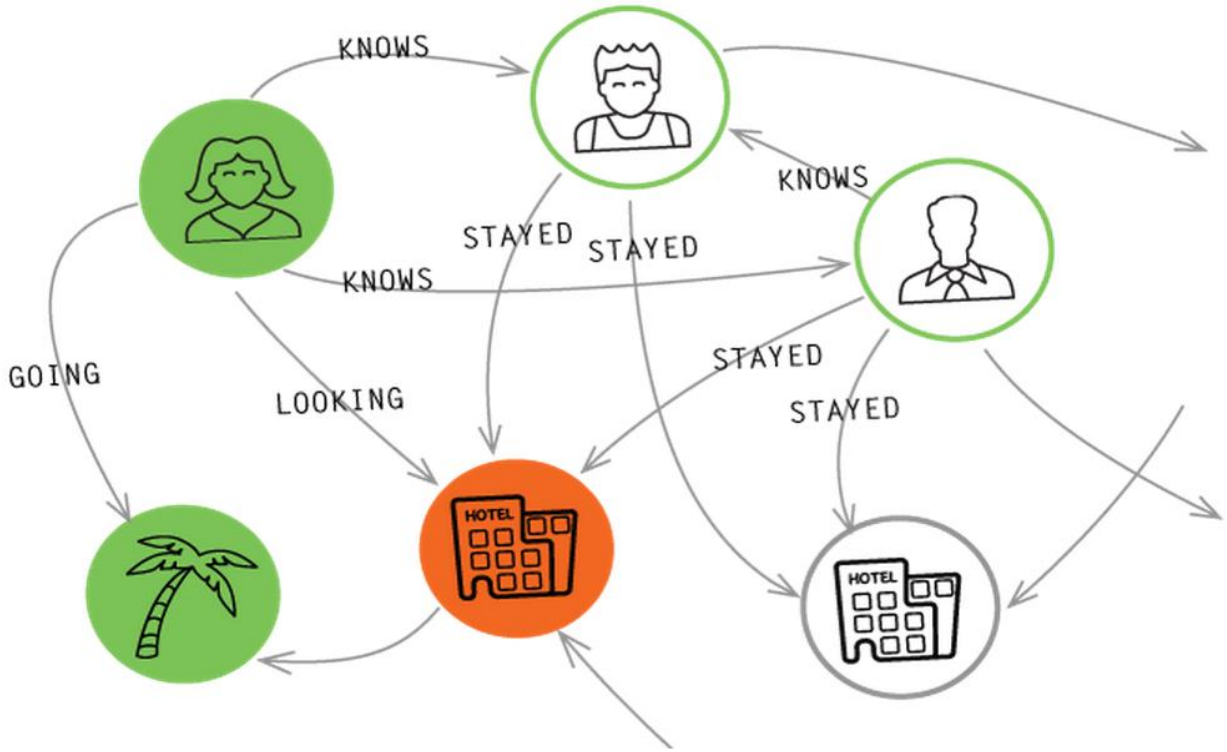
Graph Databases

Philipp Fleck, Bernhard Manfred Gruber

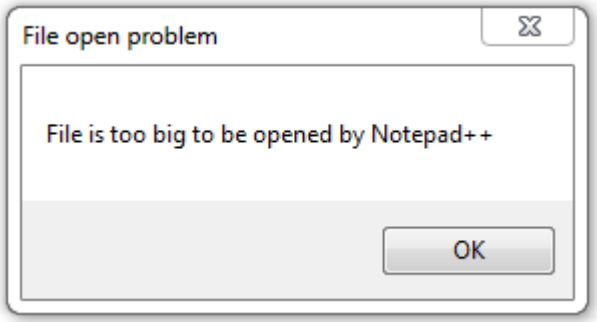
Overview

- Big files, Unicode and CSV
- Example (Wikipedia)
- Graph-DB Concepts
- Neo4j vs MS-SQL
  - Queries
  - Performance
- Conclusion



Big Data – Big Files – Big Problems

What if ...



