Datenbanken mit Linux

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

Welche gibt es für *Linux*?
Eigenschaften am Beispiel MySQL

Übersicht RDBMS für Linux

Ingres - “University” Ingres Public Domain - Source
Postgres - Public Domain, OO-RDBMS - Source
mSQL - Kostenlos für nicht kommerziellen Einsatz - Source
MySQL - Kostenlos, darf nur nicht verkauft werden - Source
QDDB - Grafische Oberfläche, kostenlos - Source
YARD SQL - SQL Client, Server - Source
Interbase (Inprise, ehem. Borland) - Linux Intel
AdabasD (Software AG) - Linux Intel

Und noch viele mehr.

Ingres / Postgres

Sind wohl die Urväter der relationalen Datenbanken.

Ingres gibt es heute in einer freien Version (University Ingres). Sehr groß, sehr mächtig, nicht besonders schnell, für die meisten Anwendungen ”Overkill”

Postgres ist eine weitere freie Implementation von Ingres mit anderen Schwerpunkten.

mSQL

Eine ”light-weight” Entwicklung an einer australischen Universität.

Gedacht für einfache Ablage und Wiederauffinden von mittleren Datenmengen.

Heute hat einer der Hauptentwickler, David Hughes, die Weiterentwicklung in einer eigenen Firma übernommen, seitdem kommerziell, Sourcecode aber immernoch verfügbar.

Gut geeignet als kleine Datenbank für WWW Server, recht gute Dokumentation des API.

MySQL

Auch wieder eine Entwicklung aus der Not: Man brauchte mal wieder 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.

MySQL soll hier später als Beispieldatenbank dienen.

QDDB

Grafischer Ansatz.

Daten werden in einer grafischen Oberfläche dargestellt und können darüber bearbeitet werden.

Starke Betonung von Daten als ”Objekte”.

Aufgebaut auf Tcl/Tk.

YARD SQL

Eine weitere freie SQL Datenbank.

Interbase

Voll kommerzielles Produkt von Inprise (ehemals Borland)

Version 4.0 frei erhältlich für Linux Intel

Version 5.0 wird nur verkauft werden

AdabasD

Voll kommerzielles Produkt der deutschen Firma ”Software AG”.

Kostenlose Demo der ”Personal Edition”, nur beschränkt in der maximalen Anzahl der Datensätze, verfügbar.

Erhältlich nur für Linux-Intel.

Enthält einige grafische Tools zur Administration und Verwaltung.

Beispiel: MySQL

- Zu finden unter `http://www.tcx.se`
- MySQL ist *ein* SQL Datenbank Server
- Hauptaugenmerk bei der Entwicklung: Geschwindigkeit und Rubustheit
- Datentypen: signed/unsigned Integers 1,2,3,4 und 8 bytes long, ‘FLOAT’, ‘DOUBLE’, ‘CHAR’, ‘VARCHAR’, ‘TEXT’, ‘BLOB’, ‘DATE’, ‘SET’ and ‘ENUM’
- Privilegien und Passwortvergabe bis auf Tabellenebene
- ODBC Unterstützung
- Lokalisierbare Fehlermeldungen (verschiedene Sprachen)

Programmteile

Mit dem Script `mysql.server` kann der Serverprozeß gestartet und gestoppt werden. Außer einer Meldung passiert hier nichts!

Das rudimentäre Interface zur Datenbank bildet das Programm `mysql` sowie das Administrationstool `mysqladmin`.

Als erstes wird eine Datenbank mit Hilfe von `mysqladmin` erstellt, also bspw. `mysqladmin create MyDB`.

Die so erstellte Datenbank ethält jetzt noch keine Daten und auch keine Tabellen!

Erstellen von Tabellen

Vor der Tebellengenerierung sollte man sich **sehr** genau überlegen, welche Struktur die Daten haben und welche Relationen zwischen den Daten bestehen.

Am besten hierzu Notizen machen und grafisch die Beziehungen einzeichnen.

Spätere Änderungen und Umstrukturierungen sind meist äußerst schwierig, wenn nicht gar unmöglich!

Beispiel

Datenbank für ein mittleres Vertriebsunternehmen.

Ein solches Unternehmen hat:

1. Lieferanten
2. Artikel
3. Kunden

Alles diese drei unterschiedlichen Datenbereiche haben jedoch Beziehungen untereinander. Die Lieferanten beliefern das Unternehmen mit Artikeln, welche die Kunden kaufen (sollen).

