Inhaltsverzeichnis

SQL: Structered Query Language

Ziele

- SQL kennenlernen und anwenden
  - create table, drop table
  - constraints (PK;FK,not null, unique, …)
  - select: where-klausel, joins, Gruppierungen
  - insert, update, delete
  - index
  - views, stored procedures
- Datentransfer mittels sql-Files
- Backup/Restore

Werkzeuge und Grundlagen

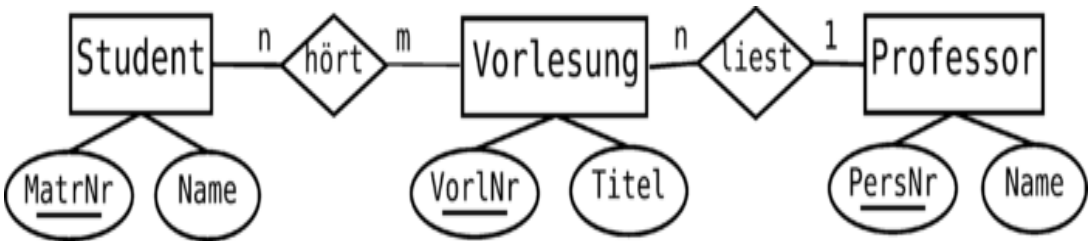
- Oracle, MySQL
- Arbeitsblatt: db-02-sql.pdf
- http://www.datenbank-sql.de/index.htm
- http://de.wikipedia.org/wiki/SQL
- http://sqlzoo.net

IS-Uni: Universitätssystem

Aufgabe: sql-create table

Aufgabe: sql-create table

Erstellen Sie auf der Grundlage des folg. ERD die create table Anweisungen. Achten Sie auf die notwendigen Primär/Fremdschlüssel.



Aufgabe: sql-insert

Aufgabe: sql-insert

Erstellen Sie zu den in der obigen (Aufgabe: sql-craeate table) erstellten Tabellen die insert into Anweisungen, sodass pro Tabelle mindestens 3 Datensätze eingefügt werden.

Aufgabe: sql-is_uni

Hier werden folg. Themen besprochen:

Select, joins, Gruppierungen, views, index

Aufgabe: sql-is_uni

- Voraussetzung: Die Tabellen müssen erzeugt worden sein: (siehe: is_uni_mysql.sql)
  -- Die Datei is_uni_mysql.sql is_uni_oracle.sql enthält alle SQL-Anweisungen zum Erzeugen der Tabellen.

