UNIVERSITY OF WATERLOO

Data Warehouses, Data Lakes, and Lakehouses

(v1.00)

Week 2: September 11, 2025

Jimmy Lin
David R. Cheriton School of Computer Science
University of Waterloo

These slides are available at `https://lintool.github.io/cs451-2025f/`

Key Questions

What are the main differences between operational and analytical infrastructure?

What are data warehouses?
What problems did they evolve to solve?

What are data lakes and lakehouses?
What problems did they evolve to solve?

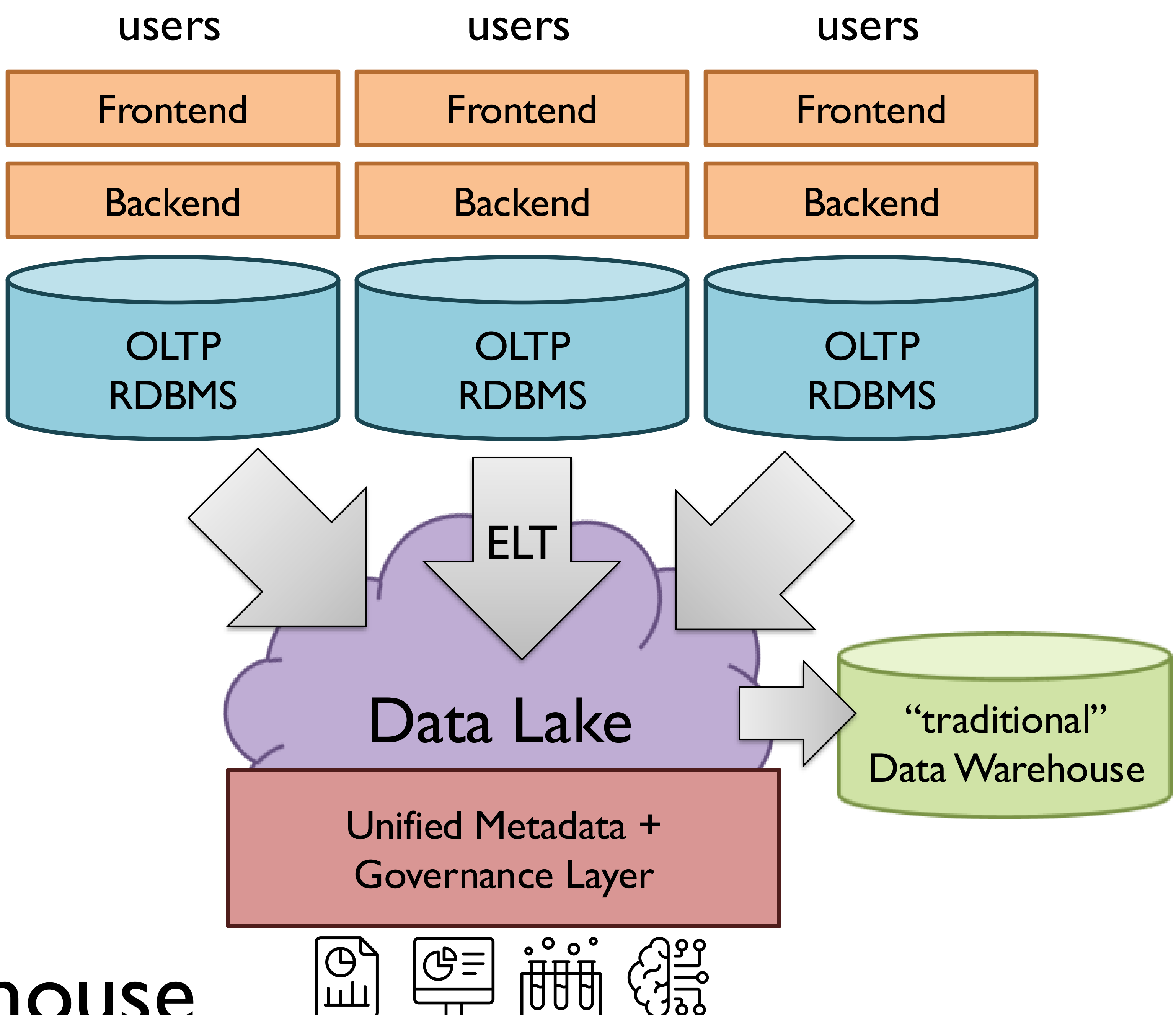
What are the components of modern data platforms?

How do operational and analytical data models differ?

What goes on in ETL/ELT?

How do different physical representations of data affect storage, compute, and other tradeoffs within data platforms?

Recap: What are we doing and why?


users
users
users
Frontend
Frontend
Frontend
Backend
Backend
Backend
OLTP RDBMS
OLTP RDBMS
OLTP RDBMS
ELT
Data Lake
"traditional" Data Warehouse
Unified Metadata + Governance Layer

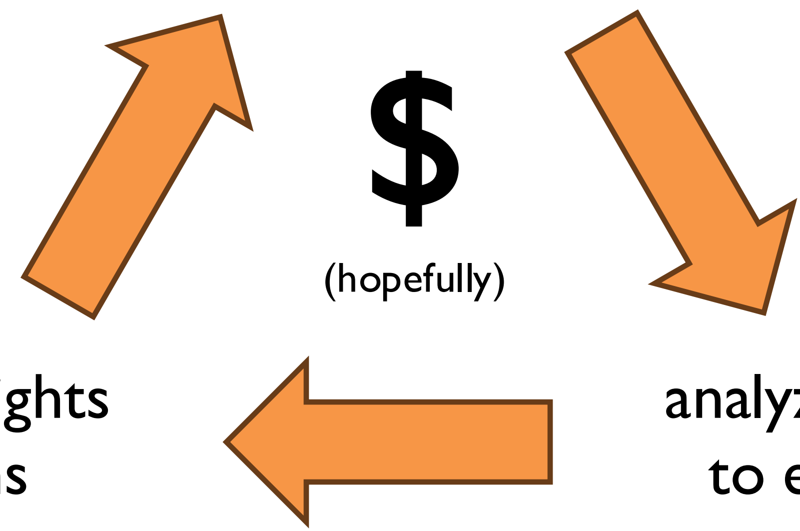
Lakehouse

Context…

The Data Flywheel

(a virtuous cycle)

Build a useful product



transform insights into actions

analyze user behavior to extract insights

Google. Facebook. Twitter. Amazon. Uber.

Context…

What's this course about?

The *infrastructure* that supports the data flywheel.

data platforms + data engineering

Context…

What problems do data platforms solve?

Ingesting, storing, manipulating, maintaining, serving…
the data that supports the data flywheel.

Context…

Transform Insights into Actions

What does that really mean?

Report generation Dashboards	Business Intelligence
Ad hoc analyses ML models	Data Science

known unknowns and unknown unknowns?

This Week

Previous: Evolution of Data Platforms

Data Warehouses, Data Lakes, and Lakehouses

Now: Three Deep Dives

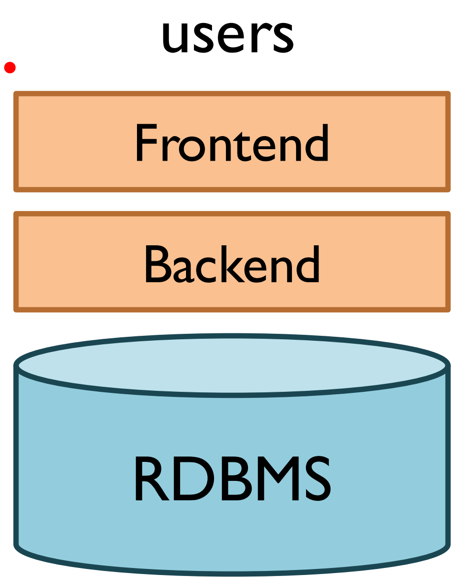
Data Modeling, ELT, Physical Representations

Deep Dive: Data Models

What's a data model?

A data model is an abstract model that organizes elements of data and standardizes how they relate to one another and to the properties of real-world entities. *(from Wikipedia)*

Let's start here…



Okay, but why *relational*?

RDBMS = Relational Database Management System

Imposes a relational view of data: tables, rows, columns

Provides a set of relational operators to manipulate data: SQL

Why is this a good idea?

Offload physical data design

Standardize query processing

Ensure data integrity, manage concurrency

Handle backup and recovery

Why Relational?

(BTW, do the readings)

Leon Bambrick
@secretGeek

There are 2 hard problems in computer science: cache invalidation, naming things, and off-by-1 errors.

9:20 AM · Jan 1, 2010

23 1.4K 1.8K 97

https://x.com/secretGeek/status/7269997868

How do we name things?

UNIVERSITY OF WATERLOO

What's the name of our school?

yupp

Seriously, what are we *actually* going to call it?

Qwen
@Alibaba_Qwen

Bye Qwen3-235B-A22B, hello Qwen3-235B-A22B-2507!

After talking with the community and thinking it through, we decided to stop using hybrid thinking mode. Instead, we'll train Instruct and Thinking models separately so we can get the best quality possible. Today, we're releasing Qwen3-235B-A22B-Instruct-2507 and its FP8 version for everyone.

This model performs better than our last release, and we hope you'll like it thanks to its strong overall abilities.

https://x.com/Alibaba_Qwen/status/1947344511988076547

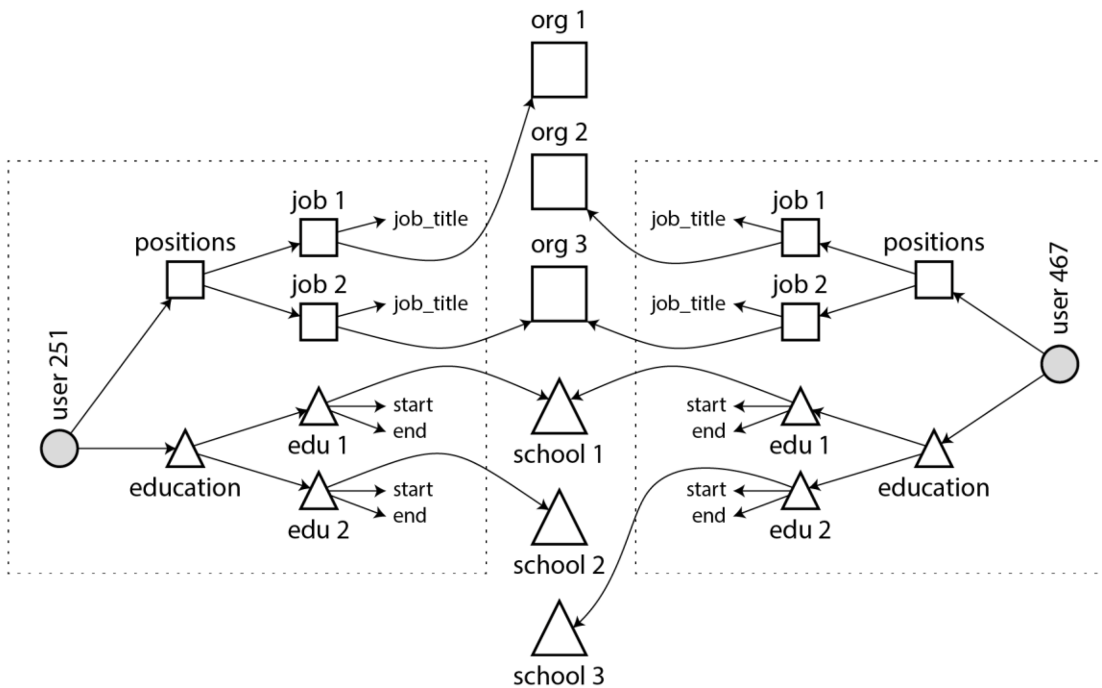
Why Relational?

