NoSQL and the idea of scaling up and out Databases

What lies beyond Relational

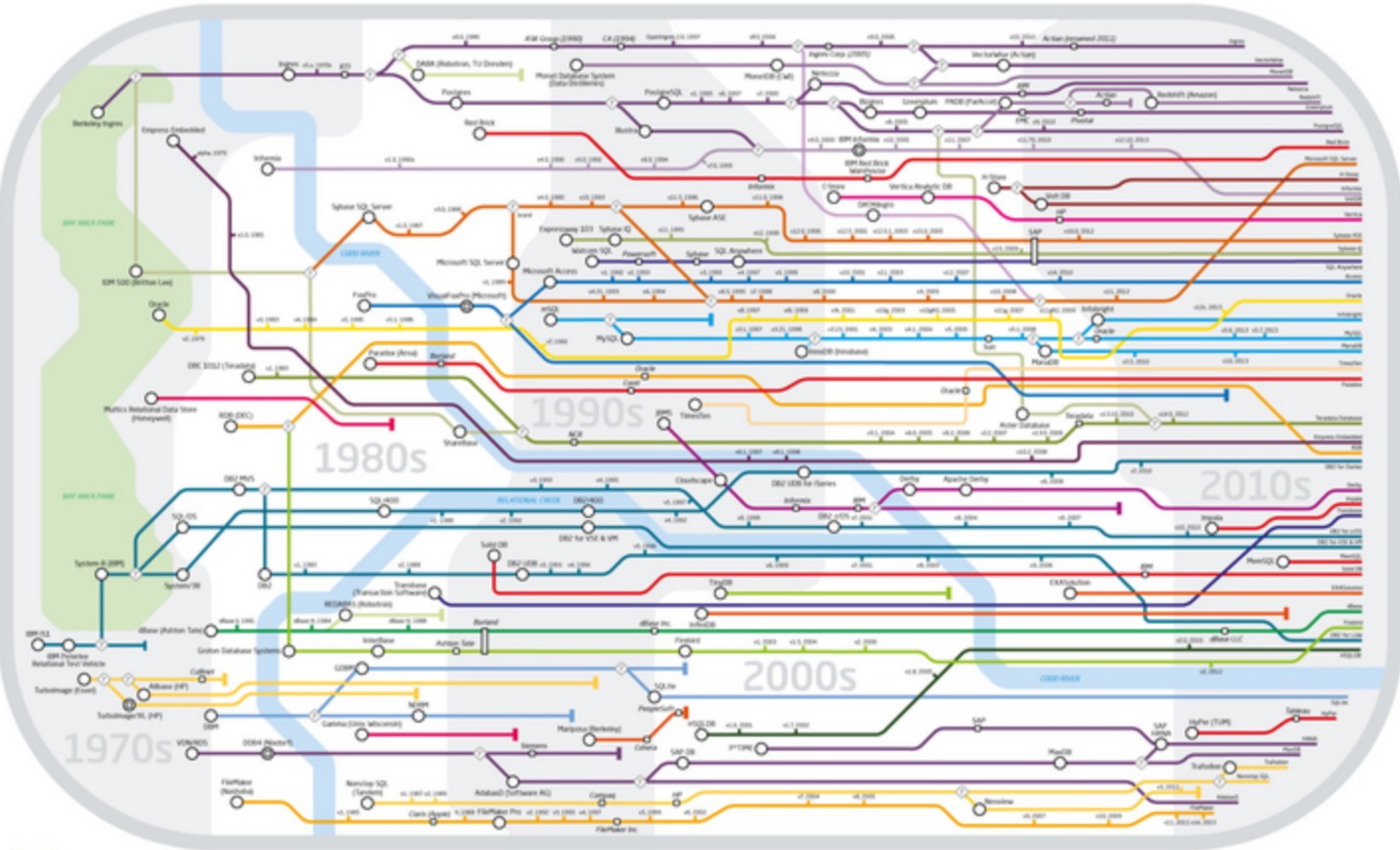
Prof. Yannis Velegrakis
Utrecht University

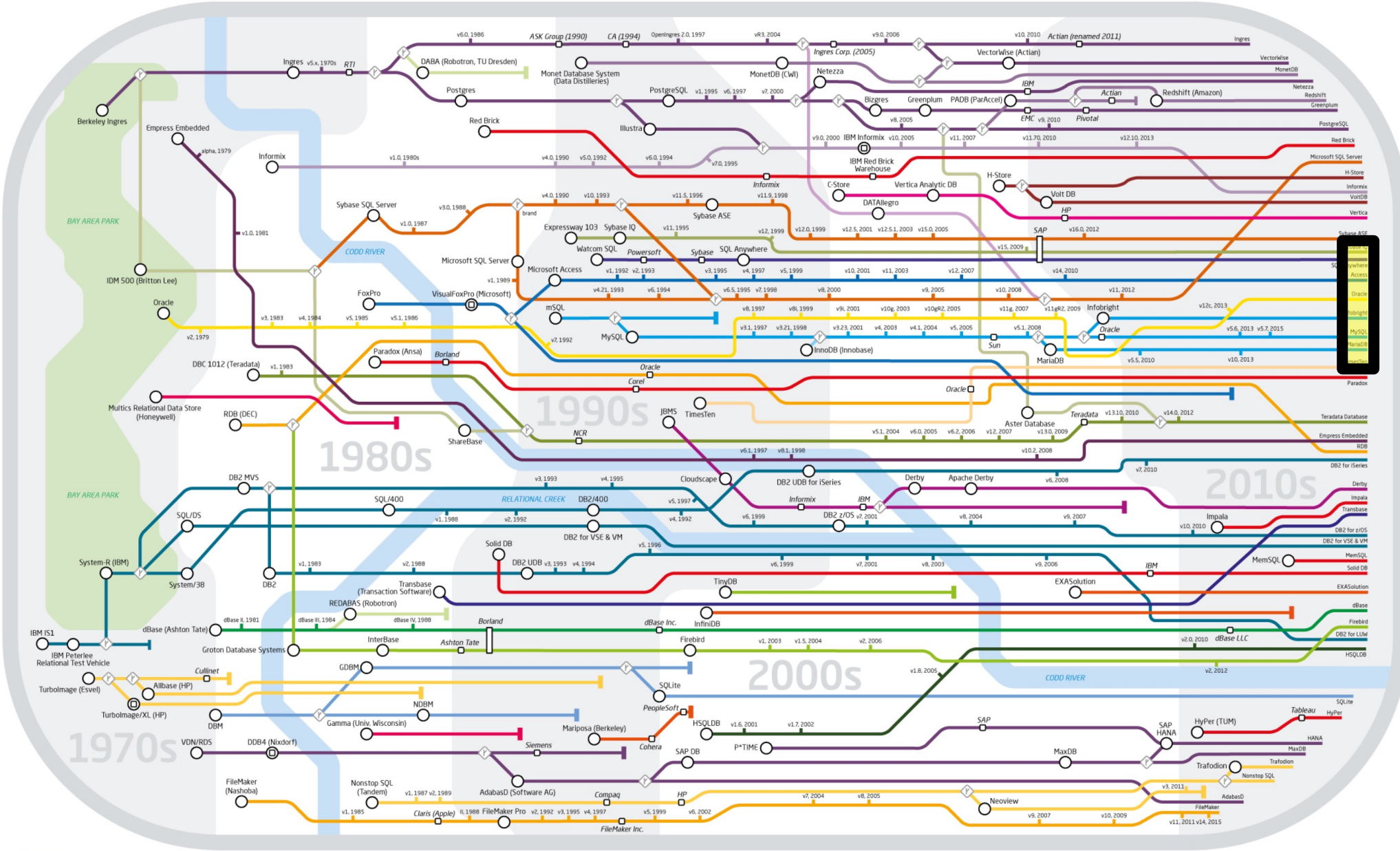
i.velegrakis@uu.nl
https://velgias.github.io

Disclaimer:

Slides courtesy of

- Martin Fowler
- Michalis Petropoulos
- Bill Howie
- Yannis Velegrakis


1970s
1980s
1990s
2000s
2010s
Berkeley Ingres
Ingres
Postgres
PostgreSQL
Red Brick
Informix
Empress Embedded
IDM 500 (Britton Lee)
Oracle
Sybase SQL Server
Microsoft SQL Server
Microsoft Access
Sybase ASE
Sybase IQ
FoxPro
mSQL
MySQL
MariaDB
Paradox (Ansa)
Borland
DBC 1012 (Teradata)
Teradata
Multics Relational Data Store (Honeywell)
RDB (DEC)
ShareBase
NCR
TimesTen
DB2 MVS
SQL/DS
SQL/400
DB2/400
DB2 for VSE & VM
System R (IBM)
System/38
DB2
Solid DB
DB2 UDB
Cloudscape
Derby
Apache Derby
Impala
MemSQL
dBase (Ashton Tate)
dBase Inc
InterBase
Firebird
Gamma (Univ. Wisconsin)
SQLite
HSQLDB
SAP HANA
MaxDB
HyPer (TUM)
Tableau
FileMaker (Nashoba)
FileMaker Pro
NonStop SQL (Tandem)
Neoview
Vertica
Vectorwise (Actian)
Netezza
Greenplum
Redshift (Amazon)
Actian
Pivotal
VoltDB
H-Store
Infobright
Cubrid
RELATIONAL CREEK
CODD RIVER
BAY AREA PARK


1970s
1980s
1990s
2000s
2010s
BAY AREA PARK
CODD RIVER
RELATIONAL CREEK
Berkeley Ingres
Ingres
RTI
Empress Embedded
Informix
Postgres
Red Brick
Illustra
PostgreSQL
Netezza
Bizgres
Greenplum
PADB (ParAccel)
Actian
Redshift (Amazon)
VectorWise (Actian)
Ingres Corp. (2005)
Actian (renamed 2011)
DABA (Robotron, TU Dresden)
Monet Database System (Data Distilleries)
MonetDB (CWI)
IBM Informix
IBM Red Brick Warehouse
EMC
Pivotal
IDM 500 (Britton Lee)
Oracle
Sybase SQL Server
Sybase ASE
Sybase IQ
Expressway 103
Watcom SQL
Powersoft
Sybase
SQL Anywhere
Microsoft SQL Server
Microsoft Access
FoxPro
VisualFoxPro (Microsoft)
mSQL
MySQL
InnoDB (Innobase)
Sun
MariaDB
Infobright
Oracle
C-Store
Vertica Analytic DB
DATAllegro
Volt DB
HP
Vertica
H-Store
SAP
Aster Database
Teradata
Paradox (Ansa)
Borland
Corel
DBC 1012 (Teradata)
Multics Relational Data Store (Honeywell)
RDB (DEC)
ShareBase
NCR
IBMS
TimesTen
Paradox
Teradata Database
Empress Embedded
RDB
DB2 for iSeries
DB2 MVS
SQL/400
DB2/400
SQL/DS
System/38
DB2
DB2 for VSE & VM
DB2 z/OS
Cloudscape
DB2 UDB for iSeries
Derby
Apache Derby
Informix
IBM
Impala
Transbase
DB2 for z/OS
DB2 for VSE & VM
System-R (IBM)
Solid DB
DB2 UDB
Transbase (Transaction Software)
REDABAS (Robotron)
TinyDB
EXASolution
MemSQL
Solid DB
EXASolution
IBM IS1
IBM Peterlee Relational Test Vehicle
dBase (Ashton Tate)
Groton Database Systems
InterBase
Ashton Tate
dBase Inc.
InfiniDB
Firebird
dBase LLC
dBase
Firebird
DB2 for LUW
HSQLDB
Cullinet
Turbolmage (Esvel)
Allbase (HP)
Turbolmage/XL (HP)
GDBM
NDBM
DBM
Gamma (Univ. Wisconsin)
Mariposa (Berkeley)
PeopleSoft
Cohera
SQLite
HSQLDB
SAP
SAP HANA
HyPer (TUM)
Tableau
HyPer
HANA
MaxDB
Trafodion
Nonstop SQL
AdabasD
FileMaker
VDN/RDS
DDB4 (Nixdorf)
Siemens
SAP DB
P*TIME
MaxDB
FileMaker (Nashoba)
Nonstop SQL (Tandem)
AdabasD (Software AG)
Compaq
HP
Claris (Apple)
FileMaker Pro
FileMaker Inc.
Neoview


"DATA IS THE NEW GOLD"

The Need for Data Storage

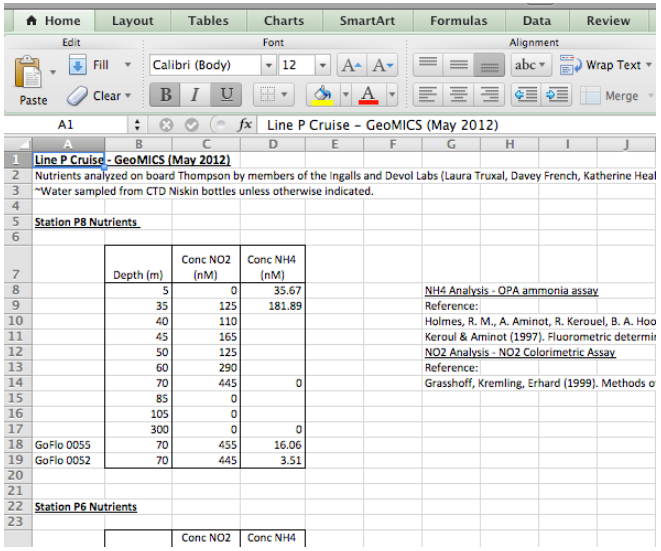
How do we store data?

Why would I want a database?

What problem do they solve?

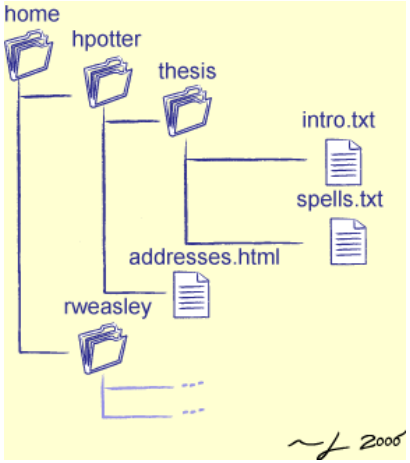
1. Sharing
   Support concurrent access by multiple readers and writers
2. Data Model Enforcement
   Make sure all applications see clean, organized data
3. Scale
   Work with datasets too large to fit in memory
4. Flexibility
   Use the data in new, unanticipated ways

How do we store data?



	A	B	C	D	E	F	G	H	I	J
1	Line P Cruise - GeoMICS (May 2012)									
2	Nutrients analyzed on board Thompson by members of the Ingalls and Devol Labs (Laura Truxal, Davey French, Katherine Heal									
3	~Water sampled from CTD Niskin bottles unless otherwise indicated.									
4										
5	Station P8 Nutrients									
6										
7		Depth (m)	Conc NO2 (nM)	Conc NH4 (nM)						
8		5	0	35.67			NH4 Analysis - OPA ammonia assay			
9		35	125	181.89			Reference:			
10		40	110				Holmes, R. M., A. Aminot, R. Kerouel, B. A. Hoo			
11		45	165				Keroul & Aminot (1997). Fluorometric determi			
12		50	125				NO2 Analysis - NO2 Colorimetric Assay			
13		60	290				Reference:			
14		70	445	0			Grasshoff, Kremling, Erhard (1999). Methods o			
15		85	0							
16		105	0							
17		300	0	0						
18	GoFlo 0055	70	455	16.06						
19	GoFlo 0052	70	445	3.51						
20										
21										
22	Station P6 Nutrients									
23										
			Conc NO2	Conc NH4						

What is the *data model*?



ANNOTATIONSUMMARY–COMBINEDORFANNOTATION16_Phaeo_genome

###query	length	COG hit #1	e-value #1	identity #1	score #1	hit length #1	description #1
chr_4[480001-580000].287	4500						
chr_4[560001-660000].1	3556						
chr_9[400001-500000].503	4211	COG4547	2.00E-04	19	44.6	620	Cobalamin biosynthesis protein
chr_9[320001-420000].548	2833	COG5406	2.00E-04	38	43.9	1001	Nucleosome binding factor SPN
chr_27[320001-404298].20	3991	COG4547	5.00E-05	18	46.2	620	Cobalamin biosynthesis protein
chr_26[320001-420000].378	3963	COG5099	5.00E-05	17	46.2	777	RNA-binding protein of the Puf
chr_26[400001-441226].196	2949	COG5099	2.00E-04	17	43.9	777	RNA-binding protein of the Puf
chr_24[160001-260000].65	3542						
chr_5[720001-820000].339	3141	COG5099	4.00E-09	20	59.3	777	RNA-binding protein of the Puf
chr_9[160001-260000].243	3002	COG5077	1.00E-25	26	114	1089	Ubiquitin carboxyl-terminal hyd
chr_12[720001-820000].86	2895	COG5032	2.00E-09	30	60.5	2105	Phosphatidylinositol kinase and
chr_12[800001-900000].109	1463	COG5032	1.00E-09	30	60.1	2105	Phosphatidylinositol kinase and
chr_11[1-100000].70	2886						
chr_11[80001-180000].100	1523						

What is a Data Model?

Three components:

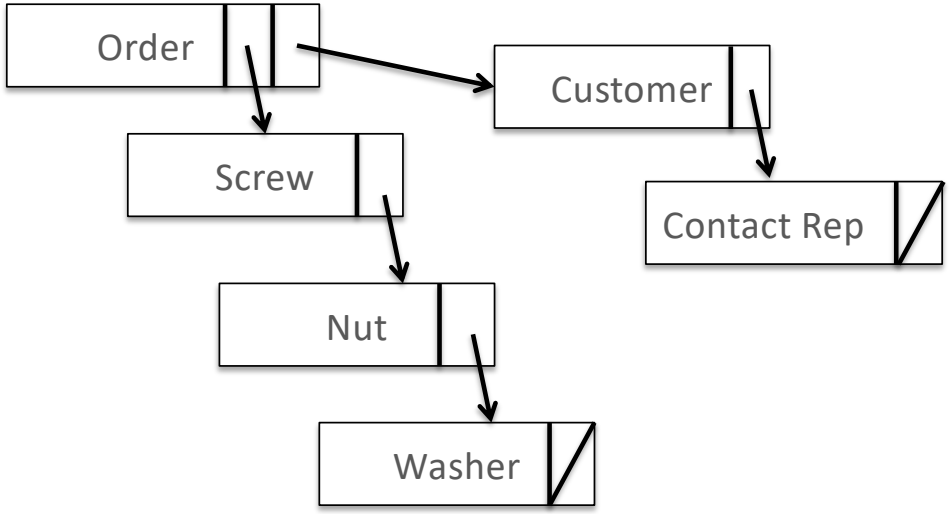
1. Structures
2. Constraints
3. Operations

Examples

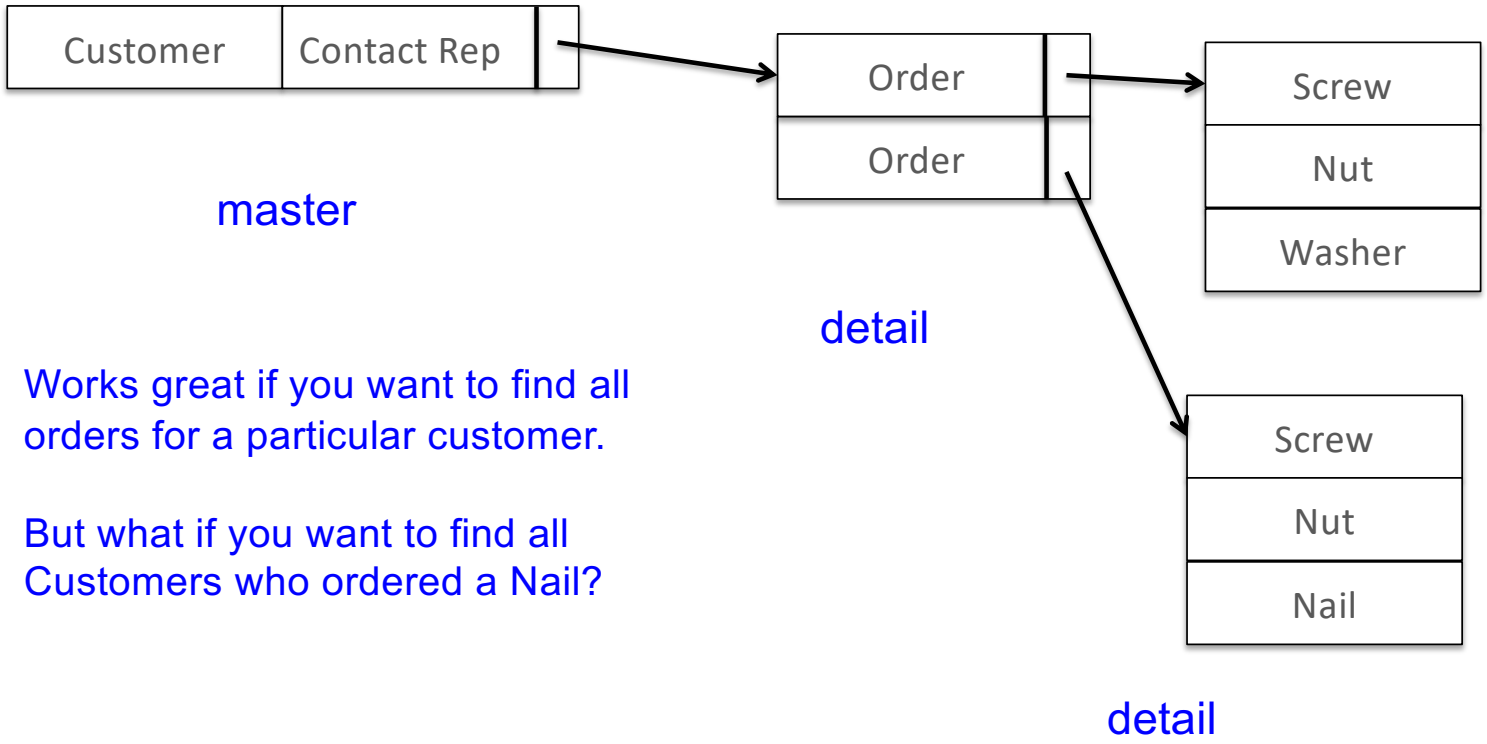
1. Structures
   - rows and columns?
   - nodes and edges?
   - key-value pairs?
   - a sequence of bytes?
2. Constraints
   - all rows must have the same number of columns
   - all values in one column must have the same type
   - a child cannot have two parents
3. Operations
   - find the value of key x
   - find the rows where column “lastname” is “Jordan”
   - get the next N bytes

Database: A collection of information organized to afford efficient retrieval

Historical Example: Network Databases



Historical Example: Hierarchical Databases



Works great if you want to find all orders for a particular customer.

