Data Design and Modeling

Università della Svizzera italiana

Graph Theory and Graph Databases

Marco Brambilla

@marcobrambi

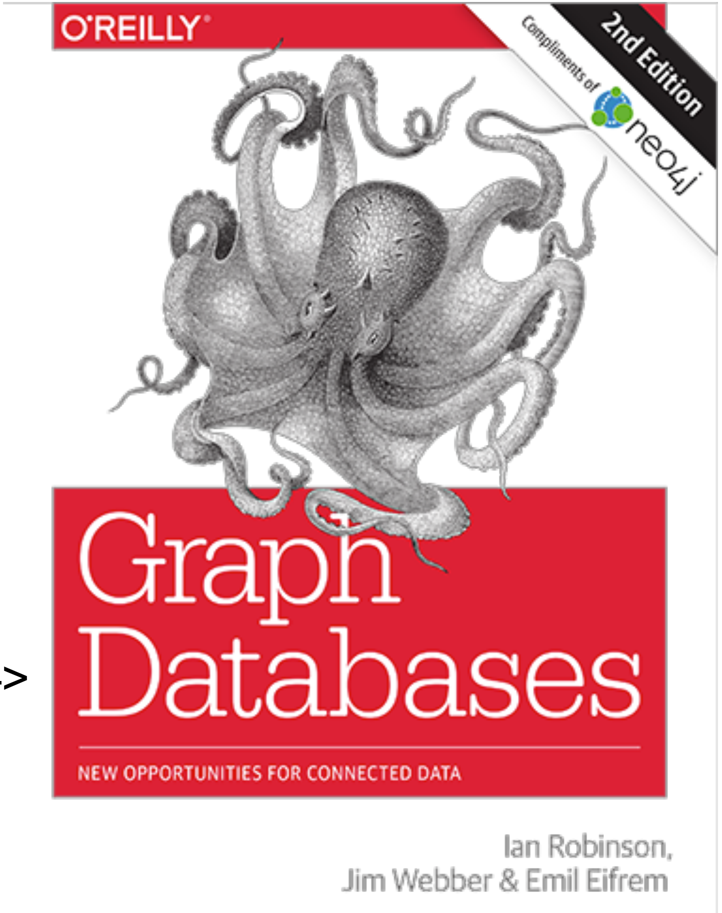
marco.brambilla@usi.ch

Agenda

Graph Theory

Graph Databases

Get it for free on Neo4J.org –>



Data Design and Modeling

1. Graph Theory

Marco Brambilla

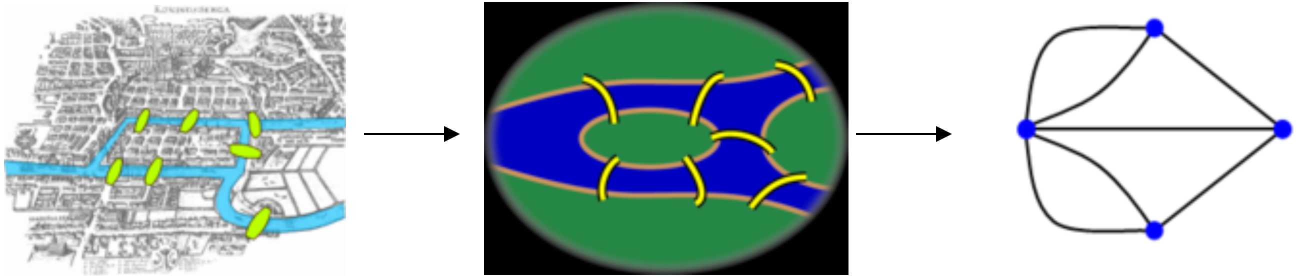
@marcobrambi

marco.brambilla@usi.ch

Graph Theory - History

Leonhard Euler's paper on "*Seven Bridges of Königsberg*",

published in 1736.



Famous problems

“The traveling salesman problem”

A traveling salesman is to visit a number of cities; how to plan the trip so every city is visited once and just once and the whole trip is as short as possible ?

