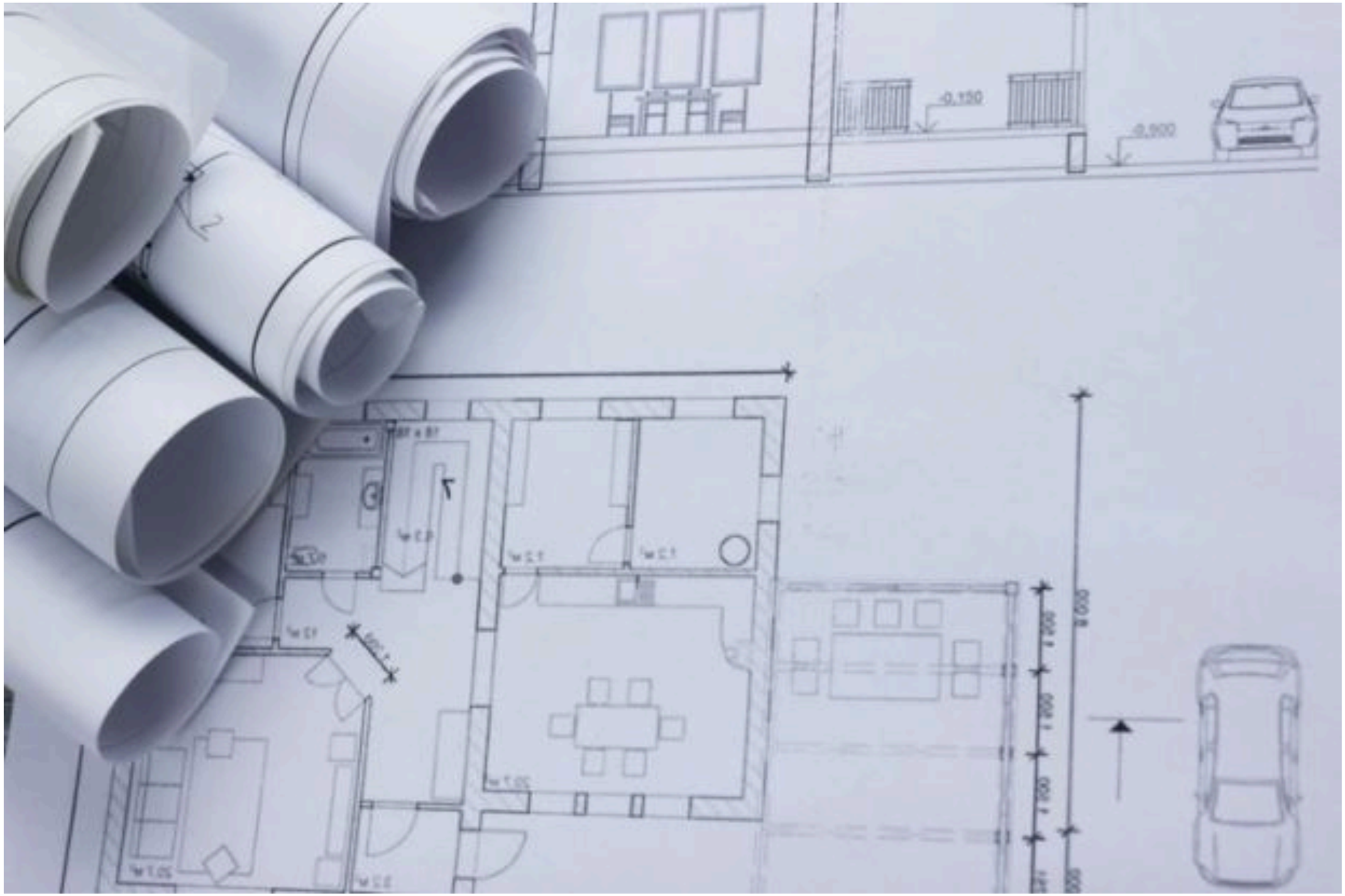


Software Architecture

» Skan.ai - chief Architect

» Ai.robotics - chief Architect

» Genpact - solution Architect

» Welldoc - chief Architect

» Microsoft

» Mercedes

» Siemens

» Honeywell

Mubarak

- Arch Requirements
- Arch Design
- Arch Doc
- Arch Eval

Attribute	Measure	Approach
• Performance <— • Maintainability <— • Security (Trust) <— • Scalability (Volume - I/O, CPU, Memory) <— • Usability x • Availability <— • Reliability (Trust) <— • Robustness (Rugud) •	• Response time • Memory • CPU • I/O • Latency • TPS • …	• Caching • Parallel • Pooling • Lazy Loading • Compression • Chunking • Reusability/ Extensible • Modular /Component • Low Coupling • EDA • ACID - transaction • Input Validation

Architecture Design
UI Design
Test Design
Module Design
Class Design
Code Design

Architecture vs Design

System quality

Code Maintainability

Detail Design
Module Design
Class Design
Low Level Design
Code design
Implementation Design

Pre

Engineering vs Tuning

Post

5 Communicate

Key functional requirements

2

Understands Domain

NFR

3

Quality attributes, Measure (Architectural Requirements)

Collect

4

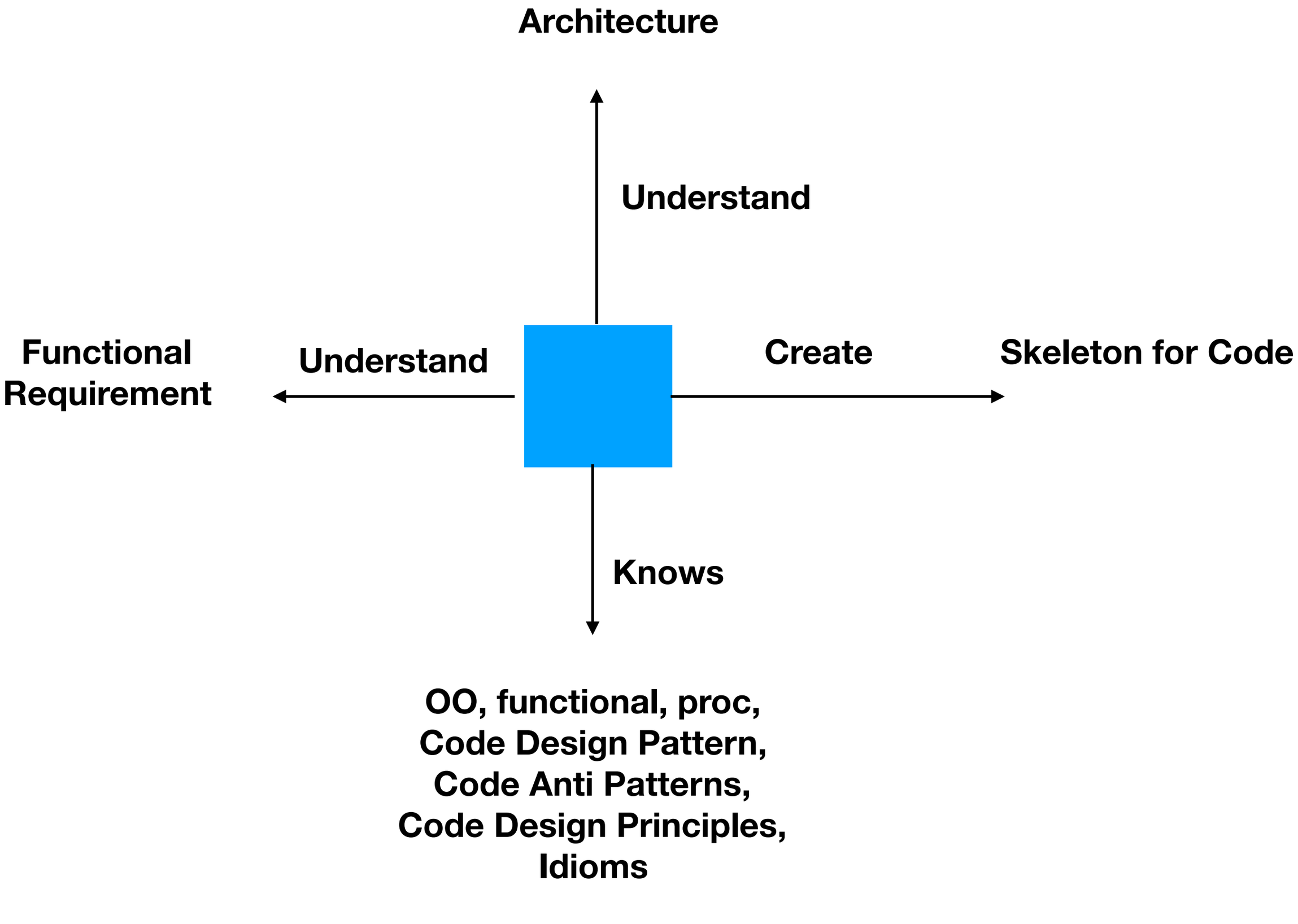
Choose

Approaches (Architectural Decisions)

1

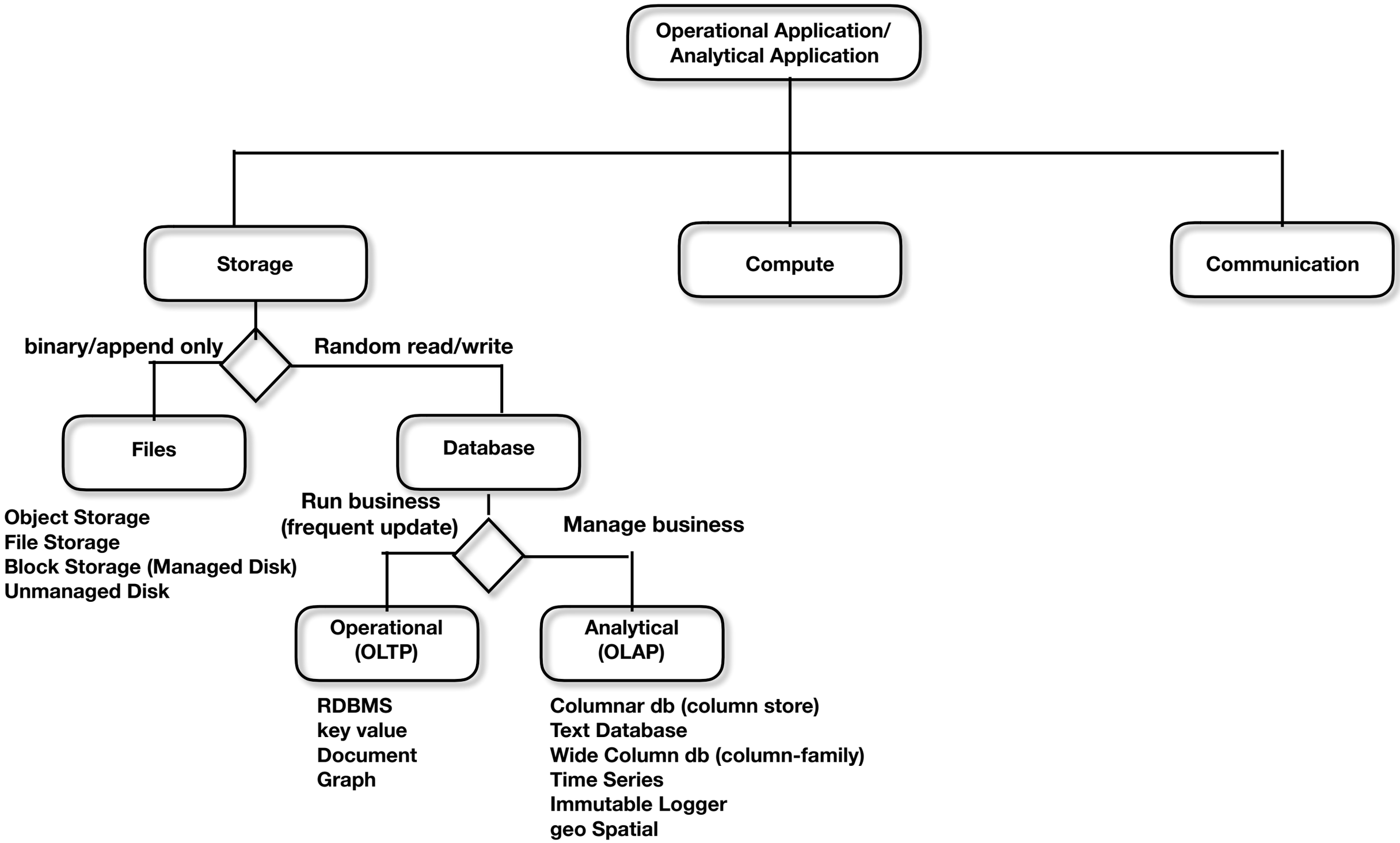
Knows

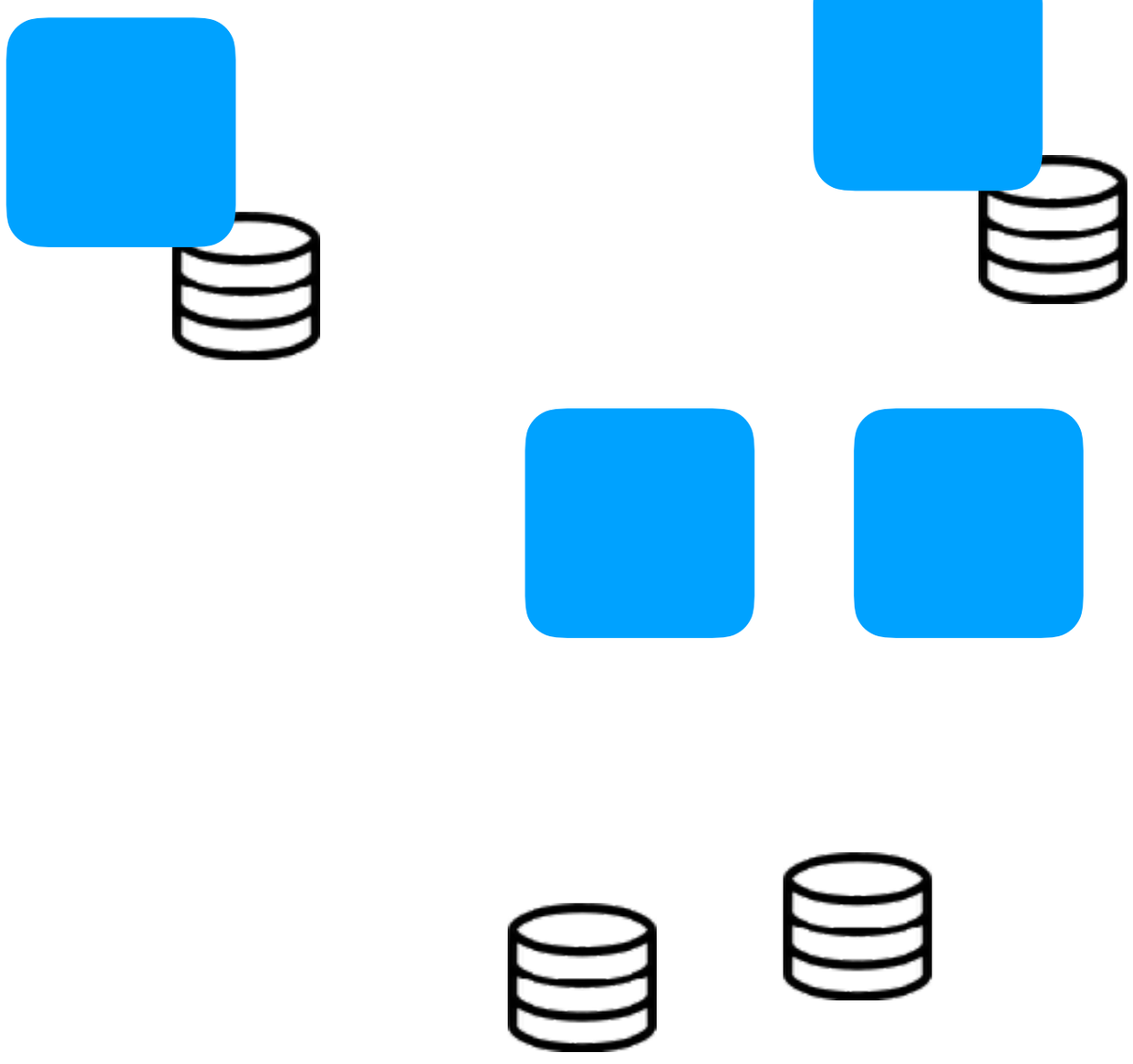
Arch Tactics, pattern, styles
Arch Principles
Arch anti patterns
Reference Arch (*)


Architecture
Understand
Functional Requirement
Understand
Create
Skeleton for Code
Knows
OO, functional, proc,
Code Design Pattern,
Code Anti Patterns,
Code Design Principles,
Idioms

Architect an Cloud App ?

- Binary (images, documents, videos,…)
- Append only
- Archive
- Scale
- Static content
- No random read





Binary Storage

Object Storage

Immutable
$
Low Perf
High Volume
0.01$/GB

- Azure Blobs
- AWS S3
- google Cloud Storage

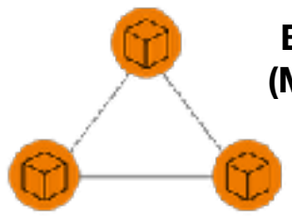
4

File Storage

Mutable
$$$$
Medium Perf (ex: 500 ms)
Medium Voulme

- Azure Files
- AWS EFS
- Google Filestore
- NFS

3



Block Storage
(Managed Disk)

Mutable
$$$
High Perf (10 ms)
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

Internet

Lan

Persistent

Temporary

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

Operational (OLTP)

Structured

**[ACID - transaction,
LIMITED SCALABLE - not distributed
Fixed Schema]**

RDBMS (OLTP)

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

1

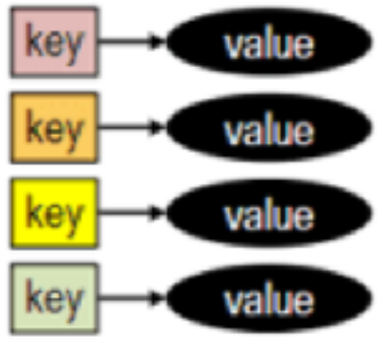
Semi Structured

**[* BASE,
SCALE- shareded (distributed),
* Flexible Schema]**

Dictionary (key value)

Persistent

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



2

Cache

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

Nested Objects

Document

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

3

Entity Relationship

Graph

- AWS Neptune
- Azure Cosmos -> Graph
- 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