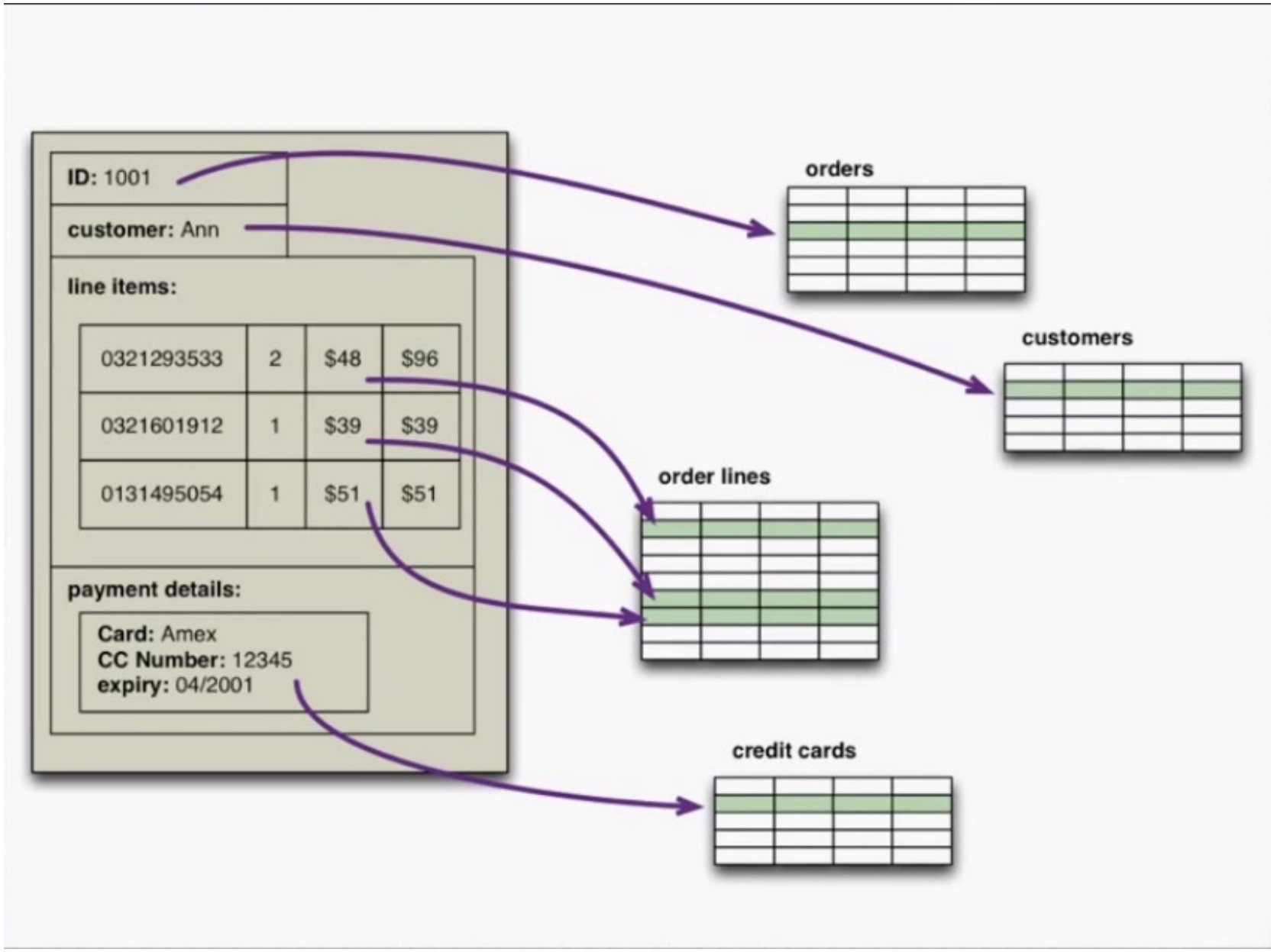
But what if you want to find all Customers who ordered a Nail?

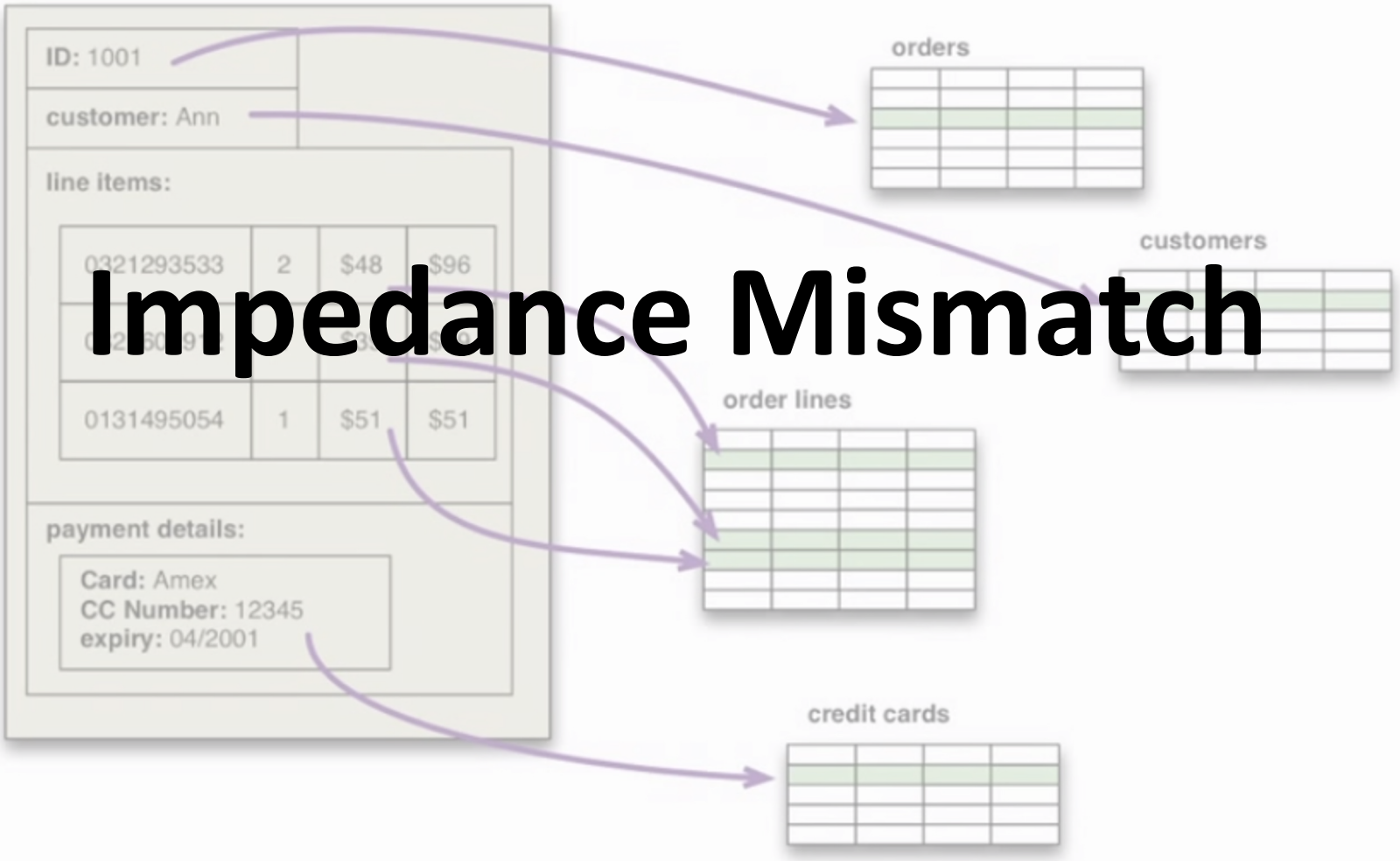
Relational Databases (Codd 1970)

- Everything is a table
- Every row in a table has the same columns
- Relationships are implicit: no pointers

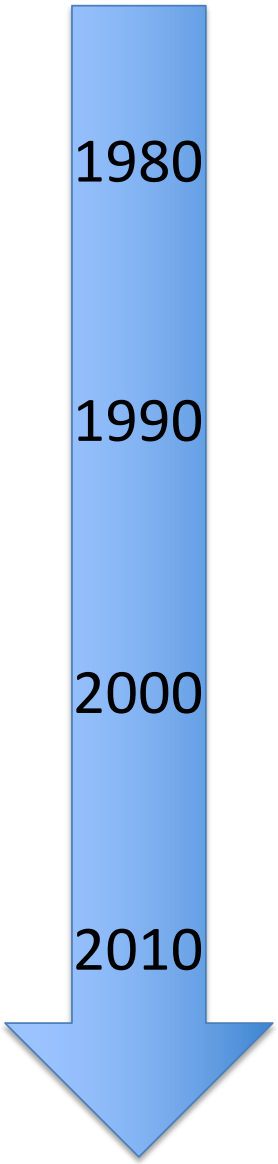
Course	Student Id
CSE 344	223…
CSE 344	244…
CSE 514	255..
CSE 514	244…

Student Id	Student Name
223…	Jane
244…	Joe
255..	Susan

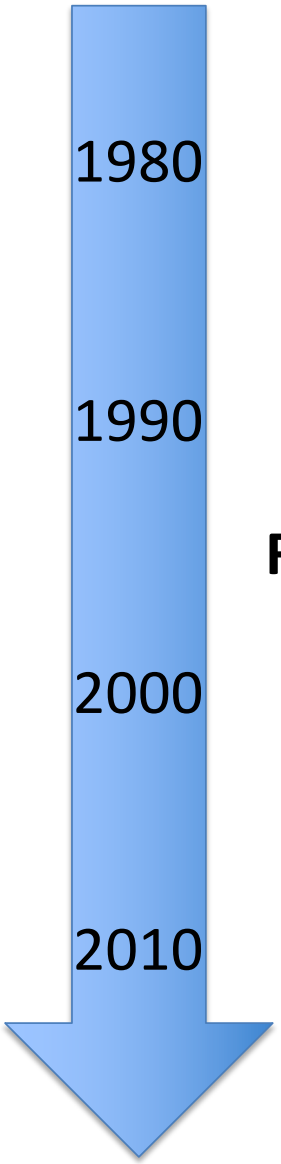

ID: 1001
customer: Ann
line items:
0321293533
2
$48
$96
0321601912
1
$39
$39
0131495054
1
$51
$51
payment details:
Card: Amex
CC Number: 12345
expiry: 04/2001
orders
customers
order lines
credit cards



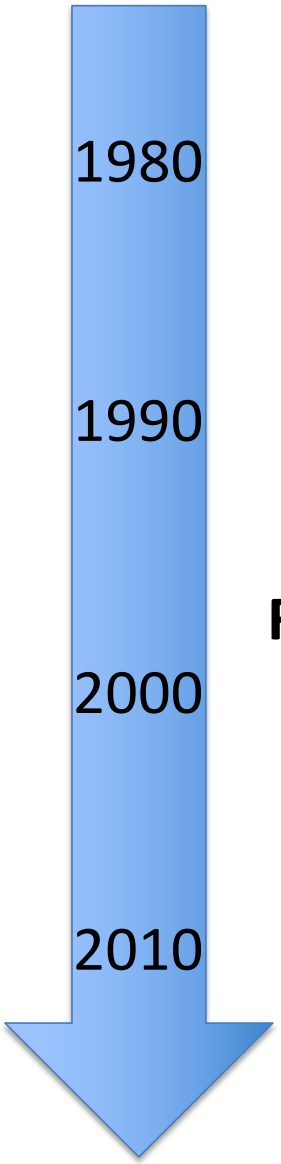
Impedance Mismatch



Rise of the Relational Databases



Rise of the Object Databases

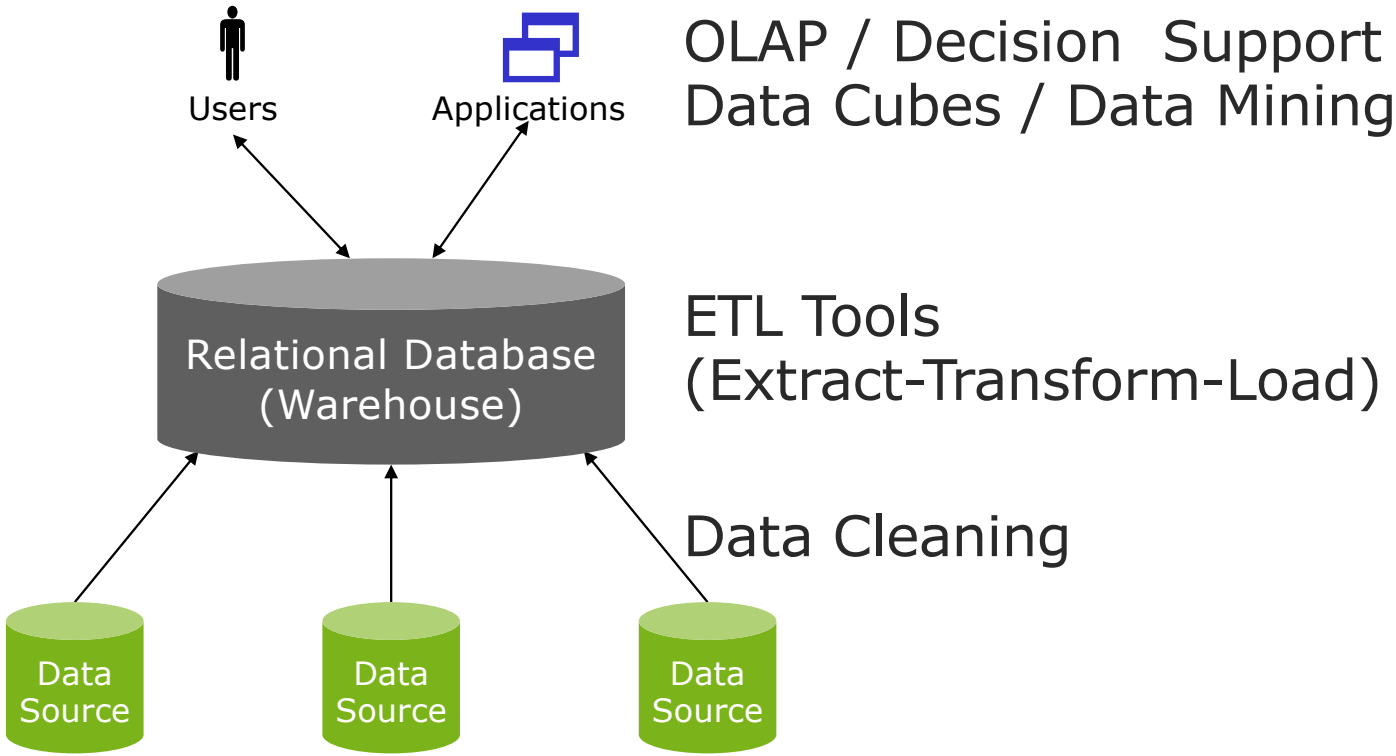


Relational Dominance

The Need for Integration

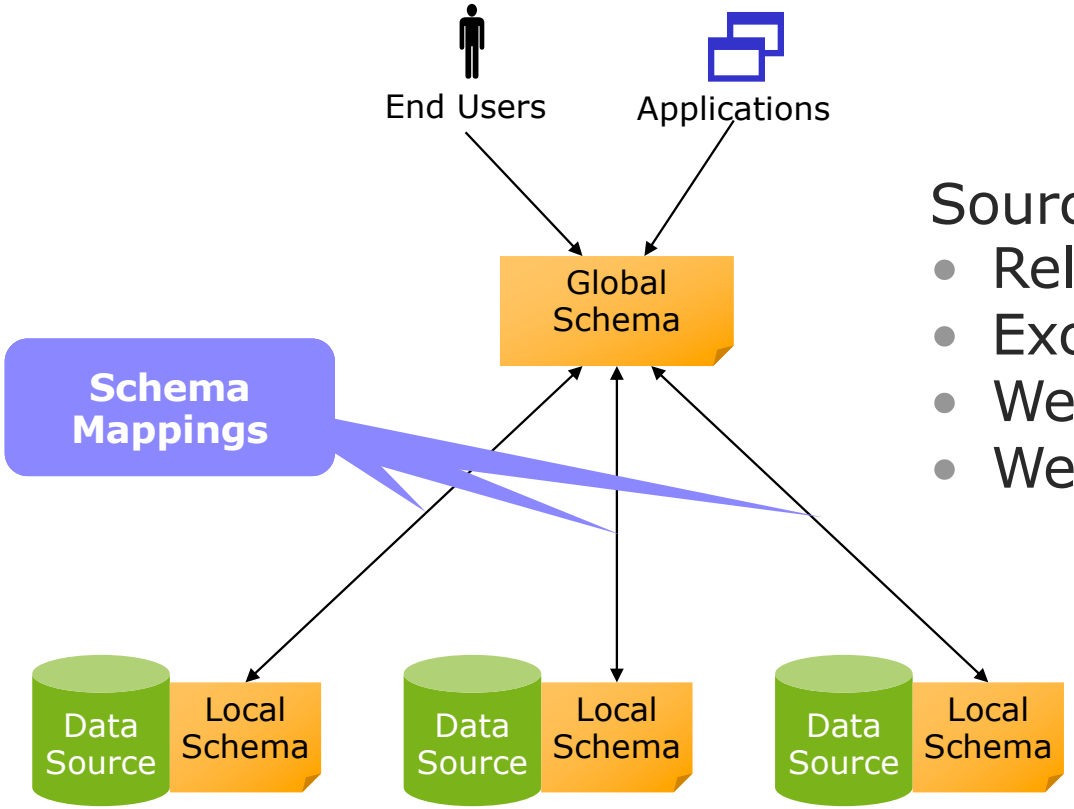

Speculative Retailers Web Application
User sessions
Financial Data
Shopping Cart
Recomendations
Product Catalog
Reporting
Analytics
User activity logs

Data Warehouse Architecture



Virtual Integration Architecture

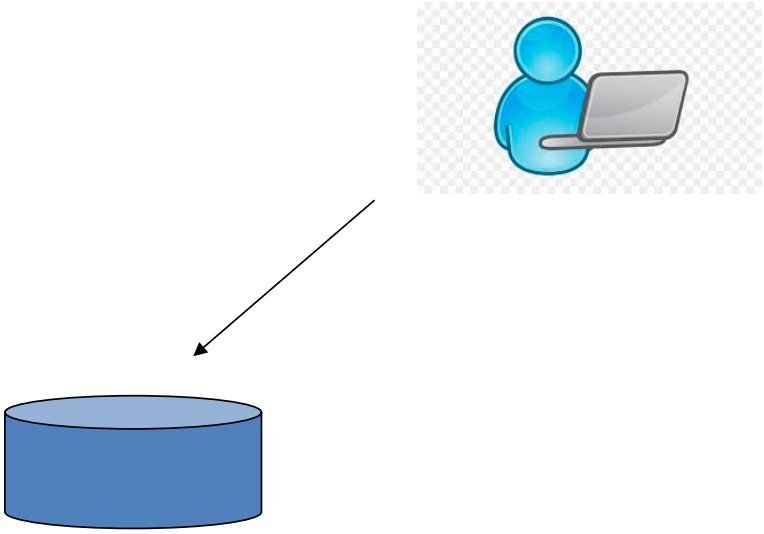
Design-Time



Sources can be:

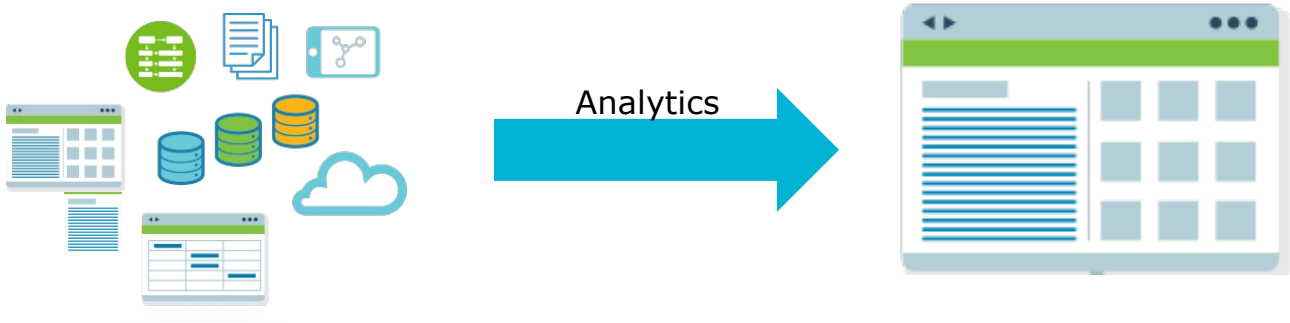
- Relational DBs
- Excel Files
- Web Sites
- Web Services

Data Management: The Old Model



Data Analytics: The New Model

Extracting Value from Data



Real Life Data Use Cases

- Education
  - Correlate academic effectiveness with various factors
- Healthcare
  - Preventive & Personalized
- Urban Planning
  - Fusion of high fidelity geographical data
- Intelligent Transportation
  - Analysis and visualization of live and detailed road network data
- Environmental Modeling
  - Sensor Networks