In 1852 Francis Guthrie posed the “four color problem” which asks if it is possible to color, using only four colors, any map of countries in such a way as to prevent two bordering countries from having the same color.

SOLVED ONLY 120 YEARS LATER!

Other Examples

Cost of wiring electronic components

Shortest route between two cities.

Shortest distance between all pairs of cities in a road atlas.

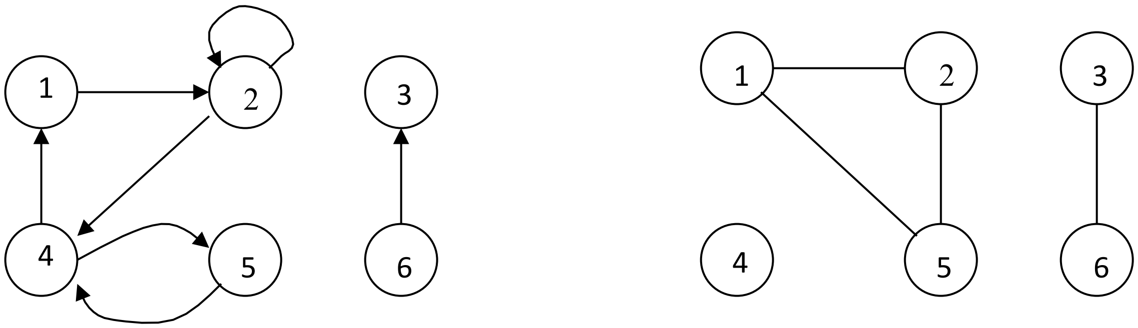
Matching / Resource Allocation

Task scheduling

Visibility / Coverage

What is a Graph?

Informally a *graph* is a set of nodes joined by a set of lines or arrows.



Definition: Graph

G is an ordered triple G:=(V, E, f)

- V is a set of nodes, points, or vertices.
- E is a set, whose elements are known as edges or lines.
- f is a function
  - maps each element of E
  - to an unordered pair of vertices in V.

Definitions

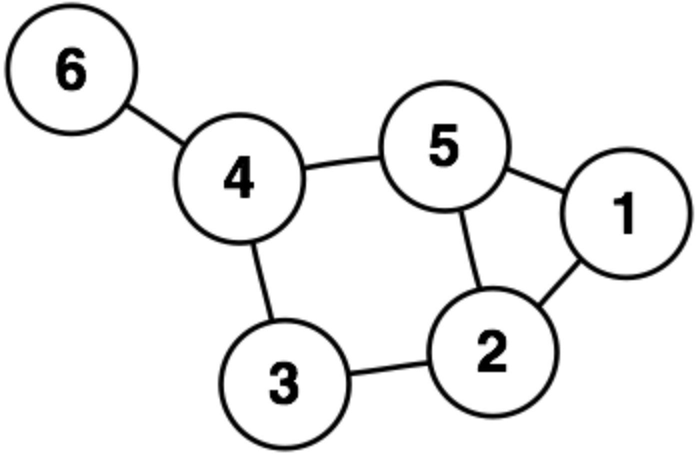
Vertex

- Basic Element
- Drawn as a *node* or a *dot*.
- V**ertex set** of *G* is usually denoted by *V*(*G*), or *V*

Edge

- A set of two elements
- Drawn as a line connecting two vertices, called end vertices, or endpoints.
- The edge set of G is usually denoted by E(G), or E.

Example



V:={1,2,3,4,5,6}

E:={{1,2},{1,5},{2,3},{2,5},{3,4},{4,5},{4,6}}

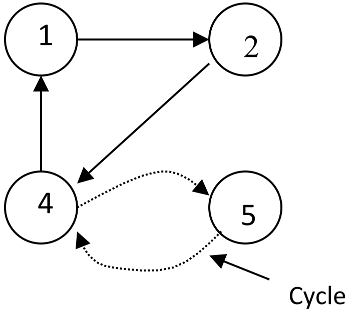
Simple Graphs

Simple graphs are graphs without multiple edges or self-loops.