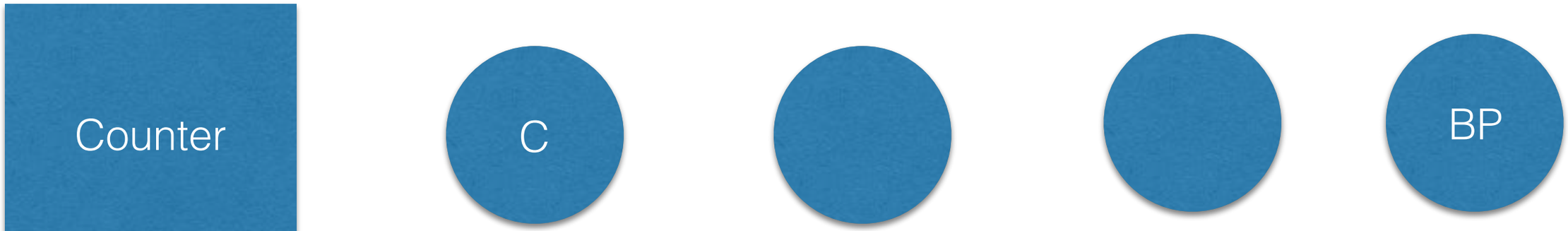
UserID	Name	Dob
1	John Smith	8/30/1964

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" }
   ]
}
```


Counter
C
BP

Counter

C

BP

Counter

E

BP

Counter

?

BP

General

C

E

?

BP

BP

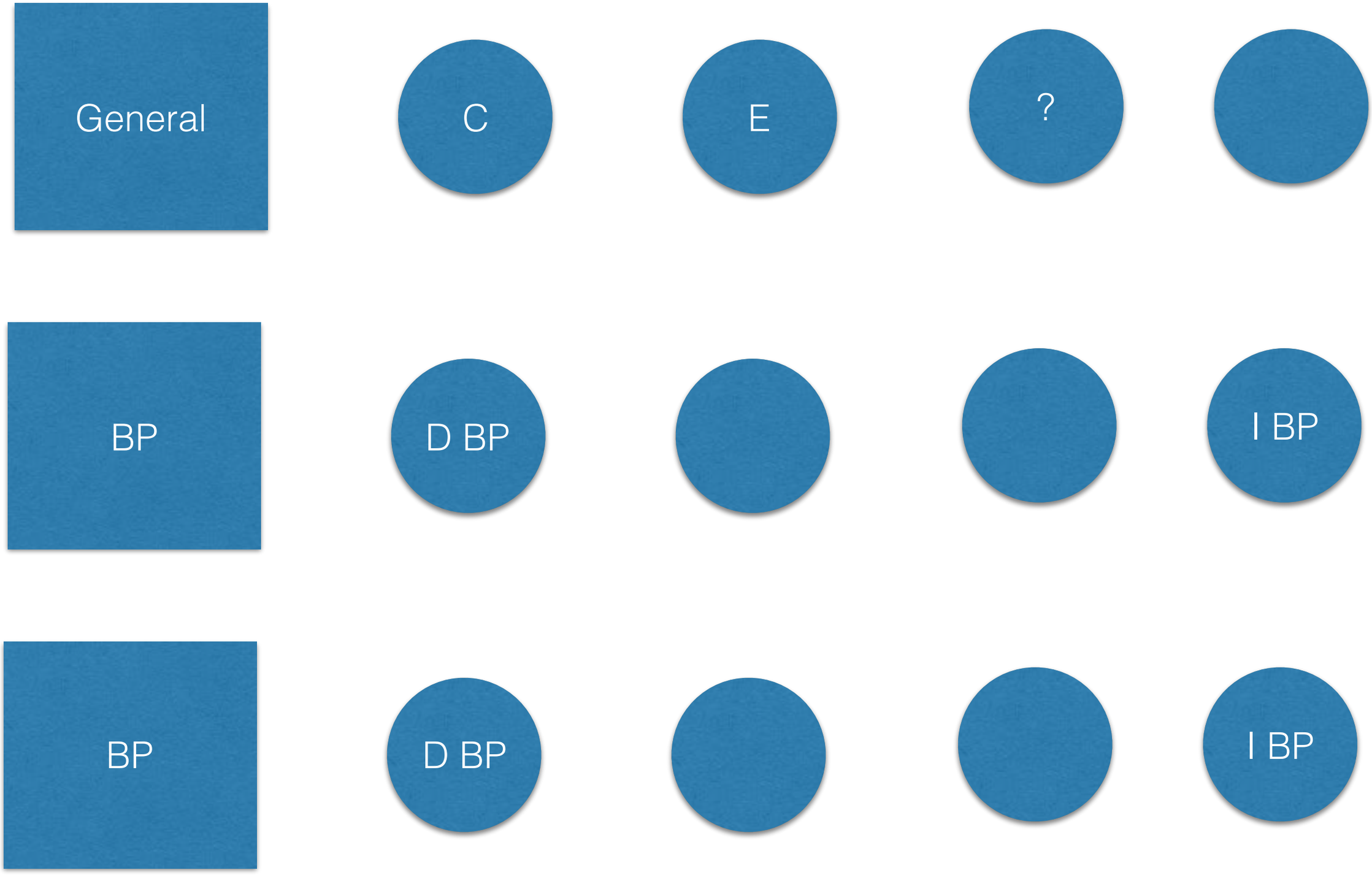
BP

BP

BP

BP

BP


General
C
E
?
BP
D BP
I BP
BP
D BP
I BP

General

C

E

?

D BP

D BP

D BP

D BP

D BP

D BP

D BP

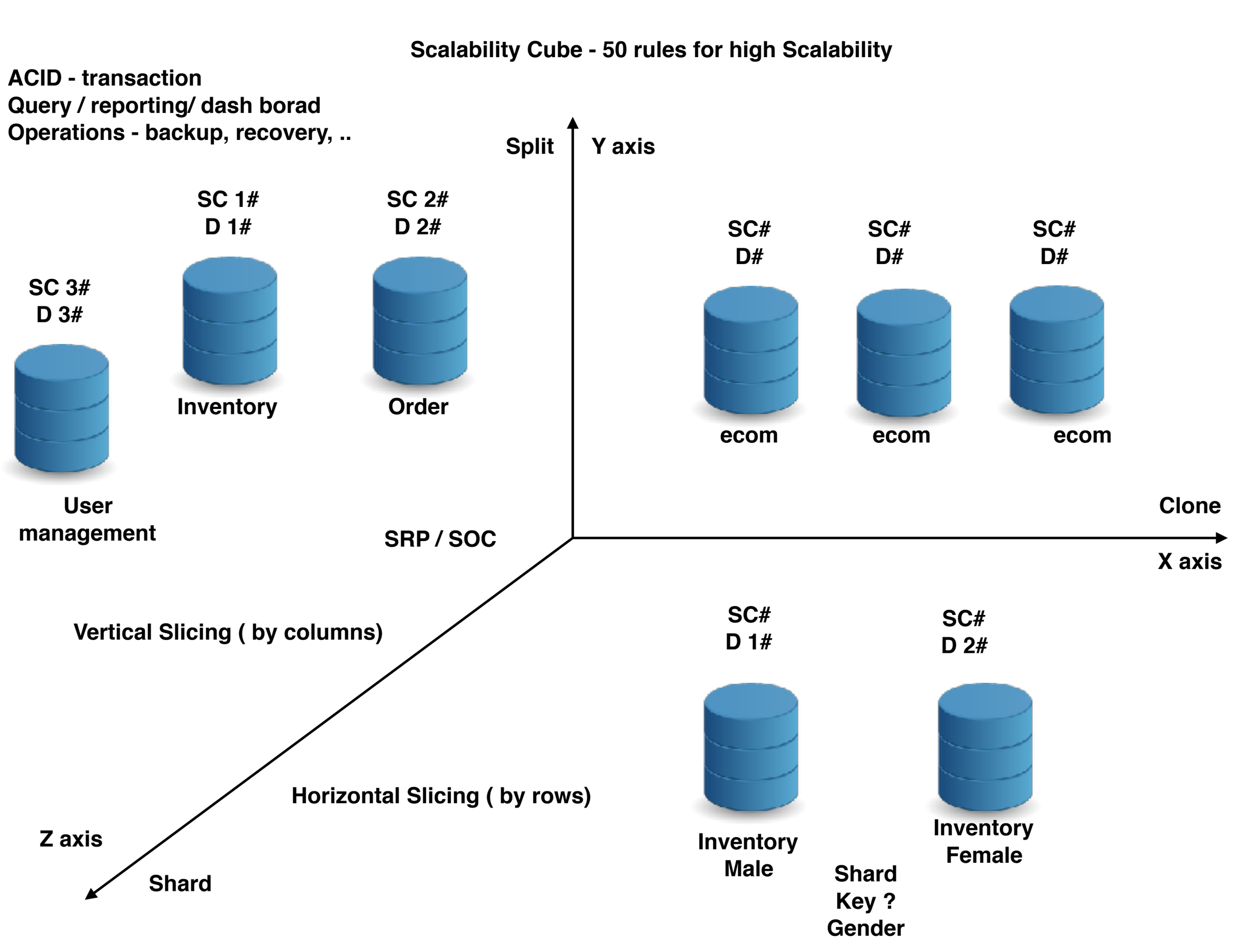
I BP

I BP

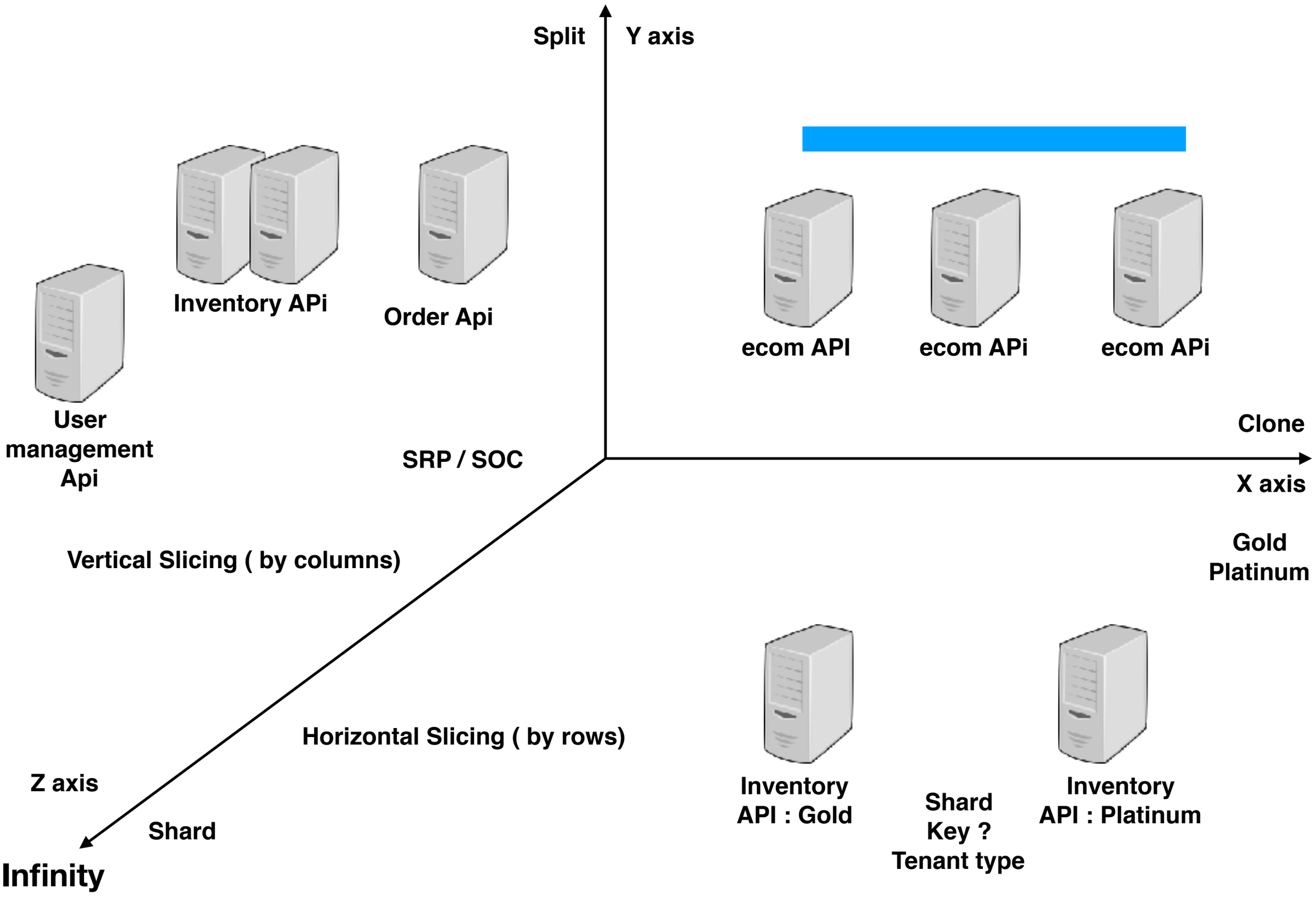
I BP

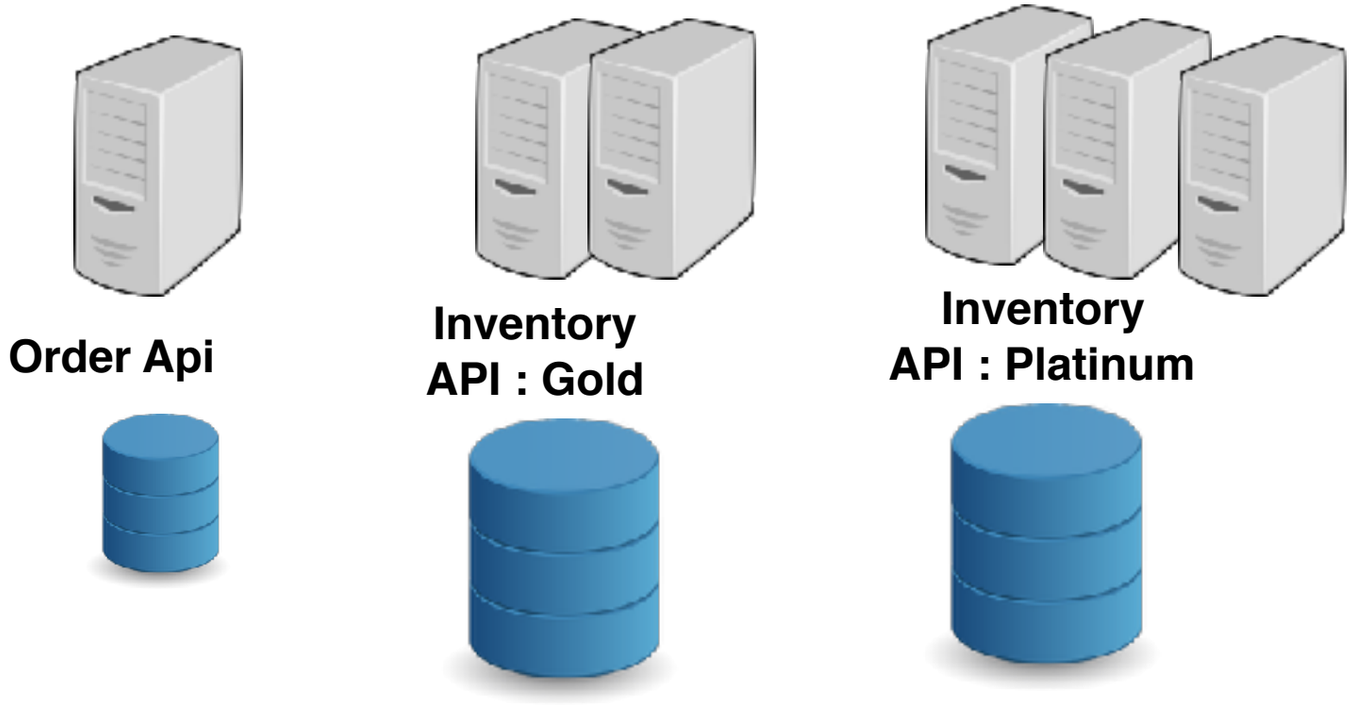
Scalability Cube - 50 rules for high Scalability

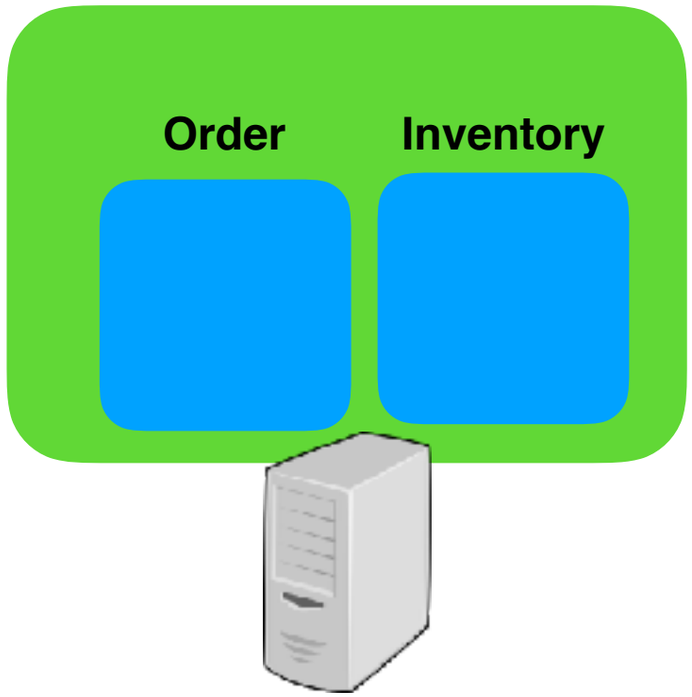
ACID - transaction
Query / reporting/ dash borad
Operations - backup, recovery, ..



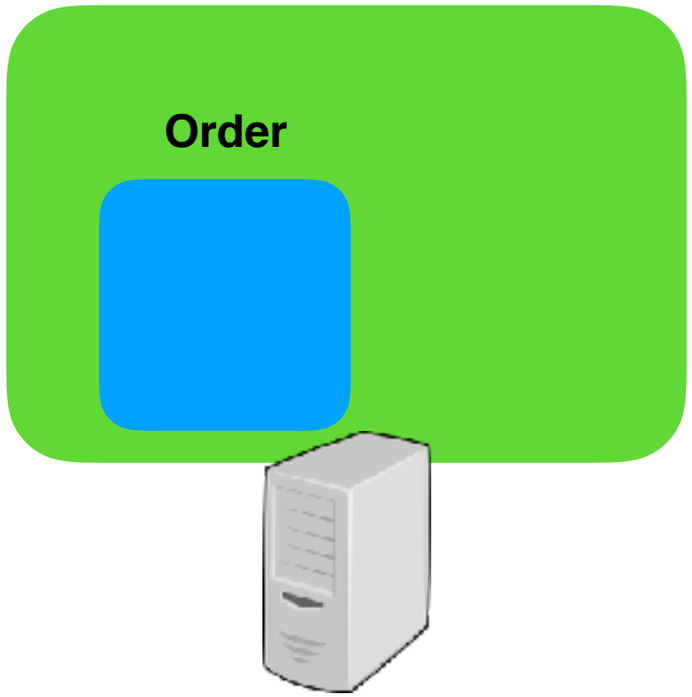
Scalability Cube - 50 rules for high Scalability



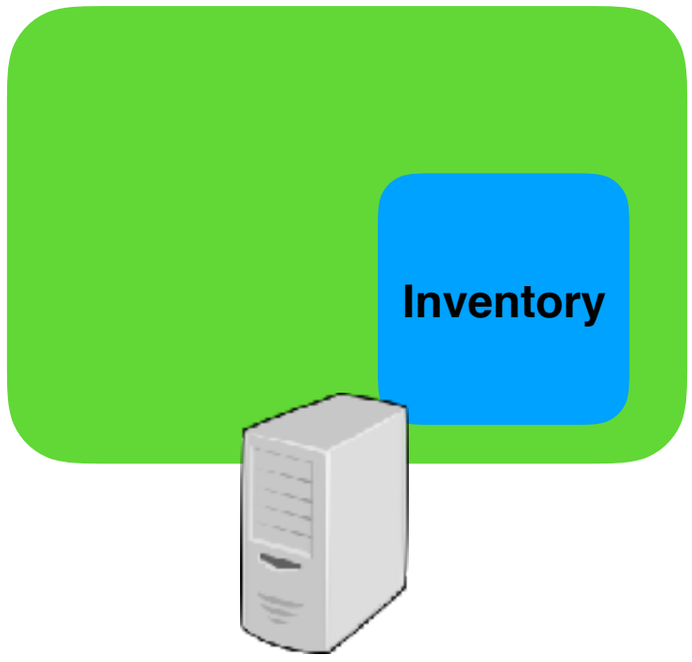

Order Api
Inventory
API : Gold
Inventory
API : Platinum



ecom

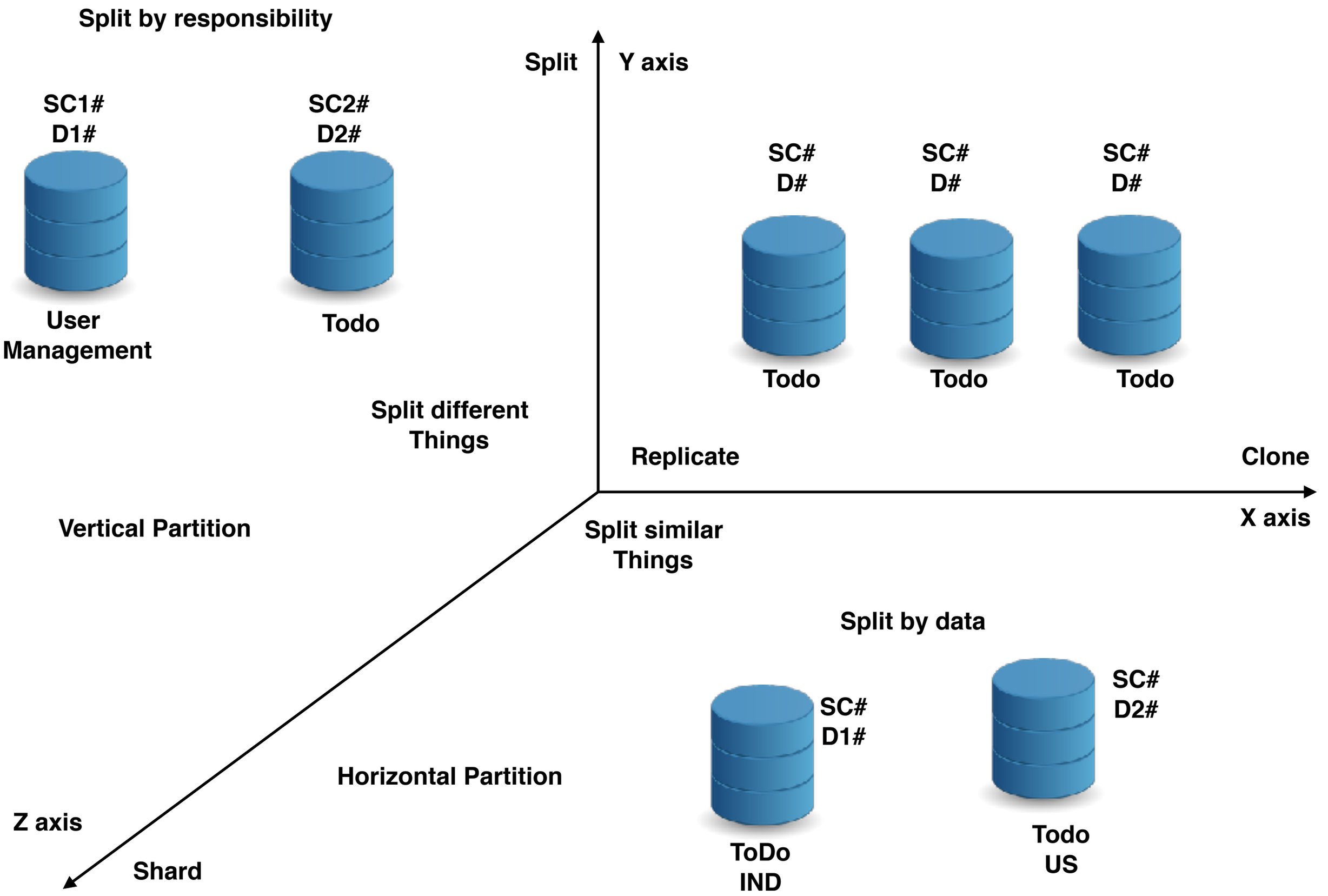


Order



Inventory

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

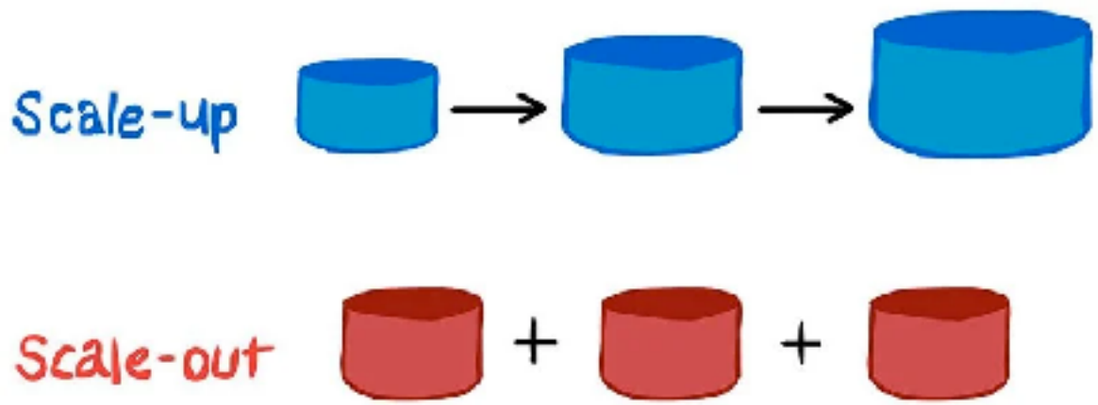
Order: USD

Clone

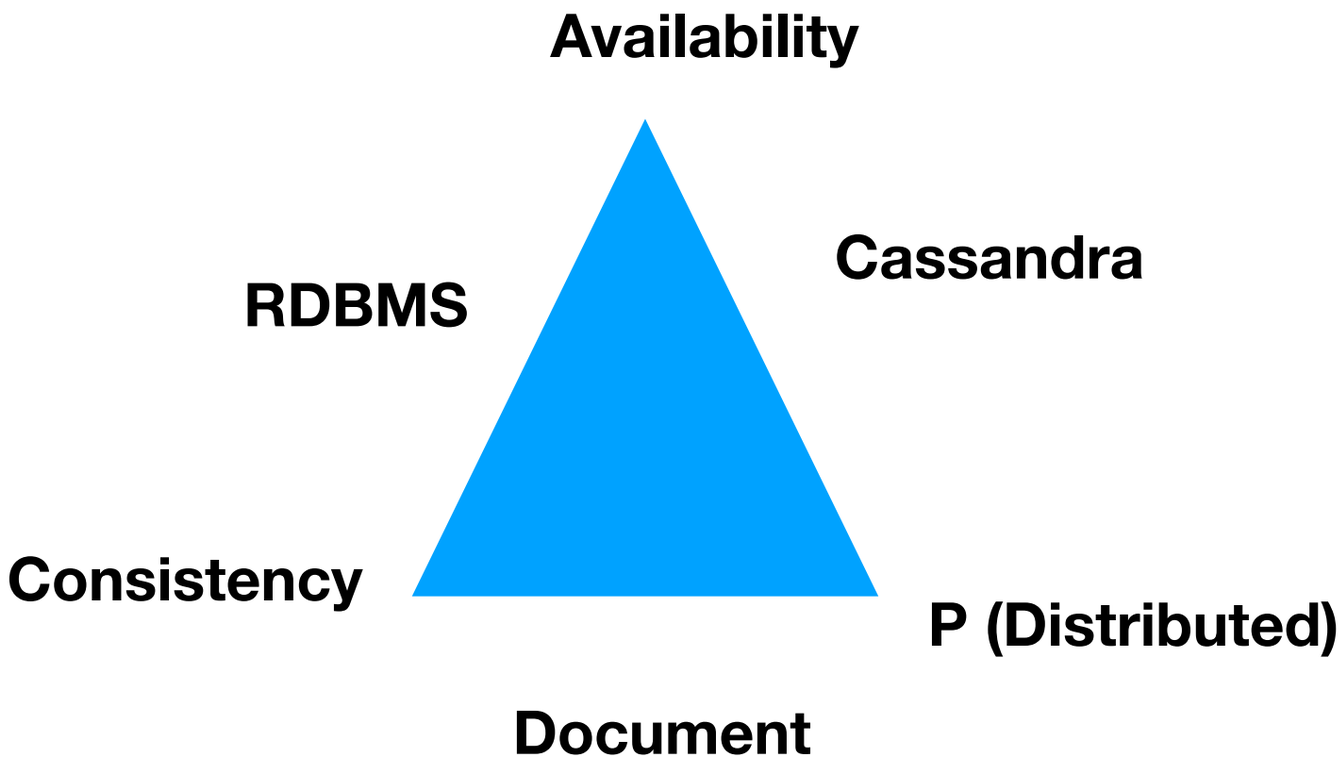
Inventory: M

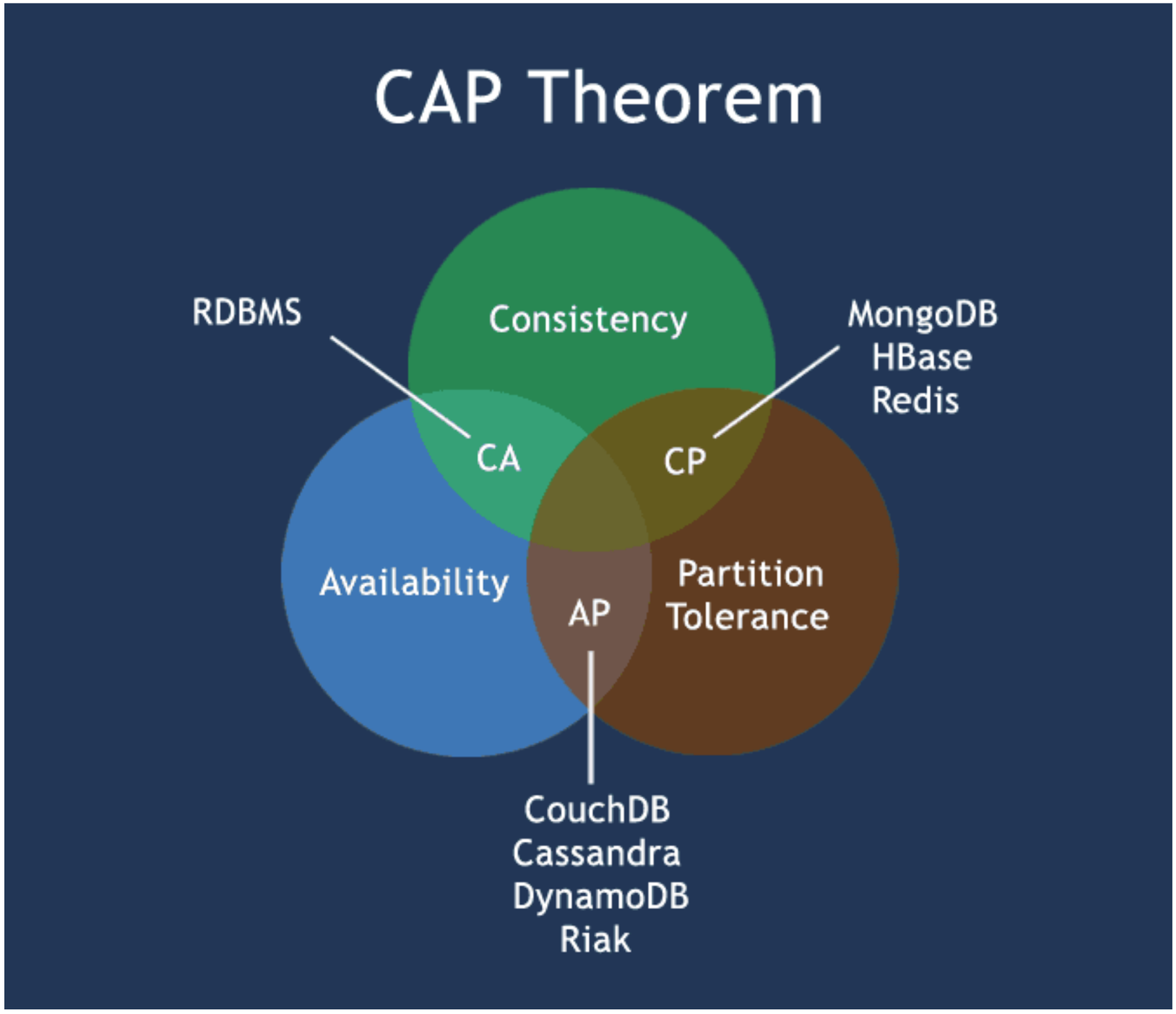
Inventory: M

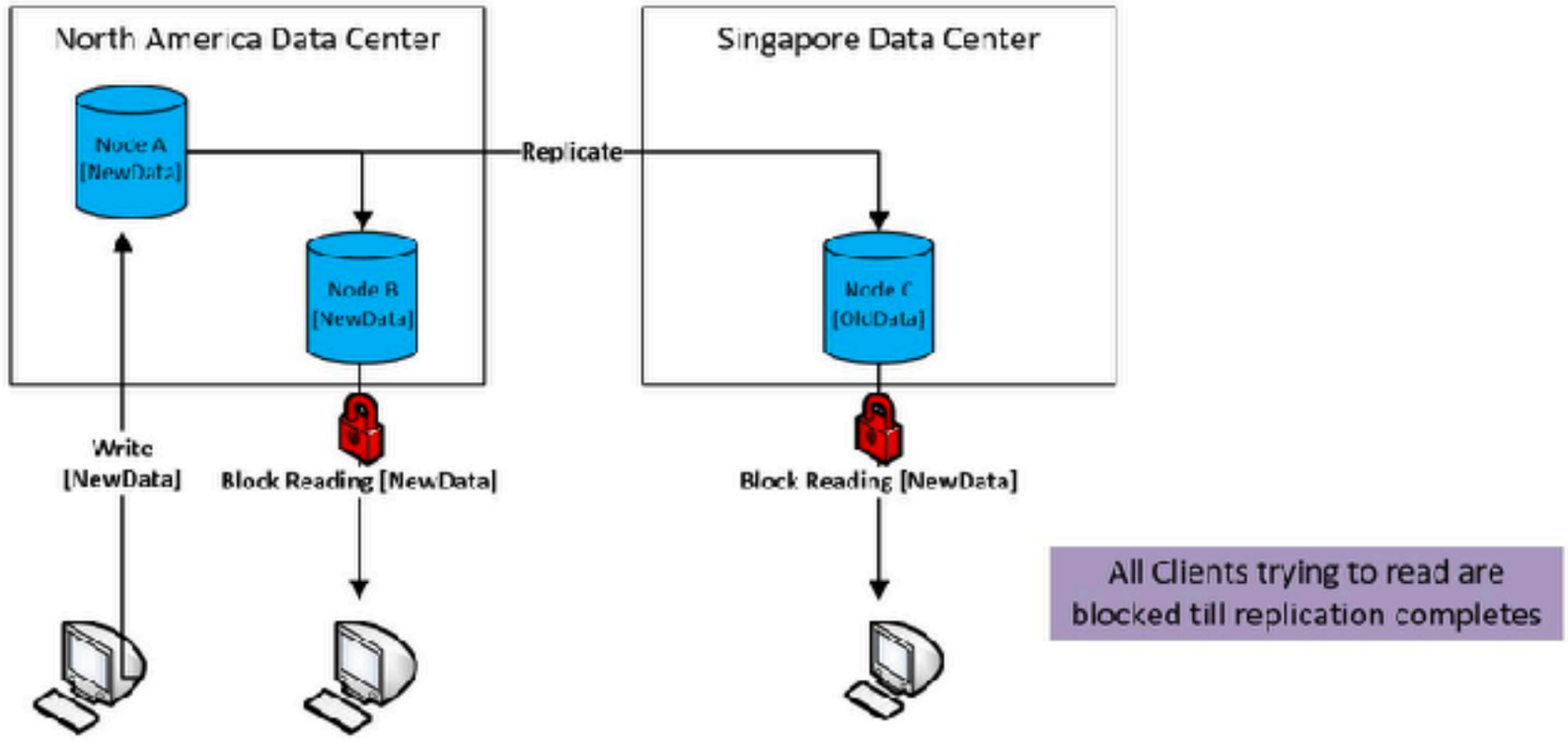
Scale-Up vs. Scale-Out

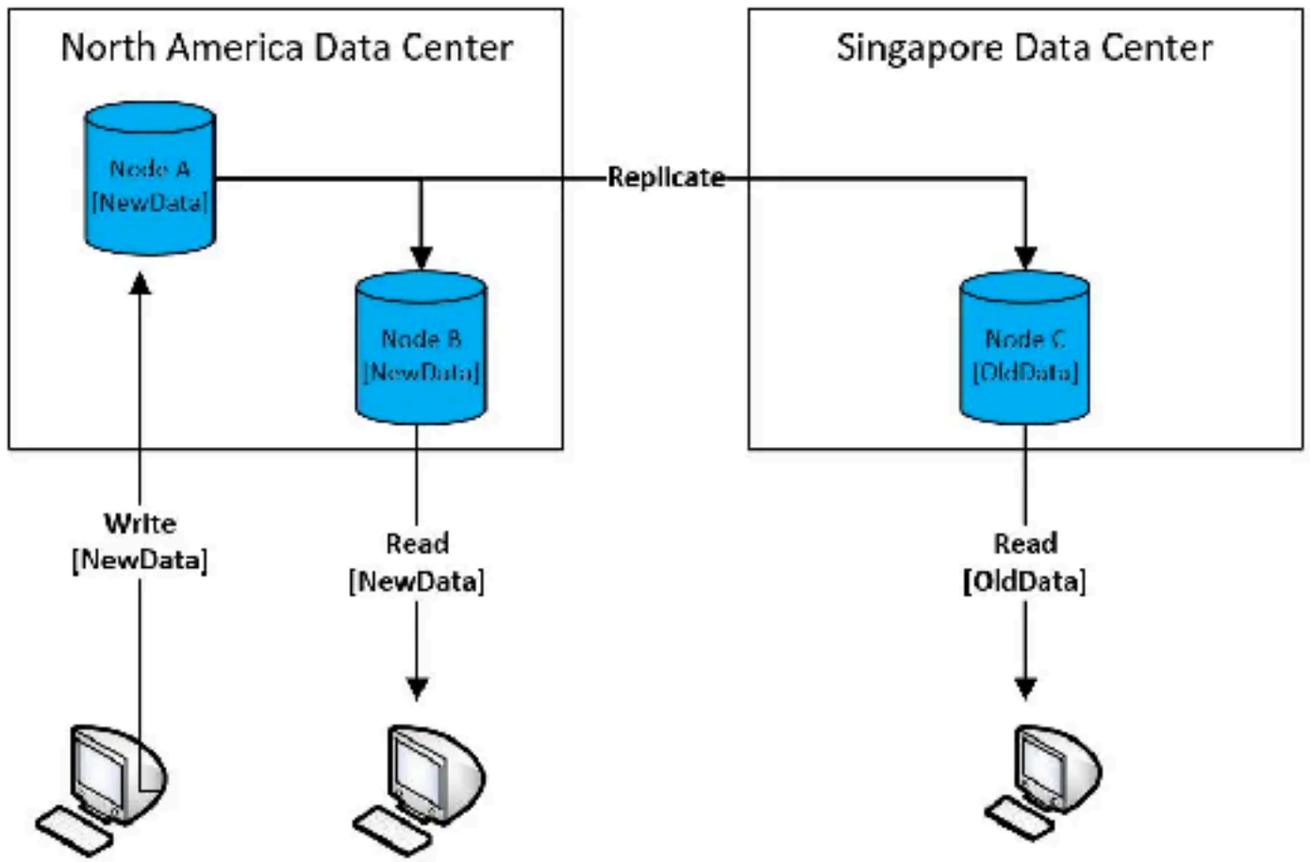


Strong Consistency vs. Eventual Consistency




CAP Theorem
RDBMS
Consistency
MongoDB
HBase
Redis
CA
CP
Availability
AP
Partition
Tolerance
CouchDB
Cassandra
DynamoDB
Riak


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

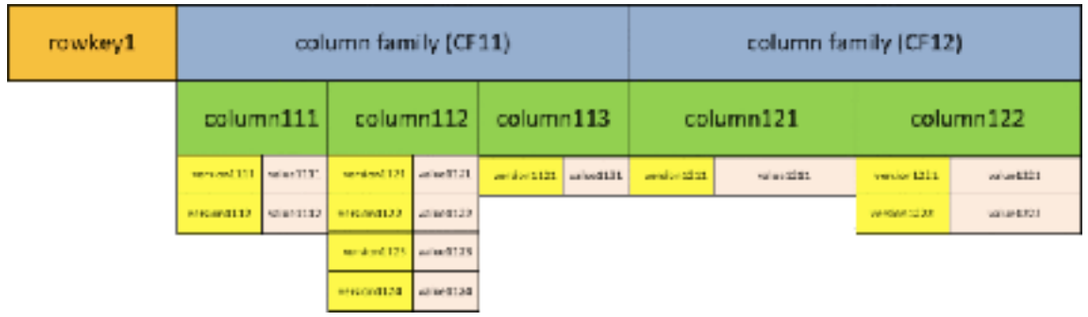
MYSQL & NOSQL

	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



Read

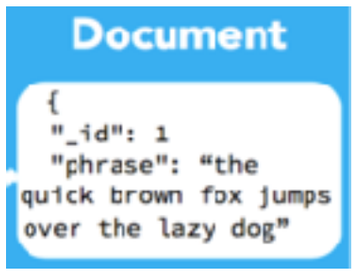
Columnar db (column store)

Cloud DWH (OLAP)

- AWS Redshit (max: 8 PB)
- **Azure Synapse**
- GCP BigQuery
- Snow flakes
- Kudu, Druid, Pinot
- Click house

Read

Text Database



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

Read/write

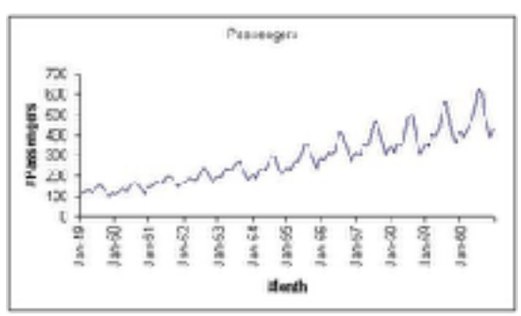
Wide Column db (column-family)

- AWS Keyspaces (unlimited)
- Cosmos
- Big Table
- **Hbase, cassandra**
- Scylla

KairosDB (cassandra)

Time Series

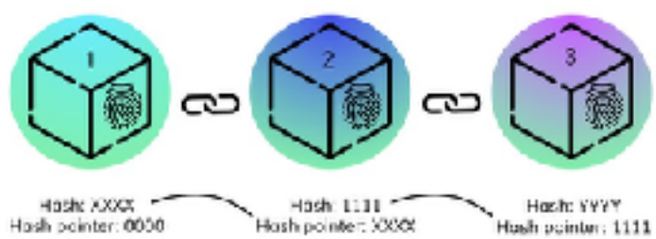
- AWS Timestream
- InfluxDB, Scylla
- OpenTSDB,
- KairosDB



Audit Trial

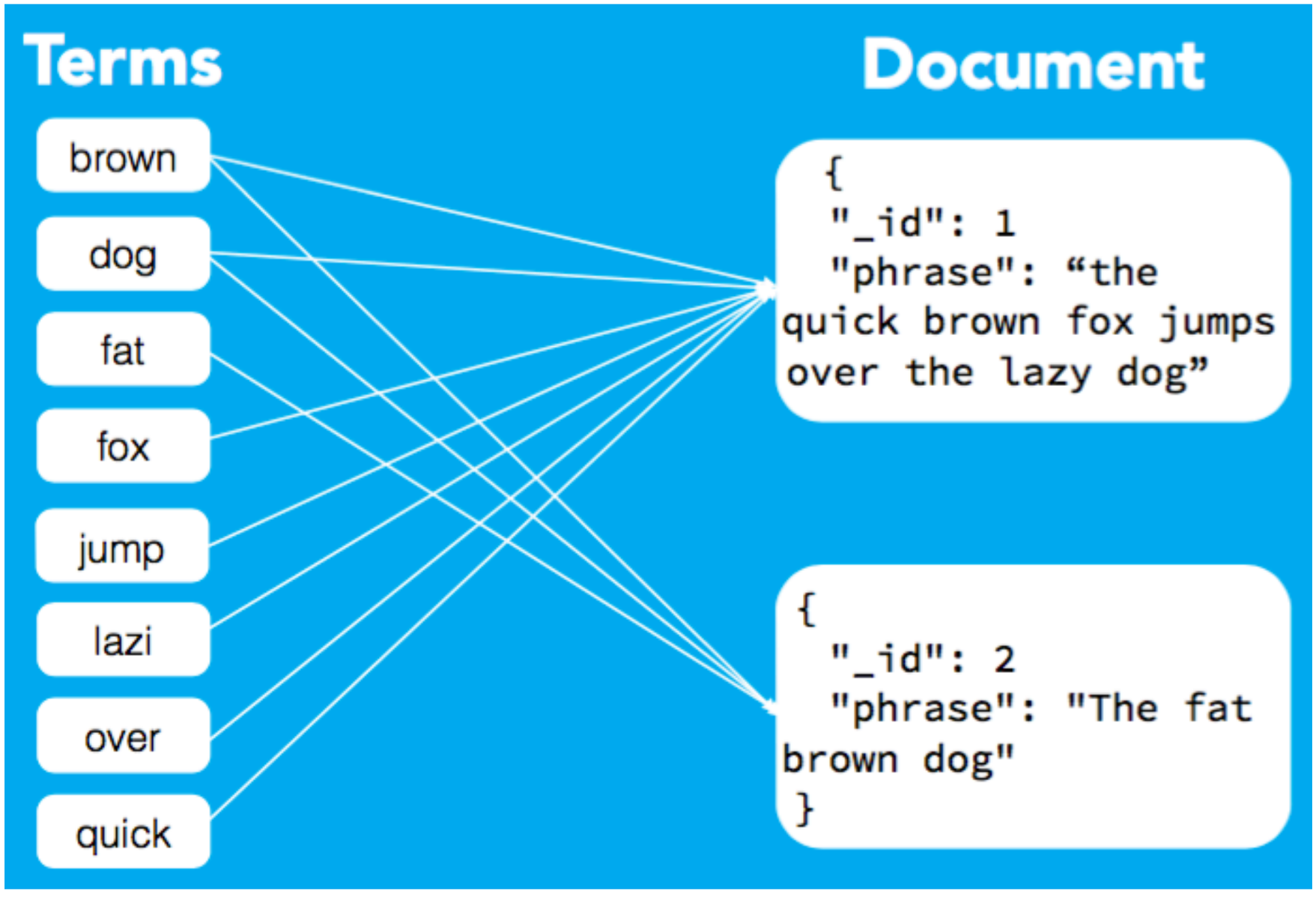
Immutable Logger

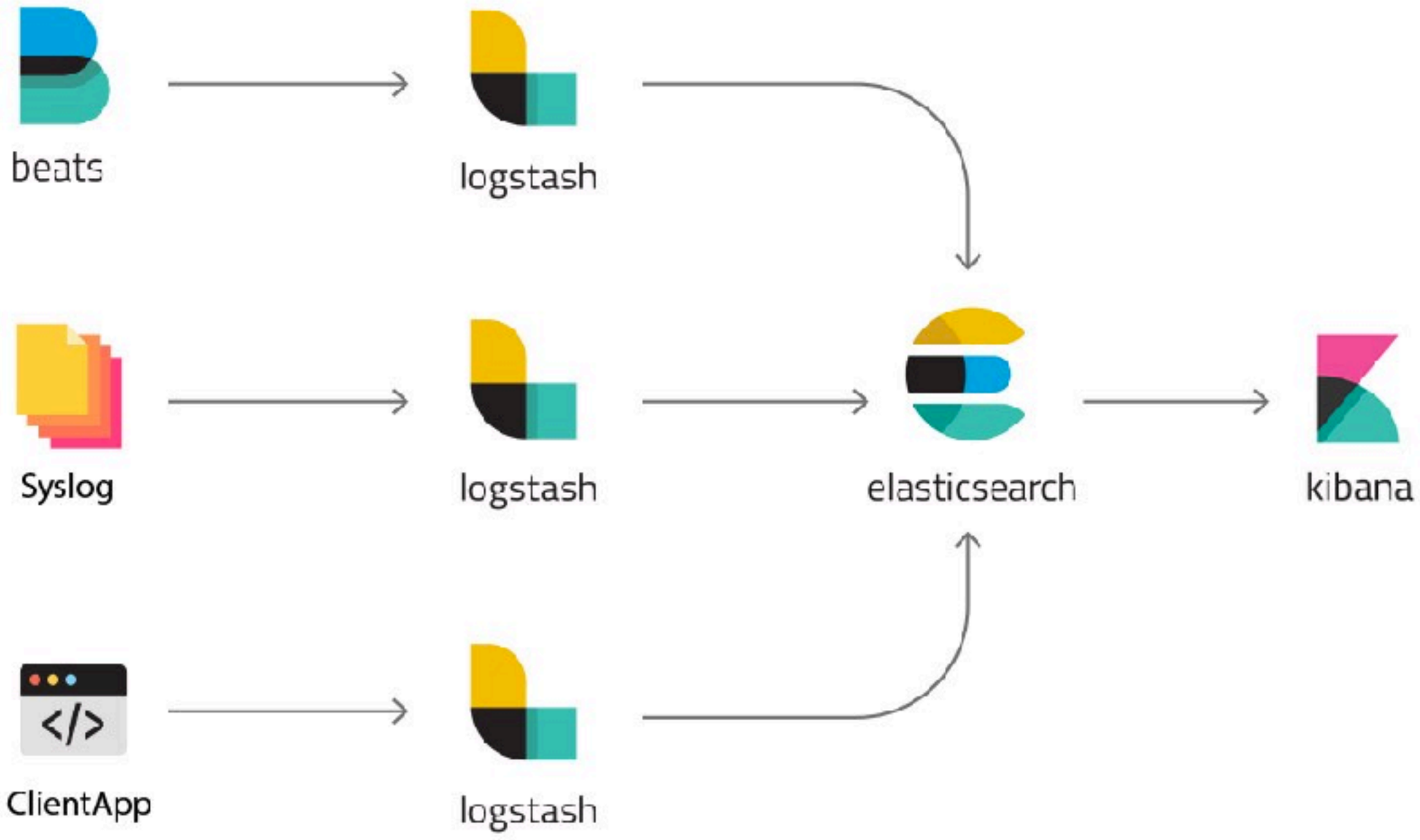
- AWS Quantum Ledger
- Azure SQL Database Ledger
- Google ?
- Hyper ledger Fabric

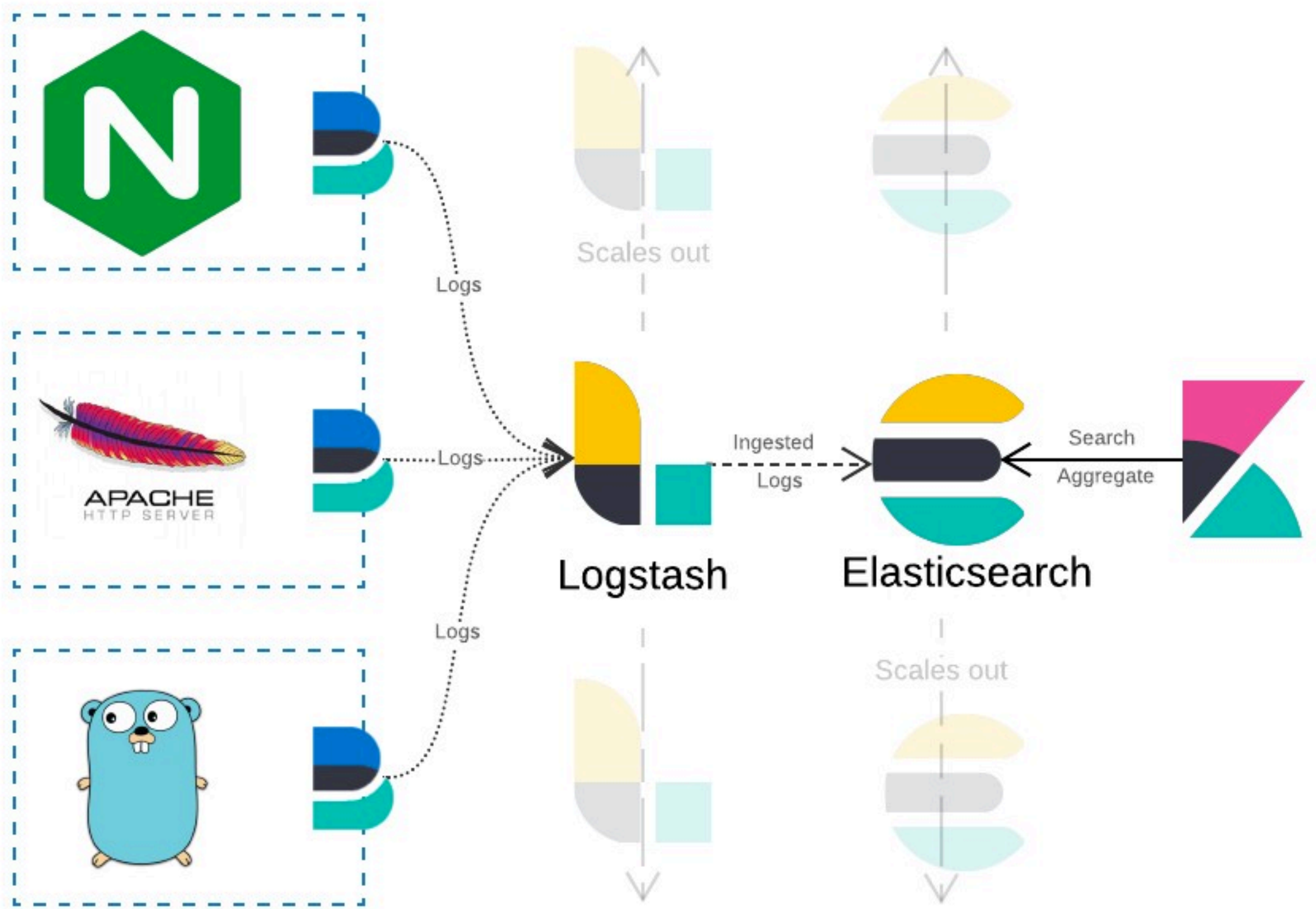


geo Spatial

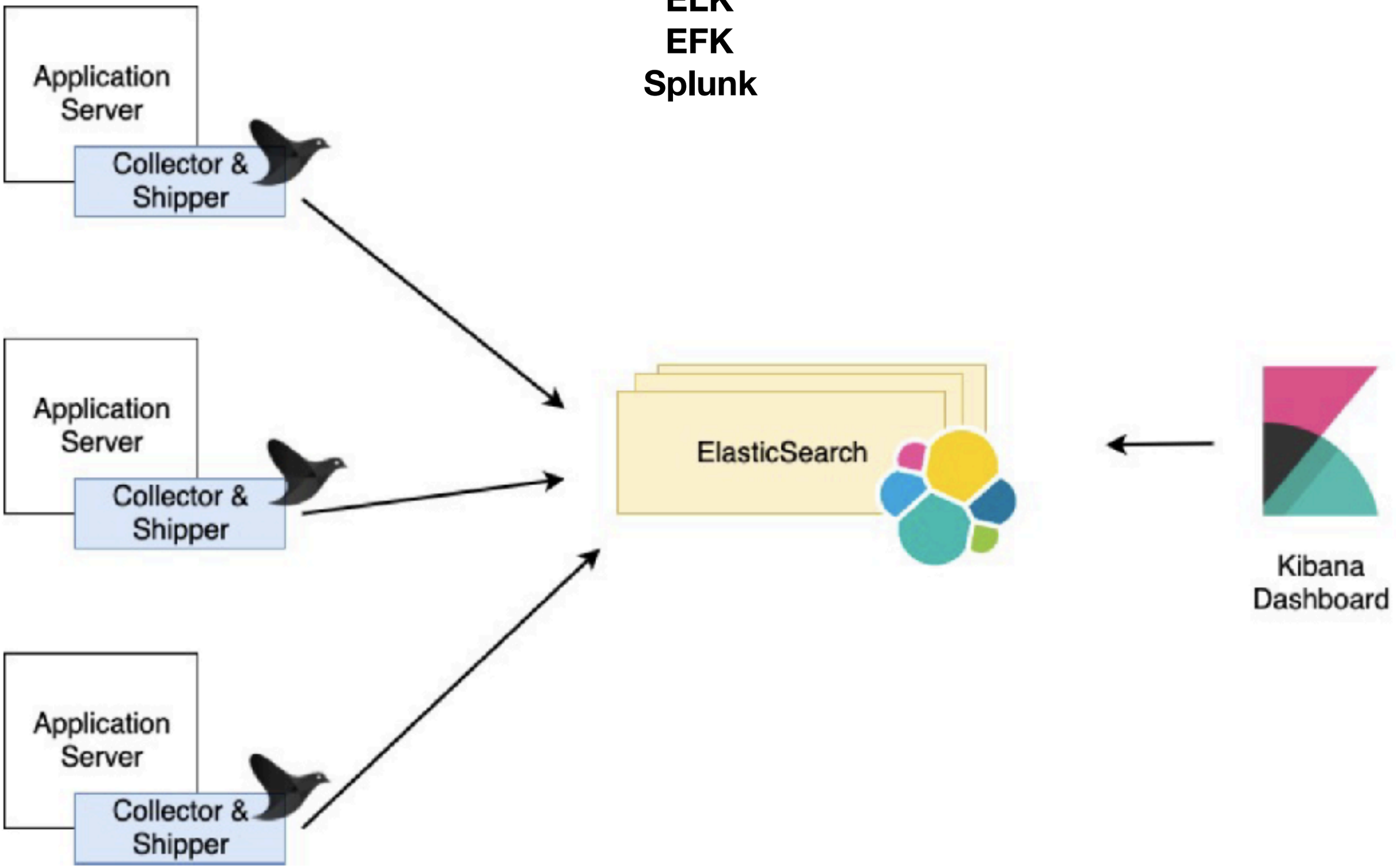
- AWS Keyspaces
- Azure Cosmos
- Big Table, Big Query
- **Solr**, PostGIS
- Mongoldb (GeoJSON)


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"
}


beats
logstash
Syslog
logstash
elasticsearch
kibana
ClientApp
logstash


Logs
Scales out
APACHE
HTTP SERVER
Logs
Ingested
Logs
Search
Aggregate
Logstash
Elasticsearch
Logs
Scales out

ELK
EFK
Splunk



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

Row-oriented

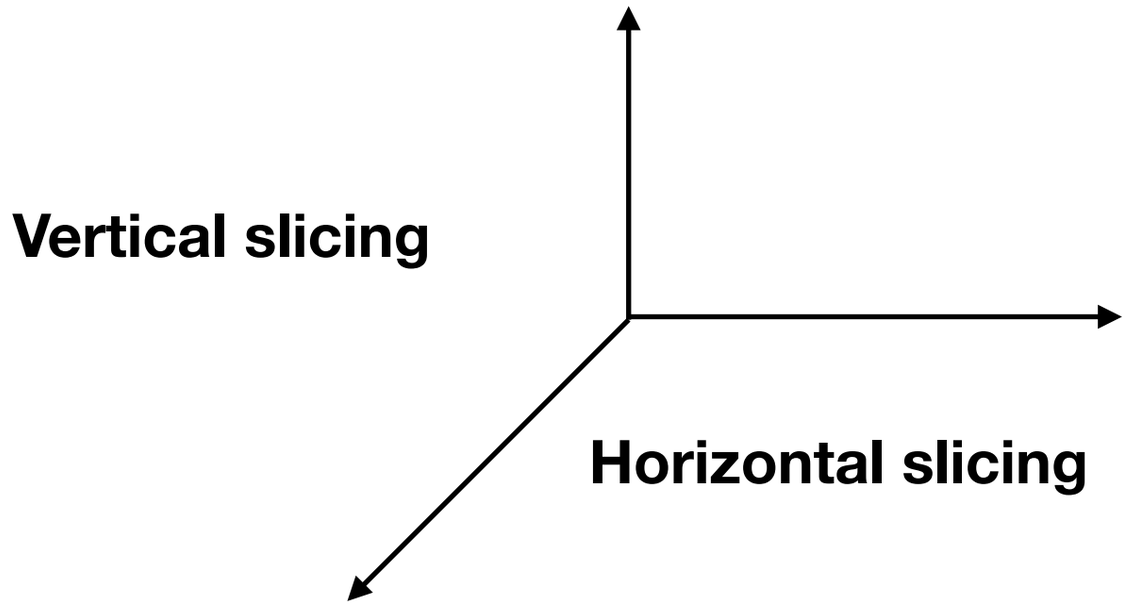
ID	Name	Grade	GPA
001	John	Senior	4.00
002	Karen	Freshman	3.67
003	Bill	Junior	3.33

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



rowkey1	column family (CF11)			column family (CF12)	
	column111	column112	column113	column121	column122
	version1111 value1111	version1121 value1121	version1121 value1131	version1211 value1211	version1221 value1221
	version1112 value1112	version1122 value1122			version1222 value1222
		version1123 value1123			
		version1124 value1124			

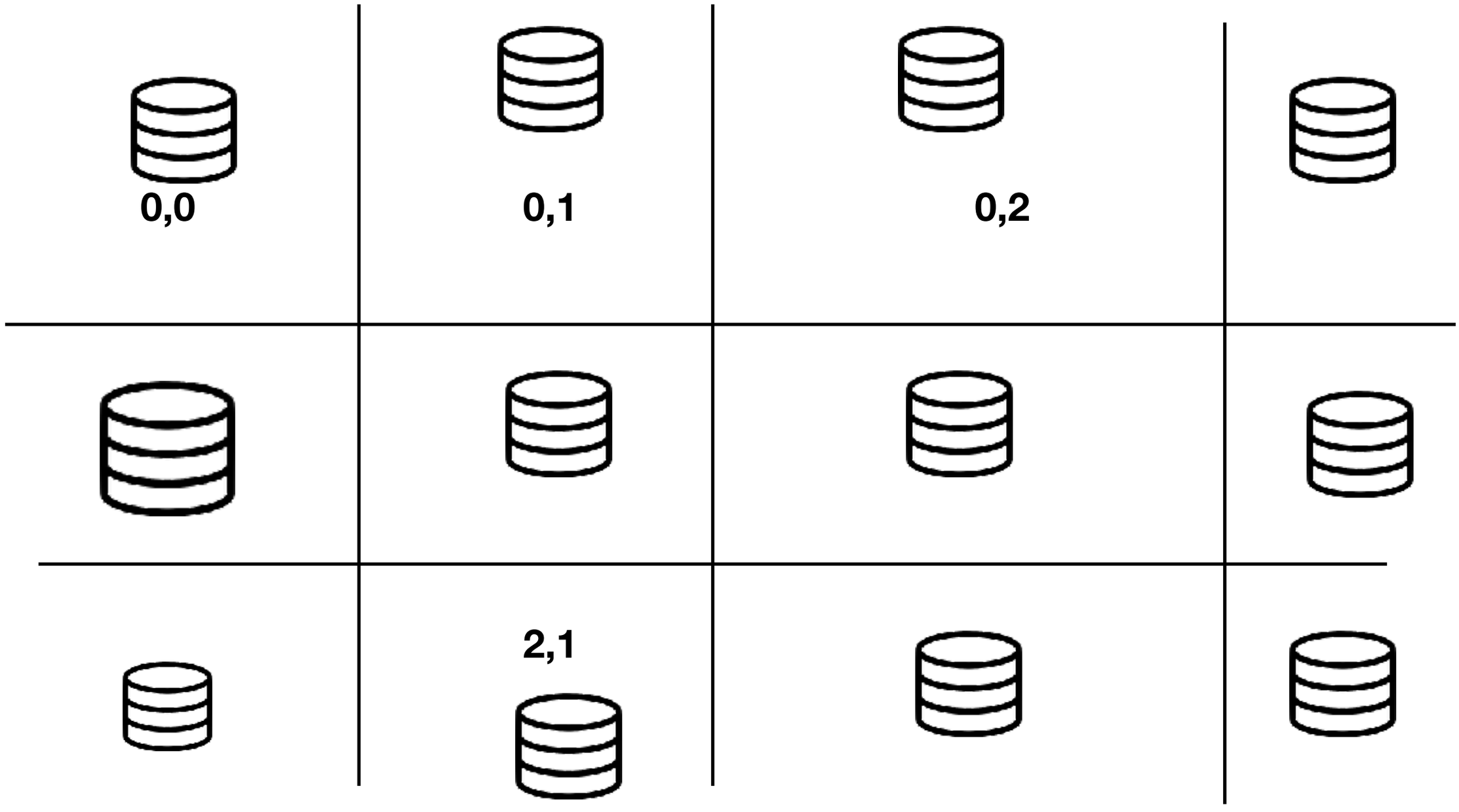
{Ordered, cust name, amount} , {itemcode, qty,price}

(Row partition id (shard key), column family id, row id)

multidimensional map (map of maps)

