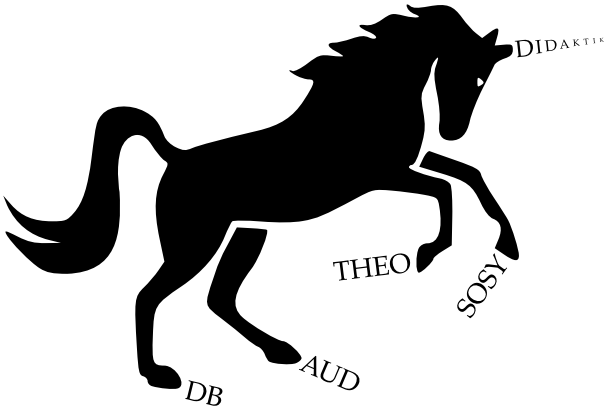
46116
Herbst
2018

Softwaretechnologie / Datenbanksysteme (nicht vertieft)

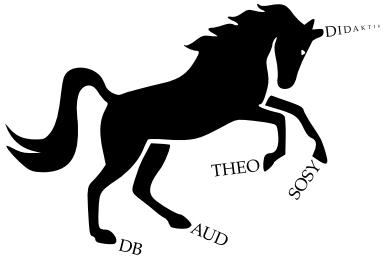
Aufgabenstellungen mit Lösungsvorschlägen



Die Bschlangaul-Sammlung

Hermine Bschlangaul and Friends

Aufgabenübersicht



Die Bschlangaul-Sammlung
Hermine Bschlangaul and Friends

Eine freie Aufgabensammlung mit Lösungen von Studierenden für Studierende zur Vorbereitung auf die 1. Staatsexamensprüfungen des Lehramts Informatik in Bayern.

Thema Nr. 1

Teilaufgabe Nr. 1

Aufgabe 3 [Verhaltens-Modellierung mit Zustandsdiagrammen. Digitaluhr]

Eine Digitaluhr kann alternativ entweder die Zeit (Stunden und Minuten) oder das Datum (Tag, Monat und Jahr) anzeigen. Zu Beginn zeigt die Uhr die Zeit an. Sie besitzt drei Druckknöpfe **A**, **B** und **C**. Mit Knopf **A** kann zwischen Zeit- und Datumsanzeige hin und her gewechselt werden.

Wird die Zeit angezeigt, dann kann mit Knopf **B** der Reihe nach erst in einen Stundenmodus, dann in einen Minutenmodus und schließlich zurück zur Zeitanzeige gewechselt werden. Im Stundenmodus blinkt die Stundenanzeige. Mit Drücken des Knopfes **C** können dann die Stunden schrittweise inkrementiert werden. Im Minutenmodus blinkt die Minutenanzeige und es können mit Hilfe des Knopfes **C** die Minuten schrittweise inkrementiert werden.

