Relationale Datenbanken:

Toolgestütztes Modellieren

Inhaltsverzeichnis:

1 Einrichten des Datenbankmodells auf einem Server

Vorbemerkung

Die Themen „Datenbankmodellierung“ und „Arbeiten mit SQL" können mit unterschiedlichen relationalen Datenbankmanagementsystemen unterrichtet werden. In dieser Handreichung wird als RDBMS die Open-Source-Software MySQL verwendet. Diese Entscheidung fiel insbesondere im Hinblick auf den im Lehrplan als Inhalt vorgegeben "Zugriff auf Datenbanken aus objektorientiert entwickelten Softwaresystemen". MySQL bietet zudem als bevorzugtes Einsatzgebiet die Datenspeicherung für Web-Services und wird häufig in Verbindung mit dem Webserver Apache und PHP, einer Skriptsprache, die hauptsächlich zur Erstellung von dynamischen Webseiten verwendet wird, eingesetzt. Diese Kombination wird entsprechend der Anfangsbuchstaben der beteiligten Software als LAMP, MAMP bzw. WAMP bezeichnet, je nachdem ob als Betriebssystem Linux, Mac OS oder Windows benutzt wird.
Die Erarbeitung der Lehrplaninhalte erfolgt in dieser Handreichung mit Hilfe eines WAMP-Servers. Er lässt sich mit Hilfe von *xampp* einfach (auch ohne Windows-Administrator-Rechte) installieren.

Erste Hinweise zur unmittelbaren Umsetzung der Ausführungen dieses Kapitels mit MySQL werden mit dieser Handreichung abgedeckt.

Exkurs: Die Arbeitsumgebung zum Arbeiten mit Datenbanken

Das Client – Server – Prinzip

In einer Client – Server – Umgebung bieten bestimmte Rechner (Server) Funktionen (Dienste) für andere Rechner (Clients) an, die diese Dienste nutzen. Der Server nimmt die Anfragen der Clients entgegen und sendet diesen daraufhin die angeforderten Daten über das Netz. Als Server bezeichnet man den zentralen Rechner (oder das zentrale Programm), der in einem Computer-Netz Dienste (zum Beispiel E-Mail), Anwendungsprogramme und Daten (zum Beispiel eine Adressdatenbank) bereitstellt. Der Client ist ein Computer oder ein Programm, das über eine Datenverbindung auf einen Server zugreift, um bestimmte Datenverarbeitungsprozesse durchzuführen oder um Daten abzurufen. Die meisten Computer-Netze, insbesondere das Internet, sind nach dem Client-Server-Prinzip aufgebaut.

Die Begriffe "Client" und "Server" bezeichnen nicht nur die entsprechenden Computer, (Hardware), sondern auch die erforderlichen Computerprogramme (Software).

Beispiele für Client – Server – Architekturen:

Client-Programm	**Server**
Browser, z. B. Internetexplorer	Webserver
E-Mailprogramm, z. B. Outlook	Mailserver
FTP-Client, z.B. Wise Ftp	Ftp- Server
Datenbank-Frontend - MySQL Workbench - phpMyAdmin	Datenbankserver, z. B. MySQL

Hinweis: In der Praxis ist die physikalische Unterscheidung zwischen Client und Server nicht immer eindeutig, weil ein Computer auch beide Funktionen wahrnehmen kann. So können auf einem Arbeitsplatzrechner gleichzeitig der Datenbankserver mit den Datenbanken (*MySQL*) und das Programm zur Kommunikation mit dem Datenbankserver (*MySQL Workbench*) installiert sein.

Im Unterricht ist es zu empfehlen, die Client – Server – Architektur zu visualisieren (Siehe Seite 6).

Herstellen einer Client – Server – Umgebung

➢ Java Runtime Environment installieren

Bevor mit dem Apache-Webserver gearbeitet werden kann, muss sichergestellt sein, dass auf dem Rechner eine Java-Plattform vorhanden ist. Unter http://java.com/de/download/ steht die aktuelle Version des benötigten *Java Runtime Environment* (JRE)zum Download bereit, die kostenlos und ohne Registrierung installiert werden kann. Die JRE ist nicht notwendig, wenn nur mit dem MySQL-Server gearbeitet wird.

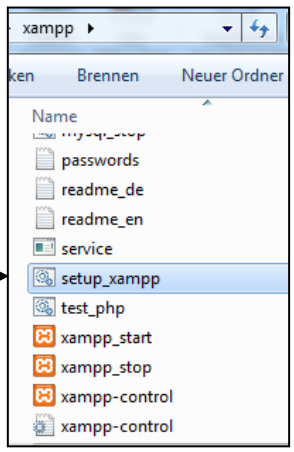
➢ XAMPP "installieren" und den MySQL-Server starten

- Laden Sie von der Seite http://www.apachefriends.org/de die Datei "*xampp-win32-1.8.2.zip*" herunter (es handelt sich hierbei um die, bei der Erstellung dieser Handreichung, aktuelle Version – Juni 2013).

  Es werden unterschiedliche Installationsvarianten angeboten:

  - Ein MSI-Paket, dass nach der Ausführung die Installation auf dem PC übernimmt;
  - ein ZIP-Archiv, das ohne weitere Installationseingaben einfach in ein gewünschtes Verzeichnis extrahiert wird und dann startbereit ist.

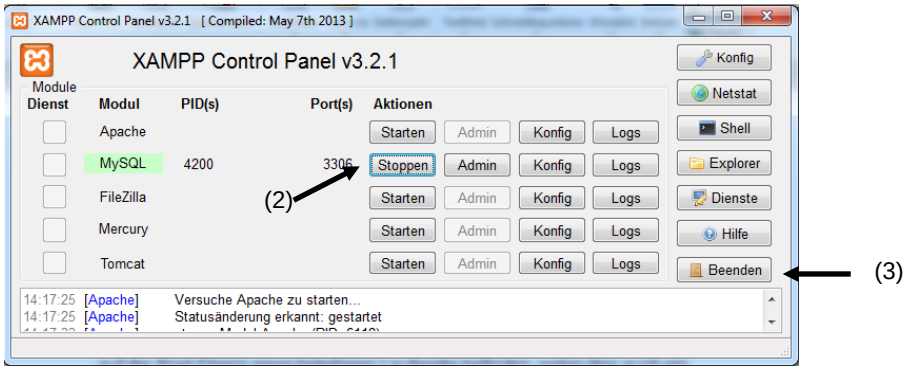
  Mit dem Ausführen des MSI-Pakets beziehungsweise dem Entpacken des ZIP-Archives wird in einem individuell festgelegten Installationslaufwerk ein Ordner "*xampp*" angelegt, in den alle notwendigen Daten kopiert werden. Das Installationsverzeichnis sollte sich auf der Root-Ebene eines beliebigen Laufwerks befinden, wobei dies auch ein Netzlaufwerk, beispielsweise *H:\XAMPP,* sein kann.



- Im Ordner XAMPP sollte anschließend die Datei "*setup_xampp.bat*" ausgeführt werden. Dabei werden die Pfadangaben des Serverlaufwerks festgelegt. Dieser Schritt ist wichtig, wenn nicht in das Root-Verzeichnis entpackt wurde.

Damit ist ein kompletter *WAMP*-Server mit *Apache*-Webserver und *php*-Unterstützung, *MySQL*-Datenbank und *phpMyAdmin* als webbasiertes Frontend zur Datenbankadministration installiert.

(1) Mit der Datei *xampp_control.exe* xampp-control im Ordner ..*xampp* wird ein Control Panel geöffnet, mit dessen Hilfe der MySQL-Server gestartet werden kann.



(2) Der Datenbankserver MySQL wird mit Hilfe des Panels gestartet oder gestoppt.

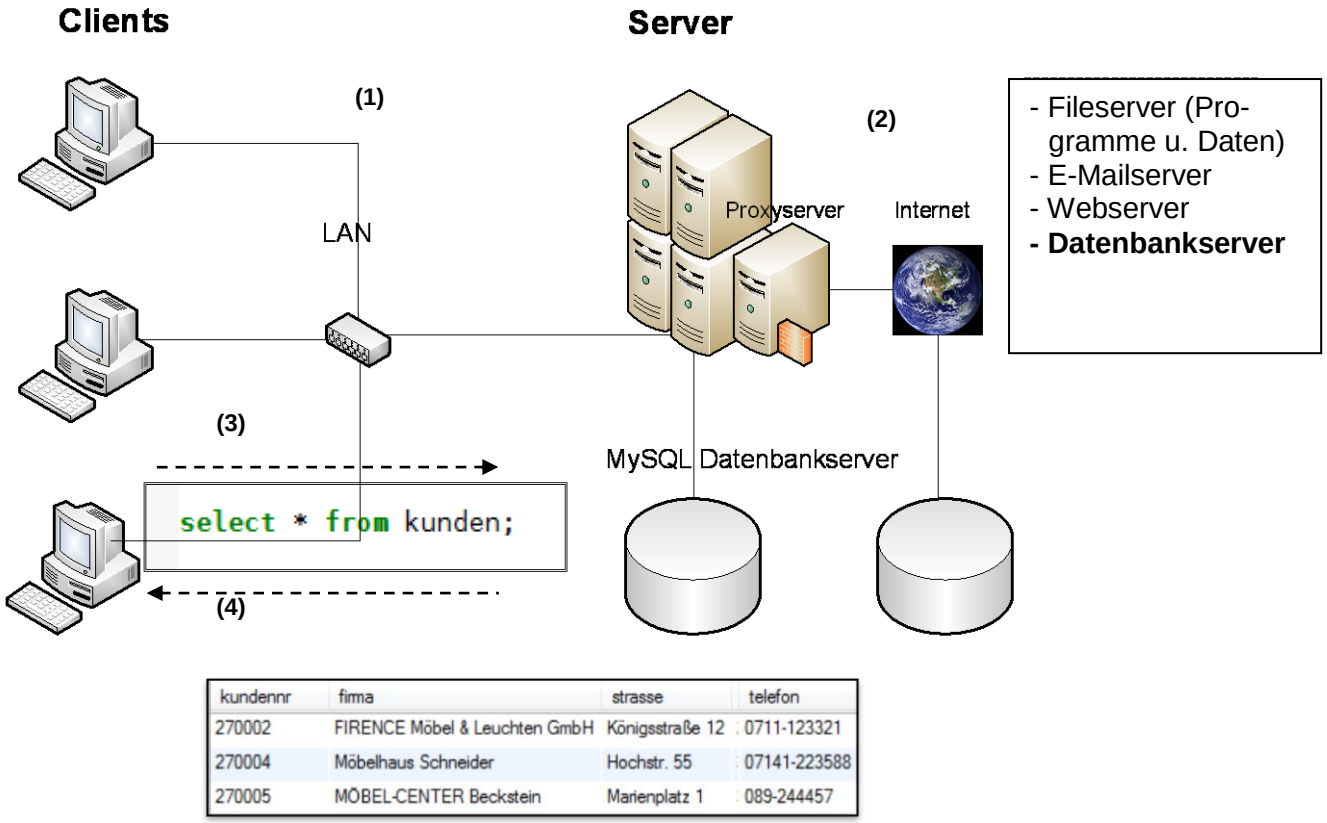
(3) Zum Beenden der Sitzung muss der Button [**`Beenden`**] gedrückt werden. Das Panel wird dann geschlossen.

Der Server auf dem eigenen Rechner (*localhost*) ist unter der IP-Adresse "127.0.0.1" oder unter dem Namen "*localhost*" erreichbar.

Mit der Installation wird für mySQL-Datenbanken als Standardpfad ***../xampp/mysql/data*** festgelegt.

Hinweis: Für den Betrieb von *XAMPP* ist darauf zu achten, dass kein anderes, konkurrierendes Webserver-Programm (z.B. *Internet Information Server IIS*) aktiv ist, da sonst der Apache-Webserver behindert würde.
Um mit dem lokalen Webserver kommunizieren zu können, muss gegebenenfalls das Internet-Telefonierprogramm "Skype" deaktiviert (heruntergefahren) werden.

Übersicht über die Elemente der Client – Server - Umgebung



kundennr	firma	strasse	telefon
270002	FIRENCE Möbel & Leuchten GmbH	Königsstraße 12	0711-123321
270004	Möbelhaus Schneider	Hochstr. 55	07141-223588
270005	MÖBEL-CENTER Beckstein	Marienplatz 1	089-244457

(1) In einer Client – Server – Umgebung nehmen die Clients mit Hilfe geeigneter Software (Client–Software) Serverdienste in Anspruch. Hier ist die Client-Software die *MySQL Workbench*.

(2) Die Serverdienste können im lokalen Netz (LAN) oder im Internet zur Verfügung stehen.

(3) Die Kommunikation mit dem Datenbankserver MySQL erfolgt mit Hilfe der Datenbanksprache **SQL (Structured Query Language)**. Die SQL-Befehle werden mit Hilfe der Client-Software wie beispielsweise ***MySQL Workbench*** zum Datenbankserver gesandt und dort bearbeitet.

(4) Der Datenbankserver liefert dann dem Client das Ergebnis.

➢ Die MySQL Workbench als Frontend zum Arbeiten mit Datenbanken

Die *MySQL Workbench* ist ein Werkzeug, um Datenbanken zu modellieren, zu implementieren und zu verwalten. Möglichkeiten zur Serververwaltung (Konfiguration, Benutzerverwaltung) sind möglich. Die *MySQL Workbench* stellt alle Funktionen zur Verfügung, die ein Datenmodellierer für die Erstellung komplexer *Entity Relationship Modelle (ERM)* sowie für ein *Forward* und Reverse *Engineering* benötigt.

Die MySQL Workbench kann von der Webseite http://www.mysql.de/products/workbench/ heruntergeladen werden. Es werden unterschiedliche Installationsvarianten angeboten:

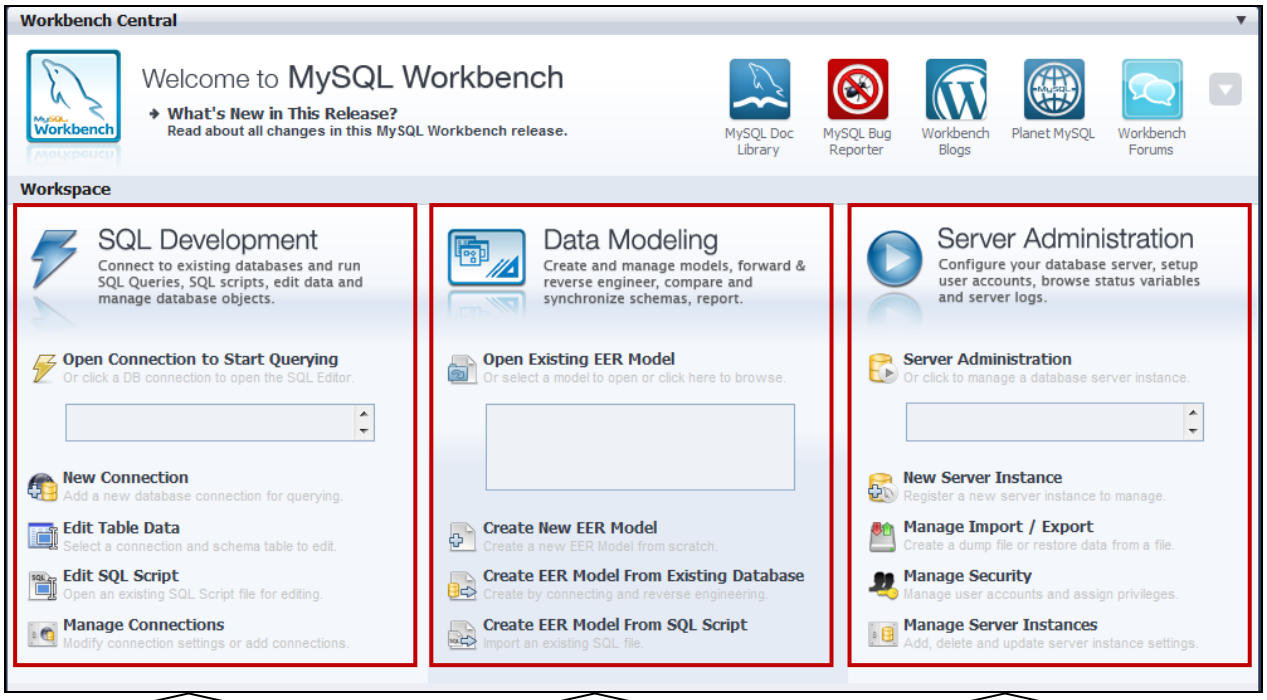
- Ein MSI-Paket, dass nach der Ausführung die Installation auf dem PC übernimmt;
- ein ZIP-Archiv, das ohne weitere Installationseingaben einfach in ein gewünschtes Verzeichnis extrahiert wird und dann startbereit ist.

Für die Installation der aktuellen Version der MySQL Workbench muss sichergestellt sein, dass auf dem PC

- das *.NET Framwork 4* installiert ist. Das .Net Framework stellt Bibliotheken und Dienste sowie eine Laufzeitumgebung zur Verfügung, mit denen Anwendungen auf Betriebssystemfunktionen zugreifen können. Das aktuelle .Net Framework 4 kann von der Webseite http://www.microsoft.com/de-de/download/details.aspx?id=17851 heruntergeladen werden.

- das *Microsoft Visual C++ 2010 Redistributable Package (x86)* installiert ist. Damit werden Laufzeitkomponenten von Visual C++ installiert, die notwendig sind, um auf PCs, auf denen kein Visual C++ installiert ist, Anwendungen auszuführen, die in Visual C++ entwickelt wurden. Das Microsoft Visual C++ 2010 Redistributable Package (x86) kann von der Webseite http://www.microsoft.com/de-de/download/details.aspx?id=5555 heruntergeladen werden.

Für die Kommunikation mit Datenbanken wird die Sprache ***SQL (Structured Query Language)*** eingesetzt. Je nachdem welche Aufgabenstellung bearbeitet werden soll, werden die SQL-Befehle in Klassen unterteilt. In der MySQL Workbench ist diese „Klasseneinteilung“ im Startbildschirm abgebildet.



- **Datenmanipulation** (**DML** = Data Manipulation Language). Daten einzufügen, ändern, oder löschen.
- **Datenbankabfragen** **(DQL =** Data Query Language) Auswahl/Auswertung.von Daten.

