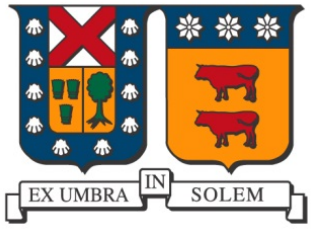
2019-09

PRIVACIDAD DIFERENCIAL EN BASE DE DATOS DE GRAFO: MÉTRICAS DE DISTANCIA

CIFUENTES VIVES, ANDRÉS EDUARDO

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO - CHILE



“PRIVACIDAD DIFERENCIAL EN BASE DE DATOS DE GRAFO: MÉTRICAS DE DISTANCIA”

ANDRÉS EDUARDO CIFUENTES VIVES

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Carlos Buil Aranda
Profesor Correferente: José Luis Martí Lara

Septiembre - 2019

AGRADECIMIENTOS

Al equipo docente del Departamento de Informática de la Universidad Técnica Federico Santa María, que me han otorgado todos los conocimientos necesarios para comprender los contenidos del área Informática. De esta forma, poder ser una persona capaz de realizar trabajos como el presente, como también estar capacitado para asumir proyectos complejos. Por otra parte, agradecimientos a los funcionarios del departamento, quienes ante necesidades siempre han estado dispuestos a ayudar. También a mis compañeros de carrera, con quienes he podido compartir y avanzar con el aprendizaje de contenidos.

Finalmente, una gran cantidad de agradecimientos a mi familia, quienes son los que más me han apoyado en el desarrollo de mi carrera. A mis padres por orientarme a seleccionar la carrera de Ingeniería Civil Informática, y que siempre me han motivado a seguir adelante con el avance de mi carrera.

RESUMEN

Resumen— En la actualidad, al preguntar por información privada suele prometerse que los datos se mantendrán anónimos. Aun así, el asegurarlos es complejo, ya que requiere técnicas que realmente lo garanticen. En bases de datos relacionales existen diversas técnicas. Sin embargo, en base de datos de grafo, utilizadas en aplicaciones como redes sociales y web semántica, el estudio de dichas técnicas es todavía incipiente. En esta memoria, se estudiará la relación entre privacidad diferencial con métricas de distancia de estas bases de datos, las cuales sirven como medida para conocer qué tan segura pueden ser ante eventuales ataques, protegiendo a los usuarios.

Palabras Clave— Privacidad, Base de Datos, Métricas

ABSTRACT

***Abstract*—** Nowadays, when asking about private information, it's common to guarantee that the data will remain anonymous. But securing the information it's a complex practice, because it needs techniques to genuinely anonymize the information. Diverse techniques exists for relational datatabases. However, this techniques aren't well researched for graph databases, used for social media and semantic web. In this research, the relation between differential privacy and distance metrics will be studied. This concepts, will ensure how secure can be the information in response of external attacks, protecting the identity of users.

***Keywords*—** Privacy, Databases, Metrics

GLOSARIO

- *Big Data*: Disciplina dentro de la ciencia de datos que estudia el uso de cantidades grandes de información, con el objetivo de ser utilizado para análisis y toma de decisiones.
- *Dataset*: Conjunto de datos. Muchas veces proviene de un motor de base de datos, los cuales tienen acciones de importación o exportación.
- *Hackathon*: Encuentro de programadores o desarrolladores, en los cuales en grupos de trabajo, se busca el desarrollo de un software o un hardware. El objetivo es aportar a la comunidad de software con desarrollos.
- *Mapeo*: Tabla de datos generada a partir de un dataset o subdataset. Este no pertenece directamente al conjunto de datos, pero es una representación de estos, con la finalidad de realizar un análisis del conjunto de datos.
- Metadato: Información extra relacionada a un dato. Esto lo que permite, es poder utilizar aquella información para poder obtener mayores detalles, o también extender funcionalidades de filtros y búsquedas.
- Serialización: Proceso de conversión de conjunto de datos a un formato almacenable. Utilizado generalmente para compatibilidad entre lenguajes de programación y optimización de rendimiento.
- *Subdataset*: Subconjunto de datos, el cual es extraído de un *dataset*. Estos son derivados desde el motor de base de datos, utilizando el *dataset* original, y algún filtro que origina al subconjunto.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La tecnología y sus usos van progresando con el tiempo, logrando cada vez ser un complemento indispensable en el diario vivir de las personas. En comparación a años anteriores una mayor cantidad de personas utiliza la Internet para poder realizar acciones de la vida cotidiana, tales como interacciones sociales, compras por Internet, entre otros, a través de aplicaciones dirigidas a ello.

Por otro lado, esto no puede funcionar sin el ente principal que hace de todo esto posible: la Información, la cual es almacenada en servidores utilizados por las aplicaciones anteriormente descritas. Por lo tanto en tiempos actuales, se puede decir que el uso de tecnología es muy importante para la interacción con el mundo, enviando y recibiéndose gran cantidad de información por segundo. Por otro lado, gran cantidad de esta información incluye información que puede identificar a una persona en particular, tales como su nombre, dirección, teléfono, entre otros. Esta es denominada información privada, debido a que describe con precisión a un individuo.

Privacidad puede definirse como todo lo que tiene relación con la vida personal de cada persona. Respecto a la información, las personas solamente con su consentimiento pueden optar a entregarla a otros individuos o entidades. Con el uso actual de la tecnología, muchas personas ignoran el concepto de datos privados junto a las políticas que aplicaciones decretan con el uso de estos, haciendo que sus usuarios opten por ingresar su información privada dentro de la plataforma.

Aun así, muchas de estas aplicaciones pueden requerir de esta información para una buena experiencia de usuario, pero a la vez tiene que prometerse que esta información tiene que permanecer privada. ¿Cómo se puede lograr tal compromiso?.

A través del tiempo junto al avance tecnológico, también ha avanzado mucho el desarrollo e investigación de la manipulación de información de la Internet. Entre ellos en el campo de Ciberseguridad, se encuentra la Privacidad diferencial, es cual es un método matemático que permite responder consultas sobre estas bases de datos, protegiendo al mismo tiempo la identidad de las personas.

¿Cómo se logra esto?. En este documento, se presentará la técnica para poder ser aplicada a un sistema de base de datos de grafo *RDF*, donde será estudiada en esta especificación de base de datos. Posteriormente, el concepto será aplicado a través de una implementación, la cual permitirá obtener resultados que señalarían si la técnica efectivamente cumple con su objetivo.

Estructura del documento

La presente memoria se divide en 5 capítulos. En capítulo 1 se desarrolla el problema que abordará el documento. En capítulo 2 se desarrolla el marco conceptual junto al estado del arte del tema, abordando en particular base de datos de grafo y avances en implementaciones de Privacidad diferencial. En el capítulo 3 se plantea una propuesta de solución para abordar el problema definido, la cual es una implementación de software con base en conceptos señalados en el marco conceptual. Capitulo 4 se desarrolla la validez de la solución propuesta, realizado pruebas que logren evidenciar que la solución propuesta puede ser útil en un contexto real. Finalmente en capítulo 5 de entregan las conclusiones del trabajo realizado en este documento.

CAPÍTULO 1
DEFINICIÓN DEL PROBLEMA

1 .1. Contextualización del Problema

1 .1.1. Introducción del Problema

El desarrollo de soluciones informáticas avanza considerablemente, dentro las cuales se proponen diversas soluciones que ayudan en generar nuevas aplicaciones las cuales mejoran la calidad de vida. Respecto a almacenamiento y procesamiento de información, las bases de datos de grafo son una de estas soluciones para aplicaciones que utilizan información conectada (redes sociales, web semántica, entre otras), debido a que logran tanto modelar como procesar de forma más natural este paradigma de datos. Por otro lado, otra tendencia de peso es el *Big Data*, el cual genera una gran utilidad permitiendo que grandes volúmenes de información puedan ser utilizados para tener mayor conocimientos o mejores predicciones sobre eventos. Es utilizado de gran forma en servicios web, Internet de las cosas, mejora de procesos, entre otros. Con el paso del tiempo, el volumen de información se incrementará de forma más drástica, requiriendo de diversas formas de control para poder ser utilizado.

Aun así junto a este crecimiento, también surgen nuevas vulnerabilidades de seguridad, siendo más atractivo para potenciales atacantes u personas con conocimientos el hacer un uso indebido de los datos. En el uso actual de la Internet, muchas aplicaciones para poder entregar una experiencia óptima al usuario, consultan a estos sobre información confidencial, como dirección, nombre, teléfono, ubicación, entre otros. La privacidad de estos datos es un derecho humano para los usuarios, por lo que existe una gran responsabilidad en que los servicios de Internet deben prometer que esta privacidad se mantendrá resguardada al ser recolectada. Aun así, el hecho de asegurar privacidad es un procedimiento complejo, debido a que en una implementación siempre puede haber vulnerabilidades las cuales al ser descubiertas, ya no logran el objetivo de resguardar la privacidad de los usuarios. Por lo anterior, se tiene que ser cauteloso en las propuestas de solución.

Por otro lado, el crecimiento del uso y volumen de información está siendo de gran utilidad para investigaciones de negocio junto a la mejora de toma de decisiones. Por esto, el interés tanto de la aplicación con el estudio de las bases de datos de la empresa es fundamental para comprender el mercado de esta organización. Aun así, el uso directo de esta puede generar vulnerabilidades, como cambios no deseados de información, operaciones de administración peligrosas para la integridad, entre otros riesgos. El no tener cautela tiene la consecuencia de no lograr la protección de la privacidad de la personas. Es por ello, que se utiliza el concepto de *dataset* analíticos.

Este tipo de conjunto de datos son creados desde la base de datos original, pero a diferencia

de estas son utilizados específicamente para fines estadísticos, permitiendo aislar la manipulación directa de la información del negocio. Con el objetivo de evitar obtener datos privados de algún individuo en particular, estos antecedentes son modificados en otro tipo de representación (histogramas, categorías, rango de datos, entre otros) a través de trasformadores, los cuales entregan el conjunto de respuestas en particular sin evidenciar los participantes de esta.

Aun así, existen diversas vulnerabilidades en estos conjuntos de datos, las cuales abren la posibilidad para ataques a estos conjuntos que vulneran la privacidad de sus participantes, a través de técnicas de desanonimato de información las cuales se explican a continuación.

1 .1.2. Vulnerabilidad de base de datos analíticas

Un ejemplo de vulnerabilidad ocurrió en Estados Unidos, donde se unieron registros médicos públicos de 1997 para investigación junto a información derivada del censo de ciudadanos en el año 2003. Con la unión de los datasets, se ha podido descubrir la enfermedad que padecía un gobernador en particular, con solo saber su código postal, el género y la edad (tabla 1)([C. Barth-Jones, 2012]). Este hallazgo hace un hincapié en aplicar técnicas de privacidad dentro de los *dataset* clínicos para la protección personal de los pacientes.

Otro caso más reciente fue con las plataformas *Netflix* e *Internet Movie Database*, donde en la primera se ha publicado un set de datos estadísticos en un evento *Hackathon* denominado *"Netflix Prize"* durante el año 2009, el cual tiene el fin de mejorar el sistema de recomendaciones de películas. Los publicistas del *dataset*, señalan que se ha aplicado técnicas de perturbaciones y anonimato a este, por lo que se promete que este se mantiene privado. Un estudio de ciberseguridad, en el cual se estudian técnicas de desanonimato de información con el fin de descubrir vulnerabilidades, logran conocer a través de un algoritmo de unión de datasets las preferencias de usuarios particulares de tanto *Netflix* como *Internet Movie Database* sobre las preferencias por algunas películas, las cuales tienen políticos y religiosos [Narayanan y Shmatikov, 2008]. Lo anterior se logra a través del *dataset* estadístico de *Netflix*, y con contenidos sobre películas del sitio *Internet Movie Database*, por lo que el primero falla en la aplicación de privacidad.

Por lo tanto, el asegurar la privacidad de los usuarios no es una tarea fácil, requiriendo de investigación tanto de prácticas como técnicas que permitan resguardar la información, y que esta no logre ser utilizada para poder descubrir información de individuos. Por otro lado, la información no siempre puede ser perfectamente privatizada, debido a que el resultado de poder tener una privacidad perfecta, sería entregar resultados que no tengan una relación con el resultado real, por lo que es tarea también de la entidad que resguarda la información, de encontrar formas de entregar privacidad real.

Nombre	Nacimiento	Sexo	Cód. Postal
Alice	1960/01/01	F	10000
Bob	1065/02/02	M	20000
Cathy	1970/03/03	F	30000
David	1975/04/04	M	40000

(a) Registro de Votantes

Nacimiento	Sexo	Cod. Postal	Enfermedad
1960/01/01	F	10000	flu
1065/02/02	M	20000	dyspepsia
1970/03/03	F	30000	pneumonia
1975/04/04	M	40000	gastritis

(b) Registros Médicos

Tabla 1: Unión de datos anonimizados, demostrando que esto no logra poder entregar privacidad real. El ataque es realizado con una unión de dos consultas en *dataset* diferentes, obteniendo las enfermedades de votantes a través de las columnas en cuadro rojo.

1 .1.3. Problema a abordar

En el campo de base de datos relacionales, se han estudiado métodos para anonimato de información tales como *K-Anonimato*, y otros de entrega de resultados con perturbación como Privacidad diferencial. Este último, ha entregado muy buenos resultados en años recientes, ya que es un método que logra asegurar privacidad en mayor medida que otros existentes, siendo utilizado por empresas como *Google* o *Apple* para estadísticas de uso de sus aplicaciones. Aun así, para base de datos de grafo *RDF*, esta técnica no ha sido actualmente muy estudiada o puesta en práctica. Esto se debe a su diferencia en uso respecto a otros modelos como el relacional[1].

Con lo anteriormente descrito, surge el problema de saber cuándo el *dataset* generado para analíticas desde una base de datos de grafo puede declararse segura, cumpliendo con el requerimiento de proteger privacidad. En esta memoria se trabajará en base a lo anterior, estudiando la influencia de la estructura de una base de datos de grafo, relacionando sus elementos con el concepto de Privacidad diferencial. Posteriormente continuando la investigación anterior, crear una implementación de software que efectivamente logre entregar resultados que resguarden la privacidad de los participantes de la base de datos de grafo, entregando resultados a consultas que sean útiles para uso analíticos.

[1]Esto se demuestra a través del sitio *DB-Engines*, que muestra un *ranking* de uso de paradigmas de base de datos. https://db-engines.com/en/ranking

1 .2. Objetivos

1 .2.1. Objetivo General

- Adaptar el uso de Privacidad Diferencial a grafos, analizando componentes de distancia, para asegurar la privacidad de la información de usuarios.

1 .2.2. Objetivos Específicos

- Aplicación de Privacidad diferencial tanto de forma teórica como práctica en bases de datos de grafo *RDF*, para aplicación de privacidad en información.
- Implementar una solución aplicando Privacidad diferencial en *RDF*, la cual permita obtener resultados seguros ante ataques.
- Determinar parámetros adecuados dentro de la implementación, con el objetivo de obtener resultados útiles para el usuario.

CAPÍTULO 2
MARCO CONCEPTUAL

2 .1. Datasets Estadísticos

Son *datasets* derivados de una base de datos en producción, con el fin de entregar información para su estudio. Se crean realizando una previa preparación de la información, la cual filtra por los campos que sean útiles para el estudio (se remueve por ejemplo información privada de los participantes como contacto, u otros que no se relacionen con el estudio). También se realiza la agrupación de cierto conjunto de valores que permiten ser categorizados, tales como edad o género. Esta preparación, logra el resultado de cumplir con la ética de la compañía de no difundir información directamente de sus usuarios, entregando información derivada de esta para futuras investigaciones o aporte al estado del arte.

Como se menciona anteriormente, la información de estos *datasets* tiene que asegurar la protección de privacidad de los usuarios. Se ha descrito en la definición del problema que no es posible encontrar una técnica de privacidad perfecta, por lo que la información de estos *datasets* tiene que aplicar diversas condiciones que puedan cumplir el objetivo de la mejor forma posible. Por ello, la entrega de estos tiene que tener solamente la información que sería útil para el estudio, no tener relaciones en si con otras fuentes de información, y aplicar técnicas de privacidad.

2 .2. Bases de Dato de Grafo

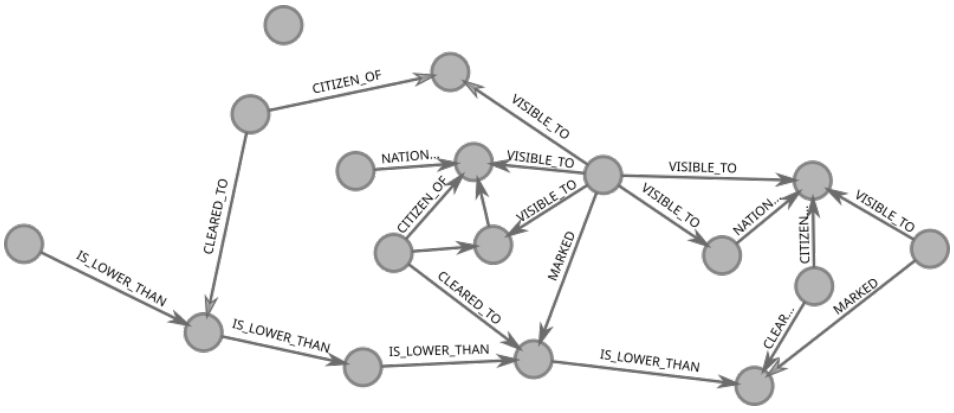


Figura 1: Ejemplo de visualización de base de datos de grafo.

Con el avance de la tecnología, surgen nuevas implementaciones de base de datos en relación a las necesidades. Previo a los años 2000, el uso de base de datos relacionales dominaba como solución para administración y almacenamiento de información. Con el surgimiento de

la Web 2.0, comienzan a surgir nuevos motores conocidos como *NoSQL*, los cuales consisten en otro tipo de implementación de base de datos que escapan del paradigma relacional, permitiendo el uso de otros tales como basado en columnas, basado en objetos, entre otros. Este cambio de tipo de paradigma, puede hacer de la base de datos a seleccionar más eficiente para un determinado fin, debido a que su implementación se adapta mejor al modelo de datos.

Dentro de estos motores, existe la base de datos basada en grafo, el cual su aplicación principal es utilizar el concepto de grafos perteneciente a las ciencias de la computación, para poder relacionar la información.

Una base de datos de grafo se compone principalmente de nodos y arcos. Los nodos o vértices, son la estructura principal, la cual representa una entidad. Este puede contener propiedades o etiquetas. Otra característica importante son las relaciones, representadas a través los arcos de un grafo. Relacionan dos o más nodos. Pueden tener dirección y propiedades, siendo útiles por ejemplo para poder definir el peso de un arco.

Las ventajas de este tipo de base de datos son múltiples [Patil *et al.*, 2014]:

- Rendimiento: A diferencia de una base de datos relacional, al no tener la dependencia de índices, las base de datos de grafo añaden un gran potencial de rendimiento en consultas de información relacionada. Como es un grafo, el tiempo de ejecución de una consulta, es proporcional al tamaño del grafo generado, y no de la unión de tablas como el caso de base de datos relacionales.
- Flexibilidad: Para la implementación de lógica de negocio, se requiere de flexibilidad en el motor de base de datos a utilizar. Por ejemplo utilizando el modelo relacional, se quiere utilizar un modelo de grafos. Si el administrador quisiera agregar una tabla o una relación, tendría que realizar modificaciones dentro del modelo para adaptarse al uso de grafos, lo que generaría mucho trabajo. En cambio con base de datos de grafo, lo que se tendría que hacer simplemente es conectar nodos con un arco, o agregar un nodo de ser necesario.
- Agilidad: Volviendo al rendimiento, este tipo de base de datos hace eficiente su uso debido a su implementación. Esto también entrega mayor agilidad en entregar la información, realizando consultas más directas (véase figura 2) a través de un lenguaje más descriptivo, a diferencia del modelo relacional el cual utiliza un lenguaje declarativo (por ejemplo, como *SQL*) [Polikoff, 2014].

En la actualidad, los usos importantes que tiene este tipo de base de datos es en *ranking* de paginas, interacciones sociales, administración de redes, bioinformatica, entre otros usos.

Dentro de esta investigación, se definirá una base de datos de grafo como un conjunto de datos (G, l), donde G corresponde al grafo que determina esta base de datos (nodos y arcos),

```
SELECT ?medication
WHERE {
?diagnosis a example:Diagnosis .
?diagnosis example:name "TB of vertebra" .
?medication example:canTreat ?diagnosis .
}
```

(a) Lenguaje *SPARQL* - Base de datos de grafo

```
SELECT DRUG.medID
FROM DIAGNOSIS, DRUG, DRUG_DIAGNOSIS
WHERE DIAGNOSIS.diagnosisID=DRUG_DIAGNOSIS.diagnosisID
AND DRUG.medID=DRUG_DIAGNOSIS.medID
AND DIAGNOSIS.name="TB of vertebra"
```

(b) Lenguaje *SQL* - Base de datos relacionales

Figura 2: Comparación de lenguajes de base de datos relacional y basado en grafo.

l, son las etiquetas de los vértices y arcos de G. Este conjunto de etiquetas de ser necesario, también puede utilizarse para almacenar una mayor cantidad de información dentro de los nodos y arcos. Para simplificación de los conceptos, se considerara a las etiquetas como un ente de información descriptivo.

2 .3. Privacidad diferencial

Es una definición matemática, la cual consiste en que a partir de la consulta sobre un resultado, este es perturbado en cierto grado para que la respuesta entregada sea una aproximación de la real, con el fin de resguardar privacidad. El método asegura que la capacidad de un adversario de realizar algún ataque a una base de datos (para encontrar por ejemplo información privada de un individuo) se mantiene en la misma probabilidad, independiente si el individuo atacado participará o no dentro del *dataset* [Dwork, 2011].

Como se ha mencionado anteriormente, los *datasets* analíticos tienen que tener mecanismos de privacidad para que esta no sea vulnerada ante ataques. Existen técnicas que pueden convertir la información en otro tipo de formato (agrupaciones por algún atributo como edad, género, entre otros). Aun así, la información de alguna manera tiene que ser también perturbada, de lo contrario surgen los problemas descritos en la definición del problema (sección 1 .1.2). Es bajo este contexto que se investiga el concepto de Privacidad diferencial, el cual a través de agregación de ruido al resultado real, hace más difícil la tarea de poder relacionar este resultado con otro tipo de información auxiliar.

Por ejemplo, puede que algún usuario busque obtener las preferencias musicales de una persona, realizando una consulta $f(D)$ dentro de una base de datos D. En primera instancia, recordar que se utilizan *dataset* analíticos, por lo que la información de la persona en particular no se encontraría disponible inmediatamente para responder la inquietud del usuario. Pero aun así, si se puede obtener información con la cual su respuesta tiene relación con otro *dataset*, entonces se vulneraría la privacidad.

Con Privacidad diferencial, se puede responder la consulta original que realiza el usuario anteriormente descrito, pero también resguarda de forma efectiva la privacidad de todos los participantes. Para ello, se le agrega ruido a la respuesta final entregada al consultor, lo que produce que este obtenga una aproximación de la respuesta que busca, la cual es útil para propósitos de estudio del conjunto de datos, pero no para encontrar las preferencias de un participante.

Regresando al proceso de obtención de $f(D)$, se define una base de datos como un *dataset* D. De este *dataset*, se puede crear otro denominado D', el cual se obtiene substrayendo o añadiendo un elemento a D. Este nuevo *dataset* se denomina vecino de D, con una distancia de 1 (debido a que es una entidad la que cambia) (véase sección 2 .5). Con ello, se puede definir un concepto de distribución de probabilidad de obtener $f(D)$ en algún *dataset* D, con la notación $Pr(f(D)|Q(D))$. Con lo anterior, se define el concepto de Privacidad diferencial como [Dwork, 2011]:

$$\frac{Pr(f(D)|Q(D))}{Pr(f(D)|Q(D'))} \leq e^{\epsilon} \tag{1}$$

Donde:

- $Pr()$: Distribución de probabilidad de encontrar resultado de consulta en *dataset*.
- $f(D)$: Resultado de una consulta $f(D)$ que un atacante quiere encontrar en la base de datos.
- D : *Dataset* real.
- D' : *Dataset* vecino del real.
- $Q()$: Algoritmo que crea el *Dataset* estadístico.
- ϵ : Grado de Privacidad diferencial a aplicar en la relación. Depende del grado deseado por el usuario.

Como se observa, el concepto relaciona la probabilidad que el resultado real de una consulta se encuentre dentro del *dataset* real como también del *dataset* vecino generado. El resultado de lo anterior tiene que ser menor o igual al parámetro e^{ϵ}, el cual determina la relación entre la probabilidad de encontrar un resultado especifico en el *dataset* real y en sus vecinos.

El caso ideal de Privacidad diferencial, ocurre cuando el valor de una consulta hacia la base de datos es el mismo tanto para el conjunto de datos real como para los vecinos ($e^{\epsilon} = 1$, con

$\epsilon = 0$). Esto describiendo efectivamente, que dependiente de la presencia de un individuo, el resultado de las consultas sigue siendo el mismo. En caso contrario, si el valor de e^{ϵ} es mayor a 1, aumentan las posibilidades que el resultado de una consulta a D y D' sean muy diferentes, lo que evidencia que la presencia de un determinado individuo, genera cambios en el resultado de la consulta. Lo anterior, entrega mayores evidencias de el valor asignado al individuo removido de la base de datos.

Por lo tanto, se busca que la distribución de probabilidad de ambos conjuntos, sea lo más similar posible.

