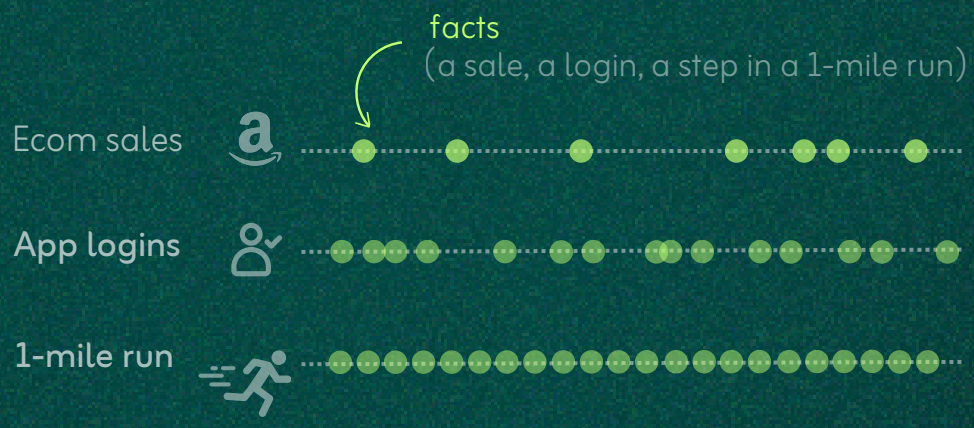
FACT DATA MODELING FUNDAMENTALS)

1 FACT ~ Truth

- Atomic (can't be broken down into smaller units)
- Can be aggregated up/down
- Not slowly changing
- Can't be changed back



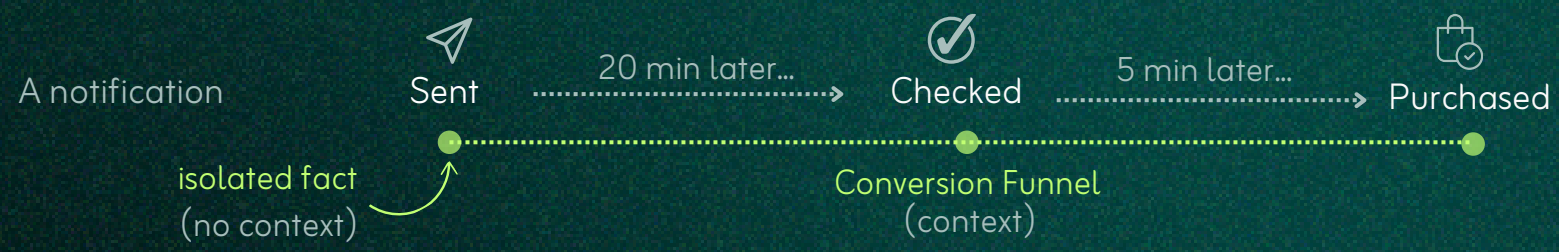
? Why Fact Data is hard to model?

1. Scalability (10-100x the amount of dimension data)

Q: How many actions taken per dimension?

Ie. : 2 bio users x 30 notifications/day ~ 50 bio notifications/day

2. Need more context for effective analysis



Q: What data dimensions do we need to make the fact data more valuable?

Ie. A user's location bring context (what's the clickthrough rate between US and Canada notification recipients?)

3. Duplicates are more common. Causes:

- Bugs

  Ie. A software engineer pushed a bug in the logger ·····› 1 click generates 2 logs ·····› Data Quality issue

- Genuine duplicates (can be handled via deduplication)

  Ie. A notification clicked now and clicked again 3 hours later

  Tracking 2 genuine actions without counting them as 2? ·····› deduplication

  No deduplication ·····› your metric will look weird! (ie. a 200% CTR)

? How Fact Data Modeling works?

Normalization

- Splits data across multiple tables
- Minimizes data redundancy
- Removes duplicates, increases data integrity

Denormalization

- Combines data into a single table
- Faster querying (use GROUP BY, not JOIN)
- High storage cost, potential duplications

Trade-off

FACT DATA MODELING FUNDAMENTALS)

Fact Data vs. Raw Logs: same thing?

