

**LENGUAJE VISUAL DE CONSULTA BASADO
EN TRANSFORMACIÓN DE GRAFOS**
Aplicación en el dominio médico

MARÍA CONSTANZA PABÓN BURBANO



UNIVERSIDAD DEL VALLE
ESCUELA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
2016

**LENGUAJE VISUAL DE CONSULTA BASADO
EN TRANSFORMACIÓN DE GRAFOS**
Aplicación en el dominio médico

MARÍA CONSTANZA PABÓN BURBANO

Trabajo de tesis para optar al título de
Doctor en Ingeniería

Directoras

Dra. Martha Millán

Escuela de Ingeniería de Sistemas y Computación
Universidad del Valle

Dra. Claudia Roncancio

Laboratoire d'Informatique de Grenoble
Grenoble INP

Asesor

Dr. Cesar Collazos

Departamento de Sistemas
Universidad del Cauca

UNIVERSIDAD DEL VALLE
ESCUELA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
2016

A Wilmar, Oscar y Diana

Agradecimientos

Mis más profundos agradecimientos a los Doctores Martha Millán, Claudia Roncancio y César Collazos por sus enseñanzas, consejos y apoyo durante el desarrollo de este proyecto. A la Doctora Millán, quien continuó, como profesora *ad-honorem*, dirigiendo mi trabajo, impulsada por su motivación para educar. A la Doctora Roncancio y su familia, por la cálida acogida y gran apoyo que me brindaron durante mi estadía en Francia.

Agradezco a los Doctores María del Pilar Villamil, profesora de la Universidad de los Andes, Toni Granollers, profesor de la Universidad de Lleida, y Cristian Alexandru Rusu, profesor de la Pontificia Universidad Católica de Valparaíso, por aceptar formar parte del jurado de esta tesis y por sus valiosas y detalladas evaluaciones.

A los doctores en medicina y profesores del área de salud que aportaron sus conocimientos y experiencia en diversas actividades realizadas durante el desarrollo de esta tesis: Dr. Villamizar, Dra. Soto, Dra. Hurtado, Dra. Borrero, Dra. Jimenez, Dr. Gómez, Dr. Holguín, Prof. Rivera y Prof. Aristizabal de la Facultad de Ciencias de la Salud de la Pontificia Universidad Javeriana - Seccional Cali; Dr. Molina, Dr. Velasco, Dra. Salazar y Dra. Pustoyrh de la Facultad de Salud de la Universidad del Valle; Dr. Mejía de la Clínica Farallones. A los estudiantes de carreras del área de salud que amablemente me ofrecieron su tiempo para participar en las pruebas de usabilidad.

A los profesores de la Escuela de Ingeniería de Sistemas y Computación de la Universidad del Valle por sus enseñanzas, en especial a la Dra. Patricia Trujillo por sus valiosos consejos y por ser el nexo para establecer el contacto con personas de la facultad de salud de la Universidad del Valle, quienes apoyaron de diversas formas el desarrollo de esta tesis. A los profesores del Departamento de Electrónica y Ciencias de la Computación de la Pontificia Universidad Javeriana quienes han compartido generosamente sus experiencias y su tiempo. A los miembros del Laboratoire d’Informatique de Grenoble por su acogida y apoyo a este trabajo. A los Doctores Oscar Corcho y Andrés García, y los Ingenieros Oswaldo Solarte y Oscar Ceballos por sus comentarios y consejos.

A la Pontificia Universidad Javeriana que brindó las condiciones necesarias para llevar a cabo mis estudios.

A Wilmar por sus consejos, por revisar mis documentos y su apoyo en los momentos

difíciles; a Oscar y Diana porque alegran mi vida y me motivan a prosperar. A la Familia Vargas Preciado por los innumerables días en que se hicieron cargo de nuestras responsabilidades en casa. A mi Familia, Pabón Burbano, por sus enseñanzas y apoyo.

Resumen

Ofrecer a los usuarios finales de un sistema de información herramientas que les faciliten acceder y consultar los datos del sistema es un reto que ha sido y que se sigue enfrentando desde diversas perspectivas. Los usuarios expertos en un dominio de aplicación pueden aportar grandes beneficios a las organizaciones si tienen mecanismos que les permitan aprovechar la información que reside en los sistemas transaccionales para el desarrollo de sus labores.

Uno de los mecanismos que se han propuesto para brindar a los usuarios finales la posibilidad de acceder a estos datos son los lenguajes visuales de consulta. En particular, las consultas sobre modelos de grafos han cobrado relevancia en los últimos años debido a su aplicación en áreas como el análisis de datos biológicos, las redes sociales y la *web* semántica. El desarrollo de sistemas visuales de consulta sobre grafos de datos ha seguido dos vertientes. La primera, se orienta hacia las herramientas de exploración y análisis de grafos haciendo énfasis en la facilidad de uso, a pesar de su limitación en términos de expresividad de consulta. La segunda, centrada en interfaces gráficas para lenguajes de consulta basados en texto, ofrece mayor expresividad y traslada a una notación visual cada cláusula del lenguaje, llevando a la notación gráfica la complejidad de formular consultas que generalmente tienen los lenguajes basados en texto.

En esta tesis se propone un lenguaje visual de consulta sobre un modelo de grafos, enfocado en el usuario final, que ofrece mayor expresividad que las herramientas de exploración de grafos sin trasladar a una notación visual los elementos de un lenguaje basado en texto. Se exploran los beneficios del uso de un modelo de grafos simple, en el cual se diferencia el esquema y la instancia, de manera que el esquema representa el modelo conceptual de los datos y su representación gráfica soporta la interacción con el usuario final. El lenguaje facilita la formulación de consultas *ad hoc* (no conocidas con anticipación) que pueden ser complejas, en el sentido que pueden incluir un patrón de filtro formado por una porción del grafo con múltiples nodos y arcos, en los cuales el usuario especifica condiciones de filtro sobre uno o varios nodos y, de manera guiada, establece combinaciones que generan expresiones de conjunción y disyunción de las condiciones. Se propone un conjunto de operadores definidos bajo la estrategia de transformar los grafos esquema e instancia para reducirlos hasta obtener el conjunto de datos que el usuario requiere como resultado de una consulta. La semántica de los

operadores incluye el manejo de datos incompletos, teniendo en cuenta que esta es una característica común en el tipo de sistemas de información hacia los cuales se enfoca este trabajo.

Durante el desarrollo del lenguaje propuesto se aplicaron algunas técnicas de diseño centrado en el usuario, enfocadas a satisfacer necesidades de usuarios en el dominio médico, particularmente las relacionadas con el acceso a los datos de las historias clínicas. Desde las primeras etapas de diseño y desarrollo se realizaron pruebas de usabilidad para identificar las necesidades de los usuarios y validar las decisiones de diseño del lenguaje. Finalmente, se realizó una prueba comparativa entre el prototipo funcional del lenguaje propuesto y una interfaz gráfica para SPARQL. Los resultados que se obtuvieron en esta prueba permiten afirmar que el lenguaje propuesto le facilita al usuario la formulación de consultas complejas.

Los operadores se implementaron con recorridos de caminos usando un motor de grafos. Las pruebas de ejecución mostraron que esta es una opción viable, teniendo en cuenta los tiempos de ejecución, en comparación con la ejecución de las mismas consultas sobre un motor de tripletes y formuladas en SPARQL.

Abstract

Providing tools that ease the access and query of data to information system's end-users is a challenge that has been and continues to be faced with different approaches. End-users that have expertise in an application domain could bring significant benefits to organisations through the use of mechanisms that let them take advantage of the information collected by transactional information systems used in their work.

Visual Query Languages (VQLs) is one of the mechanisms that have been proposed to provide end user's access to that data. In particular, visual query languages and systems for graph data models have gained importance in recent years due to its application in several areas, such as biological data analysis, social networks and semantic web. Visual query languages for graph data can be broadly classified into two groups: On the one hand VQLs with a focus on graph analysis and exploration that provide easy access by allowing direct manipulation of the graph, but with limited query expressiveness. On the other hand, textual query language based VQLs that define a visual notation for the elements of the textual language. Therefore, they have greater expressiveness than the exploration systems but, at the same time, they bring to the visual language the complexity resulting from formulating queries through a textual query language.

This thesis proposes a visual query language for a graph database model. This proposal focuses on end-users needs and offers greater expressiveness than exploration systems without bringing to a visual notation the elements of textual query languages. It explores the benefits of using a simple graph data model which separates the schema and instance graph definitions. In this way, the schema graph is used to depict a conceptual data model, and its graphical representation supports the interaction with end-users.

The proposed language facilitates the formulation of ad hoc queries (queries not known in advance) that could be complex, in the sense that they could include a pattern to filter data consisting of multiple nodes and edges. Users could specify filter conditions on several nodes and, with some guidance, express if they should be applied with a conjunction, a disjunction, or a combination of them. The query formulation follows a strategy of transforming the schema and instance graph to reduce them until the user obtains the set of data she requires. A set of operators that provides these transformations are defined. The operator's semantic include the management

of incomplete data, having into account that this is a characteristic of the information systems in which this work focuses.

During the development of the proposed language some techniques of user-centered design were employed, with the aim to satisfy medical domain users' query needs, particularly the ones related to the access to clinical data. Several usability test were applied from the first development stages to identify the users' requirements and to confirm the design decisions. Finally, a comparative study including the functional prototype of the proposed language and a graphical interface for SPARQL showed that the proposed language facilitates to the user the formulation of complex queries.

The implementation of the language operators over a graph database engine used path traversal. An execution comparison study showed that the use of path traversals is a viable option, having into account that the execution time of the same queries formulated on SPARQL and executed in a triple store was larger than the execution time of the proposed operators.

Contenido

Contenido	ix
Lista de Figuras	xii
Lista de Tablas	xiv
Lista de Anexos	xv
1 Introducción	1
1.1 Consulta de datos para usuarios finales	1
1.2 Preguntas y Objetivos de la Investigación	4
1.2.1 Preguntas de Investigación	4
1.2.2 Objetivos	4
1.3 Solución propuesta: Un lenguaje visual de consulta sobre grafos	5
1.4 Contribución	7
1.5 Organización del documento	9
I Preliminares y Estado del Arte	11
2 Grafos de Datos	13
2.1 Modelos de datos	13
2.1.1 Modelos de datos conceptuales	13
2.1.2 Modelos de datos basados en grafos	15
2.2 Lenguajes de consulta en grafos de datos	20
2.3 Síntesis sobre modelos y lenguajes de consulta basados en grafos	29
3 Sistemas de Consulta sobre Grafos de Datos Orientados al Usuario Final	31
3.1 Usabilidad	31
3.1.1 Usabilidad en <i>software</i>	32
3.1.2 Diseño centrado en el usuario	33
3.1.3 Pruebas de usabilidad	34

3.1.4	Usabilidad en los sistemas de bases de datos	36
3.2	Lenguajes visuales de consulta sobre grafos de datos	39
3.2.1	Sistemas para la exploración y el análisis de datos representados en grafos	40
3.2.2	Herramientas enfocadas en el lenguaje de consulta	44
3.2.3	Interfaces gráficas para lenguajes de consulta	50
3.2.4	Discusión	55
II	El Lenguaje Propuesto	65
4	GraphTQL	67
4.1	Ejemplo de referencia	68
4.2	Mecanismo de interacción	68
4.3	Descripción de los operadores de transformación de grafos	74
4.4	Descripción del diálogo de clarificación de filtros	79
4.5	Características de diseño de GraphTQL	79
4.6	Discusión	88
5	Operadores de Transformación de Grafos de Datos	93
5.1	Modelo de datos	93
5.2	Definición de los operadores de transformación de grafos	98
5.2.1	<i>Select a Portion</i>	99
5.2.2	<i>Filter</i>	101
5.2.3	<i>Filter+Additional Data</i>	116
5.2.4	<i>Put Two Objects Closer</i>	118
5.3	Definición de los operadores de nivel lógico	122
5.3.1	<i>Subgraph Extraction</i>	122
5.3.2	<i>Logic Level Filter</i>	123
5.3.3	<i>Selective Union</i>	125
5.3.4	<i>Path Contraction</i>	126
5.3.5	<i>Union</i> (Unión de grafos)	127
5.4	Discusión	127
6	Enfoque en el Diseño Centrado en el Usuario	131
6.1	Entrevista preliminar	133
6.2	Prueba exploratoria	134
6.3	Pruebas de evaluación	136
6.4	Prueba comparativa	139
6.5	Discusión	150

7 Implementación de GraphTQL y Evaluación de los Operadores de Nivel Lógico	153
7.1 Estructura Funcional del Sistema	153
7.2 Implementación de los operadores	156
7.2.1 Algoritmos	157
7.2.2 Evaluación	162
7.3 Arquitectura de Integración de Datos	171
7.3.1 Mapeos entre el Modelo Relacional y GDM	176
7.4 Síntesis	178
8 Conclusiones y Trabajo Futuro	179
8.1 Contribuciones	179
8.2 Conclusiones	181
8.3 Trabajo futuro	183
8.3.1 El lenguaje visual de consulta	183
8.3.2 La usabilidad del lenguaje	184
8.3.3 Aspectos relacionados con la implementación	185
Publicaciones	187
Referencias	189

Lista de Figuras

2.1 Consulta en SPARQL	23
2.2 BiQL estructuras y consulta	24
2.3 GraphDB modelo y consulta	25
2.4 Consulta en StruQL	26
2.5 GraphQL	28
2.6 Consulta en Cypher	29
3.1 Ejemplo de exploración y filtro	41
3.2 Filtro en CGV	42
3.3 Ejemplos de Explorator y Glyphlink	44
3.4 Modelo y consulta en GraphLog	45
3.5 Consulta en GOOD	45
3.6 Consulta en MashQL	46
3.7 Consulta Recursiva	47
3.8 Ejemplo de filtro	48
3.9 Consulta en OntoVQL	48
3.10 Ejemplo de consulta	49
3.11 Ejemplo de filtro	50
3.12 Consulta en QGraph	50
3.13 Consulta con filtro en Gruff	51
3.14 Consulta con filtro en NITELIGHT	52
3.15 Consulta en SPARQLFilterFlow	53
3.16 Navegación	54
3.17 Consulta con opcional en GQL	54
3.18 Consulta en RDF-GL	55
4.1 Ejemplo de referencia: Datos clínicos (grafo esquema).	69
4.2 Ejemplo de referencia: Datos clínicos (grafo instancia).	70
4.3 Interfaz de GraphTQL.	71
4.4 Diálogo de clarificación de caminos.	73
4.5 Selección de la función a cumplir en un nodo de valor básico.	74
4.6 Operador <i>Select a Portion</i>	75

4.7	Operador <i>Filter</i>	77
4.8	Operador <i>Put Two Objects Closer</i>	78
4.9	Diálogo de clarificación de filtros.	80
4.10	Consultas formuladas en Gruff	85
4.11	Consultas formuladas en Gruff	86
4.12	Operador <i>Value filter</i>	90
4.13	Aplicación de <i>Value filters</i> consecutivos.	91
5.1	Ejemplo de referencia - Esquema de Datos	94
5.2	Ejemplo de una instancia de datos clínicos	96
5.3	Operador <i>Select a Portion</i>	100
5.4	<i>Select a Portion</i> representado en Gruff.	101
5.5	Operador <i>Filter</i>	102
5.6	Ejemplo de consulta.	104
5.7	Clarificación de un nodo de bifurcación final.	105
5.8	Diálogo para un nodo de bifurcación final.	106
5.9	Diálogo para un nodo de bifurcación intermedio.	106
5.10	Diálogo en el nodo de bifurcación <i>Patient</i>	107
5.11	Clarificación de una consulta que incluye un ciclo y una condición.	109
5.12	Diálogo cuando la consulta incluye un ciclo y una condición.	110
5.13	Consulta que incluye componentes opcionales.	114
5.14	Ejemplo de <i>Filter</i> expresado en SPARQL.	116
5.15	Ejemplo de <i>Filter+Additional Data</i> expresado en SPARQL.	119
5.16	Operador <i>Put Two Objects Closer</i>	120
5.17	Patrón de filtro con ciclo y varias condiciones.	129
5.18	Condiciones entre tipos de nodo.	130
7.1	Estructura Funcional del Sistema	154
7.2	Tiempos promedio de ejecución en frío.	167
7.3	Tiempos promedio de ejecución de las repeticiones.	168
7.4	Arquitectura de integración	171
7.5	Arquitectura de integración basada en mediación y modelos de grafos .	172
7.6	Modelo global	173
7.7	Modelo relacional de la fuente S1	177
7.8	Grafo local de la fuente S1	178

Lista de Tablas

3.1	Características encontradas en los Lenguajes Visuales de Consulta	56
3.2	Características deseables, incluidas en los Lenguajes Visuales de Consulta.	63
5.1	Operadores y valores de las expresiones de filtro básicas.	124
5.2	Evaluación de las condiciones de filtro.	124
6.1	Pruebas de usabilidad en el desarrollo del proyecto.	132
6.2	Segunda prueba de evaluación - Resultados por participante.	137
6.3	Segunda prueba de evaluación - Resultados promedio.	137
6.4	Clasificación de las consultas según su complejidad.	141
6.5	Consultas para los ejemplos.	142
6.6	Consultas para las actividades.	143
6.7	Prueba comparativa - Medidas de rendimiento por participante.	148
6.8	Prueba comparativa - Medidas de rendimiento promedio.	149
6.9	Prueba comparativa - Medida de satisfacción por participante.	150
6.10	Prueba Comparativa - Medida de satisfacción promedio.	150
7.1	Consultas usadas en la prueba de ejecución.	164
7.2	Resultados de la ejecución en frío.	169
7.3	Resultados de la repetición de las ejecuciones.	170

Lista de Anexos

Anexo A. Resumen entrevistas preliminares	207
Anexo B. Escenarios y ejemplos de consultas	213
Anexo C. Plan de la prueba exploratoria	217
Anexo D. Resumen de la aplicación de la prueba exploratoria	231
Anexo E. Diseño de la segunda prueba de evaluación	235
Anexo F. Resumen de la aplicación de la segunda prueba de evaluación	241
Anexo G. Diseño de la prueba comparativa	247
Anexo H. Resumen de la prueba comparativa	295
Anexo I. Consultas de la prueba de implementación	305

Capítulo 1

Introducción

1.1 Consulta de datos para usuarios finales

El acelerado desarrollo tecnológico, particularmente de los modelos y mecanismos de almacenamiento y gestión de información, ha llevado a que la mayoría de las empresas, especialmente las grandes y medianas, registren sus datos en múltiples sistemas de información. Como resultado, esas organizaciones, públicas o privadas, almacenan grandes cantidades de datos en sistemas transaccionales. La información disponible crece, día a día, en volumen y complejidad, y se caracteriza por la heterogeneidad de los datos.

Para aprovechar mejor la información recopilada, se requiere que los usuarios, especialmente aquellos que son expertos en un dominio, tengan acceso a los datos para encontrar información relevante. Sin embargo, dada la complejidad de los datos y sus relaciones, usar un lenguaje para formular consultas *ad-hoc* (consultas no definidas previamente) resulta una tarea difícil para los usuarios finales. Por lo tanto, las organizaciones contratan administradores de bases de datos, consultores y expertos en tecnologías como intermediarios entre las bases de datos y los usuarios, ayudándoles a recuperar aquellos datos que son de su interés [97]. Los costos de este servicio exceden con creces los costos de la infraestructura de *hardware* y sistemas de bases de datos. Adicionalmente, hay un costo de oportunidad en el conocimiento que los usuarios finales podrían obtener explorando y analizando la información, que se pierde por la dependencia de los expertos en tecnología. El experto de dominio puede lograr mayor profundidad en el análisis de los datos y mayor alcance en la información obtenida, si explora los datos directamente, mediante una herramienta.

Permitir a los usuarios finales buscar fácilmente datos estructurados es un reto [20, 97, 98]. Una de las razones es la necesidad de conocer la estructura de los datos y la sintaxis y semántica de los lenguajes para formular la consulta. Además, cuando se trata del acceso a los datos de sistemas de información, los usuarios tienen requerimientos de consulta particulares [97]. Ellos requieren realizar consultas con

semántica compleja, con filtros que incluyen varias condiciones unidas por conectivos lógicos. Además, estos usuarios requieren respuestas completas y precisas. Los modelos de consulta de los lenguajes estructurados (ej. SQL, SPARQL) proveen los medios para realizar este tipo de consultas, pero resulta difícil que los usuarios finales los adopten porque para ello se requiere conocer la semántica de las operaciones del lenguaje y la estructura de la base de datos.

Una de las formas mas estudiadas para superar este reto son los Lenguajes Visuales de Consulta, que tienen como propósito ofrecer a usuarios no programadores la posibilidad de expresar consultas sobre bases de datos [36]. Para que los usuarios finales accedan a esos grandes volúmenes de información, es necesario que puedan formular consultas de manera fácil y efectiva, sin necesitar de habilidades de programación [98].

Este trabajo se centra en el desarrollo de un lenguaje visual de consulta sobre un modelo de grafos. Las bases de datos basadas en grafos han sido ampliamente estudiadas en los últimos años debido a las posibilidades de aplicación en diferentes áreas, entre ellas, el análisis de datos biológicos (ej. estudios del genoma y modelos bioquímicos o químicos) [42, 57, 67, 103], las redes sociales [144, 151, 159] y la *web* semántica [7, 9, 153].

En este trabajo se explora el uso de un grafo para representar un modelo conceptual de los datos que sirva como medio de interacción con el usuario final. De acuerdo con Jagadish [97] es inadecuado usar esquemas lógicos como medio de interacción para herramientas de consulta orientadas al usuario final, debido a su bajo nivel de abstracción. De otra parte, los modelos de grafos proporcionan un modelado natural de los datos [7, 187] y por su representación gráfica, permiten representar modelos conceptuales de datos en un diagrama fácil de entender para los usuarios. Una muestra de ello es que la representación gráfica de la mayoría de modelos de datos conceptuales tiene como base un grafo (ej. MER [41], ORM [74]). Por tanto, los modelos de datos basados en grafos son una alternativa interesante para soportar el desarrollo de lenguajes visuales de consulta.

Otra característica de los modelos de datos basados en grafos, relevante para este trabajo, es que se puede definir fácilmente su correspondencia con otros modelos de datos (*mapping*) [168]. Esto permite trasladar la vista de nivel conceptual al modelo en el que están representados los datos, de manera que las consultas se pueden ejecutar y así recuperar los datos solicitados.

Con relación a los lenguajes visuales de consulta (*Visual Query Language*, VQL) y de acuerdo con Catarci [36] son sistemas de consulta de bases de datos que usan una representación visual para describir el dominio de interés y para expresar las solicitudes sobre los datos. La especificación de un lenguaje visual de consulta implica la búsqueda de un equilibrio entre la complejidad y la expresividad del lenguaje, ya que generalmente una suele ir en detrimento de la otra [20, 79]. Como se muestra en la Sección 3.2, en el caso particular de los lenguajes visuales sobre grafos, la tensión entre la expresividad y la facilidad de uso ha desembocado en dos vertientes

principales: de una parte, se encuentran las herramientas de exploración y análisis de datos, y de otra parte, las interfaces gráficas para lenguajes de consulta. Las herramientas de exploración de datos se enfocan en brindar soluciones al problema del despliegue y análisis de grandes cantidades de información. Sin embargo, la expresividad de las consultas en estas herramientas, en particular de los filtros, es limitada. Dado que generalmente los usuarios requieren analizar un subconjunto de los datos almacenados en los sistemas de información, es necesario formular consultas para extraer las porciones de datos con las características requeridas y alimentar con ellas las herramientas de análisis. En cuanto a las interfaces gráficas para lenguajes de consulta, ellas ofrecen la expresividad de un lenguaje de consulta pero generalmente trasladan a una representación visual los elementos y conceptos del lenguaje. Con ello se dificulta, para un usuario final, la formulación de las consultas, ya que se ve obligado a conocer los elementos del lenguaje y la estructura de la base de datos.

El lenguaje que se propone ofrece mayor expresividad que las herramientas de exploración de datos y requiere que el usuario aprenda un menor número de conceptos propios de los lenguajes de consulta. En el desarrollo del lenguaje se aplicaron técnicas de diseño centrado en el usuario [60, 165], para ello se eligió el dominio médico como dominio de aplicación, debido a la trayectoria con proyectos en este dominio del grupo de investigación GEDI de la Universidad del Valle y a que se espera aportar mejores herramientas a la práctica médica de la región.

En el dominio médico grandes cantidades de datos de la historia clínica de los pacientes se registran diariamente en sistemas transaccionales [62]. Facilitar a los profesionales de la salud acceder a estos datos mediante consultas *ad-hoc*, que les permita encontrar, por ejemplo, grupos de pacientes con ciertas características, genera un impacto positivo en actividades como la educación, la investigación, la prevención y el seguimiento de la evolución de los pacientes (Anexo B). Adicionalmente, el lenguaje podría contribuir a mejorar la atención de los pacientes si se tiene en cuenta que se facilita el encontrar información relevante para el caso que está siendo tratado, relacionada, por ejemplo, con diagnósticos, con tratamientos o procedimientos previos o con sus antecedentes familiares. Por otra parte, se puede acceder a datos e información adicional de fuentes externas, como las redes sociales, que resultaría útil para analizar el contexto, los hábitos y actividades de los pacientes, siendo un complemento al perfil clínico que permite mejorar la atención brindada y encontrar otras relaciones en los grupos de pacientes que hacen parte de una investigación médica.

Un estudio [11] sobre los errores que se generan por discrepancias entre el funcionamiento de los sistemas de información de atención al paciente y las exigencias del trabajo en áreas de la salud, muestra que el uso intensivo de formatos en estos sistemas genera un incremento en el tiempo requerido por los profesionales de la salud para escribir o leer la información. Por tanto, la posibilidad de que los profesionales de la salud exploten directamente la información resultante de la práctica médica puede hacer más eficiente la labor de los especialistas, mediante la adopción de herramientas

de consulta para, entre otros, ayudar en el diagnóstico, evaluar la salud de la población, controlar la evolución de los pacientes, analizar la variación de las prácticas médicas y determinar la eficacia de los programas de promoción y prevención. De esta manera, se podría, en alguna medida, impactar positivamente los procesos de diagnóstico, de toma de decisiones médicas y de seguimiento a la evolución del paciente, además de la enseñanza y la investigación en medicina.

1.2 Preguntas y Objetivos de la Investigación

1.2.1 Preguntas de Investigación

Este estudio busca dar respuesta a las siguientes preguntas:

- ¿Cuáles pueden ser las primitivas, la sintaxis y la semántica de un lenguaje orientado al usuario final, para expresar consultas ad-hoc con un vocabulario propio del dominio de aplicación, particularmente, en el dominio médico, en la recuperación de datos clínicos?
- ¿Cuáles operadores requiere el lenguaje para soportar las primitivas que se definen en el mismo?
- ¿Cómo tomar ventaja del conocimiento del usuario experto en el dominio en la especificación del lenguaje, para que las formulación de las consultas sea simple y cercana al usuario?
- ¿Cuál modelo conceptual es adecuado para representar los datos del dominio, mostrar vistas simplificadas del modelo que guíen al usuario en la formulación de la consulta y subyacer al lenguaje de consulta?

1.2.2 Objetivos

Objetivo General

Proponer un lenguaje de recuperación de datos orientado a usuarios finales expertos en un dominio de aplicación; basado en un modelo conceptual; que ofrezca al usuario una forma simple de especificar consultas sin que esto implique conocer la estructura de los datos; permitiéndo definir, a alto nivel y usando el vocabulario propio del dominio, cuáles datos recuperar y cuáles condiciones satisfacer.

Objetivos Específicos

- Identificar las necesidades de consulta en el dominio médico, en particular en el ámbito de los datos clínicos, a través de la aplicación de métodos de indagación y de test.
- Elegir el modelo conceptual de datos subyacente al lenguaje de consulta.

-
- Definir las primitivas, la sintaxis, la semántica y los operadores del lenguaje conceptual, que permitan construir expresiones de consulta cercanas al usuario y con un vocabulario propio del dominio, para recuperar datos clínicos de diversos tipos.
 - Elegir un conjunto de operadores que permitan representar la consulta independientemente de los modelos lógicos y físicos de las fuentes de datos.
 - Evaluar la usabilidad del lenguaje, realizando un test, sobre un prototipo de software, que mida la experiencia y satisfacción de un grupo de usuarios.

1.3 Solución propuesta: Un lenguaje visual de consulta sobre grafos

Esta tesis propone un lenguaje visual de consulta sobre grafos de datos, orientado a usuarios finales. El caso de aplicación de este trabajo es el dominio médico, en particular las consultas sobre datos clínicos.

El lenguaje está diseñado para responder a las necesidades de usuarios finales en dominios de aplicación caracterizados por:

- Un gran volumen de datos que los usuarios generan, y a la vez requieren, en los procesos que ellos realizan durante el desarrollo de su trabajo. La mayor parte son datos estructurados, que en algún momento se pueden complementar con otros no estructurados. Aún así, es posible ofrecer a los usuarios una vista de la organización de los datos en un esquema de nivel conceptual.
- Usuarios expertos en el dominio (i.e. profesionales de ese dominio), que conocen bien los conceptos y procesos del mismo, pero que no conocen los detalles técnicos propios de los sistemas de información. En particular, no conocen los modelos de datos y los lenguajes de consulta de las bases de datos que esos sistemas ofrecen.
- Usuarios finales que requieren consultas *ad-hoc* (no conocidas con anticipación), de mediana complejidad, que implican el uso de filtros compuestos por varias condiciones unidas por conectivos lógicos. Siendo usuarios expertos en el dominio de la aplicación tienen diferentes necesidades de consulta que al ser satisfechas pueden generar valor para la organización.
- Datos que se caracterizan porque:
 - Suelen ser incompletos, algunos atributos de una clase podrían no tener un valor para un objeto de esa clase.
 - Aun en el caso en el cual los datos son no estructurados, es posible tener un conjunto de metadatos básico que permite ligarlos a la estructura de los datos. Por ejemplo, en el dominio médico, los textos que describen hallazgos encontrados en exámenes o imágenes diagnósticas tienen metadatos asociados que permiten identificar el paciente, el tipo de examen realizado, la parte

del cuerpo examinada y la fecha en que se realizó el examen, entre otros. Adicionalmente, es posible asociar con estos datos anotaciones que hagan referencia a palabras clave de esos hallazgos [121].

- Información que puede estar almacenada en varios sistemas, con características heterogéneas, y que se puede enriquecer con datos que provienen de sistemas externos (ej. redes sociales).

Entre otros dominios, el de aplicación de este trabajo, la información clínica de pacientes, cumple con estas características. Los médicos, enfermeros y profesionales de la salud requieren el acceso a la información de los pacientes con múltiples fines: atención, investigación, enseñanza, administración clínica, creación y realización de planes de prevención, alertas epidemiológicas, entre otros.

Dadas las características del dominio y de los datos, un modelo de datos basado en grafos es una opción adecuada para soportar el lenguaje de consulta. De una parte, un modelo de datos de nivel conceptual se puede representar fácilmente mediante un grafo y esto favorece la interacción con los usuarios. De otra parte, los grafos se han utilizado ampliamente en integración de datos de fuentes heterogéneas [13, 130, 172, 181, 182], ya que su flexibilidad facilita mapear con el grafo otros modelos de datos, estructurados o semiestructurados. Por tanto, el modelo de grafos brinda la posibilidad de que el lenguaje propuesto opere sobre una arquitectura de integración de datos basada en mediación, en la que el grafo esquema es el modelo global. El esquema presenta una vista que integra los datos de las fuentes diversas y heterogéneas, y se mapea con los modelos de datos de cada fuente. De esta manera soporta la traducción de las consultas al lenguaje propio de cada fuente, con el fin de recuperar los datos solicitados en cada consulta.

Como se mencionó en la sección anterior, entre los lenguajes visuales de consulta sobre grafos se encuentran, de una parte, las herramientas de exploración y análisis de datos y, de otra parte, las interfaces gráficas para lenguajes de consulta. Las primeras, las herramientas de exploración y análisis de grafos, se enfocan en los mecanismos de visualización, navegación y el cálculo de características propias de la conectividad del grafo (ej. grado de los nodos, centralidad, k-vecinos, *layout*). Estas herramientas permiten la manipulación directa del grafo de datos, lo cual genera problemas cuando el volumen de datos a visualizar es grande, además de que ofrecen menor expresividad, con respecto a las interfaces, en los mecanismos para filtrar datos. Las segundas, las herramientas enfocadas en el lenguaje de consulta, ofrecen mayor expresividad, pero requieren que el usuario conozca la organización de los datos y las operaciones y conceptos propios del lenguaje. El mecanismo de interacción con el grafo no es de manipulación directa, ya que generalmente el usuario debe dibujar los patrones de consulta desde cero, a partir de listas de objetos y propiedades. La realimentación se da para el usuario únicamente después de ejecutar una consulta. Adicionalmente, la mayoría de las herramientas, de ambos grupos, requieren que el usuario haga una

exploración del esquema de datos. Esta exploración se realiza a partir de nodos específicos y por tanto, ésta se afecta cuando la información es incompleta. Por ejemplo, cuando el usuario toma como punto de partida un nodo objeto que no tiene todos los atributos definidos para su clase, es posible que no se percate de que esos atributos están disponibles en otros nodos de la misma clase. Además, solo aquellos sistemas que se basan en ontologías operan sobre una representación de los datos de nivel conceptual.

Por tanto, el problema objeto de estudio de este trabajo se centra en cómo ofrecerle a usuarios finales, en contextos con las características mencionadas, un mecanismo para formular consultas *ad-hoc* sobre un modelo de grafos.

La solución que se propone en este trabajo, es un lenguaje visual de consulta definido sobre un modelo de datos que diferencia los grafos esquema e instancia. Las consultas se formulan a partir de una vista del diagrama del grafo esquema, que representa la organización de los datos a nivel conceptual. Para ello, se aplican sucesivamente operaciones que transforman vistas de los grafos esquema e instancia, de manera que los datos seleccionados se reducen hasta obtener el subconjunto de datos que el usuario necesita recuperar.

Las operaciones permiten seleccionar una porción del grafo, aplicar filtros complejos y crear nuevas relaciones reemplazando caminos que unen los nodos que son instancia de dos clases seleccionadas. Estas relaciones hacen explícita información que está implícita o que está dispersa.

El lenguaje usa el grafo esquema como mecanismo de interacción con el usuario, los parámetros de entrada de las operaciones se especifican mediante la manipulación directa del grafo esquema. En cada operación el usuario recibe realimentación ya que el grafo esquema modificado se despliega, sin requerir que se ejecute la operación.

En el desarrollo de la solución propuesta se incluyeron algunas técnicas de diseño centrado en el usuario [60, 165]: entrevistas preliminares, pruebas exploratorias, pruebas de usabilidad y pruebas comparación de productos. El perfil de los participantes de estas pruebas incluyó profesionales de la salud (médicos, enfermeros, epidemiólogos) y estudiantes de medicina.

1.4 Contribución

Las principales contribuciones de este trabajo son:

Un Lenguaje Visual de Consulta sobre grafos de datos orientado a usuarios finales.

Facilitar a los usuarios finales el acceso a los datos de un sistema de información continúa siendo un tema de investigación y desarrollo. En este sentido, un reto por resolver es la tensión entre la expresividad del lenguaje y su facilidad de uso, ya que una va en detrimento de la otra. Esta tensión, ha generado dos vertientes en el desarrollo de lenguajes visuales de consulta sobre grafos

de datos: de una parte, las herramientas de exploración y análisis, y de otra, las herramientas que se enfocan en el lenguaje de consulta. Las primeras, las herramientas de exploración y análisis de grafos, tienen generalmente limitaciones en la expresividad de las consultas, pero ofrecen manipulación directa del grafo instancia y operaciones para el análisis de propiedades del grafo. Las segundas, las herramientas que se enfocan en el lenguaje de consulta, tienen mayor expresividad pero, usualmente, trasladan a una notación visual los conceptos del lenguaje de consulta, que opera en el nivel lógico. Por tanto, los usuarios necesitan aprender esos conceptos y además, construir desde cero el patrón de consulta a partir de listas de objetos y propiedades.

GraphTQL, el lenguaje visual de consulta propuesto, se ubica en un punto intermedio entre estos extremos, proveyendo mayor expresividad que las herramientas de análisis y exploración, pero conservando algunas de sus características para facilitar al usuario final la formulación de consultas de mediana complejidad. Se consideran consultas de mediana complejidad aquellas que requieren filtros compuestos por varias condiciones unidas por conectivos lógicos o que requieren manejar datos incompletos. GraphTQL ofrece a los usuarios finales un mecanismo para seleccionar datos basado en la manipulación directa, con alto nivel de abstracción, ofreciendo realimentación durante la formulación de la consulta, y reduciendo la cantidad de conceptos propios de los lenguajes de consulta que el usuario debe aprender y aplicar. Además, dado que los usuarios finales no disponen del tiempo ni del conocimiento de la tecnología necesarios para explorar los datos con el fin de conocer el esquema [97], con el lenguaje propuesto el usuario no necesita realizar esta exploración.

Una característica adicional importante, es que GraphTQL fue desarrollado con un enfoque de diseño centrado en el usuario, para lo cual se tomó como dominio de aplicación el dominio médico.

Un conjunto de operadores formalmente definidos que soportan el lenguaje visual de consulta.

La revisión de literatura sobre lenguajes visuales de consulta sobre grafos (Sección 3) muestra que la mayoría se basa en la búsqueda de patrones. Mediante la búsqueda de patrones se recuperan los datos que coinciden de manera exacta con un patrón dado, excluyendo aquellos que tienen datos incompletos. Por tanto, se requiere usar operaciones o cláusulas particulares, como el *optional* y el *union* en SPARQL, para manejar los datos incompletos. Este manejo puede resultar difícil en algunas consultas, por ejemplo cuando se requiere filtrar los datos aplicando una disyunción de condiciones y a la vez manejar datos incompletos. Teniendo en cuenta que una característica de los dominios en los cuales se enfoca este trabajo, y de muchas bases de datos de grafos [187], es que los datos son generalmente incompletos, los operadores de GraphTQL están definidos para operar sobre datos

incompletos. Por tanto el usuario no requiere operaciones adicionales para el manejo de los mismos.

Los operadores de GraphQL permiten manipular los objetos manteniendo sus relaciones y atributos; derivar nuevas relaciones entre objetos que están conectados por caminos no dirigidos; y definir filtros complejos sobre objetos de interés.

Un mecanismo para desambiguar los filtros basado en diálogos de clarificación.

Expresar de manera correcta el patrón de búsqueda y las condiciones sobre los datos puede resultar difícil para un usuario final. Por ejemplo, en SPARQL, expresar correctamente una disyunción de condiciones puede requerir que se creen varias variables para el mismo tipo de objeto o usar de manera correcta la cláusula *optional*. En GraphQL, los diálogos de clarificación de filtros guían al usuario cuando se requiere especificar, en los diferentes objetos involucrados en una consulta, si las condiciones se conectan con una conjunción o con una disyunción.

Un prototipo funcional del lenguaje de consulta cuya usabilidad fue evaluada por usuarios finales.

La revisión de literatura sobre lenguajes visuales de consulta sobre grafos (Sección 3) muestra una aplicación de pruebas de usabilidad relativamente limitada (aproximadamente en la mitad de los trabajos reportados). En la mayoría de estos casos, la prueba se aplica al finalizar el ciclo de desarrollo del lenguaje. En contraste, GraphQL se diseñó y desarrolló con un enfoque de diseño centrado en el usuario, para lo cual se realizaron cinco pruebas de usabilidad en distintas etapas del desarrollo del proyecto. Los resultados de estas pruebas permitieron ratificar algunas decisiones de diseño del lenguaje y cambiar otras.

Una implementación de los operadores usando recorridos de caminos.

Se exploró el uso de recorridos de caminos (*path traversals*) sobre un motor de bases de datos de grafos, y se hizo la comparación de los tiempos de ejecución de esta implementación con los tiempos requeridos para ejecutar consultas equivalentes en SPARQL. En la mayoría de los casos evaluados, la implementación con recorridos de caminos arrojó mejores tiempos de ejecución.

1.5 Organización del documento

Dos campos de la computación tienen incidencia importante en este trabajo. De una parte, los lenguajes de consulta sobre modelos de datos basados en grafos y de otra, la usabilidad en el desarrollo de *software*, particularmente, la usabilidad en los sistemas

de bases de datos. El Capítulo 2 incluye algunos conceptos preliminares sobre modelos conceptuales de datos, la revisión de la literatura sobre modelos de datos basados en grafos (también llamados datos de grafos) y la revisión de la literatura sobre los lenguajes de consulta, con sintaxis en texto, sobre grafos. El Capítulo 3 presenta el marco conceptual sobre usabilidad, diseño centrado en el usuario, usabilidad en los sistemas de bases de datos y la revisión de la literatura sobre lenguajes visuales de consulta para grafos de datos. Este capítulo finaliza con una discusión sobre las características encontradas en estos lenguajes, específicamente las relacionadas con la metáfora visual, el mecanismo de interacción, el uso del esquema de datos, la expresividad, el manejo de datos incompletos y la aplicación de un proceso de diseño centrado en el usuario.

GraphTQL, el lenguaje visual de consulta propuesto, se describe en el Capítulo 4. Esta descripción incluye las características representativas de GraphTQL y su interfaz, ejemplos de formulación y una descripción informal de los operadores del lenguaje. Además se introducen los diálogos de clarificación para la selección de caminos y para la desambiguación de los filtros. Por último, se analizan las limitaciones en la expresividad del prototipo actual, y se proponen algunas formas de extender el lenguaje.

En el Capítulo 5 se describe formalmente GDM [87], el modelo de datos que subyace GraphTQL. Además, se definen los operadores propuestos para el lenguaje.

El capítulo 6 reporta las actividades realizadas dentro del marco del diseño centrado en el usuario y la evaluación de usabilidad del prototipo desarrollado.

El Capítulo 7 describe cómo el lenguaje puede ser usado como modelo global en una arquitectura de integración de datos basada en mediación y el mapeo entre el modelo de grafos del lenguaje y el modelo relacional.

Finalmente, el Capítulo 8 presenta las conclusiones del trabajo y propone algunas líneas de trabajo futuro.

Parte I

Preliminares y Estado del Arte

Capítulo 2

Grafos de Datos

Este capítulo introduce conceptos relacionados con los modelos de datos, en particular modelos de nivel conceptual y modelos de datos basados en grafos. El capítulo está organizado en dos secciones. La primera, incluye una breve descripción de los modelos de datos conceptuales y una revisión de modelos de datos basados en grafos. En la segunda sección se presentan lenguajes de consulta sobre grafos.

2.1 Modelos de datos

Un modelo de datos [178] es un conjunto de herramientas conceptuales para describir las entidades del mundo real y sus relaciones, de manera que se puedan modelar en una base de datos. Para ello, el modelo brinda [46] una estructura para representar los datos, un conjunto de operadores o reglas de inferencia para recuperar datos de esas estructuras y un conjunto de reglas de integridad para definir los estados consistentes de la base de datos.

2.1.1 Modelos de datos conceptuales

Simsion y Witt [179] clasifican los modelos de datos, de acuerdo con su nivel de abstracción, en modelos de datos físicos, lógicos y conceptuales. Los primeros, los modelos de datos físicos, se enfocan en aspectos físicos de almacenamiento y recuperación de los datos. Los segundos, los modelos de datos lógicos, son dependientes del sistema gestor de base de datos que se usa en la implementación de una aplicación. Finalmente, los modelos conceptuales [190] permiten crear una abstracción de la realidad que describe aspectos de un dominio de interés, de forma no ambigua y fácil de entender para los diseñadores y los usuarios de los datos. Esta descripción sigue un principio de conceptualización que la hace independiente del *hardware* y del *software* [164].

De acuerdo con la literatura los modelos conceptuales:

-
- Son representaciones simplificadas de objetos, fenómenos o situaciones reales [26].
 - Son precisos, completos, expresivos y fáciles de usar [26, 74].
 - Usan una notación formal para describir la esencia de un dominio de aplicación y capturar su semántica [133], estableciendo el significado de los datos en el contexto de sus relaciones con otros datos.
 - Mantienen el principio de conceptualización [139] según el cual las descripciones corresponden a un Universo de Discurso (*Universe of Discourse, UoD*) y no incluyen aspectos relativos al almacenamiento y acceso de los datos. [54, 75, 164].
 - Describen el dominio de una aplicación en alto nivel, con un vocabulario propio del dominio que se está representando, cuyos términos y conceptos son familiares a los usuarios, de manera que proveen una vista cercana a la forma como ellos perciben sus datos [54, 75].
 - Establecen bases comunes de comunicación entre desarrolladores, usuarios y otras partes interesadas en el desarrollo de una aplicación de *software* [133, 164]. Además, permiten integrar expertos de dominio, y su conocimiento, en el desarrollo de aplicaciones de *software*.
 - Capturan aspectos relevantes de un dominio, en un contexto particular, y representan un acuerdo común a un grupo de personas [133].

Algunos modelos conceptuales de datos son: el Modelo Entidad Relación (MER) [41], *Object-Role Modeling* (ORM) [74], *Integration Definition for Information Modeling* (IDEF1X)¹ [94], IFO [2], *Domain Specific Conceptual Data Model* (DSC-DM) [186] y *Semantic Database Model* (SDM) [76]. Entre los modelos de datos conceptuales también se encuentran los modelos semánticos y los orientados a objetos. Los modelos semánticos representan los datos de manera cercana a como los percibe el usuario. Estos modelos describen las características del dominio mediante entidades, relaciones y restricciones [150], e incluyen mecanismos de abstracción como la agregación, la clasificación e instanciación, la generalización y la asociación. Los modelos de datos orientados a objetos surgen a partir de las coincidencias entre los modelos de datos semánticos y los modelos de los lenguajes de programación orientados a objetos. Estos modelos incluyen el comportamiento de los objetos, la herencia y encapsulan los atributos.

Los modelos conceptuales han sido usados en diversas áreas de las ciencias de la computación [164], entre otras, la ingeniería de *software*, las bases de datos, los sistemas de información, la representación del conocimiento, la administración de procesos de negocio, la *web* semántica, la computación orientada a servicios y los sistemas multiagentes.

Los modelos de datos de nivel conceptual se usan para apoyar el diseño de bases de datos [179], en arquitecturas de integración de datos para crear vistas unificadas de repositorios de datos que tienen diferentes modelos lógicos [35, 130] y como base para

¹<http://www.idef.com/IDEF1X.html>

el desarrollo de lenguajes [73]. Un lenguaje en el nivel conceptual puede tener dos roles: la definición del modelo o la recuperación de datos. Los que cumplen con este último rol son Lenguajes de Consulta Conceptuales [140] que se caracterizan por tener un alto nivel de abstracción y ser de fácil uso. Estos lenguajes buscan brindar a los usuarios mecanismos que les permitan formular consultas complejas de forma sencilla, obviando las desventajas que se presentan en la formulación de las consultas con lenguajes basados en modelos lógicos. Un lenguaje de consulta en el nivel conceptual permite al usuario expresar consultas con mayor nivel de abstracción, de forma similar a como lo hace en su dominio y con términos específicos del mismo [114]. Conquer [28], LISAD [184], COQL [173], CUDL [104] y Constellation Query Language [83] son algunos lenguajes conceptuales propuestos en la literatura.

2.1.2 Modelos de datos basados en grafos

Angles y Gutierrez [7], definen un modelo de datos basado en grafos como aquel en el cual las estructuras de datos para el esquema y/o para la instancia se modelan usando un grafo o una generalización de la noción de grafo; la manipulación de los datos se expresa por transformaciones del grafo o primitivas que operan sobre características propias de los grafos (tales como caminos, vecinos, subgrafos, patrones de grafos o conectividad); y las restricciones de integridad se definen sobre la estructura del grafo.

En términos generales, un grafo se define como una pareja $G = (N, E)$, donde N es un conjunto finito de nodos y $E \subseteq N \times N$ es un conjunto finito de arcos. Un nodo n es incidente a un arco e si $n \in e$. Dos nodos n y m son adyacentes (o vecinos) si (n, m) es un arco en G . El grado de un nodo n es el número de arcos de n . Un camino en un grafo G es una secuencia de nodos $\{n_1, \dots, n_k\}$ tal que (n_i, n_{i+1}) es un arco de G ($1 \leq i \leq k$).

Bajo esta definición, se encuentran diferentes tipos de grafo: grafos dirigidos (en cada arco se distingue un nodo inicial y un nodo final), multigrafos (admiten múltiples arcos entre dos nodos), grafos con nodos etiquetados, grafos con nodos atribuidos (agrega datos a los nodos), grafos con arcos etiquetados y grafos con arcos atribuidos, entre otros.

Los modelos de datos de grafos se basan en la definición matemática de grafo [7, 65], pero tienen diferencias, por ejemplo, con respecto a la estructura de datos (e.g. grafos dirigidos o no dirigidos, arcos y nodos etiquetados o no etiquetados, arcos y nodos atribuidos o no atribuidos, hipernodos [117, 156] e hiperarcos [116, 157]) y si considera la separación entre esquema e instancia. La representación de entidades y sus relaciones es fundamental en un modelo de datos basado en grafos [7]. Una entidad u objeto representa algo que existe como una unidad singular y completa, mientras que una relación es un predicado que establece una conexión entre dos o más entidades. Por tanto, de manera general, la estructura de datos de un modelo basado en grafos consta de un conjunto de nodos que representan los objetos (o entidades) del mundo

real y un conjunto de arcos que relacionan los nodos y representan atributos de un objeto o relaciones entre objetos. Cuando los nodos o los arcos son etiquetados, la etiqueta puede hacer referencia al tipo de entidad, al tipo de arco o a un valor (cuando el nodo representa una clase primitiva, i.e. una clase que no tiene atributos [111], que representa un tipo de datos básico). De otra parte, los nodos o arcos atribuidos tienen asociadas parejas \langle atributo, valor \rangle , que corresponden a las propiedades del objeto que el nodo o arco representa.

El uso de modelos de grafos en diferentes aplicaciones ha motivado el desarrollo de modelos de datos basados en grafos que representan las entidades y relaciones con diferentes estructuras, que se revisan en [7]. No parece haber consenso en la literatura especializada con respecto a la denominación de los diferentes tipos de modelos de datos basados en grafos de acuerdo con su estructura de datos. Por tanto, en este documento se usarán las siguientes denominaciones para hacer referencia a los tipos de modelo que son de interés en este trabajo y se usará el término “Grafo de Datos” para hacer referencia a todos ellos:

- **Grafo de datos relacional** (*data graph* [125] o *relational graph* [162]): es un grafo finito, dirigido y etiquetado G , definido como $G = \{N, E, L_N, L_E\}$, donde N es un conjunto finito de nodos, $E \subseteq N \times N$ es un multiconjunto finito de arcos dirigidos etiquetados y L_N y L_E son, respectivamente, conjuntos de etiquetas de nodos y de arcos. Una función, λ , asigna una etiqueta a los nodos y arcos, así para un nodo $n \in N$, $\lambda(n) \in L_N$ y para cada arco $e \in E$, $\lambda(e) \in L_E$. Los datos están representados en las etiquetas de los nodos y los arcos.
- **Grafo de datos con esquema**: en este documento se denominan grafos con esquema a aquellos modelos de grafos de datos relacionales que definen de manera específica un grafo esquema y un grafo instancia. Por tanto, cada nodo (respectivamente arco) pertenece a una clase de entidad (respectivamente tipo de relación). El esquema y la instancia son grafos finitos dirigidos y etiquetados. Los nodos del esquema representan clases y tipos básicos, mientras que en la instancia representan objetos de las clases o valores de tipo básico. En ambos grafos los arcos representan atributos o relaciones entre clases.
- **Grafo de datos atribuido** (*Attributed Graph* [198] o *property graph* [183]): En estos grafos los atributos son el contenido de los nodos y/o arcos y pueden almacenar etiquetas, palabras clave, comentarios o propiedades. Son modelos semiestructurados que permiten tener definiciones diferentes de conjuntos de parejas atributo-valor en cada nodo y arco. Se definen como un grafo dirigido $G = \{N, E, \mathcal{F}_A\}$, donde N es un conjunto finito de nodos, $E \subseteq N \times N$ es un conjunto finito de arcos y $\mathcal{F}_A(u)$ es una función tal que para cada nodo o arco $u \in N \cup E$, $\mathcal{F}_A(u)$ es una tupla $(A_1 = a_1, \dots, A_n = a_n)$, donde a_i es una constante y A_i es un atributo de u .
- **Grafo de datos con tipos de clase** (*Multivariate Graph* [174]): a diferencia de los grafos atribuidos, que no definen tipos de clase, estos grafos definen tipos de

clase. Es decir las clases modelan conjuntos de nodos con propiedades estructurales comunes. La clase determina los atributos que poseen todos los nodos que son instancia de esa clase y las relaciones en las que pueden participar.

Los grafos de datos se han usado ampliamente en diferentes áreas, entre ellas, el análisis de datos biológicos (ej. estudios del genoma y modelos bioquímicos o químicos) [42, 57, 67, 103], las redes sociales [144, 151, 159] y la *web* semántica [7, 9, 153]. A continuación se listan algunos trabajos que usan o proponen modelos de datos basados en grafos, detallando dos aspectos: la estructura del grafo y la representación del esquema y la instancia.

Grafos de Datos Relacionales

Mandreoli et al. [125], Barceló [16], Libkin y Vrgoč [119] y Wood [197] usan grafos de datos relacionales con diversos propósitos.

El modelo propuesto por Mandreoli et al. [125], que tiene como objetivo soportar consultas aproximadas sobre un grafo de datos, es un multigrafo etiquetado y dirigido. El modelo incluye dos tipos de nodo, de valor y entidad, que se diferencian por el tipo de la etiqueta, literales o conceptos. Si bien este modelo no define específicamente el esquema y la instancia de datos, información del esquema se puede representar mediante el uso de arcos particulares como *type* y *subClassOf*.

Barceló [16] usa un grafo de datos para estudiar la expresividad y complejidad de evaluación de lenguajes de consulta sobre grafos y para definir extensiones que permitan expresar relaciones semánticas complejas entre caminos [15].

Libkin y Vrgoč [119] estudian la combinación de consultas sobre la topología del grafo y sobre los datos en grafos de datos.

Por su parte, Wood [197] hace una revisión de lenguajes de consulta sobre grafos y usa la definición de un grafo de datos como base para describir de manera formal los diferentes tipos de consulta.

Algunos autores consideran RDF como un grafo de datos. El grafo RDF se define como un conjunto de triples (*s*, *p*, *o*) [112] formadas por un sujeto (*s*), un predicado (*p*) y un objeto (*o*) que se identifican con un URI (*Uniform Resource Identifier*). Este conjunto de triples se puede visualizar por medio de un grafo dirigido y etiquetado. Sin embargo, el grafo RDF no corresponde con la definición clásica de grafo, ya que un URI se puede usar como sujeto (u objeto) en una tripla y como propiedad en otra. Por lo tanto, es posible tener arcos que conecten dos arcos o un arco con un nodo. Hayes y Gutierrez [80] muestran que un grafo RDF se puede representar con un hipergrafo y que éste, a su vez, se puede representar con un grafo bipartito.

Grafos de Datos con Esquema

Entre los grafos de datos con esquema están G-Log [148], Gram [5], PaMal [61], GOAL [88], GDM [87] y lgDM [200]. PaMal, GOAL y GDM extienden GOOD desde diferentes perspectivas. GOOD [69, 70] es un modelo de datos cuya instancia se representa con un grafo dirigido y etiquetado, pero su esquema se define con una tupla formada por los siguientes cinco conjuntos finitos: etiquetas de objetos no imprimibles (NPOL), etiquetas de objetos imprimibles (POL), etiquetas de arcos funcionales (FEL), etiquetas de arcos no funcionales (NFEL) y producciones (P). Las producciones definen con cuales objetos se puede relacionar cada objeto no imprimible y a través de qué tipo de arco. En la instancia los nodos corresponden a los objetos y su etiqueta pertenece a POL o NPOL, y los arcos tienen etiquetas que pertenecen a FEL o NFEL. Además los nodos de objetos imprimibles tienen una etiqueta adicional: *print label*.

GOAL [88] extiende GOOD definiendo con un grafo tanto el esquema como la instancia, y agrega nodos de asociación que permiten definir relaciones n-arias. Los nodos de asociación también pueden tener propiedades representadas por arcos. Además, GOAL agrega la noción de consistencia entre la instancia y el esquema, y ofrece un lenguaje de manipulación de datos basado en *pattern matching* que soporta la adición y borrado de nodos y arcos en la instancia. Los autores denominan “transformación” a una secuencia finita de operaciones de adición y borrado.

GDM [87] adiciona la posibilidad de representar datos semiestructurados, permitiendo que las instancias existan sin un esquema definido y soporta valores complejos y herencia.

En PaMal [61] el esquema incluye nodos que se etiquetan con nombres de clase y arcos de cuatro tipos: los que asocian atributos, los que asocian elementos de un conjunto, los que asocian el tipo de clase y los que representan relaciones IS-A. En la instancia, se tienen cuatro tipos de nodo: objetos, valor básico, tuplas y conjuntos. PAMAL ofrece un lenguaje de manipulación de datos, orientado al usuario final, basado en búsqueda de patrones (*pattern matching*), que incluye las operaciones de adición y borrado de nodos y arcos.

G-Log [148] incluye relaciones multivaluadas entre los nodos objeto. En Gram [5] las etiquetas de los nodos son los identificadores de los objetos. Finalmente, lgDM [200] sigue el mismo enfoque de los modelos mencionados, con la diferencia de que en el esquema no se definen nodos de valor básico, solamente las clases de entidad y las relaciones entre ellas. lgDM soporta una estrategia de minería de asociaciones para crear asociaciones entre entidades a partir de información de diversas fuentes. Esta asociación es la base para la implementación de consultas basadas en palabras clave.

Grafos de Datos Atribuidos

Algunos modelos de grafos atribuidos [58, 154, 198] permiten definir atributos únicamente en los nodos. Fan et al. [58] presentan una técnica de ejecución de consultas

para grafos atribuidos. Xu et al. [198] estudian el problema de encontrar *clusters* en grafos atribuidos. Pfeiffer et al. [154] presentan un *framework* para generación de modelos de redes (ej. redes sociales) cuyo modelo de datos considera solamente atributos binarios. Otros modelos soportan atributos en los nodos y en los arcos [162, 185].

Grafos de Datos con Tipos de Clase

Algunos trabajos sobre grafos de datos con tipos de clase se presentan en [30, 109, 183] Tausch et al. [183] quienes estudian polimorfismo en este tipo de grafos. Kiesel et al. [109] presentan GRAS un sistema de bases de datos cuyo modelo de datos es un grafo con tipos de clase. Braunschweig et al. [30] proponen un modelo de datos basado en grafos atribuidos para soportar la integración de datos en la Web. Este modelo les ayuda a manejar la heterogeneidad y volatilidad característica de los datos en la Web.

Otros Modelos de Datos basados en Grafos

Algunos modelos incluyen estructuras complejas en los nodos o arcos. Los hipernodos [117] encapsulan un grafo en un nodo. El modelo de hipernodos (*Hypernode Model*) tiene como estructura única el hipernodo. Los hipernodos tienen tipos, que son a su vez hipernodos, y los atributos se representan con triples dentro de ellos. El modelo soporta relaciones de alto orden porque los nombres de las propiedades pueden ser objetos complejos representados como hipernodos. Además, soporta de manera natural herencia por la anidación de las estructuras.

Por su parte, los hipergrafos se caracterizan por tener arcos que relacionan dos conjuntos de nodos [84, 95]. GROOVY [116] ofrece hipernodos e hiperarcos para representar objetos complejos y clases que comparten sub-objetos. Además, los nombres de las propiedades pueden ser objetos complejos representados como hipernodos o hiperarcos, con lo cual se tienen relaciones de alto orden.

Otros enfoques [68, 137, 171, 199] usan una definición de capas o de niveles de abstracción.

EGDM [171] es un modelo de datos conceptual orientado a objetos enfocado al modelamiento de hipertexto y datos semiestructurados. Un modelo EGDM es un grafo organizado por capas. En la primera capa están los nodos objeto, que son nodos con atributos. Los nodos objeto se agrupan para constituir nodos clase que forman la siguiente capa. Los nodos clase que tienen el mismo tipo y las mismas relaciones, a su vez, se agrupan constituyendo la siguiente capa de abstracción. Finalmente, EGDM ofrece un constructor adicional, los nodos de grupo, que permiten modelar hipertexto.

DRHM [137] se enfoca en la representación de datos multimedia. El modelo incluye varios grafos: el grafo instancia, el grafo colección (que agrupa grafos instancia con estructura similar) y el grafo forma (que representa la estructura del grafo colección).

STAG [199] es un modelo de base de datos orientado al almacenamiento de registros digitales anotados. STAG consta de tres grafos definidos sobre el mismo conjunto de nodos y arcos. En el primer grafo los nodos son elementos atómicos y los arcos, relaciones estructurales. Las relaciones estructurales se producen cuando un registro digital contiene otros. El segundo grafo incluye otras relaciones y el tercer grafo registra las anotaciones, para ello asocia un color a los subgrafos del grafo de datos.

GraphDB [68] es un modelo que incluye la definición de un grafo esquema y un grafo instancia. GraphDB ofrece tres tipos de clases: *simple*, *link* y *path*. Cada clase puede tener atributos que pueden ser, a su vez, datos u objetos de cualquiera de las tres clases, esto permite definir grafos anidados dentro de un nodo o un arco. En el grafo esquema los nodos son clases simples y los arcos clases *link*. En el grafo instancia los nodos y arcos son objetos de clases simples y clases *link* respectivamente

Modelos de Datos en Motores de Bases de Datos de Grafos

Desde el punto de vista de los motores de bases de datos basados en grafos, Neo4j¹ [160] y DEX [127] implementan un modelo de grafos de datos con atributos, que denominan *Property Graph*. OrientDB² [45] incluye modelos de documentos y modelos de grafos. El modelo de grafos es un grafo de datos atribuido y está construido sobre el modelo de documentos. AllegroGraph³, Jena TDB⁴, 4Store⁵ [77], RDF3X [134] son *triple stores*, es decir que implementan un modelo RDF. HypergraphDB⁶ [95] es una base de datos embebida cuyo modelo de datos es un hipergrafo, un grafo en el cual los arcos apuntan a conjuntos de nodos o a otros arcos.

2.2 Lenguajes de consulta en grafos de datos

El estudio de consultas sobre datos estructurados en grafos ha cobrado especial relevancia en los últimos años debido a las múltiples áreas de aplicación de los modelos de datos basados en grafos [7, 119]. La manipulación de datos en grafos incluye constructores y operaciones relacionadas con transformaciones, caminos, búsqueda de vecinos, subgrafos, patrones, conectividad y estadísticas [7]. Entre los lenguajes incluidos en esta revisión, se encontraron dos mecanismos básicos de consulta: las consultas basadas en búsqueda de patrones (*pattern matching*) y las consultas basadas en recorridos de caminos (*path traversal*).

¹<http://neo4j.com>

²<http://orientdb.com>

³<http://franz.com/agraph/allegrograph/>

⁴<https://jena.apache.org/documentation/tdb/architecture.html>

⁵<http://4store.org>

⁶<http://www.hypergraphdb.org/index>

Un patrón es un grafo cuyos nodos y arcos pueden estar etiquetados con variables o literales. Las consultas basadas en patrones recuperan de la base de datos todas las ocurrencias que coinciden con el patrón. La respuesta puede ser un conjunto de subgrafos o un conjunto de mapeos que asocian a cada variable un valor. Generalmente, la búsqueda de patrones se define en términos del isomorfismo de subgrafos [58].

El recorrido de caminos (*path traversal*) [162] consiste en navegar por el grafo siguiendo los arcos para encontrar los caminos que inician en un nodo o un conjunto de nodos determinado y siguen un camino de ciertas características, especificadas en la consulta. El recorrido de caminos se refiere a visitar elementos del grafo (nodos o arcos) avanzando con pasos individuales, es decir avanzando por los arcos que salen de un nodo o llegan a él, y luego a los nodos que son cola (o cabeza) de un arco. En cada paso se pueden tomar los valores de las propiedades del nodo o arco, y aplicar filtros con base en estos valores o en las etiquetas.

Estudios formales sobre los aspectos teóricos y fundamentales de las consultas sobre grafos, incluyendo la complejidad de su evaluación (*query complexity, data complexity, y combined complexity*) han sido realizados por Wood [197], Barceló et al. [15], y Perez et al.[151], entre otros. Estos autores estudian las consultas sobre grafos y las clasifican, según su funcionalidad, en las categorías que se presentan a continuación. La descripción de estas categorías se hace sobre un grafo de datos relacional G definido con una pareja (V, E) , donde V es un conjunto finito de nodos etiquetados, y E es un conjunto finito de arcos dirigidos y etiquetados que conectan parejas de nodos; $E \subseteq V \times \Sigma \times V$, para algún alfabeto finito Σ .

- Las **Consultas conjuntivas** (**CQ** por sus siglas en inglés *Conjunctive Query*) corresponden con la búsqueda de subgrafos que coinciden con un patrón dado. El patrón es un grafo que incluye variables. Un CQ es una expresión de la forma [197]:

$$ans(z_1, \dots, z_n) \leftarrow \bigwedge_{1 \leq i \leq m} (x_i, a_i, y_i)$$

tal que $m > 0$, para cada i , $1 \leq i \leq m$, x_i y y_i son nodos etiquetados con una variable o una constante y $a_i \in \Sigma$. z_j es algún x_i o y_i ($1 \leq j \leq n$, $1 \leq i \leq m$).

El resultado de la consulta es un conjunto de mapeos que asocian valores a cada variable.

- Las **Consultas de caminos** (**RPQ** por sus siglas en inglés *Regular Path Query* [129]) ofrecen la posibilidad de encontrar nodos conectados por un camino que se especifica con una expresión regular. Un RPQ es una expresión de la forma [197]:

$$ans(x, y) \leftarrow (x, r, y)$$

donde x y y son nodos etiquetados con una variable o una constante y r es una expresión regular sobre Σ .

El resultado de la consulta es el conjunto de todas las parejas de nodos (x, y) de G tales que hay un camino entre x y y que satisface r .

- Las **Consultas conjuntivas con caminos** (**CRPQ** por sus siglas en inglés *Conjunctive Regular Path Query*) extienden los RPQ con conjunciones de los átomos.

Un CRPQ es una expresión de la forma [197]:

$$ans(z_1, \dots, z_n) \leftarrow \bigwedge_{1 \leq i \leq m} (x_i, r_i, y_i)$$

tal que $m > 0$, para cada i , $1 \leq i \leq m$, x_i y y_i son nodos etiquetados con una variable o una constante y r_i es una expresión regular sobre Σ . z_j es algún x_i o y_i ($1 \leq j \leq n$, $1 \leq i \leq m$).

El resultado de la consulta es un conjunto de mapeos que asocian valores a cada variable.

- Los **CRPQ extendidos (ECRPQ)** por sus siglas en inglés *Extended CRPQ* [15]) agregan a los CRPQs la capacidad de especificar relaciones entre caminos (ej. comparar caminos) y de retornar caminos. Un ECRPQ es una expresión de la forma [197]:

$$ans(z_1, \dots, z_n, \tau_1, \dots, \tau_q) \leftarrow \bigwedge_{1 \leq i \leq m} (x_i, \pi_i, y_i) \bigwedge_{1 \leq j \leq t} R_j(\bar{w}_j)$$

tal que $m > 0$, $t > 0$, cada R_j es una expresión regular sobre Σ , x_i y y_i son nodos etiquetados con variables, π_i son variables que representan caminos ($1 \leq i \leq m$), $\{\bar{w}_1, \dots, \bar{w}_t\}$ son tuplas de variables de caminos tal que cada \bar{w}_j es una tupla de variables con la misma aridad de R_j . z_j es algún x_i o y_i ($1 \leq j \leq n$, $1 \leq i \leq m$) y τ_l es algún π_i ($1 \leq l \leq q$, $1 \leq i \leq m$).

El resultado de la consulta es un conjunto de mapeos que asocian valores a cada variable de nodo y de camino.

Algunos trabajos estudian otras funcionalidades, adicionales a las anteriores o contenidas en ellas, entre las que se encuentran las siguientes:

- Consultas de accesibilidad (*reachability queries* [100, 161]): “En un grafo, un par de nodos es accesible si hay un camino entre ellos” [195]. Las consultas de accesibilidad encuentran si dos nodos dados son accesibles. Estas consultas son útiles, por ejemplo en bases de datos de bioinformática, para encontrar si un gen es indirectamente regulado por otro gen [100]
- Consulta de subgrafos (*subgraph query* [44] o *graph search* [99]): estas consultas se realizan sobre bases de datos que tienen muchos grafos (N grafos). La consulta se especifica con un subgrafo (a diferencia de los patrones, el subgrafo no incluye variables), y retorna todos los grafos de la base de datos que contienen el grafo de consulta. Este tipo de consultas es útil, por ejemplo, para buscar compuestos químicos en una base de datos [44].
- Cálculo de funciones de agregación [52, 197] propias de los grafos, tales como: el grado de los nodos, la distancia entre dos nodos, el número de nodos alcanzables desde cierto nodo, el camino mas corto entre dos nodos, la centralidad de un nodo o el diámetro del grafo.
- Consultas que arrojan respuestas aproximadas [64, 124].
- Ranking de los resultados de funciones agregadas o de respuestas aproximadas [52, 93].

A continuación se describen algunos lenguajes de consulta sobre grafos. Esta descripción se centra en el modelo de datos subyacente, en la forma básica de la consulta (basada en recorridos de caminos (*path traversal*) o en patrones (*pattern matching*)), en la capacidad del lenguaje para producir un grafo como respuesta a la consulta, en las opciones para manejar datos incompletos y en el uso del esquema de la base de datos para la formulación y ejecución de la consulta. En esta sección se incluyen solamente los lenguajes de consulta con sintaxis de texto. La revisión de los lenguajes visuales de consulta para grafos se presenta en la Sección 3.2.

SPARQL

SPARQL [78, 158], el lenguaje de consulta para RDF [112], es el lenguaje de consulta sobre grafos más usado y estudiado. La estructura de consulta básica de SPARQL es similar a la de SQL (*SELECT - FROM - WHERE*). De acuerdo con Perez et al. [152] una consulta SPARQL consta de una expresión compleja de un patrón (*RDF graph pattern*), que se especifica en el *WHERE*, y una expresión que indica cómo construir la respuesta, que se especifica con el *SELECT* (o *CONSTRUCT* o *ASK*) (Figura 2.1). Un *RDF graph pattern* está formado por triplets unidas por conjunciones, disyunciones y el operador opcional. Las triplets pueden contener variables, constantes y expresiones regulares para denotar caminos que unen parejas de nodos. El operador de filtro permite definir restricciones sobre las variables.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?email
WHERE { ?person foaf:name ?name . ?person foaf:age ?age .
OPTIONAL { ?person foaf:mbox ?email } . FILTER (?age > 20) }
```

Figura 2.1: Consulta en SPARQL

La evaluación de la consulta se hace en dos pasos, en el primero, se obtiene un conjunto de mapeos (*solution mappings*), donde cada mapeo está formado por un conjunto de valores asociados con las variables del patrón. En el segundo paso, se genera la respuesta de acuerdo a la forma especificada. La forma de la respuesta puede ser *SELECT* (genera una tabla de valores), *CONSTRUCT* (genera un nuevo grafo RDF) o *ASK* (una respuesta si/no).

El álgebra de SPARQL [78, 152], se basa en los operadores relacionales (selección, proyección, *join*, *left outer join*, *union* y diferencia), adaptados para operar sobre conjuntos de mapeos. Solamente el patrón básico de consulta (formado por una conjunción de triplets) se define y ejecuta sobre el grafo de datos, su resultado es un conjunto de mapeos. Las otras operaciones (ej. *union*, *optional*, *filter*) se definen sobre los resultados de los patrones básicos.

Además, una consulta puede contener *solution modifiers* que se aplican sobre el *solution mapping*, y que incluye operadores como *distinct*, *order*, *limit*, y funciones agregadas.

Si bien en RDF no es obligatorio definir el esquema de datos, esta definición se puede incluir usando propiedades particulares de RDF-S [31], como *typeof* o *subclassof*. Por tanto, el esquema también se puede consultar mediante SPARQL.

BiQL

BiQL [52, 53] es un lenguaje de consulta, con sintaxis similar a SQL, creado con el fin de soportar el análisis y el descubrimiento de conocimiento en grandes redes. El grafo de datos se almacena en tres estructuras: un *object store*, un *link store*, y un *domain store* (Figura 2.2). Los nodos y arcos se representan de manera uniforme como objetos almacenados en el *object store*. Los objetos tienen una identificación única y un conjunto de parejas atributo-valor que describen sus características. En el *link store* se guardan las relaciones entre pares de objetos, estas son relaciones dirigidas. Finalmente, el *domain store* permite asignar nombres a conjuntos de objetos. El modelo es semi-estructurado, los atributos de los objetos no corresponden a una definición previa.

Los datos se visualizan en grafos, seleccionando los dominios que actúan como nodos y como arcos. Por tanto un conjunto de objetos puede actuar como nodos en una visualización y como arcos en otra.

obj-id	features	
X1	{label=A, color=red}	Y1 ↔ X1
X2	{label=B, color=yellow}	Y1 ↔ X2
X3	{label=C, color=blue}	Y2 ↔ X1
X4	{label=D, color=yellow}	Y2 ↔ X3
X5	{label=E, color=red}	Y3 ↔ X2
Y1	{weight=0.2}	Y3 ↔ X3
Y2	{weight=0.5}	Y4 ↔ X3
Y3	{weight=0.8}	Y4 ↔ X4
Y4	{weight=0.2}	Y5 ↔ X3
Y5	{weight=0.9}	Y5 ↔ X5
(a) Object store		(b) Link store
name	objects	Comment
X	{X1, X2, X3, X4, X5}	Nodes
Y	{Y1, Y2, Y3, Y4, Y5}	Edges
(c) Domain store		<pre> CREATE PatZ AS SELECT <N1, E1, N2, E2, N3> {E1->,E2->} FROM X N1 -- Y E1 -- X N2 -- Y E2 -- X N3 WHERE N1.color = red AND N2.color = blue AND N3.color = yellow </pre>

Figura 2.2: BiQL estructuras y consulta [52]

Una consulta BiQL tiene la forma general *CREATE - SELECT - FROM - WHERE* (Figura 2.2). El *CREATE* es opcional y permite crear nuevos objetos que agrupan conjuntos de objetos existentes. El *SELECT* expresa que variables de la consulta

son usadas para definir un nuevo dominio. El *FROM* se especifica con una lista de expresiones de camino, que pueden definirse usando expresiones regulares y permite aplicar las operaciones union, intersección y diferencia de conjuntos. El *WHERE* define una condición sobre los atributos de los objetos. En el *SELECT* y en el *WHERE* es posible usar funciones agregadas (*count*, *sum*, *min*, *avg*, ...) para calcular los valores de los atributos de nuevos objetos. El resultado de una consulta es un conjunto de tuplas de objetos.

GraphDB

GraphDB [68] define una base de datos cuyo esquema e instancia son grafos. El modelo soporta tres tipos de clases: *simple*, *link*, y *path*. Cada clase puede tener atributos que pueden ser, a su vez, datos u objetos de cualquiera de las tres clases, esto permite definir grafos anidados dentro de un nodo o un arco. La Figura 2.3 muestra el modelo de una red de transporte público con tres niveles, el primero describe la red física, el segundo las líneas (estaciones y sus conexiones) y el tercero los horarios. En el grafo esquema los nodos son clases simples y los arcos clases *link*. En el grafo instancia los nodos y arcos son objetos de clases simples y clases *link*, respectivamente.

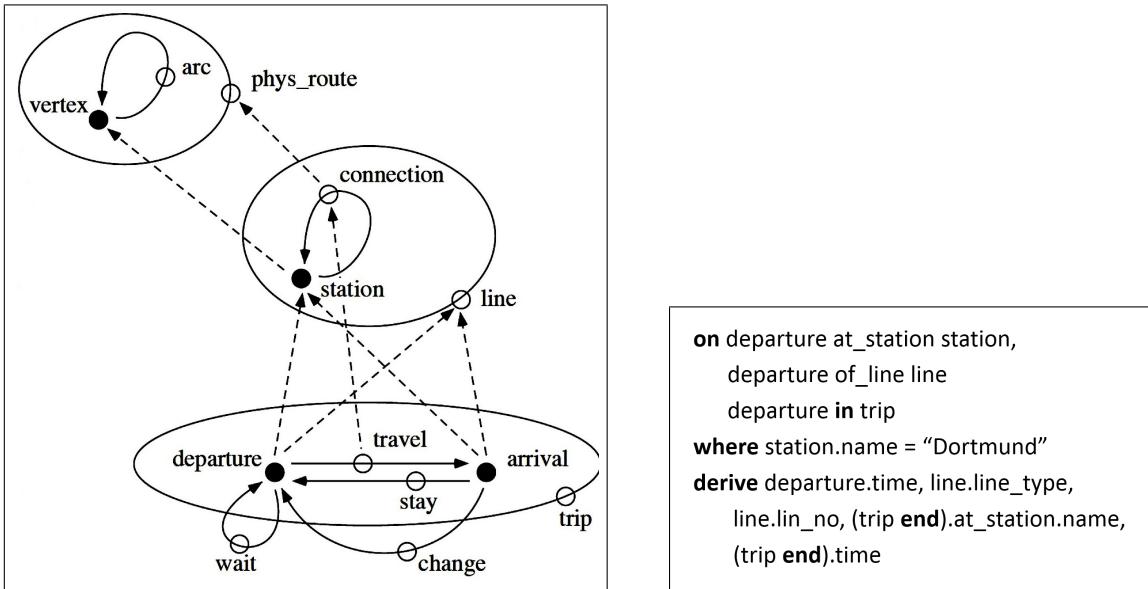


Figura 2.3: GraphDB modelo y consulta [68]

Las consultas permiten modificar temporalmente la base de datos o extraer información que se presenta en forma de tuplas. Las consultas incluyen: sentencias *DERIVE* y operaciones de reescritura, unión y grafos. La sentencia *DERIVE* tiene un estilo similar al clásico *SELECT - FROM - WHERE* y consta de tres partes: *ON*, *WHERE*, *DERIVE* (Figura 2.3). El *ON* (similar al *FROM*) especifica el subgrafo (las

clases) sobre el cual se va a realizar la consulta. La cláusula *ON* forma tuplas con los objetos seleccionados cuando estos tienen una relación establecida (ej. forman un camino), de lo contrario genera un producto cartesiano de las tuplas desconectadas. El *WHERE* filtra las tuplas del resultado del *ON* que cumplen una condición especificada. Finalmente, el *DERIVE* define cual será el resultado de la consulta, que en general crea uno o varios objetos en el grafo, de manera temporal. La operación de reescritura (*REWRITE*) permite especificar la transformación de un camino o partes del mismo. El camino y la transformación se definen con expresiones regulares, el resultado puede ser un conjunto de objetos (homogéneos o heterogéneos). La unión permite generalizar las tuplas de los objetos que se unen (encontrar una supertupla común a todos). El conjunto de operaciones de grafos (ej. camino mas corto entre dos objetos simples, subgrafo, k-vecinos de un nodo) puede ser extendido por el usuario.

StruQL

StruQL [59] es el lenguaje de consulta propuesto para Strudel, una plataforma basada en mediación para integrar datos en la red. Strudel usa grafos etiquetados y dirigidos para representar los sitios que integra y sus datos. StruQL es un lenguaje declarativo que permite hacer consultas y actualizaciones de los datos. Una consulta se aplica sobre uno o varios grafos y su resultado es también un grafo, por tanto es composicional. La consulta tiene dos partes: la cláusula *WHERE* que especifica el patrón de búsqueda con una expresión lógica sobre caminos (definidos con expresiones regulares) y predicados (ej. pertenencia a colecciones, *typeof*) (Figura 2.4). La segunda parte es una cláusula *LINK* que usa funciones Skolem para definir la creación de nodos (se crea un nodo por cada ocurrencia de una variable obtenida en el where) y especifica los arcos que asocian a estos nuevos nodos. La ejecución tiene las mismas fases, de consulta y de construcción. En la fase de consulta se hace una evaluación del *WHERE* y se obtiene una relación, resultado de un *binding* de las variables en los nodos y arcos a los valores del grafo que satisfacen la consulta. En la fase de construcción, se crean los nodos, arcos y colecciones que forman el grafo de salida.

