UNIVERSIDAD DE CONCEPCIÓN
Dirección de Postgrado
Facultad de Ingeniería - Programa de Magíster en Ciencias de la Computación

OPTIMIZACIÓN SEMÁNTICA DE CONSULTAS EN BASES DE DATOS ESPACIALES

Tesis para optar al grado de Magíster
en Ciencias de la Computación

EDUARDO ANTONIO MELLA PRUSSING
CONCEPCIÓN-CHILE
2016

Profesor Guía: Andrea Rodríguez Tastets
Departamento de Ing. Civil Informática y
Ciencias de la Computación, Facultad de Ingeniería
Universidad de Concepción

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Resumen

Una base de datos espacial es una base de datos diseñada para almacenar y consultar datos que representan objetos definidos en un espacio geométrico, tales como puntos, líneas, polígonos, etc. Al igual que en las bases de datos relacionales, en una base de datos espacial existen restricciones de integridad asociadas a un conjunto de datos en particular. La optimización semántica de consulta es el proceso en el cual se utilizan estas restricciones de integridad u otro conocimiento semántico para reformular la consulta en una equivalente que entregue el mismo conjunto de respuestas en un menor tiempo de ejecución. En este proyecto se crearon estrategias para la optimización semántica de consultas de *rango* y *join* espacial utilizando restricciones de integridad, en particular, restricciones de dependencia topológica y restricciones de referencia topológica asociadas a la base de datos. Además se generaron e implementaron algoritmos que reflejan las estrategias propuestas. Se llevaron a cabo experimentos utilizando bases de datos reales obtenidas de la página web del gobierno estadounidense (U.S. Census Bureau), cuyos resultados muestran que en algunos casos es posible mejorar el tiempo de ejecución de una consulta en casi un 100 % con respecto a la consulta sin optimizar. Las estrategias propuestas en este proyecto logran ser una alternativa real en el ámbito de optimización semántica de consultas.

Capítulo 2

Introducción

2.1. Descripción del problema

Una base de datos espacial es una base de datos donde se puede almacenar y consultar datos que representan objetos definidos en un espacio geométrico. Estas bases de datos usan un modelo espacial (geométrico) de representación, siendo el más común el que permite representar objetos espaciales en forma de puntos, líneas y polígonos. Al igual que una base de datos tradicional, en una base de datos espacial existen restricciones de integridad que permiten definir si una instancia de la bases de datos es consistente o no. Estas restricciones de integridad son típicamente validadas al momento de efectuar actualizaciones a la base de datos.

Dada la complejidad de los datos espaciales, y dependiendo de la cantidad de datos presentes, una consulta espacial puede llegar a tomar mucho tiempo en ejecutarse. Dentro de las diferentes técnicas de optimización, la optimización semántica de consulta es el proceso en el cual se utilizan restricciones de integridad u otro conocimiento semántico para reformular la consulta en una consulta equivalente que entregue el mismo conjunto de respuestas en un menor tiempo de ejecución [1].

Este trabajo apunta a mejorar la eficiencia del procesamiento de las consultas en las bases de datos espaciales, reformulando una consulta inicial en base a las restricciones de integridad definidas por el diseñador de la base de datos. En particular, este trabajo se centra en dos tipos de restricciones de integridad semántica: las restricciones de dependencia topológica y las restricciones de referencia espacial. El objetivo general es reformular la consulta de tal forma que el conjunto de respuestas obtenidas sea el mismo, pero el tiempo de procesamiento de la consulta sea menor. Se busca definir estrategias de optimización semántica para las consultas

de rango y join espacial, generando los correspondientes algoritmos y realizando experimentos sobre datos geográficos reales.

2.2. Hipótesis y Objetivos

Hipótesis. Es posible optimizar el tiempo de procesamiento de las consultas de join y de rango en bases de datos espaciales a través de la reformulación de consultas equivalentes que toman en cuenta las consideración impuestas por restricciones de dependencia y de referencia topológica.

Objetivo general. El objetivo de esta tesis es definir formalmente y evaluar experimentalmente técnicas de optimización semántica de consultas sobre bases de datos espaciales utilizando restricciones de dependencia y de referencia topológica.

Objetivo específicos.

1. Comprender el modelo de bases de datos espacial y las restricciones de integridad definidas en [3].
2. Definir dos estrategias de optimización: una para optimizar consultas de rango espacial y otra para optimizar consultas de join espacial.
3. Generar y evaluar algoritmos que reflejen las estrategias definidas.
4. Implementar los algoritmos generados en un sistema gestor de bases de datos espaciales.
5. Evaluar el rendimiento de las consultas generadas en relación a el tiempo de ejecución de la consulta original realizando experimentos usando datos reales.

2.3. Organización del informe

- Capítulo 3: Marco Teórico. Este capítulo presenta los conceptos necesarios para entender el problema a abordar y revisa algunos trabajos relacionados al tema.
- Capítulo 4: Estrategias de Optimización. Este capítulo define en detalle las estrategias creadas para la optimización de las consultas de rango y de join espacial.
- Capítulo 5: Algoritmos Propuestos. Este capítulo presenta los algoritmos creados para las diferentes estrategias, demostrando que las aplicación de estos algoritmos genera una consulta equivalente a la consulta original.

- Capítulo 6: Experimentos y Resultados. Este capítulo muestra los experimentos realizados y analiza los resultados obtenidos.
- Capítulo 7: Conclusiones. Para finalizar en este capítulo se muestran los conclusiones del proyecto y el trabajo futuro a realizar.

Capítulo 3

Marco Teórico

Este capítulo está dividido en dos secciones, en la primera se presentan definiciones y conceptos básicos relacionados con el tema a tratar y en la segunda sección se revisan algunos trabajos relacionados con la optimización semántica de consultas.

3.1. Preliminares

Base de Datos Espacial. Una base de datos espacial define tipos de datos geométricos para representar objetos localizados espacialmente y provee índices y funciones especiales para manipular estos datos. Usualmente estas bases de datos se definen como extensiones al modelo relacional u objeto-relacional [13].

Siguiendo la formalización dada en [3], un esquema de base de datos espacio-relacional puede ser formalmente definido como una tupla $\Sigma = (\mathcal{U}, \mathcal{A}, \mathcal{S}, \mathcal{R}, \mathcal{T})$, donde:

(a) $\mathcal{U}$ es el posible infinito dominio de la base de datos que incluye $\mathbb{R}$.
(b) $\mathcal{A} = \{A_1, \ldots, A_n\}$, donde cada A_i es un atributo temático que toma valores en $\mathcal{U}$.
(c) $\mathcal{S} = \{S_1, \ldots, \mathcal{S}_m\}$, donde cada S_i es un atributo espacial que toma valores en $\mathcal{P}(\mathbb{R}^2)$, el power set de $\mathbb{R}^2$. Los tipos de datos para representar objetos espaciales más utilizados en bases de datos espaciales son los puntos, líneas y polígonos.
(d) $\mathcal{R}$ es un conjunto finito de predicados, cada uno de ellos con un conjunto finito y ordenado de atributos pertenecientes a $\mathcal{A}$ o $\mathcal{S}$. $R(B_1, \ldots, B_l)$ representa el predicado con su conjunto de atributos.
(e) $\mathcal{T}$ es un conjunto de predicados topológicos binarios.

Una instancia de base de datos D de un esquema Σ es una colección finita de

átomos de la forma $R(c_1, \ldots, c_i, \ldots, c_l)$, donde (a) $R(B_1, \ldots, B_i, \ldots, B_l) \in \mathcal{R}$, (b) si $B_i \in \mathcal{A}$ entonces $c_i \in \mathcal{U}$ y (c) si $B_i \in \mathcal{S}$, entonces $c_i \in Ad \subseteq \mathcal{P}(\mathbb{R}^2)$, con Ad la clase de geometrías admisibles para una base de datos espacial. En este trabajo, nos centraremos en geometrías que representan regiones que tienen una extensión geográfica.

Ejemplo 3.1. Consideremos una base de datos espacial que almacene información sobre los límites administrativos de un país usando el siguiente esquema de base de datos $\mathcal{R} = \{\text{State}(Id_s, Name_s, Area, G_s), \text{County}(Id_c, Name_c, Id_s, G_c)\}$. En el predicado State, los atributos temáticos son Id_s, $Name_s$, y $Area$ (en km^2), y el único atributo espacial es G_s, que almacena los límites del estado. De la misma forma, el predicado County tiene Id_c, $Name_c$, y Id_s como atributos temáticos y G_c como atributo espacial. La Figura 3.1 muestra una instancia de este esquema. En ella, las líneas más gruesas representan los límites de los estados y las líneas discontinuas representan los límites de los condados.

State	Id_s	$Name_s$	*Area*	G_s
	s_1	$name_1$	1200	s_1
	s_2	$name_2$	1500	s_2

County	Id_c	$Name_c$	Id_s	G_c
	c_1	$name_3$	s_1	g_1
	c_2	$name_4$	s_1	g_2
	c_3	$name_5$	s_1	g_3
	c_4	$name_6$	s_2	g_4
	c_5	$name_7$	s_2	g_5
	c_6	$name_8$	s_2	g_6

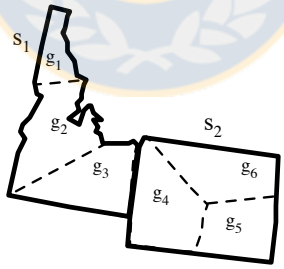

Figura 3.1: Instancia de Base de Batos

□

Relaciones Topológicas Los objetos espaciales se relacionan entre si mediante relaciones espaciales, las cuales son usualmente clasificadas en relaciones de distancia, orientación o topológicas [17, 19]. De todas estas relaciones, las relaciones topológicas han captado la mayor atención de las investigaciones [7, 15] debido a su gran uso en describir configuraciones espaciales y a que actualmente son implementadas en los lenguajes SQL espaciales (SSQLs) [14].

Tal como mencionamos anteriormente existen diferentes tipos de objetos espaciales, por lo tanto diferentes relaciones topológicas pueden definirse dependiendo de los tipos de objetos involucrados. Para los atributos geométricos que representan polígonos o regiones no vacías en el espacio, existen ocho relaciones topológicas binarias básicas[6]: Overlaps(O), CoveredBy(CB), Equal(EQ), Inside(IS), Covers(CV), Includes(IC), Touches(TO), Disjoint(D). Se ha demostrado que para dos regiones cualesquiera, al menos una de estas ocho relaciones debe cumplirse y que además esta relación es única [5].

Además de estas ocho relaciones básicas existen otras relaciones, denominadas relaciones derivadas, que pueden ser definidas como una disyunción de relaciones básicas. Por ejemplo, Contains(C) y Within(W), donde Contains es la unión de Equal, Includes y Covers, y Within es la unión de Equal, Inside y CoveredBy. Estas dos nuevas relaciones son de mucha utilidad y son utilizadas en los sistemas de consultas de datos [3]. La Figura 3.2 muestra ejemplos gráficos de las relaciones topológicas básicas y la Figura 3.3 muestra las relaciones derivadas, otra relaciones derivadas existentes son Intersects(IT) e InternallyDisjoint(IDC).

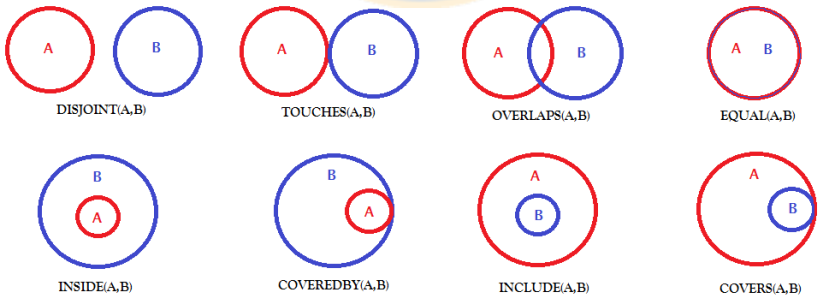

Figura 3.2: Relaciones topólogicas entre regiones

Otro concepto fundamental al momento de tratar con objetos espaciales y relaciones espaciales es la *composición* de relaciones topológicas. La composición de dos

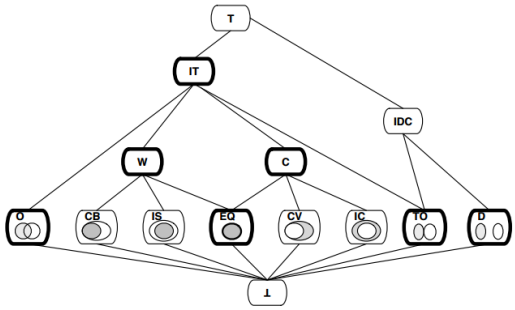

Figura 3.3: Relaciones derivadas

relaciones topológicas $T_1(A, B)$ y $T_2(B, C)$ sobre un objeto en común B, denotada por $T_1(A, B) \otimes T_2(B, C)$, permite la derivación de una relación topológica entre los objetos A y C. La composición de las relaciones topológicas básicas es mostrada en el Cuadro 3.1. Si, por ejemplo, $\mathsf{Overlaps}(A, B)$ y $\mathsf{CoveredBy}(B, C)$ se cumplen, obtenemos de la tabla de composición que la siguiente disyunción de relaciones $\{\mathsf{Overlaps}(A, C), \mathsf{CoveredBy}(A, C), \mathsf{Inside}(A, C)\}$ debe cumplirse también, es decir, la relación topológica entre A y C debe ser una de esas tres. Denotamos la composición entre dos relaciones, ejemplo Overlaps y CoveredBy, de la siguiente forma:

$$\begin{aligned}\mathsf{Overlaps} \otimes \mathsf{CoveredBy} &= (\mathsf{Overlaps} \vee \mathsf{CoveredBy} \vee \mathsf{Inside}) \\ &= \{\mathsf{Overlaps}, \mathsf{CoveredBy}, \mathsf{Inside}\}.\end{aligned}$$

-	(1)D	(2)TO	(3)O	(4)CB	(5)CV	(6)W	(7)C	(8)EQ
(1)D	Todas	1,2,3,4,5	1,2,3,4,5	1,2,3,4,5	1,2,3,4,5	1	1	1
(2)TO	1,2,3,6,7	1,2,3,5,7,8	1,2,3,4,5	3,4,5	2,3,4,5	1	1,2	2
(3)O	1,2,3,6,7	1,2,3,6,7	Todas	3,4,5	3,4,5	1,2,3,6,7	1,2,3,6,7	3
(4)IS	1	1	1,2,3,4,5	4	4	Todas	1,2,3,4,5	4
(5)CB	1	1,2	1,2,3,4,5	4	4,5	1,2,3,6,7	1,2,3,5,7,8	5
(6)IC	1,2,3,6,7	3,6,7	3,6,7	3,4,5,6,7,8	3,6,7	6	6	6
(7)CV	1,2,3,6,7	2,3,6,7	3,6,7	3,4,5	3,5,7,8	6	6,7	7
(8)E	1	2	3	4	5	6	7	8

Cuadro 3.1: Composición de relaciones topológicas básicas

Para tratar de mejor manera las relaciones topológicas en la optimización de consultas, en este trabajo se hace interesante considerar la disyunción explícita de relaciones topológicas. Por ejemplo, en una consulta espacial uno podría estar interesado en buscar dentro de una base de datos todos los pares de regiones que sean geométricamente disjuntas o que sus bordes estén intersectados, lo que se expresa por la disyunción $\mathsf{Disjoint}(g_1, g_2) \vee \mathsf{Touches}(g_1, g_2)$, y donde g_1 y g_2 son variables

que representan el atributo espacial de las regiones.

Definición 3.1. Considerando una expresión topológica como una disyunción de una o más posibles relaciones topológicas, definimos la relación de Inclusion, denotada por $\subseteq$, la cual especifica si una expresión topológica es parte de otra expresión topológica. Por ejemplo:

$$\{\mathsf{Touches}\} \subseteq \{\mathsf{Touches}, \mathsf{Overlaps}\}$$
$$\{\mathsf{Touches}\} \subseteq \{\mathsf{Touches}, \mathsf{Overlaps}, \mathsf{Within}\}$$
$$\{\mathsf{Disjoint}\} \subseteq \{\mathsf{Touches}, \mathsf{Disjoint}\}$$
$$\{\mathsf{Overlaps}\} \subseteq \{\mathsf{Contains}, \mathsf{Overlaps}\}$$
$$\{\mathsf{Overlaps}\} \subseteq \{\mathsf{Overlaps}\}$$

Notar que la relación de inclusión también se aplica a las relaciones derivadas introducidas anteriormente. En particular, la relación Within y la relación Contains se definen a partir de la disyunción de otras tres relaciones y, por lo tanto, dado que $\mathsf{Within} = \{\mathsf{Equal}, \mathsf{CoveredBy}, \mathsf{Inside}\}$ y $\mathsf{Contains} = \{\mathsf{Equal}, \mathsf{Covers}, \mathsf{Includes}\}$ también se cumple:

$$\{\mathsf{Equal}\} \subseteq \mathsf{Within}$$
$$\{\mathsf{Equal}\} \subseteq \mathsf{Contains}$$

□

Definición 3.2. Además de establecer la relación de Inclusion, también se puede establecer la Intersection, denotada por $\cap$, entre dos expresiones topológicas. Esta relación establece si existe una sub expresión que sea común a dos expresiones topológicas. Por ejemplo:

$$\{\mathsf{Touches}\} \cap \{\mathsf{Touches}, \mathsf{Overlaps}\} = \{\mathsf{Touches}\}$$
$$\{\mathsf{Touches}, \mathsf{Overlaps}\} \cap \{\mathsf{Overlaps}\} \cap \mathsf{Within} = \{\mathsf{Overlaps}\}$$
$$\{\mathsf{Disjoint}\} \cap \{\mathsf{Touches}, \mathsf{Disjoint}\} = \{\mathsf{Disjoint}\}$$
$$\{\mathsf{Overlaps}\} \cap \{\mathsf{Overlaps}\} = \{\mathsf{Overlaps}\}$$

Al igual que el caso anterior, especial atención hay que poner en la relaciones derivadas Within y Contains, donde en estos casos se cumple, por ejemplo:

$$\{\mathsf{Equal}\} \cap \mathsf{Within} = \{\mathsf{Equal}\}$$
$$\mathsf{Contains} \cap \{\mathsf{Equal}\} = \{\mathsf{Equal}\}$$
$$\mathsf{Within} \cap \mathsf{Contains} = \{\mathsf{Equal}\}$$

□

Relaciones Topológicas utilizadas en PostGIS PostGIS es uno de los sistemas de bases de datos espaciales libre más utilizados actualmente. Este sistema es una extensión de PostgreSQL el cual trabaja sobre bases de datos relacionales. En este proyecto se utilizará PostGIS para la realización de los experimentos.

En el sistema de bases de datos espacial PostGIS, se utilizan las relaciones derivadas Contains y Within y, además, las relaciones Includes e Inside tienen una relación de Inclusion con las relaciones Covers y CoveredBy, respectivamente, ya que en PostGIS se tiene:

$$\mathsf{Covers}_{PostGIS} = \{\mathsf{Includes}, \mathsf{Covers}\}$$
$$\mathsf{CoveredBy}_{PostGIS} = \{\mathsf{Inside}, \mathsf{CoveredBy}\}$$

En total el sistema de base de datos PostGIS utiliza las siguientes 8 relaciones topológicas: Overlaps, Equal, Touches, Disjoint, Covers, CoveredBy, Contains y Within. Nótese que, dado este conjunto de relaciones es perfectamente posible que dos regiones cualesquiera estén relacionadas a través de dos relaciones diferentes, por ejemplo $\mathsf{Equal}(A, B)$ y $\mathsf{Within}(A, B)$.

Considerando las 8 relaciones utilizadas por PostGIS, la tabla de composición derivada queda según lo mostrado en el Cuadro 3.2

Restricciones de Integridad. Las restricciones de integridad son restricciones aplicadas sobre los datos de la base de datos. Estas restricciones se definen con el propósito de establecer estados válidos o instancias válidas de una base de datos. La preservación de la integridad de una bases de datos, o el estado consistente de ella, se asegura mediante la satisfacción de las restricciones de integridad, las que son usualmente evaluadas cada vez que se hace una modificación a la base de datos.

-	(1)D	(2)TO	(3)O	(4)CB	(5)CV	(6)W	(7)C	(8)EQ
(1)D	Todas	1,2,3,4	1,2,3,4	1,2,3,4	1	1,2,3,4	1	1
(2)TO	1,2,3,5	Todas	1,2,3,4	2,3,4	1,2	2,3,4	1,2	2
(3)O	1,2,3,5	1,2,3,5	Todas	3,4	1,2,3,5	3,4	1,2,3,5	3
(4)CB	1	1,2	1,2,3,4	4	Todas	4	Todas	4
(5)CV	1,2,3,5	2,3,5	3,5	3,6,7	5	3,6,7	5	5
(6)W	1	2	3	4	5	6	6,7	6
(7)C	1	2	3	4	5	6,7	7	7
(8)E	1	2	3	4	5	6	7	8

Cuadro 3.2: Composición de relaciones en PostGIS

A continuación presentamos cuatro tipos de restricciones de integridad escritas según la formalización propuesta en [3] y que están orientadas al manejo de restricciones con condiciones establecidas por relaciones topológicas.
Para la definición de las restricciones de integridad se utiliza la siguiente notación: (i) variables x, y, z, x_1, x', ... representan valores en $\mathcal{U}$ o en Ad; (ii) variables g, g_1, g', ... representan valores en Ad; (iii) variables u, v, w, u_1, u', ... representan valores en $\mathcal{U}$; y (iv) símbolos $\bar{x}$, $\bar{u}$, $\bar{x}_1$, ... denotan una secuencia posiblemente vacía de variables distintas o, con un pequeño abuso de notación, un conjunto de variables. En algunos casos utilizaremos $\overline{\forall}$ para denotar que todas las variables libres de una fórmula están cuantificadas universalmente.

Definición 3.3. Dado un esquema $\Sigma = (\mathcal{U}, \mathcal{A}, \mathcal{S}, \mathcal{R}, \mathcal{T})$, y predicados $P, R \in \mathcal{R}$ y $T \in \mathcal{T}$, una *restricción de integridad semántica* (RIS) tiene una de las siguientes formas:

Dependencia Funcional (FD)
$\forall \bar{u}\bar{y}_1\bar{y}_2 v_1 v_2 (P(\bar{u}, \bar{y}_1, v_1) \wedge P(\bar{x}, \bar{y}_2, v_2) \wedge \bar{u} = \bar{x} \rightarrow v_1 = v_2)$,
donde $\bar{y}_1 \cap \bar{y}_2 = \emptyset$ y $|\bar{u}| > 0$.
Dependencia de Inclusión (IND):
$\forall \bar{u}\bar{x}(P(\bar{u}, \bar{x}) \rightarrow \exists \bar{v}\bar{y} R(\bar{v}, \bar{y}) \wedge \bar{x} = \bar{v})$.
Dependencia Topológica (TD):
$\forall \bar{x}\bar{y} g_1 g_2 (P(\bar{x}, g_1) \wedge R(\bar{y}, g_2) \wedge \varphi \rightarrow \mathcal{T}(g_1, g_2))$
donde $\bar{x} \cap \bar{y} = \emptyset$, y φ es una fórmula CNF posiblemente vacía de igualdades o desigualdades entre variables en $\bar{x}$ y/o $\bar{y}$.
en alguno casos nos referiremos como TD, $TD_=$ o $TD_\neq$ cuando tengamos una Dependencia Topológica donde la variable φ sea vacía, una igualdad de atributos o una desigualdad de atributos, respectivamente.

