It's all about me…

Prof. Mark Whitehorn
Emeritus Professor of Analytics
Computing
University of Dundee

Consultant
Writer (author)
m.a.f.whitehorn@dundee.ac.uk

Graph Databases

- Different database engines are built to be good at a specific set of operations.

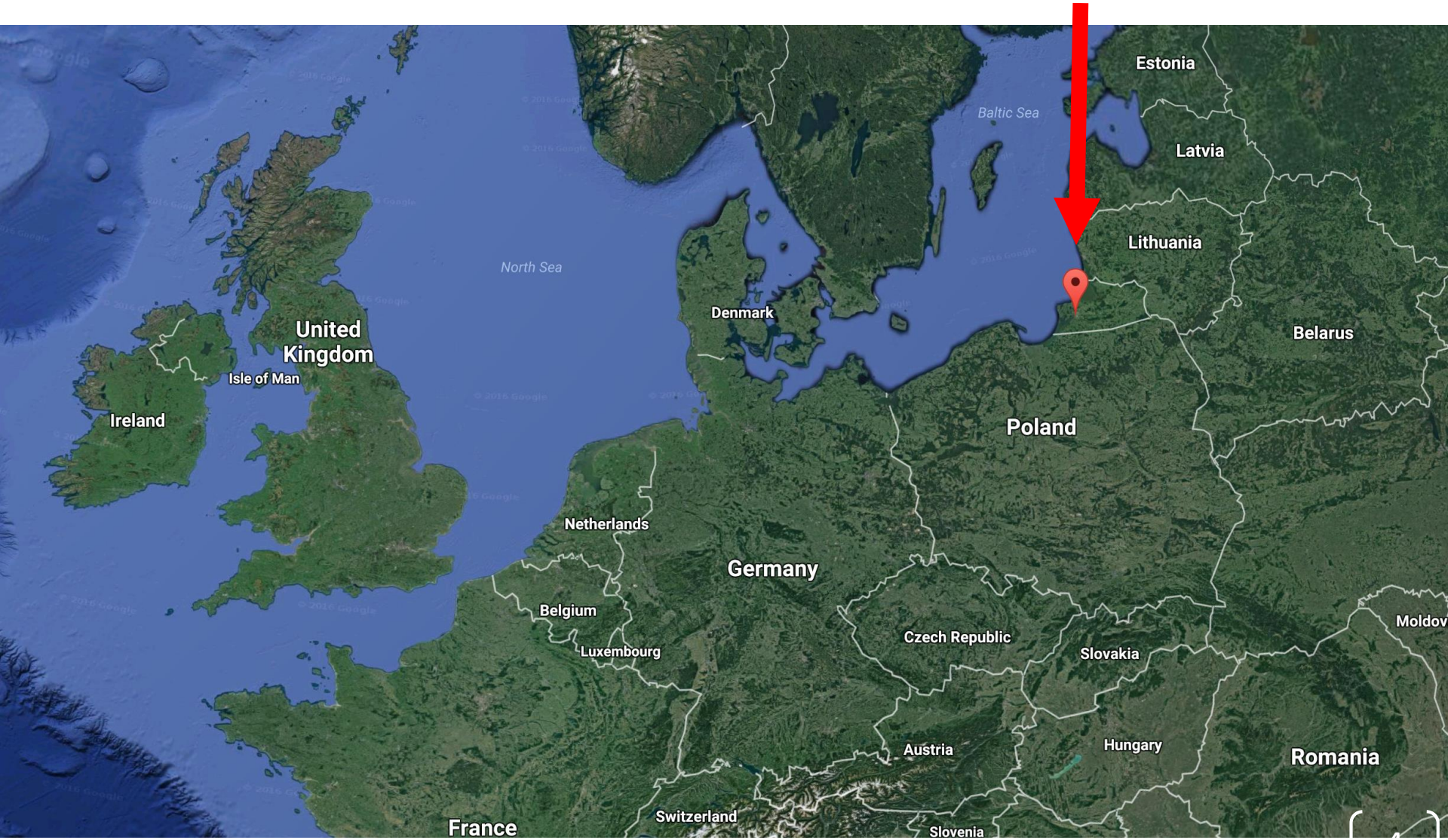
  Relational engines, for example, are typically optimised for transaction control and protecting data from damage and loss during update.

  They are typically not optimised for detecting fraud and performing recommendations ("Customers who bought this book frequently bought…."). Graph databases are essentially the opposite, poor at transactions and good at tasks such as fraud detection and recommendations. The key to using graph databases effectively is understanding not only how they work but why they were designed that way – in other words, understanding what underpins their strengths and weaknesses. So this talk will explore their origins and how and why they work.

Graph Databases

- Origins

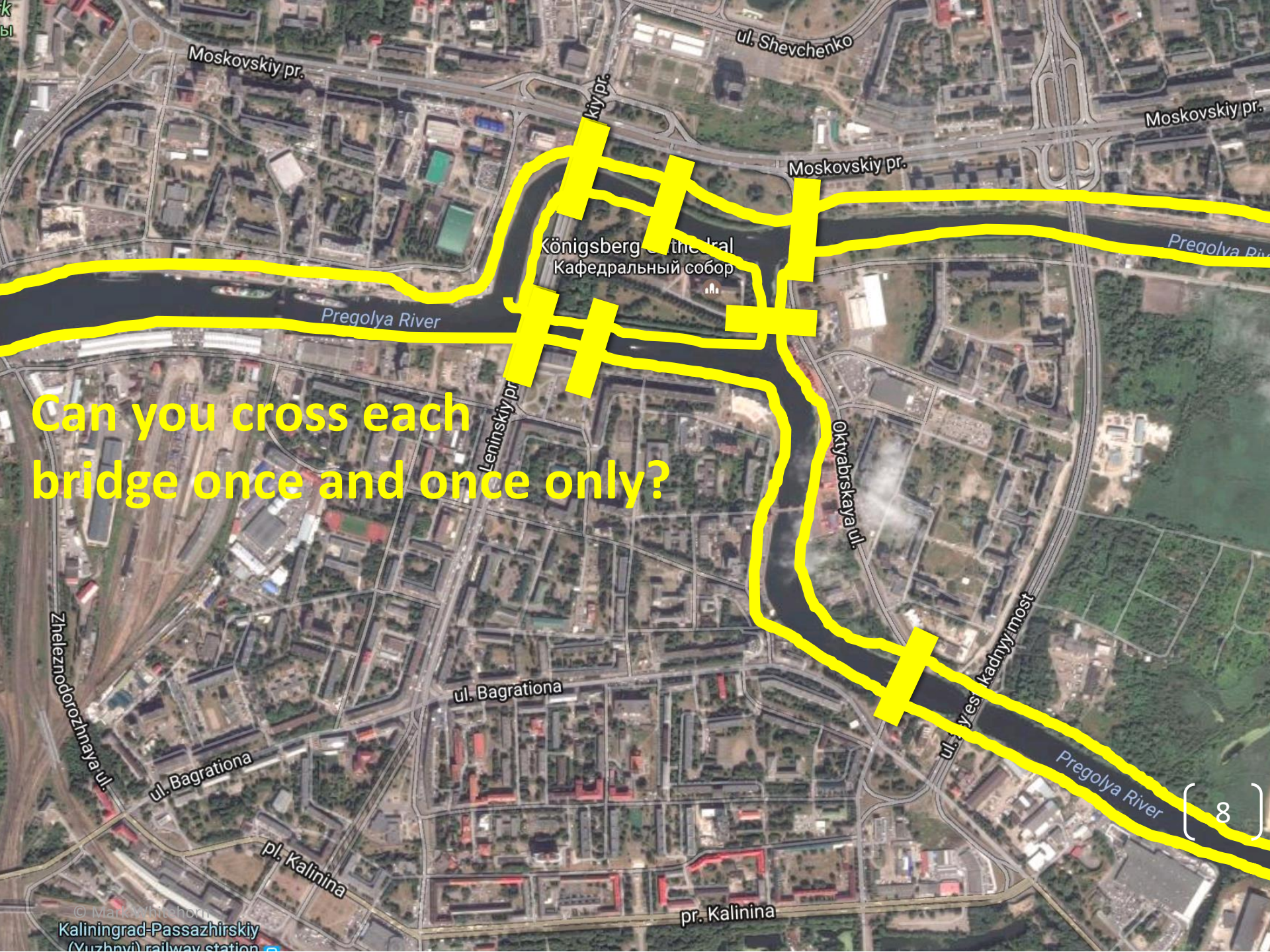
Kaliningrad, the city formerly known as Prince; errr, **Königsberg**.

