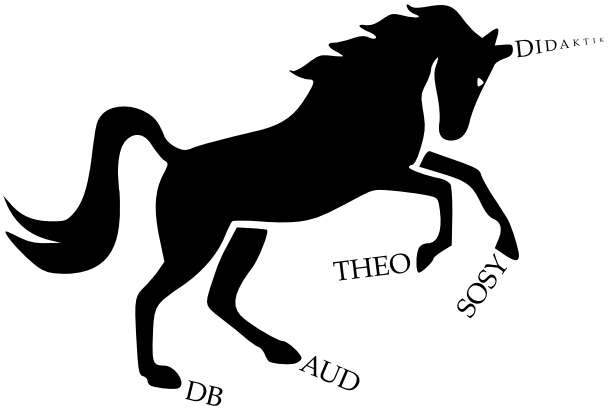
66113
Frühjahr
2003

Rechnerarchitektur / Datenbanken / Betriebssysteme (vertieft)

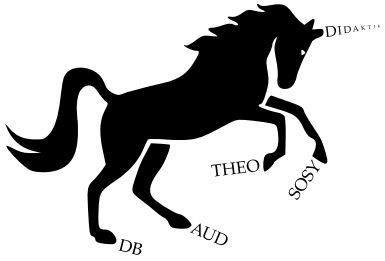
Aufgabenstellungen mit Lösungsvorschlägen



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

Aufgabe 5: SQL [Mitarbeiterverwaltung]

Gegeben seien die folgenden drei Relationen. Diese Relationen erfassen die Mitarbeiterverwaltung eines Unternehmens. Schlüssel sind fett dargestellt und Fremdschlüssel sind kursiv dargestellt. So werden Mitarbeiter, Abteilungen und Unternehmen jeweils durch ihre Nummer identifiziert. AbtNr ist die Nummer der Abteilung, in der ein Mitarbeiter arbeitet. Manager ist die Nummer des Mitarbeiters, der die Abteilung leitet. `UntNr` ist die Nummer des Unternehmens, dem eine Abteilung zugeordnet ist.

```
Mitarbeiter(Nummer, Name, Alter, Gehalt, AbtNr)
Abteilung(Nummer, Name, Budget, Manager, UntNr)
Unternehmen(Nummer, Name, Adresse)
```

```
CREATE TABLE unternehmen (
  Nummer integer NOT NULL PRIMARY KEY,
  Name VARCHAR(20) DEFAULT NULL,
  Adresse VARCHAR(50) DEFAULT NULL
);

CREATE TABLE abteilung (
  Nummer integer NOT NULL PRIMARY KEY,
  Name VARCHAR(20) DEFAULT NULL,
  Budget float DEFAULT NULL,
  Manager VARCHAR(20) NOT NULL,
  UntNr integer DEFAULT NULL REFERENCES unternehmen (Nummer)
);

CREATE TABLE mitarbeiter (
  Nummer integer NOT NULL PRIMARY KEY,
  Name VARCHAR(20) NOT NULL,
  Alter integer NOT NULL,
  Gehalt float NOT NULL,
  AbtNr integer NOT NULL REFERENCES abteilung (Nummer)
);

INSERT INTO unternehmen (Nummer, Name, Adresse) VALUES
  (1, 'Test.com', 'Alter Hafen 11'),
  (2, 'Party.de', 'Technostraße 3'),
  (3, 'IT.ch', 'Sequelweg 1');

INSERT INTO abteilung (Nummer, Name, Budget, Manager, UntNr) VALUES
  (1, 'Personal_Care', 20000, 'Huber', 1),
  (11, 'Tequilla_Mix', 50000, 'Taylor', 2),
  (21, 'Nerds', 500, 'Gates', 3);

INSERT INTO mitarbeiter (Nummer, Name, Alter, Gehalt, AbtNr) VALUES
  (1, 'Müller', 30, 30000, 1),
  (2, 'Huber', 45, 80000, 1),
  (3, 'Habermeier', 62, 40000, 1),
  (4, 'Leifsson', 27, 50000, 1),
  (5, 'Taylor', 37, 85000, 11),
```

```
(6, 'Smith', 61, 34000, 11),
(7, 'Pitt', 36, 40000, 11),
(8, 'Thompson', 54, 52000, 11),
(9, 'Gates', 69, 15000000, 21),
(10, 'Zuckerberg', 36, 10000000, 21),
(11, 'Jobs', 99, 14000000, 21),
(12, 'Nakamoto', 66, 5000000, 21);
```

(a) Wie hoch ist das Durchschnittsalter der Abteilung „Personal Care“ im Unternehmen „Test.com“?

