

## Dimension ?

- attributes of an entity (user's DOB, fav. food, city I live in)
- 1. Slowly changing - fav food
- 2. fixed | DOB / manufacturer / never changes

→ Identifier Dim = SSN / uniquely identify the entity

Who is your data consumer?  $\geq$  Data Modelling

1. Analyst / D.S. - give data that is

- 1. Easy to query (Analytical Dataset OR OLAP Cube)
- 2. Not complex

2. Other Data Engineers

- Master datasets
- Should be compact
- harder to query
- Nested Types like Struct, Array are okay

### 3. ML Models

└ Identification & flat primitive types  
└ depends on model & how it is trained

### 4. Customers

└ Charts or geometric patterns  
└ No data need to be given

"How the data is gonna be used?" ] Understand  
- Downstream Usage

### Dimensional Data Modelling

- OLTP (online transaction processing) queries <sup>low latency</sup>  
SWE do this type mostly <sup>low volume</sup>

3NF, P.K, F.K, Joins. - one user

- OLAP - most common for DE (online Analytical Processing)  
- Run a query fast is the aim  
- entire dataset is being looked at  
- optimize for large volume

- Master Data - middle of OLTP & OLAP

- Deduped

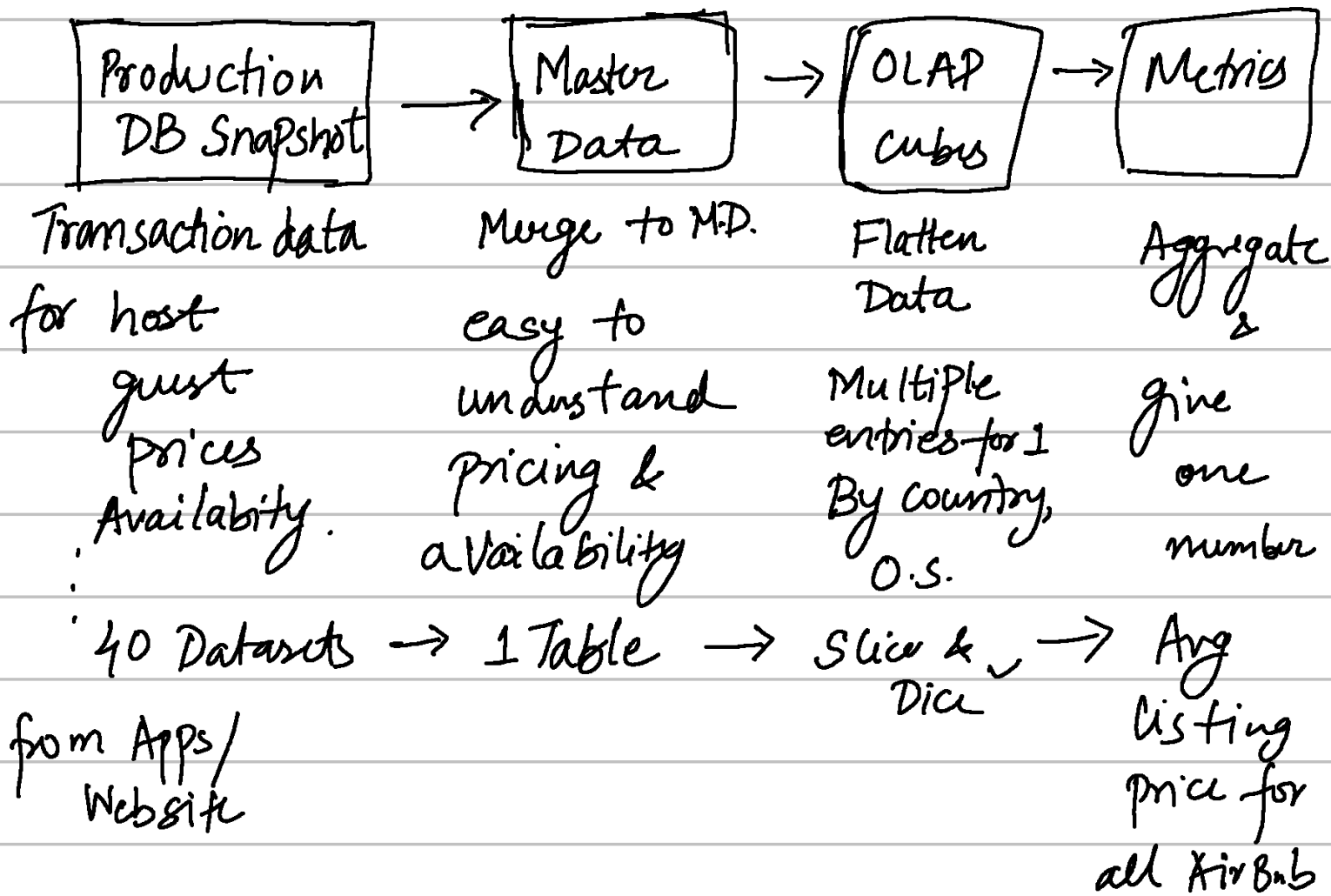
- optimize for completeness of entity definition

