GENERAL ASSEMBLY

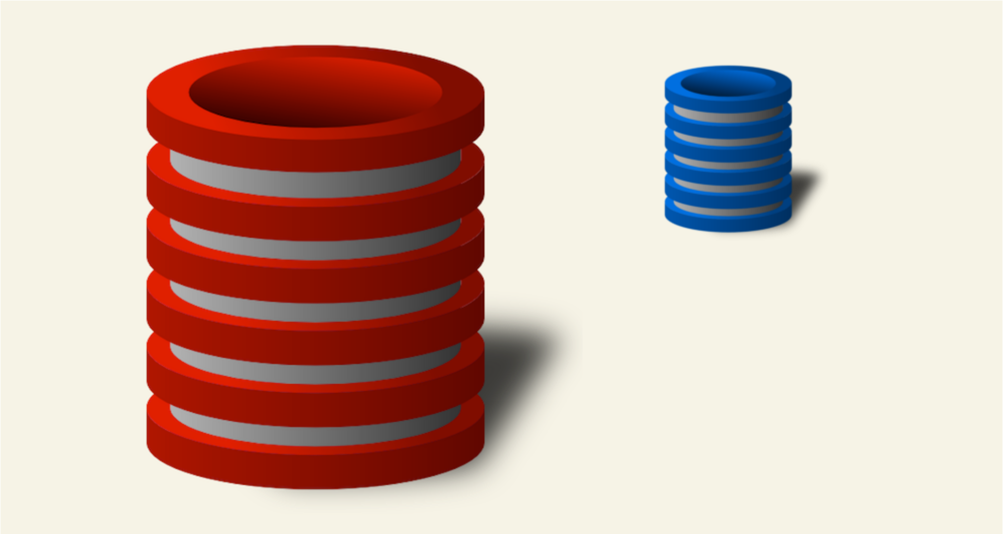
DATA SCIENCE

DATABASES / MAPREDUCE

I. DATABASES
II. SQL / NOSQL
III. MAPREDUCE
IV. IMPLEMENTATIONS OF MAPREDUCE
V. EXAMPLE OF MAPREDUCE

I. DATABASES

- An organized collection of data
- Organized using a schema (like a blueprint of a database)
- Organized into tables with different sets of data



WHY EVEN USE A DATABASE?

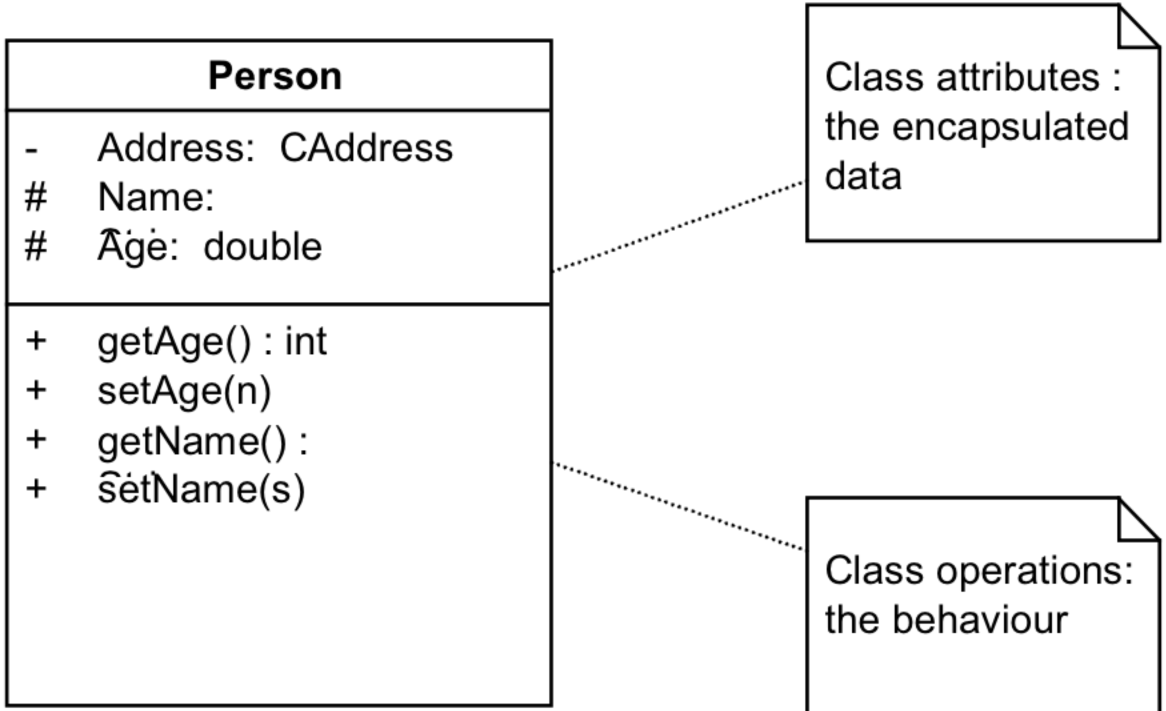
- Easy to store and more importantly, retrieve data
- Generally has a structured language for interacting with the data
- Reliable and **scalable**
- Access large amounts of data relatively quickly

