GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Wie kommen die Befehle zum DBMS

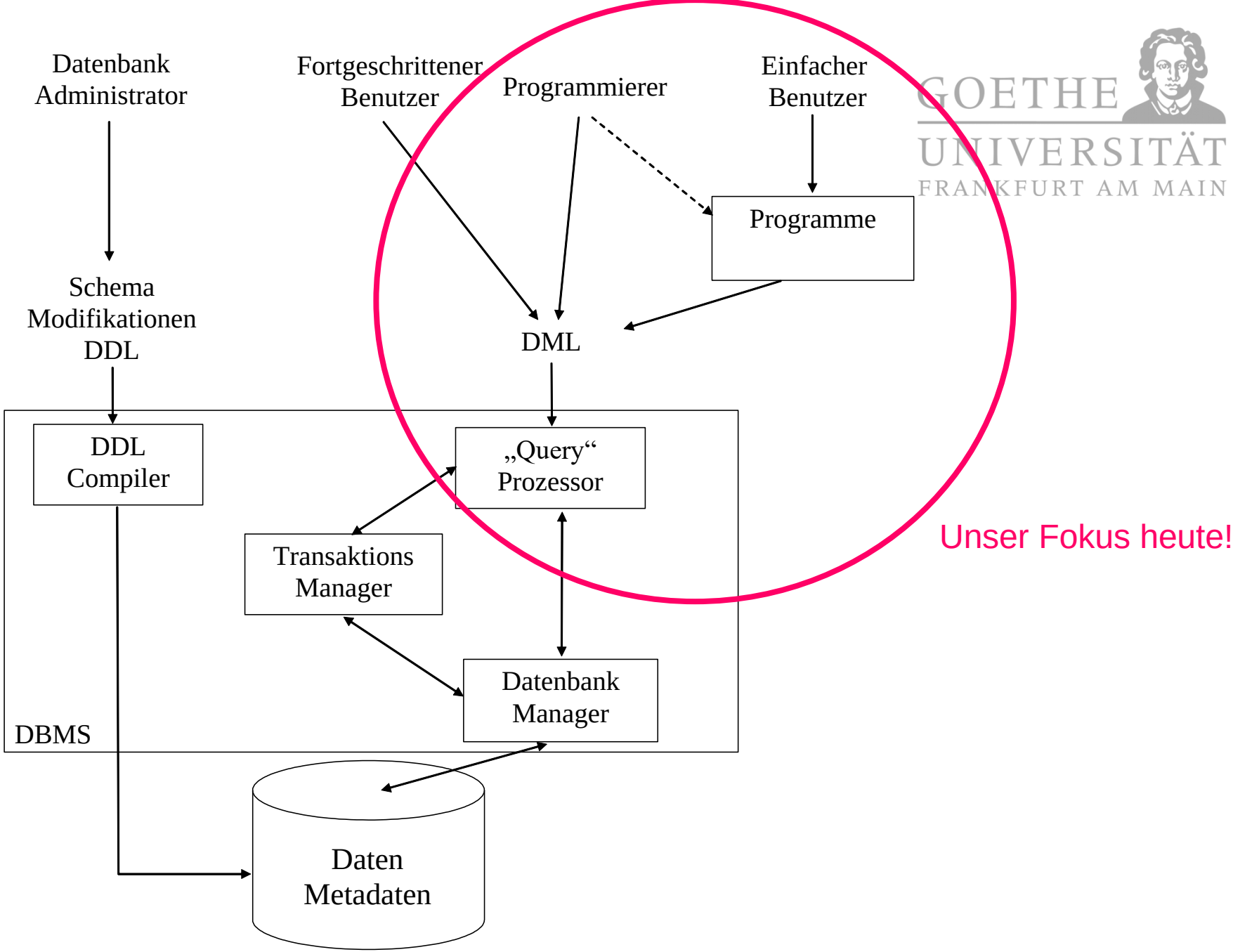
Dr. Karsten Tolle

Datenbanken und Informationssysteme

Wie kommen die Befehle zum DBMS

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

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


Datenbank Administrator
Fortgeschrittener Benutzer
Programmierer
Einfacher Benutzer
Programme
Schema Modifikationen DDL
DML
DDL Compiler
„Query“ Prozessor
Transaktions Manager
Datenbank Manager
DBMS
Daten Metadaten
Unser Fokus heute!

