Kapitel 8: Web-Anwendungen mit SQL und PHP

8.1 Embedded SQL (ESQL)

8.2 Dynamisches ESQL

8.3 JDBC

8.4 Web-Anwendungen mit Java Servlets

Kopplung von Programmier- und DB-Sprachen

1) Erweiterung der Programmiersprache um DB-Sprachkonstrukte
   → Persistente Programmiersprachen (z.B. DBPL)
2) Erweiterung der DB-Sprache um Programmierkonstrukte
   → 4GLs (z.B. PL/SQL(
3) Einbettung der DB-Sprache in die Programmiersprache
   → Embedded SQL (ESQL) für „alle“ Programmiersprachen

Beispiel für 1:

```
TYPE    Produktrecordtyp = RECORD
                              PNR: INTEGER; ...
                           END;
        Produkterelationentyp = RELATION OF Produktrecordtyp;
VAR     Produkte: PERSISTENT Produkterelationentyp;
        Ergebnis: Produkterelationentyp;
        x: Produktrecordtyp;

Ergebnis :=        SELECT * FROM Produkte WHERE ...;
FOR EACH x IN Ergebnis  DO ... END;
```

8.1 Embedded SQL (ESQL)

Kernproblem:

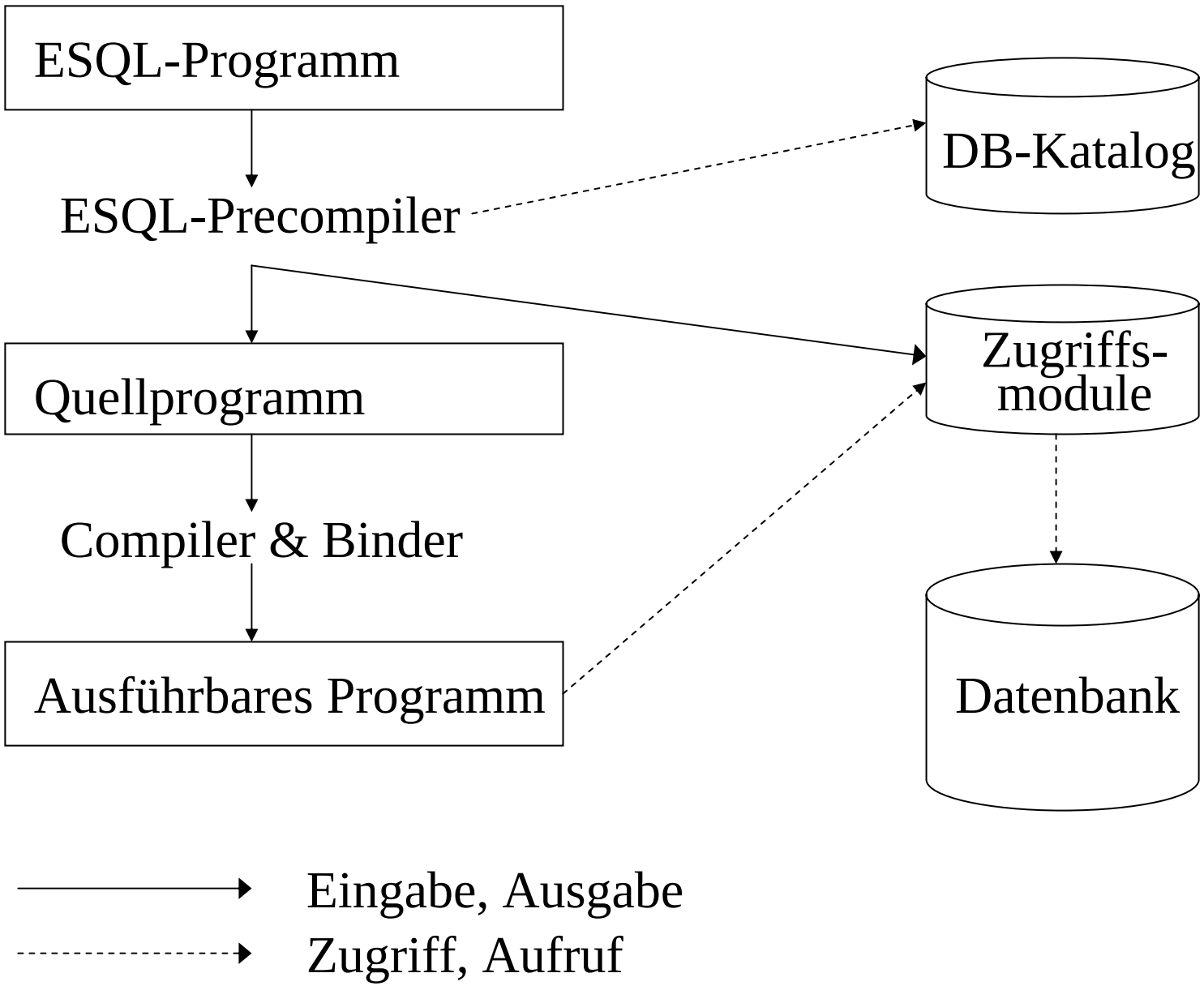
Abbildung von Tupelmengen auf
Datentypen der Wirtsprogrammiersprache

