Einführung in Datenbanken

Kapitel 2: Funktionen von DB-Managementsystemen

Prof. Dr. Stefan Brass

Martin-Luther-Universität Halle-Wittenberg

Wintersemester 2020/21

http://www.informatik.uni-halle.de/~brass/db20/

Lernziele

Nach diesem Kapitel sollten Sie Folgendes können:

- Die Verwendung eines DBMS zur Verwaltung persistenter Daten mit der (direkten) Verwendung von Dateien vergleichen.
  - Einige Vorteile der Lösung mit DBMS nennen.

    Und auch mögliche Nachteile.
  - Vor-/Nachteile für ein konkretes Projekt bewerten.
- „Datenunabhängigkeit“ erklären. Was ist ein Index?

  In diesem Kapitel wird hauptsächlich physische Datenunabhängigkeit behandelt.
- Vorteile deklarativer Anfrage/Programmiersprachen benennen.
- Den Begriff der Transaktion erklären, Vorteile der Transaktionsverwaltung im DBMS richtig einschätzen.

Inhalt

Persistente Daten (1)

Heute:

Morgen:

$\Rightarrow$ Die Ausgabe ist nur eine Funktion der Eingabe.

Persistente Daten (2)

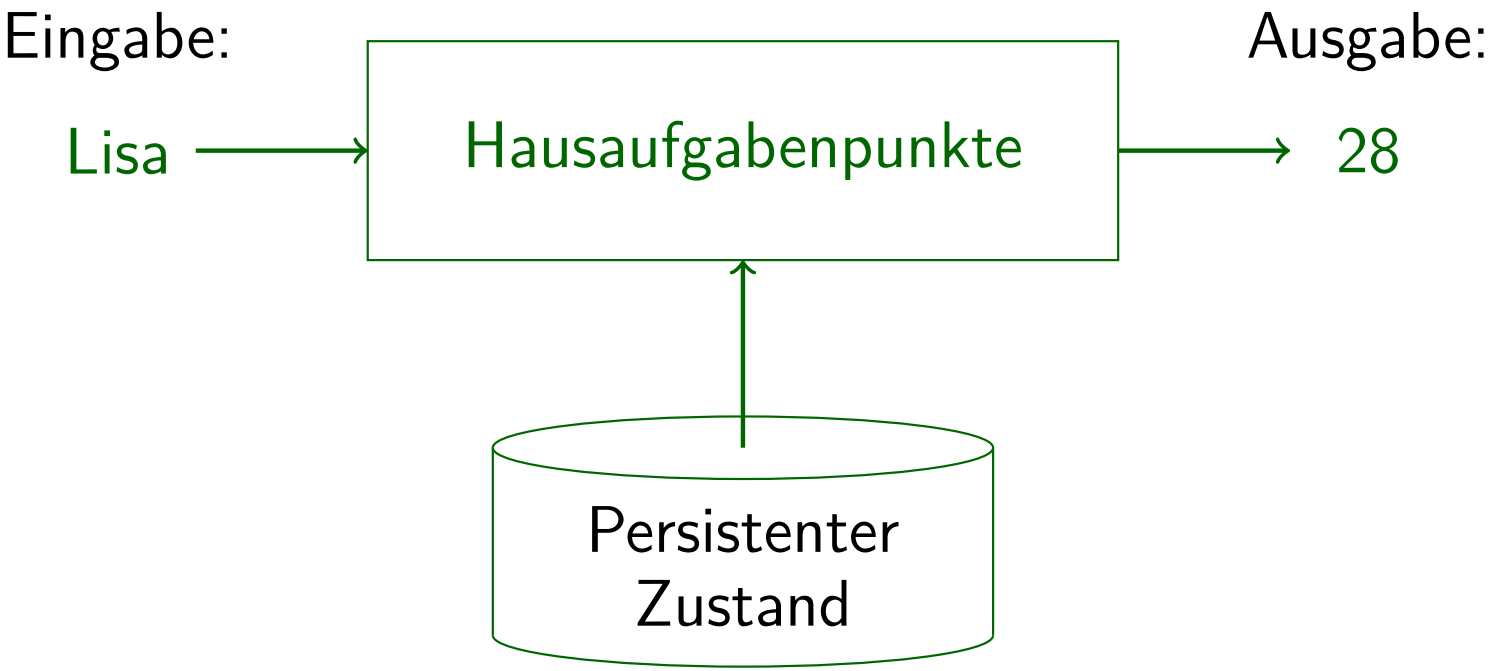
Heute:

Morgen:

$\Rightarrow$ Die Ausgabe ist eine Funktion der Eingabe
und eines persistenten Zustands.

