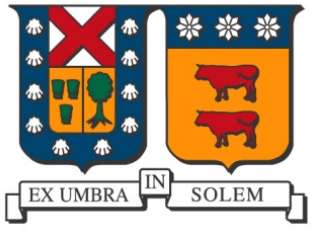
2018

ANÁLISIS DE TECNOLOGÍAS DE ANONIMIZACIÓN Y SEGURIDAD PARA LA MONETIZACIÓN DE DATOS PRIVADO

GALLEGUILLOS BARRIOS, CRISTÓBAL ANDRÉS

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



“ANÁLISIS DE TECNOLOGÍAS DE ANONIMIZACIÓN Y SEGURIDAD PARA LA MONETIZACIÓN DE DATOS PRIVADOS”

CRISTÓBAL ANDRÉS GALLEGUILLOS BARRIOS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Dr. Carlos Buil Aranda
Profesor Correferente: Dr. Horst Von Brand

Noviembre - 2018

DEDICATORIA

A mi familia, por su apoyo y cariño incondicional.

AGRADECIMIENTOS

Muchas gracias a los profesores Carlos Buil y Horst Von Brand por apoyarme y guiarme en todo el proceso de la realización de esta memoria.

A mi familia que siempre estuvo ahí apoyándome a través de todo el proceso de mi educación, motivándome a seguir adelante y a quienes agradezco por la persona en la que me he convertido.

A mis amigos de la universidad que me acogieron e hicieron que la experiencia de estudiar en otra ciudad haya sido inolvidable.

A mis amigos del colegio con los que he podido contar toda la vida.

RESUMEN

Resumen— La información personal es un bien sumamente cotizado, pero para poder analizar y trabajar con esta información es necesario tratar los datos previamente para vulnerar la privacidad de los usuarios.

En esta memoria se realizará un análisis e implementación de una solución para entregar privacidad y anonimización a la información personal que es intercambiada y analizada tanto por instituciones públicas como privadas.
La solución aplicará el método de privacidad diferencial propuesto para bases de datos SQL y se extenderá al modelo de datos RDF, el cual será consultado mediante el lenguaje SPARQL.

Palabras Clave— Anonimización - Privacidad - Información personal - RDF - Privacidad diferencial

ABSTRACT

***Abstract*—** Personal information is a highly valued asset, but in order to be able to analyze and work with this information is necessary to previously treat the data to not breach the privacy of the users.

In this undergraduate thesis an analysis and implementation of a solution will be carried out to provide privacy and anonymization to the personal information that is exchanged and analyzed by both public and private institutions.
The solution will apply the differential privacy method proposed for SQL databases and it will be extended to the RDF data model which will be queried using the language SPARQL.

***Keywords*—** Anonimization - Privacy - Personal information - RDF - Differential Privacy

GLOSARIO

ACID: *Atomicity*, *Consistency*, *Isolation*, *Durability*. Axioma fundamental de las bases de datos SQL.

BASE: *Basically*, *Available*, *Soft state Eventually Consistent*. Axioma fundamental de las bases de datos NoSQL.

bloque-$q^\star$: Es el conjunto de tuplas en T^* cuyos atributos se generalizan a $q^\star$

DB: *Data base* (o en español Base de datos). Colección organizada de datos, almacenados computacionalmente.

EQUIJOIN: Un *EQUIJOIN* es un JOIN que se realiza utilizando la igualdad de las *JOIN key* para poder asociar las tablas involucradas.

GDB: *Graph Data Base* (o base de datos de grafos en español). Base de datos que organiza sus datos como arcos y nodos.

HDT: *Header*, *Dictionary*, *Triples*. Estructura de datos compacta y un formato de serialización binario para RDF.

IRI: *Internationalized Resource Identifier*. Extensión de la URI que acepta caracteres Unicode.

JOIN: En SQL un JOIN es la combinación de columnas de dos o más tablas.

JOIN Key: Es el parámetro que se ocupa para vincular las tablas participantes en un JOIN.

ML: *Machine learning*. Campo de la inteligencia artificial que, utilizando técnicas estadísticas sobre los datos, brinda la «habilidad» a los sistemas computaciones de aprender a partir de ellos, sin ser necesario programarlo explícitamente.

Open Source: Open source o Código abierto. Software cuyo código fuente y otros derechos que normalmente son exclusivos para quienes poseen los derechos de autor, son publicados bajo una licencia de código abierto o forman parte del dominio público.

QI:Un conjunto de atributos $\mathrm{QI_T} = \{\mathrm{A_i}, \ldots, \mathrm{A_j}\}$ de una tabla T son llamados *cuasi-identificadores* si pueden ser vinculados con datos externos para identificar al menos a un individuo dentro de la base de datos.

RDF: *Resource Description Framework*. Estándar de codificación de recursos web, recomendado por la W3C, que organiza los datos como grafos.

SQL: *Structured Query Language*. Lenguaje de consultas de base de datos relacionales, que permite la definición, manipulación y creación de éstas.

Tupla: Fila o registro de una base de datos que representa un objeto único de datos implícitamente estructurados en una tabla.

URI: *Uniform Resource Identifier*. Identificador ASCII para referenciar un recurso web de forma unívoca.

URL: *Uniform Resource Locator*. Especificación de la URI, usada generalmente para referen-

ciar páginas web.

W3C: *World Wide Web Consortium*. Consorcio internacional que crea recomendaciones y estándares para el crecimiento y la estabilidad de la *World Wide Web*.

XML: *Extensible Markup Language*. Es un lenguaje de marcado que define un conjunto de reglas para codificar documentos en un formato que es tanto leíble por humanos como maquinas.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE CÓDIGOS

INTRODUCCIÓN

En la era digital que vivimos actualmente, la información personal es percibida como el petróleo o la moneda del mundo digital. Tanto en el sector privado como en el público utilizan esta información para estudios y negocios. Al presentarse una oferta y demanda de ella, comienza un mercado de información personal. Sin embargo el acceso a los datos que la conforman es restringido debido a los problemas de privacidad.

La privacidad no sólo consiste en ocultar cosas. Se trata de autonomía, integridad y autorregulación. El derecho a la privacidad no es el que asiste a las personas para poder cerrar sus puertas o bajar las cortinas porque, quizás, quieran realizar alguna actividad ilegal. Es el derecho de las personas a controlar que detalles de su vida quedan en la intimidad de sus hogares y cuáles de ellos pueden filtrarse al exterior.

Para poder atacar este problema y proteger la información sensible, usualmente es necesario sanitizar los datos antes de que éstos puedan ser distribuidos y analizados en los estudios y negocios. Realizar dicha sanitización requiere lo que se denomina «anonimización de datos». También conocida como enmascaramiento u ofuscación de datos, la anonimización de datos es el proceso de reemplazar contenidos de campos identificables en una base de datos para que los registros no puedan ser asociados con un individuo, proyecto, compañía o entidad, en específico.

Hoy en día las bases de datos son sumamente necesarias. La cantidad de información que es manejada por toda clase de instituciones es cuantiosa y precisa de las bases de datos para poder almacenar, consultar, analizar y relacionarlos. Existen múltiples modelos y sistemas de ellas. Los primeros modelos fueron las bases de datos relacionales, pero éstas pueden resultar muy rígidas en casos donde se requiere una mayor escalabilidad del sistema, en que resulta muy complejo almacenar una gran cantidad de datos, resultando en una pérdida de eficiencia al momento de realizar consultas.

Es por lo anterior, que se crean las bases de datos no relacionales, las que permiten una mayor flexibilidad de la estructura del sistema. Dentro de ellas existen las bases de datos de grafos. Éstas almacenan sus datos como un conjunto de nodos y arcos dirigidos para poder representar las relaciones entre ellos como lo hace RDF (Resource Description Framework, un tipo de bases de datos de grafos). Para la formulación de consultas en este modelo no existe un lenguaje formal para la realización de éstas, pero el más utilizado y que cumple con los estándares definidos para RDF, es SPARQL.

En esta memoria se implementa un método de anonimización conocido como privacidad diferencial para el modelo de datos RDF, teniendo en cuenta una futura aplicación monetaria.

CAPÍTULO 1
DEFINICIÓN DEL PROBLEMA

1.1. Privacidad

Actualmente, vivimos en una era donde la información constituye uno de los bienes más preciados, tanto para las personas como para las instituciones. Por ello, existe un especial interés en la seguridad, precisión y accesibilidad de dicha información, especialmente la que refiere acerca de uno mismo. La percepción de que no se puede controlar nuestra información personal es clave para nuestro concepto de la intimidad que toda persona posee. A medida de que el mundo va evolucionando la necesidad de proteger la privacidad personal se torna en uno de los derechos fundamentales inherentes a la persona. La tecnología, por sí misma, no es la que vulnera la privacidad de las personas, sino que son ellas mismas y las políticas que siguen las que crean estas violaciones de privacidad. [3]

En ese sentido, privacidad puede ser definida como:

> «El derecho de ser dejado solo o libre de interferencia o intrusión. Privacidad de la información es el derecho a tener algún control respecto a como nuestra información personal es recolectada y usada.»

1.1.1. ¿Por que es importante?

Debido a la gran velocidad de la innovación tecnológica, la privacidad de la información se ha ido complejizando cada vez más, en la medida que aumenta el volumen de datos que están siendo recolectados e intercambiados. A medida que la tecnología se vuelve más sofisticada y por tanto, más invasiva, también lo hacen los usos de esa información, provocando que las organizaciones tengan que lidiar con matrices de riesgo increíblemente complejas, a fin de asegurar que la información personal esté debidamente protegida.

1.1.2. ¿Privacidad y seguridad son lo mismo?

Son conceptos relacionados pero diferentes. La privacidad de datos se enfoca en el uso y gobernanza de la información personal como, por ejemplo, crear políticas para asegurar que la información personal de los clientes sea recolectada, compartida y usada de la manera apropiada.
Por su parte, la seguridad de datos se enfoca más en la protección de ellos frente a ataques maliciosos y explotación con fines de lucro de datos robados.
En definitiva, son diferentes, como se puede observar, pero ambos son necesarios para asegurar los datos e información de las personas. [16]

1.2. Mercado de la información personal

Como se mencionó anteriormente, la información personal es percibida como un bien de mucho valor en el mundo digital. Estos datos son extraordinariamente valiosos para el sector público y privado para poder mejorar sus productos y servicios.
Al ver oportunidades comerciales en la oferta y demanda de información nace la noción de mercado de la información personal. La información personal puede ser intercambiada en forma de *e-commerce* donde ocurren compras, ventas y transacciones financieras.

1.2.1. Anonimización de datos

Para no violar la privacidad de los dueños de la información personal que esta siendo intercambiada es necesario tratar los datos primero.

La anonimización de datos personales consiste en delimitar y suprimir aquella información concreta que permite la identificación de individuos o instituciones en específico dentro de una base de datos, con el objetivo de eliminar, de forma irreversible, las posibilidades de identificación de éstos, evitando que se pueda determinar a quien pertenecen los datos al momento de utilizarlos.

Un ejemplo es cuando el departamento de marketing de un banco necesita realizar análisis para determinar cuales son los productos que más se venden y cuales no. Para llevar a cabo esta tarea se analizan los datos de los clientes del banco, pero en estos datos hay información personal (nombre, rut, dirección, número de teléfono), éstos son considerados información personal, ya que permiten la identificación de la persona. Para poder realizar el análisis estadístico es necesario anonimizar los datos, un posible método para lograr ésto es ocultar la información personal, haciendo imposible identificar a los sujetos dueños de los datos.

1.2.2. Intercambio de información

Al momento de realizar el intercambio de información lo que se entrega a las entidades que buscan información es una versión con ruido del análisis estadístico sobre los datos. Éste es un resultado agregado de una consulta derivada de la información personal de los usuarios con un ruido aleatorio incluido para garantizar la privacidad de los dueños de estos. La magnitud del ruido que se agrega a los datos impacta directamente al precio de la consulta y a la cantidad de perdida de privacidad de los dueños de los datos, por lo que un mayor precio de la consulta implica una menor inyección de ruido a los datos, lo que permite estudios más precisos.

Existen tres entidades involucradas en el mercado de la información personal

- *Data owners*: Dueños de los datos. Son las personas dueñas de la información que se

esta intercambiando.

- *Data seekers*: Compradores o buscadores de datos. Entidad que paga una cierta cantidad de dinero para obtener los datos estadísticos con ruido.
- *Market maker*: Creador del mercado. Es una entidad confiable que se encarga de obtener los datos de los *data owners*, realizar las consultas y agregarles ruido y de calcular el precio para los *data seekers* y compensación para los *data owners*.

[22]

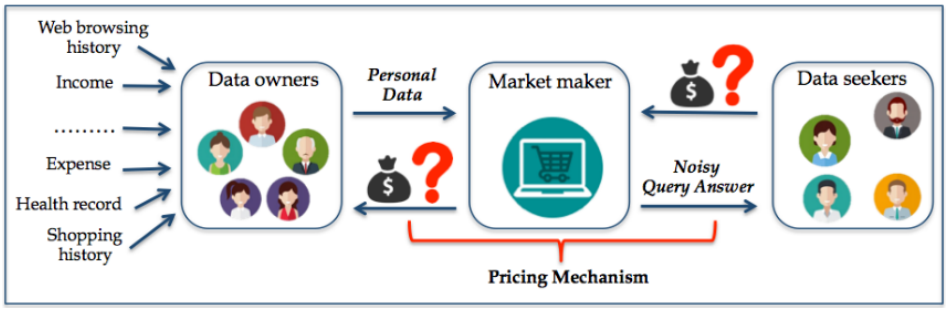


Figura 1: Cómo funciona el mercado de la información personal [22]

1.3. Bases de datos

Como se señaló anteriormente, el rápido crecimiento de la tecnología genera un gran problema en el almacenamiento y obtención eficiente de datos. Una cantidad increíble de transacciones en linea, estudios y análisis resultan en la generación de cantidades masivas de ellos que deben ser bien guardados y organizados.
En ese sentido, las bases de datos juegan un rol importantísimo para poder satisfacer la necesidad de almacenar y extraer datos de manera organizada.

