Fact Data Modeling

What is a fact?

- Something that happened
  - User logins
  - Transactions that are made
- Events (atomic pieces)

Fact Data 101:

- Fact data is 10-100x the volume of dim data
  → X actions per dimension
- Context for effective analysis is key
- Duplicates are more common (dedupe needed)
  → challenging!

Normalized vs. Denormalized Facts

- Normalized facts have no additional dim attributes (JOINs on ID)
- Denormalized facts some dim data for faster analytics but higher storage
- The smaller the scale the better Normalization will be

Raw logs

- Used by Software Engineers
- Potential quality errors
- Short retention

$\xleftrightarrow{\text{Married}}$

Fact data

- Quality guarantees
- Nice schema
- Longer retention

$\Big\}$ Highly trusted

Modeling

- „Who" fields are usually represented as ID
- „Where" also IDs but is likely to bring in dims (especially high cardinality)
- „How" similar to Where field
- „What" fields are atomic events and fundament of fact data
- „When" fields defines timestamp of the event and is also fundamentally part of fact data
  ↳ Use UTC!

Key properties

- Data Quality guarantees – no duplicates, what and when not null
- Smaller than raw logs
- Columns should be easy to query and understand

Logging

- Only log what you really need
- Use shared schema in logs and pipeline
  ↳ middle layer like thrift

Options for high volume facts

Sampling

- metric driven use cases
- not when precision is needed

Bucketing

- can be bucketed by one important dim („Who" IDs)
- Sorted-merge bucket (SMB) joins without shuffle

Retention

- Dim data as long as allowed/needed
- Fact data very costly so retention needs to be defined
  → do not decide for „just in case"

Deduplication

- Define a time frame to „care" about duplicates
- Streaming allows to capture most of the duplicates efficiently
  → 15-60 minute windows are sweet spot

- Group by every hour to remove duplicates
  - Full outer join 2 hour windows
    - JOIN those into 4 hour windows
    - .... → Tree structure

Facts vs. Dimensions

- fact is event driven like user is active
- dims are state driven
- facts can be aggregated into dims
  → CASE WHEN to bucketize aggregated facts is useful to reduce cardinality
  ↳ 5-10 buckets sweet spot

Dimensions

- Usually show up in Group By
- Can be high and low cardinality
- Come from a snapshot of state

Facts

- Usually aggregated in analytics (SUM, AVG, COUNT)
- Higher volume than dimensions (1 user can make multiple actions)
- come from events and logs

Categorical Fact/Dimensions

- Class category derived from facts
- CASE WHEN logic and

Date List data structure (extremely efficient)

- Example cumulated schema
  - User_id
  - Date
  - Dates_active

  turn → into

  - User_id (32)
  - Date (2023-01-01)
  - datelist_int (1000010001)

-1 day

current day

active

Data Shuffling

- Should be minimalized to enhance parallelism
- Bottleneck for Big Data Processing
- Spark will use shuffle partitions to distribute data (default 200)

Parallelism

- Extremely parallel
  - SELECT, FROM, WHERE
- Kinda parallel
  - GROUP BY, JOIN, HAVING → all rows of Group By needs to be on one machine
- Not parallel
  - ORDER BY → all data on one machine (use only at the end after agg)

  exception → Window function with partition by

Efficiency

- Improve GroupBy by bucketing data on object storage (pre shuffle)
- Reduce data volume as much as possible
  - ↳ Start aggregating every action into daily/monthly/yearly data
  - ↳ Some analytics flexibility gets lost
  - ↳ Time horizons of aggregated data increases for analytics

Long-Array Metrics

- Super Powerful for huge history
- Monthly metrics with an array column for every day
- start_date/month_start as an index/offset
- Position in array determines the day of the month
- Similar to datelist for non binary data
- You need to pick snapshots in time