Bases de Datos Negativas

TAI 2

Gustavo González

2008-09-21

Índice

Introducción - Representación Negativa de la Información y Bases de Datos Negativas

Con el advenimiento de computadoras y dispositivos de almacenamiento digital, la cantidad de información que está siendo colectada y que tiene el potencial de ser explotada ha crecido enormemente. Datos son recolectados de cada fuente imaginable, abarcando desde lo científico (secuencias de genomas, series de colisiones de partículas,..) a lo sociológico, donde la información sobre los individuos tales como sus datos demográficos, preferencias, y hábitos de consumo, son recolectados y estudiados.

La naturaleza y el volumen de los datos impone un nuevo desafío en términos de como éstos deben ser utilizados. Central a esto, es la cuestión de como deben ser representados, dado que la representación de los datos tiene un impacto inmenso sobre como éstos pueden ser utilizados.

Consideremos el caso de los algoritmos evolutivos donde la representación de los individuos (posibles soluciones al problema) determina en gran parte cuales son los operadores genéticos aplicables.

Otro ejemplo de la importancia de la representación de los datos proviene de la criptografía, donde el objetivo es encontrar representaciones de datos a partir de las cuales la extracción de información útil o significativa solo sea posible con el conocimiento de la clave utilizada para generar la representación.

Proteger el acceso a información sensible y restringir el tipo de inferencias que se pueden derivar a partir de la misma es una preocupación que debe ser encarada continuamente en un mundo donde las demandas sobre la disponibilidad de datos, y los criterios para su confidencialidad evolucionan. Las tecnologías actuales de encriptación (para los datos) y la restricción de consultas (para controlar el acceso a los datos) proporcionan confidencialidad, pero ninguna de estas soluciones es apropiada para todas las aplicaciones.

En el caso de la encriptación, la búsqueda de registros se ve afectada; en el caso de la restricción de consultas, los registros individuales son vulnerables a ataques internos. Además muchas de las soluciones actuales se basan en el mismo conjunto de presupuestos, asumen que la factorización entera es un problema difícil.

En el presente trabajo se presenta una nueva manera de representar datos que pretende resolver algunos de estos problemas y proveer un punto de partida para el diseño de nuevas aplicaciones.

Un posible escenario involucra una base de datos de registros personales, a la cual una entidad foránea podría desear consultar, para hacer verificaciones. Es deseable tener un base de datos que restrinja el tipo de consultas, no permita inspecciones arbitrarias a los datos, y que pueda ser actualizada sin revelar la naturaleza de los cambios a un oponente.

Bases de Datos Negativas

En una base de datos negativa (NDB) la imagen negativa del conjunto de registros es representada en vez de los registros mismos. Podemos asumir que los registros en cuestión consisten en cadenas de longitud L, definidos sobre un alfabeto binario {0,1}. El conjunto de todas las posibles cadenas de esta forma, el universo U, es particionado en dos subconjuntos disjuntos:

- DB : representando el conjunto de registros de contienen la información de interés
- U-DB: denotando el conjunto de todas las cadenas que no están en DB.

Se asume que DB no está comprimida, pero se permite que U-DB sea almacenada en una forma comprimida llamada NDB. DB es la base de datos positiva y NDB es la base de datos negativa.

Desde un punto de vista lógico, cualquiera de las bases de datos, DB o NDB, es suficiente para responder cuestiones acerca de DB. Pero las diferentes representaciones poseen distintas ventajas. Por ejemplo, en una base de datos positiva, la inspección de un solo registro provee información útil o significativa. Por otro lado la inspección de un solo registro negativo revela poca o ninguna información sobre el contenido original de la base de datos. Dado que los registros positivos nunca son almacenados explícitamente, una representación negativa hace que sea mucho más difícil utilizar los datos en formas no deseadas.