- you have a file with 49 GiB?
- you have a file with 164,379,808 lines?
- your editor search&replaces 27 lines/s
- a line is longer than a String can hold?
- String.Split() fails to allocate enough memory?
- a regular expression runs 12 minutes
- your file parser spends only 20% on disk IO?
- each line is an INSERT statement and your DB runs only 1000 queries/s?
- LocalDB cannot store tables larger than 10 GiB?
- you have to compute your primary keys yourself?

Unicode and CSV

- What‘s wrong with this code?

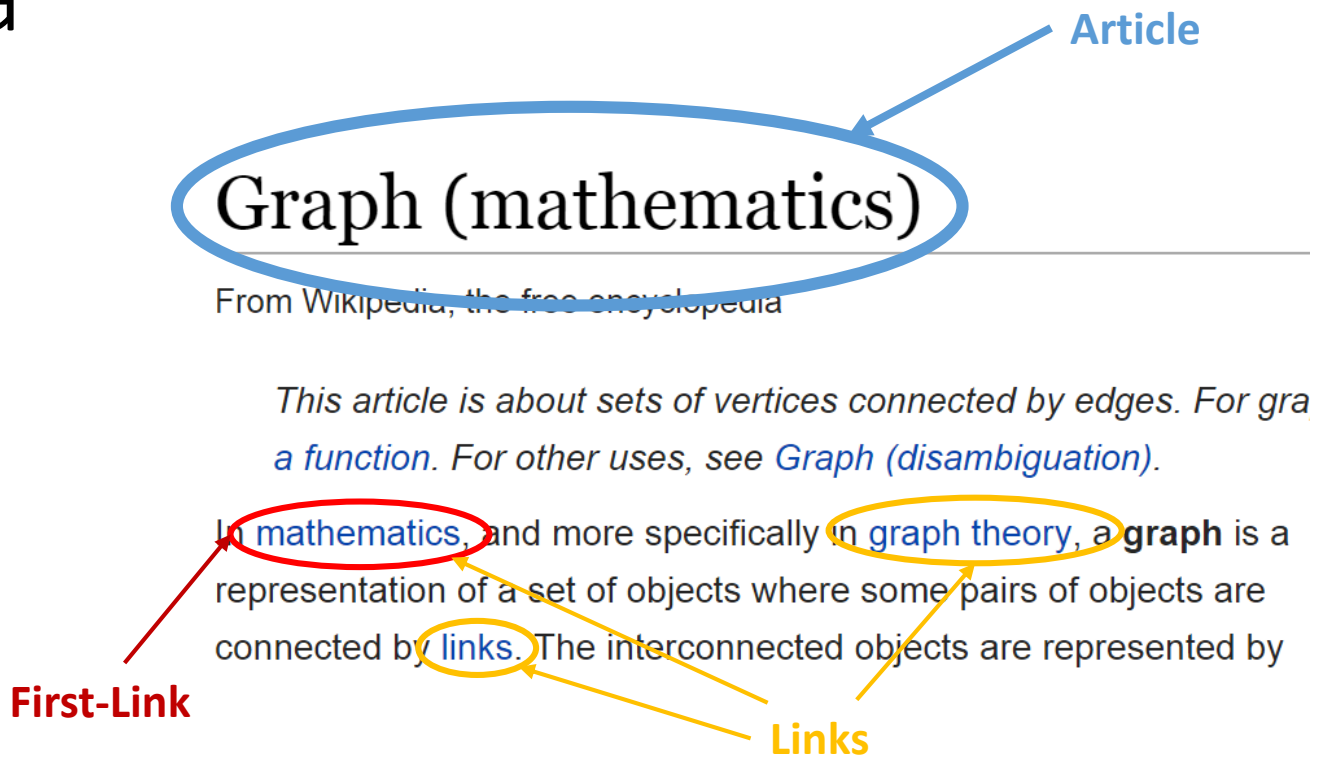
```
var t = page.Text.Substring(0, 100);
writer.WriteLine(t);
```

