Dimensional Data Modeling

Day 3

EcZachly Inc

What are we talking about today?

- Additive vs non-additive dimensions
- The power of Enums
- When should you use flexible data types?
- Graph data modeling

What makes a dimension additive?

- Additive dimensions mean that you don't "double count"
- For example, age is additive
  - The population is equal to 20 year olds + 30 year olds + 40 year olds ….
- Application interface is NOT additive
  - The number of active users != # of users on web + # of users on Android + # of users on iPhone
- Counting drivers by cars is NOT additive
  - The number of Honda drivers != # of Civic drivers + # of Corolla driver + # of Accord drivers …

The essential nature of additivity

- A dimension is additive over a specific window of time, if and only if, the grain of data over that window can only ever be one value at a time!

How does additivity help?

- You don’t need to use COUNT(DISTINCT) on preaggregated dimensions
- Remember non-additive dimensions are usually only non-additive with respect to COUNT aggregations but not SUM aggregations

When should you use enums?

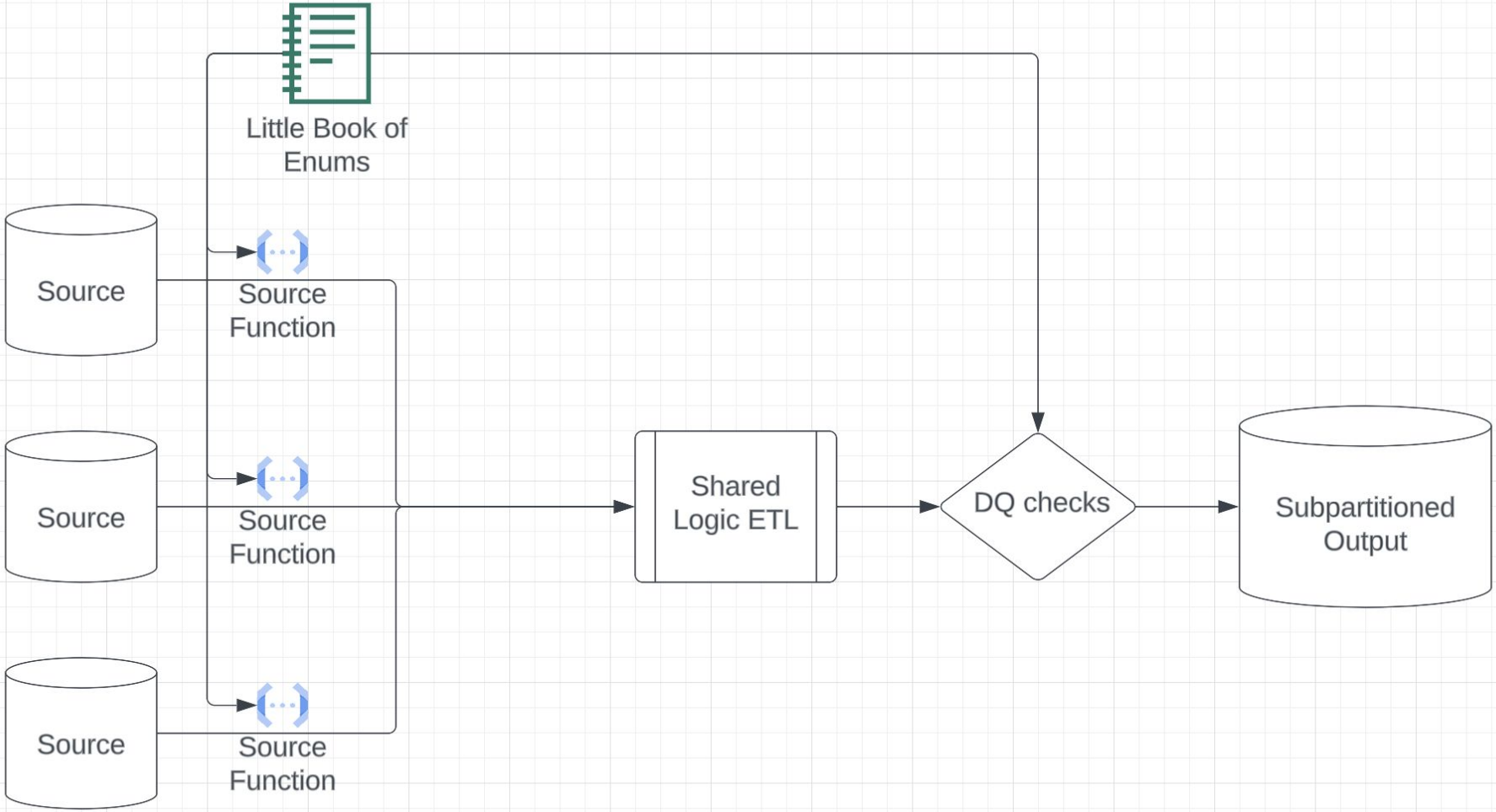
- Enums are great for low-to-medium cardinality
- Country is a great example of where Enums start to struggle.

