Lehrveranstaltungsmanagement - Die Datenbank

Die Organisation einer Lehrveranstaltung benoetigt eine Datenbank.

Beschreibung der Datenbank

Ein universitärer Arbeitsbereich plant ein System zur effizienten Verwaltung der LV Datenmodellierung. Es gibt Mitarbeiter und StudentInnen. Mitarbeiter werden eindeutig durch ihre Sozialversicherungsnummer (SVNR) identifiziert; StudentInnen eindeutig durch ihre Matrikelnummer (MNR). Zusätzlich wird bei beiden der Name (NAME) verwaltet. Bei Mitarbeitern ist auszerdem noch eine Mailadresse (EMAIL) und das Geburtsdatum (GEBDAT) relevant. Zu StudentInnen wird ferner die Studienkennzahl (SKZ) und optional das Exmatrikulationsdatum (EXMATRIKULATIONSDAT) angegeben. Von Mitarbeitern gibt es verschiedene Typen: AssistentInnen und TutorInnen. Für AssistentInnen ist die Art der Anstellung (ANSTELLUNG) relevant; für TutorInnen die Anzahl der Semester, die er/sie schon angestellt ist (ANZSEMESTER). Es gibt zwei Übungsblätter im Semester. Ein Übungsblatt wird eindeutig durch eine Nummer (NR) und das Semester (SEMESTER) identifiziert. Ferner ist bekannt, in welchem Zeitraum (VON, BIS) das Blatt zu bearbeiten ist. Die Abgabe eines Übungsblattes durch eine/n StudentIn wird eindeutig durch das Übungsblatt, sowie die/den StudentIn, dem die Abgabe zugeordnet ist, identifiziert. Zu jeder Abgabe werden der Zeitpunkt der Abgabe durch die/den StudentIn (ZEITPUNKT) und die Abgabedatei (DATEI) gespeichert. TutorInnen korrigieren die Abgaben und geben Feedback. Jede Abgabe wird durch genau einen/einer TutorIn korrigiert. Dabei wird für die korrigierte Abgabe die Antwortdatei (FEEDBACK) abgespeichert, sowie die Anzahl der erreichten Punkte (PUNKTE). Abgaben werden prinzipiell durch einen Plagiats-Checker und zusätzlich durch die TutorInnen auf Plagiatsverdacht kontrolliert. Abgaben können also Plagiate beliebig vieler anderer Abgaben sein. Für ein vermutetes Plagiat wird ein Kommentar (KOMMENTAR) abgespeichert. AssistentInnen prüfen StudentInnen nach der Abgabe der Übungsblätter in Form eines Prüfungsgespräches. Dabei prüft ein/e AssistentIn eine/n StudentIn zu einem Übungsblatt. Im Zuge des Prüfungsgespräches müssen die erreichten Punkte (PUNKTE) und das Datum (DATUM) gespeichert werden. In der LV gibt es zwei verschiedene Arten von schriftlichen Prüfungen: einen SQL-Test und eine Vorlesungsprüfung. Eine Prüfung wird eindeutig durch das Prüfungsdatum (DATUM) identifiziert. StudentInnen müssen bei einer Prüfung eine Mindestanzahl an Punkten erreichen (MINDESTPUNKTE). Ferner gibt es zu jeder Prüfung eine maximale Punkteanzahl (MAXIMALPUNKTE). Für den SQL-Test ist zusätzlich noch relevant, ob dieser der Nachtragstest (NACHTRAGSTEST) ist. Für die Vorlesungsprüfung ist zusätzlich das Stoffgebiet (STOFFGEBIET) relevant. Um die Abhaltung von Prüfungen zu ermöglichen, müssen auch Hörsäle verwaltet werden. Ein Hörsaal wird eindeutig durch das Gebäude (GEBAEUDE) und eine Nummer (NUMMER) identifiziert. Zusätzlich sind der Standort (STANDORT) und die Anzahl der Sitzplätze (SITZPLAETZE) bekannt. Mitarbeiter beaufsichtigen Prüfungen: es wird gespeichert, welcher Mitarbeiter in welchem Hörsaal bei welcher Prüfung Prüfungsaufsicht hat. StudentInnen treten zu Prüfungen an. Für jeden Antritt, sei es SQL-Test oder Vorlesungsprüfung, werden die erreichten Punkte (PUNKTE) gespeichert. Am Ende des Semesters (bzw. nach Prüfungen) werden Zeugnisse für StudentInnen ausgestellt. Ein Zeugnis ist dabei eindeutig durch den entsprechenden StudentInnen sowie das Prüfungsdatum (PRUEFUNGSDATUM) identifiziert. Ein/e StudentIn kann maximal drei Zeugnisse erhalten. Selbstverständlich muss zu jedem Zeugnis die Note (NOTE) gespeichert werden.

