Data pipeline
batch-based data pipeline
streaming data pipeline
Lambda architecture
Kappa architecture

Data first approach
data-first paradigm vs process-first model
problem-first vs data-first approach
Data-centric strategy
Data workflows
Data safety first

Message Engine comparison
Apache pulsar vs kafka vs rabbitmq

API versioning
Media type versioning
headers versioning
URI versioning
Domain versioning
parameter versioning
Date versioning

Architectural Approaches for AB testing
Canary releases
Split URLs
Server-side
Client-side

Multi-tenancy
Standalone single-tenant app with single-tenant database
Multi-tenant app with database-per-tenant
Multi-tenant app with multi-tenant databases
Multi-tenant app with a single multi-tenant database
Multi-tenant app with sharded multi-tenant databases
Hybrid sharded multi-tenant database model

System Design

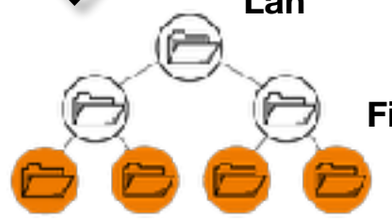
Binary Storage

Object Storage

Immutable
$
Low Perf
High Volume
0.01$/GB

- Azure Blobs
- AWS S3
- google Cloud Storage

4

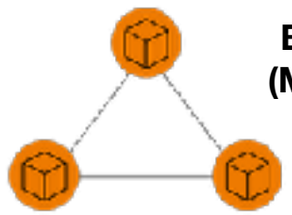


File Storage

Mutable
$$$$
Medium Perf
Medium Voulme

- Azure Files
- EFS
- Google Filestore
- NFS

3



Block Storage (Managed Disk)

Mutable
$$$
High Perf
Low Volume
1$/GB

- Azure Disks
- AWS EBS
- Google Persistent Disk

2

Unmanaged Disk

Mutable
$
High Perf

- Local Disk

1

Access across VM (Network Storage)

Single VM Storage

Rest API / SOAP protocol

Network File System (NFS) Protocol / Server Message Block (SMB) protocol

Persistent

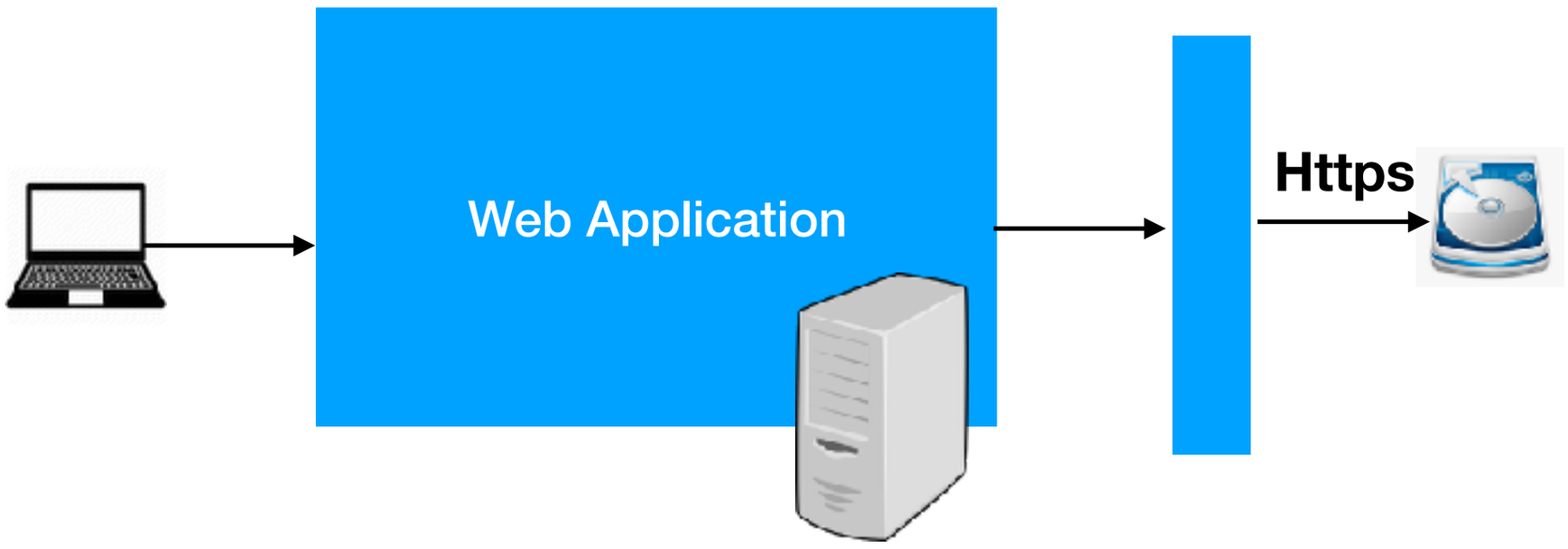
Temporary

Internet

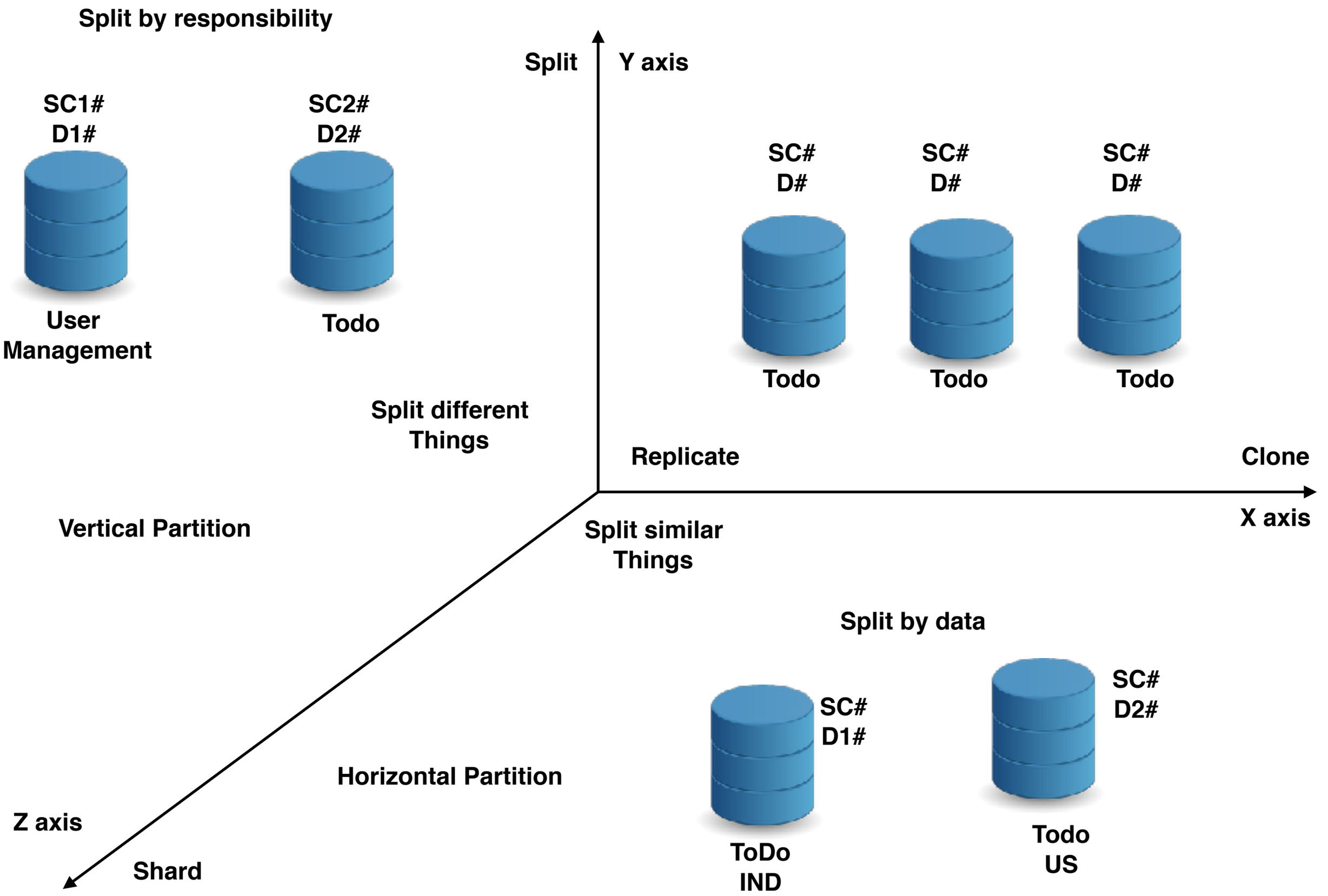
Lan

* AWS DataSync subscription is required to provide support for Server Message Block (SMB) protocol


Web Application
Https


Web Application
Https

Scalability Cube - 50 rules for high Scalability



Ecom

Split different Things

Vertical Partition

User

Inventory

Accounts

Order

Horizontal Partition

Shard

Inventory: M

Inventory : F

Order: AUS

Order: USD

Clone

Inventory: M

Inventory: M

Operational

Structured

[ACID - transaction]

RDBMS (OLTP)

- AWS RDS
- Azure SQL
- Google Cloud Sql
- Google Cloud Spanner
- **Postgres**
- **Mysql**

1

Semi Structured

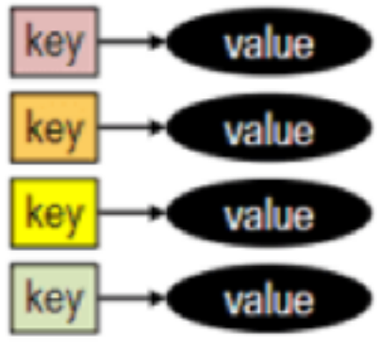
**[* BASE,
SCALE- shareded,
* Flexible Schema]**

Dictionary (key value)

Persistent | **Cache**

- AWS Elastic
- Azure Cache (redis)
- GCP memory Store
- **Redis**, Hazelcast
- Memcached, Ignite

- AWS Dynamo Db
- Azure Cosmos
- Azure Table
- Google Big Table



2

Nested Objects

Document

- AWS Document DB
- Azure Cosmos
- Firestore
- **Mongodb**, **Couchbase**,Solr

3

Entity Relationship

Graph

- AWS Neptune
- Azure Cosmos
- Google Janusgraph
- **Neo4j**
- Graph,orientdb

4

Rows vs. Documents

	Table
Row 1	Data
Row 2	Data
Row 3	Data
...	:

Document 1

```
{
  data
]
```

Document 2

```
{
  data
}
```

Document 3

```
{
  data
]
```

...

Columns vs. Properties

Col1	Col2	Col3	Col4
Data	Data	Data	Data
Data	Data	Data	Data

Document 1

```
{
  "prop1": data,
  "prop2": data,
  "prop3": data,
  "prop4": data
}
```

Document 2

```
{
  "prop1": data,
  "prop2": data,
  "prop3": data,
  "prop4": data
}
```

Document 3

```
{
```

Schema vs. Schema-Free

ID	Name	IsActive	Dob
1	John Smith	True	8/30/1964
2	Sarah Jones	False	2/18/2002
3	Adam Stark	True	7/13/1987

Document 1

```
[
  "id": "1",
  "name": "John Smith",
  "isActive": true,
  "dob": "1964-30-08"
]
```

Document 2

```
{
  "id": "2",
  "fullName": "Sarah Jones",
  "isActive": false,
  "dob": "2002-02-18"
}
```

Document 3

```
{
  "id": "3",
  "fullName":
  {
    "first": "Adam",
    "last": "Stark"
  },
  "isActive": true,
  "dob": "2015-04-19"
}
```

Normalized vs. Denormalized

User Table

UserID	Name	Dob
1	John Smith	8/30/1964

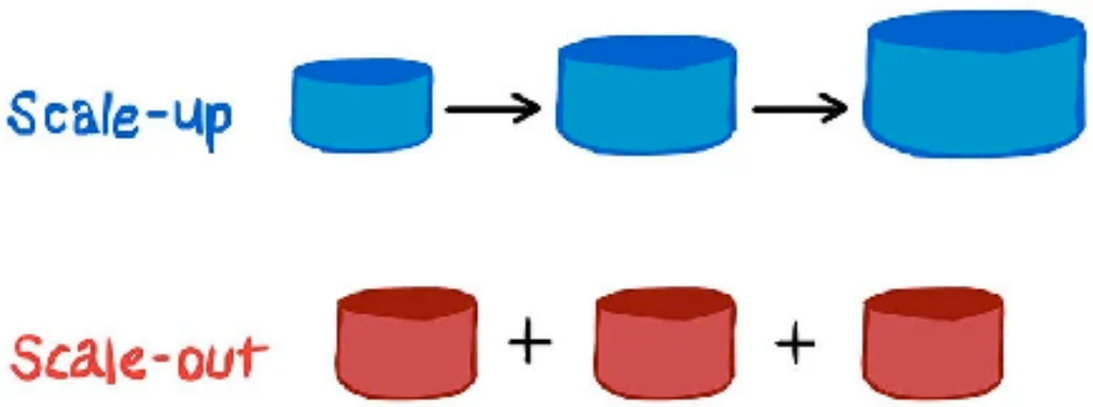
Holdings Table

StockID	UserID	Qty	Symbol
1	1	100	MSFT
2	1	75	WMT

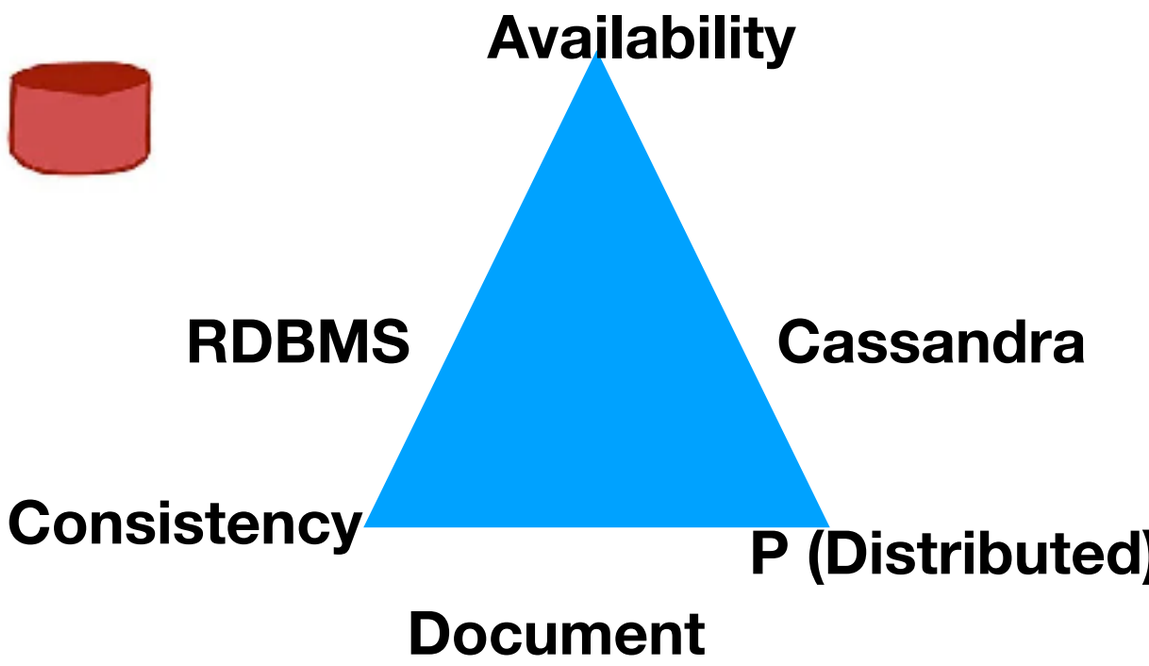
Document

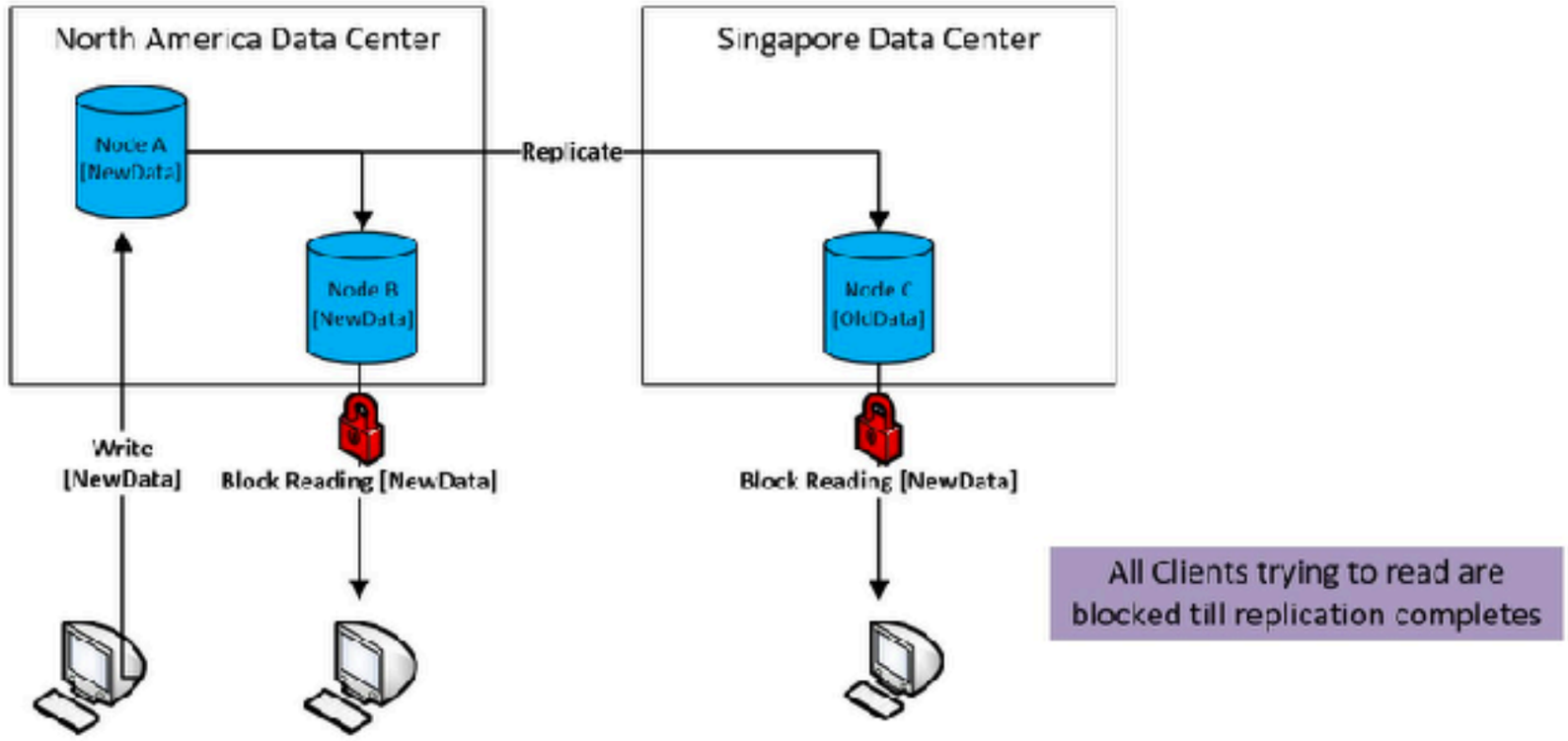
```
{
  "id": "1",
  "name": "John Smith",
  "dob": "1964-30-08",
  "holdings": [
     { "qty": 100, "symbol": "MSFT" },
     { "qty": 75, "symbol": "WMT" }
   ]
}
```

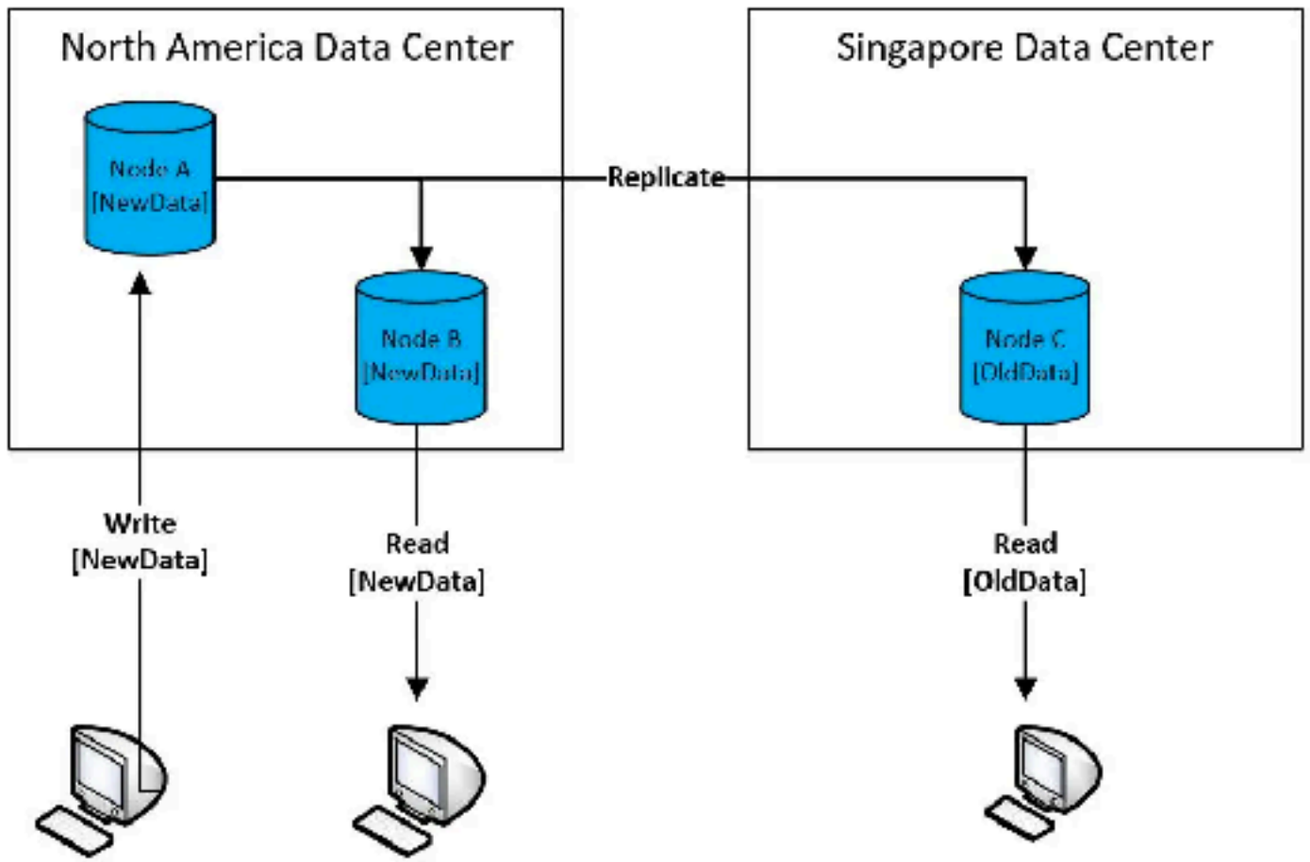
Scale-Up vs. Scale-Out



Strong Consistency vs. Eventual Consistency




North America Data Center
Singapore Data Center
Node A [NewData]
Replicate
Node B [NewData]
Node C [OldData]
Write [NewData]
Block Reading [NewData]
Block Reading [NewData]
All Clients trying to read are blocked till replication completes


North America Data Center
Singapore Data Center
Node A [NewData]
Replicate
Node B [NewData]
Node C [OldData]
Write [NewData]
Read [NewData]
Read [OldData]

COMMON COMPARISONS BETWEEN

MYSQL & NOSQL

	MySQL	NoSQL
Nature	Relational Database	Non-Relational Database
Design	Based on the concept of tables	Based on the concept of documents
Scalable	Tough to scale due to its relational nature	Easily scalable big data compared to relational
Model	Detailed database model is needed before creation	No need of a detailed database model
Community	Vast community available	Community is growing rapidly, but still smaller compared to MySQL
Standardization	SQL is standard language	Lacks standard query language
Schema	The Schema is rigid	The Schema is dynamic
Flexibility	Not very flexible in terms of design	Very flexible in terms of design
Insertions	Inserting new columns or fields affect the design	No effect on the design with the insertion of new columns or fields

Facebook, Wikipedia, Quora, Flickr	MySQL
Twitter	MySQL for tweets and users their own special kind of graph database, FlockDB, built on top of MySQL their own version of Memcached
LinkedIn	Oracle Database and Voldemort
YouTube	MySQL -> BigTable
Microsoft, Myspace	***SQL Server***
Yahoo	***PostgreSQL***

Key Value	○ Twitter uses Redis to deliver your Twitter timeline
	○ Pinterest uses Redis to store lists of users, followers, unfollowers, boards, and more
	○ Coinbase uses Redis to enforce rate limits and guarantee correctness of Bitcoin transactions
	○ Quora uses Memcached to cache results from slower, persistent databases
Graph	○ Walmart uses Neo4j to provide customers personalized, relevant product recommendations and promotions
	○ Medium uses Neo4j to build their social graph to enhance content personalization
	○ Cisco uses Neo4j to mine customer support cases to anticipate bugs
Document	○ SEGA uses MongoDB for handling 11 million in-game accounts
	○ Cisco moved its VSRM (video session and research manager) platform to Couchbase to achieve greater scalability
	○ Aer Lingus uses MongoDB with Studio 3T to handle ticketing and internal apps
	○ Built on MongoDB, The Weather Channel's iOS and Android apps deliver weather alerts to 40 million users in real-time

Analytical

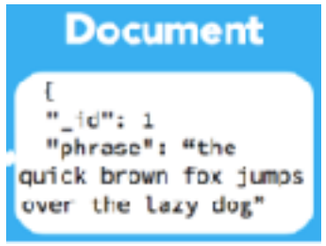
Structured

Semi Structured

Big data Db

DWH (OLAP)

- AWS Redshit
- **Azure Synapse**
- GCP BigQuery
- Snow flakes
- Kudu, Druid, Pinot
- Click house

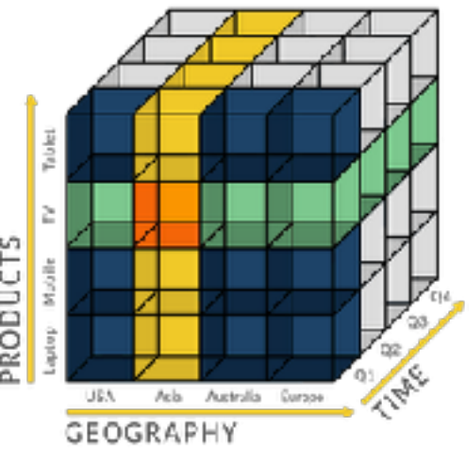


Text Database

- AWS **Elastic Search**
- AWS cloud Search
- Azure Cognitive Search
- Google Search API
- Elastic Search, Solr

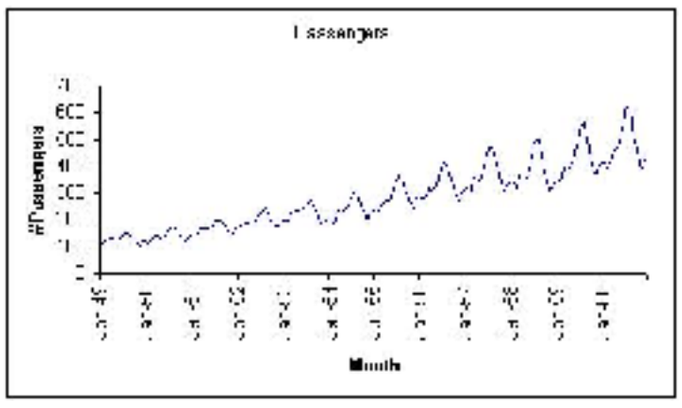
Wide Column (Column db)

- AWS Keyspaces
- Cosmos
- Big Table
- **Hbase, cassandra**
- Scylla



Time Series

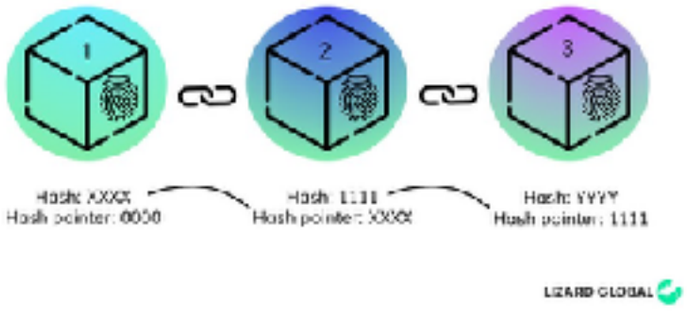
- AWS Timestream
- Cosmos
- Big Table, Big Query
- InfluxDB, Scylla
- OpenTSDB



Audit Trial

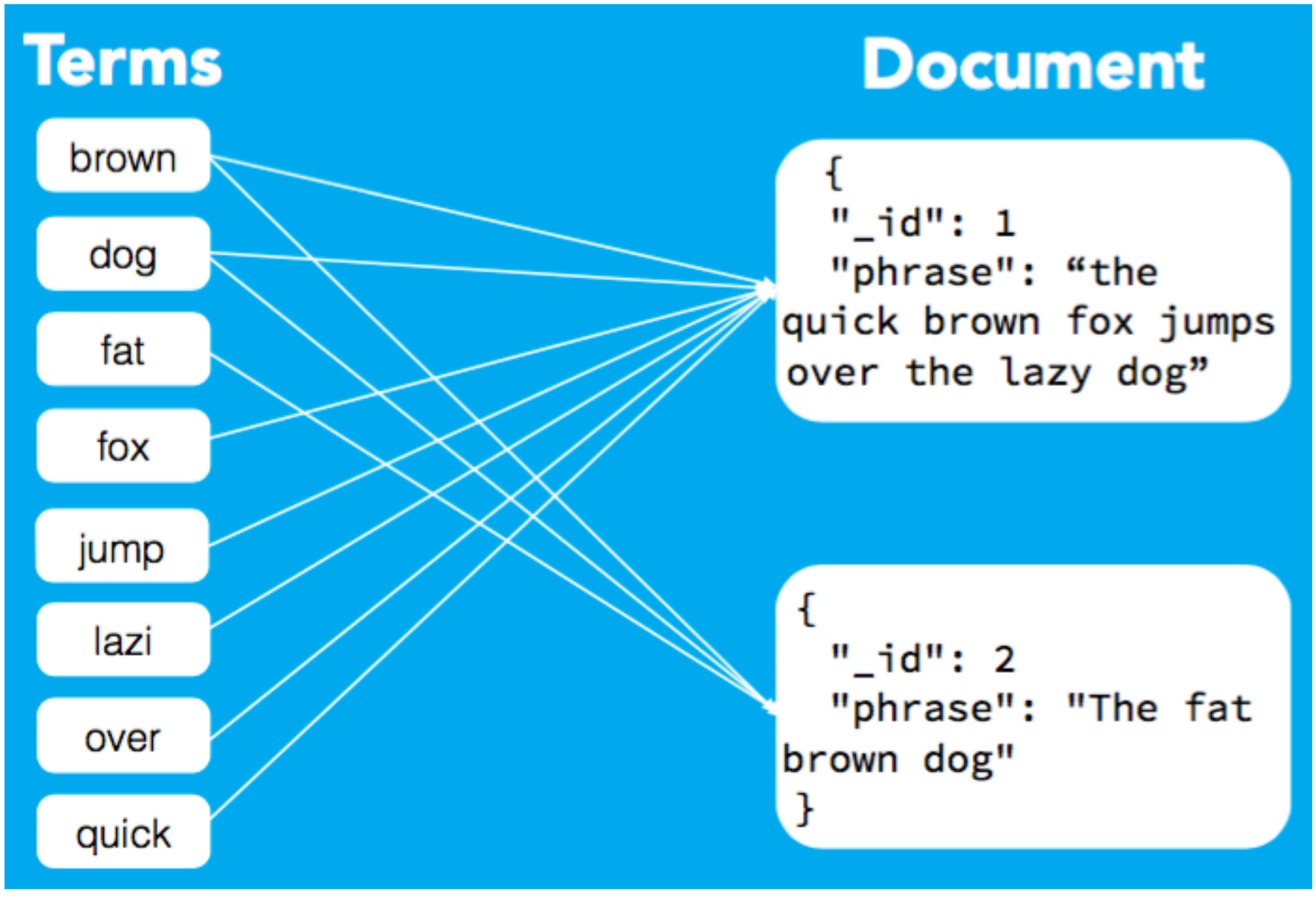
Immutable Logger

- AWS Quantum Ledger
- Azure SQL Database Ledger
- Google ?
- Hyper ledger Fabric



geo Spatial

- AWS Keyspaces
- Azure Cosmos
- Big Table, Big Query
- **Solr**, PostGIS
- Mongoldb (GeoJSON)


Terms
Document
brown
dog
fat
fox
jump
lazi
over
quick
{
"_id": 1
"phrase": “the quick brown fox jumps over the lazy dog”
{
"_id": 2
"phrase": "The fat brown dog"
}

Table

	Country	Product	Sales
Row 1	India	Chocolate	1000
Row 2	India	Ice-cream	2000
Row 3	Germany	Chocolate	4000
Row 4	US	Noodle	500

Row Store

Row 1	India
	Chocolate
	1000
Row 2	India
	Ice-cream
	2000
Row 3	Germany
	Chocolate
	4000
Row 4	US
	Noodle
	500

Column Store

Country	India
	India
	Germany
	US
Product	Chocolate
	Ice-cream
	Chocolate
	Noodle
Sales	1000
	2000
	4000
	500

Classic Relational Databases

Name	Age	Nickname	Employee
Gianarno Cuttizzoni Founder & CEO	40	Heidi	◉
Marco Botton Tuttofare	38		☑
Mariah Maclachlan Better Half	41	Potato	⊟
Valerie Liberty Head Chef	16	Val	☑

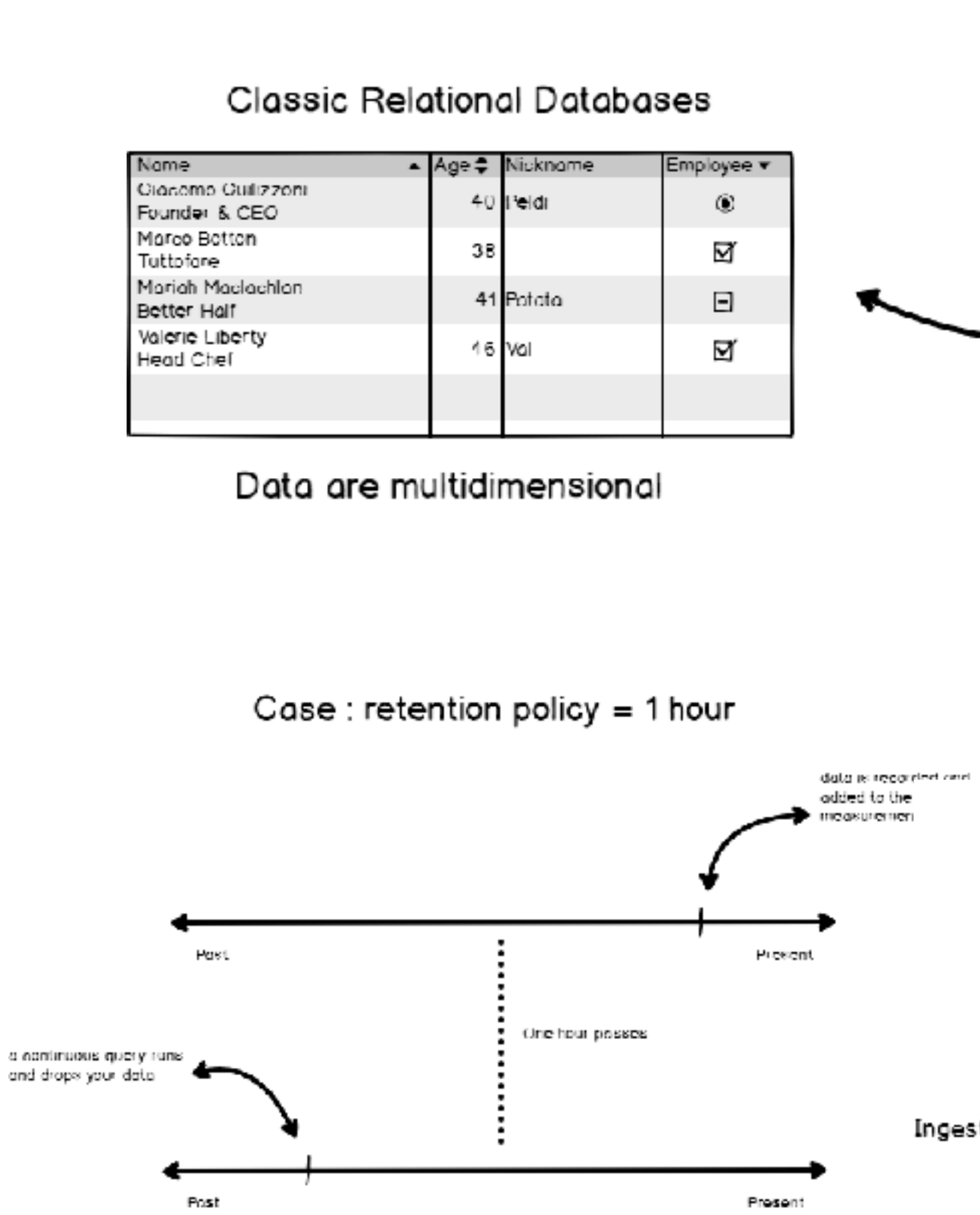
Data are multidimensional

Time Series Databases

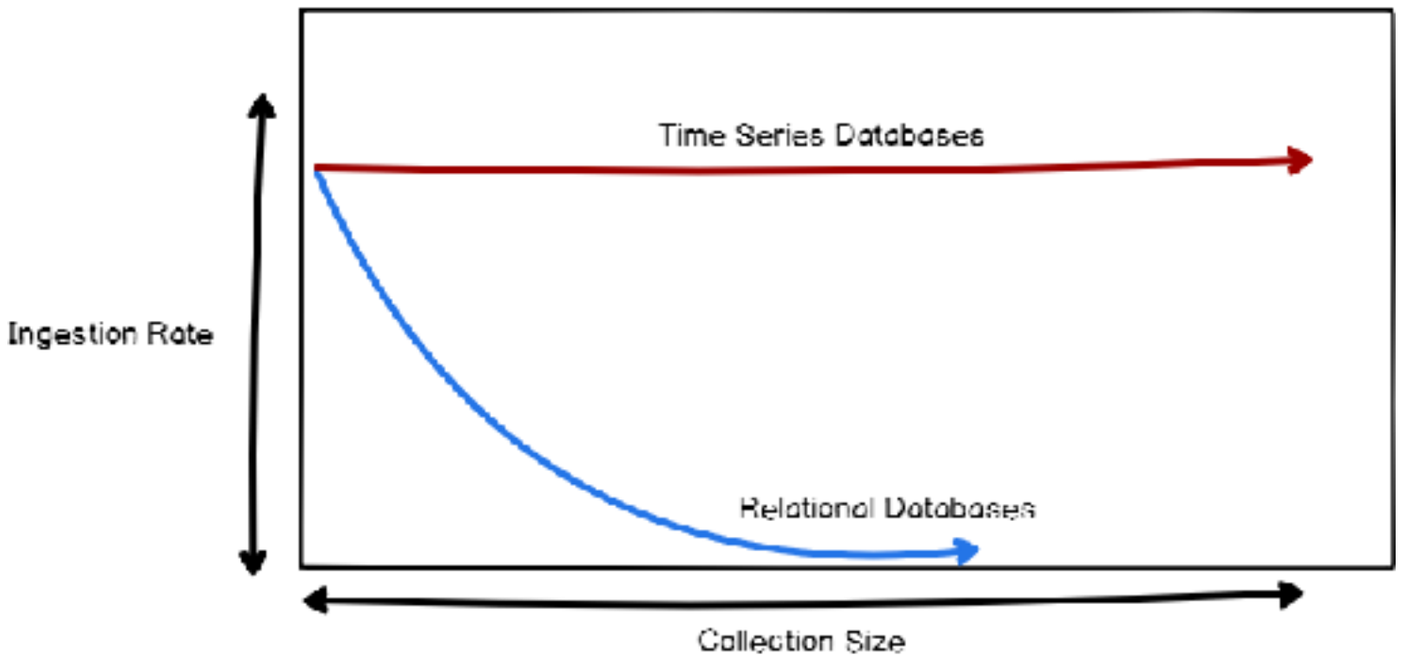
Sensor Temperature	Time
39.6	12/04/19 @ 14:12
41.2	12/04/19 @ 14:13
12.4	14/04/19 @ 12:15
18.5	16/04/19 @ 10:05

Data are aggregated over time

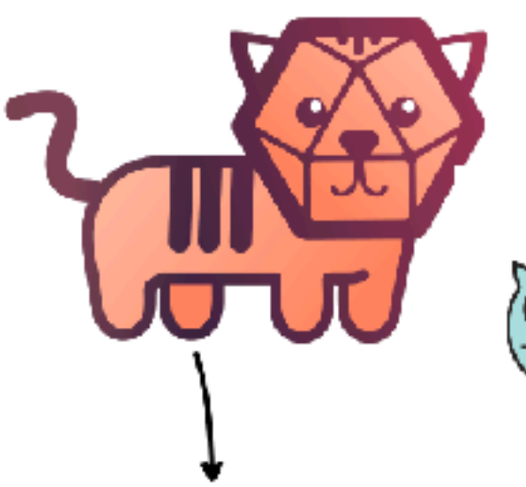
Case : retention policy = 1 hour



DBMS & TSDB Difference

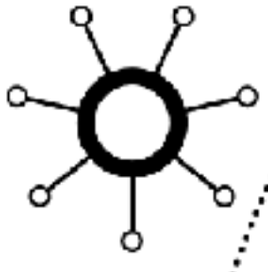


Tools that 'produce' TSDB data



monitoring done via specific query agents

collectd



StatsD

monitoring done via node exporters

Prometheus

Tools that 'consume' TSDB

chronograf

DATADOG

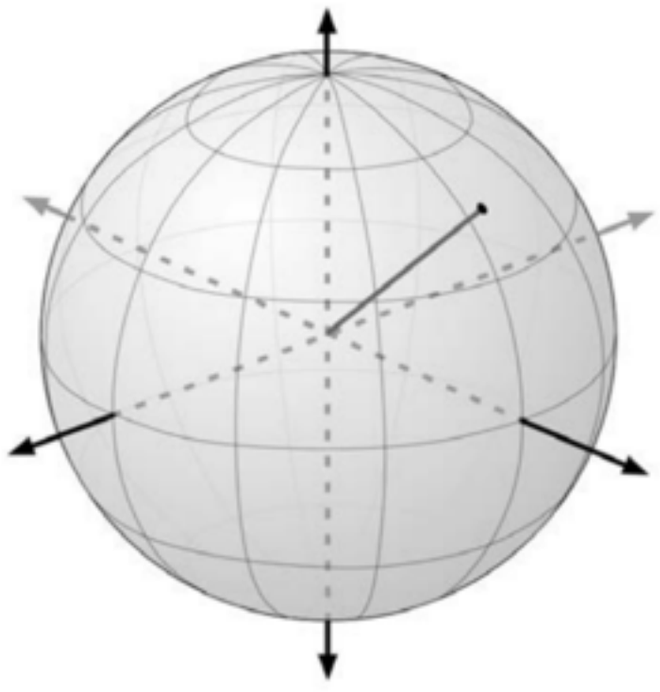
Prometheus

Grafana

kibana

*Non-exhaustive list

Car ID	Departure			Arrival		
	Date-Time	Latitude	Longitude	Date-Time	Latitude	Longitude
...	...	...	...	...	...	...
...	...	...	...	...	...	...
...	...	...	...	...	...	...