You had Transactional System <sup>but</sup> Modelled as Analytical System

→ optimized for population  
where as you just needed it for a user  
Online system will be slow

→ Vis a vis Analytical System modelled as Transactional  
↳ Joins - Expensive - too much for Analytical

∴ Master Data can help you - Agility to go wherever you want.



# > Cumulative Table Design (Master Data)

holding to all dimensions  
that ever existed.

hold to history.

(some days a  
user might not  
show up but  
you will still need  
that user in your DB)

## 1. Core Components

- 2 dataframe (Today & yesterday)
- Full outer Join the two dataframe
- COALESCE values to keep everything around
- Hang onto all of history

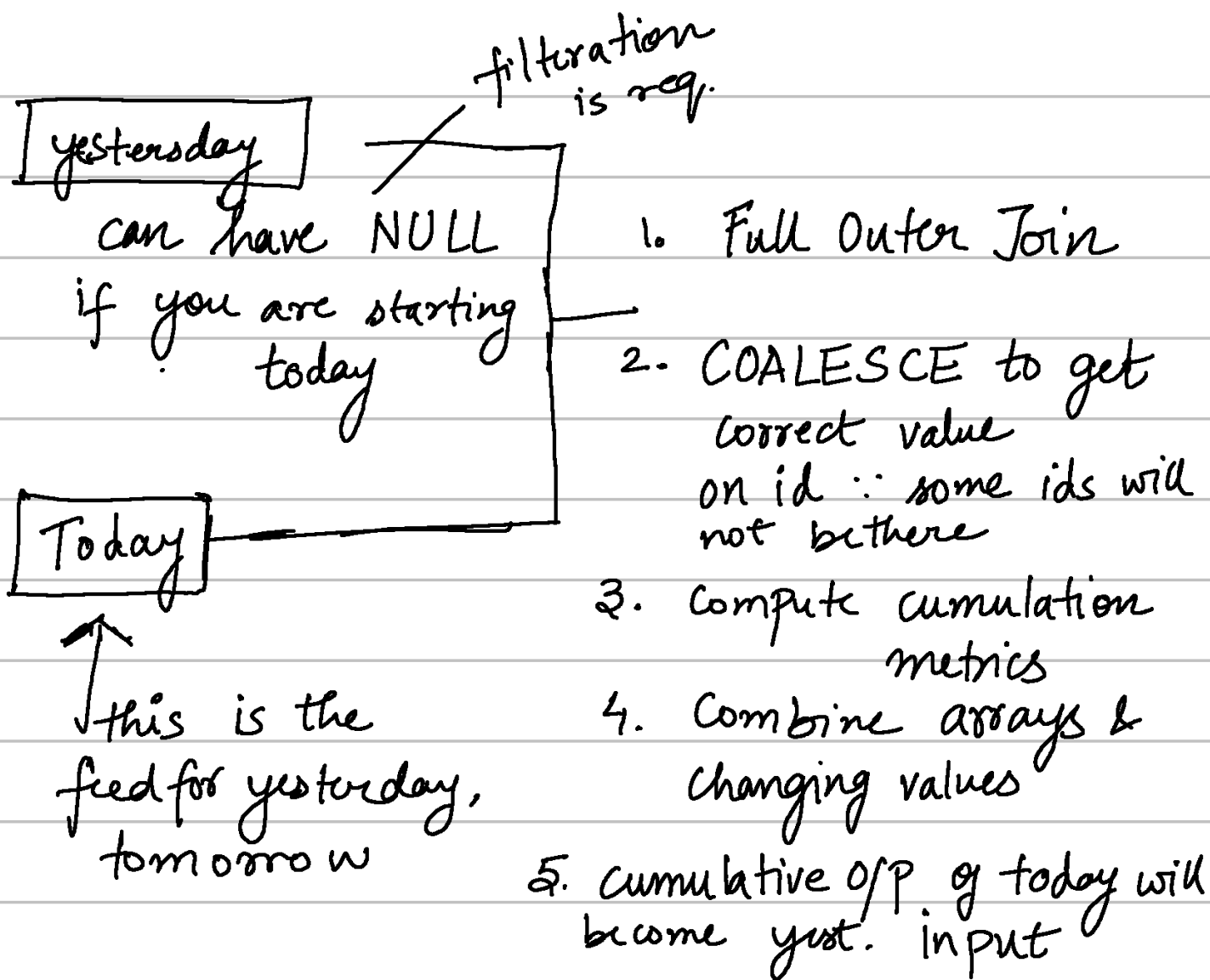
all same columns  
w/ diff data

## 2. Usage

- Growth Analytics at FB (dim-all-users)
- State Transition → Active yesterday Not Active Today  
Tracking = churn

(10,000 downstream  
pipelines.)

Not Active yest, Active today  
= Resurrective



## Strengths of Cumulative Table Design

- Historical Analysis w/o shuffle