```
WHERE Root{r}, r -> "pub" -> p, p -> "author" -> a
LINK HomePage(a) -> "name" -> a, HomePage(a) -> "paper" -> p
```

Figura 2.4: Consulta en StruQL.

GUL

GUL [87] es el lenguaje de actualización de datos propuesto para GDM. GUL está basado en patrones (*pattern matching*) y provee operaciones de adición, eliminación y reducción (unir nodos de tipo básico que tienen el mismo valor o nodos de tipo

compuesto con valores equivalentes y que pertenecen a la misma clase). La adición y eliminación se basan también en patrones.

GMOD

GMOD [6] representa el esquema y la instancia con grafos de datos (dirigidos y etiquetados). El grafo esquema tiene dos tipos de nodos: objetos abstractos (clases) y objetos primitivos (tipos básicos). Los arcos representan las propiedades de los objetos abstractos. Los autores proponen un lenguaje basado en *pattern matching* para transformación de los grafos (esquema e instancia) que se puede usar para actualizar la base de datos o hacer consultas. Una operación consta de tres patrones: uno de búsqueda, uno de eliminación y uno de adición. La operación busca los subgrafos de la instancia que coinciden con el patrón de búsqueda, y aplica sobre ellos la eliminación o adición de los patrones respectivos. El esquema solo se actualiza con la adición.

G-SPARQL

G-SPARQL [169] es un lenguaje para consulta de grafos de datos atribuidos. G-SPARQL está basado en *pattern matching*, tiene una sintaxis similar a la de SPARQL, y agrega la posibilidad de hacer referencia a los atributos de los nodos y de los arcos con una representación similar a la de las tripletas de SPARQL. Adicionalmente, G-SPARQL soporta consultas de accesibilidad (*reachability queries*) y camino más corto entre dos nodos.

NAGA

NAGA [106] es un lenguaje de consulta para YAGO, una base de conocimiento representada con un grafo de datos similar al modelo RDF, donde la metadata del esquema se representa en el grafo de datos por medio de propiedades particulares (ej. *instanceOf*, *subclassOf*). Las consultas se basan en patrones (*pattern matching*). Una consulta está formada por una conjunción de tripletas, cuyos extremos pueden ser variables o literales, y cuyas relaciones se pueden definir con expresiones regulares. El lenguaje también ofrece condiciones temporales (relaciones “desde”, “después de”) y un operador para encontrar relaciones comunes a un conjunto de entidades. Adicionalmente, NAGA hace ranking de los resultados de las consultas.

G-Path

G-Path [12] se enfoca en el procesamiento de patrones de camino en grafos de datos atribuidos. Un patrón de camino es un camino dirigido y etiquetado con condiciones en los nodos, arcos o atributos. Las condiciones se pueden especificar por medio de expresiones regulares. El resultado de una consulta es el conjunto de caminos que

satisfacen las condiciones dadas. El lenguaje ofrece además el ordenamiento de los resultados de la consulta.

GraphQL

GraphQL [81] es un lenguaje de consulta para grafos de datos atribuidos, basado en patrones (*pattern matching*). Un patrón de consulta se expresa con un *motif* con predicados en los atributos (Figura 2.5a). Un *motif* básico describe un grafo por medio de un conjunto de nodos y arcos identificados por variables. Los *motifs* complejos incluyen operaciones de concatenación, unión, y repetición (Figura 2.5b). Los predicados sobre los atributos pueden ser una combinación de operadores lógicos y de comparación. Las operaciones retornan conjuntos de grafos.

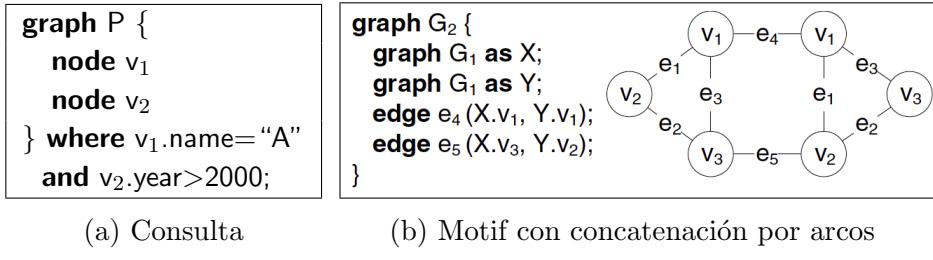


Figura 2.5: GraphQL [81].

GOQL

GOQL [176] extiende OQL con constructores para crear, manipular y consultar grafos. El lenguaje se basa en un modelo de datos orientado a objetos, en el que se definen los tipos nodo, arco, camino y grafo. Una consulta tiene la forma *SELECT – FROM – WHERE*. El lenguaje incluye cláusulas para encontrar los caminos entre dos nodos, subsecuencias en los caminos, concatenar caminos y en un camino encontrar: el primer nodo, el último nodo, el conjunto de nodos y el conjunto de arcos. Además, el lenguaje soporta predicados sobre caminos y secuencias

Cypher

Cypher [185] es un lenguaje de consulta declarativo basado en recorrido de caminos (*path traversal*), desarrollado para Neo4j. Una consulta se compone de *START*, *MATCH*, *WHERE*, *RETURN* (Figura 2.6). El *START* determina un conjunto de nodos donde inicia el recorrido del camino que se especifica en el *MATCH*. Cuando el *START* se omite, el motor busca un conjunto de nodos apropiado para iniciar el recorrido. La cláusula *MATCH* soporta, entre otros, relaciones opcionales y la

búsqueda del camino mas corto entre dos nodos. El *WHERE* permite establecer filtros sobre los datos y el *RETURN*, definir las variables a retornar. Estas pueden ser nodos, relaciones, propiedades o caminos. Cypher soporta las cláusulas *distinct*, *order by*, y funciones agregadas —*count*, *sum*, *collect*.

```
START place=node:node_auto_index(name = "CoffeeShop1")
MATCH place<-[:favorite]-person-[:favorite]->stuff
RETURN stuff.name, count(*)
ORDER BY count(*) DESC, stuff.name
```

Figura 2.6: Consulta en Cypher.

Otros Lenguajes

En esta revisión no se incluyeron lenguajes definidos para modelos de datos semiestructurados, que se pueden aplicar, entre otros modelos, a datos de grafos, tales como Lorel [3] o UnQL [34]. Tampoco se incluyeron lenguajes sobre modelos de datos de grafos específicos para algún dominio de aplicación, tales como SoQL [163] cuyo modelo de datos está enfocado a redes sociales (los nodos son los participantes y los arcos representan relaciones de amistad) o bcnQL [201] cuyo modelo es específico para redes bioquímicas, e incluye dos tipos de objetos: *biochemical entities* y *biochemical interactions*.

2.3 Síntesis sobre modelos y lenguajes de consulta basados en grafos

En los últimos años se han propuesto una amplia variedad de modelos de datos basados en grafos, tanto en dominios de aplicación específicos como de carácter general. La complejidad de las estructuras que ellos utilizan varía entre unos y otros, encontrando desde grafos con estructuras simples, como grafos de datos etiquetados con un solo tipo de arco (*single relational data*), hasta estructuras complejas que soportan una jerarquía de grafos.

Los modelos de datos basados en grafos permiten representar datos estructurados, semiestructurados o no estructurados. También son un buen mecanismo para representar, de forma integrada, datos creados en bases de datos autónomas, distribuidas y con diversos modelos de datos (ej. relacional, XML o RDF); esto debido a que la mayoría de los modelos de datos se pueden mapear fácilmente con un modelo de grafos, ya que los nodos pueden guardar datos de diferente nivel de granularidad [168].

Algunos modelos definen de manera explícita un grafo esquema separado de la instancia de datos. En otros casos, la información del esquema se incluye en el grafo de datos como una metadata asociada a los nodos y arcos o representada por nodos y arcos que se relacionan a través de propiedades particulares con los nodos y arcos que representan los objetos y valores literales.

Finalmente, la flexibilidad en la representación de los datos propia de los modelos basados en grafos hace que estos se puedan aplicar en múltiples dominios, tales como la biología, la bioinformática, el modelamiento de procesos de negocio, las redes sociales, y la *web* semántica, entre otros.

En cuanto a los lenguajes de consulta sobre grafos, se observa a lo largo de los años la evolución de las operaciones y de la funcionalidad. La mayoría de los lenguajes revisados se basa en búsqueda de patrones (*pattern matching*), pocos se basan en recorrido de caminos (*path traversal*). De acuerdo con la literatura, los recorridos de caminos pueden ser más eficientes que la búsqueda de patrones, sin embargo, para aplicar recorridos de caminos se requiere conocer o encontrar un conjunto de nodos a partir de los cuales se inicia el recorrido, lo cual puede ser una limitante para algunas consultas.

Los modelos subyacentes a los lenguajes varían también ampliamente, dependiendo del enfoque de cada lenguaje, y muchos están orientados en la consulta de datos semiestructurados, por lo que no hacen uso de un esquema de datos para soportar la formulación y ejecución de las consultas.

De otra parte, pocos lenguajes incluyen operaciones que permitan manejar de manera explícita datos incompletos (ej. la cláusula *optional* de SPARQL).

SPARQL por su parte ha evolucionado e incluido varias de las propuestas de consultas que se encuentran en la literatura.

En general, los lenguajes estudiados tienen una sintaxis y una semántica compleja lo cual los hace inapropiados para la formulación de consultas por usuarios finales.

Capítulo 3

Sistemas de Consulta sobre Grafos de Datos Orientados al Usuario Final

En este capítulo se introduce la noción de usabilidad en *software*, se describen, de manera general, los métodos más frecuentemente utilizados para definir y evaluar los atributos de usabilidad de un sistema, y se presentan algunas técnicas usadas para el desarrollo de herramientas con características de usabilidad. Además, se describen enfoques que han sido propuestos para solucionar algunos de los problemas de usabilidad en los sistemas de bases de datos. Finalmente, se revisan los trabajos relacionados con el desarrollo de lenguajes de visuales de consulta sobre modelos de grafos.

3.1 Usabilidad

Tradicionalmente, el diseño de productos se enfoca en la funcionalidad (el producto hace lo que se requiere que haga) y en la apariencia estética (el producto es agradable a la vista) pero no en cómo las personas van a usarlo (¿Es fácil o difícil de usar?, ¿Cómo se siente interactuar con el producto?) [60]. La experiencia de usuario va más allá de la funcionalidad y la estética, tiene que ver con el “cómo” se usa el producto. Cuando se trata de un producto simple (ej. una silla o una mesa) la experiencia de usuario se funde con la funcionalidad. Pero en la medida en que los productos son más complejos, la experiencia de usuario se hace independiente de la definición de los mismos. Entre más complejo es el producto, mas difícil es identificar cómo entregar una experiencia de usuario satisfactoria.

Algunos autores distinguen entre usabilidad y experiencia de usuario [192]. La usabilidad considera la habilidad del usuario para llevar a cabo una tarea, de manera satisfactoria, usando un producto. Mientras que la experiencia de usuario tiene en

cuenta la interacción con el producto, así como los pensamientos, sentimientos y percepciones que resultan de esa interacción.

El estándar ISO 9241-11 define la usabilidad como “el grado con el que un producto puede ser usado por usuarios determinados para conseguir objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso particular” [192]. En esta definición, la usabilidad se equipara con la calidad de uso [24] como resultado de la interacción entre el usuario y el producto para llevar a cabo una tarea en un contexto determinado.

La usabilidad de un producto se evidencia por medio de los siguientes atributos [165]: La **utilidad**, es el grado con que el producto permite al usuario alcanzar sus objetivos. La **eficiencia** se refiere a la rapidez con la que el usuario realiza la tarea. La **efectividad** mide la facilidad de uso y el grado con que el producto se comporta de la forma esperada por el usuario. La **satisfacción** se relaciona con la percepción del usuario respecto al uso del producto. La **facilidad de aprendizaje** tiene que ver con la habilidad del usuario para operar el sistema después de un periodo de entrenamiento determinado. Finalmente, la **accesibilidad**, que incluye elementos de interacción dirigidos a personas con algún tipo de discapacidad.

3.1.1 Usabilidad en *software*

De manera similar al diseño de productos, tradicionalmente el diseño de *software* se ha enfocado en dos aspectos: qué hace el producto (funcionalidad) y cómo lo hace (aspectos técnicos) [60]. Esto ha llevado a que las herramientas de *software* muchas veces se cataloguen como complicadas, confusas y difíciles de usar. Sin embargo, en los últimos años el diseño de las interfaces de los productos de *software* ha cobrado relevancia. Éste se centra en la selección de los elementos adecuados para que el usuario realice una tarea y en la organización de esos elementos en la pantalla [60].

De acuerdo con Nielsen [135] las siguientes son características básicas de una interfaz desarrollada con principios de usabilidad: Tiene un diseño simple y natural, que contiene solamente la información que el usuario necesita. Usa un lenguaje natural y simple, el lenguaje del usuario, y asocia los conceptos del usuario con los conceptos computacionales por medio de metáforas adecuadas. Minimiza la carga de memoria del usuario desplegando elementos de diálogo que lo guían. Es consistente, por ejemplo, en los efectos de los comandos o en la ubicación de la información. Realimenta constantemente al usuario. Mantiene tiempos de respuesta tan rápidos como sea posible y usa indicadores de avance en los procesos con duración prolongada. Ofrece, en todo momento, salidas claramente marcadas. Ofrece *shortcuts* para usuarios expertos, un historial de interacción y valores por defecto. Tiene un buen manejo de los errores, con mensajes amables, claros y precisos, que ayudan al usuario a resolver el problema. Y, finalmente, provee documentación y ayuda.

Por su parte, las interfaces gráficas de usuario se basan en el uso de ventanas, iconos,

menús y un dispositivo apuntador [135]. Entre los mecanismo de interacción usados en estas interfaces está la manipulación directa [177] que se basa en la representación visual de objetos de interés para el usuario. El usuario controla el diálogo moviendo o manipulando los objetos. Además, la manipulación directa da la posibilidad de hacer realimentación continua al usuario. Las interfaces con manipulación directa cumplen con tres características: La representación continua de los objetos de interés. El uso de acciones físicas o botones, que reemplazan los comandos con sintaxis compleja. Y la aplicación progresiva de operaciones que son reversibles y cuyo impacto en los objetos de interés se visualiza inmediatamente.

El desarrollo de productos con características de usabilidad requiere de un proceso de diseño centrado en el usuario, en el que el producto se debe evaluar constantemente para corroborar que las características se cumplen. Las siguientes secciones detallan estos tópicos.

3.1.2 Diseño centrado en el usuario

El diseño centrado en el usuario (UCD por sus siglas en inglés) incluye un conjunto de métodos y técnicas para diseñar productos y sistemas que cumplen las características de usabilidad, teniendo en cuenta la experiencia de usuario [60, 165]. El diseño UCD se caracteriza por tener en cuenta al usuario durante el desarrollo del producto y cumple con tres principios básicos: el enfoque en el usuario desde las primeras etapas del desarrollo, la evaluación del uso del producto y el diseño iterativo.

Petrelli [153] propone cuatro fases de evaluación para el desarrollo de un producto de *software* con enfoque UCD: la primera fase se centra en entender al usuario, la tarea y el ambiente en el que se realiza. En la segunda fase se corroboran las ideas de diseño con un prototipo de baja fidelidad (ej. prototipo en papel). La tercera fase incluye una evaluación con un prototipo de alta fidelidad (ej. un prototipo funcional). En esta evaluación se inspeccionan los diferentes componentes del *software* y las características de la calidad de uso. Finalmente, en la cuarta fase se evalúa el *software* en un ambiente real, pero controlado.

Las evaluaciones de cada una de estas fases se lleva a cabo por medio de alguno de los siguientes métodos y técnicas del UCD [165]:

- Estudio etnográfico: consiste en observar al usuario en su sitio de trabajo o en el lugar donde normalmente usaría el producto.
- Diseño participativo: cuando un representante de los usuarios hace parte del grupo de diseño del producto.
- Grupos Focales: una reunión con un grupo de usuarios representativos para que ellos discutan y evalúen los conceptos preliminares del diseño.
- Entrevistas: realizar entrevistas para entender las preferencias de los usuarios con respecto al producto.

-
- Recorridos cognitivos (*Walk-throughs*): un experto en usabilidad explora, a través de un prototipo, la forma como se usaría el producto.
 - Ordenamiento de cartas: son usados para obtener criterios que permitan diseñar la organización del contenido, la funcionalidad y el vocabulario a usar en la interfaz.
 - Evaluación de expertos: es una revisión del producto hecha por un especialista en usabilidad que no esté involucrado en el diseño del mismo.
 - Evaluación heurística: es una revisión del producto hecha por un especialista en usabilidad quien juzga si el producto cumple con una serie de heurísticas (principios de usabilidad) definidas. La evaluación de expertos es similar, pero no se rige por un conjunto de heurísticas.
 - Pruebas de usabilidad: incluye técnicas para recopilar datos empíricos mientras se observa a usuarios representativos usar el producto para realizar algunas tareas. Estas pueden ser experimentos formales, que buscan confirmar o refutar una hipótesis, o pruebas iterativas cuyo propósito es exponer las deficiencias en usabilidad para refinar el producto.
 - Estudios posteriores a la liberación del producto: para recopilar datos por medio de entrevistas y encuestas que permitan mejorar la siguiente versión del producto.

Estas técnicas se clasifican en métodos de inspección, métodos de indagación y pruebas de usabilidad [123]. Los primeros, los métodos de inspección, son realizados por expertos en usabilidad que evalúan un conjunto de características específicas del producto. Entre estos métodos están los recorridos cognitivos, la evaluación heurística y la evaluación por expertos. Los segundos, los métodos de indagación, se realizan en etapas tempranas del desarrollo del producto con el fin de conocer las necesidades de los usuarios. Entre éstos están los estudios etnográficos, las entrevistas y los cuestionarios. Finalmente, en las pruebas de usabilidad se observa a usuarios representativos ejecutando algunas tareas relacionadas con la aplicación a evaluar. Estas últimas se describen con mayor detalle en la siguiente sección (Sección 3.1.3).

Generalmente, durante el diseño de un producto no se aplican todas las técnicas mencionadas, normalmente algunas de ellas se adaptan y se combinan dependiendo de las necesidades y restricciones del proyecto [165].

3.1.3 Pruebas de usabilidad

Rubin y Chisnell [165] definen las pruebas de usabilidad como un proceso que involucra participantes, que representan la población de posibles usuarios, para evaluar el grado con el cual un producto cumple un criterio de usabilidad específico. Los participantes usan el sistema para desarrollar un conjunto predefinido de tareas [135]. El objetivo general de las pruebas de usabilidad es identificar y rectificar las deficiencias de usabilidad existentes en el producto o aplicación, con el fin de corroborar que es útil, valorado por los usuarios, fácil de aprender y que ayuda a las personas a realizar una tarea de manera eficaz, eficiente y satisfactoria [165].

Con base en el objetivo que se plantea para la prueba y la etapa de desarrollo del producto en la que se realiza, Rubin y Chisnell [165] clasifican las pruebas de usabilidad en cuatro tipos:

- Pruebas Exploratorias o Formativas: se realizan en las primeras etapas del desarrollo, su objetivo es examinar la efectividad de los conceptos preliminares del diseño.
- Pruebas de Evaluación o Sumativas: se realizan en las primeras etapas o en las etapas intermedias del desarrollo. Se evalúa el producto a nivel de sus operaciones para determinar si el concepto del producto se ha implementado efectivamente.
- Pruebas de Validación: se realizan al final de ciclo de desarrollo del producto, para comparar la usabilidad del producto con estándares o *benchmarks* establecidos. Los estándares pueden tener origen en los objetivos de usabilidad planteados en el proyecto, en términos de eficiencia y efectividad (cuán bien y cuán rápido el usuario realiza las tareas) o en criterios de preferencia (alcanzar cierto nivel de valoración por parte del usuario).
- Pruebas de Verificación: también se realizan al final del ciclo de desarrollo para confirmar que los problemas descubiertos en pruebas previas han sido solucionados de forma apropiada.
- Pruebas de Comparación: se pueden realizar en cualquiera de las etapas del desarrollo del producto y en combinación con los otros tipos de pruebas con el fin de entender las ventajas y desventajas de diferentes diseños. En las etapas iniciales, combinadas con pruebas exploratorias permiten, por ejemplo, comparar diferentes estilos de interfaz. En las últimas etapas, pueden usarse para comparar el producto con sus competidores.

Estos tipos de prueba comparten la misma metodología básica, que tiene origen en los experimentos controlados [165]. Los experimentos controlados incluyen la formulación de una hipótesis, la selección de una muestra aleatoria de la población objetivo, la realización controlada del experimento, grupos de control y un tamaño de muestra suficiente para generar resultados estadísticamente significativos. Sin embargo, en el desarrollo de *software* estos formalismos resultan poco apropiados, entre otras razones porque: (a) el propósito de las pruebas de usabilidad es tomar decisiones informadas sobre las mejoras en el diseño del producto, (b) se requerirían especialistas para realizar la prueba, (c) es difícil elegir los participantes de manera aleatoria y (d) en muchos casos hace falta información sobre la población que permita decidir correctamente el tamaño de la muestra.

Por tanto, los test de usabilidad se realizan con un enfoque menos formal, con un plan de prueba que incluye:

- Las preguntas y el objetivo de la prueba
- Las características del grupo de usuarios participantes. Se trata de tener una muestra representativa de usuarios, no necesariamente elegidos de manera aleatoria.

-
- La descripción de la metodología que especifica cómo se va a realizar la prueba, cómo se desarrollará cada sesión, y el diseño de la prueba. Por su diseño, la prueba puede ser un estudio *between subjects*, cuando cada participante prueba un elemento diferente; o *within subjects*, cuando cada participante prueba todos los elementos.
 - La lista de actividades a desarrollar en la prueba.
 - La descripción del entorno, los equipos a utilizar y la logística necesarios para realizar la prueba.
 - Las actividades que el moderador desarrollará durante la prueba.
 - Las medidas que se tomarán para evaluar los resultados.
 - La descripción de cómo se reportarán los resultados.

Un aspecto importante del plan de prueba son las métricas que permiten medir o evaluar un fenómeno particular [192]. Las métricas de usabilidad deben ser sistemas confiables de medida, por tanto, deben ser observables, directa o indirectamente, y cuantificables. Se miden los atributos de usabilidad o aspectos importantes de la experiencia de usuario, relacionados con la interacción entre el usuario y el producto. Por ejemplo, la efectividad (el usuario completa la tarea), la eficiencia (cantidad de esfuerzo requerido para completar la tarea) o el grado de satisfacción del usuario. Algunas métricas comúnmente usadas en pruebas de usabilidad son: el grado con el que los usuarios pueden completar una tarea, el tiempo, el esfuerzo para realizarla (ej. el número de clics, el esfuerzo cognitivo), el número de errores y la facilidad de aprendizaje. La satisfacción se mide con cuestionarios que piden al usuario calificar los productos que han probado.

3.1.4 Usabilidad en los sistemas de bases de datos

Diversos enfoques hacen parte de la investigación en la usabilidad de los sistemas de bases de datos, entre ellos se encuentran: los lenguajes visuales de consulta, las interfaces de consulta basadas en texto, los mecanismos de personalización para incluir información del usuario y del contexto en el procesamiento de la consulta, la explicación de porque un dato aparece o no en la respuesta de una consulta, y el refinamiento de las consultas basado en realimentación dada por el usuario.

Lenguajes visuales de consulta

Los lenguajes visuales de consulta, o sistemas visuales de consulta, tienen como propósito ofrecer a usuarios no programadores la posibilidad de expresar consultas sobre las bases de datos. Existen diversas propuestas que se diferencian, entre otras cosas, por los mecanismos de interacción usados y por la expresividad de las consultas. En la sección 3.2 se detallan las características de estos lenguajes y se describen algunos trabajos enfocados en consultas sobre grafos de datos.

Interfaces de consulta en texto

Las interfaces de consulta en texto para usuarios finales se enfocan principalmente en consultas con palabras clave. Este tipo de consultas es apropiado para usuarios casuales [55]. Las consultas basadas en palabras clave sobre bases de datos relacionales (*Relational Keyword Search*) tienen gran complejidad debido al amplio espacio de búsqueda [126], ya que se busca, en cada tabla y atributo, las ocurrencias de las palabras clave de la consulta. Markowitz et al. [126] clasifican estos sistemas de acuerdo con su forma de ejecución, en sistemas basados en operadores y sistemas basados en recorridos de grafos. Los primeros, basados en operadores [4, 90], enumeran y ejecutan árboles de operadores con los que se busca cualquier combinación de las palabras clave. De otra parte, para la ejecución basada en grafos [25], se usa una estructura de datos en la que los nodos representan las tuplas de las relaciones, y los arcos conectan tuplas que pueden unirse por un *join* [126].

Otros trabajos, [57, 102], se enfocan en técnicas para incrementar la eficiencia de estas búsquedas. Los mecanismos de ejecución basada en grafos son directamente aplicables a grafos de datos. Las consultas sobre modelos de grafos, basadas en palabras clave, retornan un conjunto de estructuras que representan cómo se conectan, en la base de datos, los datos que contienen las palabras clave [149]. Las estructuras frecuentemente usadas para la respuesta son árboles [48, 63] o subgrafos [105]. Usualmente, las respuestas se clasifican de acuerdo con una calificación de su relevancia para retornar las k respuestas de nivel superior (*top-k*) [82].

Las interfaces de lenguaje natural para bases de datos (NLIDB por sus siglas en inglés, *Natural Language Interface to Database*) [108, 118, 120, 136, 146, 155] son otro enfoque de consultas basadas en texto. Estas prometen grandes ventajas, entre ellas su facilidad de uso, por no ser un lenguaje formal y porque permite especificar la negación y la cuantificación universal de forma natural. Sin embargo, debido a las variaciones lingüísticas y a la ambigüedad inherente al lenguaje natural, el desarrollo de NLIs (*Natural Language Interfaces*) es una tarea complicada y que consume gran cantidad de tiempo. Los NLIs con buenos tiempos de respuesta son, generalmente, definidos para un dominio o aplicación particular y adaptarlos a otros contextos requiere una extensa labor de reconfiguración manual [107]. Los lenguajes controlados, o restringidos, reducen ampliamente la variabilidad lingüística, sin embargo, no es fácil para los usuarios encontrar cuál es el límite de la cobertura del lenguaje [136], es decir, saber que pueden preguntar [107].

Mecanismos de personalización

Estos mecanismos incluyen información del usuario y del contexto en el procesamiento de la consulta [42, 72, 113]. Cuando esta información se tiene en cuenta, las respuestas de las consultas dejan de ser exactas y pasan a estar mediadas por elecciones guiadas por las preferencias del usuario [110]. Chen y Li [42] analizan las preferencias de todos

los usuarios y los clasifican en grupos, de manera que cuando un usuario propone una consulta, se aplican las preferencias asociadas a los grupos a los que el usuario pertenece.

Mecanismos de explicación

La explicación de porqué un dato aparece o no en la respuesta de una consulta es una forma de ayudar al usuario a encontrar respuestas precisas y completas para su requerimiento de consulta. Una falla en la precisión hace que se produzcan algunos resultados que no son relevantes para el usuario, mientras que una falla en la completitud hace que resultados potencialmente relevantes no aparezcan en la respuesta [97].

Los sistemas de explicación de los resultados [39, 85, 91] permiten que el usuario pueda encontrar la razón por la cual ciertos datos están en el resultado y porqué otros no lo están. Para explicar porqué un dato está en la respuesta se aplican técnicas de procedencia (*database provenance*) [43] que explican la relación entre los datos de la base de datos y los de la respuesta. Estas explicaciones pueden incluir de donde viene el dato o porqué se produjo en la respuesta. De otra parte, para explicar porqué un dato falta en la respuesta, se han propuesto diferentes enfoques, Huang et al. [91] explican las tuplas faltantes realizando modificaciones en los datos de manera que la tupla aparezca en el resultado de la consulta cuando se ejecuta sobre la base de datos modificada. Herschel et al. [85] aplican la misma idea, pero restringen la modificación a datos con proveniencia no confiable. Champan y Jagadish [39] identifican las tuplas faltantes según la semántica de los operadores que las filtran. Y Tran y Chan [191] refinan la consulta para crear una cuyo resultado incluya las tuplas faltantes.

Refinamiento de consultas

Finalmente, algunos trabajos proponen refinar las consultas usando realimentación dada por el usuario [49, 96, 108, 118, 122, 203]. El uso de guías o diálogos con este propósito se realiza en diferentes tipos de interfaces de consulta.

En el marco de las consultas basadas en lenguaje natural, Damljanovic et al. [49] proponen mecanismos para incrementar la habitabilidad de las NLIs para consultas de ontologías. La habitabilidad se refiere a la facilidad, naturalidad y efectividad con que los usuarios se expresan en el lenguaje. Un mecanismo para incrementar la habitabilidad es la realimentación, para lo cual muestran al usuario las posibles interpretaciones de la consulta y el ranking asignado por el sistema, de manera que el usuario puede elegir la interpretación mas conveniente. Un segundo mecanismo son los diálogos de clarificación que permiten resolver ambigüedades que el sistema no puede resolver de forma automática e identificar conceptos que no se encontraron en los datos. Para ello el sistema presenta al usuario la lista de posibles significados, de manera que el usuario puede seleccionar el adecuado.

Propuestas similares a los diálogos de clarificación se habían hecho en Aqualog [122], un sistema para consultas sobre la web semántica que produce tripletas a partir de consultas expresadas en lenguaje natural, y Querix [108], una NLI para consultar ontologías. Por su parte, NaLIR [118] ofrece una explicación, en lenguaje natural, de cómo se procesa la consulta para que el usuario pueda identificar malas interpretaciones y le muestra otras interpretaciones para que el usuario elija entre ellas. La experiencia de los autores los ha llevado a concluir que ésta es una mejor alternativa que pedir al usuario que re-frasee la consulta.

En el refinamiento de consultas basadas en palabras clave, Zenz et al. [203], provee una interfaz en la que el usuario digita las palabras clave y el sistema le propone opciones de construcción, cada una con un posible significado o uso de una palabra clave, y consultas completas que resultan de la combinación de algunas opciones de construcción. Luego el usuario puede elegir una consulta completa o refinar las consultas seleccionando opciones de construcción válidas.

Islam et al. [96] combinan la explicación y el refinamiento basado en realimentación en consultas SQL. Para ello, con cada consulta dividen la base de datos en dos conjuntos: las tuplas resultado y las tuplas noresultado. Luego el usuario brinda realimentación marcando algunas tuplas que considera inesperadas en ambos conjuntos. Con esta realimentación el modelo refina la consulta.

3.2 Lenguajes visuales de consulta sobre grafos de datos

Los Lenguajes Visuales de Consulta [36] tienen, generalmente, como propósito ofrecer a usuarios no programadores la posibilidad de expresar consultas sobre las bases de datos. Para cumplir con ese propósito se requieren lenguajes amigables y fáciles de entender y de usar. La implementación de un Lenguaje Visual de Consulta constituye un Sistema Visual de Consulta (*Visual Query System*).

Catarci [20, 36] define los Lenguajes Visuales de Consulta (*Visual Query Language*) como sistemas de consulta de bases de datos que usan una representación visual para describir el dominio de interés y para expresar las solicitudes sobre los datos. Las representaciones visuales son, usualmente, iconos, tablas, diagramas o una combinación de ellos.

Los iconos son metáforas visuales que hacen referencia a un objeto, concreto o abstracto, o a una acción. En los lenguajes visuales de consulta, los iconos han sido usados para denotar los objetos de la base de datos y las operaciones sobre ellos, de manera que una consulta se expresa con una combinación de iconos. Si bien los iconos son fáciles de entender cuando representan objetos físicos, se dificulta entenderlos cuando representan objetos abstractos o acciones. Adicionalmente, cuando el número de iconos es grande, el poder de discriminación del usuario disminuye. Por estas razones

hay pocas propuestas de lenguajes icónicos para consulta de datos.

Las tablas y formas son las representaciones más usadas en lenguajes visuales de consulta sobre bases de datos relacionales. Las consultas basadas en formas son un método de consulta sencillo, pero con poca expresividad y flexibilidad debido a que se requiere desarrollar una forma para cada consulta [98]. Las consultas basadas en tablas han sido ampliamente usadas como resultado de la adopción de interfaces estilo QBE [204] en aplicaciones de consulta comerciales.

Por su parte, los diagramas son una representación simbólica y simplificada de las categorías de objetos y sus relaciones. En los lenguajes visuales de consulta, se emplean para representar el modelo de datos subyacente y para soportar la selección de los elementos, el recorrido de conexiones para encontrar elementos adyacentes y la creación de conexiones entre elementos desconectados.

Los sistemas visuales de consulta sobre modelos de grafos usan, generalmente, diagramas. Su desarrollo ha seguido dos vertientes principales: De una parte, se encuentran las herramientas de exploración y análisis de datos, que se concentran principalmente en mecanismos de agrupamiento (*clustering*) y visualización de grandes volúmenes de datos, y el cálculo de propiedades del grafo (ej. centralidad, densidad, índice de influencia, entre otros). De otra parte están los sistemas que se enfocan en el lenguaje de consulta, entre los cuales se incluyen aquellos que proponen un nuevo lenguaje y las interfaces gráficas para lenguajes de consulta existentes. Estas últimas, ofrecen una representación gráfica de los elementos de un lenguaje de consulta con sintaxis en texto.

En las siguientes secciones se describen algunas herramientas representativas de estas vertientes, haciendo especial énfasis en el modelo de datos subyacente, en la expresividad de las consultas, en los mecanismos de interacción que proporcionan para formular dichas consultas y en la aplicación de técnicas de diseño centrado en el usuario. Esta revisión se concentra en aquellos sistemas que tienen como modelo de datos un grafo con atributos y/o tipos (la Sección 2.1 presenta una caracterización de los modelos de grafos). Se incluyen además sistemas definidos sobre SPARQL [158] o RDF [159], aunque el modelo RDF se define como un conjunto de triplets, y no es un grafo en el sentido matemático, porque permite algunas particularidades, por ejemplo, un objeto o relación puede ser a la vez sujeto u objeto y propiedad (a la vez nodo y arco).

3.2.1 Sistemas para la exploración y el análisis de datos representados en grafos

Los sistemas que se describen a continuación se caracterizan por incluir diversas propuestas para la visualización de grafos con grandes volúmenes de datos y por ofrecer mecanismos de exploración y análisis de estos grafos. La visualización incluye el despliegue del grafo (*layout*), el uso de colores para identificar los datos y la aplicación de técnicas para manipular el grafo, tales como *zoom* y lentes [189]. La exploración

y el análisis se basan principalmente en la aplicación de filtros para ocultar nodos y arcos que tienen cierta característica; el cálculo de propiedades de los grafos, como la centralidad o el índice de influencia; encontrar los k-vecinos de un grupo de nodos seleccionados; y el cálculo de particiones (*clustering*). Si bien la visualización y el análisis son elementos fundamentales en estos sistemas, en esta sección enfatizan los aspectos relacionados con la exploración, la navegación del grafo y la aplicación de filtros sobre los datos.

Shamir y Stolpnik [174] proponen formas de visualización para diferentes tipos de consultas sobre grafos con tipos de clase, como solución al problema de visualización de instancias con grandes volúmenes de datos. La exploración inicia sobre un grafo resumido en una representación jerárquica que se calcula agrupando nodos y aplicando algunos filtros. Las consultas permiten acceder a los detalles que se ocultan en la vista del grafo resumido, y los resultados se presentan, aplicando un concepto similar a los lentes [189], con ventanas que se superponen sobre el grafo (ver Figura 3.1a). El sistema incluye varias operaciones de consulta, clasificadas en tres tipos: topológicas, estadísticas y de contexto. Las consultas topológicas permiten explorar, a partir de un nodo particular, niveles inferiores de la jerarquía; encontrar como ocurren las relaciones dentro de un nodo que agrupa otros nodos y los caminos más cortos que unen los nodos de un conjunto; además, filtrar los resultados por el tipo de nodo y de arco. Las consultas estadísticas permiten visualizar los valores de un atributo, la distribución de los nodos o arcos según su tipo o el valor de uno de sus atributos,

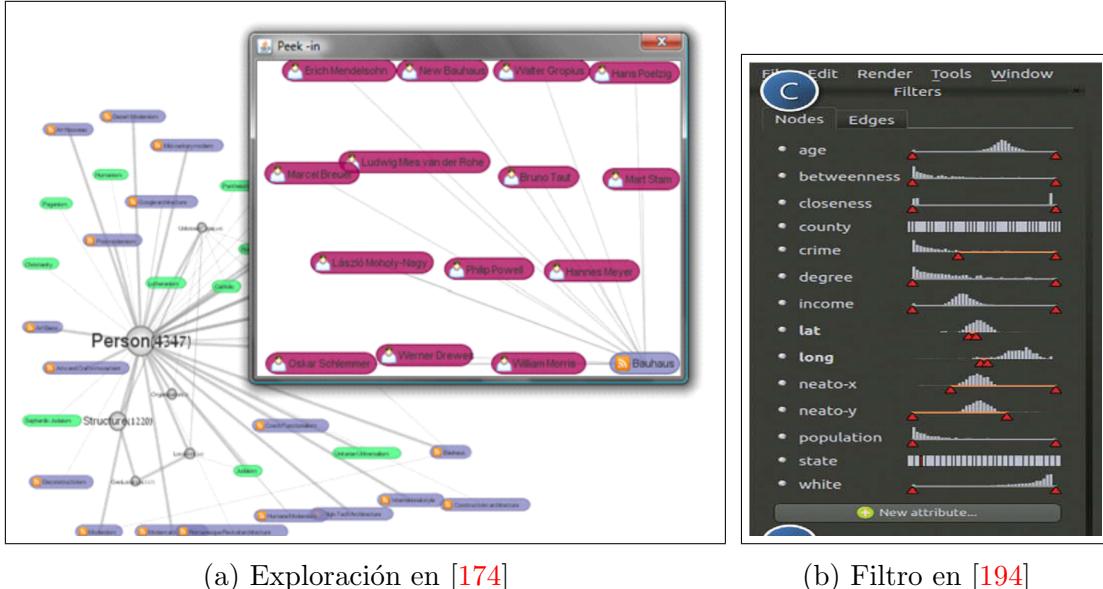


Figura 3.1: Ejemplo de exploración y filtro.

encontrar la cantidad de nodos vecinos en función de la distancia hasta un nodo particular y encontrar el número de nodos en función de su grado. Finalmente, las consultas de contexto permiten visualizar el resultado usando un contexto particular, por ejemplo una línea de tiempo o un mapa. Los autores reportan los resultados de una evaluación sumativa realizada con seis participantes.

Van den Elzen y van Wijk [194] proponen un método para el análisis y la exploración de grafos con tipos de clase, que incluye la selección de nodos y arcos por medio de filtros. El mecanismo de consulta permite seleccionar rangos de valores que deben cumplir los atributos de nodos y arcos (Ver Figura 3.1b)

CGV [188] tiene como modelo subyacente grafos con tipos y atributos. CGV soporta filtros para describir las condiciones que deben cumplir los nodos (o arcos) que se despliegan. Estos se pueden definir sobre etiquetas, valores de atributo, el grado de los nodos o el peso de los arcos. Además es posible componer estos filtros básicos con operaciones lógicas (Y, O) que se especifican por medio de una notación visual (Ver Figura 3.2). CGV ofrece diferentes vistas de los datos entre ellas el diagrama de grafo, árbol y ojo mágico. Los autores reportan que recibieron realimentación de los usuarios durante el desarrollo de CGV. Además, que los comentarios sobre la interfaz y los mecanismos de interacción de los usuarios que empezaron a utilizar la herramienta influenciaron el desarrollo del sistema en varios aspectos, entre otros, el orden de las vistas, el uso de parámetros de visualización y la forma de realizar el desplazamiento y el *zoom*. Sin embargo, los autores no reportan la realización de pruebas de usabilidad.

TouchGraph (<http://www.touchgraph.com/>) es una herramienta comercial que tiene como objetivo permitir el acceso a datos almacenados en bases de datos relacionales a través de una vista de grafos. El modelo de esta vista es un grafo con tipos de clase. El usuario carga los datos desde archivos o de MySQL y organiza el esquema identificando las entidades y las relaciones entre ellas. Una o varias entidades se pueden definir a partir de una o varias tablas. Durante la creación del esquema es posible definir relaciones y atributos dinámicos, que se calculan con base en relaciones



Figura 3.2: Filtro en CGV [188].

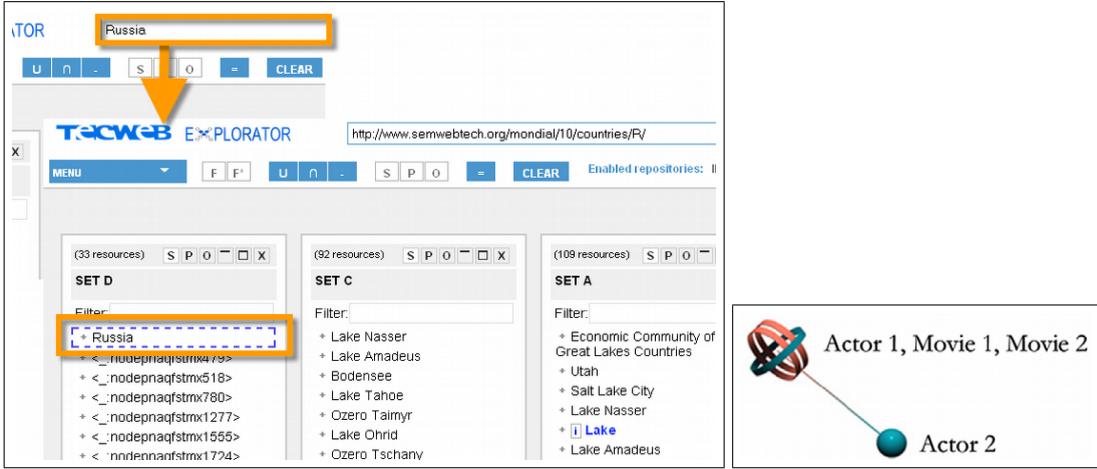
o atributos previamente definidos. Además, antes de iniciar la exploración de los datos, se deben definir los filtros que permiten seleccionar nodos y arcos por tipo o por condiciones sobre el valor de sus atributos. El esquema y los filtros se definen por medio de formularios donde, a partir de listas, se eligen las tablas y atributos que intervienen. Finalmente, el sistema despliega un grafo con los datos, y sobre esta vista el usuario puede seleccionar nodos, agregar los k vecinos de los nodos seleccionados (que también cumplen las condiciones de los filtros) y hacer particiones de los nodos (basado en *edge betweenness centrality*).

Gephi (<https://gephi.github.io>) [19] es un *software* de código abierto orientado al usuario final, cuyo modelo de datos es un grafo atribuido. Gephi permite cargar datos de formatos de grafos estándar (ej. GEXF, GML, GraphML) y de hojas de cálculo. Gephi ofrece dos vistas para la visualización de los datos: la vista de grafo y la vista de tabla. La vista de grafo permite manipular la instancia por medio de filtros para seleccionar nodos o arcos por el valor de un atributo (operaciones igual a, rango o no nulo), por el número de enlaces de los nodos o por el peso de los arcos. Es posible definir una lista de filtros, pero, en un momento dado, solamente se aplica uno de ellos. Cada filtro es un árbol, donde se componen filtros por anidación o por operaciones de unión, intersección y negación. Los filtros se construyen arrastrando atributos y operaciones desde una lista hasta el árbol. Gephi permite encontrar los k vecinos de un nodo o un grupo de nodos seleccionado.

Cytoscape (<http://www.cytoscape.org>) es una herramienta de *software* libre diseñada para la visualización y el análisis de redes de interacciones moleculares y genéticas. El modelo de datos es un grafo atribuido. Cytoscape carga los datos desde archivos con múltiples formatos, entre ellos, SIF, NNF, GML, GraphML, y excel workbook. Cytoscape permite filtrar nodos y arcos por la etiqueta, el valor de un atributo, el número de vecinos dentro de una distancia dada o la distancia a un grupo de nodos preseleccionado. Es posible combinar filtros con operadores lógicos de conjunción y disyunción. Los filtros se definen eligiendo, a partir de una lista, el atributo y el valor que debe cumplir.

GlyphLink [38] presenta un método de exploración de grafos con tipos de clase, basado en la creación de nodos que agrupan nodos de la instancia. Los grupos de nodos se representan con una metáfora, círculos enlazados, en la que la cantidad de círculos, sus colores y tamaños codifican características de los nodos miembros del grupo (Figura 3.3b). Los grupos conservan las relaciones con los elementos del grafo que permanecen por fuera el grupo. En cuanto a la selección de los datos, GlypgLink permite filtrar los nodos y arcos por tipo, para ello el usuario selecciona de listas los tipos a incluir. Los autores reportan un estudio de usabilidad con catorce participantes realizando dos tipos de tarea (siete participantes por cada tipo).

Explorator [9] es una herramienta de *software* libre para la exploración de grafos RDF. Las consultas se hacen por composición de una sola operación, denominada SPO. Un SPO define un conjunto de sujetos S que se enlazan por medio de un conjunto de



(a) Explorator [9]. (b) Glyphlink [38]

Figura 3.3: Ejemplos de Explorator y Glyphlink.

predicados P a un conjunto de objetos O. Los conjuntos resultado de las operaciones SPO se pueden usar en otra SPO (composición) o en operaciones de conjuntos (unión, intersección y diferencia). La interfaz está basada en el concepto de manipulación directa y consta de ventanas que permiten seleccionar los conjuntos y aplicar las operaciones (Ver Figura 3.3a). Los autores realizaron un estudio preliminar con seis usuarios que conocían conceptos básicos de web semántica y de RDF.

OntoVis [175] es una herramienta visual para el análisis de grafos semánticos (con tipos de clase), que permite formular consultas sobre un *ontology graph*, un grafo compuesto por los tipos de nodo y arco del grafo de datos. Las consultas permiten seleccionar nodos y arcos por su tipo y esconder arcos redundantes.

Otros sistemas, por ejemplo PGV [51], hacen la exploración agregando nodos vecinos a partir de un nodo inicial. En estos casos los mecanismos de selección de datos y filtros se aplican solamente para encontrar ese nodo inicial, por esta razón no se incluyeron en esta revisión.

3.2.2 Herramientas enfocadas en el lenguaje de consulta

GraphLog [47] y el lenguaje de consulta de GOOD [69] son pioneros de los lenguajes visuales para consulta sobre grafos.

GraphLog es un lenguaje de consulta para grafos de datos etiquetados, basado en patrones cuyos nodos se etiquetan con variables o valores literales y cuyos arcos tienen etiquetas o expresiones regulares. Las consultas permiten encontrar patrones que están (o no) presentes en el grafo, los patrones soportan expresiones regulares en los arcos. La semántica de las consultas está expresada con Datalog. Y el resultado de la consulta

es un grafo. La Figura 3.4 [47] muestra una parte de la representación en grafos de horarios de vuelos (3.4a) y una consulta de las conexiones posibles entre vuelos. La consulta se realiza con una secuencia de dos patrones: el primero, agrega la relación *feasible* (3.4b); y el segundo, agrega la relación *stop-connected* (3.4c). El artículo no describe una interfaz gráfica, y tampoco especifica el mecanismo de interacción para dibujar los patrones de las consultas.

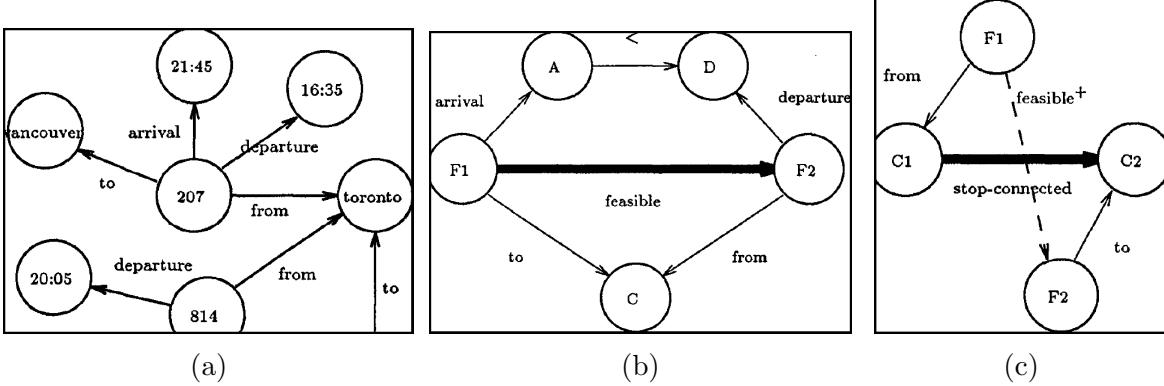


Figura 3.4: Modelo y consulta en GraphLog [47]

GOOD es un modelo de datos cuyo esquema se define con una tupla y cuya instancia se representa con un grafo dirigido, etiquetado y con tipos. La tupla del esquema incluye las etiquetas de los objetos, los arcos y las producciones. Gyssens et al. [69] proponen un lenguaje de consulta y actualización para GOOD. Las consultas se especifican con patrones y cinco operadores que transforman el grafo: cuatro operadores para agregar y borrar nodos y arcos, uno para crear nodos que reúnen grupos de nodos que comparten propiedades comunes. Los nodos, arcos y operaciones tienen una notación gráfica y los patrones se construyen a partir de la información del esquema [147]. Para seleccionar datos, se borran los nodos y los arcos que no se requieren. La adición puede ser usada para agregar marcas (nodos adicionales) que permitan identificar nodos y arcos que se requiere, o no, borrar. Por ejemplo, la Figura 3.5 muestra la consulta “*Encuentre los carros que no tienen dueño*” [69], para ello, el primer patrón (a la izquierda) agrega la propiedad “*no-own*” para todos los carros y el segundo patrón (a la derecha) borra la propiedad para aquellos que tienen dueño. Dado que la consulta se formula usando operaciones básicas, es posible que se requieran patrones largos y complejos.

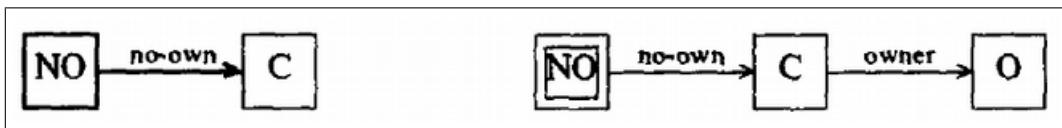


Figura 3.5: Consulta en GOOD [69].

MashQL [98] es un lenguaje de consulta de grafos orientado al usuario final, que usa un árbol como metáfora para guiar la formulación de las consultas. La raíz del árbol es una variable para la cual se define un conjunto de restricciones con patrones expresados por caminos que finalizan en un nodo con un filtro. Como resultado se forma una consulta SPARQL que incluye un patrón que contiene el *join* de los patrones generados para cada restricción. La interfaz guía al usuario en la creación del árbol de consulta, que inicia con la elección del sujeto raíz el cual es una variable, que puede estar restringida por el tipo de objeto. En este caso el tipo se selecciona de una lista. Para especificar las restricciones, el usuario forma caminos eligiendo una secuencia de propiedades. En cada paso, el sistema presenta una lista de opciones que incluye el conjunto de propiedades posibles de acuerdo con el sujeto y el camino seleccionados previamente. El filtro sobre el último nodo del camino se especifica con funciones (*equals*, *contains*, *between*, *less than*, *more than*, *one of*, *not*) que permiten comparar el valor del nodo con el valor de otra variable o con un literal. La consulta puede incluir las operaciones *union*, *optional*, y *unbound*. Además el usuario marca las variables de salida, que forman la cláusula SELECT de la consulta SPARQL. Un ejemplo de formulación de una consulta en MashQL se muestra en la Figura 3.6. El sistema no requiere que los datos tengan definido un esquema, cuando no lo tienen, MashQL calcula una firma del grafo RDF que permite ofrecer al usuario las listas de sujetos y propiedades durante la formulación de la consulta. La firma es un grafo dirigido en el que cada nodo agrupa nodos de la instancia que tienen los mismos tipos de arcos incidentes. La facilidad de uso de la herramienta fue evaluada por cuarenta participantes, veinticinco de ellos no eran especialistas en tecnologías de información.



Figura 3.6: Consulta en MashQL [98]

SNQL [170] es un lenguaje para consultar y transformar información de redes sociales representadas con un grafo SNDM. Una consulta en SNQL consta de dos patrones, uno de extracción y otro de construcción, que pueden ser expresados de manera gráfica, o en texto con una sintaxis similar a SQL. La etapa de consulta genera tuplas que asocian los valores de respuesta a las variables, y la etapa de

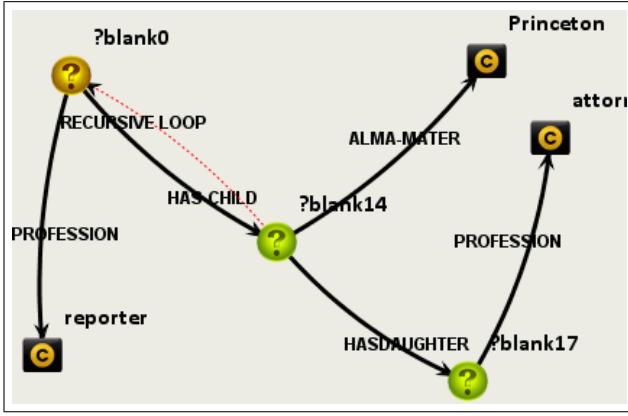


Figura 3.7: Consulta Recursiva en [167]

construcción toma ese resultado intermedio para generar las tripletas que forman el grafo de respuesta. SNQL soporta relaciones n-arias, permite transformar los atributos en nodos y crear atributos con los resultados de funciones agregadas. En [170] no se detalla el mecanismo de interacción para crear los patrones.

Sadanandan et al. [167] proponen un sistema visual para consultar bases de conocimiento representadas en RDF [159]. Para formular la consulta el usuario debe construir un patrón, este proceso inicia eligiendo una variable o un concepto de una lista de conceptos de la ontología. Luego agrega propiedades y otros conceptos hasta construir el patrón. El sistema ofrece una notación visual para soportar consultas recursivas (Ver Figura 3.7). La herramienta genera una consulta SPARQL para el patrón de la consulta. Los autores reportan una prueba de usabilidad con seis participantes con conocimientos básicos en ontologías de dominio y consultas semánticas y con dos usuarios expertos en esos temas.

Barzdzins et al. [17] proponen un sistema que usa una ontología OWL[128] para guiar a los usuarios en la formulación de consultas SPARQL sobre una base de datos RDF. La formulación inicia con la selección, en una lista de conceptos de la ontología, de los conceptos o clases que ser requieren en la consulta. Luego el sistema encuentra los caminos mas probables (los mas cortos) que unen los conceptos seleccionados. Estos caminos se presentan en una lista al usuario para que seleccione el adecuado. Además el usuario puede agregar condiciones en los atributos de algunas clases seleccionadas. Estas condiciones se conjugan con operaciones AND y OR (Ver Figura 3.8).

OntoVQL [56], un lenguaje gráfico para consulta de ontologías OWL-DL [128], propone una notación visual para los conceptos de la ontología (óvalos), instancias (rectángulos), variables (ovalo etiquetado con un signo de interrogación) y para las operaciones de unión e intersección. Los nodos en la consulta representan conjuntos de instancias o valores. Estos conjuntos se operan con la composición de operaciones de unión e intersección (Ver Figura 3.9). El sistema genera la consulta en RQL.

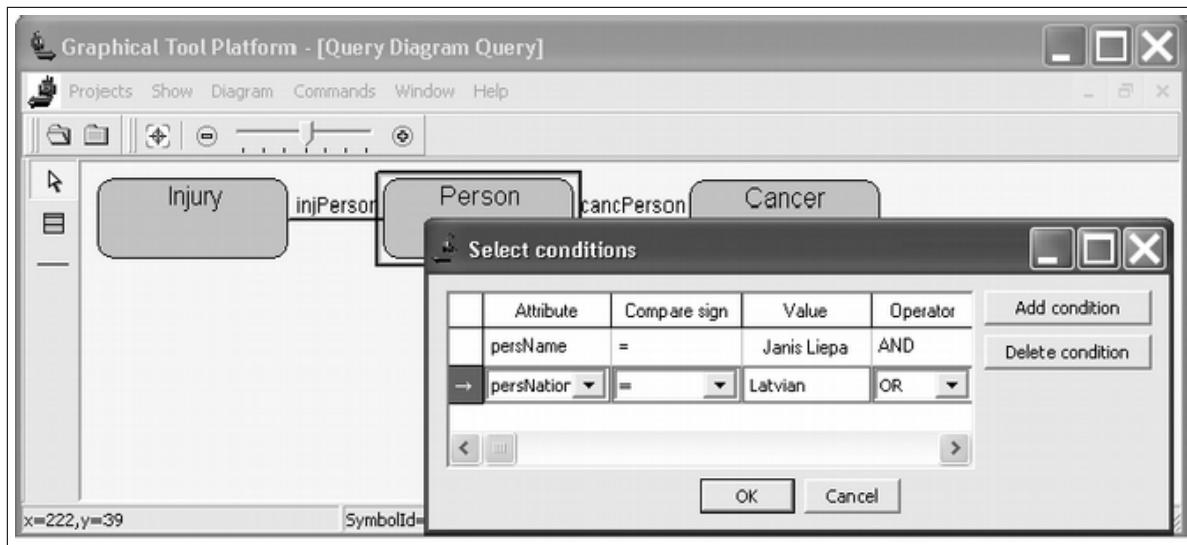


Figura 3.8: Filtro en [17].

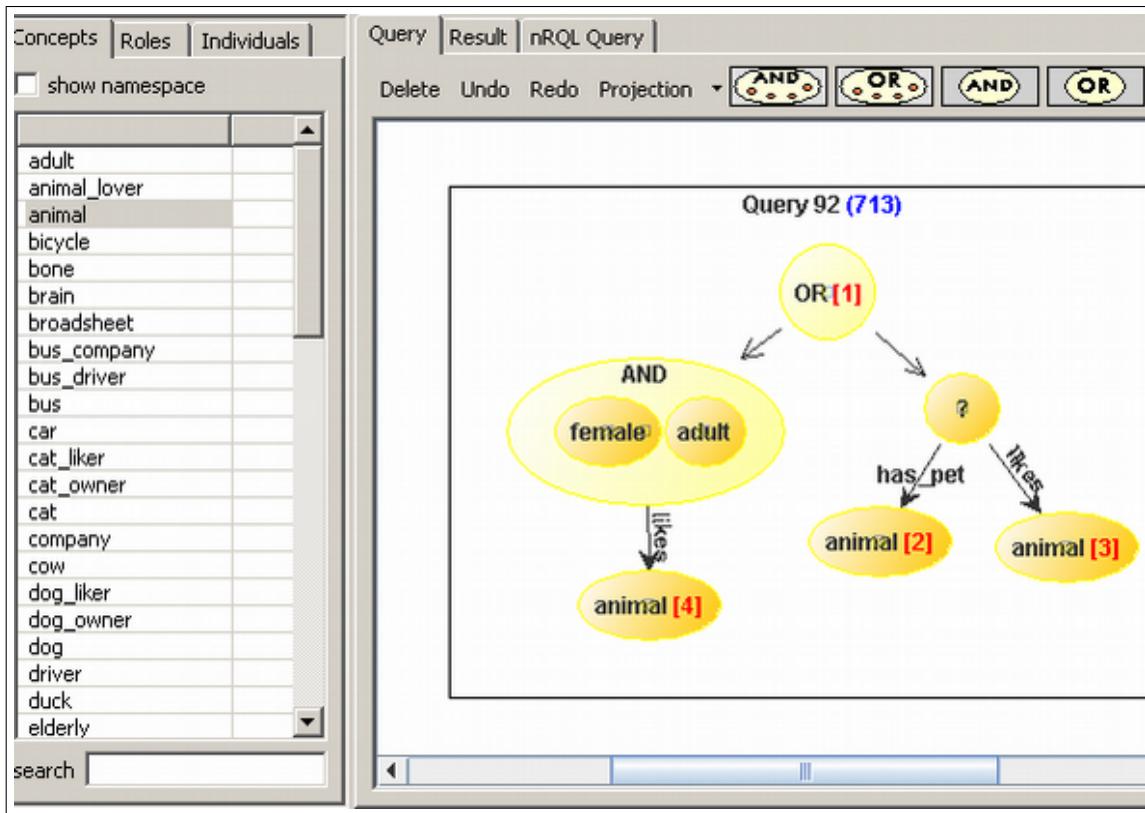


Figura 3.9: Consulta en OntoVQL [56]

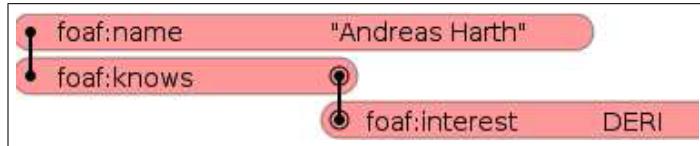


Figura 3.10: Consulta en [79]

Graphite [40] es un sistema para consultar redes sociales, representadas con grafos atribuidos. Graphite permite al usuario construir un patrón de consulta para encontrar subgrafos que coinciden de manera exacta o aproximada. La herramienta hace un ranking de los subgrafos que no coinciden exactamente con el patrón. En el patrón es posible asignar valores a los nodos, seleccionando el valor de una lista de valores posibles.

Harth et al. [79] proponen una representación gráfica para consultas sobre datos RDF [159] basada en la notación N3 [23]. Una consulta se formula con patrones formados por facetas (*facets*), tripletas que incluyen URIs, literales y variables. Las facetas se enlazan por joins sobre variables. Las variables se representan con un círculo, las variables de salida con dos círculos concéntricos, y los joins por lazos entre las facetas (Ver Figura 3.10).

Catarci et al. [37] describen el experimento de evaluación de usabilidad de un sistema que permite el acceso a datos de fuentes heterogéneas, por medio de consultas expresadas sobre una ontología. La formulación de la consulta se basa en una metáfora de árbol e incluye dos etapas: primero el usuario elige un *punto de entrada*, un concepto que será la raíz del árbol, luego el usuario construye el árbol de consulta por medio de operaciones para generalizar, especializar, agregar, quitar o reemplazar conceptos y agregar o quitar propiedades (Ver Figura 3.11). Además, el usuario puede agregar restricciones sobre los conceptos, para seleccionar ciertos valores. Los autores reportan que el proyecto siguió la metodología de diseño centrada en el usuario, e involucró a los usuarios desde las primeras etapas. Durante el desarrollo del proyecto aplicaron evaluaciones formativas y sumativas, entre ellas: una encuesta para obtener información base para analizar el contexto de la aplicación; una encuesta y una evaluación observacional sobre el primer y segundo prototipos; y finalmente, una encuesta, una evaluación observacional y un experimento controlado sobre el prototipo final y el producto terminado.

QGraph [27] es un lenguaje visual para consultar y actualizar grafos con tipos de clase. La consulta se especifica con un patrón que incluye nodos, arcos, condiciones sobre los valores de los atributos, condiciones globales y condiciones sobre el tipo de clase o relación. Las condiciones globales operan entre objetos y relaciones. Es posible indicar el número de ocurrencias de un tipo de nodo o arco en la respuesta asociando rangos a los nodos y arcos. Por ejemplo, la Figura 3.12 muestra la consulta “Películas de misterio con menos de tres actrices y sin premios Oscar”. De otra parte, el usuario

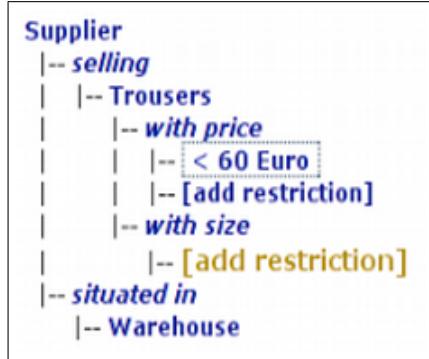


Figura 3.11: Filtro en [37]

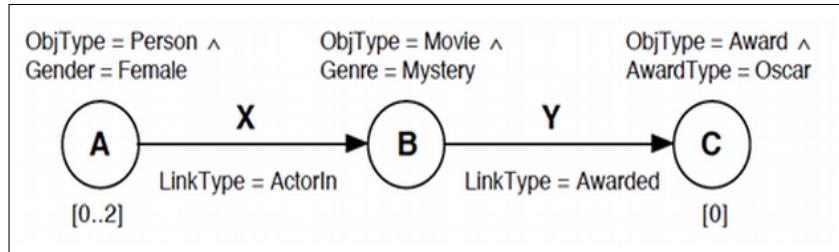


Figura 3.12: Consulta en QGraph [27]

debe señalar las variables de salida.

La propuesta de QUBLE [92] se centra en aprovechar los tiempos de latencia durante la formulación de la consulta para iniciar su ejecución. En la interfaz gráfica, la consulta se formula construyendo patrones, cuyos nodos se agregan a partir de una lista de las etiquetas y el usuario enlaza los nodos pintando arcos entre ellos. La herramienta fue evaluada por ocho participantes sin conocimiento en lenguajes de consulta sobre grafos.

3.2.3 Interfaces gráficas para lenguajes de consulta

Las interfaces gráficas para lenguajes de consulta de texto guían al usuario en la formulación de las consultas de manera que se diminuya el riesgo de incurrir en errores léxicos o sintácticos [180]. En esta sección se describen algunos de estos sistemas, en particular aquellas interfaces que han sido propuestas para apoyar la formulación de consultas en SPARQL [158].

Gruff (<http://franz.com/agraph/gruff/>) [1] es un sistema visual de consulta que ofrece un *browser* para apoyar la exploración de datos y una interfaz gráfica para la formulación de consultas SPARQL. La exploración se basa en la navegación del grafo instancia a partir de un nodo o un grupo de nodos que se seleccionan por su tipo, URI o etiqueta. Después se puede acceder a los nodos conectados con algún nodo en particular que esté en la vista, mediante menús que listan las conexiones ese nodo. El editor

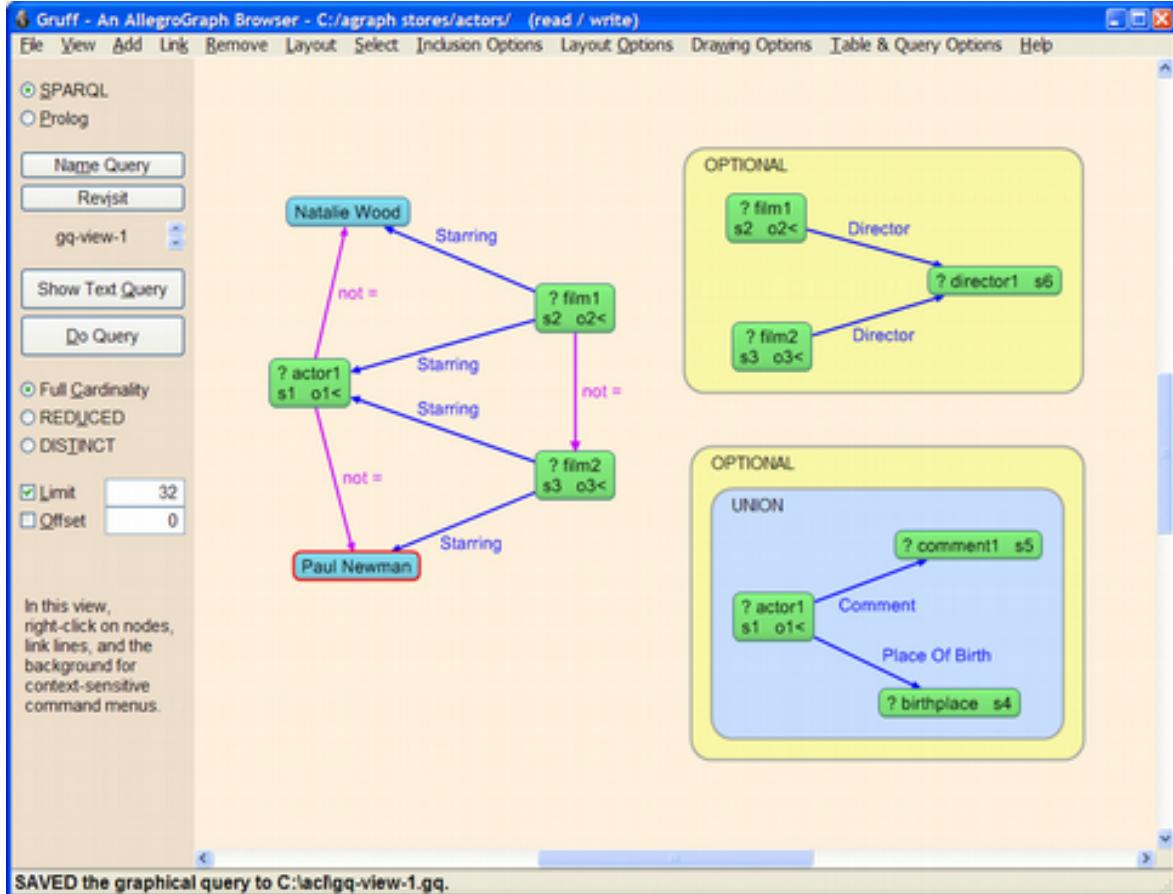


Figura 3.13: Consulta con filtro, *optional* y *union* en Gruff [1].

gráfico de consultas permite crear diagramas que representan patrones de búsqueda. Estos se crean agregando nodos y arcos. También es posible copiar nodos de la vista de navegación. Gruff permite especificar filtros sobre un nodo, aplicando operadores de comparación o verificando características como si es (o no) un IRI (*International Resource Identifier*) [?], un nodo blanco o un literal, además por el tipo del nodo. También permite definir filtros globales, de esta manera es posible incluir, por ejemplo, comparaciones entre los valores de dos variables o una expresión lógica compleja. Al agregar los arcos, permite usar una variable, agregar un filtro sobre el predicado (con operadores de comparación) o seleccionar de una lista uno de los predicados de la base de datos. Soporta agrupadores (*group graph patterns*) para operaciones *union*, *optional*, *minus*, *exists* y *not exists* (Ver Figura 3.13). Representa los agrupadores con rectángulos dentro de las cuales están dibujados otros patrones o agrupadores. Los resultados de las consultas se pueden ver en una tabla o en un diagrama de grafo en la vista de navegación.

NITELIGHT (interfaz) y vSPARQL (lenguaje visual) [166, 180] forman un editor

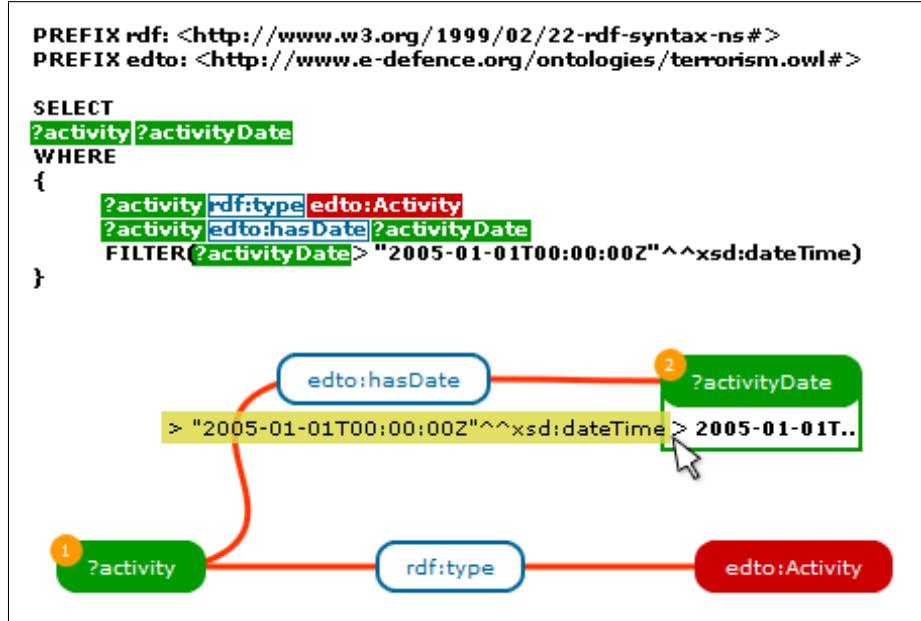


Figura 3.14: Consulta con filtro en NITELIGHT [180].

gráfico de consultas en SPARQL. El elemento básico son los patrones de tripla (*triple pattern*) que se representan con tres nodos (uno para la propiedad) y los arcos entre ellos. Los nodos representan las variables, las variables que se retornan en el resultado de la consulta, los URIs, los literales, los nodos blancos y las propiedades. Los grupos de patrones (*group graph patterns*), la unión (*union graph patterns*) y los patrones opcionales (*optional graph patterns*) se representan con rectángulos dentro de los que se pinta el patrón. El ordenamiento se marca con un triángulo dentro del nodo, cuya dirección indica si el ordenamiento es ascendente o descendente. Las condiciones de filtro se agregan con un *filter box* que se adhiere al nodo sobre el cual se aplica el filtro (Ver Figura 3.14). También soporta la cláusula *construct*, para lo cual se dibuja un patrón de construcción. El usuario construye el patrón con base en un navegador (*ontology browser*) que lista las clases y subclases de una ontología, con la opción de arrastrarlas y pegarlas (*drag and drop*) a la ventana de formulación de consultas. En esta ventana se tienen opciones para agregar los otros componentes de la consulta. Los autores advierten que la herramienta requiere conocimiento previo de SPARQL.

SparqlFilterFlow [71] esta orientado a usuarios sin experiencia en tecnologías semánticas. Usa el concepto de *filter pipes* [202]. El modelo *filter/flow* permite representar expresiones lógicas para filtrar datos, estas se visualizan como grafos dirigidos acíclicos donde los nodos tienen definidas condiciones de filtrado y los arcos marcan el flujo de los datos. Las conjunciones se modelan con flujos secuenciales, mientras que las disyunciones, con flujos paralelos (Ver Figura 3.15). Los usuarios componen las consultas agregando nodos y usando arrastrar-soltar para conectarlos.

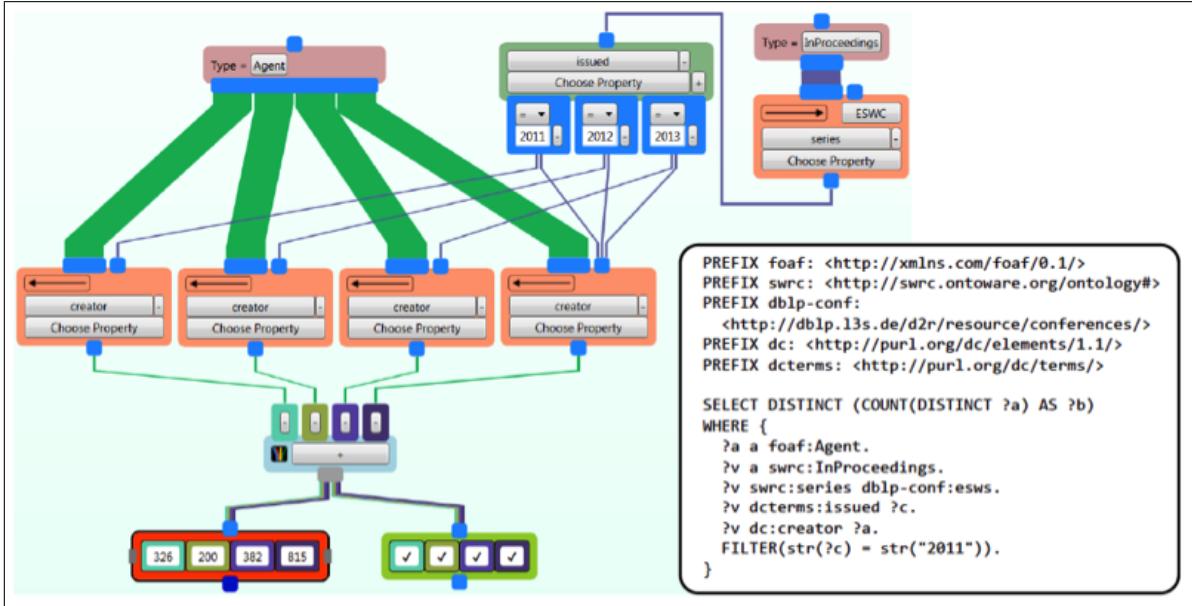


Figura 3.15: Consulta en SPARQLFilterFlow [71].

Los filtros incluyen comparación con IRIs y literales, la existencia de una propiedad o el tipo de clase. Los nodos filtro también permiten unir varios flujos. La herramienta fue evaluada por diez participantes, ocho de ellos profesionales y cinco de ellos con alguna familiaridad con la web semántica.

Groppe et al. [66] proponen un sistema visual de consulta que incluye la creación de consultas basadas en la navegación de los datos y un editor visual de SPARQL. Para la formulación de una consulta el sistema presenta una vista condensada de los datos en la que se agrupan las hojas que están conectadas a un mismo sujeto (agrupa los atributos del sujeto), los objetos que tienen un mismo sujeto y predicado y, los sujetos con los mismos predicados y objetos. Sobre esta vista, el usuario selecciona las clases que involucra la consulta, esta selección se copia en el editor de consultas y el resultado se despliega en una tabla que permite navegar sobre los datos. Sobre la tabla se puede renombrar columnas, ordenar, excluir columnas, aplicar filtros sobre los valores para excluir filas y hacer refinamiento de la consulta (Ver Figura 3.16). El refinamiento se hace agregando las tripletes que preceden un nodo o variable (aquellas en que el nodo es el destino) o las que lo suceden (aquellas en que el nodo es la fuente). Para este proceso el sistema propone posibles patrones que mezclan variables, URIs y literales. El editor visual de SPARQL permite construir patrones con una representación gráfica de los nodos y los arcos.

GQL [18] es un lenguaje visual que usa diagramas de clase de UML [138] como metáfora para formular consultas en SPARQL. Para ello representan en UML construcciones de un subconjunto de OWL-DL. GQL permite especificar grupos

Use prefixes

Prefixes:

pl: <http://www.p/>

Eliminate Duplicates

Result table:

rx301	rx311	rx330
<input type="button" value="Rename"/>	<input type="button" value="Rename"/>	<input type="button" value="Rename"/>
<input type="button" value="Sort"/>	<input type="button" value="Sort"/>	<input type="button" value="Sort"/>
<input type="button" value="Exclude"/>	<input type="button" value="Exclude"/>	<input type="button" value="Exclude"/>
<input type="button" value="Refine"/>	<input type="button" value="Refine"/>	<input type="button" value="Refine"/>
pl:Hilary_Swank <input type="button" value="Filter"/>	pl:Academy_Award_for_Best_Actress <input type="button" value="Filter"/>	pl:Buffy_the_Vampire_Slayer_film <input type="button" value="Fil"/>
pl:Hilary_Swank <input type="button" value="Filter"/>	pl:Academy_Award_for_Best_Actress <input type="button" value="Filter"/>	pl:The_Gift_film <input type="button" value="Fil"/>
pl:Hilary_Swank <input type="button" value="Filter"/>	pl:Golden_Globe_Award_for_Best_Actress_-_Nominations <input type="button" value="Filter"/>	pl:Buffy_the_Vampire_Slayer_film <input type="button" value="Fil"/>
pl:Hilary_Swank <input type="button" value="Filter"/>	pl:Golden_Globe_Award_for_Best_Actress_-_Nominations <input type="button" value="Filter"/>	pl:The_Gift_film <input type="button" value="Fil"/>
pl:Jack_Palance <input type="button" value="Filter"/>	pl:Academy_Award_for_Best_Supporting_Actor <input type="button" value="Filter"/>	pl:Batman_1989_film <input type="button" value="Fil"/>
pl:Jack_Palance <input type="button" value="Filter"/>	pl:Academy_Award_for_Best_Supporting_Actor <input type="button" value="Filter"/>	pl:Shane_film <input type="button" value="Fil"/>
pl:Jack_Palance <input type="button" value="Filter"/>	pl:Academy_Award_for_Best_Supporting_Actor <input type="button" value="Filter"/>	pl:Sudden_Fear <input type="button" value="Fil"/>

Figura 3.16: Navegación en [66]

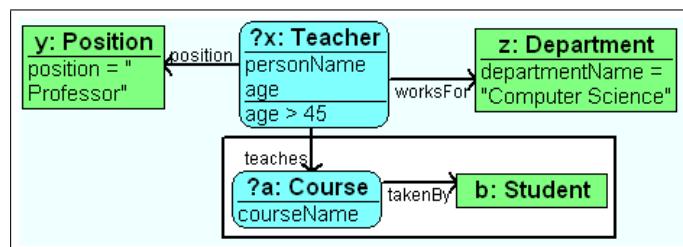


Figura 3.17: Consulta con opcional en GQL [18]

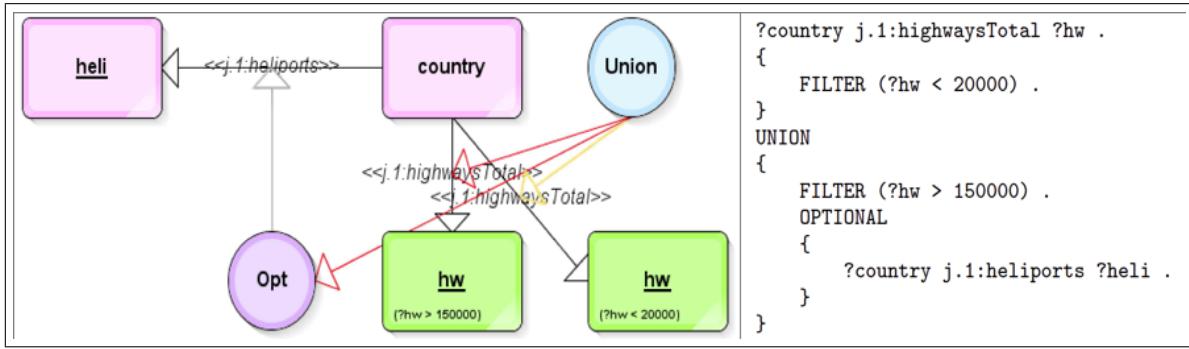


Figura 3.18: Consulta en RDF-GL [89].

opcionales (Ver Figura 3.17), grupos para la existencia (o no existencia) de subpatrones y, la intersección de conjuntos de nodos (se define para seleccionar los datos de un nodo, no es la intersección de resultados de una consulta).

RDF-GL [89] es un lenguaje visual para consultas sobre datos RDF, basado en SPARQL. Permite representar el patrón de consulta con un diagrama que incluye los nodos, arcos y algunas operaciones como *union* y *optional*. Usa códigos de color en los arcos para identificar cuando son parte de una tripla y cuando señalan los operandos de la *union* y el *optional* (Ver Figura 3.18). Usa las esquinas de los nodos para denotar las operaciones *distinct*, *order*, filtro sobre el nodo por rango de valores y, el tipo de consulta (ej. *Select*). Los autores reportan un experimento de usabilidad con un pequeño grupo de participantes con conocimiento de SPARQL.

3.2.4 Discusión

La Tabla 3.1 muestra un resumen de las características encontradas en los lenguajes visuales de consulta descritos en las secciones anteriores. Como es natural, la metáfora predominante en estos sistemas es el diagrama de grafo. Los sistemas revisados usan este diagrama con uno de los siguientes propósitos: (a) Mostrar y manipular los datos. Este es el caso de la mayoría de los sistemas enfocados en la exploración de los datos. (b) Mostrar una vista resumida de los datos, esta vista resulta de agrupar nodos por características o arcos incidentes comunes. Este enfoque se usa en un sistema de exploración (Shamir y Stolpnik [174]) y en una interfaz gráfica para SPARQL (Groppe et al. [66]). Sin embargo, el sistema de Groppe et al. [66] no usa la vista resumida para la formulación de las consultas. (c) Representar un patrón de consulta. Este es el caso de la mayoría de los sistemas enfocados en el lenguaje de consulta o en una interfaz gráfica para un lenguaje de texto. Algunos sistemas combinan metáforas (CGV [188], TouchGraph¹, Gephi [19], Groppe et al. [66]) o mecanismos de exploración (Gruff [1]).

¹<http://www.touchgraph.com/>

Tabla 3.1: Características encontradas en los Lenguajes Visuales de Consulta.

Sistema Visual de Consulta	Modelo de Datos	Enfoque	Metafora	Mecanismo de exploración para la selección de datos	Tipo de filtro	Filtros con operaciones lógicas (Y/O)	Manejo de datos incompletos	Subconsultas	Usa el Esquema de Datos	Cálcula un esquema	Evaluación de Usabilidad
Shamir y Stolpnik	Grafo con tipos de clase	Exploración y análisis	Diagrama de grafo	Grafo resumido y operaciones sobre nodos	Filtro: Local por tipo. Por conexión k-vecinos	No	Implicito	No	Tipo de nodos y arcos	Agrupan nodos en una estructura jerárquica	Una, sumativa
van den Elzen y van Wijk	Grafo con tipos de clase	Exploración	Diagrama de grafo	Lisas de tipos, seleccionar rangos de valores, marcar regiones (cuadros)	Filtro local por tipo y valor de atributos	No	Implicito	No	Tipo de nodos y arcos	No	No reportan
CGV	Grafo con tipos de clase	Exploración	Diferentes vistas, entre ellas el diagrama de grafo	Grafo de datos, agrupar y filtrar	Filtro local por tipo, valor de atributos, y características	Sí (notación visual)	Implicito	No	Tipo de nodos y arcos	No	Retroalimentación de usuarios
TouchGraph	Grafo con tipos de clase	Exploración	Diagrama de grafo y tabla coordinadas	Grafo de datos, filtros y particiones	Filtro: Local por tipo y valor de atributos. Por conexión k-vecinos	No	Implicito	No	Tipo de nodos y arcos	No	No reportan
Graph	Grafo atribuido	Exploración	Diagrama de grafo y tabla	Grafo de datos, filtros y particiones	Filtro: Local por tipo, valor de atributos, y características. Por conexión k-vecinos	Sí (árbol de expresión)	Implicito	No	Tipo de nodos y arcos	No	No reportan
Cytoscape	Grafo atribuido	Exploración	Diagrama de grafo	Grafo de datos y filtros	Lisitas para definir los filtros	No	Implicito	No	Tipo de nodos y arcos	No	No reportan
GlyphLink	Grafo con tipos de clase	Exploración	Diagrama de grafo, nodos agrupados	Grafo de datos, agrupar y filtrar	Arrastrar y soltar nodos, seleccionar tipos en listas	No	Implicito	No	Tipo de nodos y arcos	No	Una, sumativa
Explorator	RDF	Exploración	SPO, conjuntos	Definición de conjuntos	Filtros y SPO	No	Implicito	No	Sí	No	Una, preliminar, sumativa
OntoVis	Ontología	Exploración	Diagrama de grafo	Grafo de datos y filtros	No deseo	No	Implicito	No	No	No	No reportan
GraphLog	Grafo etiquetados GOOD (Grafo etiquetado)	Lenguaje Visual	Diagrama de grafo	Consultas (patrones)	No deseo	No	Implicito	No	No	No	No reportan
Gysens et al.			Diagrama de grafo	Consultas (patrones)	No deseo	No	Implicito	No	No	No	No reportan
MashQL	RDF	Lenguaje Visual	Árbol	Recorrer el esquema o la firma del grafo	Eleger de listas la raíz y las propiedades	Realizables por <i>union o join</i>	Sí, el usuario lo especifica	Sí	Si (tupla)	No	No reportan
SNOL	SNOM	Lenguaje	Diagrama de grafo	Consultas (patrones)	No se especifica	Patrón con filtro	Sí (condiciones especificadas por un predicado)	No	No	Si	No reportan

Sistema Visual de Consulta	Modelo de Datos	Enfoque	Metafora	Mecanismo de exploración	Mecanismos de interacción para la selección de datos	Tipo de filtro	Filtro con operaciones lógicas (Y/O)	Manejo de datos incompletos	Subconsultas	Usa el Esquema de Datos	Cálcula un esquema	Evaluación de Usabilidad
Sadamandan et al.	RDF	Lenguaje Visual	Diagrama de grafo	Consultas (patrones)	Dibujar el patrón, Seleccionar de listas	Patrón con filtro (literales en nodos)	No	No	Conceptos de la ontología	No	Una, sumativa	
Barzins et al.	RDF	Lenguaje Visual	Diagrama de grafo	Consultas (patrones)	Dibujar el patrón, Seleccionar de listas	Patrón con filtro	Si	No	Realizables por unión o intersección	No	No	No reportan
OntoQL	OWL-DL	Lenguaje Visual	Diagrama de operaciones	Consultas	Dibujar el diagrama	Patrón con filtro (literales en nodos)	No	Si	Tipo de nodos y arcos	No	No	Lo proponen como trabajo futuro
Graphite	Grafos atribuidos	Lenguaje Visual	Diagrama de grafo	Consultas (patrones)	Dibujar el patrón, Seleccionar de listas	Patrón con filtro (literales en nodos)	No	Parcial (matching aproximado)	No	No	No	No reportan
Harth et al.	RDF	Lenguaje Visual	Facetas (triplets)	Consultas (patrones)	No describir	Patrón con filtro (literales en nodos)	No	No	Si	No	No	No reportan
Catanei et al.	Ontología	Lenguaje Visual	Árbol	Consultas	No describir	Eleger una lista raíz, aplicar operaciones	Patrón con filtro	No específica	No	Si	Si	Varias durante el desarrollo
Graph	Grafo con tipos de clase	Lenguaje Visual	Diagrama de grafo	Consultas (patrones)	No describir	Patrón con filtro	No	No	Si	No	No	No reportan
QUBIE	Grafo etiquetado	Diagrama de grafo para un latencia en la formulación de las consultas	Diagrama de grafo	Consultas (patrones)	Dibujar el patrón, Seleccionar de listas	Patrón con filtro (literales en nodos)	No	No	No	No	No	Una, sumativa
Gruff	RDF	Interfaz gráfica lenguaje	Diagrama de grafo	Consultas (patrones) o Exploración del grafo a partir de nodos iniciales, agregando nodos adyacentes	Dibujar el patrón, Seleccionar de listas	Patrón con filtro	No	No	Si	Opcional	No	No reportan
NITELIGHT / vSPARQL	RDF	Interfaz gráfica para un lenguaje	Diagrama de grafo	Consultas (patrones)	Dibujar el patrón, Seleccionar de listas	Patrón con filtro sobre los nodos	Si	Si	Si	Opcional	No	Lo proponen como trabajo futuro
SparqlFilterFlow	RDF	Exploración, Interfaz gráfica para un lenguaje	Filterflow	Consultas	Dibujar el flujo	Filtro local por tipo y valor de atributos	Si	No	Si	No especifica	No	Una, sumativa
Gropp et al.	RDF	Interfaz gráfica para un lenguaje	Diagrama de grafo y tabla	Gráfico resumido, refinar la tabla	Refinar la tabla, dibujar un patrón	Filtro local por valor de atributos	No especifica	Implícito en las tablas. No se soporta en los patrones	No	Si (Vista resumida)	No	No reportan
GQL	OWL	Interfaz gráfica para un lenguaje	Diagrama de clases UML	Consultas	Dibujar el patrón, Seleccionar de listas	Patrón con filtro sobre los nodos	No especifica	No	Si	No	No	No reportan
RDF-GI	RDF	Diagrama de grafo lenguaje	Diagrama de grafo	Consultas (patrones)	Dibujar el patrón	Patrón con filtro sobre los nodos	No especifica	Si (cláusula optional)	No	No especifica	No	Una, sumativa

Los sistemas de exploración de datos que usan el grafo para mostrar y manipular los datos, aplican una de las siguientes formas de exploración: En la primera forma, la exploración inicia con la selección de uno o algunos nodos que se despliegan en un diagrama o en una tabla, y el usuario puede agregar nodos y arcos que están conectados a los ya seleccionados. En estos casos se usa interacción directa, el usuario marca sobre el grafo el(los) nodo(s) que va a explorar y agrega, generalmente a través de menús, los nodos vecinos. En la segunda forma, la exploración inicia con todos los nodos y arcos de la base de datos desplegados en un diagrama (referido en la tabla 3.1 como grafo de datos) y el usuario establece filtros sobre los nodos y/o los arcos para ocultar algunos datos. En estos casos se identifican dos formas de interacción: (a) Por manipulación directa del grafo, marcando los filtros sobre los nodos. (b) Sobre listas o tablas, desplegadas en ventanas adicionales, donde se pueden seleccionar los atributos y las operaciones que se van a aplicar sobre el grafo.

En general, los sistemas de exploración no usan la manipulación directa del grafo esquema en la formulación de las consultas, hacen manipulación del grafo de datos. Esto genera problemas por la cantidad de datos que se deben visualizar. Como una forma de solucionar este problema, Shamir y Stolpnik [174] proponen manipulación directa sobre un grafo que muestra una vista resumida de los datos. La manipulación permite al usuario ver los detalles en un punto particular. Sin embargo, el grafo no se usa para formular la consulta por manipulación directa de los patrones. Groppe et al. [66] también representan con el grafo una vista resumida de los datos, pero no hay interacción con esta vista, la exploración de los datos se hace a través de una metáfora de tabla y las consultas, dibujando el patrón de la consulta.

De otra parte, los grafos resumidos (calculados agrupando nodos en el grafo de datos) pueden presentar una alta dispersión, dependiendo de que tan incompletos sean los datos. Esto debido a que cuando hay datos incompletos, los objetos de la misma clase tendrán conjuntos de atributos y relaciones diferentes, lo cual los puede ubicar en grupos diferentes a pesar de que conceptualmente deberían pertenecer al mismo grupo. De la misma manera, objetos que pueden tener en común muchos tipos de relación y que conceptualmente pertenecen a clases diferentes podrían quedar dentro del mismo grupo si las relaciones que hacen la diferencia son precisamente aquellas que no están registradas.

Cuando el sistema usa formulación de consultas dibujando un patrón, el usuario debe hacer un proceso previo de exploración del grafo que le permita encontrar la forma como están estructurados los datos, de manera que pueda formar tripletas correctas (ej. por la dirección del arco) y patrones correctos (ej. por el orden en que se conectan las tripletas para formar un patrón de camino que exista en los datos). Este proceso puede resultar especialmente engorroso en bases de datos con datos incompletos, porque cuando el usuario explora un objeto de cierta clase probablemente no encuentre en él todas las propiedades y relaciones que pueden tener los objetos de esa clase. Además, pocos sistemas incluyen herramientas para explorar el grafo de datos, entre ellos Gruff

[1]. De otra parte, los sistemas que no describen el mecanismo de interacción no usan el grafo esquema como parte de la definición del lenguaje, a excepción de SNQL [170].

Adicionalmente, solo aquellos sistemas que se basan en una ontología tienen un modelo de los datos en el nivel conceptual. Sin embargo, cuando la ontología representa el conocimiento de un dominio de aplicación, el tamaño del grafo tiende a ser mucho mayor que el tamaño de un grafo que representa un modelo conceptual de la organización de los datos en la base de datos. Este último solo incluye una vista genérica de los datos que reposan en un sistema de información, a pesar de que los datos de estos sistemas puedan ser alimentados con los datos de una ontología. Por ejemplo, en el dominio médico las aplicaciones para administración de historias clínicas usan diferentes vocabularios y ontologías como ICD¹, para la descripción de los diagnósticos, o CPT², para la descripción de los procedimientos. En un modelo de datos los diagnósticos, o los procedimientos, son una clase con un conjunto de propiedades. El modelo de datos no incluye la jerarquía completa de las enfermedades, o de los procedimientos, como si lo hace la ontología.

En cuanto a las opciones para filtrar los datos se encontraron filtros con diferentes niveles de expresividad, estos se categorizaron de la siguiente forma:

- Filtro local: son los filtros que permiten ocultar de la vista los nodos o arcos que no satisfacen condiciones especificadas sobre sus atributos, su tipo o sus características (ej. grado, tener cierto tipo de arco incidente).
- Filtro por conexión: estos incluyen los mecanismos que permiten seleccionar nodos que están conectados a otro nodo que cumple ciertas condiciones. Este filtro tiene a su vez dos tipos:
 - K-vecinos: algunas herramientas permiten especificar un filtro local sobre nodos y arcos y luego hallar los k-vecinos de uno o varios nodos específicos que cumplen las condiciones. Los k-vecinos están conectados por arcos que cumplen algunas condiciones, pero no se especifica un camino como patrón.
 - Patrón con filtro: un patrón de tripletas especifica que camino o caminos conectan uno o varios nodos a seleccionar con nodos que cumplen una condición específica. Por ejemplo, los patrones y filtros en SPARQL.

Entre las categorías de filtro mencionadas, los patrones con filtro son los que proveen mayor expresividad, ya que permiten recuperar objetos que se relacionan con otros que tienen cierta condición. Por ejemplo “Pacientes que han tenido citas médicas donde se les diagnosticó diabetes” (incluye pacientes-citas-diagnósticos). También permiten recuperar objetos que a través de diversos caminos se conectan con nodos con condiciones diferentes o incluir en la consulta comparaciones entre variables, por ejemplo “Pacientes a los que se practicó una colectomía después de 30 años de edad” ($\text{fechaProcedimiento} \geq (\text{fechaNacimiento} + 30\text{años})$).

¹<http://apps.who.int/classifications/icd10/browse/2015/en>

²<http://www.ama-assn.org/go/cpt>

En los filtros locales las condiciones se aplican sobre los atributos del objeto, y cuando a estos se añade la búsqueda de los k-vecinos se tienen dos restricciones: primero no se especifica un camino de conexión particular, solamente se puede elegir los tipos de arco que se permiten en esos caminos, pero no el orden en que pueden aparecer; y segundo, los vecinos se buscan a partir de nodos particulares y no hay varias condiciones activas simultáneamente.

En general, los sistemas enfocados en la exploración de datos ofrecen filtros locales, en algunos casos complementados con la posibilidad de incluir los k-vecinos de los nodos seleccionados, mientras que los sistemas enfocados en el lenguaje o en la interfaz gráfica de un lenguaje, ofrecen una mayor expresividad a través de patrones con filtro.

Cuando los filtros incluyen conjunción o disyunción de condiciones, se ofrecen dos formas de capturar estas expresiones: mediante en un árbol de operaciones o eligiendo una sola operación, conjunción o disyunción, para combinar todas las condiciones. Las representaciones en árbol tienen mayor expresividad, y en algunos casos, por ejemplo en OntoVQL y en CGV, usan una notación visual. Sin embargo, la notación visual parece ayudar poco, dado que lo más importante en estos casos es la semántica de la expresión, que obliga al usuario a entender el sentido de la composición de las operaciones.

En la Tabla 3.1, el manejo de datos incompletos se catalogó como “Implícito” cuando los filtros son locales y no se incluyen patrones en las consultas, puesto que en estos casos los datos incompletos no afectan la respuesta. La respuesta de los filtros basados en patrones se puede afectar por datos incompletos, razón por la cual se incluye en los lenguajes cláusulas para marcar como opcional partes del patrón. En los sistemas enfocados en el lenguaje o en la interfaz gráfica de un lenguaje que permiten usar la cláusula *optional* corresponde al usuario decidir sobre el correcto uso de la misma. Aunque este es un concepto aparentemente sencillo, los resultados de las consultas varían dependiendo de cómo se agrupen los patrones dentro de los grupos opcionales. Por ejemplo, en SPARQL, son diferentes los resultados que arrojan estos dos patrones:

```
{ ?Patient :id ?patId OPTIONAL { ?Patient :encounter ?Enc .  
?Enc :date ?date . ?Enc :diagnosis ?diag } }
```

y

```
{ ?Patient :id ?patId OPTIONAL { ?Patient :encounter ?Enc }  
OPTIONAL { ?Enc :date ?date } OPTIONAL { ?Enc :diagnosis ?diag } }
```

Si a estas variaciones en el agrupamiento se suman los filtros, se pueden encontrar muchas consultas que arrojan resultados diferentes y cuya expresión varía por pequeñas decisiones que el usuario toma al momento de dibujar el patrón de consulta.

De otra parte, las interfaces gráficas de lenguajes de consulta usualmente mapean los elementos del lenguaje a una representación gráfica, es decir que la semántica de los elementos se mantiene aunque varia su representación, por tanto el usuario debe aprender el lenguaje subyacente para formular las consultas.

En cuanto al enfoque de diseño centrado en el usuario, solo el 55.5% de los trabajos referidos en este documento reporta el uso de una metodología para el desarrollo de un

sistema con características de usabilidad o la realización de pruebas de usabilidad. El 7.4% lo proponen como trabajo futuro. En el 29,6% de los trabajos reportan evaluaciones sumativas, realizadas al final del desarrollo de la herramienta. En general estas evaluaciones miden el número de tareas realizadas correctamente, el tiempo requerido para realizar las tareas y la percepción de los participantes con respecto a algunas características de usabilidad. Algunos realizaron pruebas comparativas en las que analizaron el tiempo requerido para realizar las tareas o los resultados de las mismas. Finalmente, dos trabajos (el 7.4%) reportan que los usuarios se involucraron durante todo el ciclo de desarrollo del proyecto, sin embargo, en uno de ellos no se realizaron pruebas de usabilidad.

En síntesis, la mayoría de sistemas de exploración de datos se enfocan en la visualización y navegación de los grafos, presentan para ello propuestas novedosas, estéticas y útiles, donde se maneja el color, el tamaño y la forma para describir diferentes características de los datos. También permiten la manipulación directa del grafo y el usuario obtiene realimentación en cada paso que aplica. Sin embargo, el grafo que manipula no es una representación de nivel conceptual de los datos y la expresividad de los filtros para formular las consultas está limitada a filtros locales, a los que se agregan k-vecinos, y solamente en algunos casos soportan operadores lógicos (y/o). De otra parte, los sistemas enfocados en el lenguaje y las interfaces gráficas de lenguajes proveen mayor expresividad en los filtros ya que incluyen patrones con filtro, operaciones lógicas (y/o) y en algunos casos filtros globales y otras operaciones como las cláusulas *optional* y *union*. Sin embargo, generalmente no hay manipulación directa del grafo; los patrones se deben construir desde cero a partir de listas de objetos y propiedades; requieren que el usuario haga una exploración del grafo de datos para conocer cómo están organizados los datos, lo cual puede resultar engorroso especialmente cuando hay datos incompletos; el usuario no tiene realimentación durante la formulación de la consulta; el usuario debe entender el lenguaje pues las construcciones del mismo se trasladan a una notación visual; y pequeñas diferencias en la aplicación de los agrupadores para cláusulas como *optional* y *union*, y la ubicación de los filtros dentro de esos agrupadores puede generar grandes variaciones en el resultado de la consulta. Finalmente solamente los sistemas para consulta de ontologías funcionan sobre una representación de nivel conceptual.

A partir del análisis expuesto, se definió un conjunto de características deseables en un lenguaje visual de consulta. Estas características se incluyeron en el diseño e implementación de GraphTQL. La Tabla 3.2 muestra estas características en las filas y en las columnas, los lenguajes de consulta estudiados, la última columna corresponde a GraphTQL. La intersección característica-lenguaje se marca con **✓** si el lenguaje incluye la característica, con **x** cuando el lenguaje no ofrece la característica, con **NA** cuando la característica no aplica para ese tipo de herramienta o forma de consulta, o con **ND** cuando, en la literatura revisada, no se encuentra información sobre la característica. En la identificación de las características, se consideró que el lenguaje

representa los datos a nivel conceptual cuando está basado en una ontología, y que las interfaces gráficas para lenguajes textuales están formalmente definidas si definen de manera explícita la representación visual de los elementos del lenguaje. Otras características incluidas en el diseño de GraphTQL, tal como el ofrecer la historia de las interacciones y permitir al usuario navegar sobre ella o la realimentación que se ofrece al usuario, no se incluyeron en la Tabla 3.2 porque muy pocos artículos ofrecen información al respecto. Para el caso particular de la historia de las interacciones, solamente CGV [188] destaca su importancia.

Tabla 3.2: Características deseables, incluidas en los Lenguajes Visuales de Consulta.

Característica	Sistema visual de consulta	Shamir y Stolnik van den Elzen y van Wijk	Touch Graph CGV	Gephi	Cytoscape	Glyph Link	Explorator	Onto Vis	Graph Log	Gyssens et al.	MashQL	SNQL	Sadanand an et al.
Representación conceptual de los datos	x	x	x	x	x	x	✓	x	x	x	x	x	✓
Expresiones de filtro incluyen conectores lógicos	x	x	✓	x	✓	x	x	x	x	✓	✓	✓	x
Filtros no locales	x	x	x	x	x	x	x	x	✓	✓	✓	✓	✓
La semántica de los operadores incluye manejo implícito de datos incompletos	NA	NA	NA	NA	NA	NA	x	NA	x	x	x	x	x
Evita nociones de lenguajes de consulta (ej. variables, variables de salida)	NA	NA	NA	NA	NA	NA	✓	NA	x	x	x	x	x
Guía la formulación de condiciones de filtro	x	x	x	x	x	x	x	x	x	ND	x	ND	x
Guía la selección de caminos	x	x	x	x	x	x	x	x	x	ND	✓	ND	x
Reporta aplicación de técnicas UCD	x	x	x	x	x	x	x	x	x	x	x	x	x
Reporta pruebas con usuarios, o retroalimentación	✓	x	✓	x	x	x	✓	✓	x	x	✓	x	✓
Evita explorar el esquema	x	x	x	x	x	x	✓	x	✓	x	✓	ND	x
Ofrece manipulación directa del gráfico	✓	✓	✓	✓	x	✓	x	✓	x	ND	ND	x	ND
Evita construir un patrón desde cero o a partir de listas	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	x	x	ND
Los operadores están formalmente definidos	✓	✓	ND	ND	ND	ND	✓	ND	✓	✓	✓	✓	x
Usa grafos simples	✓	✓	x	✓	x	x	✓	✓	✓	✓	✓	✓	✓
Se enfoca en la formulación de consultas (no en exploración y análisis)	x	x	x	x	x	x	x	x	x	x	x	x	x

Característica	Sistema visual de consulta	Barzdins et al.	Onto VQL	Graphite	Harth et al.	Catarci et al.	Qgraph	QUBLE	Gruff	NITE LIGHT/vSPARQL	Sparql Filter	Groppe et al.	RDFe-GQL	GraphiTQL
Representación conceptual de los datos	✓	✓	x	x	✓	x	x	x	x	x	x	✓	x	✓
Expresiones de filtro incluyen conectores lógicos	✓	✓	x	x	ND	x	✓	✓	✓	✓	✓	ND	ND	✓
Filtros no locales	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	x	✓	✓	✓
La semántica de los operadores incluye manejo implícito de datos incompletos	x	x	x	x	x	x	x	x	x	x	x	x	x	✓
Evita nociones de lengajes de consulta (ej. variables, variables de salida)	✓	x	✓	✓	✓	x	x	x	x	x	✓	x	x	✓
Guía la formulación de condiciones de filtro	x	x	ND	x	x	ND	x	x	x	x	x	x	x	✓
Guía la selección de caminos	✓	x	ND	ND	x	x	x	x	x	x	x	x	x	✓
Reporta aplicación de técnicas UCD	x	x	x	x	✓	x	x	x	x	x	x	x	x	✓
Reporta pruebas con usuarios, o retroalimentación	x	x	x	✓	x	✓	x	x	✓	x	x	✓	x	✓
Evita explorar el esquema	✓	x	x	✓	✓	ND	ND	x	x	✓	x	x	x	✓
Ofrece manipulación directa del grafo	x	x	x	ND	x	ND	x	x	x	x	x	x	x	✓
Evita construir un patrón desde cero o a partir de listas	x	x	x	x	x	x	x	x	x	✓	x	x	x	✓
Los operadores están formalmente definidos	x	✓	ND	ND	ND	ND	ND	ND	ND	✓	ND	x	x	✓
Usa grafos simples	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Se enfoca en la formulación de consultas (no en exploración y análisis)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Parte II

El Lenguaje Propuesto

Capítulo 4

GraphTQL

Las diversas aplicaciones de los modelos de datos basados en grafos ha motivado, en los últimos años, el estudio de lenguajes de consulta sobre grafos [7, 119]. Entre estos lenguajes están los visuales, algunos de los cuales se enfocan en facilitar a los usuarios finales la formulación de consultas. Los lenguajes visuales de consulta sobre grafos se pueden clasificar en dos tipos (Sección 3.2): las herramientas de exploración y análisis de grafos y las herramientas que se enfocan en el lenguaje de consulta. Las primeras, las herramientas de exploración y análisis de grafos, tienen generalmente menor expresividad en las consultas que las segundas, pero ofrecen manipulación directa del grafo instancia y operaciones para analizar propiedades del grafo (ej. centralidad). Las segundas, las herramientas que se enfocan en el lenguaje de consulta tienen mayor expresividad pero, usualmente, trasladan a una notación visual los conceptos de los lenguajes de consulta, que operan en el nivel lógico. Por tanto, los usuarios necesitan aprender esos conceptos y además, construir desde cero el patrón de consulta a partir de listas de objetos y propiedades.

GraphTQL, el lenguaje que se propone en esta tesis, es un lenguaje visual de consulta sobre grafos que se ubica en un punto intermedio entre los dos enfoques, brindando mayor expresividad que las herramientas de exploración, pero manteniendo algunas características de estas como la manipulación directa y la realimentación. Generalmente, los usuarios requieren tomar un subconjunto de los datos antes de hacer análisis sobre ellos. Por tanto, previo al uso de las herramientas de análisis es necesario hacer un preprocesamiento de los datos [101]. Este preprocesamiento se realiza formulando consultas de mediana complejidad, soportadas usualmente por los lenguajes de consulta, pero no por las herramientas de análisis y exploración de datos.

En este capítulo se describe GraphTQL y las principales características de su diseño. La Sección 4.1 presenta el esquema de referencia para los ejemplos de este y de los siguientes capítulos. La Sección 4.2 introduce el mecanismo de interacción de GraphTQL y describe su interfaz. En la Sección 4.3 se describen, de manera informal, los operadores de transformación de los grafos esquema e instancia, que se definen

formalmente en el Capítulo 5. La Sección 4.4 presenta los diálogos de clarificación de filtros que guían al usuario en la formulación de las consultas. Las características de diseño sobresalientes de GraphTQL se relacionan en la Sección 4.5. Finalmente, en la Sección 4.6 se discuten las alternativas que se consideraron en algunas decisiones de diseño y ciertas limitaciones en la expresividad de los filtros de GraphTQL.

4.1 Ejemplo de referencia

El grafo esquema que se usa como ejemplo de referencia en este documento se muestra en la Figura 4.1. Este modelo presenta una vista conceptual de un conjunto de datos clínicos y se diseñó con base en RIM (*HL7 Reference Information Model*), el modelo de referencia del estándar HL7 (*Health Level Seven International*, <http://www.hl7.org>) y en el modelo de información de OpenEHR [21] (<http://www.openehr.org>). De estos modelos se tomó un subconjunto de datos representativo que incluye el paciente, con su identificación y sus datos demográficos (fecha de nacimiento, género, dirección de residencia); el médico, con sus datos personales y su especialidad; los encuentros entre médicos y pacientes, que pueden ser de varios tipos (*ambulatory*, *inpatient*, *emergency*, *home*) y tienen una fecha. En estos encuentros se registran los datos relacionados con la atención prestada al paciente, entre ellos, los síntomas que presenta el paciente, la revisión por sistemas (de la que se registra el sistema y la observación), los diagnósticos, la historia familiar, las órdenes médicas y los procedimientos, pruebas y estudios de imágenes diagnósticas que se realizan al paciente durante el encuentro.

Un ejemplo de un grafo instancia correspondiente con el grafo esquema de la Figura 4.1 se muestra en la Figura 4.2. En el extremo superior izquierdo de esta figura, se observa que un paciente (*Patient 1*) de nombre *John*, apellido *Smith* e identificación *001*, ha tenido un encuentro (*Encounter 4*) con el doctor (*Physician 2*) de apellido *Jones* y este encuentro fue ambulatorio (*encounterType = Ambulatory*). Durante ese encuentro, el médico registró la realización de una prueba (*Colonoscopy*) cuyo resultado es *Doc1*; también registró un diagnóstico (*Diagnosis 1*) de código *C18.7* y descripción *Sigmoid Colon*. El resto de la imagen muestra información de los encuentros de éste y otros pacientes con otros médicos.

4.2 Mecanismo de interacción

La base de la interacción con el usuario en GraphTQL es la manipulación directa del grafo esquema. La idea subyacente es la transformación de los grafos esquema y de datos, para reducirlos hasta seleccionar la información que el usuario requiere. Esta transformación se logra por la aplicación sucesiva de operadores.

La interfaz de GraphTQL, que se muestra en la Figura 4.3, es simple y consta de dos partes. El marco del lado izquierdo, la ventana principal, presenta el diagrama del

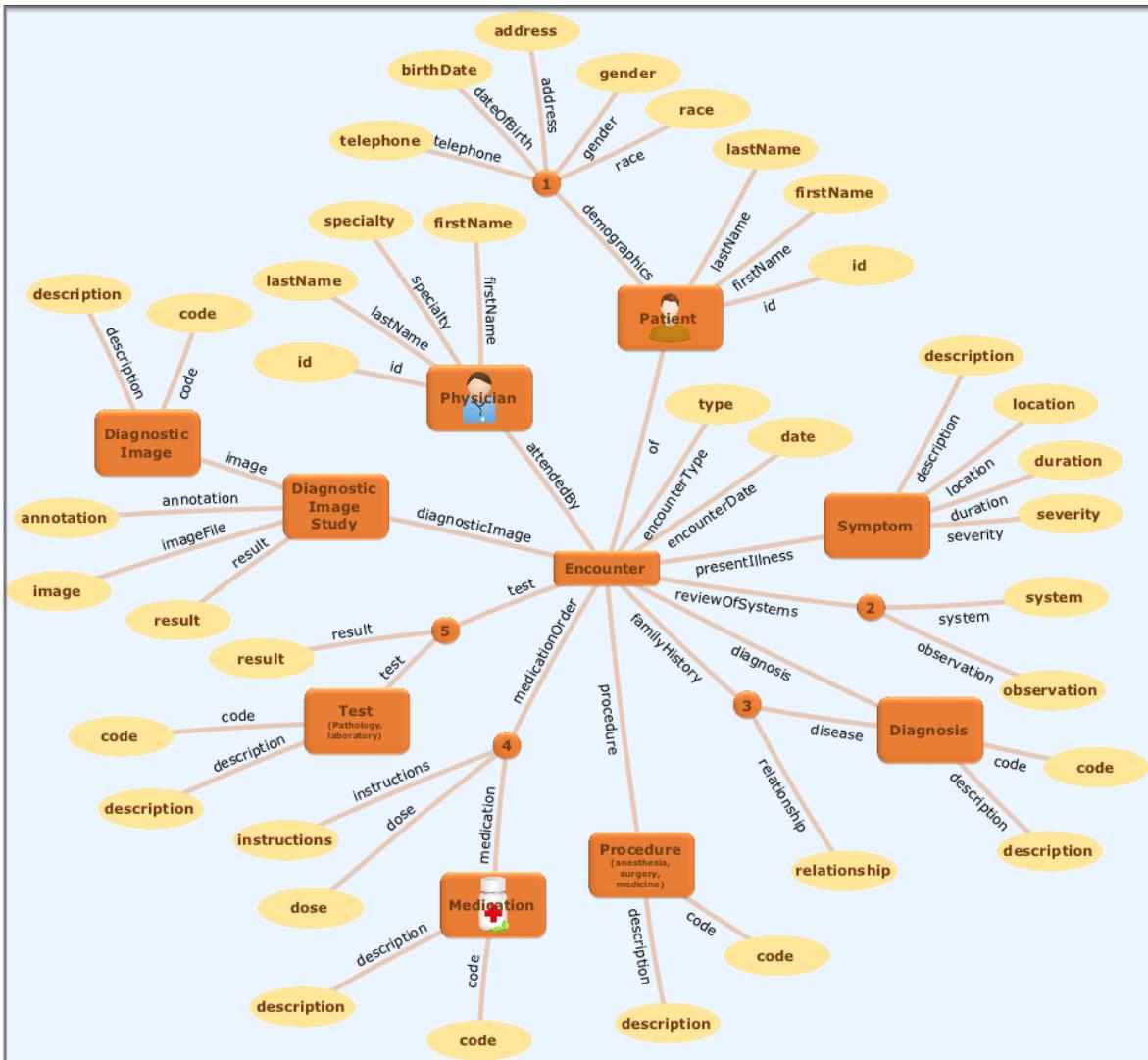


Figura 4.1: Ejemplo de referencia: Datos clínicos (grafo esquema).

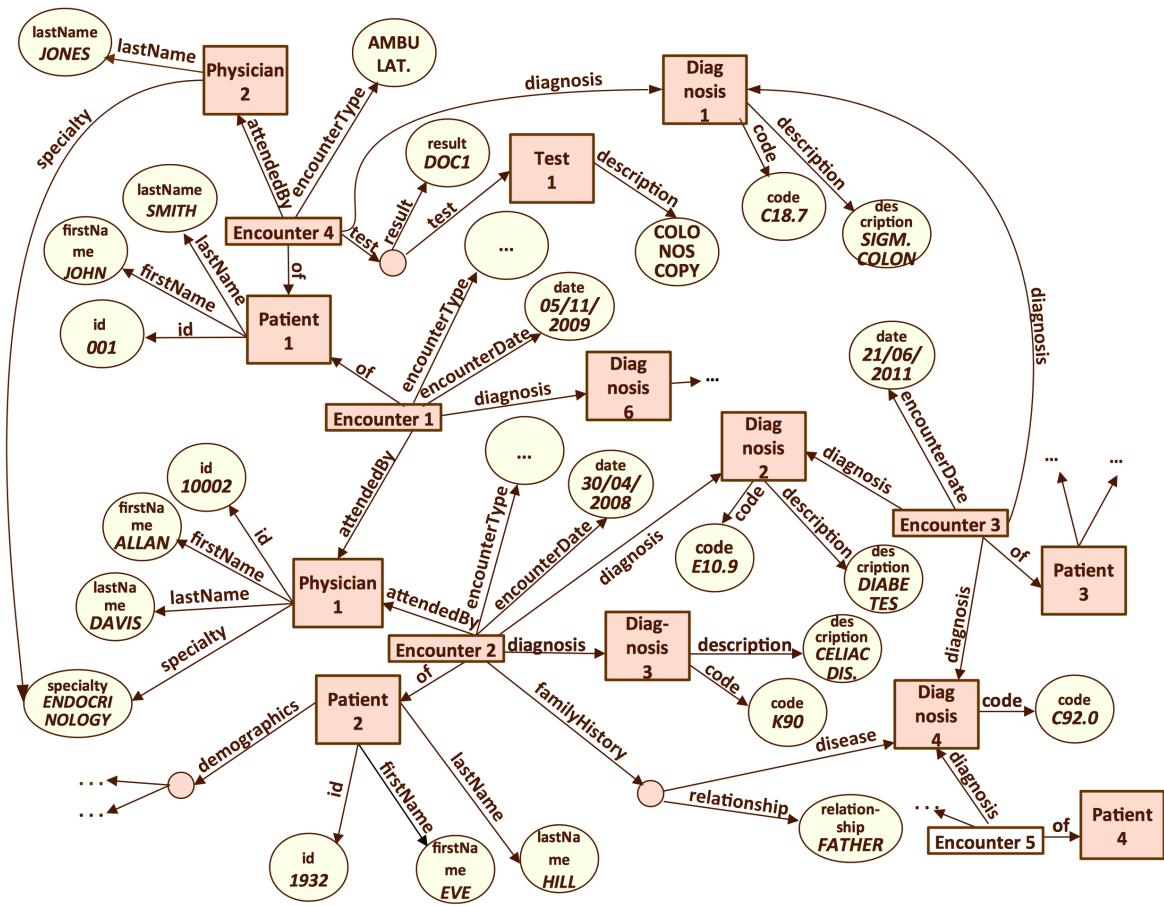


Figura 4.2: Ejemplo de referencia: Datos clínicos (grafo instancia).

grafo esquema, que muestra una vista del modelo conceptual de los datos. El marco del lado derecho contiene los comandos, organizados a su vez en cuatro marcos. El primero, la vista de referencia, facilita la ubicación y el desplazamiento en el grafo. Esta vista permite desplazar la ventana principal, arrastrándola mientras se presiona el botón izquierdo del ratón. El segundo, el marco de operadores, ofrece los botones para aplicar los cuatro operadores que ofrece GraphTQL: *Select a Portion*, *Filter*, *Filter+Additional Data* y *Put Two Objects Closer*. La entrada de cada uno de los operadores se marca sobre el grafo esquema. Los operadores se describen, de manera informal, mas adelante en esta sección. El tercero, el marco de control, incluye la ayuda, el *zoom* del grafo esquema, la ejecución de la consulta y tres opciones para deshacer acciones aplicadas. Estas últimas incluyen limpiar marcas (*Clean marks*), retroceder una operación (flecha azul) y retroceder al inicio del historial (flecha circular roja). Una consulta se formula aplicando sucesivamente varios operadores. Los operadores

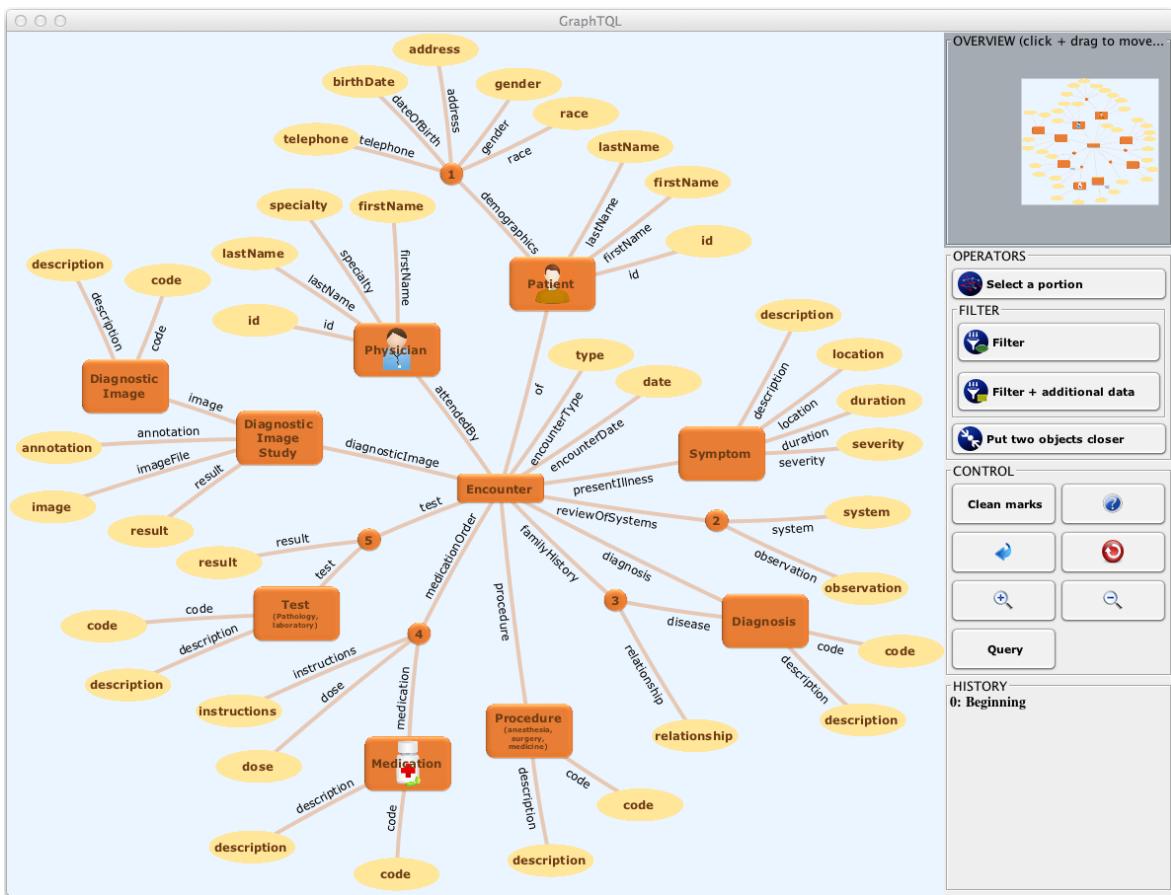


Figura 4.3: Interfaz de GraphTQL.

aplicados aparecen en el último marco, el historial.

La vista del modelo conceptual presenta los nodos con diferentes formas según su tipo. En esta vista se identifican cuatro tipos de nodo: los objetos, que se representan con rectángulos grandes (ej. *patient*, *physician*, *diagnosis*); los de valor básico (ej. *id*, *firstName*, *code*), que se representan con óvalos; los de valor compuesto etiquetados, que representan relaciones entre objetos y se dibujan con rectángulos pequeños (ej. *encounter*); y los nodos de valor compuesto, que agrupan atributos y se dibujan con círculos pequeños (ej. el nodo “1” agrupa los datos demográficos del paciente). La interfaz permite configurar el grafo agregando una imagen a los nodos tipo objeto (ej. *patient*).

Puede considerarse que el grafo esquema genera sobrecarga de información, lo cual coincide con la primera impresión de los participantes en las pruebas de usabilidad (Capítulo 6). Sin embargo, los participantes manifestaron que una vez empiezan a manipular el grafo éste se puede leer y seguir fácilmente. Esto puede ser debido a

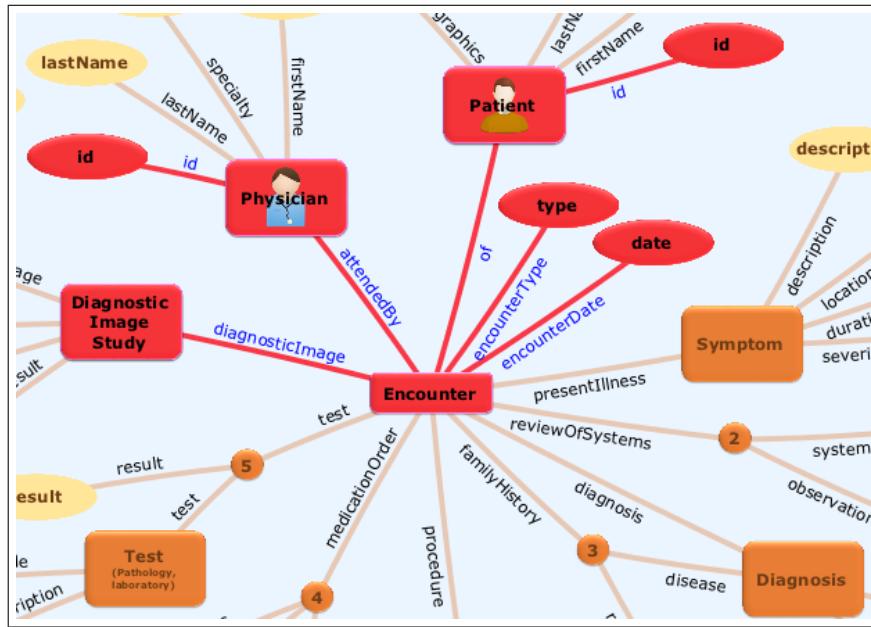
que ellos son expertos en el dominio de la aplicación y, en muchos casos, son quienes registran los datos en los sistemas de información que subyacen al modelo conceptual, por tanto conocen bien los datos almacenados. Es importante notar que, generalmente, los dominios de aplicación en los que se enfoca este trabajo tienen modelos de datos complejos y extensos, por lo cual su representación, gráfica o no, es difícil de presentar en una pantalla de forma amigable. Una posible solución para aliviar la sobrecarga de información en el esquema es manejar una jerarquía en la visualización de tal forma que en primera instancia se vean los nodos objeto, los nodos de valor compuesto etiquetados y sus relaciones, y en la medida en que se navega sobre el grafo, aparezcan los nodos de valor básico y los nodos de valor compuesto no etiquetados. Esto se propone como trabajo futuro en la Sección 8.3.

Antes de aplicar un operador se seleccionan los nodos y arcos que involucra la operación y se especifican las condiciones para filtrar los datos. Para simplificar este proceso, la interacción usa solamente el clic (izquierdo), el clic derecho, diálogos y notación visual.

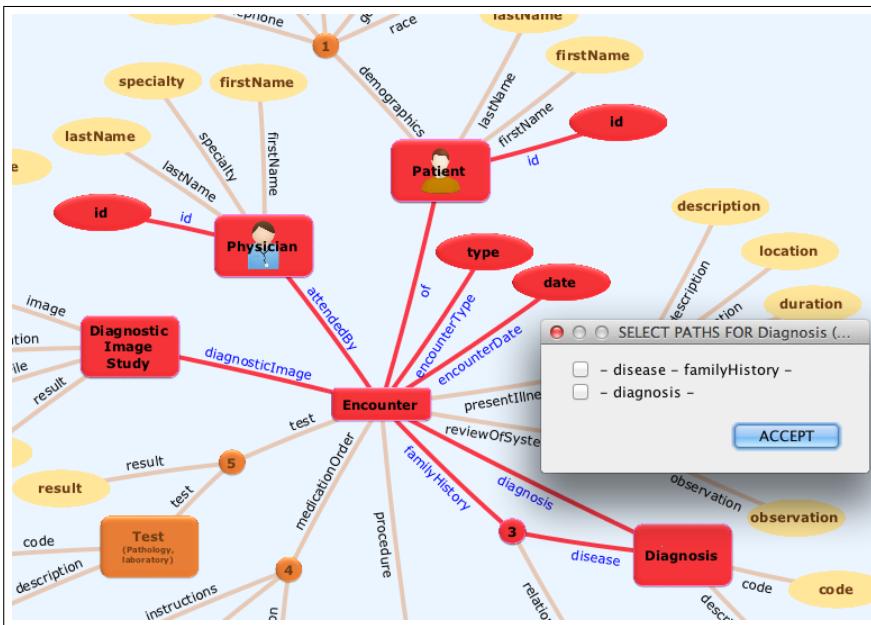
Un nodo, o un arco, se marca haciendo clic sobre él. Los nodos y arcos marcados se pintan de color rojo. Un clic sobre un nodo marcado quita la marca. Cada vez que se marca un nodo, la herramienta se encarga de encontrar y marcar el camino que lo conecta con los nodos previamente marcados. Cuando hay más de un camino, se despliega un diálogo de clarificación que presenta todos los caminos posibles para que el usuario elija uno o varios de ellos. Por ejemplo, en la Figura 4.4, se marca el nodo *Diagnosis* agregándolo a los nodos ya marcados en 4.4a. Dado que varios caminos lo conectan con ellos, se presenta al usuario el diálogo que se muestra en 4.4b. Cuando se despliega el diálogo se marcan todos los caminos en el grafo para hacerlos evidentes, una vez el usuario selecciona los caminos se quitan las marcas sobrantes.

En los operadores de filtro se requieren dos tipos de marca particular, cuya función se explica mas adelante: la clase de interés y las condiciones de filtro. Ambos se marcan con clic derecho. La clase de interés se pinta de color verde y en las condiciones de filtro, la condición se escribe bajo la etiqueta del nodo. Solamente los nodos de valor básico pueden tener definidas condiciones de filtro, además estos nodos también pueden ser objeto de interés. Por esta razón, el clic derecho sobre un nodo de valor básico hace que se despliegue una ventana de diálogo en la cual el usuario marca la función que va a asignar al nodo en la consulta y, si elige filtro, especifica la condición (Ver Figura 4.5). En el diagrama del grafo, los óvalos representan los nodos de valor básico, de esta manera el usuario puede distinguir fácilmente sobre cuales nodos especificar las condiciones de los filtros.

Con el fin de simplificar la interfaz, los operadores de filtro requieren que el usuario marque solamente la clase de interés y las condiciones. Sin embargo, la sencillez de esta representación genera dos problemas: primero, limita la expresividad de los filtros y segundo, cuando se necesita especificar más de una condición, la representación gráfica es ambigua y por tanto, se hace necesario especificar si se trata de una conjunción o



(a)



(b)

Figura 4.4: Diálogo de clarificación de caminos.

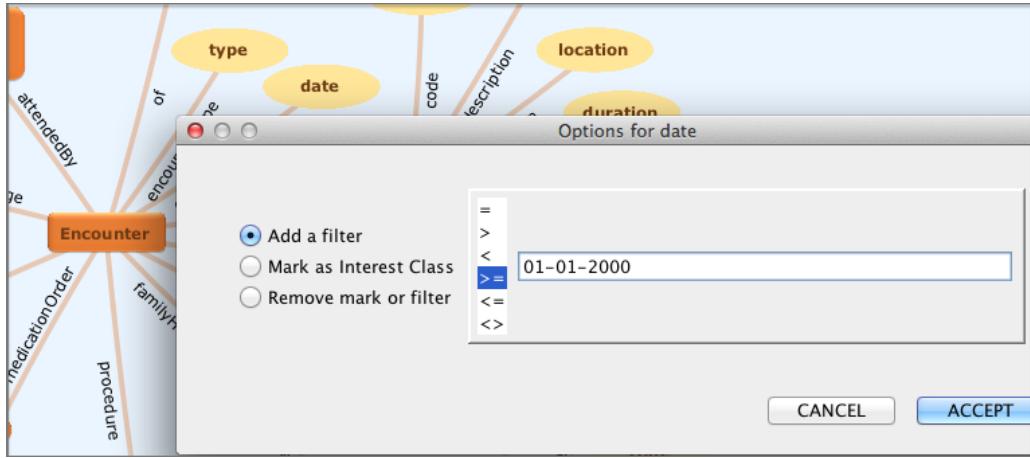


Figura 4.5: Selección de la función a cumplir en un nodo de valor básico.

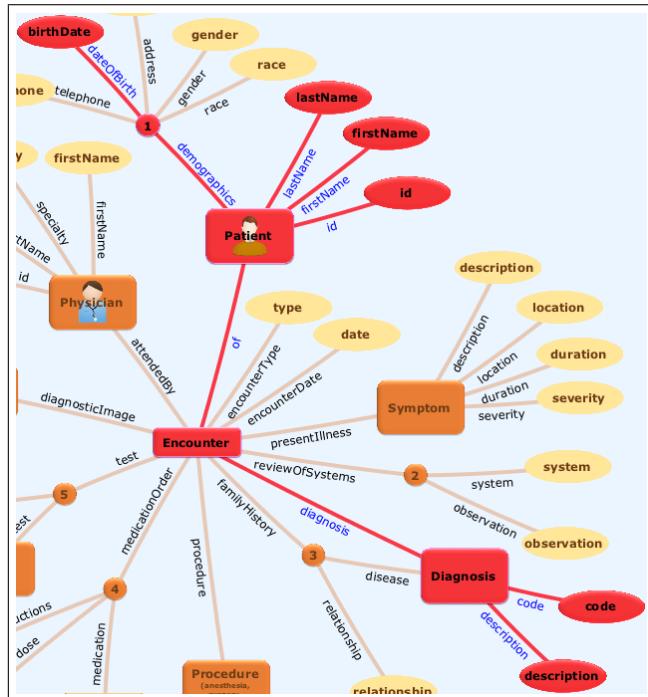
de una disyunción de las condiciones. Con el fin de proveer mayor expresividad en los filtros, sin agregar conceptos o notación gráfica que el usuario deba aprender a usar, el lenguaje realiza un diálogo de clarificación del filtro que se describe en la Sección 4.4 de este capítulo.

4.3 Descripción de los operadores de transformación de grafos

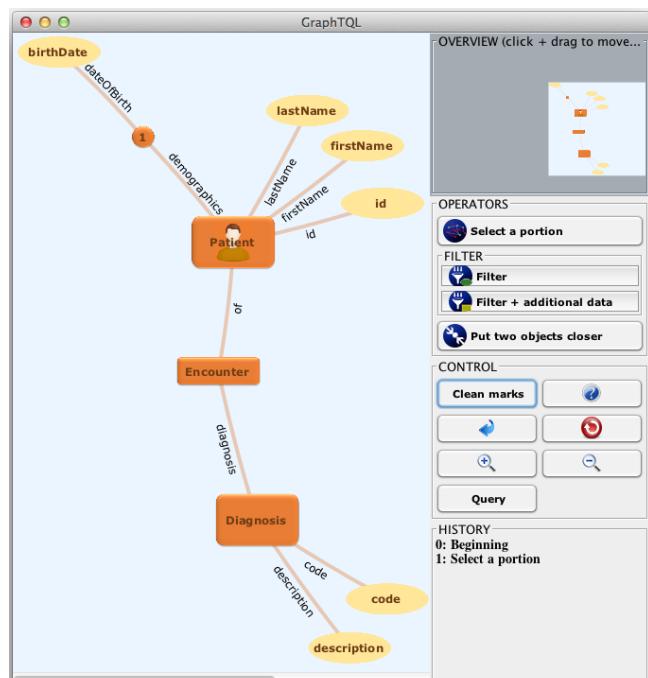
Actualmente el prototipo funcional de GraphTQL tiene implementadas cuatro operaciones: *Select a Portion*; dos tipos de filtro, *Filter* y *Filter+Additional Data*; y *Put Two Objects Closer*. La semántica de estos operadores tiene en cuenta que los datos pueden estar incompletos, por tanto se enfocan en contextos donde la topología no es la característica más importante para la selección de los datos.

Select a Portion

Mediante esta operación, se puede elegir una porción del grafo que contiene los datos de interés para el usuario. Por ejemplo, si al usuario le interesa la fecha de nacimiento, el nombre y el apellido de los pacientes, los encuentros y el código y descripción de los diagnósticos, marca los nodos y arcos existentes entre ellos (Figura 4.6a) y selecciona esa porción del grafo (Figura 4.6b). La operación no busca la coincidencia exacta de un patrón, a diferencia del *pattern matching*, sino que al seleccionar una porción se recuperan todos los nodos y arcos que corresponden con los marcados en el esquema. Por tanto, para el caso del ejemplo, se recuperan pacientes con datos incompletos.



(a)



(b)

Figura 4.6: Operador *Select a Portion*.

Filter

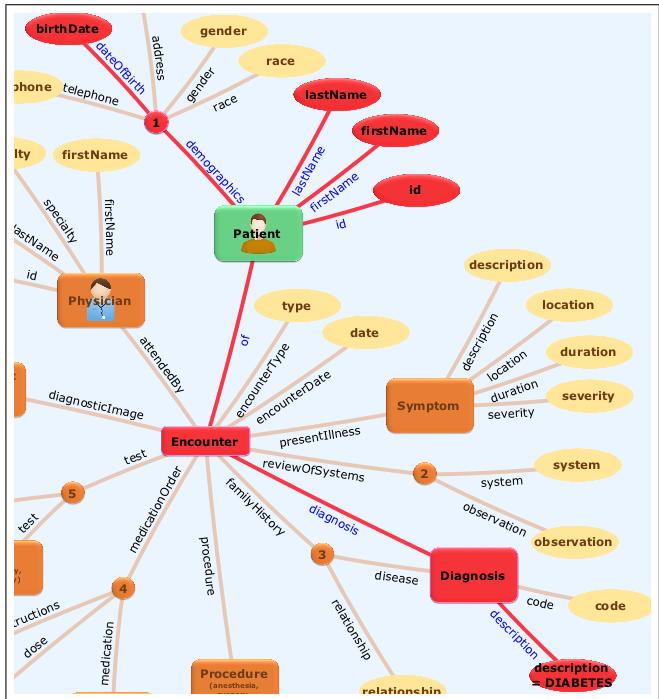
Permite seleccionar los objetos de cierto tipo, la clase de interés, que cumplen una condición específica y algunos datos conectados a ellos. Por ejemplo, si interesa la identificación, el nombre y la fecha de nacimiento de los pacientes que han recibido el diagnóstico de “*DIABETES*”, el usuario marca la clase de interés, Paciente, la condición que deben cumplir las instancias de esa clase, con diagnóstico de “*DIABETES*” y los otros datos que se requieren (nombre y fecha de nacimiento) (Figura 4.7a). Estos últimos son opcionales y aparecen en el resultado si están disponibles. El esquema del resultado contiene los nodos marcados (Figura 4.7b) y la instancia, los nodos correspondientes con el esquema y que cumplen la condición (están conectados al diagnóstico con descripción “*DIABETES*”).

Filter+Additional Data

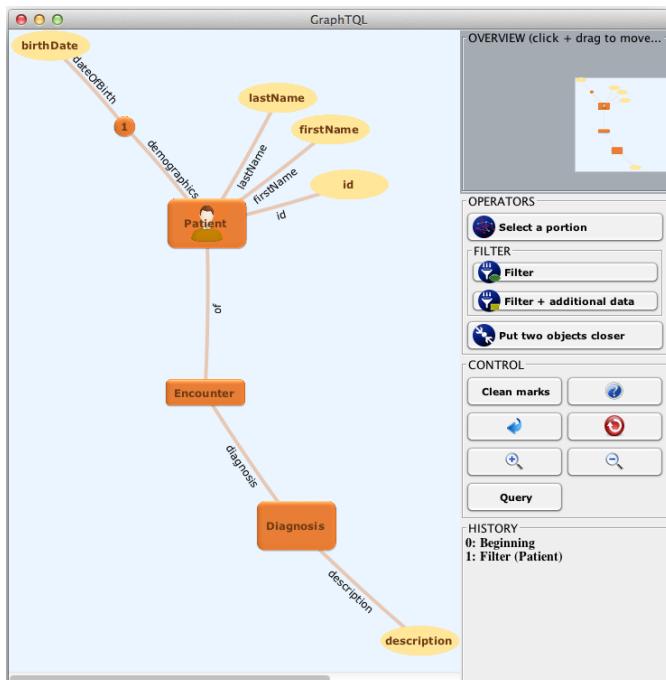
Permite seleccionar todos los datos relacionados con un objeto que cumple una condición. Por ejemplo, seleccionar todos los datos de la historia clínica de los pacientes que han tenido un diagnóstico de “*DIABETES*”. A diferencia de *Filter*, “todos los datos” incluyen otros diagnósticos recibidos, además de “*DIABETES*” y los datos de otros encuentros, adicionales a aquellos donde se diagnosticó “*DIABETES*”. Se recuperan los pacientes que cumplen la condición y los nodos que están conectados a esos pacientes a través de un camino simple marcado en el grafo esquema. Para aplicar este operador basta marcar una clase de interés (en el ejemplo, paciente), la condición que deben cumplir las instancias de esa clase y los datos que se desea recuperar.

Put Two Objects Closer

Mediante esta operación se reemplaza un camino que conecta dos nodos del esquema por una relación que se crea entre ellos, acercándolos en el diagrama del grafo. Por tanto, en el grafo instancia, los nodos objeto de las clases seleccionadas en el esquema se conectan también por la relación creada, reemplazando el camino correspondiente. En este caso no hay interés en los datos del camino y por lo tanto, desaparecen del grafo respuesta. Por ejemplo, en el resultado del ejemplo de *Filter* (Figura 4.7b), se quiere reemplazar el camino que conecta la fecha de nacimiento del paciente con la descripción de los diagnósticos que ha recibido. Para ello, se marcan los nodos y el camino a reemplazar (Figura 4.8a). El resultado en el esquema se muestra en la Figura 4.8b.

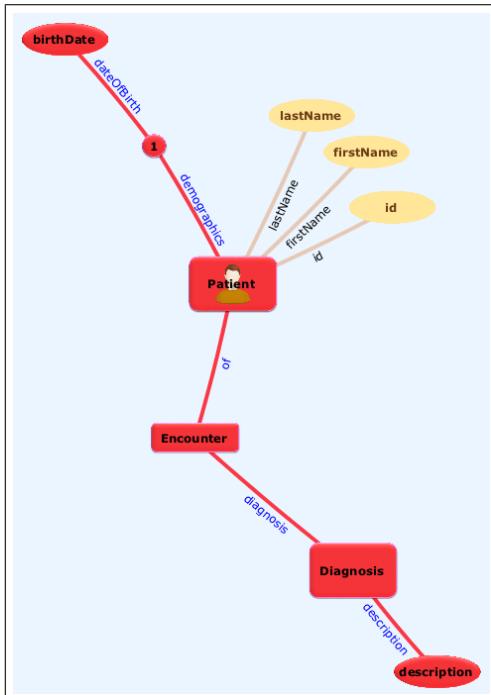


(a)

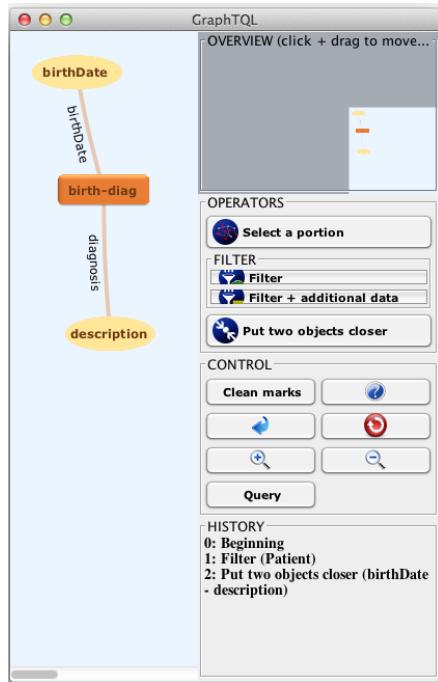


(b)

Figura 4.7: Operador *Filter*.



(a)



(b)

Figura 4.8: Operador *Put Two Objects Closer*.

4.4 Descripción del diálogo de clarificación de filtros

Con el fin de simplificar en la interfaz la representación de los operadores de filtro, se requiere que el usuario marque solamente una clase de interés, las condiciones que debe cumplir y, opcionalmente, otros nodos a seleccionar. Sin embargo, la sencillez de la representación genera dos problemas: primero, limita la expresividad de los filtros y segundo, cuando se necesita especificar más de una condición, la representación gráfica es ambigua y por tanto, se hace necesario especificar si se trata de una conjunción o de una disyunción de las condiciones.

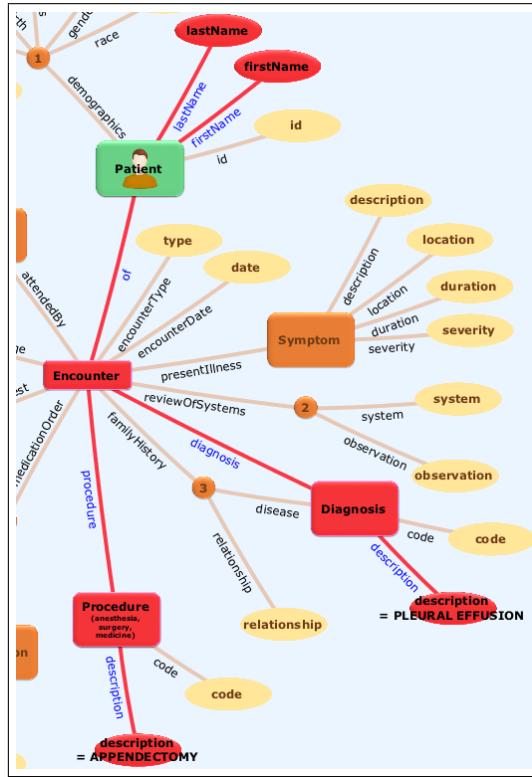
El Diálogo de Clarificación de Filtros tiene como objetivo precisar la semántica de las condiciones de filtro que el usuario marca sobre el grafo esquema. De esta manera se ofrece mayor expresividad, sin agregar conceptos o notación gráfica adicional que el usuario deba aprender a usar. Para ello se identifican los nodos de bifurcación, es decir aquellos donde se bifurcan los caminos que conectan la clase de interés con las clases de condición. El diálogo se lleva a cabo mediante una serie de preguntas que tienen en cuenta, en cada nodo de bifurcación, posibles opciones de ejecución para la condición.

Un ejemplo de este caso es el siguiente: se requieren los nombres de los pacientes a quienes se les ha realizado el procedimiento “*Appendectomy*” y que han sido diagnosticados de “*Pleural effusion*”. El patrón de filtro formado por la clase de interés (paciente) y las condiciones de filtro, que se muestran en la Figura 4.9a, puede tener varias interpretaciones: (1) el procedimiento y el diagnóstico deben ocurrir en el mismo encuentro, en cuyo caso los pacientes seleccionados cumplen con ambas condiciones (conjunción de las condiciones en el encuentro y en el paciente). (2) el procedimiento y el diagnóstico pueden ocurrir en encuentros diferentes, y el paciente debe tener ambos encuentros (disyunción de las condiciones en el encuentro y conjunción, en el paciente). (3) el procedimiento y el diagnóstico pueden ocurrir en encuentros diferentes, y el paciente puede tener alguno de ellos (disyunción de las condiciones en el encuentro y disyunción, en el paciente). El diálogo (Figura 4.9b y 4.9c) permite al usuario precisar esto en la consulta.

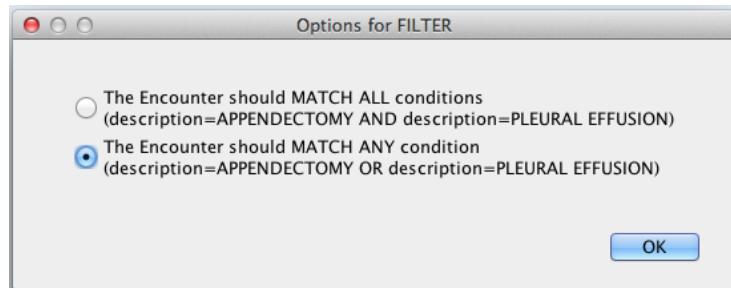
La generación de diálogos de clarificación de filtro se describen formalmente en el Capítulo 5.

4.5 Características de diseño de GraphTQL

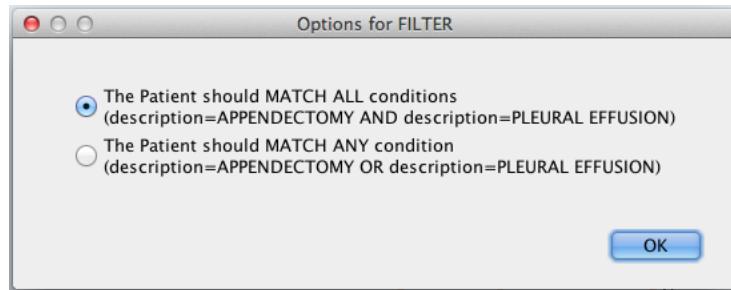
Varias características son necesarias para ofrecer a los usuarios finales un lenguaje que facilite la formulación de consultas de mediana complejidad. Primero, es necesario un modelo conceptual de datos que sirva como base para la interacción con el usuario. Segundo, se requiere guiar al usuario, por ejemplo, mediante la navegación del esquema de datos y diálogos. Tercero, es necesario que la consulta se pueda formular



(a)



(b)



(c)

Figura 4.9: Diálogo de clarificación de filtros.

incrementalmente y en cada paso ofrecer realimentación al usuario. Y cuarto, en tanto como sea posible, se requiere ocultar elementos y nociones propios de los lenguajes de consulta, tales como la definición de variables o las operaciones como *join* y *optional* (o *outer join*).

Modelo de datos basado en grafos

La selección del modelo de datos se basó en varios requerimientos: (1) Ofrecer una representación de los datos a nivel conceptual, es decir que los conceptos, el vocabulario y la organización de los datos sean familiares a los usuarios finales. (2) Facilitar el mapeo entre el modelo y otros modelos de datos, de manera que los datos puedan provenir de sistemas que usan otras representaciones. (3) Ofrecer un modelo simple y con una representación visual que facilite la comunicación con el usuario final.

Un modelo de grafos estructurado satisface estos requerimientos y, de acuerdo con Bruni y Lluch [33], ofrece las siguientes ventajas: Tiene una base matemática sólida y a la vez hace posible una notación visual accesible para personas no técnicas. Es un formalismo visual que se usa en diversas áreas como la ingeniería de software, las bases de datos, el diseño de redes y los sistemas biológicos; para representar aspectos estáticos de los datos, las topologías, las arquitecturas y la especificación de requerimientos de sistemas. Tiene diferentes formas de representación, facilitando su manipulación y almacenamiento en las aplicaciones de *software*. Además, el diagrama del grafo da la posibilidad de captar fácilmente las relaciones entre las entidades que lo conforman. El hacer evidente la relación entre los datos evita el uso explícito de conceptos como el *join*, necesario en las consultas sobre el modelo relacional.

Entre los distintos modelos de grafos, GDM [87] cumple con las características mencionadas. En particular, su representación gráfica es simple porque maneja dos grafos independientes para representar el esquema y la instancia de los datos. Entonces, los metadatos del esquema (ej. tipo de nodo) no agregan nodos o arcos en el grafo de datos, como ocurre en otros modelos (ej. RDF). Por tanto, una instancia de GDM tiene menos nodos y arcos que, por ejemplo, el modelo RDF correspondiente. En este sentido, un modelo GDM es mas fácil de navegar y entender.

La separación de los grafos esquema e instancia es particularmente útil en GraphTQL porque la interfaz usa el grafo esquema para dar al usuario una visión general de los datos disponibles y guiar la formulación de las consultas. El grafo esquema, con una representación conceptual de los datos, y el conocimiento del usuario sobre el dominio le permite a éste identificar los datos que tiene disponibles para consultar. De esta manera, GraphTQL evita que el usuario tenga la necesidad de explorar el esquema de los datos antes de formular las

consultas. Es de anotar que el grafo que se despliega es una vista simplificada de un modelo conceptual de datos, ya que no incluye elementos que usualmente se tienen en estos modelos (ej. definición de subclases, cardinalidad de las relaciones, obligatoriedad de los datos). Estos elementos, útiles para validar con el usuario los requerimientos o el diseño de un sistema, no son necesarios para la formulación de la consulta. GraphTQL tampoco visualiza la dirección de los arcos, aunque en el modelo subyacente los arcos son dirigidos. Por tanto las operaciones se basan en caminos no dirigidos.

El grafo esquema ofrece además las siguientes ventajas: (1) Usualmente es mas conciso y fácil de visualizar que el grafo instancia. (2) Dado que los nodos y arcos de un grafo GDM no tienen estructura compleja (i.e. no tienen atributos internos, no son hypernodos o hyperarcos), un grafo GDM tiene pocos constructores y los datos se representan con nodos y arcos. Esta característica permite que las operaciones del lenguaje manejen de manera uniforme los objetos y sus atributos. (3) Permite evitar errores semánticos que pueden ocurrir, por ejemplo, por la dirección equivocada en un arco, la omisión del tipo de clase cuando el mismo nombre de atributo hace parte de varias clases (ej. *name* es atributo de *Patient* y también de *Physician*) o por la especificación incorrecta de caminos (ej. olvidar que *race* se conecta a *Patient* a través de un nodo que agrupa los datos demográficos). (4) Se mapea con otros modelos de datos. Las posibilidades de hacerlo se detallan en la Sección 7.3.1.

Manipulación directa del grafo esquema

La notación visual de GraphTQL se basa en el diagrama del grafo esquema. De acuerdo con Moody [132] las notaciones visuales son importantes en la comunicación con los usuarios porque transmiten información de manera efectiva y de esta manera, los diagramas permiten presentar información de forma concisa y precisa.

GraphTQL implementa la selección de los parámetros de los operadores mediante la manipulación directa del diagrama del grafo esquema, de esta manera, el usuario no necesita construir el patrón de consulta nodo por nodo y arco por arco. Proponer los parámetros de esta forma permite ocultar nociones poco intuitivas, propias de los lenguajes de consulta, como la definición de variables, la selección de variables de salida, el manejo explícito de datos opcionales y operaciones de *join*. Estas nociones generalmente se usan en lenguajes visuales e interfaces gráficas que trasladan a una notación visual las operaciones de lenguajes de consulta con sintaxis de texto.

Operadores