```
{
  "rowkey1": {"cf11": {"column111": {"version1111": value1111,
                                     "version1112": value1112},
                       "column112": {"version1121": value1121,
                                     "version1122": value1122,
                                     "version1123": value1123,
                                     "version1124": value1124},
                       "column113": {"version1131": value1131}
                      },
             "cf12": {"column121": {"version1211": value1211},
                       "column122": {"version1221": value1221},
                                     "version1222": value1222}
                      }
             },
   "rowkey2": {"cf11": {"column111": {"version2111": value2111,
                                      "version2112": value2112},
                        "column112": {"version2121": value2121,
                                      "version2122": value2122,
                                      "version2123": value 2123,
                                      "version2124": value2124}
                       },
              "cf12": {"column121": {"version2211": value2211},
                       "column122": {"version2221": value2221}
                       }
             }
}
```

Row partition id,
Col partition id



Column Family 1

Column Family 2

	cf1:col-A	cf1:col-B	cf2:col-Foo	cf2:col-XYZ	cf2:foobar
Region 1					
row-1					
row-10					
row-18	A18 - v1	B18 - v3	Foo18 - v1	XYZ18 - v2	foobar18 - v1
Region 2					
row-2					
row-5					
row-6					
row-7					

Physical Coordinates for a Cell: *Region Directory* → *Column Family Directory* → *Row Key* → *Column Family Name* → *Column Qualifier* → *Version*

	CF1:colA	CF1:colB	CF1:colC
Row1	@time7: value3		
Row10	@time2: value1	@time2: value1	
Row11	@time6: value2		
Row2	@time4: value1		@time4: value1

Row-oriented

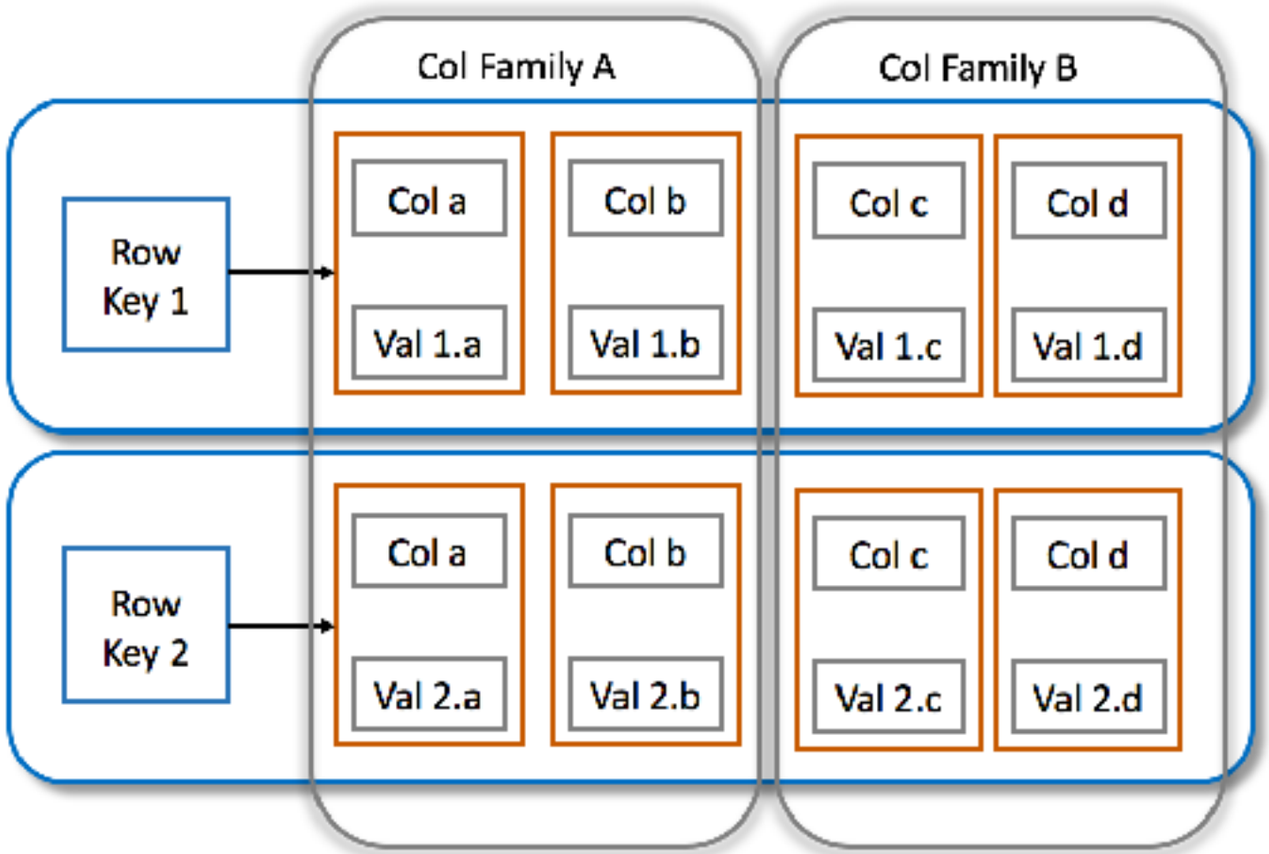
ID	Name	Grade	GPA
001	John	Senior	4.00
002	Karen	Freshman	3.67
003	Bill	Junior	3.33

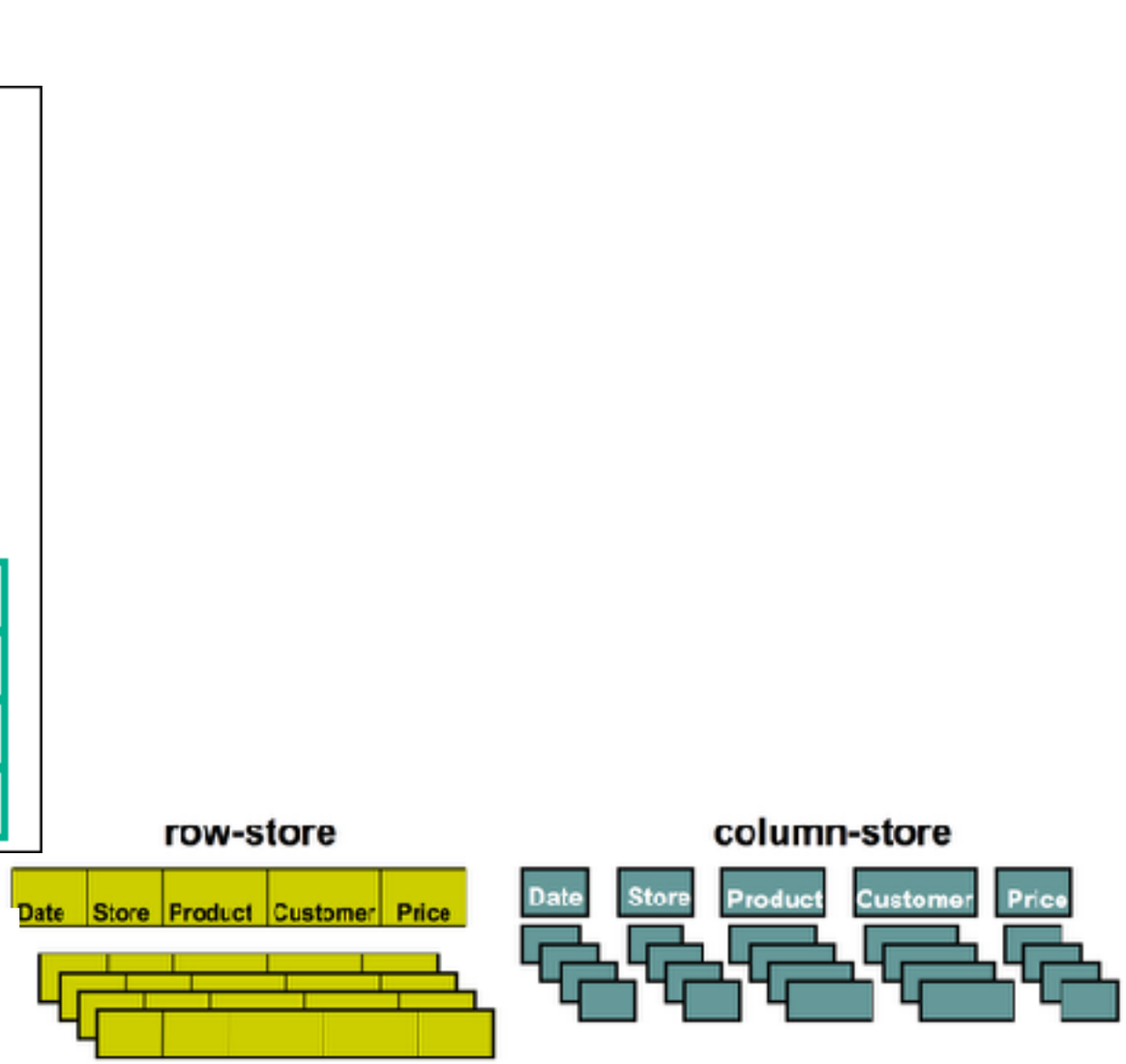
Column-oriented

Name	ID
John	001
Karen	002
Bill	003

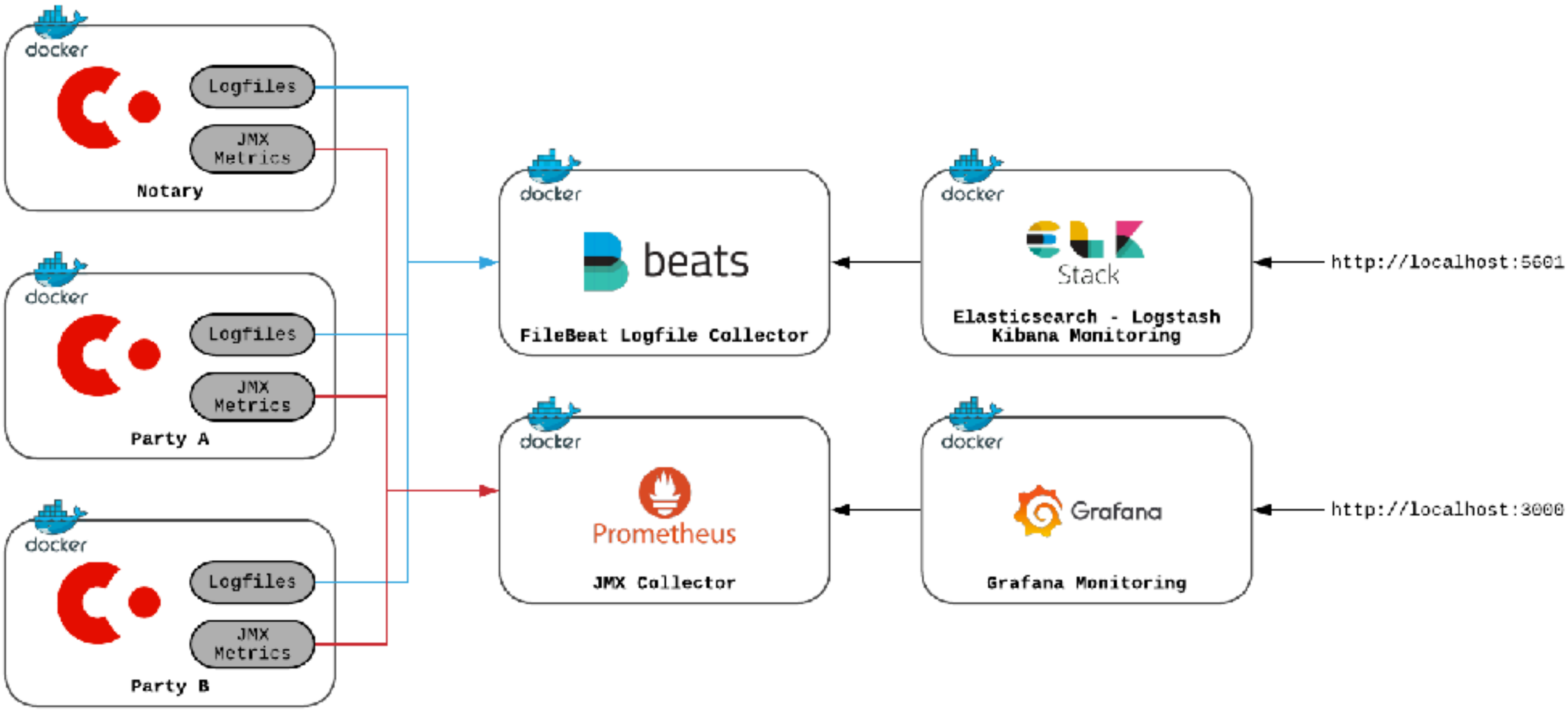
Grade	ID
Senior	001
Freshman	002
Junior	003

GPA	ID
4.00	001
3.67	002
3.33	003





column-family	Columnar
sparse data model	
multi-dimensional map	
column-families are not independently accessible.	every column is stored separately
NoSQL	SQL interface
Reads that use the partition key are incredibly fast	optimized for read-mostly analytical workloads
high throughput writes	very slow writes
	data warehouses


docker
Logfiles
JMX Metrics
Notary
docker
Logfiles
JMX Metrics
Party A
docker
Logfiles
JMX Metrics
Party B
docker
beats
FileBeat Logfile Collector
docker
Prometheus
JMX Collector
docker
Stack
Elasticsearch - Logstash
Kibana Monitoring
http://localhost:5601
docker
Grafana
Grafana Monitoring
http://localhost:3000

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	☑

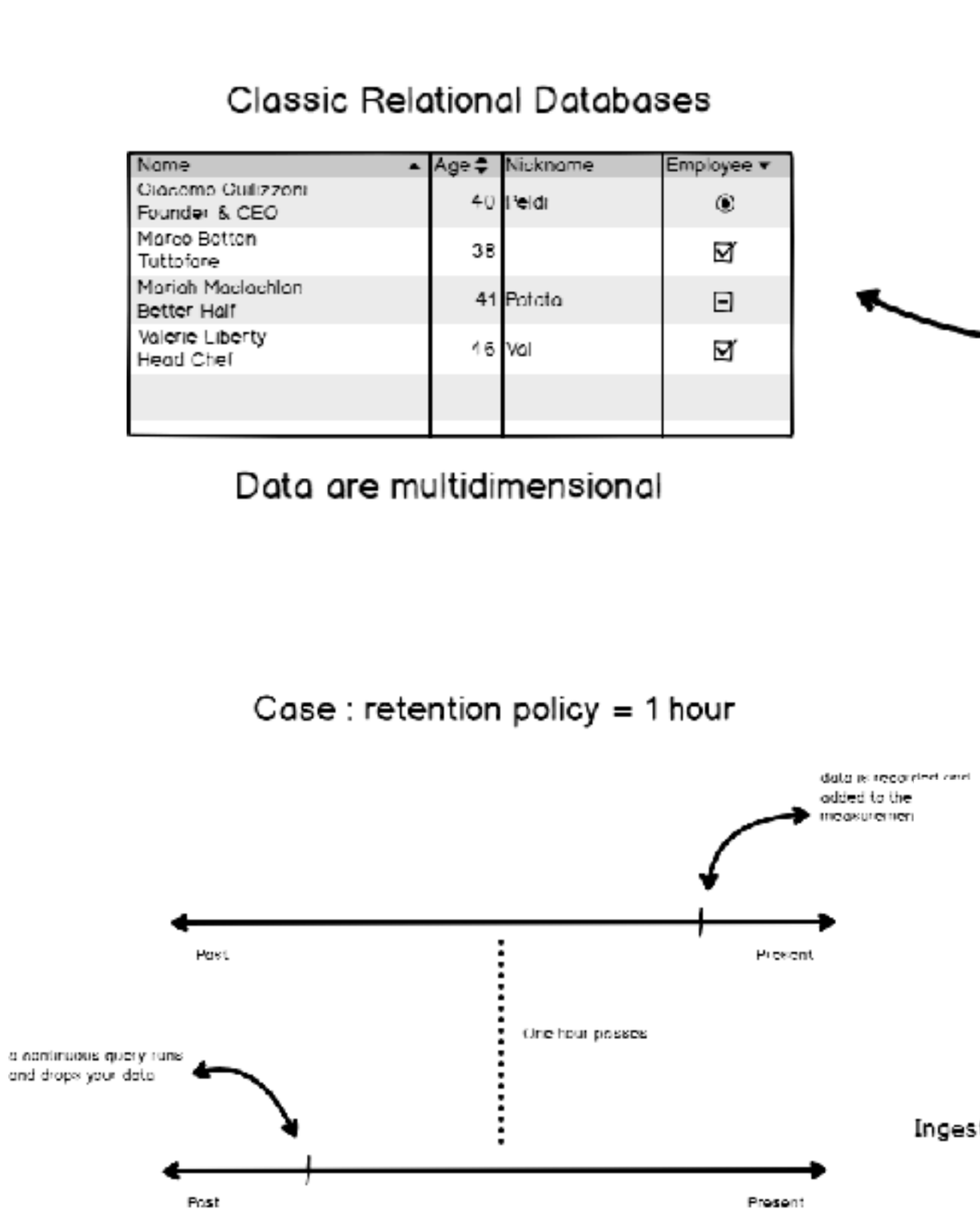
Data are multidimensional

Time Series Databases

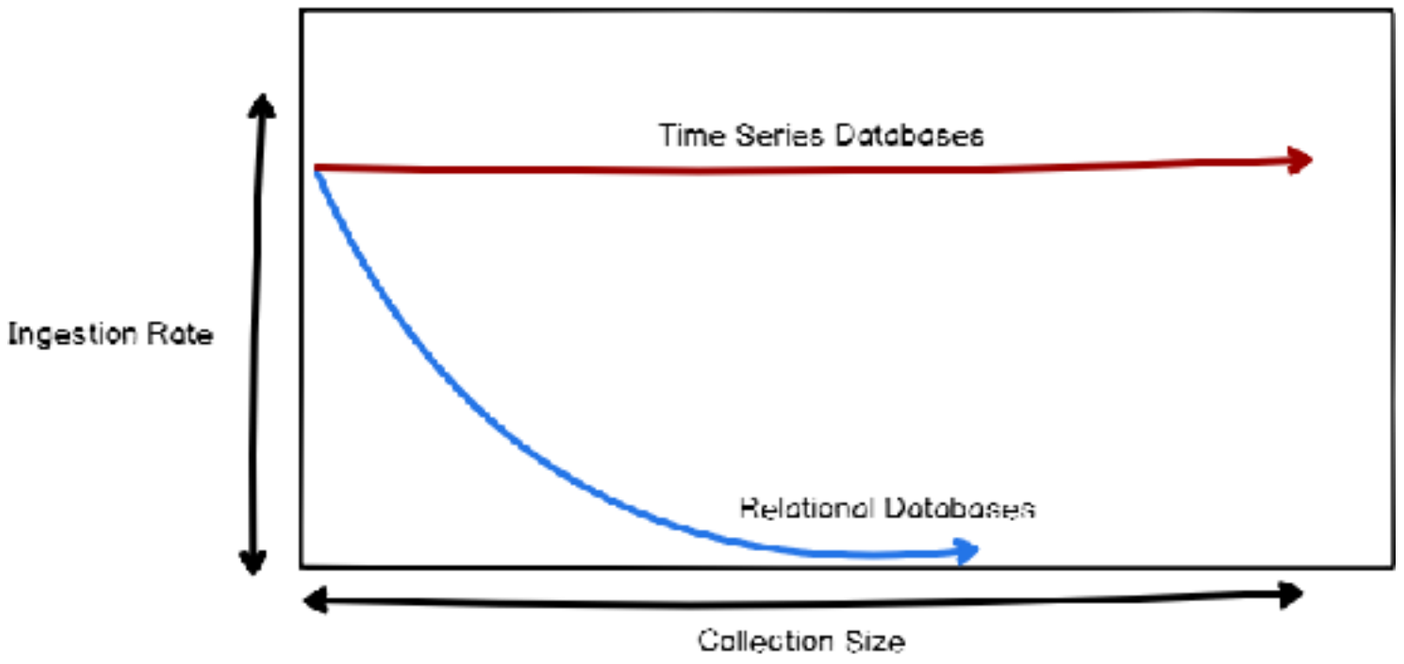
Sensor Temperature	Time
39.6	12/04/19 @ 14:12
41.2	12/04/19 @ 14:13
12.4	14/04/19 @ 12:15
18.5	16/04/19 @ 10:05

Data are aggregated over time

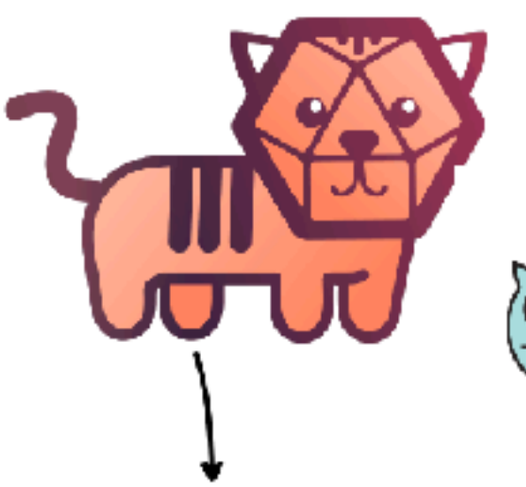
Case : retention policy = 1 hour



DBMS & TSDB Difference

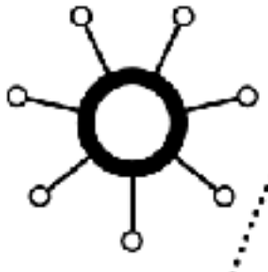


Tools that 'produce' TSDB data



monitoring done via specific query agents

collectd



StatsD

monitoring done via node exporters

Prometheus

Tools that 'consume' TSDB

chronograf

DATADOG

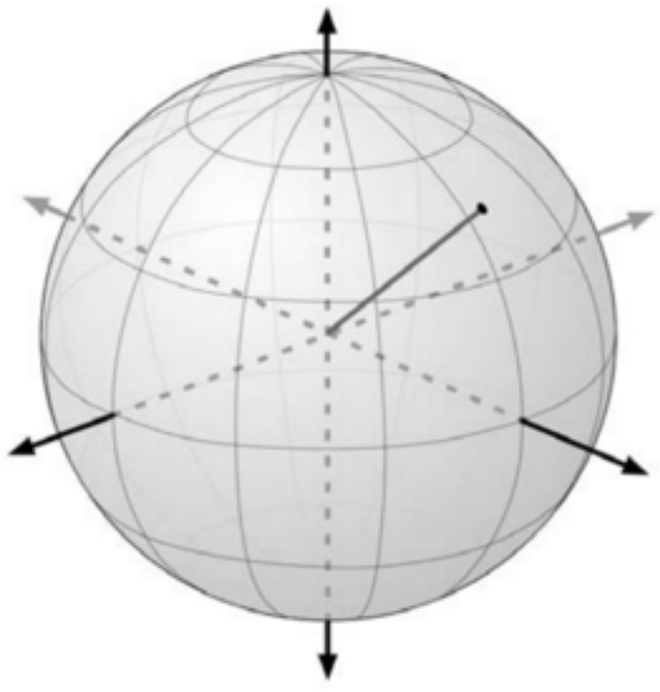
Prometheus

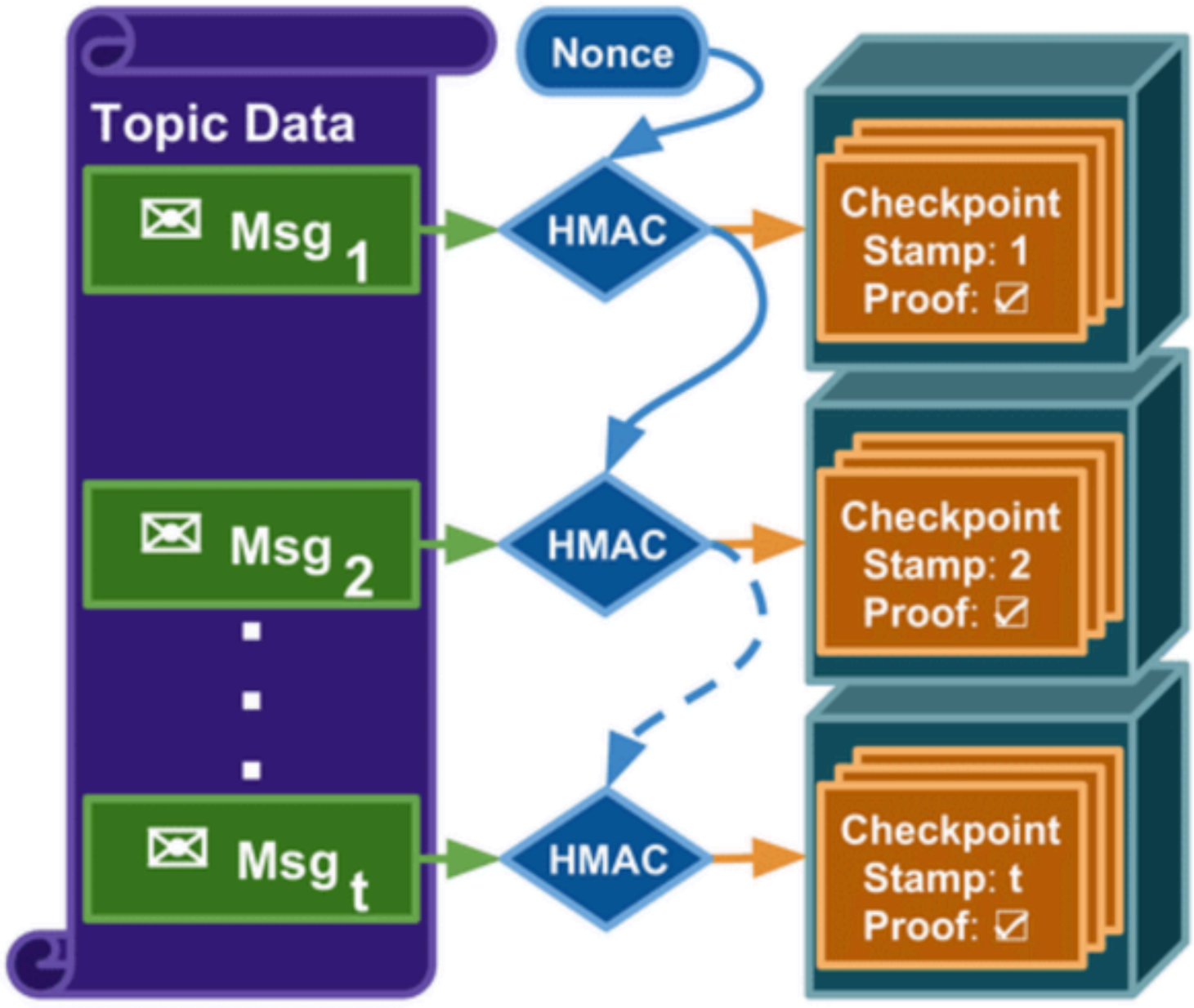
Grafana

kibana

*Non-exhaustive list

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



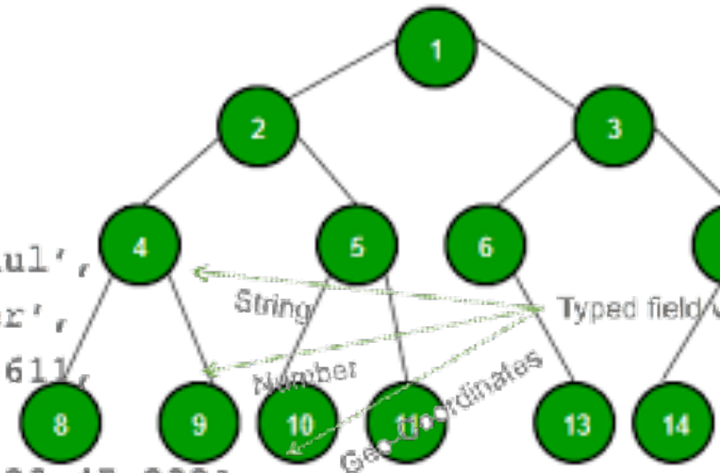

Topic Data
Msg 1
Msg 2
Msg t
Nonce
HMAC
HMAC
HMAC
Checkpoint
Stamp: 1
Proof: ☑
Checkpoint
Stamp: 2
Proof: ☑
Checkpoint
Stamp: t
Proof: ☑

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

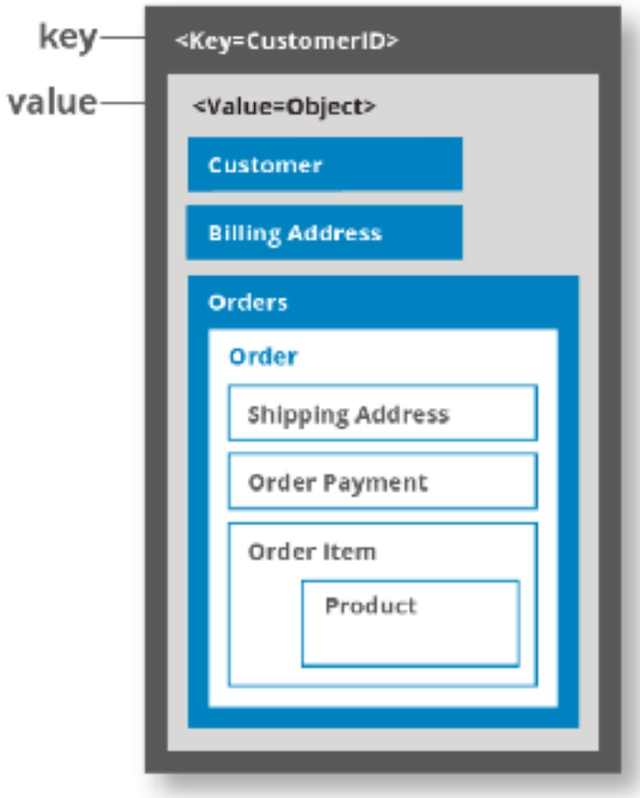
Fields

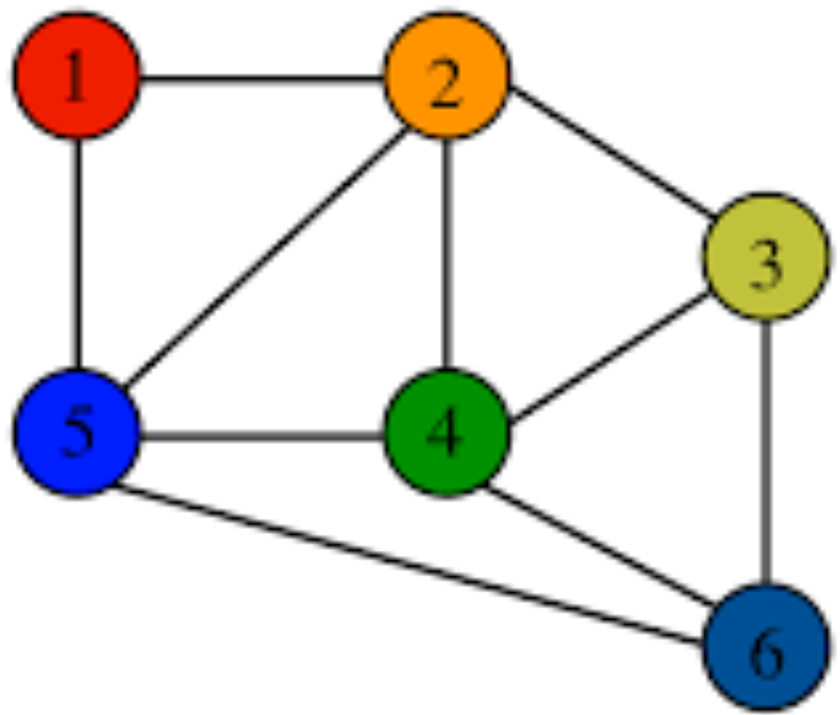



