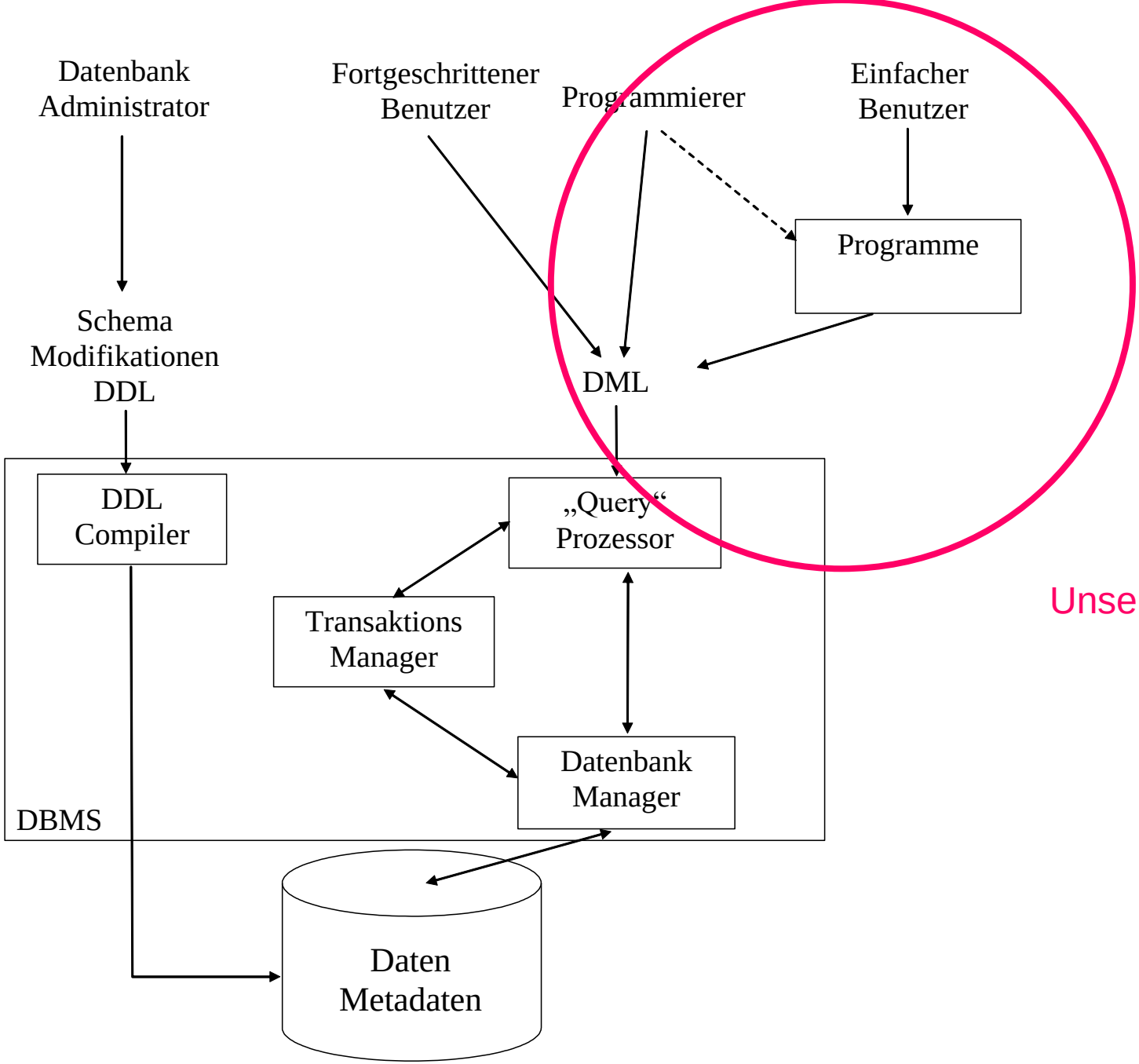
Programmieren und DBMS

Dr. Karsten Tolle

Wie kommen die Befehle zum DBMS

- Bisher gesehen:
  - SQL direkt zum DBMS
    - MySQL Workbench / HeidiSQL
    - Kommandozeile
- Weitere?



Unser Fokus heute!

- LIVE in ECLIPSE

embedded SQL

Embedded SQL (abgekürzt: ESQL) ist eine Spracherweiterung von SQL, mit der es möglich ist, SQL-Anweisungen innerhalb einer strukturierten oder objektorientierten Programmiersprache (der ***Hostsprache***) auszuführen.

Embedded SQL wurde erstmals im SQL92-Standard definiert.