Spatial Referential Dependency (RD):
$\forall \bar{x}\bar{v} g_1 (P(\bar{x}, \bar{v}, g_1) \rightarrow \exists \bar{y}\bar{u} g_2 R(\bar{y}, \bar{u}, g_2) \wedge \bar{v} = \bar{y} \wedge T(g_1, g_2))$.

Ejemplo 3.2. (Cont. Ejemplo 3.1) Las siguientes restricciones de integridad pueden ser definidas usando el esquema de bases de datos del Ejemplo 3.1:

$$\overline{\forall}\,(\mathrm{State}(u,v,w,g) \wedge \mathrm{State}(u',v',w',g') \wedge u = u' \rightarrow v = v') \tag{3.1}$$

$$\overline{\forall}\,(\mathrm{State}(u,v,w,g) \wedge \mathrm{State}(u',v',w',g') \wedge u = u' \rightarrow w = w') \tag{3.2}$$

$$\overline{\forall}\,(\mathrm{State}(u,v,w,g) \wedge \mathrm{State}(u',v',w',g') \wedge u = u' \rightarrow \mathsf{Equal}(g,g')) \tag{3.3}$$

$$\overline{\forall}\,(\mathrm{County}(u,v,w,g) \rightarrow \exists u'v'zg'\,(\mathrm{State}(u',v',z,g') \wedge w = u' \wedge \mathsf{Within}(g,g'))) \tag{3.4}$$

□

Las FDs e INDs corresponden a restricciones tradicionales dado que no contienen joins ni referencias entre los atributos espaciales. De hecho, los atributos espaciales no son usados como claves ni como atributos referenciales dado que eso tendría un alto costo computacional asociado en términos de espacio y tiempo. Además, desde un punto de vista práctico, los valores de los atributos espaciales por si mismos no identifican objetos espaciales, por lo tanto necesitan alguna clase de clave temática o id.
Las TDs imponen una relación topológica entre atributos espaciales, posiblemente en el mismo predicado y las RDs imponen una dependencia de inclusión donde el atributo espacial de la tabla referenciada debe tener una propiedad espacial en particular. En el Ejemplo 3.2 las FDs (3.1) y (3.2) juntas con la TD (3.3) imponen que *u* sea una clave de la tabla State. La RD (3.4) impone que por cada condado, deba existir un estado que lo incluya.

Restricciones de Integridad Acíclicas. Que un conjunto de restricciones de integridad sea *acíclico* es una propiedad importante a la hora de establecer mecanismos de satisfacción de restricciones de integridad y de optimización de consultas.

Definición 3.4. Dado un esquema $\Sigma = (\mathcal{U}, \mathcal{A}, \mathcal{S}, \mathcal{R}, \mathcal{T})$, un conjunto de restricciones de integridad Ψ, el *grafo de dependencia* de Ψ, denotado por $\mathcal{G}(\Psi)$, es un dígrafo donde cada nodo es un predicado en $\mathcal{R}$ con un arco desde el predicado P al predicado R si existe una restricción RD: $\overline{\forall}(P(\bar{u},\bar{x},g_1) \rightarrow \exists \bar{v}\bar{y}g_2(R(\bar{v},\bar{y},g_2) \wedge \bar{x} = \bar{v} \wedge \mathcal{T}(g_1,g_2)))$ en Ψ. □

Definición 3.5. Un conjunto de restricciones de integridad se denomina *acíclico* si su grafo de dependencia es acíclico. [1]. □

Ejemplo 3.3. El siguiente conjunto de restricciones de integridad es un conjunto que no es acíclico.

$ic_1 : \forall u, x, g_p\ (P(u, x, g_p) \rightarrow \exists\, v, y, g_r R(v, y, g_r) \wedge x = v \wedge T(g_p, g_r))$.
$ic_2 : \forall w, z, g_r\ (R(w, z, g_r) \rightarrow \exists\, s, t, g_s S(s, t, g_s) \wedge z = s \wedge T'(g_r, g_s))$.
$ic_3 : \forall l, m, g_s\ (S(l, m, g_s) \rightarrow \exists\, p, o, g_p P(p, o, g_p) \wedge m = p \wedge T''(g_s, g_p))$.

En esta definición solo se consideran las restricciones de integridad espaciales a la hora de construir el grafo de dependencia. Esta propiedad será de utilidad más adelante cuando mostremos las técnicas propuestas para la optimización de consultas.

Consultas Espaciales. Este trabajo se centra en dos tipos de consultas comúnmente utilizadas sobre bases de datos espaciales : Join Query (JQ) y Range Query (RQ), las cuales se expresan en forma conjuntiva de la siguiente forma:

Join Query (JQ):
$q(\bar{u}, g_1, g_2) : \exists \bar{z} P(\bar{x_1}, g_1) \wedge R(\bar{x_2}, g_2) \wedge T(g_1, g_2)$, donde $\bar{u} \cup \bar{z} = \bar{x_1} \cup \bar{x_2}$.

En las consultas de tipo JQ se busca recuperar las tuplas de la base de datos cuyos atributos espaciales cumplan cierta relación entre sí.

Ejemplo 3.4. (Cont. Example 3.1) Tomando en consideración la instancia de base de datos de la Figura 3.1, se puede definir la siguiente consulta de tipo Join:

$q(Id_s, Id_c, G_s, G_c) : \exists Name_s, Area, Name_c, Id'_s\ State(Id_s, Name_s, Area, G_s) \wedge$
$County(Id_c, Name_c, Id'_s, G_c) \wedge \mathsf{Within}(G_c, G_s)$

Esta consulta realizará la combinación de cada tupla en la tabla State con cada tupla en la tabla County y devolverá aquellas que cumplan con que el atributo espacial de County esté *Within* con el atributo espacial de State. En este caso la consulta anterior retornará las siguientes seis tuplas: $< s_1, c_1, s_1, g_1 >$, $< s_1, c_2, s_1, g_2 >$, $< s_1, c_3, s_1, g_3 >$, $< s_2, c_4, s_2, g_4 >$, $< s_2, c_5, s_2, g_5 >$ y $< s_2, c_6, s_2, g_6 >$ □

Range Query (RQ):
$q(\bar{u}, g) : \exists \bar{z} P(\bar{x}, g) \wedge T(g, w)$, donde $\bar{x} = \bar{z} \cup \bar{u}$, w es un parámetro espacial arbitrario y T una relación topológica.

En las consultas de tipo RQ se busca recuperar las tuplas de la base de datos cuyo atributo espacial cumpla cierta relación con el parámetro espacial w.

Ejemplo 3.5. (Cont. Example 3.1) Tomando en consideración la instancia de base de datos de la Figura 3.1 se puede definir la siguiente consulta de rango:

$q(Id_c, G_c) : \exists Name_c, Id_s\ County(Id_c, Name_c, Id_s, G_c) \wedge Overlaps(w, G_c).$

Esta consulta devolverá todas las tuplas de la tabla County cuyo atributo espacial G_c esté *Overlap* con el parámetro w, donde w representa un objeto geométrico bien definido. Tomando en cuenta el parámetro w de la Figura 3.4, representado por un rectángulo rojo, la consulta anterior retornará las siguientes tuplas: $< c_2, g_2 >$ y $< c_3, g_3 >$.

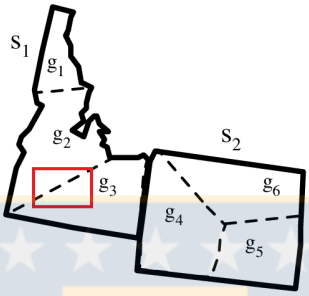

Figura 3.4: Consulta de rango

□

3.2. Trabajo relacionado

Existen varias estrategias en el ámbito de bases de datos relacionales para la optimización de consultas, entre ellas, eliminación de joins, reordenamiento de joins, eliminación de predicados, eliminación de clausulas DISTINCT etc. [11][10][18], las cuales se basan en la re-escritura de la consulta. Además existen estrategias de optimización en base a los índices en las tablas, tanto para bases de datos relacionales como para bases de datos espaciales [2][8]. Un ejemplo optimización semántica es una estrategia para detectar si una consulta tiene un conjunto de respuestas vacía analizando la inconsistencia de la consulta con respecto a las restricciones de integridad.

A pesar de que existe mucho trabajo hecho en el contexto de bases de datos relacionales para la optimización semántica de consultas [4], no existe mucho trabajo relacionado en el contexto de bases de datos espaciales. El trabajo en [12] investiga un enfoque general para la optimización semántica de consultas en bases de datos relacionales extendidas, utilizan las restricciones de integridad para dividir el proceso de consulta sobre consultas conjuntivas con el objeto de reducir el número de evaluaciones de ciertas condiciones. La idea principal es, dada una consulta Q,

verificar si existe una consulta Q' menos costosa que Q y que sea capaz de responder parcialmente la consulta original para luego dividir el proceso en tres etapas (i) responder Q' e ignorar la combinación de tuplas generadas, (ii) chequear posibles respuestas en las combinaciones de tuplas restantes, (iii) combinar las respuestas. A diferencia de este estudio, en este trabajo nos centraremos solo en bases de datos espaciales, y utilizaremos las propiedades de este dominio, tales como restricciones de integridad espaciales, relaciones topológicas y composición de relaciones, para reformular las consultas y de ese modo reducir el costo computacional de la consulta.

Una estrategia muy conocida de optimización semántica de consulta en bases de datos relacionales es la conocida como Chase & Backchase (C&B) [1]. La idea es, dada una consulta q, utilizar la técnica Chase junto con las restricciones de integridad asociadas a una base de datos para *expandir* la consulta, es decir, añadirle condiciones de búsqueda según lo que indiquen las restricciones de integridad. Un ejemplo de la aplicación del Chase es el siguiente.

Ejemplo 3.6. Consideremos una base de datos con el siguiente esquema $\mathcal{R}$: $\{P(x_1, x_2, x_3, x_4, x_5), R(y_1, y_2), Q(z_1, z_2, z_3, z_4), S(s_1, s_2, s_3, s_4), L(l_1, l_2, l_3)\}$ y las siguientes restricciones de integridad asociadas a ella:
ic_1: $\overline{\forall}(P(x_1, x_2, x_3, x_4, x_5) \rightarrow \exists y_1, y_2 R(y_1, y_2) \wedge x_4 = y_1)$,
ic_2: $\overline{\forall}(P(x_1, x_2, x_3, x_4, x_5) \wedge R(y_1, y_2) \wedge Q(z_1, z_2, z_3, z_4) \rightarrow \exists s_1, .., s_4, l_1, .., l_3$
$S(s_1, s_2, s_3, s_4) \wedge L(l_1, l_2, l_3) \wedge z_1 = s_1 \wedge l_1 = y_1)$,

Dada una consulta q definida de la siguiente forma:
$q(x_1, z_1) : P(x_1, x_2, x_3, x_4, x_5) \wedge Q(z_1, z_2, z_3, z_4) \wedge x_2 = z_1$

En una primera instancia se aplica Chase a la consulta q utilizando la restricción ic_1, generando la consulta
$q'(x_1, z_1) : P(x_1, x_2, x_3, x_4, x_5) \wedge Q(z_1, z_2, z_3, z_4) \wedge x_2 = z_1 \wedge R(y_1, y_2) \wedge x_4 = y_1$.

Nuevamente es posible aplicar Chase sobre los nuevos predicados generados, utilizando la restricción ic_2, con lo cual la consulta quedaría de la siguiente forma:
$q'(x_1, z_1) : P(x_1, x_2, x_3, x_4, x_5) \wedge Q(z_1, z_2, z_3, z_4) \wedge x_2 = z_1 \wedge R(y_1, y_2) \wedge x_4 = y_1 \wedge S(s_1, s_2, s_3, s_4) \wedge L(l_1, l_2, l_3) \wedge z_1 = s_1 \wedge l_1 = y_1$.

Esta es la forma expandida de la consulta q, ya que se aplicó el Chase a todos sus predicados. □

Ha sido demostrado[1] que la consulta expandida resultante de la aplicación de la técnica Chase es equivalente a la consulta original q, es decir, aplicada sobre la

misma instancia de alguna base de datos se obtiene el mismo conjunto de respuesta. El algoritmo Chase & Backchase definido en [1] realiza esta etapa de Chase y luego a la consulta expandida se la aplica la etapa de Backchase la cual, utilizando las misma restricciones de integridad, se encarga de obtener todas las subconsultas que sean equivalentes a la consulta original. Esto se logra quitando ciertos predicados de la consulta expandida con la condición de que las variables de la consulta original sigan presentes. Luego, con un modelo de costo, es posible escoger la subconsulta equivalente más eficiente a la hora de ejecución.

Ejemplo 3.7. (Cont. Ejemplo 3.6)
Tomando la consulta expandida $q'(x_1, z_1)$ del ejemplo anterior, la aplicación de la etapa del Backchase a la consulta generará por ejemplo, las siguientes subconsultas:
$q_1(x_1, z_1) : P(x_1, x_2, x_3, x_4, x_5) \wedge Q(z_1, z_2, z_3, z_4) \wedge S(s_1, s_2, s_3, s_4) \wedge x_2 = z_1 \wedge$
$z_1 = s_1$.
$q_2(x_1, s_1) : P(x_1, x_2, x_3, x_4, x_5) \wedge S(s_1, s_2, s_3, s_4) \wedge x_2 = s_1$.

□

Esta tesis pone especial énfasis en la técnica Chase, utilizándola para ciertas restricciones de integridad, en particular, las restricciones de inclusión topológica y las restricciones de dependencias funcionales. Además un aspecto importante a tener en cuenta es que es posible asegurar que esta técnica terminará si la restricciones de integridad utilizadas son acíclicas [1].

Teorema 3.1. Si el conjunto de restricciones de integridad de inclusión y de dependencias funcionales asociados a una base de datos es acíclico, entonces la técnica Chase aplicada sobre cualquier consulta conjuntiva terminará [1]. □

El anterior teorema se debe a que el Chase aplica iterativamente todas las restricciones de integridad asociadas a una base de datos cada vez que se genera un nuevo predicado, por lo tanto, si las restricciones no fueran acíclicas este proceso se llevaría a cabo infinitas veces.

Como hemos dicho anteriormente, y en nuestro conocimiento, no existen trabajos formales de optimización semántica para bases de datos espaciales. De hecho hasta hace poco no existía una formalización de las restricciones de integridad espaciales de tal forma que su especificación fuese uniforme y que permitiera analizar problemas de consistencia de estas restricciones. Otro problema que aún está abierto a la literatura es el problema de implicancia de restricciones de integridad, es decir, el problema de saber si una restricción de semántica espacial se puede deducir de un

conjunto de restricciones.

El trabajo propuesto en [3] realiza una especificación de las restricciones de integridad espaciales basada en la extensión de dependencias funcionales y dependencias de inclusión de bases de datos normales para soportar atributos espaciales. Dicha formalización está hecha en lógica de primer orden y sirve como base para este trabajo.

Capítulo 4

Estrategias de Optimización

En este capítulo se explican las estrategias propuestas para la optimización de las consultas. Para cada tipo de consulta (RQ y JQ) existirá una estrategia y algoritmos que la implementan y que son presentados luego en el Capítulo 6.

La optimización semántica de consultas se basa en utilizar algún conocimiento semántico para mejorar el tiempo de ejecución de una consulta dada. En nuestro caso, el conocimiento semántico se ve representado en las restricciones de integridad espaciales, en particular, nos centraremos en las restricciones de tipo RD y TD. Además, las estrategias de optimización que presentaremos a continuación requieren que el conjunto de restricciones a utilizar sea *acíclico*, y además que la base de datos donde se vaya a aplicar la optimización satisfaga dichas restricciones.

En general, en las estrategias propuestas se busca que, dada una consulta q y alguna instancia I de una base de datos espacial BD, determinar si la consulta q aplicada sobre I devuelve un conjunto de respuesta vacío sin necesidad de ejecutar la consulta o, si es posible, reescribir la consulta como una nueva consulta q', donde q' sea equivalente a q y tenga un menor costo de ejecución.

Definición 4.1. Dada una consulta q, una consulta q' será equivalente a q y viceversa si y solo si el conjunto de respuestas que se obtiene al aplicar q en cualquier instancia de una base de datos es igual al conjunto de respuesta que se obtiene al aplicar q' sobre la misma instancia de base de datos [1].

4.1. Optimización de Join Queries

Como ya fue indicado anteriormente, una consulta de tipo Join Query (JQ) se define de la siguiente forma $q(\bar{u}, g_1, g_2) : \exists \bar{z} P(\bar{x}_1, g_1) \wedge R(\bar{x}_2, g_2) \wedge T(g_1, g_2)$. Este tipo de consultas compara los atributos espaciales de dos tuplas con el fin de encontrar aquellos pares de tuplas donde se cumpla la condición de búsqueda, en este caso, $T(g_1, g_2)$. La consulta q devolverá todos las tuplas que combinan atributos de P y R cuando se cumpla la condición $T(g_1, g_2)$. La ejecución de este tipo de consultas que involucran condiciones de búsqueda sobre los atributos espaciales tiene un alto costo de procesamiento, ya que el sistema debe verificar si la relación topológica que existe entre los atributos espaciales de cada par de tuplas en P y R es la misma de la consulta.

El objetivo de la estrategia de optimización propuesta es deducir la relación topológica que existe entre cada par de tuplas de los predicados de una consulta dada, para así de esa forma poder determinar cuales de ellos cumplen con la condición de búsqueda de la consulta, sin necesidad de que el sistema realice el costoso proceso de verificación.

Para deducir la relación topológica entre los pares de tuplas se utilizarán restricciones de integridad, en particular las Dependencias Topológicas (TD). Estas restricciones imponen que ciertas tuplas de la base de datos deban cumplir con una relación topológica en particular. Por ejemplo, sea ic_1: $\forall x, y, g_1, g_2(P(x, x', x'', g_1) \wedge R(y, y', g_2) \rightarrow T'(g_1, g_2))$, está restricción impone que todos los pares de tuplas de los predicados P y R deben cumplir con la relación $T'(g_1, g_2))$. Por la tanto, dado ese conocimiento entregado por la restricción es posible determinar si las tuplas de la base de datos cumplirán con la condición de búsqueda de una consulta dada comparando la relación topológica de la consulta con la relación topológica indicada en la restricción. Además de utilizar una única restricción de integridad TD para llevar a cabo el proceso, también es posible utilizar dos restricciones TDs que actúen cada una sobre un grupo de pares de tuplas diferente.
Sea ic_2: $\forall x, y, g_1, g_2(P(x, x', x'', g_1) \wedge R(y, y', g_2) \wedge x = y' \rightarrow T''(g_1, g_2))$ y sea ic_3: $\forall x, y, g_1, g_2(P(x, x', x'', g_1) \wedge R(y, y', g_2) \wedge x \neq y' \rightarrow T'''(g_1, g_2))$, con estas dos restricciones es posible deducir por completo la relación topológica de cada par de tuplas de P y R, la cual será $T''(g_1, g_2)$ cuando se cumpla $x = y'$ y $T'''(g_1, g_2)$ cuando se cumpla $x \neq y$. Luego comparando ambas relaciones con la condición de búsqueda de la consulta será posible determinar qué tuplas devolver.

Dada la consulta q y dado un conjunto de restricciones de integridad IC del tipo TD (Definición 3.3), la estrategia de optimización propuesta buscará cumplir lo

siguiente:

i) deducir si la aplicación de la consulta q sobre alguna instancia de una base de datos espacial BD devuelve un conjunto de respuesta vacío, o
ii) generar, si es posible, una nueva consulta q' la cual sea equivalente a q pero con un costo de ejecución menor al costo de ejecución de la consulta original.

La idea principal a la hora de generar una nueva consulta equivalente es reemplazar la condición de búsqueda sobre atributos espaciales por una condición de búsqueda sobre atributos temáticos, esto debido a que los costos de procesar operadores espaciales, es decir, relaciones topológicas, es mucho más alto que comparar atributos temáticos.

Además de las restricciones de integridad de tipo TD que están explícitas en la base de datos es posible derivar más de ellas utilizando además las restricciones de tipo RD. Por ejemplo, dada la restricción ic:

$$\forall \bar{u}\bar{x}g_1(P(\bar{u},\bar{x},g_1) \rightarrow \exists \bar{v}\bar{y}g_2 R(\bar{v},\bar{y},g_2) \wedge \bar{x} = \bar{v} \wedge \mathcal{T}(g_1,g_2))$$

que indica que si existe una tupla en P, debe existir una tupla en R tal que $\bar{x} = \bar{v}$ y se cumpla $\mathcal{T}(g_1, g_2)$, y dado además que el atributo $\bar{v}$ es una super clave de R, es posible derivar una nueva TD definida de la siguiente forma

$$\forall \bar{u}\bar{x}\bar{v}\bar{y}g_1g_2(P(\bar{u},\bar{x},g_1) \wedge R(\bar{v},\bar{y},g_2) \wedge \bar{x} = \bar{v} \rightarrow \mathcal{T}(g_1,g_2)).$$

Por lo tanto la estrategia propuesta consta de dos etapas, (i) la derivación de nuevas restricciones (TDs) (ii) el proceso de optimización de la consulta utilizando todas las TDs disponibles.

4.1.1. Pre-procesamiento: derivando TDs

Utilizando las restricciones de integridad de tipo RD y dada ciertas condiciones en los atributos de los predicados utilizados es posible derivar nuevas restricciones TDs que nos servirán para llevar a cabo el proceso de optimización de las consultas. A continuación presentaremos los casos en los cuales es posible derivar nuevas restricciones.

Como ya vimos anteriormente el primer tipo de derivación es posible cuando se tiene una única restricción RD de la forma

$$\forall \bar{u}\bar{x}g_1(P(\bar{u},\bar{x},g_1) \rightarrow \exists \bar{v}\bar{y}g_2 R(\bar{v},\bar{y},g_2) \wedge \bar{x} = \bar{v} \wedge \mathcal{T}(g_1,g_2))$$

si se cumple además que el atributo $\bar{v}$ es una super clave de R entonces es posible deducir que cada vez que se cumpla la condición $\bar{x} = \bar{v}$ entonces se cumplirá además la relación $\mathcal{T}(g_1, g_2)$, esto es formalizado con el siguiente axioma.

