Graphs On Databases

at

NEDB 2014

Talking on

Alekh Jindal

collaborate

Amol Deshpande

Supervisors

Sam Madden

Mike Stonebraker

work

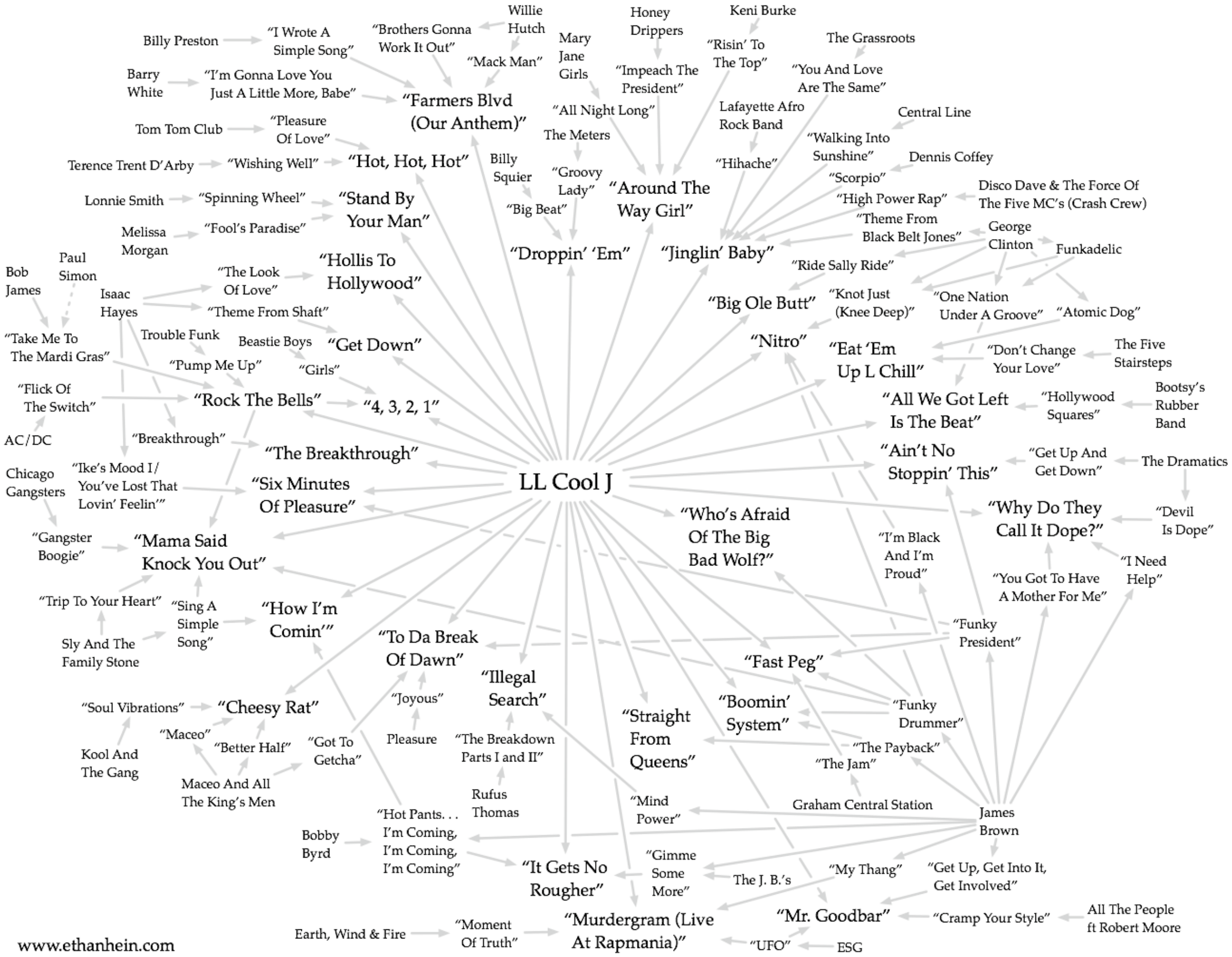
University
of Maryland

sabbatical

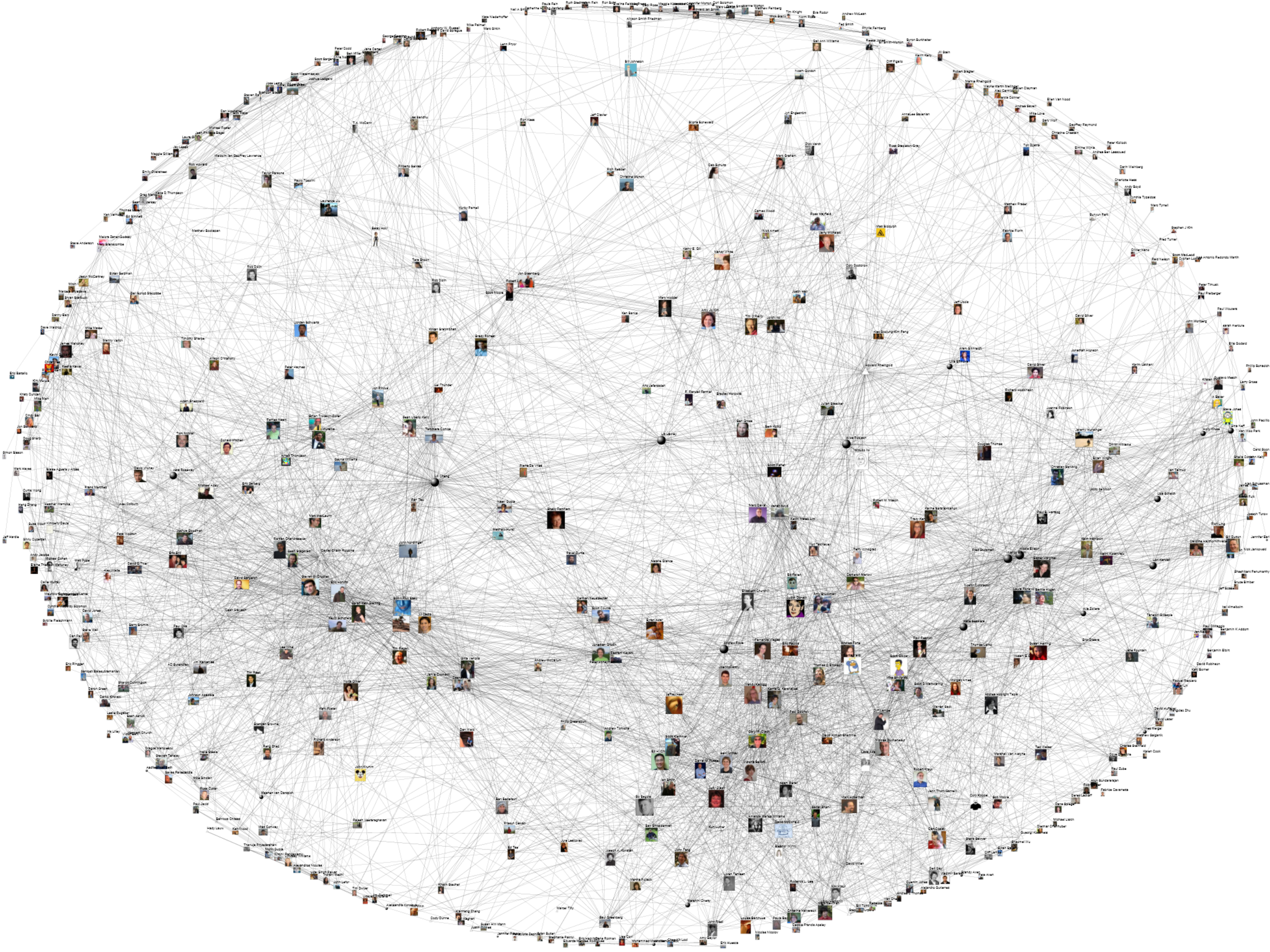
work

