehr Benutzer auf den gleichen Datenpool zugreifen, umso eher drängt sich ein DBS auf.

I) Was sind die kritischen Punkte eines relationalen DBS ?

_____

_____

K) Welche Vorteile hat ein Filesystem gegenüber einem DBS ?

_____

_____

_____