Module 3

Methods for Big Data Analysis

ALY 6110: Big Data and Data Management

Instructor: Ajit Appari

Module 3: Topics

- Carried forward from Module 2:
  - Data Quality, Cleanup and Pre-processing
- Data Analytics Vs Data Science

- Big Data Methods (Technologies)
  - Map Reduce
  - Lambda Architecture
- Big Data Properties
  - Examples
  - Raw Data
  - Structured and Unstructured Data
  - Immutable Data
- Big Data Storage Models
  - Fact based Model
  - Graph Schema

thrift.apache.org

Nathan Marz, James Warren. *Big Data Principles and Best Practices of Scalable Real-Time Data Systems*

Data Quality: Missing Values

- There are several causes of missing values:
  - Attributes values were not recorded for certain time.
  - Value of an attribute being unknown at time of collection.
  - Distraction, misunderstanding or refusal at time of collection.
  - Attribute not required for some of the objects.
  - Non-existence of a value.
  - Fault in the data collection device.
  - Costly or difficult to assign a class label to an object in classification problems.
- Since many data analysis techniques were not designed to deal with a data set with missing values, the data set must be pre-processed.
- What is the Solution?
  - Ignore them (remove observations with missing values)
  - Impute missing values using appropriate algorithm {k-NN algorithm}: missing values are replaced as a function of other correlated non-missing columns

Data Quality: Inconsistency

- A data set can also have inconsistent values.
- The presence of inconsistent values in a data set usually reduces the quality of the model induced by ML algorithms.
- Inconsistent values can be found in the predictive and/or target attributes.

- An example of an inconsistent value in a predictive attribute is a zip code that does not match the city name. This inconsistency can be due to a mistake or a fraud.

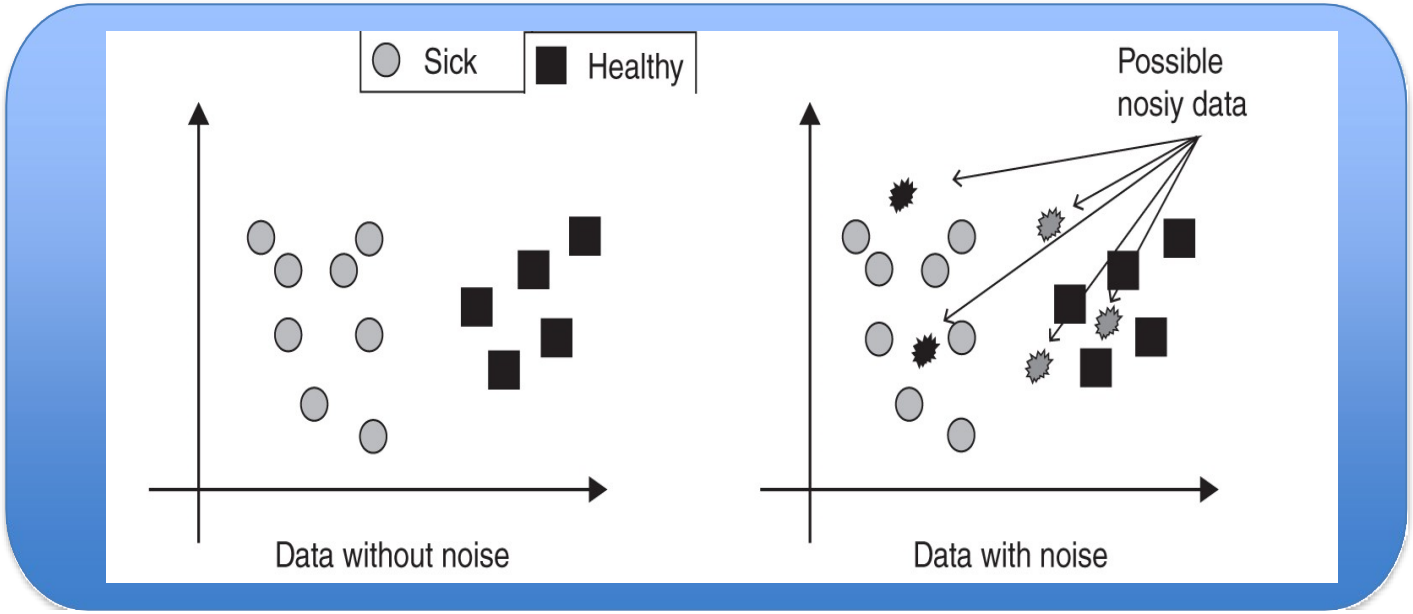
- A good policy to deal with inconsistent values in the predictive attribute is to treat them as missing values.

Data Quality: Redundancy

- Redundant objects are those that do not bring any new information to a data set.
- Objects very similar to other objects represent an irrelevant data
- Redundancy occurs mainly in the whole set of attributes.
- Redundant data could be due to small mistakes or noise in the data collection (same addresses for people whose names differ by just a single letter).
- Redundant data can be duplicate data. Deduplication is a preprocessing technique whose goal is to identify and remove copies of objects in a data set

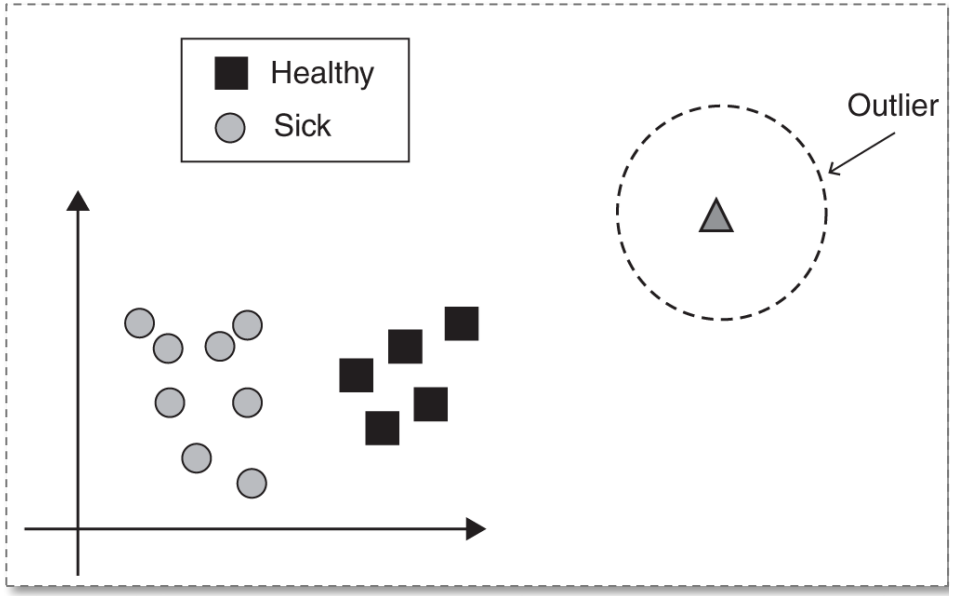
Data Quality: Noise

- Noisy data are data that do not meet the set of standards expected for them.
- Noise can be caused by incorrect or distorted measurements, human
- error or even contamination of the samples.
- Noise detection can be performed by adaptation of classification algorithms or by the use of noise filters for data preprocessing.



