FULL TEXT SEARCH

- Search feature

"good work culture"

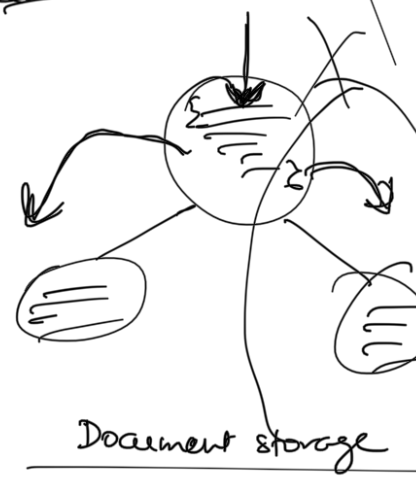
posts

id	user_id	content	ts	. . .

```
SELECT * FROM posts
WHERE content LIKE "%good work culture%"
LIMIT 10
```

Slow ←

O(N) * O(substring check)



Document storage

NoSQL

Key, value

CF

{ postid ⇒ "content" }

100 million

good work culture

↳ 5 posts

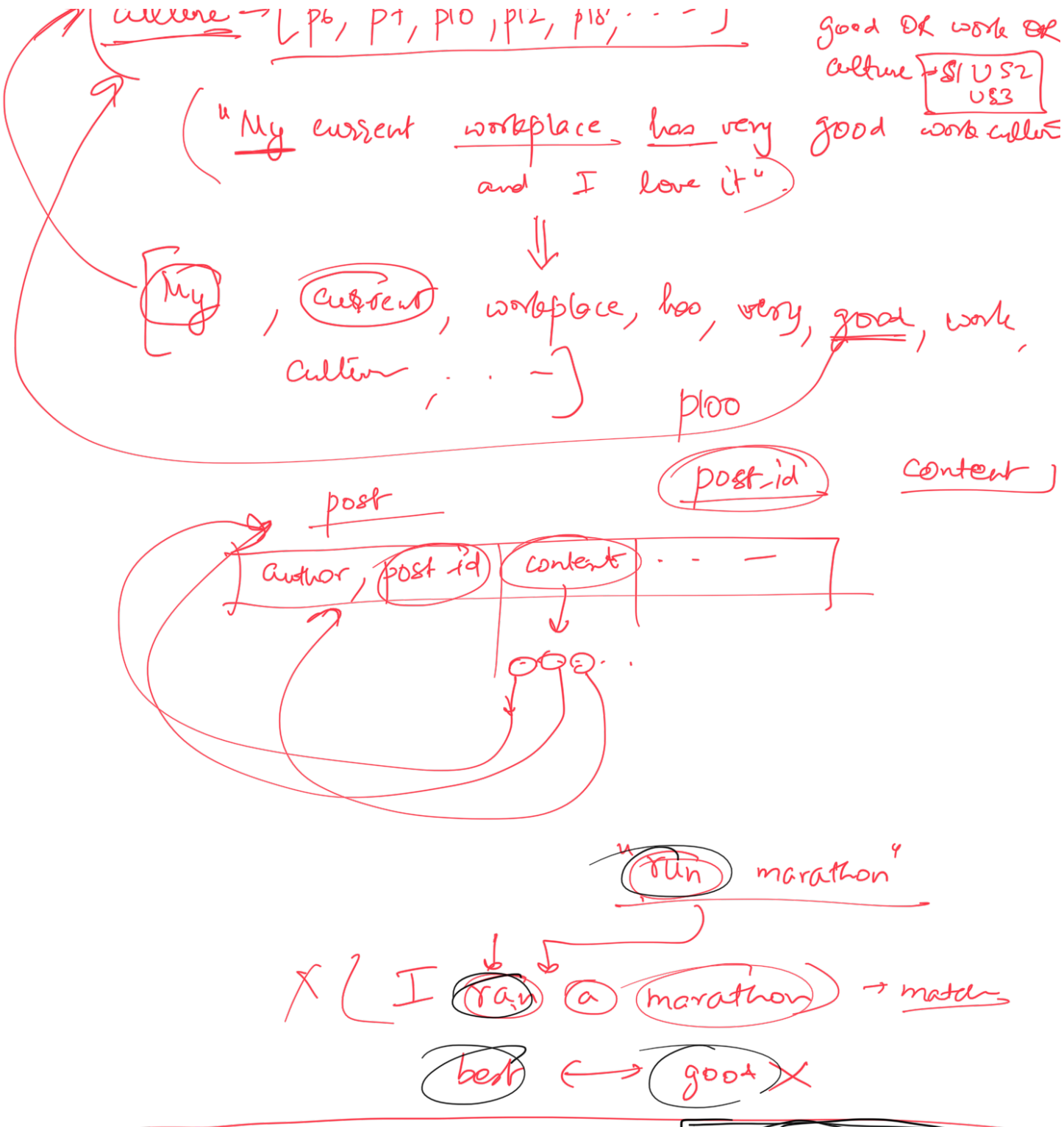
Inverse index :

good → [p1, p5, p10, p15 . . . p100]

work → [p5, p6, p8, p10, p12, p16, . . .]