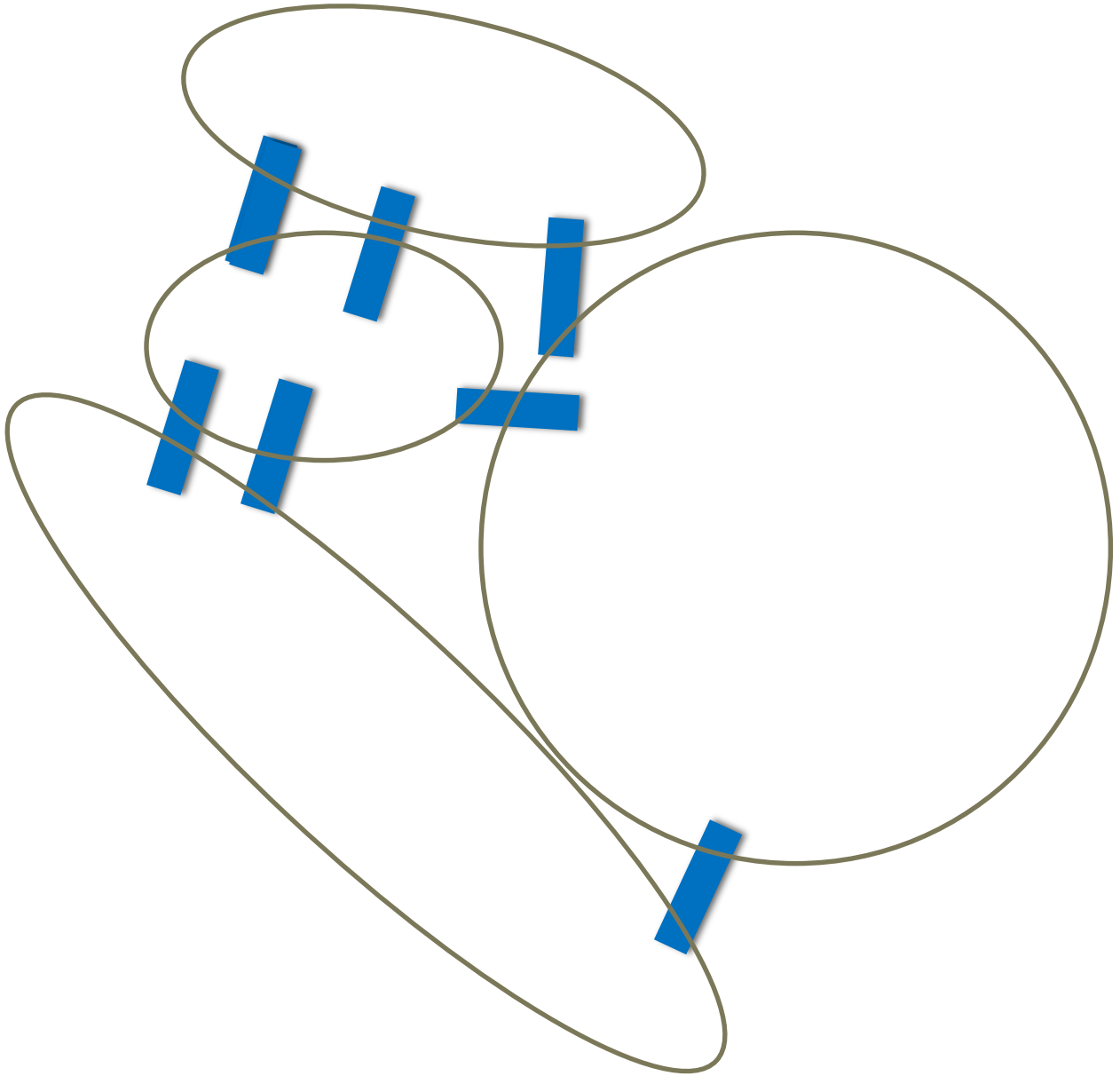


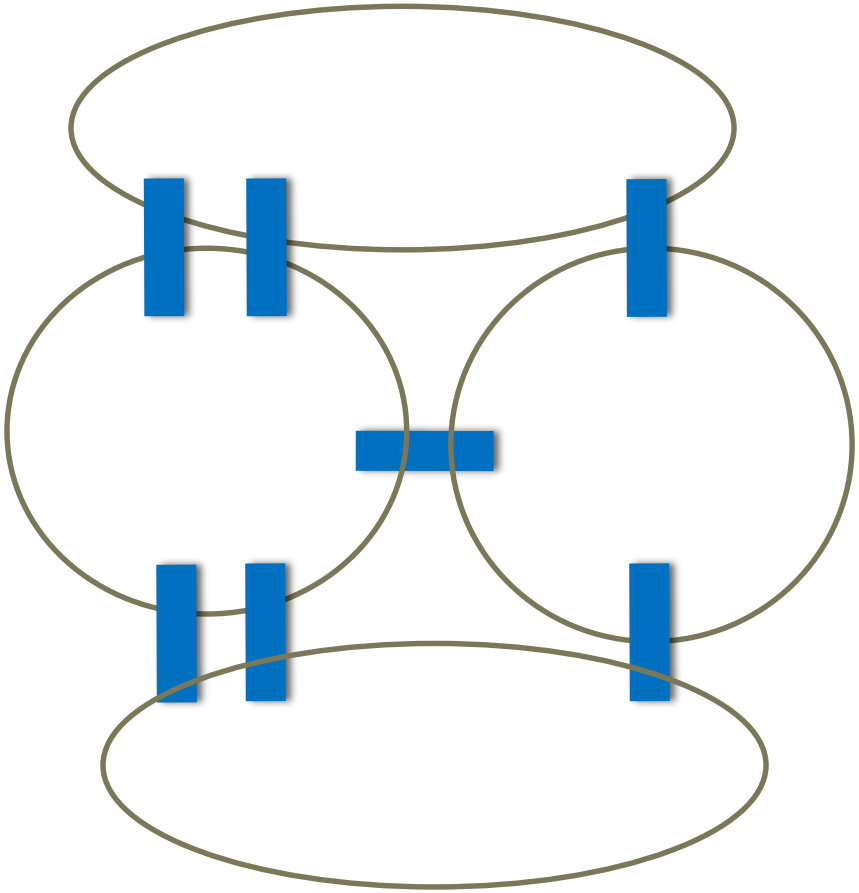
In the 1700s, Königsberg had seven bridges.

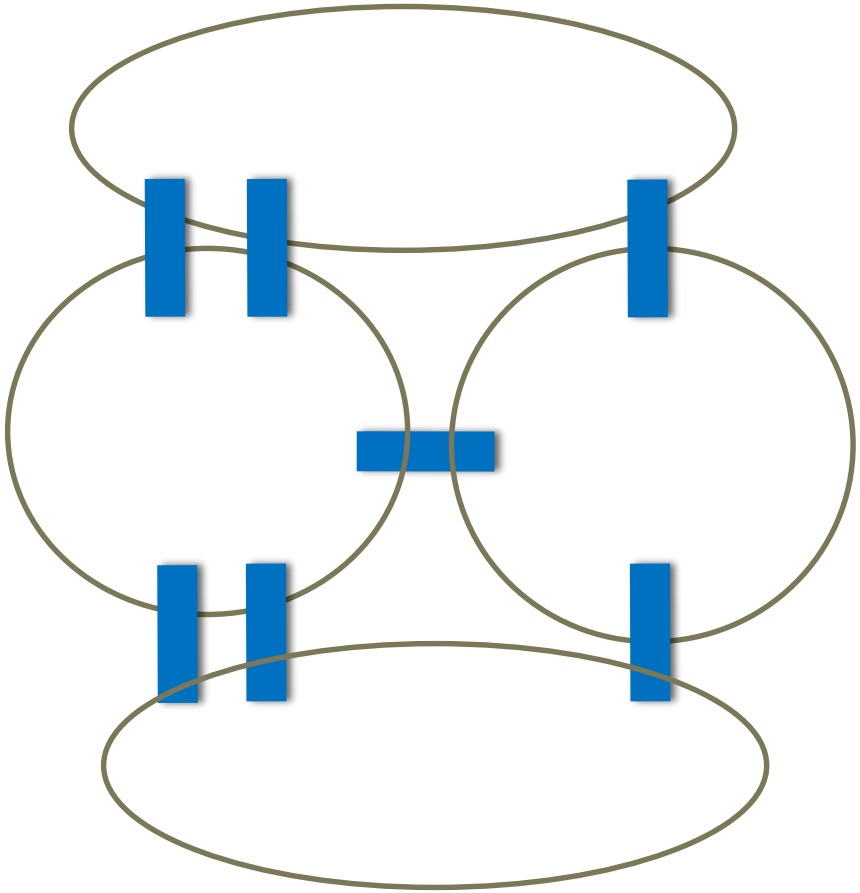

Moskovskiy pr.
ul. Shevchenko
Moskovskiy pr.
Moskovskiy pr.
Königsberg Cathedral
Кафедральный собор
Pregolya River
Pregolya River
Leninskiy pr
Oktyabrskaya ul.
Can you cross each bridge once and once only?
Zheleznodorozhnaya ul.
ul. Bagrationa
ul. Bagrationa
ul. 2-yeSestakadnyy most
Pregolya River
pl. Kalinina
pr. Kalinina
Kaliningrad-Passazhirskiy

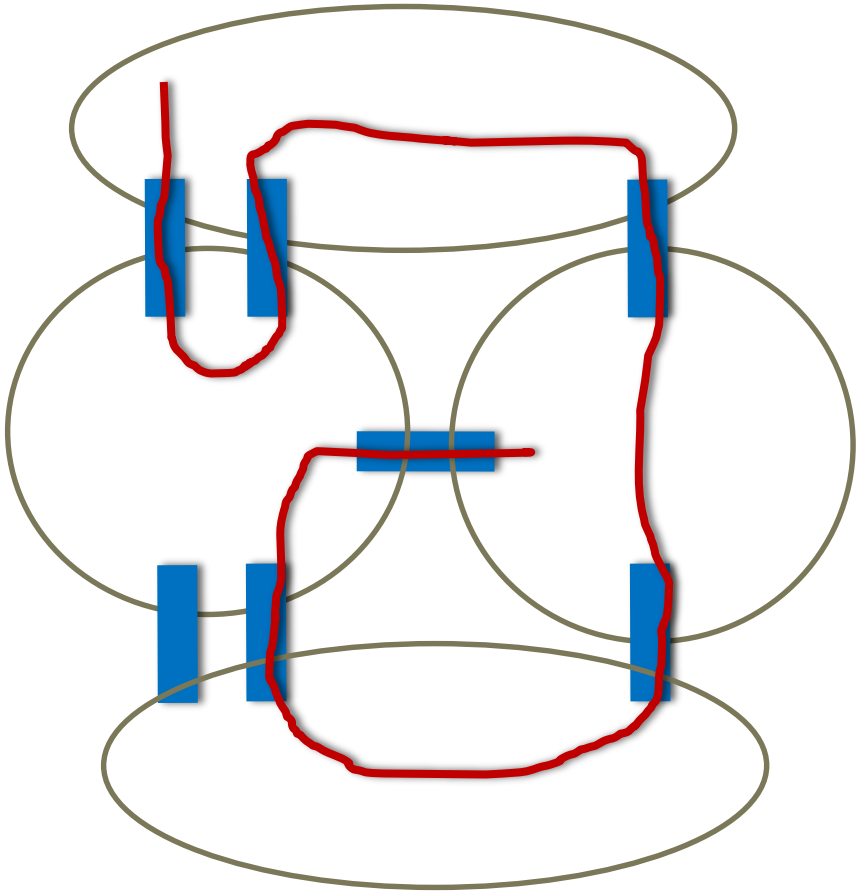
Solved by **Leonhard Euler** (1707 – 1783) Swiss mathematician.


Moskovskiy pr.
ul. Shevchenko
Moskovskiy pr.
Moskovskiy pr.
Königsberg Cathedral
Кафедральный собор
Pregolya River
Pregolya River
Leninskiy pr
Oktyabrskaya ul.
Can you cross each bridge once and once only?
Zheleznodorozhnaya ul.
ul. Bagrationa
ul. Bagrationa
pl. Kalinina
pr. Kalinina
Pregolya River
Kaliningrad-Passazhirskiy (Yuzhnyi) railway station
8

