|  |  |
| --- | --- |
| **Lösungsblatt** | **DB01\_L** |

**zu Aufgabe 1:**

|  |  |
| --- | --- |
| **1. Schritt: Objekte bestimmen**  Objekte der Fahrradvermietung:   * Bilder von Fahrrädern, * Bilder von Mietern und * Mietverträgen   ungeordnet auf einer Stellwand anheften.  🡺 Begriff: **Entität** | Fahrrad3  Kunde2  Kunde1    Fahrrad2    Fahrrad1 |
| **2. Schritt: Beziehungen erläutern**  Die einzelnen Objekte werden an der Tafel zusammengefasst, z.B. | **mietet**  Kunde  Fahrrad    **vereinbart**  **bezieht sich auf** |
| **3. Schritt: Objekte und Beziehungen gruppieren**  🡺 Begriffe: **Entitätstyp**  **Beziehungstyp** | Kunde1  Fahrrad1  Kunde2    Fahrrad2  Fahrrad3 |
| **4. Schritt: Anzahl beteiligter  Entitäten bestimmen**  🡺 Begriffe: **Kardinalität**  **Entity-Relationship-Dia­**  **gramm** |  |

|  |  |
| --- | --- |
| **5 Schritt: Merkmale der Entitäten bestimmen.**  🡺 Begriff: **Attribut**  Für Attributnamen gelten die folgenden Regeln: - maximal 64 Zeichen lang, - keine Sonderzeichen, - nur Kleinbuchstaben  Attribute (genau: Attributnamen) sind die Eigenschaften, die jeder Kunde hat.  Datentyp ist die Art der Daten, die ein Attributwert aufnehmen kann. | - vorname - von - bezeichnung  - name - bis - rahmennr  - tagesmietpreis |
| **6 Schritt: Identifizierungs­schlüssel bestimmen**  🡺 Begriff: **Primärschlüssel** | **- kundennr - vermietnr - fahrradnr**  - vorname - von - bezeichnung  - name - bis - rahmennr  - tagesmietpreis |
| **7 Schritt: Verbindungs­ schlüssel bestimmen**  🡺 Begriffe: **Fremdschlüssel**    **Relationenmodell** | - kundennr - vermietnr - fahrradnr  - vorname - von - bezeichnung  - name - bis - rahmennr  - **kundennr** - tagesmietpreis  - **fahrradnr**  Schreibweise im Relationenmodell:  Kunde(kundennr, vorname, name)  Vermietung(vermietnr, von, bis, 🡩kundennr, 🡩fahr  radnr)  Fahrrad(fahrradnr, bezeichnung, rahmennr, tagesmiet-  preis) |

**Begriffe:**

**Entität** Eindeutig bestimmbares, von anderen wohlunterscheidbares Objekt der realen Welt oder Vorstellung, dem Informationen zugeordnet sind.  
(Individuum oder Gegenstand; materiell oder immateriell)

**🡺** Datensatz, Tupel,   
 Zeile einer Datenbanktabelle.

**Tupel** Ein Satz zusammengehöriger Werte.  
Ein Tupel wird im relationalen Datenbank-Management-System als Zeile gespeichert.

**🡺** Zeile einer Datenbanktabelle.

**Beziehung** Beschreibt den Zusammenhang, der zwischen zwei Entitäten besteht.

**Entitätstyp** Gesamtheit aller gleichartigen und zusammen gehörenden Entitäten.

**🡺** Tabelle einer Datenbank.

**Beziehungstyp** Typisierung gleichartiger Beziehungen.

**Kardinalität** Gibt die mögliche Anzahl der an einer Beziehung beteiligten Entitäten an.

**Entity-Relationsship-Diagramm (ERD)**

Grafische Darstellung von Entitätstypen, ihren Beziehungen und deren Kardinalitäten. (M : N – Beziehungen können abgebildet werden.)