Row-oriented

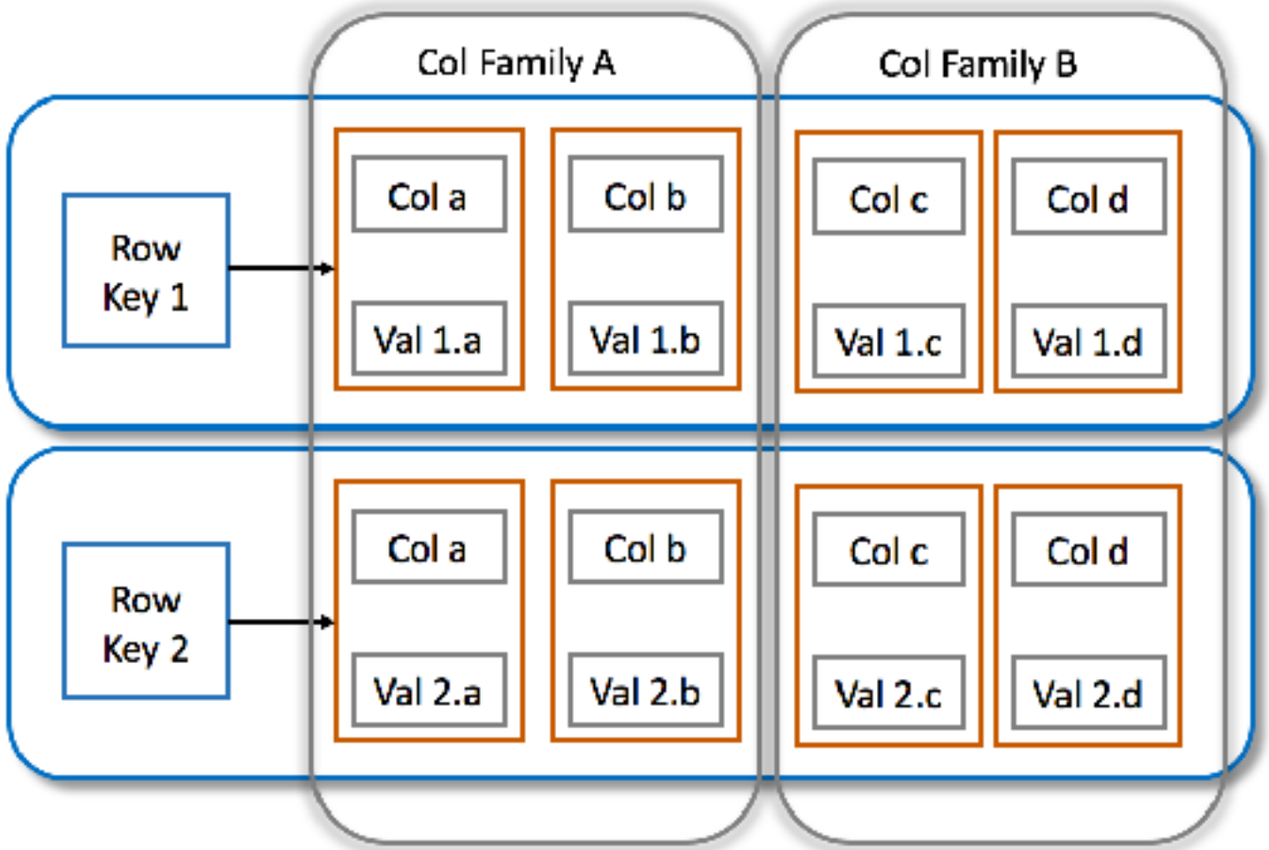
ID	Name	Grade	GPA
001	John	Senior	4.00
002	Karen	Freshman	3.67
003	Bill	Junior	3.33

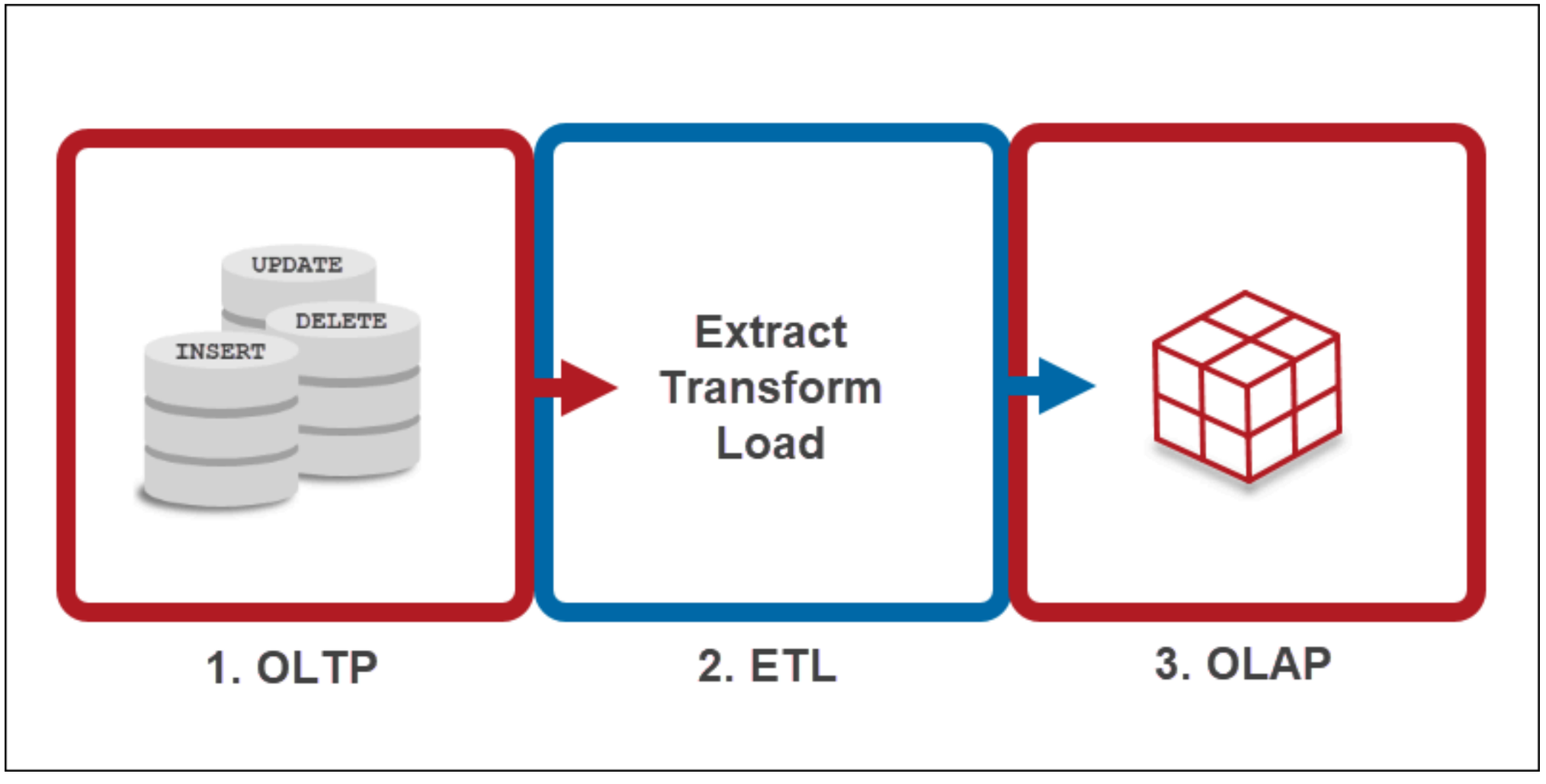
Column-oriented

Name	ID
John	001
Karen	002
Bill	003

Grade	ID
Senior	001
Freshman	002
Junior	003

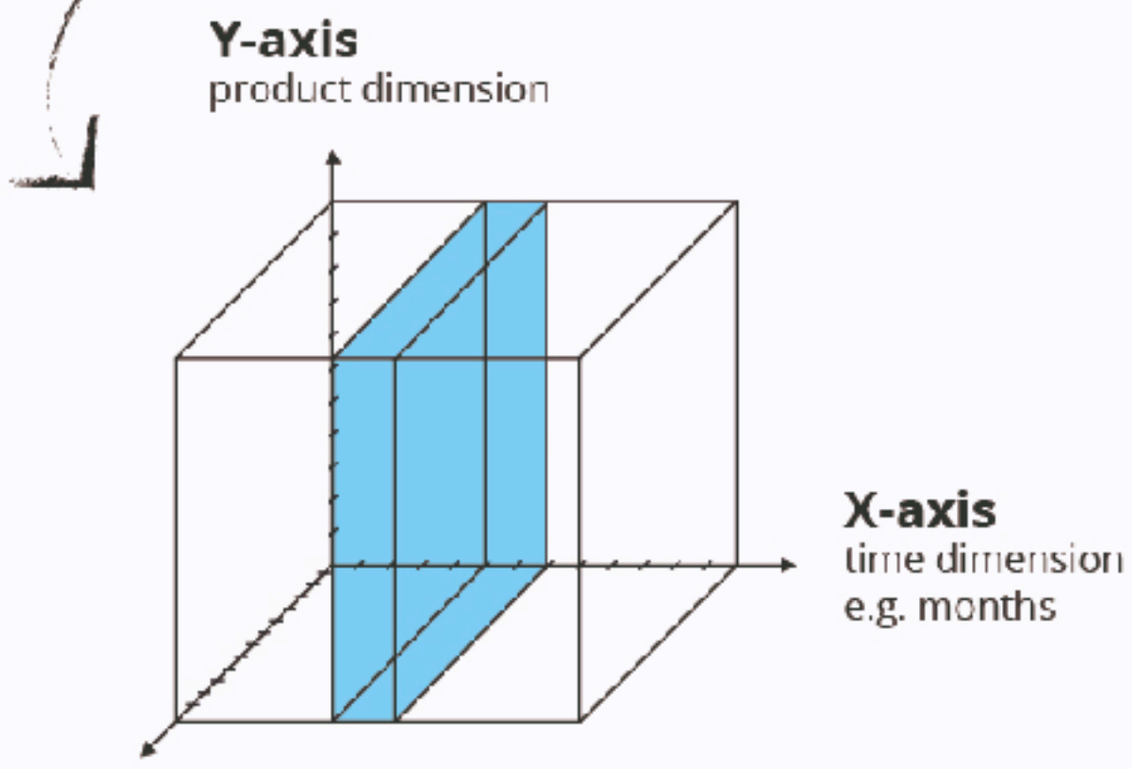
GPA	ID
4.00	001
3.67	002
3.33	003

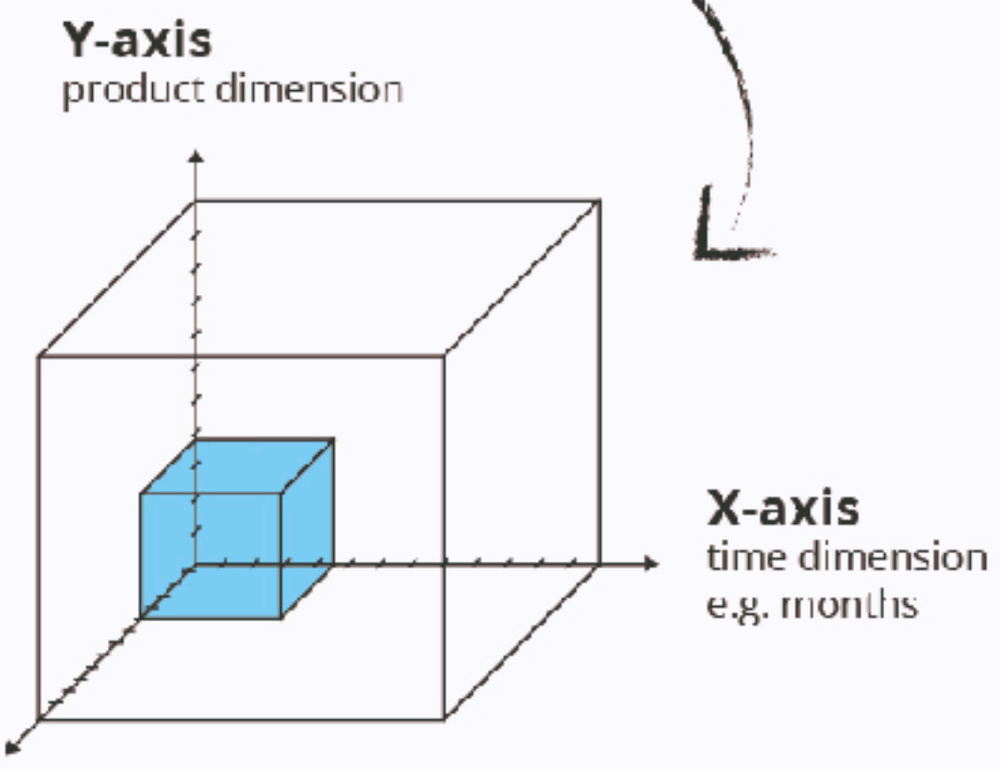


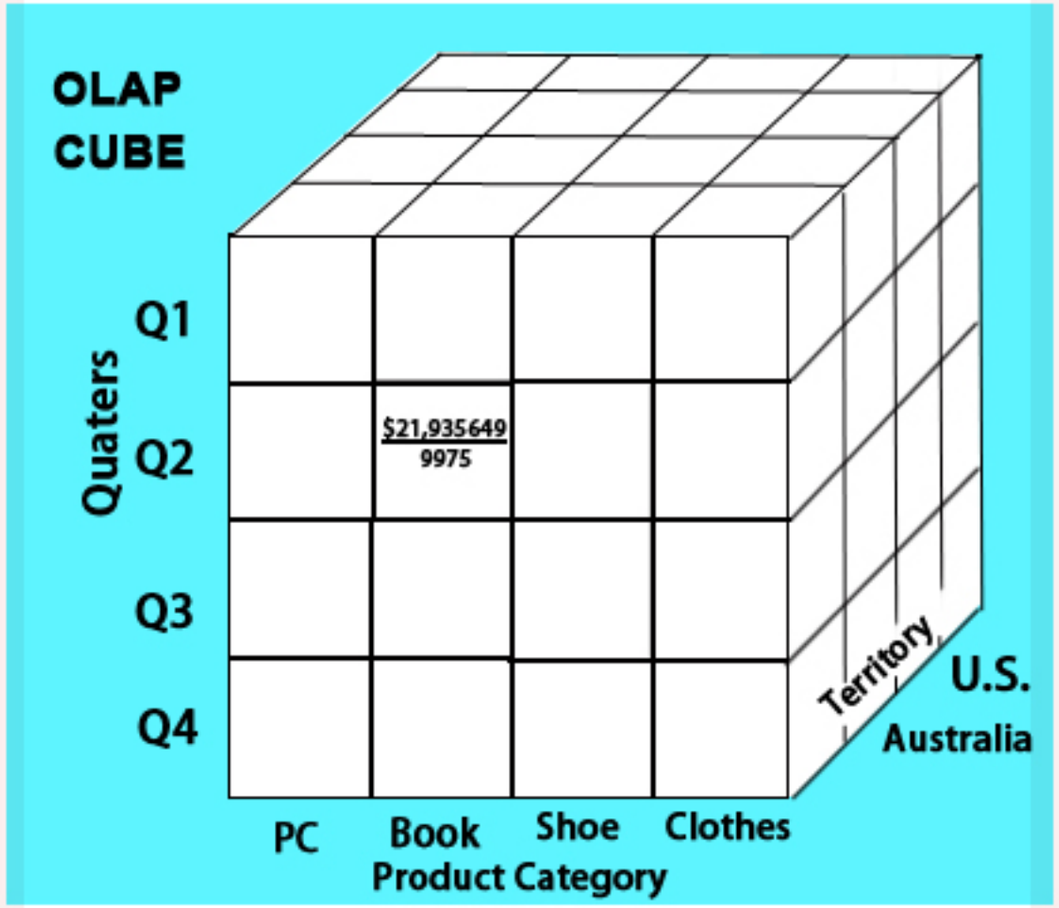

UPDATE
DELETE
INSERT
Extract
Transform
Load
1. OLTP
2. ETL
3. OLAP

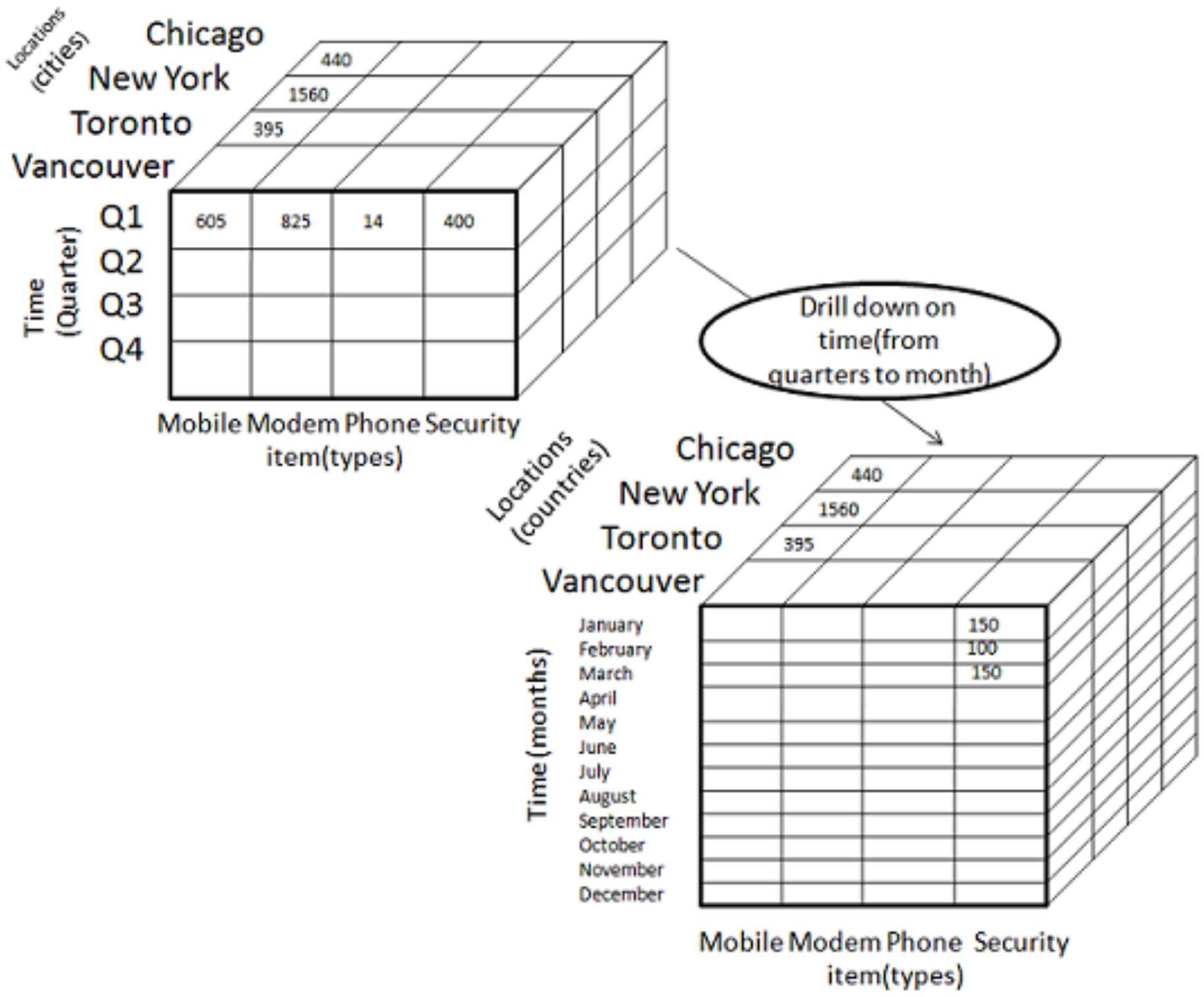
Slicing & Dicing

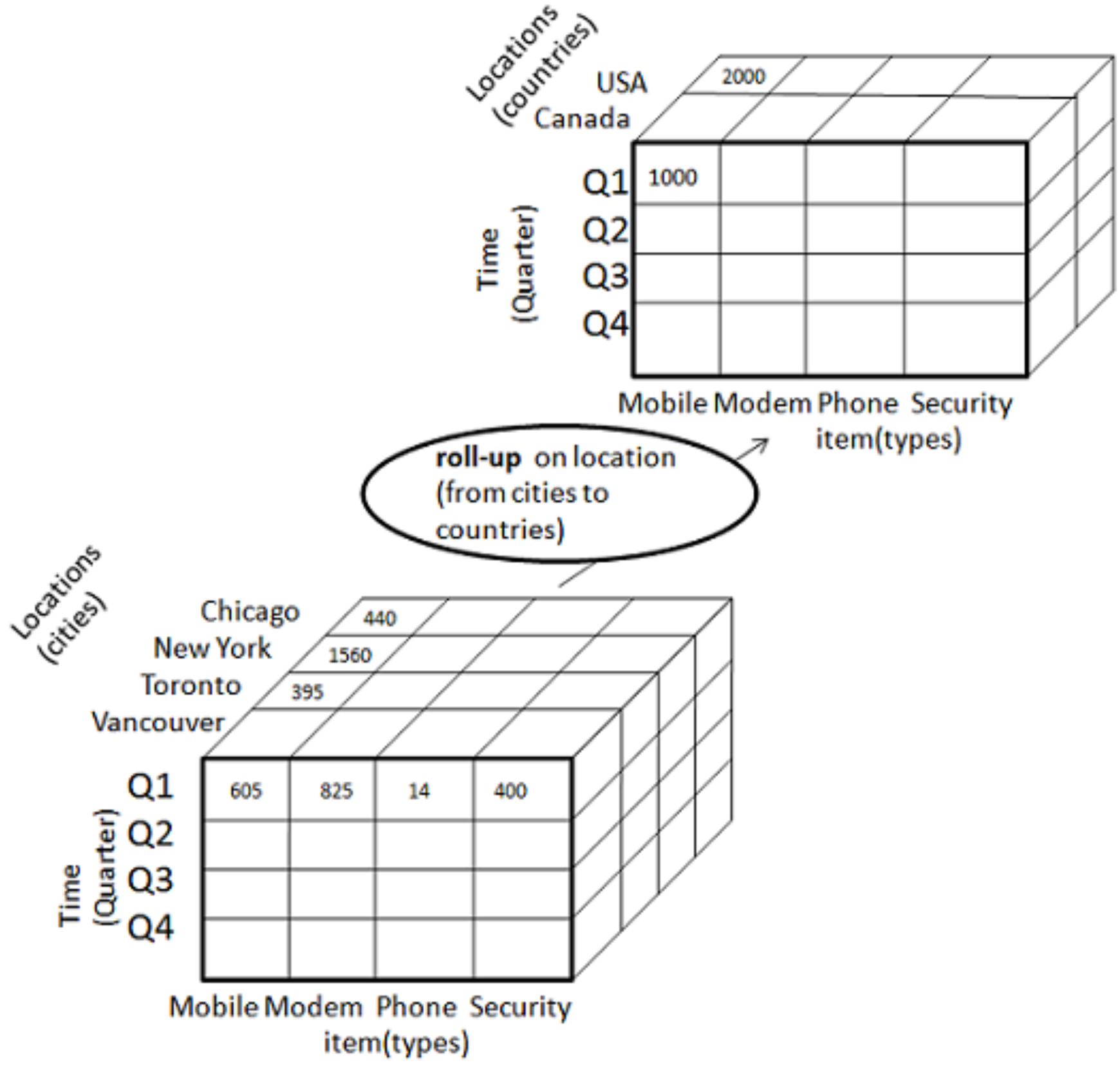
OLAP cube operations

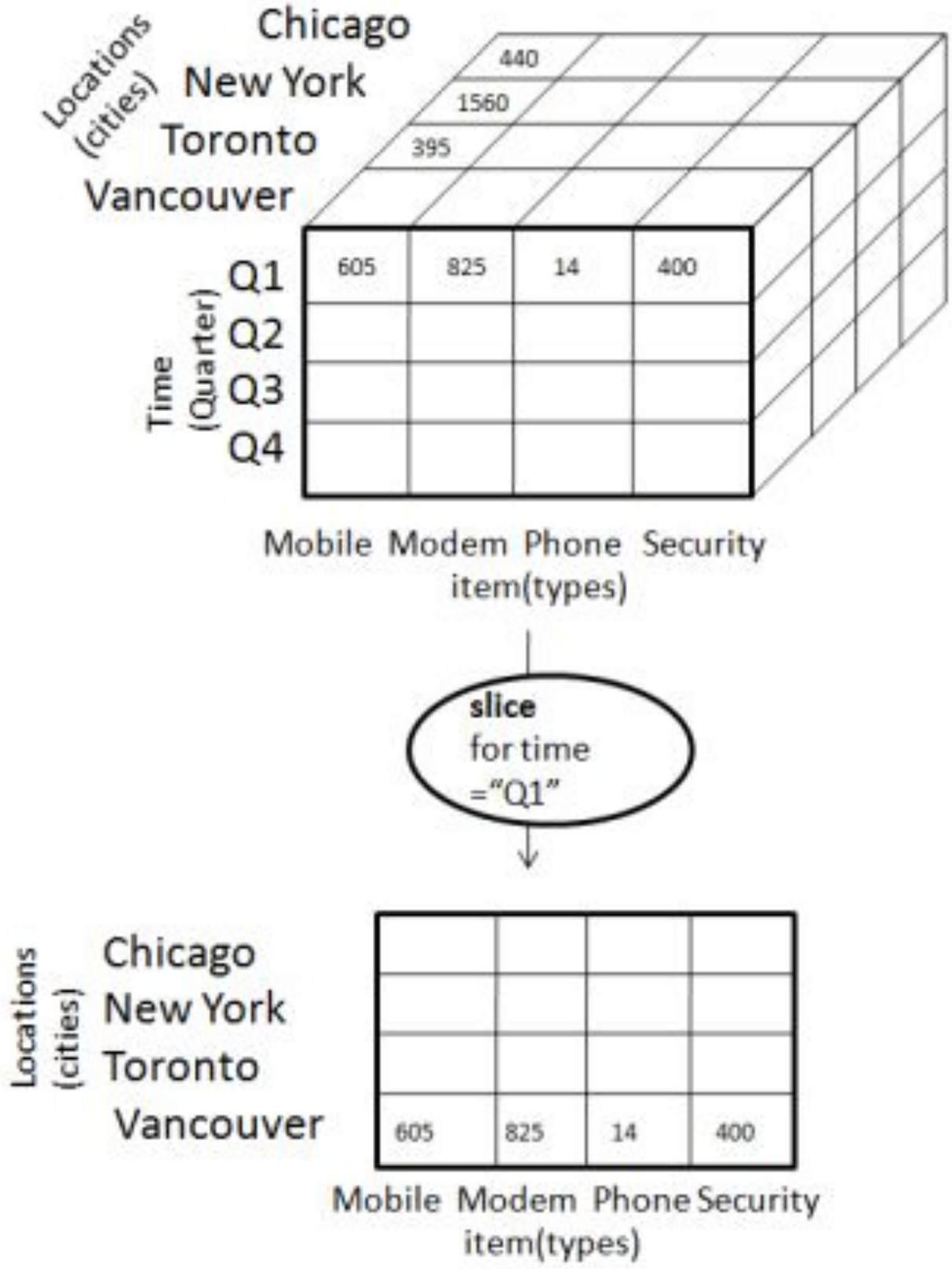


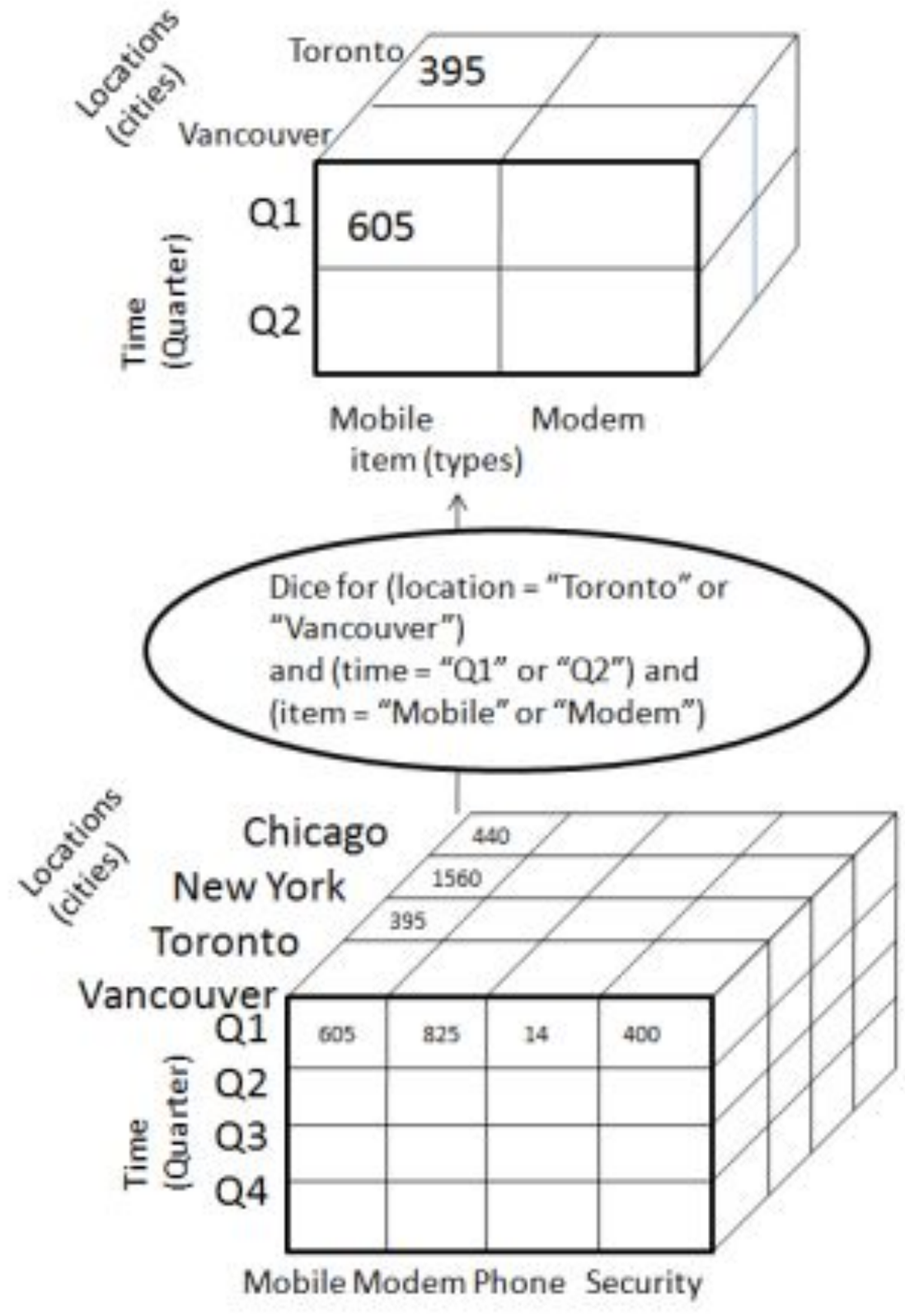


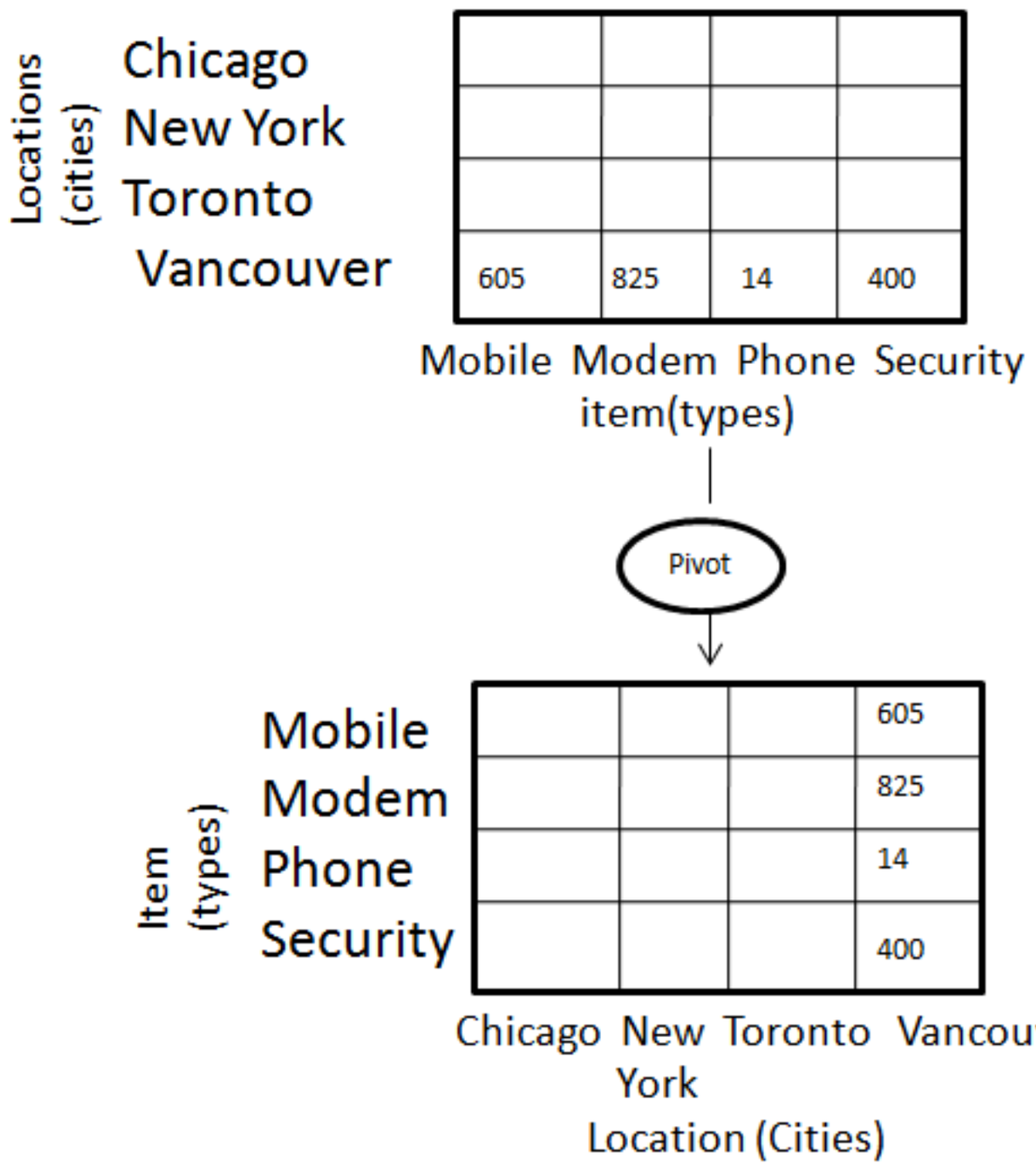

OLAP
CUBE
Quaters
Q1
Q2
Q3
Q4
$21,935649
9975
PC
Book
Shoe
Clothes
Product Category
Territory
U.S.
Australia


Locations (cities)
Chicago
New York
Toronto
Vancouver
440
1560
395
Time (Quarter)
Q1
Q2
Q3
Q4
605
825
14
400
Mobile Modem Phone Security
item(types)
Drill down on time(from quarters to month)
Locations (countries)
Chicago
New York
Toronto
Vancouver
440
1560
395
Time (months)
January
February
March
April
May
June
July
August
September
October
November
December
150
100
150
Mobile Modem Phone Security
item(types)


Locations (countries)
USA
Canada
2000
1000
Q1
Q2
Q3
Q4
Time (Quarter)
Mobile Modem Phone Security
item(types)
roll-up on location (from cities to countries)
Locations (cities)
Chicago
New York
Toronto
Vancouver
440
1560
395
605
825
14
400
Q1
Q2
Q3
Q4
Time (Quarter)
Mobile Modem Phone Security
item(types)


Locations (cities)
Chicago
New York
Toronto
Vancouver
440
1560
395
Time (Quarter)
Q1
Q2
Q3
Q4
605
825
14
400
Mobile
Modem
Phone
Security
item(types)
slice
for time
="Q1"
Locations (cities)
Chicago
New York
Toronto
Vancouver
605
825
14
400
Mobile
Modem
Phone
Security
item(types)


Locations (cities)
Toronto
395
Vancouver
Q1
605
Q2
Time (Quarter)
Mobile
Modem
item (types)
Dice for (location = "Toronto" or "Vancouver")
and (time = "Q1" or "Q2") and
(item = "Mobile" or "Modem")
Locations (cities)
Chicago
440
New York
1560
Toronto
395
Vancouver
Q1
605
825
14
400
Q2
Q3
Q4
Time (Quarter)
Mobile
Modem
Phone
Security

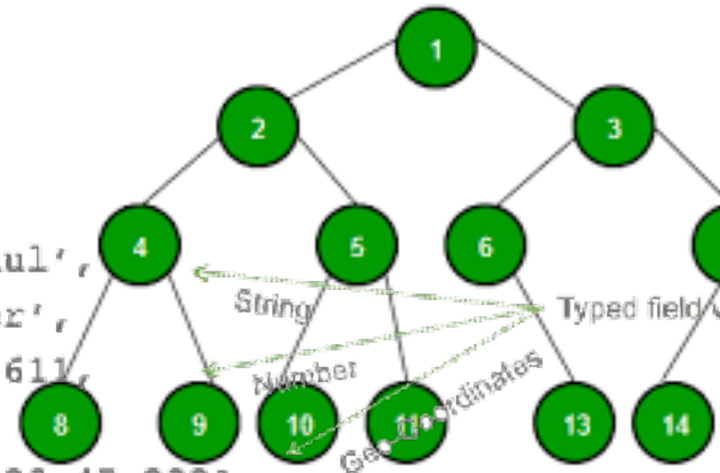

Locations (cities)
Chicago
New York
Toronto
Vancouver
605
825
14
400
Mobile
Modem
Phone
Security
item(types)
Pivot
Item (types)
Mobile
Modem
Phone
Security
605
825
14
400
Chicago
New York
Toronto
Vancouver
Location (Cities)

Rdbms (Referential Integrity)

tblPerson			
ID	Name	Email	GenderID
1	Jade	j@j.com	2
2	Mary	m@m.com	3
3	Martin	ma@ma.com	1
4	Rob	r@r.com	NULL
5	May	may@may.com	2
6	Kristy	k@k.com	NULL

tblGender	
ID	Gender
1	Male
2	Female
3	Unknown

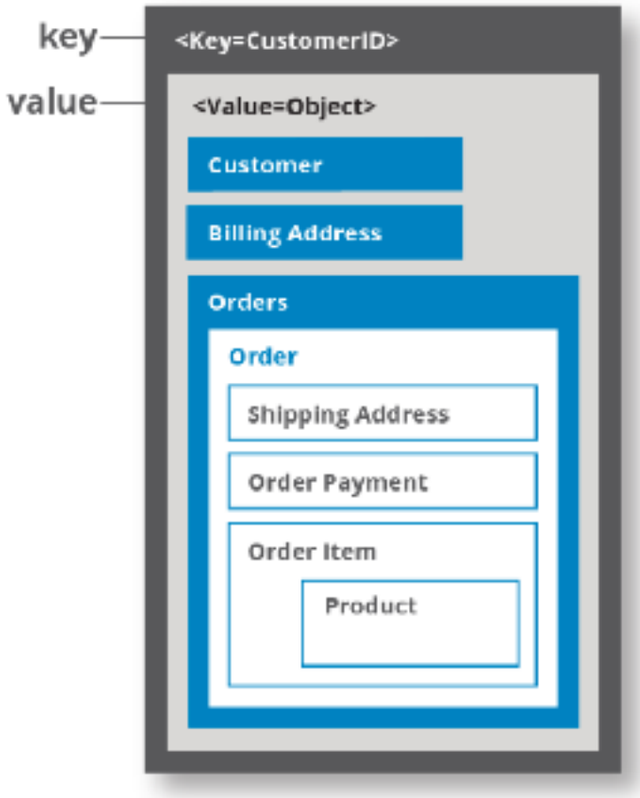
Fields

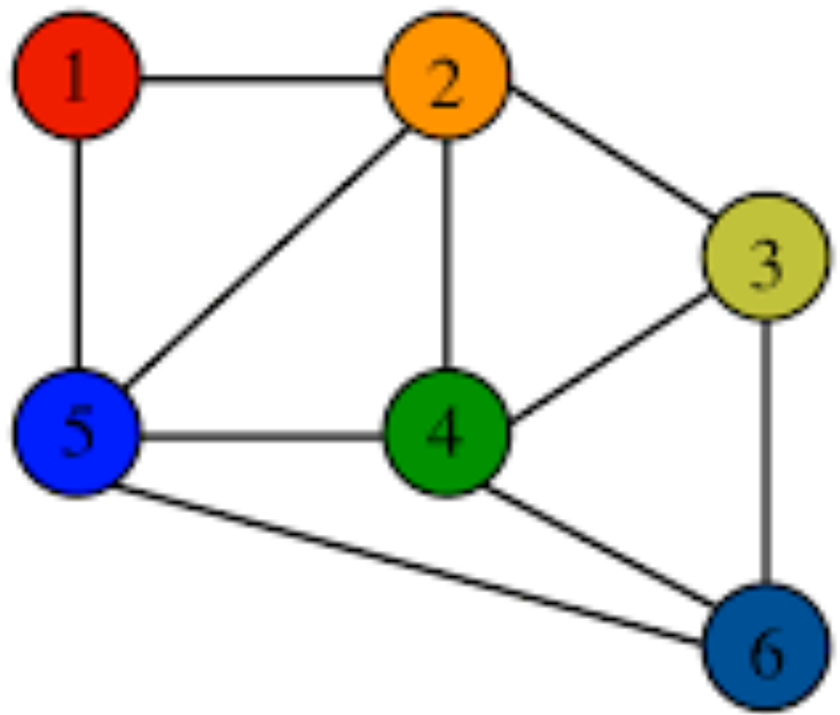



```
{
  first_name: 'Paul',
  surname: 'Miller',
  cell: 447557505611,
  city: 'London',
  location: [45.123,47.232],
  Profession: ['banking', 'finance', 'trader'],
  cars: [
    { model: 'Bentley',
      year: 1973,
      value: 100000, … },
    { model: 'Rolls Royce',
      year: 1965,
      value: 330000, … }
  ]
}
```

Fields can contain an array of documents

key	value
123	123 Main St.
126	(805) 477-3900

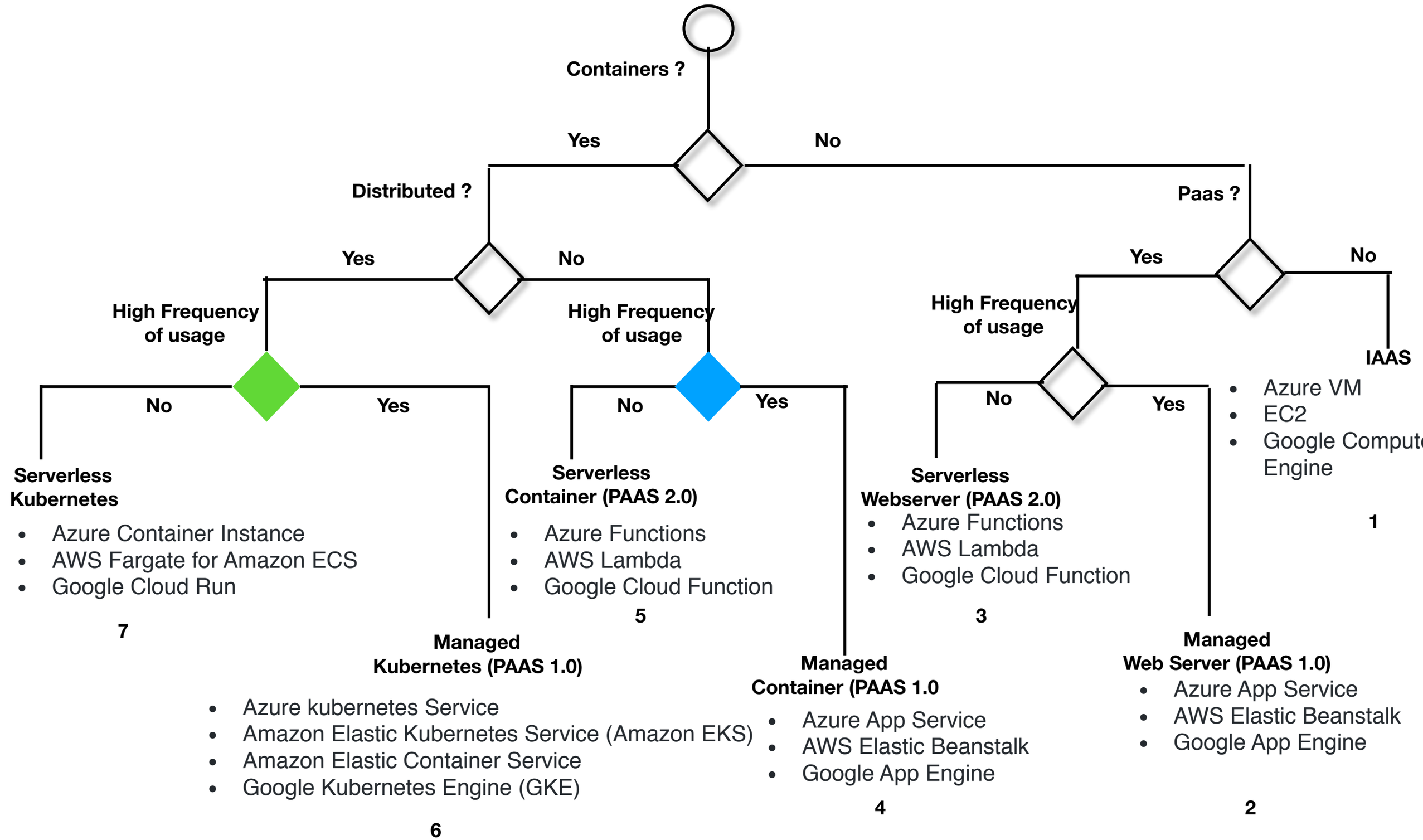




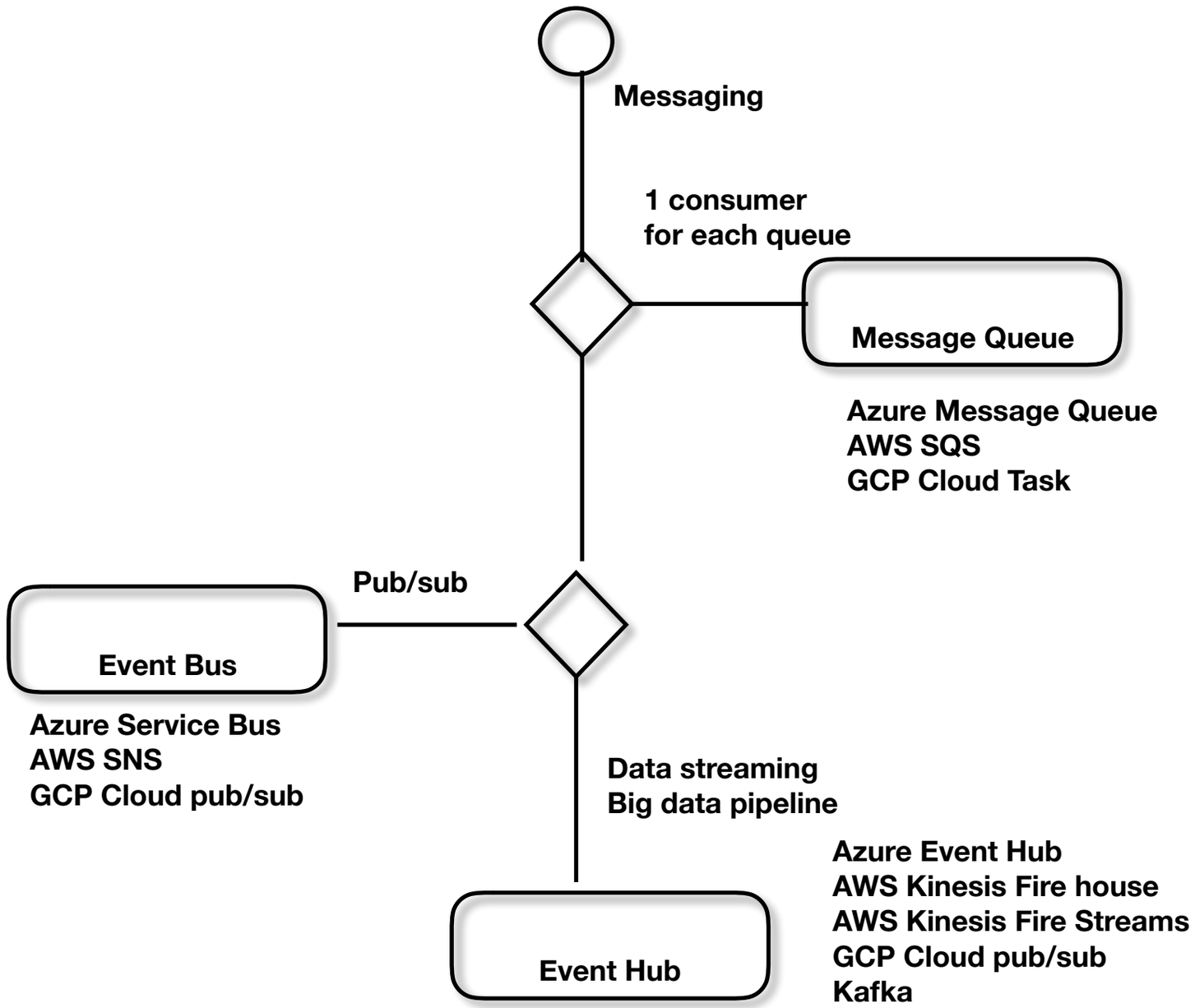
Wide Column

- Spotify uses Cassandra to store user profile attributes and metadata about artists, songs, etc. for better personalization
- Facebook initially built its revamped Messages on top of HBase, but is now also used for other Facebook services like the Nearby Friends feature and search indexing
- Outbrain uses Cassandra to serve over 190 billion personalized content recommendations each month

Choose Operational Compute



Choose Messaging



System Storage

Binary	Operational Database	Analytical Database
4	4	6

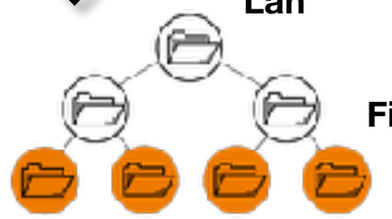
Binary Storage

Object Storage

Immutable
$
Low Perf
High Volume
0.01$/GB

- Azure Blobs
- AWS S3
- google Cloud Storage

4

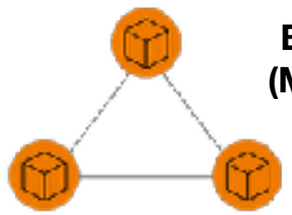


File Storage

Mutable
$$$$
Medium Perf
Medium Voulme

- Azure Files
- EFS
- Google Filestore
- NFS

3



Block Storage (Managed Disk)

Mutable
$$$
High Perf
Low Volume
1$/GB

- Azure Disks
- AWS EBS
- Google Persistent Disk

2

Unmanaged Disk

Mutable
$
High Perf

- Local Disk

1

Access across VM (Network Storage)

Single VM Storage

Rest API / SOAP protocol

Network File System (NFS) Protocol / Server Message Block (SMB) protocol

Persistent

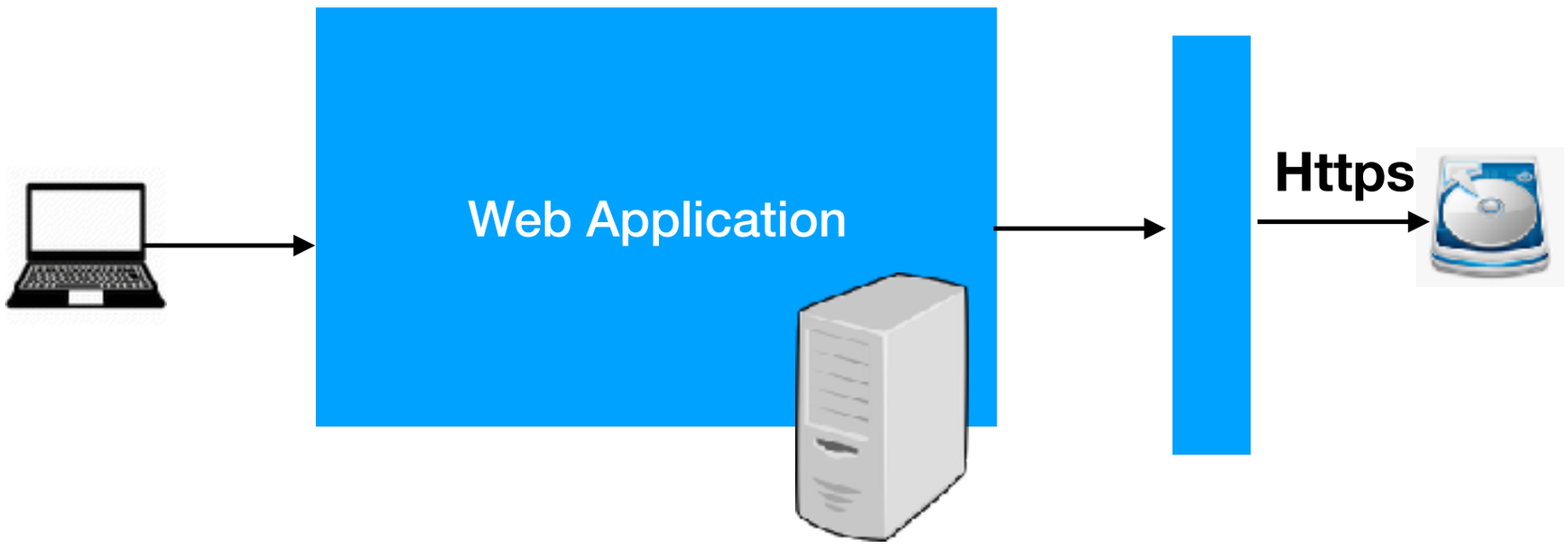
Temporary

Internet

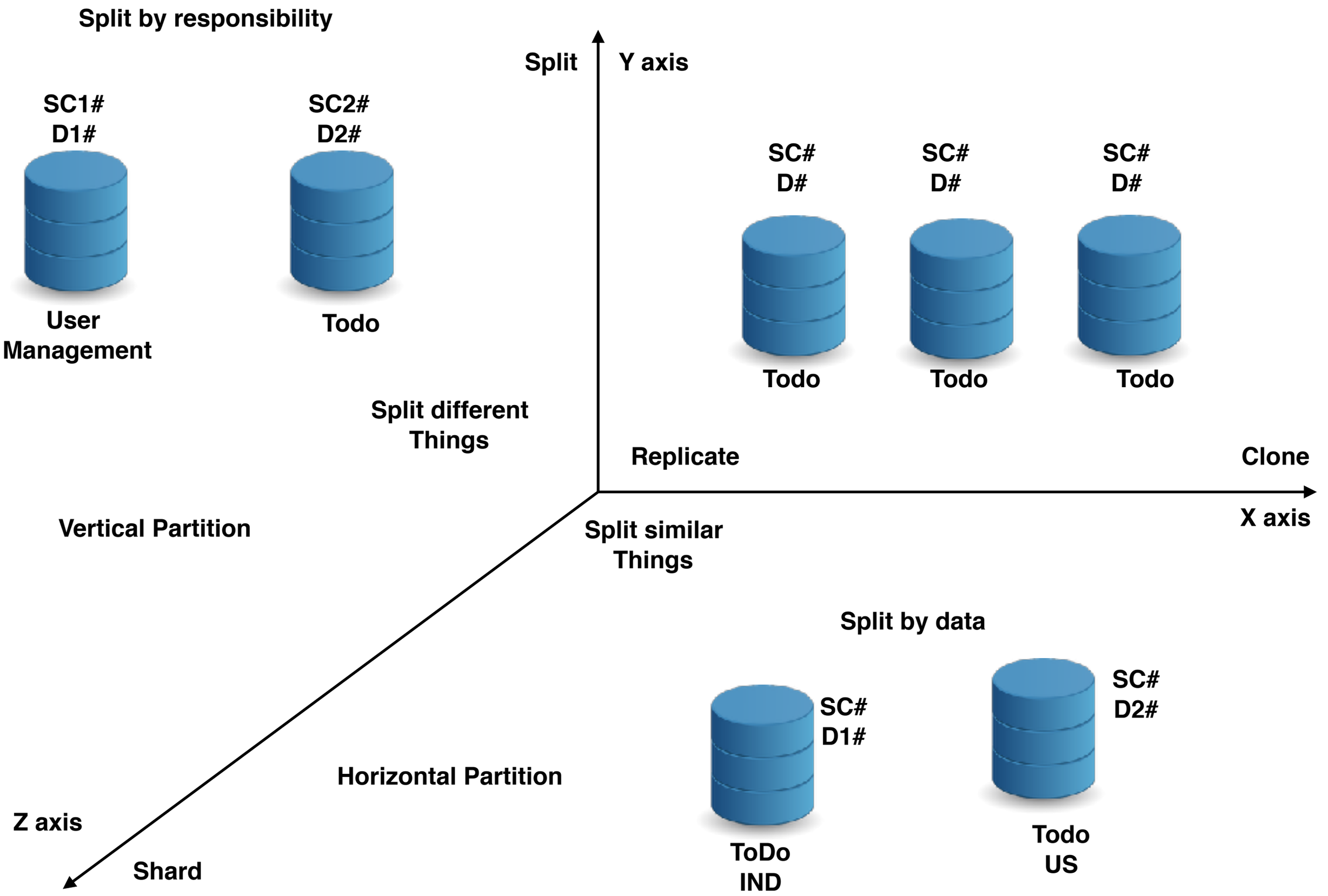
Lan

* AWS DataSync subscription is required to provide support for Server Message Block (SMB) protocol


Web Application
Https


Web Application
Https

Scalability Cube - 50 rules for high Scalability



Ecom

Split different Things

Vertical Partition

User

Inventory

Accounts

Order

Horizontal Partition

Shard

Inventory: M

Inventory : F

Order: AUS

Order: USD

Clone

Inventory: M

Inventory: M

Operational

Structured

[ACID - transaction]

RDBMS (OLTP)

- AWS RDS
- Azure SQL
- Google Cloud Sql
- Google Cloud Spanner
- **Postgres**
- **Mysql**

1

Semi Structured

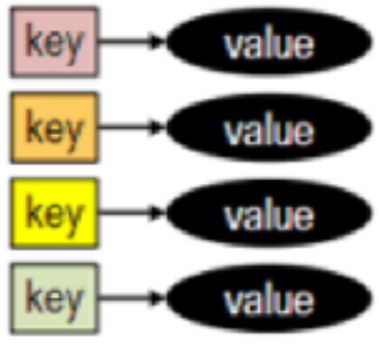
**[* BASE,
SCALE- shareded,
* Flexible Schema]**

Dictionary (key value)

Persistent | **Cache**

- AWS Elastic
- Azure Cache (redis)
- GCP memory Store
- **Redis**, Hazelcast
- Memcached, Ignite

- AWS Dynamo Db
- Azure Cosmos
- Azure Table
- Google Big Table



2

Nested Objects

Document

- AWS Document DB
- Azure Cosmos
- Firestore
- **Mongodb**, **Couchbase**,Solr

3

Entity Relationship

Graph

- AWS Neptune
- Azure Cosmos
- Google Janusgraph
- **Neo4j**
- Graph,orientdb

4

Rows vs. Documents

	Table
Row 1	Data
Row 2	Data
Row 3	Data
...	:

Document 1

```
{
  data
]
```

Document 2

```
{
  data
}
```

Document 3

```
{
  data
]
```

...

