Distribución de datos y *Map Reduce*
PEC2

Nombre: Lukaz Martin Doehne

Ejercicio 1 (25%)

Healthy Soils es una empresa multinacional especializada en la optimización de explotaciones agrarias a través de Internet of Things (IoT). Entre las múltiples soluciones que ofrece a sus clientes destaca un dispositivo, alimentado por una batería propia que, montado sobre una placa Arduino Feather M0 equipada con una tarjeta de comunicaciones LoRaWAN (un protocolo de transmisión de datos asíncrono), que incluye los siguientes sensores:

- GS3: este sensor, que se introduce bajo tierra, ofrece una medida de la temperatura, otra de la conductividad eléctrica, y otra de la permeabilidad.
- SHT31: este sensor obtiene la humedad y la temperatura del aire.
- TSL2591: este sensor ofrece tres medidas independientes sobre el grado de luminosidad: la infrarroja, la del espectro entero (full-spectrum) y de la luz visible por el ojo humano.

Se debe tener en cuenta que en el futuro es posible que se incorporen nuevos tipos de sensores a la placa, incrementando el número y variedad de los datos generados. Se desea que esos datos también sean incorporados a la misma base de datos.

Healthy Soils se ha propuesto implantar este dispositivo en miles de explotaciones por todo el mundo. Cada placa enviará los datos de sus sensores conjuntamente, en intervalos de 10 minutos.

Además, para la explotación de los datos almacenados, la empresa creará una API que será consumida por una aplicación web que muestre datos y gráficas de la evolución temporal de los parámetros medidos por los sensores. Esta API también será expuesta, bajo autenticación y autorización, a cada uno de los clientes contratados que deseen utilizarla para visualizar y explotar sus propios datos, por lo que se espera que el número de consultas sea enorme y que éstas se produzcan de forma concurrente desde cualquier parte del mundo.

Se pide:
Estudiar el caso propuesto y responder a las siguientes preguntas:

1. Razonar cuál sería el modelo de almacenamiento de datos más adecuado, mencionando sus ventajas sobre el resto de modelos tratados en el curso.
2. Explicar qué estrategia de fragmentación sería la óptima.

3. Explicar cuál sería la mejor estrategia de replicación de datos.
4. El modelo transaccional más adecuado para los requisitos de la aplicación descrita.

Exponed la solución de forma argumentada en una página y media como máximo.

1. Cuál sería el modelo de almacenamiento de datos más adecuado:

Dado que se manejará una gran cantidad de datos generados por una variedad de sensores y habrá un número enorme de consultas concurrentes, un modelo de datos relacional no sería la opción recomendable. En este caso, sería más adecuado optar por **un modelo de base de datos NoSQL**.

Dentro de los diferentes tipos de bases de datos NoSQL, el modelo de grafo puede ser descartado ya que la relación entre los datos no es de gran relevancia para nosotros. En cambio, sería más apropiado **un modelo orientado hacia agregados** como las bases de datos clave-valor, orientadas a documentos u orientadas a columnas. Estos modelos ofrecen alta disponibilidad, facilidad para la fragmentación horizontal de los datos y soportan elevados niveles de concurrencia.

2. Explicar qué estrategia de fragmentación sería la óptima:

Se propone una distribución de los datos recogidos por el dispositivo de *Healthy Soils* geográficamente según las regiones/países donde se implante el dispositivo. Esto permitirá maximizar la localidad de los datos y acelerar el acceso a los mismos.

Se realizará **una fragmentación horizontal** según las regiones donde exista un número elevado de clientes, para garantizar un acceso más óptimo ya que los clientes de una misma región accederán a los mismos datos.

Además, se almacenarán las regiones con un número reducido de clientes en un mismo nodo para mantener cargas de trabajo similares en todas las particiones. Se realizarán redistribuciones de la fragmentación cuando se produzcan cambios en el número de clientes por región para optimizar el rendimiento del sistema.

La fragmentación de los datos según regiones permite una mayor disponibilidad del sistema al minimizar la probabilidad de caída. La redistribución de la fragmentación permite una mayor escalabilidad adaptándose al crecimiento del número de clientes. La distribución de carga se logra al almacenar regiones con un número reducido de clientes en un mismo nodo, manteniendo cargas de trabajo similares en todas las particiones. Finalmente, la distribución geográfica de los datos permite la localidad de los datos, lo que se traduce en una reducción de la latencia y un mejor rendimiento en general.