Daten heissen persistent (dauerhaft), wenn sie länger leben als ein Prozess,
also z.B. auch nach einem Neustart des Betriebssystems noch vorhanden sind.

Persistente Daten (3)



Persistente Speicherung (1)

Klassischer Weg, Persistenz zu implementieren:

- Informationen, die in zukünftigen Programmläufen benötigt werden, werden in einer Datei gespeichert.
- Das Betriebssystem speichert die Datei auf einer Festplatte.
- Festplatten sind persistente Speicher:
  Der Inhalt geht nicht verloren, wenn der Rechner ausgeschaltet oder das Betriebssystem neu gestartet wird.

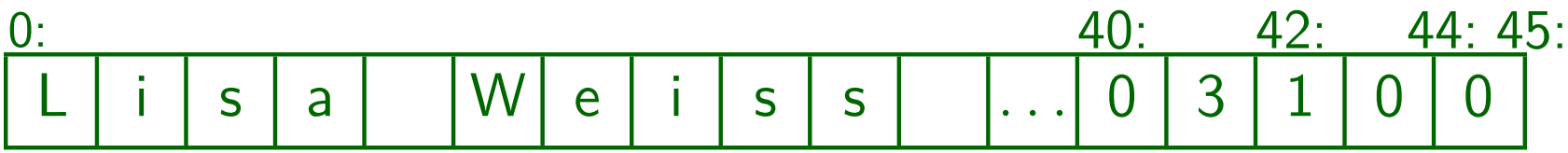
  Persistenz ist keine boolesche Eigenschaft: Festplatten gehen manchmal kaputt.

  Sorgen Sie für Backups, um die Persistenz zu verbessern.
- Dateisysteme sind Vorgänger moderner DBMS.

Persistente Speicherung (2)

Implementierung von Persistenz mit Dateien:

- Dateien sind normalerweise nur Folgen von Bytes.
- Die Struktur der Datensätze (Records) muss wie in Assembler-Sprachen festgelegt werden.



Natürlich könnte man auch irgendwelche Trennzeichen verwenden (z.B. ein Komma im CSV-Format). Es gibt viele Möglichkeiten. Das ist Teil des Problems.

- Die Information über die Dateistruktur existiert nur in den Köpfen der Programmierer.
- Das System kann nicht vor Fehlern schützen, da es die Dateistruktur nicht kennt.

Persistente Speicherung (3)

Implementierung von Persistenz mit einem DBMS:

- Die Struktur der zu speichernden Daten wird so definiert, dass sie das System versteht:

```
CREATE TABLE ABGABEN(
       NAME   VARCHAR(40)  NOT NULL,
       AUFGNR NUMERIC(2)   NOT NULL,
       PUNKTE NUMERIC(3,1) NOT NULL);
```

- Somit ist die Dateistruktur formal dokumentiert.
- Das System kann Typfehler in Anwendungsprogrammen erkennen.
- Vereinfachte Programmierung, höhere Abstraktion

DBMS als Unterprogrammbibliothek (1)

- Die meisten DBMS nutzen Dateien des Betriebssystems (z.B. Windows oder Linux), um Daten zu speichern.

  Manche nutzen aus Effizienzgründen direkten Plattenzugriff.

- Man kann ein DBMS als Unterprogrammbibliothek für Dateizugriffe verstehen.

- Verglichen mit dem Dateisystem bietet ein DBMS Operationen auf höherer Ebene an.

  Man arbeitet z.B. mit Punktzahl statt Bytes an bestimmer Position.

- Es enthält bereits viele Algorithmen, die man sonst selbst programmieren müsste.

DBMS als Unterprogrammbibliothek (2)

- Zum Beispiel enthält ein DBMS Routinen zum
  - Sortieren (Mergesort)
  - Suchen (B-Bäume)
  - Freispeicherverwaltung in Datei, Pufferverwaltung
  - Aggregationen, statistische Auswertungen
- Optimiert für große Datenmengen (die nicht in den Hauptspeicher passen).
- Es unterstützt auch mehrere Nutzer (automatische Sperren, Locks) und Sicherheitsmaßnahmen, um die Daten bei Abstürzen zu schützen (siehe unten).

