Universidad de Concepción
Dirección de Postgrado
Facultad de Ingeniería - Programa de Magíster en Ciencias de la Computación

IMPLEMENTACIÓN MODELO DE DATOS MULTIGRANULAR BASADO EN REGLAS DE INFERENCIAS

Tesis para optar al grado de
MAGÍSTER EN CIENCIAS DE LA COMPUTACIÓN

POR
Diego Gatica Romero
CONCEPCIÓN, CHILE
Junio, 2019

Profesor guía: Andrea Rodríguez
Departamento de Ingeniería Informática y Ciencias de la Computación
Facultad de Ingeniería
Universidad de Concepción

AGRADECIMIENTOS

A mi profesora guía Andrea por su disposición y ayuda durante la tesis.

A Palta por toda la ayuda brindada en el desarrollo de la tesis.

A Irri, Checho, Eva, Alexis, Charlie y a todos quienes me acompañaron durante el desarrollo de la tesis y ayudaron a que fuese una experiencia más agradable

En especial a Abigail por su cariño, compresión y apoyarme siempre que lo necesité.

Resumen

Frente al problema de poseer bases de datos con datos a distintos niveles de detalle (granularidad), el modelo multigranular presenta ventajas sobre el modelo relacional clásico. En un modelo relacional clásico, una instancia debe contener toda las información que caracterice la estructura granular a través de relaciones con claves foráneas. El modelo multigranular, por otro lado, crea una única estructura para representar las diversas granularidades y los gránulos que las componen. Una estructura multigranular permite que distintas instancias de bases de datos hagan referencia a esta estructura sin duplicación de información, y al mismo tiempo permite que instancias tengan distintos de niveles de granularidad que puedan ser integrados a través de esta estructura.

En este trabajo se presenta una implementación de un modelo multigranular que busca minimizar el espacio de almacenamiento junto con un tiempo de procesamiento comparable al tiempo que tendría procesar consultas en un modelo relacional clásico.

Índice general

Índice de cuadros

Índice de figuras

Capítulo 1

Introducción

Con la masificación de los sistemas de información se han incrementado los datos disponibles a procesar en los últimos años, lo cual ha generado la necesidad de desarrollar maneras eficientes para almacenar y procesar estos datos. Como respuesta a tal necesidad se han desarrollado diversos modelos de bases de datos, donde cada uno está orientado a satisfacer de manera eficiente diversas consultas enfocadas en ámbitos específicos. Tales paradigmas varían desde lo más tradicional, como el modelo relacional, hasta el esquema estrella utilizado en *data warehousing* [12, 13] o el modelo basado en grafos [8].

Actualmente, es bastante común tratar con información referenciada a una instancia de tiempo o a algún espacio físico, por lo que se ha vuelto indispensable desarrollar una forma eficiente, tanto en tiempo como en espacio, para responder consultas sobre datos que posean las características mencionadas anteriormente. Un modelo propuesto para abordar la representación de información con referencia espacial y temporal a distintos niveles de granularidad es el modelo de datos multigranular [4, 10, 11].

Un modelo de datos multigranular es uno en el cual sus datos se pueden encontrar en distintos niveles de detalle o granularidad [11], es decir, que los valores que puede tomar un atributo se organizan en una estructura de orden parcial que define el detalle de representación. Lo anterior presenta grandes desafíos al momento de establecer restricciones y formas de relacionar datos a distintos niveles de granularidad.

Considere, como ejemplo ilustrativo, una tabla que refleja información sobre las personas nacidas por mes a nivel de provincia de Chile y otra que contiene las personas nacidas por mes a nivel de región.

Provincia	Mes 2017	Personas nacidas
Arauco	Agosto 2017	A_1
Biobío	Agosto 2017	A_2
Concepción	Agosto 2017	A_3
Ñuble	Agosto 2017	A_4

Region	Mes 2017	Personas nacidas
O'Higgins	Agosto 2017	B_1
Maule	Agosto 2017	B_2
Biobío	Agosto 2017	B_3
Araucanía	Agosto 2017	B_4

Cada una de estas tablas refleja una granularidad; es decir, una representación de la información a un nivel específico. Cada granularidad está compuesta por gránulos, los cuales definimos informalmente como una división del dominio a representar en partes no superpuestas y que se usan como valores de referencia. En el ejemplo anterior, los atributos (columnas) Provincia y Región se refieren a dos granularidades y los valores de estos atributos corresponden a gránulos en estas granularidades.

De este caso resulta interesante el poder realizar consultas como el obtener la provincia perteneciente a la región de O'Higgins que posea la mayor cantidad de personas nacidas, o más allá, la región con la mayor cantidad de personas nacidas asumiendo que se posee la información a nivel de provincia.

Aunque existen diversos trabajos que han abordado el tema de la granularidad, el enfoque aquí propuesto se diferencia de los existentes principalmente en que soporta el uso de reglas de join, por ejemplo,

la regla de que una región esté formada por comunas que no se sobreponen entre sí. Además, permite tratar con información espacial sin recurrir, necesariamente, a atributos espaciales que almacenen geometrás y que requieren el uso de operadores geométricos para determinar relaciones espaciales con un aumento en el costo de procesamiento.

1.1. Hipótesis

El uso de un modelo de datos multigranular que utilice reglas de inferencias en bases de datos que poseen atributos espacio/temporal y que requieren información a distintos niveles de granularidad mejora el uso de espacio, y a su vez, mantiene un tiempo de consulta competitivo en comparación con una implementación en un modelo multigranular que almacena de manera explícita cada relación entre gránulos.

1.2. Objetivo general

Definir un esquema de datos y una implementación eficiente en espacio y tiempo para representar el modelo multigranular propuesto en [10].

1.3. Objetivos específicos

- Definir reglas de inferencia para las reglas básicas de *disjoint* y *subsumption* con sus respectivas negaciones lógicas con el fin de determinar la cantidad mínima de información a almacenar.
- Analizar el impacto de las relaciones bigranulares[1] y completitud en términos de la información mínima a guardar en la estructura multigranular.

[1] termino utilizado para referir la relación entre dos granularidades

- Diseñar los algoritmos para aplicar las inferencias propuestas sobre el modelo de datos.
- Desarrollar un esquema de datos para la representación de la estructura multigranular en base a lo desarrollado y analizado previamente.
- Implementar algoritmos para responder consultas de *disjoint* o *subsumption* entre gránulos utilizando reglas de inferencia definidas previamente en base al esquema de datos desarrollado.
- Evaluar el rendimiento de los algoritmos implementados en términos del espacio utilizado y el tiempo de respuesta en comparación con una implementación en un modelo relacional plano, es decir, un modelo donde la información de la granularidad y de relaciones entre gránulos es parte de la instancia de la base de datos con relaciones especificadas por claves foráneas, y por otro lado, una implementación basada en modelo multigranular sin hacer uso de métodos de inferencia, haciendo uso de datos creados artificialmente y de los datos del las divisiones censales del año 2017.

Este documento se organiza de la siguiente forma. Capítulo 2 hace una revisión sobre trabajos relacionados al problema de tratar información a distinto nivel de detalle y se proporcionan los detalles necesarios sobre el modelo multigranular desarrollado en [10] y [11]. El capítulo 3 detalla el trabajo realizado en cuanto al desarrollo de un modelo que utilice reglas de inferencia con el fin de minimizar la información almacenada. Capítulo 4 muestra los resultados obtenidos al aplicar lo desarrollado frente a diversas instancias. Finalmente, Capítulo 5 contiene las conclusiones y trabajo a futuro.

Capítulo 2

Revisión bibliográfica

2.1. Trabajo relacionado

En el mundo de las bases de datos, un modelo conocido y enfocado a responder consultas a distintos niveles de agregación es el modelo *datawarehouse* [13], el cual se caracteriza por guardar información completa al nivel de mayor detalle y luego precalcular agregaciones para las dimensiones de información definidas. En [12] se proponen extensiones del esquema de *datawarehouse* agregando el atributo de granularidad y modificando la tabla de hechos; esto con el fin de explotar la multigranularidad de los datos mas allá de lo que son capaces esquemas tales como el esquema *star* o el esquema *snowflake*, debido a que estos últimos requieren una estructura rígida de los niveles de detalle de los datos. Este esquema multigranular propuesto permite almacenar la información a distinto nivel de granularidad, sin embargo, no especifica relaciones a través de reglas de join o *disjoint* join, solo se basan en *subsumption* y está enfocado en datos que no varíen o sufran modificaciones una vez insertados.

En el contexto de manejo de datos temporales, en [2] se definen los siguientes conceptos básicos sobre granularidad temporal que han servido de base a trabajos en otros dominios:

- Dominio temporal: Par(T, $\leq$), donde T representa un conjunto de instancias de tiempo y $\leq$ es un orden total sobre el conjunto T.
- Granularidad: Mapeo G de enteros a un subconjunto del dominio temporal tal que si $i < j$ y tanto $G(i)$ como $G(j)$ no son vacíos, cada elemento de $G(i)$ será menor a todos los elementos en

$G(j)$.

- Gránulo: Cada subconjunto no vació $G(i)$ perteneciente a la granularidad G.

En este mismo trabajo se definen tipos de relaciones entre granularidades que poseen el mismo dominio temporal destacando las siguientes:

- *Groups Into*: Una granularidad G *Groups Into* la granularidad G' si cada gránulo en G' se forma por la unión de un conjunto de gránulos de G.

- *Finer Than*: Una granularidad G es *Finer Than* la granularidad G' si para cada gránulo en G' existe un conjunto de gránulos en G tal que el dominio abarcado por estos sea menor al del gránulo de G'.

- *Sub-granularity*: Una granularidad G es *Sub-granularity* de la granularidad G' si para cada gránulo en G existe un gránulo en G' tal que ambos cubren el mismo dominio.

En [7] se describen los formalismos y la forma de trabajar con multigranularidad temporal, lo cual involucra una calificación de sentencias a distintas granularidades y la definición de las relaciones entre las distintas granularidades temporales. Se plantea los siguientes componentes básicos para modelar la granularidad temporal:

- Lenguaje: Enfoque con el cual se representan las situaciones con respecto a los distintos niveles de abstracción. En el caso que la representación sea la misma a distintos niveles de abstracción, se trata de un lenguaje homogéneo; en otro caso, se trata de un conjunto de lenguajes para representar una situación.

- Capas: Representación de un nivel de abstracción (granularidad) de la información; por ejemplo, en términos de una aplicación temporal una capa sería mes.
- Operadores: Conjunto de herramientas que permiten operar con la estructura de capas.

Se distinguen dos enfoques con respecto al lenguaje:

- Cuantitativo: Modelo capaz de posicionar entidades temporales dentro de un marco métrico. Se puede basar en dos aproximaciones, una aproximación lógica o una basada en teoría de conjuntos.
- Cualitativo: Modelo capaz de caracterizar la posición de entidades temporales respecto a otras. Esta caracterización puede ser tanto topológica como vectorial.

Teoría de conjuntos se basa en álgebra y teoría de conjunto sencilla, y es ampliamente usado por bases de datos relacionales y sus extensiones. En ella se reemplaza el dominio temporal de modelos temporales planos por un universo temporal, definido como un conjunto de planos temporales ínter-relacionados, construido en base al plano más fino. Este plano es un set totalmente ordenado, donde sus elementos son la unidad temporal más fina que sea relevante para la instancia. De esta manera cada capa más gruesa se define como una partición de la capa más fina.

A diferencia de teoría de conjuntos, aproximación lógica se basa en un universo temporal basado en un posible conjunto infinito de capas de diferentes granos ínter-relacionados y herramientas lógicas capaces de calificar una sentencia temporal y cambiarla a medida que se mueve entre las capas.

En el enfoque cualitativo se aborda la multigranularidad de la información a través de relaciones algebraicas, definiendo un conjunto de relaciones entre las entidades.

En término de los operadores, en [7] se destacan dos operadores para realizar consultas básicas en una representación estructurada por capas.

- *Contextualization*: Operador para seleccionar una capa.
- *Projection*: Operador para moverse a través de las capas.

En [14] se propone una extensión del modelo relacional capaz de manejar granularidad temporal. Tal modelo se resume como una base de datos en la cual cada tupla es una marca de tiempo bajo alguna granularidad definida. En [3] se desarrolla un método para responder consultas sobre este modelo. Las respuesta se computan en base a las hipótesis vinculadas a instancia de base de datos, manejando los valores faltantes, considerándolos constantes o interpolándolos. Los autores proponen un algoritmo de arco-consistencia para comprobar cuando las granularidades son periódicas con respecto a una granularidad en común más fina, a la vez que proponen un algoritmo aproximado para manejar las restricciones de cada granularidad y la transformación de información de una granularidad a otra.

En el contexto espacial, el trabajo en [7] plantea el problema de la conversión espacial en datos multigranulares. Se define la conversión como el proceso en el cual la información almacenada en cierta granularidad se convierte a un nivel con menor o mayor detalle. Al trabajar con atributos agregados a una granularidad más gruesa se debe lidiar con valores no precisos, a la vez que al trabajar a una granularidad más fina, se debe lidiar con valores indeterminados.

En [7] se plantea que la conversión de gránulos a distintas granularidades es un proceso no reversible, por lo que puede llegar a afectar al plan de ejecución de una consulta. Se propone la creación de operadores específicos para el tratamiento de consultas con operadores geométricos. Además para tratar con el problema de reversibilidad se

propone la agregación de información en los casos en que no se posea un método de reversión. Finalmente, se plantea el problema de adaptación, que se define como la propiedad de adaptarse a cambios a nivel de granularidad y cambios en los gránulos de ellas. Para lidiar con el problema de adaptación proponen las dos siguientes alternativas:

- *object-oriented*: Utiliza propiedades de la orientación al objeto (clases, herencia, polimorfismo) para facilitar la adaptación a cambios de la estructura multigranular.
- *object-relational*: Combina las ventajas del modelo relacional y las de utilizar orientación al objeto.