What's the competition?

JSON, YAML, XML, etc. ("documents")

Now answer the questions on the previous slide

Why Relational?



Source: *Designing Data-Intensive Applications, 2nd Edition*, Chapter 3

At some point you realize that you're basically implementing an RDBMS… *poorly*

Why relational? ✓

(Starting point… but recall discussion about EDWs and EDLs and the importance of flexibility…)

What's the schema?

(Operational and analytical data models are different.)

Why?

Remember this?

RDBMS Workloads

OLTP (online transaction processing)

Typical applications: e-commerce, banking, airline reservations
Customer-facing: real-time, low latency, highly-concurrent
Tasks: relatively small set of transactional queries; CRUD
Data access pattern: random reads, updates, writes (small amounts of data)

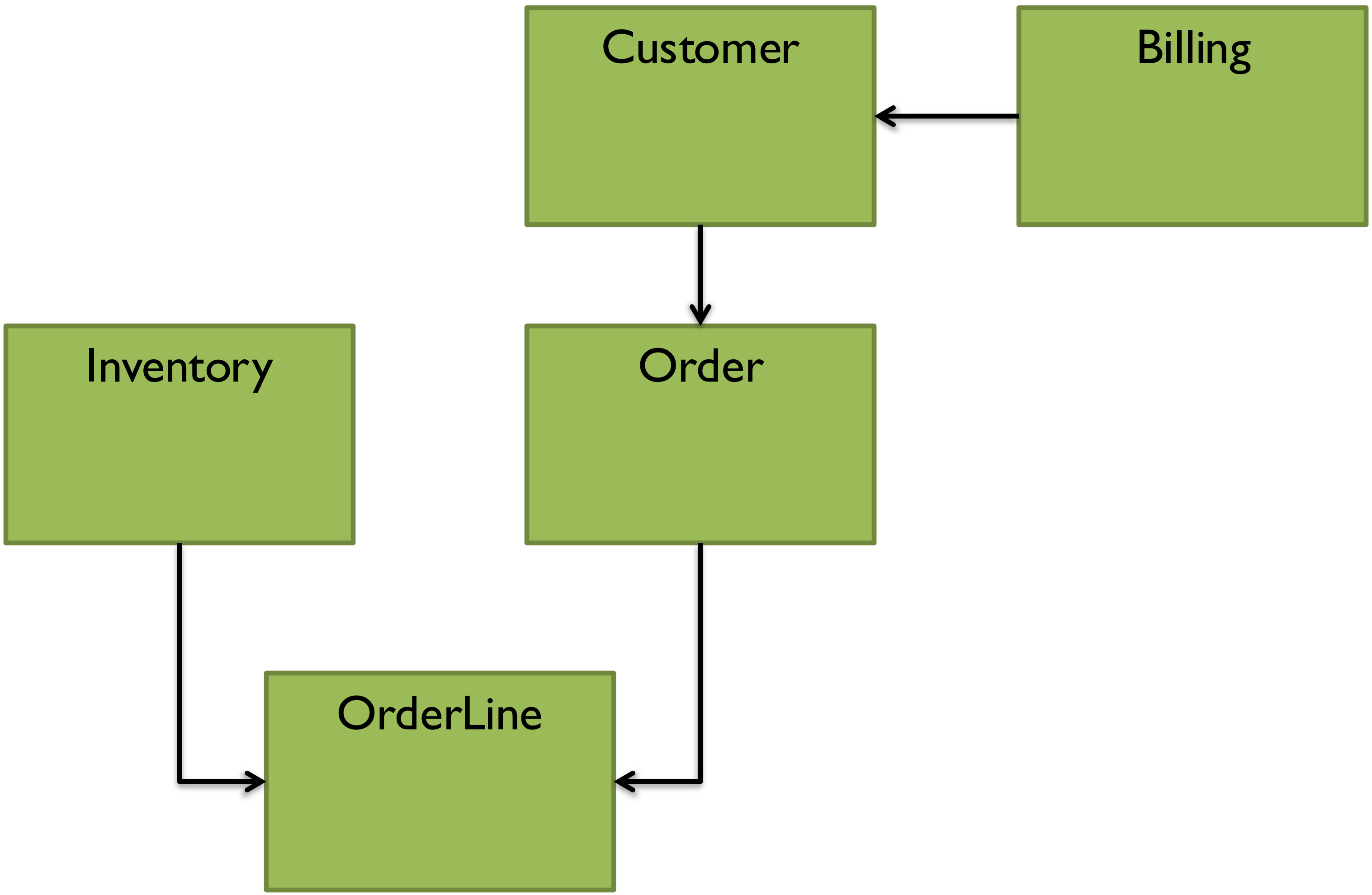
(optimize for the common case)

OLAP (online analytical processing)

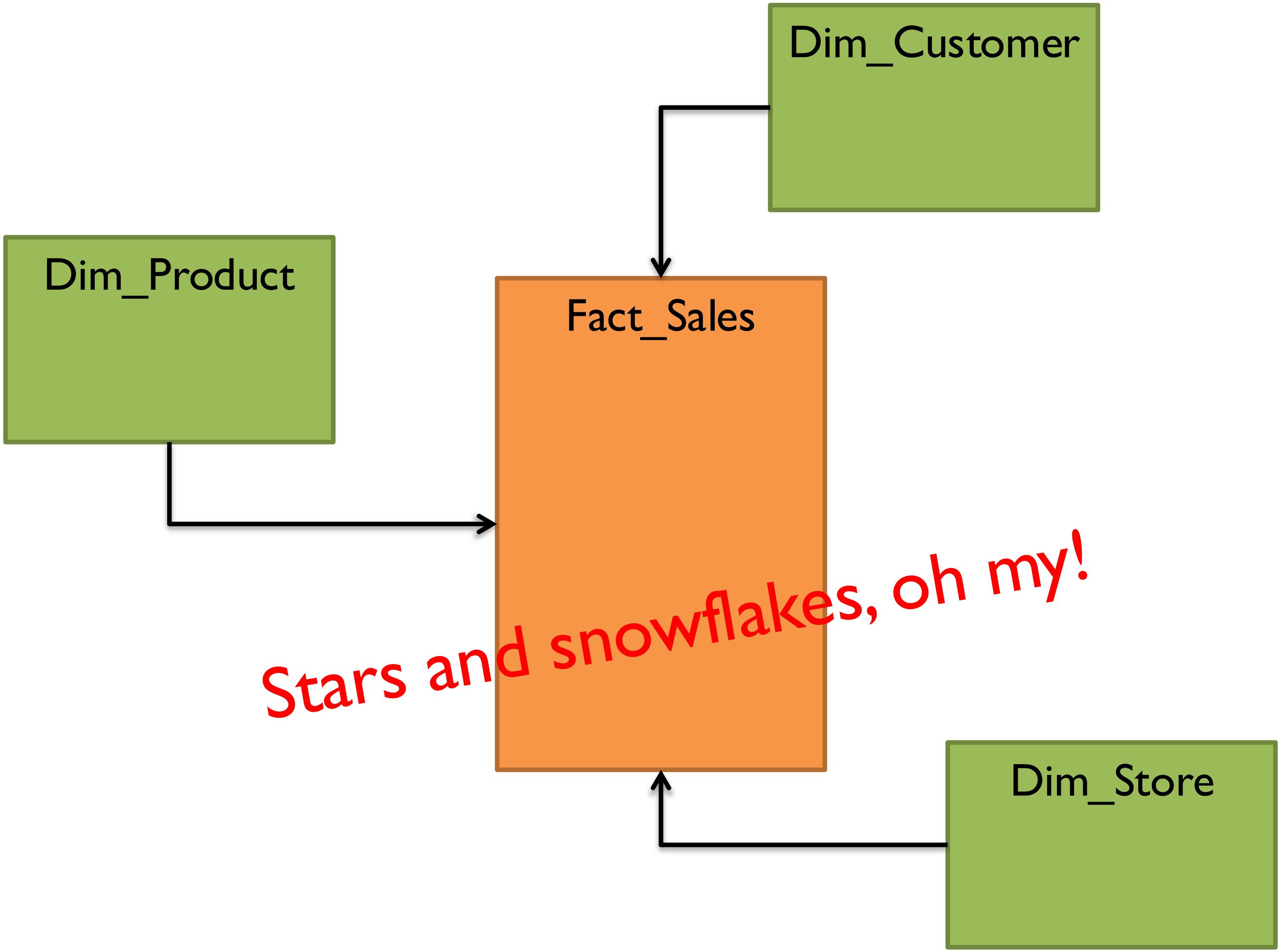
Typical applications: business intelligence, data mining
Back-end processing: batch workloads, less concurrency
Tasks: complex analytical queries, often ad hoc
Data access pattern: table scans, large amounts of data per query

tl;dr – EDWs are organized in a way that makes answering the most common questions easy

A Simple OLTP Schema



A Simple OLAP Schema



dim_product table

product_sk	sku	description	brand	category
30	OK4012	Bananas	Freshmax	Fresh fruit
31	KA9511	Fish food	Aquatech	Pet supplies
32	AB1234	Croissant	Dealicious	Bakery

dim_store table

store_sk	state	city
1	WA	Seattle
2	CA	San Francisco
3	CA	Palo Alto

fact_sales table

date_key	product_sk	store_sk	promotion_sk	customer_sk	quantity	net_price	discount_price
240102	31	3	NULL	NULL	1	2.49	2.49
240102	69	5	19	NULL	3	14.99	9.99
240102	74	3	23	191	1	4.49	3.89
240102	33	8	NULL	235	4	0.99	0.99

dim_date table

date_key	year	month	day	weekday	is_holiday
240101	2024	jan	1	mon	yes
240102	2024	jan	2	tue	no
240103	2024	jan	3	wed	no

dim_customer table

customer_sk	name	date_of_birth
190	Aaliyah	1979-03-29
191	Bryce	1961-09-02
192	Caleb	1991-12-13

dim_promotion table

promotion_sk	name	ad_type	coupon_type
18	New Year sale	Poster	NULL
19	Aquarium deal	Newsletter	Discount code
20	Coffee & cake bundle	In-store sign	NULL

Source: *Designing Data-Intensive Applications, 2nd Edition*, Chapter 3

EDWs are generally organized as stars (or snowflakes)

Why? This data model makes answering the most common questions easy

What questions?