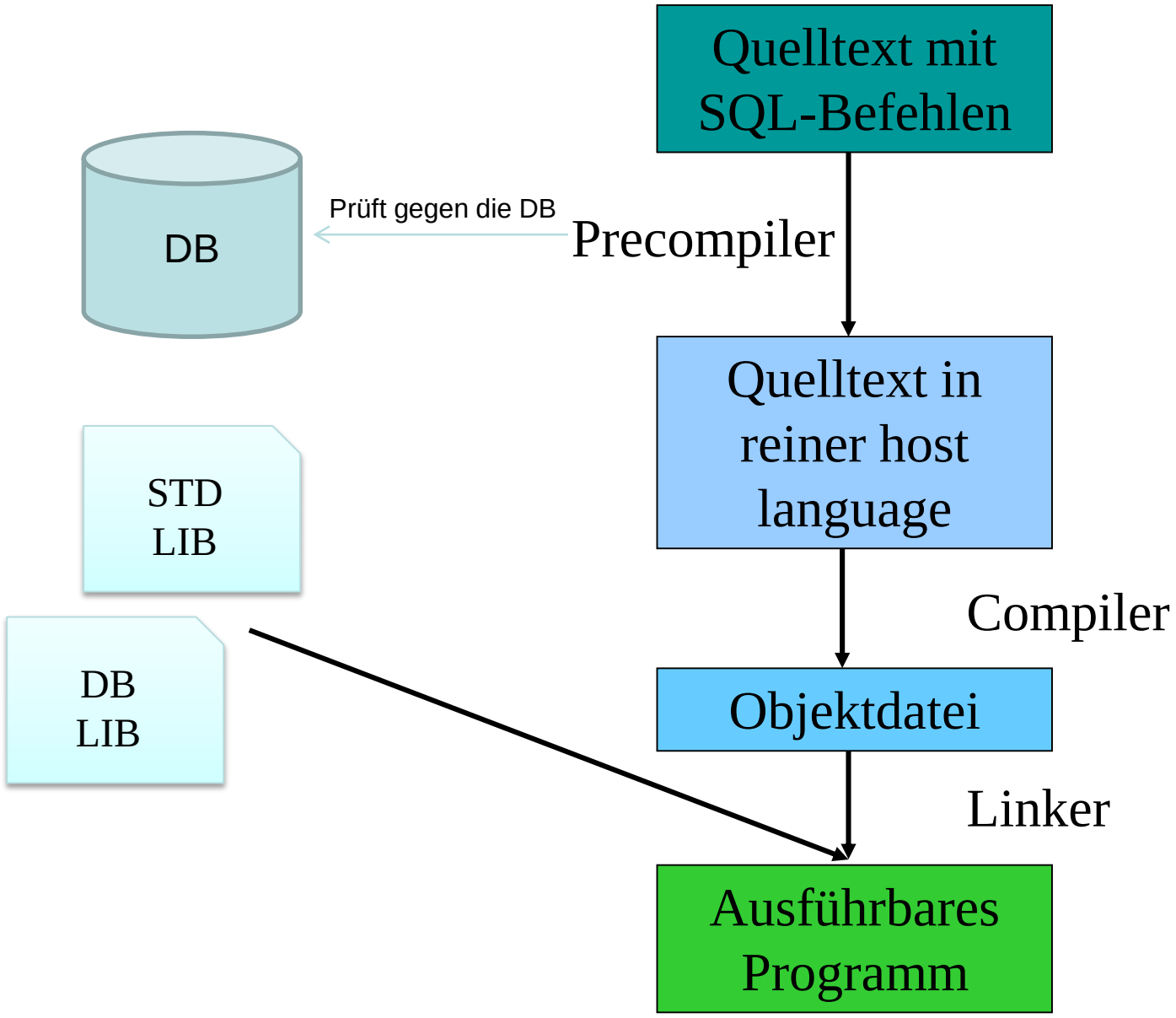
Hostsprachen

<embedded SQL host program> ::=
<embedded SQL Ada program> |
<embedded SQL C program> |
<embedded SQL COBOL program> |
<embedded SQL Fortran program> |
<embedded SQL MUMPS program> |
<embedded SQL Pascal program> |
<embedded SQL PL/I program>

Siehe Seite 489 - http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt

Embedded SQL und C - Beispiel

```
EXEC SQL BEGIN DECLARE SECTION;
   char dieBar[21], dasBier[21];
   float preis;
EXEC SQL END DECLARE SECTION;
   /* holen der Werte für dieBar und dasBier */
EXEC SQL SELECT price INTO :preis
   FROM Verkauft
   WHERE bar = :dieBar AND bier = :dasBier;
   /* die Variable preis kann nun verwendet werden*/
```


Quelltext mit SQL-Befehlen
DB
Prüft gegen die DB
Precompiler
Quelltext in reiner host language
STD LIB
DB LIB
Compiler
Objektdatei
Linker
Ausführbares Programm

Precompiler – pro/cons

- zusätzlicher Schritt beim Kompilieren (-)
- Wechsel der DB kann erneutes precompile/compile erzwingen (-)

- Validierung und Binding der SQL-Anfragen kann zur Kompilier-Zeit erfolgen (+)
  - dazu muss die Datenbank existieren (-)

SQLJ

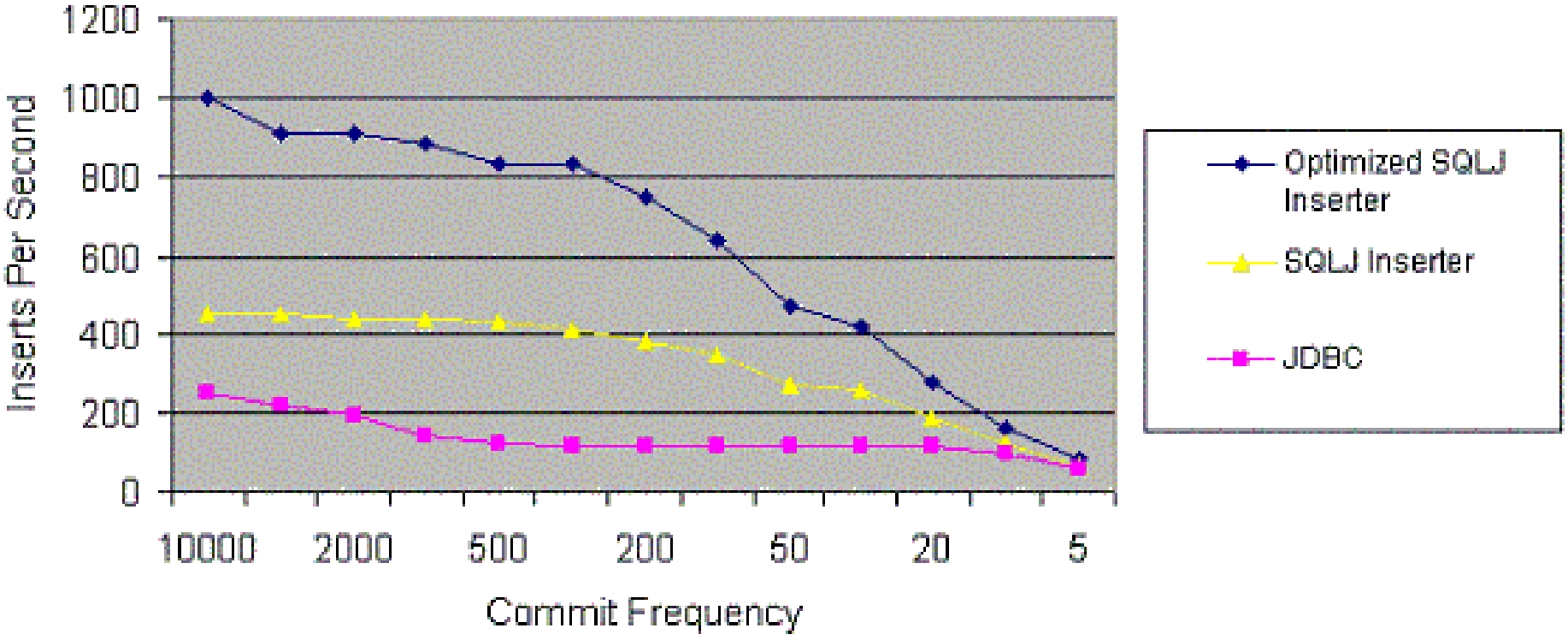
- Benötigt Präcompiler des DBMS-Herstellers!
- Überprüfungen durch Präcompiler – Syntax und Semantik (DB-Objekte richtig geschrieben?)
- Schreibweise kompakter als JDBC
- Während der Laufzeit wird JDBC verwendet! → Plattformunabhängig

SQLJ - Beispiel

```
…
#sql cur0 = {SELECT * FROM org};
 while (true) {
        // retrieve and display the result from the SELECT statement
        #sql {FETCH :cur0
                INTO :deptnumb, :deptname, :manager, :division, :location};

             if (cur0.endFetch())  { break; }
             System.out.println("    " + Data.format(deptnumb, 8) +
                        " " + Data.format(deptname, 14) +
                        " " + Data.format(manager, 7) +
                        " " + Data.format(division, 10) +
                        " " + Data.format(location, 14));
 }
 cur0.close(); // close the cursor
…
```