- Starten Sie ein DB-Client-Programm: (SQL-Developer, MySql-Browser, ...)
  Kopieren sie folgende Datei in den dortigen SQL-Editor und
- Beantworten Sie die Fragen mit dem Schwierigkeitsgrad: 1-, 2-
- Schwierigkeitsgrad: (1-)(2-)(3-)(9-)

```
-- Datei: fragen_ohne_antwort.txt
--
-- SQL-Kurs:
--
-- Folgende Beispiele beziehen sich auf die Universitätsdatenbank.
--
-- ------------------------------------------------------------------------------------------
```

```
-- ---------------
-- SELECT Abfragen
-- ---------------
1-1. Liste alle IS_Studenten:

1-2. Liste Personalnummer und Name der C4-IS_Professoren:

1-3. Zähle alle IS_Studenten:

1-4. Liste Namen und Studiendauer in Jahren von allen IS_Studenten,
die eine Semesterangabe haben:

1-5. Liste alle IS_Studenten mit Semesterzahlen zwischen 1 und 4:

1-6.Liste alle IS_Vorlesungen, die im Titel den String
Ethik enthalten, klein oder groß geschrieben:

1-7. Liste Personalnummer, Name und Rang aller IS_Professoren,
absteigend sortiert nach Rang, innerhalb des Rangs
aufsteigend sortiert nach Name:

1-8. Liste alle verschiedenen Einträge in der Spalte Rang der Tabelle IS_Professoren:


-- -------------
-- Date, Time
-- http://dev.mysql.com/doc/refman/5.1/de/date-and-time-functions.html
-- -------------

0-9. Liste alle Geburtstage mit ausgeschriebenem Monatsnamen:

0-10.Liste das Alter der IS_Studenten in Jahren:
http://dev.mysql.com/doc/refman/5.1/de/date-and-time-functions.html

0-11. Liste die Wochentage der Geburtsdaten der IS_Studenten:
http://dev.mysql.com/doc/refman/5.1/de/date-and-time-functions.html

0-12. Liste die Uhrzeiten der Geburtsdaten der IS_Studenten:
http://dev.mysql.com/doc/refman/5.1/de/date-and-time-functions.html


-- ---------------------------------
-- JOINS
-- ---------------------------------

1-13. Welcher Professor liest Mäeutik?

1-14.Liste die Namen der IS_Studenten mit ihren Vorlesungstiteln:


-- ----------------------
-- JOIN: Self-referencing
-- ----------------------
3-15. Liste die Namen der IS_Assistenten, die für
denselben Professor arbeiten, für den Aristoteles arbeitet:


-- ---------------------------------------
-- Aggregatfunktionen (avg,min,max,sum,...
```

```
-- und GROUP BY
-- ------------------------------------
2-16. Liste die durchschnittliche Semesterzahl:

3-17. Liste Geburtstage der gehaltsklassenältesten Professoren(ohne Namen !):

2-18. Liste Summe der SWS pro Professor(die gehaltenen SWS pro Professor)

2-19. Liste Summe der SWS pro Professor,
sofern seine Durchschnitts-SWS größer als 3 ist:
(Tipp: having)

3-20. Liste Summe der SWS pro C4-Professor,
sofern seine Durchschnitts-SWS größer als 3 ist:

-- ----------------------------
-- Subqueries
-- ----------------------------
3-21. Liste alle Prüfungen, die als Ergebnis
die Durchschnittsnote haben:

3-22. Liste alle IS_Professoren zusammen mit ihrer
Lehrbelastung:

3-23. Liste alle IS_Studenten, die älter sind als der jüngste Professor:
Korrelierte Formulierung:

Alternativ:
Äquivalente, unkorrelierte Formulierung
Vorteil: Unteranfrageergebnis kann materialisiert werden
Unteranfrage braucht nur einmal ausgewertet zu werden

3-24. Liste alle IS_Assistenten, die für einen jüngeren
Professor arbeiten:

3-25. Liste alle IS_Studenten mit der Zahl ihrer
IS_Vorlesungen, sofern diese Zahl größer als 2 ist:

3-26. Liste die Namen und Geburtstage der Gehaltsklassenältesten:

3-27. Liste IS_Vorlesungen zusammen mit Marktanteil,
definiert als = Studenten_pro_vorlesung/Studenten_insgesamt:
(Anm: cast(expression as type)

-- --------------------------------
-- mengen (union, intersect, ...)
-- --------------------------------
3-28. Liste die Vereinigung von IS_Professoren- und IS_Assistenten-Namen:

3-29. Liste die Differenz von IS_Professoren- und IS_Assistenten-Namen:
(Nicht mssql)
( select Name
  from IS_Assistenten )
minus
( select Name
  from IS_Professoren );


3-30. Liste den Durchschnitt von IS_Professoren- und IS_Assistenten-Namen:
```

```
(Nicht mssql)
( select Name from IS_Assistenten )
intersect
( select Name from IS_Professoren );


3-31. Liste alle IS_Professoren, die keine Vorlesung halten:
(Anm: not in)

3-32. Liste IS_Studenten mit größter Semesterzahl:
(Anm: >= all)

3-33. Liste IS_Studenten, die nicht die größte Semesterzahl haben:
(Anm: < some)

-- ------------------------------------------------
-- SQL-Queries zum Einfügen, Modifizieren, Löschen
-- ------------------------------------------------
1-1. Füge Student mit Matrikelnummer und Name ein:

1-2. Alle IS_Studenten sollen die Vorlesung 'Selber Atmen' hören:

1-3. Alle IS_Studenten um 10 Tage älter machen:

1-4. Alle IS_Studenten mit Semesterzahlen größer als 13 löschen:

1-5. Niemand soll mehr die Vorlesung 'Selber Atmen' hören:

-- ------------------------------------------------
-- Views
-- ------------------------------------------------
Nicht alle Sichten sind update-fähig, da sich eine Änderung
ihrer Daten nicht immer auf die Originaltabellen zurückpropagieren läßt

2-1. Lege Sicht an für Prüfungen ohne Note:

2-2. Lege Sicht an für IS_Studenten mit ihren IS_Professoren:

3-3. Lege Sicht an mit IS_Professoren und ihren Durchschnittsnoten:

-- ------------------------------------
-- INDEX
-- ------------------------------------
2-1. Erzeuge einen Index namens idx_pname bzgl Tabelle IS_Professoren und
der Spalte name.
```