Remember this?

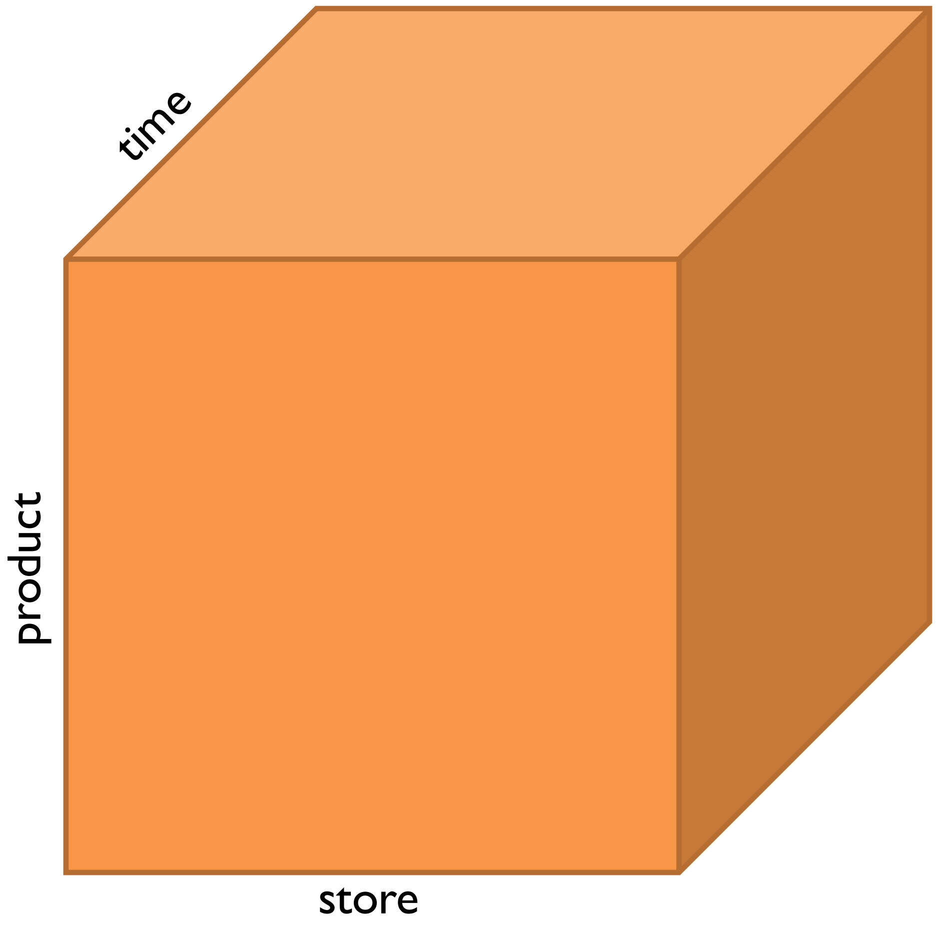
Transform Insights into Actions

What does that really mean?

$\left.\begin{array}{c}\text{Report generation}\\ \text{Dashboards}\end{array}\right\}$ Business Intelligence

$\left.\begin{array}{c}\textit{Ad hoc}\text{ analyses}\\ \text{ML models}\end{array}\right\}$ Data Science

OLAP Cubes



Common operations

- slice and dice
- roll up/drill down
- pivot

Let’s work through concrete examples of DM differences…

dim_product table

product_sk	sku	description	brand	category
30	OK4012	Bananas	Freshmax	Fresh fruit
31	KA9511	Fish food	Aquatech	Pet supplies
32	AB1234	Croissant	Dealicious	Bakery

dim_store table

store_sk	state	city
1	WA	Seattle
2	CA	San Francisco
3	CA	Palo Alto

fact_sales table

date_key	product_sk	store_sk	promotion_sk	customer_sk	quantity	net_price	discount_price
240102	31	3	NULL	NULL	1	2.49	2.49
240102	69	5	19	NULL	3	14.99	9.99
240102	74	3	23	191	1	4.49	3.89
240102	33	8	NULL	235	4	0.99	0.99

dim_date table

date_key	year	month	day	weekday	is_holiday
240101	2024	jan	1	mon	yes
240102	2024	jan	2	tue	no
240103	2024	jan	3	wed	no

dim_customer table

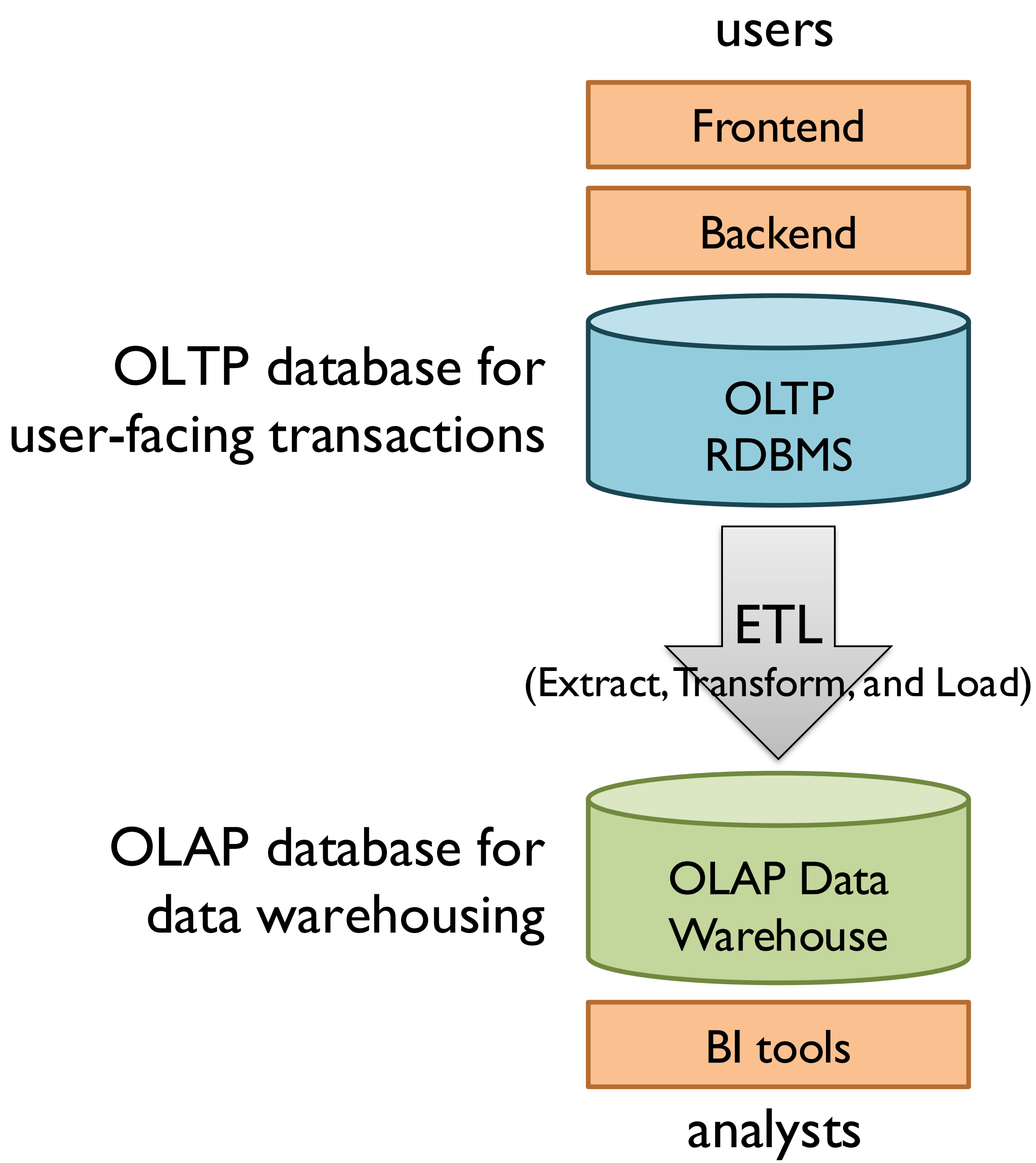
customer_sk	name	date_of_birth
190	Aaliyah	1979-03-29
191	Bryce	1961-09-02
192	Caleb	1991-12-13

dim_promotion table

promotion_sk	name	ad_type	coupon_type
18	New Year sale	Poster	NULL
19	Aquarium deal	Newsletter	Discount code
20	Coffee & cake bundle	In-store sign	NULL

Source: *Designing Data-Intensive Applications, 2nd Edition*, Chapter 3

Deep Dive: ELT


users
Frontend
Backend
OLTP database for
user-facing transactions
OLTP
RDBMS
ETL
(Extract, Transform, and Load)
OLAP database for
data warehousing
OLAP Data
Warehouse
BI tools
analysts