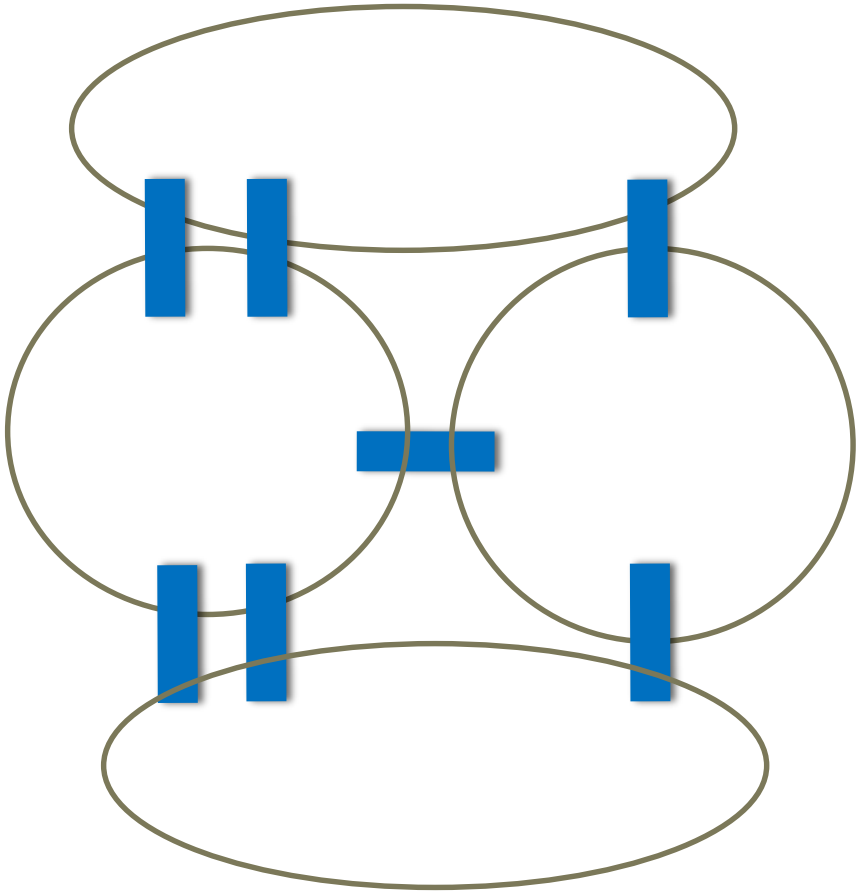


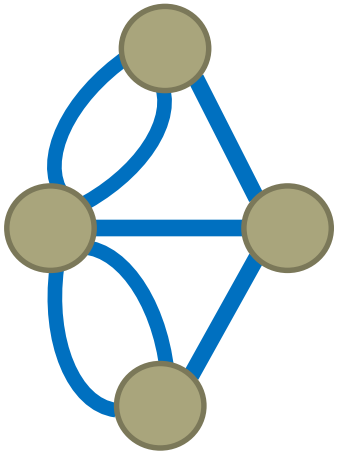


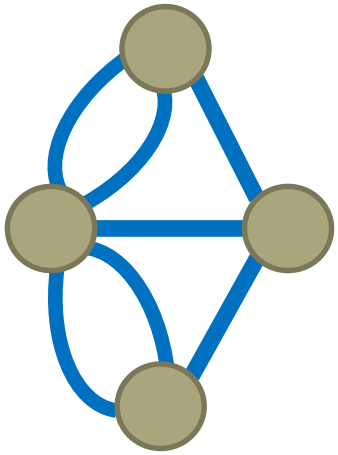


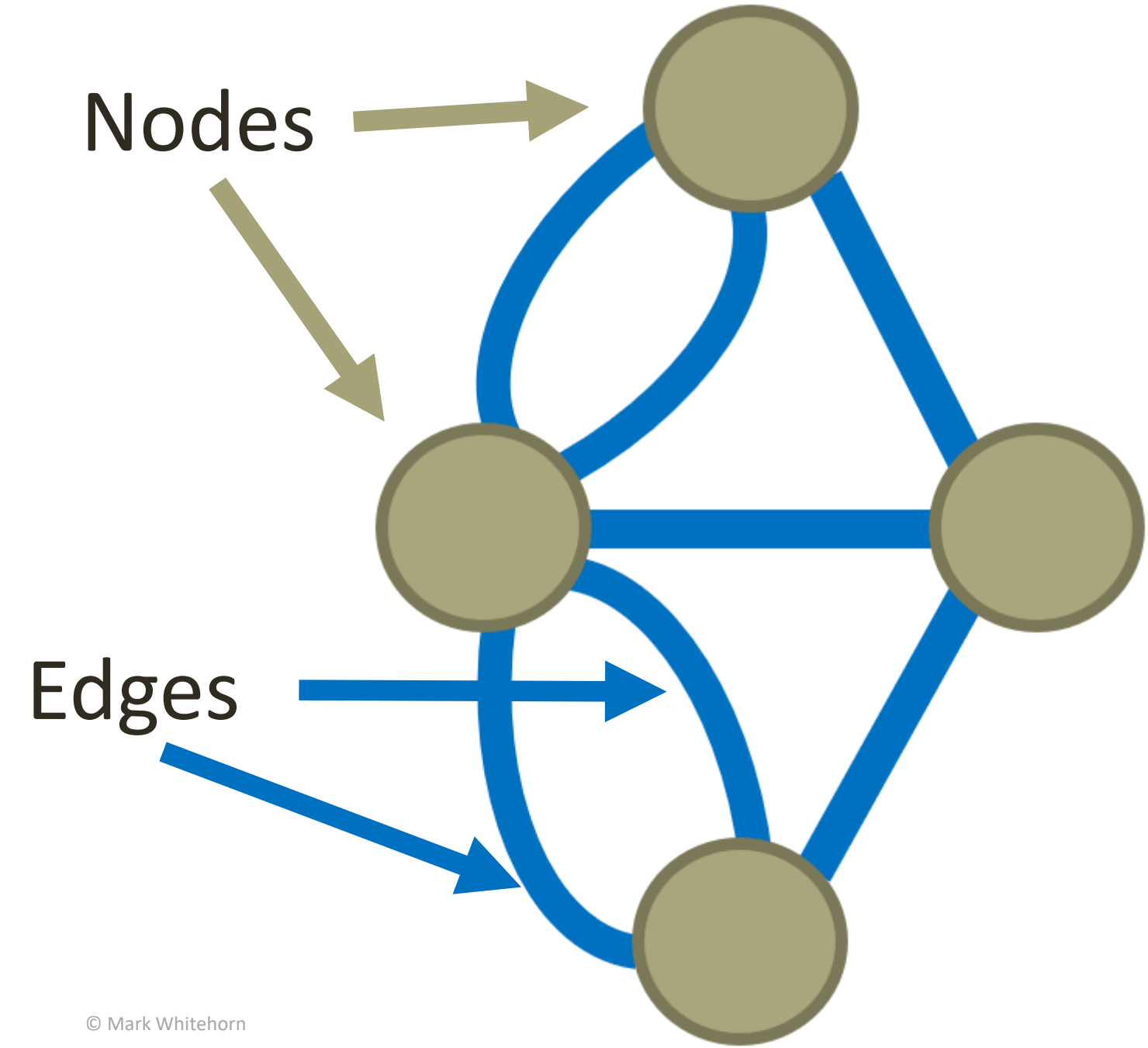


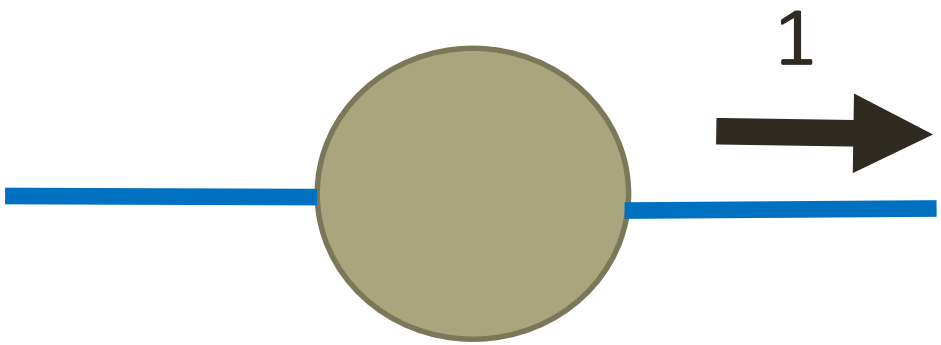
We can try (and fail) to trace a path but failing doesn't prove that it can’t be done. If we succeed we prove it can be done but no one could succeed.



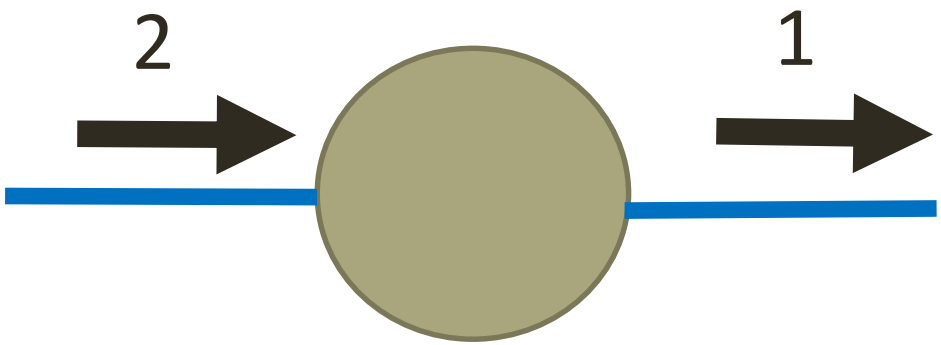




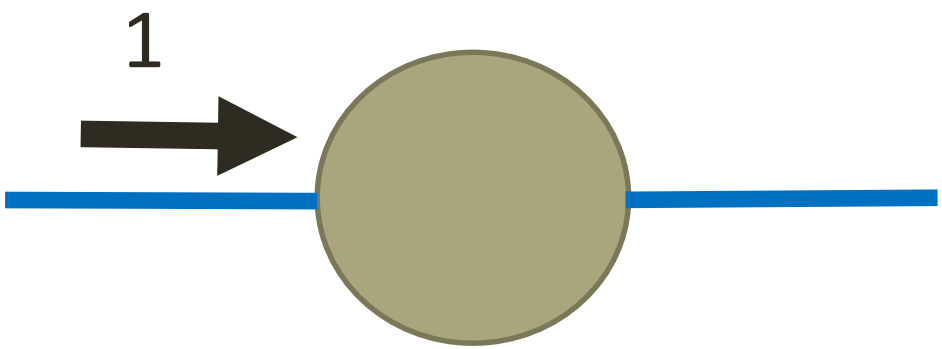




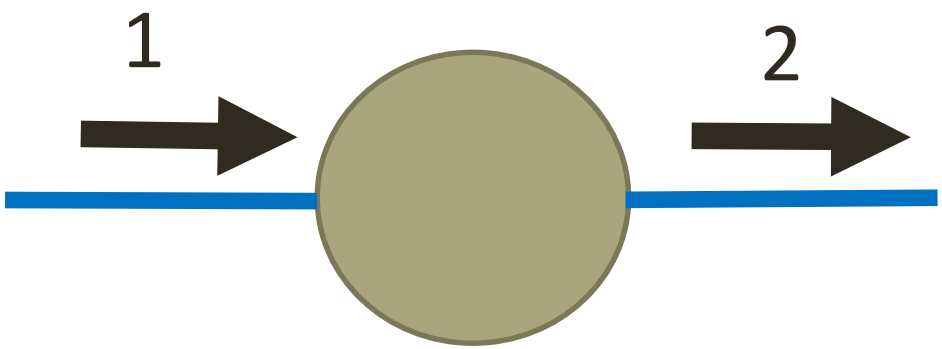
Consider a node with two edges.
If you start on the node you must finish ….



Consider a node with two edges.
If you start on the node you must finish on the node;
This remains true no matter to what the edges connect.

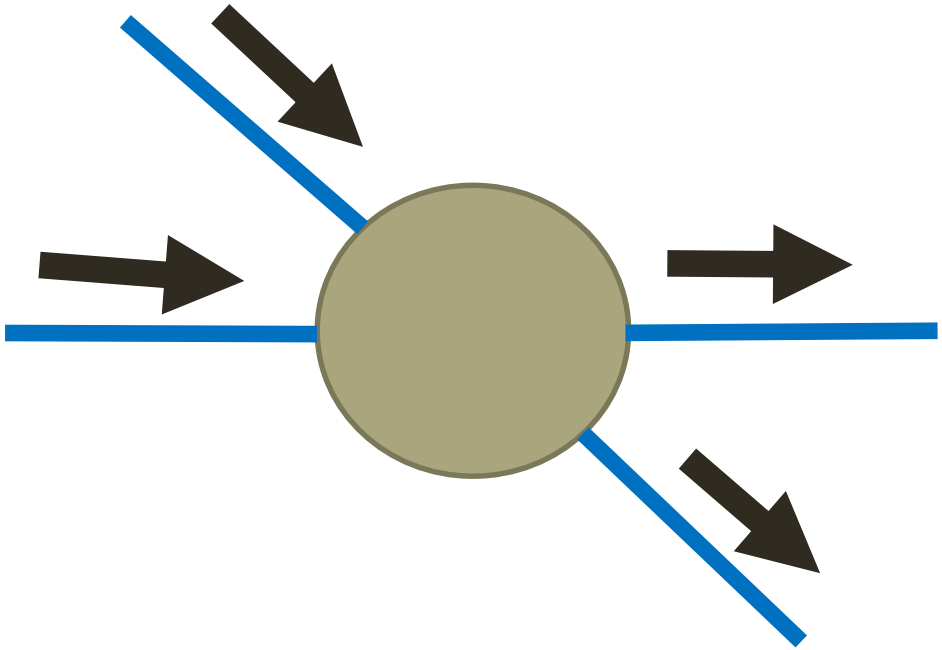


Consider a node with two edges.
If you start off the node you must finish ….

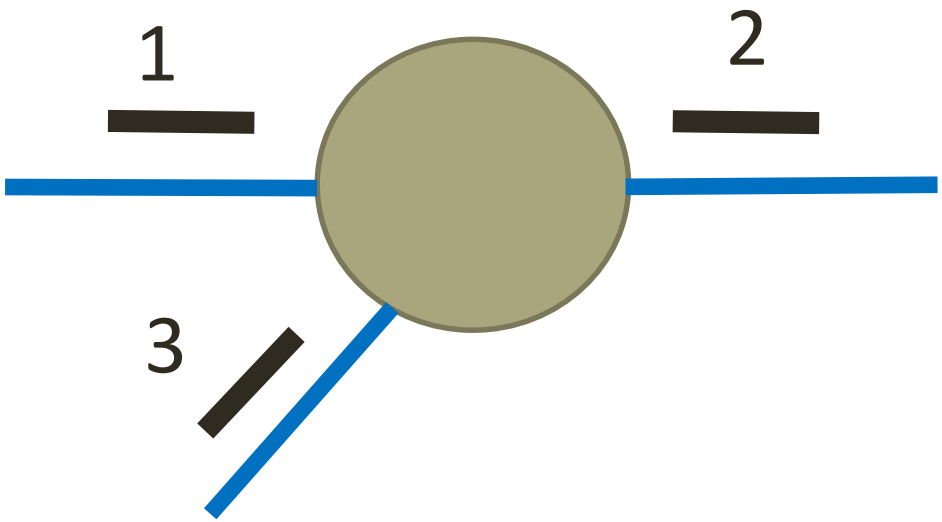


Consider a node with two edges.
If you start off the node you must finish off the node;
This remains true no matter to what the edges connect.

