



# EECS6414:

# Data Analytics & Visualization

# **Computing Platforms, Storage Systems, Data Definition**

# Big Data Technology & Analytics

Data  
Ingestion  
ETL, Distcp,  
Kafka,  
OpenRefine,  
...

Query & Exploration  
SQL, Search, Cypher, ...

Stream Processing Platforms  
Storm, Spark, ..

Batch Processing Platforms  
MapReduce, SparkSQL, BigQuery, Hive, Cypher, ...

Data Definition  
SQL DDL, Avro, Protobuf, CSV

Storage Systems  
HDFS, RDBMS, Column Stores, Graph Databases

Data  
Serving  
BI, Cubes,  
RDBMS, Key-  
value Stores,  
Tableau, ...

Computing Platforms  
Distributed Commodity, Clustered High-Performance, Single Node

# Big Data Technology & Analytics

Data  
Ingestion  
ETL, Distcp,  
Kafka,  
OpenRefine,  
...

Query & Exploration  
SQL, Search, Cypher, ...

Stream Processing Platforms  
Storm, Spark, ..

Batch Processing Platforms  
MapReduce, SparkSQL, BigQuery, Hive, Cypher, ...

Data Definition  
SQL DDL, Avro, Protobuf, CSV

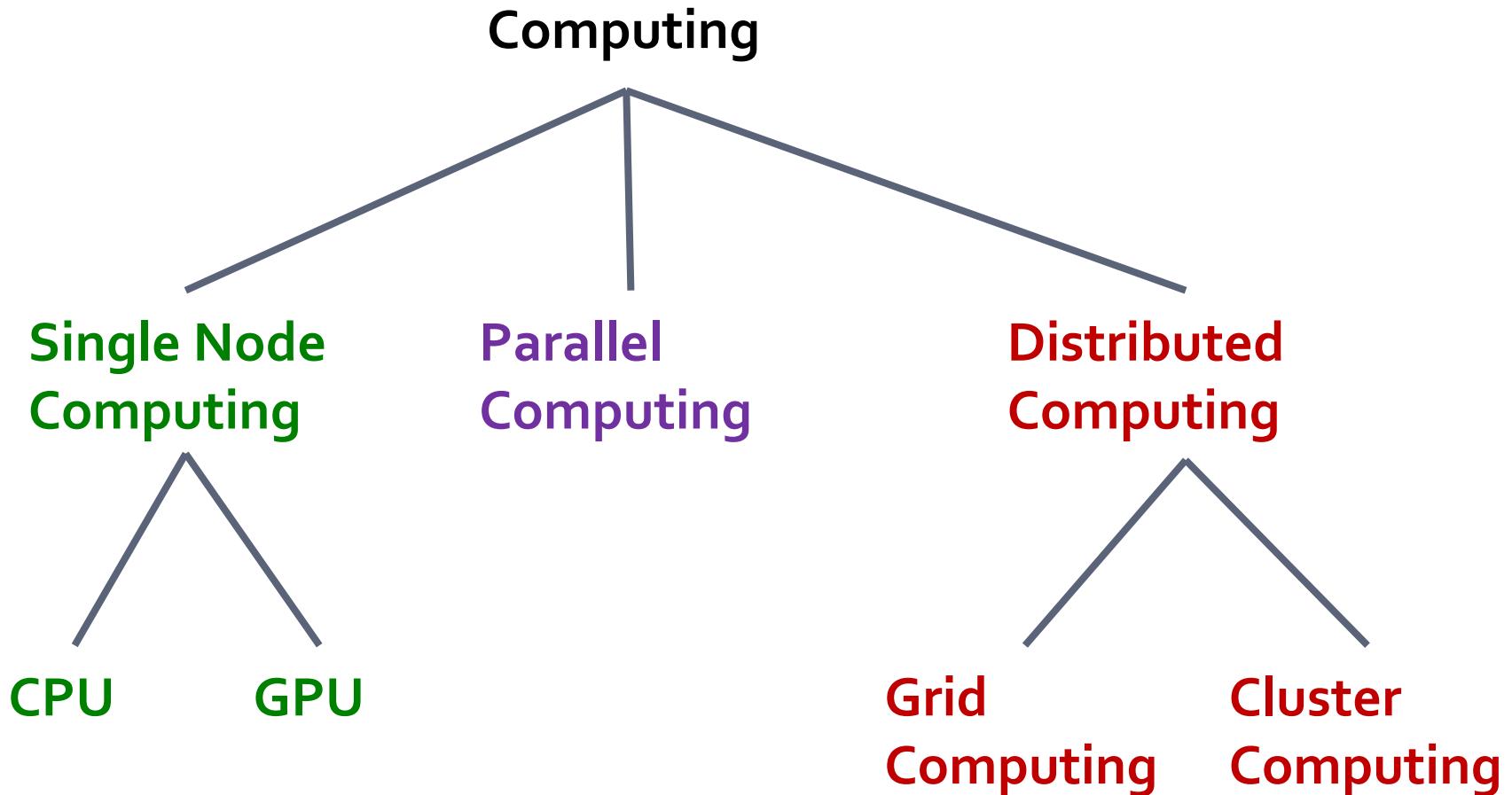
Storage Systems  
HDFS, RDBMS, Column Stores, Graph Databases

Data  
Serving  
BI, Cubes,  
RDBMS, Key-  
value Stores,  
Tableau, ...

Computing Platforms  
Distributed Commodity, Clustered High-Performance, Single Node

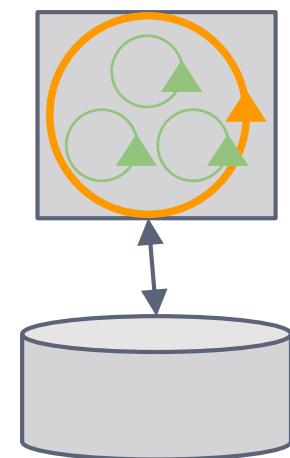
# Computing Platforms

# Computing Platforms



# Single Node Computing

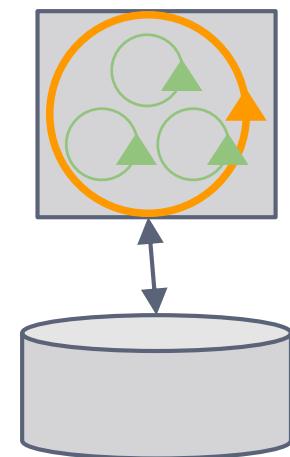
- A single node (usually multiple cores)
- Attached to a data store (Disc, SSD, ...)
- One process with potentially multiple threads



# Single Node Example

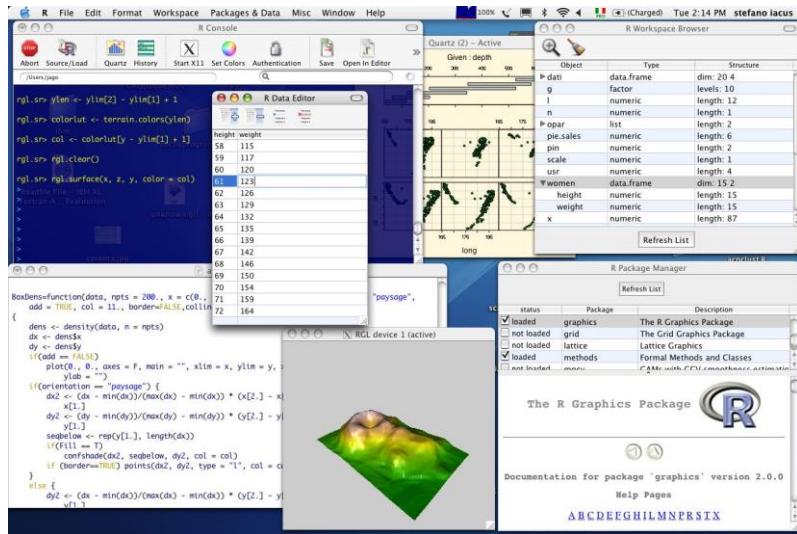
R: All processing is done on one computer

**BidMat:** All processing is done on one computer with specialized HW



# R

## Single Node In memory Retrieve/Stores from Disc

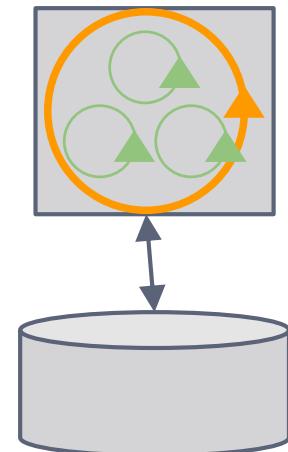


## Pros

- Simple to program and debug

## Cons

- Can only scale-up
- Does not deal with large data sets



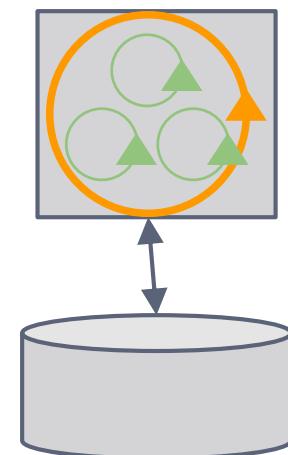
# BidMat/BidMach

Single Node solution for large scale  
exploratory analysis

Specialized HW and SW for efficient  
Matrix operations

## Elements:

- Data engine software for optimized operations
- HW design pattern for balancing Storage, CPU and GPU computing
- Optimized machine learning package
- Advanced communication patterns



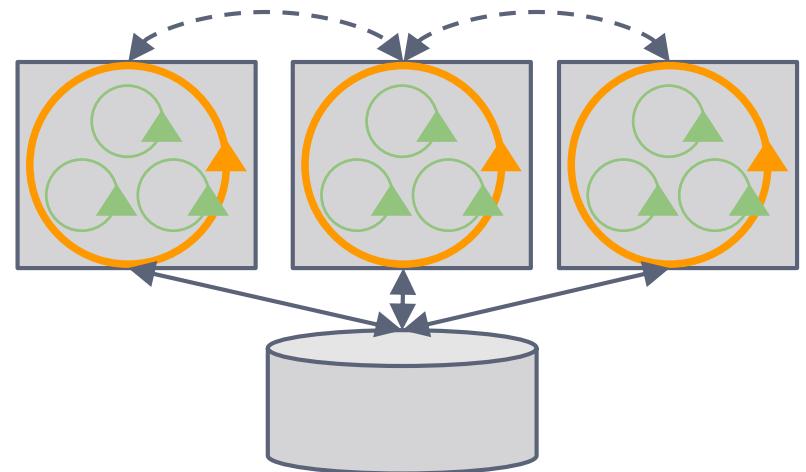
# Parallel Computing

Common Data Store

Many Processors

Parallel execution of tasks

Processor communication



# R Snow: Parallel R

R is a single thread computing application

R Snow enable multi threading/distribution

## Pros

- Distributed/parallel
- Commonly known tool and model

## Cons

- Each node requires access to all data

# Distributed Computing

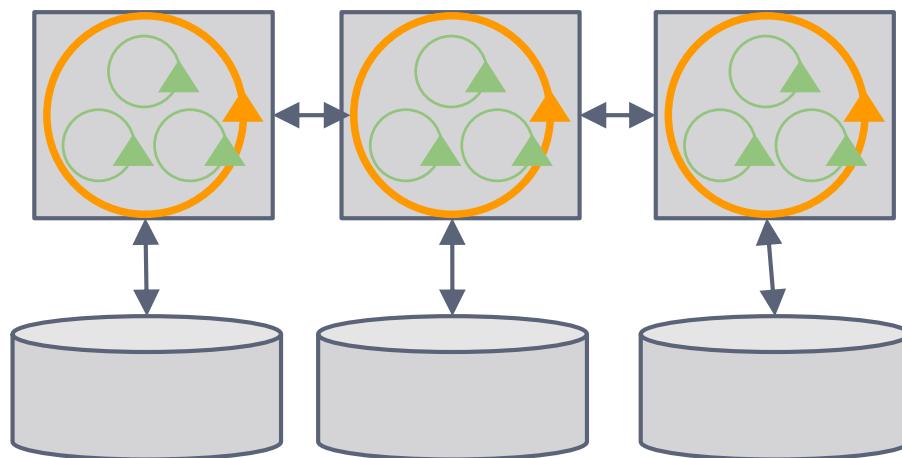
Connected processors collaborate to achieve a common goal

Requires:

- Message passing
- Coordination
- Scheduling
- Tolerate failures on individual nodes

# Cluster Computing

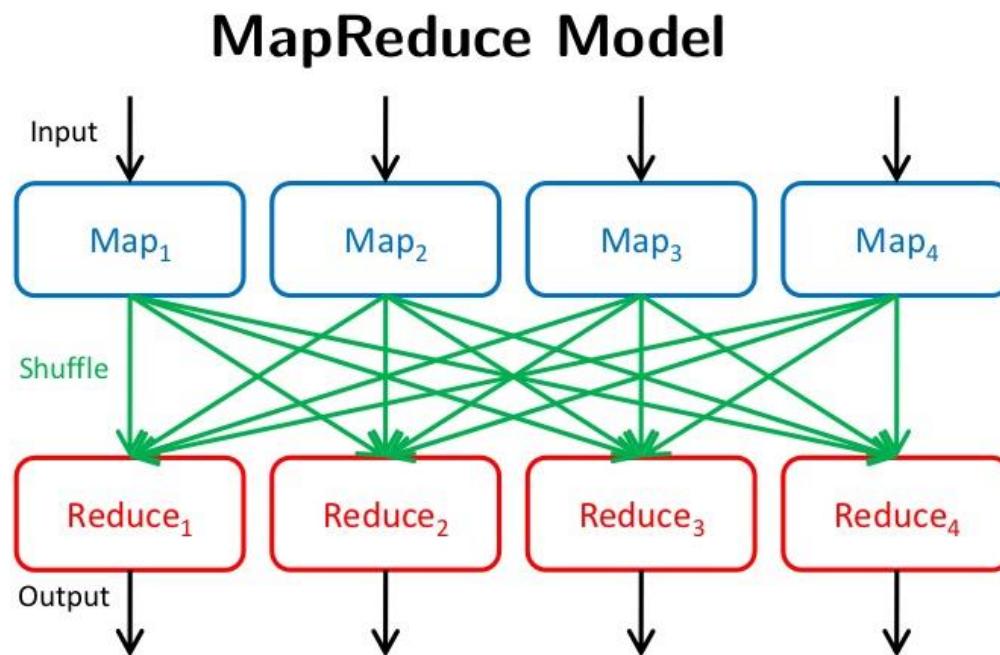
- Uniform nodes
- Data shards in a distributed storage



# Cluster Computing Example

- Hadoop
- Pregel
- Spark
- Big Data SQL
- ...