Performance Vergleich

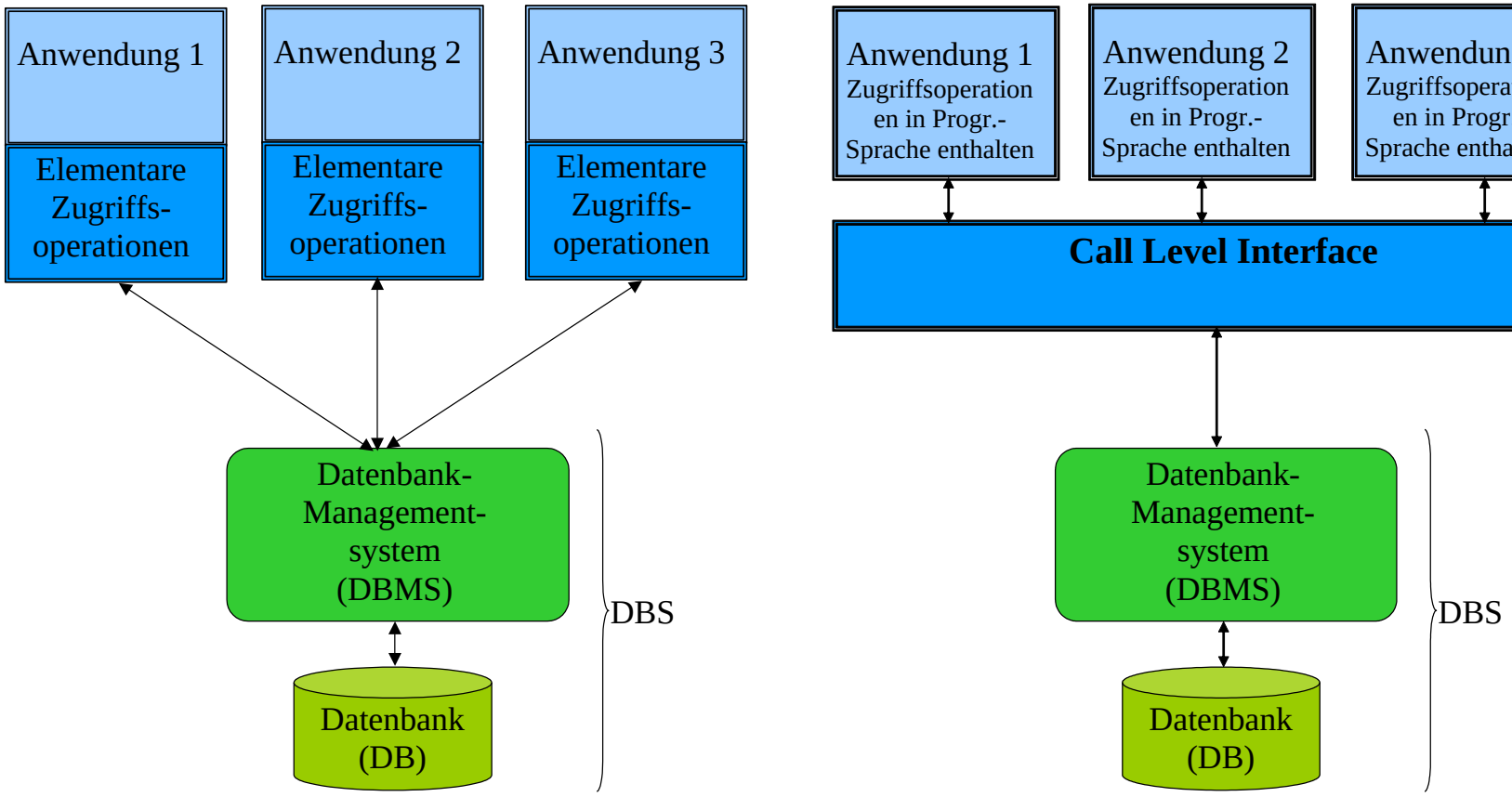


Quelle: IBM (https://www.ibm.com/developerworks/jp/data/library/dataserver/techdoc/javaperform.html)

CLI – Call Level Interface

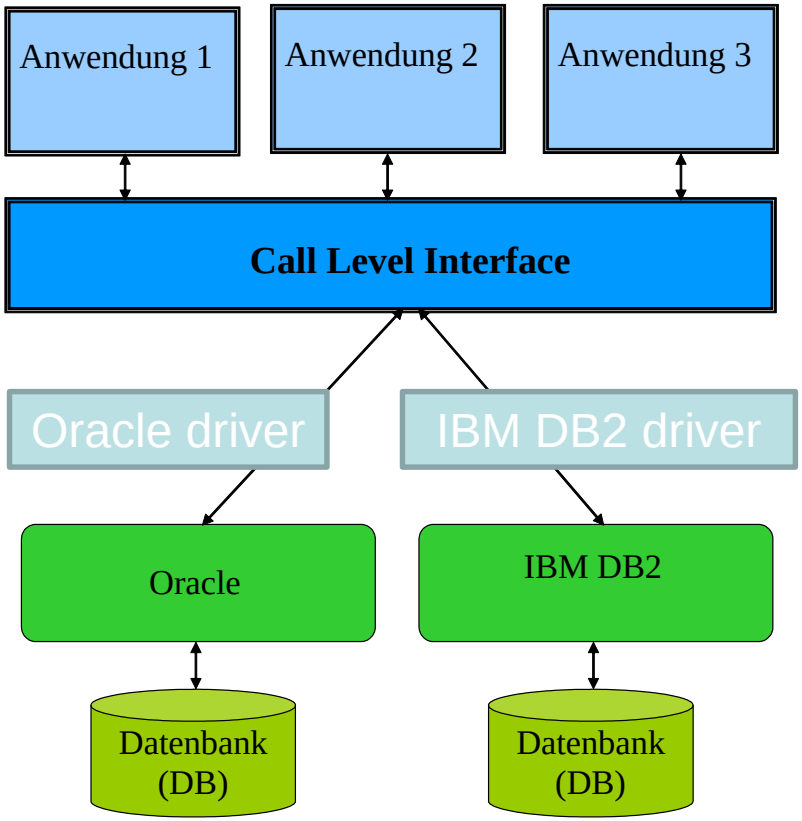
- Das **Call Level Interface** (kurz *CLI*) ist eine Datenbankschnittstellen-Spezifikation für den Zugriff auf RDBMS (baut auf SQL auf) aus anderen Anwendungen heraus.

- Weitere Details unter: https://publications.opengroup.org/standards/data-mgmt/c451


Anwendung 1
Elementare Zugriffs-operationen
Anwendung 2
Elementare Zugriffs-operationen
Anwendung 3
Elementare Zugriffs-operationen
Datenbank-Management-system (DBMS)
Datenbank (DB)
DBS
Anwendung 1
Zugriffsoperation en in Progr.-Sprache enthalten
Anwendung 2
Zugriffsoperation en in Progr.-Sprache enthalten
Anwendung 3
Zugriffsoperation en in Progr.-Sprache enthalten
Call Level Interface
Datenbank-Management-system (DBMS)
Datenbank (DB)
DBS

Vorteile CLI zu ESQL

- Kein Precompiler nötig.
- Vorteil für Client/Server Architektur, da unabhägig(er) von Zieldatenbank



- Programmierschnittstellen, die das CLI-Konzept umsetzen, sind z.B.:
  - Open Database Connectivity (ODBC),
  - Java Database Connectivity (JDBC).

