Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Datenbanksysteme I

Datenbankprogrammierung

15.6.2009

Felix Naumann

SQL mit einer Programmiersprache verbinden

- Embedded SQL
  - Kombiniert SQL mit 7 Programmiersprachen
    - ADA, C, Cobol, Fortran, M, Pascal, PL/I
  - Einbettung von SQL durch Preprocessing
- Stored procedures / PSM
  - Speicherung von Prozeduren als DBMS Objekte
  - Aufruf aus SQL Ausdrücken
- (Call-level-interface (CLI)
  - Verbindet C mit DBMS
  - Spezielle Funktionsbibliothek
  - Spart das Preprocessing)
- Java Database Connectivity (JDBC)
  - Wie CLI aber für Java
  - Relevant für die Übung

Überblick

- Embedded SQL
- Stored procedures / PSM
- Call-level-interface (CLI) und
  Java Database Connectivity (JDBC)

SQL vs. Programmiersprachen

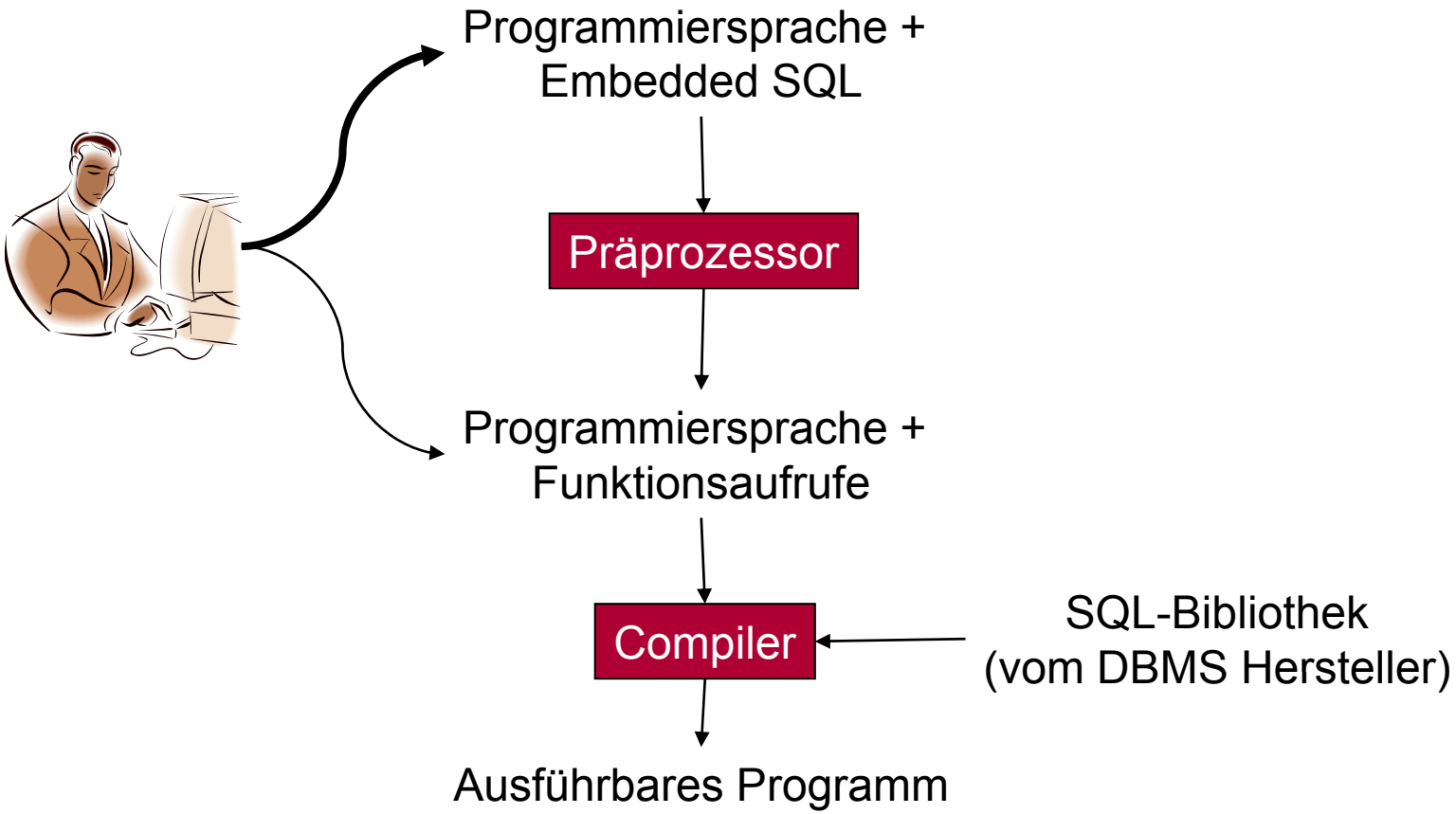
Bisher

- Generische SQL Schnittstelle
- Kommandozeile
- Von allen DBMS angeboten
- Selten genutzt
  - Datenbankadmins

Jetzt

- SQL Ausdrücke in größeren Softwarekomponenten
  - Anwendungen
  - DB Tools
- Verwendet aus einer anderen Programmiersprache heraus

Embedded SQL - Ablauf



Impedance Mismatch

Die Datenmodelle von DBMS und Programmiersprachen unterscheiden sich sehr.

- Programmiersprachen
  - Integer, String, Real, …
  - Pointer
  - Arrays
  - Insbesondere: Keine Mengen
- Relationales DBMS
  - Relationen und Attribute
  - Keine Pointer
  - Keine Schleifen
  - Keine Verzweigungen
