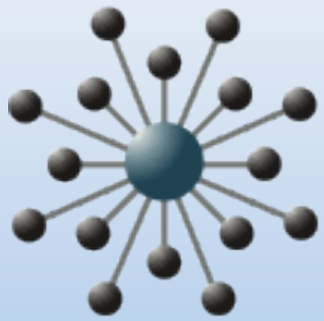


ISTC
BIG DATA

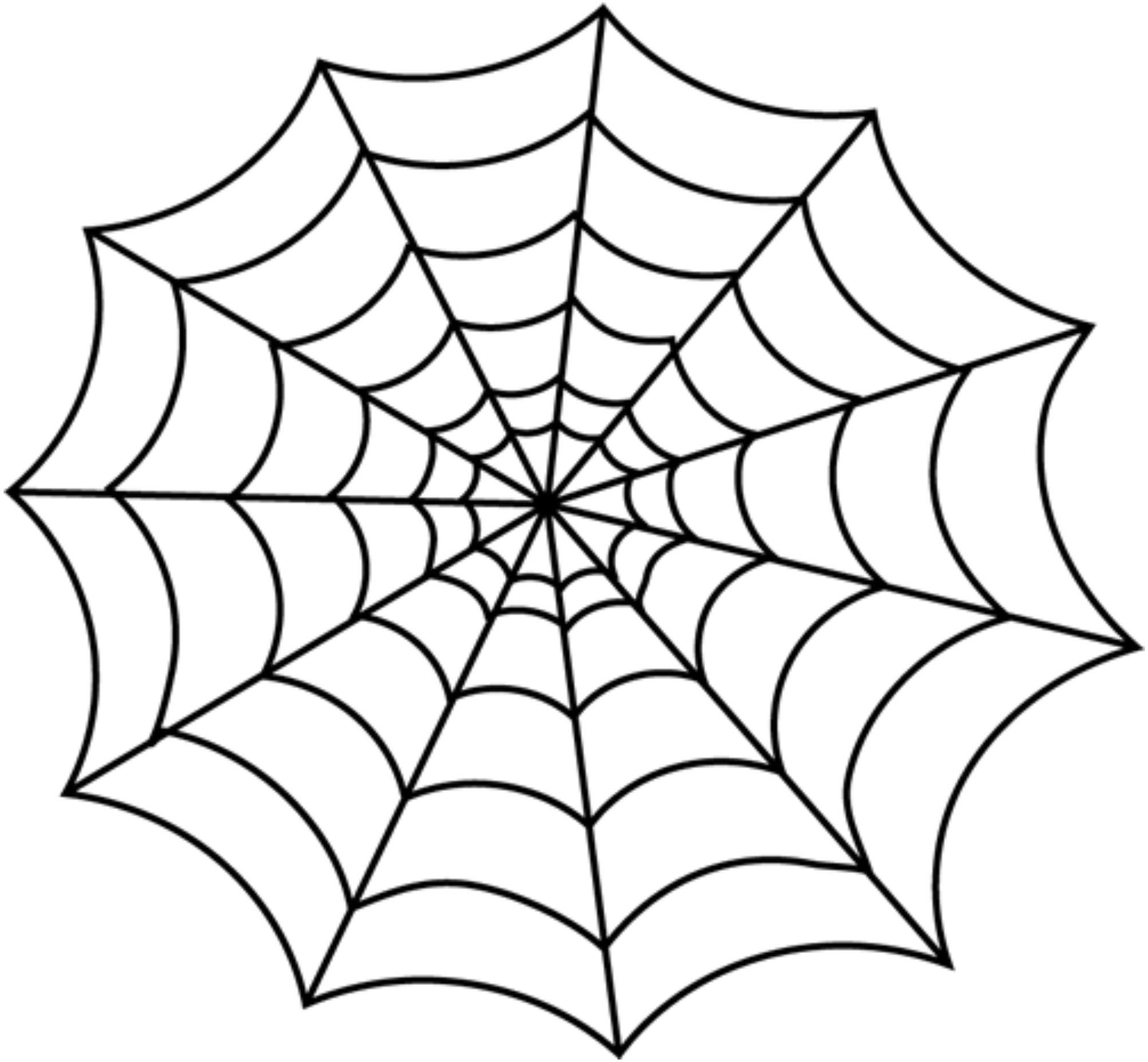
Graphs On Databases