Axioma 1. Sean $P(\bar{x}, g_1)$ y $R(\bar{y}, g_2)$ predicados donde g_1, g_2 son atributos espaciales, $\mathcal{T}$ una relación topológica, $\bar{u}, \bar{x}, \bar{v}$ y $\bar{y}$ secuencias de atributos temáticos. Entonces,

$$\frac{\forall \bar{u}\bar{x}g_1(P(\bar{u},\bar{x},g_1) \rightarrow \exists \bar{v}\bar{y}g_2 R(\bar{v},\bar{y},g_2) \wedge \bar{x} = \bar{v} \wedge \mathcal{T}(g_1,g_2)),\ \bar{v} \text{ es una super clave de } R}{\forall \bar{u}\bar{x}g_1\bar{v}\bar{y}g_2(P(\bar{u},\bar{x},g_1) \wedge R(\bar{v},\bar{y},g_2) \wedge \bar{x} = \bar{v} \rightarrow \mathcal{T}(g_1,g_2)).}$$

□

El segundo tipo de derivación extiende la noción de composición topológica de relaciones topológicas a la composición de RDs, lo cual es formalizado en el siguiente axioma

Axioma 2. Sean $P(\bar{x}, g_1)$, $Q(\bar{y}, g_2)$ y $R(\bar{z}, g_3)$ predicados con g_1, g_2, g_3 atributos espaciales, $\mathcal{T}_1, \mathcal{T}_2$ relaciones topológicas, $\bar{u}, \bar{x}$, $\bar{v}, \bar{y}$, $\bar{v'}, \bar{y'}, \bar{w}$ y $\bar{z}$ secuencias de atributos temáticos. Entonces,

$$\frac{\begin{array}{c}\forall \bar{u}\bar{x}g_1(P(\bar{u}\bar{x},g_1) \rightarrow \exists \bar{v}\bar{y}g_2 Q(\bar{y},g_2) \wedge \bar{x} = \bar{v} \wedge \mathcal{T}_1(g_1,g_2)),\ \bar{v} \text{ es super clave de } Q \\ \forall \bar{u'}\bar{y'}g_2(Q(\bar{v'},\bar{y'},g_2) \rightarrow \exists \bar{w}\bar{z}g_3 R(\bar{w},\bar{z},g_3) \wedge \bar{y'} = \bar{w} \wedge \mathcal{T}_2(g_2,g_3)),\ \bar{w} \text{ es super clave de } R\end{array}}{\begin{array}{c}\forall \bar{u}\bar{x}g_1\bar{v}\bar{y}g_2\bar{w}\bar{z}g_3(P(\bar{u},\bar{x},g_1) \wedge R(\bar{w}\bar{z},g_3) \wedge Q(\bar{v},\bar{y},g_2) \wedge \bar{x} = \bar{v} \wedge \bar{y'} = \bar{w} \wedge \mathcal{T}(g_1,g_2)), \\ \text{con } \mathcal{T} = \mathcal{T}_1 \otimes \mathcal{T}_2 \text{ y } \bar{y'} \subseteq \bar{v} \cup \bar{y}.\end{array}}$$

□

Debemos notar que el resultado de la composición produce una regla con mas de dos predicados, lo cual es una versión extendida de una relación topológica como la definida en [3], nos referiremos a esta restricción como *extended topological dependency* (ETD).

La derivación previa se puede generalizar para más de 2 restricciones de integridad de tipo RD, esto es formalizado con el siguiente axioma

Axioma 3. Sean $P_i(\bar{u}_i, \bar{x}_i, g_i)$ predicados con $n_i \geq 0$, g_i atributos espaciales, $\mathcal{T}_i$ relaciones topológicas, $\bar{u}_i$ y $\bar{x}_i$ secuencias de atributos temáticos. Entonces,

$$\frac{\begin{array}{c}\forall \bar{u}_1\bar{x}_1g_1(P_1(\bar{u}_1\bar{x}_1,g_1) \rightarrow \exists \bar{u}_2\bar{x}_2g_2P_2(\bar{u}_2\bar{x}_2,g_2) \wedge \bar{x}_1 = \bar{u}_2 \wedge \mathcal{T}_1(g_1,g_2)), \\ \bar{u}_2 \text{ es una super clave de } P_2 \\ \vdots \\ \forall \bar{u}_{n-1}\bar{x}_{n-1}g_{n-1}(P_{n-1}(\bar{u}_{n-1}\bar{x}_{n-1},g_{n-1}) \rightarrow \exists \bar{u}_n\bar{x}_ng_nP_n(\bar{u}_n\bar{x}_n,g_n) \wedge \bar{x}_{n-1} = \bar{u}_n \wedge \mathcal{T}_{n_1}(g_{n-1},g_n)), \\ \bar{u}_n \text{ es una super clave de } P_n\end{array}}{\begin{array}{c}\forall \bar{x}_ig_i(P_1(\bar{x}_1,g_1) \wedge .... \wedge P_n(\bar{x}_n,g_n) \wedge \bar{x}_1 = \bar{u}_2 \wedge .... \wedge \bar{x}_{n-1} = \bar{u}_n \wedge \mathcal{T}(g_1,g_n), \\ \mathcal{T} = \mathcal{T}_1 \otimes \mathcal{T}_2 \otimes ..... \otimes \mathcal{T}_{n_1}.\end{array}}$$

□

Hasta ahora para derivar nuevas restricciones (ETDs) hemos utilizado solo restricciones del tipo RD, las cuales imponen condiciones de iguales entre los atributos temáticos que conforman las claves foráneas de los predicados. El último tipo de derivación que utilizaremos será combinar una secuencia de RDs con una restricción de tipo $TD_{\neq}$, para así derivar una ETD con una condición de desigualdad entre los atributos temáticos que conforman las claves foráneas.

Axioma 4. Sean $P_i(\bar{x}_i, g_i)$ predicados con $n_i \geq 0$, g_i atributos espaciales, $\mathcal{T}_i$ realciones topológicas, y $\bar{x}_i$ secuencias de atributos temáticos. Entonces,

$$\frac{\begin{array}{c}\forall \bar{u}_1 \bar{x}_1 g_1 (P_1(\bar{u}_1 \bar{x}_1, g_1) \rightarrow \exists \bar{u}_2 \bar{x}_2 g_2 P_2(\bar{u}_2 \bar{x}_2, g_2) \wedge \bar{x}_1 = \bar{u}_2 \wedge \mathcal{T}_1(g_1, g_2)), \\ \bar{u}_2 \text{ una super clave de } P_2 \\ \vdots \\ \forall \bar{u}_{n-2} \bar{x}_{n-2} g_{n-2} (P_{n-2}(\bar{u}_{n-2} \bar{x}_{n-2}, g_{n-2}) \rightarrow \exists \bar{u}_n \bar{x}_n g_n P_{n-1}(\bar{u}_{n-1} \bar{x}_{n-1}, g_{n-1}) \wedge \bar{x}_{n-2} = \bar{u}_{n-1} \wedge \mathcal{T}_{n_2}(g_{n-2}, g_{n-1})), \\ \bar{u}_{n-1} \text{ una super clave de } P_{n-1} \\ \forall \bar{u}_{n-1} \bar{x}_{n-1} g_{n-1} \bar{u}_n \bar{x}_n g_n (P_{n-1}(\bar{u}_{n-1} \bar{x}_{n-1}, g_{n-1}) \wedge P_n(\bar{u}_n \bar{x}_n, g_n) \wedge \neg(\bar{x}_{n-1} = \bar{u}_n) \wedge \mathcal{T}'_{n_1}(g_{n-1}, g_n)), \\ \bar{u}_n \text{ una super clave de } P_n \end{array}}{\begin{array}{c}\forall \bar{x}_i g_i (P_1(\bar{x}_1, g_1) \wedge .... \wedge P_n(\bar{x}_n, g_n) \wedge \bar{x}_1 = \bar{u}_2 \wedge .... \wedge \neg(\bar{x}_{n-1} = \bar{u}_n) \wedge \mathcal{T}(g_1, g_n), \\ \mathcal{T} = \mathcal{T}_1 \otimes \mathcal{T}_2 \otimes ..... \otimes \mathcal{T}_{n-2} \otimes \mathcal{T}_{n_1}. \end{array}}$$

□

Las restricciones de tipo RD y la composición de estas restricciones pueden ser representadas en un grafo de dependencia y composición.

Definición 4.2. Sea $\Phi = \{\phi_1, \phi_2, \ldots \phi_n\}$ un conjunto de RDs de la siguiente forma $\forall \bar{u} \bar{x} g_1 (P(\bar{u}, \bar{x}, g_1) \rightarrow \exists \bar{v} \bar{y} g_2 Q(\bar{v}, \bar{y}, g_2) \wedge \bar{x} = \bar{v} \wedge \mathcal{T}(g_1, g_2))$, con $\bar{v}$ una super clave de Q. Un grafo de dependencia y composición es un grafo dirigido etiquetado en el cual hay un nodo por cada predicado $\phi_i \in \Phi$ y un arco desde el predicado $P(\bar{x}, g_1)$ al predicado $Q(\bar{y}, g_2)$ de ϕ_i tal que g_1 y g_2 están restringidas por una relación topológica $\mathcal{T}(g_1, g_2)$ en ϕ_i. Las etiquetas de los arcos son la relación topológica derivada, las condiciones sobre atributos temáticos y los predicados utilizados en la composición excluyendo los predicados representado por los nodos extremos al arco. □

El Ejemplo 4.1 ilustra un grafo de dependencia y composición.

Ejemplo 4.1. Consideremos las siguientes restricciones de integridad:

$$\mathsf{Place}(u, p, c, g_p) \rightarrow \exists v n c g_c (\mathsf{County}(v, n, s, g_c) \wedge c = v \wedge \mathsf{Within}(g_p, g_c)) \quad (4.1)$$

$$\mathsf{County}(v, n, s, g_c) \rightarrow \exists w m a g_s (\mathsf{State}(w, m, a, g_s) \wedge s = w \wedge \mathsf{Within}(g_c, g_s)) \quad (4.2)$$

La Figura 4.1 muestra el grafo de dependencia y composición.

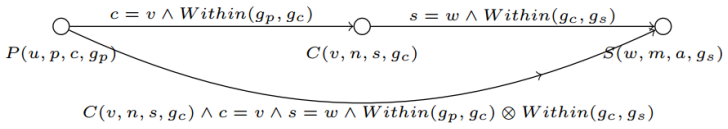

Figura 4.1: Grafo de dependencia y composición

□

El pre-procesamiento de las restricciones de integridad se puede llevar a cabo una vez y cada vez que se añadan o eliminen restricciones de integridad asociadas a la base de datos. Luego, cada vez que se realice una consulta espacial lo que se debe hacer es analizar las restricciones existentes para poder optimizar la consulta.

4.1.2. Optimización de la consulta

El proceso de optimización propiamente tal se realizará en el instante en que una consulta q es ingresada a la base de datos. Como mencionamos anteriormente mediante el análisis de las restricciones de integridad asociadas a la base de datos se busca determinar la relación topológica que existe entre cada par de tuplas de los predicados presentes en la consulta, con el fin de evitar la realización de la condición de búsqueda espacial de q debido a su alto costo de procesamiento.

Existen tres posibles resultados de aplicar este proceso de optimización:

i) lograr deducir la relación topológica de todos los pares de tuplas en cuestión, y de esta forma, eliminar por completo la condición de búsqueda espacial de la consulta.
ii) lograr deducir la relación topológica de solo algunos pares de tuplas en cuestión, de esta forma no es posible eliminar por completo la condición de búsqueda de la consulta, pero si es posible acotar el número de tuplas a consultar, logrando disminuir también el costo de procesamiento.
iii) no lograr deducir completamente ningún par de tuplas en cuestión, con lo cual la consulta ingresada no podrá ser optimizada.

Dada una consulta

$$q(\bar{x}) : \exists \bar{z}(P(\bar{y}_1, g_1) \wedge R(\bar{y}_2, g_2)[\wedge \psi_q] \wedge \mathcal{T}_q(g_1, g_2)),$$

donde $\bar{x}$ son variables libres en $\bar{y}_1 \cup \bar{y}_2 \cup \{g_1, g_2\} \setminus \bar{z}$, ψ_q es opcional y, si aparece, es una conjunción de igualdades o desigualdades de variables en $\bar{y}_1 \cup \bar{y}_2$.

Formalmente lo que se hace es **comparar** la relación topológica de la consulta con la relación topológica de alguna restricción de integridad (TD o ETD) que actúe sobre los mismos atributos espaciales que $\mathcal{T}_q$.

Como se puede apreciar, es importante definir cómo se debe realizar la **comparación** entre ambas relaciones topológicas. Es sabido que dos atributos espaciales cualesquiera deben estar relacionados a través de una relación topológica básica y, que además, está relación es única. Si en las consultas espaciales y restricciones solo se utilizara una restricción básica bastaría con comparar ambas restricciones mediante la igualdad o desigualdad, sin embargo, en la práctica los sistemas de almacenamiento y consulta de bases de datos espaciales utilizan otras relaciones topológicas definidas a partir de las relaciones básicas, tales como la relación Within, la que se define como la disyunción de las relaciones Equal, Inside y CoveredBy. Por lo tanto, es perfectamente posible que dos atributos espaciales, digamos g y g', estén relacionados entre si a través de $\mathsf{Within}(g, g')$ y $\mathsf{Equal}(g, g')$ al mismo tiempo. Además, en las consultas y restricciones es posible utilizar disyunciones de relaciones topológicas en vez de una relación en particular, como se muestra en la siguiente restricción:

$$\forall x, s, g_p, g_s (P(x, x', x'', g_p) \wedge S(s, s', g_s) \wedge x'' \neq s \rightarrow \mathsf{Disjoint}(g_p, g_s) \vee \mathsf{Touches}(g_p, g_s)$$

Por lo tanto una forma apropiada de comparación es la definida en Definición 3.1 y en Definición 3.2 las cuales utilizan la Inclusion y la Intersection para comparar las expresiones topológicas. Las siguientes afirmaciones son ciertas:

$$T \in \{T, T'\}$$
$$\{T\} \cap \{T, T'\} = \{T\}$$

Una vez definida la forma en que compararemos la relación topológica de la consulta original (T) con la relación de la restricción (T_ϕ), solo nos queda definir los posibles casos a los cuales nos podemos enfrentar al realizar dicha comparación.

Dada una consulta

$$q(\bar{x}) : \exists \bar{z} (P(\bar{y}_1, g_1) \wedge R(\bar{y}_2, g_2)[\wedge \psi_q] \wedge \mathcal{T}(g_1, g_2)),$$

y sea ic_1 una restricción de integridad de la siguiente forma:
$\exists P(\bar{x}_1, g_1) \wedge R(\bar{x}_2, g_2) \wedge \rho_\psi \wedge \psi_\phi \wedge \psi_q$

Se debe verificar cual de los siguientes tres opciones se cumple[1].

(1) $\{\mathcal{T}\} \cap \{\mathcal{T}_\phi\} = \emptyset$
(2) $\{\mathcal{T}_\phi\} \subseteq \{\mathcal{T}\}$
(3) Otro caso

Cada uno de estos casos llevará a una reescritura diferente de la consulta original. Además el proceso general tendrá una pequeña variación dependiendo de si la restricción utilizada abarca todos los pares de tuplas de los predicados o si solo abarca un grupo de ellos. A continuación se describirá el proceso tomando en cuenta restricciones que abarcan todos los pares de tuplas, más adelante se explicará el procedimiento para otros tipos de restricciones.

4.1.3. Caso 1: $\{\mathcal{T}\} \cap \{\mathcal{T}_\phi\} = \emptyset$

Primero se verifica si se cumple $\{\mathcal{T}\} \cap \{\mathcal{T}_\phi\} = \emptyset$, es decir, si la intersección del conjunto compuesto por la relación topológica de la consulta original con el conjunto de la relación topológica de la restricción de integridad es vacío. Si este fuera el caso, significaría que los pares de tuplas que abarca la restricción no cumplen con la relación topológica de la consulta y, por lo tanto, se pueden descartar del conjunto de respuesta.

Si la restricción de integridad utilizada abarca todos los pares de tuplas de la base de datos entonces se podrá deducir que el conjunto de respuesta de la consulta es vacío.

Ejemplo 4.2. Sea $q(x, y, g_1, g_2) : \exists x', x'', y' \ P(x, x', x'', g_1) \wedge R(y, y', g_2) \wedge \mathsf{Within}(g_1, g_2)$ y la restricción de integridad
ic_1: $\forall x, y, g_1, g_2 (P(x, x', x'', g_1) \wedge R(y, y', g_2) \rightarrow \mathsf{Disjoint}(g_1, g_2))$.

Según lo explicado anteriormente se verifica si $\{\mathsf{Within}\} \cap \{\mathsf{Disjoint}\} = \emptyset$, lo cual se cumple, ya que lo anterior es equivalente a $\{\mathsf{Equal}, \mathsf{CoveredBy}, \mathsf{Inside}\} \cap \{\mathsf{Disjoint}\} = \emptyset$. Por lo tanto, se concluye que en la base de datos no existe ninguna combinación de tuplas de P y R donde se cumpla $\mathsf{Within}(g_1, g_2)$, y, por lo tanto, se deduce que la consulta q tendrá un conjunto de respuesta vacío. □

[1]Con cierto abuso de notación, para un $\mathcal{T}$ definido como una disyunción de relaciones básicas (e.j. $\mathsf{Within} = \{\mathsf{Equal}, \mathsf{CoveredBy}, \mathsf{Inside}\}$), $\{\mathcal{T}\} = \mathcal{T}$

4.1.4. Caso 2: $\{\mathcal{T}_\phi\} \subseteq \{\mathcal{T}\}$

En caso de que la intersección de ambos conjuntos no sea vacía se debe verificar si se cumple $\{\mathcal{T}_\phi\} \subseteq \{\mathcal{T}\}$ es decir, si el conjunto compuesto por la relación de la restricción de integridad es un subconjunto del conjunto compuesto por la relación topológica de la consulta original. Si se cumple lo anterior, entonces se puede asegurar que los pares de tuplas que abarca la restricción cumplen con la condición de búsqueda y, por lo tanto, se puede eliminar dicha condición de la consulta y retornar los pares de tupla en cuestión.

Ejemplo 4.3. Sea la consulta $q(x, y, g_1, g_2) : \exists x' x'' y' P(x, x', x'', g_1) \wedge R(y, y', g_2) \wedge \mathsf{Within}(g_1, g_2)$ y sea ic_1: $\forall x\, y\, g_1 g_2 (P(x, x', x'', g_1) \wedge R(y, y', g_2) \rightarrow \mathsf{Equal}(g_1, g_2))$

Se comparan ambas relaciones. Primero se verifica si $\{\mathsf{Within}\} \cap \{\mathsf{Equal}\} = \emptyset$, es decir $\{\mathsf{Equal}, \mathsf{CoveredBy}, \mathsf{Inside}\} \cap \{\mathsf{Equal}\} = \mathsf{Equal}$ lo cual en este caso no se cumple. Entonces ahora se comprueba si $\{\mathsf{Equal}\} \subseteq \{\mathsf{Within}\}$, lo cual efectivamente se cumple y, por lo tanto, todas las combinaciones de tuplas en P y R cumplirán con la condición de búsqueda impuesta en la consulta. De esta forma q se puede reescribir como $q'(x, y, g_1, g_2) : \exists x', x'', y'\ P(x, x', x'', g_1) \wedge R(y, y', g_2)$, obteniendo el mismo conjunto de respuesta que se hubiese obtenido aplicando q, sin necesidad de verificar la costosa condición de búsqueda espacial. □

4.1.5. Caso 3

Por último si ninguno de los dos casos anteriores se cumplen entonces eso quiere decir que se cumplirá $\{\mathcal{T}\} \subseteq \{\mathcal{T}_\phi\}$ o simplemente $\{\mathcal{T}\} \cap \{\mathcal{T}_\phi\} \neq \emptyset$. En este caso se tendrá que dentro de esos pares de tuplas que abarca la restricción es posible que existan algunos que cumplan con la condición de búsqueda de la consulta, pero no se puede asegurar su existencia, y en caso de que hayan, no se puede determinar cuales de ellos lo hacen, por lo tanto para esos pares de tuplas se debe ejecutar la condición de búsqueda espacial de la consulta original.

Ejemplo 4.4. Sea la consulta $q(x, y, g_1, g_2) : \exists x' x'' y' P(x, x', x'', g_1) \wedge R(y, y', g_2) \wedge \mathsf{Equal}(g_1, g_2)$ y sea ic_2: $\forall x\, y\, g_1 g_2 (P(x, x', x'', g_1) \wedge R(y, y', g_2) \rightarrow \mathsf{Within}(g_1, g_2))$

Se comparan ambas relaciones. Primero se verifica si $\{\mathsf{Equal}\} \cap \{\mathsf{Within}\} = \emptyset$, lo cual en este caso no se cumple, ya que lo anterior es equivalente a $\{\mathsf{Equal}\} \cap \{\mathsf{Equal}, \mathsf{CoveredBy}, \mathsf{Inside}\} = \mathsf{Equal}$. Entonces ahora se comprueba si $\{\mathsf{Within}\} \subseteq \{\mathsf{Equal}\}$, lo cual tampoco se cumple, por lo tanto, en este caso no se puede optimizar la consulta ya que se sabe que todos los pares de tuplas de P y R cumplirán

con $\{\mathsf{Equal}, \mathsf{CoveredBy}, \mathsf{Inside}\}$, pero no se puede determinar los pares que tienen la relación topológica Equal. De esta forma la reescritura de la consulta q queda de la misma forma $q(x, y, g_1, g_2) : \exists x' x'' y' P(x, x', x'', g_1) \wedge R(y, y', g_2) \wedge \mathsf{Equal}(g_1, g_2)$. □

El procedimiento a seguir en la optimización se ve reflejado en La Tabla 4.1, donde se muestra la reescritura que se debe generar para cada uno de los casos explicados anteriormente.

Lo explicado anteriormente es para el caso cuando ocupamos una restricción de integridad que abarca todos los pares de tuplas de los predicados en cuestión, lo cual sería lo ideal, sin embargo, hay oportunidades donde ese tipo de restricciones no existen en la base de datos. Para esos casos es posible llevar a cabo el mismo procedimiento utilizando dos restricciones de integridad si es que ambas restricciones en conjunto abarcan todos los pares de tuplas de los predicados en cuestión de la base de datos.

El procedimiento a seguir al utilizar dos restricciones de integridad para la optimización de la consulta es realizar dos veces el proceso explicado anteriormente, uno para cada restricción de integridad, y luego unir los resultados obtenidos.

Ejemplo 4.5. Sea la consulta $q(x, y, g_1, g_2) : \exists x' x'' y' P(x, x', x'', g_1) \wedge R(y, y', g_2) \wedge \mathcal{T}_q(g_1, g_2)$

Sean

$$ic_1 : \forall x\, y\, g_1 g_2 (P(x, x', x'', g_1) \wedge R(y, y', g_2) \wedge x = y' \rightarrow \mathcal{T}_\phi(g_1, g_2))$$

$$ic_2 : \forall x\, y\, g_1 g_2 (P(x, x', x'', g_1) \wedge R(y, y', g_2) \wedge x \neq y' \rightarrow \mathcal{T}_{\phi'}(g_1, g_2))$$

Las restricciones indican que para las tuplas donde se cumpla $x = y'$ la relación topológica será $\mathcal{T}_\phi(g_1, g_2)$ y para las otras tuplas restantes donde se cumple $x \neq y'$ la relación topológica será $\mathcal{T}_{\phi'}(g_1, g_2)$.

Como se mencionó anteriormente se debe llevar a cabo el procedimiento para cada una de las restricciones y luego unir los resultados.

Supongamos que las siguientes afirmaciones son ciertas:

i $\{\mathcal{T}_q(g_1, g_2)\} \cap \{\mathcal{T}_\phi(g_1, g_2)\} = \emptyset$
ii $\{\mathcal{T}_{\phi'}(g_1, g_2)\} \subseteq \{\mathcal{T}_q(g_1, g_2)\}$

Para ic_1, se cumple el Caso 1, es decir ningún par de tupla donde se cumpla $x = y'$ cumplirá con la condición de búsqueda de la consulta y al analizar la Tabla 4.1 se deduce que la reescritura debe ser el conjunto vacío.

Para ic_2, se cumple el Caso 2, es decir todos los pares de tuplas donde se cumpla $x \neq y'$ cumplirán con la condición de búsqueda de la consulta, la Tabla 4.1 indica que la reescritura en este caso debe ser $q'(x, y, g_1, g_2) : \exists x', x'', y'\ P(x, x', x'', g_1) \wedge R(y, y', g_2) \wedge x \neq y'$.

Finalmente para obtener la optimización de la consulta se lleva a cabo la unión de ambos resultados obteniendo en este caso que la consulta q se puede optimizar reescribiendola como $q'(x, y, g_1, g_2) : \exists x', x'', y'\ P(x, x', x'', g_1) \wedge R(y, y', g_2) \wedge x \neq y'$.

