NoSQL - Introducción

Rafael Garrote Hernández
Profesor en NoSQL

Objetivos

- Recorrido sobre los diferentes modelos de bases de datos NoSQL.
- Aprender a hacer un modelo de datos para los diferentes tipos de bases de datos NoSQL que vamos a ver.
- Saber seleccionar qué base de datos utilizar para un caso de uso en función de sus características.

Bases de datos:

- MongoDB
- Neo4j

Índice del Módulo

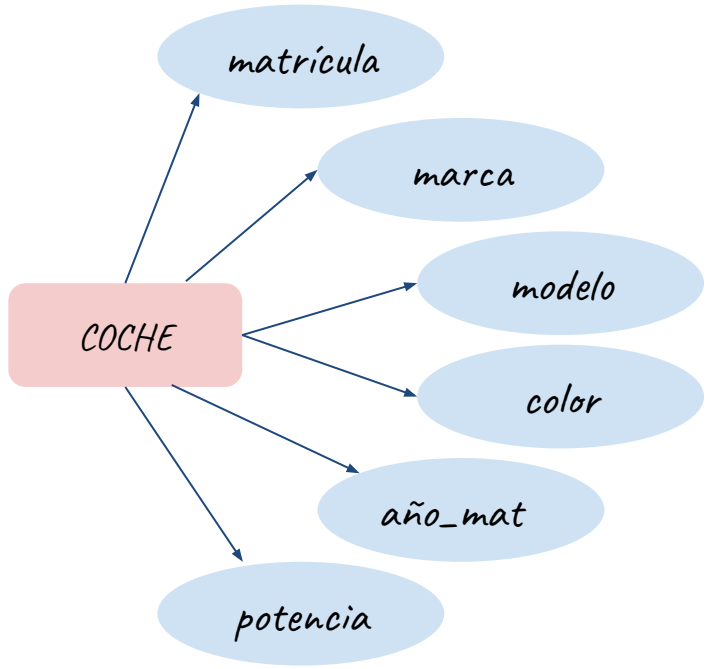
- Bases de datos relacionales
  - Modelo Entidad Relación
  - ACID y Transacciones
  - Problemas
- NoSQL Introducción
  - Teorema CAP
  - Teorema BASE
- Tipos de data stores NoSQL
  - Características
  - Modelado
- MongoDB
- Neo4j

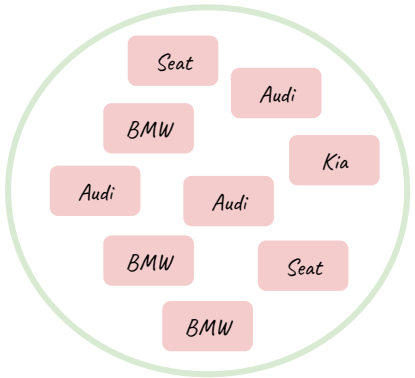
BBDD Relacionales

Matrícula	Marca	Modelo	Color	Potencia	Año_matriculación
3001 AAA	Seat	Leon	Rojo	120	2015
2322 BBB	BMW	X5	Negro	240	2018
1343 CCC	KIA	Rio	Rojo	90	2008
4456 DDD	Mercedes	C180	Azul	100	1995
1532 EEE	BMW	Serie 1	Blanco	120	2010
6786 FFF	Audi	A4	Plata	240	2009
3477 GGG	Opel	Astra	Blanco	90	2011
4858 HHH	Seat	Ibiza	Amarillo	100	2019
1979 III	Alfa Romeo	147	Negro	120	2008
0141 ABB	Mercedes	E220	Azul	220	2017
2242 ACC	Seat	Ibiza	Negro	100	2010
6343 ADD	Opel	Astra	Blanco	90	2005
4254 AEE	Audi	A1	Rojo	140	2016
1564 AFF	KIA	Rio	Amarillo	100	2000

Rafael Garrote Hernández

Matrícula	Marca	Modelo	Color	Potencia	Año_matriculación
3001 AAA	Seat	Leon	Rojo	120	2015
2322 BBB	BMW	X5	Negro	240	2018
1343 CCC	KIA	Rio	Rojo	90	2008
4456 DDD	Mercedes	C180	Azul	100	1995
1532 EEE	BMW	Serie 1	Blanco	120	2010
6786 FFF	Audi	A4	Plata	240	2009
3477 GGG	Opel	Astra	Blanco	90	2011
4858 HHH	Seat	Ibiza	Amarillo	100	2019
1979 III	Alfa Romeo	147	Negro	120	2008
0141 ABB	Mercedes	E220	Azul	220	2017
2242 ACC	Seat	Ibiza	Negro	100	2010
6343 ADD	Opel	Astra	Blanco	90	2005
4254 AEE	Audi	A1	Rojo	140	2016
1564 AFF	KIA	Rio	Amarillo	100	2000





COCHES

Rafael Garrote Hernández

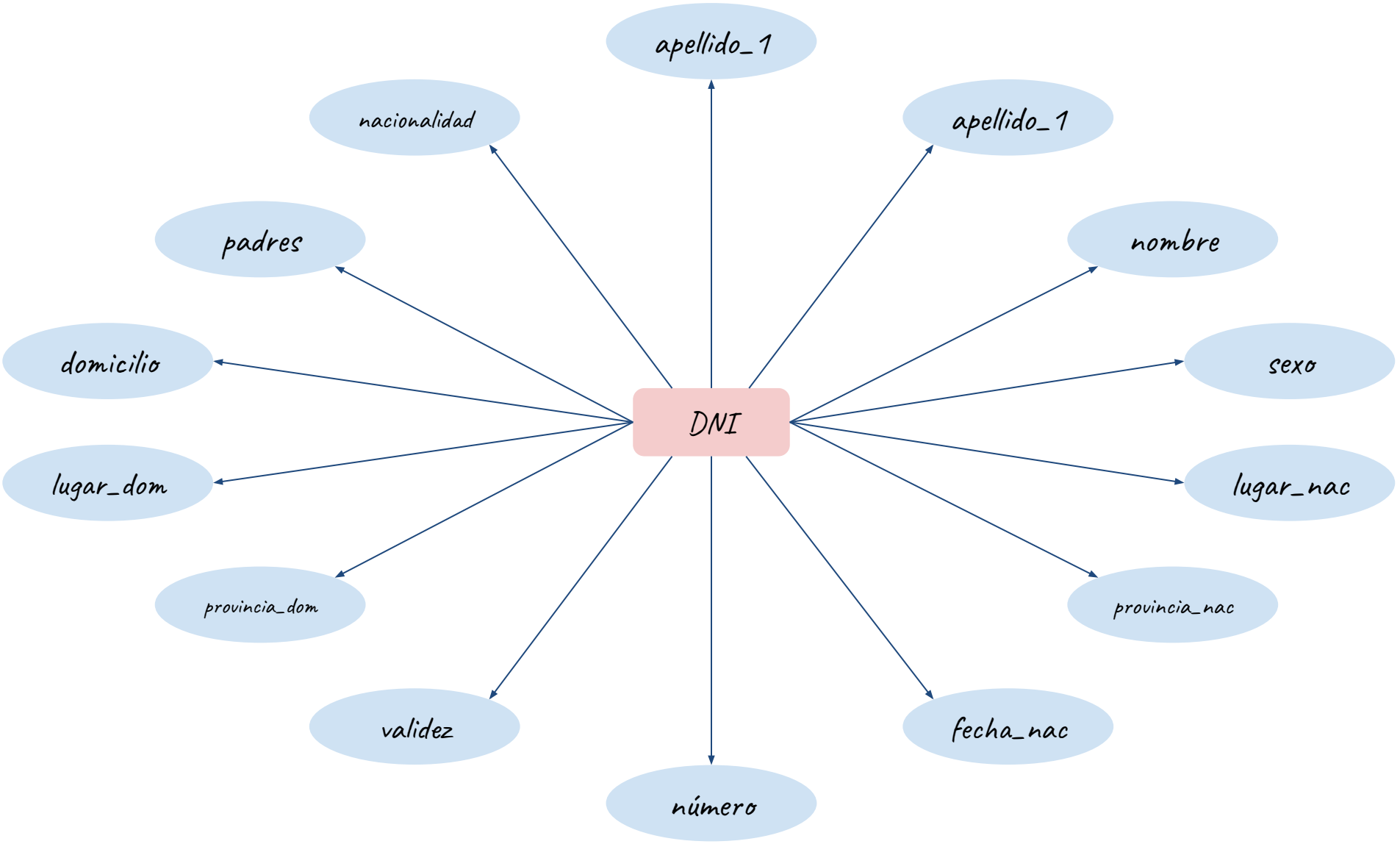
DNI

La Policía Nacional nos pide un sistema para:

- Guardar la información asociada al DNI de los Españoles.
- Poder trazar relaciones de parentesco:
  - Quién es padre/madre de quien.
  - Quién es hijx de quien.
- Qué ciudadanos comparten la misma residencia.


DOCUMENTO NACIONAL DE IDENTIDAD
ESPAÑA
PRIMER APELLIDO
ESPAÑOL
SEGUNDO APELLIDO
ESPAÑOL
NOMBRE
JUAN
SEXO
M
NACIONALIDAD
ESP
FECHA DE NACIMIENTO
10 01 1950
IDESP
AAA023112
VALIDO HASTA
17 04 2018
DNI NÚM.
65004204V

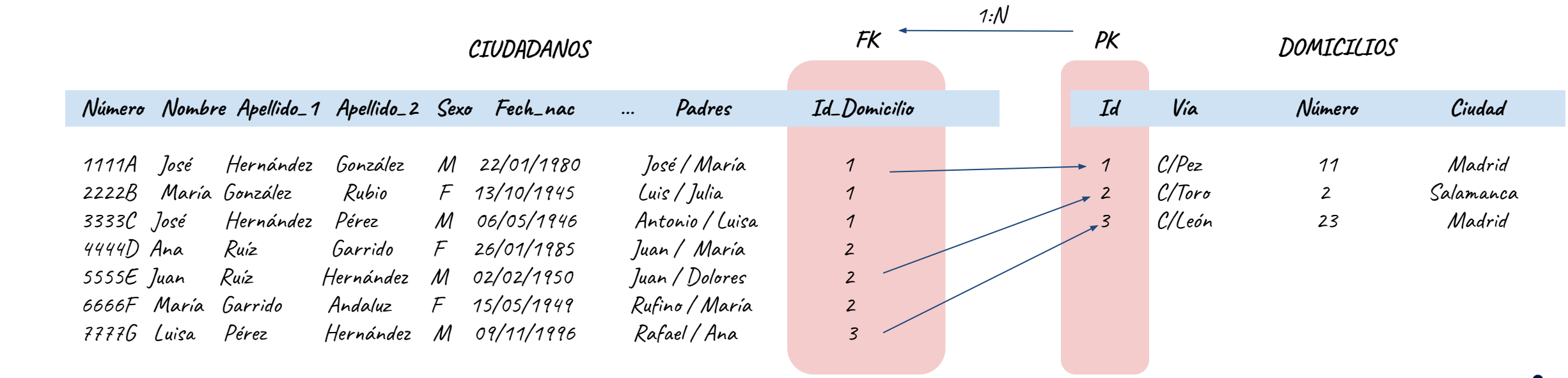

LUGAR DE NACIMIENTO
MADRID
PROVINCIA/PAIS
MADRID
HIJO/A DE
JOSE / MARIA
DOMICILIO
C. PEZ 7
LUGAR DE DOMICILIO
MADRID
PROVINCIA/PAIS
MADRID
EQUIPO
28391A6DK
IDESPAAA0231121650042O4V<<<<<<<
5001105M1804179ESP<<<<<<<<<<<2
ESPANOL<ESPANOL<<JUAN<<<<<<<<<



Rafael Garrote Hernández

CIUDADANOS

Número	Nombre	Apellido_1	Apellido_2	Sexo	Fech_nac	...	Padres	Domicilio
1111A	José	Hernández	González	M	22/01/1980		José / María	C/ Pez, 11, Madrid
2222B	María	González	Rubio	F	13/10/1945		Luis / Julia	C/Pez, 11, Madrid
3333C	José	Hernández	Pérez	M	06/05/1946		Antonio / Luisa	C/Pez, 11, Madrid
4444D	Ana	Ruiz	Garrido	F	26/01/1985		Juan / María	C/Toro, 2, Salamanca
5555E	Juan	Ruiz	Hernández	M	02/02/1950		Juan / Dolores	C/Toro, 2, Salamanca
6666F	María	Garrido	Andaluz	F	15/05/1949		Rufino / María	C/Toro, 2, Salamanca
7777G	Luisa	Pérez	Hernández	M	09/11/1996		Rafael / Ana	C/León, 23, Madrid



Rafael Garrote Hernández

CIUDADANOS

¿?

Número	Nombre	Apellido_1	Apellido_2	Sexo	Fech_nac	...	Padres	Id_Domicilio
1111A	José	Hernández	González	M	22/01/1980		José / María	1
2222B	María	González	Rubio	F	13/10/1945		Luis / Julia	1
3333C	José	Hernández	Pérez	M	06/05/1946		Antonio / Luisa	1
4444D	Ana	Ruíz	Garrido	F	26/01/1985		Juan / María	2
5555E	Juan	Ruíz	Hernández	M	02/02/1950		Juan / Dolores	2
6666F	María	Garrido	Andaluz	F	15/05/1949		Rufino / María	2
7777G	Luisa	Pérez	Hernández	M	09/11/1996		Rafael / Ana	3

CIUDADANOS

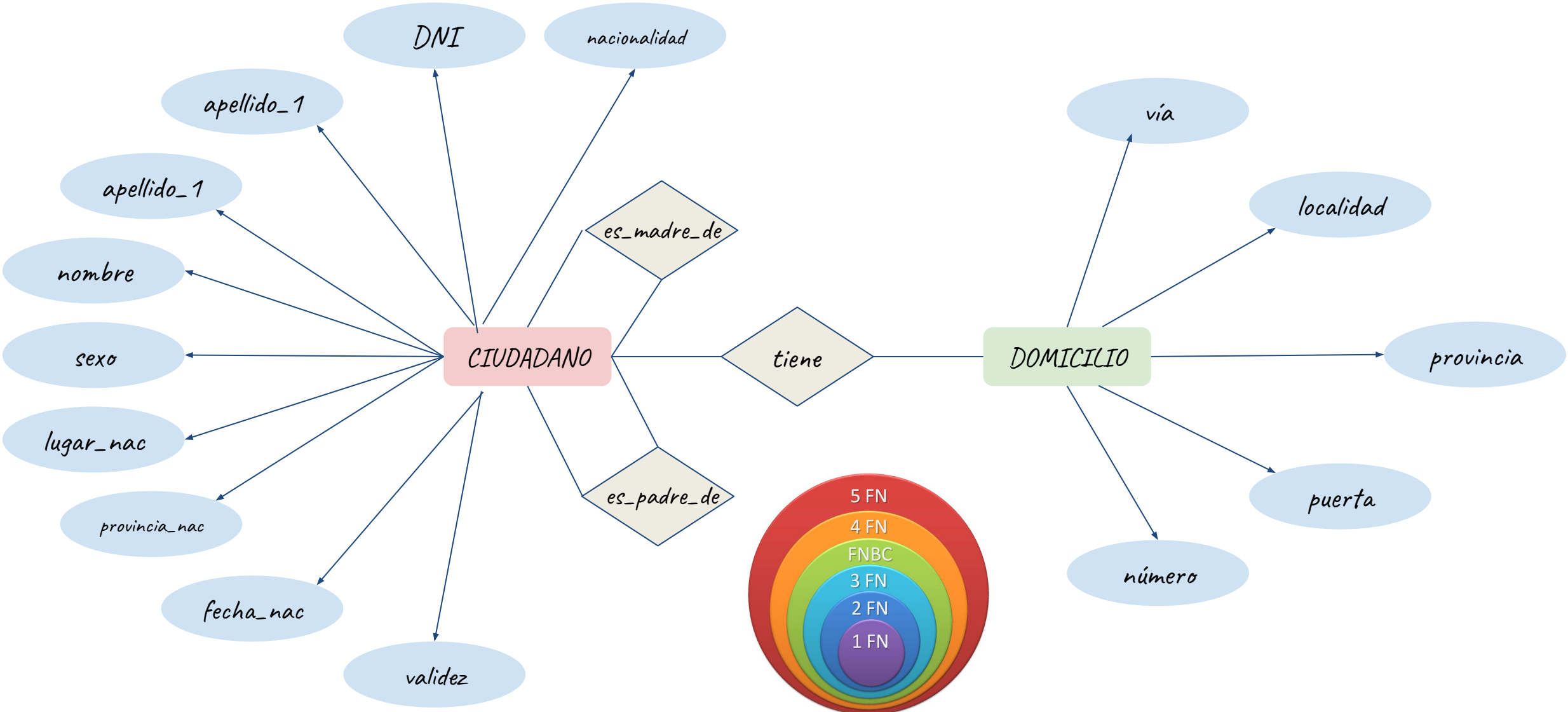
PK

FK

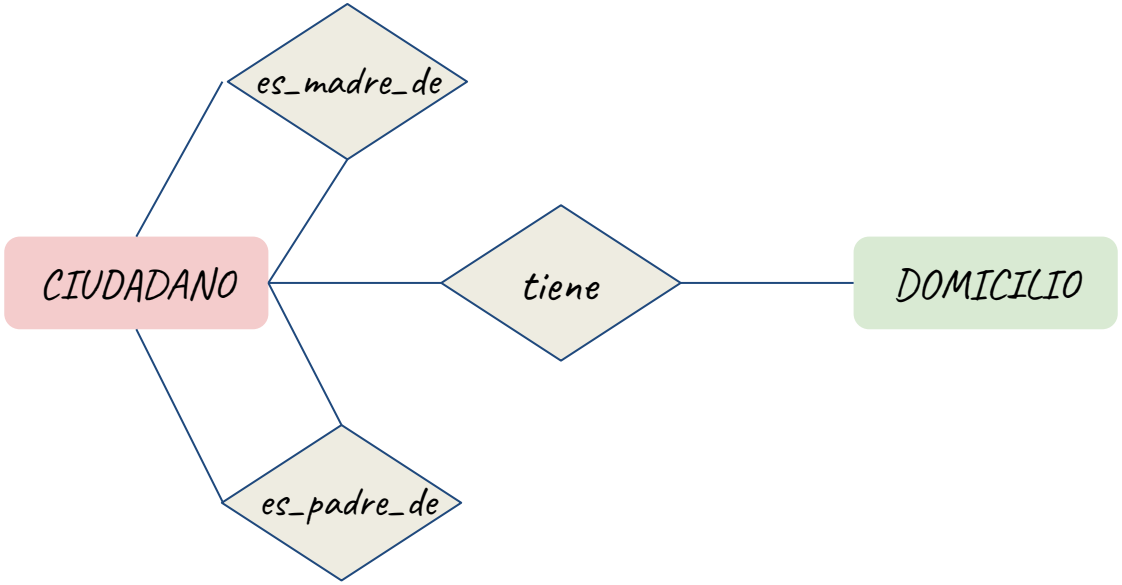
FK

Número	Nombre	Apellido_1	Apellido_2	Sexo	Fech_nac	...	Id_Madre	Id_Padre	Id_Domicilio
1111A	José	Hernández	González	M	22/01/1980		2222B	3333C	1
2222B	María	González	Rubio	F	13/10/1945		0001C	0101D	1
3333C	José	Hernández	Pérez	M	06/05/1946		1321A	1654A	1
4444D	Ana	Ruíz	Garrido	F	26/01/1985		6666F	5555E	2
5555E	Juan	Ruíz	Hernández	M	02/02/1950		1846A	9654F	2
6666F	María	Garrido	Andaluz	F	15/05/1949		2376D	2323H	2
7777G	Luisa	Pérez	Hernández	M	09/11/1996		5454F	9898J	3

Rafael Garrote Hernández

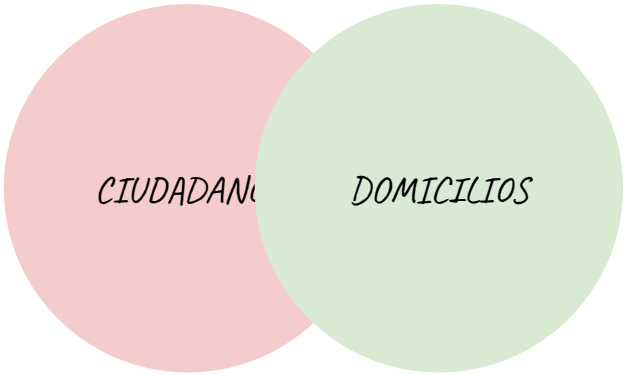


Rafael Garrote Hernández



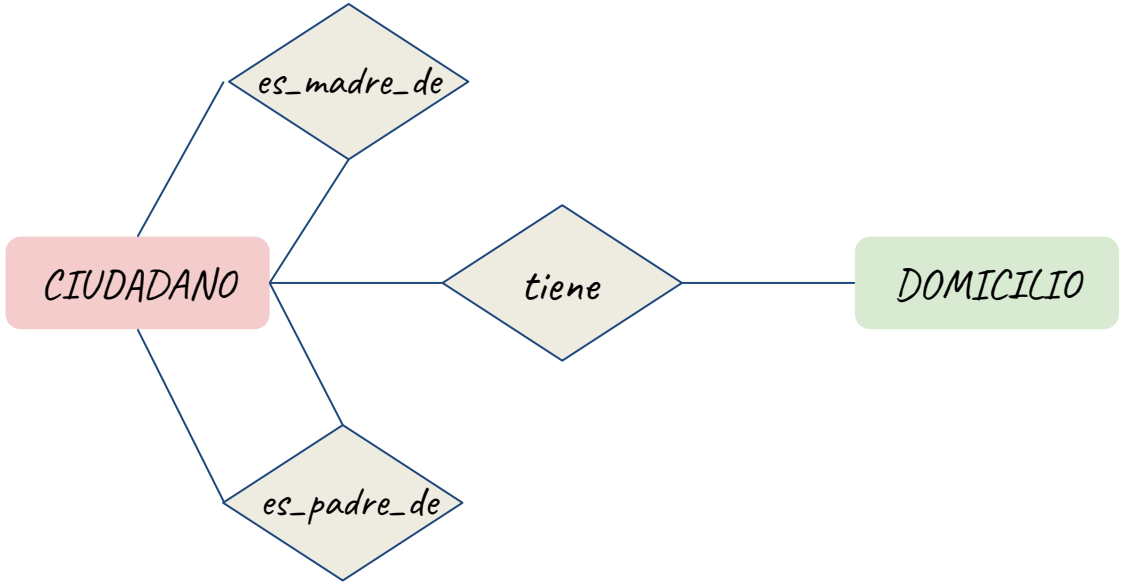
Número	Nombre	Apellido_1	Apellido_2	Sexo	Fech_nac	...	Padres	Domicilio
1111A	José	Hernández	González	M	22/01/1980		José / María	C/ Pez, 11, Madrid
2222B	María	González	Rubio	F	13/10/1945		Luis / Julia	C/Pez, 11, Madrid
3333C	José	Hernández	Pérez	M	06/05/1946		Antonio / Luisa	C/Pez, 11, Madrid
4444D	Ana	Ruíz	Garrido	F	26/01/1985		Juan / María	C/Toro, 2, Salamanca
5555E	Juan	Ruíz	Hernández	M	02/02/1950		Juan / Dolores	C/Toro, 2, Salamanca
6666F	María	Garrido	Andaluz	F	15/05/1949		Rufino / María	C/Toro, 2, Salamanca
7777G	Luisa	Pérez	Hernández	M	09/11/1996		Rafael / Ana	C/León, 23, Madrid