Data Quality: Outliers

- In a data set, outliers are anomalous values or objects.
- They can also be defined as objects whose values for one or more predictive attributes are very different from the values found in the same predictive attributes of other objects.
- In contrast to noisy data points, outliers can be legitimate values.
- There are several data analysis applications whose main goal is to find outliers in a data set.
- Particularly in anomaly detection tasks, the presence of outliers can indicate the presence of noise.

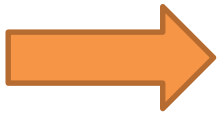


Data Pre-Processing

- **Converting to a Different Scale Type – Nominal to Relative**
  - Some ML algorithms can use only data of a particular scale type.
  - Data can be converted from a qualitative scale to a quantitative one.

Original Data

Food	Age	Distance	Company
Chinese	51	Close	Good
Italian	43	Very close	Good
Italian	82	Close	Good
Burgers	23	Far	Bad
Chinese	46	Very far	Good
Chinese	29	Too far	Bad
Burgers	42	Very far	Good
Chinese	38	Close	Bad
Italian	31	Far	Good



Converted Data

F1	F2	F3	Age	Distance	Company
0	0	1	51	Close	Good
0	1	0	43	Very close	Good
0	1	0	82	Close	Good
1	0	0	23	Far	Bad
0	0	1	46	Very far	Good
0	0	1	29	Too far	Bad
1	0	0	42	Very far	Good
0	0	1	38	Close	Bad
0	1	0	31	Far	Good

Data Science Vs Data Analytics

- Data Science: is basic science engine behind the analysis of data to produce meaningful inferences. It involves:
  - Mathematical and/or Statistical foundations,
  - Development of new Mathematical and/or Statistical Models
  - Enhancement of existing Mathematical and/or Statistical Models
  - Development of Computing Algorithms that implements mathematical or statistical models

- Data Analytics: is applied data science, i.e., application of mathematical and/or statistical models to solve business problems.
  - Creating visuals,
  - Creating explanatory models
  - Creating prediction models

Data Science Vs Data Analytics

- Scalability for processing big data
- One way to obtain scalability is by distributing the data processing tasks into several computers, which can be combined into clusters of computers.
- Clusters produced by analytics techniques (not computer clusters) is where a data set is partitioned in groups within it

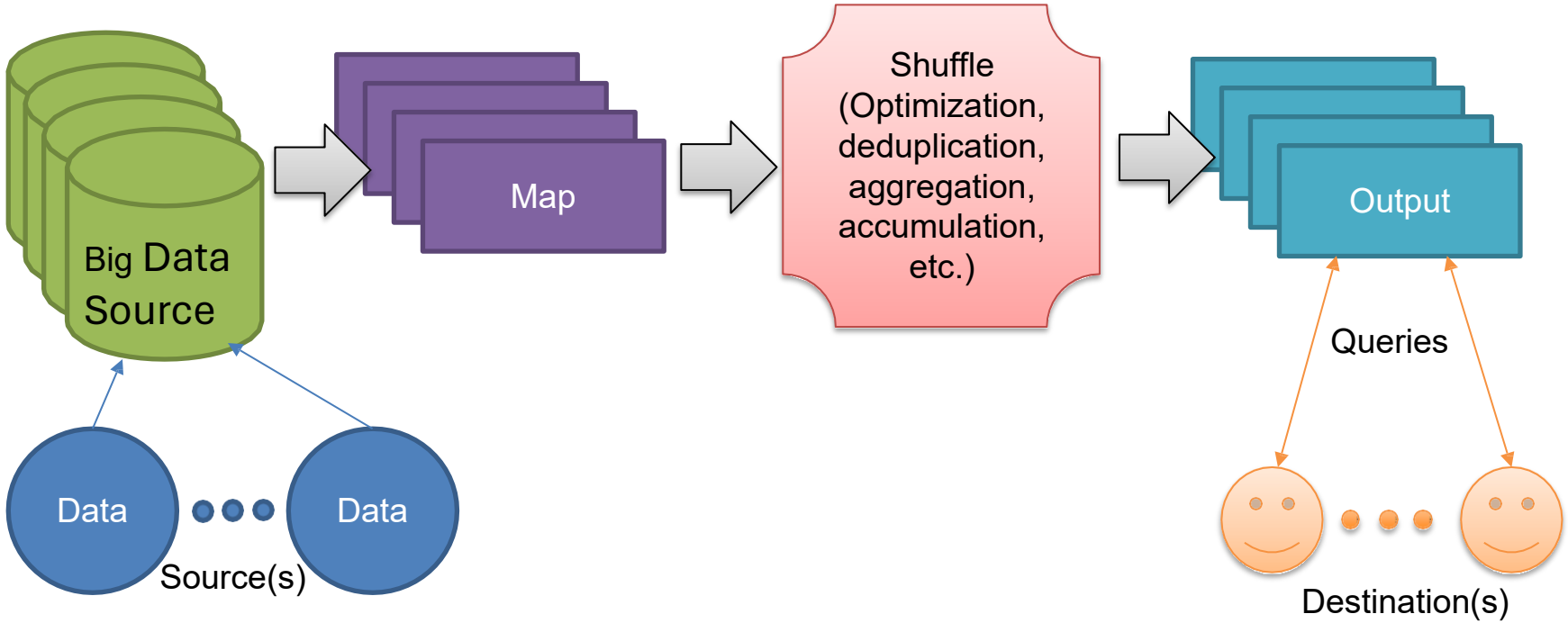
Map Reduce

Map Reduce for Cluster Computing

- **Scale Problem with Big Data:** Even if processing power is expanded by combining several computers in a cluster (i.e. a distributed system), the conventional software for distributed systems cannot keep up with big data.
  - One of the key limitations is the efficient distribution of data among multiple processing and storage units.
- To deal with these requirements, new software tools and techniques have been developed.
  - That is how Google came into existence.

Map Reduce for Cluster Computing

- **MapReduce:** One of the first techniques developed for big data (web traffic data, search of web pages) processing using clusters.
  - MapReduce is a programming model that consists of two steps: ***map*** and ***reduce***.
- **Hadoop:** The most famous implementation of MapReduce.



System Requirements for Big Data Analytics

- To efficiently solve a big data problem, a distributed system must attend the following requirements:
  - Make sure that no chunk of data is lost and the whole task is concluded.
    (If one or more computers has a failure, their tasks, and the corresponding data chunk, must be assumed by another computer in the cluster.)
  - Repeat the same task, and corresponding data chunk, in more than one cluster computer (redundancy).
    (If one or more computer fails, the redundant computer carries on with the task)
  - Failed computers can return to the cluster when they are fixed.
  - Computers can be easily removed from the cluster or extra ones included in it as the processing demand changes.