A continuación se describe en detalle el modelo multigranular sobre el cual se basa este trabajo [10].

2.2. Modelo multigranular

El modelo de datos multigranular tiene como base la noción de granularidad, la cual aparece en diversas áreas de las ciencias de la computación. Esta noción tiene como base el hecho de que los atributos con referencia espacio/temporal en general se presentan o describen a distintos niveles de granularidad o detalle según sea necesario. Lo anterior ha sido previamente trabajado en [1, 4, 5, 10] con distintos enfoques y variando la forma en que se han definido las estructuras de gránulos y granularidades. En [4] se define una única estructura sobre gránulos y sin la definición de reglas que permitan indicar que un gránulo es la composición de otros gránulos, referidas como reglas de Join. En [10] se define un modelo multigranular que considera una estructura de orden parcial de granulares que se relaciona con la estructura a nivel de gránulos y que permite definir reglas de Join.

Los componentes básicos del modelo son las granularidades y los

gránulos que las forman. Informalmente una granularidad es una forma de dividir un dominio en gránulos que lo componen, mientras que un gránulo es una porción del dominio a representar y que no se superpone a ningún otro gránulo de la misma granularidad.

Se pude establecer una relación de orden parcial entre las granularidades de manera que una granularidad G se define como más fina que una granularidad G' si para todo gránulo en ella G existe un gránulo en G' que lo contiene. De manera similar, los gránulos también poseen una estructura de orden parcial dada por la relación de que un gránulo puede estar contenido en otro gránulo, lo que denomina una relación de *subsumption*.

La figura 2.1 ilustra una estructura de granularidad asociada a subdivisiones electorales y administrativas en Chile. Los arcos en esta figura representan relaciones entre granularidades. En este ejemplo los gránulos que componen a la granularidad comuna está formado por gránulos pertenecientes a la granularidad circunscripción electoral.

El símbolo $\top$ en la figura 2.1 define una granularidad global, que funciona como límite superior en el conjunto de granularidades. El símbolo '$\perp$' representa un límite inferior en el conjunto.

En la figura 2.1, los gránulos de la granularidad *ElecTable* se originan gránulos a niveles superiores según la división territorial correspondiente. Lo anterior destaca que la noción principal para trabajar con gránulos en el dominio espacial es la de división del espacio [6], dado que el trabajar con gránulos de carácter espacial se trata principalmente de un mapeo de información instanciada a un espacio específico.

El trabajo en [10] presenta una formalización en términos de lógica matemática de un modelo de datos multigranular, definiendo un esquema de granularidad como un par ordenado $\mathfrak{S}$ = ($\mathrm{Grty}(\mathfrak{S})$,$\mathrm{GrAsgn}(\mathfrak{S})$). $\mathrm{Grty}(\mathfrak{S})$ corresponde a un conjunto de granularidades donde cada una representa un nivel distinto de detalle de la información. Este conjunto

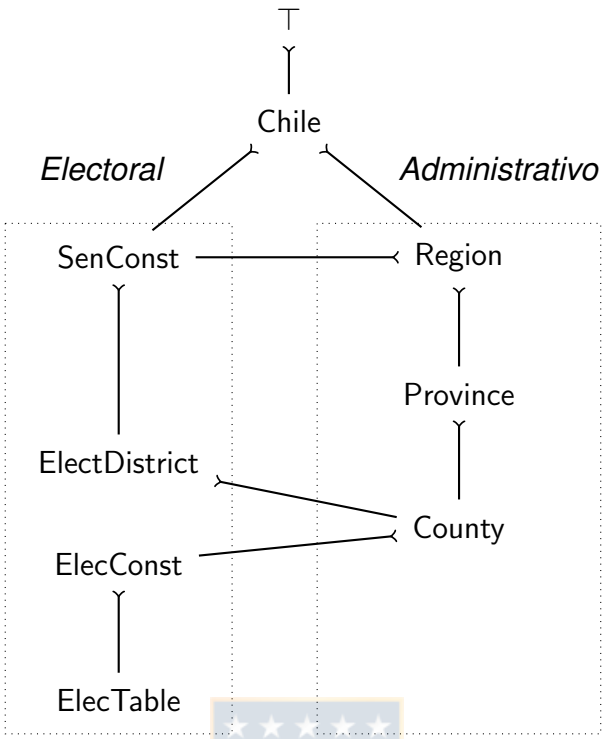


Figura 2.1: Jerarquía de granularidades Chile

posee una relación transitiva y reflexiva además de un límite superior. Por otra parte, $\mathrm{GrAsgn}(\mathfrak{S})$ extiende la idea de una asignación de dominio en un atributo relacional común al contexto multigranular, además proporciona un orden básico y asigna un conjunto de gránulos a cada granularidad. El modelo define reglas básica de manera de definir la estructura granular. Este conjunto de reglas básicas $\mathrm{PrBaRules}< \mathfrak{S} >$ son las siguientes:

- Regla de *subsumption* $g_1 \sqsubseteq_{\mathfrak{S}} \bigsqcup_{\mathfrak{S}} S$: Indica que el dominio que representa el gránulo g_1 está contenido en el dominio representado por el conjunto S. Notar que cuando $|S| = 1$ con solo un elemento g_2, esto es equivalente a $g_1 \sqsubseteq_{\mathfrak{S}} g_2$.

- Regla de *disjoint* $\bigsqcap_{\mathfrak{S}} \{g_1, g_2\} = \bot_{\mathfrak{S}}$: Indica que el dominio del gránulo g_1 no intersecta con el dominio del gránulo g_2

Este modelo define a su vez la negación de las reglas del conjunto $\mathrm{PrBaRules}< \mathfrak{S} >$, denotado como $\mathrm{NegPrBaRules}< \mathfrak{S} >$. El conjunto que contiene tanto $\mathrm{PrBaRules}< \mathfrak{S} >$ como $\mathrm{NegPrBaRules}< \mathfrak{S} >$ es denotado como $\mathrm{AllPrBaRules}< \mathfrak{S} >$.

Las reglas básicas descritas anteriormente incluyen dos relaciones fundamentales del modelo multigranular: (i) el estar **contenido** (*subsumption*), que hace referencia a cómo los gránulos están contenidos uno dentro del otro y (ii) el estar **no superpuestos** (*disjoint*) que define la no superposición total o parcial entre gránulos, lo cual es importante al momento de realizar operaciones de agregación sobre gránulos.

La combinación de reglas básicas permite la definición de reglas que permiten expresar condiciones que no fueron tratadas en trabajos anteriores que modelan granularidad. Un ejemplo de una regla más compleja sería:

$$Chile = \bigsqcup^{\perp} Region_R \mid I \leq R \leq XVI$$

Esta regla nos señala, a través del símbolo $\bigsqcup$, que Chile está formado por el join de las 16 regiones que la componen actualmente, mientras que el símbolo $\perp$ señala que el join es *disjoint*, es decir, no se sobreponen entre sí.

El trabajo desarrollado en [11] formaliza lo anteriormente planteado en [10]. Destaca que reglas complejas se definen en base a los conjuntos de reglas primitivas. Además define *join rules* como ($g \circledast \diamond S$), en donde $\circledast \in \{=, \sqsubseteq_{\mathfrak{S}}\}$ y $\diamond \in \{\bigsqcup_{\mathfrak{S}}, \bigsqcup^{\perp}_{\mathfrak{S}}\}$ siendo g un gránulo y S un conjunto de gránulos.

En [11] se introduce el concepto de pares de granularidades definido como un par $< G_1, G_2 > \in Glty(\mathfrak{S}) \times Glty(\mathfrak{S})$, tal que $G_1 \neq G_2$, y el concepto de *bigranular join rule* que define que una *join rule* φ será llamada *bigranular* ($< G_1, G_2 >$) si $\mathrm{Head}\ \langle\varphi\rangle \subseteq \mathrm{Granules}\ \langle\ \mathfrak{S}|G_1\ \rangle$ y $\mathrm{Body}\ \langle\varphi\rangle \subseteq \mathrm{Granules}\ \langle\ \mathfrak{S}|G_2\ \rangle$. Se destaca que *bigranular join rule* implican *disjoint*, es decir $\diamond\langle\varphi\rangle = \bigsqcup^{\perp}_{\mathfrak{S}}$ y que toda *bigranular join rule*

será $\mathrm{resolvable}$ para *disjoint*, es decir es posible establecer para todo caso si es verdadero o falso. Finalmente, se define el concepto de *minimal join rule* como una *join rule* φ en donde ningún elemento no vació $\Gamma \subseteq$ *Body* $\langle\varphi\rangle$ puede ser eliminado, pues de hacerlo, φ dejaría de ser consecuente con las restricciones de $\mathfrak{S}$.

Estos conceptos permiten definir relaciones entre granularidades que posibilitan capturar características del dominio representado con el fin de lograr una mejor representación. En [11] se definen las siguientes *join rules* bigranulares:

- *Equality join order*: $G_1 \trianglelefteq^{\Phi}_{\mathfrak{S}} G_2$. Esta relación indica que todo gránulo g en G_2 está compuesto por la unión (disjunta) de gránulos en G_1
- *Subsumption join order*: $G_1 \circledcirc^{\Phi}_{\mathfrak{S}} G_2$. Esta relación indica que todo gránulo g en G_2 está contenido dentro del dominio formado por la unión (disjunta) de gránulos en G_1.

Se destaca que *Equality join order* es transitivo y que implica *subsumption join order*. Además, ambas *join rules* en caso de ser minimales son tanto $\mathrm{biresolvable}$, es decir que se es $\mathrm{resolvable}$ tanto para *disjoint* como para *subsumption*, como también $\mathrm{equiresolvable}$,es decir, que la resolubilidad de *disjoint* y *subsumption* es equivalente.

Capítulo 3

Diseño del Modelo Multigranular

Se trabajó en el diseño de un modelo para representar la estructura de granularidades y gránulos buscando minimizar la cantidad de relaciones almacenadas. Se tomó como base el trabajo desarrollado en [10] y [11].

La primera sección de este capítulo define el modelo de datos utilizado para representar la estructura de granularidades y gránulos, luego se definen las reglas de inferencia a utilizar para reducir la cantidad de relaciones de *subsumption* y *disjoint* almacenadas en la estructura de granularidades y gránulos. Además, se analiza la información necesaria de almacenar utilizando las reglas de inferencia definidas previamente tanto para el caso general como también al considerar ciertas relaciones bigranulares específicas. En la segunda sección se detalla la implementación del modelo propuesto y los algoritmos desarrollados para responder consultas de *subsumption* y *disjoint* haciendo uso de las reglas de inferencia previamente definidas.

3.1. Nociones preliminares y notación

Se definen a continuación los conceptos y símbolos a utilizar en lo que resta del documento.

- Completitud: Dada dos granularidades G_1 y G_2 pertenecientes a $\mathrm{Glty}(\mathfrak{S})$, completitud de G_1 con respecto a G_2 denotará que para cualquier par de gránulos $< g_1, g_2 > \in G_1 \times G_2, \bigsqcap_{\mathfrak{S}}\{g_1, g_2\} = \bot_{\mathfrak{S}}$ y $g_1 \sqsubseteq_{\mathfrak{S}} g_2$ son conocidos.

- $Rel_{\mathfrak{S}}$: Se define como el conjunto de relaciones bigranulares existentes en $\mathfrak{S}$.

En consideración a lo desarrollado en [11], para este trabajo se extendió la cantidad de relaciones bigranulares a considerar, siendo $Rel_{\mathfrak{S}}$ formado por las siguientes relaciones:

- *Subtype relation*: $G_1 \trianglelefteq^{\Phi}_{\mathfrak{S}} G_2$: $(\forall\ g_1 \in G_1\ \exists\ g_2 \in G_2)\ (\Phi \models_{\mathfrak{S}} ([g_1]_{\mathfrak{S}} = [g_2]_{\mathfrak{S}}))$.

  Esta relación indica que para todo gránulo g en G_1 existe un gránulo g' en G_2 tal que el dominio representado por g es el mismo que el de g'.

- *Equality-join relation*: $G_1 \trianglelefteq^{\Phi}_{\mathfrak{S}} G_2$: $(\forall\ g_2 \in G_2\ \exists\ S \subseteq \mathrm{Granules}(G_1))$ $(\Phi \models_{\mathfrak{S}} (g_2 = \bigsqcup^{\perp}_{\mathfrak{S}} S))$.

  Esta relación indica que todo gránulo g en G_2 está compuesto por la unión (disjunta) de gránulos en G_1.

- *Inequality-join relation*: $G_1 \trianglerighteq^{\Phi}_{\mathfrak{S}} G_2$: $(\forall\ g_2 \in G_2\ \exists\ g_1 \in G_1)\ (\Phi \models_{\mathfrak{S}} (g_1 \sqsubseteq_{\mathfrak{S}} g_2))$.

  Esta relación indica que todo gránulo g en G_2 está contenido dentro del dominio formado por la unión (disjunta) de gránulos en G_1.

- *Granularity order relation*: $G_1 \leq^{\Phi}_{\mathfrak{S}} G_2$: $(\forall\ g_1 \in G_1\ \exists\ g_2 \in G_2)\ (\Phi \models_{\mathfrak{S}} (g_1 \sqsubseteq_{\mathfrak{S}} g_2))$.

