Conceptos básicos y modelos NoSQL
PEC1

Ejercicio 1 (30%)

A partir de la lectura de los apuntes (locuciones de los vídeos) de los temas I y II se pide responder de manera concisa (una página y media en total) a las siguientes preguntas:

1. Con respecto a los sistemas gestores de bases de datos que implementan los modelos en grafo, describe sus principales características con respecto a la gestión de transacciones y consistencia, al acceso a los datos, y a la escalabilidad.

    - **Gestión de transacciones y consistencia**. Interpretación más parecida a la de los SGBD relacionales que a la de los NoSQL de agregación. Utilizan transacciones **ACID**, lo que garantiza la atomicidad, **consistencia**, aislamiento y durabilidad de las mismas.
    - **Acceso a los datos**. Estos SGBD emplean lenguajes de **alto nivel**, lo que disminuye la complejidad lógica de las consultas (acceso a los datos) a nivel de usuario.
    - **Escalabilidad**. Debido a que se realizan transacciones de tipo ACID, la escalabilidad **horizontal** se ve comprometida. Por ello, resulta más adecuada la escalabilidad **vertical**, que es más sencilla en estos casos.

2. Describe brevemente las diferencias principales del modelo documental con respecto al modelo clave-valor y al modelo relacional.

    Documental vs. Clave-valor
    En el modelo clave-valor cada agregado solo contiene **un par clave-valor**, mientras que en el documental cada agregado (documento) presenta una **estructura interna**, lo que lo hace menos **flexible** con respecto al anterior.
    Por otro lado, esta estructura compartida entre distintos documentos hace que el **acceso** a los datos sea más rápido y sencillo a través del SGBD, que pueden incluso introducir índices y crear colecciones.

    Documental vs. Relacional
    Ambos tienen una estructura interna; sin embargo, en el modelo relacional dicha estructura está definida de antemano, lo que implica tener que adaptar los datos a ésta, mientras que la estructura del modelo documental depende directamente de cómo estén estructurado los datos en los documentos. En otras palabras, la estructura de los datos en el modelo documental está implícita en los datos. Esto

puede conducir a situaciones en las que un mismo concepto del mundo real tenga estructuras diferentes al almacenarse en distintos documentos.

3. ¿Qué significa el término *schemaless*?

   El concepto *schemaless* define la falta de un esquema fijo o estructura fija previa **predefinida** al almacenamiento de los datos en la BD. Sin embargo, esto no implica necesariamente que no exista ningún tipo de esquema en los datos ya almacenados, ya que éstos pueden tener una estructura **implícita** interna que afecta a su distribución, pero ésta no está predefinida.

4. ¿Qué trata de resolver la estrategia *one size fits all*?

   Debido a la distinta naturaleza de los diferentes datos de una organización, estos pueden presentar **distintas necesidades** de tratamiento y almacenamiento, lo que aumenta la complejidad de su gestión al requerir de distintos productos e, incluso, tecnologías. Para simplificarlo, la estrategia *one size fits all* trata de adaptar todos los datos a un **único producto** de gestión, lo que implica la compraventa, formación, mantenimiento, etc. de un solo producto, simplificando compatibilidades y ahorrando recursos.

5. Explica brevemente las implicaciones del concepto de agregado en el contexto de los modelos de agregación.

   Al poder estar un mismo dato en varios agregados, cuando en uno de ellos dicho dato se modifica, la **consistencia** se puede ver comprometida ya que el SGBD no es responsable de asegurarla, sino que es la aplicación, y por tanto su desarrollador/gestor, quien debe velar por dicha consistencia. Por lo tanto, una de las implicaciones del concepto de agregado es esta **carga** sobre los programas.

   Sin embargo, la otra cara de la moneda es la facilidad de **acceso** a los datos gracias a su desnormalización y presencia en varios agregados.

   Otra implicación que resulta ventajosa es la alta capacidad de **distribución** de los datos, ya que tan solo hay que tratar con los agregados como unidades mínimas de gestión, lo que permite que éstos estén almacenados en distintos nodos del sistema. A diferencia de los datos repetidos en distintos agregados, la consistencia de las réplicas no idénticas distribuidas de un mismo agregado sí es responsabilidad del SGBD. Como consecuencia, la **escalabilidad horizontal** se ve favorecida.

Ejercicio 2 (30%)

A partir de la lectura de los apuntes (locuciones de los vídeos) de los temas I y II indica si te parecen ciertas o falsas las siguientes afirmaciones.