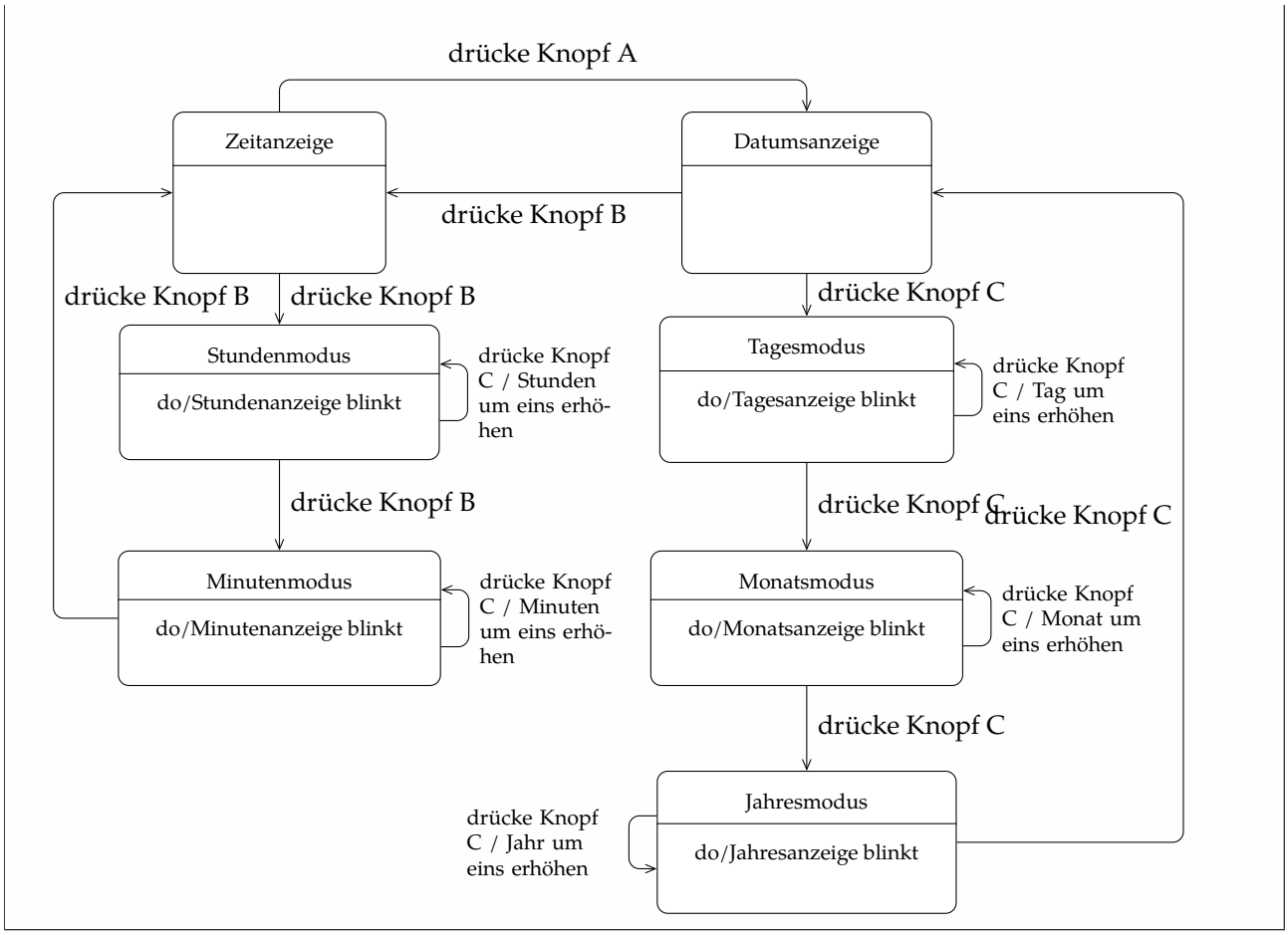
Die Datumsfunktionen sind analog. Wird das Datum angezeigt, dann kann mit Knopf **B** der Reihe nach in einen Tagesmodus, Monatsmodus, Jahresmodus und schließlich zurück zur Datumsanzeige gewechselt werden. Im Tagesmodus blinkt die Tagesanzeige. Mit Drücken des Knopfes **C** können dann die Tage schrittweise inkrementiert werden. Analog blinken mit Eintritt in den entsprechenden Einstellmodus der Monat oder das Jahr, die dann mit Knopf **C** schrittweise inkrementiert werden können.

Wenn sich die Uhr in einem Einstellmodus befindet, hat das Betätigen des Knopfes **A** keine Wirkung. Ebenso wirkungslos ist Knopf **C**, wenn gerade Zeit oder Datum angezeigt wird.

Beschreiben Sie das Verhalten der Digitaluhr durch ein UML-Zustandsdiagramm. Dabei muss - gemäß der UML-Notation - unterscheidbar sein, was Ereignisse und was Aktionen sind. Deren Bedeutung soll durch die Verwendung von sprechenden Namen klar sein. Für die Inkrementierung von Stunden, Minuten, Tagen etc. brauchen keine konkreten Berechnungen angegeben werden. Der kontinuierliche Zeitfortschritt des Uhrwerks ist nicht zu modellieren.

Zustände sind, wie in der UML üblich, durch abgerundete Rechtecke darzustellen. Sie können unterteilt werden in eine obere und eine untere Hälfte, wobei der Name des Zustands in den oberen Teil und eine in dem Zustand auszuführende Aktivität in den unteren Teil einzutragen ist.

Lösungsvorschlag



Aufgabe 4 [Kundenverwaltungssystem]

Gegeben sind folgende Relationen aus einem Kundenverwaltungssystem:

```
Kunde : {[ ID, Vorname, Nachname, PLZ ]}

Produkt : {[ GTIN, Bezeichnung, Bruttopreis, MWStSatz) ]}

Kauf : {[ ID[Kunde], GTIN[Produkt], Datum, Menge ]}
```

Verwenden Sie im Folgenden nur Standard-SQL und keine produktspezifischen Erweiterungen. Sie dürfen bei Bedarf Views anlegen. Geben Sie einen Datensatz, also eine Entity, nicht mehrfach aus.

(a) Schreiben Sie eine SQL-Anweisung, die die Tabelle *„Kauf“* anlegt. Gehen Sie davon aus, dass die Tabellen *„Kunde“* und *„Produkt“* bereits existieren.