Los operadores soportan el mecanismo de reducción de los grafo esquema e instancia, incluyen el manejo de datos incompletos de forma implícita, toman como parámetros los nodos marcados en el grafo esquema, generan los diálogos de clarificación de filtros para guiar al usuario en la formulación de consultas y manejan caminos no dirigidos.

La representación de la interconectividad de los datos, que se facilita en los modelos de grafos, ha sido explotada ampliamente en dominios de aplicación donde la topología es un factor esencial (ej. búsqueda de patrones en biología) o donde lo importante es encontrar cómo se conectan las entidades (i.e. búsqueda de los caminos que conectan dos nodos en redes sociales). En los dominios de aplicación hacia los que se enfoca este trabajo, como el dominio médico, las consultas de conectividad no son usuales y los patrones no bastan para encontrar los datos que se requieren porque éstos suelen ser incompletos. Una de las ventajas de los modelos de grafos es que los datos incompletos no se representan en el modelo (i.e. no es necesario agregar una representación de estos como el valor nulo), cuando un objeto no tiene toda las propiedades definidas para su clase, simplemente el nodo (o arco) no está en el grafo. Esta manera natural de manejar información incompleta en el modelo no se traslada a los lenguajes de consulta que se basan en búsqueda de patrones. Los operadores de GraphTQL proveen una abstracción que libera al usuario de la necesidad de razonar sobre cómo los datos incompletos afectan los resultados de sus consultas.

Los operadores tienen alto nivel de abstracción, en el sentido de que encapsulan muchas operaciones básicas de transformación (i.e. agregar y borrar nodos y arcos) y le permiten al usuario manipular los objetos manteniendo sus relaciones y atributos. Además, son compositacionales, dado que tanto las entradas como las salidas son grafos. Esta característica también permite que la formulación de la consulta se realice sobre una sola representación de los datos, a diferencia de los lenguajes basados en búsqueda de patrones (*pattern matching*). En estos últimos, generalmente la búsqueda del patrón entrega como resultado conjuntos de mapeos (*set of mappings* [152]). Un mapeo asocia valores a las variables del patrón de búsqueda. Luego, la semántica de los otros operadores del lenguaje (i.e. *union*, *optional*, *filter*) se define sobre los conjuntos de mapeos. Esta semántica puede resultar confusa para un usuario no experto, por ejemplo en los siguientes casos:

- Cuando el patrón incluye dos subgrafos desconectados la respuesta es el producto cartesiano de los conjuntos de mapeos de ambos subgrafos. Esto puede generar la falsa idea de que hay una relación entre los datos, cuando en realidad no existe.

-
- Cuando se hace unión de conjuntos mapeos, los datos relacionados con un mismo nodo pueden estar en mapeos diferentes. Por tanto, un usuario puede perder de vista una relación que existe entre algunos datos.
 - El uso de operadores como *Optional* de SPARQL debe cumplir ciertas restricciones sintácticas [152] para evitar ambigüedades.
 - El *join* entre conjuntos de mapeos se basa en la noción de compatibilidad. Dos mapeos son compatibles cuando no tienen variables en común, o cuando las tienen y los valores asociados a las variables comunes son iguales en los dos mapeos. Cuando se tienen datos incompletos (i.e. por resultado de operadores como *optional*) algunas variables no asocian valor, por lo tanto no hacen parte del mapeo. Como resultado, al hacer *join* de dos conjuntos de mapeos, el *join* de dos mapeos (uno de cada conjunto) puede tener en cuenta un conjunto de variables diferente a las del *join* de otros dos mapeos.

Formulación incremental de la consulta y realimentación

La consulta se formula con una secuencia de operaciones y cada vez que el usuario elige una operación el grafo esquema resultado se despliega, de manera que el usuario puede ver el conjunto de datos que ha seleccionado.

Enfoque de Diseño centrado en el usuario

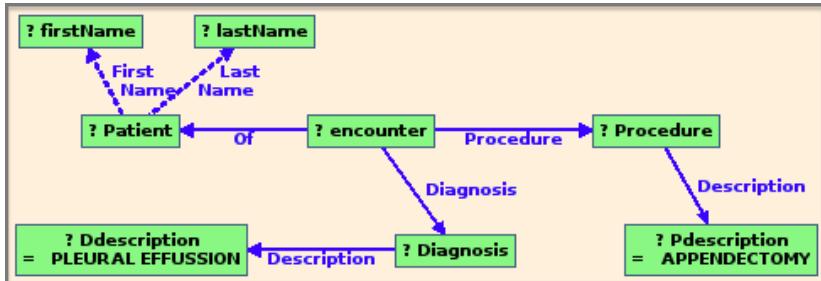
Varias decisiones de diseño de GraphTQL fueron influenciadas por los resultados de las pruebas realizadas como parte del proceso de diseño centrado en el usuario. El detalle de las pruebas realizadas y sus resultados se presentan en el Capítulo 6.

Consultas de mediana complejidad y diálogos de clarificación

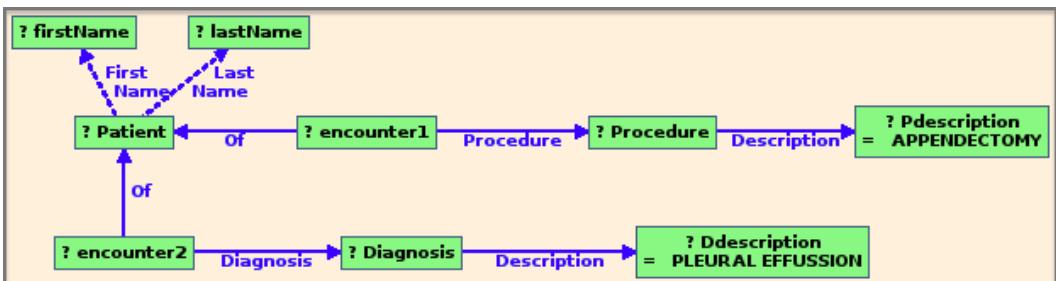
En el contexto de este trabajo las consultas de mediana complejidad se caracterizan porque: (1) Los objetos se filtran no solo por los valores de sus atributos, sino también por estar conectados con otros objetos que cumplen cierta condición. (2) Los filtros incluyen varias condiciones unidas por conectivos lógicos. (3) Requieren manejar datos incompletos. La representación gráfica de estas consultas puede generar diversas interpretaciones. Los diálogos de Clarificación de GraphTQL guían al usuario para que especifique de manera precisa las condiciones de la consulta.

La Figura 4.9, por ejemplo, muestra la formulación en GraphTQL de una consulta que recupera los nombres de los pacientes a quienes se les ha realizado el procedimiento “*Appendectomy*” y que han sido diagnosticados de “*Pleural effusion*”. Mediante el diálogo de clarificación de filtros el usuario puede elegir entre las siguientes variantes de la consulta:

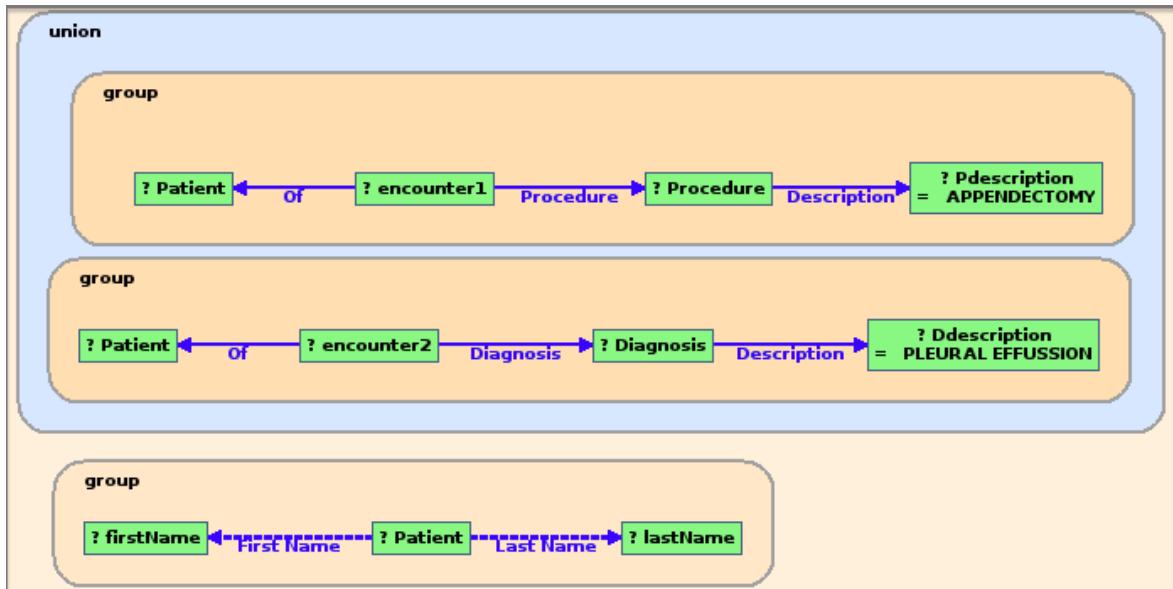
- El procedimiento y el diagnóstico deben ocurrir en el mismo encuentro, en cuyo caso los pacientes seleccionados cumplen con ambas condiciones



(a)

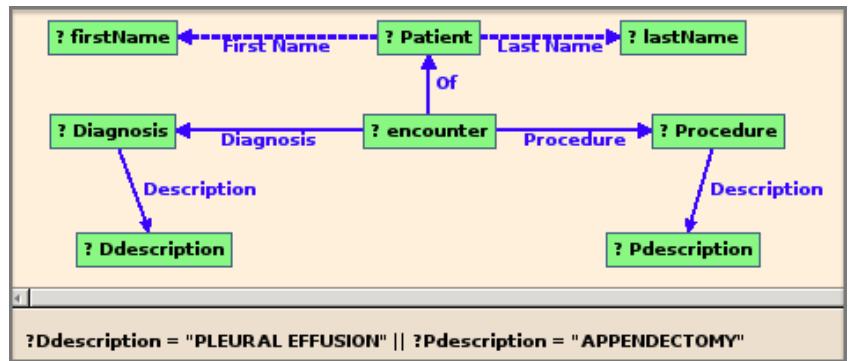


(b)

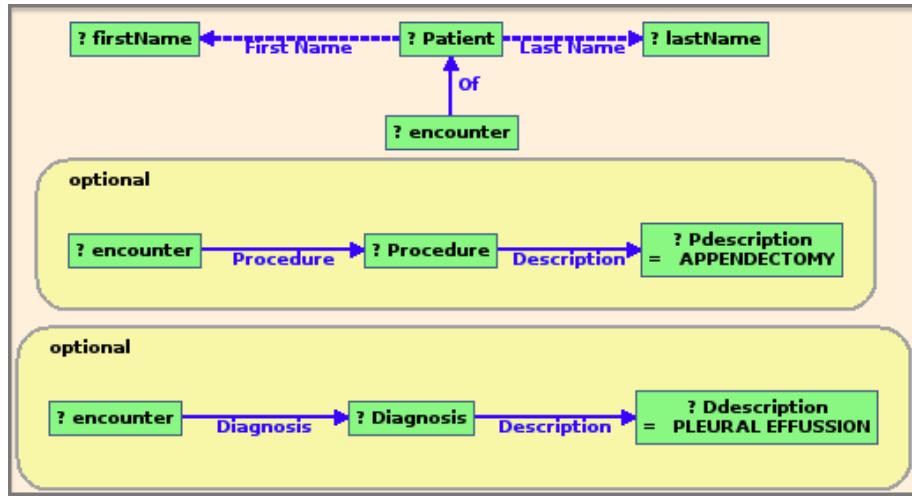


(c)

Figura 4.10: Consultas formuladas en Gruff



(a)



(b)

Figura 4.11: Consultas formuladas en Gruff

(conjunción de las condiciones en el encuentro y en el paciente). La formulación de esta consulta en Gruff (<http://franz.com/agraph/gruff/>), una interfaz gráfica para SPARQL, se muestra en la Figura 4.10a.

- El procedimiento y el diagnóstico pueden ocurrir en encuentros diferentes, y el paciente debe tener en su historia ambos encuentros (disyunción de las condiciones en el encuentro y conjunción, en el paciente). La Figura 4.10b muestra la formulación de esta consulta en Gruff, se usan dos variables para representar los encuentros, cada uno asociado a una condición diferente. Sin embargo, cuando las dos condiciones ocurren en el mismo encuentro, en la respuesta de la consulta no es evidente que las variables *encounter 1* y *encounter 2* tienen el mismo valor, es decir, se refieren al mismo nodo. Una variante que usa una disyunción en un filtro global se muestra en la Figura 4.11a sin embargo, la semántica es diferente, ya que en este caso el patrón se debe cumplir, es decir los encuentros seleccionados tienen al menos un procedimiento y un diagnóstico, aún cuando solamente uno de ellos cumpla con la condición de filtro.
- El procedimiento y el diagnóstico pueden ocurrir en encuentros diferentes, y el paciente puede tener alguno de ellos (disyunción de las condiciones en el encuentro y disyunción en el paciente). La Figura 4.10c muestra la formulación de esta consulta en Gruff. Una variante similar a esta se muestra en la Figura 4.11b, sin embargo la semántica es diferente, ya que en este caso se recuperan todos los pacientes y sus encuentros, aun cuando no cumplan con las condiciones.

Estos ejemplos muestran que los diálogos de clarificación permiten reducir el número de conceptos y elementos que se incluyen en la notación visual de GraphTQL. En el ejemplo dado, para formular las consultas en una interfaz gráfica para SPARQL el usuario necesita conocer las cláusulas *optional* y *union*, los agrupadores y la especificación de filtros globales.

Características generales de usabilidad

Durante el diseño y desarrollo de la interfaz se incluyeron algunas características que buscan facilitar su uso y la formulación de las consultas, entre ellas: Un diseño simple, que contiene solamente la información que el usuario necesita. Minimizar la carga de memoria del usuario desplegando elementos de diálogo que lo guían y que reducen la cantidad de elementos que el usuario debe aprender. Un diseño estético, tanto en la disposición (*layout*) del grafo esquema, como en los iconos de los botones, la disposición de los marcos, los colores de la aplicación y el agrupamiento de los comandos (la disposición de los marcos fue resultado de la opinión de los usuarios en una de las pruebas realizadas, Capítulo 6). Despliegue del grafo de forma estética, atendiendo a parámetros de simetría y evitando el cruce de arcos y la superposición de nodos. Uso de un lenguaje natural para

el usuario tanto en el modelo conceptual de datos como en los nombres de los operadores, que se refinaron a partir de la realimentación obtenida en las pruebas de usabilidad. Uso de iconos similares a los usualmente utilizados en aplicaciones de *software*, de manera que sean representativos para el usuario. Ofrecer realimentación al usuario durante la formulación de la consulta. Ofrecer un historial de la interacción, y permitir deshacer una o varias operaciones o volver al estado inicial. Consistencia, por ejemplo, en los efectos de los comandos y en la ubicación de la información. Incluir validaciones para detectar preventivamente posibles errores en la selección de los parámetros de los operadores.

Algunos de los principios propuestos por Moody [132] para el diseño de notaciones visuales efectivas que se tuvieron en cuenta fueron: claridad semiótica, para evitar redundancia, sobrecarga, exceso y déficit de símbolos; uso de formas, colores y texto para discriminar los símbolos y generar expresividad visual; uso de representaciones visuales que sugieran un significado, por ejemplo en los iconos de las acciones o los asociados a clases de objeto del modelo; y número reducido de símbolos visuales, para hacerlo cognitivamente manejable.

4.6 Discusión

Algunas opciones de diseño de las características de GraphQL presentadas en este capítulo se analizaron y, en otros casos, se evaluaron mediante las pruebas de usuario. El detalle de estas pruebas y sus resultados se presentan en el Capítulo 6, sin embargo, los casos más representativos se relacionan a continuación.

El primer prototipo del lenguaje fue desarrollado en el marco de un trabajo de grado de pregrado. Este prototipo no incluyó todas las características del lenguaje, como el diálogo de clarificación de caminos. En las primeras versiones del lenguaje, el usuario marcaba los caminos entre los nodos haciendo clic sobre cada arco y nodo. Las pruebas internas de desarrollo mostraron lo engorroso que esto podía ser. La primera solución que se ofreció fue incluir un botón de control con la función de buscar y marcar los caminos entre los nodos marcados. Cuando los nodos formaban ciclos todos los caminos se marcaban y el usuario debía quitar aquellos que no requerían para su consulta. Los resultados de la primera prueba de evaluación con usuarios, realizada en el marco del trabajo de pregrado, mostraron que, generalmente, los usuarios no usaban el botón. La siguiente versión del prototipo eliminó el botón, se optó por marcar los caminos cada vez que el usuario marcara otro nodo, aun en los casos en que el último nodo marcado se conectaba por varios caminos a los otros nodos. Así el usuario debía entonces quitar las marcas de los arcos que no se requerían en su consulta.

Una segunda prueba de evaluación mostró que los usuarios no revisaban las marcas que la interfaz añadía, por tanto aceptaban todos los caminos marcados, lo cual generaba errores en la formulación de las consultas. Entonces se optó por incluir el

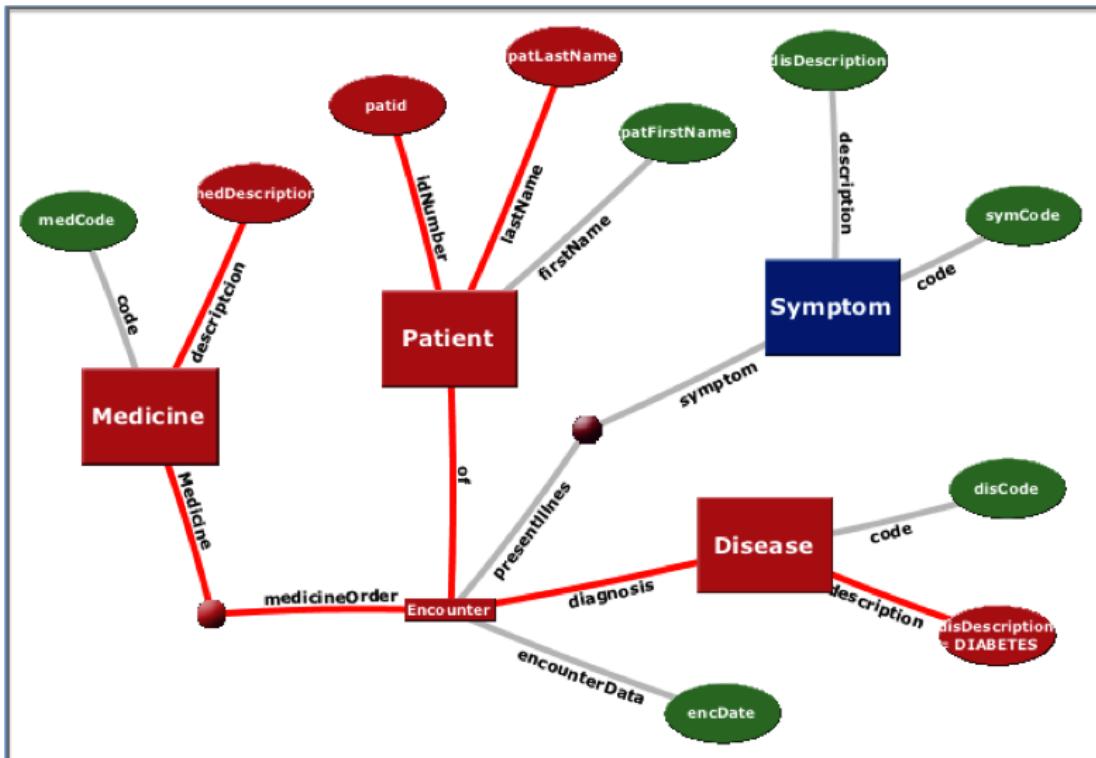
diálogo de clarificación de filtros para manejar los casos en que un nodo se conecta por varios caminos a los nodos previamente marcados. Las pruebas comparativas mostraron que esta opción no supone una sobrecarga para los usuarios, que ellos identifican rápidamente y correctamente los caminos que se muestran en el diálogo y que en la mayoría de los casos eligen los caminos adecuados para la consulta.

Los filtros y los diálogos de clarificación de filtros también tuvieron varias etapas de refinamiento. Inicialmente, los operadores *Filter* y *Filter+Additional Data* tenían grandes diferencias.

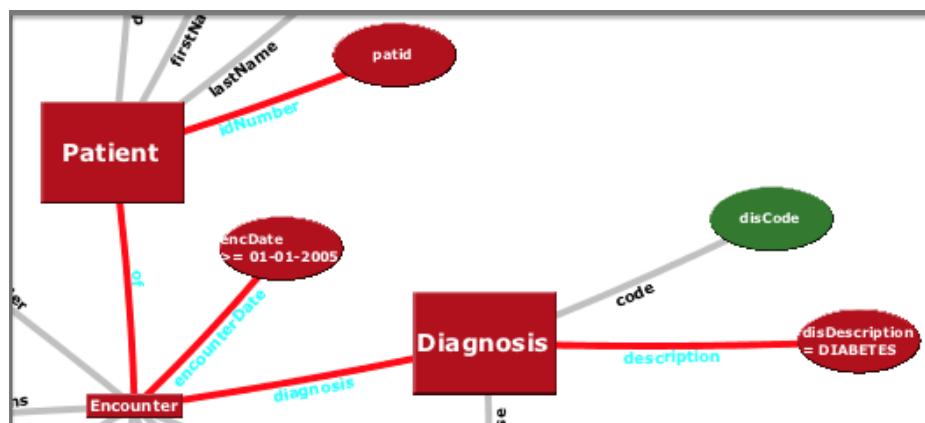
La primera versión de *Filter* se denominó *Value Filter* y su entrada constaba de un grupo de nodos de valor básico con condiciones de filtro, un grupo de nodos marcados (objetos o de valor básico) y un conjunto de caminos entre ellos. *Value Filter* recuperaba todos los caminos en la instancia que correspondían con algún camino seleccionado que conectara un nodo con condición con un nodo marcado. La unión de estos caminos formaba la respuesta de la consulta. Por ejemplo, en la Figura 4.12a se muestran las entradas marcadas para un operador *Value Filter* cuyo resultado estaría formado por la unión de: (1) los nombres de los pacientes diagnosticados con diabetes y los caminos que unen estos nombres con el nodo “*DIABETES*”; (2) la identificación de los pacientes con diagnóstico de diabetes y los caminos que unen estas identificaciones con el nodo “*DIABETES*”; (3) las descripciones de las medicinas ordenadas cuando se dio un diagnóstico de diabetes y los caminos que unen estas medicinas con el nodo “*DIABETES*”. Esta interpretación parece natural para este ejemplo. Sin embargo, no parece serlo tanto en otros casos, como el que se muestra en la Figura 4.12b donde probablemente se requiere la conjunción de las condiciones, es decir se requiere la identificación de los pacientes que fueron diagnosticados con “*DIABETES*” en una fecha igual o posterior a “01-01-2005”.

Dos posibles soluciones se plantearon: (1) Dejar que el usuario resolviera el caso, aplicando dos *Value Filter* consecutivos para lograr la conjunción (Figura 4.13). En la Figura 4.13a se selecciona la identificación de los pacientes que fueron diagnosticados con diabetes. Sobre el resultado de ese primer filtro, en la Figura 4.13b, se seleccionan aquellos diagnósticos ocurridos en una fecha igual o posterior a “01-01-2005”. (2) Agregar diálogos de clarificación que permitieran definir cuándo aplicar una conjunción y cuándo una disyunción, de manera que el lenguaje generara automáticamente la secuencia de *Value Filters* necesaria, ocultando la semántica del *Value Filter* y ofreciendo un operador de filtro con una semántica diferente.

La primera opción se descartó porque supone mayor carga cognitiva para el usuario. Para aplicar la segunda opción, se generaron diálogos de manera similar a la descrita en la Sección 4.4, pero tomando aleatoriamente un nodo marcado sin condición como raíz del árbol de caminos. Sin embargo, tomar cualquier nodo como raíz podía ocasionar que se presentaran opciones de diálogo con poco sentido para el usuario. Se necesitaba entonces una clase principal sobre la cual operara el filtro: **la clase de interés**. Ésta es la clase sobre la cual se aplican las condiciones de filtro, y representa el conjunto

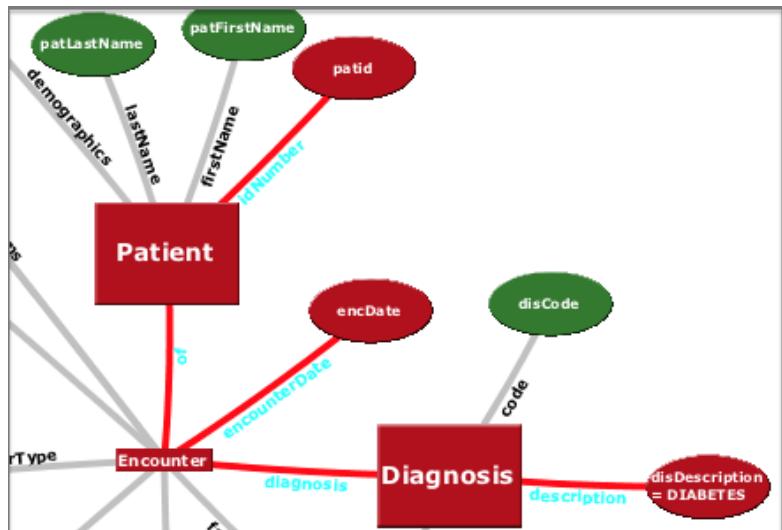


(a)

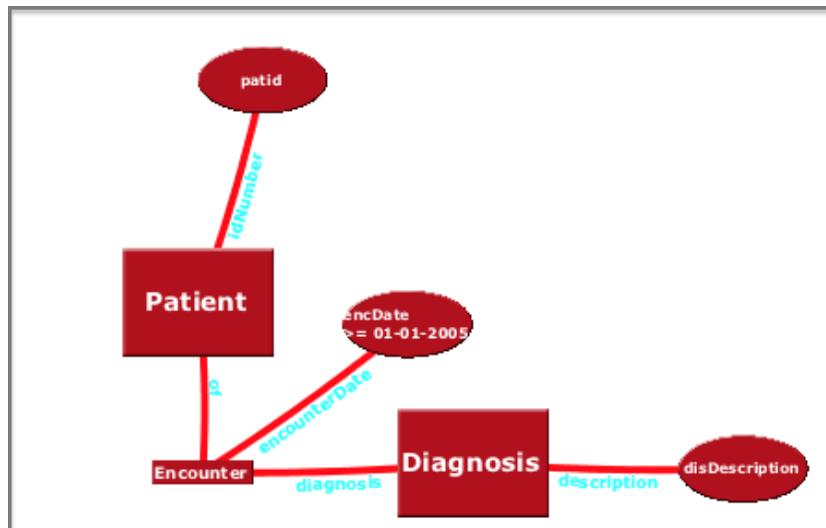


(b)

Figura 4.12: Operador *Value filter*.



(a)



(b)

Figura 4.13: Aplicación de *Value filters* consecutivos.

de objetos principal que se requiere recuperar para, a partir de ellos, recuperar otros objetos relacionados, seleccionados en la consulta. Esta clase incrementa la posibilidad de que el diálogo tenga sentido para el usuario

Estas decisiones derivaron en la definición del *Filter* de GraphTQL, y el *Value Filter* evolucionó hacia la definición del operador lógico de filtro (Ver Capítulo 5) que soporta la implementación del *Filter*.

Por su parte, la semántica del operador *Filter+Additional Data* se conserva desde las primera versión, con la única diferencia de que su entrada constaba de la clase de interés, las clases con condición y los caminos entre ellas. Todos los datos del grafo se consideraban datos adicionales (i.e. se incluían en la respuesta como datos opcionales). Por tanto el grafo esquema no cambiaba. En la segunda prueba de usabilidad se observaron dos situaciones relacionadas con este operador: (1) Bajo esta definición, generalmente se requiere aplicar un subgrafo antes de aplicar *Filter+Additional Data*, porque normalmente se necesita un subconjunto de las clases y atributos del esquema. (2) El hecho de que el grafo esquema no cambiara desconcertaba a los usuarios y los llevaba a pensar que no se había aplicado la operación. Como resultado, se cambió la definición del *Filter+Additional Data* para incluir nodos marcados que no hacen parte del patrón de filtro.

Finalmente, es de anotar que el modelo conceptual de GraphTQL en un ambiente real se define con base en el modelo de datos del sistema de información sobre el cual se van a realizar las consultas, teniendo en cuenta los principios de modelado conceptual que se describen en la Sección 2.1.1.

Capítulo 5

Operadores de Transformación de Grafos de Datos

En este capítulo se definen los operadores de GraphTQL. Inicialmente los operadores del lenguaje se definen en términos de los operadores de nivel lógico (Sección 5.2). Luego, en la Sección 5.3, los operadores de nivel lógico se definen en términos de las transformaciones del grafo esquema e instancia. Estas transformaciones se expresan sobre el modelo de datos GDM, que se describe en la Sección 5.1. Los ejemplos hacen alusión al ejemplo de referencia de este documento (Sección 4.1).

5.1 Modelo de datos

GDM [87], el modelo de datos que subyace a GraphTQL, es un grafo de datos con esquema. Los nodos y arcos de GDM son simples (i.e. sin atributos, hipernodos, o hiperarcos). GDM permite representar valores compuestos y relaciones n -arias. El modelo propuesto por Hidders [86, 87] define el grafo esquema, el grafo instancia y relaciones de extensión entre elementos de ambos grafos.

Preliminares

La siguiente notación se utiliza en la definición de los grafos esquema e instancia: \mathcal{C} es un conjunto de nombres de clase, \mathcal{A} un conjunto de nombres de atributo que no incluye *isa* o *is*, \mathcal{B} un conjunto de nombres de tipos de datos básicos que no incluye *com* o *obj*, \mathcal{D} un conjunto de valores de tipos de datos básicos, $\delta : \mathcal{B} \rightarrow \mathcal{P}(\mathcal{D})$ una función que asigna un dominio a cada tipo de datos básico, donde $\mathcal{P}(\mathcal{X})$ denota el conjunto potencia del conjunto \mathcal{X} y $\mathcal{P}_{fin}(\mathcal{X})$ denota el conjunto de subconjuntos finitos de \mathcal{X} .

En la Figura 5.1,

$$\mathcal{C} = \{Patient, Encounter, Disease, \dots\},$$

$$\begin{aligned}\mathcal{A} &= \{attendedBy, familyHistory, id, firstName, \dots\}, \\ \mathcal{B} &= \{firstName, id, code, description, \dots\}\end{aligned}$$

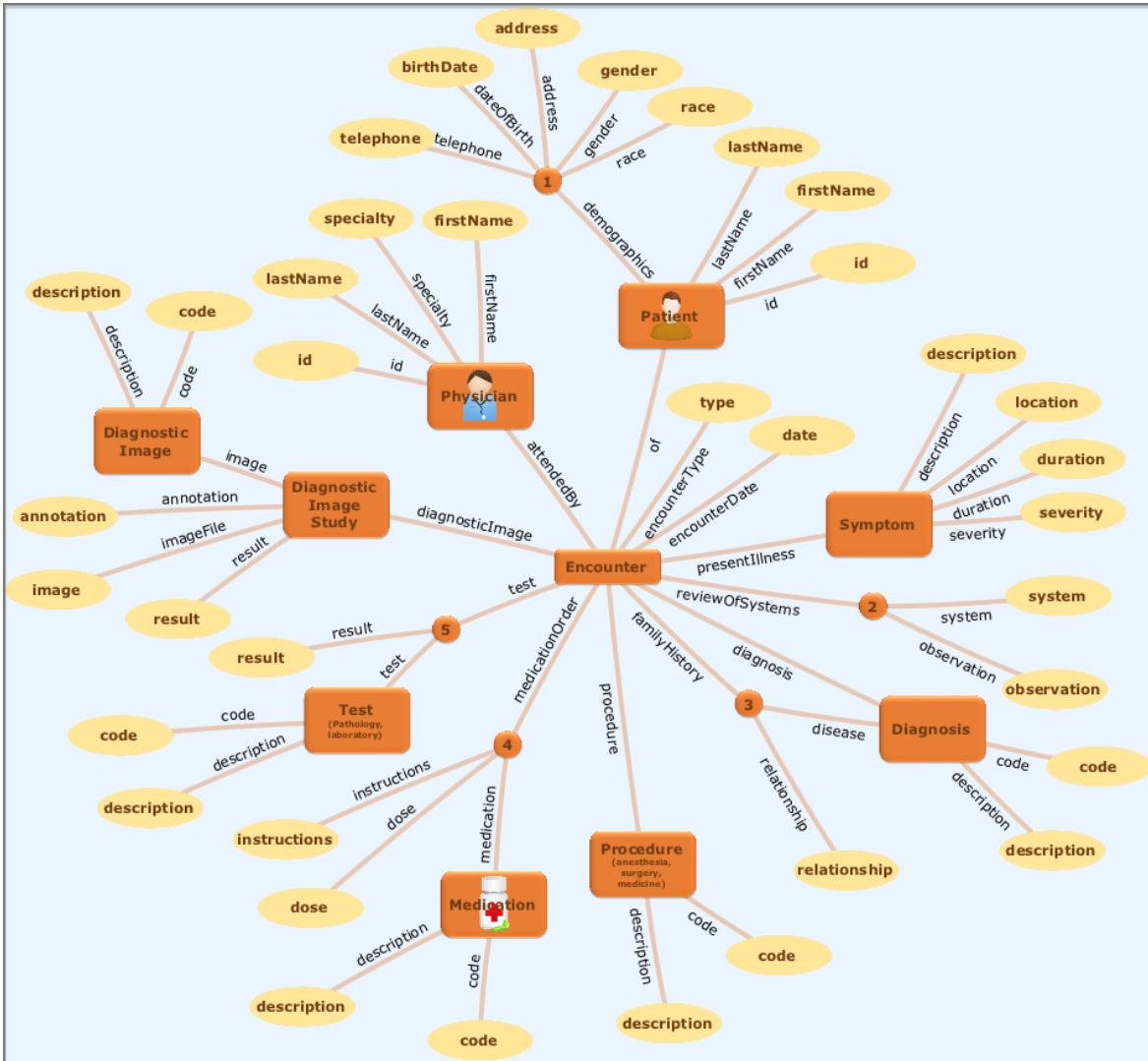


Figura 5.1: Ejemplo de referencia - Esquema de Datos (los rectángulos representan nodos objeto; los rectángulos pequeños, nodos de valor compuesto etiquetados; los círculos pequeños, nodos de valor compuesto sin etiqueta; y los óvalos, nodos de valor básico).

Grafo Esquema

Un grafo esquema está formado por un conjunto de nodos y un conjunto de arcos que representan, respectivamente, clases y atributos. Un grafo esquema S se define como $S = \langle N, E, \lambda, \sigma \rangle$, donde $\langle N, E, \lambda \rangle$ es un grafo finito parcialmente etiquetado, N es un conjunto de nodos, E es un conjunto de arcos, $E \in N \times \mathcal{A} \times N$, y λ es una función de etiquetado, $\lambda(m) : N \rightarrow \mathcal{C}$ para $m \in N$ y $\lambda(e) : E \rightarrow \mathcal{A}$ para $e \in E$. Un nodo sin etiqueta se denomina “nodo de clase implícita” (*implied class node*). Las clases en GDM pueden ser de tres tipos (*sort*): objeto (*obj*), de valor compuesto (*com*) y de valor básico (corresponde con un nombre en \mathcal{B}). La función $\sigma : N \rightarrow \{obj, com\} \cup \mathcal{B}$ asigna a cada nodo su tipo. Por lo tanto, un nodo puede ser objeto, de valor compuesto o de valor-básico. Las relaciones entre los objetos se representan mediante nodos de valor compuesto etiquetados cuyos atributos son las clases que participan en la relación. De esta manera GDM soporta relaciones n -arias.

En este documento, N_S, E_S, λ_S y σ_S se refieren a los componentes de un grafo esquema S ; m , a un nodo de N_S ; e_S , a un arco de E_S ; y α , a la etiqueta de un arco (i.e. el nombre de un atributo). Por simplicidad, en los ejemplos los nodos y arcos se referencian por su etiqueta. Los nodos de valor compuesto sin etiqueta se numeran en el grafo esquema y se referencian como β_i , donde i es el número con el que se representa en el grafo.

Los siguientes conjuntos corresponden al ejemplo de la Figura 5.1,

$$N_S = \{Patient, Physician, Disease, Encounter, id, firstName, \dots\}$$

$$E_S = \{(Patient, id, id), (Patient, firstName, firstName), \dots\}$$

$$\lambda_S(Patient) = Patient, \quad \lambda_S(firstName) = firstName, \dots$$

$$\lambda_S((Patient, id, id)) = id,$$

$$\lambda_S((Patient, firstName, firstName)) = firstName, \dots$$

$$\sigma_S(Patient) = obj, \quad \sigma_S(Physician) = obj, \quad \sigma_S(Encounter) = com, \quad \dots$$

$$\sigma_S(id), \sigma_S(firstName), \sigma_S(lastName), \dots \in \mathcal{B}$$

Observe que en el esquema de referencia, los nodos de valor básico tienen generalmente el mismo nombre que el arco correspondiente; razón por la cual aparece repetido ese nombre en el conjunto E_S .

Grafo Instancia

En el grafo instancia los nodos y los arcos representan, respectivamente, entidades y atributos. Un grafo instancia se define como $I = \langle N, E, \lambda, \sigma, \rho \rangle$, donde $\langle N, E, \lambda \rangle$ es un grafo finito etiquetado, N es un conjunto de nodos, E es un conjunto de arcos, $E \in N \times \mathcal{A} \times N$, las etiquetas de los nodos pertenecen a $\mathcal{P}_{fin}(\mathcal{C})$ y las etiquetas de los arcos a \mathcal{A} (i.e. un nodo se etiqueta con cero o mas nombres de clase, mientras que un arco se etiqueta con un nombre de atributo). La función $\sigma : N \rightarrow \{com, obj\} \cup \mathcal{B}$ asigna una categoría (*sort*) a cada nodo. Las entidades se clasifican de acuerdo con

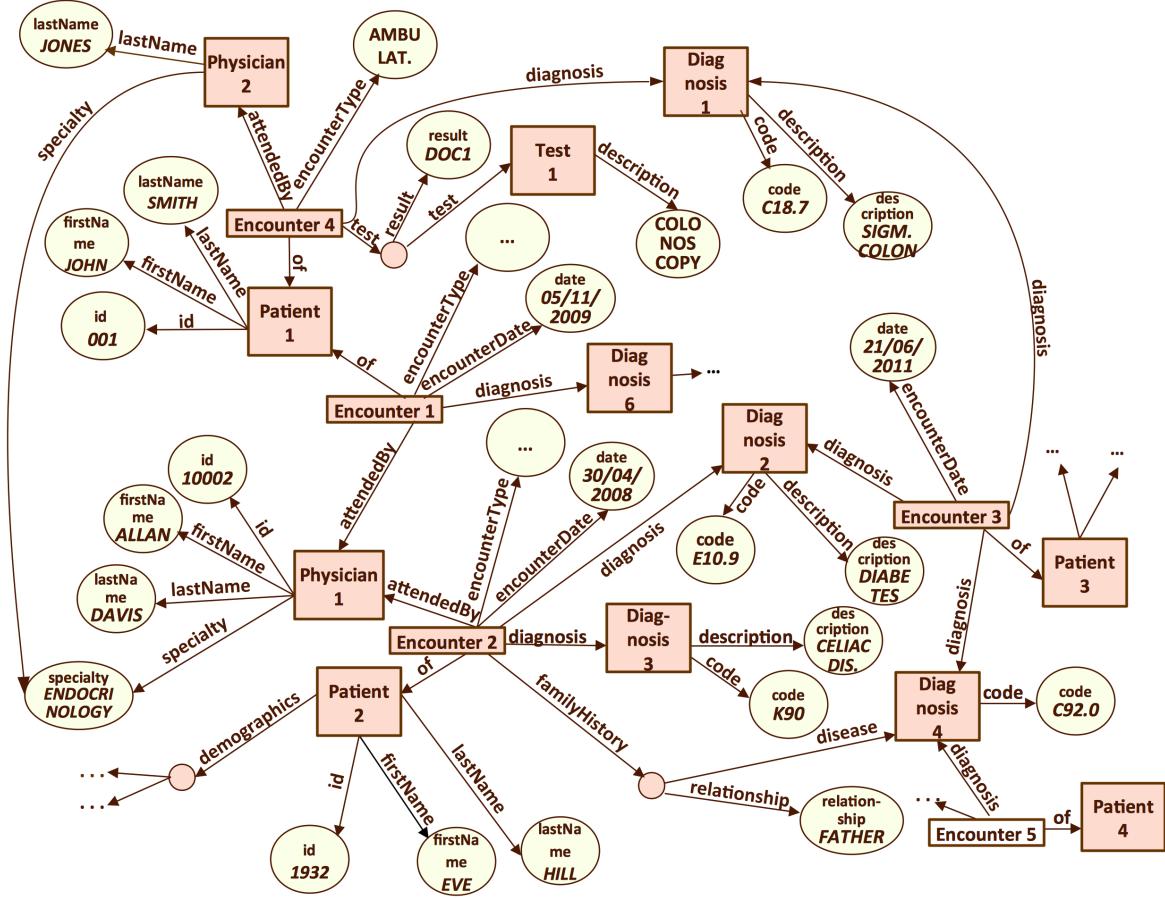


Figura 5.2: Ejemplo de una instancia de datos clínicos (por claridad, se pintan los valores de los nodos de valor básico).

su categoría (obj , com o un tipo de \mathcal{B}) en objetos, entidades de valor compuesto o entidades de valor básico. Por tanto, un nodo puede ser objeto, de valor compuesto o de valor básico. La función $\rho : \{n \in N | \sigma(n) \in \mathcal{B}\} \rightarrow \mathcal{D}$ asocia un valor básico con cada nodo de valor básico. La definición de una instancia bien formada se puede encontrar en [87].

De manera similar al grafo esquema, en adelante, $N_I, E_I, \lambda_I, \sigma_I$ y ρ_I se refieren a los componentes de un grafo instancia I ; n , a un nodo de N_I ; e_I , a un arco de E_I ; y α , a la etiqueta de un arco.

La Figura 5.2 muestra una instancia con algunos datos clínicos de pacientes. Para facilitar la lectura del grafo se han agregado números que diferencian las entidades de los nodos objeto y, el valor del atributo en los nodos de valor básico. En este ejemplo,

$$N = \{Patient1, Diagnosis2, Encounter4, id001, firstNameJOHN, \dots\}$$

$$\begin{aligned}
E &= \{(Patient1, id, id001), (Patient2, firstName, firstNameJOHN), \dots\} \\
\lambda(Patient1) &= Patient, \quad \lambda(Encounter2) = Encounter, \\
\sigma(Patient1) &= obj, \quad \sigma(Physician2) = obj, \quad \sigma(Encounter2) = com, \dots \\
\sigma(id001), \sigma(id10002), \sigma(codeE10.9), \dots &\in \mathcal{B} \\
\rho(id001) &= 001, \quad \rho(firstNameJOHN) = JOHN, \\
\rho(lastNameSMITH) &= SMITH, \dots
\end{aligned}$$

Relación de Extensión

Una relación de extensión $\xi = \{(m, n) | m \in N_S \text{ y } n \in N_I\}$ entre las entidades de un grafo instancia I y las clases de un grafo esquema S , representa las clases a las cuales pertenece cada entidad. $\xi(m, n)$ denota $\langle m, n \rangle \in \xi$ y satisface las siguientes condiciones:

- a) si $\lambda_S(m)$ está definido y $\lambda_S(m) \in \lambda_I(n)$, entonces $\xi(m, n)$;
- b) si $\xi(m_1, n_1), \langle n_1, \alpha, n_2 \rangle \in E_I$ y $\langle m_1, \alpha, m_2 \rangle \in E_S$, entonces $\xi(m_2, n_2)$; y
- c) si $\langle m_1, isa, m_2 \rangle \in E_S$ y $\xi(m_1, n)$, entonces $\xi(m_2, n)$.

Por ejemplo, entre los grafos esquema e instancia de las Figuras 5.1 y 5.2 hay una relación de extensión $\xi(firstName, firstNameJOHN)$.

Un grafo I **es una instancia de** un grafo esquema S (*belongs to*) cuando la mínima relación de extensión ξ de S a I cumple con las siguientes condiciones:

- a) Para cada $\xi(m, n)$ si $c = \lambda_S(m)$ entonces $c \in \lambda_I(n)$,
- b) ξ asocia solamente nodos de la misma categoría (i.e. obj, com, \mathcal{B}) y
- c) ξ cubre I . Es decir, para cada $n \in N_I$ existe algún $m \in N_S$ tal que $\xi(m, n)$; para cada nombre de clase $c \in \lambda_I(n)$ hay un nodo $m \in N_S$ tal que $\xi(n, m)$ y $c = \lambda_S(m)$; y para cada arco $(n_1, \alpha, n_2) \in E_I$ existe algún $(m_1, \alpha, m_2) \in E_S$ tal que $\xi(m_1, n_1)$ y $\xi(m_2, n_2)$.

Caminos no dirigidos

En las siguientes secciones se asume que los caminos de un grafo son no dirigidos. Dado un grafo dirigido G , un camino no dirigido $p = \{m_0, e_1, m_1, \dots, e_k, m_k\}$ es un camino del grafo no dirigido subyacente a G .

Para facilitar su lectura, en este documento los caminos en los ejemplos se representan por medio de las etiquetas de los nodos y de los arcos que lo forman, separadas por coma y limitadas por llaves. Por ejemplo, el conjunto de arcos $\{(Encounter, of, Patient), (Encounter, procedure, Procedure), (Procedure, description, description)\}$ de la Figura 5.1 forma el camino $\{Patient, of, Encounter, procedure, Procedure, description, description\}$ entre los nodos *Patient* y *description*.

Nodes() y *Edges()*

Dado un arco $e = (n, \alpha, m)$, $Nodes(e)$ denota el conjunto de nodos del arco, $Nodes(e) = \{n, m\}$. De la misma manera, dado un camino $p = \{m_0, e_1, m_1, \dots, e_k, m_k\}$, $Nodes(p)$ denota el conjunto de nodos del camino, $Nodes(p) = \{m_0, m_1, \dots, m_k\}$, y $Edges(e)$ denota el conjunto de arcos del camino, $Edges(p) = \{e_1, e_2, \dots, e_k\}$.

Cobertura y Correspondencia

Adicional a las definiciones dadas por Hidders [86, 87], este trabajo introduce la noción de **subgrafo cubierto** y **correspondencia de caminos**, necesarias para la definición de los operadores.

Sean I , una instancia del esquema S , y S' un subgrafo de S . **El subgrafo de I cubierto por S'** es el subgrafo que incluye los nodos y arcos en I relacionados con algún nodo o arco de S' .

Formalmente, sea $I = \{N_I, E_I, \lambda_I, \sigma_I, \rho_I\}$ una instancia del esquema S , ξ la relación de extensión de S a I y S' un subgrafo de S . Se dice que $I' = \{N_{I'}, E_{I'}, \lambda_{I'}, \sigma_{I'}, \rho_{I'}\}$ es el *subgrafo de I cubierto por S'* si las siguientes afirmaciones se cumplen:

- a) $N_{I'} = \{n \in N_I \mid \exists m, m \in N_{S'} \text{ y } \xi(m, n)\}$.
- b) $E_{I'} = \{(n_i, \alpha, n_j) \in E_I \mid \exists (m_k, \alpha, m_l) \in E_{S'}, \xi(m_k, n_i), \xi(m_l, n_j)\}$.
- c) Para cada $n \in N_{I'}$ con $\lambda_I(n) = \{c_1, c_2, \dots, c_k\}$, $\lambda_{I'}(n) = \{c_i \in \lambda_I(n) \mid \exists m, c_i = \lambda_{S'}(m) \text{ } m \in N_{S'}\}$
- d) Para cada $n \in N_{I'}$, $\sigma_{I'}(n) = \sigma_I(n)$.
- e) Para cada $e \in E_{I'}$, $\lambda_{I'}(e) = \lambda_I(e)$.
- f) Para cada $n \in N_{I'}$ such that $\sigma_{I'}(n) \in \mathcal{B}$, $\rho_{I'}(n) = \rho_I(n)$.

Adicionalmente, un camino $p_I = \{n_0, e_{I1}, n_1, \dots, e_{Ik}, n_k\}$ de un grafo instancia I **corresponde** con un camino $p_S = \{m_0, e_{S1}, m_1, \dots, e_{Sk}, m_k\}$ de un grafo esquema S si:

- a) El subgrafo formado por p_I es una instancia del subgrafo formado por p_S y
- b) Para todo j , $0 \leq j \leq k$, $\xi(m_j, n_j)\}$ y para todo l , $1 \leq l \leq k$, $\lambda_I(e_{Il}) = \lambda_S(e_{Sl})$.

5.2 Definición de los operadores de transformación de grafos

En esta sección se definen formalmente los operadores que GraphTQL ofrece al usuario final para la formulación de las consultas. Su definición se hace en términos de los operadores de nivel lógico que se presentan en la siguiente sección (Sección 5.3). Los operadores de GraphTQL se pueden traducir también a SPARQL usando la cláusula *construct*. Esta traducción se describe de manera informal para cada operador. Sin embargo, dado que la cláusula *construct* produce, en la mayoría de los motores de triplas, una lista de triplas, para implementar la composición de operadores sería

necesario representar la salida en memoria y/o disco. De esta manera se podría aplicar, sobre el resultado previo, las siguientes operaciones de la consulta.

5.2.1 Select a Portion

El operador *Select a Portion* permite seleccionar subgrafos de los grafos esquema e instancia.

Select a Portion toma como entrada un grafo esquema S , un grafo I que es instancia de S y un conjunto $\{e_{S1}, e_{S2}, \dots, e_{Sk}\}$ de arcos marcados en el diagrama del esquema (Ver ejemplo de la Figura 5.3a). Este conjunto de arcos debe formar un subgrafo conexo. El operador retorna los subgrafos S' e I' , S' es el subgrafo marcado en el diagrama e I' es el subgrafo de I cubierto por S' .

Select a Portion se define en términos del operador lógico *Subgraph Extraction* (Sección 5.3), mediante la expresión:

$$\text{Extract}(\langle S, I \rangle, \{e_{S1}, e_{S2}, \dots, e_{Sk}\}) \rightarrow \langle S', I' \rangle$$

Por ejemplo, dados los nodos y los arcos marcados (color rojo) en la Figura 5.3a, la expresión de la operación en el nivel lógico es:

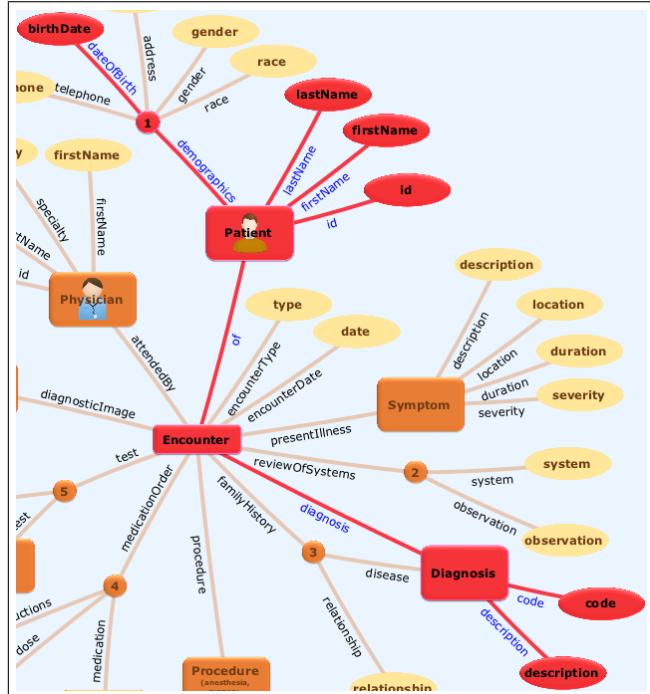
$$\begin{aligned} \text{Extract}(\langle S, I \rangle, & \{(Encounter, of, Patient), (Patient, demographics, \beta_1), \\ & (\beta_1, dateOfBirth, birthDate), (Patient, lastName, lastName), \\ & (Patient, firstName, firstName), (Encounter, diagnosis, Diagnosis), \\ & (Diagnosis, code, code), (Diagnosis, description, description)\}) \end{aligned}$$

El resultado, S' incluye (Figura 5.3b)

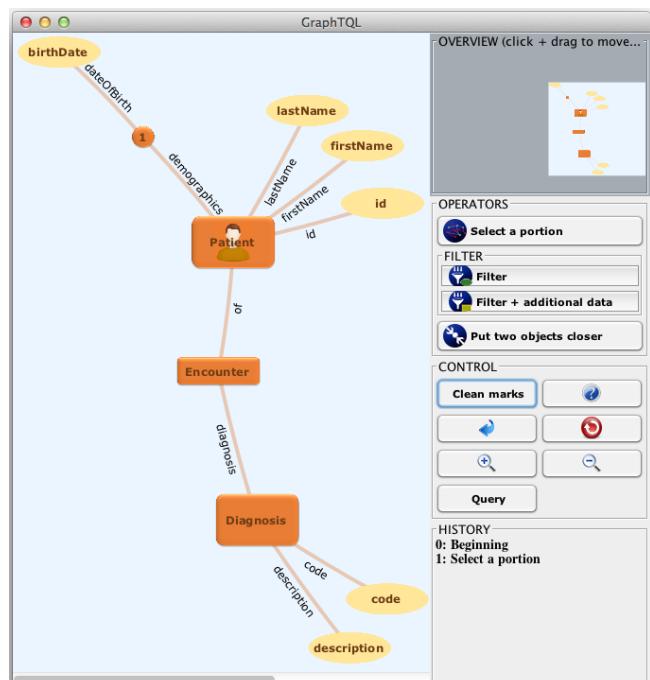
$$\begin{aligned} N_{S'} = & \{Encounter, Patient, \beta_1, birthDate, lastName, firstName, Diagnosis, \\ & code, description\} \\ E_{S'} = & \{(Encounter, of, Patient), (Patient, demographics, \beta_1), \\ & (\beta_1, dateOfBirth, birthDate), (Patient, lastName, lastName), \\ & (Patient, firstName, firstName), (Encounter, diagnosis, Diagnosis), \\ & (Diagnosis, code, code), (Diagnosis, description, description)\} \end{aligned}$$

En la instancia, I' , se recupera la fecha de nacimiento, el nombre y el apellido de los pacientes, los encuentros y el código y descripción de los diagnósticos. A diferencia del *pattern matching*, la operación no busca la coincidencia exacta del patrón marcado, cuando se selecciona una porción se recuperan todos los nodos y arcos que corresponden con algún nodo o arco marcado en el esquema, aún cuando formen un patrón incompleto.

La expresión del *Select a Portion* en SPARQL se forma por la unión de los arcos seleccionados. Por simplicidad, las expresiones de SPARQL que aparecen en este



(a)



(b)

Figura 5.3: Operador *Select a Portion*.

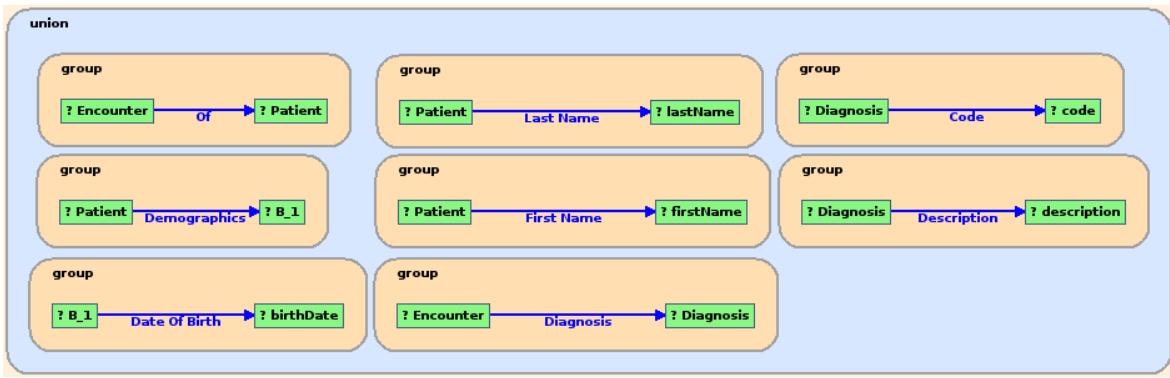


Figura 5.4: *Select a Portion* representado en Gruff.

documento usan “db:” como *namespace*. La expresión SPARQL para el ejemplo de la Figura 5.3a se muestra a continuación y su representación gráfica en Gruff, en la Figura 5.4.

```

SELECT * WHERE
{ {?Encounter db:of ?Patient} UNION {?Patient db:demographics ?dem}
UNION {?dem db:dateOfBirth ?birthDate}
UNION {?Patient db:lastName ?lastName}
UNION {?Patient db:firstName ?firstName}
UNION {?Encounter db:diagnosis ?Diagnosis}
UNION {?Diagnosis db:code ?code}
UNION {?Diagnosis db:description ?description } }