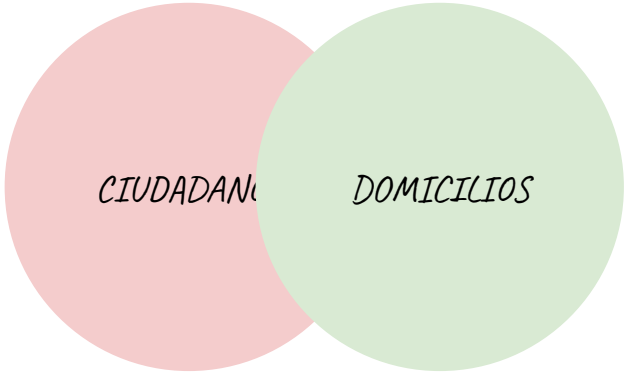
NoSQL



JOIN

Rafael Garrote Hernández





JOIN

Esquema

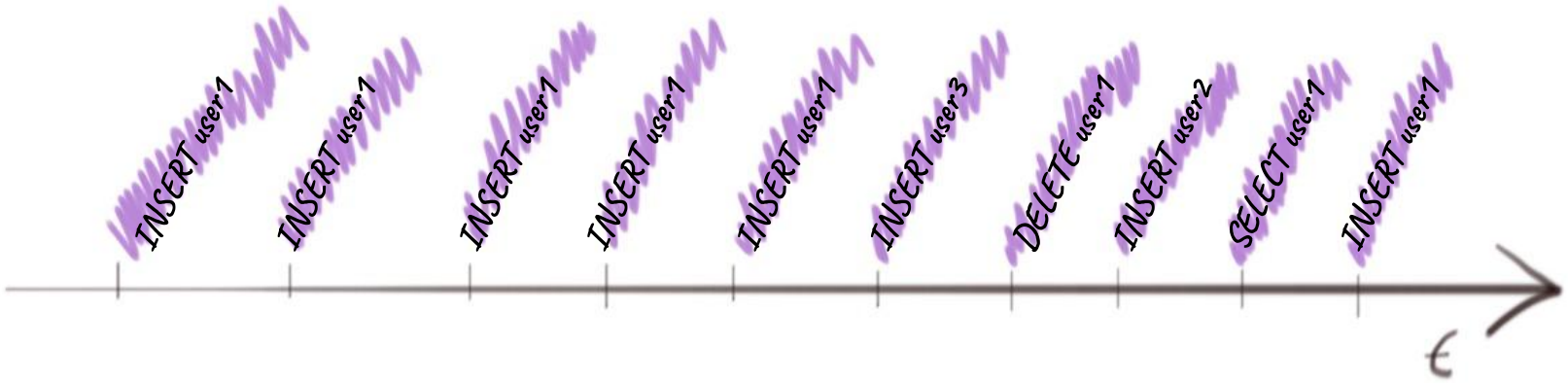
Algebra

SQL

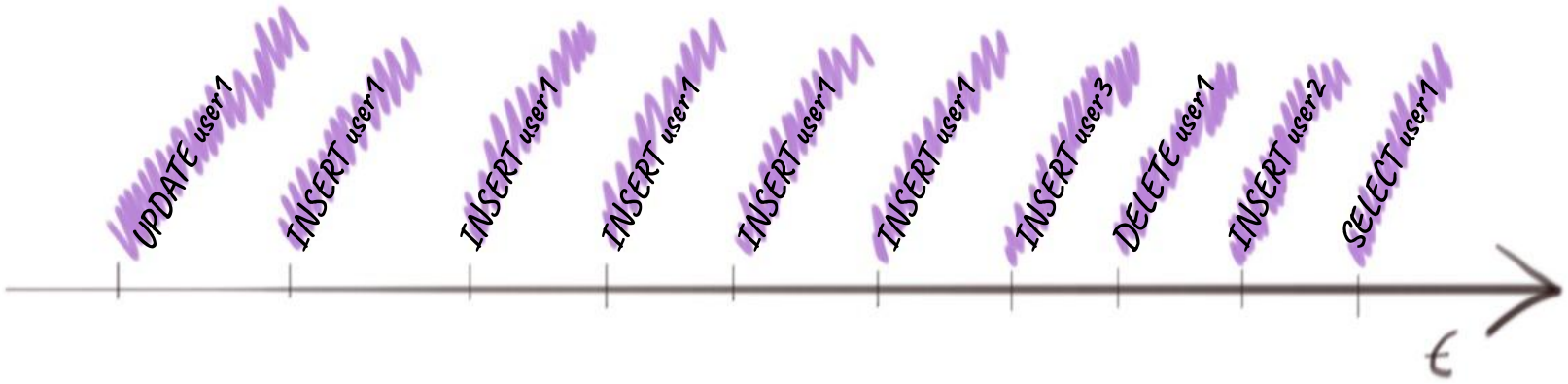
Rafael Garrote Hernández

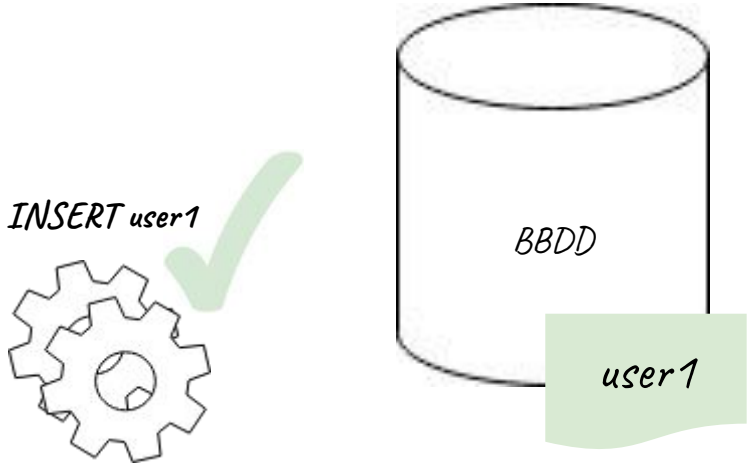
Modelo Entidad Relación

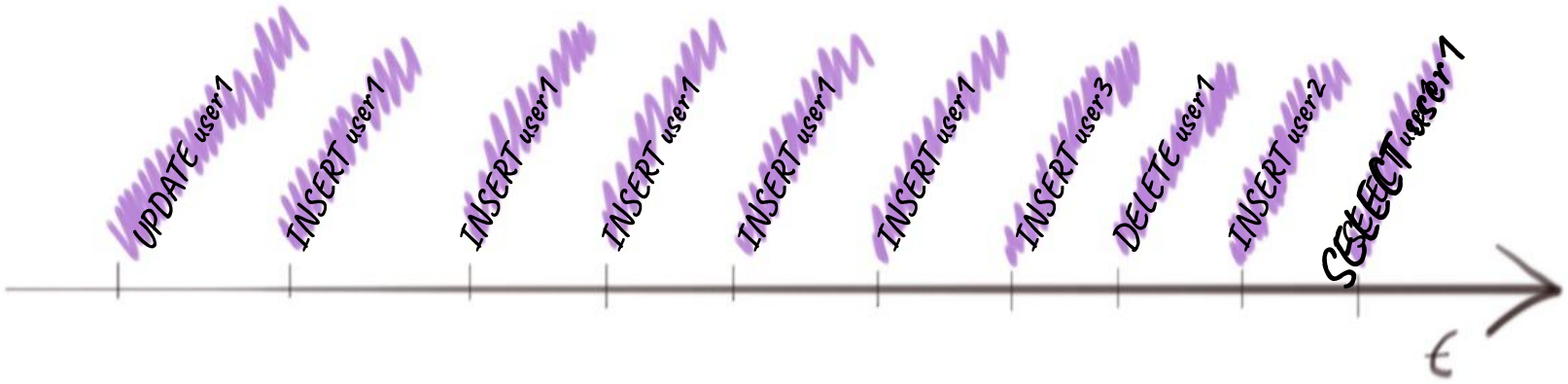
- Herramienta para modelar el negocio:
  - Dar sentido a los datos en el contexto del negocio.
  - Entidades, atributos y relaciones.
  - Estandarizado: Formas Normales.
- Consistencia:
  - No hay datos duplicados.
  - Uso eficiente de disco.
- Integridad:
  - Datos.
  - Relaciones.
- Lenguaje de definición y consulta:
  - Algebra para recuperar los datos de forma consistente.
  - Facilidad de uso que evita problemas de integridad.

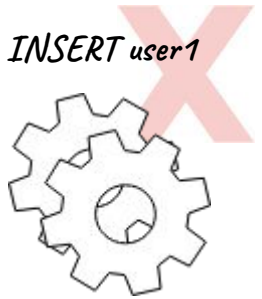

INSERT user1
INSERT user1
INSERT user1
INSERT user1
INSERT user1
INSERT user3
DELETE user1
INSERT user2
SELECT user1
INSERT user1
€

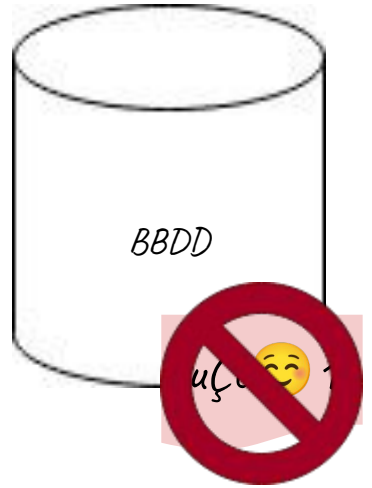

BBDD


UPDATE user1
INSERT user1
INSERT user1
INSERT user1
INSERT user1
INSERT user1
INSERT user3
DELETE user1
INSERT user2
SELECT user1
€

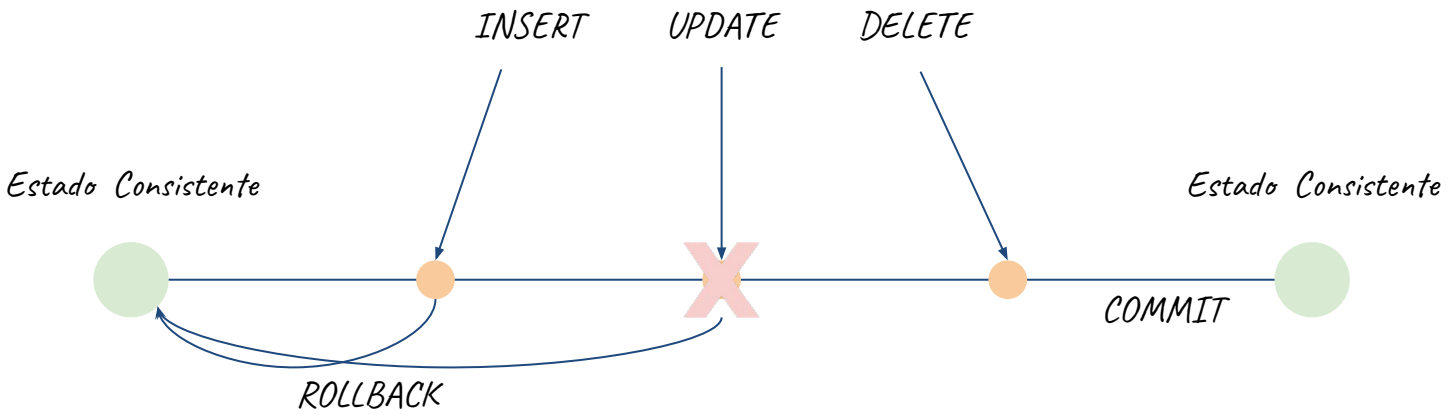

INSERT user1
BBDD
user1


UPDATE user1
INSERT user1
INSERT user1
INSERT user1
INSERT user1
INSERT user1
INSERT user3
DELETE user1
INSERT user2
SELECT user1
t


INSERT user1

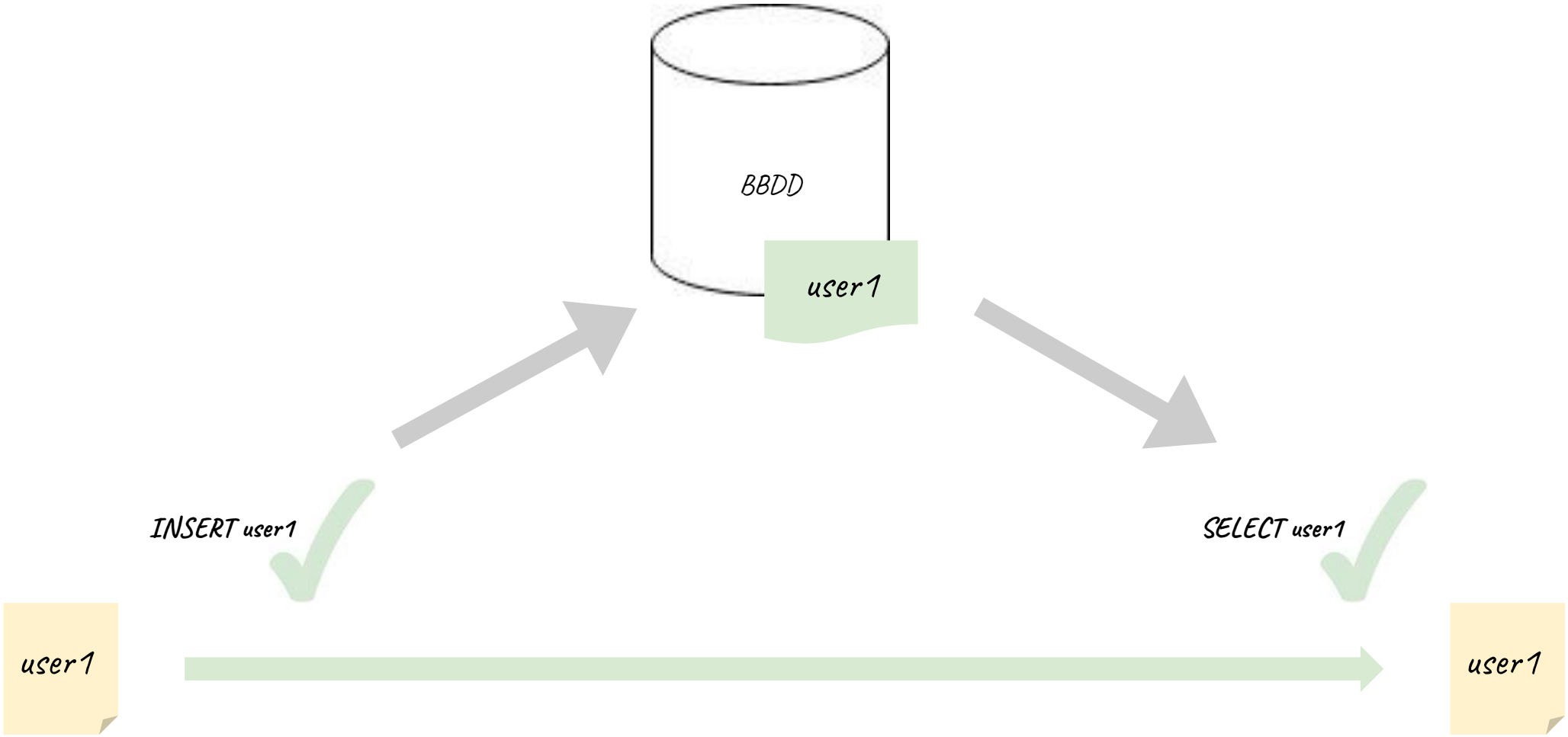

BBDD

Transacción



Atomicity

Toda transacción o se realiza correctamente o falla. No almacena datos parciales.

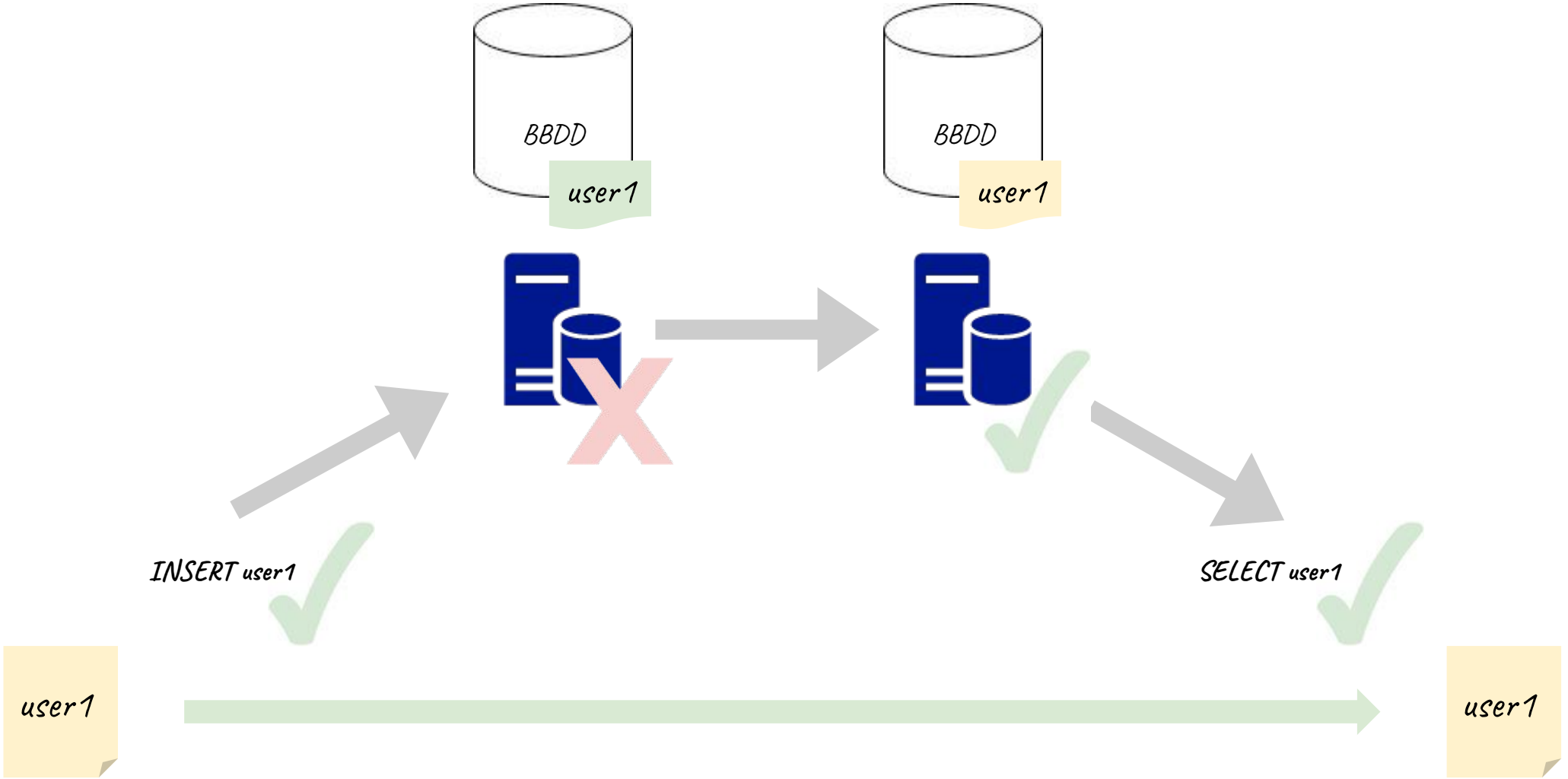

BBDD
user1
INSERT user1
SELECT user1
user1
user1

Atomicity

Toda transacción o se realiza correctamente o falla. No almacena datos parciales.

Consistency

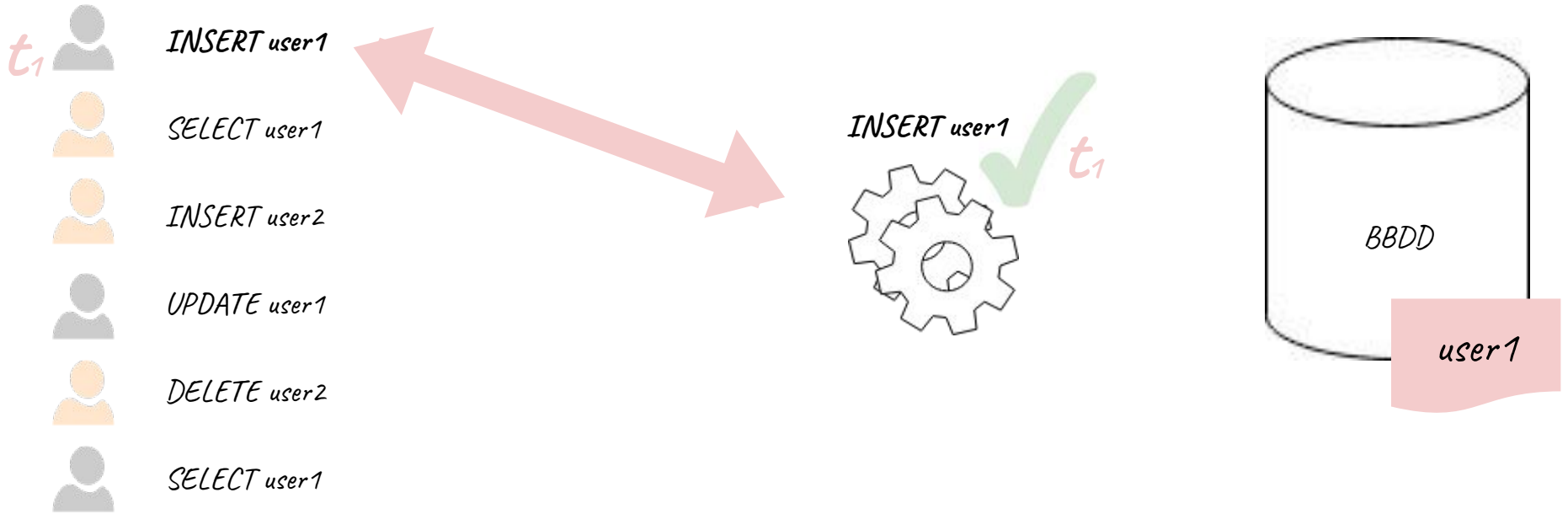
El resultado de una transacción realiza correctamente es visible a las siguientes operaciones.

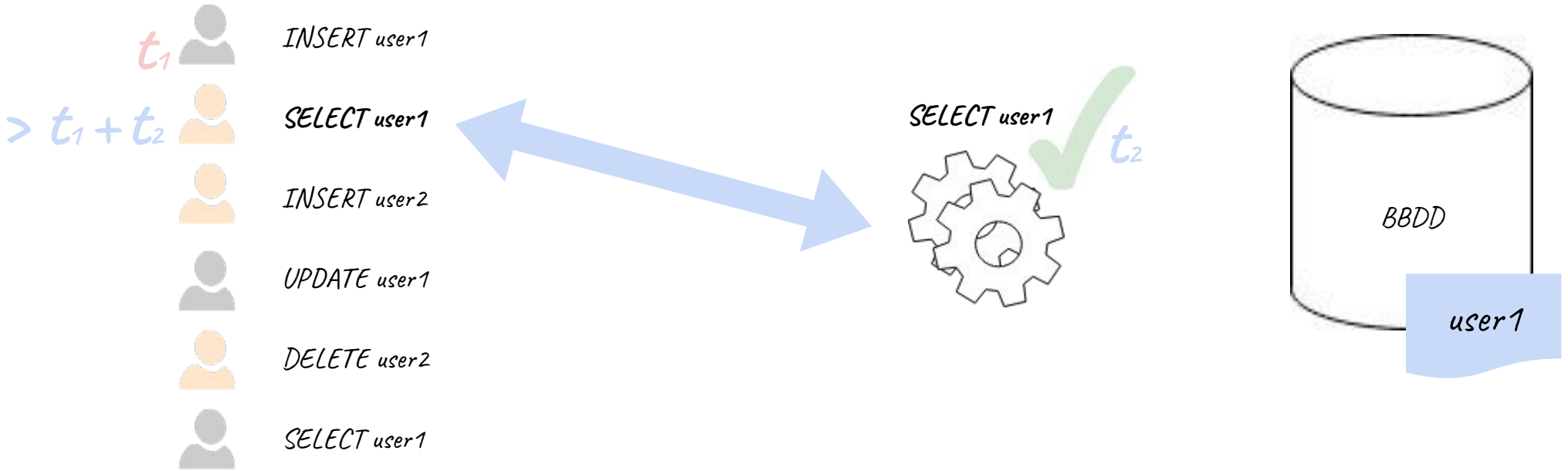

BBDD
user1
BBDD
user1
INSERT user1
SELECT user1
user1
user1

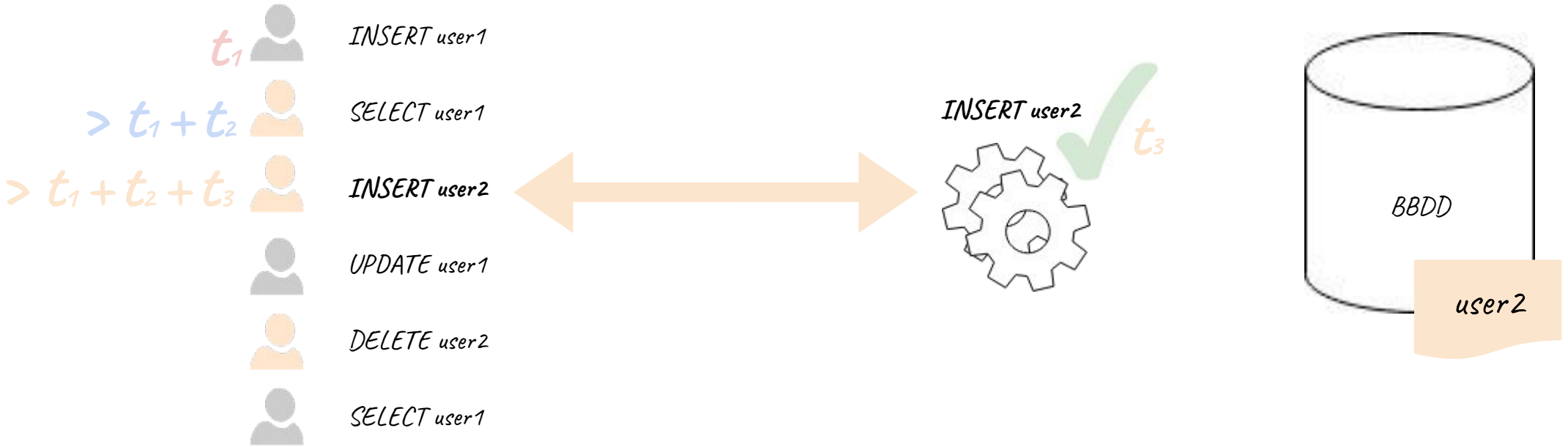
Atomicity Toda transacción o se realiza correctamente o falla. No almacena datos parciales.

