

# Informations- und Wissensmanagement

## Kapitel 7: Schnittstellen und Anwendungsprogrammierung



## Anwendungsprogrammierung

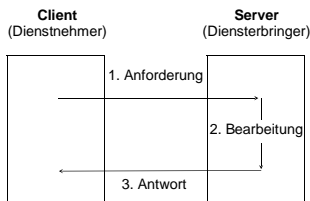
Einleitung  
Cursor  
CLI  
Embedded SQL  
Stored Proc.

- Client-Server-Architektur,
- Anbindung von Programmiersprachen,
- Call-Level-Schnittstellen: SQL/CLI, JDBC,
- Einbettung: Embedded SQL, SQLJ,
- gespeicherte Prozeduren, prozedurale Erweiterungen: SQL/PSM, PL/SQL.



## Client-Server-Architektur

Einleitung  
Cursor  
CLI  
Embedded SQL  
Stored Proc.



- Typische Architektur von DB-Anwendungen, da zentrale Verwaltung durch DBMS notwendig.
- Prinzip:  
Client nimmt Dienste eines Servers in Anspruch.



## Anwendungslogik

Einleitung  
Cursor  
CLI  
Embedded SQL  
Stored Proc.



- **Datenbank:** DB der Personalabteilung enthält Mitarbeiter-Daten – Stammdaten, Kompetenzen, vergangene Projekte, etc.
- **Anwendung:** Programm, das Teamleiter für konkrete Projekte Mitarbeiter vorschlägt.



## Anwendungslogik in der Datenbank

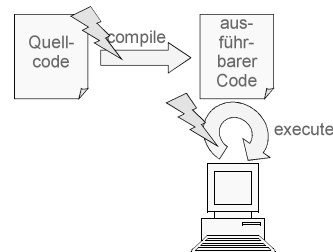
Einleitung  
Cursor  
CLI  
Embedded SQL  
Stored Proc.

- **Funktionsintegration:** unterschiedliche Applikationen nutzen gemeinsamen Code, Redundanzfreiheit
- **Transaktionsübergreifende Optimierung:** In welcher Reihenfolge erfolgen die Zugriffe auf die Datenbank?
- **Fehler:** Programmierfehler in SQL zur Übersetzungszeit entdecken
- **Sicherheit:** Zugriffskontrolle der Datenbank schützt 'Business-Wissen' im Code
- **Performanz:** Ausführung 'nahe' an den Daten



## Laufzeitfehler vs. Fehler zur Übersetzungszeit

Einleitung  
Cursor  
CLI  
Embedded SQL  
Stored Proc.



### Schema-unabhängige vs. Schema-spezifische Fehler

Einleitung  
Cursor  
CLI  
Embedded SQL  
Stored Proc.

- Beispiel für Schema-unabhängigen Fehler:
 

```

      SELEKTIERE Ename, Sal
      FROM Emp
      WHERE Job = "Professor";
      
```

falsches Schlüsselwort
- Beispiel für Schema-spezifischen Fehler:
 

```

      SELECT Ename, Sal
      FROM Emp1
      WHERE Job = "Professor";
      
```

Relation nicht in Datenbank

IWM: Schnittstellen und Anwendungsprogrammierung - 7

### Cursor-Konzept

Einleitung  
Cursor  
CLI  
Embedded SQL  
Stored Proc.

- Programmiersprachen – einzelne **Datenitems** als zugrundeliegende Struktur, SQL – **Menge** von Tupeln,

→ **Impedance Mismatch**: Cursor wird benötigt für Übergang von Mengen auf Items

IWM: Schnittstellen und Anwendungsprogrammierung - 8

### Programmiersprachenanbindung

Einleitung  
Cursor  
CLI  
Embedded SQL  
Stored Proc.