Extract-Transform-Load

Extract

Export from OLTP database

Transform

Data cleaning and integrity checking

Schema transformations and field conversions

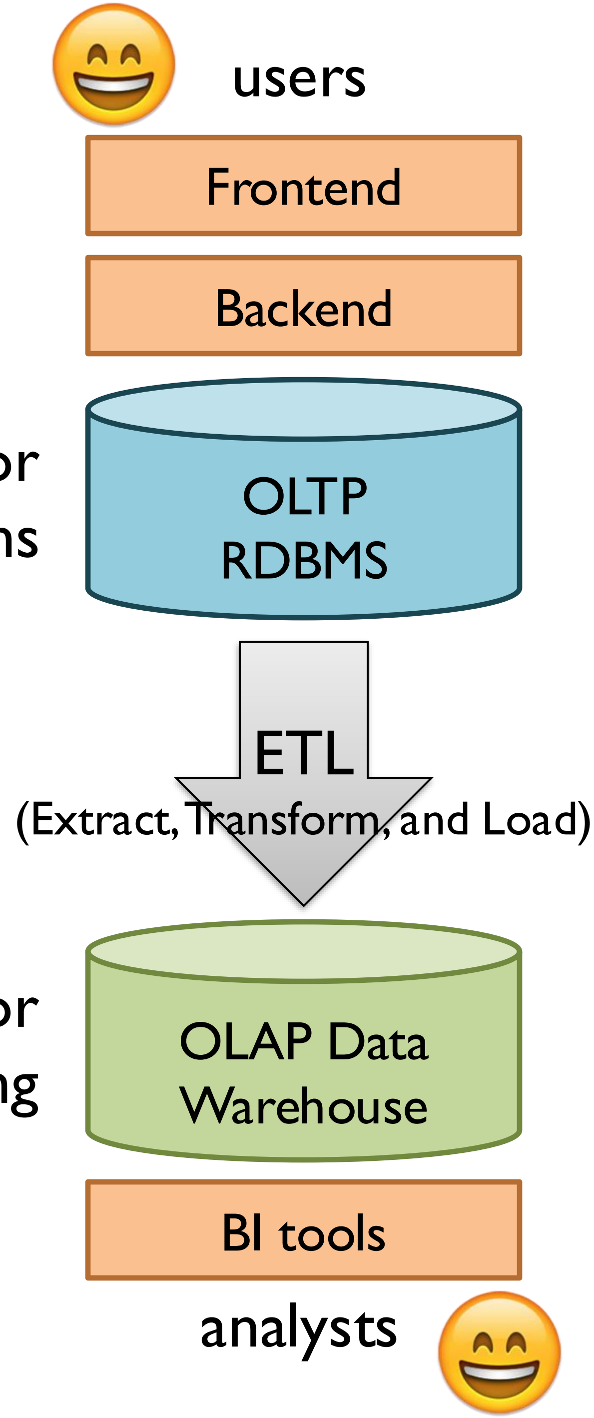
…

Load

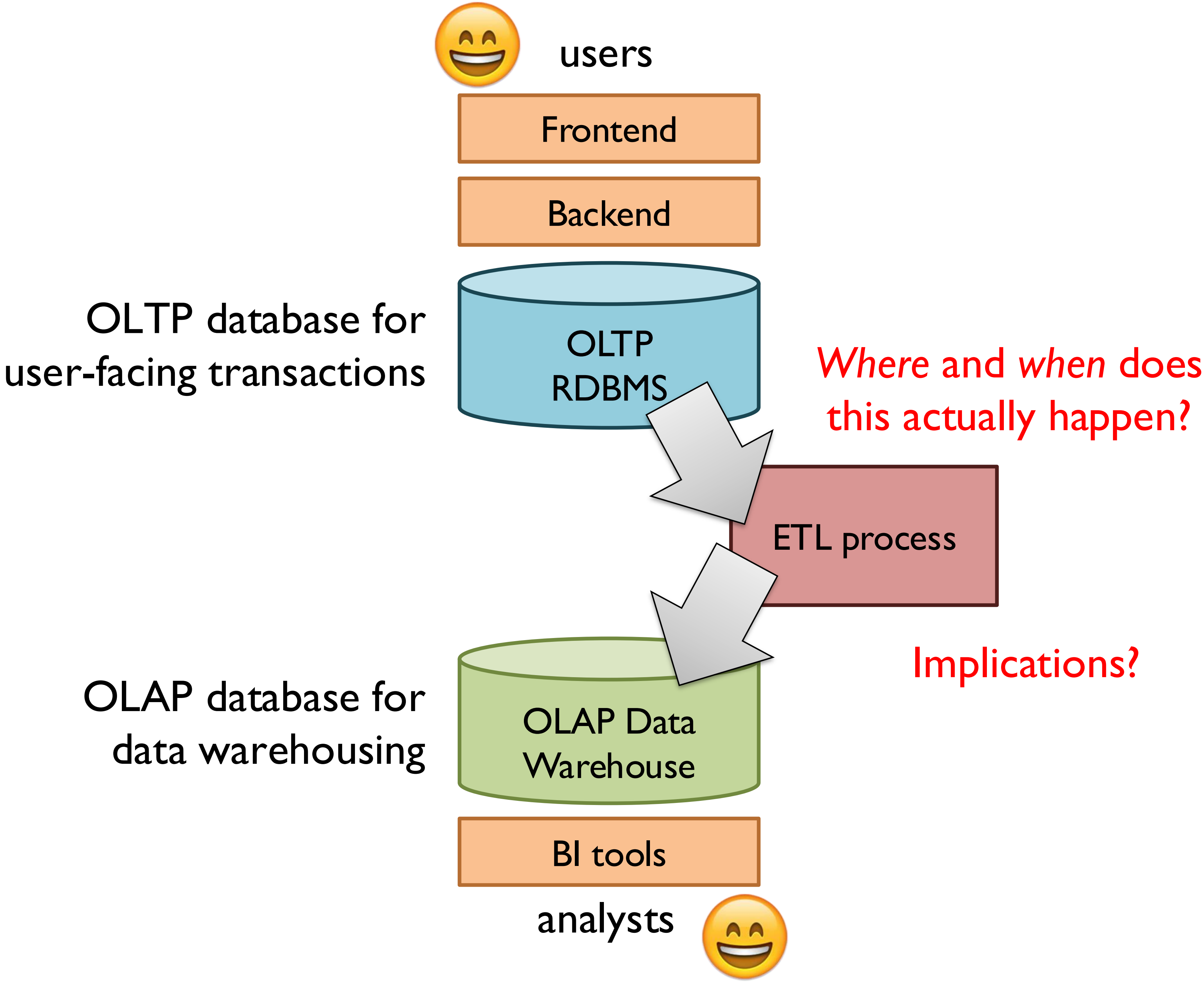
Ingest into OLAP database

OLTP database for
user-facing transactions

Where and *when* does
this actually happen?