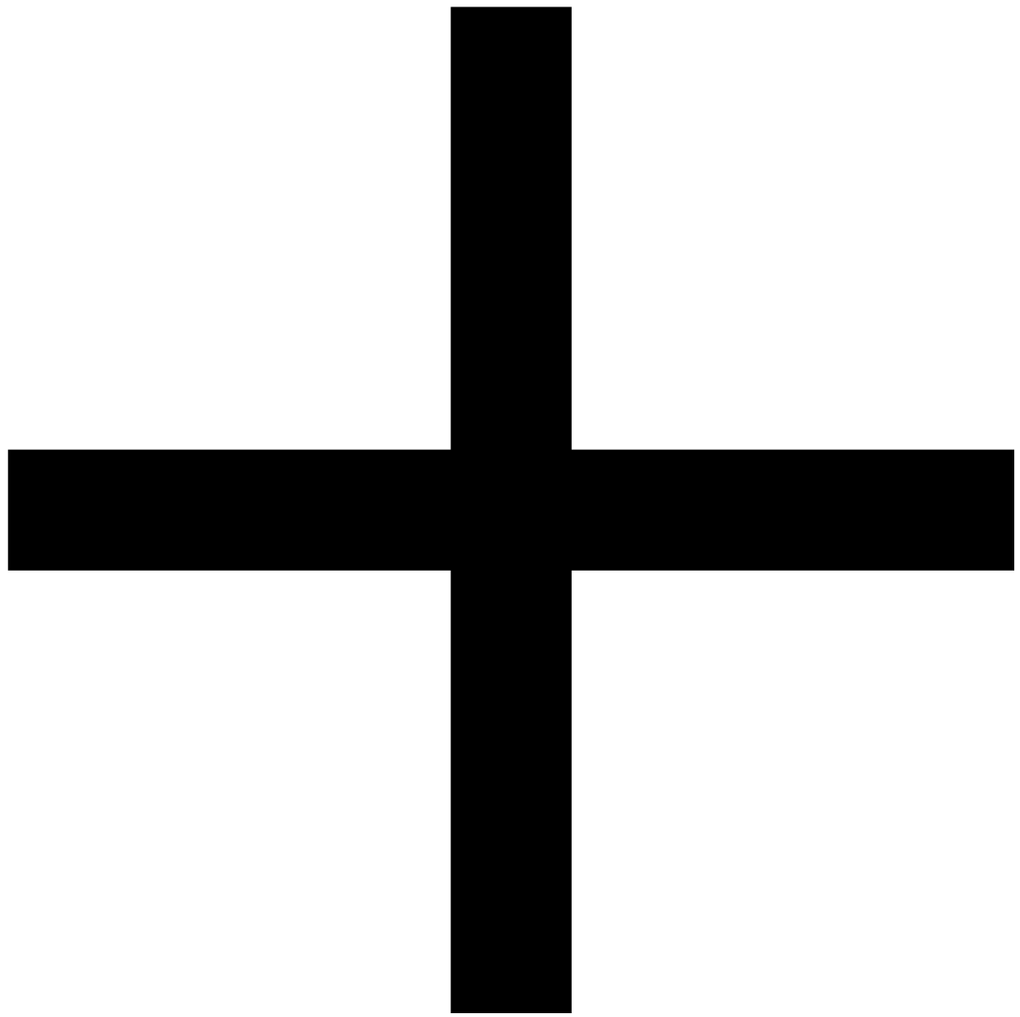
Alekh Jindal
Sam Madden
Mike Stonebraker

CSAIL, MIT







[redacted]

[redacted]

