IIC2413 – Bases de Datos

Examen

Instrucciones

El examen es individual, y el plazo (impostergable) para entregar el examen es el Lunes 13 de Diciembre, a las 23:59 hrs.

Puedes desarrollar el examen de la forma en que te sea más cómodo (latex, word, tablet, a mano), pero lo que debes subir a canvas es un único **PDF** con todas tus respuestas. Si decides escribir a mano, queda bajo tu responsabilidad que la letra sea legible.

Preguntas

1. (24 pts.) Usted esta preparándose para emprender con su propia *tech start up* y se encuentra en el proceso de diseñar la base de datos relacional de su MVP. La aplicación que va construir permite que personas ganen dinero de forma independiente haciendo las compras tanto del supermercado como de otras tiendas pequeñas para los clientes de la aplicación (wow, que creativo). La aplicación debe tener tiendas y sus respectivos catálogos (los cuales pueden ser modificados por los admins de las tiendas), los usuarios (tanto clientes como shoppers) y los pedidos hechos por los usuarios, los cuales consisten en listas de varias productos en distintas cantidades. Dado esto, usted debe construir lo siguiente:

   a) Un diagrama entidad relación con el diseño de la base de datos que permita construir la aplicación descrita (no es necesario detallar los atributos).

   b) El esquema relacional definido por el diagrama del inciso anterior.

   c) Considerando su esquema, escriba las siguientes consultas SQL:

      1) La cantidad de ordenes que ha hecho cada usuario junto con su email.
      2) El email de todos los shoppers que no han hecho ninguna orden.
      3) El id de la orden más cara junto con el id o email del usuario que la hizo.

2. (6 pts.) Considere una relación **T**(**a**, **b**, **c**) con 2.000.000 tuplas, almacenada en un disco con 20 tuplas por bloque ($B = 20$). Consideraremos el costo en términos de los bloques leídos del disco duro. Suponga que la siguiente consulta devuelve una tupla:

   ```
   SELECT a FROM T WHERE b = 7
   ```

   Indique el costo:

   a) Sin índices

   b) Con índice *hash* sobre la columna **b** (asumiendo *hashing* ideal)

   c) Con índice árbol B+ (2–3, un nodo por bloque, no en memoria principal) sobre la columna **b**

3. (9 pts) Para los siguientes 3 casos de uso, argumente que tipo de bases de datos NoSQL usaría usted. Además, considerando el triángulo CAP argumente cuales aristas de este deberían ser priorizadas por el sistema.

   a) Relaciones entre usuarios de una aplicación (ej: misma ip, misma tarjeta de crédito, mismo dispositivo, etc). Los datos serán usados para detección de fraudes.

b) Guardado de ids de usuario y su rating en un video juego, se harán muchas lecturas constantemente y deben ser de alta velocidad.

c) Base de datos de millones de artículos noticiosos, de todo el mundo en muchos idiomas distintos para alimentar una aplicación web tipo news.google.com.

4. (6 pts.)

   a) Indique qué relación lógica existe entre ℓ-diversidad y k-anonimato y argumente por qué.

   b) Indique la sensibilidad de la siguiente pregunta, y proponga la cantidad de ruido mínima para satisfacer ϵ-privacidad diferencial:

   ```
   SELECT SUM(nota) FROM IIC2413;
   ```

5. (15 pts) Se tiene una base de datos MongoDB cuyos documentos son la información de muchos millones de usuarios de una red social. Los documentos JSON tienen el mismo formato y no hay nulos. El siguiente documento es un ejemplo:

```
{
  "id": 12123,
  "country": "Chile",
  "created_at": 11-09-2002,
  "friends": [
    859,
    134,
    657,
    5345,
    12314
  ]
}
```

Notar que en esta red social la relación de amistad es siempre mutua.

Por motivos de analítica se quiere tener en un Data Warehouse SQL una tabla `friends_in_common(user_id_1, user_id_2, common_friends)`, que para cada par de amigos tiene la cantidad de amigos en común entre ellos. Se le pide a usted que diseñe un proceso map reduce que permita resolver este problema de forma eficiente mediante computación distribuida. Para eso debe definir ya sea en python o en pseudo código las siguientes 3 funciones:

- `orchestrate(n_mappers: Int) -> List[String]`: Esta función recibe como argumento la cantidad de mappers a utilizar para el procesamiento y retorna como output una lista de largo `n_mappers`, en que cada elemento es un string con una consulta MongoDB. Sólo deben preocuparse de retornar los strings de las consultas y asumir que serán ejecutadas y sus resultados pasados a un mapper.
- `execute_mapper(data: List[Dict]) -> List[Tuple]`: Esta es la función a ejecutar en cada mapper. Recibe como argumento el resultado de una de las consultas construidas en la función anterior y retorna una lista de tuplas donde cada tupla es un par `(key,value)`. No hay restricciones para el tipo de la key o el value excepto que la key debe ser inmutable.
- `execute_reducer(data: List[Tuple], sql_cursor) -> None`: Esta es la función a ejecutar en cada reducer. Recibe una lista de tuplas con pares `(key,value)` y un cursor con el método `.execute()`, que ejecuta consultas SQL de escritura en el data warehouse. La función debe ejecutar las inserciones que correspondan en la tabla `friends_in_common` del data warehouse.

Haga los supuestos que considere necesarios (con criterio) respecto al formato de los datos y de los métodos disponibles en el pseudocódigo. Explique todos los supuestos tomados y cosas en el código que le parezcan engorrosas de entender para el ayudante.

HINTS: Las llaves deberían representar de manera única a cada pareja de amigos para que los reducers puedan intersectar las listas de cada pareja. Averiguar de operadores disponibles en mongo podría servir para la función `orchestrate`.