Lösungsansatz:

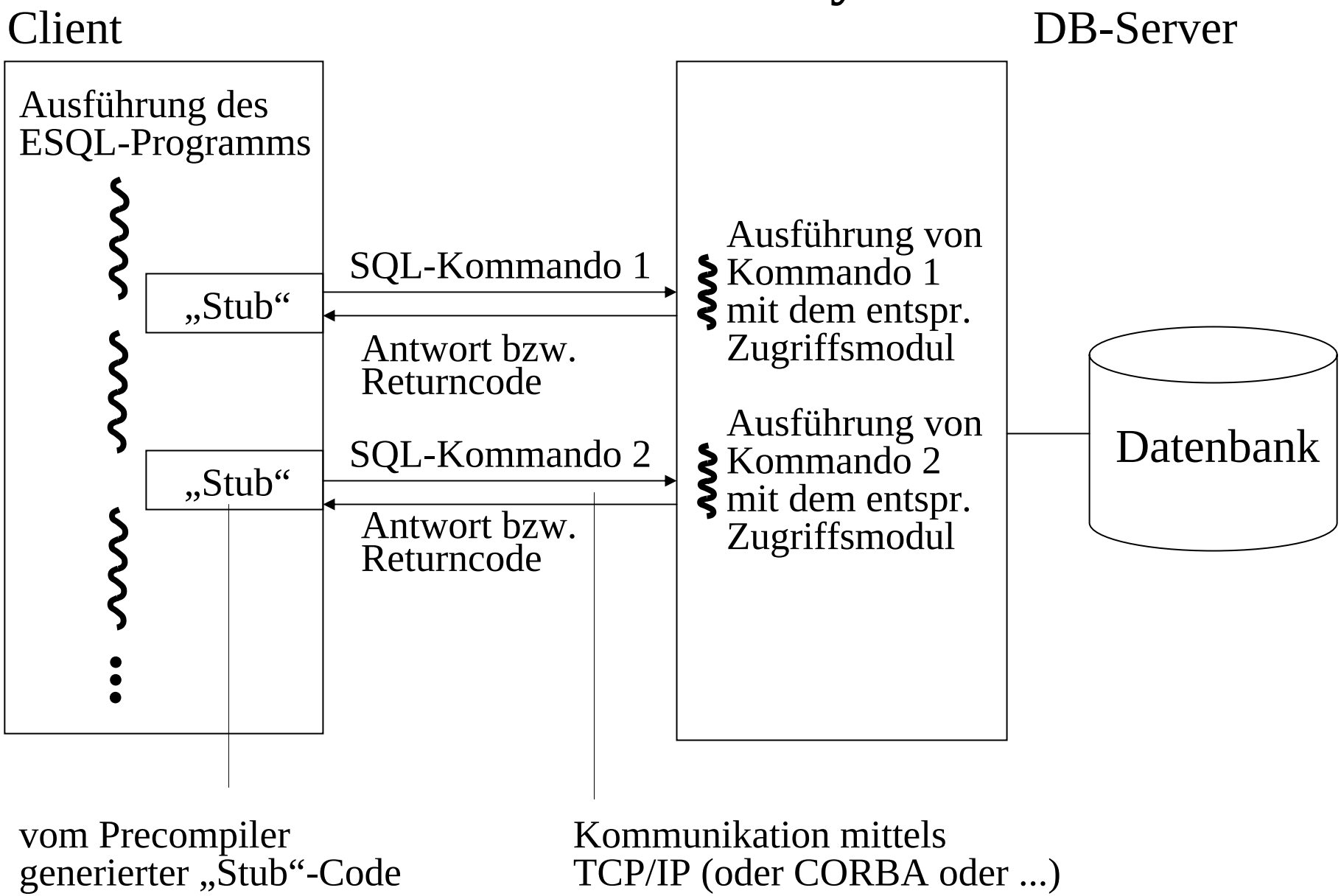
1) Abbildung von Tupeln bzw. Attributen auf die
Datentypen der Wirtsprogrammiersprache
$\rightarrow$ ***Wirtsprogrammvariablen (Host Variables)***
2) Iteratoren (Schleifen) zur Verarbeitung von Tupelmengen
$\rightarrow$ ***Cursor-Konzept***

ein universeller Lösungsansatz
für alle Wirtsprogrammiersprachen !