  Esta relación indica que para todo gránulo g en G_1 existe un gránulo g' en G_2 tal que se cumpla $g \sqsubseteq_{\mathfrak{S}} g'$.

- *Combined order relation*: $G_1 \trianglelefteq\!\!\leq^{\Phi}_{\mathfrak{S}} G_2$: $(\forall\ g_2 \in G_2\ \exists\ S \subseteq \mathrm{Granules}(G_1))$ $(\Phi \models_{\mathfrak{S}} (g_2 = \bigsqcup^{\perp}_{\mathfrak{S}} S)) \bigwedge (\forall\ g_1 \in G_1\ \exists\ g_2 \in G_2)\ (\Phi \models_{\mathfrak{S}} (g_1 \sqsubseteq_{\mathfrak{S}} g_2))$.

  Esta relación indica que se tiene tanto $G_1 \leq^{\Phi}_{\mathfrak{S}} G_2$ como $G_1 \trianglelefteq^{\Phi}_{\mathfrak{S}} G_2$.

- *Reverse order relation*: $G_1 \gtrdot^{\Phi}_{\mathfrak{S}} G_2$: ($\forall\ g_2 \in G_2\ \exists\ g_1 \in G_1$) ($\Phi \models_{\mathfrak{S}} (g_1 \sqsubseteq_{\mathfrak{S}} g_2)$).

  Esta relación indica que para todo gránulo g' en G_2 existe un gránulo g en G_1 tal que se cumpla $g \sqsubseteq_{\mathfrak{S}} g'$

- *Join particular*: $G_1 \odot G_2$.

  Indica relaciones de *disjoint* o *subsumption* a nivel de gránulos entre ambas granularidades.

Estas relaciones entre granularidades definen el comportamiento entre sus gránulos, lo cual es útil al momento de trabajar con las reglas básicas de *disjoint* y *subsumption*. Estas relaciones adquieren importancia debido a lo siguiente:

- Son de frecuente aparición en instancias reales.
- Las reglas bigranulares implican *disjoint*, dado que los gránulos de una misma granularidad son *disjoint* entre sí.
- Para el conjunto definido previamente a excepción de $\odot$, las demás relaciones bigranulares implican $\mathrm{equiresolvable}$, esto conlleva un ahorro de espacio al solo almacenar *disjoint* o *subsumption* segun sea conveniente.

En lo que resta del documento, $\perp_{\mathfrak{S}}$ representa *disjoint*, $\not\perp_{\mathfrak{S}}$ representa *not disjoint*, *uk* $\perp_{\mathfrak{S}}$ representa el caso en que *disjoint* es desconocido, $\sqsubseteq_{\mathfrak{S}}$ representa *subsumption*, $\not\sqsubseteq_{\mathfrak{S}}$ representa *not subsumption*, *uk* $\sqsubseteq_{\mathfrak{S}}$ representa el caso en que subsumption es desconocido, G_1, G_2 y $G_3 \subseteq$ Glty($\mathfrak{S}$).

3.2. Modelo

A continuación se detalla el modelo propuesto, las reglas de inferencia y los análisis para determinar la información mínima a almacenar.

Para el desarrollo del modelo se tomaron las siguientes decisiones de diseño:

- El modelo propuesto está enfocado exclusivamente a relaciones bigranulares siendo incapaz de soportar un tipo de relación diferente a ésta.
- Se asume que jamás se almacenarán relaciones entre gránulos que puedan ser derivadas.
- Para todas las relaciones bigranulares excluyendo a $\odot$, los joins serán minimales, es decir que si un elemento del *join set* es removido la relación granular se vuelve falsa.

3.2.1. Modelado de base

En consideración a lo desarrollado en [10, 11] se diseñó un modelo para representar la estructura multigranular. En la figura 3.1 se refleja el modelado conceptual desarrollado. En éste, las entidades de contorno redondeado representan un conjunto de entidades de igual estructura, esto debido a que por cada granularidad en la entidad $\mathbf{Granularities}$ existe una entidad a la cual la granularidad hace referencia. De similar manera, ocurre para la tabla $\mathbf{BigranularJoins}$ en donde por cada tupla se tiene una referencia a 4 entidades ($\mathbf{Subtable_i}$, $\mathbf{Nsubtable_i}$, $\mathbf{Disjtable_i}$, $\mathbf{Ndisjtable_i}$). Para el modelado se ocuparon 2 tipos de conectores:

- Dependencia: Dado dos atributos A y B, indica que A es clave foránea del atributo B.
- Asociación: Dado un atributo A y una entidad B, indica que el atributo A es una referencia a la tabla B.

Se tomaron las siguientes decisiones de diseño para el desarrollo del modelo:

- Se incluyen en el diseño las entidades $\mathbf{Ukdisjtable}$ y $\mathbf{Uksubtable}$, las cuales representan relaciones entre gránulos en que se desconoce su valor de verdad, esto con el fin de disminuir tiempo de respuestas para consultas en donde no se pueda determinar la relación.

- Para cualquier tupla $\{Granulehead, Granulebody\}$ perteneciente a las entidades $\mathbf{Subtable_{ij}}$, $\mathbf{Nsubtable_{ij}}$, $\mathbf{Disjtable_{ij}}$, $\mathbf{Ndisjtable_{ij}}$, $\mathbf{Ukdisjtable}$ y $\mathbf{Uksubtable}$ jamas se tendrá que $Granulehead$ pertenezca a la misma granularidad que $Granulebody$.

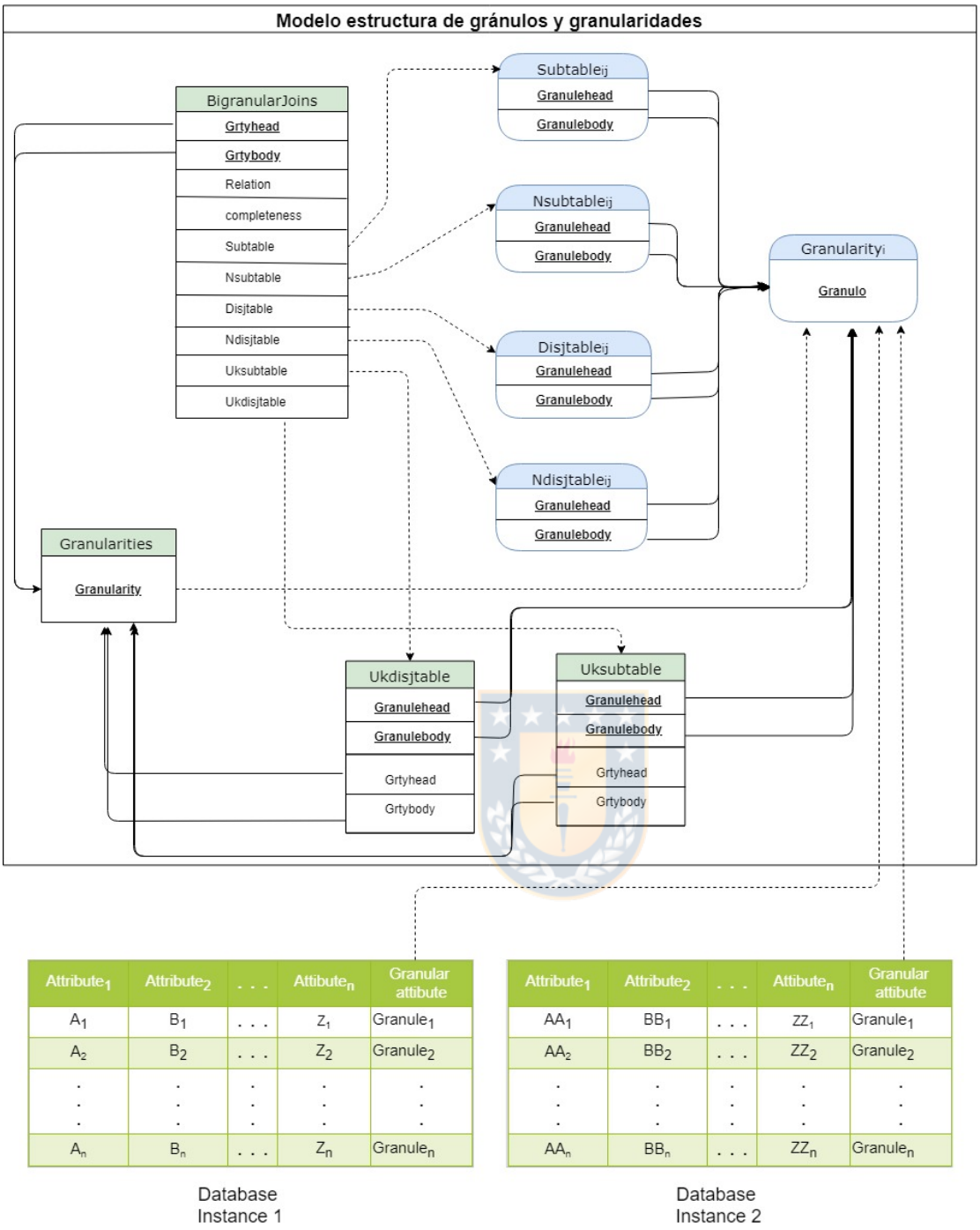
- El nombre de un gránulo puede no ser único, por lo que se identifican por el par $\{granule, granularity\}$ siendo $granularity$ la granularidad a la cual pertenece el gránulo $granule$.

Este modelo, al solo representar la relación entre gránulos y granularidades, no está ligado a una instancia específica sino que puede ser utilizado por diversas instancias de bases de datos.

3.2.2. Reglas de inferencia

En [9] se definen reglas de inferencia, las que se demuestran ser correctas y completas, para derivar relaciones de *subsumption* $\sqsubseteq_{\mathfrak{S}}$ y *disjoint* $\perp_{\mathfrak{S}}$ y sus negaciones. El uso de estas reglas permite un diseño que no requiere almacenar todas las relaciones entre gránulos. La definición parte por usar reglas de inferencia que hacen que $\sqsubseteq_{\mathfrak{S}}$ y $\perp_{\mathfrak{S}}$ sean verdaderas (reglas positivas). Ellas se derivan de la transitividad de $\sqsubseteq_{\mathfrak{S}}$ y de que si se cumple que g_1' y g_2' son disjoint, entonces los respectivos $g_1 \sqsubseteq_{\mathfrak{S}} g_1'$ y $g_2 \sqsubseteq_{\mathfrak{S}} g_2'$ también lo son. Se agregan reglas para derivar insatisfacción de relaciones (reglas negativas) e intercambiando predicados de premisa por conclusión.

$$\frac{g_1 \sqsubseteq_{\mathfrak{S}} g_2 \quad g_2 \sqsubseteq_{\mathfrak{S}} g_3}{g_1 \sqsubseteq_{\mathfrak{S}} g_3} \tag{3.1}$$



$Attribute_1$	$Attribute_2$	. . .	$Attibute_n$	Granular attibute
A_1	B_1	. . .	Z_1	$Granule_1$
A_2	B_2	. . .	Z_2	$Granule_2$
. . .	. . .	. . .	. . .	. . .
A_n	B_n	. . .	Z_n	$Granule_n$

Database Instance 1

$Attribute_1$	$Attribute_2$	. . .	$Attibute_n$	Granular attibute
AA_1	BB_1	. . .	ZZ_1	$Granule_1$
AA_2	BB_2	. . .	ZZ_2	$Granule_2$
. . .	. . .	. . .	. . .	. . .
AA_n	BB_n	. . .	ZZ_n	$Granule_n$

Database Instance 2

Figura 3.1: Modelo estructura multigranular

$$\frac{\bigsqcap_{\mathfrak{S}}\{g_1, g_2\} = \bot_{\mathfrak{S}}}{g_1 \not\sqsubseteq_{\mathfrak{S}} g_2} \tag{3.2}$$

$$\frac{g_1 \not\sqsubseteq_{\mathfrak{S}} g_2 \quad g_3 \sqsubseteq_{\mathfrak{S}} g_2}{g_1 \not\sqsubseteq_{\mathfrak{S}} g_3} \tag{3.3}$$

$$\frac{g_1 \sqsubseteq_{\mathfrak{S}} g_2 \quad g_1 \not\sqsubseteq_{\mathfrak{S}} g_3}{g_2 \not\sqsubseteq_{\mathfrak{S}} g_3} \tag{3.4}$$

$$\frac{\bigsqcap_{\mathfrak{S}}\{g_1, g_3\} \neq \bot_{\mathfrak{S}} \quad g_3 \sqsubseteq_{\mathfrak{S}} g_4 \quad \bigsqcap_{\mathfrak{S}}\{g_2, g_4\} = \bot_{\mathfrak{S}}}{g_1 \not\sqsubseteq_{\mathfrak{S}} g_2} \tag{3.5}$$

$$\frac{g_1 \sqsubseteq_{\mathfrak{S}} g_2 \quad \bigsqcap_{\mathfrak{S}}\{g_2, g_3\} = \bot_{\mathfrak{S}}}{\bigsqcap_{\mathfrak{S}}\{g_1, g_3\} = \bot_{\mathfrak{S}}} \tag{3.6}$$

$$\frac{g_1 \sqsubseteq_{\mathfrak{S}} g_2}{\bigsqcap_{\mathfrak{S}}\{g_1, g_2\} \neq \bot_{\mathfrak{S}}} \tag{3.7}$$

$$\frac{g_1 \sqsubseteq_{\mathfrak{S}} g_2 \quad g_1 \sqsubseteq_{\mathfrak{S}} g_3}{\bigsqcap_{\mathfrak{S}}\{g_2, g_3\} \neq \bot_{\mathfrak{S}}} \tag{3.8}$$

$$\frac{\bigsqcap_{\mathfrak{S}}\{g_1, g_2\} \neq \bot_{\mathfrak{S}} \quad g_2 \sqsubseteq_{\mathfrak{S}} g_3}{\bigsqcap_{\mathfrak{S}}\{g_1, g_3\} \neq \bot_{\mathfrak{S}}} \tag{3.9}$$

En base al modelo propuesto a continuación se analizan diversas relaciones bigranulares y la composición entre ellas con el objetivo de determinar la información mínima necesaria a almacenar al utilizar las reglas de inferencia propuestas para representar las relaciones de *disjoint* y *subsumption* entre gránulos.

