Datenbankenprogrammierung - MySQL

Nils Faerber
Unix-AG, Universität/Gesamthochschule Siegen
nils@unix-ag.uni-siegen.de
http://www.si.unix-ag.org/~ nils/

V1.0 – 19. Mai 1998

Zusammenfassung

Kurz: Was ist MySQL?
Schnittstellen von MySQL
Grundlagen des C API
Beispiele

Was ist MySQL?

Eine Entwicklung aus der Not: Man brauchte eine schnelle und effiziente Datenbank, welche auch große Datenbestände handhaben kann.

Entwicklung aus Schweden. Voller kommerzieller Support verfügbar.

Im Prinzip ein kommerzielles Produkt, Sourcecode ist jedoch unter einer sehr freizügigen Lizenz verfügbar.

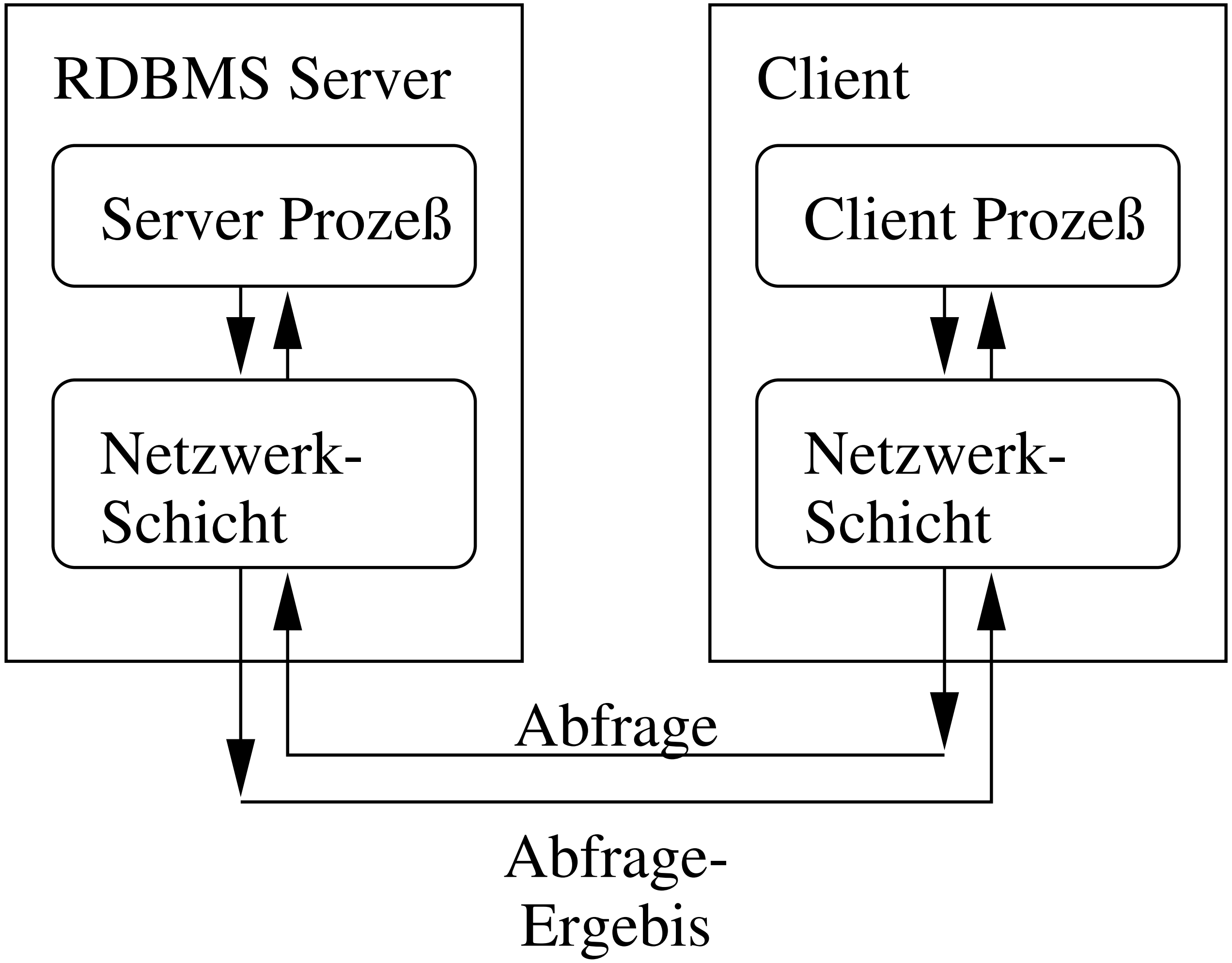
Erst wenn man MySQL als solches oder als Teil eines eigenen Produktes verkaufen will, muß man bezahlen.

