Lecture 16 – Graphs

CS2209
Information Storage and Management II

Dr Harry Nguyen
<hn@cs.ucc.ie>

A TRADITION OF INDEPENDENT THINKING

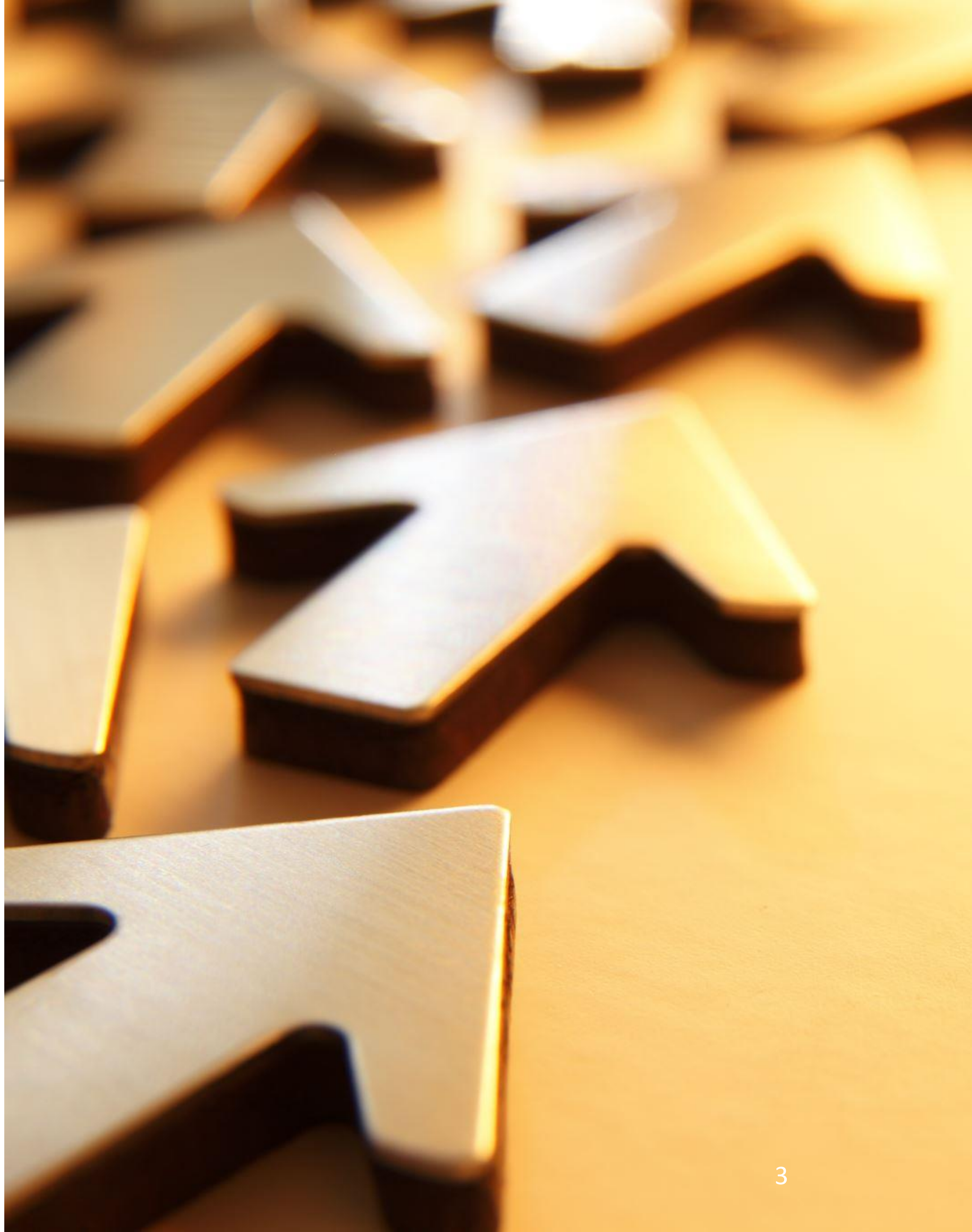
UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

What We Saw Last Lecture

- Distributed Databases
- Partitioning
- MongoDB Scaling Out

MongoDB - Operations

- Insert
- Remove
- Update
- Query
- Aggregate
- Create Indexes
- And more...



