Datenbanksysteme 2011

noch Kapitel 7:
SQL
Vorlesung vom 23.05.2011

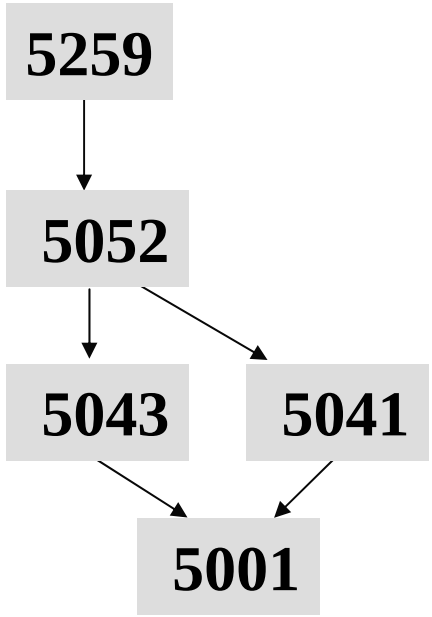
Oliver Vornberger

Institut für Informatik
Universität Osnabrück

Transitive Hülle

35.) Liste alle Voraussetzungen für
die Vorlesung "Der Wiener Kreis" (vorlnr 5259)

vorgaenger	nachfolger
5001	5041
5001	5043
5001	5049
5041	5052
5043	5052
5041	5216
5052	5259



⇒ Transitive Hülle einer rekursiven Relation

Prolog: Transitive Hülle

```
Trans(V,N) :- voraussetzen(V,N).
Trans(V,N) :- Trans(V,Z), voraussetzen(Z,N)
```

Oracle: Transitive Hülle

```
select Titel
from   Vorlesungen
where  VorlNr in (
         select Vorgaenger
         from voraussetzen
         connect by Nachfolger = prior Vorgaenger
         start with Nachfolger = (
                  select VorlNr
                  from Vorlesungen
                  where Titel = 'Der Wiener Kreis'
         )
)
```

MySQL: Transitive Hülle

```
create temporary table nach (nr integer)
create temporary table vor (nr integer)

insert into nach
      select vorlnr from Vorlesungen
      where titel = 'Der Wiener Kreis'
```

mehrfach iterieren:

```
insert into vor
    select v.vorgaenger
    from voraussetzen v
    where v.nachfolger in (select * from nach)

delete from nach
insert into nach
    select distinct nr from vor
```

SQL: insert

1.) Füge neue Vorlesung mit einigen Angaben ein:

```
insert into Vorlesungen (VorlNr, Titel, gelesenVon)
values (4711, 'Selber Atmen', 2125)
```

2.) Schicke alle Studenten in die Vorlesung *Selber Atmen*:

```
insert  into hoeren
select  MatrNr, VorlNr
from    Studenten, Vorlesungen
where   Titel = 'Selber Atmen'
```

SQL: update

3.) Erweitere die Vorlesung um ihre Semesterwochenstundenzahl:

```
update  Vorlesungen
set     SWS=6
where   Titel='Selber Atmen'
```

Erhöhe die Semesterzahl bei allen Studenten

```
update Studenten
set semester = semester + 1
```

Trage bei allen Professoren die Zahl ihrer Hörer ein:

```
update Professoren p set anzhoerer =
(select count(*)
        from Vorlesungen v, hoeren h
        where p.PersNr = v.gelesenVon
        and h.VorlNr = v.VorlNr)
```

SQL:delete

4.) Entferne alle Studenten aus der Vorlesung *Selber Atmen*:

```
delete from hoeren
where vorlnr=
      (select VorlNr from Vorlesungen
       where Titel = 'Selber Atmen')
```

5.) Entferne die Vorlesung *Selber Atmen*:

```
delete from Vorlesungen
where titel = 'Selber Atmen'
```

SQL: Sichten

1.) Lege Sicht an für Prüfungen ohne Note:

```
create view pruefenSicht as
select MatrNr, VorlNr, PersNr
from   pruefen
```

2.) Lege Sicht an für Studenten und ihre Professoren:

```
create  view StudProf
(Sname, Semester, Titel, Pname) as
select  s.Name, s.Semester, v.Titel, p.Name
from    Studenten s, hoeren h, Vorlesungen v, Professoren p
where   s.MatrNr       = h.MatrNr
and     h.VorlNr       = v.VorlNr
and     v.gelesenVon   = p.PersNr
```

SQL: Sichten

3.) Lege Sicht an für Professoren und Durchschnittsnote:

```
create view ProfNote (Persnr, Durchschnittsnote) as
select PersNr, avg(Note)
from pruefen
group by persnr
```

4.) Entferne Sichten wieder

```
drop view PruefenSicht;
drop view StudProf;
drop view ProfNote;
```

SQL: Generalisierung durch Verbund

5.) Lege Untertyp als Verbund von Obertyp und Erweiterung an:

```
create table Angestellte   (PersNr      integer not null,
                            Name       varchar(30) not null)
create table ProfDaten     (PersNr      integer not null,
                            Rang       character(2),
                            Raum       integer)
create table AssiDaten     (PersNr      integer not null,
                            Fachgebiet varchar(30),
                            Boss       integer)
create view Profs as
   select a.persnr, a.name, d.rang, d.raum
   from   Angestellte a, ProfDaten d
   where  a.PersNr = d.PersNr
create view Assis as
   select a.persnr, a.name, d.fachgebiet, d.boss
   from   Angestellte a, AssiDaten d
   where  a.PersNr = d.PersNr
```