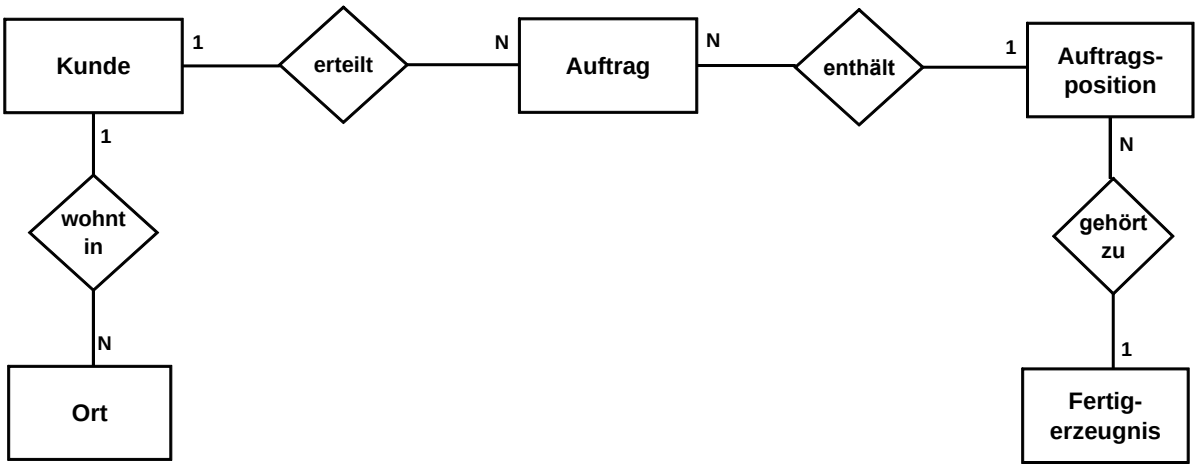
- **Datendefinition** (**DDL** = Data Definition Language) Befehle zur Definition von Tabellen und anderer Datenstrukturen

- **Datenkontrollen** (**DCL** = Data Control Language) Befehle zur Kontrolle der Zugriffsberechtigungen

1.1 Datenbankmodellierung mit Hilfe der MySQL Workbench

Problemstellung

Aus dem für die Auftragsbearbeitung der MÜCKE & PARTNER OHG entwickelten Entity Relationship Model (ERM) soll für die toolgestützte Modellierung mit Hilfe der *MySQL Workbench* zunächst nur der folgende Datenbankausschnitt, der in den ersten Modellierungsschritten erarbeitet wurde, betrachtet werden.



Das Relationenmodell zum Prozess "Auftragserfassung"

Das Relationenmodell umfasst zunächst neben den Primär- und Fremdschlüsseln nur die wichtigsten Attribute.

kunden (kundennr, firma, strasse, ↑ortnr)

auftraege(auftragsnr, auftragsdat, ↑kundennr)

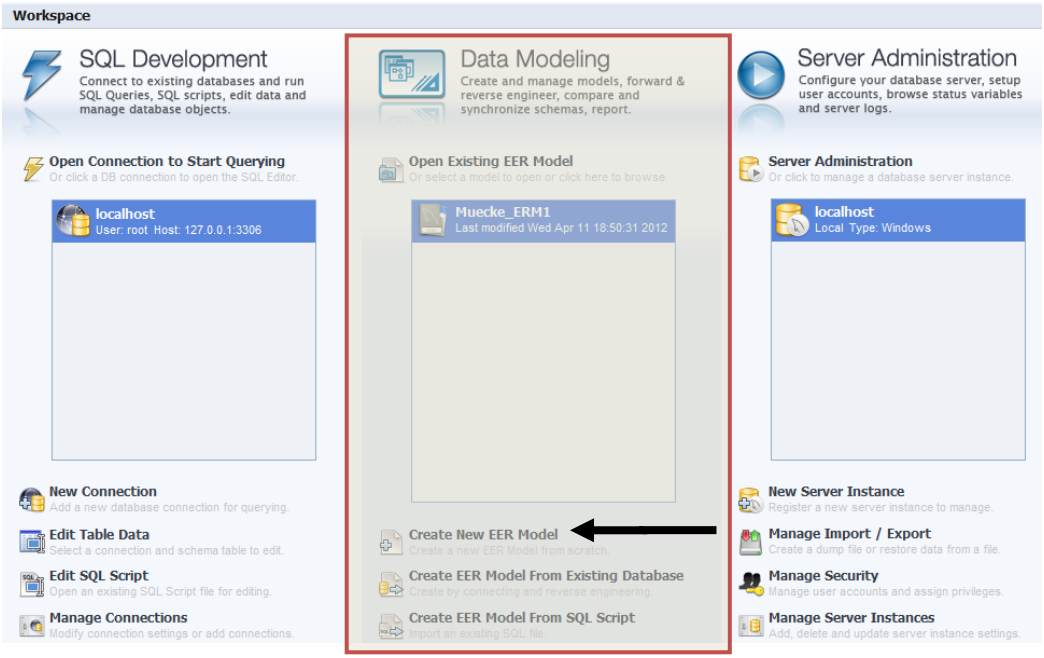
auftragspositionen (auftragsposnr, menge, ↑auftragsnr, posnr, ↑fertigerznr)

fertigerzeugnsse (fertigerznr, bezeichnung, lagerbestand, herstellkosten)

orte (ortnr, plz, ort)

1.1.1 Erstellen eines erweiterten Entity Relationship Model (EERM)

Zum Erstellen/Bearbeiten eines erweiterten Entity Relationship Models wird der Programmteil ***Data Modeling*** der *MySQL Workbench* verwendet.



➢ **Anlegen eines neuen Modells**

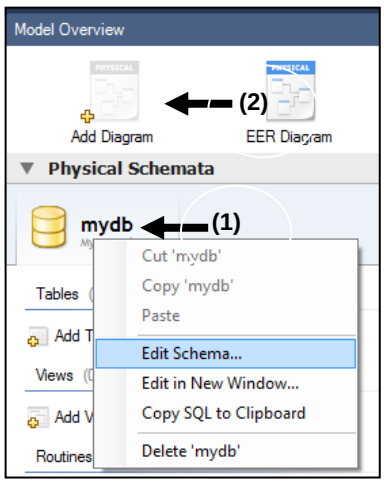
Nach dem Starten der *Workbench* wird zuerst ein neues Modell mit Namen muecke *eingerichtet*. Dazu wird aus dem Programmteile *Data Modeling* das Menü

`Create New EER Model` aufgerufen.

➢ **Datenbanknamen vergeben und Diagramm anlegen**

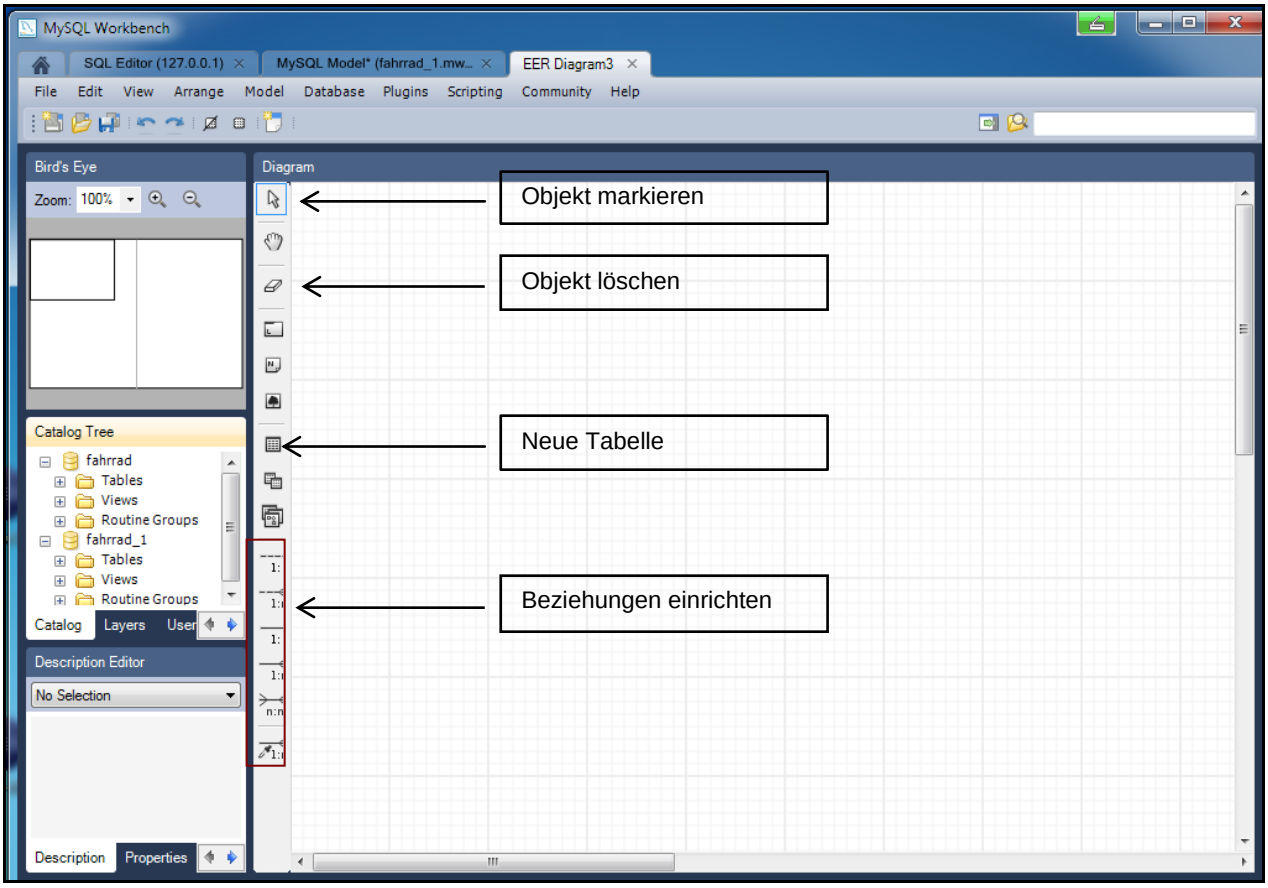
(1) Im angelegten Modell wird im Fenster *Model Overview* zuerst der Namen für die zu erstellende Datenbank festgelegt. Dazu wird für das aktuelle Datenbankschema *mydb* mit der rechten Maustaste ein Kontextmenü geöffnet und darin der Menüpunkt Edit Schema… geöffnet.

(2) Hier ist der neue Datenbankname einzugeben. Damit Umlaute, ß etc. erkannt werden, ist zusätzlich der utf8-Zeichensatz zu wählen (Collation: → *utf8_general_ci*).
Anschließend kann mit einem Doppelklick auf das Symbol Add Diagramm eine neue Entwurfsfläche zum Erstellen des ***Entity Relationship Models geöffnet*** werden.

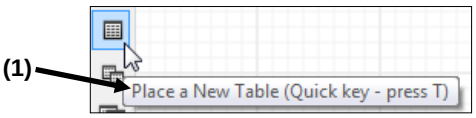


Aufbau des Diagrammfensters

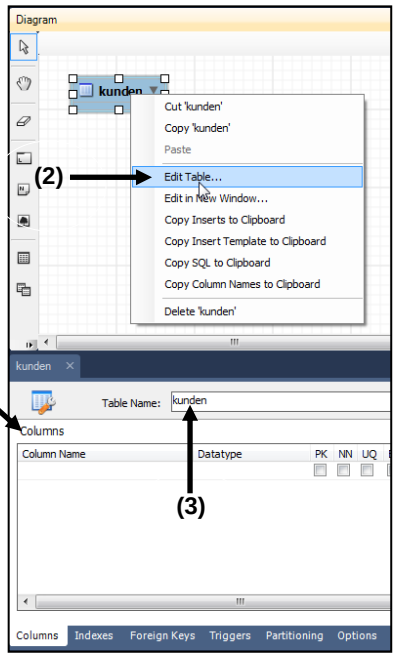
Nach einem Doppelklick auf das Symbol **`Add Diagram`** wird das Entwurfsfenster zum Modellieren des ERM geöffnet:



➢ Tabellen anlegen und editieren



(1) Mit dem Symbol **`Place a New Table`** wird eine neue Tabelle mit dem Namen *table1* im Diagrammfenster erzeugt.



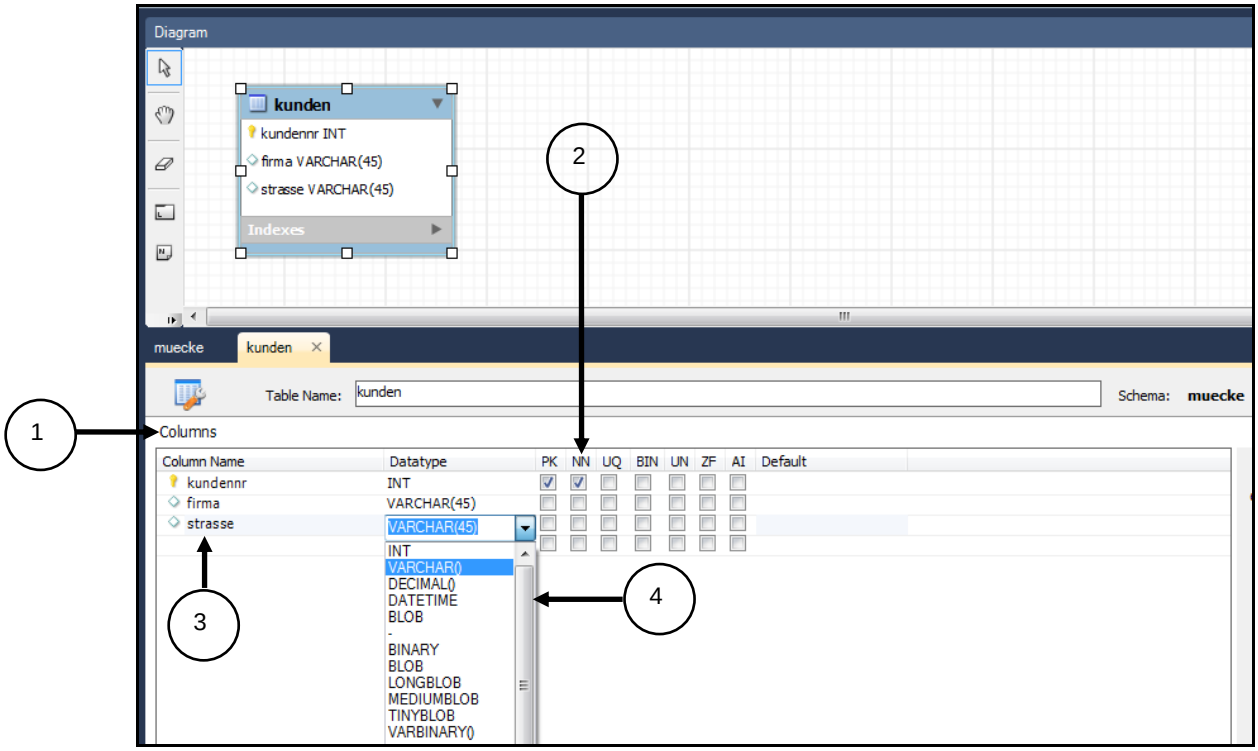
(2) Anschließend wird mit einem Doppelklick auf die neu eingerichtete Tabelle *table1* oder mit dem Kontextmenü zu *table1* die Tabelle editiert (Menüpunkt **`Edit Table`**).

(3) Zunächst wird der Namen der Tabelle eingegeben. In dieser Handreichung werden **Tabellennamen klein** geschrieben.

(4) Schließt man die Eingabe des Namens ab, wird das Register *Columns* geöffnet und die Attribute können erfasst werden.

➢ Erfassen und Editieren der Attribute

Im Register *Columns* des Menüs **`Edit Table`** können die Attribute der Objekte festgelegt werden.



(1) Zunächst wird im Register *Columns* der vom System vorgeschlagene Primärschlüssel angezeigt. Der Primärschlüssel erhält vom System den Namen der Tabelle mit dem Präfix *id,* für die Tabelle *kunden* also *idKunden*. Der vorgegebene Datentyp ist *INT*. Der vorgeschlagene Namen kann beibehalten oder geändert werden. Mit einem Doppelklick in das Feld wird es editiert und kann geändert werden (hier *kundennr*).

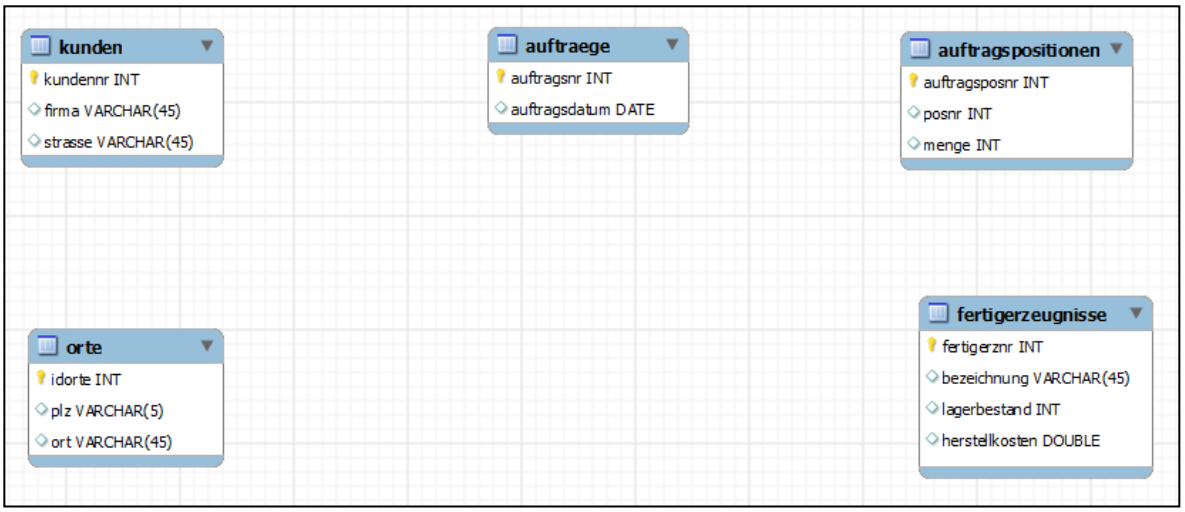
(2) Für den vorgeschlagenen Primärschlüssel ist neben der CheckBox *PK* (*Primary Key)* auch *NN (Not NULL)* ausgewählt. Dies bedeutet, dass das Attribut einen Wert erhalten muss.

(3) Die anderen Attribute werden entsprechend eingegeben. Attributnamen werden in dieser Handreichung klein geschrieben. Für jedes Attribut ist sein Attributtyp (*Datatype*) festzulegen.

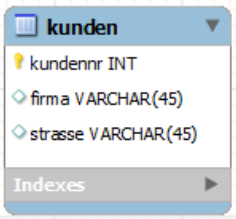
(4) Das Kombinationsfeld "*Data Type*" listet die Datentypen auf, die der jeweilige SQL-Dialekt unterstützt. Für die Felder mit dem Attributtyp *VARCHAR()* wird die vorgeschlagene Länge übernommen - hier *VARCHAR(45).*

(5) Die Datentypen werden auf Seite 10 erläutert.

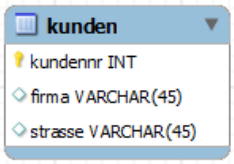
Auf der Grundlage des erarbeiteten Relationenmodells für die Firma MÜCKE & PARTNER OHG (siehe Seite 8) wird die folgende Tabellenstruktur **ohne** die Fremdschlüsselattribute erfasst:



Hinweis: Standardmäßig werden das *ERM (Entity Relationship Model)* bzw. *EERM (Extended Entity Relationship Model)* in der voreingestellten (=default) Notation dargestellt. In dieser Notation erscheint auch die Rubrik *Indexe* (siehe nebenstehende Abbildung).



