Datenbank-Programmierung

Kapitel 11: DB-Programmierung: Einführung und Übersicht

Prof. Dr. Stefan Brass

Martin-Luther-Universität Halle-Wittenberg

Sommersemester 2024

http://www.informatik.uni-halle.de/~brass/dbp24/

Lernziele

Nach diesem Kapitel sollten Sie Folgendes können:

- Die Notwendigkeit von Anwendungsprogrammen erklären.
- Verschiedene Technologien zur Anwendungsprogrammierung aufzählen.

Beispiel-Datenbank

STUDENTEN

SID	VORNAME	NACHNAME	EMAIL
101	Lisa	Weiss	...
102	Michael	Grau	NULL
103	Daniel	Sommer	...
104	Iris	Winter	...

BEWERTUNGEN

SID	ATYP	ANR	PUNKTE
101	H	1	10
101	H	2	8
101	Z	1	12
102	H	1	9
102	H	2	9
102	Z	1	10
103	H	1	5
103	Z	1	7

AUFGABEN

ATYP	ANR	THEMA	MAXPT
H	1	ER	10
H	2	SQL	10
Z	1	SQL	14

Inhalt

Einleitung (1)

SQL ist eine Datenbanksprache, keine Programmiersprache:

- Mächtige Anfragen und Updates können als kurze SQL-Befehle geschrieben werden.

  Etwas Ähnliches in Java zu schreiben, würde viel länger dauern (längerer Code).

- Man kann aber z.B. nicht in SQL schreiben:
  - Benutzerschnittstellen,
  - Schnittstellen zu anderer Software,
  - Parser für komplex strukturierte Eingabedateien,
  - komplexe Berechnungen.

- SQL ist nicht berechnungs-universell (Turing-vollständig).

  Zumindest ohne Rekursion. Nicht jede berechenbare Funktion (die man z.B. in Java schreiben könnte) kann auch in SQL geschrieben werden.

  Sonst könnte die Terminierung der Anfrageauswertung nicht garantiert werden.

Einleitung (2)

- SQL wird direkt für ad-hoc Anfragen und für einmalige Updates der Daten verwendet.

  Wenn man z.B. den Mehrwertsteuersatz in der Datenbank abgelegt hat, braucht man eventuell kein Anwendungsprogramm, um ihn zu ändern (wenn man SQL-kundiges Personal im Hause hat). Änderung ist sehr selten.

- Wiederholte Aufgaben müssen von Anwendungsprogrammen unterstützt werden:
  - Nicht alle Datenbank-Nutzer kennen SQL.

    Z.B. die meisten Verkäufer im Supermarkt an der Kasse.
  - Selbst wenn der Nutzer SQL kennt, kann es bequemer und schneller sein, ein spezialisiertes Anwendungsprogramm für die Aufgabe zu nutzen (Scannerkasse statt SQL-Befehl).

