Kapitel 1: Einführung

Motivation: Warum Datenbanken?

"... kaum eine größere Informatikanwendung ist ohne DB-Unterstützung denkbar."

"DB-Systeme ... sind heute ein selbstverständliches Hilfsmittel der betrieblichen Organisation und Verwaltung geworden."

"Datenbanken ... als Schlüsseltechnologie für die effiziente Realisierung komplexer Informationssysteme ...“

Kennzeichen der Daten

- ❏ lange Lebensdauer (Jahre, Jahrzehnte)
- ❏ reguläre Strukturen
- ❏ große Datenobjekte, große Datenmengen
- ❏ stetig anwachsende integrierte Bestände (Giga-, Terabyte an Informationen)
- ❏ immer wiederkehrende Muster von Objektbeziehungen

Motivation: Warum Datenbanken? (2)

Einsatzbeispiele für Datenbanken

- ❏ Traditionell: Für kaufmännische informationsverarbeitende Aktivitäten in Verwaltungsabteilungen großer Organisationen wie Versicherungen, Banken, Telekommunikations- oder Versandunternehmen etc.
- ❏ Heute:
  - Adreßdatenbank auf jedem PC
  - Paper-Datenbank, Warburg Electronic Library
  - Kinoprogramm, Telefonbuch, Kleinanzeigenmarkt, ...
  - Stadtkarten
- ❏ Aber auch:
  - Modulbibliotheken (Cross References), ...
  - Repositories (StP/OMT, ABAP/4), ...

Motivation: Warum Datenbanken? (3)

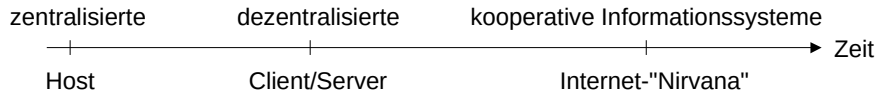
Anforderungen an die Datenverwaltung:

- ❏ Adäquate Repräsentation der Informationsstrukturen (betriebswirtschaftliche, statistische, textuelle, multimediale, grafische etc.)
- ❏ Flexible Zugriffsmodalitäten auf Informationsbestände (Hintergrundverarbeitung, interaktive Recherche, entfernter Zugriff etc.)
- ❏ Zugriff für verschiedene Benutzer und Anwendungsprogramme
- ❏ Konsistenz der Daten
  - Synchronisation
  - Fehlererholung
  - "konzeptuelle" Integrität
- ❏ Zugriffskontrolle, Datenschutz
- ❏ Effizienz beim Element- und Massendatenzugriff
- ❏ Unterstützung für Evolution der Daten und Programme

Informationssysteme

Wandel der Softwaresysteme:

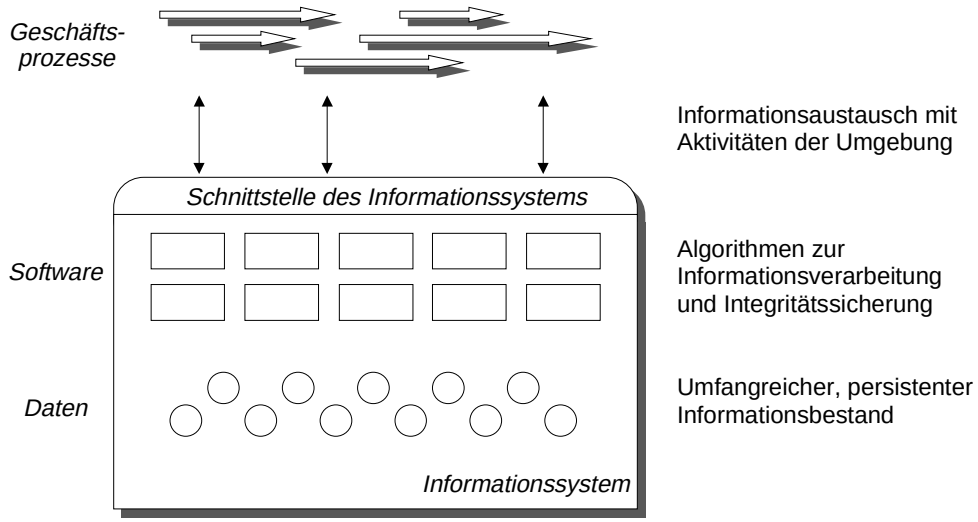
- ❏ Früher: Computer zur Lösung numerischer Berechnungsaufgaben
- ❏ Später Informationssysteme: Systeme zur Repräsentation und Verarbeitung von Informationen



