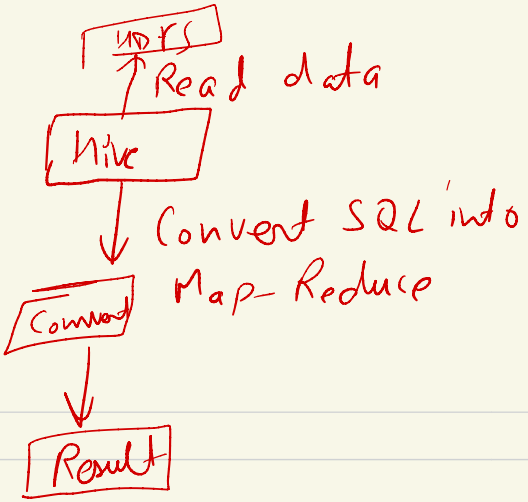
Why Hive?

- Simple to Use
- Build on top of Hadoop
- Typical SQL kind of framework
  - logic will be converted in Map-Reduce Code

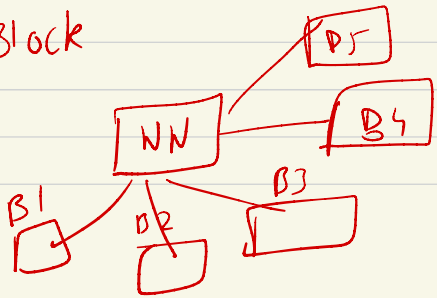


Hadoop

1) Hdfs → file storage
2) Map-Reduce → Processing
3.) Yarn → Resource Mngr

1TB ⇒ Block

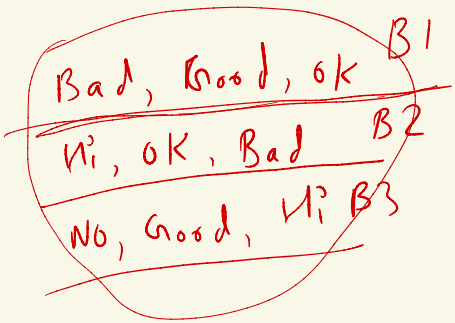
3 GB in Local Laptop

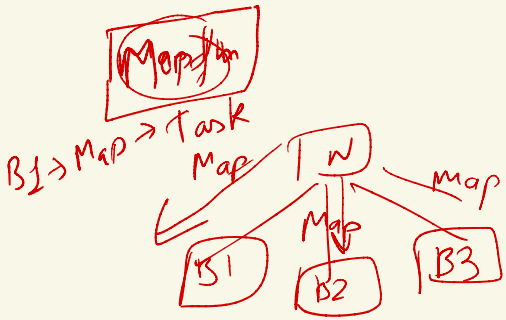


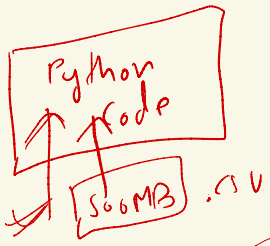
→ 4GB + 4GB

↳ external

Vertical Scaling







M1 → Bad : 1, Good : 1, ok : 1 — R1

M2 → Hi : 1, ok : 1, Bad : 1 → R2

M3 → NO : 1, Good : 1, Hi : 1 → R3

Map → Key, Value (output of Map)