**Attribut** Alle gemeinsamen Merkmale aller Entitäten eines Entitätstyps.

**🡺** Spalte einer Datenbanktabelle.

**Datentyp** Bestimmt den Wertebereich, den ein Attribut annehmen kann.

**🡺** Text, ganze Zahl, Dezimalzahl, Datum etc.

**Attributwert** Eine Ausprägung eines Merkmals einer Entität.

**🡺** Schnittpunkt zwischen Zeile und Spalte (die Zelle).

**Primärschlüssel** Bezeichnet die Information, die eine Entität eindeutig identifiziert.

**Relationenmodell** Übersichtliche Darstellung eines Entity-Relationsship-Modells, in dem die Entitätstypen als Relationen mit Attributen, Primär- und Fremdschlüssel abgebildet werden. M : N – Beziehungen sind dabei aufzulösen.

**Regeln für die Datenmodellierung (Entity-Relationship-Modellierung)**

Ein Entity-Relationship-Modell (ERM) stellt einen Standard für die Datenmodellierung dar. Es bildet die Datenstrukturen ab und dient zum einen in der konzeptionellen Phase der Anwendungsentwicklung der Verständigung zwischen Anwendern und Entwicklern. Zum anderen ist das ER-Modell in der [Implementierungsphase](http://de.wikipedia.org/wiki/Implementierung) die Grundlage für das Design der [Datenbank](http://de.wikipedia.org/wiki/Datenbank).

Die Umsetzung der in der realen Welt erkannten Objekte und Beziehungen in ein Datenbankschema erfolgt dabei in mehreren Schritten:

1) Entitäten (Objekte) und Beziehungen identifizieren

2) Festlegen der Entitäts- und Beziehungstypen

3) Festlegen der Beziehungskardinalitäten

4) Attribute und ihre Wertebereiche (Domänen) ermitteln

5) Entity-Relationship-Diagramm (ERD) für Entitäts- und Beziehungsmengen zeichnen

6) Definieren von Primär- und Fremdschlüsseln

7) Relationenmodell erstellen

**zu Aufgabe 2:**

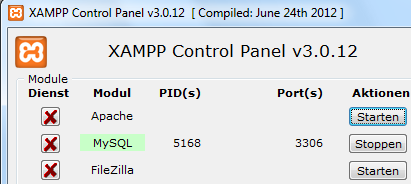
**2.1 Erweiterung der Kundendaten**

|  |  |
| --- | --- |
| **Erweiterung des ERD** |  |
| **Attribute der erweiterten  Entitätstypen** |  |

**2.2 Modellieren mit der MySQL Workbench**

MySQL ist eines der weltweit am weitesten verbreiteten Datenbank-Management-Systeme, das insbesondere für die Datenspeicherung im Internet eingesetzt wird. MySQL sieht deshalb grundsätzlich einen MySQL-Server vor, auf dem die Daten gespeichert werden. Dieser Server lässt sich mit Hilfe von xampp einfach auf dem PC einrichten.

(Siehe hierzu: Relationaler Datenbanken: Toolgestützte Modellierung;   
 http://www.schule-bw.de/unterricht/faecher/informatik/material/datenbanken/relationale-datenbanken/)