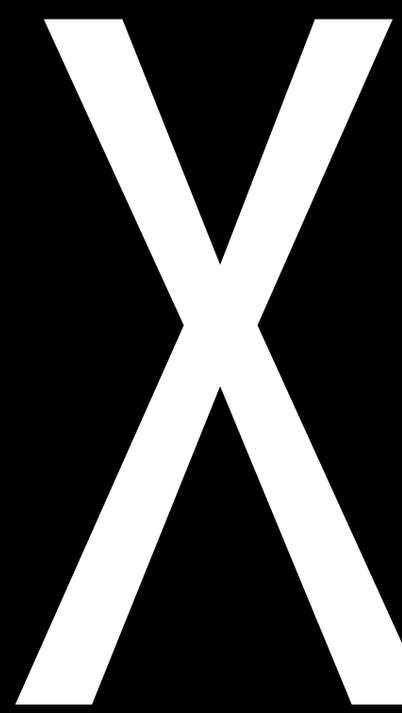
work

MIT



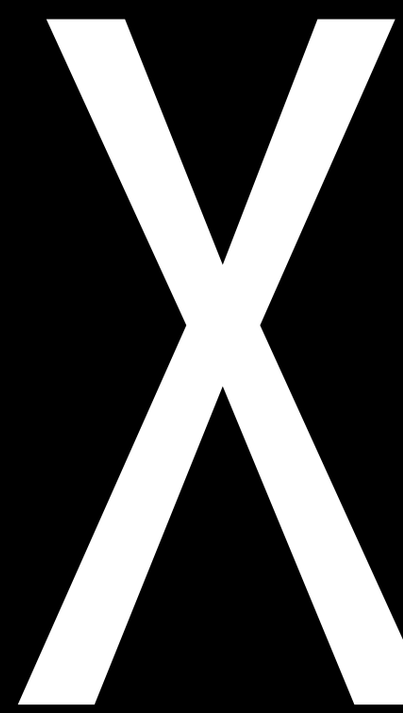
www.ethanhein.com





DATA

->



DATABASE

Relational **DATABASE**

Relational **DATA**

DATA

Streaming

DATABASE

Streaming

XML -> XML

DATA DATABASE