Erste Struktur

Die Tabellen

Datenbanken sollten i.d.R. eine Spalte mit absolut eindeutigem Inhalt enthalten, um über diesen einen *Schlüssel* Datensätze referenzieren zu können.

Im Regelfall wird man dazu eine Art *Index* verwenden, also eine fortlaufende Numerierung; MySQL kann dies automatisch übernehmen.

Die Tabellen könnten dann folgendermaßen gestaltet werden:

Die Tabellen: Lieferanten, Kunden

```
|Field   |Type              |Null|Key|Def|Extra         |
+--------+------------------+----+---+---+--------------+
|Ind     |int(11) unsigned|    |PRI|0  |auto_increment|
|Name    |char(80)          |NO  |   |   |              |
|Adresse|char(200)         |YES |   |   |              |
```

Null: Das Feld darf leer sein

Key: PRI = Das Feld ist der primäre Schlüssel (ausgezeichnete Spalte)

Def: (Default) Wert der bei nicht-Belegung automatisch eingefügt wird

Extra: auto_increment = Das Feld wird automatisch numeriert, mit jedem neuen Datensatz incrementiert

Die Tabellen: Artikel

```
|Field  |Type            |Null|Key|Def|Extra         |
+-------+----------------+----+---+---+--------------+
|Ind    |int(11) unsigned|    |PRI|0  |auto_increment|
|ArtBez |char(80)        |NO  |   |   |              |
|ArtNr  |char(20)        |NO  |   |   |              |
|Bestand|int(11) unsigned|NO  |   |0  |              |
|Liefer |int(11) unsigned|NO  |   |0  |              |
```

Erstellung der Tabellen

Zuerst wird nun die Datenbank erstellt:

```
mysqladmin create Firma
```

erstellt eine Datenbank namens *Firma*.

Mit dem Programm `mysql` können die Tabellen nun erstellt werden. Dieses Programm arbeitet über eine eigene Kommandozeile und akzeptiert SQL als Eingabesprache. Start mit

```
mysql Firma
```

also als Argument den Datenbanknamen und es meldet sich dann mit

```
mysql>
```

Erstellung der Tabellen

Die Eingabe der SQL Befehle kann auch durch ein Script mit Umlenkung geschehen, also

```
mysql Firma < SQL-script
```

Doch hier soll der interaktive Modus weiter betrachtet werden. Als nächstes nun die Erstellung der ersten Tabelle, die Lieferanten

```
CREATE TABLE Lieferanten (Ind INT NOT NULL DEFAULT
0 AUTO_INCREMENT, Name CHAR(80) NOT NULL, Adresse
CHAR(200), PRIMARY KEY (Ind)) \g
```

Das `\g` am Ende der Zeile bedeutet etwa *Go!*, also Anforderung an den Serverprozeß abschicken.

Das gleiche machen wir nun noch mit den `Kunden`, also wie oben, nur `Lieferanten` durch `Kunden` ersetzen.

Erstellung der Tabellen

Und nun noch die Tabelle für die Artikel

```
CREATE TABLE Artikel (Ind INT NOT NULL DEFAULT 0
AUTO_INCREMENT, ArtBez CHAR(80) NOT NULL, ArtNr
CHAR(200),Bestand INT DEFAULT 0, Liefer INT NOT
NULL,PRIMARY KEY (Ind)) \g
```

Einfügen von Daten

Dateneingabe in die Datenbank geschieht mit dem Befehl `INSERT` unter Angabe der zu verwendenden Spalten und der zukünftigen Werte, bspw.

```
INSERT INTO Lieferanten (Name, Adresse) values (''First
Computer'',''89763 Stuttgart'') \g
```

Der eindeutige Index wird dabei automatisch generiert!