# Hadoop/Map-Reduce



# Spark - General Model

Spark  
SQL

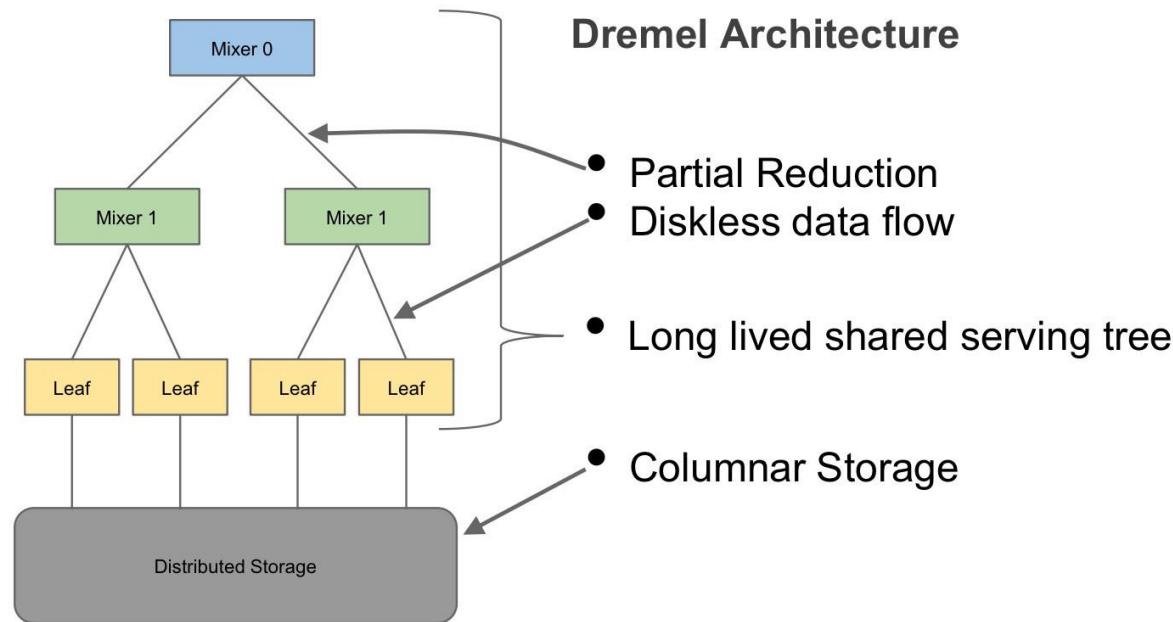
Spark  
Streaming

MLlib  
(machine  
learning)

GraphX  
(graph)

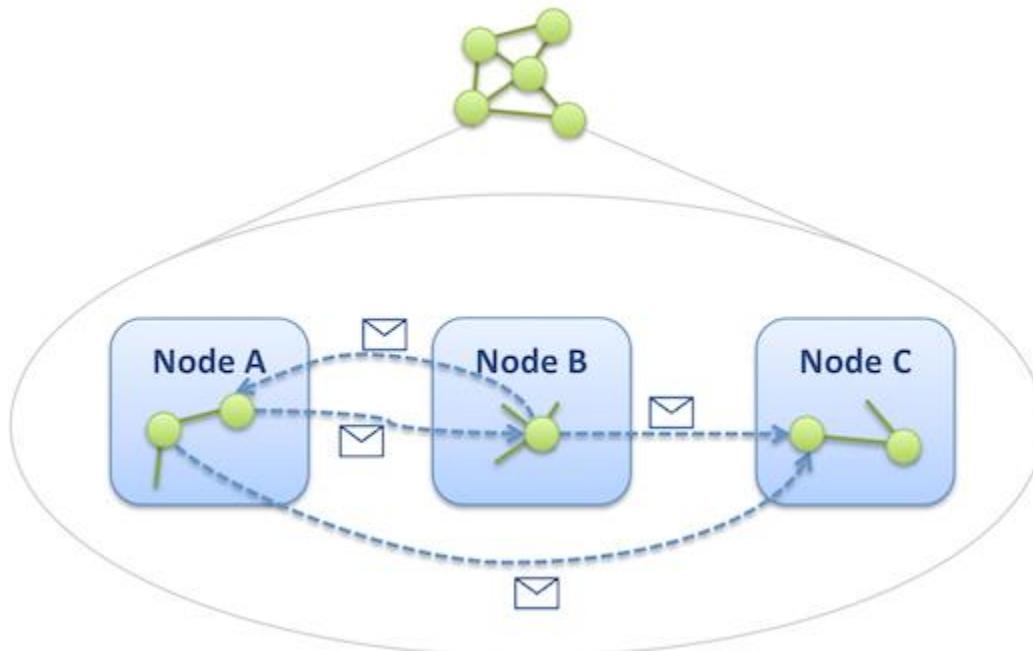
Apache Spark

# Dremel/BigQuery – General Model



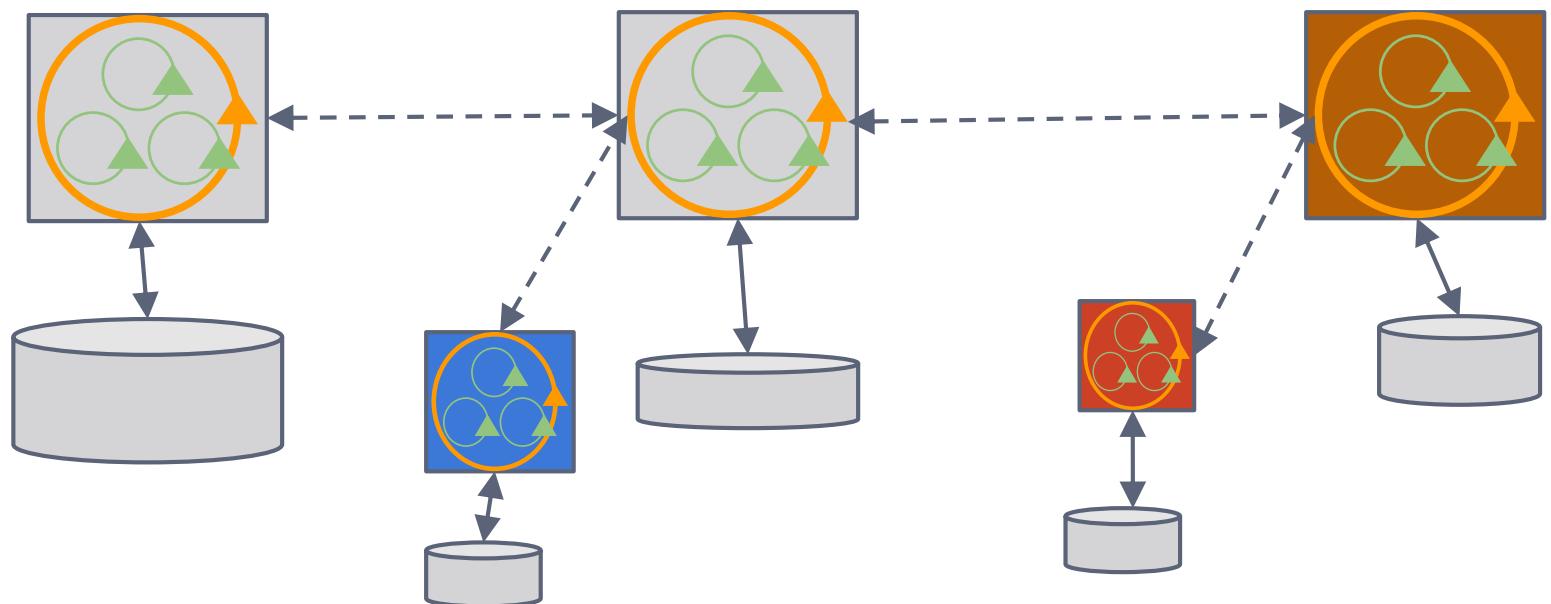
# Distributed Graph Computing

- Each node does computation
- Each node can be distributed
- Information is passed between nodes
- Execution is coordinated amongst nodes



# Grid Computing

- Distributed Nodes
- Heterogeneous and Physically Separate Nodes



# Grid Computing Example

- SETI@home (SETI Institute)
- Large Hadron Collider Computing Grid (Cern)
- NCCR Centre for Comp. Drug Discovery  
(Oxford Univ)
- Globus Toolkit (Globus Alliance)
- ...

# **Data Storage: Data Warehouse vs Data Lakes**

# Data Warehouse vs Data Lake

## Data Warehouse

- Data Transformed to defined schema
- Loaded when usage identified
- Allows for quick response of defined queries

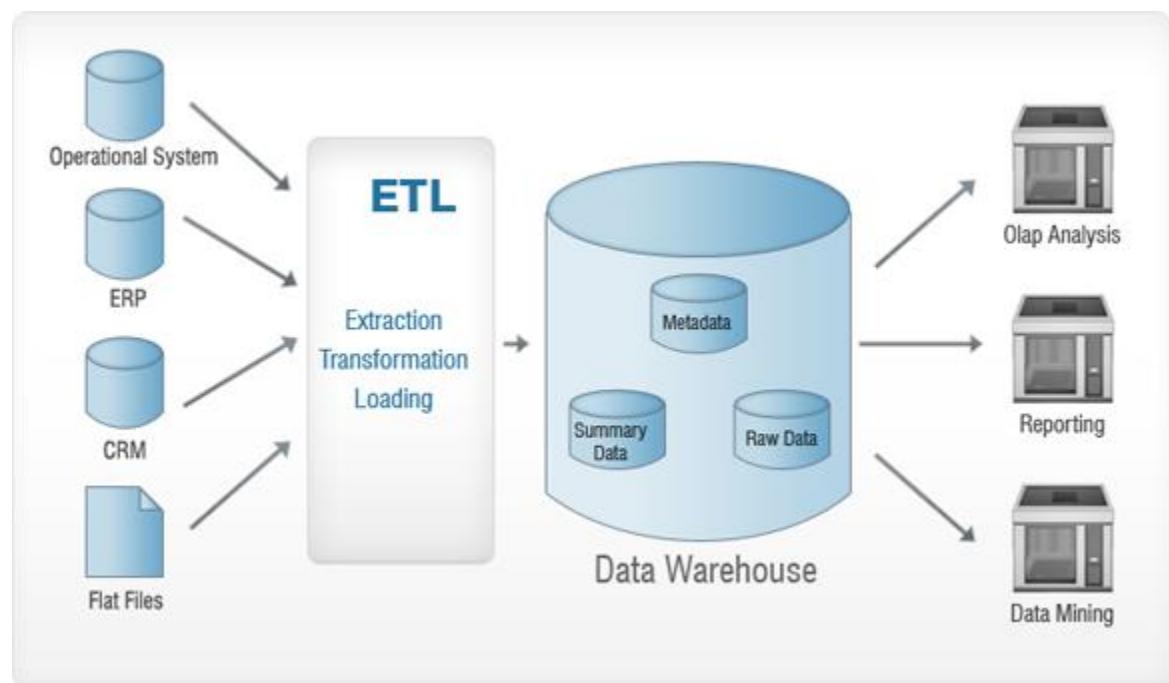
## Data Lake

- Many data sources
- Retain all data
- Allows for exploration
- Apply transform as needed
- Apply schema as needed

# Data Warehouse

Key points:

- Extract needed data
- Map to schema
- Prepare for defined use cases



# Data Lake

Key points:

- Store all data
- Transform as needed
- Apply schema as needed

Query

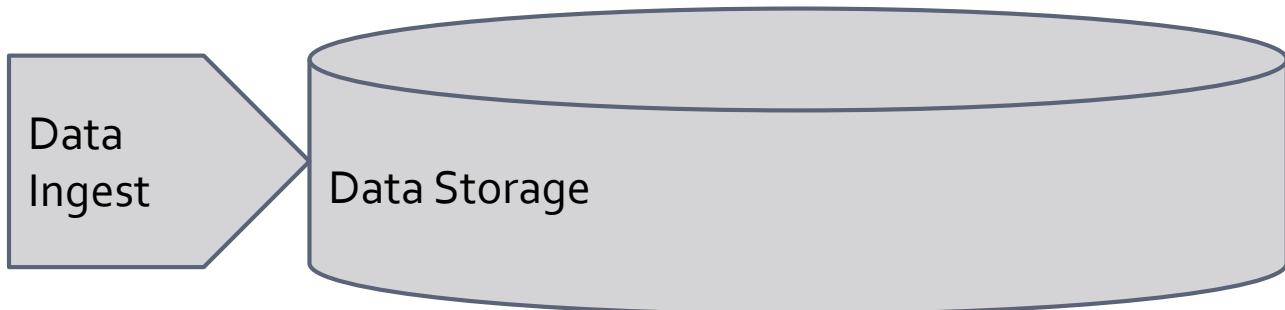
Processing

Cleaning, Transformation

Schema (Avro, Thrift, Protobuf)

Data  
Ingest

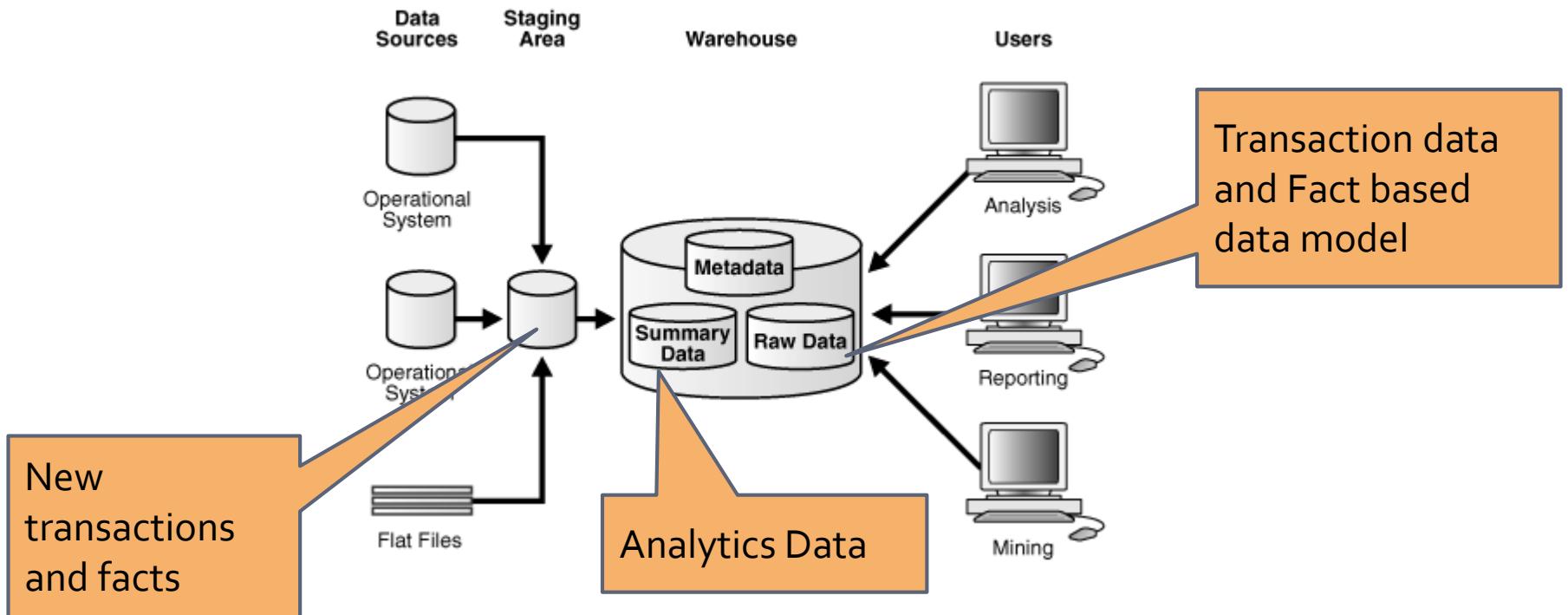
Data Storage



# Data Characteristics

- Understanding of data models and schemas in traditional BW
- Understanding of data models and schemas proposed for big data systems

# Traditional Business Warehouse



# OLTP vs OLAP

## Online Transaction Processing (OLTP)

systems and approach that facilitate and manage **transaction-oriented applications**, typically for data entry and retrieval transaction processing

## Online Analytics Processing (OLAP)

systems and approach to answering **multi-dimensional analytical queries** quickly

# OLTP vs OLAP

## OLTP

CRUD Transactions (Create,  
Read, Update, and Delete)  
Frequent Updates

## OLAP

Aggregations  
Drill-downs,  
Roll-ups

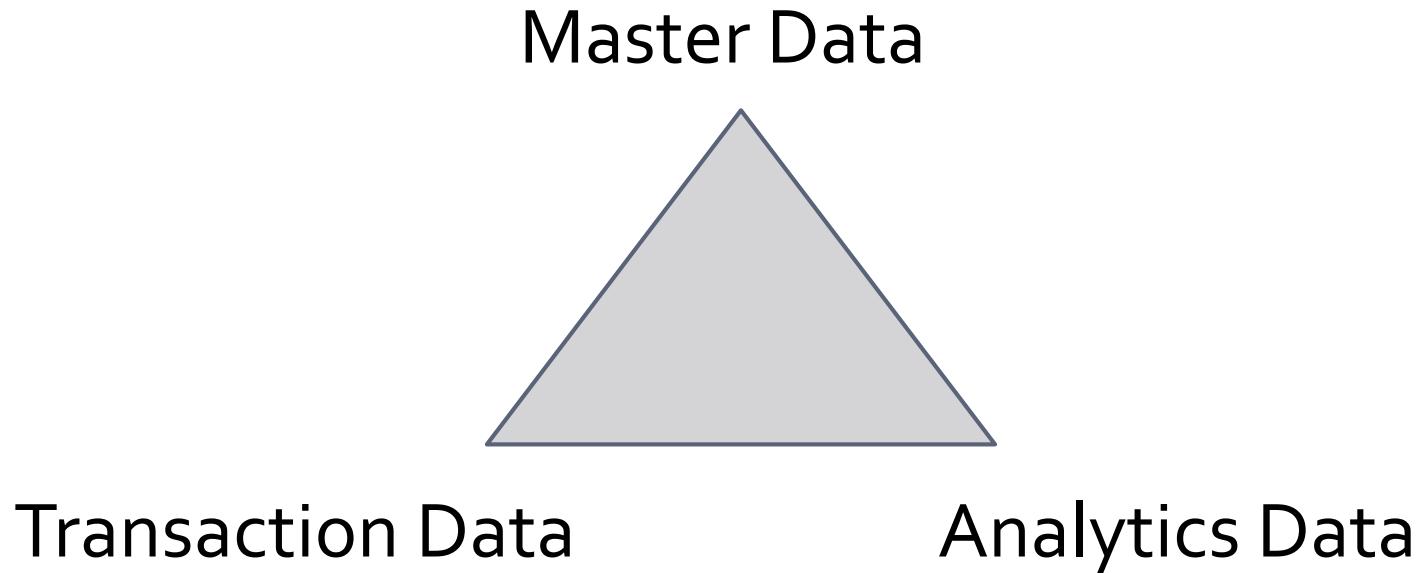
### Example query:

```
UPDATE Employees
SET Salary='100,000'
WHERE EmployeeName='Alfred Nobel';
SELECT Salary FROM Employees
WHERE Salary ='Alfred Nobel';
```

### Example query:

```
select o.customerid, o.orderid,
       o.orderdate, p.price, sum
          (p.price) over (partition by
                           o.customerid order by o.orderdate)
                           running_total
```

# The Data

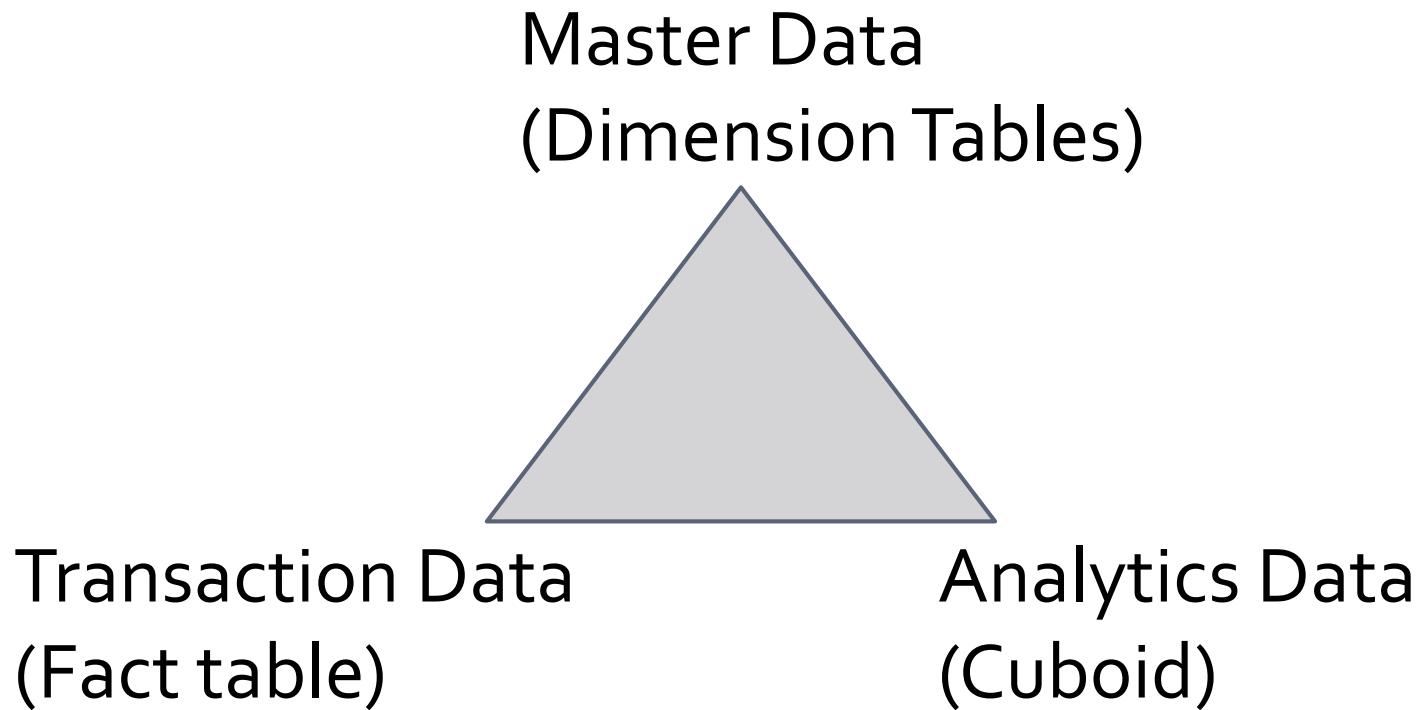


**Master Data:** The source of truth in your system, cannot withstand corruption

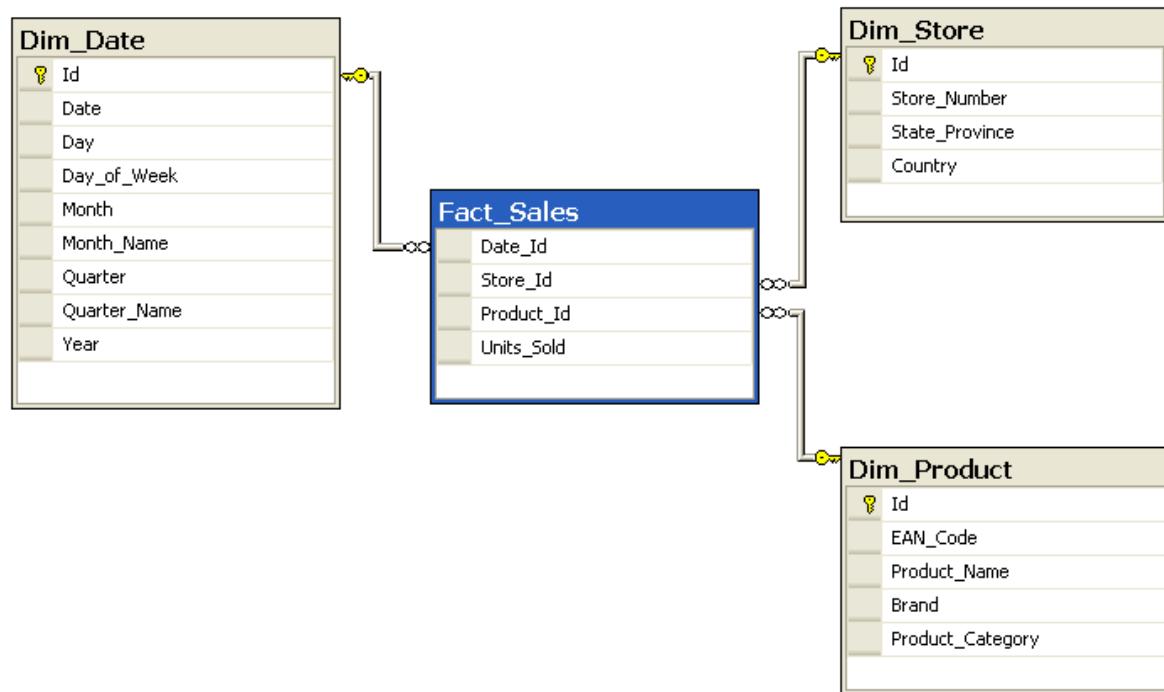
**Transaction Data:** Keeps relationships (e.g., who bought what)