Real Life Data Use Cases

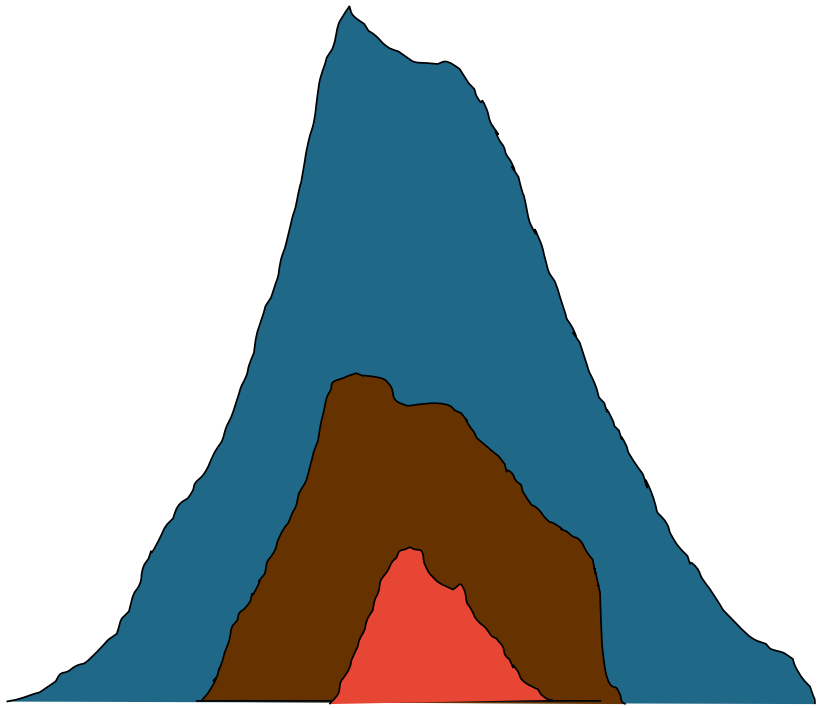
- Energy Saving
  - Unveiling Patterns of Use
- Financial Systemic Risk Analysis
  - Analyze contracts and relationships among entities
- Security
  - Social Networks, Financial Transactions, Transportation
- • Computer Security
  - Log Info, Data Transfers and Transactions
- Democracy
  - Citizens understand how the government operates

The Need for Scaling Up

Machine Generated Data

Internet Users

Employees / Data Engineers



2020 This Is What Happens In An Internet Minute

2021 This Is What Happens In An Internet Minute

A DAY IN DATA

The exponential growth of data is undisputed, but the numbers behind this explosion – fuelled by internet of things and the use of connected devices – are hard to comprehend, particularly when looked at in the context of one day

DEMYSTIFIYING DATA UNITS

From the more familiar 'bit' or 'megabyte', larger units of measurement are more frequently being used to explain the masses of data

Unit		Value	Size
b	bit	0 or 1	1/8 of a byte
B	byte	8 bits	1 byte
KB	kilobyte	1,000 bytes	1,000 bytes
MB	megabyte	$1,000^2$ bytes	1,000,000 bytes
GB	gigabyte	$1,000^3$ bytes	1,000,000,000 bytes
TB	terabyte	$1,000^4$ bytes	1,000,000,000,000 bytes
PB	petabyte	$1,000^5$ bytes	1,000,000,000,000,000 bytes
EB	exabyte	$1,000^6$ bytes	1,000,000,000,000,000,000 bytes
ZB	zettabyte	$1,000^7$ bytes	1,000,000,000,000,000,000,000 bytes
YB	yottabyte	$1,000^8$ bytes	1,000,000,000,000,000,000,000,000 bytes

*A lowercase "b" is used as an abbreviation for bits, while an uppercase "B" represents bytes.

463EB

of data will be created every day by 2025

IDC

500m

tweets are sent every day

Twitter

4PB

of data created by Facebook, including

350m photos

100m hours of video watch time

Facebook Research

95m

photos and videos are shared on Instagram

Instagram Business

320bn

emails to be sent each day by 2021

306bn

emails to be sent each day by 2020

294bn

billion emails are sent

Radicati Group

65bn

messages sent over WhatsApp and two billion minutes of voice and video calls made

Facebook

3.9bn

people use emails

4TB

of data produced by a connected car

Intel

Searches made a day → 5bn

Searches made a day from Google → 3.5bn

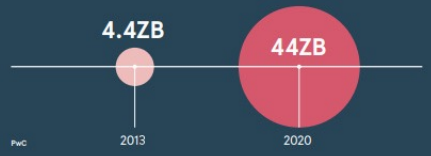
Smart Insights

28PB

to be generated from wearable devices by 2020

Statista

ACCUMULATED DIGITAL UNIVERSE OF DATA

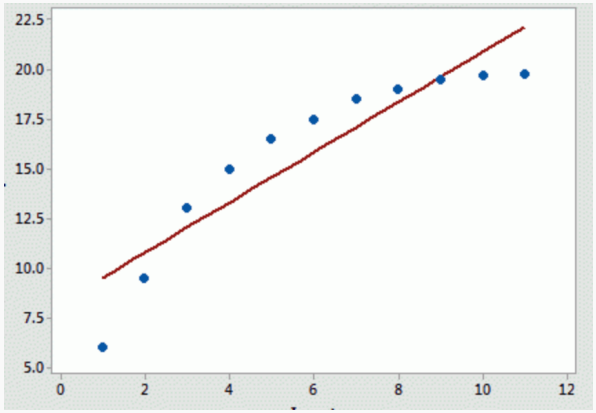


PwC

RACONTEUR

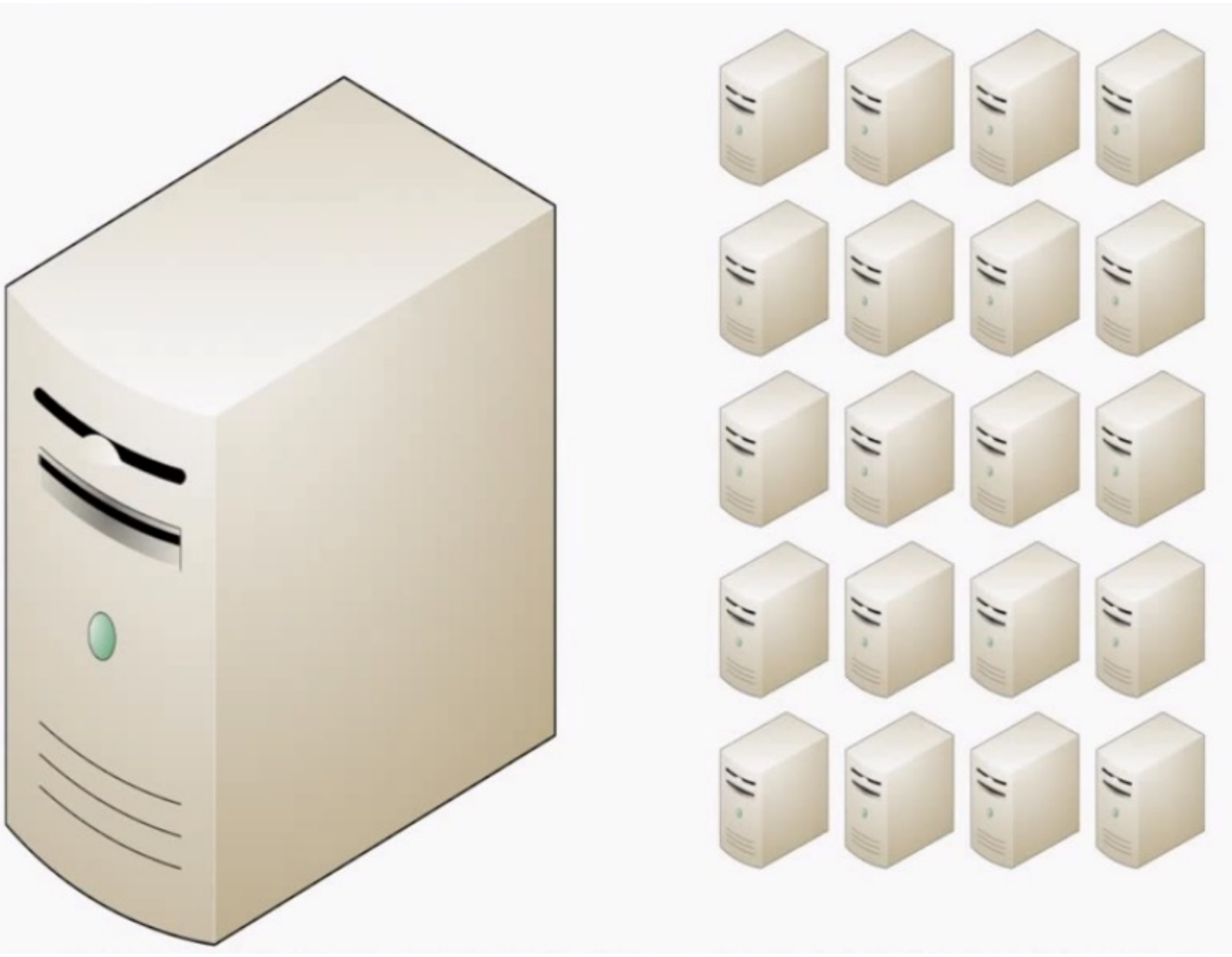

BIG DATA

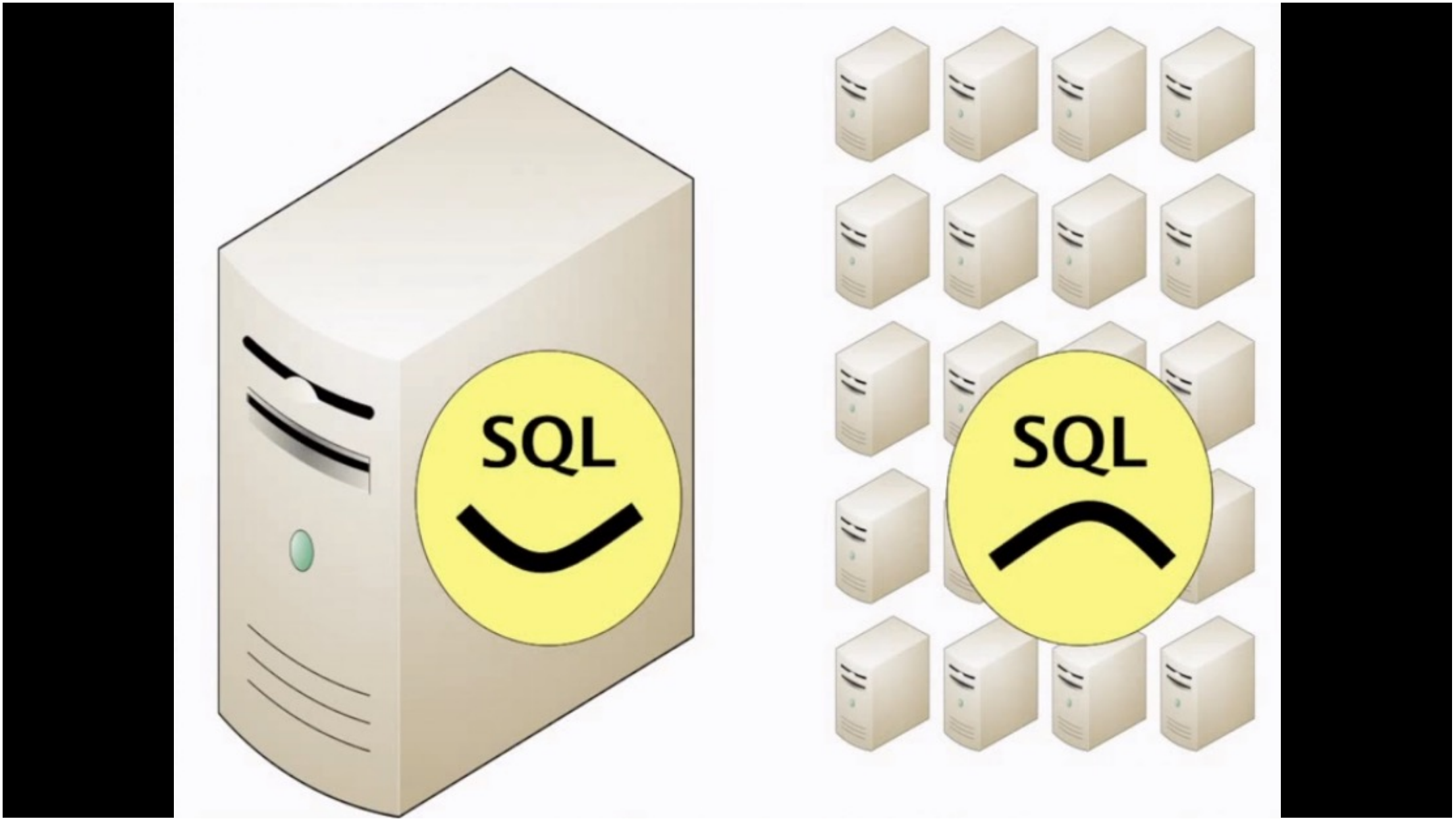

Lots of
Traffic


22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
0
2
4
6
8
10
12

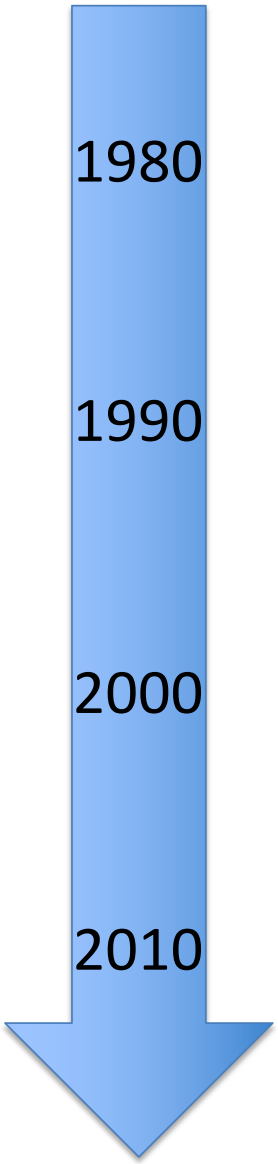

Google


amazon.com




SQL
SQL


Google
Bigtable
amazon.com
Dynamo



NoSQL

The Birth of NoSQL

Eventbrite Contact Last.fm for event and ticket information.

Email Share Tweet Like

This event has ended! View current events hosted by Last.fm

NOSQL meetup

Thursday, June 11, 2009 from 10:00 AM to 5:00 PM (PT)
San Francisco, CA

Ticket Information

TYPE	REMAINING	END		QUANTITY
Free ticket	**Sold Out**	Ended	Free	**Sold Out**

Share this! Share Tweet Like Be the first of your friends to like this.

Event Details

Introduction

This meetup is about "open source, distributed, non relational databases".

Have you run into limitations with traditional relational databases? Don't mind trading a query language for scalability? Or perhaps you just like shiny new things to try out? Either way this meetup is for you.

Join us in figuring out why these newfangled Dynamo clones and BigTables have become so popular lately. We have gathered presenters from the most interesting projects around to give us all an introduction to the field.

Preliminary schedule

When & Where

CBS interactive, Magma room
235 Second Street
San Francisco, CA 94105

Thursday, June 11, 2009 from 10:00 AM to 5:00 PM (PT)

Definition of NoSQL

Characteristics of NoSQL

non-relational

open-source

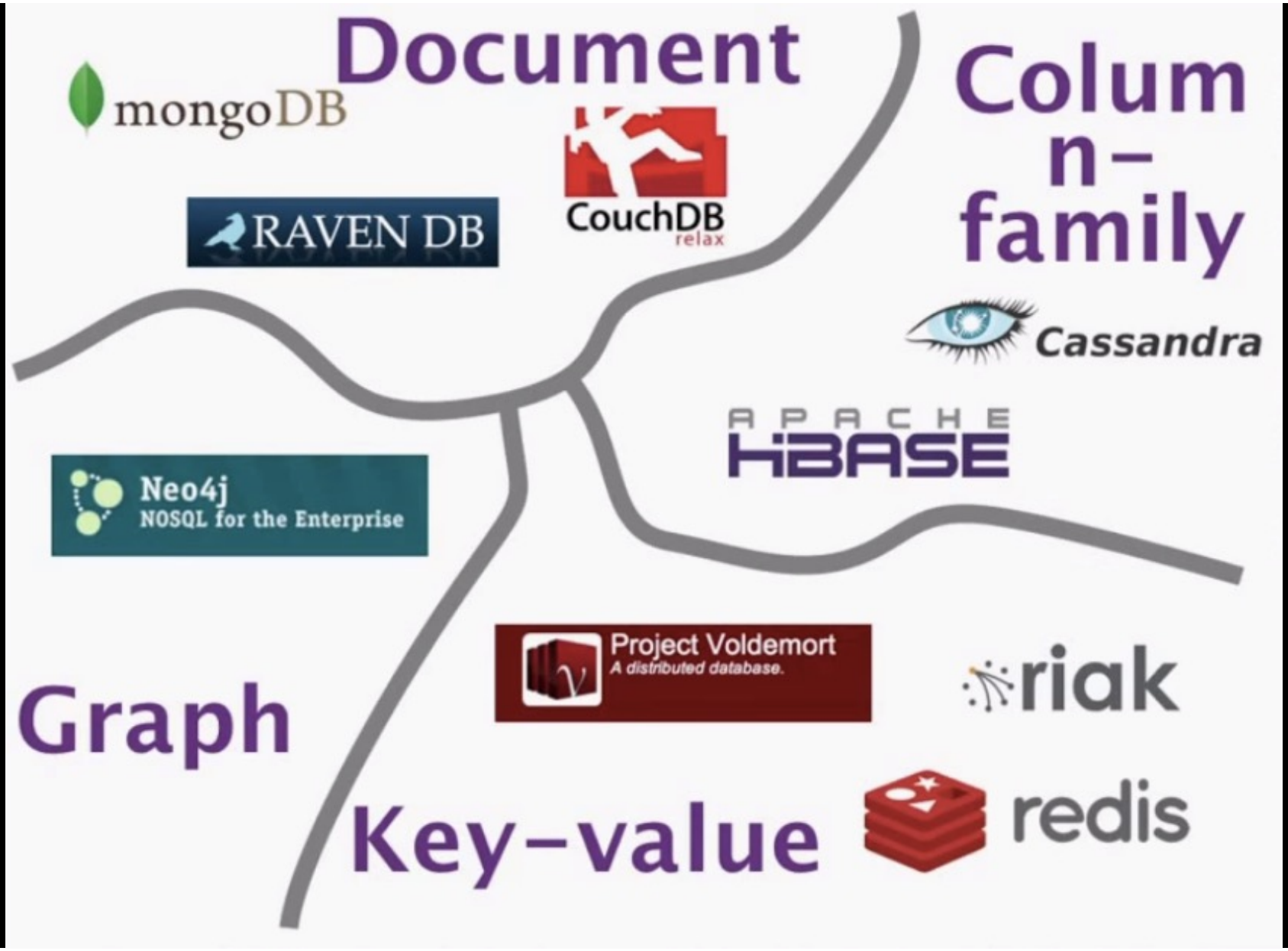
cluster-friendly

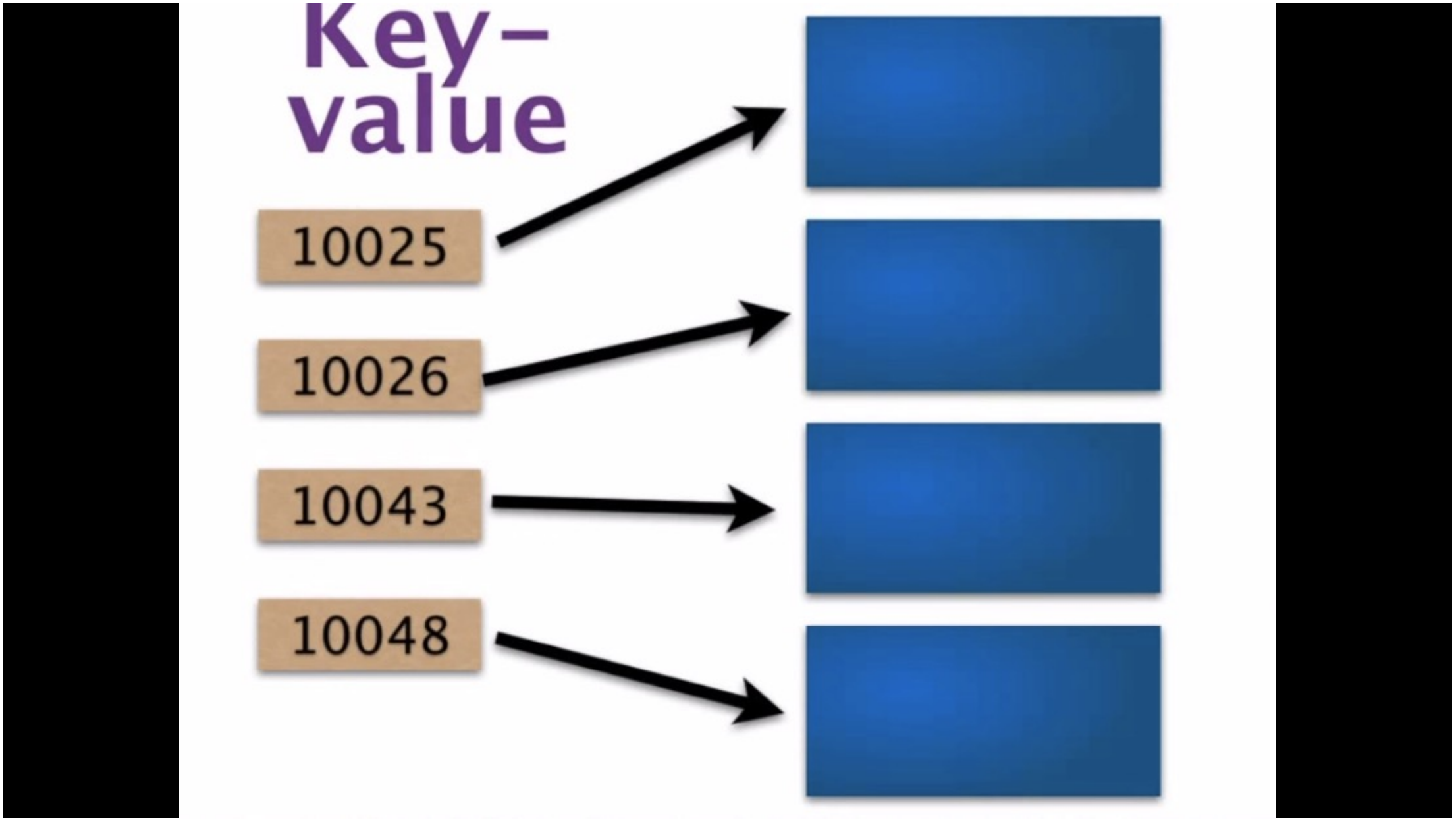
21st Century Web

schema-less

Data Model


mongoDB
RAVEN DB
CouchDB
relax
Cassandra
APACHE
HBASE
Neo4j
NOSQL for the Enterprise
Project Voldemort
A distributed database.
riak
redis