3.2.3. Información mínima a almacenar

Para el caso general en la relación bigranular $G_1\ R_1\ G_2$, en donde $G_1, G_2 \in \mathrm{Glty}(\mathfrak{S})$, $R_1 \in \mathrm{Rel}_{\mathfrak{S}}$ y no se posee completitud en la información, la siguiente tabla refleja la información mínima a almacenar para

representar la relación R_1 existente entre los gránulos de G_1 con respecto a los gránulos de G_2 en caso de utilizar las reglas de inferencia anteriormente propuestas, sin perder información en comparación al caso en que se almacene la información completa.

$\perp_{\mathfrak{G}}$	$\not\perp_{\mathfrak{G}}$	uk $\perp_{\mathfrak{G}}$	$\sqsubseteq_{\mathfrak{G}}$	$\not\sqsubseteq_{\mathfrak{G}}$	uk $\sqsubseteq_{\mathfrak{G}}$
$V_1 \cup T_1$	$V_2 \cup T_2$	T_5	$V_3 \cup T_3$	$V_4 \cup T_4$	T_6

Donde:

- T_1 almacena casos en que la relación *disjoint* es verdadera y no puede ser derivada.
- T_2 almacena casos en que la relación *not disjoint* es verdadera y no puede ser derivada.
- T_3 almacena casos en que la relación *subsumptions* es verdadera y no puede ser derivada.
- T_4 almacena casos en que la relación *not subsumptions* es verdadera y no puede ser derivada.
- T_5 almacena tuplas en las que se desconoce si la relación *disjoint* es verdadera o falsa.
- T_6 almacena tuplas en las que se desconoce si la relación *subsumption* es verdadera o falsa.
- V_1 es una vista que deriva *disjoint* utilizando la regla de inferencia 3.6.
- V_2 es una vista que deriva *not disjoint* utilizando las reglas de inferencia 3.7, 3.8 y 3.9.
- V_3 es una vista que deriva *subsumptions* utilizando la regla de inferencia 3.1.
- V_4 es una vista que deriva *not subsumptions* utilizando las reglas de inferencia 3.2, 3.3, 3.4 y 3.5.

Con el objetivo de determinar el impacto de las relaciones bigranulares en la información a almacenar se realizó un análisis de estas tanto de manera individual como en composición con otra relación. Para esto se consideró que se posee completitud de información, en caso contrario las relaciones bigranulares no permiten ninguna equivalencia entre *no-disjoint* y *subsumption* dado que sería incorrecto considerar que son $\mathrm{equiresolvable}$ debido a la falta de información por lo que se vuelve al caso general previamente analizado. La siguiente tabla refleja el análisis realizado, considerar que se posee $G_1\ R_1\ G_2$, $G_2\ R_2\ G_3$ con $G_1, G_2, G_3 \in \mathrm{Glty}(\mathfrak{S})$ y $R_1, R_2 \in \mathrm{Rel}_{\mathfrak{S}}$ y que la columna Granularidades refleja el par de granularidades a analizar.

Granularidades	R1	R2	$\sqsubseteq_{\mathfrak{S}}$	$\not\perp_{\mathfrak{S}}$	$\not\sqsubseteq_{\mathfrak{S}}$	$\perp_{\mathfrak{S}}$
G_1 y G_3	$\{\trianglelefteq^{\Phi}_{\mathfrak{S}}\}$	$Rel_{\mathfrak{S}}$	V_1	V_2	V_3	V_4
	$\{\leq^{\Phi}_{\mathfrak{S}}, \lessdot^{\Phi}_{\mathfrak{S}}\}$	$\{\leq^{\Phi}_{\mathfrak{S}}, \lessdot^{\Phi}_{\mathfrak{S}}, \trianglelefteq^{\Phi}_{\mathfrak{S}}, \trianglelefteq^{\Phi}_{\mathfrak{S}}\}$				
	$\{\trianglelefteq^{\Phi}_{\mathfrak{S}}, \oslash^{\Phi}_{\mathfrak{S}}\}$	$\{\trianglelefteq^{\Phi}_{\mathfrak{S}}, \lessdot^{\Phi}_{\mathfrak{S}}\}$				
G_1 y G_2	$\{\trianglelefteq^{\Phi}_{\mathfrak{S}}, \leq^{\Phi}_{\mathfrak{S}}, \lessdot^{\Phi}_{\mathfrak{S}}, \trianglelefteq^{\Phi}_{\mathfrak{S}}\}$	-	$T_1 \cup V_1$	V_2	V_3	V_4
G_1 y G_3	$\{\overline{<}^{\Phi}_{\mathfrak{S}}, \odot\}$	$\{\lessdot^{\Phi}_{\mathfrak{S}}, \trianglelefteq^{\Phi}_{\mathfrak{S}}\}$				
	$\{\trianglelefteq^{\Phi}_{\mathfrak{S}}, \oslash^{\Phi}_{\mathfrak{S}}\}$	$\{\leq^{\Phi}_{\mathfrak{S}}, \trianglelefteq^{\Phi}_{\mathfrak{S}}\}$				
G_1 y G_2	$\{\overline{<}^{\Phi}_{\mathfrak{S}}, \oslash^{\Phi}_{\mathfrak{S}}, \odot\}$	-	$T_1 \cup V_1$	$T_2 \cup V_2$	V_3	V_4
G_1 y G_3	$\{\overline{<}^{\Phi}_{\mathfrak{S}}, \odot\}$	$\{\leq^{\Phi}_{\mathfrak{S}}, \overline{<}^{\Phi}_{\mathfrak{S}}, \oslash^{\Phi}_{\mathfrak{S}}, \trianglelefteq^{\Phi}_{\mathfrak{S}}, \odot\}$				
	$\{\leq^{\Phi}_{\mathfrak{S}}, \lessdot^{\Phi}_{\mathfrak{S}}, \trianglelefteq^{\Phi}_{\mathfrak{S}}, \oslash^{\Phi}_{\mathfrak{S}}\}$	$\{\overline{<}^{\Phi}_{\mathfrak{S}}, \oslash^{\Phi}_{\mathfrak{S}}, \odot\}$				

- Para representar $\sqsubseteq_{\mathfrak{S}}$ se utiliza tanto la tabla T_1, la cual almacena los casos en que la relación *subsumptions* es verdadera y no puede ser derivada, como la vista V_1, la cual deriva *subsumptions* utilizando la regla 3.1. Para ciertas composiciones entre relaciones se asegura que V_1 es capaz de derivar completa la relación $\sqsubseteq_{\mathfrak{S}}$, por lo que no es necesario T_1.

- Para representar $\not\perp_{\mathfrak{S}}$ se utiliza tanto la tabla T_2, la cual almacena los casos en que la relación *not disjoint* es verdadera y no puede ser derivada, como la vista V_2, la cual deriva *not disjoint* utilizando la regla 3.7, 3.8 y 3.9. En ciertos casos no es necesario T_2 dado que existe equivalencia entre *not disjoint* y *subsumptions*,

para estos casos V_2 es equivalente a $T_2 \cup V_2$.

- Para representar $\not\sqsubseteq_{\mathfrak{S}}$ se utiliza la vista V_3, la cual equivale a los elementos que no pertenecen a $T_1 \cup V_1$.
- Para representar $\perp_{\mathfrak{S}}$ se utiliza la vista V_4, la cual equivale a los elementos que no pertenecen a $T_2 \cup V_2$.

3.3. Implementación

A continuación se detalla la forma en que se implementó el modelo propuesto y los algoritmos que operan sobre este haciendo uso de las reglas de inferencia para responder consulta de *subsumption* y *disjoint* entre gránulos.

3.3.1. Esquema de datos

Se implementó el modelo propuesto como una base de datos relacional, se tomaron las siguientes decisiones de implementación:

- Se utilizó índice *HASH* para todas las tablas implementadas, esto dado a que únicamente se realizarán consultas por igualdad por lo que *HASH* ofrece un mejor desempeño en promedio en comparación a otros índices.
- todos los atributos de las tablas son del tipo de dato *VARCHAR*.

A continuación se detalla las tablas utilizadas y se da una descripción de sus atributos. en lo que resta de esta sección considerar que los atributos subrayados son claves primarias para su respectiva tabla:

1. $\boldsymbol{Granularities}$ ($\underline{Granularity}$)

   Tabla con un único atributo el cual indica el nombre de de las granularidades pertenecientes a Glty($\mathfrak{S}$). Posee un índice *HASH* sobre $Granularity$

2. $\boldsymbol{Granularity_i}$ ($\underline{Granule}$)

   $\boldsymbol{Granularity_i} \in \boldsymbol{Granularities}$. Por cada tupla en la tabla $Granularities$ existe una tabla de igual nombre que guarda los gránulos pertenecientes a aquella granularidad. Posee un índice *HASH* sobre el atributo $Granule$.

3. $\boldsymbol{BigranularJoins}$ ($\underline{Grtyhead}$, $\underline{Grtybody}$, $Relation$, $completeness$, $subtable$, $Nsubtable$, $Disjtable$, $Ndisjtable$, $Uksubtable$, $Ukdisjtable$)
   FK $Grtyhead$, $Grtybody$ **FROM** $Granularities$

   Tabla que contiene atributos $Gltyhead$ y $Gltybody$ que identifican granularidades, atributo $Relation$ que indica el tipo de relación entre las granularidad, atributo $Completeness$ que indica si la información de relaciones entre gránulos entre estas granularidades en $Gltyhead$ y $Gltybody$ es completa y atributos $subtable$, $Nsubtable$, $Disjtable$, $Ndisjtable$, $Uksubtable$ y $Ukdisjtable$ que almacenan los nombres de las tablas que almacenan las relaciones de *subsumption*, *not-subsumption*, *disjoint*, *not-disjoint*, *unknown subsumption* y *unknown disjoint* respectivamente. Atributo $Relation$ almacena una de las siguientes alternativas :

   - *Equality-join*: EQU
   - *Sub-type*: STP
   - *subsumption-join*: INE
   - *Granularity order*: GOR
   - *Combined order*: COR
   - *Reverse order*: ROR
   - *Join* particular: JNP

   La tabla posee un índice *HASH* tanto para $Grtyhead$ como para $Grtybody$.

4. $\boldsymbol{Subtable_{ij}}$ ($\underline{Granulehead}$, $\underline{Granulebody}$) **FK** $Granulehead$ **FROM** $Granularity_i\_\mathfrak{S}$, **FK** $Granulebody$ **FROM** $Granularity_j\_\mathfrak{S}$

   Tabla que representa la relación de *subsumption* entre $Granulehead$ con respecto a $Granulebody$ ($Granulehead \sqsubseteq_{\mathfrak{S}} Granulebody$). Posee un índice *HASH* tanto para $Granulehead$ como para $Granulebody$.

5. $\boldsymbol{Nsubtable_{ij}}$ ($\underline{Granulehead}$, $\underline{Granulebody}$) **FK** $Granulehead$ **FROM** $Granularity_i\_\mathfrak{S}$, **FK** $Granulebody$ **FROM** $Granularity_j\_\mathfrak{S}$

   Tabla que representa la relación de *not-subsumption* entre $Granulehead$ con respecto a $Granulebody$ ($Granulehead \not\sqsubseteq_{\mathfrak{S}} Granulebody$). Posee un índice *HASH* tanto para $Granulehead$ como para $Granulebody$.

6. $\boldsymbol{Disjtable_{ij}}$ ($\underline{Granulehead}$, $\underline{Granulebody}$) **FK** $Granulehead$ **FROM** $Granularity_i\_\mathfrak{S}$, **FK** $Granulebody$ **FROM** $Granularity_j\_\mathfrak{S}$

   Tabla que representa la relación de *disjoint* entre $Granulehead$ con respecto a $Granulebody$ ($\bigsqcap_{\mathfrak{S}}\{Granulehead, Granulebody\} = \bot_{\mathfrak{S}}$). Posee un índice *HASH* tanto para $Granulehead$ como para $Granulebody$.

7. $\boldsymbol{Ndisjtable_{ij}}$ ($\underline{Granulehead}$, $\underline{Granulebody}$) **FK** $Granulehead$ **FROM** $Granularity_i\_\mathfrak{S}$, **FK** $Granulebody$ **FROM** $Granularity_j\_\mathfrak{S}$

   Tabla que representa la relación de *not-disjoint* entre $Granulehead$ con respecto a $Granulebody$ ($\bigsqcap_{\mathfrak{S}}\{Granulehead, Granulebody\} \neq \bot_{\mathfrak{S}}$). Posee un índice *HASH* tanto para $Granulehead$ como para $Granulebody$.

8. $\boldsymbol{Uksubsumption\_\mathfrak{S}}$ ($\underline{Granulehead}$, $\underline{Granulebody}$, $Grtyhead$, $Grtybody$) **FK** $Granulehead$ **FROM** $Granularity_i$, **FK** $Granulebody$ **FROM** $Granularity_j$

   Tabla que almacena pares de gránulos $\{Granulehead, Granulebody\}$ en los que se desconoce si la relación *subsumption* es verdadera

o falsa. El atributo $Grtyhead$ indica a que granularidad pertenece $Granulehead$, de igual manera el atributo $Granulebody$ indica a que granularidad pertenece $Granulebody$. Posee un índice *HASH* tanto para $Granulehead$ como para $Granulebody$.