Mit dem Befehl **`Database – Objekt Notation`** kann bei Auswahl der Option **`Workbench (Simplified)`** die Darstellung der Tabellen ohne die Rubrik Indexe eingestellt werden (siehe nebenstehende Abbildung).



➢ **Herstellen der Beziehungen zwischen den Tabellen *orte* und *kunden***

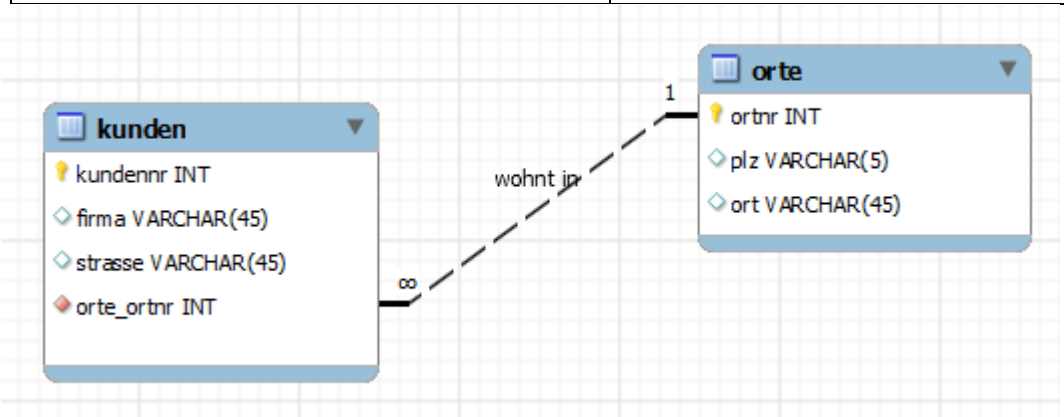
Zwischen den Tabellen *orte* und *kunden* besteht eine konkrete Beziehung:

In einem Ort können mehrere Kunden ihren Geschäftssitz haben, ein bestimmter Kunde hat jedoch nur einen Geschäftssitz. In diesem konkreten Fall ist die Kardinalität der Beziehung also 1:N.

Die Verbindung zwischen zwei Tabellen wird mit Hilfe eines gemeinsamen Attributs hergestellt. Bei den beiden Tabellen *orte* und *kunden* erfolgt die Verbindung über das gemeinsame Attribut *ortnr*.

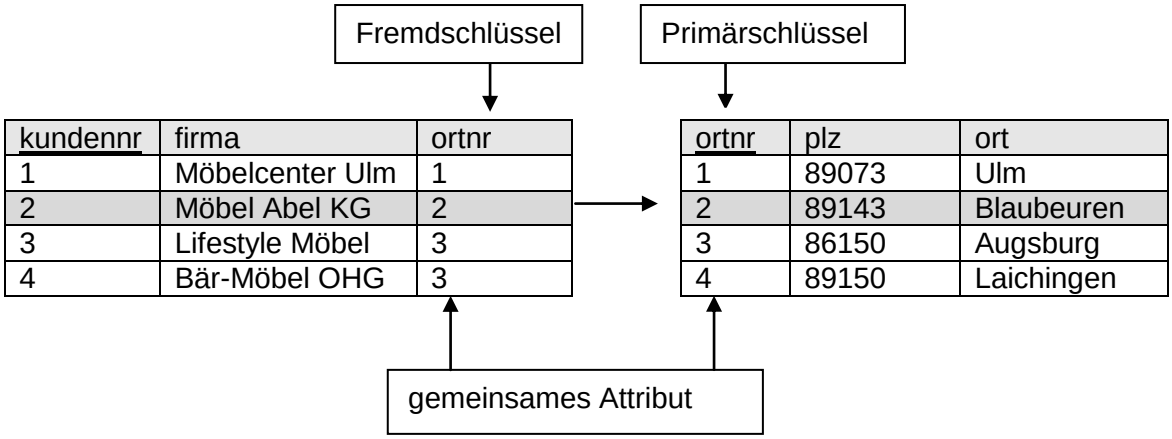
In der Tabelle *kunden* (= Child-Tabelle) wird der Primärschlüssel der Tabelle *orte* (= Parent-Tabelle) als Fremdschlüssel eingerichtet.

Child-Tabelle *kunden*	Parent-Tabelle *orte*



ortnr = Fremdschlüssel	*ortnr* = Primärschlüssel

Dadurch kann eindeutig zugeordnet werden, welcher Kunde seinen Geschäftssitz in welchem Ort hat.



kundennr	firma	ortnr
1	Möbelcenter Ulm	1
2	Möbel Abel KG	2
3	Lifestyle Möbel	3
4	Bär-Möbel OHG	3

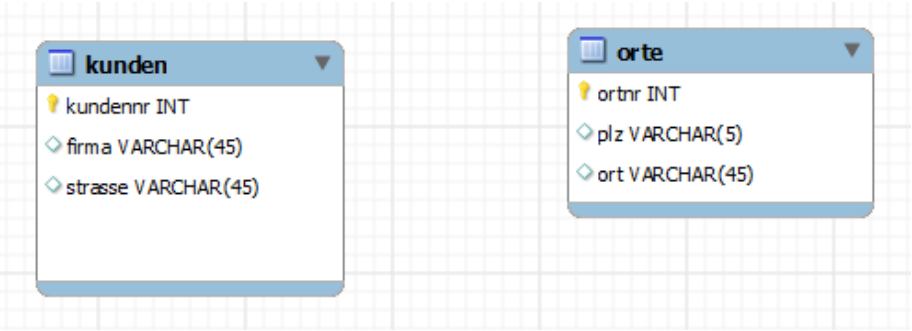
ortnr	plz	ort
1	89073	Ulm
2	89143	Blaubeuren
3	86150	Augsburg
4	89150	Laichingen

Die Beziehung zwischen Tabellen wird im EER-Diagramm hergestellt. Sie können modelliert oder durch die Zuordnung von Primär- und Fremdschlüssel definiert werden.

- **Beziehung im Diagrammfenster modellieren**

  Wenn die Beziehungen im EER-Diagramm modelliert werden, dürfen die Fremdschlüssel in den *Child*-Tabellen noch nicht angelegt sein.

  Für die Einrichtung der Beziehung zwischen der *Parent*-Tabelle o*rte* und der *Child*-Tabelle *kunden* bedeutet dies, dass in der Tabelle *kunden* das Fremdschlüsselattribut *ortnr* (siehe nachfolgende Abbildung) noch nicht eingefügt sein darf.

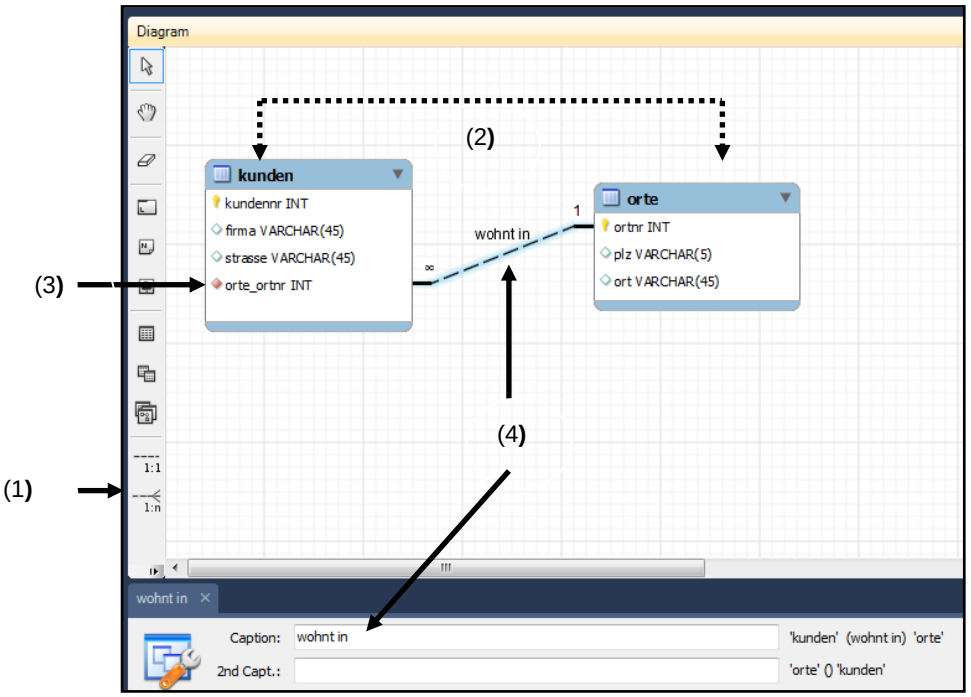


  Die Beziehung wird dann eingerichtet:

  (1) Symbol aus der Werkzeugleiste: **`Place an New 1:n Non-Identifying Relationship 1.`**

  (2) Zuerst die Child-Tabelle kunden und dann die Parent-Tabelle *orte* anklicken.

  (3) Die Beziehung wird angezeigt und der Fremdschlüssel in der Tabelle *kunden* wird eingerichtet. Er erhält standardmäßig den Namen der Parent-Tabelle als Präfix: *orte_ortnr.* Diese Vorgabe kann auch verändert werden.
  In der folgenden Abbildung wurde die Schreibweise (Notation) ***Connect to Columns*** (Menü: **`Model → Relationsip Notation → Conenect to Columns)`** **gewählt**.



[1] Während bei der *Non-Identifying Relationship* in der Child-Tabelle lediglich eine Referenz zur Parent-Tabelle (Fremdschlüssel) eingerichtet wird, wird bei der *Identifying Relationship* der Fremdschlüssel zum Primärschlüssel oder Teil des Primärschlüssels in der Child-Tabelle.

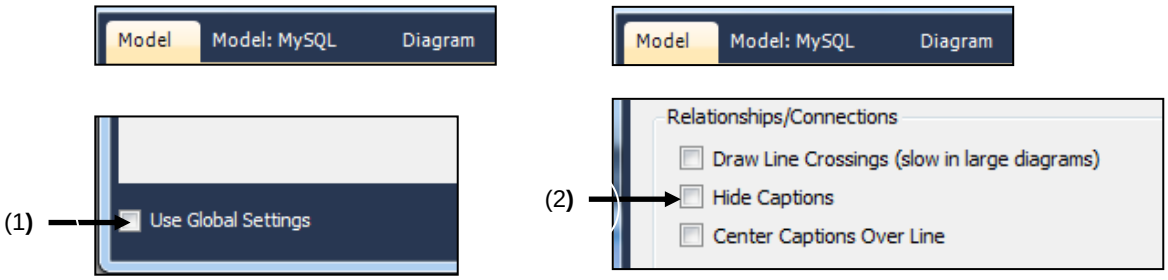
(4) Beziehungstyp beschreiben. Dazu wird die Verbindung markiert und editiert (Doppelklick). Danach kann die Beschriftung im Bezeichnungsfeld *Caption* vorgenommen.

☞ **Hinweis:** Mit dem Menübefehl **`Model → Model Options`** müssen

(1) im Register Model die globalen Einstellungen (Option Use Global Settings***)***

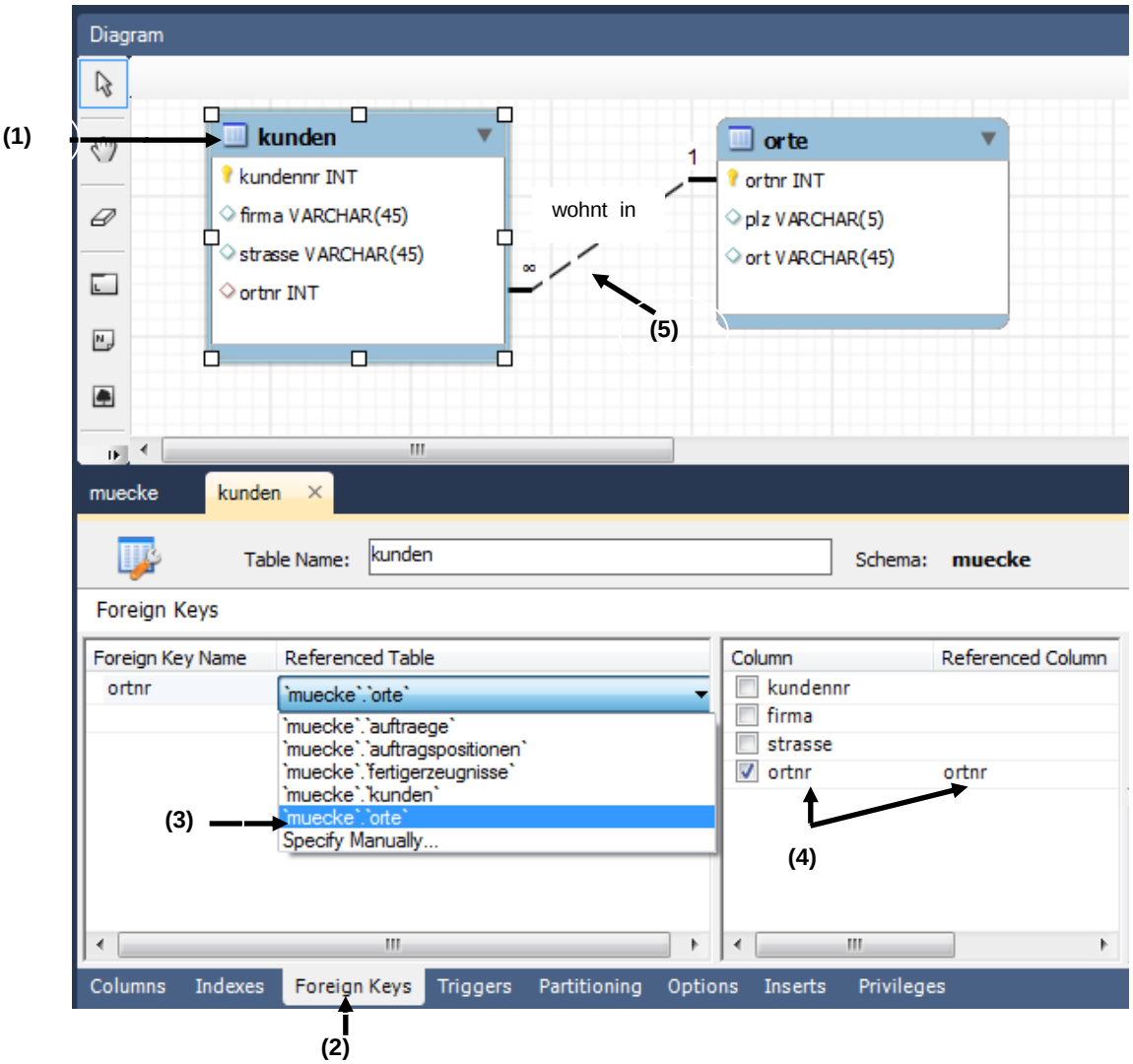
(2) und im Register ***Diagram*** unter ***Relationship Connections*** die Option ***Hide Caption***

deaktiviert werden.



- **Beziehung im Tabellenentwurf definieren**

  Wenn die Primär- und Fremdschlüssel in den zu verbindenden Tabellen wie im Relationenmodell auf Seite 8 beschrieben angelegt sind, kann im Entwurf der Child-Tabelle *kunden* die Verbindung eingerichtet werden.



Vorgehensweise

(1) Die Tabelle *kunden* auswählen (Doppelklick oder Kontextmenü **`Edit Table`**) und das Register **`Foreign Keys`** öffnen.

(2) In der Spalte *Foreign Key Name* den angelegten Fremdschlüssel *ortnr* eintragen.

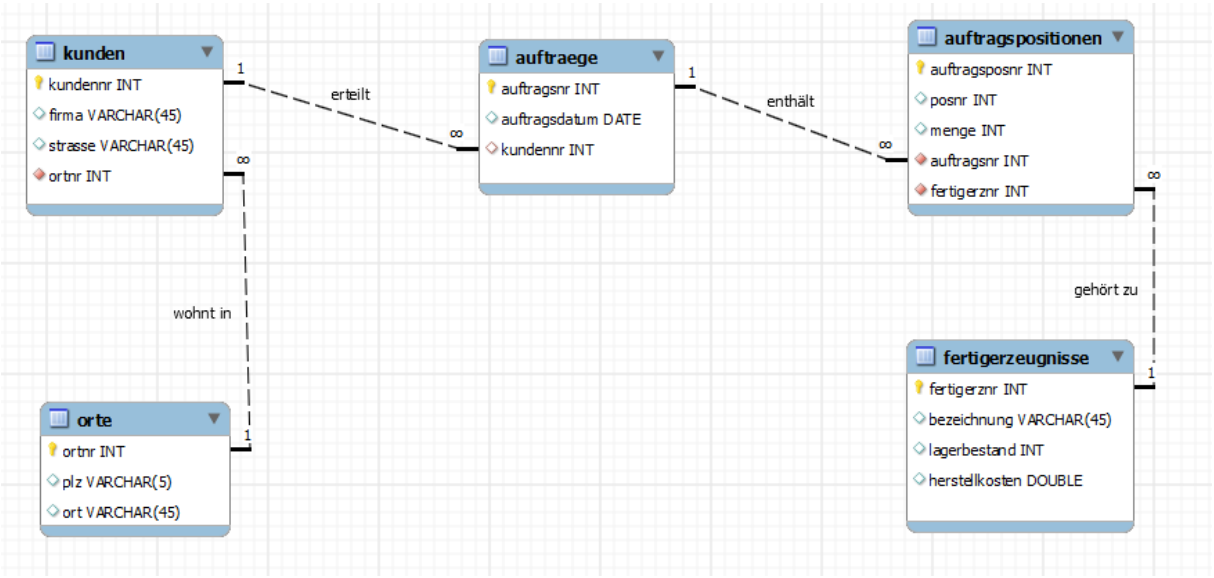
Hinweis: Der Name des Fremdschlüssels muss nicht mit dem Namen des Primärschlüssels der Parent-Tabelle übereinstimmen. Sie müssen jedoch denselben Datentyp und übereinstimmende Attributwerte haben.

(3) In der Spalte *Referenced Table* mit einem Doppelklick ein Listenfeld mit den Tabellen der Datenbank öffnen und daraus die zum Fremdschlüssel gehörende Parent-Tabelle *orte* auswählen.

(4) In der Spalte *Column* den Fremdschlüssel *ortnr* auswählen. Es erscheint dann automatisch der Referenzschlüssel in der Spalte *Referenced Column*.

(5) Die Verbindung wird im Modell angezeigt und kann jetzt noch wie auf Seite 74 beschrieben, beschriftet werden.

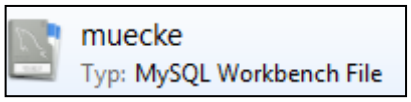
Das fertige EERM in der Notation *Connect to Columns*



➢ **Speichern und Öffnen des Modells**

Das erstellte Modell wird mit dem Befehl

- **`File – Save Model As`** *muecke* auf dem Datenträger gespeichert und erhält dort die Dateierweiterung .mwb;



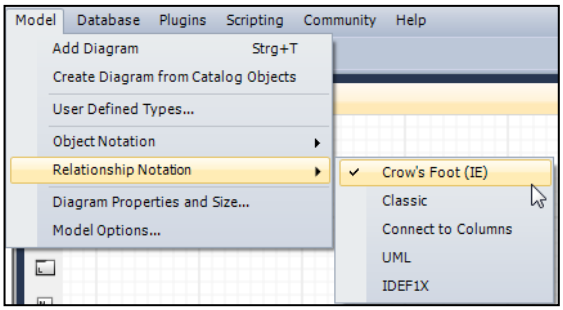
- **`File – Open Model`** wieder geöffnet.

