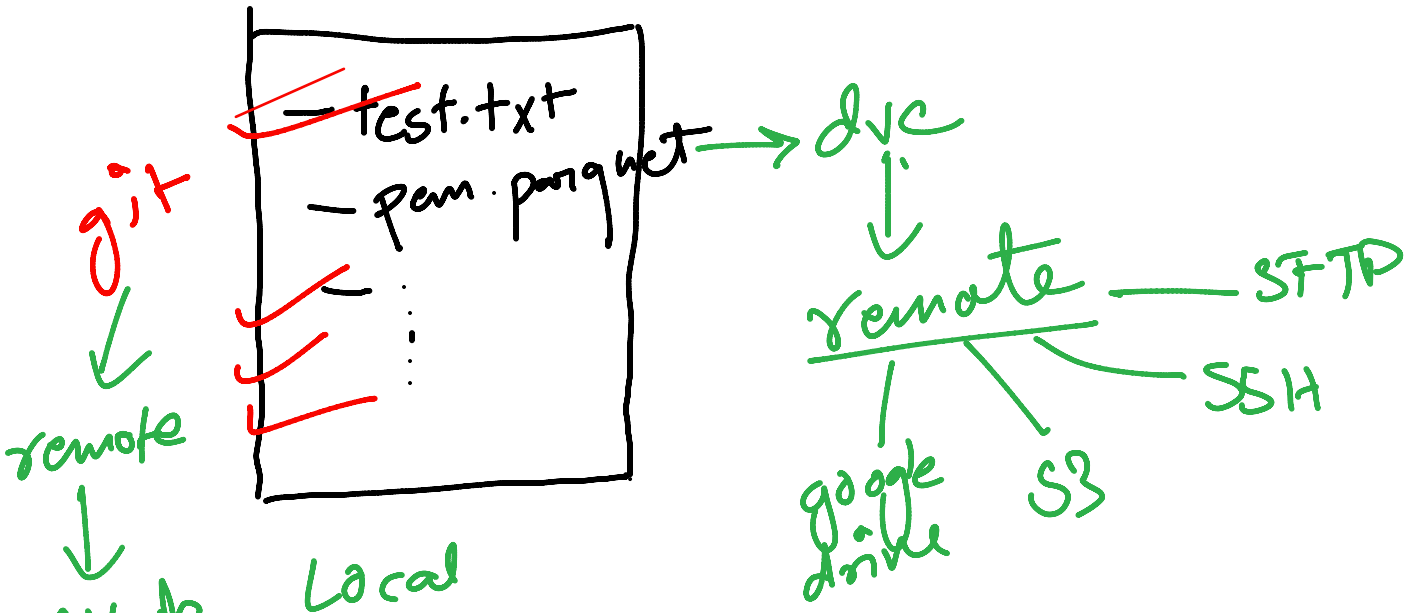
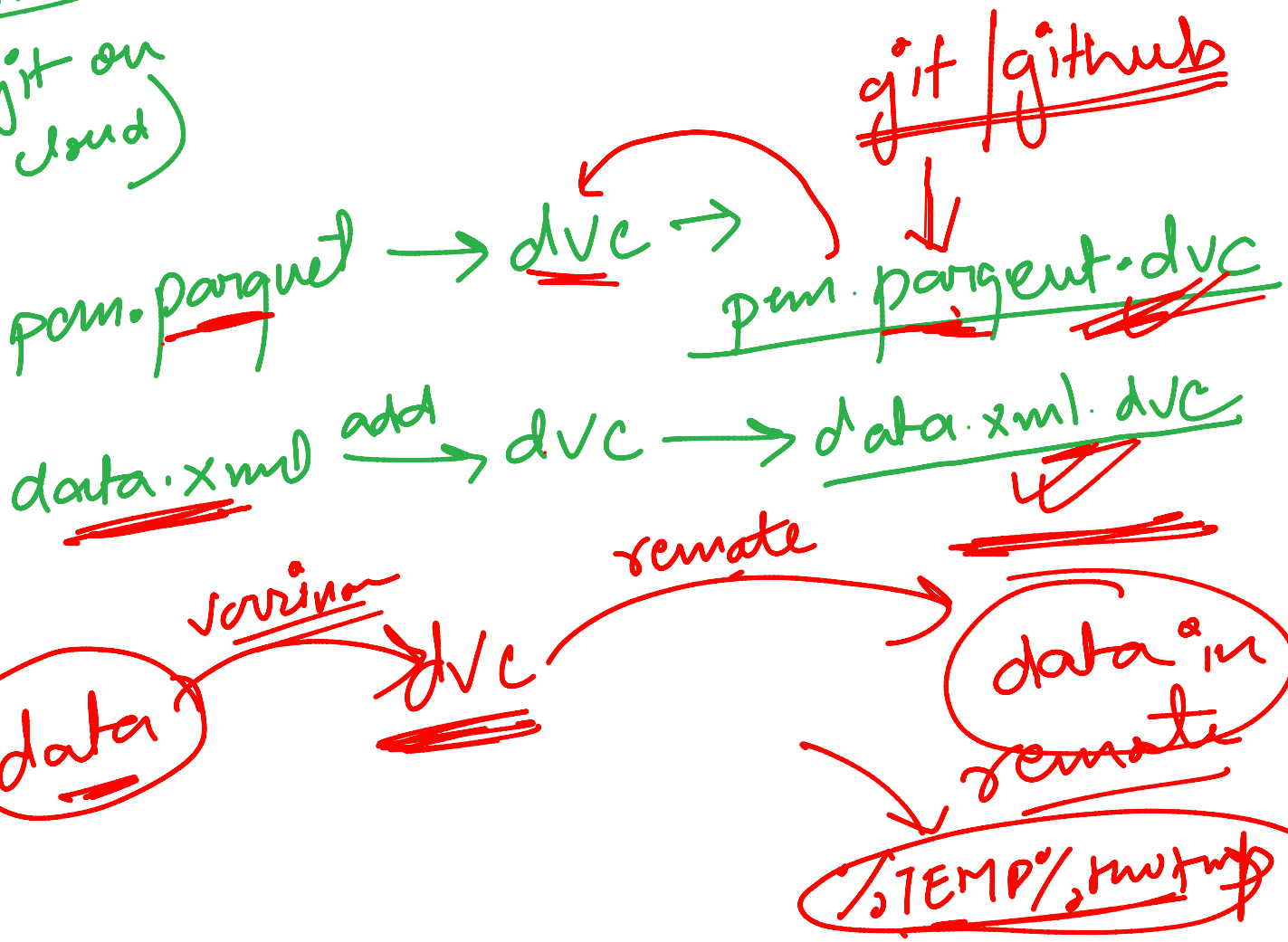