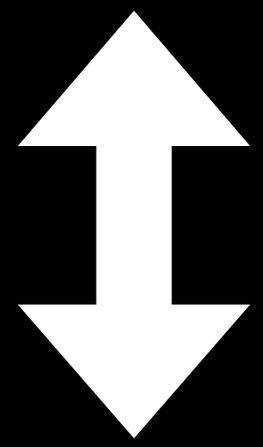
RDF -> RDF

DATA DATABASE

Graph -> Graph

DATA DATABASE

DATABASE



DATA

APPLICATIONS

Logical Data Independence

DATABASE

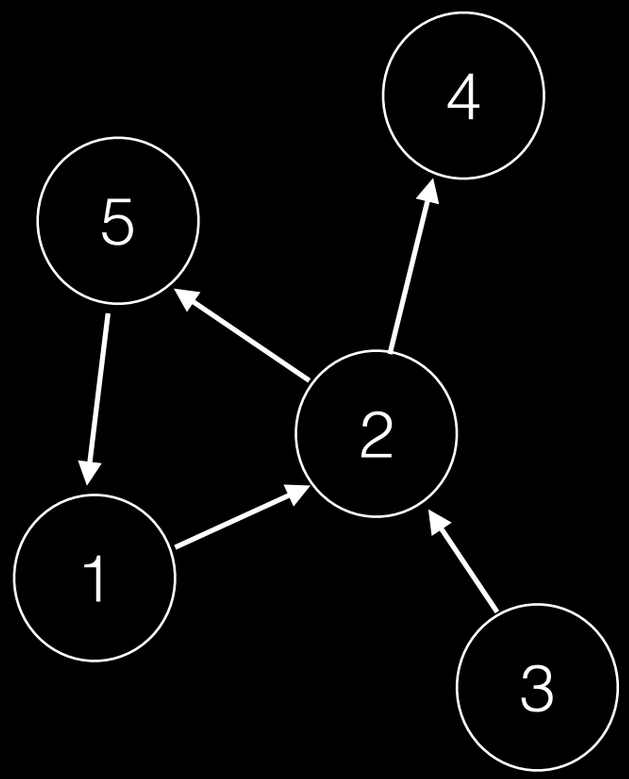
Physical Data Independence

DATA

Barriers to “Graphs on Databases”