Lösungsvorschlag

```sql
CREATE TABLE IF NOT EXISTS Kauf (
  ID INTEGER REFERENCES Kunde(ID),
  GTIN INTEGER REFERENCES Produkt(GTIN),
  Datum DATE,
  Menge INTEGER,
  PRIMARY KEY (ID, GTIN, Datum)
```

```
);
```

(b) Schreiben Sie eine SQL-Anweisung, die *Vorname* und *Nachname* aller *Kunden* mit der *Postleitzahl 20251* ausgibt, absteigend sortiert nach *Nachname* und bei gleichen *Nachnamen*, absteigend nach *Vorname*.

Lösungsvorschlag

```
SELECT Vorname, Nachname
FROM Kunde
WHERE PLZ = 20251
ORDER BY Nachname DESC, Vorname DESC;


 vorname | nachname
---------+----------
 Hanna   | Winter
 Jakob   | Sommer
 Bert    | Sommer
(3 rows)
```

(c) Schreiben Sie eine SQL-Anweisung, die zu jedem Einkauf mit mehr als 10 unterschiedlichen Produkten den *Nachnamen* des *Kunden* und den *Bruttogesamtpreis* des Einkaufs ausgibt. Ein Einkauf ist definiert als Menge aller Produkte, die ein bestimmter Kunde an einem bestimmten Datum kauft.

Lösungsvorschlag

```
SELECT Nachname, SUM(Bruttopreis * Menge)
FROM Kunde k, Produkt p, Kauf x
WHERE k.ID = x.ID AND p.GTIN = x.GTIN
GROUP BY Datum, Nachname, k.ID
HAVING COUNT (*) > 10;


  nachname   |  sum
-------------+--------
 Mustermann | 713.86
(1 row)
```

(d) Schreiben Sie eine SQL-Anweisung, die die *GTINs* aller Produkte ausgibt, die an mindestens einen in der Datenbank enthaltenen PLZ-Bereich noch nie verkauft worden sind. Als in der Datenbank enthaltener PLZ-Bereich gelten alle in der Tabelle *„Kunde“* enthaltenen PLZs. Ein Produkt gilt als an einen PLZ-Bereich verkauft, sobald es von mindestens einem Kunden aus diesem PLZ-Bereich gekauft wurde. Produkte, die bisher noch gar nicht verkauft worden sind, müssen nicht berücksichtigt werden.

Die beiden Lösungswege liefern leider unterschiedliche Ergebnisse.

```
WITH tmp AS (
  SELECT x.GTIN, k.PLZ
  FROM Kunde k, Kauf x
  WHERE x.ID = k.ID
```

```
  GROUP BY x.GTIN, k.PLZ
)

SELECT DISTINCT GTIN
FROM tmp
WHERE EXISTS (
  SELECT Kunde.PLZ
  FROM Kunde LEFT OUTER JOIN tmp
  ON Kunde.PLZ = tmp.PLZ
  WHERE tmp.PLZ IS NULL
)
ORDER BY GTIN;
```

```
 gtin
------
    4
   23
  112
  113
  123
  124
  125
  155
  189
  324
  453
  765
(12 rows)
```

oder

```
SELECT DISTINCT GTIN FROM (
  (
    SELECT GTIN, PLZ
    FROM Kunde, Produkt
  )
  EXCEPT
  (
    SELECT x.GTIN, k.PLZ
    FROM Kunde k, Kauf x
    WHERE x.ID = k.ID
    GROUP BY x.GTIN, k.PLZ
  )
) as tmp
ORDER BY GTIN;
```

```
 gtin
------
    4
```

```
   23
  112
  113
  123
  124
  125
  155
  189
  324
  453
  765
  889
(13 rows)
```

(e) Schreiben Sie eine SQL-Anweisung, die die Top-Ten der am meisten verkauften Produkte ausgibt. Ausgegeben werden sollen der Rang (1 bis 10) und die Bezeichnung des Produkts. Gehen Sie davon aus, dass es keine zwei Produkte mit gleicher Verkaufszahl gibt und verwenden Sie keine produktspezifischen Anweisungen wie beispielsweise `ROWNUM`, `TOP` oder `LIMIT`.