Columns vs. Properties

Col1	Col2	Col3	Col4
Data	Data	Data	Data
Data	Data	Data	Data

Document 1

```
{
  "prop1": data,
  "prop2": data,
  "prop3": data,
  "prop4": data
}
```

Document 2

```
{
  "prop1": data,
  "prop2": data,
  "prop3": data,
  "prop4": data
}
```

Document 3

```
{
```

Schema vs. Schema-Free

ID	Name	IsActive	Dob
1	John Smith	True	8/30/1964
2	Sarah Jones	False	2/18/2002
3	Adam Stark	True	7/13/1987

Document 1

```
[
  "id": "1",
  "name": "John Smith",
  "isActive": true,
  "dob": "1964-30-08"
]
```

Document 2

```
{
  "id": "2",
  "fullName": "Sarah Jones",
  "isActive": false,
  "dob": "2002-02-18"
}
```

Document 3

```
{
  "id": "3",
  "fullName":
  {
    "first": "Adam",
    "last": "Stark"
  },
  "isActive": true,
  "dob": "2015-04-19"
}
```

Normalized vs. Denormalized

User Table

UserID	Name	Dob
1	John Smith	8/30/1964

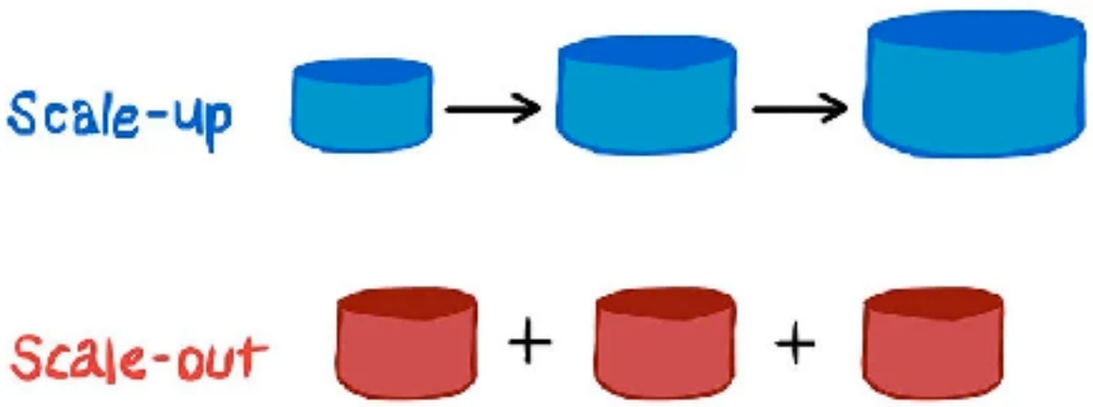
Holdings Table

StockID	UserID	Qty	Symbol
1	1	100	MSFT
2	1	75	WMT

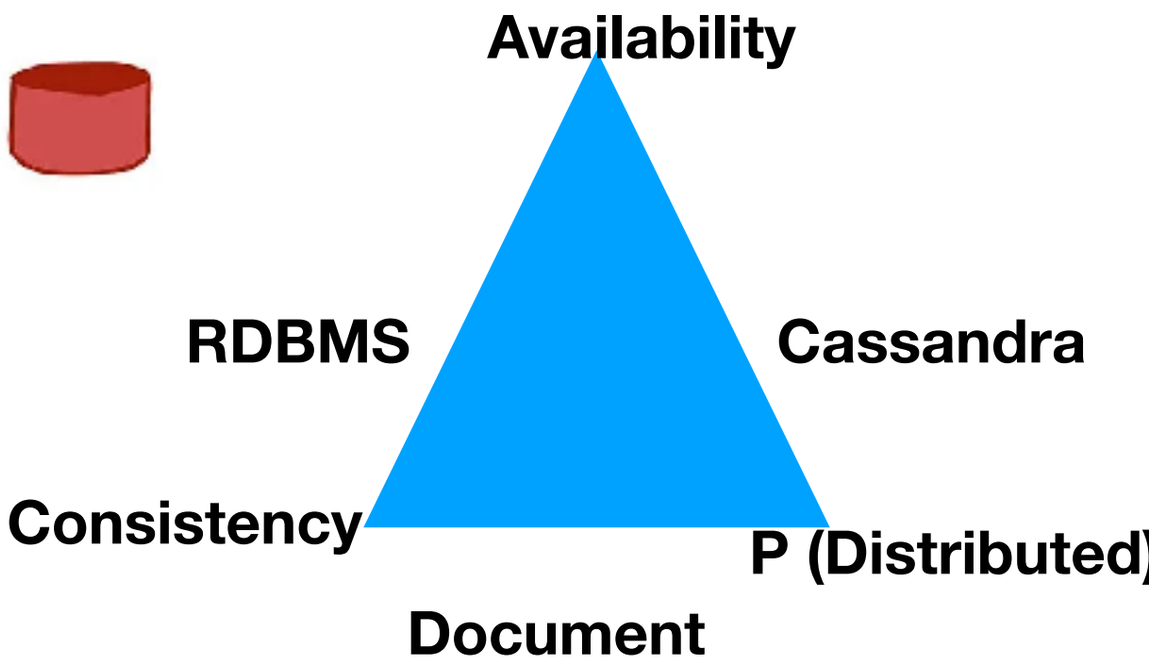
Document

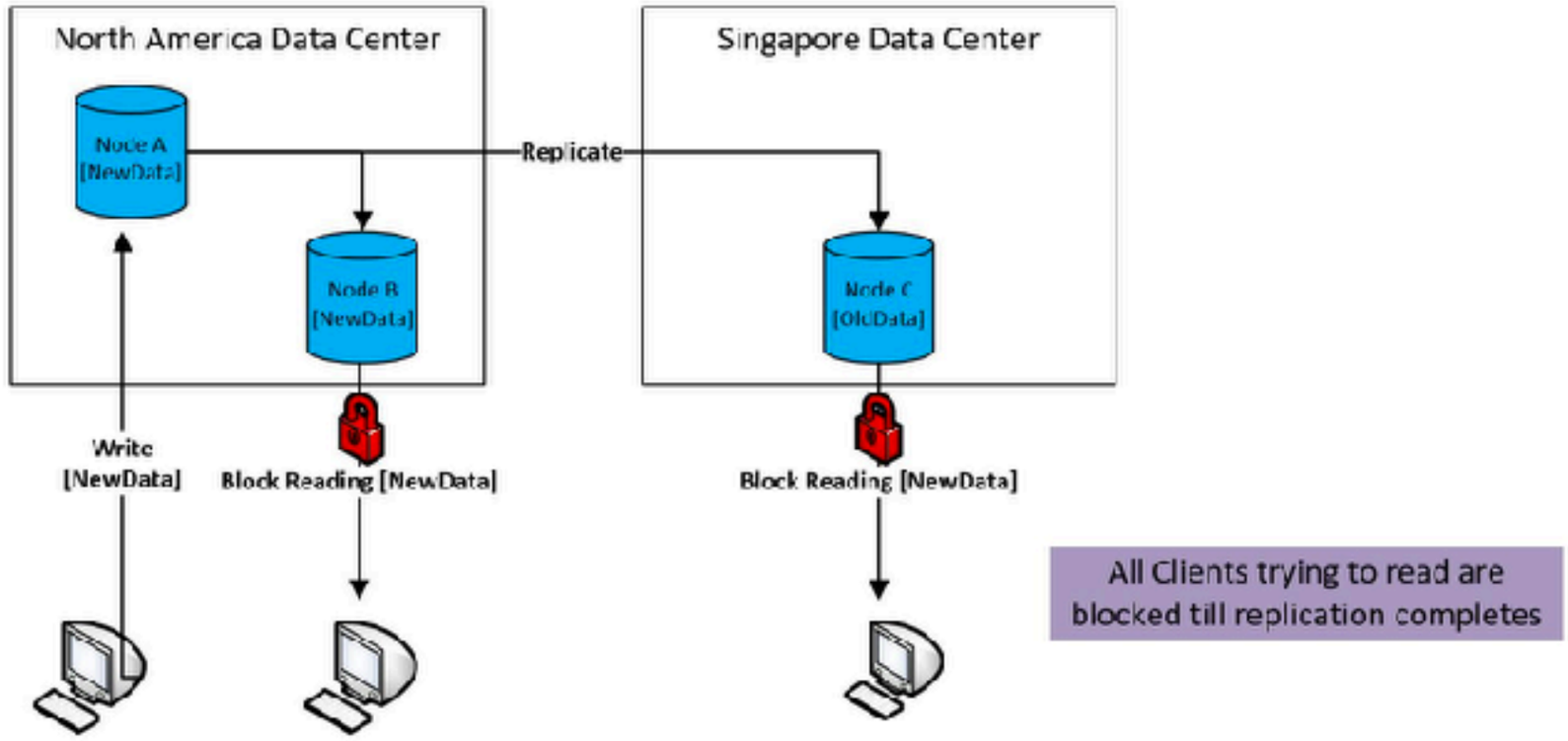
```
{
  "id": "1",
  "name": "John Smith",
  "dob": "1964-30-08",
  "holdings": [
     { "qty": 100, "symbol": "MSFT" },
     { "qty": 75, "symbol": "WMT" }
   ]
}
```

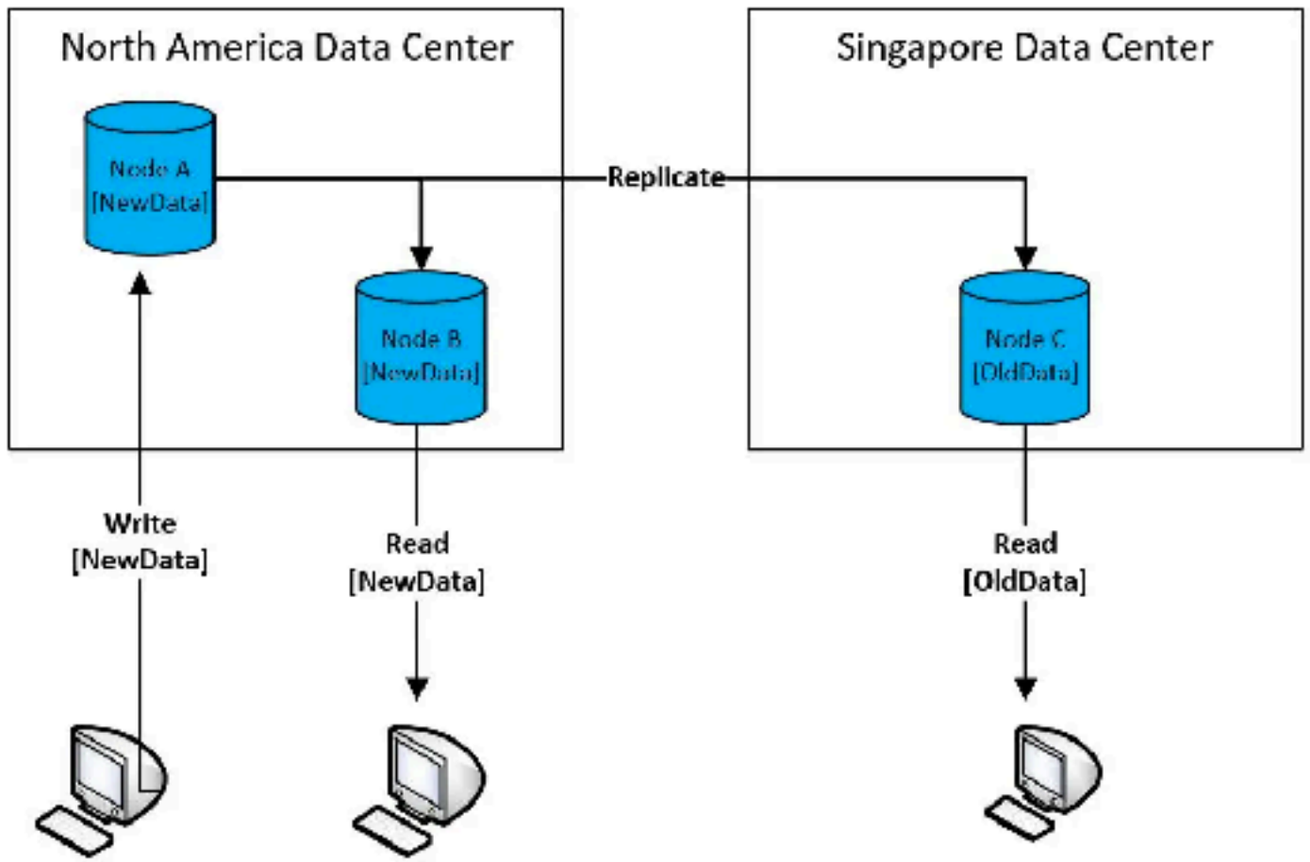
Scale-Up vs. Scale-Out



Strong Consistency vs. Eventual Consistency




North America Data Center
Singapore Data Center
Node A [NewData]
Replicate
Node B [NewData]
Node C [OldData]
Write [NewData]
Block Reading [NewData]
Block Reading [NewData]
All Clients trying to read are blocked till replication completes


North America Data Center
Singapore Data Center
Node A [NewData]
Replicate
Node B [NewData]
Node C [OldData]
Write [NewData]
Read [NewData]
Read [OldData]

COMMON COMPARISONS BETWEEN

MYSQL & NOSQL

	MySQL	NoSQL
Nature	Relational Database	Non-Relational Database
Design	Based on the concept of tables	Based on the concept of documents
Scalable	Tough to scale due to its relational nature	Easily scalable big data compared to relational
Model	Detailed database model is needed before creation	No need of a detailed database model
Community	Vast community available	Community is growing rapidly, but still smaller compared to MySQL
Standardization	SQL is standard language	Lacks standard query language
Schema	The Schema is rigid	The Schema is dynamic
Flexibility	Not very flexible in terms of design	Very flexible in terms of design
Insertions	Inserting new columns or fields affect the design	No effect on the design with the insertion of new columns or fields

Facebook, Wikipedia, Quora, Flickr	MySQL
Twitter	MySQL for tweets and users their own special kind of graph database, FlockDB, built on top of MySQL their own version of Memcached
LinkedIn	Oracle Database and Voldemort
YouTube	MySQL -> BigTable
Microsoft, Myspace	***SQL Server***
Yahoo	***PostgreSQL***

Key Value	○ Twitter uses Redis to deliver your Twitter timeline
	○ Pinterest uses Redis to store lists of users, followers, unfollowers, boards, and more
	○ Coinbase uses Redis to enforce rate limits and guarantee correctness of Bitcoin transactions
	○ Quora uses Memcached to cache results from slower, persistent databases
Graph	○ Walmart uses Neo4j to provide customers personalized, relevant product recommendations and promotions
	○ Medium uses Neo4j to build their social graph to enhance content personalization
	○ Cisco uses Neo4j to mine customer support cases to anticipate bugs
Document	○ SEGA uses MongoDB for handling 11 million in-game accounts
	○ Cisco moved its VSRM (video session and research manager) platform to Couchbase to achieve greater scalability
	○ Aer Lingus uses MongoDB with Studio 3T to handle ticketing and internal apps
	○ Built on MongoDB, The Weather Channel's iOS and Android apps deliver weather alerts to 40 million users in real-time

Analytical

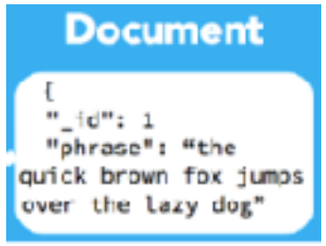
Structured

Semi Structured

Big data Db

DWH (OLAP)

- AWS Redshit
- **Azure Synapse**
- GCP BigQuery
- Snow flakes
- Kudu, Druid, Pinot
- Click house

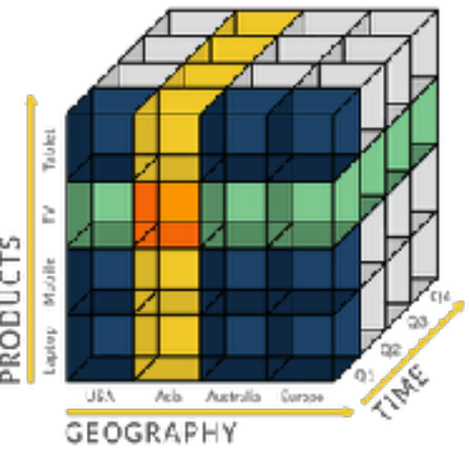


Text Database

- AWS **Elastic Search**
- AWS cloud Search
- Azure Cognitive Search
- Google Search API
- Elastic Search, Solr

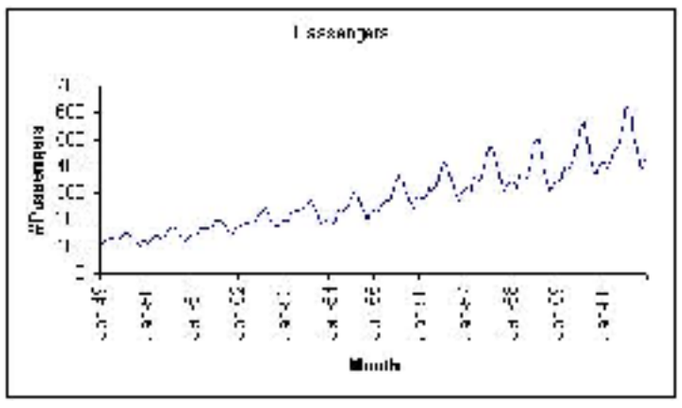
Wide Column (Column db)

- AWS Keyspaces
- Cosmos
- Big Table
- **Hbase, cassandra**
- Scylla



Time Series

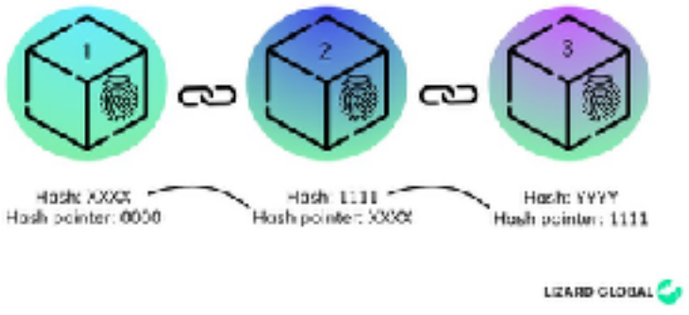
- AWS Timestream
- Cosmos
- Big Table, Big Query
- InfluxDB, Scylla
- OpenTSDB



Audit Trial

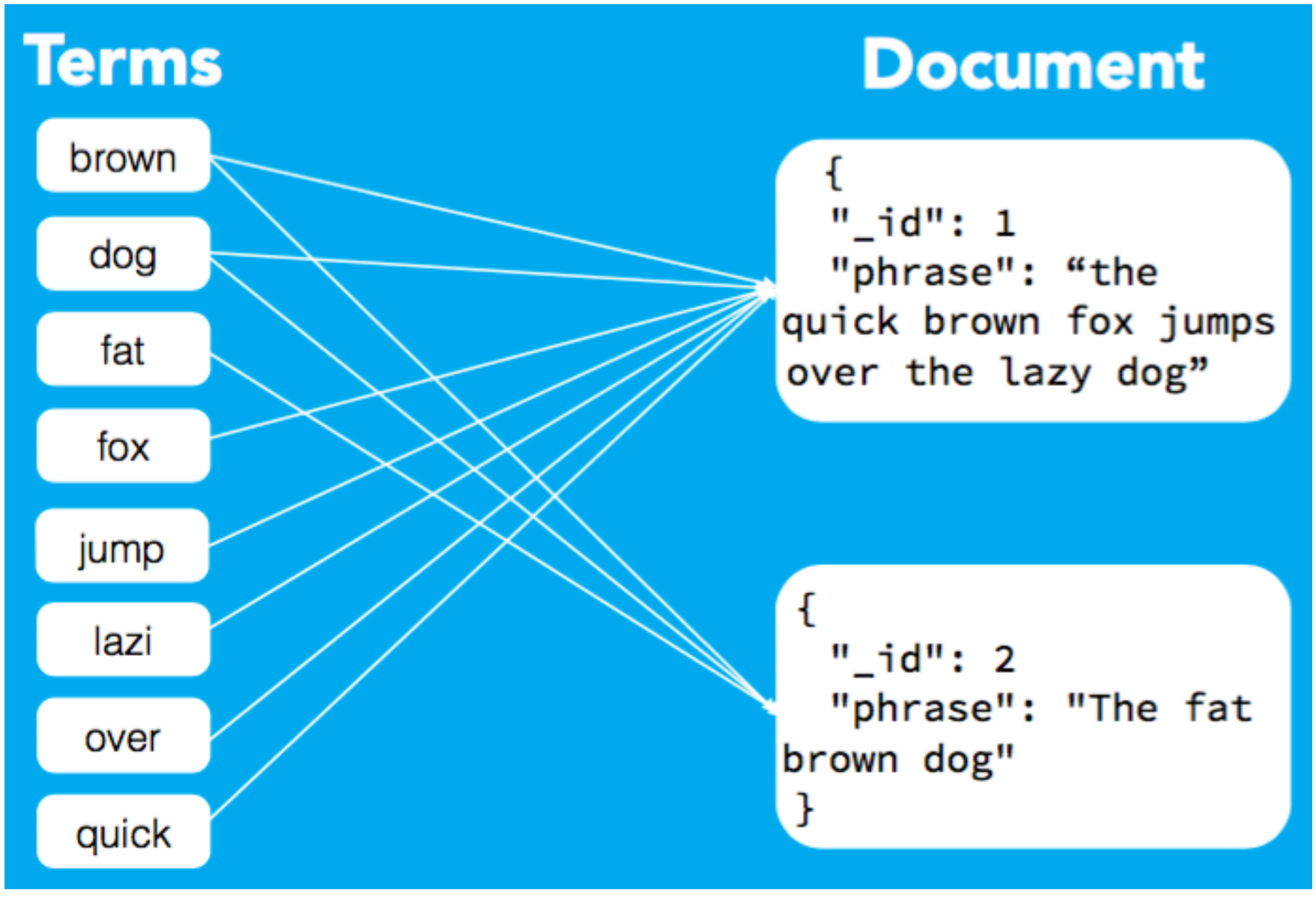
Immutable Logger

- AWS Quantum Ledger
- Azure SQL Database Ledger
- Google ?
- Hyper ledger Fabric



geo Spatial

- AWS Keyspaces
- Azure Cosmos
- Big Table, Big Query
- **Solr**, PostGIS
- Mongoldb (GeoJSON)


Terms
Document
brown
dog
fat
fox
jump
lazi
over
quick
{
"_id": 1
"phrase": “the quick brown fox jumps over the lazy dog”
{
"_id": 2
"phrase": "The fat brown dog"
}

Table

	Country	Product	Sales
Row 1	India	Chocolate	1000
Row 2	India	Ice-cream	2000
Row 3	Germany	Chocolate	4000
Row 4	US	Noodle	500

Row Store

Row 1	India
	Chocolate
	1000
Row 2	India
	Ice-cream
	2000
Row 3	Germany
	Chocolate
	4000
Row 4	US
	Noodle
	500

Column Store

Country	India
	India
	Germany
	US
Product	Chocolate
	Ice-cream
	Chocolate
	Noodle
Sales	1000
	2000
	4000
	500

Classic Relational Databases

Name	Age	Nickname	Employee
Gianarno Cuttizzoni Founder & CEO	40	Heidi	◉
Marco Botton Tuttofare	38		☑
Mariah Maclachlan Better Half	41	Potato	⊟
Valerie Liberty Head Chef	16	Val	☑

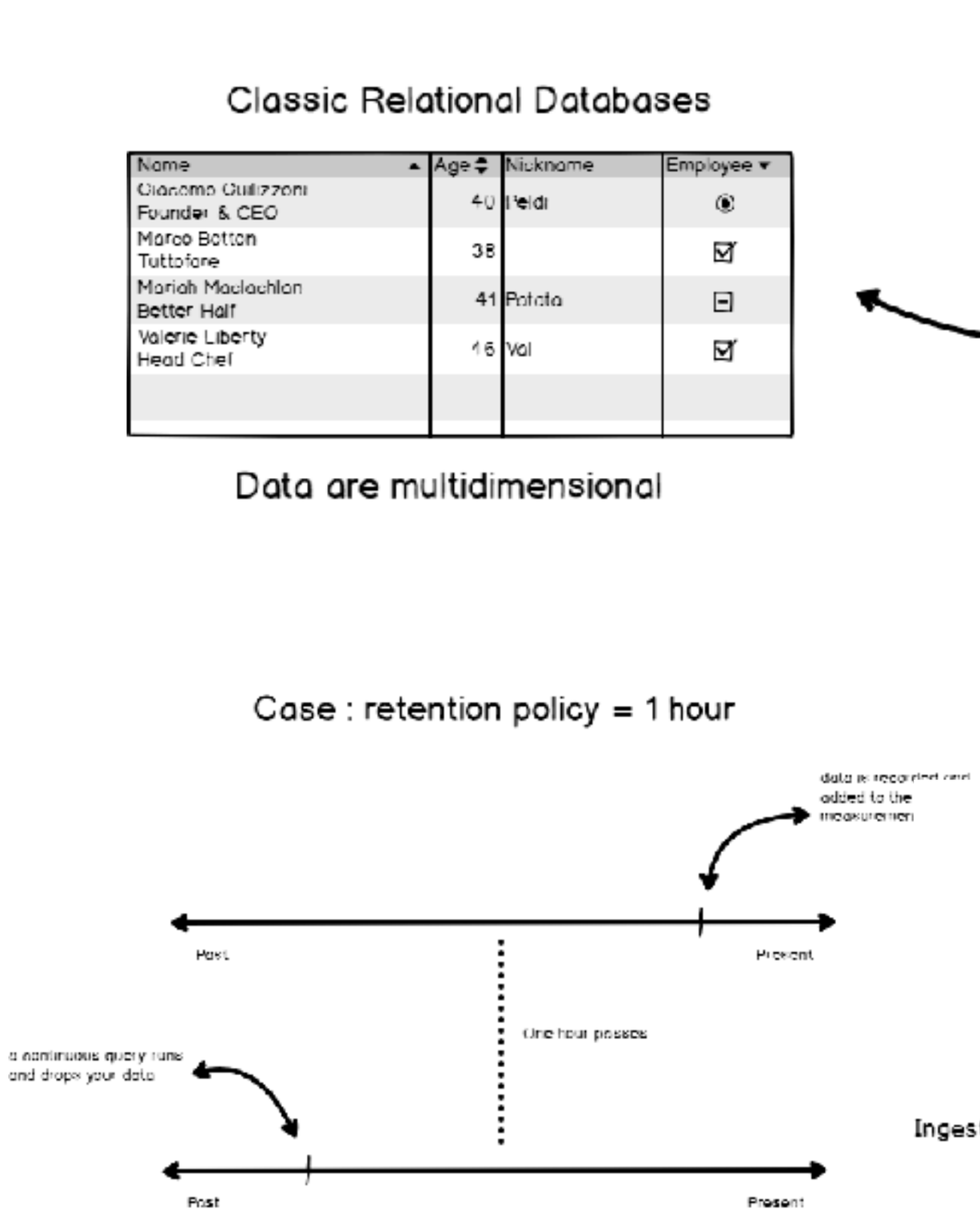
Data are multidimensional

Time Series Databases

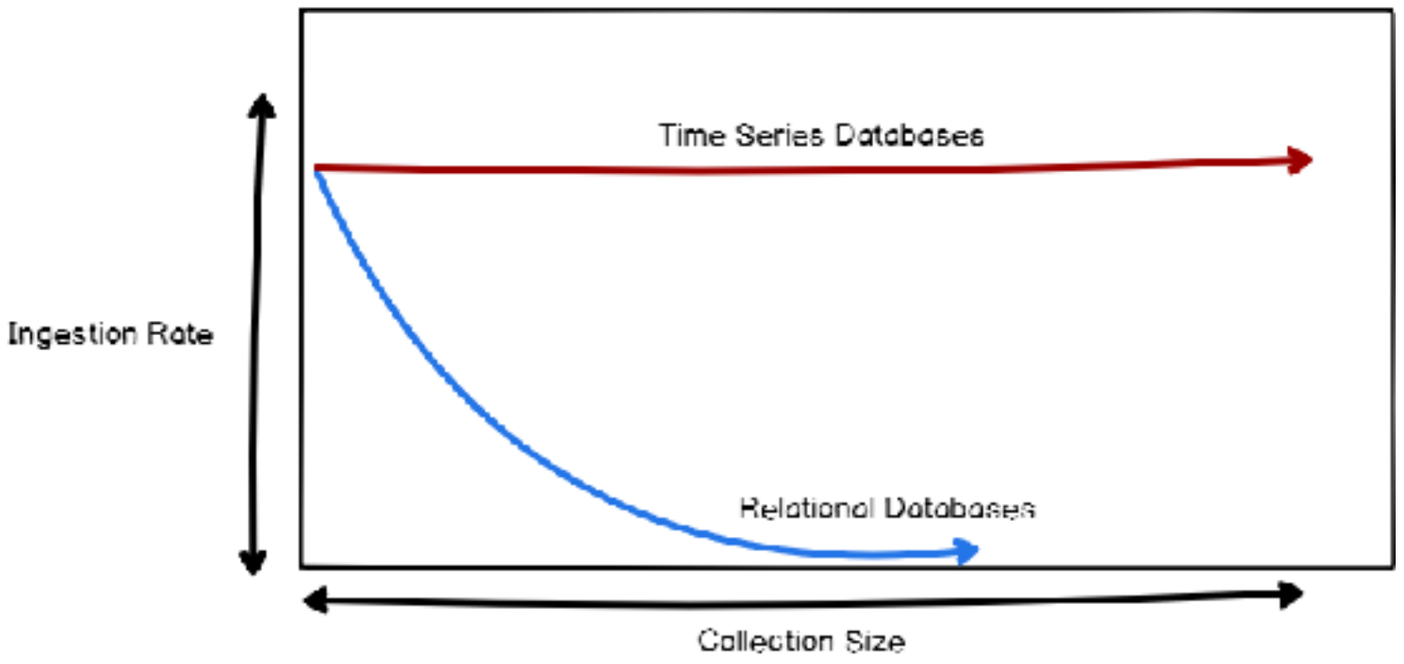
Sensor Temperature	Time
39.6	12/04/19 @ 14:12
41.2	12/04/19 @ 14:13
12.4	14/04/19 @ 12:15
18.5	16/04/19 @ 10:05

Data are aggregated over time

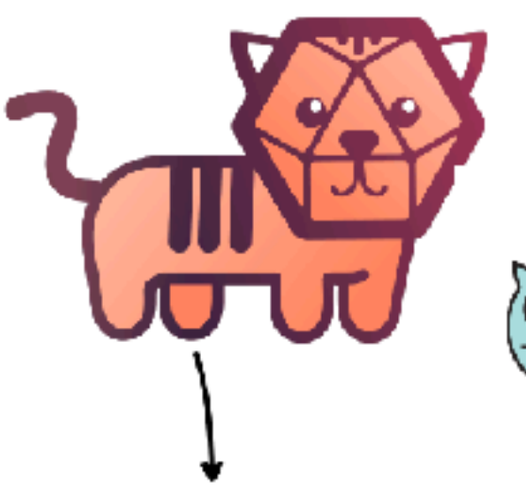
Case : retention policy = 1 hour



DBMS & TSDB Difference

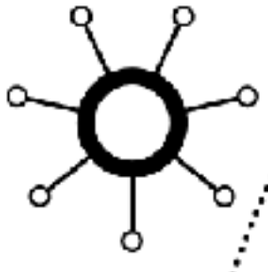


Tools that 'produce' TSDB data



monitoring done via specific query agents

collectd



StatsD

monitoring done via node exporters

Prometheus

Tools that 'consume' TSDB

chronograf

DATADOG

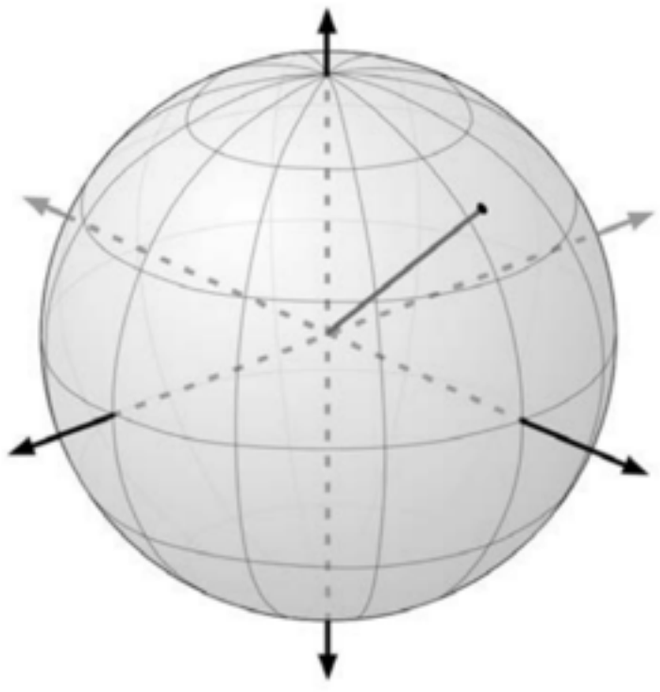
Prometheus

Grafana

kibana

*Non-exhaustive list

Car ID	Departure			Arrival		
	Date-Time	Latitude	Longitude	Date-Time	Latitude	Longitude
...	...	...	...	...	...	...
...	...	...	...	...	...	...
...	...	...	...	...	...	...



Row-oriented

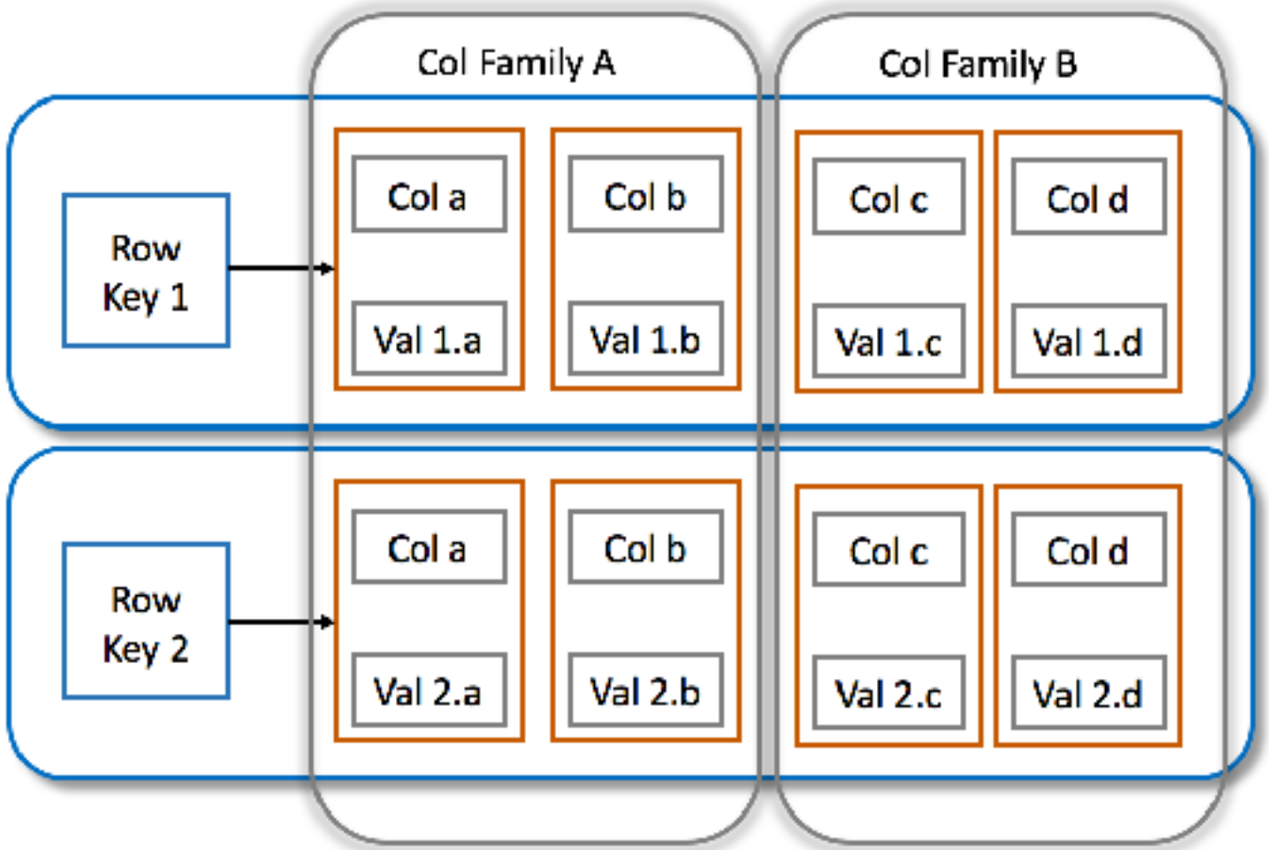
ID	Name	Grade	GPA
001	John	Senior	4.00
002	Karen	Freshman	3.67
003	Bill	Junior	3.33