3. Explicar cuál sería la mejor estrategia de replicación de datos:

Se propone replicar los datos como *backup* para asegurar su disponibilidad en caso de caídas. Debido a la distribución geográfica de los datos, las réplicas se ubican en el nodo geográfico al que fueron asignados para disminuir el tiempo de transmisión. Para ello se utilizará una estrategia de **replicación de tipo *master-slave* asíncrona**, en la que el nodo maestro es responsable de las escrituras y los nodos esclavos replican los datos. Esto permite un alto rendimiento en situaciones de carga pesada, ya que los nodos esclavos pueden manejar las lecturas sin afectar al nodo maestro.

4. El modelo transaccional más adecuado para los requisitos de la aplicación descrita:

Se sugiere la **implementación del modelo BASE**, que a diferencia del modelo ACID, acepta la existencia de inconsistencias en los datos, pero las detecta y resuelve. Este modelo da prioridad a la disponibilidad sobre la consistencia, lo que implica que una posible pérdida temporal de la consistencia de la información es asumible, ya que se tiene un margen de 10 minutos entre actualizaciones de los sensores.

Ejercicio 2 (25%)

A partir de la lectura de los apuntes del curso exponed, para cada una de las afirmaciones, si creéis que es cierta o falsa indicando una breve argumentación que justifique vuestra respuesta.

Las afirmaciones en las que no se indique si es cierta o falsa, o bien carezcan de argumentación, se considerarán no válidas. Se **valorará la concisión** (una página y media para las 5 afirmaciones como máximo) y el **uso de referencias** (sección/página del libro de referencia o apuntes del curso) para justificar las respuestas.

Afirmación 1

La fragmentación en una base de datos relacional implica que un conjunto de datos sólo esté disponible en un fragmento y, por lo tanto, el nodo que almacene ese fragmento será el único responsable de sus datos.

Cierto. Las bases de datos relacionales, en general, cumplen las propiedades ACID. Por lo tanto, para mantener la consistencia, el conjunto de datos estará disponible en un fragmento en un único nodo responsable de los datos.

Ref:

· Tema 5. Bases de datos distribuidas, Teorema CAP y modelo BASE, Sección: BD distribuidas y el teorema CAP: consideraciones finales. B3_T7_BDD_BASE.pdf - pg. 11,12

Afirmación 2

La principal ventaja del modelo de replicación maestro-esclavo es la consistencia. Nunca es posible que diferentes clientes lean diferentes nodos secundarios y los datos sean diferentes.

Falso. En una replicación *master-slave* asíncrona es posible que diferentes clientes lean diferentes nodos secundarios con datos diferentes. La principal ventaja de este modelo es la alta disponibilidad cuando hay muchas lecturas y pocas escrituras, ya que los nodos esclavos manejan las lecturas sin afectar al nodo maestro.

Ref:

· Tema 7. Modelo de transacciones BASE, Bases de datos distribuidas, Teorema CAP y modelo BASE, Sección: Replicación *master-slave*. B3_T7_BDD_BASE.pdf - pg. 7
· NoSQL Distilled, Chapter 5. Consistency, Sección: 5.3.1 The CAP Theorem & 5.4 Relaxing Durability

Afirmación 3

En el modelo MapReduce, todas las funciones de reducción son combinables.

Falso. No todas las funciones de reducción son combinables. Por ejemplo, cuando la salida de la función de reducción es diferente de su entrada.

Ref:

· NoSQL Distilled, Chapter 7. Map-Reduce, Sección: 7.2 Partitioning and Combining, Figura: 7.5 con su contexto

Afirmación 4

Las bases de datos orientadas a documentos sólo admiten replicación, resultando imposible las técnicas de distribución como sharding.

Falso. Las bases de datos orientadas a documentos admiten tanto replicación como técnicas de distribución como *sharding*, y algunas de las bases de datos NoSQL más populares como *MongoDB* permiten ambas técnicas.

Ref:

· Tema 5. Bases de datos distribuidas, Diseño, Sección: Diseño de BD distribuidas: BD NoSQL. B3_T5_3_BDD_Disen¿o.pdf - pg. 14

Afirmación 5

El teorema CAP ha permitido un cambio de paradigma en la programación de la persistencia de información, ya que implica una relajación de las propiedades de transacciones ACID.