1.1.2 Notationen

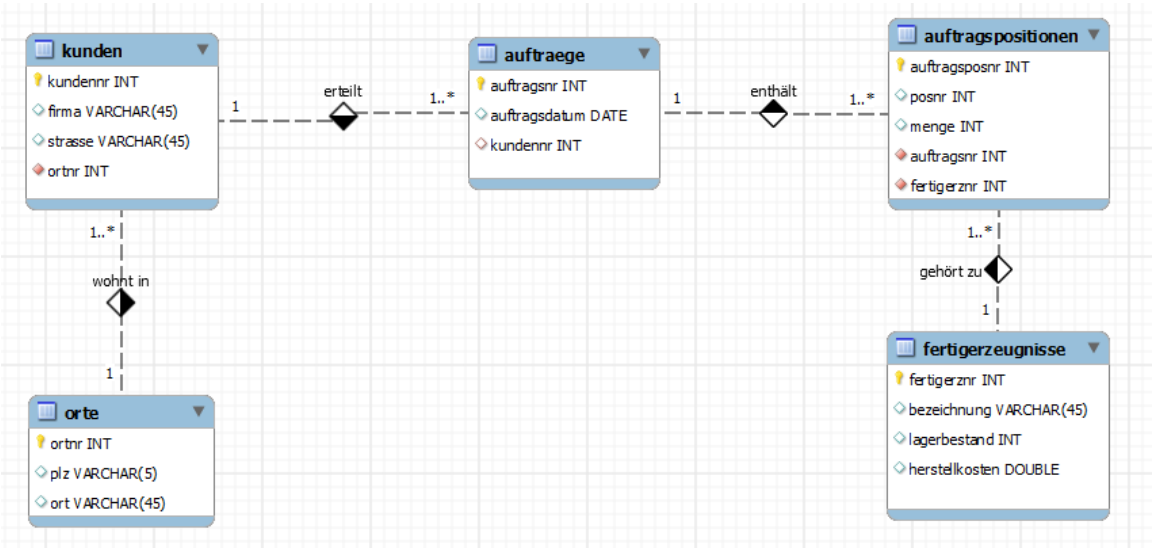
Für die Darstellung der Beziehungen zwischen den Entitätstypen werden verschiedene Darstellungsformen verwendet.

- Classic (klassische Darstellung)
- Krähenfußnotation (Crow's Foot)
- Connect to Columns-Notation
- UML-Notation
- IDEFIX-Notation

Die Notation wird mit dem Menübefehl **`Model – Relationship Notations`** ausgewählt.



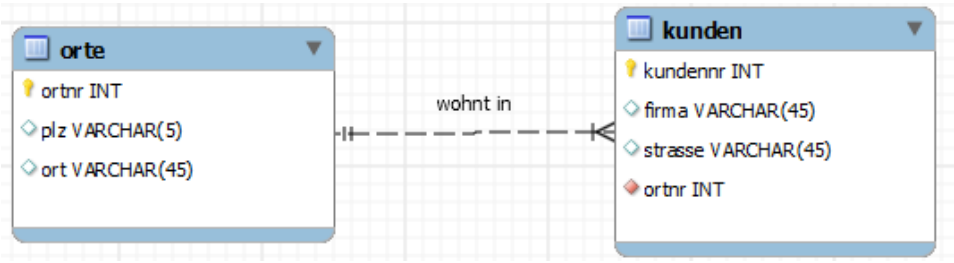
- Die klassische Notation (***Model - Relationship Notations Classic)***



Die klassische Notation ist an die Chen-Notation (nach Peter Chen, Entwickler der ER-Diagramme) angelehnt.

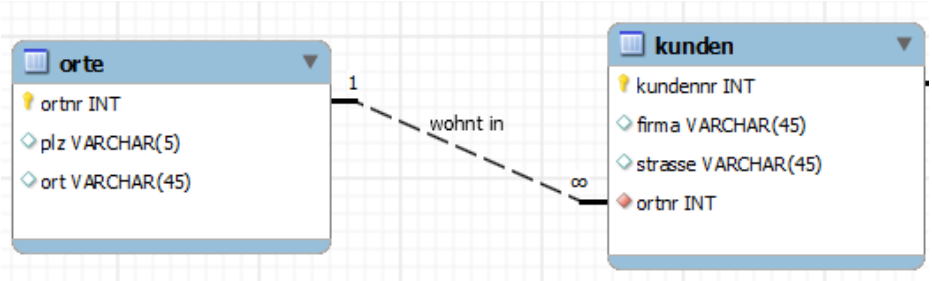
- **Die Martin-Notation (auch Crow's Foot Notation)**

nach James Martin, Bachmann und Odell ist eine Notation zur semantischen Datenmodellierung, um vereinfachte Entity-Relationship-Modelle darzustellen. Sie verwendet für eine 1:n-Beziehung sogenannte Krähenfüße und wird daher auch Krähenfußnotation genannt.



Die Krähenfußnotation ist eine häufig verwendete Darstellungsform der Beziehungen zwischen Entitätstypen.

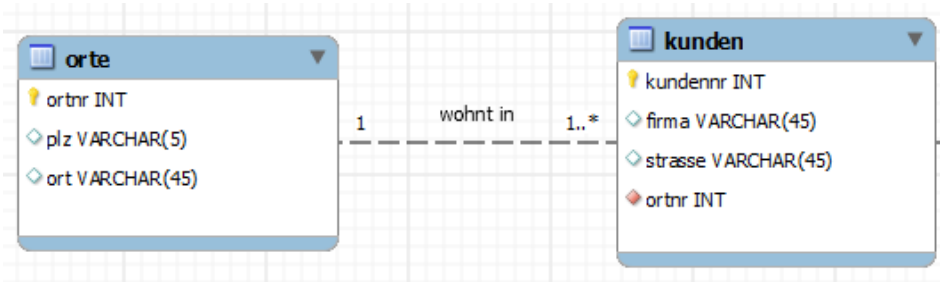
- **Die “Connect to Columns”-Notation**



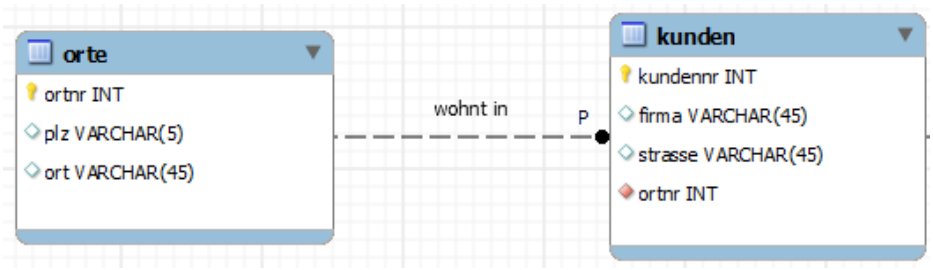
Die Verbindungslinie zeigt die Verbindung der Schlüsselattribute direkt auf.

- **Die “UML”-Notation**

  Unified Modeling Language (UML) ist eine der dominierenden Sprachen für die Softwaresystem-Modellierung.



- **Die IDEF1X-Notation**



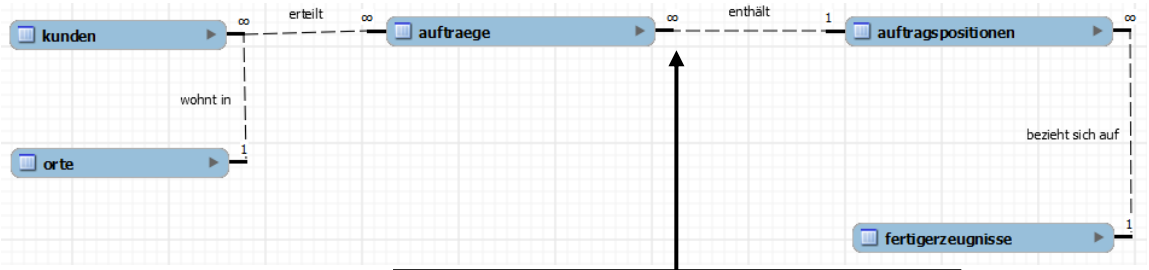
Auf die ausführliche Vorstellung der einzelnen Notationen soll hier nicht eingegangen werden.

In den folgenden Ausführungen wird die Notation ***Connect to Columns*** verwendet:

ERM und EERM

Beim Modellieren der Datenbank mit Hilfe der *MySQL Workbench* können neben den Tabellennamen auch die Attribute mit den Datentypen erfasst werden. Als Modellierungsergebnis erhält man ein erweitertes Entity Relationship Model (*Extended Entity Relationship Model = EERM)*. Im *EERM* sind somit das *ERM* und das Relationenmodell zusammengefasst.

Soll lediglich das ERM angezeigt werden, können die Attribute ausgeblendet werden.



Mit einem Klick können die Attribute ein- beziehungsweise ausgeblendet werden.

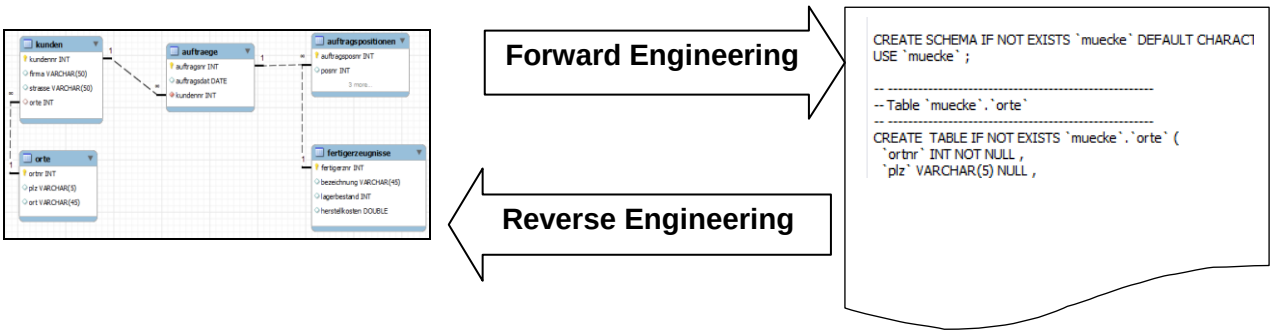
1.2 Vom EERM zum Datenbankskript (=Forward Engineering)

Problemstellung:

Auf der Grundlage des Entity Relationship Modells und des Relationenmodells (siehe Seite 8) beziehungsweise mit Hilfe des erweiterten Entity Relationship Modells (*EERM*) (siehe Seite 18), soll die Datenbank mit den Tabellen jetzt auf dem Datenbankserver eingerichtet (implementiert) werden.

Dazu muss zunächst das Datenbankmodell in einer für den Datenbankserver verständlichen Sprache formuliert werden. Man spricht hierbei von **Forward Engineering**. Die Kommunikation mit dem Datenbankserver erfolgt mit der Abfragesprache ***SQL***, im Besonderen mit den Befehlen der ***Data Definition Language (DDL).***
Das erstellte Skript mit den SQL-Anweisungen zum Erzeugen der Datenbank wird dann auf dem Datenbankserver ausgeführt.

Den umgekehrten Weg von der fertigen Datenbank zum ursprünglichen Modell bezeichnet man als **Reverse Engineering.**



Voraussetzung

Als Arbeitsumgebung werden der *MySQL*-Datenbankserver aus dem *XAMPP*-Paket sowie als Frontend die *MySQL Workbench* eingesetzt.

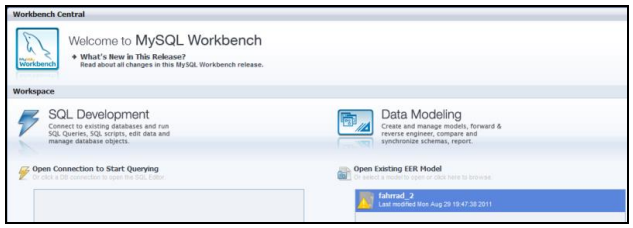
1. Datenbankserver starten

   Control-Panel und MySQL starten

   xampp-control

2. MySQL Workbench starten

   MySQLWorkbench.exe



Methodischer Hinweis:

Die modellierte Datenbank kann grundsätzlich auf unterschiedlichen Wegen auf dem Datenbankserver eingerichtet werden:

- Übertragung des gesamten *EERM* auf den Datenbankserver mit Hilfe des *Forward Engineering* der *MySQL Workbench*.

- Mit Hilfe des *SQL File Editors*. Dabei werden die Anweisungen der *DDL (Database Definition Language)* im SQL File Editor eingegeben oder mithilfe der Workbench generiert und ausgeführt. So werden zunächst die Datenbank, danach die einzelnen Tabellen und daran anschließend die Beziehungen zwischen den Tabellen erzeugt.

1.2.1 Implementieren des gesamten Modells auf dem Server

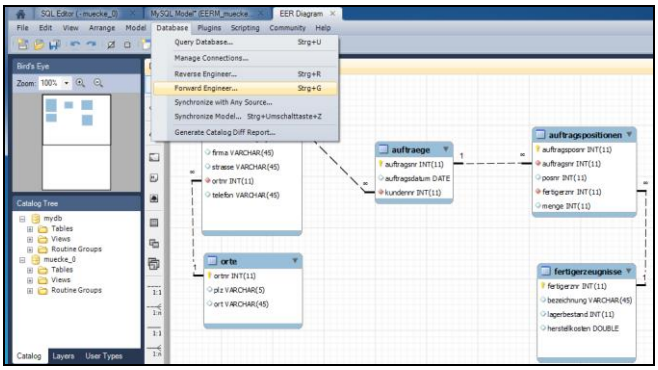
Aufgabenstellung:
Das fertig modellierte EERM zu der Firma Muecke&Partner OHG (siehe Seite 18) soll auf dem MySQL-Datenbankserver eingerichtet werden.

Um die Datenbank auf dem MySQL-Datenbankserver einzurichten, muss zunächst das Skript mit den dafür benötigten SQL-Anweisungen erzeugt werden. Dieses Skript wird dann auf dem Datenbankserver ausgeführt.

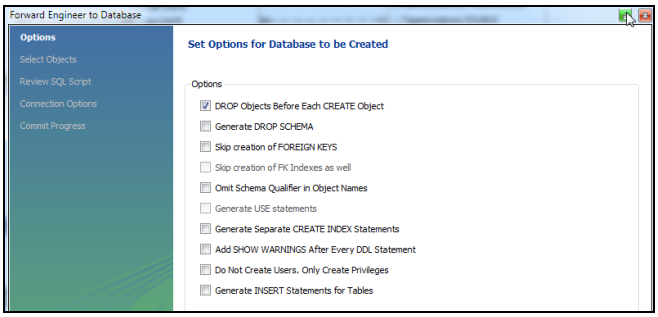
Vorgehensweise

Screenshots

➢ Aus dem EER-Diagramm heraus den Assistenten mit dem Menü **`Database→Forward Engineering`** aufrufen.

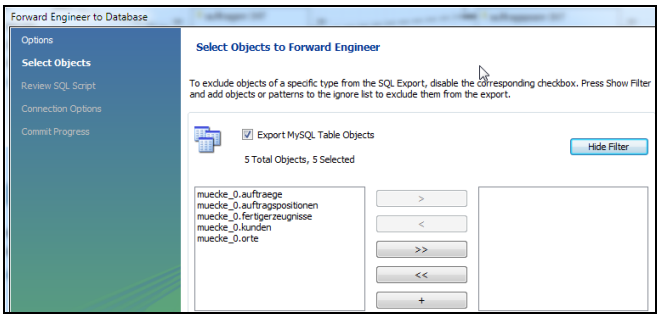


➢ Im Fenster *Options* wird die Option *Drop Objects Before…* aktiviert. Dadurch werden schon vorhandene Objekte (z. B. Tabellen) ggf. gelöscht, bevor sie neu angelegt werden.

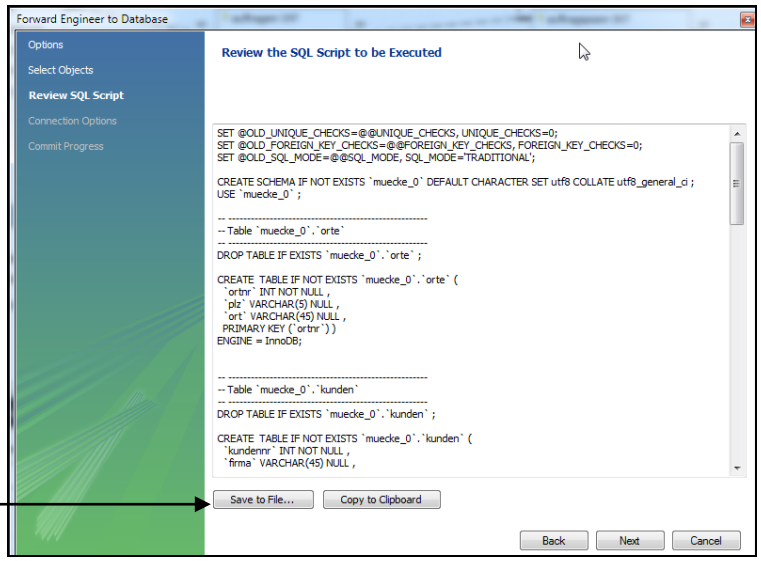


➢ Im Fenster Select Objects wird mit dem Button [**`Show Filter`**]

… überprüft beziehungsweise korrigiert, dass nur die Tabellen implementiert werden, die zu der Datenbank gehören. Tabellen, die nicht berücksichtigt werden sollen, werden markiert und in das rechte Feld verschoben.



- Das Fenster *Review SQL-Script* zeigt das auszuführende SQL-Skript.



Mit Hilfe des Buttons [**Save to File…**] kann das generierte SQL-Skript als Textdokument gespeichert oder

mit dem Button **[Copy to Clipboard]** als Bilddatei in die Zwischenablage übernommen werden.