Inhalt

Datenunabhängigkeit (1)

- DBMS = Software-Schicht über dem Dateisystem.
  Zugriff auf die Dateien nur über DBMS.
- Die Indirektion erlaubt es, interne Veränderungen zu verstecken.
- Idee abstrakter Datentypen:
  - Implementierung kann geändert werden,
  - aber die Schnittstelle bleibt stabil.
- Hier ist die Implementierung die Dateistruktur,
  die aus Effizienzgründen geändert werden muss.
  Das Anwendungsprogramm muss nicht angepasst werden:
  Schnittstelle zum Datenzugriff ist stabil.

Datenunabhängigkeit (2)

Typisches Beispiel:

- Anfangs nutzte ein Professor die Hausaufgaben-DB nur für seine Vorlesungen im aktuellen Semester.

  Zur Vereinfachung lässt die obige Tabelle „`ABGABEN`“ nur eine Vorlesung zu. Aber es könnte eine zusätzliche Spalte geben, um verschiedene Vorlesungen zu unterscheiden.

- Da die DB klein war und es wenige Zugriffe gab, genügte es, die Daten als „heap file“ zu speichern.

  D.h. die Zeilen der Tabelle (Records) werden ohne bestimmte Reihenfolge gespeichert. Um Anfragen auszuwerten, muss das DBMS die ganze Tabelle durchsuchen, d.h. jede Zeile der Tabelle lesen und überprüfen, ob sie die Anfragebedingung erfüllt („full table scan“). Für kleine Tabellen ist das kein Problem.

Datenunabhängigkeit (3)

- Später nutzte die ganze Uni die Datenbank und Daten vergangener Vorlesungen mussten aufbewahrt werden.
- Die DB wurde viel größer und hatte mehr Zugriffe.

  Das „Lastprofil“ hat sich geändert.

- Für schnelleren Zugriff wird ein Index (typischerweise ein B-Baum) benötigt.

  Ein Index eines Buches ist eine sortierte Liste mit Stichwörtern und den zugehörigen Seitenzahlen, wo die Wörter vorkommen. Ein B-Baum enthält im Wesentlichen eine sortierte Liste von Spaltenwerten zusammen mit Verweisen auf die Tabellenzeilen, die diese Werte enthalten.
  Indexe werden in der Vorlesung „Datenbank-Programmierung“ behandelt, und ganz ausführlich in „Datenbanken II B: DBMS-Implementierung“.

Datenunabhängigkeit (4)

Fußnote: Indexe vs. Indices

- Als Plural von „Index“ ist in der DB-Literatur (wenn es um B-Bäume etc. geht) „Indexe“ üblich, obwohl „Indices“ auch vorkommt (selten).

  Z.B. verwenden Kemper/Eickler, Vossen, Härder/Rahm und die deutsche Übersetzung von Elmasri/Navathe den Plural „Indexe“. In meinem deutschen Wörterbuch steht als Plural nur „Indices“ bzw. „Indizes“, aber das deckt sich nicht mit meinem persönlichen Sprachempfinden. In der Wikipedia stehen „Indexe“, „Indices“, „Indizes“ als (offenbar gleichberechtigte) Alternativen (speziell für Datenbanken). Als Genitiv sind „des Indexes“ und „des Index“ beide möglich.

- Dagegen besteht Einigkeit, dass die kleinen tiefgestellten Buchstaben (wie i in x_i) „Indices“ sind.

Datenunabhängigkeit (5)

Ohne DBMS (oder mit Prä-Relationalem DBMS):

- Die Verwendung eines Indexes für den Datenzugriff muss explizit in der Anfrage verlangt werden.
- Somit müssen Anwendungsprogramme geändert werden, wenn sich die Dateistruktur ändert.
- Vergisst man, ein selten verwendetes Anwendungsprogramm zu ändern, wird die DB inkonsistent.

  Das kann jedoch nur passieren, wenn man direkt mit Betriebssystem-Dateien arbeitet. Schon ein DBMS für das Netzwerk-Datenmodell (z.B.) machte Index-Updates automatisch. Die Anfrage-Programme mussten sich jedoch explizit auf den Index beziehen.