Architektur des ESQL-Precompilers



Laufzeitarchitektur für ESQL-Programme in Client-Server-System



Wirtsprogrammvariable

- Ergänzung von SQL-Anfragen um INTO-Klausel
  für die Angabe (speziell „markierter“) Wirtsprogrammvariable
  als Übergabebereich für Anfrageresultate

  Beispiel: EXEC SQL SELECT PNr, Menge, Status INTO :pnr, :menge, :status
  FROM Bestellungen WHERE BestNr = 555;

- Verwendung von Wirtsprogrammvariablen als Eingabeparameter
  (an Stelle von Konstanten) in SQL-Anweisungen

  Beispiel: EXEC SQL SELECT PNr, Menge, Status INTO :pnr, :menge, :status
  FROM Bestellungen WHERE BestNr = :bestnr;

- Verwendung zusätzlicher Indikatorvariablen zur Nullwertdiagnose

  Beispiel:
```
EXEC SQL BEGIN DECLARE SECTION;
    int pnr; int vorrat; short vorrat_ind;
EXEC SQL END DECLARE SECTION;
...
EXEC SQL SELECT Vorrat INTO :vorrat:vorrat_ind
        FROM Produkte WHERE PNr = :pnr;
if (vorrat_ind == 0) { /* kein Nullwert */ } else { /* Nullwert */ };
```

Beispiel eines ESQL-C-Programms (1)

```
#include <stdio.h>
#include <string.h>
EXEC SQL INCLUDE SQLCA; /* Importieren der SQL Communication Area */
/* Der Precompiler erzeugt an dieser Stelle die folgende Datenstruktur:
        struct sqlca {
        char  sqlcaid[8];
        long  sqlabc;
        long  sqlcode;
        struct { unsigned short  sqlerrml;  char sqlerrmc[70]; } sqlerrm;
        ...          }; struct sqlca sqlca;  */
main () {
EXEC SQL BEGIN DECLARE SECTION; /* Wirtsprogrammvariablen */
        int bestnr;
        int pnr;
        int menge;
        VARCHAR status[10];
        /* struct { unsigned short len; unsigned char arr[10]; } status; */
        VARCHAR user[20];
        VARCHAR passwd[10];
EXEC SQL END DECLARE SECTION;
EXEC SQL WHENEVER SQLERROR STOP; /* Globale Fehlerbehandlung */
```

Beispiel eines ESQL-C-Programms (2)

```
printf ("Benutzername?"); scanf ("%s", &(user.arr)); user.len = strlen(user.arr);
printf ("Kennwort?"); scanf ("%s", &(passwd.arr)); passwd.len = strlen(passwd.arr);
EXEC SQL CONNECT :user IDENTIFIED BY :passwd; /* Beginn Transaktion */
printf („Bestellnummer? (0 = Programmende)\n"); scanf ("%d", &bestnr);
while (bestnr != 0) {
    EXEC SQL SELECT PNr, Menge, Status INTO :pnr, :menge, :status
                    FROM Bestellungen WHERE BestNr = :bestnr;
    if (sqlca.sqlcode = = 0) /* Testen des SQL-Returncodes */ {
        status.arr[status.len] = '\0'; /* Konvertierung von VARCHAR in C-String */
        if (strcmp (status.arr, "neu") = = 0) {
            EXEC SQL UPDATE Produkte SET Vorrat = Vorrat - :menge
                            WHERE PNr = :pnr AND Vorrat >= :menge;
            if (sqlca.sqlcode = = 0) {
                strcpy (status.arr, "geliefert"); status.len = strlen (status.arr);
                EXEC SQL UPDATE Bestellungen SET Status = :status
                                WHERE BestNr = :bestnr; }
            else printf ("\n *** Ungenuegender Lagervorrat ***\n"); }
        else printf ("\n *** Lieferung bereits erfolgt ***\n"); }
    else printf ("\n *** Bestellung nicht gefunden ***\n");
    EXEC SQL COMMIT WORK;  /* Ende Transaktion und Beginn neue Trans. */
    printf ("Bestellnummer? (0 = Programmende)\n"); scanf ("%d", &bestnr);
}; /*while*/  } /*main*/
```

Fehlerbehandlung in ESQL

1) **Explizites Testen vo sqlca.sqlcode**
   (0: korrekt; < 0: Fehler; > 0: Warnung/Ausnahme)
   und ggf. Zugriff auf sqlca.sqlerrd[2], sqlca.sqlerrm.sqlerrmc, usw.

2) **Deklaration von „Exception Handling“-Maßnahmen:**
   EXEC SQL WHENEVER {SQLERROR | NOT FOUND | SQLWARNING}
   {STOP | GOTO label | CONTINUE}

Potentielle Falle: Wirkungsbereich ist statisch!

```
• void f (...) ...
     { ...  EXEC SQL UPDATE ...; ... };
  void g (..) ...
     { ...  EXEC SQL WHENEVER SQLERROR GOTO handle_error; ...
      f (...); ...
     };
```

```
• EXEC SQL WHENEVER SQLERROR GOTO handle_error;
  EXEC SQL CREATE TABLE Mytable ( ... );
   ...
  handle_error:
     EXEC SQL DROP TABLE Mytable; exit (1);
```

