



Abschnitt: Datenbanken

Folien basieren auf:

Database System Concepts

Avi Silberschatz, Henry F. Korth, S. Sudarshan

McGraw-Hill, ISBN 0-07-295886-3

Bzw. einer Vorlesung aus dem WS 2007/8





Überblick über das Gebiet Datenbanken

- Beispiel/Aufhänger
- Sinn und Zweck von Datenbanksystemen
- "Sichten" auf die Daten
- Datenbanksprachen
- Relationale Datenbanken
- Datenbankdesign
- Datenspeicherung und -abfrage
- Transaktionsmanagement
- Datenbankarchitekturen
- Datenbankbenutzer und -administratoren
- Gesamtstruktur eines DBMS
- Geschichte der Datenbanksysteme





Beispiel

- Die Verwaltungsabläufe an einer Universität sind heutzutage immer noch hochgradig papierbasiert
- Schrittweise werden einzelne Ämter/Abteilungen auf elektronische Datenver- und bearbeitung umgestellt;
Beispiel:
 - Prüfungsamt verwaltet Prüfungstermine, Noten und Daten der Studenten in Ihrer zugekauften Software XY
 - Das Bafög-Amt hat seine eigene Software YZ, um die Bafög-Empfänger unter den Studenten zu verwalten
 - Das Studentenwerk hat
- Zieht ein Student um, meldet er seine Adreßänderung vielleicht beim Bafög-Amt, vergisst aber Prüfungsamt und Studentenwerk
 - Inkonsistenter Datenbestand
 - Redundanzen





Beispiel

- Idealerweise (?) sollte es eine zentrale Instanz geben, welche alle Daten, die einen bestimmten Studenten betreffen, speichert/verwaltet
 - Database Management System
 - Datenschutz/Privatsphäre

- Stellen bzw. Ämter, welche auf Studentendaten zugreifen möchten, sollten nur auf die Studentendaten Zugriff haben, welche für ihre Arbeit relevant sind
 - sog. "Views"

- Updates wie z.B. Adressänderungen sind für alle sofort sichtbar

- Herausforderungen
 - Koordination gemeinsamer Zugriffe (Transaktionsmanagement)
 - Konsistenzerhaltung
 - Datensicherung
 - Performance





Datenbank Management System (DBMS)

- ein DBMS umfaßt
 - eine Kollektion von Daten ("Datenbank")
 - Programme, diese Kollektion von Daten zu verwalten
- DBMS Anwendungen:
 - Banken: alle Banktransaktionen
 - Fluggesellschaften: Reservierungen, Fahrpläne
 - Universitäten: Immatrikulation, Notenvergabe
 - Vertrieb: Kunden, Produkte, Verkäufe
 - Online-Shops: Bestellung, Tracking, Empfehlungen
 - Produzierendes Gewerbe: Produktion, Inventar, Bestellungen ...
 - Personalverwaltung: Arbeitnehmerdaten, Gehälter, Steuern
 - Webseiten/Content Management Systeme
- Wir treffen Datenbanken in allen Lebensbereichen an





Zweck von DBMS

- Früher speicherten Datenbankanwendungen ihre Daten direkt im Filesystem
- Nachteile:
 - Datenredundanz und -inkonsistenz
 - ▶ Viele Datenformate, Wiederholung derselben Information in vielen Dateien
 - Probleme beim Datenzugriff
 - ▶ Operationen auf den Daten müssen individuell programmiert werden
 - Datenisolation — konkurrierende, nicht-kompatible Datenformate
 - Integritätsprobleme
 - ▶ Integritätsbedingungen (z.B.. Kontostand > 0) sind im Programmcode versteckt anstatt explizit aufgeführt
 - ▶ Schwierigkeiten, neue Bedingungen hinzuzufügen oder bestehende zu ändern





Zweck von DBMS (Forts.)