Datenunabhängigkeit(6)

Mit Relationalem DBMS:

- Das System versteckt die Existenz von Indexen an der Schnittstelle.
- Anfragen und Updates müssen (und können) sich nicht auf Indexe beziehen.
- Folgendes geschieht automatisch durch das DBMS:
  - Aktualisierung des Indexes bei einem Update,
  - Nutzung des Indexes zur Anfrageauswertung, wenn dies von Vorteil (d.h. schneller) ist.

Datenunabhängigkeit (7)

Konzeptuelles Schema („Schnittstelle“):

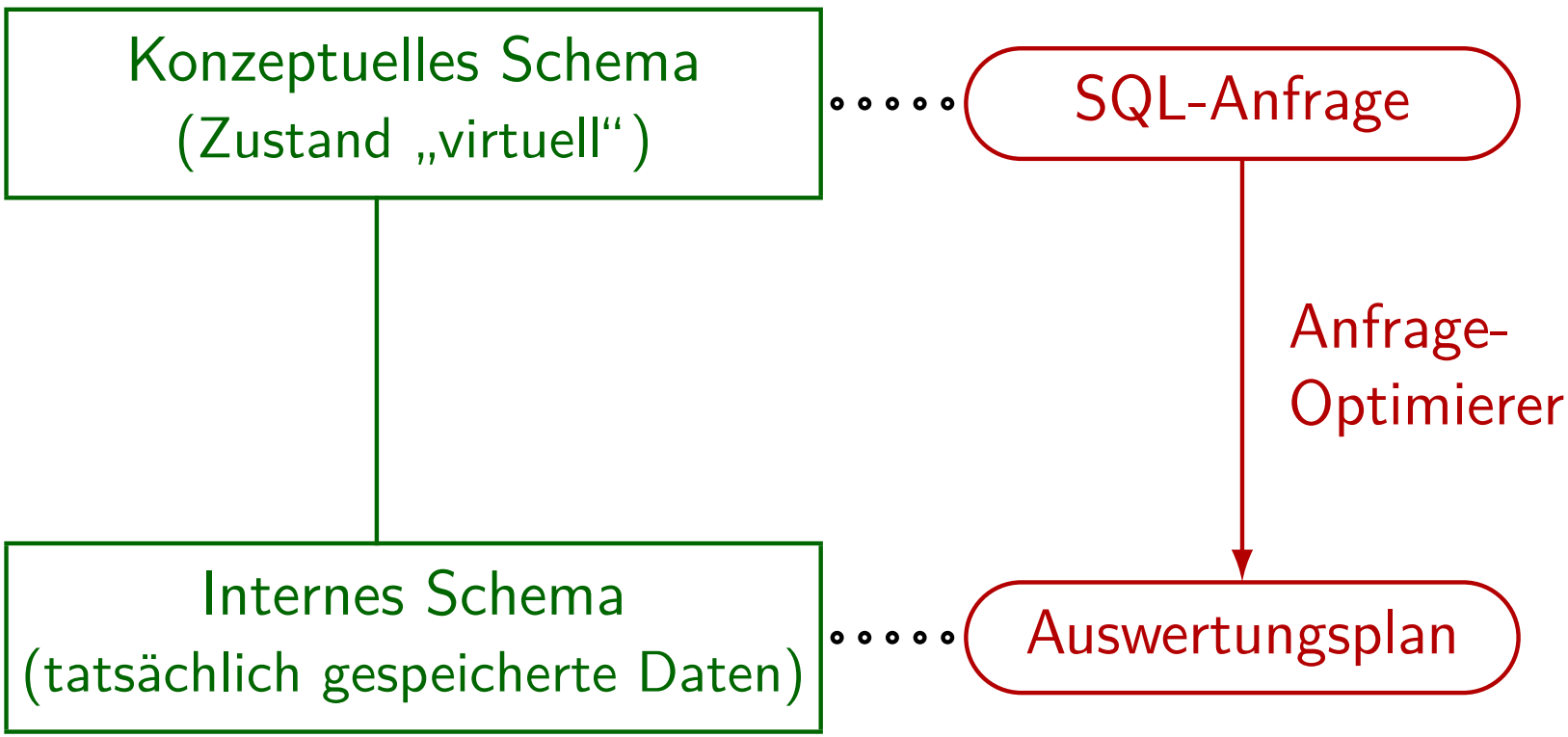
- Nur der logische Informationsgehalt der DB
- Vereinfachte Sicht: Speicherdetails sind versteckt.