---

1)

Geben Sie die Sozialversicherunsnummern, Namen und Geburtsdaten aller Mitarbeiter aus, die vor dem

1.1.1991 geboren sind. Verwenden Sie hierzu die Funktion TO_DATE mit dem Datumsformat 'DD-MM-YYYY'. Sortieren Sie das Ergebnis aufsteigend nach Sozialversicherungsnummer.

```
SELECT SVNR, NAME, GEBDAT
FROM MITARBEITER
WHERE GEBDAT < TO_DATE('01-01-1991', 'DD-MM-YYYY')
ORDER BY SVNR;
```

2.1)

Geben Sie die Anzahl der ausgestellten Zeugnisse aus.

```
SELECT COUNT(*) AS ANZAHL
FROM ZEUGNIS;
```

2.2)

Geben Sie fuer jede Note die Anzahl der ausgestellten Zeugnisse aus.

```
SELECT NOTE, COUNT(*) AS ANZAHL
FROM ZEUGNIS
GROUP BY NOTE;
```

2.3)

Geben Sie fuer jene Noten die Anzahl der ausgestellten Zeugnisse aus, welche oefter als fuenfmal ausgestellt wurden. Sortieren Sie das Ergebnis absteigend nach der Anzahl der ausgestellten Zeugnisse.

```
SELECT NOTE, COUNT(*) AS ANZAHL
FROM ZEUGNIS
GROUP BY NOTE
HAVING COUNT(*) > 5
ORDER BY ANZAHL DESC;
```

3.1)

Geben Sie fuer jede/n AssistentIn (die/der Studierende prueft) die Sozialversicherungsnummer, die Anstellung

und die Anzahl der geprueften Studierenden aus. Sortieren Sie das Ergebnis aufsteigend nach Sozialversicherungsnummer.

```
SELECT a.SVNR, a.ANSTELLUNG, COUNT(*) AS ANZAHL
FROM ASSISTENT a
JOIN PRUEFT p ON a.SVNR = p.SVNR
GROUP BY a.SVNR, a.ANSTELLUNG
ORDER BY a.SVNR;
```

3.2)

Geben Sie fuer jede/n AssistentIn (die/der Studierende prueft) die Sozialversicherungsnummer, den Namen und die Anzahl der geprueften Studierenden aus. Sortieren Sie das Ergebnis aufsteigend nach Sozialversicherungsnummer.

```
SELECT a.SVNR, m.NAME, COUNT(*) AS ANZAHL
FROM ASSISTENT a
JOIN MITARBEITER m ON a.SVNR = m.SVNR
JOIN PRUEFT p ON a.SVNR = p.SVNR
GROUP BY a.SVNR, m.NAME
ORDER BY a.SVNR;
```

```
SELECT SVNR, NAME, COUNT(*) AS ANZAHL
FROM ASSISTENT
NATURAL JOIN MITARBEITER
NATURAL JOIN PRUEFT
GROUP BY SVNR, NAME
ORDER BY SVNR;
```

4.1)

