Exercise 2
Inclusion Dependencies

- Deadline: **Monday, 01.12.14**
- The admission to the exam requires *all* exercises to be solved.
- The exercises should be solved in teams of two students.
- The Metanome project is available at GitHub:
  `https://github.com/HPI-Information-Systems/Metanome`
- The datasets and supplemental material can be found at network drive S:
  `\\fs3\bbs\DPDC`
- The submission system can be found at:
  `https://www.dcl.hpi.uni-potsdam.de/submit/`
- To solve an exercise, please submit a zip file containing the following items:
  - **<algorithm_name>.jar**: An executable Metanome algorithm.
  - **<algorithm_name>.zip**: The algorithm's source code (maven project).
  - **<algorithm_name>_docu.pdf**: Short documentation of the algorithm.
  - **<algorithm_name>_pres.pptx/ppt/pdf**: Two slides presentation of the algorithm.

Task 1: Inclusion Dependencies - A discovery algorithm

Write an algorithm that discovers *all unary* inclusion dependencies on the given datasets. The rules for your implementation are as follows:

a) The algorithm discovers *exact* results, so no approximate or fuzzy results are allowed.

b) The algorithm is not allowed to use parallelization.

c) The algorithm implements the Metanome interface and is compatible with Metanome.

d) The algorithm takes an arbitrary number of tables as input.

e) The algorithm ignores NULL values.

Name	Type	Equatorial diameter	Mass	Orbital radius	Orbital period	Rotation period	Confirmed moons	Rings	Atmosphere
Mercury	Terrestrial	0.382	0.06	0.47	0.24	58.64	0	no	minimal
Venus	Terrestrial	0.949	0.82	0.72	0.62	−243.02	0	no	CO_2, N_2
Earth	Terrestrial	1.000	1.00	1.00	1.00	1.00	1	no	N_2, O_2, Ar
Mars	Terrestrial	0.532	0.11	1.52	1.88	1.03	2	no	CO_2, N_2, Ar
Jupiter	Giant	11.209	317.8	5.20	11.86	0.41	67	yes	H_2, He
Saturn	Giant	9.449	95.2	9.54	29.46	0.43	62	yes	H_2, He
Uranus	Giant	4.007	14.6	19.22	84.01	−0.72	27	yes	H_2, He
Neptune	Giant	3.883	17.2	30.06	164.8	0.67	14	yes	H_2, He

Planet	Rotation Period	Revolution Period
Mercury	58.6 days	87.97 days
Venus	243 days	224.7 days
Earth	0.99 days	365.26 days
Mars	1.03 days	1.88 years
Jupiter	0.41 days	11.86 years
Saturn	0.45 days	29.46 years
Uranus	0.72 days	84.01 years
Neptune	0.67 days	164.79 years
Pluto	6.39 days	248.59 years

Planet	Synodic period	Synodic period (mean)	Days in retrograde
Mercury	116	3.8	~21
Venus	584	19.2	41
Mars	780	25.6	72
Jupiter	399	13.1	121
Saturn	378	12.4	138
Uranus	370	12.15	151
Neptune	367	12.07	158

Planet	Mean distance	Relative mean distance
Mercury	57.91	1
Venus	108.21	1.86859
Earth	149.6	1.3825
Mars	227.92	1.52353
Ceres	413.79	1.81552
Jupiter	778.57	1.88154
Saturn	1,433.53	1.84123
Uranus	2,872.46	2.00377
Neptune	4,495.06	1.56488
Pluto	5,869.66	1.3058

Sign	House	Domicile	Detriment	Exaltation	Fall	Planetary Joy
Aries	1st House	Mars	Venus	Sun	Saturn	Mercury
Taurus	2nd House	Venus	Pluto	Moon	Uranus	Jupiter
Gemini	3rd House	Mercury	Jupiter	N/A	N/A	Saturn
Cancer	4th House	Moon	Saturn	Jupiter	Mars	Venus
Leo	5th House	Sun	Uranus	Neptune	Mercury	Mars
Virgo	6th House	Mercury	Neptune	Pluto, Mercury	Venus	Saturn
Libra	7th House	Venus	Mars	Saturn	Sun	Moon
Scorpio	8th House	Pluto	Venus	Uranus	Moon	Saturn
Sagittarius	9th House	Jupiter	Mercury	N/A	N/A	Sun
Capricorn	10th House	Saturn	Moon	Mars	Jupiter	Mercury
Aquarius	11th House	Uranus	Sun	Mercury	Neptune	Venus
Pisces	12th House	Neptune	Mercury	Venus	Pluto, Mercury	Moon

Complexity: $O(n^2\text{-}n)$ for n attributes

Example:
10 attr ~ 90 checks
1,000 attr ~ 999,000 checks

Abbildung 1: Unary Inclusion Dependencies.

Note that in contrast UCC and FD discovery, IND discovery uses *multiple tables*. INDs should therefore be discovered within and between tables. You can re-implement an existing algorithm from literature or find your own algorithm. To test and evaluate the algorithm, use the datasets provided on the network share. Your algorithm should at least be able to process the WDC dataset! Before submitting your algorithm, check that it correctly executes within Metanome!

BONUS TASK: If you like to dive deeper, you can try to find n-ary inclusion dependencies as well. Can you think of pruning rules for the n-ary discovery? You can also try parallelization or approximate strategies on your algorithm. If you made changes to your main algorithm, please submit them as a separate algorithm, e.g. <algorithm_name>_nary.jar

Task 2: Documentation

Write a short (max one A4 page) documentation for your algorithm describing the algorithm that you implemented:

a) Describe the algorithm's basic idea. How does the algorithm cope with the complexity of the given task?

b) If you used an algorithm from literature, provide a reference to the according publication.

c) If you came up with an own approach, provide one or two arguments why it is or could be better than related algorithms.

d) If your algorithm implements an adaption or optimization of existing approaches, describe these briefly.

e) State if your algorithm (jar) should be published as Metanome algorithm providing you as the authors.

f) *If you solved a bonus task, please discuss your findings here as well.*

In the same document, answer the following questions:

a) How many inclusion dependencies did your algorithm find on the provided datasets?

b) How long did the discovery take on each dataset and what machine did you use?

c) Did you discover any limitations of your approach (e.g. runtime or memory consumption) that made computing a certain dataset impossible?

d) Why is it a good idea to ignore NULL values for IND discovery?

Task 3: Presentation

Prepare two slides for a short, 5 min presentation of your algorithm in the lecture. One slide about the algorithm and one slide about its performance. Here are some ideas for these slides:

a) Explain the idea of your approach and why you think it works well in comparison to others.

b) How did your algorithm perform on the given datasets?

c) Did you make any interesting observations?

d) What lets your algorithm crash?

e) How hard was it to implement your algorithm?

f) *This is also a good time to introduce an optimization or adaption of your algorithm.*

Note that each team will present its work once!