- Nachteile der Nutzung des Filesystems (Forts.)
 - Atomizität von Änderungen
 - ▶ Fehler können die Datenbank in einem inkonsistenten Status zurücklassen, in welchem Änderungen nur partiell ausgeführt wurden
 - ▶ Beispiel: Überweisung von Geldern von einem Konto auf das andere sollte entweder vollständig ausgeführt werden oder gar nicht
 - Gleichzeitiger Zugriff durch mehrere Benutzer
 - ▶ Gleichzeitiger Zugriff für gute Performanz unersetzlich
 - ▶ Unkontrollierte, gleichzeitige Zugriffe können zu Inkonsistenzen führen
 - Bsp: Zwei Leute ändern Kontostand gleichzeitig
 - Sicherheitsprobleme
 - ▶ Schwierig, Datenzugriff einzuschränken
- DBMS lösen all diese Probleme





Abstraktionsebenen

- **Physikalische Ebene:** beschreibt wie ein Datensatz (z.B. Kunde) gespeichert wird.
- **Logische Ebene:** beschreibt die Daten, die in der Datenbank gespeichert sind, und die Relationen zwischen den Daten.

```
type customer = record
```

```
    customer_id : string;  
    customer_name : string;  
    customer_street : string;  
    customer_city : integer;
```

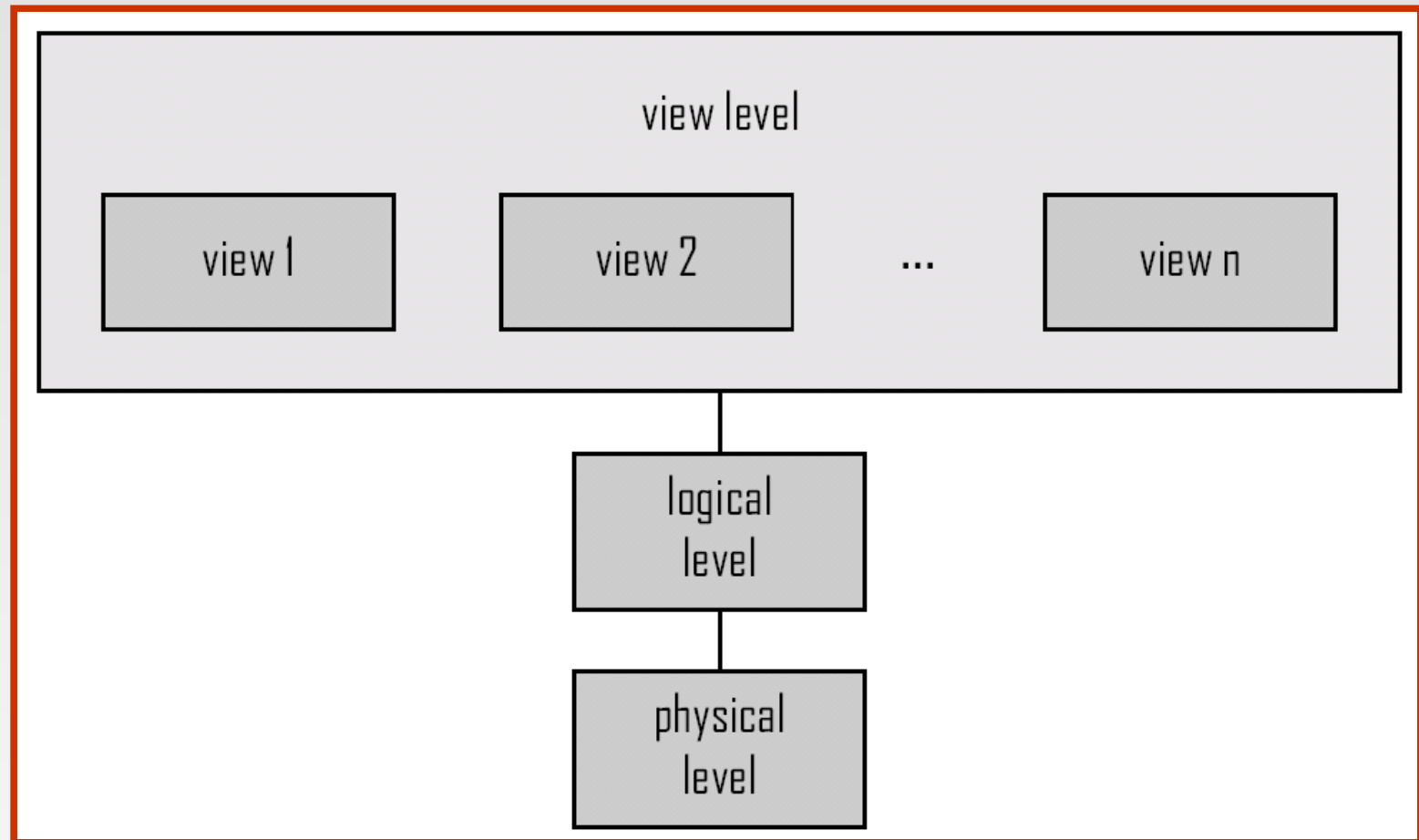
```
end;
```

- **Sichtenebene:** Anwendungsprogramme verstecken Details der Datentypen. Sichten können auch aus Sicherheitsgründen Teile der Datentypen verstecken (z.B. Gehalt).