Nachfolgend ist das erzeugte SQL-Script dargestellt und erläutert:

```
CREATE SCHEMA IF NOT EXISTS `muecke` DEFAULT CHARACTER SET utf8 COLLATE
utf8_general_ci ;
USE `muecke` ;


-- -----------------------------------------------------
-- Table `muecke`.`orte`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `muecke`.`orte` ;

CREATE  TABLE IF NOT EXISTS `muecke`.`orte` (
  `ortnr` INT NOT NULL ,
  `plz` VARCHAR(5) NULL ,
  `ort` VARCHAR(45) NULL ,
  PRIMARY KEY (`ortnr`) )
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `muecke`.`kunden`
-- -----------------------------------------------------
```

```
DROP TABLE IF EXISTS `muecke`.`kunden` ;

CREATE  TABLE IF NOT EXISTS `muecke`.`kunden` (
  `kundennr` INT NOT NULL ,
  `firma` VARCHAR(45) NULL ,
  `strasse` VARCHAR(45) NULL ,
  `ortnr` INT NOT NULL ,
  PRIMARY KEY (`kundennr`) ,
  INDEX `fk_kunden_orte` (`ortnr` ASC) ,
  CONSTRAINT `fk_kunden_orte`
    FOREIGN KEY (`ortnr` )
    REFERENCES `muecke`.`orte` (`ortnr` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Tabelle erzeugen

Attribute anlegen

Primärschlüssel anlegen

Beziehung herstellen

Tabellentyp *InnoDB* für die referentielle Integrität festlegen

```
-- -----------------------------------------------------
-- Table `muecke`.`auftraege`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `muecke`.`auftraege` ;

CREATE  TABLE IF NOT EXISTS `muecke`.`auftraege` (
  `auftragsnr` INT NOT NULL ,
  `auftragsdatum` DATE NULL ,
  `kundennr` INT NULL ,
  PRIMARY KEY (`auftragsnr`) ,
  INDEX `kundennr` (`kundennr` ASC) ,
  CONSTRAINT `kundennr`
    FOREIGN KEY (`kundennr` )
    REFERENCES `muecke`.`kunden` (`kundennr` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `muecke`.`fertigerzeugnisse`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `muecke`.`fertigerzeugnisse` ;

CREATE  TABLE IF NOT EXISTS `muecke`.`fertigerzeugnisse` (
  `fertigerznr` INT NOT NULL ,
  `bezeichnung` VARCHAR(45) NULL ,
  `lagerbestand` INT NULL ,
  `herstellkosten` DOUBLE NULL ,
  PRIMARY KEY (`fertigerznr`) )
ENGINE = InnoDB;
```

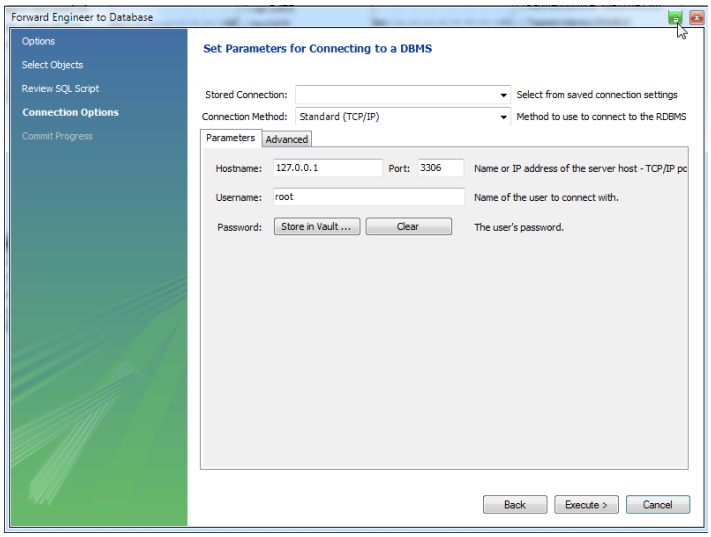
```
-- -----------------------------------------------------
-- Table `muecke`.`auftragspositionen`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `muecke`.`auftragspositionen` ;

CREATE  TABLE IF NOT EXISTS `muecke`.`auftragspositionen` (
  `auftragsposnr` INT NOT NULL ,
  `posnr` INT NULL ,
  `menge` INT NULL ,
  `auftragsnr` INT NOT NULL ,
  `fertigerznr` INT NOT NULL ,
  PRIMARY KEY (`auftragsposnr`) ,
  INDEX `fk_auftragspositionen_auftraege1` (`auftragsnr` ASC) ,
  INDEX `fk_auftragspositionen_fertigerzeugnisse1` (`fertigerznr` ASC) ,
  CONSTRAINT `fk_auftragspositionen_auftraege1`
    FOREIGN KEY (`auftragsnr` )
    REFERENCES `muecke`.`auftraege` (`auftragsnr` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_auftragspositionen_fertigerzeugnisse1`
    FOREIGN KEY (`fertigerznr` )
    REFERENCES `muecke`.`fertigerzeugnisse` (`fertigerznr` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Nun wird das erzeugte Skript auf dem MySQL-Datenbankserver ausgeführt.

- Im Fenster *Connections Options* werden die Parameter zur Herstellung der Verbindung zum Server angezeigt. In der Regel passen die vorgegebenen Parameter für die lokale Umgebung und müssen nicht verändert werden.

- Mit dem Button [**Execute…**] wird das Skript ausgeführt.

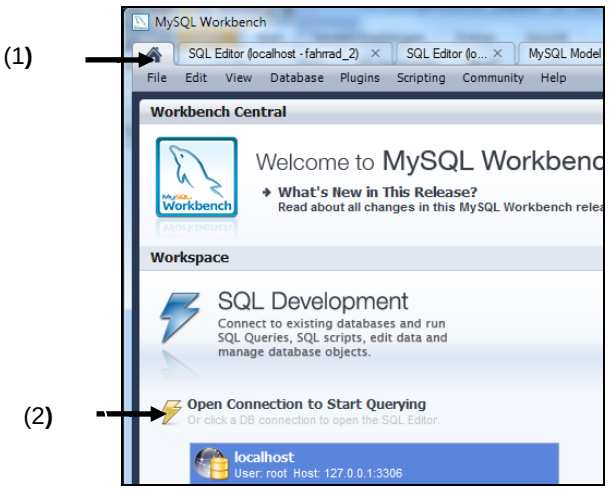


Anschließend kann die Datenbankmodellierung beendet und die Verbindung zur neu eingerichteten Datenbank hergestellt werden.

Dazu wird

(1) das Symbol Home gewählt und

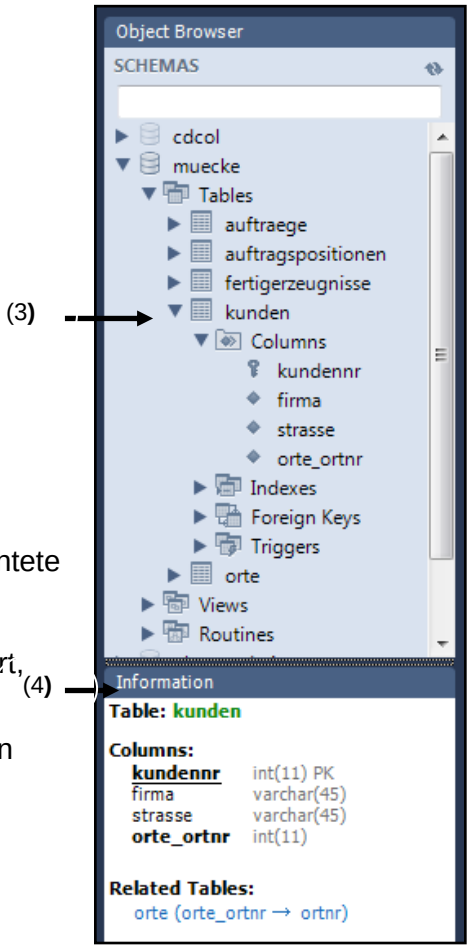
(2) die Datenbankverbindung geöffnet.



Im Fenster des *Object Browser* erscheint dann die eingerichtete Datenbank (hier *muecke*) mit den Tabellen.

(3) Wird im *Object Browser* der Tabellennamen erweitert, können die Attribute sichtbar gemacht und

(4) im Fensterbereich *Information* weitere Eigenschaften (Datentypen) angezeigt werden.

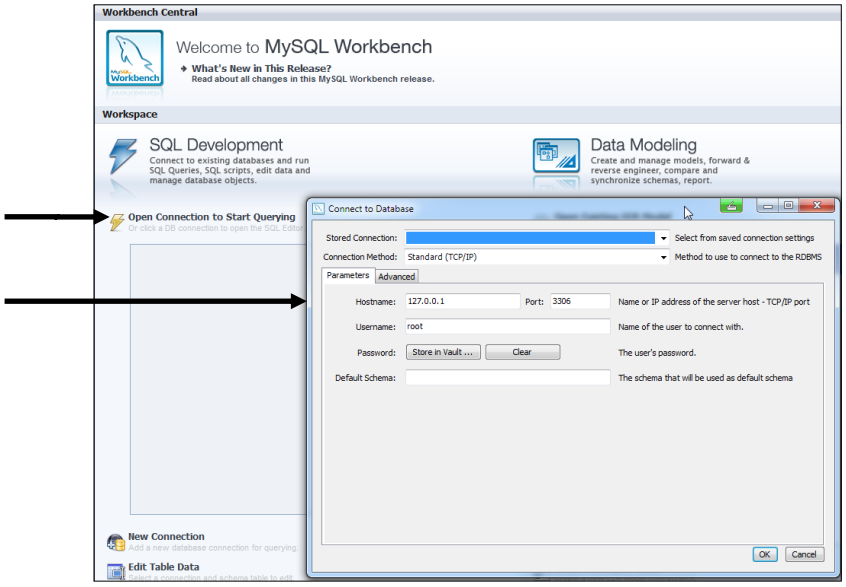


1.2.2 Einrichten einer Datenbank mit Hilfe der MySQL Workbench

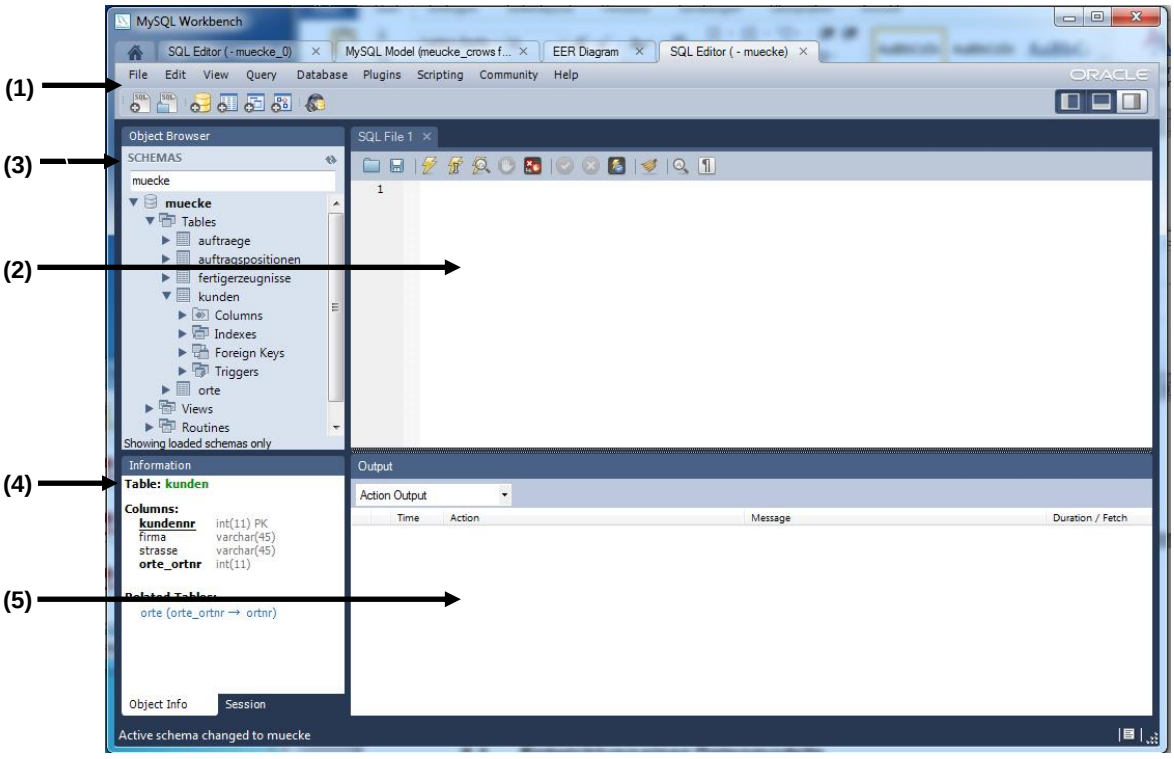
Bei dieser Vorgehensweise wird die Datenbank mit Hilfe des Datenbank-Frontends *MySQL Workbench* eingerichtet.

Dazu

- die Workbench gestartet und
- die Verbindung zum Server hergestellt

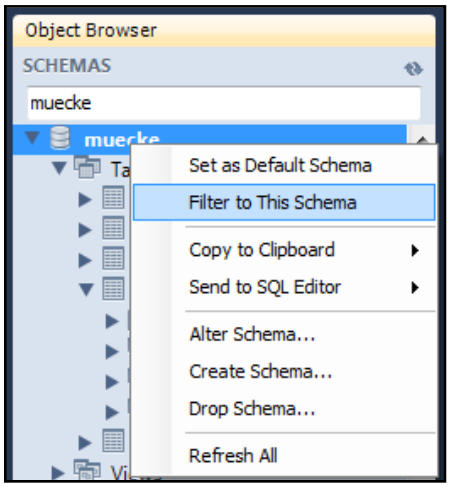


Ausgangsbildschirm der MySQL Workbench



(1) Menü- und Symbolleiste.

(2) Fenster des SQL File Editors für die Eingabe von SQL-Befehlen.



(3) Die angelegten Datenbanken (Schemas) werden angezeigt. Öffnet man zu einer Datenbank das Kontextmenü (rechte Maustaste), kann mit der Option **`Filter to this Schema`** die Anzeige im **Objekt Browser** auf eine Datenbank begrenzt werden (siehe nebenstehende Abbildung). Mit der Option **`Set As Default Schema`** oder mit einem Doppelklick auf das Schema wird die die betreffende Datenbank geöffnet und alle Befehle, die im Fenster des SQL-Editors eingegeben werden, beziehen sich auf diese Datenbank.

(4) Informationen zu der gerade aktivierten Tabelle (hier *kunden*).

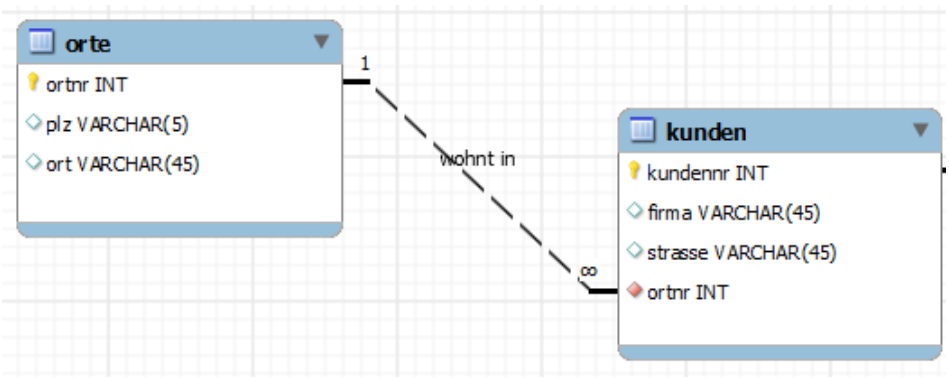
(5) Bereich, in dem Meldungen und Fehlermeldungen angezeigt werden.

Um die Arbeitsweise mit dem *SQL File Editor* zu demonstrieren, sollen aus dem ERM und Relationenmodell zur Datenbank der Firma MÜCKE & PARTNER OHG (siehe Seite 8 beziehungsweise Seite 18) die beiden Entitätstypen *kunde* und *ort* in einer neuen Datenbank mit Namen muecke_n angelegt werden.

Es soll also für die MÜCKE & PARTNER OHG

- die Datenbank *muecke_n*
- mit den Tabellen *kunden* und *orte*

erzeugt und auf dem Datenbankserver eingerichtet werden (siehe nachfolgender Ausschnitt aus dem ERM).



1.2.2.1 Anlegen und Löschen der Datenbank

Anlegen der Datenbank

Bevor Tabellenstrukturen erzeugt und Werte in die Datenbanktabellen eingetragen werden können, muss die Datenbank angelegt werden.

Syntax: CREATE DATABASE *<datenbankname>*;

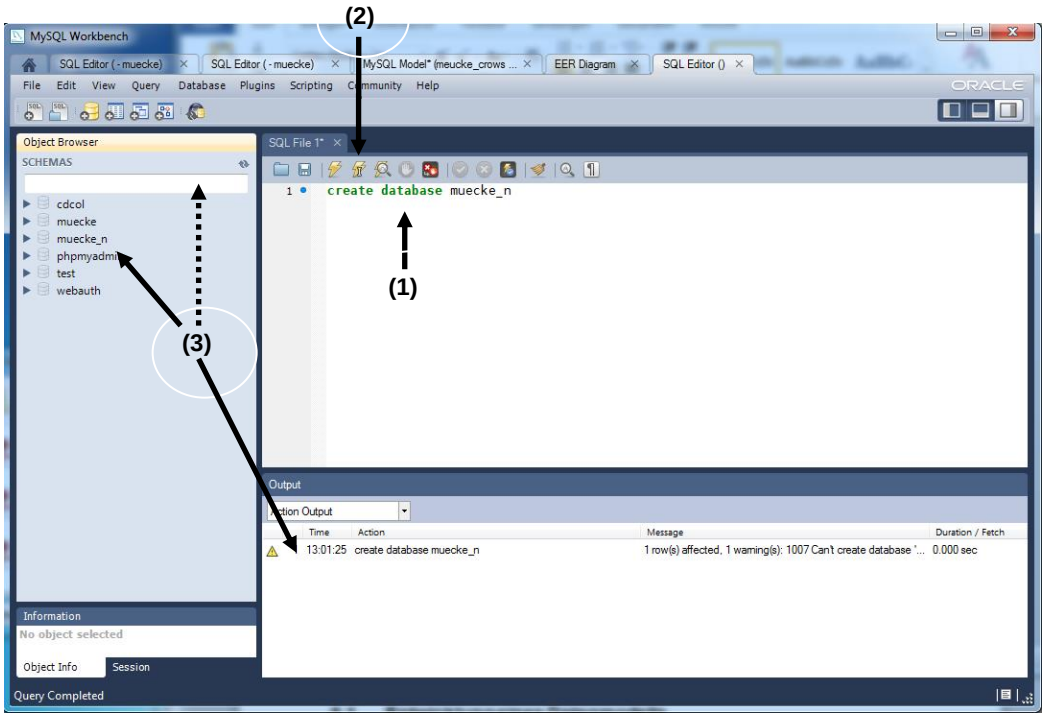
Löschen der Datenbank

Syntax: DROP DATABASE *<datenbankname>*;

Merke: Jedes SQL-Statement endet mit einem Semikolon.

SQL-Anweisungen können direkt im SQL File Editor eingegeben und ausgeführt oder mit Hilfe des Workbench-Assistenten im Dialog aufgebaut und dann ausgeführt werden.

