

Ja-(zu-)SQL

Evaluation einer Skriptsprache für Hauptspeicherdatenbanksysteme

Maximilian Schüle, Linnea Passing, Alfons Kemper, Thomas Neumann

{schuele, passing, kemper, neumann}@in.tum.de

HyPerScript

```
Statement      = ( ( VarDeclaration | VarDefinition | TableDeclaration | SelectStatement | EmbeddedSQL
                    | ReturnStatement | IfStatement | WhileStatement | ControlStatement | FunctionCall ) ";" ) * ;

VarDeclaration  = "var" NAME Type "=" Expression ;
VarDefinition   = NAME "=" Expression ;
TableDeclaration = "table" NAME "(" NAME Type ("," NAME Type)* ")";
SelectStatement = SQLSelect ("{" Statement "}") ("else" "{" Statement "}" ) ;
EmbeddedSQL     = SQLInsert | SQLUpdate | SQLDelete | CSVCopy ;
ReturnStatement = "return" NAME | "rollback" ;
IfStatement     = "if" "(" Expression ")" "{" Statement "}" "else" "{" Statement "}" ;
ControlStatement = "break" | "continue" ;
WhileStatement  = "while" "(" Expression ")" "{" Statement "}" ;
FunctionCall    = NAME "(" Expression ("," Expression ) * ")";
```

Listing 1: Sprachdefinition von *HyPerScript*: SQL{Select, Insert, Update, Delete} entsprechen den SQL:2003-Befehlen, Type den SQL-Datentypen, Expression den SQL-Ausdrücken. Neben klassischen Ausdrücken prozeduraler Sprachen ermöglicht EmbeddedSQL auf einzelnen Tupeln der SQL-Anfrage zu operieren.

Tensoroperationen

```
create or replace function pow(a_in float[][] , e_in int)
returns float[] as $$
var a=a_in; var e=e_in;
if( e<0 ) {
    e=e*-1;
    a=array_transpose(a);
}
var mask = 1<<63;
var result = array_identity(array_ndims(a));
while(mask>0){
    result=result*result;
    if(e&mask>0){
        result=result*a;
    }
    mask=mask>>1;
}
return result;
$$ language 'hyperscript' strict;
```

Listing 2: Binäre Exponentiation `pow()` von Tensoren implementiert in *HyPerScript*: als Eingabe wird ein Tensor und ein Exponent erwartet, eine Schleife mit Multiplikationen berechnet die Exponentiation.

Kreuzvalidierung

```
create or replace function linearregression(
x float[][], y float[][]) returns float[] as $$
return (array_transpose(x)*x)^-1*(array_transpose(x)*y);
$$ language 'hyperscript' strict;

create or replace function cross_validate(
x float[][], y float[][]) returns float as $$
var error=0; var n=array_length(x,2);
var m=array_length(x,1);
select index i from sequence(2,n-1){
    var weights_o=linearregression(
        array_slice(x,1,i-1,1,m)||array_slice(x,i+1,n,1,m),
        array_slice(y,1,i-1,1,n)||array_slice(y,i+1,n,1,1));
    error=error+((array_slice(x,i,i,1,m)*weights_o)[1][1]
        -y[i][1])^2;
}
error=error/(n-2); return error;
$$ language 'hyperscript' strict;
```

Listing 3: Einfache Kreuzvalidierung in *HyPerScript*: Mittels `array_slice()` und der Konkatination wird schrittweise eine Zeile aus dem Datensatz herausgeschnitten und als Testdatensatz verwendet. `array_slice()` erwartet als Argument den Tensor und für jede Dimension die Start- sowie Endposition.

Evaluation