S1 ∩ S2 ∩ S3

word → post



Full text search

- Elastic Search
- Solr

↓

Apache Lucene

1. Word elimination

→ ["a", "an", "the", "of", "on", "by" ...]

Special &

(2) Tokenisation → Break document/post into words

["ran", "marathon", "last", "weekend"]

(3) Stemming / Lemmatisation.

ran → run

ran, running, run → run

["run", "marathon", "last", "weekend"]

best, better, good → good

(4) Stop words → (abusive words, controversial)

freq-map

"recieve" → "receive"

⋮

("recieve", "receive") (1)

Spellchecker

↳ correct

re → few 100

ve → few 100

Suggest right spelling

("apple", "abple")

Anshuman → Antihuman, edit dist = 2

add a letter → (1)

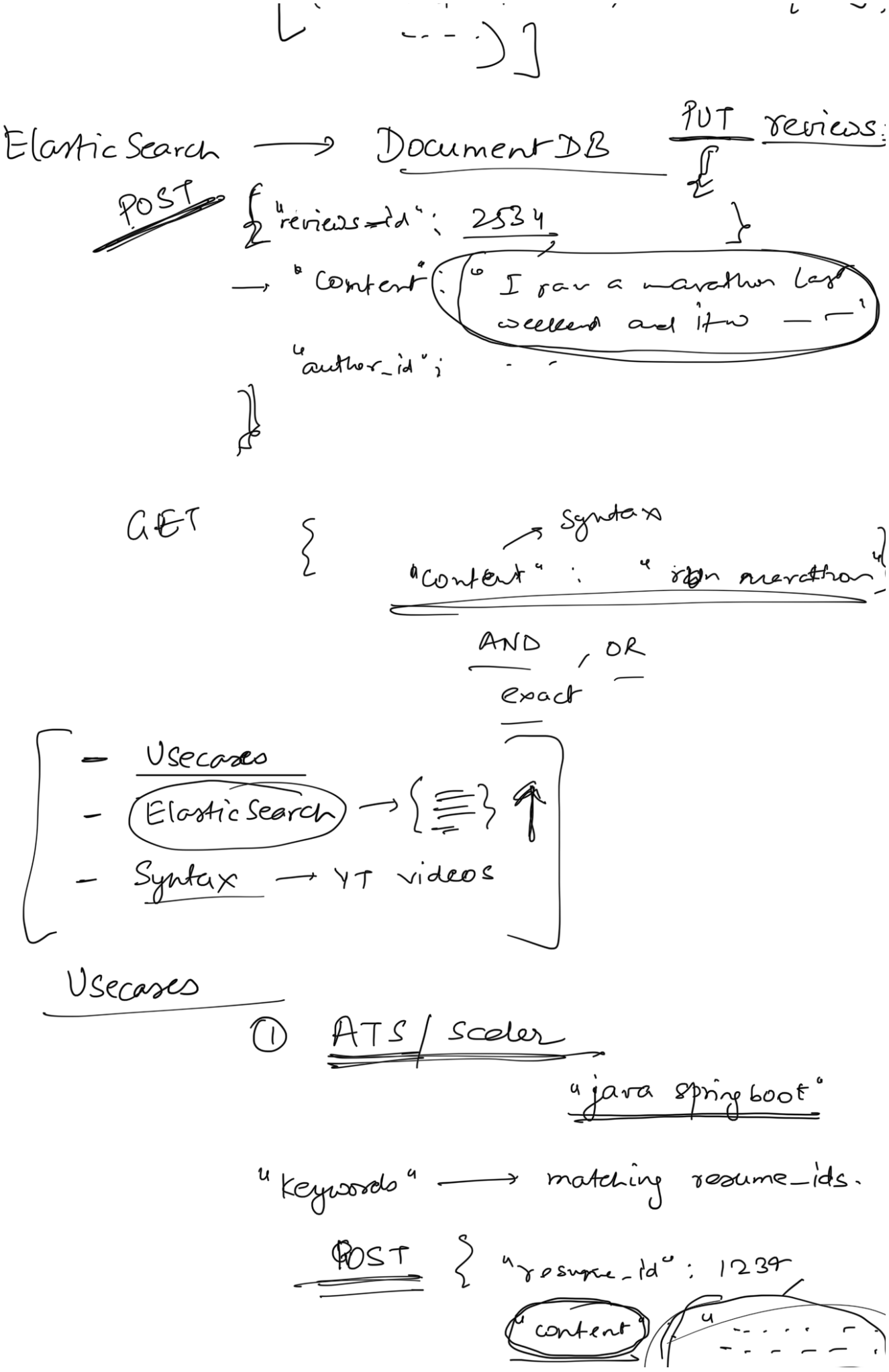
remove a → (1)