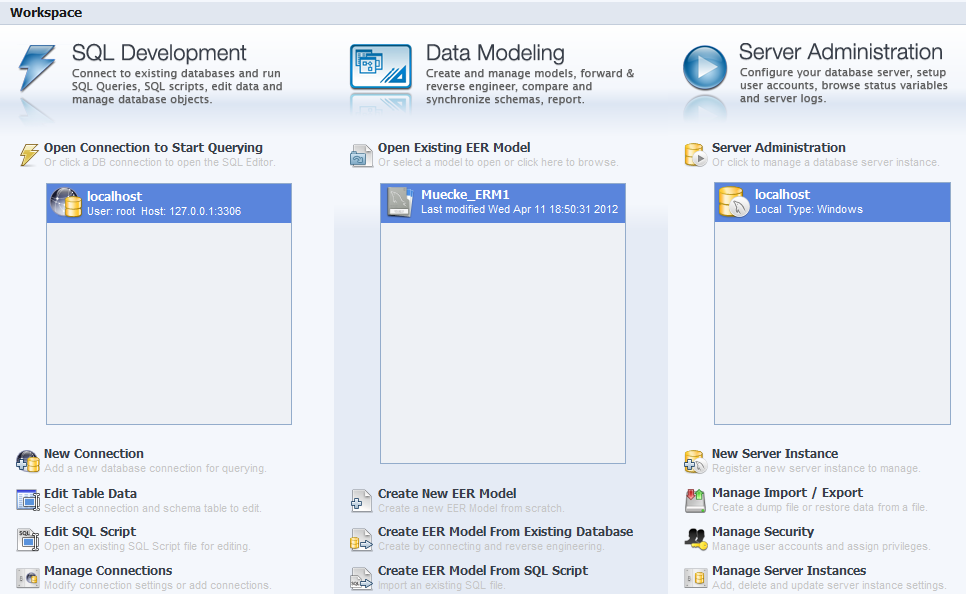


Der MySQL-Server wird über das Control Panel von XAMPP gestartet (xampp-control.exe im xampp-Ordner).



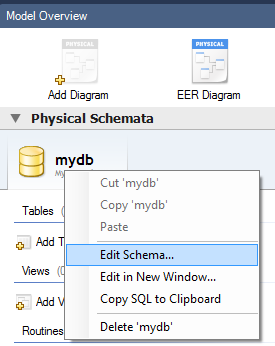
Anschließend wird die MySQL Workbench gestartet werden.

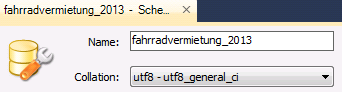
Zum Erstellen/Bearbeiten eines erweiterten Entity Relationship Models wird der Programmteil ***Data Modeling*** der MySQL Workbench verwendet.



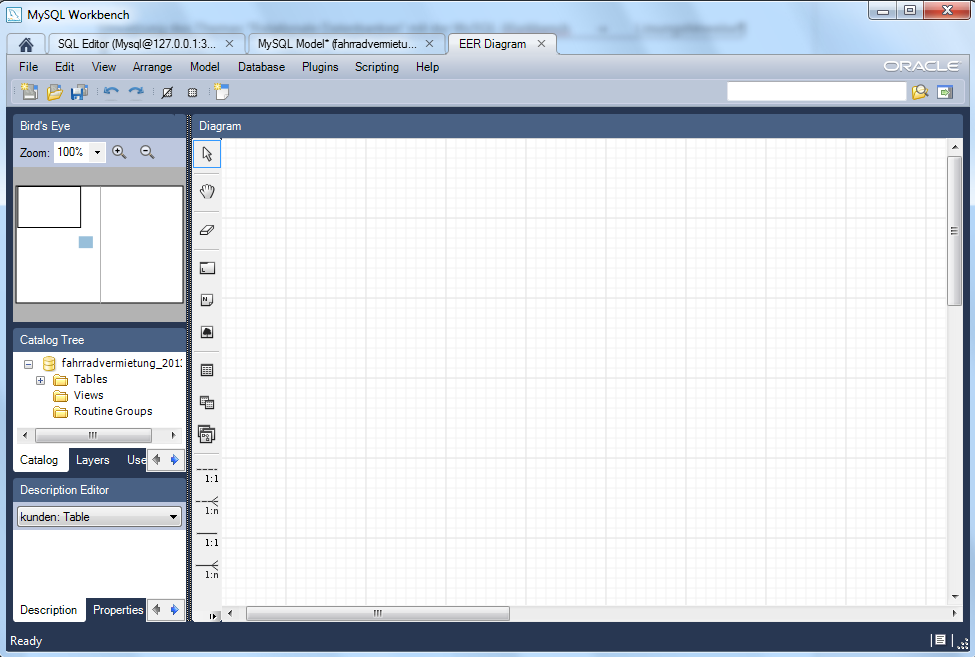
Nach dem Starten der *Workbench* wird ein Modell (Schema) mit dem Namen *fahrradvermietung\_2013* eingerichtet. Dazu wird aus dem Programmteil *Data Modeling* das Menü

Create New EER Model aufgerufen.

Datenbankname vergeben und Diagramm anlegen:

1. Im angelegten Modell wird im Fenster Model Overview zuerst der Name 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).  
   🡪
3. Mit einem Doppelklick auf die Schaltfläche **Add Diagram** wird das Entwurfsfenster zum Modellieren des ERM geöffnet.