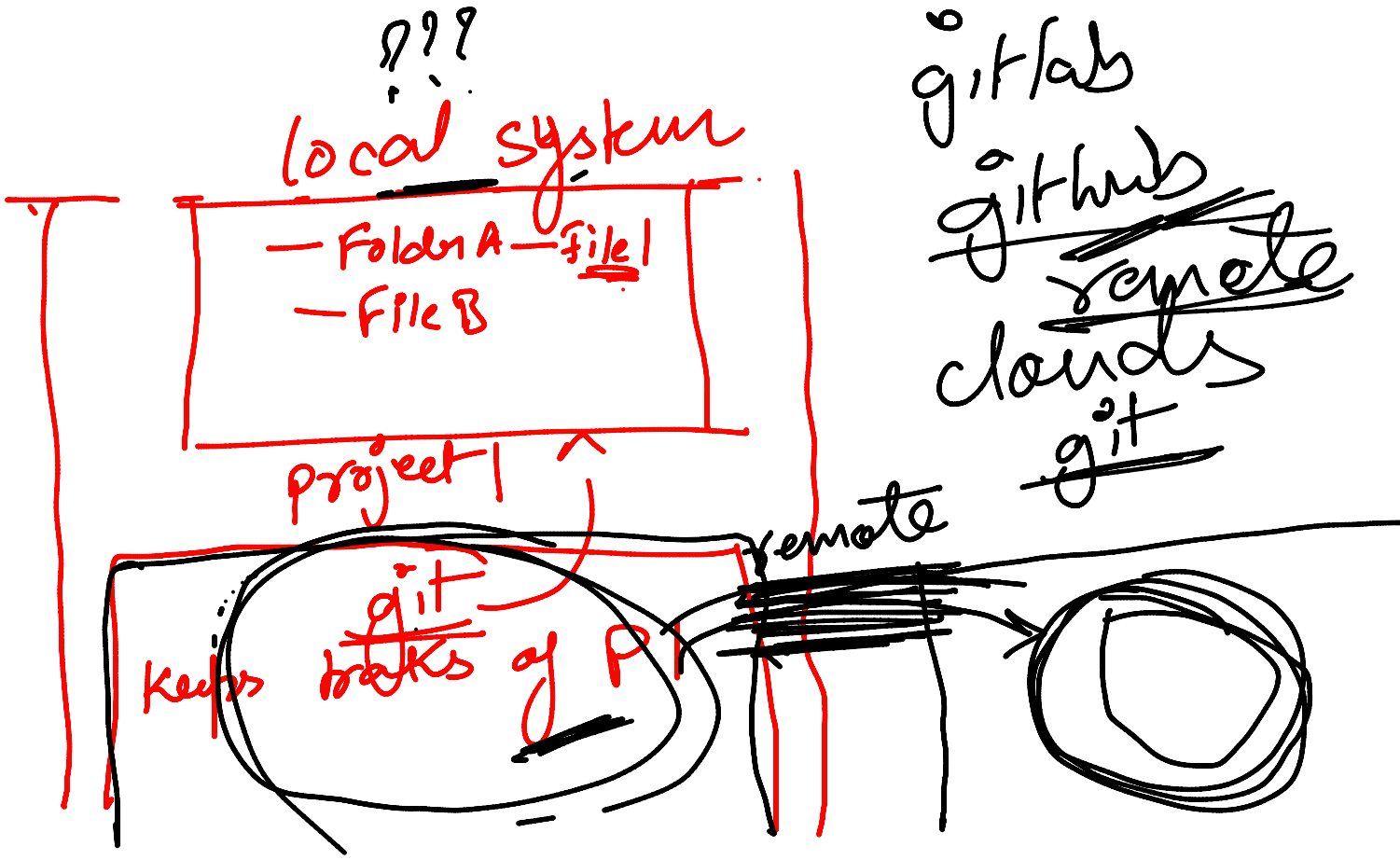
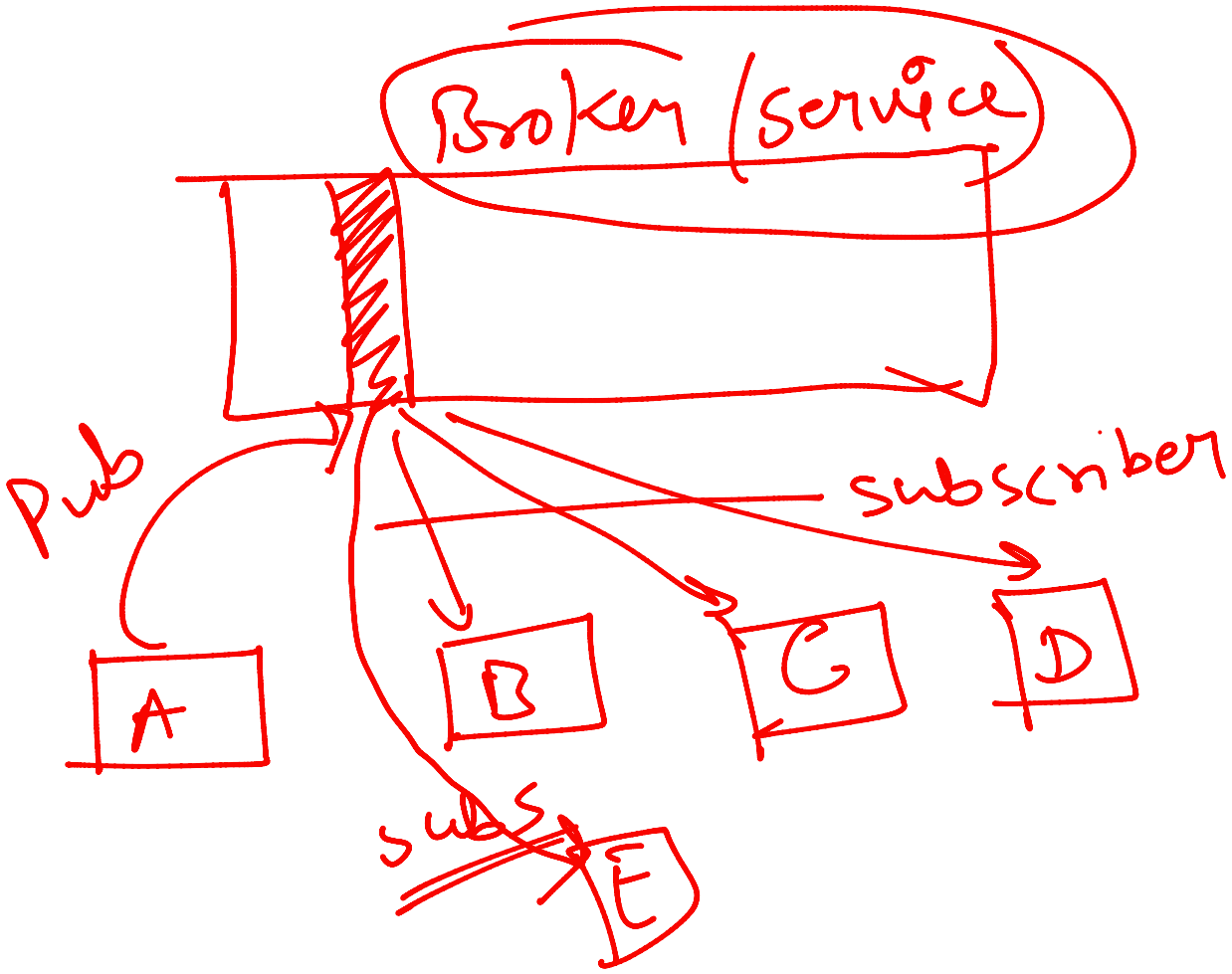
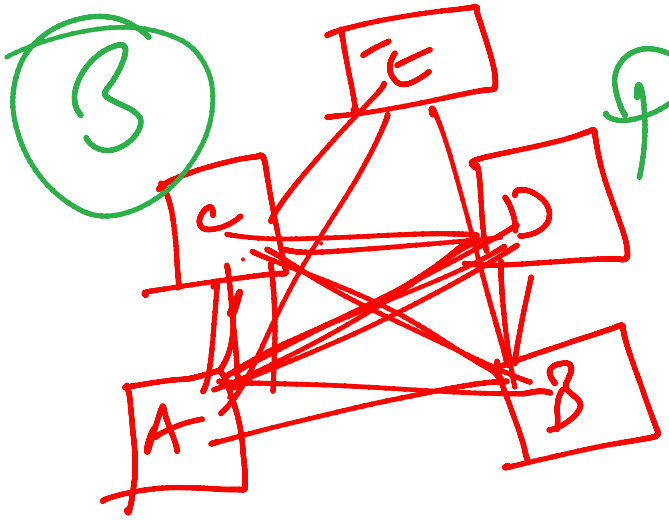
git & dvc