Lösungsvorschlag

`GROUP BY` nicht nötig, `AS` nicht vergessen.

```
SELECT AVG(m.Alter) AS Durchschnittsalter
FROM Unternehmen u, Abteilung a, Mitarbeiter m
WHERE
  a.Name = 'Personal Care' AND
  u.Name = 'Test.com' AND
  u.Nummer = a.UntNr AND
  m.AbtNr = a.Nummer;
```

(b) Geben Sie für jedes Unternehmen das Durchschnittsalter der Mitarbeiter an!

Lösungsvorschlag

Statt `a.UntNr` kann `u.Nummer` verwendet werden. `a.UntNr` nur deshalb, weil man dann eventuell den Join über die Unternehmenstabelle sparen kann.

Alles was ausgegeben werden soll, muss auch in `GROUP BY` enthalten sein.

```
SELECT a.UntNr, u.Name, AVG(m.Alter) as Durchschnittsalter
FROM Unternehmen u, Abteilung a, Mitarbeiter m
WHERE
  u.Nummer = a.UntNr AND
  m.AbtNr = a.Nummer
GROUP BY a.UntNr, u.Name;
```

(c) Wie viele Mitarbeiter im Unternehmen „Test.com“ sind älter als ihr Chef? (D.h. sind älter als der Manager der Abteilung, in der sie arbeiten.)

Lösungsvorschlag

```
SELECT COUNT(*)
FROM Mitarbeiter m, Abteilung a, Unternehmen u
WHERE
  m.AbtNr = a.Nummer AND
  a.UntNr = u.Nummer AND
  u.Name = 'Test.com'
AND m.Alter > (
  SELECT ma.Alter
  FROM Mitarbeiter ma, Abteilung ab
  WHERE
    ma.Nummer = ab.Manager AND
    a.Nummer = ab.Nummer
);
```

oder einfacher:

```
SELECT COUNT(*)
FROM Mitarbeiter m, Abteilung a, Unternehmen u
WHERE
  m.AbtNr = a.Nummer AND
  a.UntNr = u.Nummer AND
  u.Name = 'Test.com'
AND m.Alter > (
  SELECT ma.Alter
  FROM Mitarbeiter ma
  WHERE ma.Nummer = a.Manager
);
```

Alternativ Lösung ohne Unterabfragem, mit Self join:

```
SELECT COUNT(*)
FROM Mitarbeiter m, Abteilung a, Unternehmen u, Mitarbeiter m2
WHERE
  m.AbtNr = a.Nummer AND
  a.UntNr = u.Nummer AND
  u.Name = 'Test.com' AND
  a.Manager = m2.Nummer AND
  m.Alter > m2.Atler;
```

(d) Welche Abteilungen haben ein geringeres Budget als die Summe der Gehälter der Mitarbeiter, die in der Abteilung arbeiten?

Lösungsvorschlag

```
SELECT a.Name, a.Nummer
FROM Abteilung a
WHERE a.Budget < (
  SELECT SUM(m.Gehalt)
  FROM Mitarbeiter m
  WHERE a.Nummer = m.AbtNr
);
```

Ohne Unterabfrage

```
SELECT a.Name, a.Nummer
FROM Abteilung a, Mitarbeiter m
WHERE a.Nummer = m.AbtNr
GROUP BY a.Nummer, a.Name, a.Budget
HAVING a.Budget < SUM(m.Gehalt);
```

(e) Versetzen Sie den Mitarbeiter „Wagner“ in die Abteilung „Personal Care“!

Lösungsvorschlag

```
UPDATE Mitarbeiter m
SET AbtNr = (
  SELECT a.Nummer FROM
  Abteilung a
  WHERE a.Name = 'Personal Care'
```

```
)
WHERE m.Name = 'Wagner';
```

(f) Löschen Sie die Abteilung „Personal Care“ mit allen ihren Mitarbeitern!

Lösungsvorschlag

```
DELETE FROM Mitarbeiter
WHERE AbtNr = (
  SELECT a.Nummer
  FROM Abteilung a
  WHERE a.Name = 'Personal Care'
);

DELETE FROM Abteilung
WHERE Name = 'Personal Care';
```

(g) Geben Sie den Managern aller Abteilungen, die ihr Budget nicht überziehen, eine 10 Prozent Gehaltserhöhung. (Das Budget ist überzogen, wenn die Gehälter der Mitarbeiter höher sind als das Budget der Abteilung.) Zusatzfrage: Was passiert mit Mitarbeitern, die Manager von mehreren Abteilungen sind?