□

Case	Condition	**Output**
(i)	$\{\mathcal{T}_\phi\} \cap \{\mathcal{T}\} = \emptyset$	$\emptyset$
(ii)	$\{\mathcal{T}_\phi\} \subseteq \{\mathcal{T}\}$	$\exists P(x, g_1) \wedge R(y, y_2) \wedge \rho_\psi \wedge \psi_\phi \wedge \psi_q$
(iii)	Otherwise	$\exists P(x, g_1) \wedge R(y, y_2) \wedge \rho_\psi \wedge \psi_q \wedge \psi_\phi \wedge \mathcal{T}_q$

Cuadro 4.1: TD-based optimization

Al utilizar dos restricciones para la optimización se pueden tener varios resultados, por ejemplo si se da que ambas comparaciones caen en el Caso 1 se tendrá que el conjunto de respuesta de la consulta es vacío, por otro lado si ambas comparaciones caen en el Caso 3 se tendrá que la condición de búsqueda de la consulta deberá ser evaluada en cada par de tuplas, lo cual es equivalente a tener la misma consulta original y, por lo tanto, no se logrará una optimización.

Algunas cosas importantes a tener en cuenta son:

1. Cuando se utilizan dos restricciones de integridad para la optimización de la consulta siempre se debe cumplir que ambas restricciones estén conformadas por los mismos predicados y sean aplicadas sobre los mismos atributos tanto en las condiciones temáticas como en las condiciones espaciales y, además, se debe cumplir que en una restricción se tenga la condición temática ψ_ϕ y en la otra se tenga su negación $\neg\psi_\phi$, esto para estar seguros de abarcar todos los pares de tuplas que involucra la consulta.
2. Si en un caso solo se tuviera una de las dos restricciones mencionadas anteriormente, es decir, si no se tiene la restricción en la cual la condición

temática es $\neg\psi_\phi$, entonces se debe realizar el proceso para la restricción que si se tiene y unir el resultado con la consulta

$$\bar{\exists}P(x, g_1) \wedge R(y, y_2) \wedge \rho_\psi \wedge \psi_q \wedge \neg\psi_\phi \wedge \mathcal{T}_q$$

ya que no se tiene conocimiento de la relación topológica de las tuplas donde se cumple la condición $\neg\psi_\phi$ y por lo tanto se debe llevar a cabo la consulta para ese conjunto de tuplas.

3. Antes de realizar la unión de ambas consultas se deben verificar posibles contradicciones que puedan ocurrir entre las condiciones temáticas de la consulta original (ψ_q) y las condiciones temáticas de las restricciones (ψ_ϕ y $\neg\psi_\phi$), en caso de encontrar contradicciones se determina que la reescritura de dicho Caso será $\emptyset$. Esto se muestra en el Ejemplo 4.6.

Ejemplo 4.6. Sea la consulta $q(x, y, g_1, g_2) : \exists x' x'' y' P(x, x', x'', g_1) \wedge R(y, y', g_2) \wedge x \neq y' \wedge \mathsf{Within}(g_1, g_2)$

Sean

$ic_1 : \forall x\, y\, g_1 g_2 (P(x, x', x'', g_1) \wedge R(y, y', g_2) \wedge x = y' \rightarrow \mathsf{Equal}(g_1, g_2))$

$ic_2 : \forall x\, y\, g_1 g_2 (P(x, x', x'', g_1) \wedge R(y, y', g_2) \wedge x \neq y' \rightarrow \mathsf{Disjoint}(g_1, g_2))$

Tomando cada restricción por separado se tiene que para ic_1 se cae en el Caso 2 ya que $\mathsf{Equal}(g_1, g_2) \subseteq \mathsf{Within}(g_1, g_2)$ es cierto. Por otro lado se tiene que para ic_2 se cae en el Caso 1 ya que se cumple $\mathsf{Within}(g_1, g_2) \cap \mathsf{Disjoint}(g_1, g_2) = \emptyset$. Por lo tanto la consulta reescrita debiera ser la unión de $q_1(x, y, g_1, g_2) : \exists x', x'', y'\ P(x, x', x'', g_1) \wedge R(y, y', g_2) \wedge x = y'$ y $\emptyset$. Sin embargo dado que en la consulta original se tiene la condición $x \neq y'$, se produce una contradicción con q_1, por lo tanto las tuplas que abarca q_1 se deben descartar, es decir q_1 pasa a ser igual a $\emptyset$, lo cual lleva a que la consulta original tiene un conjunto de respuesta vacío.

De esta forma hemos definido completamente el procedimiento a seguir cuando se quiere optimizar una Join Query. Como hemos visto, lo más importante es contar con las restricciones de integridad adecuadas y, dependiendo de los resultados que se obtengan al comparar la condición de búsqueda de la consulta con las relaciones topológicas generadas, se podrá lograr una optimización.

Podemos resumir el proceso de optimización en los siguientes pasos:

(i) Preprocesar TDs y RDs para derivar nuevas TDs y/o ETDs.
(ii) Buscar la restricción o par de restricciones que tengan relaciones topológicas sobre geometrías en los predicados P y R de la consulta q.
(iii) Realizar por separado la comparación de la relación de la consulta con la relación de cada una de las dos restricciones de integridad tomadas.
(iv) Detectar posibles contradicciones entre condiciones temáticas de las consultas generadas. Si existe alguna, el resultado de la consulta es vacío.
(v) Basados en la Tabla 4.1 unir los resultados obtenidos

4.2. Optimización de Range Queries

Como se ha indicado anteriormente, una consulta de tipo Range Query (RQ) es de la forma $q(\bar{x}, g) : \exists \bar{u} P(\bar{x}, \bar{u}, g) \wedge T(w, g)$, donde w es un parámetro espacial arbitrario y T una relación topológica.

En un sistema de base de datos donde las tablas no cuentan con un índice espacial asociado a sus atributos (más adelante veremos qué sucede cuando sí existen estos índices), al procesar este tipo de consultas se analiza cada tupla del predicado P en la base de datos y se calcula la relación topológica que existe entre su atributo espacial y el parámetro w. Como ya hemos mencionado anteriormente, la ejecución de este tipo de consultas que involucran condiciones de búsqueda sobre los atributos espaciales tiene un alto costo de procesamiento debido al cálculo de la relación topológica. Debido a lo anterior, dada la consulta q, y dado un conjunto de restricciones de integridad IC del tipo RD y TD, la estrategia de optimización propuesta utilizará la técnica Chase para derivar nuevos conocimientos desde las restricciones de integridad para luego tratar de generar una nueva consulta q' la cual sea equivalente a q pero con un menor tiempo de ejecución.

4.2.1. Optimización de la consulta

El proceso de optimización de las consultas de rango se basa en la generación de una *consulta intermedia*, la cual denominaremos q_{int}. Esta consulta intermedia tiene como objetivo reducir, si es posible, el número de tuplas que serán consultadas en la consulta original, es decir, dada una consulta de rango $q(\bar{x}, g) : \exists \bar{u} P(\bar{x}, \bar{u}, g) \wedge T(w, g)$, q_{int} acotará el número de tuplas correspondientes al predicado P con el fin de realizar menos veces el cálculo de la relación $T(w, g)$. Luego de haber acotado el número de tuplas a consultar, se ejecutará la consulta original para obtener los resultados finales.

La nueva consulta q_{int} se generará a partir de la consulta original y utilizando una restricción de integridad de tipo RD que tenga al predicado P como antecedente, es decir, una restricción de la forma $ic : \forall\bar{x}\bar{u}g_1(P(\bar{x},\bar{u},g_1) \rightarrow \exists\bar{y}g_2R(\bar{y},\bar{v},g_2) \wedge \bar{u} = \bar{y} \wedge T'(g_1,g_2))$. Esta restricción nos asegura que cada tupla del predicado P estará relacionada con al menos una tupla del predicado R de la base de datos.

El procedimiento a seguir es tomar la consulta q y aplicarle la técnica Chase utilizando la restricción de integridad ic, generando entonces la consulta intermedia $q_{int}(\bar{x},g) : \exists\bar{u},\bar{y},\bar{v},g_2P(\bar{x},\bar{u},g) \wedge T(w,g) \wedge R(\bar{y},\bar{v},g_2) \wedge \bar{u} = \bar{y} \wedge T'(g,g_2)$. En este momento hemos agregado a la consulta un nuevo predicado y una nueva condición de búsqueda que relaciona atributos espaciales de ambos predicados, además tenemos una condición de igualdad entre atributos temáticos. La consulta generada q_{int} puede aún sufrir una última transformación, la cual se basa en la composición de relaciones topológicas generando una relación topológica entre el parámetro w y el atributo espacial g_2 del predicado R, es decir, $T(w,g) \otimes T'(g,g_2) = T_c(w,g_2)$. De esta forma la consulta intermedia q_{int} toma la forma

$$q_{int}(\bar{x},g) : \exists\bar{u},\bar{y},\bar{v},g_2P(\bar{x},\bar{u},g) \wedge R(\bar{y},\bar{v},g_2) \wedge \bar{u} = \bar{y} \wedge T_c(w,g_2).$$

Esta consulta analiza las tuplas del predicado R y calcula la relación topológica que existe entre el parametro w y el atributo g_2 para luego compararla con la condición de búsqueda T_c. Finalmente se obtendrán las tuplas del predicado P cuyo atributo $\bar{u}$ sea igual al atributo $\bar{y}$ de las tuplas en R obtenidas previamente. Luego de ejecutar la consulta q_{int} se ejecuta la consulta original q sobre las tuplas devueltas por q_{int} obteniendo los resultados finales. Se puede definir una consulta global q_{opt} que abarque todo el proceso de la siguiente forma

$$q_{opt}(\bar{x},g) : (\exists\bar{u},\bar{y},\bar{v},g_2P(\bar{x},\bar{u},g) \wedge R(\bar{y},\bar{v},g_2) \wedge \bar{u} = \bar{y} \wedge T_c(w,g_2)) \wedge T(w,g)$$

Recordemos que el objetivo de esta optimización es disminuir el número de cálculos que debe hacer el sistema al chequear la condición de búsqueda espacial, ya que realizar eso es muy costoso. Entonces analicemos los cálculos que se hacen en q y en q_{opt}.

Sea N el número de tuplas del predicado P, sea M el número de tuplas del predicado R y sea $|q_{int}|$ el número de tuplas devueltas por la consulta intermedia. Al ejecutar la consulta q se realizan N cálculos. Por otro lado, en la consulta q_{opt}

tenemos dos condiciones de búsqueda espaciales, al chequear la condición T_c se realizarán M cálculos y luego al chequear la condición T se realizarán $|q_{int}|$ cálculos. Por lo tanto, lo mínimo que se puede esperar para que la consulta optimizada trabaje mejor que la original es que se cumpla $|q_{int}| < N$, es decir, que realmente se logre reducir el número de tuplas a analizar por la consulta q. En general, también se espera que se cumpla $(M + |q_{int}|) < N$, es decir, el número de tuplas total analizadas en q_{opt} sea menor al número de tuplas analizadas en q.

Habrá más posibilidad de conseguir la desigualdad anterior si se cumplen las condiciones siguientes en los datos:

a) $M << N$, es decir, que el número de tuplas en el predicado R sea bastante menor que el número de tuplas en el predicado P. Esta condición se puede chequear fácilmente antes de realizar el proceso de optimización.
b) La variable $|q_{int}|$ representa el número de tuplas devuelto por la consulta intermedia, lo cual dependerá del tamaño de las características del parámetro w y de la relación topológica T_c, donde T_c puede ser una relación única o una disyunción de relaciones. Más adelante mostraremos las características que deben tener w y T_c para llevar a cabo el proceso de optimización satisfactoriamente.

En general no se puede asegurar en un 100 % que la optimización ocurrirá para cualquier instancia de la base de datos, todo dependerá de la distribución de los datos en el espacio, y de cómo están relacionados entre sí y con el parámetro w. A continuación mostraremos ejemplos de cuándo esta optimización logra su objetivo y cuando no.

Ejemplo 4.7. *(Cont. Example 3.1)* Consideremos una base de datos espacial que almacene información sobre los límites administrativos de un país usando el mismo esquema definido en el Ejemplo 3.1. La Figura 4.2 muestra una instancia de este esquema. En ella, las líneas más gruesas representan los límites de los estados y las líneas discontinuas representan los límites de los condados.

Considere un conjunto de restricciones de integridad IC asociada a esta base de datos el cual contiene las siguientes restricciones:

$ic_1 : \overline{\forall}(County(u, v, w, g) \rightarrow \exists u'v'w'g' State(u', v', w', g') \wedge w = u' \wedge Within(g, g'))$
$ic_2 : \overline{\forall}\,(\mathrm{State}(u, v, w, g) \wedge \mathrm{State}(u, v', w', g') \rightarrow \mathsf{Equal}(g, g'))$
$ic_3 : \overline{\forall}\,(\mathrm{County}(u, v, w, g) \wedge \mathrm{County}(u, v', w', g') \rightarrow \mathsf{Equal}(g, g'))$

Sea q una consulta de rango de la forma
$q(Id_c, G_c) \ : \ \exists Name_c, Id_s \ County(Id_c, Name_c, Id_s, G_c) \wedge \mathsf{Overlaps}(w, G_c)$,

State	Id_s	*Name*	*Area*	G_s
	s_1	$name_1$	1400	s_1
	s_2	$name_2$	1100	s_2
	s_3	$name_3$	1000	s_3
	s_4	$name_4$	1400	s_4
	s_5	$name_5$	1300	s_5
	s_6	$name_6$	1600	s_6

County	Id_c	*Name*	Id_s	G_c
	c_1	$name_7$	s_1	g_1
	c_2	$name_8$	s_1	g_2
	c_3	$name_9$	s_1	g_3
	c_4	$name_{10}$	s_1	g_4
	c_5	$name_{11}$	s_1	g_5
	c_6	$name_{12}$	s_1	g_6
	c_7	$name_{13}$	s_2	g_7
	...	.....	...	...
	...	.....	...	...
	c_{33}	$name_{39}$	s_6	g_{33}

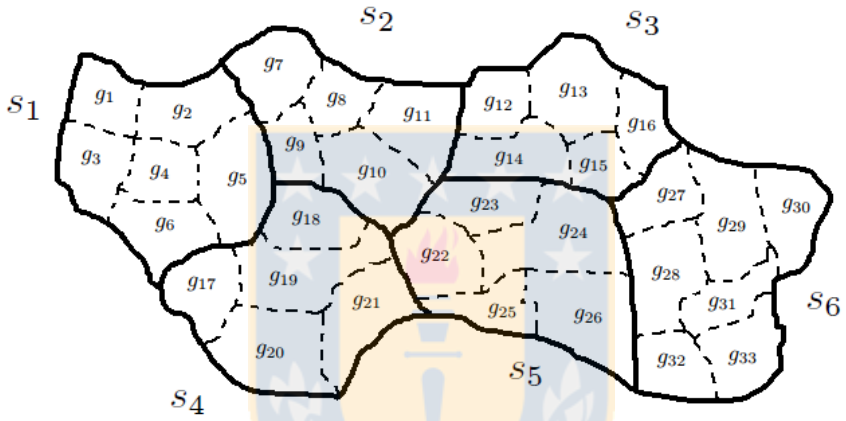

Figura 4.2: Instancia de Base de Datos

donde w es un parámetro que representa un rectángulo en el plano y cuya ubicación se muestra en la Figura 4.3.

El objetivo de la consulta q es recuperar todos aquellos condados en County que se superpongan con el parámetro w ingresado. Gráficamente podemos darnos cuenta que la consulta ejecutada en esta instancia debiera devolver los condados con valores de Id_c iguales a $c_{12}, c_{13}, c_{14}, c_{15}, c_{16}, c_{23}, c_{24}$ y c_{27}.

Al ejecutar la consulta el sistema debe calcular la relación topológica que existe entre cada condado de la base de datos y el parámetro w, es decir, debe realizar 33 comparaciones espaciales. Usando la estrategia de optimización, se buscan restricciones de tipo RD cuyo antecedente sea el predicado County. En este caso, la restricción $ic_1 : \overline{\forall}(County(u, v, w, g) \rightarrow \exists u'v'w'g' State(u', v', w', g') \wedge w = u' \wedge Within(g, g'))$ cumple con esta característica, ic_1 establece una relación en-

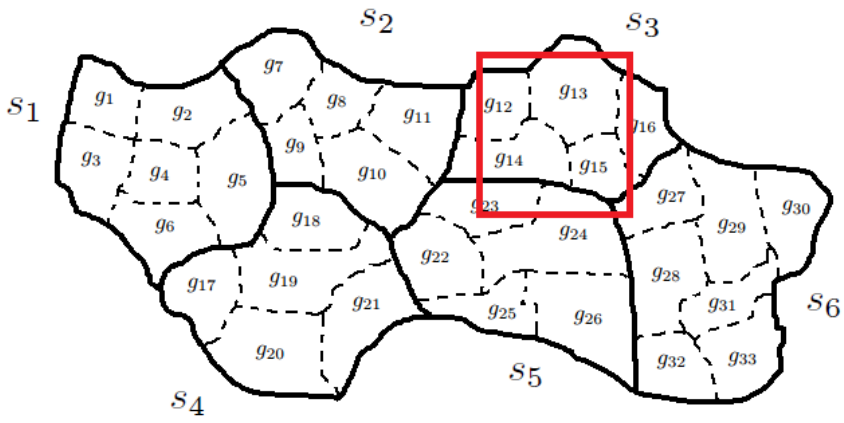

Figura 4.3: Consulta de Rango

tre el predicado County y el predicado State. Luego se comprueba si se cumple con el requisito de que en la base de datos existan más tuplas en la tabla County que en la tabla State, lo cual efectivamente se cumple. Por lo tanto, siguiendo el proceso descrito anteriormente se reescribirá la consulta, generando una consulta intermedia definida de la forma:
$q_{int}(Id_c, G_c) : \exists Name_c, Id_s, Name_s, Area, G_s\, County(Id_c, Name_c, Id_s, G_c) \wedge State(Id_s, Name_s, Area, G_s) \wedge (\mathsf{Overlaps}(w, G_s) \vee \mathsf{CoveredBy}(w, G_s))$.

Nótese que se realizó la composición entre la relación Overlaps y Within, dando como resultado $(\mathsf{Overlaps}(w, G_s) \vee \mathsf{CoveredBy}(w, G_s))$. Esta consulta intermedia buscará los estados en State que se sobrepongan o cubran al parámetro w. Se realizarán 6 comparaciones dando como resultado los estados con Id_s igual a s_3, s_5 y s_6. Finalmente se realizará la consulta original sobre los condados en County que se encuentren dentro de alguno de esos tres estados, llevando a cabo en ese proceso 17 comparaciones espaciales y entregando los resultados finales esperados $c_{12}, c_{13}, c_{14}, c_{15}, c_{16}, c_{23}, c_{24}$ y c_{27}.

La consulta optimizada queda entonces de la forma:
$q_{opt}(Id_c, G_c) : (\exists Id_c\, County(Id_c, Name_c, Id_s, G_c) \wedge State(Id_s, Name_s, Area, G_s) \wedge (\mathsf{Overlaps}(w, G_s) \vee \mathsf{CoveredBy}(w, G_s)) \wedge \mathsf{Overlaps}(w, G_c)$.

Mientras que en la consulta original q fue necesario analizar todos los condados en County, en la consulta optimizada q_{opt} se logró reducir a casi la mitad el número de condados a analizar, lo cual implica una rebaja en el costo de ejecución de la

consulta. □

Acabamos de ver un ejemplo donde la aplicación de la estrategia de optimización logra reducir el tiempo de ejecución de una consulta. Sin embargo, como mencionamos anteriormente existen casos especiales donde no siempre se podrá lograr una mejora en el tiempo, debido principalmente a que al aplicar al consulta intermedia no se logra reducir lo suficiente el número de tuplas a analizar en la consulta final.

Ejemplo 4.8. *(Cont. Example 4.7)* Tomemos la misma instancia de base de datos mostrado en el ejemplo anterior y el mismo conjunto de restricciones de integridad IC asociado a ella. Sea q una consulta de rango de la forma: $q(Id_c, G_c) : \exists Name_c, Id_s County(Id_c, Name_c, Id_s, G_c) \wedge \mathsf{Overlaps}(w, G_c)$, donde w es un parámetro que representa una rectángulo en el plano y cuya ubicación se muestra en la Figura 4.4.

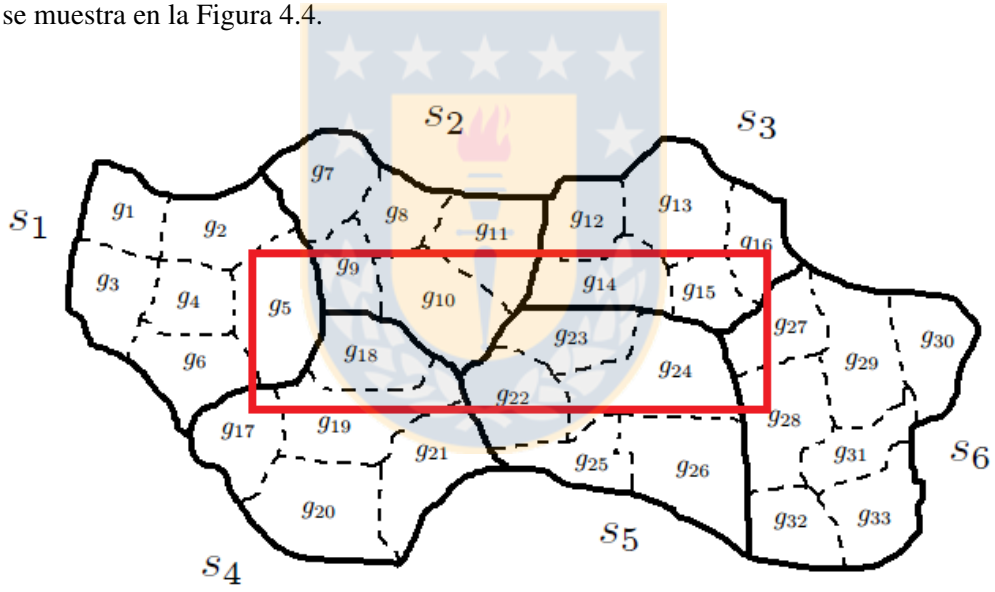

Figura 4.4: Consulta de Rango

La consulta q presentada en esta ocasión es la misma que el ejemplo anterior, con la diferencia de que en este caso el parámetro w se sobrepone con cada uno de los estados de la base de datos. Entonces al aplicar el proceso de optimización a esta consulta, se generará una consulta intermedia q_{int} que resultan ser todos los estados de la base de datos. Esto conlleva a que en la parte final del proceso de optimización se realice la consulta original sobre todos los condados de la base de datos, en otras palabras, en este caso resulta que la consulta optimizada es exactamente la

misma que la original más la consulta intermedia, por lo cual claramente no se logrará reducir el tiempo de ejecución. □

Como se dijo anteriormente, el caso del ejemplo anterior sucederá si la consulta intermedia no logra reducir el número de tuplas a analizar. Sin embargo hay que destacar que estos son casos muy puntuales y depende mucho de la distribución de los datos en el espacio y del parámetro w. Otro caso donde no se logra reducir el número de tuplas a analizar se presenta en el ejemplo siguiente.

Ejemplo 4.9. Consideremos el mismo esquema de base de datos usado en los ejemplos anteriores pero con una instancia diferente mostrada en la Figura 4.5. En esta instancia en particular se tiene que todos los condados de la base de datos están asociadas a un solo estado.