  - you don't need to group by. Why? How?
- Easy "transition" Analysis
  - Select from latest table  $\because$  it is captured

## Drawbacks, not 100%.

- can only be backfilled sequentially
- Handling PII data can be a mess since deleted / inactive users gets carried fwd.

# Compactness vs Usability Tradeoff

## Compact Tables

are compressed  
to be as small as  
possible & can't be  
queried directly until  
they are decoded.

They want to reduce I/O

↓ SE focussed

→ Online Systems where  
latency & vols matter

## Usable Tables

No complex  
data types  
Easily Manipulated

↓ Analytics  
focussed

→ OLAP Cube

## Middle Ground

↳ Use complex D.S. (Map, Array, Struct), making  
querying difficult but compacting more.

Master Data

✗

≥ Struct { Table inside a Table  
Defined keys & value

≥ Map { Values all have to be same type  
Keys are loosely defined — any power of 2  
32,000 or 64K

≥ Array { ordered list  
Ordinal  
all have to be same type.

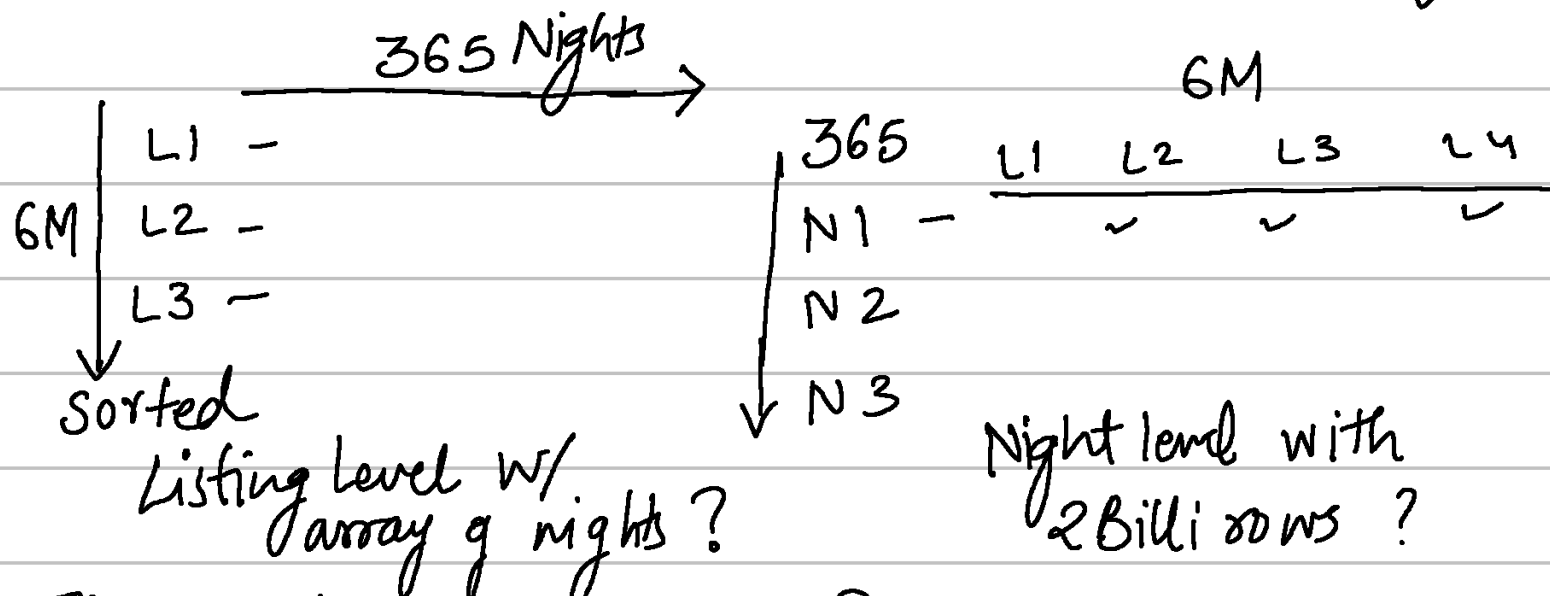
⇒ When you have Temporal Cardinality Explosion  
At Airbnb, we had