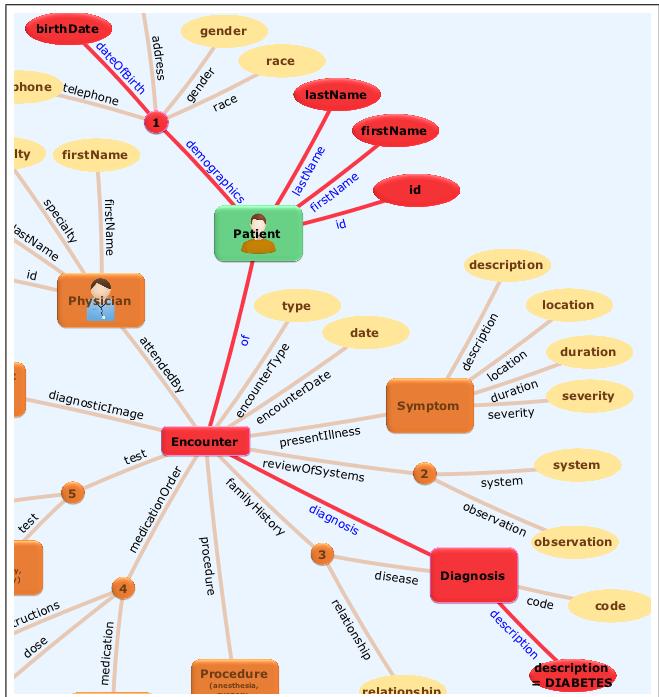
```

5.2.2 Filter

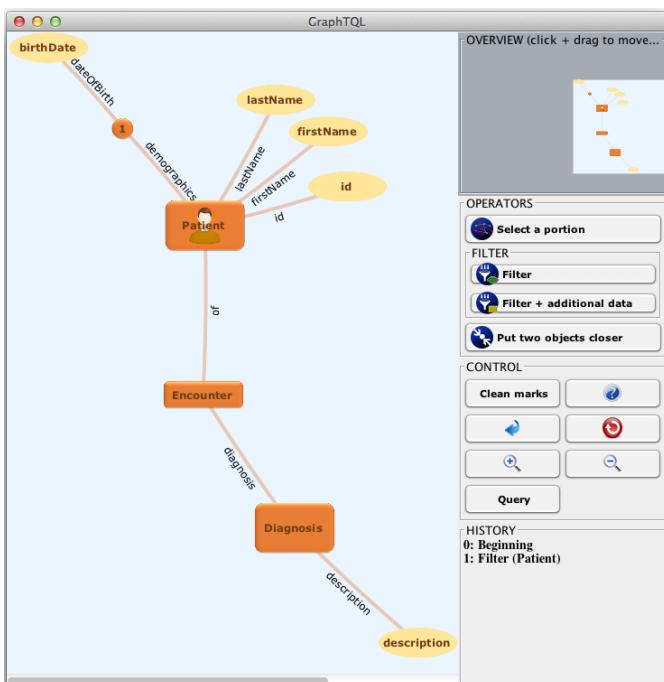
El operador *Filter* permite seleccionar las entidades una clase, llamada clase de interés, que cumplen una condición específica y otros datos conectados a dichas entidades.

Filter toma como entrada un grafo esquema S , un grafo I que es instancia de S , y los nodos y arcos marcados en el diagrama del esquema. Estos últimos incluyen: una clase de interés, m_I , un conjunto $Cc = \{c_1, c_2, \dots, c_k\}$ de nodos de valor básico con condición, un conjunto $Co = \{o_1, o_2, \dots, o_l\}$ de nodos diferentes a la clase de interés y a los nodos con condición, y un conjunto $Em = \{e_{S1}, e_{S2}, \dots, e_{St}\}$ de arcos que forman un grafo conexo e incluyen a los nodos marcados (i.e. dado $N_E = \bigcup_{e_S \in Em} Nodes(e_S)$, entonces $N_E = Cc \bigcup Co \bigcup m_I$).

Estas entradas se dividen en dos componentes: el **Patrón de Filtro** y el **Componente Opcional**. El primero, el patrón de filtro, está compuesto por la clase de interés, m_I , los nodos de valor básico con condición, Cc , y los arcos que forman



(a)



(b)

Figura 5.5: Operador *Filter*.

los caminos que conectan la clase de interés con los nodos con condición. El segundo, el componente opcional, está compuesto por la clase de interés, m_I , el conjunto C_o de nodos, y los arcos que forman los caminos que conectan la clase de interés con los nodos de C_o .

El operador *Filter* consta de dos etapas: En la primera, con base en el patrón de filtro, selecciona las entidades de la clase de interés que cumplen con la condición especificada y los caminos (de la instancia) que unen esas entidades con los nodos que son instancia de las condiciones marcadas en el esquema. Y en la segunda, recupera los datos del componente opcional disponibles en la instancia y que están conectados con algún nodo que aparece en el resultado de evaluar el patrón de filtro.

En el ejemplo de la Figura 5.5a la consulta recupera datos de pacientes con diagnóstico de “*DIABETES*”. La clase de interés *Patient* está marcada con color verde, el nodo con condición *description* está marcado con color rojo y la condición está escrita sobre él, los nodos del componente opcional (*birthDate*, *lastName*, *firstName* y *id*) están marcados con color rojo. Entonces, cuando se aplica el operador *Filter*, se buscan los pacientes (objetos de la clase *Patient*) que están conectados por el camino $\{Patient, of, Encounter, diagnosis, Diagnosis, description, description\}$ con un nodo de tipo *description* con valor “*DIABETES*”. Luego, se agregan los datos de fecha de nacimiento, nombres e identificación conectados a los pacientes recuperados con el patrón de filtro. El esquema de esta respuesta es el subgrafo marcado en el grafo esquema.

Para definir el *Filter* en términos de los operadores de nivel lógico, es necesario primero describir el proceso de clarificación de filtros y cómo su resultado se usa para dar forma a dicha expresión.

Generación del diálogo de clarificación de filtros

Los diálogos de clarificación de filtros son un mecanismo de desambiguación de las condiciones marcadas sobre el grafo esquema. Los diálogos guían al usuario cuando se requiere especificar, en los diferentes objetos involucrados en la consulta, si las condiciones se conectan con conjunción o con disyunción. Los puntos de desambiguación son los objetos donde se bifurcan los caminos que conectan las condiciones en el patrón del grafo.

Cuando el patrón de filtro incluye más de una condición (i.e. la carnalidad de C_c es mayor que uno) o cuando incluye una condición que se conecta por varios caminos con la clase de interés, se produce un diálogo de clarificación de filtros. El diálogo de clarificación de filtros es un proceso que toma como entrada el patrón de filtro. Un patrón sencillo, como el mostrado en la Figura 5.5a no requiere clarificación.

La versión actual de la generación de diálogos de clarificación de filtros soporta patrones de filtro que no contienen ciclos o patrones con ciclos que tienen a lo sumo una condición y dos nodos de bifurcación. En la Sección 5.4 se analizan opciones que

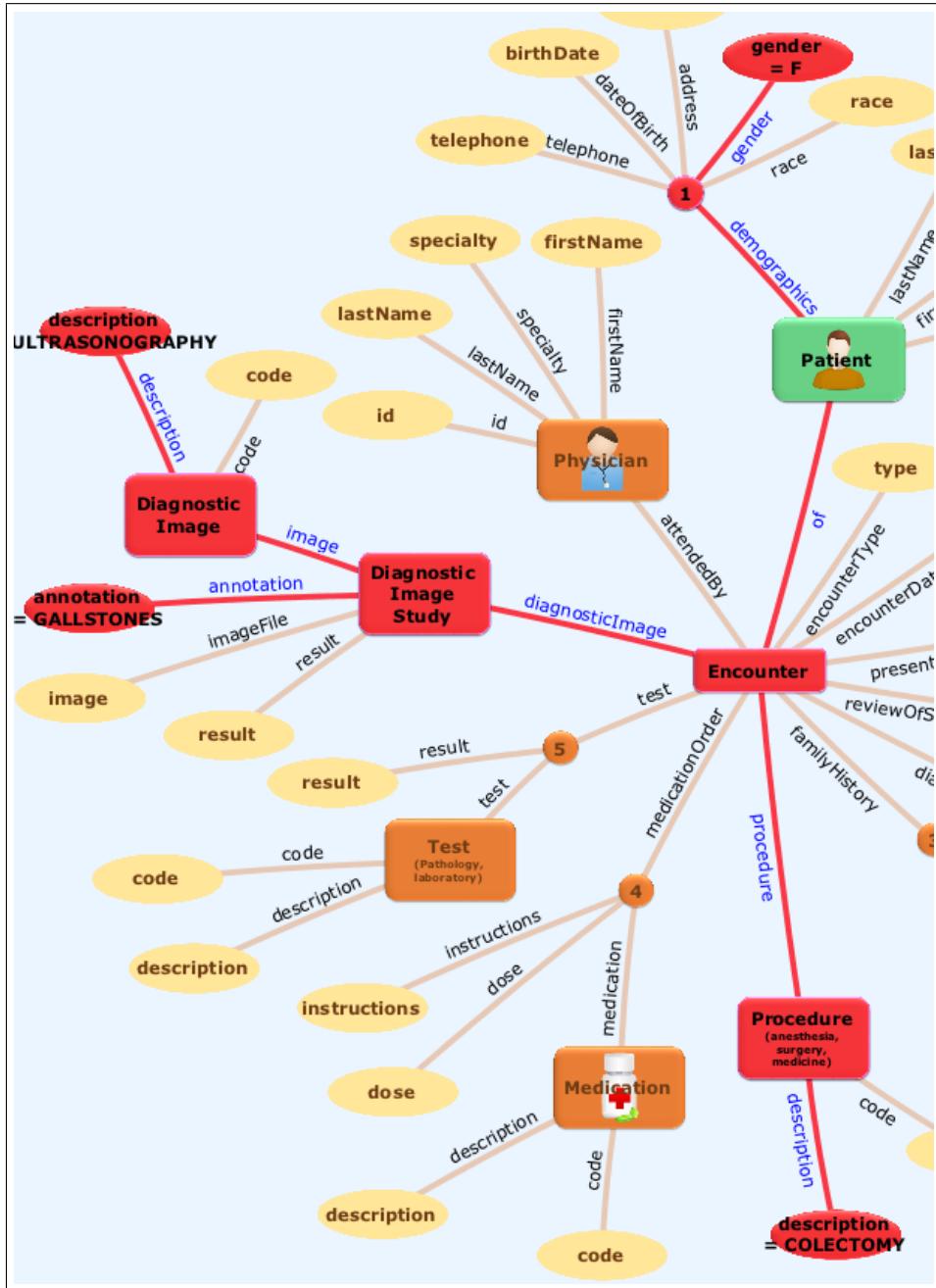


Figura 5.6: Ejemplo de consulta.

se podrían aplicar en otros casos.

Cuando el patrón de filtro tiene varias condiciones pero los caminos que las conectan con la clase de interés no forman ciclos (i.e. cada nodo con condición se conecta

con la clase de interés por medio de solo un camino simple no dirigido), el patrón es un árbol cuya raíz es la clase de interés. Por ejemplo, para encontrar los pacientes de género femenino que han tenido una ecografía (“ULTRASONOGRAPHY”) con anotación “GALLSTONES” y se les ha realizado el procedimiento “COLECTOMY”, se marca el patrón de filtro que se muestra en la Figura 5.6. En este ejemplo, el estudio de imágenes diagnósticas y el procedimiento pueden corresponder a encuentros diferentes (i.e. momentos diferentes), pero se requiere que el paciente cumpla con ambas condiciones.

Dado un patrón con forma de árbol cuya raíz es m_I , la clase de interés, y cuyas hojas son $Cc = \{c_1, c_2, \dots, c_k\}$, los nodos con condición. Sea Ec el conjunto de arcos que conectan la clase de interés con los nodos con condición. El generador del diálogo de clarificación de filtros hace un recorrido arriba-abajo del patrón, iniciando en la clase de interés y recorriendo los arcos de Ec para identificar los **nodos de bifurcación**. Un nodo de bifurcación b es un nodo en el que inciden tres o más arcos de Ec , es decir, además del camino que lo conecta con la clase de interés, tiene por lo menos dos caminos simples no dirigidos que lo conectan con nodos con condición. Cuando estos caminos incluyen otros nodos de bifurcación, se dice que b es un **nodo de bifurcación intermedio**, de lo contrario b es un **nodo de bifurcación final**. Notese que los nodos de bifurcación intermedios están conectados a otros nodos de bifurcación. Adicionalmente, la clase de interés m_I se considera como un nodo de bifurcación. En el ejemplo, la clase de interés es *Patient*, los nodos de bifurcación son $\{Patient, Encounter, DiagnosticImageStudy\}$, *Patient* y *Encounter* son nodos de bifurcación intermedios.

El diálogo de clarificación consta de varias preguntas formuladas al usuario, una por cada nodo de bifurcación. Las preguntas se realizan haciendo un recorrido abajo-arriba del patrón, por lo tanto, primero se realizan las preguntas correspondientes a los nodos de bifurcación final.

Sea b_f un nodo de bifurcación final que se conecta con los nodos con condición $\{c_{f1}, \dots, c_{fj}\}$, sea f_i la condición especificada para el nodo c_{fi} ($1 \leq i \leq j$). Para el nodo b_f se genera un diálogo con dos opciones, que se muestra en la Figura 5.7.

The $\langle \lambda_S(b_f) \rangle$ should MATCH ALL conditions
 $(\langle \lambda_S(c_{f1}) \rangle \langle f_1 \rangle \text{ AND } \dots \text{ AND } \langle \lambda_S(c_{fj}) \rangle \langle f_j \rangle)$

The $\langle \lambda_S(b_f) \rangle$ should MATCH ANY conditions
 $(\langle \lambda_S(c_{f1}) \rangle \langle f_1 \rangle \text{ OR } \dots \text{ OR } \langle \lambda_S(c_{fj}) \rangle \langle f_j \rangle)$

Figura 5.7: Clarificación de un nodo de bifurcación final.

El diálogo generado para el nodo de bifurcación final, *DiagnosticImageStudy*, de la consulta del ejemplo se muestra en la Figura 5.8.

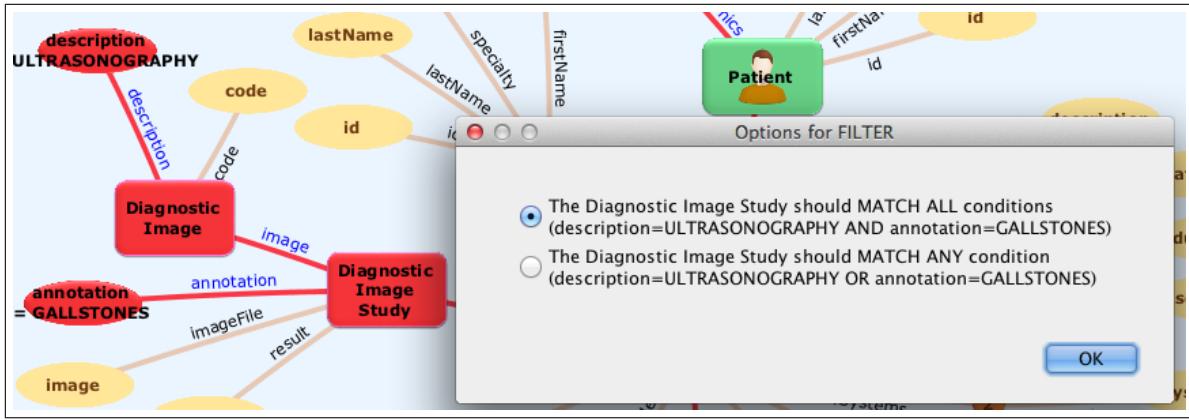


Figura 5.8: Diálogo para un nodo de bifurcación final.

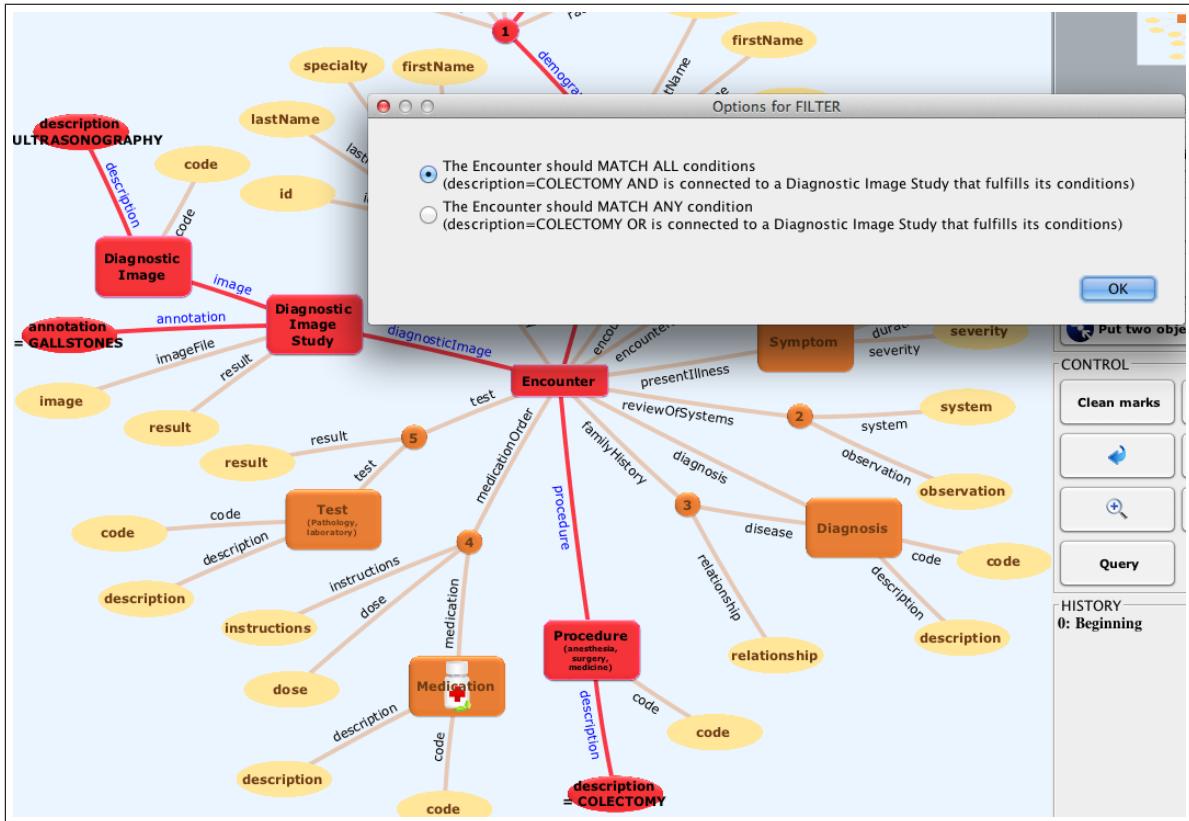


Figura 5.9: Diálogo para un nodo de bifurcación intermedio.

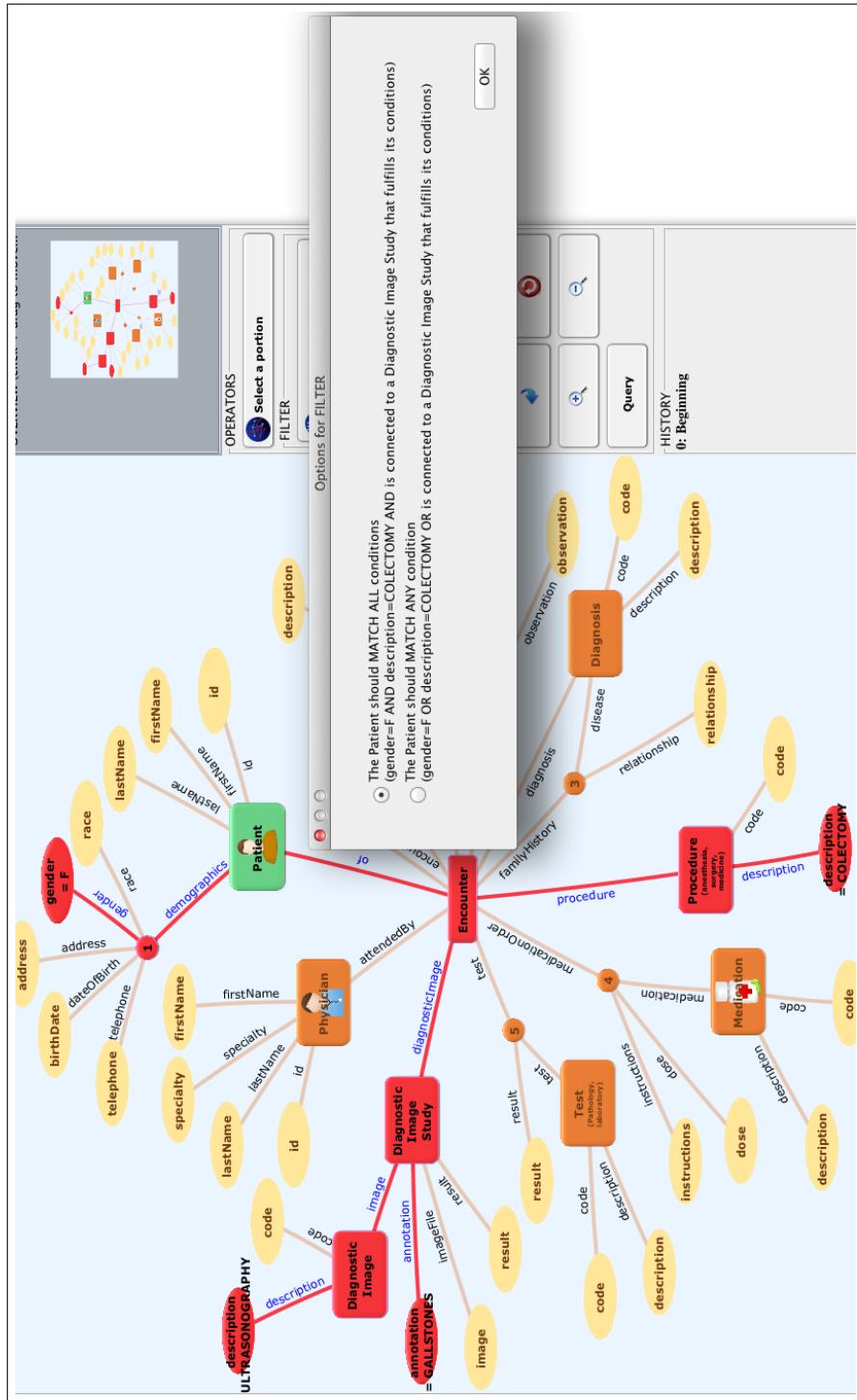


Figura 5.10: Diálogo en el nodo de bifurcación *Patient*.

Un nodo de bifurcación intermedio b_i está conectado con otros nodos de bifurcación, $\{b_1, \dots, b_j\}$, y con los nodos con condición $\{c_{i1}, \dots, c_{iq}\}$ de manera directa, es decir por caminos que no incluyen nodos de bifurcación. Las opciones del diálogo de clarificación son las mismas que para el nodo de bifurcación final: (a) “*The $\langle \lambda_S(b_f) \rangle$ should MATCH ALL conditions*”, o (b) “*The $\langle \lambda_S(b_f) \rangle$ should MATCH ANY condition*”. Para los nodos con condición, $\{c_{i1}, \dots, c_{iq}\}$ (i.e. los conectados de manera directa), las condiciones se expresan igual que en los nodos de bifurcación final : “ $\langle \lambda_S(c_{ik}) \rangle \langle f_k \rangle$ ”, donde que f_k es la condición de filtro del nodo c_{ik} para $1 \leq k \leq q$.

La expresión de la condición para un nodo b_t ($b_t \in \{b_1, \dots, b_j\}$) depende de la respuesta obtenida en el diálogo de ese nodo. Si se eligió la conjunción, entonces la condición se expresa como: “*is connected to a $\langle \lambda_S(b_t) \rangle$ that fulfills its conditions*”. Dado que b_t debe cumplir con todas las condiciones (conjunción), entonces b_i también cumplirá con el grupo de condiciones de b_t . De otra parte, si en el diálogo de b_t se eligió la disyunción, entonces la pregunta se formula incluyendo cada una de las condiciones de b_t , como si los nodos con condición correspondientes estuvieran conectados directamente con b_i . Dado que, en este caso, b_t no impone una restricción adicional (se pueden cumplir algunas de las condiciones), se ofrece al usuario la posibilidad de decidir si aplica conjunción o disyunción de las condiciones en el nodo de bifurcación b_i . Esto permite, por ejemplo, en la consulta de la Figura 5.6 decidir que el procedimiento (*COLECTOMY*) y la imagen diagnóstica (*ULTRASONOGRAPHY*) pueden estar registrados en encuentros diferentes (disyunción en el nodo *Encounter*); pero que el paciente cumple con todas las condiciones (conjunción).

La Figura 5.9 muestra el diálogo de clarificación de *Encounter* para el ejemplo anterior, dado que la respuesta para el diálogo de *DiagnosticImageStudy* es conjunción (*description* = “*ULTRASONOGRAPHY*” AND *annotation* = “*GALLSTONES*”). La respuesta en *Encounter* es disyunción (*is connected to a Diagnosis that fulfills its conditions OR description* = “*COLECTOMY*”), ya que la imagen y el procedimiento se pudieron realizar en encuentros diferentes. Luego, el diálogo de clarificación en *Patient* toma las condiciones de *Encounter* como si *Patient* estuviera conectado a esos nodos con condición directamente y agrega la condición del género (Figura 5.10). En paciente se aplica la conjunción de las condiciones porque, en el ejemplo, se buscan aquellos que las cumplan todas.

Finalmente, cuando el patrón de filtro contiene ciclos pero tiene máximo un nodo con condición y dos nodos de bifurcación, se realiza un diálogo de clarificación diferente. En este caso, uno de los nodos de bifurcación es la clase de interés. Cuando hay dos nodos de bifurcación, la clase de interés es el nodo de bifurcación intermedio y el otro, es de bifurcación final. Sea $P = \{p_1, p_2, \dots, p_q\}$ el conjunto de caminos simples no dirigidos que unen la clase de interés, m_I , con el nodo con condición, c_1 , y sea b_f el nodo de bifurcación final de los caminos de P , se genera un diálogo de clarificación en el nodo de bifurcación b_f . Las condiciones de este diálogo se forman con base en los caminos, para el camino p_i ($1 \leq i \leq q$) la condición es “*connected with $\langle \lambda_S(c_1) \rangle$ by $\langle p_i \rangle$* ”. El

The $\langle \lambda_S(b) \rangle$ should MATCH ALL conditions
 (connected with $\langle \lambda_S(c_1) \rangle$ by $\langle p_1 \rangle$ AND... AND connected with $\langle \lambda_S(c_1) \rangle$ by $\langle p_q \rangle$)

The $\langle \lambda_S(b) \rangle$ should MATCH ANY conditions
 (connected with $\langle \lambda_S(c_1) \rangle$ by $\langle p_1 \rangle$ OR... OR connected with $\langle \lambda_S(c_1) \rangle$ by $\langle p_q \rangle$)

Figura 5.11: Clarificación de una consulta que incluye un ciclo y una condición.

diálogo de clarificación de b_f se muestra en la Figura 5.11. Cuando el usuario elige la disyunción en b_f y hay dos nodos de bifurcación, la misma pregunta se realiza en el nodo de bifurcación intermedio, que sería la clase de interés.

La Figura 5.12 muestra un ejemplo de este tipo de consulta y de diálogo de clarificación.

Generación de la expresión de nivel lógico para el *Filter*

El operador *Filter* se define mediante la composición de dos operadores de nivel lógico: *Logic Level Filter* y *Selective Union*. El *Logic Level Filter* calcula el resultado correspondiente al patrón de filtro. El *Selective Union*, agrega los datos correspondientes al componente opcional. La definición formal de estos operadores se presenta en la Sección 5.3.

El *Logic Level Filter* busca en la instancia I las entidades de la clase de interés m_I que cumplen con una condición exp especificada, y retorna el conjunto de entidades junto con los caminos que satisfacen la condición.

$$LogicLFilter(\langle S, I \rangle, m_I, exp) \rightarrow \langle S', I' \rangle$$

Por su parte, *Selective Union*, recibe dos parejas de grafos esquema e instancia $\langle S, I \rangle$ y $\langle S', I' \rangle$, una clase de interés m_u y un conjunto de caminos simples y no dirigidos P_S que inician en la clase de interés. El resultado incluye $\langle S', I' \rangle$ con los datos de $\langle S, I \rangle$ que corresponden con algún camino que inicia con una entidad de la clase de interés que existe en I y en I' (La correspondencia se define en la Sección 5.1).

$$SelUnion(\langle S, I \rangle, \langle S', I' \rangle, m_u, P_S) \rightarrow \langle S', I' \rangle$$

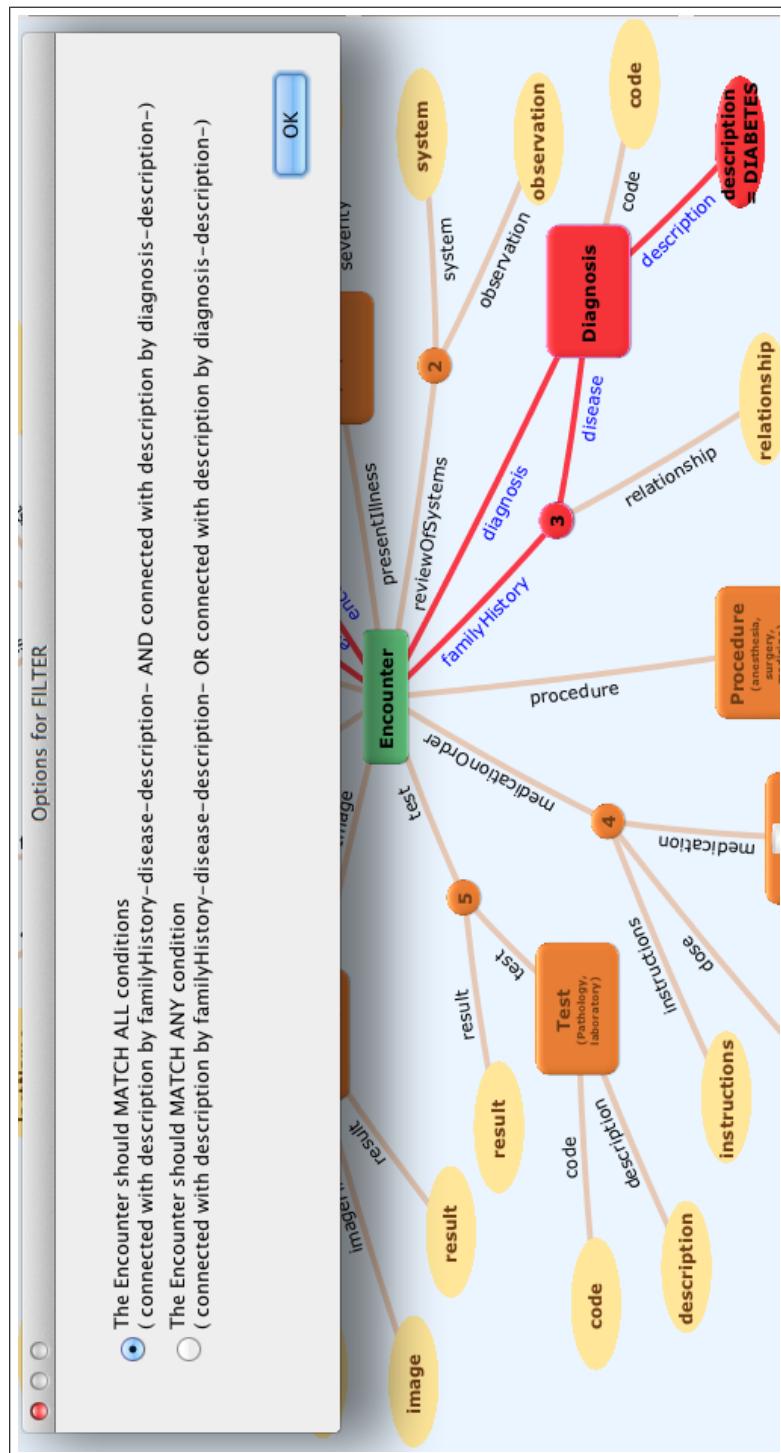


Figura 5.12: Diálogo cuando la consulta incluye un ciclo y una condición.

Generación de las expresiones de *Logic Level Filter*

La expresión del *Logic Level Filter* permite combinar condiciones especificadas sobre nodos de valor básico con filtros anidados o subfiltros, de esta manera se soportan las opciones elegidas en los diálogos de los puntos de bifurcación del patrón de filtro. Para ejecutar la consulta, se genera un subfiltro por cada nodo de bifurcación donde se aplica conjunción, y un filtro para la clase de interés.

El patrón de filtro sin ciclos se puede abstraer como un árbol formado por la clase de interés, los nodos de bifurcación y los nodos con condición. La clase de interés se considera un nodo de bifurcación y es la raíz de este árbol, los nodos con condición son las hojas y los nodos de bifurcación son nodos internos. Los hijos de un nodo son los nodos conectados directamente a él (i.e. por un camino que no incluye otros nodos de bifurcación). Con base en esta estructura de árbol se genera la expresión de filtro. Cada nodo de bifurcación cuya respuesta en el diálogo fue la conjunción genera una expresión de *Logic Level Filter* cuya clase de interés es el nodo de bifurcación. Con excepción de la clase de interés, los nodos de bifurcación cuya respuesta en el diálogo fue la disyunción no generan expresión de filtro, las condiciones se evalúan en el nodo que es su padre en el árbol.

Dado un nodo de bifurcación final b_f para el cual se eligió la conjunción, sean $\{c_{f1}, \dots, c_{fj}\}$ los nodos con condición con los que se conecta, $\{p_i\}$ el camino que conecta b_f con c_{fi} y f_i la condición especificada para el nodo c_{fi} ($1 \leq i \leq j$). f_i tiene la forma $<op_i\;o_i>$ donde op_i es un operador y o_i un operando (Los operadores soportados se especifican en la Tabla 5.1 de la Sección 5.3.2). Entonces, para el nodo b_f se genera la siguiente expresión de filtro:

$$\textit{LogicLFilter}(\langle S, I \rangle, b_f, (\langle c_{f1\{p_1\}} \rangle \langle f_1 \rangle \textit{AND} \dots \textit{AND} \langle c_{fj\{p_j\}} \rangle \langle f_j \rangle))$$

Sean b_i un nodo de bifurcación intermedio para el cual se eligió la conjunción, $\{b_1, \dots, b_t\}$ los nodos de bifurcación que están conectados directamente con b_i , $\{c_{i1}, \dots, c_{iq}\}$ los nodos con condición conectados directamente con b_i , y p_k el camino que conecta b_i con b_k ($1 \leq k \leq t$), p_{cv} el camino que conecta b_i con c_{iv} y f_v la condición especificada para el nodo c_{iv} ($1 \leq v \leq q$). Para cada nodo b_k se tienen dos posibilidades: se aplica conjunción o disyunción. Cuando para b_k se aplica conjunción, se genera un subfiltro, cuyo resultado es $\langle S_k, I_k \rangle$. En este caso, en b_i se aplica el operador *inSubgraph* para seleccionar aquellas instancias de b_k que están en I_k . De otra parte, cuando para b_k se aplica disyunción no se genera subfiltro y las condiciones de b_k se aplican como si estuviesen conectadas directamente con el nodo b_i .

Entonces, si para todos los nodos $\{b_1, \dots, b_t\}$ se aplica conjunción se genera para b_i la siguiente expresión de filtro:

$$\textit{LogicLFilter}(\langle S, I \rangle, b_i, (\langle c_{i1\{p_{c1}\}} \rangle \langle f_1 \rangle \textit{AND} \dots \textit{AND} \langle c_{iq\{p_{cq}\}} \rangle \langle f_q \rangle \textit{AND}$$

$\langle b_{1\{p_1\}} \rangle INSUBGRAPH \langle S_1, I_1 \rangle AND \dots AND \langle b_{t\{p_t\}} \rangle INSUBGRAPH \langle S_t, I_t \rangle)$

Por el contrario, si para todos los nodos $\{b_1, \dots, b_t\}$ se aplica disyunción, sea $\{c_{k_1}, c_{k_2}, \dots, c_{k_l}\}$ las condiciones conectadas al nodo b_k por los caminos $\{p_{k_1}, p_{k_2}, \dots, p_{k_l}\}$ ($1 \leq k \leq t$). Entonces, se genera para b_i , que aplica conjunción, la siguiente expresión de filtro:

$$\begin{aligned} LogicLFilter(\langle S, I \rangle, b_i, & \langle c_{i1\{p_{c1}\}} \rangle \langle f_1 \rangle AND \dots AND \langle c_{iq\{p_{cq}\}} \rangle \langle f_q \rangle \\ & AND (\langle c_{1_1\{p_1||p_{1_1}\}} \rangle \langle f_{1_1} \rangle AND \dots AND \langle c_{t_1\{p_t||p_{t_1}\}} \rangle \langle f_{t_1} \rangle)) \end{aligned}$$

donde, $c_{k_j\{p_k||p_{k_j}\}} \langle f_{k_j} \rangle$ representa la condición j del nodo b_k ($1 \leq k \leq t$) y “||” representa la concatenación de caminos, por tanto $\{p_k||p_{k_j}\}$ es la concatenación de los caminos p_k y p_{k_j} , el primero conecta los nodos b_i y b_k y el segundo, b_k con la clase con condición c_{k_j} .

La expresión de filtro de la clase de interés se genera de la misma manera que para un nodo intermedio. Cuando para la clase de interés se aplica disyunción, se genera la misma expresión con el operador lógico *OR*. Este es el único caso en el cual la disyunción genera una expresión de filtro.

En el ejemplo de la subsección anterior (Figura 5.6), se generan las siguientes expresiones de filtro:

Para el nodo de bifurcación final *DiagnosticImageStudy*, que aplica conjunción,

$$\begin{aligned} LogicLFilter(\langle S, I \rangle, DiagnosticImageStudy, & \\ (description_{\{DiagnosticImageStudy, image, DiagnosticImage, description, description\}} & = "ULTRASONOGRAPHY" AND \\ annotation_{\{DiagnosticImageStudy, annotation, annotation\}} & = "GALLSTONES")) \\ \rightarrow \langle S_1, I_1 \rangle \end{aligned}$$

El nodo de bifurcación intermedio *Encounter* no genera filtro porque aplica disyunción. Por tanto, las condiciones conectadas a este nodo pasan al nodo padre, en este caso, *Patient*.

Para el nodo de bifurcación intermedio *Patient*, que aplica conjunción

$$\begin{aligned} LogicLFilter(\langle S, I \rangle, Patient, & \\ (DiagnosticImageStudy_{\{Patient, of, Encounter, diagnosticImage, DiagnosticImageStudy\}} & \\ inSubgraph \langle S_1, I_1 \rangle) AND & \\ description_{\{Patient, of, Encounter, procedure, Procedure, description, description\}} & = \\ "COLECTOMY" AND gender_{\{Patient, demographics, \beta_1, gender, gender\}} & = "F") \\ \rightarrow \langle S_2, I_2 \rangle \end{aligned}$$

Generación de la expresiones de *Selective Union*

Sean *Co* y *Eo* los conjuntos de nodos y arcos del componente opcional de un filtro, las expresiones de *Selective Union* se generan a partir de los caminos simples no dirigidos

$P = \{p_1, p_2, \dots, p_t\}$ tal que todo $p \in P$ une un nodo de Co con algún nodo del patrón de filtro. Estos nodos del patrón de filtro son los nodos de conexión $Cx = \{x_1, x_2, \dots, x_k\}$, por cada uno de ellos se genera un *Selective Union*. Cada *Selective Union* se aplica sobre dos parejas de grafos $\langle S, I \rangle, \langle S', I' \rangle$, $\langle S, I \rangle$ es el grafo sobre el que se está aplicando el *Filter* y $\langle S', I' \rangle$ es, para x_1 el resultado del patrón de filtro y, para x_i ($2 \leq i \leq k$), es el resultado de un *Selective Union*. Sea P_{xi} el conjunto de caminos que une el nodo x_i con algún nodo de Co ($1 \leq i \leq k$) tal que $p \in P_{xi}$ si p no es subcamino de otro camino de P . Entonces, la siguiente es la expresión *Selective Union* para el nodo x_i ,

$$SelUnion(\langle S, I \rangle, \langle S', I' \rangle, x_i, P_{xi})$$

Agregando al ejemplo de la subsección anterior (Figura 5.6), como componente opcional, los arcos $\{(Patient, id, id), (Encounter, attendedBy, Physician), (Physician, specialty, specialty), (Encounter, encounterDate, date)\}$ (Figura 5.13), se encuentra que este conjunto de nodos que se agrega conforma los caminos:

$$\begin{aligned} p_1 &= \{Patient, id, id\}, p_2 = \{Encounter, encounterDate, date\} \\ p_3 &= \{Encounter, attendedBy, Physician\} \\ \text{y } p_4 &= \{Encounter, attendedBy, Physician, specialty, specialty\}. \end{aligned}$$

Estos caminos se conectan al patrón de filtro mediante dos nodos de conexión: *Patient* y *Encounter*. Cada uno de ellos genera una expresión *Selective Union*, así:

Para *Patient*,

$$SelUnion(\langle S, I \rangle, \langle S_2, I_2 \rangle, Patient, \{\{Patient, id, id\}\}) \rightarrow \langle S_2, I_2 \rangle$$

Para *Encounter*,

$$SelUnion(\langle S, I \rangle, \langle S_2, I_2 \rangle, Encounter, \{\{Encounter, encounterDate, date\}, \{Encounter, attendedBy, Physician, specialty, specialty\}\}) \rightarrow \langle S_2, I_2 \rangle$$

Nótese que el camino p_3 , por ser subcamino de p_4 , no se incluye en el *Selective Union* generado para *Encounter*.

Entonces, la expresión de nivel lógico completa para el ejemplo de la Figura 5.13 es:

$$\begin{aligned} &LogicLFilter(\langle S, I \rangle, DiagnosticImageStudy, \\ &(description_{\{DiagnosticImageStudy, image, DiagnosticImage, description, description\}} = \\ &\quad "ULTRASONOGRAPHY" \text{ AND} \\ &\quad annotation_{\{DiagnosticImageStudy, annotation, annotation\}} = "GALLSTONES") \rightarrow \langle S_1, I_1 \rangle \\ &LogicLFilter(\langle S, I \rangle, Patient, \\ &\quad (DiagnosticImageStudy_{\{Patient, of, Encounter, diagnosticImage, DiagnosticImageStudy\}} \\ &\quad inSubgraph\langle S_1, I_1 \rangle) \text{ AND} \\ &\quad description_{\{Patient, of, Encounter, procedure, Procedure, description, description\}} = \\ &\quad "COLECTOMY" \text{ AND } gender_{\{Patient, demographics, \beta_1, gender, gender\}} = "F") \rightarrow \langle S_2, I_2 \rangle \\ &SelUnion(\langle S, I \rangle, \langle S_2, I_2 \rangle, Patient, \{\{Patient, id, id\}\}) \rightarrow \langle S_2, I_2 \rangle \\ &SelUnion(\langle S, I \rangle, \langle S_2, I_2 \rangle, Encounter, \{\{Encounter, encounterDate, date\}, \\ &\quad \{Encounter, attendedBy, Physician, specialty, specialty\}\}) \rightarrow \langle S_2, I_2 \rangle \end{aligned}$$

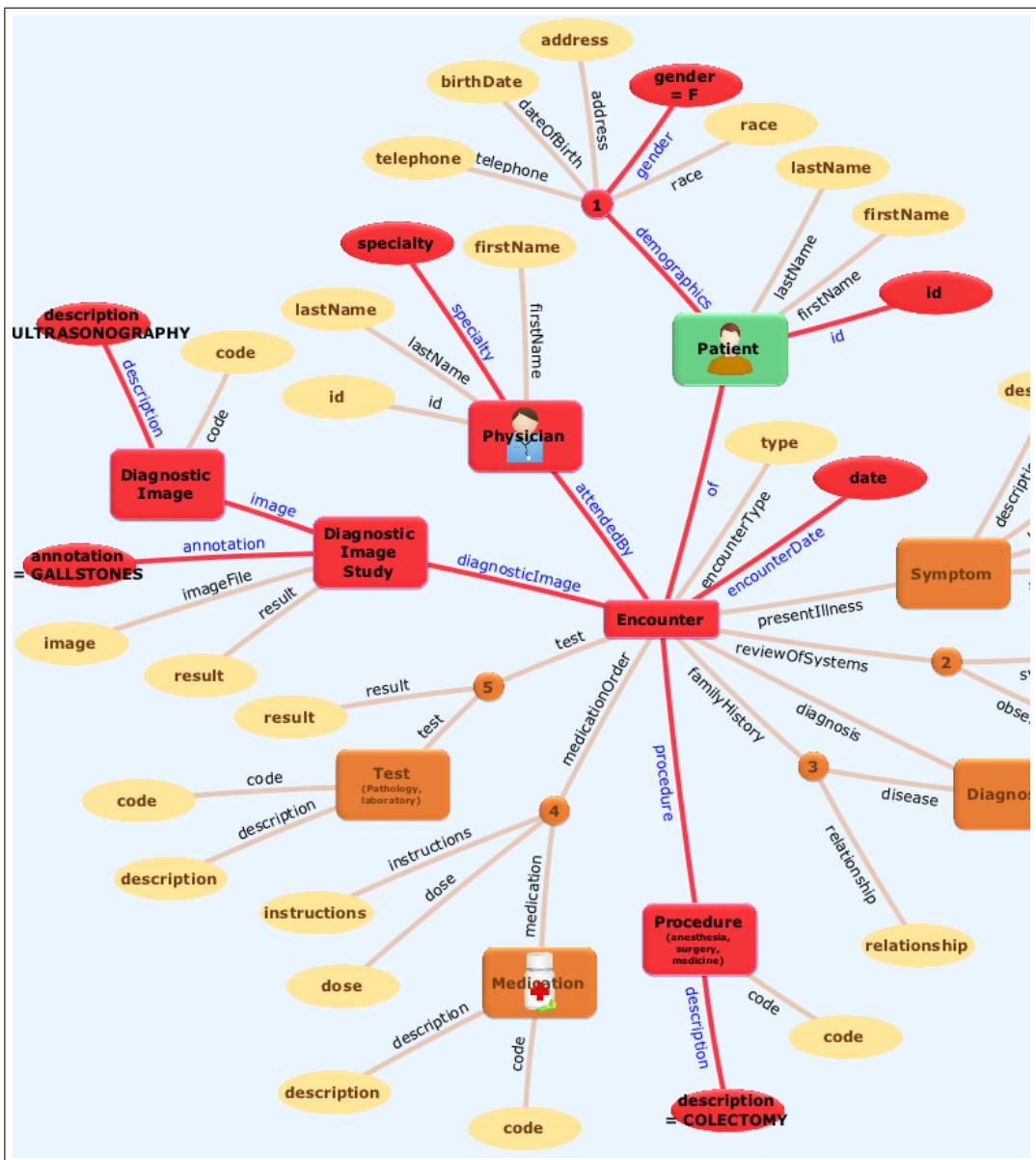


Figura 5.13: Consulta que incluye componentes opcionales.

Filter expresado en SPARQL

El patrón de filtro se puede expresar en SPARQL con patrones básicos (*pattern matching*) o con patrones básicos y la cláusula *union*, si en la clase de interés se aplica disyunción de las condiciones. El componente opcional se expresa con la cláusula *optional*.

Dado el árbol formado por la clase de interés, los nodos de bifurcación y los nodos con condición del patrón del filtro, y la operación seleccionada para cada nodo de bifurcación (*AND/OR*), el patrón de consulta se forma así: Cada nodo de bifurcación *b* en que se aplica conjunción genera un patrón de consulta formado por un *join* de subpatrones. Hay un subpatrón por cada nodo con condición conectado con *b*, formado por el camino que conecta *b* con el nodo con condición. Los nodos de bifurcación donde se aplica disyunción pasan las condiciones a su nodo padre en el árbol. En este caso, si se tienen *n* condiciones conectadas al nodo de bifurcación *b*, se tienen *n* variables que representan *b* (ej. si el nodo de bifurcación es *Encounter*, se podrían tener las variables *Encounter1* y *Encounter2*). Cada una de estas variables se incluye en un subpatrón formado por el camino que conecta *b* con uno de sus nodos con condición. Esto se aplica para todos los nodos de bifurcación, tanto los intermedios como los finales, con excepción de la clase de interés. Cuando en la clase de interés se aplica disyunción, se requiere solamente una variable para representarla y el patrón de consulta está formado por la *union* de los subpatrones correspondientes a cada condición.

El componente opcional también se genera a partir de los nodos de conexión, generando un patrón para cada camino de P_{xi} (siendo P_{xi} el conjunto que se describió en la generación de expresiones de *Selective Union*, página 112). Sin embargo, cuando un nodo de conexión *x* es a su vez un nodo de bifurcación en el que se aplicó disyunción, es necesario generar varias copias del mismo patrón, una por cada nombre de variable que hace referencia a *x*.

La expresión SPARQL para el ejemplo de la 5.13, aplicando conjunción en *DiagnosticImageStudy*, disyunción en *Encounter* y conjunción en *Patient* se muestra a continuación y su representación gráfica en Gruff, en la Figura 5.14

```
SELECT * WHERE
{ ?Patient db:demographics ?dem . ?dem db:gender db:F .
?EncounterID db:of ?Patient .
?EncounterID db:diagnosticImage ?DiagnosticImageStudy .
{ ?DiagnosticImageStudy db:image ?DiagnosticImage .
?DiagnosticImage db:description db:ULTRASOUND .
?DiagnosticImageStudy db:annotation db:GALLSTONES) } .
?EncounterPR db:of ?Patient .
?EncounterPR db:procedure ?Procedure .
?Procedure db:description db:COLECTOMY }
OPTIONAL { ?Patient db:id ?id }
OPTIONAL { ?EncounterID db:attendedBy ?PhysicianID . }
```

```

?PhysicianID db:specialty ?specialtyID }
OPTIONAL { ?EncounterID db:encounterDate ?dateID }
OPTIONAL { ?EncounterPR db:attendedBy ?PhysicianPR .
?PhysicianPR db:specialty ?specialtyPR }
OPTIONAL { ?EncounterPR db:encounterDate ?datePR } ]

```

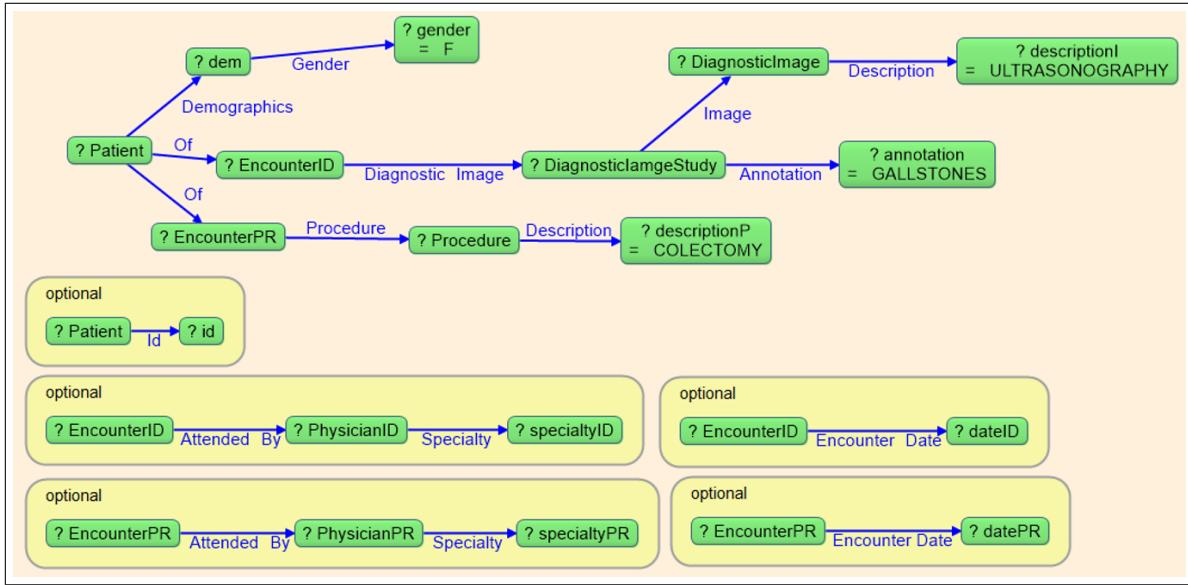


Figura 5.14: Ejemplo de *Filter* expresado en SPARQL.

5.2.3 *Filter+Additional Data*

Similar al *Filter*, *Filter+Additional Data* selecciona las entidades de una clase de interés que cumplen una condición específica y otros datos conectados a ellas. Se diferencia del *Filter* en que todos los nodos y arcos marcados, diferentes a la clase de interés, hacen parte del componente opcional, incluidos los nodos con condición. Por ello, se recuperan valores diferentes a los especificados en la condición. Es útil, por ejemplo, cuando se necesita recuperar la historia clínica de los pacientes que han sido diagnosticados con “*DIABETES*”. La historia clínica incluye otros diagnósticos, diferentes a la “*DIABETES*”, otros encuentros, diferentes a aquellos donde se diagnosticó la diabetes, y los datos relacionados con esos otros encuentros.

Filter+Additional Data tiene la misma entrada y etapas del *Filter*. La entrada se compone de los grafos esquema e instancia, *S* e *I*, la clase de interés, m_I , un conjunto *Cc* de nodos de valor básico con condición, un conjunto *Co* de nodos diferentes a la clase de interés y a los nodos con condición, y un conjunto *Em* de arcos que forman un

grafo conexo e incluyen a los nodos marcados. Al igual que en el *Filter*, esta entrada también forma un patrón de filtro y un componente opcional. El patrón de filtro incluye la clase de interés, las clases con condición y los caminos que unen a estas últimas con las clases condición. El componente opcional, a diferencia del *Filter*, incluye además de *Co*, los nodos de *Cc* y el todo el conjunto de arcos (*Em*).

El diálogo de clarificación de filtros se realiza de la misma manera como se describió para el *Filter*, al igual que la generación de las expresiones de *Logic Level Filter*. La diferencia radica en la generación de la expresión *Selective Union*, que en este caso es una sola, cuya clase de interés es m_I y cuyo conjunto de caminos P incluye los caminos que unen m_I con algún nodo de $(Co \cup Cc)$ tal que $p \in P$ si p no es subcamino de un camino $q \in P$, con $p \neq q$.

Dada la entrada marcada en la Figura 5.13, con conjunción en *DiagnosticImageStudy*, disyunción en *Encounter* y conjunción en *Patient*, la expresión de nivel lógico para el *Filter+Additional Data* es:

```

LogicLFilter(⟨S, I⟩, DiagnosticImageStudy,
  (description{DiagnosticImageStudy,image,DiagnosticImage,description,description} =
   “ULTRASONOGRAPHY” AND
   annotation{DiagnosticImageStudy,annotation,annotation} = “GALLSTONES”)) )
  → ⟨S1, I1⟩

LogicLFilter(⟨S, I⟩, Patient,
  (DiagnosticImageStudy{Patient,of,Encounter,diagnosticImage,DiagnosticImageStudy}
   inSubgraph ⟨S1, I1⟩) AND
   description{Patient,of,Encounter,procedure,Procedure,description,description} =
   “COLECTOMY” AND gender{Patient,demographics,β1,gender,gender} = “F”)
  → ⟨S2, I2⟩)

SelUnion(⟨S, I⟩, ⟨S2, I2⟩, Patient, { {Patient, id},
  {Patient, demographics, β1, gender, gender},
  {Patient, of, Encounter, encounterDate, date},
  {Patient, of, Encounter, attendedBy, Physician, specialty, specialty},
  {Patient, of, Encounter, procedure, Procedure, description, description},
  {Patient, of, Encounter, diagnosticImage, DiagnosticImageStudy, image,
   DiagnosticImage, description, description},
  {Patient, of, Encounter, diagnosticImage, DiagnosticImageStudy,
   annotation, annotation}}) → ⟨S2, I2⟩

```

Filter+Additional Data se puede expresar en SPARQL de manera similar a como se explicó para el *Filter*, con patrones básicos y cláusulas *union* y *optional*. Para el ejemplo la expresión de la consulta en SPARQL es la siguiente:

```

SELECT * WHERE
{ ?Patient db:demographics ?dem . ?dem db:gender db:F .
?EncounterID db:of ?Patient .

```

```

?EncounterID db:diagnosticImage ?DiagnosticImageStudy .
{   ?DiagnosticImageStudy db:image ?DiagnosticImage .
    ?DiagnosticImage db:description db:ULTRASONOGRAPHY .
    ?DiagnosticImageStudy db:annotation db:GALLSTONES)) } .
?EncounterPR db:of ?Patient .
?EncounterPR db:procedure ?Procedure .
?Procedure db:description db:COLECTOMY }
OPTIONAL { ?Patient db:demographics ?dem2 .
    ?dem2 db:gender ?gender2 }
OPTIONAL { ?Patient db:id ?id }
OPTIONAL { ?Patient db:of ?Encounter2 .
    ?Encounter2 db:encounterDate ?date }
OPTIONAL { ?Patient db:of ?Encounter3 .
    ?Encounter3 db:attendedBy ?Physician .
    ?Physician db:specialty ?specialty }
OPTIONAL { ?Patient db:of ?Encounter4 .
    ?Encounter4 db:procedure ?Procedure2 .
    ?Procedure2 db:description ?prDescription2 }
OPTIONAL { ?Patient db:of ?Encounter5 .
    ?Encounter5 db:diagnosticImage ?DiagnosticImageStudy2 .
    ?DiagnosticImageStudy2 db:image ?DiagnosticImage2 .
    ?DiagnosticImage2 db:description ?imDescription2 } }
OPTIONAL { ?Patient db:of ?Encounter6 .
    ?Encounter6 db:diagnosticImage ?DiagnosticImageStudy3 .
    ?DiagnosticImageStudy3 db:annotation ?annotation2 }

```

La Figura 5.15 muestra la representación gráfica de esta consulta en Gruff.

5.2.4 *Put Two Objects Closer*

El operador *Put Two Objects Closer* reemplaza un camino que conecta los nodos de dos clases por una relación que se crea entre ellos, acercándolos en el diagrama del grafo (i.e. conectándolos con un camino mas corto). Este operador es útil, por ejemplo, para establecer una relación más directa entre el paciente y las enfermedades que se le han diagnosticado. De esta manera el usuario puede ver más fácilmente los diagnósticos dados a los pacientes, pues reemplaza un camino compuesto por varios arcos.

Put Two Objects Closer toma como entrada un grafo esquema S , un grafo I que es instancia de S y los nodos y arcos marcados en el diagrama del esquema. Estos últimos forman un camino simple, con un nodo inicial m_i , un nodo final m_f y un conjunto E_m de arcos del camino.

Put Two Objects Closer se define por la composición de dos operadores de nivel lógico: *Path Contraction* y *Selective Union* (la definición formal de estos operadores

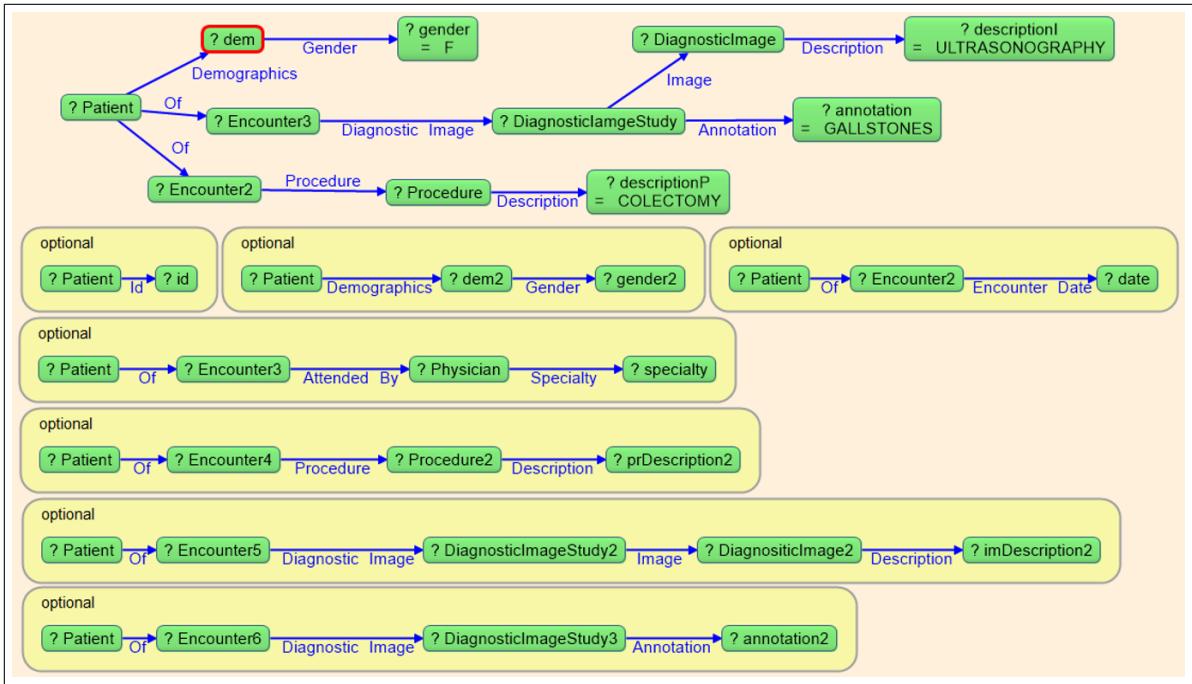


Figura 5.15: Ejemplo de *Filter+Additional Data* expresado en SPARQL.

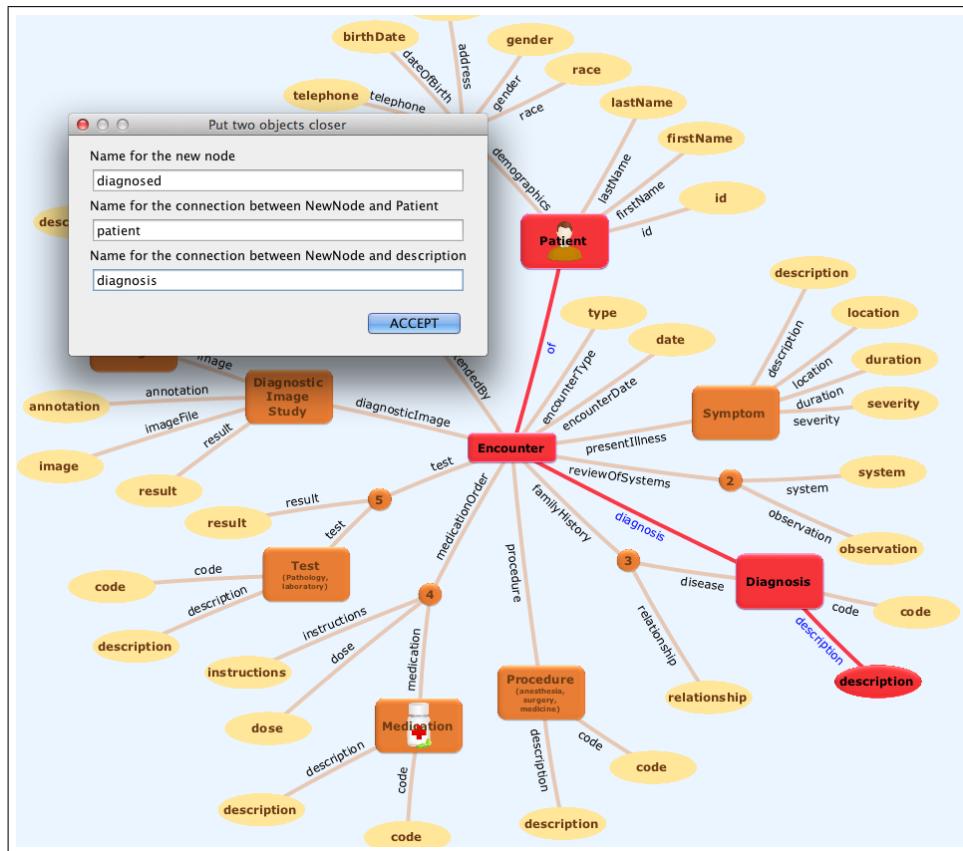
se presenta en la Sección 5.3). El primero, *Path Contraction*, reemplaza el camino y crea la nueva relación entre los extremos del mismo. Mientras que el segundo, *Selective Union*, agrega los datos conectados a los nodos extremos por algún camino simple del esquema. De esta manera, el esquema resultado es un grafo conexo.

Path Contraction se expresa como:

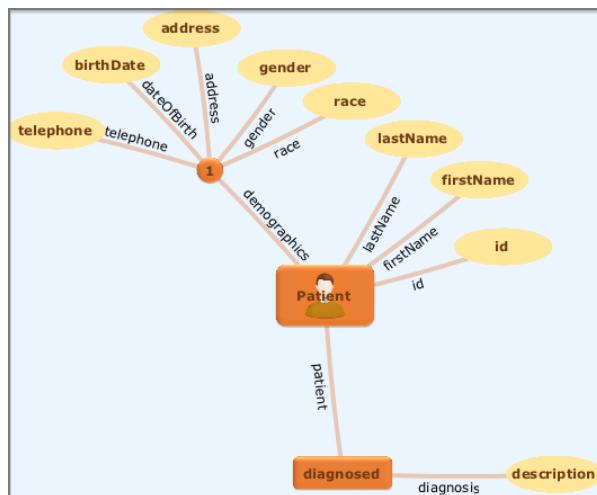
$$PathCt(\langle S, I \rangle, \{m_i, e_1, m_1, \dots, e_k, m_f\}, c, \alpha_i, \alpha_f) \rightarrow \langle S', I' \rangle$$

donde, $\{m_i, e_1, m_1, \dots, e_k, m_f\}$ es el camino formado por los arcos de E_m , c es un nombre de clase para asignar a la nueva relación m_c , α_i y α_f son los nombres de los atributos para los arcos que conectarán m_c con m_i y m_f , respectivamente.

Además, se requiere dos operaciones *Selective Union*, una para agregar los datos del nodo inicial y otra para los del nodo final. Para generar estas expresiones, se requiere encontrar, en el grafo esquema, el conjunto $P_i = \{p_{i1}, p_{i2}, \dots, p_{ik}\}$ de caminos simples no dirigidos tal que todo $p_{ij} \in P_i$ ($1 \leq j \leq k$) inicia en m_i y terminan en un nodo m , $m \in N_S$ y $m \notin Nodes(Em)$. De la misma forma, $P_f = \{p_{f1}, p_{f2}, \dots, p_{fl}\}$ es el conjunto de caminos simples no dirigidos, tal que todo $p_{fq} \in P_f$ ($1 \leq q \leq l$) inicia en m_f y terminan en un nodo m , $m \in N_S$ y $m \notin Nodes(Em)$. Las dos expresiones de *Selective Union* son:



(a)



(b)

Figura 5.16: Operador *Put Two Objects Closer*.

$$\begin{aligned} SelUnion(\langle S, I \rangle, \langle S', I' \rangle, m_i, \{p_{i1}, p_{i2}, \dots, p_{ik}\}) &\rightarrow \langle S', I' \rangle \\ SelUnion(\langle S, I \rangle, \langle S', I' \rangle, m_f, \{p_{f1}, p_{f2}, \dots, p_{fl}\}) &\rightarrow \langle S', I' \rangle \end{aligned}$$

Por ejemplo, en la Figura 5.16a, se quiere reemplazar el camino que conecta el paciente con la descripción de las enfermedades que se le han diagnosticado. El esquema resultado se muestra en la Figura 5.16b. Las expresiones de nivel lógico que corresponden a esta consulta son (en este caso solamente se requiere un *Selective Union* porque *description* no tiene otros datos):

```
PathCt(⟨S, I⟩,
  {Patient, of, Encounter, diagnosis, Diagnosis, description, description}
  “diagnosed”, “patient”, “diagnosis”) → ⟨S1, I1SelUnion(⟨S, I⟩, ⟨S1, I1⟩, Patient,
  {{Patient, id, id}, {Patient, firstName, firstName},
   {Patient, lastName, lastName}, {Patient, demographics, β1},
   {Patient, demographics, β1, race, race},
   {Patient, demographics, β1, gender, gender},
   {Patient, demographics, β1, address, address},
   {Patient, demographics, β1, dateOfBirth, birthDate},
   {Patient, demographics, β1, telephone, telephone}}}
```

Put Two Objects Closer se puede expresar en SPARQL usando las cláusulas *construct* y *optional*. El *construct* permite generar un grafo donde se crea la nueva relación que une los nodos extremos del camino. Esta nueva relación se representa con nodos blancos (*blank nodes*). El *optional*, al igual que *Selective Union*, permite agregar los datos de los nodos extremos. La siguiente es la consulta en SPARQL correspondiente al ejemplo anterior:

```
CONSTRUCT
  { _:b db:patient ?Patient . _:b: db:diagnosis ?description .
    ?Patient db:id ?id . ?Patient db:firstName ?firstName .
    ?Patient db:lastName ?lastName .
    ?Patient db:demographics ?demographics .
    ?Patient db:demographics ?dem1 . ?dem1 db:race ?race .
    ?Patient db:demographics ?dem2 . ?dem2 db:gender ?gender .
    ?Patient db:demographics ?dem3 . ?dem3 db:address ?address .
    ?Patient db:demographics ?dem4 . ?dem4 db:dateOfBirth ?birthDate .
    ?Patient db:demographics ?dem5 . ?dem5 db:telephone ?telephone }
```

WHERE

```
{ ?Encounter db:of ?Patient . ?Encounter db:diagnosis ?Diagnosis .
  ?Diagnosis db:description ?description
  OPTIONAL { ?Patient db:id ?id }
```