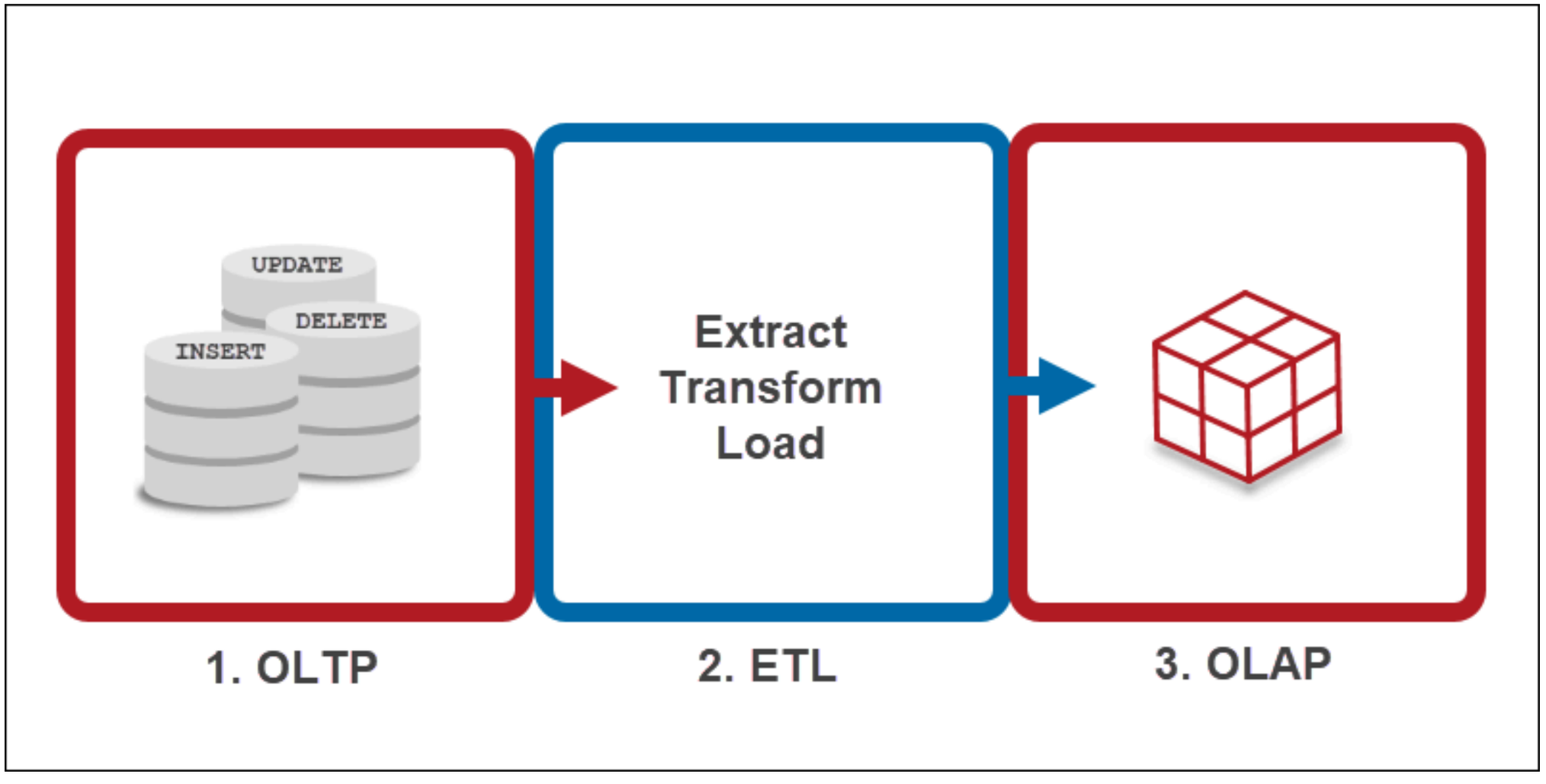
Column-oriented

Name	ID
John	001
Karen	002
Bill	003

Grade	ID
Senior	001
Freshman	002
Junior	003

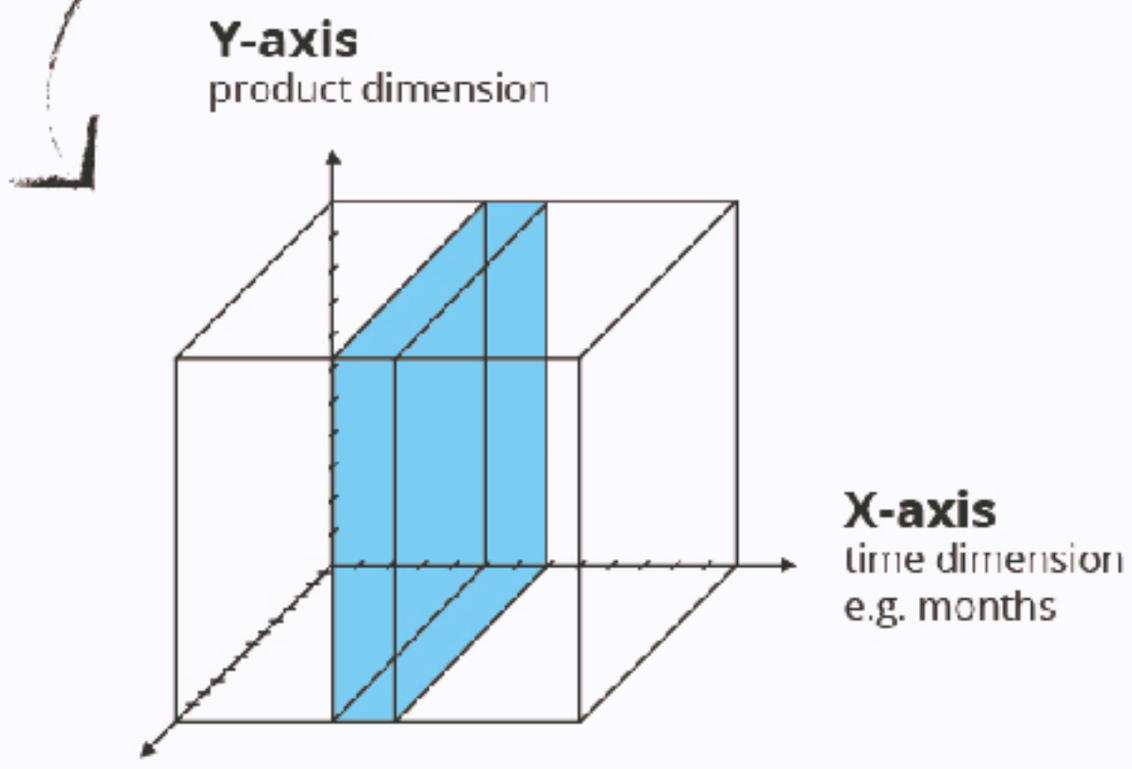
GPA	ID
4.00	001
3.67	002
3.33	003

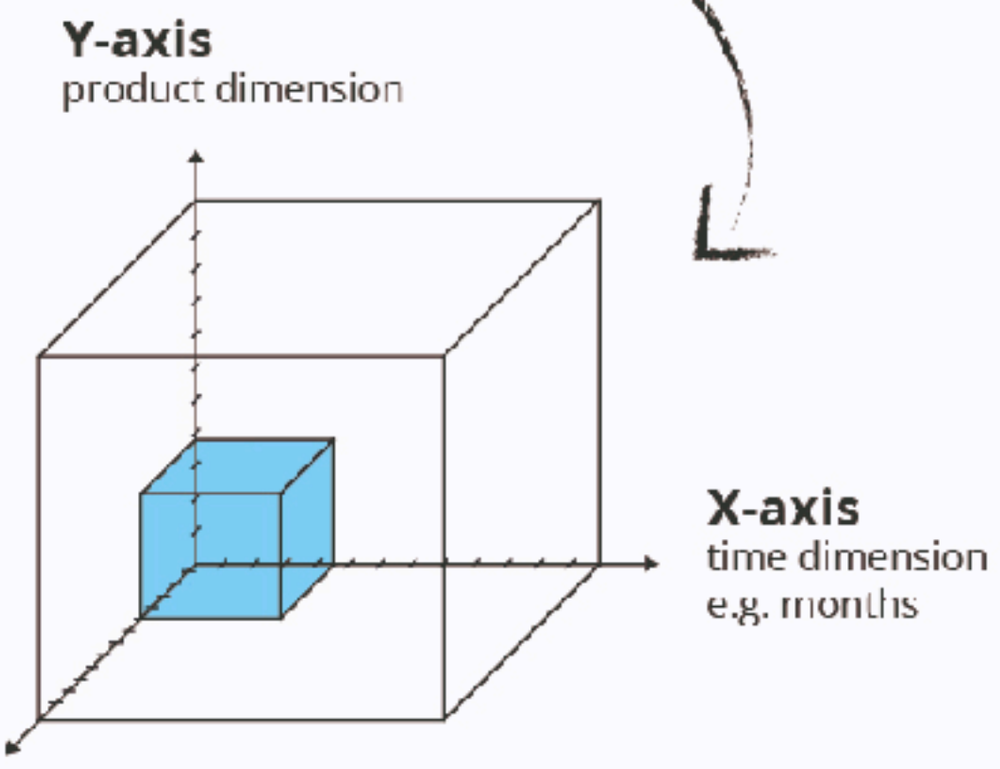


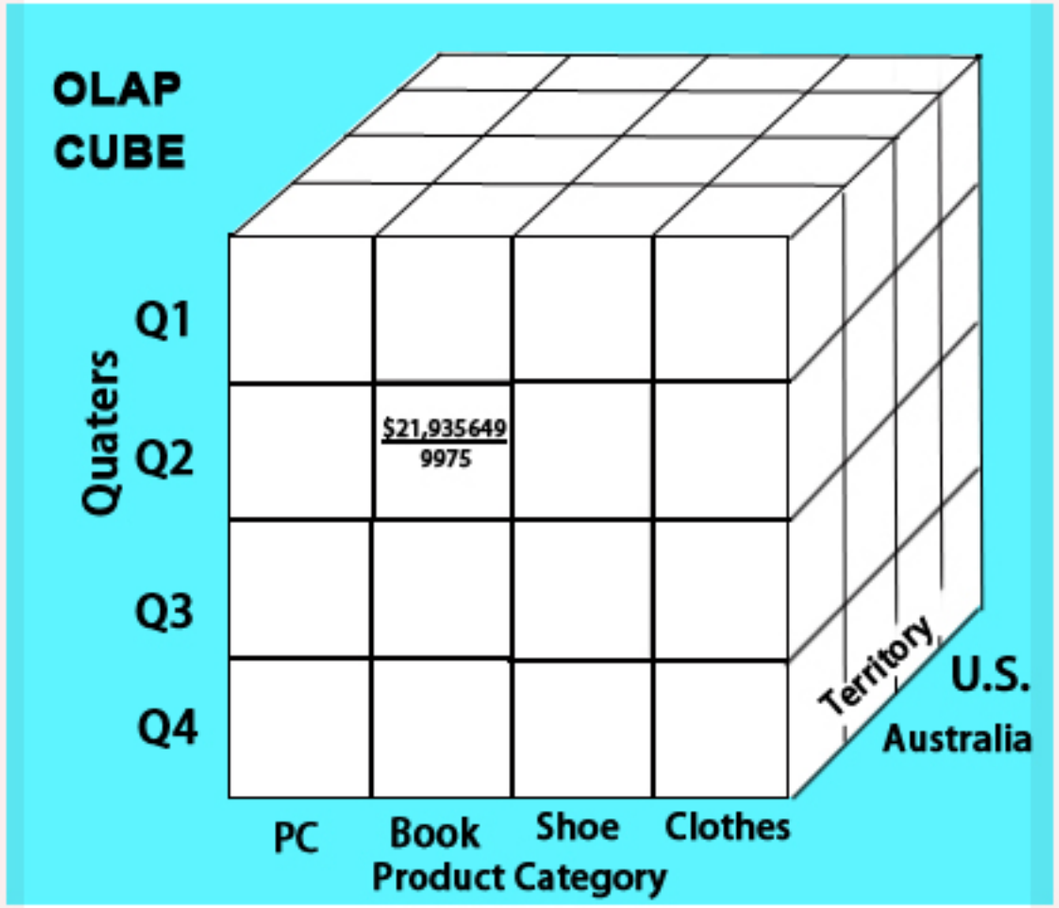

UPDATE
DELETE
INSERT
Extract
Transform
Load
1. OLTP
2. ETL
3. OLAP

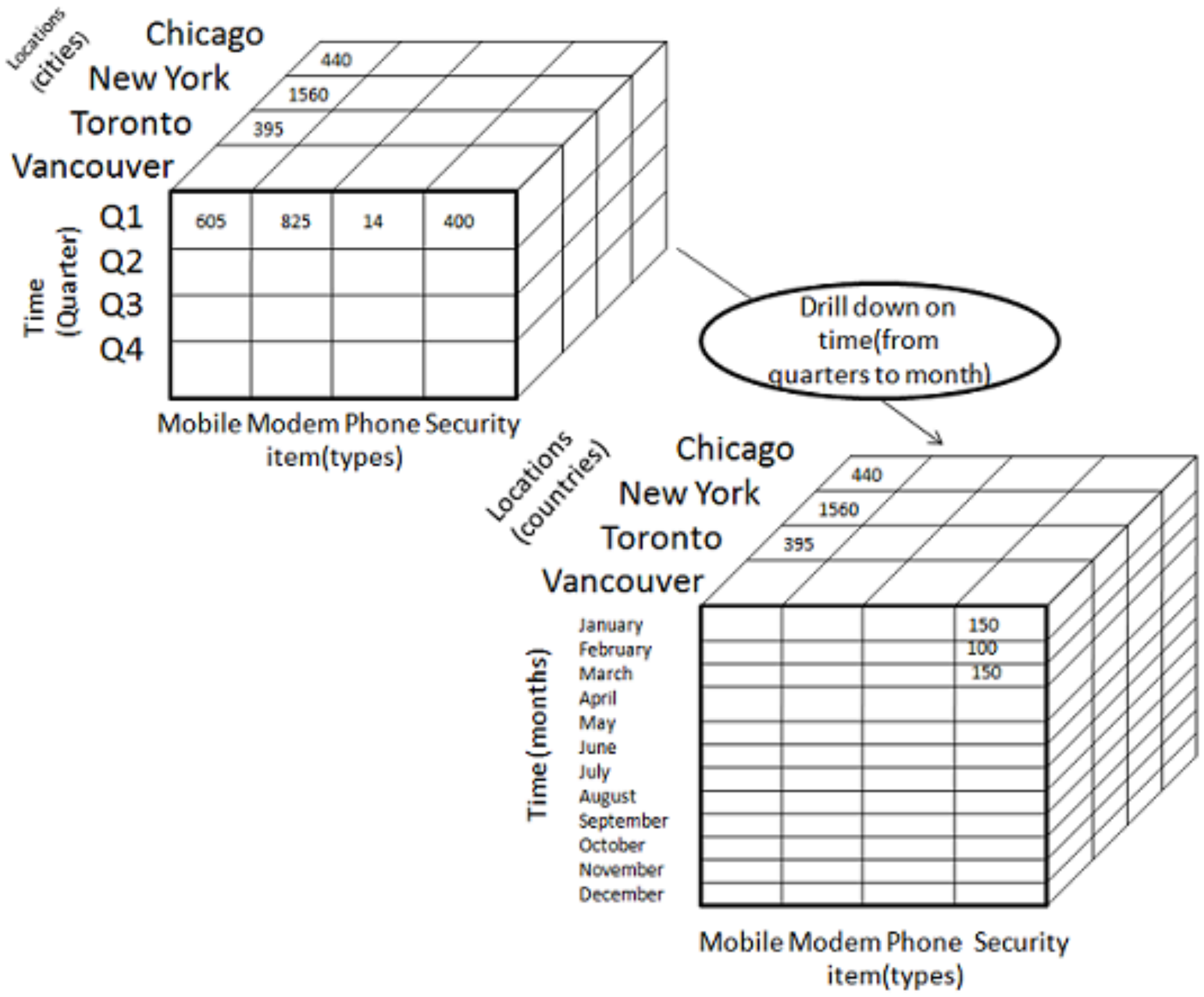
Slicing & Dicing

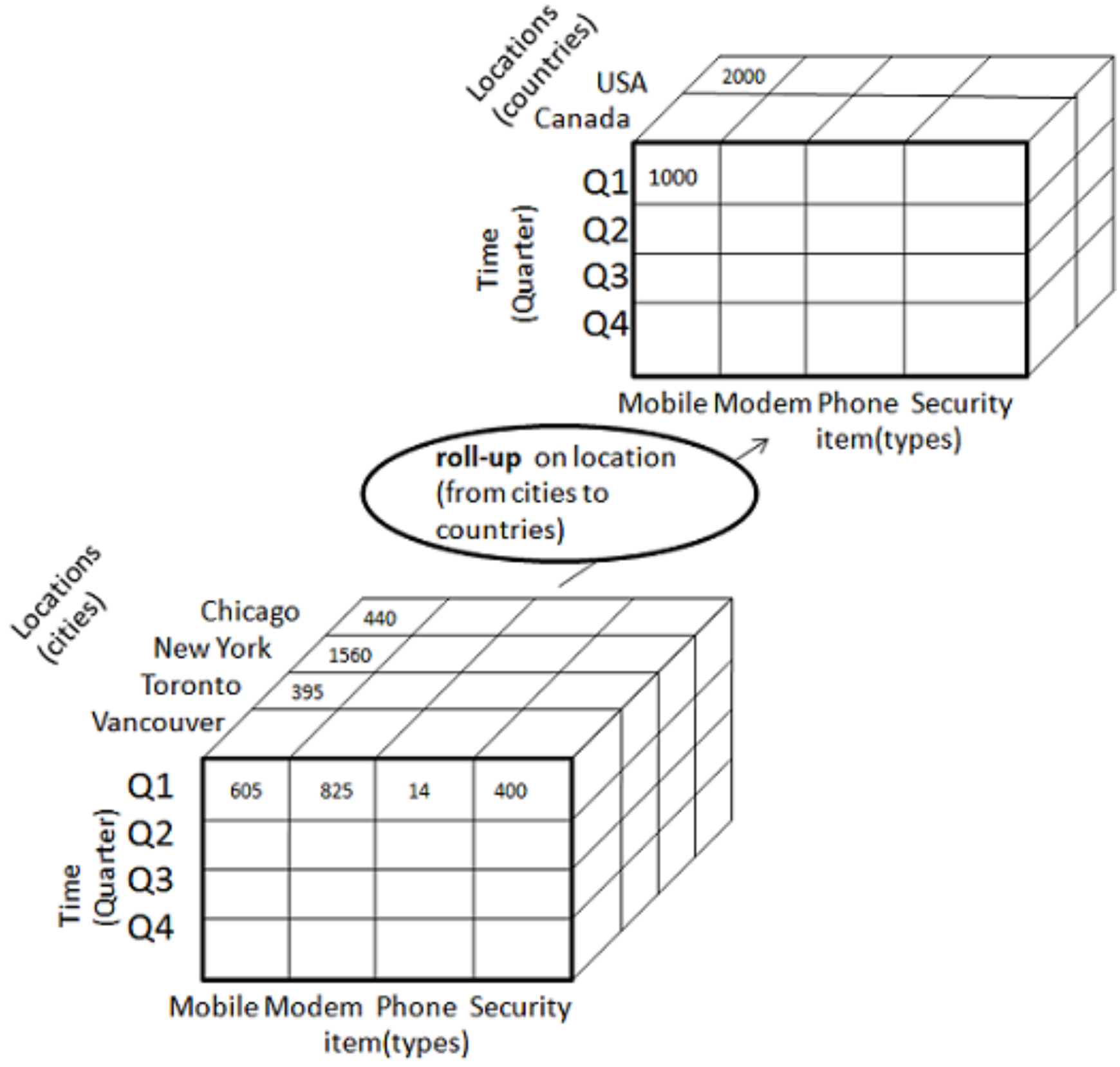
OLAP cube operations

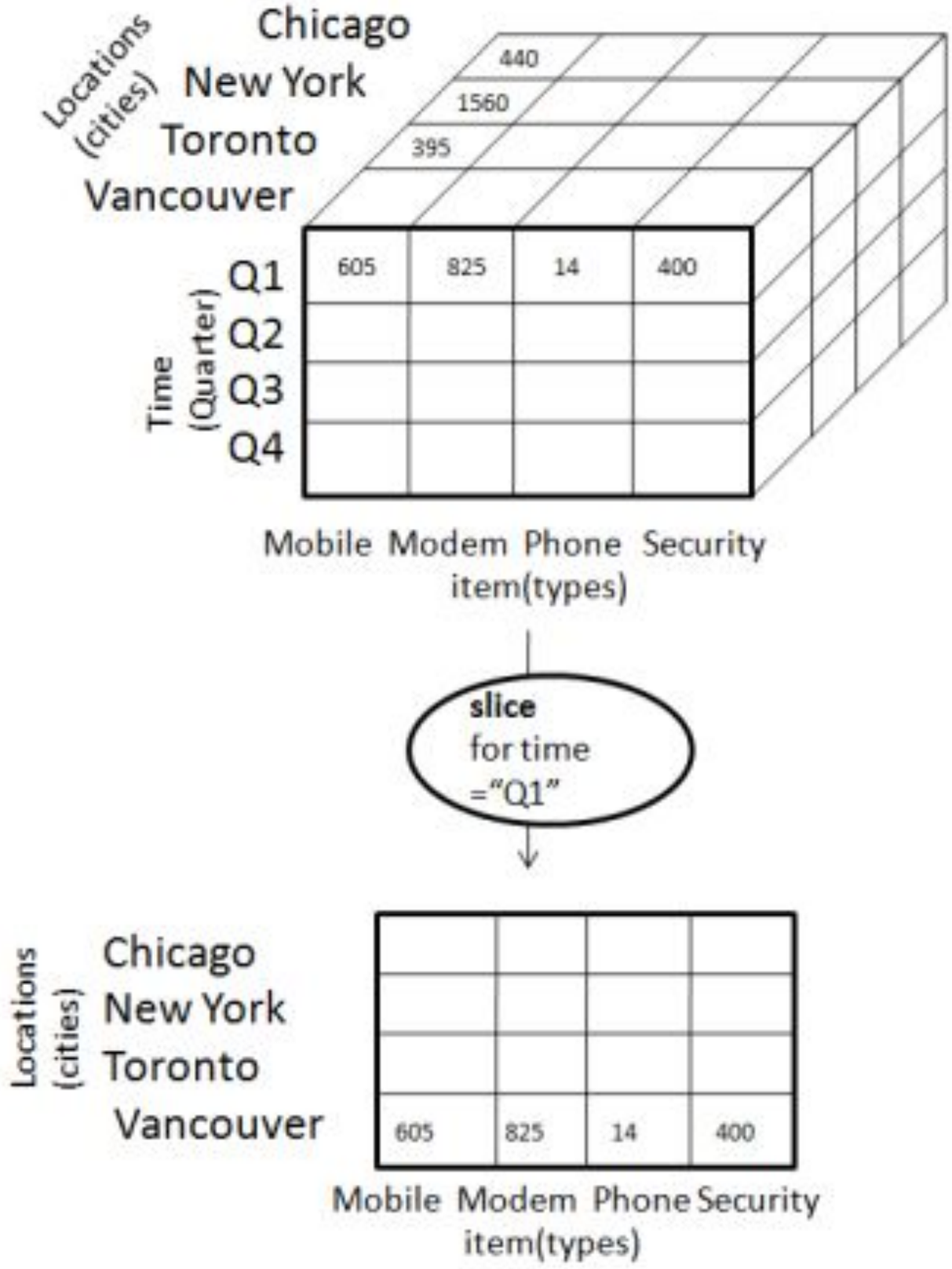


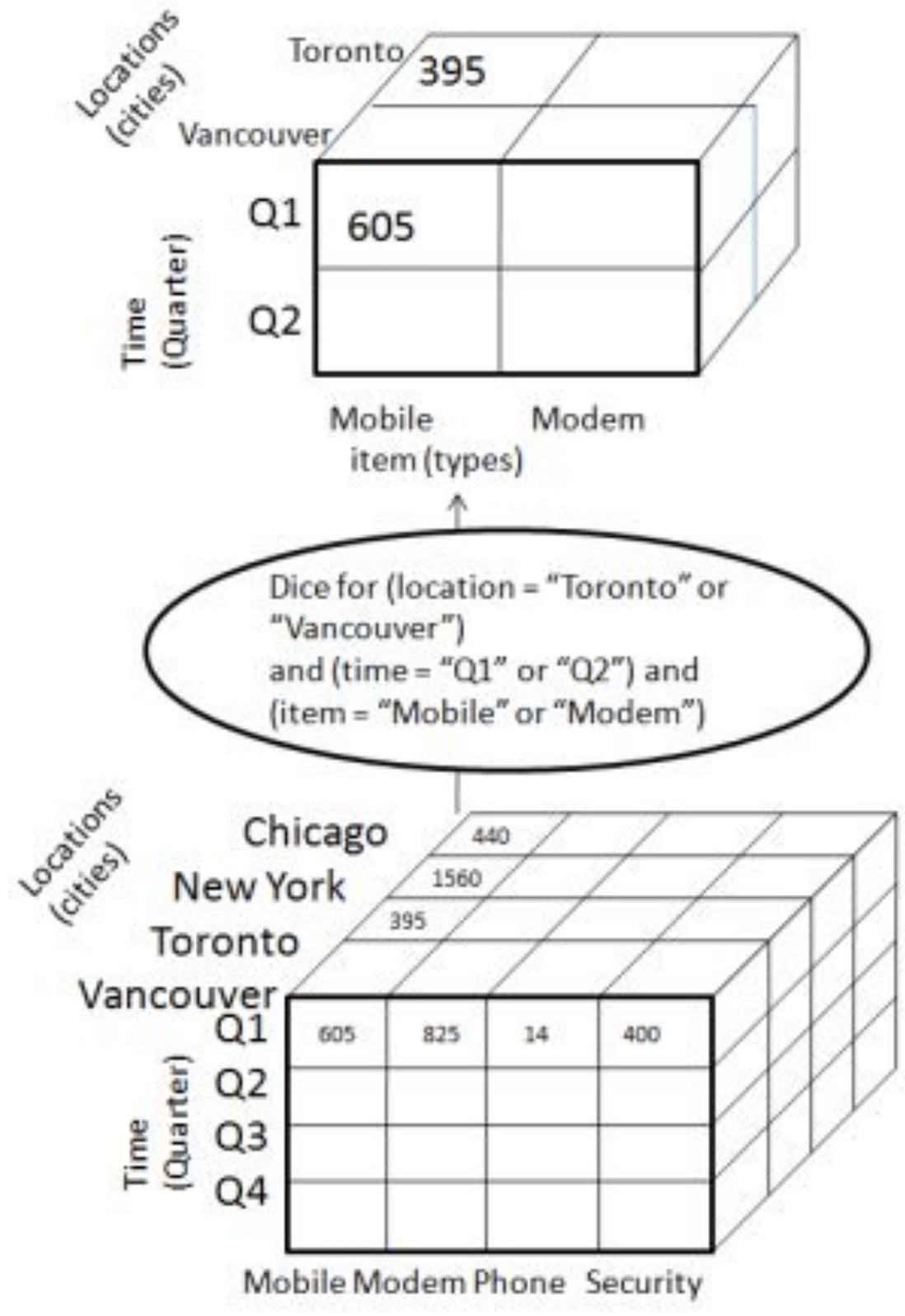


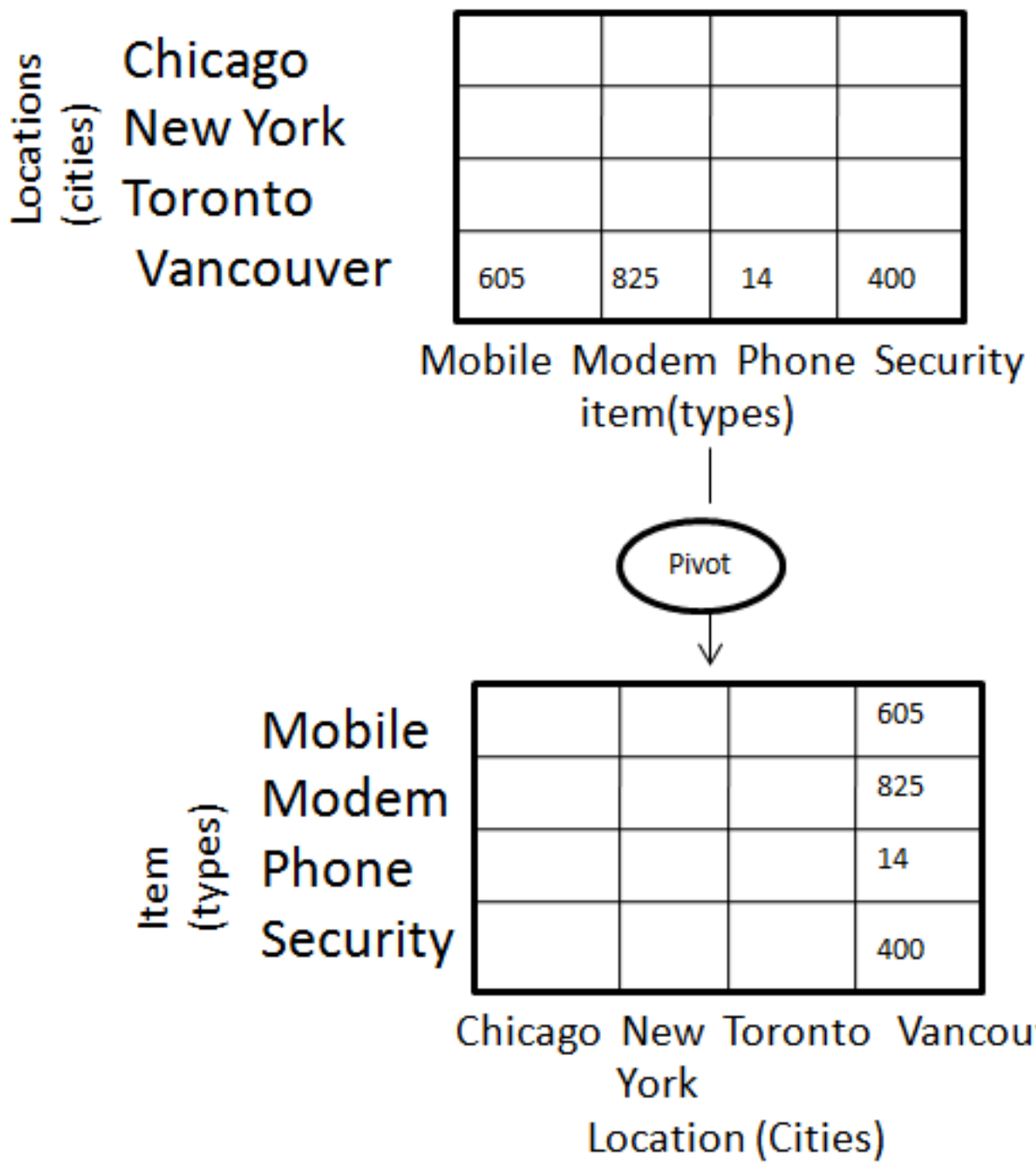

OLAP
CUBE
Quaters
Q1
Q2
Q3
Q4
$21,935649
9975
PC
Book
Shoe
Clothes
Product Category
Territory
U.S.
Australia


Locations (cities)
Chicago
New York
Toronto
Vancouver
440
1560
395
Time (Quarter)
Q1
Q2
Q3
Q4
605
825
14
400
Mobile Modem Phone Security
item(types)
Drill down on time(from quarters to month)
Locations (countries)
Chicago
New York
Toronto
Vancouver
440
1560
395
Time (months)
January
February
March
April
May
June
July
August
September
October
November
December
150
100
150
Mobile Modem Phone Security
item(types)


Locations (countries)
USA
Canada
2000
1000
Q1
Q2
Q3
Q4
Time (Quarter)
Mobile Modem Phone Security
item(types)
roll-up on location (from cities to countries)
Locations (cities)
Chicago
New York
Toronto
Vancouver
440
1560
395
605
825
14
400
Q1
Q2
Q3
Q4
Time (Quarter)
Mobile Modem Phone Security
item(types)


Locations (cities)
Chicago
New York
Toronto
Vancouver
440
1560
395
Time (Quarter)
Q1
Q2
Q3
Q4
605
825
14
400
Mobile
Modem
Phone
Security
item(types)
slice
for time
="Q1"
Locations (cities)
Chicago
New York
Toronto
Vancouver
605
825
14
400
Mobile
Modem
Phone
Security
item(types)


Locations (cities)
Toronto
395
Vancouver
Q1
605
Q2
Time (Quarter)
Mobile
Modem
item (types)
Dice for (location = "Toronto" or "Vancouver")
and (time = "Q1" or "Q2") and
(item = "Mobile" or "Modem")
Locations (cities)
Chicago
440
New York
1560
Toronto
395
Vancouver
Q1
605
825
14
400
Q2
Q3
Q4
Time (Quarter)
Mobile
Modem
Phone
Security

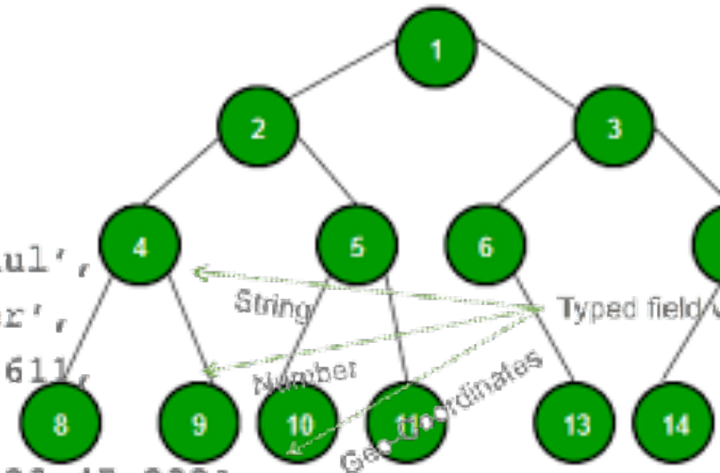

Locations (cities)
Chicago
New York
Toronto
Vancouver
605
825
14
400
Mobile
Modem
Phone
Security
item(types)
Pivot
Item (types)
Mobile
Modem
Phone
Security
605
825
14
400
Chicago
New York
Toronto
Vancouver
Location (Cities)

Rdbms (Referential Integrity)

tblPerson			
ID	Name	Email	GenderID
1	Jade	j@j.com	2
2	Mary	m@m.com	3
3	Martin	ma@ma.com	1
4	Rob	r@r.com	NULL
5	May	may@may.com	2
6	Kristy	k@k.com	NULL

tblGender	
ID	Gender
1	Male
2	Female
3	Unknown

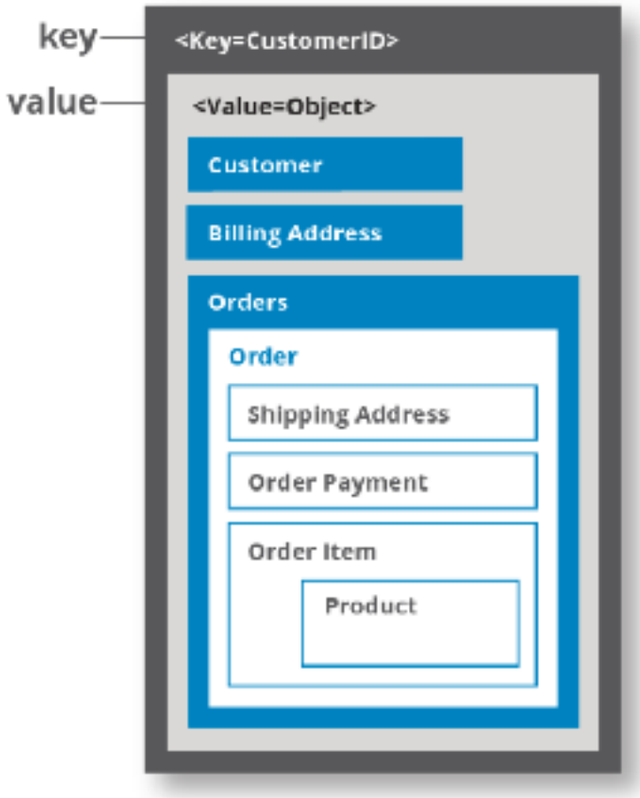
Fields

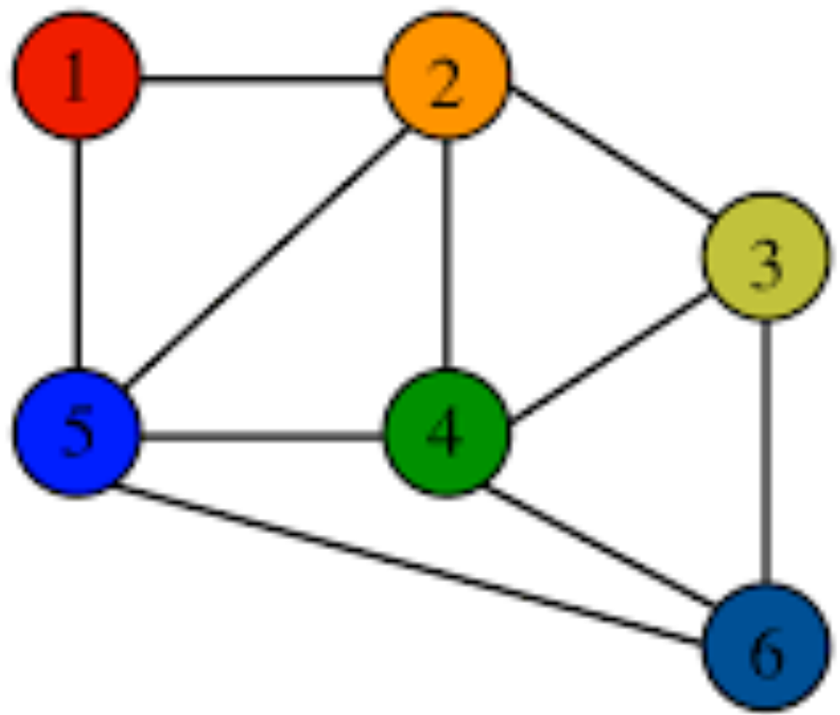



```
{
  first_name: 'Paul',
  surname: 'Miller',
  cell: 447557505611,
  city: 'London',
  location: [45.123,47.232],
  Profession: ['banking', 'finance', 'trader'],
  cars: [
    { model: 'Bentley',
      year: 1973,
      value: 100000, … },
    { model: 'Rolls Royce',
      year: 1965,
      value: 330000, … }
  ]
}
```

Fields can contain an array of documents

key	value
123	123 Main St.
126	(805) 477-3900





Wide Column

- Spotify uses Cassandra to store user profile attributes and metadata about artists, songs, etc. for better personalization
- Facebook initially built its revamped Messages on top of HBase, but is now also used for other Facebook services like the Nearby Friends feature and search indexing
- Outbrain uses Cassandra to serve over 190 billion personalized content recommendations each month

Files

Databases (Cloud / On Premises)

Cloud Sources

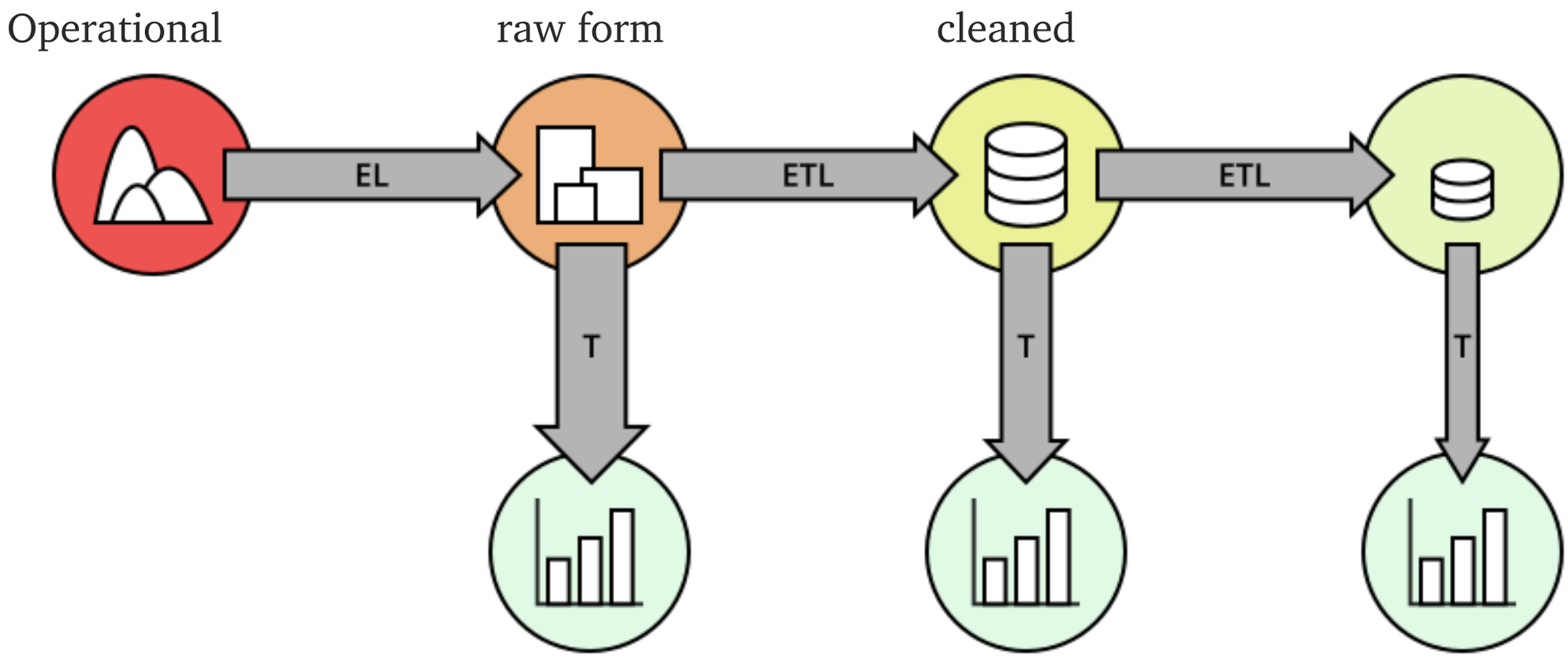
Audio/ Video/Documents

Data Lake

WebApps, Sensors, IoT

Social Feeds


Data Lake
Data Warehouse
Data Mart


Operational
raw form
cleaned
EL
ETL
ETL
T
T
T

Data Source Tier

Structured

Database,
Data warehouse,
Lob App logs

unstructured

Images,
Audio,
Word doc,
Spread Sheet

semi-structured

XMl,
Html,
Json,
Emails

Landing Zone
(immutable)

Raw

handling missing values,
filtering outliers,
& Validated data

Stage X

Stage Y

Highly refined views
of the data

Curated

Data Lake
Azure Blob
AWS S3
Azure Data Lake

Data WareHouse / Mart
Aws Redshift
Azure Synapse Analytics

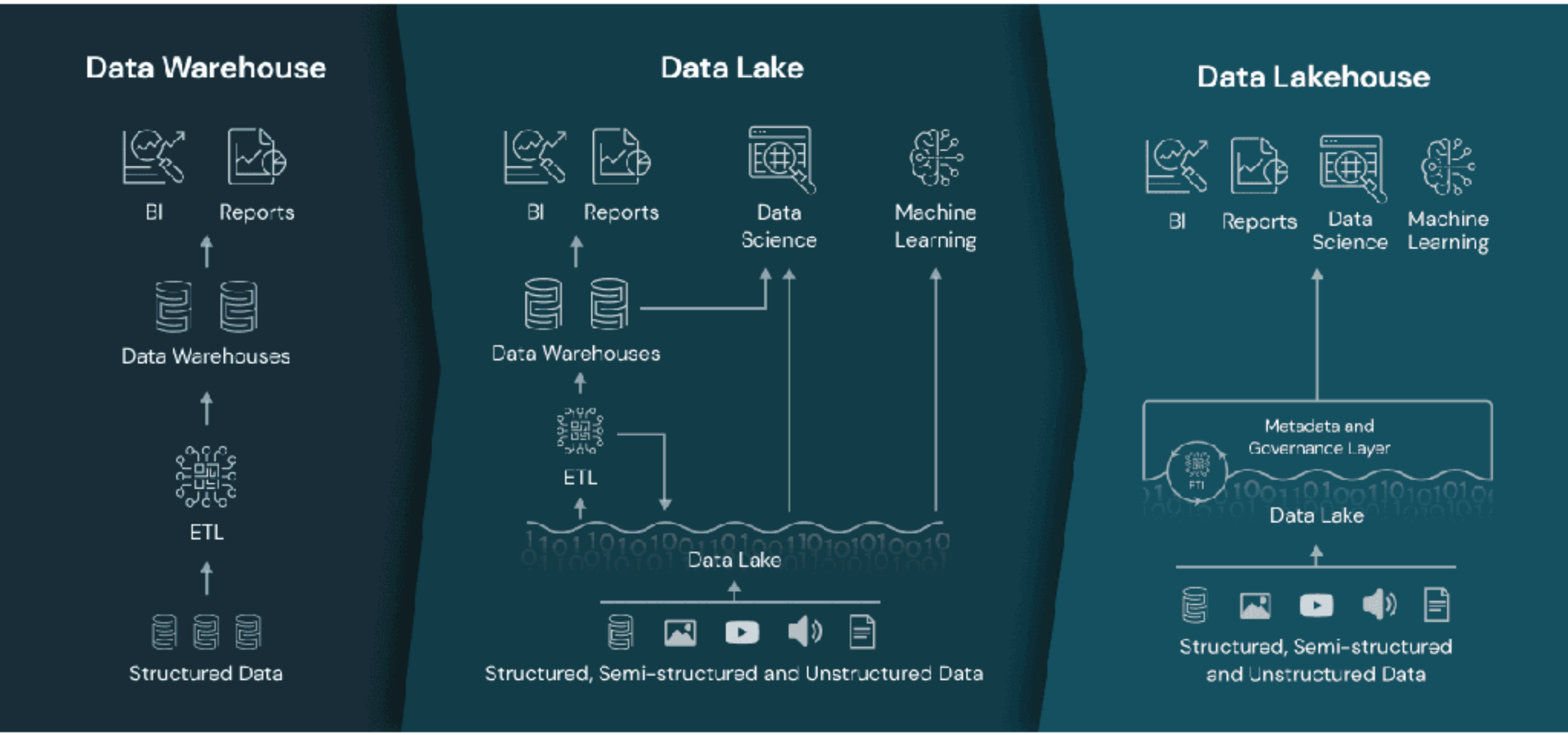
Insight/Analytics Tier

BI Tools

Applications

Automations

QuickSight
Power BI


Data Warehouse
BI
Reports
Data Warehouses
ETL
Structured Data
Data Lake
BI
Reports
Data Science
Machine Learning
Data Warehouses
ETL
Data Lake
Structured, Semi-structured and Unstructured Data
Data Lakehouse
BI
Reports
Data Science
Machine Learning
Metadata and Governance Layer
ETL
Data Lake
Structured, Semi-structured and Unstructured Data

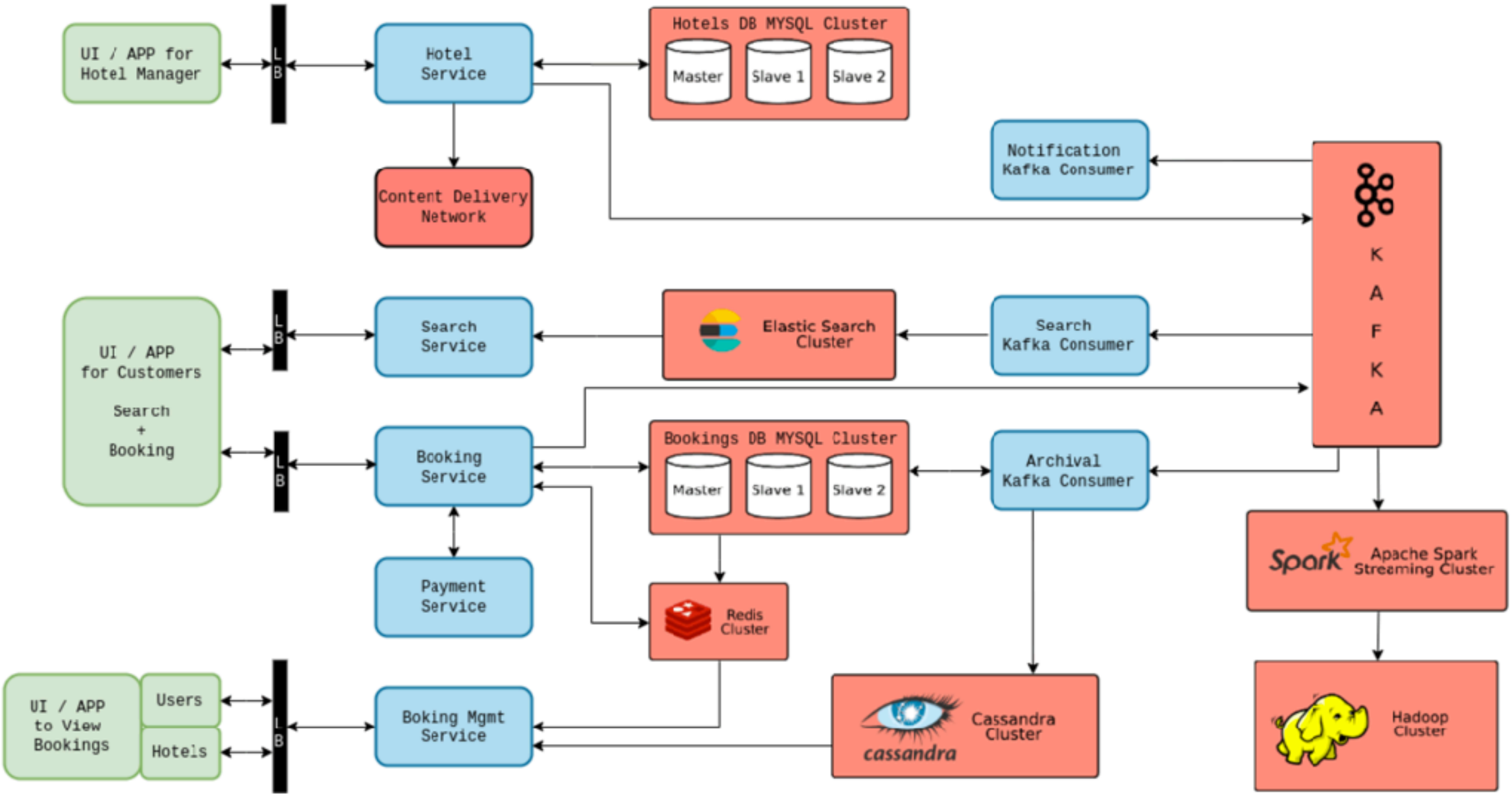
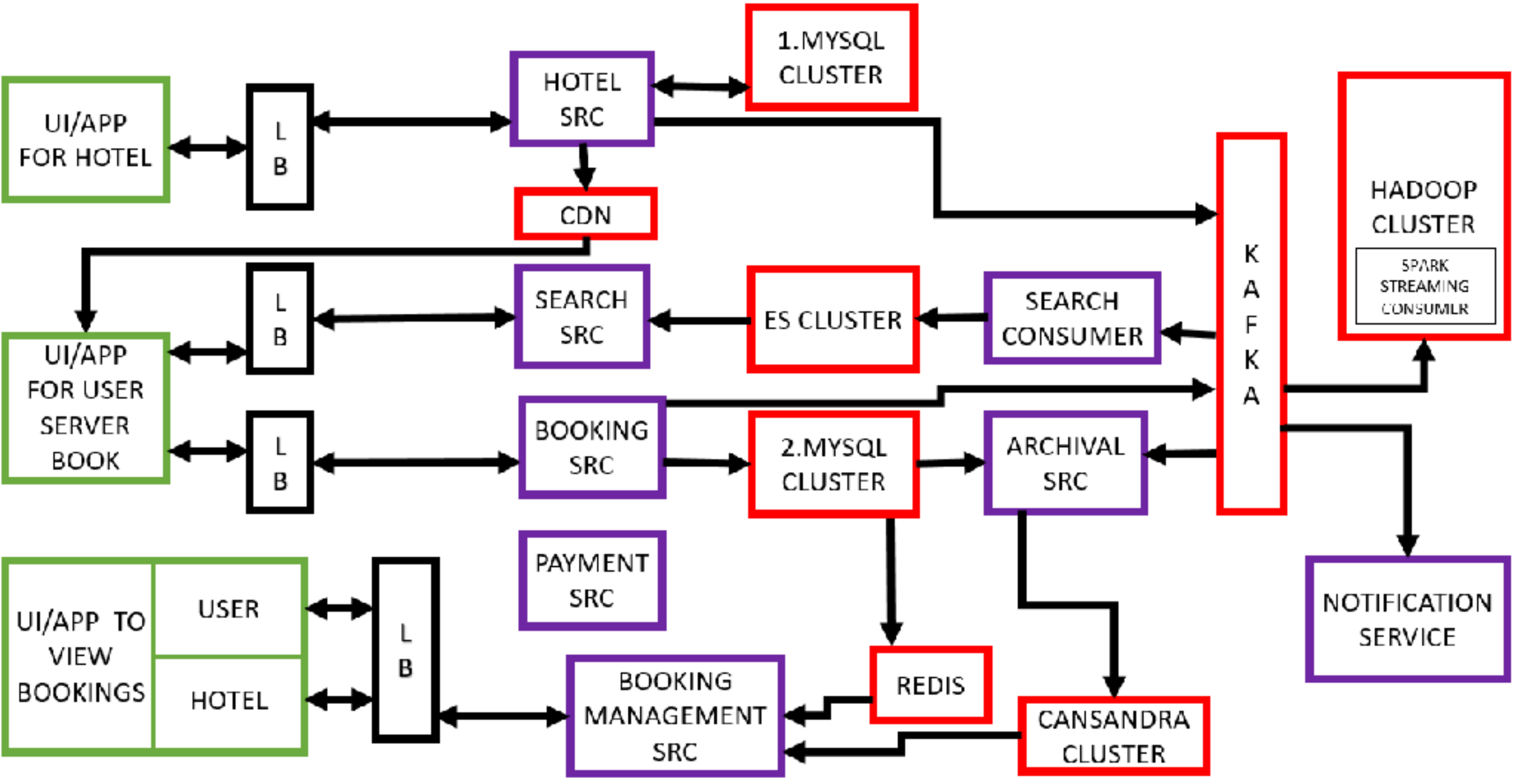


photo by sandeepkaul on github


UI/APP FOR HOTEL
L B
HOTEL SRC
1.MYSQL CLUSTER
CDN
HADOOP CLUSTER
SPARK STREAMING CONSUMER
K A F K A
UI/APP FOR USER SERVER BOOK
L B
SEARCH SRC
ES CLUSTER
SEARCH CONSUMER
L B
BOOKING SRC
2.MYSQL CLUSTER
ARCHIVAL SRC
PAYMENT SRC
NOTIFICATION SERVICE
UI/APP TO VIEW BOOKINGS
USER
HOTEL
L B
BOOKING MANAGEMENT SRC
REDIS
CANSANDRA CLUSTER