```
{
  first_name: 'Paul',
  surname: 'Miller',
  cell: 447557505611,
  city: 'London',
  location: [45.123,47.232],
  Profession: ['banking', 'finance', 'trader'],
  cars: [
    { model: 'Bentley',
      year: 1973,
      value: 100000, … },
    { model: 'Rolls Royce',
      year: 1965,
      value: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

Fields can contain an array of documents

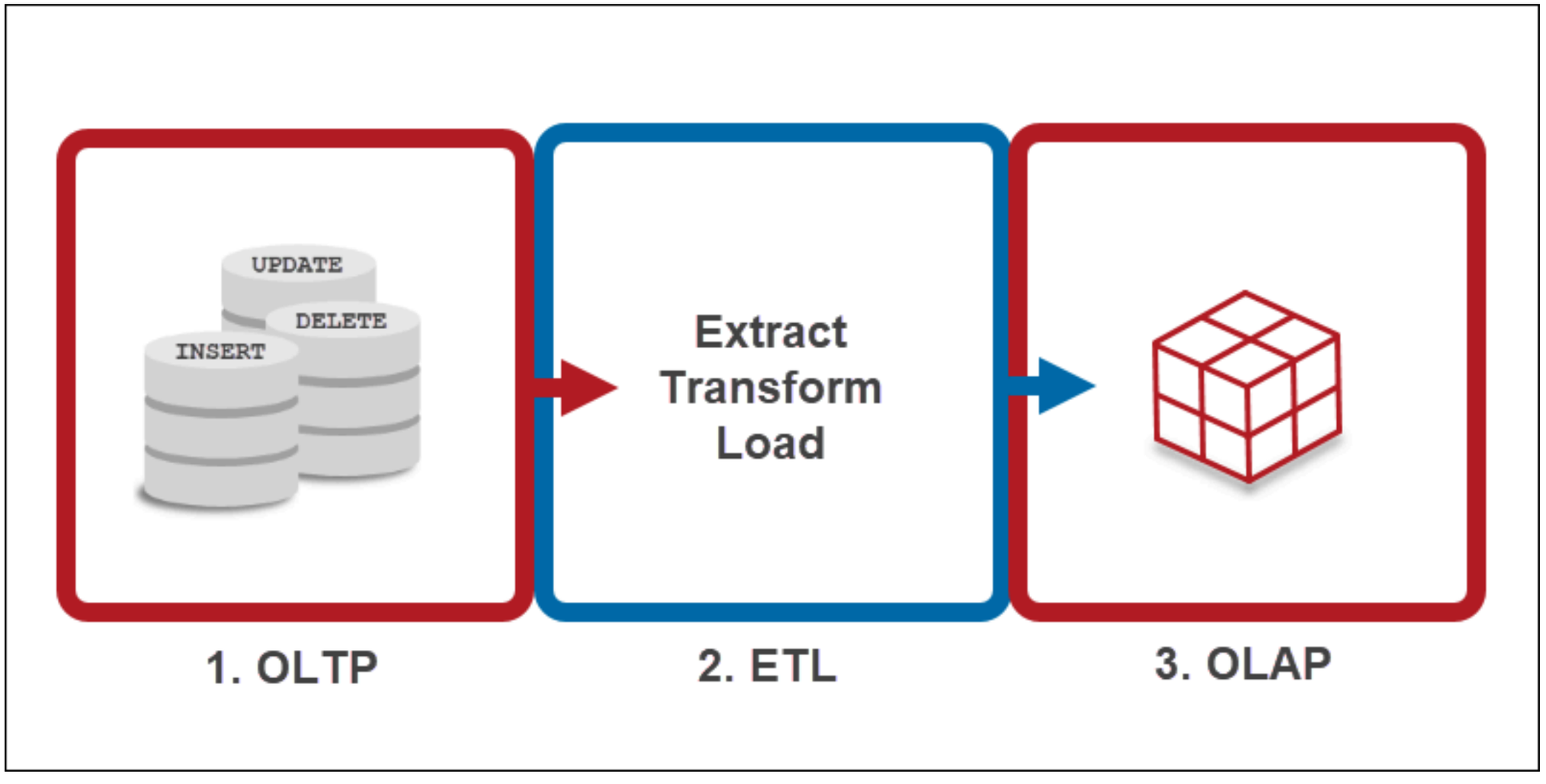
key	value
123	123 Main St.
126	(805) 477-3900





