Lab 1 Big Data Management
Problem B

Alejandro Delgado, Viktoria Gagua

April 29, 2025

Task B (Querying)

Listing 1: Query for Problem Set B

```
// Problem B    ANALYTICS QUERIES
// In this block we run four standalone Cypher queries for analytics.
// ------------------------------------------------------
// B.1   Top3   mostcited  papers per conference year
//



// We (a) group by (booktitle, year), (b) count inbound
// :CITES edges per Paper, (c) order DESC, then (d)
// slice the first three via listcomprehension.
MATCH (conf:Inproceedings)-[:HAS_TYPE]->(t:Type {name:'conference'})
MATCH (p:Paper)-[:PUBLISHED_IN]->(conf)
OPTIONAL MATCH (p)<-[c:CITES]-()
WITH conf.booktitle AS conference,
    conf.year     AS year,
    p,
    COUNT(c)      AS citations
ORDER BY conference, year, citations DESC
WITH conference, year,
    COLLECT({paper:p.title, citations:citations}) AS papers
RETURN conference,
      year,
      [x IN papers[0..3] | {title:x.paper, citation_count:x.citations}] AS
         top_cited_papers;

// B.2 Authors who have
//      published in 4 distinct editions of the same conference

// We first pair each Paper with its Author and the (booktitle,year)
// of the hosting conference. We then count how many *distinct*
// years each author has contributed to that series. Authors with
//   4  appearances are collected as community_members.
MATCH (conf:Inproceedings)-[:HAS_TYPE]->(t:Type {name:'conference'})
MATCH (p:Paper)-[:PUBLISHED_IN]->(conf)
MATCH (p)-[:WRITTEN_BY]->(a:Author)
WITH conf.booktitle AS conference,
    conf.year     AS year,
    a
WITH conference,
    a,
    COUNT(DISTINCT year) AS editions
WHERE editions >= 4
```

```
WITH conference,
     COLLECT(a.name) AS community_members,
     COUNT(a)       AS community_size
RETURN conference,
       community_size,
       community_members
ORDER BY community_size DESC;

// B.3  Twoyear  rolling impact factor for journals
// Rationale: We mimic the Thomsonstyle IF but  inhouse .
// 1.  We determine the *current* year by max(Time.year).
// 2.  We fetch papers from the previous two full years.
// 3.  We count inbound citations to those papers.
// 4.   Impactfactor  = citations   paper_count (guarded
//    against   dividebyzero  ).
//
// Step1    compute current year once and reuse.
MATCH (t:Time)
WITH MAX(toInteger(t.year)) AS current_year

// Step2    gather candidate papers per Journal.
MATCH (j:Journal)-[:HAS_PUBLICATION_DATE]->(t:Time)
WHERE toInteger(t.year) >= current_year - 2
  AND toInteger(t.year) < current_year
MATCH (p:Paper)-[:PUBLISHED_IN]->(j)
WITH j,
     current_year,
     COLLECT(p) AS papers,
     COUNT(p)  AS paper_count

// Step3    fold citation counts.
UNWIND papers AS paper
OPTIONAL MATCH (paper)<-[c:CITES]-()
WITH j,
     current_year,
     paper_count,
     COUNT(c) AS citation_count

// Step4    compute IF (float division).
WITH j,
     current_year,
     paper_count,
     citation_count,
     CASE WHEN paper_count > 0
          THEN 1.0 * citation_count / paper_count
```

```
         ELSE 0 END AS impact_factor
RETURN j.name     AS journal,
       current_year,
       paper_count  AS papers_last_two_years,
       citation_count AS citations,
       impact_factor
ORDER BY impact_factor DESC;

// B.4 Author hindex
// We implement the classic Hirsch hindex: the greatest
// h such that the author has h papers each cited h times.
// Steps:
//       collect citation counts per paper, sorted DESC
//       reduce across the array to compute h.
MATCH (a:Author)
MATCH (p:Paper)-[:WRITTEN_BY]->(a)
OPTIONAL MATCH (p)<-[c:CITES]-()
WITH a, p, COUNT(c) AS citations
ORDER BY a.name, citations DESC
WITH a,
     COLLECT(citations) AS citation_counts
WITH a,
     citation_counts,
     REDUCE(h = 0, i IN RANGE(0, SIZE(citation_counts)-1) |
       CASE WHEN i < citation_counts[i] THEN i+1 ELSE h END) AS h_index
RETURN a.name            AS author,
       h_index,
       citation_counts[0..5] AS top_citations
ORDER BY h_index DESC, a.name
LIMIT 20;
//                                                          FINALE
```