Consistency El resultado de una transacción realiza correctamente es visible a las siguientes operaciones.

Durability Una vez que la transacción se ha completado, los datos siguen siendo accesibles aunque haya una caída de la base de datos.


t1
INSERT user1
SELECT user1
INSERT user2
UPDATE user1
DELETE user2
SELECT user1
INSERT user1
t1
BBDD
user1

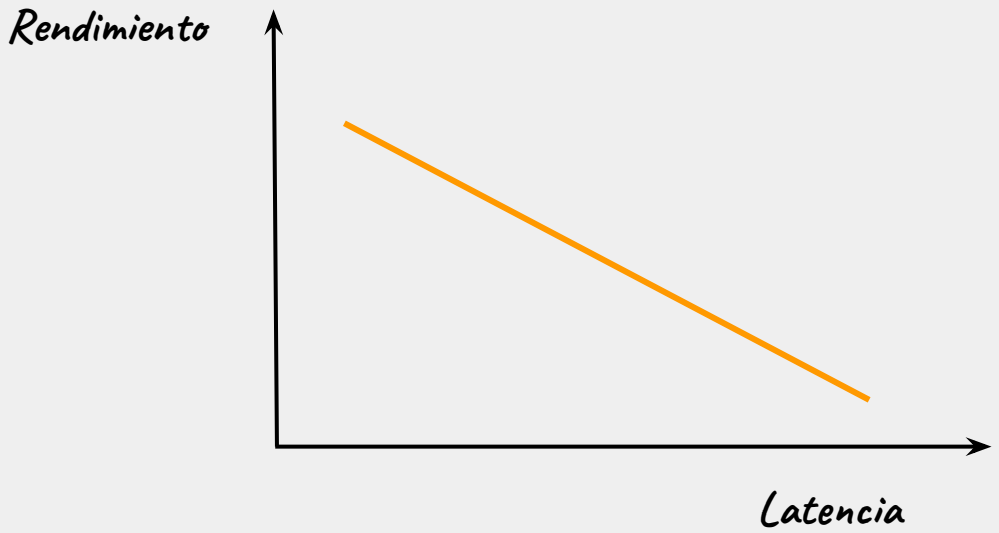

t_1
INSERT user1
> $t_1 + t_2$
SELECT user1
INSERT user2
UPDATE user1
DELETE user2
SELECT user1
SELECT user1
t_2
BBDD
user1


t_1
INSERT user1
$> t_1 + t_2$
SELECT user1
$> t_1 + t_2 + t_3$
INSERT user2
UPDATE user1
DELETE user2
SELECT user1
INSERT user2
t_3
BBDD
user2

Latencia

Tiempo transcurrido entre la petición de una consulta y la obtención de la respuesta con el resultado de la ejecución de esa consulta.

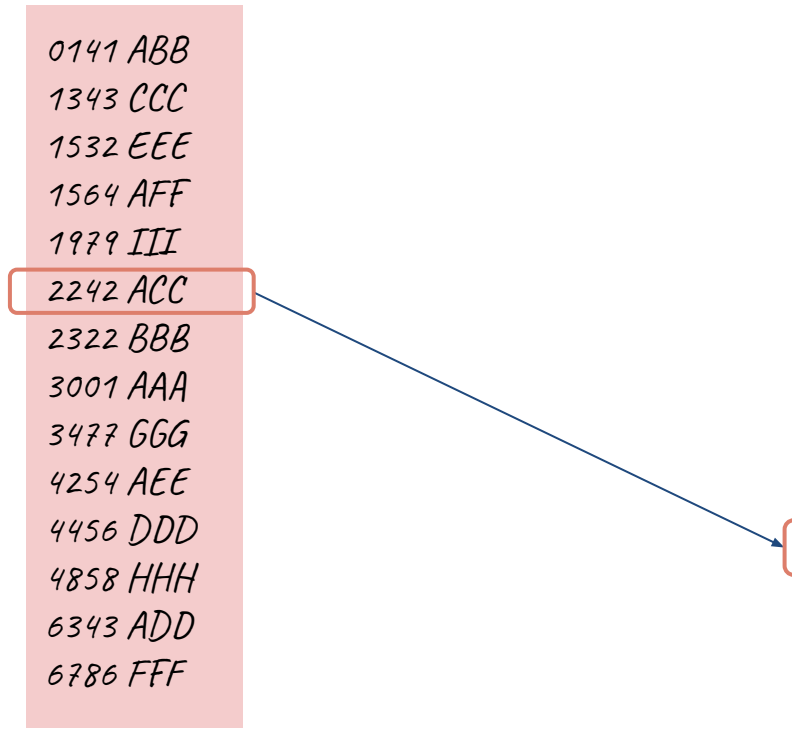
Rendimiento Vs. Latencia



El Rendimiento mejora si mejoran los tiempos de respuesta.

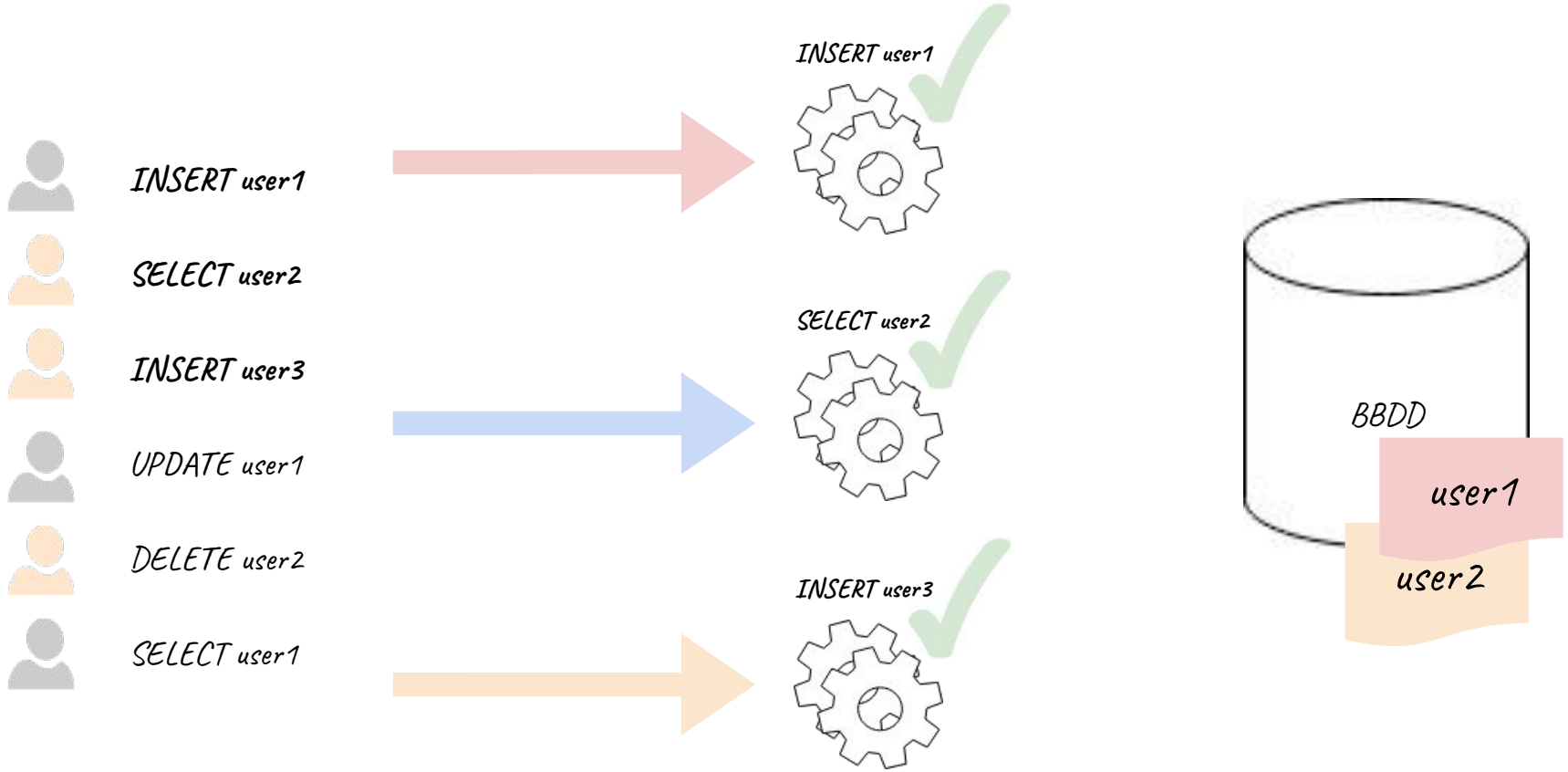
Índices

```
SELECT * FROM coches
WHERE Matricula = 2242ACC
```



Matrícula	Marca	Modelo	Color	Potencia	Año_matriculación
3001 AAA	Seat	Leon	Rojo	120	2015
2322 BBB	BMW	X5	Negro	240	2018
1343 CCC	KIA	Rio	Rojo	90	2008
4456 DDD	Mercedes	C180	Azul	100	1995
1532 EEE	BMW	Serie 1	Blanco	120	2010
6786 FFF	Audi	A4	Plata	240	2009
3477 GGG	Opel	Astra	Blanco	90	2011
4858 HHH	Seat	Ibiza	Amarillo	100	2019
1979 III	Alfa Romeo	147	Negro	120	2008
0141 ABB	Mercedes	E220	Azul	220	2017
2242 ACC	Seat	Ibiza	Negro	100	2010
6343 ADD	Opel	Astra	Blanco	90	2005
4254 AEE	Audi	A1	Rojo	140	2016
1564 AFF	KIA	Rio	Amarillo	100	2000

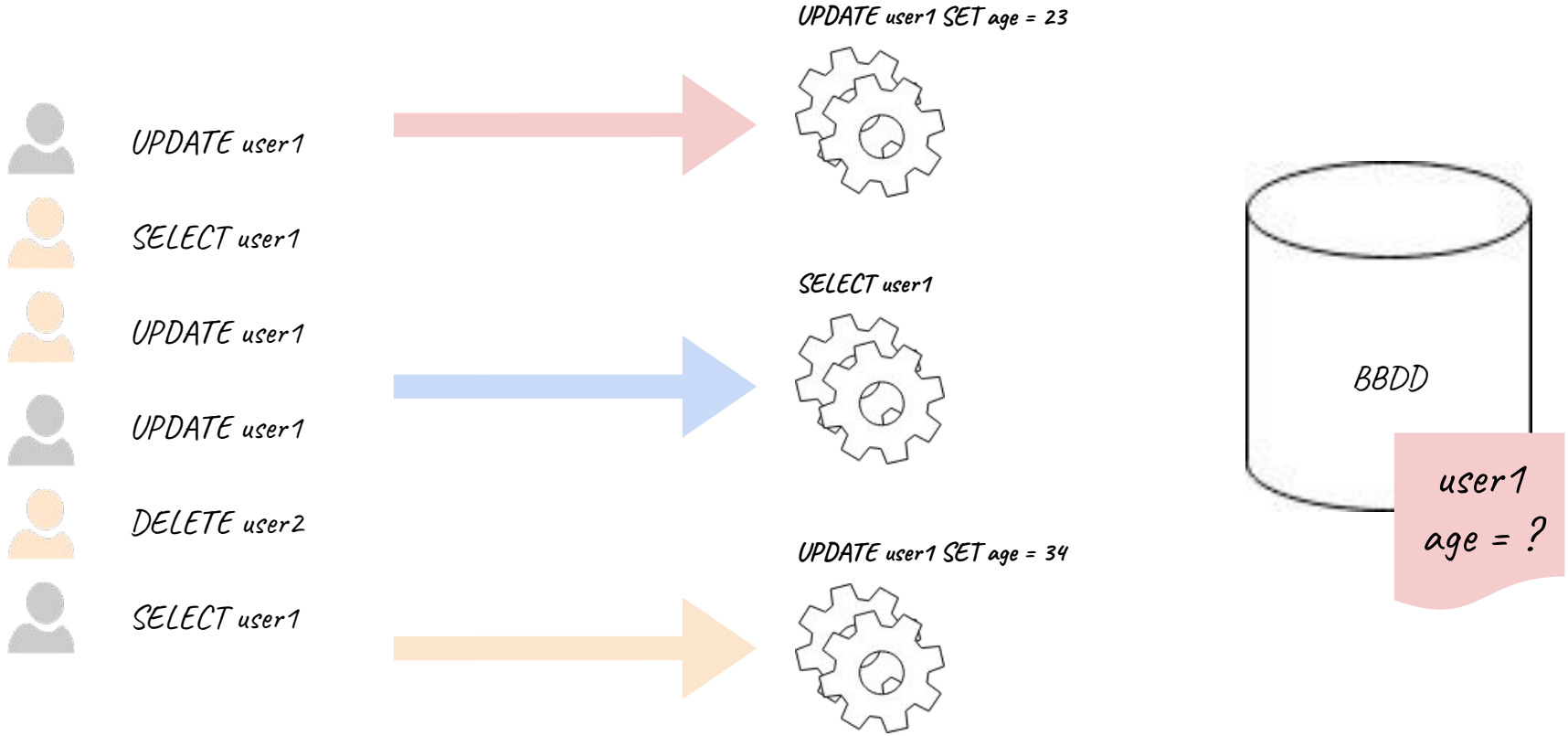
Rafael Garrote Hernández

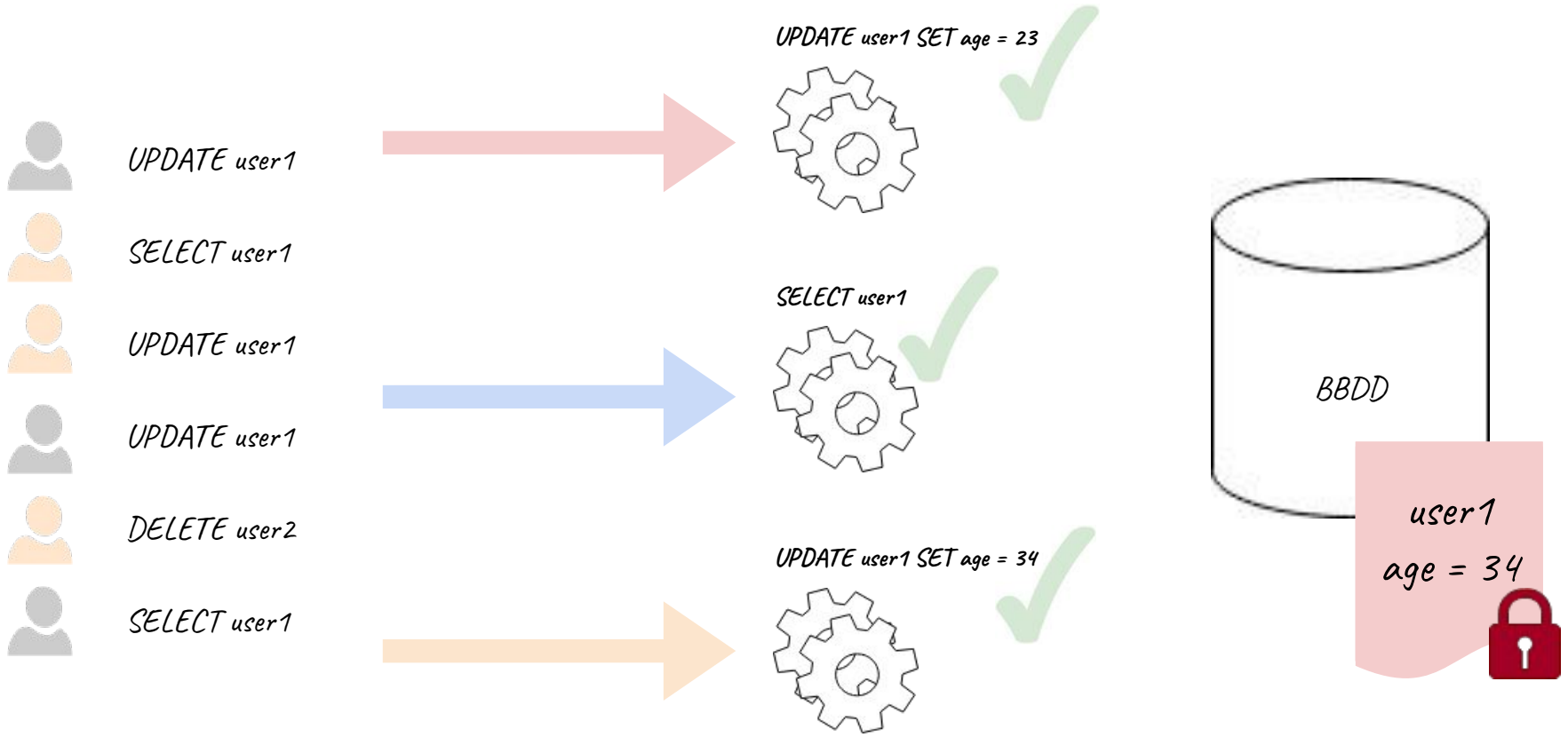


Rafael Garrote Hernández

Carga

Número total de peticiones que se pueden gestionar a la vez.


UPDATE user1
SELECT user1
UPDATE user1
UPDATE user1
DELETE user2
SELECT user1
UPDATE user1 SET age = 23
SELECT user1
UPDATE user1 SET age = 34
BBDD
user1
age = ?


UPDATE user1
SELECT user1
UPDATE user1
UPDATE user1
DELETE user2
SELECT user1
UPDATE user1 SET age = 23
SELECT user1
UPDATE user1 SET age = 34
BBDD
user1
age = 34

Contención

- Cuando tenemos dos clientes intentando acceder a un mismo recurso limitado están en competición.
- Esta competición sólo puede tener un ganador, el resto están condenados a esperar por el ganador.
- Según aumenta el número de recursos por los que se compite, el tiempo de espera aumenta.
- A medida que la carga aumenta llegará un punto en el que se excedan los límites aceptables de espera .
- La contención limita la paralelización.

Teorema ACID

Atomicity Toda operación o se realiza correctamente o falla. No almacena datos parciales.

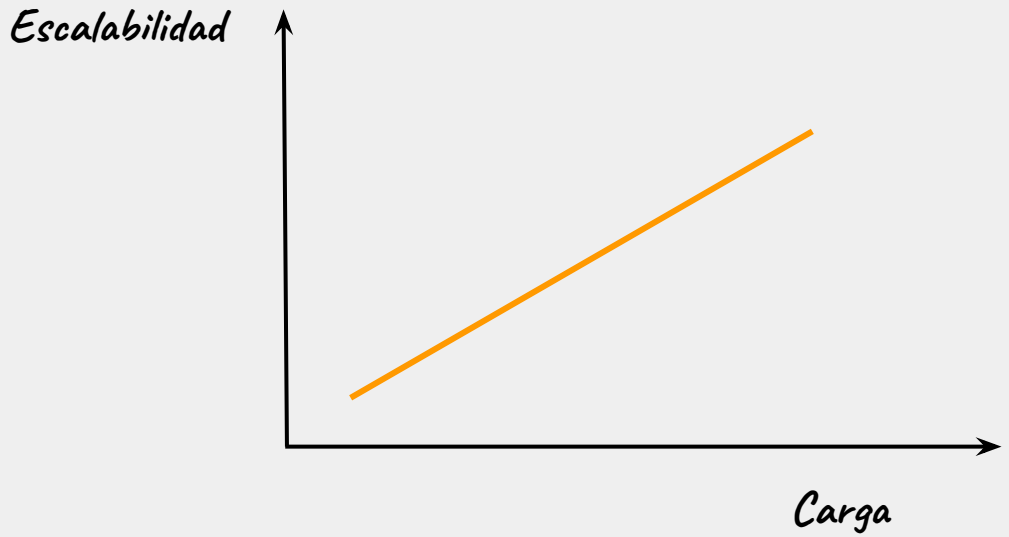
Consistency El resultado de una transacción realiza correctamente es visible a las siguientes operaciones.

Isolation La transacción incompleta de un cliente no genera efectos de lado sobre las transacciones del resto de clientes.

Durability Una vez que la transacción se ha completado, los datos siguen siendo accesibles aunque haya una caída de la base de datos.

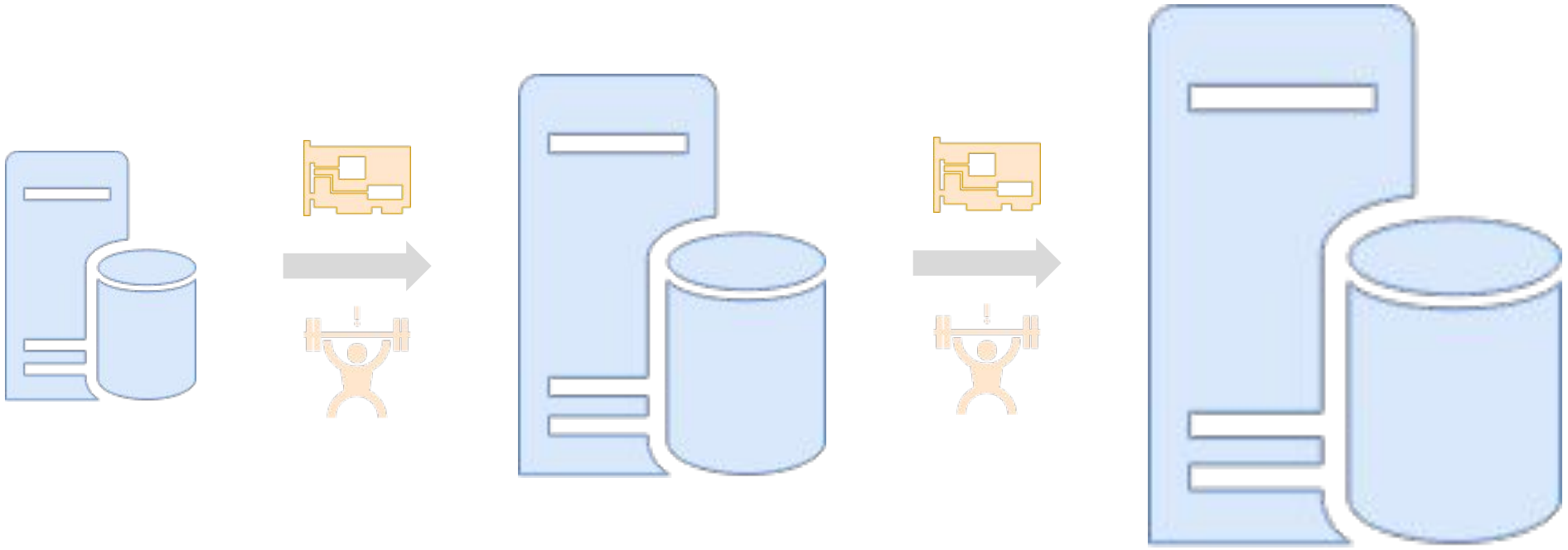
Rafael Garrote Hernández

Escalabilidad Vs. Carga



La Escalabilidad mejora si se incrementa la capacidad de gestionar la Carga.