Los dos tipos de bases de datos más utilizados son las relacionales y no relacionales.
Las bases de datos relacionales, también conocidas como *Relational Database Management Systems* o bases de datos basadas en el lenguaje de consultas SQL son las más utilizadas en la actualidad, siendo las más populares *SQLite*, *Microsoft SQL Server*, *Oracle Database*, *MySQL* e *IBM DB2*. Estas bases de datos se basan en relaciones en forma de tablas las que están compuestas por columnas, las que representan un campo y filas que contienen un registro.

Dentro de los axiomas fundamentales de las bases de datos SQL se encuentra *ACID* [8, 24]

- **Atomicity**: Atomicidad. Consiste en que al realizar actualizaciones a la base de datos, deben terminarse completamente o ser abortadas.

- **Consistency**: Consistencia. La integridad de la base de datos debe ser mantenida por todas las transacciones. La base de datos nunca debe estar en un estado inconsistente, una vez que la transacción esté completa.
- **Isolation**: Aislamiento. Cada transacción debe operar como si estuviera aislada para que no exista ningún tipo de conflicto entre múltiples transacciones.
- **Durability**: Durabilidad. Si el sistema falla de alguna manera, se debe recuperar todas las transacciones que han sido realizadas. Ninguna actualización a la base de datos debería perderse.

Sin embargo, las bases de datos relacionales presentan un conjunto de limitaciones, debido al crecimiento constante de los datos guardados y analizados. Las restricciones en la escalabilidad y almacenamiento, perdida de eficiencia al momento de realizar consultas y el manejo de volúmenes de datos cada vez más grandes resulta en una tarea sumamente compleja.

Como consecuencia de lo anterior, nacen las bases de datos no relacionales o también conocidas como bases de datos NoSQL. Una de sus principales características es que no tienen una estructura esquemática fija, de manera que los registros pueden tener diferentes campos según los requerimientos. Ello, es conocido como un esquema dinámico.

El axioma fundamental de las bases de datos NoSQL es *BASE* [27], lo cual es un opuesto a ACID.

- **Basically Available**: Consiste en que el sistema garantice la disponibilidad de los datos, de manera que siempre habrá una respuesta a una solicitud.
- **Soft State**: Significa que el estado del sistema puede cambiar a través del tiempo, aún durante los momentos en donde esté sin interacciones, por lo que dicho estado siempre es «blando».
- **Eventually consistent**: El sistema, eventualmente, se volverá consistente una vez que deje de recibir entrada de datos. Los datos se propagarán a donde corresponde, tarde o temprano. El sistema continuará recibiendo entrada de datos y no va a revisar la consistencia de cada transacción antes de avanzar a la siguiente. Esto quiere decir que al momento de realizar una consulta el resultado de ésta puede no ser el último estado de la base de datos.

Existen distintos tipos de bases de datos NoSQL, en esta memoria se trabajará con bases de datos de grafos o también conocidas como bases de datos de grafos.

1.3.1. Bases de datos de grafos

GDB o base de datos de grafos es un tipo de base de datos NoSQL diseñada para tratar las relaciones entre los datos con la misma importancia que los mismos datos. Como todas las

bases de datos NoSQL, las GDBs no tienen un modelo predefinido. Mientras otras bases de datos computan relaciones durante el tiempo de consulta con operaciones JOIN, las que son muy costosas, una GDB guarda las conexiones al igual que los datos en el modelo. [21]

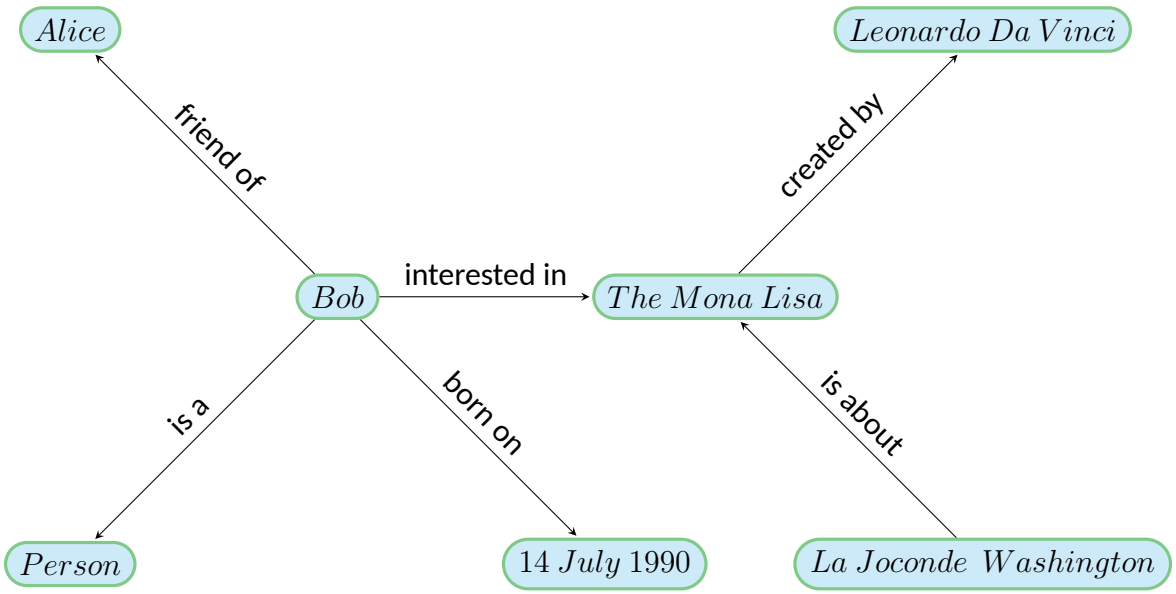


Figura 2: Ejemplo base de datos de grafo

1.4. Objetivos

Considerando lo expuesto anteriormente, el principal objetivo de este trabajo es la implementación del método privacidad diferencial en el modelo de datos RDF. Para ello, es necesario revisar distintas técnicas utilizadas en la actualidad para poder proveer de privacidad a los datos.

1.4.1. Objetivos específicos

1. Analizar técnicas para la seguridad y privacidad de los datos.
2. Estudiar en detalle privacidad diferencial.
3. Probar el uso de privacidad diferencial en bases de datos SQL.
4. Implementar un sistema para la aplicación de privacidad diferencial para un modelo de datos en RDF.
5. Validar el funcionamiento del software, sometiéndolo múltiples pruebas.

CAPÍTULO 2
MARCO CONCEPTUAL

Se comenzará por analizar múltiples técnicas y métodos para asegurar la anonimización de los datos privados como análisis previo al método que se utilizará en esta memoria. Luego se analizará el contexto en el que se aplicará la solución, esto es el uso de un modelo de datos RDF con consultas en SPARQL.

2.1. K-Anonymity

Definición 2.1.1 *Los* Cuasi-identificadores *QI son un conjunto de atributos* $\text{QI}_\text{T} = \{\text{A}_\text{i}, ..., \text{A}_\text{j}\}$ *de una tabla* T *que pueden ser vinculados con datos externos para identificar, al menos, a un individuo dentro de la base de datos.*

Al momento de querer proteger información que se libera al público se deben combinar los datos liberadas con datos disponibles externamente y analizar los posibles ataques. Esta tarea es sumamente difícil para la entidad que desea publicar información, por lo que para protegerla se presenta el concepto *K-Anonymity*, el cual busca atacar este problema satisfaciendo algunas restricciones con los datos publicados.

K-Anonymity es una propiedad de los datos anonimizados en donde se busca que los individuos correspondientes a los datos no puedan ser reidentificados.
Sea $\text{T}(A_1, ..., A_n)$ una tabla y QI_T el cuasi-identificador asociado a ella. Se puede decir que T satisface *k-anonymity* si para cada tupla $t \in \text{T}$ existen otras $k-1$ tuplas $\text{t}_1, \text{t}_2, ..., \text{t}_{k-1} \in \text{T}$ tal que $\text{t}[\text{QI}_\text{T}] = \text{t}_1[\text{QI}_\text{T}] = \text{t}_2[\text{QI}_\text{T}] = ... = \text{t}_{k-1}[\text{QI}_\text{T}]$ para todo cuasi-identificador QI_T [29].

Zipcode	Age	Sex	Disease
12211	18	M	Arthritis
12244	19	M	Cold
12245	27	M	Heart problem
12377	27	M	Flu
12377	27	F	Arthritis
12391	34	F	Diabetes
12391	45	F	Flu

Microdata table *T*

Zipcode	Age	Sex	Disease
122**	18–19	M	Arthritis
122**	18–19	M	Cold
*	27	*	Heart problem
*	27	*	Flu
*	27	*	Arthritis
12391	≥ 30	F	Diabetes
12391	≥ 30	F	Flu

2-anomymous table *T**

Figura 3: Ejemplo de k-anonymity [11]

En la figura 3 podemos observar un ejemplo de una tabla T que fue llevada a *2-Anonymity*, el cuasi-identificador para la tabla es $QI_T = \{\text{Zipcode, Age, Sex}\}$ y $k = 2$. Por lo tanto, para cada una de las tuplas contenidas en la tabla T, los valores de la tupla que conforman el cuasi-identificador aparecen al menos dos veces en T. Esto quiere decir que $\text{t1}[\text{QI}_\text{T}] = \text{t2}[\text{QI}_\text{T}]$,

$t3[QI_T] = t4[QI_T] = t5[QI_T]$ y $t6[QI_T] = t7[QI_T]$.

Esta solución protege a la tabla T contra intentos de identificación mediante la vinculación de los datos publicados con fuentes externas conocidas.

2.2. Ataques a *K-Anonymity*

Aún tomando precauciones al identificar a los cuasi-identificadores, *k-anonymity* es vulnerable a ciertos tipos de ataques, como por ejemplo:

2.2.1. Ataque de homogeneidad

K-Anonymity puede crear grupos que revelen información debido a la falta de diversidad en los atributos sensibles.

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	130XX	<30	X	Heart Disease
2	130XX	<30	X	Heart Disease
3	130XX	<30	X	Viral Infection
4	130XX	<30	X	Viral Infection
5	1485X	≥ 40	X	Cancer
6	1485X	≥ 40	X	Heart Disease
7	1485X	≥ 40	X	Viral Infection
8	1485X	≥ 40	X	Viral Infection
9	130XX	3X	X	Cancer
10	130XX	3X	X	Cancer
11	130XX	3X	X	Cancer
12	130XX	3X	X	Cancer

Tabla 1: Tabla en 4 Anonymity [19]

En la tabla anterior se puede observar que es posible obtener la información sensible de un individuo que se sabe que tiene aproximadamente 30 años y *Zip Code* 13053, debido a la falta de diversidad de los datos sensibles.

2.2.2. Ataque con conocimiento de fondo

K-Anonymity no provee protección en contra de los ataques con conocimiento de fondo. Volviendo a la misma tabla anterior, al saber que el individuo tiene *Zip Code* 13053 y tiene 21 años se puede determinar que la persona tiene una enfermedad cardiaca o una infección viral. Además si se tiene conocimiento de fondo como por ejemplo que la persona tiene

un historial familiar extenso de enfermedades cardiacas se podría determinar con un cierto grado de certeza que el individuo tiene una enfermedad cardíaca.

2.3. ℓ-*diversity*

El principio ℓ-*diversity* nace para poder enfrentar las debilidades de *K-Anonymity*, ya que éste si toma en cuenta los casos en que el adversario posee conocimientos de fondo, brindando una mayor protección a la privacidad.

Definición 2.3.1 *Un bloque-$q^\star$ es el conjunto de tuplas en T^* cuyos atributos se generalizan a $q^\star$.*

Un bloque-$q^\star$ es ℓ-diverso si contiene al menos ℓ valores bien representados para el atributo sensible S. Una tabla es ℓ-diversa si todos los bloques-$q^\star$ son ℓ-diversos.
El principio de ℓ-*diversity* busca asegurar ℓ valores bien representados para los atributos sensibles en cada bloque-$q^\star$. Esto quiere decir que al menos $1/\ell$ de las tuplas posean el valor sensible más frecuente.

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	1305X	≤ 40	X	Heart Disease
4	1305X	≤ 40	X	Viral Infection
9	1305X	≤ 40	X	Cancer
10	1305X	≤ 40	X	Cancer
5	1485X	>40	X	Cancer
6	1485X	>40	X	Heart Disease
7	1485X	>40	X	Viral Infection
8	1485X	>40	X	Viral Infection
2	1306X	≤ 40	X	Heart Disease
3	1306X	≤ 40	X	Viral Infection
11	1306X	≤ 40	X	Cancer
12	1306X	≤ 40	X	Cancer

Tabla 2: Tabla en 3-Diversity [19]

Como se puede observar la tabla anterior para cada bloque-$q^\star$ existe una diversidad de tres datos y se cumple que al menos un $50\,\%$ de las tuplas tienen el mismo valor para $Condition$. Por lo tanto, a pesar de que el adversario sepa información de antemano, no podrá determinar que enfermedad posee la persona debido a la diversidad.[19, 31]

2.4. Limitaciones de ℓ-*diversity*

- Puede que no tome en cuenta los ataques de inferencia probabilística.

    - Ejemplo: Se tienen 10 tuplas, en la área de enfermedad que sería el dato sensible una es «Cancér», una es «Enfermedad cardíaca» y las ocho restantes son «Gripe». Esto cumple con *3-diversity*, pero el atacante puede aún afirmar que la enfermedad de la persona que esta buscando es «Gripe» con una precisión de un $80\,\%$.

- ℓ*-diversity* puede ser innecesario.
    - *2-diversity* es innecesario para una tabla que contiene solamente datos negativos.
- ℓ*-diversity* es difícil de lograr.
- No toma en cuenta los ataques basados en algoritmos.
    - Los algoritmos intentan utilizar una cantidad de generalización para obtener ℓ*-diversity*.
    - El adversario puede hacer usar características de esos algoritmos para hacer «ingeniería inversa» en las tablas generalizadas.
- ℓ*-diversity* no considera el significado semántico de los datos sensibles.