Internes/Physisches Schema („Implementierung“):

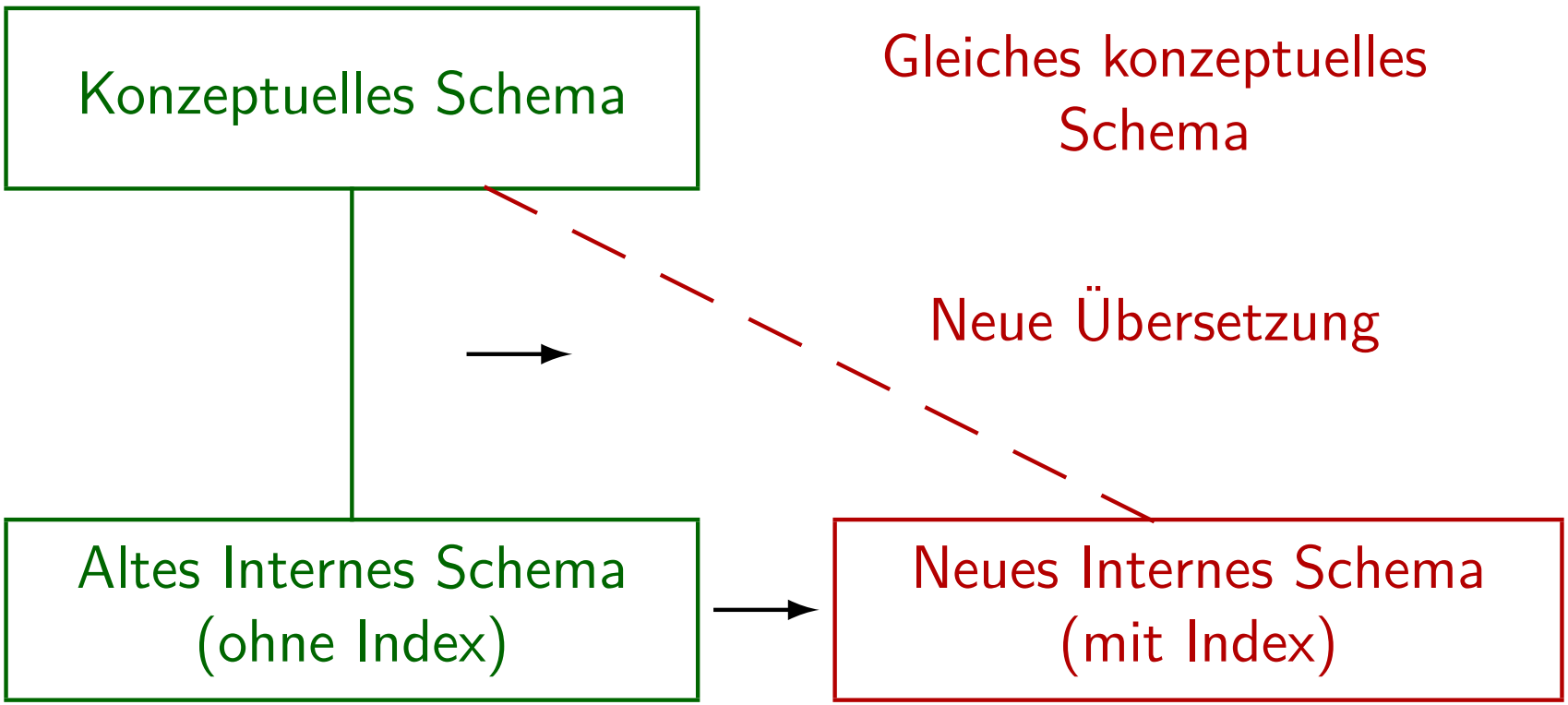
- Indexe, Aufteilung der Tabellen auf Festplatten
- Parameter für Speicher-Management,
  wenn Tabellen wachsen oder schrumpfen
- Physische Plazierung neuer Zeilen in einer Tabelle.

  Z.B. Zeilen mit gleichem Spaltenwert in den gleichen Plattenblock.