State	Id_s	*Name*	*Area*	G_s
	s_1	$name_1$	1400	s_1
	s_2	$name_2$	1100	s_2
	s_3	$name_3$	1000	s_3
	s_4	$name_4$	1400	s_4
	s_5	$name_5$	1300	s_5
	s_6	$name_6$	1600	s_6

County	Id_c	*Name*	Id_s	G_c
	c_1	$name_7$	s_5	g_1
	c_2	$name_8$	s_5	g_2
	c_3	$name_9$	s_5	g_3
	c_4	$name_{10}$	s_5	g_4
	...	.....	...	...
	...	.....	...	...
	c_{14}	$name_{20}$	s_5	g_{14}

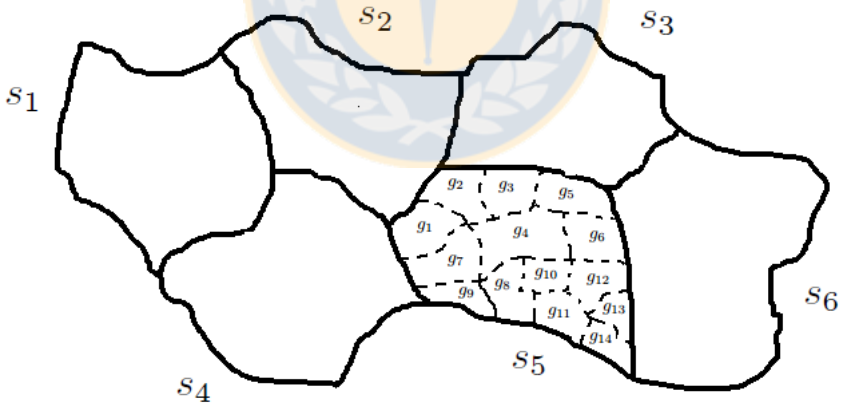

Figura 4.5: Instancia de Base de Datos

Considere además un conjunto de restricciones de integridad IC asociado a esta base de datos, el cual contiene las siguientes restricciones:
$ic_1 : \overline{\forall}(County(u, v, w, g) \rightarrow \exists u'v'w'g' State(u', v', w', g') \wedge w = u' \wedge Within(g, g'))$
$ic_2 : \overline{\forall}\,(\text{State}(u, v, w, g) \wedge \text{State}(u, v', w', g') \rightarrow \mathsf{Equal}(g, g'))$

$ic_3 : \overline{\forall}\,(\text{County}(u, v, w, g) \wedge \text{County}(u, v', w', g') \rightarrow \mathsf{Equal}(g, g'))$

Sea la consulta de rango $q(Id_c, G_c) : \exists Name_c, Id_s\ County(Id_c, Name_c, Id_s, G_c) \wedge \mathsf{Overlaps}(w, G_c)$, donde w es un parámetro que representa una rectángulo en el plano y cuya ubicación se muestra en la Figura 4.6.

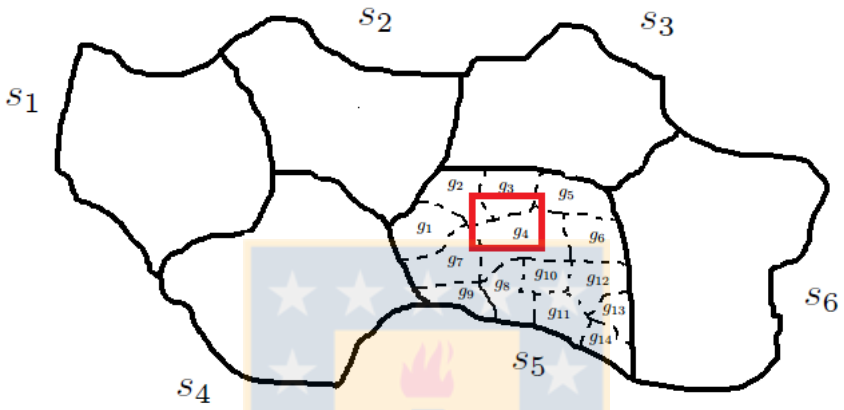

Figura 4.6: Consulta de Rango

En este caso se tiene que al aplicar el proceso de optimización, la consulta intermedia buscará primero los estados que se sobrepongan o cubran al parámetro w, lo cual en este caso corresponde solo al estado s_5, para luego analizar los condados asociados a él. Sin embargo, dado que dicho estado es el que almacena todos los condados de la base de datos, el número de tuplas de condados a analizar no se reduce en comparación a la consulta original y, por lo tanto, el tiempo de ejecución de la consulta reescrita será incluso mayor debido a la consulta intermedia. □

En los ejemplos anteriores queda demostrado que la distribución de los datos en el espacio geométrico y las características del parámetro w influyen directamente en el desempeño de la estrategia de optimización propuesta.

Hasta el momento hemos considerado bases de datos sin ningún tipo de índice en sus tablas y hemos establecido que nuestra estrategia de optimización servirá dada algunas condiciones en los datos y en las restricciones. El hecho de no tener índices en las tablas influye en la forma en la que el sistema ejecuta una consulta. Por ejemplo, para una consulta de rango como las mostradas anteriormente, al ejecutarlas sin la optimización, el sistema recorre las correspondientes tuplas de las tablas una por una en busca de las respuestas; sin embargo, en la presencia de índices

espaciales eso no siempre ocurre y, por lo tanto, a la hora de querer optimizar la consulta es posible no obtener los resultados esperados.

4.2.2. Influencia de índices espaciales

Al ejecutar una consulta espacial en una base de datos sin índices espaciales en sus tablas, el sistema debe analizar en forma secuencial cada objeto geométrico en busca de aquellos que cumplan con la condición de búsqueda de la consulta. Los índices espaciales permiten acotar el número de objetos geométricos a analizar, descartando la mayoría de los objetos que no cumplirán con la condición de búsqueda.

Consideremos el caso de un índice de tipo R-tree [9]. La forma en la que se logra esta reducción de objetos a analizar es utilizando la *minimum bounding box* o simplemente *bounding box* de cada geometría. La *bounding box* de una geometría cualquiera se define como la *caja* con la menor medida (área, volumen, etc.) que cubre todos los puntos de la geometría. Para el caso de objetos en dos dimensiones, esta caja corresponde a un rectángulo y, en general, se utiliza el área como valor de medida. La Figura 4.7 muestra la bounding box de una geometría cualquiera, la cual es representada con el rectángulo de color rojo.

Figura 4.7: Bounding Box

El índice espacial solo indexa la bounding box de cada una de las geometrías almacenándolas en un R-tree. Al ejecutar una consulta espacial, el sistema primero consultará las bounding boxes almacenadas en el índice, las cuales pueden ser rápidamente localizadas y comparadas. De esta forma se logra reducir el número de geometrías que efectivamente se compararán en la consulta. Finalmente se verifica la consulta sobre las geometrías devueltas por el índice para obtener los resultados.

Ejemplo 4.10. Tomemos la misma instancia de base de datos utilizada en los ejemplos anteriores centrando nuestra atención en los Estados. Supongamos ahora que

queremos obtener todos los estados en State que se sobreponen con cierto parámetro espacial w, dado por la consulta:
$q(Id_s, G_s) : \exists Name_s, Area\ State(Id_s, Name_s, Area, G_s) \wedge \mathsf{Overlaps}(w, G_s)$.
En la Figura 4.8 se muestra la instancia de State y el parámetro w en color celeste.

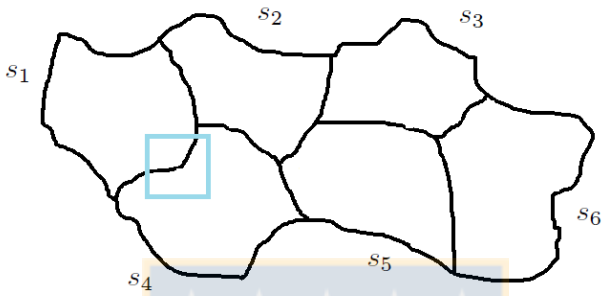

Figura 4.8: Consulta de Rango sobre Estados

Gráficamente podemos darnos cuenta que los estados con Id_s iguales a s_1 y s_4 cumplen con la condición de búsqueda de la consulta q. Veamos ahora como resuelve el sistema esta consulta en presencia de un índice espacial en la tabla $State$.

La Figura 4.9 muestra las bounding boxes que representan a las geometrías de los estados, cada una con un color diferente.

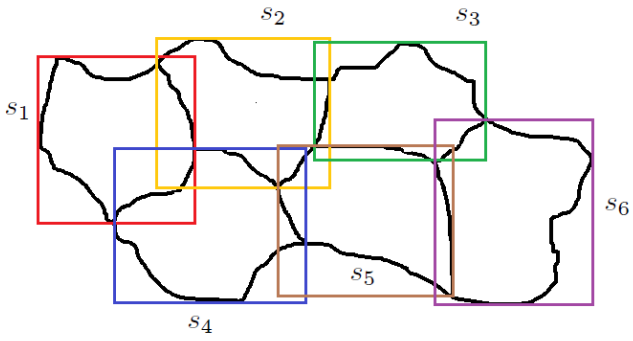

Figura 4.9: Bounding Boxes de los Estados

Como ya mencionamos anteriormente, el índice espacial almacenará solo las boun-

ding boxes de las geometrías, lo cual corresponde a lo mostrado en la Figura 4.10.

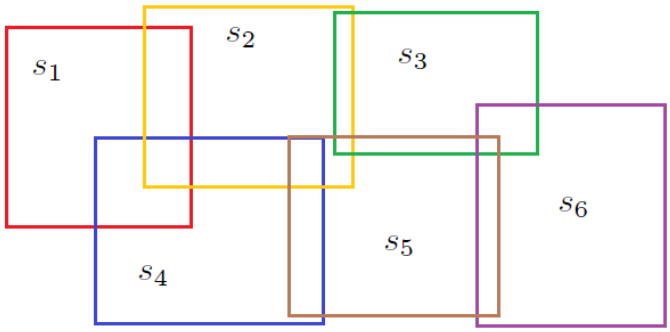

Figura 4.10: Bounding Boxes almacenadas en el índice

Por lo tanto, a la hora de ejecutar la consulta q, el sistema primero consultará las bounding boxes almacenadas en el índice devolviendo aquellas que intersecten el parámetro w, es decir, todos aquellos que compartan puntos con w (todas las que no son Disjoint). En la Figura 4.11 se puede apreciar que las bounding boxes que intersectan a w son las correspondientes a los Estados con Id_s iguales a s_1, s_2 y s_4.

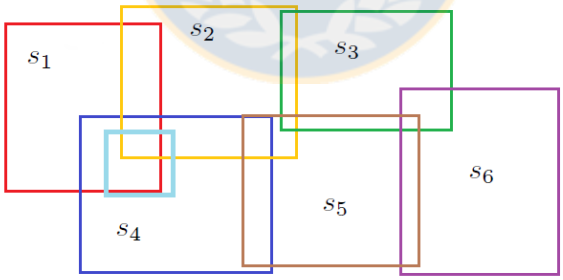

Figura 4.11: Consulta espacial sobre el índice

La consulta realizada sobre el índice tiene un costo de ejecución más bajo que una consulta secuencial, ya que en el índice los datos están organizados en forma de árbol, lo que permite descartar rápidamente los rectángulos que no intersectan la

consulta. Finalmente, se comparan las geometrías correspondientes a s_1, s_2 y s_4 con el parámetro w para obtener los resultados de la consulta. □

Gracias al índice espacial se logra reducir a la mitad el número de geometrías a comparar al ejecutarse la consulta. El sistema utilizará el índice espacial solo para las cosultas cuya condición de búsqueda sea Within, Overlaps, Contains o Touches, mientras que para Disjoint e Equal se analizarán todas las geometrías.

De cierto modo el índice espacial logra lo que nosotros estamos buscando al aplicar nuestra estrategia de optimización. Más adelante se realizarán experimentos para comprobar si efectivamente la optimización por índices espaciales es mejor que la estrategia propuesta por nosotros.

El siguiente ejemplo gráfico muestra un caso donde nuestra optimización funciona mejor que la optimización por índices espaciales.

Ejemplo 4.11. Como mencionamos anteriormente al ejecutar una consulta de rango sobre una tabla que presenta un índice espacial, se logra reducir el número de tuplas a analizar debido a que primero se consultan por las bounding boxes que intersectan al parámetro de búsqueda.
Tomemos la misma instancia de base de datos utilizada hasta el momento en los ejemplos anteriores y un conjunto de restricciones de integridad IC asociada a la base de datos que contiene la restricción:
$ic_1 : \overline{\forall}(County(u, v, w, g) \rightarrow \exists v' w g' State(u', v', w', g') \wedge w = u' \wedge Within(g, g'))$.
Supongamos además una consulta q que desea obtener todos los condados que están sobrepuestos con un parámetro espacial w. Formalmente:
$q(Id_c, G_c) : \exists Id_c\ County(Id_c, Name_c, Id_s, G_c) \wedge \mathsf{Overlaps}(w, G_c)$.
La Figura 4.12 muestra el parámetro w y la consulta sobre la instancia de la base de datos.
Si se utiliza un índice espacial, al ejecutar la consulta el sistema primero consultará el índice en búsqueda de los bounding boxes que intersecten al parámetro w. Sin necesidad de dibujar el bounding box de cada geometría, nos podemos dar cuenta que se seleccionarán los condados marcados en color celeste en la Figura 4.13.
Es decir, utilizando el índice espacial se logró reducir solo en una unidad la cantidad total de condados a analizar por la consulta.
Ahora veamos que ocurre si a la misma consulta se le aplica nuestra estrategia de optimización. Dada la existencia de la restricción de integridad ic_1, se genera un consulta intermedia que relaciona el parámetro w con los estados de la instancia, específicamente se buscarán los estados que se sobrepongan a w o los estados que contengan a w, para luego obtener los condados correspondientes a dichos

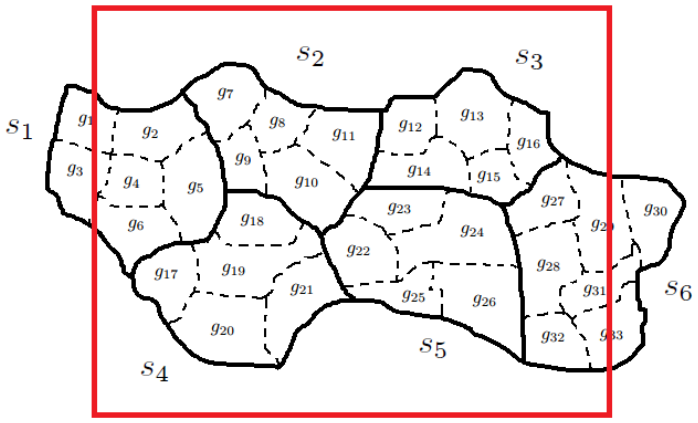

Figura 4.12: Consulta de rango

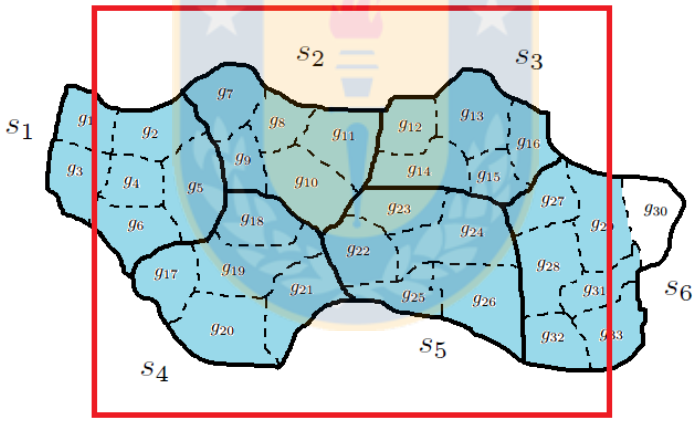

Figura 4.13: Salida del índice espacial

estados. Formalmente la consulta intermedia queda de la forma $q_{int}(Id_c, G_c)$: $\exists Id_c\, County(Id_c, Name_c, Id_s, G_c) \wedge State(Id_s, Name_s, Area, G_s) \wedge (\mathsf{Overlaps}(w, G_s) \vee \mathsf{CoveredBy}(w, G_s)$. La Figura 4.14 muestra los condados devueltos por esta consulta intermedia, los cuales están marcados en celeste.
Como podemos apreciar se logra reducir a 20 el número de condados a analizar por la consulta original. Gracias a la composición de relaciones topológicas se puede saber con anterioridad que es innecesario analizar las geometrías que están conte-

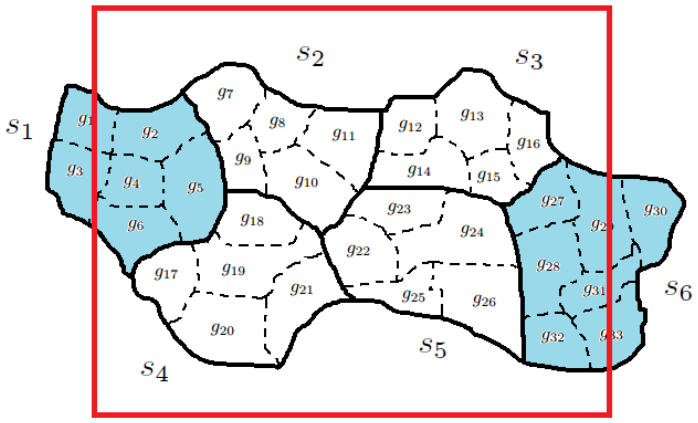

Figura 4.14: Salida de la consulta intermedia

nidas dentro del parámetro w, conocimiento que no es aplicado en la optimización con índices espaciales ya que por defecto se devolverán todas las geometrías cuyas bounding boxes se intersecten con el parámetro w. □

Es por eso que es esperable que nuestra estrategia de optimización sea mejor que la utilizada por el índice espacial cuando la composición T_c no está formada por Contains, ya que en ese caso utilizar un R-Tree es mejor.

Capítulo 5

Algoritmos de Optimización

Las estrategias de optimización explicadas anteriormente pueden ser representadas en dos diferentes grupos de algoritmos, uno por cada tipo de consulta. A continuación mostraremos los algoritmos y además se analizará su orden de complejidad. Se utilizará la siguiente notación para simplificar los algoritmos:

- $\mathsf{Q}_\emptyset$ representará una consulta vacía, es decir, una consulta cuyo conjunto de respuesta es vacío. Por otro lado, cuando a una consulta se le asigne el valor null, significará que el algoritmo no generó una nueva consulta.
- $\mathcal{C}$ es un conjunto de tuplas (T, T') de relaciones topológicas que denotan las composiciones admisibles para llevar a cabo la optimización en el algoritmo 6.
- Con un poco de abuso de notación nos referiremos a un predicado, por ejemplo $R(\bar{x}, g)$, simplemente como R. De la misma forma, algunas veces la notación de las restricciones, por ejemplo $\forall \bar{y}_1 \bar{y}_2 g_1 g_2 (R(\bar{y}_1, g_2) \wedge P(\bar{y}_2, g_2) \rightarrow T(g_1, g_2)$, quedará simplemente como $P \wedge R \rightarrow T$.
- Al utilizar una restricción RD de la forma $\forall \bar{p} \bar{u} \bar{v} g_1 (R(\bar{p}, \bar{u}, \bar{v}, g_1) \rightarrow \exists \bar{x} \bar{y} g_2$ $P(\bar{x}, \bar{y}, g_2) \wedge \bar{u} = \bar{x} \wedge T_{rc}(g_1, g_2))$, asumiremos que $\bar{x}$ en $P(\bar{x}, \bar{y}, g_2)$ es la clave primaria del predicado.

5.1. Algoritmo de Join Query

Tomando en cuenta la estrategia de optimización para las consultas de join explicada anteriormente un posible algoritmo que refleja dicha estrategia se presentará a continuación. Recordemos que para llevar a cabo la optimización de este tipo de consultas es necesario realizar un pre-procesamiento a las restricciones de integridad asociadas a la base de datos, con el fin de contar con la mayor información posible con respecto a las tuplas de la base de datos. Además, como ya se men-

cionó anteriormente, para asegurar que el algoritmo funcione es necesario:

1. Que las tuplas de la base de datos cumplan con las restricciones de integridad.
2. Que el conjunto de restricciones de integridad sea acíclico.

En caso de que las restricciones no cumplan con lo señalado anteriormente el algoritmo podría entregar un resultado erróneo o simplemente no terminar su ejecución.

El Algoritmo 1 construirá el grafo de dependencia y composición del conjunto de restricciones de integridad y el Algoritmo 3 tomará el grafo creado y generará las nuevas restricciones de integridad asociadas a la base de datos.

Algorithm 1 Preprocessing Algorithm

Input: A set Ψ of integrity constraints.
Output: A set Φ of ETDs derived from Ψ

$g \leftarrow$ newGraph() {Create a graph with no nodes}
for every $\forall x \bar{v} g_1(P(\bar{x}, \bar{u}, g_1) \rightarrow \exists \bar{y} g_2 R(\bar{y}, \bar{v}, g_2) \wedge \bar{u} = \bar{y} \wedge T_r(g_1, g_2))$ in Ψ and $\bar{y}$ a superkey of R **do**
  if g.searchNode($P(\bar{x}, \bar{u}, g_1)$) is true **then**
    $n \leftarrow g$.getNode($P(\bar{x}, \bar{u}, g_1)$)
  else
    $n \leftarrow$ newNode($P(\bar{x}, \bar{u}, g_1)$)
  if g.searchNode($R(\bar{y}, \bar{v}, g_2)$) is true **then**
    $n' \leftarrow g$.getNode($R(\bar{y}, \bar{v}, g_2)$)
  else
    $n' \leftarrow$ newNode($R(\bar{y}, \bar{v}, g_2)$)
  $l1 \leftarrow P(\bar{x}, \bar{u}, g_1)$
  $l2 \leftarrow R(\bar{y}, \bar{v}, g_2)$
  $l3 \leftarrow \bar{u} = \bar{y}$
  $l4 \leftarrow T_r(g_1, g_2)$
  g.addEdge($n, n', l1, l2, l3, l4$) {It adds an edge between n and n', with four labels on it}
$g' \leftarrow$ pathComposition(g) { It adds edges by topological composition and thematic conditions as labels on edges}
return createETDs(g,Ψ) {It creates ETDs from graph g}

Algorithm 2 pathComposition

Input: A graph g representing integrity constraints with labeled edges.
Output: A graph g with more edges representing the path composition

```
for every node n of g starting from a node with no parent in a top-down approach do
    for every edge e of n do
        m ← the child of n that is connected through edge e
        for every edge e' of m do
            s ← the child of m that is connected through edge e'
            if There is no edge from node n to node s then
                l1 ← e.l1 ∧ e.l2
                l2 ← e'.l2
                l3 ← e.l3 ∧ e'.l3
                l4 ← e.l4 ⊗ e'.l4
                g.addEdge(n, s, l1, l2, l3, l4)
return g
```

Algorithm 3 createETDs

Input: A graph g and the set Ψ of integrity constraints
Output: A set of ETDs created from g

```
Φ ← {}
for every edge e ∈ g from node n to n' with n ≠ n' do
    φ ← createETD_=(e) {It creates an ETD with n and n' using the label on e}
    if φ ∉ Ψ then
        Φ.Add(φ){It only inserts ETDs (TDs) that do not exist}
    if ∃ψ ∈ Ψ such that ψ is a TD with n' and m as predicates and with m ≠ to
    any predicate on the label of e then
        φ ← createETD(e, ψ)
        Φ.Add(φ)
return Φ
```

Algorithm 4 Optimizador Join

Input: A Join Query $q(\bar{x}) : \exists \bar{z}(P(\bar{y}_1, g_1) \wedge R(\bar{y}_2, g_2)[\wedge\psi_q] \wedge \mathcal{T}_q(g_1, g_2))$ and a set Φ of ETDs (TDs).