test.txt
pan.parquet
dvc
remote
SFTP
SSH
google drive
S3
Local
git
remote
github
(git on cloud)


git/github
pan.parquet
dvc
pan.parquet.dvc
data.xml
add
dvc
data.xml.dvc
data
versioning
dvc
remote
data in remote
/TEMP/storage


???
local system
Folder A
File1
File B
project 1
git
Keeps track of P1
remote
gitlab
github
remote
clouds
git

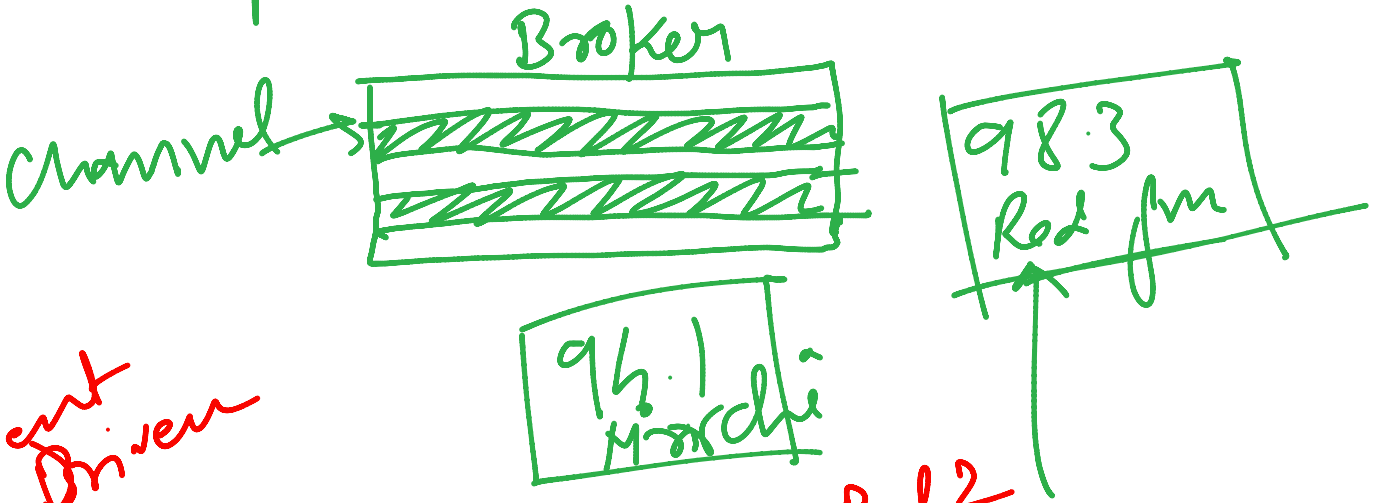

Broker (service)
pub
subscriber
A
B
C
D
subs
E