Código Postal	Edad	Salario	Enfermedad
476**	2*	20k	Ulcera Gástrica
476**	2*	30k	Gastritis
476**	2*	40k	Cáncer Estomacal
4790*	$\geq$40	50k	Gastritis
4790*	$\geq$40	100k	Gripe
4790*	$\geq$40	70k	Bronquitis
476**	3*	60k	Bronquitis
476**	3*	80k	Neumonía
476**	3*	90k	Cáncer Estomacal

Tabla 3: Ejemplo tabla 3-*diverse* vulnerable [18]

Suponiendo que el adversario tiene como datos previos que el código postal de Bob es 47678 y que su edad es 27 se pueden sacar las siguientes conclusiones:

- El salario de Bob esta en [20k,40k], lo cual es relativamente bajo.
- Bob tiene una enfermedad relacionada con el estomago.

[18, 32]

2.5. Privacidad Diferencial

Privacidad diferencial describe una promesa hecha por parte de la persona que maneja los datos al sujeto al que le pertenecen: tú no seras afectado adversamente o de otra manera al permitir que tus datos sean usados en cualquier estudio, no importando que otros estudios, conjuntos de datos o información de otras fuentes se encuentre disponible. [5]

Privacidad diferencial es una técnica que permite un análisis estadístico de los datos mientras se protegen los datos respecto a los individuos con una fuerte garantía formal de privacidad, a diferencia de los métodos anteriormente mencionados, éste si toma en cuenta los ataques basados en algoritmos y parte de la base de que solo se debería publicar información que no dependa, en gran medida, de ningún individuo en particular.

Como supuesto se tienen dos conjuntos de datos D y D', tal que D' se pueda obtener cambiando solo una tupla en D.

- Sea Q la consulta privatizada (con ruido).
- A es un valor muy cercano a 1.
- El resultado de la ejecución de la consulta es R.
    - Si este valor es muy grande, no se provee de ninguna privacidad a los datos. Si $A = 1$ los datos no tienen utilidad.
    - Se define $A = e^{\epsilon}$, siendo $\epsilon > 0$

$$\frac{Pr(Q(D) = R)}{Pr(Q(D') = R)} \leq A \quad \forall R$$

La probabilidad de que una consulta entregue un resultado en particular será similar para D y D'.

Esto significa que existen dos mundos posibles:

1. Mundo posible donde los datos de un sujeto en particular están incluidos en D. $Pr(R) = a$
2. Mundo posible donde los datos de un sujeto en particular no están incluidos en D. $Pr(R) = b$

Por lo que $a \simeq b$, de esta manera teniendo un resultado R no se podría saber cual de los dos mundos produjo R. [32, 15]

Sea D un conjunto de datos que contiene el registro médico de cada individuo en Australia y se quiere publicar acerca de los pacientes con diabetes, la manera correcta de hacerlo es publicando el número total de pacientes con diabetes y no el listado de pacientes ya que, de

esta manera, se esta cumpliendo con el principio anteriormente enunciado que señala que no se debe publicar datos que dependan de un individuo en particular. Ahora si se supone que la cantidad de pacientes con diabetes es 1000, para poder publicar el número de pacientes se debe agregar antes «ruido», ya que si no se hace no se cumpliría con el principio de privacidad diferencial.

Sin embargo, existe un gran «pero». Efectivamente, la cantidad de filtrado de información al realizar una consulta es un valor muy pequeño, pero no es cero. Mientras más consultas se realizan mayor filtrado de los datos se produce y no se puede volver atrás. Esto es uno de los mayores desafíos de privacidad diferencial que se manifiesta en dos maneras:

1. Mientras más información se quiere preguntar a la base de datos, más ruido tiene que inyectarse para poder minimizar el filtrado de privacidad. Esto significa que privacidad diferencial puede ser un gran problema al momento de querer entrenar modelos complejos de *machine learning*.

2. Una vez que los datos han sido filtrados no hay nada más que hacer. Ocurrido que se han filtrado datos hasta dónde los cálculos indican que es seguro no se puede continuar sin arriesgar la privacidad de los usuarios

La cantidad total de filtrado es conocida generalmente como «presupuesto de privacidad», el que determina cuántas consultas se permiten y cuan precisos serán los resultados. El principal problema de la privacidad diferencial es determinar el presupuesto de privacidad, ya que si el valor es muy alto se filtraran los datos sensibles, en cambio, si el valor es muy pequeño los resultados obtenidos dejaran de ser útiles. [14]
Al momento de querer realizar venta de información existe una correlación entre el presupuesto de privacidad y el costo de la consulta. [22]

Un ejemplo de la importancia al momento de escoger el valor del presupuesto de privacidad es el estudio [10]. El autor comienza con una base de datos pública para vincular la dosis de un medicamento llamado *Warfarin* con marcadores genéticos específicos. Luego usa *machine learning* para desarrollar un modelo de dosificado basado en su base de datos, pero aplicando privacidad diferencial y probando con distintos presupuestos.
Al momento de verificar la efectividad para proteger el filtrado de datos y el éxito del modelo para tratar a los pacientes (simulados) los resultados mostraron que la precisión del modelo dependía, en gran parte, del presupuesto de privacidad con el que había sido entrenado. Si el presupuesto era muy alto, la base de datos filtraba una gran cantidad de datos sensibles respecto a los pacientes, pero resultaba en que el modelo tomaba decisiones del dosificación del medicamento adecuadas. En el caso contrario, cuando el presupuesto era muy bajo, se obtenía privacidad pero el modelo tenía la tendencia a matar a los pacientes.

2.6. Conceptos para lograr privacidad diferencial

En esta sección se revisarán los distintos conceptos que son necesarios para poder entender y lograr de manera adecuada privacidad diferencial.

Como se pudo observar en la sección anterior, privacidad diferencial entrega una garantía formal para la protección de privacidad: el resultado obtenido al utilizar este método no entrega mucha información respecto a cual de las dos bases de datos vecinas fue utilizada en el cálculo del resultado.

Formalmente, privacidad diferencial considera una base de datos como un vector $x \in D^n$. La *distancia* entre dos bases de datos x, y pertenecientes al conjunto de datos D^n es $d(x, y) = |\{i|x_i \neq y_i\}|$, por lo tanto una distancia k de la verdadera base de datos es cuando a lo más k tuplas son cambiadas. Dos bases de datos son vecinas si $d(x, y) = 1$.

La *sensibilidad* de una consulta corresponde a la cantidad que puede puede variar el resultado de ésta cuando la base de datos cambia.

Una de las medidas de sensibilidad es la **sensibilidad global**[17] que corresponde a la máxima diferencia en el resultado de la consulta en cualquier combinación posible entre dos bases de datos vecinas. Para una función $f : D^n \to \mathbb{R}^d$ y todo $x, y \in D^n$, la sensibilidad global de f es:

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

Aquí se asume que f toma valores en $\mathbb{R}^d$, y se utiliza la norma L_1 en $\mathbb{R}^d$ (denotada como $||\cdot||_1$ o simplemente como $||\cdot||$) como una métrica de distancia para los resultados de f.[23]

Una transformación T es c - estable si para cualquier conjunto de datos de entrada A y B, la diferencia simétrica de la transformación sobre los conjuntos es menor o igual a una constante c por la diferencia simétrica de conjuntos.

$$|T(A) \ominus T(B)| \leq c \times |A \ominus B|$$

Las transformaciones con estabilidad acotada propagan las garantías de privacidad diferencial de sus salidas de vuelta a sus entradas, escalada por su constante de estabilidad. [20]

Estabilidad Global. Una transformación $T : D^n \to D^n$ es c - estable si para $x, y \in D^n$ tal que $d(x, y) = 1$, $d(T(x), T(y)) \leq c$.

Otra definición de sensibilidad es la **sensibilidad local**, que consiste en la máxima diferencia entre el resultado de la consulta en la verdadera base de datos y cualquier vecino de ésta. Para $f : D^n \to \mathbb{R}^d$ y $x \in D^n$ la *sensibilidad local* de f en x es

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

Estabilidad local. Una transformación $T : D^n \rightarrow D^n$ es localmente c - estable para la verdadera base de datos x si para $y \in D^n$, tal que $d(x, y) = 1$, $d(T(x), T(y)) \leq c$.

Debido a que la sensibilidad local esta basada en la base de datos verdadera, debe ser usada cuidadosamente para evitar filtrar información acerca de los datos. Por lo tanto, se utilizarán **funciones de suavizado** para que al momento de recurrir a la sensibilidad local para aplicar privacidad diferencial se tome en consideración la distancia k de la base de datos verdadera (cuando son cambiadas a lo más k tuplas) [17].

Al momento de obtener el resultado de la consulta es necesario agregarle ruido, para esto se utiliza una distribución de Laplace.

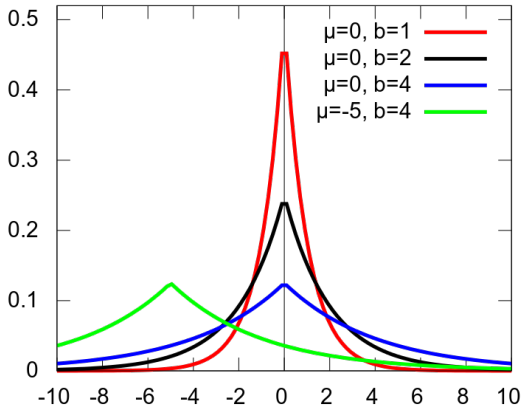


Figura 4: Distribución de Laplace

La ecuación para el ruido es:

$$f(x|\mu, b) = \frac{1}{2b} e^{-\frac{|x-\mu|}{b}}$$

Ahora para poder adaptarla a los parámetros de privacidad diferencial:

$$b = \frac{GS_f}{\epsilon} \qquad \mu = F(D^n)$$

Siendo $F(D^n)$ el resultado sobre la base de datos verdadera.

$$f(x|\mu, b) = \frac{\epsilon}{2GS_f} e^{-\frac{|x-F(D^n)|\epsilon}{GS_f}}$$

[15]
De está manera al tener todas las bases de datos vecinas a la real de donde podría haber venido el resultado se obtiene lo siguiente:

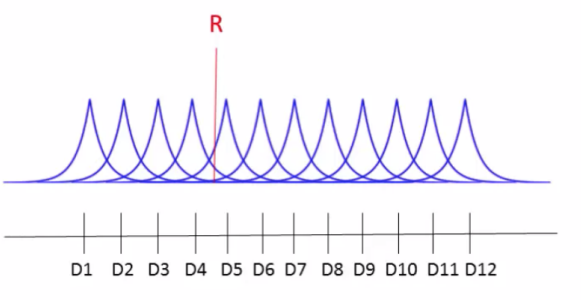


Figura 5: Distribuciones de Laplace de todas las posibles bases de datos [15]

Al generar el ruido con la distribución de Laplace no se puede saber a cual de todas las bases de datos pertenece el resultado.
Mientras más sensibilidad tenga la función y mientras ϵ (Presupuesto de privacidad) sea más pequeño sera más difícil determinar a que base de datos pertenece el resultado:

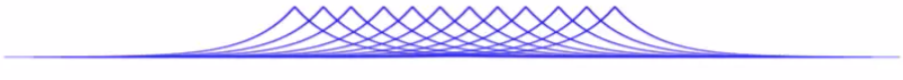


Figura 6: Base de datos con mayor sensibilidad o un menor presupuesto de privacidad [15]

2.7. Resource Description Framework

Resource Description Framework o **RDF** es un marco propuesto por la W3C para expresar hechos o relaciones entre objetos e información en la Web. Éste fue creado con la intención de cumplir con los siguientes objetivos:

- Tener un modelo simple de datos.
- Tener una semántica formal e inferencia demostrable.
- Utilizar un vocabulario extensible basado en URI.
- Usar una sintaxis basada en XML.
- Permitiendo el uso de tipos de datos basados en esquemas XML.
- Permitiendo a cualquiera a realizar declaraciones acerca de cualquier recurso.

RDF utiliza los siguientes conceptos claves:

2.7.1. Modelo de datos en grafos

La estructura de cualquier expresión en RDF es una colección de *triples*, cada uno consiste de un sujeto, un predicado y un objeto. Un conjunto de estos triples es llamado *Grafo RDF*.

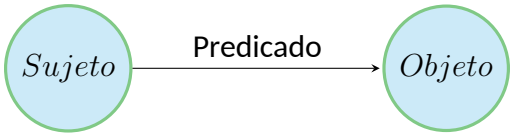


Figura 7: Ejemplo triple

Cada triple representa una declaración de una relación entre las cosas denotadas por los nodos a los que vincula. Los grafos RDF son grafos dirigidos, ya que la dirección siempre apunta al objeto.

2.7.2. Vocabulario basado en URI

Un recurso puede estar definido de tres maneras:[30]

- **URI o IRI**: Un *Internationalized Resource Identifier* o *Uniform Resource Identifier* es un string Unicode absoluto que sigue el formato especificado en [4]. Son identificadores que sirven para señalar de forma unívoca recursos en la web. Las IRIs son la generalización de las URIs y las URLs, que sirve para la identificación de páginas web.
- **Literal**: Son strings Unicode cualquieras para referenciar cadenas de caracteres, números o fechas.
- **Nodo blanco**: Es un nodo que no es una URI ni un literal. Se puede utilizar como agrupador de otros recursos.

Un ejemplo de URI sería:

`http://www.w3.org/2006/vcard/ns#title`

Con la cual se definirá de manera unívoca la propiedad título.
Ahora se completara el triple con otra URI y un literal para mostrar un triple:

Sujeto(URI/Nodo Blanco) + Predicado(URI) + Objeto(URI/Literal/Nodo Blanco)

Por lo tanto, un ejemplo de triple sería:

- Sujeto : `http://www.snee.com/hr/emp3`
- Predicado : `http://www.w3.org/2006/vcard/ns#title`

- Objeto : “Vice President”

Esta es la manera como se definen relaciones en un modelo RDF, en este caso que *emp3* tiene por título Vicepresidente. Luego, los triples se conectan entre ellos formando un grafo.