Cursor-Konzept für Iteratoren über Tupelmengen

```
#define TRUE 1
#define FALSE 0
printf ("Kundennummer? (0 = Programmende)\n");  scanf ("%d", &knr);
EXEC SQL DECLARE Kundeniterator CURSOR FOR
        SELECT Monat, Tag, Bez, Menge FROM Bestellungen
        WHERE KNr = :knr
        ORDER BY Monat DESC, Tag DESC;
...
EXEC SQL OPEN Kundeniterator;
found = TRUE;
while (found)  {
    EXEC SQL FETCH Kundeniterator INTO :monat, :tag, :bez, :menge;
    bez.arr[bez.len] = '\0';
    if ((sqlca.sqlcode >= 0) && (sqlca.sqlcode != 1403))
        printf ("%d.%d.: %d %s", tag, monat, menge, bez);
    else found = FALSE;
}; /*while*/
EXEC SQL CLOSE Kundeniterator;
```

8.2 Dynamisches ESQL

Motivation: Schreibe Programm für Suche
SELECT * FROM * WHERE * LIKE ‘%SQL%‘

Ansatz:
Durchsuche DB-Katalog und
generiere SQL-Queries auf allen Tabellen

Problem:
Anfragen entstehen erst zur Laufzeit des Programms und können nicht vom ESQL-Precompiler behandelt werden

Lösung:
Dynamisches ESQL
(für alle Anwendungen, bei denen SQL-Statements erst zur Laufzeit bekannt sind, z.B. für sqlplus u.ä.);
Spezialfall: JDBC (dynamisches ESQL für Java)

Ausschnitt aus DB-Katalog von Oracle

SYSCATALOG bzw. SYS.ALL_TABLES UNION SYS.ALL_VIEWS

OWNER	TABLE_NAME	TABLE_TYPE
DBS	BESTELLUNGEN	TABLE
DBS	KUNDEN	TABLE
DBS	PRODUKTE	TABLE
SYS	ALL_TABLES	VIEW

SYS.ALL_TAB_COLUMNS

OWNER	TABLE_NAME	COLUMN_ NAME	DATA_ TYPE	DATA_ LENGTH	...	COLUMN_ ID	...
DBS	KUNDEN	KNR	NUMBER	22		1	
DBS	KUNDEN	NAME	CHAR	30		2	
DBS	KUNDEN	STADT	CHAR	30		3	
DBS	KUNDEN	SALDO	NUMBER	22		4	
DBS	KUNDEN	RABATT	NUMBER	22		5	
DBS	PRODUKTE	PNR	NUMBER	22		1	
SYS	ALL_TABLES	OWNER	CHAR	30		1	
SYS	ALL_TABLES	TABLE_NAME	CHAR	30		2	

Programmieren mit dynamischem ESQL

Einfacher Fall: Resultatschema und Queryparameter statisch festgelegt

1) Deklaration der Wirtsprogrammvariablen
2) Aufbau der SQL-Anweisung als Zeichenkette
3) Dynamische Precompilation: PREPARE DynQuery FROM :sqltext
4) DECLARE Iterator CURSOR FOR DynQuery
5) Auswertung der Eingabeparameter und Vorbereitung der "Cursor"-Schleife: OPEN Iterator
6) FETCH Iterator INTO Wirtsprogrammvariablen
7) Resultatsattribute in den Wirtsprogrammvariablen verarbeiten
8) Wiederholung der Schritte 6 und 7 für jedes Resultatstupel
9) CLOSE Iterator

Sonderfall: höchstens ein Resultatstupel

statt 3) bis 9) einfacher: EXECUTE IMMEDIATE :sqltext

Komplexer Fall:
Resultatschema oder Queryparameter dynamisch festgelegt

Programmbeispiel mit dynamischem ESQL (1)

```
#include <stdio.h>
#include <string.h>
#define TRUE 1
#define FALSE 0
#define MAXPRODUKTE 10
EXEC SQL INCLUDE SQLCA;
main () {
EXEC SQL BEGIN DECLARE SECTION;
        int monat, tag, pnr, menge;
        VARCHAR bez[31];
        VARCHAR sqltext[512];
        VARCHAR user[20];
        VARCHAR passwd[10];
EXEC SQL END DECLARE SECTION;
EXEC SQL WHENEVER SQLERROR GOTO handle_error;
typedef char prod_bez_t[31];
typedef int bool;
prod_bez_t      prod_bez[10];
int             i;
int             num_of_prod;
bool            found;
```