Jena

FlockDB

AllegeroGraph

TAO

Neo4j

DEX

Pegasus

HypergraphDB

Titan

Pregel

GraphBase

Twister

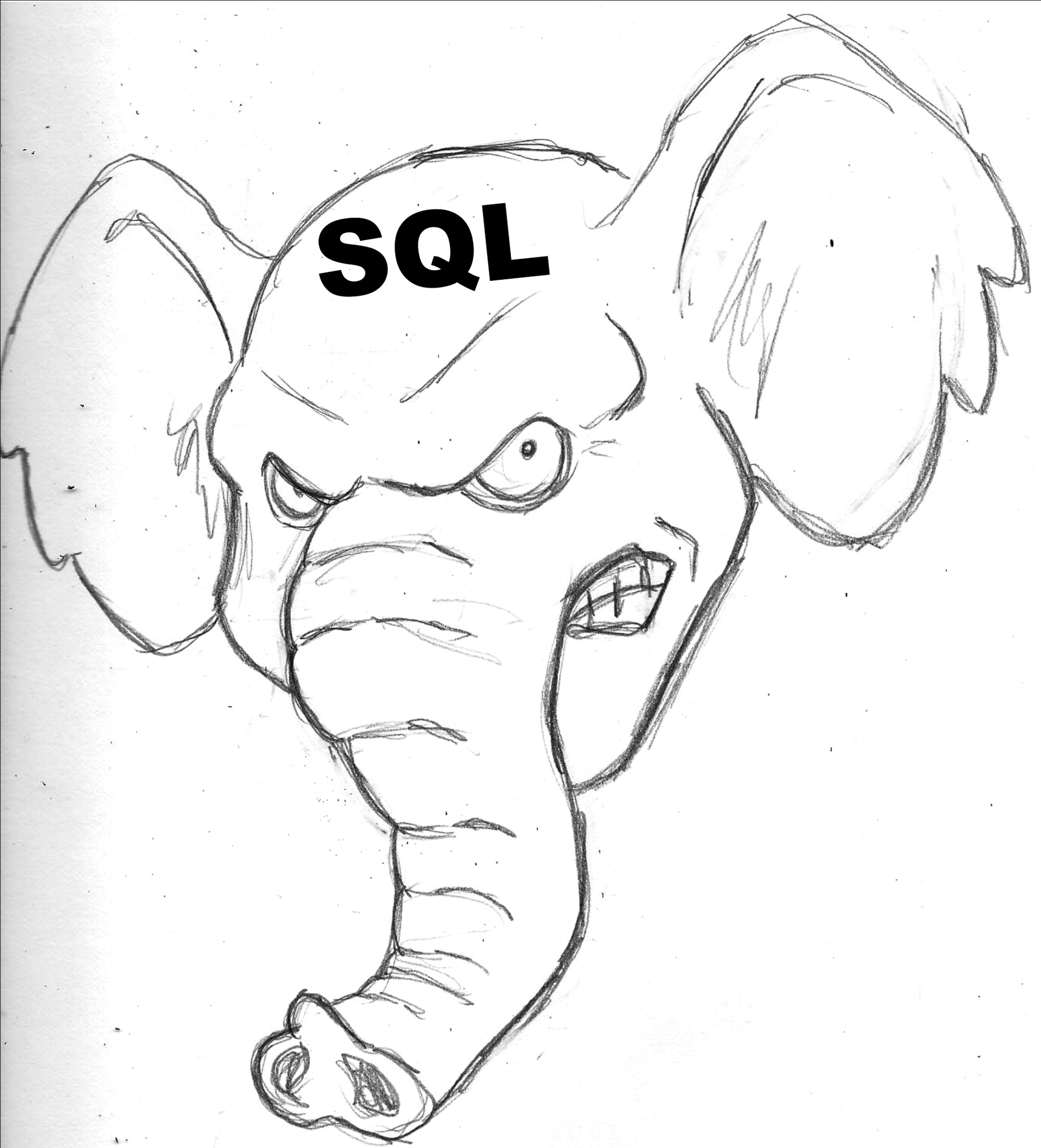
Giraph

Trinity

HaLoop

GraphLab

PrItr


SQL

?


Bata
VIDEO
FUJIFILM
EXR
Signs Of Life


VIDEO
Signs Of Life
John Van Hamersveld


Rene Mer

Can we have efficient graph analytics within a SQL Database?

Graph Analytics on SQL Databases

- Graph: set of nodes, set of edges
- Node table: nodeId and associated metadata
- Edge table: (from,to) nodeIds and associated metadata
- Undirected graph: two tuples per edge
- Node/Edge access: selection, projection on node and edge tables
- Graph traversal: successive joins between node and edge tables

Optimizations

- Number of Joins
  Parallel graph exploration

- Number of queries; round trips
  Nested queries; database handles the optimizations