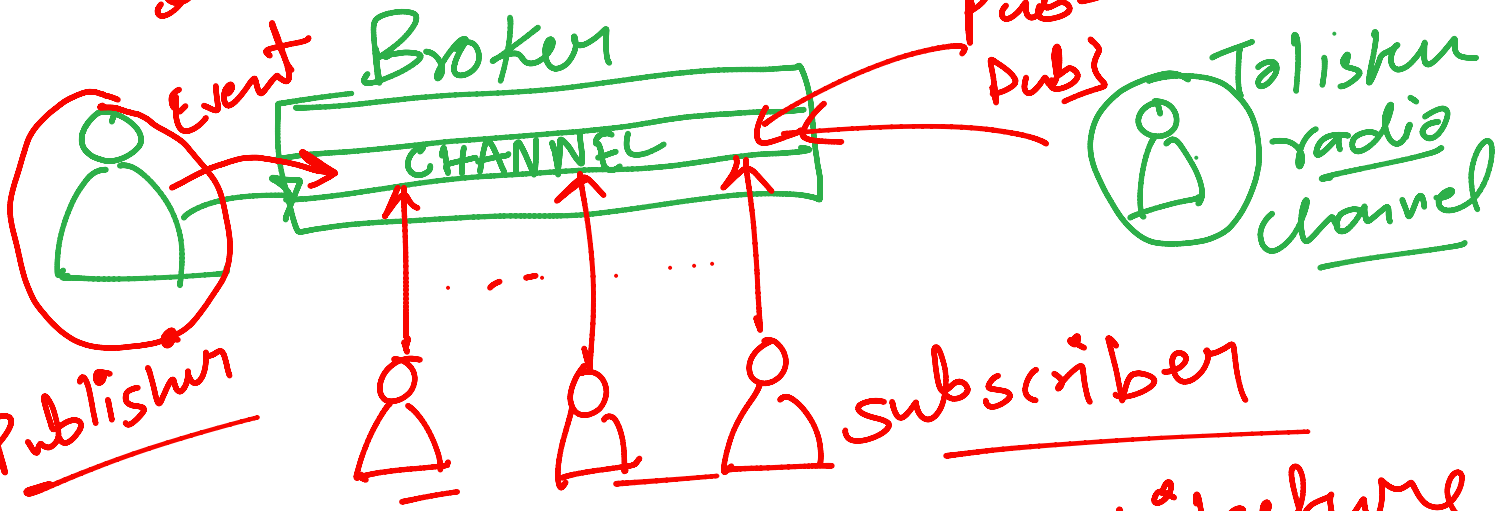
Problem

A-D ≠ D-A

the REST API conn. grows exponentially

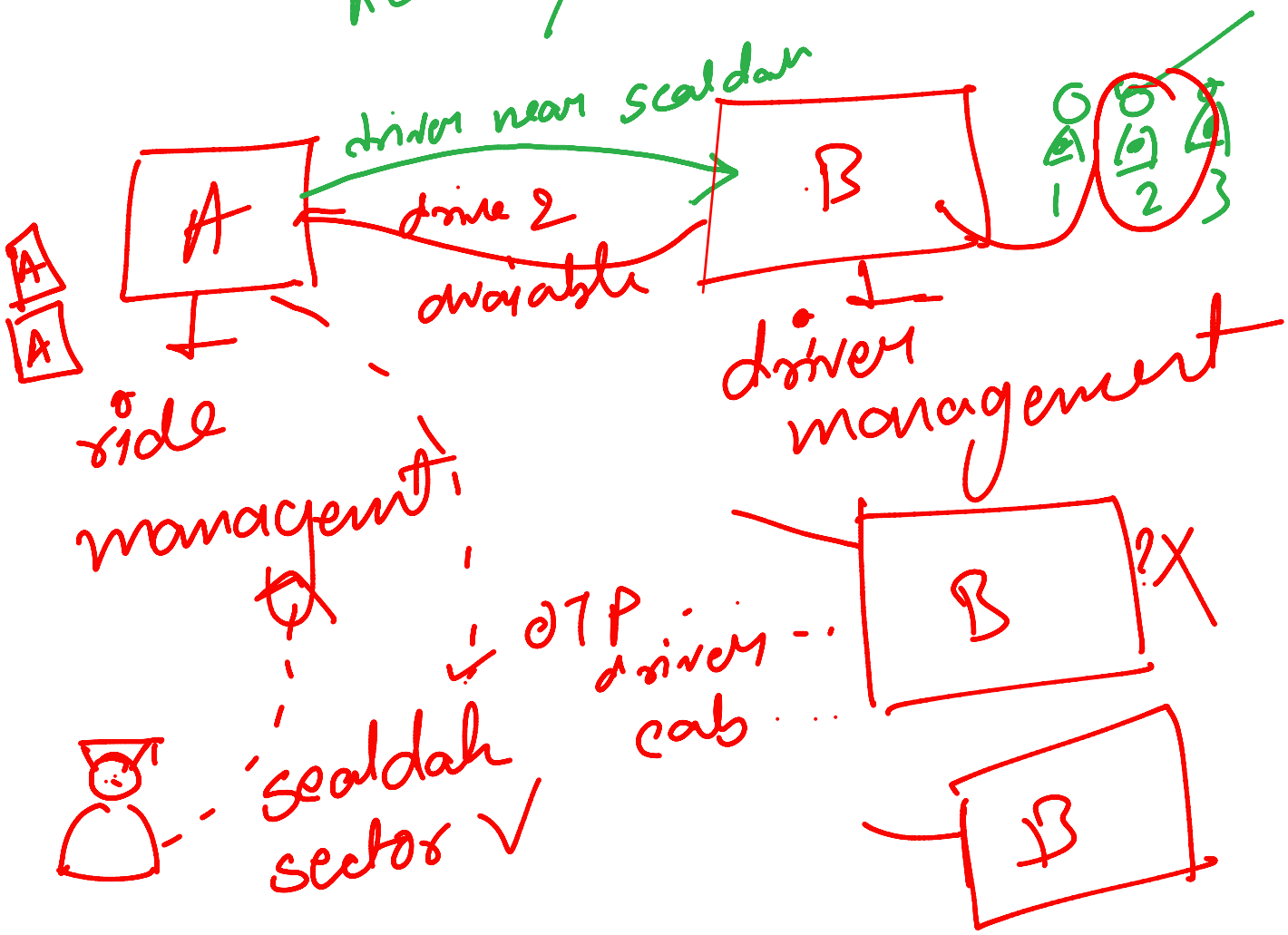


Event Driven



Pub-Sub architecture

(2) Data flow using REST / RPC

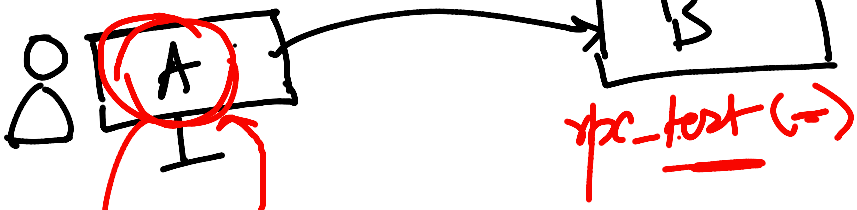


- horizontal scaling become very easy → increasing the number of servers

- No Bottleneck

RPC (remote procedure call)

API

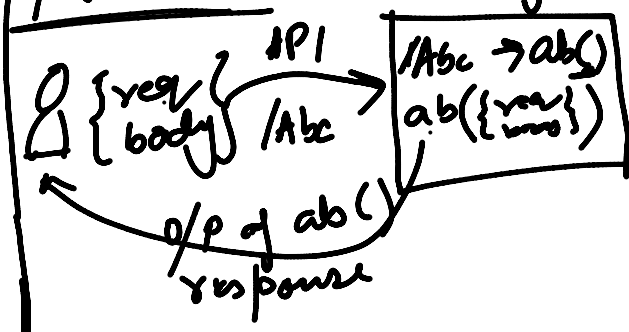


B.rpc_test(123)