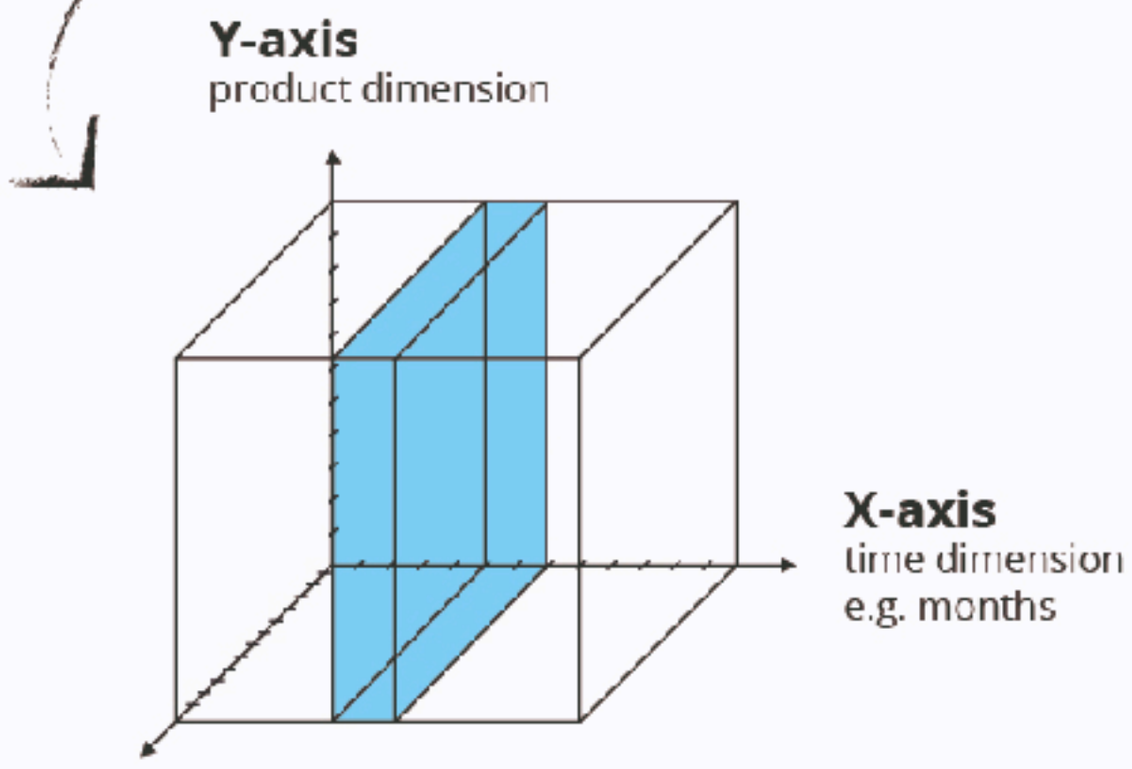
Wide Column

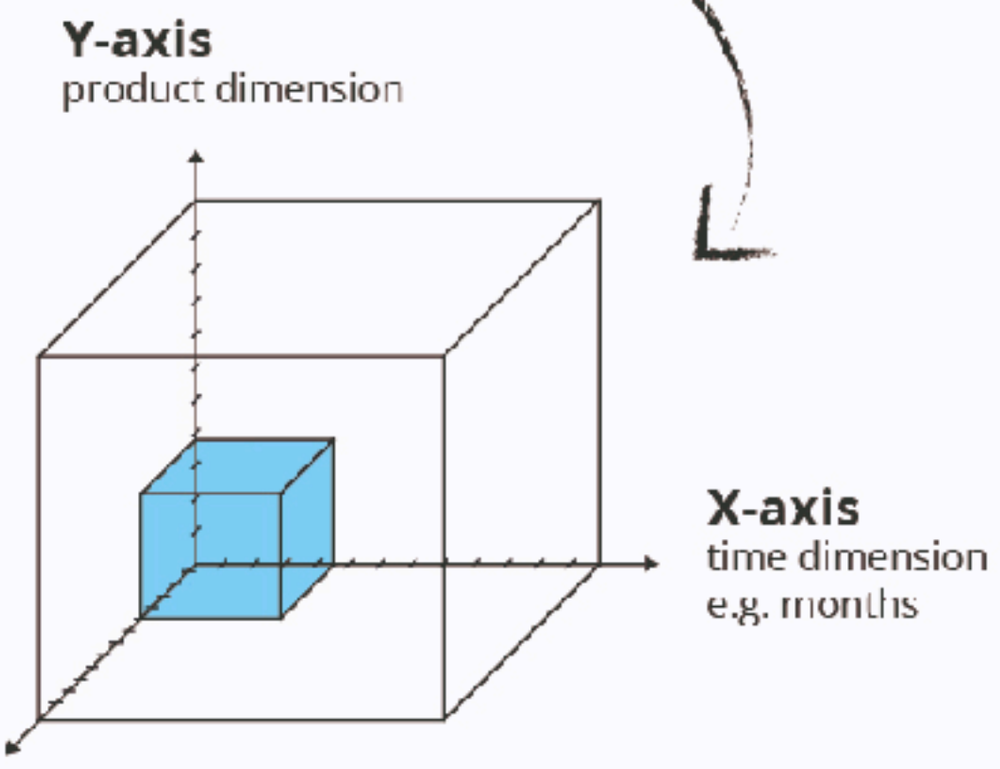
- Spotify uses Cassandra to store user profile attributes and metadata about artists, songs, etc. for better personalization
- Facebook initially built its revamped Messages on top of HBase, but is now also used for other Facebook services like the Nearby Friends feature and search indexing
- Outbrain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

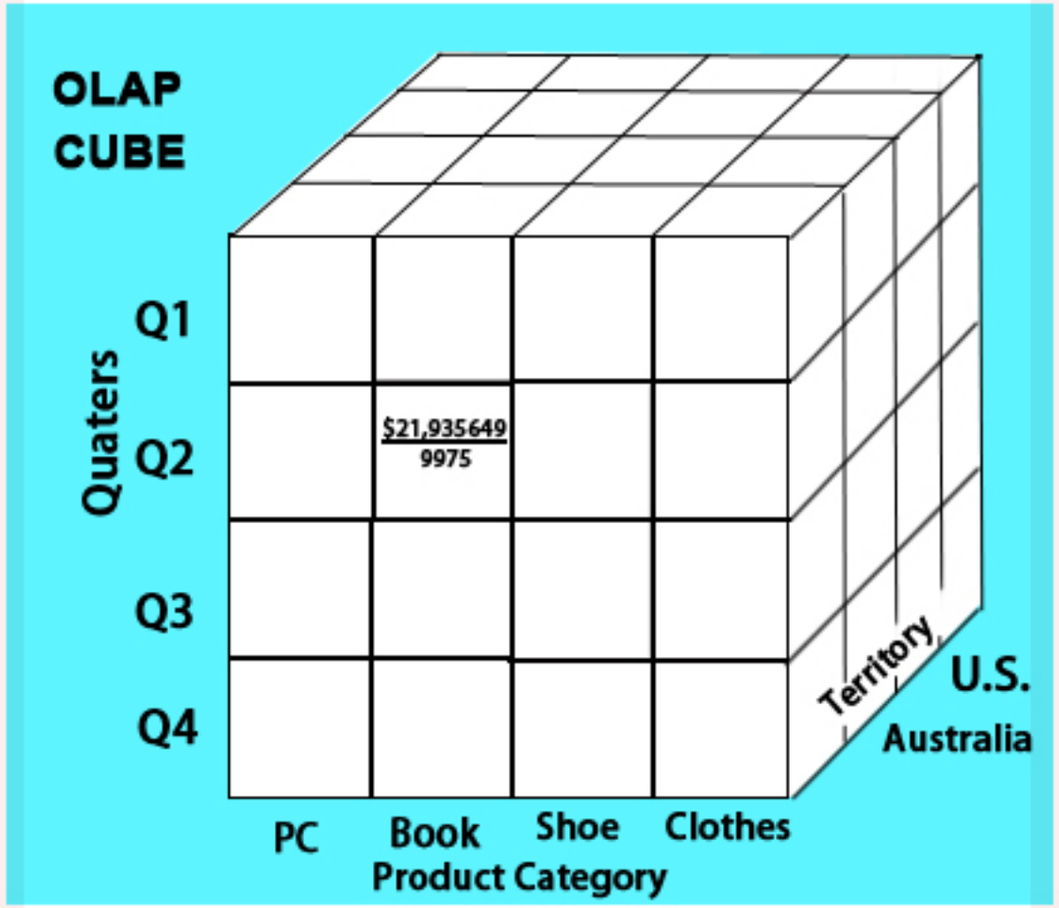

UPDATE
DELETE
INSERT
Extract
Transform
Load
1. OLTP
2. ETL
3. OLAP

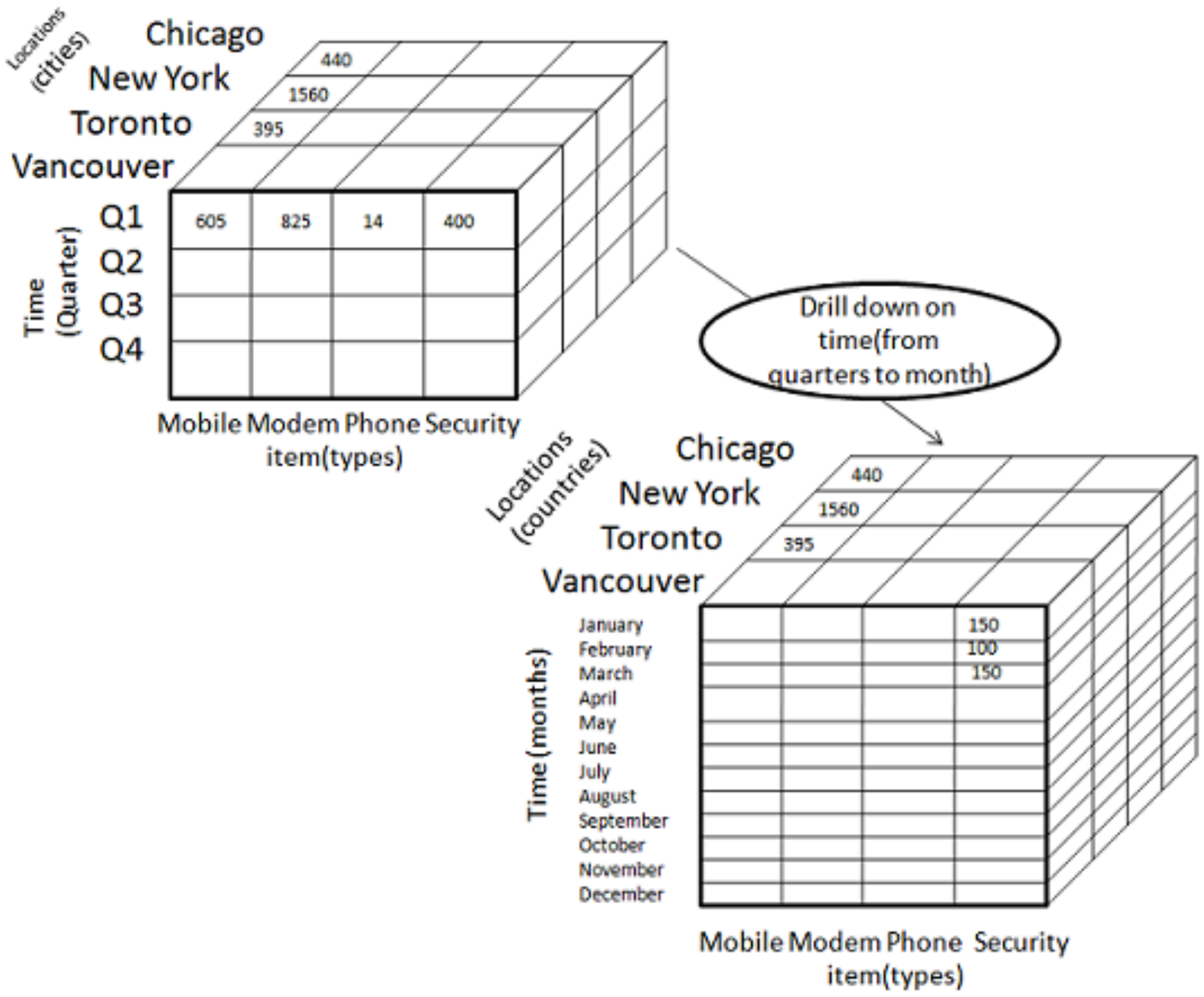
Slicing & Dicing

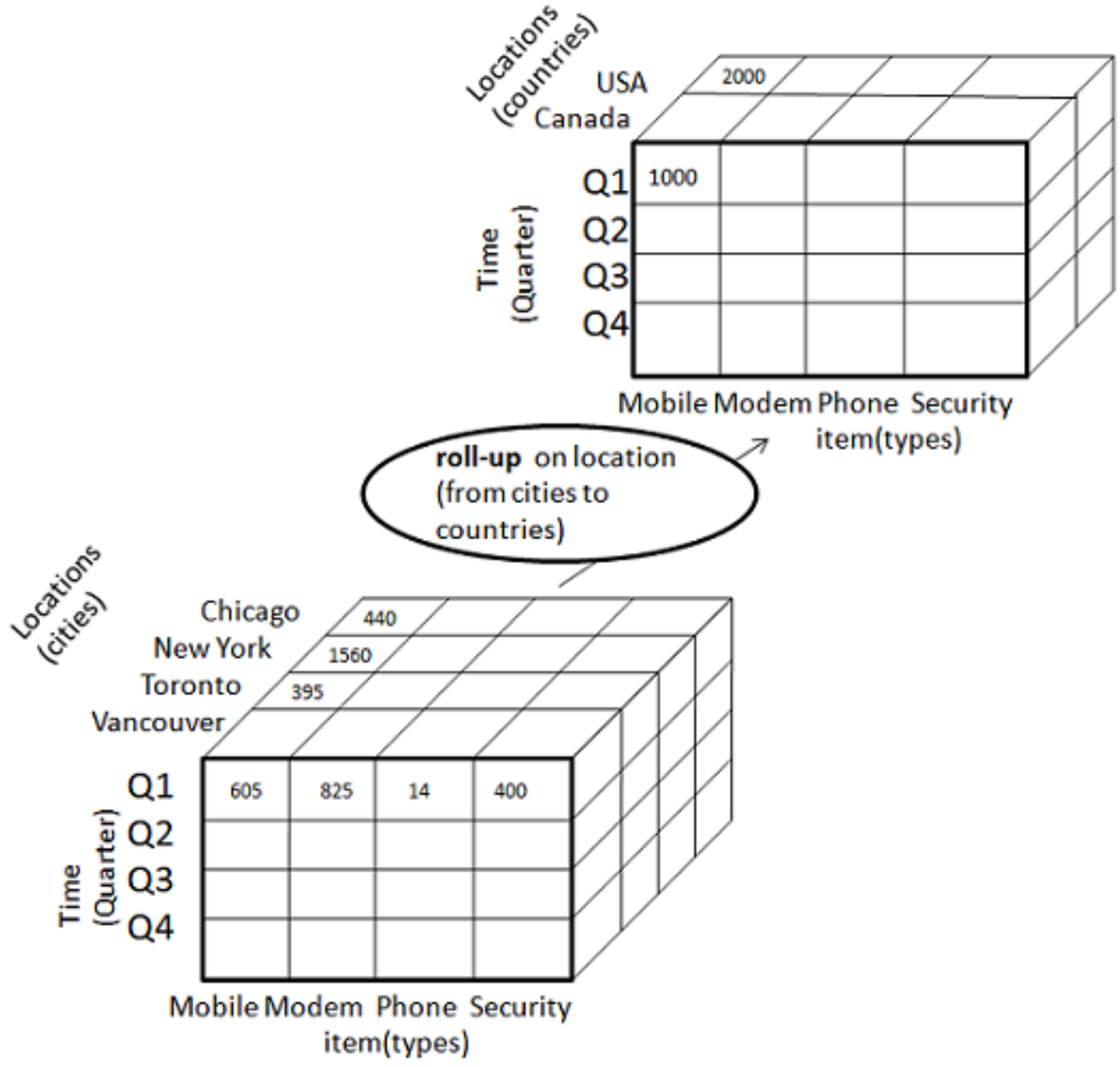
OLAP cube operations

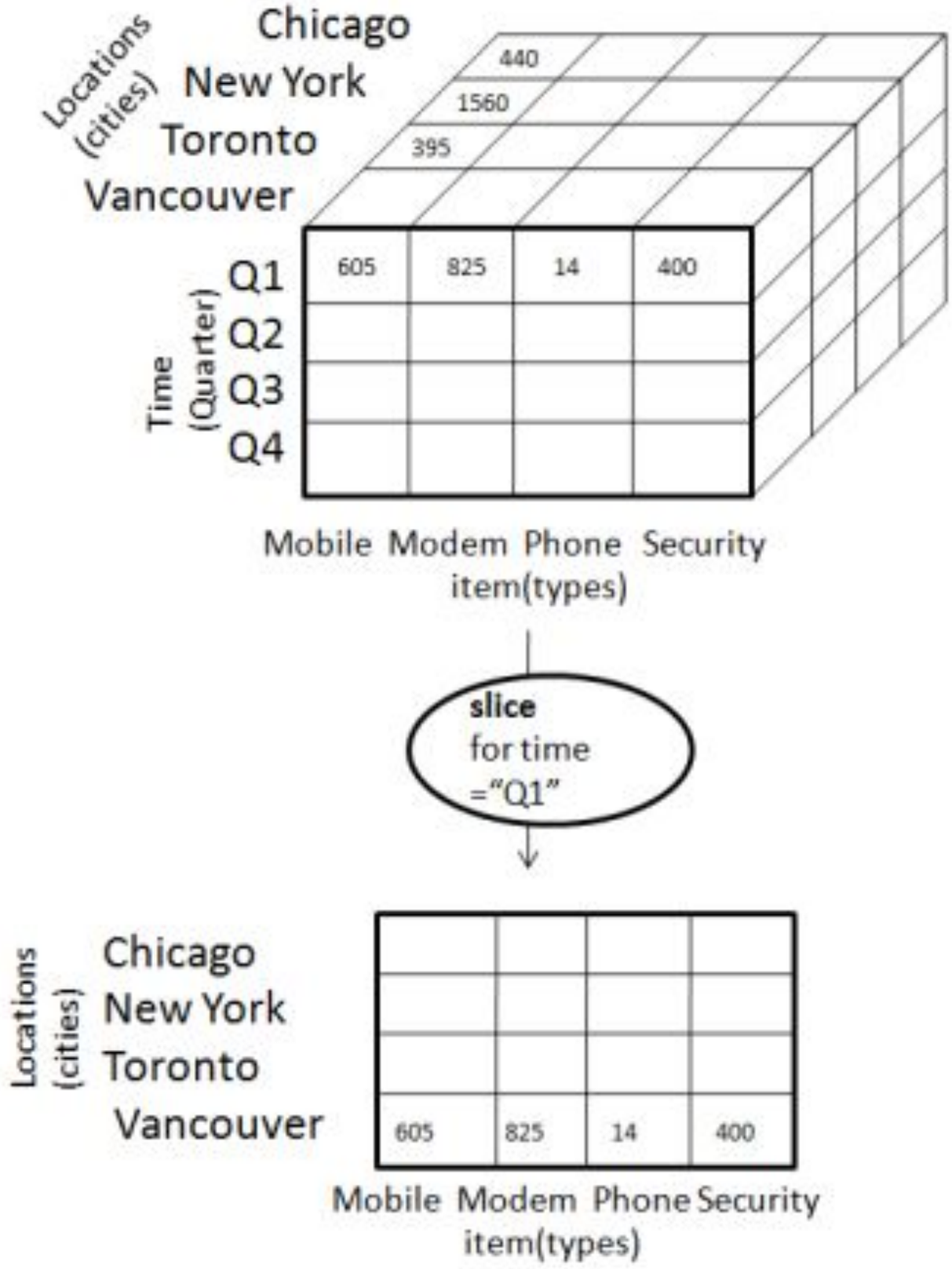


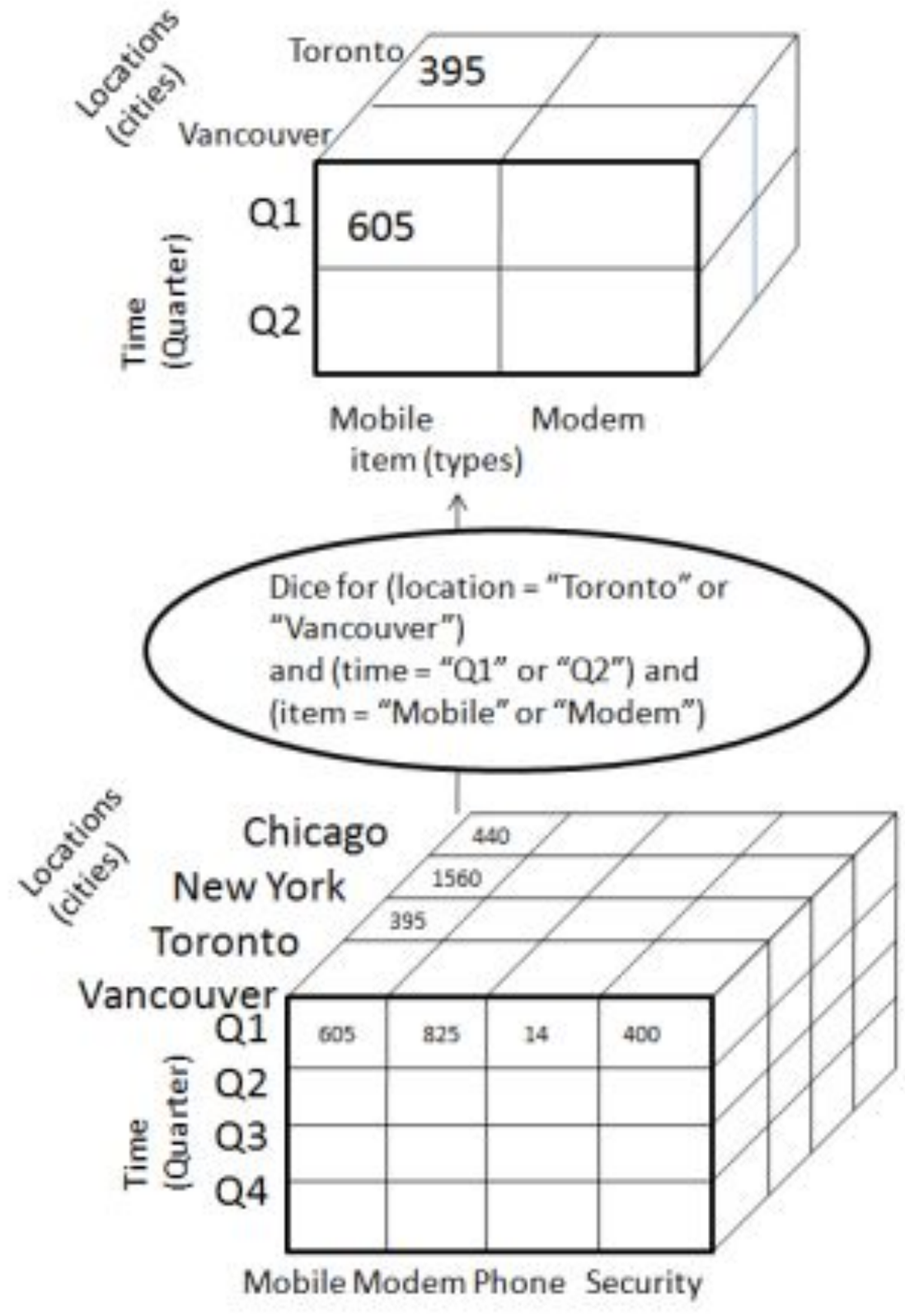


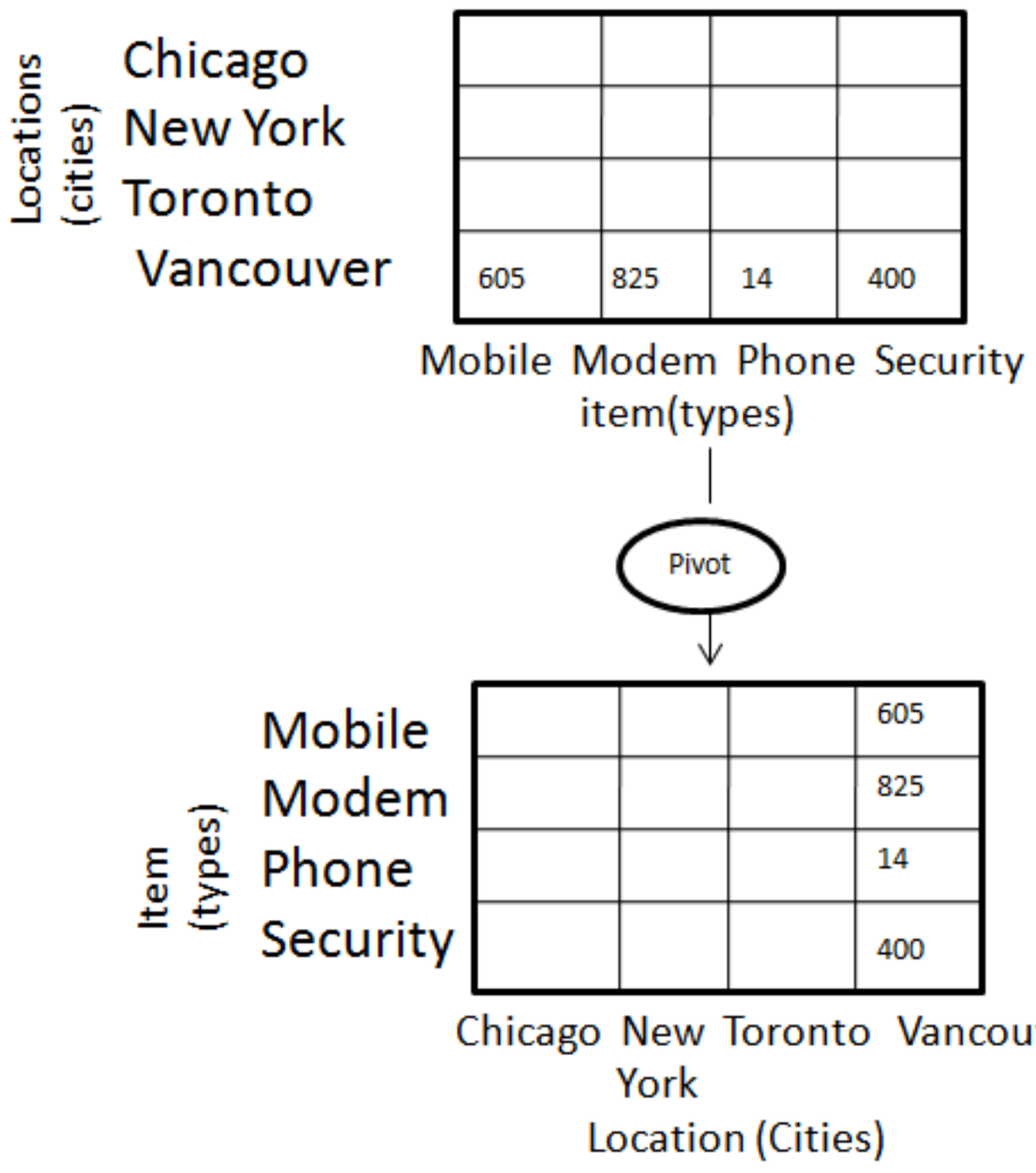

OLAP
CUBE
Quaters
Q1
Q2
Q3
Q4
$21,935649
9975
PC
Book
Shoe
Clothes
Product Category
Territory
U.S.
Australia


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)

