

Graphs and graph databases

Introduction to graph databases

NoSQL: Lecture 1

Piotr Fulmański

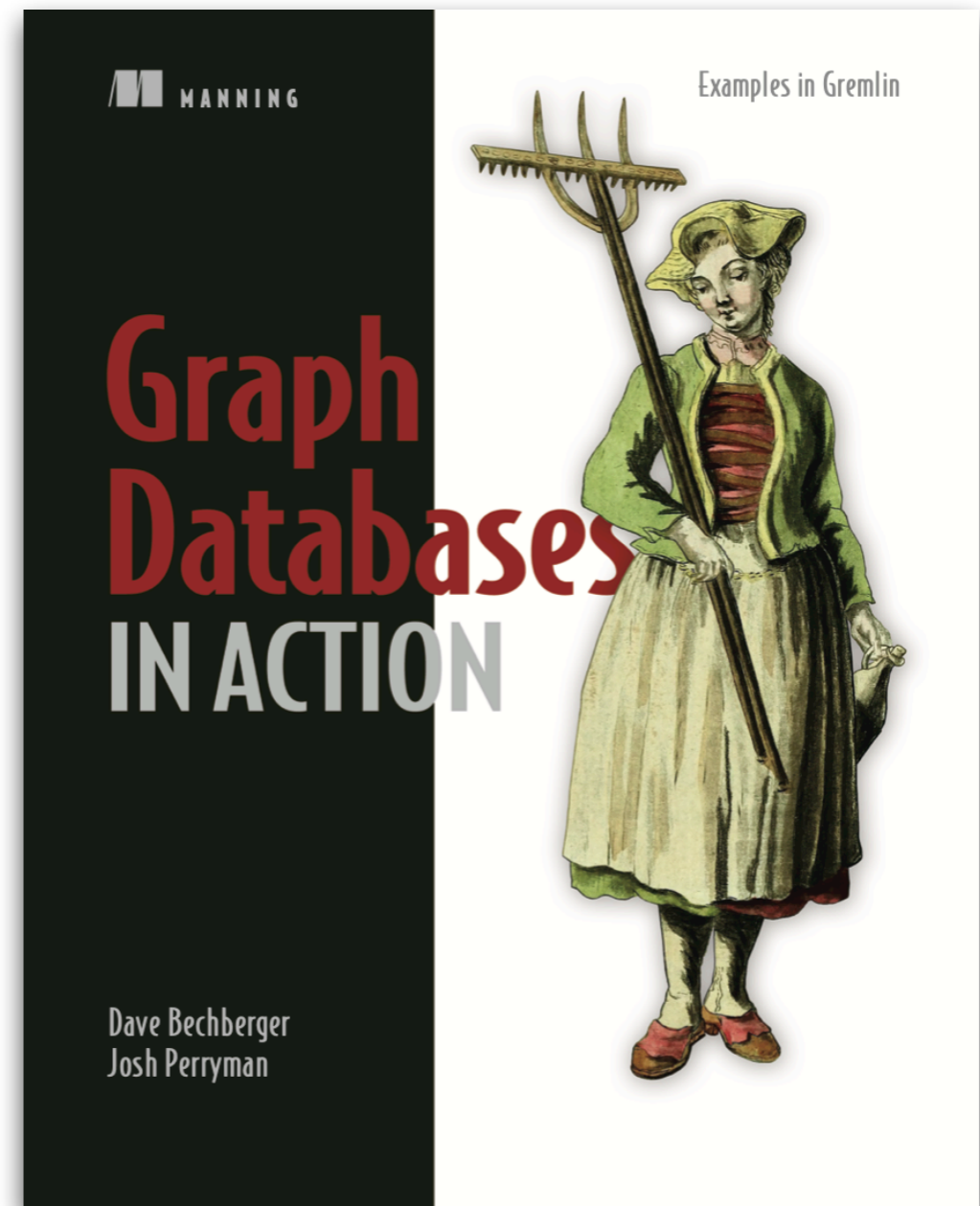


FACULTY OF MATHEMATICS
AND COMPUTER SCIENCE
University of Lodz

Graph databases In Action

by Dave Bechberger
and Josh Perryman

Manning Publications, 2020



NoSQL Theory and examples

by Piotr Fulmański

Piotr Fulmański, 2021

PIOTR FULMAŃSKI

NoSQL Theory and examples

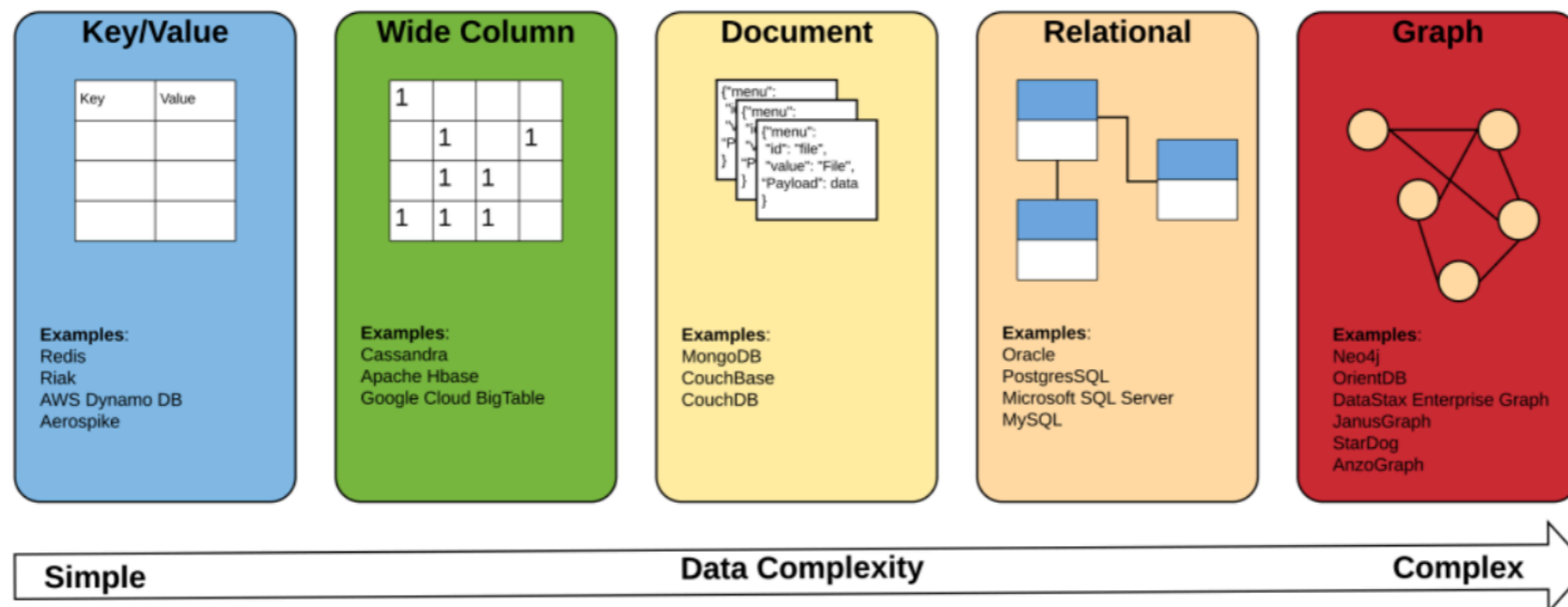


SIMPLE INTRODUCTION SERIES

Graphs and graph terminology

- **Vertices** (singular: vertex) a.k.a. nodes.
- **Edges** a.k.a. relationships, links, or connections.
- **Properties**.
- **Graph** - A set of vertices and edges along with their properties.

Comparison with other types of databases



Database engine types ordered by data complexity. Source: [Bec]

Do you really need another one database?

WHAT IS IMPORTANT

All of them stores data in one of the characteristic way. So you have few different methods to store data, but your data are dumb – they are just data. Anything more than this you have to infer on your own. There are situation when you want your data to „speak”.

Sometimes the way we organize our data is also a kind of information.

Relations between entities are as important, or more important, than the entities within data.

Graph stores are built around the simple and general-purpose

node-relationship-node

data structure. The key question is: *Do we really need a new database type?*

Do you really need another one database?

DIFFERENT RESULT TYPES - RELATIONAL DATABASE

Orders		
id	name	address
1	John Smith	123 Main. St
2	Jane Right	643 Park St.

