# Project 2 – Advanced Databases

## Problem statement

We want to execute the following query:

(?a)---follows--->(?b)---friendOf--->(?c)---likes--->(?d)---hasReview--->(?e)

Over the given datasets that are described in the section “Dataset description”

In order to solve this query we need to be able to join two tables.

The required join algorithms are further described in the section Algorithm description

## Algorithm description

Hash-join

Input two tables / list of rows /triples with one value for each table on which we join.

Join on last value of table1 and first value of table2

Use table1 to build dict with join attribute as key and list of rows that have it as values

Iterate over rows of table2 use join attribute to get all rows of table1 with value, join them together.

Return them

Sort-merge-join

Input two tables / list of rows /triples with one value for each table on which we join.

Join on last value of table1 and first value of table2

Sort both tables on join key

Iterate over table1 and table2 by increasing the indices

If same values try to first increase indicee of table2 till not same value anymore then go back to previous indice of table2 + 1

Skip sort merge join

Same as sort merge join but trying to increase by more than one

Join multiple tables

## Dataset description

The dataset consists of two files:

File1 “100k.txt” with 109’0310 triples.

Num friendOf triples: 45712

Num follows triples: 31887

Num likes triples: 1032

Num hasReview triples: 1453

Upper limit join: 2’185’693’176’253’824

File2 “watdiv.10M.nt” with 10’916’458 triples.

Num friendOf triples: 4’491’142

Num follows triples: 3’289’307

Num likes triples: 112’401

Num hasReview triples: 149’634

Upper limit join: 248’462’961’060’947’778’084’996 ≈ 2,5 \* 10^23

## Experiment and analysis

### Hash join

Join needed 0.73

Join needed 6.63

Join needed 76.08

Hash join query needed: 83.91 seconds to run the procedure

### Sort merge join

Join needed 1.96

Join needed 5.06

Join needed 17.51

Sort merge join query needed: 24.53 seconds to run the procedure

### Sort merge join skip

Join needed 1.96

Join needed 4.09

Join needed 16.39

Sort merge join skip query needed: 22.80 seconds to run the procedure

In my experiment sort-merge join with skipping was the fastest

sort merge join is second

hash join is last

## Conclusion

Use sort merge join with skipping if you want a fast join algorithm