Verdadero. El teorema CAP, en el contexto de las bases de datos NoSQL, captura la idea de elegir dos de los 3 objetivos (consistencia, disponibilidad y tolerancia a particiones) en la gestión de datos altamente distribuidos. Por lo tanto, afloja las propiedades de transacciones ACID.

Ref:

· Tema 7. Modelo de transacciones BASE, Bases de datos distribuidas, Teorema CAP y modelo BASE, Sección: Teorema CAP. B3_T7_BDD_BASE.pdf - pg. 9,20
· NoSQL Distilled, Chapter 5. Consistency, Sección: 5.3.1 The CAP Theorem

Ejercicio 3 (30%)

Considerar los datos de los atletas que participan en carreras de media y larga distancia en España.

Por un lado, se dispone de información personal de cada atleta, identificado de forma única por el ID del chip con el que participan en las competiciones.

chip_id	nombre	apellido1	apellido2	residente_en
JGK309	Alejandro	García	Rodríguez	Madrid
QNR874	María	Fernández	Pérez	Barcelona
BDM752	Pablo	González	Sánchez	Valencia
TLF931	Laura	López	Martínez	Sevilla
WXP684	Sergio	Torres	Gómez	Bilbao
HVF460	Ana	Ruiz	Jiménez	Barcelona
YTK639	Luis	Moreno	Álvarez	Murcia
EBN206	Carmen	Castro	Romero	Barcelona
IJM125	Javier	Morales	Gutiérrez	Malaga
UAO572	Marta	Navarro	Vázquez	Cordoba

Por otro lado, se cuenta con el registro de las inscripciones de las diferentes carreras oficiales de media/larga distancia en España.

chip_id	tipo_carrera	ciudad_carrera	fecha	tiempo
JGK309	Maratón	Madrid	12-05-2020	02h:43m:18s
QNR874	10Km	Barcelona	23-09-2021	01h:15m:59s
BDM752	5Km	Valencia	04-07-2021	00h:28m:27s
TLF931	Maratón	Sevilla	19-02-2021	03h:26m:45s
WXP684	10Km	Bilbao	29-11-2022	01h:02m:13s
HVF460	Maratón	Zaragoza	08-08-2022	03h:59m:38s
YTK639	Maraton	Murcia	27-01-2020	03h:21m:49s
EBN206	Maraton	Alicante	05-10-2022	03h:48m:02s
IJM125	5Km	Malaga	17-06-2021	00h:22m:56s
ABC123	Maratón	Cordoba	01-04-2021	02h:37m:04s

Se desea identificar el atleta de los residentes en Barcelona con el mejor tiempo de maratón en 2022. El output tendrá que indicar el nombre y apellidos del atleta y el mejor tiempo.

Se aconseja usar dos pasadas map-reduce. Para ello se pide explicar en cada pasada lo que ocurre en cada fase (map, shuffle, reduce).
Concretamente para cada fase:

- Mostrar todos los datos de entrada (input)
- Explicar qué acciones se producen
- Mostrar el resultado producido (output) por esas acciones en todos los datos de entrada.

Es imprescindible explicar claramente la lógica de las acciones realizadas, enseñando en detalle el input y el output de cada fase. Se valorarán la **lógica y la eficiencia del algoritmo** y las explicaciones, y no (pseudo)código.