Lösungsvorschlag

```
CREATE VIEW LowBudget AS (
  SELECT Nummer
  FROM Abteilung
  WHERE Nummer NOT IN (
    SELECT a.Nummer
    FROM Abteilung a
    WHERE
      a.Budget < (
        SELECT SUM(Gehalt)
        FROM Mitarbeiter m Abteilung A
        WHERE m.AbtNr = A.Nummer AND
        a.Nummer = A.Nummer
    )
  )
)

UPDATE Mitarbeiter
SET Gehalt = 1.1 * Gehalt
WHERE Nummer IN (
  SELECT Manager
  FROM LowBudget, Abteilung
  WHERE LowBudget.Manager = Abteilung.Nummer
)
```

Thema Nr. 2

Aufgabe 3: SQL [Universitätsverwaltung]

Gegeben sei folgendes relationales Schema, das eine Universitätsverwaltung modelliert:

```
Studenten {[MatrNr:integer, Name:string, Semester:integer]}
Vorlesungen {[VorlNr:integer, Titel:string, SWS:integer, gelesenVon:integer]}
Professoren {[PersNr:integer, Name:string, Rang:string, Raum:integer] }
hoeren {[MatrNr:integer, VorlNr:integer]}
voraussetzen {[VorgaengerVorlNr:integer, NachfolgerVorlNr:integer]}
pruefen {[MatrNr:integer, VorlNr:integer, PrueferPersNr:integer, Note:decimal]}
```

Formulieren Sie die folgenden Anfragen in SQL:

(a) Alle Studenten, die den Professor *Kant* aus einer Vorlesung kennen.

Lösungsvorschlag

```sql
SELECT DISTINCT s.Name
FROM Studenten s, Professoren p, hoeren h, Vorlesungen v
WHERE
  p.Name = 'Kant' AND
  v.gelesenVon = p.PersNr AND
  s.MatrNr = h.MatrNr AND
  h.VorlNr = h.VorlNr
```

(b) Geben Sie eine Liste der Professoren (Name, PersNr) mit ihrem Lehrdeputat (Summe der SWS der gelesenen Vorlesungen) aus. Ordnen Sie diese Liste so, dass sie absteigend nach Lehrdeputat sortiert ist! Bei gleicher Lehrtätigkeit dann noch aufsteigend nach dem Namen des Professors/der Professorin.

Lösungsvorschlag

```sql
SELECT p.Name, p.PersNr, SUM(v.SWS) AS Lehrdeputat
FROM Vorlesung v, Professoren p
WHERE v.gelesenVon = p.PersNr
GROUP BY p.Name, p.PersNr
ORDER BY Lehrdeputat DESC, p.Name ASC;
```

(c) Geben Sie eine Liste der Studenten (Name, MatrNr, Semester) aus, die mindestens zwei Vorlesungen bei *Kant* gehört haben.

Lösungsvorschlag

Mit einer VIEW

```sql
CREATE VIEW hoertKant AS
  SELECT s.Name, s.MatrNr, s.Semester, v.VorlNr
  FROM Studenten s, hoeren h, Vorlesungen v, Professoren p
  WHERE
    s.MatrNr = h.MatrNr AND
    h.VorlNr = v.VorlNr AND
    v.gelesenVon = p.PersNr AND
```

```
    p.Name = 'Kant';

SELECT DISTINCT h1.Name, h2.MatrNr, h1.Semester
FROM hoertKant h1, hoertKant h2
WHERE h1.MatrNr = h2.MatrNr AND h1.VorlNr <> h2.VorlNr;
```

oder:

```
SELECT DISTINCT Name, MatrNr, Semester
FROM hoertKant
GROUP BY Name, MatrNr, Semester
HAVING COUNT(VorlNr) > 1;
```

In einer Abfrage

```
SELECT s.Name, s.MatrNr, s.Semester
FROM Studenten s, hoeren h, Vorlesungen v, Professoren p
WHERE
  s.MatrNr = h.MatrNr AND
  h.VorlNr = v.VorlNr AND
  v.gelesenVon = p.PersNr AND
  p.Name = 'Kant'
GROUP BY s.MatrNr, s.Name, s.Semster
HAVING COUNT(s.MatrNr) > 1;
```

(d) Geben Sie eine Liste der Semesterbesten (MatrNr und Notendurchschnitt) aus.

Lösungsvorschlag

```
CREATE VIEW Notenschnitte AS (
  SELECT p.MatrNr, s.Name, s.Semester, AVG(Note) AS Durchschnitt
  FROM Studenten s, pruefen p
  WHERE s.MatrNr = p.MatrNr
  GROUP BY p.MatrNr, s.Name, s.Semester
);

SELECT a.Durchschnitt, a.MatrNr, a.Semester
FROM Notenschnitte a, Notenschnitte b
WHERE
  a.Durchschnitt >= b.Durchschnitt
  a.Semster = b.Semster
GROUP BY a.Durchschnitt, a.MatrNr, a.Semester
HAVING COUNT(*) < 2;
```