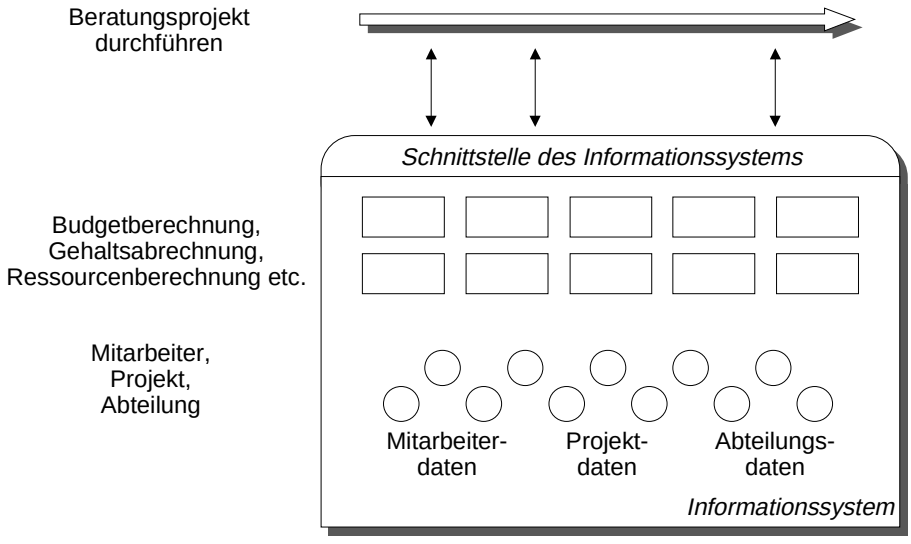
Charakteristika von Informationssystemen:

- ❏ *Persistenz* (Langlebigkeit) des Informationsbestands
- ❏ *Quantität* des regulär strukturierten Informationsbestands
- ❏ *Reaktivität* auf die Aktivitäten seiner Umgebung
- ❏ *Integrität* des Informationsbestands und der ein- und ausgehenden Informationen

Schematische Struktur eines Informationssystems

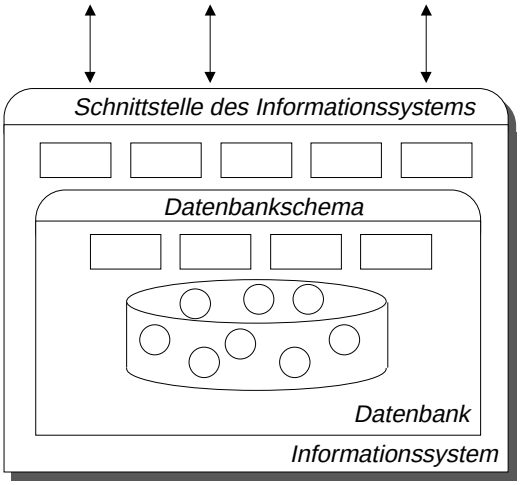


Beispiel: Ein Firmeninformationssystem



Datenbanksysteme

Realisierung eines Informationssystems mit einer Datenbank:

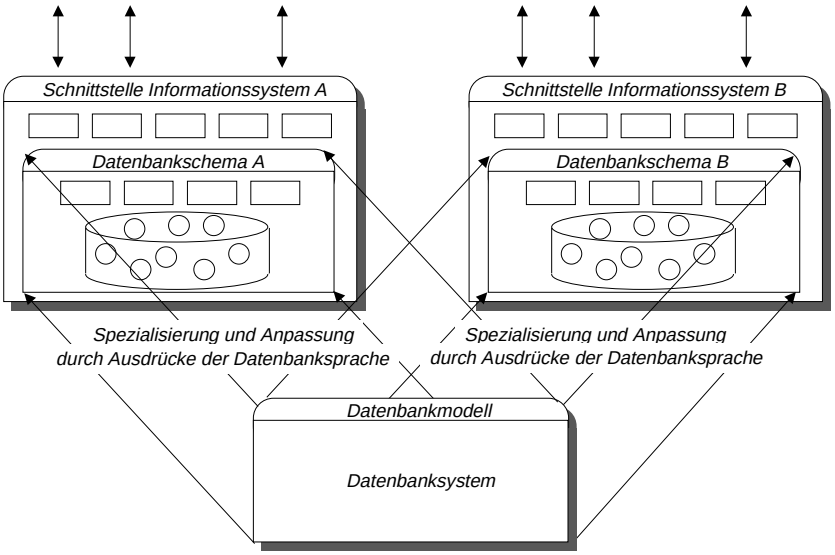


Algorithmen zur Informationsverarbeitung und Integritätssicherung

Dienste des Datenbanksystems zur Datenspeicherung und Integritätssicherung

Datenbankzustand

Dienstgestützte Informationssystem-Realisierung



Datenbankdienste (1)

Anforderungen an Datenbanksysteme:

- ❏ **Verpflichtende Datenbankdienste**: Persistente Datenspeicherung, Verwaltung einer Speicherhierarchie mit effizienter Datenselektion, dynamische Massendatenstrukturen, Synchronisationsprimitive, Dienstschnittstellen für den Datenbankzugriff aus Programmiersprachen.
- ❏ **Datenbankkerndienste**: Zusätzlich zu den verpflichtenden Datenbankdiensten: Persistenzabstraktion, Metadatenverwaltung, inkrementelle und dynamische Bindung, Datenunabhängigkeit durch Schemaschichtung, Iterationsabstraktion, transaktionale Integritätssicherung, Fehlererholung und Parallelitätskontrolle im Mehrbenutzerbetrieb.
- ❏ **Erweiterte Datenbankdienste**: Zusätzlich zu den Datenbankkerndiensten: Orthogonale Persistenz, Persistenzunabhängigkeit, deklarative Integritätssicherung, Verhaltensmodellierung, Ausnahmebehandlung, Unterstützung für Konversationen und langandauernde Aktivitäten, Benutzeroberflächen, erweiterte Dienstschnittstellen (4 GL, Benutzeroberfläche, Reportgenerator, entfernter Datenbankzugriff).

Datenbankdienste (2)

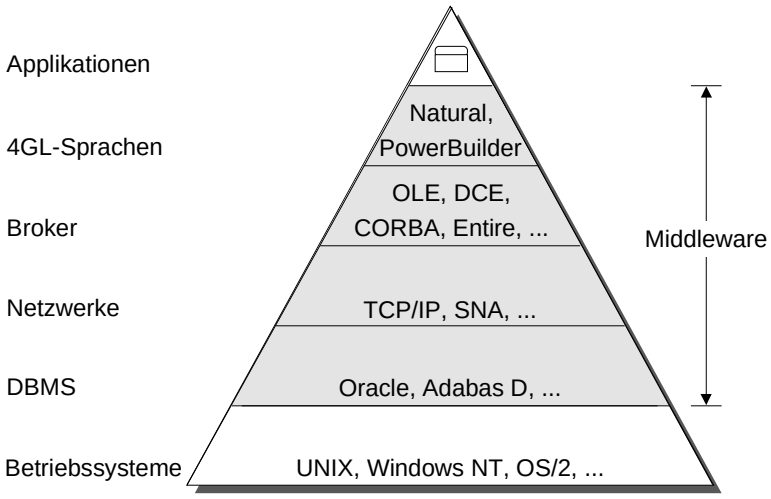
Vorteile ihrer Nutzung:

- ❏ Ausnutzung der Wiederverwendbarkeit
- ❏ Erhöhung der Portabilität
- ❏ Verbesserung der Systemskalierbarkeit (Migration eines Informationssystems von einem Einbenutzerdatenbanksystem über ein Mehrbenutzerdatenbanksystem zu einem verteilten Datenbanksystem)
- ❏ Qualitätsteigerung der auf ihnen basierenden Informationssysteme durch:
  - Datenabstraktion
  - Gemeinsame Informationsnutzung (Information Sharing)
  - Integrierte Datenbeschreibung (Information Integration)

Mögliche Nachteile ihrer Nutzung:

- ❏ Redundanz, Inkonsistenz, ineffizienter Datenzugriff

Datenbanksysteme als Middleware

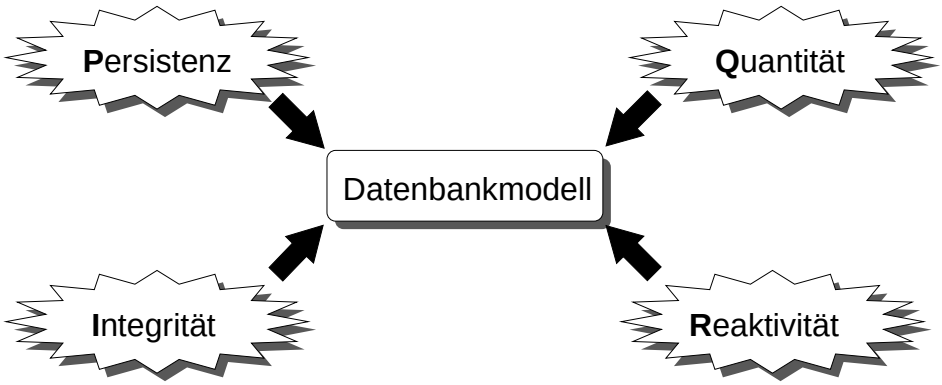


Analogie: Datenbanken ↔ Programmiersprachen

Datenbankschema (Datenstrukturen, Tabellenstrukturen)	**Modulschnittstelle** (Klassen, Typen)
Datenbank (Datenwerte, Tabelleninhalte)	**Modulimplementierung** (Objekte, Methoden)
Datenbankmodell (Relationales Modell)	**Programmiersprachen** (C++, ANSI C)
Datenbanksprache (SQL)	**Konkrete Programmiersprache** (GNU C++)
Datenbanksystem (Oracle 7.3)	**Konkreter Compiler** (GNU C++ 2.7.2 für Solaris)

Anforderungen an Datenbankmodelle

PQRI-Anforderungen:



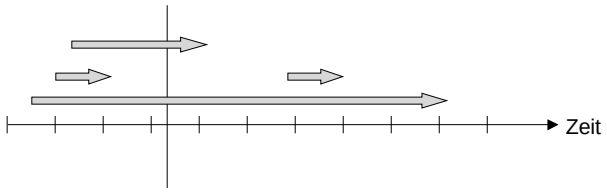
Beachte: *Synergieeffekt* bei Diensterbringung

Unterstützung der Persistenz (1)

Alle Daten, deren Lebensdauer die Lebensdauer eines einzelnen Betriebssystemprozesses überschreitet, werden als persistent bezeichnet.

Ein Informationssystem unterstützt die Speicherung unterschiedlichster Daten (mit unterschiedlicher Lebensdauer).

Ziel: Daten existieren solange, wie die Geschäftsprozesse, die sie unterstützen.



akueller Datenbankzustand

Unterstützung der Persistenz (2)

Beispiele**:**

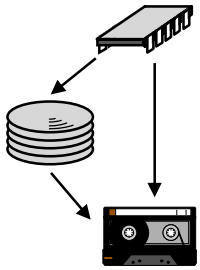
- ❏ Transienter Zustand einer langandauernden Aktivität an einem Sicherungspunkt (Minuten)
- ❏ Informationsanfragen (Stunden)
- ❏ Zustand eines Systemdialogs über mehrere Sitzungen (Tage)
- ❏ Versionen von Datenbeständen (Wochen, Monate)
- ❏ Protokollinformation (Jahre)
- ❏ Historische Datenbankzustände (Jahrzehnte)
- ❏ Datenbankzustand (unendlich)

Unterstützung der Persistenz (3)

- ❏ **Speicherhierarchie**: Datenspeicherung auf Sekundär- (Festplatte) und Tertiärspeichern (Band, Wechselplatte) ↔ Datenmanipulation im Primärspeicher (RAM)
- ❏ **Persistenzabstraktion**: Manipulation von Daten unabhängig von ihrer Lebensdauer
- ❏ **Meta- und Schemadaten** müssen zusammen mit den Nutzdaten abgelegt werden
- ❏ **Schemaevolution**: Umstrukturierung ohne Datenverlust
- ❏ **Inkrementelle und dynamische Bindungsmechanismen**: Bindung zwischen Variablennamen und persistenten Objekten → Datenunabhängigkeit
- ❏ **Fehlererholung**: Persistenz über Ausfälle hinweg