Escalado Vertical



Escalado Vertical

Ventajas

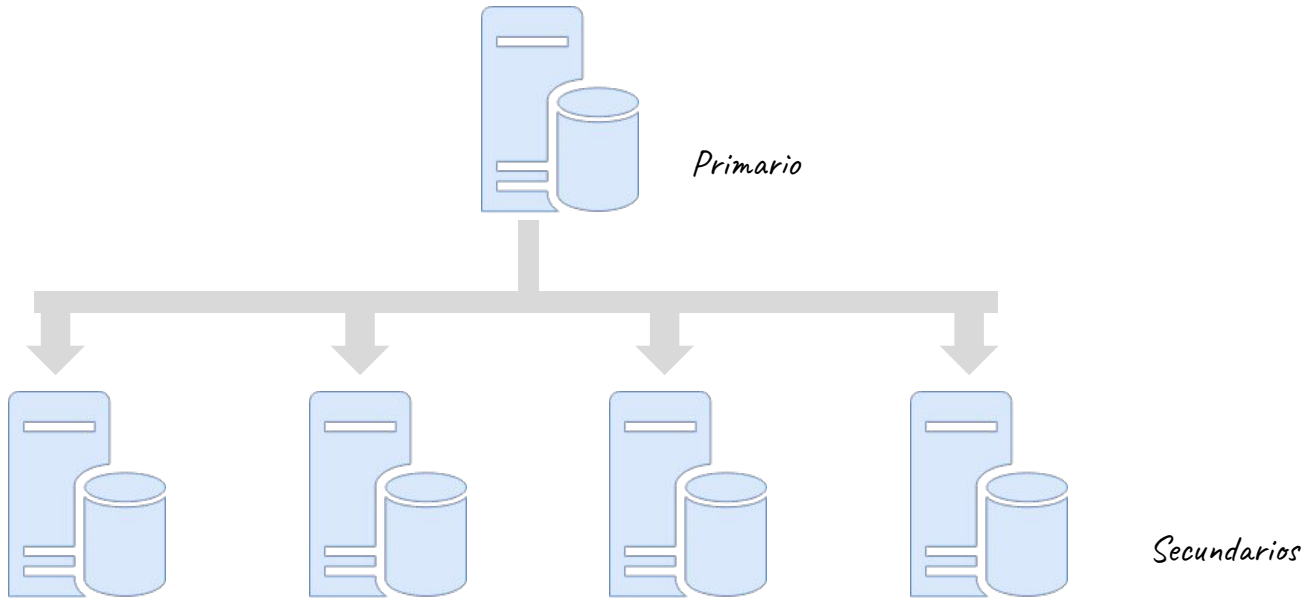
- No implica un gran problema para las aplicaciones, pues todo el cambio es sobre el hardware.
- Es mucho más fácil de implementar que el escalamiento horizontal.
- Puede ser una solución rápida y económica (comparado con modificar el software)

Inconvenientes

- El crecimiento está limitado por el hardware.
- No soporta la Alta disponibilidad.
- Hacer un upgrade del hardware al máximo pues llegar a ser muy caro, ya que las partes más nuevas suelen ser caras con respecto al rendimiento de un modelo anterior.

Rafael Garrote Hernández

Escalado Horizontal



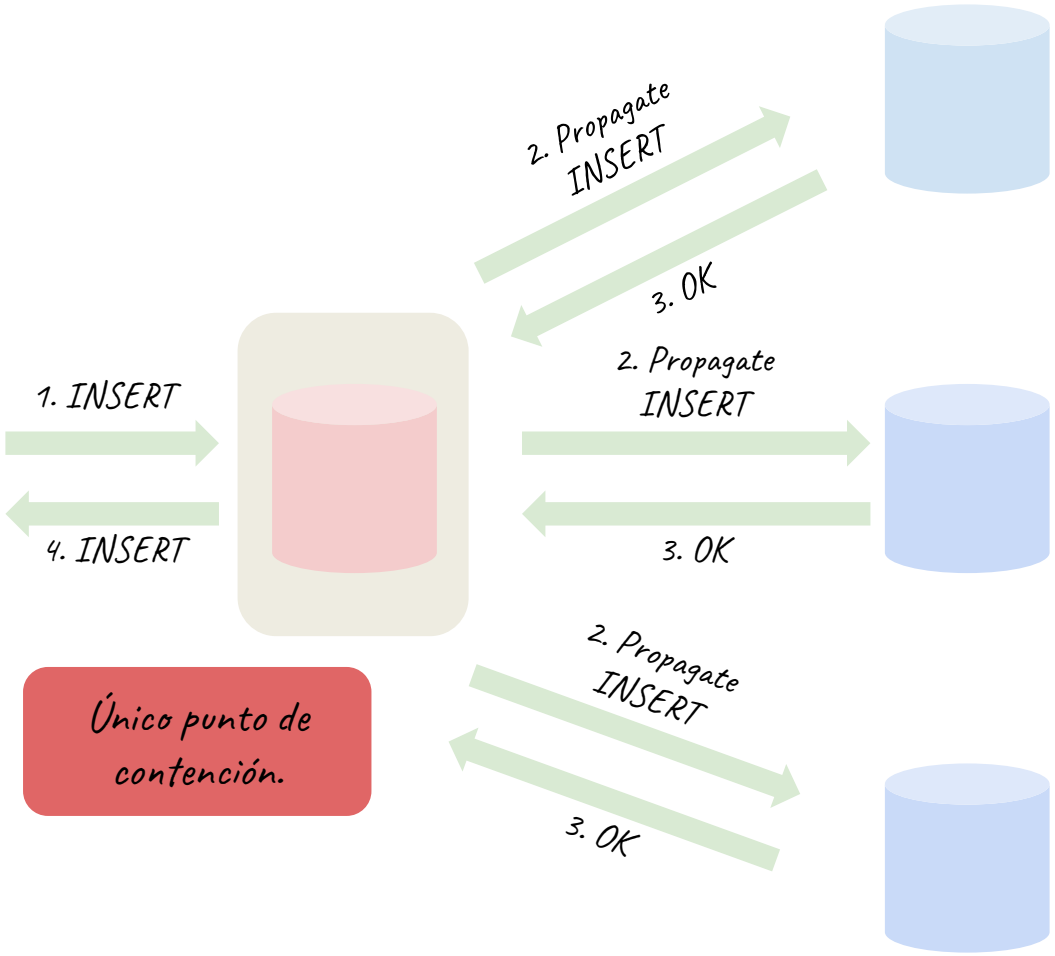
Escalado Horizontal

Ventajas

- El crecimiento es prácticamente infinito, podríamos agregar cuantos servidores sean necesarios.
- Es posible combinarse con el escalamiento vertical.
- Soporta la alta disponibilidad.
- Si un nodo falla, los demás sigue trabajando.
- Soporta el balanceo de cargas.

Inconvenientes

- Requiere de mucho mantenimiento.
- Es difícil de configurar.
- Requiere de grandes cambios en las aplicaciones (si no fueron diseñadas para trabajar en cluster).
- Requiere de una infraestructura más grande.


2. Propagate
INSERT
3. OK
1. INSERT
4. INSERT
2. Propagate
INSERT
3. OK
Único punto de
contención.
2. Propagate
INSERT
3. OK

Consistencia Estricta

- Todos los nodos tienen la misma copia de los datos.
- Todos los nodos tienen el mismo estado.
- Siempre devuelve el último valor escrito.

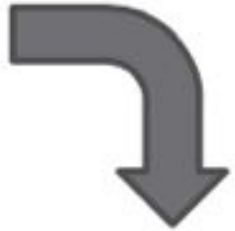
Bases de Datos Relacionales

- Guardan los datos en Tablas.
- Tienen Esquema.
- Lenguaje de consulta estándar SQL.
- Cumplen el Teorema ACID.
- Soportan Transacciones.
- Consistencia Estricta.

¿Problemas?

Esquemas Rígidos

```
CREATE TABLE Customers (
    Customer_Id Int,
    Name Varchar(100),
    ...
)
```



```
BULK INSERT Customers
FROM 'CustFile.txt'
WITH FIELDTERMINATOR = ';'
```

```
SELECT Customer_Id, Name
FROM Customers
```

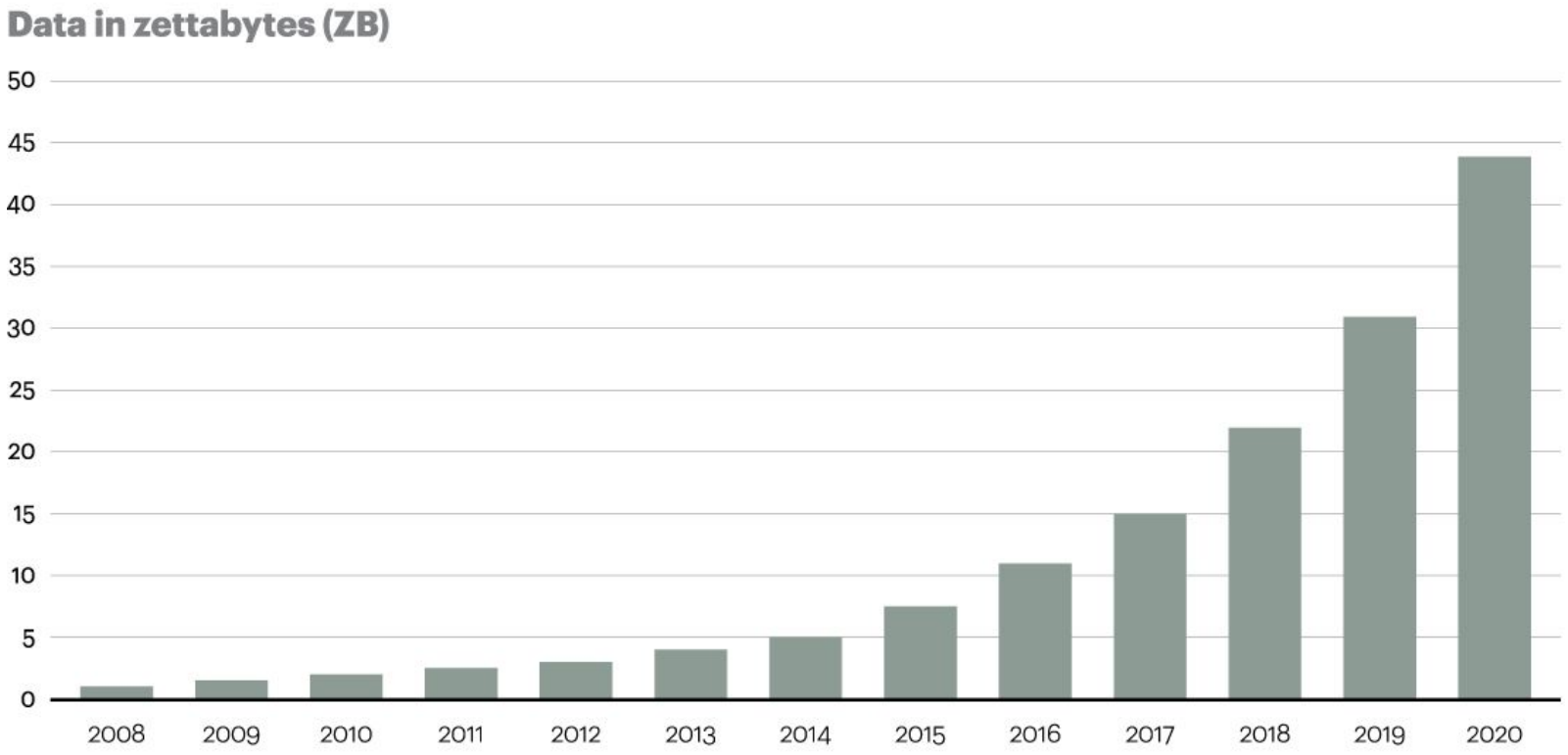
Esquemas Rígidos

- No se puede cargar la información hasta que no se cree la estructura en la base de datos.
- No se puede crear la estructura hasta que se comprenda el esquema que se va almacenar en la tabla.
- ¿Qué ocurre si los datos cambian?

Volúmenes de datos grandes

Figure 1

Data is growing at a 40 percent compound annual rate, reaching nearly 45 ZB by 2020



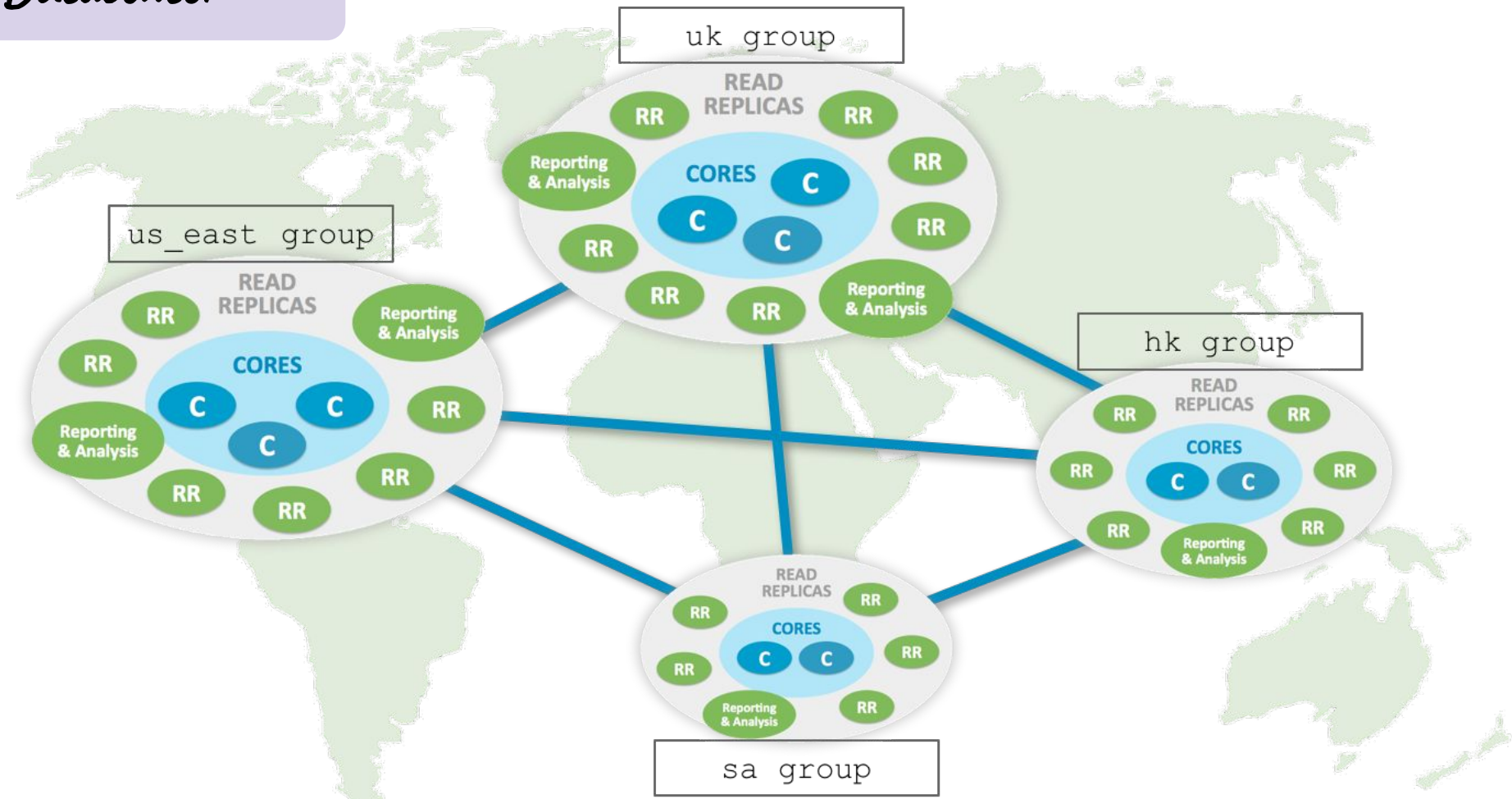
Source: Oracle, 2012

Escalabilidad

ESCALADO VERTICAL

ESCALADO HORIZONTAL

Multi-Datacenter

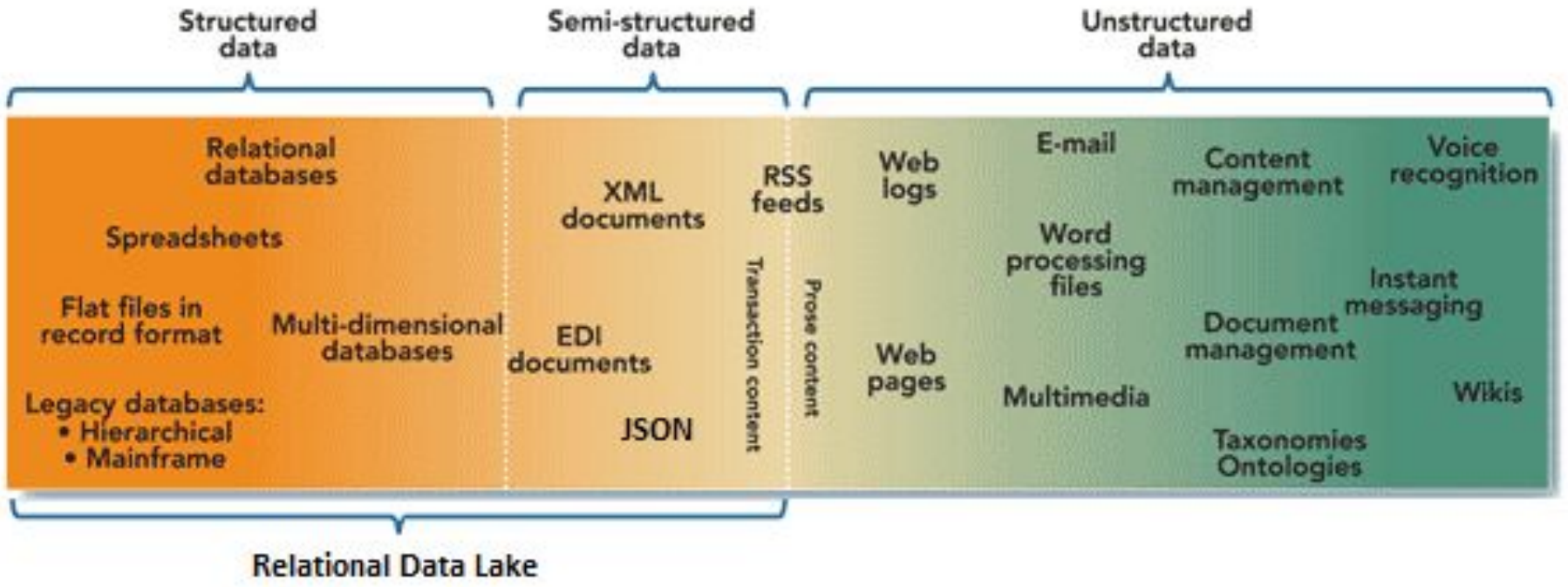


Rafael Garrote Hernández

Variedad de Formatos de la Información

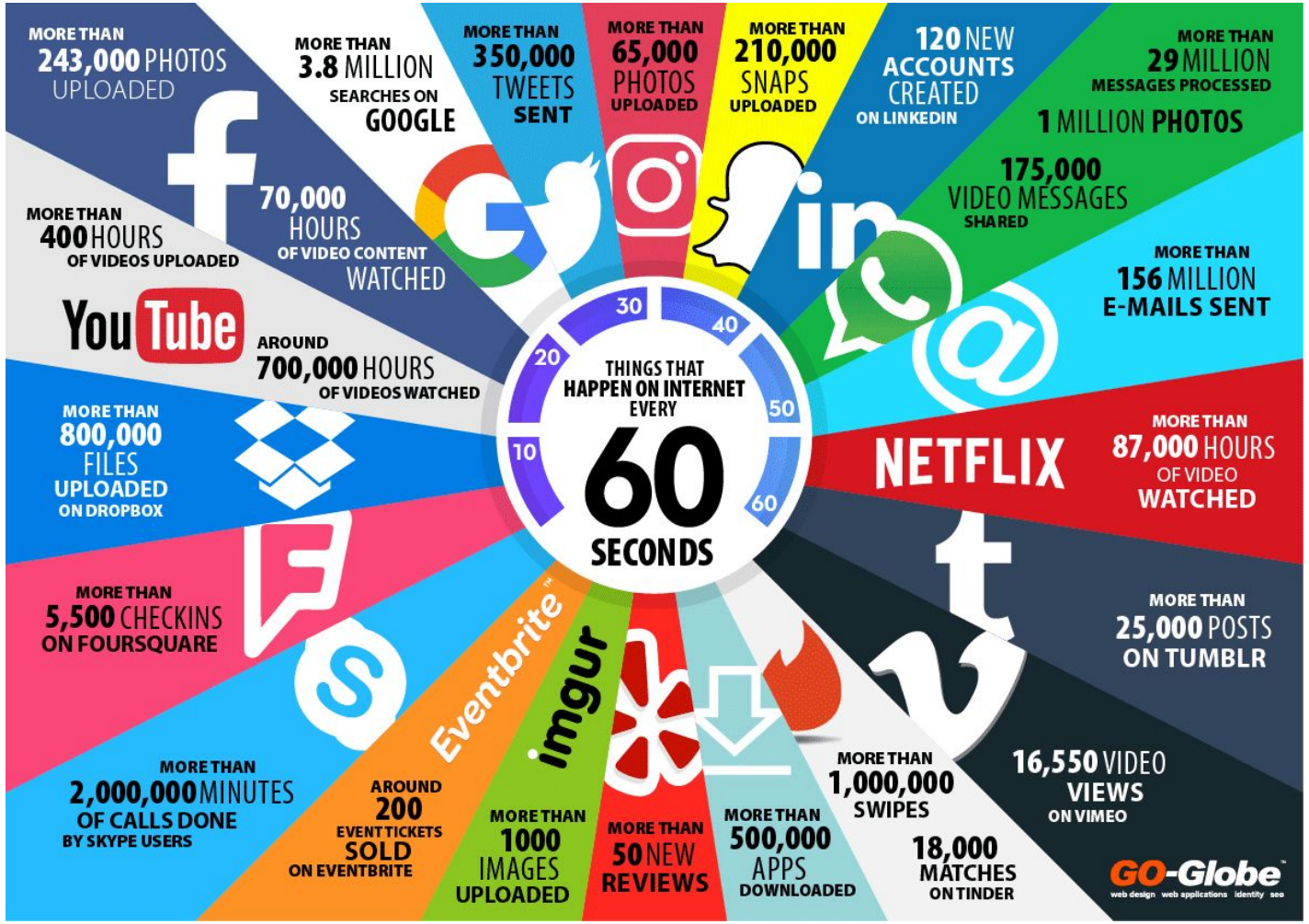
- Datos estructurados
  - Los datos dependen de un modelo de datos.
  - Los datos residen en un campo de un registro.
- Datos no estructurados
  - Los datos no son fáciles de encajar en un modelo de datos.
- Datos semi estructurados
  - Generados por máquinas.
- Basados en grafos.
- Audio, imagen, video.