Java Database Connectivity

- JDBC 1.0 (1997)
  - jdbc.sql.* als optionales Paket
  - Basierend auf SQL92
- JDBC 2.0
  - java.sql.* in JSE2* (batch-updates, SQL3-Datentypen)
  - javax.sql.* (optional - ab Java 1.3 fest) enthält Data-Source (JNDI - Java Naming and Directory Interface), Connection-Pooling, verteilte Transaktionen
  - JDBC 2.0 Treiber sind für die fast alle (bekannteren) RDBMS vorhanden

*Java 2 Standard Edition (JSE2) – auch Java2

Java Database Connectivity

- JDBC 3.0
  - Teil von Java 1.4 - neu unter anderem:
    - Savepoints in Transaktionen,
    - Wiederverwendung von PreparedStatements,
    - JDBC-Datentypen BOOLEAN und DATALINK,
    - Abrufen automatisch generierter Keys,
    - Änderungen von LOBs (Large Objects) und mehrere gleichzeitig geöffnete ResultSets
- JDBC 4.0 (aktuell 4.3 https://jcp.org/aboutJava/communityprocess/mrel/jsr221/index3.html)
  - Teil von Java 1.6 – neu hier:
    - Annotationen für SQL-Queries, Treiber werden – wenn vorbereitet – automatisch angemeldet, XMLDatentypen aus SQL:2003, Zugriff auf die SQL ROWID

Details unter: https://www.oracle.com/technetwork/java/overview-141217.html

Java Praxis - DriverManager

1. JDBC Treiber für DBMS in Classpath aufnehmen, Beispiel:

```
set JDBC_Driver="C:\MySQL\mysql-connector-java-3.1.6-bin.jar"
java –classpath %JDBC_Driver% MyAnwendung
```

2. Treiber im Program laden und aktivieren,
   Class.forName() is not needed since JDBC 4.0.
   Beispiel:

```
try {
    Class.forName(jdbcdriver);
} catch (Exception e) {}
```

3. Verbindung herstellen, Beispiel:

```
try {
    Connection con =
    DriverManager.getConnection("jdbc:mysql:///db1", "user", "pw“);
} catch (Exception e) {}
```

Java Praxis

4. SQL Statement (Objekt) erzeugen

```
Statement stmt = con.createStatement();
```

5. SQL Anfrage erzeugen und an DBS schicken:

```
ResultSet rs = stmt.executeQuery("select * from myTable");
```

6. Mit dem Ergebnis arbeiten:

```
while (rs.next()) {
        String name = rs.getString("Name");
        System.out.println("Name = "+name);
}
```

Java Praxis – DataSource Alternative für Web Server

```
// Get an initial JNDI context for locating the driver and
// database
Context ctext = new InitialContext();
// Get a DataSource object for the driver and database
// associated with a logical name
DataSource ds = (DataSource)ctext.lookup("jdbc/my_DB");
// Now, get the connection
Connection conn = ds.getConnection();
```

Vorteil der Nutzung des *javax.sql.DataSource* Interfaces ist, dass die Zugangsdaten für die Datenbank (URL, Benutzername, Passwort) nicht hart kodiert werden müssen, sondern in einer Konfigurationsdatei ausgelagert werden können. Dies erleichtert den Wechsel auf eine andere Datenbank-Instanz.

Siehe auch: http://www.w3processing.com/index.php?subMenuLoad=JDBC/DatasourceConnection.php