Path

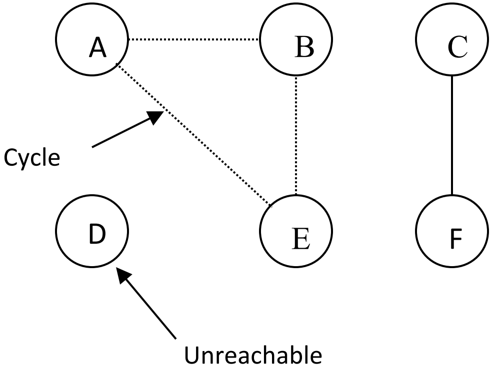
A *path* is a sequence of vertices such that there is an edge from each vertex to its successor.

A path is ***simple*** if each vertex is distinct.



Simple path from 1 to 5
= [1, 2, 4, 5]

Our text's alternates the vertices and edges.



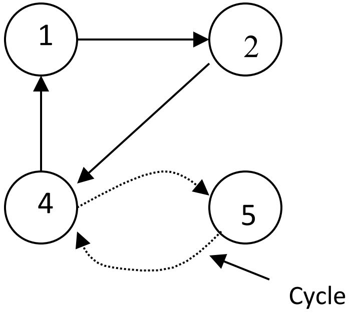
If there is path *p* from *u* to *v* then we say *v* is **reachable** from *u* via *p*.

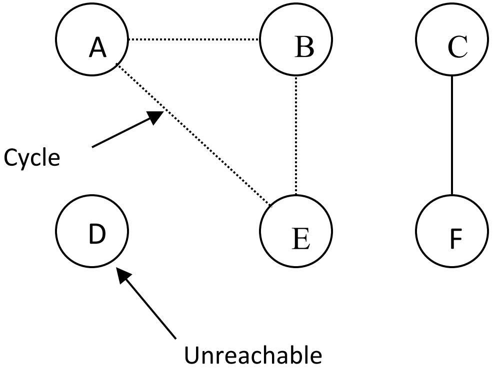
Cycle

A path from a vertex to itself is called a ***cycle***.

A graph is called ***cyclic*** if it contains a cycle;

otherwise it is called ***acyclic***





Connectivity

A graph is ***connected*** if

- you can get from any node to any other by following a sequence of edges OR
- any two nodes are connected by a path.

A directed graph is ***strongly connected*** if there is a directed path from any node to any other node.