Variedad de Formatos de la Información



Rafael Garrote Hernández

Velocidad



Rafael Garrote Hernández

Las 3-Vs del Big Data

Rafael Garrote Hernández

Datastores Distribuidos

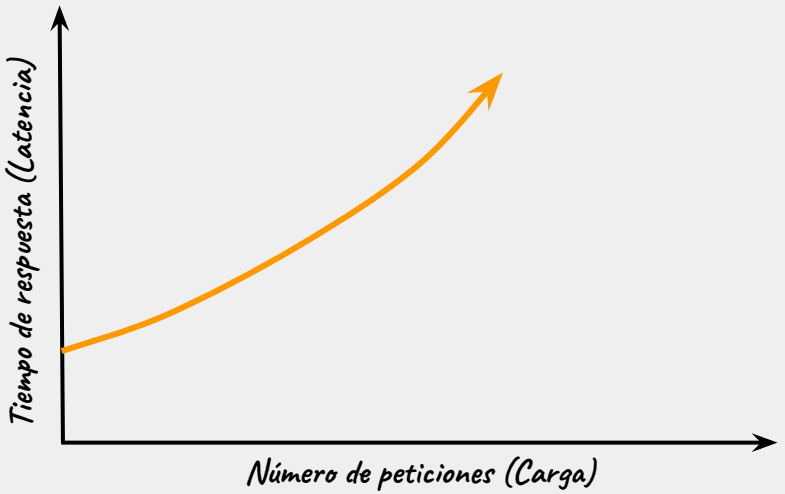
Rendimiento

El rendimiento mejora la Latencia en términos de tiempo.

Escalabilidad

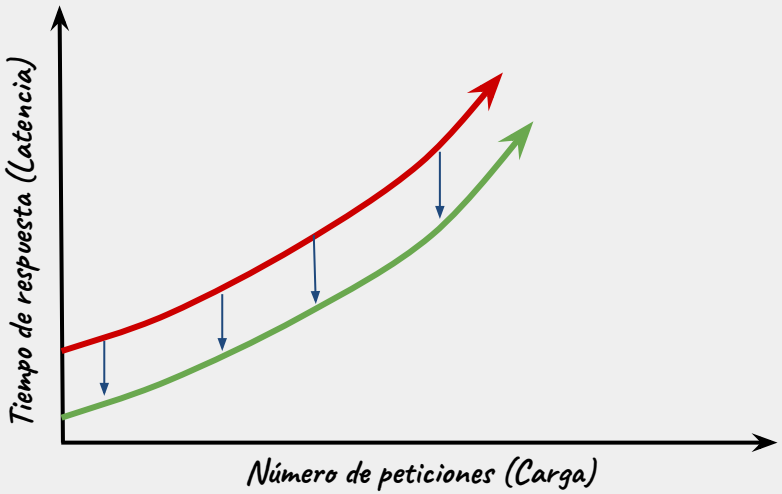
La Escalabilidad mejora la gestión de la Carga.

Escalabilidad Vs. Rendimiento



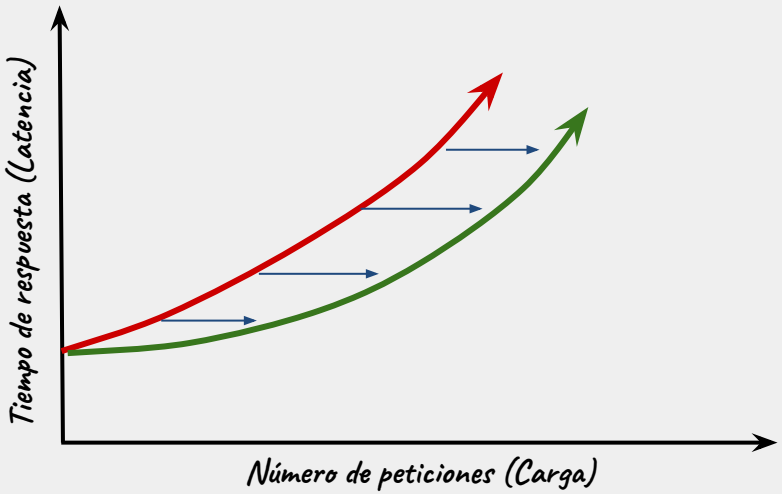
Escalabilidad y Rendimiento están realacionados pero son diferentes conceptos.

Rendimiento



- Mejora el rendimiento bajando el tiempo de respuesta (Latencia).
- El número total de peticiones (Carga) no tiene por que cambiar.
- Físicamente es imposible llegar a 0 y llegado a un límite bajar el tiempo es cada vez más costoso.

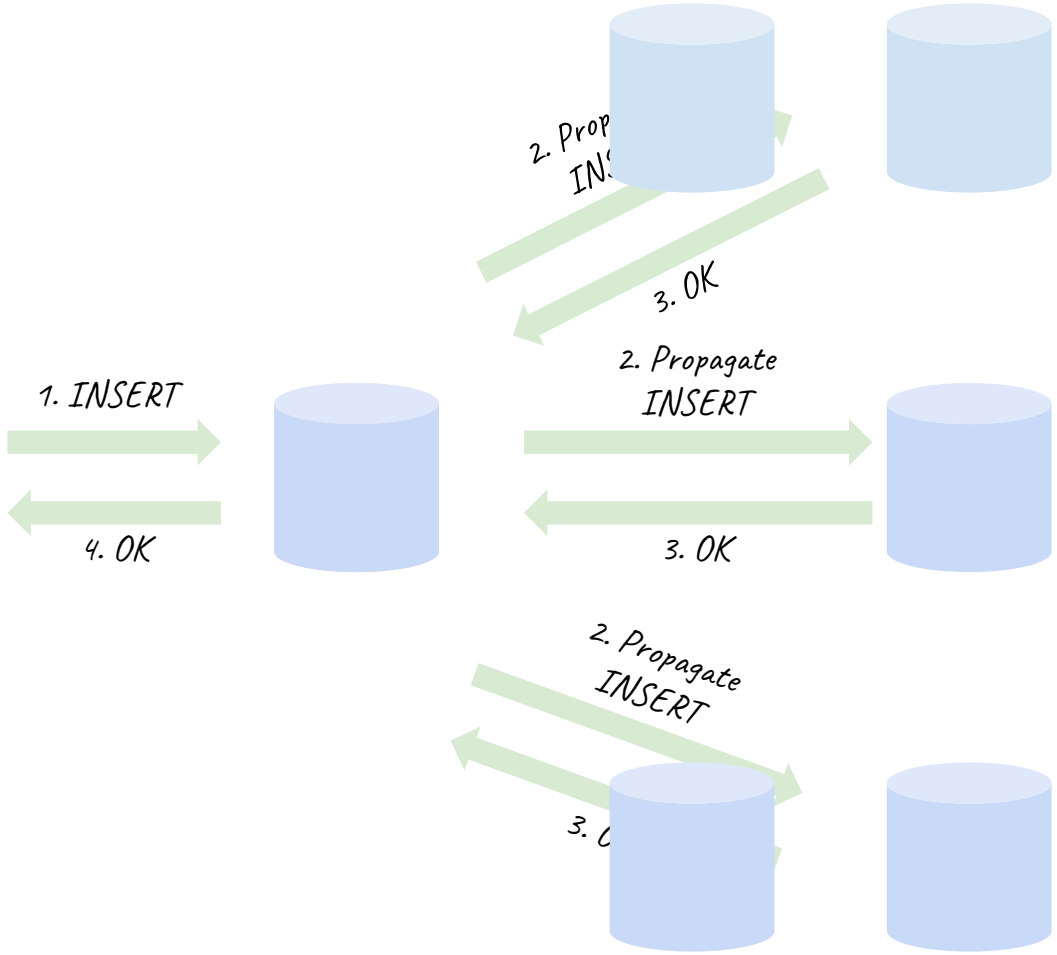
Escalabilidad

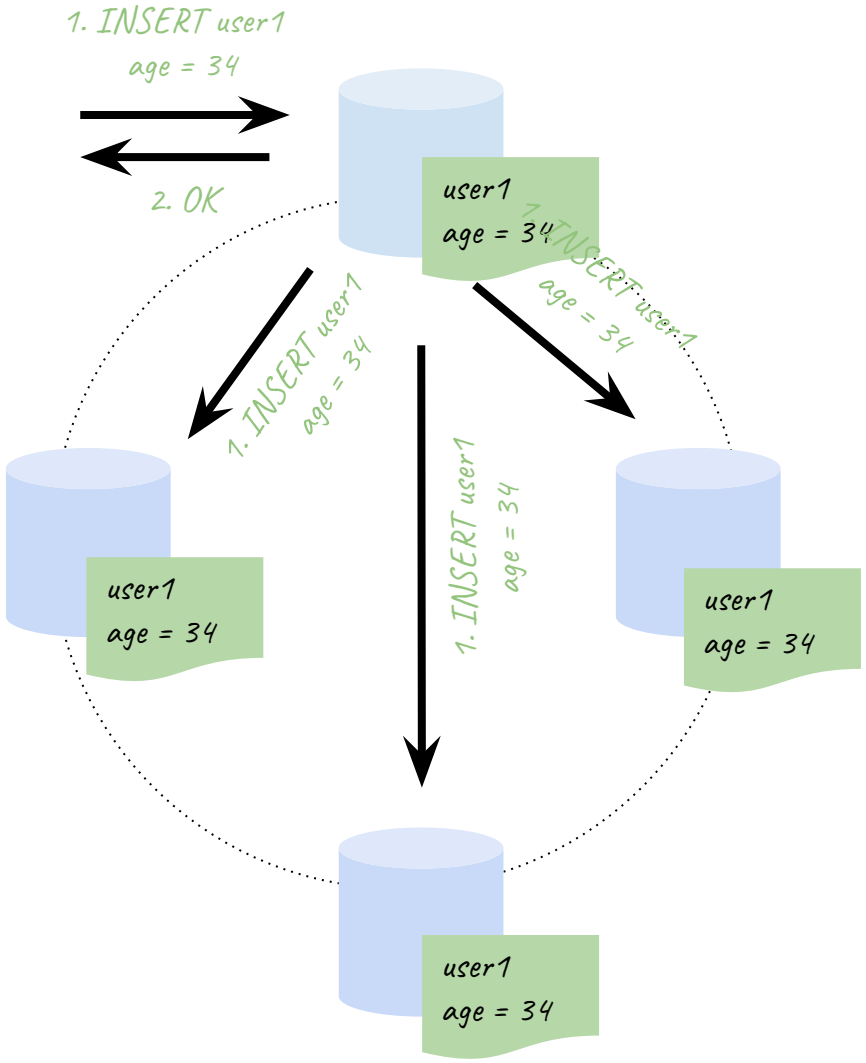


- Mejora la Escalabilidad incrementa la capacidad de gestionar la Carga.
- El Rendimiento de cada petición no tiene por que cambiar.
- La Escalabilidad no tiene límite teórico.

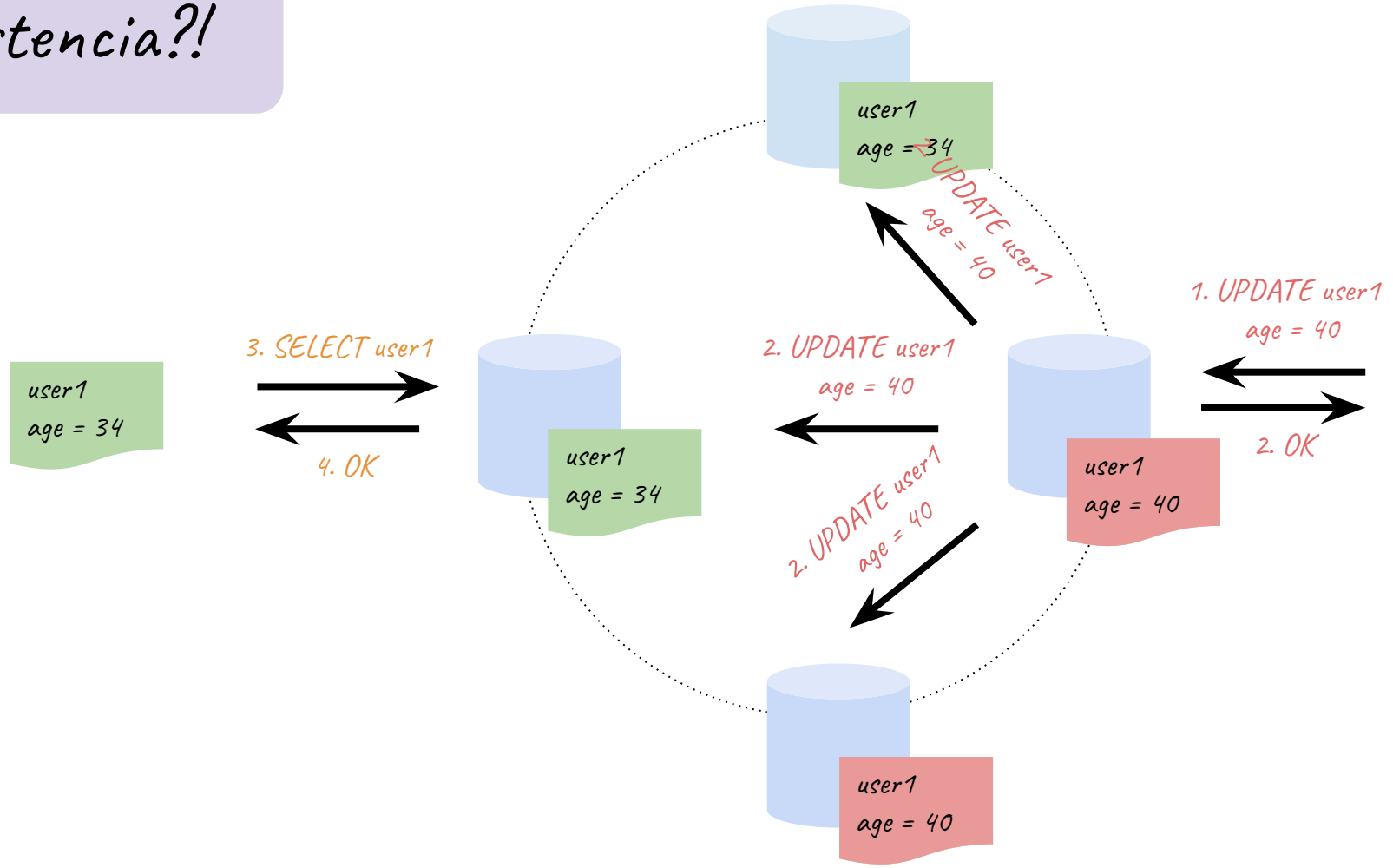
Sistemas Distribuidos

- Escalado horizontal.
- Todos los nodos son iguales, no hay nodos primarios ni nodos secundarios.
- Replican los datos.
- Tolerantes a fallos.
- No tienen un único punto de fallo.
- Alta disponibilidad
- Balanceo de carga.
- Particionado de los datos.


2. Prop
IN
3. OK
1. INSERT
2. Propagate
INSERT
4. OK
3. OK
2. Propagate
INSERT
3. C

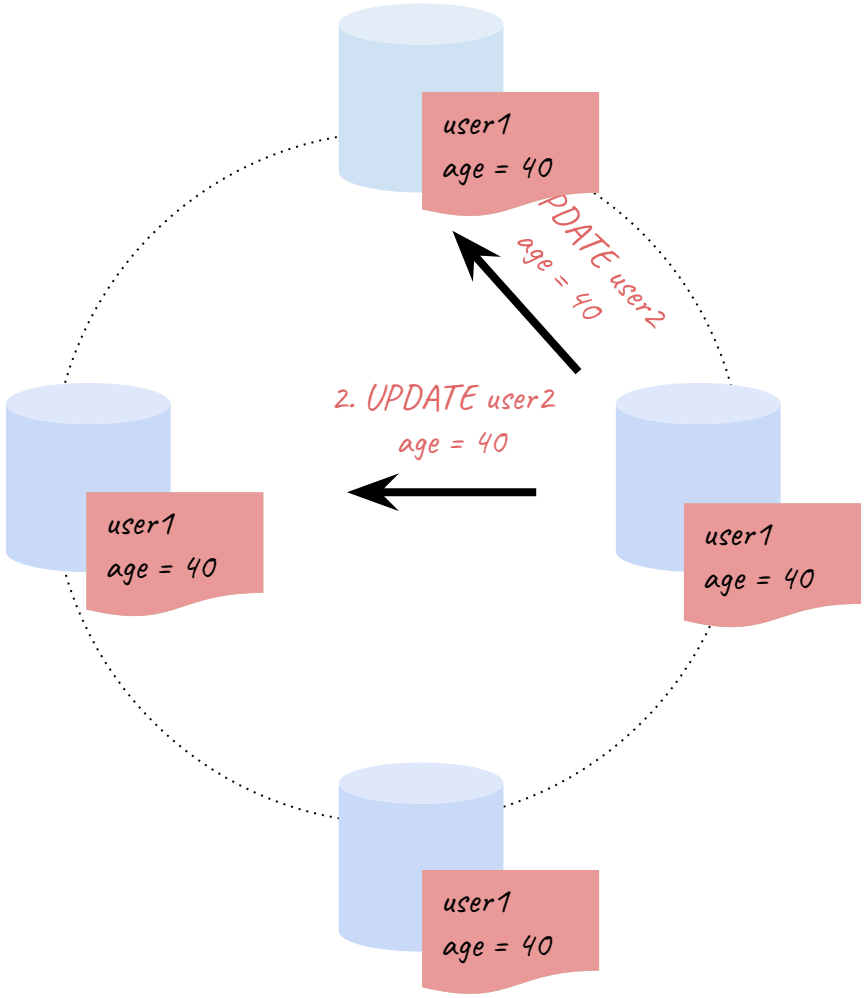

1. INSERT user1
age = 34
2. OK
user1
age = 34
1. INSERT user1
age = 34
1. INSERT user1
age = 34
1. INSERT user1
age = 34
user1
age = 34
user1
age = 34
user1
age = 34

¡¿Consistencia?!



Rafael Garrote Hernández

Eventualmente Consistente

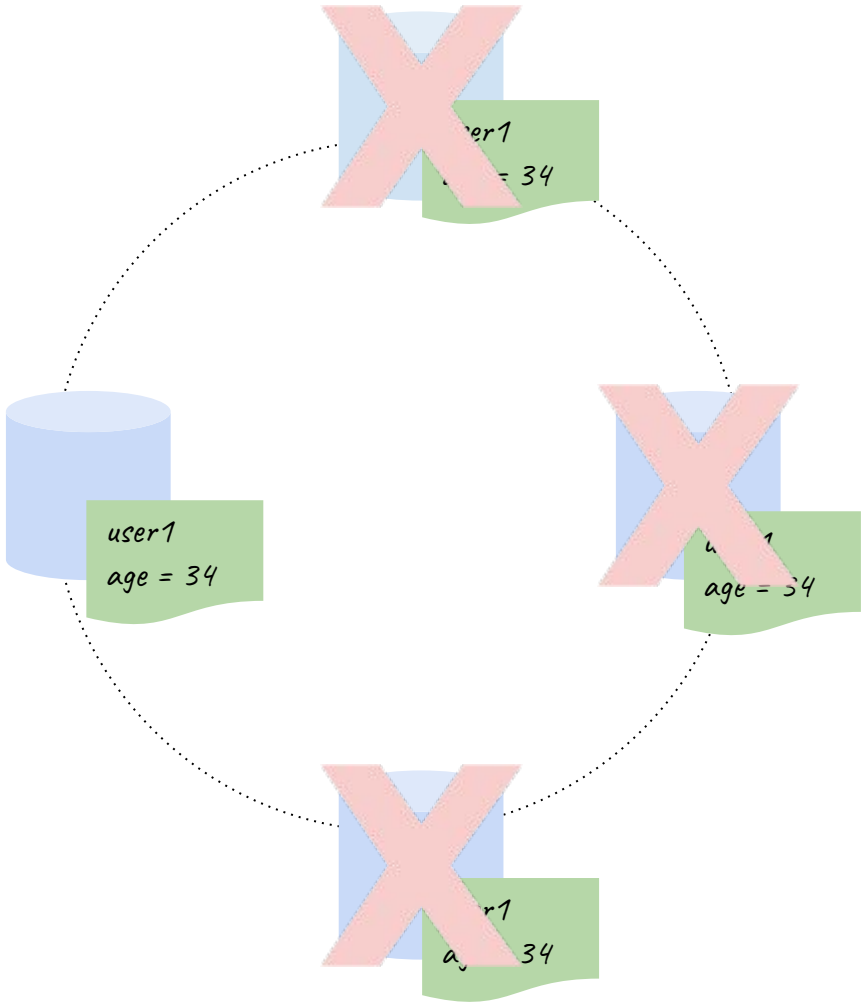


Coherency Delay

- En un sistema distribuido, sincronizar el estado de los múltiples nodos se hace utilizando un protocolo Crosstalk o Gossip.
- Cada nodo en el sistema enviará un mensaje a cada nodo informándoles cualquier cambio en el estado.
- El tiempo que toma la sincronización completa se llama Coherency Delay.
- Incrementar el número de nodos incrementa la Coherency Delay.

No S.P.O.F.

Tolerantes a Fallos



Teorema BASE

Basic Availability

El sistema funciona incluso cuando alguna parte falla.
El almacenamiento sigue los principios de distribución y replicación.

Soft State

Los nodos no tienen por que ser consistentes entre sí todo el tiempo.
No tienen por que tener la misma copia de los datos.

Eventual Consistency

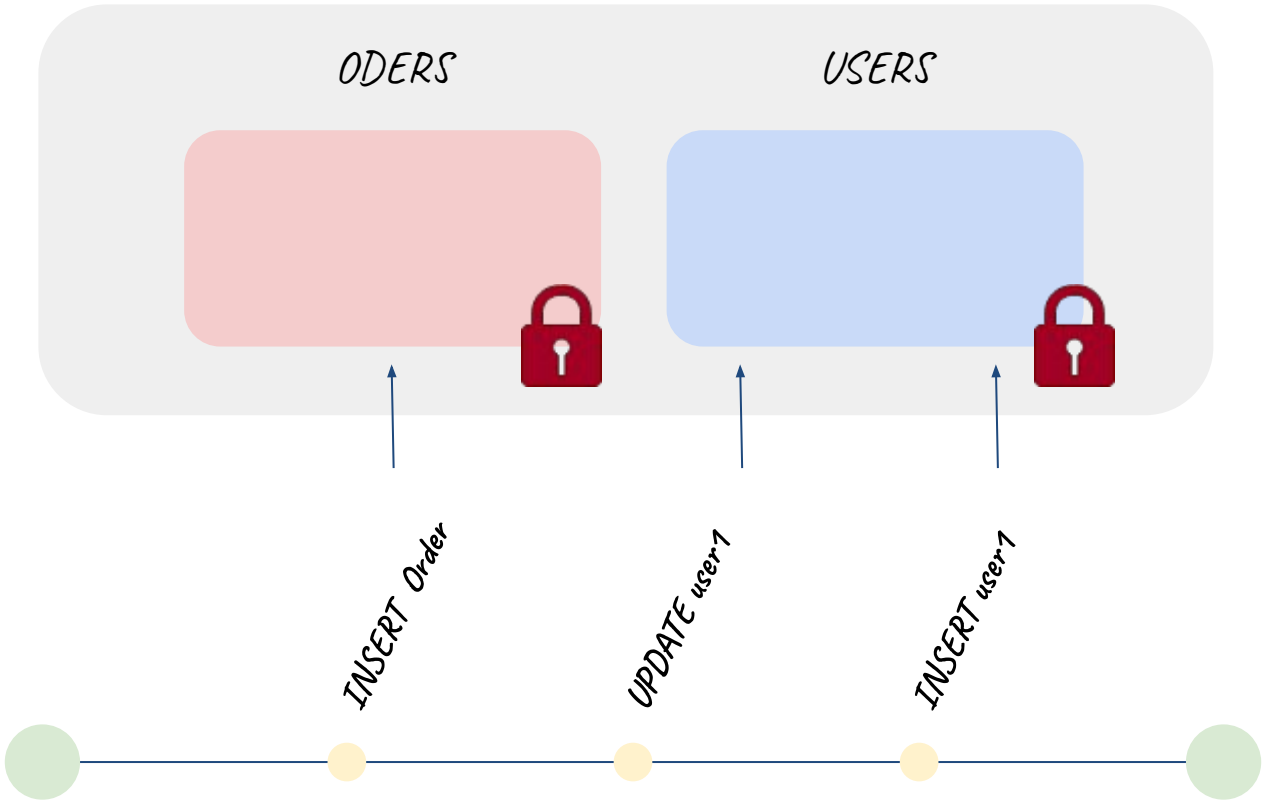
Si se dejan de procesar peticiones de modificación,
la consistencia de los datos entre nodos se consigue eventualmente.