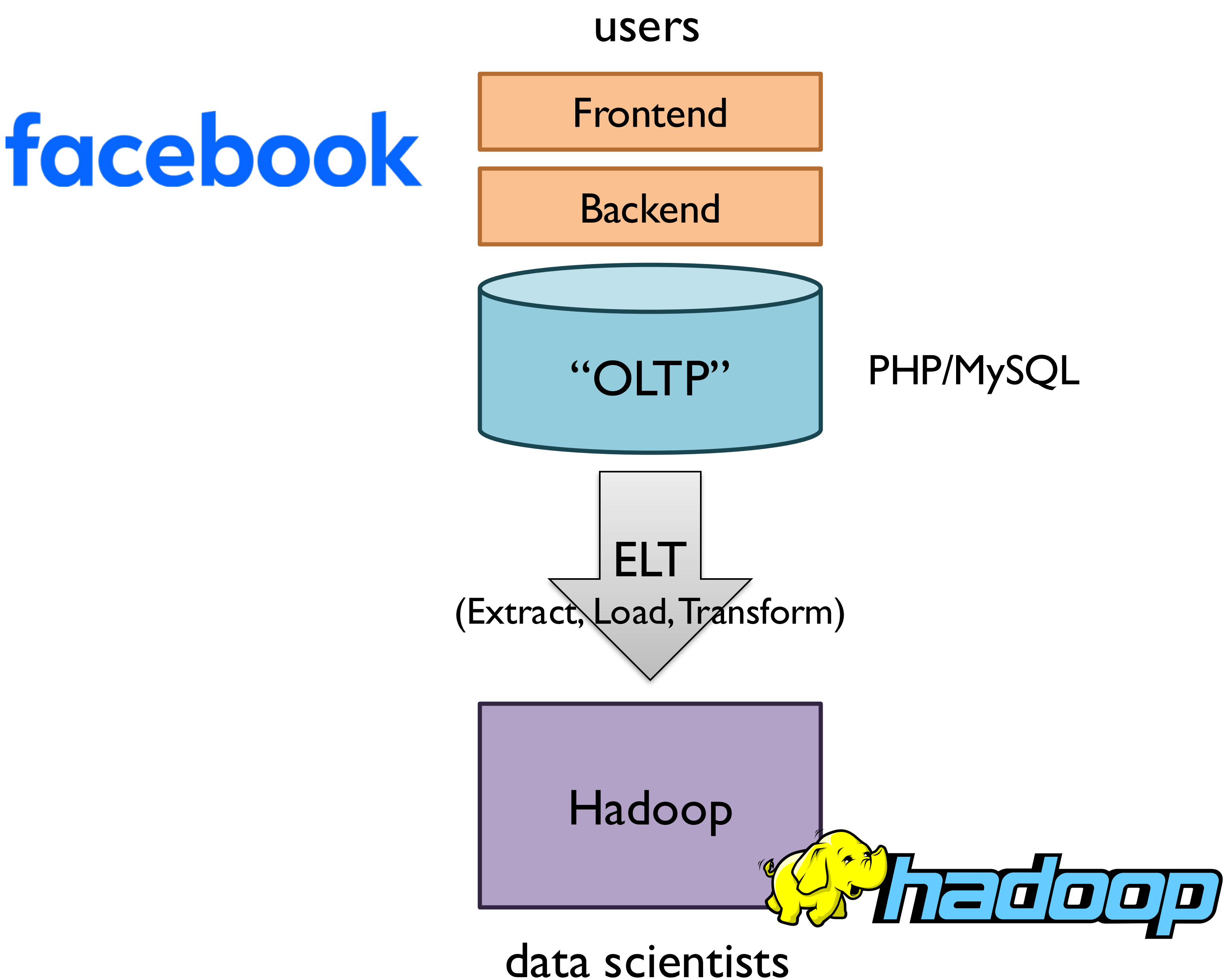
OLAP database for
data warehousing

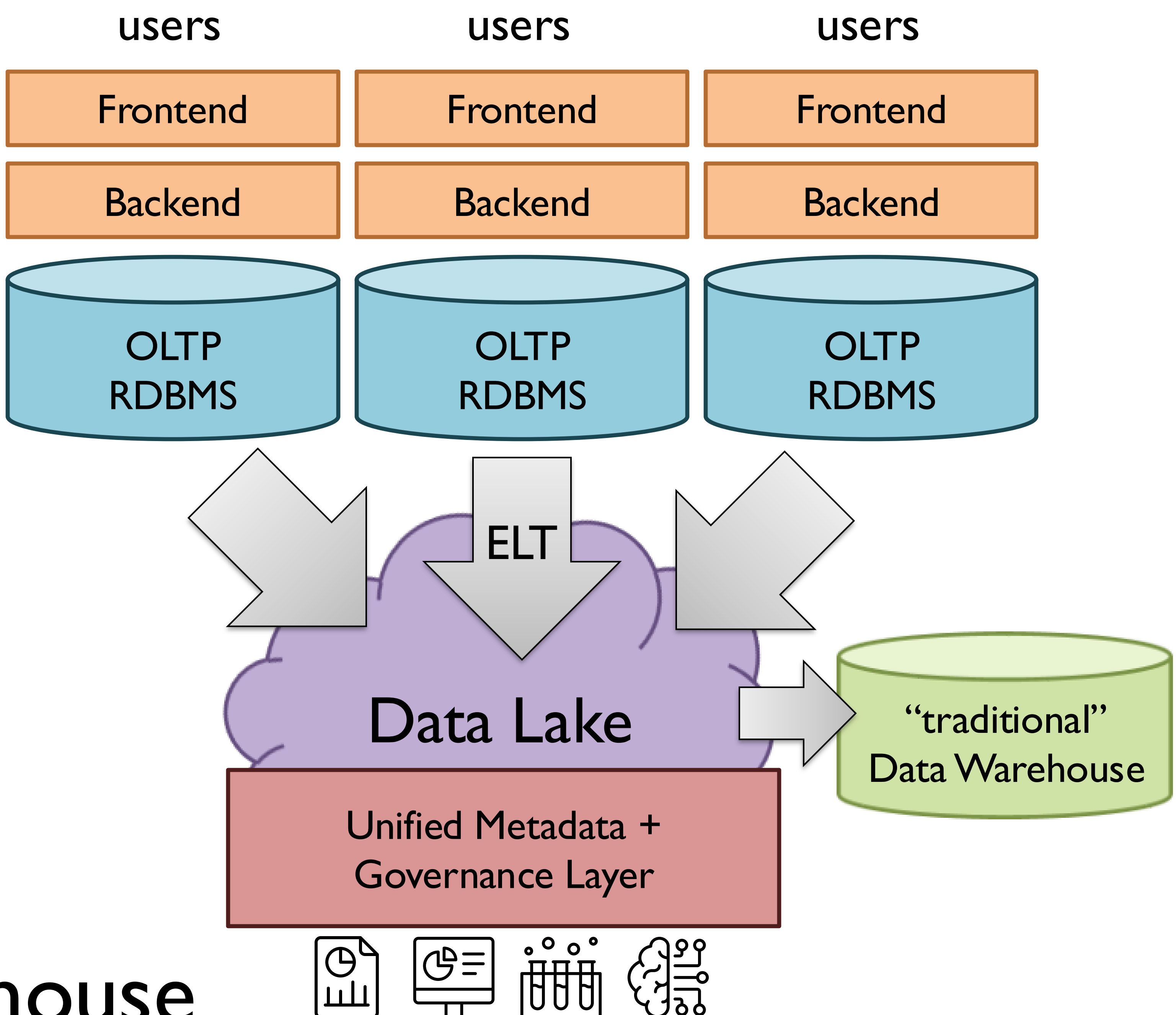

users
Frontend
Backend
OLTP database for
user-facing transactions
OLTP
RDBMS
Where and *when* does
this actually happen?
ETL process
Implications?
OLAP database for
data warehousing
OLAP Data
Warehouse
BI tools
analysts

facebook®

Jeff Hammerbacher, Information Platforms and the Rise of the Data Scientist.
In, *Beautiful Data*, O'Reilly, 2009.

"On the first day of logging the Facebook clickstream, more than 400 gigabytes of data was collected. The load, index, and aggregation processes for this data set really taxed the Oracle data warehouse. Even after significant tuning, we were unable to aggregate a day of clickstream data in less than 24 hours."


facebook
users
Frontend
Backend
"OLTP"
PHP/MySQL
ELT
(Extract, Load, Transform)
Hadoop
hadoop
data scientists


users
users
users
Frontend
Frontend
Frontend
Backend
Backend
Backend
OLTP RDBMS
OLTP RDBMS
OLTP RDBMS
ELT
Data Lake
"traditional" Data Warehouse
Unified Metadata + Governance Layer

Lakehouse

Extract-Load-Transform

Extract

Export from operational databases

Load

Ingest into the data lake

Transform

Everything else…

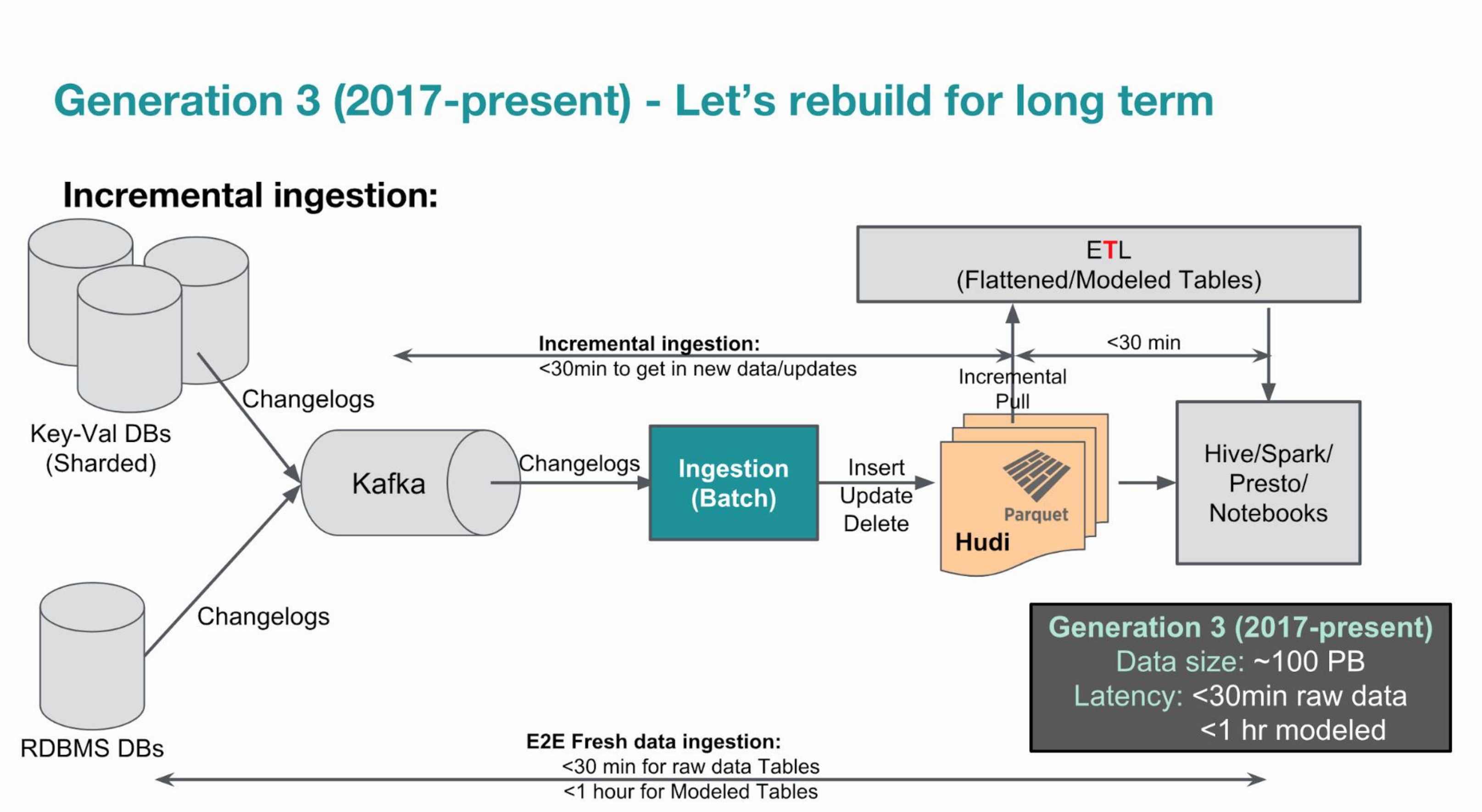
Extract + Load

"Load" is as simple as copying into the data lake

Many possible architectural patterns for extraction

Source considerations: web servers, mobile clients, APIs, etc.

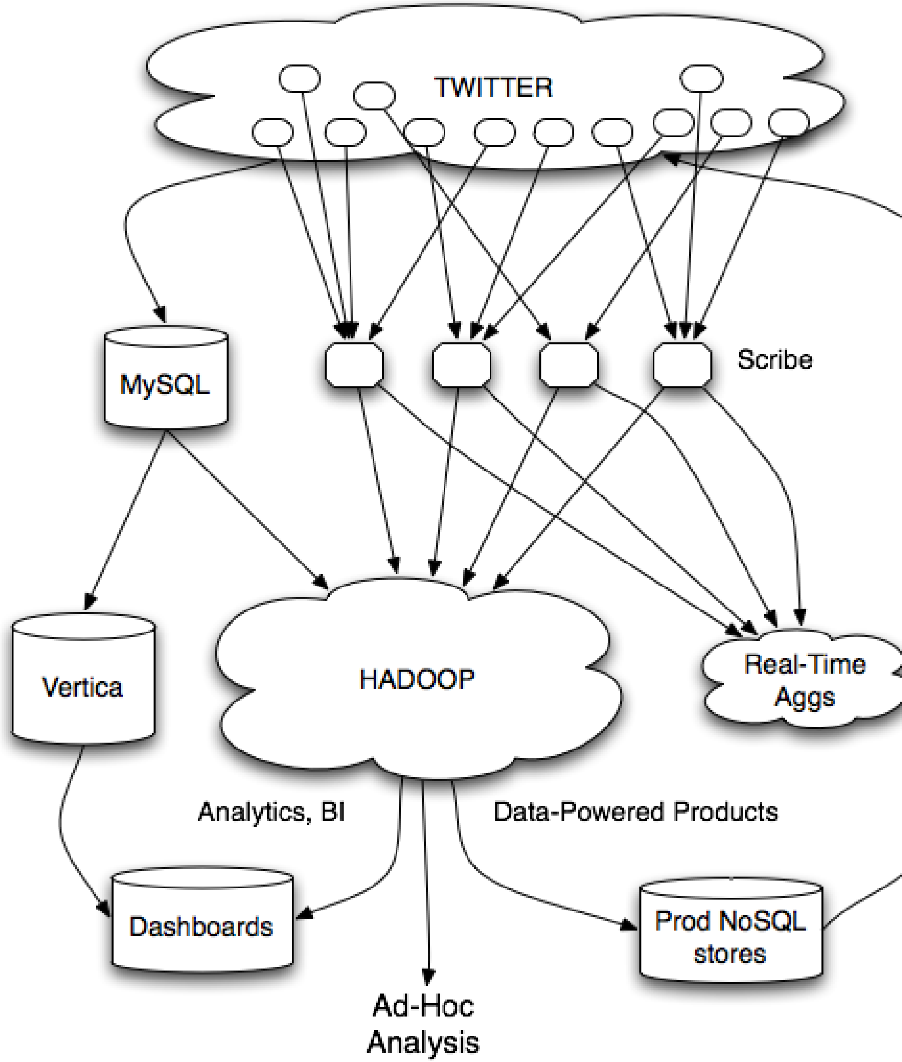
Common pattern: publish events to Kafka w/ periodic rollups



Read blog for more details!