Lösungsvorschlag

```sql
WITH Gesamtverkauf AS (
  SELECT k.GTIN, Bezeichnung, SUM(Menge) AS Gesamtmenge
  FROM Produkt p, Kauf k
  WHERE p.GTIN = k.GTIN
  GROUP BY k.GTIN, Bezeichnung
)

SELECT g1.Bezeichnung, COUNT (*) AS Rang
FROM Gesamtverkauf g1, Gesamtverkauf g2
WHERE g1.Gesamtmenge <= g2.Gesamtmenge
GROUP BY g1.GTIN, g1.Bezeichnung
HAVING COUNT (*) <= 10
ORDER BY Rang;
```

```
 bezeichnung  | rang
--------------+------
 Topf         |    1
 Kaffee       |    2
 Sonnenbrille |    3
 T-Shirt      |    4
 Klopapier    |    5
 Duschgel     |    6
 Hammer       |    7
 Heft         |    8
(8 rows)
```

(f) Schreiben Sie eine SQL-Anweisung, die alle Produkte löscht, die noch nie gekauft wurden.

```
 count
-------
    13
(1 row)

SELECT COUNT(*) FROM Produkt;

DELETE FROM Produkt
WHERE GTIN NOT IN
(
  SELECT DISTINCT GTIN
  FROM Kauf
);

SELECT COUNT(*) FROM Produkt;


 count
-------
    12
(1 row)
```

Teilaufgabe Nr. 2

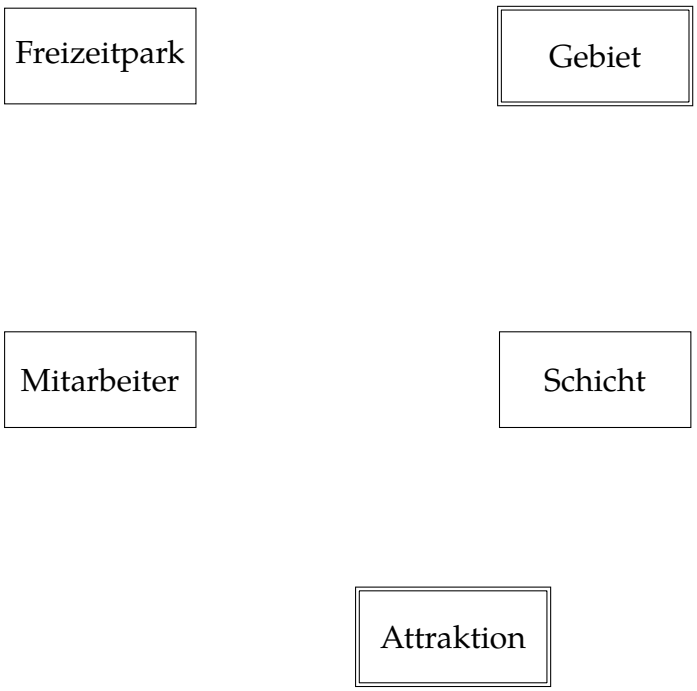
Aufgabe 2: ER-Diagramm [Freizeitparks]

Im Folgenden finden Sie die Beschreibung eines Systems zur Verwaltung von Freizeitparks. Erstellen Sie zu dieser Beschreibung ein erweitertes ER-Diagramm. Kennzeichnen Sie die Primärschlüssel durch passendes Unterstreichen und geben Sie die Kardinalitäten in Chen-Notation (= Funktionalitäten) an. Kennzeichnen Sie auch die totale Teilnahme (= Existenzabhängigkeit, Partizipität) von Entitytypen.

- Der **Freizeitpark** ist in mehrere Gebiete eingeteilt.
- Ein **Gebiet** hat einen eindeutigen *Namen* und eine *Beschreibung*.
- In jedem Gebiet gibt es eine oder mehrere **Attraktionen**. Diese verfügen über eine innerhalb ihres Gebiets eindeutige *Nummer*. Außerdem gibt es zu jeder Attraktion einen *Namen*, eine *Beschreibung* und ein oder mehrere Fotos.
- Der Freizeitpark hat **Mitarbeiter**. Zu diesen werden jeweils eine eindeutige *ID*, der *Vorname* und der *Nachname* gespeichert. Weiterhin hat jeder Mitarbeiter ein *Geburtsdatum*, das sich aus *Tag*, *Monat* und *Jahr* zusammensetzt.
- Die Arbeit im Freizeitpark ist in **Schichten** organisiert. Eine Schicht kann eindeutig durch das *Datum* und die *Startzeit* identifiziert werden. Jede Schicht hat weiterhin eine *Dauer*.
- Mitarbeiter können in Schichten an Attraktionen arbeiten. Dabei wird die *Aufgabe* ge-

□ **E: Freizeitpark**
◇ R: eingeteilt
□ **E: Gebiet**
○ *A: Namen*
○ *A: Beschreibung*
◇ R: gibt
□ **E: Attraktionen**
○ *A: Nummer*
○ *A: Namen*
○ *A: Beschreibung*
◇ R: hat
□ **E: Mitarbeiter**
○ *A: ID*
○ *A: Vorname*
○ *A: Nachname*
○ *A: Geburts-*

speichert, die der Mitarbeiter übernimmt. Pro Schicht kann der selbe Mitarbeiter nur an maximal einer Attraktion arbeiten.