+Aufgabe: sql-is_uni-fragen

Aufgabe: sql-is_uni-fragen

```
Tabelle erzeugen (hier ein mysql-code):
CREATE TABLE fragen (
  id int(11) NOT NULL auto_increment,
  thema varchar(240) default NULL,
  kapitel varchar(240) default NULL,
  frage varchar(240) default '',
```

```
  antwort text,
  hinweis text,
  PRIMARY KEY  (id)
) TYPE=MyISAM;
```

Aufgabe:
Laden Sie nun folgendes Script in ihre Datenbank.

```
INSERT INTO fragen VALUES (81,'SQL','Kapitel 01 Create table,
Primary/Foreign Keys, Constraints, ...','Frage 01 Zeigen Sie die CREATE
TABLE Anweisungen zu den Tabellen Professoren, Assistenten, Studenten,
Vorlesungen, hoeren','.','Geben Sie auch die referentielle
Integrittsregeln an.
http://www.users.sbg.ac.at/~hofmann/lva/dbkurs/is_uni.sql');
INSERT INTO fragen VALUES (82,'SQL','Kapitel 02 Grundprinzip Projektion
und Selektion','Frage 01 Liste alle Studenten','select * from
Studenten','Selektion');
INSERT INTO fragen VALUES (83,'SQL','Kapitel 02 Grundprinzip Projektion
und Selektion','Frage 02 Liste Personalnummer und Name der
Assistenten','select PersNr, Name from Assistenten','projection');
INSERT INTO fragen VALUES (84,'SQL','Kapitel 02 Grundprinzip Projektion
und Selektion','Frage 03 Liste Personalnummer und Name der C4-
Professoren','select PersNr, Name from Professoren where Rang =
\'C4\'','projection, where-prdikat');
INSERT INTO fragen VALUES (85,'SQL','Kapitel 02 Grundprinzip Projektion
und Selektion','Frage 04 Liste Namen und Studiendauer in Jahren von
allen Studenten, die eine Semesterangabe haben','select Name, Semester/2
AS Studienjahr from Studenten where Semester is not null','Arithmetik,
not null');
INSERT INTO fragen VALUES (86,'SQL','Kapitel 02 Grundprinzip Projektion
und Selektion','Frage 05 Liste alle Studenten mit Semesterzahlen
zwischen 1 und 4','select * from Studenten where Semester >= 1 and
Semester <= 4\nAlternativ: \nselect * from Studenten where Semester
between 1 and 4\nAlternativ: \nselect * from Studenten where Semester in
(1,2,3,4)\n','Boolsche Operatoren (not, and, or, x between y and b,
<,>,<=, >=, <>, IN, LIKE)');
INSERT INTO fragen VALUES (87,'SQL','Kapitel 02 Grundprinzip Projektion
und Selektion','Frage 06 Liste alle Vorlesungen, die im Titel den String
Ethik enthalten, klein oder gro geschrieben','select * from Vorlesungen
where upper(Titel) like \'%ETHIK%\'','Upper(), %, _');
INSERT INTO fragen VALUES (88,'SQL','Kapitel 02 Grundprinzip Projektion
und Selektion','Frage 07 Liste Personalnummer, Name und Rang aller
Professoren, \nabsteigend sortiert nach Rang, innerhalb des Rangs
aufsteigend sortiert nach Name','select PersNr, Name, Rang from
Professoren order by Rang desc, Name asc','Sortieren: Order by
asc/desc');
INSERT INTO fragen VALUES (89,'SQL','Kapitel 02 Grundprinzip Projektion
und Selektion','Frage 08 Liste alle verschiedenen Eintrge in der Spalte
Rang der Relation Professoren','select distinct Rang from
Professoren','Duplikatfrei (distinct)');
INSERT INTO fragen VALUES (90,'SQL','Kapitel 02 Grundprinzip Projektion
und Selektion','Frage 09 Liste alle Geburtstage der Professoren mit
ausgeschriebenem Monatsnamen','select name, to_char(GebDatum,\'month DD,
YYYY\') AS Geburtstag from studenten','Datum to_char(sysdate,
DD.MON.YYYY)');
INSERT INTO fragen VALUES (91,'SQL','Kapitel 02 Grundprinzip Projektion
und Selektion','Frage 10 Liste das Alter der Studenten in
```

```
Jahren','select name,round((sysdate - GebDatum) / 365) as
Alter_in_Jahren from studenten','Round (sysdate - gebdatum)/365');
INSERT INTO fragen VALUES (92,'SQL','Kapitel 02 Grundprinzip Projektion
und Selektion','Frage 11 Liste die Wochentage der Geburtsdaten der
Studenten','select name, to_char(GebDatum,\'day\') from
studenten','To_char(gebdatum, day)');
INSERT INTO fragen VALUES (93,'SQL','Kapitel 02 Grundprinzip Projektion
und Selektion','Frage 12 Liste die Uhrzeiten der Geburtsdaten der
Studenten','select name, to_char(GebDatum,\'hh:mi:ss\') from
studenten','To_char(gebdatum, hh:mi:ss)');
INSERT INTO fragen VALUES (94,'SQL','Kapitel 03 Verknpfung von Tabellen
(Join)','Frage 01 Welcher Professor liest Meutik?','select Name, Titel
from Professoren, Vorlesungen where PersNr = gelesenVon and Titel =
\'Meutik\'','Join/Gleichverbund, Alias Namen');
