ETH Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Prof. H.-J. Schek *A. Feßler, U. Röhm, H. Schuldt, R. Weber*

Informationssysteme IS-G SS 1997

Übung 3 Beispiellösung

Aufgabe 1: SQL in der Theorie

Professor (P_Name, Abt_Nr, Gebäude, Raum, Tel)
Abteilung(Abt_Nr, A_Name, Vorsteher)
Gebäude (Gebäude, Hausmeister, Adresse)
Student (ETH_Nr, S_Name, Semester, Abt_Nr)
Prüfung (ETH_Nr, Fach, Prüfer, Note)

Betrachten Sie nochmals obiges Schema der Universitätsdatenbank und formulieren Sie die folgenden Anfragen in SQL:

a) An welchen Hausmeister muss sich Professor Schek wenden, wenn er seinen Büroschlüssel vergessen hat?

```
SELECT Hausmeister
FROM   Gebäude, Professor
WHERE  P_Name = 'Schek'
AND    Professor.Gebäude = Gebäude.Gebäude
```

b) Welche Studenten (ETH_Nr) haben eine Prüfung beim augenblicklichen Vorsteher ihrer Abteilung abgelegt?

```
SELECT DISTINCT Student.ETH_Nr
FROM   Prüfung p, Abteilung a, Student s
WHERE  p.Prüfer = a.Vorsteher
AND    s.ETH_Nr = p.ETH_Nr
AND    s.Abt_Nr = a.Abt_Nr
```

c) Wo (Adresse, Raum) fand die Prüfung von Hugo Meier im Fach "Betriebssysteme" statt? (Annahme: Professoren prüfen in ihren Büros)

```
SELECT Adresse, Raum
FROM   Prüfung pr, Student s, Gebäude g, Professor p
WHERE  S_name = 'Hugo Meier'
AND    s.ETH_Nr = pr.ETH_Nr
AND    Prüfer = P_Name
AND    p.Gebäude= g.Gebäude
AND    Fach = 'Betriebssysteme'
```

d) Welche Studenten (ETH_Nr) mit mindestens 4 Semestern haben noch keine Prüfung abgelegt?

```
SELECT  ETH_Nr
FROM    Student S
WHERE   Semester ≥ 4
AND     NOT EXISTS  ( SELECT *
                      FROM Prüfung p
                      WHERE s.ETH_Nr = p.ETH_Nr)
```

Alternativ mit NOT IN:

```
SELECT  ETH_Nr
FROM    Student
WHERE   Semester ≥ 4
AND     ETH_Nr NOT IN (SELECT DISTINCT ETH_Nr
                       FROM Prüfung)
```

Alternativ mit MINUS:

```
SELECT  ETH_Nr
FROM    Student
WHERE   Semester ≥ 4
MINUS
SELECT  DISTINCT ETH_Nr
FROM    Prüfung
```

e) Welche Studenten (ETH_Nr) haben ausschliesslich Prüfungen bei Professoren ihrer Abteilung abgelegt?

```
SELECT DISTINCT ETH_Nr
FROM Prüfung pf
WHERE NOT EXISTS  ( SELECT *
                    FROM Student s, Prüfung pr, Professor p
                    WHERE pf.ETH_Nr = s.ETH_Nr
                    AND s.ETH_Nr =pr.ETH_Nr
                    AND pr.Prüfer = p.P_Name
                    AND s.Abt_Nr ≠ p.Abt_Nr )
```

Eine einfachere Lösung:

```
SELECT  DISTINCT ETH_Nr
FROM    Student s
WHERE   Abt_Nr = ALL ( SELECT Abt_Nr
                       FROM Professor, Prüfung p
                       WHERE Prüfer = P_Name
                       AND s.ETH_Nr = p.ETH_Nr )
```

f) Welche Studenten (ETH_Nr) haben alle ihre bisher abgelegten Prüfungen mit der Note 6 bestanden?

```
SELECT DISTINCT ETH_Nr
FROM   Prüfung po
WHERE  NOT EXISTS  ( SELECT *
                     FROM Prüfung pi
                     WHERE po.ETH_Nr= pi.ETH_Nr
                     AND   Note < 6)
```