```

OPTIONAL { ?Patient db:firstName ?firstName }
OPTIONAL { ?Patient db:lastName ?lastName }
OPTIONAL { ?Patient db:demographics ?demographics }
OPTIONAL { ?Patient db:demographics ?dem1 .
    ?dem1 db:race ?race }
OPTIONAL { ?Patient db:demographics ?dem2 .
    ?dem2 db:gender ?gender }
OPTIONAL { ?Patient db:demographics ?dem3 .
    ?dem3 db:address ?address }
OPTIONAL { ?Patient db:demographics ?dem4 .
    ?dem4 db:dateOfBirth ?birthDate }
OPTIONAL { ?Patient db:demographics ?dem5 .
    ?dem5 db:telephone ?telephone } }
```

5.3 Definición de los operadores de nivel lógico

Los operadores de nivel lógico se definen en esta sección en términos de las modificaciones que se aplican en los grafos esquema e instancia para obtener los grafos de resultado. Para ello se usan las definiciones de grafo esquema e instancia de GDM, con una particularidad: en GraphTQL el esquema es exigido, por tanto todo grafo instancia I debe ser instancia de un grafo esquema S .

5.3.1 *Subgraph Extraction*

El operador *Subgraph Extraction* se define de la misma manera que el *Select a Portion*, es decir, permite seleccionar los datos cubiertos por un conjunto de arcos del esquema. Este conjunto no representa un patrón de búsqueda, por tanto este operador permite recuperar datos incompletos.

El operador *Subgraph Extraction* toma como entradas un grafo esquema S , un grafo I que es instancia de S y un conjunto $\{e_{S1}, e_{S2}, \dots, e_{Sk}\}$ de arcos del esquema ($e_{Si} \in E_S$, para $1 \leq i \leq k$), y retorna los subgrafos S' e I' .

$$Extract(\langle S, I \rangle, \{e_{S1}, e_{S2}, \dots, e_{Sk}\}) \rightarrow \langle S', I' \rangle$$

S' incluye el conjunto de arcos de entrada, $S' = \langle N_{S'}, E_{S'}, \lambda_{S'}, \sigma_{S'} \rangle$ donde,

$$\begin{aligned} E_{S'} &= \{e_{S1}, e_{S2}, \dots, e_{Sk}\}, \\ N_{S'} &= \bigcup_{e_S \in E_{S'}} \text{Nodes}(e_S), \\ \text{y para cada } m &\in N_{S'} \text{ y } e_S \in E_{S'}, \end{aligned}$$

$$\begin{aligned} \lambda_{S'}(m) &= \lambda_S(m), \\ \lambda_{S'}(e_S) &= \lambda_S(e_S) \text{ y} \\ \sigma_{S'}(m) &= \sigma_S(m). \end{aligned}$$

Por lo tanto, los nodos mantienen sus etiquetas y categoría y los arcos, sus etiquetas.

El grafo instancia I' es el subgrafo de I cubierto por S' (la cobertura se define en la Sección 5.1).

Es de notar que los ciclos en el subgrafo de entrada no afectan la semántica del operador. Dado que *Subgraph Extraction* retorna la unión de los arcos seleccionados, si algunos arcos del esquema forman un ciclo, este ciclo estará también en los grafos de la respuesta.

5.3.2 Logic Level Filter

Como se describe en la sección anterior, *Logic Level Filter* define la forma como se procesa el patrón de filtro de los filtros de GraphTQL (i.e. *Filter* y *Filter+Additional Data*).

Este operador permite seleccionar los nodos de una clase de interés que están conectados, por un conjunto dado de caminos no dirigidos, con otros nodos que cumplen una condición específica, los nodos con condición.

En términos generales, el *Logic Level Filter* se expresa como:

$$\text{LogicLFilter}(\langle S, I \rangle, m_I, \exp) \rightarrow \langle S', I' \rangle$$

donde, $\langle S, I \rangle$ son grafos instancia y esquema, m_I es la clase de interés, y \exp , la expresión de filtro, es una expresión lógica formada por la conjunción o disyunción de expresiones de filtro básicas,

$$\exp = bexp_1 \ lop_1 \ bexp_2 \dots \ lop_{k-1} \ bexp_k$$

donde lop_i ($1 \leq i \leq k - 1$) es el operador *AND* o el operador *OR*.

Cada expresión de filtro básica $bexp_j$ ($1 \leq j \leq k$) tiene la forma $m_{j\{p_j\}} \ op_j \ o_j$ donde $m_j \in N_S$ es una clase con condición, p_j es un camino no dirigido entre la clase de interés m_I y la clase con condición m_j , y op_j y o_j son una de las parejas que se muestran en la Tabla 5.1.

El operador *Logic Level Filter* retorna dos grafos S' e I' , esquema e instancia, que incluyen los nodos y arcos seleccionados.

Formalmente, $S' = \langle N_{S'}, E_{S'}, \lambda_{S'}, \sigma_{S'} \rangle$, donde:

$$N_{S'} = \bigcup_{j=1}^k \text{Nodes}(p_j),$$

$$E_{S'} = \bigcup_{j=1}^k \text{Edges}(p_j),$$

y para cada $m \in N_{S'}$ y $e_S \in E_{S'}$,

$$\lambda_{S'}(m) = \lambda_S(m),$$

$$\lambda_{S'}(e_S) = \lambda_S(e_S) \text{ y}$$

$$\sigma_{S'}(m) = \sigma_S(m)$$

Ahora, sea C_I el conjunto de entidades que pertenecen a la clase de interés,

Tabla 5.1: Operadores y valores de las expresiones de filtro básicas.

Operador (op_j)	Operando (o_j)	Donde
$=, >, <, \leq, \geq, \neq$	v_j	$v_j \in \delta(\sigma(m_j))$ (*)
$inSubgraph$	$\langle S', I' \rangle$	
$inSubgraph$	expresión de <i>LogicLFilter</i>	
$exists$		
in	$[v_{j1}, v_{j2}, \dots, v_{jl}]$	$v_{jo} \in \delta(\sigma(m_j))$ ($1 \leq o \leq l$)
$between$	$[v_{j1}, v_{j2}]$	$v_{j1}, v_{j2} \in \delta(\sigma(m_j))$

(*) (i.e. v_j is a value in the domain of m_j)

Tabla 5.2: Evaluación de las condiciones de filtro.

Operador (op_j)	o_j	$eval(n_t, op_j, o_j)$
$=, >, <, \leq, \geq, \neq$	v_i	$\rho(n_t) op_j v_j$
$inSubgraph$	$\langle S', I' \rangle$	$n_t \in N_{I'}$
$inSubgraph$	expresión de <i>LogicLFilter</i>	$n_t \in N_{I'}$, donde $N_{I'}$ es el resultado de ejecutar el <i>LogicLFilter</i>
$exists$		verdadero (el camino existe)
in	$[v_{j1}, v_{j2}, \dots, v_{jl}]$	$\rho(n_t) = v_{jo}$ para algún j ($1 \leq o \leq l$)
$between$	$[v_{j1}, v_{j2}]$	$v_{j1} \leq \rho(n_t) \leq v_{j2}$

$$C_I = \{n | n \in N_I \wedge \exists \xi(m_I, n)\},$$

sea P_{nj} el conjunto de caminos que satisface la expresión de filtro básica $bexp_j$ e inician con el nodo n ($n \in C_I$),

$$P_{nj} = \{p | p \text{ corresponde con } p_j,$$

n es el nodo inicial de $p \wedge n \in C_I, n_t$ es el nodo final de p

$\wedge eval(n_t, op_j, o_j) = \text{verdadero}\}, 1 \leq j \leq k$,

$eval(n_t, op_j, o_j)$ es una función que evalúa la condición según el operador, de acuerdo a como se especifica en la tabla 5.2.

Decimos que el nodo n satisface la expresión básica $bexp_j$ si $P_{nj} \neq \emptyset$.

Sea $O = \{n | n \in C_I \wedge n \text{ satisface } exp\}$ donde exp se evalúa para cada nodo $n \in C_I$ operando el resultado de evaluar las expresiones básicas de filtro.

Entonces, $I' = \langle N_{I'}, E_{I'}, \lambda_{I'}, \sigma_{I'}, \rho_{I'} \rangle$ donde, dado $P_I = \{p | p \in P_{nj} \wedge n \in O \text{ para } 0 \leq j \leq k\}$,

$$N_{I'} = \bigcup_{p \in P_I} Nodes(p),$$

$$E_{I'} = \bigcup_{p \in P_I} Edges(p),$$

y para cada $n \in N_{I'}$ y $e_I \in E_{I'}$,

$$\lambda_{I'}(n) = \{c \in \lambda_I(n) | c = \lambda_{S'}(m) \text{ y } m \in N_{S'}\},$$

$$\lambda_{I'}(e_I) = \lambda_I(e_I),$$

$$\sigma_{I'}(n) = \sigma_I(n) \text{ y}$$

$$\rho_{I'}(n) = \rho_I(n)$$

Logic Level Filter asume que cada expresión de filtro básica se evalúa independientemente de las otras, esto se asegura mediante el diálogo clarificación de filtros de GraphTQL.

5.3.3 Selective Union

Selective Union es un operador binario que adiciona a los grafos $\langle S', I' \rangle$ los datos de $\langle S, I \rangle$ de las entidades de una clase de interés. Estas entidades son comunes a los grafos I' e I y los datos que se seleccionan corresponden con caminos simples no dirigidos dados.

Selective Union toma como entradas los grafos, $\langle S, I \rangle$ y $\langle S', I' \rangle$, una clase de interés m_I y un conjunto P_S de caminos simples no dirigidos del esquema S ; busca las entidades de m_I que son comunes en I e I' , y adiciona a I' los datos de I conectados con esas entidades por un camino que corresponde con un camino de P_S . Los caminos de P_S inician en la clase de interés.

El operador *Selective Union* gestiona el manejo del componente opcional de los filtros de GraphTQL, y también soporta la operación *Put Two Objects Closer*.

En términos generales,

$$SelUnion(\langle S, I \rangle, \langle S', I' \rangle, m_I, P_S) \rightarrow \langle S', I' \rangle$$

donde $P_S = \{p_{S1}, p_{S2}, \dots, p_{Sk}\}$ es un conjunto de caminos simples no dirigidos de S tal que para cada p_{Si} ($1 \leq i \leq k$), m_I es el nodo inicial.

Formalmente, $S' = \langle N_{S'}, E_{S'}, \lambda_{S'}, \sigma_{S'} \rangle$ donde S' resultado se compone de,

$$N_{S'} = (\bigcup_{p \in P_S} Nodes(p)) \bigcup N_{S'},$$

$$E_{S'} = (\bigcup_{p \in P_S} Edges(p)) \bigcup E_{S'},$$

y para cada $m \in N_{S'}$ y $e_S \in E_{S'}$,

Si $m \in S$, $\lambda_{S'}(m) = \lambda_S(m)$,

Si $e_S \in S$, $\lambda_{S'}(e_S) = \lambda_S(e_S)$ y

Si $m \in S$, $\sigma_{S'}(m) = \sigma_S(m)$

$I' = \langle N_{I'}, E_{I'}, \lambda_{I'}, \sigma_{I'}, \rho_{I'} \rangle$ donde, dados

$$O = \{n | n \in N_I \wedge n \in N_{I'} \wedge \xi(m_I, n)\} \text{ y}$$

$P_I = \{p | p \text{ es un camino en } I' \text{ que corresponde con un camino en } P_S \text{ y } n_i \in O \text{ es el nodo inicial de } p\}$,

$$N_{I'} = (\bigcup_{p \in P_I} \text{Nodes}(p)) \cup N_{I'},$$

$$E_{I'} = (\bigcup_{p \in P_I} \text{Edges}(p)) \cup E_{I'},$$

y para cada $n \in N_{I'}$ y $e_I \in E_I$,

$$\lambda_{I'}(n) = \{c \in \lambda_I(n) | c = \lambda_{S'}(m) \text{ y } m \in N_{S'}\} \cup \lambda_{I'}(n),$$

$$\text{Si } e_I \in I, \lambda_{I'}(e_I) = \lambda_I,$$

$$\text{Si } n \in I, \sigma_{I'}(n) = \sigma_I(n) \text{ y}$$

$$\text{Si } n \in I, \rho_{I'}(n) = \rho_I(n)$$

5.3.4 Path Contraction

Path Contraction transforma los grafos esquema e instancia en grafos donde se hacen explícitas relaciones que eran implícitas en los grafos originales. Este operador reemplaza los nodos y arcos internos de un camino simple no dirigido del grafo esquema por un nuevo nodo que se conecta con los nodos inicial (m_i) y final (m_f) del camino, creando una relación entre ellos. El camino también puede ser un ciclo simple no dirigido, donde los nodos inicial y final son el mismo.

En términos generales, *Path Contraction* se expresa como:

$$\text{PathCt}(\langle S, I \rangle, \{m_i, e_1, m_1, \dots, e_k, m_f\}, c, \alpha_i, \alpha_f) \rightarrow \langle S', I' \rangle$$

donde, $\{m_i, e_1, m_1, \dots, e_k, m_f\}$ es un camino (o ciclo) simple no dirigido del grafo esquema S ; c es un nombre de clase para asignar al nuevo nodo m_c ($\lambda(m_c) = c$), este es un nodo de valor compuesto; α_i y α_f son los nombres de los atributos para los arcos que conectarán m_c con m_i y m_f , respectivamente. Los nodos inicial y final del camino, m_i y m_f , son nodos de clase objeto o de valor básico.

Formalmente, $S' = \langle N_{S'}, E_{S'}, \lambda_{S'}, \sigma_{S'} \rangle$ donde,

$N_{S'} = \{m_i, m_f, m_c\}$ donde m_c es un nuevo nodo de clase valor compuesto, tal que $\sigma(m_c) = \text{com}$ y $\lambda(m_c) = c$.

$E_{S'} = \{(m_c, \alpha_i, m_i), (m_c, \alpha_f, m_f)\}$ donde (m_c, α_i, m_i) y (m_c, α_f, m_f) son arcos nuevos.

Adicionalmente,

$$\lambda_{S'}(m_i) = \lambda_S(m_i), \lambda_{S'}(m_f) = \lambda_S(m_f),$$

$$\lambda_{S'}(m_c, \alpha_i, m_i) = \alpha_i, \lambda_{S'}(m_c, \alpha_f, m_f) = \alpha_f,$$

$$\sigma_{S'}(m_i) = \sigma_S(m_i), \sigma_{S'}(m_f) = \sigma_S(m_f)$$

El grafo instancia, $I' = \langle N_{I'}, E_{I'}, \lambda_{I'}, \sigma_{I'}, \rho_{I'} \rangle$ donde, dado $P_I = \{p | p \text{ es un camino en } I \text{ que corresponde con } \{m_i, e_1, m_1, \dots, e_k, m_f\}\}$,

$$E_{I'} = \{(n_c, \alpha_i, n_i) \cup (n_c, \alpha_f, n_f) | \text{ para algún } p \in P_I, n_i \text{ es el nodo inicial de } p,$$

n_f es el nodo final de p , $\xi(m_i, n_i), \xi(m_f, n_f), n_c$ es un nuevo nodo tal que $\sigma_{I'}(n_c) = com, \lambda_{I'}(n_c) = c, \xi(m_c, n_c)\}$

$$N_{I'} = \bigcup_{e_{I'} \in E_{I'}} \text{Nodes}(e_{I'})$$

y para cada $n \in N_{I'}$ tal que $n \in N_I$ y para cada $e_{I'} = (o, \alpha, q) \in E_{I'}$,

$$\lambda_{I'}(n) = \{d \in \lambda_I(n) | \exists m, m \in N_{S'}, d = \lambda_{S'}(m) \text{ y } \xi(m, n)\},$$

$$\lambda_{I'}(e_{I'}) = \alpha,$$

$$\sigma_{I'}(n) = \sigma_I(n) \text{ y}$$

$$\rho_{I'}(n) = \rho_I(n)$$

5.3.5 Union (Unión de grafos)

Union es un operador binario que recibe los grafos esquema e instancia $\langle S, I \rangle$, $\langle S', I' \rangle$, y entrega como resultado la unión de los conjuntos de nodos y arcos de los grafos esquema ($S'' = S \bigcup S'$) y de los grafos instancia ($I'' = I \bigcup I'$).

En términos generales,

$$\text{Union}(\langle S, I \rangle, \langle S', I' \rangle) \rightarrow \langle S'', I'' \rangle$$

Formalmente, $S'' = \langle N_{S''}, E_{S''}, \lambda_{S''}, \sigma_{S''} \rangle$ donde,

$$N_{S''} = N_S \bigcup N_{S'},$$

$$E_{S''} = E_S \bigcup E_{S'},$$

y para cada $m \in N_{S''}$ y $e_S \in E_{S''}$,

$$\text{Si } m \in S, \lambda_{S''}(m) = \lambda_S(m) \text{ y si } m \in S', \lambda_{S''}(m) = \lambda_{S'}(m),$$

$$\text{Si } e_S \in S, \lambda_{S''}(e_S) = \lambda_S(e_S) \text{ y si } e_S \in S', \lambda_{S''}(e_S) = \lambda_{S'}(e_S) \text{ y}$$

$$\text{Si } m \in S, \sigma_{S''}(m) = \sigma_S(m) \text{ y si } m \in S', \lambda_{S''}(m) = \sigma_{S'}(m)$$

$I'' = \langle N_{I''}, E_{I''}, \lambda_{I''}, \sigma_{I''}, \rho_{I''} \rangle$ donde,

$$N_{I''} = N_I \bigcup N_{I'},$$

$$E_{I''} = E_I \bigcup E_{I'},$$

y para cada $n \in N_{I''}$ and $e_I \in E_{I''}$,

$$\lambda_{I''}(n) = \lambda_I(n) \bigcup \lambda_{I'}(n),$$

$$\text{Si } e_I \in I, \lambda_{I''}(e_I) = \lambda_I, \text{ y si } e_I \in I', \lambda_{I''}(e_I) = \lambda_{I'}(e_I),$$

$$\text{Si } n \in I, \sigma_{I''}(n) = \sigma_I(n), \text{ y si } n \in I', \sigma_{I''}(n) = \sigma_{I'}(n) \text{ y}$$

$$\text{Si } n \in I, \rho_{I''}(n) = \rho_I(n), \text{ y si } n \in I', \rho_{I''}(n) = \rho_{I'}(n)$$

5.4 Discusión

La definición de los diálogos de clarificación de filtros y de los operadores evolucionó a lo largo del proyecto. A continuación se describen algunos aspectos a destacar.

En la Sección 4.6 se describió cómo la definición de los operadores de filtro fue cambiando hasta requerir que el usuario marque una clase de interés sobre la que se aplican las condiciones de filtro. Esta clase es un elemento principal del proceso de generación de los diálogos de clarificación de filtros. Una vez identificada la necesidad de solicitar al usuario elegir una clase de interés, para la clarificación de filtros se consideraron varias opciones. La primera, generar las preguntas del diálogo solamente para la clase de interés. En esta opción no se buscan nodos de bifurcación y las preguntas consideran los caminos que unen la clase de interés con cada clase con condición. De esta manera, se tiene la conjunción o la disyunción de las condiciones, es decir se recuperan entidades de la clase de interés que cumplen todas las condiciones (i.e. están conectadas con nodos que cumplen cada una de las condiciones especificadas) o entidades que cumplen al menos una de las condiciones. La expresividad de estos filtros es limitada porque no es posible combinar conjunción y disyunción de las condiciones. Para permitir algunas combinaciones de conjunción y disyunción de las condiciones, se optó por tomar ventaja de la representación gráfica del patrón de filtro. Los nodos de bifurcación de los caminos son fácilmente distinguibles en el diagrama del grafo y la representación gráfica ayuda a dar sentido a las preguntas de clarificación. De esta manera, se encuentran subpatrones que permiten definir composición de los filtros y ofrecer mayor expresividad en el patrón de filtro.

Sin embargo, la definición actual de los diálogos de clarificación de filtros tienen al menos dos limitantes: la primera, en el mismo nodo de bifurcación no es posible mezclar la conjunción y disyunción de las condiciones que se derivan del nodo (i.e. las conectadas a él). Y la segunda, no se incluyen casos en los cuales el patrón de filtro tiene ciclos y hay más de un nodo con condición conectado a esos ciclos.

Una opción para manejar el primer caso, la mezcla de operadores lógicos en las condiciones de un nodo, es agregar a GraphTQL niveles de experticia del usuario y ofrecer un nivel en el que el usuario pueda manipular las expresiones de las condiciones, expresadas en texto, para crear la expresión lógica que más le convenga. En este caso, cada nodo de bifurcación generaría en un subfiltro, ya que el operador *Logic Level Filter* permite evaluar cualquier expresión lógica formada por expresiones de filtro básicas.

El segundo caso es complejo, porque los ciclos producen combinaciones de caminos que hacen que las preguntas del diálogo puedan tener poco sentido para el usuario y porque esas combinaciones pueden generar muchas preguntas o hacer que la pregunta de un nodo incluya muchas condiciones (o muchos caminos hacia la misma condición). Por ejemplo, en el patrón de filtro marcado en la Figura 5.17, con el método de generación de diálogos actual hay tres nodos de bifurcación: *Encounter*, β_3 y *Diagnosis*. Las condiciones para β_3 y *Diagnosis* son las mismas (i.e. *relationship = FATHER AND (OR) description = DIABETES*). Buscar en la historia familiar los registros que incluyen “DIABETES” y “FATHER” puede tener mas sentido que buscar los diagnósticos de “DIABETES” conectados a la relación “FATHER”. Por lo tanto, el diálogo para *Diagnosis* podría carecer de sentido para el usuario. La situación es mas

compleja cuando hay varios ciclos, que generan muchas combinaciones de caminos y, por tanto, de condiciones.

Una primera aproximación para manejar estos casos sería cambiar los diálogos de clarificación de caminos. Primero, desplegando el diálogo en el momento que se elige una operación de filtro y no, como actualmente se realiza, cada vez que se agrega un nodo. Y segundo, teniendo en cuenta la clase de interés y los nodos con condición, de tal manera que el usuario identifique cuáles caminos tienen sentido para su consulta. Por ejemplo, para el patrón de la Figura 5.17, hay cuatro caminos que unen la clase de interés y los nodos con condición:

$$\begin{aligned} p_1 &= \{Encounter, diagnosis, Diagnosis, description, description\}, \\ p_2 &= \{Encounter, familyHistory, \beta_3, disease, Diagnosis, description, \\ &\quad description\}, \\ p_3 &= \{Encounter, diagnosis, Diagnosis, disease, relationship, relationship\}, \\ p_4 &= \{Encounter, familyHistory, \beta_3, relationship, relationship\}, \end{aligned}$$

Para este ejemplo, el usuario decide que p_3 no debe ser incluido en la consulta, con lo cual *Diagnosis* deja de ser nodo de bifurcación, y por lo tanto no genera pregunta de clarificación.

De otra parte, los cambios en la definición de los operadores de filtro de GraphTQL (descritos en la Sección 4.6) también ocasionaron cambios en la definición de los operadores de nivel lógico. Inicialmente, se tenía una relación uno-a-uno entre los dos conjuntos de operadores. Como se ha mostrado en este Capítulo, la definición actual tiene una semántica diferente. En particular, el operador *Selective Union* es útil para definir *Filter*, *Filter+Additional Data* y *Put Two Objects Closer*.

Adicionalmente, los operadores de filtro actuales, no permiten expresar consultas que comparan los valores de nodos de valor básico. Por ejemplo, encontrar médicos

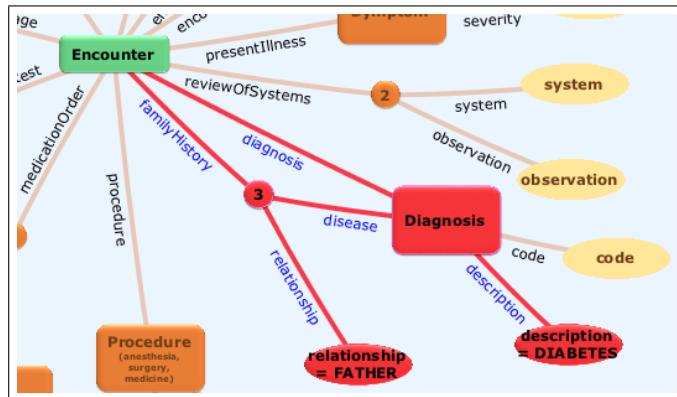


Figura 5.17: Patrón de filtro con ciclo y varias condiciones.

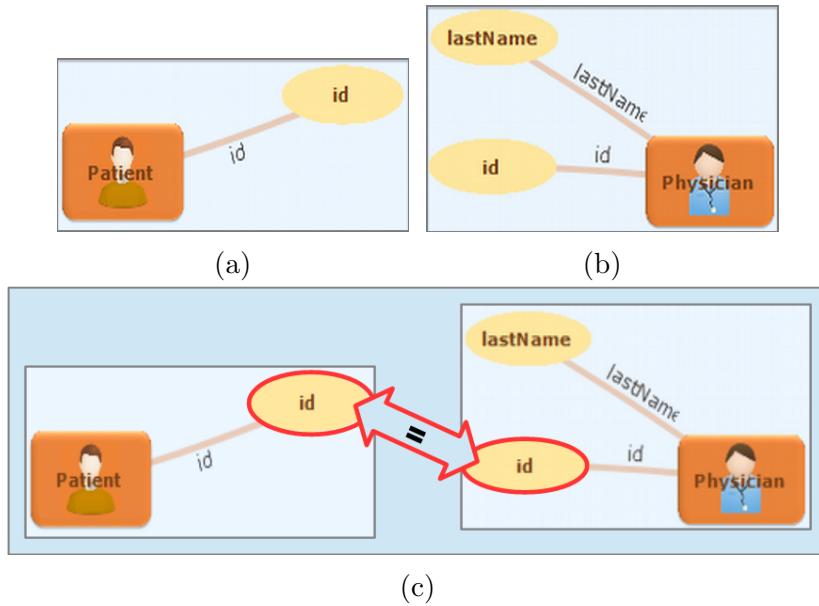


Figura 5.18: Condiciones entre tipos de nodo.

que a su vez están registrados como pacientes. Para ello se requiere especificar como condición que el *id* del médico sea igual al *id* del paciente y además establecer la relación entre dos subgrafos no conexos, pues un camino en el esquema entre paciente y médico implicaría que la consulta recupera los médicos que son paciente de sí mismos. Una posible solución para manejar estas consultas es ofrecer operaciones binarias entre los grafos que se obtienen como respuesta en pasos anteriores de la formulación de la consulta. Así, para el ejemplo, los grafos de las Figuras 5.18a y 5.18b resultan de operaciones *Select a Portion*, y la Figura 5.18c muestra lo que podría ser, graficamente, la formulación de la consulta del ejemplo.

Capítulo 6

Enfoque en el Diseño Centrado en el Usuario

Durante el desarrollo de GraphTQL se aplicaron algunas técnicas de Diseño Centrado en el Usuario [60, 165], enfocadas a satisfacer las necesidades de los usuarios en el dominio médico, en particular, las relacionadas con el acceso a los datos de las historias clínicas.

Era de interés de este estudio brindar una herramienta de consulta para usuarios finales en dominios de aplicación donde los usuarios son expertos en el dominio (i.e. profesionales de ese dominio), quienes requieren realizar consultas *ad-hoc* (no conocidas con anticipación), de mediana complejidad. Estas consultas son en muchos casos el punto de partida, en cuanto a la selección de los datos, para realizar el análisis de porciones de información con herramientas que permiten realizar otros tipos de análisis, por ejemplo, análisis estadístico. El interés de este estudio también se centró en dominios donde los datos pueden ser incompletos y se puede requerir incluir datos no estructurados.

Entre otros dominios, el de aplicación de este trabajo, la información clínica de pacientes, cumple con estas características. Los médicos, enfermeros y profesionales de la salud requieren el acceso a la información de los pacientes con múltiples fines: atención, investigación, enseñanza, administración clínica, creación y realización de planes de prevención, alertas epidemiológicas, entre otros.

De otra parte, el grupo de investigación GEDI de la Universidad del Valle ha realizado varios proyectos enfocados al dominio médico, con lo cual espera aportar mejores herramientas a la práctica médica de la región. Siguiendo esta trayectoria, se decidió realizar este estudio tomando como caso de aplicación dicho dominio.

La aplicación de técnicas de diseño centrado en el usuario durante el desarrollo de GraphTQL se realizó en cuatro momentos, que se muestran en la Tabla 6.1. Como se puede observar en la tabla, durante el desarrollo del proyecto se privilegió la aplicación de pruebas de usabilidad sobre los métodos de indagación e inspección. Sin embargo,

de estos últimos se realizó una entrevista, la entrevista preliminar, y en la primera prueba de evaluación se invitó a participar a un experto en diseño de interfaces que realizó una evaluación de expertos. En la literatura, los expertos [135, 165] dan especial importancia a las pruebas de usabilidad y recomiendan realizarlas siempre y, en lo posible, combinarlas con otros métodos.

Tabla 6.1: Pruebas de usabilidad en el desarrollo del proyecto.

Prueba	Objetivo	Momento de aplicación
Entrevista preliminar	Conocer en cuáles situaciones es útil una herramienta para formular consultas sobre datos clínicos y cuáles consultas se requieren	Antes de iniciar el diseño del lenguaje
Prueba exploratoria	Validar si la representación de los datos en grafos y las operaciones de transformación de grafos son entendibles para el usuario	Punto intermedio en el diseño, antes de iniciar la implementación
Pruebas de evaluación	Primera Prueba de Evaluación: organización, iconos y colores de dos prototipos de la interfaz	Punto intermedio en la implementación del prototipo del lenguaje
	Segunda Prueba de Evaluación: facilidad de uso del lenguaje visual de consulta	Terminada la primera etapa de implementación del prototipo del lenguaje
Prueba comparativa	Evaluar la facilidad de uso del lenguaje visual de consulta y compararlo con una interfaz gráfica para formulación de consultas en SPARQL	Terminada la primera etapa del prototipo del lenguaje

En este proyecto las pruebas de usabilidad tienen especial relevancia porque las tareas que los usuarios realizan son complejas. Por ello, el principal objetivo de las pruebas fue corroborar si el lenguaje propuesto facilita la especificación de las consultas. Se trataba de observar la experiencia del usuario con la herramienta y además, recopilar información sobre los tipos de consulta que los usuarios requieren y los escenarios donde se requieren. La facilidad de uso sería, posiblemente, difícil de evaluar por parte de un experto en usabilidad, quien tiene generalmente una formación técnica que lo ubica en un nivel y un contexto diferente al de los usuarios finales. Por tanto el experto podría pasar por alto características del lenguaje que pueden generar dificultades para el usuario o, por el contrario, considerar que para el usuario puede resultar difícil usar ciertas características que en realidad no le representen problemas.

El diseño de las pruebas siguió recomendaciones propuestas por Rubin et al. [165], Nielsen [135], Petrelli [153] y Tullis y Albert [192], descritas en las Secciones 3.1.2 y 3.1.3 de este documento.

6.1 Entrevista preliminar

El objetivo de la primera prueba, la entrevista preliminar, fue conocer cuál información se registra en una historia clínica, cuál necesita consultar un médico cuando está atendiendo un paciente y en cuáles situaciones es útil una herramienta para formular consultas sobre datos clínicos.

Se entrevistaron cuatro médicos especialistas en gastroenterología, pediatría, nutriología, epidemiología, ginecología y otorrinolaringología. Las entrevistas se desarrollaron en formato libre, sin cuestionario guía. Después de explicar el objetivo del proyecto, el moderador guió la conversación para responder a las siguientes preguntas: ¿Cómo se desarrolla una consulta externa?, ¿Cuáles datos se registran en una historia clínica?, ¿Durante la atención a un paciente qué información puede necesitar?, ¿Sería importante relacionar los datos de un paciente y los de las historias clínicas de sus familiares? y ¿En cuáles procesos (ej. investigación o enseñanza) puede ser útil una herramienta de consulta de datos clínicos?. Cada entrevista tuvo una duración aproximada de una hora y se grabaron en audio. En el [Anexo A](#) se resume cada entrevista realizada.

Con el desarrollo de estas entrevistas se obtuvo información sobre: diferencias entre tipos de consulta (ej. de urgencias, externa, de control), ejemplos de datos que orientan la búsqueda de una patología y son importantes para la vigilancia epidemiológica, la importancia de las fechas en que se realizan los registros en la historia clínica, el uso común de la negación para expresar hechos (ej. el diagnóstico “no ictericia”), escenarios en los que es necesario consultar datos de las historias clínicas y ejemplos de consultas.

A partir de la información recolectada se concluyó que las consultas *ad hoc* sobre la historia clínica de los pacientes son necesarias en los siguientes escenarios: evaluación y vigilancia epidemiológica, consulta externa y citas de control, administración de los servicios de salud, investigación de casos clínicos, seguimiento de la evolución de los pacientes, procesos de enseñanza-aprendizaje y planeación de programas de prevención. Dentro de estos escenarios los médicos dieron 31 ejemplos de consultas, que se recopilan en el [Anexo B](#), entre ellas:

- En consulta externa y citas de control: Revisar los conceptos dados por médicos de otra especialidad al paciente que está en consulta.
- En administración de los servicios de salud: Obtener estadísticas de diagnósticos o de resultados de exámenes. Por ejemplo, ¿Cuántos hemogramas se hicieron en el mes?, ¿Cuáles son las patologías más frecuentes?.
- En investigación de casos clínicos y procesos de enseñanza-aprendizaje: Encontrar grupos de pacientes que cumplen un perfil. Por ejemplo, datos de los pacientes de género femenino que estuvieron hospitalizados en la sala de pediatría general durante los años 2009 y 2010 con diagnóstico de enfermedad por reflujo gastroesofágico.
- En planeación de programas de prevención y seguimiento de la evolución de los

pacientes: Revisar las historias clínicas de recién nacidos cuyas madres tuvieron diagnóstico de hipertensión durante el embarazo.

Adicionalmente, las consultas de ejemplo dadas por los médicos se clasificaron en tres tipos: consultas que recuperan datos de la historia clínica de un paciente, consultas para seleccionar grupos de pacientes que cumplen un perfil y consultas que incluyen agrupamiento y operaciones de agregación. El desarrollo actual del lenguaje visual de consulta se centra en los dos primeros tipos, las operaciones de agregación se proponen como trabajo futuro.

6.2 Prueba exploratoria

Con la prueba exploratoria se trató de corroborar si el uso de un grafo como modelo de datos, las operaciones propuestas para el lenguaje, los iconos de las operaciones y sus textos explicativos eran entendibles. Para esta prueba el perfil de usuario fue más amplio, en esta oportunidad participaron tres profesionales de la salud, entre ellos un epidemiólogo y un enfermero.

Con esta prueba se trató de responder a las siguientes preguntas: ¿Los participantes interpretan correctamente los grafos esquema e instancia dados como ejemplo?, ¿Es fácil para los participantes interpretar estos grafos?, ¿Es fácil para los participantes entender las transformaciones que los operadores realizan en los grafos? y ¿Cuál es la opinión de los usuarios sobre la herramienta que se propone, sería útil en su trabajo?.

En la prueba se utilizaron prototipos en papel y se dividió en cuatro partes:

1. Presentación de la noción y uso del grafo como modelo de representación de datos.
2. Presentación del grafo esquema como representación general de la organización de los datos y del grafo instancia como una representación particular de hechos acorde al esquema. Para identificar si los participantes interpretan correcta y fácilmente los grafos, se les pidió que respondieran las siguientes preguntas sobre los grafos esquema e instancia de ejemplo:
 - ¿Qué datos se tienen de cada paciente?
 - ¿Qué datos se tienen del paciente 1 en particular?
3. Presentación de las operaciones del lenguaje. Se presentaron cuatro operaciones: *Select a Portion*, *Put Two Objects Closer*, *Value Filter* y *Class Filter* (los últimos son las definiciones previas a los operadores de filtro actuales). Se presentó a los participantes ejemplos gráficos de los operadores *Put Two Objects Closer* y *Value Filter*. Adicionalmente, cada operación se asoció con un ícono y un breve texto descriptivo.
4. Identificación de operaciones aplicadas en cuatro casos. En cada caso se mostró el grafo esquema inicial y el grafo esquema transformado. Luego se hicieron cuatro preguntas sobre la facilidad para entender el modelo, los íconos y textos de las operaciones, se mostraron dos formas de pintar los filtros sobre el esquema y se

pidió que eligieran la forma de su preferencia.

El moderador de la prueba podía aceptar comentarios y preguntas del usuario en cualquier momento y responder a ellas. Las pruebas se realizaron en sesiones de aproximadamente una hora con cada participante y se grabaron en video. El plan de la prueba se presenta en el [Anexo C](#) y un resumen de las respuestas dadas por los participantes, en el [Anexo D](#).

Se obtuvieron los siguientes resultados:

- Los participantes identificaron y entendieron la información representada en el esquema y la instancia y respondieron acertadamente a las preguntas de la parte 2.
- Dos de los participantes tardaron en identificar la relación *Encounter* entre los nodos *Physician* y *Patient*, por la notación que se usó en los prototipos para los nodos de valor compuesto etiquetados. Sus observaciones se tuvieron en cuenta para cambiar la representación de ese tipo de nodo.
- Los participantes hicieron comentarios como “Me parece interesante” y “Es un diagrama que permite encontrar y amarrar la relación entre los datos”.
- En la actividad de la parte 4, el 91,6% de los casos fueron identificados correctamente, y en un 16,6% el moderador intervino para hacer notar alguna parte del grafo.
- De acuerdo con los participantes las operaciones permiten realizar consultas pertinentes para el ejercicio clínico y la investigación en medicina.
- Todos los participantes coincidieron en que los iconos y los textos propuestos son claros.
- Como comentarios adicionales, los participantes afirmaron que la primera impresión al ver los grafos es que esta representación (“tipo red”) es compleja, pero que cuando empezaron a usarlos y responder las preguntas notaron que era fácil encontrar la información y las relaciones entre los datos.

De la prueba exploratoria se derivaron las siguientes conclusiones:

- Para los participantes fue fácil interpretar los grafos esquema e instancia y las operaciones del lenguaje transformación de grafos.
- Los iconos y textos propuestos para la aplicación fueron claros para los participantes.
- Los participantes opinaron que la herramienta sería útil en su trabajo.
- A los participantes les tomó más tiempo, y aparentemente más esfuerzo, interpretar el grafo instancia. Esto confirma que es una buena alternativa usar el esquema para guiar la formulación de las consultas.

6.3 Pruebas de evaluación

La etapa de evaluación incluyó dos pruebas que se describen a continuación.

La primera prueba de evaluación se realizó en el marco de un trabajo de grado de pregrado [67] desarrollado por estudiantes de la Pontificia Universidad Javeriana - Seccional de Cali. En este trabajo se implementó el primer prototipo del lenguaje. El objetivo de la prueba fue evaluar dos prototipos de la interfaz con personas con perfiles diferentes, incluyendo un experto en diseño de interfaces y algunos usuarios finales. Cada participante evaluó uno de los dos prototipos.

En esta prueba participaron un experto en diseño de interfaces, tres médicos y seis estudiantes (de medicina, odontología e ingeniería de sistemas). El experto es diseñador gráfico, especializado en estética, con experiencia en publicidad, y desarrollo y diseño de estilos y de páginas web. En la prueba se presentaron dos diseños con diferencias en la organización de la pantalla y en los colores usados para la interacción con el grafo. Con base en las opiniones de los participantes se eligió la organización de la pantalla, se cambiaron algunos colores, se rediseñaron los iconos de los botones de operación y de control, y se cambiaron los nombres de los operadores. Las decisiones se tomaron teniendo en cuenta el prototipo mejor calificado, de acuerdo con una evaluación cuantitativa en la que se asignaron ponderaciones diferentes según el perfil del participante, siendo la opinión del experto y de los profesionales en medicina las que mayor peso tuvieron.

El plan de la primera prueba de evaluación es una adaptación del plan de la prueba exploratoria ([Anexo C](#)). El plan y los resultados detallados de la prueba se encuentran en [67]. Como era de esperarse, en esta prueba se observó que el perfil del participante incide en los aspectos de la herramienta que son relevantes para él. Los médicos y estudiantes de carreras del área de salud se concentraron en el uso del lenguaje como herramienta para apoyar la investigación y la toma de decisiones. El experto en usabilidad se centró en el diseño de la interfaz, sin prestar mayor atención al propósito de la herramienta. Por su parte, el interés de los estudiantes de Ingeniería de Sistemas se centró en el uso del grafo como modelo de representación de datos y las operaciones definidas sobre él.

En la segunda prueba de evaluación, cuatro participantes, dos profesionales y dos estudiantes de carreras relacionadas con áreas de la salud, evaluaron el prototipo del lenguaje visual de consulta propuesto. El objetivo de esta prueba era evaluar la facilidad de uso del prototipo de GraphTQL, pidiendo a los participantes formular cuatro consultas. La prueba se enfocó en la formulación de las consultas, no incluyó la presentación de los resultados de las mismas.

Se aplicó el plan de la prueba comparativa, con excepción del cuestionario SUS (*System Usability Scale*) [14]. Este plan consta de:

1. Apertura: El moderador explicó el propósito de la prueba y las partes que incluye.
Los participantes diligenciaron un cuestionario pre-test, que recoge información de

su perfil de formación, y firmaron el formato de consentimiento para realizar la prueba.

2. Demostración de la herramienta: se hizo utilizando una base de datos sobre películas y series de televisión. El moderador explicó las opciones, botones y operaciones de la herramienta, e ilustró la formulación de consultas mediante tres ejemplos.
3. Actividades: se usó una base de datos clínicos para que el usuario realizara cuatro actividades. En cada actividad se pidió al participante que formulara una consulta. La complejidad de las consultas aumentó de una actividad a la siguiente. En las consultas se aplican las operaciones de filtro y diálogos de clarificación de caminos y de condiciones de filtro de GraphTQL. El tiempo máximo que los participantes tuvieron para formular las consultas fue de 15 minutos.

Las consultas propuestas para las actividades fueron las mismas de la prueba comparativa a excepción de un cambio en un valor de una condición de filtro. Estas consultas y las razones por las que se eligieron se muestran en el plan de la prueba comparativa ([Anexo G](#)) y en la Sección [6.4](#). Antes de aplicar la prueba se realizó un *pre-test* en el que participó un ingeniero de sistemas, con el fin de corregir detalles del plan de pruebas.

Tabla 6.2: Segunda prueba de evaluación - Resultados por participante.

Participante	Consulta 1		Consulta 2		Consulta 3		Consulta 4	
	Correc-ta	Erro-res	Correc-ta	Erro-res	Correc-ta	Erro-res	Correc-ta	Erro-res
1	No	1	No	2	No	2	No	1
2	No	1	Si	0	Si	0	No	1
3	No	1	Si	0	No	2	No	2
4	No	2	No	2	No	3	No	2

Tabla 6.3: Segunda prueba de evaluación - Resultados promedio.

Consulta	Correctas	Errores
1	0%	1,25
2	50%	1,2
3	25%	2
4	0%	1,5
Todas	18,8%	1,5

El resumen de la aplicación de esta prueba se presenta en el [Anexo F](#). Las Tablas [6.2](#) y [6.3](#) muestran los resultados por participantes con respecto a si la consulta se formuló correctamente y el número de errores, por tipo de error, en los que incurrió,

si fue el caso. Si bien la tasa de respuestas correctas fue baja (18,8%), se pudieron identificar algunos errores frecuentes que se evitarían al mejorar la interfaz, entre ellos:

- La versión que se evaluó no incluía el diálogo de clarificación de caminos, la herramienta marcaba todos los caminos posibles entre los nodos seleccionados, para hacerlos evidentes al usuario, y se esperaba que éste desmarcara los que no requería. Es importante notar que los usuarios tienden a aceptar por defecto las sugerencias que les hace la herramienta. Este comportamiento se evidenció en las consultas que incluyen los nodos *Encounter* y *Diagnosis*, ya que estos forman un ciclo y tienen dos caminos que los conectan. En el 58,3% de las consultas que incluyen dichos nodos se presentó este error. Hubo un caso en el participante no cometió el error porque la pregunta de clarificación de filtros, que hacia referencia a los caminos, le permitió identificar que debía quitar la marca de ese camino.
- El nombre de los operadores de filtro no parecían ayudarle al usuario a diferenciar la correspondiente operación. En el 25% de las actividades el participante aplicó una operación de filtro que no correspondía con lo solicitado en la consulta. Los participantes afirmaron que entendían la diferencia entre los filtros pero que en la herramienta no identificaban cuál era cada uno de ellos.
- En ese prototipo, el operador *Object With Characteristic* no cambia el esquema y retorna todos los datos que corresponden con el esquema de entrada. Sin embargo, las consultas generalmente requieren incluir en la respuesta solo una porción de las clases del grafo esquema de entrada. Entonces el usuario se ve obligado a aplicar un *Select a Portion* antes del *Object With Characteristic*.
- Cuando los participantes aplicaban el operador *Object With Characteristic* los desconcertaba que el esquema no cambiara y dudaban de que el operador se hubiera aplicado.
- Se identificaron algunas validaciones que se podían agregar en cada operación para prevenir posibles errores. En particular, validaciones de parámetros marcados que no son requeridos para el operador elegido, por ejemplo, si el usuario marca condiciones sobre algunos atributos y luego elige el operador *Select a Portion*, las condiciones marcadas no tienen incidencia en la operación, por tanto es posible que el operador esté confundiendo la operación que requiere aplicar. Esto ocurrió en dos casos en la primera actividad.

Además se observó que en el 31,25% de las actividades los participantes eligieron respuestas del diálogo de clarificación de filtro que no se ajustan a lo que la consulta requiere. Algunas alternativas de mejora se plantean como trabajo futuro de esta tesis.

De otra parte, los participantes coincidieron en que la herramienta sería de gran utilidad en su campo profesional, que es fácil de usar, que la fase de entrenamiento que requiere es corta y que la representación de los datos en el grafo es fácil de entender.

Los prototipos evaluados en las pruebas de evaluación se desarrollaron en Java¹,

¹<https://www.oracle.com/java/>

se usó el *framework* JUNG¹ para la visualización del grafo, el algoritmo Kamada-Kawai [103] (KKLayout) para la disposición inicial de los nodos, el formato de archivos GraphML [29] para describir el grafo esquema y XML² para especificar la consulta que el usuario formula.

El prototipo de la primera prueba de evaluación se desarrolló en el marco del trabajo de grado de pregrado de Guanga y Londoño [67], en éste se usan los comandos que por defecto provee JUNG para la interacción con el grafo, se calcula la disposición del diagrama del grafo esquema con KKLayout cada vez que se ejecuta la aplicación y se implementaron las primeras versiones de los operadores.

La decisión de usar KKLayout para calcular la disposición del diagrama del grafo se basó en los resultados de un estudio y pruebas realizados sobre diversos algoritmos que proveen las herramientas de visualización de grafos [67]. Para ello se tuvo en cuenta, como características principales, las siguientes capacidades del mecanismo de visualización: capacidad para visualizar grafos de tamaño medio (alrededor de 1000 nodos), visualización estética y capacidad para disminuir el cruce de arcos y el solapamiento de nodos.

En el prototipo de la segunda prueba de evaluación se realizaron varios cambios: La interacción por defecto de JUNG se modificó para que los comandos requeridos en GraphTQL se aplicarán usando solamente el clic, izquierdo y derecho, del ratón. Cambiaron los nombres y la semántica de los operadores. Se incluyó el diálogo de clarificación de filtros. En este prototipo, el algoritmo KKLayout se usa para generar una distribución base cuando se utiliza por primera vez un esquema. Sobre esta base se hacen ajustes manuales para lograr mayor legibilidad del grafo. La distribución ajustada del grafo se guarda con coordenadas en un archivo GraphML. Esta distribución ajustada manualmente es la que se despliega las siguientes veces que se usa el esquema. La última modificación se realizó debido a que la disposición que arroja el KKLayout no siempre es lo suficientemente legible, se presentan algunos cruces de arcos y mezcla, en la distribución espacial, de atributos de diferentes clases. Esto, unido al hecho de que cada vez que el usuario ingresa a la aplicación encuentra una distribución diferente del esquema, lo cual puede generale confusión y llevarlo a cometer errores en la formulación de la consulta.

6.4 Prueba comparativa

El objetivo de esta prueba de usabilidad es comparar una herramienta gráfica de consulta sobre grafos y el prototipo de la herramienta desarrollada, para evaluar qué tanto la herramienta les facilita a los usuarios formular consultas que involucran filtros y datos incompletos. La prueba se enfocó en la formulación de la consulta y no incluyó

¹<http://jung.sourceforge.net/>

²<http://www.w3.org/XML/>

la presentación de los resultados de la misma.

Este fue un estudio de usabilidad, exploratorio, comparativo y *between-subjects* [135, 165]. La prueba incluyó la interfaz gráfica para la formulación de consultas en SPARQL de *Gruff* y GraphTQL. *Gruff*¹ es un navegador interactivo para el *triple-store* AllegroGraph², permite editar y administrar consultas y datos, también despliega los grafos de datos en una representación visual del grafo o en tablas. *Gruff* provee un editor gráfico de consultas SPARQL que permite crear las consultas por medio de diagramas formados por nodos y arcos. Estos nodos y arcos representan variables u objetos del *triple-store*.

Cuatro profesionales de la salud y cuatro estudiantes de medicina participaron en la prueba comparativa. La mitad de los participantes evaluó GraphTQL y la otra mitad, la interfaz gráfica de consulta. El diseño de la prueba se presenta en el Anexo G. El plan de esta prueba incluyó, además de los items de la segunda prueba de evaluación, una última etapa en la que los participantes respondieron un cuestionario SUS (*System Usability Scale*) [14]. De esta manera, el plan de esta prueba consta de:

1. Apertura: El moderador explica el propósito de la prueba y las partes que la integran. El participante diligencia un cuestionario pre-test, que recoge información de su perfil de formación, y firma el formato de consentimiento para realizar la prueba.
2. Demostración de la herramienta que el participante va a evaluar: se hace sobre una base de datos de películas y series de televisión. El moderador explica las opciones, botones y operaciones de la herramienta, y desarrolla tres ejemplos de formulación de consultas.
3. Actividades: el participante realiza cuatro actividades, en cada una se pide al participante que formule una consulta. Para este caso se toma una base de datos clínicos. Los participantes cuentan con un tiempo máximo de 15 minutos para formular la consulta de cada actividad. Durante las actividades de evaluación el usuario puede pedir la ayuda del moderador o acudir a las ayudas de la herramienta.
4. Cuestionario post-test. El usuario responde el SUS y una pregunta abierta para expresar sus opiniones y sugerencias con respecto a la herramienta.

Los modelos de datos usados en la demostración y en las actividades son diferentes porque una de las preguntas del estudio apunta a evaluar si la representación del grafo esquema facilita la formulación de las consultas. Por lo tanto, es importante observar qué ocurre cuando el participante empieza a usar un modelo de datos diferente.

En ambos modelos, de películas y de historias clínicas, se usaron datos sintéticos. Los datos de las películas se generaron modificando la base de datos que provee Gruff³. Mientras que para los datos de las historias clínicas se creó un generador de datos. Para la prueba comparativa se generaron conjuntos de datos pequeños, basados en casos en

¹<http://franz.com/agraph/gruff/>

²<http://franz.com/agraph/allegrograph/>

³<http://franz.com/agraph/gruff/>

los que se relacionan diagnósticos con síntomas, pruebas de laboratorio, procedimientos, medicamentos, imágenes diagnósticas y especialidades médicas. De esta manera, los datos tenían alguna coherencia para los participantes.

Consultas de las actividades

Las consultas se ambientan en dos escenarios: en el primero, un médico está en su consultorio atendiendo la consulta de un paciente; en el segundo escenario, un médico está buscando historias clínicas de pacientes que cumplen ciertas características para iniciar una investigación. Estos escenarios se eligieron entre aquellos identificados en la entrevista preliminar y las pruebas anteriores. Durante el desarrollo de estas pruebas, los profesionales de la salud participantes dieron 31 ejemplos de consulta ([Anexo B](#)). De las 31 consultas recopiladas, 11 requieren funciones de agregación y 4, comparar variables (comparaciones entre nodos del grafo). Estos dos tipos de pregunta no se incluyeron en esta prueba dado que las comparaciones entre variables no se han definido para la herramienta y por tanto no están implementadas, además de que las funciones agregadas hacen parte de trabajo futuro del sistema de consulta propuesto. Las 16 preguntas restantes, que se pueden especificar en la versión prototipo actual del sistema de consulta, se clasificaron de acuerdo con el nivel de complejidad que se presenta en la Tabla [6.4](#). De acuerdo con esta clasificación se recopilaron 5 consultas de nivel 1, 8 de nivel 2 y 3 de nivel 3. La escala de complejidad de las consultas (Tabla [6.4](#)), se propuso de manera similar a la clasificación propuesta por Bell y Rowe [\[22\]](#) y Owei et al. [\[141\]](#), teniendo en cuenta los operadores de transformación de grafos y la hipótesis de este estudio.

Tabla 6.4: Clasificación de las consultas según su complejidad.

Nivel de Complejidad	La consulta incluye
1	<i>Pattern matching</i> con una condición de filtro, con o sin datos opcionales
2	<i>Pattern matching</i> con más de una condición de filtro, con o sin datos opcionales
3	<i>Pattern matching</i> con condiciones de filtro y recuperando todos los datos conectados a una clase seleccionada
4	<i>Pattern matching</i> con condiciones de filtro y otras operaciones (ej. unión)

Las consultas seleccionadas para las actividades de la prueba son de mediana complejidad, en el sentido que incluyen patrones formados por varias tripletas y que requieren encontrar objetos de una clase A que están conectados por medio de un

camino en particular con objetos de la clase B que cumple un filtro. Por tanto, en esas consultas los filtros no tienen solamente alcance local (i.e. para afectar solamente el tipo de nodo o la variable involucrada en la expresión de filtro), sino que también inciden sobre otros nodos. Por ejemplo, en la consulta “Encuentre los pacientes que han recibido un diagnóstico de diabetes”, el filtro se especifica sobre “diagnóstico”; sin embargo, la consulta pide los pacientes que están conectados con ese diagnóstico. Dado que todos los patrones de las consultas de la prueba tienen este nivel de complejidad, los niveles planteados en la Tabla 6.4 no tienen en cuenta cuan simple (una o dos tripletas) o complejo (varias tripletas) es el patrón de consulta. Las consultas se formularon teniendo en cuenta además otras consideraciones, por ejemplo, cuando se usa Gruff se requiere un tiempo para dibujar las consultas, por lo tanto se limitó la cantidad de nodos que se requieren en las consultas. Adicionalmente, no hay un equivalente al *Path Contraction* en Gruff, por lo cual las consultas no incluyen esta operación. Resultados equivalentes se pueden obtener usando la cláusula *construct*, que no está soportada en Gruff. Adicionalmente, los ejemplos no necesitan de operaciones SPARQL como la *union*, para evitar sobrecargar de conceptos a los participantes que evalúan la herramienta.

En total se seleccionaron 7 consultas: 3 ejemplos para ilustrar el uso de la herramienta y 4 para las actividades de evaluación. Estas consultas, que se muestran en las Tablas 6.5 y 6.6, son de nivel 1, 2 o 3, debido a que un alto porcentaje de los ejemplos dados por los médicos corresponden a estos niveles.

Tabla 6.5: Consultas para los ejemplos.

Consulta	Nivel de Complejidad
Se requiere encontrar los títulos de las películas galardonadas con la categoría “ <i>Best Picture Academy Award</i> ” a partir del año 2000. Además, se desea incluir el nombre de los actores y el género de la película, si están disponibles	2
De los actores que han ganado un premio en la categoría “ <i>Outstanding Lead Actor</i> ”, se requieren todos los datos disponibles de: títulos de las películas en que han actuado, categoría y año de los premios que han recibido. (Note que estos datos incluyen los otros premios, además de “ <i>Outstanding Lead Actor</i> ”)	3
Se requiere los nombres de los actores que han trabajado en películas de género “ <i>Comedy</i> ” y también han trabajado bajo la dirección de “ <i>Robert Zemeckis</i> ”. (Observe que pudieron ser películas diferentes, pero el actor cumple con ambas condiciones)	2

Tabla 6.6: Consultas para las actividades.

Consulta	Nivel de Complejidad	Escenario
Se requieren las enfermedades (incluyendo su descripción) registradas en la historia familiar del paciente con identificación 723628; incluir la relación familiar si está disponible	1	I
Se requiere encontrar los diagnósticos (incluyendo su descripción) dados al paciente con identificación 723628 por médicos con especialidad en “ENDOCRINOLOGY” y, si está disponible, las fechas de las consultas y las medicinas que le recetaron en ese momento (incluyendo descripción y dosis)	1	I
De los pacientes que han tenido un diagnóstico de “OSTEOMYELITIS” desde el año 2000 (desde 01-01-2000), se requieren todos los datos de: identificación del paciente, diagnósticos recibidos, y procedimientos que se le han realizado (incluir la descripción de los diagnósticos y procedimientos). Note que estos datos incluyen los otros diagnósticos, además de OSTEO MYELITIS, y todos los datos que se piden son independientes de la fecha en que se registraron	3	II
Se requiere encontrar los pacientes que han tenido un estudio de imágenes diagnósticas con anotación “GALLSTONES” y se les ha realizado el procedimiento “COLECTOMY”. De estos pacientes se requiere la fecha de nacimiento, el género y la raza, si están disponibles. El estudio y el procedimiento pudieron realizarse en encuentros diferentes, pero cada paciente cumple con ambas condiciones	2	II

La primera consulta de las actividades es sencilla, incluye una condición (identificación del paciente) y datos opcionales (relación familiar). La segunda consulta incluye la conjunción de dos condiciones de filtro y datos opcionales. La tercera consulta también incluye la conjunción de dos condiciones de filtro y datos opcionales. Adicionalmente, para formular la consulta en SPARQL se requieren dos variables para representar los diagnósticos, y dos para su descripción. De esta manera se incluyen otros diagnósticos adicionales a los de la condición de filtro. En GraphTQL esto se logra usando el operador *Filter+Additional Data*. La cuarta consulta incluye la conjunción de dos condiciones de filtro y datos opcionales. Para formular la consulta en SPARQL el patrón de consulta debe incluir dos variables para los encuentros, uno que se conecte con cada condición. En GraphTQL basta con responder a las preguntas

del diálogo de clarificación. De manera que en el encuentro (nodo *encounter*) se aplica una disyunción, y en el paciente una conjunción.

Juicio de expertos

Un juicio de expertos busca la opinión informada de expertos cualificados en un tema a evaluar. Como parte de la especificación de la prueba comparativa de usabilidad, se planteó la conveniencia de realizar un juicio de expertos para evaluar la idoneidad de las actividades que se proponen en la prueba. Para ello, se pidió a un experto en consultas sobre datos de grafos que evaluara si el objetivo, las preguntas y la hipótesis del estudio son coherentes con:

- La clasificación de las consultas del estudio según su nivel de dificultad.
- Las consultas que se proponen para cada actividad.
- Las métricas y la clasificación de las ayudas.

El experto que realizó el juicio es doctor (PhD), investigador en web semántica, redes sociales y análisis de contenidos. Su trabajo en estas áreas implica el uso de técnicas de procesamiento de lenguaje natural, recuperación de la información y ontologías. Adicionalmente, es experto en SPARQL y ha trabajado con otras bases de datos orientadas a grafos, como Neo4j.

Para la evaluación, se entregó un documento incluyendo:

1. Una introducción que explica el contexto, el trabajo desarrollado y la motivación de la realización de la prueba.
2. La descripción de las herramientas a comparar y la forma como se formulan las consultas en cada una de ellas.
3. El objetivo, preguntas e hipótesis del estudio comparativo.
4. La explicación de los niveles de dificultad de las consultas, de la clasificación de ayudas y de las métricas de la prueba.
5. Las consultas propuestas para cada actividad y una descripción de la respuesta a estas consultas en cada una de las herramientas.
6. Un cuestionario para que el experto realice su evaluación.

Las respuestas del experto fueron las siguientes:

- **¿Es coherente la clasificación del nivel de dificultad de las consultas con el objetivo, las preguntas y las hipótesis del estudio?**

“Creo que si son coherentes ya que los niveles definidos indican un valor que se incrementa proporcionalmente a la complejidad, medida en términos de las operaciones que se llevan a cabo, de las consultas que cada nivel representa.”

Sin embargo, creo que desde el punto de vista de complejidad en la formulación no es lo mismo una consulta donde se usa una sola tripleta que una consulta donde se encadenen 3 o más tripletas dado que esto requiere conocer mejor el modelo y a que se pueden cometer más errores. Observe que no estoy diciendo que deba haber más

niveles sino que hay que justificar porqué se decide mantener la categoría *Pattern Matching* como única.”

■ **¿Es coherente la clasificación de las ayudas con el objetivo, las preguntas y la hipótesis del estudio?**

“Si, la clasificación ayuda a determinar que tanto soporte y de qué nivel necesitan los usuarios para llevar a cabo sus tareas, y por tanto podrá ser útil al determinar si un usuario fue totalmente autónomo o no al conseguir sus resultados.”

■ **¿Son coherentes las métricas con el objetivo, las preguntas y la hipótesis del estudio?** (Las métricas que se tenían en ese momento para la prueba se cambiaron atendiendo a los comentarios del experto)

“Respecto a esto tengo algunas dudas. La definición de completa y precisa parece que se refiere a los conceptos de *recall* y *precisión* usados en recuperación de información. Primero, las definiciones dadas en el texto acerca de estos conceptos no se entienden. Sugiero hacer algún tipo de formalización y ejemplos que ayuden a entender los conceptos. Segundo, si realmente se refieren a estos conceptos de *precisión* y *recall* entonces hay que tener en cuenta que fueron desarrollados para motores de búsqueda que podían dar respuestas tanto erróneas como incompletas de ahí la necesidad de medir los dos aspectos. Sin embargo, en esta evaluación se supone que el motor de consulta funciona bien y lo que se está evaluando es la formulación correcta de una consulta.

La pregunta que hay que plantearse es si realmente son necesarias estas dos métricas, o si una única métrica del tipo respuesta errónea sería suficiente en este caso. Otra opción sería buscar en la literatura si existen otras métricas para la formulación de consultas.”

■ **¿Son coherentes las consultas que se proponen para cada actividad con el objetivo, las preguntas y la hipótesis del estudio?**

“Si, son coherentes porque cubren los diferentes niveles y la casuística de interés en esta evaluación. No obstante, me parece que son pocas consultas por nivel para poder sacar alguna conclusión. Al menos debería haber 2 preguntas por cada nivel. Sería interesante ver para un mismo nivel de consulta como el mismo usuario se desempeña con el objetivo de contrastar si la formulación correcta fue debido a que entiende cómo funciona el sistema o a la suerte.”

La última observación, sobre el número de consultas por nivel, es completamente razonable, sin embargo no se hizo esta modificación a la prueba porque el tiempo de realización de la misma se incrementa, lo cual dificulta conseguir participantes.

Aplicación de la prueba y resultados

Antes de aplicar la prueba se hizo un *pre-test* en el que participó un ingeniero de sistemas, con el fin de corregir detalles del plan de pruebas.

Este estudio midió dos aspectos de la experiencia de usuario: rendimiento y satisfacción [192]. Las métricas de rendimiento para cada actividad son:

- **Efectividad:** mide el grado con que la consulta que propone el usuario, retorna información correcta. Se definen dos medidas de efectividad: (a) número de consultas correctamente formuladas, y (b) número de errores cometidos en la formulación de la consulta. Dado que en una misma actividad pueden ocurrir varios errores del mismo tipo, en cada actividad se contabiliza el número de errores de diferente tipo, no el total de apariciones del mismo tipo de error. Por ejemplo, si en una consulta hay tres arcos que se debían marcar como opcionales, pero no se marcaron, se contabiliza un error.
- **Eficiencia** en la formulación de la consulta: mide el tiempo (minutos y segundos) requerido para formular cada consulta.
- **Número de veces que recurre a la ayuda:** se registra clasificada según la clase de ayuda. Las ayudas se clasificaron de acuerdo con el nivel de impacto que pueden tener en la eficiencia del desarrollo de la actividad, así:
 - Ayudas Clase A: Con mucho impacto en la efectividad. Estas son las ayudas en las que el moderador da pistas o genera preguntas que llevan al participante a re-pensar la solución que está proponiendo.
 - Ayudas Clase B: Con poco impacto en la efectividad. Son ayudas en las que el moderador aclara o explica de nuevo la semántica de las operaciones.
 - Ayudas Clase C: Sin impacto en la efectividad. Estas incluyen la consulta de manuales o ayudas de la herramienta y las ayudas en las que el moderador explica el funcionamiento de la interfaz (ej. use clic derecho, arrastre, en que menú aparece una opción) o aclara lo que se busca recuperar en la consulta de la actividad. Se considera que las ayudas clase C afectan la eficiencia (tiempo que toma realizar la actividad) mas no la efectividad (capacidad de terminar la actividad con éxito) ya que esta información no afecta la solución que el usuario plantea, sino la forma de representar esa solución en la herramienta.

La **satisfacción del usuario** se midió a partir de la percepción de los participantes con respecto a la usabilidad del lenguaje. Para ello se aplicó el cuestionario SUS [32] (*System Usability Scale*), con las modificaciones propuestas por Bangor et al. [14]. Las respuestas del cuestionario se traducen en un valor que representa la medida de usabilidad del sistema, este número se calcula con base en la respuesta de cada una de las diez preguntas del cuestionario. La mitad de esas preguntas son afirmaciones positivas sobre la herramienta y la otra mitad son afirmaciones negativas. El puntaje final califica la herramienta en un rango de 0 a 100.

Las Tablas 6.7 y 6.9 muestran los resultados por participante y las Tablas 6.8 y 6.10 los resultados promedio. El Anexo H presenta un resumen de las pruebas realizadas.

Hay diferencias importantes en el tiempo requerido para la formulación de consultas

en la interfaz gráfica de SPARQL y en el requerido en GraphTQL. La formulación en SPARQL requirió en promedio tres veces el tiempo que se requirió en GraphTQL. Posiblemente esto se debe a que el usuario construye el patrón de consulta desde cero y a que constantemente debe cambiar de ventana, de *graphical query view* a *graph view*, para buscar cómo construir de forma correcta el patrón de consulta. Algunos usuarios trataron de minimizar este cambio de ventanas dejando como muestra de ejemplo el patrón que habían construido en la consulta anterior o modificando ese patrón.

A juzgar por el número de respuestas correctas (cero) y de errores (3.1 en promedio por consulta) que se presentaron al usar la interfaz gráfica de SPARQL, la formulación de consultas en SPARQL es difícil si no se tiene un entrenamiento prolongado. A pesar de ello, se notó que los participantes realizaron las actividades con seguridad y confianza, y aunque requirieron mas ayuda que los participantes que usaron GraphTQL, la mayoría de estas ayudas estaban relacionadas con el manejo de la interfaz (ej. ubicación de menús, cómo llegar a ciertas opciones), no con la formulación de la consulta.

Es interesante notar que, en algunos casos, la evaluación de usabilidad para la interfaz gráfica de SPARQL fue alta a pesar de que las consultas se formularan de forma incorrecta. Esto puede indicar que a estos participantes les gustó la herramienta por su potencial utilidad en su campo profesional.

De otra parte, el tiempo total de realización de la prueba fue mas prolongado para la interfaz gráfica de SPARQL, en parte por el tiempo usado por los participantes para formular las consultas y también debido a que el tiempo para la demostración de la herramienta fue mayor para la interfaz gráfica. El tiempo promedio requerido para la demostración fue de 50 y 35 minutos respectivamente para la interfaz gráfica y GraphTQL.

A partir de las pruebas con la interfaz gráfica de SPARQL es posible afirmar que la necesidad de explorar el grafo de datos para construir el patrón de consulta tiende a introducir errores porque se equivoca la dirección de los arcos o se asume alguna representación de una relación de manera diferente a la que se establece en los datos. En general los participantes no tienen en cuenta el concepto de datos opcionales y pueden confundir nodos que representan una variable con nodos que representan un valor literal. Además, aunque los participantes identificaron en cuales consultas requerían definir mas de una variable para representar objetos del mismo tipo, no encontraron la forma de especificar los patrones de manera que se diera respuesta a la consulta requerida.

Tabla 6.7: Prueba comparativa - Medidas de rendimiento por participante.

Participante, Herramienta	Consulta 1				Consulta 2			
	Tiempo	Correcta	Errores	Ayuda	Tiempo	Correcta	Errores	Ayuda
1, Gruff	4min 44seg	No	3	Clase C	9min 32seg	No	3	No
2, Gruff	13min 46seg	No	2	Clase B y C	15min 47seg	No	4	Clase A
3, Gruff	12min 22seg	No	3	Clase C	14min 58seg	No	2	Clase C
4, Gruff	13min 56seg	No	1	Clase B y C	12min 6seg	No	2	Clase B
5, GraphTQL	8min 28seg	Si	0	Clase B	3min 22seg	No	1	No
6, GraphTQL	2min 52seg	Si	0	Clase C	2min 44seg	Si	0	Clase C
7, GraphTQL	1min 56seg	Si	0	No	4min 8seg	Si	0	No
8, GraphTQL	3min 38seg	Si	0	Clase C	2min 51seg	No	1	No
Participante, Herramienta	Consulta 3				Consulta 4			
	Tiempo	Correcta	Errores	Ayuda	Tiempo	Correcta	Errores	Ayuda
1, Gruff	8min 42seg	No	5	Clase C	12min 52seg	No	1	Clase A
2, Gruff	10min 30seg	No	8	Clase B y C	6min 58seg	No	6	No
3, Gruff	11min 38seg	No	4	Clase C	10min 31seg	No	1	No
4, Gruff	12min 21seg	No	3	Clase B y C	7min 39seg	No	2	No
5, GraphTQL	4min 25seg	Si	0	Clase B	3min 13seg	No	4	No
6, GraphTQL	3min 35seg	Si	0	Clase C	1min 58seg	No	2	Clase C
7, GraphTQL	3min 32seg	Si	0	No	3min 48seg	No	2	No
8, GraphTQL	4min 45seg	Si	0	Clase C	1min 21seg	No	1	No

Tabla 6.8: Prueba comparativa - Medidas de rendimiento promedio.

Consulta	Gruff			GraphTQL				
	Tiempo	Correctas	Errores	Ayuda	Tiempo	Correctas	Errores	Ayuda
1	11min 12seg	0%	2,25	B:2, C:4	4min 11seg	100%	0	B:1, C:2
2	13min 8seg	0%	2,75	A:1, B:1, C:1	3min 17seg	50%	0,5	C:1
3	10min 48seg	0%	5	Ninguna	4min 25seg	100%	0	C:3
4	10min 31seg	0%	2,25	A:1	2min 35seg	0%	2,5	C:1
Todas	11min 24seg	0%	3,1	A:2, B:3, C:5	3min 31seg	62,5%	0,7	B:1, C:7

Tabla 6.9: Prueba comparativa - Medida de satisfacción por participante.

Participante	Herramienta	Puntaje SUS
1	Gruff	22,5
2	Gruff	80
3	Gruff	55
4	Gruff	55
5	GraphQL	75
6	GraphQL	70
7	GraphQL	72,5
8	GraphQL	70

Tabla 6.10: Prueba Comparativa - Medida de satisfacción promedio.

Herramienta	Puntaje SUS
Gruff	53,0
GraphQL	71,9

De otra parte, comparando los resultados de la formulación de consultas en GraphQL de la segunda prueba de evaluación (Tabla 6.3) con los de la prueba comparativa (Tabla 6.8), se observa un incremento sustancial en el porcentaje de consultas formuladas correctamente (18,8% en la prueba de evaluación y 62,5% en la prueba comparativa) y un descenso importante en el número promedio de tipos de error ocurridos en la formulación (1,5 en la prueba de evaluación y 0,7 en la prueba comparativa). Esto muestra que los cambios que se incorporaron en la herramienta, como resultado de las observaciones obtenidas de la prueba de evaluación, tuvieron un impacto positivo. Sin embargo, en la prueba comparativa se identifica que es necesario hacer más claridad en la pregunta de desambiguación para el caso en que el usuario necesita elegir diferentes operadores lógicos (conjunction/disjunction) en diferentes puntos de bifurcación del patrón exigido en la consulta (caso de la actividad 4).

Finalmente, todos los participantes coincidieron en que una herramienta que les permita formular este tipo de consultas sería muy útil en la atención a pacientes, la administración y la investigación médica.

6.5 Discusión

Como parte de la aplicación de técnicas de diseño centrado en el usuario, durante el desarrollo de este proyecto, se realizaron cinco pruebas de usabilidad: una entrevista

preliminar, una prueba exploratoria, dos pruebas de evaluación de prototipos y una prueba comparativa. Los resultados de estas pruebas y la información recopilada en ellas, permitieron identificar con más precisión las necesidades del usuario y tomar decisiones de diseño con criterios diferentes, más amplios, de los que se aplican en un proceso de desarrollo tradicional. Estas decisiones reflejan la importancia que tiene la interacción entre el usuario y la aplicación en el resultado esperado.

Si bien, GraphQL es un lenguaje de uso general, que puede ser usado en otros dominios de aplicación, varias decisiones de diseño se tomaron con base en la información recopilada en los diferentes estudios realizados con médicos, profesionales de áreas de la salud y estudiantes de medicina. Gracias a la realimentación obtenida en dichos estudios se podría esperar que GraphQL se ajuste mejor a las necesidades de este dominio.

En todos los estudios realizados los participantes resaltaron la utilidad de una herramienta de consulta de este tipo para el desarrollo de sus actividades. La entrevista preliminar permitió conocer los diferentes escenarios de la actividad médica en que la herramienta sería útil y, a través de los ejemplos de consulta recopilados, identificar que estas son de mediana complejidad y que las funciones agregadas tienen un papel importante, una vez se han seleccionado los grupos de datos que son de interés para un análisis particular.

A partir de los ejemplos de consulta recopilados, se pudo identificar que la pregunta se propone alrededor de una clase u objeto particular, lo cual permitió definir en GraphQL el concepto de *Clase de Interés*, similar a lo propuesto en otros lenguajes [37, 98]. En esos ejemplos también se observa que las consultas de conectividad no son usuales y, dado que los datos suelen ser incompletos, expresar dichas consultas con patrones genera la necesidad de usar operaciones para realizar el manejo explícito de los datos incompletos. El uso de estos operadores hace más complejo el análisis que se requiere para realizar la formulación de la consulta. Para reducir ese grado de complejidad en la formulación, los operadores de GraphQL se diseñaron para que manejen de manera implícita los datos incompletos.

De otra parte, por requerirse consultas de mediana complejidad, que involucran la composición de expresiones lógicas, se vio la necesidad de facilitarle al usuario la formulación de dichas expresiones. Esto llevó a incluir en GraphQL los diálogos de clarificación de las condiciones de filtro.

Las pruebas exploratoria, de evaluación y comparativa mostraron que, si bien el grafo esquema puede parecer complejo y sobrecargado de información, los usuarios del dominio médico pudieron rápidamente asimilarlo, leerlo y usarlo en la formulación de las consultas. Esto puede ser debido a su profundo conocimiento del dominio de aplicación y a que en sus labores diaria usan sistemas de información donde registran los datos representados en con el grafo esquema. Esto permite suponer que el grafo esquema puede ser igualmente aceptado por usuarios en otros dominios de aplicación que cumplan las mismas características. Sin embargo, es importante usar el vocabulario

correcto para representar los conceptos del dominio en el grafo esquema. Las pruebas mostraron que usar vocabulario incorrecto genera dificultades para entender y usar el grafo esquema.

Las pruebas de evaluación evidenciaron dificultades con la selección de los caminos que conectan los nodos seleccionados para la consulta. Por ello se decidió incluir en GraphTQL el diálogo de clarificación de caminos para los casos en que se encuentra más de un camino posible, y la marca automática para los casos en que solamente hay un camino.

Finalmente, en las pruebas se evidenció la importancia que dan los usuarios a vocabularios estructurados propuestos en el dominio médico tales como el ICD10¹ usado para describir los diagnósticos. Dado que en este dominio se dispone además de ontologías de dominio y otros vocabularios estructurados, como CPT², para la descripción de los procedimientos; LOINC³, para observaciones médicas de laboratorio; o SNOMED CT⁴ que incluye términos usados en reportes y documentos médicos. Incluir estos vocabularios en los datos del grafo instancia permite estandarizar esos datos en el modelo y, a futuro, extender GraphTQL para tomar ventaja de el conocimiento representado en ellos para, por ejemplo, extender las consultas.

¹<http://apps.who.int/classifications/icd10/browse/2015/en>

²<http://www.ama-assn.org/go/cpt>

³<http://loinc.org>

⁴https://www.nlm.nih.gov/research/umls/Snomed/snomed_main.html

Capítulo 7

Implementación de GraphTQL y Evaluación de los Operadores de Nivel Lógico

GraphTQL se propone como la capa superior, de interacción con el usuario final, en una arquitectura que permite acceder a los datos de un sistema de información en un dominio de aplicación específico. La estructura funcional del sistema, implementada en el prototipo desarrollado, se presenta a continuación, en la Sección 7.1. Los detalles de la implementación y evaluación de la ejecución de los operadores de nivel lógico se describen en la Sección 7.2. Adicionalmente, se exploraron algunas ideas relacionadas con la utilidad de GraphTQL como mecanismo de acceso a datos de fuentes de datos heterogéneas, integradas en una arquitectura basada en mediación. Estas ideas se describen en la Sección 7.3.

7.1 Estructura Funcional del Sistema

El prototipo funcional desarrollado para GraphTQL (Figura 7.1) consta de tres componentes: la GUI de GraphTQL, el ejecutor de las consultas y el gestor de la base de datos. La GUI de GraphTQL soporta el proceso de formulación de las consultas, el ejecutor de consultas implementa los algoritmos de los operadores de nivel lógico, presentados en la Sección 7.2.1, y el gestor de base de datos soporta la administración de los datos en un modelo de datos orientado a grafos.

Los dos primeros componentes, la GUI y el ejecutor, se desarrollaron en Java¹ y para el último componente, el gestor, se usó Neo4j² [160].

La GUI de GraphTQL incluye los módulos que soportan: a) la interacción general

¹<https://www.oracle.com/java/>

²<http://neo4j.com>

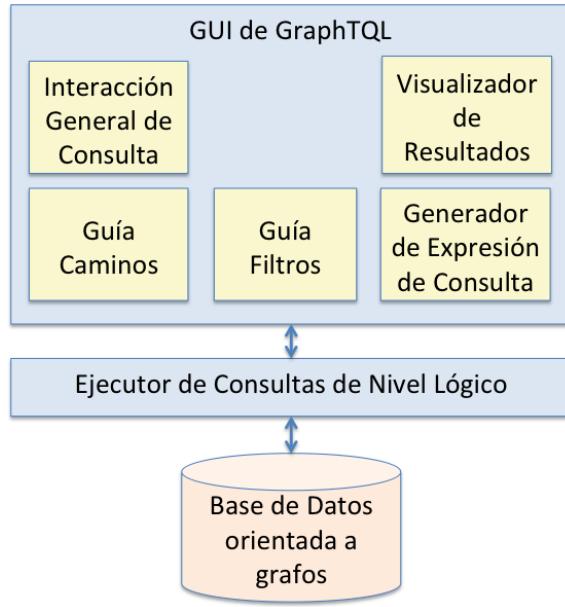


Figura 7.1: Estructura Funcional del Sistema

durante la formulación de la consulta; b) la guía para la selección de los caminos en el grafo; c) la guía para la definición de las condiciones de los filtros en las consultas; d) la generación de la expresión de la consulta usando los operadores de nivel lógico; e) el visualizador de resultados, que aún no está incluido en el prototipo funcional desarrollado.

Interacción general durante la formulación de la consulta

El paradigma de manipulación directa del grafo esquema se implementó usando el *framework* JUNG¹ para la visualización del grafo. La interacción que provee JUNG por defecto, se modificó para que los comandos requeridos en GraphTQL se aplicarán usando solamente el clic, izquierdo y derecho, del ratón. Los nodos y arcos se seleccionan haciendo clic izquierdo sobre ellos y, si están seleccionados, se de-seleccionan de la misma forma.

El clic derecho aplicado sobre un nodo objeto o de valor compuesto, lo marca como Clase de Interés. Cuando el nodo se aplica sobre nodos de valor básico, se despliega un diálogo que permite al usuario marcarlo como clase de interés o definir una condición de filtro sobre su valor.

GraphTQL usa el formato de archivos GraphML [29] para describir el grafo esquema. GraphML provee un lenguaje para describir las propiedades estructurales del grafo a través de atributos asociados a cada nodo o arco en el grafo.

¹<http://jung.sourceforge.net/>

JUNG provee diferentes algoritmos para calcular la distribución de los nodos y arcos del grafo a desplegar. En GraphTQL se usa el algoritmo Kamada-Kawai [103] (KKLayout) cuando en el archivo GraphML no se incluye la ubicación de los nodos en el diagrama del grafo. Este algoritmo se seleccionó a partir de un estudio y pruebas que incluyeron diferentes algoritmos [67]. Este estudio tuvo en cuenta, como características principales, las siguientes capacidades del mecanismo de visualización: capacidad para visualizar grafos de tamaño medio (alrededor de 1000 nodos), visualización estética y capacidad para disminuir el cruce de arcos y el solapamiento de nodos. Sin embargo, al calcular la distribución de los grafos esquema de prueba, se observaron algunos cruces de arcos y, si bien los nodos no se solapan, la distribución espacial mezcla atributos de diferentes clases. Esto afecta la legibilidad del diagrama y puede hacerlo difícil de entender para los usuarios. Adicionalmente, si la distribución del grafo esquema se calcula cada vez que se ejecuta la aplicación, el usuario encontrará cada vez una distribución diferente del mismo grafo, lo cual puede generar confusión, requerir un tiempo de adaptación y por tanto, hacer menos eficiente el proceso de formulación de la consulta. Por estas razones se optó por usar KKLayout se usa para generar una distribución base cuando se utiliza por primera vez un esquema, hacer ajuste manuales para incrementar la legibilidad del grafo y almacenar este grafo con las posiciones de sus nodos en un archivo GraphML que se usa en adelante. De esta manera, el usuario siempre encuentra desplegado, al inicio de la ejecución, el mismo diagrama de grafo.

En el prototipo funcional, soporta parcialmente la generación del archivo GraphML con las coordenadas de los nodos. Estos archivos se generan como parte del historial de la interacción, y actualmente, se debe hacer la copia manual del archivo a la carpeta donde se ubican los archivos GraphML que se despliegan al iniciar la ejecución de la herramienta. En una versión posterior, este proceso se debería incluir como parte de las tareas del administrador del sistema, de manera que con apoyo de la herramienta se puedan realizar todos los pasos.

Guía para la selección de caminos en el grafo

Cuando se selecciona un nodo n , GraphTQL lo conecta automáticamente a los otros nodos previamente seleccionados, siempre y cuando haya un solo camino para ello. Cuando hay más de un camino, GrahTQL presenta un diálogo de clarificación de caminos. En este diálogo el usuario elige uno o varios caminos para marcar en el grafo. Los caminos se encuentran haciendo un recorrido en profundidad del grafo a partir del nodo n .

Los recorridos sobre el grafo esquema son eficientes en términos de tiempo de ejecución, ya que este grafo tiene, generalmente, un tamaño mucho menor que el grafo instancia correspondiente.

Guía para la definición de las condiciones de filtro

Este módulo implementa el comportamiento de los diálogos de clarificación de filtros explicado en el Capítulo 4 (página 103).

Generación de la expresión de la consulta

GraphTQL usa XML¹ para especificar, en nivel lógico, la consulta que el usuario formula. Ejemplos de los archivos XML generados se muestran al final del Anexo I.

Ejecutor de consultas de nivel lógico

Este módulo se encarga de ejecutar las operaciones de la expresión de nivel lógico de las consultas. Para ello implementa los algoritmos presentados en la Sección 7.2.1.

7.2 Implementación de los operadores

De manera amplia, se identifican dos mecanismos básicos de consulta en los lenguajes de consulta sobre grafos de datos: la búsqueda de patrones (*pattern matching*) y el recorrido de caminos (*path traversal*).

Un patrón es un grafo cuyos nodos y arcos pueden estar etiquetados con variables o literales. Las consultas basadas en patrones recuperan de la base de datos todas las ocurrencias que coinciden con el patrón. Generalmente, la búsqueda de patrones se define en términos del isomorfismo de subgrafos [58].

El recorrido de caminos (*path traversal*) [162] consiste en navegar por el grafo siguiendo los arcos para encontrar los caminos que parten de un nodo o un conjunto de nodos determinado y siguen un camino de ciertas características, especificadas en la consulta. Por tanto, en el recorrido se visitan elementos del grafo (nodos o arcos) avanzando con pasos individuales, es decir avanzando por los arcos que salen de un nodo o llegan a él.

De acuerdo con Rodriguez y Neubauer [162] y Dayarathna y Suzumura [50] una de las ventajas de las bases de datos de grafos es el bajo costo computacional de los recorridos de caminos. Esto se debe a que los datos se representan en estructuras de grafos, por tanto cada elemento del grafo tiene referencias directas a los elementos adyacentes. En este sentido, Angles et al. [8] comparan los tiempos de respuesta en la ejecución de doce consultas sobre datos de una red social gestionadas en motores de bases de datos con modelos diferentes. Dos motores implementan un modelo de grafos (DEX y Neo4j), uno el modelo RDF (RDF3X) y dos el modelo relacional (Virtuosu y PostgreSQL). Las consultas están clasificadas según la operación que realizan:

¹<http://www.w3.org/XML/>

selección, *path traversal*, *pattern matching* y funciones agregadas. Los resultados del *benchmark* muestran que las bases de datos de grafos tienen mejor desempeño al ejecutar estas operaciones.

Como se describió en el Capítulo 5, los operadores de alto nivel de GraphTQL se reescriben en términos de los operadores de nivel lógico. Estos a su vez, se pueden reescribir en SPARQL (consultas basadas en *pattern matching*) o implementar sobre un lenguaje o un API basado en *path traversals*. La reescritura en SPARQL requiere patrones básicos y las cláusulas *optional*, *filter* y *union*. En cuanto a los *path traversals*, dado que las entradas de *Logic Level Filter*, *Selective Union* y *Path Contraction* se definen en términos de los caminos que unen la clase de interés con los otros nodos involucrados en la consulta, es posible ejecutarla resolviendo la parte que corresponde a cada camino y consolidando los resultados parciales. Con el ánimo de tomar ventaja de los tiempos de ejecución que los motores de bases de datos de grafos prometen para los recorridos de caminos, se aplicó este enfoque en la implementación de los operadores de nivel lógico de GraphTQL. Para ello se usó la base de datos Neo4j y el Neo4j Traversal API [185].

Con el fin de realizar una prueba de concepto de la ejecución de los operadores de alto nivel de GraphTQL, se implementó el *Filter*, el *Filter+Additional Data* y el *Select a Portion*. En esta implementación, estos operadores reciben, entre otras entradas, los grafos esquema e instancia y entregan como resultado un conjunto de arcos (o triplets). La respuesta se despliega en pantalla como una lista de triplets. No se implementó la composición de los operadores porque que Neo4j no ofrece representación del grafo en memoria y el almacenamiento en disco de los grafos resultado es costoso. Entonces, para soportar los operadores de alto nivel mencionados, se implementaron los operadores de nivel lógico *Logic Level Filter*, *Selective Union* y *Subgraph Extraction*, bajo el supuesto de que se ejecuta un solo operador de alto nivel. Por tanto, el *Selective Union* opera sobre resultados previos de *Logic Level Filter*, el *Logic Level Filter* actúa sobre los grafos de entrada o resultados previos de otros *Logic Level Filter* (subfiltros) y *Subgraph Extraction* actúa sobre los grafos de entrada. Para las condiciones del *Logic Level Filter* se implementaron los operadores de comparación y el operador *inSubGraph*.

7.2.1 Algoritmos

En esta sección se presentan los algoritmos que se implementaron para los operadores *Logic Level Filter*, *Selective Union* y *Subgraph Extraction*.

Logic Level Filter

Con el Algoritmo 1 se implementa el operador *Logic Level Filter* haciendo recorridos de caminos. Dado que el algoritmo se implementó usando el *path traversal framework* de Neo4j y que los resultados que entregan estos *path traversal* son caminos, el resultado

del filtro es un conjunto de caminos asociados a los nodos de la clase de interés que satisfacen la condición especificada para el filtro.

Para cada expresión básica que compone la expresión del filtro se busca el conjunto de nodos instancia de la clase de interés que satisface la condición (R_j) y se calcula un resultado parcial (*Result*) que depende también del resultado obtenido al evaluar las condiciones anteriores. Una expresión básica se evalúa aplicando una de las dos estrategias siguientes: en la primera, se encuentran las instancias del nodo con condición que satisfacen la condición, y se sigue el camino desde esos nodos hasta las instancias de la clase de interés. En la segunda, se toman los nodos las instancias de la clase de interés que cumplieron con las condiciones de las expresiones básicas previas, se recorren los caminos que unen estos nodos con los nodos instancia de la clase con condición y se verifica si estos satisfacen la condición especificada. La primera estrategia se aplica en tres casos: (a) Cuando la expresión del filtro es una disyunción de expresiones básicas, (b) Cuando el operador de la expresión básica es *inSubgraph* (i.e. cuando es un filtro anidado) y (c) Cuando se evalúa la primera expresión básica de una conjunción de expresiones básicas. La segunda estrategia se aplica cuando la expresión es una conjunción de expresiones básicas, la expresión básica que se está evaluando no es la primera y tiene un operador de comparación.

Cuando el operador de la expresión básica es *inSubgraph*, el resultado R_j incluye solamente los caminos entre las instancias de la clase de interés y las instancias del nodo de bifurcación (i.e. la clase de interés del subfiltro). Por tanto, para tener los caminos desde la clase de interés hasta las clases con condición, es necesario agregar los caminos del resultado del subfiltro.

Después de evaluar una expresión básica, se calcula el resultado parcial, *Result*. Cuando se evalúa la primera expresión básica, el resultado parcial es el resultado de la evaluación de la expresión (R_j). A partir de la segunda expresión básica, cuando la expresión de filtro es una disyunción de expresiones básicas, se agrega al resultado parcial el resultado de evaluar la expresión básica (i.e. se hace una unión del resultado parcial anterior y R_j). De otra parte, si la expresión de filtro es una conjunción y el operador de la expresión básica es de comparación, esta expresión se evalúo tomando como inicio los nodos que satisfacían las condiciones anteriores. Entonces, el resultado parcial se obtiene de agregar al resultado de la expresión básica R_j , los caminos que hacían verdaderas las condiciones previamente evaluadas (i.e. de esta manera se encuentra la intersección de las instancias de la clase de interés de R_j y *Result* y se agregan los caminos de *Result* para dichos nodos). Esto mismo ocurre cuando el operador es *inSubgraph*, con la diferencia que antes de agregar los caminos se borra del resultado de evaluar la expresión básica aquellos nodos instancia de la clase de interés que no eran parte del resultado anterior y los caminos asociados a ellos.

Cabe notar que el algoritmo de *Logic Level Filter* no es ingenuo (*naive*), en el sentido de que involucra algunas características que buscan mejorar la eficiencia en la ejecución del operador. Entre ellas, recorrer, en algunos casos, el camino inverso, y

aprovechar los resultados anteriores para obtener de manera directa la intersección de los resultados de las expresiones básicas cuando sobre estas se aplica conjunción.

Algoritmo 1: Logic Level Filter

Datos de entrada

S , un grafo esquema

I , un grafo instancia

m_I , la clase de interés

exp , una expresión de filtro. exp está formada por la conjunción o disyunción de un conjunto de expresiones básicas $bexp_j$ ($1 \leq j \leq k$). Cada $bexp_j$ tiene la forma $m_j \{p_j\} op_j o_j$ donde m_j es una clase con condición, p_j es un camino no dirigido, op_j es un operador de comparación o el operador $inSubgraph$, y o_j un valor literal o el resultado de un filtro ($Result'$).

Datos de salida

$Result$, un conjunto de parejas $\langle n, P \rangle$ donde n es una instancia de m_I y P es un conjunto de caminos de I que hacen que n satisfaga la condición del filtro

Inicio

Para cada $bexp_j$ **haga**

Si ($j == 1$) o ($op_j == inSubgraph$) o (exp es una disyunción de expresiones básicas) **entonces**

/* Evaluar el camino desde m_j hasta m_I *//* Elegir el conjunto de nodos de inicio para el *path traversal*: estos son las instancias del nodo con condición que satisfacen la condición */

Si $op_j == inSubgraph$ **entonces**

$nodos =$ conjunto de nodos de n' de las parejas $\langle n', P' \rangle$ en $Result'$

Sino

$nodos =$ conjunto de nodos de I que son instancia de m_j y cuyo valor v satisface la condición ($v op_j o_j$)

Fin si

Realice un *traversal* en I , que inicie en $nodos$, recorra el **inverso** del camino p_j y retorne un conjunto R_j de parejas $\langle n_j, P_j \rangle$ donde P_j es el conjunto de caminos que corresponden con p_j y terminan en n_j , y n_j es una instancia de m_I