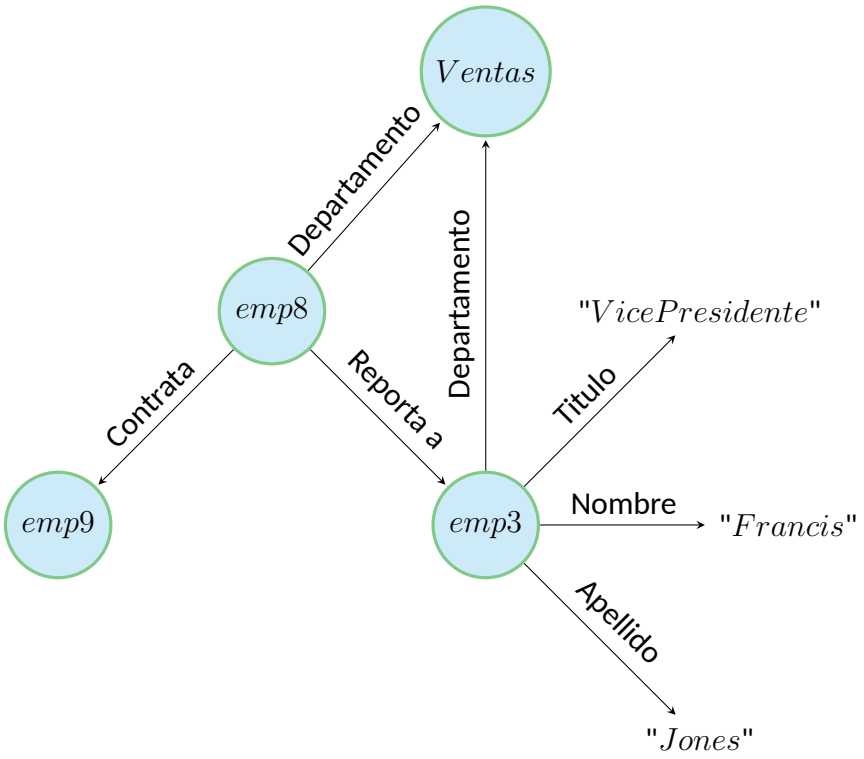


Figura 8: Ejemplo grafo RDF

2.8. SPARQL

Existen muchos estándares para cubrir a RDF como tal, pero no se ha hecho el trabajo de crear estándares para la realización de consultas y acceso a datos en RDF. No existe un lenguaje formal y públicamente estandarizado para la consulta de información en RDF.

A pesar de la falta de estándares, desarrolladores en proyectos comerciales y código abierto han creado muchos lenguajes de consulta para RDF, pero estos lenguajes carecen, tanto, de una sintaxis, como de una semántica en común [13].

La W3C recomienda a los desarrolladores que doten sus lenguajes con las funcionalidades necesarias para cubrir los casos de uso que se pueden ver en [13], que incluyen la de encontrar dirección de email, objetos multimedia desconocidos, monitorizar noticias, explorar

vecindarios, encontrar documentos de entrada y salida para pruebas de casos de uso y descubrir recursos de aprendizaje.

Uno de esos lenguajes es SPARQL, lenguaje diseñado para cumplir con todos los casos de uso mencionados en [13]. SPARQL es un lenguaje estandarizado para la consulta de datos RDF. Éste puede ser utilizado para expresar consultas a través de diversas fuentes de datos, tanto los datos sean guardados nativamente como RDF o vista como RDF por un software intermedio. SPARQL contiene capacidades para consultar patrones opcionales y requeridos en un grafo a lo largo de sus conjunciones y disjunciones. Los resultados de las consultas en SPARQL pueden ser tanto grafos RDF como tablas de resultados. [25]

Las consultas SPARQL son simplemente patrones de grafos básicos o complejos. Estos deben ser calzados en un grafo RDF, por ejemplo si la consulta:

```
PREFIX vcard: <http://www.w3.org/2006/vcard/ns#>

SELECT ?person
WHERE
{
   ?person vcard:family-name "Smith" .
}
```

Código 1: Ejemplo de consulta en SPARQL

Que busca calzar todos los triples que tengan como predicado `vcard:family-name`, como objeto `"Smith"` y como sujeto cualquier dato. Lo que se imprimirá en el resultado es `?person`, que será el sujeto de los triples calzados.
Se ejecuta sobre el grafo:

```
@prefix vcard: <http://www.w3.org/2006/vcard/ns#> .
@prefix sn: <http://www.snee.com/hr/> .

sn:emp1  vcard:given-name "Heidi" .
sn:emp1  vcard:family-name "Smith" .
sn:emp1  vcard:title "CEO" .
sn:emp1  sn:hireDate "2015-01-13" .
sn:emp1  sn:completedOrientation "2015-01-30" .

sn:emp2  vcard:given-name "John" .
sn:emp2  vcard:family-name "Smith" .
sn:emp2  sn:hireDate "2015-01-28" .
sn:emp2  vcard:title "Engineer" .
sn:emp2  sn:completedOrientation "2015-01-30" .
sn:emp2  sn:completedOrientation "2015-03-15" .

sn:emp3  vcard:given-name "Francis" .
sn:emp3  vcard:family-name "Jones" .
```

```
sn:emp3  sn:hireDate "2015-02-13" .
sn:emp3  vcard:title "Vice President" .

sn:emp4  vcard:given-name "Jane" .
sn:emp4  vcard:family-name "Berger" .
sn:emp4  sn:hireDate "2015-03-10" .
sn:emp4  vcard:title "Sales" .
```

Código 2: Ejemplo de datos en RDF

Se obtendrá el siguiente resultado:

?person
sn:emp1
sn:emp2

Tabla 4: Resultado al ejecutar la consulta SPARQL 1 en el grafo RDF 2

Esto indica que tanto el empleado 1 como el empleado 2 tienen como apellido “Smith”.

Otro ejemplo es el de la siguiente consulta:

```
PREFIX vcard <http://www.w3.org/2006/vcard/ns\#>
PREFIX sn:<http://www.snee.com/hr/>

SELECT ?givenName ?familyName ?oDate
WHERE
{
 ?person vcard:given-name ?givenName .
 ?person vcard:family-name ?familyName .
 ?person sn:completedOrientation ?oDate .
}
```

Código 3: Ejemplo 2 de datos en RDF

Esta consulta busca obtener el nombre, apellido y fecha de orientación de todos los empleados.

?givenName	?familyName	?oDate
John	Smith	2015-01-30
John	Smith	2015-03-15
Heidi	Smith	2015-01-30

Tabla 5: Resultado al ejecutar la consulta SPARQL 3 en el grafo RDF 2

El resultado no entrega todos los empleados por que no todos tienen una fecha de orientación, por lo que al no calzar con uno de los triples indicados en la consulta, no aparecen, pero si pueden aparecer dos veces en caso de que éste calce dos veces.

CAPÍTULO 3
PROPUESTA DE SOLUCIÓN

En el capítulo anterior se estudiaron distintas técnicas para asegurar la privacidad de los datos, siendo privacidad diferencial la escogida para poder trabajar en esta memoria. Además se hizo una contextualización respecto al modelo de datos RDF y su lenguaje de consulta más popular, SPARQL.

A continuación se describe cómo se aplicarán los conceptos de sensibilidad elástica definidos en ***Towards Practical Differential Privacy for SQL Queries***[17] para lograr privacidad diferencial en análisis estadísticos sobre un modelo de datos RDF al que se le harán consultas en SPARQL.

3.1. Sensibilidad elástica

La sensibilidad elástica de una consulta es definida recursivamente en la estructura de la misma. La sensibilidad elástica de una consulta q a una distancia k de la base de datos verdadera x se define como $\hat{S}^{(k)}(q, x)$. La función $\hat{S}$ esta definida en términos de la **estabilidad elástica** de trasformaciones relacionales ($\hat{S}^{(k)}_R$).

La *estabilidad elástica* $\hat{S}^{(k)}_R$ es definida en términos de la **máxima frecuencia** de un atributo a en una relación r a una distancia k de la base de datos x.

La métrica de *máxima frecuencia* se utilizara para la sensibilidad de JOINs. La máxima frecuencia mf(a, r, x) es definida como la frecuencia del valor más frecuente del atributo a en la relación r en la base de datos x.

Para vincular la sensibilidad local de una consulta a una distancia k de la base de datos verdadera, se debe también vincular la máxima frecuencia de cada JOIN Key a la distancia k, por lo que a este valor se le denota como mf$_k(a, r, x)$ que se define en términos de mf.

Ancestros. Los ancestros ($\mathcal{A}$) de una relación se utilizan para identificar *SELF JOINs*. Los SELF JOINs provocan un efecto mucho mayor en la sensibilidad que JOINs con relaciones no superpuestas. Un JOIN entre dos relaciones r_1 y r_2 es un SELF JOIN cuando r_1 y r_2 se superponen, esto ocurre cuando una tabla t contribuye filas tanto a r_1 como a r_2. r_1 y r_2 no se superponen cuando $|\mathcal{A}(r_1) \cap \mathcal{A}(r_2)| = 0$.

Sensibilidad elástica solo permite el uso de *EQUIJOIN*, incluyendo *SELF JOIN* y todos los tipos de realaciones de JOINs (uno a uno, uno a muchos y muchos a muchos). La definición formal de sensibilidad elástica es descrita en base un subconjunto del algebra relacional estándar.

Incluyendo JOIN $(\bowtie)$ y la función de agregado para contar $(Count)$.

A continuación se mostraran las definiciones formales de los términos expresados anteriormente:

Estabilidad Elástica

$$\hat{S}_R^{(k)}(t,x) \qquad = 1$$

$$\hat{S}_R^{(k)}(r_1 \underset{a=b}{\bowtie} r_2, x) = \begin{cases} \max\big(\mathsf{mf}_k(a,r_1,x)\hat{S}_R^{(k)}(r_2,x), & \\ \qquad \mathsf{mf}_k(b,r_2,x)\hat{S}_R^{(k)}(r_1,x)\big) & |\mathcal{A}(r_1)\cap\mathcal{A}(r_2)| = 0 \\ \mathsf{mf}_k(a,r_1,x)\hat{S}_R^{(k)}(r_2,x)+ & \\ \mathsf{mf}_k(b,r_2,x)\hat{S}_R^{(k)}(r_1,x)+ & \\ \hat{S}_R^{(k)}(r_1,x)\hat{S}_R^{(k)}(r_2,x) & |\mathcal{A}(r_1)\cap\mathcal{A}(r_2)| > 0 \end{cases}$$

Sensibilidad elástica

$$\hat{S}^{(k)}(Count(r),x) = \hat{S}_R^{(k)}(r,x)$$

Máxima frecuencia a una distancia k

$$\mathsf{mf}_k(a,t,x) = \mathsf{mf}(a,t,x) + k$$

$$\mathsf{mf}_k(a_1, r_1 \underset{a_2=a_3}{\bowtie} r_2, x) = \begin{cases} \mathsf{mf}_k(a_1,r_1,x)\mathsf{mf}_k(a_3,r_2,x) & a_1 \in r_1 \\ \mathsf{mf}_k(a_1,r_2,x)\mathsf{mf}_k(a_2,r_1,x) & a_1 \in r_2 \end{cases}$$

Ancestros de una relación

$$\mathcal{A}(t) = \{t\}$$
$$\mathcal{A}(r_1 \underset{a=b}{\bowtie} r_2) = \mathcal{A}(r_1) \cup \mathcal{A}(r_2)$$

La utilización de estos conceptos para el cálculo de la sensibilidad afectan la implementación la distribución de Laplace.

A medida que la máxima frecuencia aumenta, también lo hace la estabilidad de la consulta como se puede ver en las relaciones matemáticas planteadas anteriormente y la máxima frecuencia depende de la distancia, por lo que la estabilidad depende de ésta de igual manera.

La utilización de los ancestros permite la identificación de SELF JOINs como se mencionó anteriormente, al momento de realizar SELF JOINs se hace un cálculo distinto de la estabilidad generando un aumento del valor de ésta.

El cálculo del ruido con la distribución de Laplace depende directamente de la sensibilidad y debido a que ésta depende directamente del cálculo de la estabilidad, se puede determinar que la máxima frecuencia, la distancia y los ancestros afectan en el cálculo del ruido.

El diseño planteado en [17] presenta una manera de implementar privacidad diferencial en bases de datos relacionales, por lo que para poder adecuar la solución para un modelo de datos RDF es necesario realizar algunos cambios y adaptaciones.
Dentro de los supuestos a realizar se tiene que se tomará en cuanta que todos los triples dentro de la consulta en SPARQL participaran en un JOIN.

3.2. Extensión a RDF

Al momento de realizar una consulta en el grafo G, esta se realizará con un patrón P en SPARQL que será definido recursivamente de la siguiente manera:

- Un patrón P puede estar conformado por subpatrones: $P = (P_1 \text{ AND } P_2)$, por lo tanto $[\![P]\!]_G = [\![P_1]\!]_G \bowtie [\![P_2]\!]_G$
- Un patrón puede ser un triple: $P = t$

Al cambiar de modelo de datos de uno relacional a RDF también cambia la estructura de las operaciones que se pueden hacer, dentro de las que se encuentra JOIN.
En un patrón SPARQL el operador JOIN funciona por cada triple y se van agregando hasta el último triple que participa en la operación.

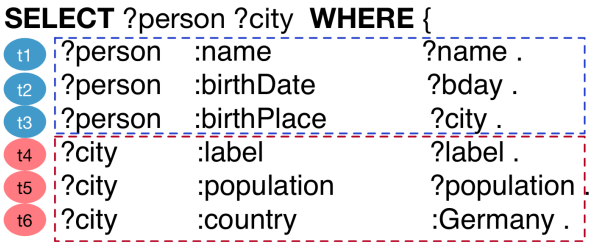


Figura 9: Ejemplo consulta SPARQL con JOIN [12]

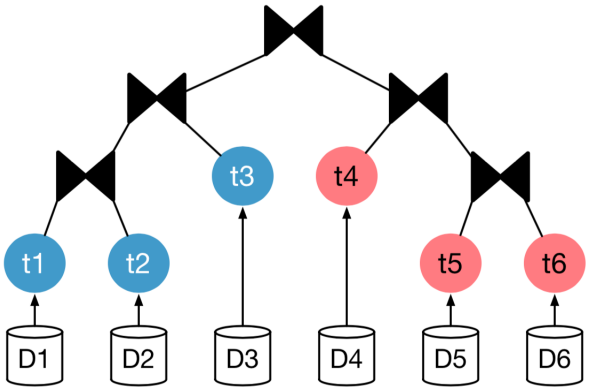


Figura 10: Modelo de como se hace un JOIN en SPARQL [12]

Como se puede observar en las figuras 9 y 10 la consulta SPARQL esta conformada por seis triples $(t_1, ..., t_6)$ que estan agrupados en subconsultas con forma de estrella.
Primero se realiza el JOIN entre t_1 y t_2, luego se hace un JOIN de este resultado con t_3, esto mismo ocurre con t_4, t_5 y t_6, por último se hace un JOIN final entre los resultados se las dos subconsultas.