Datenunabhängigkeit (8)



Datenunabhängigkeit (9)



Datenunabhängigkeit (10)

Zur Bezeichnung „Datenunabhängigkeit“:

- Ziel ist die Entkopplung von Programmen und Daten.
- Daten sind eine unabhängige Ressource.

  Früher hatten sie nur im Zusammenhang mit den Anwendungsprogrammen eine Bedeutung, mit denen sie ursprünglich erfasst wurden.
- Physische Datenunabhängigkeit:
  - Programme sollten nicht von Speichermethoden (Datenstrukturen) abhängen.
  - Umgekehrt werden die Dateistrukturen nicht durch die Programme festgelegt.

  $\Rightarrow$ Schützt Investitionen in Programme und Daten.

Logische Datenunabhängigkeit

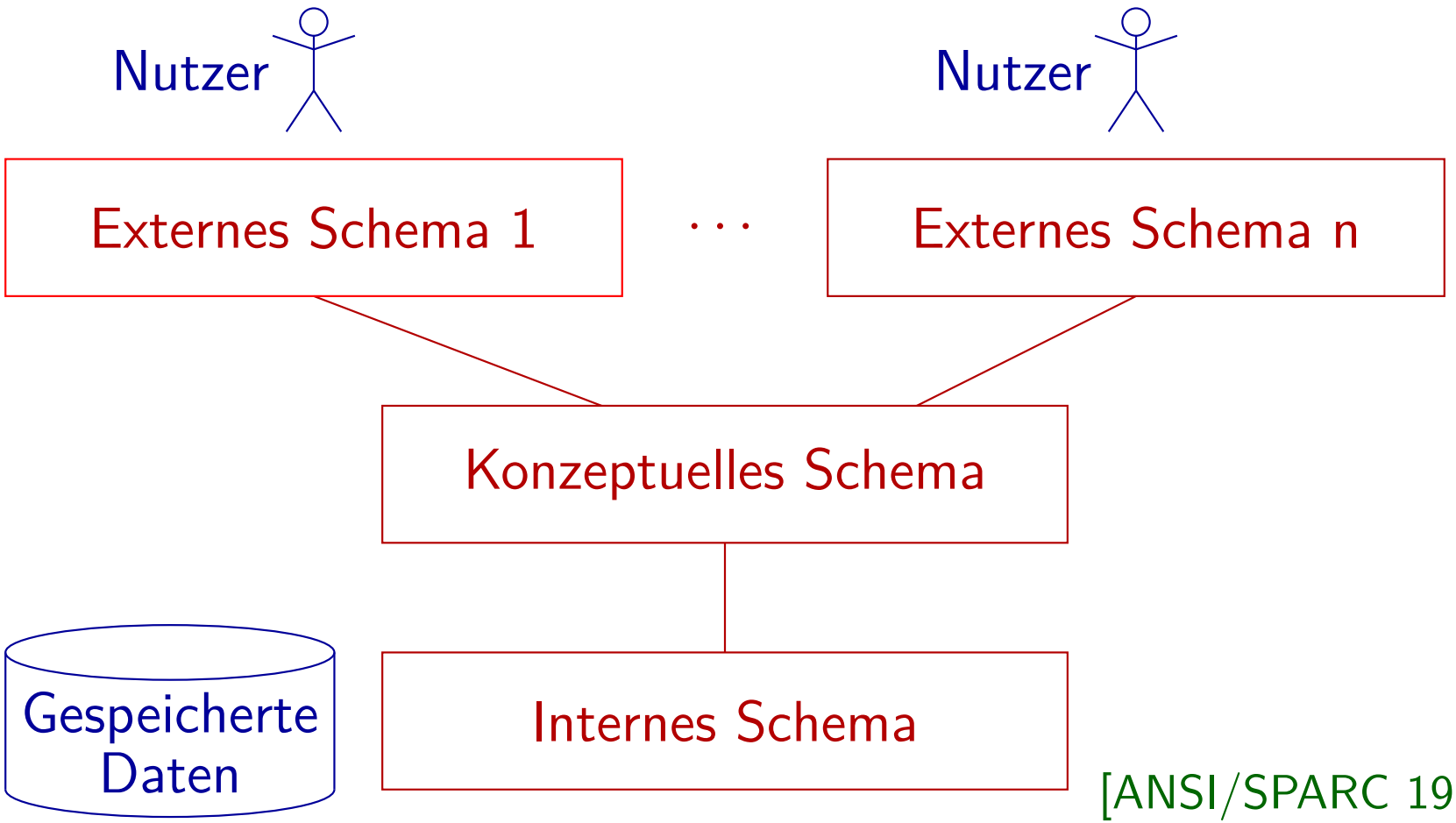
- Ein DBMS kann auch jedem Nutzer bzw. jeder Nutzergruppe oder jedem Anwendungsprogramm eine individuelle Sicht auf die gespeicherten Daten anbieten.