PARALLEL VS. DISTRIBUTED

Parallel DBMSs	Distributed DBMSs
Nodes are physically close to each other	Nodes can be far from each other
Nodes connected with high-speed LAN	Nodes connected using public network
Communication cost is assumed to be small	Communication cost and problems cannot be ignored

Database Partitioning

CS2209
Information
Storage and
Management II

- Some DBMSs allow you to specify the disk location of each individual database
- This is also easy to do at the filesystem level if the DBMS stores each database in a separate directory
- Split a single logical table (or JSON file) into disjoint physical segments that are stored/managed separately
- Ideally partitioning is transparent to the application

VERTICAL PARTITIONING

CS2209 Information Storage and Management II

- Store a table's attribute in a separate location
- Have to store tuple information to reconstruct the original table

Partition 1

Tuple 1	$attr_{11}$	$attr_{12}$	$attr_{13}$	$attr_{14}$
Tuple 2	$attr_{21}$	$attr_{22}$	$attr_{23}$	$attr_{24}$
Tuple 3	$attr_{31}$	$attr_{32}$	$attr_{33}$	$attr_{34}$
Tuple 4	$attr_{41}$	$attr_{42}$	$attr_{43}$	$attr_{44}$

Partition 2

Tuple 1	$attr_{14}$
Tuple 2	$attr_{24}$
Tuple 3	$attr_{34}$
Tuple 4	$attr_{44}$

Horizontal Partitioning

- Divide the tuples of a table up to disjoint segments based on some partitioning policy
  - Round robin
  - Range partitioning
  - Hash partitioning

Tuple 1	$attr_{11}$	$attr_{12}$	$attr_{13}$	$attr_{14}$
Tuple 2	$attr_{21}$	$attr_{22}$	$attr_{23}$	$attr_{24}$

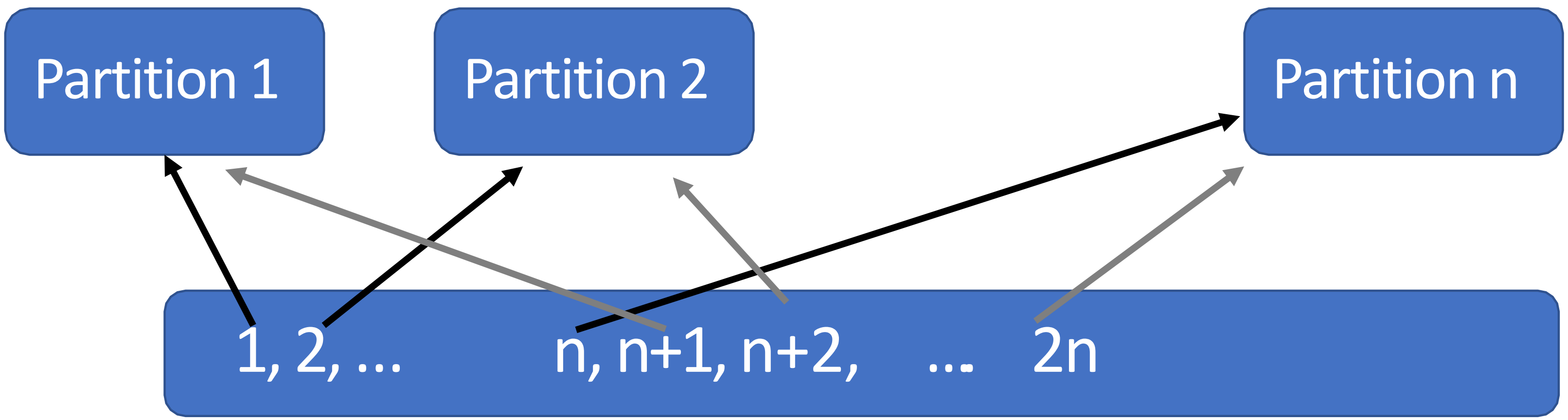
Tuple 3	$attr_{31}$	$attr_{32}$	$attr_{33}$	$attr_{34}$
Tuple 4	$attr_{41}$	$attr_{42}$	$attr_{43}$	$attr_{44}$

Load Balancing

- Improves the distribution of workloads
- Aims to optimize resource usage, maximize throughput
- Avoid overload of any single resource
- Usually involves dedicated software or hardware

Round-Robin Data Partitioning

- Each record is allocated to a partition in a clockwise manner
- Equal partitioning
- Data is evenly distributed, hence supporting **load balance**
- But data is not grouped semantically