- Graphs in relational model
- Graph operations in SQL
- Expressing iterative graph queries
- Efficient graph analytics performance
- Ease-of-use

Graphs in Relational Model



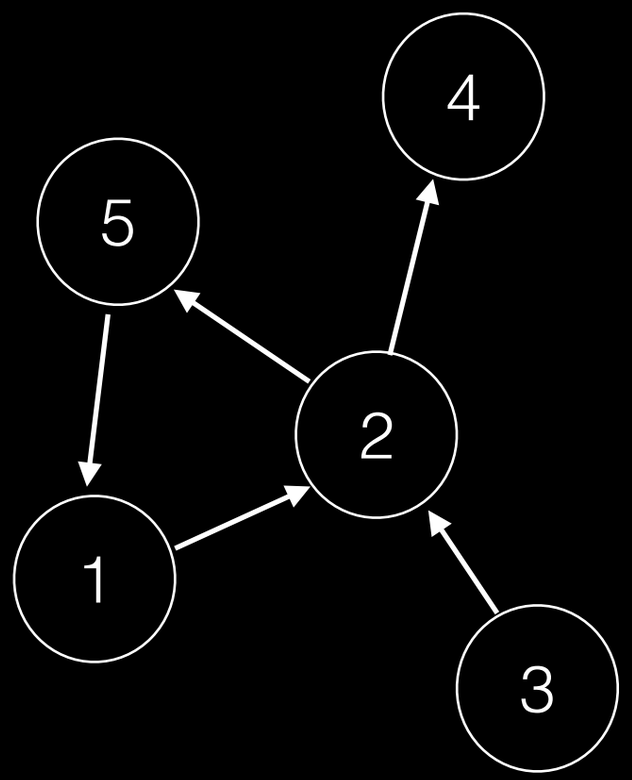
Nodes

id	value
1	1
2	1
3	1
4	1
5	1

Edges

fromId	toId	weight
1	2	1
2	4	1
2	5	1
3	2	1
5	1	1

Graphs Operations in SQL



- Node access
  Select * From Nodes Where Id=ID
- Neighborhood access
  Select * From Edges Where fromId=ID
- Parallel neighborhood access
  Select * From Edges Group By fromId
- 1-hop neighbors
  Select * From Edges e1,Edges e2 Where e1.toId=e2.fromId

Example: Shortest Paths

```
UPDATE Nodes AS node SET value= new_node.value
    FROM(
        SELECT e.toId AS Id, min(n1.value+1) AS value
        FROM Nodes AS n1, Edges AS e, Nodes AS n2
        WHERE n1.Id=e.fromId AND n2.Id=e.toId
        GROUP BY e.toId, n2.value
        HAVING min(n1.value+1) < n2.value
) AS new_node
WHERE node.Id = new_node.Id;
```