Primera pasada de map-reduce

1. **Fase *map***

Datos de entrada:

```
Atletas = [
        (“JGK309”, ”Alejandro”, ”García”, ”Rodríguez”, ”Madrid”),
        (“QNR874”, ”María”, ”Fernández”, ”Pérez”, ”Barcelona”),
        (“BDM752”, ”Pablo”, ”González”, ”Sánchez”, ”Valencia”),
        (”TLF931”, ”Laura”, ”López”, ”Martínez”, ”Sevilla”),
        (“WXP684”, ”Sergio”, ”Torres”, ”Gómez”, ”Bilbao”),
        (“HVF460”, ”Ana”, ”Ruiz”, ”Jiménez”, ”Barcelona”),
        (“YTK639”, ”Luis”, ”Moreno”, ”Álvarez”, ”Murcia”),
        (“EBN206”, ”Carmen”, ”Castro”, ”Romero”, ”Barcelona”),
        (“IJM125”, ”Javier”, ”Morales”, ”Gutiérrez”, ”Malaga”),
        (“UA0572”, ”Marta”, ”Navarro”, ”Vázquez”, ”Cordoba”)
    ]
Carreras = [
        (“JGK309”, “Maratón”, “Madrid”, “12-05-2020”, “02h:43m:18s”),
        (“QNR874”, “10Km”, “Barcelona”, “23-09-2021”, “01h:15m:59s”),
        (“BDM752”, “5Km”, “Valencia”, “04-07-2021”, “00h:28m:27s”),
        (“TLF931”, “Maratón”, “Sevilla”, “19-02-2021”, “03h:26m:45s”),
        (“WXP684”, “10Km”, “Bilbao”, “29-11-2022”, “01h:02m:13s”),
        (“HVF460”, “Maratón”, “Zaragoza”, “08-08-2022”, “03h:59m:38s”),
        (“YTK639”, “Maratón”, “Murcia”, “27-01-2020”, “03h:21m:49s”),
        (“EBN206”, “Maratón”, “Alicante”, “05-10-2022”, “03h:48m:02s”),
        (“IJM125”, “5Km”, “Malaga”, “17-06-2021”, “00h:22m:56s”),
        (“ABC123”, “Maratón”, “Cordoba”, “01-04-2021”, “02h:37m:04s”)
    ]
```

Acciones:

La operación *map* va a procesar las dos listas como entrada. Para cada tupla de las listas, la función *map* devolverá un objeto clave-valor donde la clave será el campo ***chip_id*** y el valor será la tupla completa. Es decir, leemos la tupla, extraemos el identificador en la primera posición de la tupla y generamos la pareja (identificador, tupla). Primero se procesa el documento de Atletas y luego el de Carreras.

Datos de salida:

clave	valor
JGK309	(“JGK309”, ”Alejandro”, ”García”, ”Rodríguez”, ”Madrid”)
QNR874	(“QNR874”, ”María”, ”Fernández”, ”Pérez”, ”Barcelona”)
BDM752	(“BDM752”, ”Pablo”, ”González”, ”Sánchez”, ”Valencia”)
TLF931	(”TLF931”, ”Laura”, ”López”, ”Martínez”, ”Sevilla”)
WXP684	(“WXP684”, ”Sergio”, ”Torres”, ”Gómez”, ”Bilbao”)
HVF460	(“HVF460”, ”Ana”, ”Ruiz”, ”Jiménez”, ”Barcelona”)
YTK639	(“YTK639”, ”Luis”, ”Moreno”, ”Álvarez”, ”Murcia”)
EBN206	(“EBN206”, ”Carmen”, ”Castro”, ”Romero”, ”Barcelona”)
IJM125	(“IJM125”, ”Javier”, ”Morales”, ”Gutiérrez”, ”Malaga”)
UA0572	(“UA0572”, ”Marta”, ”Navarro”, ”Vázquez”, ”Cordoba”)
JGK309	(“JGK309”, “Maratón”, “Madrid”, “12-05-2020”, “02h:43m:18s”)
QNR874	(“QNR874”, “10Km”, “Barcelona”, “23-09-2021”, “01h:15m:59s”)
BDM752	(“BDM752”, “5Km”, “Valencia”, “04-07-2021”, “00h:28m:27s”)
TLF931	(“TLF931”, “Maratón”, “Sevilla”, “19-02-2021”, “03h:26m:45s”)
WXP684	(“WXP684”, “10Km”, “Bilbao”, “29-11-2022”, “01h:02m:13s”)
HVF460	(“HVF460”, “Maratón”, “Zaragoza”, “08-08-2022”, “03h:59m:38s”)
YTK639	(“YTK639”, “Maratón”, “Murcia”, “27-01-2020”, “03h:21m:49s”)
EBN206	(“EBN206”, “Maratón”, “Alicante”, “05-10-2022”, “03h:48m:02s”)
IJM125	(“IJM125”, “5Km”, “Malaga”, “17-06-2021”, “00h:22m:56s”)
ABC123	(“ABC123”, “Maratón”, “Cordoba”, “01-04-2021”, “02h:37m:04s”)