Durch die Verfügbarkeit des Sourcecodes, existiert eine große Anwendergemeinde und damit auch breites Spektrum an Zugriffsmöglichkeiten.

Programmierschnittstellen von MySQL

- **C Library** (`libmysqlclient`)
- Perl Interface
- REXX Interface
- Tcl/Tk Interface

Struktur der Kommunikation



Eigenschaften

- Client- und RDBMS-Prozesse laufen **nicht** notwendigerweise auf ein und demselben Rechner
- Das bedeutet, daß alle Abfragen Netzwerktransparent ablaufen!
- Kommunikation verläuft zu einem Zeitpunkt immer nur in einer Richtung, also
  - Client sendet Abfrage zum Server
  - Server liefert Antwort (z.T. nur ein Stück der Antwort und Client fordert weitere Teile anschließend nach)

Struktur eines Client Prozesses

Öffnen einer Verbindung zum Server

↓

Erstellung der Abfrage

↓

Absenden der Abfrage

↓

Empfang des Ergebnisses

↓

ggf. weitere Teile der Antwort holen

↓

Verbindung schließen

Die C Schnittstelle

Durch Einbinden des Headerfiles `mysql.h` stehen alle Funktionen zur Kommunikation mit der Datenbank zur Verfügung.

Zentrale Datentypen:

- `MYSQL` Datenstruktur zur Verbindung
- `MYSQL_RES` Datenstruktur für Abfrageergebnisse
- `MYSQL_ROW` Datenstruktur für eine Ergebniszeile

Nach dem (erfolgreichen) kompilieren muß zu dem Ergebnis die Bibliothek `libmysqlclient.a` gebunden werden.

Um die Folien etwas zu verkürzen werden nachfolgend nur Sourcefragmente präsentiert...

Öffnen einer Verbindung - Funktion

Verwendete Funktion:

```
mysql_connect(&mysql,"host","user","passwort")
```

`&mysql` - Variable zur Identifikation der Verbindung vom Typ `MYSQL`

`host` - Hostname des Rechners auf dem der Server läuft

`user` - Username als welcher zugegriffen werden soll, wenn `NULL` wird der Login-Name verwendet

`passwort` - Passwort für diesen Benutzer, wenn `user==NULL` entfällt dieses und es sollte `passwort=NULL` gesetzt werden

Öffnen einer Verbindung - Beispiel

```
/* Die zentralen Datenstrukturen */
MYSQL mysql;
MYSQL_RES *res;
MYSQL_ROW row;

if (!(mysql_connect(&mysql,HOST,NULL,NULL))) {
  fprintf(stderr,"Fehler beim Verbindungsaufbau!\n");
  exit(-1);
}
```

Abfragen - Funktion

Abfragen sind nun ”normale” SQL Kommandos, welche mit der Funktion `mysql_query()` zur Datenbank geschickt werden.

Die Funktion hat folgenden Prototyp

```
int mysql_query(MYSQL *mysql, const char *query);
```

Es wird also die eben erhaltene Verbindungsidentifikation und ein String welcher die SQL Anfrage enthält übergeben.

