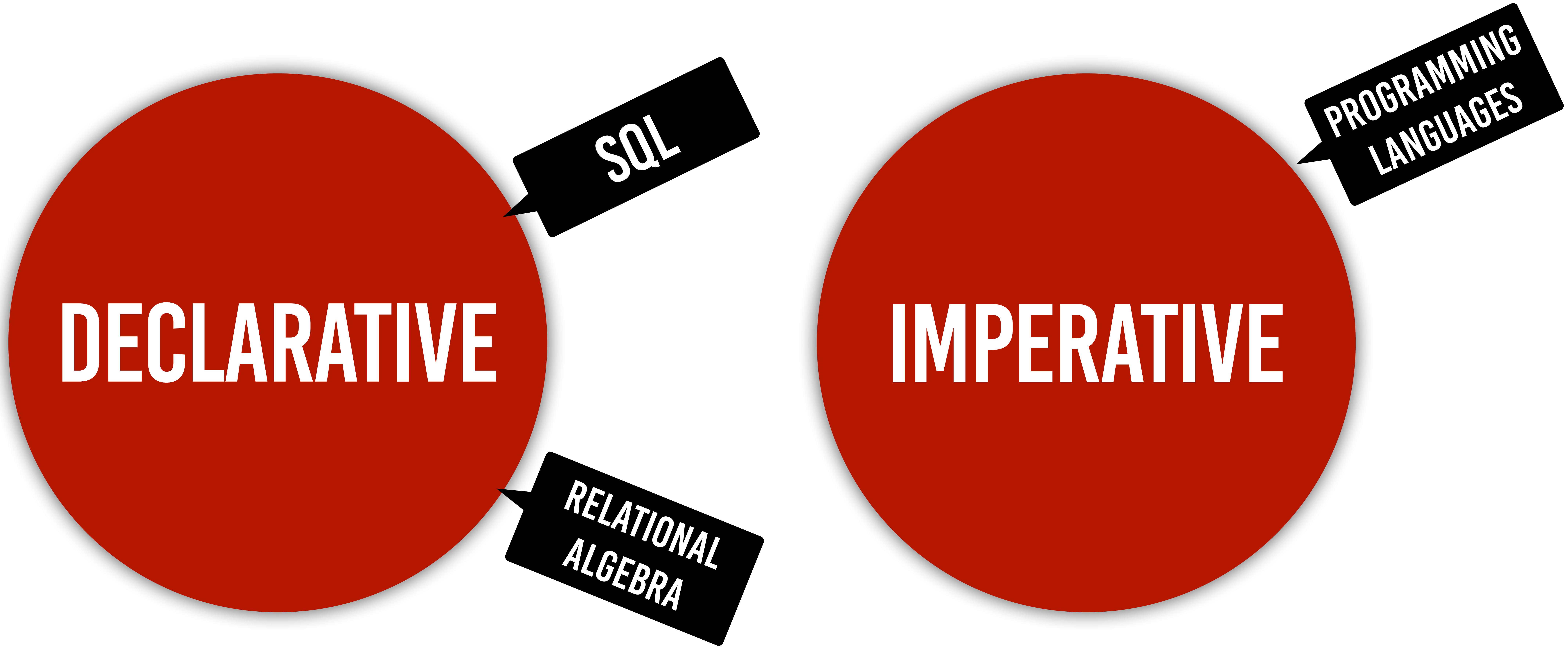
DESIGNING DATA-INTENSIVE APPLICATIONS

HOW WE WILL COMMUNICATE WITH DATA MODELS?



DECLARATIVE

```
SELECT * FROM animals WHERE family = 'Sharks';
```

IMPERATIVE

```
function getSharks() {
    var sharks = [];
    for (var i = 0; i < animals.length; i++) {
        if (animals[i].family === "Sharks") {
            sharks.push(animals[i]);
        }
    }
    return sharks;
}
```

DECLARATIVE QUERIES ON THE WEB

```
<ul>
    <li class="selected"> ❶
        <p>Sharks</p> ❷
        <ul>
            <li>Great White Shark</li>
            <li>Tiger Shark</li>
            <li>Hammerhead Shark</li>
        </ul>
    </li>
    <li>
        <p>Whales</p>
        <ul>
            <li>Blue Whale</li>
            <li>Humpback Whale</li>
            <li>Fin Whale</li>
        </ul>
    </li>
</ul>
```

```
li.selected > p {
    background-color: blue;
}
```

```
var liElements = document.getElementsByTagName("li");
for (var i = 0; i < liElements.length; i++) {
    if (liElements[i].className === "selected")
        var children = liElements[i].childNodes;
        for (var j = 0; j < children.length; j++) {
            var child = children[j];
            if (child.nodeType === Node.ELEMENT_NODE &&
child.tagName === "P") {
                child.setAttribute("style", "background-color:
blue");
            }
        }
    }
}
```