Similarmente, dependiendo de la representación específica de la NDB, la eficiencia de ciertos tipos de operaciones es muy diferente a la eficiencia de las mismas operaciones aplicadas a una DB. Algunas aplicaciones pueden verse beneficiadas con este cambio de perspectiva. La mayoría de las aplicaciones busca, recuperar información acerca de DB tan eficientemente como sea posible. Pero en situaciones donde la privacidad es una preocupación, puede ser útil adoptar un esquema en el cual ciertas consultas sean eficientes y otras sean demostradamente ineficientes.

Consideremos, por ejemplo, una lista de vigilancia que está disponible a agentes de aerolíneas. Es deseable que estos agentes tengan la habilidad de verificar si un nombre dado figura en la lista, pero al mismo tiempo que no tengan la habilidad de "ver" el contenido de la lista, o siquiera saber su tamaño. O imaginemos dos entidades que tienen la necesidad de determinar en forma privada la intersección de los conjuntos de datos de su propiedad. Por ejemplo dos o más entidades pueden querer determinar cuales de un conjunto posible de ítems (transacciones) tienen en común, sin revelar la totalidad de los contenidos de sus respectivas bases de datos o su cardinalidad.

Representación

Con el fin de crear una base de datos NDB que tenga un tamaño razonable, es necesario comprimir la información contenida en U-DB. Con este fin se introduce un símbolo adicional, conocido como "no importa", escrito como *. Por lo tanto los registros de la NDB serán cadenas de longitud L sobre el alfabeto {0,1,*}.

El símbolo "no-importa" tiene la interpretación usual, haciendo coincidir con un 0 o con un 1 en la posición donde * aparece. Las posiciones de una cadena que han sido establecidas a cero o a uno, tienen en nombre de posiciones definidas o especificadas, y las posiciones donde aparece un * son posiciones no-especificadas. Con este nuevo símbolo, grandes subconjuntos de U-DB pueden ser representados con solo unas cuantas entradas.

DB	$U-DB$	NDB
000	001	001
100	010	*1*
101	011	
	110	
	111	

DB	$U-DB$	NDB
0001	0000	11**
0100	0010	001*
1000	0011	011*
1011	0101	0000
	0110	0101
	0111	1001
	1001	1010
	1010	
	1100	
	1101	
	1110	
	1111	

La convención es que una cadena binaria x está en DB si y solo si x no coincide con ninguna de las entradas de NDB. Esta condición se cumple solo si por cada entrada de la NDB, la cadena s no concuerda con la entrada en al menos una posición definida.

Las consultas también son representadas como cadenas sobre el mismo alfabeto. Una cadena, Q, consistente solo de posiciones definidas- solo 0s y 1s- es interpretada como “¿Está Q en DB?”, y nos referimos a la misma como una consulta de membresía simple o consulta de autenticación. Responder a tal consulta requiere examinar NDB, lo cual puede hacerse en tiempo proporcional a |NDB|.

Por otro lado se ha demostrado un mapeo eficiente entre formulas de satisfacibilidad booleana y NDBs y con esto se muestra que el problema de revertir una NDB – o sea recuperar la DB original – es un problema NP-hard, determinar el tamaño de la DB o incluso saber si está vacía o no es un problema NP-hard también. Consecuentemente, responder consultas con un número arbitrario de símbolos * es también intratable.

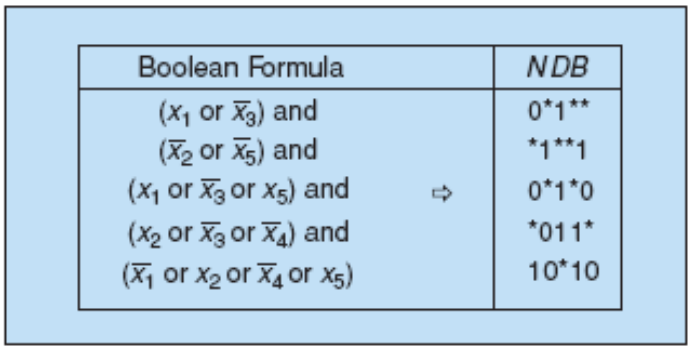
A la fecha, existen tres algoritmos básicos para crear NDBs:

1. Prefix Algorithm: determinístico y siempre genera una NDB que es fácil de revertir.
2. Randomized Algorithm: no-determinístico y produce NDBs difíciles de revertir.
3. On-line Algorithms: diseñados para actualizar NDBs (insertar y eliminar cadenas)

SAT es reducible a NDB

Los siguientes son algunos resultados concernientes a la complejidad de extraer información de una NDB dada:

- Reconstrucción (recuperación de todos los registros) de DB a partir de NDB es NP-Hard
- Saber si DB está vacía es NP-Hard
- Saber si DB no esta vacía es NP-Hard
- Dadas 2 NDBs NDB1 y NDB2 saber si las dos representan distintas DBs es NP-Hard



La figura muestra un mapeo de SAT a una NDB: una fórmula booleana expresada en forma normal conjuntiva es transformada a una base de datos negativa al traducir cada cláusula a un registro negativo.

Cada registro tiene una posición para cada variable en la formula (posición 1 corresponde a variable x1, etc.) y su valor es 1 si la variable aparece negada en la cláusula, 0 si aparece sin negar, * caso contrario. En este ejemplo la asignación {x1 = FALSE, x2 = TRUE, x3 = TRUE, x4 = TRUE, x5 = FALSE} no satisface la formula y la correspondiente cadena 01110 es macheada por NDB (lo que implica que no está en DB), mientras que la asignación {x1=TRUE, x2=FALSE, x3=TRUE, x4=FALSE, x5=FALSE} satisface la fórmula booleana y la cadena 10100 no está en NDB (y por lo tanto está en DB).

Aún cuando ha sido demostrado que una base de datos negativa es difícil de revertir en general, muchas instancias son fáciles, de hecho el algoritmo de prefijo (el primer algoritmo diseñado para generar bases de datos negativas) siempre retorna NDBs fácilmente reversibles. El desafío consiste en diseñar algoritmos que generen bases de datos negativas difíciles de revertir en promedio. Esto ha sido logrado adaptando algoritmos usados para crear formulas SAT difíciles.

Consultas a una NDB

Usando la representación expuesta anteriormente, una base de datos negativa, es un conjunto de cadenas definidas sobre {0,1,*}^L. Consultas a tales bases de datos también son expresadas como cadenas definidas sobre el mismo alfabeto y tienen la forma "¿Es Q un miembro de BD?". Más precisamente estas consultas simplemente preguntan por la membresía de una cadena x a un conjunto BD. Consultas tales como "¿Cuales son los ingenieros en BD?" pueden ser construidas usando consultas de membresía simples, tales como se discute más adelante.

Si una cadena X esta compuesta solo de 0s y 1s (o sea no contiene el símbolo '*'), entonces determinar la membresía es sencillo dado que solo requiere verificar si X encaja con alguna de las cadenas en NDB.

Por otro lado, si X contiene un numero arbitrario de posiciones sin especificar (contiene '*'s), responder a la consulta es equivalente a preguntar si una formula booleana correspondiente es satisfacible cuando un numero arbitrario de sus variables tiene valores preasignados. Este sigue siendo un problema NP-hard. Esto marca un contraste con una base de datos positiva DB, donde los registros son almacenados explícitamente y responder a consultas de esta naturaleza toma un tiempo proporcional al tamaño de DB.

Por ejemplo, consideremos una base de datos DB con registros de la forma <Nombre, Dirección, Profesión> y la consulta Q "¿Cuales son los ingenieros en DB?", la cual puede escribirse como una cadena sobre {0,1,*}, donde el campo profesión se establece a la codificación binaria de "Ingeniero" y las demás posiciones se establecen a '*'.

Si la consulta Q es enviada a DB (la base de datos positiva), retornará solo las cadenas que concuerden con el campo especificado aún cuando Q puede en realidad representar un número exponencial de cadenas.

