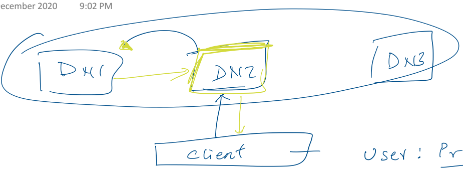
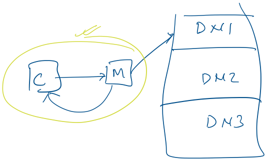
Cassandra

Thursday, 3 December 2020 9:02 PM


DN1
DN2
DN3
client

User : Prashant


C
M
DN1
DN2
DN3

Cassandra

Masterless

① De Normalise Data

No-SQL

RDBMS : tables

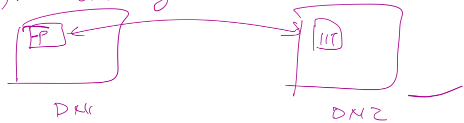
Table 1 User

name	email	University	~~# of students~~
Manideep	abc	BITS	
Pradeep	pqr	IIT	

Table 2 University

name	# of students	ph No.
BITS	600	—
IIT	7200	—

① how many students are studying in Pradeep's university.


FP
IIT
DN1
DN2

User , University

* more space

*

→ immutable

→ append & less updates.

only create new
no update
append.

② storage

i) Table : RDBMS ⇒ databases

ii) K.V : (key , value) [Email]

Redis

Manideeps < > json format

caching

⊙ last 5 emails: ⇐

⊙ size is always a restrictⁿ.

⊙ search is not posible.

akash : {

email1 : {______ }

email2 : {______ }

scan : o(n)

em 3 : {______ } from : raunak

: {______ }

: ? ______

name sender ______

1M ; ______

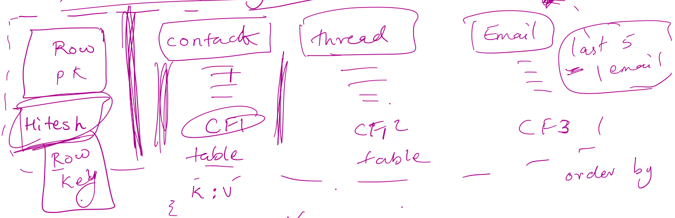
}

Redis

DynamoDB

Memcache

iii) Column family


Row
PK
Hitesh
Row
key
contact
CF1
table
thread
CF2
table
Email
CF3
last 5
1 email
order by

K : V

{

name : ✓

k1 : V1

K2 : V2

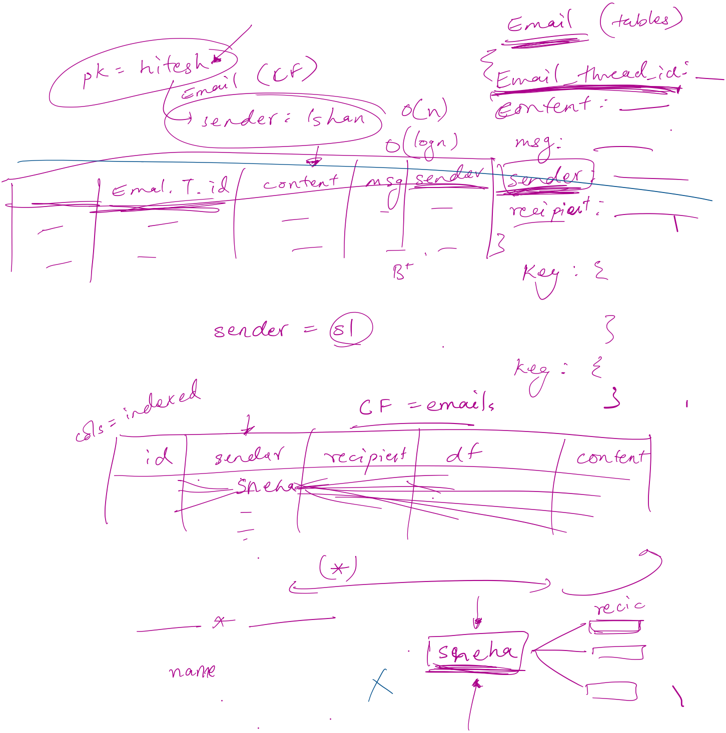
}

last 5 email

O(1)

- sorted in asc. order based on the time it arrived.

ts


pk = hitesh
Email (CF)
sender: Ishan
O(n)
O(logn)
Emal. T. id
content
msg
sender
B+
Email (tables)
{
Email_thread_id:
content:
msg:
sender
recipient:
}
Key : {
}
key : {
}
sender = S1
cols = indexed
CF = emails
id
sender
recipient
df
content
Sneha
(*)
*
name
recic
sneha

---

① column Family store

primary key
main acting entity of your design
user_id
	Contact: CF
	table.
	key: column
	value: row value
	{
	name: sneha
	ph.c : xyz
	email: —
		Emails : CF
		{
		recipient : ✓
		sender : ✓
		msg: ✓
		}{
		cc:
		}