Listings → calendar  
/ per night  
~ 6 million 365 days

want to calculate  
nightly pricing &  
availability for  
next year?

$$365 * 6M = 2B \text{ nights}$$

How to Model this?



If you do sorting right, Parquet will keep same size.

Join in Spark → mix up the ordering of  
rows & ruin your compression

Run length encoding compression will not work

What to do? Sort Again?

But you should sort only once  
in Lab Explained Better.

# Seed Query for Cumulation in Code \*

Array Concat  $\rightarrow$  Row = Season Stats

Primary Key (Player, curr-season)

with yesterday ( )

with Today ( )

(  
Joined for all  
years the  
players are  
playing for  
w/ year info

Problem: The table has

multiple entries for a player

if we join w/ something downstream

too many rows. sol<sup>n</sup> = one row per player

Slowly Changing Dimensions

Lec-2

food preferences  
imp to model correctly

\* idempotency - ability to reproduce same results  
in prod or backfill no matter

1. when you run it - day or hour you run it
2. no. of times you run it

Run at 't' then backfill after 7 days - you get  
different data

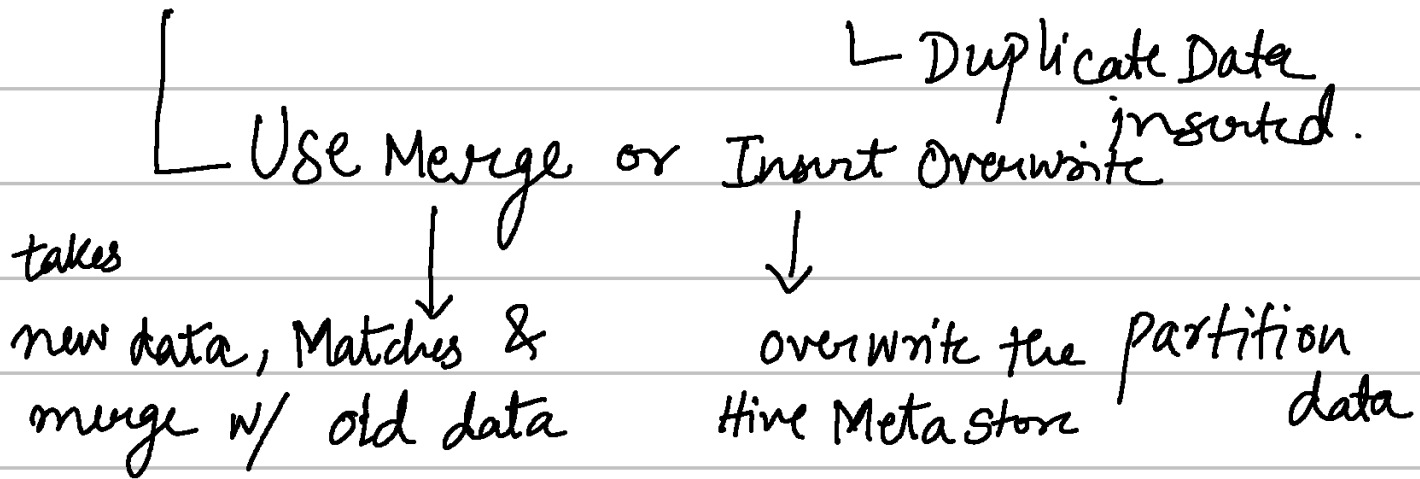
Why is it difficult to troubleshoot  
a non-idempotent table?



