Complex Data Type
Week - 1

Dimension ?

- attributes of an entity (user's DOB fav. food)
- Slowly changing - fav food
- fixed | DOB / manufacturer / never changes

identifier Dim - SSN / uniquely identify the entity.

- Who is the consumer? = Data Modelling
  1. Analyst / D.S.
     - Easy to query (Analytical Dataset OLAP Cube)
     - Not complex
  2. Other Data Engineers
     - Master datasets
     - Should be compact
     - harder to query
     - Nested types are okay

③ — ML Models
    └ Identifier & flat primitive types

④ └ Customers
    └ Charts or geometric patterns
      No data need to be given

"How the data is gonna be used ?"
                Downstream Usage.

Dimensional Data Modelling

- OLTP (online transaction processing)
  SE do this type
  3NF, P.K, F.K, Joins - one user

- OLAP - most common for DE
  Run a query fast is the aim
  entire dataset is being looked at

- Master Data - middle of OLTP & OLAP
  - Deduped
  - optimitive for completeness of entity definition

Transactional System – 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 Transactio
└ Joins – Expensive – too much for Analytical

∴ Master Data can help you – Agility to go whereever you want.

Production Database → Master Data → OLAP cubes → Metrics

Snapshots
Transactional Data
for host
guest
prices
Availabity.
⋮
40 Datasets → 1 Table

Merge to M.D.
easy to understand pricing & availability

Flatten Data
Multiple entries for 1
By country, O.S.
Slice & Dice ✓

Aggregate & give one number
Avg listing price for all AirBnb

Cumulative Table Design (Master Data)

(some days a user might not show up but you still need that user)

holding to all dimensions that ever existed.

hold to history.

- Core components
  - 2 dataframe (Today & yesterday) — all same columns
  - Full outer Join the two dataframe
  - COALESCE values to Keep everything around
  - Hang onto all of history

- Usage
  - Growth Analytics at FB (dim_all_users) (10,000 downstream pipelines.
  - State Transition Tracking — Active yesterday Not Active Today — Churn
    N.A yesterday Active today — Ressuractive

filtration is req.

yesterday

can have NULL if you are starting today

1. Full outer Join
2. Coalesce to get entry
3. Compute cumulation metrics
4. Combine array & Changing Values

Today

feed to yesterday tomorrow.

Strengths of Cumulative Table Design

- Historical Analysis w/o shuffle
  - you don't need to group by. Why? How?
- Easy "transition" Analysis

Drawbacks

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

Compactness vs Usability Tradeoff

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

They wan't to reduce I/O

SE focussed

→ Online Systems where latency & vols matter

No complex data types

Easily Manipulated

Analytics focussed

→ OLAP Cube

<u>Middle Ground</u>

Use complex D.S. (Map, Array, Struct), querying difficult but compact

Master Data

⇒ Struct   Table inside a Table

Define keys & value

⇒ Map   Same Datatype for values

65,000 no. of keys — Random

Array - ordered list
Ordinal
all have to be same type.

Temporal Cardinality Explosion of Dimensions

Listing - Calendar
└ per night.
next year fwd - 2 billion nights

6M

$365 * 6M = 2B$

365 →

6M ↓ sorted
L1 -
L2 -
L3 -

365	L1	L2	L3	L4
N1 -		✓	✓	✓
N2				
N3				

Join in spark - mix up the ordering of rows & ruin your compression

Sort your data again
But should you ?
└ What about downstream DEs ?

Seed Query for cumulation

Array Concat — ROW = Season Stats

(
Joined for all years the players are playing for

Check out the case when.

Postgress file

Slowly Changing Dimensions

food preferences

imp to model correctly

idempotency — ability to reproduce same results in prod or backfill no matter when you run it / no. of times you run it.

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

Why difficult?

- it fails silently
- it produce inconsistent data in your pipeline if you have

* insert into without Truncate
  - └ Duplicate Data inserted.
  - └ user Merge or Insert Overwrite

takes new data, Matches & merge w/ old data

overwrite the data Hive Meta store

* using start data > w/o corresponding end data <

running today (t)

yest - 1 day of data
nextday - 2 day of data
+ - 3 day ...
..
& so on

When it is run

* Not using full set of partition sensors
  - └ pipeline run when there is no/partial data

* Not using "depends_on_past" for cumulative pipelines

yesterday data is not there

* classify fake accounts at Fb

can be legit
then go back to fake

dim_all_fake_accs

not idempotent

dim_all_users → - rely on latest data from pipeline
- landing time ↓

When it was behind
dim_all_fake_accounts
was using yesterdays data

inflow & outflow of fake users didn't make sense.