Programmbeispiel mit dynamischem ESQL (2)

```
printf ("Benutzername?"); scanf ("%s", &(user.arr)); user.len = strlen(user.arr);
printf ("Kennwort?"); scanf ("%s", &(passwd.arr)); passwd.len = strlen(passwd.arr);
EXEC SQL CONNECT :user IDENTIFIED BY :passwd;
i = 0;
printf ("%s%s", "Produktbezeichnung oder \"ende\" \n"); scanf ("%s", &(prod_bez[i]));
while ((i < MAXPRODUKTE) && (strcmp (prod_bez[i], "ende") != 0)) {
    i++;
    printf ("%s%s", "Produktbezeichnung oder \"ende\" \n");
    scanf ("%s", &(prod_bez[i])); }; /*while*/
num_of_prod = i;
if (num_of_prod = = 0) exit(-1);
/* Aufbau der SQL-Anfrage */
strcpy (sqltext.arr, "SELECT Monat, Tag, Bestellungen.PNr, Bez, Menge ");
strcat (sqltext.arr, "FROM Bestellungen, Produkte ");
strcat (sqltext.arr, "WHERE Bestellungen.PNr = Produkte.PNr ");
strcat (sqltext.arr, "AND ( "); strcat (sqltext.arr, "Bez LIKE \'");
strcat (sqltext.arr, prod_bez[0]); strcat (sqltext.arr, "%\' ");
for (i=1; i < num_of_prod; i++) {
    strcat (sqltext.arr, "OR Bez LIKE \'"); strcat (sqltext.arr, prod_bez[i]);
    strcat (sqltext.arr, "%\' ");  }; /*for*/
strcat (sqltext.arr, ") "); strcat (sqltext.arr, "ORDER BY Monat DESC, Tag DESC");
sqltext.len = strlen(sqltext.arr);
```

Programmbeispiel mit dynamischem ESQL (3)

```
EXEC SQL PREPARE DynQuery FROM :sqltext;
EXEC SQL DECLARE BestIterator CURSOR FOR DynQuery;
EXEC SQL OPEN BestIterator;
found = TRUE;
while (found) {
    EXEC SQL FETCH BestIterator INTO :monat, :tag, :pnr, :bez, :menge;
    if (sqlca.sqlcode = = 0) {
        bez.arr[bez.len] = '\0';
        printf ("%d.%d.: %d Stueck %s (PNr %d)\n", tag, monat, menge, bez.arr, pnr); }
    else
        found = FALSE;
    }; /*while*/
EXEC SQL CLOSE BestIterator;
EXEC SQL COMMIT WORK RELEASE;
exit (0);
/* Fehlerbehandlung */
handle_error:
    printf ("\n*** Error %d in program. ***\n", sqlca.sqlcode);
    printf ("%.70s\n", sqlca.sqlerrm.sqlerrmc);
    exit (-1);
} /*main*/
```

Stored Procedures in 4GL PL/SQL

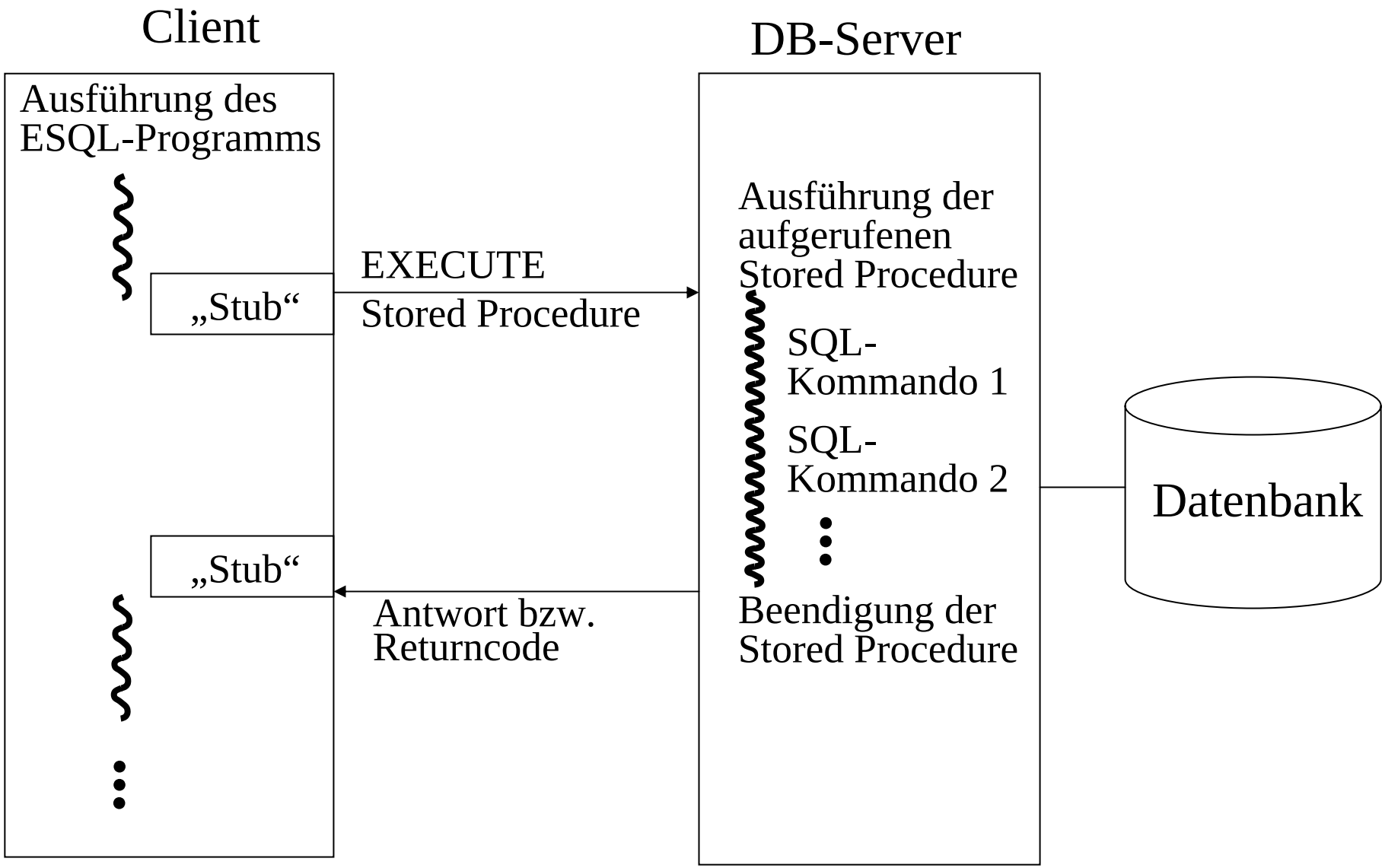
```
CREATE PROCEDURE Lager_Auffuellen AS
DECLARE ...;
BEGIN
DECLARE CURSOR Knappe_Produkte IS
    SELECT PNr, Vorrat FROM Produkte WHERE Vorrat < 100;
KP Knappe_Produkte%ROWTYPE;
BEGIN
    OPEN Knappe_Produkte;
    LOOP
        FETCH Knappe_Produkte INTO KP;
        EXIT WHEN Knappe_Produkte%NOTFOUND;
        ...
        IF KP.Vorrat > 10 THEN
            EXECUTE Nachbestellen (KP.PNr, 1000 - KP.Vorrat)
        ELSE
            EXECUTE Express_Bestellen (KP.PNr, 10);
            EXECUTE Nachbestellen (KP.PNr, 1000 - KP.Vorrat - 10);
        END IF;
    END LOOP;
END; END;

EXECUTE Lager_Auffuellen;
```

