I. EINLEITUNG

Motivation

50% weniger Aufwand bei Anwendungsentwicklung mit DB

Ermöglicht neue Anwendungen, die ohne DB zu komplex wären

Ausfaktorisieren der Verwaltung großer Datenmengen

ohne Datenbanken

- Daten in Dateien abgelegt, Zugriffsfunktionalität Teil der Anwendung
- Redundanz (in Daten und Funktionalität)
- Programme oft nicht *atomar* (= Programm wird entweder ganz oder gar nicht ausgeführt) – nur bei nicht fehlerfreien Systemen relevant
- *Transaktionen* (= Programm oder Kommandofolge) oft nicht *isoliert* (= keine inkonsistenten Zwischenzustände sichtbar) – nur bei mehreren Transaktionen, aber auch bei fehlerfreien Systemen relevant
- Nebenläufigkeit (*concurrency* – paralleler Zugriff auf dieselben Daten) schwer umsetzbar
- Anwendungsentwicklung abhängig von der physischen Repräsentation der Daten (z.B. Datenspeicherung als Tabelle: Reihenfolge Zeilen/Spalten muss bekannt sein)
- Datenschutz (= kein unbefugter Zugriff) nicht gewährleistet
- Datensicherheit (= kein Datenverlust, insb. bei Defekten) nicht gewährleistet

Relationale Datenbanken

auch RDBMS (*relational database management system*)

≅ Menge von Tabellen

Relation = Menge von Tupeln = Tabelle

RDBMS – Terminologie

Relationenschema: **Fett** geschrieben

Relation: Weitere Einträge der Tabelle

Tupel: Eine Zeile der Tabelle

Attribut: Spaltenüberschrift

Relationenname: Name der Tabelle

DBS: Datenbanksystem = DBMS + Datenbank(en)

Schlüssel: Attribut, das nicht doppelt vergeben werden darf

Fremdschlüssel: Attribut taucht in anderem Relationenschema als Schlüssel auf

Integritätsbedingungen:

1. **lokal**: Schlüssel in Relationenschema
2. **global**: Fremdschlüssel in Datenbankschema

DB-Schema: = Menge der Relationsschemata + globale Integritätsbedingungen

Sicht (*view*): Häufig vorkommende Datenabfrage, kann mit Sichtnamen als „virtuelle“ Tabelle gespeichert werden

```
create view CArtist as
  select NAME, JAHR
  from Kuenstler
  where LAND == "Kanada"
```

Verwendung wie „normale“ Relation:

```
select * from CArtist where JAHR < 2000
```

Nutzung für Datenschutz: Unterschiedliche Benutzer sehen unterschiedlichen DB-Ausschnitt

RDBMS – Anfrageoperationen

Selektion: Zeilen (Tupel) wählen ($\sigma_{\text{KID}=1012}(\text{Titel})$)

Projektion: Spalten (Attribute) wählen ($\pi_{\text{KID, NAME}}(\text{Kuenstler})$)

Beispiel komplexer Ausdruck: $\pi_{\text{NAME,ART}}(\sigma_{\text{KID}=1012}(\text{Titel}))$

Ausgangsrelation:

TITLE ID	NAME	ART	GRÖSSE	KID
102	Neil Young – Heart of Gold	mp3	2.920kb	1012
103	Rammstein – Ich liebe Neil Young	wma	4.234kb	1014
104	Neil Young – Old Man	mp3	3.161kb	1012
105	Neil Young – Four Strong Winds	wma	5.125kb	1012

Ergebnis:

NAME	ART
Neil Young – Heart of Gold	mp3
Neil Young – Old Man	mp3
Neil Young – Four Strong Winds	wma

Weitere Operationen: Verbund (*join*), Vereinigung, Differenz, Durchschnitt, Umbenennung

Operationen beliebig kombinierbar (⇝ Query-Algebra)

RDBMS – Andragenoptimierung

Algebraische Ausdrücke äquivalent, Abfrage aber unterschiedlich komplex, z.B.
$\sigma_{\text{Vorname='Klemens'}}(\sigma_{\text{Wohnort='KA'}}(SNUSER))$ vs.
$\sigma_{\text{Wohnort='KA'}}(\sigma_{\text{Vorname='Klemens'}}(SNUSER))$

RDBMS – Physische Datenunabhängigkeit

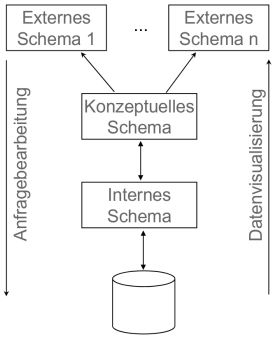
Anfragen deklarativ: Nutzer entscheidet nicht, wie Ergebnis ermittelt wird

Datenunabhängigkeit: DBMS stellt sicher:

1. stabile Anfragenfunktionalität bei physischer Darstellungsänderung
2. Anfrage funktinoiert bei unterschiedlichen Datenbanken (gleiches Schema, unterschiedliche Datenhäufigkeit)

⇝ erlaubt höhere Komplexität bei Anwendungsentwicklung

RDBMS – 3-Ebenen-Architektur



Konzeptionelles Schema: Diskursbereich? Welche Entitäten interessant (bei Studierenden Noten interessant, Hobbies usw. nicht)?

Internes Schema: physische Datenrepräsentation

Externe Schemata: Unterschiedlicher Datenausschnitt für unterschiedliche Nutzer (Datenschutz, Übersichtlichkeit, organisatorische Gründe, Verstecken von Änderungen am konzeptionellen Schema)

⇝ **Logische Datenunabhängigkeit**

Datenbankprinzipien – Coddsche Regeln

1. Integration: Einheitliche, nichtredundante Datenverwaltung
2. Operationen: Speichern, Suchen, Ändern
3. Katalog: Zugriff auf Datenbankbeschreibungen im data directory
4. Benutzersichten
5. Integritätssicherung: Korrektheit des DB-Inhalts
6. Datenschutz: Ausschluss unauthorisierter Zugriffe
7. Transaktionen: mehrere DB-Operationen als Funktionseinheit (= Atomarität)
8. Synchronisation: parallele Transaktionen koordinieren (= Isolation)
9. Datensicherung: Wiederherstellung von Daten nach Systemfehlern

Strengste bekannte Datenbankdefinition

Funktionale Anforderungen (nichtfunktional z.B.: Wie schnell/zuverlässig muss Dienst sein?)

Prüfungsfragen

1. Was ist eine Sicht?
2. Was ist die relationale Algebra? Wozu braucht man sie?
3. Geben Sie Beispiele für Algebra-Ausdrücke an, die nicht identisch, aber äquivalent sind, an.
4. Was leistet der Anfragenoptimierer einer Datenbank?
5. Erklären Sie: Drei-Ebenen-Architektur, physische/logische Datenunabhängigkeit.