Amazon/Flipkart System Design

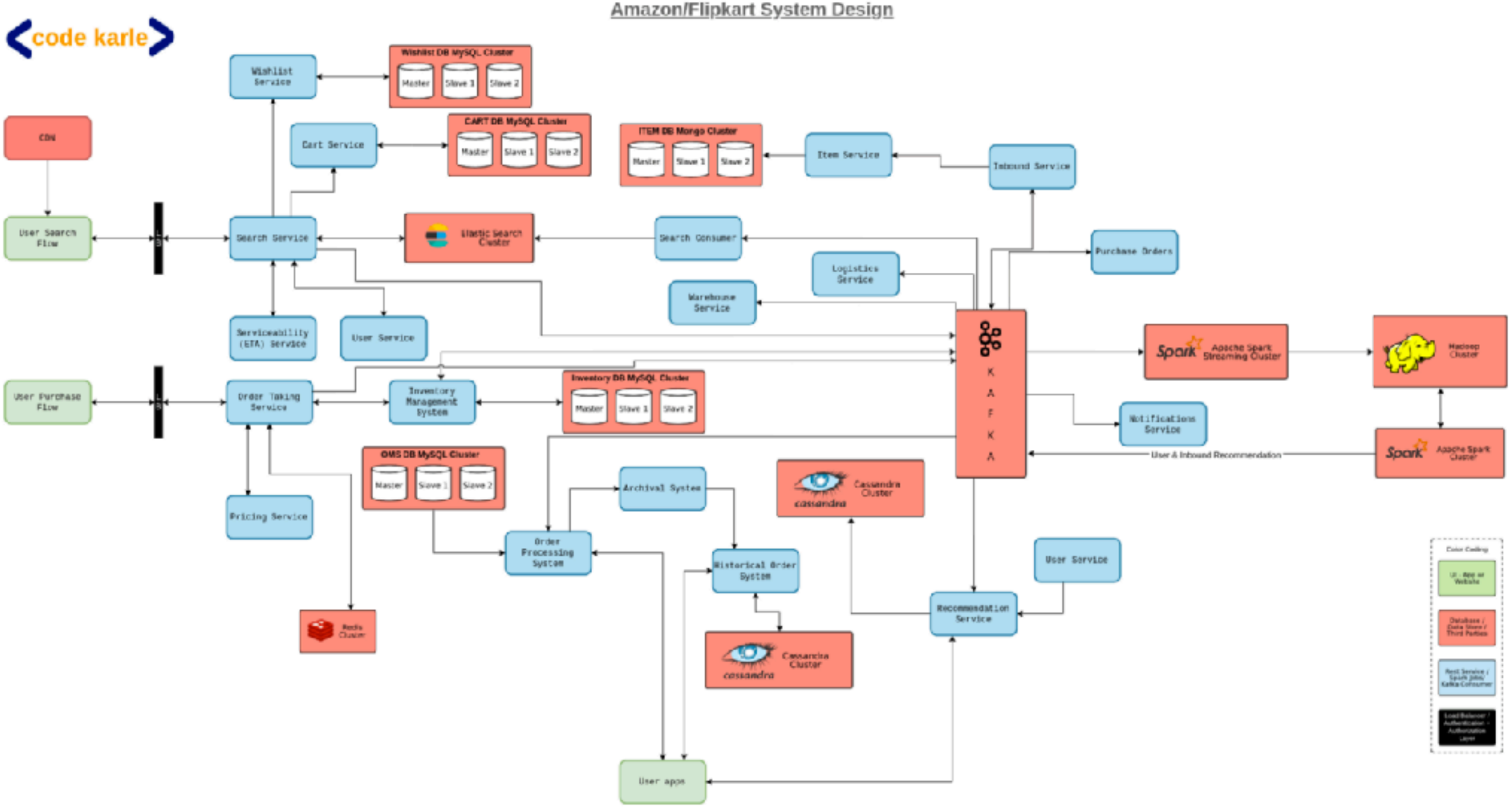
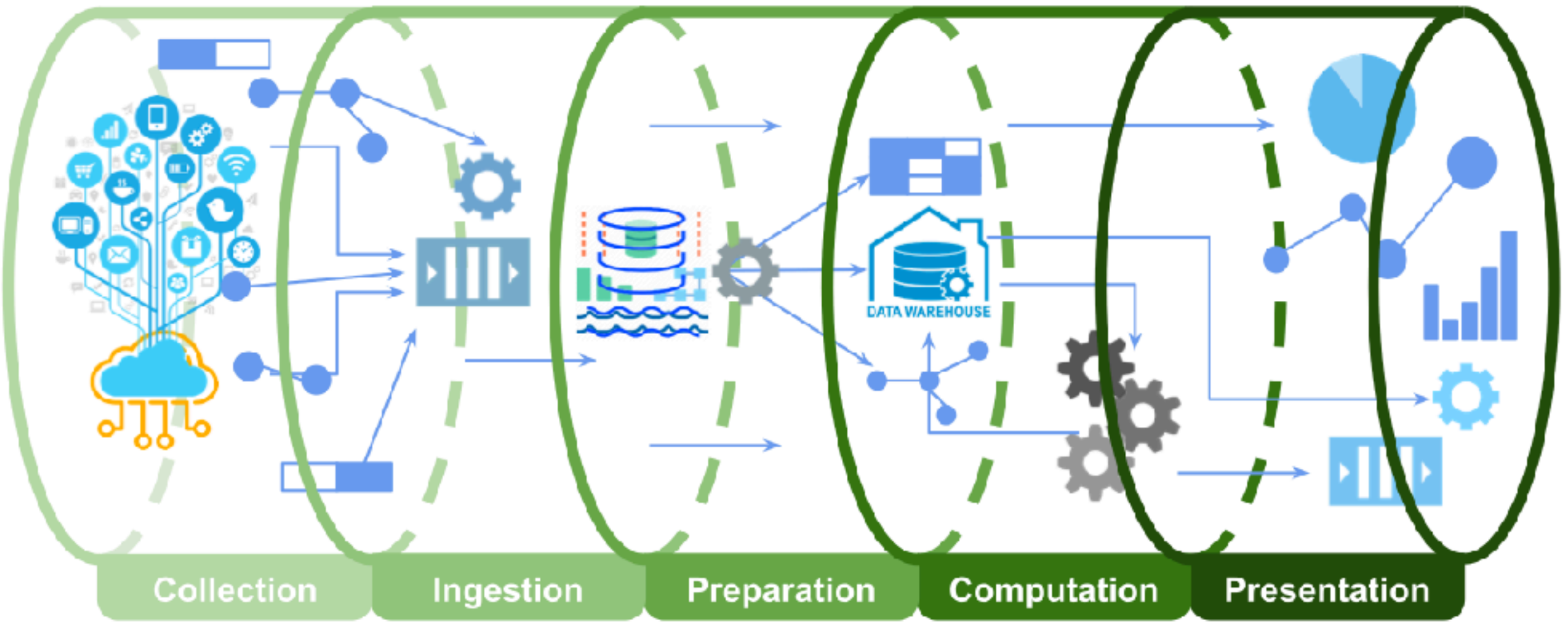
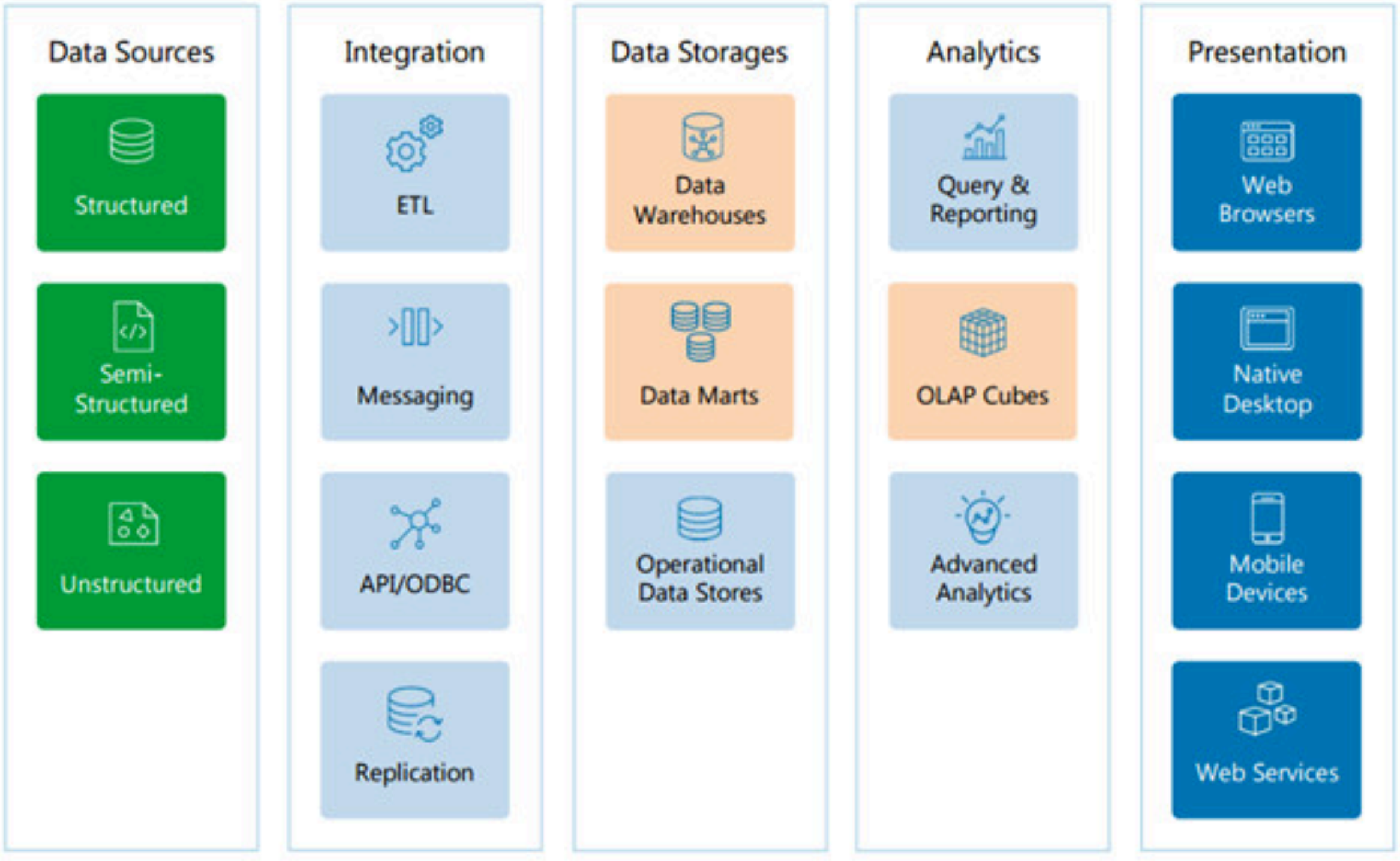


photo by sandeepkaul on github

Data Pipeline

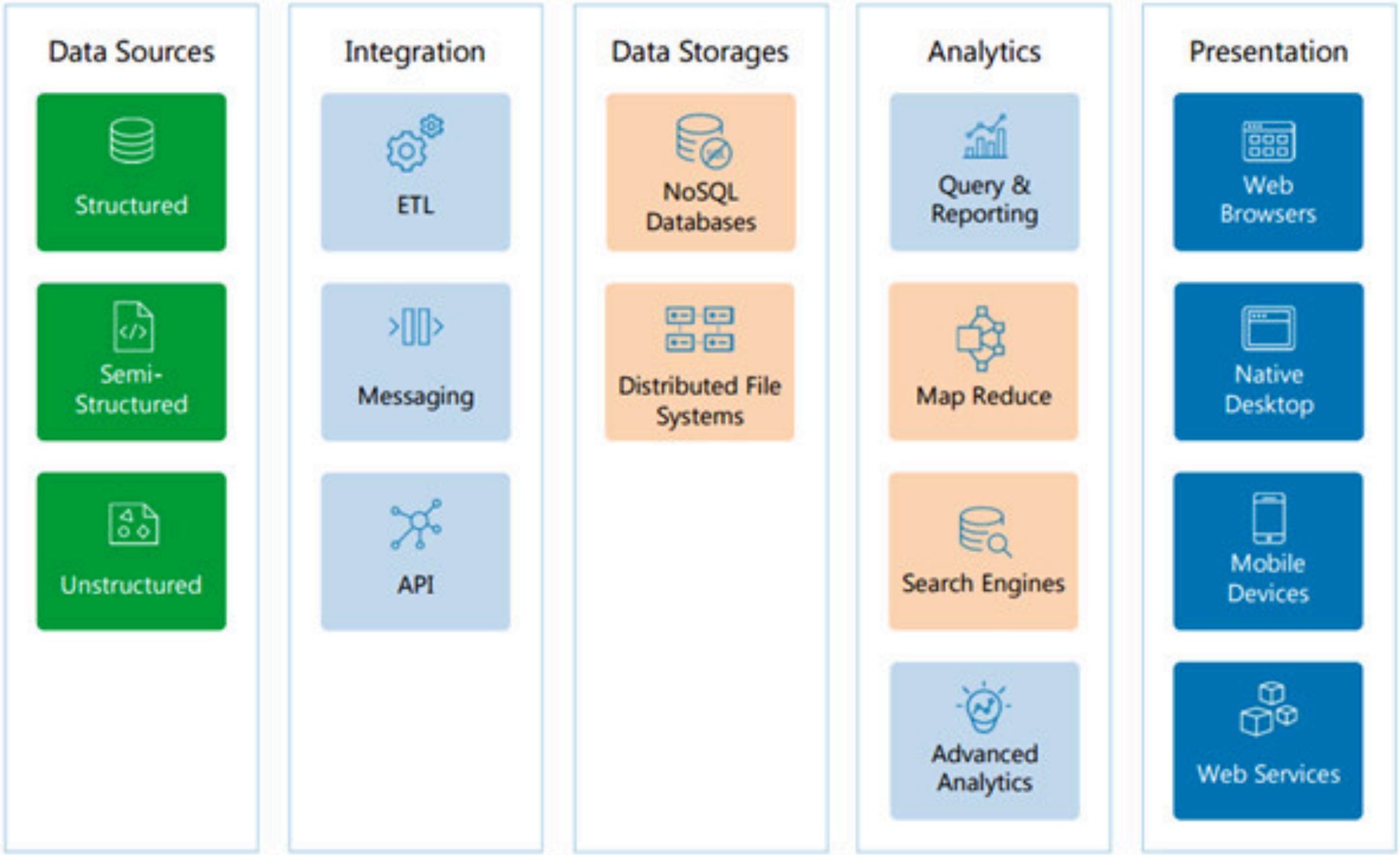


Relational Reference Architecture



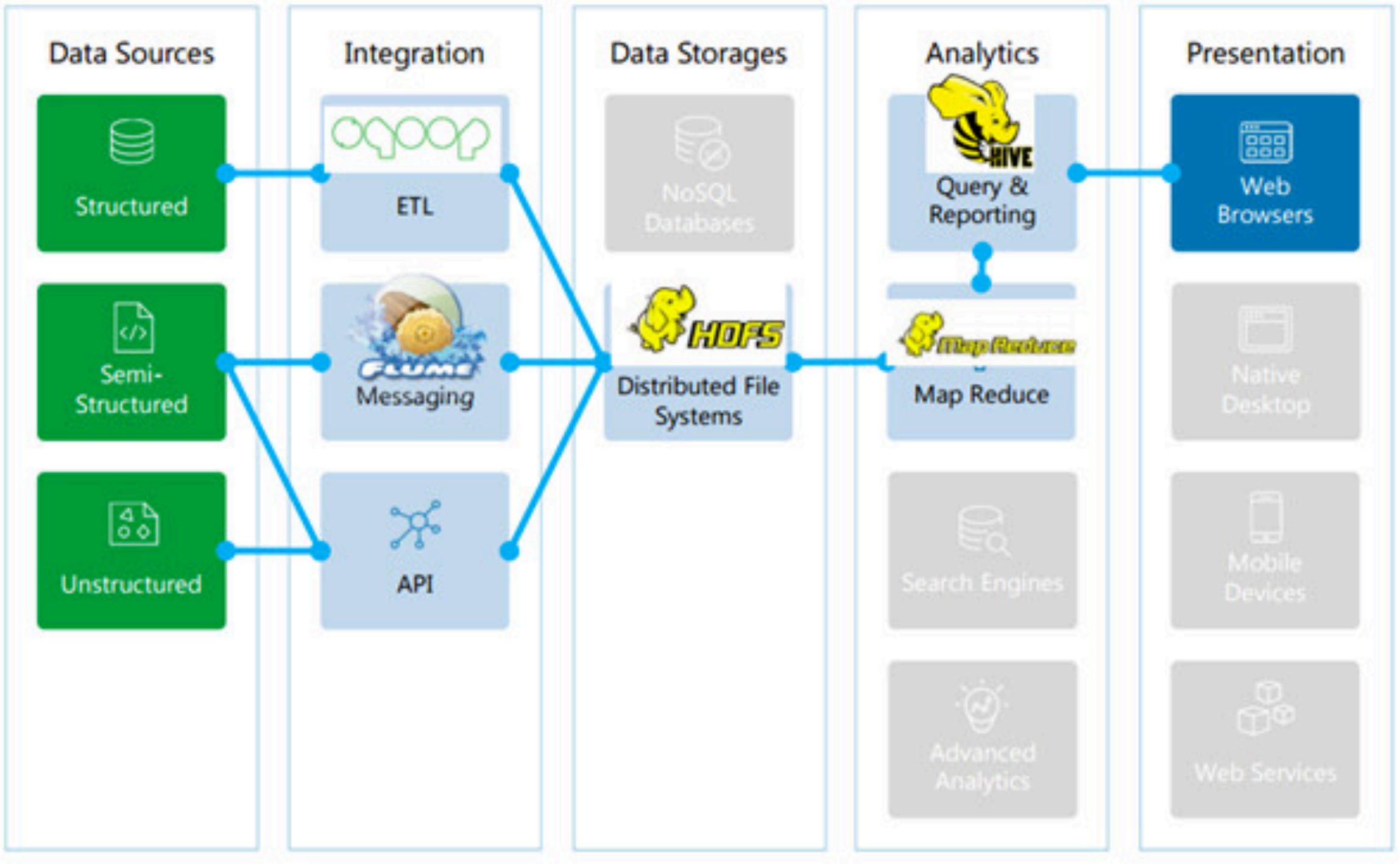
components with pink blocks cannot handle big data challenges.

Non-Relational Reference Architecture



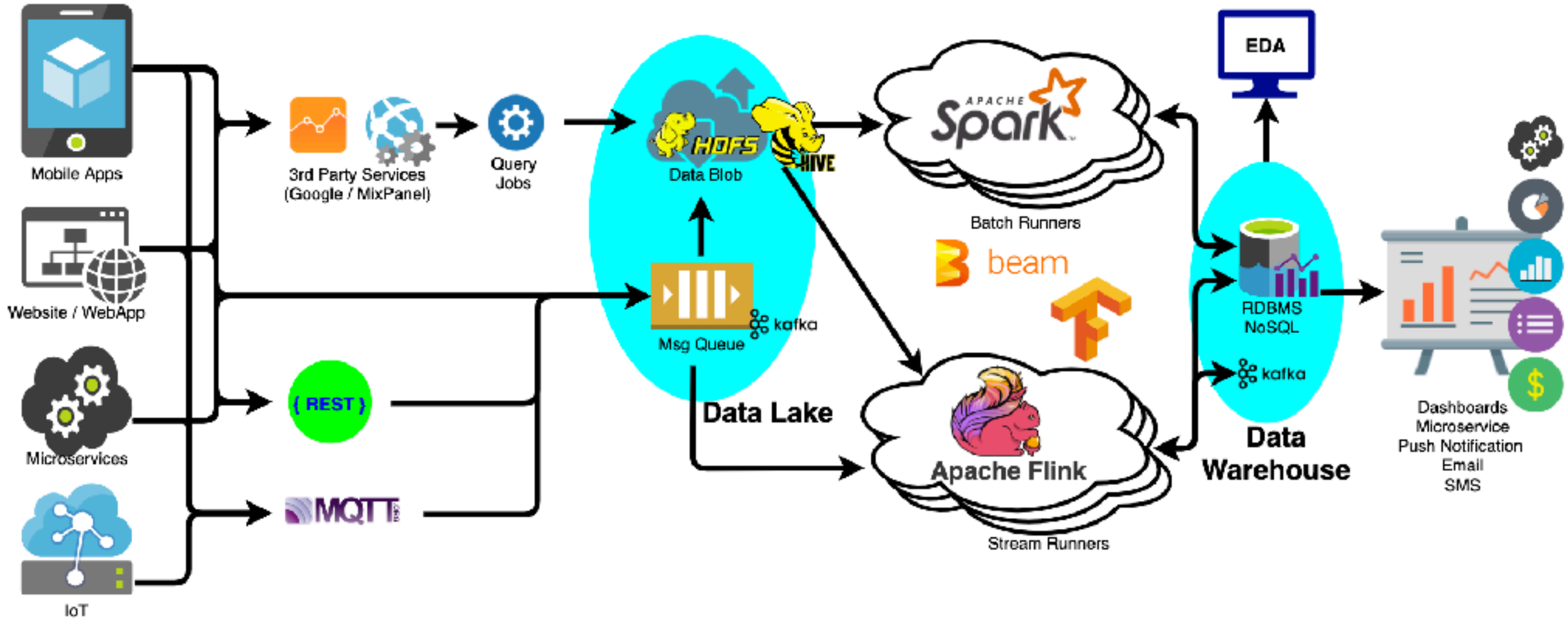
pink blocks cannot handle big data challenges completely.

Big Data Reference Architecture



Hadoop based Big Data Architecture which can be handle few core components of big data challenges but not all

Open Source Data analytics Architecture

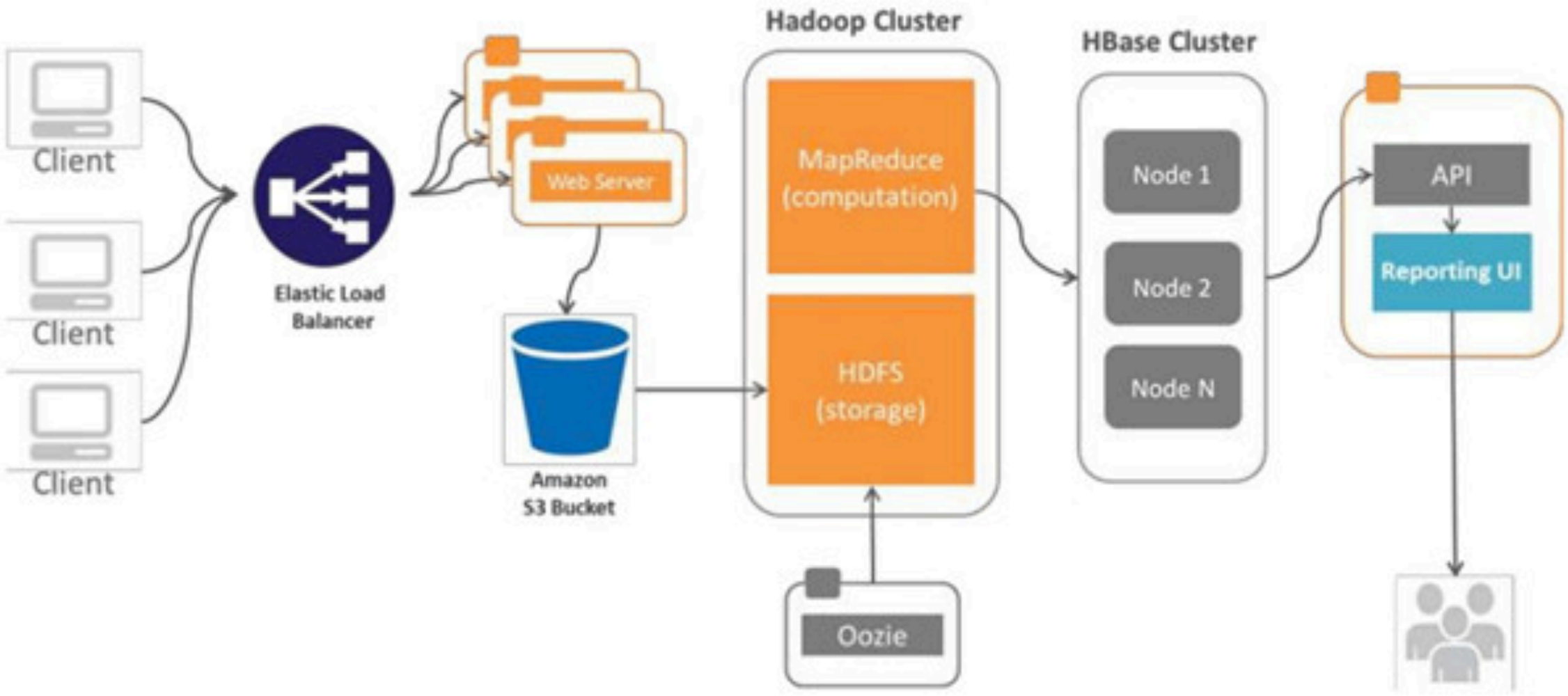


AWS Data analytics Architecture

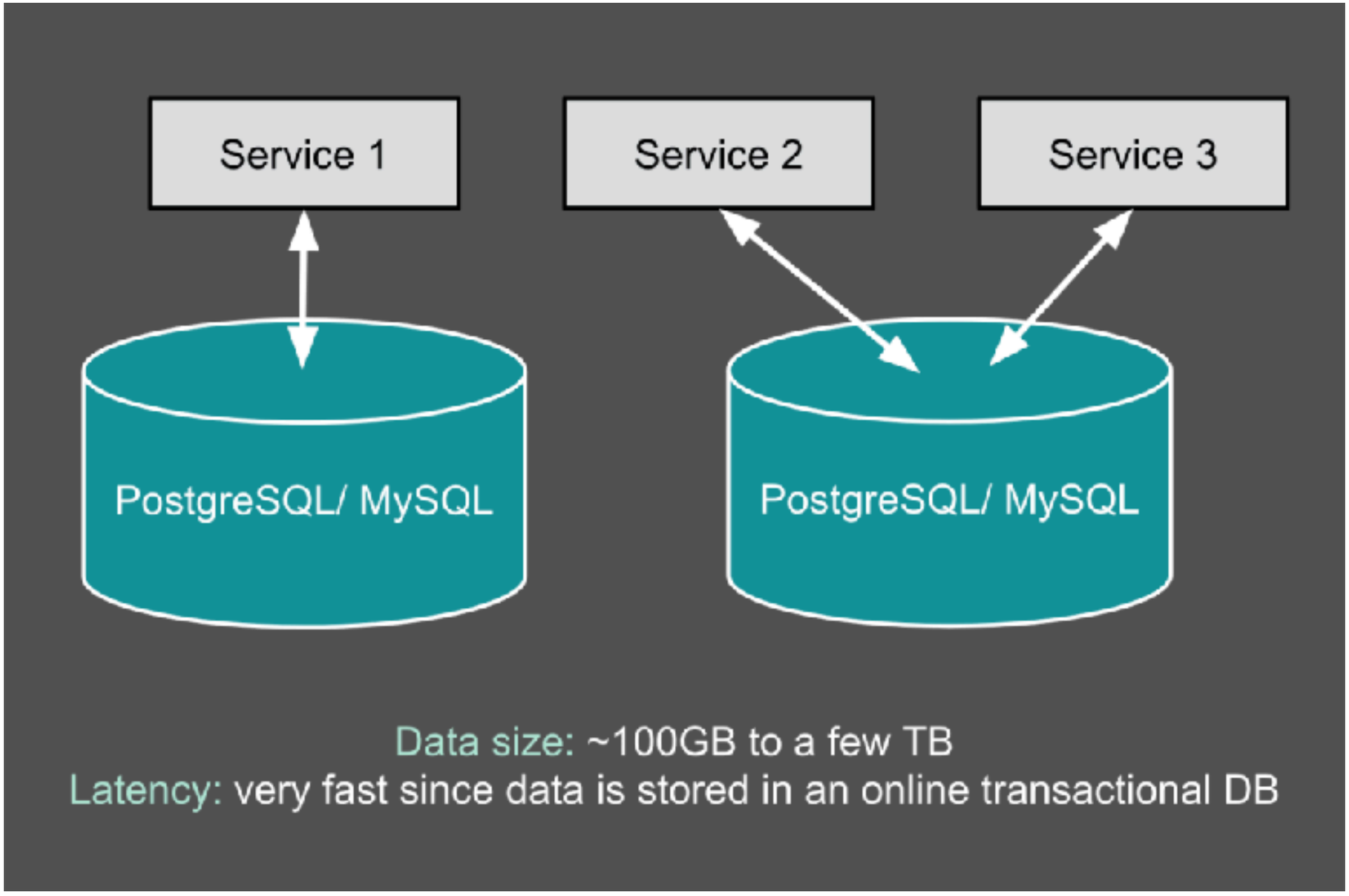
Solution Architecture

Technologies:

- Amazon S3
- Flume
- Hadoop/HDFS, MapReduce
- HBase
- Oozie
- Hive

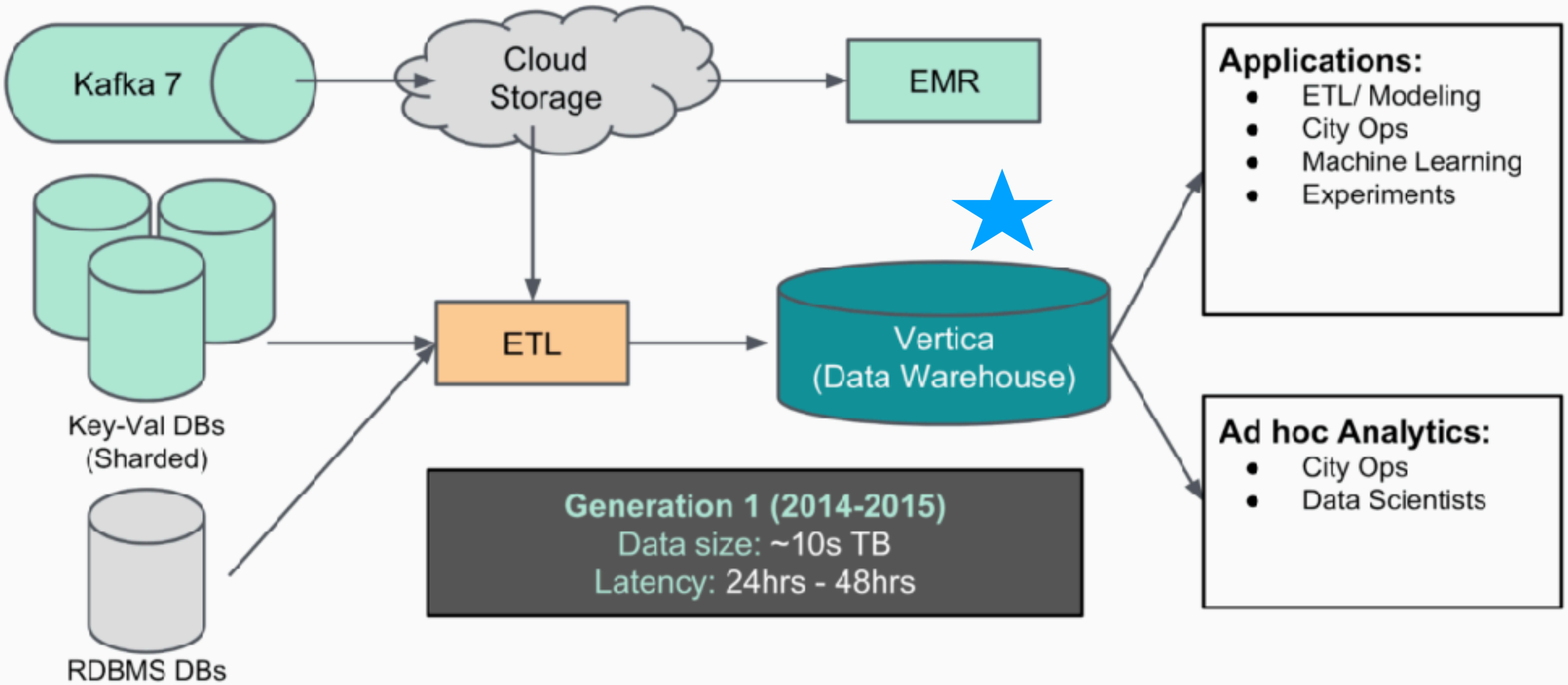


Generation 1: The beginning of Big Data at Uber (Before 2014)

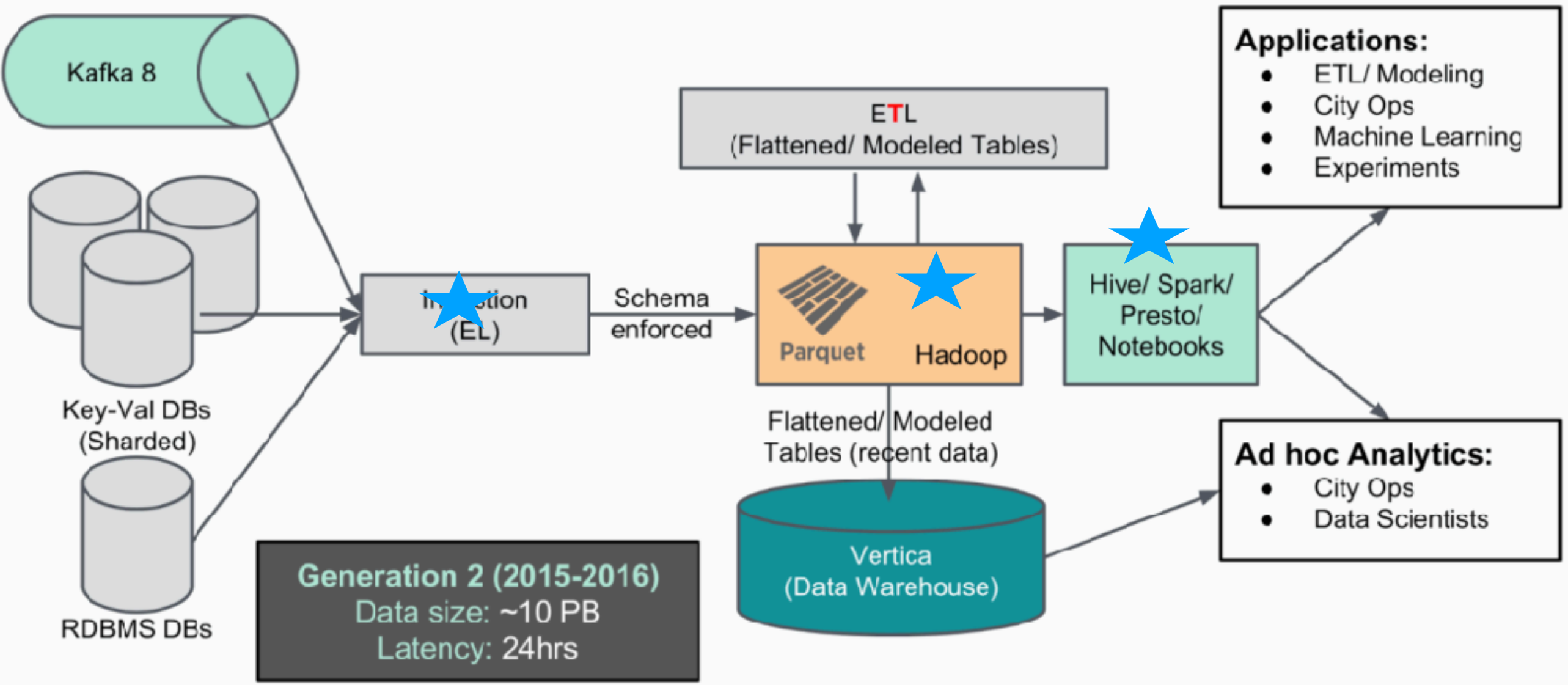


Data was scattered across different OLTP databases. it was left to users to write their own code if they needed to combine data from different databases.

Generation 1 (2014-2015) - The beginning of Big Data at Uber



The first generation of our analytical data warehouse focused on aggregating all of Uber's data in one place as well as streamlining data access. developed multiple ad hoc ETL (Extract, Transform, and Load) jobs that copied data from different sources (i.e. AWS S3, OLTP databases, service logs, etc.) into Vertica.



all raw data was ingested from different online data stores only once and with no transformation during ingestion.

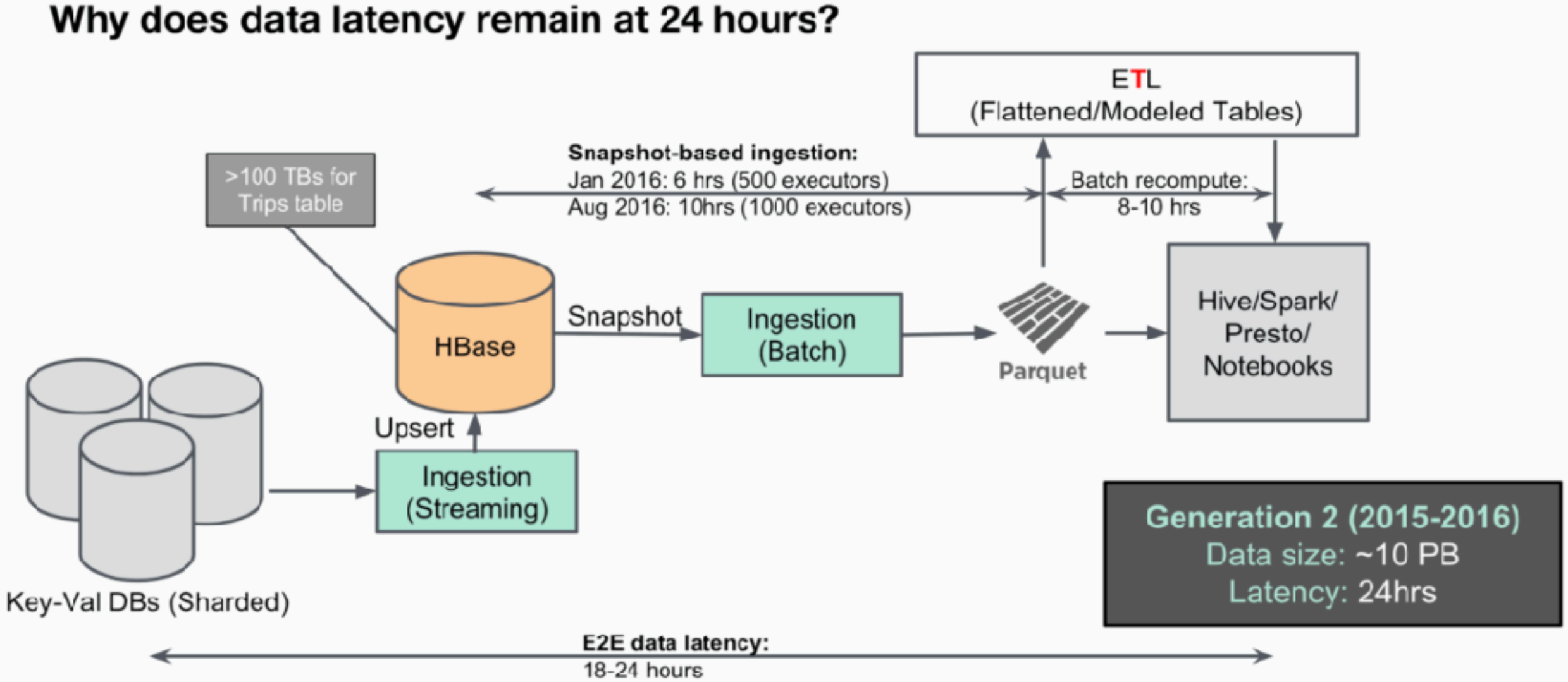
- New data was only accessible to users once every 24 hours, which was too slow to make real-time decisions.

- The ingestion jobs had to return to the source datastore, create a new snapshot, and ingest or convert the entire dataset into consumable, columnar Parquet files during every run. With our data stores growing, these jobs could take over twenty hours with over 1,000 Spark executors to run.

- While only over 100 gigabytes of new data was added every day for each table, each run of the ingestion job had to convert the entire, over 100 terabyte dataset for that specific table.

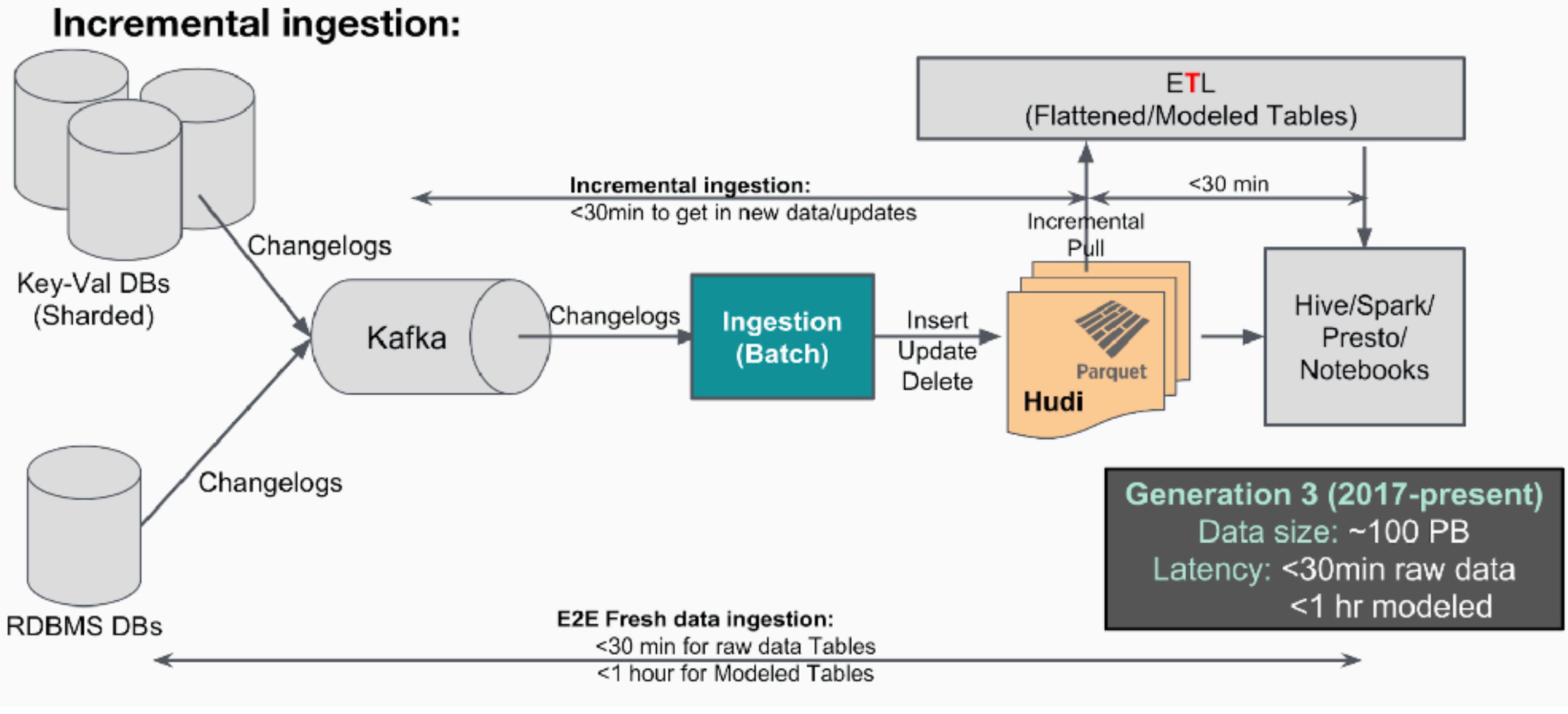
- Since HDFS and Parquet do not support data updates, all ingestion jobs needed to create new snapshots from the updated source data, ingest the new snapshot into Hadoop, convert it into Parquet format, and then swap the output tables to view the new data.