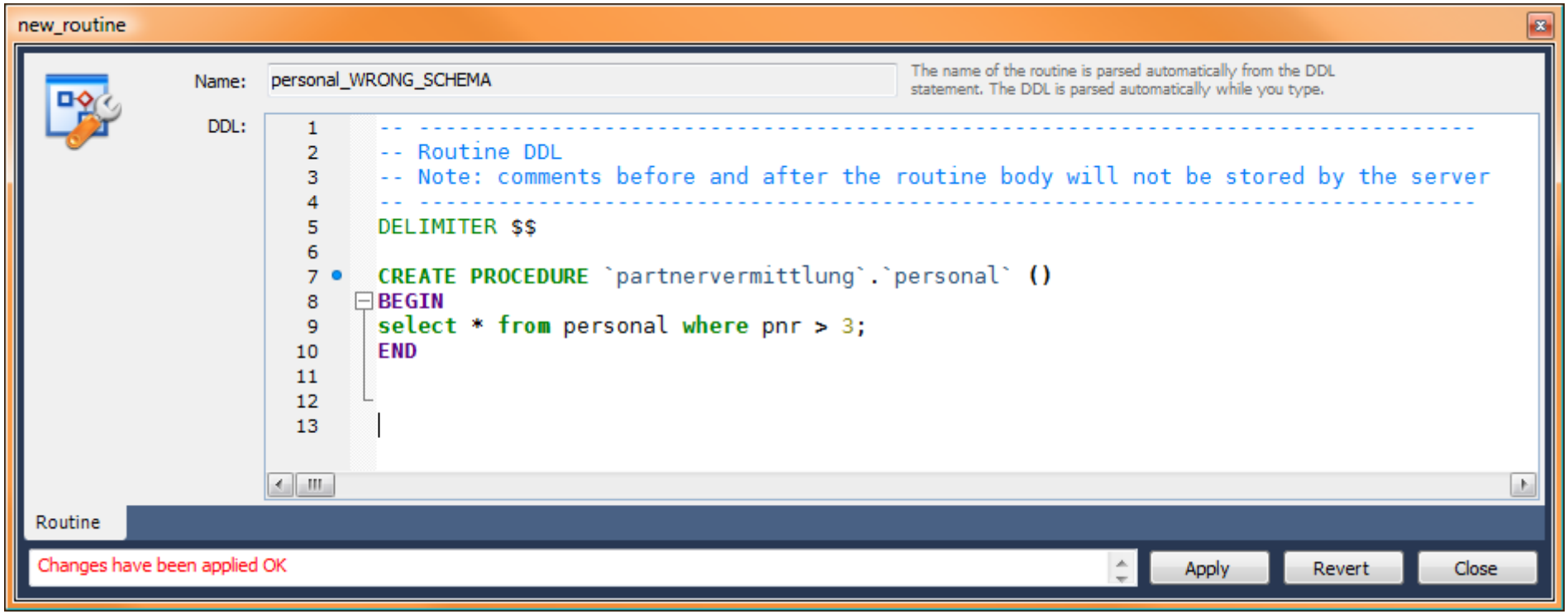
Code auf der DB-Server Seite

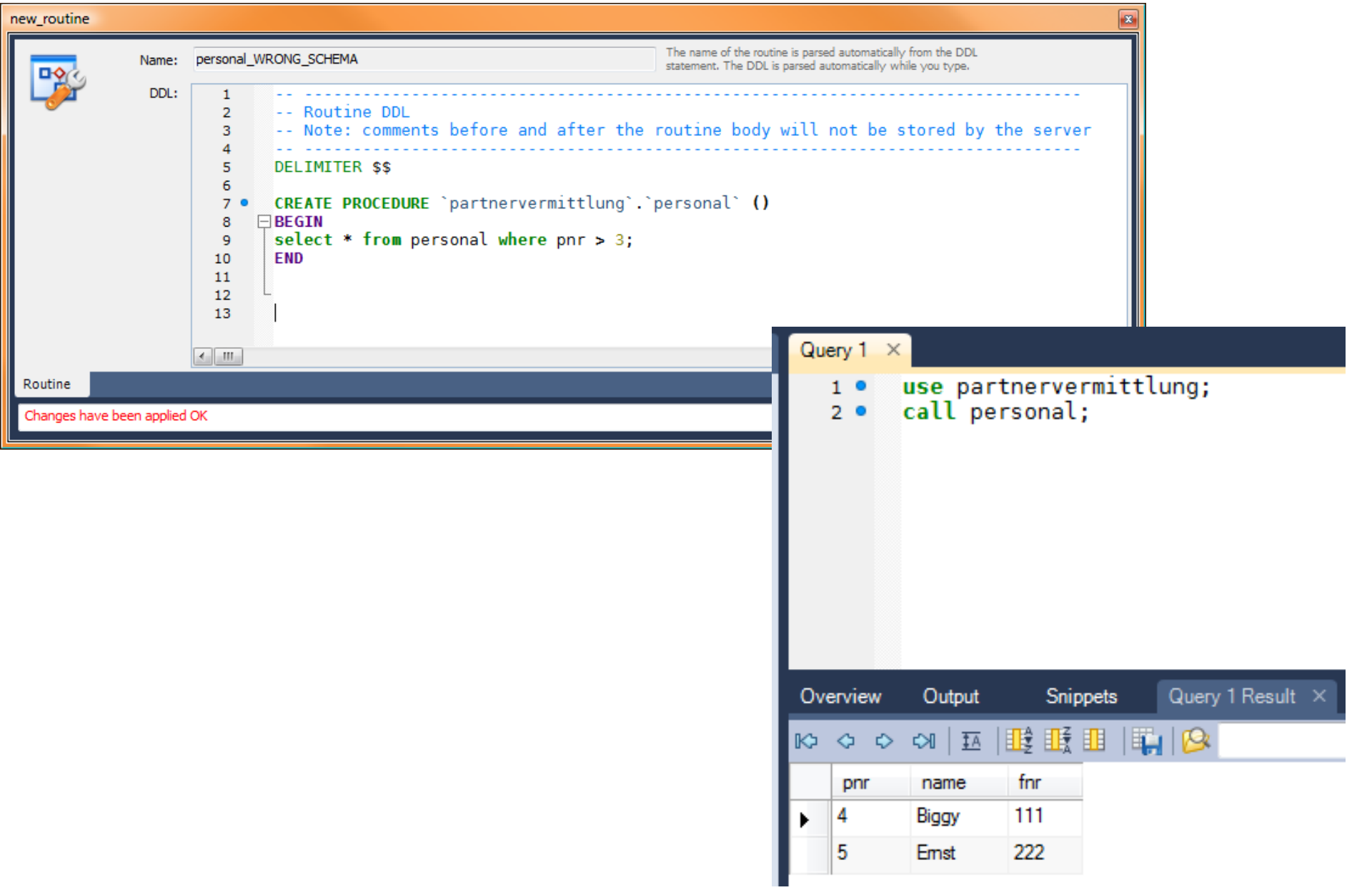
- Stored Funtions, Stored Procedures, Trigger
  - SQL-Statements werden auf der Serverseite gehalten
  - Schleifen, Bedingungen etc. vorhanden (aber Syntax oft abh. vom DBMS!)

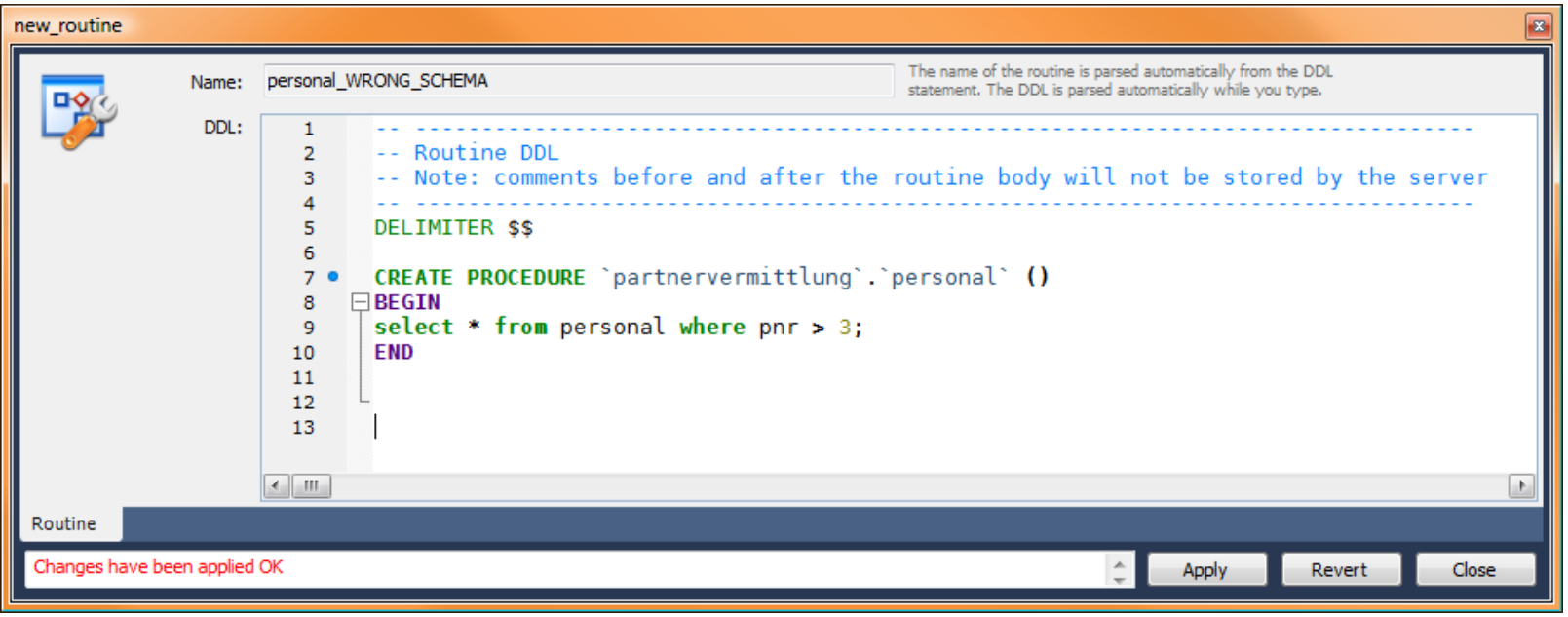
Wann insb. ist dies von Vorteil?