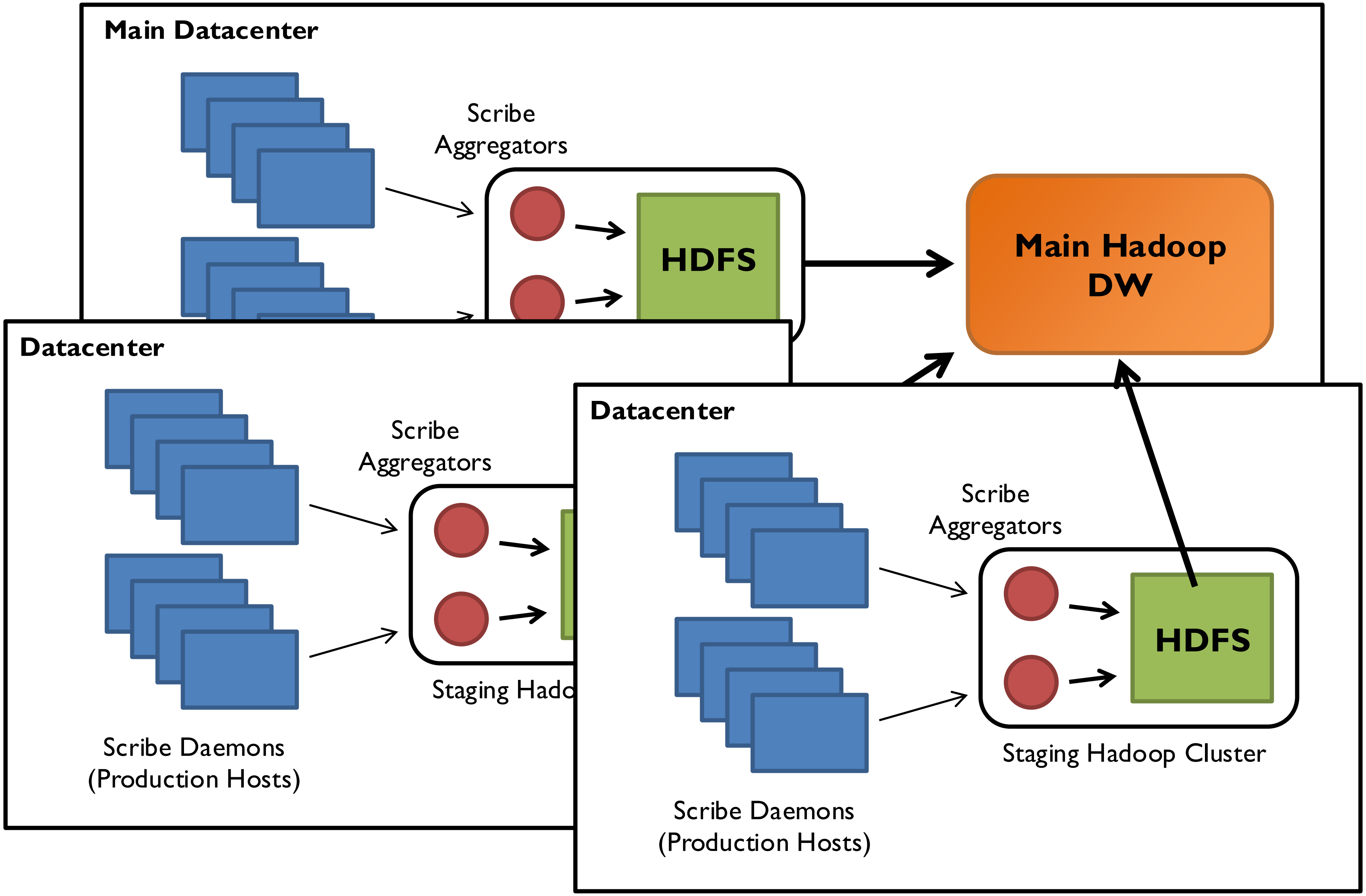
Uber’s data platform (circa 2018)

`https://www.uber.com/en-CA/blog/uber-big-data-platform/`



Twitter's data platform (circa 2012)

Importing Log Data



(Examples of) Transformation

= "everything else"

Field conversions

(e.g., munging timestamps)

Schema transformations

(e.g., pre-joins to bridge OLTP/OLAP data models)

Data profiling

(e.g., distribution of keys/values)

Data cleaning

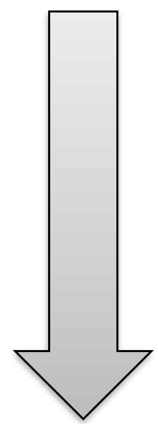
Missing keys, dangling pointers, null values

Outliers, inconsistent values

…

Medallion system

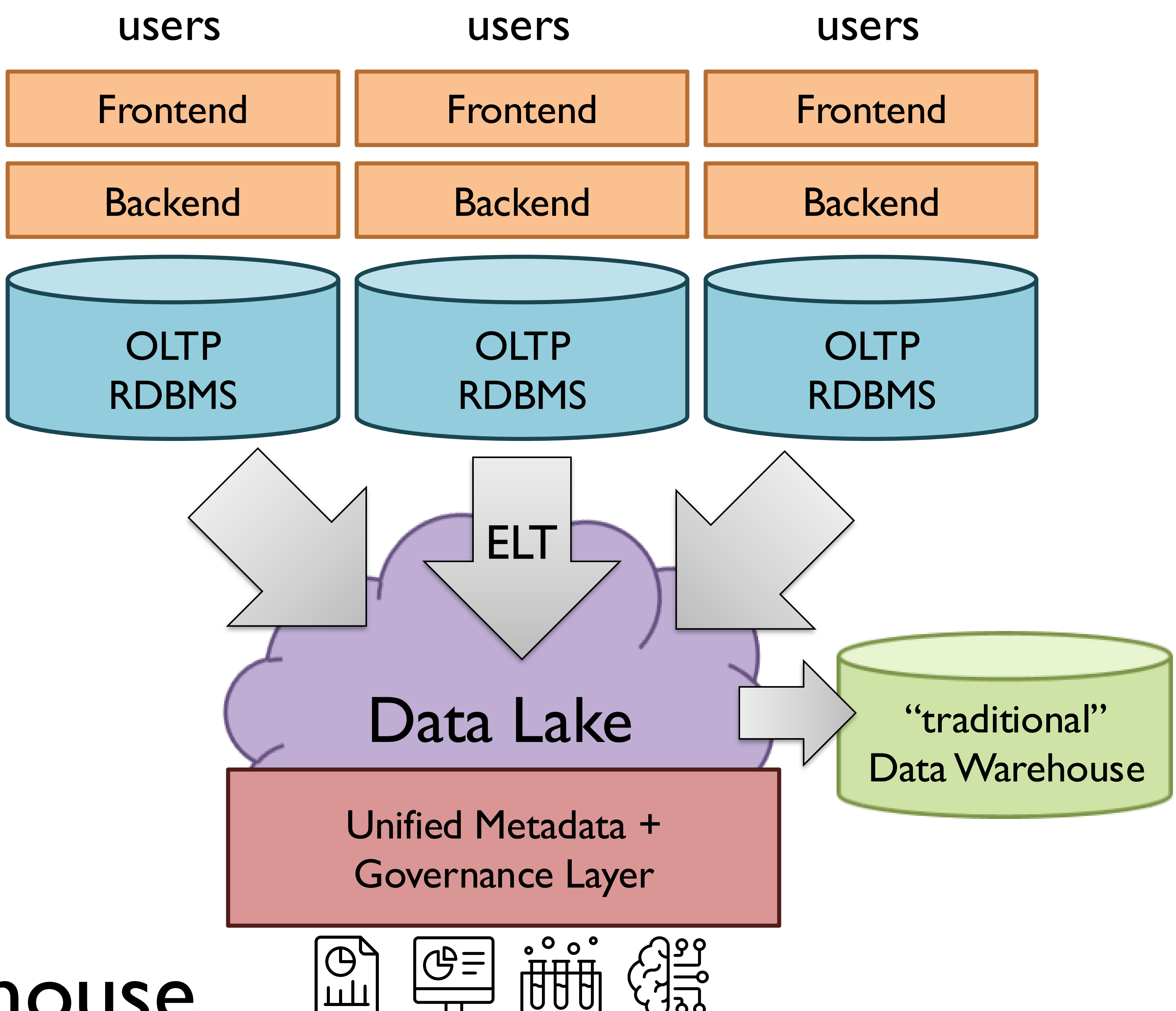
"Raw" ingested data



- Bronze
- Silver
- Gold

"Refined" transformed data

Deep Dive: Physical Representations

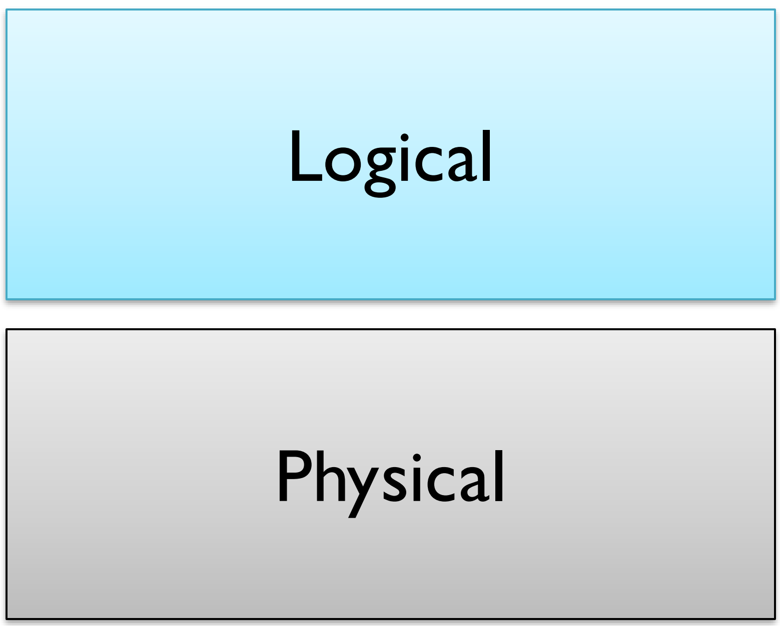

users
users
users
Frontend
Frontend
Frontend
Backend
Backend
Backend
OLTP RDBMS
OLTP RDBMS
OLTP RDBMS
ELT
Data Lake
"traditional" Data Warehouse
Unified Metadata + Governance Layer