MapReduce Example

- MapReduce divides the data set into parts (chunks) and stores in the memory of each cluster computer the chunk of the data set needed by this computer to accomplish its processing task.
- As an example: suppose that you need to calculate the average salary of 1 billion people and you have a cluster with 1000 computers, each with a processing unit and a storage memory.
- The people can be divided into 1000 chunks (subsets) with data from 1 million people each.
- Each chunk can be processed independently by one of the computers.
- The results (1000) produced by each these computers (the average salary of 1 million people) can be averaged, returning the final salary average.

Lambda Architecture

Big Data Architecture and Models: Lambda Architecture

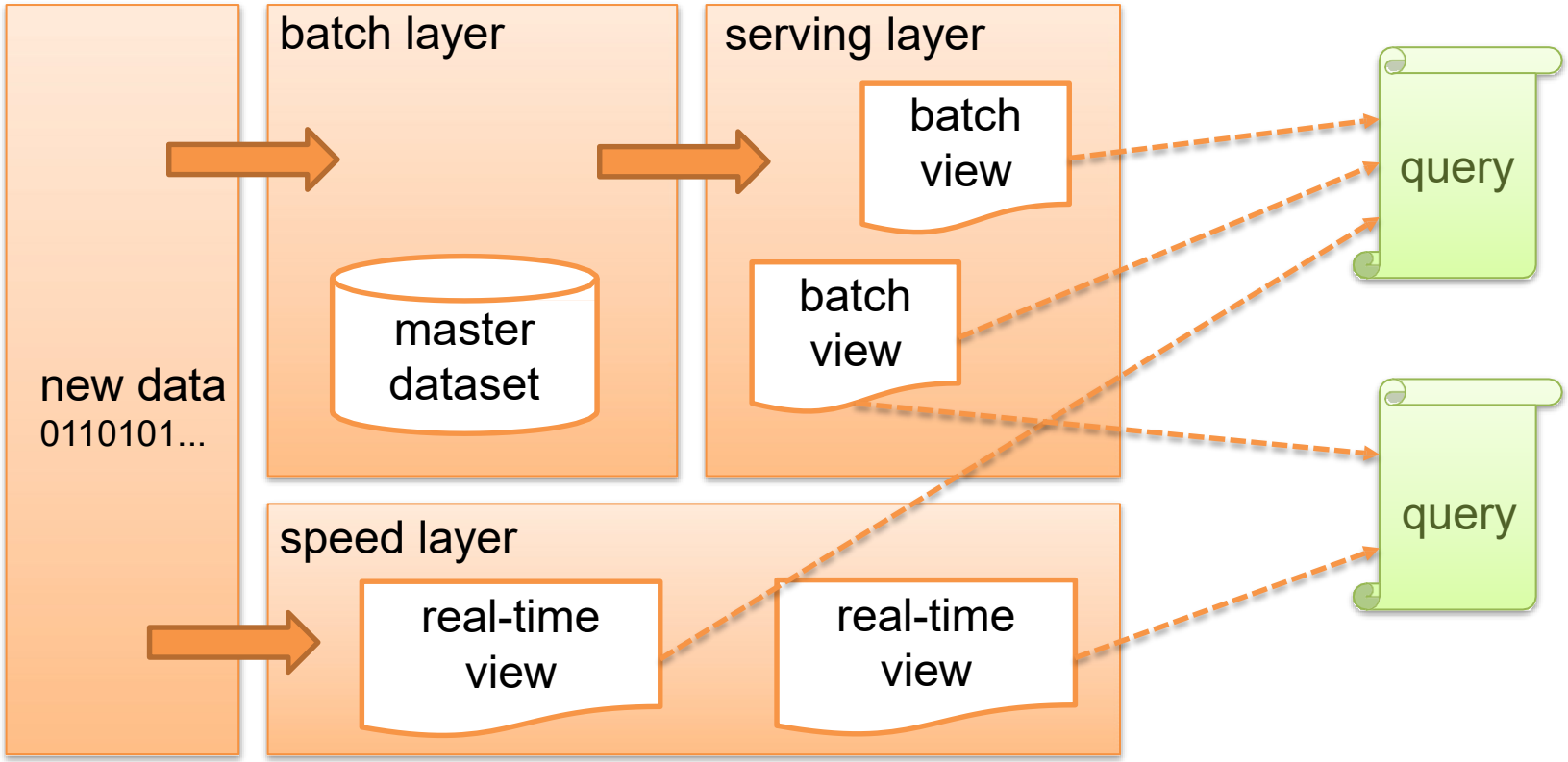
- Nathan Marz (2013) developed the concept of Lambda Architecture (LA) for a generic, scalable and fault-tolerant data processing architecture, based on his experience working on distributed data processing systems at Backtype (his start-up and Twitter.