- Varchar vs. nvarchar
- MS SQL Server: no UTF-8
  Neo4j: only UTF-8
- MS SQL CSV: no enquoted fields, no field interpretation
  Neo4j CSV: enquoting optional, requires escapting quotes
- String equality

```
WHERE str1 = str2 COLLATE ..._BIN
```

Example Wikipedia

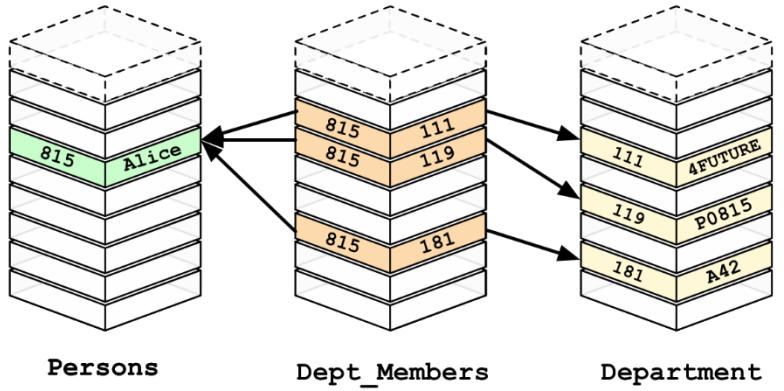
- Wikipedia English
  - 15,113,788 articles
  - 164,379,808 links
- Wikipedia Plattdüütsch
  - 31,814 articles
  - 299,032 links
  - 21,993 first-links
- Philosophy-Game
- 5-clicks-to-Jesus