  Z.B. sieht jeder/jede nur die Teilmenge der Daten, die er/sie wirklich braucht. Das können auch abgeleitete Daten sein, z.B. Anzahlen oder Summen (weniger Details). Jeder/Jede hat ein eigenes Schema, dass sich aus virtuellen Tabellen zusammensetzt, die durch Anfragen aus den tatsächlich gespeicherten Daten berechnet werden.

- Dadurch kann auch der logische Informationsgehalt der Datenbank geändert (im wesentlichen nur erweitert) werden, ohne individuelle Schemata zu ändern, die die Erweiterung nicht benötigen.

  Z.B. zusätzliche Spalten in Tabellen. Logische Datenunabhängigkeit ist nur interessant, wenn es mehrere Nutzer (bzw. Anwendungsprogramme) mit unterschiedlichem, aber überlappenden Informationsbedürfnis gibt.

Drei-Schema Architektur



[ANSI/SPARC 1978]

Inhalt

Deklarative Sprachen (1)

- Physische Datenunabhängigkeit verlangt, dass sich die Anfragesprache nicht auf Indexe beziehen kann.
- Deklarative Anfragesprachen gehen noch weiter:
  - Anfragen sollen nur beschreiben,
    was (welche Information) gesucht wird,
  - aber sollen keine spezielle Methode vorschreiben,
    wie diese Information berechnet werden soll.
- Algorithmus = Logik + Kontrolle (Kowalski)

  Imperative/Prozedurale Sprachen: Kontrolle explizit, Logik implizit.

  Deklarative/Deskriptive Sprachen: Logik explizit, Kontrolle implizit.

Deklarative Sprachen (2)

- SQL ist eine deklarative Anfragesprache.
  Man gibt nur Bedingungen für die gesuchten Daten an:

```
SELECT X.PUNKTE
FROM   ABGABEN X
WHERE  X.NAME = 'Lisa Weiss'
AND    X.AUFGNR = 3
```

- Häufig leichtere Formulierungen: Der Nutzer muss nicht über eine effiziente Auswertung nachdenken.
- Oft viel kürzer als imperative Programmierung.
  Daher ist z.B. die Programmentwicklung billiger.

Deklarative Sprachen (3)

- Deklarative Anfragesprachen
  - erlauben leistungsstarke Optimierer

    weil sie keine Auswertungsmethode vorschreiben.
  - benötigen leistungsstarke Optimierer

    weil ein naiver Auswertungsalgorithmus ineffizient wäre.
- Größere Unabhängigkeit von aktueller Hardware/Software-Technologie:
  - Einfache Parallelisierung
  - Heutige Anfragen können zukünftige Algorithmen verwenden (bei neuer DBMS-Version).

Inhalt

Weitere DBMS-Funktionen (1)

Transaktionen:

- Folge von DB-Befehlen, die als atomare Einheit ausgeführt werden („ganz oder gar nicht“)

  Wenn das System während einer Transaktion abstürzen sollte, werden alle Änderungen rückgängig gemacht („roll back“), sobald das DBMS neu startet. Stürzt das System nach einer Transaktion ab (nach bestätigtem „commit“), sind alle Änderungen garantiert im DB-Zustand gespeichert.

- Typisches Beispiel: Überweisung (Abbuchung+Zubuchung)
- Unterstützung von Backup und Recovery

  Daten abgeschlossener Transaktionen sollten Plattenfehler überleben.

- Unterstützung von gleichzeitigen Nutzern

  Nutzer/Programmierer sollen nicht über parallele Zugriffe anderer Nutzer nachdenken müssen (z.B. durch automatische Sperren).

Weitere DBMS-Funktionen (2)

Sicherheit:

- Zugriffsrechte: Wer darf was womit machen?

  Es ist sogar möglich, Zugriffe nur für einen Teil der Tabelle oder nur für aggregierte Daten zuzulassen.