- **Eingabe und Ausführen der SQL-Befehle im SQL File Editor**

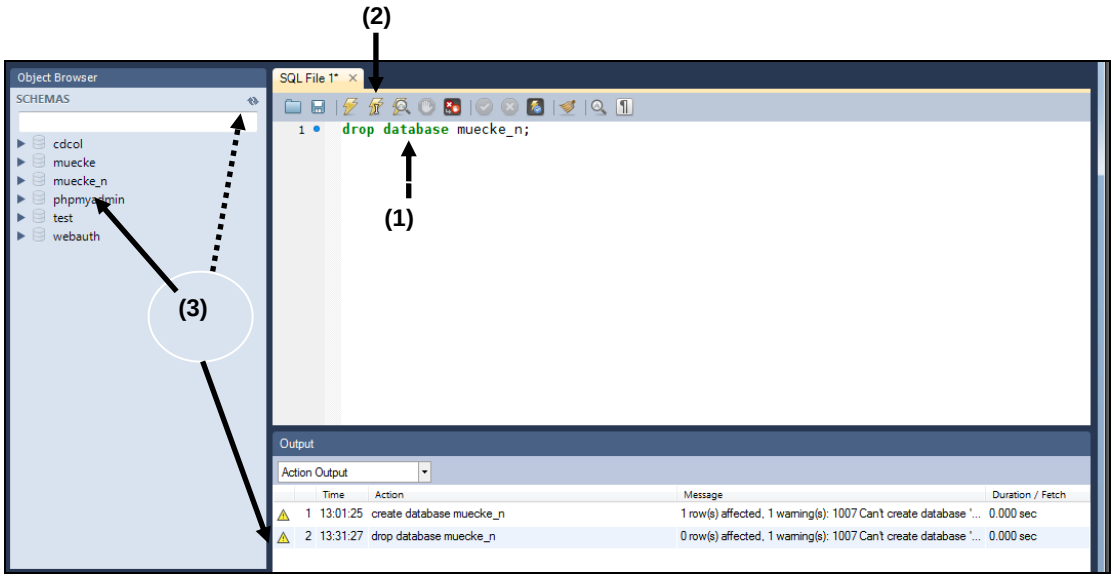


(1) Befehl im Fenster des SQL File Editors eingeben.

(2) SQL-Anweisung ausführen.
Hinweis: Die Schaltfläche führt den aktuellen SQL-Befehl aus.
Die Schaltfläche führt alle SQL-Befehle im Fenster des SQL File Editors aus.

(3) Ansicht im *Object Browser* aktualisieren. (Kontextmenü → **`Refresh All`**)
Das angelegte Datenbankschema erscheint in der Liste der angelegten Datenbanken (Schemas).

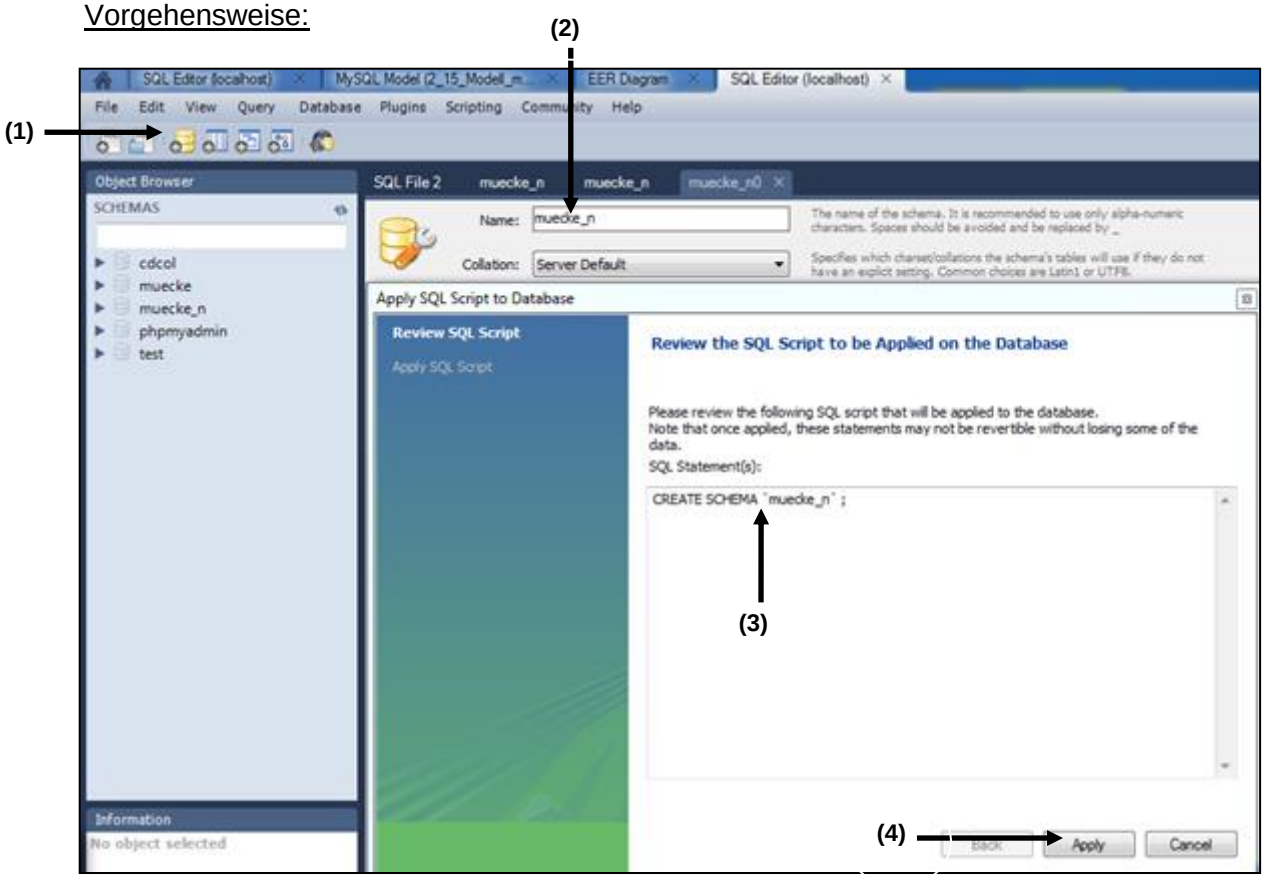
Anschließend soll die Datenbank wieder vom Server gelöscht werden. Dabei werden dieselben Schritte durchgeführt.



- **Eingabe und Ausführen der SQL-Befehle mit Hilfe des Assistenten**

  Die MySQL Workbench hält ein komfortables Assistenten-Tool bereit, mit dem SQL-Anweisungen generiert oder eingegeben werden können. Dadurch können Schreibfehler vermindert werden.

  <u>Vorgehensweise:</u>



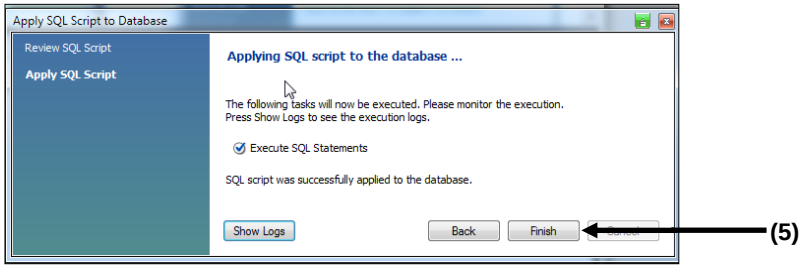
(1) Im Fenster des *Object Browsers* das Symbol **`Create a New Schema`**  wählen.

(2) Es wird ein neues Fenster im SQL File Editor mit dem Namen „*new_schema*“ erzeugt. Hier wird der Namen der Datenbank (hier *muecke_n*) eingegeben und mit dem Button **`[Apply]`** bestätigt.

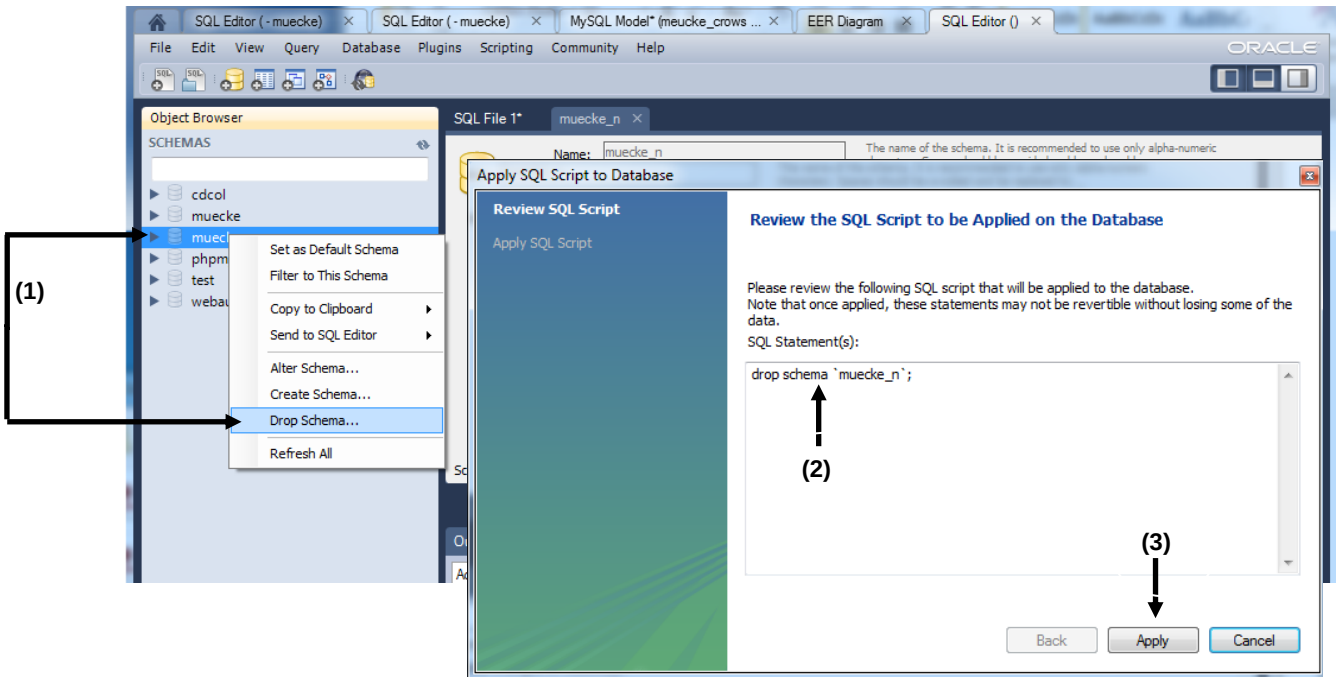
(3) Es erscheint dann das Fenster *Apply SQL Script to Database* mit dem generierten SQL-Befehl zum Erzeugen der Datenbank. Bestandteile der SQL-Anweisung, die nicht zur Befehlssyntax gehören, erscheinen zwischen Hochkomma.

(4) Mit dem Button **`[Apply ]`** des *Fensters Apply SQL Script to Database* wird die SQL-Anweisung ausgeführt.

(5) Abschließend wird mit dem Button [**`Finish`**] der Vorgang beendet und die angelegte Datenbank *muecke_n* erscheint in der Auflistung der angelegten Datenbanken.



Anschließend kann die Datenbank wieder vom Server gelöscht werden:



(1) Kontextmenü (rechte Maustaste zur Datenbank *muecke_n* öffnen und den Befehl *Drop Schema…* auswählen.

(2) Im Fenster *Apply SQL Script to Database* erscheint der generierte SQL-Befehl.

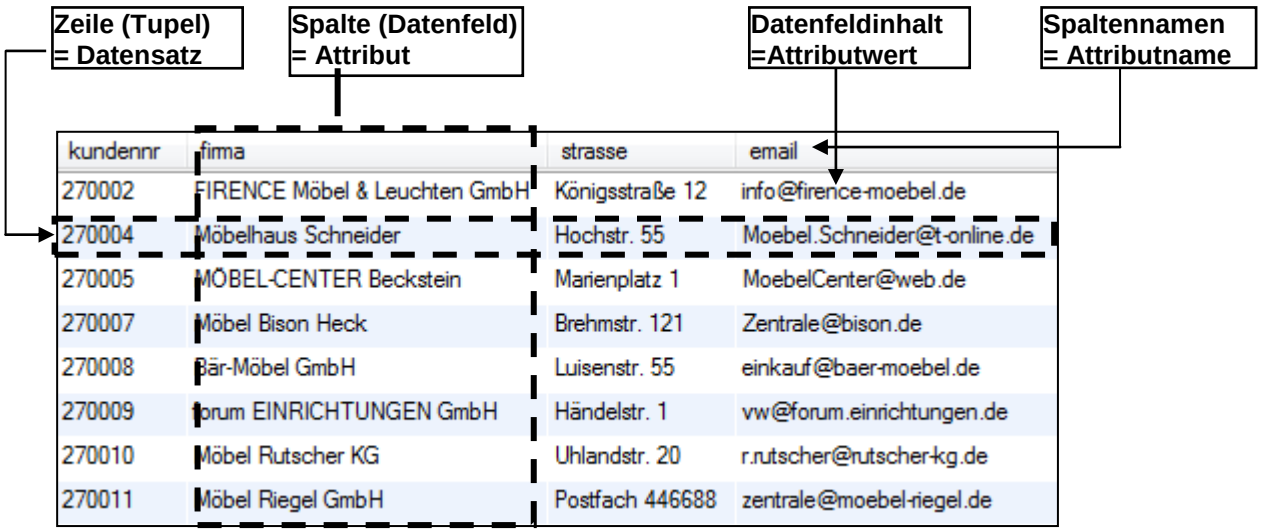
(3) Mit dem Button [**`Apply`**] des *Fensters Apply SQL Script to Database* wird die SQL-Anweisung ausgeführt.

(4) Abschließend wird mit dem Button [**`Finish`**] der Vorgang beendet und die Datenbank *muecke_n* aus dem Fenster des *Object Browsers* entfernt.

1.2.2.2 Erzeugen, Ändern und Löschen von Datenbanktabellen

Die Kunden der Mücke & Partner OHG sollen in der **Datenbank** *muecke_n* und dort in der **Tabelle** *kunden* gespeichert werden.

Für die weiteren Ausführungen werden nachfolgend die Bestandteile (Aufbau) einer Datenbanktabelle vorgestellt:



kundennr	firma	strasse	email
270002	FIRENCE Möbel & Leuchten GmbH	Königsstraße 12	info@firence-moebel.de
270004	Möbelhaus Schneider	Hochstr. 55	Moebel.Schneider@t-online.de
270005	MÖBEL-CENTER Beckstein	Marienplatz 1	MoebelCenter@web.de
270007	Möbel Bison Heck	Brehmstr. 121	Zentrale@bison.de
270008	Bär-Möbel GmbH	Luisenstr. 55	einkauf@baer-moebel.de
270009	forum EINRICHTUNGEN GmbH	Händelstr. 1	vw@forum.einrichtungen.de
270010	Möbel Rutscher KG	Uhlandstr. 20	r.rutscher@rutscher-kg.de
270011	Möbel Riegel GmbH	Postfach 446688	zentrale@moebel-riegel.de

Elemente einer Tabelle

⇒ Eine Tabelle besteht aus **Zeilen und Spalten.**
⇒ Eine Zeile (= ein **Datensatz / Tupel**) enthält die Daten eines Kunden. Alle Zeilen sind gleich aufgebaut.
⇒ Die Identifizierung eines Datensatzes erfolgt über einen eindeutigen **Primärschlüssel** (Zeilenname, hier die *kundennr*).
⇒ Die Identifizierung einer Spalte erfolgt über einen eindeutigen Spaltennamen **(=Attributname**). Jede Spalte enthält einen bestimmten Typ von Attributen (z. B. Zahl, Text, Datum).

Die Bedeutung des Primärschlüssels

Wird für ein Datenfeld (=Attribut) die **Eigenschaft des Primärschlüssels** festgelegt, kann mit Hilfe dieses Datenfeldes der einzelne Datensatz **eindeutig identifiziert** werden. Dies hat zur Folge, dass

- jede Kundennummer nur einmal vorkommen kann,
- die Auflistung der Datensätze nach dem Primärschlüsselfeld (*kundennr*) sortiert ist.

☞ **Hinweis:** Wie auf den Seiten ff. beschrieben, ist es möglich Datenbank-operationen mit Hilfe von SQL-Anweisungen im SQL File Editor einzugeben und auszuführen oder den Assistenten der Workbench zu nutzen um die Anweisungen teilweise oder ganz zu generieren und dann auszuführen.
In den folgenden Ausführungen werden beide Vorgehensweisen angewandt.

☞ **Vorbereitung:** Zunächst ist die Datenbank mit Namen muecke_n wieder wie auf den Seiten 29 beschrieben, zu erstellen.

➢ Erstellen von Datenbanktabellen

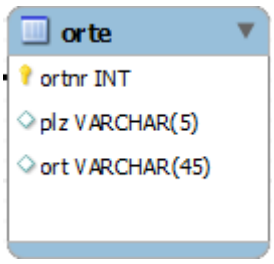
Das Erstellen von Tabellen erfolgt mit der *CREATE TABLE*- Anweisung:

Syntax: CREATE TABLE <*tabellenname*>(<*attributbezeichnung*> <*datentyp*>, <*attributbezeichnung*> <*datentyp*>, PRIMARY KEY (<*attributbezeichnung*>), FORREIGN KEY (<*attributbezeichnung*>)
REFERENCES <tabellenname>(<*attributbezeichnung*>))
[ENGINE = InnoDB];

Merke: Jede Datenbanktabelle benötigt ein Attribut, das der eindeutigen Identifikation eines Datensatzes in einer Tabelle dient. Dieses Attribut wird Primärschlüssel bezeichnet. Der Inhalt des Primärschlüssels darf sich in den anderen Zeilen nicht wiederholen (eindeutiger Schlüssel).
Der Primärschlüssel wird innerhalb der *CREATE TABLE* - Anweisung mit dem Ausdruck *PRIMARY KEY* (*<attributbezeichnung>)* definiert.

Um in *MySQL* beim Erstellen von Tabellen *referentielle Integrität* zwischen diesen Tabellen zu gewährleisten, müssen die Tabellen im Tabellentyp *InnoDB* erzeugt werden. Die InnoDB-Engie beitet sichere Schreib-/Lesezugriffe auf Tabellen. (Alternative Schreibweise: *TYPE = InnoDB*;)

- **Die Tabelle *orte* mit dem SQL File Editor erstellen**



In der Datenbank *muecke_n* soll die Tabelle *orte* mit den Attributen, wie im nebenstehenden EERM-Auszug dargestellt, erzeugt werden. Geben Sie den dafür notwendigen SQL-Befehl an.