ELASTICITY

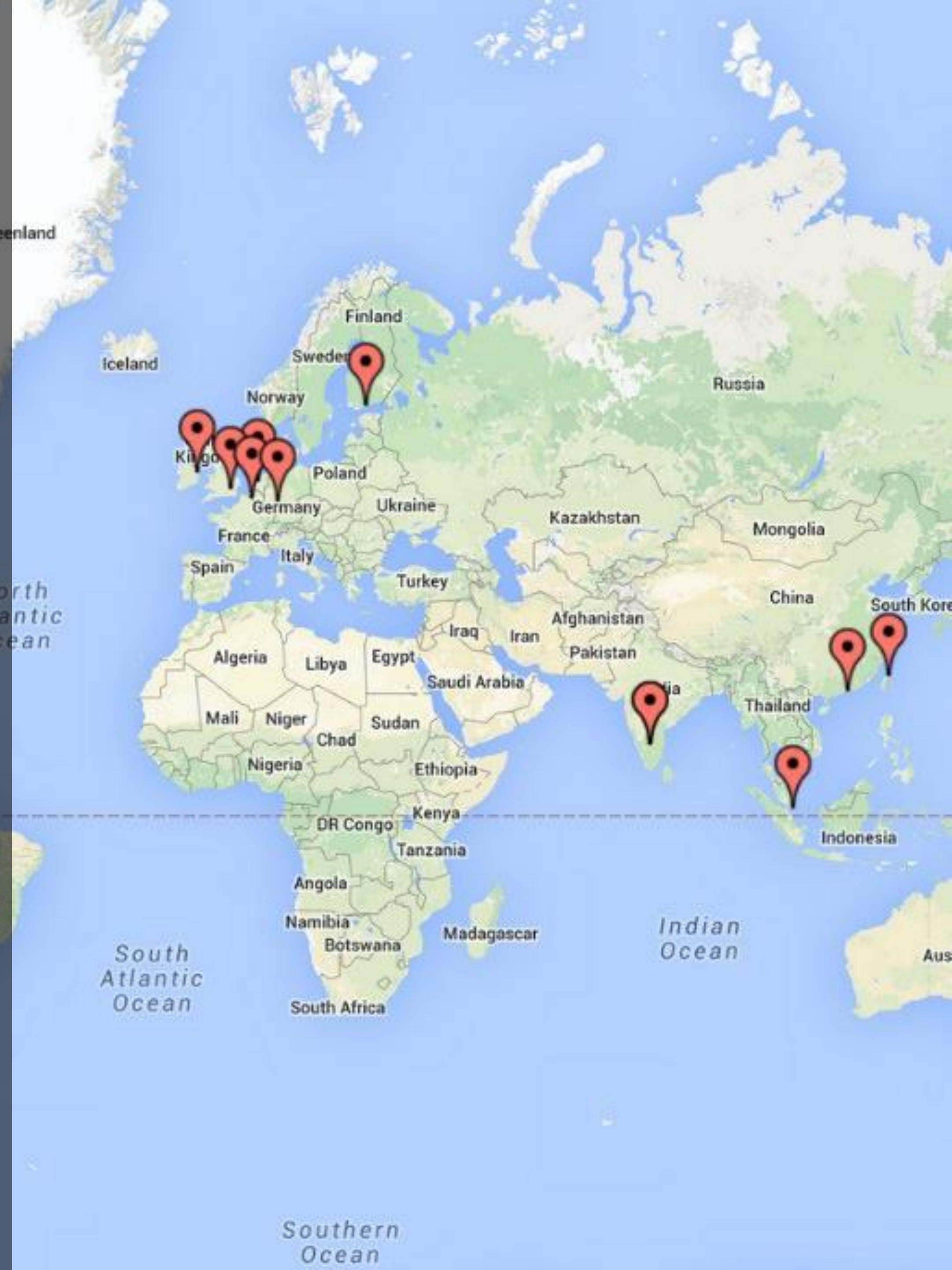
Elasticity allows flexibility: model and clustering capabilities

Allows (a gradual) increment or removal of nodes from the distributed data storage