9. $\boldsymbol{Ukdisjoint\_\mathfrak{S}}$ ($\underline{Granulehead}$, $\underline{Granulebody}$, $Grtyhead$, $Grtybody$) **FK** $Granulehead$ **FROM** $Granularity_i\_\mathfrak{S}$, **FK** $Granulebody$ **FROM** $Granularity_j\_\mathfrak{S}$

   Tabla que almacena pares de gránulos $\{Granulehead, Granulebody\}$ en los que se desconoce si la relación *disjoint* es verdadera o falsa. El atributo $Grtyhead$ indica a que granularidad pertenece $Granulehead$, de igual manera el atributo $Granulebody$ indica a que granularidad pertenece $Granulebody$. Posee un índice *HASH* tanto para $Granulehead$ como para $Granulebody$.

3.3.2. Algoritmos

En base a lo expuesto en las secciones anteriores se desarrollaron 6 algoritmos, 4 de ellos utilizan las reglas de inferencia propuestas para responder si se cumple *disjoint* o *subsumption* entre dos gránulos mientras los otros 2 tienen por objetivo extraer información sobre *subsumption* entre gránulos, esto es requerido dado que las reglas de inferencia propuestas tienen como premisa el conocer *subsumption* entre gránulos. Se tomaron las siguientes decisiones para el diseño de los algoritmos:

- Para los algoritmos se asume que en las tablas no habrá almacenada de manera explícita información sobre relaciones entre gránulos que se puedan derivar a través de las reglas de inferencia propuestas.

- Se asume que en los casos que no se posee completitud las relaciones en las que desconoce su valor de verdad ya están

almacenadas ya sea en la tabla de **Uksubtable** o **Ukdisjtable** según corresponda.

- Para los análisis de complejidad se asume que los accesos a datos de cualquier tabla serán en tiempo constante, esto debido a que se asume que no existen colisiones al usar el índice *HASH*.
- Para aplicar las reglas de inferencia los algoritmos almacenan información relevante para el uso de las reglas en vectores, esto con el fin de evitar deber calcular más de una vez relaciones de *subsumption* entre gránulos, esta información una vez terminada la ejecución de los algoritmos es descartada.

A continuación se analiza en detalle cada algoritmo desarrollado y se incluye un análisis de complejidad para cada operador. No se incluyen en el análisis factores relacionados al procesamiento de consultas SQL, ni de tiempo de extracción de datos.

3.3.3. Funciones auxiliares

Se desarrollaron 2 funciones con el objetivo de extraer información de la estructura de gránulos y granularidades. Los algoritmos que se presentan a continuación tienen un enfoque de búsqueda en grafos debido a que las relación de *subsumption* entre gránulos es transitiva y forma un grafo dirigido. Para esto las funciones realizan una búsqueda por anchura partiendo de un gránulo objetivo a través de diversos gránulos con los que se posea la relación de *subsumption*. La información obtenida es usada posteriormente por los demás algoritmos desarrollados para aplicar las reglas de inferencias previamente propuestas, dado que gran parte de éstas tienen como premisa conocer *subsumption* entre dos gránulos.

Función SubsumpDown

Algorithm 1 Función $\mathbf{SubsumpDown}$

Input: Granule g, Granularity G
Output: Vector granulesisub

```
Vector Visitados, Queue Analizar
{Analizar es una cola de pares}
Analizar.push((G,g))
while Analizar.size() != empty() do
  Granularity grty ← Analizar.top().first
  Granule grano ← Analizar.top().second
  Analizar.pop()
  if !Visitados[grano] then
    granulesisub ← {grty, grano}
    for grty' IN IGrtysub(grty) do
      for grano' IN IGranulos(grty', grty, grano) do
        Analizar.push( {grty', grano'} )
    Visitados[grano] ← True
return granulesisub
```

La función $\mathbf{SubsumpDown}$ recibe un gránulo g y su respectiva granularidad G y a través de una búsqueda por anchura en la estructura de gránulos y granularidades retorna todos los gránulos que son subsumidos por g utilizando la regla de inferencia 3.1. Utiliza dos funciones para extraer información de la estructura de gránulos y granularidades: la función $\mathbf{IGrtysub}$ que recibe una granularidad y retorna las granularidades con las que posee una relación directa en la estructura multigranular, y la función $\mathbf{IGranulos}$ que recibe una granularidad objetivo grty', una granularidad grty y un gránulo g perteneciente a ella y retorna un conjunto de gránulos S perteneciente a grty' que cumple $\bigsqcup_{\mathfrak{G}} \mathrm{S} \sqsubseteq_{\mathfrak{G}} \mathrm{g}$.

Siendo N la cantidad de gránulos g' tal que cumplen $\mathrm{g}' \sqsubseteq_{\mathfrak{G}} \mathrm{g}$ y M la cantidad total de gránulos en la estructura multigranular, la función tiene un costo de N dado que realiza una búsqueda a través de la estructura multigranular almacenando solo los gránulos que cumplen con $\mathrm{g}' \sqsubseteq_{\mathfrak{G}} \mathrm{g}$, teniendo un costo constante la búsqueda y acceso en la estructura multigranular. El costo para el peor caso es de $O(\mathrm{M})$ dado que en el peor caso todos los gránulos en la estructura cumplirán con

$\mathrm{g}' \sqsubseteq_{\mathfrak{G}} \mathrm{g}$, por lo que N será equivalente a M.

Función SubsumpUp

Algorithm 2 Funcion $\mathbf{SubsumpUp}$

Input: Granule g, Granularity G
Output: Vector granulesisub

```
Vector Visitados, Queue Analizar
{Analizar es una cola de pares}
Analizar.push((G,g))
while Analizar.size() != empty() do
  Granularity grty ← Analizar.top().first
  Granule grano ← Analizar.top().second
  Analizar.pop()
  if !Visitados[grty] then
    Visitados[grty] ← True
    granulesisub ← {grty, grano}
    for grty' IN Grtysub(grty) do
      Analizar.push((grty',Granulo(grty', grty, grano))
return granulesisub
```

La función $\mathbf{SubsumpUp}$ recibe un gránulo g y su respectiva granularidad G y a través de una búsqueda por anchura en la estructura de gránulos y granularidades retorna todos los gránulos que subsumen a g utilizando la regla de inferencia 3.1. Utiliza dos funciones para extraer información de la estructura de gránulos y granularidades: la función $\mathbf{Grtysub}$ que recibe una granularidad y retorna las granularidades con las que posee una relación directa en la estructura multigranular y la función $\mathbf{Granulo}$ que recibe una granularidad grty' objetivo, una granularidad grty y un gránulo g perteneciente a ella, y retorna un gránulo g' perteneciente a grty' que cumple $\mathrm{g} \sqsubseteq_{\mathfrak{G}} \mathrm{g}'$.

Siendo N la cantidad de gránulos g' tal que cumplen $\mathrm{g} \sqsubseteq_{\mathfrak{G}} \mathrm{g}'$ y M la cantidad de granularidades en $\mathfrak{G}$, la función tiene un costo de N dado que realiza una búsqueda a través de la estructura multigranular almacenando solo los gránulos que cumplen con $\mathrm{g} \sqsubseteq_{\mathfrak{G}} \mathrm{g}'$, con un peor caso de O(M). Nota que $\mathrm{g} \sqsubseteq_{\mathfrak{G}} \mathrm{g}'$ puede ser verdadero solo una única

vez por cada granularidad, por lo que el peor caso ocurre cuando $\mathrm{g} \sqsubseteq_{\mathfrak{S}} \mathrm{g}'$ se cumple para cada granularidad, y en tal caso N seria equivalente a M.

Función Gquery

Se creó una función para realizar las consultas con el objetivo de simplificar los retornos de las funciones y facilitar el análisis de éstas.

Algorithm 3 Funcion $\mathbf{Gquery}$

Input: Granule g, Granule g', Granularity G, Granularity G', String type
Output: Boolean

```
Boolean retorno
if type = "Sub" then
  retorno = Sub(g, g', G, G')
  if retorno IS NULL then
    retorno = Nsub(g, g', G, G')
    if retorno = True then
      retorno = False
    if retorno = False then
      retorno = True
if type = "Nsub" then
  retorno = Nsub(g, g', G, G')
if type = "Dis" then
  retorno = Dis(g, g', G, G')
  if retorno IS NULL then
    retorno = Ndis(g, g', G, G')
    if retorno = True then
      retorno = False
    if retorno = False then
      retorno = True
if type = "Ndis" then
  retorno = Ndis(g, g', G, G')
  if retorno IS NULL then
    retorno = Dis(g, g', G, G')
    if retorno = True then
      retorno = False
    if retorno = False then
      retorno = True
return retorno;
```

La función $\mathbf{Gquery}$ recibe como parámetros dos gránulos g, g', sus respectivas granularidades y el tipo de consulta a realizar, y retorna el resultado obtenido según la consulta. Se omite el análisis de peor caso, dado que no realiza ninguna operación interna más allá de llamar a la consulta correspondiente.

3.3.4. Funcion Sub

Algorithm 4 Funcion $\mathbf{Sub}$

Input: Granule g, Granule g', Granularity G, Granularity G'
Output: Boolean

```
Vector Visitados, SubUg, Queue Analizar
Analizar.push((G,g)) {esto es una prueba}
if Directsub(g, G, g', G') then
  return True
if Directnsub(g, G, g', G') then
  return False
if Uk(g, G, g', G') then
  return NULL
SubUg ← SubsumpUp(g, G)
if g' ∈ SubUg then
  return True
if Completeness(G,G') then
  return False
return NULL
```

La función $\mathbf{Sub}$ recibe dos gránulos g y g', sus respectivas granularidades y retorna si se cumple $\mathrm{g} \sqsubseteq_{\mathfrak{G}} \mathrm{g}'$. La única regla para inferir $\sqsubseteq_{\mathfrak{G}}$ es la regla 3.1. El operador comprueba primero si es que existe alguna instancia explícita que relacione los gránulos g y g' a través de las consultas $\mathbf{Directsub}$, $\mathbf{Directnsub}$ y $\mathbf{Uk}$ (sea $\sqsubseteq_{\mathfrak{G}}$, $\not\sqsubseteq_{\mathfrak{G}}$ o $unknown$). En caso de no existir se realizará inferencia por la regla 3.1 utilizando la función $\mathbf{SubsumpUp}$ para obtener todos los gránulos que subsumen a g y comprobando si alguno de ellos corresponde a g'.

En caso de que no se logre comprobar $\sqsubseteq_{\mathfrak{G}}$, se retorna False si se posee completitud; en caso contrato retorna Null. La completitud es

una condición almacenada en la tabla $\mathbf{BigranularJoins}$.

Esta función tiene la misma complejidad de peor caso que la función $\mathbf{SubsumpUp}$, es decir, O(M) siendo M la cantidad de granularidades en $\mathrm{Glty}(\mathfrak{S})$, esto dado que el costo principal de la función está asociado al momento de utilizar la función $\mathbf{SubsumpUp}$.

3.3.5. Función Ndis

Algorithm 5 Funcion $\mathbf{Ndis}$

Input: Granule g, Granule g', Granularity G, Granularity G'
Output: Boolean

```
Vector NdisjUg, NdisjUg', NdisjDg, NdisjDg'
{NdisjUg, NdisjUg', NdisjDg y NdisjDg' son vectores de pares}
Boolean completitud ← Completeness(G,G')
if DirectNdis(g, G, g', G') then
  return True
if DirectDis(g, G, g', G') then
  return False
if Uk(g, G, g', G') then
  return NULL
if completitud AND EqualSub(g, G, g', G') then
  return Sub(g, g', G, G')
NdisjUg ← SubsumpUp(g, G)
NdisjUg' ← SubsumpUp(g', G')
NdisjDg ← SubsumpDown(g, G)
NdisjDg' ← SubsumpDown(g', G')
if G ∈ NdisjUg' OR G' ∈ NdisjUg then
  if Cont(NdisjUg, NdisjUg', g, G, g', G') then
    return True
  else
    return False
if Equal(NdisjDg, NdisjDg') then
  return True
if ndisjdirect(NdisjDg, NdisjDg') then
  return True
if completitud then
  return False
return NULL
```

La función $\mathbf{Ndis}$ recibe dos gránulos g y g', sus respectivas granularidades y responde si se cumple $\bigsqcap_{\mathfrak{S}}\{\mathrm{g},\mathrm{g}'\} \neq \bot_{\mathfrak{S}}$. La función $\mathbf{Ndis}$ utiliza las reglas de inferencia 3.7, 3.8 y 3.9. Primero se comprueba si es que existe una tupla en la instancia que relacione a los gránulos g y g' a través de las consultas $\mathbf{DirectDis}$, $\mathbf{DirectNdis}$ y $\mathbf{Uk}$ ($\bigsqcap_{\mathfrak{S}}\{g_1,g_2\} = \bot_{\mathfrak{S}}$, $\bigsqcap_{\mathfrak{S}}\{g_1,g_2\} \neq \bot_{\mathfrak{S}}$ o $unknown$). Luego comprueba si existe completitud de los datos para la relación entre ambas granularidades; en caso de existir, comprueba si se cumple que se posee un camino que garantice completitud y equivalencia de $\mathrm{g}_1 \sqsubseteq_{\mathfrak{S}} \mathrm{g}_2$ con $\bigsqcap_{\mathfrak{S}}\{\mathrm{g}_1,\mathrm{g}_2\} \neq \bot_{\mathfrak{S}}$, esto a través de la consulta $\mathbf{EqualSub}$. En caso de existir, según lo analizado con respecto a la composición de relaciones bigranulares, basta con responder por medio de la función $\mathbf{Sub}$. Dado que $\mathbf{Sub}$ es el algoritmo con menor costo, es conveniente responder a través de él. En caso de no existir un camino que garantice completitud y equivalencia, utiliza $\mathbf{SubsumpUp}$ y $\mathbf{SubsumpDown}$ para llenar los vectores $\mathrm{NdisjUg}$, $\mathrm{NdisjUg}'$, $\mathrm{NdisjDg}$ y $\mathrm{NdisjDg}'$. Este pre-cálculo es necesario para poder realizar una comprobación de manera eficiente las reglas de inferencia, pues esta información es necesaria para poder aplicar las reglas de inferencia y, en caso de no almacenar y calcular cuando sea necesario, se derivaría más de una vez una misma relación granular, luego se comprueba si se puede utilizar alguna de las reglas de inferencia para responder.