Pero si la consulta Q es remitida a NDB (la base de datos negativa), será necesario encontrar cuales de todas las posibles cadenas de longitud l cuyas posiciones definidas corresponden a "Ingeniero" no están en NDB y retornarlas. Lograr esto es un problema NP-hard para una elección arbitraria de posiciones definidas. Pero es posible construir NDBs con estructuras específicas para las cuales consultas más complejas, puedan ejecutarse en forma más eficiente. Intuitivamente, lo que hace que algunas consultas sean ineficientes no es el tamaño de NDB, dado que es solo polinomialmente más grande que DB, sino que un solo registro, es representado por varias entradas en NDB y que una sola entrada NDB representa múltiples registros. Esto hace muy difícil determinar incluso si es que existe algún ingeniero en DB.

Bases de Datos Negativas Difíciles de Revertir

Se ha demostrado que revertir una NDB es un problema NP-Hard, pero, dado que esta es una propiedad del peor caso, se presenta el desafío crear instancias que sean difíciles de revertir en la práctica.

Para lograr esto, se toma ventaja de la relación existente entre bases de datos negativas y el problema de satisfacibilidad booleana (SAT), se

usan las investigaciones llevadas a cabo para generar instancias SAT difíciles como base para crear NDBs difíciles de revertir.

Usando Formulas SAT como un Modelo para Bases de Datos Negativas

Dado que el objetivo es crear NDBs difíciles de revertir, las investigaciones se centraron en adaptar algoritmos existentes para la generación de formulas SAT. El objetivo de estos algoritmos es crear una formula que se sabe que es satisfacible, pero que los resolvedores SAT no sean capaces de resolver.

El método consiste en tomar un asignación A (un cadena binaria que representa los valores de verdad para las variables en la formula), y crear una fórmula satisfacible por A, además de ser satisfacible por otras asignaciones desconocidas.
Dada una asignación A, el algoritmo genera aleatoriamente cláusulas con $t > 0$ literales satisfacibles por A con probabilidad proporcional a q^t para $q < 1$ (q es un parámetro específico del algoritmo). El propósito del método es balancear la distribución de literales de forma tal a hacer que las fórmulas sean indistinguibles entre sí. El proceso retorna una colección de cláusulas, todas satisfacibles por A, las cuales pueden ser inmediatamente transformadas en una base de datos negativa.

Dada una base de datos de tamaño a lo más uno (un solo registro), conteniendo una cadena binaria A de longitud L, se crea una base de datos negativa (NDB) con las siguientes propiedades:

1. Cada entrada en la base de datos negativa tiene exactamente tres bits especificados
2. A no encaja con ninguna de las entradas de NDB.
3. Dada una cadena de L-bits, es fácil verificar si pertenece a NDB o no (en tiempo proporcional al tamaño de NDB).
4. El tamaño de NDB es lineal en términos de la longitud de A (o sea de L)
5. El tamaño de NDB es independiente de los contenidos de BD.
6. A es “casi” la “única” cadena no encajada por los patrones de NDB. Las otras cadenas están a una distancia Hamming cercana de A.
7. La base de datos negativa NDB es muy difícil de revertir, ningún método conocido de búsqueda puede descubrir A en un tiempo razonable (dado que el numero de bits en A sea mayor que 1000)

Las propiedades de 1 a 5 se siguen de la correspondencia entre bases de datos negativas y formulas 3-SAT y las características del algoritmo.

Bases de Datos Negativas de Múltiples Registros

En la sección anterior se explicaba como crear una representación negativa de un BD con cero entradas o una entrada, ahora delinea a grandes rasgos como esto se puede extender a DBs de un tamaño arbitrario.

El esquema anterior puede ser usado para generar una representación negativa de cualquier conjunto de cadenas DB al crear una NBDi para cada cadena Ai en DB, o sea, cada registro de la NDB resultante es una base de datos negativa. Es importante recalcar que todas las NDBAi son del

mismo tamaño (siendo indistinguibles por esta métrica) y que algunas pueden representar el conjunto vacío.

DB	NDB_0	NDB_4	NDB_5	$NDB_\emptyset$
000	*1*	*1*	0**	0**
100	1**	**1	**0	*1*
101	0*1	000	*11	10*

La figura muestra una posible DB con posibles NDBAi (NDB0 representa el conjunto vacío). La NDB final es la colección de todas las NDBAi.