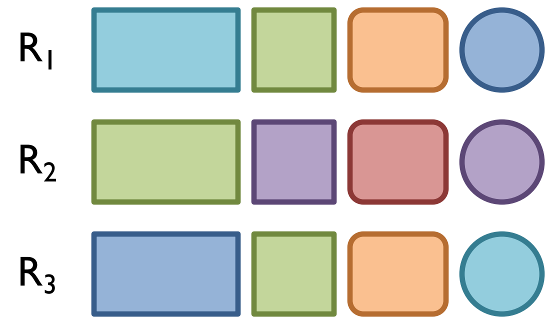
Lakehouse

Flexibility

In a data lake, you can store whatever you want, and "load" is just copying in data…

But what do you *actually* store?





How records are *actually* represented…

CSV

okay for “quick + dirty”, avoid for prod

Many sources of confusion

What’s the delimiter?

What about escape characters?

Bad performance

Verbose and slow

Compression limits parallelism

json

commonly used, but lots of gotchas

Lack of schema

What's the schema?
Are you sure?
Really sure? (error vs. evolution)
What if it isn't an integer?
How do you represent a null?

Bad performance

Verbose (plain-text encoding, repeated field names)
Slow (complex parsing)
bson? Worst of both worlds!

protobuf / avro

best practice

Schemas are good!

Validation for free

Possible forward/backward compatibility support for evolution

Binary encodings are good!

Efficient, unambiguous, compact

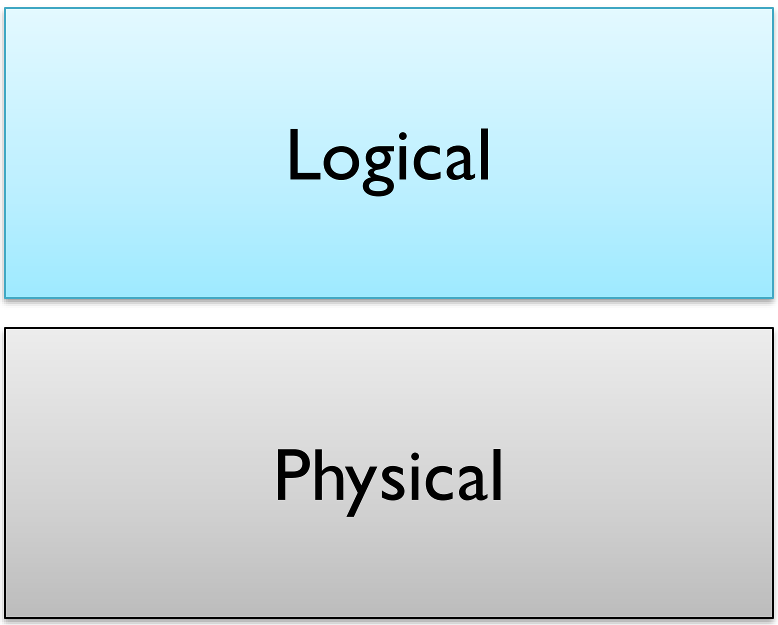
Amenable to different encodings + compression

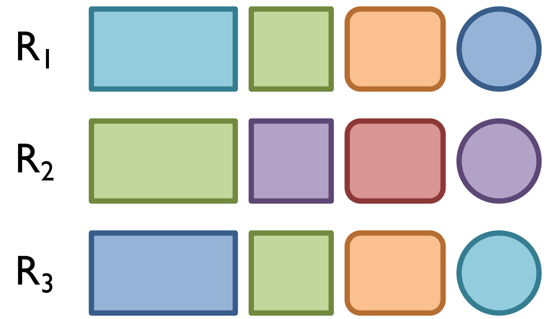
Fast serialization + deserialization

Decouple logical from physical!

Best Practice

Create well-defined schemas…

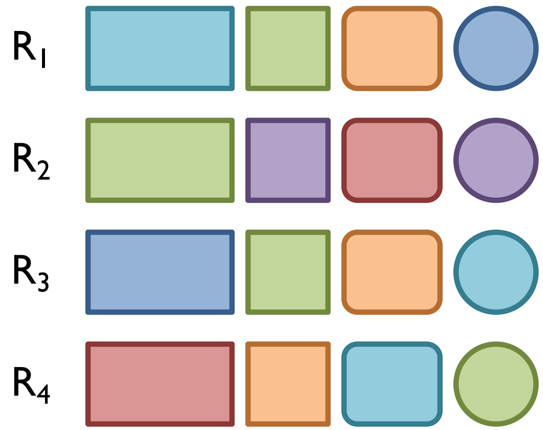




How records are *actually* represented…

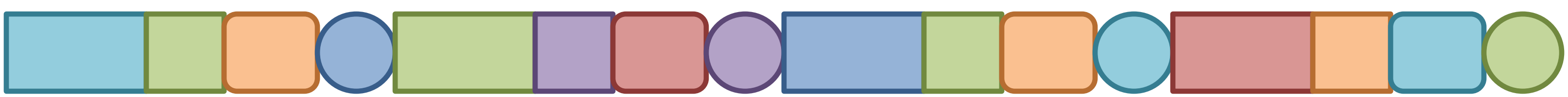
Encoded in protobuf or avro

Row vs. Column Representations



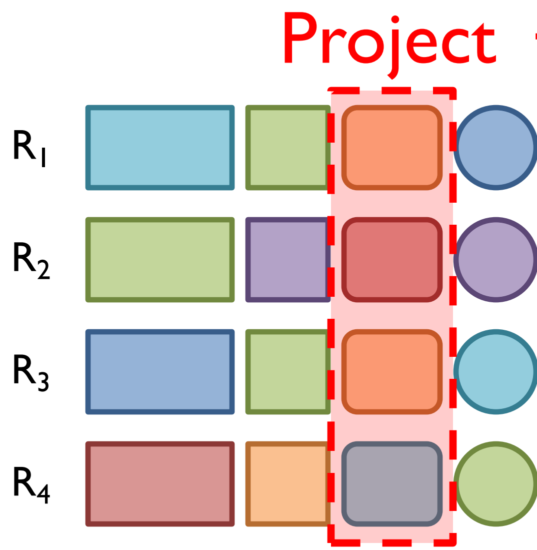
Why does it matter?

Row representation

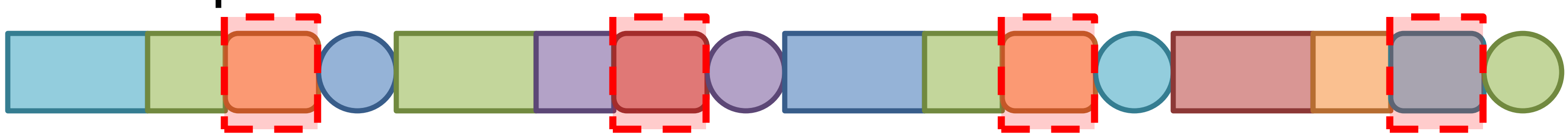


Column representation

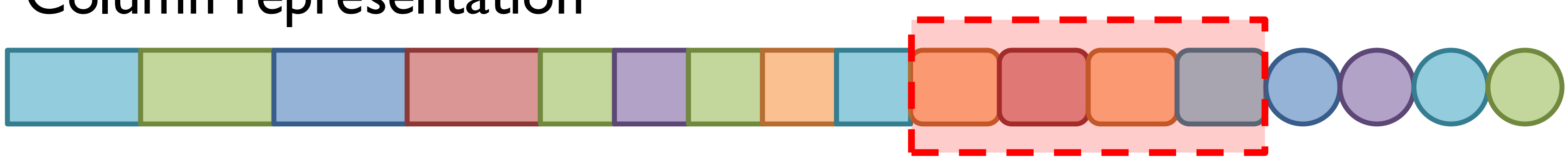
Row vs. Column Representations



Row representation



Column representation



Row vs. Column Representations

Row representations Suitable for OLTP

Easier to modify a record: in-place updates

Might read unnecessary data when processing

Column representations Suitable for OLAP

Only read necessary data when processing

Tuple writes require multiple operations

Tuple updates are complex

Common ELT pattern:

load row representation, *transform* into column representation

Why Column Representations?

More amenable to analytics

Process only data necessary for the query: big win in DWs

Difficulty in updates: manageable drawback

More compact representation

Further enhances processing performance

Encoding Columns

Column representation brings semantically similar values together

Big wins if cardinality of values is small

Allows application of encoding tricks…

Dictionary encoding

Run-length encoding

Bit-packing

…

Compressing Columns

Compression can be applied on top of encoding

Speed more important than compression ratio

Most commonly used codec today is Snappy

Fast, lightweight

Based on dictionaries + backpointers

What's Parquet?

Semi-structured data meets column representation

Answers this question: What does a "column" mean in json?