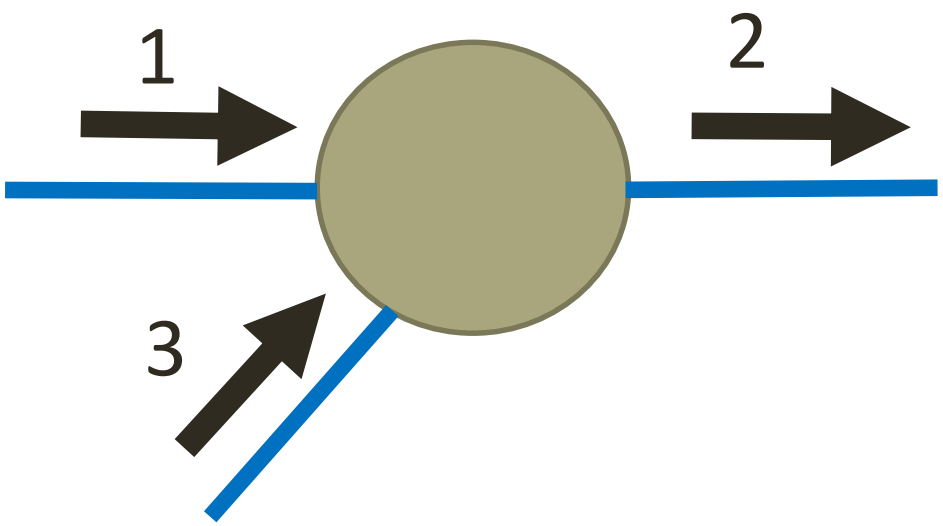
What further generalisation can we make?



Consider a node with an even number of edges.
If you start on the node you must finish on the node;
If you start off the node you must finish off the node.
This remains true no matter to what the edges connect.

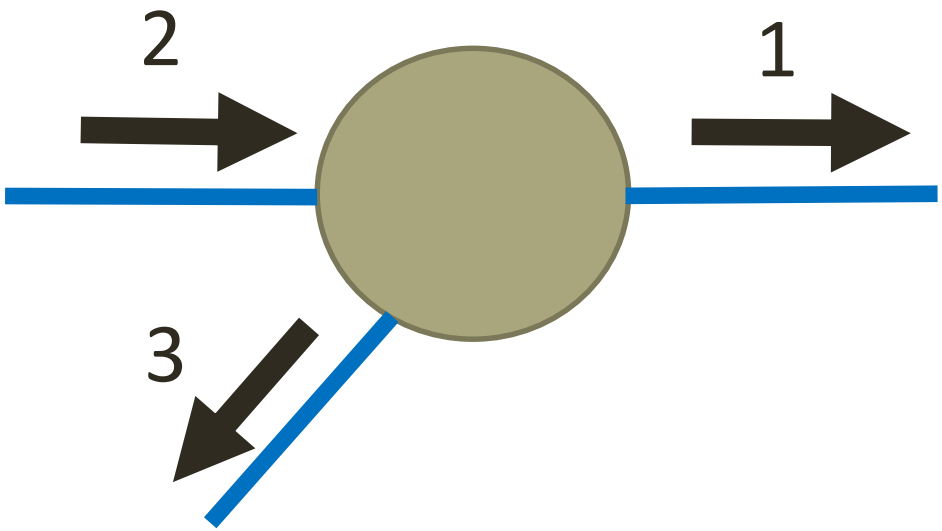


Consider a node with three edges.
If you start off the node you must finish ? the node;
if you start on the node you must finish ? the node.



Consider a node with three edges.
If you start off the node you must finish on the node.

This remains true no matter to what the edges connect.

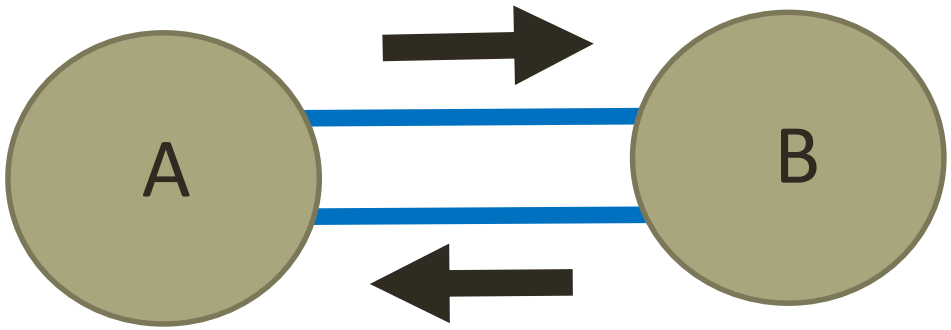


Consider a node with three edges.
If you start on the node you must finish off the node.

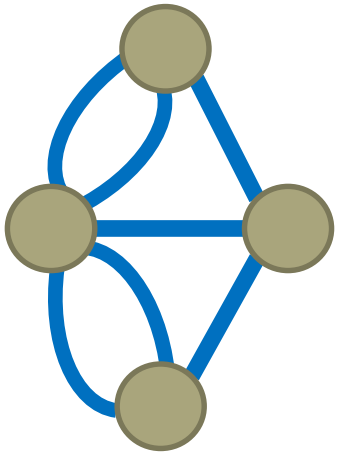
This remains true no matter to what the edges connect.
And this is true for all nodes with an odd number of edges.

Se we have a set of rules that logic (or Euler) tells us is irrefutable.
The table shows us where you must **finish** for a given set of starting conditions:

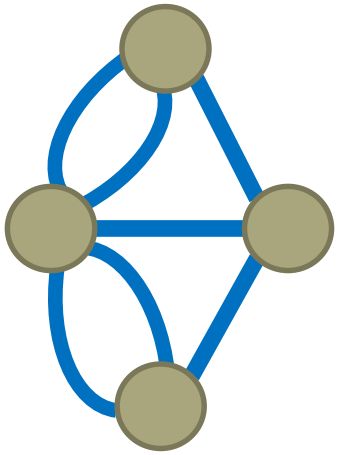
	Even no. of edges	Odd no. of edges
Start on node	**On**	**Off**
Start off node	**Off**	**On**



Suppose we have two nodes, both having two edges.
We can start on and finish on node A,
which agrees with the rules.
We can start off and finish off node B,
which also agrees with the rules.



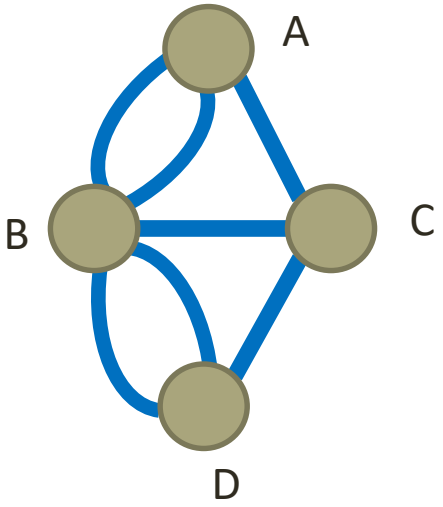
So, in the Königsberg bridge problem, a really important general question to ask is:
“How many nodes have an even number of edges and how many have an odd number?”



There are four nodes and they all have an odd number of edges.

What do we know about nodes with odd numbers of edges?

If you don’t start on a node, you must finish on that node.



Suppose we choose to start on node A, that means we don't start on B C or D. But the rules tell us that, if we don't start on a node we have to finish on it. So we have to finish on three nodes (B C D).

That is impossible, so the Königsberg bridge problem is unsolvable.

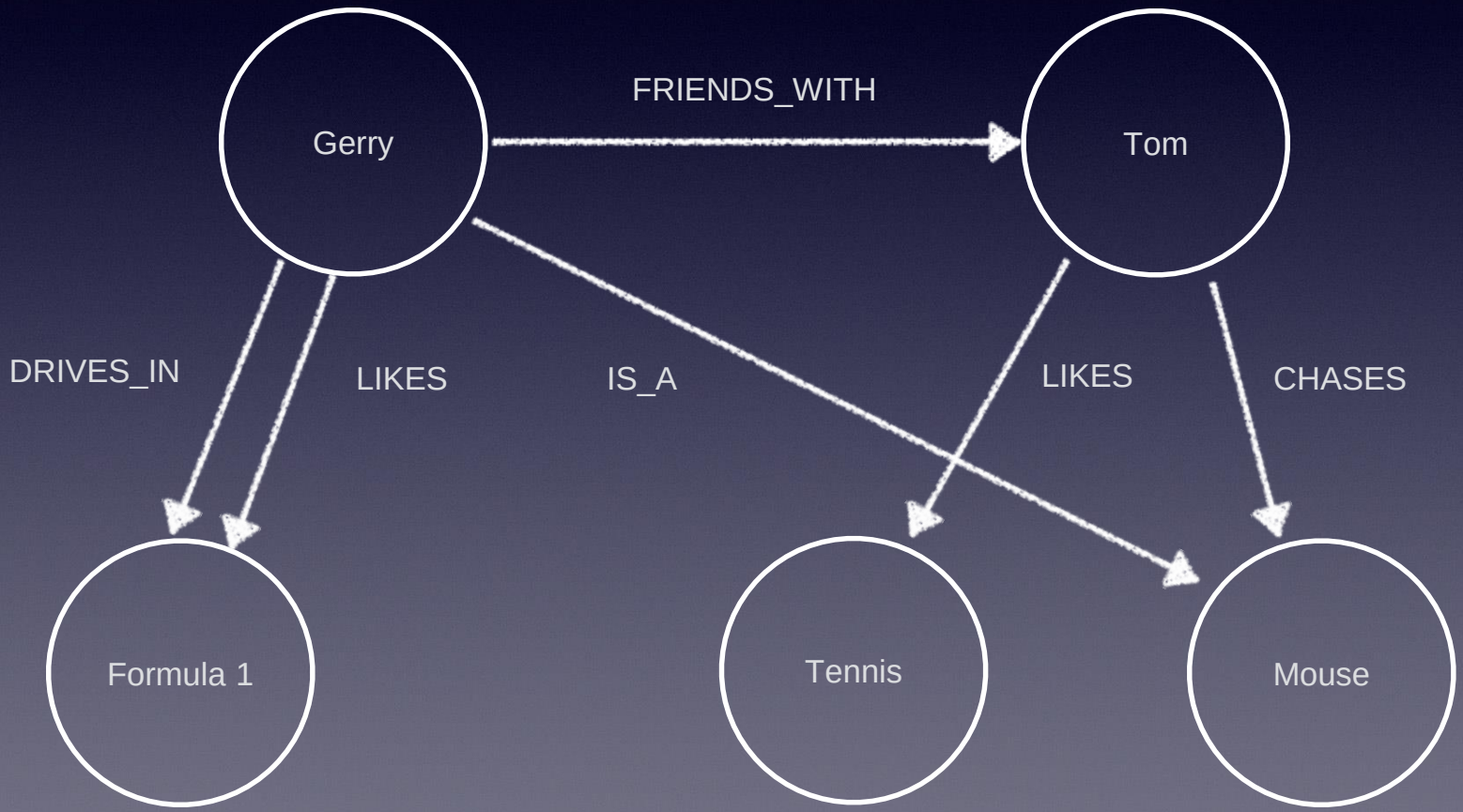
Not only did Euler induce the general rules, he developed an entire branch of mathematics from this - Graph. In turn this led to the development of graph databases which are a very important class of NoSQL database engines.