Example: Shortest Paths

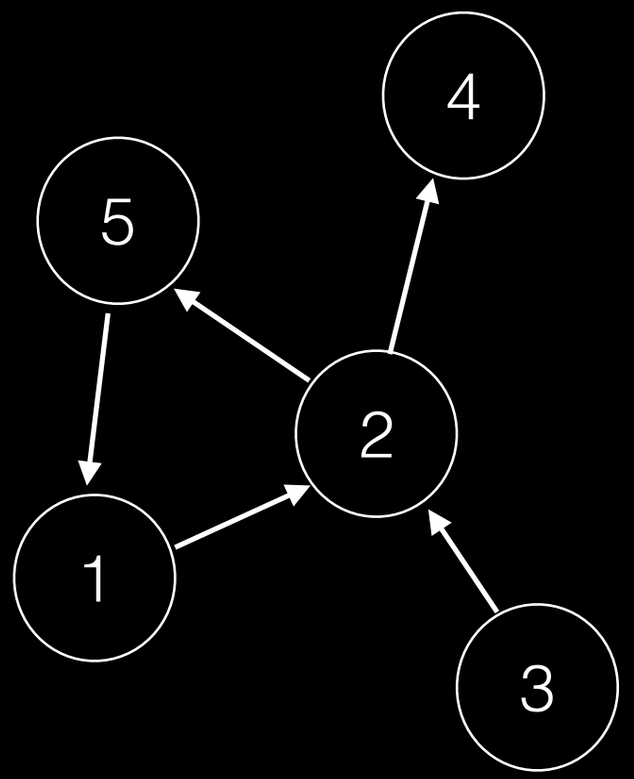
```
UPDATE Nodes AS node SET value= new_node.value
    FROM(
        SELECT e.toId AS Id, min(n1.value+1) AS value
            FROM Nodes AS n1, Edges AS e, Nodes AS n2
            WHERE n1.Id=e.fromId AND n2.Id=e.toId
            GROUP BY e.toId, n2.value
            HAVING min(n1.value+1) < n2.value
    ) AS new_node
    WHERE node.Id = new_node.Id;
```

Nested Query

Parallel Graph Exploration

Sorting/Indexing

Iterative Graph Queries



- Driver program:
  UDF / Stored Procedure

- Three Things:
  - initialization
  - actual graph query (in a loop)
  - termination condition

Example: Shortest Paths

Initialization:

1. Set the value of start node to 0
2. Set the value of all other node to inf

Loop:

The shortest paths SQL

```
UPDATE Nodes AS node SET value=new_node.value
 FROM(
  SELECT e.toId AS Id, min(n1.value+1) AS value
     FROM Nodes AS n1, Edges AS e, Nodes AS n2
     WHERE n1.Id=e.fromId AND n2.Id=e.toId
     GROUP BY e.toId, n2.value
     HAVING min(n1.value+1) < n2.value
 ) AS new_node
 WHERE node.Id = new_node.Id;
```

Termination Condition:

No more nodes to Update

Efficient Graph Analytics

- Three SQL Databases:
  - row store
  - column store
  - main-memory store

- Two Graph Databases:
  - transactional graph database
  - graph analytics system

- Two queries: PageRank, Shortest Paths

- Social network dataset from snap.stanford.edu/data

PageRank


10000
1000
100
10
1
Time (seconds)
Graph Database
Main-memory Database
Row Store Database
Apache Giraph
Column Store Database
589.0
28.0
17.4
47.0
3.3
101.5
53.5
4.2
4,172.4
218.1
29.4
Twitter
GPlus
LiveJournal

Shortest Paths

Ease-of-Use

SQL

```
UPDATE Nodes AS node SET value=new_node.value
  FROM(
    SELECT e.toId AS Id, min(n1.value+1) AS value
       FROM Nodes AS n1, Edges AS e, Nodes AS n2
       WHERE n1.Id=e.fromId AND n2.Id=e.toId
       GROUP BY e.toId, n2.value
       HAVING min(n1.value+1) < n2.value
) AS new_node
WHERE node.Id = new_node.Id;
```

Pregel

```
void compute(vector<float> messages){
  // get the minimum distance
  float mindist = id==START_NODE ? 0 : DBL_MAX;
  for(vector<float>::iterator it = messages.begin();
                        it != messages.end(); ++it)
    mindist = min(mindist,*it);
  // send messages to all edges if new minimum is found
  float vvalue = getVertexValue();
  if(mindist < vvalue){
    modifyVertexValue(mindist);
    vector<int> edges = getOutEdges();
    for(vector<int>::iterator it = edges.begin();
                          it != edges.end(); ++it)
      sendMessage(*it, mindist+1);
  }
  // halt
  voteToHalt();
}
```