Zeitlinie … ?

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

DBL Project History

early 70's	Ted Codd's first papers on Relational Algebra
1975	CODASYL Database Specifications
1977	database Project Initiated in U.S.
1978	ANSI Database Project Approved
1979	ISO Database Project Initiated
1982	ANSI Project Split into NDL and SQL
1983	ISO Project Split into NDL and SQL
1986	ANSI SQL Published - December
1987	ISO/IEC 9075:1986 (SQL86)
1989	ISO/IEC 9075:1989 (SQL89)
1992	ISO/IEC 9075:1992 (SQL92)
1995	ISO/IEC 9075-3:1995 (SQL/CLI for SQL92)
1996	ISO/IEC 9075-4:1996 (SQL/PSM for SQL92)

embedded SQL

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

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

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

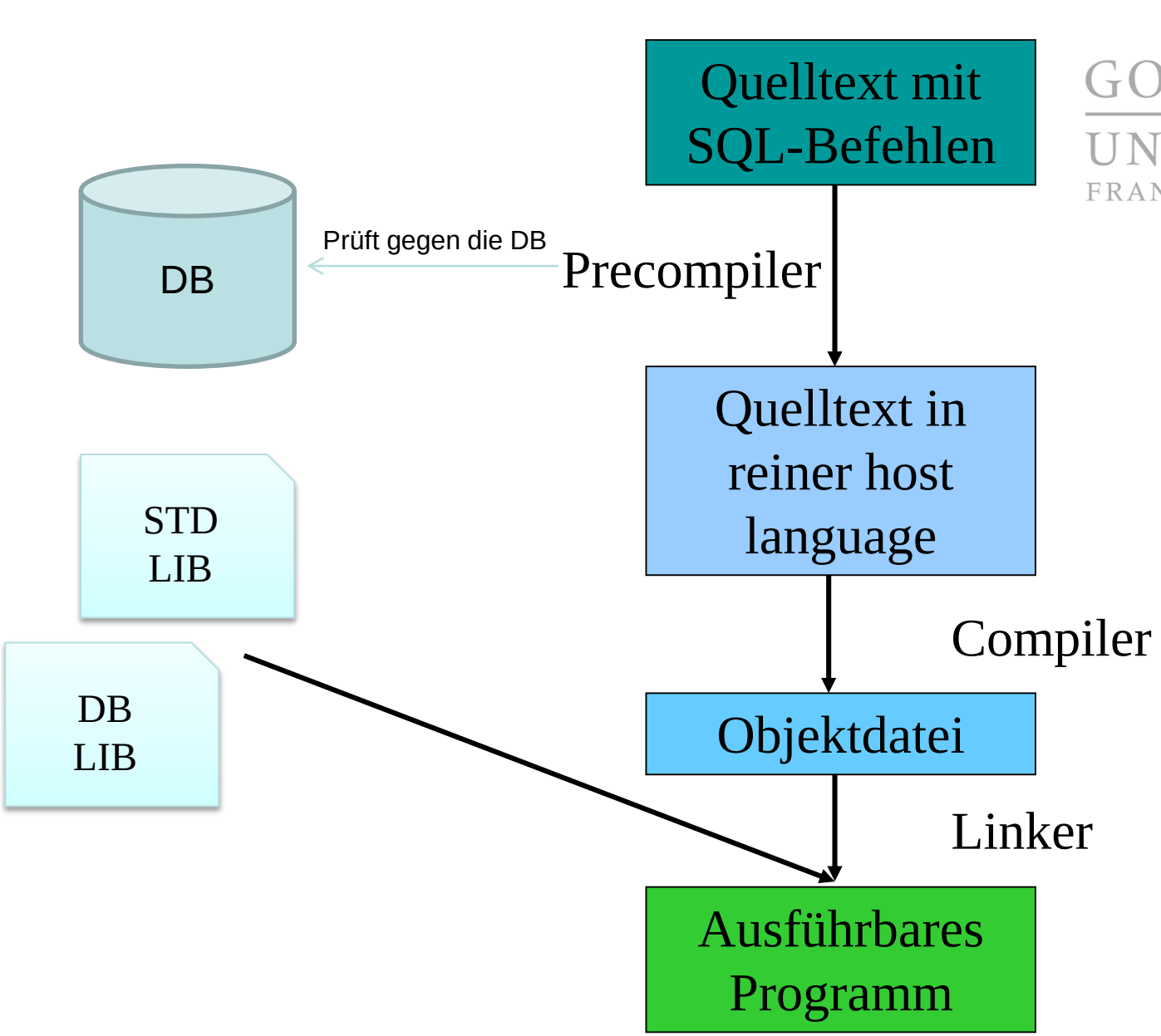
<embedded SQL host program> ::=
<embedded SQL Ada program> |
<embedded SQL C program> |
<embedded SQL COBOL program> |
<embedded SQL Fortran program> |
<embedded SQL MUMPS program> |
<embedded SQL Pascal program> |
<embedded SQL PL/I program>

