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

Institut für Informationssysteme, Dr. Markus Tresch
Uwe Röhm

Verteilte Objektverwaltung: Client, Server & Middleware
Lösungsvorschlag Übung 6

Aufgabe 1: Ausdrücke höherer Ordnung vs. SQL

Voraussetzungen

In den folgenden SQL Anfragen gehen wir davon aus, daß die drei Metadaten-Relationen gegeben sind:

sys_databases (dbID, name, ...)

sys_tables (tableID, name, dbID, ...)

sys_attributes (attrID, name, tableID, ...)

Wir verwenden die Syntax *relation*@*database*, um auszudrücken, daß wir eine Relation *relation* in der Datenbank *database* ansprechen.

In mehreren Anfragen ergibt sich das Problem, daß man in SQL die Metadaten auslesen und mit den erhaltenen Werten auf entsprechende Relationen und Attribute zugreifen müßte. Das ist in SQL (und der zugrundeliegenden relationalen Algebra) nicht möglich. Für diese Anfragen verwenden wir daher eine erfundene Syntax, um die Indirektion über die ausgelesenen Metadaten auszudrücken. Diese Syntax ist an die Verwendung von Unix Shell-Variablen angelehnt: ${*var*} soll dem Wert entsprechen, an den *var* gebunden ist („Dereferenzierung“).

Einfache Anfragen (siehe Folie 21)

Ausdrücke höherer Ordnung:

```
?.reuter.r(.stkCode=hp, .clsPrice>200)
```

SQL:

```
select *
from   r@reuter
where stkCode = 'hp'
and    clsPrice > 200
```

Ausdrücke höherer Ordnung:

```
?.reuter.r(.stkCode=hp, .clsPrice>60, .date=D),
 .reuter.r(.stkCode=ibm, .clsPrice>150, .date=D)
```

SQL:

```
select r1.date
from   r@reuter r1, r@reuter r2
where r1.stkCode = 'hp'
and    r1.clsPrice > 60
and    r1.date = r2.date
and    r2.stkCode = 'ibm'
and    r2.clsPrice > 150
```

Ausdrücke höherer Ordnung:

```
?.reuter.r(.stkCode=hp, .clsPrice=P, .date=D),
 .reuter.r¬(.stkCode=hp, .clsPrice>P)
```

SQL:

```
select r.date, r.clsPrice
from   r@reuter
where r.stkCode = 'hp'
and    r.clsPrice = ( select max(clsPrice)
                      from   r@reuter
                      where stkCode = 'hp' )
```

Metadaten Anfragen (siehe Folie 22)

?.X

```
select   d.name
from     sys_databases d
```

?.reuter.X

```
select   t.name
from     sys_databases d, sys_tables t
and      d.name = 'reuter'
where    d.dbID = t.dbID
```

?.X.Y

```
select   d.name, t.name
from     sys_databases d, sys_tables t
where    d.dbID = t.dbID
```

?.X.hp

```
select   d.name
from     sys_databases d, sys_tables t
where    d.dbID = t.dbID
and      t.name = 'hp'
```

?.X.Y(.date)

```
select   d.name, t.name
from     sys_databases d, sys_tables t, sys_attributes a
where    d.dbID = t.dbID
and      t.tableID = a.tableID
and      a.name = 'date'
```

?.reuter.X, .schwab.X, .source.X

```
select   t.name
from     sys_databases d, sys_tables t
where    d.dbID = t.dbID
grouped by t.dbID, t.name
having   count(t.dbID) = (select = count(*)
                          from  sys_databases db)
```

Anfragen höherer Ordnung (siehe Folie 23)

?.reuter.r(.stkCode = S, .clsPrice > 200)

```
select  r.stkCode
from    r@reuter
where   r.clsPrice > 200
```

Die nächsten beiden SQL-Anfragen gehen normalerweise nicht...