Ease-of-Use

SQL

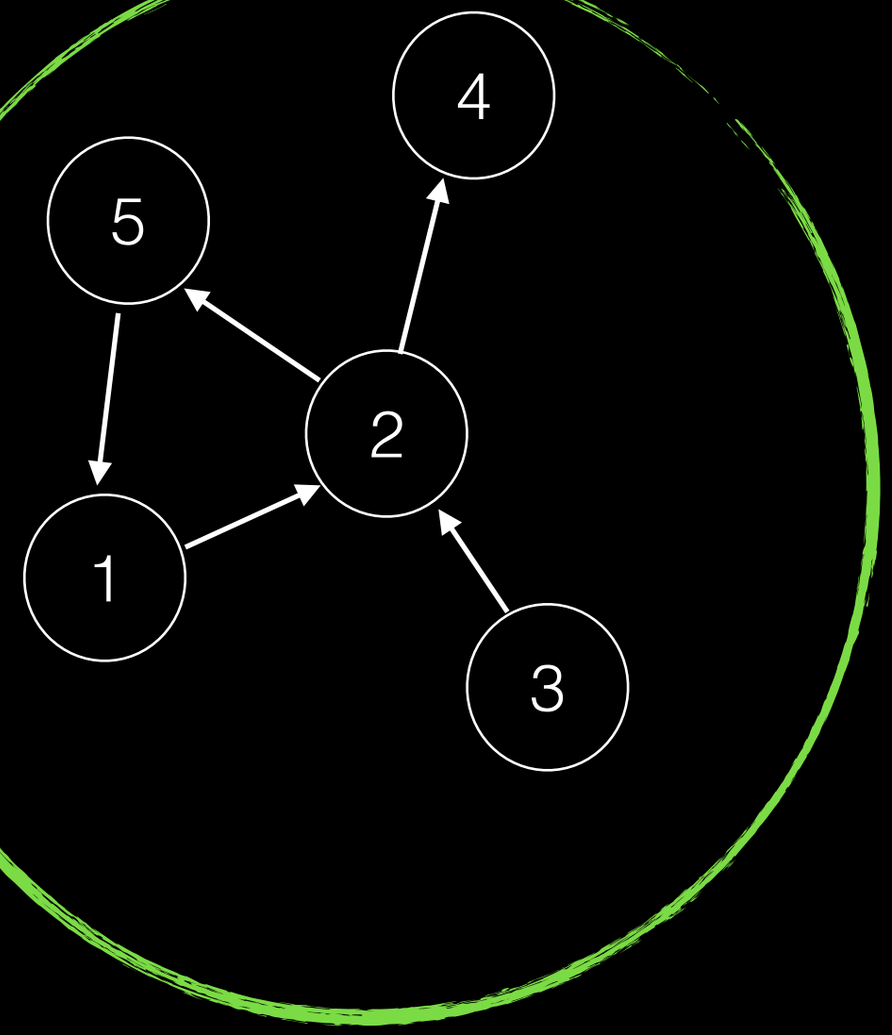
Nodes

id	value
1	1
2	1
3	1
4	1
5	1

Edges

fromId	toId	weight
1	2	1
2	4	1
2	5	1
3	2	1
5	1	1

Pregel



Vertex-centric Query Interface

APPLICATION

Vertex Programs

Pregel-style API:

- getMessages()
- getEdges()
- sendMessages()
- voteToHalt(), etc.

Logical Data Independence

DATABASE

Vertex UDF

Invokes the vertex program if:

- the vertex is active, or
- the vertex has incoming messages

Coordinator

Synchronizes supersteps

Redistributes Messages

Physical Data Independence

DATA

Vertex (V), Edge (E), Message (M)

Vertex-centric Query Interface

APPLICATION

Vertex Programs

Pregel-style API:

- getMessages()
- getEdges()
- sendMessages()
- voteToHalt(), etc.

Logical Data Independence

Batching

DATABASE

Vertex UDF

Invokes the vertex program if:

- the vertex is active, or
- the vertex has incoming messages

No in-place Updates

Coordinator

Synchronizes supersteps

Redistributes Messages

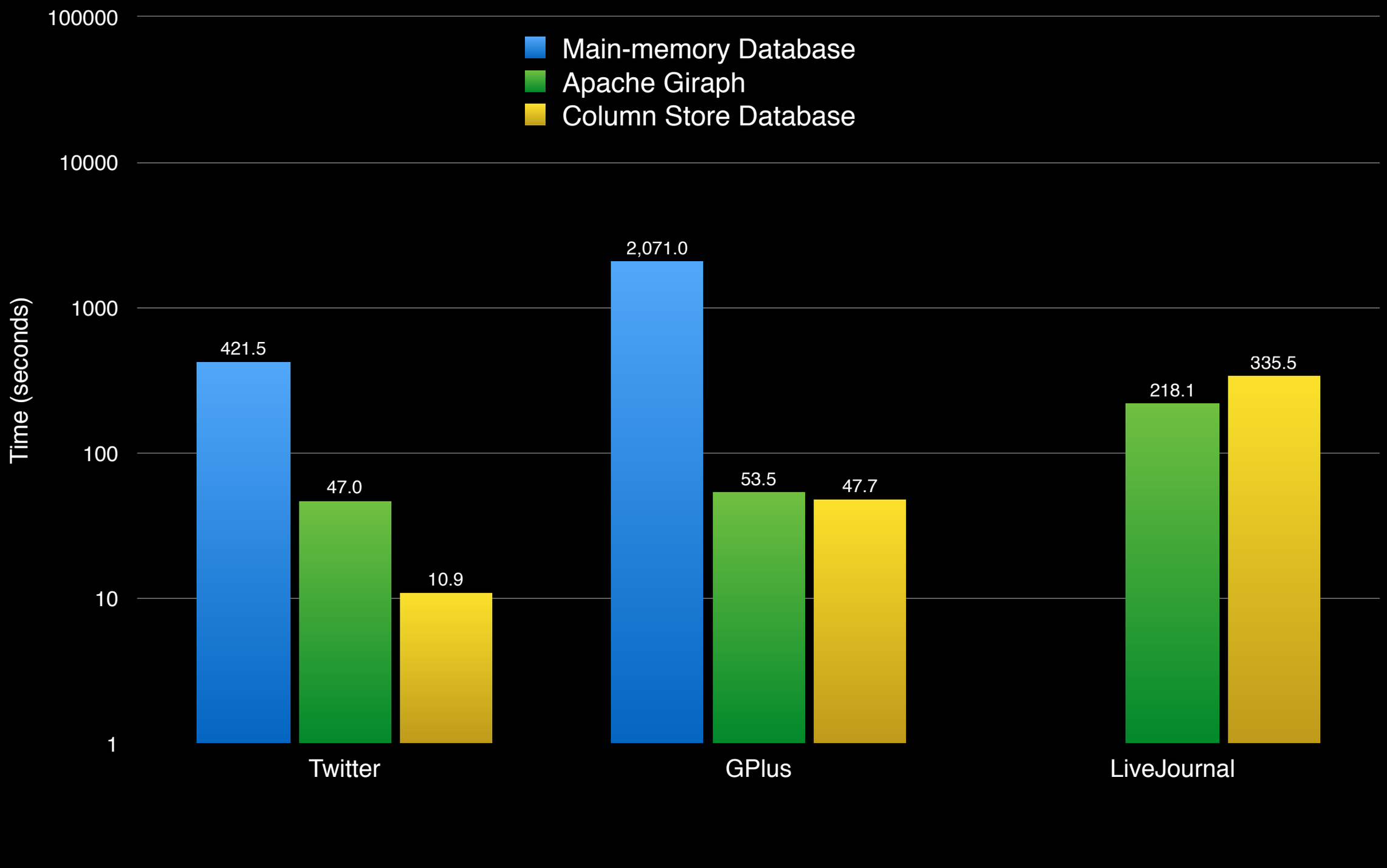
Physical Data Independence

DATA

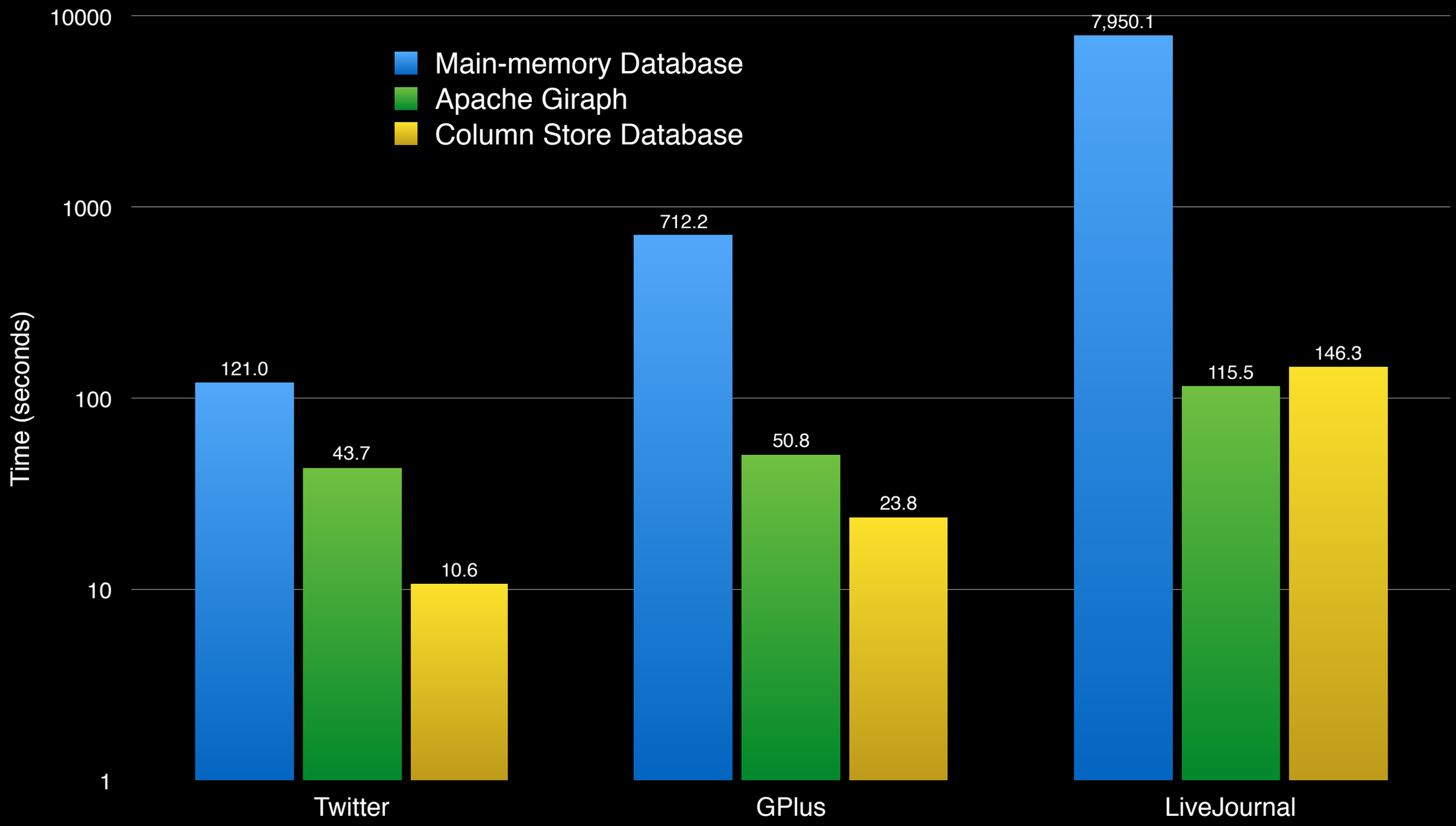
Union

Vertex (V), Edge (E), Message (M)

PageRank (Vertex)



Shortest Paths (Vertex)



Vertex-centric interface allows…

- Connected Components
- Random Walks with Restart
- Stochastic Gradient Descent
- Or, other message Passing Algorithms

…. right within the database system!

Advantages of “Graphs on Databases”

- Running arbitrary SQL queries
- Pre- and post- processing of data
- Updates are trivial
- ACID for free
- Don’t need to deal with Yet-Another-System!

Summary

- Graph analytics can be mapped to relational queries (plus UDFs)
- SQL systems can offer very good performance over relational queries
- We can extend SQL systems to provide more graph-natural query interfaces