Einfacher wird die Lösung mit der Benutzung von MINUS:

```
SELECT  DISTINCT ETH_Nr
FROM    Prüfung
MINUS
SELECT  DISTINCT ETH_Nr
FROM    Prüfung
WHERE   Note < 6
```

Die erste Teilanfrage ergibt alle, die eine Prüfung abgelegt haben, die Zweite alle, die mindestens eine Prüfung schlechter als 6 abgelegt haben. Die Differenz ergibt die gesuchten 'Spitzenkräfte'.

Eine noch elegantere Lösung wäre:

```
SELECT ETH_Nr
FROM   Prüfung
GROUP  BY ETH_Nr
HAVING min(Note) = 6
```

g) Bestimmen Sie die Durchschnittsnote für jeden Studenten.

```
SELECT ETH_Nr, AVG(Note)
FROM   Prüfung
GROUP  BY ETH_Nr
```

h) Welche Studenten (ETH_Nr) haben ein Prüfungsfach (Fach) besser abgeschlossen als der Durchschnitt in diesem Fach?

```
SELECT  DISTINCT ETH_Nr
FROM    Prüfung p
WHERE   Note > ( SELECT AVG(Note)
                 FROM Prüfung ps
                 WHERE p.Fach = ps.Fach )
```

Alternativ geht hier auch eine Lösung in zwei Schritten mit einem View.

```
CREATE VIEW AVG_Fach AS
SELECT Fach, AVG(Note) AVG_Note
FROM   Prüfung
GROUP  BY Fach

SELECT DISTINCT ETH_Nr
FROM   AVG_Fach a, Prüfung p
WHERE  Note > AVG_Note
AND    a.Fach = p.Fach
```

Eine vermeintlich einfache Lösung funktioniert leider nicht:

```
SELECT    DISTINCT ETH_Nr
FROM      Prüfung
GROUP     BY  Fach
HAVING    Note > AVG(Note)
```

Hier kommen mit ETH_NR im SELECT und Note im HAVING zwei Nichtgruppeneigenschaften vor. Dies ist in Queries mit GROUP BY nicht erlaubt.

i) Welches Prüfungsfach (Fach) wurde mit der besten Durchschnittsnote abgeschlossen?

```
SELECT   Fach
FROM     Prüfung
GROUP BY Fach
HAVING   AVG(Note) ≥ All
         ( SELECT AVG(Note) FROM Prüfung GROUP BY Fach )
```

Eine alternative Formulierung mit Hilfe einer View sieht so aus:

```
CREATE VIEW AVG_Fach AS
SELECT Fach, AVG(Note) AVG_Note
FROM   Prüfung
GROUP BY Fach

SELECT Fach
FROM   AVG_Fach
WHERE  AVG_Note = ( SELECT MAX ( AVG_Note ) FROM AVG_Fach );
```

Ein anderer Ansatz, der ohne View auskommt:

```
SELECT Fach, AVG(Note)
FROM Prüfung
GROUP BY Fach
Order by 2 DESC
```

Bei dieser Formulierung werden die Prüfungen nach Fächern gruppiert und nach Durchschnittsnote absteigend sortiert ausgegeben. Das erste Resultattupel ist also die gewünschte Antwort. Insofern ist diese Antwort keine "exakte" Lösung, aber unter pragmatischen Gesichtspunkten trotzdem brauchbar.

j) Welcher Student (Name) hat alle Prüfungen als Bester abgeschlossen?

```
SELECT  DISTINCT S_Name
FROM    Student s
WHERE   NOT EXISTS
        (SELECT *
         FROM   Prüfung p1
         WHERE  s.ETH_Nr = p1.ETH_Nr
         AND    EXISTS
                (SELECT *
                 FROM   Prüfung p2
                 WHERE  p1.Fach = p2.Fach
                 AND    p1.Note < p2.Note))
```