- SQL-Code zur Entwicklungszeit / Laufzeit erzeugen
  - wann Fehler entdecken?
  - wann optimieren?
- Art der Anbindung an Programmiersprachen
  - Entwicklungsaufwand, Sicherheit der Sprache
- Art der Kopplung an DB
  - Code eingebettet im DB-Server vs. Zugriff über externe Schnittstellen

IWM: Schnittstellen und Anwendungsprogrammierung - 9

### Programmiersprachenanbindung

Einleitung  
Cursor  
CLI  
Embedded SQL  
Stored Proc.

enge Kopplung mit DB-Server

Kopplung mit Datenbank

Zugriff von irgendwo

Erzeugung SQL-Code

Embedded SQL, SQLJ (statisch)

Dynamic SQL (dynamisch)

generisch Einbettung in Programmiersprache

CLI, JDBC

IWM: Schnittstellen und Anwendungsprogrammierung - 10

### SQL/CLI: Der Standard (1)

Einleitung  
Cursor  
CLI  
Embedded SQL  
Stored Proc.

- Call-Level-Interface: Bibliothek von Prozeduren/Funktionen zur
  - Kommunikation mit dem DBMS,
  - Definition und Ausführung von Anfragen,
  - Verarbeitung von Ergebnissen.

IWM: Schnittstellen und Anwendungsprogrammierung - 11

### SQL/CLI: Der Standard (2)

Einleitung  
Cursor  
CLI  
Embedded SQL  
Stored Proc.

Anwendung

ResultSet rs = ...

Treiber (für Oracle)

Treiber (für DB2)

DBMS (z. B. Oracle)

DBMS (z. B. DB2)

IWM: Schnittstellen und Anwendungsprogrammierung - 12

## JDBC

Einleitung  
Cursor  
CLI  
Embedded  
SQL  
Stored  
Proc.

- JDBC – ‚Java Database Connectivity‘.
- SQL/CLI-konforme Implementierung des Datenbankzugriffs für Java
  - API für die Übergabe von SQL-Kommandos
- Zugriff auf verschiedene Datenbanksysteme über systemspezifische Treiber möglich,
- Registrierung von Datenquellen mit Name, System, Treiber, Verbindungsinformation.
- *Achtung:* nur die Schnittstelle ist definiert, d.h., spezifische Eigenheiten von unterschiedlichen DBMS müssen weiter beachtet werden!

## JDBC: Überblick

Einleitung  
Cursor  
CLI  
Embedded  
SQL  
Stored  
Proc.

- Java-Package `java.sql` – wichtige Klassen dieses Packages:
  - `DriverManager`: Einstiegspunkt, Laden von Treibern und Grundlage für Aufbau der Datenbankverbindung,
  - `Connection`: Datenbankverbindung,
  - `Statement`: Ausführung von Anweisungen über eine Verbindung,
  - `ResultSet`: verwaltet Ergebnisse einer Anfrage in Form einer Relation, Zugriff auf einzelne Spalten.

## JDBC: Ablauf

Einleitung  
Cursor  
CLI  
Embedded  
SQL  
Stored  
Proc.

1. Aufbau einer Verbindung zur Datenbank:
  - Angabe der Verbindungsinformationen,
  - Auswahl und Laden des Treibers,
2. Senden einer SQL-Anweisung:
  - Definition der Anweisung als String,
  - Übergabe von Parametern  
`insert into <wohin?> values (<was?>)`
3. Verarbeiten der Anfrageergebnisse:
  - Navigation über Ergebnisrelation,
  - Zugriff auf Spalten.

## JDBC: Verbindungsaufbau (1)

Einleitung  
Cursor  
CLI  
Embedded  
SQL  
Stored  
Proc.

- Treiber laden, z. B.:  
`DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());`  
Auch möglich: System-Property mit Treiber-Namen, die automatisch geladen werden.

## JDBC: Verbindungsaufbau (2)

Einleitung  
Cursor  
CLI  
Embedded  
SQL  
Stored  
Proc.

- Verbindung herstellen:  

```
Connection con; // Connection-Objekt
con = DriverManager.getConnection(
    "jdbc:oracle:thin:@benriach1521:tox",
    "user", "passwd");
```

  - Erster Parameter spezifiziert
    - Datenquelle/Datenbank,
    - Verbindungsmechanismus – Protokoll, Server-Host und Port
    - zu verwendender Treiber.
  - mehrere parallele Verbindungen möglich, d.h. paralleler Zugriff auf unterschiedliche Datenbanken



## JDBC: Anfrageausführung (1)

Einleitung  
Cursor  
CLI  
Embedded  
SQL  
Stored  
Proc.