REST

port: 3000

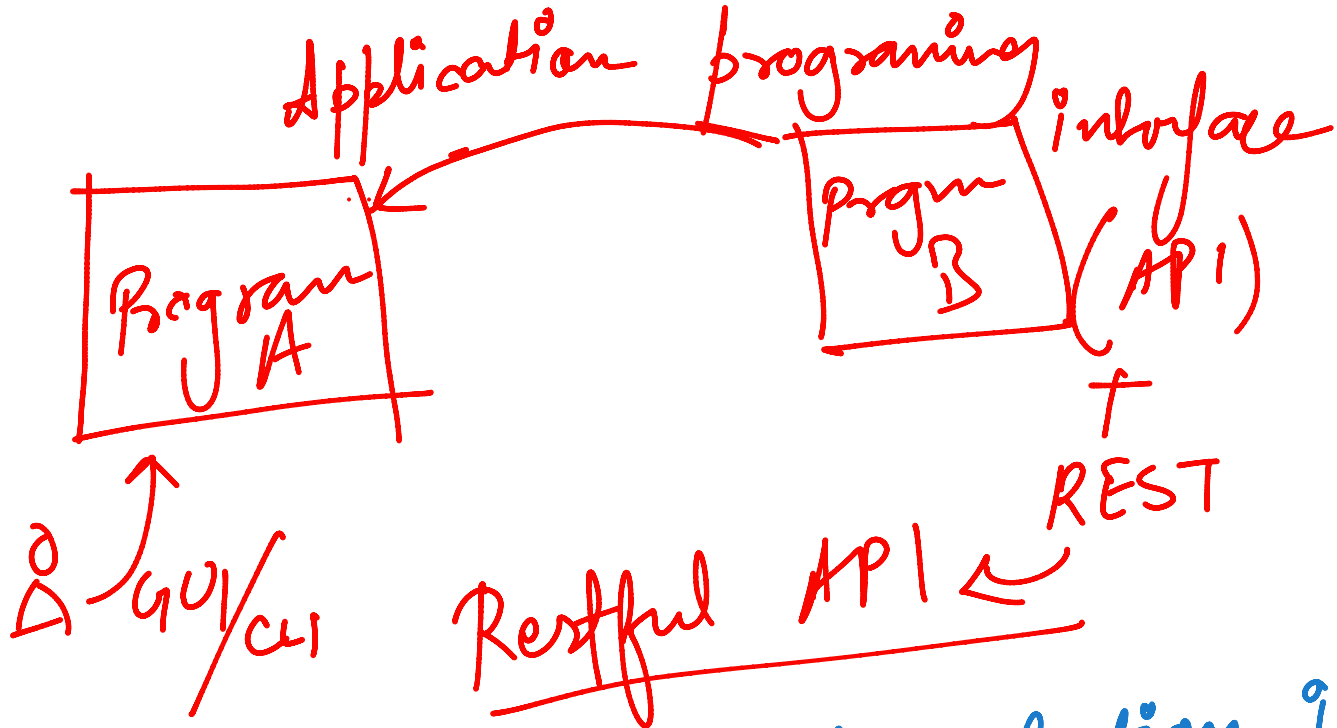
Program



① A calls the rpc_test() in (just computer B

② rpc_test() is executed in computer B

③ rpc_test() returns the result to the caller (ie computer A)



① Route → identify a function in that program in that computer

② Request Body

③ Response Body

192.168.1.1 : 3000 /test/myfnc

IP | Port | Route

/test/myfnc → hello()

IP identified computer

port identify program in a computer → 0 - 65535 ($2^{16} - 1$)

request body → JSON (Javascript object Notation)

↳ Python diction

↳ user input to hello()