Reduce → R1 + R2 + R3

Sort - Shuffle

hash("Bad") → 1 → Rd2

Rd1 0

Rd2 1

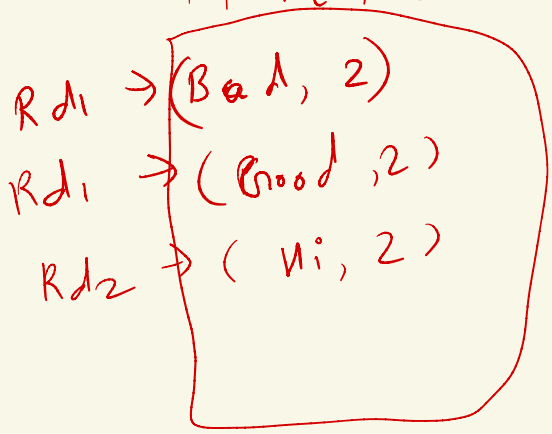
Rd3 2

Bad : [1, 1] Rd2

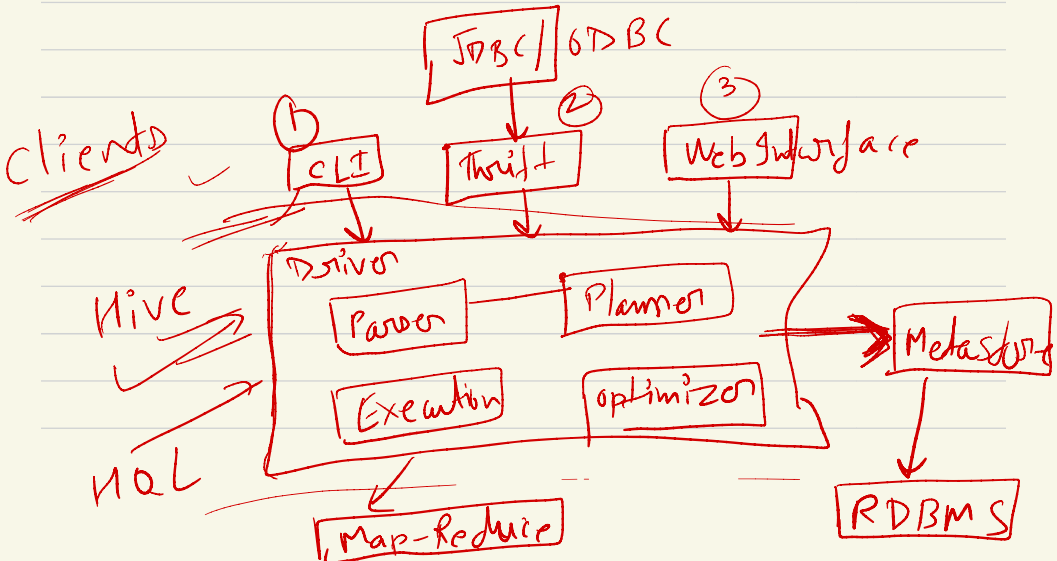
Good: [1, 1]

hash("Hi") → 0

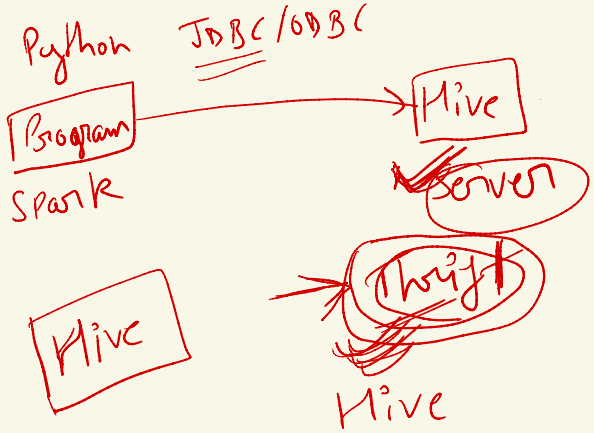
Hi : [1, 1] → Rd1



16 Reducer



HDFS



→ Request - 1 → Driver

Driver
- ↳ Controller for HQL statements
- ↳ Creates sessions for query
- ↳ maintain lifecycle of HQL
- ↳ maintain metadata for execution
- ↳ Collects output and display

Sql → Syntax & Symentics
- ↳ execution plan
- ↳ Optimization

> Processing

Parsing / Compilation

↳ Syntax check
↳ Execution plan
↳ Steps to get the output
↳ Raise Compile time Errors

Optimize

↳ Compare execution plans

NQL → E1, E2, E3 } Select * from Tbl where = group by = (sub query)

↳ Calculate Cost

group by → Where
where → E1 → Cost 1
group by → E2 → Cost 2

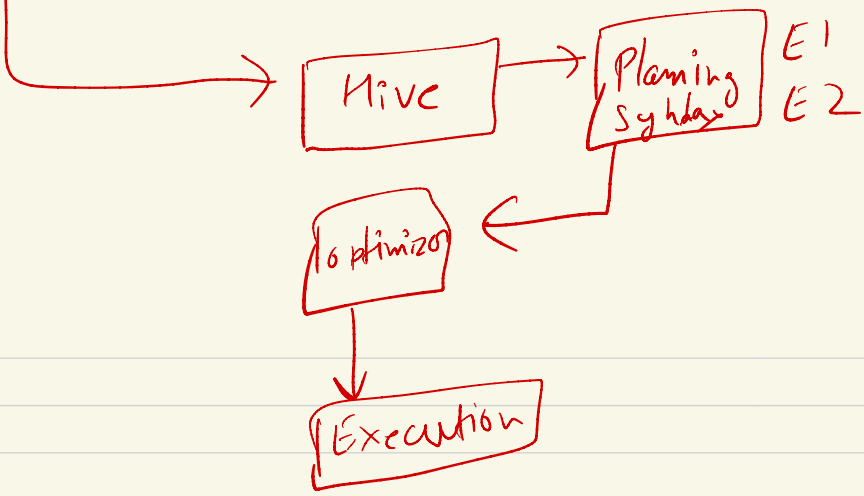
↳ Execution plan
DAG

↳ Try to place or Combine steps or methods together

Execution plans & DAG

department, count(x)