**Aufbau des Diagrammfensters**

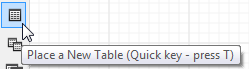


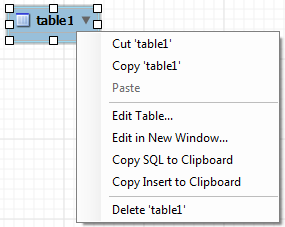
Objekt markieren

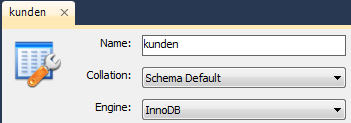
Objekt löschen

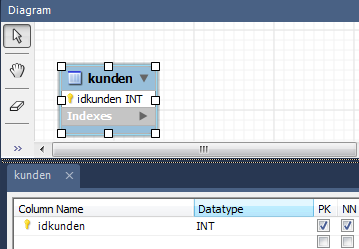
Neue Tabelle

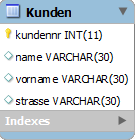
Beziehungen einrichten

**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 Name der Tabelle eingegeben (kunden).   
   (Alle Tabellennamen sollen klein und im Plural geschrieben werden!)
4. Schließt man die Eingabe des Namens ab, wird das Register   
   Columns geöffnet und die Attribute können erfasst werden.

Je nach Voreinstellung, erstellt das Programm einen Primärschlüssel mit dem Namen der Tabelle und vorangestellter id (hier: *idkunden*)

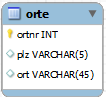
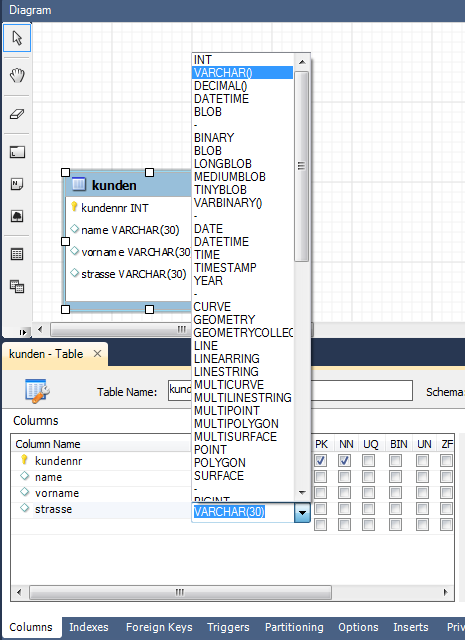


**Hinweis:** Standardmäßig wird das EERM (in der voreingestellten (= default) Notation dargestellt. In dieser Notation erscheint auch die Rubrik Indexe (siehe nebenstehende Abbildung).

Mit dem Befehl Model – Objekt Notation kann mit der Auswahl Workbench (Simplified) die Darstellung der Tabellen ohne die Rubrik Indexe eingestellt werden.

**Erfassen und Editieren der Attribute**

1. Zunächst wird im Register *Columns* der vom System vorgeschlagene Primärschlüssel angezeigt. Der vorgegebene Datentyp ist *INT*. Der vorgeschlagene Name kann beibehalten oder geändert werden. Mit einem Doppelklick in das Feld wird es editiert und kann geändert werden (hier *kundennr*).



1. Für den Primärschlüssel ist neben der CheckBox *PK* (*Primary Key)* auch *NN (Not NULL)* ausgewählt. Dies bedeutet, dass das Attribut einen Wert erhalten muss.
2. Die anderen Attribute werden entsprechend eingegeben. Für jedes Attribut ist sein Attributtyp (Datatype) festzulegen (siehe Folgeseite).
3. Das Kombinationsfeld "*Data Type*" listet die Datentypen auf, die der jeweilige SQL-Dialekt unterstützt. Für die Felder mit dem Attributtyp VARCHAR () kann ein gewünschte Länge eingetragen werden.