response body → JSON

↳ hello() o/p to the user

2

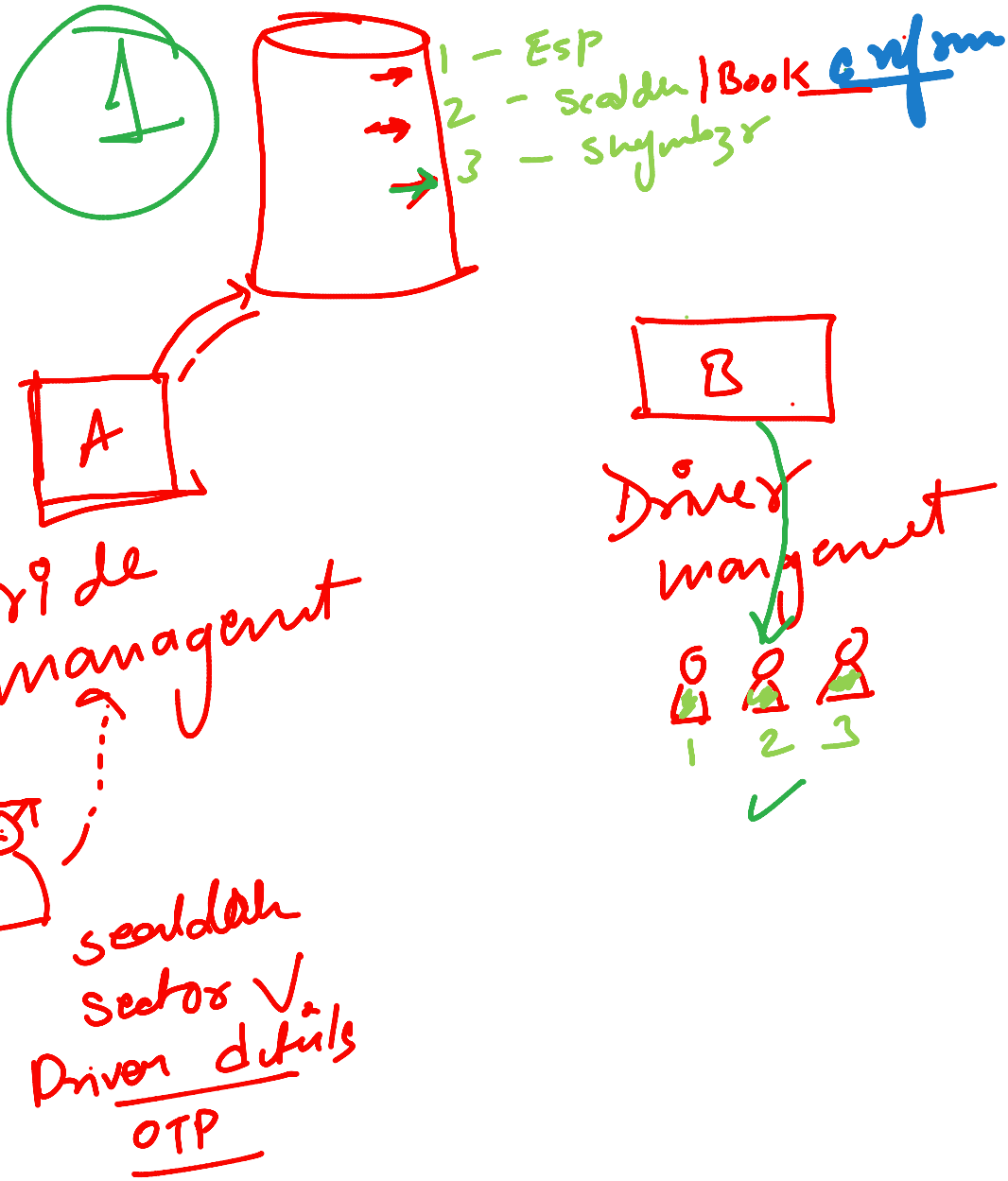
REST → representational State Transfer

RPC – Remote Procedure call

asks the user to send all the information of the current state to the program and not expect the program to remember which state the user was/is in.

Modes of Dataflow

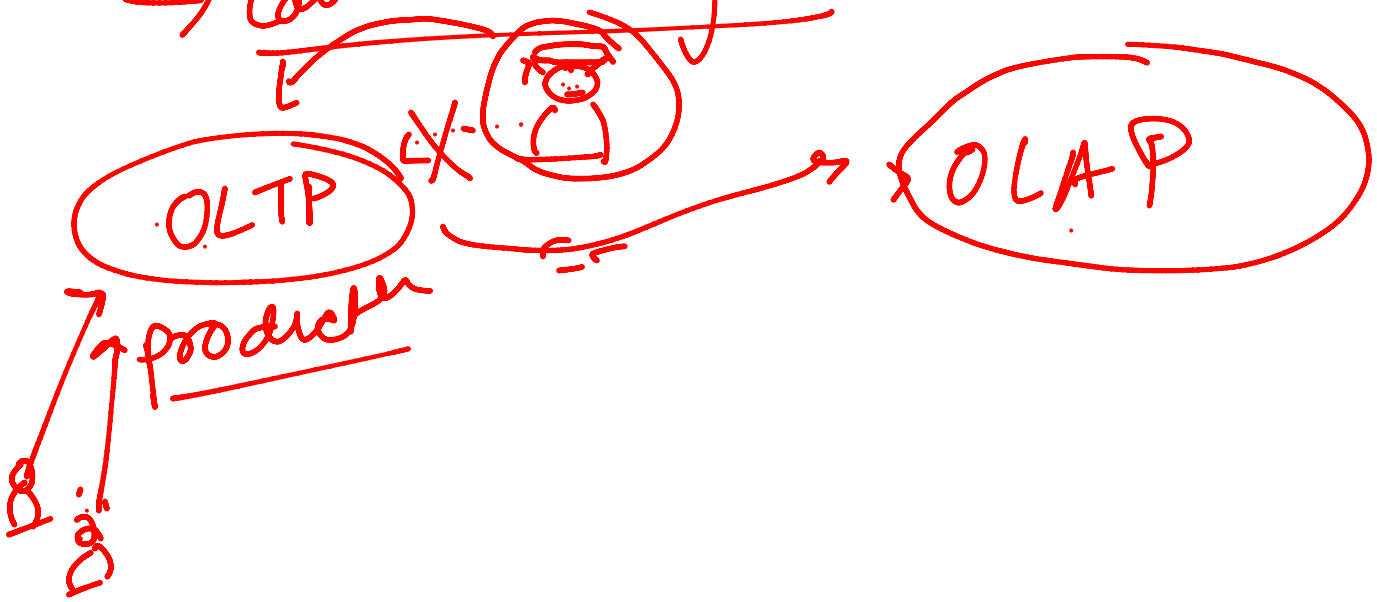
1. through shared database
2. using REST / RPC service
3. using event bus



online analytical processing (OLAP)

I want my analytics question to be answered fairly quickly & accurately

→ column major



Transaction processing

→ inserted (generate)
- updated (something changes)
- Deleted (No longer needed)

Online Transaction processing (OLTP)

fast, high availability

ACID

- Atomicity → either both be done or not
- Consistency → valid & follows set rules
- Isolation → No dependence on any transaction
- Durability → Commit stays even any failure no change

Data Storage Engine & Processing

Storage engine → Databases implementation of how Data stored.