- Data movement between server and client
  UDFs, Stored Procedures

- Database connections
  Keep connections alive between iterations

- SQL query performance
  Sort orders, indexes

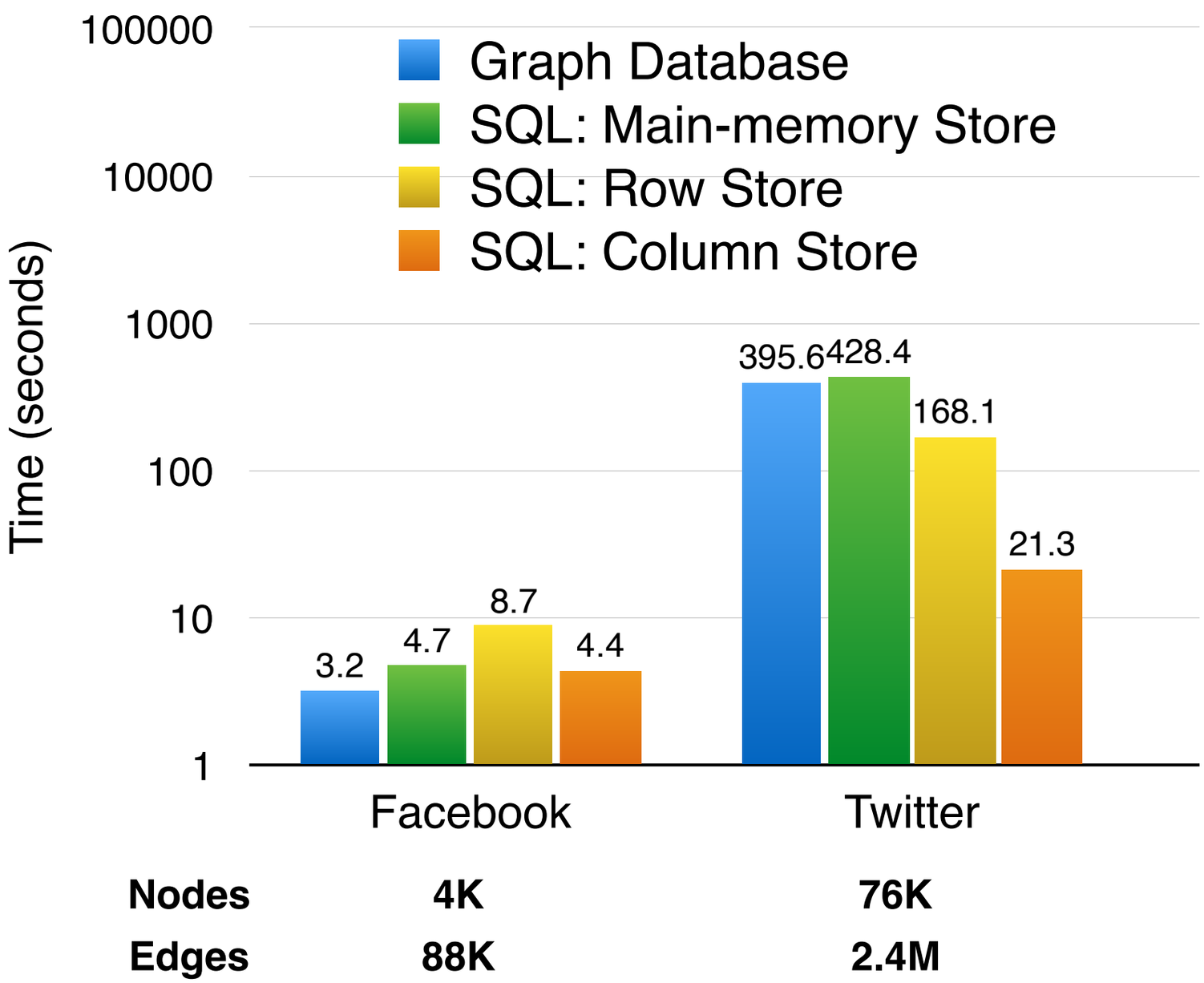
How does the performance look like?