WHY EVEN USE A DATABASE?

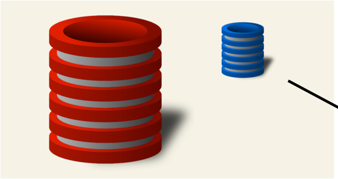
- Easy to store and more importantly, retrieve data
- Generally has a structured language for interacting with the data
- Reliable and **scalable**
- Access large amounts of data relatively quickly

HOW CAN YOU VISUALIZE A DATABASE?

UNIFIED
MODELING
LANGUAGE
UML™


Person
- Address: CAddress
Name:
Age: double
+ getAge() : int
+ setAge(n)
+ getName() :
+ setName(s)
Class attributes : the encapsulated data
Class operations: the behaviour

II. SQL / NOSQL

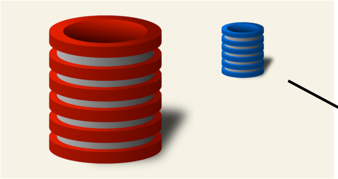


Relational

- Traditional rows and columns data
- **Strict** structure / Primary Keys
- Entire column for each feature
- Industry standard

NoSql

- No well defined data structure
- Works better for unstructured data
- Cheaper hardware
- Popular among Startups



Relational

Examples

- MySQL
- Oracle
- Postgres
- SQLite

NoSql

Examples

- MongoDB
- CouchDB
- Redis
- Cassandra

S
Q
L

Structured
Query
Language

Is a language for database communication

III. MAPREDUCE

"Nothing is particularly hard if you divide it into small jobs."

- Henry Ford

MapReduce is broken up into steps:

1. Map: produces key value pairs depending on the task
2. Shuffle/Sort (*optional*) sorts the key value pairs, could make new ones
3. Reduce combines the key value pairs into a single output

MapReduce is broken up into steps:

1. Map: produces key value pairs depending on the task
2. Shuffle/Sort (*optional*) sorts the key value pairs
3. Reduce combines the key value pairs into a single output

NOTE

Ideally the map function is run over a *cluster* of computers in *parallel*

Temperatures

- **Given**: several data tables consisting of data organized in a (City: Temperature) over several days
- **Goal**: Find the max temperature for each city

File 1

New York City: 32
Chicago: 22
New York City: 36
Miami: 67
…

File 2

Miami: 77
New York City: 32
New Haven: 29
…

File 1

New York City: 32
Chicago: 22
New York City: 36
Miami: 67
Chicago: 21
New Haven: 32

File 2

Miami: 77
New York City: 32
New Haven: 29
Chicago: 29
Miami: 78
Chicago: 44

File 1

New York City: 32
Chicago: 22
New York City: 36
Miami: 67
Chicago: 21
New Haven: 32

map →

(NYC: 32,
CHI: 22,
NYC: 36,
MIA: 67,
CHI: 21
NH: 32)