Ishaan 3,

{ Name: Ravi
ph: —
email: —
}

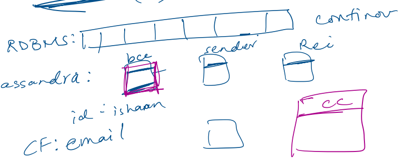
# of col can ↑ at a later stage

bcc	sender	Recipie	msg	cc
—	—	—	—	—

sender = sneha
user_id = ishaan O(1)

RDBMS:

Cassandra:

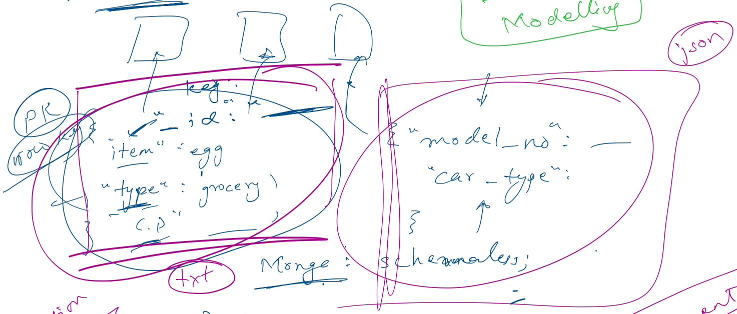

continuous
bcc
sender
Rei
cc
id : ishaan
CF: email

LSM →

demo = ?

Data Modelling

iv) Document


PK
row key
key:
"_id":
"item": egg
"type": grocery
"c.p"
txt
json
{ "model_no":
"car_type":
}

Mongo : schemaless;

Redis

① DynamoDB : KV, Document

collection
database
RDBMS

document
table

K.V stores : caches

(CF)

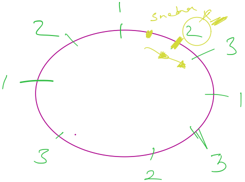
column : cassandra.

Mongo + JS (Node js)
(json) (json)

3) Sharding : 3 DN

S.K:

(user_id) % 360
sneha
hash


1
2
sneha
2
3
1
1
3
3
2

Replication factor:

⊙ DN 2 will hold sne's data

R.F = 2

DN3 –
DN1 –

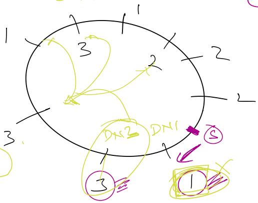
RF – ③

(RDMBMS disk space)

1000 server

hash (sneha) % 360

xyz@abc.com


1
1
3
2
2
2
3
DN2
DN1
S
3
1

C.H Ring

RF=2

⊙ DN1 : will have sneha's data.

R.F=3
in total 3 nodes in mys cluster every in must hold data (key).

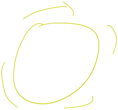
default mode : only 1 server has the data.

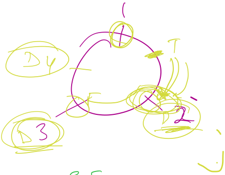
D~~N~~1: –

DN1
DN3

R.F = n

②





⊙ Read :- Replicatⁿ
⊙ Write :-

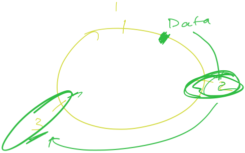
user-id- Tushar
contc.no:

RF:

R : min of reads to be done.

R = 2

W : " " writes



RF : 3
R : 2
W :

RF = 2

data loss

D2
tushar : 90

D3
tushar : 85

W = 1
R = 1

If Read = 30
31

consistency ==

R+W > R.F = 2

R, W

RF = 3
W = 2

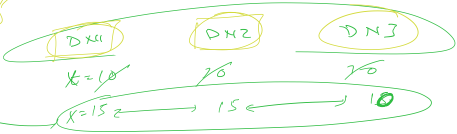
RF = 3
W = 1 , R = ~~2~~ 3

		DN1	DN2	DN3
	t=0	x=10	x=10	x=10
success ←	t=1	x=15	x=10	x=10
latest (10) ←	t=2	x=15	x=10	x=10

consistent

RF = 2

W = 3 R = 1
W = 2 R = 2


DN1
DN2
DN3
~~x=10~~
~~10~~
~~10~~
x=15
15
10

↑ W ↑

R+W > RF = high consistency

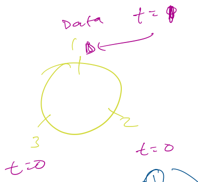
→ high write throughput : W small = 1

→ read " : R small = 1

R+W > RF

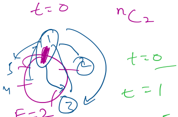
R = 1   W = 1

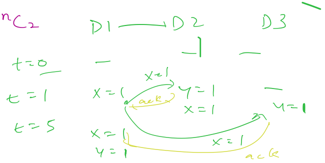
highly available system with chances of data loss.