PageRank

Nodes

Edges

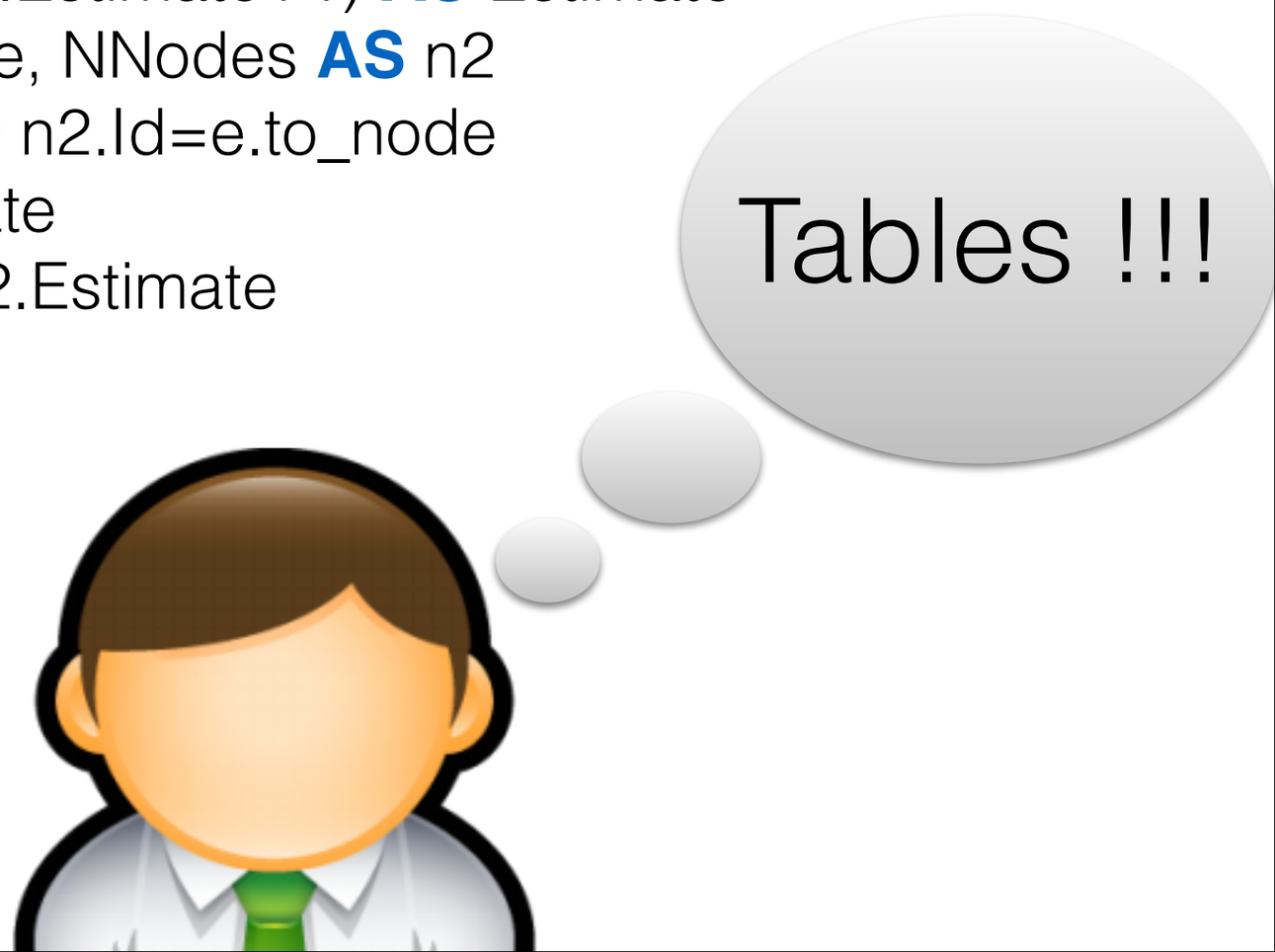
Shortest Paths



What about the query interface? Is SQL the right choice for graph queries?

Shortest Path in SQL

```
UPDATE NNodes AS nnode
  SET Estimate = new_nnode.Estimate, Predecessor = new_nnode.Predecessor
  FROM
    (SELECT temp.Id, temp.Estimate, edge.from_node AS Predecessor
      FROM NNodes AS nn, edge,
        (SELECT e.to_node AS Id, min(n1.Estimate+1) AS Estimate
          FROM NNodes AS n1, edge AS e, NNodes AS n2
          WHERE n1.Id=e.from_node AND n2.Id=e.to_node
          GROUP BY e.to_node, n2.Estimate
          HAVING min(n1.Estimate+1) < n2.Estimate
        ) AS temp
    WHERE nn.Id=edge.from_node
      AND edge.to_node=temp.Id
      AND nn.estimate=temp.estimate-1
    ) AS new_nnode
WHERE nnode.Id = new_nnode.Id;
```