Aufgabe 4 [Kundenverwaltungssystem]

Gegeben sind folgende Relationen aus einem Kundenverwaltungssystem:

```
Kunde (ID, Vorname, Nachname, PLZ)
Produkt (GTIN, Bezeichnung, Bruttopreis, MWStSatz)
Kauf (ID[Kunde], GTIN[Produkt], Datum, Menge)
```

(a) Schreiben Sie eine SQL-Anweisung, die *Vorname* und *Nachname* aller *Kunden* mit der Postleitzahl *20251* ausgibt, *absteigend* sortiert nach *Nachname* und bei gleichen Nachnamen absteigend nach *Vorname*.

ASC (ascending) = aufsteigend DESC (descending) = absteigend

Lösungsvorschlag

```sql
SELECT Vorname, Nachname
FROM Kunde
WHERE PLZ = 20251
ORDER BY Nachname, Vorname DESC;
```

(b) Schreiben Sie eine SQL-Anweisung, die die Bezeichnung aller Produkte ausgibt, deren Bruttopreis größer ist als 10 €.

Lösungsvorschlag

```sql
SELECT Bezeichnung
FROM Produkt
WHERE Bruttopreis > 10;
```

(c) Schreiben Sie eine SQL-Anweisung, die die Tabelle „Kauf“ anlegt. Gehen Sie davon aus, dass die Tabellen „Kunde“ und „Produkt“ bereits existieren.

Lösungsvorschlag

```
CREATE TABLE Kauf (
  ID INTEGER REFERENCES Kunde(ID),
  GTIN INTEGER REFERENCES Produkt(GTIN),
  Datum DATE,
  Menge INTEGER,
  PRIMARY KEY (ID, GTIN, Datum)
);
```

Thema Nr. 2

Teilaufgabe Nr. 2

Aufgabe 2 [Relationen R, S und T]

Geben Sie die Ergebnisrelation folgender Ausdrücke der relationalen Algebra als Tabellen an. Begründen Sie Ihr Ergebnis, gegebenenfalls durch Zwischenschritte. Gegeben seien folgende Relationen:

R

A	B	C	D	E	F
6	8	1	7	3	7
5	3	4	4	5	7
0	6	3	0	1	7

S

A	C	X	Z
7	8	6	1
0	3	0	0
2	3	0	5
0	6	1	6
6	7	1	7
7	1	2	2
1	8	8	0
5	1	5	5
7	3	0	2
4	8	2	7

T

X	Y
5	3
0	5
8	6
3	6
5	7
2	8

(a) $\sigma_{A>6}(S) \bowtie_{S.X=T.Y} \pi_Y(T)$

Lösungsvorschlag

A	C	X	Z	Y
7	8	6	1	6

(b) $\pi_{A,C}(S) - (\pi_A(R) \times \pi_C(\sigma_{x=1}(S)))$

Lösungsvorschlag

$\sigma_{x=1}(S)$:

A	C	X	Z
0	6	1	6
6	7	1	7

$\pi_C(\sigma_{x=1}(S))$:

C
6
7

$\pi_A(R)$:

A
6
5
0

$(\pi_A(R) \times \pi_C(\sigma_{x=1}(S)))$

A	C
6	6
5	6
0	6
6	7
5	7
0	7

$\pi_{A,C}(S)$

A	C
7	8
0	3
2	3
0	6
6	7
7	1
1	8
5	1
7	3
4	8

A	C
7	8
0	3
2	3
7	1
1	8
5	1
7	3
4	8

(c) $(\pi_D(R) \times \pi_E(R)) \div \pi_E(R)$

Lösungsvorschlag

$\pi_D(R) \times \pi_E(R)$ $\pi_E(R)$ $(\pi_D(R) \times \pi_E(R)) \div \pi_E(R)$

A	E
7	3
4	3
0	3
7	5
4	5
0	5
7	1
4	1
0	1

E
3
5
1

D
7
4
0

Aufgabe 3 [Schuldatenbank]

Gegeben sei das folgende Datenbank-Schema, das für die Speicherung der Daten einer Schule entworfen wurde, zusammen mit einem Teil seiner Ausprägung. Die Primärschlüssel-Attribute sind jeweils unterstrichen.

Die Relation *Schüler* enthält allgemeine Daten zu den Schülerinnen und Schülern. Schülerinnen und Schüler nehmen an Prüfungen in verschiedenen Unterrichtsfächern teil und erhalten dadurch Noten. Diese werden in der Relation *Noten* abgespeichert. Prüfungen haben ein unterschiedliches Gewicht. Beispielsweise hat ein mündliches Ausfragen oder eine Extemporale das Gewicht 1, während eine Schulaufgabe das Gewicht 2 hat.