**Anwendung häufig verwendeter Datentypen in SQL-Datenbanken:**

|  |  |  |
| --- | --- | --- |
| **Typ** | **Bedeutung** | **Beispiel** |
| CHAR(*länge*) /  VARCHAR(länge) **\*\*** | Text | CREATE TABLE Kunden(Name CHAR(30));/  CREATE TABLE Kunden(Name VARCHAR(30)); |
| INTEGER | Ganzzahl | CREATE TABLE Inspektionen(Dauer INTEGER); |
| DOUBLE | Fließkommazahl | CREATE TABLE Fahrraeder (Wert DOUBLE); |
| DATE | Datumsangaben | CREATE TABLE Vermietungen(von DATE); |
| TIME | Zeitangaben | CREATE TABLE Unterricht (Beginn TIME); |
| DATETIME | Datums- mit Zeitangaben | CREATE TABLE Auftraege(erstlief DATETIME); |

Der **CHAR**-Datentyp ist ein Datentyp fester Länge. Wird ein Wert, der kürzer als die Länge der Spalte ist, in eine **CHAR** NOT NULL-Spalte eingefügt, so wird der Wert rechts bis zum Erreichen der Spaltengröße mit Leerzeichen aufgefüllt. Wenn z. B. eine Spalte als **CHAR (**10**)** definiert wurde und "music" die zu speichernden Daten darstellt, speichert der SQL-Server diese Daten als "music\_\_\_\_\_", wobei "\_" ein Leerzeichen anzeigt.

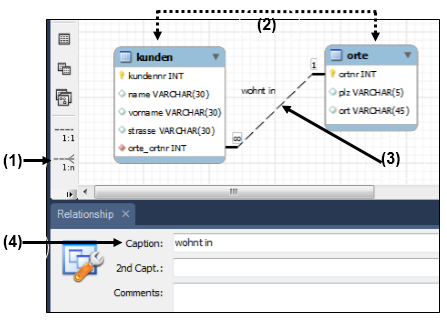
**VARCHAR**-Werte werden dagegen beim Speichern nicht aufgefüllt. Es werden nur so viele Zeichen gespeichert, wie benötigt werden (zuzüglich ein Byte für die Länge des Wertes).

**Herstellen der Beziehung zwischen den Tabellen kunden und orte**

Für die Einrichtung der Beziehung zwischen der Parent-Tabelle orte und der Child-Tabelle kunden ist zu beachten, dass in der Tabelle **kunden** das Fremdschlüsselattribut *ortnr* noch nicht eingefügt sein darf (siehe nachfolgende Abbildung).



Die Beziehung wird dann eingerichtet:

1. Symbol aus der Werkzeugleiste: **Place an New 1:n Non-Identifying Relationship [[1]](#footnote-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 🡪 ConenecttoColumns)gewählt.
4. Beziehungstyp beschreiben. Dazu wird die Verbindung markiert und editiert (Doppelklick). Danach kann die Beschriftung im Bezeichnungsfeld *Caption* vorgenommen werden.

**Hinweis:** Damit die Beschriftung der Beziehung angezeigt wird, müssen mit dem Menübefehl Model 🡪 Model Options

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

⚫ und im Register *Diagram* unter Relationship Connections die Option Hide Caption

deaktiviert werden.

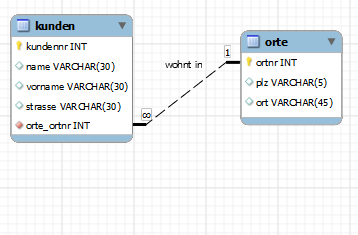
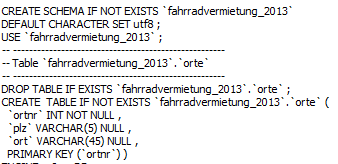
**Vom EERM zum Datenbankskript (=Forward Engineering)**

Auf der Grundlage des entwickelten erweiterten Entity-Relationship-Diagramms (EERM) soll nun die Datenbank mit den Tabellen 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.



**Forward Engineering**

**Reverse Engineering**

|  |  |
| --- | --- |
| **Vorgehensweise** | **Screenshots** |

|  |  |
| --- | --- |
| * Aus dem EER-Diagramm heraus den Assistenten mit dem Menü   Database🡪Forward Engineering  aufrufen.  Anschließend wird die Verbindung zum MySQL-Server hergestellt.  **Hinweis:** Der MySQL-Server muss zuvor mit Hilfe des Control Panels gestartet werden (siehe oben Gliederungspunkt 2.2). |  |

|  |  |
| --- | --- |
| * 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] in die Zwischenablage übernommen werden. |  |

|  |  |
| --- | --- |
| * Im abschließende Fenster *Commit Progress* wird die Verbindung zum MySQL Server hergestellt und das zuvor generierte SQL-Skript ausgeführt.   Die Schaltfläche [Close] beendet den Vorgang.  Die Struktur der Datenbank *fahrradvermietung\_2013* mit den beiden Tabellen *kunden* und *orte* befindet sich nun auf dem MySQL-Server. |  |