Lösung:

```
CREATE TABLE orte (
   ortnr INT NOT NULL,
   plz VARCHAR(5) NULL,
   ort VARCHAR(45) NULL,
   PRIMARY KEY (ortnr))
ENGINE = InnoDB;
```

Tabelle erzeugen

Attribute und Datentypen anlegen

Primärschlüssel festlegen

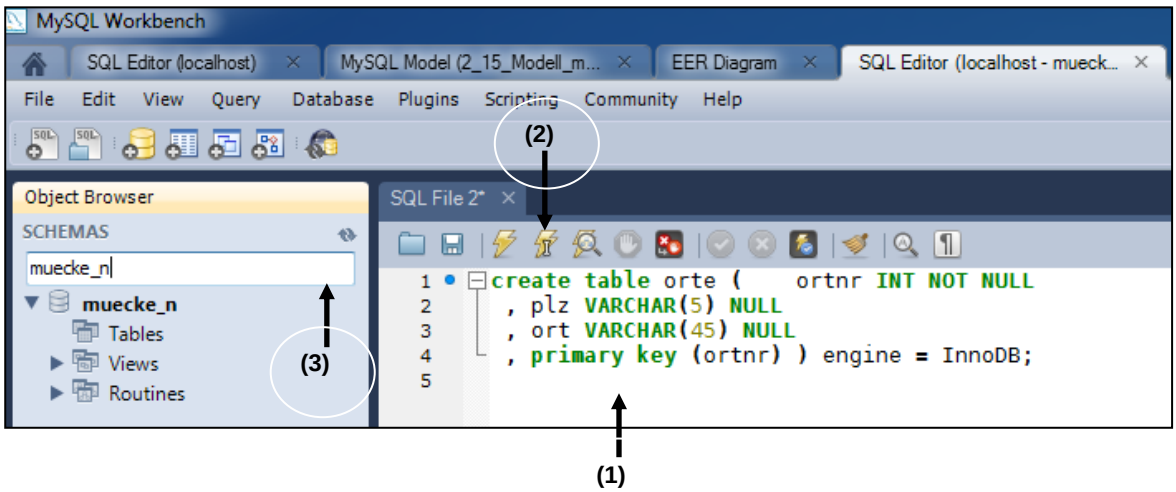
Tabellentyp definieren (Gewährleistung der referentiellen Integrität)

Diese SQL-Anweisung wird im Fenster SQL File Editor

(1) eingegeben und

(2) ausgeführt.

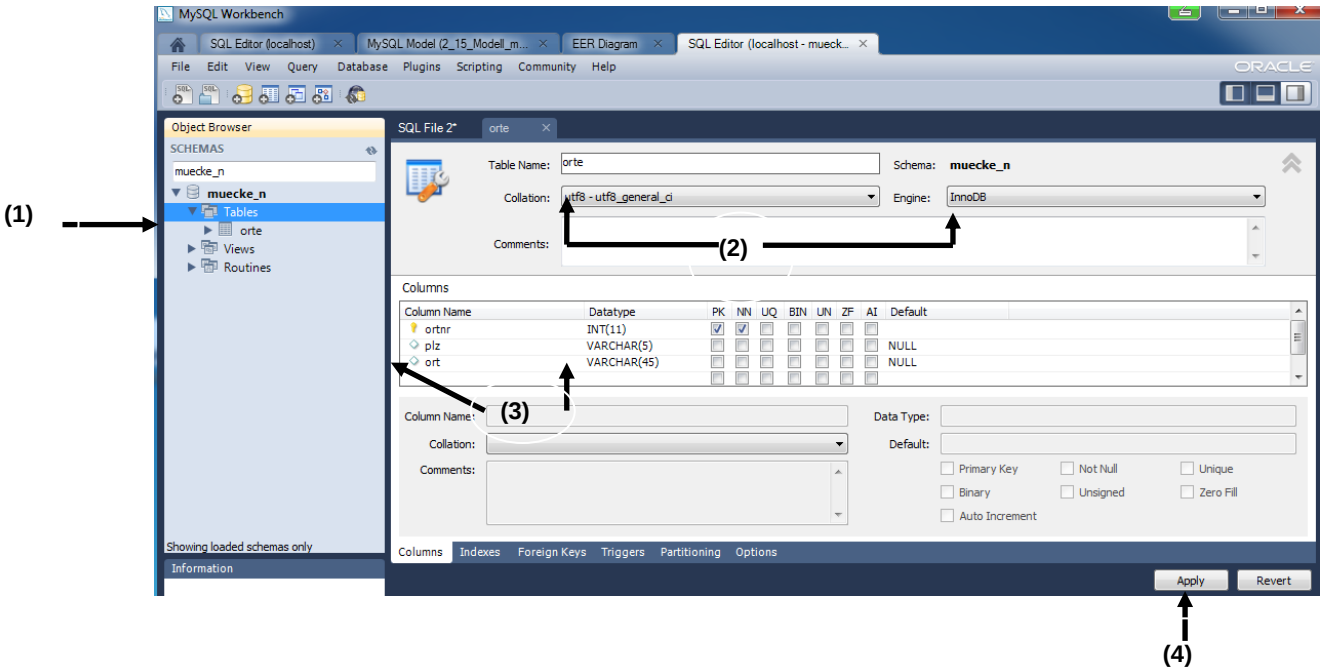
(3) Aktualisiert man anschließend das Fenster des *Object Browsers*, wird die angelegte Tabelle angezeigt.



- **Die Tabelle *orte* mit Hilfe des MySQL Workbench-Assistenten erstellen**

Zuerst wird die Tabelle erzeugt. Dazu wird

(1) im Object Browser die Datenbank (Schema) muecke_n auswählen (Doppelklick) und dann im Ordner *Tables* über das Kontextmenü der Befehl **`Create Table`** aufgerufen.



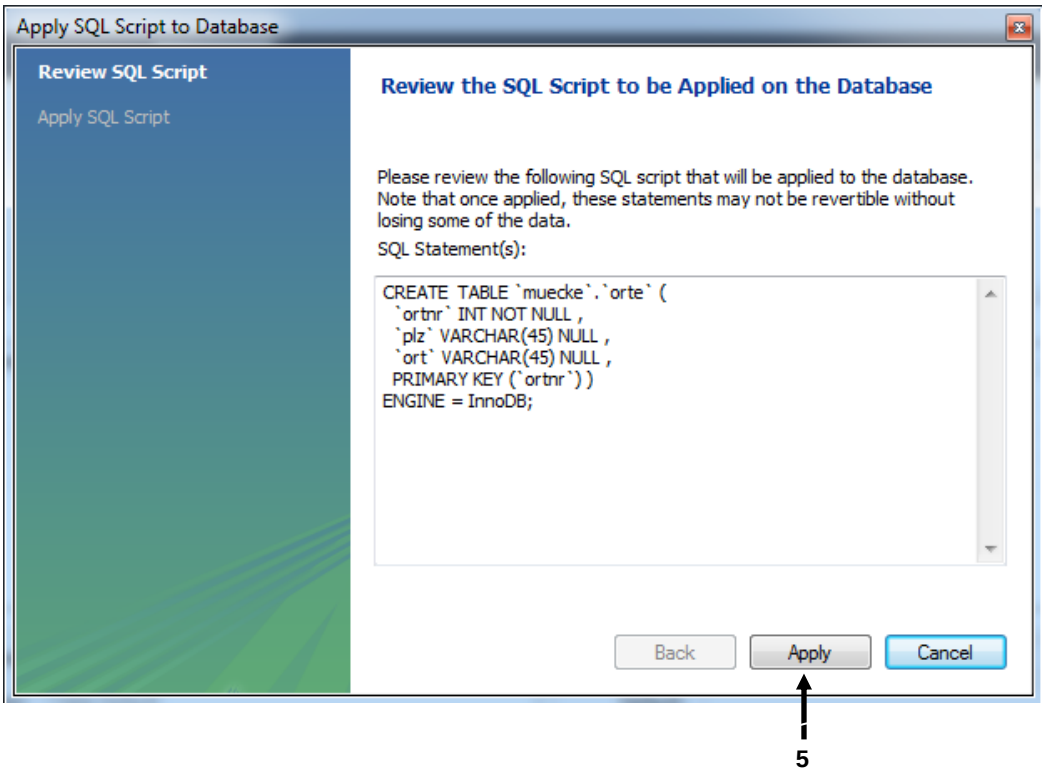
(2) Im Fenster *new_table* wird dann der Name der Tabelle (hier *orte*) eingetragen und als Engine (Tabellentyp) InnoDB ausgewählt. Die *InnoDB*-Engine bietet sichere Schreib-/ Lesezugriffe auf Tabellen und gewährleistet die Überprüfung der *referentiellen Integrität*.

(3) Anschließend werden im Register *Columns* die Attribute und Datentypen angelegt. Ein Attributname wird zum Anlegen oder Ändern editiert, wenn man einen Doppelklick in das Feld vornimmt. Standardmäßig erhält das Primärschlüsselattribut den Namen *idTabellenname*, für die Tabelle *orte* also den Namen *idOrte*. Diese Bezeichnung kann beibehalten oder geändert werden.

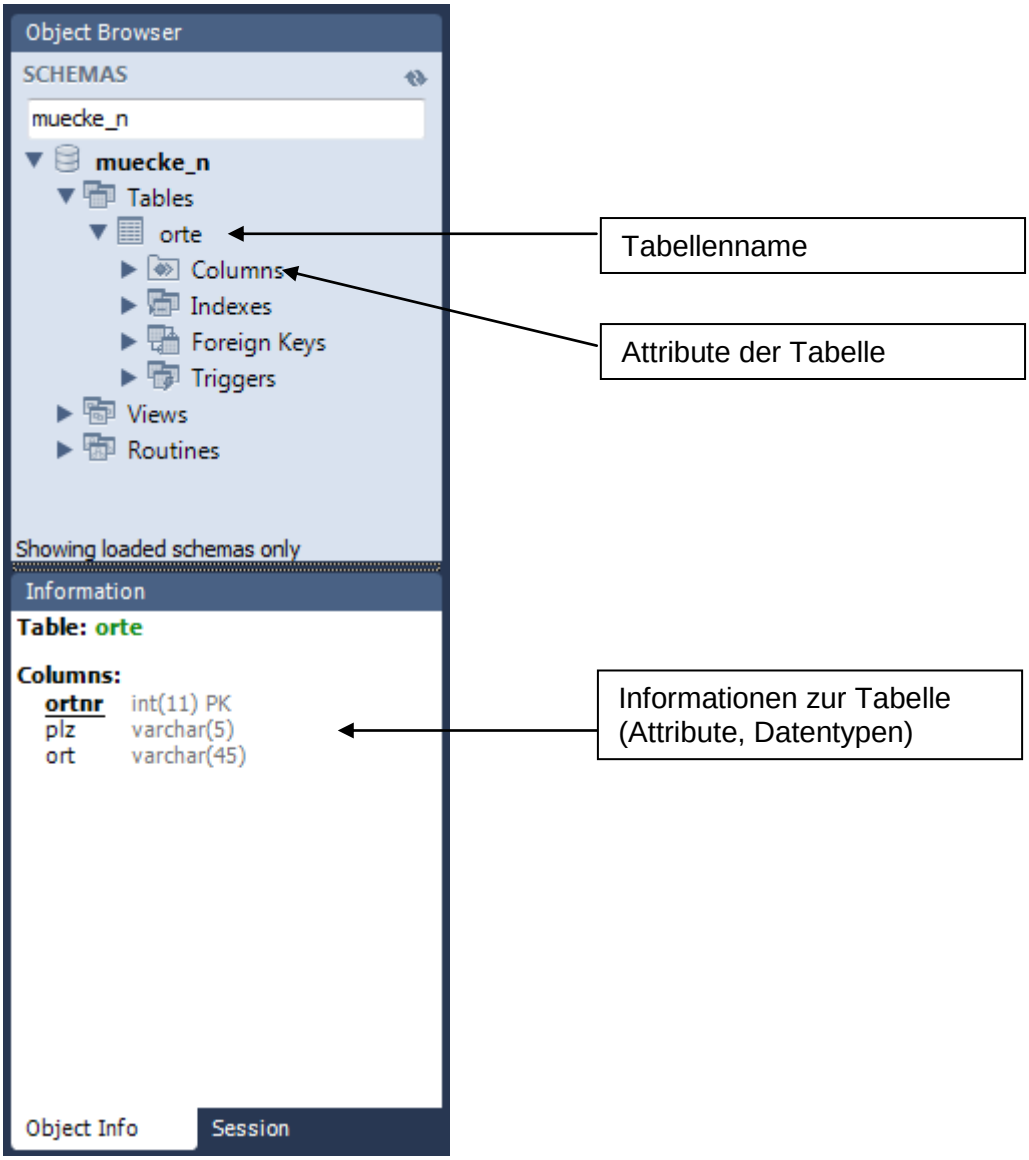
Weitere Eigenschaften, die für die Attribute festgelegt werden können:

- *NOT NULL*: Wenn die Option gewählt ist, ist das betreffende Attribut ein Pflichtattribut und benötigt eine Eingabe. Ist die Option *NN* nicht gewählt, kann das Attribut auch ohne Inhalt bleiben.
- *ZF: Zero Fill*. Nicht benötigte Stellen in numerischen Attributen werden mit 0 aufgefüllt.
  Wird beispielsweise das Attribut *ortnr* als Datentyp Int mit der Länge 5 vereinbart und die Option ZF aktiviert, dann werden später Eingaben mit weniger als fünf Stellen mit 0 aufgefüllt.
- *AI: Auto increment*. Diese Eigenschaft legt fest, dass bei numerischen Attributen fortlaufend weitergezählt wird.

(4) Wird der Tabellenentwurf mit dem Button **[Apply]** abgeschlossen, erscheint der generierte SQL-Code im Fenster *Apply SQL Script to Database*.



(5) Abschließend wird das SQL-Skript mit dem Button [**Apply**] ausgeführt. Es erscheint die angelegte Tabelle mit den Attributen im Fenster des *Object Browsers* unter dem Datenbankschema *muecke_n* im Ordner *Tables*.



➢ **Löschen einer Tabelle aus der Datenbank**

Falls eine Datenbanktabelle (Tabellenstruktur und Tabellendaten) aus der Datenbank entfernt werden muss, erfolgt dies mit der SQL-Anweisung.

Syntax: DROP TABLE <tabellenname>;

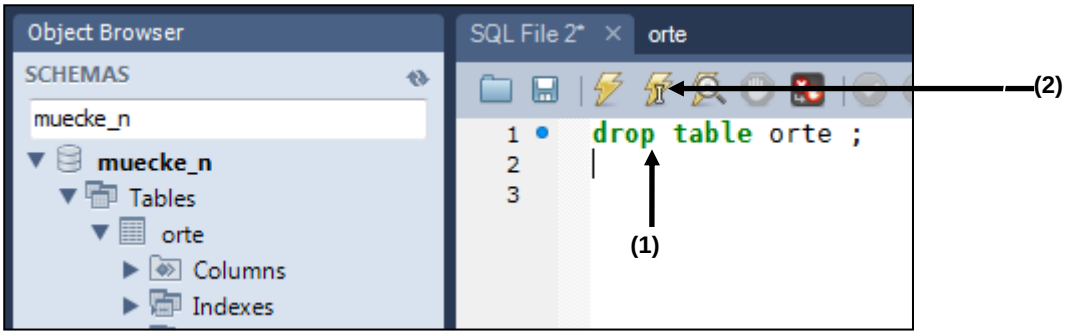
Zum Löschen der Tabelle orte ist der SQL-Befehl

```
DROP TABLE orte;
```

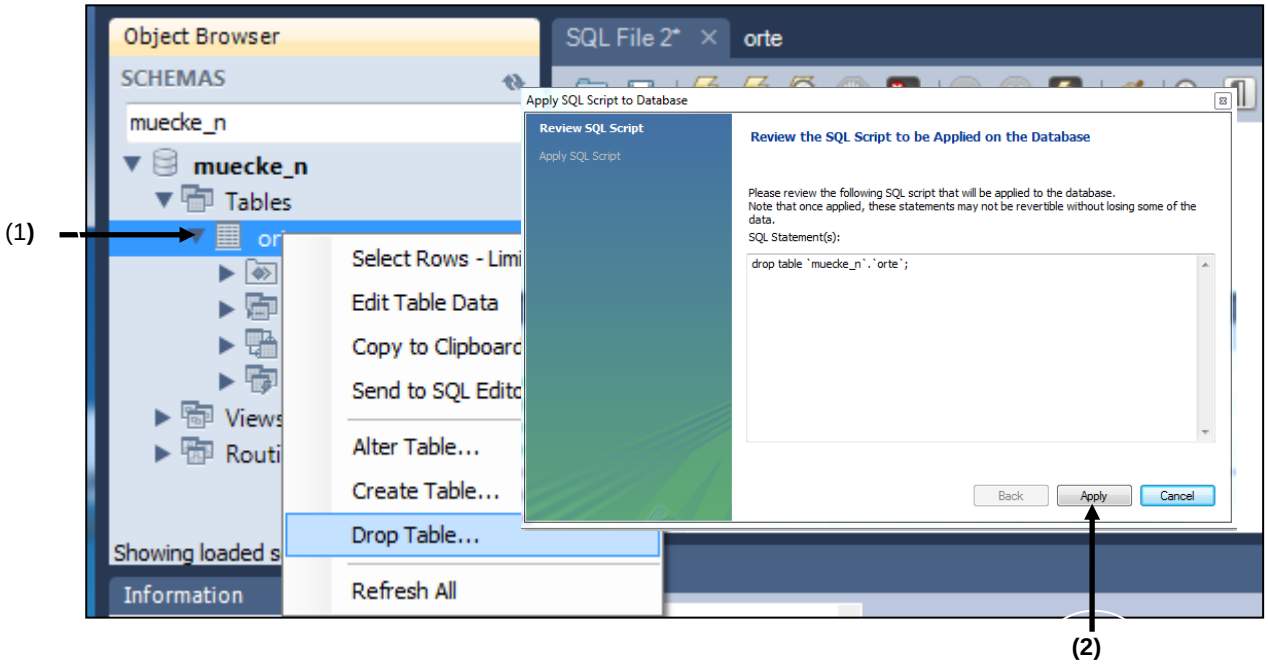
Einzugeben.

Die SQL-Anweisung zum Löschen der Tabelle wird entweder

- **im SQL File Editor eingegeben (1) und ausgeführt (2) oder**



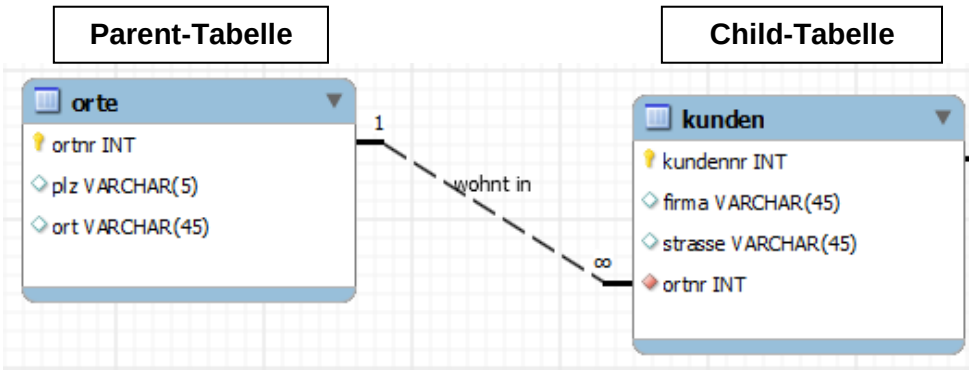
- **mit Hilfe des Kontextmenüs zur Tabelle *orte* (1) generiert und ausgeführt (2).**



➢ Die Tabelle *kunden* erstellen

Die Tabelle kunden steht mit der Tabelle orte in Beziehung.
Die Kardinlität der Beziehung ist 1 – N, das bedeutet, dass an einem Ort mehrere Kunden leben können, während ein Kunde an einem Ort lebt.