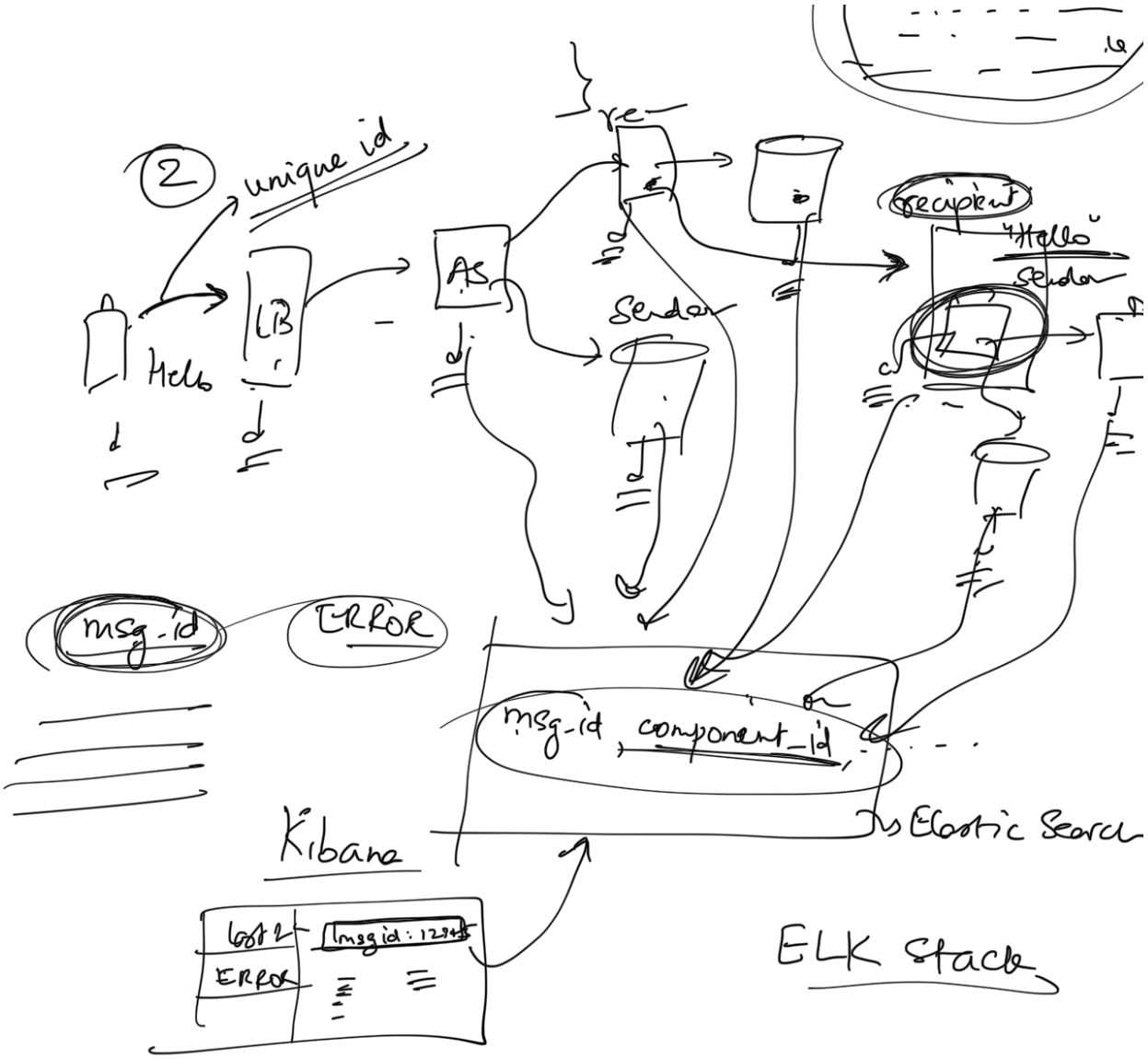
replace any letter → (1)

edit distance

Inverse Index

"word" → [(doc_id, [pos1, pos2 ...]), (doc_id2, [...])]