Products		
id	product_name	cost
123	widget 1	5.95
234	widget 2	10.76

```
SELECT id,  
       name,  
       address,  
       null AS product_name,  
       null AS cost,  
       'Order' AS object_type  
FROM Orders  
UNION  
SELECT id,  
       null AS name,  
       null AS address,  
       product_name,  
       cost,  
       'Product' AS object_type  
FROM Products;
```

Do you really need another one database?

DIFFERENT RESULT TYPES - RELATIONAL DATABASE

Orders		
id	name	address
1	John Smith	123 Main. St
2	Jane Right	643 Park St.

Products		
id	product_name	cost
123	widget 1	5.95
234	widget 2	10.76

```
SELECT id,  
       name,  
       address,  
       null AS product_name,  
       null AS cost,  
       'Order' AS object_type  
FROM Orders  
UNION  
SELECT id,  
       null AS name,  
       null AS address,  
       product_name,  
       cost,  
       'Product' AS object_type  
FROM Products;
```

<u>id</u>	<u>Name</u>	<u>Address</u>	<u>product_name</u>	<u>cost</u>	<u>object_type</u>
1	John Smith	123 Main St	<null>	<null>	Order
2	Jane Right	234 Park St	<null>	<null>	Order
123	<null>	<null>	widget 1	5.95	Product
234	<null>	<null>	widget 2	10.76	Product


Do you really need another one database?

DIFFERENT RESULT TYPES - GRAPH DATABASE


All data as vertices in a graph

Orders		
id	name	address
1	John Smith	123 Main. St
2	Jane Right	643 Park St.


Products		
id	product_name	cost
123	widget 1	5.95
234	widget 2	10.76




order	
id	1
name	John Smith
address	123 Main St



product	
id	123
product_name	widget 1
cost	5.95



order	
id	2
name	Jane Right
address	643 Park St.



product	
id	234
product_name	widget 2
cost	10.76

```
gremlin> g.V().valueMap(true)
==>[label:order, address:[123 Main St], name:[John Smith], id:1]
==>[label:order, address:[234 Park St], name:[Jane Right], id:2]
==>[label:product, cost:[10.76], id:234, product_name:[widget 2]]
==>[label:product, cost:[5.95], id:123, product_name:[widget 1]]
```

Do you really need another one database?

PATHS

River crossing puzzle: you have a fox, a goose, and a bag of barley that must be transported across a river by a farmer on a boat. However, this movement is bound by the following constraints:

- The boat can only carry one item in addition to the farmer on each trip.
- The farmer must go on each trip.
- The fox cannot be left alone with the goose or it will eat it.
- The goose cannot be left alone with the grain or it will eat it.

Do you really need another one database?

PATHS

Let's start by modeling the initial state of your system as a vertex in your graph, which you'll call: **TGFB_** with each character representing part of the problem:

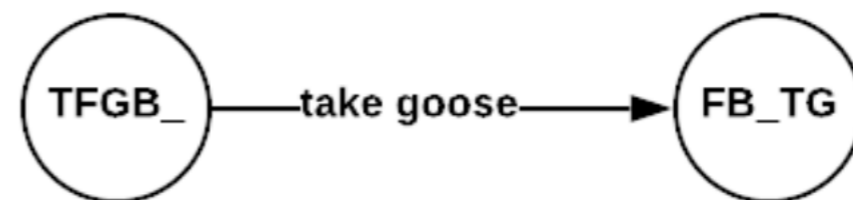
T – the boat & farmer

G – the goose

F – the fox

B – the barley

_ – the river



Do you really need another one database?