HQL
```
Select * from employee
where location = "INDIA"
group by department;
```



employee → 200K

E1 → where, group by } C1

E2 → group by, where } C2

C1 > C2

C1 < C2

line1 = "Hi, Bad, Good"

one → many

flat_map(line1)

R1 → {"Hi", "Bad", "Good"}

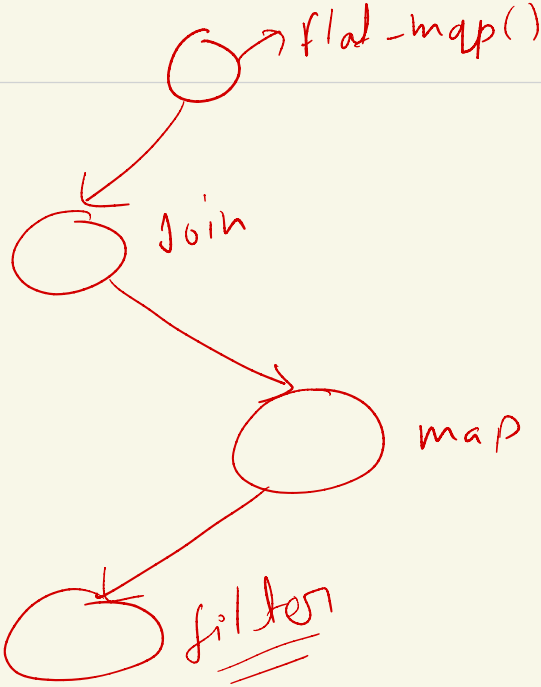
map(R1)

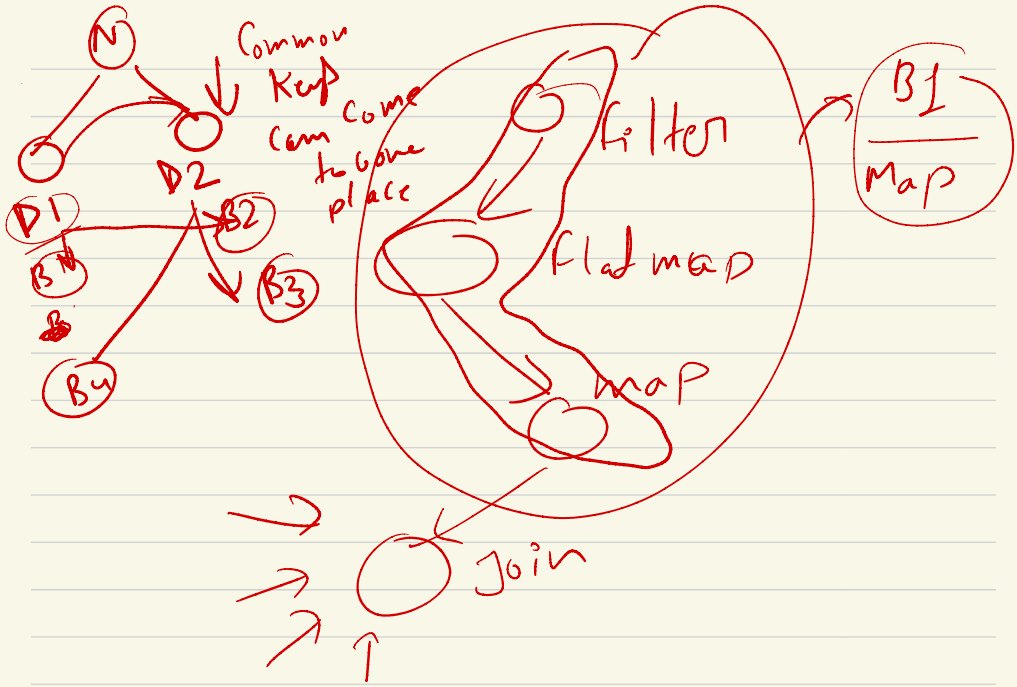
R2 → Hi:1, Bad:1, Good:1

Joins, group by

- flat_map(?)
- Join()
- map()
- filter()

} Transformation


flat-map()
Join
map
filter


N
Common Key
Come
com
to come
place
D2
D1
B2
B3
Bn
B1
filter
flat map
map
B1
Map
Join