Para cada una de las afirmaciones indica si es cierta o falsa, **justificando la respuesta** mediante lo que has leído en los materiales. En **cada justificación deberá indicarse la cita de los apuntes, vídeo o libros en la que se sustenta**.

No serán válidas las respuestas que no se justifiquen o no tengan cita. Se valorará la concisión (una página y media para las 5 afirmaciones como máximo).

Afirmación 1

En un modelo de agregación, la estructuración de agregados de un mismo tipo no puede variar en ningún caso. Es decir, todos los agregados del mismo tipo deben seguir la misma estructura.

Falso. La estructura de un agregado no es explícita, sino que viene **implícita** en los datos que se insertan. Por ello, como cada agregado puede contener distintos datos (aunque sean del mismo tipo), su estructura variará en función de estos.

Página 7 de la locución del vídeo del tema 3: *Modelos de agregación: Características.* ([1])

Afirmación 2

En una arquitectura de almacenamiento *push*, para trabajar con los datos se requiere, en cualquier caso, que la información esté previamente almacenada en la base de datos.

Falso. Ese sería el caso de las arquitecturas *pull*. En el caso de las *push*, los flujos de datos en tiempo real se pueden transmitir de origen a destino **sin necesidad** de ser almacenados previamente en la base de datos, lo que agiliza su acceso.

Página 15 de la locución del vídeo del tema 2: *Persistencia Políglota*. ([2])

Afirmación 3

El esquema *map-reduce* presenta como problema que se transmitan datos irrelevantes a través de la red.

Falso. La función map se ejecuta localmente en cada nodo, seleccionando y transmitiendo **únicamente** los datos **relevantes** a través de la red; por lo tanto, se minimiza el transporte de datos irrelevantes.

Página 8 de la locución del vídeo del tema 3: *Modelos de agregación: Características.* ([1])

Afirmación 4

En general, en el modelo de agregación de columnas se organizan los datos por filas que se guardan en tablas. Cada fila constituye un agregado que incluye columnas y/o conjuntos de columnas, donde cada conjunto de columnas tiene un significado concreto.

Verdadero. Cada fila es un **agregado** que incluye un conjunto de columnas, formando cada una de ellas la tripleta nombre-valor-*timestamp*.

Página 14 de la locución del vídeo del tema 3: *Modelos de agregación: Tipos*. ([3])

Afirmación 5

Las bases de datos basadas en un modelo de grafo facilitan la escalabilidad horizontal.

Falso. En un modelo de grafo, los datos están altamente relacionados y se realizan transacciones ACID para asegurar su consistencia, lo que **dificulta** notablemente su distribución y, con ello, su escalabilidad horizontal.

Página 17 y 18 de la locución del vídeo del tema 4: *Modelos en grafo*. ([4])

Ejercicio 3 (20%)

Una empresa quiere desarrollar una *app* que permita procesar los datos que se obtienen de los relojes inteligentes cuando las personas corren en una competición deportiva. Se sabe que la información que ofrecen los diferentes relojes a los que está dirigida esta aplicación se encuentran en el mismo formato textual y estructural, y se quiere almacenar en una base de datos de tipo documental.

Se pide diseñar los tipos de documentos más eficientes para las siguientes dos consultas que se desean realizar:

Consulta 1: Interesa conocer información mensual acerca de los corredores que compitieron en ese mes agrupados por categoría (alevín, juvenil, senior, veterano). El objetivo es hacer un estudio de estacionalidad, por lo tanto, no consideraremos el año, sólo consideraremos los meses. Concretamente para cada mes y por cada categoría, se desea recuperar los datos del corredor (dni, club), y una lista de las competiciones realizadas en ese mes (tipo de competición -5k, 10k, 21k o 42k-, nombre de la misma, fecha y marca obtenida en horas, minutos y segundos)).