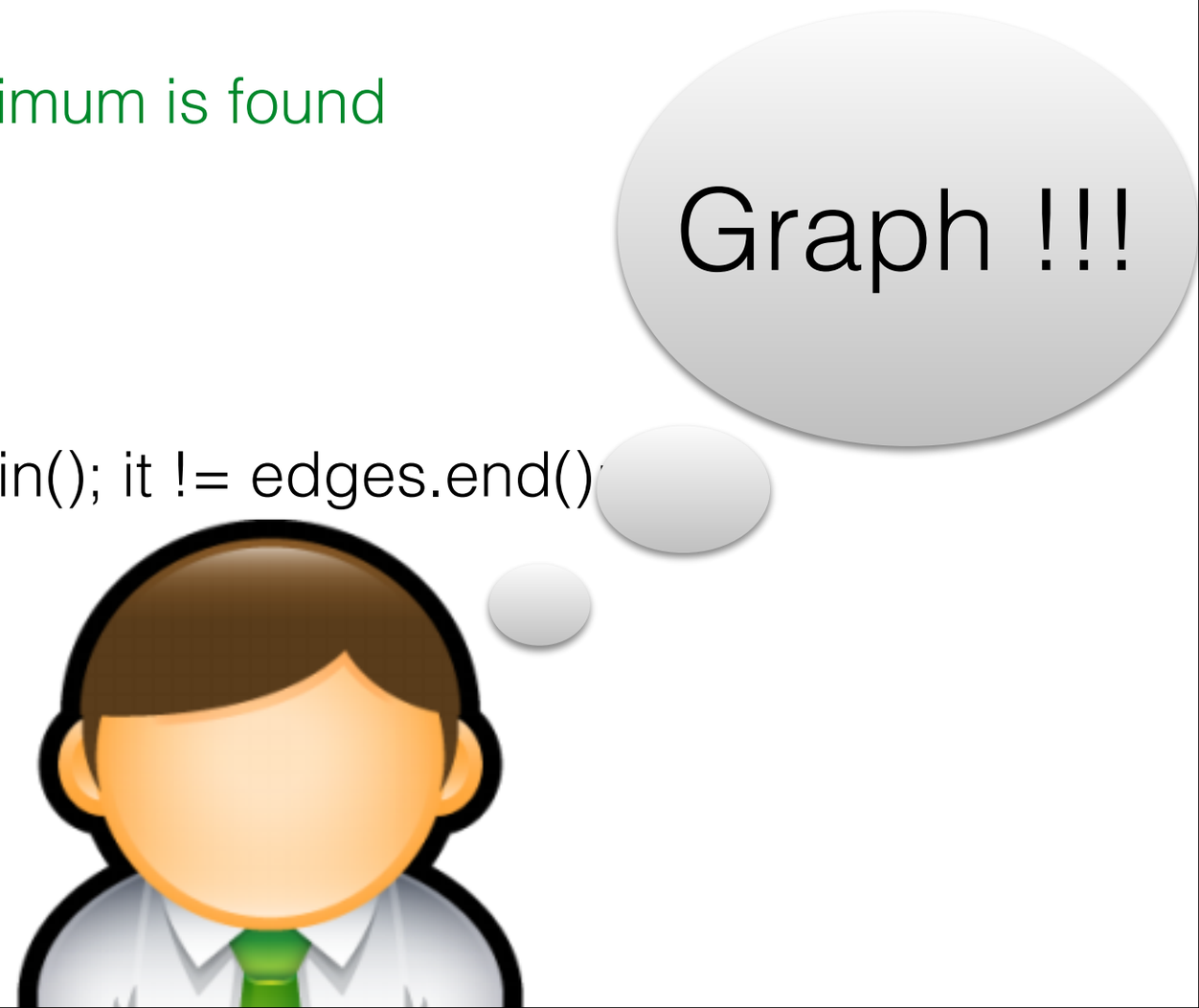
Shortest Path in Pregel

```
void compute(vector<vfloat> messages){

  // get the minimum distance
  vfloat mindist = id==START_NODE ? 0 : DBL_MAX;
  for(vector<vfloat>::iterator it = messages.begin(); it != messages.end(); ++it)
    mindist = min(mindist,*it);

  // send messages to all edges if new minimum is found
  vfloat vvalue = getVertexValue();
  if(mindist < vvalue){
    modifyVertexValue(mindist);
    vector<vint> edges = getOutEdges();
    for(vector<vint>::iterator it = edges.begin(); it != edges.end()
      sendMessage(*it, mindist+1);
  }

 // halt
  voteToHalt();
}
```



What about having a vertex-centric interface in a SQL Database?

Vertex-centric Interface in SQL Databases

- Idea: Map vertex-centric program execution to SQL queries in a SQL database

- The programmer specifies what happens on each vertex in the graph

- Vertices are executed as long as they are in active state or if they have an incoming message

- Exact same API as in Pregel, e.g. getting incoming messages, vertex value, vertex edges, etc.

Implementation Details

- Vertex (V), Edge (E), Message (M)
- The vertex programs are executed in parallel (super step) as UDFs in the SQL database
- Vertex Input: (V,E,M) for the vertex
- Vertex Output: outgoing M from the vertex
- A coordinator synchronizes between super steps, i.e. redistributes the messages from one super step to the next
- The coordinator stops when there are no more messages