Formulieren Sie folgende Datenerfassungen und -änderungen in SQL:

k) Der Student mit der ETH_Nr. 91-999-999 hat die Prüfung im Fach Informationssysteme bei Prof. Weikum mit der Traumnote 6.0 absolviert.

```
INSERT INTO Prüfung VALUES
       ('91-999-999', 'Informationssysteme','Weikum', 6.0);
```

l) Der Student mit der ETH_Nr. 85-999-999, der die ETH schon lange verlassen hat, besteht darauf, dass sämtliche Informationen über ihn gelöscht werden.

```
DELETE FROM Student WHERE ETH_Nr='85-999-999';
DELETE FROM Prüfung WHERE ETH_Nr='85-999-999';
```

m) Alle Prüfungsnoten im Fach Informationssysteme werden um 0.5 Notenpunkte angehoben.

```
UPDATE Prüfung SET note=note+0.5 WHERE Fach='Informationssysteme' AND note
≤ 5.5;
```

Aufgabe 2: SQL in der Praxis: Das Datenbanksystem ORACLE

Diese praktische Aufgabe ist diesmal per WWW zu lösen, und zwar mittels Html-Forms unter *http://www-dbs.inf.ethz.ch/is-cgi/pu.tcl*. Sie gelangen dorthin auch von unserer IS-G-Homepage über den Link *Übungen und Terminplan*. Beim erstmaligen Zugriff werden Sie nach Ihrem Namen, Ihrer ETH-Nummer und Ihrem Übungsleiter gefragt. Bitte füllen Sie dies korrekt aus, damit wir Ihre Übung für das Testat werten können, indem alle Ihre SQL-Anfragen registriert werden. Hierfür ist es unerlässlich, dass Ihr Browser Cookies ohne Nachfragen akzeptiert (in *Netscape* unter *Options – Network Preferences – Protocols* einstellen) oder Sie andernfalls beim Nachfragen des Browsers das Cookie akzeptieren. Damit können Sie sich ein nochmaliges Anmelden sparen, wenn Sie zu einem späteren Zeitpunkt an der Übung weiterarbeiten wollen.

a) Kennenlernen von ORACLE und der Musikdatenbank

Folgende Tabellen sind vorhanden:

```
select * from all_catalog where owner='IS0';
```

TABLE_NAME	*TABLE_TYPE*
DISK	*TABLE*
FK_CONSTRAINTS	*VIEW*
INTEG_CONSTRAINTS	*VIEW*
MUSIKSTUECK	*TABLE*
MUSIKSTUECK_AUTOR	*TABLE*
MUSIKSTUECK_INTERPRET	*TABLE*
PERSON	*TABLE*

Bei den Tabellen kann mit `describe` *nur Informationen über die Zulassung von Nullwerten erhalten werden. Die anderen definierten Integritätsbedingungen sind nicht so einfach zu erhalten. Die definierte Sicht* `Fk_Constraints` *gibt Informationen über die Fremdschlüssel, während die Sicht* `Integ_Constraints` *Informationen über weitere Integritätsbedingungen liefert.*

Zum Beispiel:

```
select * from fk_constraints
where owner='IS0' and table_name = 'MUSIKSTUECK_INTERPRET';
```

Achtung: Letzteres dauert lange in der Ausführung!

b) SQL-Anfragen stellen

i) Welche Musikstücke (DiskID, StückID) hat Paul McCartney arrangiert (Tätigkeit = Texter).

```
SELECT DiskID, StueckID
FROM Musikstueck_Autor A, Person P
WHERE P.PID = A.PID AND Name = 'McCartney, Paul'
        AND Tätigkeit = 'Texter';
```

ii) Von welchen Barry Gibb-Stücken existieren Aufnahmen mit Barbra Streisand als Solistin?

```
SELECT DISTINCT Titel
FROM Musikstueck M, Musikstueck_Autor A,
      Musikstueck_Interpret I, Person PA, Person PI
WHERE PA.PID = A.PID AND PA.Name = 'Gibb, Barry' AND
        PI.PID = I.PID AND PI.Name = 'Streisand, Barbra' AND
        A.DiskID = I.DiskID AND A.StueckID = I.StueckID AND
        I.Funktion = 'Solist' AND
        A.DiskID = M.DiskID AND A.StueckID = M.StueckID;
```