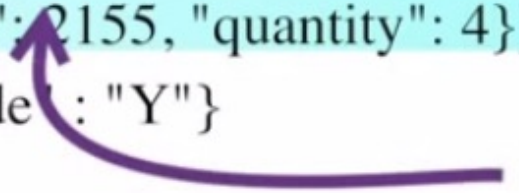

Document
mongoDB
RAVEN DB
CouchDB
relax
Colum
n-
family
Cassandra
APACHE
HBASE
Neo4j
NOSQL for the Enterprise
Project Voldemort
A distributed database.
riak
Graph
Key-value
redis


Key–
value
10025
10026
10043
10048

Document

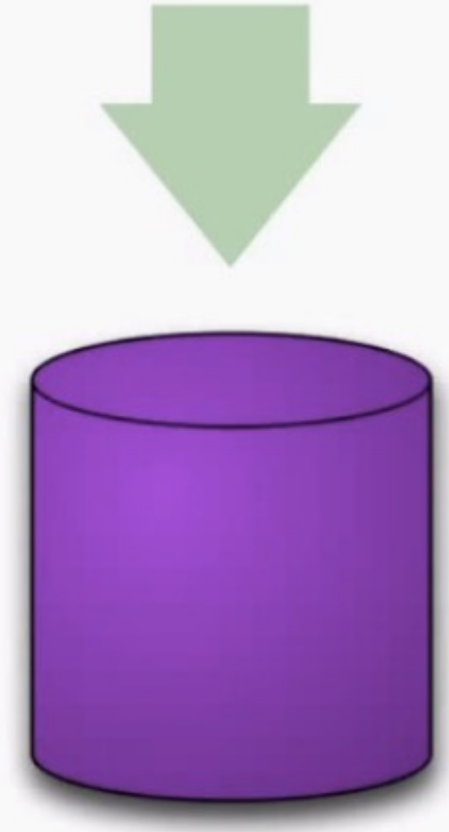

```
{"id": 1001,
"customer_id": 7231,
"line-itmes": [
{"product_id": 4555, "quantity": 8},
{"product_id": 7655, "quantity": 4}, {"product_id": 8755,
```

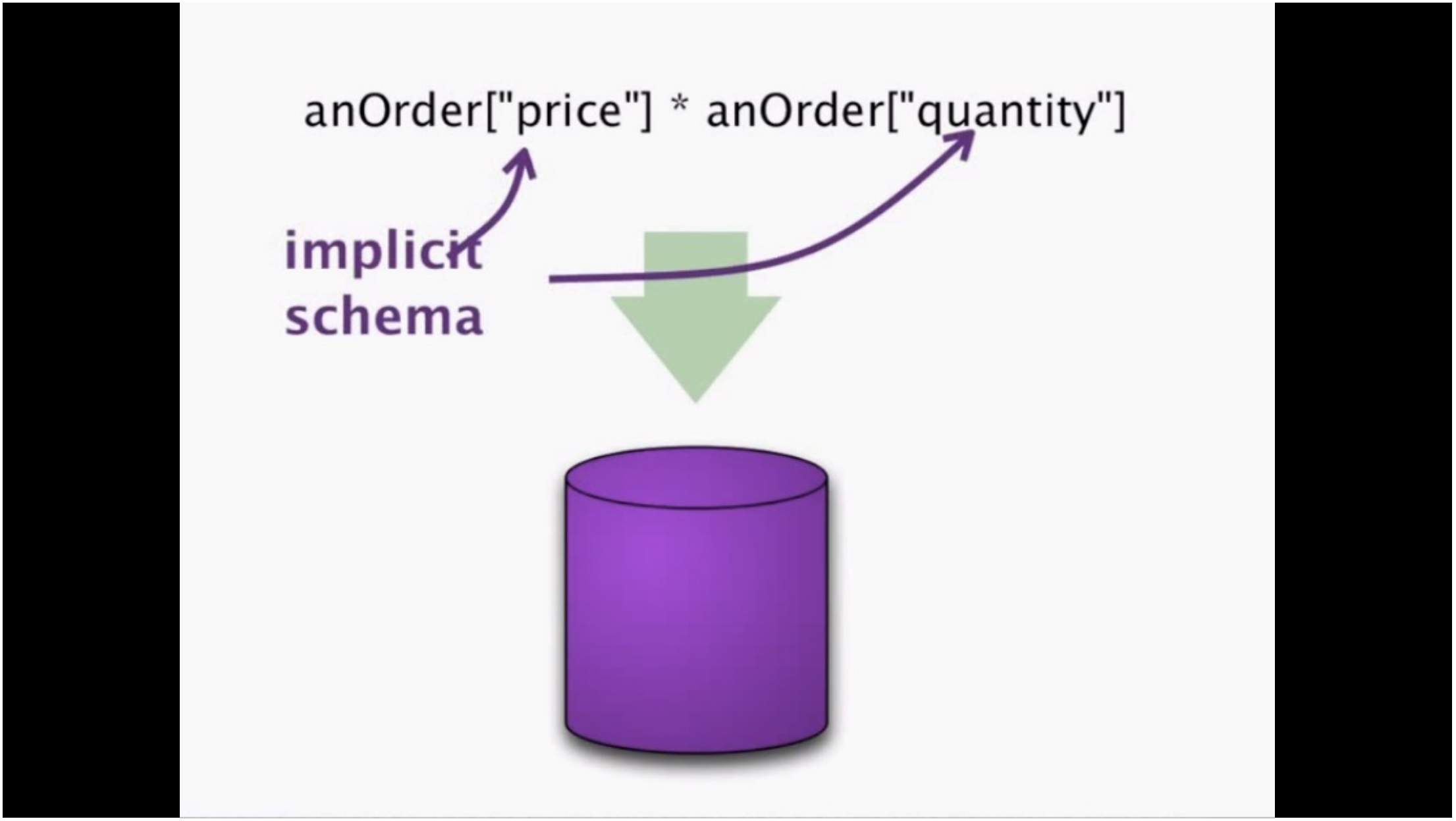
```
{"id": 1002,
"customer_id": 9831,
"line-itmes": [
{"product_id": 4555, "quantity": 3},
{"product_id": 8155, "quantity": 4}],
"discount-code": "Y"}
```

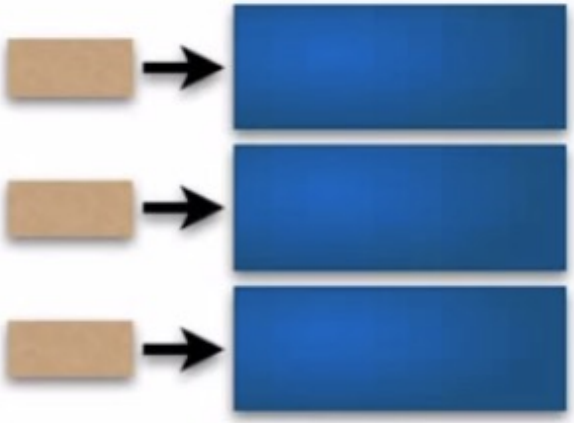
no schema

anOrder["price"] * anOrder["quantity"]




anOrder["price"] * anOrder["quantity"]
implicit
schema

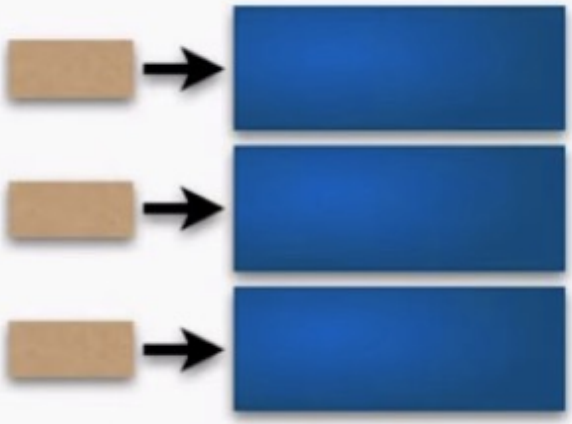
Key-Value



Document

```
{"id": 1001,
{"id": 1002,
   "customer_id": 7231,
   "line-itmes": [
      {"product_id": 4555, "quantity": 8},
      {"product_id": 7655, "quantity": 4},
}     {"product_id": 8755, "quantity": 3}]
}
```

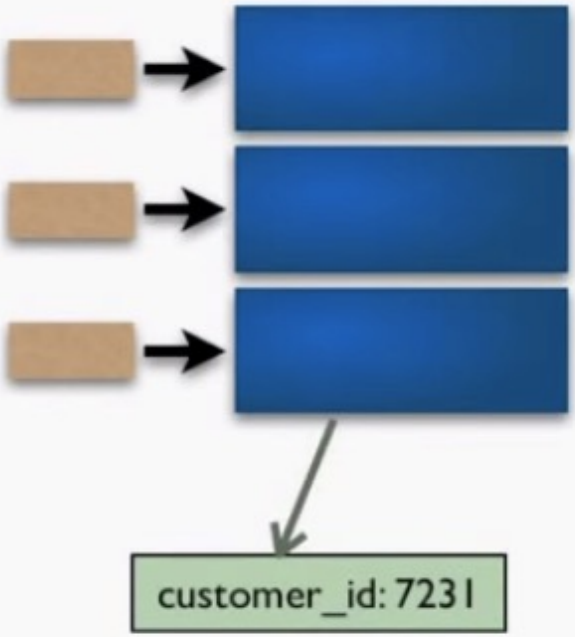
Key-Value



Document

```
{"id": 1001,
  {"id": 1002,
    "customer_id": 7231,
    "line-itmes": [
      {"product_id": 4555, "quantity": 8},
      {"product_id": 7655, "quantity": 4},
}     {"product_id": 8755, "quantity": 3}]
  }
```

Key-Value

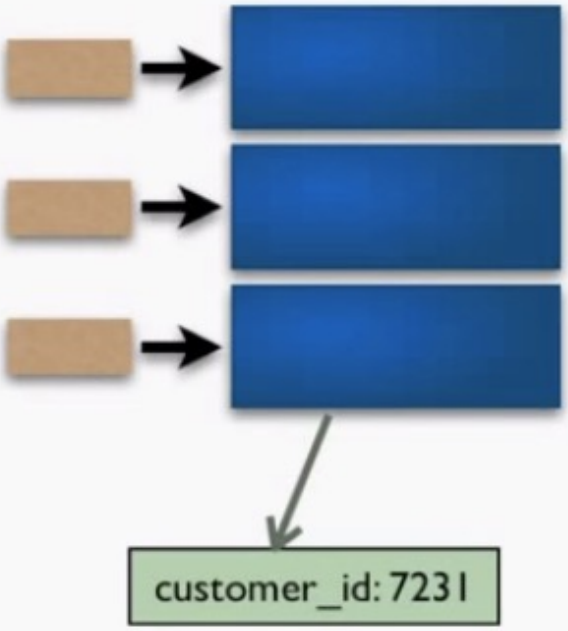


metadata

Document

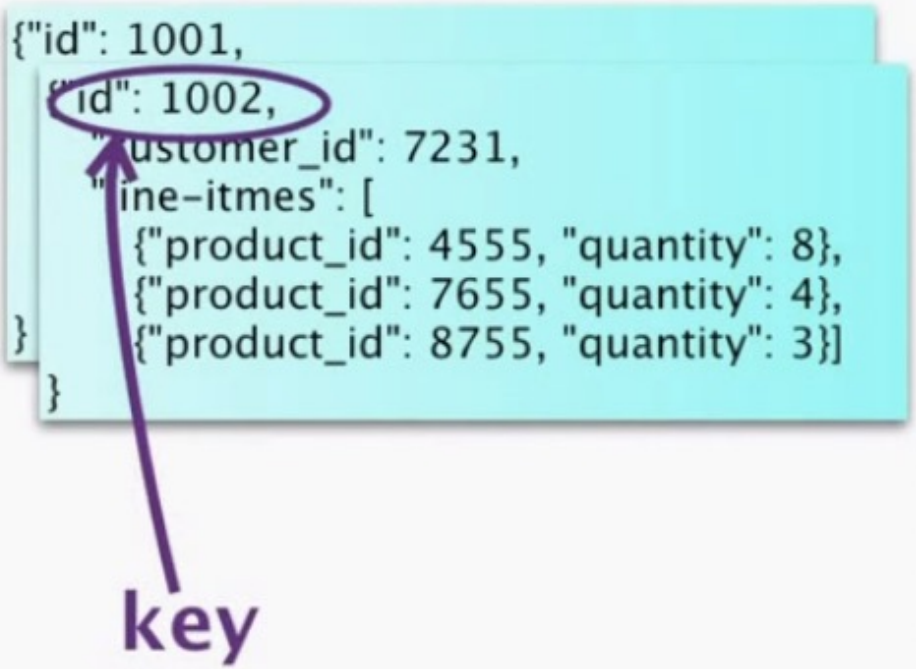
```
{"id": 1001,
  {"id": 1002,
     "customer_id": 7231,
     "line-itmes": [
        {"product_id": 4555, "quantity": 8},
        {"product_id": 7655, "quantity": 4},
}       {"product_id": 8755, "quantity": 3}]
  }
```

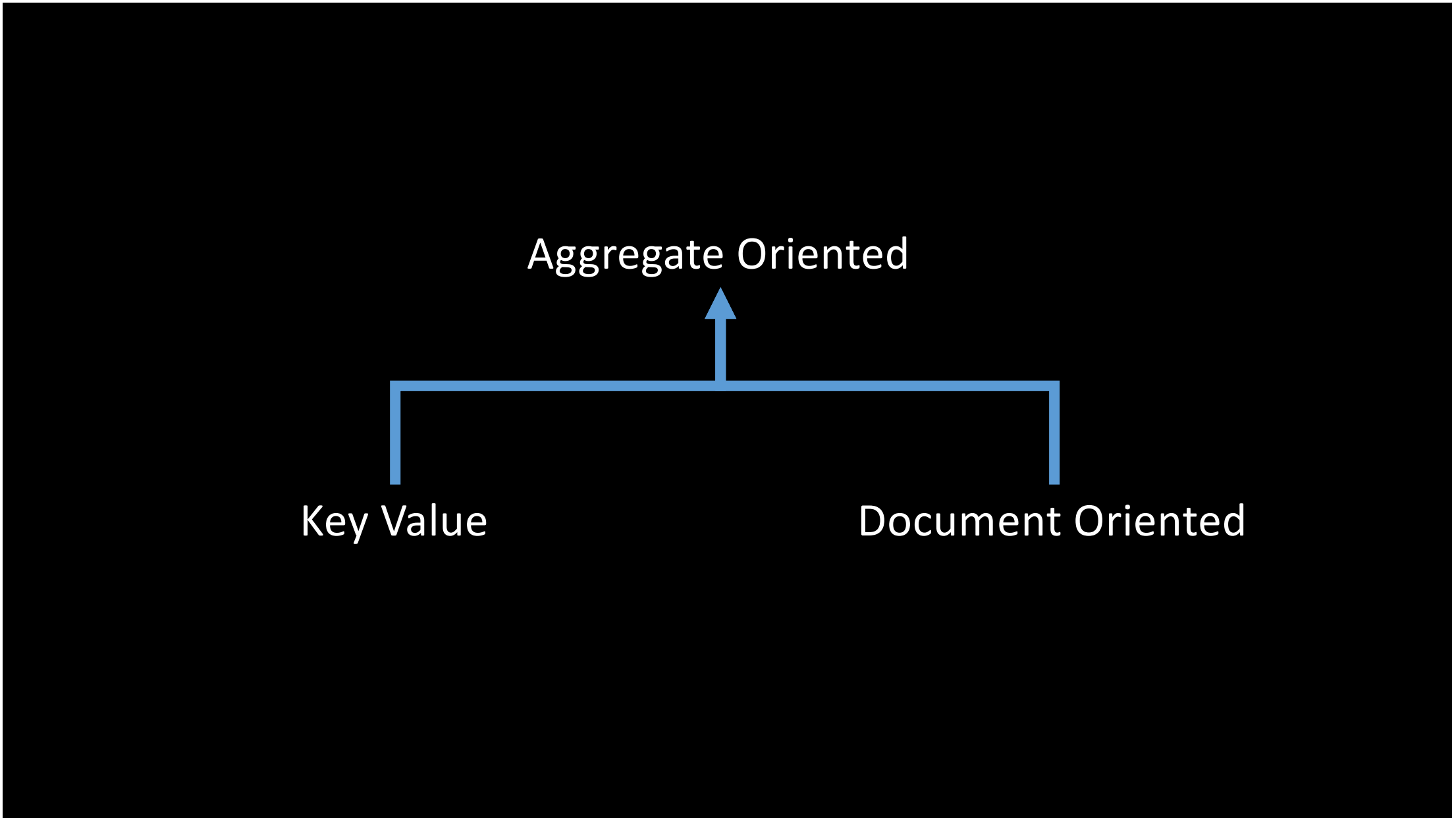
Key-Value

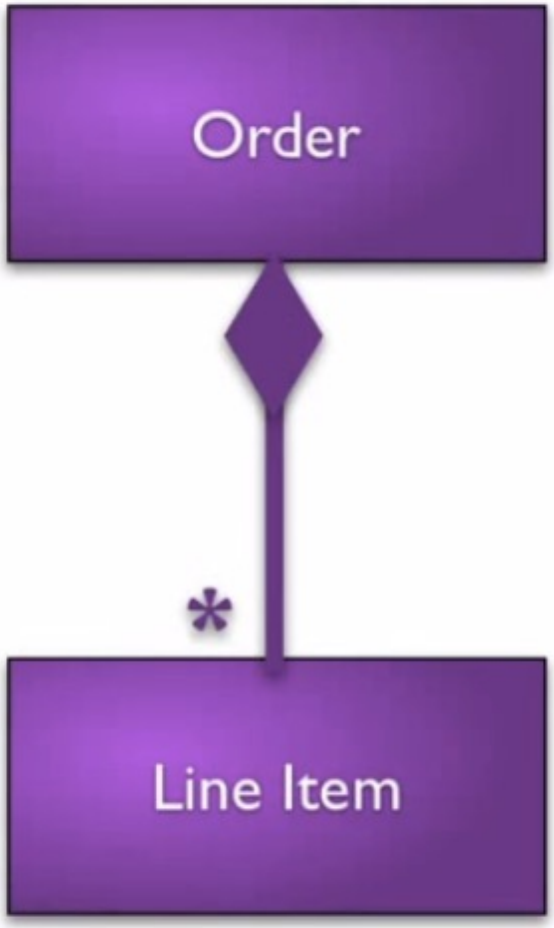


metadata

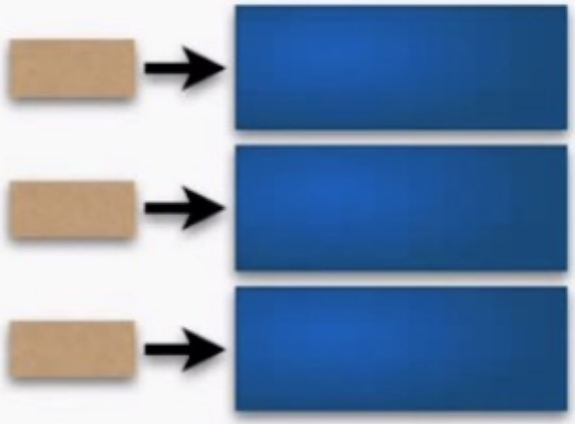
Document




Aggregate Oriented
Key Value
Document Oriented


Order
*
Line Item

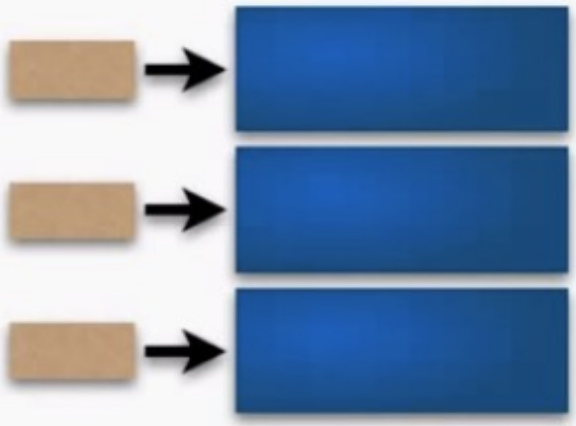
Key-Value



Document

```
{"id": 1001,
 {"id": 1001,
    "customer_id": 7231,
    "line-itmes": [
       {"product_id": 4555, "quantity": 8},
       {"product_id": 7655, "quantity": 4},
       {"product_id": 8755, "quantity": 3}]}
```