"Views" auf die Daten





Instanzen und Schemata

- Ähnlich Typen und Variablen in Programmiersprachen
- **Schema** – die logische Struktur einer Datenbank
 - Bsp.: Die Datenbank besteht aus einer Menge von Kunden, Konten und der Beziehung zwischen diesen
 - Entspricht dem TYP einer Variable in einem Programm
 - **Physikalisches Schema:** Datenbankdesign auf physikalischer Ebene
 - **Logisches Schema:** Datenbankdesign auf logischer Ebene
- **Instanz** – der eigentliche Inhalt der Datenbank zu einem bestimmten Zeitpunkt
 - Entspricht dem WERT einer Variable in einem Programm
- **Physikalische Datenunabhängigkeit** – die Möglichkeit, das physikalische Schema zu ändern, ohne das logische Schema abändern zu müssen
 - Anwendungen hängen vom logischen Schema ab
 - Im Allgemeinen sollten die Schnittstellen zwischen den Ebenen und Komponenten wohldefiniert sein, sodass Änderungen in einem Teil keinen großen Einfluß auf andere Teile haben.





Datenmodellierung

- Handwerkszeug um Beschreibungen zu erstellen für
 - Daten
 - Beziehungen zwischen den Daten
 - Semantik der Daten
 - Einschränkungen über den Daten
- Relationales Modell
- Entity-Relationship Modell
- [Objektbasierte Datenmodelle (Objektorientiert und Objektrelational)]
- [Semistrukturierte Datenmodelle (XML)]





Data Manipulation Language (DML)

- Sprache, um auf die Daten zuzugreifen und sie zu manipulieren
 - DML wird oft auch Anfragesprache/Query Language genannt
- Zwei Sprachklassen
 - **Prozedural** – Benutzer spezifiziert, welche Daten gebraucht werden, und wie sie ermittelt werden können
 - **Deklarativ (nichtprozedural)** – Benutzer spezifiziert welche Daten gebraucht werden, ohne zu sagen, wie sie genau ermittelt werden sollen
- SQL ist die verbreitetste Anfragesprache





Data Definition Language (DDL)

- Spezifikationssprache, um Datenbankschema zu definieren

Beispiel: **create table** *account* (
 account-number **char**(10),
 balance **integer**)

- DDL Compiler erzeugt eine Menge von Tabellen, im *data dictionary* gespeichert werden
- Data dictionary enthält auch Metainformationen(Daten über Daten)
 - Datenbankschema
 - Data *storage and definition* language
 - ▶ Spezifiziert die Speicherstruktur und die Zugriffsmethoden
 - Integritätsbedingungen
 - ▶ Bereichsbeschränkungen
 - ▶ Referentielle Integrität (Referenzbedingung in SQL)
 - ▶ Assertions
 - Zugriffsbeschränkungen





Relationales Modell

- Beispiel einer Tabelle im relationalen Modell

Attribute

<i>customer_id</i>	<i>customer_name</i>	<i>customer_street</i>	<i>customer_city</i>	<i>account_number</i>
192-83-7465	Johnson	12 Alma St.	Palo Alto	A-101
192-83-7465	Johnson	12 Alma St.	Palo Alto	A-201
677-89-9011	Hayes	3 Main St.	Harrison	A-102
182-73-6091	Turner	123 Putnam St.	Stamford	A-305
321-12-3123	Jones	100 Main St.	Harrison	A-217
336-66-9999	Lindsay	175 Park Ave.	Pittsfield	A-222
019-28-3746	Smith	72 North St.	Rye	A-201





Beispiel einer relationalen Datenbank

<i>customer_id</i>	<i>customer_name</i>	<i>customer_street</i>	<i>customer_city</i>
192-83-7465	Johnson	12 Alma St.	Palo Alto
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsay	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

(a) The *customer* table

<i>account_number</i>	<i>balance</i>
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