De esta manera ahora el cálculo de la sensibilidad elástica será:

$$\hat{S}^{(k)}(Count(P), G) = \hat{S}_R^{(k)}(P, G)$$

Estabilidad elástica, siendo V el conjunto de variables que se utilizan para realizar el JOIN.

$$\hat{S}_R^{(k)}(t, G) \qquad = 1$$

$$\hat{S}_R^{(k)}(P_1 \text{ AND } P_2, G) = \begin{cases} \max\big(\text{mf}_k(V, var(P_1), G)\hat{S}_R^{(k)}(var(P_2), G), \\ \qquad \text{mf}_k(V, var(P_2), G)\hat{S}_R^{(k)}(var(P_1), G)\big) & |\mathcal{A}(P_1) \cap \mathcal{A}(P_2)| = 0 \\ \\ \text{mf}_k(V, var(P_1), G)\hat{S}_R^{(k)}(var(P_2), G) + \\ \text{mf}_k(V, var(P_2), G)\hat{S}_R^{(k)}(var(P_1), G) + \\ \hat{S}_R^{(k)}(var(P_1), G)\hat{S}_R^{(k)}(var(P_2), G) & |\mathcal{A}(P_1) \cap \mathcal{A}(P_2)| > 0 \end{cases}$$

Para el calculo de la máxima frecuencia existe una nueva definición ya que no solo existe una única posible JOIN Key

$$\text{mf}_k(V, t, G) = \text{mf}(V, t, G) + k$$

$$\text{mf}_k(V, P_1 \text{ AND } P_2, G) = \begin{cases} \text{mf}_k(V', P_1, G)\text{mf}_k(V, P_2, G) & V' \in var(P_1) \cap V \\ \text{mf}_k(V', P_2, G)\text{mf}_k(V, P_1, G) & V' \in var(P_2) \cap V \end{cases}$$

$$\text{mf}_k(V', P, G) = min(\text{mf}_k(V_1, P, G), \text{mf}_k(V_2, P, G))$$

Esta solución propuesta tomara como supuesto una distancia de $k = 1$ entre las distintas bases de datos, lo que nos entregará una privacidad diferencial acotada.

3.3. Tecnologías

Para el desarrollo de la solución se utilizaran las siguientes tecnologías

3.3.1. IntelliJ IDEA

- IntelliJ IDEA 2018.2.1 (Ultimate Edition)
- Build #IU-182.3911.36, built on August 6, 2018
- JRE: 1.8.0_152-release-1248-b8 x86_64
- JVM: OpenJDK 64-Bit Server VM by JetBrains s.r.o

3.3.2. *Header, Dictionary, Triples*

Actualmente los datos en RDF se almacenan y envían en formatos de serialización de texto muy detallados que desperdician una gran cantidad de ancho de manda y son muy costosos al momento de parsear e indexar.

Header, *Dictionary*, *Triples* o HDT es una estructura de datos compacta y un formato de serialización binario para RDF que mantiene grandes conjuntos de datos de manera comprimida para ahorrar espacio, pero manteniendo las operaciones de búsqueda y navegación sin necesidad de descomprimir los datos previamente.

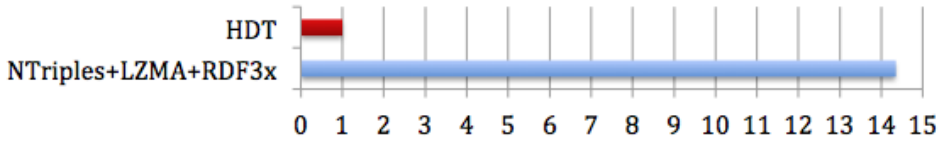


Figura 11: Comparación del tiempo para la descarga y consulta de un conjunto de datos entre HDT y las técnicas tradicionales [26]

Algunos datos sobre HDT:

- **El tamaño de los archivos es más pequeño**. Son menores los costos de ancho de banda para el proveedor y menor tiempo de espera para los consumidores al momento de la descarga.

- **El archivo HDT esta ya indexado**. No es necesario perder tiempo con herramientas de parseo e indexado.
- **Alto rendimiento de las consultas**. Las técnicas de compresión de HDT permiten que gran parte de los datos se pueda mantener en la memoria principal, por lo que el tiempo al realizar consultas se reduce.
- **Altamente concurrente**. HDT es un archivo de solo lectura, por lo que se pueden realizar múltiples consultas simultaneamente.

[26]

```java
public HdtDataSource(String hdtFile) throws IOException
    {
        datasource = HDTManager.mapIndexedHDT(hdtFile, null);
        dictionary = new NodeDictionary(datasource.getDictionary());
        graph = new HDTGraph(datasource);
        triples = ModelFactory.createModelForGraph(graph);
    }
```

Código 4: Ejemplo del uso de HDT en Java

El Código 4 muestra como se utilizará HDT para el manejo del modelo de datos RDF.

3.3.3. Apache JENA

Apache Jena [9] es un framework de Java para crear aplicaciones de Web Semántica. El framework esta compuesto de diferentes APIs que interactúan para el procesamiento de datos en RDF utilizando los elementos anteriormente mencionados como nodos, variables, grafos, etc. En esta memoria se utilizará Apache Jena para el manejo de las consultas SPARQL al modelo RDF HDT.

Sea una consulta query.rq:

```
SELECT (COUNT(?v2) as ?count) WHERE {
  ?v2 <http://schema.org/eligibleRegion>
  <http://db.uwaterloo.ca/~galuc/wsdbm/Country20> .
  ?v0 <http://purl.org/goodrelations/offers> ?v2 .
  ?v2 <http://purl.org/goodrelations/price> ?v3 .
  ?v2 <http://purl.org/goodrelations/validFrom> ?v4 .
  ?v2 <http://purl.org/goodrelations/validThrough> ?v5 .
  ?v0 <http://purl.org/goodrelations/name> ?v1 .
  ?v0 <http://schema.org/contactPoint> ?v8 .
  ?v0 <http://schema.org/email> ?v10 .
  ?v0 <http://schema.org/openingHours> ?v11 .
  ?v6 <http://purl.org/stuff/rev#reviewer> ?v8 .
  ?v6 <http://purl.org/stuff/rev#rating> ?v7 .
```

```
14 }
```

Código 5: Consulta en SPARQL para aplicar con Apache Jena

```
1 Query query = QueryFactory.create("query.rq");
2 try(QueryExecution qexec = QueryExecutionFactory.create(query,triples)){
3           ResultSet results = qexec.execSelect();
4           results = ResultSetFactory.copyResults(results) ;
5       }
```

Código 6: Ejemplo del manejo de consultas SPARQL utilizando Apache Jena

CAPÍTULO 4
VALIDACIÓN DE LA SOLUCIÓN

4.1. Solución para SQL

En esta sección se analizará la solución propuesta en [17] para la aplicación de privacidad diferencial en un sistema basado en bases de datos SQL.

El código con el que se realizaron las pruebas se encuentra en el repositorio público:
https://github.com/uber/sql-differential-privacy

Para la realización de las pruebas se utilizará TPC Benchmark H (TPC-H), que es un benchmark para el apoyo de decisiones. Consiste en un conjunto de consultas y modificaciones concurrentes de datos orientadas a los negocios. Este benchmark muestra sistemas de toma de decisiones que:

- Examinan grandes volúmenes de datos
- Ejecuta consultas con un alto grado de complejidad
- Entrega respuestas a preguntas de negocios críticas

La razón principal para este estudio es el obtener un modelo de datos con un volumen adecuado [28].
El esquema de la base de datos es el siguiente:

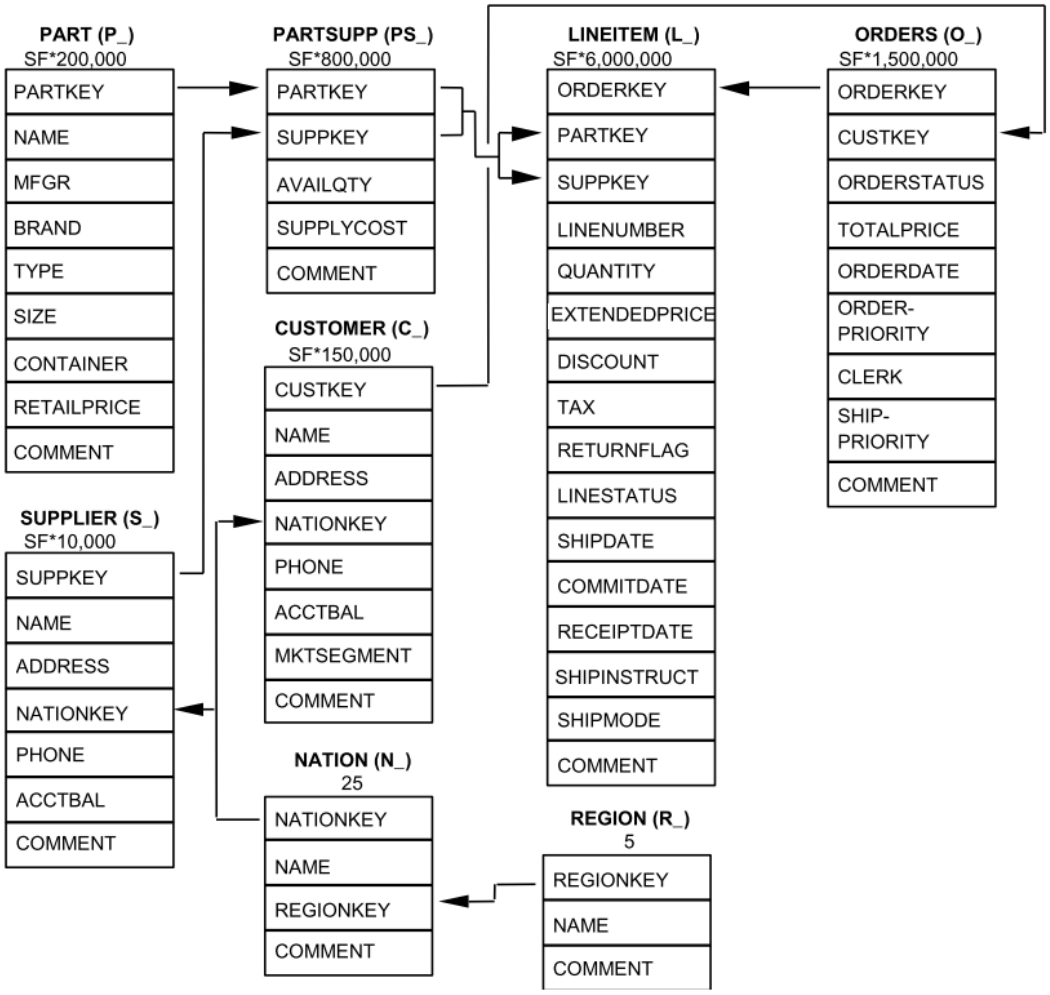


Figura 12: Esquema de la base de datos TPC-H [28]

TPC-H ofrece varias consultas diseñadas para probar el sistema, pero debido a los requerimientos específicos para realizar un análisis adecuado se crearan consultas propias.

```
1 SELECT count(distinct orders.ORDERKEY)
2    FROM orders
3        JOIN customer ON orders.CUSTKEY = customer.CUSTKEY
4        JOIN lineitem ON orders.ORDERKEY = lineitem.ORDERKEY
5        JOIN partsupp ON lineitem.PARTKEY = partsupp.PARTKEY
6        JOIN supplier ON supplier.SUPPKEY = partsupp.SUPPKEY
```

Código 7: Consulta en SQL para pruebas de privacidad diferencial

Se realizara un análisis de los resultados de la consulta anterior con 1, 2, 3 y 4 JOINs por separado. Para cada JOIN se hicieron 10 pruebas para poder obtener una media de los resultados, se calculo también la sensibilidad de la consulta.

Los siguientes resultados corresponden a la consulta realizada con un presupuesto de privacidad de 0.1, el resultado real de la consulta es 3000000.

Sensibilidad	185,743907315323	119574,652436273	1,34E+08	2,13E+11
Resultado	**1 JOIN**	**2 JOINs**	**3 JOINs**	**4 JOINs**
1	3001409	3186991	-106515480	5609094101702
2	3010577	3338064	822028528	-3177398578921
3	2999156	1820343	-5446198804	-10259541839504
4	3002255	-1218458	1039573335	1032114075152
5	3006904	-12784393	-154243712	-1279560675446
6	3003355	7362824	3922580479	2655182865759
7	2995146	4214312	512727038	-3182304950465
8	2995948	-9168371	699244618	-10943317044049
9	3005730	3112813	4236332884	-18807924204031
10	2992522	6933045	-1068866105	13986661493692

Tabla 6: Resultados consulta en TPC-H con presupuesto 0.1

Luego se realizará el mismo procedimiento con presupuestos de 0.01 y 1 respectivamente, la sensibilidad de las consultas se mantiene.

Resultado	**1 JOIN**	**2 JOINs**	**3 JOINs**	**4 JOINs**
1	3075570	981908377	-32236534481495	-881198301425774000
2	2044417	-351817563	36547713613072	-109622543429750000
3	2974177	759774685	53707846309799	-40166105928726000
4	2884122	1502341051	-156797809867347	-178387273104453000
5	2773564	-3697047045	-265963053227	-238442820542574000
6	3213672	-1860556435	-9127051828237	136250204686892000
7	3000618	-795159203	-27597339137777	-573701220488475000
8	4333462	5823464610	-50762641944880	751668325711051000
9	2902474	-568120169	-7510012562787	81075420234410600
10	3366613	-1175383101	9161757160113	544815619130790000

Tabla 7: Resultados consulta en TPC-H con presupuesto 0.01

Resultado	1 JOIN	2 JOINs	3 JOINs	4 JOINs
1	2999962	3001344	2520721	159421996
2	3000024	2995076	2654485	-107951860
3	3000366	3002488	2704077	98652399
4	3000068	2996019	3335976	24350527
5	2999988	3002717	3132098	-89473855
6	3000031	3006473	2942385	-67505108
7	3000027	2999828	5663294	-100721051
8	3000005	3004353	2787156	-6162960
9	2999866	3001340	2506062	-12811490
10	2999935	2996467	3040507	60880165

Tabla 8: Resultados consulta en TPC-H con presupuesto 1

Ahora para poder visualizar mejores los resultados se generará un gráfico que mostrará la diferencia entre el resultado real y la media de los resultados con privacidad diferencial.

