Database Practicum

Course info

- Ján Mazák - mazak@dcs.fmph.uniba.sk
- Michal Rjaško – rjasko@dcs.fmph.uniba.sk
- **Evaluation:**
  - **3 homework assignments of 30 points each (given in roughly thirds of the semester)**
  - **12 lessons of 1 point each, at least 9 must be obtained**
    - within 3 days after the lesson, the solved tasks (at least half) must be se over by e-mail
  - A: 92 and more
  - B: 84 - 91 points
  - C: 76 - 83 points
  - D: 68 - 75 points
  - E: 60 - 67 points

Course plan

- Datalog
- SQL, DDL, DML
- working with database in Java
- Explain - query analysis / optimization
- SQLite, PostgreSQL

Declarative programming

- The program defines **what** needs to be calculated, but does not describe **how**
- All common database languages (e.g. SQL) do not even contain the means to describe the calculation
- The database itself chooses the calculation procedure according to the query and existing data
  (non-trivial optimization)

Prolog

- One of the most common languages for declarative programming
- Used in natural language processing and artificial intelligence (e.g. IBM Watson)
- The program consists of facts and rules of inference
- Prolog interpreter makes inference (deriving the consequences of rules from known facts) using backtracking
- We will use it as an environment to work with Datalog

Prolog / Datalog

- Rule syntax:

  p :- x, y, \+ z.

  good_car(X) :- car(X), reliable(X), fast(X), costs_less(X, 30 000).

  reliable(toyota).

- Semantics:

  p is true if x and y are true and z is false

- The rules define new *predicates*, e.g. good_car, using existing predicates
- The number of predicate arguments is *arity*
- A set of rules with the same head serves to express a logical or, for example:
  - p :- x.
    p :- y.

Prolog / Datalog

- Example of a datalog rule:

  res(N, J) :- emp(_, N, J, _, _, S, _, _), S >= 2000.

- on the left side only one positive atom
- Variables start with a capital letter
- Constants in lowercase
- _ means anonymous variable

  (If _ is used in multiple places, the values may not be the same, they are different variables)
- The IS operator is used to evaluate arithmetic expressions:
  - E.g. X is 2+3,
  - not X = 2+3

    (symbol = is interpreted as the unification of terms and no arithmetic operation occurs)

Prolog / Datalog

bigger(elephant, horse).

bigger(horse, donkey).

bigger(donkey, dog).

bigger(donkey, monkey).

?- bigger(donkey, dog).

true

?- bigger(monkey, elephant).

false

?- bigger(elephant, monkey).

false

Prolog / Datalog

```
bigger(elephant, horse).
bigger(horse, donkey).
bigger(donkey, dog).
bigger(donkey, monkey).

is_bigger(X, Y) :- bigger(X, Y).
is_bigger(X, Y) :- bigger(X, Z), is_bigger(Z, Y).
```

```
?- is_bigger(monkey, elephant).
false
?- is_bigger(elephant, monkey).
true

?- is_bigger(elephant, X).
X = horse
X = donkey
X = dog
X = monkey
false
```

Negation

- Atoms (claims) to which proof can be derived from facts (using the rules of inference) are assumed to be true in the Prologue
- Other things are false (negation as failure)
- it is impossible to add a fact of falsity (the head of the rule does not contain negation)
- the so-called assumption of a closed world (what we have in the database is true, the rest is false)
- Truthfulness is generally not the same as provability

(“This theorem cannot be proven", more in mathematical logic)

Negation

- Negation combined with recursion can cause problems in interpretation:

  p :- \+ q.

  q :- \+ p.

- What the world described by this program looks like?

two stable models: p is true and q is not; or vice versa (more about models on Database systems)

- All related problems can be avoided if we only use safe rules: every variable must occur in some positive fact
- Example of a unsafe rule (because of Y and Z):

  p(X, Y) :- a(X), \+ b(Y, Z), Z is Y + 1.

Prolog vs. Datalog

- When writing down rules, always list positive things, and only then negative ones. In the datalog it does not matter, but the prologue takes this into account in the calculation.

Práca s datalogom: SWI-Prolog

- Three options:
  - on servers *cvika*, login using ssh on cvika.dcs.fmph.uniba.sk

    (username/password from AISe)
  - using SWI-Prolog on your local computer
  - online at https://swish.swi-prolog.org/

- We recommend opening 3 windows
  - In one, you edit a file with queries, eg. **vim queries_emp.pl**
  - In the second window, you are running the Prolog environment: **swipl -s queries_emp.pl**
  - in the third window you have a database (list of facts)

Práca s datalogom

- After you write a query to a file, you need to save it to disk
  (vim: ESC, ":w", ENTER).
- then compile the new version: **make.** (even with that dot)
  - Be sure to check if the compiler reports errors and fix them if necessary
- Calculation of queries:
  ?- q(job(J)).
- the predicate "q(_)" is used to nicely format the output and eliminate apparent duplicates (Prolog does full backtracking and can find a specific value in multiple branches)

Databáza EMP

```
%emp(Empno, Ename, Job, Manager, Hiredate, Sal, Deptno)
emp(7839, king, president, null, 19811117, 5000, 10).
emp(7698, blake, manager, 7839, 19810501, 2850, 30).
emp(7782, clark, manager, 7839, 19810609, 1500, 10).
emp(7566, jones, manager, 7839, 19810402, 2975, 20).
emp(7654, martin, salesman, 7698, 19810928, 1250, 30).
emp(7499, allen, salesman, 7698, 19810220, 1600, 30).
emp(7844, turner, salesman, 7698, 19810908, 1500, 30).
emp(7900, james, clerk, 7698, 19811203,  950, 30).
emp(7521, ward, salesman, 7698, 19810222, 1250, 30).
emp(7902, ford, analyst, 7566, 19811203, 3000, 20).
emp(7369, smith, clerk, 7902, 19801217,  800, 20).
emp(7788, scott, analyst, 7566, 19821209, 3000, 20).
emp(7876, adams, clerk, 7788, 19830112, 1100, 20).
emp(7934, miller, clerk, 7782, 19820123, 1300, 10).
```

```
%dept(Deptno, Dname, Location)
dept(10, accounting, newyork).
dept(20, research, dallas).
dept(30, sales, chicago).
dept(40, operations, boston).
```