(b) The *account* table

<i>customer_id</i>	<i>account_number</i>
192-83-7465	A-101
192-83-7465	A-201
019-28-3746	A-215
677-89-9011	A-102
182-73-6091	A-305
321-12-3123	A-217
336-66-9999	A-222
019-28-3746	A-201

(c) The *depositor* table





SQL

- **SQL:** sehr verbreitete, nicht-prozedurale Anfragesprache
 - Bsp.: Finde den Namen des Kunden mit Kundennr. 192-83-7465

```
select  customer.customer_name
from    customer
where   customer.customer_id = '192-83-7465'
```
 - Bsp.: Finde die Kontostände aller Konten des Kunden mit Kundennr. 192-83-7465

```
select  account.balance
from    depositor, account
where   depositor.customer_id = '192-83-7465' and
        depositor.account_number = account.account_number
```
- Anwendungsprogramme greifen auf Datenbanken typischerweise zu mittels
 - Spracherweiterungen, die es erlauben, SQL einzubetten
 - Anwendungsprogramminterfaces (API) (z.B., ODBC/JDBC), welche SQL Queries an eine Datenbank absetzen können





Datenbankdesign

Der Prozess die Gesamtstruktur der Datenbank zu entwerfen:

- Logisches Design – Entscheidung über das Datenbankschema.
 - Businessentscheidung – Welche Attribute sollten wir in der Datenbank speichern?
 - Informatikentscheidung – Welche Schemata sollten wir haben, und wie sollten die Attribute über die Schemata verteilt sein?

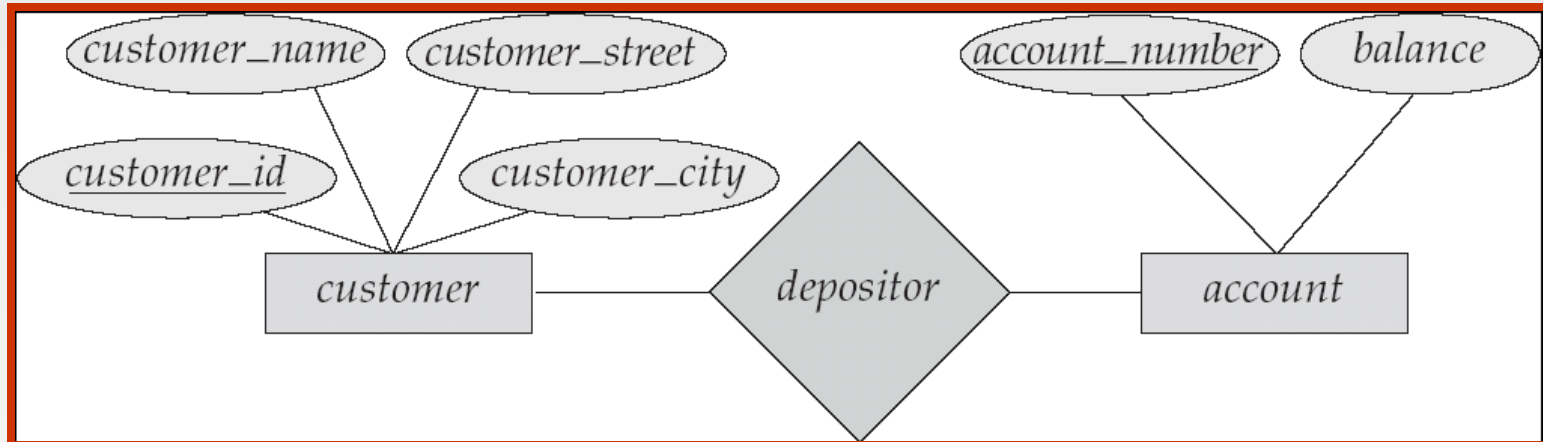
- Physikalisches Design – Entscheidung über die physikalische Implementierung der Datenbank





Das Entity-Relationship Modell

- Modellierung durch eine Menge von *Entities* und *Relationships*
 - Entity/Entität: ein "Ding" oder "Objekt" welches von anderen Objekten unterscheidbar ist
 - ▶ wird durch eine Menge von *Attributen* beschrieben
 - Relationship/Relation: eine Assoziation zwischen mehreren Entitäten
- Repräsentation mittels eines *Entity-Relationship Diagramms*:





Storage Management

- Der **Storage manager** ist ein Programmmodul, welches die Schnittstelle zwischen den "rohen" Daten, die in der Datenbank gespeichert sind, und den Anwendungsprogrammen und Anfragen, die an das System geschickt werden.
- Der Storage Manager ist verantwortlich für:
 - Interaktionen mit dem Filemanager
 - Effiziente Speicherung, Zugriff und Modifikation der Daten
- Aufgabengebiete:
 - Speicherzugriff
 - Dateiorganisation
 - Indizierung und Hashing

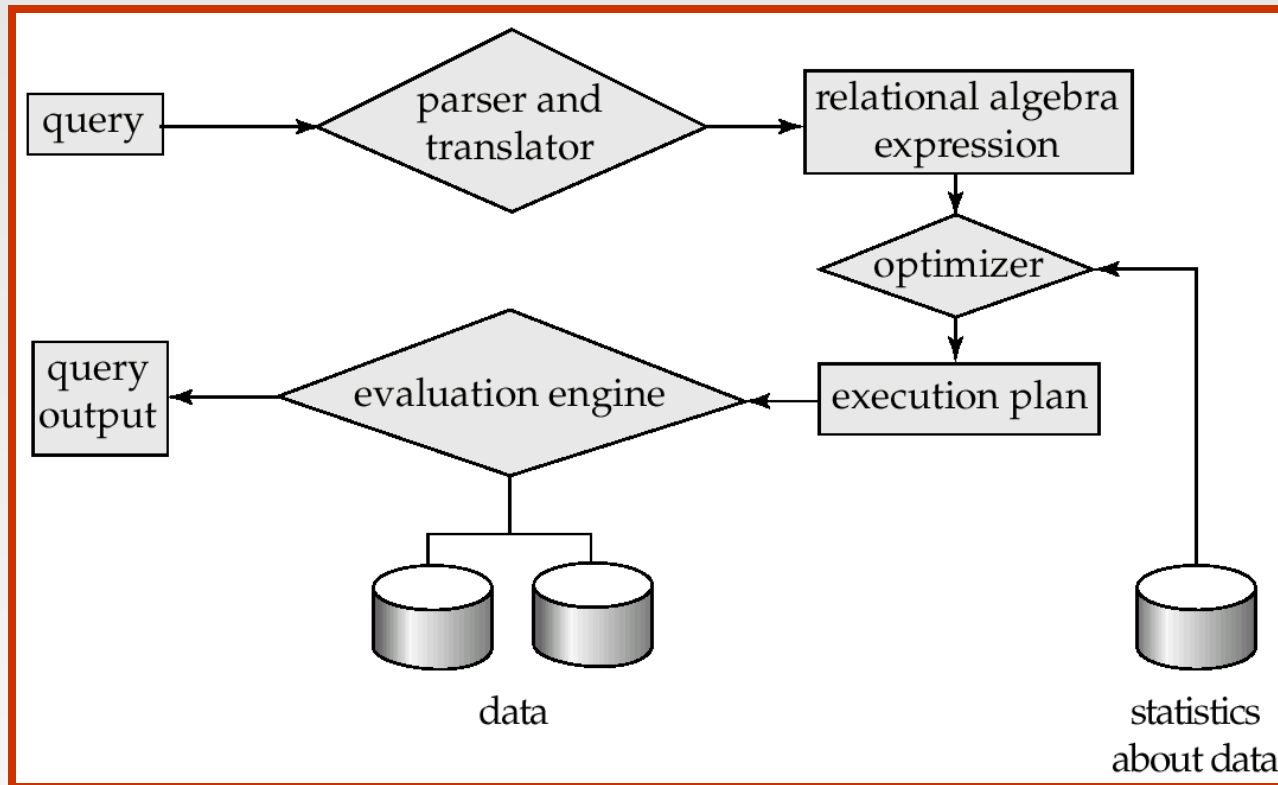




Query Processing

"Anfragenbearbeitung"

1. Parsen und Übersetzen der Anfrage
2. Optimierung der Anfrage
3. Auswertung





Query Processing (Forts.)