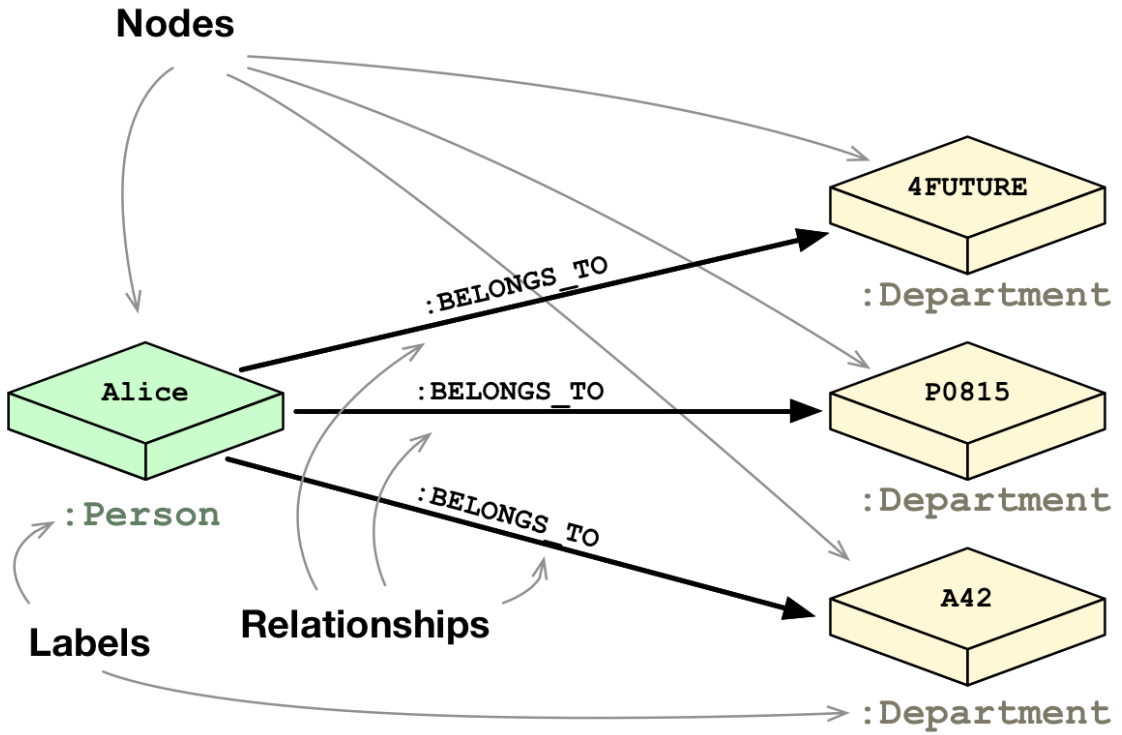


Graph-DB Concepts

- Ideas
  - RDBMS: Slow Joins
  - Natural Relations
  - Graph Theory (math)

- Components
  - Nodes (Entities)
  - Edges (Relationships)
  - Properties (Data)





Neo4j vs SQL (Basics Queries)

```
CREATE (p:Page {id:1, title:"Philosophy"})
CREATE (l:Page {id:2, title:"Logic"})
CREATE (l)-[:links_to]->(p);
```

```
INSERT INTO Pages (id, title)
VALUES (1, 'Philosophy'), (2, 'Logic');
INSERT INTO Links VALUES (2, 1);
```

```
MATCH (p:Page {title:"Philosophy"})
SET p.text="Hello Graph";
```

```
UPDATE Pages SET text='Hello Graph,
WHERE title='Philosophy';
```

```
MATCH (p:Page {title:"Philosophy"})
DELETE p;
```

```
DELETE FROM Pages
WHERE title='Philosophy';
```

```
MATCH (p:Page {title:"Philosophy"})
MATCH (p)<-[:links_to]-(a)
RETURN a;
```

```
SELECT a.*
FROM Pages p
  JOIN Links l ON p.id=l.dst
  JOIN Pages a ON a.id=l.src
WHERE p.title='Philosophy';
```

Neo4j vs SQL (Advanced Queries)

```
MATCH (p:Page {title:"Philosophy"})
MATCH (p)<-[:first links_to*1..3]-(a)
RETURN DISTINCT a;
```

```
MATCH (p:Page {title:"Philosophy"})
MATCH (p)<-[:first links_to*]-(a)
RETURN DISTINCT a;
```

```
MATCH path=shortestPath(
   (p:Page {title:"Philosophy"})
     <-[:first links_to*]-
   (g:Page {title:"Graph"})
) RETURN path;
```

- Dijkstra, A*

```
JOIN UNION JOIN UNION JOIN ...
```

Recursive Join ☺
CONNECT BY / Common Table Expression

```
WITH temp (id) AS (
  SELECT p.id
  FROM Page p
  WHERE p.title='Philosophy'
 UNION ALL
  SELECT fl.src
  FROM FirstLink fl
    JOIN temp t ON fl.dst=t.id
)
SELECT a.*
FROM Page a
  JOIN temp t ON a.id=t.id;
```

Neo4j vs SQL (Performance)

Operation		Neo4j		SQL	
Insert	All	3m30s		12m45s	
	Pages (31,841)	4.7s		4.6s	
	Links (299,032)	3m15s		12m	
	First Links (21,993)	11.3s		42s	
Find sources with first links to Philosophy (33)		50ms		32ms	
Find sources with links to Jesus		Count	All Data	Count	All Data
	1 Hop (166)	40ms	38ms	37ms	35ms
	2 Hops (2,057)	55ms	142ms	103ms	105ms
	3 Hops (14,240)	91ms	682ms	210ms	375ms
	4 Hops (25,558)	2s	4.8s	421ms	452ms
	5 Hops (27,609)	aborted after 10min		3s	3s

Conclusion

Pro

- Fast Insert
- Easy Queries

Contra

- Slow
- Not mature
- Memory consumption

Maybe Pro (didn’t looked into it)

- Scalability
- Other DBs than Neo4j

Our Recommendation

- Use Cypher where time is not important
- Cypher to SQL transpiler would be awesome

Thanks

Questions?