Schüler:

SchülerID	**Vorname**	**Nachname**	**Klasse**
1	Laura	Müller	4A
2	Linus	Schmidt	4A
3	Jonas	Schneider	4A
4	Liam	Fischer	4B
5	Tim	Weber	4B
6	Lea	Becker	4B
7	Emilia	Klein	4C
8	Julia	Wolf	4C

Noten:

SchülerID[Schüler]	Schulfach	Note	Gewicht	Datum
1	Mathematik	3	2	23.09.2017
1	Mathematik	1	1	03.10.2017
1	Mathematik	2	2	15.10.2017
1	Mathematik	4	1	11.11.2017

(a) Geben Sie die SQL-Befehle an, die notwendig sind, um die oben dargestellten Tabellen in einer SQL-Datenbank anzulegen.

Lösungsvorschlag

```sql
CREATE TABLE IF NOT EXISTS Schüler (
  SchülerID INTEGER PRIMARY KEY NOT NULL,
  Vorname VARCHAR(20),
  Nachname VARCHAR(20),
  Klasse VARCHAR(5)
);

CREATE TABLE IF NOT EXISTS Noten (
  SchülerID INTEGER NOT NULL,
  Schulfach VARCHAR(20),
  Note INTEGER,
  Gewicht INTEGER,
  Datum DATE,
  PRIMARY KEY (SchülerID, Schulfach, Datum),
  FOREIGN KEY (SchülerID) REFERENCES Schüler(SchülerID)
);
```

(b) Entscheiden Sie jeweils, ob folgende Einfügeoperationen vom gegebenen Datenbanksystem (mit der angegebenen Ausprägungen) erfolgreich verarbeitet werden können und begründen Sie Ihre Antwort kurz.

```sql
INSERT INTO Schüler
  (SchülerID, Vorname, Nachname, Klasse)
VALUES
  (6, 'Johannes', 'Schmied', '4C');
```

Lösungsvorschlag

Nein. Ein/e Schüler/in mit der ID 6 existiert bereits. Primärschlüssel müssen eindeutig sein.

```sql
INSERT INTO Noten VALUES (6, 'Chemie', 1, 2, '1.4.2020');
```

Lösungsvorschlag

Nein. Ein *Datum* ist zwingend notwendig. Da *Datum* im Primärschlüssel enthalten ist, darf es nicht NULL sein. Es gibt auch keine/n Schüler/in mit der ID 9. Der/die Schüler/in müsste vorher angelegt werden, da die Spalte *SchülerID* von der Tabelle *Noten* auf den Fremdschlüssel *SchülerId* aus der Schülertabelle verweist.

(c) Geben Sie die Befehle für die folgenden Aktionen in SQL an. Beachten Sie dabei, dass die Befehle auch noch bei Änderungen des oben gegebenen Datenbankzustandes korrekte Ergebnisse zurückliefern müssen.

- Die Schule möchte verhindern, dass in die Datenbank mehrere Kinder mit dem selben Vornamen in die gleiche Klasse kommen. Dies soll bereits auf Datenbankebene verhindert werden. Dabei sollen die Primärschlüssel nicht verändert werden. Geben Sie den Befehl an, der diese Änderung durchführt.

  Lösungsvorschlag

```
ALTER TABLE Schüler
ADD CONSTRAINT eindeutiger_Vorname UNIQUE (Vorname, Klasse);
```

- Der Schüler *Tim Weber* (SchülerID: 5) wechselt die Klasse. Geben Sie den SQL-Befehl an, der den genannten Schüler in die Klasse *„4C“* überführt.

  Lösungsvorschlag

```
UPDATE Schüler
SET Klasse = '4C'
WHERE
  Vorname = 'Tim' AND
  Nachname = 'Weber' AND
  SchülerID = 5;
```

- Die Schülerin *Laura Müller* (SchülerID: 1) zieht um und wechselt die Schule. Löschen Sie die Schülerin aus der Datenbank. Nennen Sie einen möglichen Effekt, welcher bei der Verwendung von Primär- und Fremdschlüsseln auftreten kann.

  Lösungsvorschlag

  Alle Noten von *Laura Müller* werden gelöscht, falls `ON DELETE CASCADE` gesetzt ist. Oder es müssen erst alle Fremdschlüsselverweise auf diese *SchülerID* in der Tabelle *Noten* gelöscht werden

```
DELETE FROM Noten
WHERE SchülerID = 1;
```

- Erstellen Sie eine View *„DurchschnittsNoten“*, die die folgenden Spalten beinhaltet: *Klasse*, *Schulfach*, *Durchschnittsnote*

  Hinweis: Beachten Sie die Gewichte der Noten.