Transaction Processing	Analytical Processing

Data Model

Relational

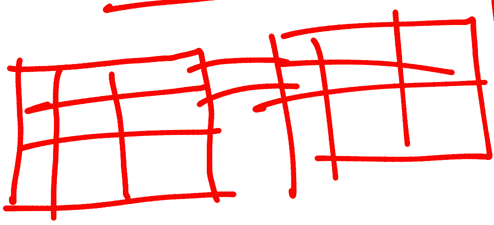
MS PG —

D1 D2 D3

- 1:1
- 1:M
- M:1
- M:M

has / can

DB2



Non Relational (No SQL)

D1 D2 D3

X X

- collection
- document

collection

→ D1 D2

doc — row

coll — table

Graph

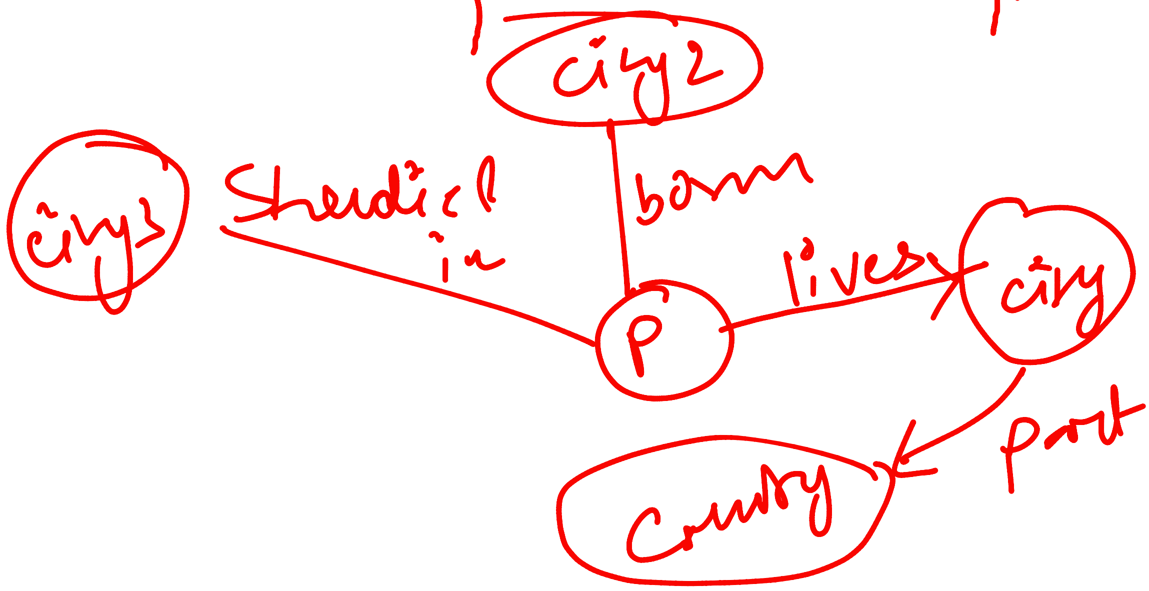
nodes — properties

- edges (properties)

unsure about relation data explore

person
city
country

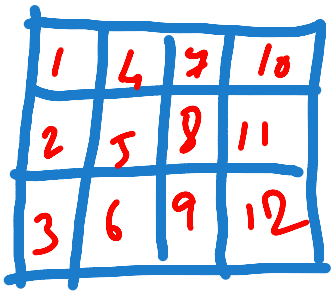
1:1 lives
1:1 part of



Data schema

enforced → writing → structured

enforcement → reading → semi structured



Structured (tabular)

row major — CSV

column major — parquet — arrow

CSV

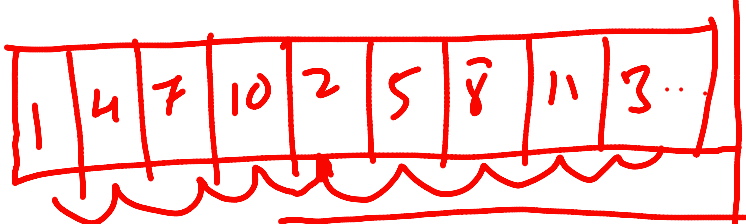
row major

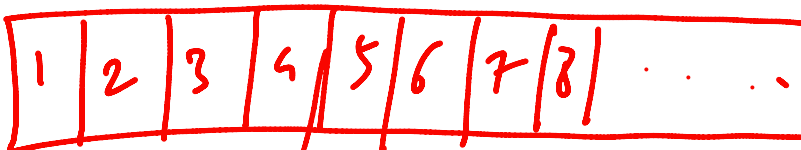
memory

parquet

column major

memory





≡ Write more

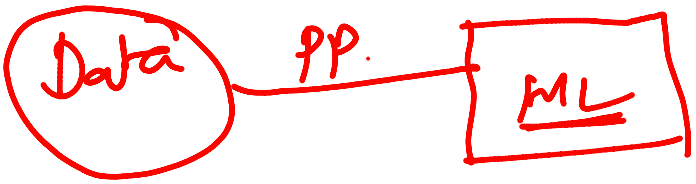
OLTP

Aggregate analysis

OLAP

Data Eng Fwdwk's

- structured
- semi structured
- unstructured. (Big size susceptible corruption)



means schema

→ predefined structure/predefined ways d data can be broken down and sorted into

Eng lang

Ram is going to the market