INSERT INTO fragen VALUES (95,'SQL','Kapitel 03 Verknpfung von Tabellen
(Join)','Frage 02 Liste die Namen der Studenten mit ihren
Vorlesungstiteln','select Name, Titel from Studenten, hoeren,
Vorlesungen\nwhere Studenten.MatrNr = hoeren.MatrNr and hoeren.VorlNr =
Vorlesungen.VorlNr\n\nAlternativ: \nselect s.Name, s.Titel\nfrom
Studenten s, hoeren h, Vorlesungen v\nwhere s.MatrNr = h.MatrNr and\n
h.VorlNr = v.VorlNr','Join');
INSERT INTO fragen VALUES (96,'SQL','Kapitel 03 Verknpfung von Tabellen
(Join)','Frage 03 Finde die Angestellten ANG(NAME,GEHALT, MANAGER
references ANG(ID),...) die mehr als ihre (direkten) Manager verdienen
(Ausgabe NAME,GEHALT, NAME des Managers)','Select p.name, p.gehalt,
m.name\nFrom ang p, ang m\nWhere p.mnr = m.id and\np.gehalt >
m.gehalt','Exkurs: Self-referencing');
INSERT INTO fragen VALUES (97,'SQL','Kapitel 03 Verknpfung von Tabellen
(Join)','Frage 04 Liste die Namen der Assistenten, die fr denselben
Professor arbeiten, fr den Aristoteles arbeitet','select a2.Name\nfrom
assistenten a1, assistenten a2\nwhere a2.boss = a1.boss\nand a1.name =
\'Aristoteles\'\nand a2.name != \'Aristoteles\'\n','Self-referencing');
INSERT INTO fragen VALUES (98,'SQL','Kapitel 04 GROUP BY
(Aggregatfunktionen)','Frage 01 Liste die durchschnittliche Semesterzahl
der Studenten','select avg(Semester) from Studenten','Aggregatfunktionen
(max, min, avg, sum, count)');
INSERT INTO fragen VALUES (99,'SQL','Kapitel 04 GROUP BY
(Aggregatfunktionen)','Frage 02 Liste die hchste Semesterzahl der
Studenten','select max(Semester)from Studenten','max');
INSERT INTO fragen VALUES (100,'SQL','Kapitel 04 GROUP BY
(Aggregatfunktionen)','Frage 03 Zhle alle Studenten','select count(*)
from Studenten','Count()\nCount(*)    Anzahl der
Datenstze\nCount(Spalte) Anzahl der Datenstze, die in Spalte einen Wert
haben\nCount(distinct Spalte) Anzahl der Datenstze, die in Spalte einen
versch. Wert haben\n');
INSERT INTO fragen VALUES (101,'SQL','Kapitel 04 GROUP BY
(Aggregatfunktionen)','Frage 04 Liste Geburtstage der
Gehaltsklassenltesten (ohne Namen !)','select rang, max(GebDatum)\nfrom
Professoren\ngroup by rang','Gruppierungen, Group by\nBsp: Gesucht ist
pro gehaltsklasse=Rang das Geburtsdatum des ltesten =max(gebdatum)\nPro
Gruppe wird eine Zeile ermittelt\nEine Gruppe wird durch die GROUP BY
Klausel festgelegt\nIn der selectKlausel drfen nur Konstante,
Aggregatfunktionen, Spalten (die auch in der group by-klausel sind)
vorkommen\n');
INSERT INTO fragen VALUES (102,'SQL','Kapitel 04 GROUP BY
(Aggregatfunktionen)','Frage 05 Liste Summe der SWS pro Professor (die
gehaltenen SWS pro Professor)','select gelesenVon, sum(SWS)\nfrom
Vorlesungen\ngroup by gelesenVon','group by');
```

```
INSERT INTO fragen VALUES (103,'SQL','Kapitel 04 GROUP BY
(Aggregatfunktionen)','Frage 06 Liste Summe der SWS pro Professor,
sofern seine Durchschnitts-SWS grer als 3 ist','select gelesenVon,
sum(SWS)\nfrom Vorlesungen\ngroup by gelesenVon\nhaving avg(SWS) >
3','having\n1. evtl. Join der beteiligten Tabellen\n2. Selektion der
Zeilen (WHERE)\n3. Gruppierung\n4. Selektion der Gruppen (having)\n5.
Projektion auf gesuchte Attribute\n');
INSERT INTO fragen VALUES (104,'SQL','Kapitel 04 GROUP BY
(Aggregatfunktionen)','Frage 07 Liste Summe der SWS pro C4-Professor,
sofern seine Durchschnitts-SWS grer als 3 ist','select gelesenVon, Name,
sum(SWS)\nfrom Vorlesungen, Professoren\nwhere gelesenVon = PersNr and
Rang = \'C4\'\ngroup by gelesenVon, Name\nhaving avg(SWS) > 3','.');
INSERT INTO fragen VALUES (105,'SQL','Kapitel 05
Teilabfragen/Subqueries','Frage 01 Liste alle Prfungen, die als Ergebnis
die Durchschnittsnote haben','select *\nfrom pruefen\nwhere Note =
(select round(avg(Note))\n              from pruefen)','Einfache
Teilabfrage (muss  nur einmal ausgefhrt werden)\nRound (avg(note))\n');
INSERT INTO fragen VALUES (106,'SQL','Kapitel 05
Teilabfragen/Subqueries','Frage 02 Liste alle Professoren (PersNR, Name,
Lehrbelastung) zusammen mit ihrer Lehrbelastung','select PersNr, Name,
(select sum(SWS) \n                      from Vorlesungen\n
where gelesenVon = PersNr) as Lehrbelastung\nfrom
Professoren','Korrelierte Teilabfrage\n(select Sum(sws) from vorlesungen
where ...) as Lehrbelastung)');