Map1

DB1

B1 → id, salary

1,	1000
2,	2000
1,	5000

DB2 Map

B2 → id, Salary

1,	1000
3,	5000
1,	2000

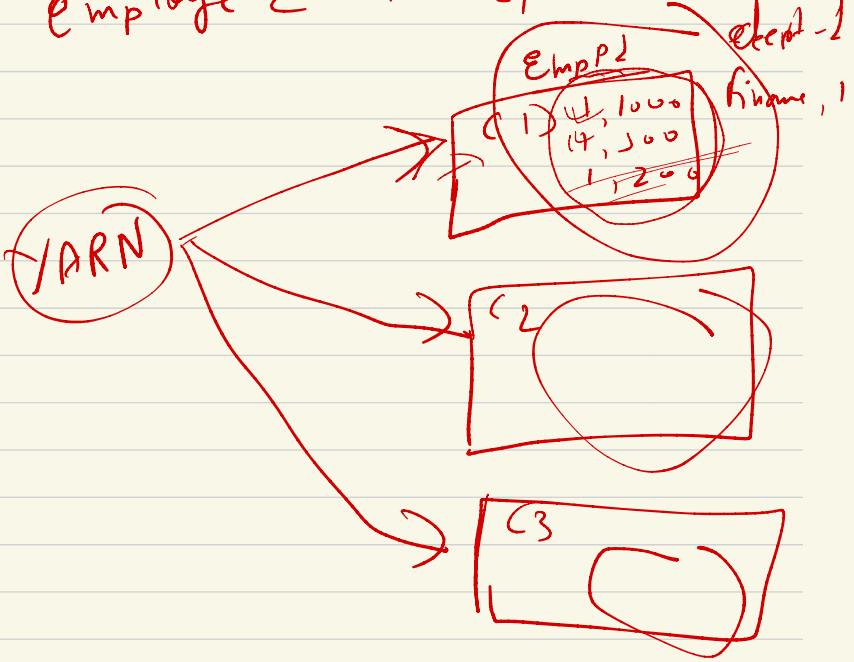
B3 → Dept_name, id

finance,	1
IT,	2

B4 → Dept name, id

OPS, 3

Employee & Department



HQL

↓ (Driver)