Exception

- Relying on the "latest" partition of anything else
Backfilling w/ SCD table

* Issues of backfilling w/ non-idempotent pipelines.
* Unit testing cannot replicate the production behaviour if pipelines are not idempotent

└ check how?

SCD — Age

How to Model?

- Latest Snapshot — Backfill it will be issue — not idempotent.
- Daily/Monthly/Yearly Snapshot
- SCD

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

Collapsing Daily based on weather the data changes.

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

→ Annu 12 Jan 2012 - Jan 2013

- Daily Partitioned snapshot
- SCD 1, 2, 3

Type - 0 — Not changing Dim like DOB

Type - 1 Only store latest values

OLTP — current value is okay

Type 2 — Gold Standard by Airbnb
L what the value was from start date to end date

very far in future or NULL

when it changed

is_current_boolean.

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.

<u>SCD 2 Loading</u> - 1. One giant Query - All data
2. Cumulative way data -
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

was processing one time

⇒ Focus on Business Needs

Marginal Values << impact

LAB 2.

Graph Data Modelling

- how things are connected
- no schema
- edge & Property

Data Layer

└ iceberge / 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

Addi^n vs Non-Addi^n -

All subtotals - "Don't Double Count"

1 + 2 + 3 + 4 + ... 100 year olds = total Population

All Honda Driver ≠ WRV drivers + Civic Drivers + ↳ can have overlap.

= WRVs + Civic Cars.

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

<u>How it help?</u>

* 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 ≠ # of user on iphone + # users Android

can have 1 user on both

Analytics & Growth - User counting.

When should you use enums?

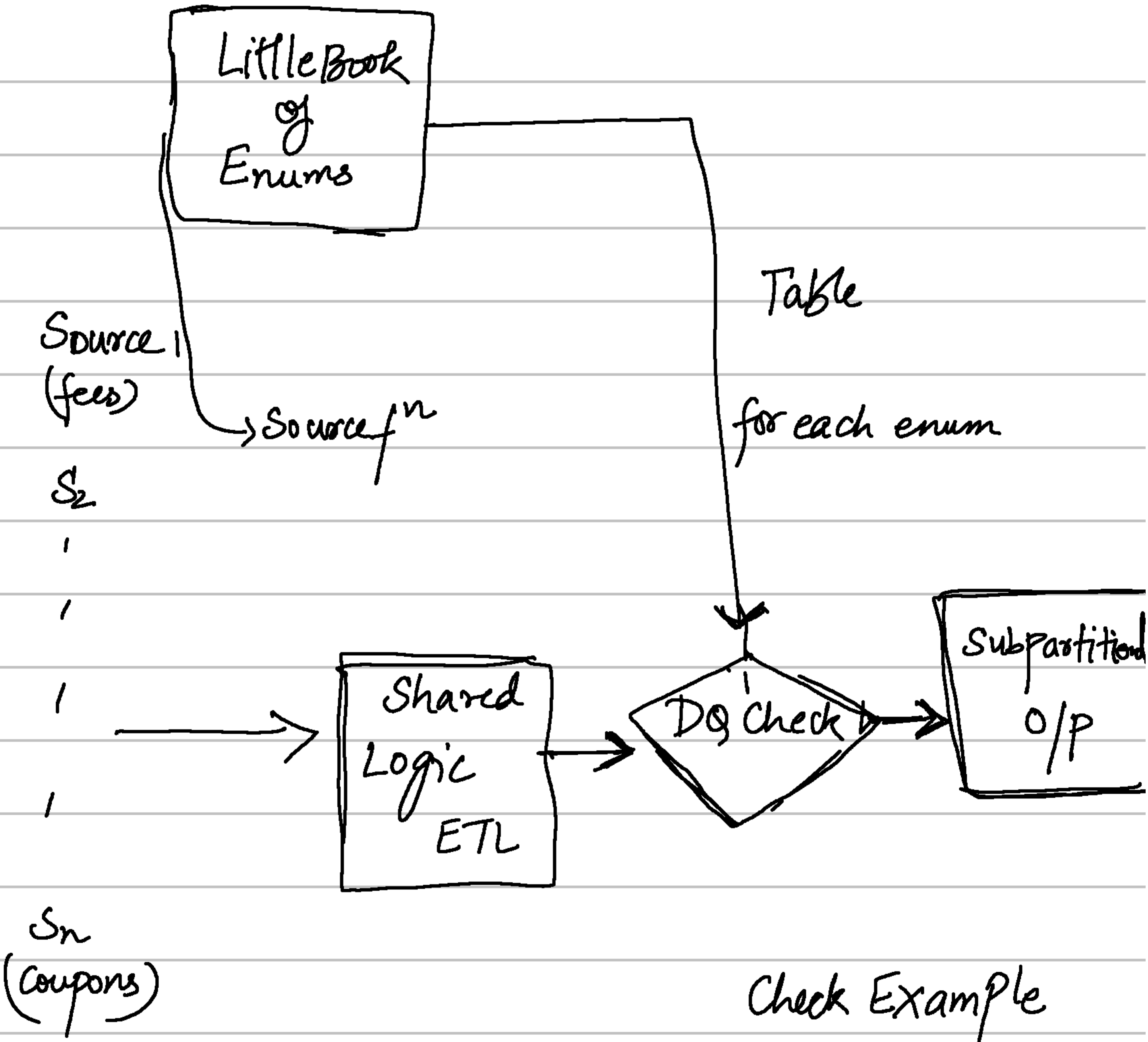
* there is a limit to enumerate - < 50 ✓
* country — NO!
* build in data Quality
* built in static fields — Unit economics
* built in documentation
  └ revenue or cost