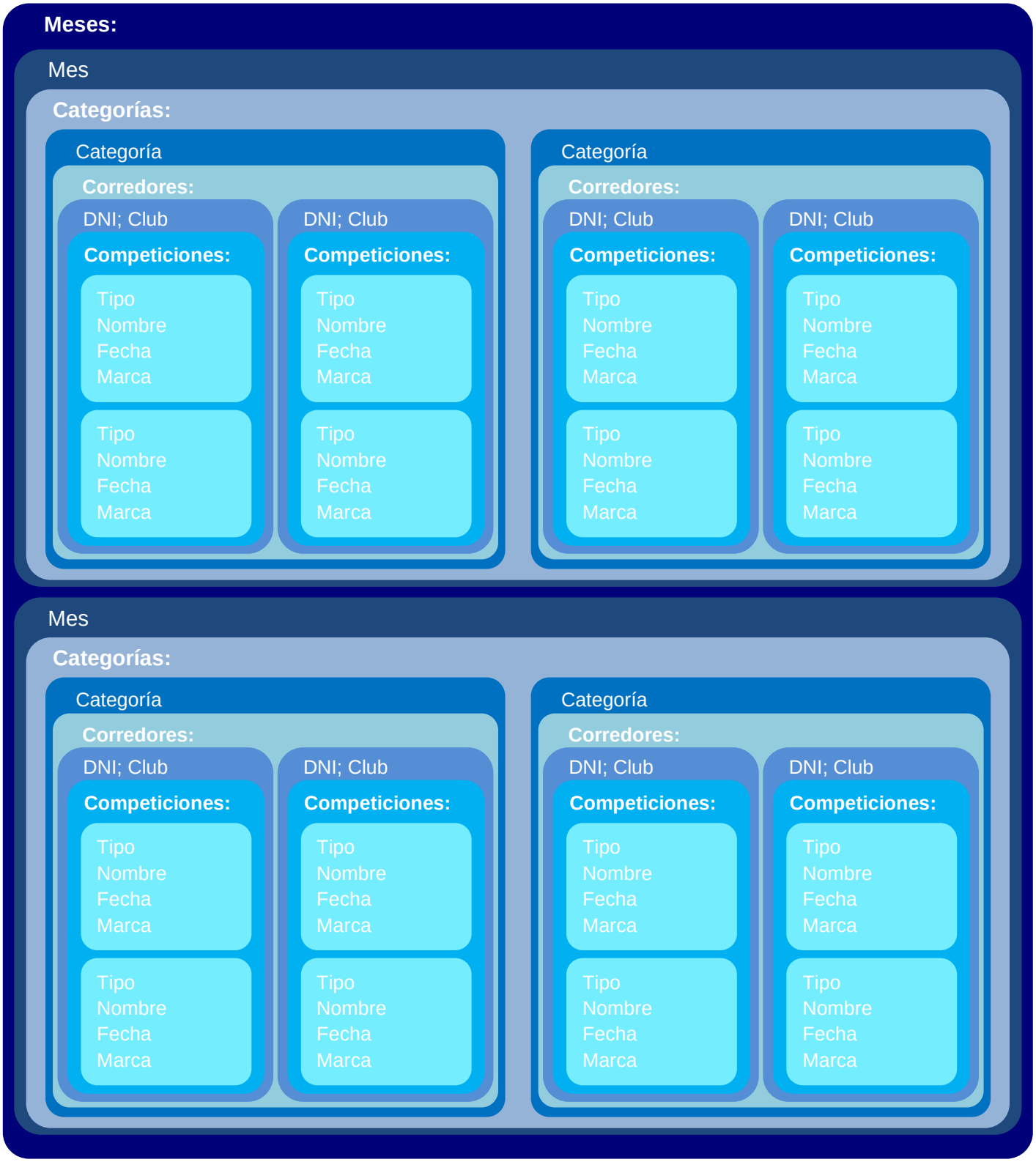
Consulta 2: También interesa conocer información de los resultados por club en las competiciones en las que han participado sus miembros. Concretamente para cada club y por cada competición (una competición queda especificada indicando su nombre y la fecha en la que se realizó), se desea recuperar los datos principales de cada corredor del club que ha participado en la misma agrupados por categoría (alevín, juvenil, senior, veterano): dni, tipo de participación (5k, 10k, 21k o 42k), y marca obtenida.

Cada consulta requiere un tipo concreto de documento. En este sentido, se pide indicar:

- Una representación gráfica del documento propuesto utilizando un diagrama de cajas anidadas como el que se explica en los apuntes sobre "Diseño de agregados" suministrado en el aula virtual.
- Una breve explicación de la estructura del documento y una justificación del porqué de la estructura propuesta.
- Un documento JSON que ejemplifique el resultado de una potencial consulta a cada agregado.