?.schwab.r(.S > 200)

```
select  a.name
from    sys_databases d, sys_tables t, sys_attributes a, r@schwab
where   d.dbID = t.dbID
and     t.tableID = a.tableID
and     d.name = 'schwab'
and     t.name = 'r'
and     r.${a.name} > 200
```

?.source.S(.clsPrice > 200)

```
select  t.name
from    sys_databases d, sys_tables t
where   d.dbID = t.dbID
and     d.name = 'source'
and     exists ( select  *
                 from    ${t.name}@source
                 where   clsPrice > 200 )
```

```
?.reuter.r(.date=D, .clsPrice=P, .stkCode=S),
 .reuter.r¬(.clsPrice>P, .date=D)
```

```
select r.stkCode, r.date, r.clsPrice
from   r@reuter
where r.clsPrice = ( select  max(r2.clsPrice)
                     from    r@reuter r2
                     where r2.date = r.date )
```

Erneut zwei eigentlich nicht formulierbare SQL-Anfragen...

```
?.schwab.r(.S=P, .date=D),
 .schwab.r¬(.S'=P, .date=D)
```

```
select a.name, r.date, r.${a.name}
from   sys_databases d, sys_tables t,
       sys_attributes, r@schwab
where d.dbID = t.dbID
and    t.tableID = a.tableID
and    d.name = 'schwab'
and    t.name = 'r'
and    r.${a.name} = ( select  max(r.${a.name})
                       from    sys_databases d2, sys_tables t2,
                               sys_attributes a2, r@schwab
                       where d2.dbID = t2.dbID
                       and    t2.tableID = a2.tableID
                       and    d2.name = 'schwab'
                       and    t2.name = 'r'
                       and    r2.date = r.date )
```

```
?.source.S(.clsPrice=P, .date=D),
 .source.S'¬(.clsPrice>P, .date=D)
```

```
select t.name, ${t.name}.date ${t.name}.clsPrice
from   sys_databases d, sys_tables t, ${t.name}@source
where d.dbID = t.dbID
and    d.name = 'source'
and    ${t.name}.clsPrice = ( select  max(${t2.name}.clsPrice)
                              from    sys_databases d2,
                                      sys_tables t2,
                                      ${t2.name}@source
                              where d2.dbID = t2.dbID
                              and    d2.name = 'source'
                              and    ${t2.name}.date =
                                      ${t.name}.date )
```

Update Operationen (siehe Folie 25)

```
?.reuter.r+(.date=3.3.85, .stkCode=hp, .clsPrice=50)
```

```
insert into r@reuter
values (3.3.85, 'hp', 50);
```

```
?.reuter.r-(.data=3.3.85, .stkCode=hp)
```

```
delete r@reuter
where r.date = 3.3.85
and    r.stkCode = 'hp'
```

```
?.schwab.r-(.date=3.3.85, .hp=C),
 .schwab.r+(.date=3.3.85, .hp=C+10)
```

```
update r@schwab
set    r.hp = r.hp + 10
where r.date = 3.3.85
```

Die folgende letzte SQL-Anfrage zu dieser Folie ist wieder kein korrektes SQL...

?.source-.R(.clsPrice > 200)

```
drop   table ${t.name}
where  exists ( select t.*
                from   sys_databases d, sys_tables t,
                       ${t.name}
                where  d.dbID = t.dbID
                and    d.name = 'source'
                and    ${t.name}.clsPrice > 200 )
```

Sichtendefinitionen (siehe Folie 26)

Die Sichtdefinition über alle drei Datenbanken ist nicht ohne Rückgriff auf die Metadaten formulierbar, also in reinem SQL nicht enthalten:

```
create   view p@dbI ( stkCd, dat, clsPr, db) as
       ( select r.stkCode, r.date, r.clsPrice, 'reuter'
         from   r@reuter )
         union
       ( select a.name, r.date, r.${a.name}, d.name
         from   sys_databases d, sys_tables t, sys_attributes a, r@schwab
         where  d.dbID = t.dbID
         where  t.tableID = a.tableID
         and    d.name = 'schwab'
         and    t.name = 'r' )
         union
       ( select t.name, ${t.name}.date, ${t.name}.clsPrice, d.name
         from   sys_databases d, sys_tables t, ${t.name}@source
         where  d.dbID = t.dbID
         and    d.name = 'source' )
```

Inverse Sichtendefinitionen (siehe Folie 27)

Auch die letzten beiden Sichtendefinitionen sind nicht als korrektes SQL formulierbar...

```
create view r@db1 ( stkCode, date, clsPrice) as
       select  stkCd, dat, clsPr
       from    r@dbI

create view r@db2 ( ${stkCd}, date ) as
       select  stkCd, clsPr, date
       from    r@dbI

create view ${stkCd}@db3 ( clsPrice, date ) as
       select  stkCd, clsPr, dat
       from    r@dbI
```