QUERY LANGUAGES

SQL

MAPREDUCE

CYPHER

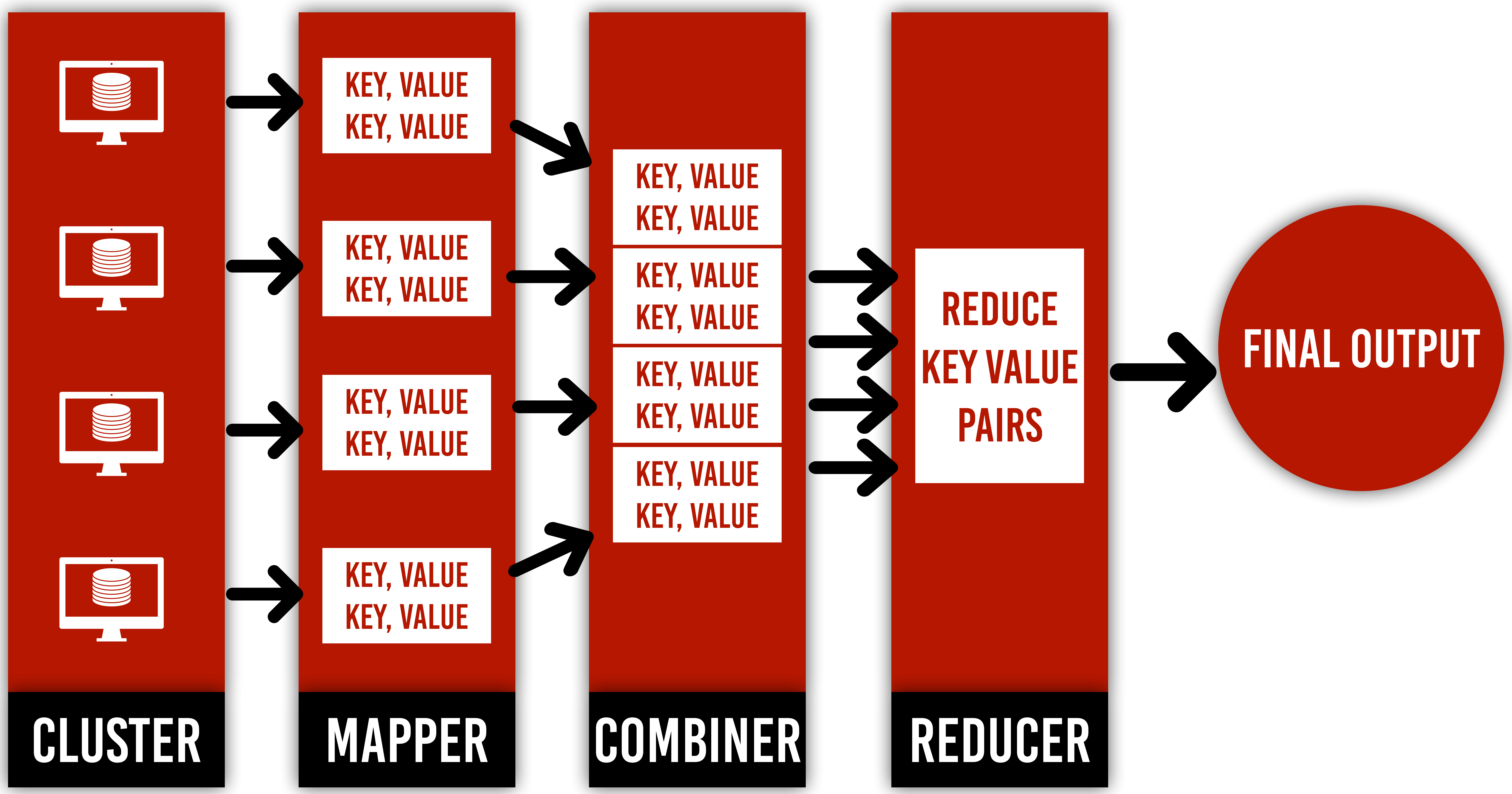
SPRAWL

MAPREDUCE

PROGRAMMING MODEL

FOR DISTRIBUTED EXECUTION ON A CLUSTER OF MACHINES

HOW MAPREDUCE WORKS?