- No, but both interconnected (no fact without raw logs)
- The level of trust of fact data should be higher than the trust in raw logs

Raw Logs

- Data for systems, server running.
- Usually owned by SEs (software engineers)
- Ugly schemas, potential duplicates, DQ issues
- Short retention

Fact Data

- Data for analysis.
- Usually owned by DEs (data engineers)
- Nice schemas/column names
- Quality guarantess (uniqueness, not nulls, ...)
- Longer retention

Goal as a DE: collaborate closely with SEs, be able to convert raw logs into highly trusted fact data

Aspects of Fact Modeling

A WHO ·····→ 'Who is part of that event?'

- 'Pushed out as IDs (ie. Tesla car ·····→ <'user_id', 'car_id', 'operating_system_id'>)

B WHERE ·····→ 'Where is the event taking place?'

- Modeled with ids (Who) + additional dimensions

  Ie. location (Country, City, State); app pages (Homepage, Profile Page... modeled w/ backlashes)

C HOW ·····→ 'How is part of the event?'

- Similar to 'Where' and 'Who'

  Ie. "Make a click using an iPhone" → where (where in the app, which button they clicked)
  → how (methodology that lead to click)

D WHAT ·····→ 'What is the event taking place?'

- Atomic events

  Ie. 1. Atomic : A notification ·····→ 'generated', 'sent', 'clicked', 'converted into purchase'

  2. Not Atomic: 'Burning Man' ·····→ Higher-level event (group/aggregation of smaller events)

E WHEN ·····→ 'When is the event taking place?'

- Modeled as timestamp
- Align all app logins to same time zone (UTC), client-side login (higher fidelity vs server side login

FACT DATA MODELING FUNDAMENTALS (II)

*RL: Raw Logs

MUSTs of FACT data

- Quality guarantees: no duplicates, no NULLs in 'what', 'when' event fields
- Size: keep it small, FACT < RL* (what's needed for analysis only)
- Parse out hard-to-understand columns

✓✓ Strings, integers, decimals, enumerates
✓ Array of strings
✗ JSON files

WHEN to model dimensions in FACT data

Network Logs (Traffic) Pipeline NETFLIX

- Goal: understand how Netflix microservices (apps) 'talk' to each other. Analyze traffic data
  ie. What is the impact of app A 'talking' to n apps if being hacked?
- Problem: Huge dataset (2PB/day)
- Solution (2-sided):

IPv4 IP addreses (32-bit)

- IP addresses as app identifier, stored in a smaller database!
- BROADCAST JOIN (Spark) of IDs with Network Logs data

1028x the IPv4 volume!

IPv6 IP addresses (128-bit hexadecimal)

Denormalize FACT data & log it ahead of time. To get rid of the join, Each app adopts a "sidecar proxy" to log of which app they are

- Learning for Data Engineers:

✓ DO: Go upstream & solve the problem at source.
✗ DONT: Optimize a pipeline at first

HOW logging fits into FACT data

- Brings in all critical context (columns)
  Collaborative work with system engineers (handle event generation in the app)
- Necessary logs only (RL ~ expensive)
- Comformance: logging needs to be in some type of contract, shared schema/vision

Setting Pricing Comformance at

- Goal: What describes a price, what pricing is available?
- Problems: two teams (Pricing team Online Service Team), two programming frameworks (Ruby, Scala), potential difference in pricing understanding/definition.
  How to ensure definition and replication on both framework is aligned?
- Solution: shared schema (middle layer)

Thrift Schema
- Way to describe schemas/functions in a specific, language agnostic way
  Ruby/Scala reference both to same schema for price (ie. Scala adds a new pricing column, then schema sends push notification to Ruby to update code)