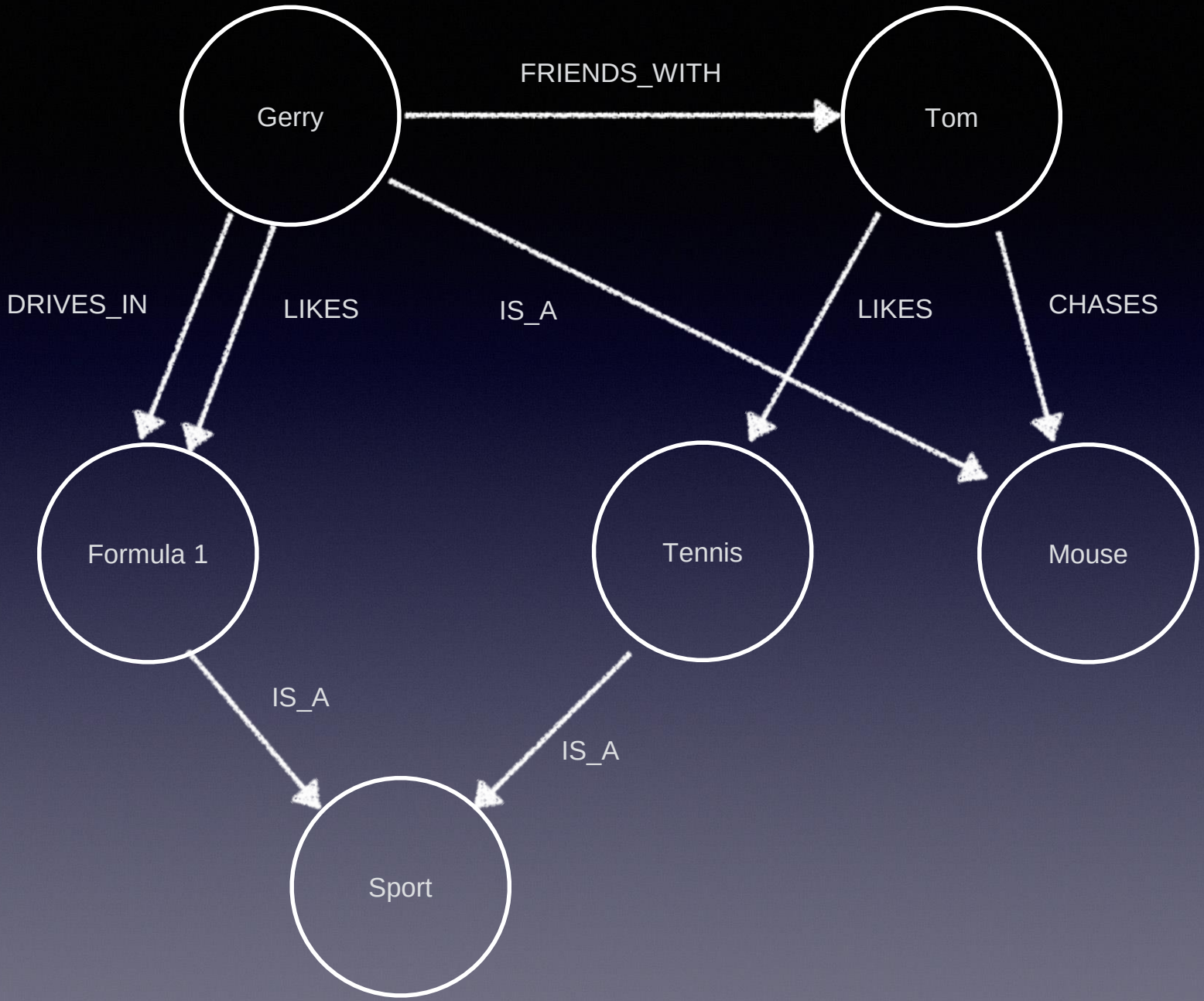
What is a Graph Database?

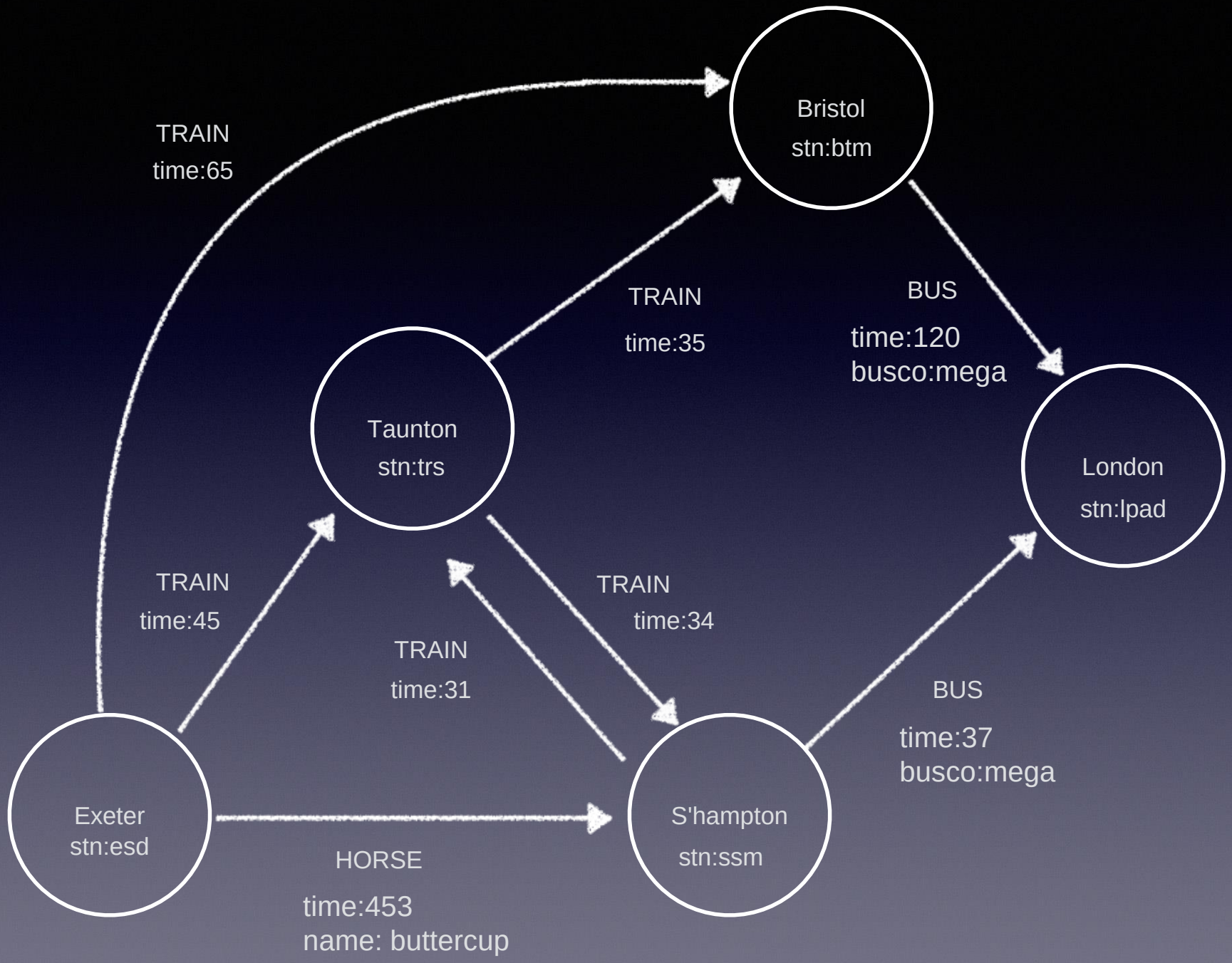
Slides courtesy of Gerry McNicol

@gerrymcnicol

What is a Graph?