  - Keine komplexen Strukturen

Datentransfer zwischen beiden Modellen ist also schwierig.

Impedance Mismatch

- Alternative 1: Nur Programmiersprache verwenden
  - Aber: SQL sehr nützliche und einfache (very high-level) Sprache
  - Aber: Programmier sollen nichts über Speicherstruktur wissen
    - Physische Datenunabhängigkeit!
  - Aber: DBMS sehr effizient
- Alternative 2: Nur SQL verwenden
    - Aber: In Basis-SQL nicht alles ausdrückbar
      - Z.B. n!
    - Aber: Ausgabe sind immer nur Relationen
      - Und z.B. nicht Graphiken
- Alternative 3: Einbettung von SQL in eine Programmiersprache
  - Programmiersprache: „*Host language*“ („Wirtssprache“)
  - SQL: Embedded SQL (eingebettetes SQL)

Beispiel

- Tupel einfügen
  - ```
    EXEC SQL BEGIN DECLARE SECTION;
            char studioName[50], studioAdr[256];
            char SQLSTATE[6];
    EXEC SQL END DECLARE SECTION;

    /* … Einlesen von Werten in die Variablen … */

    EXEC SQL INSERT INTO Studio(Name, Adresse)
                    VALUES (:studioName, :studioAdr);
    ```
- Klappt für jeden SQL Ausdruck, der kein Ergebnis produziert.
  - Wegen *Impedance Mismatch*
  - **INSERT, DELETE, UPDATE, CREATE, DROP**, …
- Für Anfrage mit Anfrageergebnis
  - Nur ein Tupel: Direkte Bindung an gemeinsame Variablen
  - Mehrere Tupel: Cursor
    ```

Cursor

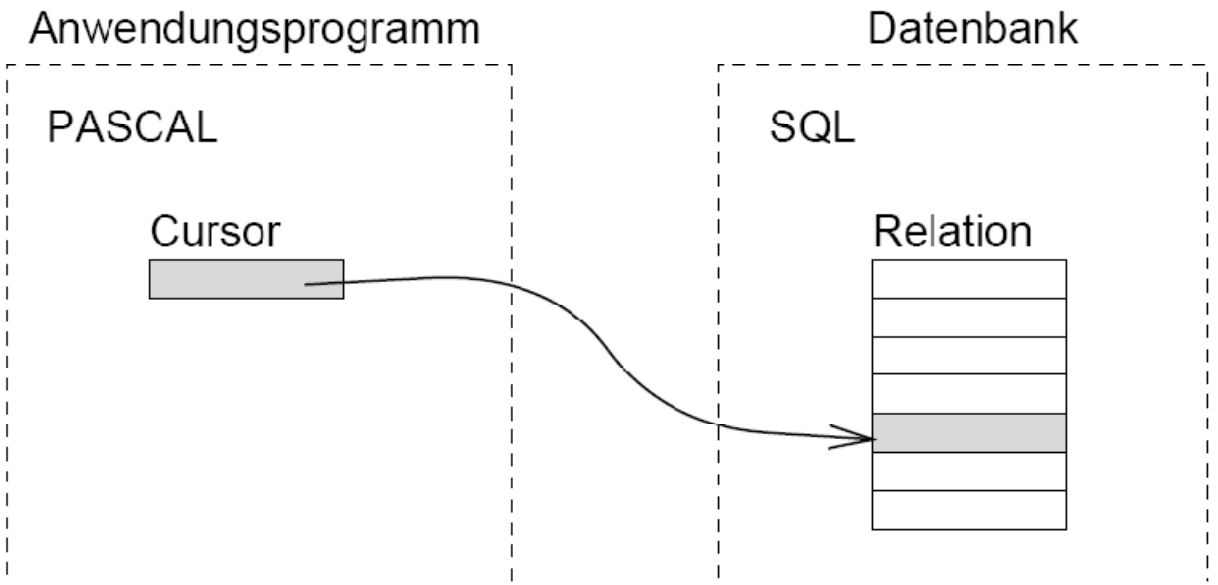


Abbildung: Kai-Uwe Sattler (TU Ilmenau)

Cursor – Beispiel