Output: $\mathsf{Q}_\emptyset$, q or a query q', which is a rewrite of query q

Let ϕ a ETD$_=$ (or TD$_=$) in Φ of the form $\forall \bar{x}_i g_i(P(\bar{x}_1, g_1) \wedge R(\bar{y}_2, g_2)[\wedge\rho][\wedge\psi_1][\wedge\psi_2] \rightarrow \mathcal{T}_\phi(g_1, g_2)$; otherwise ϕ is $null$.

Let ϕ' a ETD$_{\neq}$ (or TD$_{\neq}$) in Φ of the form $\forall \bar{x}_i g_i(P(\bar{x}_1, g_1) \wedge R(\bar{y}_2, g_2)[\wedge\rho][\wedge\psi_1][\wedge\psi'_2] \rightarrow \mathcal{T}_{\phi'}(g_1, g_2)$ with $\psi'_2 = \neg\psi_2$; otherwise ϕ' is $null$.

if $\phi = null$ **then**

  return q

else if $\phi \neq null$ **then**

  $q_1 \leftarrow$ Rewrite(q, ϕ)

  if $\phi' \neq null$ **then**

    $q_2 \leftarrow$ Rewrite(q, ϕ')

  else

    $q_2 \leftarrow \bar{\exists} P(x, g_1) \wedge R(y, y_2) \wedge \rho_\phi[\wedge\psi_1][\wedge\neg\psi_2] \wedge \psi_q \wedge \mathcal{T}_q$

$q_r \leftarrow$ CleanQuery$(q_1 \cup q_2)$ {It returns $Q_\emptyset$ for contradictions in thematic conditions or returns a query without duplicated conditions}

return q_r

Algorithm 5 Rewrite

Input: A Join Query $q(\bar{x}) : \exists\bar{z}(P(\bar{y}_1, g_1) \wedge R(\bar{y}_2, g_2)[\wedge\psi_q] \wedge \mathcal{T}_q(g_1, g_2))$ and a ETD (TD) Φ of the form $\forall\bar{x}_i g_i(P(\bar{x}_1, g_1) \wedge R(\bar{y}_2, g_2)[\wedge\rho_\phi][\wedge\psi_\phi] \rightarrow \mathcal{T}_\phi(g_1, g_2)$.
Output: $Q_\emptyset$ or a query q'

```
if (T_q ∩ T_φ = ∅) then
    return Q_∅
else if (T_φ ⊆ T_q) then
    return ∃̄P(x, g1) ∧ R(y, y2) ∧ ρ_φ ∧ ψ_φ ∧ ψ_q
else
    return ∃̄P(x, g1) ∧ R(y, y2) ∧ ρ_φ ∧ ψ_φ ∧ ψ_q ∧ T_q
```

Demostración del algoritmo

A continuación demostraremos que la consulta generada por el algoritmo 4 es equivalente a la consulta original q. Es decir, el conjunto de respuesta que se obtiene de aplicar q sobre una instancia D de una base de datos es el mismo que se obtiene al aplicar q^o sobre la misma instancia.

Dada una consulta $q(\bar{u}, g_1, g_2) : \exists\bar{z}P(\bar{x}, g_1) \wedge R(\bar{y}, g_2) \wedge T(g_1, g_2)$, con $x_1 \in \bar{x}$ la clave primaria del predicado P y $y_1 \in \bar{y}$ la clave primaria del predicado R. La idea del algoritmo es poder, a través de las restricciones de integridad, deducir la relación topológica que existe entre cada par de tuplas P, R de la base de datos, para de esa manera retornar las tuplas que cumplen con la consulta q sin necesidad de calcular $T(g_1, g_2)$ para cada par de tuplas.

Para demostrar que ambas consultas son equivalentes debemos demostrar que con nuestra estrategia logramos efectivamente:

1) Deducir, a través de las restricciones de integridad, la relación topológica que existe entre todos los pares de tuplas de P y R.
2) Escoger correctamente, a través de la condición de búsqueda de la nueva consulta, aquellos pares de tuplas que satisfacen la consulta original q.

Esto debdo a que se está trabajando en una base de datos que satisface sus restricciones de integridad, por lo tanto si logramos cumplir el punto 1 y el punto 2 estaremos seguros de que la consulta reescrita generará el mismo conjunto de respuesta que la consulta original, es decir, será equivalente a la consulta original.

Lema 5.1. *Con las restricciones de integridad es posible deducir la relación topológica que existe entre cada par de tuplas P, R de una base de datos.*

Demostración. (i) Sea ic una restricción de integridad de tipo TD de la forma $\forall \bar{x}\bar{y}\ g_1\ g_2(P(\bar{x}, g_1) \wedge R(\bar{y}, g_2) \rightarrow T'(g_1, g_2))$.

La restricción ic nos indica que cada par de tuplas P, R cumple con la relación $T'(g_1, g_2)$ sobre sus atributos espaciales, y dado que la estrategia de optimización se aplica sobre bases de datos que satisfacen las restricciones de integridad asociadas a ellas, se tiene que efectivamente con esta restricción se puede deducir la relación topológica que existe entre cada par de tuplas P, R de la base de datos.

(ii) Sea ic_1 una restricción de integridad de tipo $TD_{=}$ de la forma $\forall \bar{x}\bar{y}\ g_1\ g_2(P(\bar{x}, g_1) \wedge R(\bar{y}, g_2) \wedge x_1 = y_1 \rightarrow T_1(g_1, g_2))$, sea ic_2 una restricción de integridad de tipo $TD_{\neq}$ de la forma $\forall \bar{x}\bar{y}\ g_1\ g_2(P(\bar{x}, g_1) \wedge R(\bar{y}, g_2) \wedge x_1 \neq y_1 \rightarrow T_2(g_1, g_2))$.

La restricción ic_1 abarca los pares de tuplas donde se cumple la condición $x_1 = y_1$, y la restricción ic_2 abarca los pares de tuplas donde se cumple la condición $x_1 \neq y_1$. Estas condiciones corresponden a igualdades o desigualdades entre atributos temáticos, para cada par de tuplas P, R se tienen solo dos opciones con respecto a dos atributos en particular, o son iguales o son diferentes, por lo tanto el conjunto total de pares de tuplas de los predicados P y R se puede dividir en aquellos donde se cumple la condición $x_1 = y_1$ y aquellos donde se cumple la condición $x_1 \neq y_1$.

La restricción ic_1 nos indica que los pares de tuplas P, R donde ψ_ϕ es cierto, cumplen con la relación $T_1(g_1, g_2)$ sobre sus atributos espaciales y la restricción ic_2 nos indica que los pares de tuplas P, R donde $\neg\psi_\phi$ es cierto, cumplen con la relación $T_2(g_1, g_2)$ sobre sus atributos espaciales, y dado que la base de datos satisface las restricciones de integridad se tiene que, efectivamente, estas restricciones nos permite deducir la relación topológica que existe entre cada par de tuplas P, R de la base de datos.

(iii) Sea ic_3 una restricción de integridad de tipo ETD de la forma:
$\forall \bar{x}\bar{y}\ g_1\ g_2(P(\bar{x}, g_1) \wedge R(\bar{y}, g_2) \wedge W(\bar{w}, g_3) \wedge x_2 = w_1 \wedge w_2 = y_1 \rightarrow T_1(g_1, g_2))$,
y sea ic_4 una restricción de integridad de tipo ETD de la forma:
$\forall \bar{x}\bar{y}\ g_1\ g_2(P(\bar{x}, g_1) \wedge R(\bar{y}, g_2) \wedge W(\bar{w}, g_3) \wedge x_2 = w_1 \wedge w_2 \neq y_1 \rightarrow T_2(g_1, g_2))$.

Se sabe que ambas ETDs fueron generadas a través de restricciones de tipo RD, por lo tanto se sabe que para cada una de las tuplas de P debe existir al menos una tupla de W tal que se cumpla $x_2 = w_1$, por lo tanto hasta este punto la restricción considera todas las tuplas de P. Luego para esas tuplas W consideradas y cada una de las tuplas de R se pueden cumplir dos cosas: (i) $w_2 = y_1$ es cierto o (ii) $w_2 \neq y_1$

es cierto.

La restricción ic_3 abarcará los pares de tuplas entre cada tupla de P con aquellas tuplas de R que cumplan con $w_2 = y_1$ y la restricción ic_4 abarca los pares de tuplas entre cada tuplas P con el resto de tuplas R que cumplen con $w_2 \neq y_1$.

Por lo tanto con estas dos restricciones es posible deducir la relación topológica entre cada par de tuplas de P y R.

Esta demostración se extiende al caso cuando las ETDs utilizadas tienes 4 o más predicados, sea ic_5 una restricción de integridad de tipo ETD de la forma: $\forall \bar{x}_i g_i(P(\bar{x}_1, g_1) \wedge W_2 .... \wedge W_{n-1} \wedge R(\bar{x}_n, g_n) \wedge x_1 = \bar{x}_2 \wedge .... \wedge x_{n-1} = \bar{x}_n \wedge \mathcal{T}(g_1, g_n)$, y sea ic_6 una restricción de integridad de tipo ETD de la forma: $\forall \bar{x}_i g_i(P(\bar{x}_1, g_1) \wedge .... \wedge R(\bar{x}_n, g_n) \wedge \bar{x}_1 = \bar{u}_2 \wedge .... \wedge \bar{x}_{n-1} \neq \bar{u}_n \wedge \mathcal{T}(g_1, g_n)$. Al igual que el caso anterior se sabe que ambas ETDs fueron generadas a través de restricciones de tipo RD, por lo tanto se sabe que para cada una de las tuplas de P debe existir al menos una tupla de W_2 tal que se cumpla $x_1 = \bar{x}_2$ y así sucesivamente, deben existir tuplas de cada W_i tal que se cumpla la igualdad entre atributos temáticos de la restricción, hasta el punto de considerar tuplas de W_{n-1}. Luego para esas tuplas W_{n-1} consideradas y cada una de las tuplas de R se pueden cumplir dos cosas: (i) $\bar{x}_{n-1} = \bar{u}_n$ es cierto o (ii) $\bar{x}_{n-1} \neq \bar{u}_n$ es cierto.

La restricción ic_5 abarcará los pares de tuplas entre cada tupla de P con aquellas tuplas de R que cumplan con $\bar{x}_{n-1} = \bar{u}_n$ y la restricción ic_6 abarca los pares de tuplas entre cada tuplas P con el resto de tuplas R que cumplen con $\bar{x}_{n-1} \neq \bar{u}_n$.

□

Teorema 5.1. *La consulta q^o generada por el algoritmo 4 es equivalente a la consulta original q*

Demostración. Sea T la condición de búsqueda espacial de la consulta q y sea T' la relación topológica que existe entre los pares P, R de la base de datos. Tanto T como T' pueden ser una sola relación o una disyunción de relaciones.

i) Si la intersección de T y T' es vacío, entonces ningún par de tuplas de P y R cumple con la condición de búsqueda en q, por lo tanto esos pares de tuplas no deben ser devueltas.
ii) Si T' es un subconjunto de T, entonces eso quiere decir que todos los pares de tuplas de P y R cumplen con al menos una de las disyunciones de la

condición de búsqueda, por lo tanto todos los pares de tuplas de P y R deben ser devueltos.

iii) En cualquier otro caso, es decir, si T es un subconjunto de T' o si simplemente su intersección no es vacía, se tendrá que existe la posibilidad de que algún par de tuplas de P y R cumpla con alguna disyunción de la condición de búsqueda, por lo tanto es necesario chequear los pares con la condición de búsqueda T para poder retornarlos.

En caso de utilizar dos restricciones para la optimización, se tendrá que un grupo (G_1) de pares de tuplas P, R cumplirá con la relación topológica T_1 y otro grupo (G_2) de pares de tuplas P, R cumplirá con la relación topológica T_2, donde se sabe que $G_1 \cap G_2 = \emptyset$ y que $G_1 \cup G_2$ son todos los pares de tuplas P, R de la base de datos. Por lo tanto realizando el procedimiento explicado anteriormente a cada grupo de tuplas y luego uniendo los resultados se logrará obtener el mismo conjunto de respuesta que la consulta original.

Se tiene entonces que la consulta reescrita logrará obtener el mismo conjunto de respuesta que se obtendría de ejecutar la consulta original, por lo tanto se demuestra que ambas consultas, q y q^o son equivalentes.

□

5.1.1. Complejidad algoritmo de Join Query

Los algoritmos 1 y 2 corresponden al pre-procesamiento de las restricciones. Dicho procedimiento depende de la cantidad de restricciones de tipo RD asociadas a la base de datos. Sea m el número de restricciones total en la base de datos, donde m_r es el número de restricciones de tipo RD y m_t el número de restricciones de tipo TD, se tiene que $m_r + m_t = m$. Sea n el número de nodos del grafo, en el peor de los casos el grafo tendrá $n = 2m$ nodos, lo cual se dará si todas las restricciones son del tipo RD y si todos los predicados de las restricciones son distintos.

Para generar el grafo el algoritmo debe analizar todas las m restricciones. Cada vez que se toma una RD se verifica si existen los nodos correspondientes a sus dos predicados, de no existir se crean y se añade el arco correspondiente. Para eso se debe tomar primero un predicado de la restricción y compararlo con todos los nodos existentes hasta el momento en el grafo. Cuando se toma la primera restricción el grafo no contiene nodos, por lo cual no se realizan comparaciones, luego al tomar la segunda restricción el grafo contiene dos nodos, por lo tanto se comparará el primer predicado con los 2 nodos existentes y luego el segundo predicado con los mismos dos nodos, realizando cuatro comparaciones. La Tabla 5.1 muestra el número de comparaciones que se realiza en cada ocasión.

Restricción	Nodos existentes	Comparaciones
1	0	0
2	2	4
3	4	8
4	6	12
...	...	...
m_r	$2m_r$-2	$4m_r$-4

Cuadro 5.1: Número de nodos en el grafo

Como se puede ver el número de comparaciones totales que se debe realizar para generar el grafo se puede representar mediante la siguiente sumatoria

$$\sum_{i=1}^{m_r}(4i-4) = 4\sum_{i=1}^{m_r}(i-1) = 4\sum_{i=0}^{m_r-1} i = 4 \cdot \frac{m_r(m_r-1)}{2} = 2m_r^2 - 2m_r$$

Es decir, el orden de complejidad de la creación del grafo será de $O(m_r^2)$.

Luego de crear el grafo a partir de las restricciones de integridad, el algoritmo pathComposition genera nuevos arcos realizando la composición de relaciones en las etiquetas de los arcos existentes, este procedimiento toma cada uno de los n nodos del grafo, partiendo desde los nodos que no tienen padre, y verifica cada uno de sus hijos para crear nuevos arcos hacia los demás nodos. El peor caso para esta parte del algoritmo se da cuando en la etapa anterior se genera un grafo conectado con la mayor cantidad de nodos posibles dependiendo de las restricciones, lo cual corresponde a tener $m_r + 1$ nodos, es decir, cada nodo del grafo tiene a lo más 1 hijo. La Figura 5.1 muestra un ejemplo de este peor caso, donde se tienen ocho arcos, correspondientes a ocho restricciones de integridad del tipo RD y nueve nodos, formando un grafo conectado. Al tener un grafo de estas características, el algoritmo pathComposition generará un grafo completo.

En un comienzo el algoritmo tomará el nodo 1 del grafo, luego tomará cada uno de sus hijos, en este caso el nodo 2, al tomar el nodo 2 se verifican todos sus hijos, en este caso el nodo 3, por lo tanto en ese momento se verificará si existe un arco entre el nodo 1 y el nodo 3, luego dicho arco se agregará, por lo tanto el nodo 1 tendrá un nuevo hijo (el nodo 3) al cual se le debe aplicar el procedimiento, el algoritmo tomará el hijo de nodo 3 (el nodo 4) y verificará si existe un arco entre 1

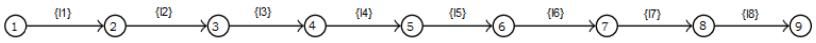

Figura 5.1: Ejemplo de grafo creado

y 4, para luego agregarlo, el procedimiento continuará para cada nodo del grafo. En resumen, teniendo n nodos en el grafo, al tomar el primero de ellos se verificarán y crearán $n-2$ arcos nuevos, luego al tomar el siguiente nodo se verificarán y crearán $n-3$ arcos nuevos, etc. El número de verificaciones y nuevos arcos generados estará dado por la siguiente sumatoria

$$\sum_{i=0}^{n-2}(i) = \frac{n^2 - 3n + 2}{2}$$

Es decir, el orden de complejidad de esta parte del algoritmo es de $O(n^2)$, donde n corresponde al número de nodos del grafo. Dado que el peor caso del algoritmo es cuando se cumple que $n = m_r - 1$, se tiene que el orden de complejidad de esta parte del algoritmo será $O(m_r^2)$.

Finalmente al recorrer el grafo generado para crear las nuevas restricciones se verifican todos los arcos existentes, en el peor de los casos, como se dijo anteriormente, si se generó un grafo completo la cantidad de arcos será de $\frac{n^2-n}{2}$, es decir el número de arcos será $\frac{(m_r-1)^2-(m_r-1)}{2}$, además por cada uno de ellos se buscará una TD para así generar una posible nueva ETD. Por lo tanto, el algoritmo para crear las restricciones tendrá una complejidad de $O(m_r^2 + m_t m_r^2)$, lo cual es equivalente a $O(m_r^2)$ en caso de existir pocas TDs en comparación a las RDs, y a $O(m_t)$ en caso de existir muchas TDs en comparación a las RDs. Suponiendo que existen un número similar de TDs y RDs, el orden de complejidad para crear las restricciones sería de $O(m_t m_r^2)$

Por lo tanto el orden de complejidad total del pre-procesamiento será de $O(m_r^2) + O(m_r^2) + O(m_t m_r^2) = O(m_t m_r^2)$

El pre-procesamiento de las restricciones de integridad se lleva a cabo solo una vez en el tiempo, antes de que se haga cualquier optimización.

El máximo número posible de restricciones existentes que son útiles para la optimización luego del pre-procesamiento esta dado por: el número de TDs existentes + el número de restricciones generadas a partir de las RDs + el posible número de restricciones generadas al componer una restricción del grafo con una TD existente, lo cual es equivalente a $p = m_t + \frac{(m_r-1)^2-(m_r-1)}{2} + m_t * \frac{(m_r-1)^2-(m_r-1)}{2}$.

El algoritmo de optimización propiamente tal debe verificar cada una de las restricciones de integridad existentes en busca de la adecuada para la optimización, con la posibilidad de llevar a cabo dos veces esa búsqueda cuando se trabaja con dos restricciones para la optimización, por lo tanto si p es el número de restricciones existentes, en el peor caso se verificarán $2p$ restricciones, luego el orden de complejidad de la optimización será de $O(p)$, donde, dependiendo de la relación entre m_t y m_r, p puede variar entre m_t y m_r^3.

5.2. Algoritmo de Range Query

Tomando en cuenta la estrategia de optimización para las consultas de rango explicadas anteriormente un posible algoritmo que refleja dicha estrategia es el algoritmo 6.

Algorithm 6 RQO Algorithm

Input: A Range Query $q(\bar{u}, g) : \exists \bar{z} P(\bar{p}, \bar{x}, \bar{v}, g) \wedge T(g, w)$, where $\bar{p}$ is the primary key of P and $\bar{p} \cup \bar{x} \cup \bar{v} = \bar{z} \cup \bar{u}$, a set of integrity constraints Ψ, a database instance D, and a set $\mathcal{C}$ of admisible topological compositions.

Output: A query which is equivalent to q.

1: $\Phi \leftarrow$ a subset of Ψ of the form: $\forall x \bar{v} g_1 (P(\bar{p}, \bar{u}, \bar{v}, g_1) \rightarrow \exists \bar{y} g_2 R(\bar{x}, \bar{y}, g_2) \wedge \bar{u} = \bar{x} \wedge T'(g_1, g_2))$, for each $R \in \mathcal{R}$.
2: **while** $\Phi \neq \emptyset$ **do**
3: $\psi \leftarrow$ a constraint $\forall x \bar{v} g_1 (P(\bar{p}, \bar{u}, \bar{v}, g_1) \rightarrow \exists \bar{y} g_2 R(\bar{x}, \bar{y}, g_2) \wedge \bar{u} = \bar{x} \wedge T'(g_1, g_2))$ in Φ
4: **if** (SIZE(R) $<$ SIZE(P) $\wedge$ $((T, T') \in \mathcal{C} \vee (T', T) \in \mathcal{C})$ {SIZE(R) returns the number of tuples from predicate R in D} **then**
5: $q_{int}(\bar{p}, \bar{x}, \bar{v}, g) \leftarrow$ CHASE$_{RD}(q, \psi)$ {CHASE$_{RD}(q, \psi)$ applies Chase to q using the constraint ψ}
6: $q_{int}(\bar{p}, \bar{x}, \bar{v}, g) \leftarrow$ REPLACE$_c(q', T, T')$ {REPLACE$_c(q', T, T')$ replaces the relations T and T' in q_{int} by the composition $T \otimes T'$ }
7: $R'(\bar{p}) \leftarrow q_{int}(\bar{p}, \bar{x}, \bar{v}, g)[D]$ {$R'(\bar{p})$ the reslt of applying $q_{int}(\bar{p}, \bar{x}, \bar{v}, g)$ over D, that is, a predicate with a single attribute, p}
8: **return** $q^o(\bar{u}, g) : \exists \bar{z} \cup y (P(\bar{p}, \bar{x}, \bar{v}, g) \wedge R'(y) \wedge \bar{p} = \bar{y} \wedge T(g, w))$
9: $\Phi \leftarrow \Phi \setminus \psi$
10: **return** $q(\bar{u}, g)$

Demostración del algoritmo

A continuación demostraremos que la consulta generada por el algoritmo RQO es equivalente a la consulta original q, es decir, el conjunto de respuesta que se obtiene de aplicar q sobre una instancia D de una base de datos es el mismo que se obtiene de aplicar q^o sobre la misma instancia.

Primero que todo recordemos el procedimiento que se lleva a cabo: el objetivo principal del algoritmo es reducir el número de tuplas a analizar en la consulta q. Esta reducción se lleva a cabo mediante la ejecución de una consulta intermedia q_{int}, la consulta q_{int} es generada a partir de q utilizando las restricciones de integridad junto a la técnica Chase y la composición de relaciones topológicas. Finalmente la consulta original se ejecuta tomando en cuenta las tuplas obtenidas de la ejecución de q_{int}.

Dada una consulta de rango $q(\bar{u}, g) : P(\bar{x}, \bar{v}, g) \wedge T(w, g)$; donde w es un parámetro espacial arbitrario y T una relación topológica y dada una restricción de integridad ic de la forma $\forall \bar{x}\bar{v}g_1(P(\bar{x}, \bar{v}, g_1) \rightarrow \exists \bar{z}\bar{y}g_2 R(\bar{z}, \bar{y}, g_2) \wedge \bar{v} = \bar{z} \wedge T'(g_1, g_2))$.

Sea q_{int} la consulta intermedia generada de la aplicación del algoritmo RQO a la consulta q definida de la siguiente forma $q_{int}(\bar{u}, g) : P(\bar{x}, \bar{v}, g) \wedge R(\bar{v}, \bar{y}, g_2) \wedge T''(w, g_2)$.

Sea D la instancia de una bases de datos, es decir, el set de tuplas que conforman la base de datos, sea $q[D]$ el conjunto de tuplas resultantes de la aplicación de q sobre la instancia D y sea $q_{int}[D]$ el conjunto de tuplas resultantes de la aplicación de q_{int} sobre la instancia D. Sea q^o la consulta final optimizada obtenida del algoritmo, el conjunto de tuplas resultantes de la aplicación de q^o sobre la instancia D se puede definir como: $q^o[D] : (q[q_{int}[D]])$.