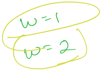

Data
t = 0
1
2
3
t=0

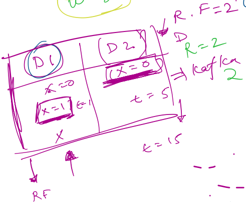
5 ms

RF = 3


t=0
nC2
1
2
3
4
5


t=0
t=1
t=5
D1
D2
D3
x=1
x=1
y=1
ack
x=1
y=1
x=1
y=1
x=1
ack

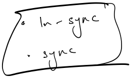



R.F=2
D
R=2
kafka
2
D1
D2
x=0
x=1
t=1
t=5
x
t=15
RF

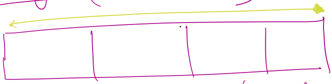
sync mode (w)

async mode (periodically → broadcast)

gossip protocol


In-sync
sync

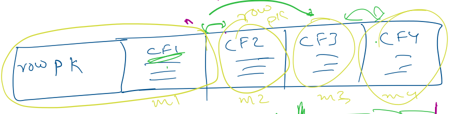
- Storage (Cassandra) : 15

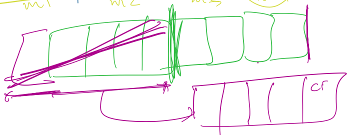


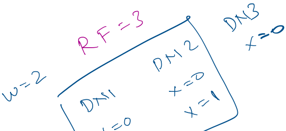
memory (HDD)

Row based

- schema is fixed.
- size of each record.

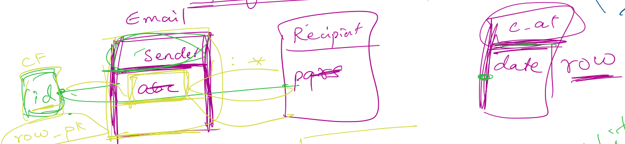

row
PK
row pk
CF1
CF2
CF3
CF4
m1
m2
m3
m4




RF = 3
w=2
DN1
x=0
DN2
x=0
x=1
DN3
x=0

⊙ fragmentation of disk

x = 1
in-sync: 3
2


Email
CF
id
Sender
~~abc~~
: *
Recipient
~~pq~~
row_pk
c_at
date
row

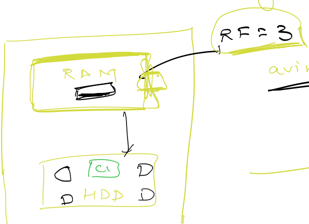
Same
datatype

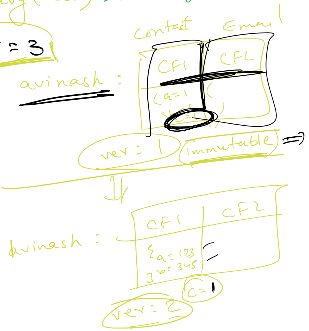
*

avg(col) ⇒ redshift
druid

x secs
4 MB/GB

RAM
+
HDD
combine all
copies
store merged
result


RF = 3
RAM
C1
HDD

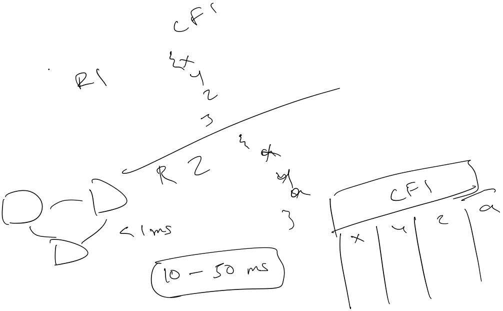

contact
Email
CF1
CF2
avinash :
(a=1
ver : 1
immutable ⇒
avinash :
CF1
CF2
{a=123
b=345
}
C=1
ver : 2

multiple copies of
same row PK,

- disk flush ( )
- reads are slow ⇒?
- automactic
  = compactn] →
- manual

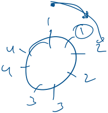
usert: { a = 123
b = 345
c = 1
}

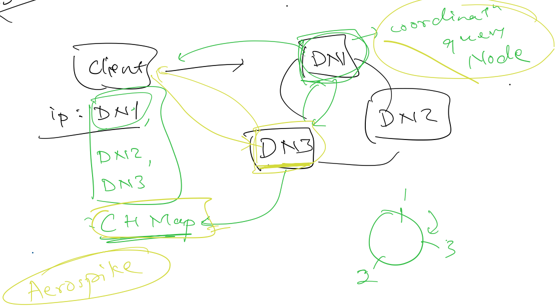
Read ←


CF1
{x
4
2
3
R1
R2
{ ~~a~~
4
a
}
< 1ms
10 – 50 ms
CF1
x
4
2
a

⊙ LSM ⇒ Tree
log seperated
Merkle
Tree

(B+





Client
ip : DN1,
DN2,
DN3
CH Map
Aerospike
DN1
DN2
DN3
coordinator query Node
1
2
3
2