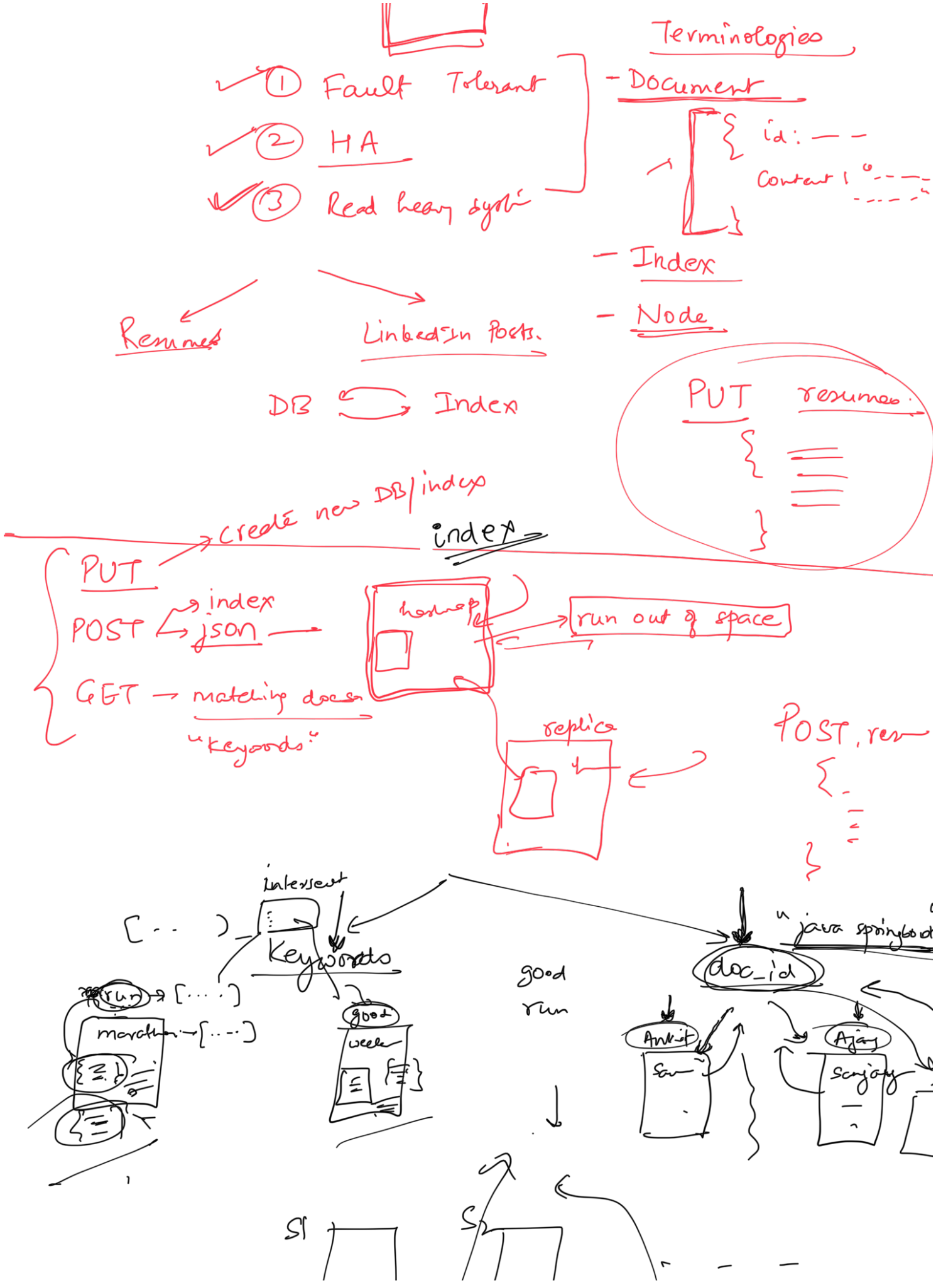

Elastic Search → Document DB
PUT reviews
POST
"reviews_id": 2534
"content": "I ran a marathon last weekend and it w
"author_id":
GET
"content": "ran marathon"
Syntax
AND , OR
exact
- Usecases
- Elastic Search → { }
- Syntax → YT videos
Usecases
① ATS / scaler
"java spring boot"
"keywords" → matching resume_ids.
POST { "resume_id": 1234
"content"

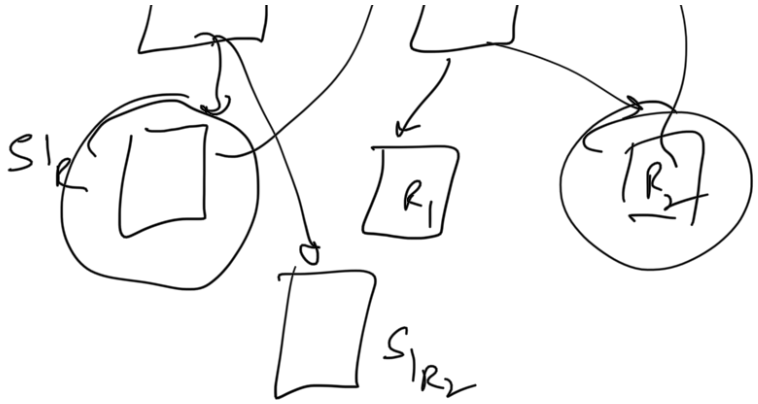


3 Search on your own website

{ page: " . . . — ",
"course pause" → content: " . . . — — .
. . . — — "}

Non-functional requirements


Terminologies
- Document
{ id: ---
Content: ---
}
- Index
- Node
① Fault Tolerant
② HA
③ Read heavy syst
Resumes
LinkedIn Posts.
DB
Index
PUT resumes
{
}
create new DB/index
index
PUT
POST
index
json
GET → matching docs
"keywords"
heap
run out of space
replica
POST, res
{
}
interest
[...]
Keywords
run
[...]
marathon
[...]
good
week
good
run
u java spring
doc_id
Ankit
Sam
Ajay
Sanjay
SI