El grado de privacidad, denominado como ϵ, representa lo siguiente: si este es mayor o igual a 1, quiere decir que el algoritmo aplicado a la base de datos no entrega seguridad, existiendo una gran probabilidad que un valor sacado del *dataset*, cambie el valor de la respuesta. Ahora si este es menor a 1, entonces quiere decir que es segura: mientras menor sea el valor, se puede asegurar una mayor cantidad de privacidad. Es por ello que por lo general, se utilizan valores como $0,1$ o $0,01$. La elección de estos valores, depende del uso del *dataset* analítico. Esto es lo que se conoce como Privacidad diferencial estricta ($\epsilon, 0$), debido a que protege en general la información de todo el conjunto.

Por otro lado, debido a que no existe un grado de privacidad perfecto, la privacidad entregada puede obtener un margen de error. Por ello existe el concepto (ϵ, δ) Privacidad diferencial, donde se tiene un parámetro δ, el cual puede entregar un margen de vulnerabilidad a la Privacidad diferencial agregada para poder encontrar la información de algún individuo. Lo que hace particularmente, es "relajar" en cierta medida la privacidad aplicada a cambio de un pequeño factor de riesgo, entregando otro margen de resultados. Como se ha señalado, esto depende en gran medida del como se quiere trabajar la información del *dataset*. La relación de Privacidad diferencial queda expresada de la siguiente forma:

$$Pr(f(D)|Q(D)) \leq e^{\epsilon} Pr(f(D)|Q(D')) + \delta \tag{2}$$

El parámetro δ puede ser definido de múltiples formas dependiendo de las características de un *dataset*, pero este tiene que ser lo más insignificante posible, pues de lo contrario puede filtrar información de individuos [Dwork y Roth, 2014]. Es un número que se le agrega a la distribución de probabilidad del *dataset* vecino D', por lo que esta aumenta para encontrar el valor de cierto individuo. Si existen n individuos en un *dataset*, entonces puede ser probable en $\delta * n$ veces, que la privacidad de cierto individuo pueda vulnerarse. Es por lo anterior, que se tiene que intentar que el valor de δ sea lo menor posible, en términos de las características del conjunto de datos.

Aun así con la definición anterior, no se puede llevar a la práctica para entregar resultados que logren preservar privacidad. Por ello, se adapta la definición anterior de Privacidad diferencial a lo siguiente:

$$Pr(\mathcal{A}(D) \in S) \leq e^{\epsilon} Pr(\mathcal{A}(D') \in S) + \delta \tag{3}$$

Donde:

- $\mathcal{A}$: Función que retorna el resultado de una consulta, pero que esta es perturbada con un ruido aleatorio.
- D : *Dataset* real.
- D' : *Dataset* vecino del real.
- S : Subconjunto de $rango(\mathcal{A})$

La función aleatoria $\mathcal{A}(D)$ es aquella que entrega resultados que protegen la privacidad, esto con el ajuste de los parámetros de Privacidad diferencial (ϵ, δ). Esta función por otro lado, tiene que utilizar el concepto de probabilidad anteriormente mencionado.

2 .4. Trasformador *JOIN*

Para el trabajo con subconjuntos de base de datos, es necesario realizar ciertas transformaciones $T : \mathbb{D} \to \mathbb{D}$ (donde $\mathbb{D}$ es universo de base de datos) en los conjuntos de datos presentes para poder trabajar con ellos. Se utilizarán diversos conceptos de Álgebra Relacional [Codd, 1972] para exponer las transformaciones necesarias de *datasets*.

Para trabajo con semánticas de base de datos, se define el concepto de Transformadores ($\mathcal{T}(D)$) como conjuntos derivados de datos de otros subconjuntos a través de operaciones de teoría de conjuntos.

Dentro de las múltiples operaciones de Trasformaciones, se tiene la operación *JOIN*, la cual a partir de un atributo en común entre dos bases de datos, crea un mapeo de estos en una sola tabla. A continuación se definen los diferentes tipos de operaciones *JOIN* más utilizados:

***JOIN* natural ($\bowtie$)** Este tipo de *JOIN* es un operador binario, el cual trabaja con dos conjuntos de datos R y S. Su resultado, es un conjunto de todas las combinaciones de tuplas pertenecientes a R y S, las cuales tienen atributos en común.

En álgebra relacional, se define como:

$$R \bowtie S = \{r \cup s | r \in R \wedge s \in S \wedge F(r \cup s)\} \tag{4}$$

Donde $F()$ es un predicado lógico (es decir, una función que retorna Verdadero o Falso), dependiendo de los atributos en común del conjunto de datos. Es necesario que los conjuntos

R y S tengan un atributo en común, de lo contrario, se retornaría la operación *JOIN* de todos los datos (producto cartesiano $R \times S$).

Estudiante	ID	Asignatura
Ana	234	Programación
Juan	122	Cálculo
Pablo	144	Seguridad
Joaquín	253	Cálculo

(a) Estudiantes

Asignatura	Profesor
Cálculo	Nicolás
Seguridad	Carla
Programación	Rafael

(b) Profesor

Estudiante	ID	Asignatura	Profesor
Ana	234	Programación	Rafael
Juan	122	Cálculo	Nicolás
Pablo	144	Seguridad	Carla
Joaquín	253	Cálculo	Nicolás

(c) Estudiante $\bowtie_{Asignatura}$ Profesor

Figura 3: Ejemplo de *Join*. Se determina que profesores (tabla 3b) tienen los estudiantes (tabla 3a) a través de la transformación (tabla 3c).

Equijoin* y θ*-JOIN Suponer que a diferencia del *JOIN* natural, se quiere colocar condiciones para realizar comparaciones entre datos. Por ejemplo, se quiere comparar el precio de una cafetera con un moledor de granos de café. Se quiere retornar solamente los valores, los cuales el valor de la cafetera, sea mayor al precio del moledor de granos, como se observa en figura 4.

Lo anterior, es lo que se conoce como un θ*-JOIN*, debido a que se utiliza una relación de condición para poder realizar las uniones entre datos.

$$R \bowtie_\theta S = \sigma_\theta(R \times S) \tag{5}$$

Donde θ, corresponde a una operación lógica tales como $=, <, >, \leq, \geq$.

Por otro lado, si esta condición de unión fuera la operación de equivalencia "=", a esta es la que se conoce como un *Equijoin* ($\bowtie_=$).

A diferencia de un *JOIN* natural, los *Equijoin* solamente retornan las filas, las cuales cumplen con la condición propuesta. Es decir, aquellas filas que efectivamente cumplan con la relación $r = s$.

***Semijoin* ($\ltimes$)** Esta operación es similar al *JOIN* natural, pero con la diferencia para el caso de un *Semijoin* en el lado derecho ($R \ltimes S$), se retornar aquellas tuplas de R, de las cuales existe una tupla en S los cuales tienen atributos en común, como se muestra en la figura 5.

Cafetera	Precio ($)
Cafetera 1	80000
Cafetera 2	120000
Cafetera 3	160000

(a) Cafeteras

Moledor	Precio ($)
Moledor 1	30000
Moledor 2	90000
Moledor 3	170000

(b) Moledores

Cafetera	Precio Cafetera ($)	Moledor	Precio Moledor ($)
Cafetera 1	80000	Moledor 1	30000
Cafetera 2	120000	Moledor 1	30000
Cafetera 2	120000	Moledor 2	90000
Cafetera 3	160000	Moledor 1	30000
Cafetera 3	160000	Moledor 2	90000

(c) Cafeteras $\bowtie_{\geq}$ Moledores

Figura 4: Ejemplo de θ-*JOIN*, con θ siendo comparación de mayor o igual. Se compara precio de cafeteras (tabla 4a) con moledores de granos de café (tabla 4b). La transformación, muestra aquellos resultados, los cuales el precio de la cafetera, es mayor al precio del moledor (tabla 4c).

Estudiante	ID	Carrera
Ana	234	Civil
Juan	122	Industrias
Pablo	144	Civil
Joaquín	253	Informática

(a) Estudiantes por Carrera

Carrera	Centro Alumnos
Industrias	Juan
Civil	Ana

(b) Centro de alumnos de Carrera

Estudiante	ID	Carrera
Ana	234	Civil
Juan	122	Industrias

(c) Estudiantes $\ltimes$ Centro de alumnos

Figura 5: Ejemplo de *Semijoin*. A partir de juntar estudiantes con centro de alumnos (tabla 5a y 5b), se obtiene una transformación con los ID de los estudiantes que son parte del centro de alumnos de su carrera (tabla 5c).

Su expresión matemática es la siguiente:

$$R \ltimes S = \{t : t \in R \wedge \exists s \in S(F(t \cup s))\} \tag{6}$$

Donde se define $F()$ como la función de *JOIN* natural.

Anitjoin (▷) Operación similar al *Semijoin*, pero donde el resultado entregado para $R \triangleright S$, son aquellas tuplas de R, de las cuales no existen tuplas en S que tengan atributos equivalentes.

Estudiante	ID	Carrera
Ana	234	Civil
Juan	122	Industrias
Pablo	144	Civil
Joaquín	253	Informática

(a) Estudiantes por carrera

Carrera	Centro Alumnos
Industrias	Juan
Civil	Ana

(b) Centro de alumnos de carrera

Estudiante	ID	Carrera
Pablo	144	Civil
Joaquín	253	Informática

(c) Estudiantes ▷ Centro de alumnos

Figura 6: Ejemplo de Antijoin. A partir de unir estudiantes con centro de alumnos (tabla 6a y 6b), se obtiene una transformación, con los ID de los estudiantes que no son centro de alumnos de su carrera (tabla 6c).

$$R \triangleright S = \{t : t \in R \wedge \neg \exists s \in S(F(t \cup s))\} \tag{7}$$

Para este trabajo, se trabajará con la operación de *Equijoins* ($R \bowtie_{=} S$), por lo que considerar como la operación por defecto de los transformadores.

2 .4.1. Operaciones de agregación

Para estadísticas, no interesa el uso directo de la base de datos, si no se busca obtener información a partir de estos datos. Por lo tanto, la mayoría de los motores existentes integran diversos tipos de consultas de agregación, que responden información basada en la real.

Estas consultas pueden ser diversas, incluso dan la opción a los consultores para definir sus propias operaciones, para poder obtener respuestas más precisas a sus requerimientos.

Entre las operaciones básicas (ejemplos en tabla 7), se encuentran las siguientes:

- ***COUNT()*** : Retorna la cantidad de resultados, tanto numéricos como discretos, que cumplen con el patrón de la consulta entregada como argumento.
- ***SUM()*** : Utilizada solo para valores numéricos. Realiza una adición de todos los resultados filtrados por la consulta realizada.
- ***AVG()*** : Utilizada solo para valores numéricos. Retorna el promedio de los valores filtrados por la consulta realizada.

Cafetera	Precio	Tipo
Cafetera 1	80000	Manual
Cafetera 2	100000	Manual
Cafetera 3	120000	Manual
Cafetera 4	130000	Automática
Cafetera 5	160000	Manual
Cafetera 6	220000	Automática

(a) Tabla de Cafeteras.

Consulta	Resultado
COUNT(“cafetera WHERE tipo = manual”)	4
SUM(“precio WHERE tipo = automática”)	350000
AVG(“precio”)	135000

(b) Consultas de agregación

Figura 7: Ejemplo de aplicación de consultas de agregación, donde se tiene una tabla con cafeteras y sus atributos (tabla 7a), junto a las operaciones mencionadas con sus resultados (tabla 7b).

2 .5. Distancia

El primero de los conceptos importantes para introducir los elementos significativos que permitan asegurar privacidad, es el de Distancia. Este se define como la cantidad de entidades en que difieren dos bases de datos D, D'. Esto quiere decir, por ejemplo si se tuviera una Base de Datos (G_1, l_1) con una copia de esta (G_2, l_2), las distancias de ambas sería 0, debido a que no hay diferencias entre ambas. Si a (G_2, l_2) se le agrega una entidad (por ejemplo, tanto un nodo o una etiqueta), entonces la distancia sería 1, debido a que es un valor lo que diferencia a (G_1, l_1) de (G_2, l_2). Si a este se le agregan k entidades, entonces la distancia entre ambas base de datos sería k.

Debido a que se utiliza base de datos de Grafo, los elementos que definen la distancia son arcos, nodos y etiquetas de estos, a diferencia de un modelo relacional en el cual es determinado por las filas de dos tablas. Dependiendo de la implementación, la medida de distancia puede definirse con la presencia de uno o algunos de los elementos anteriores.

2 .6. Sensibilidad

Se ha mencionado anteriormente que se puede obtener base de datos vecinas a partir de la entidad elegida para el análisis (nodo, arco, etiqueta). Dependiendo de esta, el añadir o quitar alguna de estas entidades puede provocar cambios en el resultado de una consulta, que pueden ir de pequeños, a otros de mayor grado (por ejemplo, dentro de una consulta *COUNT*). Esto es lo que se define como Sensibilidad (véase figura 8).

Por ejemplo, se quiere encontrar un valor para una operación *COUNT* en alguna consulta, de la cual podría realizarse varias veces para encontrar un valor en particular, entregando el mismo resultado. Entonces, si dentro del *dataset* se añade un valor, la misma consulta anterior puede cambiar el valor de su resultado en a lo menos 1 unidad (debido a que se agrega o quita la información de una persona) [Dwork, 2011].

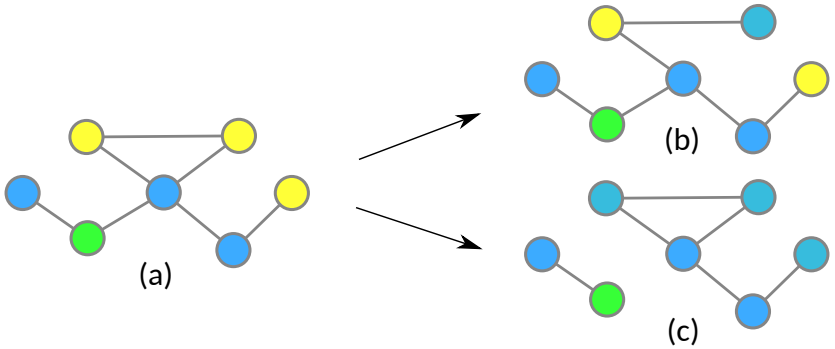


Figura 8: Ejemplo del concepto de sensibilidad con dos bases de datos vecinas (b y c). La consulta realizada es del estilo: ¿Cuantos amigos de amigos tiene el nodo verde?. Se observa que el resultado de (a) es 3, mientras que (b) tiene 2 y (c) tiene 0. En consecuencia, la diferencia de resultados con (b) es 1, mientras que con (c) en 3. Por ello, el cambio (c) tiene mayor sensibilidad que (b)

Sensibilidad se define con la siguiente fórmula:

$$\Delta f = \max_{D,D'} \|f(D) - f(D')\|_1 \tag{8}$$

Donde:

- Δf: Sensibilidad. Esta señala, en cuanto variaría la información de la base de datos.
- $f()$: Función de aplicación de ruido.
- D : *Dataset* real
- D' : *Dataset* vecino del real

Dentro del estudio, para los diferentes *dataset* generados a través de consultas a la base de datos $f(D)$, existen dos formas de definir la Sensibilidad para estos, los que se explican a continuación.

2 .6.1. Sensibilidad global

Para consultas sobre el *dataset* $f : D^n \to \mathbb{R}^d$, en los cuales *subdatasets* $x, y \in D^n$ son vecinos ($x \sim y$), cumpliendo tener distancia 1 ($d(x, y) = 1$). Entonces para todas las combinaciones de *datasets* posibles, Sensibilidad global se define como:

$$GS_f = \max_{x,y:d(x,y)=1} ||f(x) - f(y)|| \tag{9}$$

Para ejemplificar su uso, suponer que se tiene un *dataset* con n datos. Imaginar que para una consulta *COUNT*, en un principio se puede obtener el valor 0, pero debido al cambio de un cierto dato (la distancia es 1), este conecta con todos los datos del *dataset* (cantidad n), por lo que el resultado de la consulta es n. En este caso, la Sensibilidad global tendría el valor de n, siendo el mayor valor posible de una perturbación dentro el universo posible de *datasets*.

2 .6.2. Sensibilidad local

Para consultas $f : D^n \to \mathbb{R}^n$, el *dataset* real $x \in D^n$, y vecinos de este *dataset* $y \in D^n$ ($x \sim y$), la Sensibilidad local de la consulta aplicada es:

$$LS_f = \underset{y:d(x,y)=1}{\text{máx}} ||f(x) - f(y)|| \tag{10}$$

A diferencia de la Sensibilidad global, en Local solamente se utiliza el verdadero *dataset* x basado en la consulta realizada, el cual es un subconjunto derivado del *dataset* global. Lo variable de ello, serían los vecinos y, los cuales corresponden a todas las posibilidades de vecino que puede tener el *dataset* anterior.

Por ello, este concepto es siempre menor o igual a la Sensibilidad global ($LS_f \leq GS_f$), ya que el conjunto de *datasets* analizados es acotado.

2 .6.3. Sensibilidad para transformadores *JOIN*

Continuando con el concepto de Sensibilidad, existe un caso especial para los transformadores *JOIN*, debido a que estos afectan en gran forma la sensibilidad de una consulta $f(D)$ debido a la unión de conjuntos.

Considerar el siguiente ejemplo: se tienen tres conjuntos de datos X, Y, Z, en los cuales X comparte atributos con Y, y este comparte con Z, por lo cual se puede realizar las operaciones *JOIN* $X \bowtie_= Y$ y $Y \bowtie_= Z$. Por lo tanto, el resultado de las operaciones anteriores se expresaría como $(X \bowtie_= (Y \bowtie_= Z))$, generando un nuevo resultado de datos, al cual se busca realizar una consulta tipo *COUNT*.

Para obtener la sensibilidad total, se tiene que obtener el máximo cambio del resultado de la consulta *COUNT* realizada al conjunto. Como se ha visto anteriormente (figura 8), el cambio dentro del resultado tiene relación con la conexión entre atributos de los conjuntos X, Y, Z.

En el caso de conjuntos trasformados a través del operador *JOIN*, la dependencia entre atributos aumenta de forma considerable, ya que estos son los que unen los resultados de otros conjuntos. En el ejemplo, un atributo de la relación $Y \bowtie_= Z$, afecta directamente un resul-

tado de la unión $X \bowtie_= Y$, aumentando la magnitud de la sensibilidad de la consulta (figura 9).

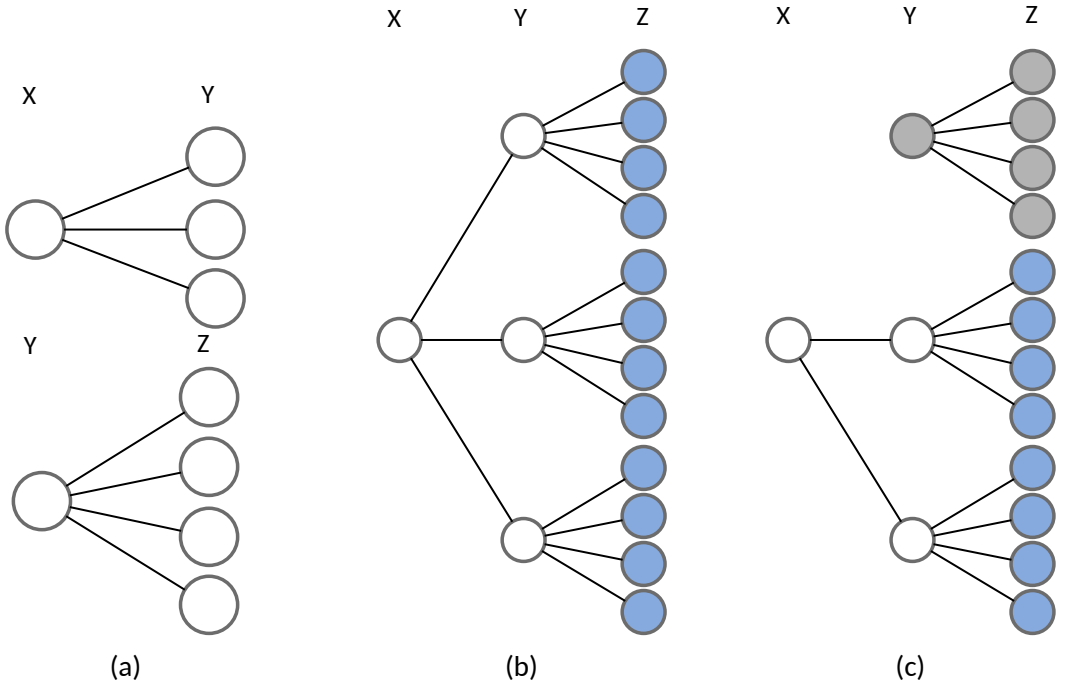


Figura 9: Ejemplo gráfico de un cambio de sensibilidad en *JOIN*. Se tienen dos consultas: $X \bowtie_= Y$ y $Y \bowtie_= Z$ (figura 9a). Posteriormente, la consulta realizada de la combinación de ambos arrojaría el resultado de 12 (figura 9b. Finalmente, se crea un *dataset* vecino removiendo un arco, y el resultado equivale a 8 (figura 9c), por lo que la sensibilidad es 4.

De la misma forma anterior, si el transformador *JOIN* se aplicara a una mayor cadena de uniones (por ejemplo, $(X \bowtie_= (Y \bowtie_= (Z \bowtie_= ...(\bowtie_= N))))$), la sensibilidad podría aumentar de forma muy considerable.

Esto depende en gran parte de la máxima frecuencia que puedan obtener los operadores *JOIN*. Por ejemplo, si se realiza la misma unión de los conjuntos X, Y, Z de la figura 9, donde el resultado es 4, si en la relación Y, Z hubiese un atributo en Y con 6 arcos a Z, entonces su sensibilidad sería este valor.

2 .7. Aplicaciones de Privacidad diferencial

2 .7.1. Privacidad diferencial a través de Sensibilidad global

Dentro de las aplicaciones de Privacidad diferencial, la más simple es utilizando la sensibilidad global del *dataset*. Se dice que se puede entregar resultados seguros de consultas

$f : \mathbb{D} \to \mathbb{R}$ realizando una perturbación simple en la respuesta a través de la distribución de Laplace [Dwork *et al.*, 2006]. Esto a través de la siguiente expresión:

$$\mathcal{A}(D) = f(D) + \text{Lap}\left(\frac{GS_f}{\epsilon}\right) \tag{11}$$

Esta entrega un grado de privacidad $(\epsilon, 0)$ de la consulta $f(D)$.

Aun así, debido a que se utiliza el concepto de sensibilidad global el cual considera todo el volumen de un *dataset*, si este presenta una gran cantidad de datos relacionados, entonces los resultados que se entregan para consultas a estas relaciones pueden ser muy ruidosos a lo esperado, por lo que para algunos casos puede no ser ideal utilizar esta forma.

Otro caso a considerar es cuando se procesa la información para poder responder consultas, por ejemplo para operaciones *JOIN*. Para esto se aplica el concepto de transformador de *dataset* $T : \mathbb{D} \to \mathbb{D}$ (sección 2 .4), el cual es globalmente α estable si $d(T(D), T(D')) \leq \alpha d(D, D')$ para cada $D, D' \in \mathbb{D}$ posible. Si estos transformadores se encuentran con una sensibilidad global acotada, entonces producirían sensibilidad globales $GS_{f \circ T} \leq \alpha GS_f$ siempre que t sea globalmente α estable.

Pero no siempre puede ocurrir dicho caso. Hay veces donde al realizar transformaciones para poder obtener resultados, al crear un *dataset* vecino, las cotas de sensibilidad globales pueden ser sobrepasadas, aumentando considerablemente la sensibilidad. Lo cual se debe a la conexión afectada entre tripletas, donde la manipulación de alguna de estas lleva como consecuencia también la modificación de todas aquellas relacionadas (sección 2 .6.3). Esto produce una alta cantidad de sensibilidad, lo cual tiene como consecuencia valores muy altos en la sensibilidad global, entregando resultados que no son útiles como respuesta.

2 .7.2. Privacidad diferencial a través de Sensibilidad local

Continuando con el problema anterior, aun así existen otro tipo de implementaciones que pueden solucionar el problema de la sensibilidad de uniones. Este tema ha sido estudiado para implementaciones de uniones de tablas [Johnson *et al.*, 2018b], y dentro de ello se utiliza el concepto de sensibilidad local.

El utilizar este concepto permite reducir el grado de error, debido que a diferencia de la sensibilidad global, se utiliza solamente la sensibilidad del *dataset* generado a través de la estructura de la consulta. Por otro lado, si se utilizara directamente este concepto, la respuesta puede no cumplir con protección de privacidad, por el motivo que el ruido agregado puede revelar información del conjunto [Johnson *et al.*, 2018b].

Se utiliza una aproximación de la sensibilidad local llamado cota superior β *smooth* [Nissim *et al.*, 2007], la cual consiste en una función $\mathcal{U} : \mathbb{D} \to \mathbb{R}_{\geq 0}$, la cual acota a la sensibi-

lidad local $LS_f : \mathbb{D} \to \mathbb{R}$ de la consulta $f : \mathbb{D} \to \mathbb{R}$ si cumple con las siguientes condiciones:

- $\mathcal{U}(\mathcal{D}) \geq LS_f(D)$ para todo *dataset* D
- $\mathcal{U}(\mathcal{D}) \leq e^{\beta}\mathcal{U}(D')$ para todos los *dataset* vecinos D y D'.

Posterior a ello, se puede realizar aproximaciones de la sensibilidad local utilizando los conceptos anteriores, las cuales como cumplen con estas condiciones, entonces sirven para ser utilizadas como Sensibilidad local. En contexto, se tiene una consulta numérica $f : \mathbb{D} \to \mathbb{R}$, y asumir que el valor de $\mathcal{U}^{(k)}$ es una cota superior para la sensibilidad local $LS_f^{(k)}$ para la consulta $f(D)$ a una distancia entre *datasets* de valor k:

$$\mathcal{U}^{(k)}(D) = LS_f^{(k)}(D) \forall D \in \mathbb{D} \tag{12}$$

Entonces se cumple:

$$\mathcal{U}(D) = \max_{0 \leq k \leq \text{tamaño}(D)} e^{-\beta k}\mathcal{U}^{(k)}(D) \tag{13}$$

La ecuación 13 es una β-*Smooth* Cota Superior para la Sensibilidad local $LS_f(D)$ de la consulta f dentro de D.