INSERT INTO fragen VALUES (107,'SQL','Kapitel 06 Teilabfragen und
Prdikte (ANY, ALL,IN, NOT IN, EXISTS)','Frage 01 Liste alle Professoren,
die keine Vorlesung halten','select Name\nfrom Professoren\nwhere PersNr
not in ( select gelesenVon\n                      from Vorlesungen )\n
Alternativ: \nselect Name\nfrom Professoren\nwhere not exists ( select
*\n                   from Vorlesungen\n                    where
gelesenVon = PersNr )\n','Wenn Ergebnis der Teilanfrage nicht ein
einzelnes Tupel ist, sondern eine Menge von Tupeln\nANY Bedingung muss
fr irgendein Tupel der Teilabfrage erfllt sein\nALL Bedingung muss fr
alle Tupel der Teilabfrage erfllt sein\nIN  Bedingung mit Prdikat IN ist
erfllt, wenn Ergebnismenge der Teilabfrage\n    den aktuellen Wert
enthlt\nEXISTS Bedingung ist erfllt, wenn Ergebnismenge der Teilabfrage
nicht leer ist\n');
INSERT INTO fragen VALUES (108,'SQL','Kapitel 06 Teilabfragen und
Prdikte (ANY, ALL,IN, NOT IN, EXISTS)','Frage 02 Liste Studenten mit
grter Semesterzahl','select Name\nfrom Studenten\nwhere Semester >= all
( select Semester\n                          from Studenten )','>= ALL ');
INSERT INTO fragen VALUES (109,'SQL','Kapitel 07 Mengen (union, minus,
intersect, ...)','Frage 01 Liste die Vereinigung von Professoren- und
Assistenten-Namen','( select Name\n  from Assistenten )\nunion\n( select
Name\n  from Professoren )','union');
INSERT INTO fragen VALUES (110,'SQL','Kapitel 07 Mengen (union, minus,
intersect, ...)','Frage 02 Liste die Differenz von Professoren- und
Assistenten-Namen','( select Name\n  from Assistenten )\nminus\n( select
Name\n  from Professoren )','minus');
INSERT INTO fragen VALUES (111,'SQL','Kapitel 07 Mengen (union, minus,
intersect, ...)','Frage 03 Liste den Durchschnitt von Professoren- und
Assistenten-Namen','( select Name from Assistenten )\nintersect\n(
select Name from Professoren )','.');
INSERT INTO fragen VALUES (112,'SQL','Kapitel 08 Datenmanipulation:
Insert , Update, Delete','Frage 01 Fge Student mit Matrikelnummer und
Name ein','insert into Studenten (MatrNr, Name)\n           values
(28121, \'Archimedes\')','insert');
INSERT INTO fragen VALUES (113,'SQL','Kapitel 08 Datenmanipulation:
```

```
Insert , Update, Delete','Frage 02 Alle Studenten sollen die Vorlesung
\'Selber Atmen\' hren','insert into hoeren\n   select MatrNr, VorlNr\n
from Studenten, Vorlesungen\n   where Titel = \'Selber
Atmen\'','insert');
INSERT INTO fragen VALUES (114,'SQL','Kapitel 08 Datenmanipulation:
Insert , Update, Delete','Frage 03 Alle Studenten um 10 Tage lter
machen','update studenten\nset GebDatum = GebDatum + 10','update');
INSERT INTO fragen VALUES (115,'SQL','Kapitel 08 Datenmanipulation:
Insert , Update, Delete','Frage 04 Alle Studenten mit Semesterzahlen
grer als 13 lschen','delete from Studenten\n   where Semester >
13','delete');
INSERT INTO fragen VALUES (116,'SQL','Kapitel 08 Datenmanipulation:
Insert , Update, Delete','Frage 05 Niemand soll mehr die Vorlesung
\'Selber Atmen\' hren','delete from hoeren\n   where vorlnr = \n
(select VorlNr from Vorlesungen\n   where Titel = \'Selber
Atmen\')','.');
INSERT INTO fragen VALUES (117,'SQL','Kapitel 09 Create Views','Frage 01
Lege Sicht an fr Prfungen ohne Note','create view pruefenSicht as\n
select MatrNr, VorlNr, PersNr\n   from pruefen','Nicht alle
Views/Sichten sind update-fhig, da sich eine nderung \nihrer Daten nicht
immer auf die Originaltabellen zurckpropagieren lt \n');
INSERT INTO fragen VALUES (118,'SQL','Kapitel 09 Create Views','Frage 02
Lege Sicht an fr Studenten mit ihren Professoren','create view StudProf
(Sname, Semester, Titel, PName) as\n  select s.Name, s.Semester,
v.Titel, p.Name\n  from Studenten s, hoeren h, Vorlesungen v,
Professoren p\n  where s.MatrNr = h.MatrNr and h.VorlNr = v.VorlNr\n
and v.gelesenVon = p.PersNr','.');
INSERT INTO fragen VALUES (119,'SQL','Kapitel 09 Create Views','Frage 03
Lege Sicht an mit Professoren und ihren Durchschnittsnoten','create view
ProfNote (PersNr, Durchschnittsnote) as\n  select PersNr, round(avg
(Note))\n  from pruefen\n  group by PersNr','.');
INSERT INTO fragen VALUES (120,'SQL','Kapitel 10 Create INDEX','Frage 01
Lege Index an fr Professoren Namen','CREATE INDEX PROF_NAME_IDX ON
PROFESSOREN (NAME)','.');
```