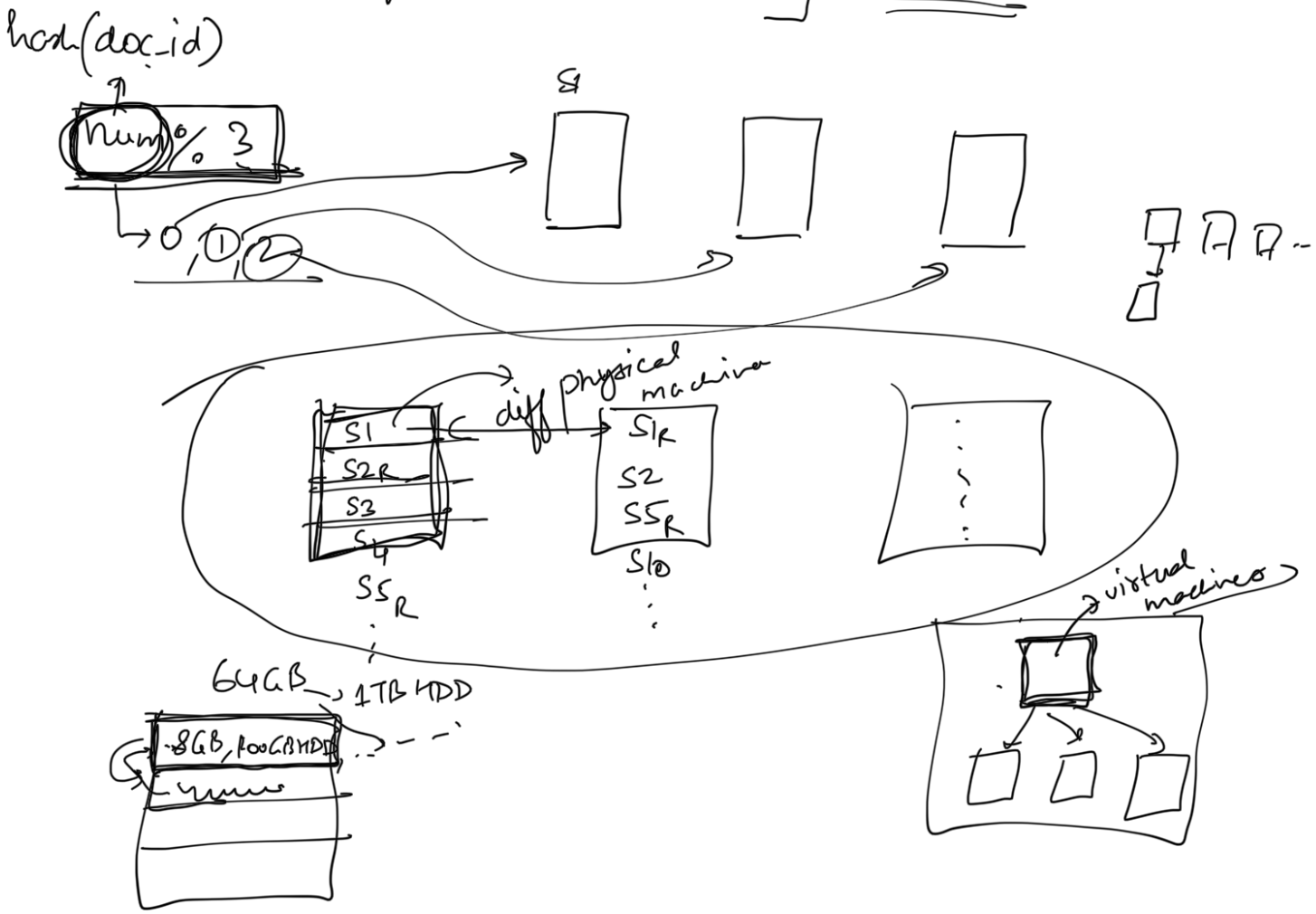
(1) Sharding_key : document_id.