Why should you use enums?

- Built in data quality
- Built in static fields
- Built in documentation

Enumerations and subpartitions

- Enumerations make amazing subpartitions because
  - You have an exhaustive list
  - They chunk up the big data problem into manageable pieces
- The little book of pipelines example


Little Book of Enums
Source
Source Function
Source
Source Function
Source
Source Function
Shared Logic ETL
DQ checks
Subpartitioned Output

What type of use cases is this enum pattern useful?

- Whenever you have tons of sources mapping to a shared schema
  - Airbnb:
    - Unit Economics (fees, coupons, credits, insurance, infrastructure cost, taxes, etc)
  - Netflix:
    - Infrastructure Graph (applications, databases, servers, code bases, CI/CD jobs, etc)
  - Facebook
    - Family of Apps (oculus, instagram, facebook, messenger, whatsapp, threads, etc)

How do you model data from disparate sources into a shared schema?

EcZachly Inc

Flexible schema!

Flexible schemas

- Benefits
  - You don’t have to run ALTER TABLE commands
  - You can manage a lot more columns
  - Your schemas don’t have a ton of “NULL” columns
  - “Other_properties” column is pretty awesome for rarely-used-but-needed columns
- Drawbacks
  - Compression is usually worse, (especially if you use JSON)
  - Readability, queryability

How is graph data modeling different?

Graph modeling is RELATIONSHIP focused, not ENTITY focused.

- Because of this, you can do a very poor job at modeling the entities.
  - Usually the model looks like
    - Identifier: STRING
    - Type: STRING
    - Properties: MAP<STRING, STRING>

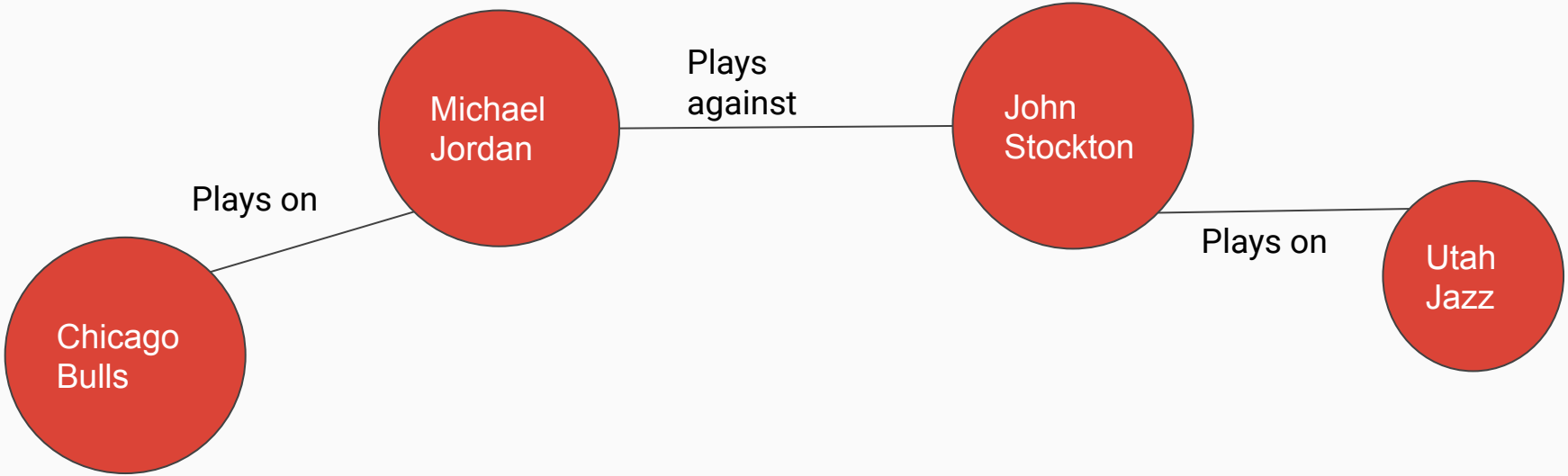
How is graph data modeling different?

Graph modeling is RELATIONSHIP focused, not ENTITY focused.

- The relationships are modeled a little bit more in depth
  - Usually the model looks like
    - subject_identifier: STRING
    - Subject_type: VERTEX_TYPE
    - Object_identifier: STRING
    - Object_type: VERTEX_TYPE
    - Edge_type: EDGE_TYPE
    - Properties: MAP<STRING, STRING>

Graph Diagram

EcZachly Inc



The lab today

In the lab today we'll be building a graph data model out of the NBA data sets we've been working with so far!