A continuación se detalla como se utiliza cada regla de inferencia relacionada a la consulta de *not-disjoint*

- Regla 3.7: Comprueba si se cumple que G esté en $\mathrm{NdisjUg}'$ o G' en $\mathrm{NdisjUg}$. En caso de estar, se comprobará si se cumple $\mathrm{g} \sqsubseteq_{\mathfrak{S}} \mathrm{g}'$ a través de la regla 3.1. Para esto, utiliza la consulta $\mathbf{Cont}$, la cual verifica si g está en $\mathrm{NdisjUg}'$ o g' $\mathrm{NdisjUg}$, retornando True en caso que se cumpla y False en caso contrario.

- Regla 3.8: Comprueba con la consulta $\mathbf{Equal}$ si los vectores $\mathrm{NdisjDg}$ y $\mathrm{NdisjDg}'$ poseen algún elemento en común. En caso

que se cumpla retorna True.

- Regla 3.9: Comprueba si algún elemento g_1 en $\mathrm{NdisjDg}$ cumple $\bigsqcap_{\mathfrak{S}}\{\mathrm{g}_1, \mathrm{g}_2\} \neq \bot_{\mathfrak{S}}$ con algún elemento g_2 de $\mathrm{NdisjDg}'$ usando la consulta $\mathbf{ndisjdirect}$. En caso de cumplirse retornará True. No es posible el requerir derivar la relación $\bigsqcap_{\mathfrak{S}}\{\mathrm{g}_1, \mathrm{g}_2\} \neq \bot_{\mathfrak{S}}$ para aplicar la regla, pues en caso de requerir derivar, sería por la regla 3.7 ó 3.8, pero en caso de derivar por alguna de estas reglas también derivaría $\bigsqcap_{\mathfrak{S}}\{\mathrm{g}, \mathrm{g}'\} \neq \bot_{\mathfrak{S}}$, con esto se puede concluir que $\bigsqcap_{\mathfrak{S}}\{\mathrm{g}_1, \mathrm{g}_2\} \neq \bot_{\mathfrak{S}}$ debe estar almacenado de manera explícita.

En el caso de que nada de lo anterior se cumpla se retorna False, si existe completitud, o NULL en caso contrario.

Se puede distinguir las siguientes 4 etapas principales en la función:

- Llenar los vectores $\mathrm{NdisjUg}$, $\mathrm{NdisjUg}'$, $\mathrm{NdisjDg}$ y $\mathrm{NdisjDg}'$ tiene un costo de $O(\mathsf{N}+\mathsf{M})$ en el peor caso, siendo N la cantidad de gránulos en la base y M la cantidad de granularidades en $\mathrm{Glty}(\mathfrak{S})$, este costo es aportado por las funciones $\mathbf{SubsumpDown}$ y $\mathbf{SubsumpUp}$ respectivamente.

- Regla 3.7 tiene un costo de $O(\mathsf{M})$ en el peor caso, dado que a lo más pueden haber M elementos en los vectores $\mathrm{NdisjUg}$ y $\mathrm{NdisjUg}'$. y la comprobación se realiza de manera lineal.

- Regla 3.8 tiene un costo de $O(\mathsf{N}^2)$ en el peor caso, dado que por cada elemento en el vector $\mathrm{NdisjDg}$ se debe comprobar si se encuentra también en el vector $\mathrm{NdisjDg}'$, siendo N el máximo tamaño que pueden alcanzar ambos vectores.

- Regla 3.9 tiene un costo de $O(\mathsf{N}^2)$ en el peor caso, dado que por cada elemento en el vector $\mathrm{NdisjDg}$ se debe comprobar si

se cumple *no-disjoint* con algún elemento en el vector $\mathrm{NdisjDg'}$, dado que en caso de cumplirse la regla la relación de *no-disjoint* debe estar almacenada de manera explicita, comprobar si se posee la relación toma tiempo constante.

Debido a lo anterior se tiene un costo para la consulta de $O(\mathrm{N}^2 + \mathrm{M})$ en el peor caso.

3.3.6. Operador Dis

Función $\mathbf{Dis}$ recibe dos gránulos g y $\mathrm{g'}$, sus respectivas granularidades y responde si se cumple $\bigsqcap_{\mathfrak{S}}\{\mathrm{g}, \mathrm{g'}\} = \bot_{\mathfrak{S}}$. La función $\mathbf{Dis}$ utiliza únicamente las reglas de inferencia 3.6. Primero se comprueba si es que existe una tupla en la instancia que relaciona a los gránulos g_1 y g_2 utilizando las consultas $\mathbf{DirectDis}$, $\mathbf{DirectNdis}$ y $\mathbf{Uk}(\bigsqcap_{\mathfrak{S}}\{g_1, g_2\} = \bot_{\mathfrak{S}}$, $\bigsqcap_{\mathfrak{S}}\{g_1, g_2\} \neq \bot_{\mathfrak{S}}$ o $unknown$), de igual manera que en la función $\mathbf{ndis}$ se utiliza la consulta $\mathbf{Equalsub}$ en caso de tener completitud de datos. En caso que se cumpla, basta con responder con la función $\mathbf{Sub}$, en el contrario, responde a través de la función $\mathbf{Ndis}$. En caso de no poseer completitud se utiliza la función $\mathbf{SubsumpUp}$ para llenar los vectores disjg y $\mathrm{disjg'}$, luego se comprueba si se puede utilizar la regla de inferencia 3.6 para responder.

A continuación se detalla como se utiliza la regla de inferencia 3.6 para responder la consulta de *disjoint*

- Regla 3.6: para comprobar la regla 3.6 se ven dos casos. El primer caso es en que en la regla de inferencia g_2 pertenece a misma granularidad de g_3, por ende, son *disjoint*. Para este caso se comprueba si $\mathrm{G'} \in \mathrm{disjg}$. En caso de estar se retornará True si cumple que $\mathrm{g'} \in \mathrm{disjg}$, False en caso contrario. Se comprueba de igual manera con G con respecto a $\mathrm{disjg'}$. Luego se comprueba, a través de la consulta $\mathbf{NotEqual}$, si en ambos vectores existe una misma granularidad G pero con distintos gránulos asociados

Algorithm 6 Operador $\mathbf{Dis}$

Input: Granule g, Granule g', Granularity G, Granularity G'
Output: Boolean

```
Vector disjg, disjg'
{disjg y disjg' son vectores de pares}
Boolean completitud ← Completeness(G,G')
if DirectDis(g, G, g', G') then
  return True
if DirectNdis(g, G, g', G') then
  return False
if Uk(g, G, g', G') then
  return NULL
if completitud then
  if EqualSub(g, G, g', G') then
    return Sub(g, g', G, G')
  else
    return Ndis(g, g', G, G')
disjg ← SubsumpUp(g, G)
disjg' ← SubsumpUp(g', G')
if G' ∈ disjg then
  if {G', g' } ∈ disjg then
    return False
  else
    return True
if G ∈ disjg' then
  if {G, g } ∈ disjg' then
    return False
  else
    return True
if NotEqual(disjg', disjg) then
  return TRUE
if disjdirect(disjg, disjg') then
  return True
return NULL
```

a ella. En caso de cumplirse se retornará True. Para el segundo caso (caso general), se comprueba si existe algún gránulo $\mathrm{g_i}$ en disjg que sea *disjoint* con algún gránulo $\mathrm{g'_i}$ de $\mathrm{disjg'}$ utilizando la consulta $\mathbf{disjdirect}$, de cumplirse, se concluye que todo gránulo

que es subsumido por $\mathrm{g_i}$ será *disjoint* con respecto a todo gránulo subsumido por $\mathrm{g_i'}$.

En caso de no cumplir nada de lo anterior retornará NULL.

Se pueden distinguir las siguientes 3 etapas principales en la función:

- Llenar los vectores disjg y $\mathrm{disjg'}$ tiene un costo de O(M) en el peor caso, siendo M la cantidad de granularidades en $\mathrm{Glty}(\mathfrak{S})$. Este costo es aportado por las función $\mathbf{SubsumpUp}$.
- Regla 3.6 primer caso: tiene un costo de $O(\mathrm{M} + \mathrm{M}^2)$ en el peor caso, dado que en primer lugar realiza una comprobación lineal en disjg y $\mathrm{disjg'}$ si $\mathrm{G'}$ o G respectivamente pertenece a ellos y luego por cada granularidad en disjg comprueba si esta se encuentra en $\mathrm{disjg'}$.
- Regla 3.6 segundo caso: tiene un costo de $O(M^2)$ en el peor caso, dado que por cada elemento en el vector disjg debe comprobar si se cumple *disjoint* con algún elemento en el vector $\mathrm{disjg'}$, comprobar*disjoint* para cualquier elemento del vector disjg con respecto a un elemento en $\mathrm{disjg'}$ toma tiempo constante, dado que en caso de cumplirse esta relación debe estar almacenada de manera explicita en la estructura multigranular .

Debido a lo anterior se tiene un costo para la consulta de $O(\mathrm{M}^2 + \mathrm{M})$ en el peor caso.

3.3.7. Operador Nsub

Función $\mathbf{Nsub}$ recibe dos gránulos g y $\mathrm{g'}$, sus respectivas granularidades y responde si se cumple $\mathrm{g} \not\sqsubseteq_{\mathfrak{S}} \mathrm{g'}$, utiliza las reglas de inferencia 3.2, 3.3, 3.4, y 3.5. Primero se comprueba si existe una tupla en la instancia que relacione a los gránulos $\mathrm{g_1}$ y $\mathrm{g_2}$ a través de las consultas $\mathbf{Directsub}$, $\mathbf{Directnsub}$ y $\mathbf{Uk}$ ($g_1 \sqsubseteq_{\mathfrak{S}} g_2$, $g_1 \not\sqsubseteq_{\mathfrak{S}} g_2$ o $unknown$).

Algorithm 7 Operador $\mathbf{Nsub}$

Input: Granule g, Granule g', Granularity G, Granularity G'
Output: Boolean

```
Vector Ndisjg, Ndisjg', NdisjgU, Ndisjg'U {Ndisjg, Ndisjg', NdisjgU y
Ndisjg'U son vectores de pares}
if DirectNsub(g, G, g', G') then
  return True
if DirectSub(g, G, g', G') then
  return False
if Uk(g, G, g', G') then
  return NULL
if completitud then
  return Sub(g, g', G, G')
Ndisjg ← SubsumpDown(g, G)
Ndisjg' ← SubsumpDown(g', G'
Ndisjg'U ← SubsumpUp(g', G')
if Sub(g, g', G, G') = True then
  return False
if Dis(g, g', G, G') then
  return True
Boolean ndisdirectDisj ← ndisjdirectDis(Ndisjg, Ndisjg',g, G, g', G')
if ndisdirectDisj != NULL then
  return ndisdirectDisj
if ndisdirect(Ndisjg, Ndisjg'U) then
  return True
return NULL
```

Después comprueba si se posee completitud, en caso de darse, le basta con responder utilizando $\mathbf{Sub}$, en caso contrario se rellena los vectores Ndisjg, $\mathrm{Ndisjg'}$ y $\mathrm{Ndisjg'U}$ con las funciones $\mathbf{SubsumpDown}$ y $\mathbf{SubsumpUp}$, además se comprueba si se cumple que $\mathrm{g} \sqsubseteq_{\mathfrak{S}} \mathrm{g'}$. En caso de cumplirse retorna False, de lo contrario, se comprueba si se puede utilizar alguna de las reglas de inferencia para responder.

A continuación se detalla como se utiliza las reglas de inferencia para responder la consulta de *not Subsumption*

- Regla 3.2: Comprueba si se cumple $\bigsqcap_{\mathfrak{S}} \{\mathrm{g}, \mathrm{g'}\} = \bot_{\mathfrak{S}}$ utilizando la función $\mathbf{Dis}$.

- Regla 3.5: Comprueba si se cumple (tomando como g_2 y g_4 dos gránulos de una misma granularidad, por tanto se cumple que ambos serán *disjoint*) utilizando la consulta $\mathbf{ndisjdirectDis}$, la cual opera de similar manera a la regla 3.9 utilizada en la función $\mathbf{Ndis}$, esta función retornará True en caso de cumplirse la inferencia buscada. Para un caso general no es viable aplicar esta regla dado el alto costo asociado a ella, esto debido a que es requerido computar todas las posibles relaciones de $\bigsqcap_{\mathfrak{S}}\{\mathrm{g}_2, \mathrm{g}_4\} = \bot_{\mathfrak{S}}$ y $\bigsqcap_{\mathfrak{S}}\{\mathrm{g}_1, \mathrm{g}_3\} = \bot_{\mathfrak{S}}$.