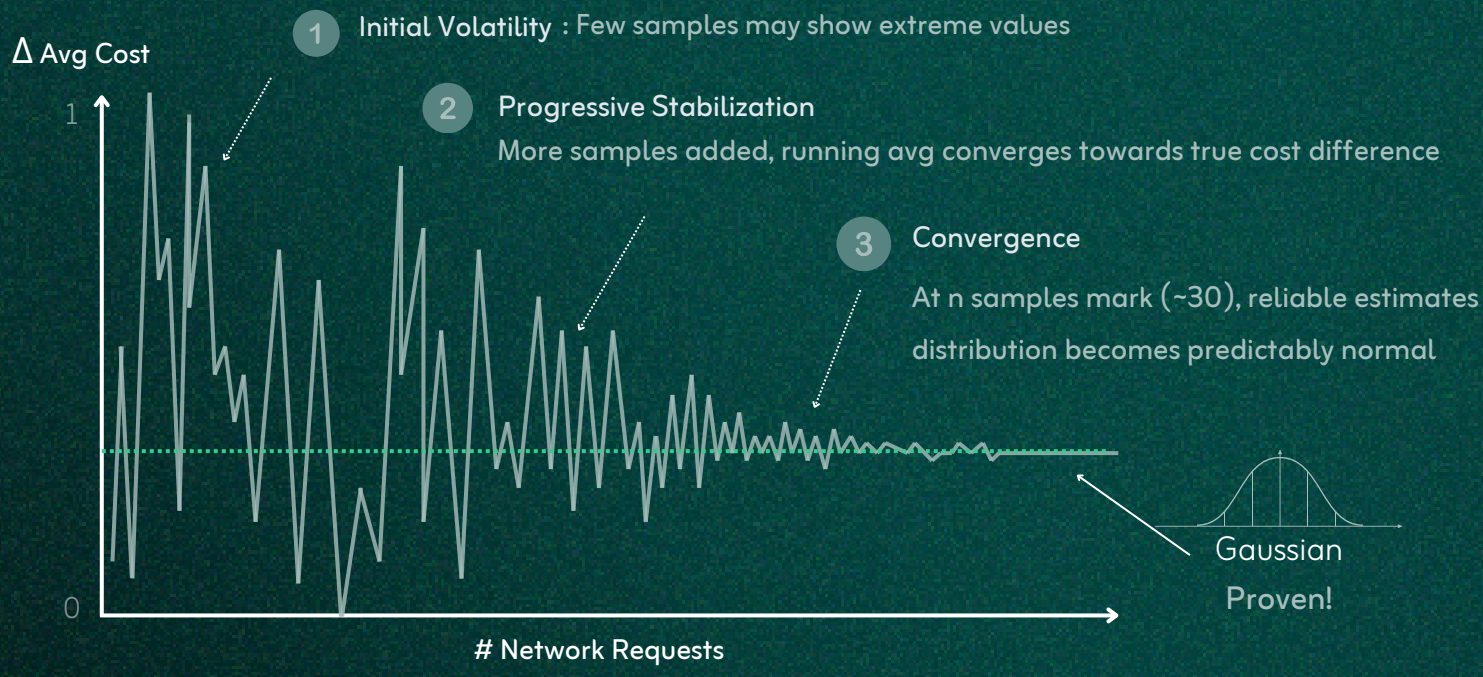
FACT DATA MODELING FUNDAMENTALS (II)

1 SAMPLING ~ small data subset; validated by law of Large Numbers

- Works for problems/ metrics that gauge directionality of things.
- Does not work for security related problems, low probability events, specific row data events.

The 'Infrastructure Cost' metric case at NETFLIX

Q: Does an A/B test cause a higher AWS S3/EC2 infra cost in test group vs control group?



1 BUCKETING

- Bucket on the 'who' (ie. user_id)
- Join the bucketed column w/o shuffling across the entire dataset. Faster, help minimize shuffle in Spark.
  SMB join (sorted merged bucket) ·····→ joins w/o shuffle at all. Both buckets lined up & sorted

? How Long to hold onto FACT data?

Data Size

\+ ←---------- RETENTION

Dimensional	FACT	FACT
As long as legally possible	10 TB	100TB

? How to handle duplicates in data?

Deduplication ~ remove duplicate/redundant copies of data (storage & efficiency improvement)

- Analyze distribution of duplicates over time periods (week/day/hour) & set a timeframe where deduplication matters (latency).