Fragen

Kunden,Rechnung: Überblick über das Datenbank-Schema

Die verwendete Datenbank besteht aus 3 Tabellen mit folgendem Aufbau:

Kunden

Attribut	*Typ*	*Bedingung*
nr	INTEGER	PRIMARY KEY
name	VARCHAR(100)	NOT NULL
strasse	VARCHAR(50)	
ort	VARCHAR(20)	

Rechnungen

Attribut	*Typ*	*Bedingung*
nr	INTEGER	PRIMARY KEY
kunde	INTEGER	FOREIGN KEY kunden(nr)

datum	DATE	NOT NULL
summe	REAL	DEFAULT 0

Positionen

Attribut	***Typ***	***Bedingung***
nr	INTEGER	PRIMARY KEY
rechnung	INTEGER	PRIMARY KEY FOREIGN KEY rechnungen(nr)
Bezeichnung	VARCHAR(50)	NOT NULL
Menge	INTEGER	
Preis	REAL	

Aufgabe 1

Schreiben Sie die für die Erstellung des oben dargestellten Datenbankschemas notwendigen SQL-Befehle (create table). Stellen Sie dabei sicher, dass Sie die Fremdschlüssel nicht vergessen!

Aufgabe 2

Schreiben Sie eine SQL-Klausel, die alle Datensätze ausgibt, in denen die Summe der Positionen ungleich der Rechnungssumme ist.

Aufgabe 3

Schreiben Sie eine SQL-Klausel, die nachfolgendes Attribut in die Tabelle kunden einfügt:
plz VARCHAR(8)

Aufgabe 4

Schreiben Sie die erforderlichen SQL-Anweisungen um nachfolgende Daten in die Datenbank einzufügen. In welcher Reihenfolge müssen Sie ausgeführt werden um keine Integritätsverletzung zu verursachen?