Abfrage der Daten

Um Daten wieder aus einer Datenbank zu lesen, wird der sehr mächtige SQL Befehl `SELECT` verwendet. Mit Hilfe von Selects werden auch die Relationen zwischen den Daten (Tabellen) hergestellt, bspw. Inhalt einer Tabelle anzeigen:

```
SELECT * from Lieferanten \g
```

Welches besagt, es sollen alle Felder (repräsentiert durch den '*') aus der Tabelle `Lieferanten` ausgegeben werden:

```
+-----+----------------+--------------------+
| Ind | Name           | Adresse            |
+-----+----------------+--------------------+
|   1 | First Computer | 89763 Stuttgart    |
+-----+----------------+--------------------+
1 row in set (0.01 sec)
```

Erweiterte Abfrage

Weitere Optionen (von vielen mehr!) für den Select:

- `ORDER BY` *sortiere nach*
- `WHERE` Einschränkungen der Suche
  - `AND` Logische Verknüpfung mehrerer Einschränkungen
  - `OR`
  - `= < > <= >= <>`
  - `LIKE` Suche mit Platzhaltern `% und ?`

Sicherheit

MySQL unterstützt eine Reihe von Sicherheitsfunktionen. Dazu werden in der Datenbank `mysql` zwei Tabellen geführt:

- `user` Zugriffsrechte auf Basis des Benutzers, der eine Anfrage an die Datenbank stellt.
- `db` Zugriffsrechte auf Basis des Datenbanknamens an welche eine Anfrage gestellt wird.
- `host` Zugriffsrechte auf Basis des Rechners von welchem aus auf die Datenbank zugegriffen wird; wird in größeren Netzwerken benutzt, in welchen Benutzernamen auf unterschiedlichen Rechnern mehrmals vorkommen (bspw. Benutzer A auf Rechner A ist nicht derselbe wie Benutzer A auf Rechner B)

Sicherheit - `user`

Noch mehr Optionen:

- `Host` - Hostname für den Benutzer
- `User` - Username
- `Password` - Verschlüsseltes Passwort, bspw. für den Zugriff über ODBC, wenn also noch keine Authentifizierung des Benutzers stattgefunden hat
- `Select_priv` - Darf dieser Benutzer aus ein `SELECT` ausführen?
- `Insert_priv` - Darf dieser Benutzer aus ein `INSERT` ausführen?
- `Update_priv` - Darf dieser Benutzer aus ein `UPDATE` (aktualisieren von Werten einer Tabelle) ausführen?

- `Delete_priv` - Darf dieser Benutzer aus ein `DELETE` (Löschen von Werten aus einer Tabelle) ausführen?
- `Create_priv` - Darf dieser Benutzer aus eine neue Tabelle erstellt werden?
- `Drop_priv` - Darf dieser Benutzer aus eine bestehende Tabelle löschen?
- `Reload_priv` - Darf der Benutzer ein Neuladen der Tabellen und Konfigurationsdateien veranlassen?
- `Shutdown_priv` - Darf der Benutzer den Datenbankserver beenden?
- `Process_priv` -
- `File_priv` - Darf der Benutzer in lokale Dateien schreiben?

Sicherheit - `db`

Enthaltene Einträge:

- `Host` - Hostname
- `Db` - Auf welche Datenbank sich der Eintrag bezieht
- `User` - Username der zugelassen werden soll
- `Select_priv`,`Insert_priv`,`Update_priv`, `Delete_priv`,`Create_priv`,`Drop_priv` - wie in der `user`-Tabelle

Sicherheit - `host`

Enthaltene Einträge:

- `Host` - Hostname
- `Db` - Auf welche Datenbank sich der Eintrag bezieht
- `Select_priv`,`Insert_priv`,`Update_priv`, `Delete_priv`,`Create_priv`,`Drop_priv` - wie in der `user`-Tabelle

ODBC

ODBC ist eine von MicroSoft standardisierte Programm-Schnittstelle für Datenbankzugriffe.

Für MySQL existiert ein Windows ODBC Treiber.

Der Treiber bildet die Schnittstelle zwischen Windows-Programmen und einem MySQL Server im Netzwerk.

Dadurch ist es möglich, einen Linux MySQL Server praktisch unmerklich in eine Windows Umgebung zu integrieren.

ODBC - Vorteile

- Viele Windows Programme unterstützen ODBC, bspw. MS-Access, Visual Basic, Excel, etc.
- Durch die ODBC Schnittstelle hat man Zugang zu einem leistungsfähigen Datenbankserver; MySQL. In der Regel ist MySQL wesentlich schneller als vergleichbare Windows oder WindowsNT Produkte.
- MySQL ist zudem wesentlich preisgünstiger als bspw. der MS-SQL-Server.
- Dies ermöglicht den Aufbau zentraler Datenbanken, was sonst nur über Umwege oder teure Produkte möglich wäre.