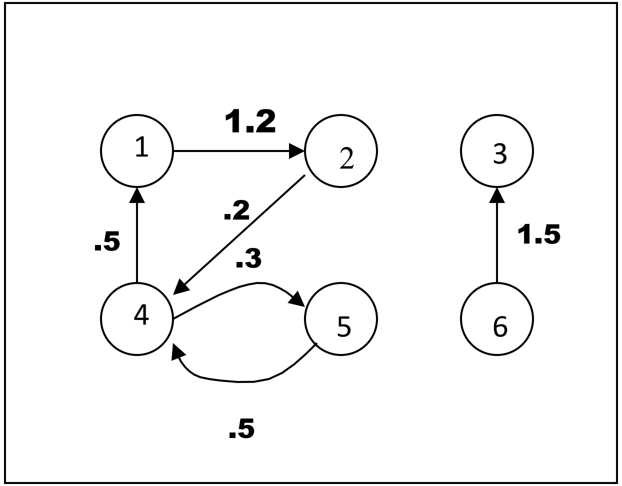
Sparse/Dense

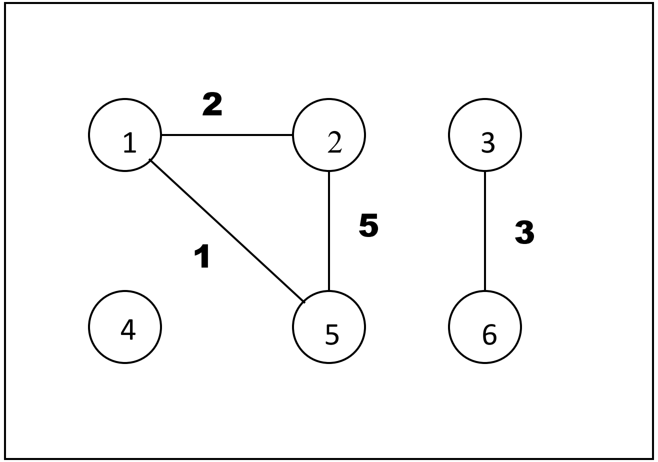
A graph is ***sparse*** if $| E | \approx | V |$

A graph is ***dense*** if $| E | \approx | V |^2$.

A ***weighted graph***

is a graph for which each edge has an associated ***weight***, usually given by a ***weight function*** *w: E →* **R**.

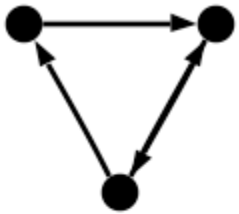




Directed Graph (digraph)