Entendiendo los conceptos anteriores, se pueden entregar resultados que cumplen con Privacidad diferencial a través de la siguiente expresión:

$$\mathcal{A}(D) = f(D) + \text{Lap}\left(\frac{2\mathcal{U}(D)}{\epsilon}\right) \tag{14}$$

Esta expresión logra entregar Privacidad diferencial debido a que la calibración anteriormente realizada de la cota superior, asegurando la entrega de un resultado que cumple con proteger la privacidad [Nissim *et al.*, 2007].

Lo anterior tiene múltiples ventajas con respecto al uso de Sensibilidad global. En primer lugar, logra responder consultas que no logran tener una cota para la Sensibilidad global (debido a condiciones del modelo de información), pero que sí logran tener una cota para la Sensibilidad local. Por otro lado, no requiere del cálculo directo de la Sensibilidad local de las consultas, al contrario, se utiliza el cálculo de la expresión 13, lo que permite realizar menos procedimientos, y poder ser aplicado en software. Por otro lado, para el caso donde tiene que realizarse uniones de tablas, este método es mucho más eficiente, debido a que sería costoso el calcular la Sensibilidad local para todas las tablas unidas [Johnson *et al.*, 2018b].

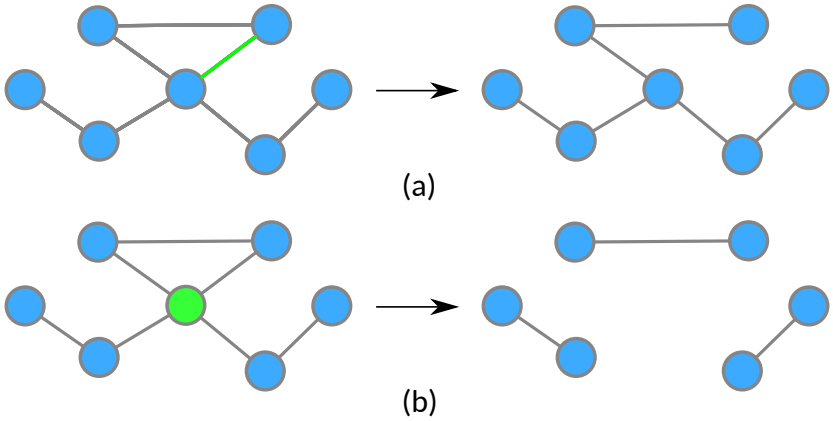


Figura 10: Métricas de distancia para base de datos de grafo a nivel de arco (a) y vértice (b). En ambos casos, la distancia equivale a 1.

2 .8. Métricas de Distancia

El concepto de sensibilidad sirve para medir el cambio del resultado de una consulta sobre una base de datos con respecto a otra base de datos vecina. Aun así, este cambio de resultado puede originarse desde múltiples elementos de la base de datos, tales como una fila, un valor particular de algún individuo, la inserción de un campo, entre otros. Esto quiere decir, que al analizar la Privacidad diferencial de algún *dataset*, tiene que definirse que elemento de esta va a ser el que se modificará en una unidad para poder declarar una base de datos vecina. Esto es lo que se le conoce como métrica de distancia, debido a que es la entidad que genera un distanciamiento con la base de datos real.

Se refiere al elemento en particular de una base de datos que genera esta diferencia. Por ejemplo, en base de datos relacionales, métricas pueden ser el añadido o borrado de filas, el agregado de un campo de información, el borrado de esta, entre otros.

En el caso de bases de datos de grafo, estas contienen la inserción, borrado o editado de tanto arcos como vértices (véase figura 10). Por ejemplo, se puede quitar un vértice como se señala en la figura 11. Esto provocará una distorsión en la base de datos, con los datos que se encuentran indirectamente conectados con este vértice, ya que al realizar consultas, este lado de la base de datos se encontraría vulnerado. Por otro lado, si se quitase otro vértice, entonces una parte de los datos conectados, no se ve tan afectada como en el primer caso.

Es lo que se refiere a una “Métrica de Distancia”, debido a que, dependiendo de la conectividad de algún elemento (como un Nodo o un Arco), esta afectará en la declaración del concepto de distancia para bases de datos vecinas D'.

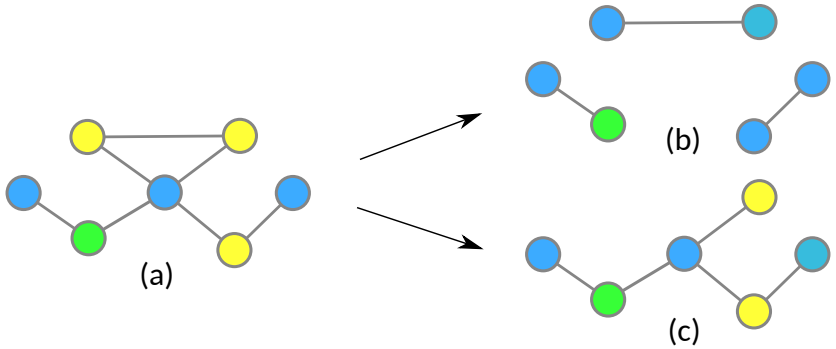


Figura 11: Cambios de sensibilidad de una consulta respecto a métrica de distancia de vértice. Se repite la consulta señalada en figura 8, pero utilizando los vértices como métrica y no los arcos. Se observa que el resultado en (a) es 3. En (b) se remueve un vértice que tiene varias arcos conectados, entregando valor 0. Mientras que en (c), la sensibilidad no es muy afectada, y el resultado cambia a 2.

2 .9. Sensibilidad elástica (*Elastic sensitivity*)

El encontrar la Sensibilidad local para consultas con *JOIN* puede ser una tarea complicada, por lo que se buscan métodos que logren encontrar una aproximación a esta. Es aquí donde entra el concepto de Sensibilidad elástica, el cual entrega una cota superior para la Sensibilidad local de consultas que utilicen *equijoins* (figura 9). El cálculo de esta cota se realiza a través de conceptos algebraicos de conjuntos, los que se señalan a continuación.

Sensibilidad elástica se establece como un concepto matemático, denominado como $\hat{S}^k(q, x)$, el cual para una consulta q hacia la base de datos real x, se calcula para una distancia en particular entre *datasets* k (donde $k = 1$ establece ser vecinos). La función $\hat{S}$ se define desde otro concepto denominado Estabilidad elástica (*Elastic Stability*), utilizado para transformaciones de información relacional ($\hat{S}^k_R(r, x)$).

Un concepto que determina la sensibilidad de consultas *JOIN* (sección 2 .6.3), es la máxima frecuencia de un atributo, determinado a raíz de eventuales cambios dentro de los *dataset* vecinos generados. Se define por la expresión $mf_k(a, r, x)$, la cual se encuentra en términos de un atributo en particular de la información a, en la relación r del *dataset* original, y junto a la instancia x de *dataset* generado.

Se busca obtener la Sensibilidad local a una distancia k para uniones de mapeos de datos. Dentro de uniones de tablas, pueden ocurrir dos casos. El primero, es que ambos mapeos de tablas comparten información entre ellas, mientras que en el segundo, solamente son unidas por el parámetro de unión, siendo estos dos mapeos conjuntos disjuntos.

Ante cambios en el *dataset* en el primer caso, la sensibilidad aumenta en gran cantidad, debido a que dentro de las tablas participantes de la unión, el cambio de la sensibilidad afectaría a las dos tablas. En caso de uniones de conjuntos disjuntos, esto no ocurriría, debido

```
SELECT COUNT(*) FROM arcos a1
JOIN arcos a2 ON a1.dest = a2.source AND a1.source < a2.source
JOIN arcos a3 ON a2.dest = a3.source AND a3.dest = a1.source
AND a2.source < a3.source
```

Figura 12: Código SQL de ejemplo de una consulta compuesta por *equijoins*.

a que afectaría a un lado de la unión.

Por ello, se tiene que definir el concepto de Ancestros de una relación, el cual se define como una condición que cumplen la unión de tablas. Considerar el mapeo de una tabla como un *dataset* $\mathcal{T}(r_n)$, donde r_n corresponde al *dataset* unido. Entonces, al realizar la unión de dos conjuntos r_1, r_2, se tiene que comprobar si se sobreponen o son disjuntos (se cumple la relación $\mathcal{T}(r_1) \cap \mathcal{T}(r_2) = 0$).

Con los conceptos anteriores, se tiene un conocimiento base para poder realizar el cálculo de Sensibilidad elástica de una consulta. Para ejemplificar su uso, considerar la consulta de la figura 12. Esta es una consulta tipo *COUNT* de triángulos. Como se observa, la consulta contiene la unión de *equijoin* entre el origen y destino de arcos a través de la operación "$=$", y por otro lado, se obtiene un mapeo del *dataset* real $\mathcal{T}(r_n)$ utilizando la operación "$\leq$", la cual filtra información para crear un mapeo.

Respecto al procedimiento de Sensibilidad elástica, considerar que se tiene un conjunto de relaciones $\mathcal{R} = r_1, r_2, ..., r_n$, donde cada relación secuencial r_n, r_{n+1} son unidas a través de un atributo formando un *equijoin*. Para cada mapeo de una relación, su Estabilidad elástica tiene el valor de 1 ($\hat{S}^{(k)}_R(r_n, x) = 1$), debido a que no existen parámetros que al eliminarse pueda afectar a otros parámetros pertenecientes a otras tablas.

Se ha mencionado que al realizar *JOIN* de tablas, puede haber cambios en la Estabilidad elástica, los cuales dependen de la intersección de los conjuntos con los cuales se trabaja. Se tiene la siguiente expresión para Sensibilidad elástica:

$$\hat{S}^{(k)}_R(r_1 \bowtie_{a=b} r_2, x) = \begin{cases} \text{máx}(\text{mf}_k(a, r_1, x)\hat{S}^{(k)}_R(r_2, x), \\ \text{mf}_k(b, r_2, x)\hat{S}^{(k)}_R(r_1, x)) & \text{si } |\mathcal{T}(r_1) \cap \mathcal{T}(r_2)| = 0 \\ \text{mf}_k(a, r_1, x)\hat{S}^{(k)}_R(r_2, x) + & \\ \text{mf}_k(b, r_2, x)\hat{S}^{(k)}_R(r_1, x) + & \\ \hat{S}^{(k)}_R(r_1, x)\hat{S}^{(k)}_R(r_2, x) & \text{si } |\mathcal{T}(r_1) \cap \mathcal{T}(r_2)| \geq 0 \end{cases} \tag{15}$$

Con lo anterior, se realiza iteraciones para cada conjunto perteneciente a $\mathcal{R}$. El resultado final de estas operaciones, es la Sensibilidad elástica del conjunto $\mathcal{R}$, el cual logra ser una cota superior para la Sensibilidad local.

2 .10. *RDF* y lenguaje de consultas *SPARQL*

Resource Description Framework (RDF) es una familia de especificaciones definidas por el organismo *World Wide Web Consortium* (W3C), las cuales definen metadatos para contenido web. Está basado en la idea de definir etiquetas para recursos, a través del uso de tripletas de información en forma de sujeto, predicado y objeto. El resultado que produce esto, es la capacidad de conectar elementos web con propiedades entre si.

SPARQL 1.1 es el lenguaje de consultas para realizar sobre *datasets RDF*. Permite tanto el control como uso de esta información almacenada en *RDF*, a través de múltiples operaciones que la mayoría de los tipos de base de datos pueden realizar, tales como agregaciones, sumas, promedios, filtros, entre otros.

A continuación, se introducen elementos de este lenguaje que definen la estructura necesaria para poder aplicar conocimientos sobre Privacidad diferencial. Estos se dividen dentro de dos categorías: características dirigidas por información y estructurales.

2 .10.1. Características Dirigidas por Información

Dentro de *RDF*, la información en forma de tripletas puede representarse a través de un grafo G, el cual cada vértice puede ser representado tanto como el sujeto y el objeto, mientras que los predicados tienen el rol de ser el arco que une vértices, siempre en dirección sujeto a objeto (Figura 13).

Un grafo G se compone de subgrafos $\{g_1, g_2, ..., g_n\}$, representando elementos junto a sus propiedades. Un grafo G_1 puede tener una distancia k de otro grafo G_2 si uno de estos presenta una cantidad de subgrafos de diferencia (corresponden a las métricas de distancia en este caso), por ejemplo, se tiene $G_1 = \{g_1, g_2, g_3\}$ y $G_2 = \{g_1, g_2\}$, entonces la distancia entre ambos *datasets* sería 1.

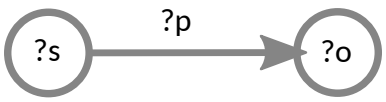
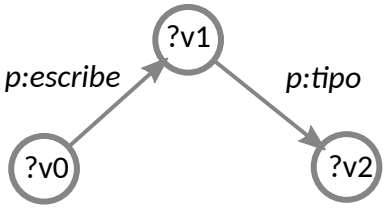


Figura 13: Representación de una tripleta en un diagrama, representando nodos (Sujeto y Objeto), con arcos (Predicado).

Como se ha señalado en la sección anterior, dentro de un modelo de datos las métricas de distancia pueden ser definidas de múltiples formas (Arco, Vértice, o una combinación de estos). En este caso, entre ambos elementos pueden crear subgrafos, o también por solo un arco o vértice, definirse como un subggrafo. En relación a la aplicación de Privacidad diferencial, este método protege a entidades de base de datos relacionales respecto a su presencia



(a)

```
1 SELECT ?v0 ?v2
2 WHERE { ?v0 p:escribe ?v1 .
3         ?v1 p:tipo ?v2 }
```

(b)

Figura 14: Demostración gráfica (figura 14a) y en código (figura 14b) de un Basic Graph Pattern.

en el resultado entregado. Con respecto a *RDF*, se puede definir una entidad a través de un subgrafo, por lo que son estos los que se buscarían proteger.

Por otra parte tiene que diferenciarse cuales subgrafos presentan información pública y privada. Por ejemplo, si se tuviera una tabla con el listado de artistas de un género musical, esta información es de carácter pública, debido a que integralmente no presenta información que puede ser catalogada como confidencial. La información que sería en este caso privada, serían los gustos musicales de los usuarios de la plataforma, por lo que esta información es lo que busca protegerse. Tanto las métricas de distancia seleccionadas como la información que sería de tanto carácter público como privado, tienen que ser seleccionadas por los administradores del *dataset*, debido a que estos tienen el conocimiento de saber que información o conexiones entre esta son las que pueden comprometer información.

***Basic Graph Patterns* (BGP)** Continuando con la definición de tripletas *RDF*, todo el conjunto de información tiene esa forma, pero esto representa una lista gigante de texto simples con tres elementos. Para que esta información logre ser utilizada como información conectada, se debe obtener una definición que logre entregar una estructura.

Para aquello se utiliza el concepto de *Basic Graph Patterns* (*BGP*), los cuales son patrones de información de un *dataset RDF* que identifican un subconjunto de relación entre sujetos y objetos. Estos son extraídos del grafo *RDF*, y luego se retornan como resultado en forma de tripletas, en base a relación entre variables ([Fletcher, 2008])(figura 14). Estos además, son la estructura principal para realizar consultas en *SPARQL*, debido a que definen la estructura de la información consultada.

Consultas Estrella Estas estructuras además, pueden anidarse entre estas, teniendo la capacidad de realizar consultas más complejas con relación de información entre estas, lo cual es el equivalente de agregar mayor cantidad de información dentro de una consulta [Aluç *et al.*, 2014]. Esto, seleccionando tanto un sujeto o un objeto de una tupla $(?X, p, ?Y)$ en común para los *BGP* de la consulta, creando una relación entre esta información a través de un grafo. El vértice central de una consulta se le denomina como Vértice de Unión.

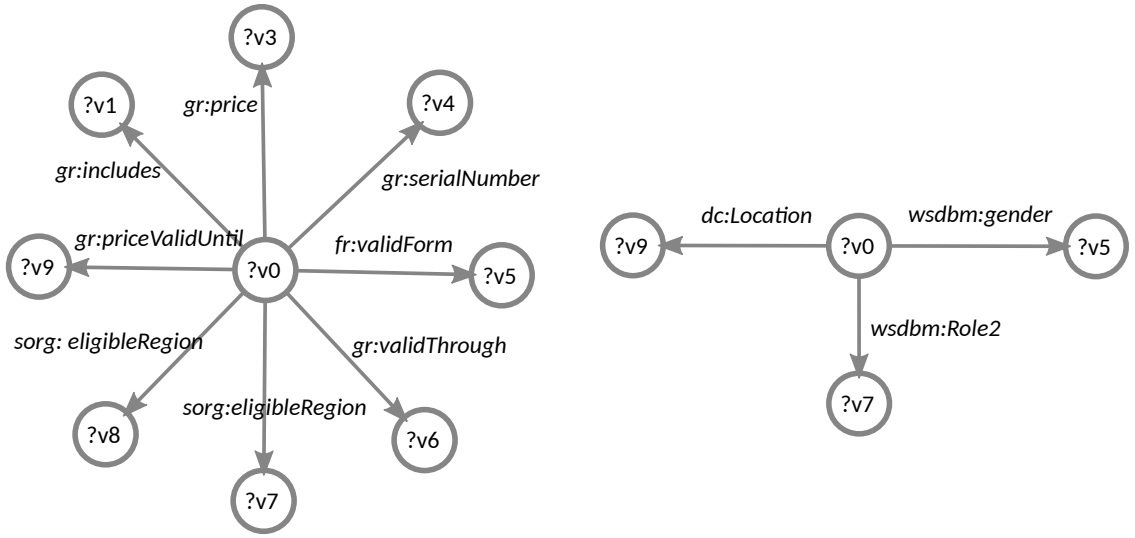


Figura 15: Dos tipos de consultas estrellas. Una de estas tiene 8 objetos (figura 15a), siendo esta cantidad de tripletas lo que define la estrella. Figura 15b es una estrella compuesta de 3 tripletas.

Por otro lado, si el vértice de unión de un grafo corresponde al sujeto de la consulta, entonces esta cadena de *BGP* se denomina como consulta estrella. Este tipo de estructura son la entidad básica al momento de realizar consultas en *RDF*, debido a que a partir de un grafo G, entrega los resultados de las relaciones entre el sujeto con sus objetos, lo que tiene equivalencia con realizar una consulta a una tabla con sus columnas en modelo Relacional. Se dice que una consulta estrella es simple, si tanto el sujeto como el objeto de cada tupla de esta corresponden a variables.

Por otro lado, existen otro tipo de propiedades importantes de mencionar para consultas estrella. Si dos de estas no comparten predicados en común, entonces se señala que estas son disjuntas (su intersección es 0), debido a que el sujeto apunta a predicados que son diferentes (figura 15).

Esquema de Privacidad diferencial Con los conceptos anteriores, se puede definir el esquema para *BGP* denominado Esquema de Privacidad diferencial ($\mathcal{P}$), el cual consiste en un set finito de Estrellas simples disjuntas. Este concepto es importante, debido a que selecciona la cadena de *BGP* que se le aplicaría posteriormente Privacidad diferencial. Los grafos derivados de este esquema de Privacidad diferencial, son aquellos donde los usuarios realizarían sus consultas [Buil, 2019].

Si no se definiera este esquema, entonces el uso de los predicados es ilimitado, lo que puede provocar que la información no sea protegida. Por ello, es necesario definir el Esquema de Privacidad diferencial $\mathcal{P}$ para la mayor cantidad de estrellas posibles.

Otra característica importante dentro de la definición de grafos *RDF*, es el concepto de Sensibilidad (sección 2 .6). Se tiene por ejemplo un empleado participante en el *dataset*, podría tener información tales como dirección, números de teléfonos, posición laboral, entre otros. Se asume que una persona puede tener más de un número de teléfono, pero dentro del *dataset*, la máxima cantidad de números por persona puede restringirse a un valor, por ejemplo 3. Este límite de información es lo que se define como Sensibilidad del grafo, debido a que define la máxima cantidad de objetos que un sujeto puede tener de cierto predicado y viceversa. Para una estrella simple S, se denomina a la Sensibilidad $sen(S)$ como la suma de las sensibilidades de los patrones triples dentro de S.

Con lo anterior, como punto de partida se puede tener un *dataset* como un grafo G, el cual se compila en esquema de Privacidad diferencial $\mathcal{P}$, considerando la Sensibilidad de los triples. Desde este $\mathcal{P}$, se puede extraer un set de subgrafos $\{g_1, ..., g_n\}$. Un grafo g pertenece al subconjunto si es una estrella S con centro en el sujeto $?X$, con soluciones $\mu \in [\![S]\!]_G$. Notar para cada par de subgrafos, la intersección es vacía.

2 .10.2. Características estructurales de consultas *SPARQL*

Otra característica que debe considerarse son los filtros aplicados a los *BGP* mencionados. Estos se realizan a través de expresiones de filtro para *BGP* (*CBGP*), el cual es un *BGP* con la notación $\overline{B} = \langle B, F \rangle$, donde B corresponde al *BGP*, mientras que $F = f_1, ..., f_n$ es un conjunto de filtros finitos.

En la práctica, el concepto de $\overline{B}$ se representa como $P = ((...(B \text{ FILTER } f_1)...) \text{ FILTER } f_n)$, el cual como resultado genera un mapeo del *dataset* determinado por la expresión $[\![\overline{B}]\!]_G$.

Las expresiones de filtro F no tienen disjunciones, conjunciones o igualdades entre dos variables, debido a que si estas existieran, entonces serían renombradas para no generar conflictos. La negación de expresiones es válida.

Se asume también que las variables compartidas en triples dentro de $\overline{B}$ se pueden definir dentro de $[\![\overline{B}]\!]_G$ utilizando solamente operaciones *JOIN*. Finalmente, se asume para un grafo G derivado desde un Esquema de Privacidad diferencial $\mathcal{P}$, todos los predicados de los triples dentro de un *CBGP* aparecen también dentro de $\mathcal{P}$.

Debido a que el análisis se realiza en consultas tipo *COUNT*, una consulta realizada por el usuario será un *CBGP* $\overline{B}$ si estuviera encapsulada dentro de las siguientes operaciones:

- COUNT$(\overline{B})$ donde la semántica de los resultados $[\![\text{COUNT}(\overline{B})]\!]_G$, esta definida por la cardinalidad del multiset $[\![\overline{B}]\!]_G$
- COUNT$_{?X_1,...,?X_n}(\overline{B})$ donde $?X_1, ..., ?X_n$ son variables que aparecen dentro de $\overline{B}$ y donde sus semánticas $[\![\text{COUNT}_{?X_1,...,?X_n}]\!]_G$ se definen como un conjunto de mapeo de soluciones desde $[\![\overline{B}]\!]_G$, de acuerdo a las variables $?X_1, ..., ?X_n$.

Definir la expresión $var(E)$ como el conjunto de variables que aparecen dentro de una expresión E, la cual puede ser un *CBGP*, *BGP* o Esquema de Privacidad diferencial. Para una estrella simple S, una tripleta perteneciente a esta se denomina $t_p = (s, p, o)$. El sujeto de algún triple, el cual deriva hacia el resto de los objetos del *BGP* se denomina como centro de la estrella ($centro_S(t_p) = s$). En caso contrario, el predicado define el centro ($centro_S(t_p) = o$).

Para cada consulta $\overline{B} = \langle B, F \rangle$ con un Esquema de Privacidad diferencial $\mathcal{P}$, se puede definir un partición de estos con la notación $B_{\div\mathcal{P}} = B_1, ..., B_n$. Este conjunto cumple que cada $B_i \in B_{\div\mathcal{P}}$ es un *subset* máximo de B, para el cual existe una estrella simple $S \in \mathcal{P}$ tal que $pred(B_i) \subseteq pred(S)$. Para los triples del conjunto $t_p, t'_p \in B_i$, el centro se define $centro_s(t_p) = centro(t'_p)$. Esta estrella se denomina como Estrella Cobertora de B_i.

El uso de la expresión anterior $B_{\div\mathcal{P}}$ es de suma importancia para poder aislar las consultas B_i. Esto entrega la base para las siguientes definiciones:

Se determina que S_i es la estrella cobertora de B_i, y G_{S_i} es el grafo inducido por S_i, entonces:

$$[\![\overline{B_i}]\!]_G = [\![\overline{B_i}]\!]_{G_{S_i}} \tag{16}$$

De la expresión anterior, al realizar las operaciones de *JOIN* para cada mapeo de soluciones del *multiset* generado [Pérez *et al.*, 2009], se puede extender la expresión anterior a lo siguiente:

$$[\![B]\!]_G = [\![B_1]\!]_{G_{S_1}} \bowtie [\![B_2]\!]_{G_{S_2}} \bowtie ... \bowtie [\![B_n]\!]_{G_{S_n}} \tag{17}$$

Lo anterior permite definir $[\![B]\!]_G$ como un conjunto de *equijoins*, los cuales tienen un orden en particular. Este orden implica que las operaciones *JOIN* son iteradas de forma $[\![B_i]\!]_{G_{S_i}} \bowtie [\![B_{i+1}]\!]_{G_{S_{i+1}}}$, los conjuntos de variables cumplen $var(B_i) \cap var(B_{i+1}) \neq \emptyset$. Lo anterior es lo que se denomina orden normal de partición $B_{\div\mathcal{P}}$.

2 .11. Privacidad diferencial para *RDF/SPARQL*

En las secciones anteriores, se ha descrito un álgebra para conjuntos de subgrafos *SPARQL*, el cual consiste en realizar operaciones matemáticas entre los mapeos generados por estos. En esta sección, se utilizarían estos conceptos para poder fundamentar el uso de Privacidad diferencial en los subgrafos [Buil, 2019].