Grafische Oberflächen

Sind nicht Bestandteil des MySQL Paketes, jedoch verfügbar

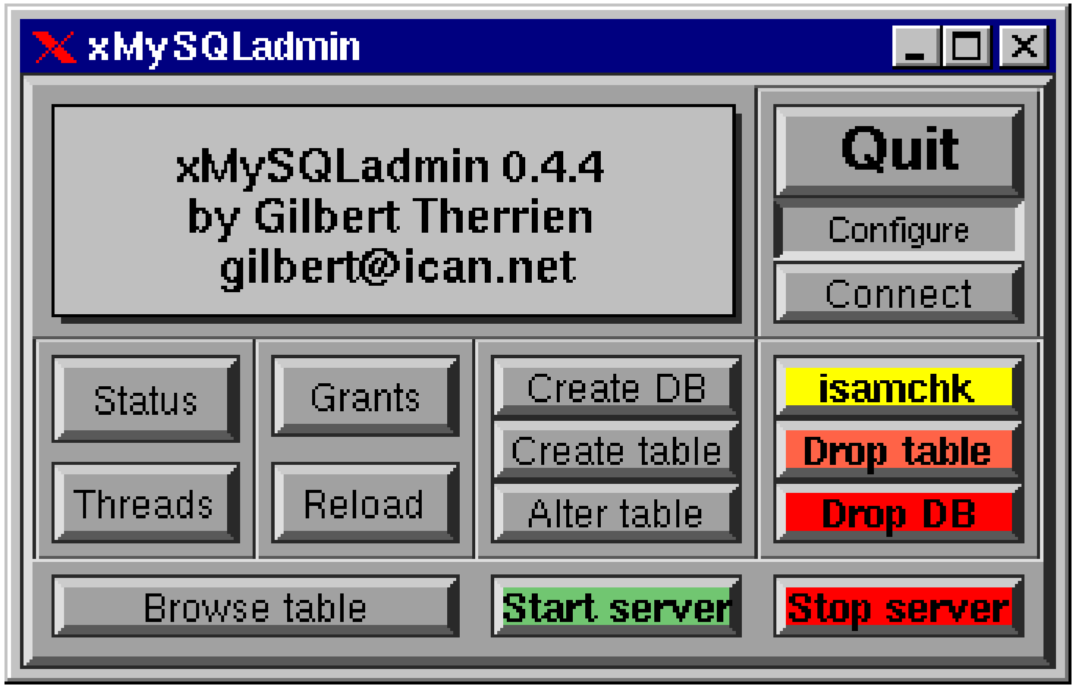
Als Beispiele: `xmysqladmin` und `xmysql`

Beides grafische X11 Oberflächen zur Administration und Abfrage

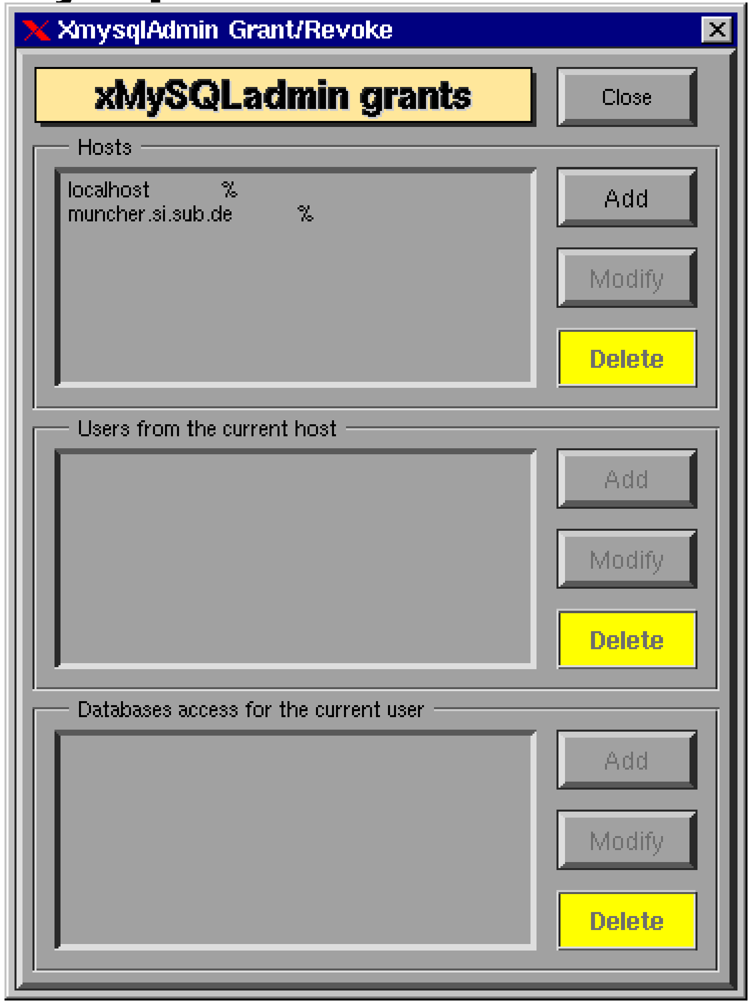
Xmysql bietet darüber hinaus Hilfe bei der Erstellung der SQL Queries (noch eingeschränkt)