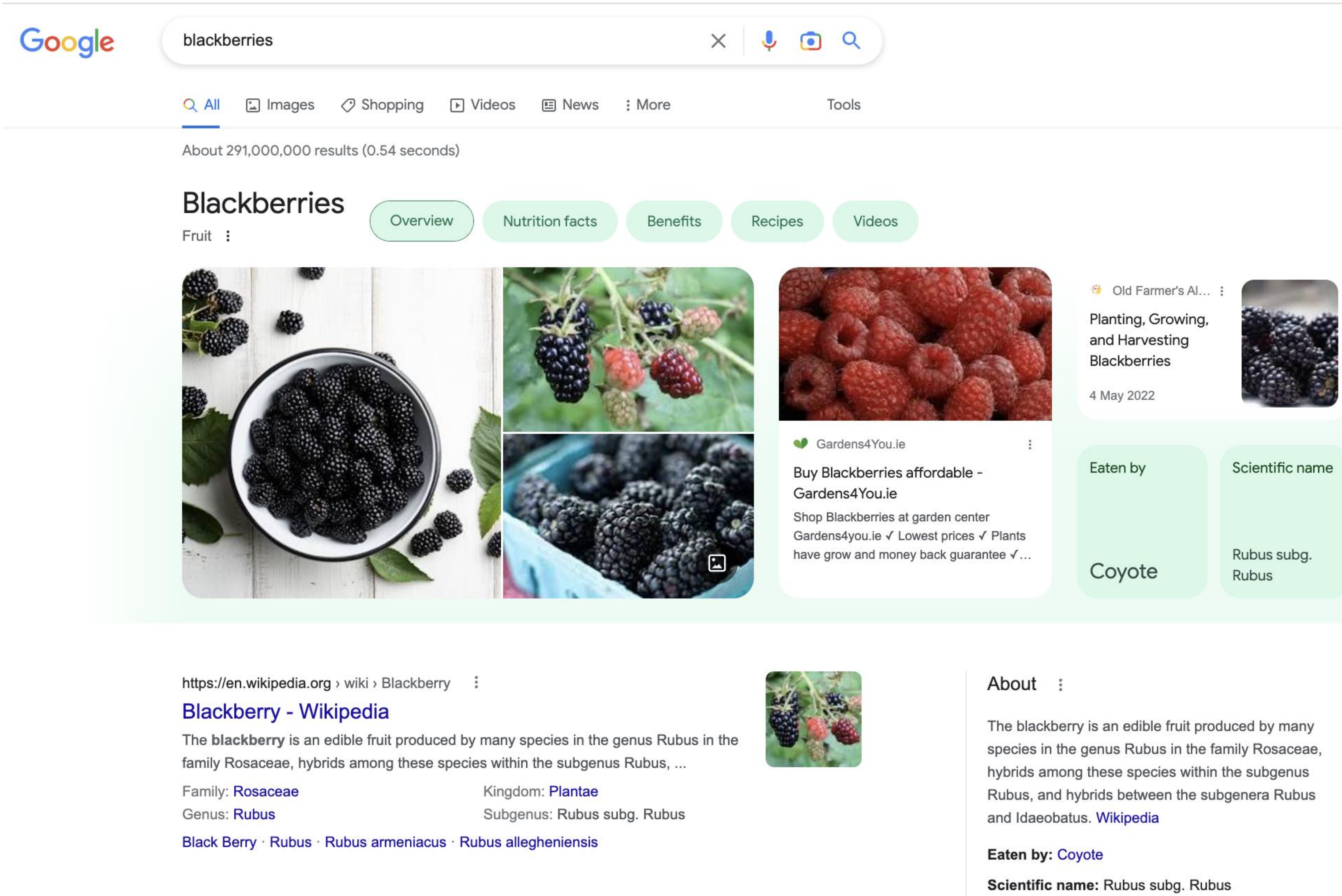
Usually a difficult task - rebalance partitions

MongoDB for Scaling Out

- If a company has many DCs
  - The collection users will occupy a huge amount of data
  - The collection is divided across multiple DCs



Google Search Example



GRAPHS AS MODELS

- Networks:
  - Transportation
  - Roadmaps
  - Computers
  - Electrical
  - …

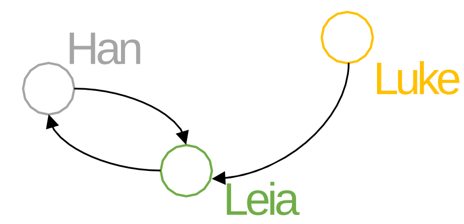
GRAPH

- A graph is a formalism for representing relationships among items. One way to represent graphs:

- A graph **G = (V,E)**
  - A set of vertices, also known as nodes

    $\mathbf{V = \{v_1,v_2,\ldots,v_n\}}$
  - A set of edges

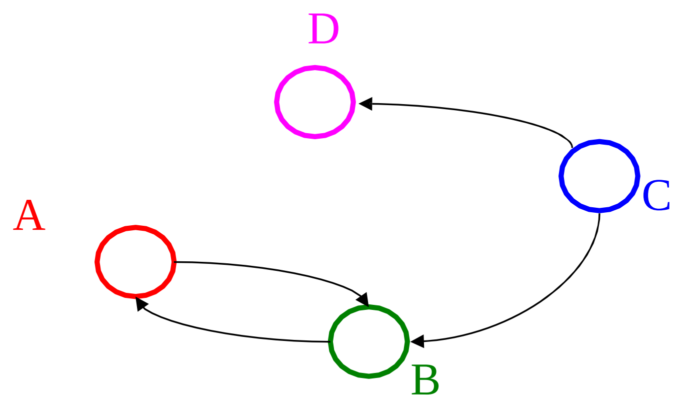
    $\mathbf{E = \{e_1,e_2,\ldots,e_m\}}$
    - Each edge $\mathbf{e_i}$ is a pair of vertices $\mathbf{(v_j,v_k)}$
    - An edge "connects" the vertices
    - It can also be represented as $\mathbf{v_jv_k}$




```
V = {Han,Leia,Luke}
E = {(Luke,Leia),
     (Han,Leia),
     (Leia,Han)}
```

Graph Example

- How are the sets V and E for this graph?

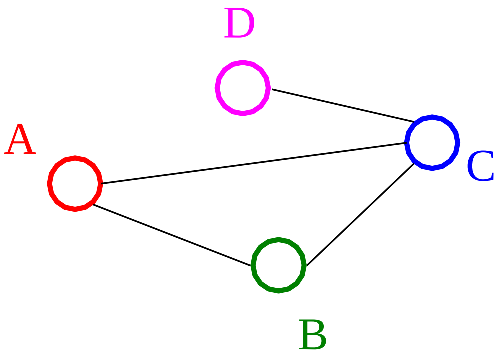



```
V = {A, B, C, D}

E = { (C, B),
      (A, B),
      (B, A),
      (C, D) }
```

UNDIRECTED GRAPHS

- In undirected graphs, edges have no specific direction
  - Edges are always “two-way” D

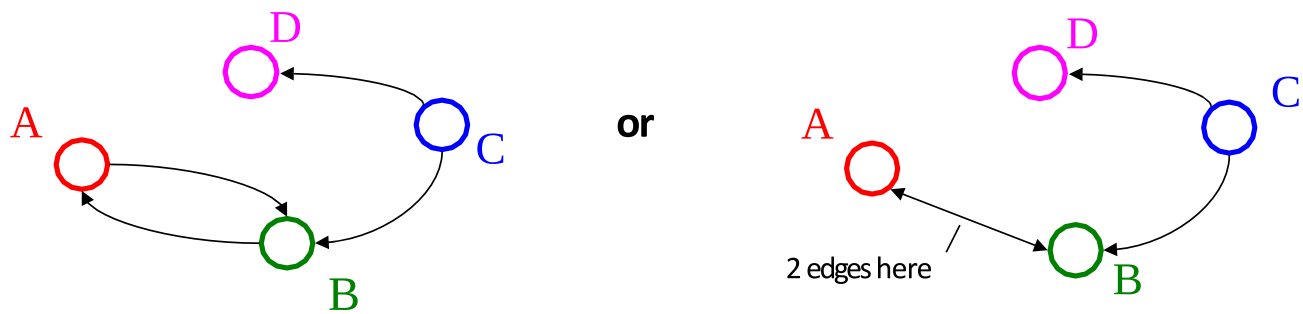


- Thus, (u,v) ∈ E implies (v,u) ∈ E
  - Only one of these edges needs to be in the set, the other one is implicit.
- Degree of a vertex: number of edges containing that vertex
  - Put another way: the number of adjacent vertices

Directed Graphs

CS2209 Information Storage and Management II

- In directed graphs (sometimes called digraphs), edges have a direction.