Tutorials: http://www.mysqltutorial.org/mysql-stored-procedure-tutorial.aspx

einfache Stored Procedure



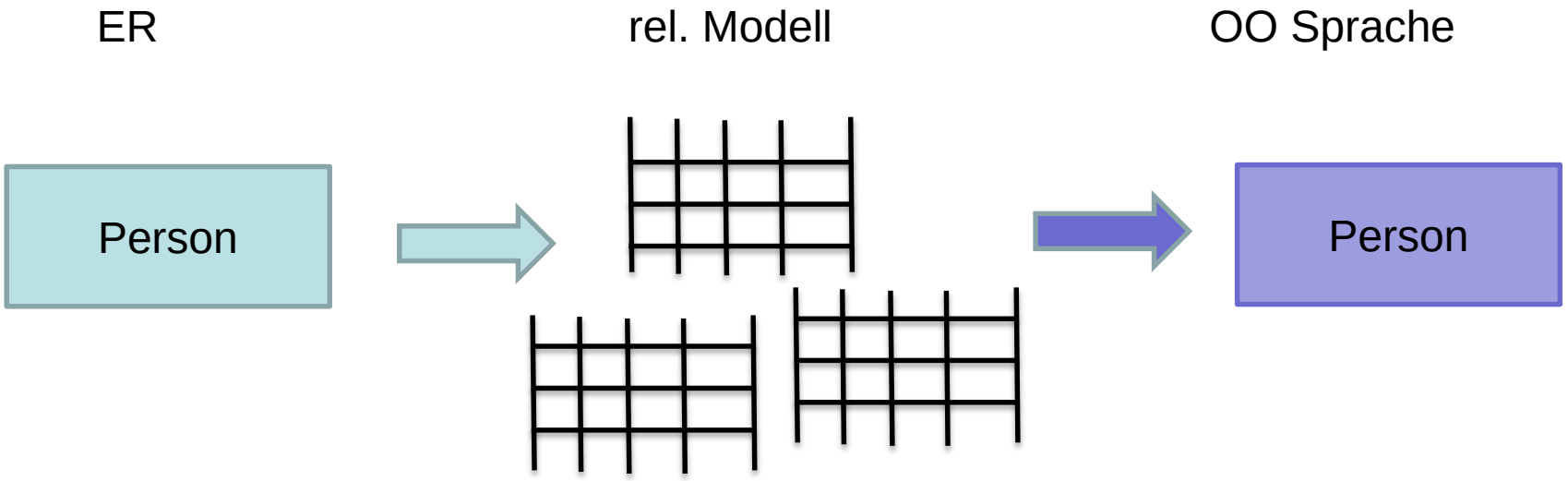

new_routine
Name:
personal_WRONG_SCHEMA
The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.
DDL:
1 -- ----------
2 -- Routine DDL
3 -- Note: comments before and after the routine body will not be stored by the server
4 -- ----------
5 DELIMITER $$
6
7 CREATE PROCEDURE `partnervermittlung`.`personal` ()
8 BEGIN
9 select * from personal where pnr > 3;
10 END
11
12
13
Routine
Changes have been applied OK
Query 1
1 use partnervermittlung;
2 call personal;
Overview
Output
Snippets
Query 1 Result
pnr
name
fnr
4
Biggy
111
5
Ernst
222




```
// Beispielaufruf aus Java
…
CallableStatement cStmt = con1.prepareCall("{call personal()}");
boolean hadResults = cStmt.execute();
if (hadResults) {
        ResultSet rs = cStmt.getResultSet();
…
```

Vorgehen bisher …

- Erstellen eines ER-Diagramms
- Übersetzen in das relationale Datenmodell
- Zugriff auf das relationale Datenmodell aus z.B. Java



Object-relational Impedance Mismatch

Definition

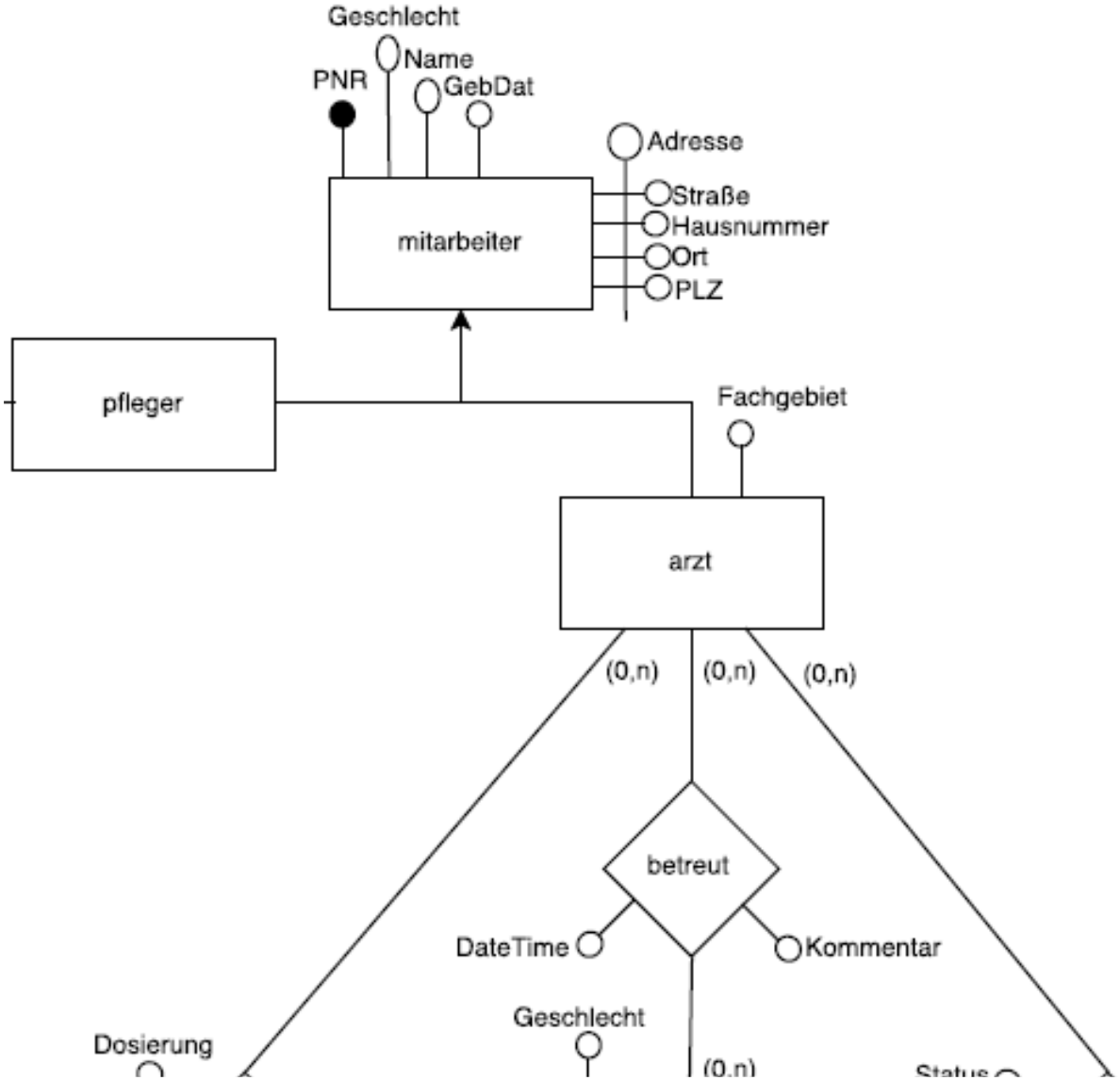
Der *Object-relational Impedance Mismatch* (kurz: IM) bezeichnet die Unverträglichkeit zwischen dem relationalen Datenmodell und dem objektorientierten Programmierparadigma.

