# Dimensional Data Modeling

Day 1

EcZachly Inc

# What is a dimension?


- Dimensions are attributes of an entity (e.g. user's birthday, user's favorite food)
  - Some of these dimensions may IDENTIFY an entity (e.g. a user's ID)
  - Others are just attributes
- Dimensions come in two flavors
  - Slowly-changing
  - Fixed

# What we'll cover today


- Knowing your data consumer
- OLTP vs OLAP data modeling
- Cumulative Table design
- The compactness vs usability tradeoff
- Temporal cardinality explosion
- Run-length encoding compression gotchas

# Knowing your Consumer


- Data analysts / Data scientists
  - Should be very easy to query. Not many complex data types
- Other data engineers
  - Should be compact and probably harder to query. Nested types are okay
- ML models
  - Depends on the model and how its trained
- Customers
  - Should be a very easy to interpret chart

# OLTP vs master data vs OLAP


- OLTP (online transaction processing)
  - Optimizes for low-latency, low-volume queries
- OLAP (online analytical processing)
  - Optimizes for large volume, GROUP BY queries, minimizes JOINs
- Master Data
  - Optimizes for completeness of entity definitions, deduped

# Mismatching needs = less business value!!!!


- Some of the biggest problems in data engineering occur when data is modeled for the wrong consumer!

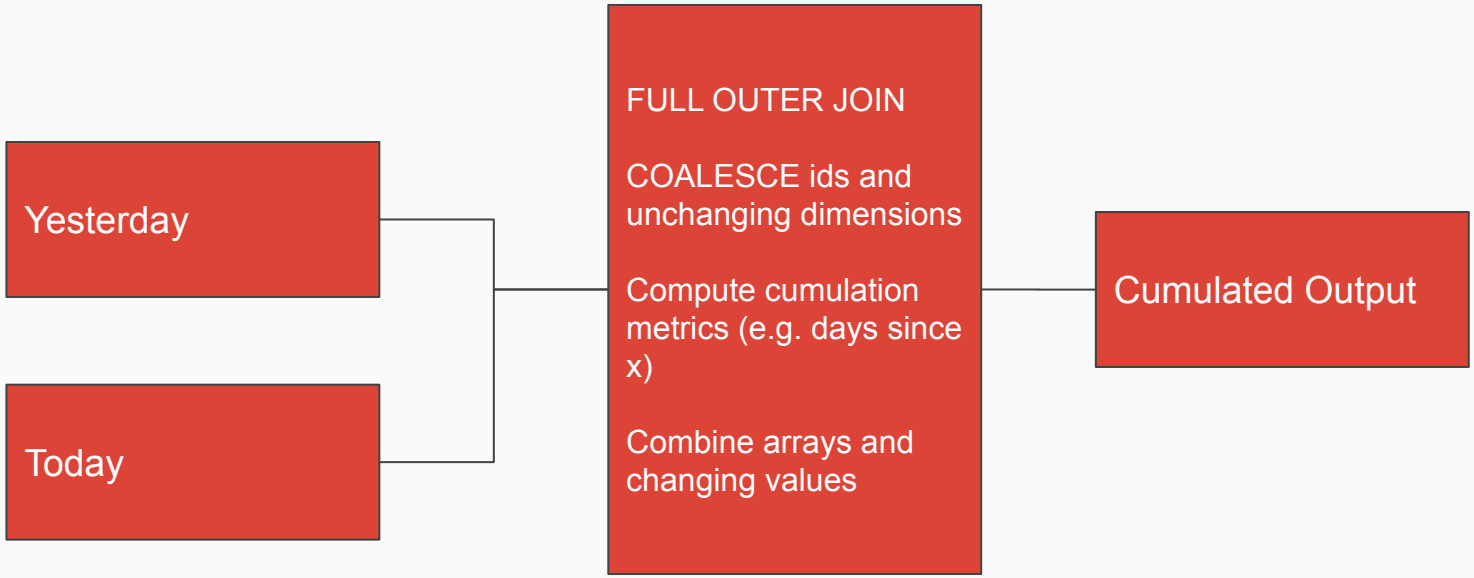
# OLTP and OLAP IS A CONTINUUM


# Cumulative Table Design


- Core components
  - 2 dataframes (yesterday and today)
  - FULL OUTER JOIN the two data frames together
  - COALESCE values to keep everything around
  - Hang onto all of history
- Usages
  - Growth analytics at Facebook (dim_all_users)
  - State transition tracking **(we will cover this more in Analytics track, Applying analytical patterns later)**