Abfragen - Beispiel

```
char query[]="SELECT * FROM Artikel ORDER BY ArtNr";

if (mysql_query(&mysql,query)) {
  fprintf(stderr,"Fehler bei Abfrage!\n");
  exit(-1);
};
```

Zu diesem Zeitpunkt hat der Server die Abfrage bearbeitet und stellt die Antwort zur Verfügung.

Das Ergebnis befindet sich noch beim Server!

Abfragen - Holen des Ergebnisses - Funktionen

- Zur Verfügung stellen des Ergebnisses am Server:

```
MYSQL_RES *mysql_store_result(MYSQL *mysql);
```

- Abfrage, wieviele Zeilen (Datensätze) das Ergebnis hat:

```
int mysql_num_rows(MYSQL_RES *res);
```

- Holen einer Zeile (eines Datensatzes), gibt bei Erreichen des Endes `NULL` zurück:

```
MYSQL_ROW mysql_fetch_row(MYSQL_RES *mysql);
```

- Freigabe des Ergebnispuffers (wichtig!):

```
void mysql_free_result(MYSQL_RES *result);
```

Abfragen - Holen des Ergebnisses - Beispiel

```
res=mysql_store_result(&mysql);

while ((row=mysql_fetch_row(res))) {
  printf("%s\n",row[0]);
}

mysql_free_result(res);
```

Abfragen - Holen des Ergebnisses - Anmerkung

Wie gesehen, werden die Teile des Abfrageergebnisses als String-Arrays geliefert.

Es muß sich also anschließend noch selbst darum gekümmert werden, daß die Strings wieder in passende Datenformate konvertiert werden, also bspw. Zahlen nach Integer o.ä.

Die gelieferten Teile eines Datensatzes werden in der Reihenfolge geliefert, wie sie auch durch den Query angefordert wurden, also bspw. liefert

```
SELECT Ind,ArtBez,ArtNr,Bestand,Liefer from Artikel
```

auch das Ergebnis als

```
row[0]=Ind; row[1]=ArtBez; row[2]=ArtNr;

row[3]=Bestand; row[4]=Liefer;
```

Abfragen - Beenden der Verbindung - Funktion

Schließt die Verbindung zur Datenbank und gibt ggf. noch anstehende Ergebnisse eines Query wieder frei. Dies ist äußerst wichtig, da erst nach 8 Stunden eine Verbindung durch Time-Out vom Server geschlossen wird!

```
void mysql_close(MYSQL *mysql);
```

Weitere nützliche Funktionen

- Im Fehlerfall liefert

  ```
  char *mysql_error(MYSQL *mysql);
  ```

  eine Fehlermeldung in der passenden Landessprache.

- Liste der verfügbaren Tabellen

  ```
  MYSQL_RES *mysql_list_tables(MYSQL *mysql, const char
  *wild);
  ```

Aminitsrative Funktionen

- Neue Datenbank erstellen

```
int mysql_create_db(MYSQL *mysql, const char *DB);
```

- Bestehende Datenbank löschen (Vorsicht! Es gehen damit auch darin enthaltene **Tabellen** verloren!)

```
int mysql_drop_db(MYSQL *mysql, const char *DB);
```

- Den Datenbankserver beenden

```
int mysql_shutdown(MYSQL *mysql);
```

- Grant-Tables neuladen und Cache invalidieren

```
int mysql_reload(MYSQL *mysql);
```

Fragen? - Kritik? - Anregungen?

Ich bin immer offen für kreative Anregungen!

Dies wird sicherlich nicht das letzte mal gewesen sein, daß dieser Vortrag von mir gehalten wurde. Konstruktive Kritik hilft also zukünftigen Zuhörern!

Kontakt:

Nils Faerber

eMail `nils@unix-ag.org`

WWW `http://www.si.unix-ag.org/~ nils/`

Vielen Dank für die Aufmerksamkeit!

Fröhliches *Linux'en*!