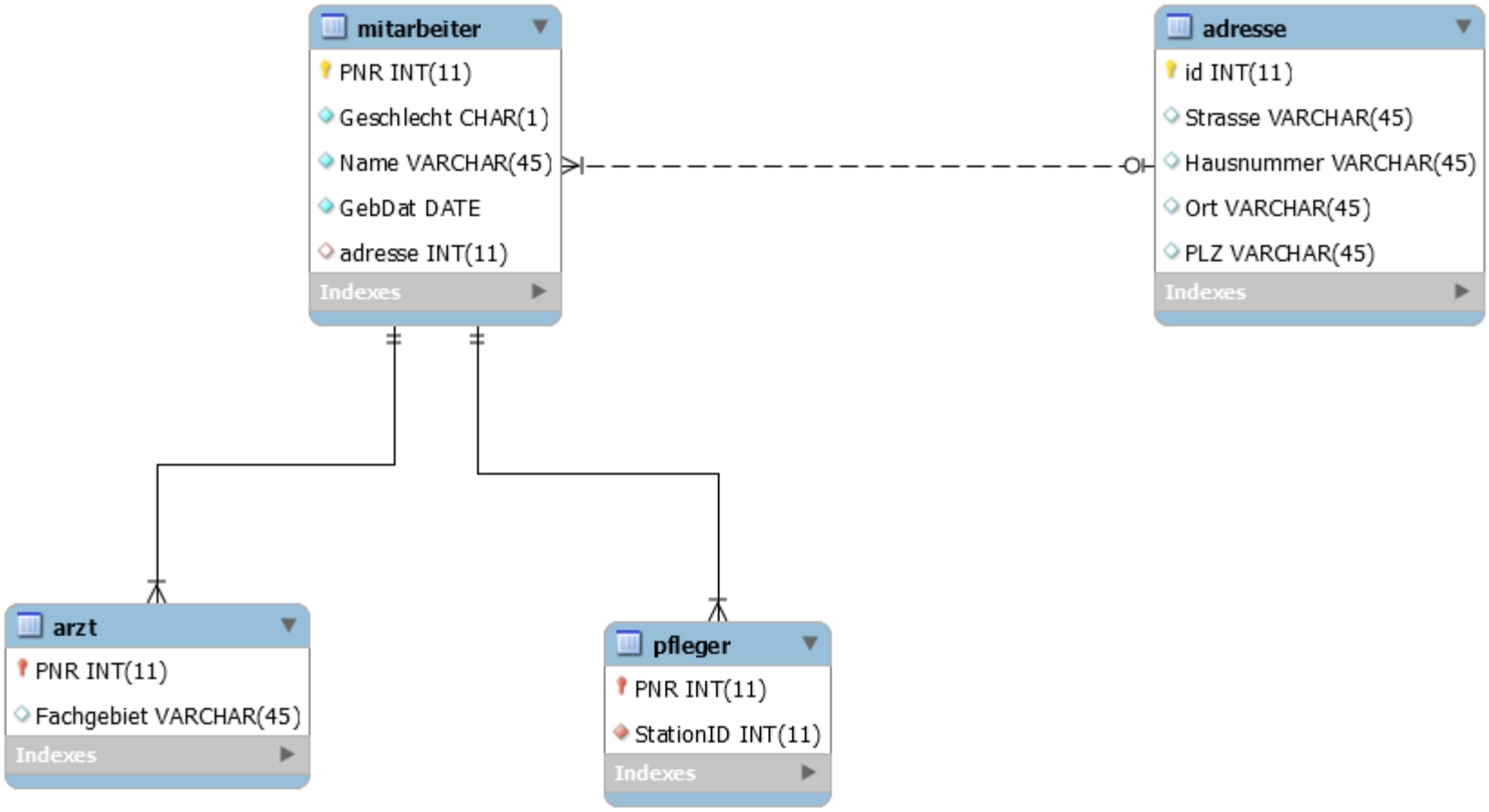
Probleme

	Rel. Modell	OO
Identität	Primary Key / Daten	Object Identifier (intern)
Generalisierung	Wird nicht direkt unterstützt!	Vererbung enthalten
Datenkapselung	Wird nicht direkt unterstützt!	Zugriff nur über Methoden
Beziehungen zwischen Objekten	Teilweise durch Foreign Keys	über Methoden, Attribute
Interaktionen zwischen Objekten	Wird nicht direkt unterstützt!	über Methoden

Trigger und Stored Procedures können dies teilweise simulieren!

… weiterführende Literatur: http://www.cs.utexas.edu/~wcook/Drafts/2005/PLDBProblem.pdf


Geschlecht
Name
PNR
GebDat
Adresse
Straße
Hausnummer
Ort
PLZ
mitarbeiter
pfleger
Fachgebiet
arzt
(0,n)
(0,n)
(0,n)
betreut
DateTime
Kommentar
Geschlecht
Dosierung
(0,n)


mitarbeiter
PNR INT(11)
Geschlecht CHAR(1)
Name VARCHAR(45)
GebDat DATE
adresse INT(11)
Indexes
adresse
id INT(11)
Strasse VARCHAR(45)
Hausnummer VARCHAR(45)
Ort VARCHAR(45)
PLZ VARCHAR(45)
Indexes
arzt
PNR INT(11)
Fachgebiet VARCHAR(45)
Indexes
pfleger
PNR INT(11)
StationID INT(11)
Indexes

Java - Serializieren

- *Serializieren* := Speichern eines Objektes auf einen Festspeicher. Man spricht auch von *deflating* oder *marshalling*.
- *Deserializieren* (*inflating* oder *unmarshalling*) := … wieder Laden

- Serializieren via Java: Alle relevanten Objekte müssen das Interface “java.io.Serializable” implementieren.

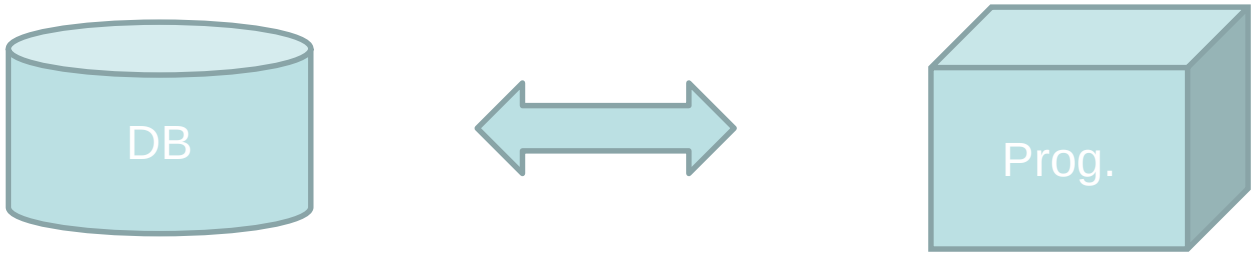
Unmarshalling, Hydrating or Object Retriving

- … bezeichnet den Vorgang aus einem Abfrageergebnis konkrete Objekte zu erzeugen.

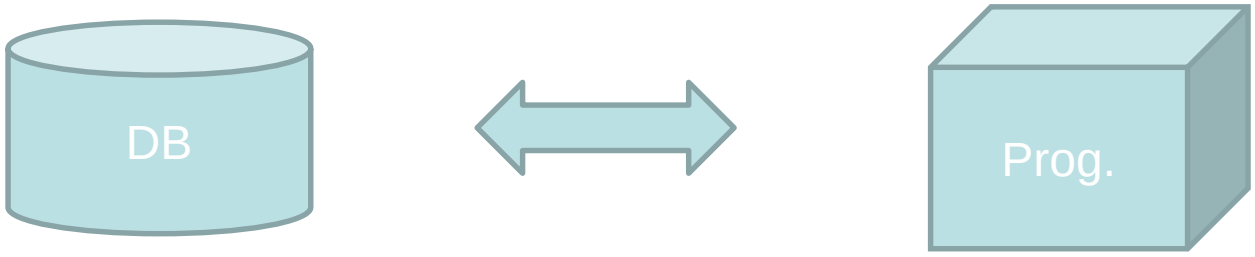
```
select * from Mitarbeiter where PNR > 12;
```

Hilfe …?