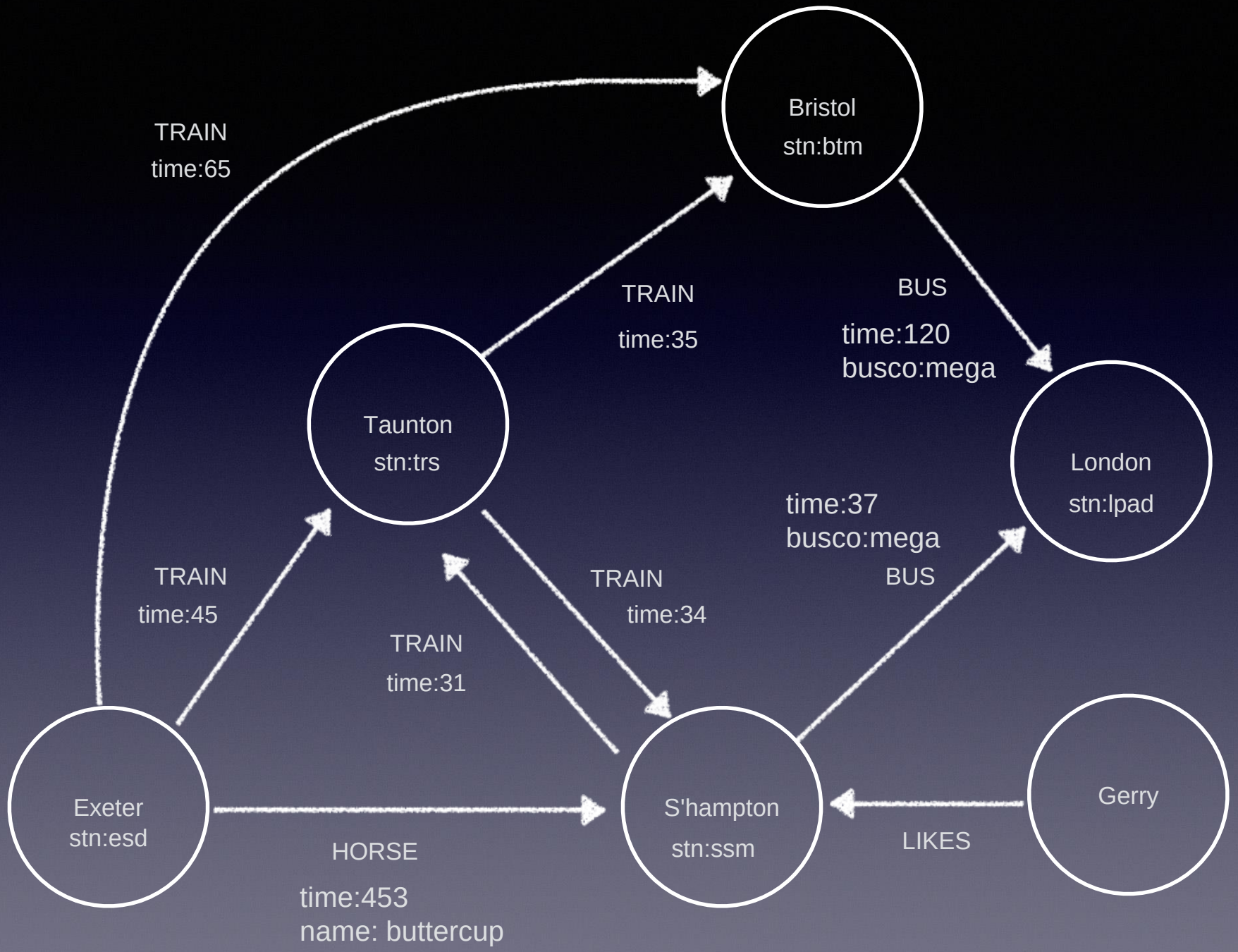


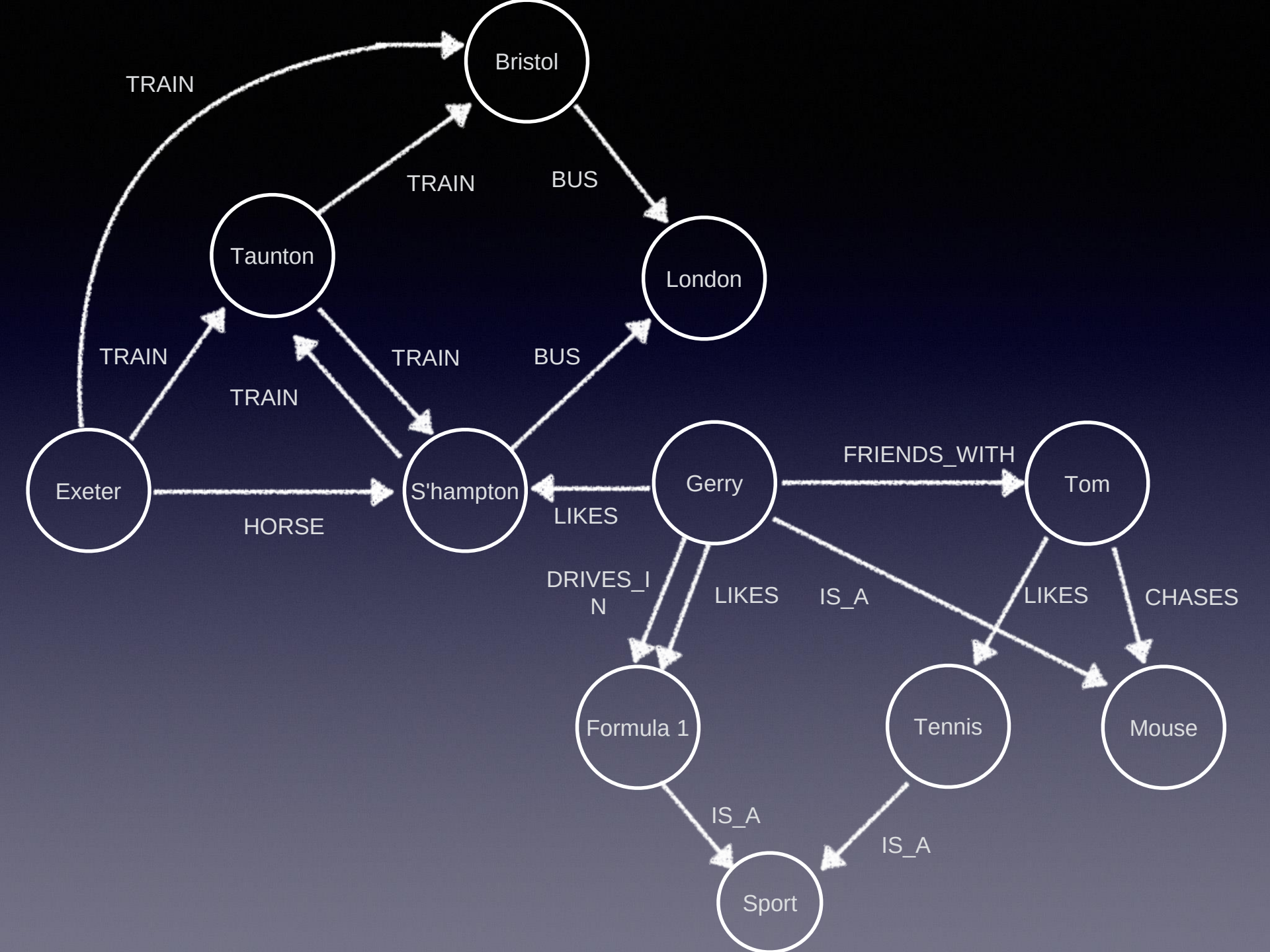

Gerry
FRIENDS_WITH
Tom
DRIVES_IN
LIKES
IS_A
LIKES
CHASES
Formula 1
Tennis
Mouse
IS_A
IS_A
Sport


Bristol
stn:btm
TRAIN
time:65
TRAIN
time:35
BUS
time:120
busco:mega
Taunton
stn:trs
London
stn:lpad
TRAIN
time:45
TRAIN
time:34
TRAIN
time:31
BUS
time:37
busco:mega
Exeter
stn:esd
S'hampton
stn:ssm
HORSE
time:453
name: buttercup

What is a Graph?

- Made up of Nodes and Edges (Relationships)
- Nodes are connected by Edges
- Every Edge has ...
  - a starting and ending Node
  - a direction
- Both Nodes and Edges can have properties.
- Very flexible data structure


Bristol
stn:btm
TRAIN
time:65
TRAIN
time:35
BUS
time:120
busco:mega
Taunton
stn:trs
London
stn:lpad
time:37
busco:mega
BUS
TRAIN
time:45
TRAIN
time:34
TRAIN
time:31
Exeter
stn:esd
S'hampton
stn:ssm
Gerry
HORSE
time:453
name: buttercup
LIKES


Bristol
TRAIN
TRAIN
BUS
Taunton
London
TRAIN
TRAIN
TRAIN
BUS
Exeter
S'hampton
HORSE
LIKES
Gerry
FRIENDS_WITH
Tom
DRIVES_IN
LIKES
IS_A
LIKES
CHASES
Formula 1
Tennis
Mouse
IS_A
IS_A
Sport

Use Cases

- Very powerful and flexible data model
- Semantically rich - very descriptive
- Densely-connected data sets
- Variably Structured data sets

Graph – Database engines

Clearly there are multiple graph engines and they can differ. However we can talk in generalisations that will apply to most.

The data is stored in both nodes and edges

- Both are equally important

There is no need for nodes (or edges) to store the same data

Graph – Database engines

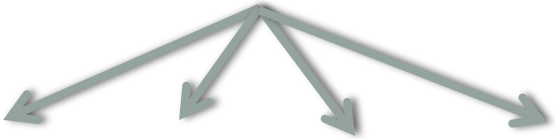
- Data is typically stored as key value pairs (KVPs)

KVPs?

Going back to relational data for a moment

Columns

Car



Rows

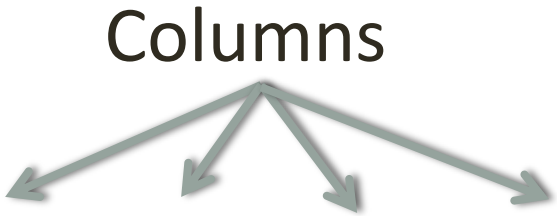
LicenceNo	Make	Model	Year	Colour
CER 162 C	Triumph	Spitfire	1965	Green
EF 8972	Bentley	Mk. VI	1946	Black
YSK 114	Bentley	Mk. VI	1949	Red

All entities have the same set of attributes, and only one of each.

In practical terms we could also say that each row will have data for each column.

Going back to relational data for a moment

Car



Rows

LicenceNo	Make	Model	Year	Colour
CER 162 C	Triumph		1965	Green
EF 8972	Bentley	Mk. VI	1946	Black
YSK 114	Bentley	Mk. VI	1949	**Blue/Red**

Nulls are tolerated, but frowned upon:

- All cars should have a model

Duplicated are not tolerated:

- A car cannot have more than one colour

Nulls can be common in big data

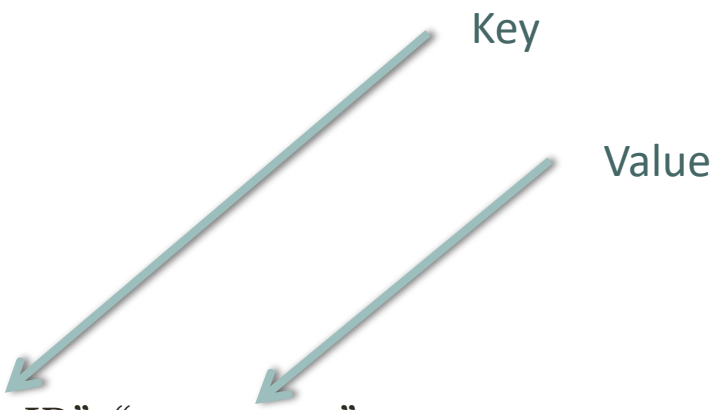
SensorID	Manufacturer	TimeDate	Pressure	Humidity	Temp	Wind	Depth	And so on
213342332	34	1/1/2016:11:23			23			
2-BSDEFF76	12	2016/1/1:11:34	1034				12	

This particular data (sensor data) sits poorly in a table. But note that each reading can be identified by the column name and the row identifier.

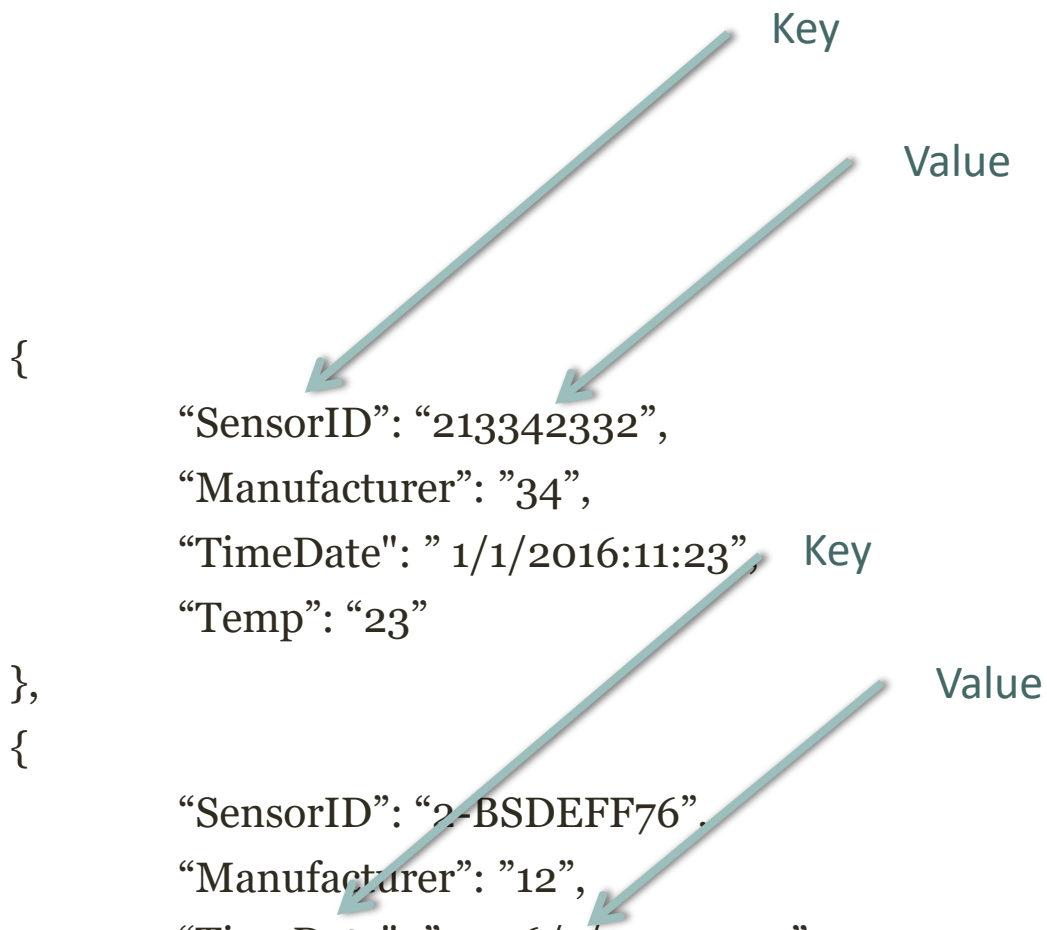
So we could store, for each row, only the columns that do have data.

SensorID	Manufacturer	TimeDate	Pressure	Humidity	Temp	Wind	Depth	And so on
213342332	34	1/1/2016:11:23			23			
2-BSDEFF76	12	2016/1/1:11:34	1034				12	

{
“SensorID”: “213342332”,
“Manufacturer”: ”34”,
“TimeDate": ” 1/1/2016:11:23”,
“Temp”: “23”
},
{
“SensorID”: “2-BSDEFF76”,
“Manufacturer”: ”12”,
“TimeDate": ” 2016/1/1:11:34:43”,
“Pressure”: “1034”,
“Depth”: “12”
}




```
{
        “SensorID”: “213342332”,
        “Manufacturer”: ”34”,
        “TimeDate": ” 1/1/2016:11:23”,
        “Temp”: “23”
},
{
        “SensorID”: “2-BSDEFF76”,
        “Manufacturer”: ”12”,
        “TimeDate": ” 2016/1/1:11:34:43”,
        “Pressure”: “1034”,
        “Depth”: “12”
}
```


Key
Value
{
“SensorID”: “213342332”,
“Manufacturer”: ”34”,
“TimeDate": ” 1/1/2016:11:23”,
“Temp”: “23”
},
Key
Value
{
“SensorID”: “2-BSDEFF76”,
“Manufacturer”: ”12”,
“TimeDate": ” 2016/1/1:11:34:43”,
“Pressure”: “1034”,
“Depth”: “12”
}

Key Value Pairs

Key Value Pairs (KVPs) are a very effective way of storing sparse data (data where we expect a large number of nulls).

They are also excellent in cases where we know the data collected will vary over time.