Nachfolgend ist das erzeugte SQL-Script dargestellt und erläutert:

CREATE SCHEMA IF NOT EXISTS `fahrradvermietung\_2013` DEFAULT CHARACTER SET utf8 ;USE `fahrradvermietung\_2013` ;-- ------------------------------------------------------- Table `fahrradvermietung\_2013`.`orte`-- -----------------------------------------------------DROP TABLE IF EXISTS `fahrradvermietung\_2013`.`orte` ;CREATE TABLE IF NOT EXISTS `fahrradvermietung\_2013`.`orte` ( `ortnr` INT NOT NULL , `plz` VARCHAR(5) NULL , `ort` VARCHAR(45) NULL , PRIMARY KEY (`ortnr`) )ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

-- ------------------------------------------------------- Table `fahrradvermietung\_2013`.`kunden`-- -----------------------------------------------------DROP TABLE IF EXISTS `fahrradvermietung\_2013`.`kunden` ;

Tabelle erzeugen

CREATE TABLE IF NOT EXISTS `fahrradvermietung\_2013`.`kunden` ( `kundennr` INT NOT NULL , `name` VARCHAR(30) NULL DEFAULT NULL , `vorname` VARCHAR(30) NULL DEFAULT NULL , `strasse` VARCHAR(30) NULL DEFAULT NULL , `orte\_ortnr` INT NOT NULL , PRIMARY KEY (`kundennr`) , INDEX `fk\_kunden\_orte\_idx` (`orte\_ortnr` ASC) , CONSTRAINT `fk\_kunden\_orte` FOREIGN KEY (`orte\_ortnr` ) REFERENCES `fahrradvermietung\_2013`.`orte` (`ortnr`) ON DELETE NO ACTION ON UPDATE NO ACTION)ENGINE = InnoDB

Tabellentyp *InnoDB* für die referentielle Integrität

festlegen

Beziehung herstellen

Primärschlüssel anlegen

Attribute anlegen

DEFAULT CHARACTER SET = utf8;

Unicode-Zeichentabelle festlegen

Datenbankstruktur der Datenbank fahrradvermietung\_2013 auf dem MySQL-Server anzeigen:

|  |  |
| --- | --- |
| Mit Hilfe der Home-Schaltfläche kann auf die Startseite der MySQL Workbench gewechselt werden. |  |

|  |  |
| --- | --- |
| Im Programmteil SQL Development wird die Verbindung zum Datenbankserver hergestellt.  Die Anmeldung beim Datenbankserver erfolgt mit 3 Parametern:  Hostname: localhost oder  127.0.0.1 Port: 3306 Username: root |  |

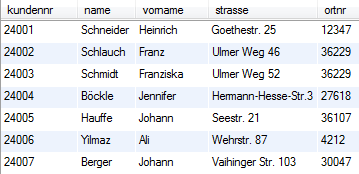
|  |  |
| --- | --- |
| Standardmäßig wird der Object-Browser (links) und der SQL-Editor (rechts) geöffnet  Im Object-Browser findet man die eigenen und die Systemdatenbanken.  Mit einem Rechtsklick in den Objekt Browser kann die Anzeige mit **Refresh All** aktualisiert werden. |  |
| Im Objekt-Browser können die Datenbanken zur Bearbeitung mit Doppelklick ausgewählt und die Tabellen und Attribute angezeigt werden. |  |

**zu Aufgabe 3:**

Zunächst werden die Daten in die Parent-Tabelle und erst dann die Daten in die Child-Tabelle eingetragen.

Die Werte für das Attribut *ortnr* als Fremdschlüssel in der Tabelle *kunden* kann erst dann erfasst werden, wenn der entsprechende Wert als Primärschlüssel in der Parent-Tabelle vorhanden ist   
🡺 referentielle Integrität.