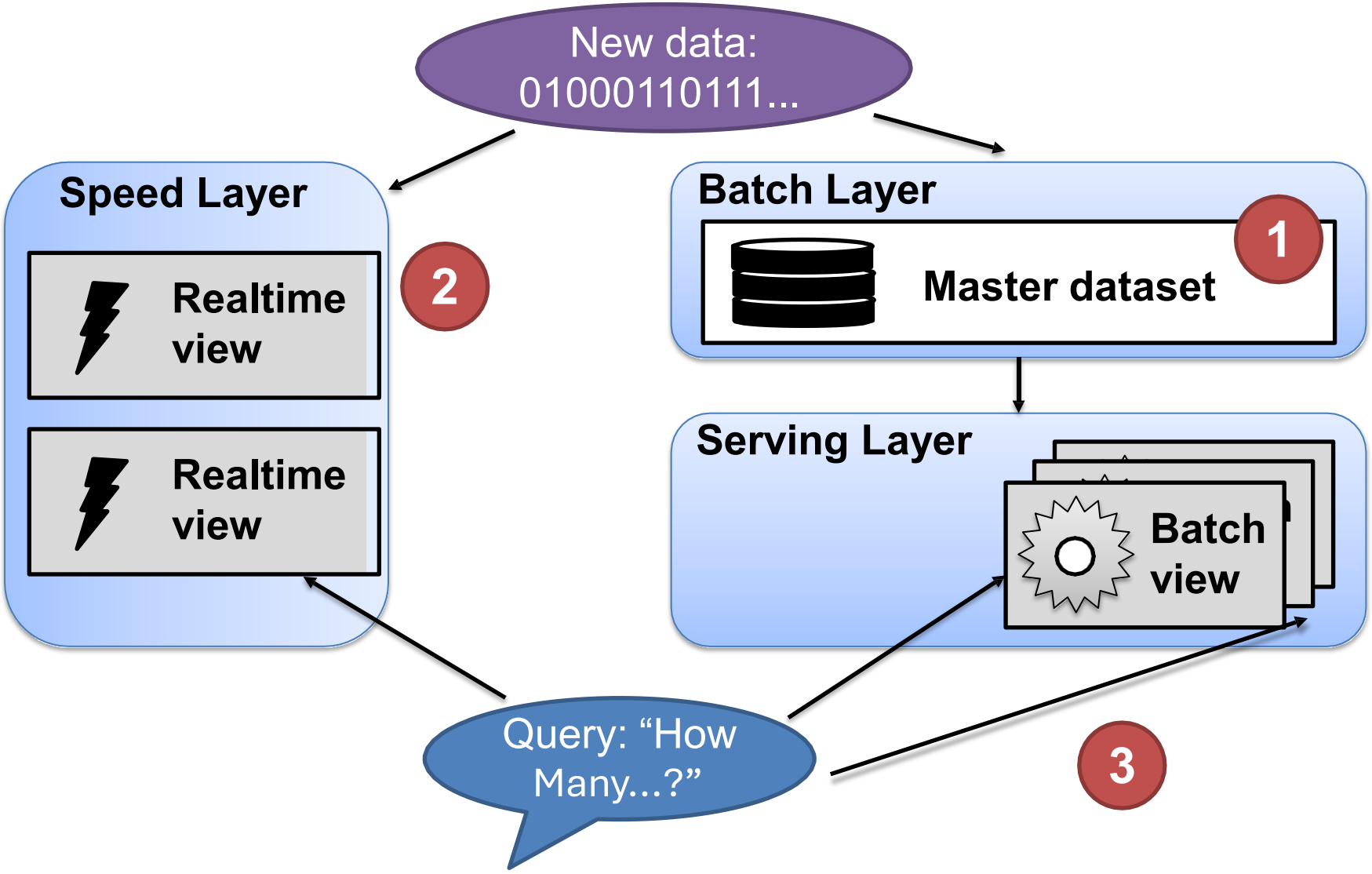
- The Lambda Architecture aims to satisfy the needs for a robust system that is fault-tolerant, both against hardware failures and human mistakes, being able to serve a wide range of workloads and use cases, and in which low-latency reads and updates are required.

- The resulting system should be linearly scalable, and it should scale out rather than up.

Big Data Architecture and Models: Lambda Architecture

- The main idea of the Lambda Architecture is to build Big Data systems as a series of layers
- Lambda Architecture Overview:




New data:
01000110111...
Speed Layer
Realtime view
Realtime view
2
Batch Layer
Master dataset
1
Serving Layer
Batch view
Query: “How Many...?”
3

1

The master dataset is the source of truth in Big Data system and cannot withstand corruption.

2

The data in the speed layer realtime views has a high turnover rate, so any errors are quickly expelled.

3

Any errors introduced into the serving layer batch views are overwritten because they are continually rebuilt from the master dataset.

Big Data Architecture and Models: Lambda Architecture

- All data entering the system is dispatched to both the **batch layer** and the **speed layer** for processing.
- The batch layer has two functions:
  - managing the master dataset (an immutable, append-only set of raw data)
  - pre-compute the batch views.

- The serving layer indexes the batch views so that they can be queried in low-latency, ad-hoc way.
- The speed layer compensates for the high latency of updates to the serving layer and deals with recent data only.
- Any incoming query can be answered by merging results from batch views and real-time views.

Big Data Properties

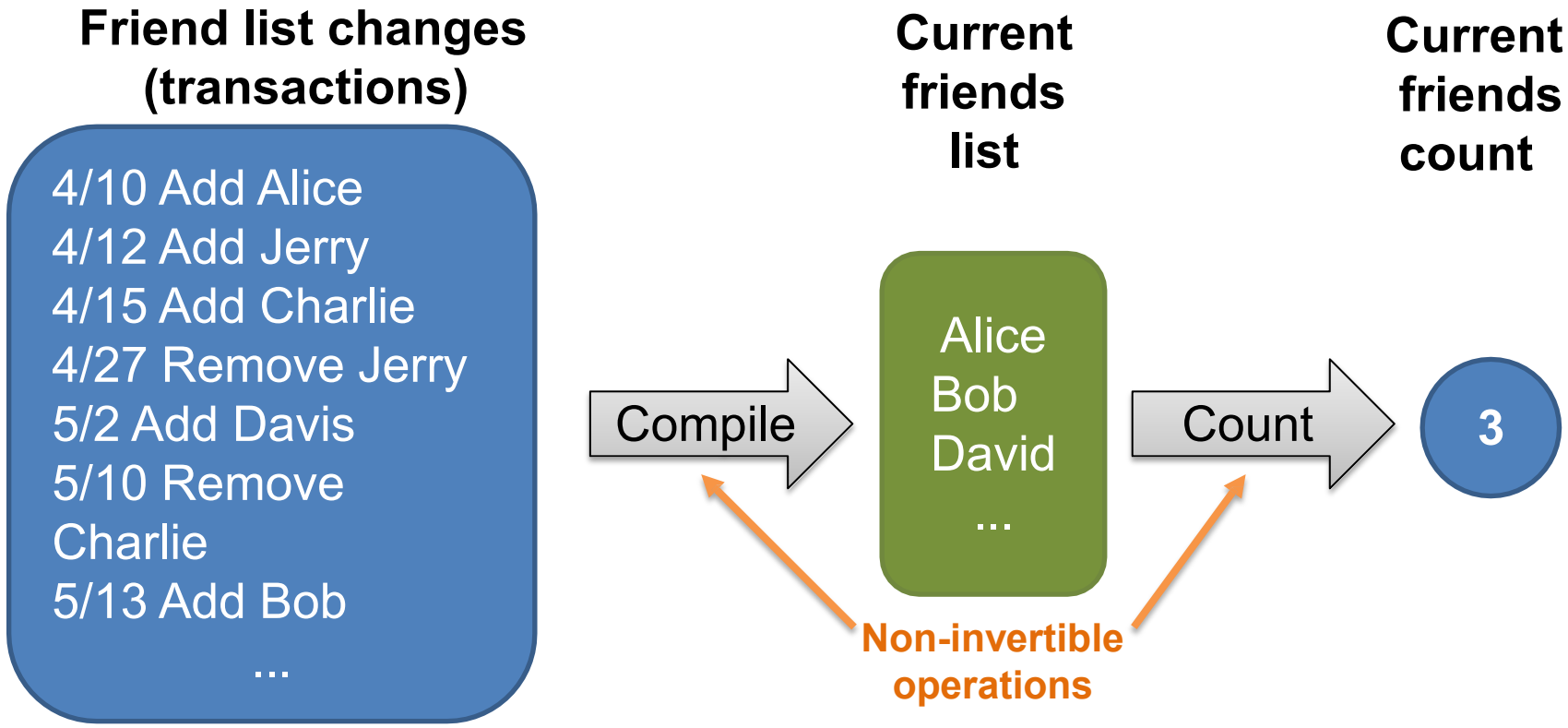
Big Data Properties

- Information is the general collection of knowledge relevant to your Big Data system.
- Data refers to the information that can't be derived from anything else. Data serves as the axioms from which everything else derives.
- Queries are questions you ask of your data. (query financial transaction history to determine your current bank account balance).
- Views are information that has been derived from your base data. They are built to assist with answering specific types of queries.