\- ←---------- → + LATENCY

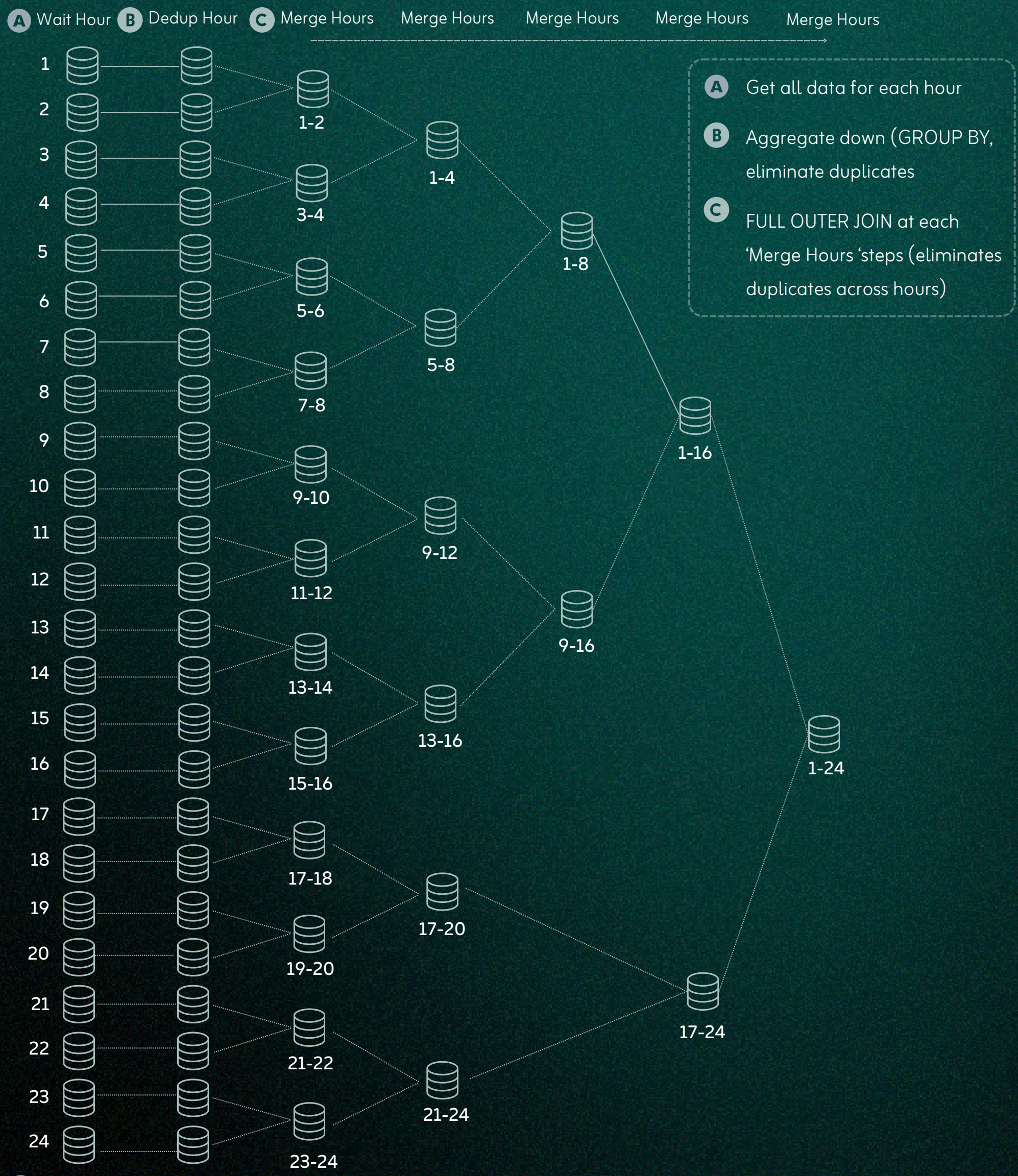
Usage:	STREAMING	MICROBATCH
	Moderate data sets	Larger data sets
	Immediate data consistency	Efficient large-scale deduplication
	Usage of in-memory state stores	Slight delays are acceptable

Albert Campillo

Repost

FACT DATA MODELING FUNDAMENTALS (II)

B MICROBATCH ~ Example: Hourly Deduplication with daily microbatch



A Get all data for each hour

B Aggregate down (GROUP BY, eliminate duplicates

C FULL OUTER JOIN at each 'Merge Hours 'steps (eliminates duplicates across hours)