- Oft gibt es verschiedene Arten eine Anfrage auszuwerten
 - gleichwertige Ausdrücke
 - alternative Algorithmen für jede Operation
- Der Unterschied zwischen effizienter und nicht-effizienter Auswertung einer Anfrage ist enorm
- Oft wird zur Optimierung eine Schätzung der Kosten einer einzelnen Operation benötigt
 - hängt typischerweise stark von statistischen Informationen über die Relationen in der Datenbank ab
 - manchmal müssen Statistiken auch für Zwischenergebnisse berechnet/geschätzt werden, um den Aufwand komplexer Anfragen abschätzen zu können





Transaction Management

- Eine **Transaktion** ist eine Menge von Operationen, die eine einzelne logische Funktion in einer Datenbankanwendung ausführt
- Die **Transaction-management Komponente** stellt sicher, dass die Datenbank in einem konsistenten Zustand verbleibt, auch im Falle von Systemfehlern (Stromausfall, OS-Abstürzen) und Transaktionsfehlern.
- Der **Concurrency-control Manager** überwacht die Interaktion zwischen nebenläufigen Transaktionen, um die Konsistenz der Datenbank zu garantieren.





Datenbankarchitektur

Die Architektur eines DBMS wird stark beeinflußt vom zugrundeliegenden Rechnersystem, auf welchem die Datenbank ausgeführt wird:

- zentralisiert
- Client-server
- Parallel (multi-processor)
- verteilt





Datenbankbenutzer

Benutzer unterscheiden sich in der Art und Weise, wie sie mit dem System interagieren möchten

- **Anwendungsprogrammierer** – interagieren mit dem System durch DML-Aufrufe
- **Fortgeschrittene Benutzer** – stellen Anfragen in einer Anfragesprache (z.B. SQL)
- **Naïve Benutzer** – rufen eine der schon entwickelten Datenbank Anwendungen auf
 - Beispiele: Leute, die Datenbanken über das Web anfragen, Bankangestellte, ...