Tabelle orte (Auszug) Tabelle kunden (Auszug)

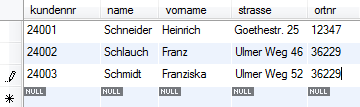
****

Weiter mit Zugriff auf DB 🡺 Daten in DB importieren ….

**Datenerfassung mit Hilfe eines vorbereiteten SQL-Skript**

|  |  |
| --- | --- |
| Das SQL-Script mit den vorbereiteten Daten der Datenbanktabellen über die Befehle  File 🡺 Open SQL Script…  öffnen |  |
| Die SQL-Anweisungen werden im SQL-Editor angezeigt.  (Die verwendeten SQL-Anweisungen können in der SQL-Befehlsliste nachgeschlagen werden.) |  |
| Anschließend muss das SQL-Script ausgeführt werden. |  |
| Im Protokoll unter *Output* kann geprüft werden, ob Fehler aufgetreten sind.  Im nebenstehenden Beispiel war alles perfekt… |  |

**Exkurs: Daten manuell erfassen**



|  |  |
| --- | --- |
| Sollen die Daten der Datenbank­tabellen manuell erfasst werden, so ist im Objekt Browser die entsprechende Tabelle zu markieren. Im Kontextmenü wird die Option *Edit Table Data* gewählt. |  |
| Nun können die Daten in die einzelnen Spalten eintragen werden. |  |

|  |  |
| --- | --- |
| Mit der Schaltfläche *Apply* wird der SQL-Befehl zum Einfügen eines neuen Datensatzes generiert.  Es besteht auch die Möglichkeit, mehrere Datensätze in die Tabelle einzutragen und einen SQL-Befehl für alle Datensätze generieren zu lassen. |  |

Die nochmalige Bestätigung des **Apply –Buttons** führt die SQL-Anweisung aus und fügt die Kundendaten in die Tabelle ein. Mit der Schaltfläche Finish wird der Vorgang abgeschlossen.

**zu Aufgabe 4:**

**Befehlssyntax der SELECT-Anweisung**

**SELECT** <attributbezeichnung1>, <attributbezeichnung2> … ( \* = alle Felder )

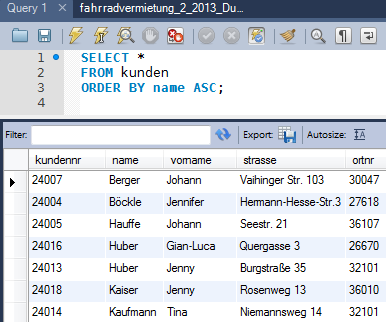
**FROM** <tabelle> [, <tabelle>, <tabelle>....]

[**WHERE** <bedingung> ]

[**GROUP BY** <attributbezeichnung> ]

[**HAVING** <bedingung> ];

[**ORDER BY** <attributbezeichnung> [ASC|DESC]..]

4.1 Die SQL-Anweisungen werden im SQL-Editor   
der MySQL Workbench eingegeben und mit   
dem Execute-Button für ein Statement ausge-  
führt.

SELECT \*  
FROM kunden  
ORDER BY name ASC;

Anmerung:

\* Platzhalter; steht für alle Attribute

ASC aufsteigend (Default-Wert)

DESC absteigend

4.2 SELECT name, vorname   
FROM kunden;

**Anmerkung:** ⮚ Auswahl von bestimmten Attributen einer Datenbanktabelle, die angezeigt werden sollen (vertikale Auswahl)   
🡺 Projektion

4.3 SELECT name, vorname  
FROM kunden  
WHERE name = 'Schlauch';

**Anmerkung:** ⮚ Auswahl bestimmter Zeilen (Datensätze) einer Datenbanktabelle, die angezeigt werden sollen.(horizontale Auswahl)   
🡺 Selektion

⮚ Die Auswahl erfolgt mit Hilfe von Bedingungen innerhalb der WHERE-Klausel.

⮚ Aufbau einer Bedingung:

attributbezeichnung vergleichsoperator vergleichswert

z.B. name = 'Huber'

⮚ Bei einer Bedingung mit einfachem Textvergleich steht der Vergleichswert in Anführungszeichen.

⮚ Bei einer Bedingung mit numerischem Vergleich steht der Vergleichswert ohne Anführungszeichen.

