Complex Data Types

Week - 1

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? ≥ 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
   - Identifier & 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?" — Downstream Usage] Understand

Dimensional Data Modelling

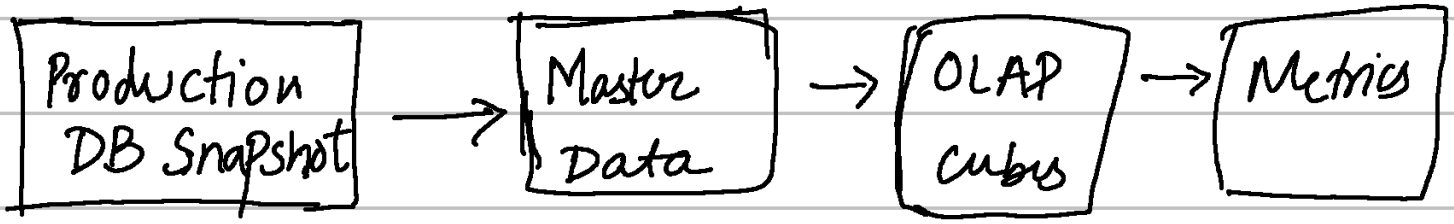
- OLTP (online transaction processing) (low latency, low volume queries)
  SWE do this type mostly
  3NF, P.K, F.K, Joins — one user
- OLAP — most common for DE (online Analytical Processing)
  - Run a query fast is the aim
  - entire dataset is being looked at
  - optimize for large volume
- Master Data — middle of OLTP & OLAP
  - Deduped
  - optimitize for completeness of entity definition

You had Transactional System $\xrightarrow{\text{but}}$ 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.



Transaction data for host guest prices Availability.

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

Flatten Data Multiple entries for 1 By country, O.S.

Aggregate & give one number

40 Datasets → 1 Table → Slice & Dice → Avg Listing price for all AirBnb

from Apps/ Website

Cumulative Table Design (Master Data)

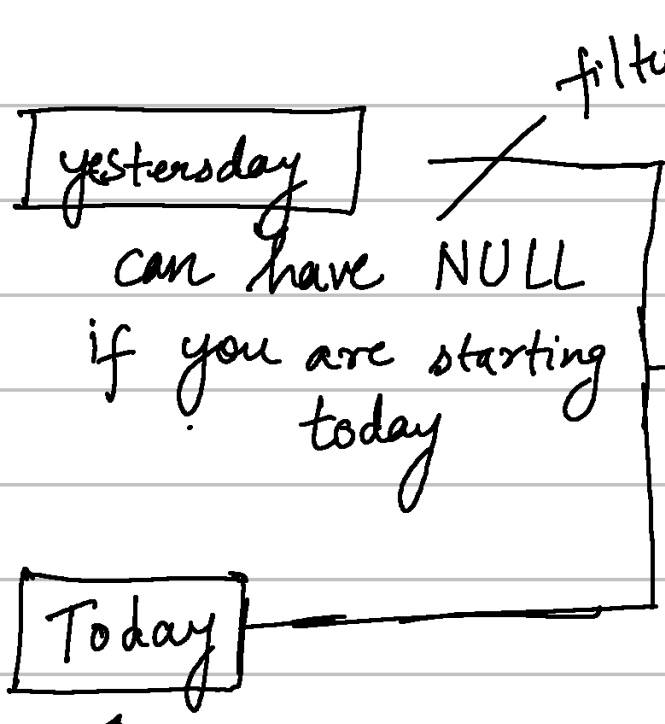
holding to all dimensions that ever existed.
hold to history.

(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) (all same columns w/ diff data)
   - Full outer Join the two dataframe
   - COALESCE values to keep everything around
   - Hang onto all of history

2. Usage (10,000 downstream pipelines.)
   - Growth Analytics at FB (dim_all_users)
   - State Transition → Active yesterday Not Active Today = Churn
     Tracking

     Not Active yest, Active today = Ressuractive



1. Full Outer Join
2. COALESCE to get correct value on id :: some ids will not be there
3. Compute cumulation metrics
4. Combine arrays & changing values
5. cumulative o/p of today will become yest. input

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 :: it is captured

Drawbacks

- can only be backfilled sequentially, not likely.
- 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

/ | 365 days

~ 6 million

Want to calculate nightly pricing & availability for next year?

365 * 6M = 2B nights

How to Model this?

365 Nights →

6M ↓ L1 –, L2 –, L3 –

Sorted Listing level w/ array of nights?

365 ↓ N1 –, N2, N3

6M: L1 ~ L2 ~ L3 ~ L4 ~

Night level with 2Billi rows?