Pulls together everything we discussed above

Given a schema, records are divided into row groups

Within each row group, data is represented in columns

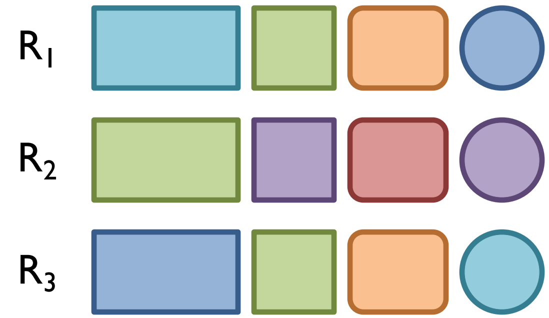
(Establishes precise semantics on what columns mean in json)

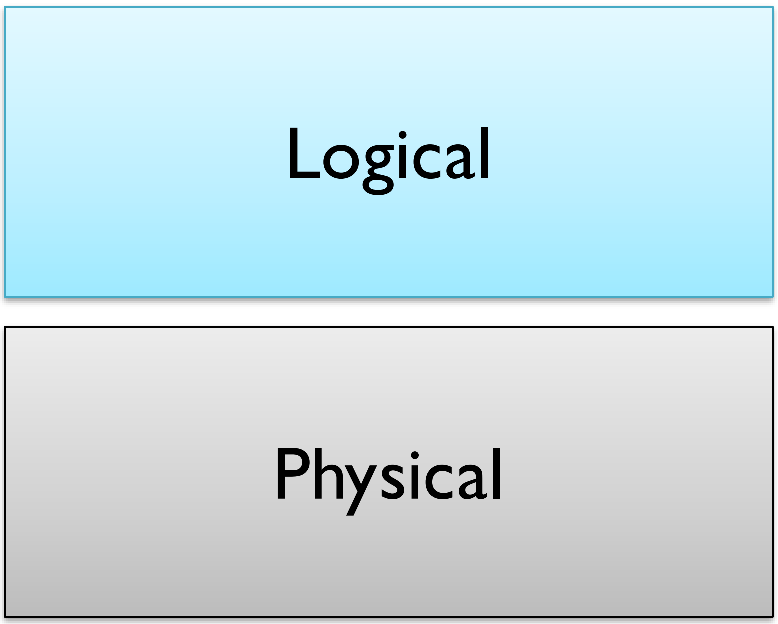
Each column is encoded + compressed separately

Parquet

Best Practice

Create well-defined schemas…

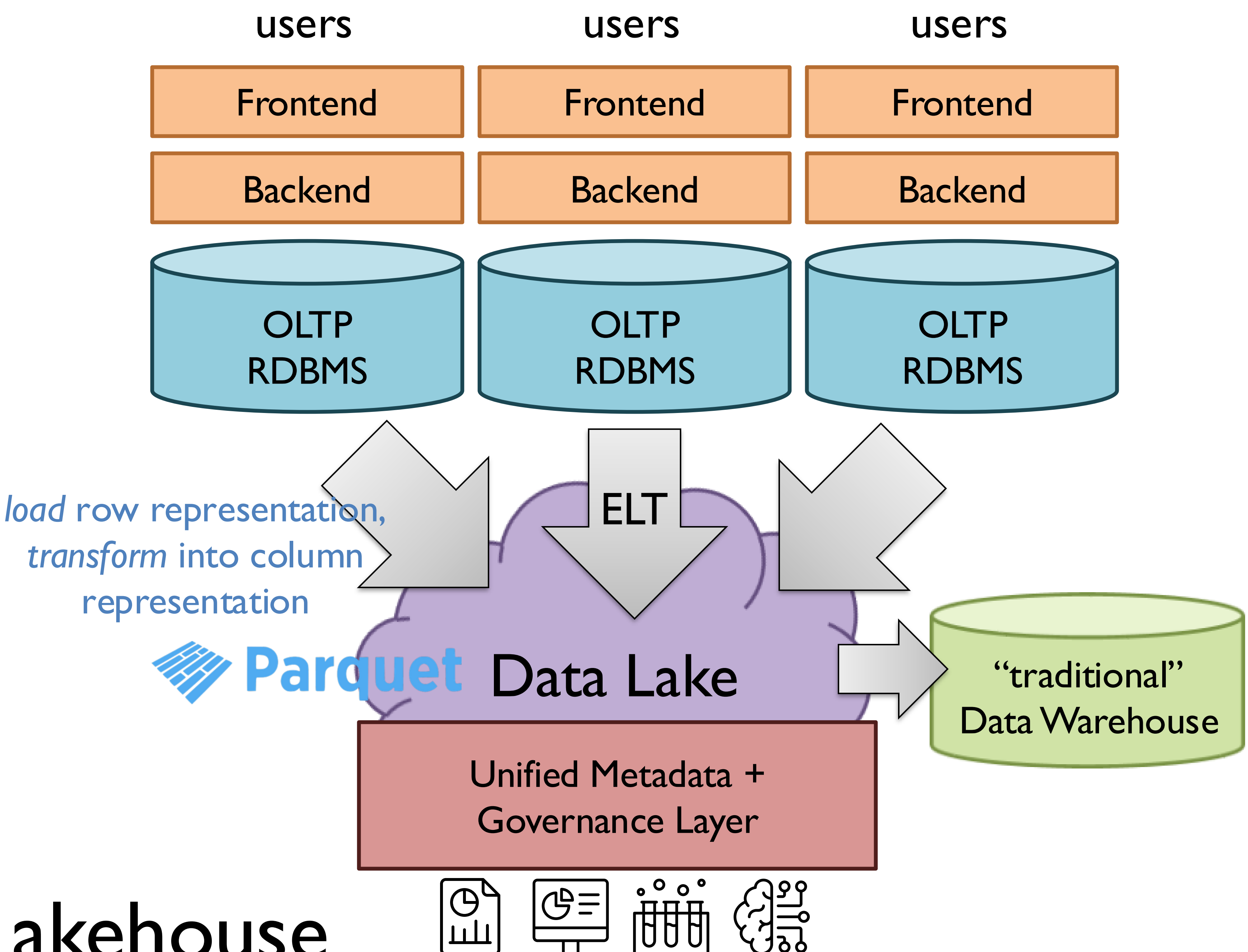




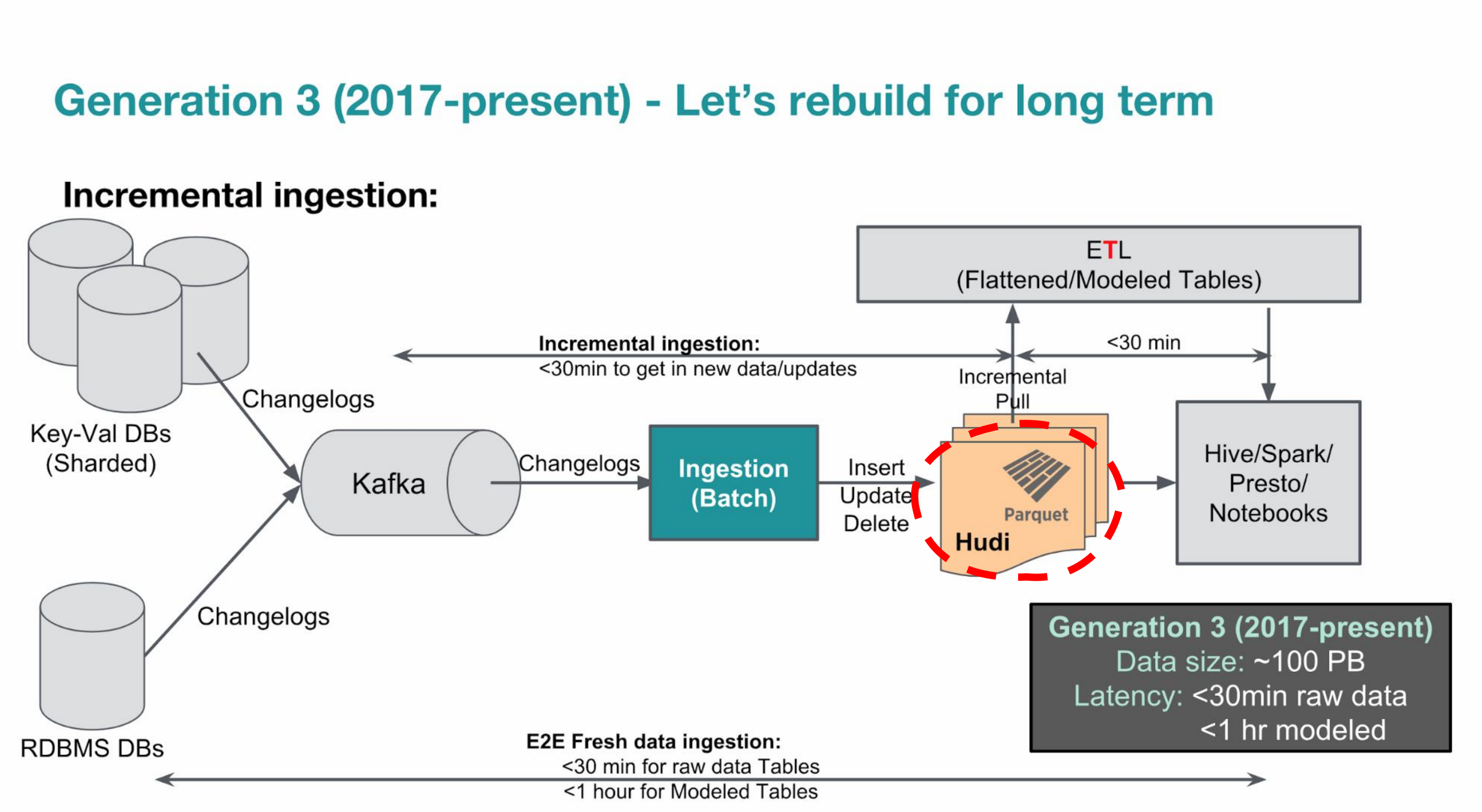
How records are *actually* represented…

Encoded in protobuf or avro
(for row representations)

Encoded in parquet
(for column representations)


users
users
users
Frontend
Frontend
Frontend
Backend
Backend
Backend
OLTP RDBMS
OLTP RDBMS
OLTP RDBMS
ELT
load row representation, *transform* into column representation
Parquet
Data Lake
"traditional" Data Warehouse
Unified Metadata + Governance Layer

Lakehouse



Uber's data platform (circa 2018)

https://www.uber.com/en-CA/blog/uber-big-data-platform/

What is Hudi

Apache Hudi is an open data lakehouse platform, built on a high-performance open table format to bring database functionality to your data lakes. Hudi reimagines slow old-school batch data processing with a powerful new incremental processing framework for low latency minute-level analytics.


冨嶽三十六景
神奈川沖
浪裏