Generation 2 (2015-2016) - The arrival of Hadoop

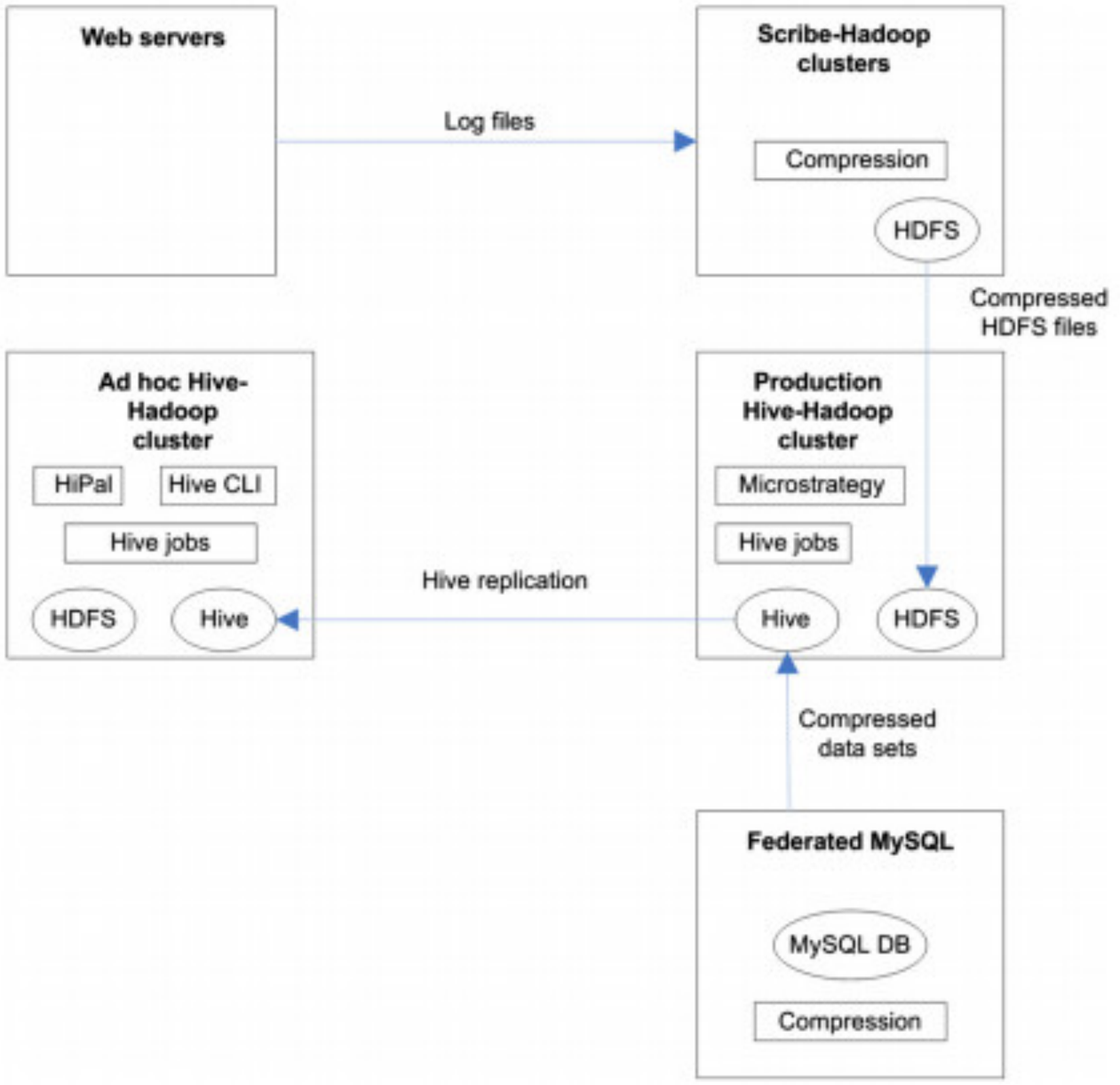


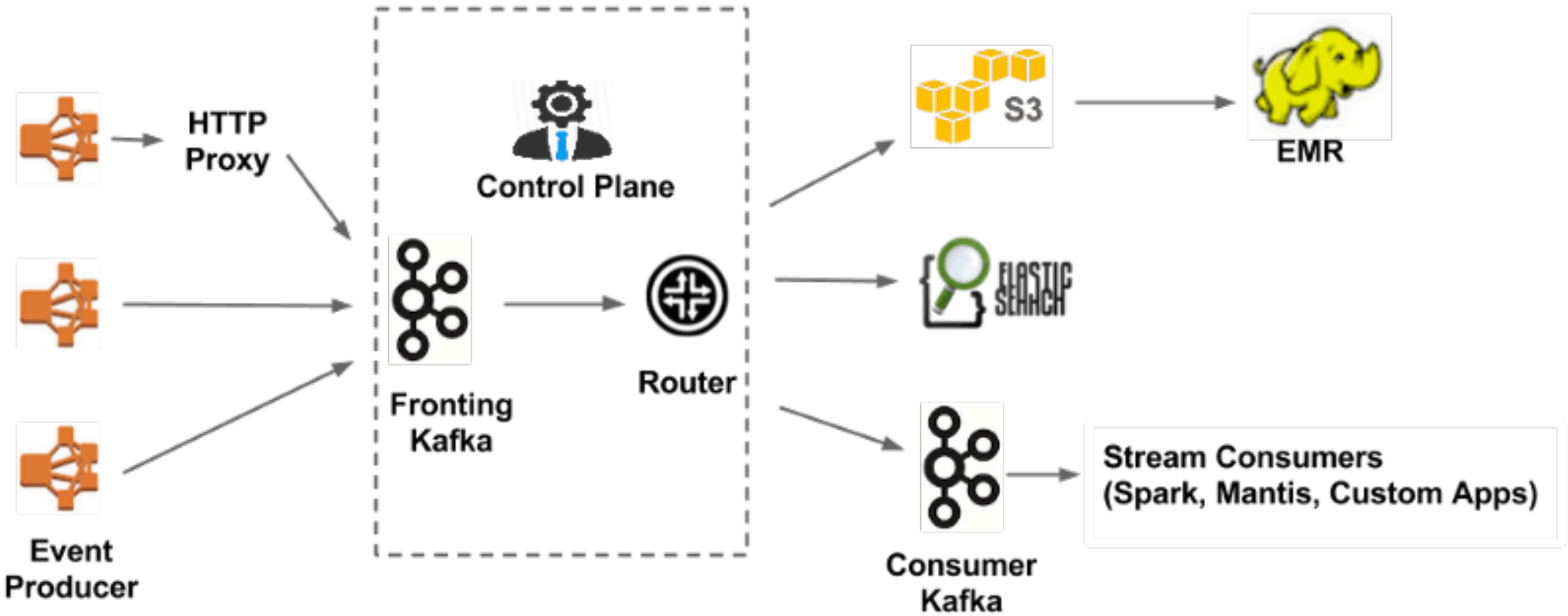
While Hadoop enabled the storage of several petabytes of data in our Big Data platform, the latency for new data was still over one day, a lag due to the snapshot-based ingestion of large, upstream source tables that take several hours to process.

Generation 3 (2017-present) - Let's rebuild for long term



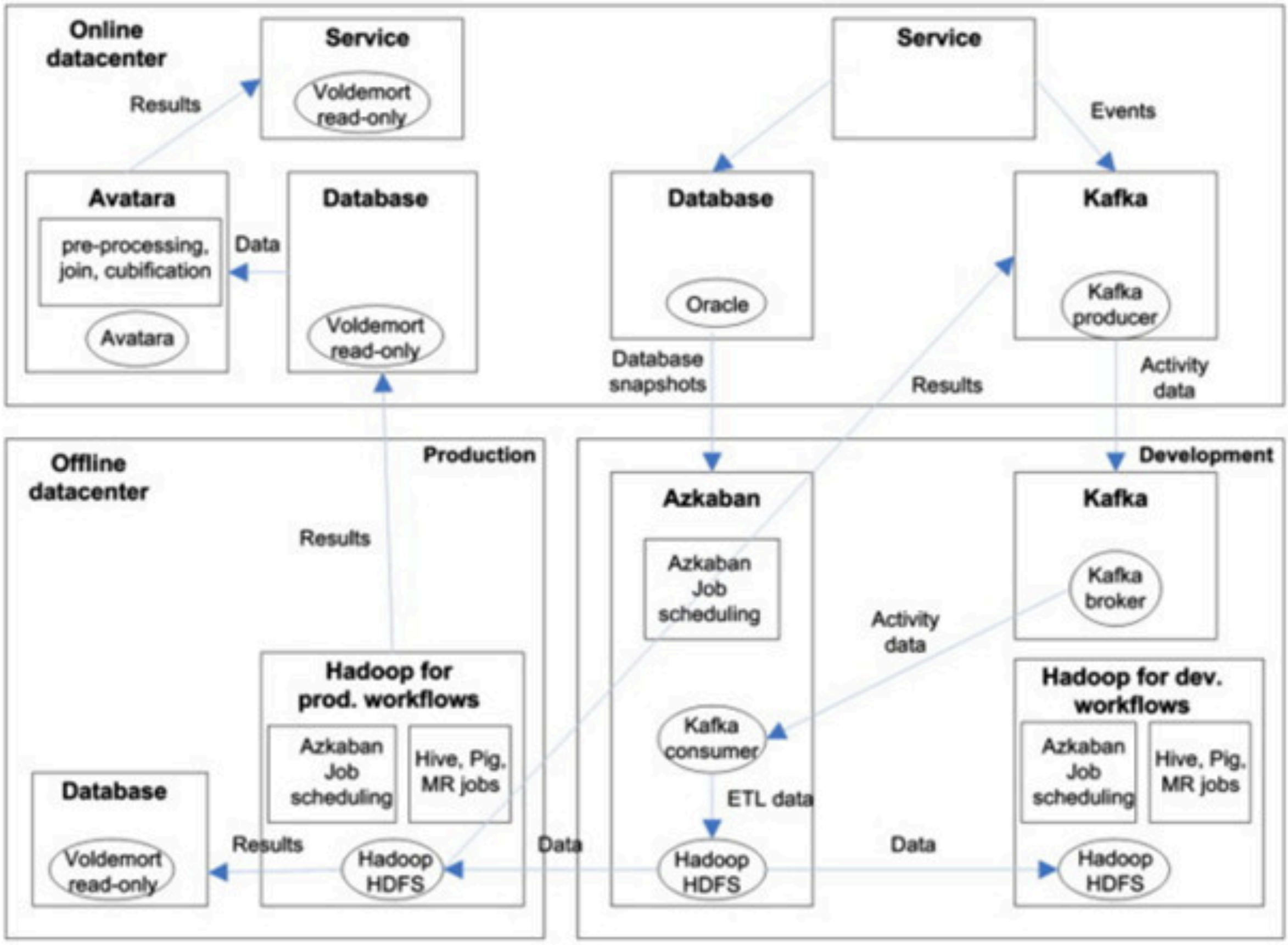
Data analytics Architecture adopted by Facebook



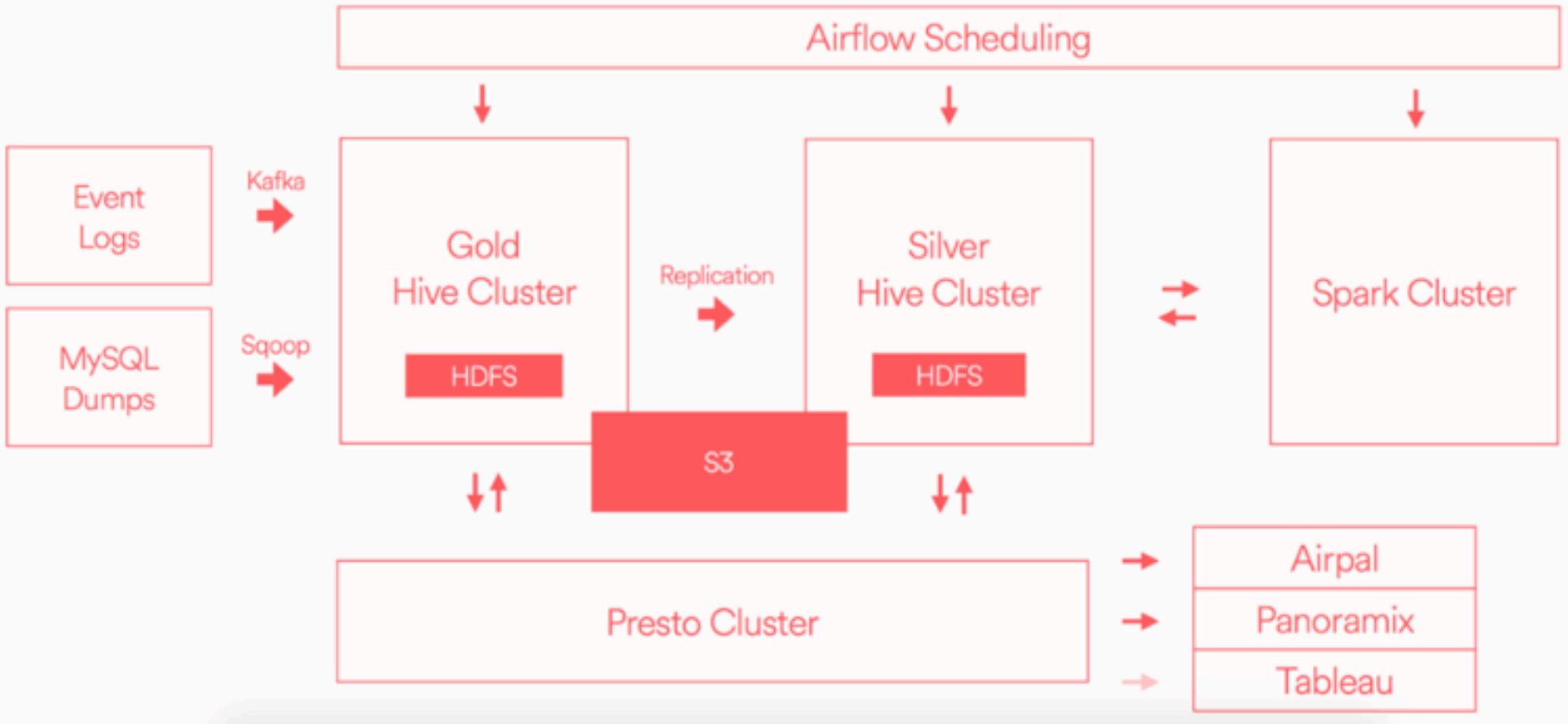


http://techblog.netflix.com/2016/02/evolution-of-netflix-data-pipeline.html

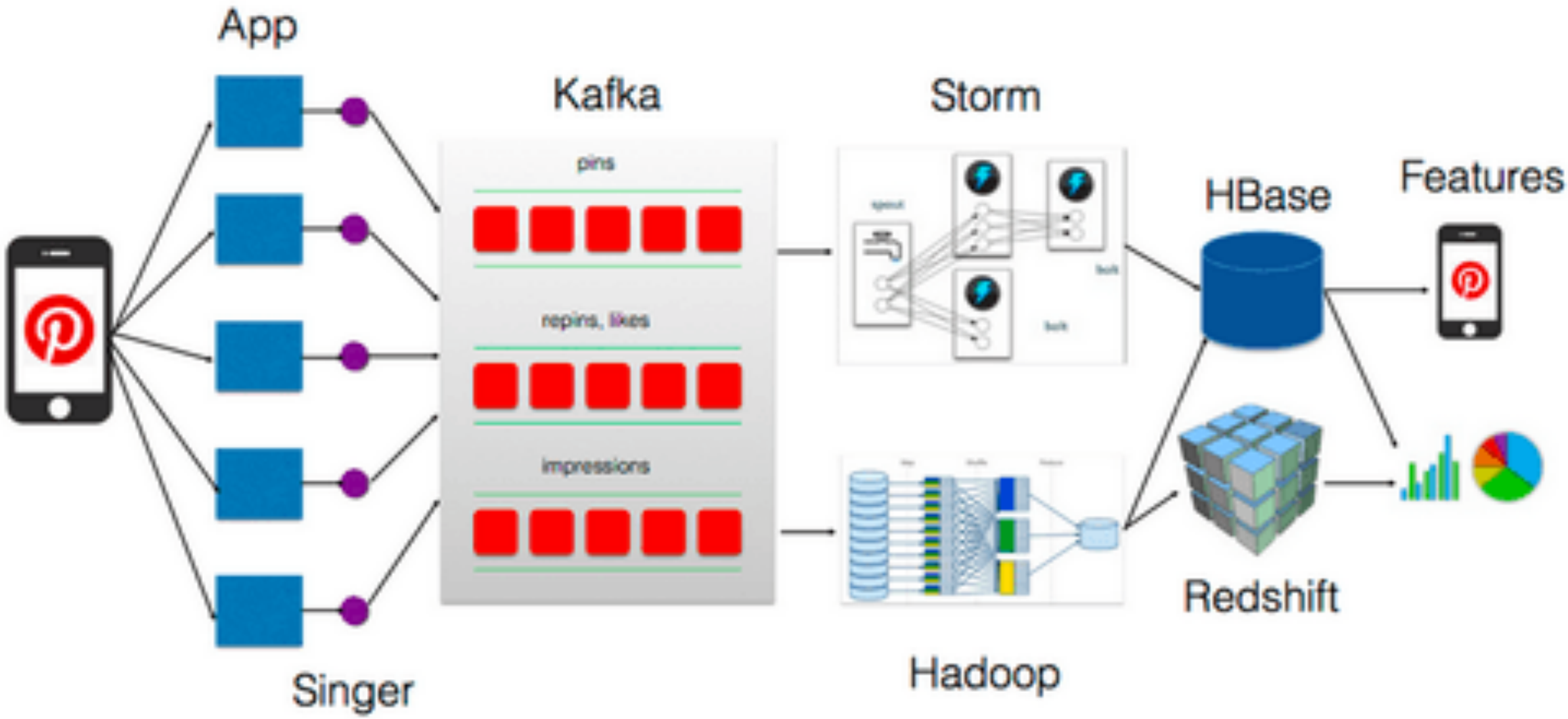
Data analytics Architecture adopted by LinkedIn



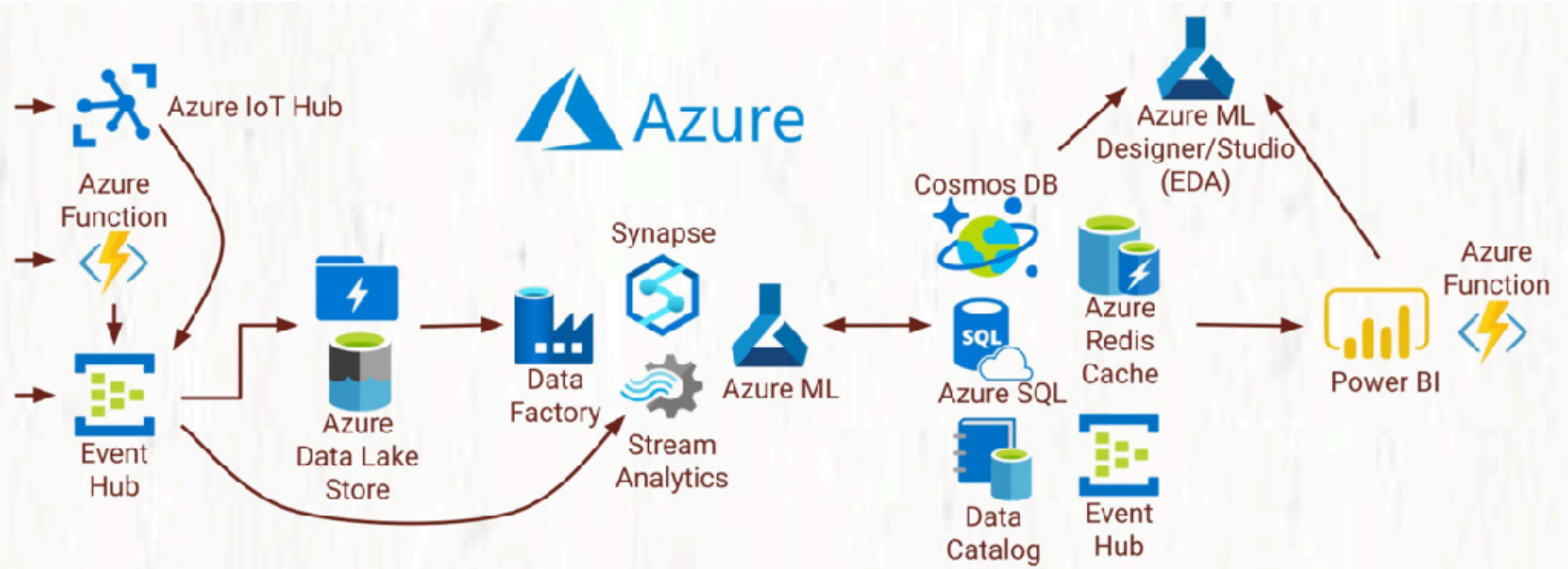
AIRBNB DATA INFRA

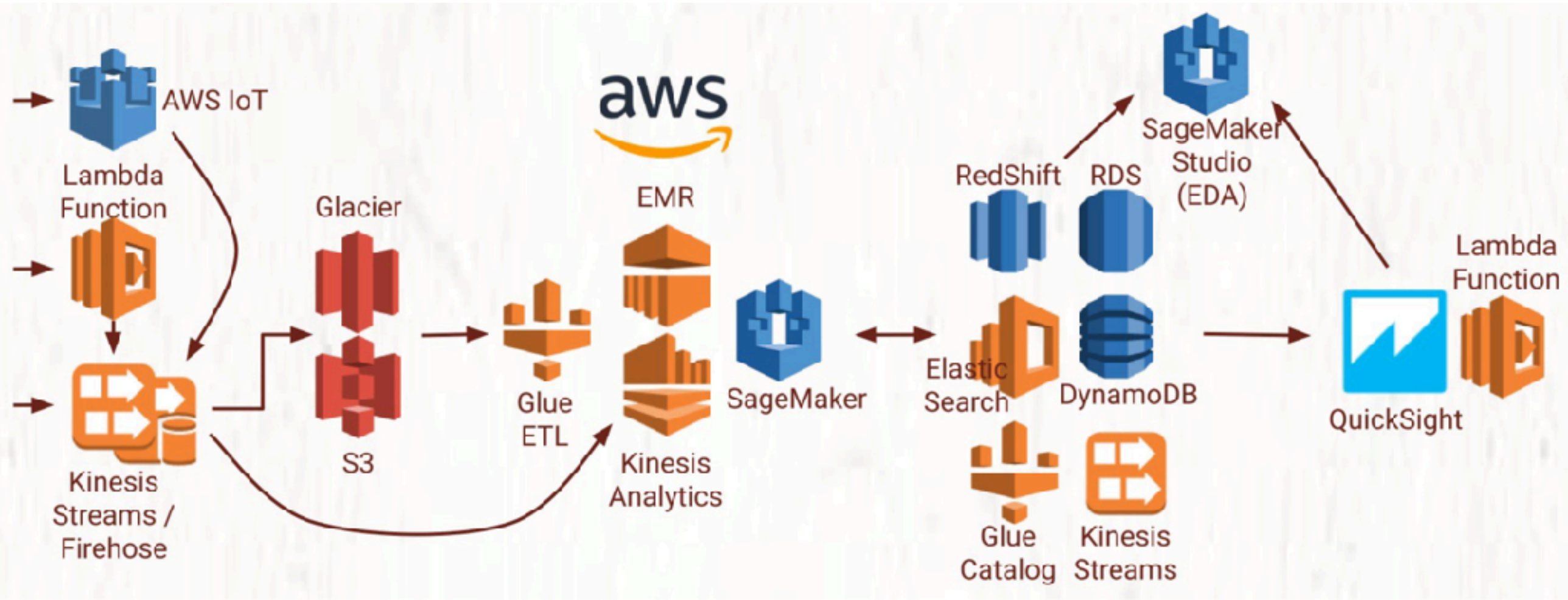


Data analytics Architecture adopted by Pinterest

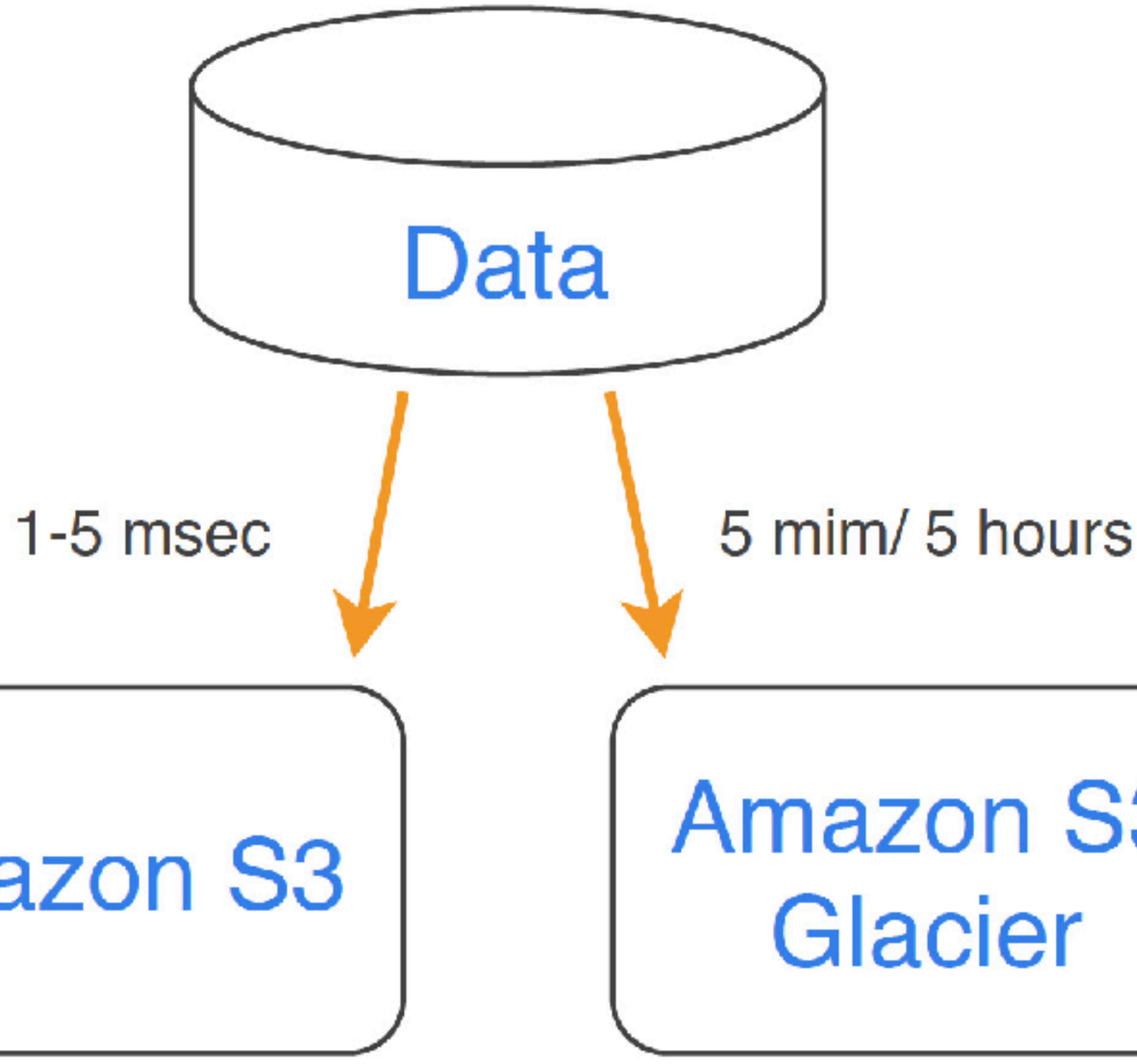


Data Architecture overview


Azure IoT Hub
Azure
Function
Event
Hub
Azure
Data Lake
Store
Synapse
Data
Factory
Stream
Analytics
Azure ML
Azure ML
Designer/Studio
(EDA)
Cosmos DB
SQL
Azure SQL
Data
Catalog
Azure
Redis
Cache
Event
Hub
Power BI
Azure
Function


AWS IoT
Lambda Function
Glacier
EMR
aws
RedShift
RDS
SageMaker Studio (EDA)
Lambda Function
Glue ETL
S3
SageMaker
Elastic Search
DynamoDB
QuickSight
Kinesis Analytics
Kinesis Streams / Firehose
Glue Catalog
Kinesis Streams

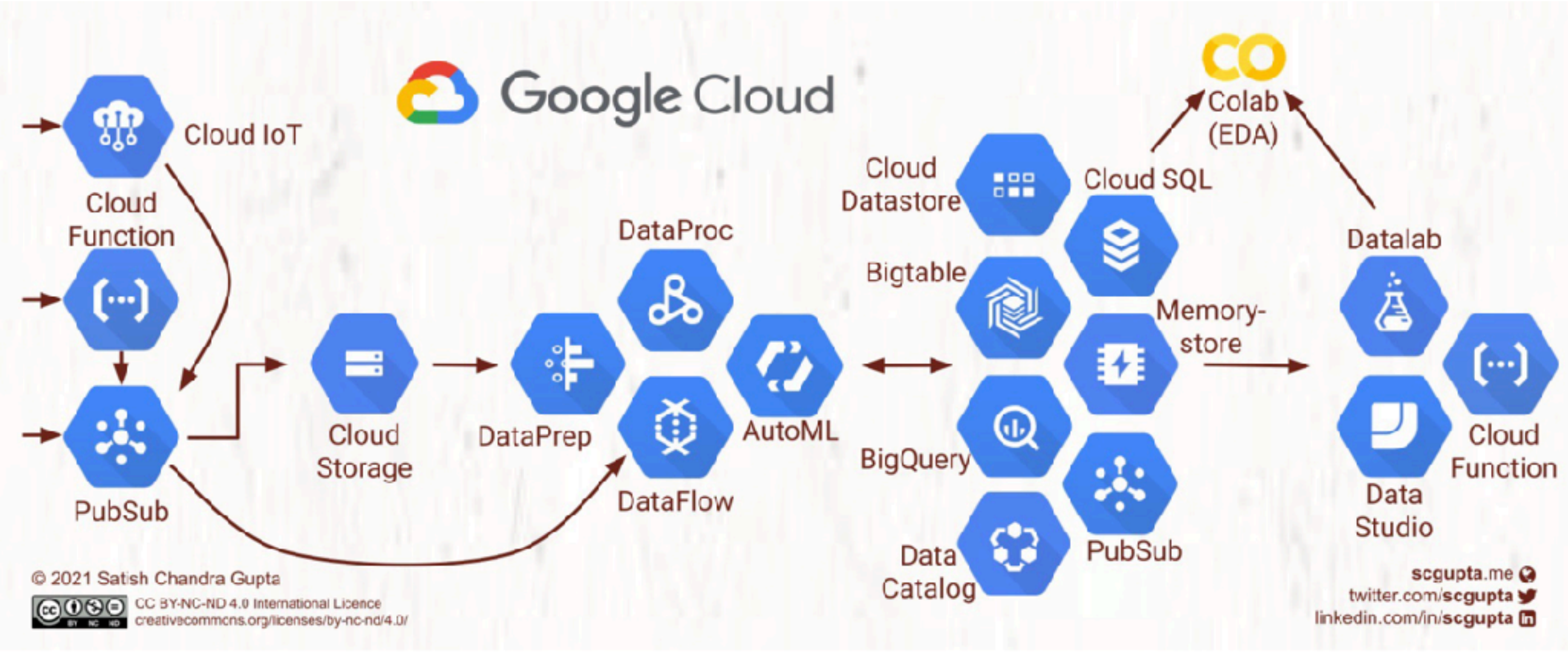
	Kinesis Data Streams	Kinesis Firehose
Purpose	Low latency streaming service for ingest at scale	Data transfer service to load streaming data into Amazon S3, Redshift, Elasticsearch & Splunk
Provisioning	Managed service but needs configuration for shards.	Fully managed service, no administration.
Processing	Real Time (~200ms latency for classic, ~70ms for enhanced fan out)	Near real time (depends on buffer size OR buffer time min. 60 secs)
Scaling	Must manage scaling (configure shards)	Automated Scaling – as per the demand
Data Storage	Configurable from 1 to 7 days	Does not provide data storage
Replay capability	Supports replay capability	Does not support replay capability
Producers	Need to write code for producer. Supports SDK, Kinesis Agent, KPL, CloudWatch, IoT	Need to write code for producer. Supports KPL, Kinesis Agent, Data Streams, CloudWatch, IoT
Consumers	Open ended. Supports multiple consumers and destinations. Supports KCL and Spark.	Closed ended. Handled by Firehose. Does not support KCL or Spark.


Data
1-5 msec
5 mim/ 5 hours
Amazon S3
Amazon S3 Glacier
$0.025 per GB per month
$0.004 per GB per month + extraction cost

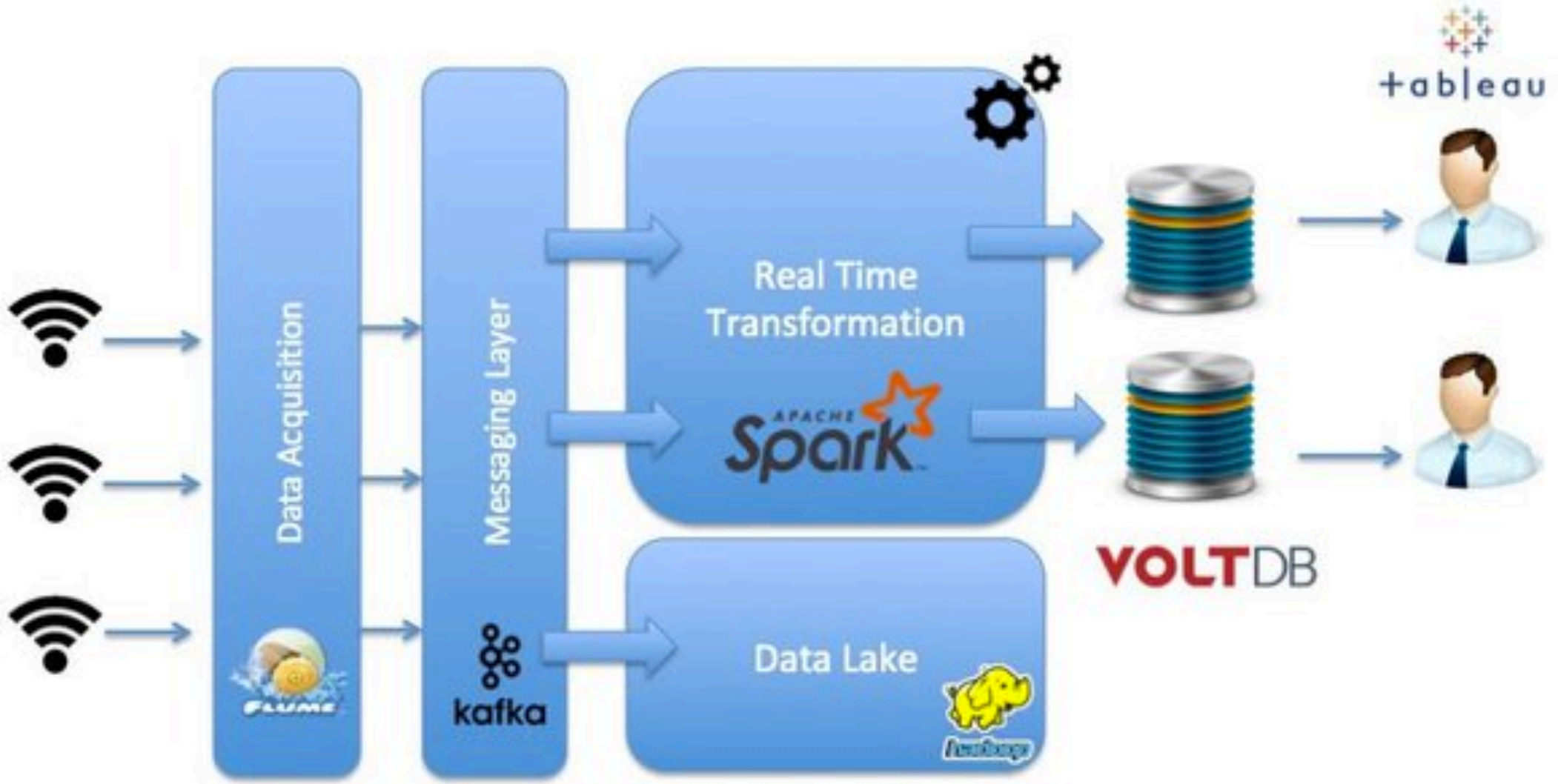
	S3	S3 Glacier
Access speed	Hi	Low
API availability	Yes	Yes
Data storage cost	Relatively high	Very low
Object size	Up to 5 Tb	40 Tb
Accommodation in regions	Many regions	Average
Static Web Content	Yes	No
Supporting Versioning	Yes	No

	Data Streams	Data Firehose	Data Analytics	Video Streams
Short definition	Scalable and durable real-time data streaming service.	Capture, transform, and deliver streaming data into data lakes, data stores, and analytics services.	Transform and analyze streaming data in real time with Apache Flink.	Stream video from connected devices to AWS for analytics, machine learning, playback, and other processing.
Data sources	Any data source (servers, mobile devices, IoT devices, etc) that can call the Kinesis API to send data.	Any data source (servers, mobile devices, IoT devices, etc) that can call the Kinesis API to send data.	Amazon MSK, Amazon Kinesis Data Streams, servers, mobile devices, IoT devices, etc.	Any streaming device that supports Kinesis Video Streams SDK.
Data consumers	Kinesis Data Analytics, Amazon EMR, Amazon EC2, AWS Lambda	Amazon S3, Amazon Redshift, Amazon Elasticsearch Service, generic HTTP endpoints, Datadog, New Relic, MongoDB, and Splunk	Analysis results can be sent to another Kinesis stream, a Kinesis Data Firehose delivery stream, or a Lambda function	Amazon Rekognition, Amazon SageMaker, MxNet, TensorFlow, HLS-based media playback, custom media processing application
Use cases	– Log and event data collection – Real-time analytics – Mobile data capture – Gaming data feed	– IoT Analytics – Clickstream Analytics – Log Analytics – Security monitoring	– Streaming ETL – Real-time analytics – Stateful event processing	– Smart technologies – Video-related AI/ML – Video processing

S. No.	Amazon DynamoDB	Amazon Redshift
1	It was developed by Amazon in 2012.	It was developed by Amazon in 2012.
2	It is hosted, scalable database service by Amazon with data stored in Amazon cloud.	It is large scale data warehouse service for use with business intelligence tools.
3	It does not support SQL query language.	It supports SQL query language. But it does not fully support an SQL-standard.
4	It does not provide concept of Referential Integrity. Hence, no Foreign Keys.	It provides concept of Referential Integrity. Hence, there are Foreign Keys.
5	Its Primary database models are Document store and Key-value store.	Its primary database model is Relational DBMS.
6	It does not support Server-side scripting.	It supports user-defined functions for Server-side scripting in python.
7	Eventual Consistency and Immediate Consistency are used to ensure consistency in distributed system.	Immediate Consistency is used to ensure consistency in distributed system.
8	It does not offers API for user-defined Map/Reduce methods. But maybe implemented via Amazon Elastic MapReduce.	It does not offer API for user-defined Map/Reduce methods.
9	It supports secondary indexes.	It supports restricted secondary indexes.


Google Cloud
Cloud IoT
Cloud Function
PubSub
Cloud Storage
DataProc
DataPrep
DataFlow
AutoML
BigQuery
Cloud Datastore
Bigtable
Data Catalog
Cloud SQL
Memory-store
PubSub
Colab (EDA)
Datalab
Data Studio
Cloud Function
© 2021 Satish Chandra Gupta
CC BY-NC-ND 4.0 International Licence
creativecommons.org/licenses/by-nc-nd/4.0/
scgupta.me
twitter.com/scgupta
linkedin.com/in/scgupta

Real Time Analytics with NewSQL



Data Sources | Ingestion | Transformation & Data Lake | Storage & Analytics | Presentation

VoltDB, NuoDB or MemSQL

acid
SQL query
- Schema
- Scale

Schema
Domain Query
Scale

acid
SQL query
Scale
- Schema

SQL

No Sql

Streaming

New Sql

- Key Value
- Document
- Graph
- Column
- Text
- Time
- Geo
- Logger

Storage ?

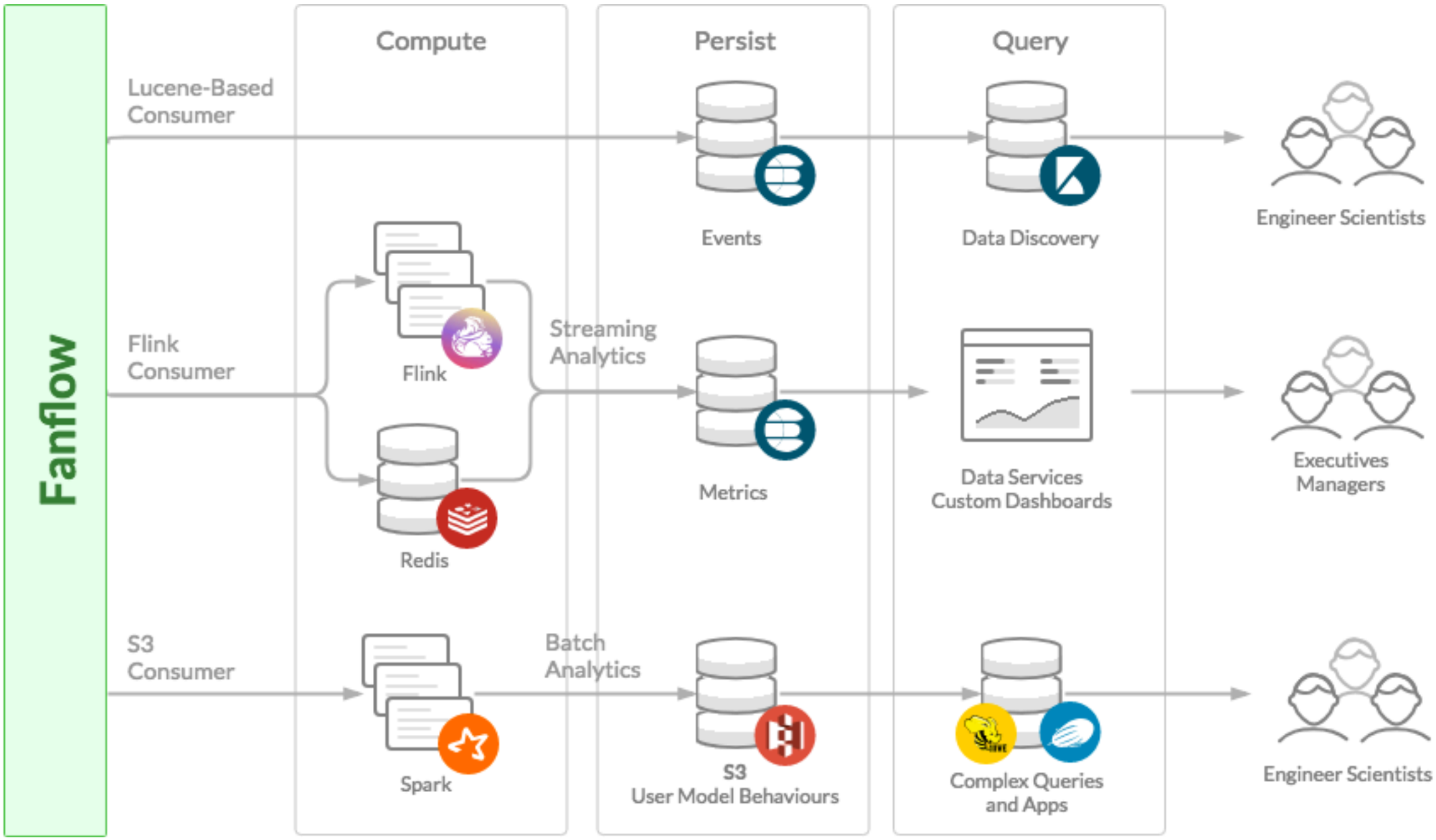
* hw

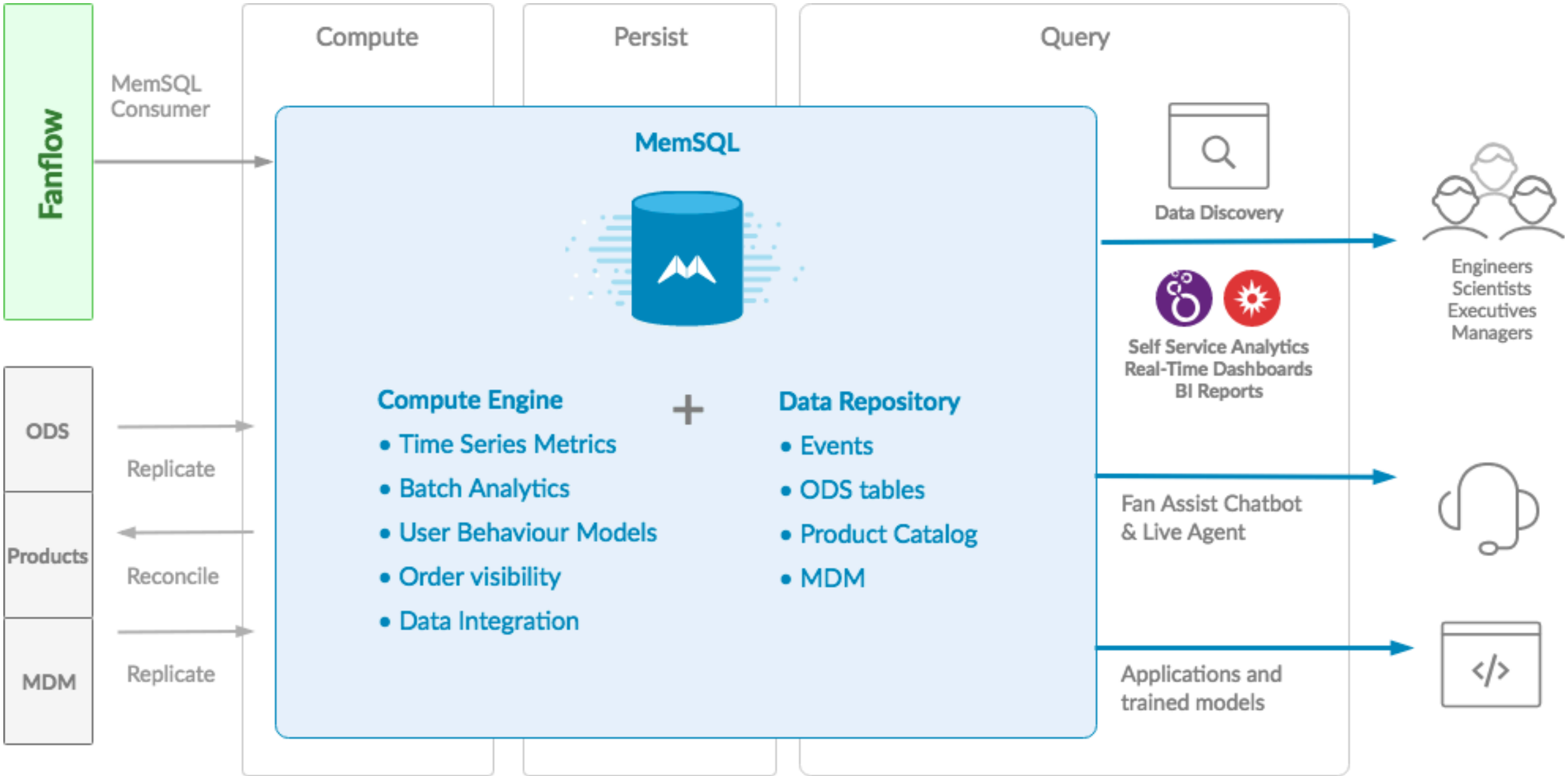
** hw

** hw

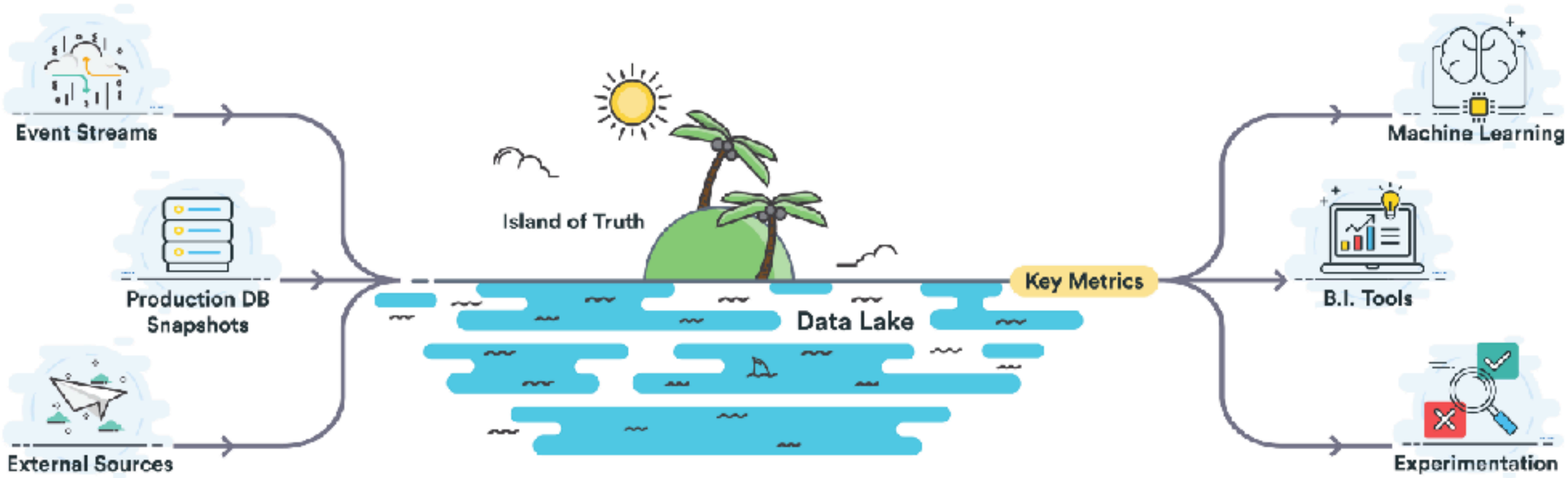
**** hw

Previous Fanflow analytics architecture



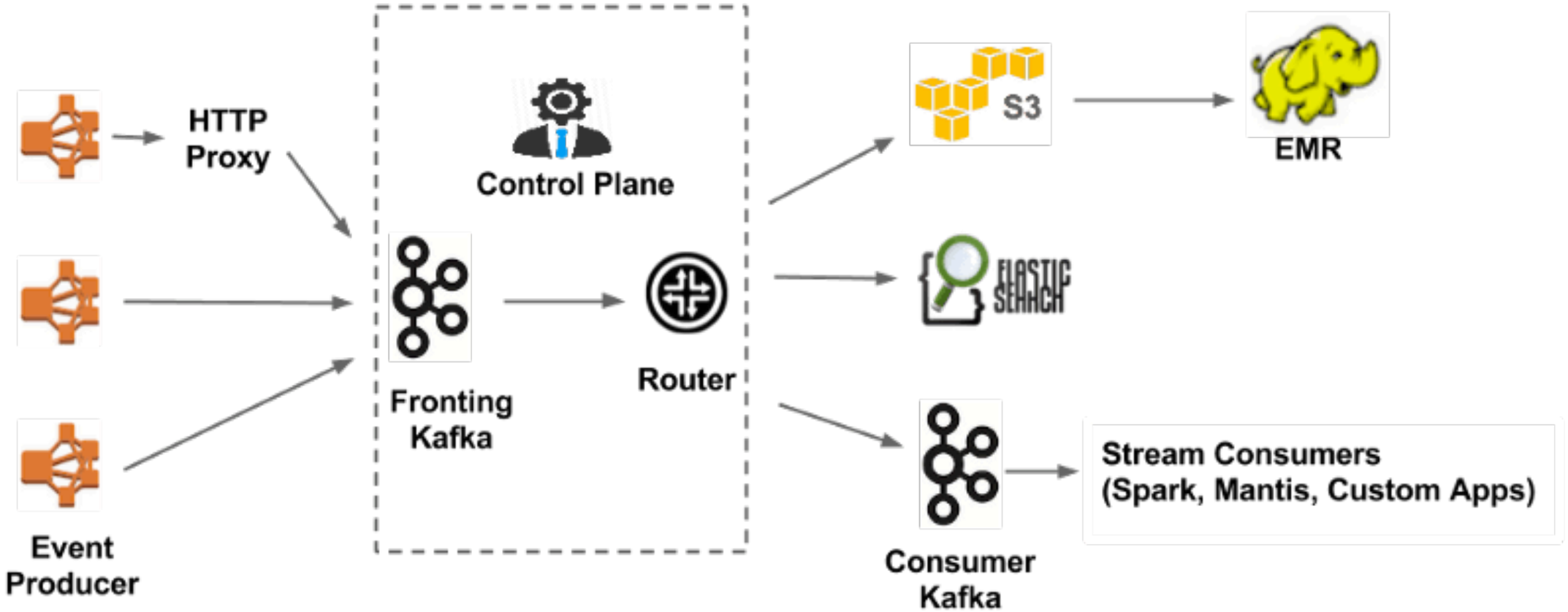


An operational database replaced the Lucene-based indexers, and Spark and Flink jobs were converted to SQL-based processing jobs

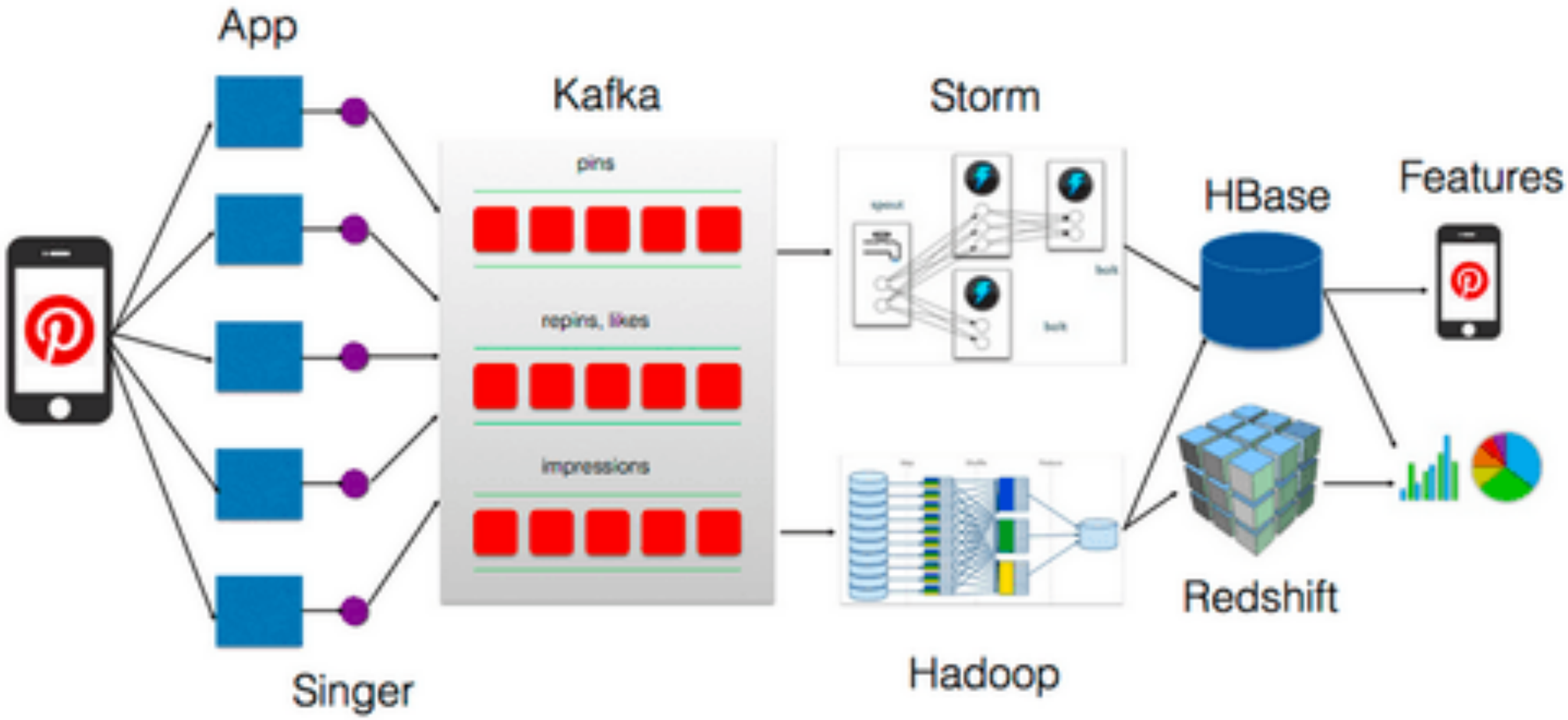

Event Streams
Production DB
Snapshots
External Sources
Island of Truth
Data Lake
Key Metrics
Machine Learning
B.I. Tools
Experimentation

NetFlix

500 billion events per day/ 1.3 PB per day
At peak hours, they'll record **8 million events per second./ 24 GB**
They employ **over 100 people** as data engineers or analysts.