Key-Value

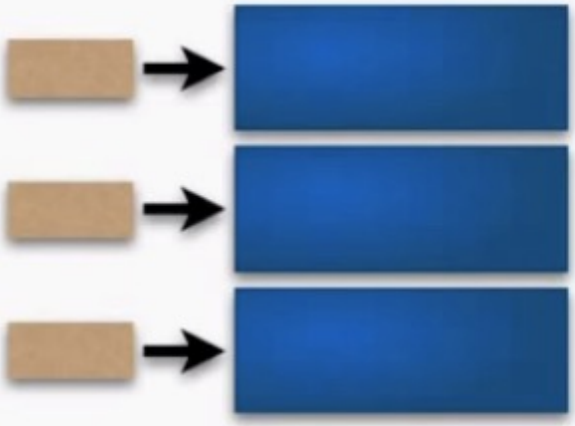


Document

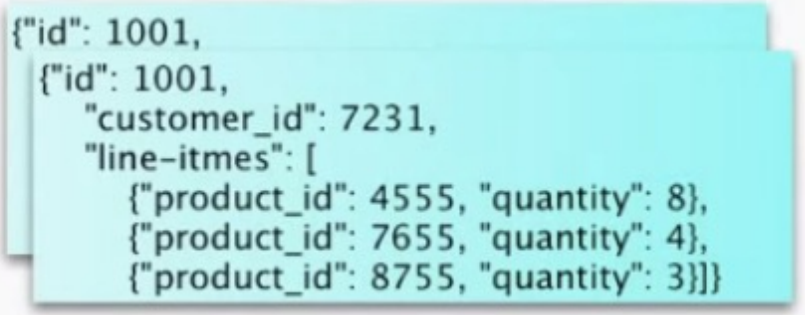
```
{"id": 1001,
 {"id": 1001,
    "customer_id": 7231,
    "line-itmes": [
       {"product_id": 4555, "quantity": 8},
       {"product_id": 7655, "quantity": 4},
       {"product_id": 8755, "quantity": 3}]}
```

Value ==
Aggregate

Key-Value



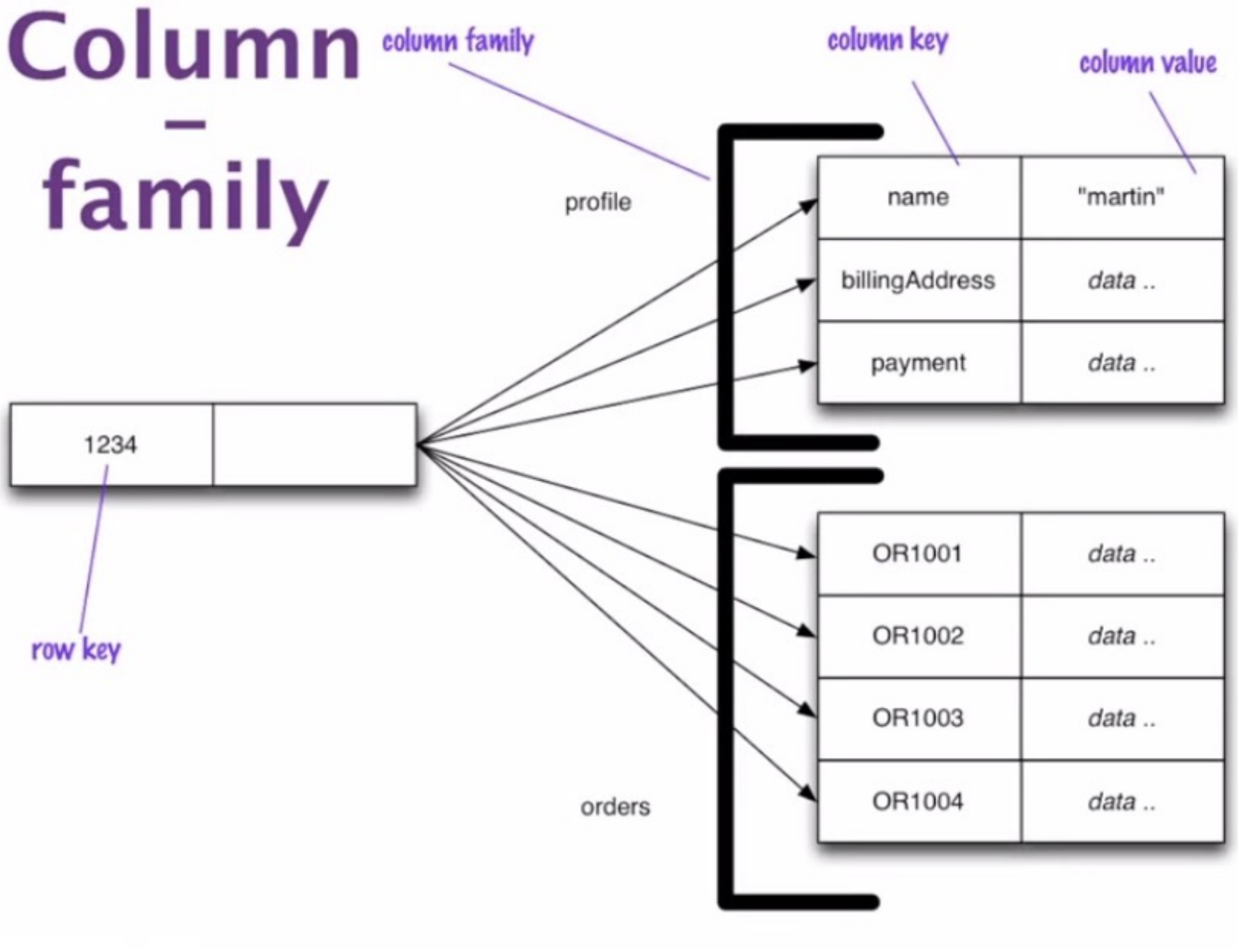
Document

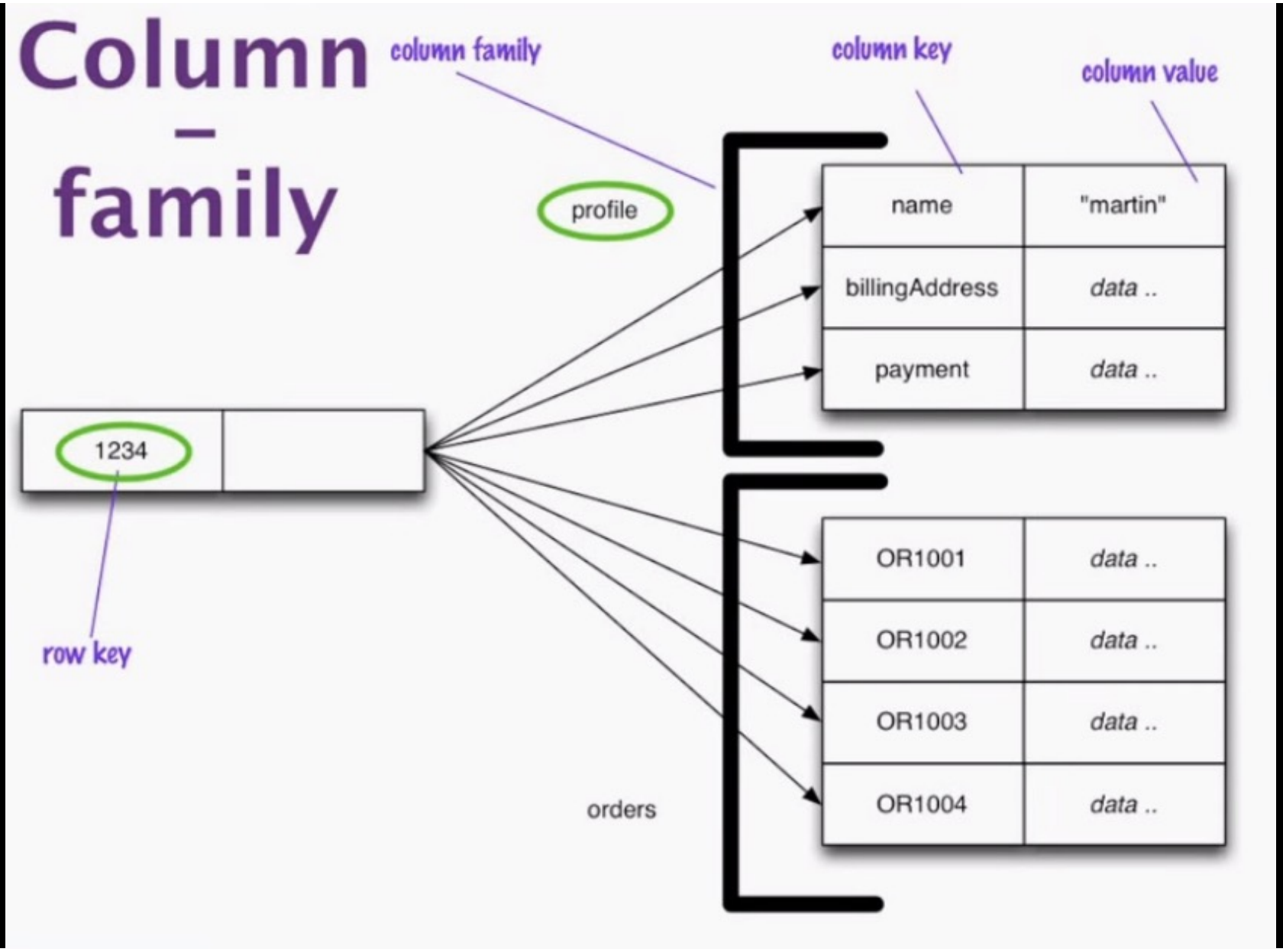


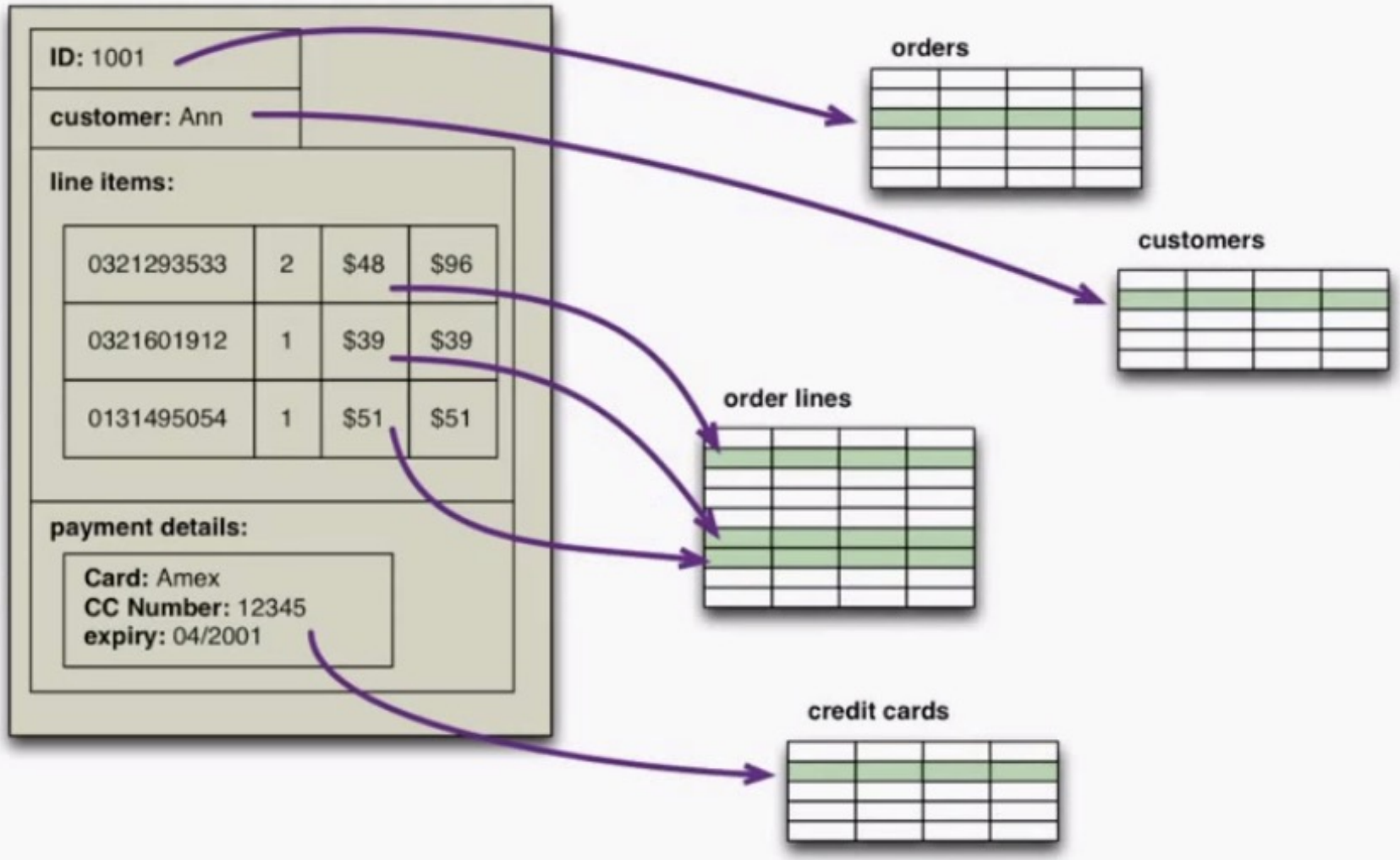
Value ==
Aggregate

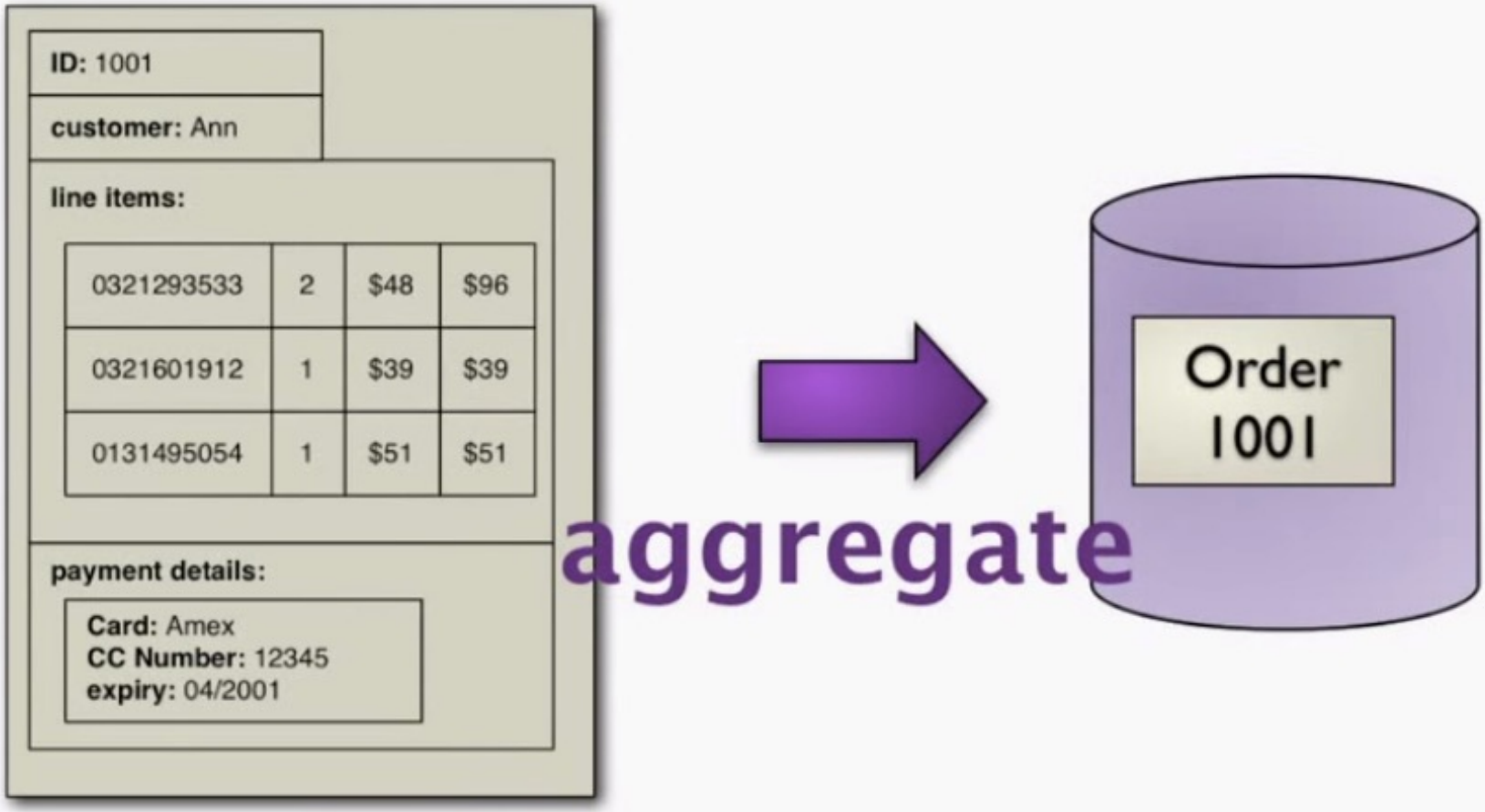
Document ==
Aggregate

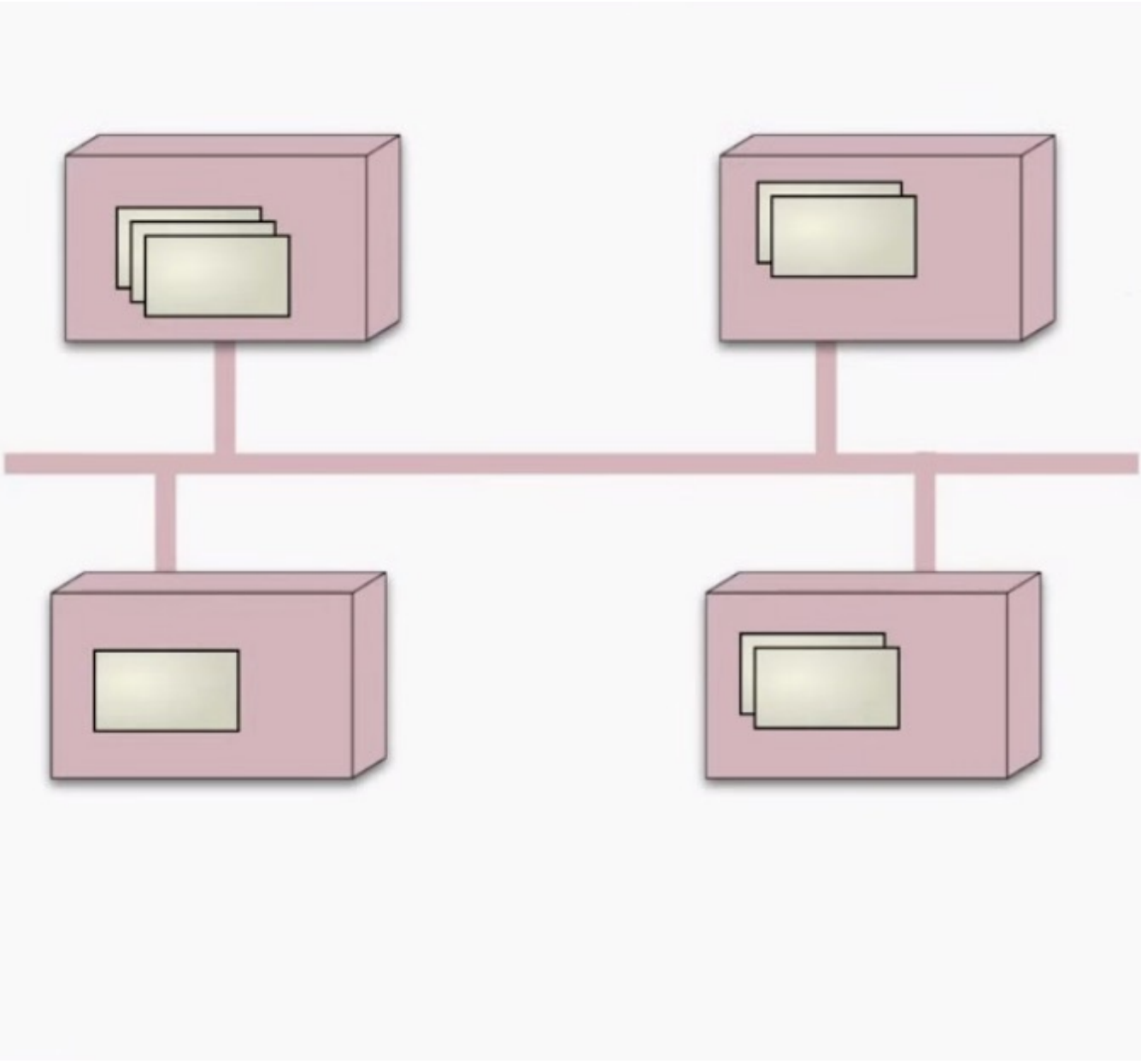
Column family



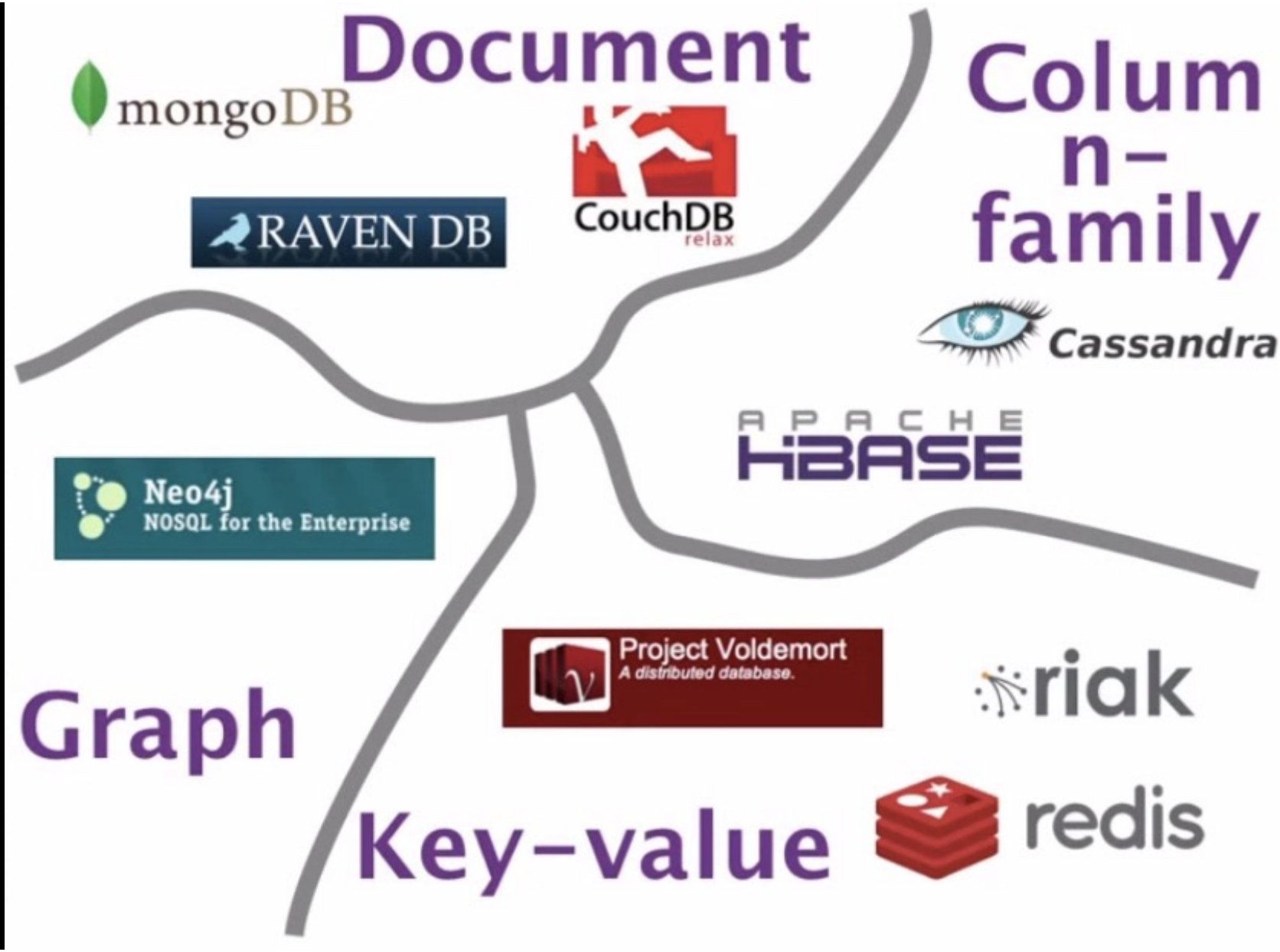

Column
–
family
column family
column key
column value
profile
name
"martin"
billingAddress
data ..
payment
data ..
1234
row key
OR1001
data ..
OR1002
data ..
OR1003
data ..
OR1004
data ..
orders