Unterstützung der Persistenz (4)

Kosten	Zugriffszeit	Durchsatz	Umfang	Lebensdauer
sehr teuer	Nanosekunden	50 MB/s	begrenzt	flüchtig
teuer	Millisekunden	5 MB/s	begrenzt	persistent
billig	> Sekunden	500 KB/s	"unendlich"	persistent



Unterstützung der Quantität

Durch die Regularität der Informationsstrukturen läßt sich der Informationsbestand in Klassen aufteilen, zwischen denen Beziehungen bestehen können (→ Massendatenstrukturen) und auf denen Invarianten definiert werden können (→ statische Typisierung).

- **Iterationsabstraktion**: stereotype imperative Programmuster ersetzt durch deklarative Beschreibungen
- **Effizienz** der Datenselektion zur Vermeidung teurer Sekundär- und Tertiärspeicherzugriffe

Beispiel:

```
firstPerson()
result:= true
while dbstatus = 0 do
  person = getPerson()
  result:= result and person.salary > 100
  nextPerson()
end
```

```
∀ p in Person: p.salary > 100
```

Unterstützung der Reaktivität

Ein Informationssystem reagiert auf eingehende Daten, antwortet auf Anfragen und löst selbständig Aktionen aus (Benutzeroberflächen, Dienstschnittstellen).

- **Synchronisation** nebenläufiger Aktivitäten auf dem gleichen Informationsbestand
- **Transaktionen**: zusammengehörige Aktionen müssen atomar, konsistenzerhaltend, isoliert und mit dauerhaftem Effekt ausgeführt werden
  (→ Integritätssicherung, Synchronisation, Fehlererholung)
- **Verhaltensmodellierung** erfolgt mit algorithmisch vollständigen Sprachen
  (→ Programmierspracheneinbettung)

Unterstützung der Integrität (1)

Aspekte der Integritätssicherung:

- **Binnenwirkung**: Konsistenz eines gekapselten langlebigen Systemzustandes
- **Außenwirkung**: Informationsstrukturen und Geschäftsanforderungen für alle Aktivitäten der Umgebung

Klassifizierung von Integritätsbedingungen:

- **Modellinhärente Integritätsbedingungen** werden implizit durch das Datenbankmodell erzwungen (z.B. Typisierung: das Alter einer Person ist ein Integer).
- **Applikationsspezifische Integritätsbedingungen** werden durch explizite Deklaration im Datenbankschema oder durch explizite Überprüfung in Algorithmen erzwungen (z.B. das Alter einer Person liegt zwischen 0 und 100).

Unterstützung der Integrität (2)

Klassifizierung nach der zeitlichen Ausdehnung:

- **Statische Integritätsbedingungen** müssen in jedem Zustand erfüllt sein (quantifizierte Prädikate, z.B. jeder Mitarbeiter arbeitet in genau einer Abteilung).
- **Dynamische Integritätsbedingungen** müssen in jeder Zustandsänderung erfüllt sein (Prädikattransformator, temporale Logiken, z.B. Gehälter von Mitarbeiter nehmen nie ab).

Integritätsbedingungen können auch nach ihrem Sichtbarkeits- und Wirkungsbereich unterschieden werden (z.B. Objekte, Klassen).

Integrität wird erhalten durch:

- **Deklarative Integritätssicherung**: Klauseln oder quantifizierte boolesche Prädikate im Datenbankschema (z.B. zu jedem Student s gehört genau eine Universität u)
- **Prozedurale Integritätssicherung**: explizit programmierte Tests (z.B. `if person.age < 18 then abort else insertStudent(person))`

Deklarative Integritätssicherung

Vorteile:

- Verbesserte Systemwartbarkeit
- Verbesserte Verstehbarkeit
- Optimierbarkeit

Probleme:

- Synchronisation verschiedener Integritätsbedingungen kann zu nicht-deterministischem Verhalten führen.
- Terminierung und Konsistenz von Ausnahmebehandlungen ist nicht garantiert.
- Lokalisierung von Integritätsbedingungen
- Inadäquate Sprachmittel für die Ausnahmebehandlung