IS MAPREDUCE DECLARATIVE OR IMPERATIVE??

```
SELECT date_trunc('month', observation_timestamp) AS
observation_month, ❶
       sum(num_animals) AS total_animals
FROM observations
WHERE family = 'Sharks'
GROUP BY observation_month;
```

MONTH	TOTAL_ANIMALS
2	290
3	12
4	45
12	23

```
db.observations.mapReduce(
    function map() { ❷
        var year  = this.observationTimestamp.getFullYear();
        var month = this.observationTimestamp.getMonth() + 1;
        emit(year + "-" + month, this.numAnimals); ❸
    },
    function reduce(key, values) { ❹
        return Array.sum(values); ❺
    },
    {
        query: { family: "Sharks" }, ❶
        out: "monthlySharkReport" ❻
    }
);
```

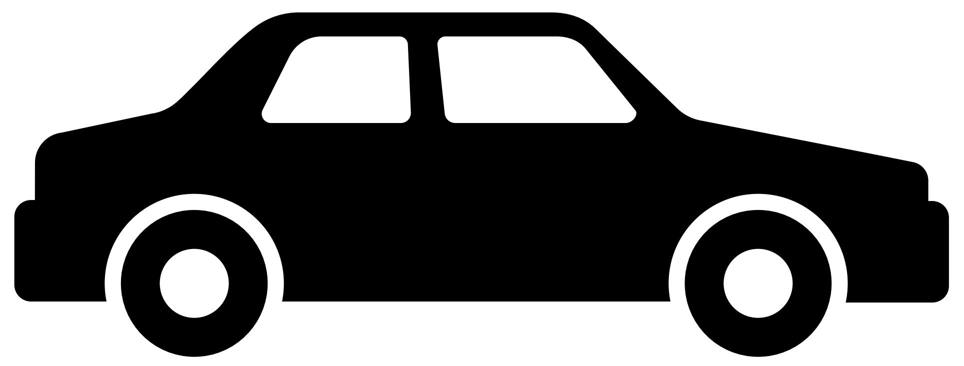
```
db.observations.aggregate([
    { $match: { family: "Sharks" } },
    { $group: {
        _id: {
            year:  { $year:  "$observationTimestamp" },
            month: { $month: "$observationTimestamp" }
        },
        totalAnimals: { $sum: "$numAnimals" }
    } }
]);
```