- Auditing: Das DBMS speichert ab, wer was mit welchen Daten gemacht hat.

Integrität:

- Das DBMS prüft, ob die Daten plausibel bzw. konsistent sind. Es lehnt Updates ab, die definierte Geschäftsregeln verletzen würden.

Weitere DBMS-Funktionen (3)

Data Dictionary / Systemkatalog (Meta-Daten):

- Informationen wie Schema, Nutzer und Zugriffsrechte sind in Systemtabellen verfügbar, z.B.:

SYS_TABLES	
TABLE_NAME	OWNER
ABGABEN	BRASS
SYS_TABLES	SYS
SYS_COLUMNS	SYS

SYS_COLUMNS		
TABLE_NAME	SEQ	COL_NAME
ABGABEN	1	NAME
ABGABEN	2	AUFGNR
ABGABEN	3	PUNKTE
SYS_TABLES	1	TABLE_NAME
SYS_TABLES	2	OWNER
SYS_COLUMNS	1	TABLE_NAME
SYS_COLUMNS	2	SEQ
SYS_COLUMNS	3	COL_NAME

Dies ist nur ein Beispiel, die genauen Tabellen sind systemabhängig.

Inhalt

Zusammenfassung (1)

Funktionen von Datenbanksystemen:

- Persistenz
- Integration, keine Redundanz/Duplikatspeicherung
- Datenunabhängigkeit
- Weniger Programmieraufwand: Viele Algorithmen enthalten, besonders für Externspeicher (Platten).
- Ad-hoc-Anfragen

  D.h. wenn jemand eine neue Anfrage im Kopf hat, kann er sie sofort stellen. Früher benötigte man dafür ein neues Programm. Natürlich stammen auch heute viele der ausgeführten Anfragen aus Anwendungsprogrammen.
  Aber man ist eben nicht darauf eingeschränkt. Durch die Deklarativität sind die Anfragen heute auch viel kürzer als die entsprechenden Programme früher.

Zusammenfassung (2)

Funktionen von Datenbanksystemen, Fortsetzung:

- Hohe Datensicherheit (Backup & Recovery)
- Zusammenfassung von Updates zu Transaktionen

  Transaktionen werden ganz oder gar nicht ausgeführt (sind atomar).
- Mehrbenutzerbetrieb: Synchronisation von Zugriffen
- Integritätsüberwachung
- Sichten für verschiedene Nutzer (Nutzergruppen)
- Datenzugriffskontrolle
- System-Katalog (Data Dictionary)

Zusammenfassung (3)

- Hauptziel eines DBMS: dem Nutzer eine vereinfachte Sicht auf den persistenten Speicher geben.
- Der Nutzer muss nicht nachdenken über:
  - Physische Speicherdetails
  - Unterschiedlichen Informationsbedarf von verschiedenen Nutzern
  - Effiziente Anfrageformulierung
  - Möglichkeit von Systemabsturz/Plattenfehlern
  - Möglichkeit von störenden gleichzeitigen Zugriffen anderer Nutzer

Übungen (1)

Aufgabe:

- Angenommen, Sie sollen ein System zur Evaluation der Lehre an dieser Universität entwickeln:
  Studierende stimmen über die Vorlesungs-Qualität ab.

  Es gibt ein Formular im Internet, in das Studierende ihre Daten eingeben können. Diese werden auf dem Web-Server abgespeichert.
  Später werden die gesammelten Daten ausgewertet,
  z.B. Durchschnittswerte berechnet.

- Vorschlag: Die Daten werden in einer Datei gespeichert (z.B. im CSV-Format).

- Welche Argumente gibt es, stattdessen ein DBMS zu verwenden?

Übungen (2)

- Stellen Sie sich vor, die Hausaufgabenpunkte sind in einer Textdatei gespeichert mit dem Format

  Vorname,Nachname,Aufgabennummer,Punkte

  (d.h. ein Tupel der Tabelle `ABGABEN` pro Zeile).

- Welchen Aufwand schätzen Sie für die Entwicklung eines Java-Programms, das die Gesamtpunktzahl je Student ausgibt (alphabetisch geordnet).

  Anzahl Zeilen: _____ (ohne Kommentare)

  Arbeitszeit: _____

- In SQL braucht man 4 Zeilen und 2 Minuten Zeit.