  Lösungsvorschlag

```
CREATE VIEW DurchschnittsNoten AS (
  (SELECT s.Klasse, n.Schulfach, (SUM(n.Note * n.Gewicht) / SUM(n.Gewicht))
↪  AS Durchschnittsnote
  FROM Noten n, Schüler s
  WHERE s.SchülerID = n.SchülerID
  GROUP BY s.Klasse, n.Schulfach)
);

SELECT * FROM DurchschnittsNoten;
 klasse | schulfach  | durchschnittsnote
--------+------------+-------------------
```

```
 4A     | Mathematik |                2
(1 row)
```

- Geben Sie den Befehl an, der die komplette Tabelle *„Noten“* löscht.

Lösungsvorschlag

```
DROP TABLE Noten;
```

(d) Formulieren Sie die folgenden Anfragen in SQL. Beachten Sie dabei, dass sie SQL-Befehle auch noch bei Änderungen der Ausprägung die korrekten Anfrageergebnisse zurückgeben sollen.

- Gesucht ist die durchschnittliche Note, die im Fach Mathematik vergeben wird. Hinweis: Das Gewicht ist bei dieser Anfrage nicht relevant

Lösungsvorschlag

```
SELECT AVG(Note)
FROM Noten
WHERE Schulfach = 'Mathematik';
```

- Berechnen Sie die Anzahl der Schüler, die im Fach Mathematik am 23.09.2017 eine Schulaufgabe (ðGewicht=2) geschrieben haben.

Lösungsvorschlag

```
SELECT COUNT(*) AS Anzahl_Schüler
FROM Noten
WHERE Datum = '23.09.2017' AND Gewicht = 2 AND Schulfach = 'Mathematik';
 anzahl_schüler
----------------
              1
(1 row)
```

- Geben Sie die *SchülerID* aller Schüler zurück, die im Fach Mathematik mindestens drei mal die Schulnote 6 geschrieben haben.

Lösungsvorschlag

```
SELECT SchülerID
FROM Noten
WHERE Schulfach = 'Mathematik' AND Note = 6
GROUP BY SchülerID
HAVING COUNT(*) >= 3;
 schülerid
-----------
(0 rows)
```

- Gesucht ist der Notendurchschnitt bezüglich jedes Fachs der Klasse „4A“.

Lösungsvorschlag

```
SELECT n.Schulfach, AVG(n.Note)
FROM Schüler s, Noten n
WHERE s.SchülerID = n.SchülerID AND s.Klasse = '4A'
GROUP BY n.Schulfach;
schulfach  |        avg
-----------+--------------------
```

```
 Mathematik | 2.5000000000000000
(1 row)
```

(e) Geben Sie jeweils an, welchen Ergebniswert die folgenden SQL-Befehle für die gegebene Ausprägung zurückliefern.

```
SELECT COUNT(DISTINCT Klasse)
FROM
Schüler NATURAL JOIN Noten;
```

Lösungsvorschlag

```
 count
-------
     1
(1 row)
```

4A von Laura Müller. Ohne `DISTINCT` wäre das Ergebnis 4.

```
SELECT COUNT(ALL Klasse)
FROM
Noten, Schüler;
```

Lösungsvorschlag

```
 count
-------
    32
(1 row)
```

Es entsteht das Kreuzprodukt $(8 \cdot 4 = 32)$.

```
SELECT COUNT(Note)
FROM
Schüler NATURAL LEFT OUTER JOIN Noten;
```

Lösungsvorschlag

```
SELECT * FROM Schüler NATURAL LEFT OUTER JOIN Noten;
```

ergibt:

```
 schülerid | vorname | nachname  | klasse | schulfach  | note | gewicht |   datum
-----------+---------+-----------+--------+------------+------+---------+------------
         1 | Laura   | Müller    | 4A     | Mathematik |    3 |       2 | 2017-09-23
         1 | Laura   | Müller    | 4A     | Mathematik |    1 |       1 | 2017-10-03
         1 | Laura   | Müller    | 4A     | Mathematik |    2 |       2 | 2017-10-15
         1 | Laura   | Müller    | 4A     | Mathematik |    4 |       1 | 2017-11-11
         2 | Linus   | Schmidt   | 4A     |            |      |         |
         5 | Tim     | Weber     | 4B     |            |      |         |
         8 | Julia   | Wolf      | 4C     |            |      |         |
         6 | Lea     | Becker    | 4B     |            |      |         |
         4 | Liam    | Fischer   | 4B     |            |      |         |
         3 | Jonas   | Schneider | 4A     |            |      |         |
         7 | Emilia  | Klein     | 4C     |            |      |         |
(11 rows)

 count
-------
     4
(1 row)
```

`COUNT` zählt die `{NULL}`-`Werte` nicht mit. Die *Laura Müller* hat 4 Noten.

```
SELECT COUNT(*)
FROM
Schüler NATURAL LEFT OUTER JOIN Noten;
```

Lösungsvorschlag

```
 count
-------
    11
(1 row)
```

Siehe Zwischenergebistabelle in der obenstehenden Antwort. Alle Schüler und die Laura 4-mal, weil sie 4 Noten hat.