Xmysqladmin macht alles in allem den ausgereifteren Eindruck

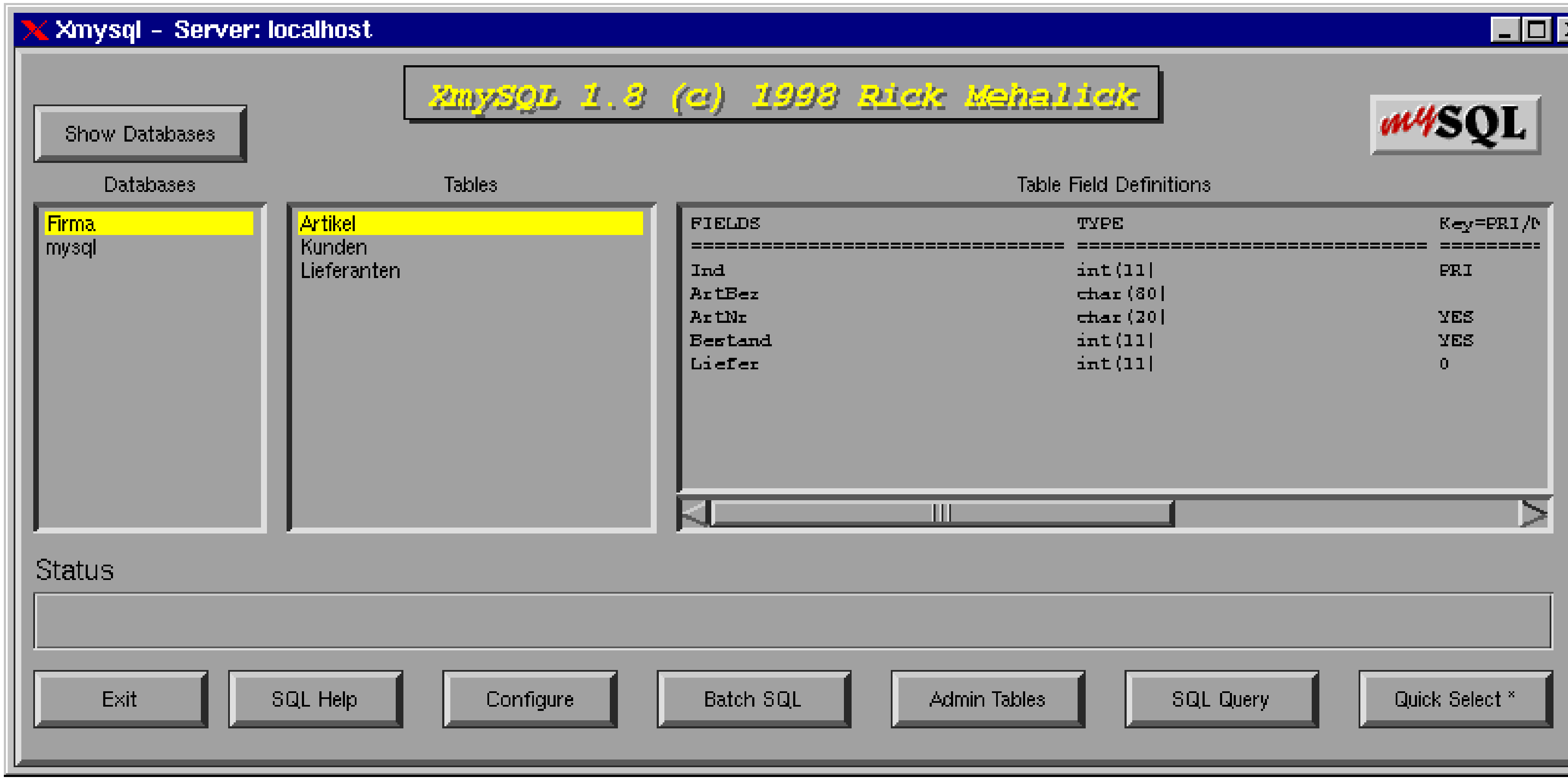
Xmysqladmin - Hauptmenü