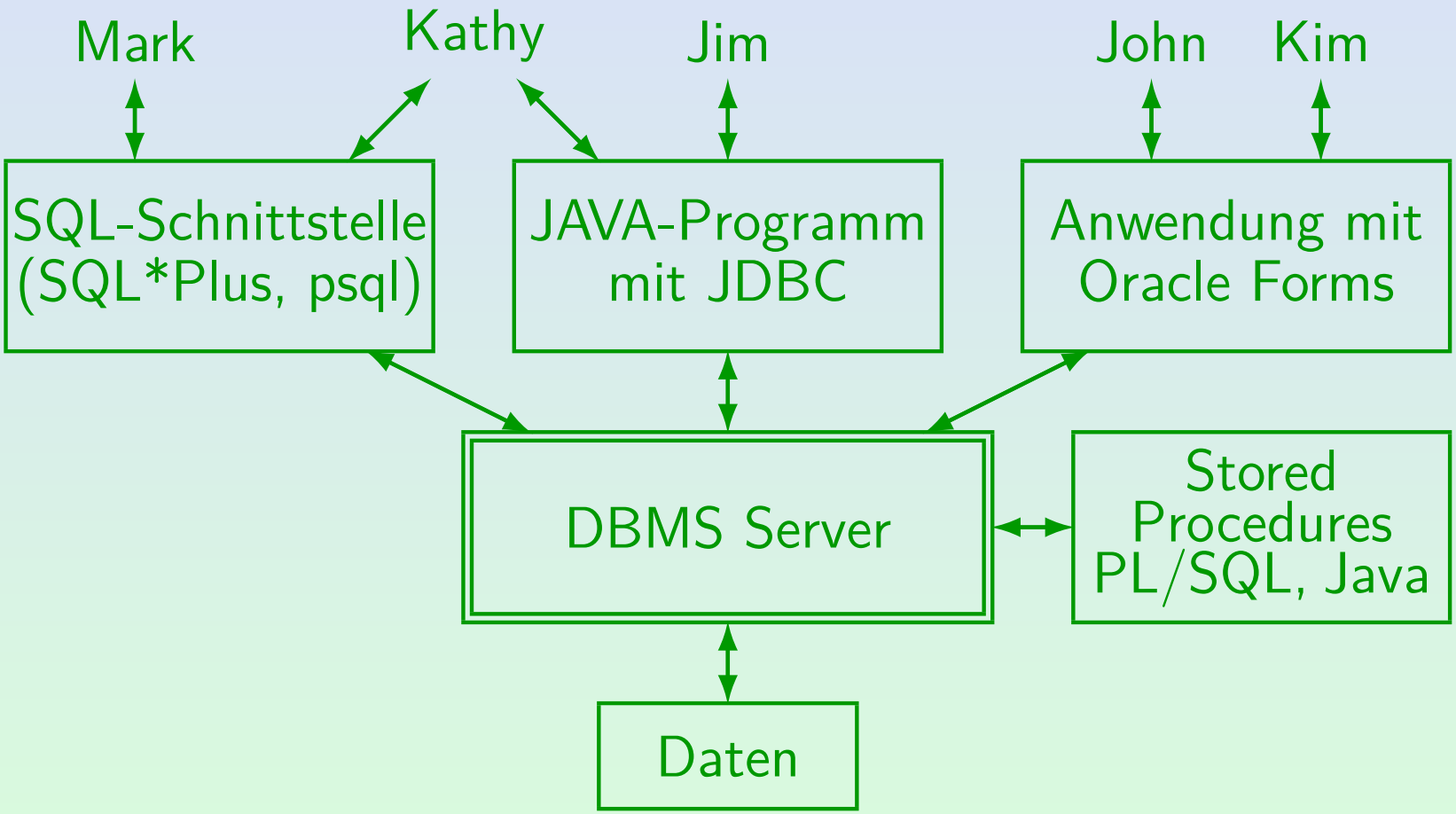
    Das Anwendungsprogramm kann auch das Anfrageergebnis übersichtlicher anzeigen, oder zusätzliche Prüfungen auf den Eingabedaten vornehmen.

Einleitung (3)

- Intern nutzen die Anwendungsprogramme aber SQL zur Kommunikation mit der Datenbank:
  Das DBMS spricht nur SQL.
- Zur Entwicklung von Anwendungsprogrammen muss man also im Normalfall SQL kennen.
- Es gibt allerdings Bibliotheken und Werkzeuge, die SQL auch für den Programmierer verstecken.

  Ich persönlich sehe das sehr kritisch, da man dann nicht die volle Mächtigkeit von SQL ausnutzen kann.
- Spätestens bei Problemen ist es dann aber sehr nützlich, wenn man SQL beherrscht.

Einleitung (4)



Inhalt

Übersicht (1)

Einige Sprachen/Werkzeuge zur Anwendungsprogrammierung:

- Skripte der interaktiven SQL-Schnittstelle (SQL*Plus, psql)
- Java, C etc. mit Prozeduraufrufen (JDBC, ODBC, SQL/CLI)
- Java, C etc. mit Embedded SQL (SQLJ bzw. SQL/OLB)

  Erfordert Precompiler, der SQL-Befehle im Java durch Prozeduraufrufe ersetzt.
- Object-Relational Mapper (JDO, JPA, Hibernate)

  Gewissermaßen virtuelle objektorientierte DB, implementiert mit RDBMS.
- Formular-basierte Anwendungen (Oracle Forms, OpenXava)

  Einfache Oracle Forms Anwendungen kann man als Editor für eine bestimmte Relation verstehen. Es gibt dafür eine graphische Entwicklungsumgebung ohne „echte“ Programmierung.
- Report-Generatoren (Erzeugung von Berichten, Diagrammen)

Übersicht (2)

Techniken zur Entwicklung von Web-Schnittstellen z.B.:

- Beliebige Programme mit CGI-Schnittstelle
- PL/SQL-Prozeduren im DB-Server

  Oracle hat dafür eine Bibliothek zur Ausgabe von HTML.
- Java Servlets, Java Server Pages

  Z.B. ausgeführt von Apache Tomcat oder Eclipse GlassFish (Application Server).
- PHP bzw. HTML mit eingebetteten PHP-Code
- JavaScript mit Datenbank-Zugriff (oft über AJAX)

  AJAX: Aynchronous JavaScript and XML (Nachladen von Daten für Webseiten im Hintergrund in XML bzw. JSON über ein XMLHttpRequest Objekt).
  Oft geschieht das über die `jQuery`-Bibliothek für JavaScript. Es gibt eine Vielzahl von Frameworks/Bibliotheken zur Erstellung von Web-Anwendungen im Browser, z.B. AngularJS, Angular 2 (in TypeScript), Ember.js, Vue.js, React.

Probleme und Lösungen (1)

- Man muss oft mit mehr als einer Sprache arbeiten, um eine Anwendung zu entwickeln (z.B. Java und SQL).

  Oft auch mit HTML, CSS, JavaScript, PL/SQL, ...

- Dies führt zu verschiedenen Problemen:
  - Unterschiedliche Typsysteme in DB und Programmiersprache.
  - „Impedance mismatch problem“:
    SQL ist deklarativ und mengenorientiert,
    Programmiersprachen sind meist imperativ, tupelorientiert.
  - Nur lokale Optimierung einzelner SQL-Kommandos.
  - Auswertungspläne für SQL-Anweisungen müssen zwischen mehreren Ausführungen des gleichen Programms aufbewahrt werden, aber Programme sind extern zu der DB.

Probleme und Lösungen (2)

- Diese Probleme können durch integrierte Systeme aus Programmiersprache und Datenbank vermieden werden, z.B.:
  - Persistente Programmiersprachen (z.B. PJama).

    Ganz klassisch: Pascal/R: Pascal mit einem Typkonstruktor „Relation“.
  - 4GLs: Sprachen der vierten Generation.

    4GL: GUI + DB + Regeln/Trigger(?).

    Visuelle Entwicklungsumgebung, nur wenig „klassischer Code“.
  - Prozeduren, die im DB-Server gespeichert sind und dort ausgeführt werden.

    Sprachen wie PL/SQL von Oracle, PL/pgSQL für PostgreSQL, oder Transact-SQL von Sybase/Microsoft. Oracle erlaubt auch Java.
  - Objektorientierte Datenbanken.
  - Deduktive Datenbanken.

Lassen Sie die Datenbank ihren Job machen!

- Manche Anwendungsprogramme nutzen eine relationale Datenbank nur, um Objekte persistent zu machen, aber führen alle Berechnungen in der Programmiersprache durch.

  D.h. Anfragen selektieren nur einzelne Zeilen, und enthalten keine Verbunde (Joins) oder Aggregationen. Manchmal wird der „Nested Loop Join“ (eine einfache Join-Implementierung) nachprogrammiert.

- Mächtige SQL-Anweisungen könnten
  - das Programm einfacher und kürzer machen,
  - die Leistung signifikant verbessern.

    Es gibt einen Overhead für jede einzelne SQL-Anweisung: Sie muss über das Netz an den Server geschickt werden, und die Ergebnisse entsprechend zurück. Wenn man also weniger (und dafür mächtigere) SQL-Anweisungen verwendet, ist das besser. Auch der Anfrageoptimierer hat nur wenig Optionen bei ganz simplen SQL-Anfragen.

Literatur/Quellen

- Elmasri/Navathe: Fundamentals of Database Systems, 2nd Edition. Section 10.5, „Programming Oracle Applications“
- Oracle8 Application Developer's Guide, Oracle Corporation, 1997, Part No. A58241-01.
- Wikipedia: Object-Relational Mapping [https://en.wikipedia.org/wiki/Object-relational_mapping]
- W3Schools: AJAX Database Example [https://www.w3schools.com/js/js_ajax_database.asp]
- Vue.js - The Progressive JavaScript Framework. [https://vuejs.org/]
- Wikipedia: Open Database Connectivity. [https://en.wikipedia.org/wiki/Open_Database_Connectivity]
- Oracle: Java SE Technologies — Database. [https://www.oracle.com/technetwork/java/javase/jdbc/index.html]