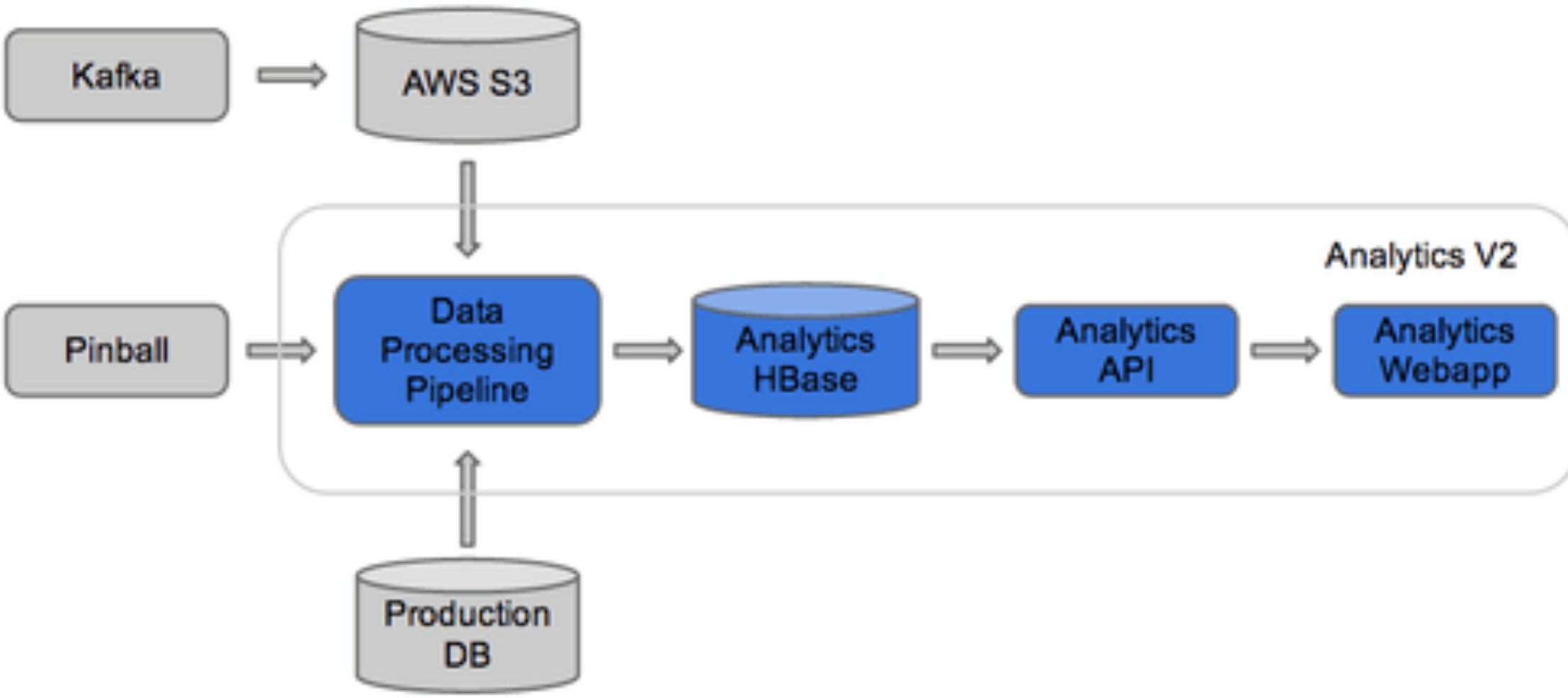


Pinterest



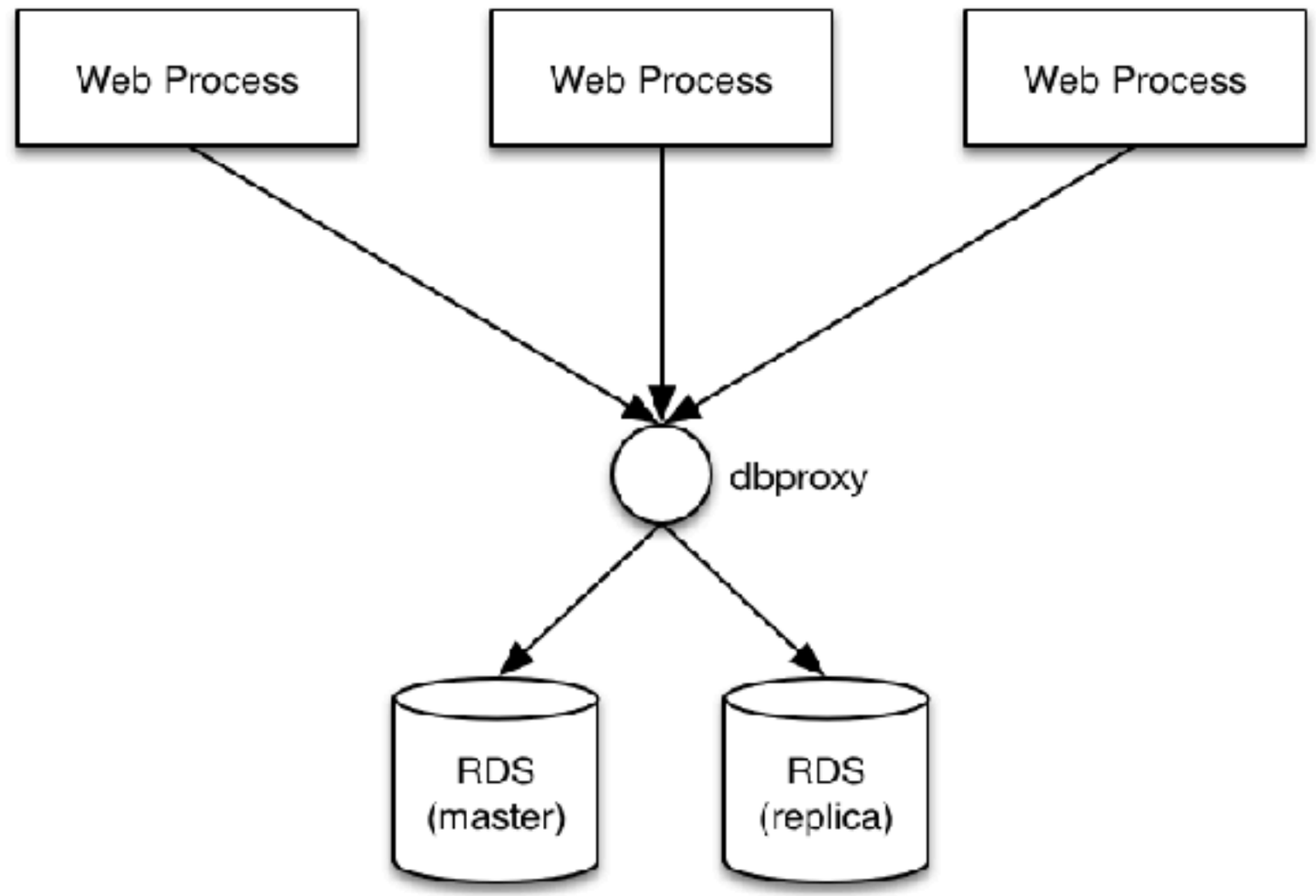
Data Architecture overview

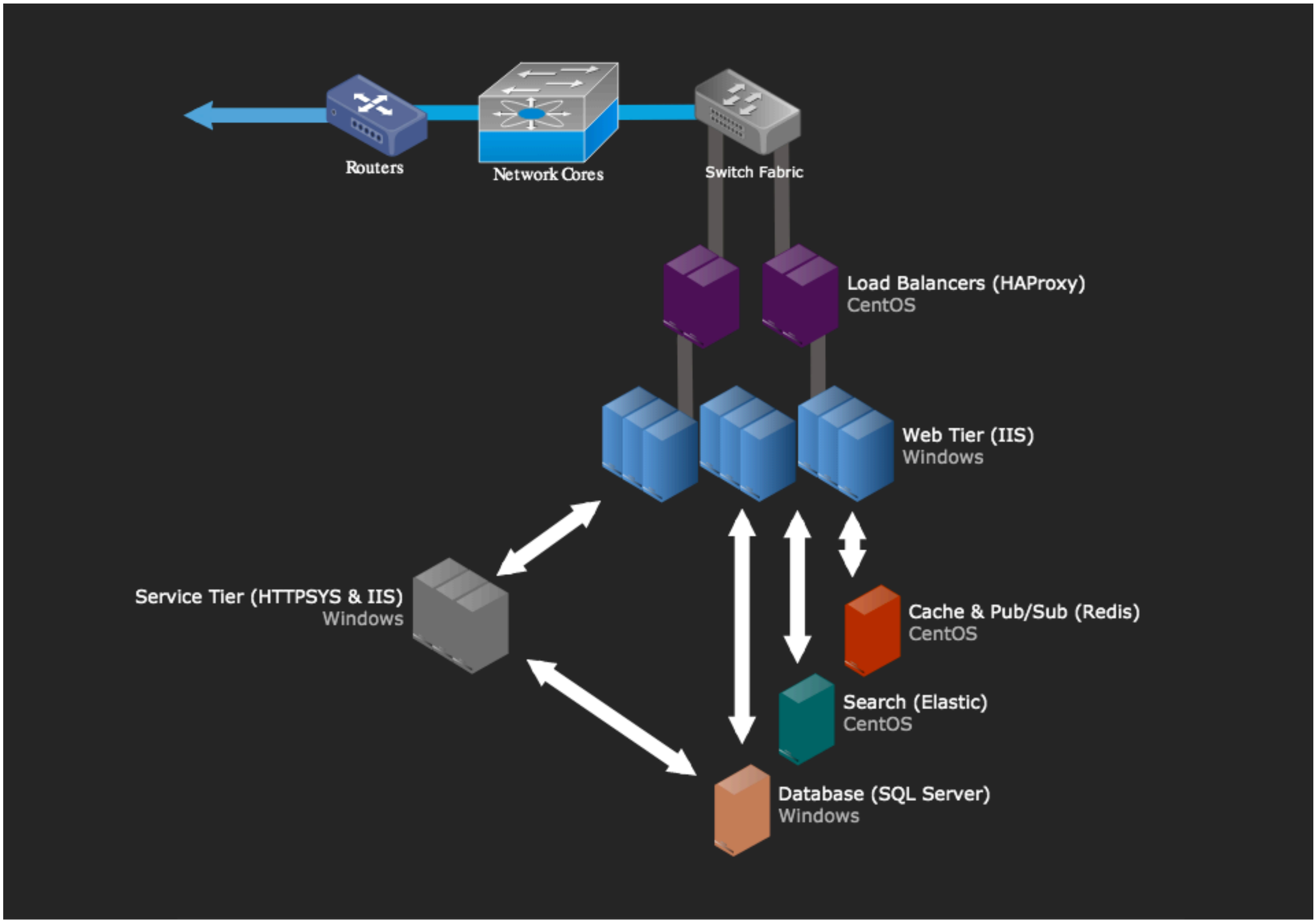
they also need to provide detailed analytics to their ad buyers.



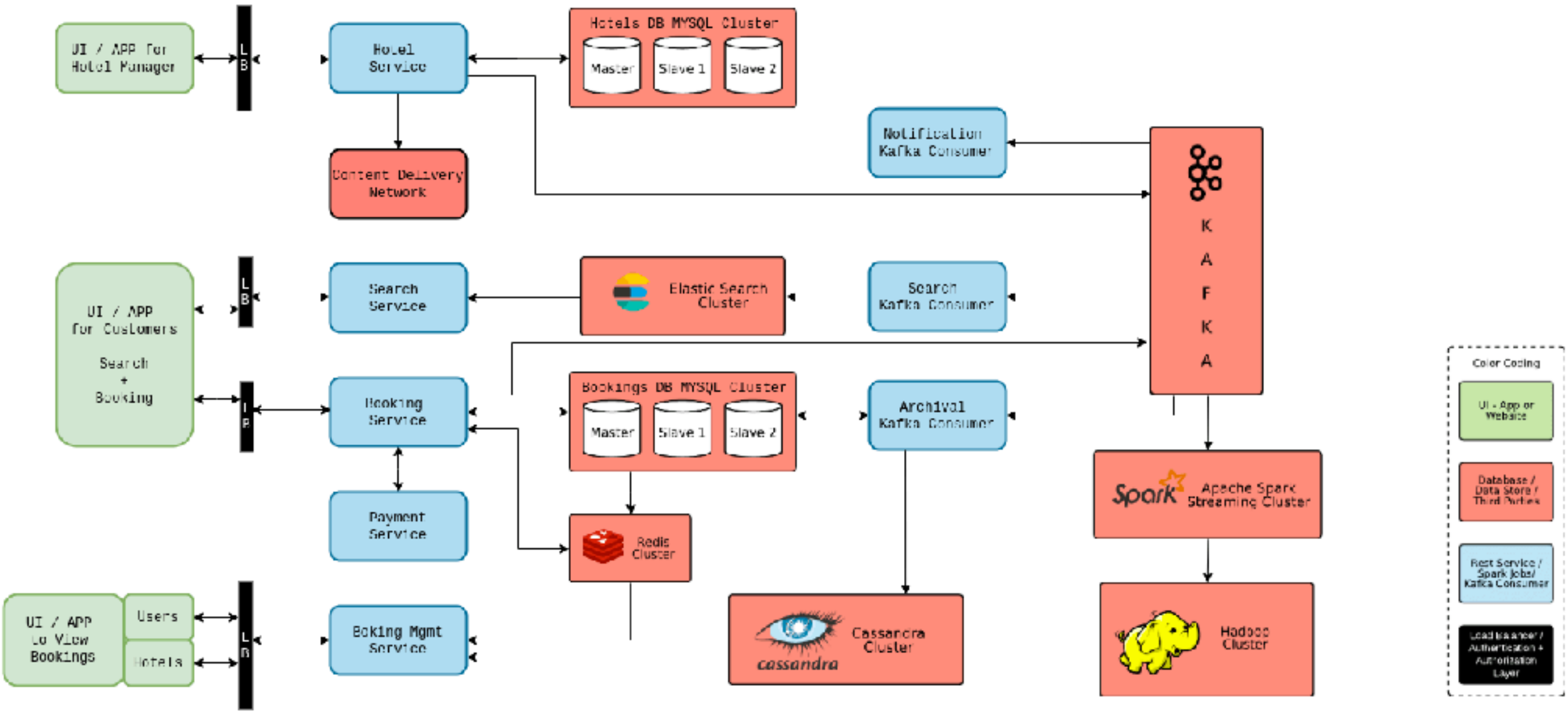
Data Tier

Airbnb uses Amazon RDS as main MySQL database. The databases are deployed in multi-AZ (availability zone). Below 3-tier architecture reflects the basic pattern.

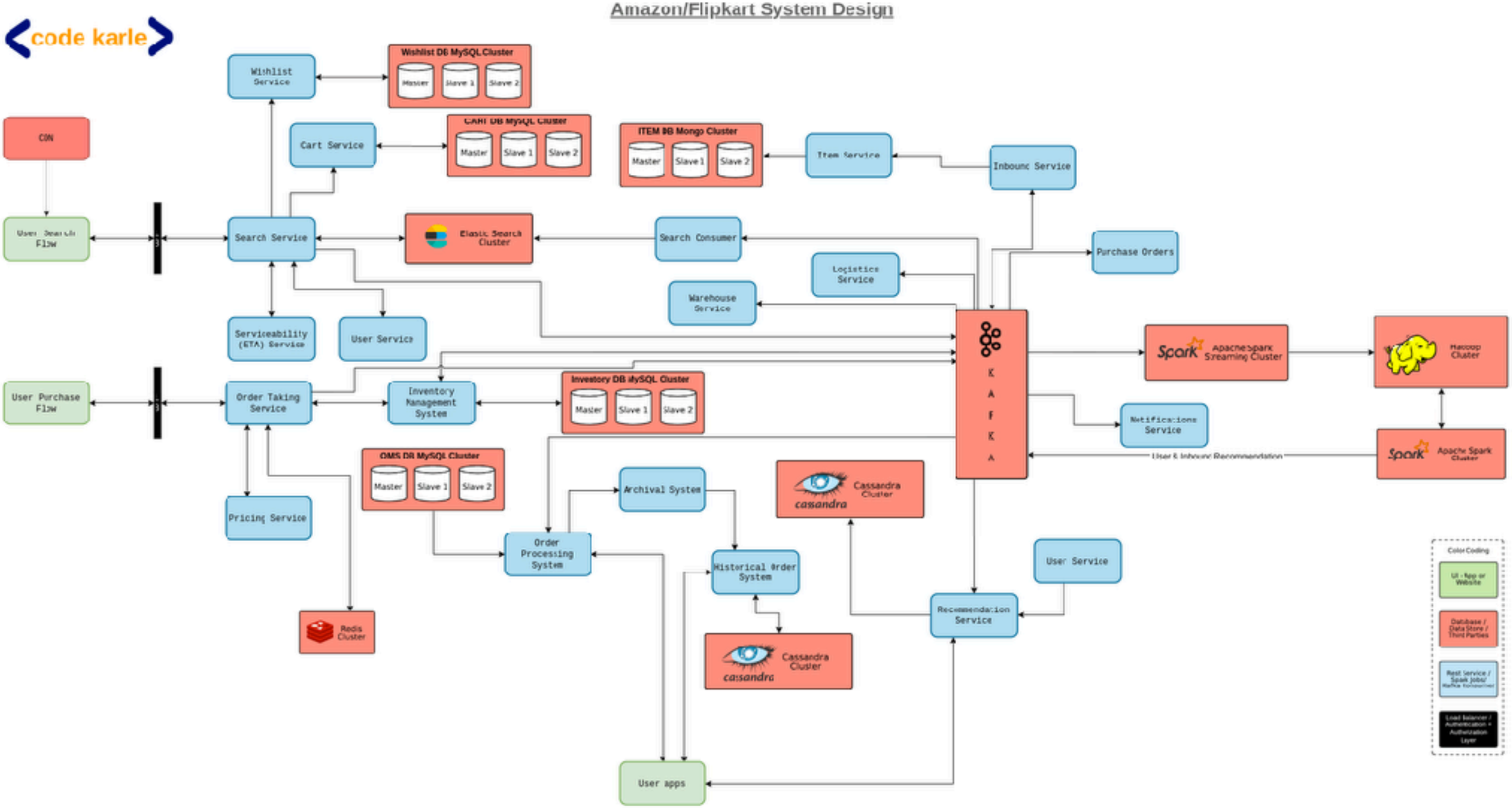



Routers
Network Cores
Switch Fabric
Load Balancers (HAProxy)
CentOS
Web Tier (IIS)
Windows
Service Tier (HTTPSYS & IIS)
Windows
Cache & Pub/Sub (Redis)
CentOS
Search (Elastic)
CentOS
Database (SQL Server)
Windows

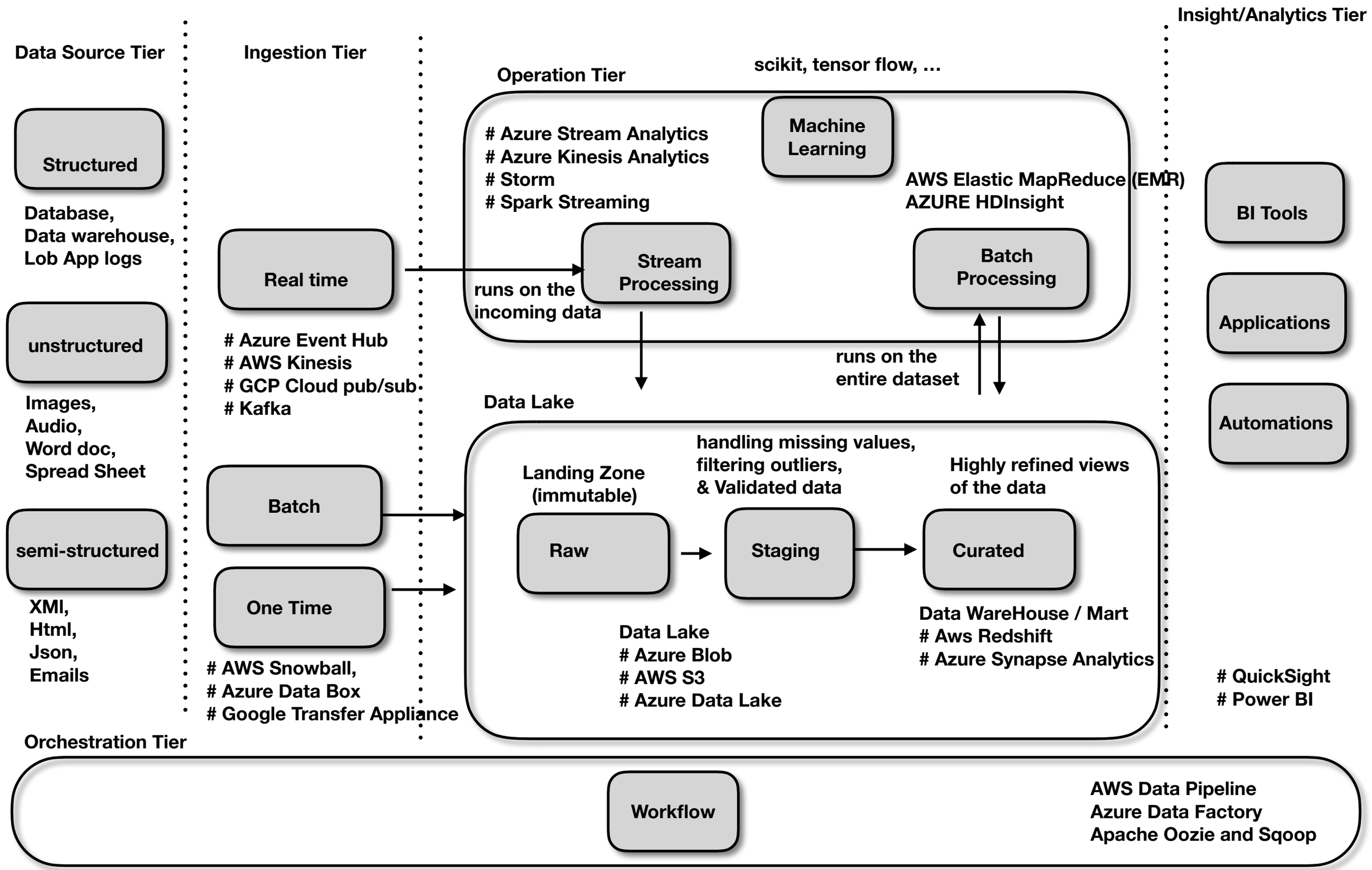
Airbnb/Booking.com System Design



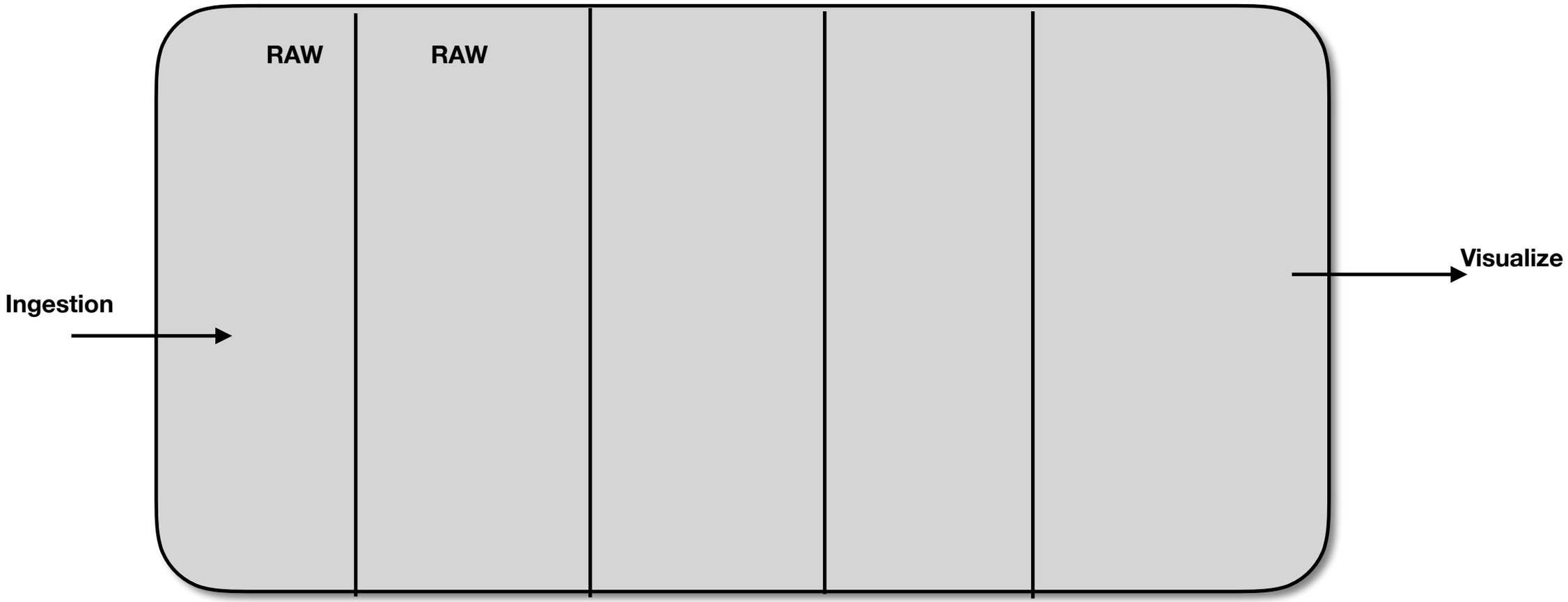
Amazon/Flipkart System Design



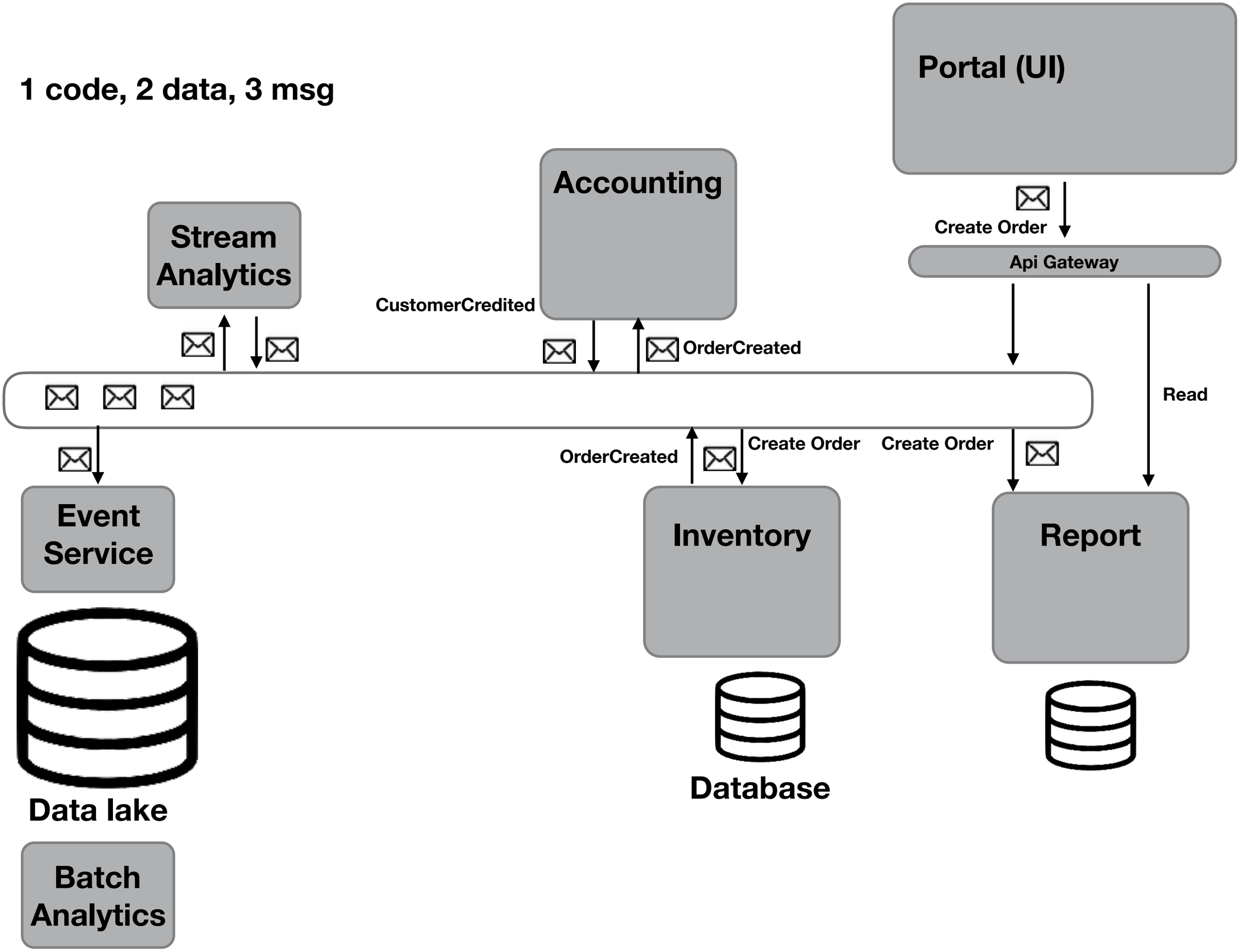
Reference Architecture
Analytical Application



Streaming
Sqoop
Data logs
Data Bricks



1 code, 2 data, 3 msg



	Kafka	Pulsar	RabbitMQ (Mirrored)
Peak Throughput (MB/s)	605 MB/s	305 MB/s	38 MB/s
p99 Latency (ms)	5 ms (200 MB/s load)	25 ms (200 MB/s load)	1 ms* (reduced 30 MB/s load)
Number of Programming Languages Supported	17	6	22
Stack Overflow Questions	21,233	134	11,430
Pub/sub	Yes	Yes	Yes
Message routing	Medium	Medium	High
Queueing	Low	Medium	High
Event Streaming	High	Medium	No
Mission-critical	High	Low	High
Slack Community Size	23,057	2,332	7,492
Meetups	486	1	1
Message replay, time travel	+++	+++	-
Exactly-once processing	+++	+	-
consumption	Pull	Push	Push
Permanent storage	Yes	Partial	No

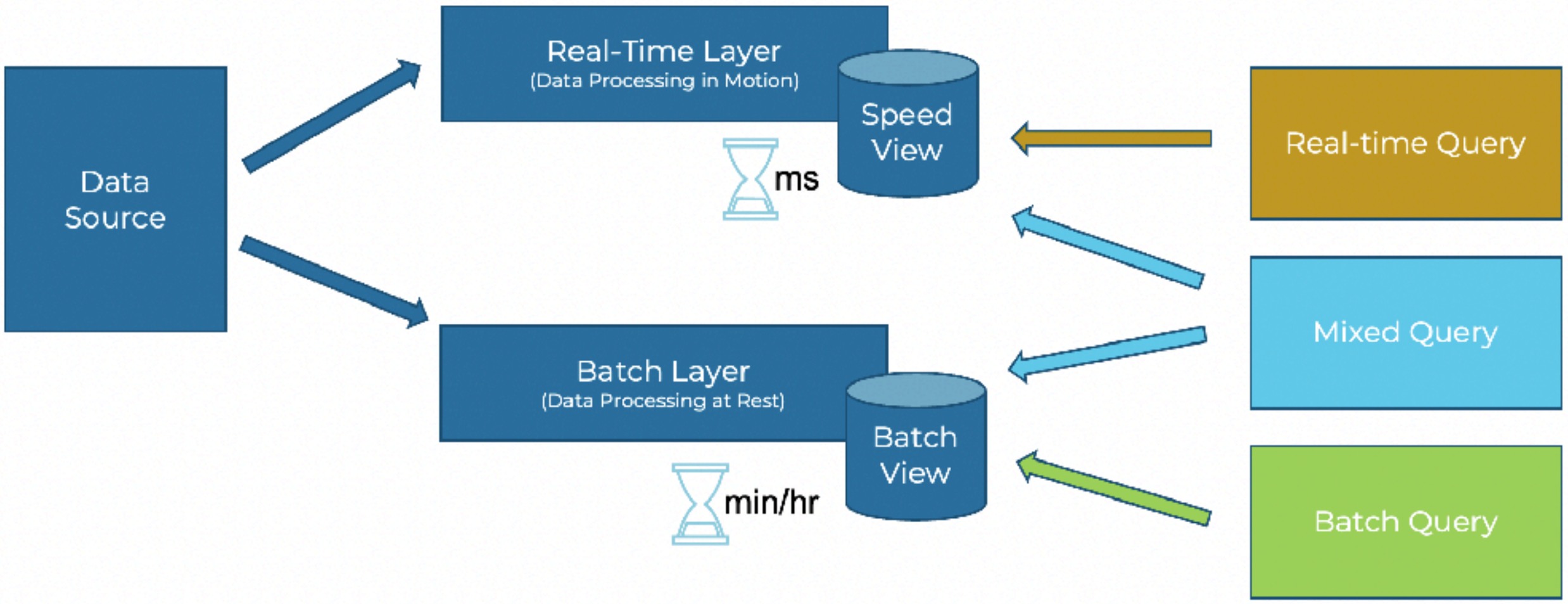
	Rabbitmq	Kafka	Redis
Scale	50K msg per second	1 million/ sec	1 million/ sec
Transient messages	Yes	No	Y
Persistent yes	Yes	Yes	Y
Protocol	AMQP		
One to one	Y	N	Y
One to many	Y	Y	Y
Advanced Message Queueing Protocol-based routing	Y	N	N
consumption pattern	Pull+Push	Pull	
Client Libraries	Py, java, php, .net, js,…	Py, java, php, .net, js,…	
Managed Service	yes, but not native in aws	Azure, aws, Confluent	Azure, aws
License	MPL	Apache	
Message Priority	Yes	No	
Monitoring	yes	yes	
Authentication	OAuth2, Std Auth	Kerberos, Oauth2, std Auth	
Message delay	microsecond	Millisecond	

Table 1

	Rabbitmq	Kafka	Redis
Scale	50K msg per second	1 million/ sec	1 million/ sec
Transient messages	Yes	No	Y
Persistent yes	Yes	Yes	Y
Protocol	AMQP		
One to one	Y	N	Y
One to many	Y	Y	Y
Advanced Message Queueing Protocol-based routing	Y	N	N
consumption pattern	Pull+Push	Pull	
Client Libraries	Py, java, php, .net, js,…	Py, java, php, .net, js,…	
Managed Service	yes, but not native in aws	Azure, aws, Confluent	Azure, aws
License	MPL	Apache	
Message Priority	Yes	No	
Monitoring	yes	yes	
Authentication	OAuth2, Std Auth	Kerberos, Oauth2, std Auth	
Message delay	microsecond	Millisecond	
Potential data loss	Yes	Yes	
Community and vendor support	Good	Good	Good
Building an event store system (used as a store)	No	Yes	No
Ordering	not guaranteed	Guaranteed	
Replay events	No	Yes	No
Transactions	No	Yes	No

Lambda Architecture

Option 2: Separate serving layers



RAW DATA

DATA STORAGE

BATCH PROCESSING

PROCESSING DATA

BATCH LAYER

QUERY

DATA

VIEW 1

VIEW 2

VIEW 3

BATCH VIEWS

MERGED VIEW

PARTIAL AGGREGATIONS

REAL-TIME VIEWS

VIEW 1

VIEW 2

VIEW 3

SERVING LAYER

STREAM

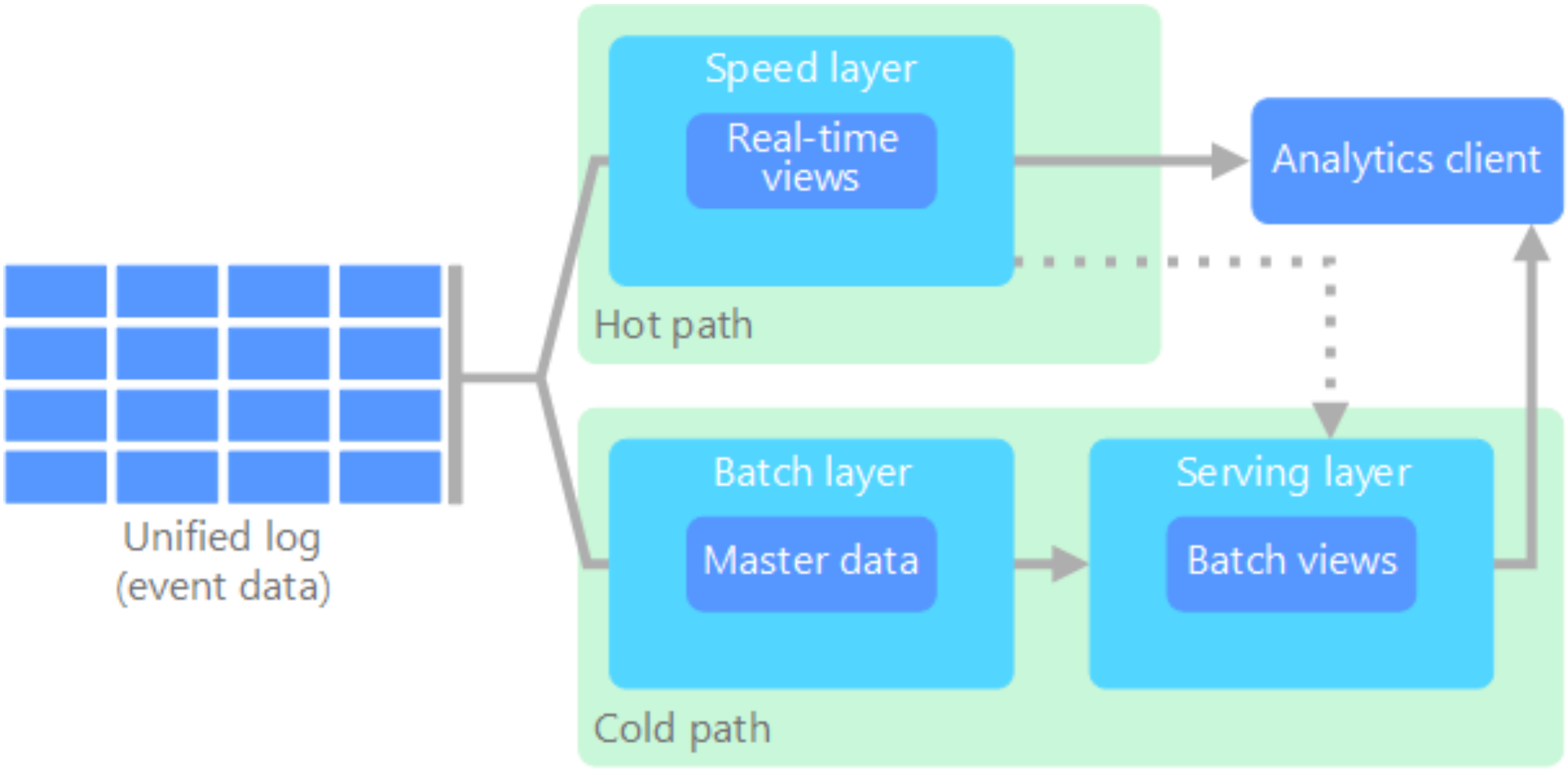
DATA INCREMENT

REAL-TIME PROCESSING

PROCESSED INCREMENT

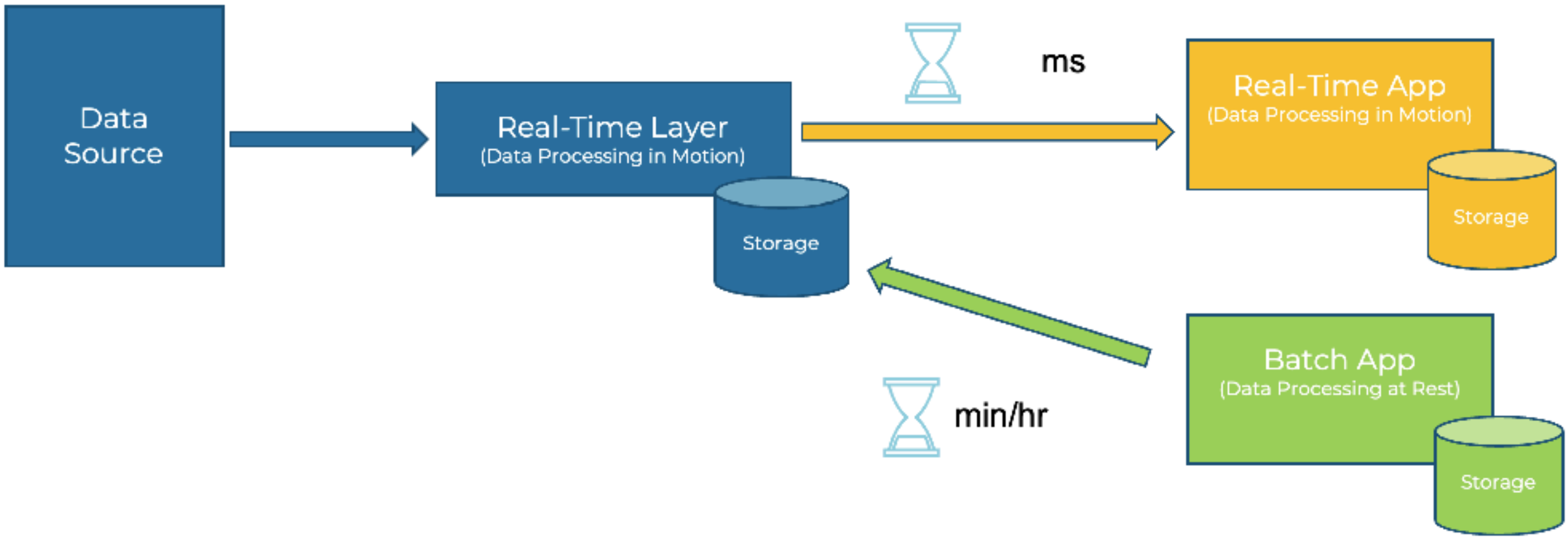
SPEED LAYER

Lambda Architecture

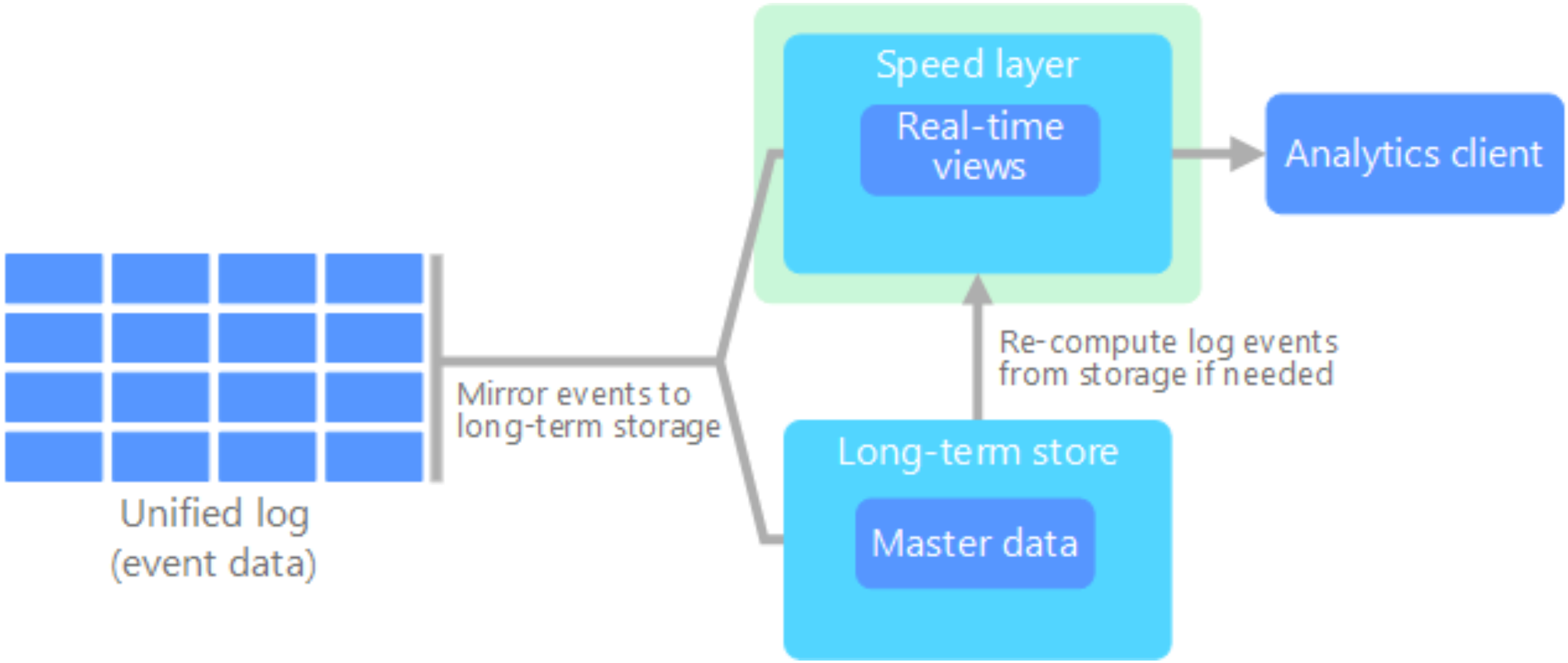


Kappa Architecture

One pipeline for real-time and batch consumers



Kappa Architecture



SERVING LAYER

PARTIAL AGGREGATES

QUERY

REAL-TIME VIEWS

VIEW 1

VIEW 2

VIEW 3

DATA

STREAM

RAW DATA

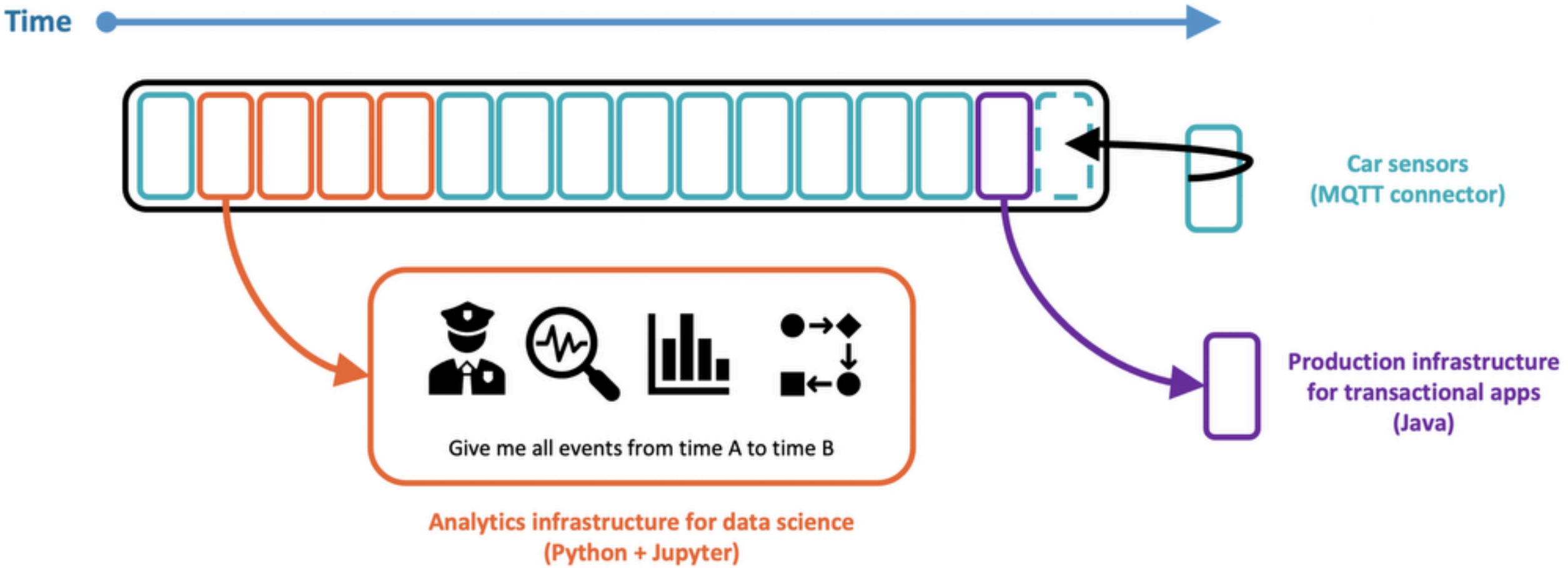
IMMUTABLE DATA

LOG

REAL-TIME PROCESSING

PROCESSED DATA

SPEED LAYER



Kappa architectures enable transactional workloads in addition to analytical workloads too.