Big Data Storage Models

Social Network System

- Example of FaceData social network system (your own FaceBook)
- There are three possible options for storing friendship information for Social Network System.
- Each option can be derived from the one to its left, but it's a one-way process.



Why Do We Need Performance?

- Prior to Big Data era the term Data Warehousing was widely used.
- The key difference between the two is that Data Warehousing just one part of the Big Data System, which primary responsibility is performance.
- The idea is to pre-calculate the data to serve the most common queries faster – create views in advance
- The native human behavior is to expect most current data immediately. There is no patience to wait for what is here as of Today.
- At the same time natively people would understand that if historical data needed to be extracted, it should take more time and there is more patience there.

Inventory Status Report Example

- Most common example from the business side of Big Data Counting and Management is a Inventory Status report.
- Lets say company buys and sells small number of parts (part numbers) but in large volumes
- Each transaction is stored for each part for with the date
- Receiving transaction recorded with “+” quantity amount
- Selling transaction recorded will “-” quantity amount

Transactions History

Date	Part Number	Qty
01/05/2016	Part A	+1,500
02/09/2016	Part A	-350
02/09/2016	Part B	+25,000
02/10/2016	Part C	+15,000
02/11/2016	Part B	-5,000
03/01/2016	Part C	+12,000
03/01/2016	Part B	-5,500
03/02/2016	Part C	-10,000
03/03/2016	Part A	+12,000
03/04/2016	Part B	- 1,000
...		

Inventory Status Report Example

- To calculate the current inventory status report, we use a simple formula:
- For each Part Number, calculate the summary or quantity
  - As a result we have a table were each part only shown once
- The quantity field shows the total quantity of this part we currently have in the inventory.
- It runs every day as the decision has t be made to spend more money buying more parts or not

Inventory Status Report as of Now

Part Number	Current Qty
Part A	13,150
Part B	13,500
Part C	17,000
Part D	25,718
Part E	6,789
Part F	123,238

Stock Exchange Example

- Stock market trading is a fountain of information, with millions of shares and billions of dollars changing hands on a daily basis.
- With so many trades taking place, stock prices are historically recorded daily as an opening price, high price, low price, and closing price.
- But those bits of data often don't provide the big picture and can potentially skew your perception of what happened.
- For instance, lets look at data that show the price data for Google, Apple, and Amazon stocks on a day when Google announced new products targeted at their competitors.

Stock Exchange Example

- This data suggests that Amazon may not have been affected by Google's announcement, as its stock price moved only slightly.
- The data also suggest that the announcement had either no effect on Apple, or a positive effect.
- Financial reporting promotes daily net change in closing prices.
- What conclusion would you draw about the impact of Google's announcements?

Company	Symbol	Previous	Open	High	Low	Close	Net
Google	GOOG	564.68	567.70	573.99	566.02	569.30	+4.62
Apple	AAPL	572.02	575.00	576.74	571.92	574.50	+2.48
Amazon	AMZN	225.61	225.01	227.50	223.30	225.62	+0.01

Stock Exchange Example

- But if you have access to data stored at a finer time granularity, you can get a clearer picture of the events on that day and probe further into potential cause and effect relationships.
  - Apple held steady throughout the day
  - Amazon's stock dipped in late day trading
  - Google's stock price had a slight boost on the day of the announcement

- The minute-by-minute relative changes in stock prices suggests that both Amazon and Apple were indeed affected by the announcement, Amazon more so than Apple.
  - Additional data can suggest new ideas not considered when examining the original daily stock price summary.

Stock Exchange Example

- Storing raw data is hugely valuable because you rarely know in advance all the questions you want answered.
- By keeping the rawest data possible, you maximize your ability to obtain new insights, whereas summarizing, overwriting, or deleting information limits what your data can tell you.
- Big Data technologies are designed to manage petabytes and exabytes of data.
- Specifically, they manage the storage of your data in a distributed, scalable manner while supporting the ability to directly query the data.

Data is Raw

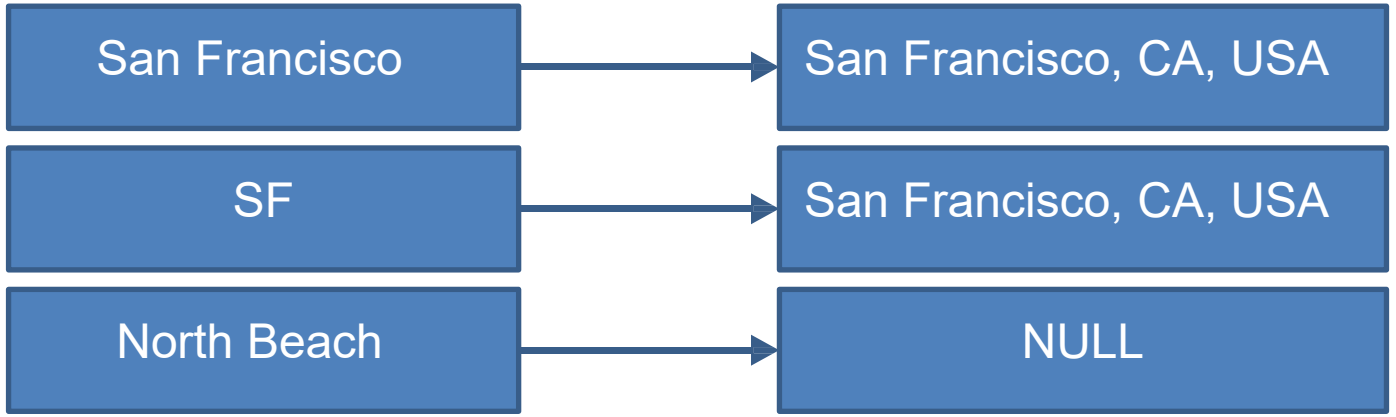
The Data is Raw

- A data system answers questions about information you've acquired in the past.
- When designing your Big Data system, you want to be able to answer as many questions as possible.
- The master data set is more valuable than views because view can be regenerated from the master data set data but not the other way around.
- We call this property rawness of data
- Although the concept is straightforward, it's not always clear what information you should store as your raw data.

Unstructured vs Normalized

Unstructured vs Normalized

- The unstructured data is rawer than normalized data.
- When deciding what raw data to store, a common hazy area is the line between parsing and semantic normalization.
- Semantic normalization is the process of reshaping freeform
- information into a structured form of data.
- Semantic normalization of unstructured location responses to city, state, and country.