Este método de generación tiene un inconveniente dado que el tamaño de la DB subyacente puede ser estimado por el número de registros (NDBAi) en la NDB, esto nos da una cota dado que NDB puede contener cualquier número de registros que representan el conjunto vacío.

Por otro lado una NDB creada de esta manera es mucho más fácil de actualizar: insertar una cadena Ai en DB se implementa mediante la búsqueda y eliminación de los registros de NDB que representen Ai. Borrar Ai de DB se implementa mediante la generación de su correspondiente NDBAi y la concatenación de NDBAi como un registro a NDB.

El resultado es una base de datos en la cual las actualizaciones toman tiempo lineal en |DB|, además la naturaleza de las actualizaciones permanecerá ambigua para un observador dado que un registro negativo puede representar el conjunto vacío y varios registros negativos pueden representar el mismo registro positivo (entrada de DB).

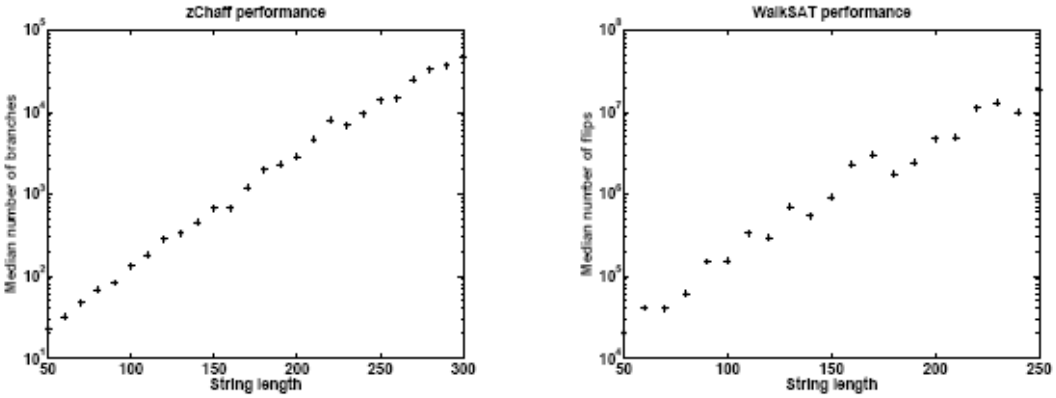
¿Qué tan difícil es revertir una NDB?

Para ilustrar cuan difícil es revertir estas NDBs, se realizaron experimentos en los cuales se creaban NDBs con longitud de registro variable (de 50 a 300 bits)
Luego se intentaba extraer registros utilizando afamados resolvedores SAT.

Existen dos tipos de resolvedores:

Resolvedores Completos: Buscan el espacio de posibles soluciones en forma exhaustiva.

Resolvedores incompletos: Exploran solo una fracción del espacio de soluciones y pueden manejar instancias mucho más grandes (en términos del número de variables), pero a diferencia de los resolvedores completos, cuando éstos no encuentran una solución, esto no significa que no exista una solución.

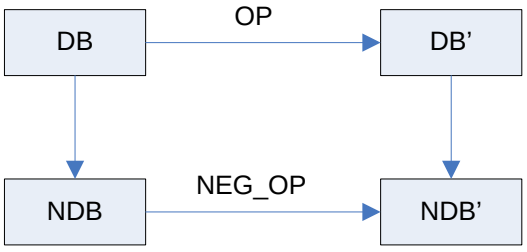


Las figuras muestran los resultados para el resolvedor completo zChaff (zChaff es a menudo el campeón de la competencia SAT anual) y para walkSAT, un resolvedor incompleto muy conocido. Los experimentos muestran que tanto zChaff como WalkSat encuentran una entrada en DB en tiempo exponencial en la longitud de la cadena l (longitud del registro). Hay que tener en cuenta que una reversión completa de una NDB, hallar todos los registros de DB, implicaría ejecutar el resolvedor |DB|+1 veces, o sea tantas veces como registros tenga DB, más una vez para establecer que ya no quedan más registros por hallar.