Dado que en la parte final de la optimización se aplica la consulta original q sobre las tuplas devueltas por q_{int} intuitivamente se puede notar que $(q^o[D] = q[D])$ se cumplirá si y solo si en las tuplas devueltas por la consulta q_{int} $(q_{int}[D])$ están todas las tuplas pertenecientes al conjunto $q[D]$.

Lema 5.2. $(q^o[D] = q[D]) \leftrightarrow (q[D] \subseteq q_{int}[D])$

Demostración. (i) Se demuestra $(q^o[D] = q[D]) \rightarrow (q[D] \subseteq q_{int}[D])$ por contradicción.

Supongamos que $(q^o[D] = q[D])$ es cierto pero que existe una tupla $t : (\bar{a}, s)$ que

pertenece al conjunto $q[D]$ pero no pertenece al conjunto $q_{int}[D]$. Es decir, supongamos $((t \in q[D]) \wedge (t \notin q_{int}[D]))$. Sabemos que toda tupla en $q[D]$ está en $q^o[D]$, por lo tanto, como se cumple $(t \in q[D])$, también se cumple $(t \in q^o[D])$.

Además sabemos que $q^o[D] = q[q_{int}[D]]$. De esto se deduce que cualquier tupla que está en $q^o[D]$ debe estar en $q_{int}[D]$, ya que las consultas de rango (q) solo seleccionan y no agregan tuplas, entonces se concluye $(t \in q_{int}[D])$, hemos alcanzado una contradicción.

(ii) Se demuestra $(q[D] \subseteq q_{int}[D]) \rightarrow (q^o[D] = q[D])$

Partiendo de $(q[D] \subseteq q_{int}[D])$ se concluirá $(q^o[D] = q[D])$. Se sabe que el conjunto $q_{int}[D]$ es un subconjunto de D. Ahora, que se cumpla $(q[D] \subseteq q_{int}[D])$ significa que en $q_{int}[D]$ están todas las tuplas de D que cumplen con la condición de búsqueda de q, por lo tanto al ejecutar la consulta q sobre el conjunto $q_{int}[D]$ se obtendrán todas esas tuplas, es decir $q[q_{int}[D]] = q[D]$ y dado que por definición tenemos $q^o[D] = q[q_{int}[D]]$, se cumple también $q^o[D] = q[D]$.

□

Ahora veremos que en nuestro algoritmo la relación $(q[D] \subseteq q_{int}[D])$ siempre se cumplirá.

Lema 5.3. $(q[D] \subseteq q_{int}[D])$

Demostración. Se demuestra $(q[D] \subseteq q_{int}[D])$ por contradicción.

Supongamos que existe una tupla $(\bar{u}, g)$ definida como $t : (\bar{a}, s)$ que pertenece al conjunto $q[D]$ pero no pertenece al conjunto $q_{int}[D]$, es decir supongamos $((t \in q[D]) \wedge (t \notin q_{int}[D]))$.

Dado que $t \in q[D]$ (sup.) entonces debe existir una tupla p del predicado P y un atributo $\bar{b}$ tal que $p : (\bar{a}, \bar{b}, s)$ donde el atributo s de la tupla p cumpla con la relación $T(w, s)$. Además debido a la restricción de integridad ic, también debe existir una tupla r del predicado R, un atributo $\bar{c}$ y un atributo espacial s' tal que $r : (\bar{c}, s')$, donde $\bar{b} = \bar{c}$ y la relación $T'(s, s')$ se cumpla.

En estos momentos sabemos que se cumplen las siguientes dos relaciones topológicas: $T(w, s)$ y $T'(s, s')$, por lo tanto, utilizando la definición de composición de relaciones, es posible establecer una relación topológica entre el parámetro w y el atributo espacial s', la cual corresponde a $T''(w, s')$.

Entonces si aplicáramos la consulta q_{int} sobre la instancia D, la tupla $r : (\bar{c}, s')$ sería una de las que cumpliría con la condición de búsqueda $T''(w, g_2)$, luego se recolectarán todas las tuplas del predicado P donde el atributo $\bar{b}$ sea igual a $\bar{c}$, entre ellas se incluirá la tupla $p : (\bar{a}, \bar{b}, s)$, para finalmente devolver tuplas del tipo $(\bar{u}, g)$, en este caso la tupla $t : (\bar{a}, s)$, lo cual quiere decir que $t \in q_{int}[D]$. Luego hemos llegado a una contradicción.

Por lo tanto se concluye $\neg((t \in q[D]) \wedge (t \notin q_{int}[D]))$, utilizando algunas equivalencia lógicas se obtiene:

$$
\begin{aligned}
&\neg((t \in q[D]) \wedge (t \notin q_{int}[D])) \equiv \\
&(\neg(t \in q[D]) \vee \neg(t \notin q_{int}[D])) \equiv \\
&((t \notin q[D]) \vee (t \in q_{int}[D])) \equiv \\
&(\neg(t \notin q[D]) \rightarrow (t \in q_{int}[D])) \equiv \\
&((t \in q[D]) \rightarrow (t \in q_{int}[D]))
\end{aligned}
$$

Es decir, si existe una tupla t que está en $q[D]$ también debe estar en $q_{int}[D]$, lo que es equivalente a decir $(q[D] \subseteq q_{int}[D])$.

□

Finalmente demostraremos que las consultas q y q^o son equivalentes.

Teorema 5.2. $q^o \equiv q$

Demostración. Dada cualquier instancia de una base de datos, por definición se tiene que dos consultas serán equivalentes si y solo si al aplicar ambas consultas sobre la misma instancia se obtiene el mismo conjunto de respuesta. Es decir $q^o \equiv q \leftrightarrow q^o[D] = q[D]$. Por el Lemma 5.2 sabemos que $q^o[D] = q[D] \leftrightarrow (q[D] \subseteq q_{int}[D])$, por lo tanto se tiene $q^o \equiv q \leftrightarrow (q[D] \subseteq q_{int}[D])$ y dado que $(q[D] \subseteq q_{int}[D]$ es siempre cierto (Lemma 5.3) se demuestra que q^o es equivalente a q.

□

5.2.1. Complejidad algoritmo de Range Query

Sea n el número de restricciones de integridad presentes en el conjunto Ψ y sea m el número de composiciones admisibles en el conjunto $\mathcal{C}$. En el peor de los casos debemos tomar las n restricciones presentes en el conjunto y al escoger cada una debemos verificar si la composición generada esta dentro de las m composiciones admisibles para llevar a cabo la optimización, por lo tanto podemos decir que la complejidad del algoritmo será $O(nm)$.

Capítulo 6

Experimentos y Resultados

En este capítulo se describirán los datos espaciales utilizados para llevar a cabo los experimentos. Además se definirán las consultas de rango y de join que fueron optimizadas con los algoritmos propuestos, finalmente se compararán los tiempos de ejecución de las consultas y se discutirán los resultados.

6.1. Conjunto de Datos

En los experimentos realizados se utilizaron datos reales, disponibles para todo público, tomados desde la página web del gobierno estadounidense (U.S. Census Bureau). La primera base de datos tomada consiste en un esquema de tres tablas, en cada una de ellas se almacena información sobre los Estados, Condados y Lugares de Estados Unidos, además en cada tabla existe un atributo espacial que define la geografía de la entidad en cuestión. La base de datos fue cargada en el sistema gestor de base de datos PostGIS el cual es una extensión de PostgreSQL permitiendo trabajar de mejor manera con datos espaciales.

Figura 6.1: Relación entre Datos

La Figura 6.1 muestra la relación que existe entre las diferentes tablas de la base de datos. En ella se puede apreciar que un Estado en particular puede contener cualquier cantidad de Condados, mientras que un Condado solo puede estar contenido en un Estado. La misma relación ocurre con los Condados y los Lugares. Esto se traduce en la estructura de las tablas a través de la inserción de un atributo que

representa una clave foránea en los predicados County y Place, los cuales representan el atributo primario del predicado State y County, respectivamente.

Para la realización de los experimentos también se utilizó otra tabla extraída de la misma página y la cual hace referencias a los LandMarks del estado de Texas, esta tabla fue utilizada en un benchmark de evaluación de rendimiento de bases de datos espaciales en [16]. Las consultas que se utilizarán para esta tabla también fueron sacadas del benchmark nombrado.

Schema	Description	Keys	Tuples
State$(sid, name, division, geom)$	State	PK:sid	56
County$(cid, sid, name, geom)$	County	PK:cid, FK:sid	3234
Place$(pid, cid, name, geom)$	Place	PK:pid, FK:cid	28556
Land$(lid, \ldots, geom)$	Landmark	PK:lid	5985

Cuadro 6.1: Datos de la evaluación experimental

6.2. Restricciones y Consultas utilizadas

Para poder llevar a cabo los experimentos es necesario primero definir las restricciones de integridad asociadas a la base de datos, de igual forma se deben definir las consultas de rango y de join que se realizarán sobre los datos.

6.2.1. Restricciones de Integridad

1. $\overline{\forall}\,(\text{State}(gid, \bar{u}, geom) \wedge \text{State}(gid', \bar{v}, geom') \wedge gid = gid' \rightarrow \bar{u} = \bar{v})$
2. $\overline{\forall}\,(\text{State}(gid, \bar{u}, geom) \wedge \text{State}(gid', \bar{v}, geom') \wedge gid = gid' \rightarrow \text{Equal}(geom, geom'))$
3. $\overline{\forall}\,(\text{County}(gid, \bar{u}, geom) \wedge \text{County}(gid', \bar{v}, geom') \wedge gid = gid' \rightarrow \bar{u} = \bar{v})$
4. $\overline{\forall}\,(\text{County}(gid, \bar{u}, geom) \wedge \text{County}(gid', \bar{v}, geom') \wedge gid = gid' \rightarrow \text{Equal}(geom, geom'))$
5. $\overline{\forall}\,(\text{Place}(gid, \bar{u}, geom) \wedge \text{Place}(gid', \bar{v}, geom') \wedge gid = gid' \rightarrow \bar{u} = \bar{v})$
6. $\overline{\forall}\,(\text{Place}(gid, \bar{u}, geom) \wedge \text{Place}(gid', \bar{v}, geom') \wedge gid = gid' \rightarrow \text{Equal}(geom, geom'))$

Las restricciones 1-6 nos permiten definir la clave primaria de cada una de las tablas.

7. $\overline{\forall}\,(\text{State}(gid, \bar{u}, geom) \wedge \text{State}(gid', \bar{v}, geom') \wedge gid \neq gid' \rightarrow$
$(\mathsf{Disjoint}(geom', geom) \vee \mathsf{Touches}(geom', geom)))$
8. $\overline{\forall}\,(\text{County}(gid, \bar{u}, geom) \wedge \text{County}(gid', \bar{v}, geom') \wedge gid \neq gid' \rightarrow$
$(\mathsf{Disjoint}(geom', geom) \vee \mathsf{Touches}(geom', geom)))$
9. $\overline{\forall}\,(\text{Place}(gid, \bar{u}, geom) \wedge \text{Place}(gid', \bar{v}, geom') \wedge gid \neq gid' \rightarrow$
$(\mathsf{Disjoint}(geom', geom) \vee \mathsf{Touches}(geom', geom)))$

Las restricciones 7-9 indican que 2 regiones diferentes del mismo tipo deben estar disjuntas o tocándose una con la otra.

10. $\overline{\forall}\,(\text{County}(gid, \bar{x}, gid', geom) \rightarrow$
$\exists gid' geom'\,(\text{State}(gid', \bar{y}, geom') \wedge \mathsf{Within}(geom, geom')))$
11. $\overline{\forall}\,(\text{Place}(gid, \bar{x}, gid', geom) \rightarrow$
$\exists geom'\,(\text{County}(gid', \bar{y}, geom') \wedge \mathsf{Within}(geom, geom')))$

Las restricciones 10-11 indican la relación que existe entre las tuplas State-County y County-Place. Notar que en cada caso el atributo gid' es la clave primaria del predicado State (10) y del predicado County.

12. $\overline{\forall}\,(\text{State}(gid, \bar{x}, geom) \wedge \text{County}(gid', \bar{y}, statefk, geom') \wedge gid = statefk \rightarrow$
$\mathsf{Within}(geom', geom))$
13. $\overline{\forall}\,(\text{State}(gid, \bar{x}, geom) \wedge \text{County}(gid', \bar{y}, statefk, geom') \wedge gid \neq statefk \rightarrow$
$(\mathsf{Disjoint}(geom', geom) \vee \mathsf{Touches}(geom', geom)))$
14. $\overline{\forall}\,(\text{County}(gid, \bar{x}, geom) \wedge \text{Place}(gid', \bar{y}, countyfk, geom') \wedge gid = countyfk \rightarrow$
$\mathsf{Within}(geom', geom))$
15. $\overline{\forall}\,(\text{County}(gid, \bar{x}, geom) \wedge \text{Place}(gid', \bar{y}, countyfk, geom') \wedge gid \neq countyfk \rightarrow$
$(\mathsf{Disjoint}(geom', geom) \vee \mathsf{Touches}(geom', geom)))$

Las restricciones 12-15 nos permiten establecer las relaciones topológicas que deben existir entre cada par de tuplas State-County y County-Place.

16. $Land(i, \bar{u}, g) \wedge Land(i', \bar{v}, g') \wedge i = i' \rightarrow \bar{u} = \bar{v}$
17. $Land(i, \bar{u}, g) \wedge Land(i', \bar{v}, g') \wedge i = i' \rightarrow (\mathsf{Equal}(g', g))$

Finalmente las restricciones 16-17 nos permiten establecer la clave primaria de la tabla Landmark.

6.2.2. Consultas Espaciales

Debido a que existen una gran cantidad de posibles consultas espaciales se decidió tomar solo las más *comunes* según experiencia propia para llevar a cabo los

experimentos. Además éstas contienen solo una condición de búsqueda. Las consultas espaciales definidas para realizar los experimentos son las siguientes:

Join Query (JQ):

1. $q(gid, gid', geom, geom') : \exists \bar{x}\bar{y} fk_s State(gid, \bar{x}, geom) \wedge$
$County(gid', \bar{y}, fk_s, geom') \wedge \mathsf{Within}(geom', geom)$;
2. $q(gid, gid', geom, geom') : \exists \bar{v}\bar{u}\ fk_c County(gid, \bar{v}, geom) \wedge$
$Place(gid', \bar{u}, fk_c, geom') \wedge \mathsf{Within}(geom', geom)$;
3. $q(gid, gid', geom, geom') : \exists \bar{x}\bar{u}\ fk_c State(gid, \bar{x}, geom) \wedge$
$Place(gid', \bar{u}, fk_c, geom') \wedge \mathsf{Within}(geom', geom)$;
4. $q(gid, gid', geom, geom') : \exists \bar{u}\bar{v} County(gid, \bar{u}, geom) \wedge$
$County(gid', \bar{v}, geom') \wedge \mathsf{Overlaps}(geom', geom)$;
5. $q(gid, gid', geom, geom') : \exists \bar{u}\bar{v} County(gid, \bar{u}, geom) \wedge$
$County(gid', \bar{v}, geom') \wedge \mathsf{Touches}(geom', geom)$;
6. $q(gid, gid', geom, geom') : \exists \bar{x}\bar{y} State(gid, \bar{x}, geom) \wedge Place(gid', \bar{u}, fk_c, geom') \wedge$
$\mathsf{Overlaps}(geom', geom)$;
7. $q(gid, gid', geom, geom') : \exists \bar{x}\bar{y} State(gid, \bar{x}, geom) \wedge Place(gid', \bar{u}, fk_c, geom') \wedge$
$\mathsf{Touches}(geom', geom)$;
8. $q(id, id', g, g') : \exists \bar{x}\bar{x}' Land(id, \bar{x}, g) \wedge Land(id', \bar{x}', g') \wedge \mathsf{Equal}(g', g)$;
9. $q(id, id', g, g') : \exists \bar{x}\bar{x}' Land(id, \bar{x}, g) \wedge Land(id', \bar{x}', g') \wedge \mathsf{Touches}(g', g)$;
10. $q(id, id', g, g') : \exists \bar{x}\bar{x}' Land(id, \bar{x}, g) \wedge Land(id', \bar{x}', g') \wedge \mathsf{Overlaps}(g', g)$;
11. $q(id, id', g, g') : \exists \bar{x}\bar{x}' Land(id, \bar{x}, g) \wedge Land(id', \bar{x}', g') \wedge \mathsf{Within}(g', g)$;
12. $q(id, id', g, g') : \exists \bar{x}\bar{x}' Land(id, \bar{x}, g) \wedge Land(id', \bar{x}', g') \wedge \mathsf{Contains}(g', g)$;

Range Query (RQ):

13. $q(gid, geom) : \exists \bar{x} fk_s\ County(gid, \bar{x}, fk_s, geom) \wedge \mathsf{Contains}(w, geom)$;
14. $q(gid, geom) : \exists \bar{x} fk_c Place(gid, \bar{x}, fk_c, geom) \wedge \mathsf{Contains}(w, geom)$;
15. $q(gid, geom) : \exists \bar{x} fk_s\ County(gid, \bar{x}, fk_s, geom) \wedge \mathsf{Overlaps}(w, geom)$;
16. $q(gid, geom) : \exists \bar{x} fk_c Place(gid, \bar{x}, fk_c, geom) \wedge \mathsf{Overlaps}(w, geom)$;
17. $q(gid, geom) : \exists \bar{x} fk_s\ County(gid, \bar{x}, fk_s, geom) \wedge \mathsf{Within}(w, geom)$;
18. $q(gid, geom) : \exists \bar{x} fk_c Place(gid, \bar{x}, fk_c, geom) \wedge \mathsf{Within}(w, geom)$;

Donde w es un parámetro espacial, en este caso fue definido como un rectángulo de diferentes tamaños y con diferentes coordenadas, más adelante se darán más detalles sobre el parámetro w.

6.3. Resultados

Pre-procesamiento de Restricciones.

La primera tarea a realizar es deducir las nuevas restricciones de integridad a partir de las 17 existentes. Para eso se ejecuta el pre-procesamiento explicado en el Capítulo 4.

Tomando las restricciones 10) y 11) se forma el grafo de dependencia y composición mostrado en la Figura 4.1

Se generan dos nuevas restricciones de integridad:

18. $\overline{\forall}\,(\mathrm{State}(gid, \bar{x}, geom) \wedge \mathrm{County}(gid', \bar{y}, statefk, geom') \wedge \mathrm{Place}(gid'', \bar{v}, countyfk, geom'') \wedge gid' = countyfk \ \wedge \ gid = statefk \rightarrow \mathsf{Within}(geom'', geom))$
19. $\overline{\forall}\,(\mathrm{State}(gid, \bar{x}, geom) \wedge \mathrm{County}(gid', \bar{y}, statefk, geom') \wedge \mathrm{Place}(gid'', \bar{v}, countyfk, geom'') \wedge gid' = countyfk \wedge gid \neq statefk \rightarrow \mathsf{Disjoint}(geom'', geom) \vee \mathsf{Touches}(geom'', geom))$

Los experimentos consistieron en ejecutar las consultas originales (1-18) para luego ejecutar las consultas optimizadas según el algoritmo correspondiente. Además este proceso se realizó dos veces: el primero sin la presencia de índices espaciales en las tablas y el segundo con la presencia de índices espaciales en las tablas, esto para verificar si la optimización propuesta lograba mejorar la optimización realizada por los índices. La unidad de medida del tiempo de ejecución es de milisegundos.

Todos los experimentos fueron ejecutados en un computador con las siguientes especificaciones: Procesador: Intel Corporation Xeon E3-1200 v2/3rd Gen Core processor DRAM Controller, Memoria RAM: 16 GB, SO: Ubuntu 13.10, software de bases de datos utilizado pgAdmin PostgreSQL 1.18.1 con extension PostGIS. Además los algoritmos fueron implementados en el mismo computador utilizando el lenguaje de programación JAVA.

A continuación se muestran las consultas optimizadas y los tiempos de ejecución de cada una de ellas. El tiempo que toma el algoritmo en generar la nueva consulta está incluida en el tiempo de la consulta optimizada, en general el algoritmo tomo entre 8 y 9 milisegundo en ejecutarse.

Join Query (JQ):

Las consultas se ejecutaron 5 veces y se obtuvo un promedio de tiempo. La Tabla 6.2 muestra que restricciones fueron utilizadas para llevar a cabo la optimización. La Tabla 6.3 muestra las consultas originales y sus correspondientes reescrituras. Mientras que la Tabla 6.4 muestra los tiempos de ejecución de las consultas en milisegundos.