Se asume que se utilizará un grafo que cumple con Esquema de Privacidad diferencial $\mathcal{P}$, y que todo subconjunto cumple con este. Para una consulta Q realizada sobre un grafo *RDF* G, se tiene una cota superior $\mathcal{U}_Q^{(k)}$ de la sensibilidad local de la consulta Q ($LS_Q^{(k)}$). Para poder

```
SELECT (COUNT(?v0) as ?v0_count) WHERE {?v0 ?p ?o} LIMIT 1
ORDER BY ?v0
```

Figura 16: Código en *SPARQL* para poder obtener la máxima frecuencia de un atributo de un *BGP*.

aplicar Sensibilidad elástica (véase sección 2 .9), se tiene que obtener una Cota Superior β-*smooth* $\mathcal{U}_Q(G)$ de la sensibilidad local $LS_Q(G)$ de la consulta Q hacia el grafo G.

Sensibilidad elástica para *RDF* Con esto, se puede adaptar el concepto de Sensibilidad elástica a las uniones de grafos *BGP* B. El concepto de Sensibilidad elástica para un grafo derivado de una consulta de usuario Q a una distancia k del verdadero grafo *RDF* G se denota por la expresión $S^k(Q, G)$, la cual será definida posteriormente para el uso de *CBGP*. Esta expresión de Sensibilidad, es una cota superior para la consulta. Por ejemplo, si se realizara una consulta de conteo COUNT(B), entonces esta consulta tendría como cota superior $S^{(k)}_{RDF}(B, G)$, la cual es determinada a una distancia k.

Para poder definir la estabilidad de la consulta, se comienza por calcular la máxima frecuencia de las variables que realizan *JOIN* dentro de un *BGP*. Por ejemplo, si el sujeto es la variable central que realiza *JOIN*, entonces se debe realizar una consulta de máxima frecuencia utilizando este sujeto como variable (figura 16).

Esta consulta de frecuencia obtiene un mapeo del *BGP*, donde se cuenta el número de ocurrencias que tiene la variable respecto a los resultados, retornando el número de máximas repeticiones del atributo que obtenga mayor cantidad de ocurrencias.

Este valor se denota por la expresión $mp(?X, \overline{B}, G)$. Se requiere también conocer el valor del resultado más repetido para diversas distancias k del grafo, el cual se denota por $mp_k(?X, \overline{B}, G)$. Este valor se define de forma inductiva dependiendo de la cardinalidad de $B_{\div\mathcal{P}} = B_1, ..., B_n$:

- Para $|B_{\div\mathcal{P}}| = 1$:

$$mp(?X, \overline{B}, G) + k * \text{Sens}(S) \tag{18}$$

  donde Sens(S) es la Sensibilidad de la estrella cobertora de B_1.

- Para $|B_{\div\mathcal{P}}| = n + 1$:
  Hacer que $B_1 \in B_{\div\mathcal{P}}$ y $V = \text{var}(B_{\div\mathcal{P}}/\{B_1\} \cup \text{var}(B_1)$, siendo $?v$ y $?u$ las variables del conjunto V, entonces:

$$mp_k(?v, \overline{B}, G) = \text{máx} \left\{ \begin{array}{l} \arg\text{máx}\, mp_k(?v, B_{\div\mathcal{P}}/\{B_1\}, G) * mp_k(?u, B_1, G) \\ \arg\text{máx}\, mp_k(?u, B_{\div\mathcal{P}}/\{B_1\}, G) * mp_k(?v, B_1, G) \end{array} \right\} \tag{19}$$

- Para variables que no se encuentran dentro de $\overline{B}$:

$$mp_k(?v, \overline{B}, G) = mp_k(?u, \overline{B}, G) = 0 \tag{20}$$

Por otro lado, también se puede definir la Sensibilidad elástica para los *CBGP* $\overline{B}$ por inducción de la cardinalidad de $B_{\div} = B_1, ..., B_n$, teniendo conocimiento que B_i se encuentra indexado en orden normal:

- Para $|B_{\div\mathcal{P}}| = 1$:

$$S^{(k)}_{\mathsf{RDF}}(B, G) = \mathsf{sen}(S) \tag{21}$$

  donde S es la estrella cobertora de B_1

- Para $|B_{\div\mathcal{P}}| = n + 1$: Hacer $B' = B_{\div\mathcal{P}}/\{B_1\}$. Se tienen dos casos:

  - Si la estrella cobertora S de B_1 no es la estrella cobertora para otros $B_i \in B'$:

$$S^{(k)}_{\mathsf{RDF}}(B, G) = \mathsf{máx} \left\{ \begin{array}{l} mp_k(\mathsf{var}(B_1) \cap \mathsf{var}(B'), B_1, G) \times S^{(k)}_{\mathsf{RDF}}(B', G) \\ mp_k(\mathsf{var}(B_1) \cap \mathsf{var}(B'), B', G) \times S^{(k)}_{\mathsf{RDF}}(B_1, G) \end{array} \right\} \tag{22}$$

  - Si la estrella cobertora S de B_1 es también la estrella cobertora para otros $B_i \in B'$:

$$\begin{aligned} S^{(k)}_{\mathsf{RDF}}(B, G) = {} & mp_k(\mathsf{var}(B_1) \cap \mathsf{var}(B'), B_1, G) \times S^{(k)}_{\mathsf{RDF}}(B', G) + \\ & mp_k(\mathsf{var}(B_1) \cap \mathsf{var}(B'), B', G) \times S^{(k)}_{\mathsf{RDF}}(B_1, G) + \\ & S^{(k)}_{\mathsf{RDF}}(B_1, G) \times S^{(k)}_{\mathsf{RDF}}(B', G) \end{aligned} \tag{23}$$

Para cada consulta *CBGP* $\overline{B} = \langle B, F \rangle$ y grafos G que cumplen con $\mathcal{P}$:

$$S^{(k)}_{RDF}(B, G) \geq \mathsf{LS}^{(k)}_{COUNT(B)}(G) \tag{24}$$

La expresión anterior es paralela a lo descrito en la definición de Sensibilidad elástica [Johnson *et al.*, 2018b] para acotamiento de resultados. La principal observación de la expresión anterior, si existiera algún cambio dentro de los subgrafos g_i dentro de G, y este subgrafo perteneciera a la estrella relacionada S de G, entonces el número de tuplas dentro del grafo cambiaría en el peor caso en $sen(S)$. Si por otro lado existen variables compartidas entre dos subgrafos, por ejemplo $B = \{(?v0, p, ?u), (?u, p', ?v1)\}$, y la tripleta (s, p, o) perteneciera a g_1, entonces este objeto sería el resultado más popular del subgrafo. Por lo tanto, a lo más habrían $mp(?u, (?u, p', ?v1), G)$ nuevos resultados. Por otro lado, si el resultado $(?u, p', ?v1)$ perteneciera a S, entonces habrían mucho más nuevos resultados, debido a que este se duplicaría.

Con lo anterior, para cada consulta de un usuario Q realizada a un grafo G que cumple con Esquema de Privacidad diferencial $\mathcal{P}$, se describe el teorema:

$$S^{(k)}(Q, G) \geq \mathsf{LS}_Q^{(k)}(G) \tag{25}$$

Lo cual define para los grafos G una cota para la Sensibilidad local. Lo anterior, permite ser una cota que cumple con (ϵ, δ) Privacidad diferencial para la aplicación de algoritmo $\mathcal{A}$.

2 .12. Publicación de Información

El objetivo principal de la Privacidad diferencial es el proteger la información de los participantes. Una de las vulnerabilidades de privacidad más grandes, ocurre cuando la información es publicada como resultado o como un *dataset* analítico.

El problema se puede representar por lo siguiente: un administrador tiene su dataset D, y recibe un conjunto de consultas $F = f_1, f_2, ..., f_n$. Se necesita responder a este conjunto cumpliendo con Privacidad diferencial.

Existen dos conjuntos de métodos para responder: interactivos y no interactivos. En los interactivos (sección 2 .12.2), una consulta f_i no puede responderse sin anteriormente haber respondido a la consulta f_{i-1}. Mientras que los no interactivos (sección 2 .12.3), todo el conjunto de consultas a responder es entregado al administrador en una unidad de tiempo, respondiendo el conjunto de consultas en si.

Con relación al presupuesto de privacidad ϵ, se presentan dos teoremas de composición del resultado de una consulta:

- Composición Paralela (*Parallel Composition*): Si cada M_i entrega ϵ -Privacidad diferencial. Entonces, M entrega $max(\epsilon_1, \epsilon_2, ..., \epsilon_n)$ Privacidad diferencial.
- Composición Secuencial (*Sequential Composition*): Se aplican m_i secuenciales, y cada uno entrega ϵ-Privacidad diferencial (se aplica $m_1, m_2, .., m_n$ en el *dataset*). Por lo tanto, el conjunto M de operaciones, entrega $(m * \epsilon)$-Privacidad diferencial.

2 .12.1. Mecanismos de Publicación de Información

Existen diversos mecanismos y estrategias para procesar los *dataset* a los cuales se les aplicará privacidad, y así pode ser entregados a público.

Para entender el contenido de la siguiente sección, tener en consideración el siguiente ejemplo: Se tienen registros médicos de pacientes que padecen diabetes, donde la información

Edad	Pacientes con diabetes	variable
60, 79	41	x_1
40, 59	32	x_2
20, 39	8	x_3
0, 19	1	x_4

Tabla 2: Ejemplo de información real de número de pacientes con diabetes.

es el paciente, y un valor verdadero o falso que indica la condición de la persona. Esto es lo que se busca proteger. Un usuario para una investigación quisiera realizar las siguientes dos consultas:

- $f1$: Cuantos pacientes tienen diabetes entre la edad de 40 a 79?
- $f2$: Cuantos pacientes tienen diabetes entre la edad de 40 a 59?

La información necesaria para responder estas consultas se encuentra expresada en la tabla 2.

Transformación : Realiza una transformación del *dataset* original aplicando privacidad a los resultados a través de añadido de ruido, respondiendo a consultas de forma segura. Estas transformaciones se pueden utilizar por ejemplo para crear una tabla de frecuencias. Utilizando el ejemplo, se tiene:

- f_1 : f_2 1era fila con ruido ($\hat{f}_1$), sensibilidad = 2
- f_2 : Se responde directo
- Perturbación total : $3 * Lap(1/\epsilon)$

Particionamiento de *Dataset* (*Dataset Partitioning*) : Se realiza una partición desde el *dataset*, dividiendo su información en múltiples partes para reducir el ruido total aplicado, como consecuencia de utilizar menor cantidad de información por partición. El ruido aplicado es relativo a las características de la partición, teniendo la posibilidad de ser menor que en el caso de aplicar a todo el conjunto.

- f_1 : Utilizando tabla 2, se tiene $2 * Lap(1/\epsilon)$, pero si hubiera entre $40, 79$ precalculado (ejemplo, consulta de conteo), entonces el ruido sería $Lap(1/\epsilon)$
- f_2 : Se responde directamente

Como desafío de este método, se encuentra la implementación de estrategias para responder consultas, dependiendo del modelo de datos.

Separación de Consultas (*Query Separation*) : Resultados de consultas anteriores son reutilizados para nuevas consultas. Este método permite limitar el número de consultas realizadas, junto a reutilizar estos resultados para mantener el ruido aplicado.

Iteración : Se utiliza un *dataset* alternativo D_n derivado del original D_0, el cual tiene diferencias respecto al original junto a su tabla de frecuencias precalculadas con las consultas respondidas. Esto a través de lo siguiente:

- Se define el *dataset* inicial D_0, que equivale a la tabla 2.
- Se hace una consulta f_1 en D_0
- Se compara el resultado de $\hat{f}_1(D)$ con $f_1(D)$
- Si la distancia entre ambos es menor a una definida como límite, entonces:
  - Verdadero: $f_1(D_0)$ es publicado, D_0 se utiliza en próxima iteración
  - Falso: $\hat{f}_1(D_0)$ es publicado, y D_0 se actualiza a D_1 con alguna estrategia

Lo anterior permite poder responder una mayor cantidad de consultas a diferencia de aplicar directamente Laplace. Como desafíos, se encuentra el diseño de estrategias para actualizar los *datasets* D_n, junto a la selección de parámetros, tales como la distancia en las iteraciones.

2 .12.2. Métodos Interactivos

Publicación Transaccional (*Transaction Data Publishing*) : Todo registro representa un individuo con una cantidad de d valores. Se aplican los siguientes métodos:

- *Query Separation* : Método de partición de consultas, con el objetivo de responderlas de forma más eficiente utilizando resultados anteriores. Estas se dividen en dos tipos: consultas difíciles, las cuales se aplica Laplace, y por otro lado consultas blandas, en las cuales se utiliza la mediana de las consultas difíciles respondidas. Este método tiene complejidad $O(\log m \log |x|)$.
- Iterativos : Se utiliza método de Pesos Multiplicativos Privados (*Private multiplicative weights*), donde el *dataset* se reparte en bloques con pesos (histogramas). Esto entrega respuestas de consultas con un error de cota $O(\log m/\sqrt{n})$. Esto significa que el error de muestra (*sampling error*) crece en forma logarítmica, mientras que perturbación de Laplace crece linealmente. Por lo tanto, se puede decir que el método responde a las consultas con precisión.

Este método interactivo, puede responder más consultas que Laplace directamente. La cuota de error aumenta en $\log m$, mientras que Laplace en $m \log m$.

Publicación de Histogramas (*Histogram Publishing*) : El objetivo de este método, es dividir el *dataset* en un número N de contenedores, en los cuales el objetivo de aplicar Privacidad diferencial es ocultar la frecuencia real de estos contenedores. La modificación de información en alguno de los contenedores, significaría un cambio de sensibilidad solamente en aquel contenedor, por lo que la magnitud de ruido aplicado sería pequeña en este caso.

Publicación de particionamiento de *dataset* : Dependiendo de la consulta realizada, se puede unir o separar contenedores de datos a través de estrategias de partición. Se utiliza como medida para reducción el Error Cuadrático (*Squared Error*) [Xu *et al.*, 2013].

Por otro lado, se puede relacionar valores de una forma jerárquica [Qardaji *et al.*, 2013], utilizando árboles con el rango completo de valores como raíz, los nodos como unión de los rangos y las hojas para el rango de unidades de longitud.

Se tiene que tener precaución con la consistencia del Histograma, debido a que el método de particionamiento de *dataset* puede traer inconsistencias, por lo cual se tiene que aplicar restricciones. Existen dos tipos de estas, *Sotred Constraints* (restricciones ordenadas), la cual requiere los resultados de las consultas, y *Hierarchical constrains* (restricciones de jerarquía), que define la secuencia de la Jerarquía.

Publicación de flujo de Información (*Stream Data Publishing*) : La información es publicada por cada fracción de tiempo. Se utilizará la notación binaria, donde el 0 señala que un evento no ocurre, y 1 que si ocurre. Se utiliza las siguientes notaciones:

- $D\{0,1\}^t$: Cuenta de t eventos
- $T = \{t_k : k = 0, .., n\}$: Tiempos de ocurrencia
- $f(t_k)$: Cuenta de bits
- $\hat{f}(t_k)$: Cuenta de bits, con ruido añadido

Particionamiento de datos : La información es particionada en subdominios. Se utiliza el método de sumas parciales (*psums*) [Chan *et al.*, 2011], el cual estima el resultado de consultas *COUNT* en intervalos de tiempo a través de sumas. Tiene un error de $O(\log^{3/2} |T|/\epsilon)$. Otro método es el árbol de cuadruples (*quadtree*) [Zhang *et al.*, 2016] para *datasets* espaciales.

Publicación de Información en Grafo (*Graph Data Publishing*) : Publica información proveniente de una base de datos en forma de Grafo, el cual se define como un conjunto $G = (V, E)$. Se divide en dos grandes grupos. Primero se tiene Privacidad diferencial de Arcos (*Edge differential privacy*) , donde la consulta realizada no revelaría información sobre el añadido o borrado de un arco en el *dataset*.

Por otro lado, se tiene el *Node Differential Privacy* (Privacidad diferencial de nodo o vértice), donde el elemento que define la vecindad entre dos *datasets* es un nodo (o vértice) junto al conjunto de todos los arcos de este. Una forma de asegurar privacidad, es limitar el grado de los vértices, creando así un nuevo *dataset* filtrando sus arcos con un grado mayor al permitido.

2 .12.3. Métodos no Interactivos

Consultas por Lotes (*Batch Query Publishing*) : Este método consiste principalmente en descomponer la correlación entre las consultas para que la sensibilidad total pueda tener una disminución. Se intenta responder un conjunto de consultas $F = f_1, ..., f_n$. El borrar algún dato del *dataset*, puede cambiar la información de la respuesta dependiendo de la correlación entre consultas. Existen los siguientes métodos para lograr la disminución de sensibilidad:

- **Transformación**: Una forma de bajar esta sensibilidad es re-calibrar esta hacia una nueva estructura [Xiao *et al.*, 2011b]. Se propone el uso de una transformación hacia una Ondícula de la frecuencia del *dataset* D para generar otra con coeficiente A. Estos se consideran combinaciones lineales de las entradas del *dataset*, de las cuales se realizan los cálculos.
- **Particionamiento de *Dataset***: Otra forma es descomponer los *datasets* afectados por las consultas en grupos disjuntos [Kellaris y Papadopoulos, 2013], a los cuales se les aplica ruido de Laplace. El resultado se responde con el valor calculado de la columna de los datos. Por otro lado, se limita el número de operaciones de conteo originales, haciendo que la sensibilidad de cada grupo disminuya, como así también el ruido de Laplace aplicado.
- **Iteración**: Al responder varias consultas, se produce el problema que el ruido se acumula progresivamente [Xiao *et al.*, 2011a]. Por otro lado, existe el problema que en consultas con respuestas de baja magnitud, el ruido aplicado puede ser muy notable a diferencia de un resultado de mayor magnitud, el cual puede no tener mucha alteración. Se propone para bajar este factor un método denominado *iReduct* [Xiao *et al.*, 2011a], el cual a partir de las respuestas respondidas, se utiliza esta información para redefinir el error de las próximas evaluaciones.

Publicación de *Dataset* Sintético (*Syntetic Dataset Publishing*) : Consiste en publicar información en formato de filas, pero con Privacidad diferencial aplicada.

El objetivo principal es compartir los *dataset* con el público a diferencia de responder consultas. Esto se realiza generando un *Dataset* exclusivo para las respuestas de consultas comúnmente realizadas. Hay dos métodos que logran cumplir con lo anterior, uno de ellos es

aplicar técnicas de anonimato para crear el *dataset*. Otro método es tomar muestras del universo de datos para crear el *dataset*, lo que mantiene la estructura del original sin la misma información.

- **Basado en anonimato**: Se aplica técnicas de anonimato de información, aplicando en cada paso Privacidad diferencial. Por ejemplo, el algoritmo *DiffGen* [Mohammed *et al.*, 2011] cumple con aplicar anonimato en *datasets* para propósitos de Minería de Datos. Consiste en generar particiones, y posteriormente perturbar con ruido. El primer paso es a través de árboles de taxonomía, donde se agrupa la información aplicando mecanismo exponencial para definir las categorías. Posteriormente, se añade perturbación de Laplace para la cuenta de elementos en los grupos.

- **Basado en Aprendizaje**: Otra forma, es utilizando teoría de aprendizaje para aplicar perturbación [Kasiviswanathan *et al.*, 2008] [Blum *et al.*, 2008]. Esto principalmente se basa en aprender de un set conceptual $\mathcal{C}$ del cual posteriormente se utiliza para agregar ruido a las respuestas.

  La precisión de las respuestas dependería en gran cantidad del tamaño del *dataset*, y de los subconjuntos creados. A pesar que los mecanismos cumplen con entregar buenas cotas de privacidad, tienen el problema de ser muy ineficientes en tiempos de cómputos. Este tiempo es exponencial en relación al tamaño del universo de datos con el set conceptual $\mathcal{C}$.

CAPÍTULO 3
ESTADO DEL ARTE

3 .1. Privacidad diferencial

El concepto es planteado por primera vez en el articulo [Dwork, 2011], en el cual se contextualiza la definición de una privacidad efectiva para bases de datos analíticas. Se explica en sí, la idea que la capacidad de ataque a una base de datos analítica (para poder encontrar algún valor en esta) sea siempre la misma, a pesar que la base de datos tenga alguna modificación con un añadido de información o borrado de esta. Este método, a diferencia del anonimato de información, entrega garantías de privacidad aplicadas. Es por esto que es utilizado para estudios posteriores de entregar privacidad a los *datasets* generados.

También se ha estudiado en gran medida la versatilidad de la Privacidad diferencial, para comprobar si efectivamente puede entregar privacidad frente a consultas. Un caso estudiado ha sido cuando se integra Privacidad diferencial en un *dataset*, donde el atacante conoce previamente información de esta cuando no se ha integrado alguna medida de privacidad [Kifer y Machanavajjhala, 2011]. Lo que ocurre en este caso, Privacidad diferencial no lograría ser efectiva, ya que el atacante puede relacionar la información preliminar con la ya procesada. Por ello, el conjunto de datos en primera instancia, tiene que estar restringido de acceso general para garantizar privacidad.

Por otro lado, la definición de Privacidad diferencial solamente entrega una definición matemática para entregar privacidad, pero no se explica como esta puede ser integrada para consultas de base de datos. El implementar de forma práctica Privacidad diferencial es un campo en constante desarrollo. A continuación de mencionan algunos desarrollos a la fecha.

3 .2. Implementaciones Prácticas de Privacidad diferencial

Estas en su mayoría estas son con licenciamiento privado de sus organizaciones, por lo cual en Internet no se encuentran muchas. Aun así, el equipo de investigación de *Uber* recientemente ha publicado en *Github* de forma pública su implementación de Privacidad diferencial, basada en el articulo de consultas *SQL* que utilizaba *FLEX* [Johnson *et al.*, 2018b], para uso general de consultas *SQL* con licencia de código libre [2].

Para poder lograr el objetivo, los autores han investigado diversos métodos de implementación de Privacidad diferencial:

[2]Github de proyecto: https://github.com/uber/sql-differential-privacy

- PINQ (*Privacy Integrated Queries)*
- *wPINQ (Weighted PINQ)*
- *Sensibilidad restringida (Restricted sensitivity)*
- *DJoin*
- *Sensibilidad elástica (Elastic sensitivity)*

Dentro de los anteriores, el método de Sensibilidad elástica es el seleccionado para utilizar en este trabajo, debido a que tiene el soporte para poder encadenar operaciones *JOIN* sobre mapeos de grafos de forma más práctica que otros métodos.

Por otro lado, se podría pensar que sería buena idea implementar Sensibilidad restringida en este trabajo, debido a la naturaleza del método con grafos. Pero esto no sería posible en este caso, debido a que en *RDF* las consultas *BGP* realizadas entrega resultados en formato de tablas a partir de un grafo extraído de la forma de la consulta.

3 .2.1. PINQ

Privacy Integrated Queries (PINQ) es un mecanismo que entrega Privacidad diferencial a consultas de agregación [McSherry, 2009] .

Este mecanismo es un módulo desarrollado para *C#* utilizando un lenguaje declarativo similar a *SQL* denominado *Language Integrated Queries* (LINQ). La implementación tiene el objetivo de entregar a analistas de datos una gran cantidad de opciones de transformación de la información previo a aplicar los mecanismos de Privacidad diferencial, por ejemplo en operaciones como *WHERE*, *GROUPBY*, *JOIN*.

Por otro lado, para separar el uso de la información respecto del análisis de esta, la plataforma presenta una restricción en la información entregada, donde solamente se responden a las operaciones de agregación. Por ello, no se entregan tablas con información al usuario. También implementa un límite de consultas que puede responder hacia el usuario para no sobrepasar la cuota de privacidad definida por ϵ. Es decir, como se ha visto en el marco teórico, una vez sobrepasada esta cota, la información tiene la probabilidad de perder su privacidad, por lo que *PINQ* no respondería más consultas una vez alcanzado este límite.

Respecto a las operaciones *JOIN*, debido a que utiliza su propio lenguaje de consultas, este implementa otro tipo de operaciones nuevas, que lo hace incompatible con otro tipo de base de datos. Por otro lado, para *JOIN*, utiliza su propia implementación para realizarlos. Para operaciones *JOIN*, permite entregar la cantidad de resultados, pero para operaciones de uno a muchos y muchos a muchos, cuenta la cantidad de llaves primarias únicas, pero no el resultado de operación *COUNT* de la operación *JOIN* [Johnson *et al.*, 2018b].

(a) (b)

Figura 17: Representación de una consulta en forma de triángulos. Figura 17a muestra el caso donde los arcos se encuentran conectados a un nodo, aumentando su sensibilidad. Caso de figura 17b muestra la distribución de pesos del triangulo, lo que disminuye la sensibilidad finalmente.

3 .2.2. wPINQ

Weighted PINQ (wPINQ) es una generalización de lo realizado en *PINQ*, con el objetivo de superar varias de las dificultades que tiene el método original en relación con la sensibilidad de operaciones *JOIN*, empleando el concepto de pesos dentro de la información [Proserpio *et al.*, 2014].

Utiliza el mismo lenguaje de su base *LINQ*, pero con mayor flexibilidad dentro de las operaciones, lo que permite que el método pueda trabajar para otro tipo de problemas tales como análisis de uniones en grafos.

Lo principal del método es definir la forma que un cierto dato puede afectar la sensibilidad del *dataset*. Para ello, se caracteriza a través de pesos de los datos, los cuales son asignados dependiendo de la influencia de estos en el resultado final. Por ejemplo en el caso de conexión de nodos a través de triángulos, para el agregado de ruido a través de sensibilidad local (usados en métodos como *Smooth sensitivity*), los pesos de distribuirían de forma uniforme, donde para cada triangulo este sería $1/|V|$ (V corresponde a la cantidad de vértices totales de todos los triángulos) (figura 17a). Con distribución de peso, este se establecería como $1/$ máx$\{d_a, .., d_n\}$, donde el conjunto $\{d_a, .., d_n\}$ corresponde al grado de unión de un nodo (cuantos arcos se conectan a este nodo) (figura 17b).