: ER-Modell [Schulverwaltung]

Überführen Sie das Datenbankschema in ein ER-Diagramm. Verwenden Sie hierfür die bereits eingezeichneten Entity-Typen und Relationship-Typen. Weisen Sie die Relationen zu und schreiben Sie deren Namen in die dazugehörigen Felder. Fügen Sie, falls erforderlich, Attribute hinzu und beschriften Sie die Beziehungen. Markieren Sie Schlüsselattribute durch unterstreichen.

Gegeben sei das folgende Datenbankschema, wobei Primärschlüssel unterstrichen und Fremdschlüssel überstrichen sind. Die von einem Fremdschlüssel referenzierte Relation ist in eckigen Klammem nach dem Fremdschlüsselattribut angegeben.

Schüler (SchülerID, SVorname, SNachname, $\overline{\text{KlassenID}}$[Klassen], Geburtsdatum)

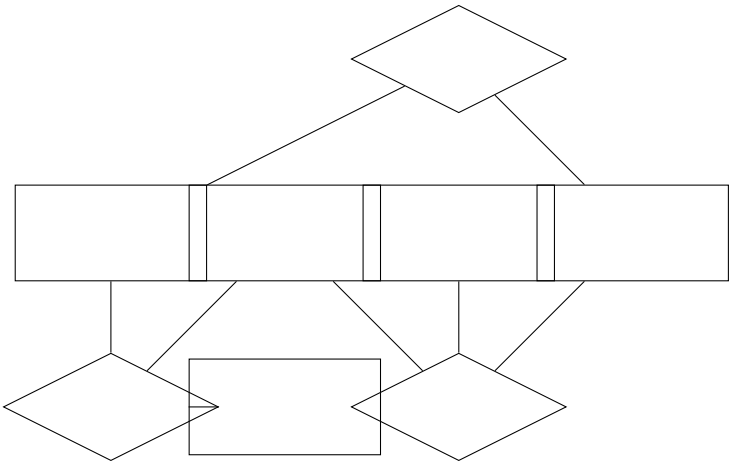
Lehrer (LehrerID, LVorname, LNachname)

Klassen (KlassenID, Klassenstufe, Buchstabe)

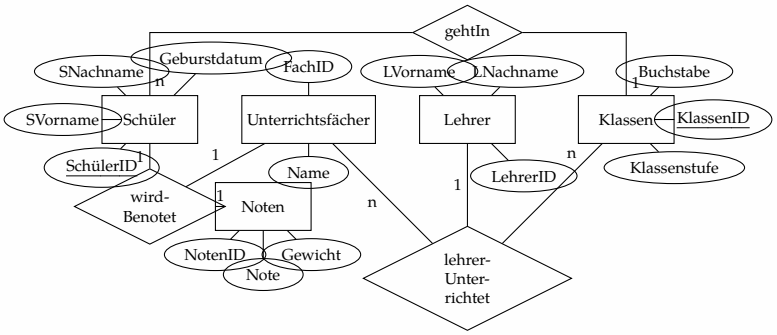
Unterrichtsfächer (FachID, Name)

Noten (NotenID, $\overline{\text{SchülerID}}$[Schüler], $\overline{\text{FachID}}$[Unterrichtsfächer], Note, Gewicht)

LehrerUnterrichtet ($\overline{\text{LehrerID}}$[Lehrer], KlassenID[Klassen], FachID[Unterrichtsfächer])



Lösungsvorschlag



Funktionalitäten

gehtIn n:1 Eine Klasse hat n Schüler, einE SchülerIn geht in eine Klasse

wirdBenotet 1:1:1: Das ist anderes nicht möglich, da nur `NotenID` Primärschlüssel ist, dann muss aber die Kombination aus `Schüler` und `Unterrichtsfach` auch einmalig sein, ðUNIQUE und es gilt `Note` und `Unterrichtsfach` bestimmt `Schüler`, `Note` und `Schüler` bestimmt `Unterrichtsfach` und `Schüler` und `Unterrichtsfach` bestimmt `Noten`.

LehrerUnterrichtet 1(Lehrer):n:n