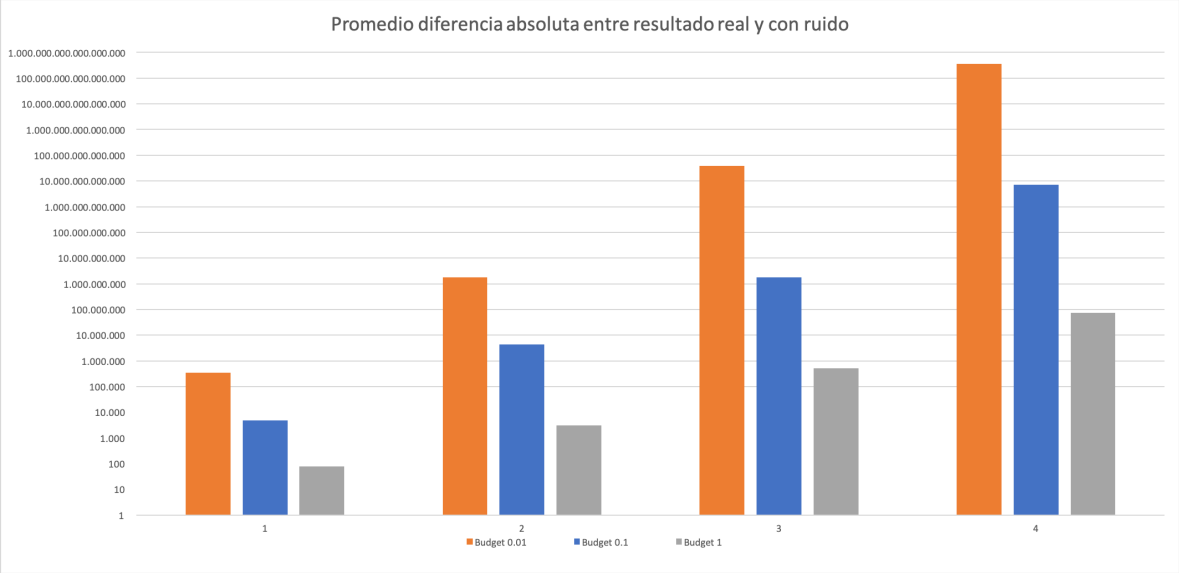


Figura 13: Diferencia entre resultados reales y con ruido en SQL

Como se puede observar en la figura 13, a medida que la cantidad de JOIN aumenta también lo hace la sensibilidad de la consulta, lo que provoca un aumento en el valor del resultado final. También es posible observar de manera empírica que mientras mayor es el presupuesto de privacidad, más preciso es el resultado de la consulta, mientras que si el presupuesto es bajo, los valores del resultado se disparan.

4.2. Solución para RDF

En esta sección se analizará el correcto funcionamiento de la solución implementada para el modelo de datos RDF mediante la realización de diversas pruebas.
El software implementado se encuentra disponible en el repositorio público:
https://github.com/cbuil/PrivacyTest1

Con el crecimiento de las bases de datos NoSQL existe una alta demanda de sistemas RDF para el manejo de datos de alto rendimiento, existen múltiples sistemas que cumplen con estas características, pero las consultas cada vez se vuelven más y más diversas. Es por esta razón que para las pruebas que se realizarán al software desarrollado se utilizarán las estructuras de consultas expuestas en [1].

Antes de hablar de los tipos de estructuras de consultas SPARQL es necesario definir algunos términos importantes.

- **Vértice de JOIN**: Un elemento x es un vértice de JOIN si existen al menos dos triples en los que x es el sujeto o el objeto.
- **Grado de un vértice de JOIN**: El grado de un vértice de JOIN x corresponde a el número de triples donde el sujeto o el objeto corresponde a x.

Existen 3 estructuras posibles para consultas en SPARQL.

```
1 SELECT (COUNT(?v0) as ?v0_count) WHERE {
2   ?v0 <http://db.uwaterloo.ca/~galuc/wsdbm/subscribes>
        <http://db.uwaterloo.ca/~galuc/wsdbm/Website658> .
3   ?v0 <http://db.uwaterloo.ca/~galuc/wsdbm/likes> ?v2 .
4   ?v2 <http://schema.org/caption> ?v3 .
5 }
```

Código 8: Código consulta con estructura linear

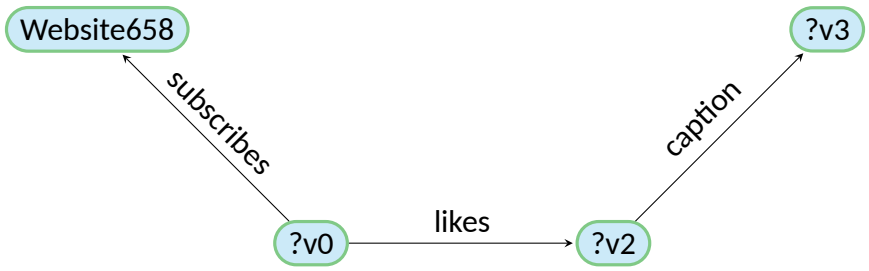


Figura 14: Grafo consulta con estructura linear

En la figura 14 se puede observar una consulta linear, este tipo de consultas se caracteriza por seguir una linea con múltiples (2) vértices de JOIN con un bajo grado (2).

```
SELECT (COUNT(?v1) as ?v1_count) WHERE {
   ?v0 <http://purl.org/dc/terms/Location> ?v1 .
   ?v0 <http://schema.org/nationality>
       <http://db.uwaterloo.ca/~galuc/wsdbm/Country14> .
   ?v0 <http://db.uwaterloo.ca/~galuc/wsdbm/gender> ?v3 .
   ?v0 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
       <http://db.uwaterloo.ca/~galuc/wsdbm/Role2> .
}
```

Código 9: Código consulta con estructura de estrella

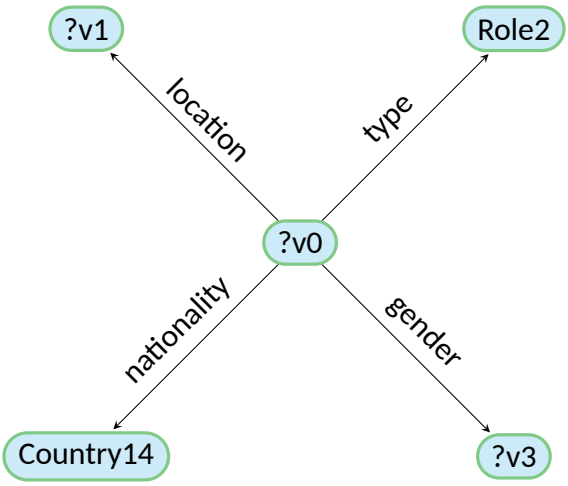


Figura 15: Grafo consulta con estructura de estrella

Como se puede observar en la figura 15, las consultas con estructura de estrella están compuestas por un único vértice con un alto grado de JOIN (5).

```
SELECT (COUNT(?v6) as ?v6_count) WHERE {
   ?v0 <http://xmlns.com/foaf/homepage> ?v1 .
   ?v2 <http://purl.org/goodrelations/includes> ?v0 .
   ?v0 <http://ogp.me/ns#tag> <http://db.uwaterloo.ca/~galuc/wsdbm/Topic154>
       .
   ?v0 <http://schema.org/description> ?v4 .
   ?v0 <http://schema.org/contentSize> ?v8 .
   ?v1 <http://schema.org/url> ?v5 .
   ?v1 <http://db.uwaterloo.ca/~galuc/wsdbm/hits> ?v6 .
   ?v1 <http://schema.org/language>
       <http://db.uwaterloo.ca/~galuc/wsdbm/Language0> .
   ?v7 <http://db.uwaterloo.ca/~galuc/wsdbm/likes> ?v0 .
}
```

Código 10: Código consulta con estructura de copo de nieve

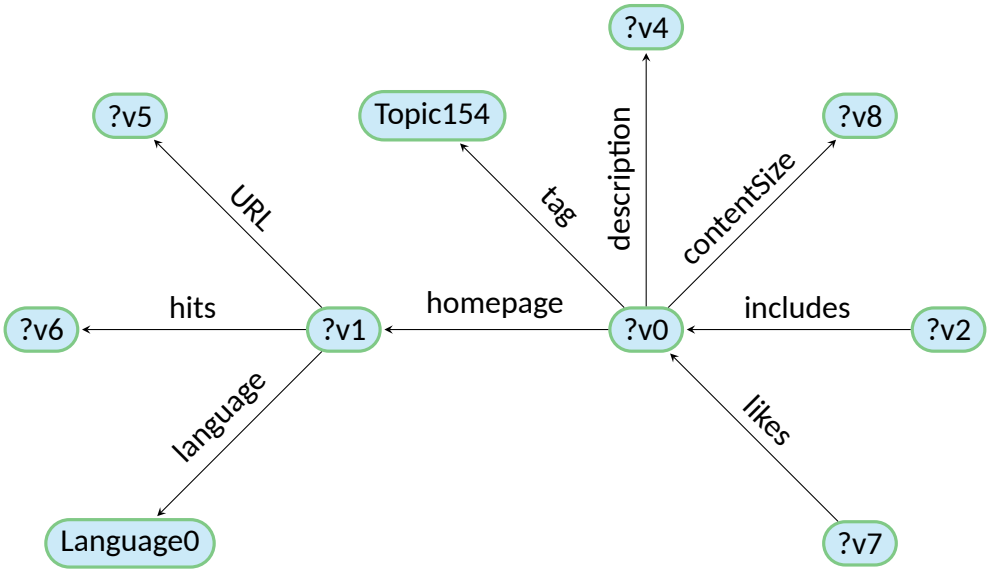


Figura 16: Grafo consulta con estructura de copo de nieve

Por último se puede observar en la figura 16 que las consultas con estructura de copo de nieve están compuestas por múltiples vértices con un alto grado de JOIN.

Ahora para poder realizar el análisis similar al anterior se separaran las consultas por el presupuesto de sensibilidad, nuevamente se probará con los presupuestos 0.1, 0.01 y 1.

Se ejecutará un aproximado de 15.000 consultas entre todos los tipos de las estructuras mencionadas anteriormente, a continuación se mostrarán las consultas realizadas.

4.2.1. Consultas con estructura de copo de nieve

1. Resultado original: 6

```
1  SELECT (COUNT(?v3) as ?v3_count) WHERE {
2  ?v0 <http://ogp.me/ns#tag>
       <http://db.uwaterloo.ca/~galuc/wsdbm/Topic66> .
3  ?v0 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?v2 .
4  ?v3 <http://db.uwaterloo.ca/~galuc/wsdbm/hasGenre> ?v0 .
5  ?v3 <http://schema.org/trailer> ?v4 .
6  ?v3 <http://schema.org/keywords> ?v5 .
7  ?v3 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
       <http://db.uwaterloo.ca/~galuc/wsdbm/ProductCategory2> .
8   }
```

Código 11: Ejemplo consulta F1

2. Resultado original: 9

```
1   SELECT (COUNT(?v1) as ?v1_count) WHERE {
2  ?v0 <http://xmlns.com/foaf/homepage> ?v1 .
3  ?v0 <http://ogp.me/ns#title> ?v2 .
4  ?v0 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?v3 .
5  ?v0 <http://schema.org/caption> ?v4 .
6  ?v0 <http://schema.org/description> ?v5 .
7  ?v1 <http://schema.org/url> ?v6 .
8  ?v1 <http://db.uwaterloo.ca/~galuc/wsdbm/hits> ?v7 .
9  ?v0 <http://db.uwaterloo.ca/~galuc/wsdbm/hasGenre>
       <http://db.uwaterloo.ca/~galuc/wsdbm/SubGenre87> .
10  }
```

Código 12: Ejemplo consulta F2

3. Resultado original: 3

```
1   SELECT (COUNT(?v5) as ?v5_count) WHERE {
2  ?v0 <http://schema.org/contentRating> ?v1 .
3  ?v0 <http://schema.org/contentSize> ?v2 .
4  ?v0 <http://db.uwaterloo.ca/~galuc/wsdbm/hasGenre>
       <http://db.uwaterloo.ca/~galuc/wsdbm/SubGenre31> .
5  ?v5 <http://db.uwaterloo.ca/~galuc/wsdbm/purchaseFor> ?v0 .
6  ?v4 <http://db.uwaterloo.ca/~galuc/wsdbm/makesPurchase> ?v5 .
7  ?v5 <http://db.uwaterloo.ca/~galuc/wsdbm/purchaseDate> ?v6 .
8  }
```

Código 13: Ejemplo consulta F3

4. Resultado original: 35

```
1   SELECT (COUNT(?v6) as ?v6_count) WHERE {
2  ?v0 <http://xmlns.com/foaf/homepage> ?v1 .
3  ?v2 <http://purl.org/goodrelations/includes> ?v0 .
4  ?v0 <http://ogp.me/ns#tag>
       <http://db.uwaterloo.ca/~galuc/wsdbm/Topic154> .
5  ?v0 <http://schema.org/description> ?v4 .
6  ?v0 <http://schema.org/contentSize> ?v8 .
7  ?v1 <http://schema.org/url> ?v5 .
8  ?v1 <http://db.uwaterloo.ca/~galuc/wsdbm/hits> ?v6 .
9  ?v1 <http://schema.org/language>
       <http://db.uwaterloo.ca/~galuc/wsdbm/Language0> .
10 ?v7 <http://db.uwaterloo.ca/~galuc/wsdbm/likes> ?v0 .
11  }
```