- … wie kann man automatisiert
  - Zustände von Objekten zur Datenbank synchronisieren?
  - Abbildungen und Object Retrival handhaben?
  - das ganze effizent gestalten?

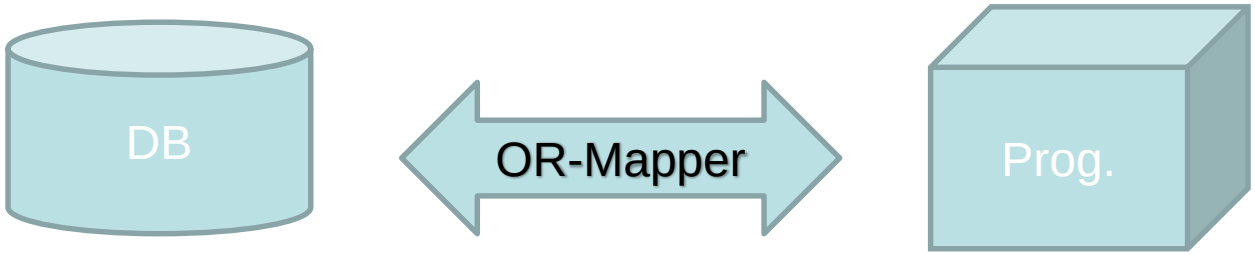


… anderer DBMS Ansatz …



- Probleme würden hinfällig, wenn man eine OO-Datenbank nutzen würde☺
- Viele DBMS bieten OO-Features an, man spricht auch von ORDBMS (Objektrelationales Datenbank-managementsystem) z.B. Oracle, IBM DB2 …

OR-Mapper

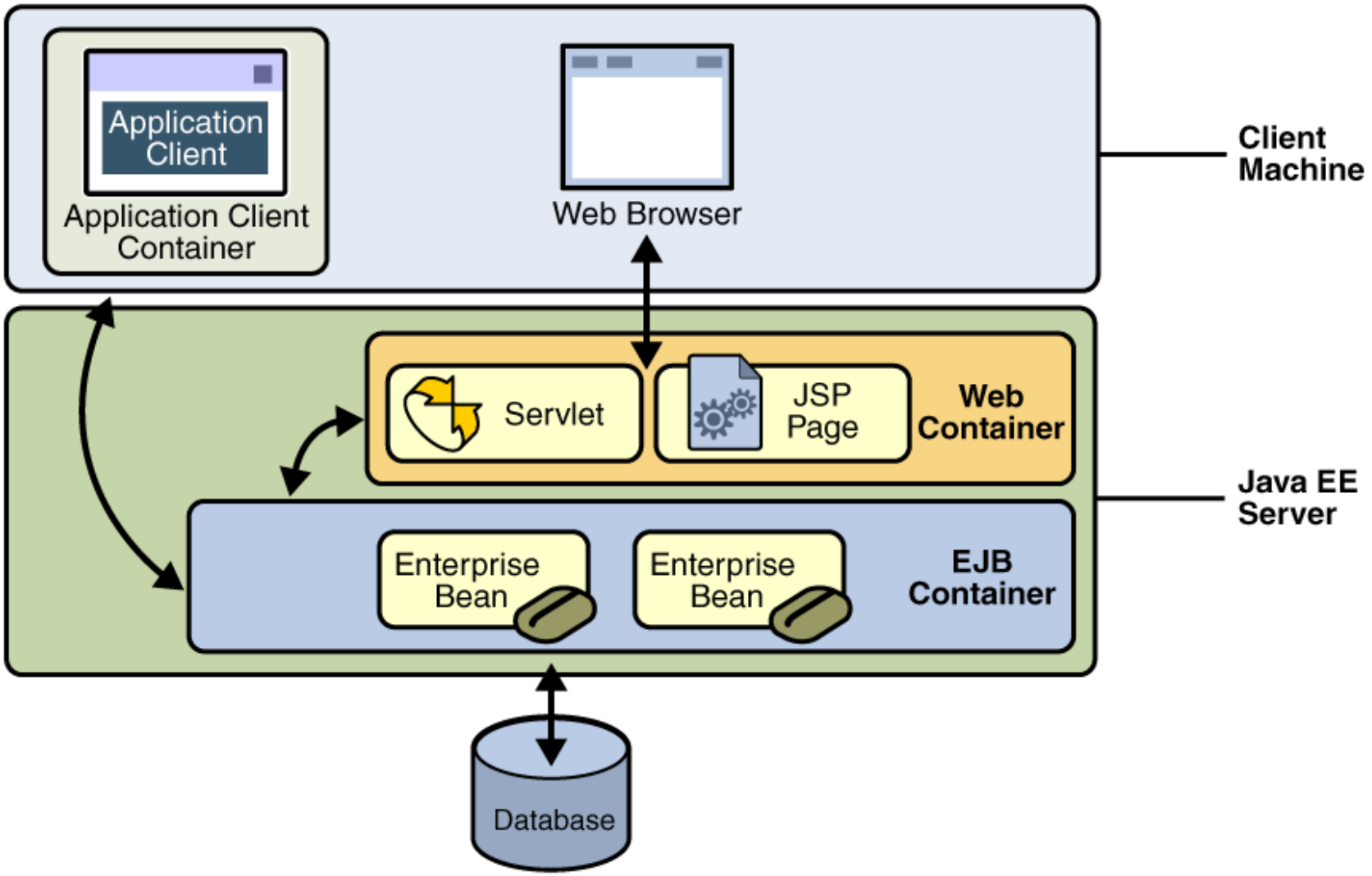


Framework	Beschreibung
Java Persistence API (JPA)	Ist eine Schnittstelle für die objektrelationale Abbildung von POJOs (Plain old Java Object). Wurde im Rahmen der **EJB 3.0** von der Software Expert Group als Teil der JSR 220 entwickelt und herausgegeben. Sie soll die besten Ideen der APIs von Hibernate, Toplink und JDO beinhalten.
Hibernate (Nhibernate für .NET)	Open-Source Framework für Java entwickelt von der JBoss Community. Setzt auch die JPA um (Hibernate EntityManager). Hibernate wird von den meisten als die Referenz für die Lösung des IM im Open-Source-Bereich für Java gesehen.
…	

EJB – Enterprise Java Bean

Standardisierte Komponente innerhalb JEE (Java Platform Enterprise Edition)

EJB – Enterprise Java Bean



- Praxis / Live …
  … mit NetBeans Version 8 (mit Glassfish-Server)

NetBeans Tutorials and Links

- Übersicht JEE:
  https://netbeans.org/kb/trails/java-ee.html

- E-Commerce Tutorial (Web-Shop):
  https://netbeans.org/kb/docs/javaee/ecommerce/intro.html

History zu Netbeans: https://de.wikipedia.org/wiki/NetBeans_IDE