```
void gehaltsBereiche() {
    int i, stellen, counts[15];
    EXEC SQL BEGIN DECLARE SETION;
        int gehalt; char SQLSTATE[6];
    EXEC SQL END DECLARE SETION;
    EXEC SQL DECLARE managerCursor CURSOR FOR
        SELECT Gehalt FROM Manager;

    EXEC SQL OPEN managerCursor;
    for(i=0; i < 15; i++) counts[i] = 0;
    while(1) {
        EXEC SQL FETCH FROM managerCursor INTO :gehalt;
        if(strcmp(SQLSTATE,“02000“)) break;
        stellen = 1;
        while((gehalt /= 10) > 0) stellen++;
        if(stellen <= 14) counts[stellen]++;
        }
    EXEC SQL CLOSE managerCursor;
    for (i=0; i<15; i++)
        printf(„Stellen = %d: Anzahl Manager = %d\n“, i,
counts[i]);
}
```

Überblick

- Embedded SQL
- Stored procedures / PSM
- Call-level-interface (CLI) und Java Database Connectivity (JDBC)

PSM / Stored Procedures

- PSM: Persistent, Stored Modules
  - „Gespeicherte Prozeduren“
  - *Stored Procedures*
- Speicherung von Prozeduren als Datenbankelemente
- Mischung aus SQL und Programmiersprache
- Prozeduren können in SQL Anfragen oder anderen SQL Ausdrücken verwendet werden.
- CREATE PROCEDURE <name> (<parameter>)
  lokale Variablendeklarationen
  body der Prozedur;
- CREATE FUNCTION <name> (<parameter>) RETURNS <Typ>
  lokale Variablendeklarationen
  body der Funktion;

Beispiel – Cursor und Schleifen

- Gegeben ein Studioname, berechne Mittelwert und Standardabweichung der Filmlängen des Studios
-
```
CREATE PROCEDURE MeanVar(
          IN s CHAR[15],
          OUT mean REAL,
          OUT variance REAL
    )
    DECLARE Not_Found CONDITION FOR SQLSTATE '02000';
    DECLARE FilmCursor CURSOR FOR
          SELECT Laenge FROM Filme WHERE StudioName=s;
    DECLARE neueLaenge INTEGER;
    DECLARE filmAnzahl INTEGER;
```

Beispiel – Cursor und Schleifen

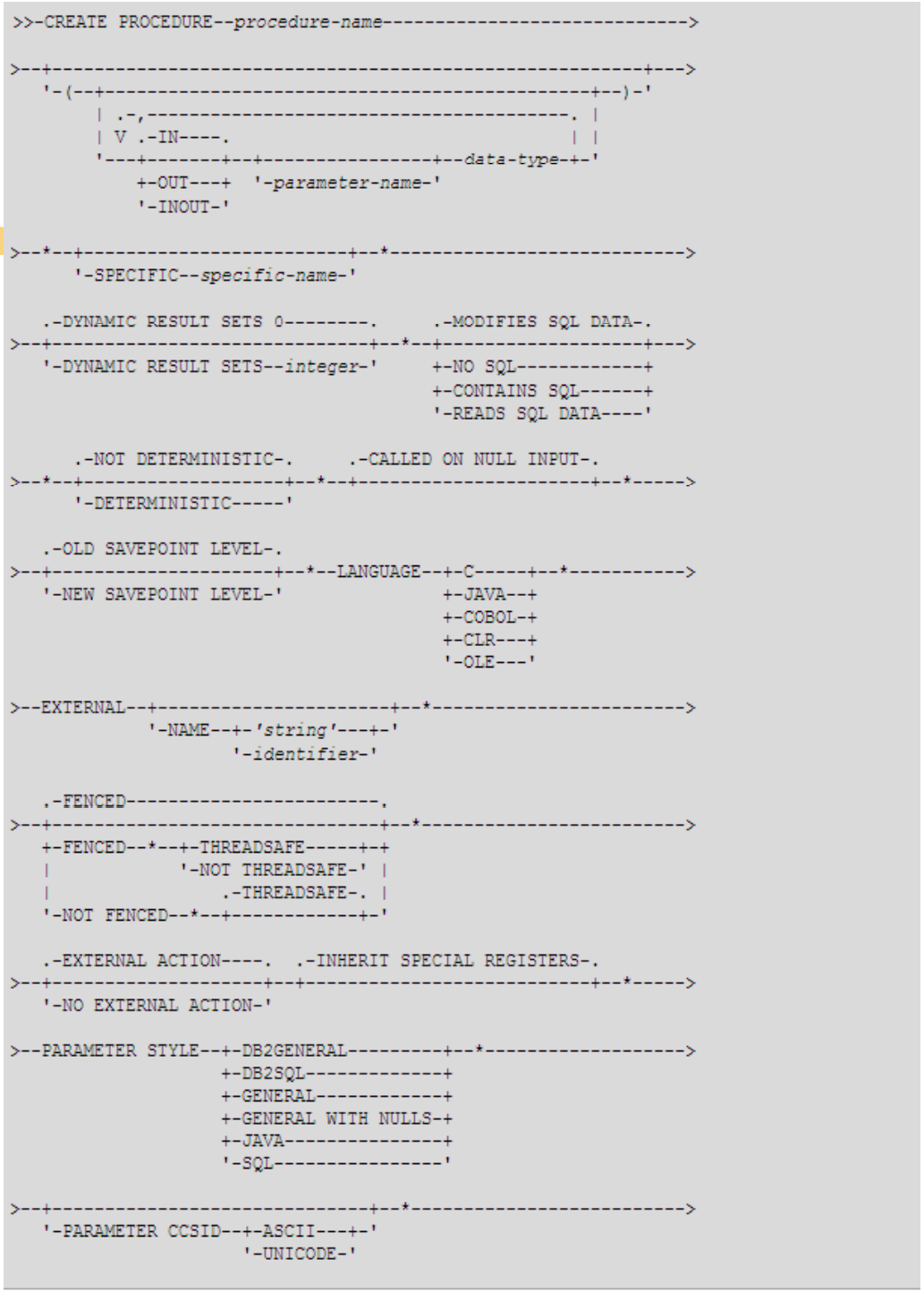

```
BEGIN
    SET mittelwert = 0.0;
    SET varianz = 0.0;
    SET filmAnzahl=0;
    OPEN FilmCursor;
    FilmLoop: LOOP
           FETCH FilmCursor INTO newLength;
           IF Not_Found THEN LEAVE FilmLoop END IF;
           SET filmAnzahl = filmAnzahl + 1;
           SET mittelwert = mittelwert + neueLaenge ;
           SET varianz = varianz + neueLaenge * neueLaenge;
    END LOOP;
    CLOSE FilmCursor;
    SET mittelwert = mittelwert/filmAnzahl;
    SET varianz = varianz/filmAnzahl - mittelwert *
                                          mittelwert;
END;
```

Externe Stored Procedures

- Stored Procedures basierend auf einem Stück Code
- Beispiel für DB2

