



Query Optimization

2. Exercise, Summer 2009

Due 2009-05-06.

1. Consider the following SQL queries. For each of them, perform the canonical translation, logical optimization and (plausible) physical optimization. Give a short justification for the physical optimization.

(a)

```
select distinct s.name, l.name  
from           students s, lectures l, attends a  
where          s.id=a.sid and a.lid=l.id and l.year=2007
```

(b)

```
select distinct s.name  
from           students s, lectures l, attends s, professors p  
where          s.id=a.id and a.lid=l.id and l.pid=p.id and p.name="Sokrates"
```

(c)

```
select distinct p.name, avg(e.grade)  
from           lectures l, professors p, exams e  
where          l.pid=p.id and e.lid=l.id and l.year=2007  
group by      p.id, p.name
```

2. Implement a program that accepts input of the form

```
select *  
from  
vorlesungen v  
professoren p  
where  
p.persnr=v.gelesenvon  
;
```

transforms it into a canonical execution plan, and executes the plan. You can assume that the input is well formed (one relation per line etc.).

Hints:

Parsing the input requires some effort, but try to keep the parsing part as generic as possible. We need an "SQL"-style parser for other exercises, too. The Java class *StreamTokenizer* might be handy for parsing.

Queries can be executed using the *tinydb* package from the web site. There are some examples in the *samples* directory. To check that everything is working, go to *samples* and execute `java -cp .:. ScanSample` (Windows users might have to replace `:` with `;`). It should give the names of all students. The example programs show how to use the system for basic tasks (scanning, joining, and selection), which is enough for this exercise. *JoinSample* shows a simple join plan.