A single pipeline for everything. No need for a Lambda architecture! Kappa enables transactional and analytical workloads.

What Architecture to Choose?

Kappa
Real-Time
Data in Motion

VS

Lambda
Batch
Data at Rest

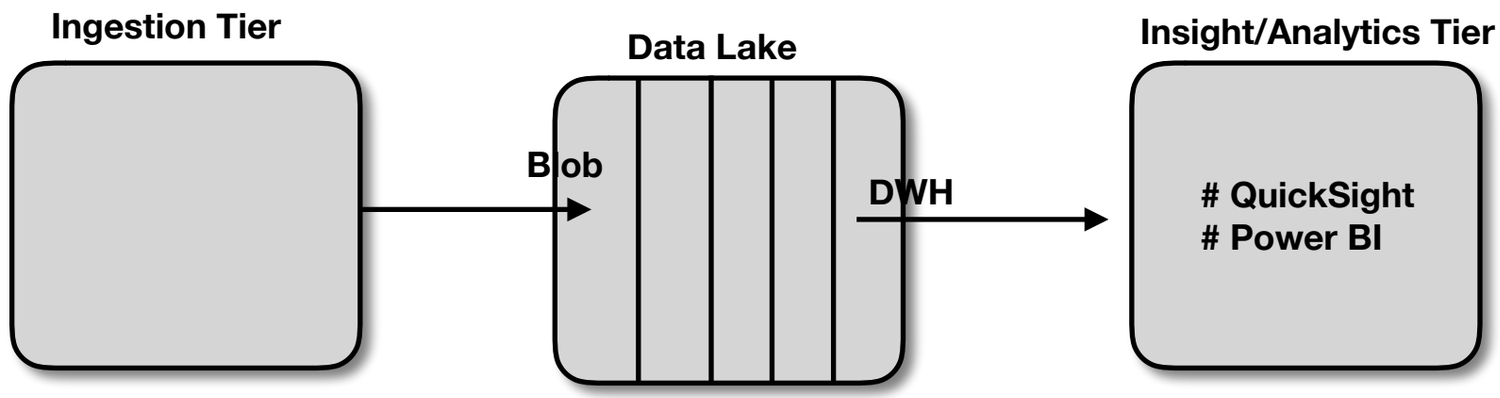
KAI WAEHNER

DataBricks

Source	Ingestion	Storage	Compute	Curated	Visualize
			u1e1,u1e2,u2e3,…..		
Click Stream On desktop	EventHub	DataLake	SparkStreaming	Synapse	PowerPI
			AzureML		

e1,e2,e3,…..
a1,a2,a3,…..
c1,c2,c3,c4….
p1,p2, …

Order,
Session,
Size of window,
Reprocessing



e1,e2,e3,e4,e,e,e,e,e,,,e,e,e,e,e,e,e,e,e,e,…..

Ae1, Ae2, Ae3,……

CEP **Be1,Be2,…**

Ce1

e1,e2,e3,a1,e4,,e5,a2,e6,a3,e7,b1,….c1,….

BG event, medication, Exercise,….

e1,e2,e3,

Kappa Architecture - Challenges & Limitations

Disney STREAMING

Re-Processing Data

What happens when you need to add a field? Or fix your algorithm?

Out of order data

Event time vs. processing time - what do you do with records that arrive late?

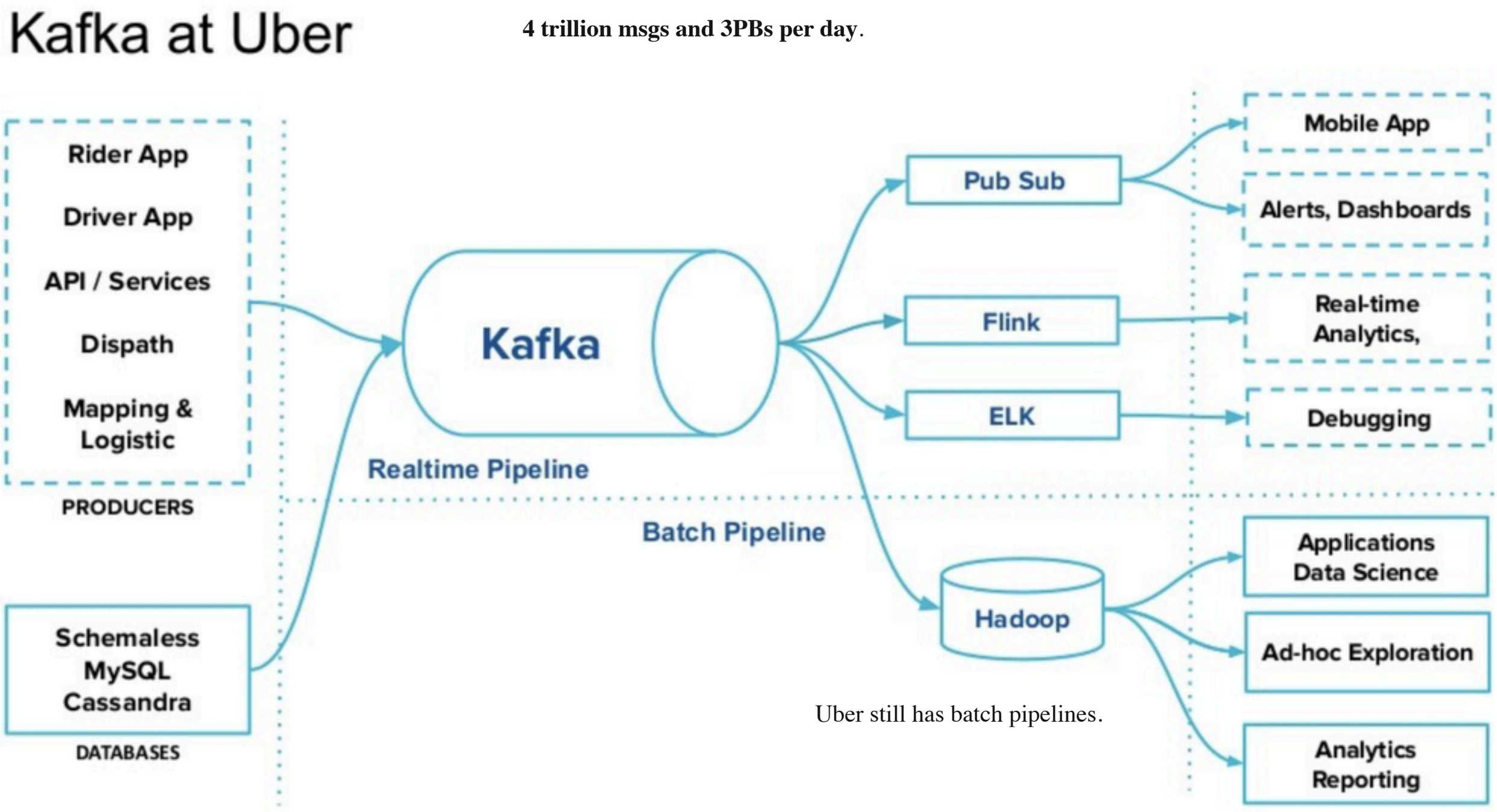
Added Cost?

Paying for compute vs. cold storage. Trade - soft costs (developers) vs. increased hardware costs.

Complex Joins

A few joins are ok - but what happens when you want to join together 25 “tables” from a relational data store?

Kappa at Uber

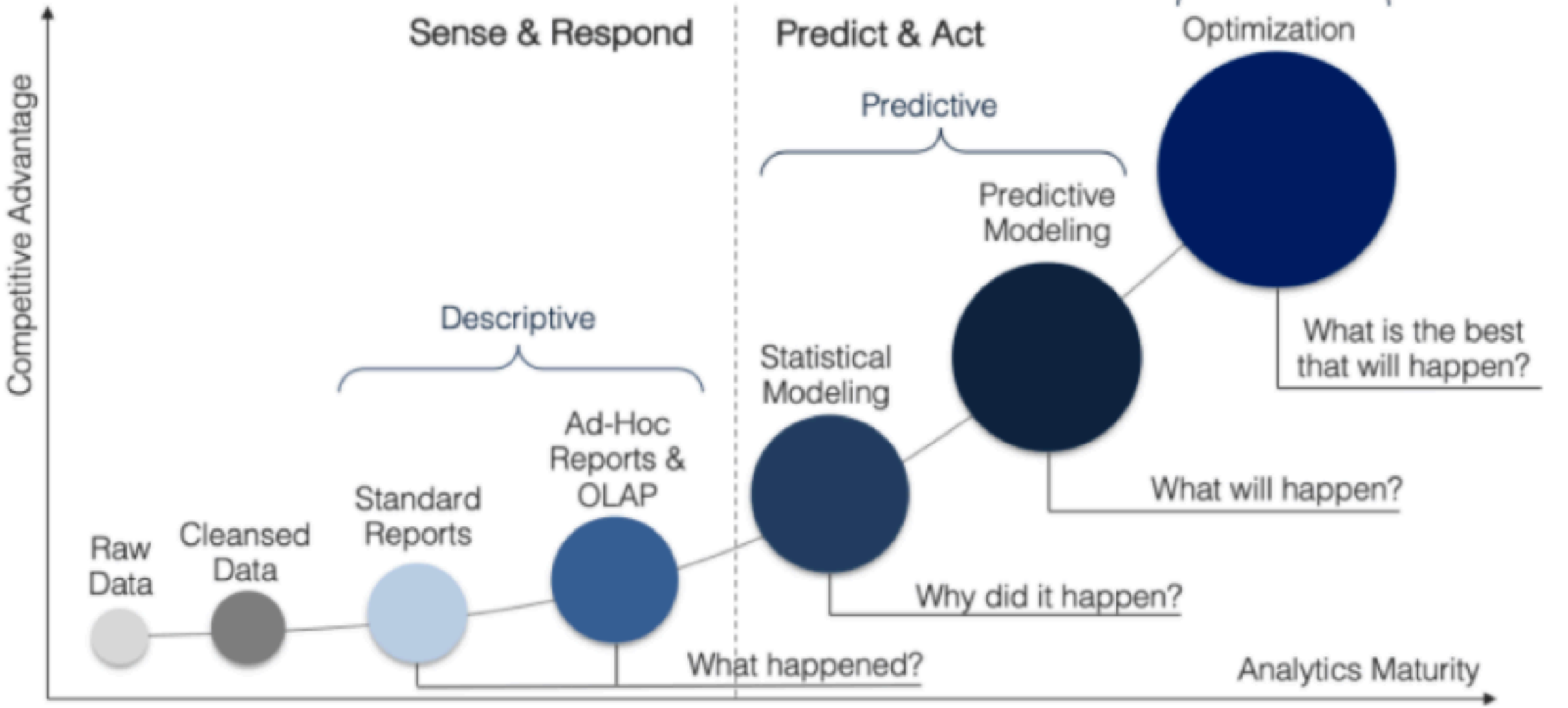


4 trillion msgs and 3PBs per day.

Uber still has batch pipelines.

. **Uber is one of the most significant Kafka users** in the world.

The Analytics Continuum




PROCESS FIRST
PRODUCT FIRST
DATA FIRST
AI FIRST

“Data first” Paradigm

TRADITIONAL BIZ PROCESS PARADIGM

Business Process
Workflow mapping

Codification
Rules based programing

Data Capture

DATA FIRST PARADIGM

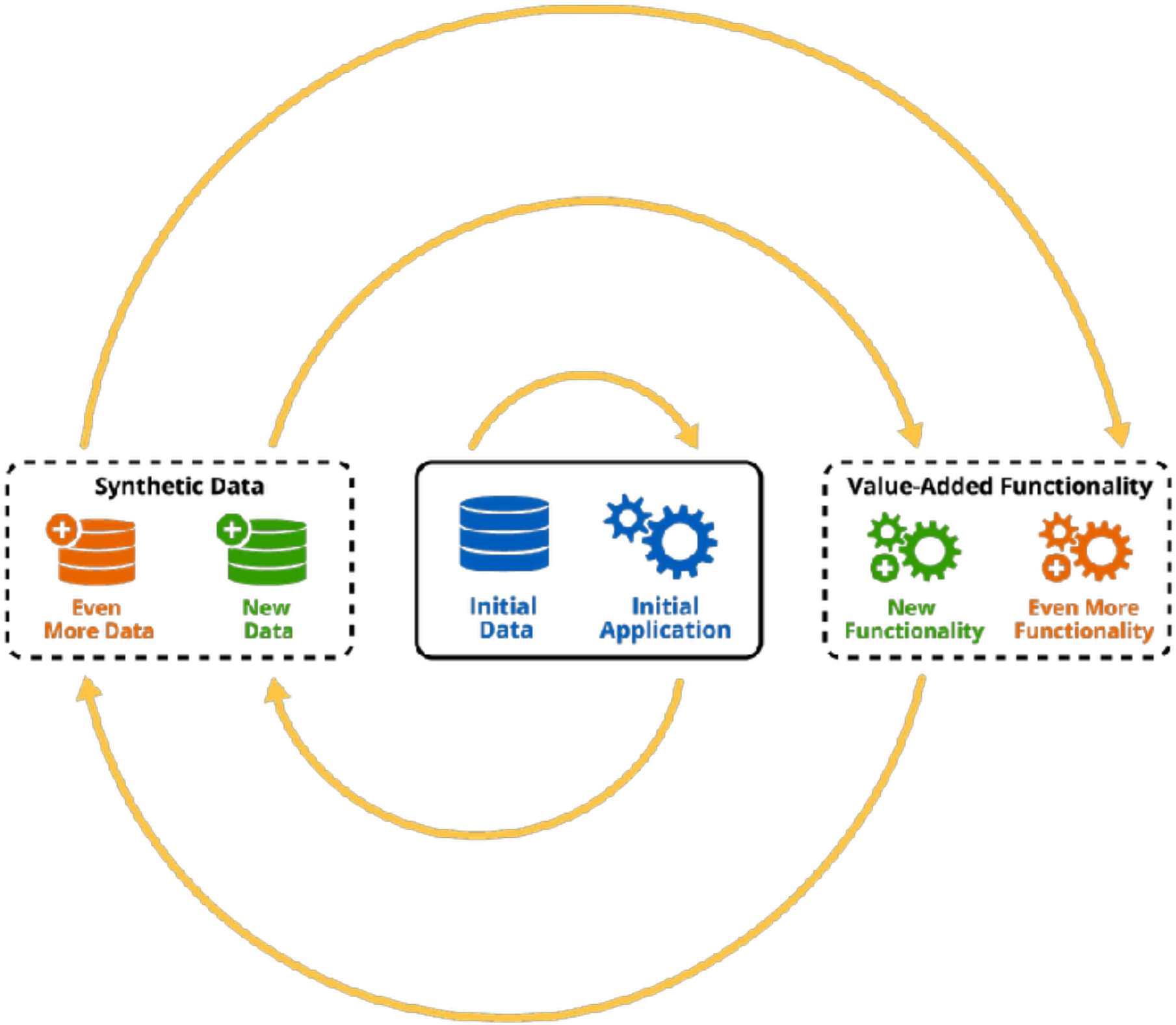
Enterprise Data
Lifecycle and transformations

AI/ML algorithms
Insightful programming

Applications
Intelligent applications

New paradigm enables data centric flexible, maintainable, self learning and adapting application design

Driving A Virtuous Data Cycle

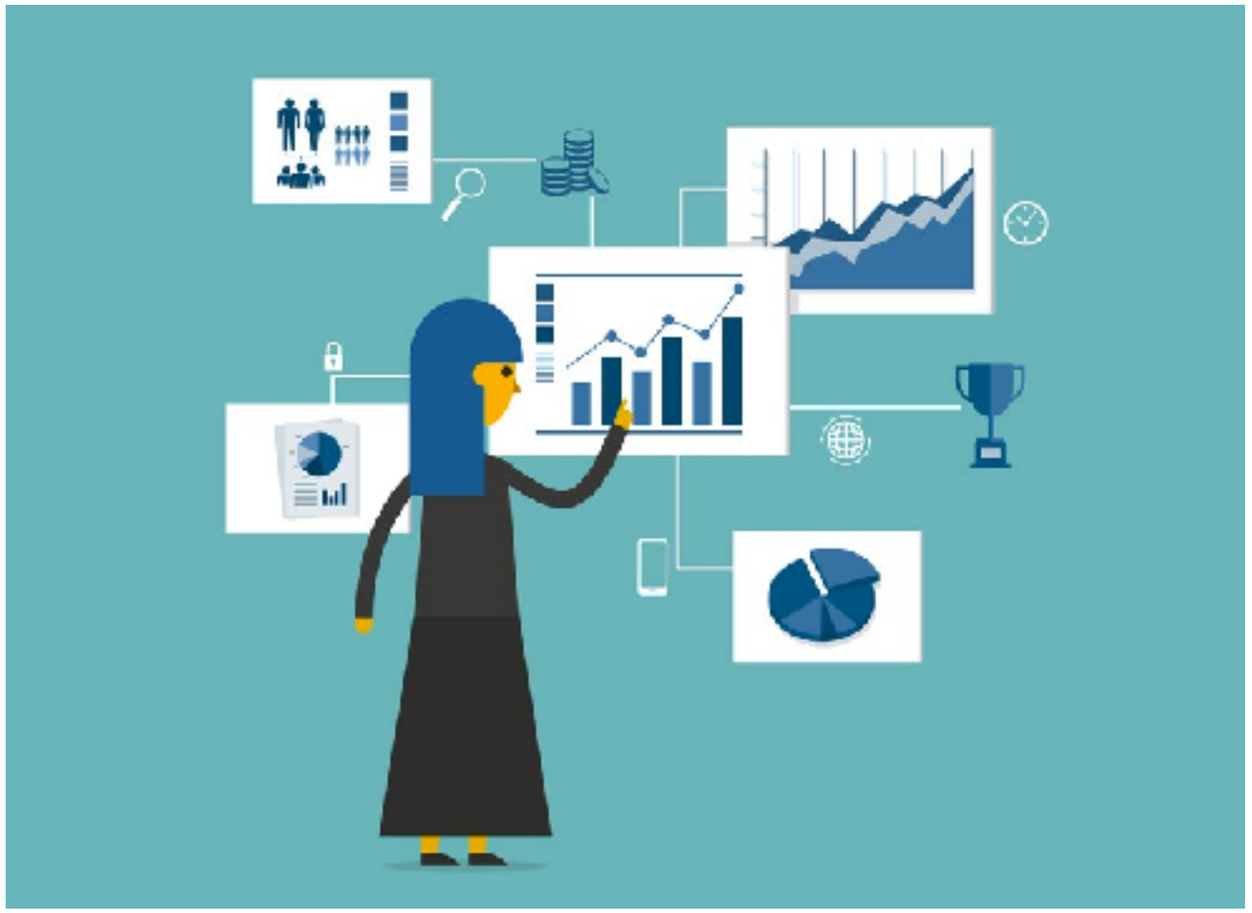


Data is like the life blood of business, processes are like body functions, made up of steps that convert inputs to outputs

Gartner®

Gartner Business Intelligence and Analytics Summit, the failure rate of BI and analytics projects was estimated at **70 percent**-plus worldwide. While disconcerting, sadly this is nothing new and represents a failure rate that has been relatively consistent for the last 20 years.

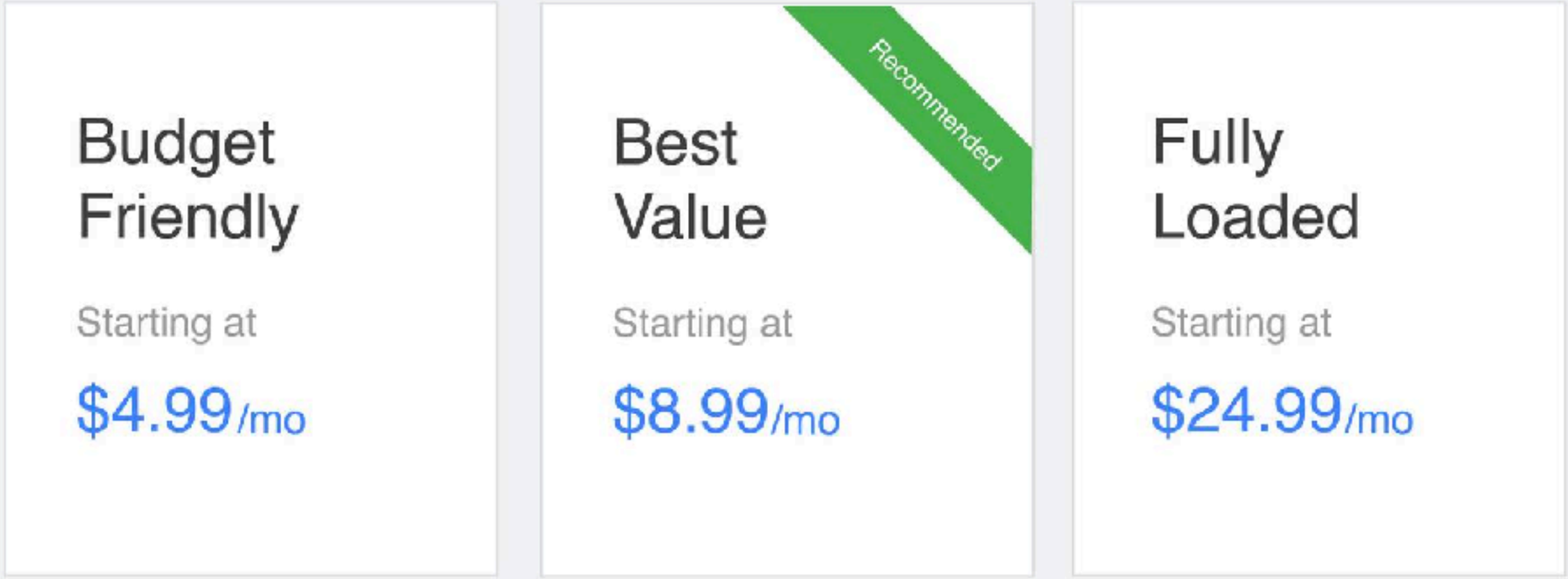
While organizations depend upon data for their day-to-day business operations, they fail to treat it like a strategic asset, in reality.



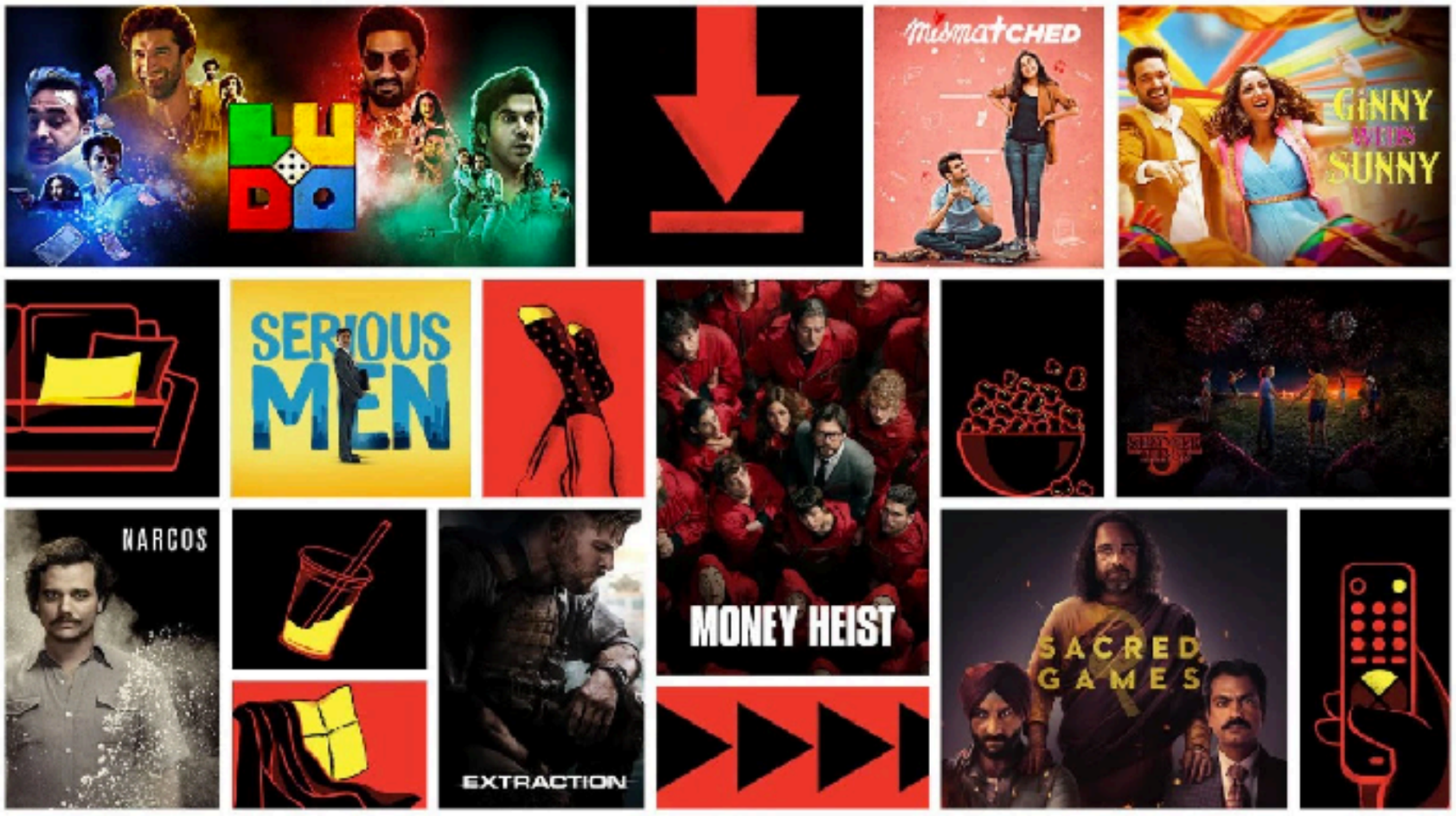
The ability to get data in front of as many users as possible.

The prior generation of business applications required specially trained analysts to query databases and then recommend actions to line-of-business users.

Slow doesn’t always mean no. It’s hard to deny that, when it comes to site speed, faster is far better. The longer people wait for your site to load, the more you’ll lose. But a study by Harvard Business School found that, due to what it called the “operational transparency,” people can tolerate — and, in some cases, prefer — websites with longer waits as long as there is an understanding of the work being performed. If you demonstrate that your site is exerting effort on a customer’s behalf (consider Domino’s Pizza tracker, which keeps you updated every step of the way in the journey to getting your pizza), it can contribute to having a stronger sense of loyalty and reciprocity toward your company.



- Facing the challenge of increased customer attrition, many service companies will start to recommend lower-cost pricing plans to their customers in an effort to demonstrate care for their customers and the greater benefits they can provide. Researchers from Columbia, Wharton, and IAE Business Schools found this tactic had the opposite effect: Encouraging customers to switch to cost-minimizing plans can actually increase churn. In some cases, it inspires customers to be less inert about making a change, and they start to look at other service providers.



By digitizing interaction with its 151 million subscribers, Netflix collects data from each of its customers to understand their behavior and watching patterns. Using this information, Netflix is able to recommend TV shows and movies customized as per the subscriber's interest and choices. Instituting a data-first culture helps Netflix make its viewers' job easier, giving them a better and customized viewer experience.

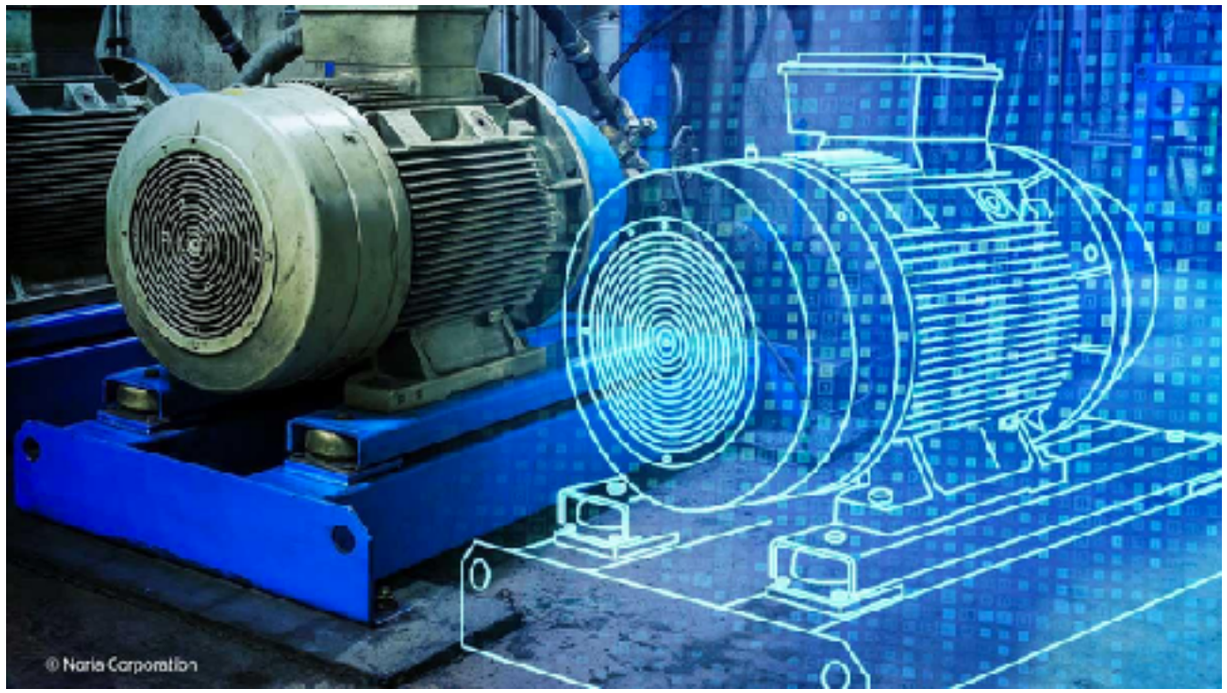
DigitalGenius uses neural networks to analyze the content of incoming emails, social media messages, and texts. The application leverages this information to keep learning about the best ways to handle inbound queries and suggests responses to them to customer support agents. An agent can then decide whether to send the machine-generated reply to a customer or to personalize it themselves.

DigitalGenius Solvvy TAKT.

Companies such as Solvvy and DigitalGenius ingest vast amounts of historical data about prior customer interactions via customer service transcripts, support forums, and other sources, and then use this to learn how best to deal with fresh requests for support.

“chatbots” resolving simpler queries and bringing a human operator into the loop only when more complex issues need to be resolved.

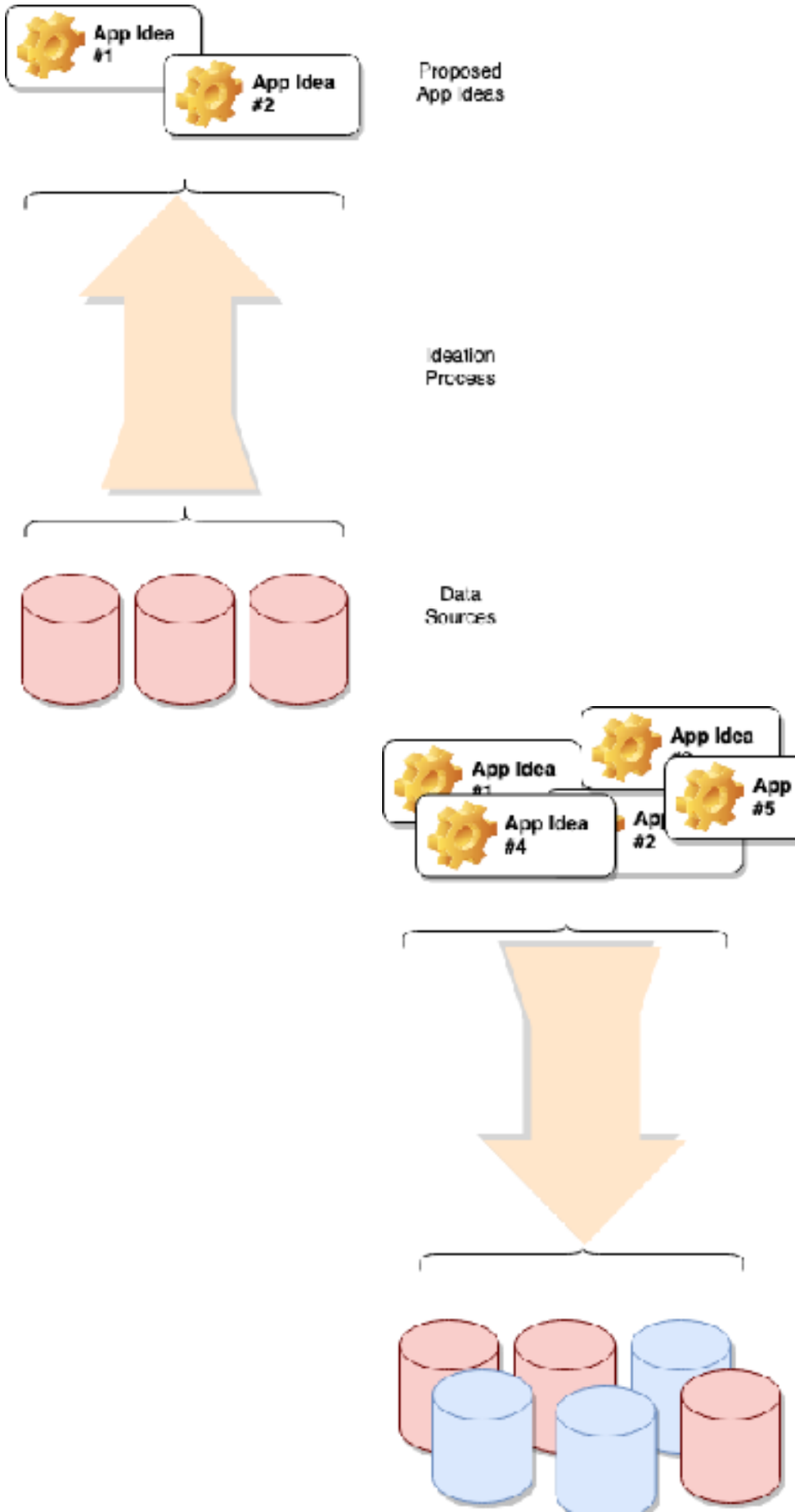
Chorus and Gong use a combination of speech recognition technology, natural language processing and other approaches to capture, transcribe and analyze sales conversations. That analysis drives insights into how future calls can be more effective and provides feedback that can be used by managers of sales teams to coach employees.

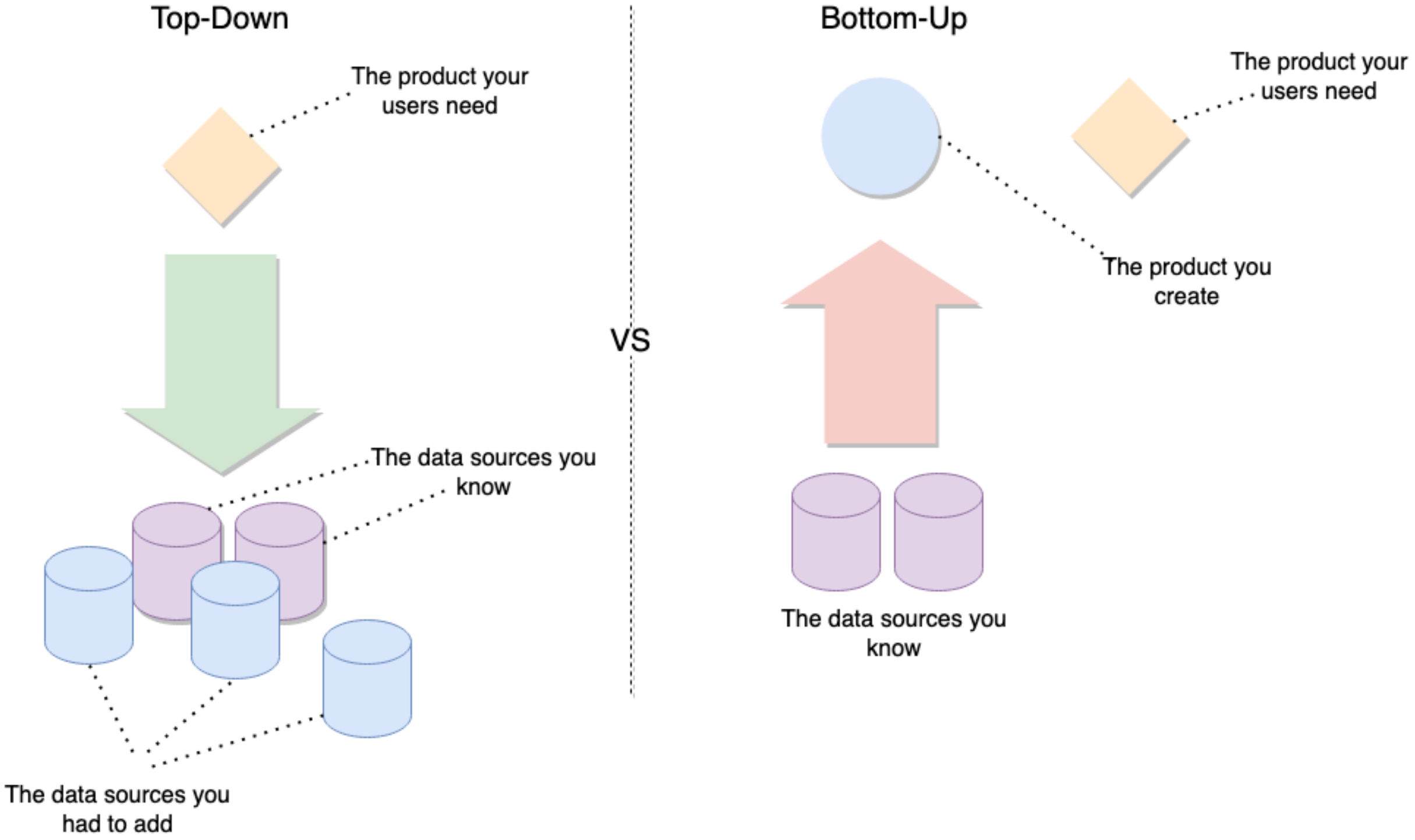


Sight Machine does this for manufacturers through their digital twin technology which takes raw data from plants, lines, machines and parts, interprets it, and uses additional inputs to provide a cross-process view of an operation.

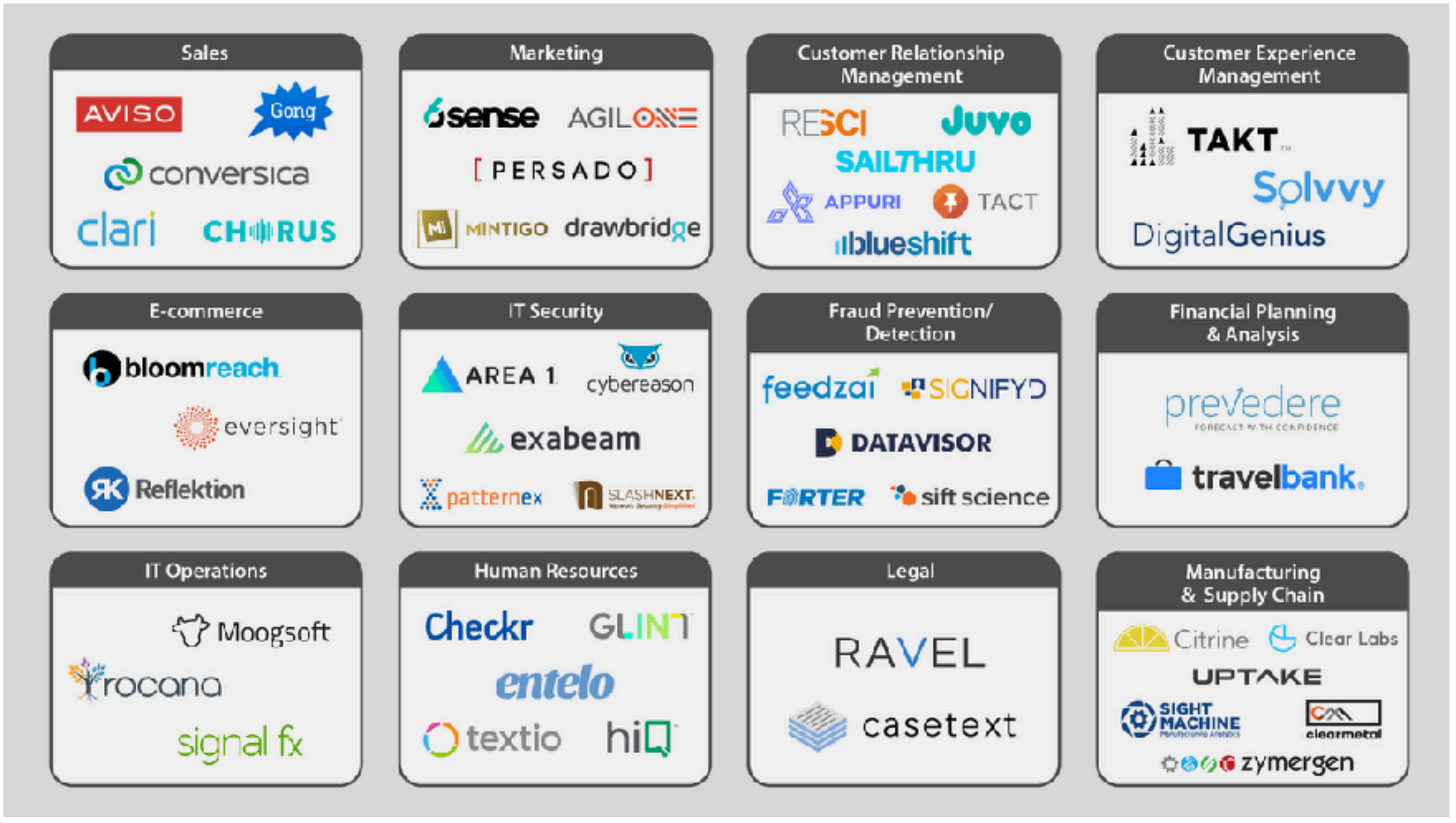
Just like a chemist creating new chemicals from ones formed in the past, we will now want to use data to create new synthetic distillations of the signal and new combinations that create yet new signals. You're in a state of experimentation in which you're testing different data sets and attempting to combine the signals in different ways to discover what happens.

- Bottom-up design: or in other words, you start at the data, look at it, and based on what you have, decide what you can build with it.
- Top-down design: or also known as listening to the user, then finding out what they want. With the information they give you, you decide what you're going to do and the data you'll need.




Top-Down
The product your users need
The data sources you know
The data sources you had to add
VS
Bottom-Up
The product your users need
The product you create
The data sources you know

50 Data First Apps



- https://www.wing.vc/content/announcing-the-wing-data-first-50-ai-powered-business-applications