Geben Sie den Namen und die Anzahl der Semester der dienstaeltesten TutorInnen aus.

```
SELECT m.NAME, t.ANZSEMESTER
FROM TUTOR t
JOIN MITARBEITER m ON t.SVNR = m.SVNR
WHERE t.ANZSEMESTER = (SELECT MAX(ANZSEMESTER)
                       FROM TUTOR);
```

```
SELECT NAME, ANZSEMESTER
FROM TUTOR
NATURAL JOIN MITARBEITER
WHERE ANZSEMESTER >= ALL(SELECT ANZSEMESTER
                         FROM TUTOR);
```

```
SELECT NAME, ANZSEMESTER
FROM TUTOR t1
NATURAL JOIN MITARBEITER
WHERE NOT EXISTS (SELECT 'x'
                  FROM TUTOR t2
                  WHERE t2.ANZSEMESTER > t1.ANZSEMESTER);
```

4.2)

Geben Sie den Namen und die Anzahl der Semester der dienstaeltesten TutorInnen aus, der nach dem 1.1.1991 geboren ist. Verwenden Sie hierfuer die Funktion TO_DATE mit dem Datumsformat 'DD-MM-YYYY'.

```
SELECT NAME, ANZSEMESTER
FROM TUTOR
NATURAL JOIN MITARBEITER
WHERE GEBDAT > TO_DATE('01-01-1991', 'DD-MM-YYYY') AND
    ANZSEMESTER >= ALL(SELECT ANZSEMESTER
                       FROM TUTOR
                       NATURAL JOIN MITARBEITER
                       WHERE GEBDAT > TO_DATE('01-01-1991', 'DD-MM-YYYY'));
```

```
SELECT m.NAME, t.ANZSEMESTER
FROM TUTOR t
JOIN MITARBEITER m ON t.SVNR = m.SVNR
WHERE m.GEBDAT > TO_DATE('01-01-1991', 'DD-MM-YYYY') AND
    t.ANZSEMESTER = (SELECT MAX(t1.ANZSEMESTER)
                     FROM TUTOR t1
                     JOIN MITARBEITER m1 ON t1.SVNR = m1.SVNR
                     WHERE m1.GEBDAT > TO_DATE('01-01-1991', 'DD-MM-YYYY'));
```

```
SELECT NAME, ANZSEMESTER
FROM TUTOR t1
NATURAL JOIN MITARBEITER
WHERE GEBDAT > TO_DATE('01-01-1991', 'DD-MM-YYYY') AND
    NOT EXISTS (SELECT 'x'
                FROM TUTOR t2
            NATURAL JOIN MITARBEITER
            WHERE GEBDAT > TO_DATE('01-01-1991', 'DD-MM-YYYY') AND
            t2.ANZSEMESTEr > t1.ANZSEMESTER);
```

5.1)

Geben Sie die Matrikelnummern und Namen jener Studierenden aus, welche kein Pruefungsgespraech absolviert haben, sowie keine Uebung abgegeben haben.

```
SELECT s.MNR, s.NAME
FROM STUDENT s
WHERE s.MNR NOT IN (SELECT MNR
                    FROM PRUEFT)
AND s.MNR NOT IN (SELECT MNR
                  FROM ABGABE);
```

```
SELECT s.MNR, s.NAME
FROM STUDENT s
WHERE NOT EXISTS (SELECT 'x'
                  FROM PRUEFT
              WHERE MNR = s.MNR) AND
NOT EXISTS (SELECT 'x'
            FROM ABGABE
          WHERE MNR = s.MNR);
```

5.2)