**Analytics Data:** Aggregated/processed transaction data

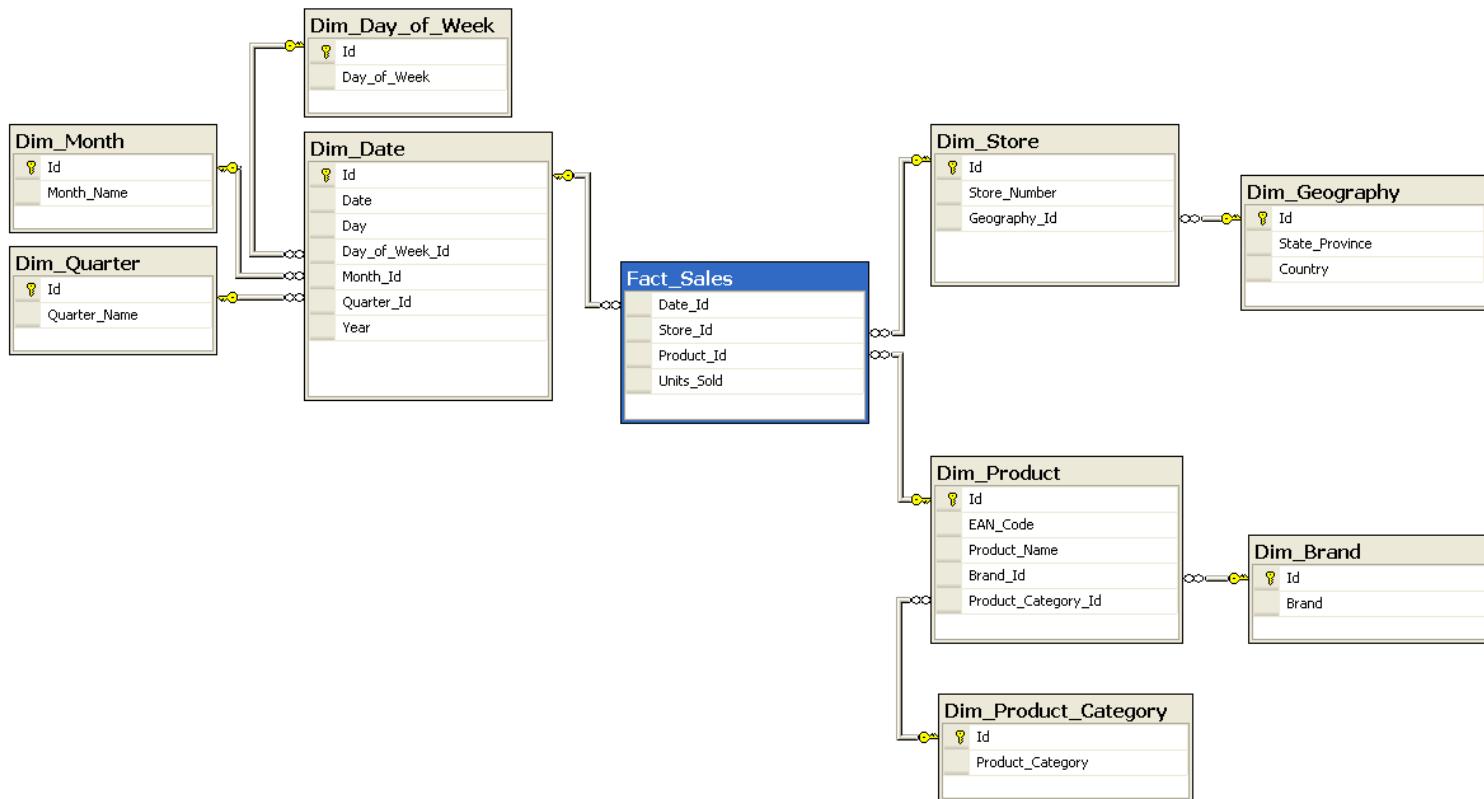
# Data Warehouse Model



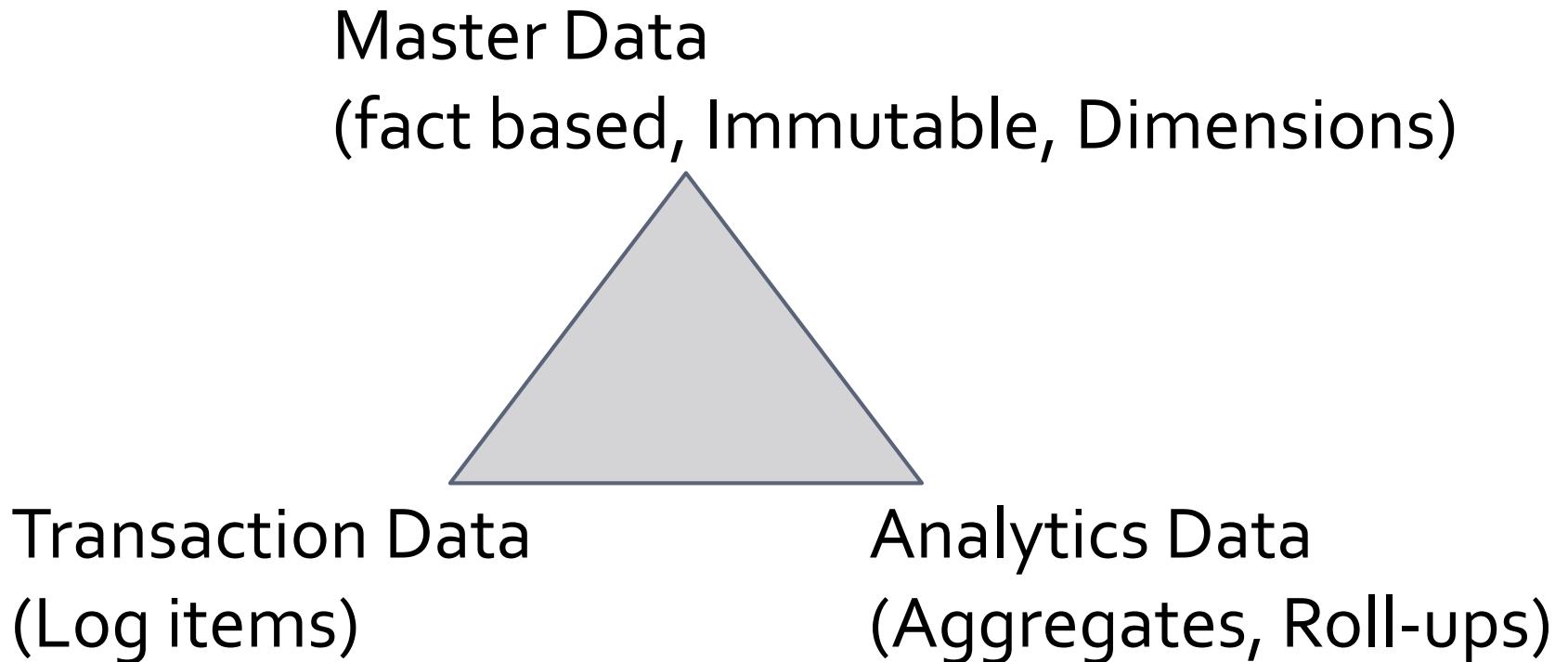
# Star Schema



# Snowflake Schema



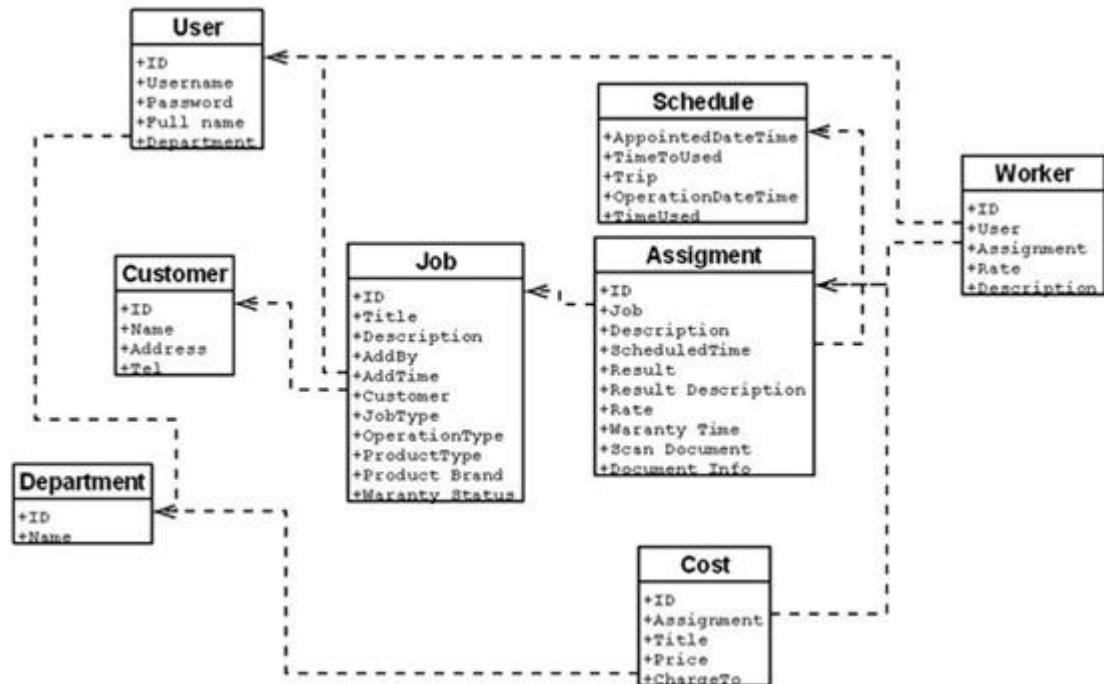
# Big Data Architecture Model



# Master Data (Dimensions)

Captures concepts and relationships that participate in transactions

*object, relationship, object attributes*



# Transaction Data (Fact Table)

Each Record Captures a Transaction  
*(id, object, amount, time, ...)*

Complaint ID	Product	Sub-product	Issue	Sub-issue	State	ZIP code	Submitted via	Date received	Date sent to con	Company	Company resp	Timely response
1354490	Debt collection		Cont'd attempts collect debt not owed	Debt is not mine	OH	44077	Web	04/30/2015	04/30/2015	Expert Global Solutions, Inc.	In progress	Yes
1355160	Student loan	Non-federal student loan	Dealing with my lender or servicer		NJ	8807	Web	04/30/2015	04/30/2015	Transwirld Systems Inc.	In progress	Yes
1356730	Credit reporting		Incorrect information on credit report	Account status	IL	60618	Web	04/30/2015	04/30/2015	FNIS (Fidelity National Information Services, Inc.)	Closed with explanation	Yes
1356607	Debt collection	Other (phone, health club, etc.)	Disclosure verification of debt	Right to dispute notice not received	WA	98133	Web	04/30/2015	04/30/2015	Stellar Recovery Inc.	Closed with explanation	Yes
1354249	Bank account or service	Checking account	Problems caused by my funds being low		AL	35127	Web	04/30/2015	04/30/2015	Wells Fargo	Closed with explanation	Yes
1354326	Bank account or service	Checking account	Account opening, closing, or management		TX	78575	Web	04/30/2015	04/30/2015	Ally Financial Inc.	In progress	Yes
1351925	Bank account or service	Checking account	Account opening, closing, or management		FL	34677	Web	04/29/2015	04/29/2015	HSBC	Closed with explanation	Yes
1352573	Debt collection	Medical	Cont'd attempts collect debt not owed	Debt was paid	NV	89143	Web	04/29/2015	04/29/2015	Nevada Credico, Inc.	Closed with explanation	Yes
1354227	Debt collection	Medical	False statements or representation	Indicated committed crime not paying	FL	32792	Web	04/29/2015	04/30/2015	Transwirld Systems Inc.	In progress	Yes
1354200	Debt collection	Credit card	False statements or representation	Indicated committed crime not paying	AZ	85304	Web	04/29/2015	04/30/2015	Patenaude & Felix APC	Closed with explanation	Yes
1352929	Debt collection	Other (phone, health club, etc.)	Cont'd attempts collect debt not owed	Debt is not mine	NC	27534	Web	04/29/2015	04/29/2015	M&S Recovery Solutions	Closed with explanation	Yes
1354191	Bank account or service	Checking account	Problems caused by my funds being low		CA	90044	Web	04/29/2015	04/30/2015	Wells Fargo	Closed with explanation	Yes
1354115	Debt collection	Medical	Cont'd attempts collect debt not owed	Debt is not mine	TX	77449	Web	04/29/2015	04/29/2015	Commonwealth Financial Systems, Inc.	Closed with explanation	Yes
1353502	Consumer loan	Vehicle loan	Problems when you are unable to pay		TX	75287	Web	04/29/2015	04/29/2015	Ally Financial Inc.	In progress	Yes
1353732	Mortgage	Conventional fixed mortgage	Loan servicing, payments, escrow account		GA	35114	Web	04/29/2015	04/29/2015	United Security Financial Corp	Closed with non-monetary relief	Yes
1353334	Bank account or service	Checking account	Account opening, closing, or management		ID	83705	Web	04/29/2015	04/29/2015	Wells Fargo	Closed with explanation	Yes
1353247	Money transfers	Domestic (US) money transfer	Fraud or scam		AR	72712	Web	04/29/2015	04/29/2015	MoneyGram	In progress	Yes
1352727	Mortgage	Conventional adjustable mortgage (ARM)	Loan modification, collection, foreclosure		FL	32808	Web	04/29/2015	04/29/2015	Bayview Loan Servicing, LLC	In progress	Yes