Parsing / Compilation / Planning

Syntax, Execution Plan

↓

Optimizer

Pick minimal ← Execution

↓

Execution

Executing ← Actual Execution

Metadata [Internal, External

Databases

MySQL → default database

Create table

→ Hive

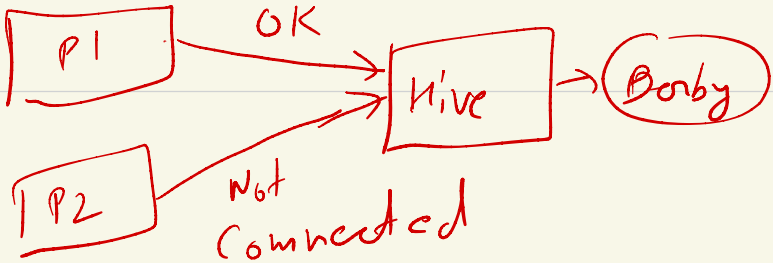
table_abc

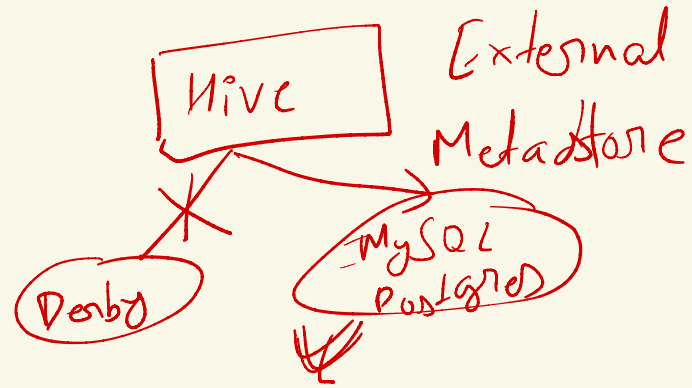
{ name string

salary int }

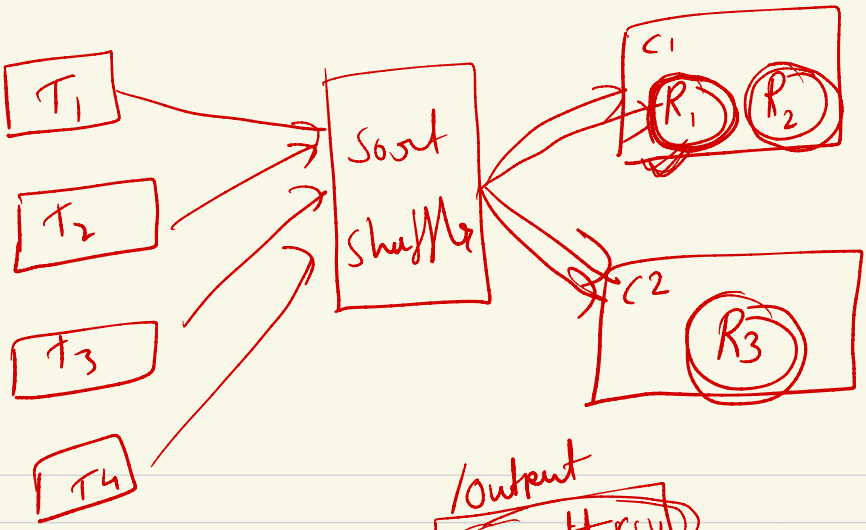
→ Derby (RDBMS)

(not more than 1 con.





- External DB
- Metadata Backup
- More Multiple Concurrent Connection
- Expose to external use cases



/Output

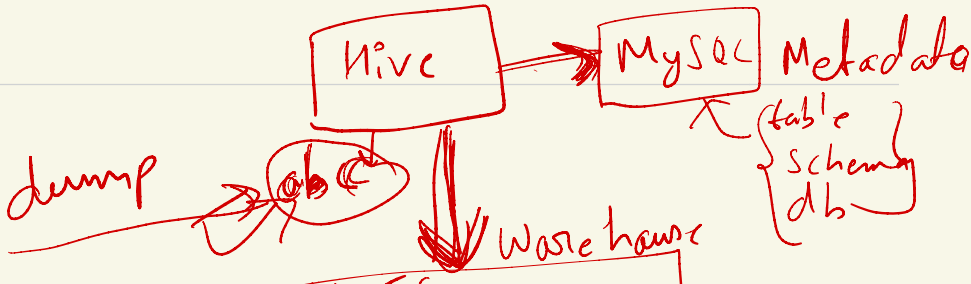
~~Result.csv~~

/Output

Part1.csv

Part2.csv

Part3.csv



→ Warehouse → store

B1 → Input (One Line) → Mapper → Output (Key, Value)

sort-shuffle → Reduce (Key, Value) (Key, Value)

mapper ( )

{

For (0 till last line)

{

(B1) ① ②

① "Hi, Hello, Bad"

→ Hi : 1, Hello : 1, Bad : 1

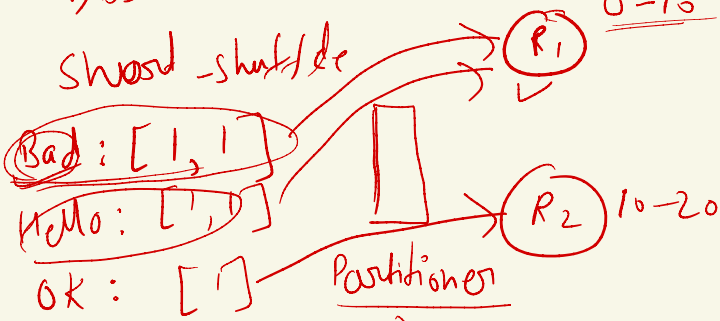
② "ok, Hello, Bad"

ok : 1, Hello : 1, Bad : 1

Map → (K, V) → Reduce

Disk

Shared-shuffle



Part2.csv

OK : 1 } R2

Part1.csv

Bad : 2
Hello : 2 } R1

hash(Bad) = 5

hash(hello) = 9

hash(OK) = 15

employee

d1 → Read

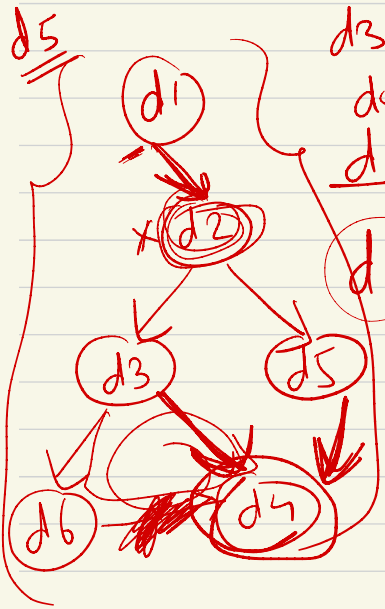
d2 → filter (d1)

d3 → derive (d$ salary * 2))

d5 → derive (d2)

d4 → join (d3, d5)

d6 → group (d3)



DAG → Directed Acyclic Group