iii) Welche seiner eigenen Lieder singt Bob Dylan, d.h. er ist Autor und Interpret zugleich?

```
SELECT A.DiskID, A.StueckID
FROM Musikstueck_Interpret I, Musikstueck_Autor A, Person P
WHERE P.PID = A.PID AND P.PID = I.PID AND
        A.DiskID = I.DiskID AND A.StueckID = I.StueckID AND
        I.Instrument = 'Stimme' AND Name = 'Dylan, Bob';
```

iv) Welche Disks (DiskTitel) enthalten Stücke, an denen Paul McCartney beteiligt ist?

```
SELECT DISTINCT DiskTitel FROM Disk, Person P
WHERE Name = 'McCartney, Paul'
AND DiskID IN (
      SELECT DiskID FROM Musikstueck_Interpret M
      WHERE (P.PID = M.PID)
      UNION
      SELECT DiskID FROM Musikstueck_Autor M
      WHERE (P.PID = M.PID));
```

v) In welchen Aufnahmen hat Ernst Mosch sowohl als Solist als auch als Dirigent mitgewirkt?

```
SELECT DISTINCT D.DiskID, D.StückID
FROM Musikstueck_Interpret D, Musikstueck_Interpret S, Person P
WHERE P.PID = D.PID AND P.PID = S.PID AND
        D.DiskID = S.DiskID AND D.StueckID = S.StueckID AND
        S.Funktion = 'Solist' AND D.Funktion = 'Dirigent' AND
        Name = 'Mosch, Ernst';
```

c) Kompliziertere SQL-Anfragen

i) Welche Disks (DiskID) enthalten mehr als 18 Musikstücke und wieviele sind es?

```
SELECT DiskID, count(*)
FROM Musikstueck
GROUP BY DiskID
HAVING count(StueckID) > 18;
```

ii) Welche Musiker (Namen) haben in mehr als 30 Musikstücken mitgespielt (Einträge in Musikstück_Interpret) ?

```
SELECT Name FROM Person
WHERE PID IN (SELECT PID
                FROM (SELECT distinct PID, DiskID, StueckID
                        FROM Musikstueck_Interpret)
                GROUP BY PID
                HAVING count(*) > 30);
```

Achtung: das Subselect im FROM-Teil muss unter Umständen durch eine View ersetzt werden.

iii) Welche Musiker spielen mindestens drei Instrumente?

```
SELECT Name FROM Person
WHERE PID IN (SELECT PID
                FROM Musikstueck_Interpret
                GROUP BY PID
                HAVING count(distinct instrument) ≥ 3);
```

iv) Welche Musikstücke (Titel) werden ohne Instrumentalbegleitung („a capella“) gesungen?

```
SELECT Titel FROM Musikstueck
WHERE (DiskID, StueckID) IN (SELECT DiskID, StueckID
                               FROM Musikstueck_Interpret
                               WHERE Instrument = 'Stimme'
                               MINUS SELECT DiskID, StueckID
                               FROM Musikstueck_Interpret
                               WHERE Instrument <> 'Stimme')
```

Einige ältere SQL-Implementationen verfügen nicht über den MINUS-Operator. In solchen Systemen kann die Teilaufgabe wie folgt gelöst werden:

```
SELECT Titel FROM Musikstueck M
WHERE EXISTS (SELECT * FROM Musikstueck_Interpret I
                WHERE Instrument = 'Stimme' AND
                M.DiskID = I.DiskID AND M.StueckID = I.StueckID)
AND NOT EXISTS (SELECT * FROM Musikstueck_Interpret I
                  WHERE Instrument <> 'Stimme' AND
                  M.DiskID = I.DiskID AND M.StueckID = I.StueckID);
```