Geben Sie die Matrikelnummern und Namen jener Studierender aus, welche nie zu einer Vorlesungspruefung angetreten sind und bei denen mindestends eine Abgabe mit 0 Punkten bewertet wurde.

```
SELECT s.MNR, s.NAME
FROM STUDENT s
WHERE s.MNR NOT IN (SELECT t.MNR
                    FROM TRITT_AN t
                JOIN VORLESUNGSPRUEFUNG v ON t.DATUM = v.DATUM) AND
```

```
EXISTS (SELECT *
        FROM ABGABE
       WHERE PUNKTE = 0 AND MNR = s.MNR);
```

```
SELECT s.MNR, s.NAME
FROM STUDENT s
WHERE NOT EXISTS (SELECT 'x'
                  FROM TRITT_AN
                NATURAL JOIN VORLESUNGSPRUEFUNG
                WHERE MNR = s.MNR) AND
s.MNR IN (SELECT MNR
          FROM ABGABE
         WHERE PUNKTE = 0);
```

6.1)

Geben Sie Pruefungsdatum und Punktezahl aller Prueuefungen aus, die der/die StudentIn mit der Matrikelnummer 1200220 absolviert hat.

```
SELECT DATUM, PUNKTE
FROM TRITT_AN
WHERE MNR = '1200220';
```


```
SELECT DATUM, PUNKTE
FROM TRITT_AN
WHERE MNR = '1200220';
```

6.2)

Geben Sie Matrikelnummer, Name und durchschnittliche Punktezahl fuer alle Pruefungsantritte jedes Studierenden aus. Falls kein Pruefungsantritt vorliegt, so soll 0 anstelle der Punkte ausgegeben werden. Verwenden Sie hierfuer die Funktion COALESCE). Sortieren Sie das Ergebnis aufsteigend nach Matrikelnummer.

```
SELECT s.MNR, s.NAME, COALESCE(AVG(t.PUNKTE), 0) AS DURCHSCHNI
TT
FROM STUDENT s
LEFT OUTER JOIN TRITT_AN t ON s.MNR = t.MNR
```

```
GROUP BY s.MNR, s.NAME
ORDER BY s.MNR;
```

```
SELECT s.MNR, s.NAME, COALESCE(AVG(t.PUNKTE), 0) AS DURCHSCHNI
TT
FROM TRITT_AN t
RIGHT OUTER JOIN STUDENT s ON t.MNR = s.MNR
GROUP BY s.MNR, s.NAME
ORDER BY s.MNR;
```

6.3)

Geben Sie Matrikelnummer, Name ALLER Studierender zusammen mit der Summe der Punktezahl der Antritte fuer Pruefungen am 14.01.2013 und 25.01.2013 aus. Verwenden Sie hierfuer die Funktion TO_DATE mit dem Datumsformat 'DD-MM-YYYY'. Sortieren Sie das Ergebnis aufsteigend nach Matrikelnummer.

```
SELECT s.MNR, s.NAME, COALESCE(SUM(t.PUNKTE), 0) AS SUMME
FROM STUDENT s
LEFT OUTER JOIN TRITT_AN t ON s.MNR = t.MNR AND
                t.DATUM IN (TO_DATE('14-01-2013', 'DD-MM-YYYY'),
                  TO_DATE('25-01-2013', 'DD-MM-YYYY'))
GROUP BY s.MNR, s.NAME
ORDER BY s.MNR;
```

```
SELECT s.MNR, s.NAME, COALESCE(SUM(t.PUNKTE), 0) AS SUMME
FROM TRITT_AN t
RIGHT OUTER JOIN STUDENT s ON t.MNR = s.MNR AND
                (t.DATUM = TO_DATE('14-01-2013', 'DD-MM-YYYY') OR
          t.DATUM = TO_DATE('25-01-2013', 'DD-MM-YYYY'))
GROUP BY s.MNR, s.NAME
ORDER BY s.MNR;
```

---

7)