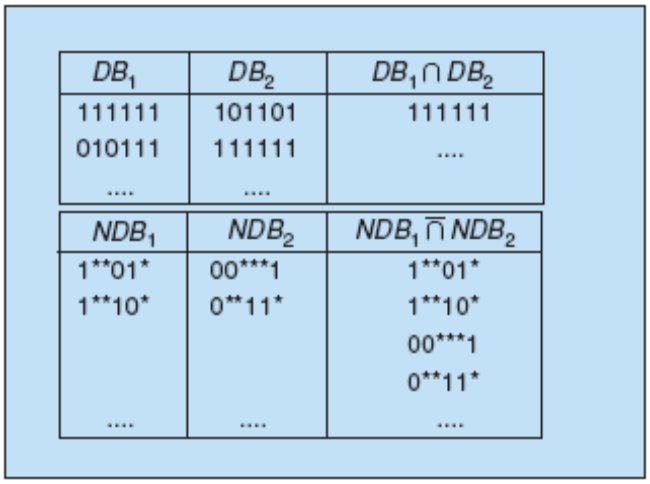
Además se hicieron experimentos con 100 NDBs con registros de longitud 1000, los resolvedores zChaff, WalkSAT, y otros dos resolvedores: SATz y SP (el primero es un resolvedor completo y el segundo es incompleto). Ninguna entrada de DB fue encontrada por ninguno de los resolvedores, dado que los resolvedores incompletos terminaron sin hallar ninguna solución, y los resolvedores completos agotaron la memoria.

Algebra Negativa

Cuando se usan bases de datos negativas para representación de datos, las mismas son manipuladas de forma tal que la NDB resultante, cuando revertida, concuerde con el resultado de efectuar la operación correspondiente en su contraparte positiva.



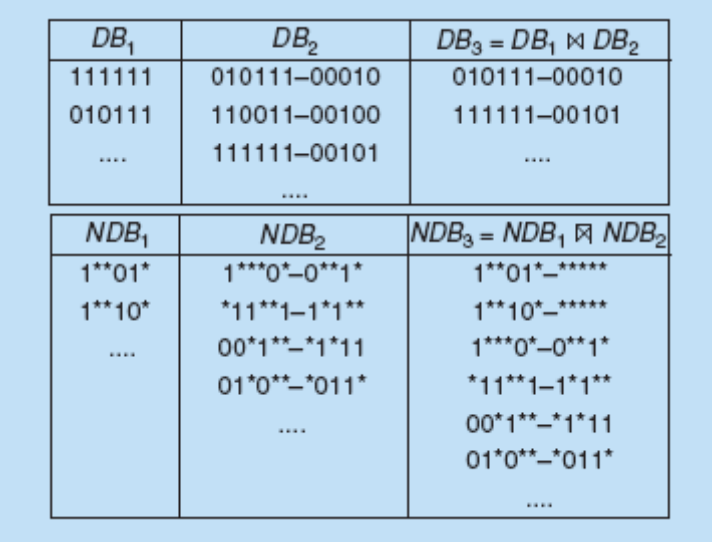
Por ejemplo, la operación Unión-Negativa recibe como entrada NDB1 y NDB2 y retorna una base de datos negativa NDB3 que concuerda exactamente con las cadenas que la base de datos negativa creada de la unión de DB1 y DB2, o sea, la reversión de NDB3 es DB1 U DB2.



DB_1	DB_2	$DB_1 \cap DB_2$
111111	101101	111111
010111	111111	....
....	....	

NDB_1	NDB_2	$NDB_1 \overline{\cap} NDB_2$
1**01*	00***1	1**01*
1**10*	0**11*	1**10*
		00***1
		0**11*
....	....	....

La figura arriba muestra la intersección de dos bases de datos DB1 y DB2 y la correspondiente operación (Intersección negativa) de NDB1 y NDB2.