Kunden

Nr	***Name***	***Strasse***	***Ort***	***Plz***
1	Maier	Ahornstr. 3	Salzburg	5020
2	Schmidt	Kräuterweg 5	Salzburg	5020
3	Huber	Schwertstr. 7	Linz	4020

Rechnungen

Nummer	***Kunde***	***Datum***	***Summe***
1	1	1.1.00	300
2	1	7.5.00	230
3	2	4.4.99	650
4	3	7.7.99	550

Positionen

Nr	***Rechnung***	***Bezeichnung***	***Menge***	***Preis***
1	1	Kaffe Arabica	5	10
2	1	Milchschaeumer	1	12
1	2	Praesident	5	11
1	3	Kaffeemaschine	1	300
2	3	African Blue	3	15
1	4	Espressomat	1	500

Aufgabe 5

Formulieren Sie eine SQL-Klausel, die alle Rechnungspositionen von Herrn Maier ausgibt.

Aufgabe 6

Formulieren Sie eine SQL-Klausel, die die Anzahl der Bestellungen von Herrn Maier ausgibt.

Aufgabe 7

Formulieren Sie eine SQL-Klausel, die die kumulierten Umsätze nach Postleitzahlen gruppiert ausgibt.

Aufgabe 8

Geben Sie all jene Rechnungen aus, bei denen mehr als 4 Stück (Menge) eingekauft wurden.

Weitere Aufgaben

Aufgabe: SQL-Grundlagen

Aufgabe: SQL-Grundlagen

Studieren Sie den Inhalt von
http://www.torsten-horn.de/techdocs/sql-examples.htm