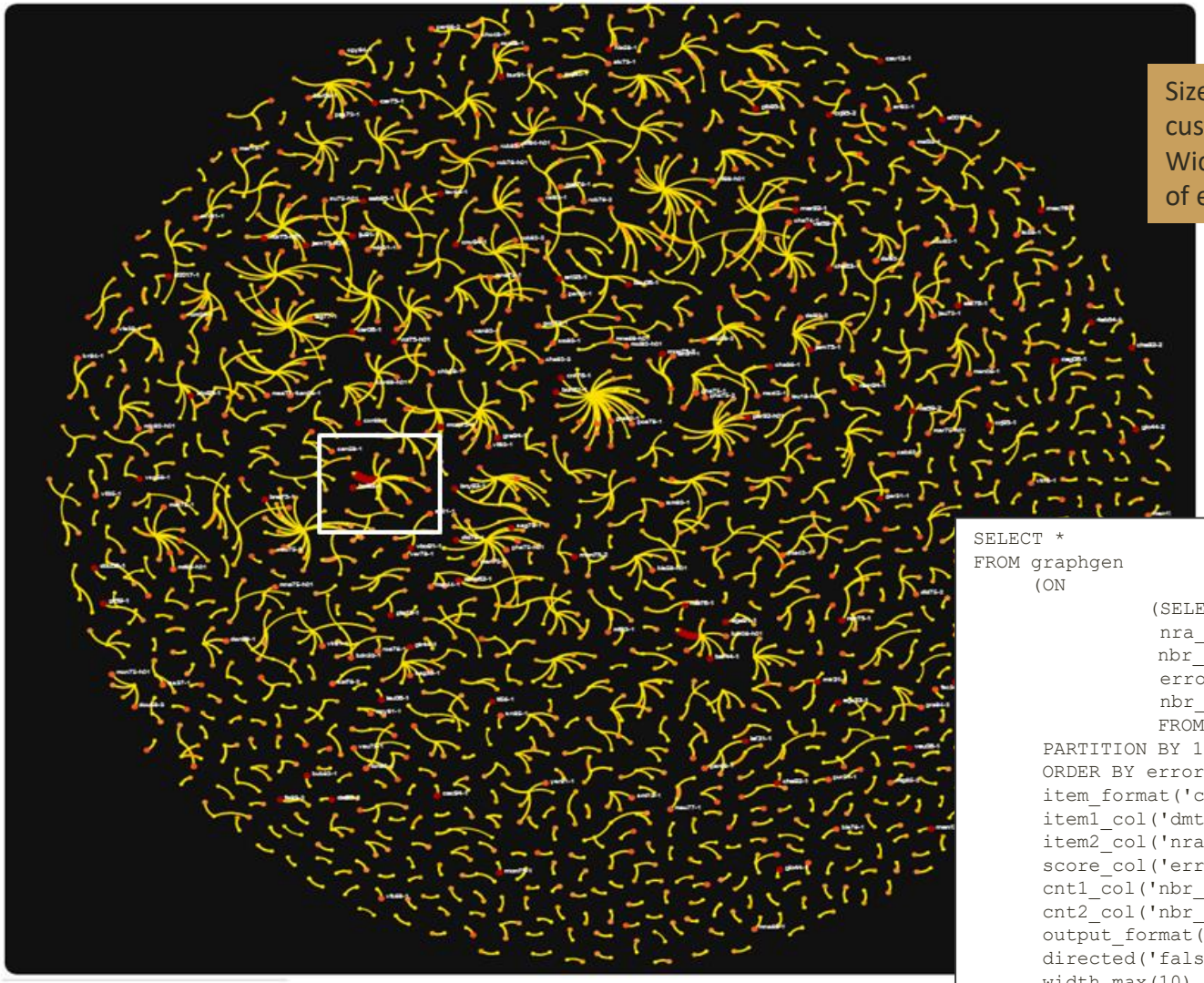
Graph – Database design

Where should we put an attribute such as “occupation”, e.g. Data Scientist?

Three options:

- In the person node
- in an edge (hard!)
- In an occupation node

Visualise as a Graph



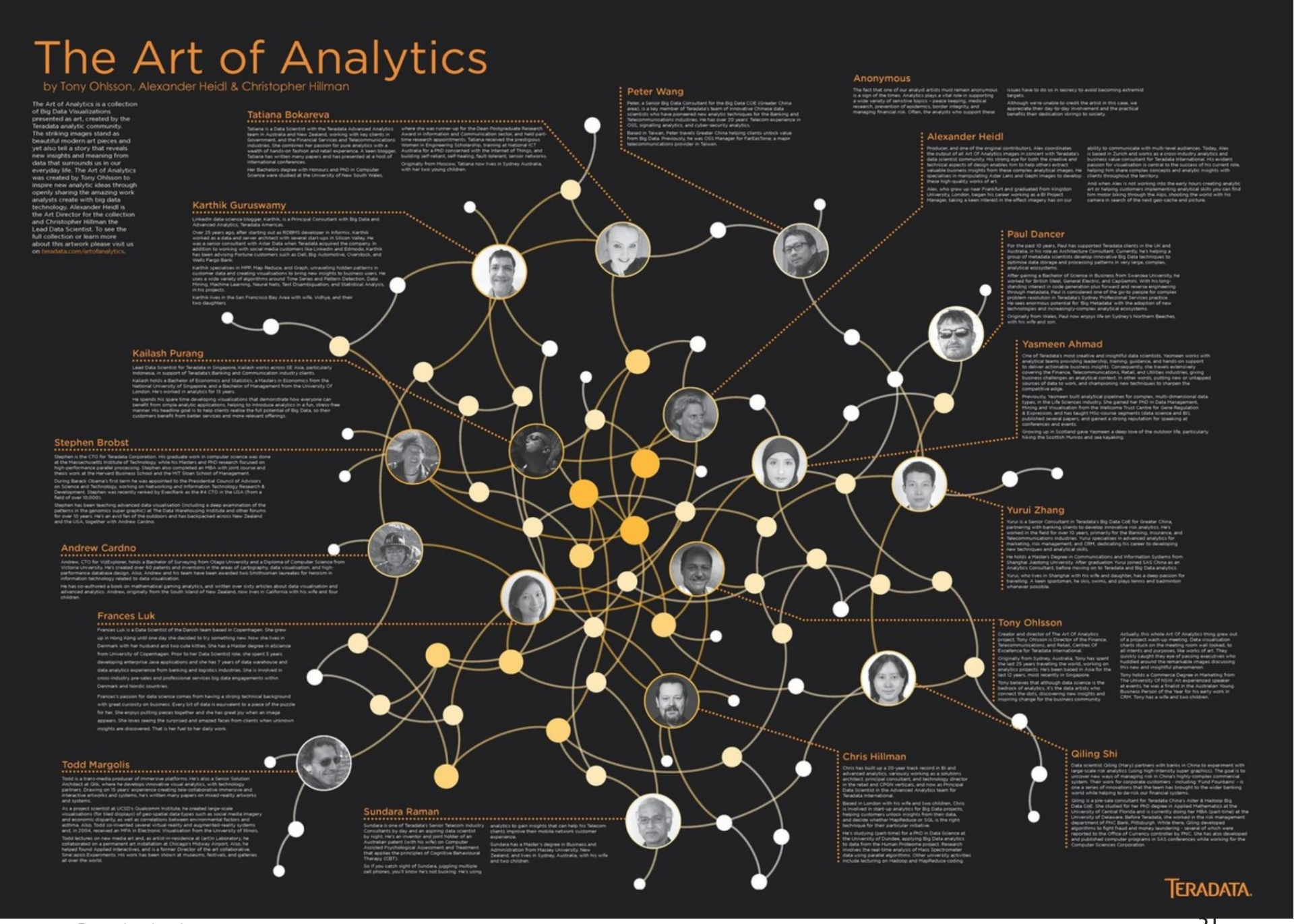
Size of Node = number of customers
Width of Edge = number of errors

```
SELECT *
FROM graphgen
     (ON
               (SELECT DISTINCT dmt_act_dslam,
                nra_id,
                nbr_of_srvid,
                errorspersrv,
                nbr_of_dslam
                FROM wrk.srvid_dslam_err)
      PARTITION BY 1
      ORDER BY errorspersrv
      item_format('cfilter')
      item1_col('dmt_act_dslam')
      item2_col('nra_id')
      score_col('errorspersrv')
      cnt1_col('nbr_of_srvid')
      cnt2_col('nbr_of_dslam')
      output_format('sigma')
      directed('false')
      width_max(10)
      width_min(1)
      nodesize_max (3)
      nodesize_min (1));
```