DB_1	DB_2	$DB_3 = DB_1 \bowtie DB_2$
111111	010111–00010	010111–00010
010111	110011–00100	111111–00101
....	111111–00101	....
	....	

NDB_1	NDB_2	$NDB_3 = NDB_1 \overline{\bowtie} NDB_2$
1**01*	1***0*–0**1*	1**01*–*****
1**10*	*11**1–1*1**	1**10*–*****
....	00*1**–*1*11	1***0*–0**1*
	01*0**–*011*	*11**1–1*1**
	....	00*1**–*1*11
		01*0**–*011*
		....

La figura arriba muestra el Join de DB1 y DB2 y la operación equivalente sobre sus bases de datos negativas.

Una consecuencia de trabajar con la imagen negativa de un conjunto es que la complejidad de algunas operaciones se invierte. Por ejemplo, el producto cartesiano o join de dos DBs es, por lo general, cuadrático, mientras que el producto cartesiano negativo y el join negativo son lineales; la unión de dos DBs es lineal, la unión negativa es cuadrática.

Aplicaciones de Bases de Datos Negativas

Lo interesante de las NDBs difíciles de revertir es que, a pesar del hecho de que DB no puede ser fácilmente deducida, las mismas pueden ser usadas para determinar en forma eficiente la presencia o ausencia de cadenas específicas. Además por la manera en que se distribuye la información entre los registros negativos de una NDB, la única manera de saber si una cadena pertenece o no a DB es parseando toda la NDB.

Si juntamos esto al hecho de que existen esquemas para distribuir una NDB en varias NDBi, donde para saber si una cadena pertenece o no a DB tendremos que consultar todas las NDBi, tendremos que si un oponente se apodera de una copia de una NDBi, esto le servirá de muy poco, dado que para acceder a cualquier registro positivo de DB, tendrá que consultar todas las NDBi, en otras palabras, para acceder a cualquier registro de DB, el oponente tendría que tener en su poder toda la NDB.

Algunos ejemplos de posibles aplicaciones de NDBs:

Por ejemplo, una empresa desea mantener una lista de números de tarjetas de crédito que han sido involucradas en actividades sospechosas. Una NDB difícil de revertir puede ser creada de forma tal que prevenga que un oponente tenga acceso a la lista completa de números y que la capacidad de validación de números específicos sea preservada.

Supongamos que el gobierno G solicita a dos bancos B1 y B2 la lista de clientes que tienen en común, pero ni B1 ni B2 desean que el otro banco o que el gobierno puedan ver su lista completa de clientes. B1 y B2 generan las NDBs (difíciles de revertir) de sus respectivas bases de datos de clientes, computan la intersección negativa NDB3 de NDB1 y NDB2 (lo cual consiste en simplemente concatenar las dos NDBs), a continuación cualquiera de los dos bancos puede hallar la lista de nombres simplemente consultando a NDB3 por cada uno de sus clientes. Dado que NDB3 es difícil de revertir, podemos asumir que ni B1 ni B2 tendrán la capacidad computacional suficiente requerida para revertirla. De esta manera se computa la intersección de ambos conjuntos, sin exponer información a la competencia, y sin comprometer la privacidad de los demás clientes.

Temas Relacionados

Existen muchos temas relacionados con bases de datos negativas. Entre los más relevantes están las técnicas para proteger los contenidos de bases de datos - encriptación de bases de datos, zero-knowledge sets, privacy-preserving data mining, y restricción de consultas – sistemas de seguridad basados en problemas NP-Hard y one-way functions.

Algunos enfoques para protección de contenidos de una base de datos involucran el uso de métodos criptográficos, por ejemplo, mediante la encriptación de cada registro con su propia clave. Zero-knowledge sets proveen una primitiva para construir bases de datos que tienen muchas de las mismas propiedades que las bases de datos negativas, por ejemplo la restricción de consultas a simples preguntas de pertenencia a un conjunto. Pero, las mismas están basadas en métodos que se cree que son criptográficamente seguros, requieren una entidad de control para responder a consultas, y son difíciles de actualizar.