- A simple algorithm will normalize “North Beach” to NULL if it doesn’t recognize it as a San Francisco neighborhood.

Unstructured vs Normalized

- If you come across a form of data such as an unstructured location string, should you store the unstructured string or the semantically normalized form?
  - If you store the unstructured string, you can renormalize that data at a later time when you have improved your algorithms.
  - If your algorithm for extracting the data is simple and accurate you should store the results of that algorithm.
  - If the algorithm is subject to change, due to improvements or broadening the requirements, store the unstructured form of the data.

Unstructured vs Normalized

- More information does not necessary mean more rawer data
- Let's say that Tom is a blogger, and he wants to add his posts to his FaceSpace profile.
- What exactly should you store once Tom provides the URL of his blog?
- Storing the pure text of the blog entries is certainly a possibility. But any phrases in italics, boldface, or large font were deliberately emphasized by Tom and could prove useful in text analysis.
- Storing the full HTML of Tom's blog as your data will require more space: color schemes, stylesheets, JavaScripts, etc.

Data is Immutable

Data is Immutable

- Immutable data may seem like a strange concept if you're well versed in relational databases.
- In the relational database world update is one of the fundamental operations.
- For immutability you don't update or delete data, you only add more.
- By using an immutable schema for Big Data systems, you gain two vital advantages:
  - Human-fault tolerance
  - Simplicity

Data is Immutable: Human-Fault Tolerance

- Human-fault tolerance is an essential property of data systems.
- People will make mistakes, and you must limit the impact of such mistakes and have mechanisms for recovering from them.
- With a mutable data model, a mistake can cause data to be lost, because values are actually overridden in the database.
- With an immutable data model, no data can be lost.
- If “bad” data is written - earlier (good) data units still exist.
- Fixing the data system is just a matter of deleting the bad data units and re-computing the views built from the master dataset.

Data is Immutable: Simplicity

- Mutable data models imply that the data must be indexed in some way so that specific data objects can be retrieved and updated.
- In contrast, with an immutable data model you only need the ability to append new data units to the master dataset.
- This doesn't require an index for your data, which is a huge simplification.
- Storing a master dataset is as simple as using flat files.

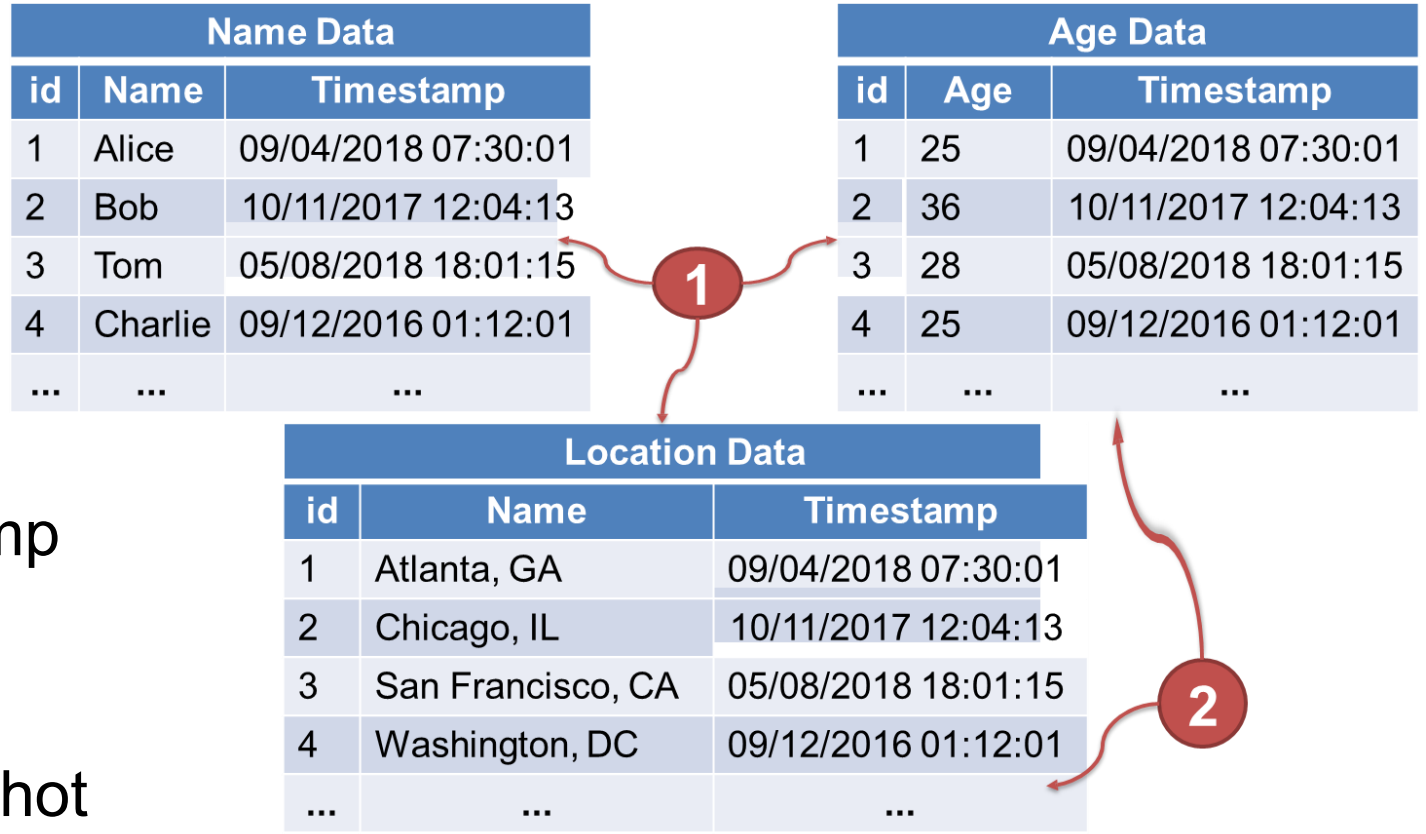
Data is Immutable

- A mutable schema for Social Network user information.
- The advantages of keeping your data immutable become evident when comparing with a mutable schema.
- When details change - say, Tom moves to Los Angeles - previous values are overwritten and lost.

id	Name	Age	Gender	Employer	Location
1	Alice	25	Female	Apple	Atlanta, GA
2	Bob	36	Male	SAS	Chicago, IL
3	Tom	28	Male	Google	San Francisco, CA
4	Charlie	25	Male	Microsoft	Washington, DC
...	...	...	...	...	...

Data is Immutable

- An immutable schema for Social Network user information.
  - Each field of user information is kept separately.
  - Each record is timestamped when it is stored.
- Each field is tracked in a separate table, and each row has a timestamp for when it's known to be true.
- Rather than storing a current snapshot of the world, as done by the mutable schema, immutable one creates a separate record every time a user's information evolves.