Para representar los *datasets*, se utilizan cadenas de Markov para la distribución de los pesos. El enfoque de *wPINQ* es trasformar la información, y más que calibrar el ruido, se calibra la información entregada. De esta manera, si hay información que puede ser más delicada de procesar por su sensibilidad, entonces esta puede modificarse en el *dataset* a procesar.

Respecto al soporte de operaciones de operaciones *JOIN*, la distribución de pesos estabiliza la sensibilidad para que esta sea en promedio 1. También integra soporte para poder retornar el resultado de uno a muchos nodos, y muchos a muchos nodos, lo que permite el análisis de múltiples uniones de información. Aun así, sigue no siendo compatible con otro tipo de bases de datos, por lo cual no podría servir para ser utilizado con *RDF*.

3 .2.3. DJoin

Dentro de las implementaciones que utilizan Privacidad diferencial se han investigado diversos mecanismos, pero ninguno de estos ha sido pensado en la necesidad de utilizar información distribuida entre dos bases de datos de diferentes orígenes. Por ejemplo, se desearía analizar la relación entre personas que tienen alguna enfermedad junto a algún viaje realizado. Para esto es necesario realizar consultas tanto a la agencia del viaje como a los registros médicos de la persona, presentando dos bases de datos a consultar.

Por ello se establece el mecanismo denominado *DJoin* (*Distributed Join*), el cual cubre el caso anterior de consultar hacia dos bases de datos para responder una consulta, pero incorporando las nociones de la Privacidad diferencial [Narayan y Haeberlen, 2012].

DJoin obtiene la información desde los servidores de información (aquellos los cuales se busca procesar la unión de información), posterior a ello se realizan múltiples procedimientos de criptografía para mantener resguardada la información. Por otro lado, las operaciones de base de datos se traducen matemáticamente en otro tipo de transformaciones, por ejemplo, una consulta `SELECT COUNT(A.x) FROM A,B WHERE A.x=B.y`, es decir, obtener un mapeo de la información juntando el atributo `X.A = Y.B` se traduce en la intersección de los conjuntos de ambos $\pi_a(X) \cap \pi_b(Y)$. Por lo anterior, *DJoin* siempre trabaja con la intersección de información. Toda información que no este relacionada es ignorada de la consulta realizada.

Debido a que se utilizan múltiples procedimientos de Criptografía junto a consultas en lenguaje relacional, este método no podría implementarse dentro de *RDF*. Además, el soporte de consultas *JOIN* sería solamente para uno es a uno, mientras que uno a muchos y muchos a muchos no es compatible.

3 .2.4. FLEX (Aplicación de Sensibilidad elástica)

Muchas implementaciones realizadas a la fecha han sido definidas para usos especiales, tales como analizar el comportamiento de navegación en Internet, o el uso de *emojis* en teclados. El problema ha sido que aún no se ha presentado un enfoque analítico del uso de técnicas de Privacidad diferencial para información. Este documento es uno de los primeros en ofrecer este tipo de implementación para bases de datos relacionales.

La solución se crea a partir de un análisis de las consultas que se realizan en un *dataset*, en este caso específicamente de *Uber*, donde en general se obtiene el resultado que el $62{,}1\,\%$ de las consultas realizadas utilizan *JOIN*. Por otro lado, un tercio del total de consultas ($34\,\%$) son de agregación, mientras que el otro porcentaje es de información pura. Para el primer conjunto, el uso de Privacidad diferencial es ideal para ser implementado. Lamentablemente para el segundo caso de obtener información pura, el estado del arte no es muy avanzado para poder entregar un *dataset* que cumpla con Privacidad diferencial, por lo que no se con-

sidera en esta técnica.

Debido a la ausencia de mecanismos de Privacidad diferencial a la fecha de la publicación, existían diversos métodos de Privacidad diferencial que entregan privacidad para analíticas de datos, pero estos mecanismos no tienen soporte a muchas de las características de bases de datos con *SQL* utilizadas en la práctica. Es bajo este contexto, donde se propone la idea de utilizar Sensibilidad elástica (sección 2 .9), el cual es una aproximación de la Sensibilidad local pero para *equijoins* en general.

Los investigadores del concepto anterior [Johnson *et al.*, 2018b] aplican dichos conceptos para crear *FLEX*, el cual es una implementación que permite entregar resultados que cumplen con este protocolo a través de consultas *SQL*.

La implementación sigue el procedimiento teórico, el cual depende de los parámetros de Privacidad diferencial (ϵ, δ), y el tamaño del los mapeos:

- Calcular valor de $\beta = \epsilon/2 \log(2/\delta)$, donde $\delta = n^{-\epsilon \log n}$
- Calcular la Sensibilidad elástica máxima: $S = \text{máx}_{k=0,1,\ldots} e^{(-\beta k)} \hat{S}^{(k)}(q, x)$
- Entregar el resultado de $f(x) + \text{Lap}(2S/\epsilon)$

Lo que entrega resultados que cumplen con Privacidad diferencial.

3 .3. Privacidad diferencial en base de datos de Grafo

El estudio de aplicación de Privacidad diferencial en base de datos de grafo no es tan avanzado como lo es en el caso de relacional, debido a la naturaleza de sus paradigmas. Pero aun así, han surgido diversos estudios del como poder implementarla en grafos.

Un caso de interés, es el poder implementar Privacidad diferencial en base de datos de redes sociales [Hay *et al.*, 2009], donde se ha implementado algoritmos para poder entregar privacidad en la información. Esto por el motivo que investigaciones anteriores se concentran en agregar anonimato a la información, lo cual no es lo mismo que asegurar privacidad (se ha mencionado anteriormente, que el anonimato no necesariamente genera garantías de privacidad). También se ha realizado investigaciones para Privacidad diferencial para arcos de un grafo, aplicando una transformación a grafo a través de "*Attributed Graph Model*" [Jorgensen *et al.*, 2016].

Dentro de los avances de Privacidad diferencial dentro de Grafos, un concepto estudiado es el de Sensibilidad restringida, el cual se centra en la estructura de un Grafo, estudiando la influencia de tanto arcos o vértices, como métricas de distancia.

3 .3.1. Sensibilidad restringida (*Restricted sensitivity*)

Se ha descrito en el concepto anterior sobre la implementación *FLEX* para bases de datos *SQL*, donde se utiliza el concepto de *Elastic sensitivity* para poder entregar información que cumple los conceptos de Privacidad diferencial.

Lo anterior, puede reutilizarse para el análisis de base de datos de grafo.Aun así, para el concepto de sensibilidad a diferencia de las bases de datos relacionales presenta un problema. Las bases de dato de grafo pueden presentar sensibilidades muy grandes por su naturaleza de conexión de variables, la magnitud del ruido agregado a un resultado pueden ser tan grande, que el resultado sería inutilizable.

Un ejemplo de ello es un grafo de red social con muchos nodos. Dependiendo de la métrica de distancia, puede quitarse un arco el cual se conecta a un cierto nodo con muchos amigos. Si en primera instancia se midiera la sensibilidad de un nodo con un arco conectado a pocos amigos, entonces el resultado de la sensibilidad sería bajo. Pero si está conectado a un cierto nodo el cual se encuentra conectado a otra gran cantidad de nodos de forma reiterativa, el valor de la consulta puede incrementar inclusive a la cantidad total de nodos n (concepto de Sensibilidad global).

Para poder evitar que algo como lo anterior ocurra en base de datos de grafo, se aplica el concepto de Sensibilidad restringida (*Restricted sensitivity*).

El método consiste en utilizar una proyección del conjunto de *datasets* posibles, llamado *dataset* de Hipótesis ($\mathcal{H}$). Dentro de los conjuntos de Hipótesis, uno práctico para este estudio es limitar el grado de los vértices de un grafo ($deg(v)$) de la forma: $\mathcal{H}_k = \{G \in \mathcal{G} : \forall v, deg(v) \leq k\}$, donde $G \in (G)$ son grafos o redes sociales ([Blocki *et al.*, 2013]).

Con este concepto, al limitar el grado máximo de los vértices (en el caso de este estudio, limitando valores como por ejemplo la máxima frecuencia de algún nodo), entonces la sensibilidad en el peor caso no sería tan alta, entregando resultados más utilizables.

Con lo anterior, la Sensibilidad restringida se define con la siguiente expresión:

$$RS_f(\mathcal{H}) = \max_{D_1, D_2 \in \mathcal{H}} \left(\frac{|f(D_1) - f(D_2)|}{d(D_1, D_2)} \right) \tag{26}$$

Donde $d(D_1, D_2)$ representa la distancia más pequeña posible (utilizando métricas de Vértice o Nodo) entre cualquier *dataset* D_1 y D_2 perteneciente al universo de *datasets* posibles dentro de $\mathcal{H}$.

Este concepto, posteriormente permite realizar consultas $f_h(D)$ sobre $\mathcal{H}$. La Sensibilidad global de esta consulta es la equivalente a la Sensibilidad restringida para el conjunto del *dataset* original, como es demostrado en [Blocki *et al.*, 2013].

Aun así, el procedimiento anterior no es muy eficiente para múltiples consultas, por lo que se presentan otras formas de calcular la Sensibilidad restringida utilizando conceptos de métricas. Se busca los siguientes objetivos para entregar resultados con privacidad:

- Entregar un resultado no muy cercano al posible real. Pero que aun así, preserve la privacidad del *dataset*.
- Entregar un resultado con mayor precisión, debido a que el subconjunto utilizado tiene una mayor coincidencia con el resultado real. Esto quiere decir, que el *dataset* utilizado $\mathcal{H}_k = D$, por lo que el resultado es el esperado.

***Smooth Proyection* - Arco Adyacencia** En primer lugar se tiene que definir unos conceptos que permitan la creación de proyecciones de *datasets*, para crear los de hipótesis $\mathcal{H}$. Primero, se tiene el concepto de distancia "*c-smooth*", el cual señala que la máxima distancia entre dos *datasets* G_1 y G_2, es un valor *c*.

Continuando la idea anterior, una proyección "*c-smooth*" se define como una función $\mu : \mathcal{D} \rightarrow \mathcal{H}$ aplicada al universo de *datasets*, donde se cumple para sus *datasets* $d(\mu(D), \mu(D')) \leq c$. Luego para consultas sobre la proyección, estas cumplen con $GS_{f_h} \leq c * RS_f(\mathcal{H})$.

Con los conceptos anteriores, para un modelo de adyacencia de arcos, se requiere de una "*3-Smooth proyection*" ($d(\mu(D), \mu(D')) \leq 3$) según explica [Blocki *et al.*, 2013], el cual esta basado en diversos casos que pueden ocurrir en el valor de la distancia para arcos al aplicar proyección sobre *dataset*.

Con todo lo anterior, se puede obtener un concepto para poder obtener Privacidad diferencial $(\epsilon, 0)$ (forma estricta) con la siguiente expresión:

$$A(f, G) = f(\mu(G)) + Lap(3 * RS_f(\mathcal{H}_k/\epsilon)) \tag{27}$$

Por lo cual, es necesario obtener el *Dataset* en particular, calcular la proyección de este, y obtener la Sensibilidad restringida, para obtener el resultado de la consulta con ruido agregado $A(f, G)$.

***Proyections and Smooth distance estimators* - Vértice Adyacencia** Para la métrica de Vértices, es un problema NP-Difícil calcular la distancia $d(G, \mathcal{H}_k)$, por lo que no se puede obtener una proyección de complejidad $O(1)$. Aun así, existe una forma eficiente para poder aproximar este valor. Para ello, se utilizan diversos mapeos del *dataset* de hipótesis subconjuntos de este ($\bar{\mathcal{H}} \supset \mathcal{H}$).

La función $\hat{d}_\mu : \mathcal{D} \rightarrow \mathbb{R}$, denominada estimador de distancia *c-smooth*, satisface lo siguiente:

- Para cada $D \in \mathcal{H}$: $\hat{d}_\mu(D) = 0$
- Esta acotado inferiormente (*lower bounded*) por la distancia de D a su proyección: $\forall D \in \mathcal{D}, \hat{d}_\mu(D) \geq d(D, \mu(D))$
- Valor sobre las bases de datos vecinas, cambia en a lo más c : $\forall D \sim D', |\hat{d}_\mu(D) - \hat{d}_\mu(D')| \leq c$

Esta distancia posteriormente, se puede calcular como un problema de satisfacción de restricciones, el cual se compone de lo siguiente:

Fijar $\bar{\mathcal{H}} \supset \mathcal{H}$ y permitir $\mu : \mathcal{D} \to \bar{\mathcal{H}}$ ser una proyección de $\mathcal{H}$. Hacer de $\hat{d} : \mathcal{D} \to \mathbb{R}$ un estimador de distancia computable *c-smooth*. Para cada consulta f, se define la composición $f_h = f \circ \mu$. Se define la función:

$$S_{f_{\mathcal{H}},\beta}(D) = \max_{d \in \mathbb{Z}, d \geq \hat{d}(D)} e^{(-\frac{\beta}{c}(d - \hat{d}(D)))}(2d + c + 1) * RS_f(\bar{H}) \tag{28}$$

Ahora, se explica el como se puede obtener la proyección μ, y el *smooth distance estimator* para el modelo de vértice adyacencia (el $\hat{d}_\mu$)

Se tiene que realizar una Programación Lineal, y determinar una distancia fraccional de un grafo hacia $\mathcal{H}_k$. El problema tiene $n + \binom{n}{2}$ variables, donde:

- x_u : Indica si vértice u tiene que ser removido o no
- $w_{u,v}$: Si el arco entre u y v tiene que continuar en el grafo proyectado o no
- $a_{u,v}$: Si es 1, el arco $\{u, v\}$ se encuentra en G, se lo contrario (no se encuentra), $a_{u,v}$ = 0

Restricciones:

$min \sum_{v \in V} x_v$ s.t.

- $\forall v, x_v \geq 0$
- $\forall u, b, w_{u,v} \geq 0$
- $\forall u, v, a_{u,v} \geq w_{u,v} \geq a_{u,v} - x_u - x_v$
- $\forall u, \sum_{v \neq u} w_{u,v} \leq k$

Para convertir la solución que se obtiene $(\bar{x}^*, \bar{w}^*)$ a un grafo $\mu(G) \in \mathcal{H}_{2k}$, se define removiendo todos los arcos $(u, v) \in E(G)$ donde sus puntos finales tenga un peso $x_u^* > 1/4$ o $x_v^* \geq 1/4$.

Con los pasos anteriores resueltos, se puede utilizar Privacidad diferencial para cualquier consulta de redes sociales $f(D)$ junto al mecanismo de proyección μ de arcos que señala la existencia de una proyección computable "*3-smooth*" para $\mathcal{H}_k$, y la cota superior (*upper bound*) β *smooth* de ecuación 30, responde a esta consulta utilizando:

$$A(f, G) = f(\mu(G)) + Lap(2 * S_{f_h, -\epsilon/2ln\delta}(G)/\epsilon) \tag{29}$$

Esto preserva (ϵ, δ) Privacidad diferencial para cada grafo G.

Por lo tanto, como se ha planteado en el caso de arcos, se tiene que calcular la proyección $\mu(G)$ y realizar la consulta sobre esta. Luego, se calcula el valor de $S_{f_h, -\epsilon/2ln\delta}(G)$, con la fórmula anteriormente descrita, denominada estimador de distancia "*smooth*".

3 .4. Arquitecturas

Como todo sistema, este tiene que tener una arquitectura en la cual se presenten módulos de software que interactúen entre si, para cumplir la necesidad del usuario con la plataforma.

3 .4.1. Integración Profunda

Estas arquitecturas utilizan su propia bases de datos para poder realizar consultas aplicando Privacidad diferencial. Las bases de datos particulares son construidas desde la información privada, sobre las cuales se aplican diferentes técnicas del método implementado (tales como conteo o uniones) para poder cumplir con proteger la privacidad. Esto por otro lado, entrega la desventaja de requerir un nuevo motor de base de datos, que tiene que ser integrado la arquitectura existente de software, entregando a este una mayor complejidad. Por otro lado, se tiene que crear otros sistemas para poder integrar nuevos mecanismos de privacidad.

Métodos de Privacidad diferencial que siguen esta arquitectura son *PINQ*, *wPINQ*, *GUPT* y *Airavat*.

3 .4.2. Procesamiento Posterior

A diferencia del método anterior, no se crea una nueva base de datos para responder consultas; al contrario, se utiliza un módulo que permite procesar consultas a partir de la información. Este posteriormente utiliza la misma consulta en la base de datos existente, para luego aplicar las técnicas de Privacidad diferencial en la respuesta entregada por el *dataset* real.

De esta mantera, entregar al usuario a través de las aplicaciones de privacidad un resultado seguro.

Esta permite mayor flexibilidad que la Integración Profunda, esta arquitectura podría adaptarse al modelo de datos y a los tipos de consultas realizadas. Por otro lado, lo negativo sería que la compatibilidad estaría relacionada con su objetivo en particular, por lo que no sería utilizable en múltiples plataformas.

Métodos que utilizan esta arquitectura son Sensibilidad elástica y Sensibilidad restringida.

3 .4.3. *Chorus*: Privacidad diferencial a través de reescritura de consulta

La idea de esta arquitectura es poder entregar los resultados de una consulta de un usuario aplicando Privacidad diferencial, pero la privacidad sería aplicada antes de la ejecución de la consulta del usuario. Así de forma automática, siempre se entregarían resultados que cumplen con las promesas de privacidad [Johnson *et al.*, 2018a].

Chorus es desarrollado bajo dos de los mayores desafíos que tiene que tener una implementación de Privacidad diferencial. Entre estos, es integración dentro de las arquitecturas y sistemas de datos utilizados en la práctica, esperando mantener la consistencia y rendimiento. Otro es el soporte de diversos mecanismos. *Chorus* cumple con estos desafíos: en primer lugar, permite una mayor consistencia y escalabilidad al solo utilizarse el mismo motor de base de datos para poder responder consultas. Por otro lado, *Chrous* soporta múltiples mecanismos, lo que permite responder una mayor cantidad de consultas.

Este mecanismo utiliza el procedimiento de reescritura de consultas (*Query Rewritting*), en cual consiste en procesar la consulta realizada por el usuario, y luego a partir de patrones y reglas, crear una nueva consulta dentro de la base de datos, para que sea el motor de base de datos quien finalmente realiza los procedimientos de obtener la información y aplicación de ruido. Esto permite una mayor eficiencia dentro de la respuesta a consultas, y también bajar los elementos de una arquitectura, lo que la hace más simple de implementar y utilizar.

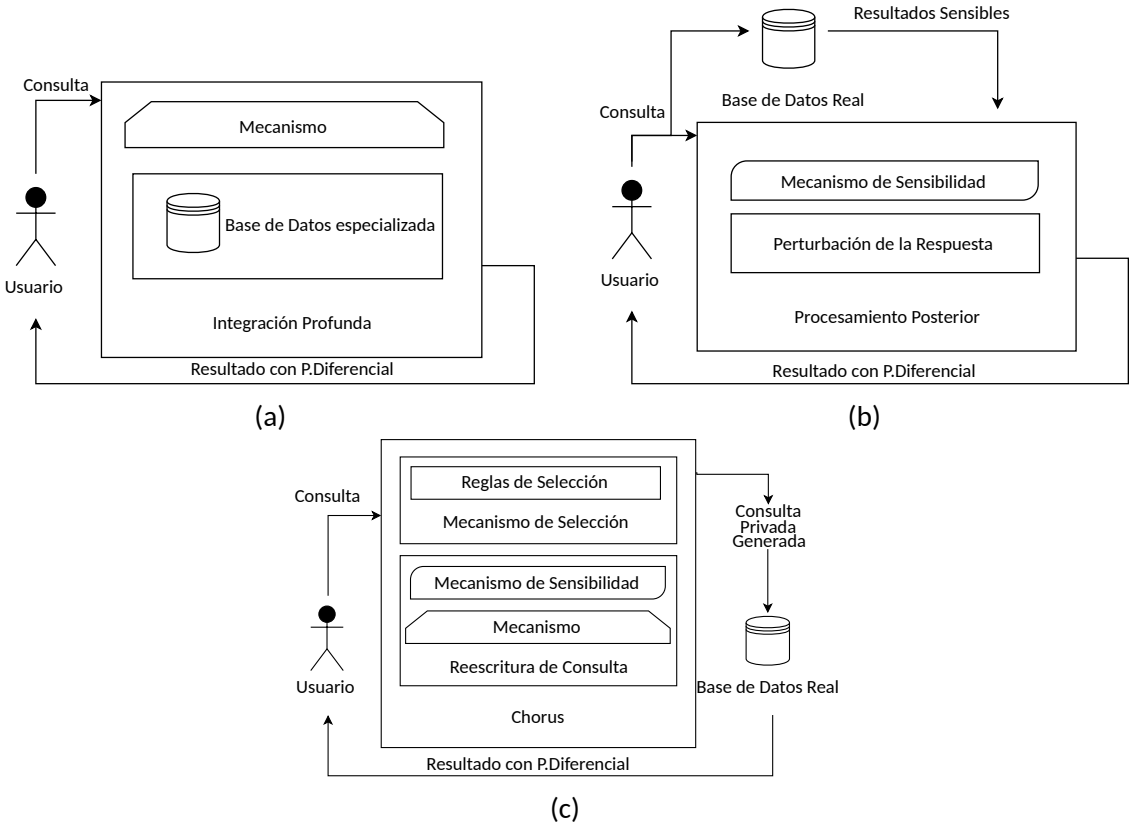


Figura 18: Arquitecturas de Privacidad diferencial: Integración profunda (figura 18a), Procesamiento posterior (figura 18b) y arquitectura *Chrous* (figura 18c).

CAPÍTULO 4
PROPUESTA DE SOLUCIÓN

Para lograr los objetivos señalados en la definición del problema (sección 1 .2), se implementará una solución de software que permita responder consultas de conteo, preservando la privacidad de la información a través de Privacidad Diferencial, realizando los siguientes procedimientos:

- Desarollo de una arquitectura de software, la cual permita realizar consultas privadas sobre bases de datos de grafo RDF, utilizando el lenguaje de consultas SPARQL.
- Dentro de esta arquitectura, incluir un módulo de software el cual implemente los conceptos de Privacidad diferencial, entregando a esta arquitectura la capacidad de responder consultas que aseguran la privacidad de la información.
- Evaluar la aplicación y los resultados obtenidos.

4 .1. Conceptos y Herramientas

4 .1.1. *Resource Description Framework* (*RDF*)

Es un conjunto de especificaciones de la *W3C* para metadatos de contenido Web. Posteriormente, se utiliza para poder describir recursos generales dentro de la Internet, con la finalidad que estos se puedan relacionar con otros recursos a través de etiquetas (utilizadas en el predicado), y así un computador pueda relacionar esta información de forma ordenada.

Aun así, para que un computador pueda procesar información, esta debe estar en un formato determinado, de lo contrario no se podría analizar. Si esta se presentará en un formato que requiere de múltiples procesamientos de la información, en términos de rendimiento puede ser costoso realizar esta operación. Por lo anterior, es un requerimiento que esta se encuentre serializada en formas de tripletas. Los formatos más conocidos para esto son los siguientes:

- Turtle (.ttl) : Formato compacto y legible.
- N-Triples (.n3) : Formato simple, y amigable para su parseo por programas externos. Aun así no es compacto como "*.ttl*".
- N-Quads : Formato para almacenar supersets de N-Tripletas. Útil para la serialización de múltiples grafos *RDF*.

```
<?xml version="1.0"?>

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:si="https://www.w3schools.com/rdf/">

<rdf:Description rdf:about="https://www.w3schools.com">
  <si:title>W3Schools</si:title>
  <si:author>Jan Egil Refsnes</si:author>
</rdf:Description>

...

</rdf:RDF>
```

Figura 19: Ejemplo de expresión en *RDF/XML*

- JSON-LD : Formato basado en *JSON*.
- RDF/XML : El primer formato utilizado para serializar *RDF*. Utiliza notación *XML*.
- RDF/JSON : Alternativa a *JSON-LD*.
- Header Data Triples (*.hdt*) : Formato que entrega compresión y eficiencia al uso de *HDT*. Creado con la idea de ser utilizado para *Big Data*.

4 .1.2. *Apache Jena*

Framework desarrollado en Java, el cual incluye múltiples herramientas para poder trabajar con Web Semántica a través de manipulación directa a grafos *RDF*, tales como una *API* con funciones para manipular estas estructuras y una extensión del lenguaje *SPARQL* con *ARQ*. También se incluyen herramientas de servidor de base de datos para poder trabajar con múltiples tripletas. Con *Jena Fuseki*, se puede tener un servidor de base de datos de grafo, permitiendo utilizar otro tipo de aplicaciones o lenguajes que interactúen con un servidor *RDF*.

Finalmente, se incluyen herramientas de *Web Ontology Language* (*OWL*), que extienden el uso de tripletas con clasificadores de clase dentro de los modelos.

4 .1.3. *Header Triples Dictionary* (*HDT*)

Es un formato de serialización de tripletas para bases de datos *RDF*, el cual entrega múltiples ventajas en relación a las implementaciones existentes, tales como:

- Menor tamaño de archivos, que permite menores consumo de información.
- Indexación de los tripletas, lo que permite una mayor agilidad en realizar consultas
- Compresión de la información, permitiendo que se pueda procesar mejor en memoria.

Lo anterior permite una mejora de rendimiento en comparación a otras serializaciones, siendo siendo una gran alterativa para aplicaciones que requieren escalamiento. Dentro del sitio de *HDT*, se incorporan diversas opciones para poder utilizar el formato, tales como herramientas en *C++* o Java [Fernández *et al.*, 2013]. En la versión de Java, se incluye una extensión del servidor web de *Apache Jena*: *Fuseki*, el cual entrega compatibilidad para poder utilizar este tipo de archivos.