- it fails silently
- it produce inconsistent data

What can make your pipeline not Truncate

\* insert into without Truncate



\* using start data > w/o corresponding end data <  
 if you Run today at 't'  
 yest - 1 day of data  
 next day - 2 day of data  
 + - 3 day ...  
 ..  
 & so on

When "  
 > it is =  
 < run

\* Not using "full set" of partition sensors  
 L pipeline run when there is no /  
 in backfill full data will come Partial data

\* Not using "depends-on-past" for cumulative  
 if yesterday data is not there L cannot process in parallel  
 Pipelines

\* classify fake accounts at Fb

can be legit  
then go back being  
fake

dim-all-fake-accounts

not idempotent

SCD

dim-all-users →

- relied on latest data  
from pipeline

- needed this data ASAP

↓  
When it was behind  
dim-all-fake-accounts  
was using yesterdays data - Not okay for  
Analytics

inflow & outflow of fake users didn't make  
sense ∴ it was using yesterdays data some time

### Exception

- Relying on the "latest" partition

When Backfilling w/ properly modelled SCD Table

\* Silent failures

\* Issues of backfilling w/ non-idempotent pipelines.

\* Unit testing cannot replicate the production  
behaviour

< check now? > if pipelines are  
not idempotent

∴

SCD - Changes over Time  
Age

not idempotent.

How to Model?

- Latest Snapshot - Backfill it will be issue
- Daily/Monthly/Yearly Snapshot
- SCD

Storage is so cheap - just do daily snapshot (Max Article Creator of Airflow)

Collapsing Daily Snapshot when data changes

Annu	Age	Year
	12	2012
:	12	2012
:	12	2012
:	:	:
	365	2012

→ Annu 12 Jan 2012 - Jan 2013

1. Singular Snapshot - Don't do this, Backfill will be major issues
2. Daily Partitioned Snapshot
3. SCD 1, 2, 3

idempotent Type - 0 - Not changing Dim like DOB

Type - 1 only store latest values - Don't use in Analysis  
— OLTP - current value is okay

Type 2 — Gold Standard by Airbnb

Used  
idempotent L what the value was from start date  
to end date  
very far in future  
or  
NULL  
when it  
changed

is - current - boolean - current value?

More than one row per dimension, need to be careful about filtering on time.

Type 3 - You care "original" & "current" value  
if dim changes more than once then  
what?

You loose out on history - but have 1 row per dimension.

SCD 2 Loading

1. Load entire history in data

2. Cumulative way data (incrementally)

Process one new data at a time - no need for processing all history all the time - if data is small they will be same.

Unit Economics Table - Airbnb - SCD2  
Pay → Profit → Refund → Start Time → End Time

⇒ Focus on Business Needs

Marginal Values << Impact

LAB 2

Was processing everything at one time  
Wanted to do incrementally.

## Graph Data Modelling

Data Layer

- how things are connected

- no schema

- edge & Property

└ iceberg / Trino

- Additive vs Non-Additive Dimensions

- Enums - scoring class

- flexible data types - Map - Key Value - can vary

- Struct X define datatypes

- Graph Data Modelling - Array - Mid

Additive vs Non-Additive -

| All subtotals - "Don't Double Count"

$1 + 2 + 3 + 4 + \dots + 100 \text{ year olds} = \text{total Population}$

All Honda Driver  $\neq$  WRV drivers + Civic Drivers +  
But = WRVs + Civic cars. └ can have overlap.

if one entity can assume value in two subgroups then non-additive.

How it helps ?

\* You don't have to do count (DISTINCT)  
Pre-aggregated dimension - don't need to go to raw layer

