FACT/ DIMENSIONAL DATA MODELING

Blurry line between a dimension and a fact (can be even based on each other!)

Examples:

- 'dim_is_active' -> activity-driven (log-in event) ✓ Fact

  Q: Did the user have any activity? (ie. 'show up in the app for at least 1min')
- 'dim_is_activated' -> state-driven ✓ Dimension

Both metrics are important, and can be used separately or combined.

Which one matters most depends on what you aim to answer ('signups?', 'growth?')

📌 Fact vs. Dimensions: Properties

DIMENSION

- Generally derived from a snapshot of state and/or aggregation of facts.
- Outside aggregations.
- Show up in GROUP BY queries ('user_id', 'gender', 'scoring_class').
- Wide cardinality range: high ('user_id'), mid ('country'), low ('gender').

FACT

- Derived from logs or events
- Inside aggregations (SUM, AVG, COUNT).
- Can be aggregated and turned into dimensions (buckets using CASE WHEN) to reduce cardinality
- Can change dimensions.
- Generally 'higher' dimensions (ie. a user of an app can do many things -log in, like a post, watch a video, delete content...-).
- Low dimensions occasionally (rare events).

Facts or Dimensions at Facebook

- 'dim_is_active': Did the user have any activity? (ie. showed up in the app for 1min, showed < 1min but had an engagement -like/comment/...)

  ✓ Dimension
- 'dim_is_deactivated': a user being activated/ deactivated is not an aggregation. It changes the state of a user's account (an attribution to your user object)

  ✓ Dimension ✓ Fact

Pricing & Availability at Airbnb

Q: Price attribute of Airbnb listing: Fact or Dimension?

- Host sets/changes price settings (ie. 'weekday/weekend price', 'early-bird/discount promo')

  ⟶ logs an event ✓ Fact

BUT...

- **Dimension**: price as a state (attribute of the night), derived from settings set by the host. Particular case, as double or decimal type rarely a dimension (extremely high cardinality). ✓ Dimension

FACT/ DIMENSIONAL DATA MODELING

Other examples

- 'dim_bought_something' -> based on buy event (non-zero fact events) ✓ Fact
- 'dim_has_ever_booked', 'dim_ever_labelled_fake' -> ever logged/ showed up/ fake account? ✓ Dimension
- 'days_since_last_active', 'days_since_signup' ✓ Dimension
  (used in Retention Cohort Analytics)

📌 Bucketization

Create dimensions out of aggregated facts

- Difficult & expensive to change once set
- Helpful to reduce cases of extremely high cardinality

❓ Q: What is the cardinality of the dimension?

- Analyze distribution of data to determine number of buckets (ie. boxplots are helpful)
- Bucketize ~5-10 values max to keep data insights clear/relevant.
- Don't bucketize too little values (1) to avoid normality issues (predictive power loss)

Key takeaway: bucketize into at most 5-10 values to derive insights from buckets

The Date List data structure (very efficient)

Storing historical growth data at

Q: How many weekly/ monthly/ yearly active users Facebook has?

Facebook has 2 bio users. Each user has, on avg, ~ 50 facts/day (= 100 bio rows/day, 3 trillion rows/month)

Monthly base implied a daily processing if a user is active, check last 30 days of user Fact data, group by... expensive & ineffective

- Naive approach: having 'user_id', 'current_date', 'dates_active' as an array where user was active and add it cumulatively to the array. Has unnecessary array of dates
- Smart approach: Cumulative Table design

  Creates new table, processes 30 days, then next day drop the n-30 day & keep the rest as is

  Date list integer structure (datelist_int): 'user_id' + 'date' + 'datelist_int' (1=active; 0= inactive)

user_id	date	datelist_int
32	2024-01-01	100010110101100

  - Shows activity history
  - Efficient compression (stores n days of history in 1 array)

1st integer represents '2024-01-01' activity - bit position (0 indexed)

Albert Campillo

Repost