4 .2. Implementación de sistema de bases de datos de grafo

Con conocimiento de las herramientas anteriores, se procede a implementar el sistema propuesto. En primer lugar, debido a que se busca trabajar con base de datos de grafo *RDF*, es necesario tener las herramientas que permitan entregar un entorno utilizando este tipo de datos. Por otro lado, como se busca que el sistema sea tanto eficiente como escalable (en caso de requerir mayor demanda), se tiene que elegir las alternativas que puedan entregar estas mejoras para el desarrollo, por lo que se selecciona el formato *HDT* para el almacenamiento de tripletas.

La herramienta que cumple con estos requerimientos, es el *fork* de *Apache Jena* realizado por el equipo de *HDT*[3], que ha modificado el código fuente de *Apache Jena* para que sea compatible con este tipo de formato.

Como se ha descrito en el marco conceptual, los conceptos de Privacidad diferencial son netamente estadísticos matemáticos, por lo que es necesario tener un módulo que pueda complementarse con el servidor de *Fuseki* para consultas, implementando los conceptos matemáticos mencionados en el Marco Conceptual, tales como sensibilidad, ruido, entre otros. Para este objetivo, se utiliza el lenguaje *Python*.

Esto implica que no se puede utilizar todas las características de *Apache Jena* dentro del código, ya es nativamente para Java. Aun así, lo que interesa para el desarrollo de esta implementación, es el poder usar el servidor para poder realizar diversas consultas *SPARQL* dentro

[3]Enlace de repositorio: https://github.com/rdfhdt/hdt-java/tree/master/hdt-fuseki

del archivo de tripletas, por lo que se utiliza la *API REST* del servidor de *Apache Jena/Fuseki*, para poder integrarlo junto a *Python*.

Continuando con *Python*, el lenguaje tiene una gran extensibilidad para poder trabajar con matemática y estadísticas, tales como la librería *NumPy* y *SciPi*. Ademas, al ser un lenguaje utilizado para tanto desarrollo web como para estudios matemáticos, se puede usar librerías de consultas web. Una implementación completa para *RDF*, es la biblioteca *SPARQLWrapper*[4], que permite conectarse a servidor web de *Jena Fuseki*, obteniendo la posibilidad de entregar la información de las consultas en un diccionario *Python*.

Con lo anterior, se tiene lo necesario para poder realizar una implementación que permita evaluar la Privacidad diferencial del *dataset* a partir de las métricas de distancia. Esta presenta la arquitectura mostrada en la figura 20, que cumple con un esquema de arquitectura de procesamiento posterior.

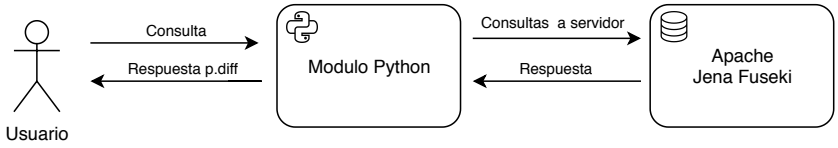


Figura 20: Arquitectura básica de implementación.

4 .3. *Dataset* a utilizar

Los *dataset* a utilizar, se basan en un *dataset* diseñado para pruebas de stress en sistemas *RDF* del conjunto de herramientas *Waterloo SPARQL Diversity Test Suite* (*WatDiv*)[Aluç *et al.*, 2014][5], el cual tiene múltiples factores que pueden ser útiles para el estudio, tales como múltiple tipos de consultas, gran cantidad de tipo de información (útil para pruebas de stress), entre otros.

La particularidad de este generador de *dataset*, es que permite realizar consultas de estilo tanto estrella, camino, o unión de estrellas, lo cual es lo necesario para poder utilizar los conceptos de Sensibilidad elástica.

4 .4. Módulo Python para Privacidad diferencial

Con motivo de la no existencia de implementaciones de Privacidad diferencial para bases de datos *RDF* para Python, se implementará un módulo como parte de este trabajo. Aun así, se

[4]https://rdflib.github.io/sparqlwrapper/
[5]URL de *Dataset*: https://dsg.uwaterloo.ca/watdiv/watdiv.10M.tar.bz2

ha mencionado anteriormente que existe actualmente una implementación para *SQL* realizada para *Uber* para el caso de Sensibilidad elástica[6], la cual será utilizada como referencia en la implementación.

Se estudiará el código de esta aplicación en *SQL*, y se realizará una adaptación en *Python* de las partes necesarias para poder aplicar Privacidad diferencial.

Dentro del estudio, se ha realizado una implementación para obtener la Sensibilidad elástica en base de datos *RDF*. Posteriormente, dentro del cálculo de Sensibilidad, se realizara un estudio de las métricas de distancia en el procesamiento del *dataset*.

4 .4.1. Principales bibliotecas de *Python* utilizadas

- *NumPy* y *SciPi* : Bibliotecas para agregar funcionalidades matemáticas a *Python*. Principalmente, la escritura de ecuaciones junto a operaciones de exponentes para Sensibilidad elástica, y utilizar la Distribución de Laplace para realizar consultas con Privacidad diferencial.
- *SparqlWRAPPER* : Implementa la funcionalidad para poder interactuar con un servidor de *Apache Jena/Fuseki*.
- *RDFlib* : Para realizar operaciones "*CONSTRUCT*" es necesaria esta librería, debido a que esta operación es la encargada de generar un nuevo *dataset* a partir de transformadores *JOIN* . Con esta librería, se puede trabajar con este *Dataset* cargado en memoria dentro de *Python*.

4 .4.2. *Elastic sensitivity* (Sensibilidad elástica)

Para poder implementar este concepto dentro de *Python*, se define lo siguiente:

Definición de variables para Privacidad diferencial

- ϵ : Se utiliza el valor de $0{,}1$. Para comparaciones, también se puede utilizar otros valores.
- δ : Se utilizara la expresión $n^{-\epsilon \ln n}$, la cual tiene relación respecto al tamaño del *dataset* n [Johnson *et al.*, 2018b]

[6] https://github.com/uber/sql-differential-privacy

El próximo paso es realizar las consultas señaladas a partir de un archivo de entrada de texto, donde se definen las consultas que se realizarán dentro del programa. Estas también pueden estar separadas en una carpeta, con la extensión "*.rq*". De estas se extraen los *Basic Graph Patterns*, para poder verificar a que tipo de consulta pertenece (estrella o si requiere de transformador *JOIN*), y obtener las frecuencias máximas de estos en particular. Lo anterior se conserva en memoria, y con este se comienza a realizar el análisis de Sensibilidad como lo señala la expresión 20, 21, 22.

Con los pasos anteriores, se obtiene el valor de la Sensibilidad resultante de la consulta, con el cual es posible aplicar Privacidad diferencial al resultado real de la consulta. A este valor se le suma una perturbación aleatoria, obtenida a través de distribución de Laplace (método "*numpy.random.laplace()*" de *numpy*), utilizándose como argumento de la distribución la sensibilidad junto a ϵ, como se demuestra en la ecuación 14.

4 .5. Arquitectura

El Software será implementado utilizando diversas Herramientas de Software, las cuales interactúan entre estas para poder entregar un resultado (figura 21).

El software se divide en seis componentes con una tarea en particular, los cuales son:

- *app.py* : El usuario tiene interacción con esta parte. Se hacen llamados a las otras funciones, para realizar el flujo de consultas.
- *sparql.py* : Se comunica con el servidor *Jena/Fuseki*, y permite a través de código *Python*, obtener valores y atributos del servidor.
- *functions.py* : Implementa lo necesario para poder aplicar Sensibilidad elástica. Utiliza la estructura, y las máximas frecuencias del *BGP* entregado, para realizar cálculos matemáticos, y obtener las variables de Elastic sensitivity.
- *results.py* : Posterior a obtener los parámetros de Sensibilidad elástica necesarios, esta funcionalidad realiza la consulta real que necesita el usuario, y posteriormente entrega resultados con Privacidad diferencial.
- *Jena/Fuseki* : *Framework* de *RDF/SPARQL* para entregar un servidor *REST* para realizar consultas desde aplicaciones externas.
- *Dataset* : Es el conjunto de Datos *RDF* utilizado para poder realizar las pruebas. Se encuentra en formato *HDT*, para optimizar rendimiento, y uso de disco.

Con ello, se tiene un software que puede aceptar consultas de *SPARQL*, el cual realiza los siguientes pasos para poder entregar el resultado de una consulta:

- Recibir una consulta tipo *COUNT* en lenguaje *RDF*, donde dentro de esta, se tiene una secuencia de tripletas *RDF* que forman un *BGP*.
- El módulo *Python* interactúa con el servidor Apache *Jena/Fuseki*, para realizar operaciones con el *BGP* entregado, obteniendo diversos parámetros necesarios para entregar Privacidad diferencial, tales como el largo del *dataset* generado por el *BGP*, comprobar de que tipo de consulta se entrega (estrella, uniones, etc).
- A partir de los parámetros anteriores, realizar los cálculos necesarios para poder utilizar la técnica de Sensibilidad elástica dentro del *dataset*, obteniendo el valor de parámetros tales como β y δ.
- Con la información anterior, se procede a calcular el valor de la Sensibilidad elástica, utilizando para ello la estabilidad, y comprobando el valor máximo de S iterando por cada distancia k.
- Al tener la información, se procede a entregar el valor de la consulta con privacidad agregada. Para ello, se utiliza la Distribución de Laplace, y se entrega un resultado aleatorio que utiliza como argumento el tamaño del *dataset*, y la Sensibilidad elástica obtenida en el paso anterior.
- Los resultados son posteriormente exportados a un archivo separado por comas (*.CSV*), para el estudio de los resultados.

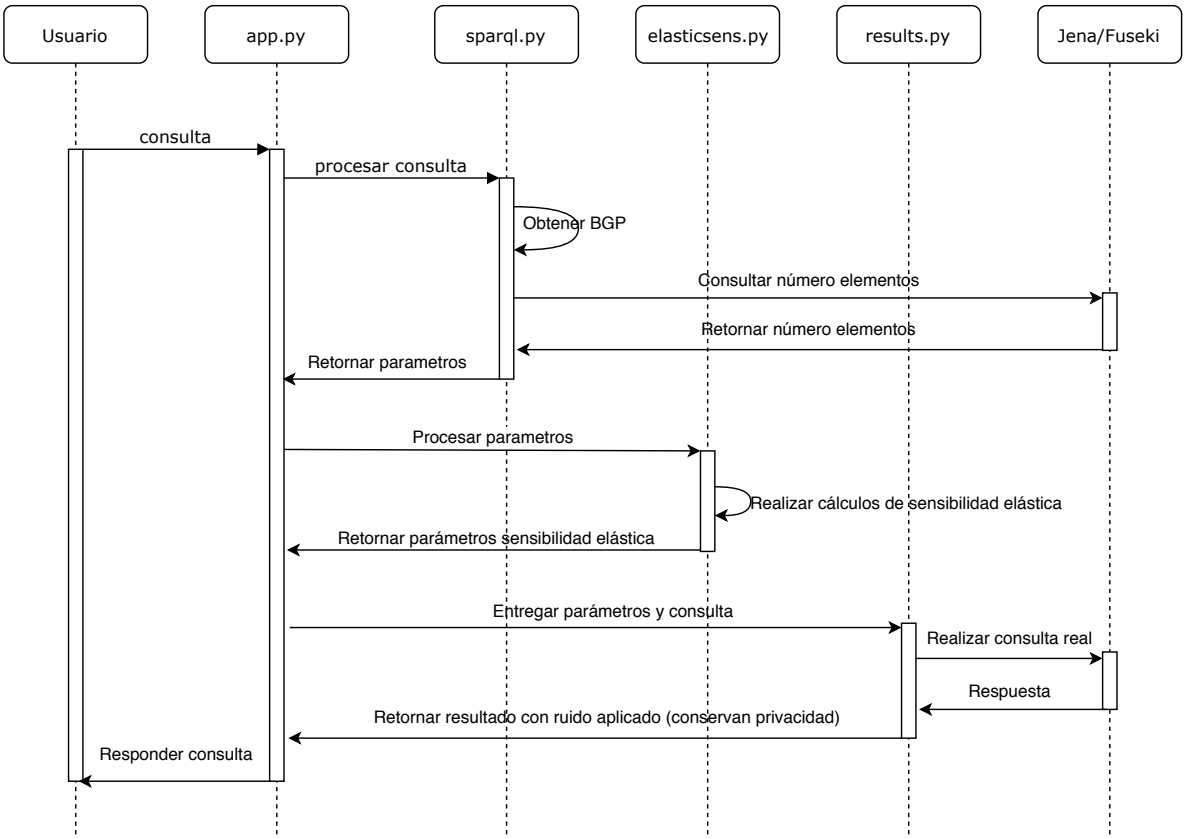


Figura 21: Diagrama de flujo de la interacción entre las diversas partes de la aplicación.

CAPÍTULO 5
VALIDACIÓN DE LA SOLUCIÓN

Posterior a la implementación, se procede a realizar pruebas con consultas *COUNT* al programa. Estos resultados se exportan en hojas de cálculo, con las cuales se estudiaran para comprobar que efectivamente la implementación cumple con el objetivo de entregar resultados que aseguran Privacidad diferencial. En esta sección, se describirán las pruebas realizadas a la implementación. Posteriormente se presentan sus resultados, y finalmente se estudiará la validación de estos.

5 .1. Expectativas de la propuesta

Al tener un sistema de consultas analíticas para usuarios, el objetivo de ellos es utilizar la plataforma para poder obtener información de interés. Para atacantes, es obtener el resultado real para poder vulnerar privacidad, utilizando esta para la unión con otros *datasets* relacionando la información.

La implementación desarrollada busca evitar el segundo caso, pero ello tiene como consecuencia que el primer caso no sería el resultado real, pero que aun así, este resultado tiene que tener sentido y utilidad para el objetivo del usuario de obtener información de la base de datos. Para ello, se aplicará Privacidad diferencial a los resultados de la implementación realizada, y a esta a través de "*results.py*", se añade código para obtener indicadores estadísticos que pueden entregar información sobre la fiabilidad de las soluciones entregadas.

Por lo tanto, se espera de los resultados que cumplan con lo siguiente:

- Ningún resultado sea exactamente igual al original, de lo contrario como consecuencia, el atacante obtendría su objetivo.
- El ruido aplicado sobre los resultados sea de una magnitud aceptable, debido a que esto tiene como consecuencia un resultado que no es coherente con lo que es usuario espera.

Las pruebas han sido ejecutadas en un computador Asus X556UR con 8GB de Ram, procesador Intel Core i5 7200 2.50Ghz, con la distribución Linux *Manjaro* 18.0.4, utilizando el Kernel 4.14.118 x86-64. Se utiliza la versión de *Python* 3.71, junto a *Java Runtime Environment* versión 9.0.4.

5 .2. Experimentos realizados

5 .2.1. Sensibilidad de *JOIN* y valor de ϵ

Para comenzar, se utiliza un conjunto de consultas *SPARQL* (tabla 3), las cuales se encuentran guardadas por separado en una carpeta de archivos. Estas se entregan a la implementación como argumento de ejecución. Además, se realizan las pruebas utilizando tres valores para el grado de privacidad ϵ, los cuales son $0{,}1$, $0{,}4$ y $0{,}8$, donde el valor $0{,}1$ representa un grado alto de privacidad.

El programa realizará 10 iteraciones de respuestas a las consultas realizadas, el cual se considera un número considerable para poder ser utilizado en la práctica.

Consulta	Archivo	Tipo	BGP	Variables Join	Sensibilidad
Query 1	query_1074.rq	Snowflake	10	3	$k + 1188$
Query 2	query_2019.rq	Snowflake	10	3	$k + 1188$
Query 3	query_1777.rq	Snowflake	8	2	$k + 13$
Query 4	query_1115.rq	Snowflake	8	2	$k + 13$
Query 5	query_1783.rq	Star	4	1	k
Query 6	query_123.rq	Snowflake	4	2	$k + 4243$
Query 7	query_1368.rq	Star	3	1	k
Query 8	query_117.rq	Star	3	1	k
Query 9	query_1090.rq	Star	3	1	k
Query 10	query_1044.rq	Snowflake	3	2	$k + 35256$
Query 11	query_1870.rq	Path	2	1	k
Query 12	query_31.rq	Path	2	1	k

Tabla 3: Tabla de consultas para propuesta de solución.

Posterior a la evaluación, se obtiene los resultados de las iteraciones (anexo A .2). De ellos, para poder obtener un análisis, se utilizan medidas matemáticas para verificar la factibilidad de las respuestas. En este caso, se utiliza el error de la mediana (ecuación 32), por la naturaleza de Privacidad diferencial de entregar resultados aleatorios siguiendo la distribución de Laplace.

$$\text{MedError} = \text{mediana}\left(\left(\frac{(f(\mathcal{D}) - \mathcal{A}(\mathcal{D}))}{f(\mathcal{D})} * 100\right)_{i=1,10}\right) \tag{30}$$

Se puede observar los siguientes comportamientos de los resultados:

- Los resultados de consultas estilo Estrella entregan un error de mediana aceptable, el cual es menor al 10 %, lo que indica que los resultados son útiles para su

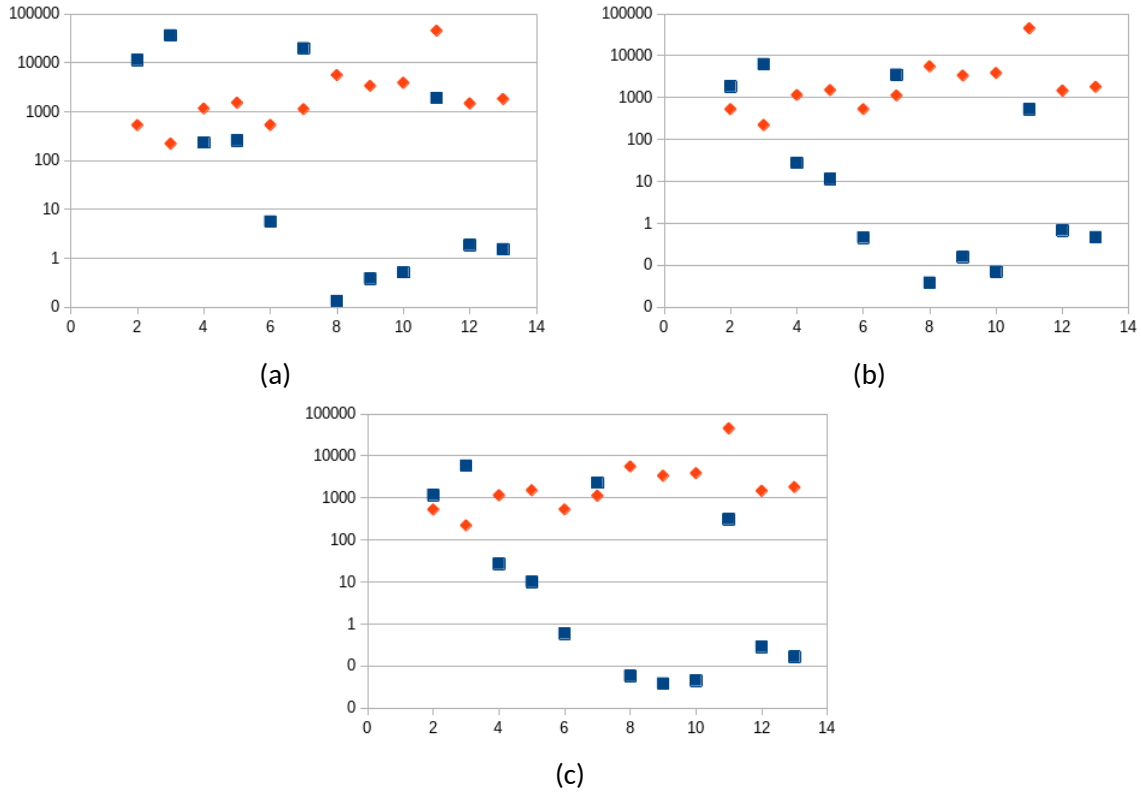


Figura 22: Resultados en forma gráfica de error de la mediana (□) en comparación al tamaño de *dataset* generado por *BGP* (◊). Esto para valores de ϵ equivalentes a $0{,}1$ (figura 22a), $0{,}4$ (figura 22b) y $0{,}8$ (figura 22c).

uso[Dwork y Roth, 2014].

- De las consultas estrellas, las consultas 7, 8 y 9 entregan una menor cantidad de error de mediana. Esto puede tener relación, con el tamaño del mapeo de las consultas.
- Otras dos consultas entregan errores de la mediana alrededor de un 250 % para $\epsilon = 0{,}1$. Sin embargo, para $\epsilon = \{0{,}4, 0{,}8\}$, el mismo valor disminuye en alrededor del 20 %, por lo que los resultados pueden tener una utilidad.
- Otros resultados, en especial las operaciones *JOIN* en *datasets*, entregan resultados que sobrepasan el porcentaje estimado (errores de mediana, mas allá del 100 %). Por aquel motivo, son resultados que no pueden ser utilizados y que requieren reconsiderar la estructura de la consulta.

Como se observa, efectivamente se cumple el primer objetivo de no entregar el resultado real para todas las consultas. Respecto al segundo objetivo de esta validación, se cumple de forma relativa a la estructura de la consulta.

5 .2.2. Factibilidad de la Solución

Como se observa en los resultados, en relación a la estructura de la consultas realizadas en la implementación, estos pueden presentar variaciones dispersas en los ruidos. En algunos casos una proporción normal, y en otros una gran diferencia. Esto depende en gran parte de la transformación realizada al *dataset* para obtener el conjunto que responde la consulta.

Consultas estilo estrella y caminos Para el primer caso de estrellas sin uniones, se observa que su ecuación de estabilidad es sencillamente k. Esto de debe al hecho que las consultas estrella son mapeos de la base de datos *RDF* consultada, donde en las uniones realizadas con otra información, no existe una repetición de la clave primaria (en este caso, la clave primaria es el sujeto de la consulta estrella).

Por lo tanto, como la tabla es formada con el añadido de información a través de los *BGP* pero no existen operaciones *JOIN* que afecten a la clave primaria, entonces la tabla cumple con no tener intersecciones, y su ecuación de estabilidad resultante es "k".

Otras observaciones, son para las consultas 7, 8 y 9, las cuales tienen el menor error de mediana de todas las evaluaciones realizadas (figura 22). Se observa que a comparación de otras consultas estrellas, el tamaño del mapeo generado es mayor. Esto tiene relación con que Privacidad diferencial funciona mejor para tamaños de *dataset* grandes [Dwork y Roth, 2014].

Consultas con *JOIN* (*Snowflakes*) Por otro lado, en el caso de operaciones *JOIN* entre tablas, ocurren dos casos que se observan en las ecuaciones de estabilidad (tabla 3) y los resultados (figura 22). Estos son en agregado de sensibilidad (el elemento que suma a "k" en el cálculo de sensibilidad). Primero, las consultas 1, 2, 6 y 10 tienen un añadido de distancia muy grande, mientras que por otro lado, para 3 y 4 no es muy agresivo.

En el caso de las consultas 3 y 4 para el valor de $\epsilon = 0{,}8$, se observa que el error de la mediana obtenido es un valor que se encuentra alrededor del 10 %, lo cual es mucho mayor que en el caso de consultas estrella, aun así estos valores son aceptables.

Esto se debe a la existencia de operadores *JOIN* entre llaves primarias sobre dos consultas estrellas. El no agregar ruido, puede producir vulneraciones dentro de la información que se está entregando al usuario, pues este entrega como argumento la unión de tablas (puede ingresar el *BGP*). Por ello, para el aseguramiento de privacidad tiene que agregarse este valor.

Por otro lado, para los valores más altos, se observa en las consultas que en su estructura tienen una cantidad de triples muy grande dentro de sus *BGP* (columna "Variables Join" en tabla 3). Esto quiere decir que el consultor está entregando muchos detalles dentro de la consulta que se está procesando. Se tiene conocimiento que esto puede entregar una gran cantidad de información, por ello su sensibilidad es muy grande, entregando una mayor cantidad de

ruido.

Sobre el análisis anterior, se puede observar que dependiendo de la cantidad de Variables de *JOIN* que tenga un grafo, aumenta de gran forma la sensibilidad de la información, la cual es representada a través de su expresión *BGP*.

5 .3. Cuota de privacidad

Por otro lado, se realiza un estudio sobre la cuota de privacidad para las consultas propuestas. Esto, realizando el siguiente procedimiento:

- Fijar $\epsilon = 0{,}1$ para las consultas.
- Realizar las cantidades de 10, 30, 55, 100, 1000 y 100000 consultas al Software y comparar estos con el resultado real, registrando en memoria la diferencia entre estos.
- De los resultados registrados, obtener el promedio de estos para cada cantidad de iteraciones.
- Comparar el promedio obtenido con el resultado real.

La idea de realizar este estudio, es comprobar la cantidad de consultas que tendría que realizar un usuario para poder encontrar el valor real de la consulta. Por otro lado, a partir de que cantidad de respuestas el resultado tiene similitud con el real, en especial para aquellas consultas que tienen una gran cantidad de ruido agregada. Los resultados obtenidos se muestran en figura 23, los cuales han sido separados respecto al grado de magnitud. Los valores obtenidos se presentan en anexo A .3.

De los resultados, se puede observar lo siguiente:

- Para 10 iteraciones, se obtiene un valor que puede tener similitud con la respuesta perturbada con Laplace. Pero a partir de 30, este valor obtiene una estabilidad.
- Se observa con claridad, que a mayor cantidad de iteraciones, la diferencia entre el resultado real con el promedio va decreciendo. Para consultas estrella, se acerca en gran medida al resultado real con diferencias insignificantes. Otras consultas con mayor ruido no se acercan lo suficiente (su magnitud sigue siendo una gran cantidad), pero si en gran medida en comparación a menos iteraciones.
- Para una consulta real, es absurdo realizar una cantidad de 100000 consultas para un usuario. Pero se puede demostrar por otro lado para consultas de *JOIN*, que todavía la privacidad se logra mantener, debido a que no se logra obtener el resultado real.