Rafael Garrote Hernández

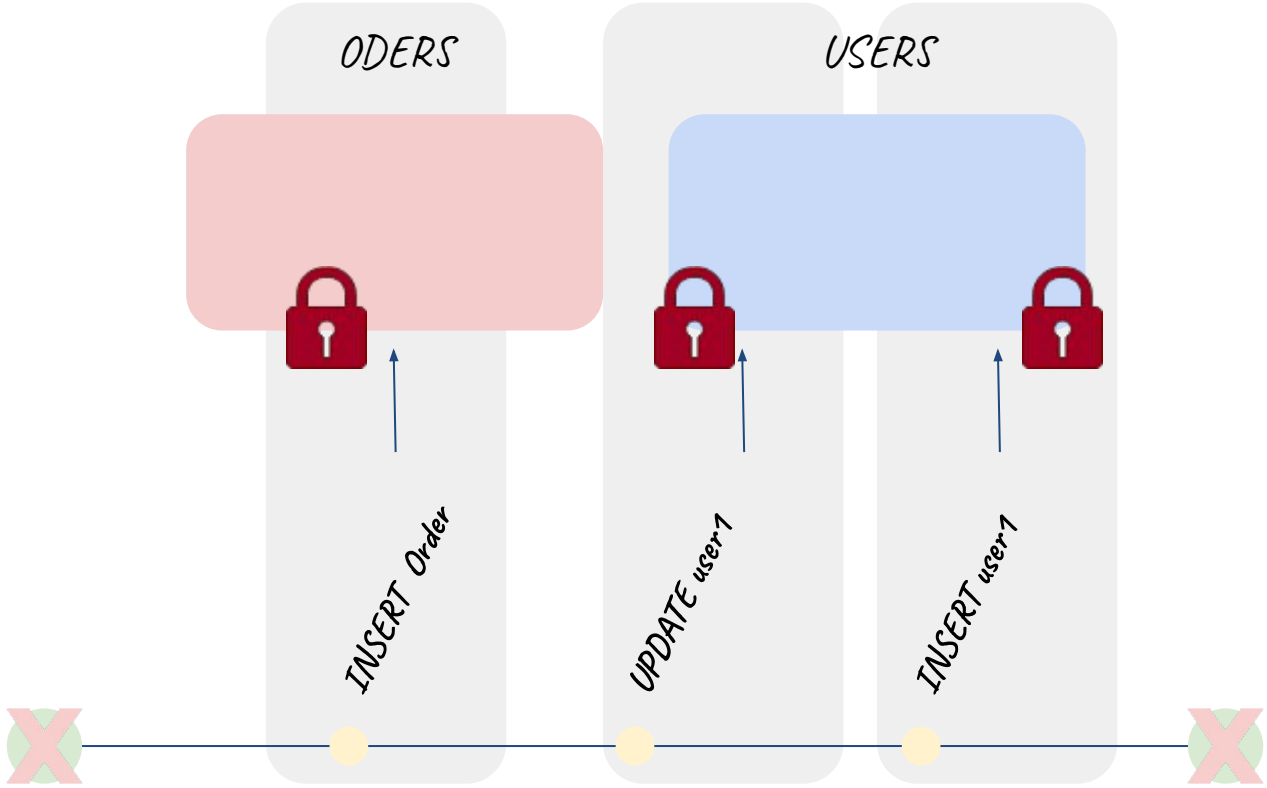
ACID Vs. BASE

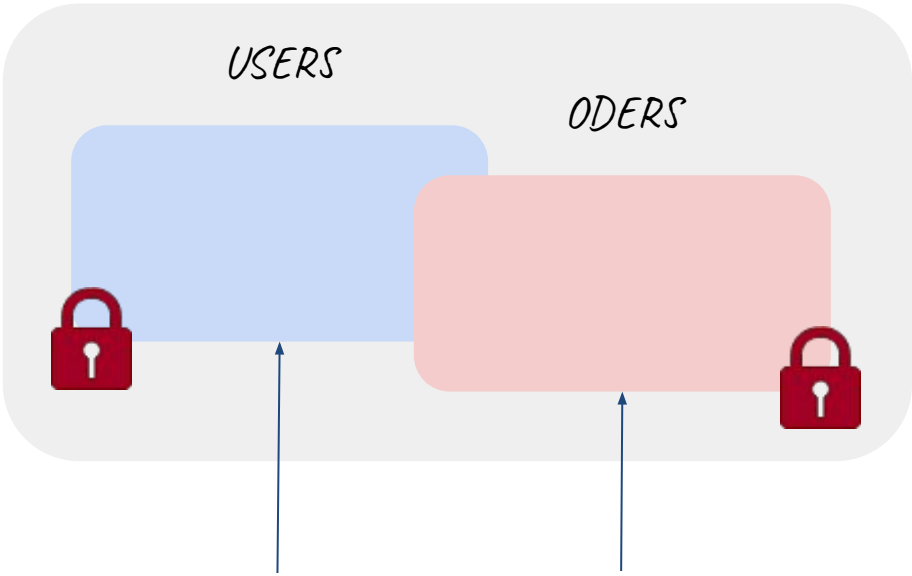
- Ganamos disponibilidad.
  - Tolerantes a fallos.
  - No tienen un único punto de fallo.
  - Alta disponibilidad.
- Ganamos eficiencia.
  - Balanceo de carga.
  - Eliminamos contención de las operaciones distribuidas.

- Perdemos consistencia.
  - El sistema no devuelve el dato más actual.
- Perdemos aislamiento.
  - Las operaciones tienen efecto de lado en el resto de nodos.

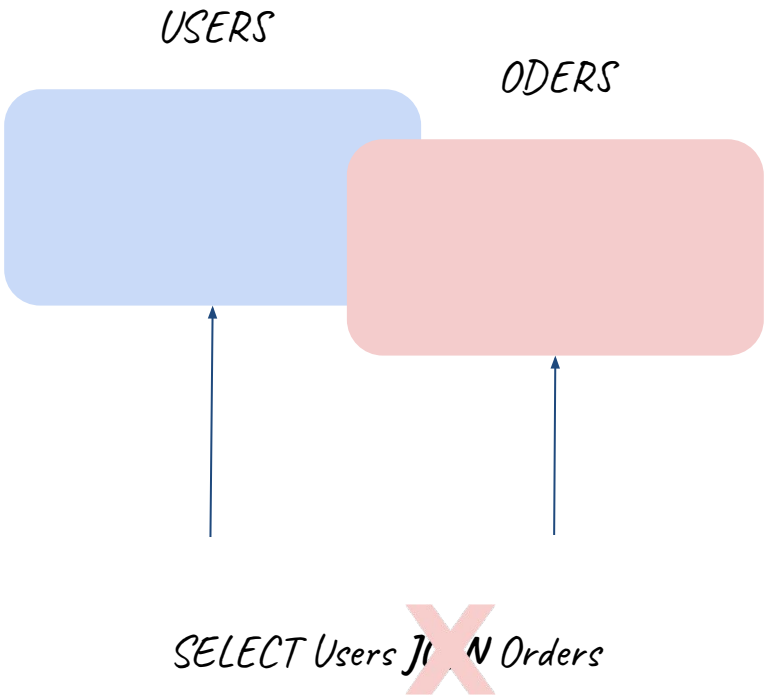

ODERS
USERS
INSERT Order
UPDATE user1
INSERT user1

Aislar la Contención

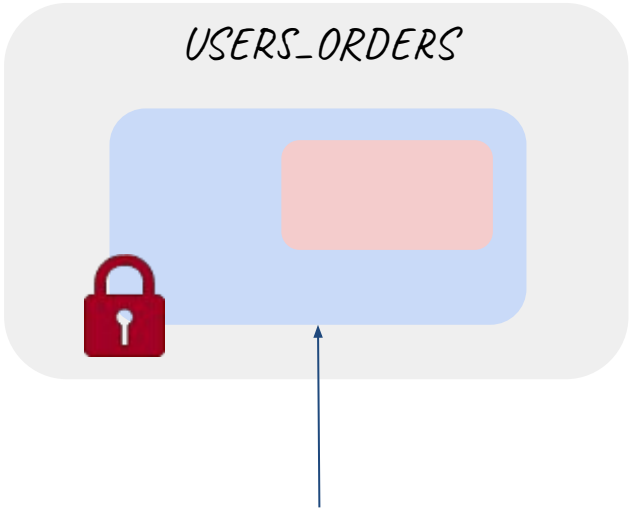


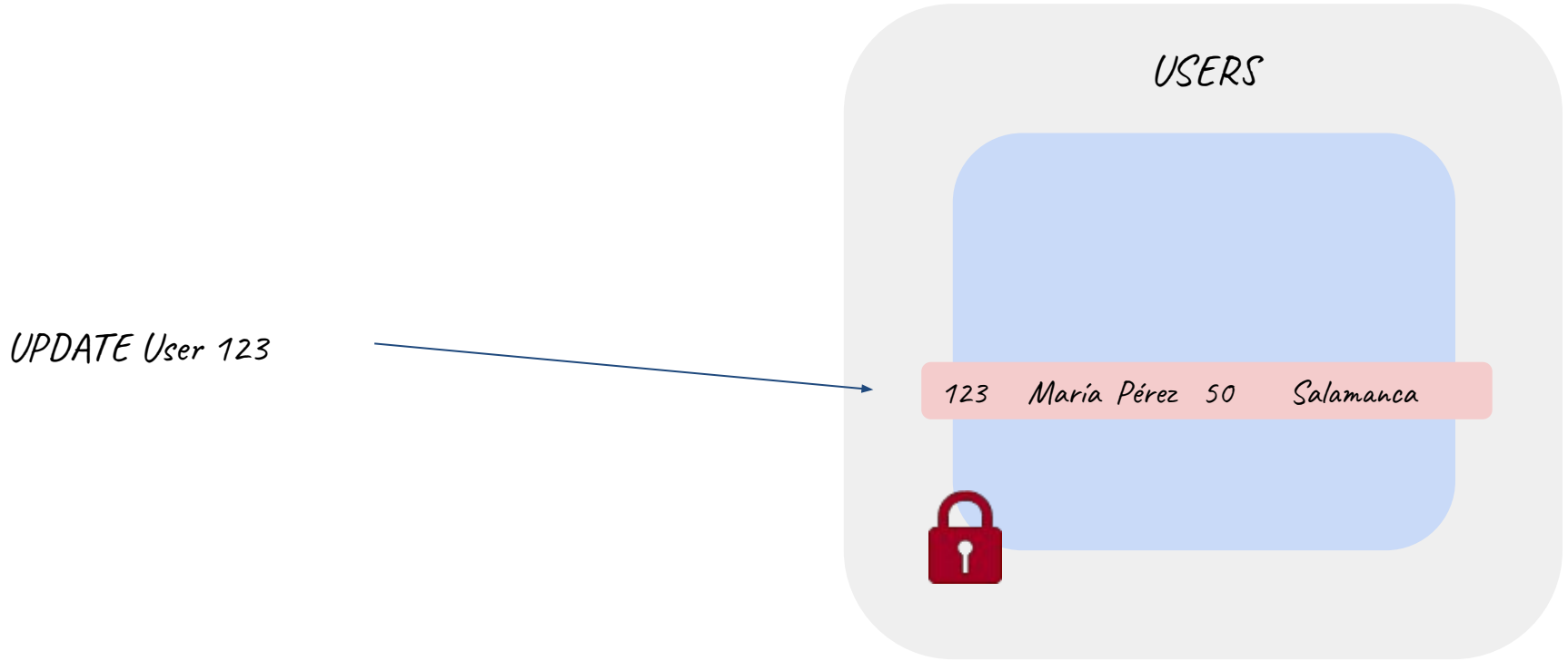


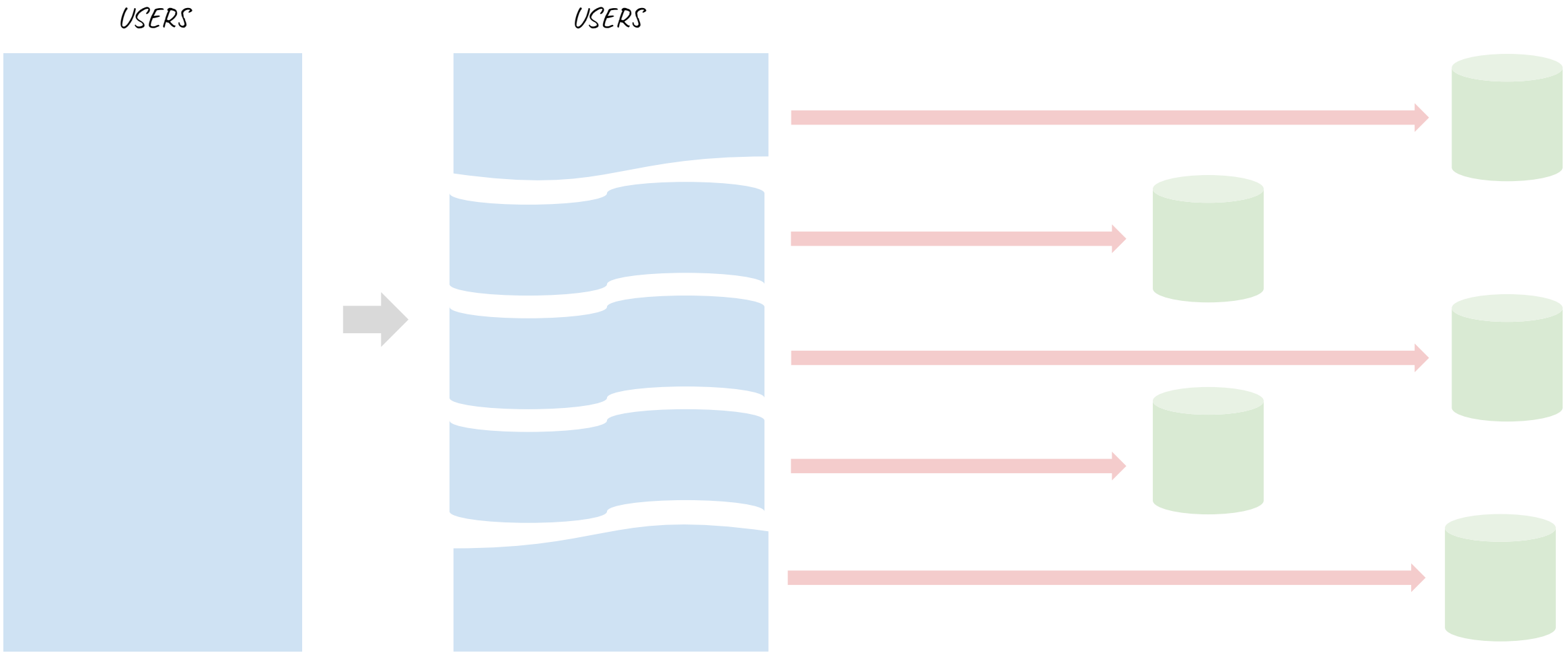
SELECT Users **JOIN** Orders


USERS
ODERS
SELECT Users JOIN Orders

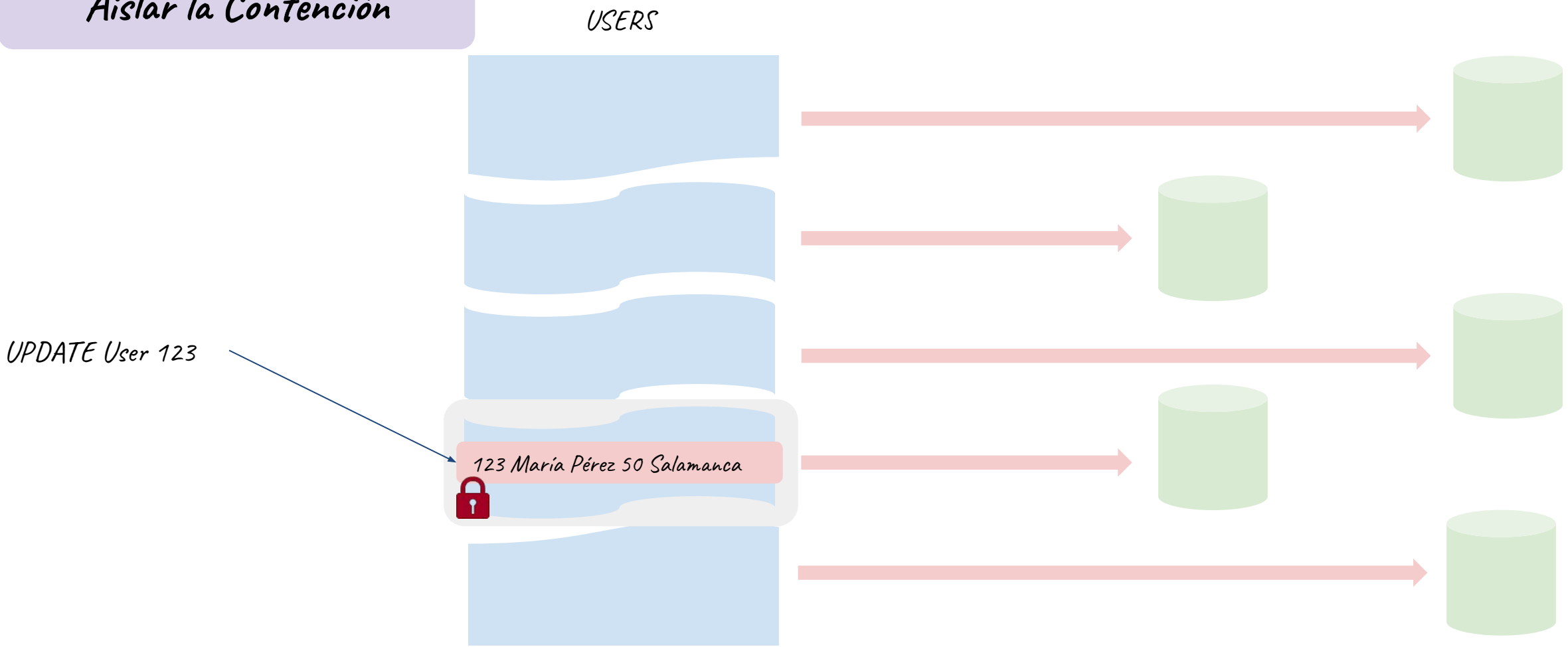
Aislar la Contención




USERS
UPDATE User 123
123 María Pérez 50 Salamanca


USERS
USERS

Aislar la Contención

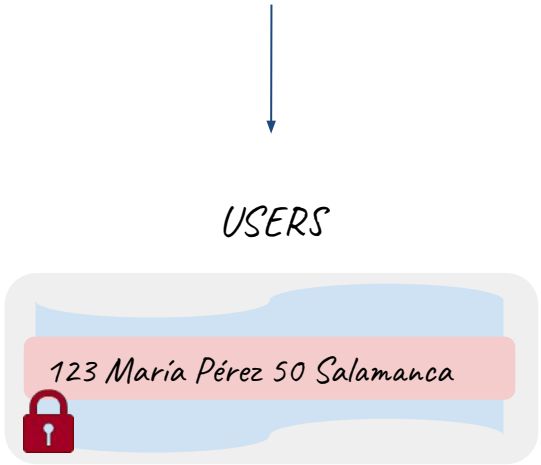


Rafael Garrote Hernández

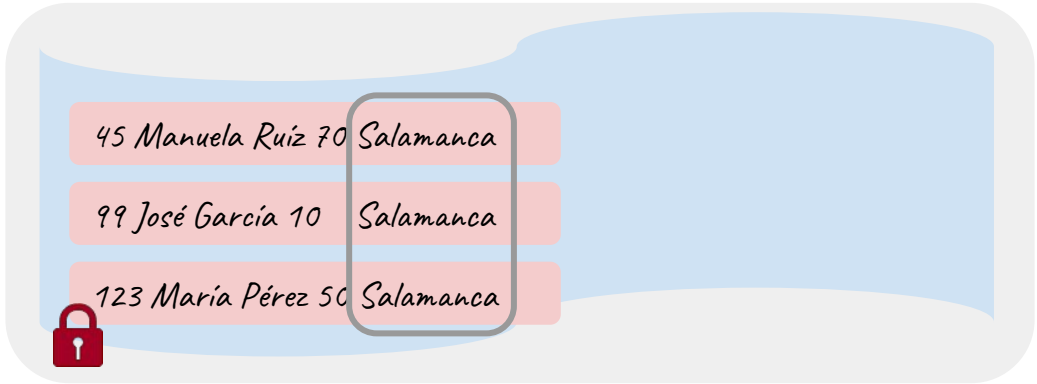
Sharding

```
SELECT avg(age)
FROM Users
WHERE City = 'Salamanca'
```

UPDATE User 123



Random Keys

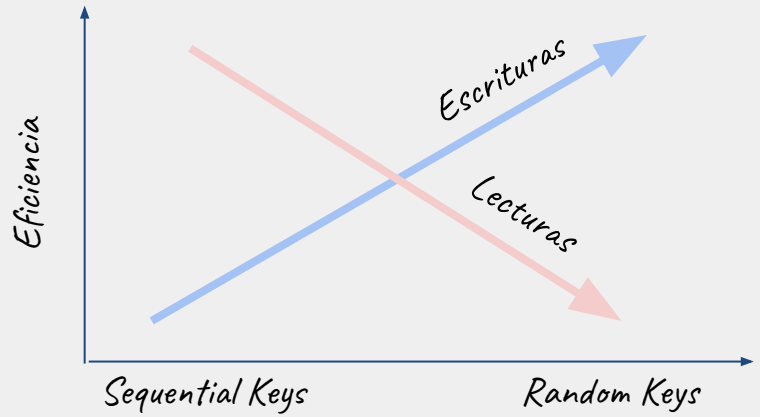


Sequential Keys

Contención

Rafael Garrote Hernández

Sharding



Particionar los datos de forma que la información que es frecuentemente accedida a la vez se encuentre en la misma partición.