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

Embedded SQL und C - Beispiel

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

```
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

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

- 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

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

- 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

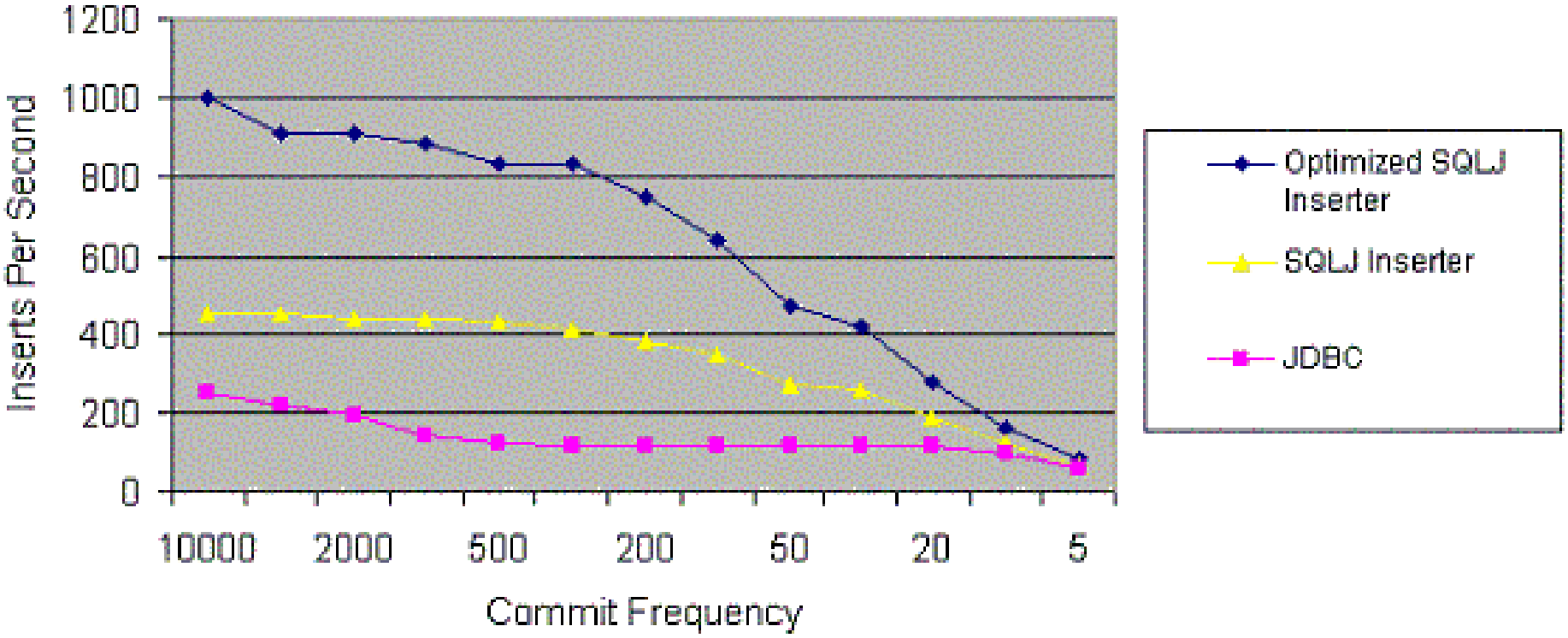
GOETHE UNIVERSITÄT FRANKFURT AM MAIN