Consulta 1

Documento propuesto



Explicación

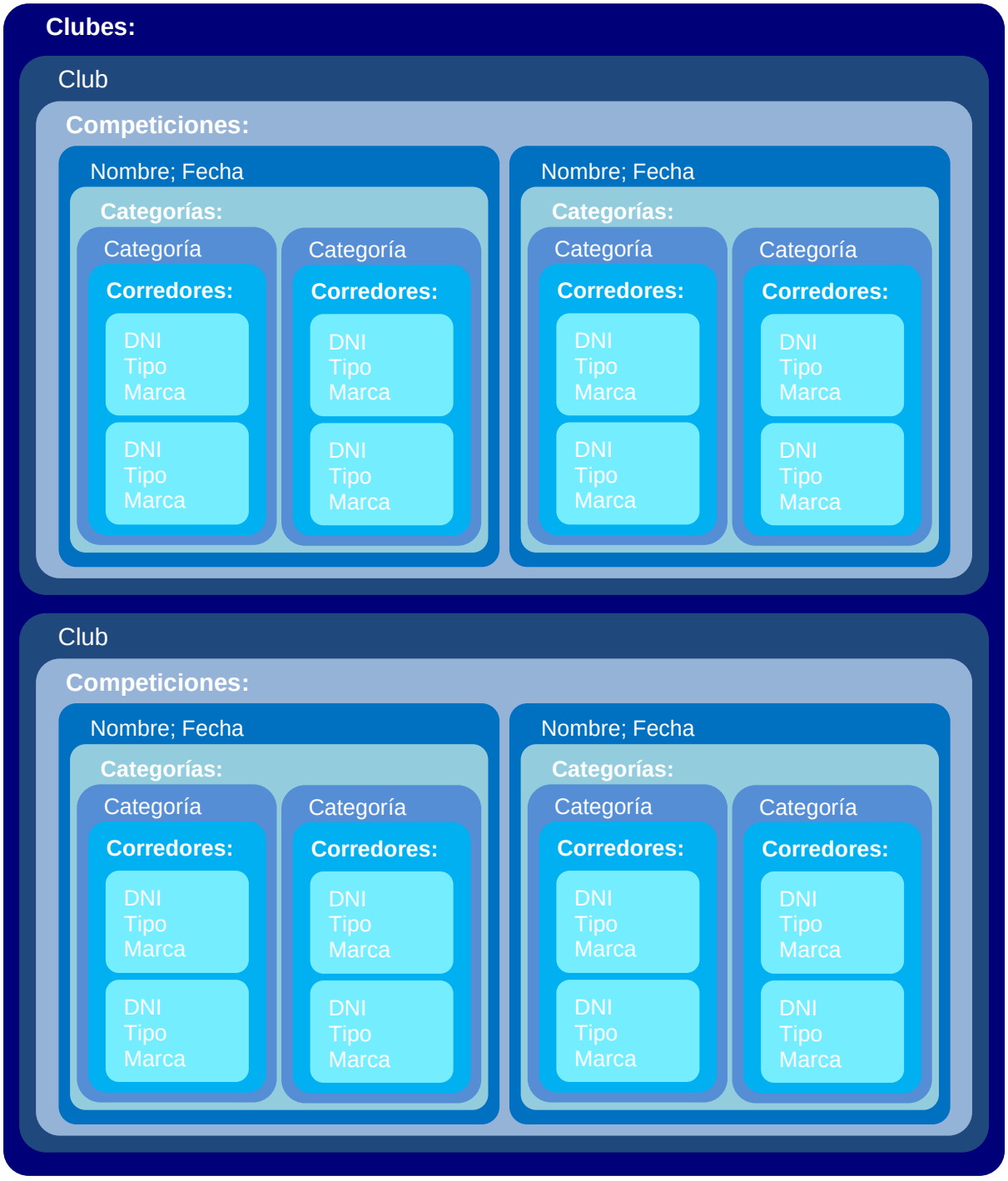
Los datos se piden estructurados, primeramente, por **Meses**; por ello, el primer nivel del documento es una lista que contiene agregados por cada **Mes**. A su vez, cada uno de éstos tienen una lista de las distintas **Categorías**, ya que se requiere agregar los datos según la categoría. Cada **Categoría** (alevín, juvenil, senior, veterano) incluye una lista de **Corredores** que almacena un agregado por cada corredor de dicha categoría. Los datos de cada **Corredor** se tratan de ***DNI***, el ***Club*** al que pertenece y una lista con las **Competiciones** en las que ha participado. Cada **Competición** de dicha lista contiene el ***Tipo*** de competición de la que se trata (5k, 10k, 21k o 42k), el ***Nombre***, la ***Fecha*** en la que tuvo lugar y la ***Marca*** que registró el corredor en particular.