Edges have directions

An edge is an *ordered* pair of nodes

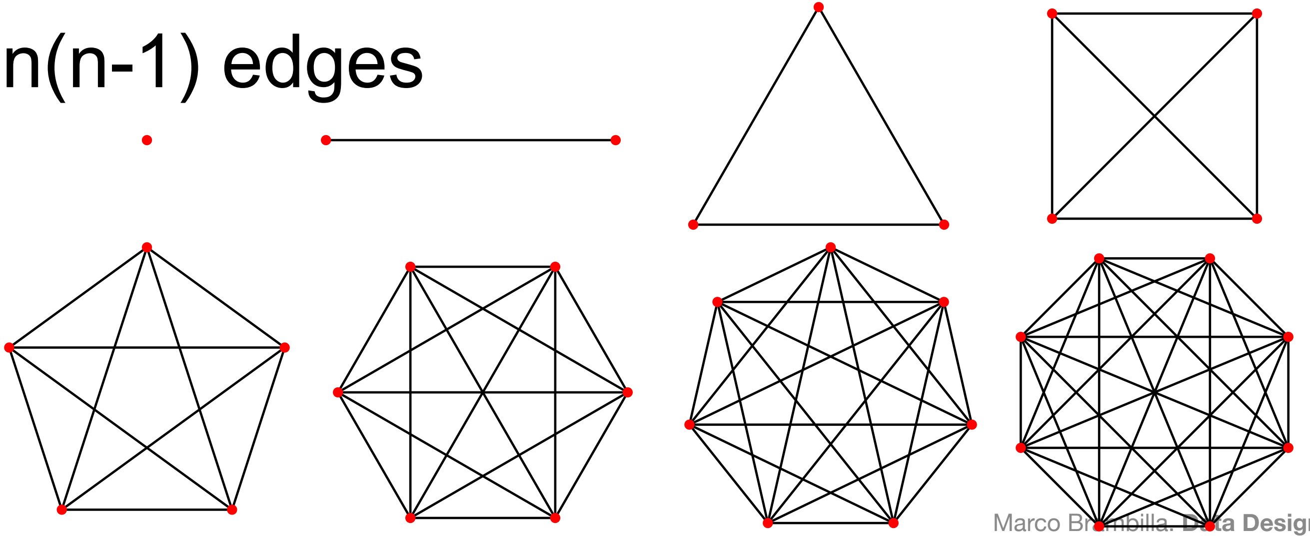


Complete Graph

Denoted K_n

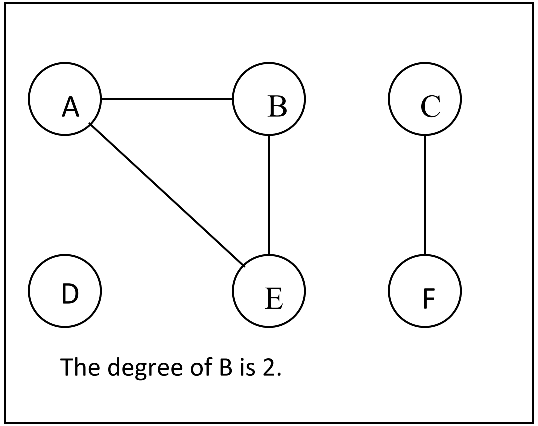
Every pair of vertices are adjacent

Has n(n-1) edges



Degree

Number of edges incident on a node

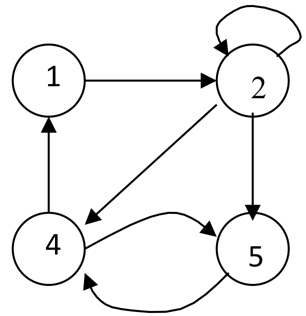


Degree (Directed Graphs)

In degree: Number of edges entering

Out degree: Number of edges leaving

Degree = indegree + outdegree



The in degree of 2 is 2 and
the out degree of 2 is 3.

Subgraph

Vertex and edge sets are subsets of those of G

- a *supergraph* of a graph G is a graph that contains G as a subgraph.

Representation (Matrix)

Incidence Matrix

- E x V
- [edge, vertex] contains the edge's data

Adjacency Matrix

- V x V
- Boolean values (adjacent or not)
- Or Edge Weights

Representation (List)

Edge List

- pairs (ordered if directed) of vertices
- Optionally weight and other data

Adjacency List

Graph Algorithms

- Shortest Path
  - Single Source
  - All pairs (Ex. Floyd Warshall)
- Network Flow
- Matching
  - Bipartite
  - Weighted
- Topological Ordering
- Strongly Connected

Graph Algorithms

Biconnected Component / Articulation Point
Bridge
Graph Coloring
Euler Tour
Hamiltonian Tour
Clique
Isomorphism
Edge Cover
Vertex Cover
Visibility

Data Design and Modeling

2. Graph Databases

Marco Brambilla
@marcobrambi
marco.brambilla@usi.ch

Motivation

Relational Databases

(incredibly!)

are not good in managing relationships!

Graph Databases

Database that uses graph structures with **nodes, edges and properties to store data**

Provides **index-free adjacency**

- Every node is a pointer to its adjacent element

Edges hold most of the important information and connect

- nodes to other nodes
- nodes to properties

Advantage of Graph Databases

When there are relationships that you want to analyze, Graph databases become a very nice fit because of the data structure

Graph databases are very fast for associative data sets

- Like social networks

Map more directly to object oriented applications

- Object classification and Parent->Child relationships

Relational Database Representation

Sailor(<u>sid:integer</u>, sname:char(10), rating: integer, age:real)

Boat(<u>bid:integer</u>, bname:char(10), color:char(10))

Reserve(<u>sid:integer, bid:integer, day:date</u>)