SQL: Tabellen und Sichten entfernen

Entferne die Tabellen und Sichten wieder:

```
drop  table Angestellte
drop  table AssiDaten
drop  table ProfDaten
drop  view  Profs
drop  view  Assis
```

SQL: Generalisierung durch Vereinigung

6.) Lege Obertyp als Vereinigung von Untertypen an (zwei der drei Untertypen sich schon vorhanden):

```
create table AndereAngestellte (PersNr integer not null,
                                Name varchar(30) not null)

create view Angestellte as
   (select PersNr, Name from Professoren) union
   (select PersNr, Name from Assistenten) union
   (select PersNr, Name from AndereAngestellte)
```

Entferne die Tabelle und die Sichten wieder:

```
drop table andereAngestellte
drop view Angestellte
```

Index

```
create index titelindex
on Vorlesungen(titel asc)

create index semesterindex
on Studenten(semester asc)

drop index titelindex on Vorlesungen
drop index semesterindex on Studenten
```

Bulkinsert

4711;Willi;C4;339;1951-03-24
4712;Erika;C3;222;1962-09-18

```
LOAD DATA INFILE '/tmp/prof.txt'
INTO TABLE Professoren
FIELDS TERMINATED BY ';'
LINES TERMINATED BY '\n'
```

Tabellen konto, gebucht, abgelehnt

```
create table konto (nr int, stand int)

insert into konto values (1,100)
insert into konto values (2,100)
insert into konto values (3,100)

create table gebucht (
  datum date, nr_1 int, nr_2 int, betrag int)

create table abgelehnt (
  datum date, nr_1 int, nr_2 int, betrag int)
```

Stored Procedure ueberweisung

```
drop procedure if exists ueberweisung
create procedure ueberweisung (x int, y int, betrag int)
begin
  set @s = (select stand from konto where nr = x);
  if (@s < betrag)
  then
    insert into abgelehnt values (now(), x, y, betrag);
  else
    update konto set stand = stand-betrag where nr = x;
    update konto set stand = stand+betrag where nr = y;
    insert into gebucht values (now(), x, y, betrag);
  end if;
end
```

```
call ueberweisung(2,3,50)
```

Stored Function f2c

```
drop function if exists f2c

create function f2c (fahrenheit int)
returns int
deterministic
begin
  set @celsius=(fahrenheit-32)/9.0 * 5.0;
  return @celsius;
end
```

```
select celsius, f2c(celsius) from klimatabelle
```

Stored Function ggt

```
drop function if exists ggt

create function ggt(a int, b int)
returns int
deterministic
begin
  while a != b DO
    if (a>b)
      then set a = a-b;
      else set b = b-a;
    end if;
  end while;
  return a;
end
```

```
select ggt(315,60)
```

Stored Procedure berechneVorlesungen

```
drop procedure if exists berechneVorlesungen
create procedure berechneVorlesungen() -- speichere pro Professor
begin                                  -- die Zahl seiner Vorlesungen
  declare done int default 0;
  declare prof_name CHAR(16);
  declare prof_cursor cursor for
    select p.name
    from Professoren p, Vorlesungen v
    where p.persnr = v.gelesenvon;
  declare continue handler for sqlstate '02000' set done = 1;
  update Professoren set anzVorlesungen = 0;
  open prof_cursor;
  repeat
    fetch prof_cursor into prof_name;
    if not done then
      update Professoren set anzVorlesungen = anzVorlesungen + 1
      where name = prof_name;
    end if;
  until done end repeat;
  close prof_cursor;
end
```

```
call berechneVorlesungen()
```

Stored Procedure zuwenighoerer

```
DROP PROCEDURE if exists zuwenigHoerer

CREATE PROCEDURE zuwenigHoerer() -- setze Professor auf Rang C2
begin                            -- falls weniger als 2 Hoerer
  declare done int default 0;
  declare nr, anzahl int;
  declare prof_cursor cursor for
    select gelesenvon, count(*) from Vorlesungen v, hoeren h
    where v.vorlnr = h.vorlnr
    group by gelesenvon;
  declare continue handler for sqlstate '02000' set done = 1;
  open prof_cursor;
  fetch prof_cursor into nr, anzahl;
  while not done do
    if (anzahl <2) then
      update Professoren set rang = 'C2' where persnr = nr;
    end if;
    fetch prof_cursor into nr, anzahl;
  end while;
  close prof_cursor;
end
```

Stored Procedure für Transitive Hülle

```
create procedure transitivehuelle (start int)
begin
  create temporary table nach (nr integer);
  create temporary table vor (nr integer);
  insert into nach values (start);
  set @alt = 0;
  set @neu = 0;
  repeat
    insert into vor
    select distinct v.vorgaenger
      from voraussetzen v
      where v.nachfolger in (select * from nach) ;
    delete from nach;
    insert into nach select distinct nr from vor;
    set @alt = @neu;
    set @neu = (select count(distinct nr) from vor);
   UNTIL (@neu = @alt) END REPEAT;
   select distinct * from vor;
end
```

```
call transitivehuelle(5259);
```

Workbench