JSON


```
{
  "Meses": [
    {
      "Mes": "enero22",
      "Categorías": [
        {
          "Categoría": "Juvenil",
          "Corredores": [
            {
              "DNI": "12345678A",
              "Club": "Spain Running",
              "Competiciones": [
                {
                  "Tipo": "10k",
                  "Nombre": "Popular de León",
                  "Fecha": "22-01-2022",
                  "Marca": "53:27"
                },
                {
                  "Tipo": "42k",
                  "Nombre": "NYC Marathon",
                  "Fecha": "30-01-2022",
                  "Marca": "2:49:13"
                }
              ]
            },
            {
              "DNI": "87654321Z",
              "Club": "Live & Let Run",
              "Competiciones": [
                {
                  "Tipo": "21k",
                  "Nombre": "NYC Half Marathon",
                  "Fecha": "30-01-2022",
                  "Marca": "1:57:59"
                }
              ]
            }
          ]
        }
      ]
    }
  ]
}
```

Consulta 2

Documento propuesto



Explicación

Los datos se piden agregados por clubes principalmente. Por lo tanto, el documento presenta, a primer nivel, una lista de **Clubes** que contiene un agregado para cada **Club**, que es el segundo nivel de agrupación requerido. Cada uno de éstos presenta una lista de las **Competiciones** en las que se ha participado que incluye, por cada **Competición**, su ***Nombre***, la ***Fecha*** en la que tuvo lugar y una lista de las **Categorías** que participaron. Esta lista contiene agregados de cada **Categoría** individual (alevín, juvenil, senior, veterano) que, a su vez, contiene una lista de **Corredores** perteneciente a cada una de ellas. Esta lista contiene agregados por cada **Corredor** de la misma, incluyendo cada uno de ellos el ***DNI***, el ***Tipo*** de participación (5k, 10k, 21k o 42k) y la ***Marca*** registrada.

JSON


```
{
  "Clubes": [
    {
      "Club": "Run Forest Run",
      "Competiciones": [
        {
          "Nombre": "NYC Marathon",
          "Fecha": "30-01-2022",
          "Categorías": [
            {
              "Categoría": "Senior",
              "Corredores": [
                {
                  "DNI": "12345678A",
                  "Tipo": "42k",
                  "Marca": "3:12:03"
                },
                {
                  "DNI": "87654321Z",
                  "Tipo": "42k",
                  "Marca": "3:00:51"
                }
              ]
            },
            {
              "Categoría": "Junior",
              "Corredores": [
                {
                  "DNI": "19283746X",
                  "Tipo": "42k",
                  "Marca": "2:43:39"
                }
              ]
            }
          ]
        }
      ]
    }
  ]
}
```

Ejercicio 4 (20%)

En el ejercicio 3 se ha propuesto una aplicación para procesar la información biométrica de los relojes inteligentes, y se ha supuesto que la información tenía el mismo formato y estructura. Sin embargo, en la realidad la información es heterogénea y puede encontrarse en diferentes formatos. Así mismo, hay que considerar otros aspectos tales como el gran número de datos que van a ser necesarios procesar, o la localización de los deportistas que hacen uso de la aplicación (supóngase que la aplicación solo se utiliza en España). Además, habrá que tener en cuenta que el tipo de relojes a los que se quiere dar cobertura puede cambiar en el futuro, añadiendo nuevos tipos, (y por tanto nuevos formatos de datos a procesar), y que el tipo de consultas y procesamientos de la información también son susceptibles de cambio.

Se pide realizar la lectura del artículo: "Data management in cloud environments: NoSQL and NewSQL data stores" que se puede encontrar en la siguiente dirección web:

https://cs.uwaterloo.ca/~david/cs848/papers/Data%20management%20in%20cloud%20environments-NoSQL%20and%20NewSQL%20data%20stores.pdf

Tomando como base la lectura del artículo así como los apuntes de los temas vistos, se pide responder de forma breve y concisa, pero justificada, a las siguientes preguntas acerca del supuesto planteado:

1. De acuerdo a las características del sistema que se plantea construir, ¿Qué tipo de base de datos sería mejor? NoSQL (¿de qué tipo?) /Relacional/ NewSQL/ Otros tipos de bases de datos.