File 2

Miami: 77
New York City: 32
New Haven: 29
Chicago: 29
Miami: 78
Chicago: 44

map →

(MIA: 77,
NYC: 32,
NH: 29
CHI: 29,
MIA: 78,
CHI: 44)

MAPREDUCE: EXAMPLE

File 1

New York City: 32
Chicago: 22
New York City: 36
Miami: 67
Chicago: 21
New Haven: 32

$\xrightarrow{\text{map}}$

(NYC: 32,
CHI: 22,
NYC: 36,
MIA: 67,
CHI: 21
NH: 32)

File 2

Miami: 77
New York City: 32
New Haven: 29
Chicago: 29
Miami: 78
Chicago: 44

$\xrightarrow{\text{map}}$

(MIA: 77,
NYC: 32,
NH: 29
CHI: 29,
MIA: 78,
CHI: 44)

$\xrightarrow{\text{sort}}$

(NYC: [32,32,36],
CHI: [21,22,29,44],
MIA: [67,77,78],
NH: [29,32])

MAPREDUCE: EXAMPLE

File 1

New York City: 32
Chicago: 22
New York City: 36
Miami: 67
Chicago: 21
New Haven: 32

File 2

Miami: 77
New York City: 32
New Haven: 29
Chicago: 29
Miami: 78
Chicago: 44

map →

(NYC: 32,
CHI: 22,
NYC: 36,
MIA: 67,
CHI: 21
NH: 32)

map →

(MIA: 77,
NYC: 32,
NH: 29
CHI: 29,
MIA: 78,
CHI: 44)

sort

(NYC: [32,32,36],
CHI: [21,22,29,44],
MIA: [67,77,78],
NH: [29,32])

reduce

(NYC: 36,
CHI: 44,
MIA: 78,
NH: 32)

MapReduce Benefits

- Extremely scalable: algorithm for a MB of Data will work for a PetaByte of Data
- Many implementations with documentation exist for Java, C, Python, etc..
- Relatively fast compared tostraight though processing

MECHANICAL TURK ALERT!!!

You are all about to become a MapReduce cluster. All of you will be either a:

- Mapper
- Sorter
- Reducer

- Mappers

File 1
big data big big data

→ (big: 1), (data: 1), (big, 1), (big: 1), (data: 1)

- Sorters
- Reducer

- Mappers
- Sorters
- Reducer

File 1
big data big big data

→

(big: 1),
(data: 1),
(big, 1),
(big: 1),
(data: 1)

(big: 1),
(data: 1),
(big, 1),
(big: 1),
(data: 1),
……..

→

(big: [1, 1, 1, 1, …]),
(data: [1, 1, 1, …]),
(sql: [1, 1, 1, …]),
……..

- Mappers

File 1

big data big big data

→

(big: 1),
(data: 1),
(big, 1),
(big: 1),
(data: 1)

- Sorters

(big: 1),
(data: 1),
(big, 1),
(big: 1),
(data: 1),
……..

→

(big: [1, 1, 1, 1, …]),
(data: [1, 1, 1, …]),
(sql: [1, 1, 1, …]),
……..

- Reducer

(big: [1, 1, 1, 1, …]),
(data: [1, 1, 1, …]),
(sql: [1, 1, 1, …]),
……..

→

(big: 16),
(data: 22),
(sql: 2),
……..

IV. IMPLEMENTATIONS OF MAPREDUCE

IMPLEMENTATIONS OF MAPREDUCE

- Creator: Apache
- Implemented in: Java
- Has API for python
- Open source

- Creator: UC Berkley
- Owned by: Apache
- Implemented in: Java
- Built in API, ML, Graphing capabilities

- Creator: Nokia
- Implemented in: Python
- Open source on Github

V. EXAMPLE OF MAPREDUCE