Sharding

ID	nombre	edad	sexo	ciudad
1	María	22	M	Madrid
2	Luis	50	H	Barcelona
3	José	35	H	Salamanca
4	Lucía	65	M	Madrid
5	Ramón	12	H	Barcelona
6	Rafael	36	H	Barcelona
7	Juán	76	H	Madrid
8	Julia	32	M	Madrid
9	Luisa	54	M	Barcelona
10	Antonio	40	H	Zaragoza

Shard Key = ID

1	María	22	M	Madrid
4	Lucía	65	M	Madrid
7	Juán	76	H	Madrid
10	Antonio	40	H	Zaragoza

2	Luis	50	H	Barcelona
5	Ramón	12	H	Barcelona
8	Julia	32	M	Madrid

3	José	35	H	Salamanca
6	Rafael	36	H	Barcelona
9	Luisa	54	M	Barcelona

Rafael Garrote Hernández

Sharding

Cardinalidad

ID	nombre	edad	sexo	ciudad
1	María	22	M	Madrid
2	Luis	50	H	Barcelona
3	José	35	H	Salamanca
4	Lucía	65	M	Madrid
5	Ramón	12	H	Barcelona
6	Rafael	36	H	Barcelona
7	Juán	76	H	Madrid
8	Julia	32	M	Madrid
9	Luisa	54	M	Barcelona
10	Antonio	40	H	Zaragoza

Shard Key = **sexo**

1	María	22	M	Madrid
4	Lucía	65	M	Madrid
8	Julia	32	M	Madrid
9	Luisa	54	M	Barcelona

2	Luis	50	H	Barcelona
3	José	35	H	Salamanca
5	Ramón	12	H	Barcelona
6	Rafael	36	H	Barcelona
7	Juán	76	H	Madrid
10	Antonio	40	H	Zaragoza

Rafael Garrote Hernández

Sharding

Frecuencia

ID	nombre	edad	sexo	ciudad
1	María	22	M	Madrid
2	Luis	50	H	Barcelona
3	José	35	H	Salamanca
4	Lucía	65	M	Madrid
5	Ramón	12	H	Barcelona
6	Rafael	36	H	Barcelona
7	Juán	76	H	Madrid
8	Julia	32	M	Madrid
9	Luisa	54	M	Barcelona
10	Antonio	40	H	Zaragoza

Shard Key = **ciudad**

1	María	22	M	Madrid
4	Lucía	65	M	Madrid
7	Juán	76	H	Madrid
8	Julia	32	M	Madrid

3	José	35	H	Salamanca
10	Antonio	40	H	Zaragoza

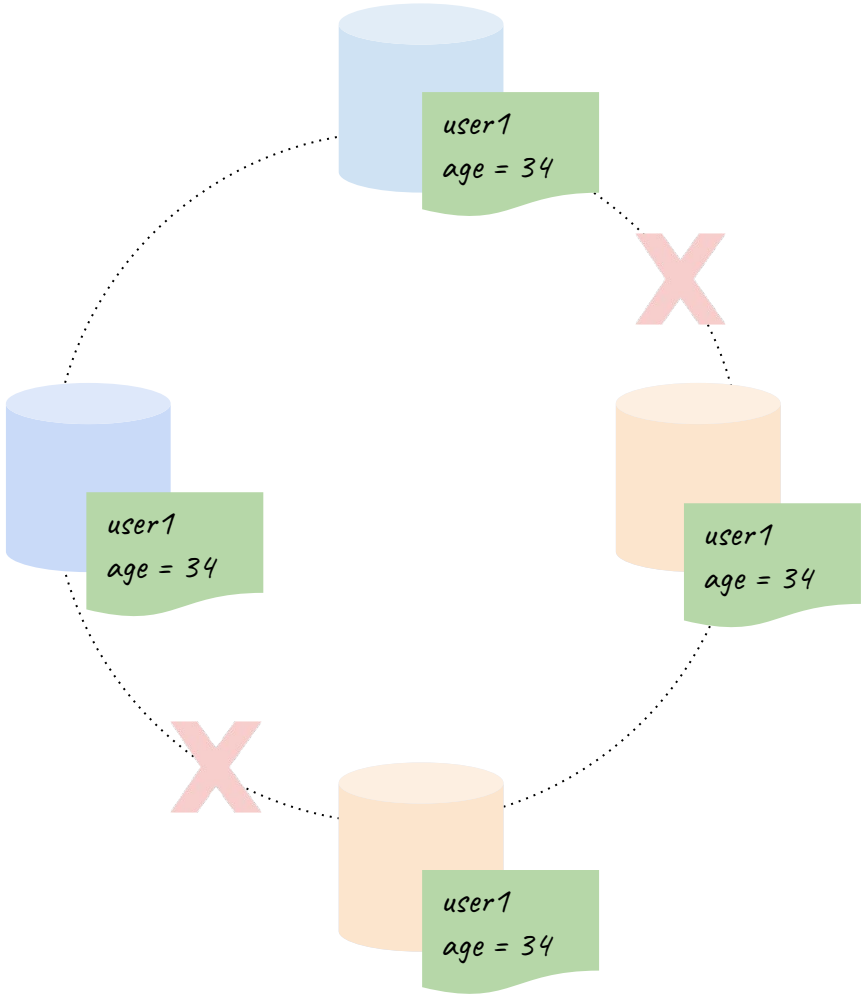
2	Luis	50	H	Barcelona
5	Ramón	12	H	Barcelona
6	Rafael	36	H	Barcelona
9	Luisa	54	M	Barcelona

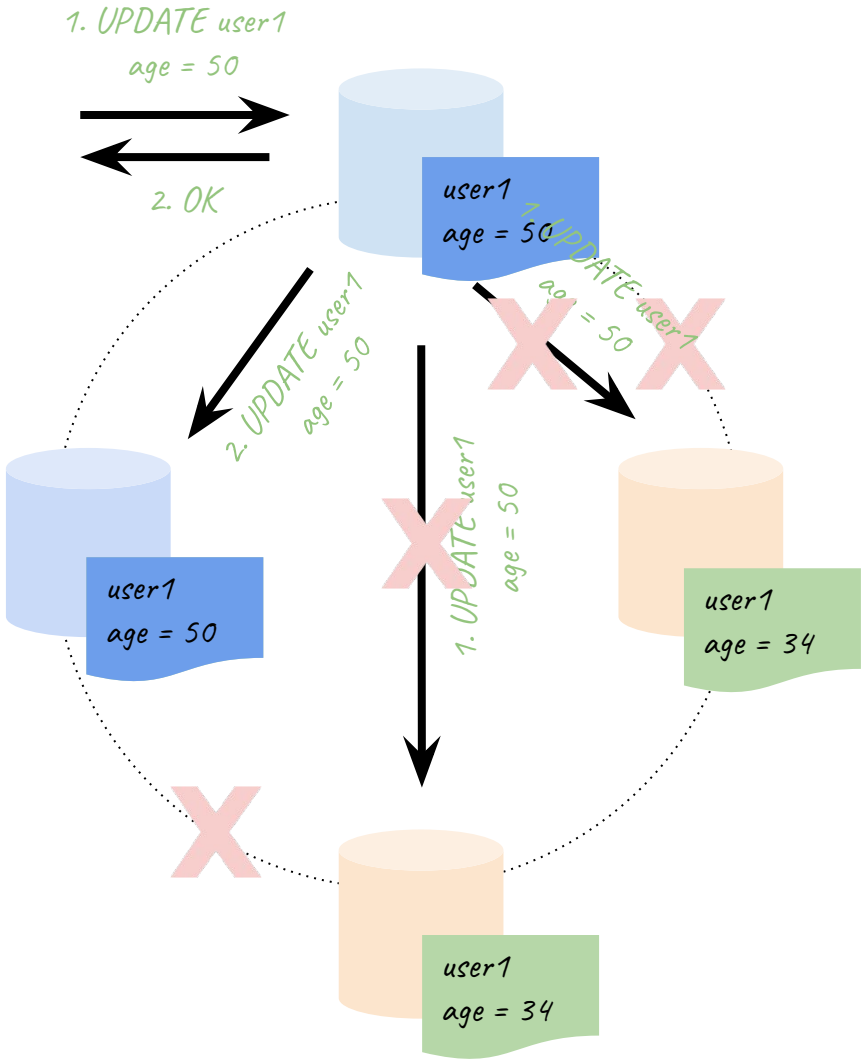
Rafael Garrote Hernández

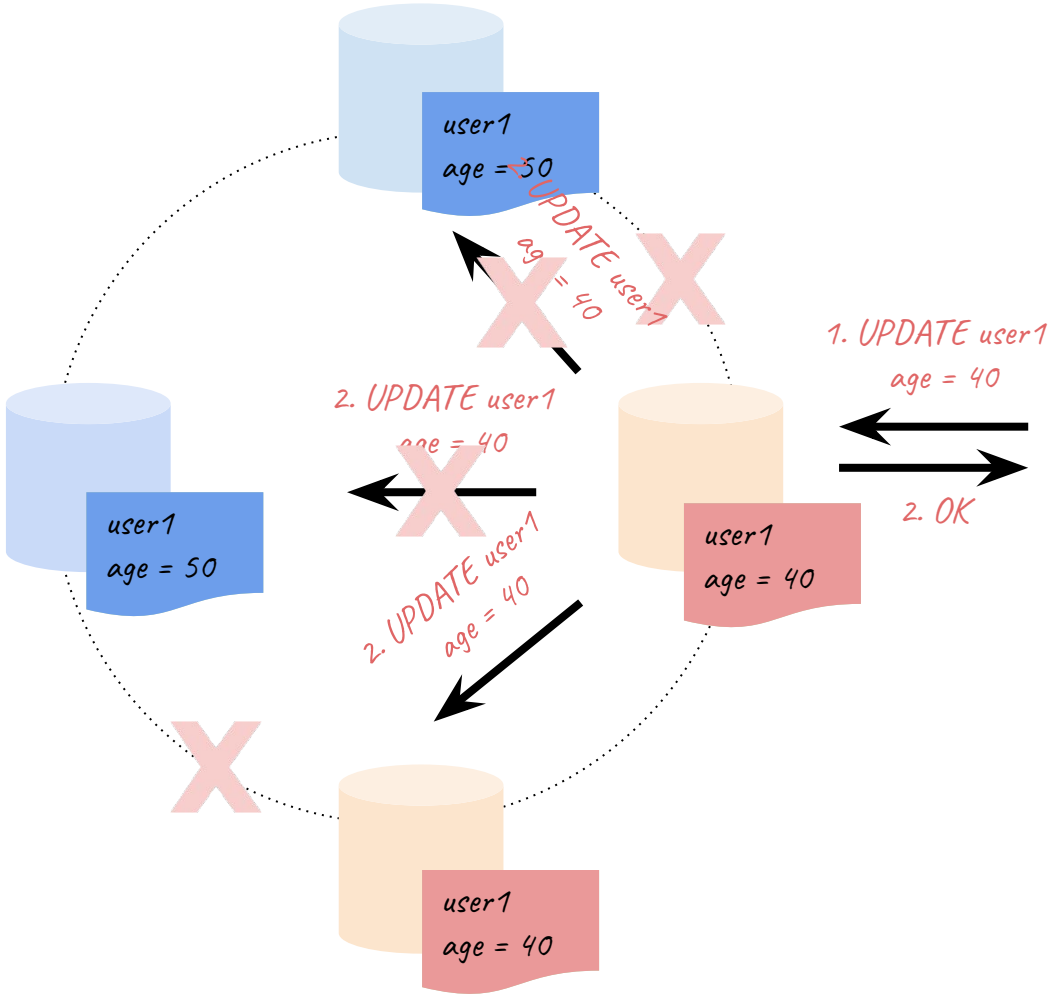
Aislar la Contención

- No permiten la operación JOIN.
- No tienen transacciones.
- Soportan Sharding de los datos.

Particionado de Red

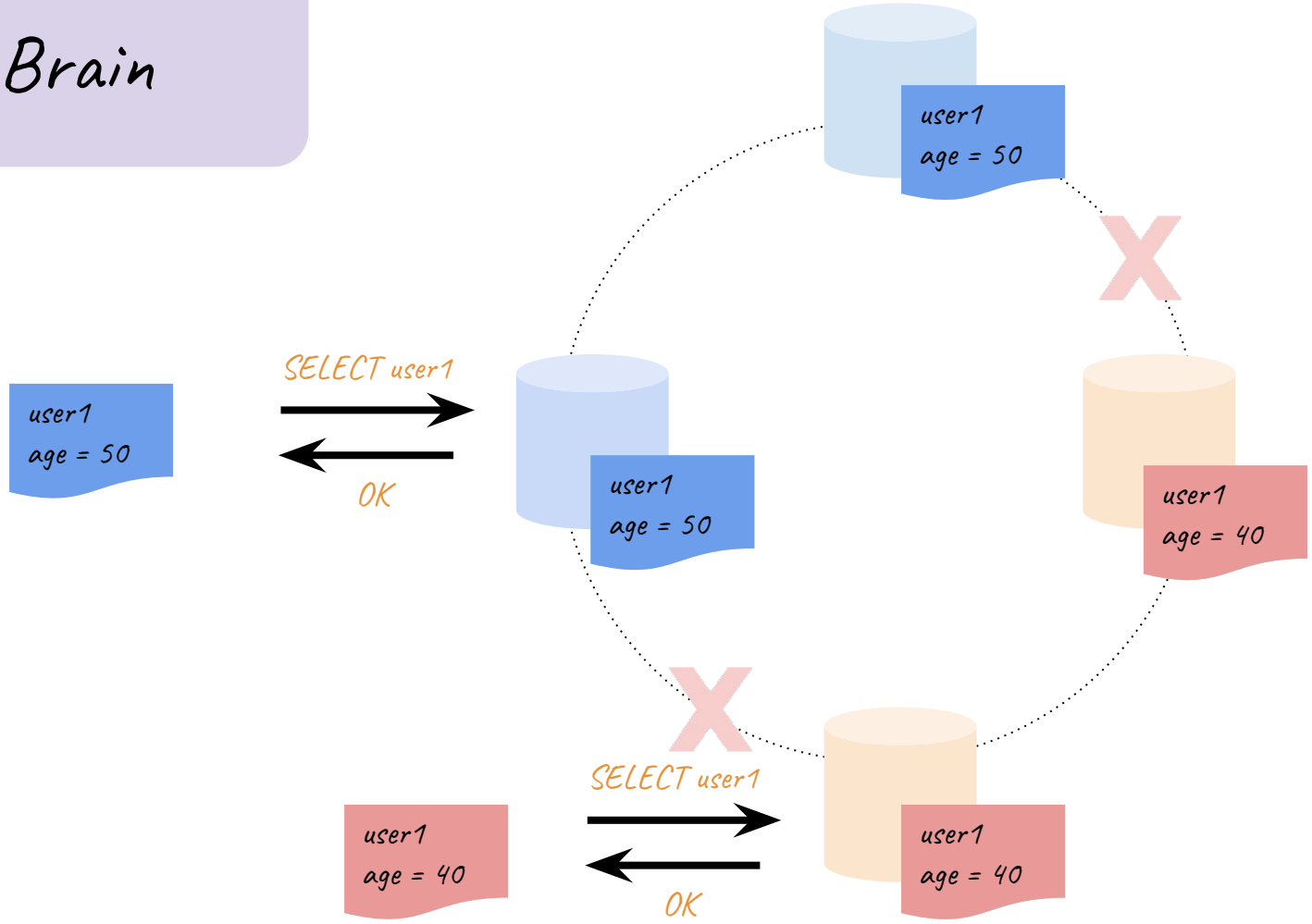



1. UPDATE user1
age = 50
2. OK
user1
age = 50
2. UPDATE user1
age = 50
1. UPDATE user1
age = 50
1. UPDATE user1
age = 50
user1
age = 50
user1
age = 34
user1
age = 34



Rafael Garrote Hernández

Split Brain



Rafael Garrote Hernández

Teorema CAP

Consistency

La base de datos es consistente después de cada operación.
Todos los clientes ven los mismos datos.

Availability

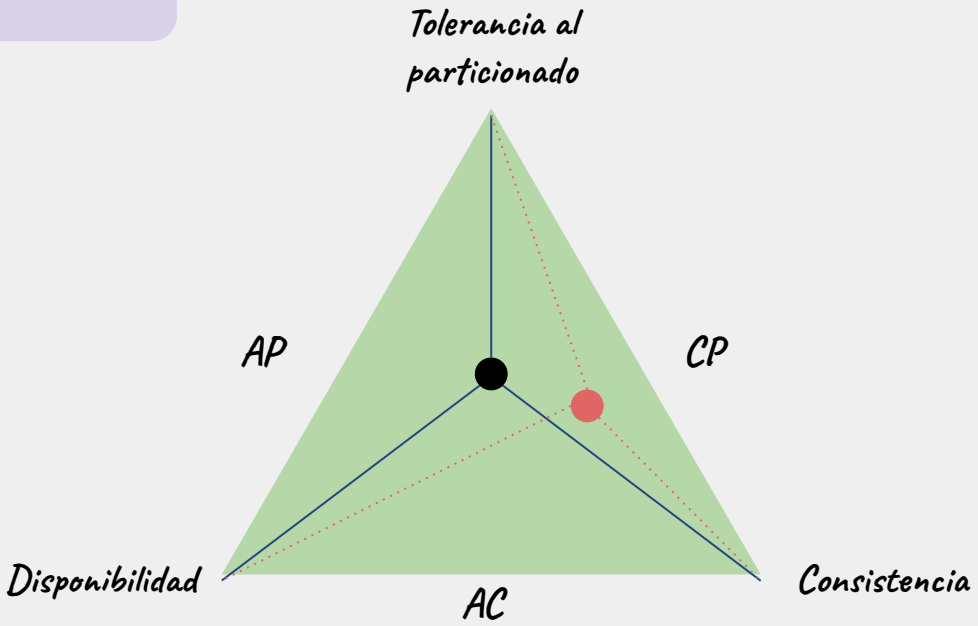
El sistema siempre está disponible.
Sin downtime.

Partition Tolerance

El sistema sigue funcionando aunque se produzca un error de comunicación entre los nodos.

Rafael Garrote Hernández

Teorema CAP

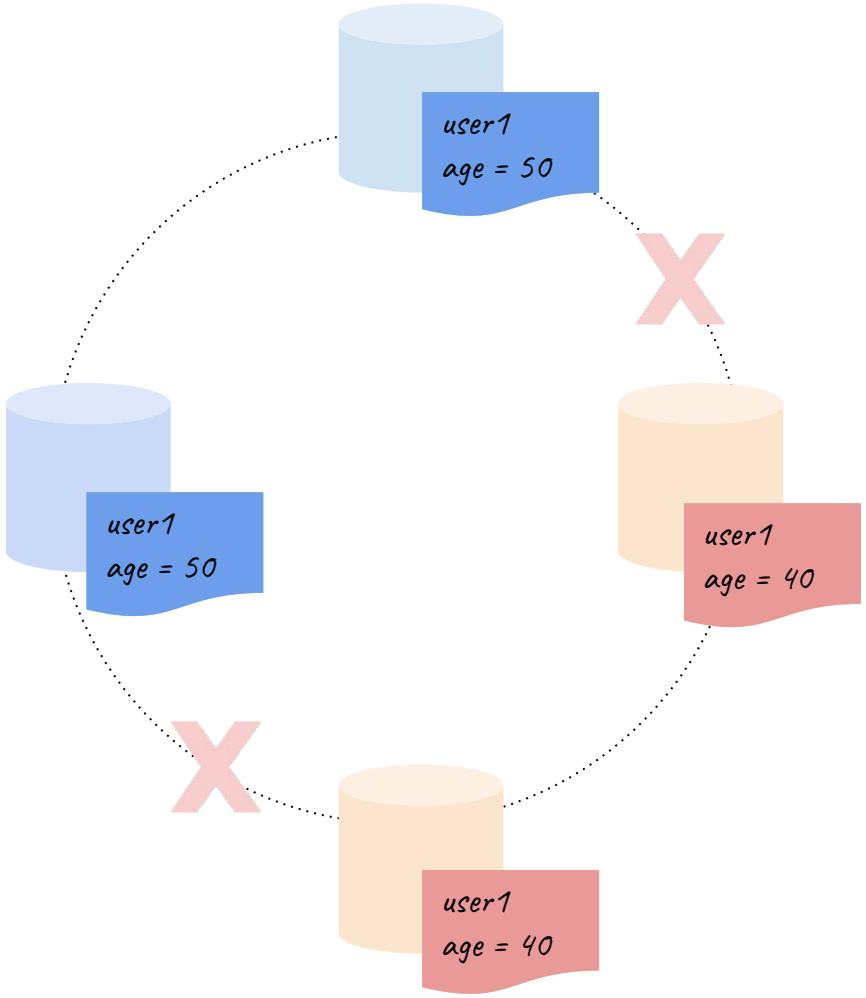


- Todos los sistemas distribuidos cumplen el teorema CAP.
- Un sistema distribuido sólo puede garantizar dos de las siguientes propiedades.
  - Consistencia.
  - Disponibilidad.
  - Tolerancia al particionado de red.

(AP) Sacrificar la Consistencia, permite escrituras en las dos partes de la partición. Cuando la partición se resuelve se necesita una forma de mergear los datos en orden para poder recuperar la consistencia.

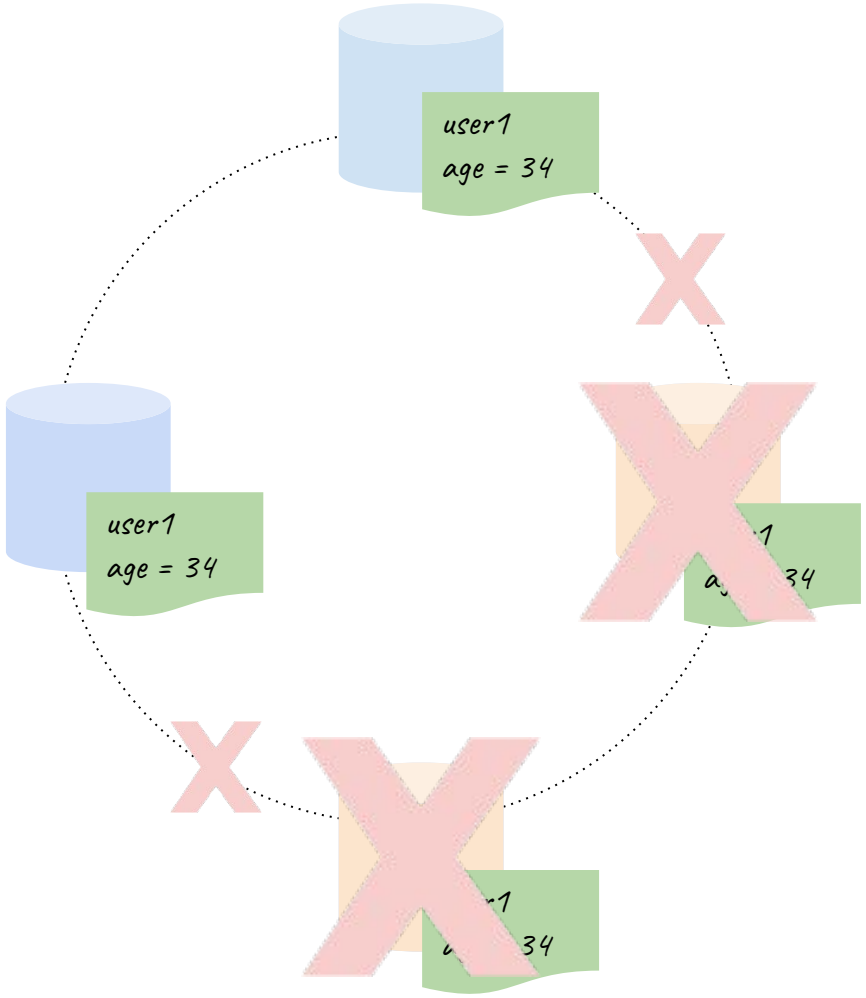
\+ Disponibilidad

\- Consistencia



(CP) Se sacrifica la disponibilidad deshabilitando un lado de la partición. Durante la partición, alguna parte de todo el sistema no estará disponible.

+ Consistencia
- Disponibilidad



Aspectos de Diseño

Elasticidad vs Consistencia

- Cuanto más carga es capaz de gestionar un sistema, más "rápido es":
  - Mayor disponibilidad del dato.
  - Gestión de volúmenes de datos más grandes en menor tiempo.
- Cuanto más carga es capaz de gestionar un sistema menor consistencia del dato.
  - La consistencia limita la elasticidad (contención).
  - Cuantos más nodos tenga un cluster, más capacidad tendrá para repartir cargas, pero más tardará en replicar los datos y por tanto perdemos consistencia (Coherency Delay).
  - Tolerancia al particionado, AP.

Consistencia vs Disponibilidad

- Dentro del sistema o caso de uso que queremos diseñar tenemos que identificar que partes requieren de consistencia estricta del dato y cuales disponibilidad alta del dato.
  - Consistencia / disponibilidad en la lectura.
  - Consistencia / disponibilidad en la escritura.
- Seleccionar adecuadamente el datastore teniendo en cuenta el teorema CAP.
  - AP: orientado a la disponibilidad, pérdida de consistencia.
  - CP: orientado a la consistencia, pérdida de disponibilidad.