# Analytics Data

Roll-ups, drill downs, summaries  
*summary, average, grouping*

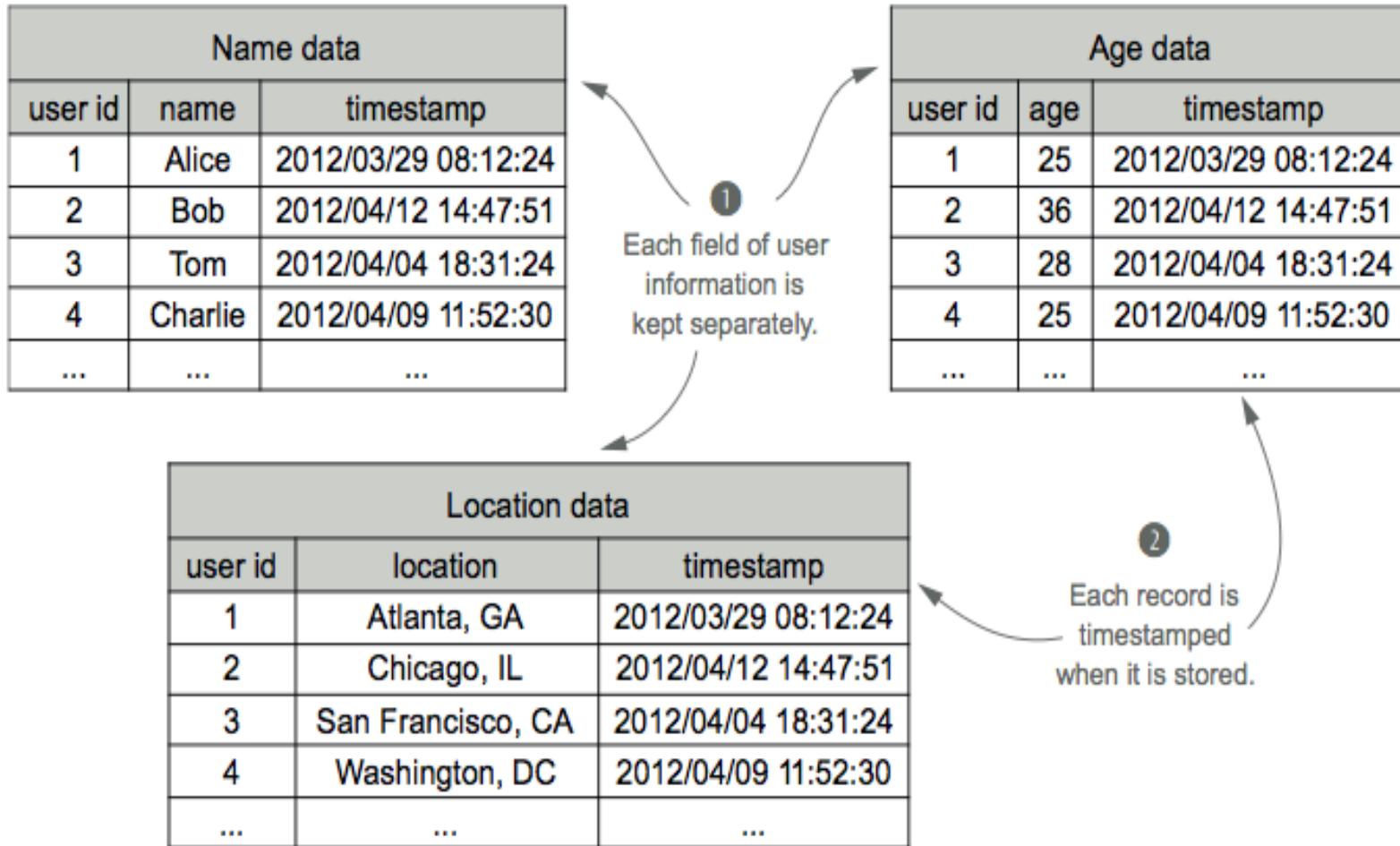
	Total Family Households, All Races	Total	Age of Householder														Mean age
			Under 20 years	20-24 years	25-29 years	30-34 years	35-39 years	40-44 years	45-49 years	50-54 years	55-64 years	65-74 years	75+ years				
3.	ALL FAMILIES	81,353		575	2,786	5,479	7,829	8,061	8,562	8,602	8,836	15,039	9,780	5,804		48.3	
4.	TENURE																
5.	.Own/Buying	58,426		237	787	2,181	4,062	4,955	5,914	6,470	7,087	12,814	8,742	5,177		51.3	
6.	.Rent	22,146		329	1,934	3,189	3,674	3,033	2,573	2,060	1,668	2,129		974	582	40.4	
7.	.No Cash Rent /3	781		9	65	109	93	73	75	72	81	96	64	45		44.5	
8.	SIZE OF FAMILY																
9.	.Two members	37,486		150	1,266	2,140	2,154	1,653	1,885	2,379	3,611	9,516	7,855	4,878		55.2	
10.	.Three members	17,931		181	850	1,500	2,090	1,939	2,037	2,194	2,297	3,081	1,164		597	45.2	
11.	.Four members	15,231		114	388	1,152	2,041	2,441	2,708	2,466	1,861	1,468		413	179	42.8	
12.	.Five members	6,838		76	183	456	969	1,265	1,274	1,052		694	588	200	81	42.4	
13.	.Six members	2,494		37	59	145	396	477		456	338	236	213	83	52	42.5	
14.	.Seven or more members	1,374		16	40	85	179	286		202	173	139	173	65	17	43.4	
15.	AGE OF OWN CHILDREN																
16.	.Without own children, any age	33,344		467	1,323	1,828	1,685	1,192	1,179	1,701	3,038	8,969	7,710	4,251		56	
17.	.With own children, any age	48,009		109	1,462	3,651	6,144	6,869	7,383	6,901	5,798	6,070	2,070	1,553		44.2	
18.	.Without own children under 25 years	39,873		471	1,330	1,837	1,709	1,238	1,227	1,936	3,815	11,317	9,324	5,669		57.5	
19.	.With own children under 25 years /4	41,479		104	1,456	3,642	6,120	6,823	7,336	6,665	5,021	3,722		456	135	41.6	
20.	.Without own children under 18 years	46,398		477	1,351	1,853	1,753	1,374	1,858	3,368	5,673	13,450	9,529	5,713		56.6	
21.	.With own children under 18 years	34,955		98	1,435	3,626	6,076	6,687	6,705	5,234	3,163	1,589		252	91	39.9	
22.	.Without own children under 12 years	55,801		487	1,368	1,901	2,075	2,473	3,826	5,962	7,815	14,511	9,640	5,744		54.5	
23.	.With own children under 12 years	25,551		88	1,418	3,578	5,754	5,588	4,736	2,640	1,022		528	141	60	37.2	
24.	.Without own children under 6 years	66,374		495	1,414	2,463	3,620	4,783	6,726	7,885	8,585	14,904	9,721		5,778	52	
25.	.With own children under 6 years	14,978		80	1,372	3,016	4,209	3,278	1,836		717	251	135	59	26	34.2	
26.	.Without own children under 5 years	68,534		495	1,488	2,743	4,126	5,342	7,167	8,067	8,647	14,942	9,735		5,782	51.4	
27.	.With own children under 5 years	12,818		80	1,298	2,735	3,703	2,719	1,395		534	189	97	45	22	33.7	
28.	.Without own children under 3 years	72,929		501	1,790	3,496	5,265	6,400	7,842	8,361	8,735	14,990	9,758		5,792	50.3	
29.	.With own children under 3 years	8,423		74	996	1,982	2,564	1,661		720	241	101	48	23	12	32.8	
30.	.Without own children under 1 year	78,604		537	2,347	4,797	6,953	7,605	8,396	8,552	8,818	15,023	9,772		5,804	48.9	
31.	.With own children under 1 year	2,749		39	439	682	876	456	166	50	18	15	8	-		31.5	
32.	.Without own children 3-5 years	71,884		560	2,145	3,759	5,199	5,825	7,211	8,061	8,667	14,928	9,742		5,787	50.3	
33.	.With own children 3-5 years	9,468		16	641	1,720	2,630	2,236	1,351		540	169	111	38	17	35	
34.	.Without own children 6-11 years	64,970		565	2,606	3,854	4,587	4,190	4,793	6,350	7,963	14,614	9,681		5,766	51.1	
35.	.With own children 6-11 years	16,383		10	179	1,625	3,242	3,871	3,769	2,252		873	425	99	38	39.2	
36.	.Without own children 12-17 years	64,774		566	2,756	5,271	6,265	5,031	4,646	4,777	6,256	13,807	9,640		5,759	49.6	
37.	.With own children 12-17 years	16,579		10	30	207	1,564	3,030	3,916	3,824	2,580	1,231		141	45	44.3	
38.	FAMILIES WITH OWN CHILDREN, SPECIFIC AGES/4																
39.	.No own children under 25	39,910		471	1,330	1,839	1,709	1,243	1,235	1,948	3,821	11,318	9,325	5,670		57.5	
40.	.Own children, more than one age group	17,611		13	379	1,637	3,256	3,819	3,648	2,714	1,467		587	69	23	39.7	
41.	.Own children 18-24 only	6,488		6	21	14	44	131		623	1,420	1,852	2,132		203	42	51.9
42.	.Own children 12-17 only	6,201		8	17	47	292	831	1,294	1,570	1,286		735	91	29	46.6	
43.	.Own children 6-11 only	5,046		8	41	512	876	1,021	1,204		760	333	207	62	24	40.3	
44.	.Own children 3-5 only	2,535		6	322	529	624	486		328	124	42	45	19	10	34.3	
45.	.Own children under 3 only	3,562		63	676	900	1,029	532	231	66	35	15	10	6		31.2	

# Mutable table

User information					
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
...	...	...	...	...	...

Should Tom move to  
a different city, this value  
would be overwritten.

# Fact Based, Immutable



In the fact-based model, you deconstruct the data into fundamental units called **facts**

# Benefits of a fact based model

- Supports queries about any time in history
- Enables complex queries (that a predefined schema might have have ignored)

Location data		
user id	location	timestamp
1	Atlanta, GA	2012/03/29 08:12:24
2	Chicago, IL	2012/04/12 14:47:51
3	San Francisco, CA	2012/04/04 18:31:24
4	Washington, DC	2012/04/09 11:52:30
3	Los Angeles, CA	2012/06/17 20:09:48
...	...	...

1 The initial information provided by Tom (user id 3), timestamped when he first joined FaceSpace.

2 When Tom later moves to a new location, you add an additional record timestamped by when you received the new data.

# Benefits of a fact based model

More easily correctable (remove erroneous facts)

Location data		
user id	location	timestamp
1	Atlanta, GA	2012/03/29 08:12:24
2	Chicago, IL	2012/04/12 14:47:51
3	San Francisco, CA	2012/04/04 18:31:24
4	Washington, DC	2012/04/09 11:52:30
3	Los Angeles, CA	2012/06/17 20:09:48
...	...	...

Human faults can easily be corrected by simply deleting erroneous facts. The record is automatically reset by using earlier timestamps.

# Storage Systems

# DBMS: Relational and Columnar

Two kinds of *database management systems*

## ■ Relational Databases

- Presents via Declarative Query Languages
- Organize underlying storage **row-wise**
  - Sometimes column-wise

## ■ Columnar Databases

- Presents via API and Declarative Query Languages
- Organize underlying storage **column-wise**

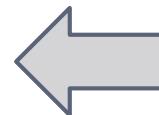
# How Do Relational DBs Work?

Logical Schema

<RowID>	EmployeeID	FirstName	LastName	Dept	Salary
001	12	John	Smith	Marketing	90
002	24	Sue	Richards	Engine	130
003	35	Maggie	Smith	DataSci	120
004	44	Bobby	Jones	DataSci	120

Physical Schema

001:12,John,Smith,Marketing,90\n  
002:24,Sue,Richards,Engine,130\n  
003:35,Maggie,Smith,DataSci,120\n  
004:44,Bobby,Jones,DataSci,120\n



- Records stored as **tables**
  - Schema validated **on-write**
  - Typically **indexed**
- Records may be persisted
  - Row-wise
  - Column-wise
- Additional structures can be applied to enhance access
  - Secondary Indices
  - Materialized Views
  - Stored Procedures

Row-Oriented

- Very fast to insert rows
- Very fast to retrieve **whole** rows
- Slower to retrieve whole columns

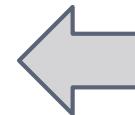
# How Do Columnar DBs Work?