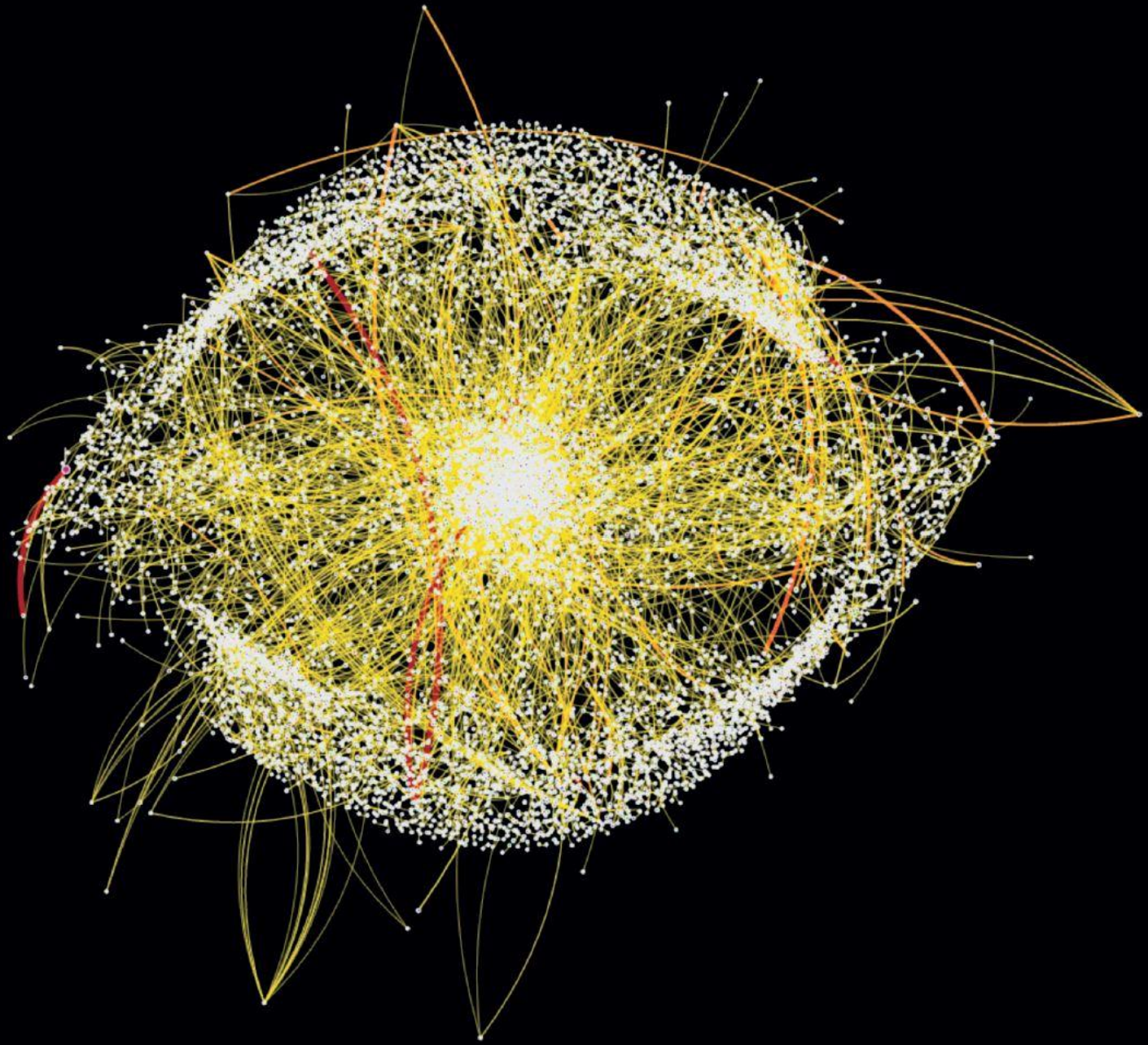
ART OF ANALYTICS

Chris Hillman

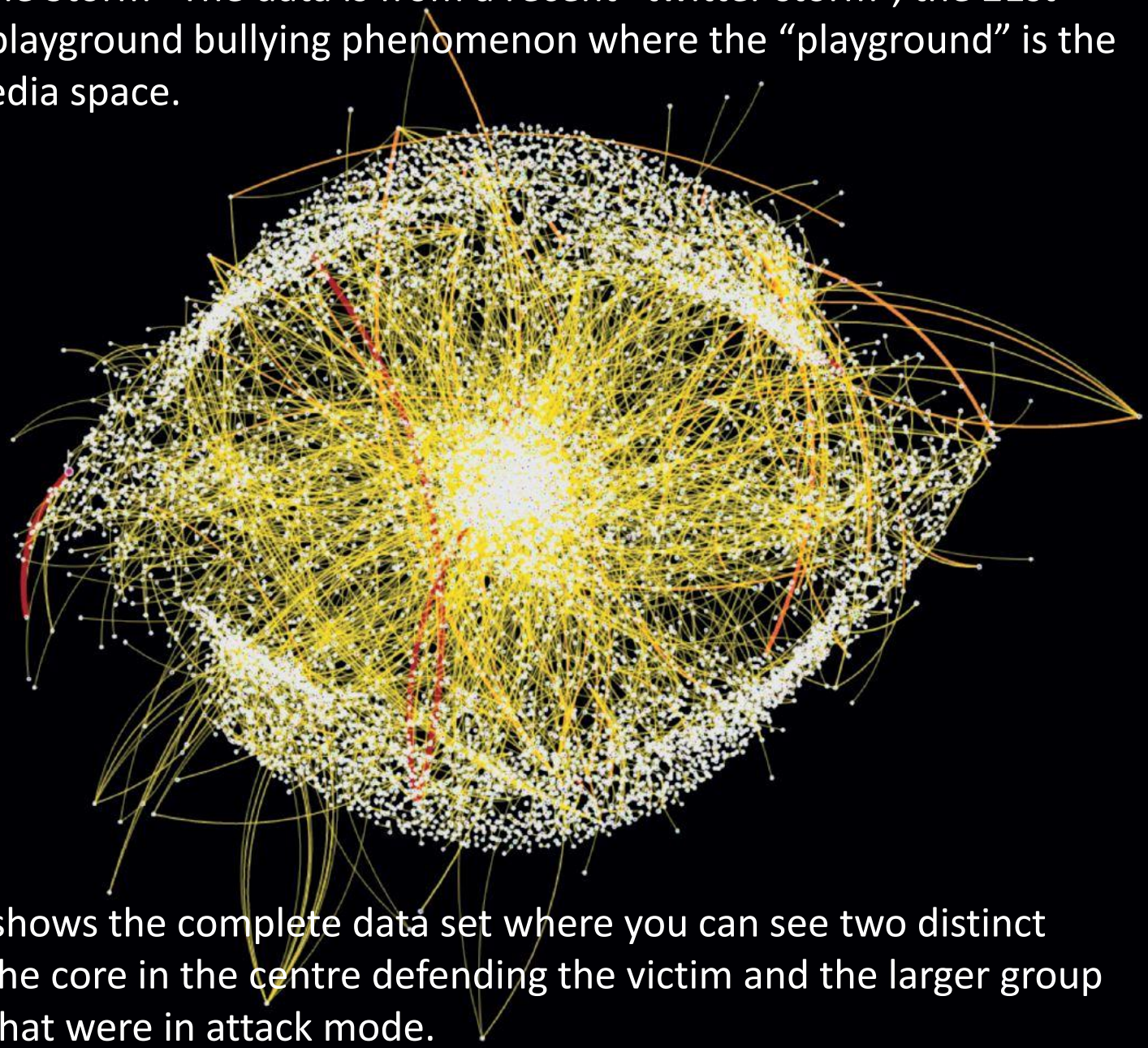
Yasmeen Ahmad



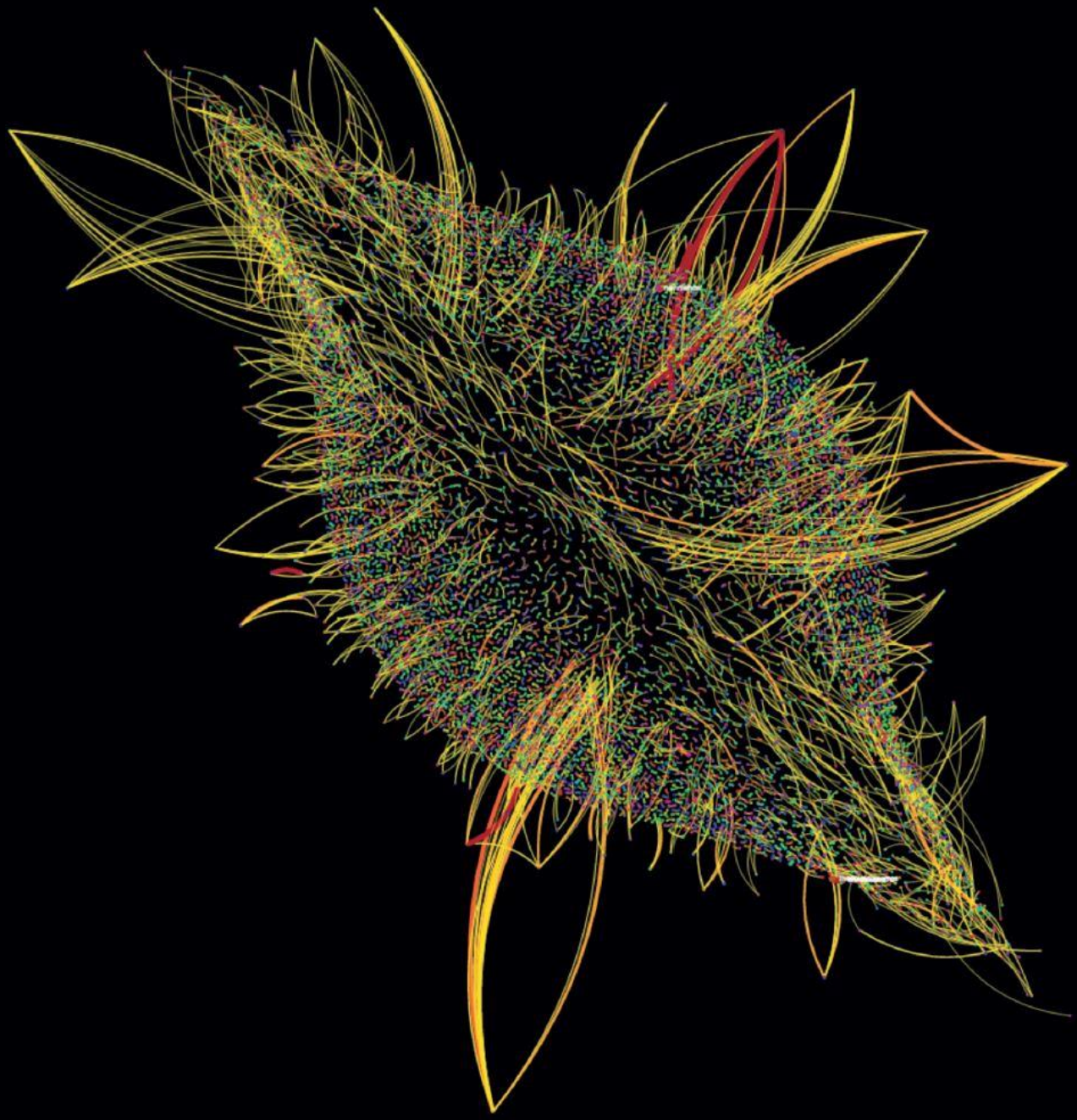
https://community.teradata.com/t5/Learn-Data-Science/The-Art-of-Analytics-Poster/ta-p/80316

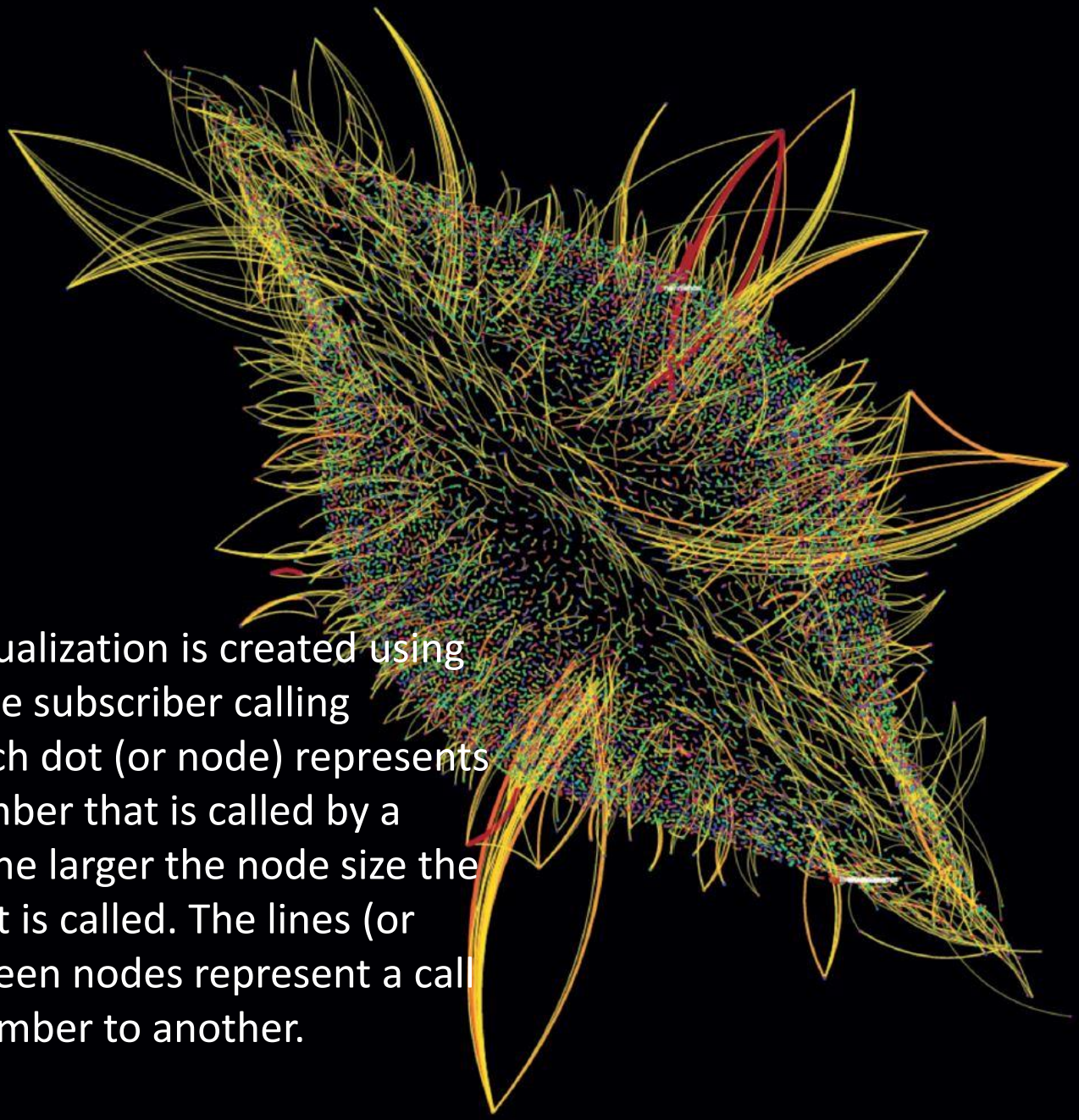


“Eye of the Storm” The data is from a recent "twitter storm”, the 21st century playground bullying phenomenon where the “playground” is the social media space.



The eye shows the complete data set where you can see two distinct groups, the core in the centre defending the victim and the larger group outside that were in attack mode.





This data visualization is created using mobile phone subscriber calling patterns. Each dot (or node) represents a phone number that is called by a subscriber, the larger the node size the more often it is called. The lines (or edges) between nodes represent a call from one number to another.

Graph – Neo4J

- Pros – excellent for examining relationships between objects, think:
  - Facebook
  - Travel problems
  - Customers
  - Fraud
- Cons – rubbish at anything else
- Tipping points – the need to track nodes and edges

Graph – Neo4J

- Schema applied when data stored
- But schema is light(ish) because all of the nodes and edges don't have to store the same data
- Analytical rather than transactional (although ACID compliant).

FEEDBACK FORMS

PLEASE FILL OUT AND PASS TO YOUR ROOM HELPER BEFORE YOU LEAVE THE SESSION