- Anweisungsobjekt (Statement) erzeugen:  
`Statement stmt = con.createStatement();`
- Anweisung ausführen:  

```
String query =
    "SELECT titel, preis FROM buch";
ResultSet rs = stmt.executeQuery(query);
```

## JDBC: Anfrageausführung (2)

Einleitung  
Cursor  
CLI  
Embedded SQL  
Stored Proc.

- Klasse `java.sql.Statement`:
    - Ausführung von Anfragen (SELECT) mit `executeQuery`,
    - Ausführung von Änderungsanweisungen (DELETE, INSERT, UPDATE) mit `executeUpdate`.
- Hat Ergebnis vom Typ `int` – Anzahl der Tupel, die geupdatet wurden.

## JDBC: Ergebnisverarbeitung

Einleitung  
Cursor  
CLI  
Embedded SQL  
Stored Proc.

- Navigation über Ergebnismenge (Cursor-Prinzip):
 

```
while (rs.next ()) {
    // Verarbeitung der einzelnen Tupel
    ...
}
```
- Zugriff auf Spaltenwerte des aktuellen Tupels über `getType`-Methoden
  - über Spaltenindex:
 

```
String titel = rs.getString(1);
```
  - über Spaltenname:
 

```
String titel = rs.getString("titel");
```

## JDBC: Fehlerbehandlung

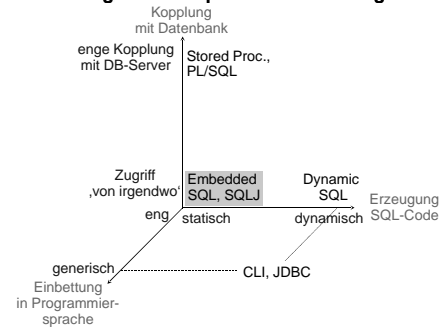
Einleitung  
Cursor  
CLI  
Embedded SQL  
Stored Proc.

- Fehlerbehandlung mittels Exception-Mechanismus,
- `SQLException` für alle SQL- und DBMS-Fehler:
 

```
try {
    // Aufruf von JDBC-Methoden
    ...
} catch (SQLException exc) {
    System.out.println ("SQLException:"+
        exc.getMessage ());
}
```

## Programmiersprachenanbindung

Einleitung  
Cursor  
CLI  
Embedded SQL  
Stored Proc.



## Embedded SQL (1)

Einleitung  
Cursor  
CLI  
Embedded SQL  
Stored Proc.

- In Anwendungsprogramm eingestreute SQL-Anweisungen.
- **Precompiler** erledigt Umsetzung in Prozeduraufrufe einer Bibliothek.
- Schlüsselwort `exec sql` oder anderes – Erkennung durch Precompiler

## Embedded SQL (2)

Einleitung  
Cursor  
CLI  
Embedded SQL  
Stored Proc.

- **Beispiel – SQLJ Codefragment:**

```
String vName; int vSalary; ...
#sql { SELECT Ename, Sal
    INTO :vName, :vSalary
    FROM Emp
    WHERE Job = "Professor" };
```

## Vorteile

Einleitung  
Cursor  
CLI  
Embedded SQL  
Stored Proc.

Insbesondere im Vergleich zu JDBC:

- Kompakter Code
- Syntax- und Semantik-Check der SQL-Statements zur Übersetzungszeit, gut für Performance.

## SQLJ vs. JDBC (1)

Einleitung  
Cursor  
CLI  
Embedded SQL  
Stored Proc.

- **SQLJ Codefragment:**  

```
String vName; int vSalary; String vJob;
Java.sql.Timestamp vDate; ...
#sql { SELECT Ename, Sal
      INTO :vName, :vSalary
      FROM Emp
      WHERE Job = :vJob
            and HireDate = :vDate };
```
- D. h. wir greifen auf das erste Tupel zu.

## SQLJ vs. JDBC (2)

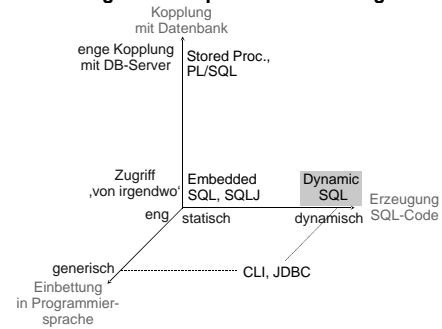
Einleitung  
Cursor  
CLI  
Embedded SQL  
Stored Proc.