En privacy-preserving data mining, la meta es proteger la confidencialidad de registros individuales y la vez proveer ciertas operaciones de minería de datos. Por ejemplo, distribuciones estadísticas son preservadas, pero los detalles de los registros individuales son ocultados. Las bases de datos negativas contrastan con

esto, en que soportan simples consultas de membresía en forma eficiente, pero consultas de alto nivel pueden ser costosas.

Las bases de datos negativas también tienen relación con la restricción de consultas, donde se diseña un lenguaje que soporte solo consultas que tengan ciertas propiedades deseadas. A pesar de que la restricción de consultas controla el acceso de usuarios externos a los datos, no puede protegerlos de un usuario interno con privilegios.

Sistemas criptográficos basados en problemas NP-completos han sido previamente estudiados, un ejemplo es el sistema de Merkle-Hellman, el cual está basado en el problema knapsack general (problema de la mochila). Estos sistemas se basan en una serie de trucos para ocultar la existencia de un “trapdoor” que permite la recuperación de la información oculta en forma eficiente, pero casi todos los sistemas criptográficos basados en el knapsack problem han sido rotos. Las bases de datos negativas no tienen trapdoors.

Funciones one-way y acumuladores one-way toman una cadena o un conjunto de cadenas y producen un digest a partir del cual es difícil obtener la cadena original. Una diferencia entre éstos métodos y las bases de datos negativas es que la salida de una función one-way es a menudo compacta, y que el mensaje que codifica tiene una única representación (lo cual hace que sea fácil verificar si una cadena corresponde a un digest determinado).

A medida que crecen la disponibilidad de datos y los medios para accederlos, también lo hacen nuestros requerimientos de seguridad y privacidad. No existe una solución para todas las demandas. Las bases de datos negativas NDBs, con sus características únicas, se suman a la amplia gama de herramientas disponibles para lograr la seguridad de datos y asegurar la privacidad de las personas.

Conclusión

Una NDB es una forma de compacta de representar la imagen negativa de un conjunto de datos, que cuenta con propiedades muy interesantes, sobre todo para el campo de seguridad de datos y privacidad.

La relación entre NDBs y SAT abrió paso al desarrollo de NDBs demostradamente difíciles de revertir, pero que soportan consultas simples en forma eficiente.

La introducción de bases de datos negativas para seguridad es reciente, y todavía hay muchas cuestiones que deben resolverse, antes de que las NDBs se vuelvan ampliamente aplicables. Por ejemplo, algoritmos eficientes para la creación de bases de datos negativas que representen millones de entradas positivas serán necesarios para aplicaciones futuras. También, entre los atributos que hacen que las NDBs sean atractivas es la cantidad de operaciones que pueden ser aplicadas a los datos sin conocer los datos.

La mayor parte del trabajo sobre bases de datos negativas aplicadas a seguridad se basa en bases de datos negativas difíciles de revertir, pero, hay otras propiedades de las NDBs que pueden ser relevantes para la seguridad de los datos. Por ejemplo, la forma en que la información

sobre el conjunto de datos positivo se distribuye entre las entradas negativas: examinar un pequeño subconjunto de la NDB provee solo información limitada acerca de la identidad de cualquier entrada de la DB, toda la NDB debe estar disponible para poder recuperar DB y toda la NDB debe ser parseada para estar seguro de que una entrada dada está en DB.

Referencias:

Everything That is Not Important: Negative Databases – IEEE Computational Inteligence, Mayo 2008

Protecting Data Privacy though Hard-to-Reverse Negative Databases

Enhacing Privacy through Negative Representations of Data – Fernando Esponda, Stephanie Forrest, Paul Helman.

A Relational Algebra for Negative Databases – Fernando Esponda, Eric D.Trias, Elena S. Ackley.

On-Line Negative Databases – Fernando Esponda, Elena S. Ackley, Stephanie Forrest, Paul Helman.

Negative Representations of Information – Fernando Esponda.