```
CREATE PROCEDURE PARTS_ON_HAND
    (IN PARTNUM INTEGER,
     OUT COST DECIMAL(7,2),
     OUT QUANTITY INTEGER)
EXTERNAL NAME 'parts.onhand'
LANGUAGE JAVA
PARAMETER STYLE JAVA
```

- Code muss in bestimmtem Verzeichnis liegen



Überblick

- Embedded SQL
- Stored procedures / PSM
- Call-level-interface (CLI) und Java Database Connectivity (JDBC)

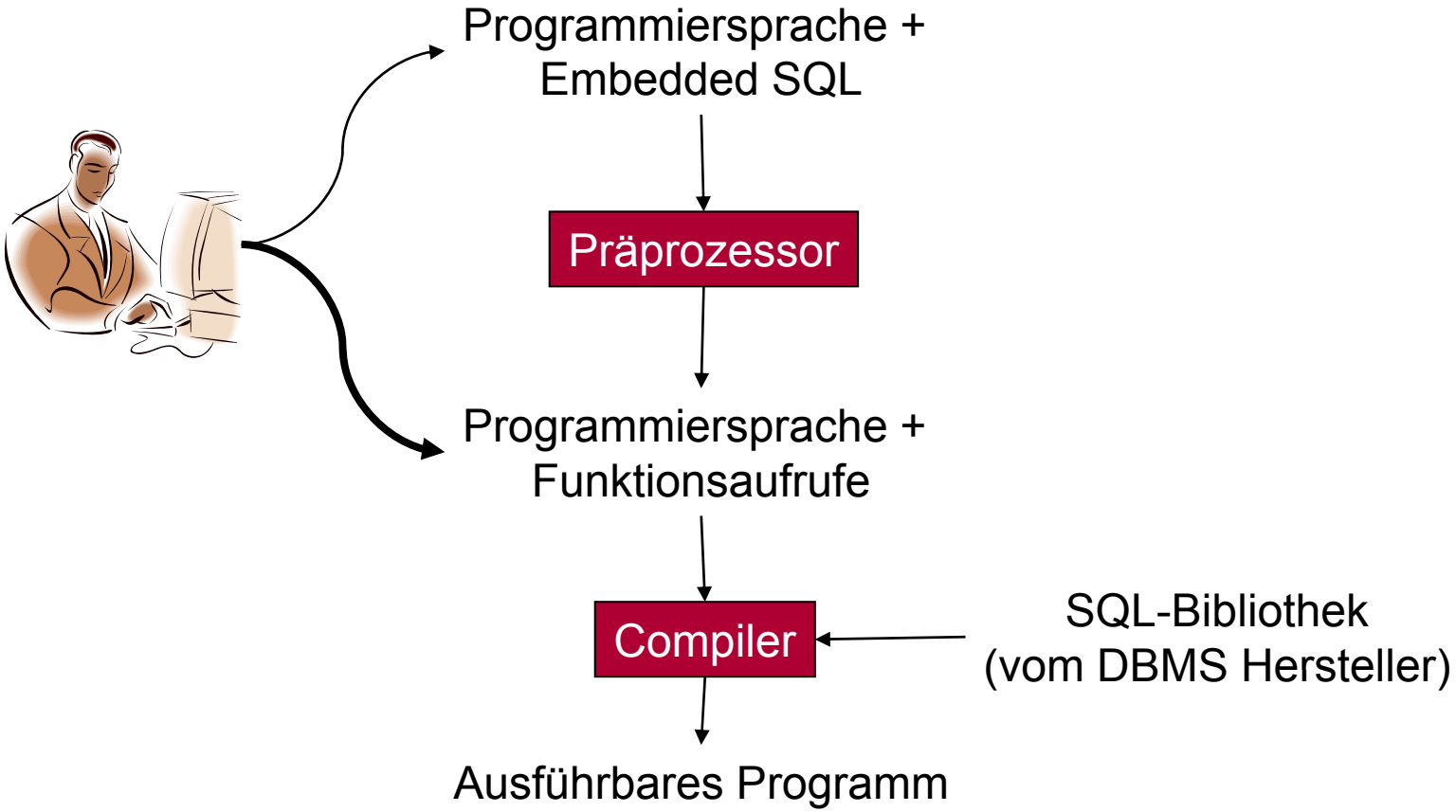
CLI: Call-Level-Interface

Idee

- Programmieren in einer Programmiersprache (Wirtssprache)
- Verwendung spezieller Funktionen für den Datenbankzugriff
  - Funktionsbibliothek
- Umgehung des Präprozessors
  - Kompiliertes Ergebnis ist gleich

- Hier: SQL/CLI
  - Adaptiert von ODBC (Open Database Connectivity)
  - Für die Programmiersprache C
- Und: JDBC (Java Database Connectivity)

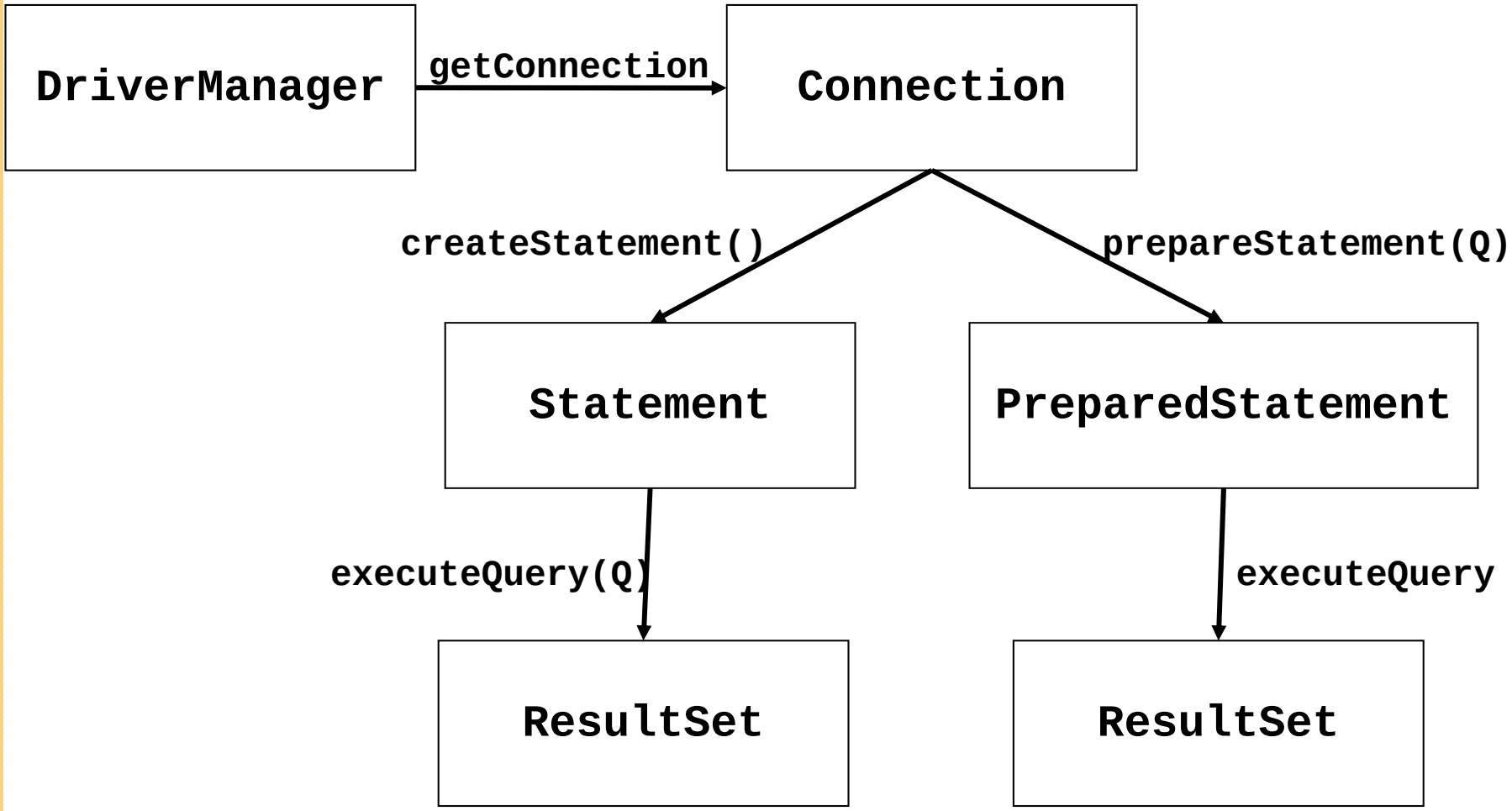
Embedded SQL - Ablauf



JDBC: Vergleich zu SQL/CLI

- Gleiches Ziel wie SQL/CLI
- Java statt C als Programmiersprache
- Unterschiede aufgrund der Objektorientierung von Java