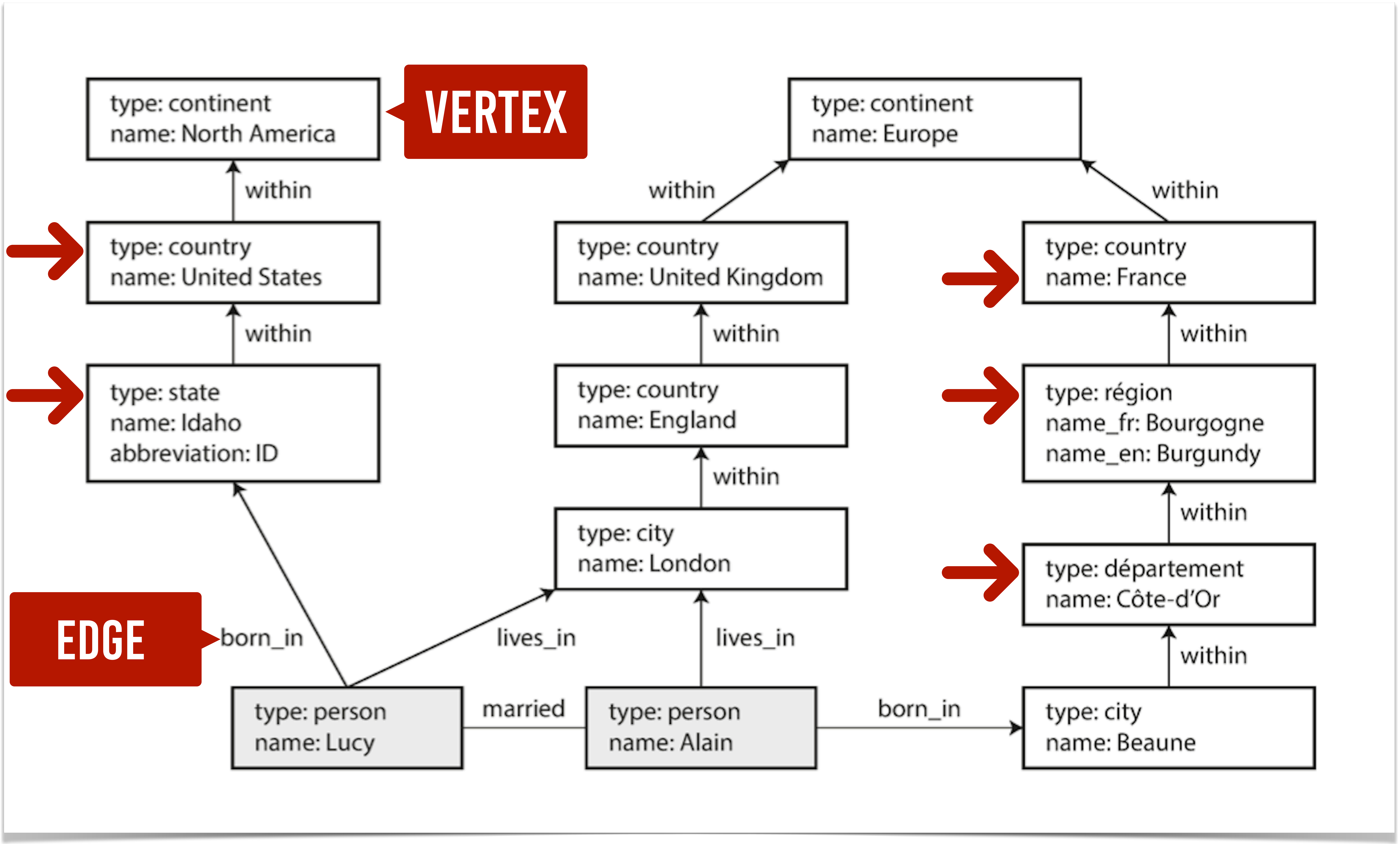
CYPHER

QUERY LANGUAGE FOR GRAPH DATABASES

WHAT IS GRAPH DATA MODEL?

ENTITY CALLED VERTEX LINK CALLED EDGE




type: continent
name: North America
VERTEX
type: continent
name: Europe
within
within
within
type: country
name: United States
type: country
name: United Kingdom
type: country
name: France
within
within
within
type: state
name: Idaho
abbreviation: ID
type: country
name: England
type: région
name_fr: Bourgogne
name_en: Burgundy
within
within
type: city
name: London
type: département
name: Côte-d'Or
EDGE
born_in
lives_in
lives_in
within
type: person
name: Lucy
married
type: person
name: Alain
born_in
type: city
name: Beaune

CYPHER

DECLARATIVE QUERY LANGUAGE

id

type

Edge

```
CREATE
  (NAmerica:Location {name:'North America',
type:'continent'}),
  (USA:Location      ame:'United States', type:'country'
}),
  (Idaho:Location    {name:'Idaho',          type:'state'
}),
  (Lucy:Person       {name:'Lucy' }),
  (Idaho) -[:WITHIN]->  (USA)  -[:WITHIN]-> (NAmerica),
  (Lucy)  -[:BORN_IN]-> (Idaho)
```

```
MATCH
  (person) -[:BORN_IN]-> () -[:WITHIN*0..]-> (us:Location
{name:'United States'}),
  (person) -[:LIVES_IN]-> () -[:WITHIN*0..]-> (eu:Location
{name:'Europe'})
RETURN person.name
```

neo4j

```sql
WITH RECURSIVE

  -- in_usa is the set of vertex IDs of all locations within the United States
  in_usa(vertex_id) AS (
      SELECT vertex_id FROM vertices WHERE properties->>'name' = 'United States' ❶
    UNION
      SELECT edges.tail_vertex FROM edges ❷
        JOIN in_usa ON edges.head_vertex = in_usa.vertex_id
        WHERE edges.label = 'within'
  ),

  -- in_europe is the set of vertex IDs of all locations within Europe
  in_europe(vertex_id) AS (
      SELECT vertex_id FROM vertices WHERE properties->>'name' = 'Europe' ❸
    UNION
      SELECT edges.tail_vertex FROM edges
        JOIN in_europe ON edges.head_vertex = in_europe.vertex_id
        WHERE edges.label = 'within'
  ),

  -- born_in_usa is the set of vertex IDs of all people born in the US
  born_in_usa(vertex_id) AS ( ❹
    SELECT edges.tail_vertex FROM edges
      JOIN in_usa ON edges.head_vertex = in_usa.vertex_id
      WHERE edges.label = 'born_in'
  ),

  -- lives_in_europe is the set of vertex IDs of all people living in Europe
  lives_in_europe(vertex_id) AS ( ❺
    SELECT edges.tail_vertex FROM edges
      JOIN in_europe ON edges.head_vertex = in_europe.vertex_id
      WHERE edges.label = 'lives_in'
  )

SELECT vertices.properties->>'name'
FROM vertices
-- join to find those people who were both born in the US *and* live in Europe
JOIN born_in_usa     ON vertices.vertex_id = born_in_usa.vertex_id ❻
JOIN lives_in_europe ON vertices.vertex_id = lives_in_europe.vertex_id;
```