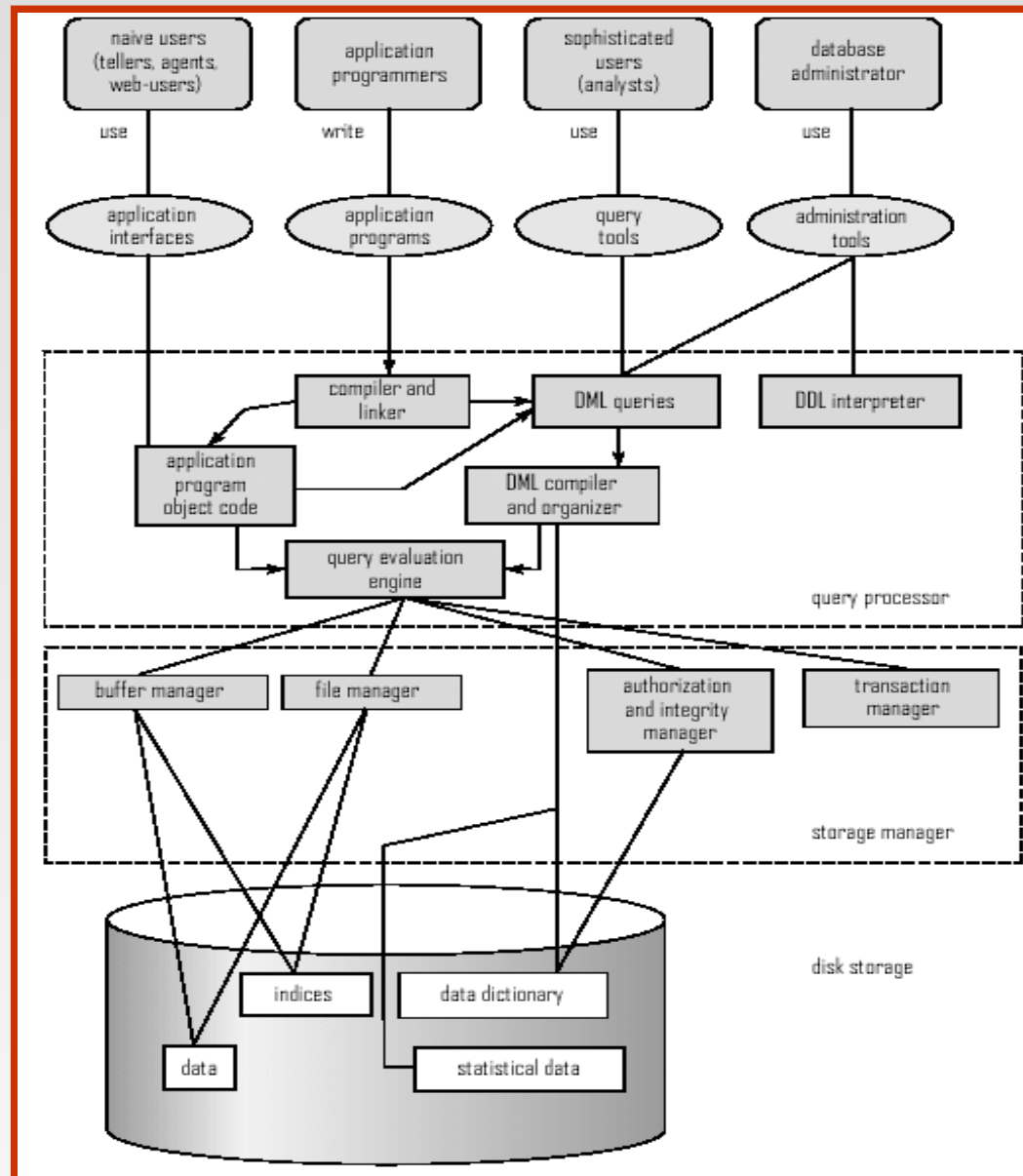
Datenbankadministrator

- Koordiniert alle Aktivitäten des DBMS; der DBMS Admin sollte einen guten Überblick über die Informationsressourcen und -bedürfnisse haben.
- Die Pflichten eines DBMS Admins schließen ein:
 - Schemadefinition
 - Festlegung der Speicherstruktur und der Zugriffsmethode
 - Änderungen an Schema oder physikalischer Organisation
 - Benutzerverwaltung der Datenbank
 - Spezifikation von Integritätsbedingungen
 - Überwachung der Performanz des Systems und ggf. Modifikation der Systemstruktur





Systemstruktur





Geschichte der DBMS

- 1950s und frühe 1960s:
 - Datenverarbeitung mittels Magnetbänder zur Speicherung
 - ▶ nur sequentieller Zugriff
 - Lochkarten als Eingabemedium
- Späte 1960s und 1970s:
 - Festplatten erlauben direkten Datenzugriff
 - Ted Codd definiert das relationale Datenmodell
 - ▶ gewinnt später dafür den ACM Turing Award
 - ▶ IBM Research startet mit dem System R Prototype
 - ▶ UC Berkeley beginnt mit dem Ingres Prototype





Geschichte (Forts.)

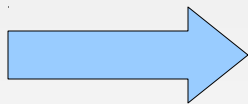
- 1980s:
 - Relationale DBMS Prototypen finden Weg in kommerzielle Systeme
 - ▶ SQL wird Industriestandard
 - Parallele und verteilte DBMS
 - OO DBMS
- 1990s:
 - Große Data Mining Anwendungen
 - Große Multiterabyte Datawarehouses
 - Erstes Erscheinen des Handels im "Netz"
- 2000s:
 - XML und XQuery Standards
 - Automatisierte Datenbankadministration Automated database administration





Grenzen der DBMS

- DBMS sind gut geeignet, Daten zu verwalten, welche
 - in homogener Form vorliegen
 - "von Natur aus" strukturiert sind
- ... und Anfragen zu beantworten welche
 - präzise das gewünschte Ergebnis formulieren
- Eine Unmenge an Daten liegt heutzutage jedoch vor in Form von
 - wissenschaftliche Publikationen
 - Ton-/Bildmaterial
 - Webseiten im Internet
- DBMS sind nicht besonders gut darin, diese Art von Daten zu verwalten



Information Retrieval Systeme





Beispiel: (Vereinfachter) Entwurf einer Datenbank

- Die Entitäten und Betriebsabläufe einer Universität sollen in einer Datenbank abgebildet werden
- Erster Schritt:
 - Skizzieren der abzubildenden Entitäten/Abläufe
- Zweiter Schritt:
 - Abbildung der Skizze in ein relationale Schemata
- Dritter Schritt:
 - Korrektur/Optimierung der relationalen Schemata
 - ▶ Minimierung von Redundanzen
 - ▶ Untersuchung auf Existenz von Primärschlüsseln
- Vierter Schritt:
 - Definition von Views auf die Daten für die einzelnen Anwender/Anwendungen

