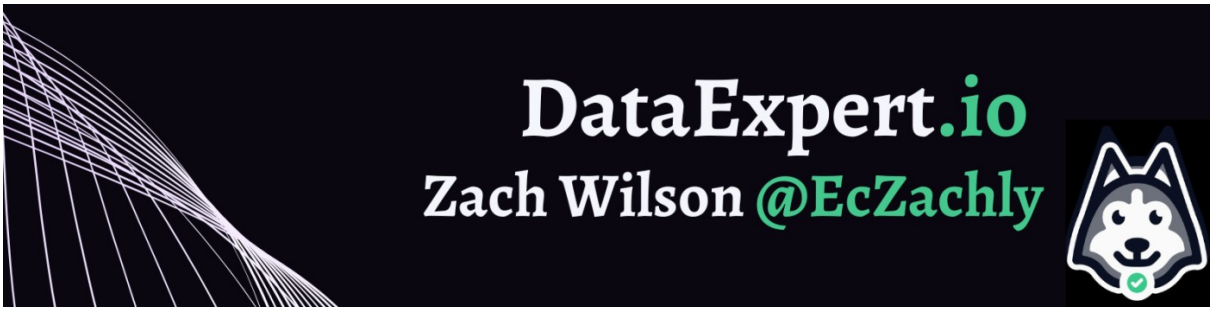

Facts Data Modelling - Day 1

# class	4
Topic	Data Modelling
Notes Status	completed
Completion Date	@December 2, 2024

What is a Fact?

- Something that happened or occurred.
- It's **atomic**, cannot be broken down into a smaller piece
- Facts are like past, **they don't change**

Example:

- User logs in
- Transaction is made

What makes Fact modelling hard?

- Huge volume of data and keeps scaling. Generally 10-100X of dimension data.
- How much fact data is gonna go up depends on how many actions a dimension can make (Ex: Facebook, 25 notifications per user, 2B active users daily = 50B notifications/ day).
- You need a context for effective analysis

- Notification sent alone doesn't mean anything, Notification sent → Clicked → Purchased makes more sense. Conversion funnels .
- CTR alone doesn't make much sense. CTR along with region, CTR is more in US than Canada(ex). Something to do analysis on.

> 💡 Google tried 40 different shades of blue on the links on their website and one shade of blue had more conversion rates resulting in some **millions of dollars increment in revenue**.

- Duplicates are way more common in fact datathan in dimensional data.
  - Duplicate due to some logging bug.
  - Genuine duplicates. Ex: Clicking a notification now and then clicking again 1 hour later. This will make CTR looks inflated and results in wrong interpretation of the data.

How Facts Modelling work?

Both Normalization and Denormalization have a place in Facts Data Modelling.

Facts and raw log are not the same thing

Raw logs

- Owned by engineers that don’t have strong data skills.
- Ugly schemas designed for online systems.
- Potentially contains duplicates and quality errors.
- Shorter retention

Fact Data

- Nice and descriptive column names.
- Quality guarantees like uniqueness, completeness, etc.
- Longer retention.

> Trust in the Fact Data should be in order of magnitude, higher than the trust in raw logs.

💡 One of the best thing you can do as a Data Engineer is to convert those raw logs into a highly trusted fact data

Understanding Who, What, Where, When and How of Facts

Who

- ID, who is the part of the event.

Where

- Location
- Where in the app, PC, Tablet, Phone.

How

- How and Where are sometimes similar, app, PC, Tablet, Phone.

What

- What fields are fundamentally part of the nature of the fact.
- In notification world : Sent, Clicked, Delivered.

When

- Mostly event_timestamp or event_date.

💡 **Important Note:** Timestamp should be common for all the data, preferably UTC

- Fact tables should have some quality guarantees. The What and When field should never be NULL, otherwise the data cannot be.
- Fact data should be smaller than raw logs.

- Fact data should parse out hard to understand columns. Should be STR, INT, DOUBLE, etc.

> *"Your job as a Data Engineer is to make the data fun and delightful to use. If you are not doing that, you're probably a bad data engineer."* 🫠

When should you model in Dimensions (When to de-normalize)

- **[IMPORTANT]** Network Logs pipeline example from Netflix (Embed the URL)
- Broadcast join is optimal if the smaller table is less than 5-6 GB.

> ***"A lot of time the impact of you have as a Data Engineer is not writing and optimizing a pipeline. It's actually going upstream and solving the problem at the source."***

How does logging fit into Fact data?

- Brings critical context for your fact data. Usually done in collab with system engineers.
- Do not log everything. If it's just in case, it will never be used.
- **[IMPORTANT]** Logging should conform to some sort of schema or contract.
- Thrift Schema: Specification that is language-agnostic. ❓ (Need more info).

Potential options when working with high volume fact data

- Do we need to use all the data or can we filter some of it?
- Sampling: Does not work for all use cases, works best for metric-driven use-cases.
- **Bucketing**
  - Fact data can be bucketed on **WHO** i.e. ID or a unique identifier.

- Bucket joins can be much faster as they minimize shuffle.
- **Sorted merge bucket (SMB) joins** can do joins **without shuffle** at all. This was more used during Hive days, in Spark shuffling is more optimized. But it's not gone, **it is still relevant.**

Fact Data Retention

- Any fact tables > 100TB had ~14 days or less retention.
- If size < 10 TB, retention does not matter as long as PII are masked.
- Depends on the company policy.

Deduplication of Fact Data

- Think about the timeframe.
- Window of deduplication : hour, day, week? how to decide?

Intraday deduping options

1. Streaming

- Captures most duplicates in a very effective manner for a short time
- A large memory of duplicate events happen within the short interval of the first event (from notification example).
- Entire duplicates can be harder for streaming because it needs to hold on to such a huge data. (This depends on the use case. If you have the smaller data, we can hold more).

2. Microbatch

Example below. 🔽

💡 Hourly Microbatch Dedupe (Zach's solution at Facebook)

- Reduced the latency from 9 hours to 1 hour⚡

Implementation 🎛️

Step 1

- Collect each hour data and dedupe with GROUP BY.

- Use SUM and COUNT to aggregate duplicates, use COLLECT_LIST to collect metadata about the duplicates that might be diiferent.

Step 2

- Dedupe between hours with FULL OUTER JOIN
    - Use left.value + right.value to keep duplicates aggregation correctly counting or CONCAT to build a continuous list (What ❓)

Step 3

- Keep merging until we have a final merged table.