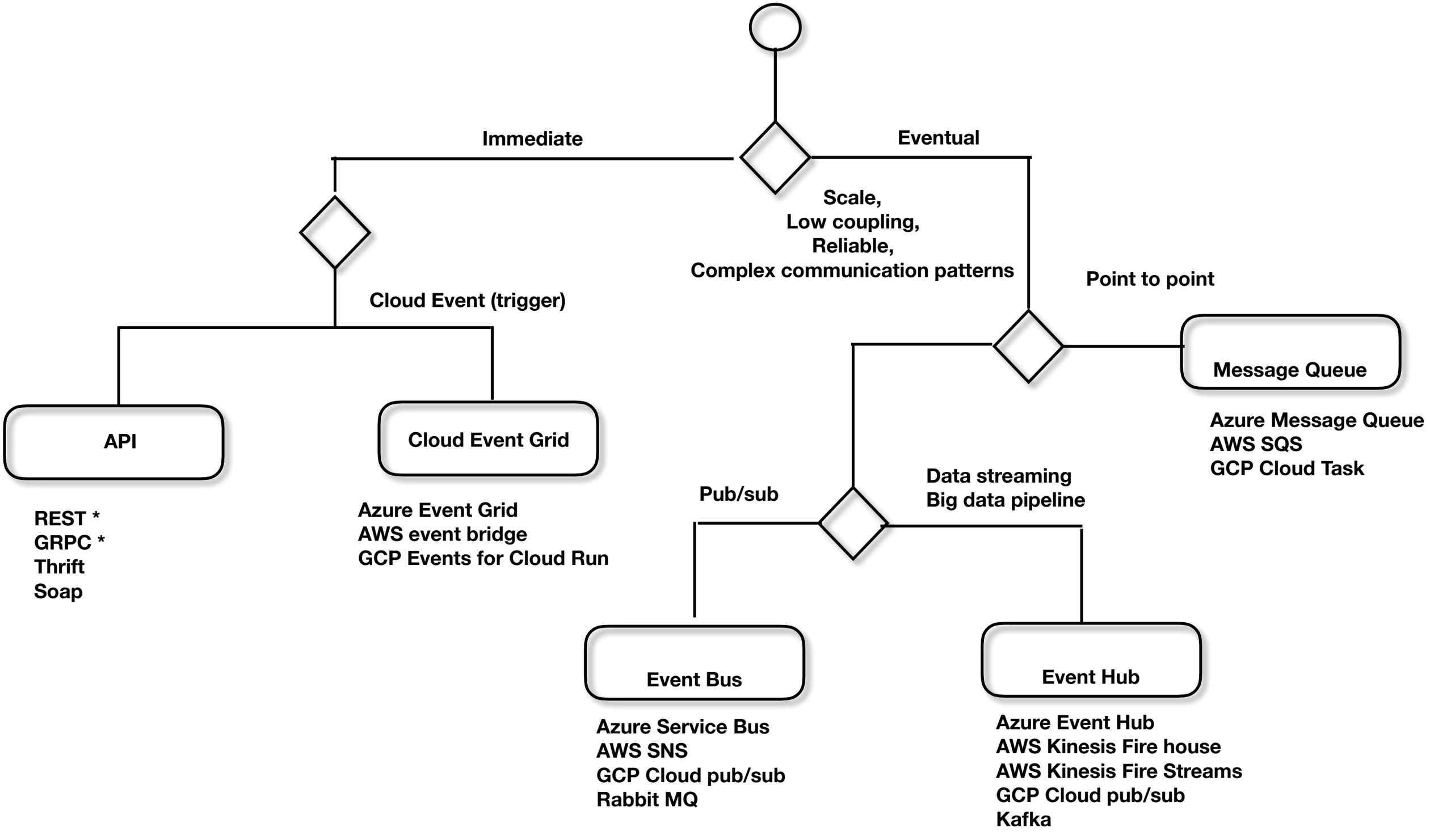

Locations (countries)
USA
Canada
2000
1000
Q1
Q2
Q3
Q4
Time (Quarter)
Mobile Modem Phone Security
item(types)
roll-up on location (from cities to countries)
Locations (cities)
Chicago
New York
Toronto
Vancouver
440
1560
395
605
825
14
400
Q1
Q2
Q3
Q4
Time (Quarter)
Mobile Modem Phone Security
item(types)


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)

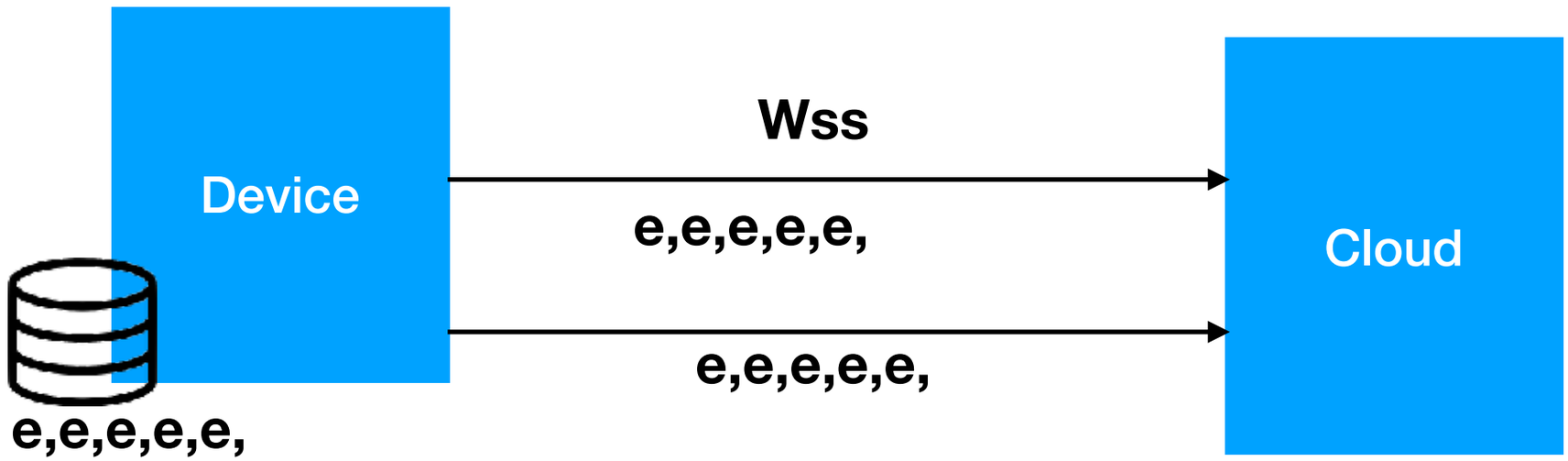
Choose Communication protocol

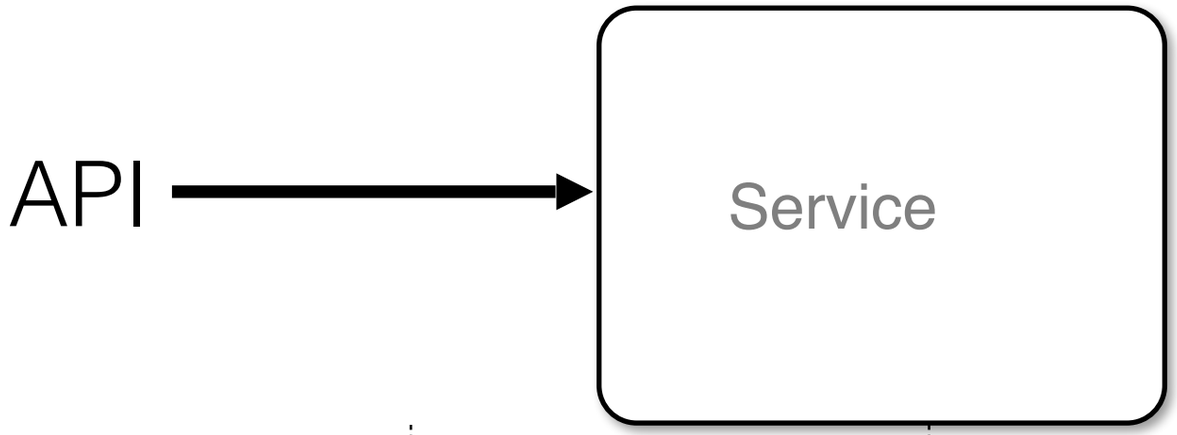


Connected protocol
(REST, WS, GRPC , Thrift)

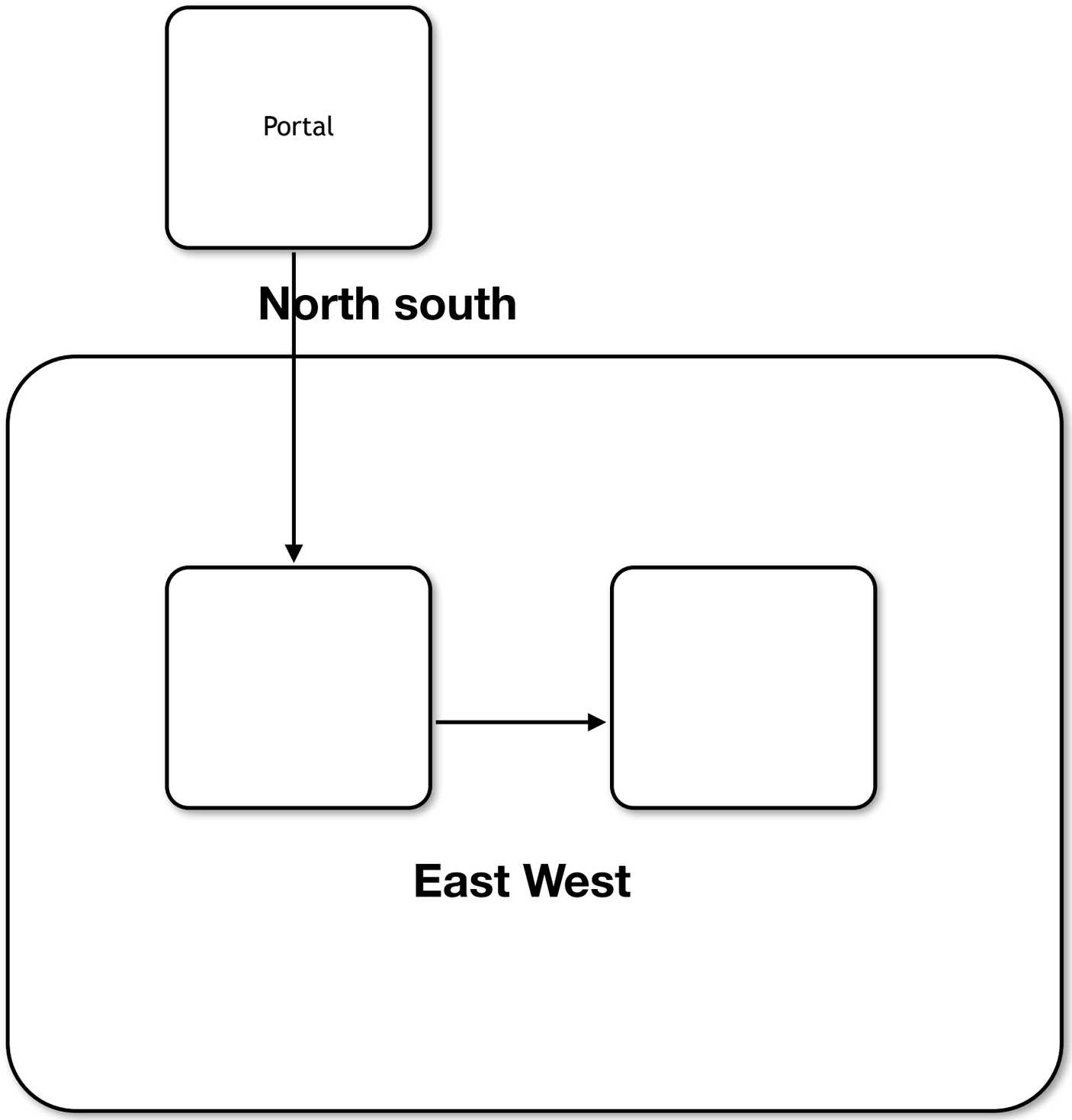
Message protocol
(AMQP, MQTT, …)

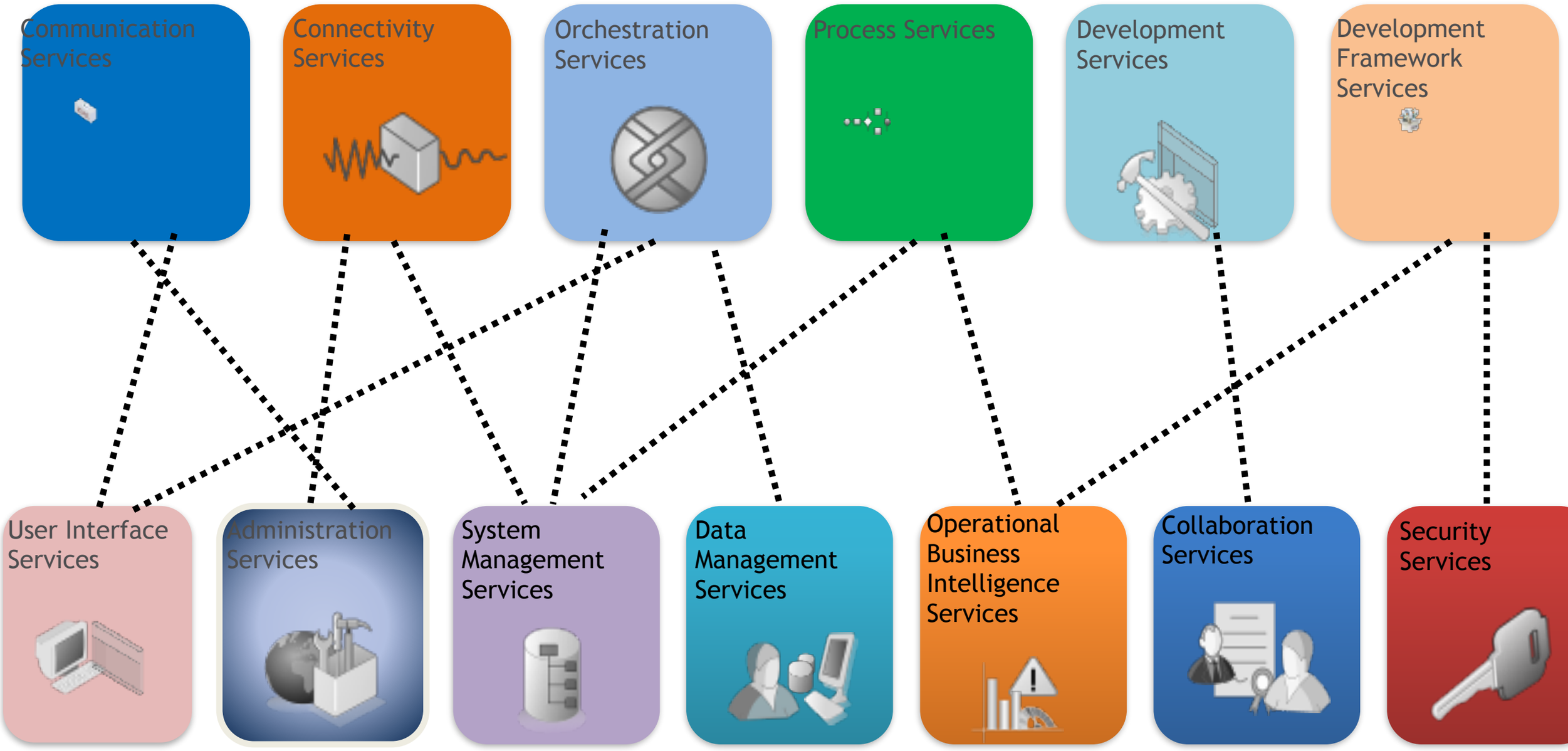
	Connected	Msg database (ACID)
Scale	- -	+ +
Coupling	- -	+ +
Reliability	- -	+ +
Developer effort		
Consistency	Immediate	Eventual
Debugging		
Delivery Order	Ordered ++	Unordered * - -
Duplicate	- -	- -
Dev Ops Effort		
Exception Handling	+ +	- - ?


Device
Wss
e,e,e,e,e,
e,e,e,e,e,
Cloud
e,e,e,e,e,



	REST	WSS	GRPC
Browser Support	**Y**	**Y**	**N**
Format	**TEXT**	**TEXT**	**BINARY, HTTP2**
Serialization	**JSON**	**JSON**	**Proto Buf**
Performance	**- -**	**+**	**+ +**
Use case	**North South**	**Streaming**	**East West**


Portal
North south
East West


Communication Services
Connectivity Services
Orchestration Services
Process Services
Development Services
Development Framework Services
User Interface Services
Administration Services
System Management Services
Data Management Services
Operational Business Intelligence Services
Collaboration Services
Security Services

Communication Services

Connectivity Services

Orchestration Services

Process Services

Development Services

Development Framework Services

User Interface Services

Administration Services

System Management Services

Data Management Services

Operational Business Intelligence Services

Collaboration Services

Security Services

Communication Services

Connectivity Services

Orchestration Services

Process Services

Development Services

Development Framework Services

User Interface Services

Administration Services

System Management Services

Data Management Services

Operational Business Intelligence Services

Collaboration Services

Security Services

Event vs Command

Post **Pre**

OrderCreated **InvoiceCreated** **OrderCanceled**

Communication Services

Connectivity Services

Orchestration Services

Process Services

Development Services

Development Framework Services

CreateOrder

OderCreated **OderCanceled** **OderCreated**

User Interface Services

Administration Services

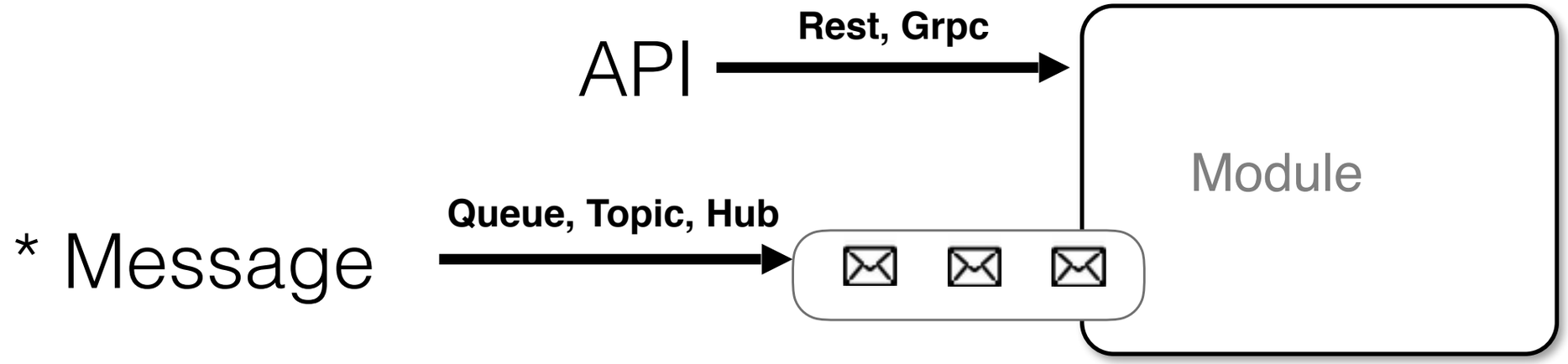
System Management Services

Data Management Services

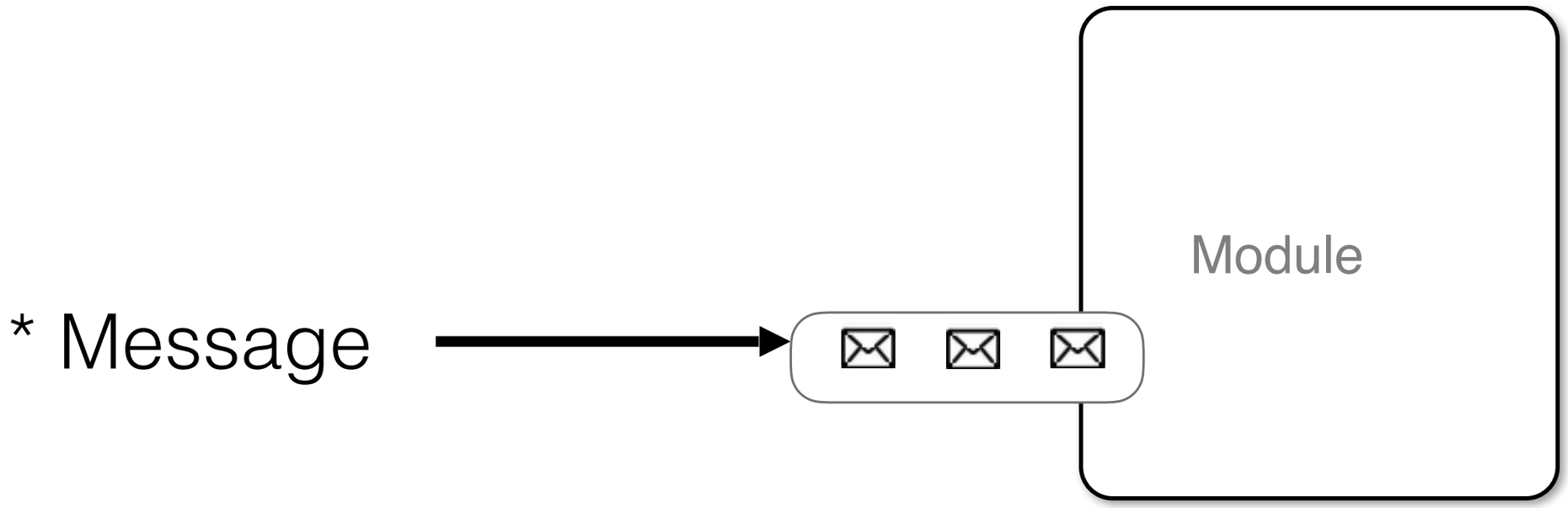
Operational Business Intelligence Services

Collaboration Services

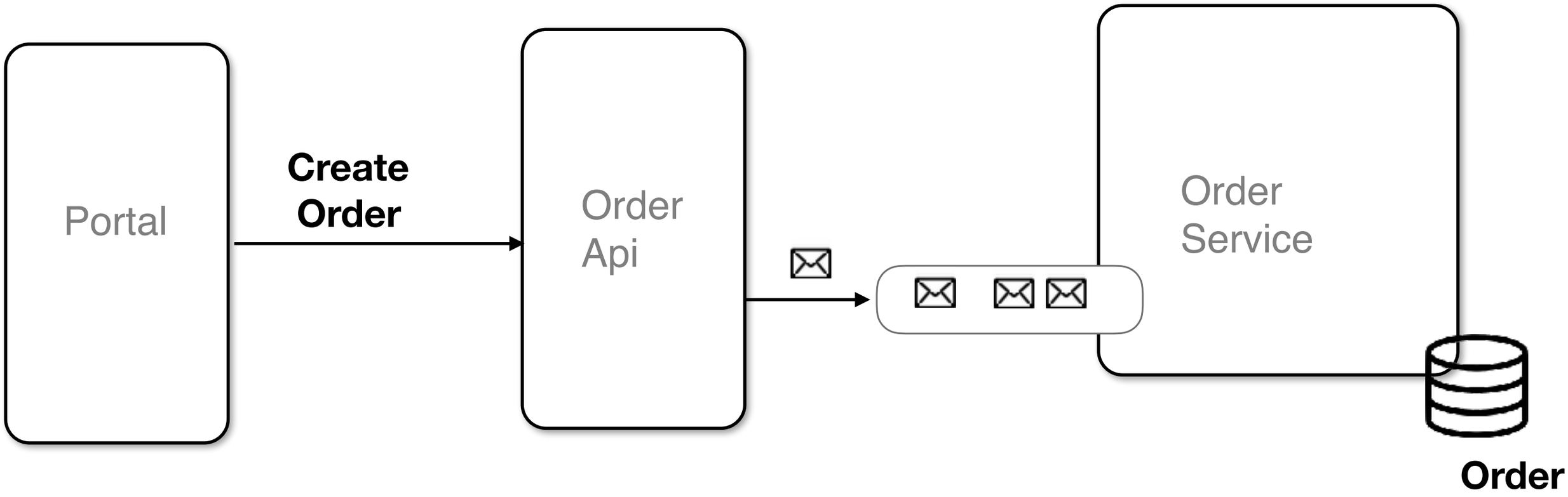
Security Services



	<< API >>	<< Message >>	
Ordering	Ordered (+)	Unordered(-)	
Duplicate	Yes (-)	Yes(-)	Idempotency
Protocol	2 way (+)	One way (-)	
Resilence (recover)	No (-) retry logic	Yes (+)	
Connection	Always connected	Occousionaly connected	
Scalability	- -	+ + +	
Consistency	Immediate (++)	Eventual (- -)	
Load Leveling	- -	+ +	
Low Coupling (maintainability)	- -	+ +	
Distributed Comm Patterns	- -	+ +	SAGA, Materialized View
Internet	Yes	Yes *	
Browser support	Yes	No	
Dev effort	++	- -	
Operational effort	+ +	- -	




```
fun(){
    while(true){
        Msg m = GetMessage();
        .....
        m.ack();
    }
}
```