Überblick


DriverManager
getConnection
Connection
createStatement()
prepareStatement(Q)
Statement
PreparedStatement
executeQuery(Q)
executeQuery
ResultSet
ResultSet

Erste Schritte

- Treiber für das DBMS einbinden
  - DBMS spezifisch
  - Wird jeweils mit DBMS mitgeliefert
- Verbindung zur Datenbank aufbauen
  - **`Connection meineConnection = DriverManager.getConnection(URL, name, password);`**
  - URL ist DBMS und Datenbank-spezifisch
    - "jdbc:subprotocol:datasource";
  - **`Connection meineConnection = DriverManager.getConnection("jdbc:db2://paprika:50010/FILM1DB", "db2stud","1hugo2");`**

Ausdrücke in JDBC

- Statement
  - **`createStatement();`**
  - Noch keiner SQL-Anfrage assoziiert
- PreparedStatement
  - **`prepareStatement(Anfrage);`**
  - Falls Anfrage öfters ausgeführt wird.
- SQL Ausdrücke ausführen (4 Varianten)
  1. **`ResultSet meinErgebnis = executeQuery(Anfrage);`**
     - ResultSet als Rückgabewert
  2. **`ResultSet meinErgebnis = executeQuery();`**
     - Für PreparedStatements
     - ResultSet als Rückgabewert
  3. **`excuteUpdate(UpdateAnfrage)`**
     - Kein Rückgabewert
  4. **`excuteUpdate()`**
     - Für PreparedStatements
     - Kein Rückgabewert

JDBC – Beispiel

- Gegeben die Connection **meineConnection**
- Anfrage: SELECT Gehalt FROM Manager; (2 Varianten der Ausführung)
  1. ```
     Statement managerStat = meineConnection.createStatement();
     ResultSet gehaelter = managerStat.executeQuery(
                     „SELECT Gehalt FROM Manager“);
     ```
  2. ```
     PreparedStatement managerStat =
     meineConnection.prepareStatement(
                     „SELECT Gehalt FROM Manager“);
     ResultSet gehaelter = managerStat.executeQuery();
     ```
- Updates: Schauspieler einfügen
  - ```
    Statement spielerStat = meineConnection.craeteStatement();
    spielerStat.executeUpdate(
    „INSERT INTO spielt_in VALUES(„ +
    „‘Star Wars‘, 1979, ‚Harrison Ford‘ “);
    ```
    ```