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.



Check Example on github.

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 colums
- 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

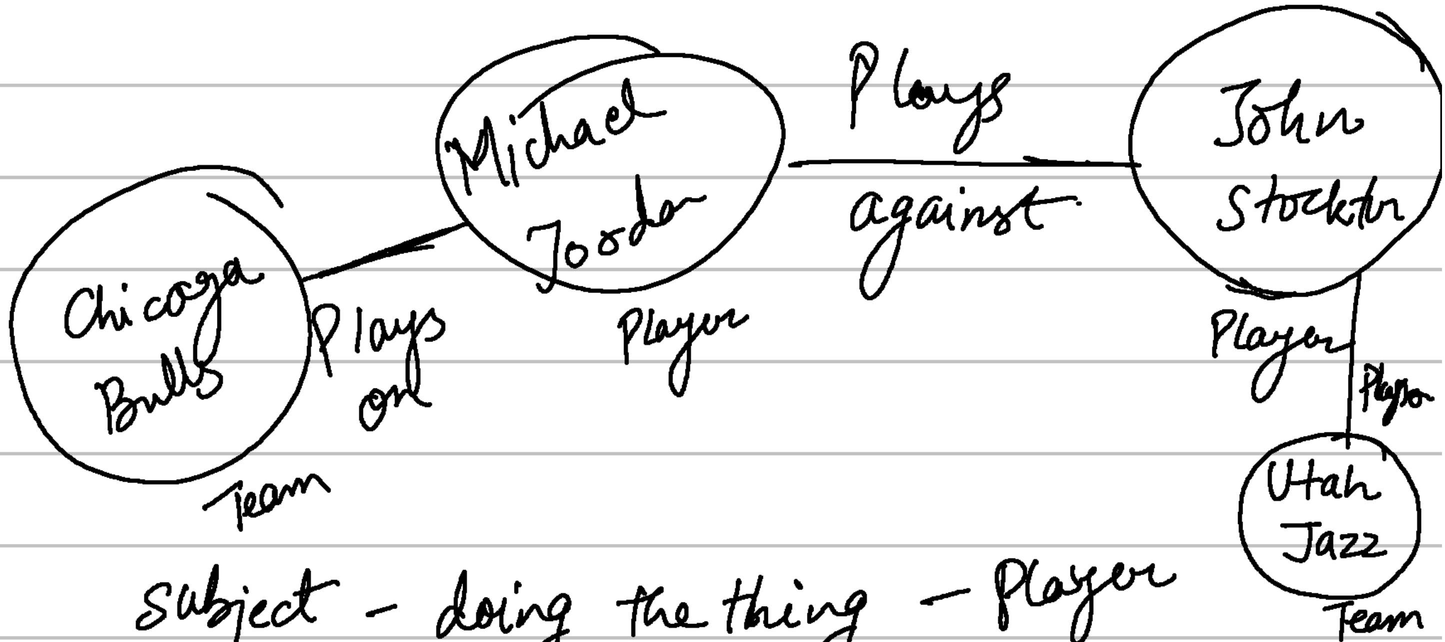
identifier - String
type - string
Properties - Map <String, string>

Vertex schema

This schema can work for most graph data Modelling.

sub_identifier string
sub-type vertex_type
object_identifier string
object_type vertex_type
Edge type : Edge-type
Properties : MAP <String, String>

Edge schema



subject - doing the thing - Player
doing on - object - team

No Map in Postgres use JSON.

Array AGG
Json_build_object (   )

Dedup data - Partion & 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.

i]

jsc