- Regla 3.3 y 3.4: Comprueba si se cumple utilizando la consulta $\mathbf{ndisdirect}$, la que comprueba si existe $\not\sqsubseteq_{\mathfrak{S}}$ entre algún elemento g de Ndisjg con respecto a un elemento g' de $\mathrm{Ndisjg'U}$. En caso de cumplirse se tiene que todo granulo que subsuma a g será $\not\sqsubseteq_{\mathfrak{S}}$ con respecto a todo gránulo subsumido por g'.

Finalmente retorna NULL. Se pueden distinguir las siguientes 4 etapas principales en la función:

- Llenar los vectores Ndisjg, $\mathrm{Ndisjg'}$ y $\mathrm{Ndisjg'U}$ tiene un costo de $O(\mathrm{N} + \mathrm{M})$ en el peor caso, siendo N la cantidad de gránulos en la base y M la cantidad de granularidades en $\mathrm{Glty}(\mathfrak{S})$. Este costo es aportado por las funciones $\mathbf{SubsumpDown}$ y $\mathbf{SubsumpUp}$ respectivamente.

- Regla 3.2 tiene el costo de la función $\mathbf{Dis}$, es decir $O(\mathrm{M}^2 + \mathrm{M})$ en el peor caso.

- Regla 3.5 tiene un costo de $O(\mathrm{N}^2)$ en el peor caso, dado que por cada elemento en el vector Ndisjg se debe comprobar si se cumple *no-disjoint* con algún elemento en el vector $\mathrm{Ndisjg'}$, el comprobar *no-disjoint* entre dos gránulos tiene un costo constante debido a que en caso de cumplirse debe estar almacenado de manera explicita en la estructura multigranular.

- Regla 3.3 y 3.4 tiene un costo de $O(\mathrm{N} \times \mathrm{M})$ en el peor caso, dado que por cada elemento en el vector Ndisjg se debe comprobar si se cumple *no-subsumption* con algún elemento en el vector $\mathrm{Ndisjg'U}$, el comprobar *no-disjoint* entre dos gránulos tiene un costo constante debido a que en caso de cumplirse debe estar almacenado de manera explicita en la estructura multigranular.

Debido a lo anterior se tiene un costo para la consulta de $O(\mathrm{M}^2 + \mathrm{N}^2 + \mathrm{N} \times \mathrm{M} + \mathrm{M} + \mathrm{N})$ en el peor caso.

Capítulo 4

Experimentos y Resultados

Se realizaron experimentos para evaluar el comportamiento de los algoritmos propuestos frente a sus peores casos. Además, se evaluó el comportamiento de la estrategia de inferencia frente a dos alternativas: (i) Una implementación en un modelo relacional plano; es decir, una implementación en la que cada instancia incluye la información de la estructura multigranular. (ii) Una implementación en un modelo multigranular que no utiliza las regla de inferencias propuestas, por ende, se almacenan de manera explícita las relaciones entre gránulos en la estructura multigranular.

Se implementaron los algoritmos en PlPgSQL 9.15.14 como funciones. Los experimentos se corrieron en una máquina con $Intel\ Xeon$ E3-1220 v5, 64GB $DDR4memory$, y $PostgreSQL$ 9.5.12.

4.1. Comportamiento frente peor caso

Para evaluar el comportamiento de los algoritmos propuestos se creó una estructura multigranular con la cual se pudiese controlar el peor caso para los algoritmos, esto con el objetivo de medir su rendimiento frente a situaciones desfavorables. La estructura posee las siguientes características:

- 2000 granularidades.
- 10 gránulos por cada granularidad, esto pues el aumentar la cantidad de gránulos por granularidad tiene impacto en el costo para acceder a los datos pero no en el desempeño de los algoritmos, dado que estos se ven afectados por la cantidad de gránulos

que cumplan con la relación de *subsumption* lo cual no es dependiente de la cantidad de gránulos por granularidad. Además al utilizar una baja cantidad de gránulos por granularidad se posee un mayor control de cuantos gránulos cumplen la condición de *subsumption* con respecto a un gránulo g'. Finalmente destacar que se asume que el índice *HASH* no produce colisiones, por lo que se deprecia el tiempo de acceso a la información.

- $\mathrm{Rel}\mathfrak{S}$ está conformado únicamente por JNP, esto para forzar a los algoritmos a aplicar todas las reglas de inferencia correspondientes según sea el caso y que estos no apliquen uso de la composición entre relaciones bigranulares para responder.

- En toda relación entre granularidades no se posee completitud, esto para evitar obtener *not-subsumption* y *disjoint* a través de *subsumption* y *not-disjoint*.

En base a esta estructura multigranular se realizó un total de 50 consultas agrupadas según la cantidad de gránulos requeridos de analizar (10 consultas por cada múltiplo de 400 hasta los 2000 gránulos). Todas las consultas retornan $False$ como respuesta, esto con el fin de asegurar que los algoritmos realicen todas las inferencias posibles. La figura 4.1 refleja en promedio el tiempo por cada conjunto de consultas.

4.2. Comportamiento datos reales

Para evaluar el comportamiento del modelo propuesto frente a una instancia real se utilizó como conjunto de datos reales los datos de divisiones electorales y divisiones administrativas para el censo 2017. La propuesta de implementación se comparó con dos alternativas: (i) implementada en un modelo relacional plano y (ii) implemetación explícita de relaciones entre gránulos sin reglas de inferencia.

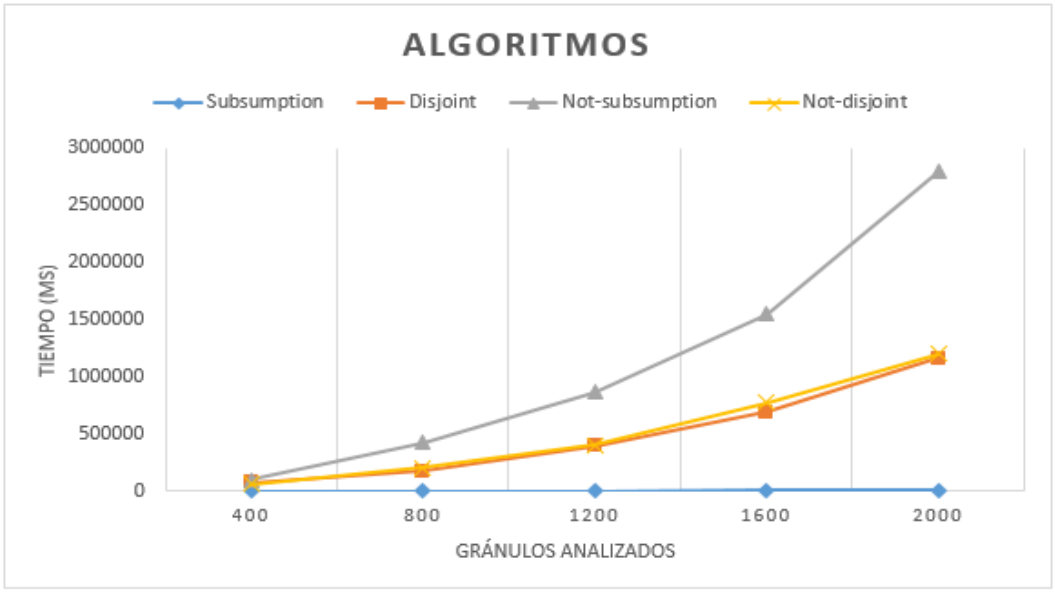


Figura 4.1: Gráfico desempeño función *disjoint*

En la figura 4.2 se refleja la jerarquía de granularidades asociada a la división electoral y administrativa de Chile, cuyos siglas se describen en la tabla 4.1. Cabe destacar que no se posee completitud entre todos los pares $< G_1 , G_2 > \in \mathrm{Glty}(\mathfrak{S}) \times \mathrm{Glty}(\mathfrak{S})$, específicamente para las granularidades más finas a Distrito censal no se posee completitud. En base a esta se desarrollaron dos implementaciones:

1. Implementación propuesta: la cual utiliza reglas de inferencia para calcular *disjoint* y *subsumption*.

2. Implementación que no utiliza reglas de inferencia: almacena todas las relaciones de *disjoint* y *subsumption* entre gránulos.

Con respecto a la implementación basada en un modelo relacional plano, se poseen dos instancias originales: (i) la instancia que guarda la información censal e (ii) instancia que guarda la información con respecto a la votación electoral 2013. Ambas instancias son una tabla plana que no usan índices. Se evaluó el uso de índice *HASH*, lo que implicaba crear índices por cada atributo y aumentando, en consecuencia, el espacio en alrededor de un 50 % del espacio de la tabla

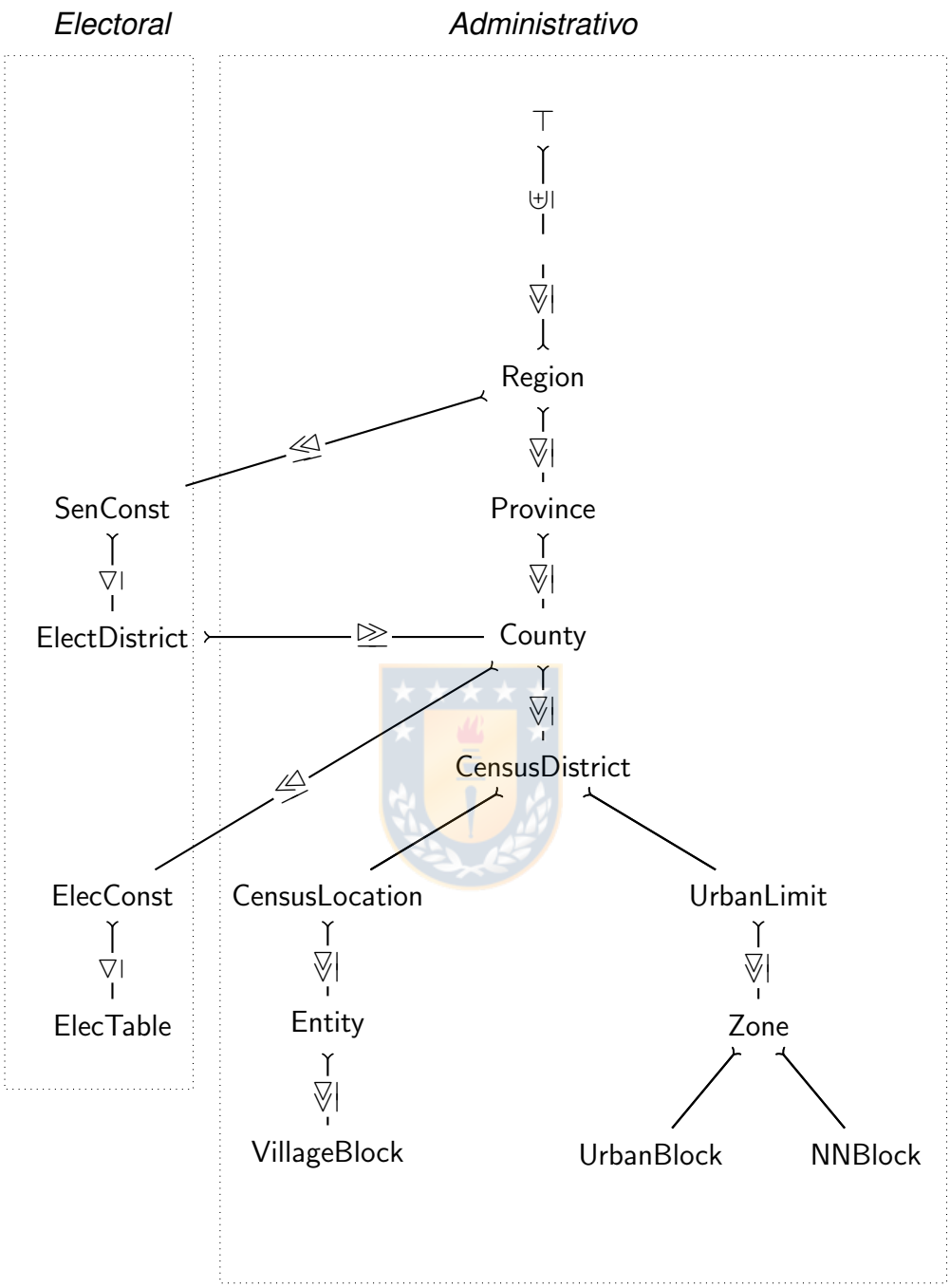


Figura 4.2: Jerarquía de granularidades Chile

original por índice creado. En la tabla 4.2 se muestra en detalle los datos para la implementación basada en un modelo relacional plano.

En la tabla 4.3 se encuentra un resumen del tamaño ocupado por

Sigla	Descripción
$\top$	Todos los paises del mundo
Chile	Pais Chile
SenConst	circunscripción senatorial
ElectDistrict	distrito electoral
ElecConst	circunscripción electoral
ElecTable	mesa electoral
Region	región administrativa
Province	provincia administrativa
County	comuna administrativa
CensusDistrict	distrito censal
CensusLocation	localidad censal
Entity	entidad censal
UrbanLimit	limite urbano censal
Zone	zona censal
VillageBlock	manzana aldea censal
UrbanBlock	manzana urbana censal
NNBlock	manzana sin información

Tabla 4.1: Descripción siglas jerarquía de granularidades

Tabla	Cant. tuplas
DivAdm(Reg,Prov,Com,DC,Ent,Area,Manz)	180499
DivEle(Reg,Sen,Dist,Com,Const,Mesa)	41379

Tabla 4.2: Datos implementación modelo relacional plano

las 3 implementaciones realizadas, considerar que para el cálculo del tamaño ocupado por la implementación en modelo relacional plano se consideró solo los atributos que poseen una contra-parte en las estructura multigranular; es decir, datos de votaciones o de población para la instancias electoral y censal, respectivamente, no fueron considerados; además, para las tres implementaciones los atributos poseen los mismos tipos de datos.