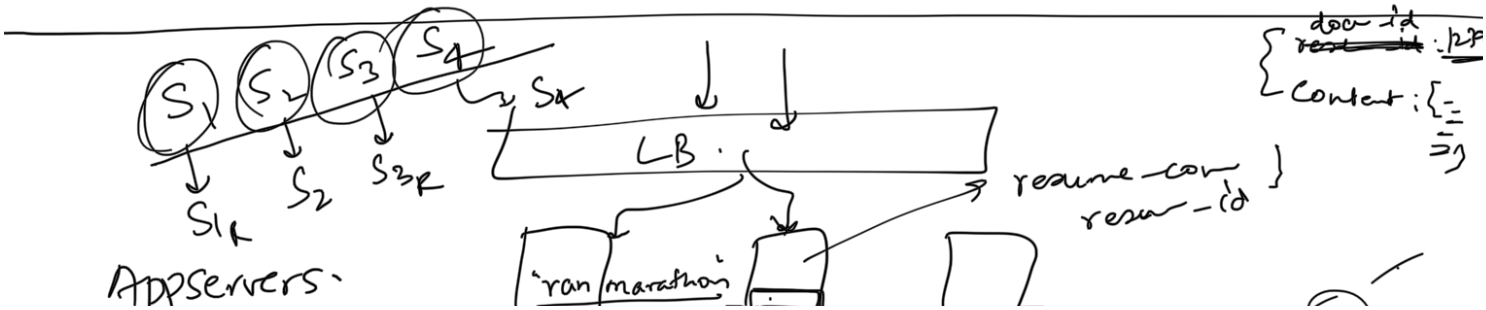
Shards : ~~100~~ 50

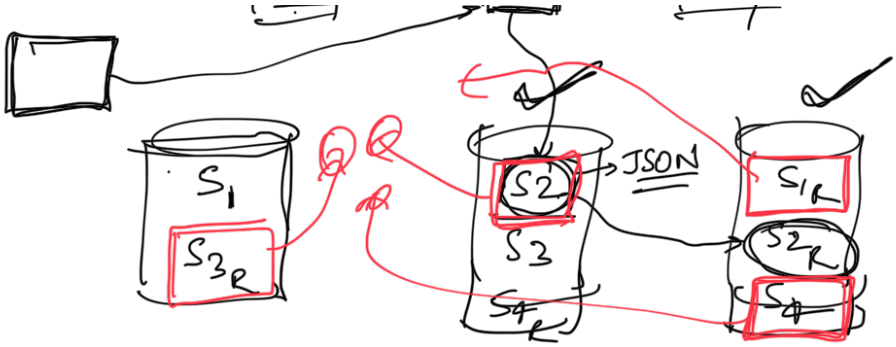
replicas : 3

hash(doc_id) % num_shards

150 machines



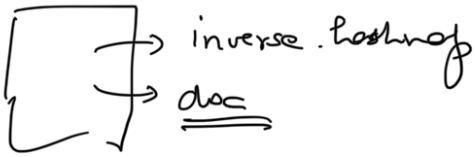




(4) shards
↓
2 replicas

S1 → S1R
S2 → S2R
S3 → S3R
S4 → S4R

(1) Request to all shards
↓
every shard
↓
random pick



(2) Wait for response

HA → low consistency.

ES:

(1) Indexing takes time

(2) Good for reads, (HA)
↳ slow for writes.

(3) Read

Regular exp:

Score: --

~~twitcheat~~

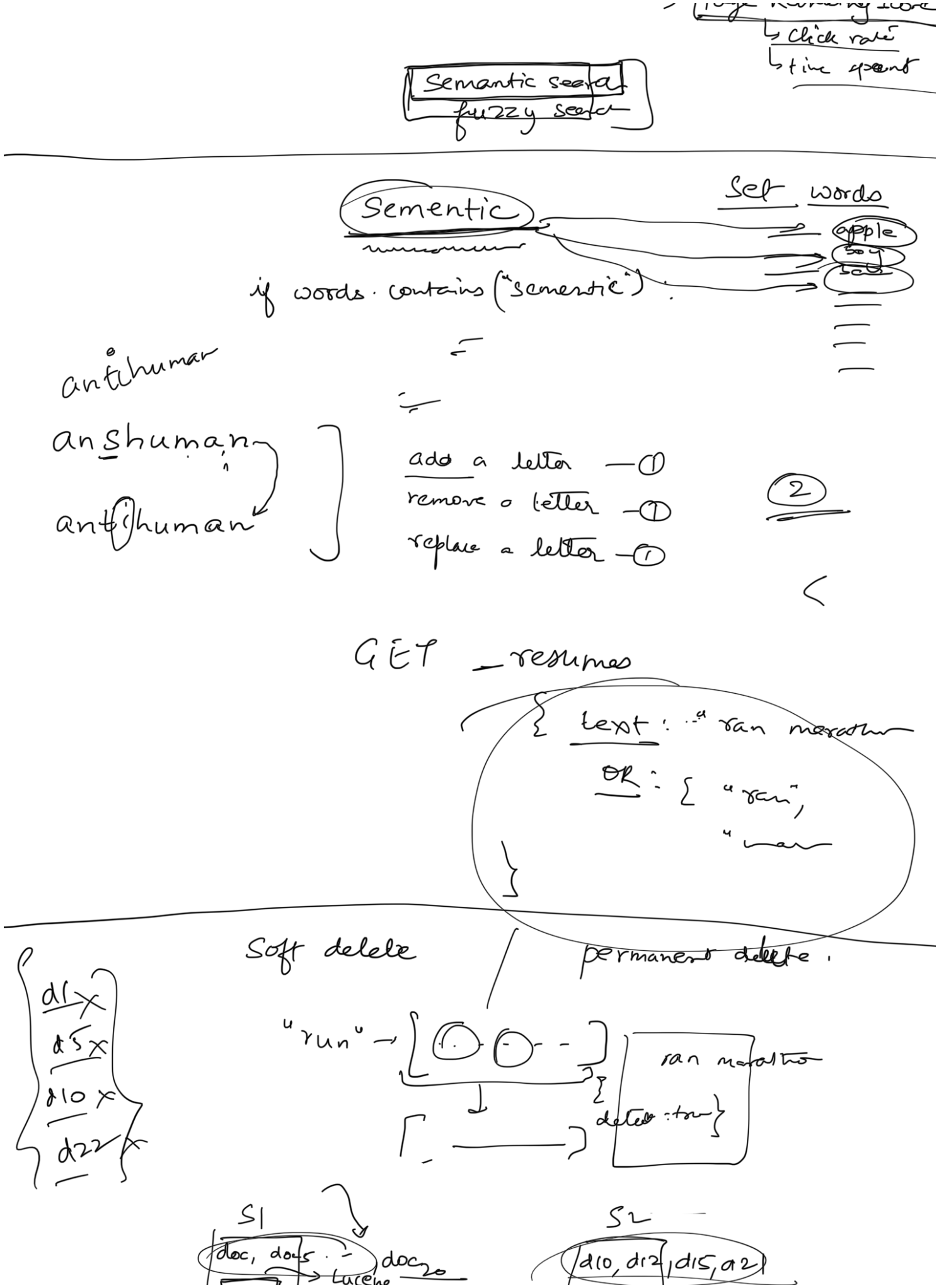
mikael jakson

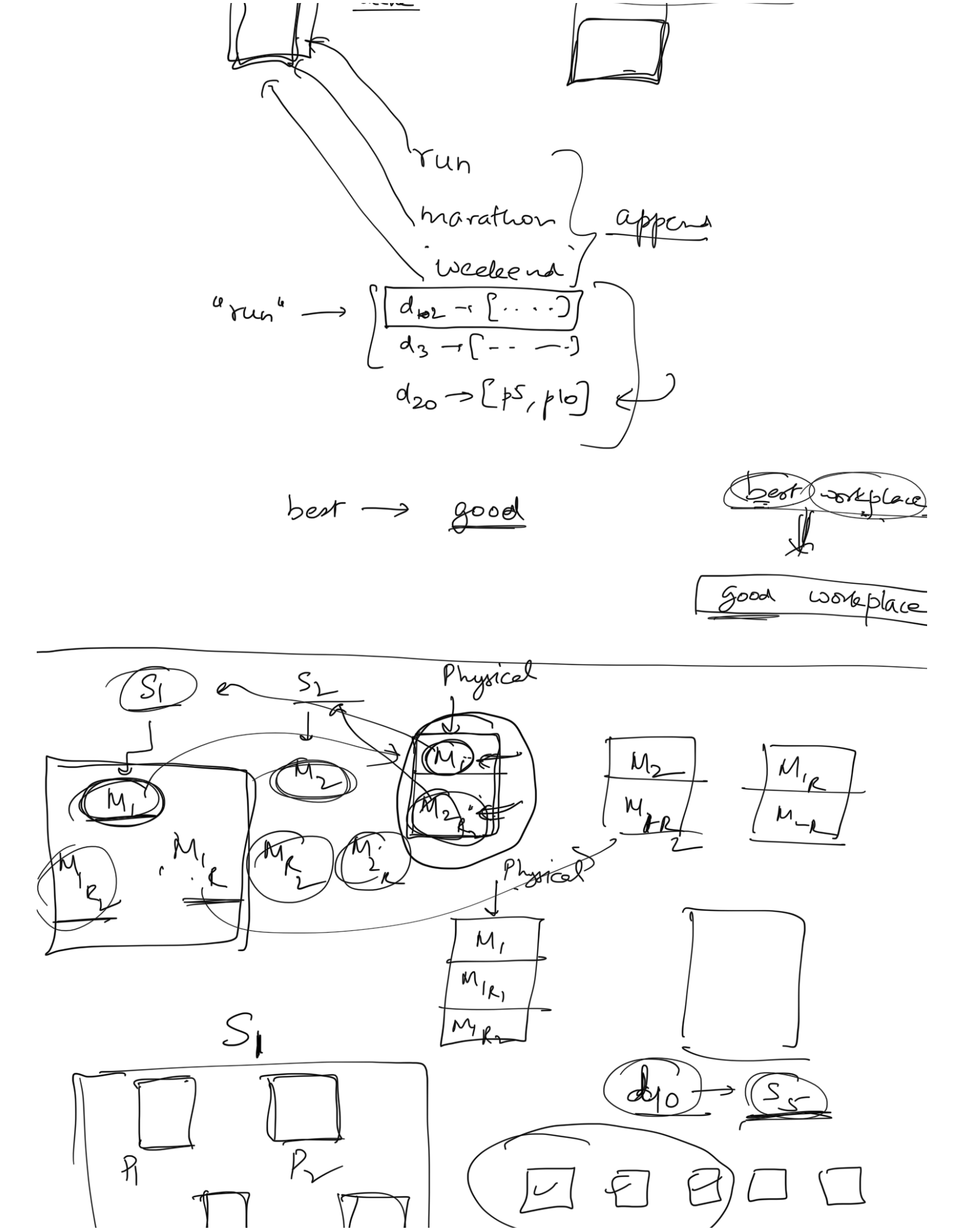
Ranking * (ban) marathon → 1.00, 0.9, 0.5