```
…
#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

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

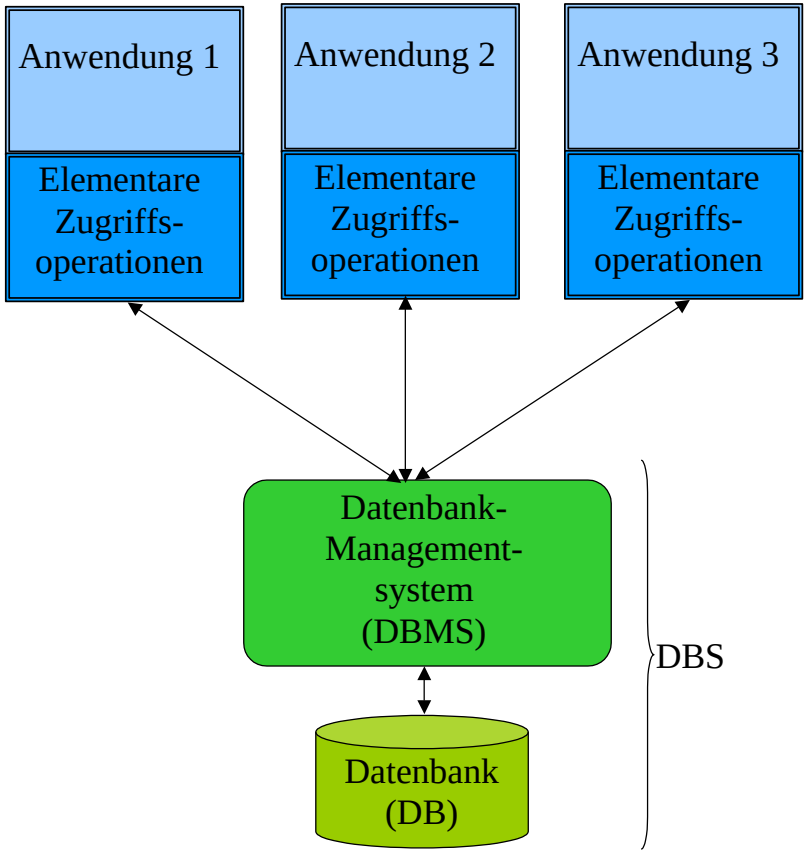


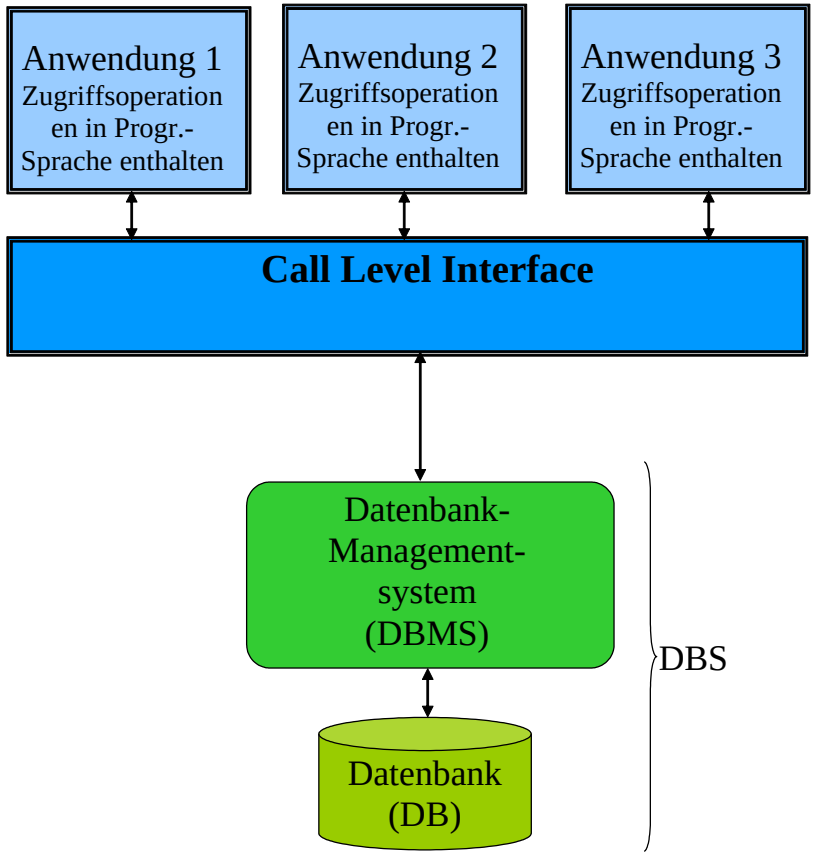
Quelle: IBM

CLI – Call Level Interface

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

- 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://www2.opengroup.org/ogsys/jsp/publications/PublicationDetails.jsp?catalogno=c451
  - PDF des Technical Standards von 1995!
    ~320 Seiten

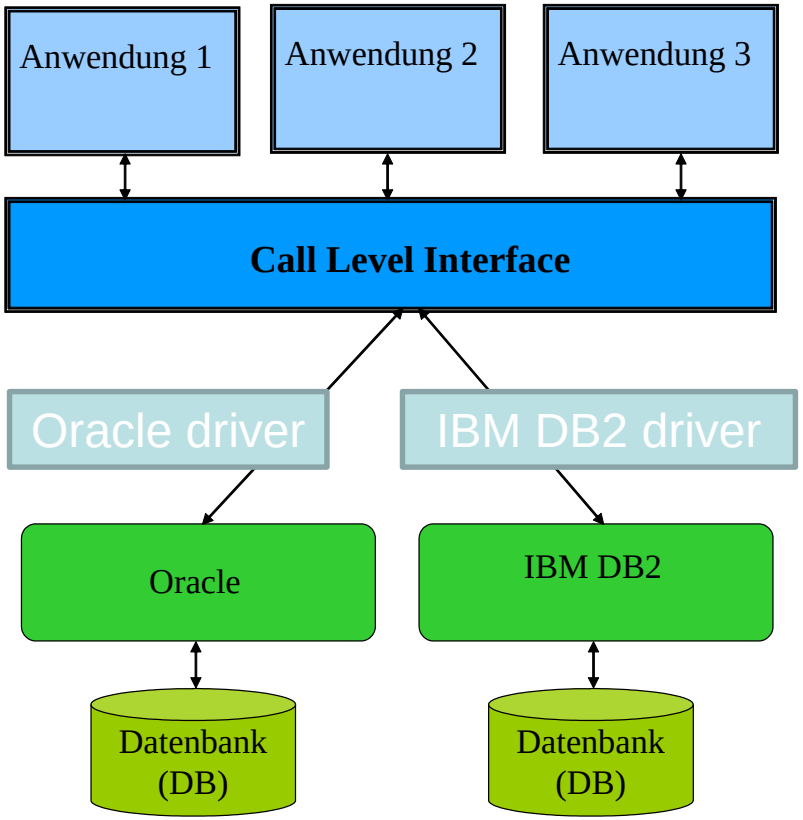

Anwendung 1
Elementare Zugriffs-operationen
Anwendung 2
Elementare Zugriffs-operationen
Anwendung 3
Elementare Zugriffs-operationen
Datenbank-Management-system (DBMS)
Datenbank (DB)
DBS


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

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

- 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

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

- 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

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

- 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 (von 2006 – 4.1 in 2011)
  - 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:

http://www.oracle.com/technetwork/java/javase/jdbc/index.html

Java Praxis - DriverManager

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

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, 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