Sailor

<u>sid</u>	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

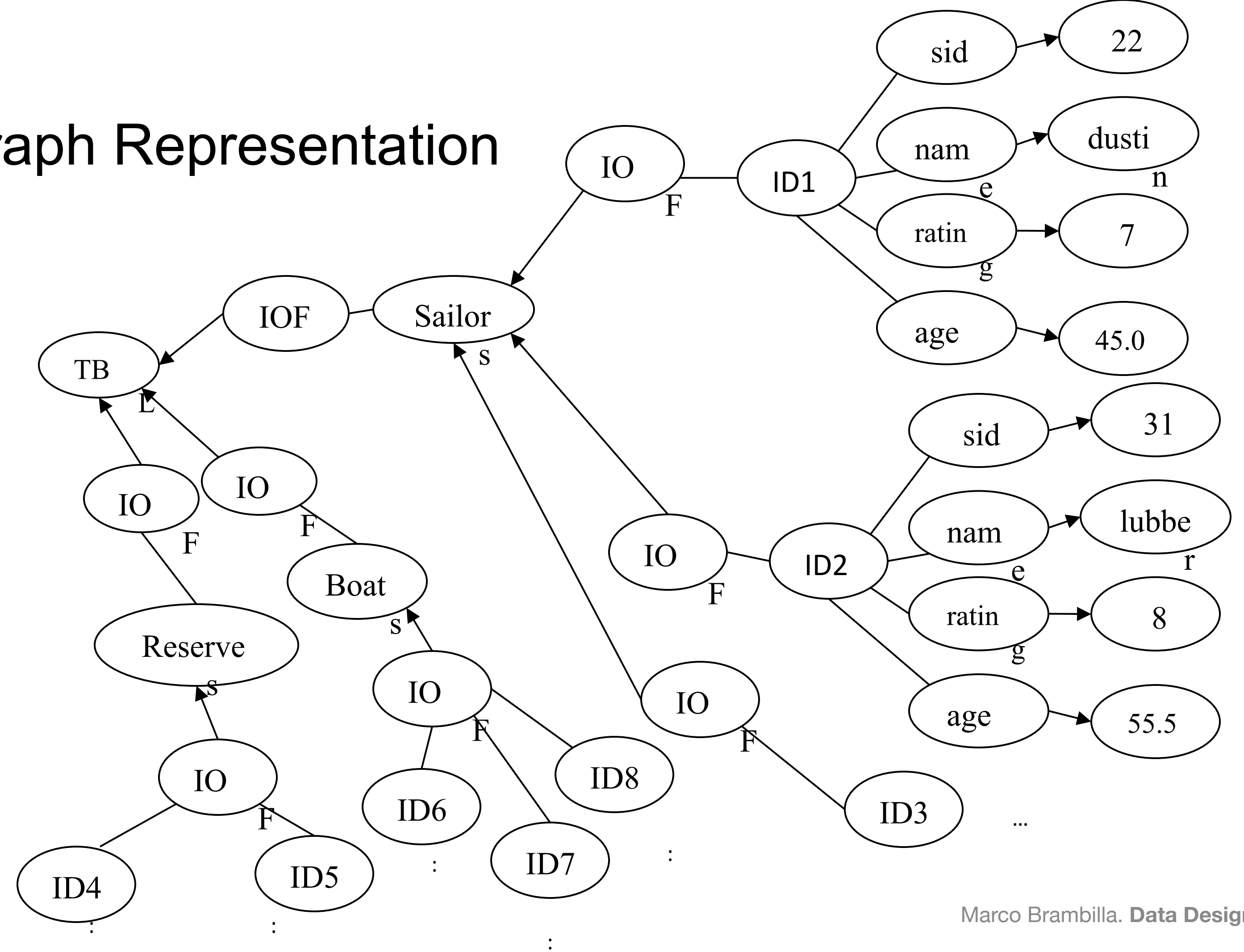
Reserve

<u>sid</u>	<u>bid</u>	<u>day</u>
22	101	10/10/96
58	103	11/12/96