Logical Schema

<RowID>	EmployeeID	FirstName	LastName	Dept	Salary
001	12	John	Smith	Marketing	90
002	24	Sue	Richards	Engine	130
003	35	Maggie	Smith	DataSci	120
004	44	Bobby	Jones	DataSci	120

Physical Schema

```
12:001;24:002;35:003;44:004\n
John:001;Sue:002;Maggie:003;Bobby:004\n
Smith:001,003;Richards:002;Jones:004\n
Marketing:001;Engine:002;DataSci:003,004\n
90:001;130:002;120:003,004\n
```



Column-Oriented

- Very fast to retrieve columns
- May save space for sparse data
- Slow to insert, slow for row reads

- Records stored as **tables**
- Largely for analytical workloads
  - Read-mostly
  - Bulk-insertion
- Additional access structures
- **Presumption**
  - More likely to read all values of a column than all values of a row
  - Optimize storage for fast column retrieval

# When are They Useful?

## Relational

- Most common data storage and retrieval system
- Many drivers, declarative language (SQL)
- Good for fast inserts
- Good for (some) fast reads
- Good for sharing data among applications
- Limited schema-on-read support
- Can be costly or difficult to scale

## Columnar

- Optimized for analytical workloads
- Maintains relational data model
- Good for analytical operations
- Good for horizontal scaling
- Bad for fast writes
- Bad for fast row-wise reads

# NoSQL and HDFS

Two distributed approaches to data storage

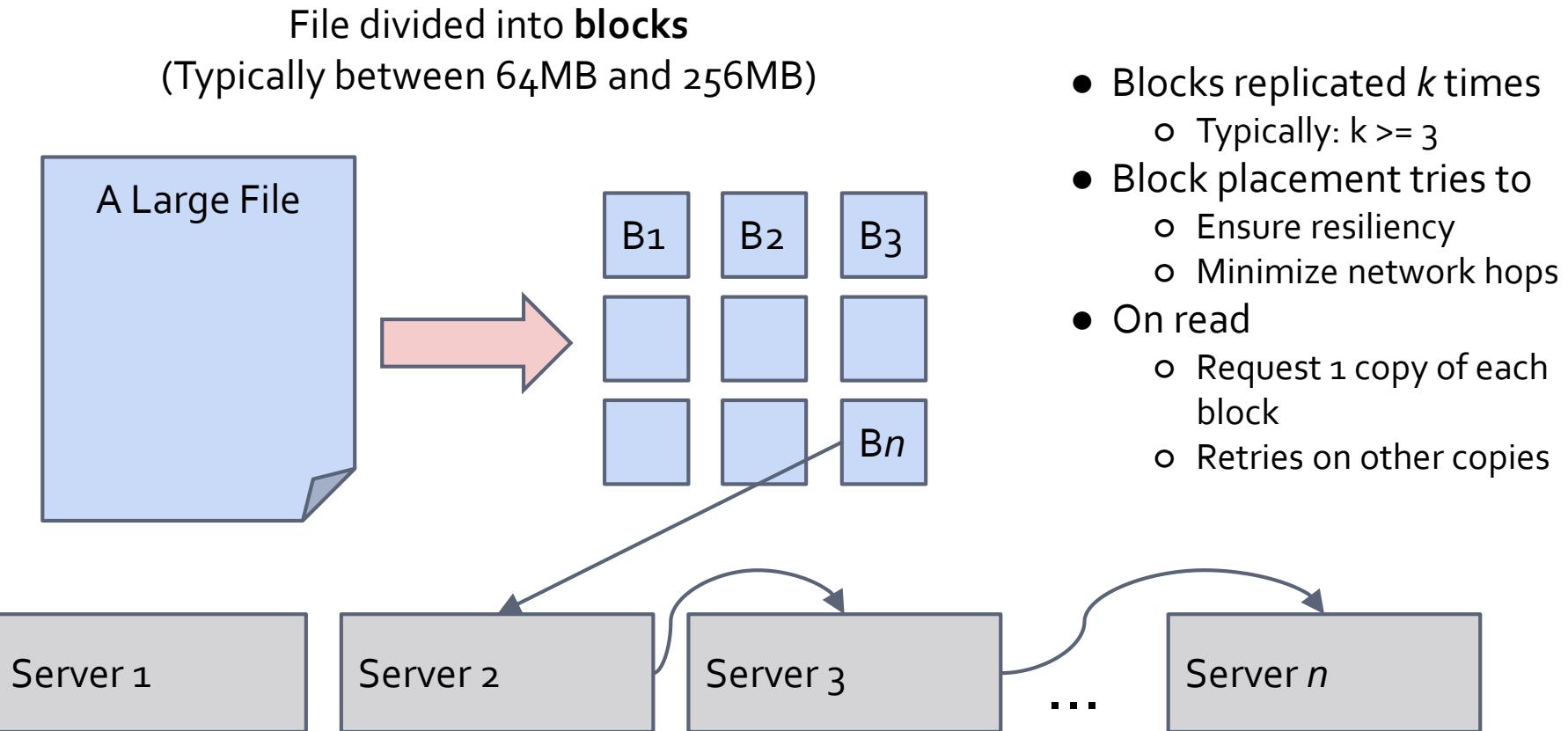
- **HDFS (Hadoop Distributed File System)**

- Presents like a local filesystem
- Distribution mechanics handled automatically

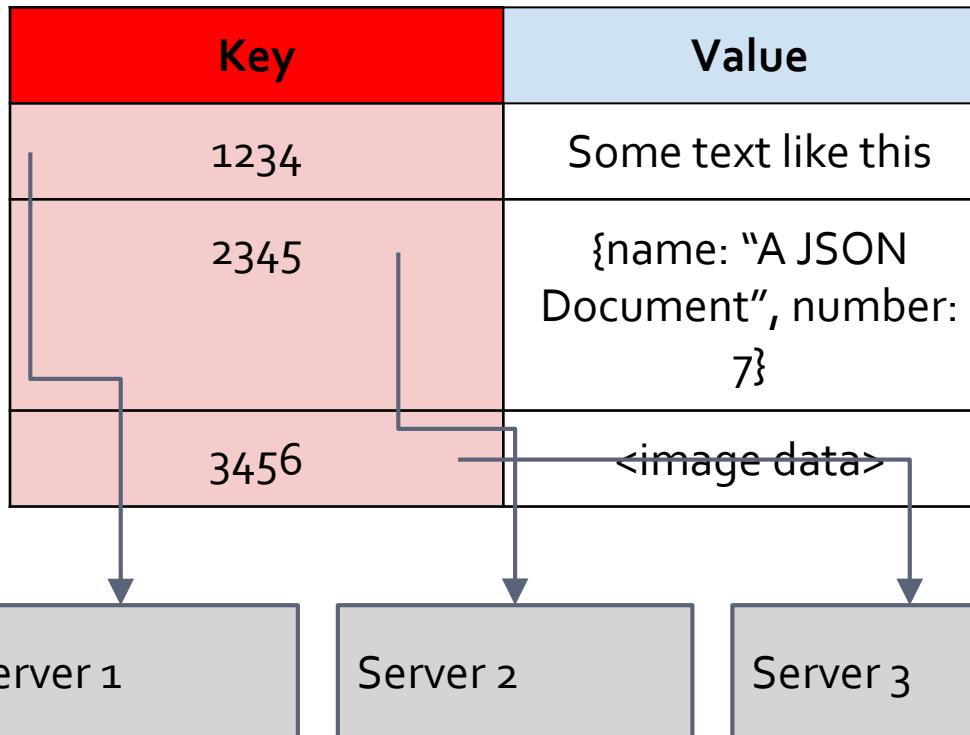
- **NoSQL Databases (Key/Value Stores)**

- Typically store records as “**key-value pairs**”
- Distribution mechanics tied to record keys

# How Does HDFS Work?



# How Do Key-Value Stores Work?



- Records stored by **key**
  - Provides a simple index
- Record placement
  - Keys are used to *shard*
  - All keys in a certain range go to a certain (set of) servers
- On read
  - Driver process reads from servers based on key-ranges

# When Are They Useful?

## HDFS

- Popular bulk data store
- Many, large files
  - File size  $\geq$  Block size
- Agnostic to file content
- Good as an immutable data store
- Good for parallel reads of lots of blocks
- Bad for small, specific reads
- Bad for fast writes

## Key-Value Stores

- Many popular choices
  - Redis, Berkeley DB
  - MongoDB
  - Cassandra (column-families imposed on value)
- Good for fast, key-based access
- Good for fast writes
- Bad for off-key access
- Complicated for merging datasets

# OS and SDS Storage

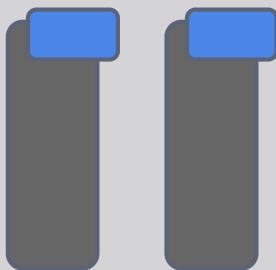
## Two Concepts

- **Object Storage (OS)**: as a new abstraction for storing data
- **Software Defined Storage (SDS)**: An architecture that enables cost effective, scalable, highly available (HA) storage systems

**Combining OS and SDS provides an efficient solution for certain data applications**

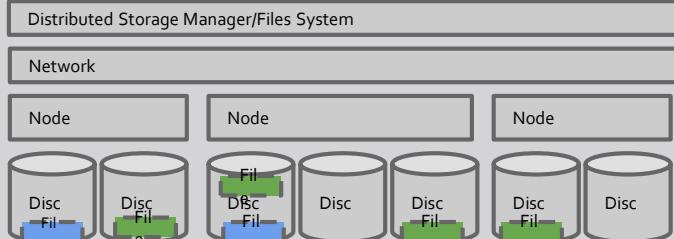
# What is it?

## Object Storage



Manage data as one (large) logical object  
Consists of meta data and data  
Unique identifier across system  
Data is an uninterpreted set of bytes

## Software Defined Storage



Data Replication for resilience and HA  
Store anything  
Runs on commodity hardware  
Manages data automatically, blocks hidden

# What Problem Does it Solve?

- Simplified model for managing data growth
- Lowers management cost
- Lowers hardware cost
- Meet Resiliency and HA needs and lower cost

# How does it work?

- Simple API with an Object Abstraction
- Data is partitioned, partitions are replicated for resiliency and HA
- Software layer
  - manages replication and data placement
  - automatically re-distributes data in case of expansion, contraction or failure

# Examples

Oracle Storage Service

OpenStack SWIFT based Storage.  
HA, scalable storage with eventual consistency

Amazon S3

HA, scalable storage with eventual consistency

Google Cloud Storage

HA, scalable storage with strong consistency

Windows Azur Storage

HA, scalable storage with strong consistency

Rackspace Files

OpenStack SWIFT based Storage.  
HA, scalable storage with eventual consistency

**Strong consistency** offers **up-to-date** data but at the cost of **high latency**

**Eventual consistency** offers **low latency** but may reply to read requests with **stale data** since all nodes of the database may not have the updated data.

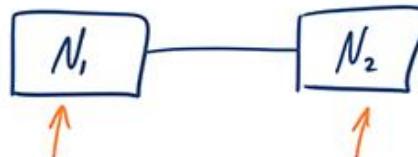
# CAP Theorem

**It is impossible for a distributed data store to simultaneously provide more than two out of the following three guarantees**

**Consistency**



**Availability**



**Partition Tolerance**

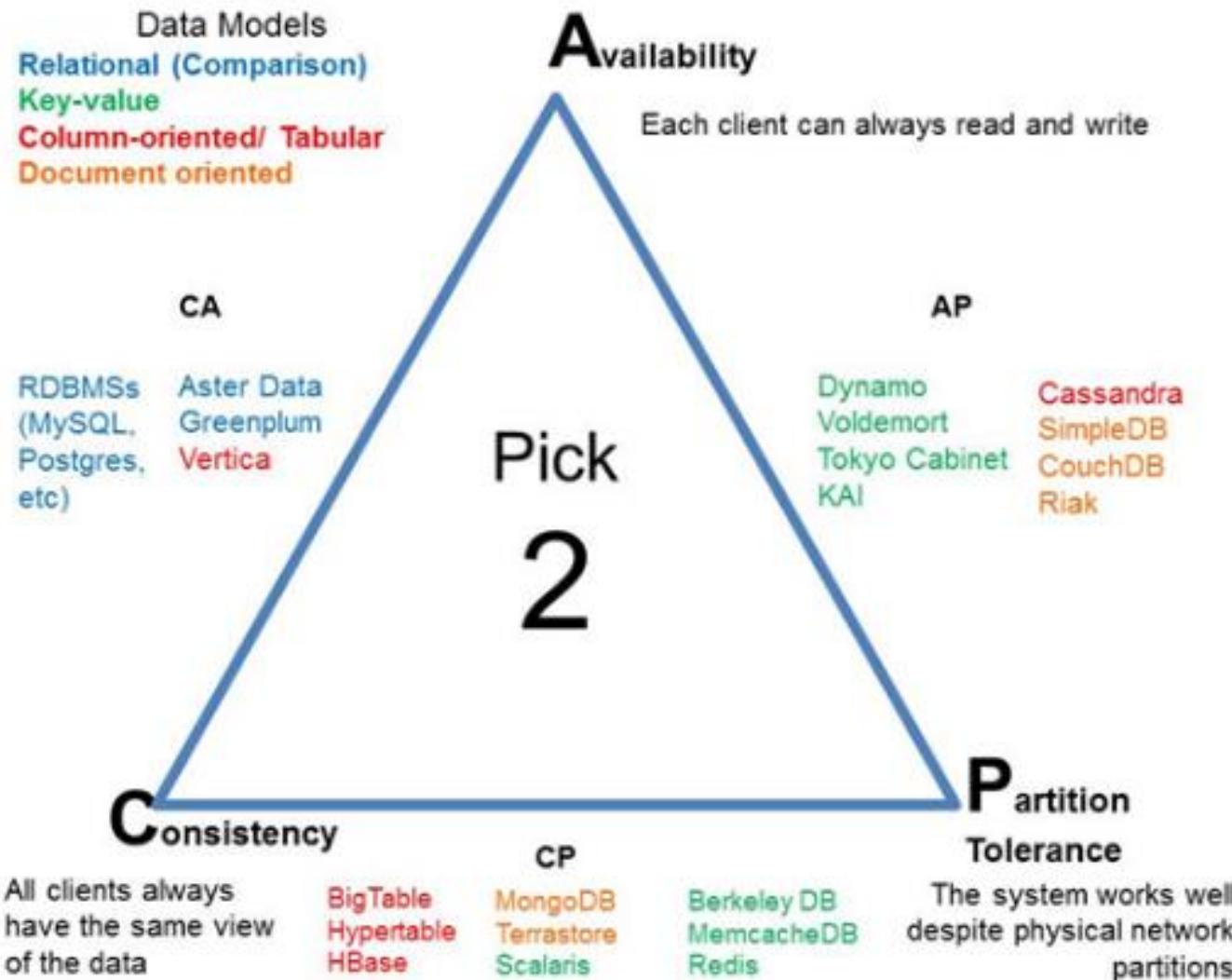


**Consistency:** Every read receives the most recent write or an error

**Availability:** Every request receives a (non-error) response – without guarantee that it contains the most recent write

**Partition tolerance:** The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes

# CAP Theorem (Simplified)



# Moving Large Data Considerations

# Moving Large Data

- If data is distributed how can you leverage parallelism?
- What kind of source and sink is involved?
- How do you use network bandwidth efficiently?
- How to handle different formats and structures?
- Large files take a long-time, how are failures handled?