Prozedur-deklaration

Prozedur-aufruf

Laufzeitarchitektur für Stored Procedures



8.3 JDBC (Java Database Connectivity)

```
import java.io.*; import java.util.*; import java.sql.*;
public class myjdbc {
public static void main (String args[]) {

Connection con = null; Statement stmt = null;
String sqlstring;

String driver = "oracle.jdbc.driver.OracleDriver";
try { Class.forName(driver);}
   catch(Exception e) {
   System.out.println ("error when loading jdbc driver");};

try {
String jdbcURL = "jdbc:oracle:thin:";
String db = "(DESCRIPTION =(ADDRESS_LIST = (ADDRESS = " +
   "(PROTOCOL = TCP)(HOST = TOKYO)(PORT = 1521)))" +
   "(CONNECT_DATA =(SERVICE_NAME = TOKYO.WORLD)))";
String user = "lehre40"; String password = "leere4zig";
con = DriverManager.getConnection
      (jdbcURL + "@" + db, user, password);
con.setAutoCommit (false);
stmt = con.createStatement(); }
   catch (Exception e) {
   System.out.println ("error with jdbc connection"); };
```

JDBC-Beispiel (2)

```
String inputline = null;
try{
System.out.println ("\n Please type name. \n");
DataInputStream myinput = new DataInputStream (System.in);
inputline = myinput.readLine(); }
catch (IOException e) {System.out.println ("IO error");};
```

```
try{
sqlstring = "SELECT * FROM CUSTOMER " +
            "WHERE NAME LIKE '" + inputline + "'";
ResultSet result = stmt.executeQuery (sqlstring);
System.out.println ("CUSTOMER:");
System.out.println ("=========");
while (result.next()) {
   String namevar = result.getString("NAME");
   String ipnumbervar = result.getString("IPNUMBER");
   String cityvar = result.getString("CITY");
   System.out.println ("NAME: " + namevar + " IPNUMBER: "
                       + ipnumbervar + " CITY: " + cityvar);
}; //while
con.commit(); }
   catch (SQLException sqlex)
   {System.out.println (sqlex.getMessage());};
System.out.println ("\n");
```

JDBC-Beispiel (3)

```
try {
sqlstring = "SELECT * FROM ACCOUNT " +
            "WHERE NAME LIKE '" + inputline +"'";
ResultSet result = stmt.executeQuery (sqlstring);

System.out.println ("ACCOUNT:");
System.out.println ("========");
while (result.next()) {
   String namevar = result.getString("NAME");
   int balancevar = result.getInt("BALANCE");
   System.out.println ("NAME: " + namevar +
                       " BALANCE: " + balancevar);
}; //while
con.commit(); }
   catch (SQLException sqlex)
   {System.out.println (sqlex.getMessage());};

try{ con.close(); }
   catch (SQLException sqlex)
   {System.out.println (sqlex.getMessage());};

}; //main
} //myjdbc
```