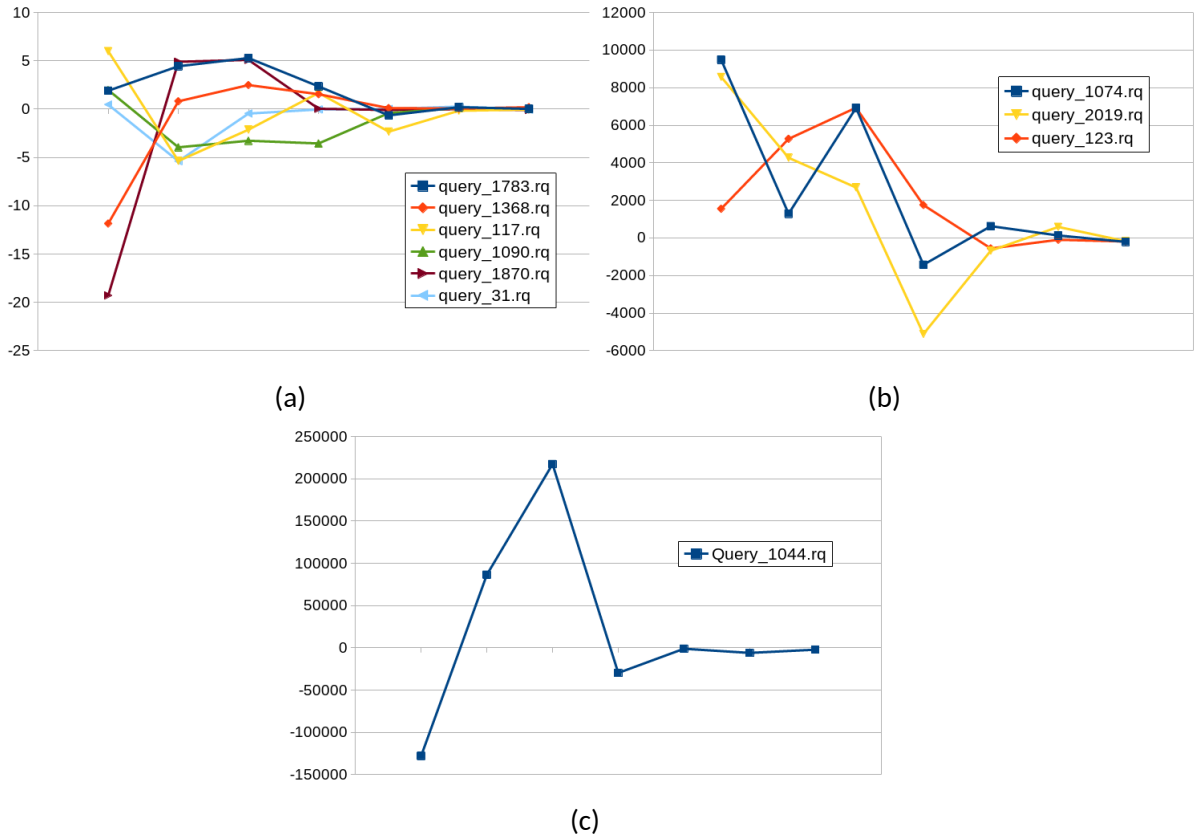


Figura 23: Representación gráfica de la diferencia entre resultado real de la consulta y el promedio de estas después de $10, 30, 55, 100, 10^3, 10^4$ y 10^5 consultas. Los gráficos se dividen por la sensibilidad de las consultas de estrellas (figura 23a) y consultas de *JOIN* (figura 23b y 23c), utilizando sensibilidades de tabla 3.

Este estudio, demuestra que es importante considerar la cuota de privacidad para las consultas realizadas. Esto se debe, aunque se entreguen garantías de privacidad ($\epsilon = 0{,}1$), si un usuario realizara una gran cantidad de consultas, el promedio de estas se puede acercar de gran forma al resultado real.

CAPÍTULO 6
CONCLUSIONES

6 .1. Cumplimiento de objetivos

El principal objetivo de este documento es el cumplimiento de los objetivos propuestos dentro de la definición del problema, en el cual existe un objetivo general, desglosado en objetivos secundarios:

6 .1.1. Aplicación de Privacidad diferencial tanto de forma teórica como práctica en bases de datos de grafo *RDF*, para aplicación de privacidad en información

Dentro de esta memoria, se ha investigado una posible implementación de este concepto de privacidad en bases de datos de Grafo *RDF*.

El concepto de Privacidad diferencial, busca asegurar la protección de la identidad de participantes de una base de datos. Esto en términos prácticos, la probabilidad de obtener un determinado resultado de una consulta sobre un *dataset*, sea lo más similar posible si a este mismo conjunto se le removiera un participante. El concepto anterior, hace que la participación de un individuo no influya en el resultado, debido a que su participación en el conjunto de datos no afecta al resultado de la consulta real.

Para la implementación de los conceptos anteriores en *RDF*, se requiere el estudio tanto de características estructurales como dirigidas por la información del lenguaje de consultas *SPARQL*. Las consultas realizadas, se rigen por la estructura de *Basic Graph Patterns* (*BGP*), los cuales se pueden visualizar como un grafo. Este tipo de consultas tienen tres formas principales, las cuales son consultas estrella, combinación de estrellas, y caminos. Las consultas estrella, replican el obtener los atributos de cierto elemento dentro de una consulta, siendo los atributos, los nodos conectados al sujeto principal. Mientras que las operaciones *JOIN* juntan sujetos con otros sujetos, resultando en consultas más complejas.

Junto a lo anterior, se utiliza el concepto de Esquema de Privacidad diferencial, el cual define la estructura que debiesen tener consultas SPARQL, que permitan entregar garantías de privacidad en los resultados.

6 .1.2. Implementar una solución aplicando Privacidad diferencial en *RDF*, la cual permita obtener resultados seguros ante ataques

La implementación de Privacidad diferencial sigue una secuencia. Primero, esta tiene que definirse de forma teórica para cumplir con las garantías que un cierto dato pueda mantenerse como privado para usos analíticos del *dataset*. Luego, estos conceptos matemáticos son implementados en un Software para poder ser puestos en práctica con consultas *SPARQL* reales, estudiando implementaciones tales como *FLEX*, las cuales entregan una referencia sobre la aplicación de los conceptos.

Este Software es desarrollado utilizando una arquitectura de Procesamiento Posterior (figura 18b), la cual consiste en un servidor para bases de datos *RDF Apache Jena/Fuseki* junto a un módulo intermediario en lenguaje *Python*. Este módulo es encargado de múltiples tareas, tales como recibir las consultas en lenguaje *SPARQL* del usuario, procesar estas consultas junto al servidor de bases de datos, aplicar los conceptos matemáticos de Privacidad diferencial a los resultados, y finalmente entregarlos al usuario.

Posteriormente, este desarrollo es puesto a prueba utilizando un *Dataset* generado desde una implementación para pruebas de stress sobre *RDF* denominado *WatDiv*. Se utiliza el *dataset* de 10M de triples junto a 12 consultas generadas, de las cuales 4 pertenecen a consultas estrella, 6 a consultas estrellas con uniones, y 2 a caminos (véase tabla 3). Al realizar las pruebas, se obtienen resultados que efectivamente son diferentes al original, los cuales difieren en cierto grado dependiendo de las características de la consulta realizada.

Estos resultados son obtenidos a través de la teoría estudiada dentro del objetivo anterior. Por lo tanto, estos resultados cumplen con implementar Privacidad diferencial. De esta forma, se asegura la protección de la información de los participantes del *dataset*.

6 .1.3. Determinar parámetros adecuados dentro de la implementación, con el objetivo de obtener resultados útiles para el usuario

El rol de administrador de datos es muy importante en la disposición de consultas analíticas dentro de la base de datos. Si este utilizara la implementación desarrollada en este trabajo, debe configurar ciertos parámetros. Estos entregarían diversos grados de ruido en los resultados de consultas, donde dependiendo de la cuota de privacidad que quisiera aplicar, puede seleccionar los parámetros de Privacidad diferencial: ϵ y δ.

Por otro lado, el administrador tiene que definir un Esquema de Privacidad diferencial para los usuarios. Para ello, puede identificar que estructura de consultas entregan resultados coherentes y cuales no los hacen, a través de un estudio similar al realizado en validación de la solución. De esta manera, se concluye que consultas estrella pueden entregar resultados privados, mientras que en el caso de transformaciones *JOIN*, tiene relación con el grado de Sensibilidad de la consulta realizada.

Con el cumplimiento de los objetivos anteriores, el objetivo principal de “Adaptar el uso de Privacidad diferencial a grafos, analizando componentes de distancia, para asegurar la privacidad de la información de usuarios” también lo hace, entregando un desarrollo de Privacidad diferencial para grafos *RDF* utilizando lenguaje de consultas *SPARQL*.

6 .2. Conclusiones sobre resultados

Los resultados obtenidos a través de la plataforma aseguran que la privacidad es aplicada en estos, protegiendo la participación de los usuarios dentro del conjunto de datos.

Dependiendo de la estructura de la consulta realizada, se aplica una cantidad de ruido particular al resultado. Para consultas estrellas, su grado de ruido de mantiene. En el caso de consultas de unión de tablas disjuntas, el ruido aplicado cambia de manera drástica. Esto debido a la operación *JOIN* de bases de datos, la cual al unir registros, dependiendo de la máxima frecuencia de la repetición de un elemento, resulta en un aumento de la sensibilidad total de la consulta. Por ello, se requiere aplicar mayores cantidades de ruido.

En relación a la utilidad de los resultados para los futuros usuarios, es muy relativo respecto a ciertos factores. En primer lugar, respecto a las consultas estrella, son las que entregan resultados más precisos, debido a que su grado de ruido es menor a un 10 %. Aun así, se tiene que tener precaución con la cuota de privacidad, debido a que al realizar una mayor cantidad de iteraciones, el promedio se acerca al valor real.

Segundo, las consultas de uniones son las que entregan más ruido, siendo debatible su uso final. En este tipo de consultas como se ha visto en la validación de la solución, se puede ajustar el grado de ϵ para poder acercarse a un error de la mediana de un 10 %, de esta forma haciendo los resultados utilizables.

Por otro lado, si eso no fuera posible o también se acerca a un valor de ϵ similar a 1 (las garantías de Privacidad son pobres), entonces es recomendable realizar una reconstrucción de la estructura de las consultas, las cuales logren disminuir el grado de la Sensibilidad total de la transformación por *JOIN*.

6 .3. Trabajo a Futuro

6 .3.1. Sobre este trabajo

Como se ha visto, se ha trabajado con Base de datos de Grafo *RDF*. Aun así, debido a que su naturaleza es obtener mapeos en forma de tablas, se pueden aplicar otros mecanismos de Privacidad diferencial para bases de datos de otros tipos (tales como relacionales, de clave

valor, objetos), pero con las adaptaciones realizadas para *RDF* como se ha mostrado en el marco teórico.

Por otro lado, respecto a la implementación realizada, queda como trabajo pendiente la investigación para otro tipo de consultas de agregación, tales como Suma, Promedio, entre otras. En el presente trabajo se ha estudiado para la consulta de tipo *COUNT*, donde se asegura que los resultados entregados cumplen con el método. Pero con lo realizado, no se puede asegurar lo mismo para otro tipo de consultas de agregación. Esto tendría como requisito el estudio del estado del arte para este tipo de consultas, junto a realizar un análisis matemático. Posteriormente, se debe relacionar estos conceptos con álgebra de conjuntos para bases de datos *SPARQL*, similar a lo señalado en el marco teórico en este documento.

Dentro del método realizado dentro de esta memoria, como trabajo a futuro quedaría estudiar como administrar la cuota de privacidad de la cual un usuario puede realizar consultas. En estos momentos, este concepto no está en la implementación, por lo que el usuario puede realizar la cantidad de consultas que desee. Lo anterior, al ser utilizado una cierta cantidad de veces, puede abrir la opción de realizar un ataque a través de procedimientos estadísticos para encontrar el valor real.

Otro concepto importante, es asegurar que un usuario solamente pueda cumplir con una determinada cuota de consultas. Dentro de las soluciones es tener un registro de estos usuarios (tales como una cuenta en una plataforma segura, su *IP*, entre otros) para poder limitar la cantidad de consultas que este puede realizar. O de otra forma, entregar el mismo resultado al realizar sus consultas. Estos dos enfoques, requieren de otro módulo dentro de la arquitectura que implemente estas operaciones.

Una mejora que podría realizarse para el rendimiento, es aplicar una arquitectura de software similar a *Chorus*. Como se ha descrito en el estado del arte, esta es la que entrega mejor consistencia y escalabilidad de los mencionados. Para su desarrollo, se requiere poder implementar la característica de reescritura de consultas. Se hace necesario una investigación más profunda sobre el lenguaje de consultas *SPARQL*, en especial sobre las operaciones matemáticas que este soporta. De esta manera, sería la misma consulta la que aplica los grados de ruido, y no un módulo externo.

6 .3.2. Investigaciones dentro del modelo

Como se ha descrito en el Documento, existen muchas formas para poder implementar una aplicación de Privacidad diferencial para base de datos de Grafo. En caso de *RDF*, debido a que la implementación de base de datos de Grafo se basa en realizar mapeos del *dataset* real, es que no puede aplicarse directamente otros métodos de privacidad realizados para el caso de grafos, como por ejemplo Sensibilidad restringida.

Por lo tanto, para profundizar el estudio de base de datos de grafo, este puede expandirse

hacia otro tipo de motores que trabajen directamente con el modelo de datos de grafo, o también en el caso de *RDF*, realizar adaptaciones a los resultados, las cuales permitan la manipulación de la información con el modelo de grafo. De esta forma permitiendo un estudio de Privacidad diferencial especifico para el paradigma de base de datos.

6 .3.3. Trabajo a futuro respecto al estado del arte

Dentro del estado del arte, se ha mencionado sobre los avances del campo de la Privacidad diferencial en general, en temáticas tales como implementaciones, publicación de información, usos, entre otros.

Se ha mencionado que dentro de las bases de datos existen dos tipos de consultas, las de agregación y de resultados en filas. El primer tipo responde preguntas tales como la cantidad de datos que cumplen con un patrón, o el promedio de un valor numérico entre todos los participantes. El segundo tipo, retorna las filas de la base de datos que cumplen con lo solicitado por el usuario.

Respecto a la publicación de información, la investigación para poder utilizar Privacidad diferencial en la respuesta del segundo tipo de consultas mencionado anteriormente que logren cumplir con medidas de privacidad, es todavía un gran desafío.

Esto es muy necesario, especialmente porque dos tercios de las consultas generalmente realizadas en un *dataset*, son de carácter de información pura. En campos como ciencias de datos e inteligencia artificial, esta información es muy requerida, debido a que se requiere trabajar con gran cantidad de volumen de información para poder mejorar los modelos de datos y predicciones. Actualmente, existen métodos que logran entregar *datasets*, pero estos son muy ineficientes en tiempos de cómputo.

ANEXOS

A .1. Consultas SPARQL realizadas

query_1044

```
SELECT (COUNT(?v0) as ?v0_count) WHERE { <http://db.uwaterloo.ca/~galuc
   /wsdbm/User17008> <http://schema.org/nationality> ?v2 . ?v0 <http://
   schema.org/eligibleRegion> ?v2 . ?v0 <http://schema.org/
   eligibleQuantity> ?v1 . }
```

query_1074.rq

```
SELECT (COUNT(?v0) as ?v0_count) WHERE { ?v0 <http://schema.org/
   eligibleRegion> <http://db.uwaterloo.ca/~galuc/wsdbm/Country1> . ?v0
    <http://purl.org/goodrelations/includes> ?v1 . ?v2 <http://purl.org
   /goodrelations/offers> ?v0 . ?v0 <http://purl.org/goodrelations/
   price> ?v3 . ?v0 <http://purl.org/goodrelations/serialNumber> ?v4 .
   ?v0 <http://purl.org/goodrelations/validFrom> ?v5 . ?v0 <http://purl
   .org/goodrelations/validThrough> ?v6 . ?v0 <http://schema.org/
   eligibleQuantity> ?v8 . ?v0 <http://schema.org/priceValidUntil> ?v10
    . ?v1 <http://schema.org/author> ?v7 . }
```

query_1090.rq

```
SELECT (COUNT(?v0) as ?v0_count) WHERE { ?v0 <http://www.w3.org
   /1999/02/22-rdf-syntax-ns#type> <http://db.uwaterloo.ca/~galuc/wsdbm
   /ProductCategory12> . ?v0 <http://ogp.me/ns#tag> ?v1 . ?v0 <http://
   schema.org/keywords> ?v3 . }
```

query_1115.rq

```
SELECT (COUNT(?v0) as ?v0_count) WHERE { ?v1 <http://ogp.me/ns#tag> <
   http://db.uwaterloo.ca/~galuc/wsdbm/Topic249> . ?v0 <http://purl.org
   /goodrelations/includes> ?v1 . ?v1 <http://schema.org/description> ?
   v6 . ?v0 <http://purl.org/goodrelations/serialNumber> ?v2 . ?v0 <
   http://purl.org/goodrelations/validFrom> ?v3 . ?v0 <http://purl.org/
   goodrelations/validThrough> ?v4 . ?v0 <http://schema.org/
   eligibleQuantity> ?v7 . ?v0 <http://schema.org/eligibleRegion> ?v8 .
    }
```

query_117.rq

```
SELECT (COUNT(?v0) as ?v0_count) WHERE { ?v0 <http://www.w3.org
   /1999/02/22-rdf-syntax-ns#type> <http://db.uwaterloo.ca/~galuc/wsdbm
```

```
/ProductCategory1> . ?v0 <http://ogp.me/ns#tag> ?v1 . ?v0 <http://
schema.org/keywords> ?v3 . }
```

query_123.rq

```
SELECT (COUNT(?v0) as ?v0_count) WHERE { ?v0 <http://www.w3.org
/1999/02/22-rdf-syntax-ns#type> <http://db.uwaterloo.ca/~galuc/wsdbm
/ProductCategory7> . ?v0 <http://ogp.me/ns#title> ?v1 . ?v0 <http://
schema.org/caption> ?v3 . ?v4 <http://db.uwaterloo.ca/~galuc/wsdbm/
likes> ?v0 . }
```

query_1368.rq

```
SELECT (COUNT(?v0) as ?v0_count) WHERE { ?v0 <http://www.w3.org
/1999/02/22-rdf-syntax-ns#type> <http://db.uwaterloo.ca/~galuc/wsdbm
/ProductCategory14> . ?v0 <http://ogp.me/ns#tag> ?v1 . ?v0 <http://
schema.org/keywords> ?v3 . }
```

query_1777.rq

```
SELECT (COUNT(?v0) as ?v0_count) WHERE { ?v1 <http://ogp.me/ns#tag> <
http://db.uwaterloo.ca/~galuc/wsdbm/Topic145> . ?v0 <http://purl.org
/goodrelations/includes> ?v1 . ?v1 <http://schema.org/description> ?
v6 . ?v0 <http://purl.org/goodrelations/serialNumber> ?v2 . ?v0 <
http://purl.org/goodrelations/validFrom> ?v3 . ?v0 <http://purl.org/
goodrelations/validThrough> ?v4 . ?v0 <http://schema.org/
eligibleQuantity> ?v7 . ?v0 <http://schema.org/eligibleRegion> ?v8 .
 }
```

query_1783.rq

```
SELECT (COUNT(?v0) as ?v0_count) WHERE { ?v0 <http://db.uwaterloo.ca/~
galuc/wsdbm/hasGenre> <http://db.uwaterloo.ca/~galuc/wsdbm/
SubGenre87> . ?v0 <http://xmlns.com/foaf/homepage> ?v1 . ?v0 <http
://ogp.me/ns#tag> ?v2 . ?v0 <http://schema.org/keywords> ?v3 . }
```

query_1870.rq

```
SELECT (COUNT(?v0) as ?v0_count) WHERE { ?v0 <http://db.uwaterloo.ca/~
galuc/wsdbm/hasGenre> <http://db.uwaterloo.ca/~galuc/wsdbm/
SubGenre55> . ?v0 <http://ogp.me/ns#title> ?v1 . }
```

query_2019.rq

```
SELECT (COUNT(?v0) as ?v0_count) WHERE { ?v0 <http://schema.org/
eligibleRegion> <http://db.uwaterloo.ca/~galuc/wsdbm/Country3> . ?v0
 <http://purl.org/goodrelations/includes> ?v1 . ?v2 <http://purl.org
```

```
/goodrelations/offers> ?v0 . ?v0 <http://purl.org/goodrelations/
price> ?v3 . ?v0 <http://purl.org/goodrelations/serialNumber> ?v4 .
?v0 <http://purl.org/goodrelations/validFrom> ?v5 . ?v0 <http://purl
.org/goodrelations/validThrough> ?v6 . ?v0 <http://schema.org/
eligibleQuantity> ?v8 . ?v0 <http://schema.org/priceValidUntil> ?v10
 . ?v1 <http://schema.org/author> ?v7 . }
```

query_31.rq

```
SELECT (COUNT(?v0) as ?v0_count) WHERE { ?v0 <http://db.uwaterloo.ca/~
   galuc/wsdbm/hasGenre> <http://db.uwaterloo.ca/~galuc/wsdbm/
   SubGenre78> . ?v0 <http://ogp.me/ns#title> ?v1 . }
```

A .2. Tabla de Resultados generales

Notación de columnas

- Resultado de consulta real : $f(q)$
- Resultado de consulta con ruido aplicado : $\mathcal{A}(q)$
- Expresión de Estabilidad Elástica : $\hat{S}_R^{(k)}$
- Distancia k de cálculo de Sensibilidad Elástica : k
- Tamaño del Dataset generado por la consulta q : n
- Máximo valor obtenido para Estabilidad Elástica: máx $\hat{S}$
- Error de la mediana : MedError

A .2.1. Resultados para $\epsilon = 0{,}1$

filename	$f(q)$	$\mathcal{A}(q)$	$\hat{S}_R^{(k)}$	k	n	máx $\hat{S}$	MedError
query_1368.rq	4656	4650.29	k	1	5602	1	0.1330
query_117.rq	2794	2786.15	k	1	3378	1	0.3823
query_2019.rq	48	-1742.69	k + 1188	1	222	1173	36375.3170
query_1870.rq	738	743.16	k	1	1476	1	1.8718
query_1783.rq	409	403.17	k	1	535	1	5.6239
query_1777.rq	442	25.16	k + 13	100	1161	47	232.9077
query_1115.rq	576	48.77	k + 13	108	1524	50	256.4421
query_1090.rq	3232	3243.57	k	1	3888	1	0.5153
query_1074.rq	190	1852.25	k + 1188	1	531	1176	11412.5878
query_123.rq	629	-10794.79	k + 4243	1	1130	4207	19852.7606
query_31.rq	909	896.75	k	1	1818	1	1.5290
query_1044.rq	22648	-405939.96	k + 35256	1	45297	35113	1892.3877

filename	$\mathcal{A}(q)1$	$\mathcal{A}(q)2$	$\mathcal{A}(q)3$	$\mathcal{A}(q)4$	$\mathcal{A}(q)5$
query_1368.rq	4650.10	4568.91	4650.49	4661.03	4666.89
query_117.rq	2748.79	2782.13	2807.31	2784.51	2802.33
query_2019.rq	-53610.68	4348.95	-2835.68	-649.70	-88140.11
query_1870.rq	740.08	752.52	846.30	751.11	703.15
query_1783.rq	407.26	400.25	367.96	406.08	385.89
query_1777.rq	-1502.03	1003.03	-2802.46	1532.13	536.41
query_1115.rq	-2093.36	-545.71	4357.43	-20.36	2748.90
query_1090.rq	3244.91	3335.88	3276.58	3204.37	3239.98
query_1074.rq	36373.28	40616.79	-35137.49	3906.30	-22863.10
query_123.rq	115491.82	-27799.23	186038.47	-175720.79	-85370.77
query_31.rq	905.71	935.05	886.18	898.18	895.33
query_1044.rq	-115606.28	296853.41	-1597554.35	-256898.04	-554981.88

filename	$\mathcal{A}(q)6$	$\mathcal{A}(q)7$	$\mathcal{A}(q)8$	$\mathcal{A}(q)9$	$\mathcal{A}(q)10$
query_1368.rq	4649.51	4658.46	4645.71	4631.44	4655.95
query_117.rq	2778.17	2791.84	2790.60	2787.79	2781.41
query_2019.rq	-6594.54	15223.36	43235.88	-56890.53	19792.95
query_1870.rq	767.07	743.81	742.51	714.29	739.85
query_1783.rq	441.83	434.44	386.10	409.25	370.18
query_1777.rq	14.41	1410.77	-2819.49	35.90	-879.46
query_1115.rq	30.83	-1110.84	2365.65	66.72	1843.38
query_1090.rq	3242.22	3251.07	3216.64	3239.50	3249.95
query_1074.rq	-201.80	-2803.19	17223.45	-20124.73	97596.70
query_123.rq	6209.65	155748.08	-94586.23	260530.89	-134255.90
query_31.rq	887.41	905.01	917.98	870.06	894.87
query_1044.rq	-653333.64	-749993.57	286430.22	91878.62	-1228294.17