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

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

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

```
// 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://w3processing.com/index.php?subMenuId=134

Code auf der DB-Server Seite

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

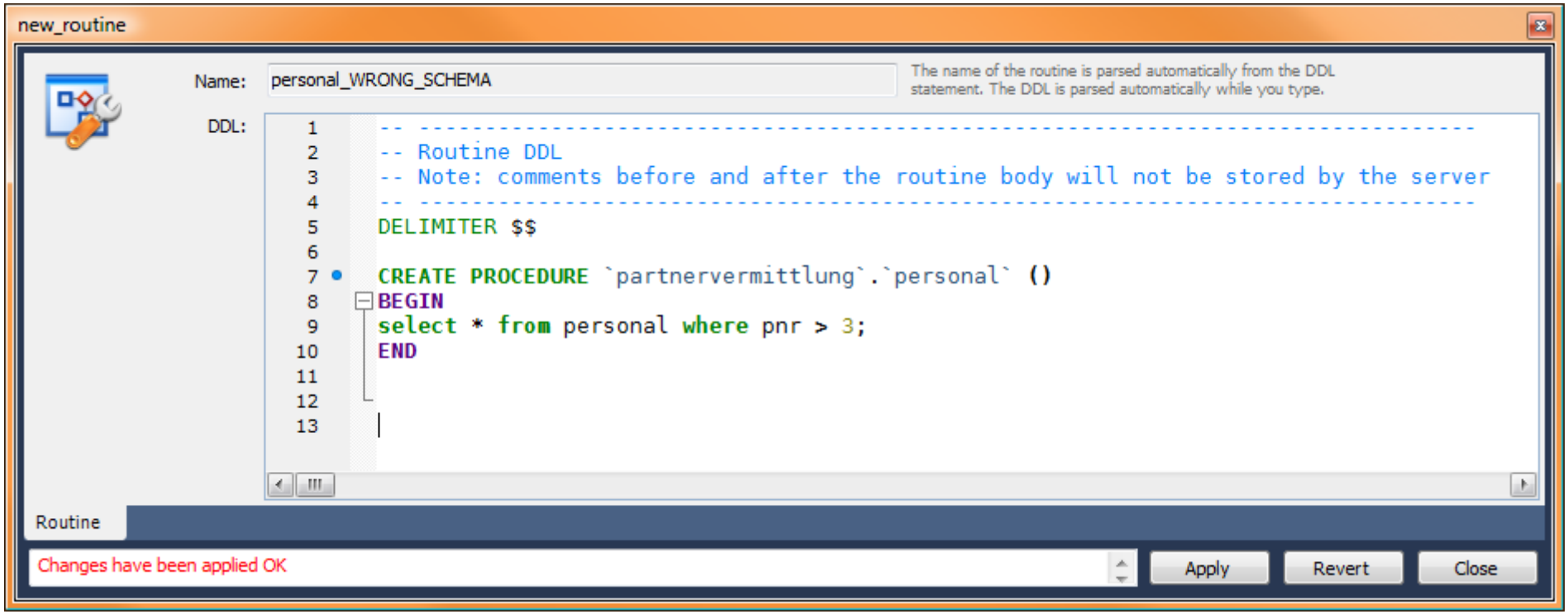
- Stored Funtions, Stored Procedures, Trigger
  - SQL-Statements werden auf der Serverseite gehalten
  - Schleifen, Bedingungen etc. vorhanden (aber Syntax oft abh. vom DBMS!)