\* Mostly in count NOT Sum  
\* Most dimensions are additive

# of users on app  $\neq$  # of user on iPhone + # users Android  
└ can have one user on both

# Analytics & Growth - User counting.

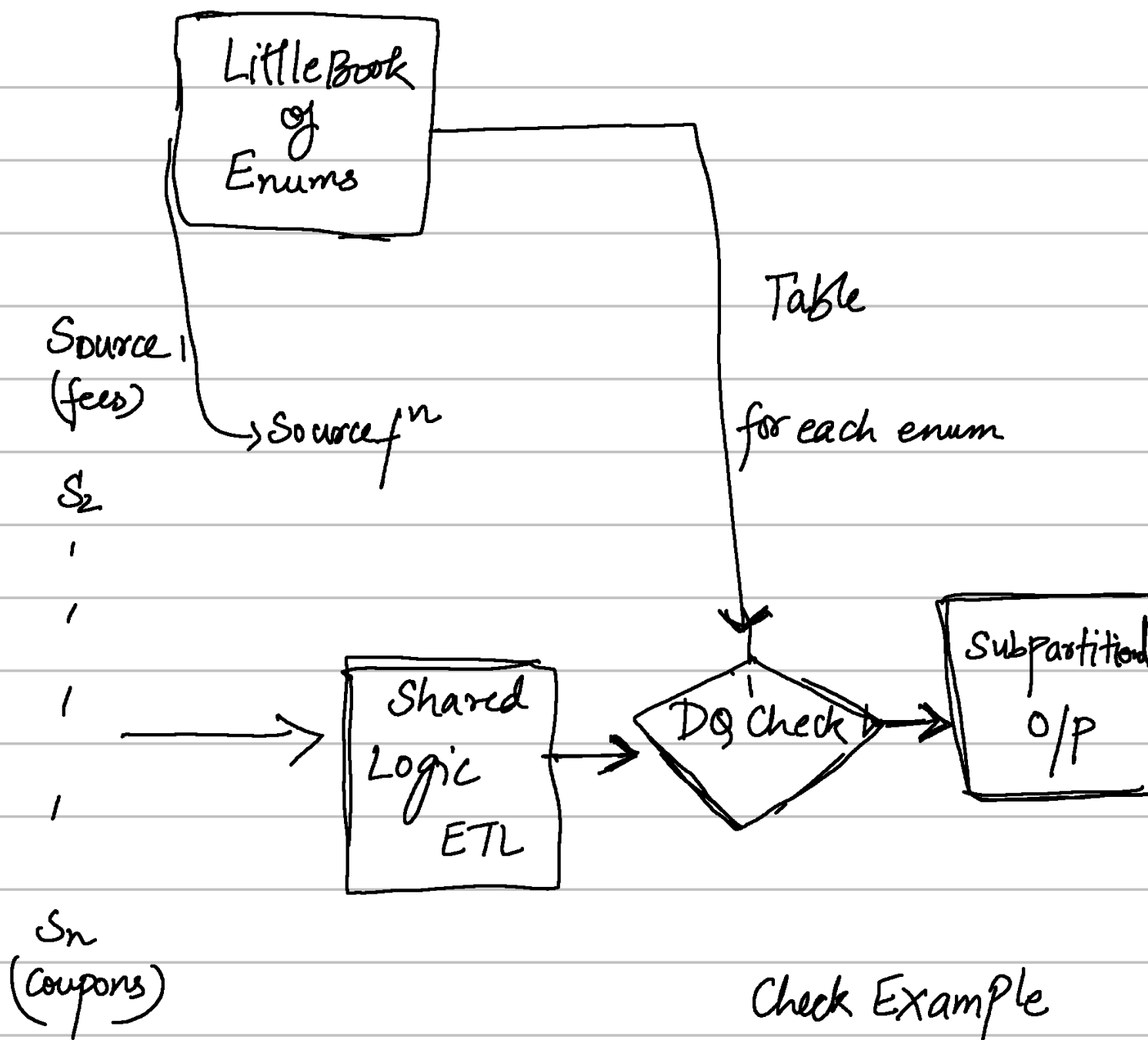
## When should you use enums?

- \* There is a limit to enumerate -  $< 50$  ✓
- \* country - NO!
- \* built in data quality
- \* built in static fields - Unit economics
- \* built in documentation
  - ↳ revenue or cost

channel - { Push, email, sms, logged out push }

Partition on Data & channel / Helpful in Dedup  
logging layer → ETL Layer  
↳ Thrift - Manage schema

Little Book of Pipeline - Enum Design Pattern  
V - Variety 50 upstream Datasets  
↳ enum.



Shared Schema - for all DataSources  
how to build it  
flexible schema - MAP datatype  
Vertex type = enum type. - ?