Boat

<u>bid</u>	bname	color
101	Interlake	red
102	Clipper	green
103	Marine	red

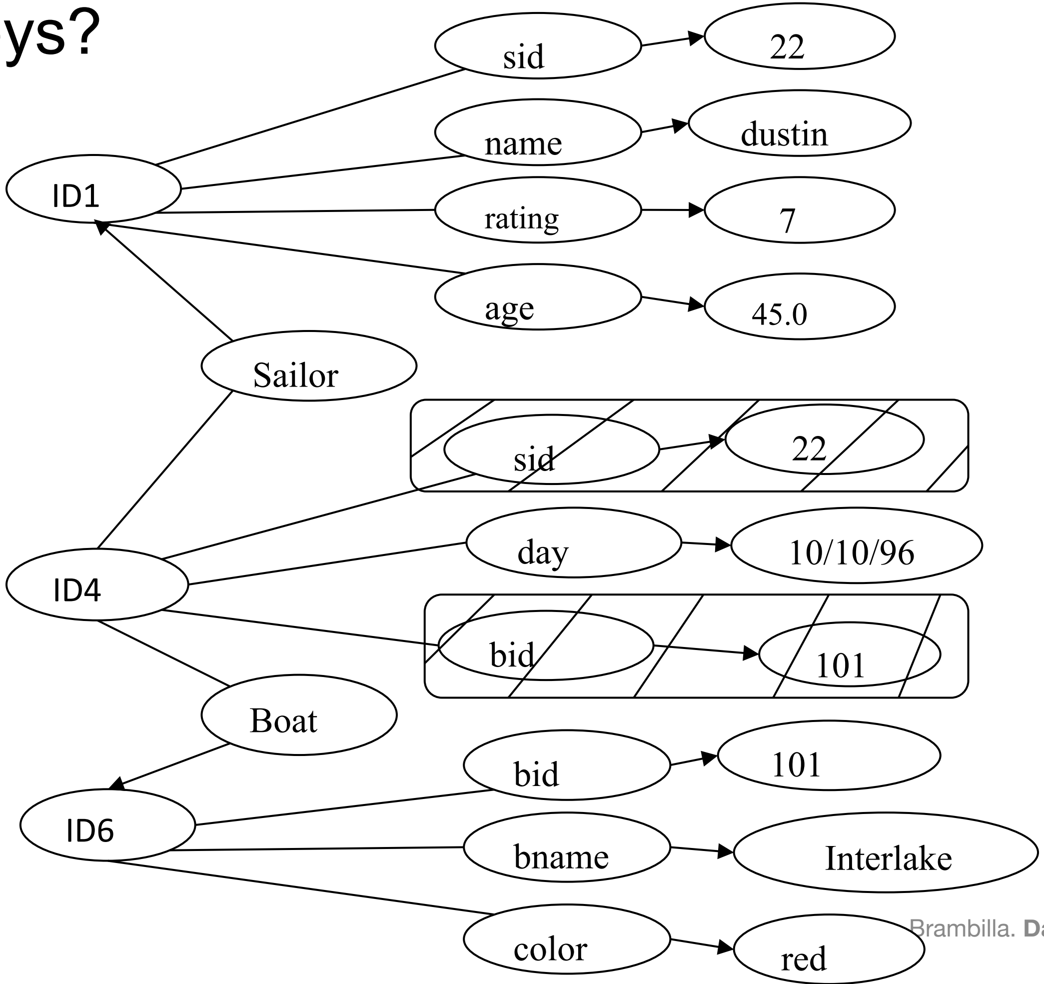
Graph Representation



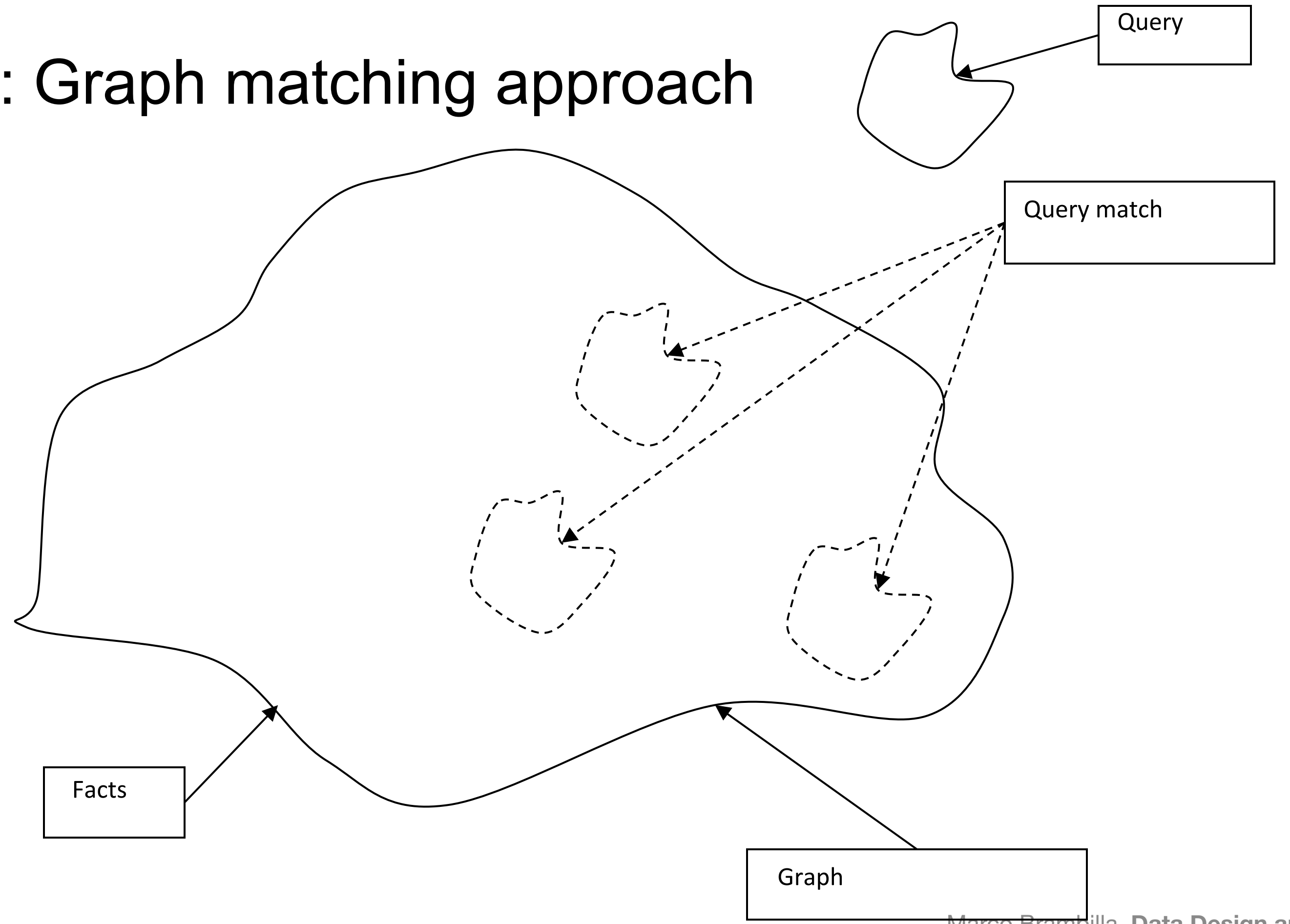