Se realizaron 4 tipos de consultas distintas para cada algoritmo:

1. Primer conjunto de consultas asegura que el retorno de cada

Implementación	Cant. tuplas	Tamaño (Mb)
Propuesta	263096	17.2
Sin uso reglas de inferencia	5577345015	358124.5
Relacional plana	221878	22.0

Tabla 4.3: Tamaño ocupado implementaciones

algoritmo sea $True$ y que se posee completitud entre las granularidades visitadas.

2. Segundo conjunto de consultas asegura que el retorno de cada algoritmo sea $False$ y que se posee completitud entre las granularidades visitadas.

3. Tercer conjunto de consultas asegura que el retorno de cada algoritmo sea $True$ y que no se posee completitud entre las granularidades visitadas.

4. Cuarto conjunto de consultas asegura que el retorno de cada algoritmo sea $False$ y que no se posee completitud entre las granularidades visitadas.

Por cada conjunto se realizaron un total de 50 consultas, agrupadas según la cantidad de gránulos analizados. La selección de gránulos se realizó de manera aleatoria asegurando solo que cumplieran con el retorno correspondiente según el conjunto al que pertenezcan. Las figuras 4.3, 4.4, 4.5 y 4.6 reflejan los resultados (en promedio) obtenidos para la implementación propuesta además de incluir una referencia del tiempo promedio de las otras dos implementaciones. La tabla 4.4 refleja en detalle del tiempo promedio obtenido por consulta, tanto para la implementación en modelo relacional plano, como para la implementación en modelo multigranular sin uso de reglas de inferencia.

Se observa un tiempo de ejecución acorde al análisis teorico realizado previamente para los peores casos en cada algoritmos. Para

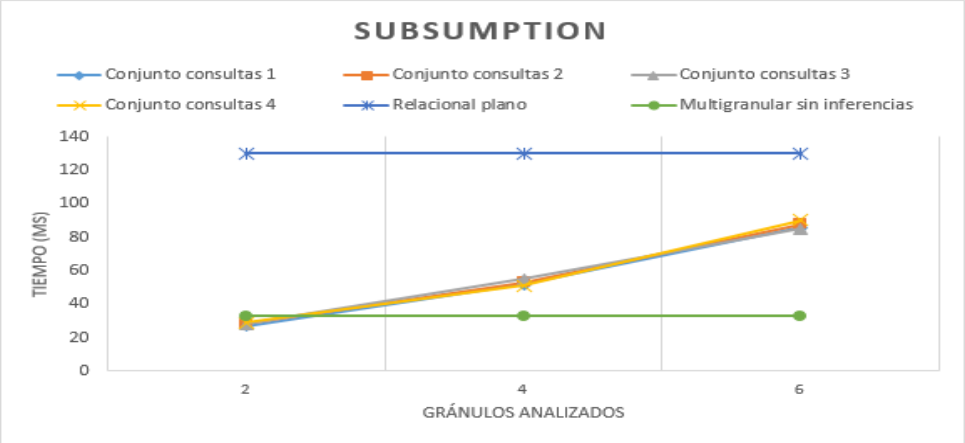


Figura 4.3: Gráfico función *subsumption* datos de división chilena

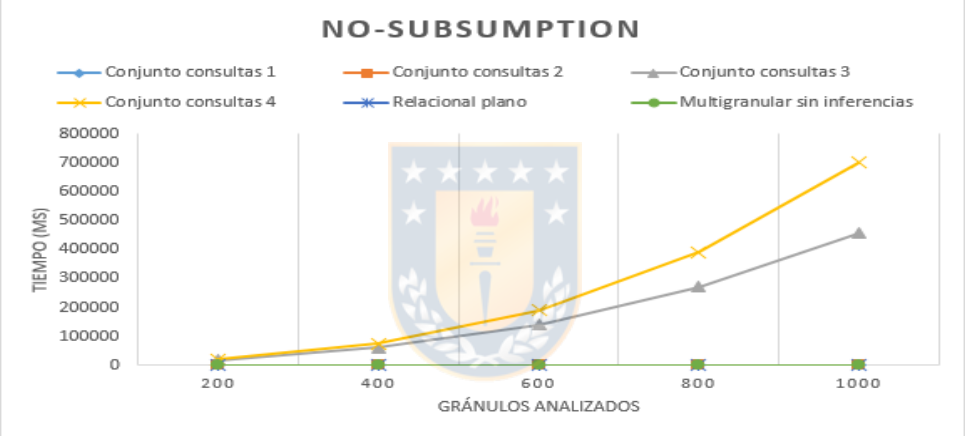


Figura 4.4: Gráfico función *no-subsumption* datos de división chilena

Implementación	C. tipo 1	C. tipo 2	C. tipo 3	C. tipo 4
Relacional plano	180	176	98	97
Sin uso reglas de inferencia	32	33	31	33

Tabla 4.4: Tiempo promedio por consulta (ms)

la consulta de *subsumption* y *disjoint* se poseen tiempos competitivos respecto a las otras dos implementaciones, aunque para *not-subsumption* se prueba que para consultas en las que no se posee completitud se posee un tiempo de ejecución demasiado lento mientras que para *not-disjoint* para toda consulta se tiene un tiempo de ejecución lento

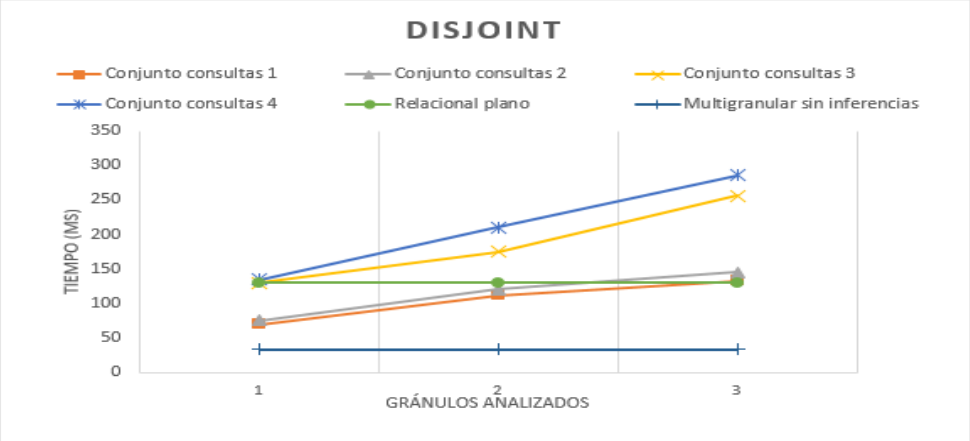


Figura 4.5: Gráfico función *disjoint* datos de división chilena

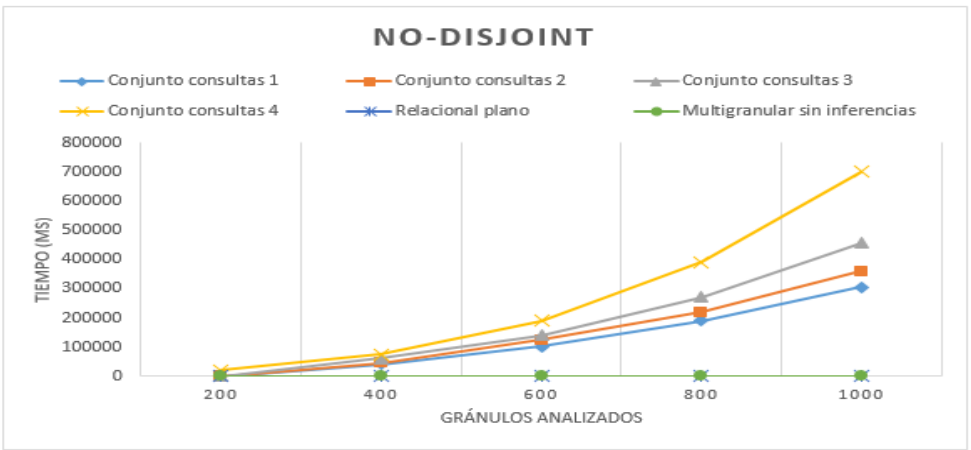


Figura 4.6: Gráfico función *no-disjoint* datos de división chilena

Capítulo 5

Conclusiones y trabajo futuro

En este trabajo se llevó a cabo una implementación del modelo multigranular propuesto en [10] desarrollando a la vez un esquema para la representación de la estructura multigranular. En la búsqueda de una implementación eficiente para el modelo multigranular se exploró la idea de disminuir la información almacenada en la estructura multigranular. Para esto, se desarrollaron reglas de inferencia y se analizó el impacto tanto de poseer completitud como la composición entre relaciones bigranulares. En base a lo anterior se desarrollaron algoritmos para responder las consultas de *subsumption* y *disjoint* entre gránulos. La evaluación experimental demostró un ahorro de espacio a utilizar para la representación de la estructura multigranular por parte de la implementación propuesta con respecto a otras dos implementaciones: una basada en un modelo relacional plano y una basada en modelo multigranular sin hacer uso de lo desarrollado en este documento. Se evaluaron las tres implementaciones frente una instancia real en donde se reflejó que la implementación propuesta posee tiempo competitivos con respecto a las otras dos evaluadas para las consultas sobre *subsumption* y *disjoint*, aunque para las respectivas negaciones de estas consultas, la implementación propuesta demostró un bajo desempeño en comparación a las otras implementaciones evaluadas.

Como trabajo futuro se propone analizar en detalle la completitud, debido a que se demostró el alto impacto que tenía el poseer completitud para los algoritmos. En este trabajo solo se analizó desde un

punto de vista práctico basado en su utilidad en la composición de relaciones bigranulares sin formalizar este concepto ni analizar de manera particular tanto para *subsumption* como para *disjoint*. También se considera interesante el diseño e implementación de una estructura que ayude a responder la consulta de *no-subsumption* y *no-disjoint* dado la similitud que se posee con el problema de intersección de intervalos, buscando de esta manera, alcanzar tiempos de respuesta competitivos en comparación a otras implementaciones, se propone además analizar la implementación del modelo multigranular para un caso general y no solo para relaciones bigranulares y evaluar el uso de distintos niveles de grados de inferencia dado que en este trabajo se priorizó el disminuir el espacio requerido sin evaluar el almacenar información extra para disminuir el tiempo de ejecución de los algoritmos.

Bibliografía

[1] Alberto Belussi, Carlo Combi, and Gabriele Pozzani. Formal and conceptual modeling of spatio-temporal granularities. In *IDEAS*, ACM International Conference Proceeding Series, pages 275–283. ACM, 2009.

[2] Claudio Bettini, Curtis E. Dyreson, William S. Evans, Richard T. Snodgrass, and Xiaoyang Sean Wang. A glossary of time granularity concepts. In *Temporal Databases, Dagstuhl*, pages 406–413, 1997.

[3] Claudio Bettini, Sushil Jajodia, and Xiaoyang Sean Wang. *Time Granularities in Databases, Data Mining, and Temporal Reasoning*. Springer, 2000.

[4] Loreto Bravo and M. Andrea Rodríguez. A multi-granular database model. In *FoIKS*, volume 8367 of *Lecture Notes in Computer Science*, pages 344–360. Springer, 2014.

[5] Elena Camossi, Michela Bertolotto, and Elisa Bertino. A multigranular object-oriented framework supporting spatio-temporal granularity conversions. *International Journal of Geographical Information Science*, 20(5):511–534, 2006.

[6] Martin Erwig and Markus Schneider. Partition and conquer. In *COSIT*, volume 1329 of *Lecture Notes in Computer Science*, pages 389–407. Springer, 1997.

[7] Jérôme Euzenat and Angelo Montanari. Time granularity. In *Handbook of Temporal Reasoning in Artificial Intelligence*, volume 1 of *Foundations of Artificial Intelligence*, pages 59–118. Elsevier, 2005.

[8] Marc Gyssens, Jan Paredaens, Jan Van den Bussche, and Dirk Van Gucht. A graph-oriented object database model. *IEEE Trans. Knowl. Data Eng.*, 6(4):572–586, 1994.

[9] S. J. Hegner and M. A. Rodríguez. Inferences rules for binary predicates in a multigranular database, 2018.

[10] Stephen J. Hegner and M. Andrea Rodríguez. A model for multigranular data and its integrity. *Informatica, Lith. Acad. Sci.*, 28(1):45–78, 2017.

[11] Stephen J. Hegner and M. Andrea Rodríguez. Implicit representation of bigranular rules for multigranular data. In *DEXA (1)*, volume 11029 of *Lecture Notes in Computer Science*, pages 372–389. Springer, 2018.

[12] Nadeem Iftikhar and Torben Bach Pedersen. Schema design alternatives for multi-granular data warehousing. In *DEXA (2)*, volume 6262 of *Lecture Notes in Computer Science*, pages 111–125. Springer, 2010.

[13] W. H. Inmon. *Building the data warehouse*. John wiley sons, 2005.

[14] Xiaoyang Sean Wang, Sushil Jajodia, and V. S. Subrahmanian. Temporal modules: An approach toward federated temporal databases. *Inf. Sci.*, 82(1-2):103–128, 1995.