2. Fase *shuffle*

Datos de entrada: (datos de salida de la fase *map*)

clave	valor
JGK309	(“JGK309”, ”Alejandro”, ”García”, ”Rodríguez”, ”Madrid”)
QNR874	(“QNR874”, ”María”, ”Fernández”, ”Pérez”, ”Barcelona”)
BDM752	(“BDM752”, ”Pablo”, ”González”, ”Sánchez”, ”Valencia”)
TLF931	(”TLF931”, ”Laura”, ”López”, ”Martínez”, ”Sevilla”)
WXP684	(“WXP684”, ”Sergio”, ”Torres”, ”Gómez”, ”Bilbao”)
HVF460	(“HVF460”, ”Ana”, ”Ruiz”, ”Jiménez”, ”Barcelona”)
YTK639	(“YTK639”, ”Luis”, ”Moreno”, ”Álvarez”, ”Murcia”)
EBN206	(“EBN206”, ”Carmen”, ”Castro”, ”Romero”, ”Barcelona”)
IJM125	(“IJM125”, ”Javier”, ”Morales”, ”Gutiérrez”, ”Malaga”)
UA0572	(“UA0572”, ”Marta”, ”Navarro”, ”Vázquez”, ”Cordoba”)
JGK309	(“JGK309”, “Maratón”, “Madrid”, “12-05-2020”, “02h:43m:18s”)
QNR874	(“QNR874”, “10Km”, “Barcelona”, “23-09-2021”, “01h:15m:59s”)
BDM752	(“BDM752”, “5Km”, “Valencia”, “04-07-2021”, “00h:28m:27s”)
TLF931	(“TLF931”, “Maratón”, “Sevilla”, “19-02-2021”, “03h:26m:45s”)
WXP684	(“WXP684”, “10Km”, “Bilbao”, “29-11-2022”, “01h:02m:13s”)
HVF460	(“HVF460”, “Maratón”, “Zaragoza”, “08-08-2022”, “03h:59m:38s”)
YTK639	(“YTK639”, “Maratón”, “Murcia”, “27-01-2020”, “03h:21m:49s”)
EBN206	(“EBN206”, “Maratón”, “Alicante”, “05-10-2022”, “03h:48m:02s”)
IJM125	(“IJM125”, “5Km”, “Malaga”, “17-06-2021”, “00h:22m:56s”)
ABC123	(“ABC123”, “Maratón”, “Cordoba”, “01-04-2021”, “02h:37m:04s”)

Acciones:

En la función *Shuffle*, agrupamos los valores por clave. Si hay varias tuplas con la misma clave, se adjuntan todas las tuplas en una lista. Es decir, generamos nuevas parejas (clave, valor) donde la nueva clave será el identificador del chip y el valor una lista de tuplas (una tupla por cada ocurrencia). En caso de que un corredor haya participado en varias carreras, estás se adjuntan al final de la lista (en el ejemplo no existen registros así).

Datos de salida:

clave	valor
JGK309	[(“JGK309”, ”Alejandro”, ”García”, ”Rodríguez”, ”Madrid”), (“JGK309”, “Maratón”, “Madrid”, “12-05-2020”, “02h:43m:18s”)]
QNR874	[(“QNR874”, ”María”, ”Fernández”, ”Pérez”, ”Barcelona”), (“QNR874”, “10Km”, “Barcelona”, “23-09-2021”, “01h:15m:59s”)]
BDM752	[(“BDM752”, ”Pablo”, ”González”, ”Sánchez”, ”Valencia”), (“BDM752”, “5Km”, “Valencia”, “04-07-2021”, “00h:28m:27s”)]
TLF931	[(”TLF931”, ”Laura”, ”López”, ”Martínez”, ”Sevilla”), (“TLF931”, “Maratón”, “Sevilla”, “19-02-2021”, “03h:26m:45s”)]
WXP684	[(“WXP684”, ”Sergio”, ”Torres”, ”Gómez”, ”Bilbao”), (“WXP684”, “10Km”, “Bilbao”, “29-11-2022”, “01h:02m:13s”)]
HVF460	[(“HVF460”, ”Ana”, ”Ruiz”, ”Jiménez”, ”Barcelona”), (“HVF460”, “Maratón”, “Zaragoza”, “08-08-2022”, “03h:59m:38s”)]
YTK639	[(“YTK639”, ”Luis”, ”Moreno”, ”Álvarez”, ”Murcia”), (“YTK639”, “Maratón”, “Murcia”, “27-01-2020”, “03h:21m:49s”)]
EBN206	[(“EBN206”, ”Carmen”, ”Castro”, ”Romero”, ”Barcelona”), (“EBN206”, “Maratón”, “Alicante”, “05-10-2022”, “03h:48m:02s”)]
IJM125	[(“IJM125”, ”Javier”, ”Morales”, ”Gutiérrez”, ”Malaga”), (“IJM125”, “5Km”, “Malaga”, “17-06-2021”, “00h:22m:56s”)]
UA0572	[(“UA0572”, ”Marta”, ”Navarro”, ”Vázquez”, ”Cordoba”)]
ABC123	[(“ABC123”, “Maratón”, “Cordoba”, “01-04-2021”, “02h:37m:04s”)]