Limit of Map - 65000 Keys  
Java Limit



- Not lot of NULL columns
- Other Properties Column. - rarely used but needed columns

- Worst compression in MAP & JSON

↳ column header is stored as data

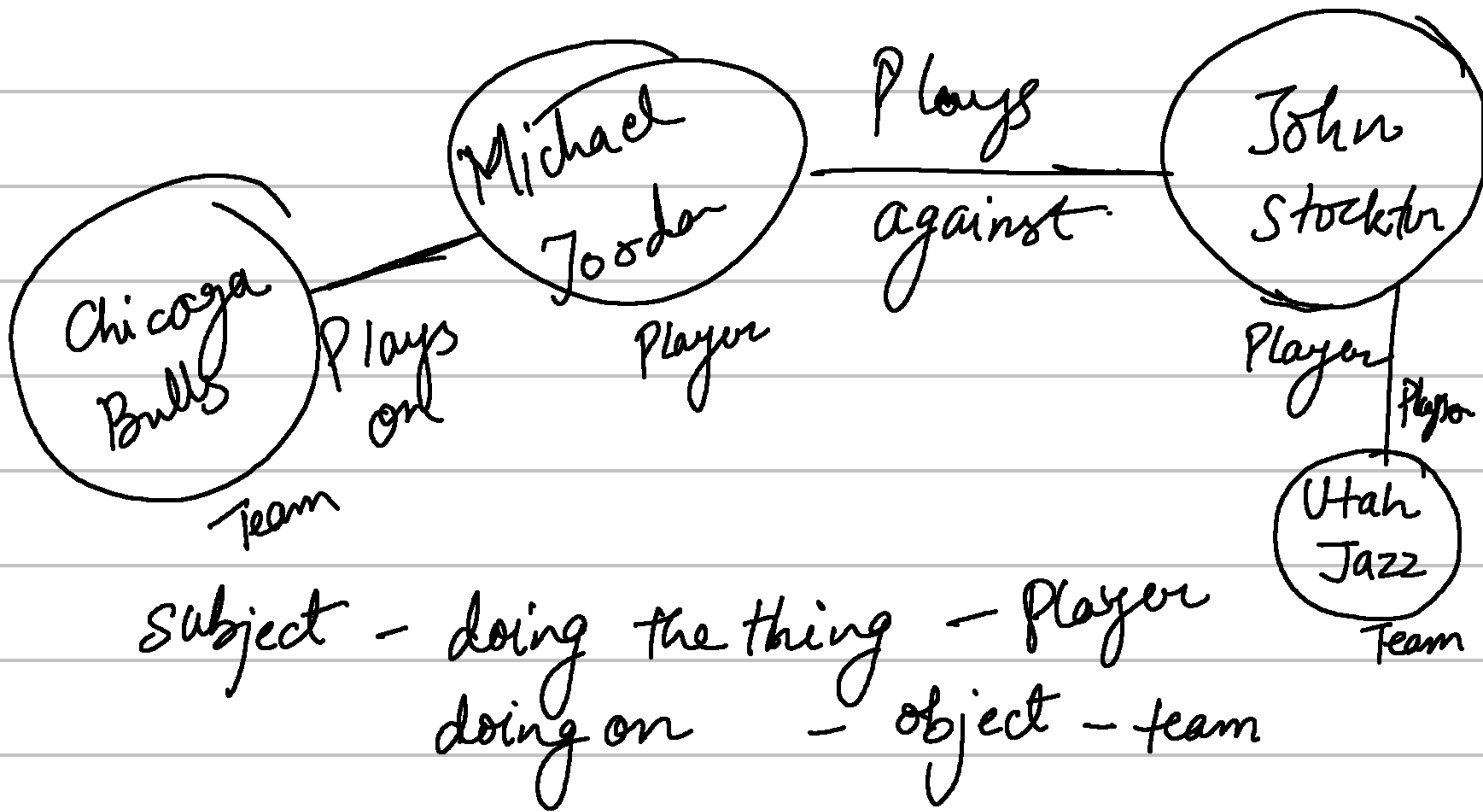
Graph data Modelling is

- Relationship Focused Not Entity Focused

identifier - String	} Vertex Schema
type - String	
Properties - Map <String, String>	

This schema can work for most graph data Modelling.

sub-identifier String	} Edge Schema
sub-type vertex-type	
object-identifier String	
object-type vertex-type	
Edge type : Edge-type	
Properties : MAP <String, String>	



No Map in Postgres use JSON.

Array AGG

Json-build-object ( )

Setup data - Partition & row number.

- MAX (CAST(e.properties → 'pts' AS Integer)
- QUALIFY.

Self Join  
 ⌞ f1 f2  
 on game-id,

1. Dedup
2. Division by 0
3. NULL