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 Cummulation in Code *

Array Concat → ROW = Season Stats

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

Primary Key (Player, curr_season)
with yesterday ( )
with Today ( )

Problem: The table has multiple entries for a player if we join w/ something downstream too many rows. soln = one row per player

Slowly Changing Dimensions

Lec-2

| imp to model correctly

└ food preferences

* 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
  - Duplicate Data inserted.
  - Use Merge or Insert Overwrite
    - takes new data, Matches & merge w/ old data
    - overwrite the partition Hive Metastore data

* using start data > w/o corresponding end data <
  - if you Run today at '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 in backfill full data will come

* Not using "depends_on_past" for cumulative
  - if yesterday data is not there
  - pipelines cannot process in parallel

* Classify fake accounts at Fb

Can be legit then go back being fake

dim_all_fake_accounts

not idempotent

SCD

- relied on latest data from pipeline
- needed this data ASAP

dim_all_users →

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 sometime

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 behaviors

(Check how?) 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 @ Airflow)

Collapsing Daily snapshot when data changes

	Age	Year
Annu	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 — No Temporal component

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

Type 2 — Gold Standard by Airbnb
↳ what the value was from start date to end date
≈ Used idempotent
end date: very far in future or NULL
start date: 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.

SCD2 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

Was Processing everything at one time wanted to do incrementally.

⇒ 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 ✗ define datatypes
- Graph Data Modelling – Array – Mid

Addi~ vs Non-Addi~ -

All subtotals – "Don't Double Count"

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

All Honda Driver ≠ WRV drivers + Civic Drivers +
(Civic Drivers: can have overlap)

But = WRVs + Civic Cars.

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 ≠ # of user on iphone + # users Android
(iphone + 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!
* build in data quality
* built in static fields — Unit economics
  - revenue or cost
* built in documentation