8.4 Web-Anwendungen mit Servlets

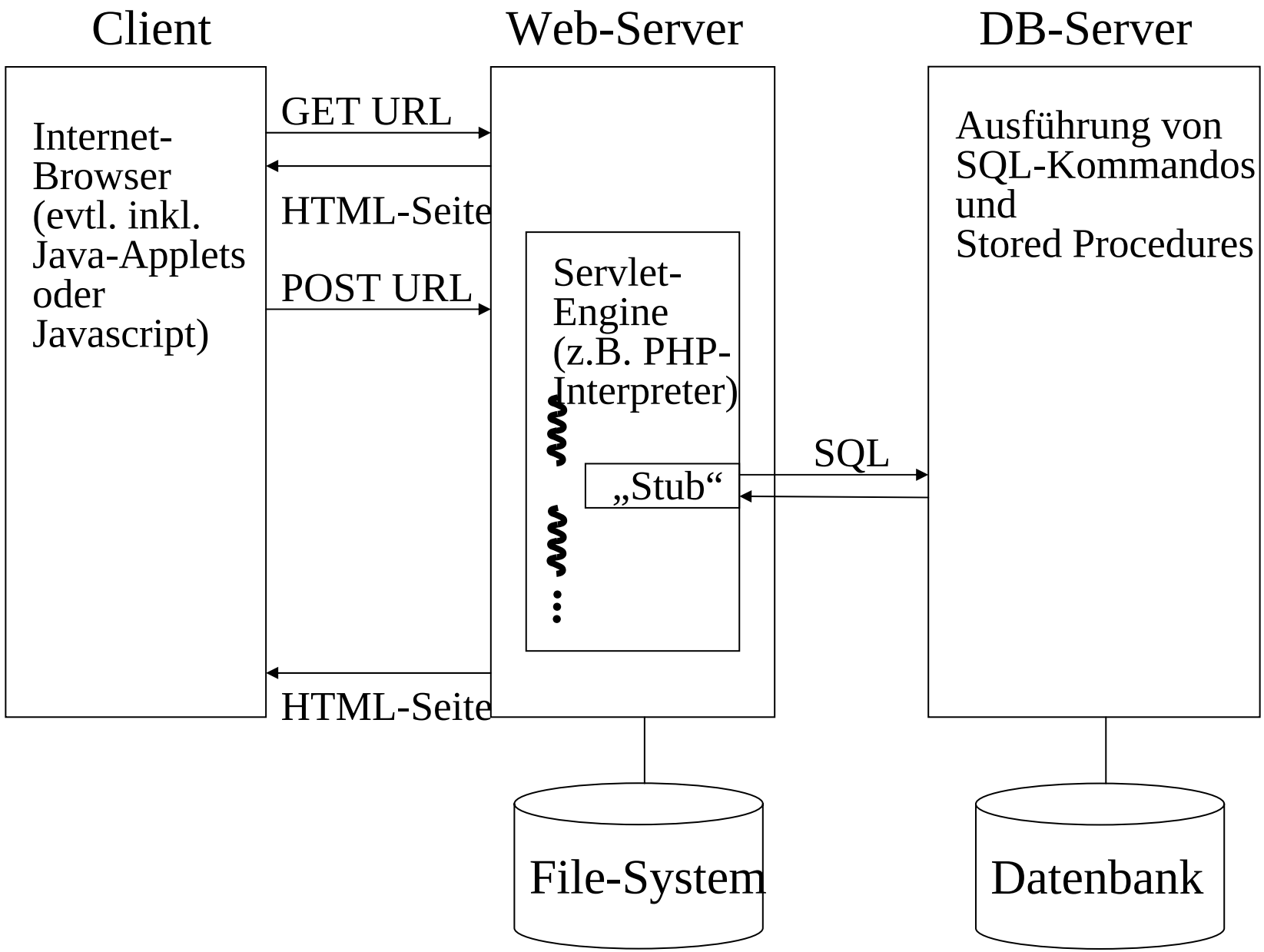
3-Schichten-Architektur (3-Tier Architecture):

- Client (Front-End):
  Präsentation / GUI mittels Internet-Browser
  (plus ggf. Javascript oder Java Applets)
- Applikations-Server (Middle Tier):
  Applikationslogik gesteuert durch Servlet-Engine
  (z.B. Apache/Tomcat und JVM mit JSDK,
  Apache plus PHP-Interpreter)
  mit Datenbankzugriffen via JDBC bzw. ODBC
- Daten(bank)-Server (Back-End)

Vorteile gegenüber Applikationslogik im Client (z.B. Applets):

- einfacheres Management (Installation, Softwarepflege)
- höhere Sicherheit (Applikationscode unsichtbar für Client)
- bessere Effizienz und Skalierbarkeit (z.B. weniger DB-Sessions)