⮚ Bei einer Bedingung die einen Vergleich mit einem Datumswert enthält, erfolgt die Datumswertangabe in folgender Schreibweise: 'yyyy-mm-tt'.

z.B. eintritt > '2010-10-31'

4.4 SELECT name, vorname  
FROM kunden  
WHERE name LIKE 'S%';

**Anmerkung:** ⮚ Soll eine Selektion nicht durch einen präzisen Vergleichswert sondern nur durch Teile davon erfolgen, muss im Vergleichswert ein Platzhalter eingefügt werden. In diesem Falle lautet der Vergleichsoperator LIKE.

⮚ MySQL verwendet als Platzhalter für beliebige Zeichen innerhalb einer Zeichenkette das %-Symbol.

z.B. name LIKE 'Heinz%'

⮚ Platzhalter für ein beliebiges Zeichen ist das \_-Symbol (Underline).

z.B. name LIKE 'Ma\_er'

4.5 SELECT \*

FROM kunden

WHERE name = 'Huber'

AND vorname = 'Jenny';

**Anmerkung:** ⮚ Die Auswahl von Datenbankinhalten kann durch mehrere Bedingungen eingeschränkt werden.  
Sollen alle Selektionsbedingungen erfüllt sein, so müssen die einzelnen Bedingungen mit dem Operator AND verknüpft werden.  
(Sowohl Bedingung 1 als auch Bedingung 2 .… **müssen** erfüllt sein)

z.B. gehalt > 5000 AND kostnr = '03'

4.6 SELECT vorname, name

FROM kunden

WHERE name ='Berger'

OR name = 'Huber';

Anmerkung: ⮚ Soll mindestens eine der Selektionsbedingungen erfüllt sein, so müssen die einzelnen Bedingungen mit dem Operator OR verknüpft werden.  
(Entweder Bedingung 1, oder Bedingung 2 .… oder alle Bedingungen können erfüllt sein!)

z.B. ortnr = 29521 OR ortnr = 29531

4.7 SELECT name, vorname, ort   
FROM kunden, orte  
WHERE orte.ortnr=kunden.ortnr;

**Anmerkung:** ⮚ Die gewünschten Daten entstammen aus zwei Tabellen (kunden, orte).   
Die Verknüpfung der beiden Tabellen erfolgt mit Hilfe eines **Equi-Join**   
(Join ≈ Verbund), der zunächst jede Zeile einer Tabellen mit allen Zeilen der zweiten Tabelle verbindet (kartesisches Produkt) und dann diejenigen auswählt, bei denen die Inhalte der Spalten *ortnr* identisch sind (Selektion).

4.8 SELECT name, vorname, ort   
FROM kunden, orte  
WHERE orte.ortnr=kunden.ortnr  
AND orte.ort= 'Freiburg‘;

4.9 SELECT name, vorname, plz, ort  
FROM kunden, orte  
WHERE orte.ortnr=kunden.ortnr  
AND plz LIKE '7%';

**zu 5.:**

Unterrichtsidee:   
Die Lehrkraft ist DB-Administrator und die Schüler sind die Sachbearbeiter der Fahrradvermietung.

Nun können alle Schüler der Klasse als 'Sachbearbeiter' die Kundendaten in der Datenbank der Lehrkraft pflegen. So kann gezeigt werden, wie eine Client-Server-Architektur funktioniert.

Im DB-Unterricht sind die Schüler aber nur selten 'Sachbearbeiter', meistens sind sie 'Datenbankadministrator'.



Hinweis: Zur besseren Handhabung wird ein nach Ortsname alphabetisch sortiertes Listenfeld verwendet.

**zu 6.:**

|  |  |
| --- | --- |
| Abfrage erstellen: |  |
| View erstellen und ausführen: |  |
|  |  |
| In Excel: | Daten/Externe Daten abrufen/Aus anderen Quellen/Von Microsoft Query/ DSN auswählen bzw. erstellen/Query auswählen |
|  |  |
| Diagramm erstellen: |  |

**zu 7.:**

|  |  |
| --- | --- |
| Abfrage erstellen: |  |
| View erstellen und ausführen: |  |
|  |  |
| In Access: | Bericht erstellen mit dem Berichts-Assistent |

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. [↑](#footnote-ref-1)