MapReduce vs. DBMS

NoSQL

datafile — text, binary

— load file into memory
partial file

cursor →
iterator

NAME	AGE

rows
= records
(fixed-length)

cursor c = file.begin()
for ( ) {
    string r = c.getField2()
    c.nextRecord()
}

sort file by ~~field~~ field 2 (age)

Out-of-core
external algorithm

file
64GB

I/O

Chunk$_1$ } 2GB 2GB
}

Sorted chunk

⋮

Chunk$_n$

Sorted c_1
Sorted c_2

⋮

2GB

2GB

Sorted

1 computer

n computers N disks] log number of steps

sorting

"easy"

database is read only

in the face of updates

DB today ⟹ sort

Indexes / Indices

index   records

| 1 |
| 4 |
| 8 |
| 2 |
| 3 |
| 6 |
| 7 |

binary tree

```
        [3]
       /    \
     [2]    [5]
     /      /   \
   [1]    [4]   [6]
                  \  add
                  [7]
```

```
[1]
   \
   [2]
      \
      [3]
         .
          .
           .
```

balanced
binary tree

RB trees

$log(n)$

_____

- locality
- space consumption

```
       [ ]
      /   \
    [ ]   [ ]
    / \   / \
  [][] [][]
```

```
||||||||
```

HDD

hard
drive disk

$+ \approx 1$
$- \approx 0$

disk latency
- rotational
  latency ①
- seek latency

tracks

15,000
RPM

dust   media

SSD

solid state disk
aka Flash

minimizing
seek
latency

B⁺-tree

keys

3	5			

<3 $=3$ ≥ 5

1	2	

3	4	

5	6	7

values

locality optimal (minimizes
                  arises & seeks)
range queries "free"
(balanced)

secret
ugliness of
FLASH.

SSD — persistent
stores

① fixed
"read-write" cycles
    wear

    wear
    -leveling

② 128K writes

"database"

SSN	NAME	AGE

SSN	NAME	SALARY

SSN	NAME	AGE	SALARY

join ordering

while
while

domain-
specific language

"PL for Database"

— easy / readable
— flexible

PBE / QUEL

Codd   SEQUEL

               SQL
normal forms
               structured query language
relational model
    relational calculus   projection
                                        π
        ⋈   SELECT (✗)
                       FROM A,B
    tables   WHERE A.NAME = "Joe"
row   B.NAME = "Joe"
                       AND B.SALARY
                                > $10

           declarative

query optimization
  query plan

      — partitioning   ▢▢▢▢▢
      — joins
      — selectivity (# returned
                       by query)
           statistics
              tracking

      — indizes

INDEX ALL THE THINGS

| Name | Age | SSN | Favorite Color | | | | | K fields |

DBAs

TPC/C

Parallelization
1. time
2. space

COTS hardware

Commodity
off-the-shelf

indices — space on disk