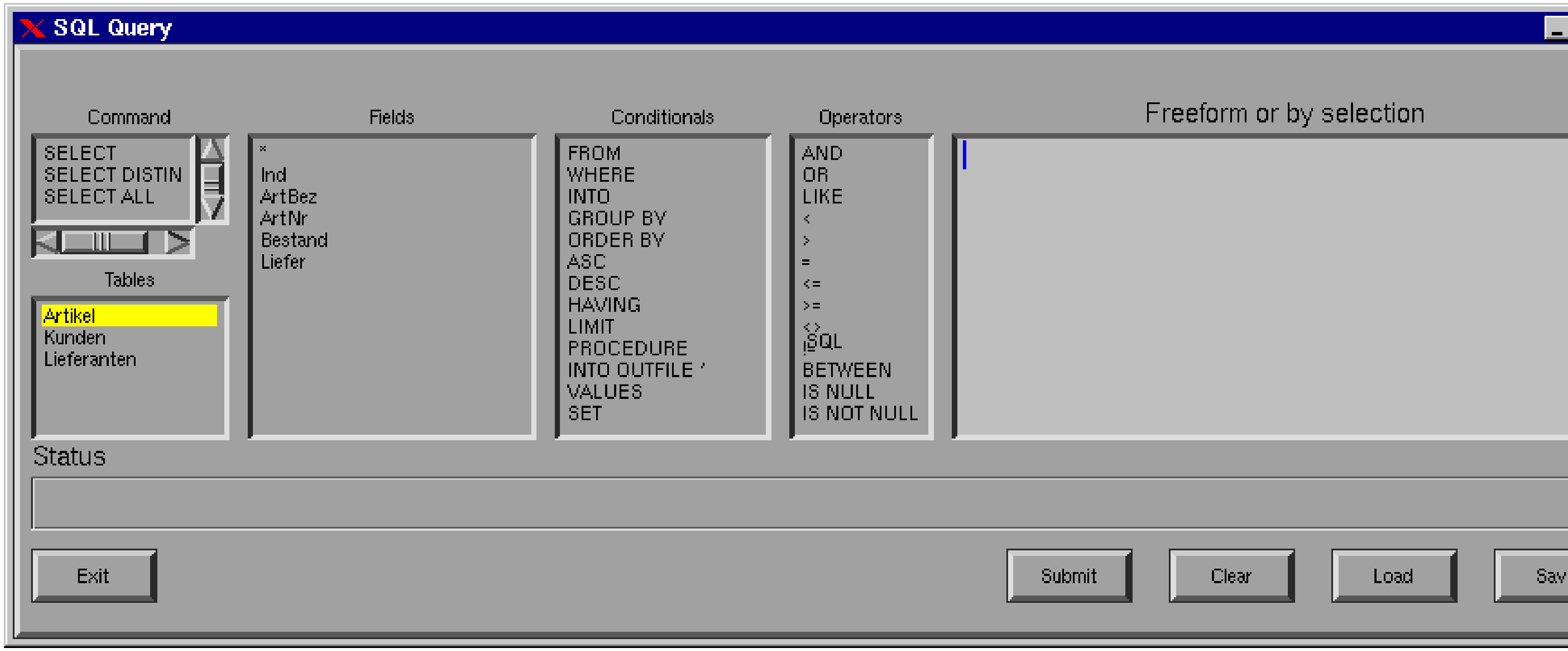
Xmysqladmin - Granttables



Xmysql - Hauptmenü



Xmysql - Query



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*!