Query	Integrity Constraint
1	12 y 13
2	14 y 15
3	18 y 19
4	4 y 8
5	4 y 8
6	18 y 19
7	18 y 19
8	17
9	17
10	17
11	17
12	17

Cuadro 6.2: Restricciones de integridad utilizadas

#	Query
q_1	$q_1(i,i',g,g') : \exists ndsn'(S(i,n,d,g) \wedge C(i',s,n',g') \wedge \mathsf{Within}(g',g))$
$q_{1_{opt}}$	$q_{1_{opt}}(i,i',g,g') : \exists ndsn'(S(i,n,d,g) \wedge C(i',s,n',g') \wedge i = s)$
q_2	$q_2(i,i',g,g') : \exists sncn'(C(i',s,n',g') \wedge P(i',c,n',g') \wedge \mathsf{Within}(g',g))$
$q_{2_{opt}}$	$q_{2_{opt}}(i,i',g,g') : \exists sncn'(C(i',s,n,g') \wedge P(i',c,n',g') \wedge i = c)$
q_3	$q_3(i,i',g,g') : \exists ndcn'(S(i,n,d,g) \wedge P(i',c,n',g') \wedge \mathsf{Within}(g',g))$
$q_{3_{opt}}$	$q_{3_{opt}}(i,i',g,g') : \exists ndcn'i''sn''g''(S(i,n,d,g) \wedge P(i',c,n',g') \wedge C(i'',s,n'',g'') \wedge i = s \wedge i'' = c)$
q_4	$q_4(i,i',g,g') : \exists sns'n'(C(i,s,n,g) \wedge C(i',s',n',g') \wedge \mathsf{Overlaps}(g',g))$
$q_{4_{opt}}$	Empty query
q_5	$q_5(i,i',g,g') : \exists sns'n'(C(i,s,n,g) \wedge C(i',s',n',g') \wedge \mathsf{Touches}(g',g))$
$q_{5_{opt}}$	$q_{5_{opt}}(i,i',g,g') : \exists sns'n'(C(i,s,n,g) \wedge C(i',s',n',g') \wedge i \neq i' \wedge \mathsf{Touches}(g',g))$
q_6	$q_6(i,i',g,g') : \exists ndn'd'(S(i,n,d,g) \wedge P(i',n',d',g') \wedge \mathsf{Overlaps}(g',g))$
$q_{6_{opt}}$	Empty query
q_7	$q_7(i,i',g,g') : \exists ndcn'(S(i,n,d,g) \wedge P(i',c,n',g') \wedge \mathsf{Touches}(g',g))$
$q_{7_{opt}}$	$q_{7_{opt}}(i,i',g,g') : \exists ndcn'i''sn''g''(S(i,n,d,g) \wedge P(i',c,n',g') \wedge C(i'',s,n'',g'') \wedge i = s \wedge i'' \neq c \wedge \mathsf{Touches}(g',g))$
q_8	$q_1(i,i',g,g') : \exists ndsn'(\mathbf{L}(i,n,d,g) \wedge \mathbf{L}(i',s,n',g') \wedge \mathsf{Equal}(g',g))$
$q_{8_{opt}}$	$q_{1_{opt}}(i,i',g,g') : \exists ndsn'(\mathbf{L}(i,n,d,g) \wedge \mathbf{L}(i',s,n',g') \wedge i = i' \vee \mathsf{Equal}(g',g))$
q_9	$q_2(i,i',g,g') : \exists ndsn'(\mathbf{L}(i,n,d,g) \wedge \mathbf{L}(i',s,n',g') \wedge \mathsf{Touches}(g',g))$
$q_{9_{opt}}$	$q_{2_{opt}}(i,i',g,g') : \exists ndsn'(\mathbf{L}(i,n,d,g) \wedge \mathbf{L}(i',s,n',g') \wedge i \neq i' \wedge \mathsf{Touches}(g',g))$
q_{10}	$q_3(i,i',g,g') : \exists ndsn'(\mathbf{L}(i,n,d,g) \wedge \mathbf{L}(i',s,n',g') \wedge \mathsf{Overlaps}(g',g))$
$q_{10_{opt}}$	$q_{3_{opt}}(i,i',g,g') : \exists ndsn'(\mathbf{L}(i,n,d,g) \wedge \mathbf{L}(i',s,n',g') \wedge i \neq i' \wedge \mathsf{Overlaps}(g',g))$
q_{11}	$q_4(i,i',g,g') : \exists ndsn'(\mathbf{L}(i,n,d,g) \wedge \mathbf{L}(i',s,n',g') \wedge \mathsf{Within}(g',g))$
$q_{11_{opt}}$	$q_{4_{opt}}(i,i',g,g') : \exists ndsn'(\mathbf{L}(i,n,d,g) \wedge \mathbf{L}(i',s,n',g') \wedge i = i' \vee \mathsf{Within}(g',g))$
q_{12}	$q_5(i,i',g,g') : \exists ndsn'(\mathbf{L}(i,n,d,g) \wedge \mathbf{L}(i',s,n',g') \wedge \mathsf{Contains}(g',g))$
$q_{12_{opt}}$	$q_{5_{opt}}(i,i',g,g') : \exists ndsn'(\mathbf{L}(i,n,d,g) \wedge \mathbf{L}(i',s,n',g') \wedge i = i' \vee \mathsf{Contains}(g',g))$

Cuadro 6.3: Consultas generadas

Query	**No Index**		**Index**	
i	q_i	$q_{i_{opt}}$	q_i	$q_{i_{opt}}$
1	14000	208	14000	208
2	420000	1608	14000	1608
3	150000	1608	10000	1608
4	210000	8	135000	8
5	180000	140008	120000	60008
6	189903	8	185052	8
7	190000	70285	189748	66008
8	9276	9123	1281	193
9	9578	9809	1653	461
10	9890	9710	1723	474
11	9479	10315	1442	251
12	9409	10023	1422	202

Cuadro 6.4: Costo de procesamiento de consulta en ms

Range Query (RQ):

Como mencionamos anteriormente, diferentes parámetros w fueron creados para ejecutar las consultas de rango. Se tomó el centroide de cada una de las geometrías de los Estados de la base de datos (56 en total) y a partir de ese punto se creó un rectángulo de diferentes tamaños. En particular se consideraron 5 tamaños distintos, que van desde un radio de 10km hasta un radio de 800km. Por lo tanto cada consulta se ejecutó considerando los 5 tamaños diferentes de w y además 56 veces para cada tamaño (1 por cada centroide considerado).

Los 5 tamaños considerados para el radio del parámetro w son los siguientes: $w1$: 10 kilómetros, $w2$: 100 kilómetros, $w3$: 400 kilómetros, $w4$: 600 kilómetros y $w5$: 800 kilómetro.

Estos tamaños fueron elegidos de tal forma que el área del parámetro más pequeño fuera aproximadamente igual al área promedio de los Condados más pequeños y que el área del parámetro más grande fuera aproximadamente igual al área del Estado más grande.

La Tabla 6.5 muestra que restricciones fueron utilizadas para llevar a cabo la optimización. La Tabla 6.6 muestra las consultas originales y sus correspondientes reescrituras separadas en dos consultas intermedias.

Query	Integrity Constraint
13	10
14	11
15	10
16	11
17	10
18	11

Cuadro 6.5: Restricciones de Integridad utilizadas

A continuación se muestran los tiempos de ejecución de cada consulta (en ms). La tabla compara la consulta original q con la consulta optimizada q^o considerando los 5 diferentes tamaños del parámetro w mencionados anteriormente, además de mostrar cuando las consultas fueron ejecutadas con y sin índices espaciales en la base de datos.
Adicionalmente mostraremos los tiempos de ejecución de las dos subconsultas que componen la consulta optimizada q^o.

#	Query
q_{13}	$q_{13}(gid, geom) : \exists \bar{x} fk_s\ C(gid, \bar{x}, fk_s, geom) \wedge \mathsf{Contains}(w, geom)$
$q_{13_{int}}$	$q_{13_{int}}(gid, \bar{x}, fk_s, geom) : \exists \bar{x} \bar{y}\ fk_s\ gid'\ geom\ geom' C(gid, \bar{x}, fk_s, geom) \wedge S(gid', \bar{y}, geom') \wedge$ $fk_s = gid' \wedge \{\mathsf{Contains}(w, geom') \vee \mathsf{Overlaps}(w, geom') \vee \mathsf{Within}(w, geom')\}$
q'_{13}	$q'_{13}(gid, geom) : \exists \bar{x} fk_s\ C(gid, \bar{x}, fk_s, geom) \wedge gid = q_{13_{int}}(gid) \wedge \mathsf{Contains}(w, geom)$
q_{14}	$q_{14}(gid, geom) : \exists \bar{x} fk_c P(gid, \bar{x}, fk_c, geom) \wedge \mathsf{Contains}(w, geom)$
$q_{14_{int}}$	$q_{14_{int}}(gid, \bar{x}, fk_c, geom) : \exists \bar{x} \bar{y}\ fk_c\ gid'\ geom\ geom'\ P(gid, \bar{x}, fk_c, geom) \wedge C(gid', \bar{y}, geom') \wedge$ $fk_c = gid' \wedge \{\mathsf{Contains}(w, geom') \vee \mathsf{Overlaps}(w, geom') \vee \mathsf{Within}(w, geom')\}$
q'_{14}	$q'_{14}(gid, geom) : \exists \bar{x} fk_c\ P(gid, \bar{x}, fk_c, geom) \wedge gid = q_{int}(gid) \wedge \mathsf{Contains}(w, geom)$
q_{15}	$q_{15}(gid, geom) : \exists \bar{x} fk_s\ C(gid, \bar{x}, fk_s, geom) \wedge \mathsf{Overlaps}(w, geom)$
$q_{15_{int}}$	$q_{15_{int}}(gid, \bar{x}, fk_s, geom) : \exists \bar{x} \bar{y}\ fk_s\ gid'\ geom\ geom'\ C(gid, \bar{x}, fk_s, geom) \wedge S(gid', \bar{y}, geom') \wedge$ $fk_s = gid' \wedge \{\mathsf{Overlaps}(w, geom') \vee \mathsf{Within}(w, geom')\}$
q'_{15}	$q'_{15}(gid, geom) : \exists \bar{x} fk_s\ C(gid, \bar{x}, fk_s, geom) \wedge gid = q_{int}(gid) \wedge \mathsf{Overlaps}(w, geom)$
q_{16}	$q_{16}(gid, geom) : \exists \bar{x} fk_c P(gid, \bar{x}, fk_c, geom) \wedge \mathsf{Overlaps}(w, geom)$
$q_{16_{int}}$	$q_{16_{int}}(gid, \bar{x}, fk_c, geom) : \exists \bar{x} \bar{y}\ fk_c\ gid'\ geom\ geom'\ P(gid, \bar{x}, fk_c, geom) \wedge C(gid', \bar{y}, geom') \wedge$ $fk_c = gid' \wedge \{\mathsf{Overlaps}(w, geom') \vee \mathsf{Within}(w, geom')\}$
q'_{16}	$q'_{16}(gid, geom) : \exists \bar{x} fk_c\ P(gid, \bar{x}, fk_c, geom) \wedge gid = q_{int}(gid) \wedge \mathsf{Overlaps}(w, geom)$
q_{17}	$q_{17}(gid, geom) : \exists \bar{x} fk_s\ C(gid, \bar{x}, fk_s, geom) \wedge \mathsf{Within}(w, geom)$
$q_{17_{int}}$	$q_{17_{int}}(gid, \bar{x}, fk_s, geom) : \exists \bar{x} \bar{y}\ fk_s\ gid'\ geom\ geom'\ C(gid, \bar{x}, fk_s, geom) \wedge S(gid', \bar{y}, geom') \wedge$ $fk_s = gid' \wedge \mathsf{Within}(w, geom')$
q'_{17}	$q'_{17}(gid, geom) : \exists \bar{x} fk_s\ C(gid, \bar{x}, fk_s, geom) \wedge gid = q_{int}(gid) \wedge \mathsf{Within}(w, geom)$
q_{18}	$q_{18}(gid, geom) : \exists \bar{x} fk_c P(gid, \bar{x}, fk_c, geom) \wedge \mathsf{Within}(w, geom)$
$q_{18_{int}}$	$q_{18_{int}}(gid, \bar{x}, fk_c, geom) : \exists \bar{x} \bar{y}\ fk_c\ gid'\ geom\ geom'\ P(gid, \bar{x}, fk_c, geom) \wedge C(gid', \bar{y}, geom') \wedge$ $fk_c = gid' \wedge \mathsf{Within}(w, geom')$
q'_{18}	$q'_{18}(gid, geom) : \exists \bar{x} fk_c\ P(gid, \bar{x}, fk_c, geom) \wedge gid = q_{int}(gid) \wedge \mathsf{Within}(w, geom)$

Cuadro 6.6: Consultas generadas

	$w1$		$w2$		$w3$		$w4$		$w5$	
RQ13	q	q^o	q	q^o	q	q^o	q	q^o	q	q^o
sin índice	14.3	8.4	20.5	13.1	67.2	85.4	117.7	148.6	176.6	220.5
con índice	0.4	8.0	3.6	13.2	53.6	82.9	105.0	149.2	164.2	218.9

El formato de las tablas será el mismo para todas las consultas de rango siguientes.

	$w1$	$w2$	$w3$	$w4$	$w5$
RQ13a	q_{int}	q_{int}	q_{int}	q_{int}	q_{int}
sin índice	7.5	8.7	30.2	43.8	54.6
con índice	7.9	9.3	29.7	44.5	55.2

	$w1$	$w2$	$w3$	$w4$	$w5$
RQ13b	q'	q'	q'	q'	q'
sin índice	0.9	4.4	55.2	104.8	165.9
con índice	0.1	3.9	53.2	105.3	163.7

Cuadro 6.7: Costo de procesamiento RQ13

	$w1$		$w2$		$w3$		$w4$		$w5$	
RQ14	q	q^o	q	q^o	q	q^o	q	q^o	q	q^o
sin índice	51.1	44.8	59.4	66.0	152.2	258.1	246.0	437.6	351.0	637.8
con índice	1.1	30.9	10.2	55.6	105.5	244.2	204.5	422.9	313.8	630.1

	$w1$	$w2$	$w3$	$w4$	$w5$
RQ14a	q_{int}	q_{int}	q_{int}	q_{int}	q_{int}
sin índice	17.5	30.2	125.9	208.5	297.9
con índice	2.8	15.4	112.0	197.2	290.0

	$w1$	$w2$	$w3$	$w4$	$w5$
RQ14b	q'	q'	q'	q'	q'
sin índice	27.3	35.8	132.2	229.1	339.9
con índice	28.1	40.2	132.2	225.7	340.1

Cuadro 6.8: Costo de procesamiento RQ14

	$w1$		$w2$		$w3$		$w4$		$w5$	
RQ15	q	q^o	q	q^o	q	q^o	q	q^o	q	q^o
sin índice	17.3	10.4	30.2	24.4	151.6	148.8	266.4	235.1	404.0	309.2
con índice	2.3	10.3	15.2	25.3	137.2	148.9	251.8	233.2	382.9	310.5

	$w1$	$w2$	$w3$	$w4$	$w5$
RQ15a	q_{int}	q_{int}	q_{int}	q_{int}	q_{int}
sin índice	7.5	8.5	28.2	42.4	55.8
con índice	7.1	8.5	28.6	43.5	56.3

	$w1$	$w2$	$w3$	$w4$	$w5$
RQ15b	q'	q'	q'	q'	q'
sin índice	2.9	15.9	120.6	192.7	253.4
con índice	3.2	16.8	120.3	189.7	254.2

Cuadro 6.9: Costo de procesamiento RQ15

	$w1$		$w2$		$w3$		$w4$		$w5$	
RQ16	q	q^o	q	q^o	q	q^o	q	q^o	q	q^o
sin índice	43.8	43.9	73.0	76.4	329.0	246.7	602.5	395.1	894.2	538.2
con índice	1.0	28.0	27.1	54.7	284.2	225.9	549.7	371.2	837.4	522.2

	$w1$	$w2$	$w3$	$w4$	$w5$
RQ16a	q_{int}	q_{int}	q_{int}	q_{int}	q_{int}
sin índice	17.3	30.7	154.6	272.6	403.8
con índice	2.9	15.9	138.7	257.7	395.2

	$w1$	$w2$	$w3$	$w4$	$w5$
RQ16b	q'	q'	q'	q'	q'
sin índice	26.6	45.7	92.1	122.5	134.4
con índice	25.1	38.8	87.2	114.5	127.0

Cuadro 6.10: Costo de procesamiento RQ16

	$w1$		$w2$		$w3$		$w4$		$w5$	
RQ17	q	q^o	q	q^o	q	q^o	q	q^o	q	q^o
sin índice	15.6	6.9	14.9	4.0	18.5	1.8	21.1	1.7	23.8	2.0
con índice	1.2	6.9	1.2	4.0	4.2	1.6	7.6	1.5	10.6	2.0

	$w1$	$w2$	$w3$	$w4$	$w5$
RQ17a	q_{int}	q_{int}	q_{int}	q_{int}	q_{int}
sin índice	5.2	2.9	1.4	2.2	1.6
con índice	5.2	3.0	1.5	1.4	1.7

	$w1$	$w2$	$w3$	$w4$	$w5$
RQ17b	q'	q'	q'	q'	q'
sin índice	1.7	1.1	0.4	0.5	0.4
con índice	1.7	1.0	0.1	0.1	0.3

Cuadro 6.11: Costo de procesamiento RQ17

	$w1$		$w2$		$w3$		$w4$		$w5$	
RQ18	q	q^o	q	q^o	q	q^o	q	q^o	q	q^o
sin índice	49.6	24.3	48.1	16.7	49.4	18.2	52.7	20.8	60.9	23.9
con índice	0.7	10.6	1.3	1.7	6.1	4.6	11.0	7.3	16.7	10.8

	$w1$	$w2$	$w3$	$w4$	$w5$
RQ18a	q_{int}	q_{int}	q_{int}	q_{int}	q_{int}
sin índice	15.7	15.4	18.0	20.7	23.8
con índice	0.7	1.3	3.7	7.1	10.7

	$w1$	$w2$	$w3$	$w4$	$w5$
RQ18b	q'	q'	q'	q'	q'
sin índice	8.6	1.3	0.2	0.1	0.1
con índice	9.9	0.4	0.9	0.2	0.1

Cuadro 6.12: Costo de procesamiento RQ18

6.3.1. Análisis de resultados

En las consultas de rango, los resultados arrojados por los experimentos nos muestran que al generarse una consulta intermedia que incluya la condición de búsqueda $\mathsf{Contains}(w, geom)$, el tiempo total que toma la ejecución de la consulta optimizada es mayor al tiempo que toma la consulta original. Esto se condice con lo explicado anteriormente en la sección de los índices espaciales, donde se había señalado que para consultas de este tipo era muy probable que la optimización no fuera mejor que la consulta original. Por otro lado, en las otras consultas de rango sí se logró mejorar los tiempos de ejecución, los resultados muestran que el mejor caso para utilizar nuestra optimización es cuando se cuentan con índices espaciales y el parámetro de búsqueda w no es tan pequeño.

Gracias a los experimentos se puede concluir que los siguientes pares de relaciones generan una relación T_c (parte de la consulta intermedia) que logrará una optimización mejor a la utilizada por la optimización por índices:

$\mathsf{Overlaps} \otimes \mathsf{Within}$
$\mathsf{Equal} \otimes \mathsf{Within}$
$\mathsf{Touches} \otimes \mathsf{Within}$
$\mathsf{Within} \otimes \mathsf{Within}$
$\mathsf{Equal} \otimes \mathsf{Equal}$
$\mathsf{Equal} \otimes \mathsf{Overlaps}$
$\mathsf{Equal} \otimes \mathsf{Touches}$
$\mathsf{Equal} \otimes \mathsf{Within}$

Por lo tanto, nuestra estrategia de optimización se aplicará solo cuando se generen esas composiciones entre la relación proveniente de la consulta y la relación proveniente de la restricción de integridad.

En las consultas de join, los resultados nos indican que cuando fue posible reemplazar en la consulta la condición de búsqueda espacial por una condición de búsqueda sobre atributos temáticos, el tiempo de ejecución de la consulta generada por nuestro algoritmo fue siempre mejor que la consulta original. Esto nos reafirma que una consulta con una condición de búsqueda que involucra solo atributos temáticos es mucho más rápida que otra que contenga una condición de búsqueda sobre atributos espaciales.

Existen casos en los cuales no es posible eliminar por completo la condición de

búsqueda espacial de la consulta, esto debido a las características de las restricciones de integridad, las cuales no aseguran 100 % el tipo de relación topológica que existe entre 2 pares de tuplas. Sin embargo, se los experimentos mostraron que siempre la consulta generada será mas rápida que la consulta original, en la consultas sacada del benchmark ocurre esto.

En general, ambas optimizaciones logran el objetivo planteado de mejorar el tiempo de ejecución de la consulta, además las estrategias pueden ser utilizadas en bases de datos con y sin índices espaciales, ya que de todas formas se logran mejores rendimientos en los procesos.

Capítulo 7

Conclusión

En este trabajo se diseñaron estrategias para la optimización semántica de consultas espaciales de join y de rango, utilizando las restricciones de integridad espaciales asociadas a una base de datos. Se generaron e implementaron algoritmos para dichas estrategias. Los experimentos realizados demostraron que las estrategias propuestas efectivamente lograron mejorar el tiempo de ejecución de las consultas, en algunos casos de forma bastante significativa, lo cual es de gran utilidad a la hora de manejar grandes volúmenes de datos. En particular todas las consultas extraídas del benchmark utilizado fueron optimizadas en un alto porcentaje. Queda claro además, que este proceso de optimización es imposible de llevar a cabo si la base de datos en la cual se trabajará no cuenta con restricciones de integridad que relacionen la semántica de los datos con relaciones espaciales.

En general, se logra una mejor optimización en las consultas de Join debido a que en algunas de ellas se logra eliminar por completo la condición de búsqueda que involucra atributos espaciales. Por otro lado, en las consultas de Rango eso no es posible, de hecho se añade una condición espacial más. Esto demuestra lo complicado que es para el sistema trabajar con datos espaciales y que se requieren técnicas sofisticadas de optimización de consulta.

Debido a lo anterior, es muy importante saber escoger de buena forma cuándo se debe y cuándo no se debe aplicar el proceso de optimización sobre las consultas de rango. Esto depende de los tamaños de las diferentes tablas involucradas y de las relaciones topológicas presentes en las consultas y en las restricciones.

Trabajo Futuro

En este trabajo se propuso un algoritmo de optimización para las consultas utilizando restricciones de integridad, las cuales tenían condiciones sobre atributos temáticos. Estas condiciones involucraban la igualdad o desigualdad de solo un atributo por lo que una primera extensión a este trabajo es considerar condiciones que involucren más de un atributo, como sería el caso de una clave foránea compuesta de más de un atributo atómico. Para el algoritmo actual y considerando más de un atributo en la condición temática, no se puede deducir la relación topológica para todo par de tuplas de la base de datos.

Además, sería importante incluir al proceso de optimización los casos en los cuales para algunas tuplas el atributo perteneciente a la condición de búsqueda de la restricción es nulo, por ejemplo, si se tiene un atributo x de algún predicado, ninguna de las siguiente condiciones sería cierta: (i) $x = y$, (ii) $x \neq y$, lo cual llevaría a una falta de conocimiento de la relación topológica que involucra esa tupla.

Por lo tanto, como trabajo futuro queda la inclusión de los casos mencionados anteriormente y además la implementación de los algoritmos de tal forma que funcionen en forma integrada con los gestores de base de datos como PostGIS.

Bibliografía

[1] Lucian Popa Alin Deutsch and Val Tannen. *Query Reformulation with Constraints*. 2005.

[2] Ştefan Olaru Anda Velicanu. *Optimizing Spatial Databases*. Informatica Economică vol. 14, no. 2, 2010.

[3] Loreto Bravo and M. Andrea Rodríguez. *Formalization and Reasoning about Spatial Semantic Integrity Constraints*. Data Knowl. Eng., 72:63-82, 2012.

[4] Ludascher Deutsch and Nash. *Rewriting queries using views with access patterns under integrity constraints.* Theor. Comput. Sci., 371(3):200-226, 2007.

[5] P. Di Felice E. Clementini and P van Oosterom. A small set of formal topological relationships suitable for end-user interaction. In *3rd Int. Symp. on advances in spatial databases*, pages 277–295, 1993.

[6] M. Egenhofer and J. Herring. Categorizing binary topological relations between regions, lines, and points in geographic databases. Technical report, Tech. Report Dept. of Surveying Engineering, Univ. of Maine, Orono, 1991.

[7] Max J. Egenhofer and Robert D. Franzosa. Point set topological relations. *International Journal of Geographical Information Systems*, 5:161–174, 1991.

[8] D.M. Gavrila. *R-tree Index Optimization*. Advances In GIS Research II: Proceedings of the Sixth International Symposium on Spatial Data Handling, pp.771–791, 1994.

[9] A. Guttman. R-trees, a dynamic index structure for spatial searching. *SIGMOD '84 Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, 14:47–57, 1984.

[10] J.M. Hellerstein. *Optimization Techniques For Queries with Expensive Predicates*. ACM Transactions on Database Systems, 23(2), 1998.

[11] Majid Khan and M. N. A. Khan. Exploring query optimization techniques in relational databases. *SIGMOD '84 Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, 6, 2013.

[12] Michael J. Maher and Junhu Wangh. *Optimizing queries in extended relational databases*. In DEXA, volume 1873 of Lecture Notes in Computer Science, pages 386-396, 2000.

[13] Regina O. Obe and Leo S. Hsu. *PostGIS in Action*. Manning Publications, 2011.

[14] OpenGis. Opengis simple features specification for sql. Technical report, Open GIS Consortium, 1999.

[15] David A. Randell, Zhan Cui, and Anthony G. Cohn. A spatial logic based on regions and connection. In *KR'92*, pages 165–176, 1992.

[16] Suprio Ray, Bogdan Simion, and Angela Demke Brown. *Jackpine: A benchmark to evaluate spatial database performance.* Proceedings of the 27th International Conference on Data Engineering, ICDE 2011, April 11-16, 2011, Hannover, Germany, 2011.

[17] O. Stock. *Spatial and Temporal Reasoning*. Kluwer Academic Publishers, 1997.

[18] Kyuseok Shim Surajit Chaudhuri. *Including Group-By in Query Optimization*. the 20th International Conference on VLDB, Santiago, Chile, 1994.

[19] M. Worboys. A geometric model for planar geographical objects. *International Journal of Geographical Information Systems*, 6(5):353–372, 1992.