Data is Immutable

- Accomplishing this requires two changes.
  - Each field of user information tracked in a separate table.
  - Each unit of data is tied to a moment in time when the information is known to be true.

- When Tom moves to Los Angeles on June 17, 2018, new record added to the location table, timestamped by when he changed his profile
- Now there are two location records for Tom (user ID #3), and because the data units are tied to particular times, they can both be true.
- Tom's current location involves a simple query on the data: look at all the locations, and pick the one with the most recent timestamp.

Data is Immutable

- By keeping each field in a separate table, you only record the information that changed.
- This requires less space for storage and guarantees that each record is new information and is not simply carried over from the last record.

Location Data		
id	**Name**	**Timestamp**
1	Atlanta, GA	09/04/2018 07:30:01
2	Chicago, IL	10/11/2017 12:04:13
3	San Francisco, CA	05/08/2018 18:01:15
4	Washington, DC	09/12/2016 01:12:01
3	Los Angeles	06/17/2018 15:30:05
...	...	...

Data is Eternally True

Data is Eternally True

- The key consequence of immutability is that each piece of data is true in perpetuity.
- Each piece of data, once true, must always be true.
- Immutability wouldn't make sense without this property, this is why each piece of data tagged with a timestamp as a practical way to make data eternally true.
- In general, your master dataset consistently grows by adding new immutable and eternally true pieces of data.

Data is Eternally True

- There are some special cases, though, in which data deleted and these cases are not incompatible with data being eternally true:

- **Garbage collection:**
  - Implemented to delete all data units that have low value
  - Used to implement data retention policies that control the growth of the master dataset. (keep only one location per person per year)

- **Regulations:**
  - Government regulations may require to purge data from databases under certain conditions.

- In both cases, deleting the data is not a statement about the truthfulness of the data but instead, it's a statement about the value of the data.

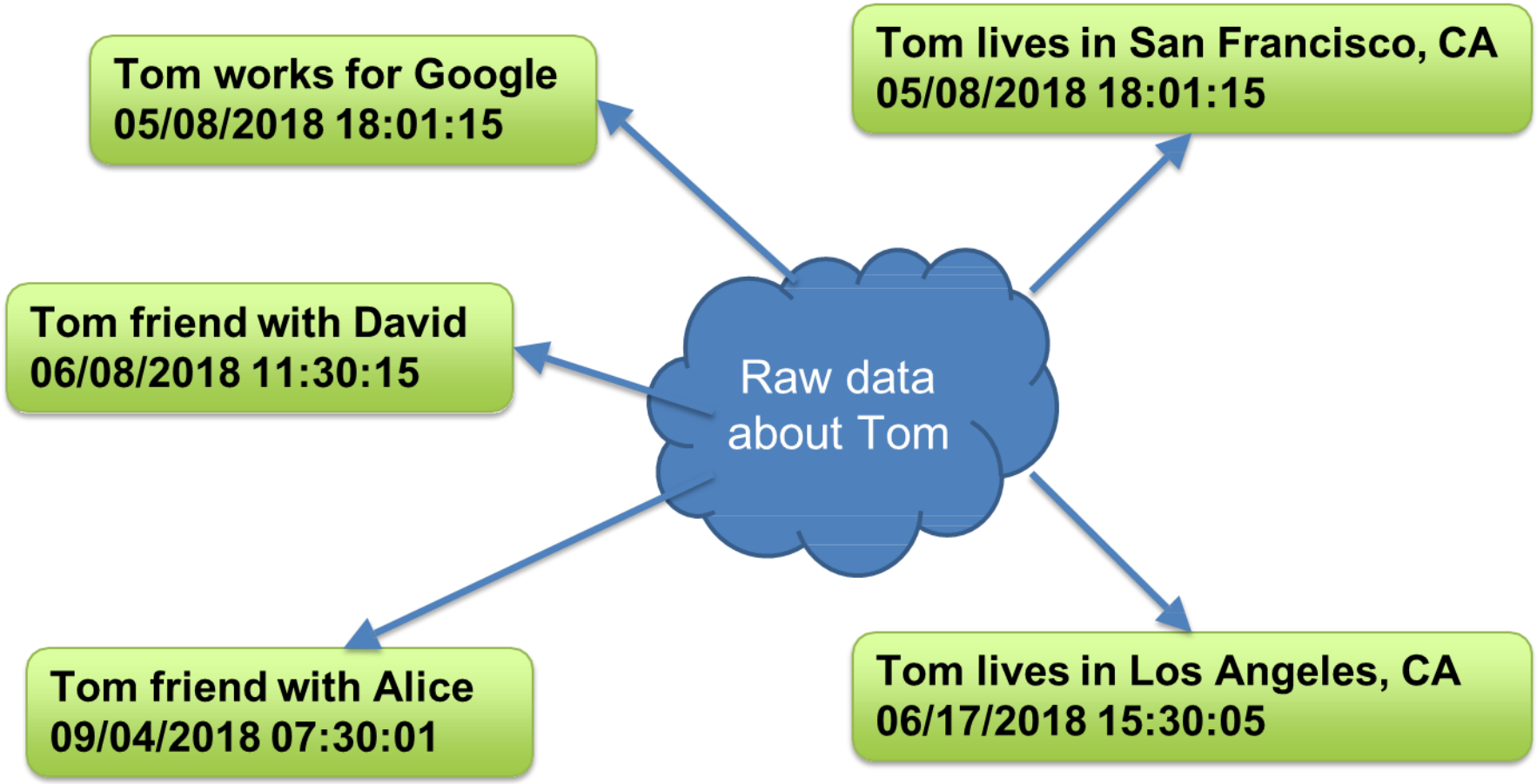
Fact-based Data Models

Fact-based Data Models

- Data is the set of information that can't be derived from anything else, but there are many ways you could choose to represent it within the master dataset.
- Besides traditional relational tables, structured XML and semi structured JSON documents there are other possibilities for storing data.
- In the fact-based model, you deconstruct the data into fundamental units called (unsurprisingly) facts.
- Facts are atomic because they can't be subdivided further into meaningful components.

Fact-based Data Models

- Collective data, such as Tom's friend list in the figure, are represented as multiple, independent facts.



Fact-based Data Models

- Identifiability- facts should be associated with a uniquely identifiable piece of data.
- If facts come at the same time, we might encounter the same exact data record
- If we encounter two identical pageview records, there's no way to tell whether they refer to two distinct events or if a duplicate entry was accidentally introduced into our dataset.
- To distinguish different pageviews, you can add a nonce to your schema - a 64-bit number randomly generated for each pageview

Fact-based Data Models

- The addition of the nonce makes it possible to distinguish pageview events from each other, and if two pageview data units are identical (all fields, including the nonce), you know they refer to the exact same event.
- Making facts identifiable means that we can write the same fact to the master dataset multiple times without changing the semantics of the master dataset.
- Queries can filter out the duplicate facts when doing their computations.
- Having distinguishable facts makes implementing the rest of the Lambda Architecture much easier.