Optimizations

- 3-way join, instead of 2-way
  Table unions in place of joins

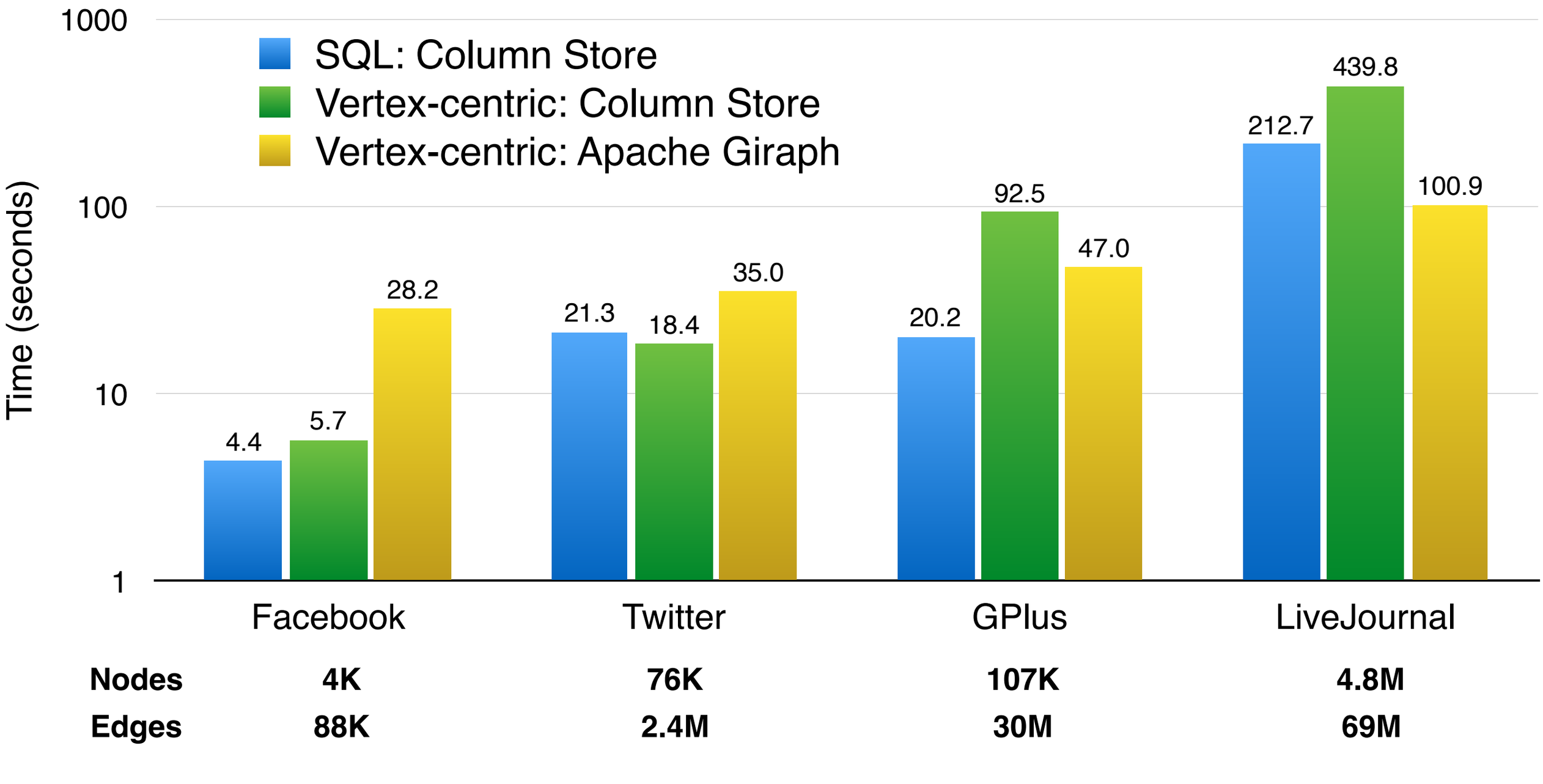
- UDF call overheads
  Batching several vertices in each UDF call

- Too many new messages in each super step
  Replace messages table, no in-place updates

- SQL query performance
  Sort orders, indexes

How does the performance look like?

Shortest Paths



Vertex-centric interface allows…

- Connected Components
- Random Walks with Restart
- Stochastic Gradient Descent
- Message Passing Algorithms
- Or, any other vertex-centric algorithm