Código 14: Ejemplo consulta F4

5. Resultado original: 22

```
1  SELECT (COUNT(?v1) as ?v1_count) WHERE {
2  ?v0 <http://purl.org/goodrelations/includes> ?v1 .
3  <http://db.uwaterloo.ca/~galuc/wsdbm/Retailer46>
       <http://purl.org/goodrelations/offers> ?v0 .
4  ?v0 <http://purl.org/goodrelations/price> ?v3 .
5  ?v0 <http://purl.org/goodrelations/validThrough> ?v4 .
6  ?v1 <http://ogp.me/ns#title> ?v5 .
7  ?v1 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?v6 .
8  }
```

Código 15: Ejemplo consulta F5

4.2.2. Consultas con estructura linear

1. Resultado original: 5

```
1  SELECT (COUNT(?v0) as ?v0_count) WHERE {
2  ?v0 <http://db.uwaterloo.ca/~galuc/wsdbm/subscribes>
       <http://db.uwaterloo.ca/~galuc/wsdbm/Website658> .
3  ?v0 <http://db.uwaterloo.ca/~galuc/wsdbm/likes> ?v2 .
4  ?v2 <http://schema.org/caption> ?v3 .
5  }
```

Código 16: Ejemplo consulta L1

2. Resultado original: 236

```
1  SELECT (COUNT(?v2) as ?v2_count) WHERE {
2  <http://db.uwaterloo.ca/~galuc/wsdbm/City74>
       <http://www.geonames.org/ontology#parentCountry> ?v1 .
3  ?v2 <http://schema.org/nationality> ?v1 .
4  ?v2 <http://db.uwaterloo.ca/~galuc/wsdbm/likes>
       <http://db.uwaterloo.ca/~galuc/wsdbm/Product0> .
5  }
```

Código 17: Ejemplo consulta L2

3. Resultado original: 21

```
1  SELECT (COUNT(?v1) as ?v1_count) WHERE {
2  ?v0 <http://db.uwaterloo.ca/~galuc/wsdbm/likes> ?v1 .
3  ?v0 <http://db.uwaterloo.ca/~galuc/wsdbm/subscribes>
       <http://db.uwaterloo.ca/~galuc/wsdbm/Website262> .
4  }
```

Código 18: Ejemplo consulta L3

4. Resultado original: 60

```
1  SELECT (COUNT(?v2) as ?v2_count) WHERE {
2  ?v0 <http://ogp.me/ns#tag>
       <http://db.uwaterloo.ca/~galuc/wsdbm/Topic11> .
3  ?v0 <http://schema.org/caption> ?v2 .
4  }
```

Código 19: Ejemplo consulta L4

5. Resultado original: 269

```
1  SELECT (COUNT(?v0) as ?v0_count) WHERE {
2  ?v0 <http://schema.org/jobTitle> ?v1 .
3  ?v0 <http://schema.org/nationality> ?v3 .
4  <http://db.uwaterloo.ca/~galuc/wsdbm/City84>
       <http://www.geonames.org/ontology#parentCountry> ?v3 .
5  }
```

Código 20: Ejemplo consulta L5

4.2.3. Consultas con estructura de estrella

1. Resultado original: 16

```
1  SELECT (COUNT(?v0) as ?v0_count) WHERE {
2  ?v0 <http://purl.org/goodrelations/includes> ?v1 .
3  <http://db.uwaterloo.ca/~galuc/wsdbm/Retailer118>
       <http://purl.org/goodrelations/offers> ?v0 .
4  ?v0 <http://purl.org/goodrelations/price> ?v3 .
5  ?v0 <http://purl.org/goodrelations/serialNumber> ?v4 .
6  ?v0 <http://purl.org/goodrelations/validFrom> ?v5 .
7  ?v0 <http://purl.org/goodrelations/validThrough> ?v6 .
8  ?v0 <http://schema.org/eligibleQuantity> ?v7 .
9  ?v0 <http://schema.org/eligibleRegion> ?v8 .
10 ?v0 <http://schema.org/priceValidUntil> ?v9 .
11 }
```

Código 21: Ejemplo consulta S1

2. Resultado original: 16

```
1  SELECT (COUNT(?v1) as ?v1_count) WHERE {
2  ?v0 <http://purl.org/dc/terms/Location> ?v1 .
3  ?v0 <http://schema.org/nationality>
       <http://db.uwaterloo.ca/~galuc/wsdbm/Country14> .
```

```
4    ?v0 <http://db.uwaterloo.ca/~galuc/wsdbm/gender> ?v3 .
5    ?v0 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
        <http://db.uwaterloo.ca/~galuc/wsdbm/Role2> .
6     }
```

Código 22: Ejemplo consulta S2

3. Resultado original: 0

```
1     SELECT (COUNT(?v4) as ?v4_count) WHERE {
2    ?v0 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
        <http://db.uwaterloo.ca/~galuc/wsdbm/ProductCategory11> .
3    ?v0 <http://schema.org/caption> ?v2 .
4    ?v0 <http://db.uwaterloo.ca/~galuc/wsdbm/hasGenre> ?v3 .
5    ?v0 <http://schema.org/publisher> ?v4 .
6     }
```

Código 23: Ejemplo consulta S3

4. Resultado original: 1

```
1     SELECT (COUNT(?v0) as ?v0_count) WHERE {
2    ?v0 <http://xmlns.com/foaf/age>
        <http://db.uwaterloo.ca/~galuc/wsdbm/AgeGroup4> .
3    ?v0 <http://xmlns.com/foaf/familyName> ?v2 .
4    ?v3 <http://purl.org/ontology/mo/artist> ?v0 .
5    ?v0 <http://schema.org/nationality>
        <http://db.uwaterloo.ca/~galuc/wsdbm/Country1> .
6     }
```

Código 24: Ejemplo consulta S4

5. Resultado original: 0

```
1     SELECT (COUNT(?v3) as ?v3_count) WHERE {
2    ?v0 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
        <http://db.uwaterloo.ca/~galuc/wsdbm/ProductCategory14> .
3    ?v0 <http://schema.org/description> ?v2 .
4    ?v0 <http://schema.org/keywords> ?v3 .
5    ?v0 <http://schema.org/language>
        <http://db.uwaterloo.ca/~galuc/wsdbm/Language0> .
6     }
```

Código 25: Ejemplo consulta S5

4.2.4. *Testing*

En esta sección se verán los promedios de los resultados obtenidos al efectuar las consultas mencionadas anteriormente:

Consulta	Resultado original	Resultado privado	Sensibilidad	# Triples
F1	6	145591	19786	6
F2	6.12	9003.3	4916	8
F3	9.15	4389735.16	2341791	6
F4	28.48	-32414026,75	28598577	9
F5	40.72	883.05	547	6
L1	2.97	-14.37	22	3
L2	70.53	-27969.99	26207	3
L3	37.6	28.31	6	2
L4	59.39	74.38	5	2
L5	93.1	90.85	17	3
S1	5.42	30042.11	29508	9
S2	38.4	-176.42	58	4
S3	19.8	-523.63	115	4
S4	3.22	4602.96	2471	4
S5	2.27	419.35	58	4

Tabla 9: Resultados consultas SPARQL con presupuesto 0.1

Consulta	Resultado original	Resultado privado	Sensibilidad	# Triples
F1	6	-5024344	19786	6
F2	6.12	9437.58	4916	8
F3	9.15	-40516848.46	2341791	6
F4	28.48	-131449131.7	28598577	9
F5	40.72	-6681.48	547	6
L1	2.97	54.11	22	3
L2	70.53	761329.17	26207	3
L3	37.6	268.05	6	2
L4	59.39	57.41	5	2
L5	93.1	425.27	17	3
S1	5.42	235848.36	29508	9
S2	38.4	-1752.44	58	4
S3	19.8	-1750.38	115	4
S4	3.22	-50427.26	2471	4
S5	2.27	2683.29	58	4

Tabla 10: Resultados consultas SPARQL con presupuesto 0.01

Consulta	Resultado original	Resultado privado	Sensibilidad	# Triples
F1	6	686.48	19786	6
F2	9	139701.52	4916	8
F3	9.15	2890094	2341791	6
F4	28.48	141270.15	28598577	9
F5	40.72	106.53	547	6
L1	2.97	7.16	22	3
L2	70.53	-1227.37	26207	3
L3	37.6	37.93	6	2
L4	59.39	60.1	5	2
L5	93.1	86.39	17	3
S1	5.42	2039.45	29508	9
S2	38.4	50.55	58	4
S3	19.8	58.48	115	4
S4	3.22	-731	2471	4
S5	2.27	-52,16	58	4

Tabla 11: Resultados consultas SPARQL con presupuesto 1

A continuación se mostrará una comparación de la diferencia absoluta entre el promedio del resultado real y el original según la estructura de consultas y los presupuestos:

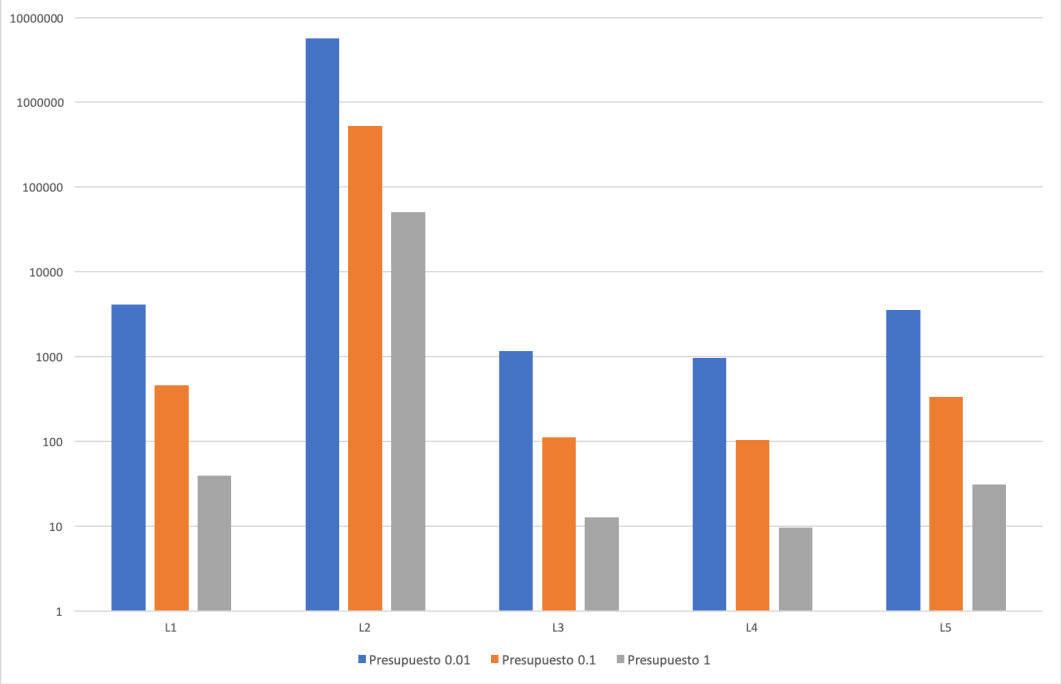


Figura 17: Diferencia entre resultado real y con ruido de consultas SPARQL lineales

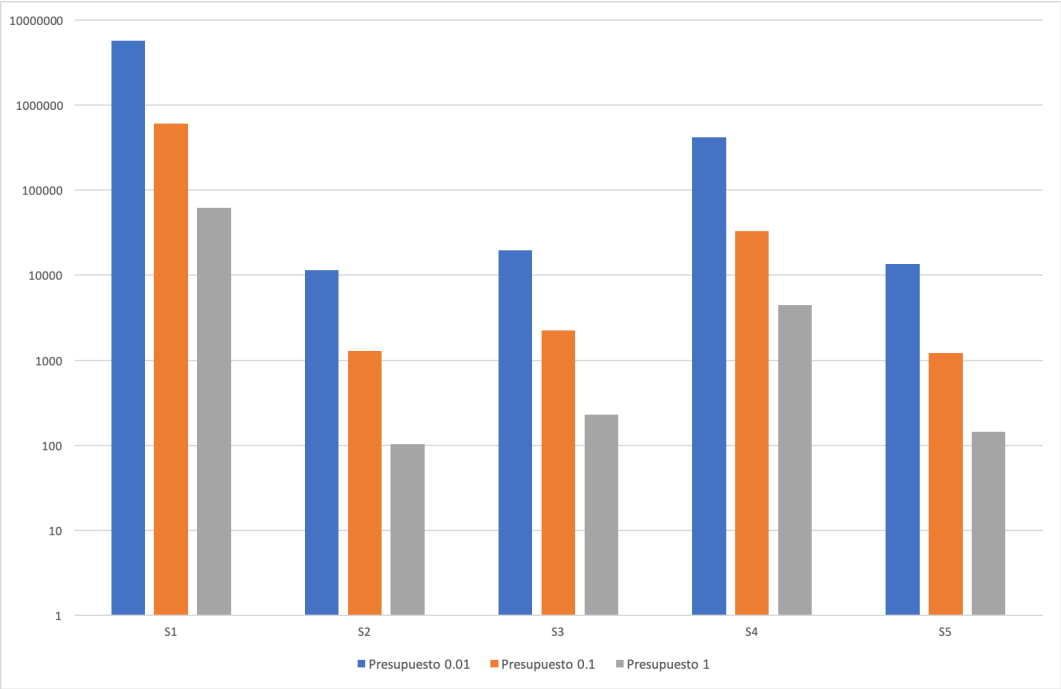


Figura 18: Diferencia entre resultado real y con ruido de consultas SPARQL con estructura de estrella