PATHS

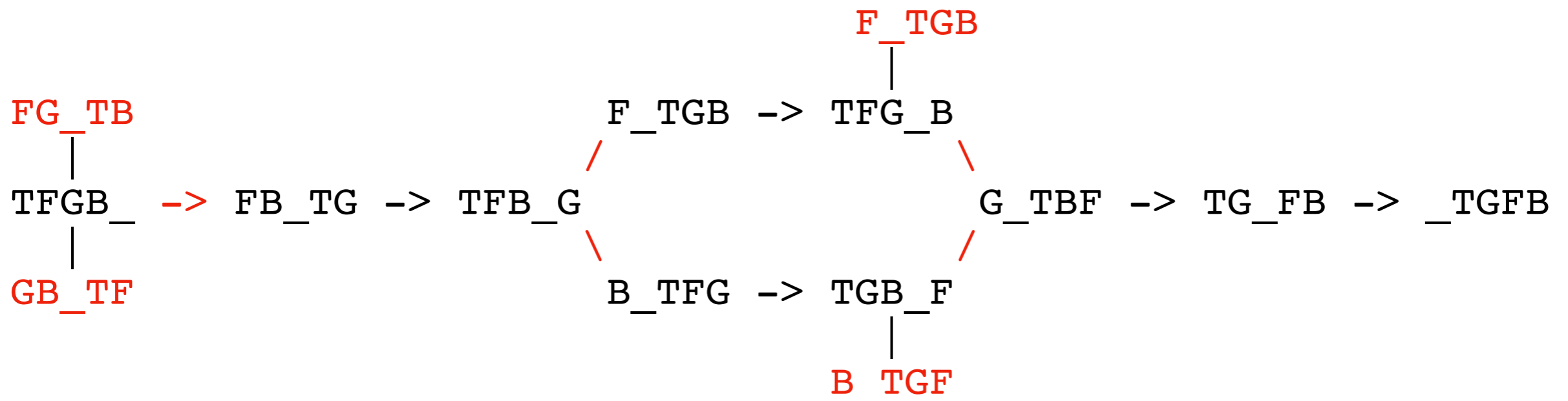
River crossing puzzle full graph:

[Let's play and draw it here]

Do you really need another one database?

PATHS

River crossing puzzle full graph:



Red state - unacceptable state

\ - bidirectional edge

Do you really need another one database?

PATHS

River crossing puzzle full graph:

[Put drawing here]

```
g.V('TFGB_').  
  repeat(  
    out()  
  ).until(hasId('_TGFB')).  
  path().next()
```

```
TFGB_ -take goose-> FB_TG -take empty-> TFB_G -take barley-> F_TGB -return  
goose-> TFG_B -take fox-> G_TBF -return empty-> TG_FB -take goose-> _TGFB
```

```
TFGB_ -take goose-> FB_TG -take empty-> TFB_G -take fox-> B_TFG -return goose->  
TGB_F -take barley-> G_TBF -return empty-> TG_FB -take fox-> _TGFB
```

Do you really need another one database?

RELATIONAL DEPENDENCIES

Table: person

PK : id
: name

Table: project

PK : id
: name

Table: team_member

PK, FK1 : person_id
PK, FK2 : project_id
: role

Do you really need another one database?

RELATIONAL DEPENDENCIES

Table: person

PK : id
: name

Table: project

PK : id
: name

Table: team_member

PK, FK1 : person_id
PK, FK2 : project_id
: role

You want to find all persons who have ever worked with XYZ in the same project.

Do you really need another one database?

RELATIONAL DEPENDENCIES

Table: person

PK : id
: name

Table: project

PK : id
: name

Table: team_member

PK, FK1 : person_id
PK, FK2 : project_id
: role

If you want to find all persons who have ever worked with XYZ in the same project you can use the following SQL query:

```
SELECT p2.name, p.name
FROM person p1
      JOIN team_member t1 ON (p1.id = t1.person_id)
      JOIN project p ON (t1.project_id = p.id)
      JOIN team_member t2 ON (p.id = t2.person_id)
      JOIN person p2 ON (t2.person_id = p2.id)
WHERE p1.name = 'XYZ';
```

Do you really need another one database?

RECURSIVE QUERIES USING CTE

id	name	manager_id
1	Alice	NULL
2	Betty	1
3	Carolina	1
4	Diana	3

```
WITH RECURSIVE hierarchy AS (  
  SELECT id, name, manager_id, 1 as level  
  FROM employee  
  WHERE id = 1  
  UNION  
  SELECT e.id, e.name, e.manager_id, h.level + 1 AS level  
  FROM employee AS e  
  INNER JOIN hierarchy AS h ON h.id = e.manager_id  
)  
SELECT * FROM hierarchy;
```

id	name	manager_id	level
1	Alice	NULL	1
2	Betty	1	2
3	Carolina	1	2
4	Diana	3	3

Do you really need another one database?