Die Beziehung wird durch die Verknüpfung von Schlüsselattributen erzeugt. Hierzu muss auf der N-Seite der Child-Tabelle der Primärschlüssel der Parent-Tabelle als Fremdschlüssel eingefügt werden.



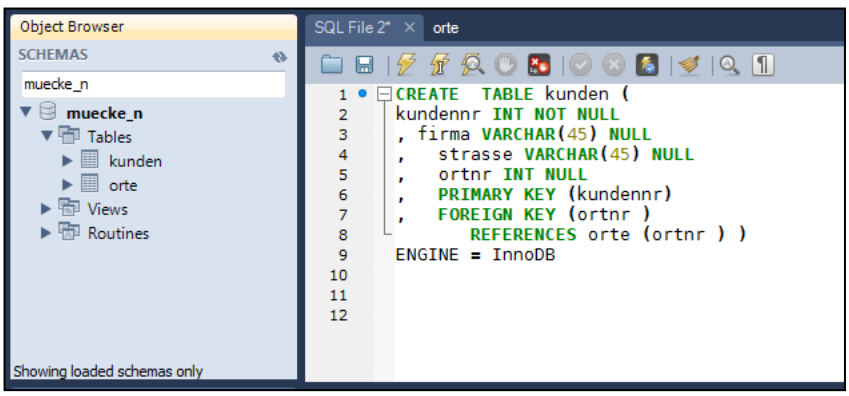
SQL-Anweisung zum Erzeugen der Tabelle *kunden*

```
CREATE  TABLE kunden (
kundennr INT NOT NULL
, firma VARCHAR(45) NULL
,   strasse VARCHAR(45) NULL
,   ortnr INT NULL
,   PRIMARY KEY (kundennr)
,   FOREIGN KEY (ortnr )
       REFERENCES orte (ortnr ) )
ENGINE = InnoDB ;
```

- Tabelle erzeugen
- Attribute und Datentypen anlegen
- Primärschlüssel festlegen
- Fremdschlüssel einrichten
  Verbindung zum Primärschlüssel der Parent-Tabelle
- Tabellentyp (Gewährleistung der referentiellen Integrität)

Die Anweisung kann nun wie auf den Seiten 34ff beschrieben

- mit Hilfe des *SQL File* **Editors** eingegeben und ausgeführt;



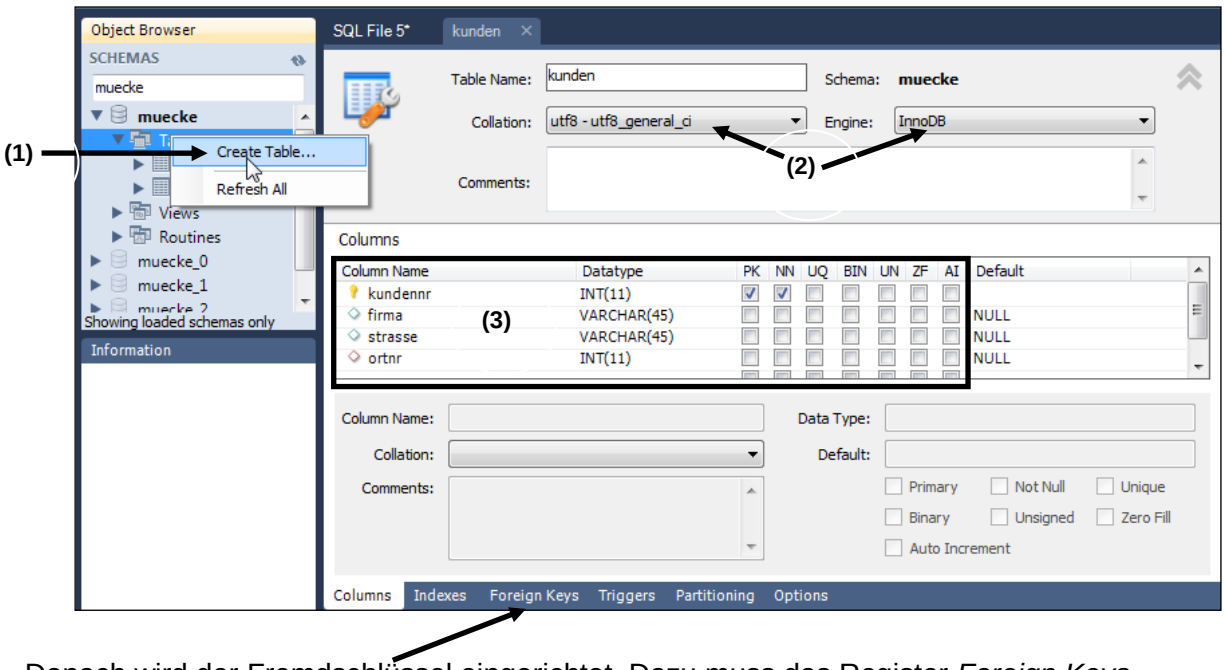
- sowie mit der *Workbench* generiert werden. Bei dieser Vorgehensweise muss beachtet werden, dass der Fremdschlüssel angelegt werden muss.

  Vorgehensweise

  (1) Im *Object-Browser* mit dem Kontextmenü zum Ordner *Table* den Befehl *Create Table…* aufrufen und anlegen.

  (2) Den Zeichensatz für die Datenbank (utf8) sowie den Typ der Datenbankengine (InnoDB) festlegen.

  (3) Die Attribute mit den Attributtypen eingeben.

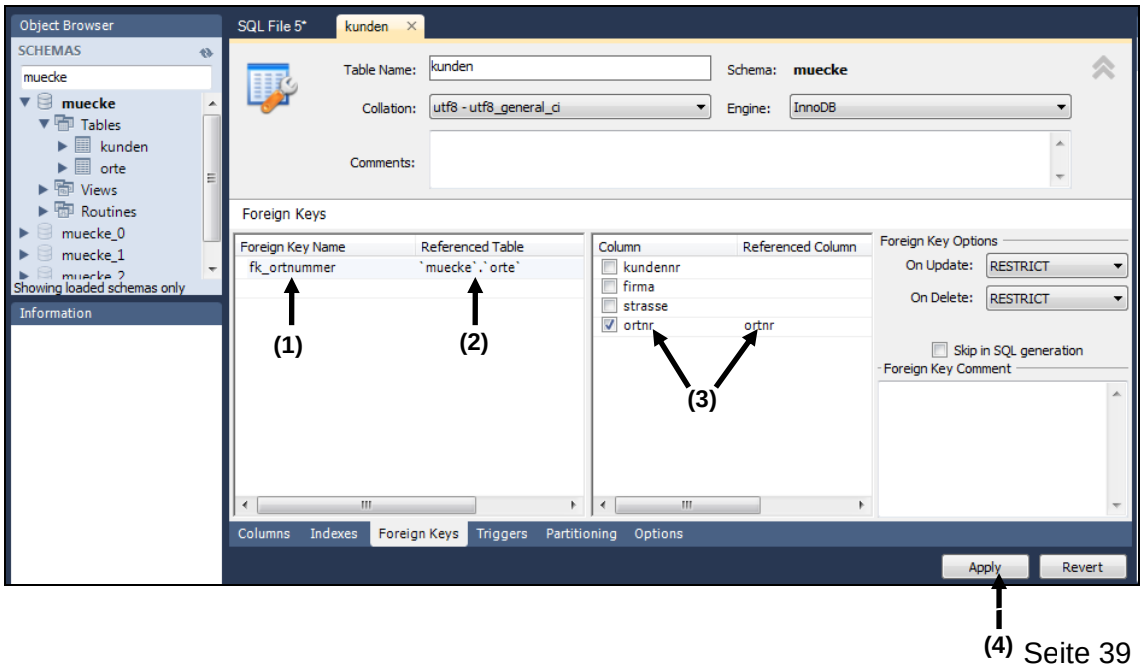


Danach wird der Fremdschlüssel eingerichtet. Dazu muss das Register *Foreign Keys* geöffnet werden. Dann wird

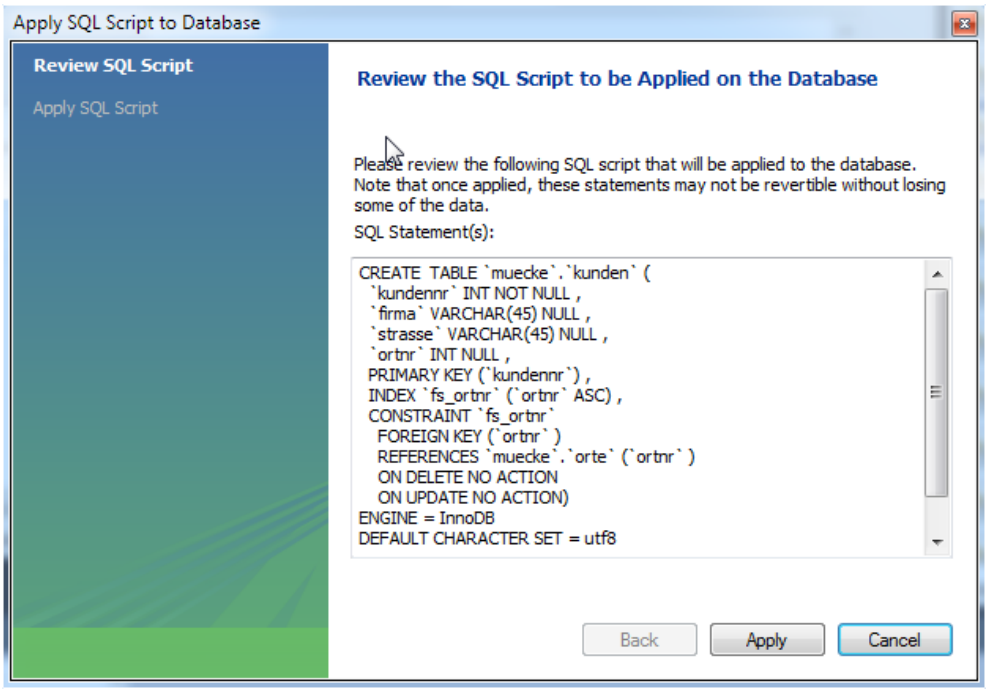
(1) ein Name für die Verbindung vergeben,

(2) die (Parent-)Tabelle, die den Referenzschlüssel enthält ausgewählt und

(3) der Fremdschlüssel dem Referenzschlüssel, zugeordnet.



(4) Drückt man die Schaltfläche [**Apply**], wird die SQL-Anweisung generiert und sie kann zum Datenbankserver gesendet werden.



➢ Ändern einer Tabellenstruktur

• Ändern der Tabellenstruktur mit Hilfe von SQL-Anweisungen

Mit der SQL-Anweisung *ALTER TABLE* kann die Struktur einer bestehenden Tabelle geändert werden. Beispielsweise können Spalten (Attribute) hinzugefügt oder gelöscht, die Bezeichnung bestehender Attribute geändert, der Datentyp bestehender Attribute geändert, oder die Tabelle selbst umbenannt werden.

- Attribute hinzufügen

Syntax:ALTER TABLE <tabellenname> ADD COLUMN <attributbezeichnung> <datentyp>;

Beispiel: Für jeden Kunden sollen dessen Telefonnummer (Integer-Zahl) und die E-Mail- Adresse (Text mit 25 Zeichen) erfasst werden können.

```
ALTER TABLE kunden ADD COLUMN telefon INT;
ALTER TABLE kunden ADD COLUMN email VARCHAR(25);
```

- Attributstyp ändern

Syntax:ALTER TABLE <tabellenname> MODIFY <attributname> <datentyp>;

Beispiel: Der Datentyp des Attributs *telefon* wurde falsch definiert. Der Attributtyp muss wegen der führenden 0 in der Vorwahl in einen Text - *VARCHAR(25)* - geändert werden.

```
ALTER TABLE kunden MODIFY telefon VARCHAR(25);
```

- Attribute entfernen

Syntax:ALTER TABLE <tabellenname> DROP COLUMN <attributbezeichnung>

Beispiel: Das Attribut *telefon* der Tabelle *kunden* ist zu entfernen.

```
ALTER TABLE kunden DROP COLUMN telefon;
```

- Attributname ändern

Syntax:ALTER TABLE <tabellenname> CHANGE <attributname_alt> <attributname_neu> <datentyp>

Beispiel: In der Tabelle *kunden* soll anstelle der E-Mail-Adresse die URL gespeichert werden. Das Attribut *email* ist in *url* umzubenennen. Für die URL-Adresse sind 50 Zeichen zu reservieren.

```
ALTER TABLE kunden CHANGE email url VARCHAR(50);
```

- Tabellennamen ändern

Syntax:ALTER TABLE <tabellenname> RENAME TO <neuerName>

Beispiel: Die Tabelle *kunden* soll den neuen Namen *debitoren* erhalten.

```
ALTER TABLE kunden RENAME TO debitoren;
```

- **Ändern der Tabellenstruktur mit Hilfe der MySQL Workbench**

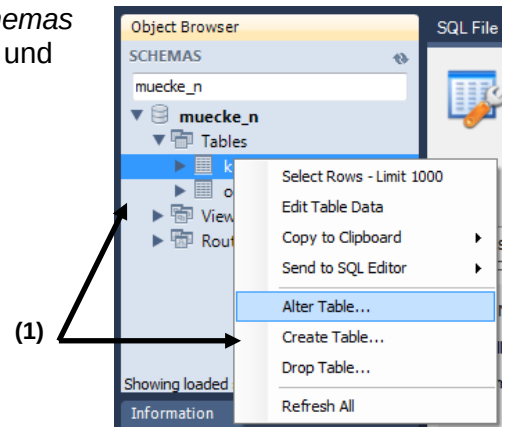
Die beschriebenen Änderungen in der Tabellenstruktur können in der *Workbench* auch mit Hilfe des *Workbench*-Assistenten vorgenommen werden

Aufgabenstellung:

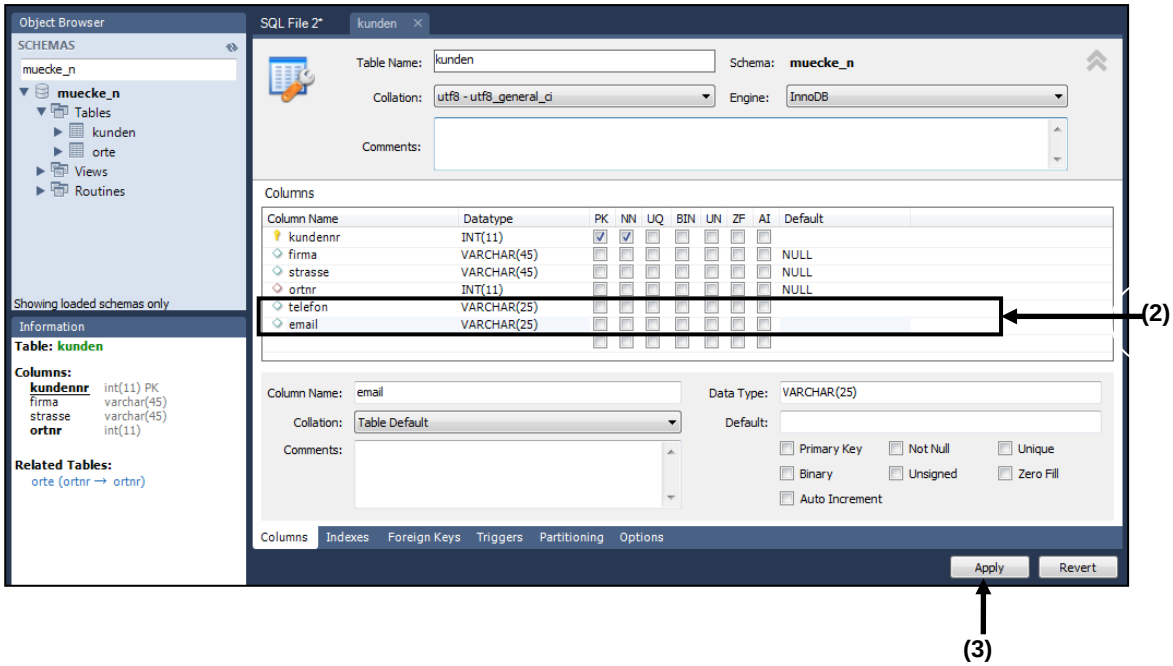
Für jeden Kunden sollen dessen Telefonnummer und die E-Mail-Adresse erfasst werden können. Für beide Attribute soll der Datentyp VARCHAR(25) gewählt werden.

Vorgehensweise:

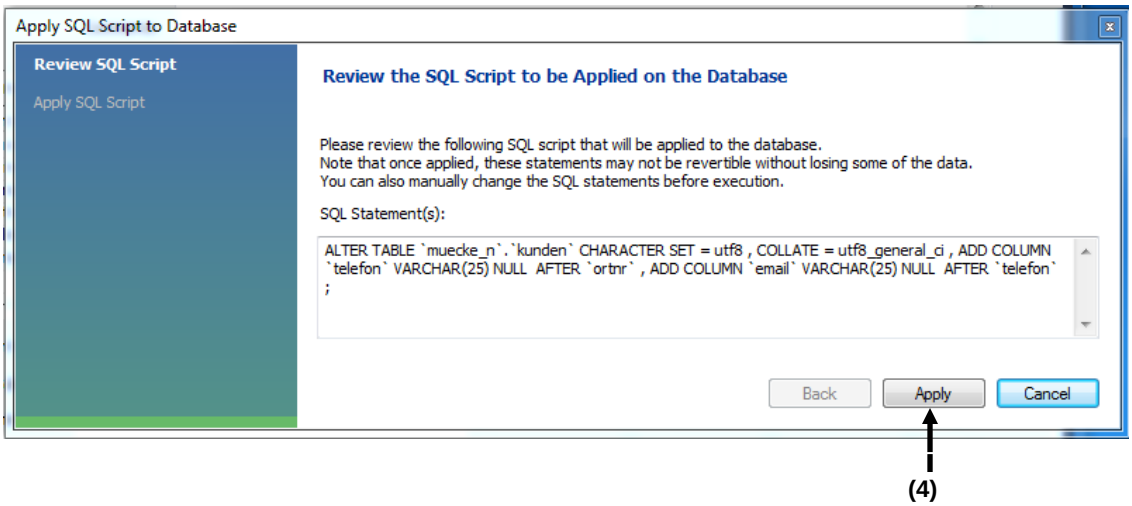
(1) Im Fenster *Objekt Browser* in der Rubrik *Schemas* das Kontextmenü zur Tabelle *kunden* öffnen und den Menübefehl *Alter Table*… auswählen.



(2) Jetzt kann die Tabelle *kunden* verändert werden, indem in den verschiedenen Registern die Struktur der Tabelle je nach Bedarf verändert wird. Im folgenden Beispiel werden die beiden Attribute telefon und email neu hinzugefügt.



(3) Wird die Schaltfläche [**`Apply`**] ausgeführt, so wird der SQL-Befehl generiert.



(4) Durch erneutes Betätigen der Schaltflächen **`[Apply]`** und danach **`[Finish]`** wird die Anweisung ausgeführt und die Tabellenstruktur geändert.