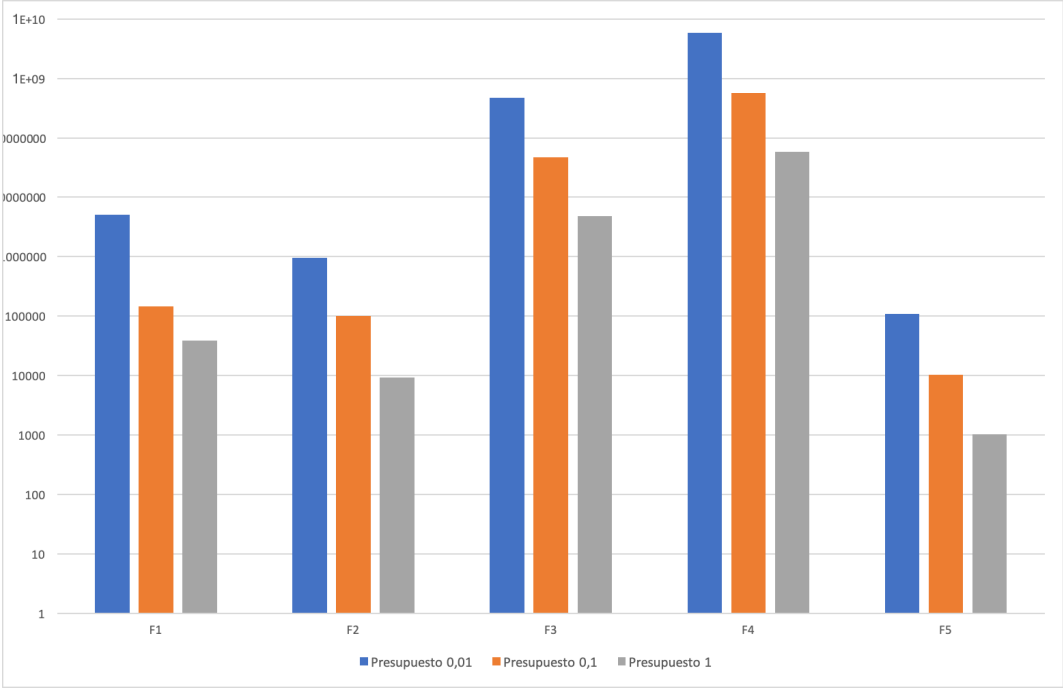


Figura 19: Diferencia entre resultado real y con ruido de consultas SPARQL con estructura de copo de nieve

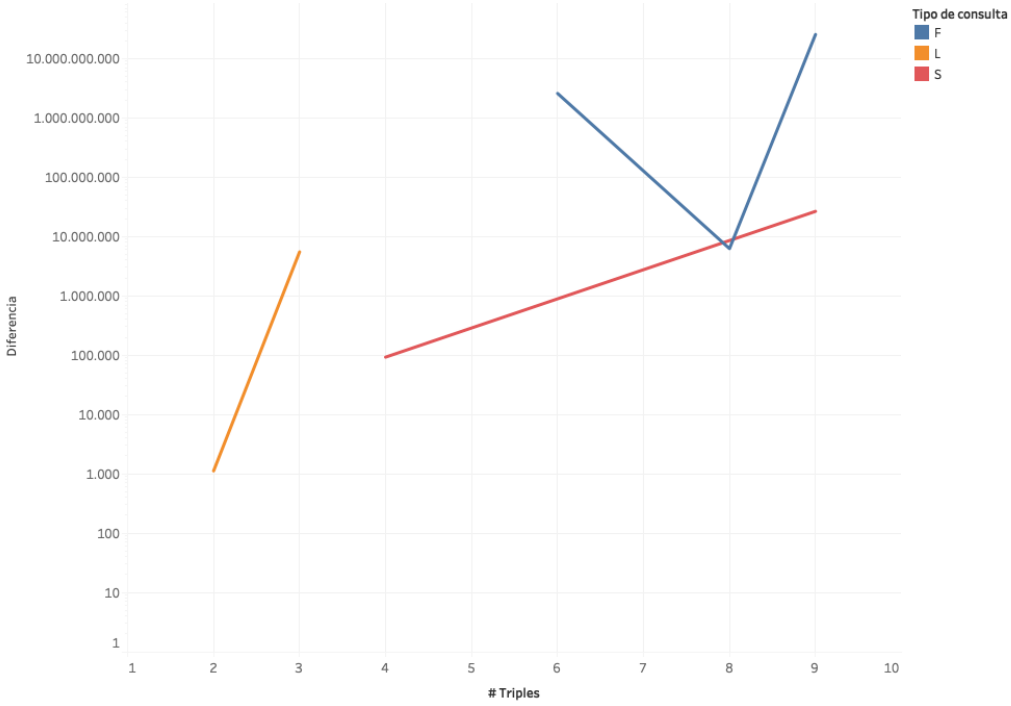


Figura 20: Diferencias de los tipos de consultas según el número de triples

CAPÍTULO 5
CONCLUSIONES

En esta memoria se realizó un análisis de múltiples técnicas de privacidad logrando, con éxito, un primer acercamiento de privacidad diferencial al modelo de datos RDF con el desarrollo de un software basado en la implementación de este método propuesto en *Towars Practical Differential Privacy for SQL Queries* [17]. Al momento de realizar los *tests* descritos en la sección anterior los resultados obtenidos por la solución implementada son adecuados.

Posteriormente, se pudo comprobar que importantes compañías dentro del mundo como *Apple* [2], *Google* [7] y *Microsoft* [6] utilizan este método para asegurar la privacidad de la información de sus usuarios. Ello, permite afirmar, con cierta seguridad, que el método de anonimización de datos escogido para el desarrollo de esta memoria está vigente y se continua investigando su desarrollo, hoy en día.

5.1. Objetivos

Se logró cumplir con los objetivos especificados para esta memoria, en cuanto a:

1. Se hizo un análisis respecto a múltiples técnicas y métodos para asegurar la privacidad de los datos, revisando los problemas asociados a cada uno de ellos, hasta llegar al método de privacidad diferencial.
2. Se elaboró una completa contextualización respecto a cómo funciona el método de privacidad diferencial, determinando la dificultad de su aplicación a modelos de *machine learning*. También se pudo identificar que el mayor obstáculo de dicho método radica en la elección del presupuesto de privacidad.
3. Se analizó la solución propuesta por [17] para poder realizar la implementación en RDF
4. Se desarrolló la solución propuesta adaptando la solución para SQL a NoSQL.
5. Se efectuaron las pruebas necesarias para validar la solución.

5.2. Resultados

Como se puede ver en los resultados reflejados en la figura 20, en la mayor parte de los casos, a medida que aumenta la cantidad de triples dentro de la consulta, también lo hace la sensibilidad de ésta. Esto se debe a que, a medida que aumenta el número de triples, también lo hace el número de JOINs, demostrado en la figura 13.

Existen casos en donde no se da esta relación, como se puede observar para la consulta

F con 8 triples en la figura 20. Ello, ocurre debido a que en el cálculo de la sensibilidad se encuentra el valor de la máxima frecuencia involucrada, por lo que si una consulta tiene valores de máxima frecuencia muy elevados y baja cantidad de JOIN, aún puede tener una alta sensibilidad.

Como se puede observar en las figuras 17, 18 y 19, a medida que el presupuesto disminuye, la privacidad aumenta, lo que cumple con lo expuesto a lo largo de esta memoria y que de igual manera se puede observar en la figura 13. Por tanto, es factible implementar una monetización de los datos privados utilizando esta solución.

No se puede concluir si existe una relación entre los resultados de una consulta y el tipo de ella, ya que las consultas de tipo L tienen una cantidad menor de triples en comparación con las S y las F, pero de todas formas se puede ver una ligera tendencia en la figura 20 que indica que las consultas tipo F tienen una mayor sensibilidad que las consultas tipo S, generando así una mayor diferencia entre el resultado real y el con ruido.

5.3. Trabajo futuro

- Extender la solución para incluir consultas en donde hayan triples que no participen en un JOIN.
- Al momento de elegir el presupuesto de privacidad siempre se presenta el problema de que cuándo éste es muy alto se pierde privacidad y, en cambio, cuándo es muy bajo, la utilidad de los datos es muy poca. En ese sentido, el buscar alguna manera de vincular el presupuesto con la sensibilidad de la consulta y la cantidad de JOINs que se está realizando es sumamente importante.
- A pesar que en [17] se demuestra que más de tres cuartas partes de las consultas con JOIN son EQUIJOIN, sigue quedando como trabajo futuro la inclusión de consultas con NON-EQUIJOINs. Sin embargo, según el mismo articulo, ésto no es una tarea trivial, ya que para lograrlo es necesario saber todos los datos de las dos tablas que participan en el JOIN.
- A pesar de que el porcentaje mayoritario de las consultas sean *count*, trabajo futuro es la inclusión de otras funciones de agregación.

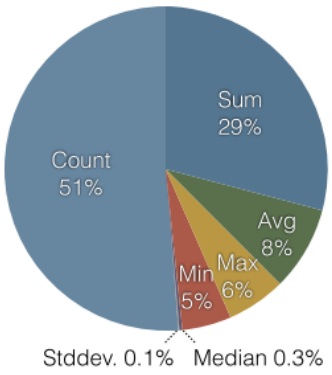


Figura 21: Uso de las distintas funciones de agregación [17]

- Dentro de la solución propuesta se pudo apreciar una vinculación entre el presupuesto de privacidad y el valor monetario que podría costar una consulta privatizada, pero queda como trabajo a futuro el plantear, al respecto, un modelo de negocios completo donde se pueda vincular la sensibilidad de la consulta, debido a la cantidad de JOINs presentes, con el presupuesto de privacidad y un valor monetario en concreto.

REFERENCIAS BIBLIOGRÁFICAS

[1] Güneş Aluç, Olaf Hartig, M. Tamer Özsu, and Khuzaima Daudjee. Diversified stress testing of rdf data management systems. In Peter Mika, Tania Tudorache, Abraham Bernstein, Chris Welty, Craig Knoblock, Denny Vrandečić, Paul Groth, Natasha Noy, Krzysztof Janowicz, and Carole Goble, editors, *The Semantic Web – ISWC 2014*, pages 197–212, Cham, 2014. Springer International Publishing.

[2] Apple. Differential Privacy Overview.

[3] V Devedzic. Chapter 1 : Introduction To Chapter 1 : Introduction To. *Semantic Web and Education*, 5(Ccc):1–12, 2006.

[4] Martin Duerst and Michel Suignard. Internationalized Resource Identifiers (IRIs). https://www.w3.org/International/iri-edit/draft-duerst-iri.html, 2004.
Visitado: 09/10/18.

[5] Cynthia Dwork. The promise of differential privacy: A tutorial on algorithmic techniques. *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 1–2, 2011.

[6] Cynthia Dwork and Sergey Yekhanin. Database Privacy. https://www.microsoft.com/en-us/research/project/database-privacy.
Visitado: 24/10/18.

[7] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 1054–1067, 2014.

[8] Keith D. Foote. A Review of Different Database Types: Relational versus Non-Relational - DATAVERSITY. http://www.dataversity.net/review-pros-cons-different-databases-relational-versus-non-relational/.

Visitado: 12/10/18.

[9] Apache Software Foundation. Apache Jena. https://jena.apache.org.
Visitado: 14/10/18.

[10] Matt Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in Pharmacogenetics: An End-to-End Case Study of Personalized Warfarin Dosing. *Proceedings of the 23rd USENIX Security Symposium*, pages 17–32, 2014.

[11] Johann Christoph Freytag. Privacy, kanonymity, and differential privacy, June 2017.

[12] Mikhail Galkin, Kemele M. Endris, Maribel Acosta, Diego Collarana, Maria-Esther Vidal, and Sören Auer. SMJoin. *Proceedings of the 13th International Conference on Semantic Systems - Semantics2017*, (September):104–111, 2017.

[13] Kendall Grant Clark. RDF Data Access Use Cases and Requirements. https://www.w3.org/TR/rdf-dawg-uc/.
Visitado: 09/10/18.

[14] Matthew Green. What is Differential Privacy? – A Few Thoughts on Cryptographic Engineering. https://blog.cryptographyengineering.com/2016/06/15/what-is-differential-privacy/, 2016.

[15] Bill Howe. Differential Privacy Defined - Privacy and Ethics | Coursera. https://www.coursera.org/learn/data-results/lecture/phj4C/differential-privacy-defined.
Visitado: 09/04/18.

[16] IAPP. What is Privacy. https://iapp.org/about/what-is-privacy/.
Visitado: 10/10/18.

[17] Noah Johnson, Joseph P. Near, and Dawn Song. Towards practical differential privacy for sql queries. *Proc. VLDB Endow.*, 11(5):526–539, January 2018.

[18] Murat Kantarcioglu. Other Privacy Definitions: l-diversity and t-closeness, 2009.

[19] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. L -diversity. *ACM Transactions on Knowledge Discovery from Data*, 1(1):3–es, 2007.

[20] Frank McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. *Commun. ACM*, 53(9):89–97, September 2010.

[21] Neo4j. What Is a Graph Database and Property Graph | Neo4j. https://neo4j.com/developer/graph-database/.
Visitado: 12/10/18.

[22] Rachana Nget, Yang Cao, and Masatoshi Yoshikawa. How to balance privacy and money through pricing mechanism in personal data market. *CoRR*, abs/1705.02982, 2017.

[23] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing - STOC '07*, page 75, 2007.

[24] Supriya S Pore and Swalaya B Pawar. Comparative Study of SQL & NoSQL Databases. *Ijarcet*, 4(5):1747–1753, 2015.

[25] Eric Prud'hommeaux and Andy Seaborne. SPARQL Query Language for RDF. https://www.w3.org/TR/rdf-sparql-query/, 2007.
Visitado: 09/10/18.

[26] rdfhdt.org. What is HDT - RDF HDT. http://www.rdfhdt.org/what-is-hdt/. Visitado: 13/10/18.

[27] Charles Roe. ACID vs. BASE: The Shifting pH of Database Transaction Processing - DATAVERSITY. http://www.dataversity.net/acid-vs-base-the-shifting-ph-of-database-transaction-processing/, 2012.
Visitado: 12/10/18.

[28] Standard Specification and San Francisco. TPC Benchmark TM H Standard Specification Revision 2.17.3 TPC BENCHMARK TM H. pages 1–137, 1993.

[29] Latanya Sweeney. k-ANONYMITY: A MODEL FOR PROTECTING PRIVACY. *International Journal on Uncertainty*, 10(5):557–570, 2002.

[30] Unicode Consortium. The Unicode standard. https://www.w3.org/TR/rdf-concepts/, 2000.
Visitado: 09/10/18.

[31] Xiaokui Xiao and Yufei Tao. Anatomy: Simple and effective privacy preservation. *Proceedings of the 32nd international conference on Very Large Database, ACM*, pages 150, 139, 2006.

[32] Xiaokui Xiao and Yufei Tao. Anatomy: Simple and effective privacy preservation. In *Proceedings of the 32Nd International Conference on Very Large Data Bases*, VLDB '06, pages 139–150. VLDB Endowment, 2006.