Fact-based Data Models: Benefits

- The dataset is queryable at any time in its history
- The data is human-fault tolerant
- The data easily handles partial information
- The data storage and query processing layers are separate

Fact-based Data Models: Benefits

- The dataset is queryable at any time in its history
- Instead of storing only the current state of the world, we have the ability to query the data for any time covered by the dataset.
- This is a direct consequence of facts being timestamped and immutable.
- “Updates” and “Deletes” are performed by adding new facts with more recent timestamps, but because no data is actually removed, we can reconstruct the state of the world at the time specified by the query.

Fact-based Data Models: Benefits

- The data is human-fault tolerant
- Human-fault tolerance is achieved by simply deleting any erroneous facts.
- Suppose you mistakenly stored that Tom moved from San Francisco to Los Angeles.
- By removing the Los Angeles fact, Tom's location is automatically "reset" because the San Francisco fact becomes the most recent information.

Fact-based Data Models: Benefits

- The data easily handles partial information

- Storing one fact per record makes it easy to handle partial information about an entity without introducing NULL values into your dataset.
- Suppose Tom provided his age and gender but not his location or profession.
- Dataset would only have facts for the known information - any "absent" fact would be logically equivalent to NULL.
- Additional information that Tom provides at a later time would naturally be introduced via new facts.

Fact-based Data Models: Benefits

- The data storage and query processing layers are separate

- Another key advantage of the fact-based model that is in part due to the structure of the Lambda Architecture itself.

- By storing the information at both the batch and serving layers, we have the benefit of keeping your data in both normalized and de- normalized forms and reaping the benefits of both.

- Data normalization is completely unrelated to the semantic normalization term, it refers to storing data in a structured manner to minimize redundancy and promote consistency.

Fact-based Data Models: Benefits

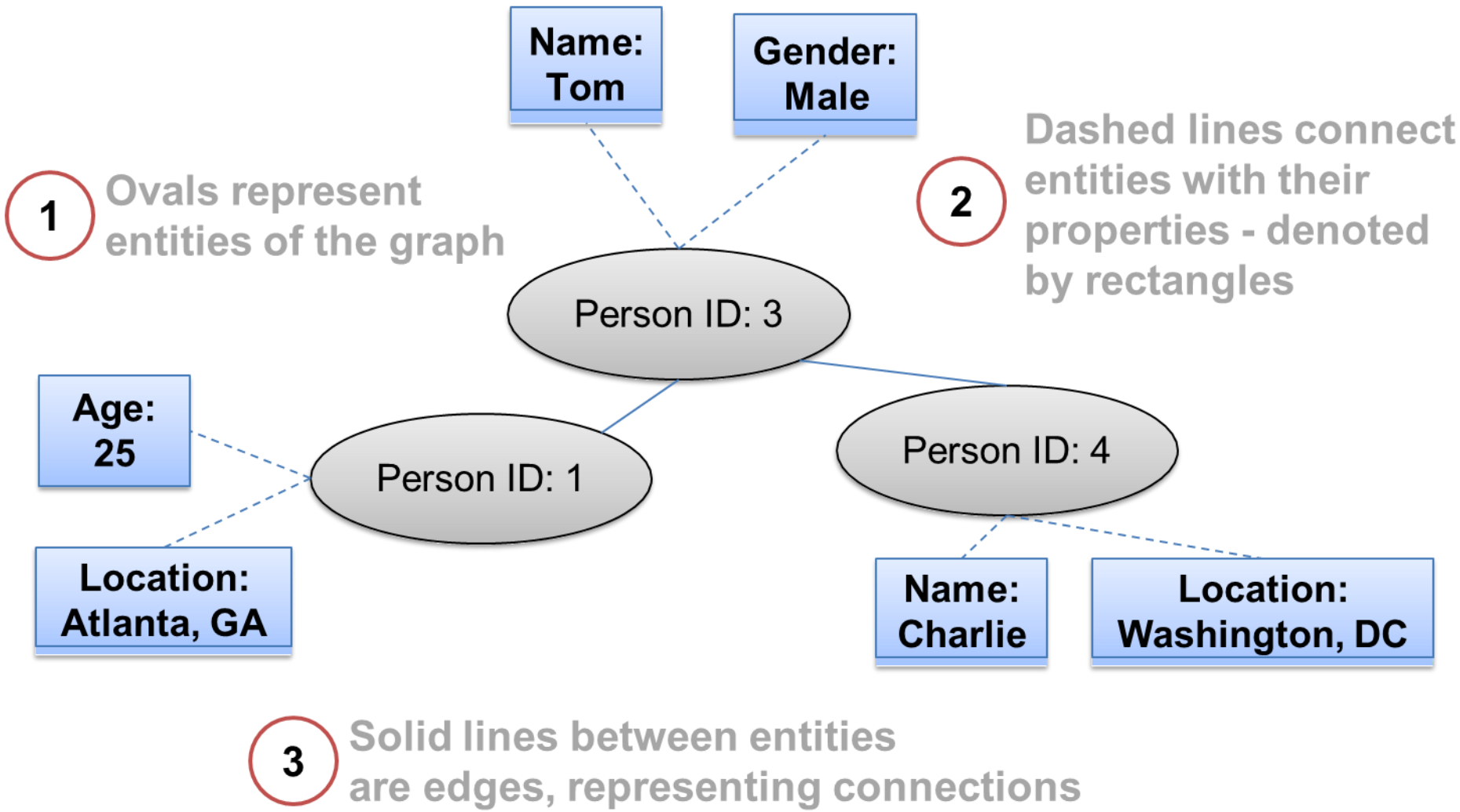
- The mutually exclusive choice between normalized and de- normalized schemas is necessary because, for relational databases, queries are performed directly on the data at the storage level.
- The objectives of query processing and data storage are cleanly separated in the Lambda Architecture.
- In the Lambda Architecture, the master dataset is fully normalized and no data is stored redundantly.
- Updates are easily handled because adding a new fact with a current timestamp “overrides” any previous related facts.

Graph schemas

Graph Schemas

- Each fact within a fact-based model captures a single piece of information.
- The facts alone don't convey the structure behind the data.
- There is no description of the types of facts contained in the dataset, nor any explanation of the relationships between them.
- Graph schemas capture the structure of a dataset stored using the fact-based model.
- A graph schema represents the relationships between the facts.
- Graph provides a useful visualization of existing facts and the relationship between them.

Graph Schemas



Graph Schemas

- There are three core components of a graph schema:
  - Nodes (Person 1, etc)
  - Edges
  - Properties (Age, Name, Location, etc)

- Nodes are the entities in the system.

- Edges are relationships between nodes.

- Properties are information about entities.

- Edges are strictly between nodes. Even though properties and nodes are visually connected, these lines are not edges. They are present only to help illustrate the association between entity and its information.