As a data scientist you need to understand how to think about data transfer and movement

# Performance Measures

- **Bandwidth** measured in bits/sec is the maximum transfer rate
- **Throughput** is the actual rate that information is transferred
- **Latency** the delay between the sender and the receiver, this is a function of the calls/signals travel time and intermediate processing time
- **Jitter** variation in the time of arrival at the receiver of the information
- **Error rate** # of corrupted bits as a percentage or fraction of the total sent

# Transfer Times

## Limiting Factors:

- Available Network Bandwidth
- Read/Write performance
- Error rates

Some info (for Disk storage we use base 10)

- 1 byte = 8 bits
- 1000 Bytes = 1 Kilobyte
- 1000 Kilobytes = 1 Megabyte
- 1000 Megabytes = 1 Gigabyte
- 1000 Gigabytes = 1 Terabyte

## Example

How long would it take to transfer **2 TB file** over a **100 Mbit / sec** connection?

## Answer

$$2\text{TB} = 2 * 1000 * 1000 * 1000 * 8 = 1.6E+13 \text{ bit}$$

$$100 \text{ Mbit} = 100 * 1000 * 1000 = 100000000$$

$$\text{seconds} = 1.6E+13 / 100000000 = 160000$$

$$\text{hours} = 160000 / 60 / 60 = \underline{\text{44 hours}}$$

# Example Tools

Tool	What
Sqoop	RDBMS, BDW to Hadoop
distcp2	HDFS to HDFS copy
Rsync	FS to FS copy, FS to FS synchronization

# Tools Comparison

	Sqoop	rsync	Distcp
Type(s) of source and sink	RDBMS to HDFS	A unix / linux system inter transfer tool	HDFS to HDFS Storage Service (S3) to HDFS
Network usage	High	Smart use of bandwidth for sync	Uses available bandwidth
Level of parallelism	Low	No built-in parallelism	High. Configurable
Structures and formats	Per table or free form SQL	Agnostic to file content	Agnostic to structure in files
Resilience to failures	If task fails it will be rolled back and result in a partial export	Need to be reinitiated/restarted	High, will retry per job. If one job fails it will be restarted

# Summary: Moving Large Data

- Understand the bottleneck and your use case
- Define your time constraints, can you move all the data or do you need to segment
- Pick the right tool

SQL DDL, Avro, Protobuf, CSV

# Data Definition

# When is Schema Applied?

- Schemas represent the logical view of data
- We can apply them
  - When data is **written**
  - When data is **read**
- The application of schema comes with trade-offs

# Schema-on-Read

- Data is stored without constraint
  - Store data without validation
- Apply the plan to the data stream on every read
  - Extract the record from the data stream
  - Extract the details from the record
  - Validate the record

# Examples of Schema-on-Read

- File systems
  - The local disk on your computer
- “Big Data Frameworks”
  - Apache Hadoop
  - Apache Spark

Rely on runtime logic to apply schema
- Some NoSQL Databases
  - MongoDB
  - CouchDB

Rely on self-describing data to apply schema

# Example: Avro Schema

Consider this Avro schema expressed in JSON

```
{  
  "type": "record",  
  "name": "Person",  
  "fields": [  
    { "name": "userName", "type": "string"},  
    { "name": "favoriteNumber", "type": ["null", "long"] },  
    { "name": "interests", "type": {"type": "array", "items": "string"} }]  
}
```

# Other Schema-encod. Formats

- Google protobuf
- messagepack
- Cap'n Proto
- Parquet
  - Designed for large, column-oriented data

# Trade-Offs in Schema-on-Read

## Pros

- Data can be subject to varying interpretation
- Validation can be as strict as needed
- Schema can mutate over time

## Cons

- Higher cost to read data
  - Assemble records
  - Validate records
  - Access details
- No implicit guarantees about data contents

# Schema-on-Write

- Data is stored only if it fits constraints
  - Validate the schema before persisting
    - # and type of attributes
    - uniqueness, size, etc.
- Extract data quickly, without re-validation
  - Data shape, size, and location is already known

# Examples of Schema-on-Write

- Databases (Typically Relational or Object-Relational)
  - MySQL
  - PostgreSQL
  - Oracle Database
  - Microsoft SQL Server
  - and many more

# Trade-Offs in Schema-on-Write

## Pros

- Data characteristics are guaranteed
- Storage can be optimized to speed retrieval
  - Lower-cost reads

## Cons

- Schema must be provided **before** data is written
- Schema modifications may cause writes to existing data
- Multiple interpretations require
  - Duplicate definitions
  - Sometimes duplicate materializations

# BigTable: a Hybrid Approach

## ■ Column-Family Database

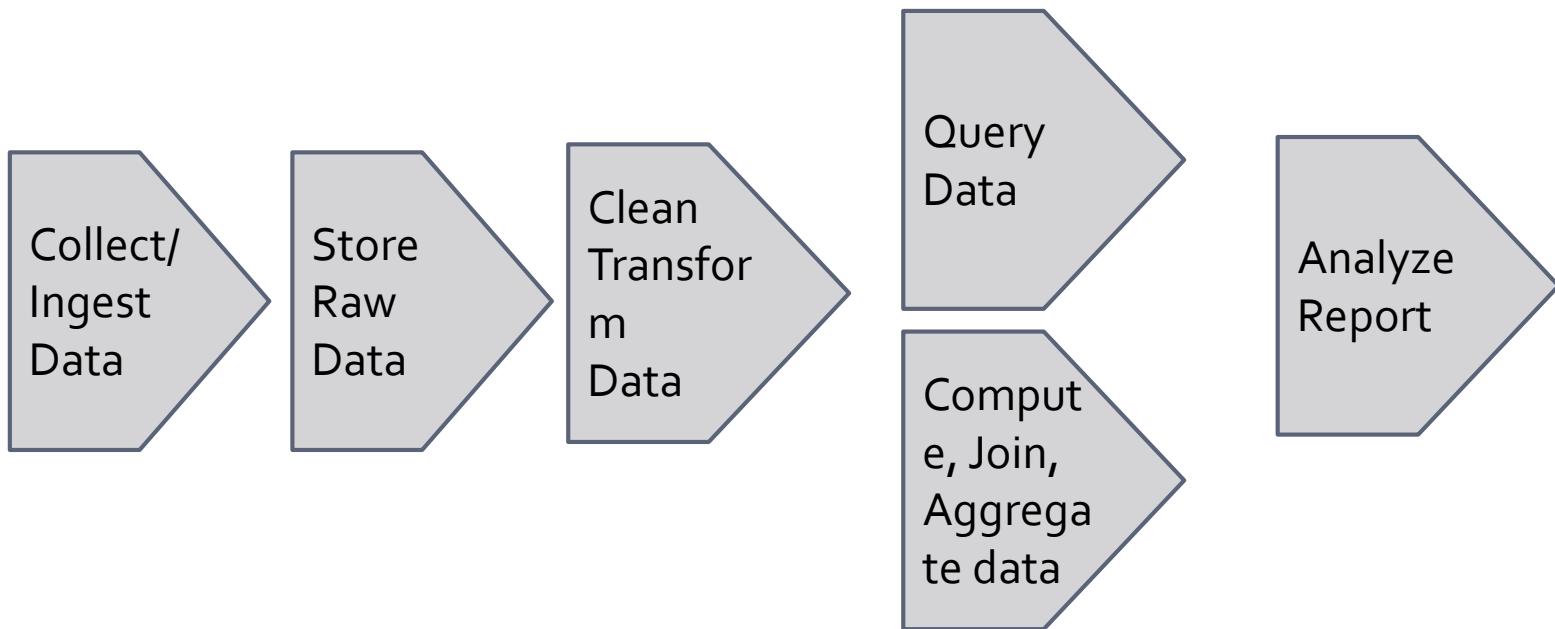
- Organize data into a hierarchy
  - Columns → record details
  - Column families → groups of columns
- Column families are **schema-on-write**
- Columns are **schema-on-read**
  - Can add columns, interpret bytes variably