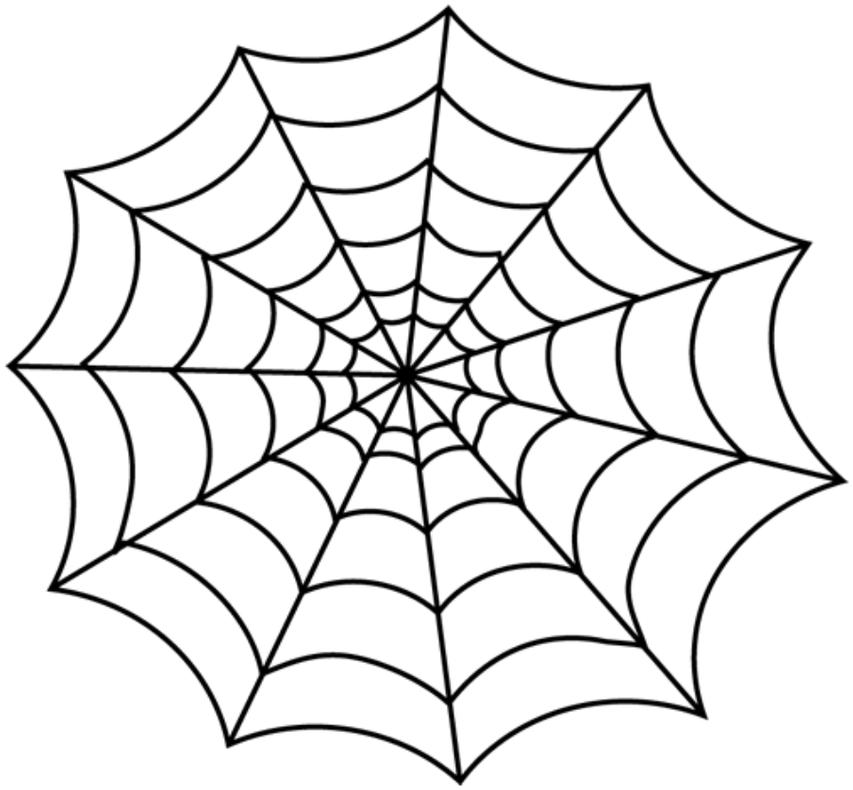
…. right within the SQL database system!

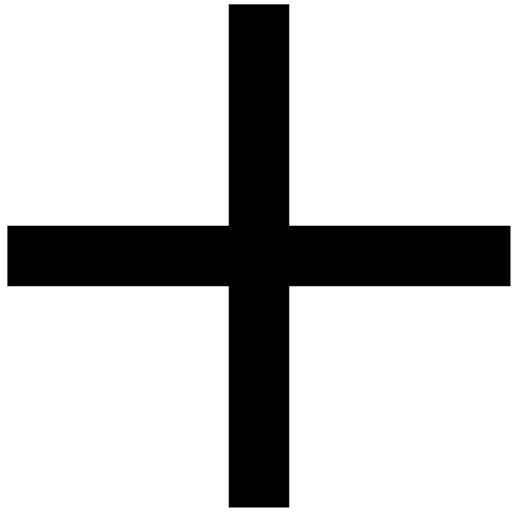
Summary

- Graph analytics can be mapped to relational queries (plus UDFs)

- SQL systems can offer very good performance over relational queries

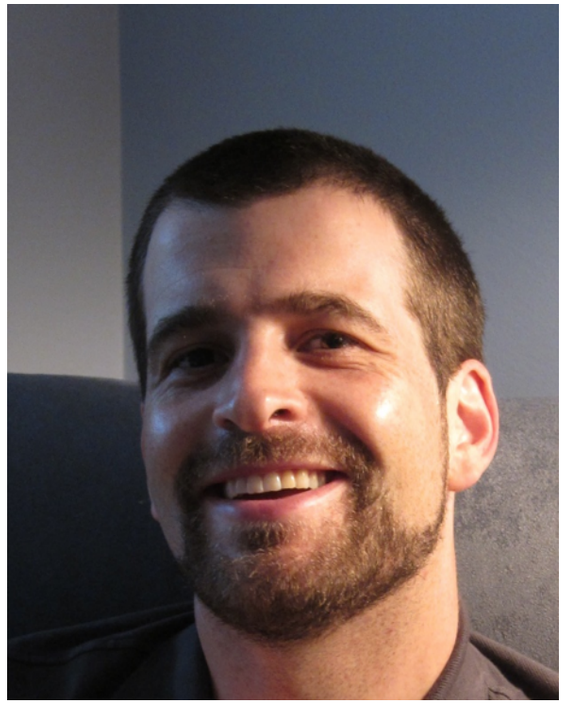
- We can extend SQL systems to provide more graph-natural query interfaces





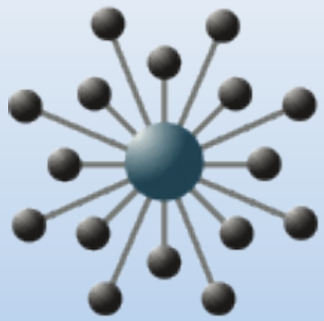

Rene Mensen

Team Members



Non-faculty Members

Non-faculty, non-postdoc …



ISTC
BIG DATA

Graphs On Databases

Alekh Jindal
Sam Madden
Mike Stonebraker

CSAIL, MIT