A .2.2. Resultados para $\epsilon = 0{,}4$

filename	$f(q)$	$\mathcal{A}(q)$	$\hat{S}_R^{(k)}$	k	n	máx $\hat{S}$	MedError
query_1368.rq	4656	4656.61	k	1	5602	1	0.0382
query_117.rq	2794	2796.89	k	1	3378	1	0.1584
query_2019.rq	48	-1209.08	k + 1188	1	222	1170	6333.3708
query_1870.rq	738	737.00	k	1	1476	1	0.6790
query_1783.rq	409	409.96	k	1	535	1	0.4571
query_1777.rq	442	352.81	k + 13	90	1161	43	27.8419
query_1115.rq	576	582.45	k + 13	98	1524	46	11.3716
query_1090.rq	3232	3234.07	k	1	3888	1	0.0695
query_1074.rq	190	596.92	k + 1188	1	531	1175	1851.1477
query_123.rq	629	-5758.56	k + 4243	1	1130	4203	3536.8564
query_31.rq	909	910.43	k	1	1818	1	0.4649
query_1044.rq	22648	49779.17	k + 35256	1	45297	35106	527.8344

filename	$\mathcal{A}(q)1$	$\mathcal{A}(q)2$	$\mathcal{A}(q)3$	$\mathcal{A}(q)4$	$\mathcal{A}(q)5$
query_1368.rq	4657.50	4657.98	4656.93	4652.89	4648.30
query_117.rq	2791.91	2798.23	2802.70	2793.48	2803.01
query_2019.rq	-8297.85	-9266.78	-3928.69	1406.57	3415.14
query_1870.rq	728.11	725.11	737.04	762.71	745.52
query_1783.rq	409.44	408.89	415.20	419.71	408.34
query_1777.rq	319.43	279.38	386.18	200.23	565.56
query_1115.rq	574.57	497.94	762.09	127.34	634.97
query_1090.rq	3244.57	3232.49	3241.29	3234.98	3233.16
query_1074.rq	9942.19	5755.78	12427.40	-2793.72	2311.46
query_123.rq	25888.74	11154.07	-2447.08	-9070.03	-22150.30
query_31.rq	909.02	917.46	913.61	904.71	893.24
query_1044.rq	-135641.80	336694.95	-17195.42	139466.53	229231.88

filename	$\mathcal{A}(q)6$	$\mathcal{A}(q)7$	$\mathcal{A}(q)8$	$\mathcal{A}(q)9$	$\mathcal{A}(q)10$
query_1368.rq	4665.95	4654.42	4656.29	4655.13	4662.19
query_117.rq	2803.78	2793.97	2795.55	2798.81	2789.37
query_2019.rq	2760.90	-667.64	-15783.80	-1750.51	-396.62
query_1870.rq	738.88	736.96	735.49	754.70	736.88
query_1783.rq	411.25	410.49	412.62	408.67	404.56
query_1777.rq	122.07	556.73	86.86	495.26	524.36
query_1115.rq	556.54	1066.56	567.22	648.03	590.34
query_1090.rq	3241.12	3231.94	3230.48	3232.68	3237.87
query_1074.rq	603.31	-15123.12	590.53	19.78	-3860.64
query_123.rq	-29818.55	10652.87	-32111.76	22343.35	-31385.69
query_31.rq	915.93	913.17	911.84	906.63	906.98
query_1044.rq	9606.11	89952.23	-262840.62	144917.34	-92446.16

A .2.3. Resultados para $\epsilon = 0{,}8$

filename	$f(q)$	$\mathcal{A}(q)$	$\hat{S}_R^{(k)}$	k	n	máx $\hat{S}$	MedError
query_1368.rq	4656	4655.72	k	1	5602	1	0.0578
query_117.rq	2794	2794.25	k	1	3378	1	0.0382
query_2019.rq	48	-1714.17	k + 1188	1	222	1169	5862.8440
query_1870.rq	738	736.68	k	1	1476	1	0.2851
query_1783.rq	409	410.07	k	1	535	1	0.5916
query_1777.rq	442	451.80	k + 13	88	1161	42	27.1519
query_1115.rq	576	573.93	k + 13	96	1524	45	10.0210
query_1090.rq	3232	3232.98	k	1	3888	1	0.0451
query_1074.rq	190	-69.12	k + 1188	1	531	1174	1177.6482
query_123.rq	629	4768.39	k + 4243	1	1130	4202	2303.4863
query_31.rq	909	907.89	k	1	1818	1	0.1655
query_1044.rq	22648	-37358.46	k + 35256	1	45297	35105	312.3214

filename	$\mathcal{A}(q)1$	$\mathcal{A}(q)2$	$\mathcal{A}(q)3$	$\mathcal{A}(q)4$	$\mathcal{A}(q)5$
query_1368.rq	4652.03	4655.36	4654.83	4652.90	4656.90
query_117.rq	2792.28	2794.34	2794.42	2791.35	2796.15
query_2019.rq	298.78	-13719.31	2578.45	-3141.21	-3654.91
query_1870.rq	736.93	732.28	736.40	737.61	735.02
query_1783.rq	412.76	407.10	405.64	413.02	411.53
query_1777.rq	568.96	296.35	431.31	472.29	229.44
query_1115.rq	550.44	656.93	595.79	468.19	505.31
query_1090.rq	3232.75	3225.50	3234.51	3234.23	3232.85
query_1074.rq	-376.34	-500.26	-8520.49	238.09	-2246.21
query_123.rq	23576.77	25356.93	7421.97	2114.81	-33729.55
query_31.rq	910.10	908.57	907.07	907.12	913.48
query_1044.rq	-145976.72	77099.55	-50500.29	36178.56	46648.66

filename	$\mathcal{A}(q)6$	$\mathcal{A}(q)7$	$\mathcal{A}(q)8$	$\mathcal{A}(q)9$	$\mathcal{A}(q)10$
query_1368.rq	4660.12	4658.28	4660.18	4656.08	4649.76
query_117.rq	2794.12	2794.15	2791.28	2794.40	2797.61
query_2019.rq	-2994.40	2633.93	853.47	-433.93	-5760.32
query_1870.rq	735.39	736.43	747.53	738.22	743.95
query_1783.rq	408.84	411.31	408.38	407.46	413.52
query_1777.rq	568.91	328.89	525.41	223.56	500.84
query_1115.rq	588.48	559.37	693.60	531.25	1117.83
query_1090.rq	3233.70	3233.04	3230.09	3233.22	3232.92
query_1074.rq	-5382.87	6889.70	14202.65	2213.83	2228.85
query_123.rq	-1537.96	-15241.22	8398.88	13736.64	-17492.89
query_31.rq	909.19	908.26	907.47	907.52	903.52
query_1044.rq	90968.81	-60087.02	-24216.64	-109108.55	-106809.31

A .3. Tablas Diferencias entre resultado real y promedio

Notación de columnas

- Resultado de consulta real : $f(q)$
- Promedio de iteraciones en i iteraciones : $\bar{\mathcal{A}}(q)_i$
- Diferencia entre valores : $\Delta(f(q))$

query_1074.rq

	$f(q)$	$\bar{\mathcal{A}}(q)_{10}$	$\bar{\mathcal{A}}(q)_{30}$	$\bar{\mathcal{A}}(q)_{55}$	$\bar{\mathcal{A}}(q)_{100}$	$\bar{\mathcal{A}}(q)_{10^3}$	$\bar{\mathcal{A}}(q)_{10^4}$	$\bar{\mathcal{A}}(q)_{10^5}$
	190.00	9672.60	1476.59	7115.13	-1242.39	815.46	320.52	-21.96
$\Delta(f(q))$	0.00	9482.60	1286.59	6925.13	-1432.39	625.46	130.52	-211.96

query_2019.rq

	$f(q)$	$\bar{\mathcal{A}}(q)_{10}$	$\bar{\mathcal{A}}(q)_{30}$	$\bar{\mathcal{A}}(q)_{55}$	$\bar{\mathcal{A}}(q)_{100}$	$\bar{\mathcal{A}}(q)_{10^3}$	$\bar{\mathcal{A}}(q)_{10^4}$	$\bar{\mathcal{A}}(q)_{10^5}$
	48.00	8621.43	4314.92	2733.89	-5069.56	-629.13	632.77	-127.43
$\Delta(f(q))$	0.00	8573.43	4266.92	2685.89	-5117.56	-677.13	584.77	-175.43

query_1777.rq

	$f(q)$	$\bar{\mathcal{A}}(q)_{10}$	$\bar{\mathcal{A}}(q)_{30}$	$\bar{\mathcal{A}}(q)_{55}$	$\bar{\mathcal{A}}(q)_{100}$	$\bar{\mathcal{A}}(q)_{10^3}$	$\bar{\mathcal{A}}(q)_{10^4}$	$\bar{\mathcal{A}}(q)_{10^5}$
	442.00	262.24	378.88	502.99	175.03	433.38	419.54	437.92
$\Delta(f(q))$	0.00	-179.76	-63.12	60.99	-266.97	-8.62	-22.46	-4.08

query_1115.rq

	$f(q)$	$\bar{\mathcal{A}}(q)_{10}$	$\bar{\mathcal{A}}(q)_{30}$	$\bar{\mathcal{A}}(q)_{55}$	$\bar{\mathcal{A}}(q)_{100}$	$\bar{\mathcal{A}}(q)_{10^3}$	$\bar{\mathcal{A}}(q)_{10^4}$	$\bar{\mathcal{A}}(q)_{10^5}$
	576.00	556.87	798.22	769.04	540.33	550.02	551.37	575.99
$\Delta(f(q))$	0.00	-19.13	222.22	193.04	-35.67	-25.98	-24.63	-0.01

query_1783.rq

	$f(q)$	$\bar{\mathcal{A}}(q)_{10}$	$\bar{\mathcal{A}}(q)_{30}$	$\bar{\mathcal{A}}(q)_{55}$	$\bar{\mathcal{A}}(q)_{100}$	$\bar{\mathcal{A}}(q)_{10^3}$	$\bar{\mathcal{A}}(q)_{10^4}$	$\bar{\mathcal{A}}(q)_{10^5}$
	409.00	410.91	413.44	414.29	411.35	408.36	409.20	409.03
$\Delta(f(q))$	0.00	1.91	4.44	5.29	2.35	-0.64	0.20	0.03

query_123.rq

	$f(q)$	$\bar{\mathcal{A}}(q)_{10}$	$\bar{\mathcal{A}}(q)_{30}$	$\bar{\mathcal{A}}(q)_{55}$	$\bar{\mathcal{A}}(q)_{100}$	$\bar{\mathcal{A}}(q)_{10^3}$	$\bar{\mathcal{A}}(q)_{10^4}$	$\bar{\mathcal{A}}(q)_{10^5}$
	629.00	2178.41	5903.78	7564.09	2378.81	69.68	529.83	436.73
$\Delta(f(q))$	0.00	1549.41	5274.78	6935.09	1749.81	-559.32	-99.17	-192.27

query_1368.rq

	$f(q)$	$\bar{\mathcal{A}}(q)_{10}$	$\bar{\mathcal{A}}(q)_{30}$	$\bar{\mathcal{A}}(q)_{55}$	$\bar{\mathcal{A}}(q)_{100}$	$\bar{\mathcal{A}}(q)_{10^3}$	$\bar{\mathcal{A}}(q)_{10^4}$	$\bar{\mathcal{A}}(q)_{10^5}$
	4656.00	4644.15	4656.81	4658.49	4657.55	4656.11	4656.06	4656.20
$\Delta(f(q))$	0.00	-11.85	0.81	2.49	1.55	0.11	0.06	0.20

query_117.rq

	$f(q)$	$\bar{\mathcal{A}}(q)_{10}$	$\bar{\mathcal{A}}(q)_{30}$	$\bar{\mathcal{A}}(q)_{55}$	$\bar{\mathcal{A}}(q)_{100}$	$\bar{\mathcal{A}}(q)_{10^3}$	$\bar{\mathcal{A}}(q)_{10^4}$	$\bar{\mathcal{A}}(q)_{10^5}$
	2794.00	2800.00	2788.64	2791.88	2795.68	2791.67	2793.84	2793.96
$\Delta(f(q))$	0.00	6.00	-5.36	-2.12	1.68	-2.33	-0.16	-0.04

query_1090.rq

	$f(q)$	$\bar{\mathcal{A}}(q)_{10}$	$\bar{\mathcal{A}}(q)_{30}$	$\bar{\mathcal{A}}(q)_{55}$	$\bar{\mathcal{A}}(q)_{100}$	$\bar{\mathcal{A}}(q)_{10^3}$	$\bar{\mathcal{A}}(q)_{10^4}$	$\bar{\mathcal{A}}(q)_{10^5}$
	3232.00	3233.98	3228.05	3228.73	3228.45	3231.59	3232.09	3232.14
$\Delta(f(q))$	0.00	1.98	-3.95	-3.27	-3.55	-0.41	0.09	0.14

query_1044.rq

	$f(q)$	$\bar{\mathcal{A}}(q)_{10}$	$\bar{\mathcal{A}}(q)_{30}$	$\bar{\mathcal{A}}(q)_{55}$	$\bar{\mathcal{A}}(q)_{100}$	$\bar{\mathcal{A}}(q)_{10^3}$	$\bar{\mathcal{A}}(q)_{10^4}$	$\bar{\mathcal{A}}(q)_{10^5}$
	22648.00	-105347.29	109131.79	240022.24	-7080.72	21542.11	16645.83	20461.17
$\Delta(f(q))$	0.00	-127995.29	86483.79	217374.24	-29728.72	-1105.89	-6002.17	-2186.83

query_1870.rq

	$f(q)$	$\bar{\mathcal{A}}(q)_{10}$	$\bar{\mathcal{A}}(q)_{30}$	$\bar{\mathcal{A}}(q)_{55}$	$\bar{\mathcal{A}}(q)_{100}$	$\bar{\mathcal{A}}(q)_{10^3}$	$\bar{\mathcal{A}}(q)_{10^4}$	$\bar{\mathcal{A}}(q)_{10^5}$
	738.00	718.70	742.91	743.10	738.04	737.92	738.00	737.91
$\Delta(f(q))$	0.00	-19.30	4.91	5.10	0.04	-0.08	0.00	-0.09

query_31.rq

	$f(q)$	$\bar{\mathcal{A}}(q)_{10}$	$\bar{\mathcal{A}}(q)_{30}$	$\bar{\mathcal{A}}(q)_{55}$	$\bar{\mathcal{A}}(q)_{100}$	$\bar{\mathcal{A}}(q)_{10^3}$	$\bar{\mathcal{A}}(q)_{10^4}$	$\bar{\mathcal{A}}(q)_{10^5}$
	909.00	909.47	903.61	908.54	908.97	909.02	909.27	908.94
$\Delta(f(q))$	0.00	0.47	-5.39	-0.46	-0.03	0.02	0.27	-0.06

REFERENCIAS BIBLIOGRÁFICAS

[DBL, 2011] (2011). *20th USENIX Security Symposium, San Francisco, CA, USA, August 8-12, 2011, Proceedings*. USENIX Association.

[Aluç *et al.*, 2014] Aluç, G., Hartig, O., Özsu, M. T., y Daudjee, K. (2014). Diversified stress testing of rdf data management systems. En *Proceedings of the 13th International Semantic Web Conference - Part I*, ISWC '14, pp. 197–212, New York, NY, USA. Springer-Verlag New York, Inc.

[Blocki *et al.*, 2013] Blocki, J., Blum, A., Datta, A., y Sheffet, O. (2013). Differentially private data analysis of social networks via restricted sensitivity. En *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS '13, pp. 87–96, New York, NY, USA. ACM.

[Blum *et al.*, 2008] Blum, A., Ligett, K., y Roth, A. (2008). A learning theory approach to non-interactive database privacy. En *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pp. 609–618, New York, NY, USA. ACM.

[Buil, 2019] Buil, C. (2019). Differential privacy and sparql. unpublished article.

[C. Barth-Jones, 2012] C. Barth-Jones, D. (2012). The 're-identification' of governor william weld's medical information: A critical re-examination of health data identification risks and privacy protections, then and now. *SSRN Electronic Journal*.

[Chan *et al.*, 2011] Chan, T.-H. H., Shi, E., y Song, D. (2011). Private and continual release of statistics. *ACM Trans. Inf. Syst. Secur.*, 14(3):26:1–26:24.

[Chen y Zhou, 2013] Chen, S. y Zhou, S. (2013). Recursive mechanism: Towards node differential privacy and unrestricted joins. En *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pp. 653–664, New York, NY, USA. ACM.

[Codd, 1972] Codd, E. F. (1972). Relational completeness of data base sublanguages. En *Database Systems*, pp. 65–98. Prentice-Hall.

[Cyganiak *et al.*, 2014] Cyganiak, R., Hyland-Wood, D., y Lanthaler, M. (2014). Rdf 1.1 concepts and abstract syntax. *W3C Proposed Recommendation*.

[Dinur y Nissim, 2003] Dinur, I. y Nissim, K. (2003). Revealing information while preserving privacy. En [Neven *et al.*, 2003], pp. 202–210.

[Dwork, 2011] Dwork, C. (2011). A firm foundation for private data analysis. *Commun. ACM*, 54(1):86–95.

[Dwork *et al.*, 2006] Dwork, C., McSherry, F., Nissim, K., y Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. En Halevi, S. y Rabin, T., editores, *Theory of Cryptography*, pp. 265–284, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Dwork *et al.*, 2010] Dwork, C., Naor, M., Pitassi, T., y Rothblum, G. N. (2010). Differential privacy under continual observation. En *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC '10, pp. 715–724, New York, NY, USA. ACM.

[Dwork y Roth, 2014] Dwork, C. y Roth, A. (2014). The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407.

[Engineering, 2015] Engineering, L. J. (2015). Attribute-based access control with a graph database. `https://leanjavaengineering.wordpress.com/2015/04/13/attribute-based-access-control-with-a-graph-database/`. Consulta: 2018-03-07.

[Fernández *et al.*, 2013] Fernández, J. D., Martínez-Prieto, M. A., Gutiérrez, C., Polleres, A., y Arias, M. (2013). Binary rdf representation for publication and exchange (hdt). *Web Semantics: Science, Services and Agents on the World Wide Web*, 19:22–41.

[Fletcher, 2008] Fletcher, G. H. L. (2008). An algebra for basic graph patterns. En *In Proc. of the Workshop on Logic in Databases (LID*.

[Haeberlen *et al.*, 2011] Haeberlen, A., Pierce, B. C., y Narayan, A. (2011). Differential privacy under fire. En [DBL, 2011].

[Harris y Seaborne (eds), 2010] Harris, S. y Seaborne (eds), A. (2010). SPARQL 1.1 query language. Working draft, W3C.

[Hay *et al.*, 2009] Hay, M., Li, C., Miklau, G., y Jensen, D. D. (2009). Accurate estimation of the degree distribution of private networks. En [Wang *et al.*, 2009], pp. 169–178.

[Huang *et al.*, 2015] Huang, D., Han, S., Li, X., y Yu, P. S. (2015). Orthogonal mechanism for answering batch queries with differential privacy. En *Proceedings of the 27th International Conference on Scientific and Statistical Database Management*, SSDBM '15, pp. 24:1–24:10, New York, NY, USA. ACM.

[Johnson *et al.*, 2018a] Johnson, N. M., Near, J. P., Hellerstein, J. M., y Song, D. (2018a). Chorus: Differential privacy via query rewriting. *CoRR*, abs/1809.07750.

[Johnson *et al.*, 2018b] Johnson, N. M., Near, J. P., y Song, D. (2018b). Towards practical differential privacy for SQL queries. *PVLDB*, 11(5):526–539.

[Jorgensen *et al.*, 2016] Jorgensen, Z., Yu, T., y Cormode, G. (2016). Publishing attributed social graphs with formal privacy guarantees. En [Özcan *et al.*, 2016], pp. 107–122.

[Kasiviswanathan *et al.*, 2008] Kasiviswanathan, S. P., Lee, H. K., Nissim, K., Raskhodnikova, S., y Smith, A. D. (2008). What can we learn privately? *CoRR*, abs/0803.0924.

[Kellaris y Papadopoulos, 2013] Kellaris, G. y Papadopoulos, S. (2013). Practical differential privacy via grouping and smoothing. En *Proceedings of the 39th international conference on Very Large Data Bases*, PVLDB'13, pp. 301–312. VLDB Endowment.

[Kellaris *et al.*, 2014] Kellaris, G., Papadopoulos, S., Xiao, X., y Papadias, D. (2014). Differentially private event sequences over infinite streams. *Proc. VLDB Endow.*, 7(12):1155–1166.

[Kifer y Machanavajjhala, 2011] Kifer, D. y Machanavajjhala, A. (2011). No free lunch in data privacy. En [Sellis *et al.*, 2011], pp. 193–204.

[Lee *et al.*, 2015] Lee, J., Wang, Y., y Kifer, D. (2015). Maximum likelihood postprocessing for differential privacy under consistency constraints. En *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pp. 635–644, New York, NY, USA. ACM.

[Li *et al.*, 2014] Li, C., Hay, M., Miklau, G., y Wang, Y. (2014). A data- and workload-aware algorithm for range queries under differential privacy. *Proc. VLDB Endow.*, 7(5):341–352.

[Lin y Kifer, 2013] Lin, B.-R. y Kifer, D. (2013). Information preservation in statistical privacy and bayesian estimation of unattributed histograms. En *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pp. 677–688, New York, NY, USA. ACM.

[Martínez-Prieto *et al.*, 2012] Martínez-Prieto, M. A., Arias, M., y Fernández, J. D. (2012). Exchange and consumption of huge rdf data. En *The Semantic Web: Research and Applications*, pp. 437–452. Springer.

[McSherry, 2009] McSherry, F. D. (2009). Privacy integrated queries: An extensible platform for privacy-preserving data analysis. En *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, pp. 19–30, New York, NY, USA. ACM.

[Mohammed *et al.*, 2011] Mohammed, N., Chen, R., Fung, B. C., y Yu, P. S. (2011). Differentially private data release for data mining. En *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pp. 493–501, New York, NY, USA. ACM.

[Narayan y Haeberlen, 2012] Narayan, A. y Haeberlen, A. (2012). Djoin: Differentially private join queries over distributed databases. En *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation*, OSDI'12, pp. 149–162, Berkeley, CA, USA. USENIX Association.

[Narayanan y Shmatikov, 2008] Narayanan, A. y Shmatikov, V. (2008). Robust de-anonymization of large sparse datasets. En *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, SP '08, pp. 111–125, Washington, DC, USA. IEEE Computer Society.

[Neven *et al.*, 2003] Neven, F., Beeri, C., y Milo, T., editores (2003). *Proceedings of the Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 9-12, 2003, San Diego, CA, USA*. ACM.

[Nissim *et al.*, 2007] Nissim, K., Raskhodnikova, S., y Smith, A. (2007). Smooth sensitivity and sampling in private data analysis. En *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pp. 75–84, New York, NY, USA. ACM.

[Özcan *et al.*, 2016] Özcan, F., Koutrika, G., y Madden, S., editores (2016). *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*. ACM.

[Patil *et al.*, 2014] Patil, S., Vaswani, G., y Bhatia, A. (2014). Graph databases - an overview. pp. 657–660.

[Pérez *et al.*, 2009] Pérez, J., Arenas, M., y Gutierrez, C. (2009). Semantics and complexity of sparql. *ACM Trans. Database Syst.*, 34(3):16:1–16:45.

[Polikoff, 2014] Polikoff, I. (2014). Comparing sparql with sql. `https://www.topquadrant.com/2014/05/05/comparing-sparql-with-sql/`. Consulta: 2018-03-07.

[Proserpio *et al.*, 2014] Proserpio, D., Goldberg, S., y McSherry, F. (2014). Calibrating data to sensitivity in private data analysis: A platform for differentially-private analysis of weighted datasets. *Proc. VLDB Endow.*, 7(8):637–648.

[Qardaji *et al.*, 2013] Qardaji, W., Yang, W., y Li, N. (2013). Understanding hierarchical methods for differentially private histograms. *Proc. VLDB Endow.*, 6(14):1954–1965.

[Sellis *et al.*, 2011] Sellis, T. K., Miller, R. J., Kementsietsidis, A., y Velegrakis, Y., editores (2011). *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12-16, 2011*. ACM.

[Uber, 2018] Uber (2018). sql-differential-privacy. `https://github.com/uber/sql-differential-privacy`. Consulta: 2018-03-07.

[Wang *et al.*, 2009] Wang, W., Kargupta, H., Ranka, S., Yu, P. S., y Wu, X., editores (2009). *ICDM 2009, The Ninth IEEE International Conference on Data Mining, Miami, Florida, USA, 6-9 December 2009*. IEEE Computer Society.

[Xiao *et al.*, 2011a] Xiao, X., Bender, G., Hay, M., y Gehrke, J. (2011a). ireduct: Differential privacy with reduced relative errors. En *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pp. 229–240, New York, NY, USA. ACM.

[Xiao *et al.*, 2011b] Xiao, X., Wang, G., y Gehrke, J. (2011b). Differential privacy via wavelet transforms. *IEEE Trans. on Knowl. and Data Eng.*, 23(8):1200–1214.

[Xu *et al.*, 2013] Xu, J., Zhang, Z., Xiao, X., Yang, Y., Yu, G., y Winslett, M. (2013). Differentially private histogram publication. *The VLDB Journal*, 22(6):797–822.

[Yuan *et al.*, 2015] Yuan, G., Zhang, Z., Winslett, M., Xiao, X., Yang, Y., y Hao, Z. (2015). Optimizing batch linear queries under exact and approximate differential privacy. *ACM Trans. Database Syst.*, 40(2):11:1–11:47.

[Zhang *et al.*, 2015] Zhang, J., Cormode, G., Procopiuc, C. M., Srivastava, D., y Xiao, X. (2015). Private release of graph statistics using ladder functions. En *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pp. 731–745, New York, NY, USA. ACM.

[Zhang *et al.*, 2016] Zhang, J., Xiao, X., y Xie, X. (2016). Privtree: A differentially private algorithm for hierarchical decompositions. En *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, pp. 155–170, New York, NY, USA. ACM.

[Zhu *et al.*, 2017] Zhu, T., Li, G., Zhou, W., y Yu, P. S. (2017). Differentially private data publishing and analysis: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 29(8):1619–1638.