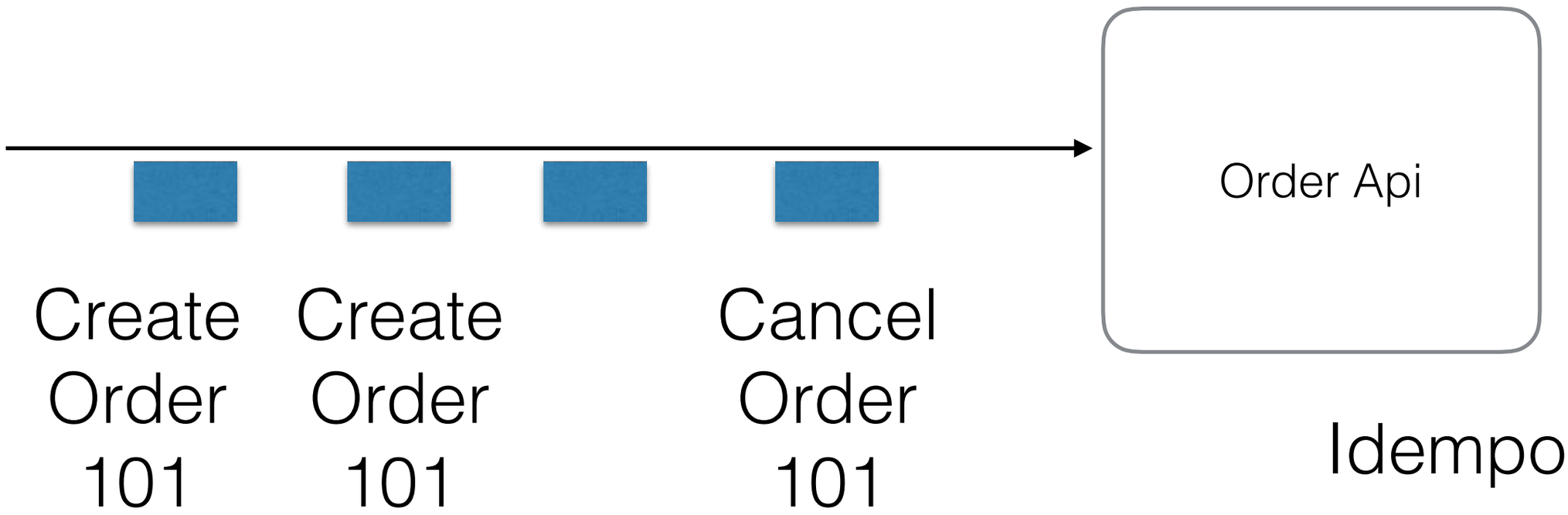

Portal
Create Order
Order Api
Order Service
Order
Order Service
Account
Account

Client

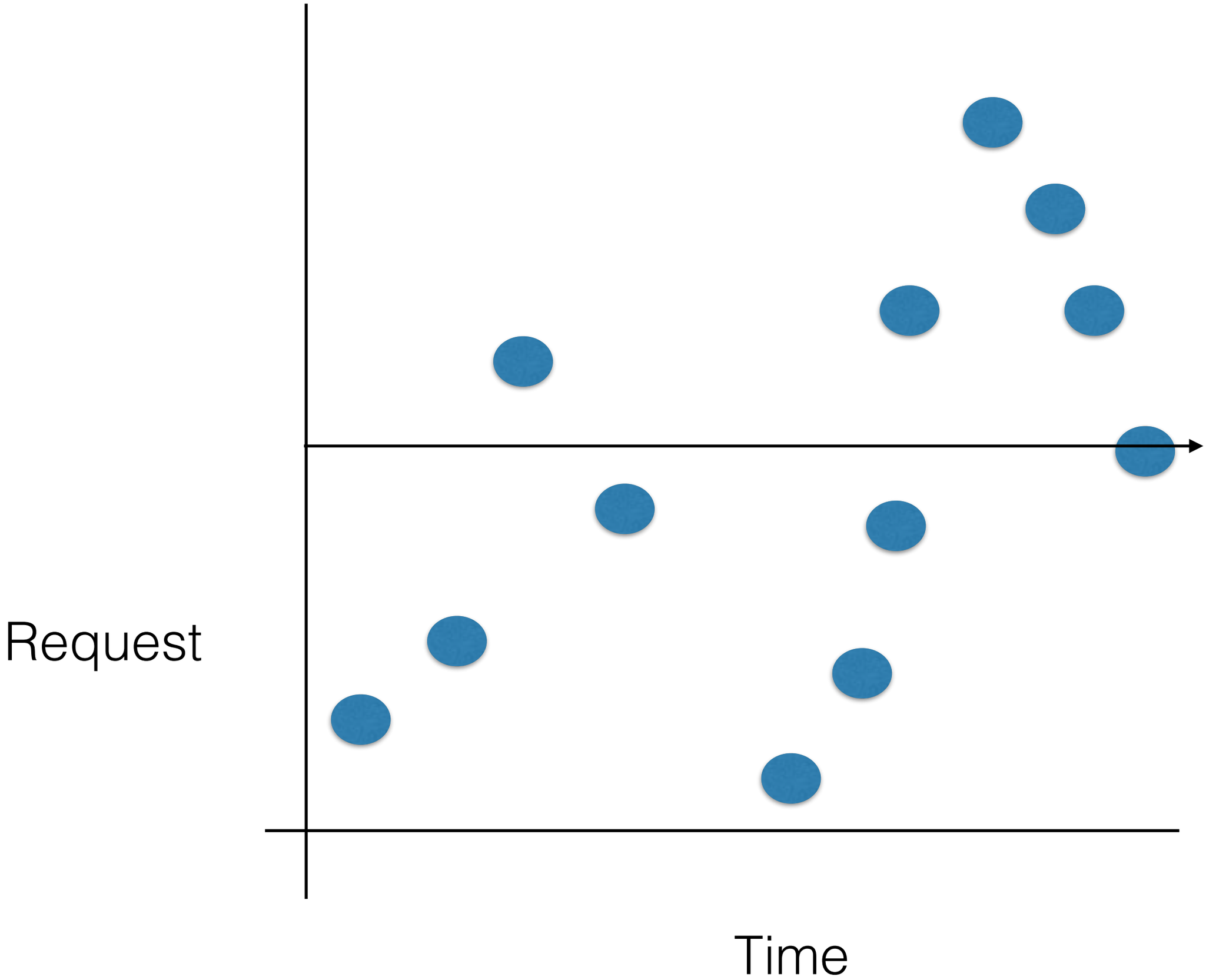
Server

Appp

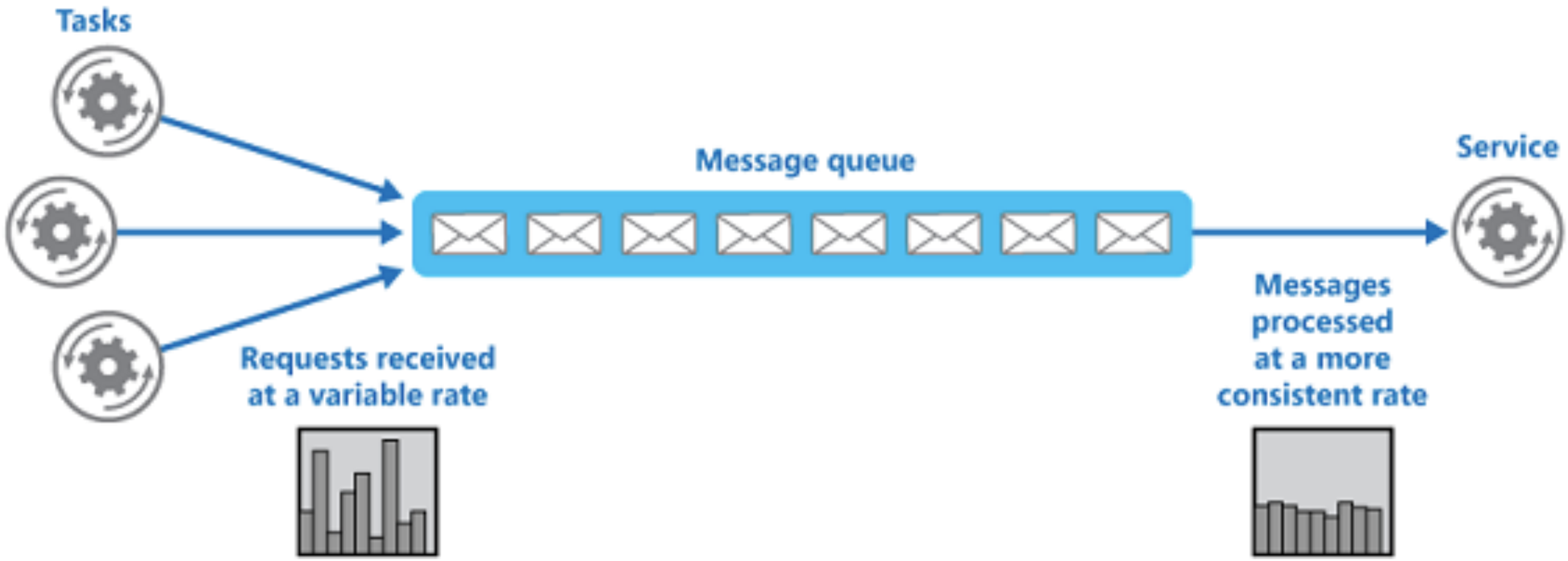
Service



Idempotency

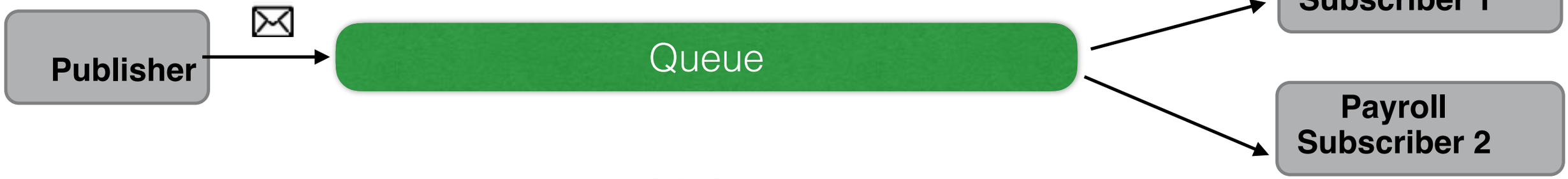

Request
Time

Queue-based Load Levelling

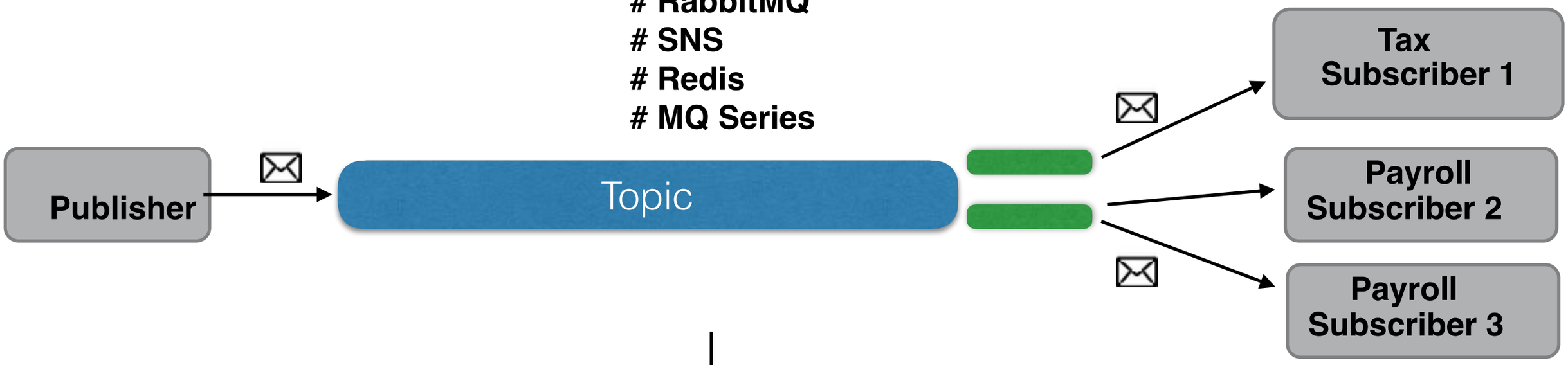


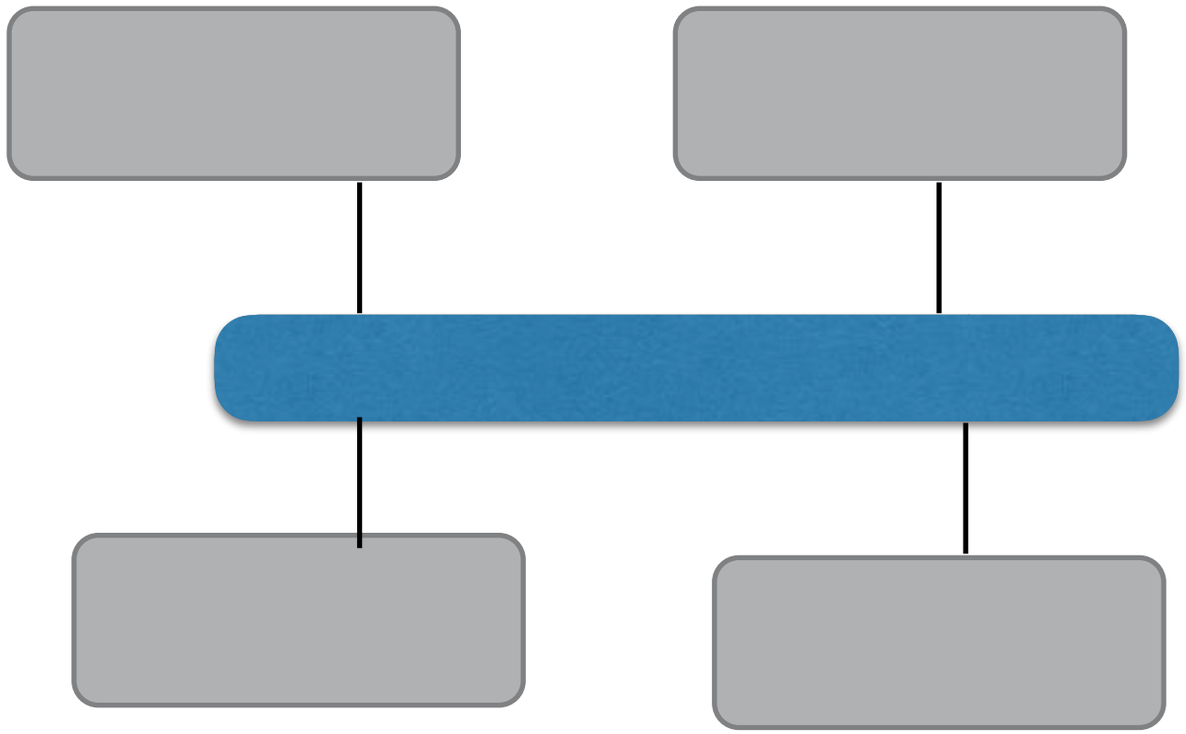
source:msdn

p2p
 # SQS
 # MSMQ
 # …



pubSub
 # RabbitMQ
 # SNS
 # Redis
 # MQ Series





<<Topic>>

Image

Task

Parallelism

Logic 1(data) , Logic 2(data) , Logic 3(data)

OCR(data) , Feature(data) ,Masking(data)

<<queue>>

Event1
Event 2
Event 3

Data

Parallelism

Logic(data1) , Logic(data2) , Logic(data3)

Anonyimization (Event1) ,
Anonyimization(Event2) ,
Anonyimization(Event3)

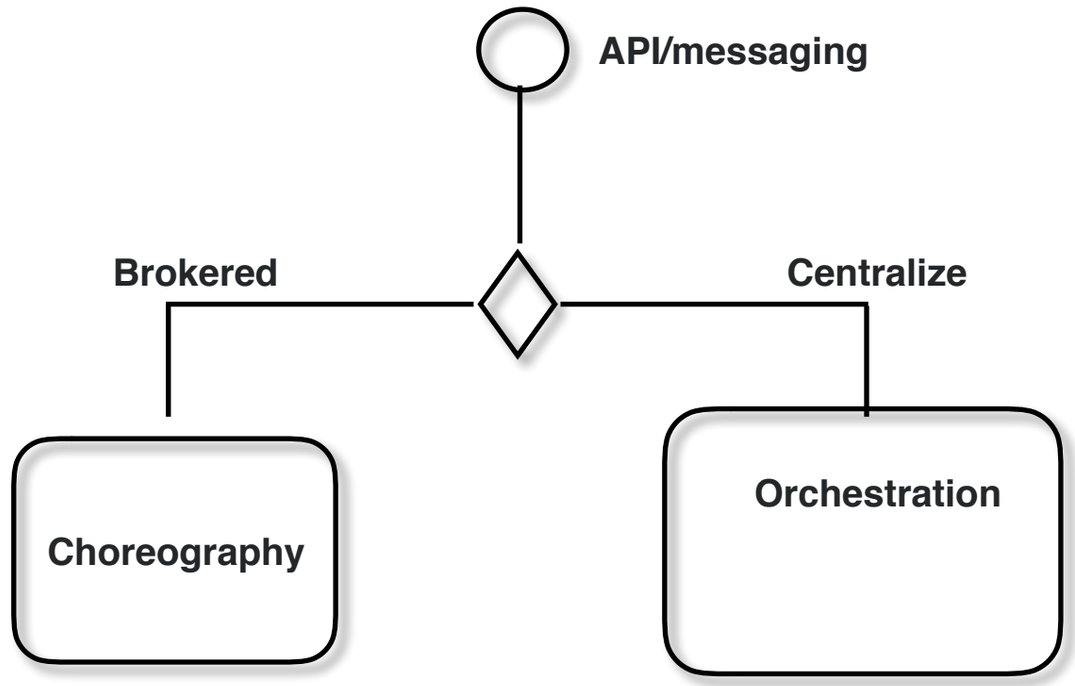
	Kafka	Pulsar	RabbitMQ (Mirrored)
Peak Throughput (MB/s)	605 MB/s	305 MB/s	38 MB/s
p99 Latency (ms)	5 ms (200 MB/s load)	25 ms (200 MB/s load)	1 ms* (reduced 30 MB/s load)
Number of Programming Languages Supported	17	6	22
Stack Overflow Questions	21,233	134	11,430
Pub/sub	Yes	Yes	Yes
Message routing	Medium	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

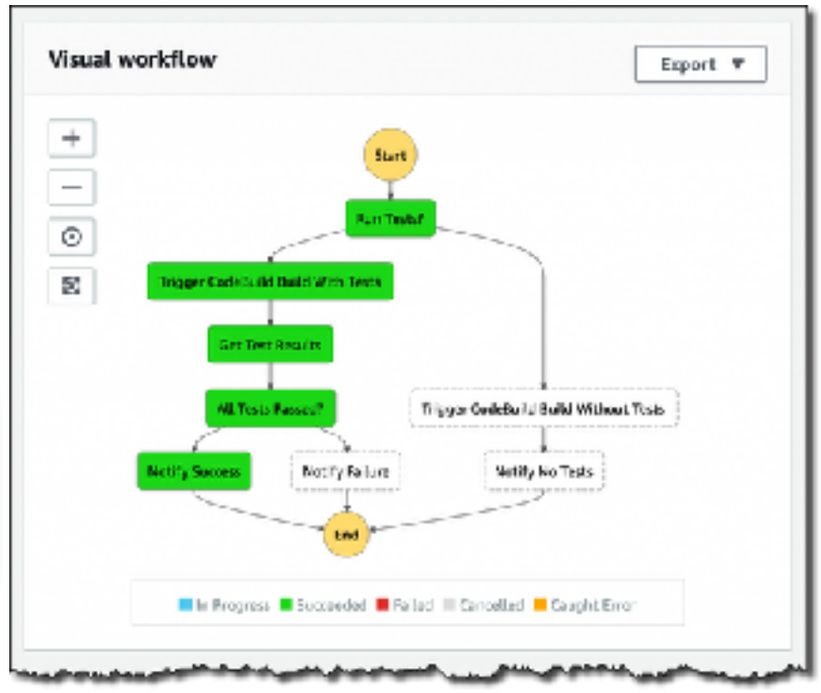
	Rabbitmq	Kafka	Redis
Scale	50K msg per second	1 million/ sec	1 million/ sec
Transient messages	Yes	No	Y
Persistent yes	Yes	Yes	Y
Protocol	AMQP		
One to one	Y	N	Y
One to many	Y	Y	Y
Advanced Message Queueing Protocol-based routing	Y	N	N
consumption pattern	Pull+Push	Pull	
Client Libraries	Py, java, php, .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

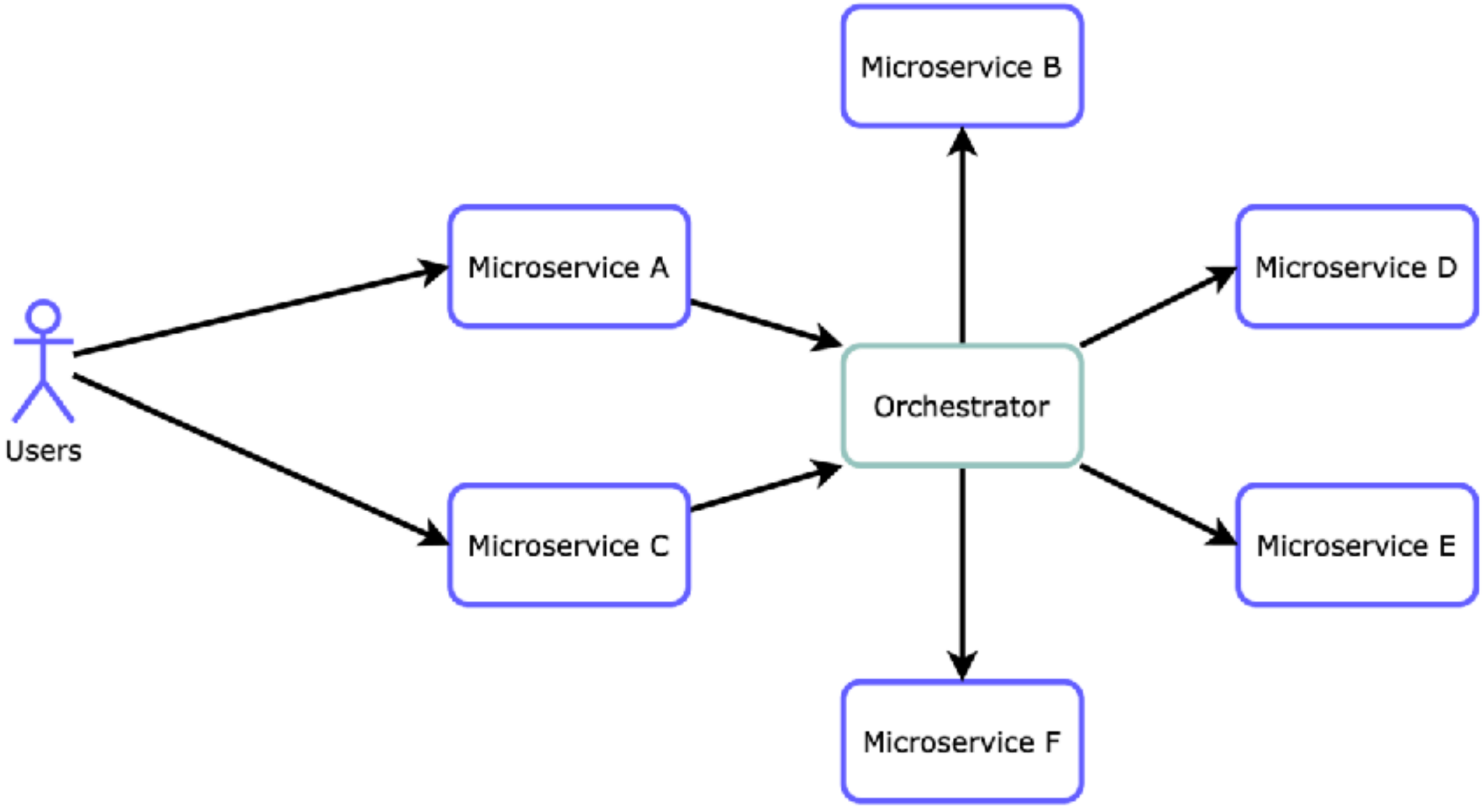
	Rabbitmq	Kafka	Redis
Scale	50K msg per second	1 million/ sec	1 million/ sec
Transient messages	Yes	No	Y
Persistent yes	Yes	Yes	Y
Protocol	AMQP		
One to one	Y	N	Y
One to many	Y	Y	Y
Advanced Message Queueing Protocol-based routing	Y	N	N
consumption pattern	Pull+Push	Pull	
Client Libraries	Py, java, php, .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

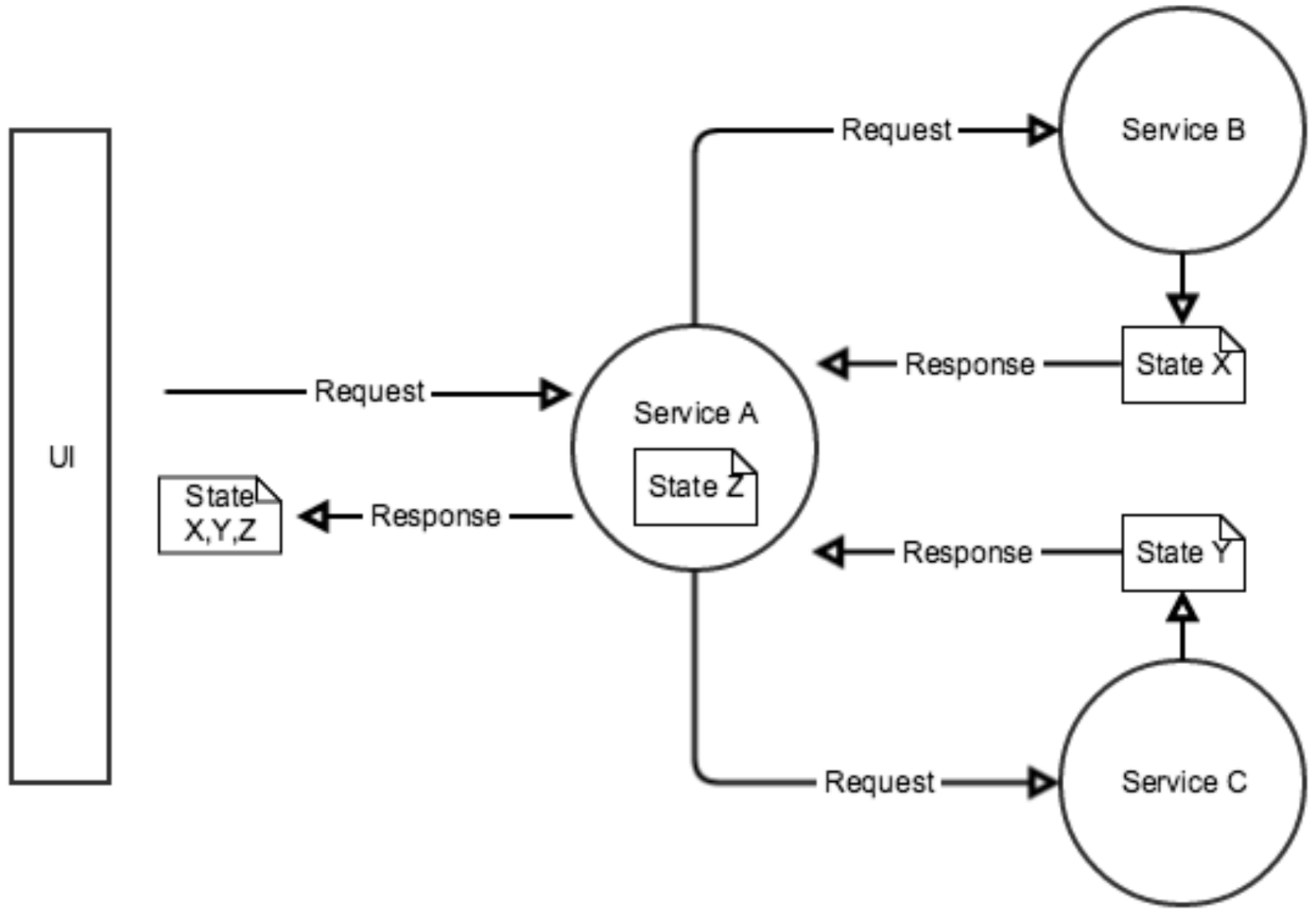
Choose Communication Patterns

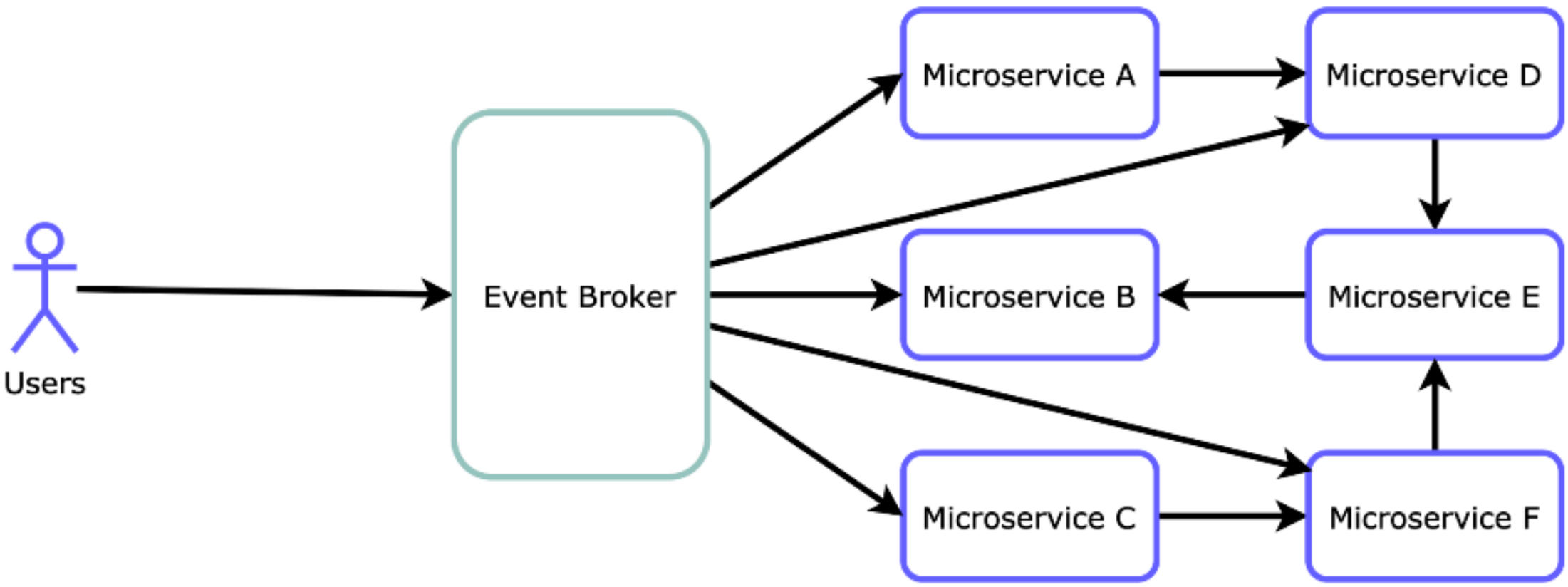




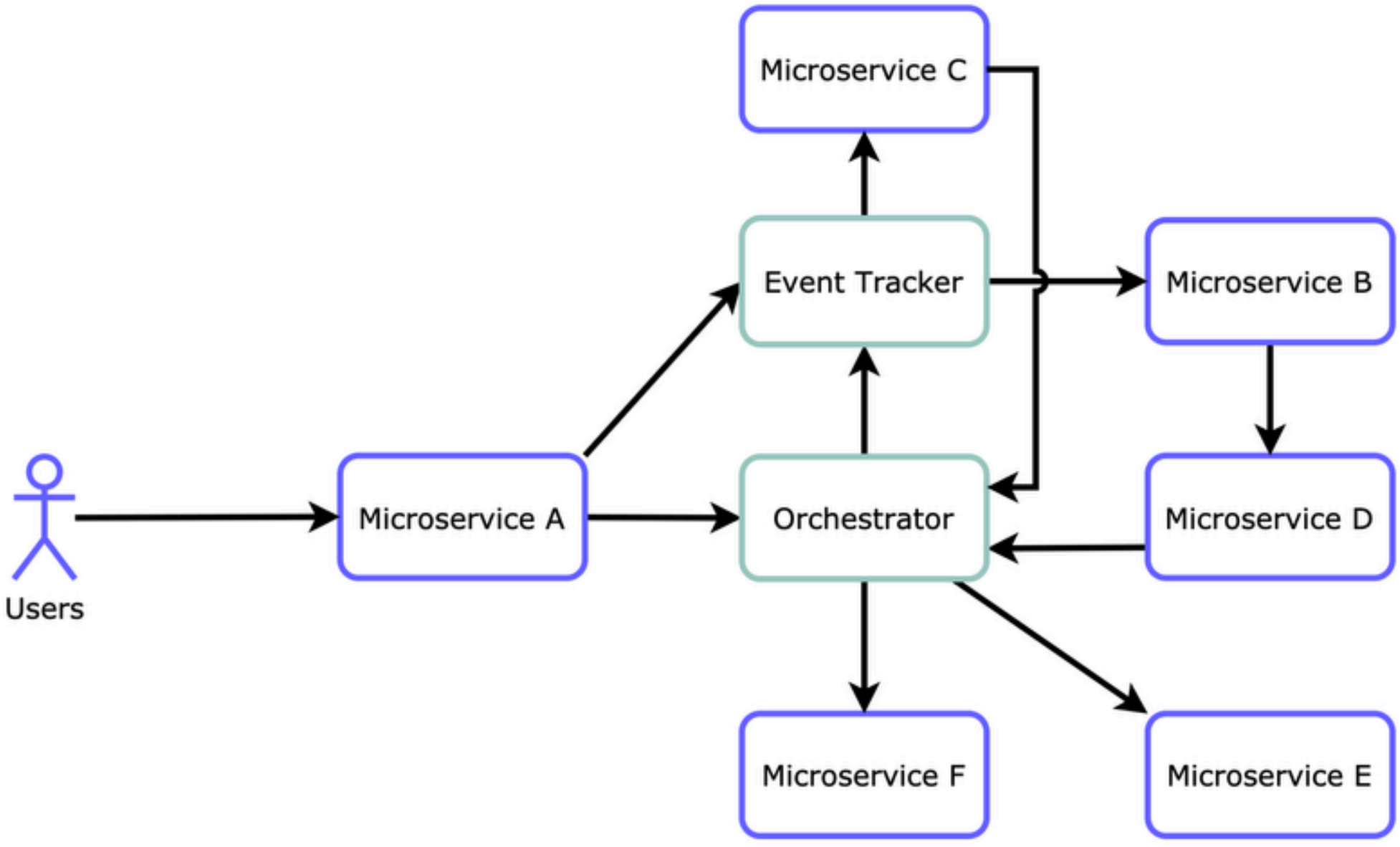
 # AWS Step
 # Azure Logic App
 # Custom