Particionado

- El particionado nos ayuda a repartir la carga dentro del sistema:
  - Particionar para minimizar la contención y maximizar la elasticidad.
- Una partición debería contener únicamente el conjunto de registros a los que se va a acceder de forma conjunta (patrón de acceso al dato):
  - Random access keys vs Sequential access keys.
  - Identificar qué es lo más frecuente y modelar las particiones para ese caso.
- Particiones homogéneas:
  - Una partición debería entrar entera en un nodo.
  - Tener en cuenta la cardinalidad y la frecuencia de la clave de particionado para que las particiones sean del mismo tamaño y que soporten el mismo volumen de carga.

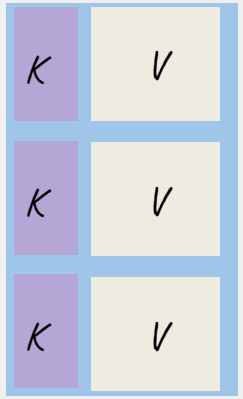
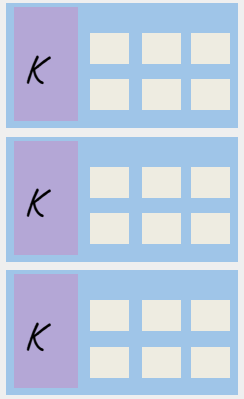
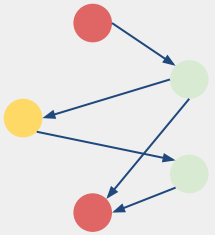
Replicación

- La replicación nos ayuda a repartir la carga dentro del sistema:
  - Cuantas más copias del dato haya, más se puede repartir la carga por lo tanto más elástico es el sistema.
  - Cuantas más copias del dato existan, menor consistencia del dato o menor disponibilidad (Coherency Delay).
  - Llegar a un compromiso entre consistencia y disponibilidad.
- Tolerancia a fallos:
  - Cuantas más copias del dato existan, más tolerancia al fallo y menos riesgo a perder información. Mínimo 3 copias.
  - Dimensionar correctamente el cluster, que cada réplica esté en un nodo distinto.

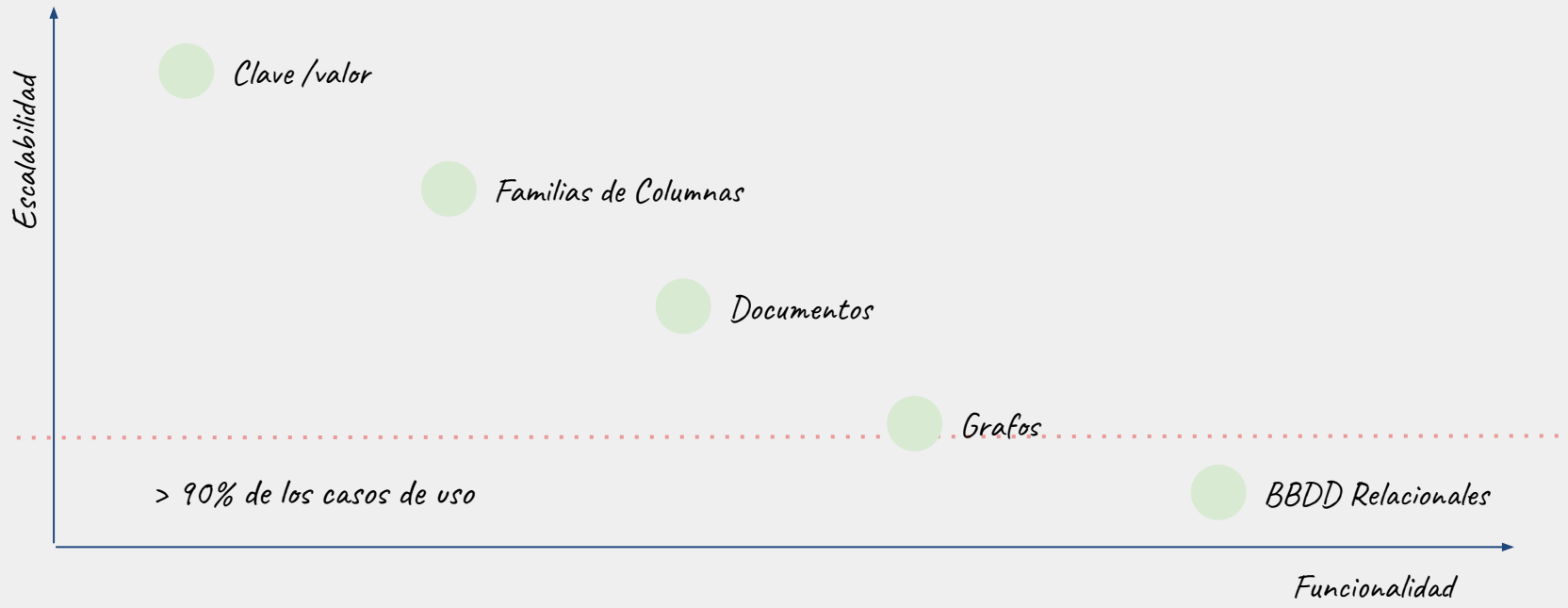
Duplicidad de información vs Disponibilidad

- Tener el dato desnormalizado aumenta su disponibilidad:
  - Unir todos los datos a los que queremos acceder en el mismo registro.
  - Aumenta la duplicidad de información. El mismo dato está en más de un sitio en el sistema.
  - Es necesario velar por la integridad del dato.
- Atomicidad:
  - Tener en cuenta cual es la unidad mínima de información para cada datastore. Normalmente el datastore es atómico para su unidad mínima de información.
  - Desnormalizar y unir los datos en la misma unidad mínima de información ayuda a suplir la falta de transacciones y por tanto ayuda a gestionar la integridad de la información.

Tipos De Bases de Datos NoSQL

Clave / Valor	Familias de Columnas	Documentos	Colas de Mensajes	Grafos
				

Escalabilidad Vs. Funcionalidad

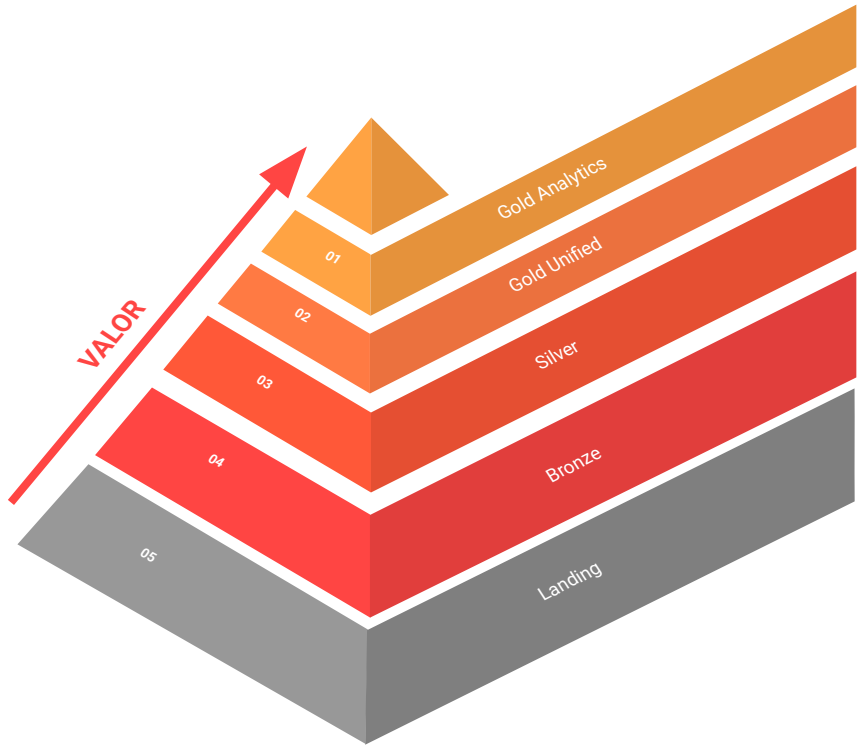


Arquitecturas Data Lake

Niveles del ciclo de vida del dato. Conceptual

Es fundamental contar con un ciclo de vida del dato, que soporte todas las necesidades de almacenamiento y explotación y permita instrumentalizar nuevas prácticas de gestión y de calidad de datos.

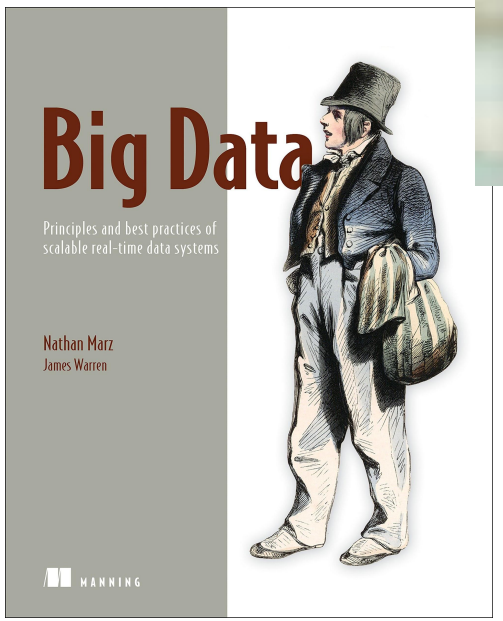
- **Gold Analytics:** Tablón (tablones en 1NF) y modelos analíticos OLAP (cubos, modelos en estrella, copo de nieve, etc.) con datos agregados, ejes secundarios y dimensiones conformadas y visión 360 corporativa.
- **Gold Unified:** Modelos normalizados de datos corporativos al máximo nivel de granularidad (maestros gobernados, tablas de referencia gobernadas, dimensiones conformadas, datos transaccionales, etc.). Incluye toda la información base corporativa.
- **Silver:** Pequeños cambios estructurales, con adaptaciones a tipos de datos para que reflejen su naturaleza (en origen). Validaciones sobre dominios en origen y transformaciones a valores canónicos.
- **Bronze:** Datos no modificados. Incluye información estructurada y no estructurada. Información segregada en entidades a nivel físico.
- **Landing:** Datos no modificados, en estructura y en envoltorio (ej. ficheros). Capa de almacenamiento temporal.



Rafael Garrote Hernández

Arquitectura Lambda

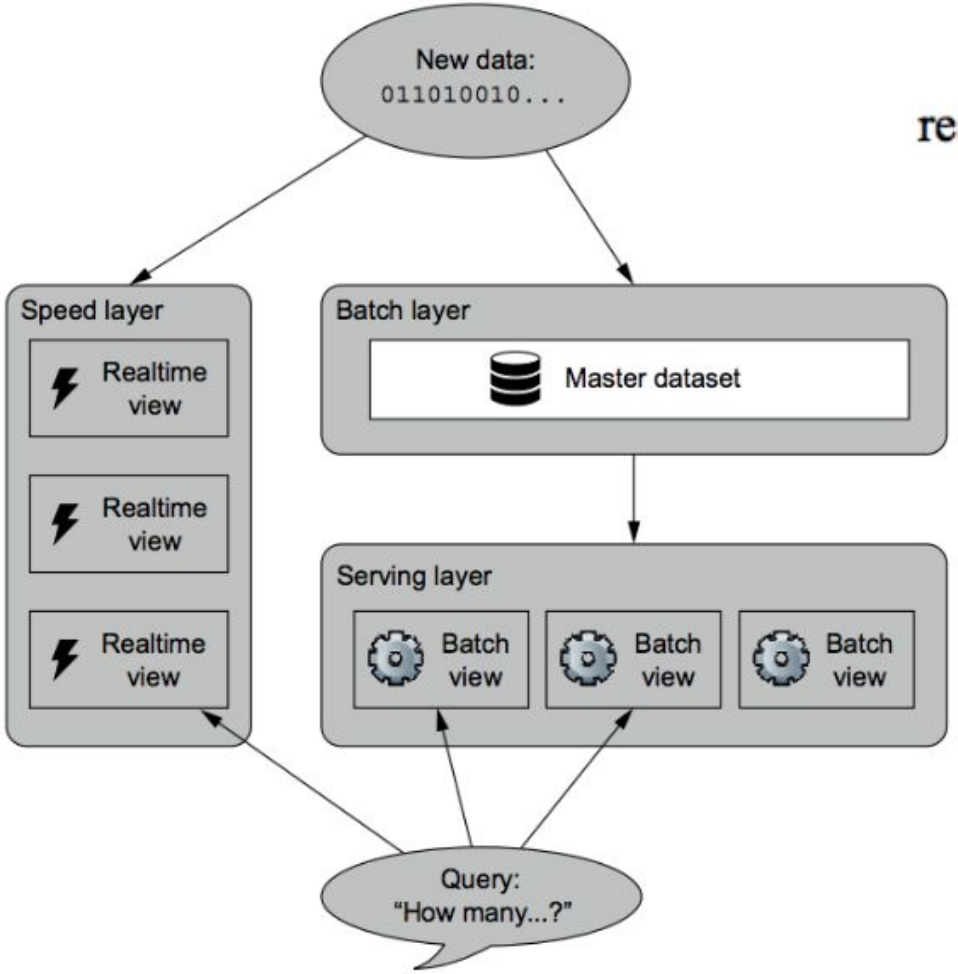
- Desarrollada por Nathan Marz
- Lead Engineer en BackType
- Twitter
- Storm



Tres capas:

- Batch Layer
- Speed Layer
- Serving Layer

Arquitectura Lambda



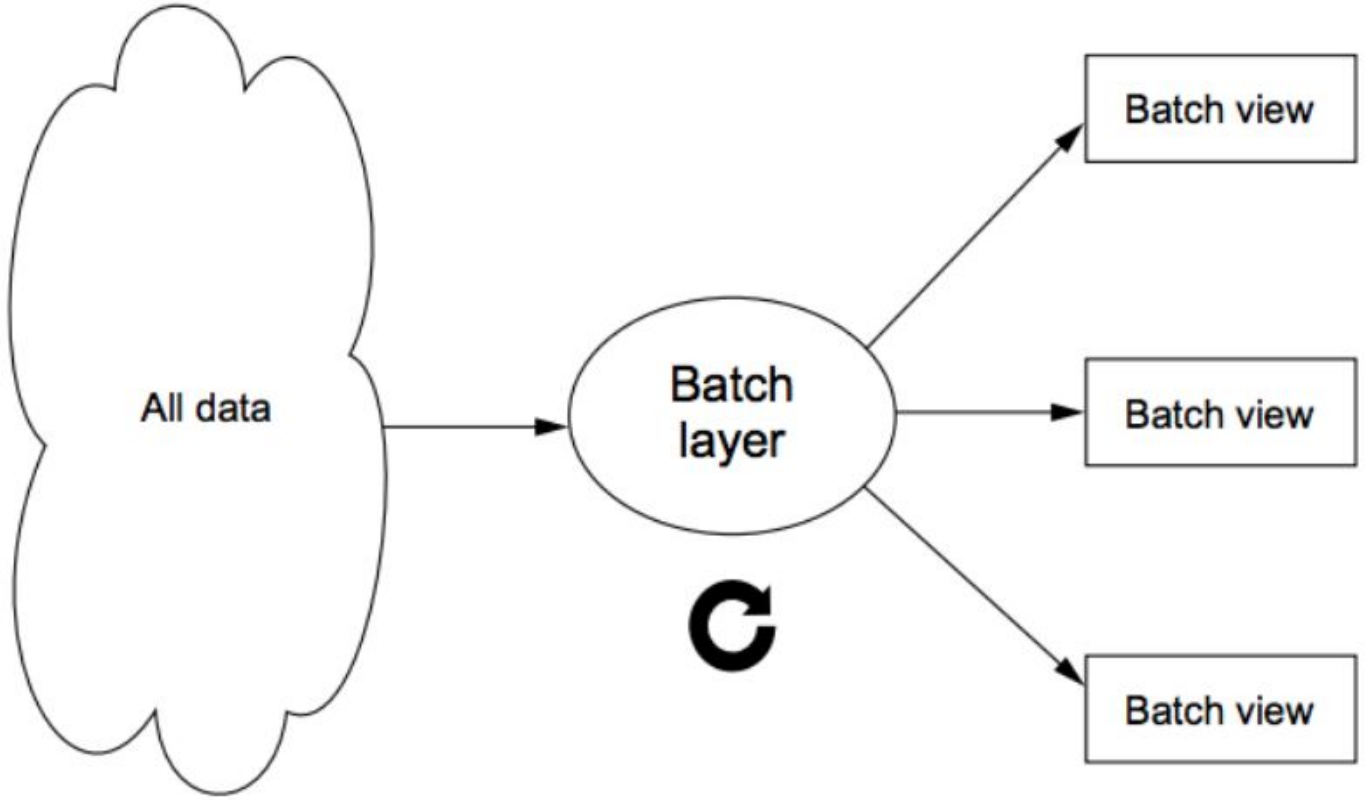
$$\text{batch view} = \text{function}(\text{all data})$$
$$\text{realtime view} = \text{function}(\text{realtime view, new data})$$
$$\text{query} = \text{function}(\text{batch view. realtime view})$$

Arquitectura Lambda

Batch Layer

- Responsable de almacenar todos los datos que queramos recopilar, Master Dataset.
- **Master dataset**: inmutable y en crecimiento continuo.
- Se computan funciones arbitrarias sobre esos datos y se generan vistas como resultados de esos procesos, *Batch Views*.
- Estas funciones son *procesos batch*.
- ***batchView = function(all data)***

Arquitectura Lambda



Arquitectura Lambda

Serving Layer

- Responsable de almacenar e indexar las batch view resultantes de la capa batch.
- Permite poder *acceder de forma aleatoria a los datos* indexados para ejecutar consultas sobre los mismos.
- Una base de datos de esta capa debe soportar *batch updates* y *random reads*, pero las random writes no son necesarias.

Arquitectura Lambda

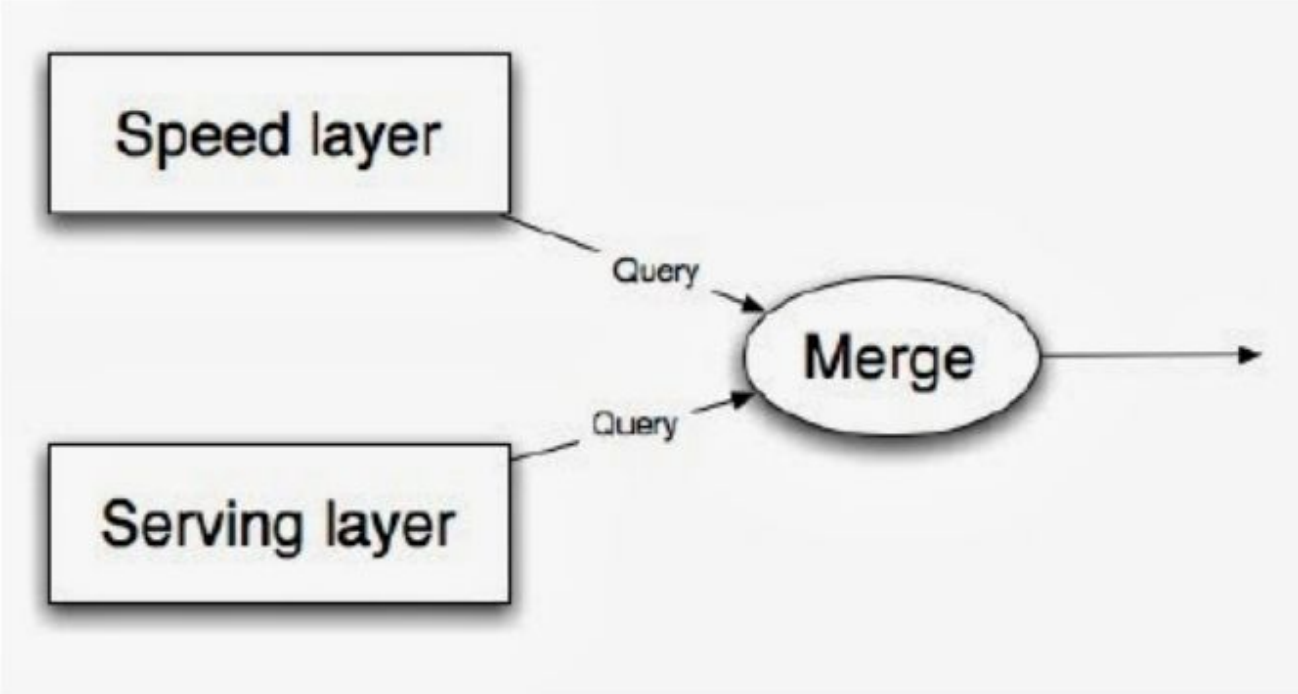
Speed Layer

- La Batch Layer no deja de ser un *proceso batch*, lo que implica *latencias altas* a la hora de poder acceder al dato computado.
- La Speed Layer intenta compensar esta latencia.
- Se encarga de efectuar *cómputo en tiempo real*.
- Es igual que la Batch Layer, pero realiza los cómputos sobre datos actuales.
- El resultado es una *realtime view*.
- El proceso de la Batch Layer sobrescribe los datos de las realtime views.

Arquitectura Lambda

Speed Layer

- realtimeView = function(realtimeView, new data)



Arquitectura Lambda

Problemas

- Latencias alta en la Batch Layer
- Problemas de conflictos en la Serving Layer.
- Duplicidad de código, código muy parecido para los procesos batch y los procesos streaming.
- Mantener dos tecnologías, batch y streaming (aunque hoy en día a tecnologías que te permiten realizar los dos tipos de procesamiento, como Spark).

Arquitectura Kappa

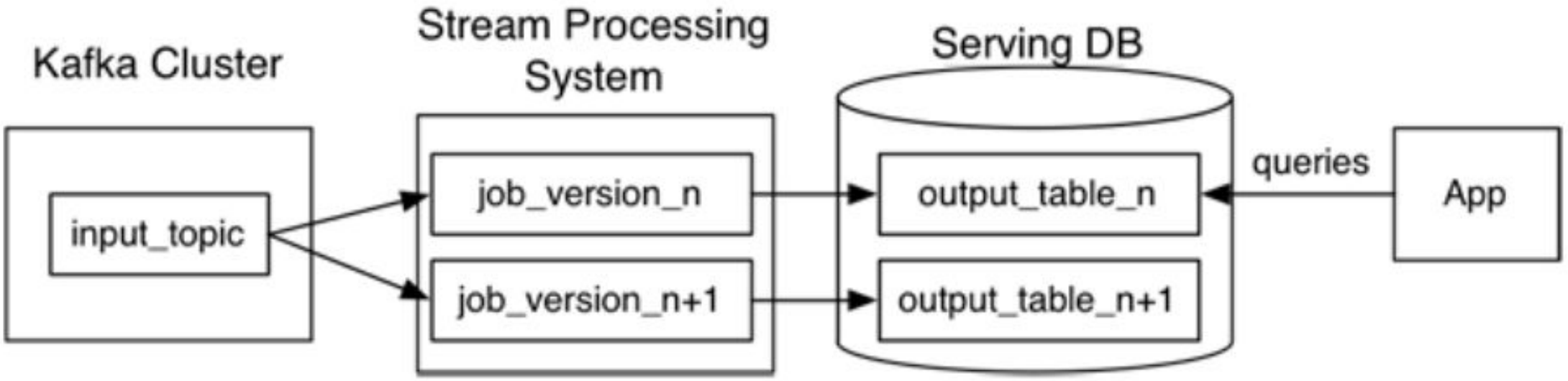
- Desarrollada por Jay Kreps
- Linkedin
- Cofundador y CEO Confluent
- Kafka
- Samza

Propone eliminar la Batch Layer dejando sólo la Speed Layer.
https://www.oreilly.com/ideas/questioning-the-lambda-architecture

Arquitectura Kappa

- Todo es un stream: las operaciones batch son un subconjunto de operaciones en streaming, por lo que pueden ser tratados como un stream.
- Los datos de partida no se modifican: los datos son almacenados sin ser modificados y las vistas se derivan de ellos. Un estado concreto puede ser recalculado puesto que la información no se modifica.
- Sólo existe un flujo de procesamiento: el código, el mantenimiento y la actualización se simplifican.
- Posibilidad de volver a lanzar un procesamiento: para variar los resultados previamente obtenidos.
- Pre-requisito: hay que garantizar que los datos se leen y almacenan en el orden en el que se han generado.

Arquitectura Kappa



Muchas gracias

datahack
INSPIRING BIG DATA