   Dado que el principal inconveniente es el **distinto** y **cambiante** formato de los datos, los **modelos relacionales** resultan altamente **ineficientes**, ya que habría que definir multitud de procesos para adaptar e integrar los datos a las estructuras predefinidas. Además, estos procesos quedarían obsoletos con cualquier cambio en la fuente de datos, obligando a estar redefiniéndolos constantemente. Esta falta de flexibilidad también se presenta en las bases de datos **NewSQL**. Por lo tanto, los modelos que mejor se adaptarían a estas circunstancias son las **NoSQL.**

   Modelo NoSQL de Agregación de tipo Clave-Valor.
   Dado que el principal inconveniente es el distinto y cambiante formato de los datos, el modelo que mejor se adapta para realizar esta gestión es el de Clave-Valor. El SGBD **ignora la estructura** interna de los datos, lo que lo hace **robusto** frente a las diferencias y cambios en los formatos, ya que el valor del agregado se almacena como un *binary large object*. Esto implica que su estructura sea interpretada a nivel de aplicación. Además, los datos son accesibles a través de claves, siendo un sistema muy funcional cuando se trata de datos asociados a clientes o usuarios mediante un identificador, como en este caso.

Página 4 de la locución del vídeo del tema 3: *Modelos de agregación: Tipos*. ([3])
Data management in cloud environments-NoSQL and NewSQL data stores. ([5])

2. Describe breve y conceptualmente cómo serían las características del sistema en cuanto a disponibilidad, escalabilidad, consistencia de los datos, y distribución geográfica.

Disponibilidad.
Alta. Al carecer de estructuras predefinidas, los datos almacenados están disponibles desde el momento en el que se almacenan. Sin embargo, solo se pueden acceder a los datos a través de su clave ya que el valor es opaco para la base de datos, lo que añade complejidad a su accesibilidad.

Escalabilidad.
Alta. Es uno de los puntos fuertes de estos modelos debido a su poca rigidez. Esto permite incrementar el número de nodos de forma sencilla.

Consistencia.
Baja. Debido a que los datos se almacenan sin estructura previa, éstos no son sometidos directamente a procesos de transformación. Por lo tanto, la variabilidad en los tipos y estructuras de los datos puede hacer que surjan inconsistencias entre los mismos, que deberán ser tratadas a nivel de aplicación. Una consecuencia de la sencillez conceptual del modelo.

Distribución geográfica.
Alta. Presentan una gran eficiencia a la hora de distribuir los datos gracias a la flexibilidad que le otorga la falta de esquema.

Data management in cloud environments-NoSQL and NewSQL data stores. ([5])

3. Supón que, aparte de las consultas indicadas anteriormente, se quiere también obtener información sobre los corredores que han coincidido en las carreras y sus resultados. Las consultas concretas aún están por determinar y evolucionarán con el tiempo. Teniendo en cuenta esta novedad ¿Sería mejor una solución políglota en la que se usarán distintos modelos de bases de datos?

Esto añade un grado de complejidad al tener que **cruzar** los datos entre los agregados. Como el modelo Clave-Valor no almacena la relación entre los datos, esta tarea es altamente ineficiente. Por ello, **sí** sería conveniente incluir un modelo relacional que almacenase dichas relaciones para este tipo de consultas específicas en una solución políglota.

Locución del vídeo del tema 2: *Persistencia Políglota*. ([2])

Propiedad intelectual

Esta PEC ha sido realizada por Javier Gómez de Diego.

Todos los recursos utilizados y citados en la realización de esta PEC han sido provistos por la UOC.

Referencias

[1] M. E. R. Gónzalez, J. C. i. Caralt y P. U. Bayers, «B2_T3_2_ModelosAgregacionCaracteristicas».

[2] À. B. Muñoz, M. E. R. González y J. C. i. Caralt, «B1_T2_PersistenciaPoliglota».

[3] J. C. i. Caralt, M. E. R. González y P. U. Bayes, «B2_T3_3_ModelosAgregacionTipos».

[4] J. C. i. Caralt y M. E. R. González, «B2_T3_4_ModelosEnGrafo».

[5] K. Grolinger, W. A. Higashino, A. Tiwari y M. A. Capretz, «Data management in cloud environment: NoSQL and NewSQL data stores,» *Journal of Cloud Computing.*