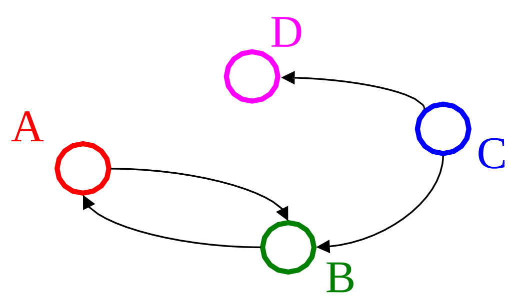
- Thus, (u,v) ∈ E does *not* imply (v,u) ∈ E
  - Let (u,v) ∈ E mean u → v
  - We call u the source and v the destination

DIRECTED GRAPHS: DEGREE

In-degree of a vertex: number of in-bound edges, i.e., edges where the vertex is the destination

Out-degree of a vertex: number of out-bound edges, i.e., edges where the vertex is the source

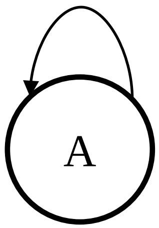
Degree of a vertex: Sum of in-degree and out-degree



SELF-EDGES (A.K.A. LOOPS)

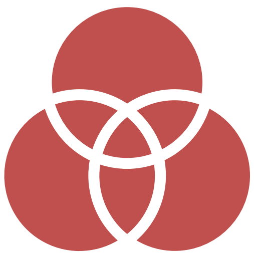
CS2209 Information Storage and Management II

- A self-edge a.k.a. a loop is an edge of the form (a,a):

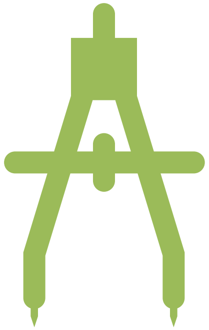


- Depending on the use/algorithm, a graph may have:
  – No loops
  – Some loops
  – All loops (often therefore implicit)

REPRESENTATIONS



Incidence Matrix:
Depicts the incidents of an edge with a vertex

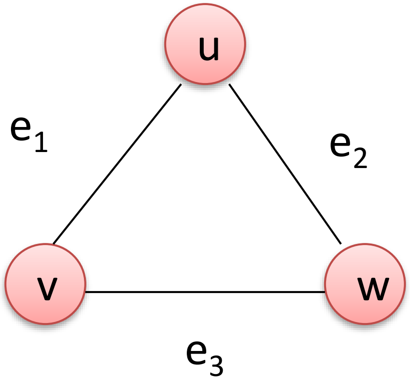


Adjacency Matrix/List:
Depicts the connections between two vertices

Incidence Matrix

CS2209
Information Storage and Management II

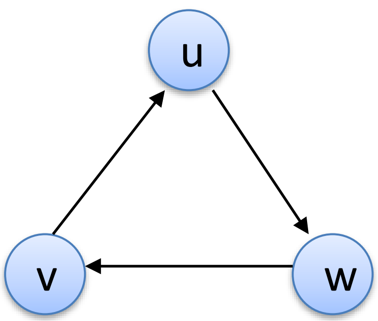
- A *n* x *m* matrix $\boldsymbol{B}$
  - *n* is the number of vertices
  - *m* is the number of edges
  - $\boldsymbol{B}_{ij} = \begin{cases} 1 \text{ if edge } e_j \text{ is incident with vertex } v_i \\ 0 \text{ otherwise} \end{cases}$



	e_1	e_2	e_3
v	1	0	1
u	1	1	0
w	0	1	1

ADJACENCY MATRIX

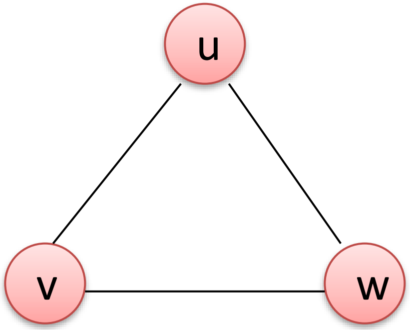
- A n x n matrix $\boldsymbol{A}$
  - n is the number of vertices
  - $\boldsymbol{A}_{ij} = \begin{cases} 1 \text{ if } \{v_i, v_j\} \text{ is an edge of the graph} \\ 0 \text{ otherwise} \end{cases}$



	v	u	w
v	0	1	0
u	0	0	1
w	1	0	0

Adjacency List

- Each vertex has a list of adjacent vertices



Vertex	Adjacency List
u	v , w
v	w, u
w	u , v

WHAT IS A GRAPH DATABASE?

- A database with an explicit graph structure
- Each node knows its adjacent nodes
- As the number of nodes increases, the cost of a local step (or hop) remains the same
- Plus an Index for lookups

SUMMARY

- Graphs
- Graph Representations
- Graph Database