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

Partition on Data & channel / Helpful in Dedup

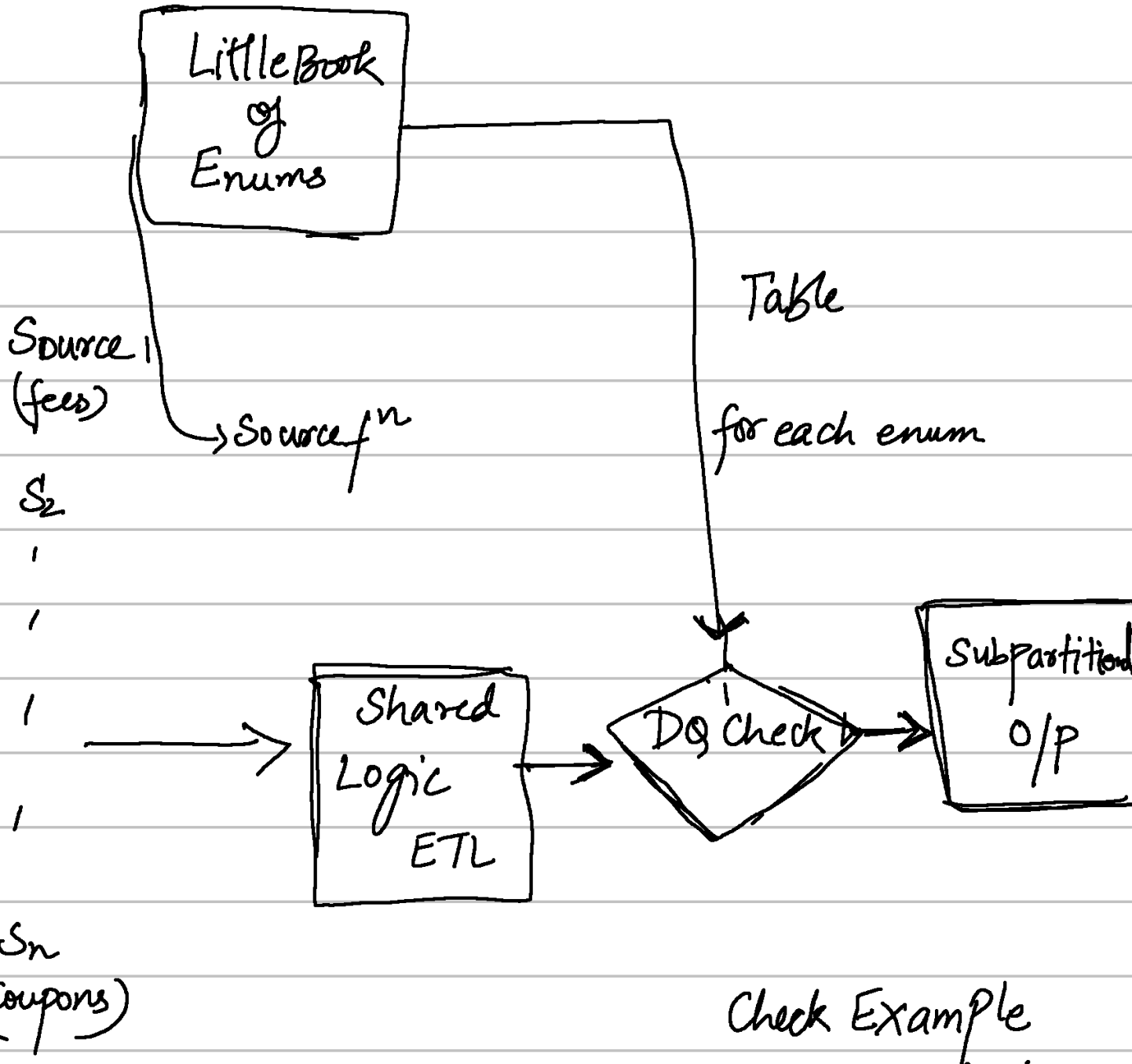
logging layer → ETL Layer

- 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

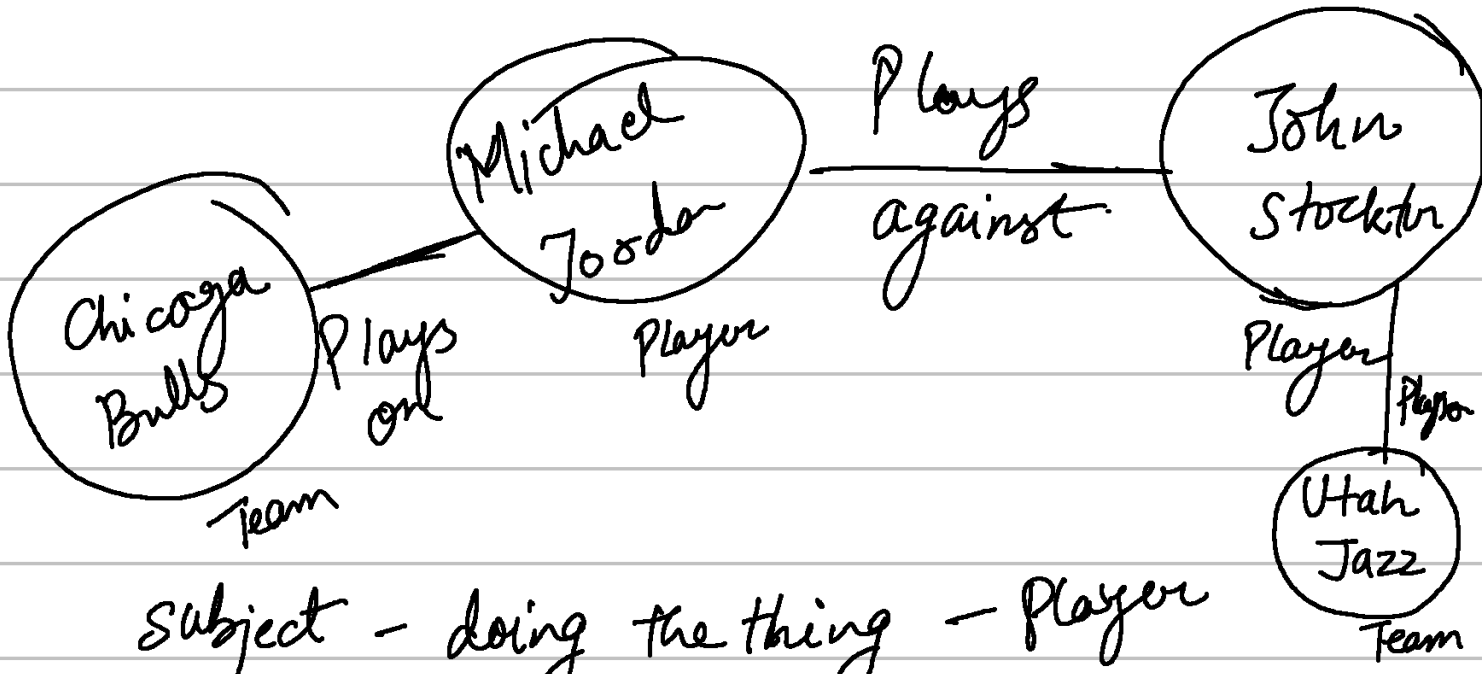
identifier - String
type - String
Properties - Map <String, String>

] 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 - 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