Cursor in JDBC (ResultSet)

- Methoden des ResultSet
  - **`next()`**
    - liefert nächstes Tupel
    - FALSE falls kein weiteres Tupel vorhanden
  - **`getString(i)`**
    - Liefert Wert des i-ten Attributs
    - **`getInt(i)`**, **`getFloat(i)`** usw.
  -
```
while(gehaelter.next()){
        gehalt = gehaelter.getInt(1);
        // gehalt ausgeben o.ä.
}
```

Parameter übergeben

- Mittels **`PreparedStatement`**
- Fragezeichen als Platzhalter für Parameter
  - Bindung mittels **`setString(i, v)`**, **`setInt(i, v)`** usw.
-
```
PreparedStatement studioStat =
meineConnection.prepareStatement(
    „INSERT INTO Studio(Name, Adresse) VALUES(?, ?)“);
// Werte für studioName und studioAdr vom Nutzer
einholen
studioStat.setString(1, studioName);
studioStat.setString(2, studioAdr);
studioStat.executeUpdate();
```

Zusammenfassung

- Embedded SQL
  - Kombiniert SQL mit 7 Programmiersprachen
    - ADA, C, Cobol, Fortran, M, Pascal, PL/I
  - Einbettung von SQL durch Preprocessing
- Stored procedures / PSM
  - Speicherung von Prozeduren als DBMS Objekte
  - Aufruf aus SQL Ausdrücken
- Call-level-interface (CLI)
  - Verbindet C mit DBMS
  - Spezielle Funktionsbibliothek
  - Spart das Preprocessing
- Java Database Connectivity (JDBC)
  - Wie CLI aber für Java