- Entsprechendes JDBC Codefragment:  

```
String vName; int vSalary; String vJob;
Java.sql.Timestamp vDate; ...
PreparedStatement stmt =
connection.prepareStatement(
"SELECT Ename, Sal " +
"INTO :vName, :vSalary " + "FROM Emp " +
"WHERE Job = :vJob and HireDate=:vDate");
stmt.setString(1, vJob);
stmt.setTimestamp(2, vDate);
ResultSet rs = stmt.executeQuery();
rs.next();
vName = rs.getString(1);
vSalary = rs.getInt(2);
rs.close();
```

## Programmiersprachenanbindung

Einleitung  
Cursor  
CLI  
Embedded SQL  
Stored Proc.



## Dynamic SQL (Oracle)

Einleitung  
Cursor  
CLI  
Embedded SQL  
Stored Proc.

- Sowohl enge Einbettung in Programmiersprache als auch dynamische Erzeugung des SQL-Codes.

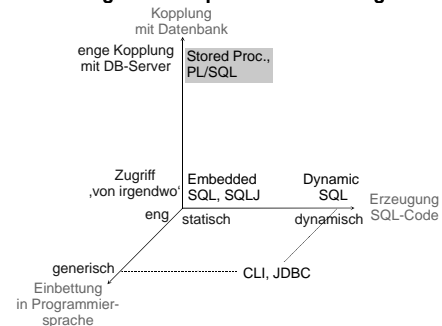
- Anfragen als Zeichenketten:

```
exec sql begin declare section;
      AnfrageString char(256) varying;
exec sql end declare section;
exec sql declare AnfrageObjekt
statement;
AnfrageString :=
      'DELETE FROM Buch_Versionen ' +
      'WHERE ISBN = ? AND Auflage = ?' ;
exec sql prepare AnfrageObjekt
      from :AnfrageString;
exec sql execute AnfrageObjekt
      using :LöschISBN, :LöschAuflage;
```

Wertübernahme aus Wirtssprache

## Programmiersprachenanbindung

Einleitung  
Cursor  
CLI  
Embedded SQL  
Stored Proc.



## Gespeicherte Prozeduren

Einleitung  
Cursor  
CLI  
Embedded  
SQL  
Stored  
Proc.



- Probleme von CLI und Embedded SQL:
  - Ständiger Wechsel der Ausführungskontrolle zwischen Anwendung und DBS; *Kontextwechsel*.
  - Anweisungsübergreifende Optimierung kompliziert

## Gespeicherte Prozeduren (2)

Einleitung  
Cursor  
CLI  
Embedded  
SQL  
Stored  
Proc.

- Ausweg – *gespeicherte Prozeduren* („SQL um Ablaufkonstrukte erweitern“):
  - Im Datenbank-Server verwaltete und auch dort ausgeführte Software-Module in Form von Prozeduren bzw. Funktionen – *Funktionsintegration*.
  - Aufruf aus Anwendungen und Anfragen heraus.

## PL/SQL Programm – Beispiel

Einleitung  
Cursor  
CLI  
Embedded  
SQL  
Stored  
Proc.

- Erzeuge Cursor, Navigiere über Cursor, modifiziere/lösche Tupel unter dem Cursor
- ```
DECLARE
  a T1.e$TYPE; b T1.f$TYPE;
  CURSOR T1Cursor IS
    SELECT e, f FROM T1 WHERE e < f FOR UPDATE;
BEGIN
  OPEN T1Cursor;
  LOOP
    FETCH T1Cursor INTO a, b;
    EXIT WHEN T1Cursor%NOTFOUND;
    DELETE FROM T1 WHERE CURRENT OF T1Cursor;
    /* Insert the reverse tuple: */
    INSERT INTO T1 VALUES(b, a);
  END LOOP; /* Free cursor */
  CLOSE T1Cursor;
END;
```
- run;**

## Weitere prozedurale SQL-Erweiterungen

Einleitung  
Cursor  
CLI  
Embedded  
SQL  
Stored  
Proc.

- IBM DB2: SQL/PSM,
- Informix: SPL,
- Sybase, Microsoft SQL Server: Transact-SQL,
- externe Routinen: Implementiert z. B. in C, Java.