Actual Graph Model

(:Sailor) –[:reserves]-> (:Boat)

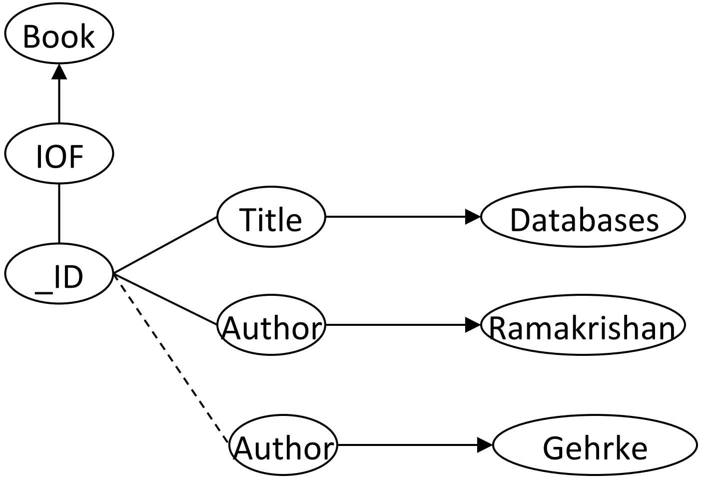
Foreign Keys?
No thanks



Query: Graph matching approach



Easy to Extend



Easy to Change