und installieren Sie auf ihrem Server folg. SQL-Script und testen Sie die einzelnen SQL-Befehle:

```
DROP TABLE Personen;

CREATE TABLE Personen
(
  id          INTEGER      NOT NULL,
  Name        VARCHAR(32)  NOT NULL,
  Abteilung   VARCHAR(32),
  Tel         VARCHAR(32),
  Plz         INTEGER,
  Ort         VARCHAR(32),
  UNIQUE ( id ),
  PRIMARY KEY ( id )
);

INSERT INTO Personen VALUES (  01, 'Gabi',          'Sekr', '100', null,
'' );
INSERT INTO Personen VALUES (  02, 'Tatiana',       'GF',   '202', null,
'' );
INSERT INTO Personen VALUES (  03, 'Manfred',       'GF',   '201', null,
'' );
INSERT INTO Personen VALUES (  04, 'Iris',          'MK',   '401', null,
'' );
INSERT INTO Personen VALUES (  05, 'René',          'PM',   '501', null,
'' );
INSERT INTO Personen VALUES (  06, 'Elena',         'MM',   '301', null,
```

```
'' );
INSERT INTO Personen VALUES (  07, 'Aidar',         'MM',   '301', null,
'' );
INSERT INTO Personen VALUES (  08, 'Oleg',          'MM',   '301', null,
'' );
INSERT INTO Personen VALUES (  09, 'Andreas',       'Entw', '610',
52525, 'Heinsberg' );
INSERT INTO Personen VALUES (  10, 'Arthur',        'Entw', '612',
52499, 'Baesweiler' );
INSERT INTO Personen VALUES (  11, 'Gregor',        'Entw', '611',
52351, 'Düren' );
INSERT INTO Personen VALUES (  12, 'Michael',       'Entw', '616',
50859, 'Köln' );
INSERT INTO Personen VALUES (  13, 'Norbert',       'Entw', '614',
52134, 'Herzogenrath' );
INSERT INTO Personen VALUES (  14, 'Roland',        'Entw', '601',
52134, 'Herzogenrath' );
INSERT INTO Personen VALUES (  15, 'Stefan',        'Entw', '613',
52062, 'Aachen' );
INSERT INTO Personen VALUES (  16, 'Torsten',       'Entw', '601',
52072, 'Aachen' );
INSERT INTO Personen VALUES (  17, 'Werner',        'Entw', '615',
52076, 'Aachen' );
INSERT INTO Personen VALUES (  18, 'FAX',           null,   '101', null,
'' );
INSERT INTO Personen VALUES (  19, 'Konferenzraum', null,   '700', null,
'' );

-------------------------------------------------------------------------
-----------;

DROP TABLE Speisen;

CREATE TABLE Speisen
(
  id         INTEGER       NOT NULL,
  Gericht    VARCHAR(255)  NOT NULL,
  Preis      FLOAT         NOT NULL,
  Zutaten    VARCHAR(255),
  UNIQUE ( id ),
  UNIQUE ( Gericht ),
  PRIMARY KEY ( id )
);

INSERT INTO Speisen VALUES (  101, 'Pizza Diabolo', 5.50, 'Teufelsohren'
);
INSERT INTO Speisen VALUES (  102, 'Pizza Vulkano', 6.00, 'Teig, Käse,
Vesuvtomaten' );
INSERT INTO Speisen VALUES (  103, 'Pizza Feuro', 6.50, 'Pepperoni' );
INSERT INTO Speisen VALUES (  104, 'Lasagno', 6, 'Nudeln, Hackfleisch'
);
INSERT INTO Speisen VALUES (  105, 'Salat Eskimo', 4.50, 'Eiswürfel' );

-------------------------------------------------------------------------
-----------;

DROP TABLE Bestellung;
```

```
CREATE TABLE Bestellung
(
  id         INTEGER  NOT NULL,
  id_Kunde   INTEGER  NOT NULL,
  id_Speise  INTEGER  NOT NULL,
  UNIQUE ( id ),
  PRIMARY KEY ( id )
);

INSERT INTO Bestellung VALUES (  01, 09, 105 );
INSERT INTO Bestellung VALUES (  02, 11, 103 );
INSERT INTO Bestellung VALUES (  03, 12, 103 );
INSERT INTO Bestellung VALUES (  04, 14, 103 );
INSERT INTO Bestellung VALUES (  05, 15, 101 );
INSERT INTO Bestellung VALUES (  06, 15, 102 );
INSERT INTO Bestellung VALUES (  07, 15, 103 );
INSERT INTO Bestellung VALUES (  08, 16, 104 );
INSERT INTO Bestellung VALUES (  09, 17, 103 );

-------------------------------------------------------------------------
-----------;
SELECT * FROM Personen ORDER BY id;
SELECT * FROM Personen ORDER BY Abteilung, Name;
SELECT * FROM Personen ORDER BY Abteilung DESC, Name;

SELECT Name, Ort FROM Personen WHERE Ort = 'aaCHen';

SELECT * FROM Personen WHERE Abteilung = 'Entw' AND (Ort = 'Düren' OR
Ort = 'Köln');
SELECT * FROM Personen WHERE Abteilung <> 'Entw';
SELECT * FROM Personen WHERE Ort LIKE '%zog%';
SELECT * FROM Personen WHERE Tel LIKE '__0';
SELECT * FROM Personen WHERE Plz BETWEEN 52351 AND 52499;
SELECT * FROM Personen WHERE Abteilung IS NULL;

SELECT DISTINCT Abteilung FROM Personen WHERE Abteilung IS NOT NULL;

SELECT COUNT(*) FROM Personen WHERE Abteilung = 'Entw';
SELECT Abteilung, COUNT(*) "Anzahl Personen pro Abteilung"
 FROM Personen
 GROUP BY Abteilung;

SELECT Abteilung, COUNT(*)
 FROM Personen
 GROUP BY Abteilung
 HAVING COUNT(*) >= 3;
-------------------------------------------------------------------------
-----------;

SELECT * FROM Speisen ORDER BY id;
SELECT MIN(PREIS), MAX(PREIS), AVG(PREIS), SUM(PREIS), COUNT(PREIS)
 FROM Speisen;

-------------------------------------------------------------------------
-----------;
SELECT * FROM Bestellung ORDER BY id;

SELECT * FROM Bestellung, Personen, Speisen
```

```
 WHERE id_Kunde = Personen.id AND id_Speise = Speisen.id;

SELECT Name, Gericht, Preis
 FROM Bestellung, Personen, Speisen
 WHERE id_Kunde = Personen.id AND id_Speise = Speisen.id
 ORDER BY Name;


SELECT Name, COUNT(*), SUM(Preis)
 FROM Bestellung, Personen, Speisen
 WHERE id_Kunde = Personen.id AND id_Speise = Speisen.id
 GROUP BY Name
 ORDER BY Name;

SELECT Gericht, COUNT(*), SUM(Preis)
  FROM Bestellung, Personen, Speisen
 WHERE id_Kunde = Personen.id AND id_Speise = Speisen.id
 GROUP BY Gericht ORDER BY Gericht;
-------------------------------------------------------------------------
-----------;
```

Aufgabe: sqlzoo

Aufgabe: sqlzoo

Gehen Sie zu http://sqlzoo.net und trainieren Sie ihre SQL-Kenntnisse.

Ausblick

In der Folge wollen wir die Begriffe Referentiellen Integrität und Normalformen kennen lernen.