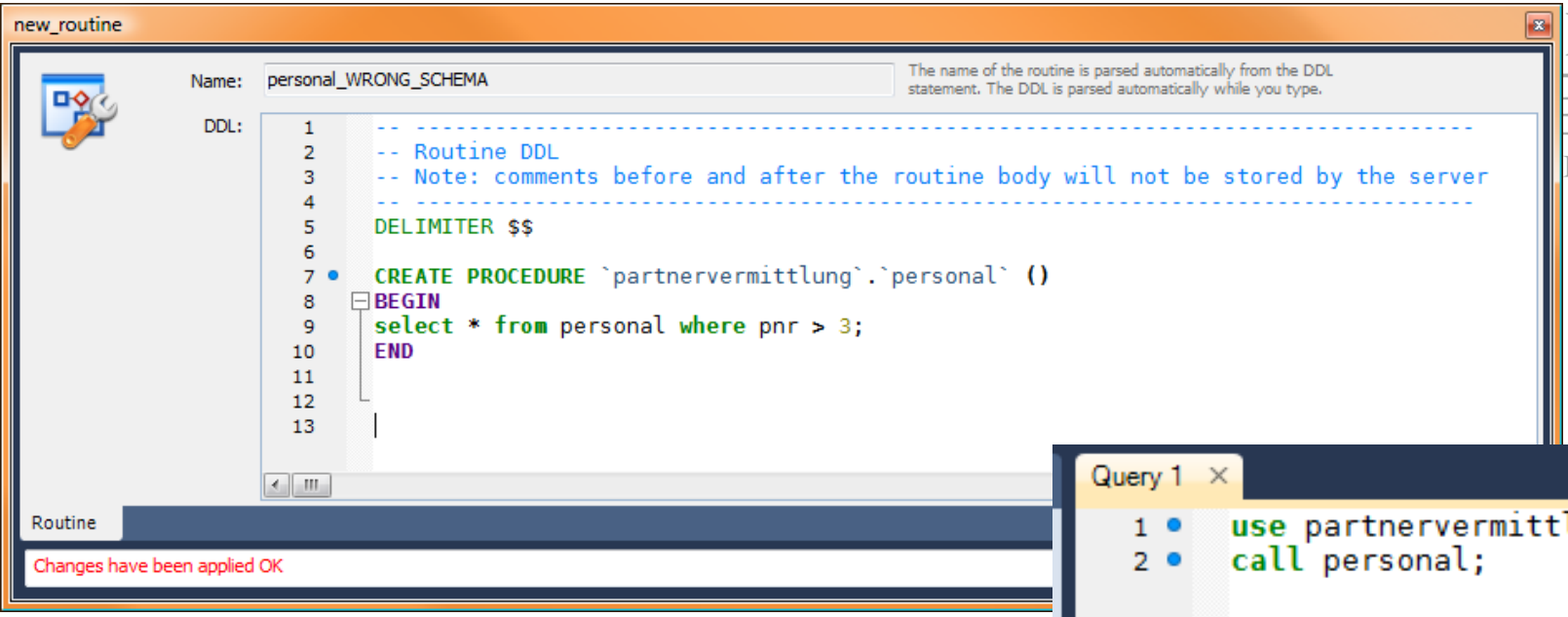
Wann insb. ist dies von Vorteil?

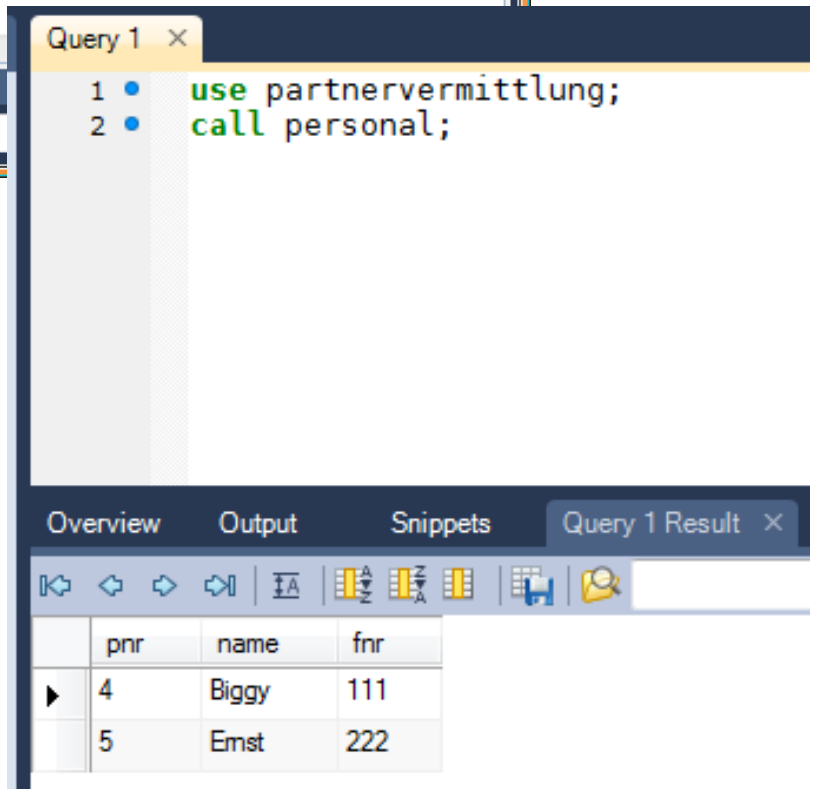
Tutorials: http://www.mysqltutorial.org/mysql-stored-procedure-tutorial.aspx

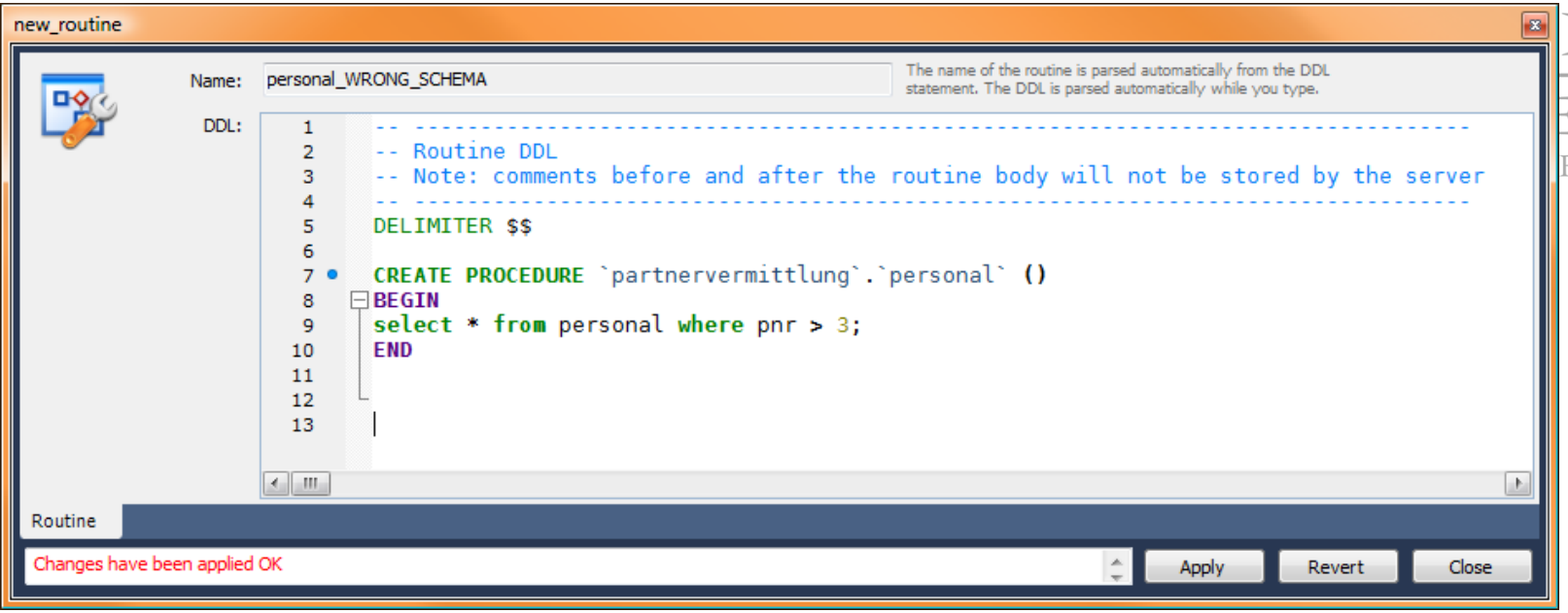
einfache Stored Procedure

GOETHE UNIVERSITÄT FRANKFURT AM MAIN




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
Emst
222




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

komplexere Stored Procedure

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

```
01 DELIMITER $$
02 CREATE PROCEDURE `Capitalize`(INOUT str VARCHAR(1024))
03 BEGIN
04    DECLARE i INT DEFAULT 1;
05    DECLARE myc, pc CHAR(1);
06    DECLARE outstr VARCHAR(1000) DEFAULT str;
07    WHILE i <= CHAR_LENGTH(str) DO
08        SET myc = SUBSTRING(str, i, 1);
09        SET pc = CASE WHEN i = 1 THEN ' '
10                      ELSE SUBSTRING(str, i - 1, 1)
11                 END;
12        IF pc IN (' ', '&', '''', '_', '?', ';', ':', '!', ',', '-', '/', '(', '.')
    THEN
13            SET outstr = INSERT(outstr, i, 1, UPPER(myc));
14        END IF;
15        SET i = i + 1;
16    END WHILE;
17    SET str = outstr;
18 END$$
19 DELIMITER ;
```