3. Fase *reduce*

Datos de entrada: (datos de salida de la fase *shuffle*)

clave	valor
JGK309	[(“JGK309”, ”Alejandro”, ”García”, ”Rodríguez”, ”Madrid”), (“JGK309”, “Maratón”, “Madrid”, “12-05-2020”, “02h:43m:18s”)]
QNR874	[(“QNR874”, ”María”, ”Fernández”, ”Pérez”, ”Barcelona”), (“QNR874”, “10Km”, “Barcelona”, “23-09-2021”, “01h:15m:59s”)]
BDM752	[(“BDM752”, ”Pablo”, ”González”, ”Sánchez”, ”Valencia”), (“BDM752”, “5Km”, “Valencia”, “04-07-2021”, “00h:28m:27s”)]
TLF931	[(”TLF931”, ”Laura”, ”López”, ”Martínez”, ”Sevilla”), (“TLF931”, “Maratón”, “Sevilla”, “19-02-2021”, “03h:26m:45s”)]
WXP684	[(“WXP684”, ”Sergio”, ”Torres”, ”Gómez”, ”Bilbao”), (“WXP684”, “10Km”, “Bilbao”, “29-11-2022”, “01h:02m:13s”)]

HVF460	[(“HVF460”, ”Ana”, ”Ruiz”, ”Jiménez”, ”Barcelona”), (“HVF460”, “Maratón”, “Zaragoza”, “08-08-2022”, “03h:59m:38s”)]
YTK639	[(“YTK639”, ”Luis”, ”Moreno”, ”Álvarez”, ”Murcia”), (“YTK639”, “Maratón”, “Murcia”, “27-01-2020”, “03h:21m:49s”)]
EBN206	[(“EBN206”, ”Carmen”, ”Castro”, ”Romero”, ”Barcelona”), (“EBN206”, “Maratón”, “Alicante”, “05-10-2022”, “03h:48m:02s”)]
IJM125	[(“IJM125”, ”Javier”, ”Morales”, ”Gutiérrez”, ”Malaga”), (“IJM125”, “5Km”, “Malaga”, “17-06-2021”, “00h:22m:56s”)]
UA0572	[(“UA0572”, ”Marta”, ”Navarro”, ”Vázquez”, ”Cordoba”)]
ABC123	[(“ABC123”, “Maratón”, “Cordoba”, “01-04-2021”, “02h:37m:04s”)]

Acciones:

En la función *Reduce*, se leen la lista de parejas generadas de la fase *Shuffle*. Se mantienen las parejas (clave, valor) con el identificador chip como clave y la lista de tuplas como valor de los registros que cumplen con las condiciones (residente en Barcelona, maratón y 2022). Para determinar si el atleta es residente en Barcelona, se comprueba si el valor de la última posición de la primera tupla es igual a “Barcelona”. Para determinar si la carrera fue de maratón, se comprueba si el valor del segundo elemento de la segunda tupla (si existe) es igual a “Maratón”. Para determinar si fue en el año 2022, se comprueba si los últimos 4 carácteres del cuarto elemento de la segunda tupla son “2022”. Finalmente, devolvemos la pareja (clave, valor) donde se han cumplido las condiciones.

Nota: Observamos que el último registro del documento de Carreras no tiene un correspondiente elemento de Atleta, por lo que quedaría descartado al comprobar que “Barcelona” no está en la última posición de la primera tupla. Además, en el ejemplo proporcionado, no hay registros donde alguien haya realizado más de una carrera. Si este fuera el caso, en lugar de comprobar solo la segunda tupla, deberíamos comprobar todas las tuplas posteriores a la primera y mantener solo las que cumplen las condiciones.