## ■ Examples:

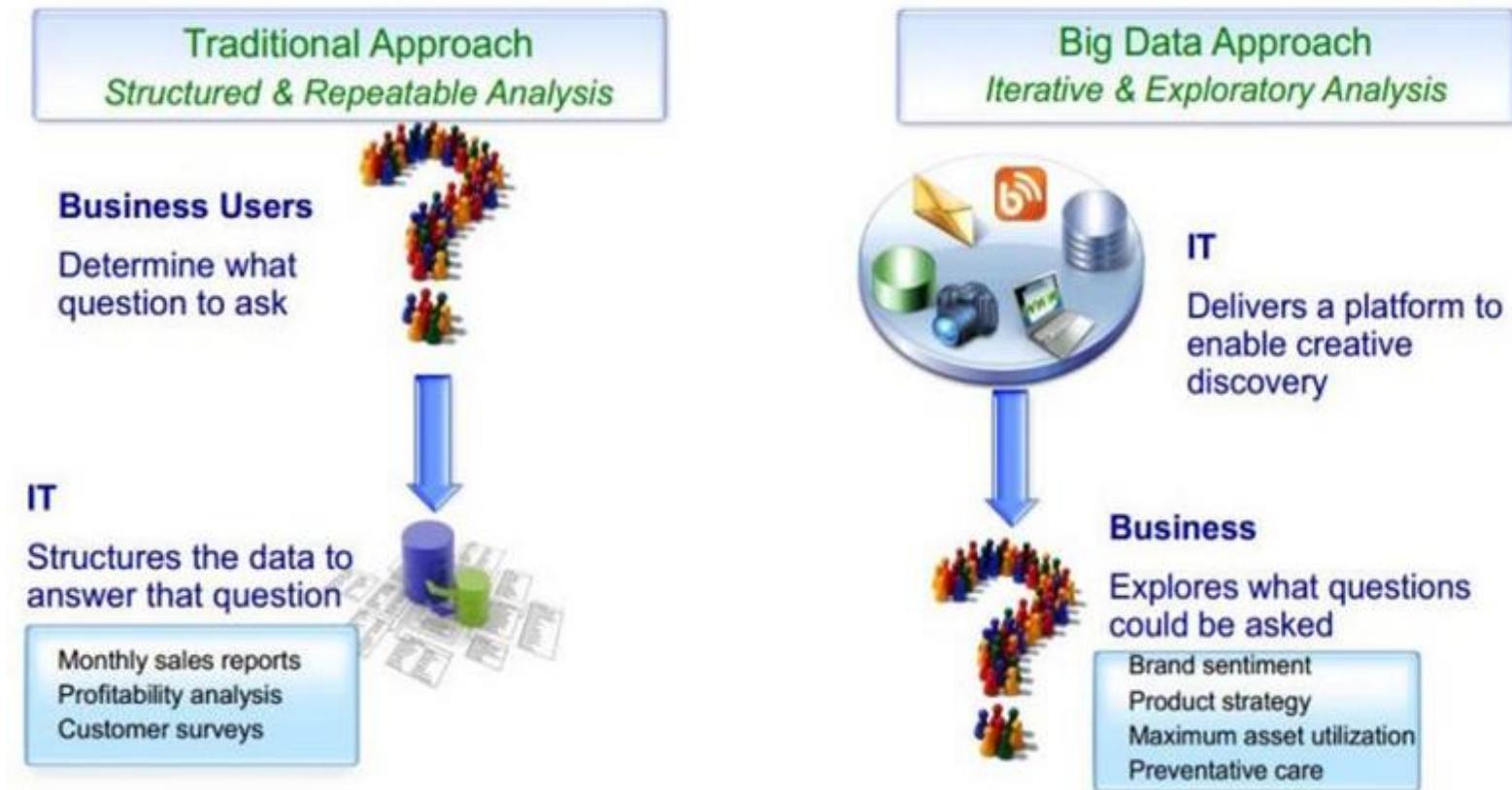
- Apache HBase
- Apache Cassandra

# **Big Data Architectures**

# Elements of DAV Architecture

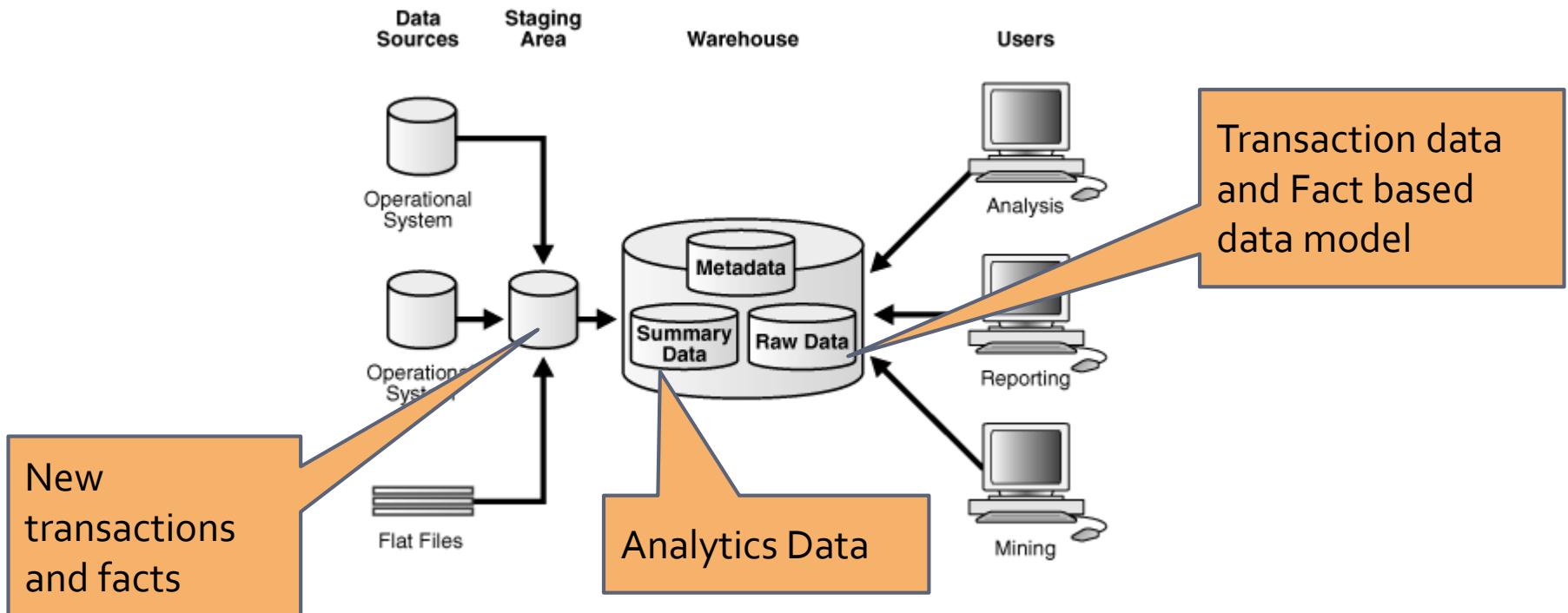


# Difference in Approach



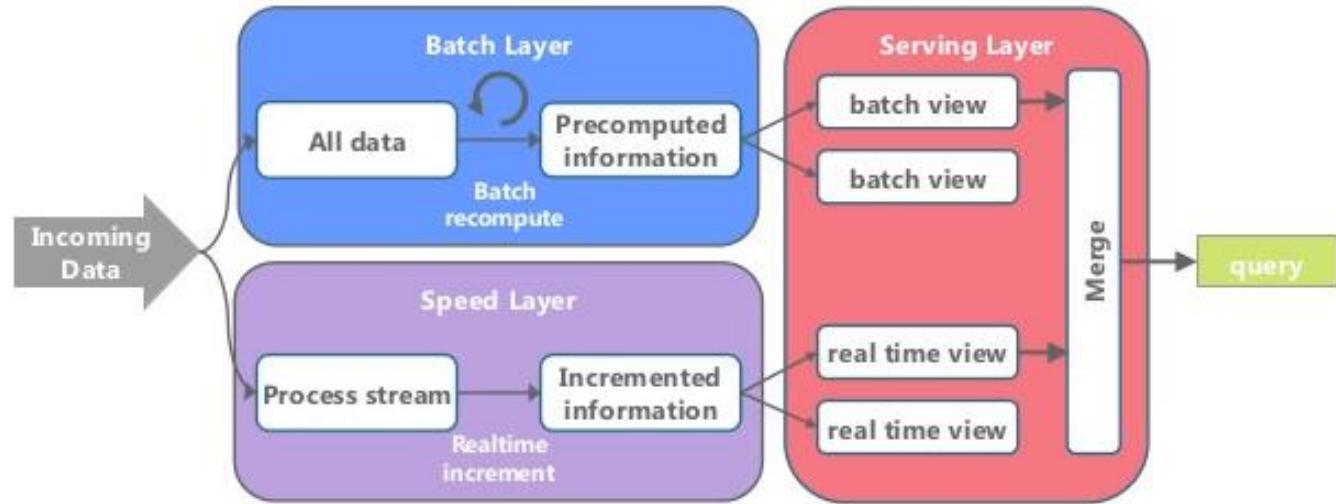
Notice the difference!

# Traditional Business Warehouse



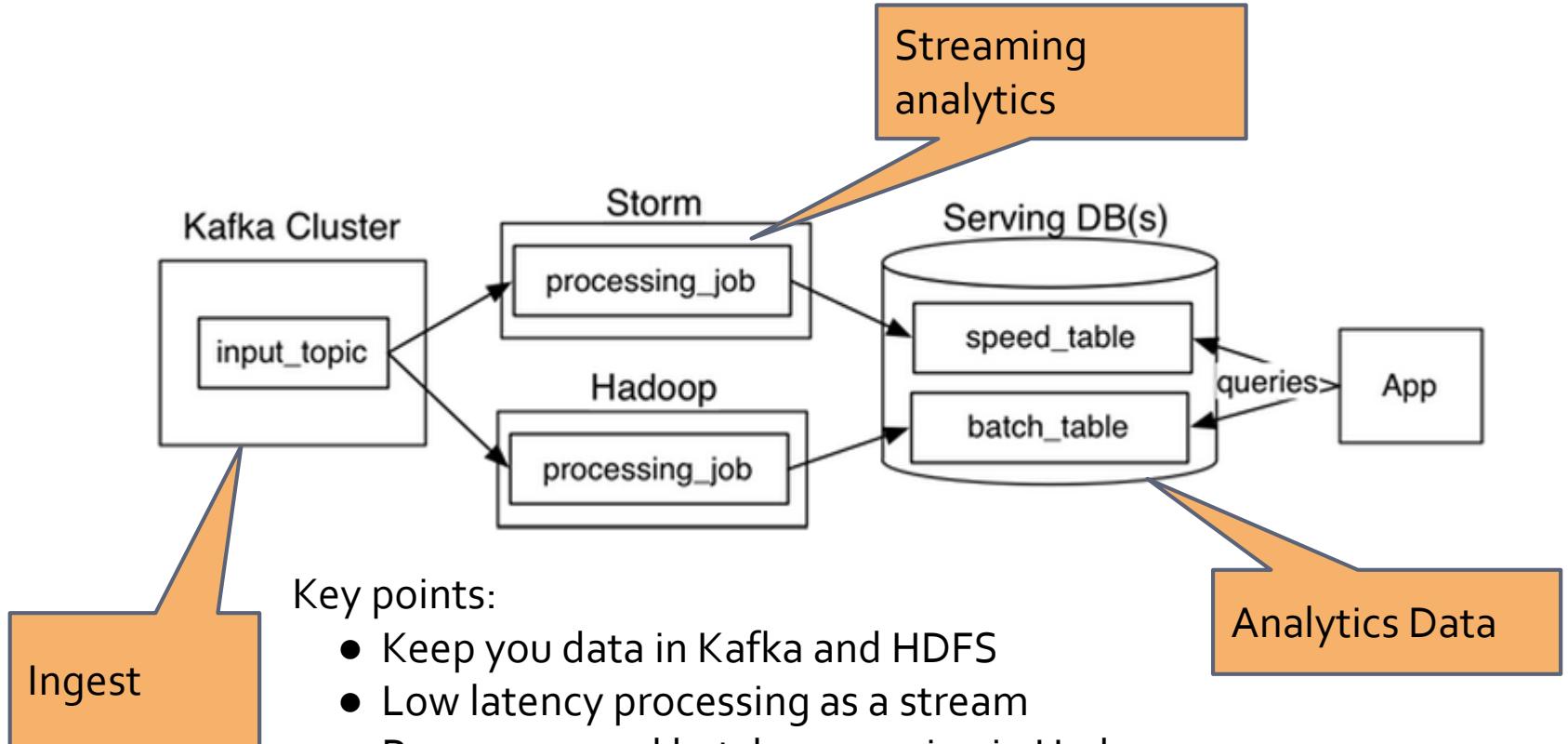
# Big Data Analytics Architecture

Example:  
Lambda Architecture

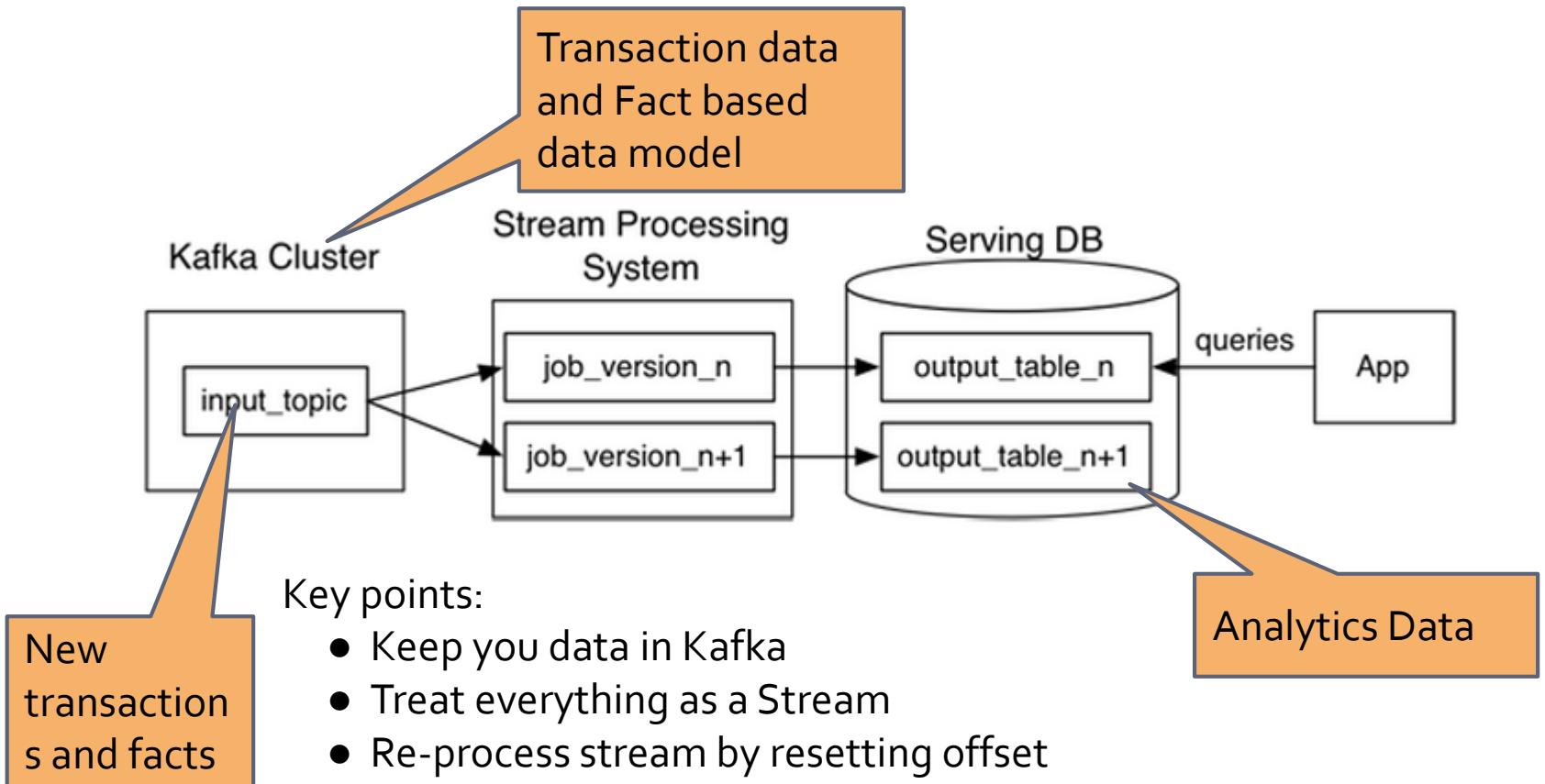


Other examples:  
Kappa Architecture  
Netflix Architecture

# Lambda Architecture



# Kappa Architecture



MapReduce, Spark, BigQuery, ...

# Processing Platforms

# Processing Platforms

- Batch Processing
  - Google GFS/MapReduce (2003)
  - Apache Hadoop HDFS/MapReduce (2004)
- SQL
  - BigQuery (based on Google Dremel, 2010)
  - Apache Hive (HiveQL) (2012)
- Streaming Data
  - Apache Storm (2011) / Twitter Huron (2015)
- Unified Engine (Streaming, SQL, Batch, ML)
  - Apache Spark (2012)