Laufzeitarchitektur für Servlets



Servlets in Java

anhand eines typischen Beispiels einer Web-Anwendung

basierend auf folgender DB:

```
create table customer (name varchar(20) primary key,
                       ipnumber varchar(20) not null, city varchar(20));
create table account (name varchar(20) primary key, balance integer);
create table history (name varchar(20),
                       bookingdate date, amount integer);
```

mit der folgenden Applikationslogik:

```
look up customer in database
check if customer has filled form
if (form) parameters available
then
   insert customer info into database
   send personalized greeting
else send blank form
create output with links to „transfer“ and „audit“ servlets
```

Servlet-Beispiel (1)

```
import java.lang.*; import java.io.*; import java.util.*;
import javax.servlet.*; import javax.servlet.http.*;
import java.sql.*; import java.net.*;
public class welcome extends HttpServlet {

public void doGet
(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

String namevar; String cityvar;
BufferedReader readervar; String linestring;
String ipnumbervar = null; String browservar = null;
boolean newcustomer; String nexturl;

PrintWriter out = response.getWriter();
out.println("<html><body>");
namevar = request.getParameter ("inputname");
cityvar = request.getParameter ("inputcity");
ipnumbervar = request.getRemoteAddr();
// create JDBC connection
// ...
```

Servlet-Beispiel (2)

```
// actual application code
newcustomer = true;
try{
sqlstring = "SELECT * FROM CUSTOMER " +
            "WHERE IPNUMBER='" + ipnumbervar + "'";
ResultSet result = stmt.executeQuery (sqlstring);
if (result.next()) {
   newcustomer = false;
   namevar = result.getString("NAME");
   cityvar = result.getString("CITY");
}; //if
}
catch (SQLException sqlex)
   {System.out.println (sqlex.getMessage());};
```

Servlet-Beispiel (3)

```
if ((namevar != "") & (namevar != null)) {
  // insert customer info into db
  if (newcustomer){
    try {
    sqlstring = "INSERT INTO CUSTOMER " +
         "(NAME,IPNUMBER,CITY) VALUES (' " + namevar +
         "','" + ipnumbervar + "','" + cityvar + "')";
    stmt.execute (sqlstring);
    sqlstring = "INSERT INTO ACCOUNT (NAME,BALANCE) " +
         " VALUES (' " + namevar + "',0)";
    stmt.execute (sqlstring); con.commit(); }
    catch (SQLException sqlex)
       {System.out.println (sqlex.getMessage());};
   } } //then
else {
  // post form for inquiring customer info
  out.println("<form method=post action=welcome>");
  out.println("<br> Please type your name and city. <br>");
  out.println("Name: <input type=text name=inputname><br>");
  out.println("City: <input type=text "  +
              "name=inputcity value=Saarbruecken><br>");
  out.println("<input type=submit name=inputenter " +
              "value=\"Here you go!\"><br>");
  out.println("</form>");
}; //if
```

Servlet-Beispiel (4)

```
if ((namevar != "") && (namevar != null)) {
  out.println ("Welcome, " + namevar + "!<br>");
  out.println ("Today's date is ");
  out.println (new java.util.Date());
  out.println ("<br>");
  out.println ("Your IP number is " + ipnumbervar + "<br>");
  out.println ("How is the weather in " +
               cityvar + "?<br><br>");
  out.println ("How can we help you today?<br>");
  nexturl = "transfer?customername=" +
            URLEncoder.encode(namevar);
  out.println ("<a href=" + nexturl +
               "> Deposit or withdraw money</a><br>");
  nexturl = "audit?customername=" +
            URLEncoder.encode(namevar);
  out.println ("<a href=" + nexturl +
               "> Audit trail of your account</a><br>");
}; //if
out.println("</body></html>");
} //doGet
```

Servlet-Beispiel (5)

```
public void doPost
(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

   doGet(request, response);

}  //  doPost

} // class welcome
```

Servlet mit Session-Verwaltung (1)

- Sitzungsdaten über mehrere Dialogschritte hinweg
- Session-Objekt wird von Servlet-Engine verwaltet und ist für Servlet zugreifbar
- Cookie enthält Session-Identifier

```
import java.lang.*; import java.io.*; import java.util.*;
import javax.servlet.*; import javax.servlet.http.*;
public class sessionexample extends HttpServlet {

public void doGet
(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

Integer mycounter;
PrintWriter out = response.getWriter();
out.println("<html><body>");
out.println("<h1>Welcome!</h1><br><br>");
```

Servlet mit Session-Verwaltung (2)

```
// create session if not yet existing,
// otherwise fetch session data
HttpSession session = request.getSession(true);
if (session.isNew()) {
   mycounter = new Integer(1);
   session.putValue("mycounter", mycounter); }
else {
   mycounter = (Integer) session.getValue("mycounter");
   mycounter = new Integer(mycounter.intValue() + 1);
   out.println("You are visiting us the ");
   out.println(mycounter.intValue());
   out.println("th time");
   session.putValue("mycounter", mycounter);
}; //if
out.println("</body></html>");
} //doGet

} // class sessionexample
```