Geben Sie die Matrikelnummer, Namen und Studienkennzahl jener Studierender aus, welche zu allen Vorlesungspruefungen angetreten sind.

```
SELECT s.MNR, s.NAME, s.SKZ
FROM STUDENT s
WHERE NOT EXISTS (SELECT *
                  FROM VORLESUNGSPRUEFUNG v
                  WHERE v.DATUM NOT IN (SELECT t.DATUM
                                        FROM TRITT_AN t
                                        WHERE t.MNR = s.MNR));
```

```
SELECT MNR, NAME, SKZ
FROM STUDENT
NATURAL JOIN TRITT_AN
NATURAL JOIN VORLESUNGSPRUEFUNG
GROUP BY MNR
HAVING COUNT(DISTINCT DATUM) = (SELECT COUNT(*) FROM VORLESUN
GSPRUEFUNG));
```

---

8)

Geben Sie die Sozialversicherungsnummer und Namen aller Mitarbeiter aus. Vermerken Sie in einer zusaetzlichen Spalte um welchen Typ von Mitarbeiter es sich handelt: handelt es sich um eine/n AssistentIn, so geben Sie dessen Anstellung aus; handelt es sich um eine/n TutorIn, so geben Sie 'Tutor' aus. Sie koennen davon ausgehen, dass jeder Mitarbeiter genau einen dieser beiden Typen hat.

```
SELECT m.SVNR, m.NAME, a.ANSTELLUNG
FROM MITARBEITER m JOIN ASSISTENT a ON m.SVNR = a.SVNR
UNION
SELECT m.SVNR, m.NAME, 'Tutor'
FROM MITARBEITER m JOIN TUTOR t ON m.SVNR = t.SVNR;
```

```
SELECT SVNR, NAME, ANSTELLUNG
FROM MITARBEITER
NATURAL JOIN ASSISTENT
UNION
SELECT SVNR, NAME, 'Tutor'
FROM MITARBEITER
NATURAL JOIN TUTOR;
```

```
SELECT SVNR, NAME, ANSTELLUNG
FROM MITARBEITER
```

```
JOIN ASSISTENT USING(SVNR)
UNION
SELECT SVNR, NAME, 'Tutor'
FROM MITARBEITER
JOIN TUTOR USING(SVNR);
```

```
select m.svnr, m.name, coalesce(anstellung, 'Tutor')
from mitarbeiter m left outer join assistent a on m.svnr = a.svnr
```

---

9)

Geben Sie fuer jede Pruefung eine Statistik aus, bestehend aus dem Pruefungsdatum, der Anzahl der Teilnehmer, sowie der durchschnittlich erreichten Punkte. Sortieren Sie das Ergebnis aufsteigend nach Pruefungsdatum.

```
SELECT DATUM, COUNT(DISTINCT MNR) as teilnehmer, AVG(PUNKTE) as pu
nkte
FROM TRITT_AN t
GROUP BY DATUM
ORDER BY DATUM;
```

---

10)

Geben Sie jene Vorlesungspruefung aus (Datum, Punktedurchschnitt), bei der der hoechste Punktedurchschnitt erreicht wurde.

```
SELECT pr.DATUM, AVG(t.PUNKTE) AS PUNKTE
FROM VORLESUNGSPRUEFUNG pr JOIN TRITT_AN t ON t.DATUm = pr.DAT
UM
HAVING avg(t.PUNKTE) = (SELECT MAX(PUNKTE)
        FROM (SELECT pr.DATUM, AVG(t.PUNKTE) AS PUNKTE
            FROM VORLESUNGSPRUEFUNG pr JOIN TRITT_AN t ON t.
DATUM = pr.datum GROUP BY pr.DATUM))
GROUP BY pr.DATUM;
```

```
SELECT pr.DATUM, AVG(t.PUNKTE) AS PUNKTE
FROM VORLESUNGSPRUEFUNG pr JOIN TRITT_AN t ON t.DATUm = pr.DAT
UM
```

```
                HAVING avg(t.PUNKTE) >= ALL (SELECT AVG(t.PUNKTE)
                                             FROM VORLESUNGSPRUEFUNG pr JOIN TRITT_AN t ON t.D
ATUM = pr.datum GROUP BY pr.DATUM)
                 GROUP BY pr.DATUM;
```