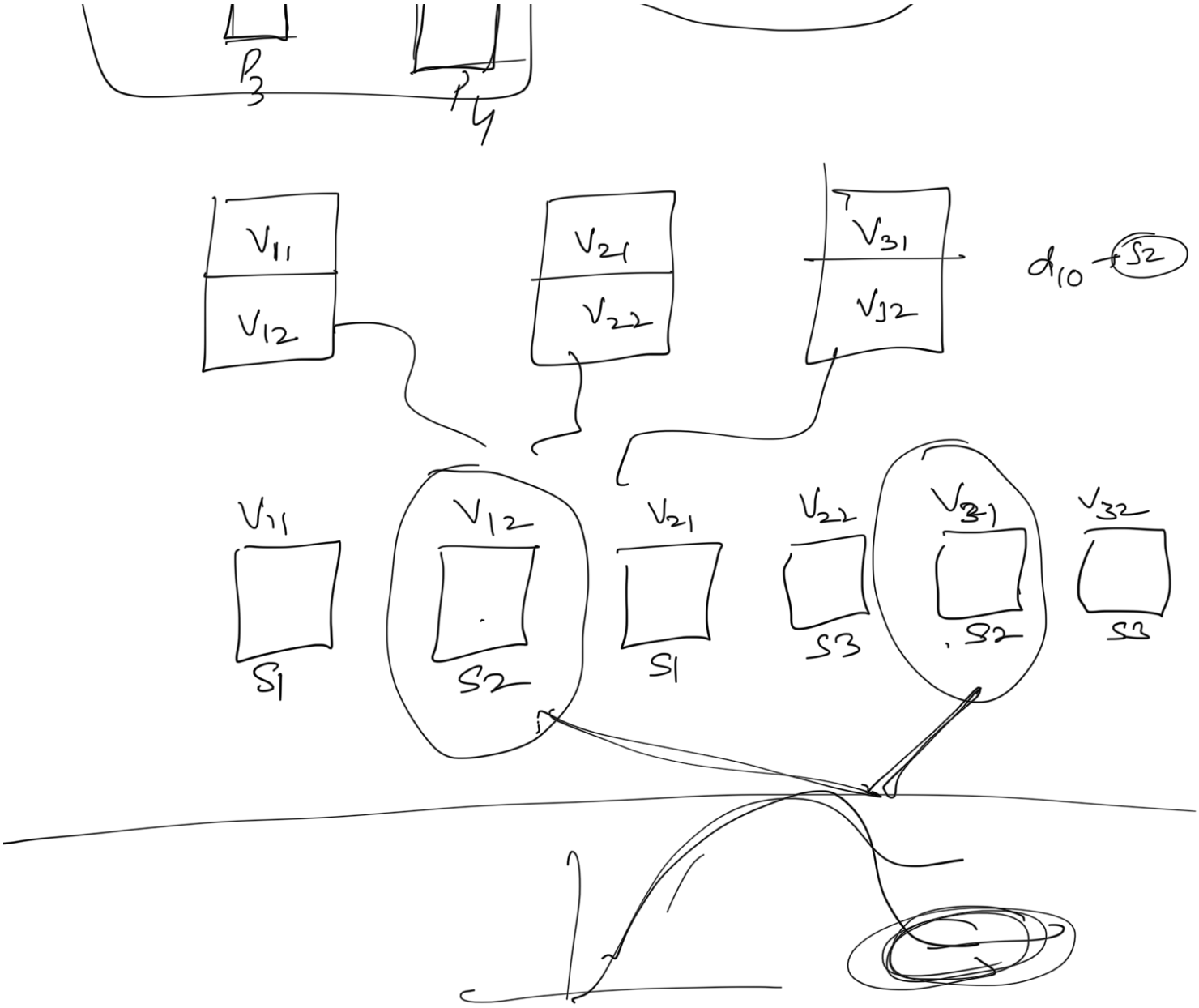
michael | jackson | michael jackson great dance

"Keywords" → matching pages

Ranking → Page score


Semantic search
fuzzy search
Click rate
time spent
Semantic
Set words
apple
if words.contains("semantic")
antihuman
anshuman
antihuman
add a letter —1
remove a letter —1
replace a letter —1
2
GET _resumes
{ text: "ran marathon
OR: ["run",
"man
}
Soft delete
permanent delete.
d1 x
d5 x
d10 x
d22 x
"run"
ran marathon
deleted: true}
S1
doc1, doc5 ... doc20
Lucene
S2
d10, d12, d15, d21


run
marathon
weekend
append
"run" →
best → good
best
workplace
good workplace
Physical
Physical
Physical
S_1
S_2
M_1
M_2
M_{1R_1}
M_{1R_2}
M_{2R_1}
M_{2R_2}
M_{1R}
M_{2R}
M_{1R_1}
M_{1R_2}
S_1
P_1
P_2
d_{10}
S_5


P3
P4
V11
V12
V21
V22
V31
V32
d10
S2
S1
S2
S1
S3
S2
S3