Datos de salida:

clave	valor
HVF460	[(“HVF460”, ”Ana”, ”Ruiz”, ”Jiménez”, ”Barcelona”), (“HVF460”, “Maratón”, “Zaragoza”, “08-08-2022”, “03h:59m:38s”)]
EBN206	[(“EBN206”, ”Carmen”, ”Castro”, ”Romero”, ”Barcelona”), (“EBN206”, “Maratón”, “Alicante”, “05-10-2022”, “03h:48m:02s”)]

Segunda pasada de map-reduce

1. Fase *map*

Datos de entrada:

clave	valor
HVF460	[(“HVF460”, ”Ana”, ”Ruiz”, ”Jiménez”, ”Barcelona”), (“HVF460”, “Maratón”, “Zaragoza”, “08-08-2022”, “03h:59m:38s”)]

EBN206	[(“EBN206”, ”Carmen”, ”Castro”, ”Romero”, ”Barcelona”), (“EBN206”, “Maratón”, “Alicante”, “05-10-2022”, “03h:48m:02s”)]

Acciones:

Cada pareja generada en la fase *Reduce* se convierte en una nueva pareja clave-valor donde la clave es el identificador del chip y el valor es una concatenación de los campos nombre, apellido1 y apellido2 de la primera tupla y el tiempo de la segunda tupla (la que tiene el menor tiempo si hubiera varias).

Datos de salida:

clave	valor
HVF460	(”Ana”, ”Ruiz”, ”Jiménez”, “03h:59m:38s”)
EBN206	(”Carmen”, ”Castro”, ”Romero”, “03h:48m:02s”)

2. **Fase *shuffle***

Datos de entrada:

clave	valor
HVF460	(”Ana”, ”Ruiz”, ”Jiménez”, “03h:59m:38s”)
EBN206	(”Carmen”, ”Castro”, ”Romero”, “03h:48m:02s”)

Acciones:

Para cada tupla de entrada se emite la misma tupla para la salida.

Datos de salida:

clave	valor
HVF460	(”Ana”, ”Ruiz”, ”Jiménez”, “03h:59m:38s”)
EBN206	(”Carmen”, ”Castro”, ”Romero”, “03h:48m:02s”)

3. **Fase *reduce***

Datos de entrada:

clave	valor
HVF460	(”Ana”, ”Ruiz”, ”Jiménez”, “03h:59m:38s”)
EBN206	(”Carmen”, ”Castro”, ”Romero”, “03h:48m:02s”)

Acciones:

La fase *reduce* procesa las parejas generadas y busca la tupla de menor tiempo para cada identificador (última posición de la tupla). Si este tiempo es menor que el tiempo mínimo registrado hasta ese momento (inicializado a infinito), se actualiza una variable local con el identificador de ese atleta y el nuevo tiempo mínimo. Este proceso se

repite para todas las parejas. En caso de empate, se podría optar por devolver todos los atletas con el mínimo tiempo. Al final, se tendrá el identificador en la variable local del atleta que tardó el menor tiempo en realizar una carrera de maratón en el año 2022 siendo residente en Barcelona. Se devuelve el valor asociado a ese identificador.

Datos de salida:

("Carmen", "Castro", "Romero", "03h:48m:02s")

Ejercicio 4 (20%)

Considerar los siguientes sistemas:

1. Un sistema de sensores de la calidad del aire de una ciudad (consistencia final en el tiempo, Escrituras más frecuentes que lecturas)
2. Un periódico online que es visitado por lectores de diferentes lugares de un país (consistencia final en el tiempo, Lecturas más frecuentes que escrituras)
3. Un sistema de control aéreo (consistencia fuerte, Escrituras y lecturas de la misma importancia)
4. Un sistema de banca online (consistencia fuerte, Lecturas más frecuentes que escrituras)

Se pide argumentar qué configuraciones de los valores W, R y N encajan mejor con cada caso con respecto al tipo de consistencia (fuerte/final en el tiempo) y al tipo de lecturas y escrituras que son necesarias realizar (Lecturas más frecuentes que escrituras, Escrituras más frecuentes que lecturas o Escrituras y lecturas de la misma importancia).