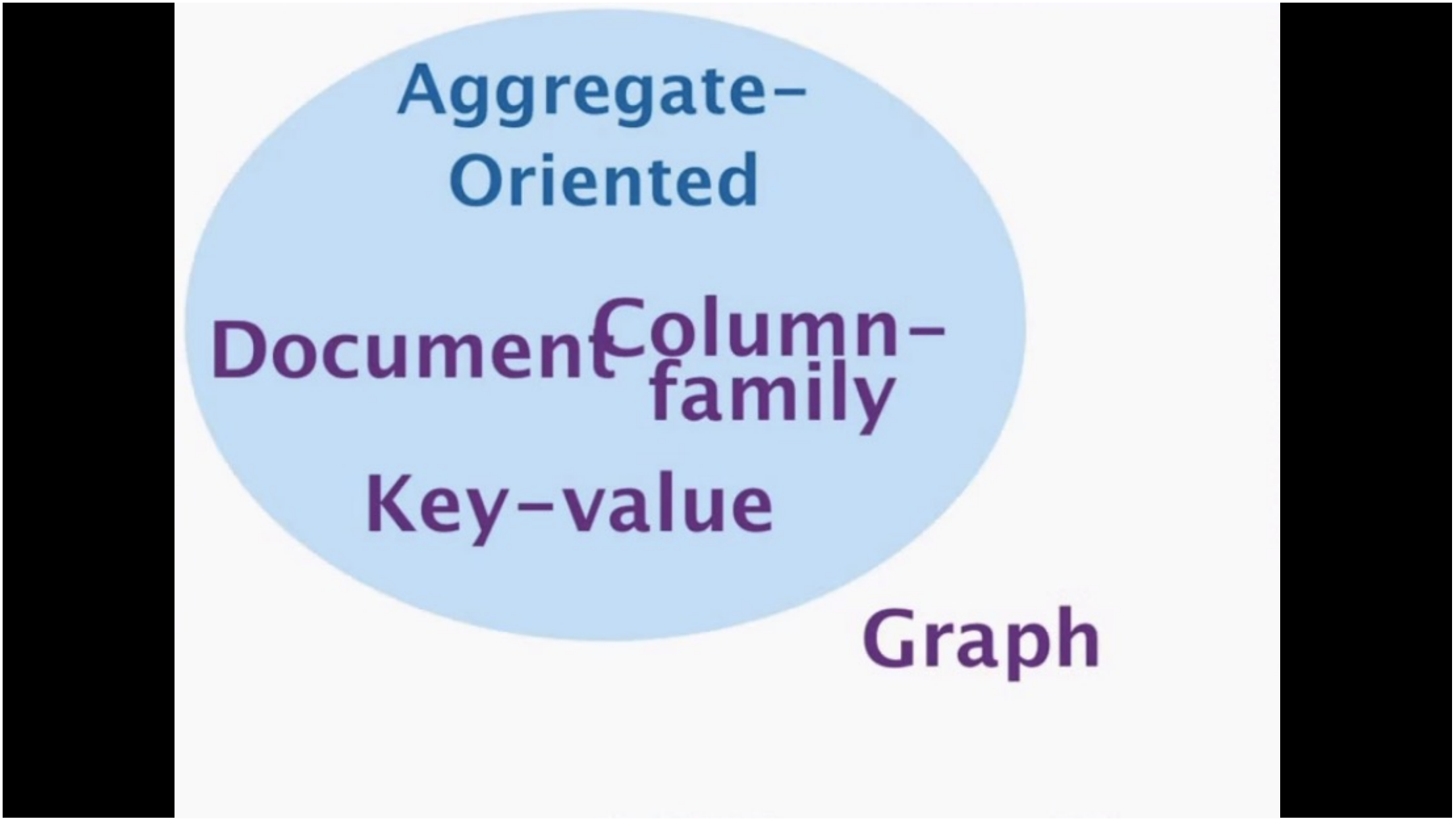

ID: 1001
customer: Ann
line items:
0321293533
2
$48
$96
0321601912
1
$39
$39
0131495054
1
$51
$51
payment details:
Card: Amex
CC Number: 12345
expiry: 04/2001
orders
customers
order lines
credit cards


ID: 1001
customer: Ann
line items:
0321293533
2
$48
$96
0321601912
1
$39
$39
0131495054
1
$51
$51
payment details:
Card: Amex
CC Number: 12345
expiry: 04/2001
aggregate
Order
1001



Product	revenue	prior revenue
321293533	3083	7043
321601912	5032	4782
131495054	2198	3187
...	...	...
...	...	...
...	...	...
...	...	...
...	...	...
...	...	...


Document
mongoDB
RAVEN DB
CouchDB
relax
Colum
n-
family
Cassandra
APACHE
HBASE
Neo4j
NOSQL for the Enterprise
Project Voldemort
A distributed database.
riak
Graph
Key-value
redis

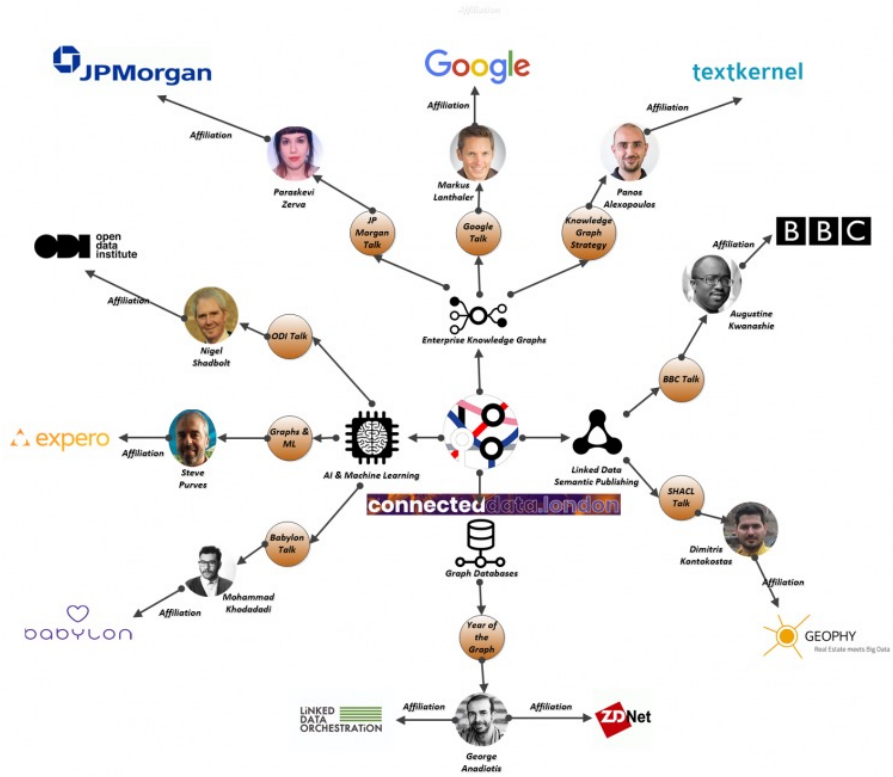

Aggregate-
Oriented
Document
Column-
family
Key-value
Graph

Humans think in graphs

- We understand better things by matching them to things that we know.
- And the things we know … are in the form of graphs

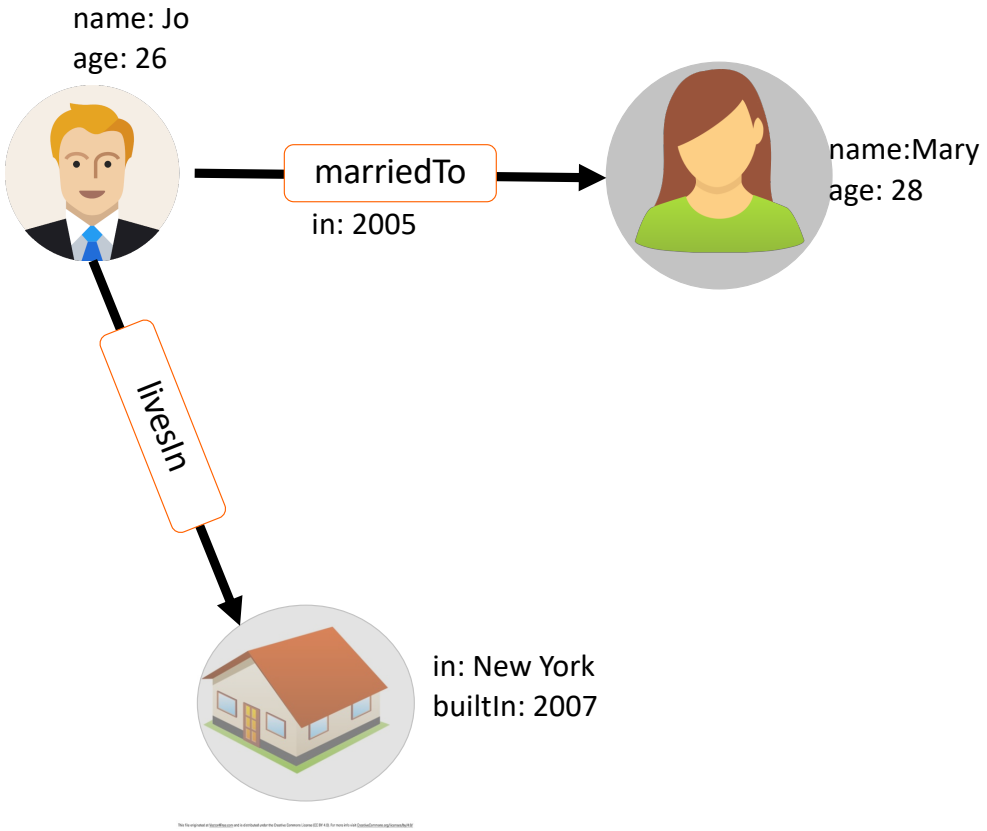
Graphs capture naturally heterogeneous data

- No specific schema
- Shows how different parts connect to each other.
- Hard to model with relational joins
- Very important for more informative insights



The property Graph Model

- Nodes
  - Represent Oblects
  - Can be labeled
- Relationships
  - Relate Nodes
  - Have a label (used as a type)
  - Have direction
- Properties
  - <Name: value> pairs
  - Describe characteristics
  - Can be on nodes or on edges



Graph Data: Social Networks

Facebook social graph

4-degrees of separation [Backstrom-Boldi-Rosa-Ugander-Vigna, 2011]

Gremlin

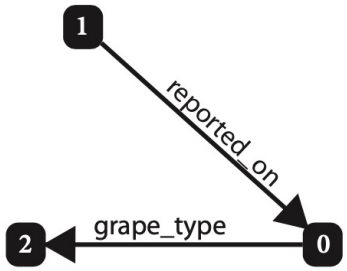
- Java, Rest, Cypher, Gremlin
- Gremplin: Graph traversal language
- g: variable that represents a graph
- Profiles

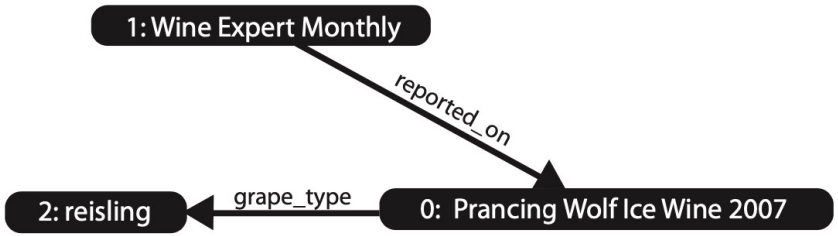
```
gremlin> g.V
==>v[0]
==>v[1]
==>v[2]

gremlin> g.E
==> e[0][1-reported_on->0]
==> e[1][0-grape_type->2]

gremlin> g.v(0)
==> v[0]

List properties:
gremlin> g.v(0).map()
==> name=Prancing Wolf Ice Wine 2007
```

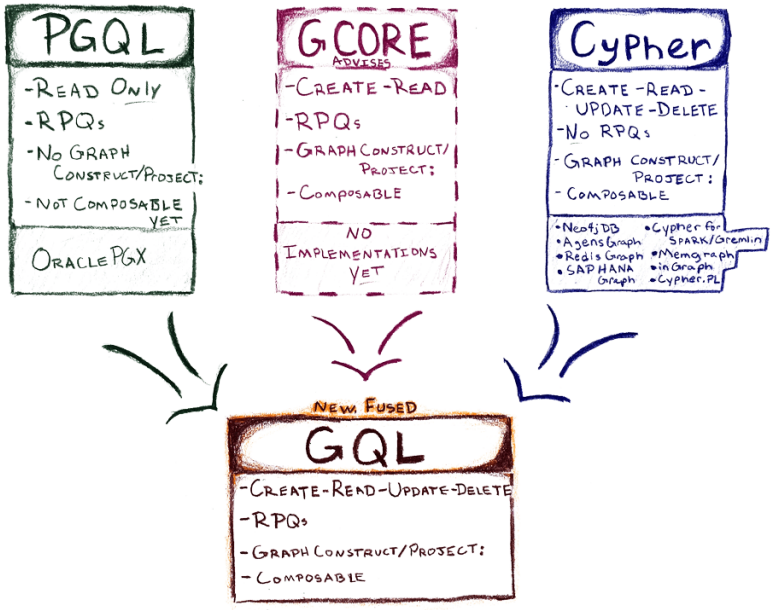




The Property Graph Query Language

- Why not SQL
  - Cannot express graph patterns
  - Not easy to do recursions
  - Go easier with heterogeneous structures
  - Conceptually easier to understand

```
...
MATCH (a)
CALL {
  MATCH (a)-[:FRIEND_OF]-(x)-[:FRIEND_OF]-(b)
  WHERE a <> x AND x <> b AND a <> b
  RETURN count(DISTINCT b) AS num_foafs
}
RETURN a.name, num_foafs
 ORDER BY num_foafs DESC
 LIMIT 100
```



Cypher

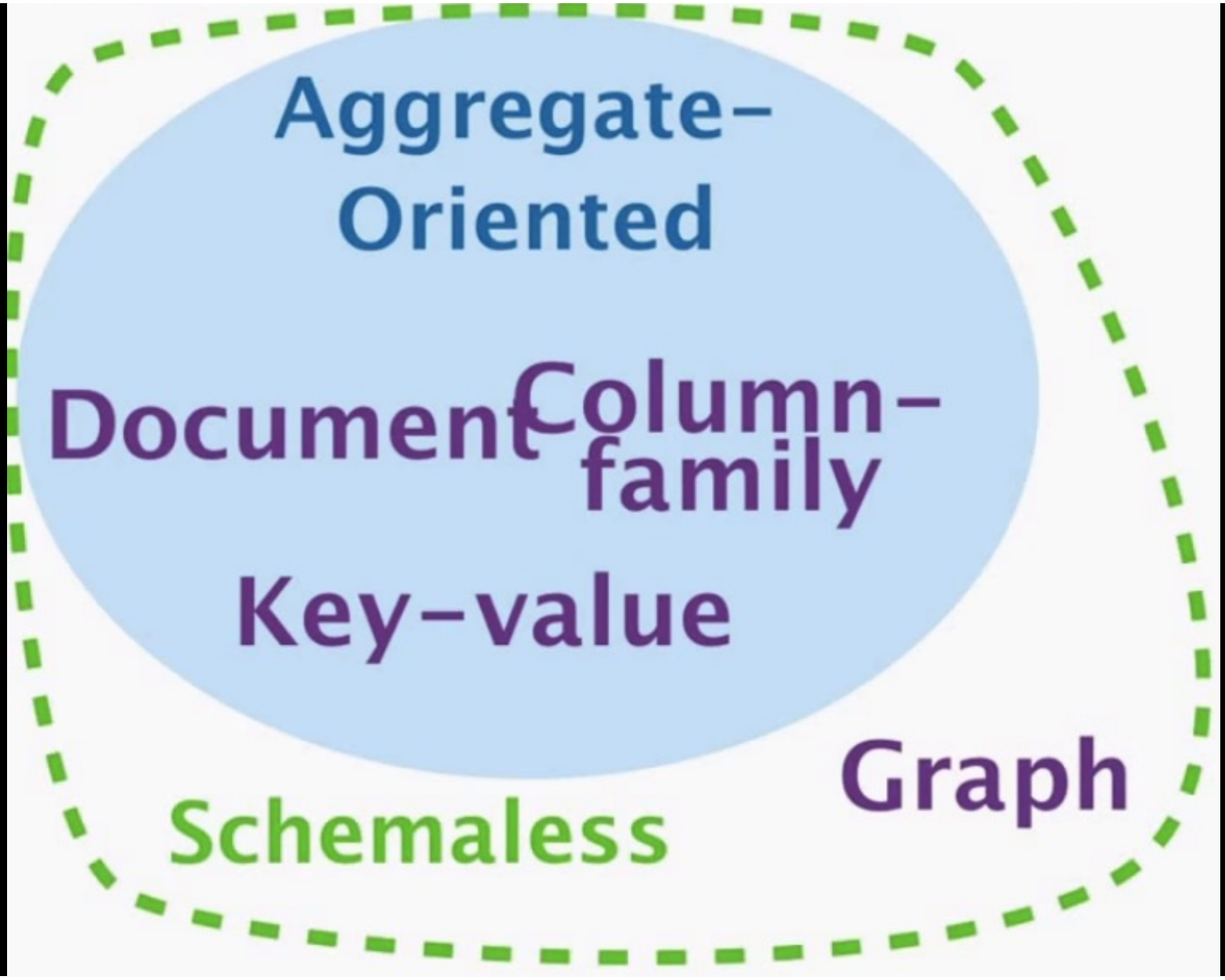
What Breweries Near Me That My Friends Also Like Have The Highest Rating?


```
1 MATCH (c:Category)<-[:IN_CATEGORY]-(brew:Business)
2 MATCH (brew)<-[:REVIEWS]-(r:Review)<-[:WROTE]-(u:User)-[:FRIENDS]-(me:User {user_id: 'qZB8BZ2ZzMF0VfIGhttu7w'})
3 WHERE c.name CONTAINS 'Breweries' AND brew.city = "Phoenix" AND r.stars > 4
4 MATCH (brew)<-[:REVIEWS]-(r2:Review)
5 RETURN brew.name, brew.address, avg(r2.stars) AS average_reviews ORDER BY average_reviews DESC
```

- `MATCH ...`
- `(node)-[:RELATIONSHIP]->(node)`
- Predicates
- Aggregations
- Ordering
- Graph vs tabular data