RECURSIVE QUERIES USING CTE

Very often SQL recursive data are no longer recursive in graph databases, so it's much simpler to make a query.

You can try this query:

```
g.V().  
  repeat(  
    out('works_for')  
  ).path().next()
```

Ask yourselves

Is my problem a graph problem?

- What problem are you trying to solve?
 - Selection / search

Ask yourselves

Is my problem a graph problem?

- What problem are you trying to solve?
 - Selection / search
 - Give me everyone who works at X?
 - Who in my system has a first name like John?
 - Locate all stores within X miles?

Ask yourselves

Is my problem a graph problem?

- What problem are you trying to solve?
 - Selection / search
 - These sorts of questions do not require rich relationships within the data. In most databases, answering these questions requires using a single filtering criterion or, potentially, an index. While you can answer these with a graph database, these problems do not use or require graph-specific functionality.

Ask yourselves

Is my problem a graph problem?

- What problem are you trying to solve?
 - Selection / search
 - Related or recursive data

Ask yourselves

Is my problem a graph problem?

- What problem are you trying to solve?
 - Selection / search
 - Related or recursive data
 - What's the easiest way for me to be introduced to an executive at X?
 - How do John and Paula know each other?
 - How's company X related to company Y?

Ask yourselves

Is my problem a graph problem?

- What problem are you trying to solve?
 - Selection / search
 - Related or recursive data
 - Graph databases leverage this information better than any other type of data engine, and their query languages are better suited to reasoning over the relationships within the data.

Ask yourselves

Is my problem a graph problem?

- What problem are you trying to solve?
 - Selection / search
 - Related or recursive data
- Aggregation
 - How many companies are in my system?
 - What are my average sales for each day over the past month?
 - What's the number of transactions processed by my system each day?

Ask yourselves

Is my problem a graph problem?

- What problem are you trying to solve?
 - Selection / search
 - Related or recursive data
 - Aggregation
 - These same sorts of queries can be performed in graph databases, but the nature of graph traversals requires that much more of the data is touched.

Ask yourselves

Is my problem a graph problem?

- What problem are you trying to solve?
 - Selection / search
 - Related or recursive data
 - Aggregation
- Pattern matching, influence
 - Who in my system has a similar profile to me?
 - Does this transaction look like other known fraudulent transactions?
 - Is the user J. Smith the same as Johan S.?

Ask yourselves

Is my problem a graph problem?

- What problem are you trying to solve?
 - Selection / search
 - Related or recursive data
 - Aggregation
 - Pattern matching, influence
 - Pattern-matching use cases are so commonly done in graph databases that graph query languages have specific, built-in features to handle precisely these sorts of queries.

Ask yourselves

Is my problem a graph problem?

- What problem are you trying to solve?
 - Selection / search
 - Related or recursive data
 - Aggregation
 - Pattern matching, influence

Ask yourselves

Is my problem a graph problem?

- Do you care about the relationships between entities as much or more than the entities themselves?
- Does my sql query perform multiple joins on the same table or require a recursive CTE?
- Is the structure of my data continuously evolving?
- Is my domain a natural fit for a graph?

Querying a graph

In a relational database you use a query to answer the question. But in a graph, you perform a ***traversal*** (imperative querying) or ***pattern matching*** (declarative querying).

Querying a graph

Imperative form

Traversal is the process of moving through the graph defined as the set of steps and actions you perform to retrieve data. Thus querying in a graph database may focus on how to traverse from one element to another.

Get the names of the creatures that the creature represented by vertex with the unique identifier of 1 classify as enemy:

```
g.V(1).outE('enemy').inV().values('name')
```

Querying a graph

Declarative form

Another possibility is querying by specifying a **pattern** like for example

```
(c:Customer {name:'C3PO'})-[rel:ORDERED]->(i:Item)
```

where you match items ordered by customers.

As any **declarative language**, it allows you to state what you want to do with your graph data (select, insert, update or delete) without requiring you to describe exactly how to do it.

Bibliography

- [Bec] Dave Bechberger, Josh Perryman, *Graph Databases in Action*, Manning Publications, 2020
- [Ful] Piotr Fulmański, *NoSQL. Theory and examples*, Piotr Fulmański, 2021