# Diagram of cumulative table design

EcZachly Inc



# Cumulative Table Design (cont)


- Strengths
  - Historical analysis without shuffle
  - Easy “transition” analysis
- Drawbacks
  - Can only be backfilled sequentially
  - Handling PII data can be a mess since deleted/inactive users get carried forward

# The compactness vs usability tradeoff


- The most usable tables usually
  - Have no complex data types
  - Easily can be manipulated with WHERE and GROUP BY
- The most compact tables (not human readable)
  - Are compressed to be as small as possible and can’t be queried directly until they’re decoded
- The middle-ground tables
  - Use complex data types (e.g. ARRAY, MAP and STRUCT), making querying trickier but also compacting more

# The compactness vs usability tradeoff


- When would you use each type of table?
  - Most compact
    - Online systems where latency and data volumes matter a lot. Consumers are usually highly technical
  - Middle-ground
    - Upstream staging / master data where the majority of consumers are other data engineers
  - Most usable
    - When analytics is the main consumer and the majority of consumers are less technical

# Struct vs Array vs Map


- Struct
  - Keys are rigidly defined, compression is good!
  - Values can be any type
- Map
  - Keys are loosely defined, compression is okay!
  - Values all have to be the same type
- Array
  - Ordinal
  - List of values that all have to be the same type

# Temporal Cardinality Explosions of Dimensions


- When you add a temporal aspect to your dimensions and the cardinality increases by at least 1 order of magnitude
- Example
  - Airbnb has ~6 million listings
    - If we want to know the nightly pricing and available of each night for the next year
      - That's 365 * 6 million or about ~2 billion nights
    - Should this dataset be:
      - Listing-level with an array of nights?
      - Listing night level with 2 billion rows?
    - If you do the sorting right, Parquet will keep these two about same size

# Badness of denormalized temporal dimensions

EcZachly Inc

If you explode it out and need to join other dimensions,

Spark shuffle will ruin your compression!

# Run-length encoding compression

EcZachly Inc

- Probably the most important compression technique in big data right now
  - It's why Parquet file format has become so successful
- Shuffle can ruin this. **BE CAREFUL!**
  - Shuffle happens in distributed environments when you do JOIN and GROUP BY

| | season | player_name | age | height | college |
|---|---|---|---|---|---|
| 1 | 1996 | A.C. Green | 33 | 6-9 | Oregon State |
| 2 | 1997 | A.C. Green | 34 | 6-9 | Oregon State |
| 3 | 1998 | A.C. Green | 35 | 6-9 | Oregon State |
| 4 | 1999 | A.C. Green | 36 | 6-9 | Oregon State |
| 5 | 2000 | A.C. Green | 37 | 6-9 | Oregon State |

# Run-length encoding compression

EcZachly Inc

| | season | player_name | age | height | college |
|---|---|---|---|---|---|
| 1 | 1996 | A.C. Green 5 | 33 | 6-9 5 | Oregon State5 |
| 2 | 1997 | [redacted] | 34 | [redacted] | [redacted] |
| 3 | 1998 | [redacted] | 35 | [redacted] | [redacted] |
| 4 | 1999 | [redacted] | 36 | [redacted] | [redacted] |
| 5 | 2000 | [redacted] | 37 | [redacted] | [redacted] |

# Spark Shuffle

EcZachly Inc

After a join, Spark may mix up the ordering of the rows and ruin your compression

| | season | player_name | age | height | college |
|---|---|---|---|---|---|
| 1 | 1996 | A.C. Green | 33 | 6-9 | Oregon State |
| 2 | 1996 | Michael Jordan | 34 | 6-6 | North Carolina |
| 3 | 1997 | A.C. Green | 34 | 6-9 | Oregon State |
| 4 | 1997 | Michael Jordan | 35 | 6-6 | North Carolina |
| 5 | 1998 | A.C. Green | 35 | 6-9 | Oregon State |
| 6 | 1999 | [redacted] | 36 | [redacted] | [redacted] |

# Let’s start the workshop!

EcZachly Inc