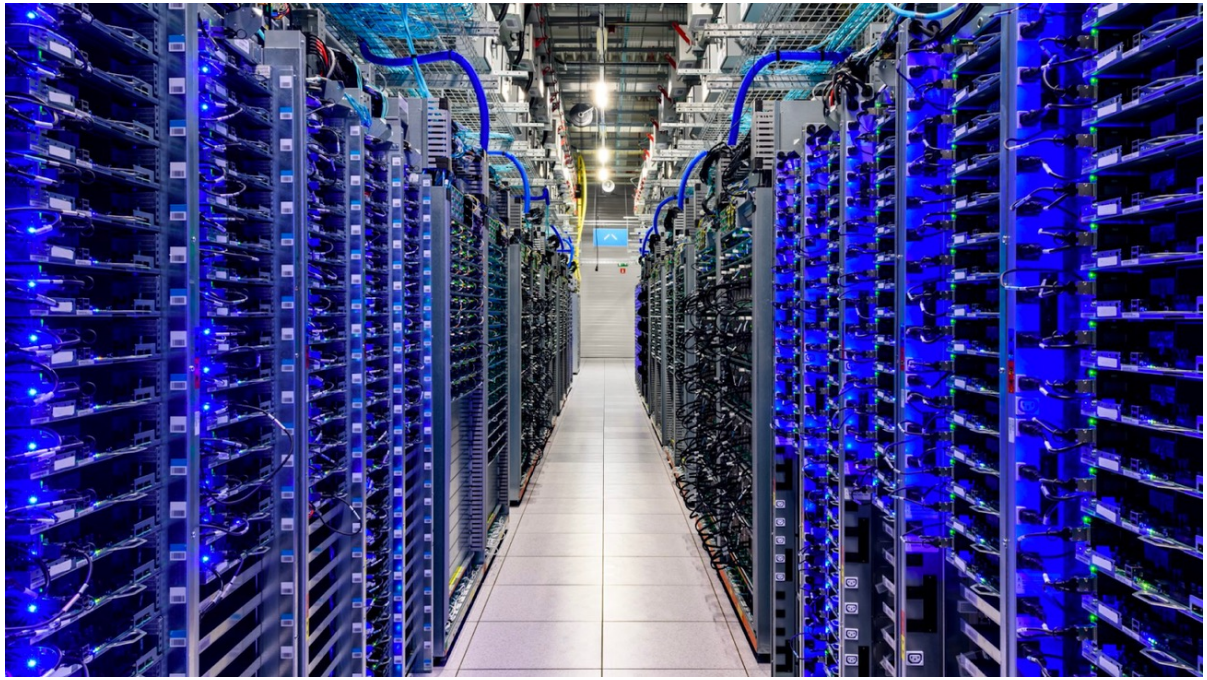
Graph Management Systems

Visit https://people.cs.aau.dk/~matteo/gdb.html for a comparison


Aggregate-
Oriented
Document
Column-
family
Key-value
Graph
Schemaless

A file system for Clusters

A Data Center



A Container style Data Center

A google data center may contain
45 containers which is ~60,000 servers

The Need for Concistency … or Not

ACID

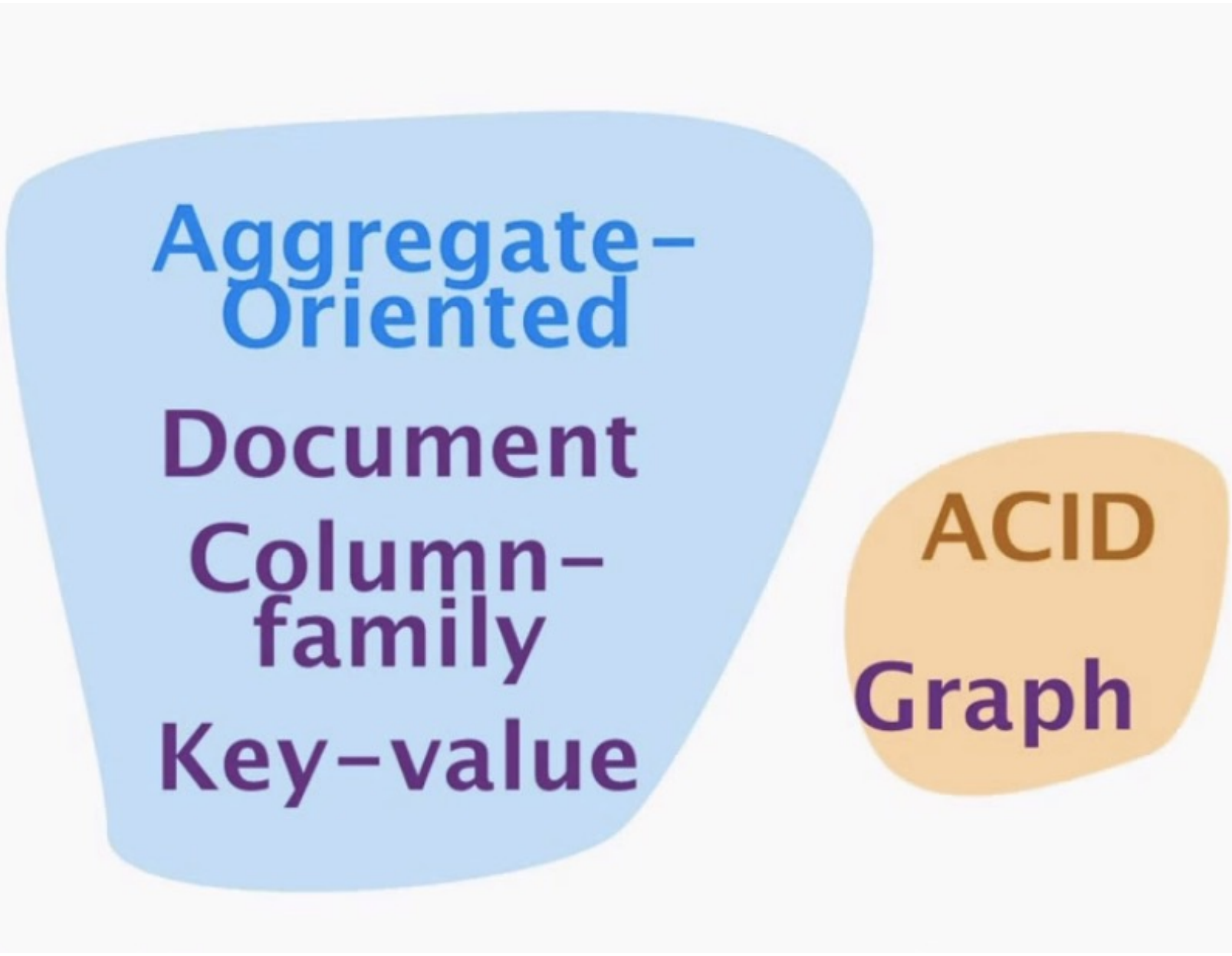
- ACID
  - **Atomic** – Each transaction is either properly carried out or the process halts and the database reverts back to the state before the transaction started. This ensures that all data in the database is valid.
  - **Consistent** – A processed transaction will never endanger the structural integrity of the database.
  - **Isolated** – Transactions cannot compromise the integrity of other transactions by interacting with them while they are still in progress.
  - **Durable** – The data related to the completed transaction will persist even in the cases of network or power outages. If a transaction fails, it will not impact the manipulated data.

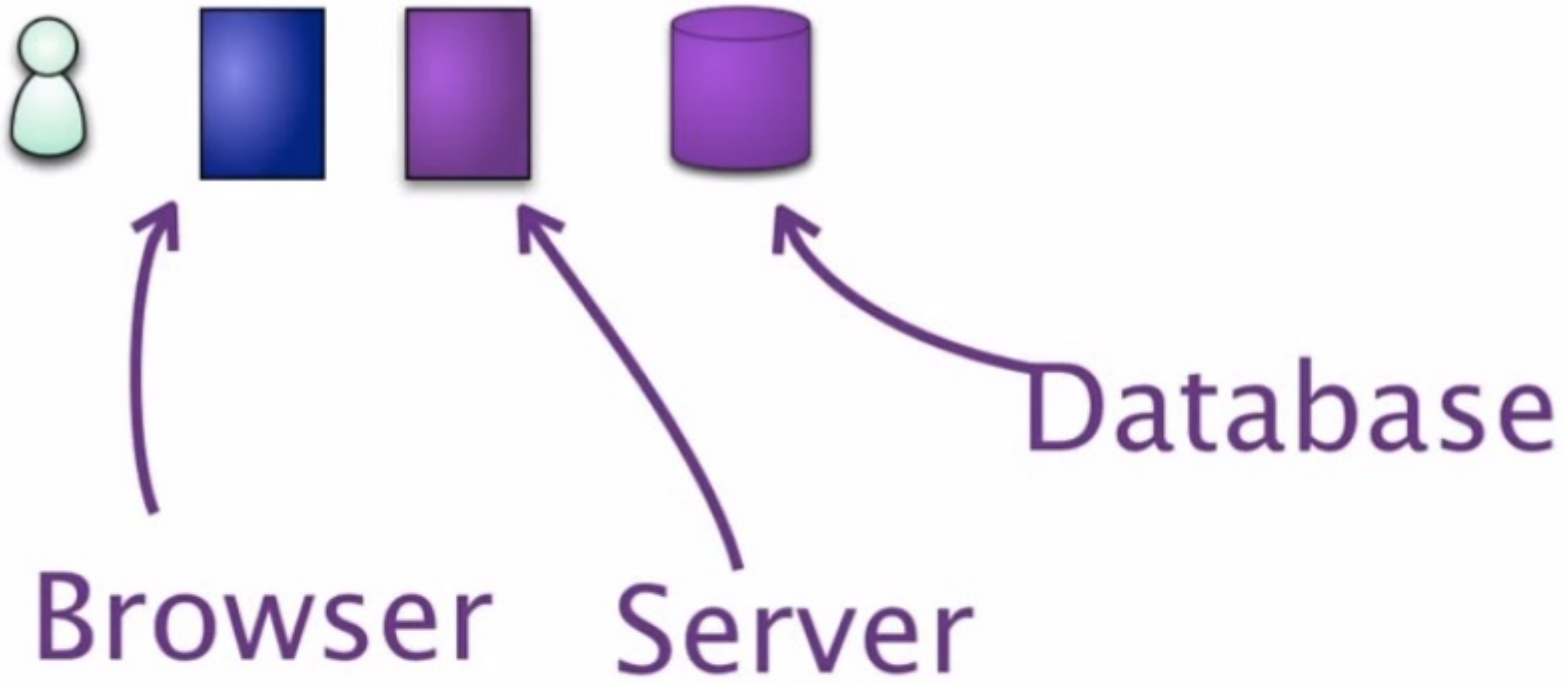
BASE

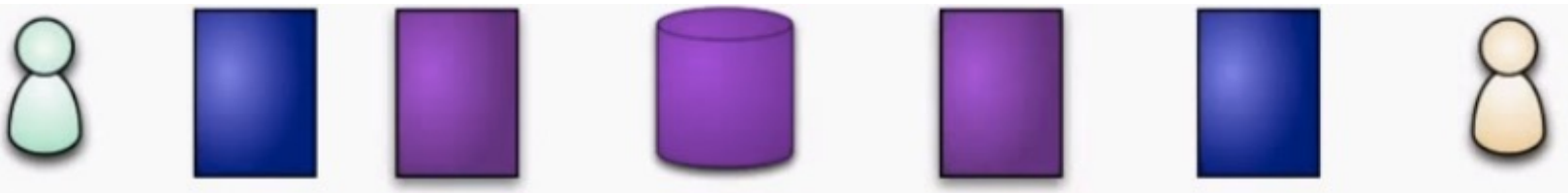
- Base
  - **Basically Available** – Rather than enforcing immediate consistency, BASE-modelled NoSQL databases will ensure availability of data by spreading and replicating it across the nodes of the database cluster.
  - **Soft State** – Due to the lack of immediate consistency, data values may change over time. The BASE model breaks off with the concept of a database which enforces its own consistency, delegating that responsibility to developers.
  - **Eventually Consistent** – The fact that BASE does not enforce immediate consistency does not mean that it never achieves it. However, until it does, data reads are still possible (even though they might not reflect the reality).

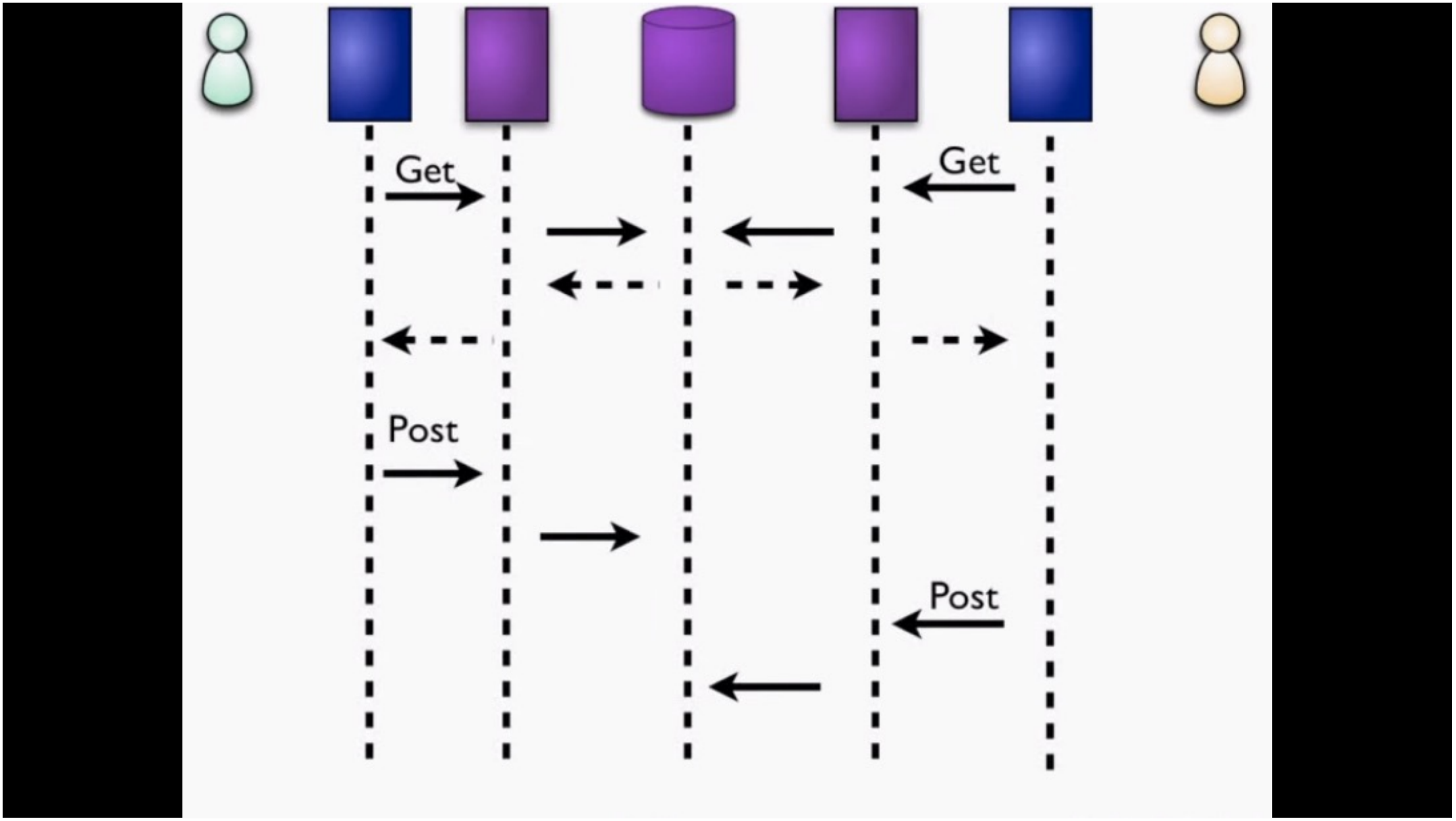
BASE

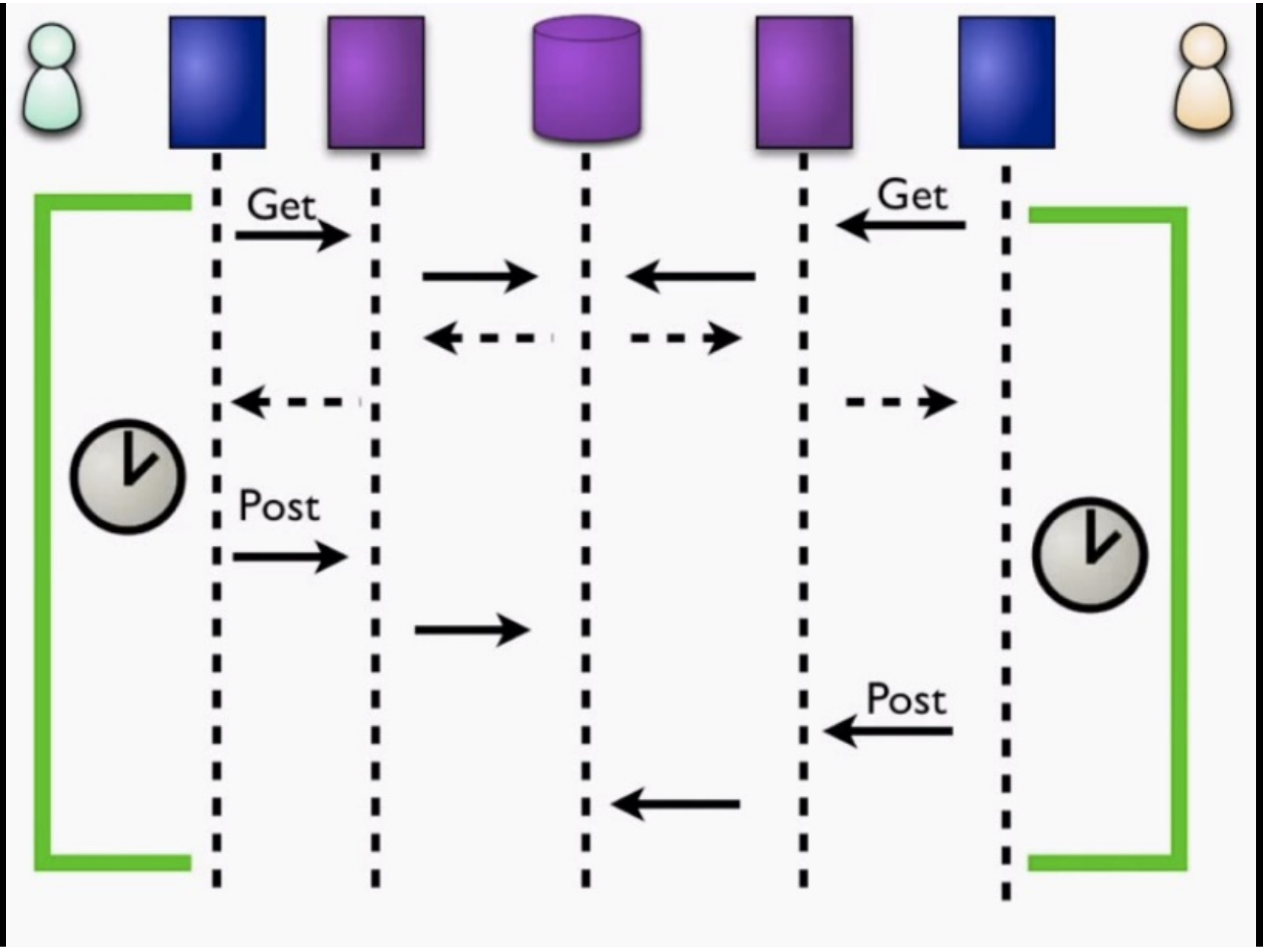
- RDBMS = ACID
- NoSQL = BASE (Well …. almost)