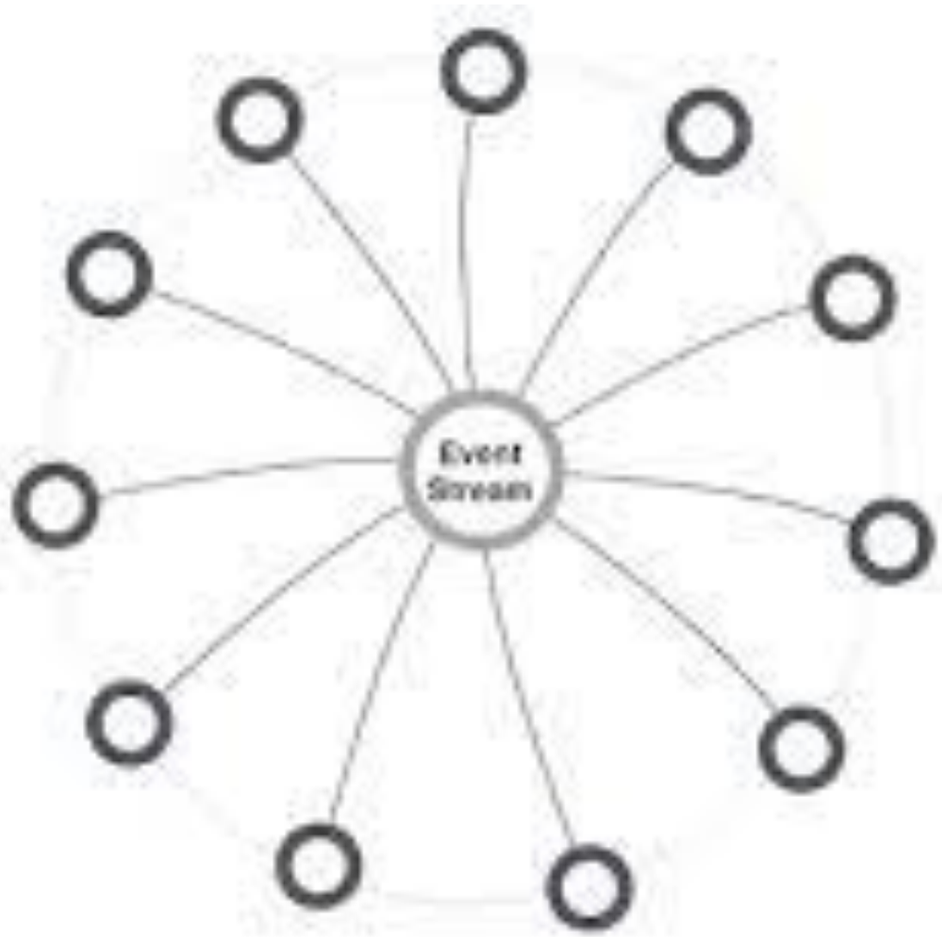

Microservice B
Microservice A
Microservice D
Users
Orchestrator
Microservice C
Microservice E
Microservice F


UI
Request
Service A
State Z
Request
Service B
Response
State X
State X,Y,Z
Response
Response
State Y
Request
Service C

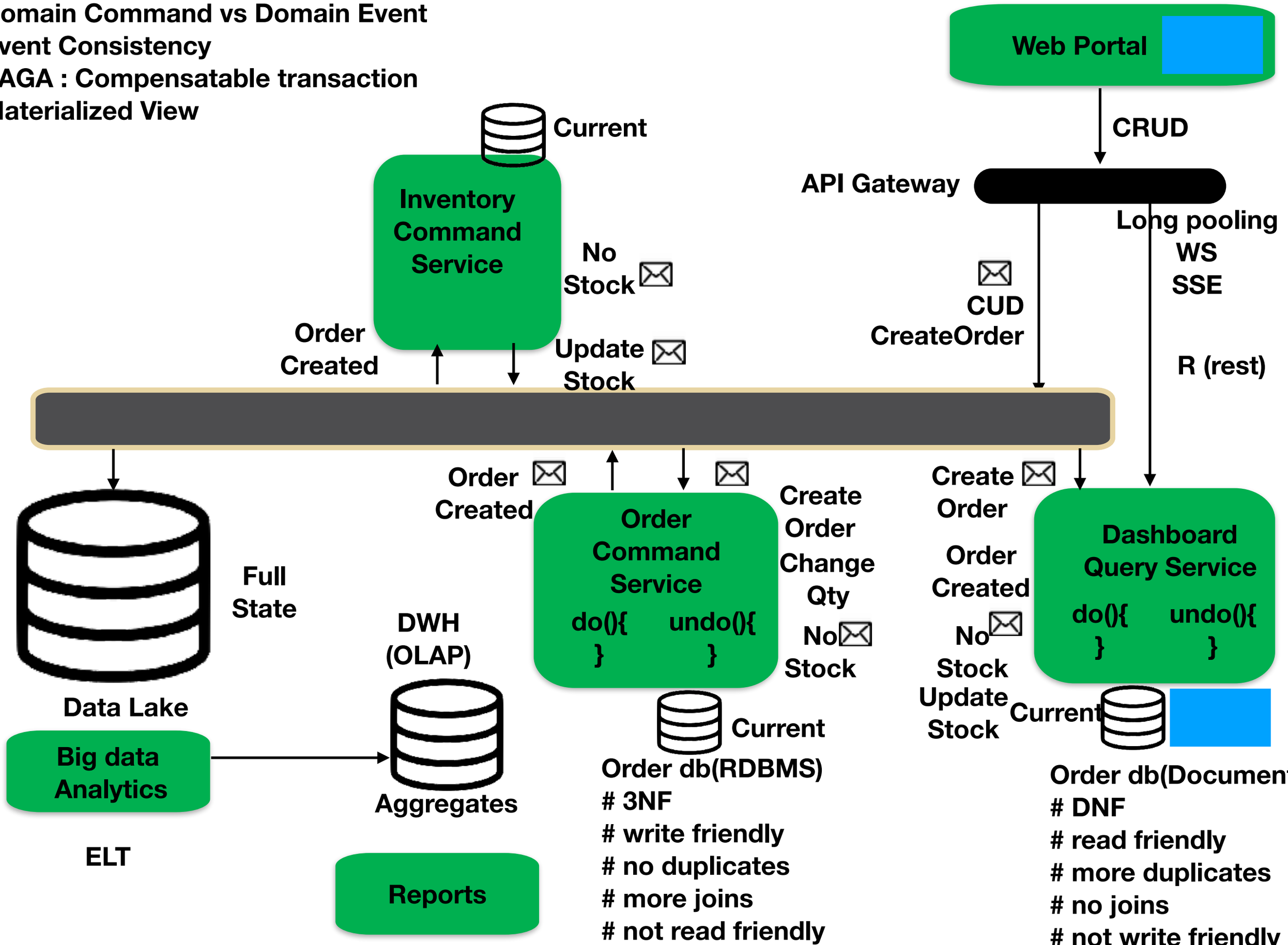

Microservice A
Microservice D
Users
Event Broker
Microservice B
Microservice E
Microservice C
Microservice F

Hybrid

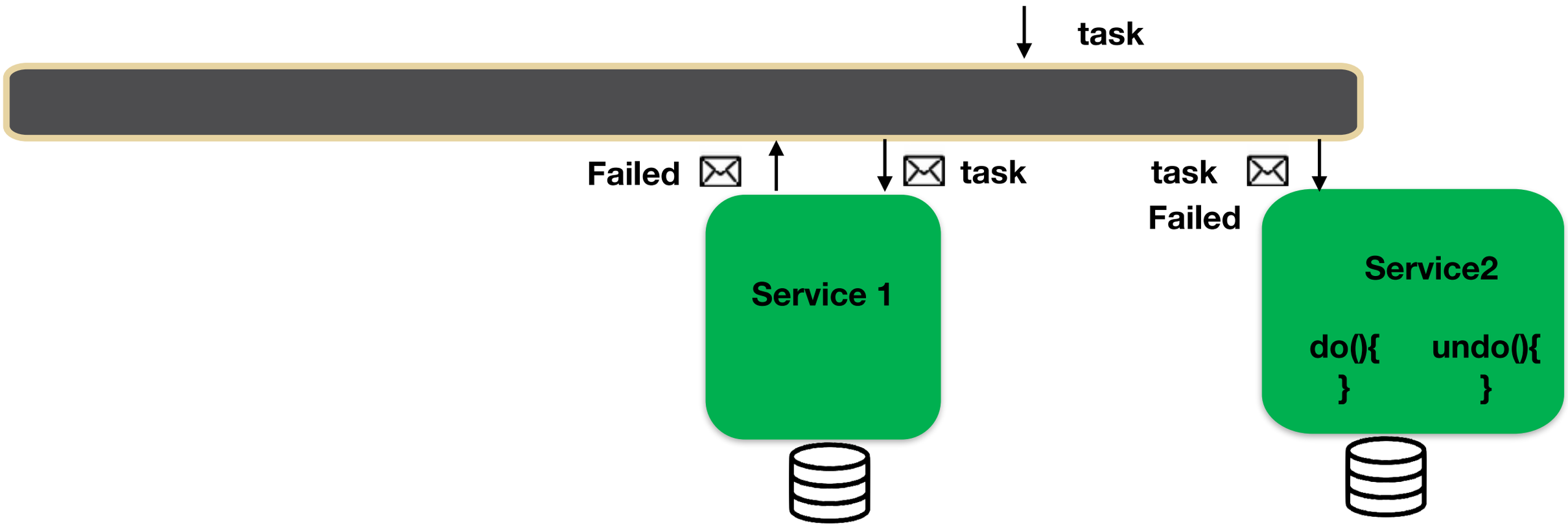

Microservice C
Event Tracker
Microservice B
Users
Microservice A
Orchestrator
Microservice D
Microservice F
Microservice E

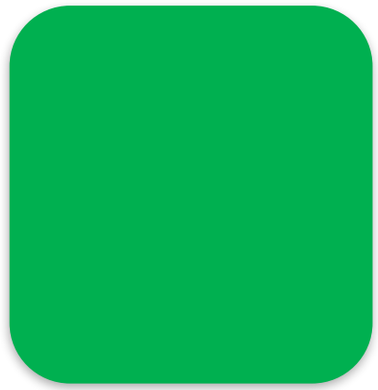

Event
Stream

CQRS (Command Query Responsibility)
Domain Command vs Domain Event
Event Consistency
SAGA : Compensatable transaction
Materialized View



SAGA



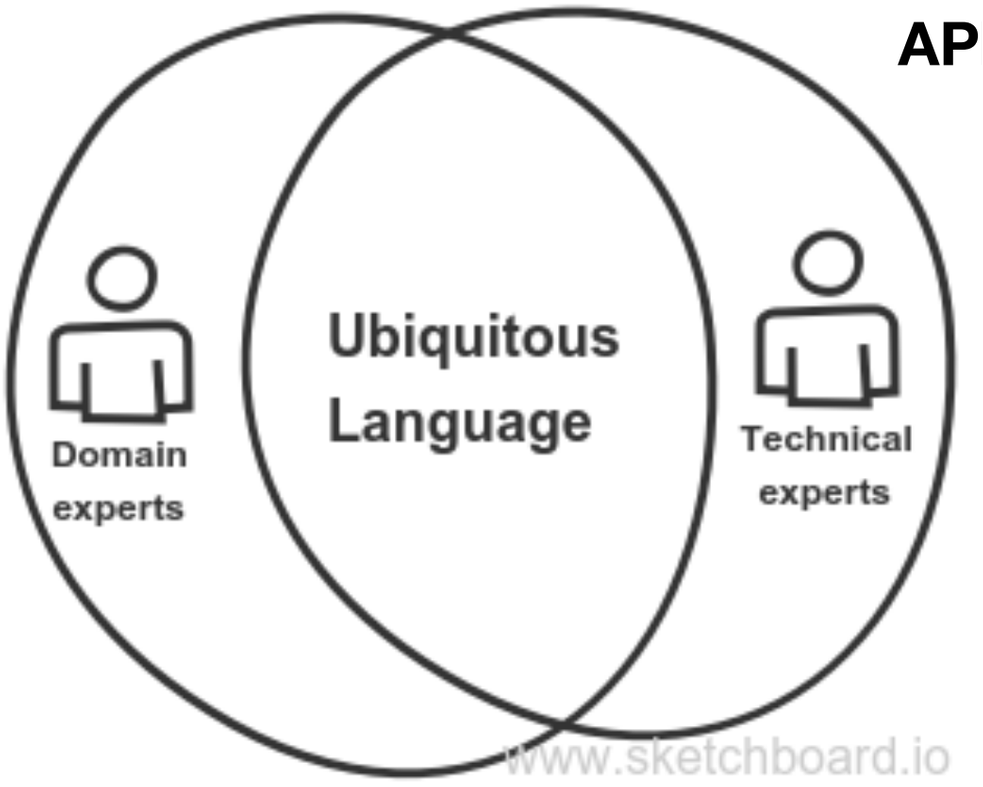


U3

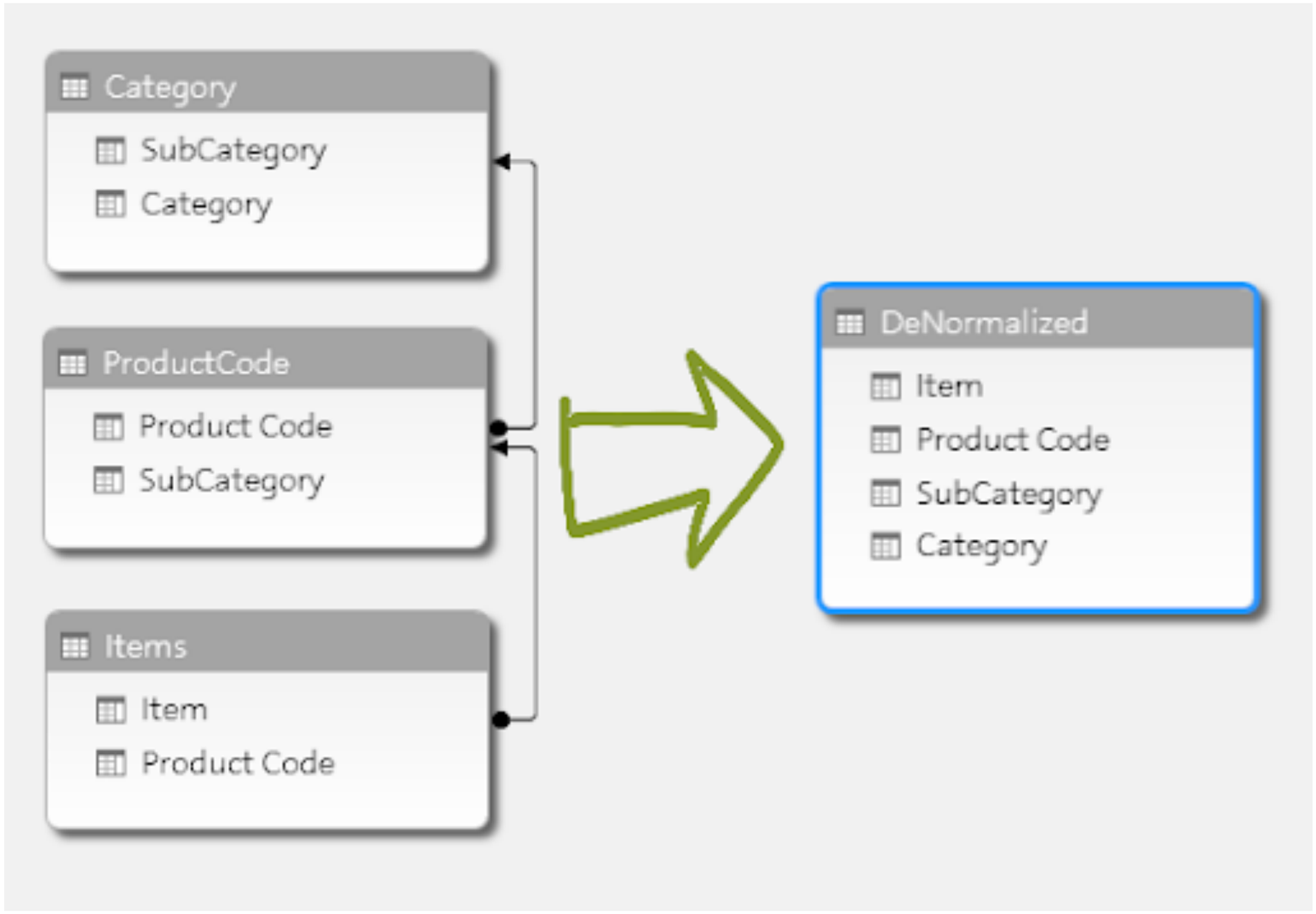
U2

U1

DDD



Domain Events


Category
SubCategory
Category
ProductCode
Product Code
SubCategory
Items
Item
Product Code
DeNormalized
Item
Product Code
SubCategory
Category

Portal

R (grpc)

CreateOrder

Order Command Services

CreateOrder

OrderCreated

Query Services

EDA
SAGA
Event sourcing
Eventual Consistency

Portal

CRUD (rest + json)

API

Account Command Services

Stream Analytics

CUD

CreateOrder

R (grpc)

OrderCreated

NoSufficient Stock

Order Command Services

CreateOrder

OrderCreated

NoSufficient Stock

Query Services

Data lake
full state

DWH
Snapshot state

RDBMS
write friendly
Current state
* NF

Materialized View

Document / RDBMS
read friendly
Current state
* DNF

Batch (Big Data) Analytics

DWH Analytics

Files

Databases (Cloud / On Premises)

Cloud Sources

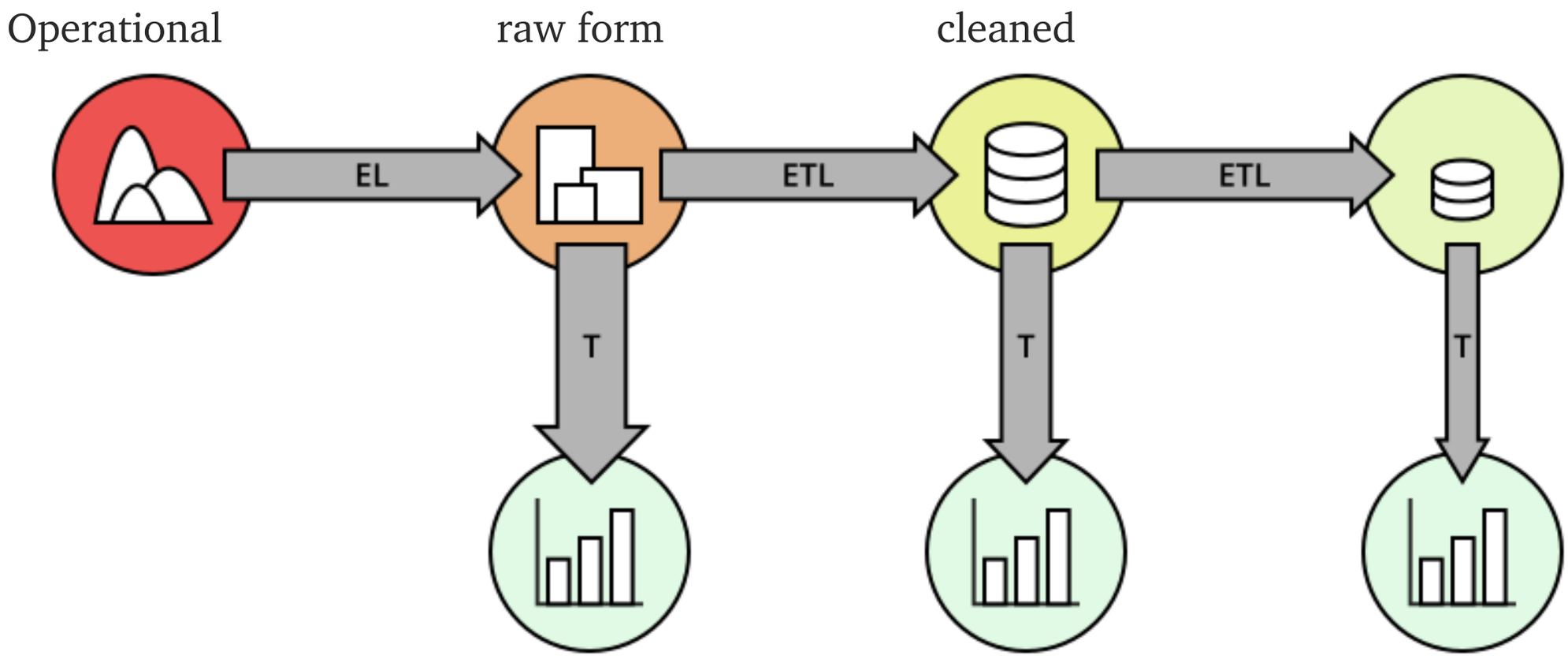
Audio/ Video/Documents

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

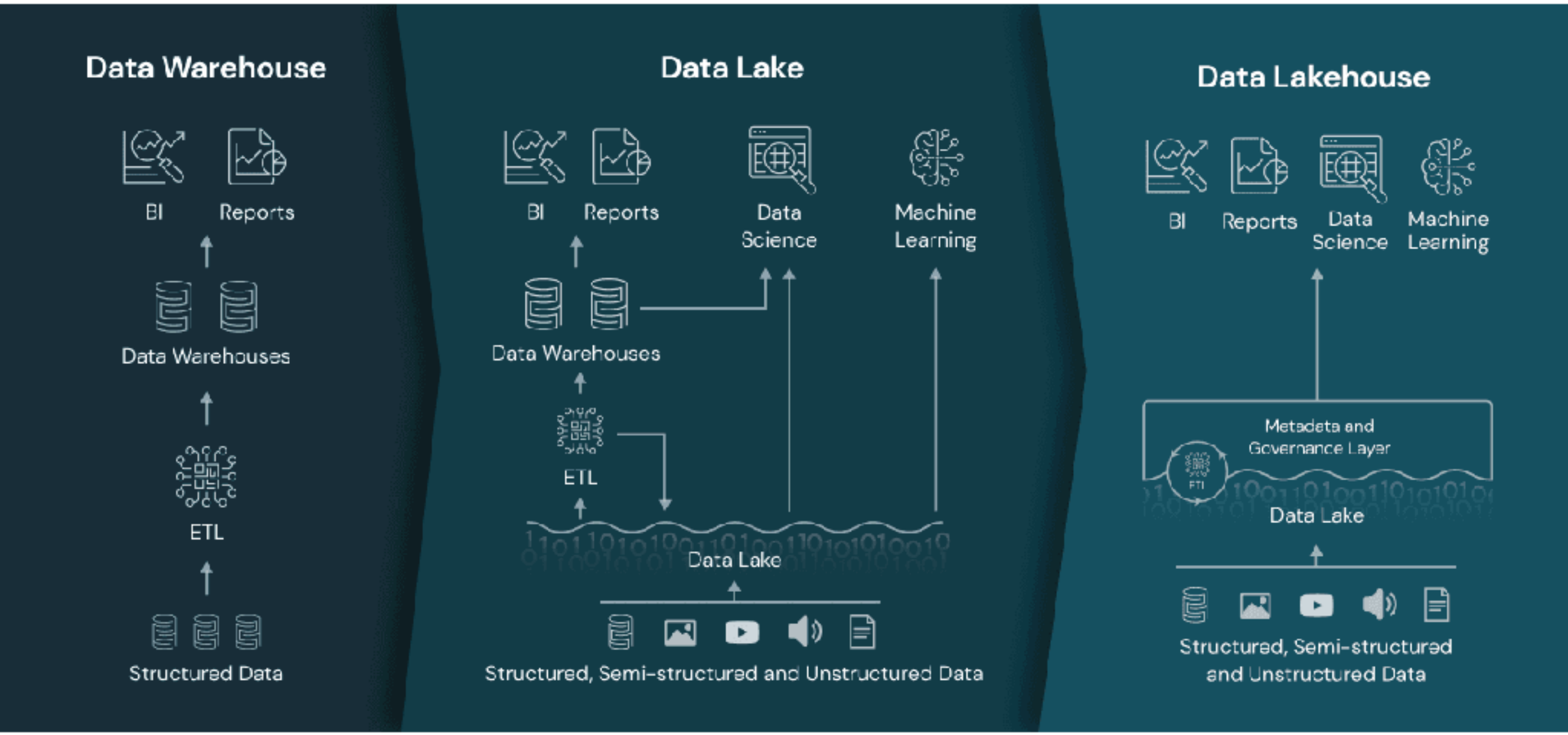
Insight/Analytics Tier

BI Tools

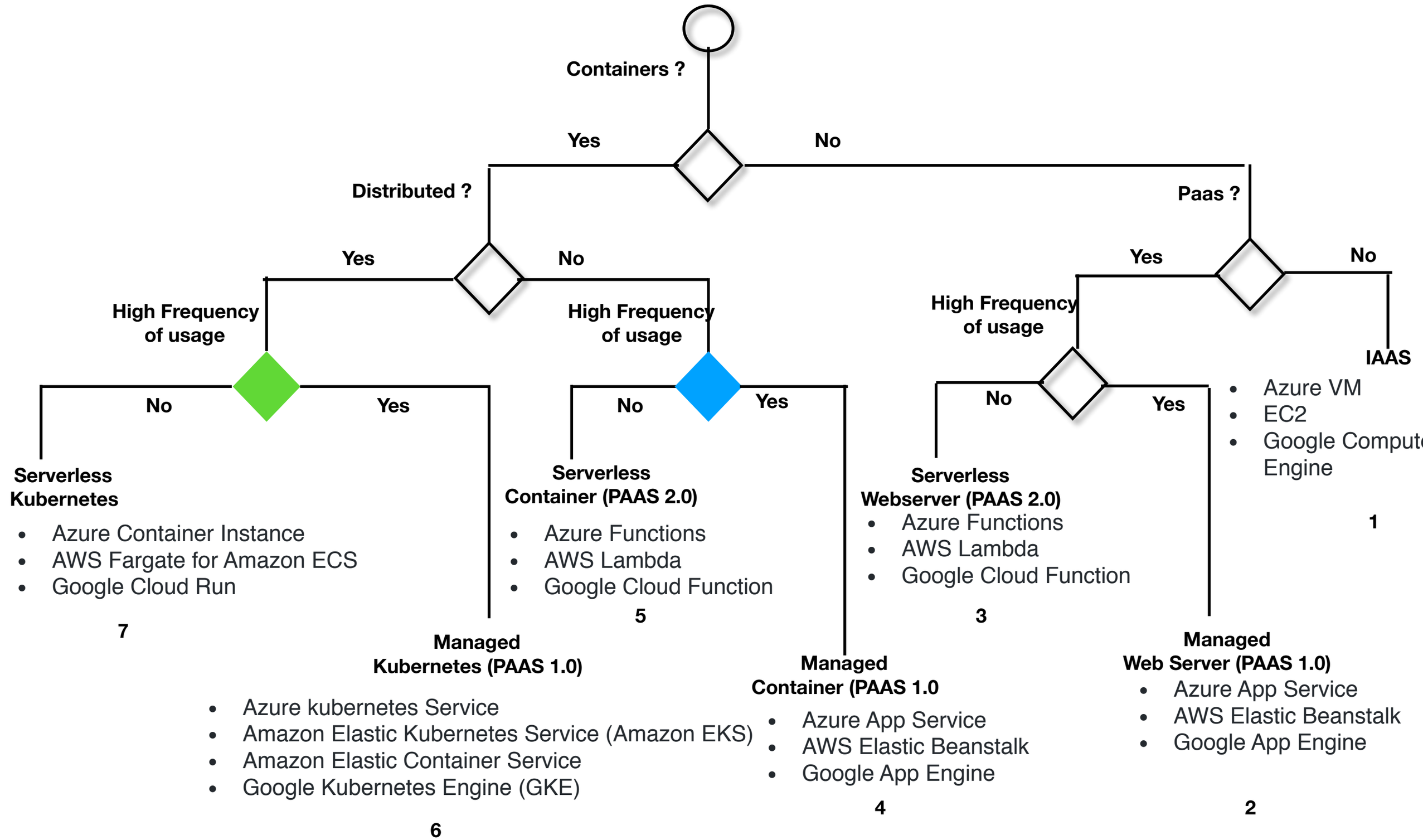
Applications

Automations

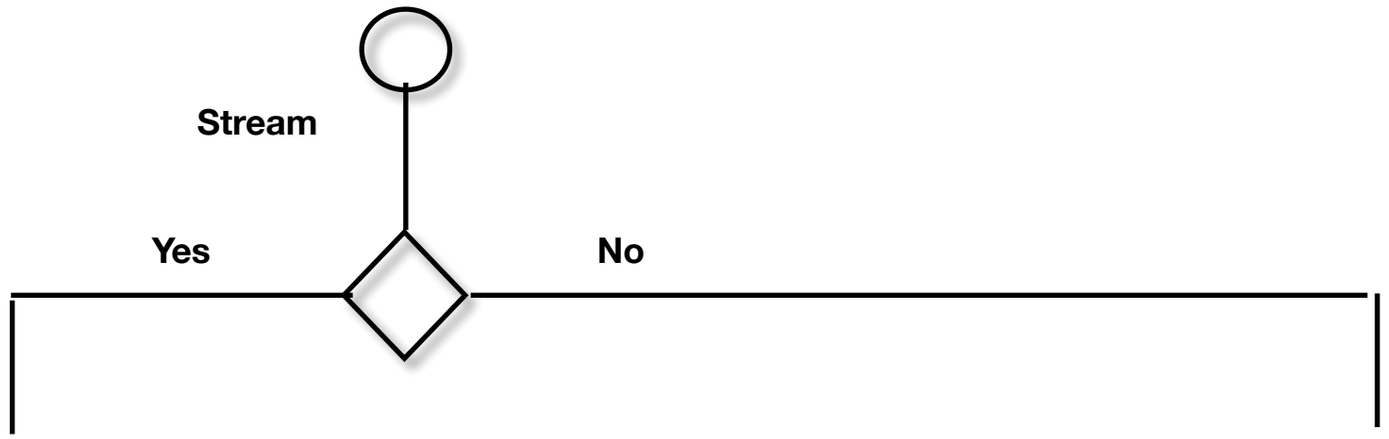
QuickSight
Power BI


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

Choose Operational Compute



Choose Analytical Compute



Stream Analytics

- Azure Stream Analytics
- AWS Kinesis Analytics
- Google Data Flow
- Apache Flink
- Apache Storm
- Spark Streaming

Batch Analytics

- Map Reduce
- Apache Spark
- Apache Flink