C1: N=3, W=1, R=1 → Consistencia final en el tiempo (W+R<=3)
C2: N=3, W=4, R=2 → Consistencia fuerte (W> 1.5, W+R>3)
C3: N=5, W=1, R=3 → Consistencia final en el tiempo (W+R<=5)
C4: N=5, W=3, R=1 → Consistencia final en el tiempo (W+R<=5)
C5: N=7, W=4, R=5 → Consistencia fuerte (W>3.5, W+R>7)
C6: N=7, W=5, R=4 → Consistencia fuerte (W>3.5, W+R>7)
C7: N=5, W=5, R=1 → Consistencia estricta (N=W)

Para resolverlo:

- Primero identificar y razonar tipo de consistencia, y tipo de lecturas y escrituras.
- A continuación, analizar cuál de las configuraciones dadas encaja con el sistema propuesto.

La consistencia de un sistema se define con los 3 parámetros N (número de réplicas), W (número de réplicas de escritura) y R (número de réplicas de lectura). A partir de aquí, se pueden identificar 3 tipos de consistencias:

- Consistencia fuerte:
  - W > N/2
  - W + R > N
- Consistencia final en el tiempo:
  - W + R <= N
- Consistencia estricta:
  - N=W

1. **Un sistema de sensores de la calidad del aire de una ciudad**

El sistema debe implementar una consistencia final en el tiempo (W+R<=N), ya que las otras opciones ofrecerán menos disponibilidad. Además, cómo las escrituras serán más frecuentes que las lecturas entonces idealmente buscamos una opción donde W<R. Las condiciones de consistencia final en el tiempo la cumplen las opciones:

C1: N=3, W=1, R=1 → Consistencia final en el tiempo (1+1<=3)
C3: N=5, W=1, R=3 → Consistencia final en el tiempo (1+3<=5)
C4: N=5, W=3, R=1 → Consistencia final en el tiempo (3+1<=5)

De estás opciones, la más eficiente para las escrituras más frecuentes que lecturas es C3. Por lo que la mejor configuración es N=5, W=1 y R=3.

2. **Un periódico online que es visitado por lectores de diferentes lugares de un país**

El sistema debe implementar una consistencia final en el tiempo (W+R<=N), ya que las otras opciones ofrecerán menos disponibilidad. Además, cómo las lecturas serán más frecuentes que las escrituras entonces idealmente buscamos una opción donde R<W. Las condiciones de consistencia final en el tiempo la cumplen las opciones:

C1: N=3, W=1, R=1 → Consistencia final en el tiempo (1+1<=3)
C3: N=5, W=1, R=3 → Consistencia final en el tiempo (1+3<=5)
C4: N=5, W=3, R=1 → Consistencia final en el tiempo (3+1<=5)

De estás opciones, la más eficiente para las lecturas más frecuentes que escrituras es C4. Por lo que la mejor configuración es N=5, W=3 y R=1.

3. **Un sistema de control aéreo**

El sistema debe implementar una consistencia fuerte (W>N/2, W+R>N). Además, cómo las escrituras y lecturas serán de la misma importancia entonces idealmente buscamos una opción donde R=W. Las condiciones de consistencia fuerte la cumplen las opciones:

C2: N=3, W=4, R=2 → Consistencia fuerte (W> 1.5, W+R>3)
C5: N=7, W=4, R=5 → Consistencia fuerte (W>3.5, W+R>7)

C6: N=7, W=5, R=4 → Consistencia fuerte (W>3.5, W+R>7)

De estás opciones, las más eficientes para escrituras y lecturas de misma importancia son C5 y C6. Por lo que la mejor configuración es N=7, W=4 y R=5 o N=7, W=5 y R=4.

4. **Un sistema de banca online**

El sistema debe implementar una consistencia fuerte (W>N/2, W+R>N). Además, cómo las lecturas serán más frecuentes que las escrituras entonces idealmente buscamos una opción donde R<W. Las condiciones de consistencia final en el tiempo la cumplen las opciones:

C2: N=3, W=4, R=2 → Consistencia fuerte (W> 1.5, W+R>3)
C5: N=7, W=4, R=5 → Consistencia fuerte (W>3.5, W+R>7)
C6: N=7, W=5, R=4 → Consistencia fuerte (W>3.5, W+R>7)

De estás opciones, la más eficiente para las lecturas más frecuentes que escrituras es C2. Por lo que la mejor configuración es N=3, W=4 y R=2.