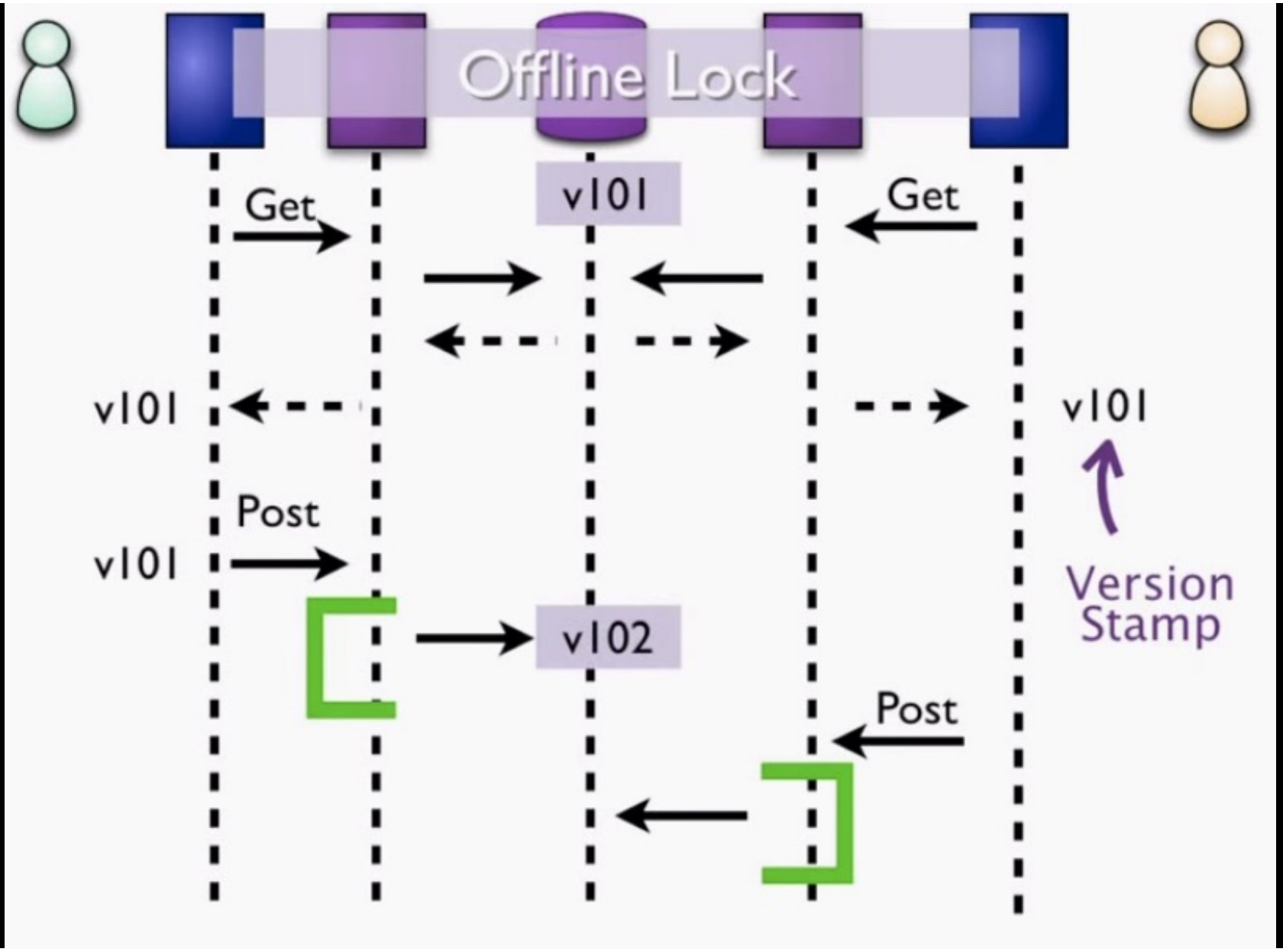

Aggregate-Oriented
Document
Column-family
Key-value
ACID
Graph


Database
Browser
Server



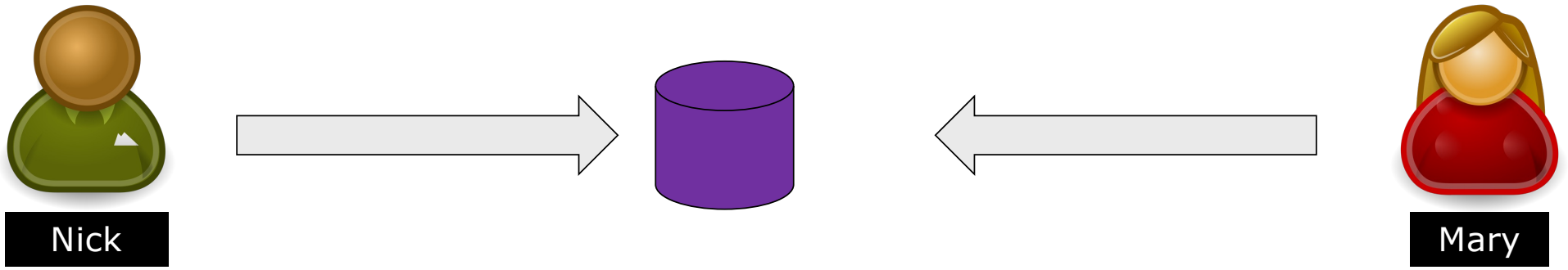

Get
Get
Post
Post


Get
Get
Post
Post

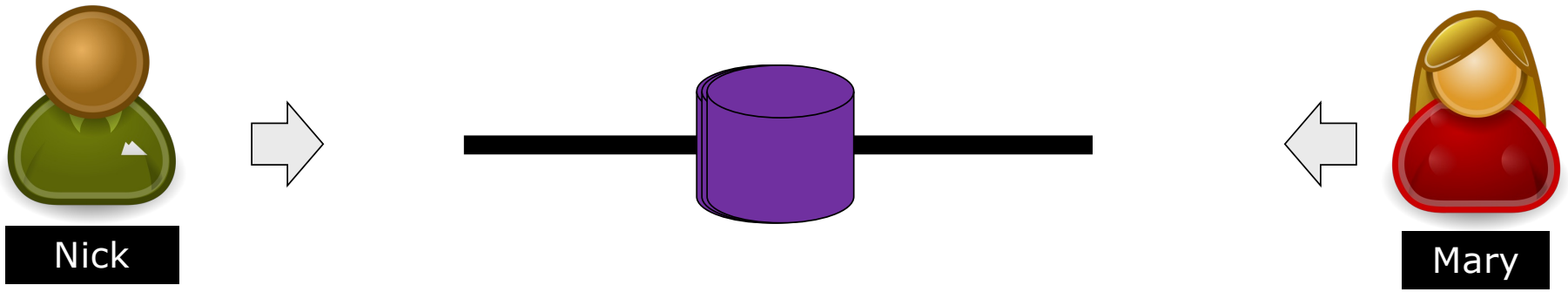

Offline Lock
v101
Get
Get
v101
v101
Post
v101
v102
Version
Stamp
Post

The Effect of Replication

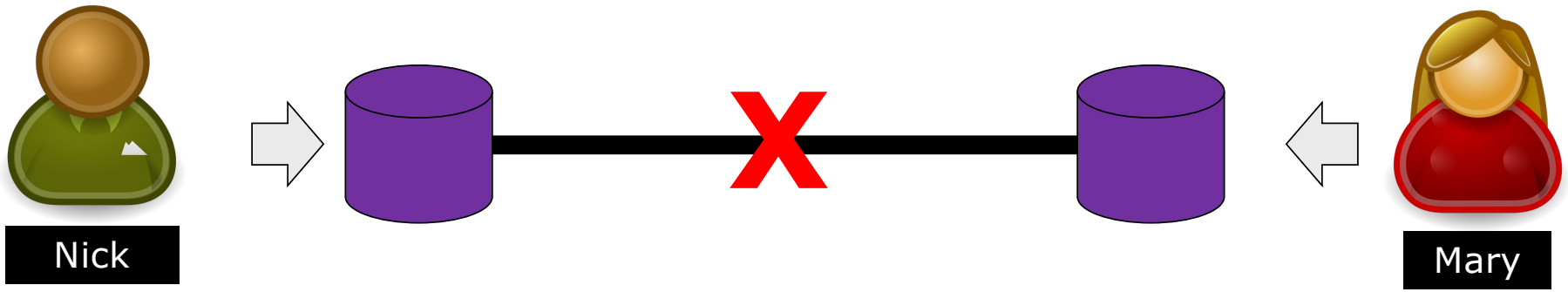
The Effect of Replication



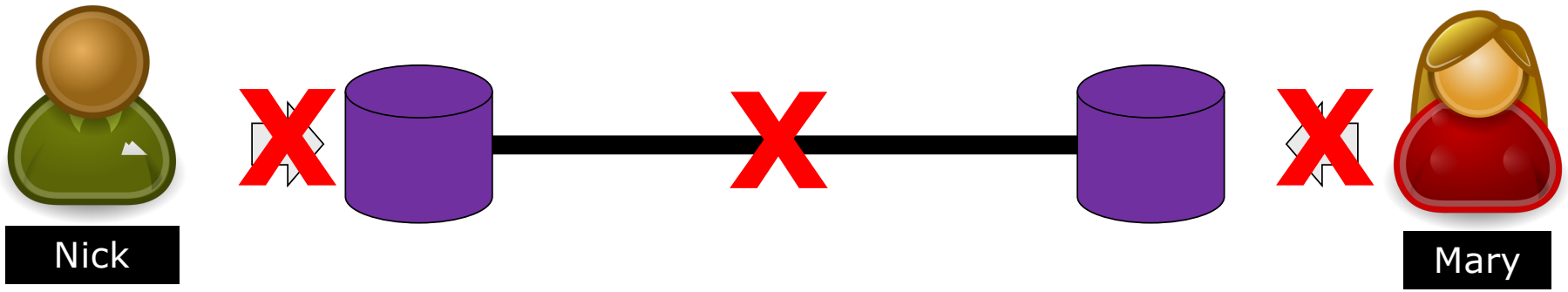
The Effect of Replication



The Effect of Replication

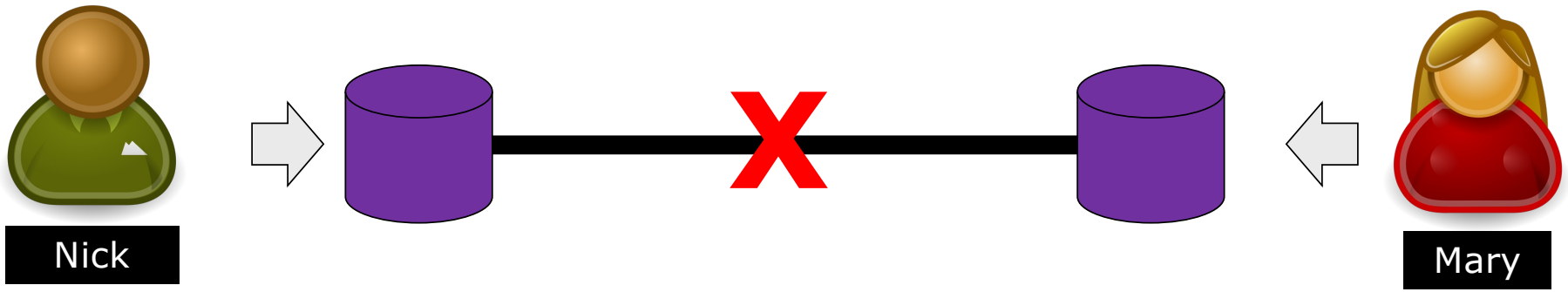


The Effect of Replication



Consistency Guaranteed

The Effect of Replication



Availability Guaranteed

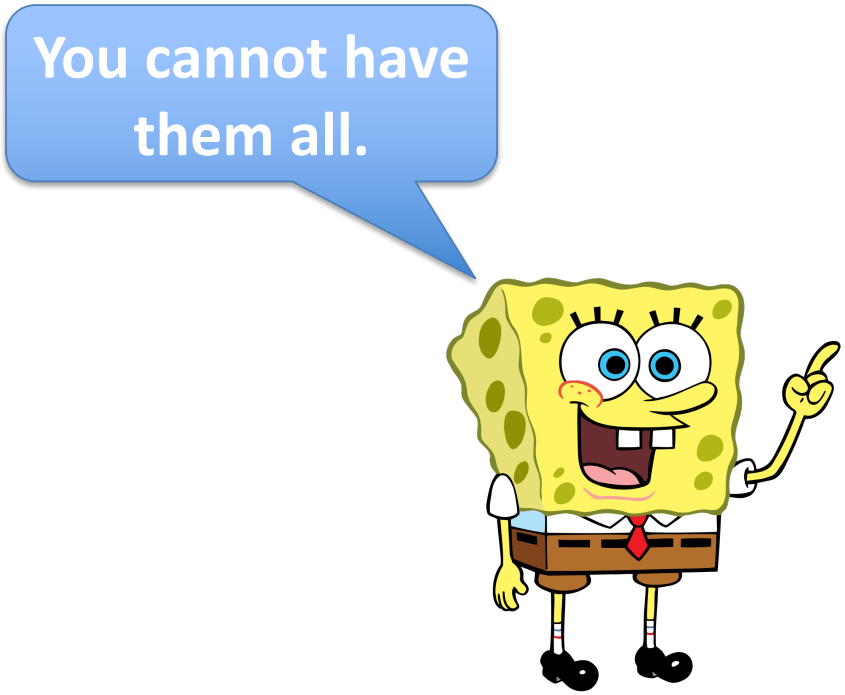
- **Consistency**
  - Do all applications see all the same data?
- **Availability**
  - If some nodes fail, does everything still work?
- **Partitioning**
  - If your nodes can’t talk to each other, does everything still work?

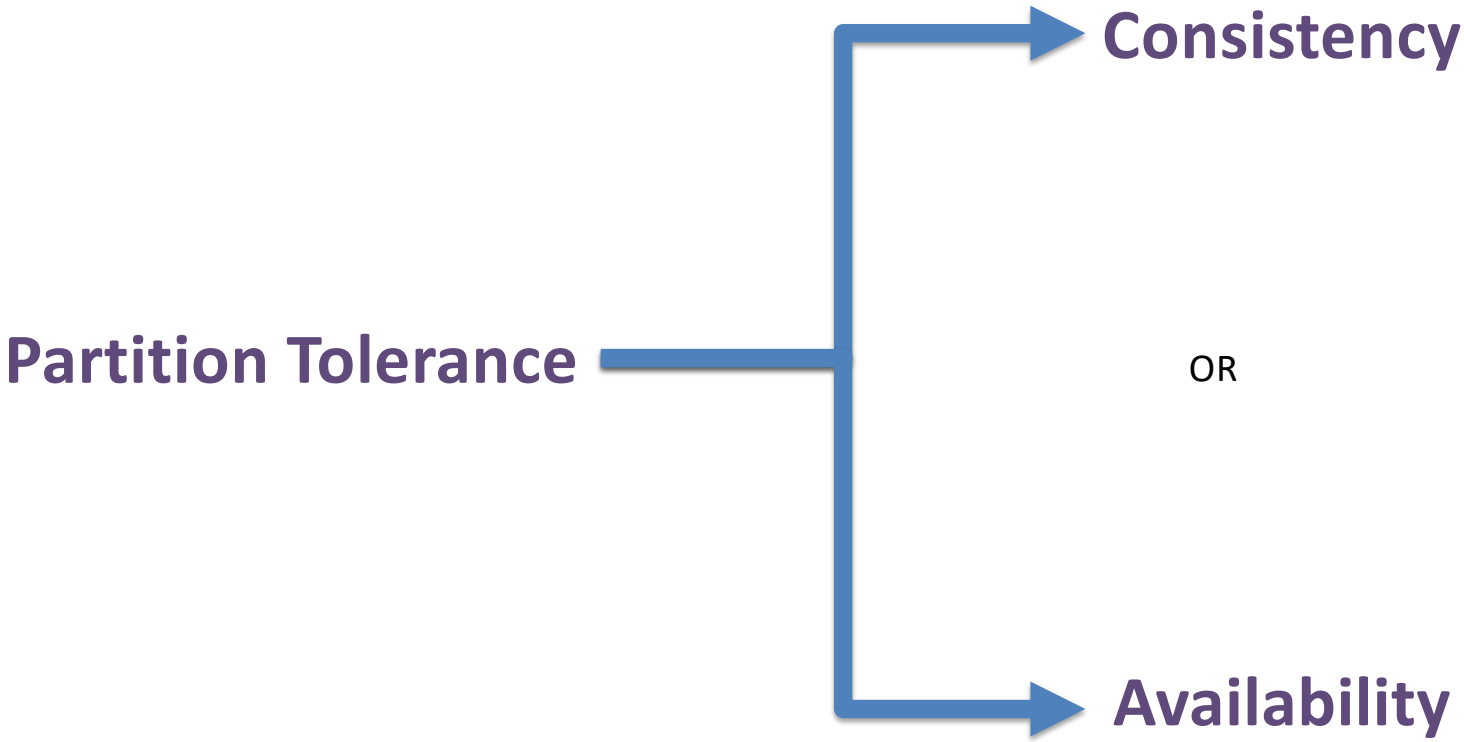
Partition Tolerance

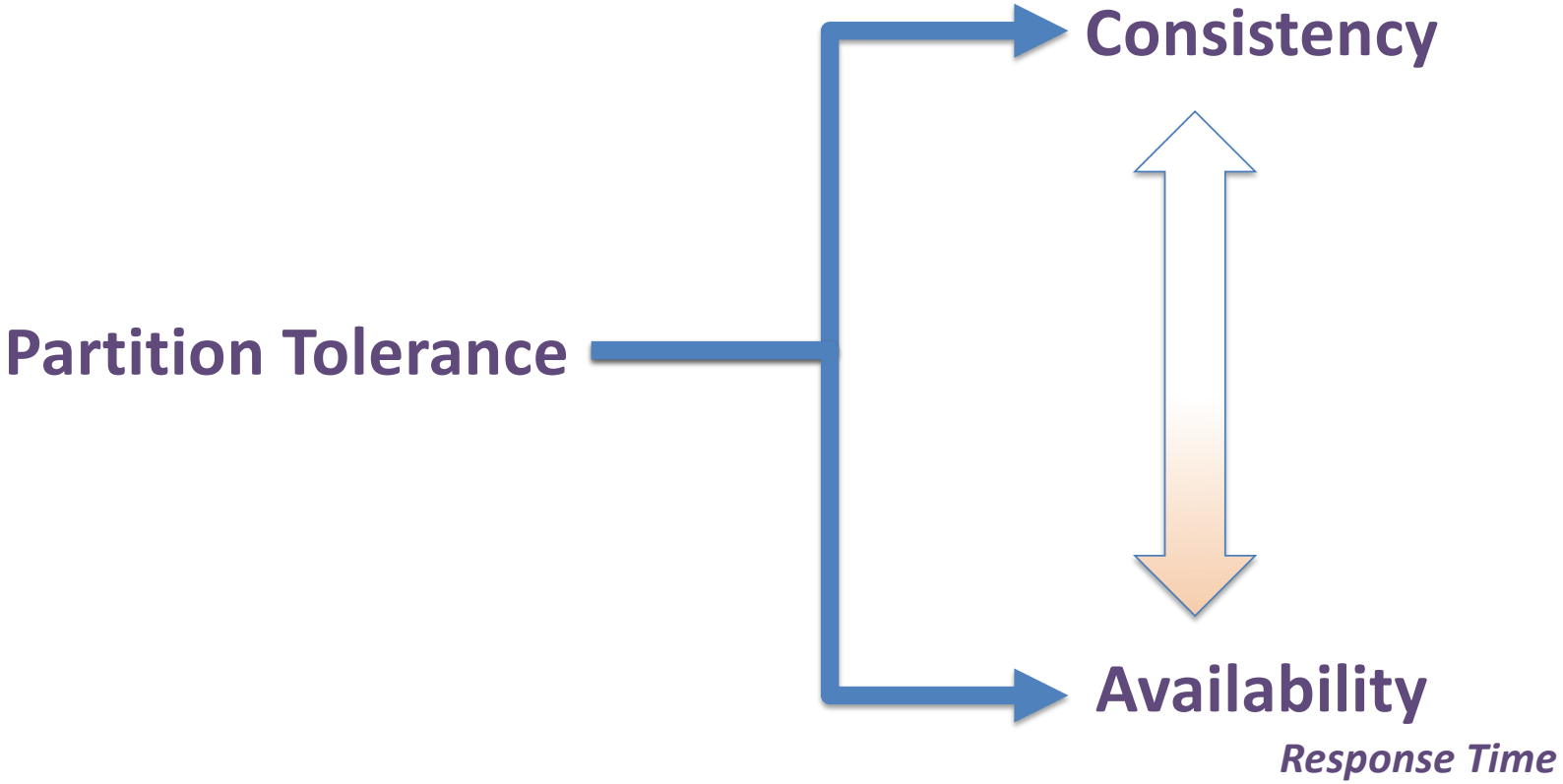
Consistency

CAP Theorem [Brewer 2000, Lynch 2002]

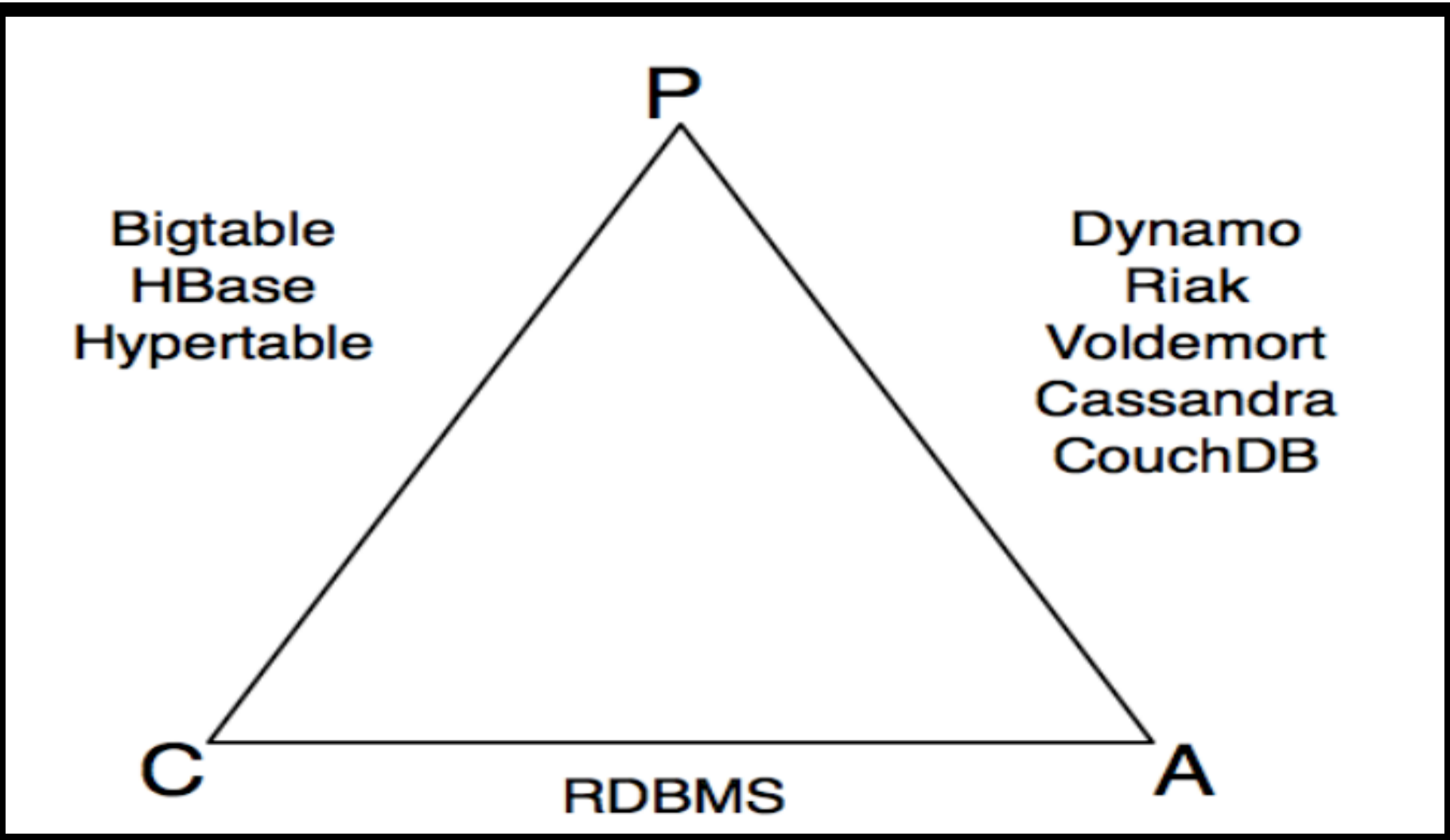
Availability




Consistency
Partition Tolerance
OR
Availability


Consistency
Partition Tolerance
Availability
Response Time

124



src: Shashank Tiwari

Thank you.