```

Si  $op_j == inSubgraph$  entonces
    /* Se agrega a  $R_j$  los caminos que van desde  $m_j$  hasta
       los nodos con condición del subfiltro */
    Para cada  $P_j$  en  $R_j$  haga
        Agregue a  $P_j$  los caminos asociados a  $n_j$  en  $Result'$ 
    Fin para
    Fin si
    Sino /*  $j > 1$ ,  $op_j <> inSubgraph$  y  $exp$  es una conjunción */
    /* Evaluar el camino desde  $m_I$  hasta  $m_j$ . Dado que  $j > 1$ , hay
       un resultado parcial asignado a  $Result$  */
     $nodos =$  conjunto de nodos de  $n$  de las parejas  $\langle n, P \rangle$ 
    en  $Result$ 
    Realice un traversal en  $I$ , que inicie en  $nodos$ , recorra el
    camino  $p_j$  y retorne un conjunto  $R_j$  de parejas  $\langle n_j, P_j \rangle$ 
    donde  $n_j$  es un nodo de  $nodos$  (es instancia de  $m_I$ ) y
     $P_j$  es un conjunto de caminos que inician en  $n_j$ , corresponden
    con  $p_j$  y terminan en  $n_t$ , y  $n_t$  satisface la condición  $n_t op_j o_j$ 
    Fin si
    /* Consolidar el resultado parcial */
    Si  $j == 1$  entonces  $Result = R_j$ 
    Sino, si  $exp$  es una conjunción entonces
        /* Calcula la intersección de los nodos de la clase de interés entre
           el resultado anterior y  $R_j$ , e incluye los caminos asociados */
    Si  $op_j == inSubgraph$  entonces
        Borre de  $R_j$  las parejas cuyo  $n_j$  no está en alguna pareja
        de  $Result$ 
    Fin si
    Para cada  $\langle n_j, P_j \rangle$  en  $R_j$  haga
        Agregue a  $P_j$  los caminos de  $Result$  asociados con  $n_j$ 
         $Result = R_j$ 
    Fin para
    Sino /*  $exp$  es una disyunción */
        Agregue a  $Result$  las parejas de  $R_j$ 
    Fin si
Fin para
Fin

```

Selective Union

El operador *Selective Union* toma el resultado $Result'$ de un *Logic Level Filter* o un *Path Contraction* y le agrega los arcos de los caminos que corresponden con un conjunto

de caminos del esquema dados. Para ello aplica un recorrido de caminos (*path traversal*) que inicia en los nodos de $Result'$ que son instancia de la clase de interés (esta puede ser diferente de la clase de interés del *Logic Level Filter*) y recorre los caminos dados.

Algoritmo 2: Selective Union

Datos de entrada

S , un grafo esquema

I , un grafo instancia

S' , un grafo esquema

$Result'$, un conjunto de parejas $\langle n', P' \rangle$ resultado de operaciones anteriores

m_I , la clase de interés

P_S , conjunto de caminos simples no dirigidos de S que inician en m_I

Datos de salida

$Result'$, al que se ha agregado los datos de S y I correspondientes a los caminos de P_S

Inicio

/* Busca los nodos instancia de la clase de interés en $Result'$ */

Encontrar $R_{S'}$, el conjunto de arcos de S incidentes a m_I (i.e.

m_I es nodo inicial o final), usando el *GraphDatabaseService*

Encontrar R_I , el conjunto de las relaciones en P' correspondientes con alguna relación en $R_{S'}$

$nodos$ = los nodos instancia de m_I que son nodo inicial (o final) de los arcos en R_I

/* Encuentra los caminos correspondientes a P_S que inician en los nodos */

Para cada p de P_S haga

Realice un *traversal* de I que inicie en los $nodos$, recorra p y

retorne un conjunto P_I de caminos que corresponden con p

Agregue a $Result'$ las parejas $\langle n, P_I \rangle$

Fin para

Elimine de $Result'$ los caminos duplicados

Fin

Subgraph extraction

El operador *Subgraph Extraction* no requiere aplicar recorrido de caminos, se implementa usando las operaciones que provee el *graph database service* de Neo4j. El algoritmo de *Subgraph Extraction* encuentra el conjunto de arcos de la instancia que corresponden con los arcos del esquema dados como parámetro.

Algoritmo 3: Subgraph Extraction

Datos de entrada

S , un grafo esquema

I , un grafo instancia

$Edges$, un conjunto de arcos de S

Datos de salida

$Edges'$, el conjunto de arcos de I correspondientes a los arcos de $Edges$

Inicio

Para cada e de $Edges$ **haga**

 Encontrar E_I , el conjunto de arcos de I correspondientes a e

 Aregar a $Edges'$ los arcos de E_I

Fin para

Fin

7.2.2 Evaluación

Objetivo

La evaluación de la implementación de los operadores de nivel lógico se hizo mediante una prueba que tuvo como objetivo evaluar la factibilidad de la implementación de los operadores y comparar los tiempos de respuesta en la ejecución de las consultas con los obtenidos en consultas formuladas en SPARQL y ejecutadas sobre un motor de tripletas (o un motor RDF).

En la prueba se comparó el tiempo de ejecución de los operadores de GraphTQL, implementados sobre Neo4j, con el tiempo de ejecución de las consultas SPARQL en AllegroGraph. En ambos casos, Neo4j y AllegroGraph, se desarrolló una aplicación en java, que se conecta a la base de datos respectiva y ejecuta una consulta que lee de un archivo.

Hipótesis

Esta prueba tiene como hipótesis que se pueden obtener buenos tiempos de ejecución de los operadores de nivel lógico en una implementación sobre un motor de bases de datos de grafos que se basa en el recorrido de caminos. Se considera que se obtiene un buen tiempo de ejecución si este es similar o menor al tiempo de ejecución de consultas equivalentes expresadas en SPARQL y ejecutadas sobre un motor de tripletas. Esta hipótesis se basa en las bondades que los motores de bases de datos de grafos reclaman con respecto al recorrido de caminos y en las siguientes características de los operadores de GraphTQL: (a) Los grafos esquema de entrada y resultado son grafos conexos. (b) En las consultas se identifican clases de interés, nodos de bifurcación y nodos de conexión, que están conectados por algún camino con los otros nodos involucrados.

(c) La ejecución de los operadores se puede definir con base en resultados parciales obtenidos por la evaluación de los caminos involucrados.

Generación de datos

Para la prueba se usaron datos y consultas del dominio médico. Se desarrolló un generador de datos sintéticos que corresponden con el esquema del ejemplo de referencia de este documento (Figura 4.1). Esta aplicación genera dos tipos de datos, en la primera parte la generación se basa en un conjunto de casos especificados en archivos planos. Los casos especifican síntomas, pruebas e imágenes diagnósticas, procedimientos, medicamentos y especialidades médicas relacionados con algunos diagnósticos. De esta manera se generaron datos sintéticos que guardan alguna coherencia, con el fin de usarlos en las pruebas de usabilidad. En la segunda parte, el generador toma listas de síntomas, pruebas e imágenes diagnósticas, procedimientos, medicamentos y especialidades médicas y los mezcla aleatoriamente para generar diversos encuentros entre pacientes y médicos. De esta forma se generan conjuntos de datos de mayor tamaño, apropiados para la prueba de ejecución.

De otra parte, en las aplicaciones del dominio médico es cada vez mas común el uso de ontologías de dominio o vocabularios estructurados, tales como ICD10¹, LOINC², CPT³ o SNOMED⁴. Por tanto, en los grafos instancia se incluyeron porciones de algunos de estos vocabularios para registrar enfermedades, procedimientos, *test* e imágenes médicas.

Consultas

La prueba de ejecución se hizo con las consultas usadas en la segunda prueba de evaluación y en la prueba comparativa de usabilidad. Dado que la consulta 4 no estaba contemplada en los casos incluidos en la generación de los datos, cada vez que se genera un conjunto de datos de prueba es necesario adecuar los valores literales de las condiciones. Además, se agregó una quinta consulta que se expresa en SPARQL usando la cláusula *union*, cláusula que no se incluyó en las pruebas de usabilidad. La tabla 7.1 presenta cada consulta expresada en lenguaje natural junto con los operadores de GraphTQL y las cláusulas de SPARQL necesarias para su formulación. La expresión de consulta completa en SPARQL y con los operadores de nivel lógico de GraphTQL se presenta en el Anexo I.

¹<http://apps.who.int/classifications/icd10/browse/2015/en>

²<http://loinc.org>

³<http://www.ama-assn.org/go/cpt>

⁴https://www.nlm.nih.gov/research/umls/Snomed/snomed_main.html

Tabla 7.1: Consultas usadas en la prueba de ejecución.

Consulta	Cláusulas SPARQL*	Operadores GraphTQL**
Se requieren las enfermedades (incluyendo su descripción) registradas en la historia familiar del paciente con identificación 723628; incluir la relación familiar si está disponible	1 BP (con 4 trip), 2 OPT (1 trip cada uno), 1 F	1 LLF (con 1 cond.), 2 SU (1 cam. cada uno)
Se requiere encontrar los diagnósticos (incluyendo su descripción) dados al paciente con identificación 723628 por médicos con especialidad en “ENDOCRINOLOGY” y, si está disponible, las fechas de las consultas y las medicinas que le recetaron en ese momento (incluyendo descripción y dosis)	1 BP (con 5 trip), 4 OPT (1,1,3,2 trip. respectivamente), 2 F (conjunción)	2 LLF (un subfiltro conjunción de 2 cond.), 2 SU (1,3 cam. respectivamente)
De los pacientes que han tenido un diagnóstico de “OSTEOMYELITIS” desde el año 2000, se requieren todos los datos de: identificación del paciente, diagnósticos recibidos (incluyendo los diferentes a osteomyelitis), y procedimientos que se le han realizado (incluir la descripción de los diagnósticos y procedimientos). Note que estos datos incluyen eventos diferentes a los del diagnóstico de OSTEO MYELITIS	1 BP (con 4 trip), 4 OPT (1,2,3,3 trip. respectivamente), 2 F (conjunción)	2 LLF (un subfiltro conjunción de 2 cond.), 1 SU (3 cam.)
Se requiere encontrar los pacientes que han tenido un estudio de imágenes diagnósticas con anotación “GALLSTONES” y se les ha realizado el procedimiento “COLECTOMY”. De estos pacientes se requiere la fecha de nacimiento, género y raza, si están disponibles. Estos resultados pudieron realizarse en encuentros diferentes, pero el paciente cumple con ambas condiciones	1 BP (con 6 trip), 3 OPT (2 trip. cada uno), 2 F (conjunción)	1 LLF (conjunción de 2 cond.), 1 SU (3 cam.)
Se requiere encontrar a los pacientes que han tenido una imagen diagnóstica “CAROTID PULSE TRACING WITH ECG LEAD” y a los pacientes a los que se les ha realizado el procedimiento “INSERTION OF PALATAL IMPLANT”. Recuperar además la identificación de estos pacientes, si está disponible.	1 U (1 BP (4 trip) y 1 F, 1 BP (3 trip) y 1 F), 1 OPT (1 trip.)	1 LLF (disyunción de 2 cond.), 1 SU (1 cam.)

* BP=Basic Pattern, OPT=Optional, F=Filter (cada filtro corresponde con una condición),

U=union, trip=tripletas

** LLF=Logic Level Filter, SU=Selective Union, cam=camino, cond=condición

Métricas

Se consideraron dos métricas de tiempo de ejecución: tiempo de ejecución en frío y tiempo de las repeticiones de la ejecución. Se consideró tiempo de ejecución en frío a la ejecución de la primera consulta realizada una vez se inicia el servicio de la base de datos en AllegroGraph, y de la primera consulta que se realiza una vez se inicia la ejecución de la aplicación y la conexión a la base de datos en Neo4j.

La medida de tiempo para la ejecución de consultas en AllegroGraph inicia antes de enviar la consulta al servidor (i.e. después de que se ha leído la cadena de texto del archivo) y termina después de desplegar los resultados en pantalla. Para GraphTQL, la medida inicia antes de abrir el archivo que contiene la consulta y termina también después de desplegar los resultados en pantalla. Esto con el fin de compensar, en alguna medida, el procesamiento de la cadena de texto que realiza AllegroGraph con el procesamiento del archivo de entrada en GraphTQL.

Para ambos casos, en frío y repetición, se tomaron 10 medidas por cada consulta.

Realización de la prueba

Se tomaron dos conjuntos de medidas en equipos diferentes y con diferente tamaño de base de datos, una con un millón de triplets y la segunda con cinco millones. Estos son conjuntos de datos pequeños pero sirven para dar un indicio respecto al desempeño de la ejecución de las consultas en las dos opciones de implementación. El límite de 5 millones se tomó porque la versión de uso libre de AllegroGraph soporta esa cantidad de triplets.

Un conjunto de medidas se tomó ejecutando las consultas sobre una base de datos de un millón de triplets, en una máquina virtual con sistema operativo Ubuntu 14.04LTS de 64bits, con 1024MB de memoria RAM, ejecutándose sobre un computador con procesador Intel Core I3-4010Y con 4G de memoria RAM y sistema operativo Windows 8 de 64 bits. AllegroGraph versión 4.14.1, Java JRE 1.7 y Neo4j versión 2.1.3. Como herramienta de virtualización se usó VirtualBox versión 4.3.28.

El segundo conjunto de medidas se tomó ejecutando las consultas sobre una base de datos de cinco millones de triplets, en un *cluster* compuesto de un *front-end* con 32 cores (procesador AMD) de 2.3 Ghz, 32G de RAM y Distribución Ubuntu Server 14.04; y cuatro nodos de trabajo, cada uno con 64 cores (AMD) de 2.3 GHz, 64G de RAM y Ubuntu Server 14.04. Ellos están interconectados por medio de ethernet, con 10Gbit para comunicación de procesamiento y 1Gbit en conexión de administración y sistema de archivos. Con AllegroGraph versión 5.1, Java JRE 1.8 y Neo4j versión 2.1.3.

Para AllegroGraph la ejecución en frío se realizó bajando el servidor de base de datos y reiniciándolo para ejecutar la consulta. Para Neo4j se cerraba la conexión y la aplicación, y se reiniciaba la aplicación. Además, para tomar los tiempos de ejecución en frío se intercaló la ejecución de una consulta en AllegroGraph y una en GraphTQL.

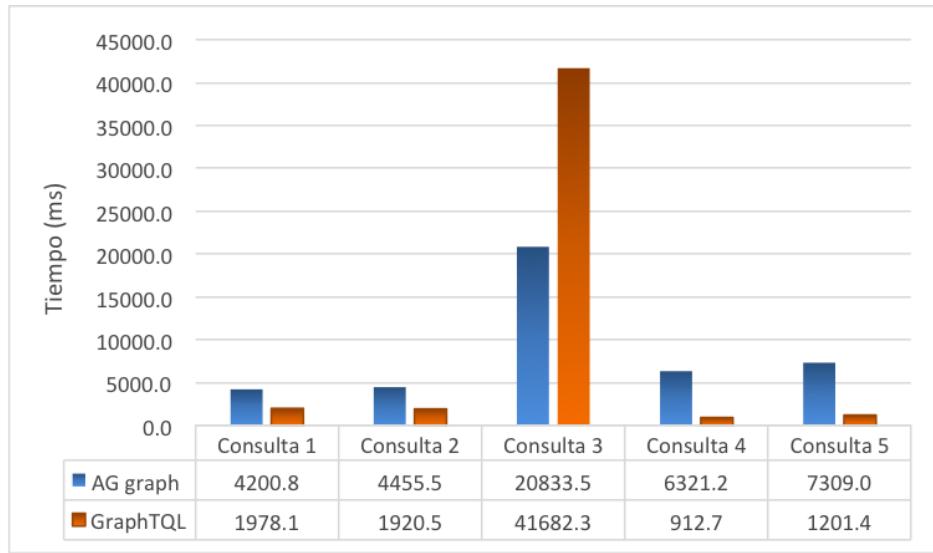
Resultados

Las tablas 7.2 y 7.3 muestran el tamaño de la respuesta de cada consulta (número de tripletas) y los tiempos de ejecución (milisegundos) en la máquina virtual y el *cluster*. Las Figuras 7.2a y 7.2b muestran el promedio de las ejecuciones en frío, mientras que las Figuras 7.3a y 7.3b muestran el promedio de las repeticiones.

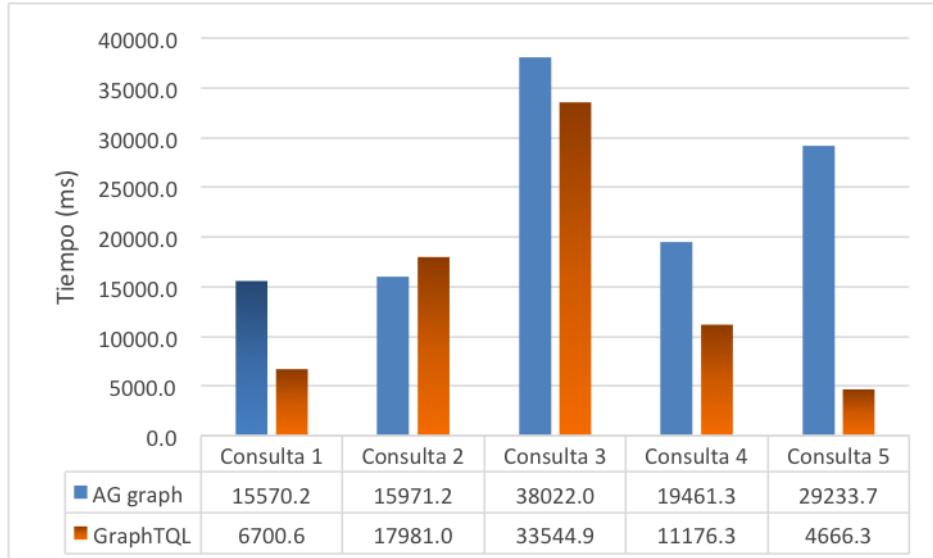
Los resultados tienen tendencias similares a pesar de las diferencias en la capacidad de los equipos de cómputo y en los tamaños de las bases de datos de prueba y de los conjuntos de tripletas resultado de cada consulta.

Los resultados son interesantes teniendo en cuenta que en la implementación de GraphTQL hay una parte del procesamiento que no tiene optimización de la consulta. Esta es la parte de los algoritmos que corresponde al manejo de los datos que arroja el *path traversal framework*. Algunos de los mecanismos de optimización que se podrían estudiar para esta parte del procesamiento son, por ejemplo: (a) Usar estadísticas de los datos para cambiar el orden de evaluación de las expresiones básicas, aplicando primero aquellas que pueden ser más selectivas. (b) Modificar el orden de la ejecución de los operadores y tener algoritmos de ejecución de los subfiltros que puedan aprovechar resultados intermedios del filtro padre. (c) Implementar un manejo de memoria caché para los resultados intermedios de los operadores.

Los resultados muestran que el 80% de las medidas de tiempo de ejecución son menores para GraphTQL que para AllegroGraph. En el 20% restante no se identifica un elemento particular que genere la diferencia en el resultado, por ejemplo, estos resultados corresponden a consultas diferentes (consulta 1, 2 y 3), el resultado se presenta en las medidas tomadas en frío y en las repeticiones, y también se presenta en los dos ambientes de prueba.

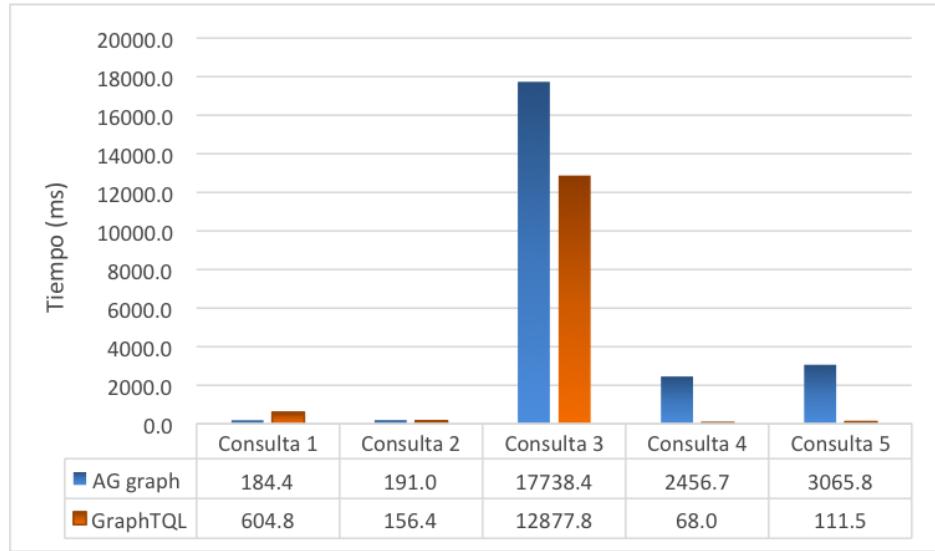


(a) Ejecución en el *Cluster*

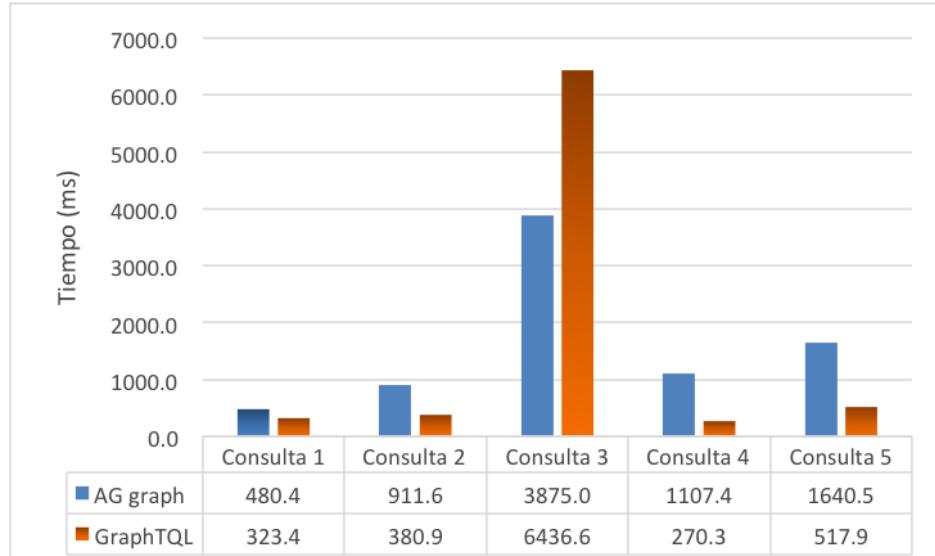


(b) Ejecución en la Máquina Virtual

Figura 7.2: Tiempos promedio de ejecución en frío.



(a) Ejecución en el *Cluster*



(b) Ejecución en la Máquina Virtual

Figura 7.3: Tiempos promedio de ejecución de las repeticiones.

Tabla 7.2: Resultados de la ejecución en frío.

Máquina	Tamaño BD	Consulta	Tamaño resultado	Motor	Medidas (tiempo ms)						Promedio (ms)				
					1	2	3	4	5	6	7				
Cluster	48 tripletas	Consulta 1	48 tripletas	AG graph	4708	4366	4351	3682	4370	3728	4394	4317	4367	3725	4200.8
		Consulta 2	30 tripletas	GraphnQL	1704	2080	2065	1755	1783	2150	1953	2146	2057	2088	1978.1
	4.999.993 tripletas	Consulta 3	13.763 tripletas	AG graph	5173	4481	4520	3920	4434	4505	3900	4649	4459	4514	4455.5
		Consulta 4	100 tripletas	GraphnQL	1648	1804	2329	1536	1752	1720	2273	1959	2298	1886	1920.5
		Consulta 5	1615 tripletas	AG graph	22484	17410	24963	20453	21086	18916	17504	24787	19600	21132	20833.5
Máq. Virtual	39 tripletas	Consulta 1	39 tripletas	GraphnQL	44487	41960	36095	41635	41543	44773	36720	41298	46392	41920	41682.3
		Consulta 2	28 tripletas	AG graph	6011	6050	5730	6796	6861	6175	5423	6727	6681	6758	6321.2
	1.000.139 tripletas	Consulta 3	2190 tripletas	GraphnQL	927	910	921	944	913	793	963	901	931	924	912.7
		Consulta 4	15 tripletas	AG graph	7779	6913	7604	7036	7352	7664	7669	5802	7680	7591	7309.0
		Consulta 5	46722	AG graph	14247	16213	17558	16315	13961	15004	15068	14734	17416	15186	15570.2
				GraphnQL	6312	6173	6182	6480	6264	6853	6993	7172	7625	6952	6700.6

Tabla 7.3: Resultados de la repetición de las ejecuciones.

Máquina	Tamaño de la BD	Consulta	Tamaño resultado	Motor	Repeticiones (tiempo ms)									Promedio (ms)		
					1	2	3	4	5	6	7	8	9			
Cluster 4.999.993 tripletas	Consulta 1 48 tripletas	AG graph	178	186	169	172	208	173	196	175	212	184.4		184.4		
					GraphTQL	317	398	379	458	549	657	684	786	875	945	604.8
	Consulta 2 30 tripletas	AG graph	205	182	184	194	184	198	182	184	191	191	191	191	191	191
	GraphTQL				385	162	140	127	122	120	119	116	118	155	156.4	
	Consulta 3 13.763 tripletas	AG graph	20680	13971	20485	20220	20175	17465	14875	16250	18943	14320	17738	17738	17738.4	
	GraphTQL				16202	13556	12901	12694	12651	11553	12063	12348	12389	12421	12877.8	
Máq. Virtual Ubuntu	Consulta 4 100 tripletas	AG graph	2477	2486	2429	2498	2436	2469	2445	2419	2447	2461	2456.7		2456.7	
	GraphTQL				121	96	59	84	79	75	69	34	33	30	68	
	Consulta 5 1615 tripletas	AG graph	3374	2001	3369	3401	3325	3406	3325	3329	1823	3305	3065.8		3065.8	
	GraphTQL				151	141	125	114	119	100	94	98	87	86	111.5	
	Consulta 1 39 tripletas	AG graph	1019	664	442	333	247	244	846	515	250	244	244	244	240.4	
	GraphTQL				661	628	398	319	280	251	241	155	143	158	323.4	
Máq. Virtual Ubuntu	Consulta 2 28 tripletas	AG graph	4041	1530	413	629	287	374	272	338	968	264	911.6		911.6	
	GraphTQL				789	729	507	425	407	337	196	165	112	142	380.9	
	Consulta 3 2190 tripletas	AG graph	8280	4152	3718	3761	3442	3378	3109	2878	2908	3124	3875.0		3875.0	
	GraphTQL				7880	7186	5743	6380	6849	6714	6597	5320	6515	5182	6436.6	
	Consulta 4 15 tripletas	AG graph	5845	869	426	937	319	436	418	314	428	1082	1107.4		1107.4	
	GraphTQL				353	350	236	269	231	283	280	275	230	196	270.3	
	Consulta 5 332 tripletas	AG graph	56555	1352	1060	1511	1283	1242	1075	1152	991	1084	1640.5		1640.5	
	GraphTQL				647	498	794	228	406	561	552	555	502	436	517.9	

7.3 Arquitectura de Integración de Datos

El uso de un modelo de datos basado en grafos como base de GraphTQL puede facilitar el uso del lenguaje como interfaz de acceso a los datos de fuentes de datos heterogéneas a través de una arquitectura de integración. En esta sección se presentan algunas ideas con las que se explora esta posibilidad.

La integración de datos [115] busca proveer a los usuarios de una vista unificada, integrada y global de datos que se encuentran en fuentes diversas, heterogéneas e independientes. Entre los diferentes enfoques y arquitecturas existentes para resolver el problema de integración de datos, ésta propuesta usa la traducción de consultas. La integración en el enfoque de traducción de consultas es virtual, de forma tal que las consultas se procesan en cada una de las fuentes de datos. Comúnmente, este enfoque usa una arquitectura basada en mediación [196] que incluye la interfaz de usuario, un mediador, *wrappers* y las fuentes de datos. La interfaz de usuario captura la consulta y la entrega al mediador. El mediador se encarga del procesamiento global de la consulta, que se realiza con base en un modelo de datos global y los mapeos entre este y los esquemas de datos locales de las fuentes que integra. En el procesamiento global el mediador divide la consulta y genera subconsultas que posteriormente se procesan en las respectivas fuentes de datos. El lenguaje o la representación de las subconsultas en el mediador puede ser diferente del lenguaje o de la representación usada para acceder a las fuentes de datos. Cuando el lenguaje de las subconsultas es diferente al lenguaje de consulta de la fuente de datos, se necesita un *wrapper* que haga la traducción entre los dos lenguajes. Finalmente, después de que las subconsultas se ejecutan en cada fuente de datos, el mediador recibe los resultados, los integra y los entrega al usuario. La Figura 7.4, muestra una arquitectura genérica de integración basada en mediación, adaptada de [193].

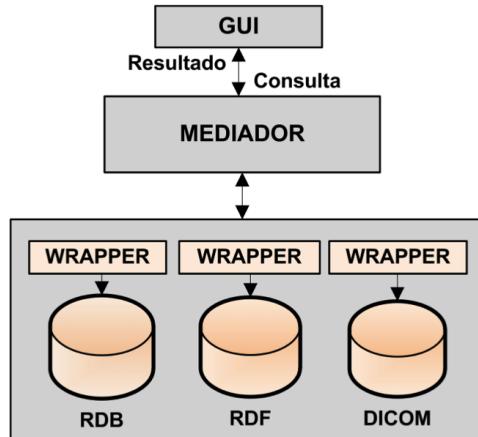


Figura 7.4: Arquitectura de integración

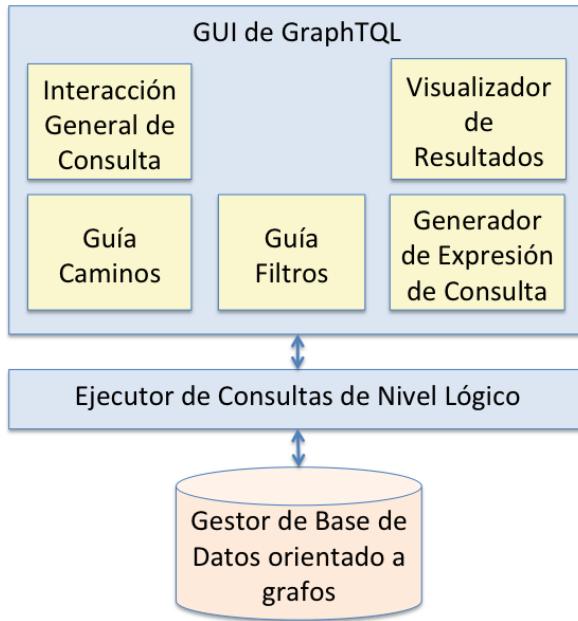


Figura 7.5: Arquitectura de integración basada en mediación y modelos de grafos

A continuación, se describe de manera general la arquitectura propuesta. La Figura 7.5 muestra los componentes de la arquitectura, los cuales corresponden con los comúnmente utilizados en mediación: la interfaz de usuario (el GUI de GraphTQL), el mediador, los *wrappers* y las fuentes de datos. Un componente diferenciador incluye un conjunto de metadatos asociados a los *wrappers* y a las fuentes de datos. La incorporación de los metadatos responde a una característica que se presenta con frecuencia en aplicaciones en el dominio médico, en las cuales es común encontrar datos no estructurados. Un ejemplo de ello son las imágenes médicas y los campos de texto usados en reportes e historias clínicas. En estos últimos, el médico hace una descripción libre de la información relevante, como en el caso de la descripción del estado del paciente en el momento de la consulta. Para manejar la ambigüedad o la falta de semántica que se presenta en los datos no estructurados, la arquitectura provee la posibilidad de asociar metadatos. Estos metadatos son descriptores semánticos (e.g. conceptos de ontologías de dominio). Los metadatos permiten limitar el vocabulario de las consultas, ya que se restringen los términos cuando se hace referencia a los datos de atributos que tienen asociados metadatos.

El mediador

El mediador incluye un modelo global de datos que representa la información que se encuentra en las fuentes que integra, mapeos GAV (*Global as a View*) entre el modelo de datos global y los modelos de datos locales, y un módulo de procesamiento de

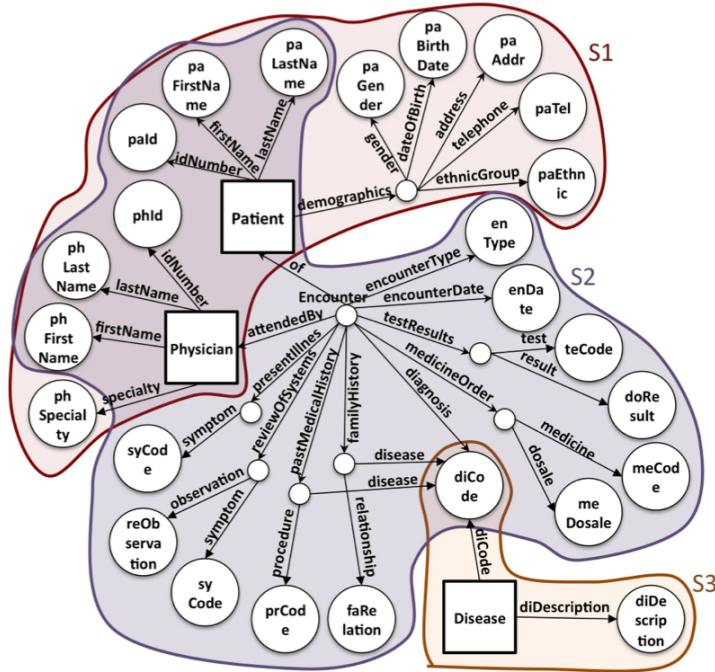


Figura 7.6: Modelo global

consultas. Los mapeos GAV asocian los elementos del modelo global con consultas expresadas sobre los modelos de las fuentes, a diferencia de los mapeos LAV (*Local as a View*) que asocian los elementos de los modelos locales (de cada fuente) con consultas expresadas sobre el modelo global [115].

El modelo global de la arquitectura está definido con un grafo esquema GDM. Conceptualmente, este grafo está formado por un conjunto de subgrafos que se intersectan. Cada subgrafo representa la información almacenada en una fuente de datos. La Figura 7.6 muestra un modelo global donde se integra información clínica, el subgrafo de cada fuente está delimitado por una región de color.

El funcionamiento del mediador supone que toda clase objeto o de valor compuesto tiene uno o varios atributos que identifican cada una de sus entidades de manera única. Además, cuando los subgrafos de dos fuentes se intersectan, la integración de los datos provenientes de las mismas se realiza unificando—uniendo en un solo nodo—los nodos de los atributos que identifican las clases objeto o de valor compuesto que las fuentes tienen en común.

Cada fuente de datos se representa con un **grafo local**, definido como un grafo esquema GDM en el que solo se tienen nodos de clase objeto y nodos de clase valor básico. Este grafo local no está almacenado en ninguno de los componentes de la arquitectura, pero es esencial para definir los **mapeos** entre el modelo global y las fuentes de datos. De este modo, el mediador tiene una representación uniforme de

las fuentes de datos y el mapeo del modelo global al grafo local se hace con una representación de grafos. El mapeo del grafo local al modelo de la fuente es 1 a 1, es decir un elemento del grafo local corresponde con un elemento del modelo de la fuente de datos. De esta manera, se simplifica la re-escritura de la consulta en el *wrapper*.

Si los datos correspondientes a un arco e del grafo global los puede suministrar la fuente F , el arco se mapea con un patrón definido sobre el grafo local de F . El patrón se define con un conjunto de arcos—o tripletas. Un mapeo se define con una expresión de la forma $(n, a, m) : F_1[t_1 \dots t_i], \dots F_k[t_j \dots t_l]$ donde (n, a, m) es un arco del grafo global, $F_1 \dots F_k$ son las fuentes de datos, y t_1, t_2, \dots, t_l son arcos que forman los patrones que definen cuáles datos recuperar en cada fuente.

Adicionalmente, para cada clase objeto o de valor compuesto se define un conjunto de mapeos de integración. Estos mapeos son patrones que permiten recuperar, de cada fuente, los atributos que identifican las entidades de la clase. Un mapeo de integración es una expresión de la forma $C : F_1[t_1 \dots t_i], \dots, F_k[t_j \dots t_l]$, donde C es una clase objeto o de valor compuesto, $F_1 \dots F_k$ son las fuentes de datos, t_1, t_2, \dots, t_l son arcos que definen los patrones en las fuentes.

A partir del modelo global y los mapeos, el mediador identifica las fuentes relevantes para la ejecución de la consulta y divide la consulta en subconsultas que cada fuente puede responder.

Los *wrappers*

El *wrapper* es el componente de *software* que encapsula el acceso a la fuente de datos. El *wrapper* recibe una consulta del mediador y la convierte en una consulta que la fuente de datos es capaz de procesar. Cada *wrapper* se encarga de tres aspectos: el primero, identificar y procesar las consultas que incluyen una tripla o una condición de filtro con un atributo que tiene anotaciones; el segundo, hacer la traducción de la consulta al lenguaje o mecanismo de acceso propio de la fuente de datos; y el tercero, integrar los resultados recuperados del repositorio de anotaciones con los recuperados de la fuente de datos. Estos procesos se definen en cada *wrapper* según las características de la fuente de datos a la que accede. Para ello se usan los metadatos, el modelo de datos lógico de la fuente y el grafo local de la fuente. Los mapeos entre estos modelos son sencillos, uno a uno (i.e. un elemento de un modelo se mapea con un elemento del otro). La Sección 7.3.1 ilustra cómo se pueden definir estos mapeos.

Las subconsultas generadas por el mediador se expresan con una sintaxis similar a la de un *Select Query* de SPARQL, sin embargo, el *SELECT* incluye solamente la lista de variables a retornar y el *WHERE*, un grupo de patrones de triples que, además de las triples, pueden incluir solamente las cláusulas *FILTER* y *OPTIONAL*. La siguiente consulta muestra un ejemplo de esta sintaxis:

```
SELECT ?paID ?lastName  
WHERE { ?Patient id ?id .
```

OPTIONAL { ?Patient lastName ?lastName } }

Implementación de la arquitectura

Como prueba de concepto, se hizo un estudio preliminar sobre cómo realizar la recuperación de los datos necesarios para ejecutar, sobre esta arquitectura, una consulta expresada con los operadores de nivel lógico de GraphTQL. Para ello y como parte de un trabajo de grado [131] de un estudiante de pregrado de la Universidad del Valle, se implementó la generación de subconsultas para los operadores *Subgraph Extraction* y *Logic Level Filter*.

Durante el proceso de generación de subconsultas se distinguen en el modelo global dos tipos de arco, los que conectan una clase objeto o de valor compuesto etiquetada con una clase de valor básico (arcos tipo 1), y las demás combinaciones de tipos de nodo posibles (arco tipo 2). Esta diferencia se usa para reconocer los objetos cuyos atributos de identificación es necesario recuperar ya que se pueden requerir para realizar la integración de los datos del mismo objeto provenientes de diversas fuentes.

El operador *Subgraph Extraction* recupera los nodos y arcos de la instancia de datos que corresponden con un conjunto de arcos del esquema dados. Cada uno de estos arcos tiene definido un conjunto de mapeos que permiten identificar los patrones de consulta para extraer los datos correspondientes, que están almacenados en una o varias fuentes de datos. El mediador genera una consulta por cada fuente y clase objeto o de valor compuesto C que aparece en arcos tipo 1 del subgrafo. El patrón de esta consulta inicia con el mapeo de integración definido para la clase C . Los mapeos de integración son aquellos con los que se recuperan los atributos de identificación de la clase y forman el patrón básico de la consulta. A ese patrón se agregan, con *OPTIONAL*, los mapeos definidos para cada arco tipo 1 que incluye a la clase C . Adicionalmente, se genera una consulta por cada arco tipo 2 del subgrafo. En el patrón de esta consulta se unen los mapeos de integración de las clases que el arco conecta y el mapeo definido para este arco en particular. Las consultas que generan los arcos tipo 2 tienen el objetivo de recuperar los datos necesarios para integrar los resultados de las subconsultas. Finalmente, cada consulta se forma con una cláusula *SELECT*, cuya lista de variables incluye los nodos de clase de valor básico que están en el patrón de consulta, y una cláusula *WHERE* con el patrón de consulta que se construyó. El siguiente ejemplo muestra la subconsulta generada para la operación

Extract($\langle S, I \rangle, \{(Patient, id, id), (Patient, lastName, lastName)\})$

la subconsulta correspondiente es,

```
SELECT ?id ?lastName  
WHERE { ?Patient id ?id .  
        OPTIONAL { ?Patient lastName ?lastName } }
```

Por su parte, *Logic Level Filter* recupera un conjunto M de nodos instancia de la clase de interés y un conjunto P de caminos que inician con algún nodo de M

y conectan ese nodo con otros que satisfacen las condiciones del filtro. Para cada camino de la expresión de filtro, se generan las consultas para las fuentes que aportan los datos correspondientes. Por cada fuente y clase objeto o de valor compuesto C que aparece en arcos tipo 1 del camino se genera una consulta. El patrón de esa consulta inicia con el mapeo de integración definido para la clase C , a éste se unen los mapeos definidos para cada arco tipo 1 que incluye a la clase C . En este caso no se usa la cláusula *OPTIONAL* porque a diferencia del comportamiento del subgrafo, se requiere encontrar conjuntos de datos que correspondan con el patrón completo definido en cada camino. Para cada arco tipo 2 se genera una consulta, en cuyo patrón se unen los mapeos de integración de las clases que el arco conecta y el mapeo definido para este arco en particular. Finalmente, cada consulta se forma con una cláusula *SELECT*, cuya lista de variables incluye los nodos de clase de valor básico que están en el patrón de consulta, y una cláusula *WHERE* con el patrón de consulta que se construyó. El siguiente ejemplo muestra la subconsulta generada para la operación $\text{LogicLFilter}(\langle S, I \rangle, \text{Diagnosis}, (\text{description}_{\{\text{Diagnosis}, \text{description}, \text{description}\}} = \text{DIABETES}))$ la subconsulta correspondiente es,

```
SELECT ?description ?code
WHERE { ?Diagnosis code ?code . ?Diagnosis description ?description .
      FILTER (?description = DIABETES) } .
```

En este caso, $?Diagnosis code ?code$ se agrega por ser el mapeo de integración de *Diagnosis*.

7.3.1 Mapeos entre el Modelo Relacional y GDM

El acceso a datos almacenados en fuentes de datos con diversos modelos requiere la definición de mapeos entre el modelo de datos representado en GDM y el modelo de datos de la fuente. En particular, en la arquitectura planteada, cada *wrapper* requiere de la definición de mapeos entre el grafo local y el modelo de datos de la fuente. En esta sección se describe cómo se pueden definir estos mapeos para fuentes de datos con modelo relacional.

Adaptando la recomendación de la W3C para mapeos del modelo relacional a RDF [10], los mapeos de un modelo relacional a GDM se pueden definir como sigue:

- Mapeo del esquema:
 - Para cada relación, el nombre de la relación es la etiqueta de un nodo objeto del grafo esquema
 - Los atributos de la relación son también atributos en el grafo, es decir son etiquetas de un arco que une el nodo objeto etiquetado con el nombre de la relación con un nodo de valor básico. Este último, está etiquetado con el nombre del atributo y su dominio es el dominio del atributo en el modelo relacional.
 - Por cada llave foránea en una relación R se crea un arco entre el nodo objeto que representan la relación R y el nodo objeto que representa la relación a la

Person (ID:string, firstName:str, lastName:string, address:string)

Telephone (ID:string, number:integer, FOREIGN KEY ID
REFERENCES Person.ID)

Patient (ID:string, gender:string, birthDate:date, ethnicGroup:string,
FOREIGN KEY ID REFERENCES Person.ID)

Physician (ID:string, code:integer, FOREIGN KEY phID
REFERENCES Person.ID, FOREIGN KEY code REFERENCES
Specialty.code)

Specialty (code:integer, description:string)

Figura 7.7: Modelo relacional de la fuente S1

que hace referencia la llave foránea.

■ Mapeo de los datos:

- Por cada tupla de una relación hay un nodo objeto en la instancia y por cada atributo no nulo que no es llave foránea se crea un nodo de valor básico. El nodo objeto es instancia del nodo objeto correspondiente en el esquema, al igual que el nodo de valor básico es instancia del nodo de valor básico correspondiente. Adicionalmente, hay un arco que une el nodo objeto con cada uno de los nodos de valor básico creados, cuya etiqueta es el nombre del atributo correspondiente.
- Por cada llave foránea de una tupla de una relación R hay un arco que conecta los dos nodos objeto que representan la tupla de la relación R y la tupla a la que hace referencia la llave foránea.

La re-escritura, en SQL, de las consultas expresadas con patrones se realiza de la siguiente manera: los nodos de valor básico hacen parte de la cláusula *select*, los nodos objeto forman la cláusula *from* y los filtros generan la cláusula *where*. Cuando el patrón de la consulta involucra un arco entre nodos objeto, se requiere un *join* (o *outer join* en el caso de la cláusula *optional*) entre las relaciones involucradas.

Por ejemplo, si la fuente $S1$ del grafo global de la Figura 7.6 (datos de *Patient* y *Physician*) organiza los datos con el conjunto de relaciones que se muestra en la Figura 7.7, el grafo local para $S1$ es el que aparece en la Figura 7.8. Luego, si se requiere recuperar el patrón $\{?Patient\ first\Name\ ?first\Name\}$ especificado sobre el grafo global (Figura 7.6), este patrón re-escrito en términos del grafo local de la fuente $S1$ es: $\{?Patient\ Patient-Person\ ?Person\ .\ ?Person\ first\Name\ ?first\Name\}$ y la sentencia SQL correspondiente:

```
SELECT first\Name
FROM Patient JOIN Person ON (Patient.ID = Person.ID)
```

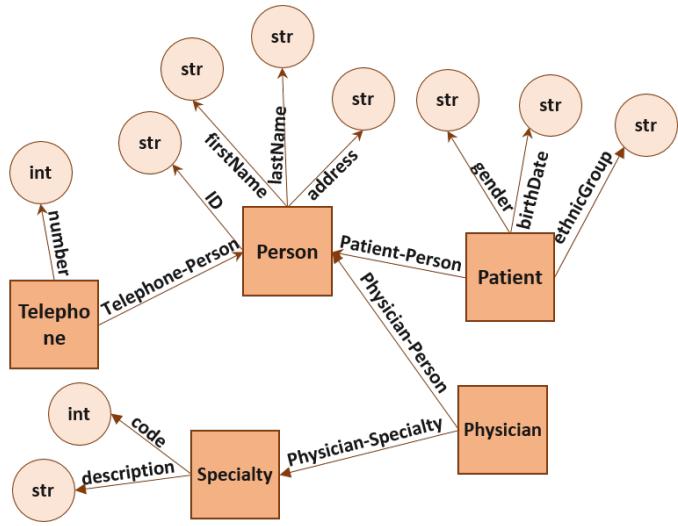


Figura 7.8: Grafo local de la fuente S1

7.4 Síntesis

En este capítulo se describió la estructura funcional del sistema y cómo esta se implementó para generar un prototipo funcional de GraphTQL. Se describió también la implementación de los operadores de nivel lógico de GraphTQL usando recorridos de caminos y se mostraron los resultados en tiempo de ejecución de esta implementación, comparados con consultas escritas en SPARQL y ejecutadas sobre un motor de grafos basado en triplets. Los resultados de estas pruebas muestran que la implementación usando recorridos de caminos puede ser la mejor opción para los operadores de GraphTQL.

Además se presentó una arquitectura para integración de datos de fuentes heterogéneas en la que GraphTQL se ubica en el nivel superior de interacción con el usuario final. La experiencia obtenida con el diseño e implementación de esta arquitectura permiten concluir que, para efectos de integración de datos de fuentes heterogéneas, la vía más adecuada podría ser generar consultas SPARQL y usar un motor de bases de datos federadas basado en SPARQL *end points* que haga la función de mediador para acceder a bases de datos con diversos modelos de datos. Esto debido a que las investigaciones actuales en esta área incorporan en dichos motores optimización del proceso de distribución de la consulta basado en estadísticas de los datos que se tienen en cada fuente.

Capítulo 8

Conclusiones y Trabajo Futuro

Este estudio explora el uso de un modelo de grafos, que diferencia los grafos esquema e instancia, en el desarrollo de un lenguaje visual de consulta orientado a usuarios finales. Este lenguaje facilita la formulación consultas *ad hoc* (no conocidas con anticipación). Las consultas pueden ser complejas, en el sentido que pueden incluir patrones de filtro formados por una porción del grafo con múltiples nodos y arcos. El usuario puede especificar condiciones de filtro sobre uno o varios nodos y, de manera guiada, establecer combinaciones que generan expresiones de conjunción y disyunción de las condiciones.

En el estudio se revisan los beneficios de uso del grafo esquema como modelo conceptual, cuya representación gráfica soporta la interacción con el usuario final. Para esto se propone un conjunto de operadores que transforman los grafos esquema e instancia reduciéndolos hasta obtener el conjunto de datos que el usuario requiere como resultado de su consulta. También se explora el uso del proceso de diseño centrado en el usuario, enfocado en satisfacer necesidades de usuarios en el dominio médico, particularmente en lo relacionado con el acceso a datos de las historias clínicas de los pacientes.

En este capítulo se resumen las principales contribuciones y las conclusiones derivadas de este trabajo en las Secciones 8.1 y 8.2. En la Sección 8.3 se proponen algunas perspectivas de trabajo futuro.

8.1 Contribuciones

Las principales contribuciones de este trabajo son:

Un Lenguaje Visual de Consulta (GraphTQL) basado en la transformación de los grafos esquema e instancia. El lenguaje usa el diagrama del grafo esquema como base del mecanismo de interacción directa, por lo cual el usuario no necesita explorar los datos para conocer su estructura y formular la consulta. El grafo esquema, que se despliega, es una vista simplificada de un modelo conceptual de los datos. La característica de simplificada hace referencia a que solamente representa la información necesaria para que el usuario formule la consulta. Por ejemplo, esta vista no incluye la cardinalidad de las relaciones, no distingue atributos multivaluados u obligatorios

y no especifica la dirección de los arcos. La formulación de la consulta se basa en aplicar sucesivas transformaciones sobre las vistas de los grafos esquema e instancia, para reducir progresivamente el conjunto de datos, hasta obtener el subconjunto que el usuario requiere recuperar en la consulta. Estas transformaciones se efectúan por medio de la aplicación sucesiva de un conjunto de operaciones.

El lenguaje fue desarrollado siguiendo un enfoque de diseño centrado en el usuario y tomó como dominio de aplicación el dominio médico. Para ello se realizaron cinco pruebas de usabilidad, en distintas etapas del desarrollo del proyecto. La primera, una entrevista preliminar, para conocer las necesidades de los usuarios potenciales. La segunda, una prueba exploratoria, para validar la decisión del uso de grafos como modelo de datos y las primeras versiones de la definición de los operadores. La tercera, para evaluar la organización de la interfaz, así como los iconos y colores usados en ella. En la cuarta se evaluó un prototipo funcional mediante la formulación de cuatro consultas. Y finalmente, en la quinta se hizo prueba comparativa del prototipo funcional de GraphQL y una herramienta gráfica para formulación de consultas en SPARQL.

Un conjunto de operadores, formalmente definidos, que soportan el lenguaje visual de consulta. Estos operadores permiten a los usuarios manipular los objetos manteniendo sus relaciones y atributos, derivar nuevas relaciones entre objetos que están conectados por caminos no dirigidos y definir filtros complejos sobre objetos de interés. Los operadores soportan el mecanismo de reducción de los grafos esquema e instancia, incluyen el manejo de datos incompletos de forma implícita, toman como parámetros los nodos marcados en el grafo esquema, generan los diálogos de clarificación de filtros para guiar al usuario en la formulación de consultas, manejan caminos no dirigidos y ocultan algunos conceptos propios de los lenguajes de consulta como la definición de variables, la selección de variables de salida y operaciones de *join*.

Un mecanismo para desambiguar los filtros basado en diálogos de clarificación. El Diálogo de Clarificación de Filtros tiene como objetivo precisar la semántica de las condiciones de filtro que el usuario marca sobre el grafo esquema. De esta manera se ofrece mayor expresividad, sin agregar conceptos o notación gráfica adicional que el usuario deba aprender a usar, y se tiene una representación gráfica simple de las entradas requeridas para los operadores de filtro. Para ello se identifican los nodos de bifurcación, es decir aquellos donde se bifurcan los caminos que conectan la clase de interés con las clases de condición. El diálogo se lleva a cabo mediante una serie de preguntas que tienen en cuenta, en cada nodo de bifurcación, posibles opciones de ejecución para la condición.

Un prototipo funcional de la interfaz gráfica del lenguaje de consulta, que usa el grafo esquema para ofrecer a los usuarios una vista general de los datos y para guiarlos en la exploración de los datos, evitando construir patrones de consulta desde cero. El diagrama del grafo esquema usa diferentes formas y colores para diferenciar los tipos de nodo y ofrece la posibilidad de tener iconos sobre los nodos objeto. Mediante la interfaz gráfica: una consulta se construye incrementalmente por aplicación sucesiva de

operaciones; se brinda realimentación al usuario cada vez que aplica una operación; se facilita la selección de los arcos involucrados en cada operación aplicando algoritmos de búsqueda de caminos para señalar los caminos posibles entre los nodos que el usuario selecciona; y se implementa el mecanismo de desambiguación de filtros. La interfaz es simple, en el sentido de que tiene pocos componentes, que el usuario debe decidir entre un conjunto pequeño de acciones y que requiere aprender pocos controles (i.e. para la interacción solo se requiere usar clic izquierdo y derecho)

Una implementación de la ejecución los operadores de nivel lógico. Los operadores de nivel lógico se implementaron usando un *framework* basado en recorridos de caminos (*path traversals*) sobre un motor de bases de datos de grafos. Los tiempos de ejecución de consultas en esta implementación se compararon con los tiempos de ejecución de consultas equivalentes en SPARQL. En la mayoría de los casos evaluados, la implementación con recorridos de caminos arrojó mejores tiempos de ejecución.

8.2 Conclusiones

El desarrollo del lenguaje visual de consulta propuesto, GraphTQL, tuvo como objetivo facilitar a los usuarios finales la formulación de consultas de mediana complejidad. En el Capítulo 4 se describe el lenguaje y en la Sección 4.5, sus principales características del diseño. Entre estas últimas se destacan el uso del grafo esquema como modelo de datos conceptual y como base del mecanismo de interacción basado en la manipulación directa, la estrategia de reducción de los grafos esquema e instancia mediante la aplicación de los operadores del lenguaje, el liberar al usuario de la necesidad de usar operaciones adicionales para el manejo explícito de datos incompletos y el guiarlo en la formulación de filtros complejos por medio de diálogos de clarificación. Estas características son resultado del enfoque de diseño centrado en el usuario (DCU) que tuvo como dominio de aplicación de referencia el dominio médico. La interacción con los usuarios de este dominio permitió conocer las necesidades de consulta e incluir en el diseño de GraphTQL cambios derivados de los resultados de las pruebas de usabilidad.

El enfoque DCU tuvo una fuerte repercusión en el diseño de GraphTQL ya que llevó a tener cuenta el impacto de cada decisión de diseño del lenguaje en su facilidad de uso. Para ello, durante el diseño y desarrollo, se aplicó la estrategia de reducir la carga cognitiva para el usuario, en el sentido de reducir el número de conceptos que debe aprender y los factores que debe tener en cuenta cuando razona sobre cómo obtener los datos que requiere.

La prueba comparativa (Sección 6.4) mostró que los participantes logran un mejor desempeño formulando consultas en GraphTQL, en comparación con la formulación de las mismas en una herramienta gráfica para SPARQL. Este desempeño se refleja en el número de consultas formuladas correctamente, el número de errores que se presentan en la formulación de las consultas, el tiempo requerido para formularlas y el tipo de ayuda que los participantes necesitaron. También se comprobó que en la formulación de las consultas con GraphTQL se redujeron errores previstos como la especificación de arcos con dirección incorrecta o de caminos no existentes, la falta de manejo de datos incompletos (i.e. con la cláusula *optional*),

el olvidar incluir datos requeridos, las dificultades para especificar filtros complejos o para explorar los datos, especialmente cuando se tienen datos incompletos. Es de notar que también se redujeron errores que no se habían previsto, por ejemplo, confundir el nombre de la variable con la condición de filtro o ubicar la condición sobre un nodo que representa un objeto y no sobre uno que representa un valor.

La prueba permitió también hacer evidente que los usuarios crean estrategias para optimizar su trabajo. Esto se evidenció en el grupo de participantes que evaluaron la interfaz gráfica de SPARQL. Algunos de ellos, como estrategia para disminuir el tiempo requerido en la formulación de la consulta, decidieron no borrar el patrón de la consulta formulada previamente, de manera que podían tomar de ella la organización de la información común en las dos consultas. De otra parte, a pesar de que los usuarios entienden la representación de los datos en el esquema y la instancia, extraer las implicaciones de los efectos de los operadores puede resultar difícil en algunos casos, por ejemplo, para especificar las condiciones de filtro en los diferentes puntos de bifurcación donde se generan las preguntas de clarificación de filtros, en el caso de GraphTQL, o para identificar cuándo es necesario crear más de una variable para el mismo objeto, en el caso de SPARQL.

Las pruebas de evaluación del prototipo (Sección 6.3), permitieron implementar varias mejoras que en su mayoría generaron pequeños cambios. Sin embargo, estos cambios tuvieron impacto importante en los resultados de las métricas tomadas durante la formulación de consultas en GraphTQL. Entre estos cambios están incluir los diálogos de clarificación de caminos, cambios en el nombre de los operadores, ajustes en la semántica de los operadores, cambio de los iconos, redefinición de los controles del ratón usados en la interfaz, modificación de los mensajes de error y adaptación manual del diagrama del grafo esquema, a partir de la disposición calculada por el KKLLayout [103], para mejorar su legibilidad.

Con la realización de las pruebas de usabilidad (Capítulo 6) se logró identificar algunas necesidades de los usuarios en cuanto a consultas sobre datos clínicos en diferentes escenarios. Se comprobó también que la representación gráfica del grafo esquema es útil como estrategia de interacción para el lenguaje visual, ya que los usuarios interpretan fácilmente este diagrama y permite reducir algunos errores que ocurren cuando se formula la consulta construyendo un patrón a partir de la exploración de los datos. De otra parte, se observó que los participantes tienden a aceptar lo que la herramienta les propone por defecto, por ejemplo, las marcas de caminos o el operador que se marca cuando se especifica una condición de filtro. Por tanto, las guías y ayudas deben ser hechas para que el usuario termine la acción y en ese proceso decida, de manera consciente, sobre el aspecto en cuestión.

En el capítulo 7 se mostró que la implementación de los operadores de GraphTQL usando recorridos de caminos sobre una base de datos de grafos es una buena opción. Las consultas implementadas con recorridos de caminos se comparan favorablemente, en tiempo de ejecución, con consultas expresadas en SPARQL y ejecutadas en un motor de triplets, con ganancias significativas en varios casos (Sección 7.2.2). Se esperaría que al incluir opciones de optimización en la parte de procesamiento que se realiza por fuera del motor de grafos esos tiempos mejoren.

La tensión entre la expresividad y la facilidad de uso es siempre una limitante en los lenguajes visuales [20, 79], esto produce una brecha entre las consultas que se pueden formular en el lenguaje visual y las que se pueden expresar en un lenguaje basado en texto como

SPARQL. En el caso particular de GraphTQL hay tres limitantes que se discuten en la Sección 5.4: la primera, se presenta a nivel de la interfaz, donde no es posible especificar para un nodo de bifurcación una expresión lógica de filtro que combine conjunción y disyunción; la segunda, se requiere agregar un operador que permita filtrar los datos con base en comparaciones lógicas en los valores de los nodos de dos clases particulares; y la tercera, de acuerdo con las necesidades expresadas por los participantes en la pruebas de usabilidad, es importante ofrecer funciones agregadas y funciones de estadística básica.

El estudio de técnicas de despliegue ofrecidas por las herramientas de visualización de grafos (Sección 6.3) mostró que, entre las técnicas probadas, la mejor disposición para el diagrama del grafo esquema es la que calcula el algoritmo KKLLayout. Aun así, se tienden a presentar cruces de arcos y mezcla, en la distribución espacial, de atributos de diferentes clases que pueden generar confusión en los usuarios. Estos problemas de la disposición del diagrama se pueden deber a que, generalmente, los esquemas de datos combinan jerarquía (ej. la representación de atributos de un tipo de objeto) y red (i.e. la representación de relaciones entre objetos). Los *layouts* que se probaron no parecen combinar estas características, por lo cual el resultado es visualmente aceptable pero puede ser mejor. De otra parte, estos problemas conllevan a que el usuario necesite más tiempo para interpretar el modelo y a que, posiblemente, cometa errores en la formulación de la consulta. Para evitarlo, se hizo un ajuste manual, sobre los resultados de KKLLayout, en la posición de los nodos de los diagramas de los grafos esquema que se usaron en las pruebas.

8.3 Trabajo futuro

Algunas de las líneas de trabajo futuro derivadas de este estudio al igual de trabajos relacionados con el desarrollo de GraphTQL tendientes a producir una herramienta que pueda ser usada en un ambiente real, se presentan a continuación.

8.3.1 El lenguaje visual de consulta

La expresividad de GraphTQL se puede ampliar agregando, entre otras, la posibilidad de establecer condiciones comparando los valores de diferentes tipos de nodo o adicionando operadores para calcular funciones agregadas sobre los datos. Estas adiciones a la funcionalidad del lenguaje se deben realizar manteniendo la simplicidad del lenguaje y la carga cognitiva baja. Una forma posible de abordar la comparación de valores entre nodos se planteó en la Sección 5.4.

En cuanto a las funciones agregadas, posiblemente sea más claro para los usuarios si se aplican sobre una representación de los datos en tablas. Para ello sería necesario ofrecer al usuario la opción de cambiar la representación de los resultados de las consultas, de grafo a tabla. De esta manera las funciones agregadas se aplicarían sobre resultados de unas operaciones previas. Una manera de implementarlo puede ser acoplando GraphTQL con un sistema de exploración y análisis de datos con la capacidad de calcular datos estadísticos y operaciones agregadas.

De acuerdo con los resultados de la prueba comparativa, persiste una dificultad cuando los participantes usan GraphTQL, que se evidencia por el número de errores debido a respuestas dadas en las preguntas del diálogo de clarificación de filtros. Una línea de trabajo futuro puede apuntar a mejorar estos resultados. Una alternativa a explorar sería incluir una representación visual de las opciones que se proponen al usuario en cada pregunta de los diálogos. Esto puede resultar sencillo en los diálogos de clarificación de caminos, donde se espera que hayan pocos caminos a tener en cuenta, dado que es un grafo esquema y que el diálogo se requiere solamente para desambiguar ciclos. Por el contrario, en el diálogo de clarificación de filtros llevar a cabo esta opción puede resultar complejo, debido a la cantidad de posibles combinaciones que se pueden dar por la respuesta que el usuario da en cada nodo de bifurcación.

Otra vía que es interesante explorar es cómo la información semántica que se pueda agregar al grafo esquema ayudaría a mejorar los diálogos de clarificación. Esta es información que no se necesita desplegar en la interfaz, pero que se puede usar para que las preguntas de los diálogos tengan más sentido y coherencia. Por ejemplo, identificar en el esquema cuáles atributos son multivaluados puede servir para generar preguntas que guíen mejor al usuario en la selección de caminos o en la clarificación de los filtros.

De otra parte, dado que ontologías de dominio o vocabularios estructurados, tales como ICD10¹, LOINC², CPT³ o SNOMED⁴, se pueden usar en la representación de los datos, un trabajo futuro puede estar enfocado en la implementación de técnicas de expansión de consultas basadas, por ejemplo en la jerarquía de los conceptos representada en estas ontologías y vocabularios.

8.3.2 La usabilidad del lenguaje

A nivel de la interfaz, se podría tener en cuenta el nivel de experticia del usuario en el uso del lenguaje. De manera que, a mayor experticia, se usen menos los diálogos para guiar al usuario en la formulación de la consulta. Esto implica que se reemplacen los diálogos de clarificación por otras opciones en las que el usuario, de manera más directa, establece los parámetros de los operadores. Esto a su vez deja la posibilidad de que el usuario pueda hacer uso mas libre de la especificación de los parámetros, pudiendo, por ejemplo, usar conjunciones y disyunciones combinadas en la expresión de filtro de un nodo de bifurcación, ya que esta es una limitante que se genera en la interfaz y no por la definición de los operadores.

De otra parte, para cerrar el ciclo de diseño centrado en el usuario, es deseable realizar pruebas de usabilidad en ambientes reales controlados. Esto permite conocer otras necesidades de consulta, comprobar en qué medida la herramienta ayuda al usuario, en una situación real, a encontrar los datos que requiere e identificar el proceso y análisis que el usuario realiza, desde el momento en que tiene su propia necesidad de consulta, hasta plasmarlo en GraphTQL y usar los resultados.

¹<http://apps.who.int/classifications/icd10/browse/2015/en>

²<http://loinc.org>

³<http://www.ama-assn.org/go/cpt>

⁴https://www.nlm.nih.gov/research/umls/Snomed/snomed_main.html

Es importante también estudiar otras técnicas de visualización del grafo esquema para facilitar la navegación cuando el grafo esquema es grande (i.e su tamaño es varias veces el tamaño de la pantalla) y para reducir la sobrecarga de información que el grafo puede generar. Una opción es ocultar los atributos mientras estos no hayan sido seleccionados y mostrarlos solamente cuando el ratón pasa por el nodo objeto, aplicando la técnica de lentes [189]. De esta manera se destacarían en el modelo los objetos y sus relaciones, que servirían como punto de partida para que el usuario explore el grafo y le darían una vista más general de la organización de los datos. Asociado a esto, sería necesario estudiar como impactan estos cambios la interacción con el usuario. Por ejemplo, como puede sesgar las preguntas que el usuario se formule, dado que ya no tiene la vista completa de la información que está disponible para consultar.

8.3.3 Aspectos relacionados con la implementación

Teniendo en cuenta que parte del procesamiento de los operadores se implementó en Java, manipulando los resultados que arroja el *path traversal API* de Neo4j, es importante estudiar alternativas que permitan optimizar su ejecución. Un mecanismo que puede ser útil es ejecutar las condiciones de filtro en un orden diferente al que se especifica en los parámetros, dependiendo de su selectividad, calculada a partir de estadísticas de los datos. Otra opción es implementar el manejo de memoria caché en estas partes del procesamiento.

En cuanto a las técnicas de visualización de grafos, un trabajo a futuro consiste en encontrar o diseñar un algoritmo de *layout* que, teniendo en cuenta características usuales de los grafos esquema, calcule una disposición de nodos y arcos suficientemente legible, de manera que no se requiera realizar un ajuste manual.

Finalmente, para usar GraphTQL como mecanismo de consulta de datos de un conjunto de fuentes heterogéneas, se puede acoplar el lenguaje con un sistema de integración de datos basado en grafos (ej. RDF) que aplique heurísticas y estadísticas de los datos para hacer una distribución de la consulta que tenga tiempos de ejecución eficientes.

Publicaciones

- [1] M. C. Pabón and C. A. Collazos. Lenguaje visual de consulta sobre grafos de datos: un enfoque desde el diseño centrado en el usuario. In *16th International Conference Interacción 2015*, Vilanova i la Geltrú, Spain. ACM Press, 2015.
- [2] M. C. Pabón, C. Roncancio, and M. Millán. Graph Data Transformations and Querying. In *Proceedings of the 7th International C* Conference on Computer Science & Software Engineering - C3S2E'14*, Montreal, Canada. ACM Press, 2014.
- [3] M. C. Pabón, C. Roncancio, and M. Millán. Graph Management to Improve Querying of Health and Social Data. In *Proceedings of the 7th International Conference of Health Informatics, HealthInf*, Angers, Francia. Scitepress, 2014.
- [4] M. C. Pabón, G. A. Montoya, and M. Millán. Mediation and graph data models for medical data integration. In *Proceedings of the XXXIX Latin American Computing Conference (CLEI)*, Venezuela. IEEE, 2013.

Referencias

- [1] J. Aasman and K. Cheetham. RDF Browser for Data Discovery and Visual Query Building. In *Procs. of the Workshop on Visual Interfaces to the Social and Semantic Web*, 2011. [50](#), [51](#), [55](#), [59](#)
- [2] S. Abiteboul and R. Hull. IFO: a formal semantic database model. *ACM Transactions on Database Systems*, 12(4):525–565, Nov. 1987. [14](#)
- [3] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener. The Lorel Query Language for Semistructured Data. *International Journal on Digital Libraries*, 1(68–88), 1997. [29](#)
- [4] S. Agrawal, S. Chaudhuri, and G. Das. DBXplorer: a system for keyword-based search over relational databases. In *Procs. 18th Intl. Conf. on Data Engineering*. IEEE Comput. Soc, 2002. [37](#)
- [5] B. Amann and M. Scholl. Gram: a graph data model and query languages. In *Proceedings of the ACM conference on Hypertext*, ECHT ’92, pages 201–211. ACM, 1992. [18](#)
- [6] M. Andries, M. Gemis, J. Paredaens, I. Thyssens, and J. V. den Bussche. Concepts for Graph-Oriented Object Manipulation. In *Proc. of the 3rd Intl. Conf. on Extending Database Technology: Advances in Database Technology*, {EDBT} ’92, London, UK, 1992. Springer-Verlag. [27](#)
- [7] R. Angles and C. Gutierrez. Survey of Graph Database Models. *ACM Computing Surveys*, 40(1), Feb. 2008. [2](#), [15](#), [16](#), [20](#), [67](#)
- [8] R. Angles, A. Prat-Pérez, D. Dominguez-Sal, and J.-L. Larriba-Pey. Benchmarking database systems for social network applications. In *First International Workshop on Graph Data Management Experiences and Systems - GRADES ’13*, pages 1–7, New York, New York, USA, June 2013. ACM Press. [156](#)
- [9] S. F. C. Araujo and D. Schwabe. Explorator: a tool for exploring RDF data through direct manipulation. In *Linked Data on the Web LDOW*, 2009. [43](#), [44](#)
- [10] M. Arenas, A. Bertails, E. Prud’hommeaux, and J. Sequeda. A Direct Mapping of Relational Data to RDF, 2012. [176](#)

-
- [11] J. S. Ash, M. Berg, and E. Coiera. Some unintended consequences of information technology in health care: the nature of patient care information system-related errors. *Journal of the American Medical Informatics Association : JAMIA*, 11(2):104–12, 2004. [3](#)
 - [12] Y. Bai, C. Wang, X. Ying, M. Wang, and Y. Gong. Path Pattern Query Processing on Large Graphs. In *2014 IEEE Fourth International Conference on Big Data and Cloud Computing*, pages 767–774. IEEE, Dec. 2014. [27](#)
 - [13] M. Baitaluk, X. Qian, S. Godbole, A. Raval, A. Ray, and A. Gupta. PathSys: integrating molecular interaction graphs for systems biology. *BMC bioinformatics*, 7(1):55, 2006. [6](#)
 - [14] A. Bangor, P. Kortum, and J. Miller. An Empirical Evaluation of the System Usability Scale. *Intl. Journal of Human-Computer Interaction*, 24(6), 2008. [136](#), [140](#), [146](#)
 - [15] P. Barceló, L. Libkin, A. W. Lin, and P. T. Wood. Expressive Languages for Path Queries over Graph-Structured Data. *ACM Transactions on Database Systems*, 37(4):1–46, Dec. 2012. [17](#), [21](#), [22](#)
 - [16] P. Barceló Baeza. Querying graph databases. In *Procs. of the 32nd symposium on Principles of database systems PODS '13*, New York, USA, June 2013. ACM Press. [17](#)
 - [17] G. Barzdins, E. Liepins, M. Veilande, and M. Zviedris. Ontology Enabled Graphical Database Query Tool for End-Users. In H.-M. Haav and A. Kalja, editors, *Databases and Information Systems V: Selected Papers from the Eighth Intl. Baltic Conf., DB&IS 2008*. IOS Press, 2009. [47](#), [48](#)
 - [18] G. Barzdins, S. Rikacovs, and M. Zviedris. Graphical Query Language as SPARQL Frontend. In *13th East-European Conf. (ADBIS 2009)*, 2009. [53](#), [54](#)
 - [19] M. Bastian, S. Heymann, and M. Jacomy. Gephi: An Open Source Software for Exploring and Manipulating Networks. In *Intl. AAAI Conf. on Weblogs and Social Media*, 2009. [43](#), [55](#)
 - [20] C. Batini, T. Catarci, M. F. Costabile, and S. Levialdi. Visual Query Systems: A Taxonomy. In *Procs. of the IFIP TC2/WG 2.6 Second Working Conf. on Visual Database Systems II*. North-Holland Publishing Co., 1991. [1](#), [2](#), [39](#), [182](#)
 - [21] T. Beale, S. Heard, D. Kalra, and D. Lloyd, editors. *EHR Information Model*. The openEHR Foundation, 2007. [68](#)
 - [22] J. Bell and L. Rowe. An exploratory study of ad hoc query languages to databases. In *Eight Intl. Conf. on Data Engineering*. IEEE Computer Society Press, 1992. [141](#)
 - [23] T. Berners-lee and D. Connolly. Notation3 (N3): A readable RDF syntax, 2011. [49](#)

-
- [24] N. Bevan, J. Kirakowski, and J. Maissel. What is Usability. In *Proc. of the 4th Intl. Conf. on Human Computer Interaction*, Stuttgart, Sept. 1991. Elsevier. [32](#)
 - [25] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using BANKS. In *Proceedings 18th International Conference on Data Engineering*, pages 431–440. IEEE Comput. Soc, 2002. [37](#)
 - [26] J. A. Blakeley, D. Campbell, S. Muralidhar, and A. Nori. The ADO.NET entity framework: making the conceptual level real. *ACM SIGMOD Record*, 35(4):32, Dec. 2006. [14](#)
 - [27] H. Blau, N. Immerman, and D. Jensen. A Visual Language for Querying and Updating Graphs. Technical report, University of Massachusetts, Amherst, 2002. [49](#), [50](#)
 - [28] A. C. Bloesch and T. A. Halpin. ConQuer: A conceptual query language. In *Conceptual Modeling ER'96*, volume 1157 of *LNCS*. Springer-Verlag, Berlin/Heidelberg, 1996. [15](#)
 - [29] U. Brandes, M. Eiglsperger, J. Lerner, and C. Pich. Graph Markup Language (GraphML). In R. Tamassia, editor, *Handbook of Graph Drawing and Visualization*, chapter 16. Chapman and Hall/CRC, 2013. [139](#), [154](#)
 - [30] K. Braunschweig, M. Thiele, and W. Lehner. A flexible graph-based data model supporting incremental schema design and evolution. In A. Harth and N. Koch, editors, *Procs. of the 11th Intl. Conf. on Current Trends in Web Engineering ICWE'11*, volume 7059 of *Lecture Notes in Computer Science*, pages 302–306, Berlin, Heidelberg, June 2012. Springer Berlin Heidelberg. [19](#)
 - [31] D. Brickley and R. Guha. RDF Schema 1.1, 2014. [24](#)
 - [32] J. Brooke. SUS A quick and dirty usability scale. In *Usability evaluation in industry*, 1996. [146](#)
 - [33] R. Bruni and A. L. Lafuente. Ten virtues of structured graphs, 2009. [81](#)
 - [34] P. Buneman, M. Fernandez, and D. Suciu. UnQL: a query language and algebra for semistructured data based on structural recursion. *The VLDB Journal*, 9(1):76, Mar. 2000. [29](#)
 - [35] D. Calvanese, G. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Conceptual Modeling for Data Integration. In A. T. Borgida, V. K. Chaudhri, P. Giorgini, and E. S. Yu, editors, *Conceptual Modeling: Foundations and Applications*, pages 173–197. Springer-Verlag, Berlin, Heidelberg, 2009. [14](#)
 - [36] T. Catarci. Visual Query Language. In *Encyclopedia of Database Systems*, pages 3399–3404. Springer US, 2009. [2](#), [39](#)

-
- [37] T. Catarci, T. Di Mascio, P. Dongilli, E. Franconi, G. Santucci, and S. Tessaris. Usability evaluation in the SEWASIE (SEmantic Webs and AgentS in Integrated Economies) project. In *11th Intl. Conf. on Human- Computer Interaction (HCII)*, 2005. [49](#), [50](#), [151](#)
 - [38] M. Cayli, M. C. Cobanoglu, and S. Balcisoy. GlyphLink: An interactive visualization approach for semantic graphs. *Journal of Visual Languages & Computing*, 24(6), Dec. 2013. [43](#), [44](#)
 - [39] A. Chapman and H. V. Jagadish. Why not? In *Proceedings of the 35th SIGMOD international conference on Management of data - SIGMOD '09*, page 523, New York, USA, June 2009. ACM Press. [38](#)
 - [40] D. H. Chau, C. Faloutsos, H. Tong, J. I. Hong, B. Gallagher, and T. Eliassi-Rad. GRAPHITE: A Visual Query System for Large Graphs. In *2008 IEEE International Conference on Data Mining Workshops*, pages 963–966. IEEE, Dec. 2008. [49](#)
 - [41] P. Chen. The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, Mar. 1976. [2](#), [14](#)
 - [42] Z. Chen and T. Li. Addressing diverse user preferences in SQL-query-result navigation. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data - SIGMOD '07*, page 641, New York, USA, June 2007. ACM Press. [37](#)
 - [43] J. Cheney, L. Chiticariu, and W.-C. Tan. Provenance in Databases: Why, How, and Where. *Foundations and Trends in Databases*, 1(4):379–474, Apr. 2007. [38](#)
 - [44] J. Cheng, Y. Ke, W. Ng, and A. Lu. Fg-index: : towards verification-free query processing on graph databases. In *Procs. of the 2007 ACM SIGMOD Intl. Conf. on Management of Data - SIGMOD '07*, New York, USA, June 2007. ACM Press. [22](#)
 - [45] T. Claudio. *Getting Started with OrientDB*. Packt Publishing, 2013. [20](#)
 - [46] E. F. Codd. Data models in database management. In *Proc. of the 1980 workshop on Data abstraction, databases and conceptual modeling -*, pages 112–114, Pingree Park, Colorado, United States, 1980. [13](#)
 - [47] M. P. Consens and A. O. Mendelzon. GraphLog: a visual formalism for real life recursion. In *Proc. of the ninth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, PODS '90, pages 404–416, New York, NY, USA, 1990. ACM. [44](#), [45](#)
 - [48] B. B. Dalvi, M. Kshirsagar, and S. Sudarshan. Keyword search on external memory data graphs. *Procs. of the VLDB Endowment*, 1(1), Aug. 2008. [37](#)

-
- [49] D. Damjanovic, M. Agatonovic, H. Cunningham, and K. Bontcheva. Improving habitability of natural language interfaces for querying ontologies with feedback and clarification dialogues. *Web Semantics: Science, Services and Agents on the World Wide Web*, 19, Mar. 2013. [38](#)
 - [50] M. Dayarathna and T. Suzumura. Graph database benchmarking on cloud environments with XGDBench. *Automated Software Engineering*, 21(4):509–533, Nov. 2013. [156](#)
 - [51] L. Deligiannidis, K. J. Kochut, and A. P. Sheth. RDF data exploration and visualization. In *Proceedings of the ACM first workshop on CyberInfrastructure: information management in eScience - CIMS '07*, page 39, New York, USA, Nov. 2007. ACM Press. [44](#)
 - [52] A. Dries and S. Nijssen. Analyzing graph databases by aggregate queries. In *Proc. of the Eighth Workshop on Mining and Learning with Graphs*, MLG '10, pages 37–45, New York, USA, 2010. ACM. [22](#), [24](#)
 - [53] A. Dries, S. Nijssen, and L. De Raedt. A query language for analyzing networks. In *Proc. of the 18th ACM conf. on Information and knowledge management*, CIKM '09, pages 485–494, New York, USA, 2009. ACM. [24](#)
 - [54] H. El-Ghalayini, M. Odeh, and R. McClatchey. Developing ontology-driven conceptual data models. In *Proc. of the 1st Intl. Conf. on Intelligent Semantic Web-Services and Applications*, ISWSA '10. ACM, 2010. [14](#)
 - [55] A. Elsayed, A. S. Eldin, and D. S. El Zanfaly. Supporting ontology-driven Keyword Search over Relational Databases. In *2014 World Symposium on Computer Applications & Research (WSCAR)*, pages 1–6. IEEE, Jan. 2014. [37](#)
 - [56] A. Fadhlil and V. Haarslev. OntoVQL: A Graphical Query Language for OWL Ontologies. In *Description Logics*, 2008. [47](#), [48](#)
 - [57] S. Fakhraee and F. Fotouhi. Effective Keyword Search over Relational Databases Considering Keywords Proximity and Keywords N-grams. In *2011 22nd International Workshop on Database and Expert Systems Applications*, pages 190–194. IEEE, Aug. 2011. [37](#)
 - [58] W. Fan, J. Li, S. Ma, N. Tang, Y. Wu, and Y. Wu. Graph pattern matching. *Procs. of the VLDB Endowment*, 3(1-2), 2010. [18](#), [21](#), [156](#)
 - [59] M. Fernández, D. Florescu, A. Levy, and D. Suciu. Declarative specification of Web sites with Strudel. *The VLDB Journal*, 9(1):38–55, Mar. 2000. [26](#)
 - [60] J. J. Garrett. *Elements of User Experience, The: User-Centered Design for the Web and Beyond*. Pearson Education, 2010. [3](#), [7](#), [31](#), [32](#), [33](#), [131](#)

-
- [61] M. Gemis and J. Paredaens. An Object-Oriented Pattern Matching Language. In *JSSST, Intl. Symp. on Object Technologies for Advanced Software*, pages 339–355, Kanazawa, Japan, 1993. Springer-Verlag. [18](#)
 - [62] G. L. Glandon, D. H. Smalz, and D. J. Slovensky. *Austin and Boxerman's Information Systems For Healthcare Management*. Health Administration Press/AUPHA, 7 edition, 2008. [3](#)
 - [63] K. Golenberg, B. Kimelfeld, and Y. Sagiv. Keyword proximity search in complex data graphs. In *Procs. of the 2008 ACM SIGMOD Intl. Conf. on Management of Data - SIGMOD '08*, page 927, New York, New York, USA, June 2008. ACM Press. [37](#)
 - [64] G. Grahne and A. Thomo. Regular path queries under approximate semantics. *Annals of Mathematics and Artificial Intelligence*, 46(1-2):165–190, Feb. 2006. [22](#)
 - [65] M. Graves, E. R. Bergeman, and C. B. Lawrence. A graph-theoretic data model for genome mapping databases. In *Procs. of the 28th Hawaii Itl. Conf. on System Sciences, HICSS '95*, Washington, DC, USA, 1995. IEEE Computer Society. [15](#)
 - [66] J. Groppe, S. Groppe, and A. Schleifer. Visual query system for analyzing social semantic web. In *Proc. of the 20th Intl. Conf. Companion on World Wide Web, WWW '11*. ACM, 2011. [53](#), [54](#), [55](#), [58](#)
 - [67] F. D. Guanga and J. A. Londoño. *Interfaz gráfica para consulta de datos basada en operaciones de transformación de grafos*. Trabajo de Grado, Pontificia Universidad Javeriana - Seccional de Cali, 2014. [136](#), [139](#), [155](#)
 - [68] R. H. Güting. GraphDB: Modeling and Querying Graphs in Databases. In *Proc. of the 20th Intl. Conf. on Very Large Data Bases, VLDB '94*, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc. [19](#), [20](#), [25](#)
 - [69] M. Gyssens, J. Paredaens, and D. V. Gucht. A graph-oriented object model for database end-user interfaces. *SIGMOD Record*, 19(2):24–33, May 1990. [18](#), [44](#), [45](#)
 - [70] M. Gyssens, J. Paredaens, and D. van Gucht. A graph-oriented object database model. In *Proc. of the ninth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, PODS '90*, pages 417–424. ACM, 1990. [18](#)
 - [71] F. Haag, S. Lohmann, and T. Ertl. SparqlFilterFlow: SPARQL Query Composition for Everyone. In V. Presutti, E. Blomqvist, R. Troncy, H. Sack, I. Papadakis, and A. Tordai, editors, *The Semantic Web: ESWC 2014 Satellite Events*, volume 8798 of *LNCS*. Springer International Publishing, 2014. [52](#), [53](#)
 - [72] B. Hafnerichter and W. Kieß ling. Optimization of relational preference queries. In *Procs. of the 16th Australasian Database Conf. ADC'05*. Australian Computer Society, Inc., Jan. 2005. [37](#)

-
- [73] T. Halpin. Conceptual Queries. *Database Newsletter*, 26(3), Apr. 1998. [15](#)
- [74] T. Halpin. Object-Role Modeling (ORM/NIAM). In P. Bernus, K. Mertins, and G. Schmidt, editors, *Handbook on Architectures of Information Systems*, International Handbooks on Information Systems, pages 81–103. Springer Berlin Heidelberg, 2006. [2](#), [14](#)
- [75] T. Halpin and A. C. Bloesch. Data Modeling in UML and ORM: A Comparison. *Journal of Database Management (JDM)*, 10(4), 1999. [14](#)
- [76] M. Hammer and D. Mc Leod. Database description with SDM: a semantic database model. *ACM Transactions on Database Systems*, 6(3):351–386, Sept. 1981. [14](#)
- [77] S. Harris. 4store: The design and implementation of a clustered RDF store. In *Procs. of the 5th Intl. Workshop on Scalable Semantic Web Knowledge BaseSystems (SSWS2009)*, 2009. [20](#)
- [78] S. Harris and A. Seaborne, editors. *SPARQL 1.1 Query Language*. W3C Recommendation, 2013. [23](#)
- [79] A. Harth, S. R. Kruk, and S. Decker. Graphical representation of RDF queries. In *Procs. of the 15th intl. conf. on World Wide Web - WWW '06*, page 859, New York, USA, May 2006. ACM Press. [2](#), [49](#), [182](#)
- [80] J. Hayes and C. Gutierrez. Bipartite Graphs as Intermediate Model for RDF. In *The Semantic Web - ISWC 2004*, volume 3298, pages 47–61. Springer Berlin Heidelberg, 2004. [17](#)
- [81] H. He and A. K. Singh. Graphs-at-a-time. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data - SIGMOD '08*, page 405, New York, New York, USA, June 2008. ACM Press. [28](#)
- [82] H. He, H. Wang, J. Yang, and P. S. Yu. BLINKS: Ranked keyword searches on graphs. In *Procs. of the 2007 ACM SIGMOD Intl. Conf. on Management of Data - SIGMOD '07*, page 305, New York, USA, June 2007. ACM Press. [37](#)
- [83] C. Heath. The Constellation Query Language. In *On the Move to Meaningful Internet Systems: OTM 2009 Workshops*, volume 5872 of *LNCS*, pages 682–691. Springer, 2009. [15](#)
- [84] L. S. Heath and A. A. Sioson. Multimodal networks: structure and operations. *IEEE/ACM Trans. on Computational Biology and Bioinformatics / IEEE, ACM*, 6(2):321–32, Jan. 2009. [19](#)
- [85] M. Herschel and M. A. Hernández. Explaining missing answers to SPJUA queries. *Proceedings of the VLDB Endowment*, 3(1-2):185–196, Sept. 2010. [38](#)

-
- [86] A. J. H. Hidders. *A graph-based update language for object-oriented data models*. PhD thesis, Eindhoven University, 2001. [93](#), [98](#)
 - [87] J. Hidders. Typing Graph-Manipulation Operations. In *Proc. of the 9th Intl. Conf. on Database Theory*, ICDT. Springer-Verlag, 2002. [10](#), [18](#), [26](#), [81](#), [93](#), [96](#), [98](#)
 - [88] J. Hidders and J. Paredaens. GOAL, A Graph-based Object and Association Language. *CISM - Advances in Database Systems*, pages 247–265, 1993. [18](#)
 - [89] F. Hogenboom, V. Milea, F. Frasincar, and K. Uzay. RDF-GL: A SPARQL-Based Graphical Query Language for RDF. In R. Chbeir, Y. Badr, A. Abraham, and A.-E. Hassanien, editors, *Emergent Web Intelligence: Advanced Information Retrieval*, Advanced Information and Knowledge Processing, pages 87–116. Springer London, 2010. [55](#)
 - [90] V. Hristidis and Y. Papakonstantinou. Discover: keyword search in relational databases. In *Procs. of the 28th Intl. Conf. on Very Large Data Bases VLDB '02*. VLDB Endowment, Aug. 2002. [37](#)
 - [91] J. Huang, T. Chen, A. Doan, and J. F. Naughton. On the provenance of non-answers to queries over extracted data. *Proceedings of the VLDB Endowment*, 1(1):736–747, Aug. 2008. [38](#)
 - [92] H. H. Hung, S. S Bhowmick, B. Q. Truong, B. Choi, and S. Zhou. QUBLE: blending visual subgraph query formulation with query processing on large networks. In *Procs. of the 2013 Intl. Conf. on Management of Data - SIGMOD '13*. ACM Press, 2013. [50](#)
 - [93] C. Hurtado, A. Poulovassilis, and P. Wood. Ranking Approximate Answers to Semantic Web Queries. In L. Aroyo, P. Traverso, F. Ciravegna, P. Cimiano, T. Heath, E. Hyvonen, R. Mizoguchi, E. Oren, M. Sabou, and E. Simperl, editors, *The Semantic Web: Research and Applications LNCS 5554*. Springer Berlin Heidelberg, 2009. [22](#)
 - [94] IEEE. IEEE Standard for Conceptual Modeling Language Syntax and Semantics for IDEF1X/97, 1998. [14](#)
 - [95] B. Iordanov. HyperGraphDB: A Generalized Graph Database. In H. T. Shen, J. Pei, M. T. Özsü, L. Zou, J. Lu, T.-W. Ling, G. Yu, Y. Zhuang, and J. Shao, editors, *Web-Age Information Management*, volume 6185 of *LNCS*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. [19](#), [20](#)
 - [96] M. S. Islam, C. Liu, and R. Zhou. On modeling query refinement by capturing user intent through feedback. In *Proceedings of the Twenty-Third Australasian Database Conference Vol. 124 ADC '12*. Australian Computer Society, Inc., Jan. 2012. [38](#), [39](#)
 - [97] H. V. Jagadish, A. Chapman, A. Elkiss, M. Jayapandian, Y. Li, A. Nandi, and C. Yu. Making database systems usable. In *Procs. of the 2007 ACM SIGMOD Intl. Conf. on Management of data - SIGMOD '07*, page 13, New York, USA, June 2007. ACM Press. [1](#), [2](#), [8](#), [38](#)

-
- [98] M. Jarrar and M. D. Dikaiakos. A Query Formulation Language for the Data Web. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):783–798, May 2012. [1](#), [2](#), [40](#), [46](#), [151](#)
 - [99] H. Jiang, H. Wang, P. S. Yu, and S. Zhou. GString: A Novel Approach for Efficient Search in Graph Databases. In *2007 IEEE 23rd Intl. Conf. on Data Engineering*. IEEE, 2007. [22](#)
 - [100] R. Jin, Y. Xiang, N. Ruan, and H. Wang. Efficiently answering reachability queries on very large directed graphs. In *Procs. of the 2008 ACM SIGMOD Intl. Conf. on Management of Data - SIGMOD '08*, New York, USA, June 2008. ACM Press. [22](#)
 - [101] A. Jindal. Graph Analytics: The New Use Case for Relational Databases. Intel Science & Technology Center for Big Data, 2014. [67](#)
 - [102] V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, and H. Karambelkar. Bidirectional expansion for keyword search on graph databases. In *Procs. of the 31st Intl. Conf. on Very Large Data Bases VLDB '05*. VLDB Endowment, Aug. 2005. [37](#)
 - [103] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, Apr. 1989. [139](#), [155](#), [182](#)
 - [104] N. N. Karanikolas. Conceptual universal database language. In *Proc. of the 2009 Euro American Conf. on Telematics and Information Systems New Opportunities to increase Digital Citizenship - EATIS '09*, pages 1–5, Prague, Czech Republic, 2009. [15](#)
 - [105] M. Kargar and A. An. Keyword search in graphs. *Proceedings of the VLDB Endowment*, 4(10):681–692, July 2011. [37](#)
 - [106] G. Kasneci, M. Ramanath, F. Suchanek, and G. Weikum. The YAGO-NAGA approach to knowledge discovery. *ACM SIGMOD Record*, 37(4):41, Mar. 2009. [27](#)
 - [107] E. Kaufmann and A. Bernstein. Evaluating the usability of natural language query languages and interfaces to Semantic Web knowledge bases. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(4):377–393, Nov. 2010. [37](#)
 - [108] E. Kaufmann, A. Bernstein, and R. Zumstein. Querix: A Natural Language Interface to Query Ontologies Based on Clarification Dialogs. In *Procs. of the 5th Intl. Semantic Web Conf.* Springer, 2006. [37](#), [38](#), [39](#)
 - [109] N. Kiesel, A. Schurr, and B. Westfechtel. GRAS, a graph-oriented database system for (software) engineering applications. In *Procs. of 6th Intl. Workshop on Computer-Aided Software Engineering*. IEEE Comput. Soc. Press, 1993. [19](#)
 - [110] W. Kieß ling. Foundations of preferences in database systems. In *Procs. of the 28th Intl. Conf. on Very Large Data Bases VLDB '02*. VLDB Endowment, Aug. 2002. [37](#)

-
- [111] W. Kim. A model of queries for object-oriented databases. In *VLDB '89 Procs. of the 15th Intl. Conf. on Very large data bases*. Morgan Kaufmann Publishers Inc., July 1989. [16](#)
 - [112] G. Klyne and J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. <http://dret.net/biblio/reference/rdfconcepts>, 2004. [17](#), [23](#)
 - [113] G. Koutrika and Y. Ioannidis. Personalization of queries in database systems. In *Procs. 20th Intl. Conf. on Data Engineering*. IEEE Comput. Soc, 2004. [37](#)
 - [114] M. Lawley and R. Topor. A Query Language for EER Schemas. *ADC 94 Proc. of the 5th Australian Database Conf., Global Publications Service*, pages 292–304, 1994. [15](#)
 - [115] M. Lenzerini. Data integration: a theoretical perspective. In *Proc. of the 21st ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '02, pages 233–246, New York, USA, 2002. ACM. [171](#), [173](#)
 - [116] M. Levene and A. Poulovassilis. An Object-Oriented Data Model Formalised through Hypergraphs. *Data Knowl. Eng.*, 6(3):205–224, May 1991. [15](#), [19](#)
 - [117] M. Levene and A. Poulovassilis. The hypernode model and its associated query language. In *Proceedings of the fifth Jerusalem conference on Information technology*, JCIT, pages 520–530, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press. [15](#), [19](#)
 - [118] F. Li and H. V. Jagadish. NaLIR. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data - SIGMOD '14*, pages 709–712, New York, USA, June 2014. ACM Press. [37](#), [38](#), [39](#)
 - [119] L. Libkin and D. Vrgoč. Regular path queries on graphs with data. In *Procs. of the 15th Intl. Conf. on Database Theory ICDT '12*, page 74, New York, USA, Mar. 2012. ACM Press. [17](#), [20](#), [67](#)
 - [120] M. Llopis and A. Ferrández. How to make a natural language interface to query databases accessible to everyone: An example. *Computer Standards & Interfaces*, 35(5):470–481, Sept. 2013. [37](#)
 - [121] F. López, O. Ceballos, and N. Díaz. SMITAG: red social para la anotación semántica de imágenes médicas. In *XXXVIII Latin American Conf. on Informatics CLEI*, Colombia, 2012. [6](#)
 - [122] V. Lopez, V. Uren, E. Motta, and M. Pasin. AquaLog: An ontology-driven question answering system for organizational semantic intranets. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):72–105, June 2007. [38](#), [39](#)
 - [123] J. Lorés, M. Sendín, and J. Agost. Evaluación. In J. Lorés, editor, *La interacción Persona-Ordenador*. AIPO, 2002. [34](#)

-
- [124] F. Mandreoli and R. Martoglia. Semantics-driven Approximate Query Answering on Graph Databases. *Procs. Italian Symposium on Advanced Database Systems (SEBD 2009)*, 2009. [22](#)
- [125] F. Mandreoli, R. Martoglia, G. Villani, and W. Penzo. Flexible query answering on graph-modeled data. In *Procs. of the 12th Intl. Conf. on Extending Database Technology Advances in Database Technology EDBT '09*, New York, USA, 2009. ACM Press. [16](#), [17](#)
- [126] A. Markowetz, Y. Yang, and D. Papadias. Keyword search over relational tables and streams. *ACM Transactions on Database Systems*, 34(3):1–51, Aug. 2009. [37](#)
- [127] N. Martinez-Bazan, V. Muntes-Mulero, S. Gomez-Villamor, J. Nin, M.-A. Sanchez-Martinez, and J.-L. Larriba-Pey. Dex: high-performance exploration on large graphs for information retrieval. In *Proc. of the Sixteenth ACM Conf. on Information and Knowledge Management*, pages 573–582, Lisbon, Portugal, 2007. ACM. [20](#)
- [128] D. L. McGuinness, M. K. Smith, and C. Welty. OWL Web Ontology Language Guide, 2004. [47](#)
- [129] A. O. Mendelzon and P. T. Wood. Finding regular simple paths in graph databases. *SIAM Journal on Computing*, 24(6):185–193, July 1995. [21](#)
- [130] P. Mitra, G. Wiederhold, and M. Kersten. A Graph-Oriented Model for Articulation of Ontology Interdependencies. In C. Zaniolo, P. Lockemann, M. Scholl, and T. Grust, editors, *Advances in Database Technology EDBT 2000*, pages 86–100. Springer Berlin Heidelberg, 2000. [6](#), [14](#)
- [131] G. A. Montoya. *Mediador para la recuperación e integración de datos médicos de fuentes heterogéneas*. Trabajo de Grado, Escuela de Ingeniería de Sistemas y Computación, Universidad del Valle, 2013. [175](#)
- [132] D. Moody. The Physics of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. *IEEE Transactions on Software Engineering*, 35(6):756–779, Nov. 2009. [82](#), [88](#)
- [133] J. Mylopoulos. Conceptual Modelling and Telos. In P. Loucopoulos and R. Zicari, editors, *Conceptual Modelling, Databases and CASE: An Integrated View of Information Systems Development*, pages 49–68. John Wiley & Sons Inc., New York, 1992. [14](#)
- [134] T. Neumann and G. Weikum. The RDF-3X engine for scalable management of RDF data. *The VLDB Journal*, 19(1):91–113, Sept. 2009. [20](#)
- [135] J. Nielsen. *Usability Engineering*. Morgan Kaufmann, 1993. [32](#), [33](#), [34](#), [132](#), [140](#)
- [136] N. Nihalani, S. Silakari, and M. Motwani. Natural language interface for database: a brief review. *IJCSI Intl. Journal of Computer Science Issues*, 8(2), 2011. [37](#)

-
- [137] Y. Ohira, T. Hochin, and H. Nomiya. Introducing Specialization and Generalization to a Graph-Based Data Model. In A. König, A. Dengel, K. Hinkelmann, K. Kise, R. J. Howlett, and L. C. Jain, editors, *Knowledge-Based and Intelligent Information and Engineering Systems*, volume 6884 of *LNCS*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. [19](#)
 - [138] OMG. Unified Modeling Language (UML). <http://www.uml.org/>. [53](#)
 - [139] V. V. Ovchinnikov. A Conceptual Modeling Technique without Redundant Structural Elements. *Journal of Conceptual Modeling*, 29, 2003. [14](#)
 - [140] V. Owei. Development of a Conceptual Query Language: Adopting the UserCentered Methodology. *The Computer Journal*, 46(6):602–624, 2003. [15](#)
 - [141] V. Owei, S. B. Navathe, and H.-S. Rhee. An abbreviated concept-based query language and its exploratory evaluation. *Journal of Systems and Software*, 63(1):45–67, July 2002. [141](#)
 - [142] M. C. Pabón and C. A. Collazos. Lenguaje visual de consulta sobre grafos de datos: un enfoque desde el diseño centrado en el usuario. In *16th Intl. Conf. Interacción 2015*, Vilanova i la Geltrú, Spain, 2015. ACM Press.
 - [143] M. C. Pabón, G. A. Montoya, and M. Millan. Mediation and graph data models for medical data integration. In *Proc. XXXIX Latin American Computing Conference (CLEI)*. IEEE, 2013.
 - [144] M. C. Pabón, C. Roncancio, and M. Millán. Graph Data Transformations and Querying. In *Procs. of the 7th Intl. C* Conference on Computer Science & Software Engineering - C3S2E '14*, Montreal, Canada, 2014. ACM Press.
 - [145] M. C. Pabón, C. Roncancio, and M. Millán. Graph Management to Improve Querying of Health and Social Data. In *7th Intl. Conf. on Health Informatics, Healthinf*, Angers, Francia, 2014. Scitepress.
 - [146] N. Papadakis, P. Kefalas, and M. Stilianakakis. A tool for access to relational databases in natural language. *Expert Systems with Applications*, 38(6):7894–7900, June 2011. [37](#)
 - [147] J. Paredaens, J. den Bussche, M. Andries, M. Gemis, M. Gyssens, I. Thyssens, D. Van Gucht, V. Sarathy, and L. Saxton. An overview of GOOD. *ACM Comput. Surv.SIGMOD Record*, 21(1):25–31, Mar. 1992. [45](#)
 - [148] J. Paredaens, P. Peelman, and L. Tanca. G-Log: A Graph-Based Query Language. *IEEE Trans. on Knowledge and Data Eng.*, 7(3):436–453, June 1995. [18](#)
 - [149] C.-S. Park and S. Lim. Efficient processing of keyword queries over graph databases for finding effective answers. *Information Processing & Management*, 51(1), Jan. 2015. [37](#)

-
- [150] J. Peckham and F. Maryanski. Semantic data models. *ACM Comput. Surv.*, 20(3):153–189, Sept. 1988. [14](#)
- [151] J. Perez, M. Arenas, and C. Gutierrez. Semantics and Complexity of SPARQL. In *Intl. Semantic Web Conf.*, 2006. [21](#)
- [152] J. Pérez, M. Arenas, and C. Gutierrez. Semantics and complexity of SPARQL. *ACM Transactions on Database Systems*, 34(3):1–45, Aug. 2009. [23](#), [83](#), [84](#)
- [153] D. Petrelli. On the role of user-centred evaluation in the advancement of interactive information retrieval. *Information Processing & Management*, 44(1):22–38, Jan. 2008. [33](#), [132](#)
- [154] J. J. Pfeiffer, S. Moreno, T. La Fond, J. Neville, and B. Gallagher. Attributed graph models. In *Procs. of the 23rd Intl. Conf. on World wide web WWW ’14*, New York, USA, Apr. 2014. ACM Press. [18](#), [19](#)
- [155] A.-M. Popescu, A. Armanasu, O. Etzioni, D. Ko, and A. Yates. PRECISE on ATIS: semantic tractability and experimental results. In *Procs. of the 19th National Conf. on Artifical Intelligence AAAI’04*, pages 1026–1027. AAAI Press, July 2004. [37](#)
- [156] A. Poulovassilis and M. Levine. A nested-graph model for the representation and manipulation of complex objects. *ACM Trans. Inf. Syst.*, 12(1):35–68, Jan. 1994. [15](#)
- [157] A. Poulovassilis and P. McBrien. A general formal framework for schema transformation. *Data Knowl. Eng.*, 28(1):47–71, Oct. 1998. [15](#)
- [158] E. Prud’hommeaux and A. Seaborne. SPARQL Query Language for RDF. W3C Recommendation, Jan. 2008. [23](#), [40](#), [50](#)
- [159] RDF Working Group. Resource Description Framework (RDF), 2014. [40](#), [47](#), [49](#)
- [160] I. Robinson, J. Webber, and E. Eifrem. *Graph Databases*. O'Reilly Media, 2013. [20](#), [153](#)
- [161] R. Rodrigues Veloso, L. Cerf, W. Meira Junior, and M. J. Zaki. Reachability Queries in Very Large Graphs: A Fast Refined Online Search Approach. In *Procs. of the 17th Intl. conf. on Extending Database Technology EDBT*. OpenProceedings.org, 2014. [22](#)
- [162] M. A. Rodriguez and P. Neubauer. The Graph Traversal Pattern. In S. Sakr and E. Paredede, editors, *Graph Data Management: Techniques and Applications*. IGI Global, 2011. [16](#), [19](#), [21](#), [156](#)
- [163] R. Ronen and O. Shmueli. SoQL: A Language for Querying and Creating Data in Social Networks. In *2009 IEEE 25th Intl. Conf. on Data Engineering*. IEEE, Mar. 2009. [29](#)

-
- [164] N. Roussopoulos and D. Karagiannis. Conceptual Modeling: Past, Present and the Continuum of the Future. In A. T. Borgida, V. K. Chaudhri, P. Giorgini, and E. S. Yu, editors, *Conceptual Modeling: Foundations and Applications*, pages 139–152. Springer-Verlag, 2009. [13](#), [14](#)
 - [165] J. Rubin, D. Chisnell, and J. Spool. *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*. Wiley Publishing, Inc., 2nd editio edition, 2008. [3](#), [7](#), [32](#), [33](#), [34](#), [35](#), [131](#), [132](#), [140](#)
 - [166] A. Russell, P. Smart, D. Braines, and N. Shadbolt. NITELIGHT: A Graphical Tool for Semantic Query Construction. In *Workshop, Semantic Web User Interaction (SWUI 2008)*, Florence, 2008. [51](#)
 - [167] A. A. Sadanandan, K. W. Onn, and D. Lukose. Ontology Based Graphical Query Language Supporting Recursion. In *Proc. of the 14th Intl. Conf. on Knowledge-based and Intelligent Information and Eng. Systems: Part I*, volume 6276 of *LNCS*. Springer, 2010. [47](#)
 - [168] Y. Sagiv. A personal perspective on keyword search over data graphs. In *Procs. of the 16th Intl. Conf. on Database Theory ICDT '13*, page 21, New York, New York, USA, Mar. 2013. ACM Press. [2](#), [29](#)
 - [169] S. Sakr, S. Elnikety, and Y. He. Hybrid query execution engine for large attributed graphs. *Information Systems*, 41:45–73, May 2014. [27](#)
 - [170] M. San Martin, C. Gutierrez, and P. T. Wood. SNQL: A Social Networks Query and Transformation Language. In P. Barcelo and V. Tannen, editors, *Proc. of the 5th Alberto Mendelzon Intl. Workshop on Foundations of Data Management*, volume 749 of *CEUR Workshop Proceedings*, Santiago, Chile, 2011. CEUR-WS.org. [46](#), [47](#), [59](#)
 - [171] A. Sanyal and S. Choudhury. An object-oriented conceptual level design for web data model. In *2009 Procs. of Intl. Donf. on Methods and Models in Computer Science (ICM2CS)*. IEEE, Dec. 2009. [19](#)
 - [172] K.-U. Sattler, I. Geist, and E. Schallehn. Concept-based querying in mediator systems. *The VLDB Journal*, 14(1):97–111, 2005. [6](#)
 - [173] A. Savinov. Concept-Oriented Model and Query Language. In L. Yan and Z. Ma, editors, *Advanced Database Query Systems*. IGI Global, 2011. [15](#)
 - [174] A. Shamir and A. Stolpnik. Interactive visual queries for multivariate graphs exploration. *Computers & Graphics*, 36(4), June 2012. [16](#), [41](#), [55](#), [58](#)
 - [175] Z. Shen, K.-L. Ma, and T. Eliassi-Rad. Visual analysis of large heterogeneous social networks by semantic and structural abstraction. *IEEE transactions on visualization and computer graphics*, 12(6):1427–39, Jan. 2006. [44](#)

-
- [176] L. Sheng, Z. Ozsoyoglu, and G. Ozsoyoglu. A graph query language and its query processing. In *Procs. 15th Intl. conf. on Data Engineering*. IEEE, 1999. [28](#)
 - [177] B. Shneiderman. Direct Manipulation: A Step Beyond Programming Languages. *Computer*, 16(8), Aug. 1983. [33](#)
 - [178] A. Silberschatz, H. F. Korth, and S. Sudarshan. Data models. *ACM Computing Surveys*, 28(1):105–108, Mar. 1996. [13](#)
 - [179] G. C. Simsion and G. C. Witt. *Data Modeling Essentials*. Morgan Kaufmann, 3rd revise edition, Nov. 2004. [13](#), [14](#)
 - [180] P. R. Smart, A. Russell, D. Braines, Y. Kalfoglou, J. Bao, and N. R. Shadbolt. A Visual Approach to Semantic Query Design Using a Web-Based Graphical Query Designer. In *Proc. of the 16th Intl. Conf. on Knowledge Engineering: Practice and Patterns*, volume 5268 of *LNCS*, Berlin, Heidelberg, 2008. Springer. [50](#), [51](#), [52](#)
 - [181] S. Sundaresan. Schema integration of distributed databases using hyper-graph data model. In *IRI 2005 IEEE Intl. Conf. on Information Reuse and Integration*, pages 548–553. IEEE, 2005. [6](#)
 - [182] J. Taubert, M. Hindle, A. Lysenko, J. Weile, J. Kohler, and C. J. Rawlings. Linking Life Sciences Data Using Graph-Based Mapping. In N. W. Paton, P. Missier, and C. Hedeler, editors, *Data Integration in the Life Sciences*, volume 5647 of *LNCS*. Springer Berlin Heidelberg, 2009. [6](#)
 - [183] N. Tausch, M. Philippse, and J. Adersberger. A statically typed query language for property graphs. In *Procs. of the 15th Symposium on Intl. Database Engineering & Applications IDEAS '11*, New York, USA, Sept. 2011. ACM Press. [16](#), [19](#)
 - [184] A. ter Hofstede, H. A. Proper, and T. van der Weide. Formal definition of a conceptual language for the description and manipulation of information models. *Journal Information Systems*, 18(7):489–523, 1993. [15](#)
 - [185] The Neo4j Team. *The Neo4j Manual v2.2.1*. NeoTechnologies, 2015. [19](#), [28](#), [157](#)
 - [186] H. Tian and R. Sunderraman. A Domain-Specific Conceptual Data Modeling and Querying Methodology. In *1st Intl. Conf. on Information Systems, Technology and Management*, Mar. 2007. [14](#)
 - [187] Y. Tian, J. M. Patel, V. Nair, S. Martini, and M. Kretzler. Periscope/GQ. *Procs. of the VLDB Endowment*, 1(2):1404–1407, Aug. 2008. [2](#), [8](#)
 - [188] C. Tominski, J. Abello, and H. Schumann. CGV An interactive graph visualization system. *Computers & Graphics*, 33(6):660–678, Dec. 2009. [42](#), [55](#), [62](#)

-
- [189] C. Tominski, J. Abello, F. van Ham, and H. Schumann. Fisheye Tree Views and Lenses for Graph Visualization. In *Tenth International Conference on Information Visualisation (IV'06)*, pages 17–24. IEEE, 2006. [40](#), [41](#), [185](#)
 - [190] H. Topi and V. Ramesh. Human factors research on data modeling: a review of prior research, and extended framework and future research directions. *Journal of Database Management*, 13(2):3–19, Apr. 2002. [13](#)
 - [191] Q. T. Tran and C.-Y. Chan. How to ConQuer why-not questions. In *Procs. of the 2010 Intl. Conf. on Management of Data - SIGMOD '10*, page 15, New York, New York, USA, June 2010. ACM Press. [38](#)
 - [192] T. Tullis and W. Albert. *Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics*. Morgan Kaufmann, 2010. [31](#), [32](#), [36](#), [132](#), [146](#)
 - [193] J. D. Ullman. Information Integration Using Logical Views. *Proc. of the 6th Int. Conf. on Database Theory ICDT'97*, 1186:19–40, 1997. [171](#)
 - [194] S. van den Elzen and J. J. van Wijk. Multivariate Network Exploration and Presentation: From Detail to Overview via Selections and Aggregations. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2310–2319, Dec. 2014. [41](#), [42](#)
 - [195] Wasserman, Stanley, Faust, and Katherine. *Análisis de redes sociales. Métodos y aplicaciones*. CIS- Centro de Investigaciones Sociológicas, 2013. [22](#)
 - [196] G. Wiederhold. Mediators in the architecture of future information systems. *Computer*, 25(3):38–49, 1992. [171](#)
 - [197] P. T. Wood. Query languages for graph databases. *SIGMOD Record*, 41(1), 2012. [17](#), [21](#), [22](#)
 - [198] Z. Xu, Y. Ke, Y. Wang, H. Cheng, and J. Cheng. A model-based approach to attributed graph clustering. In *Procs. of the 2012 Intl. Conf. on Management of Data SIGMOD '12*, New York, USA, 2012. ACM Press. [16](#), [18](#), [19](#)
 - [199] S. Yamini and A. Gupta. Spatiotemporal Annotation Graph (STAG): A Data Model for Composite Digital Objects. In *21st Intl. Conf. on Data Engineering (ICDE'05)*. IEEE, 2005. [19](#), [20](#)
 - [200] D. Yang, L. Li, and L. Sun. Layered graph Data Model for dataspaces management. In *2011 IEEE 3rd Intl. Conf. on Communication Software and Networks*. IEEE, 2011. [18](#)
 - [201] H. Yang, R. Sunderraman, and H. Tian. bcnQL: A Query Language for Biochemical Network. In *2008 IEEE Intl. conf. on Bioinformatics and Biomedicine*. IEEE, 2008. [29](#)

-
- [202] D. Young and B. Shneiderman. A Graphical Filter/Flow Representation of Boolean Queries: A prototype implementation and evaluation. *Journal of the American Society for Information Science*, 44(6):327–339, July 1993. [52](#)
 - [203] G. Zenz, X. Zhou, E. Minack, W. Siberski, and W. Nejdl. From keywords to semantic queries: Incremental query construction on the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3), Sept. 2009. [38](#), [39](#)
 - [204] M. M. Zloof. Query-by-Example: A data base language. *IBM Systems Journal*, 16(4):324–343, Dec. 1977. [40](#)

Anexo A. Resumen entrevistas preliminares

Este anexo presenta un resumen de cada una de las cuatro entrevistas preliminares realizadas a profesionales de la salud. El objetivo de estas entrevistas era conocer cuál es la información que se registra en la historia clínica de los pacientes, cuál información necesita consultar un médico cuando está atendiendo un paciente y en qué situaciones es útil una herramienta para formular consultas sobre datos clínicos.

La metodología utilizada en las entrevistas y el análisis de los resultados se describen en la Sección 6.1. En la sección inicial de este anexo se presenta un resumen de los datos que comúnmente se encuentran en la historia clínica de los pacientes. En el Anexo B se recopilan los ejemplos de consultas que los médicos dieron durante las diferentes pruebas y los escenarios donde estas consultas son útiles. Estos items, por tanto, no se incluyen en este anexo.

¿Qué contiene la historia clínica?

A partir de los datos que, de acuerdo con Rodriguez y Rodriguez¹ contiene la historia clínica y con base en la información recolectada en las entrevistas preliminares a médicos y especialistas, se identificaron los siguientes datos relevantes.

- Datos de identificación y datos demográficos del paciente: nombre, apellidos, género, fecha de nacimiento, dirección, grupo étnico, estado civil, origen (lugar de nacimiento), procedencia (de donde viene —ciudad, barrio), escolaridad, ocupación, servicio de salud (EPS, prepagada, ...), acompañante (si el paciente es menor de edad).
- Motivo de la consulta: se registra con las palabras que usa el paciente.
- Enfermedad actual: narración clara, completa y cronológica del comienzo y evolución de la enfermedad. Es guiada por el médico. Los síntomas principales se detallan según el esquema: aparición (fecha y forma), localización e irradiación (en caso de referirse como síntoma el dolor), calidad o carácter (sensación peculiar del síntoma), intensidad (ligera, moderada, severa), factores que se relacionan con el aumento o el alivio (con sustancias o circunstancias), frecuencia (periodicidad, ritmo y horario), duración (en el tiempo), evolución y síntomas acompañantes o asociados (síntomas que poseen íntima o simultánea presencia).
- Antecedentes personales: enfermedades pediátricas, de la adultez y mentales (antecedentes patológicos personales), alergias e intolerancias, inmunizaciones, operaciones, traumas, hospitalizaciones previas, terapéuticas habituales (automedicación, prótesis, etc.), pruebas médicas anteriores, donaciones de sangre y transfusiones, historia gineco-obstétrica (menarquia, fórmula menstrual, menopausia, fecha de la última menstruación, embarazos, partos, abortos y sus causas, complicaciones durante el embarazo, anticonceptivos, prueba citológica) y hábitos (tóxicos, dietéticos, de sueño y de ejercicio).
- Antecedentes familiares: edad, sexo, enfermedades y causas de muerte de padre, madre, hermanos, cónyuge e hijos. En ocasiones de otros familiares (abuelos, tíos, primos).
- Revisión física: En Pediatría: peso, talla, perímetrocefálico, temperatura. En adultos:

¹Rodriguez García P.L., Rodriguez Pupo L., Principios técnicos para realizar la anamnesis de un paciente adulto. Rev. Cubana Medicina Gen. Integral 1999;15(4):409-14. En: http://bvs.sld.cu/revistas/mgi/vol15_4_99/mgi11499.htm

temperatura, frecuencia cardíaca, frecuencia respiratoria.

- Revisión de Sistemas: órganos de los sentidos, cabeza, cuello, corazón, pulmones, (no hace parte de la historia clínica de pediatría).
- Impresión diagnóstica: es un diagnóstico no confirmado (tiene un 90 a 95% de confianza).
- Orden de exámenes
- Resultado de exámenes
- Diagnóstico: Se basa en el CIE10 (Clasificación Internacional de Enfermedades). Se ordena del diagnóstico principal al menos importante.
- Tratamiento
- Incapacidades

Siendo esta una descripción genérica, es importante notar que existen diferencias en la historia clínica, según la especialidad del médico y la edad del paciente, por ejemplo, son diferentes las historias clínicas de los recién nacidos, los niños y los adultos. Adicionalmente, cuando un médico atiende por primera vez a un paciente, le solicita todos los datos de la historia clínica para empezar a formar su propio juicio de valor con relación al diagnóstico.

Entrevista al primer participante

Los siguientes items resumen las respuestas y comentarios del participante, en los que dió información adicional a los datos que tiene la historia clínica, las consultas sobre datos clínicos que puede requerir un médico y los posibles escenarios donde requiere realizar esas consultas.

- Existen diferencias entre la consulta de urgencias y la consulta externa. Durante esta última se tiene más tiempo para registrar información en la historia clínica.
- Los datos demográficos del paciente orientan la búsqueda de la patología y son importantes para la vigilancia epidemiológica.
- Las descripciones de consultas pasadas pueden ser indicador de diagnóstico de las nuevas consultas. Ej. “A los 2 años estuvo amarillo”.
- En algunas ocasiones se revisan los diagnósticos anteriores, por ejemplo, saber qué dijo el reumatólogo en un momento determinado. En estos casos puede ser mejor “preguntar” al sistema y no al paciente.
- En Colombia, por ley, en el registro de información clínica se usan los estándares de la OMS. Sin embargo, los estándares y esquemas de vacunación cambian con el paso del tiempo. Por tanto, el médico debe tener en cuenta las fechas en que se realizaron las anotaciones o en que se dieron los resultados, para adaptarse al estándar vigente en ese momento.
- En las historias clínicas se encuentran expresiones negadas, por ejemplo, una anotación “no ictericia”.
- En pediatría es importante tener información de la historia clínica de los padres (datos demográficos y clínicos), estos datos pueden cambiar completamente el diagnóstico. Sin embargo, estos datos pueden no estar disponibles, por ejemplo, cuando los padres no asisten a la consulta u olvidan mencionar sus enfermedades y tratamientos.
- Las consultas pueden servir para:
 - Hacer evaluación epidemiológica (ej. número de casos de cierta patología)

-
- Revisar, selectivamente, los conceptos de otros especialistas
 - Acceder a exámenes anteriores y sus resultados, y compararlos. O simplemente saber si el paciente hizo los exámenes.
 - Obtener estadísticas
 - Caracterizar pacientes. Esto es importante para los médicos y es útil como insumo a trabajos académicos de los especialistas en formación (tesis de grado).
 - La consulta de imágenes puede ser útil. Las imágenes son necesarias tanto a nivel de publicaciones como para recuperar datos de otros pacientes.

Entrevista al segundo participante

Los siguientes items resumen las respuestas y comentarios del participante, en los que dió información adicional a los datos que tiene la historia clínica, las consultas sobre datos clínicos que puede requerir un médico y los posibles escenarios donde requiere realizar esas consultas.

- “Se requiere una historia clínica lo más maleable posible”, para evitar dificultades que hoy se presentan. Por ejemplo: el médico durante la consulta se dedica a escribir (hay que llenar muchos datos) y casi no mira al paciente, esto dificulta crear una relación con el paciente.
- En las citas de control, es necesario tener acceso a toda la información del paciente (antecedentes, imágenes, resultados de exámenes), de esta forma el control puede hacerse más rápido.
- En algunas especialidades, ej. oftalmología, el especialista no diligencia de forma completa la historia clínica, sin embargo, el especialista requiere, y debe, mirar la historia previa para informarse sobre su paciente de manera general.
- Generalmente en las instituciones de salud, los médicos pueden acceder a la historia de cualquier paciente, sin embargo, solo la pueden modificar durante la consulta.
- Desde el punto de vista epidemiológico, se pueden tener consultas para generar alertas (ej. brote de una enfermedad infecciosa). Si este tipo de información se puede recoger de una red, se crea un sistema de vigilancia.
- Para los administradores, gerentes médicos, investigadores, y coordinadores de programas de salud es importante contar con estadísticas (ej. ¿cuántos hemogramas se pidieron en el mes?, ¿cuántas embarazadas tuvieron diagnóstico de hipertensión en un año dado?) y consolidar información de diagnósticos o de resultados de exámenes.

Entrevista al tercer participante

Los siguientes items resumen las respuestas y comentarios del participante, en los que dió información adicional a los datos que tiene la historia clínica, las consultas sobre datos clínicos que puede requerir un médico y los posibles escenarios donde requiere realizar esas consultas.

- Para algunos especialistas es importante tener acceso a los resultados dados por otros especialistas, revisar lo que registró el médico general, o lo que el mismo médico registró en consultas anteriores.
- Puede ser útil acceder a la información clínica de la familia del paciente (ej. padre, madre, hermanos) para confirmar antecedentes que van asociados a cierto tipo de enfermedades.

Entrevista la cuarto participante

Se centró en ejemplos de preguntas que se harían mediante un sistema de consulta.

Anexo B. Escenarios y ejemplos de consultas

En este anexo se recopilan los escenarios, mencionados por los participantes en las pruebas, en los que puede ser útil una herramienta de consulta orientada al usuario final. Además, se relacionan las consultas que los participantes dieron como ejemplo en las pruebas realizadas.

Escenarios

- Consulta externa y citas de control
- Seguimiento de la evolución de los pacientes
- Investigación de casos clínicos
- Procesos de enseñanza-aprendizaje
- Planeación de programas de prevención
- Evaluación y vigilancia epidemiológica
- Administración de los servicios de salud

Ejemplos de consultas

Consulta	Elementos que requiere (*)
Procedimientos que se le han realizado al paciente X	PM, 1 cond.
Exámenes médicos realizados al paciente X desde 2005 y sus resultados	PM, 2 cond.
Medicamentos formulados por el médico Z al paciente X y fechas en que se formularon	PM, 2 cond, OPT
Diagnósticos que el paciente X recibió de médicos especialistas en Y y los nombres de esos médicos	PM, 2 cond, OPT
Miembros de la familia del paciente X diagnosticados con la enfermedad Q	PM, 1 cond
Fechas en que el paciente X tuvo el diagnóstico “Cáncer de Pulmón” y tratamientos que ha recibido desde entonces	PM, 2 cond, OPT
Diagnósticos que el paciente X recibió de médicos con una especialidad particular (ej. neurólogo, reumatólogo) y tratamientos que le recomendaron esos médicos	PM, 2 cond, OPT
Diagnósticos y tratamientos que le dio (el mismo médico que está atendiendo la consulta) al paciente X en consultas anteriores	PM, 2 cond, OPT
Registros en la información clínica de la familia del paciente (padre, madre, hermanos) sobre una enfermedad	PM, 3 cond, OPT

(*) PM (*pattern matching*), cond (*condición*), OPT (*optional*)

Consulta	Elementos que requiere
Enfermedades que ha padecido el paciente X (o cirugías, o informes de patología, o transfusiones, o medicamentos, o traumas) y fecha en que se registró	PM, 1 cond, OPT
Datos demográficos del paciente X	PM, 1 cond, OPT
Datos demográficos de los padres del paciente X	PM, 2 cond, OPT
Dosis de un medicamento administrado al paciente X, quien lo recetó y que formación tiene la persona que lo recetó (especialista, estudiante residente, subespecialista)	PM, 2 cond, OPT
Medicamentos han sido formulados a pacientes con diagnóstico de R y síntomas presentaron estos pacientes cuando se formularon esos medicamentos	PM, 1 cond, OPT
Historia clínica de los pacientes con resultado X en un examen diagnóstico Y (ej. hemograma)	PM, 1 cond, OPT
Historia clínica de los pacientes con diagnóstico de Y que viven en la comuna Z	PM, 2 cond, OPT
Imágenes de estudios diagnósticos que tienen un resultado P y que fueron tomadas a pacientes de género femenino	PM, 2 cond
Resultados del examen diagnóstico Q realizados en los últimos 5 años, y datos demográficos de los pacientes a los que se realizó la prueba	PM, 2 cond, OPT
Historia clínica de los pacientes de género femenino que durante el año 2013 estuvieron hospitalizados por un diagnóstico de enfermedad por reflujo gastroesofágico	PM, 4 cond, OPT
Historia clínica de recién nacidos, hijos de mujeres que durante el embarazo tuvieron diagnóstico de hipertensión. Resultados de los exámenes que se hicieron a esos niños	PM, 2 cond, OPT

Consulta	Elementos que requiere
De los pacientes que durante el año 2013 estuvieron hospitalizados en la sala de pediatría general, de genero femenino, y tuvieron el diagnóstico de enfermedad por reflujo gastroesofágico: Cuántos se fueron para la casa, cuántos se quedaron en la UCI y cuántos fallecieron.	Func. agregadas
Cuántos niños se hospitalizaron por reflujo gastroesofágico durante los últimos 10 años	Func. agregadas
Cuántos niños se hospitalizaron en la sala de pediatría general en un rango de fechas dado. Cuántos de ellos eran del género masculino y cuántos del género femenino. Porcentaje de estos niños que fueron atendidos por el Dr. X	Func. agregadas
Cuartil de la población que mas se enferma de cáncer según el barrio o la raza	Func. Agregadas
Frecuencias relativas y acumuladas para caracterización de pacientes	Func. agregadas
Frecuencias, modas, promedios, intervalos, cuartiles	Func. Agregadas
Cruces de variables. Ej. pacientes con diagnóstico de diabetes: cuantos hombres/mujeres, cuantos por grupos de edad	Func. agregadas, agrupamiento
Resultados de los exámenes que se hicieron a bebés, hijos de mujeres que durante el embarazo tuvieron diagnóstico de hipertensión. Información fragmentada por fechas o rangos de tiempo	Func. agregadas, agrupamiento
Diagnósticos dados a pacientes con síntomas similares a los del paciente X	Op. similitud
Medicinas ordenadas a pacientes con diagnóstico Y, y síntomas similares a los del paciente X	Op. similitud
Una persona tiene cáncer de colon y su mamá también lo tuvo. Ver las diferencias o la similitudes comparando variables claves de la enfermedad (ej. edad, hábitos alimenticios, los síntomas). Las variables clave son diferentes de una enfermedad a otra	Op. para comparar variables clave

Anexo C. Plan de la prueba exploratoria

Plan de la Prueba Exploratoria

El propósito de esta prueba es conocer la opinión del usuario sobre el diseño de una herramienta de consulta de datos. Este diseño es una etapa preliminar de la construcción de la herramienta. Los moderadores de la prueba le presentarán los conceptos sobre los que se basa el diseño y le harán preguntas para conocer su opinión al respecto. No hay respuestas correctas o incorrectas, todo lo que usted opine nos ayudará enormemente a mejorar el diseño de la herramienta. Sus preguntas y comentarios son bienvenidos en cualquier momento. Los ejemplos están basados en datos de historias clínicas de pacientes.

Datos del entrevistado:

- Profesión
- Especialidad/Maestría o Doctorado en

1. Grafos y representación de datos

Entre otras funciones, los Sistemas de Información permiten almacenar datos en un computador. Por ejemplo, almacenar los datos de las historias clínicas de los pacientes de una entidad prestadora de servicios de salud. Esos datos son una fuente de información valiosa, por ello interesa también recuperar partes de esos datos que responden a preguntas particulares. Por ejemplo, ¿Qué pacientes fueron atendidos en consulta el día de ayer?

Para almacenar datos en un computador, estos deben estar organizados de alguna manera. Los grafos se pueden usar para organizar datos. Un grafo está formado por nodos (cuadrados y círculos en la Figura 1) y arcos (líneas que unen los nodos). Para organizar datos en un grafo, agregamos etiquetas en los nodos y los arcos, como se observa en las Figuras 2 y 3.

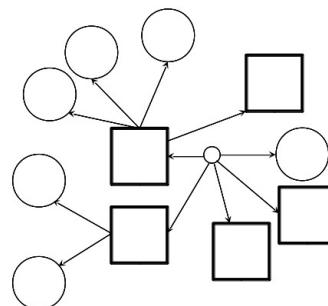


Figura 1. Un grafo.

2. Los conceptos generales y los hechos particulares

En la representación que usamos para este desarrollo, un grafo describe de manera general cómo están organizados los datos y otro grafo representa los datos en sí. Por ejemplo la Figura 2 muestra, de manera general, que se almacenan datos de

Pacientes. De cada paciente se tiene su número de identificación, nombre, apellido, y como datos demográficos: género, fecha de nacimiento, teléfono y ciudad.

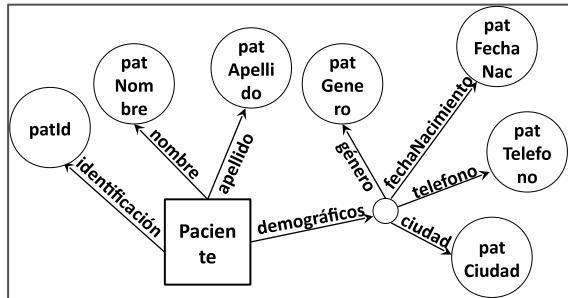


Figura 2. Organización general.

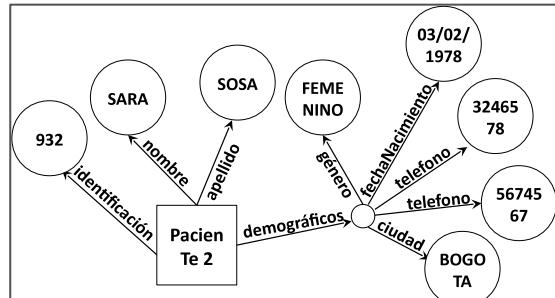


Figura 3. Un paciente en particular.

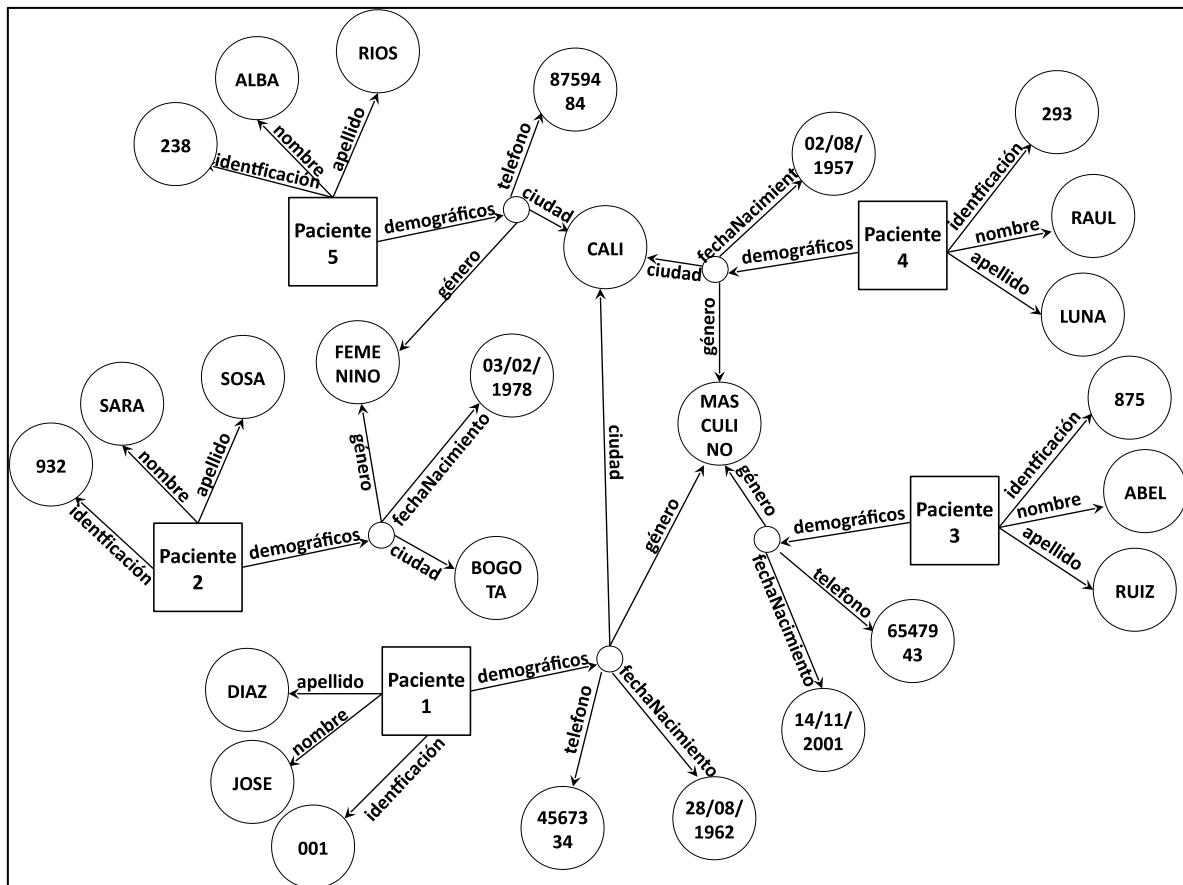


Figura 4. Grafo de Datos.

La Figura 3 muestra un grafo que representa los datos de un paciente en particular, cuyo nombre es SARA SOSA, su identificación es 932, de sexo FEMENINO, nacida el 03/02/1978, tiene registrados dos números de teléfono 3246578 y 5674567, y su ciudad es BOGOTÁ. Se observa que los datos en sí (fechas, nombres, números...) están en los nodos pintados como círculos grandes. Los nodos cuadrados y los círculos pequeños sirven para organizar esos datos y darles sentido. Ellos representan personas, objetos o eventos. En algunos casos los círculos pequeños se usan solamente para agrupar datos (por ejemplo el círculo a donde llega la flecha de datos demográficos del paciente).

En la figura 4 se muestran los datos de varios pacientes, organizados en la forma descrita en la Figura 2. En algunos casos los datos de un paciente pueden estar incompletos. Por ejemplo, no se conoce la fecha de nacimiento del paciente 5 o la ciudad del paciente 3. Además, en esta figura se observa que algunos datos (por ejemplo el género y la ciudad) son comunes en varios pacientes y unen sus grafos (Figura 4).

La herramienta que se propone pretende tomar ventaja de estas conexiones, para permitir al médico encontrar relaciones entre los pacientes. Para ello se va a transformar el grafo de manera que se tenga solamente la información de interés, de manera que se facilite la observación de las relaciones entre los datos.

Las Figuras 5 y 6 muestran un grafo con más datos de pacientes. La Figura 5 muestra el grafo con la organización general y la Figura 6 un grafo con datos particulares (los puntos suspensivos indican que hay más datos que no se representaron en la figura por efectos de espacio).

Preguntas sobre las Figuras 5 y 6:

- ¿Qué datos se tienen de cada paciente?
- ¿Qué datos se tienen del paciente 1 en particular?

Acciones de Moderador:

- Registrar las respuestas
- Aclarar dudas e interpretaciones que no sean adecuadas

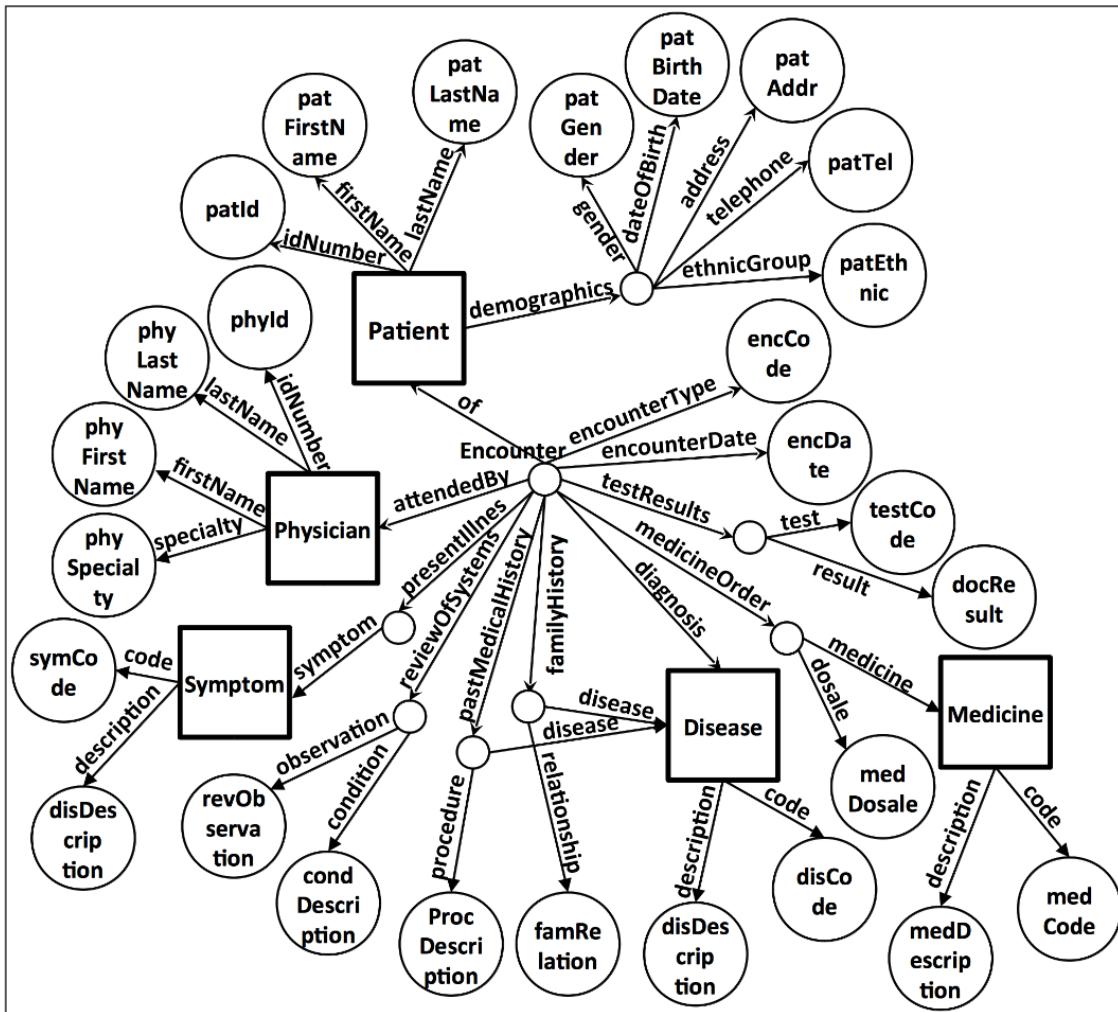


Figura 5. Organización general de los datos.

3. Transformación del grafo

Cuando se necesita recuperar información almacenada, generalmente se requieren partes específicas de los datos. Para recuperar esas partes se usan herramientas de consulta de datos.

En el caso de los grafos, una manera de encontrar esos datos es transformar el grafo que muestra la organización general, reduciéndolo para dejar solamente los datos que nos interesan. Una porción del grafo de la Figura 5 se muestra en la Figura 7. Este último tiene solamente la identificación, nombre y apellido del paciente y los diagnósticos que ha recibido.

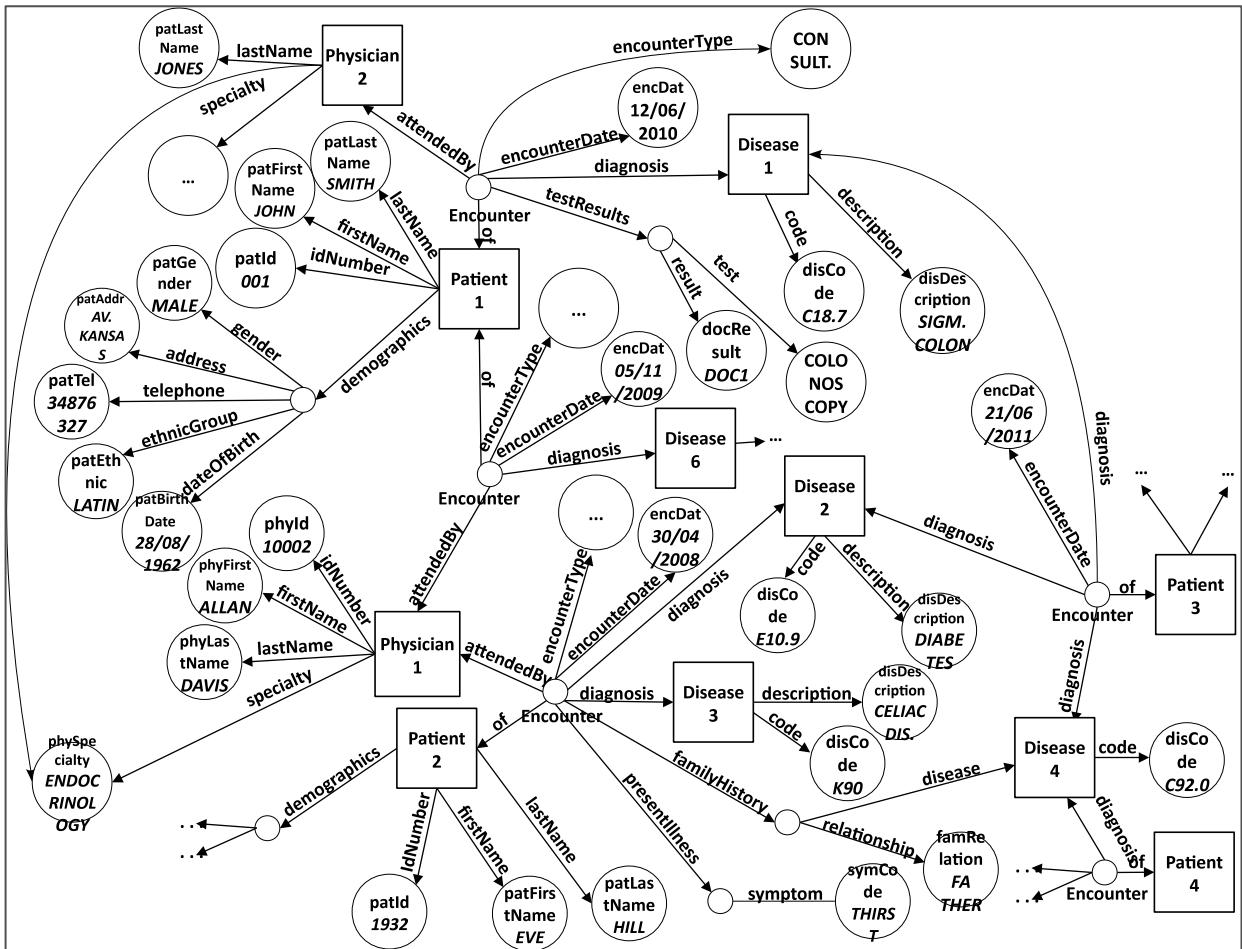


Figura 6. Datos de pacientes.

La Figura 8 muestra una transformación del grafo de la Figura 7, en ésta se acerca el paciente con el nombre de la enfermedad quitando otros datos que no interesan. Luego, el grafo de datos que corresponde con esta descripción es más pequeño (ver Figura 9).

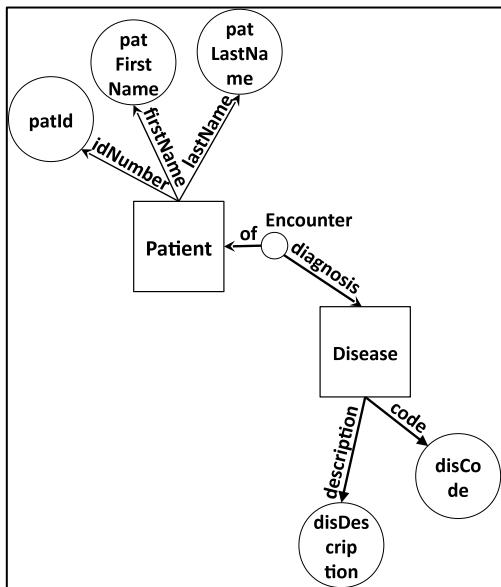


Figura 7. Grafo en que se "acercan" Paciente y Enfermedad.

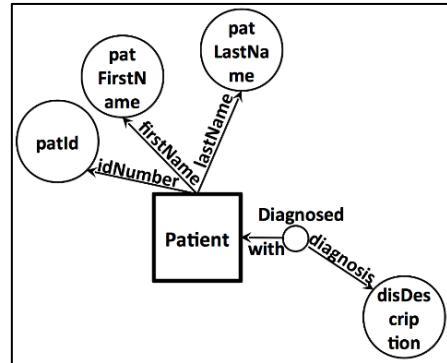


Figura 8. Grafo con datos de pacientes y diagnósticos.

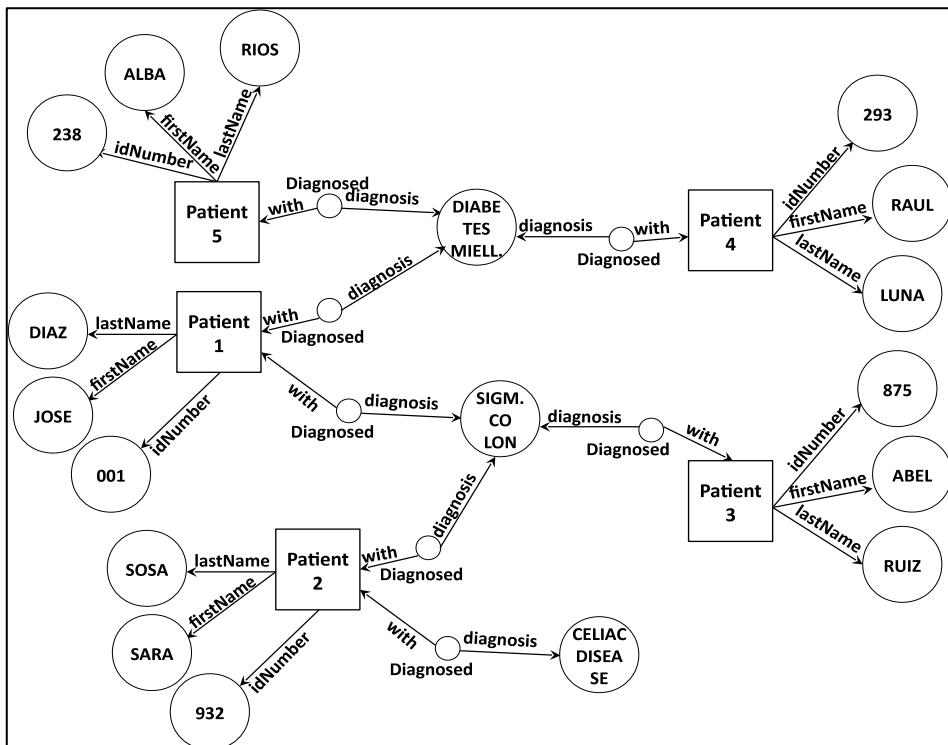
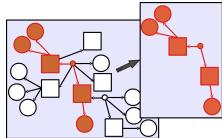
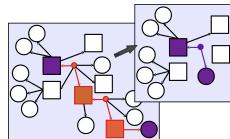


Figura 9. Diagnósticos dados a pacientes.

Para lograr estas transformaciones la herramienta ofrece varias opciones:

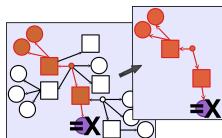


Marcar una porción del grafo que tiene los datos que interesan. Por ejemplo, el grafo de la Figura 7 es una porción del grafo de la Figura 5.



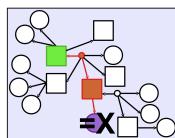
Acercar dos nodos en el grafo reemplazando el camino que los une. En este caso no hay interés en los datos de los nodos que se reemplazan, éstos desaparecen del grafo.

Por ejemplo, el grafo de la Figura 7 se transforma para acercar los nodos *Patient* y *disDescription*.



Tomar una porción del grafo pero solamente aquellos datos que están conectados a un valor en particular.

Por ejemplo, nombres y apellidos de pacientes que tienen un diagnóstico de diabetes. La Figura 10 muestra el grafo general resultado de esta consulta y la Figura 11 un ejemplo de los datos.



Tomar toda la información relacionada con objetos o eventos que tienen una característica en particular. Por ejemplo, los datos de los pacientes que tienen un diagnóstico de “Diabetes”. El grafo no cambia, solo se toman algunos datos.

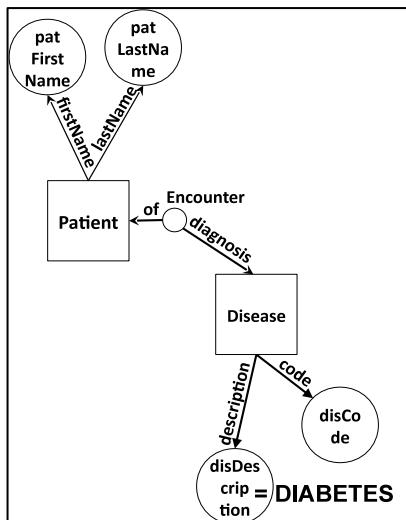


Figura 10. Datos conectados a un valor particular.

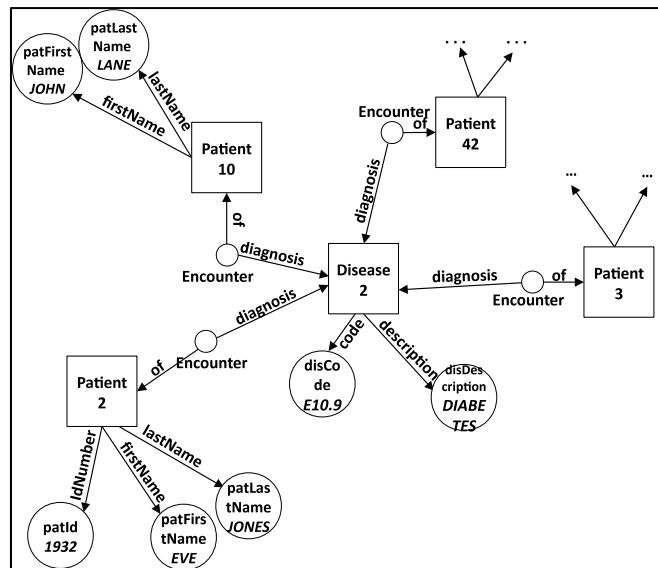
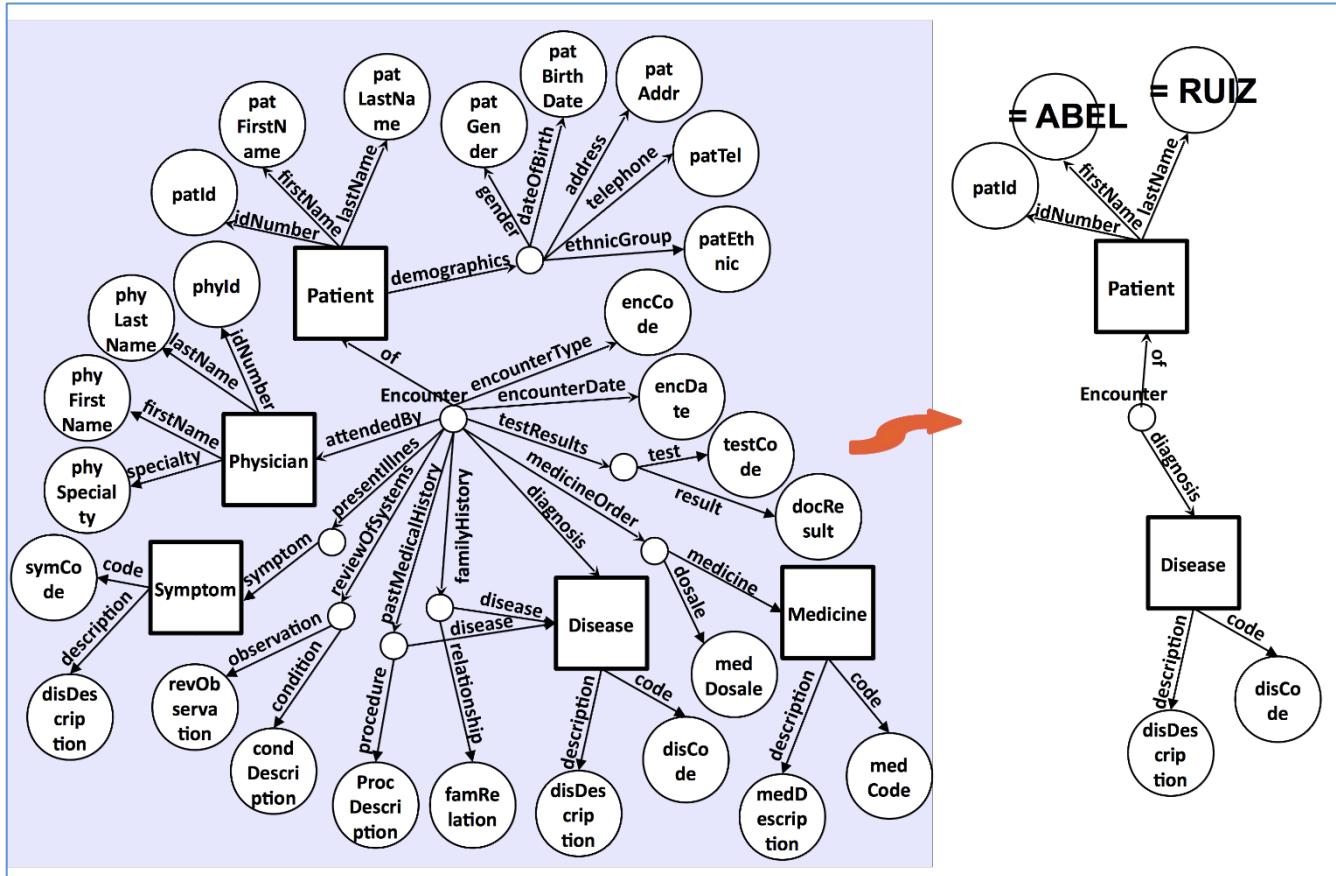


Figura 11. Datos.

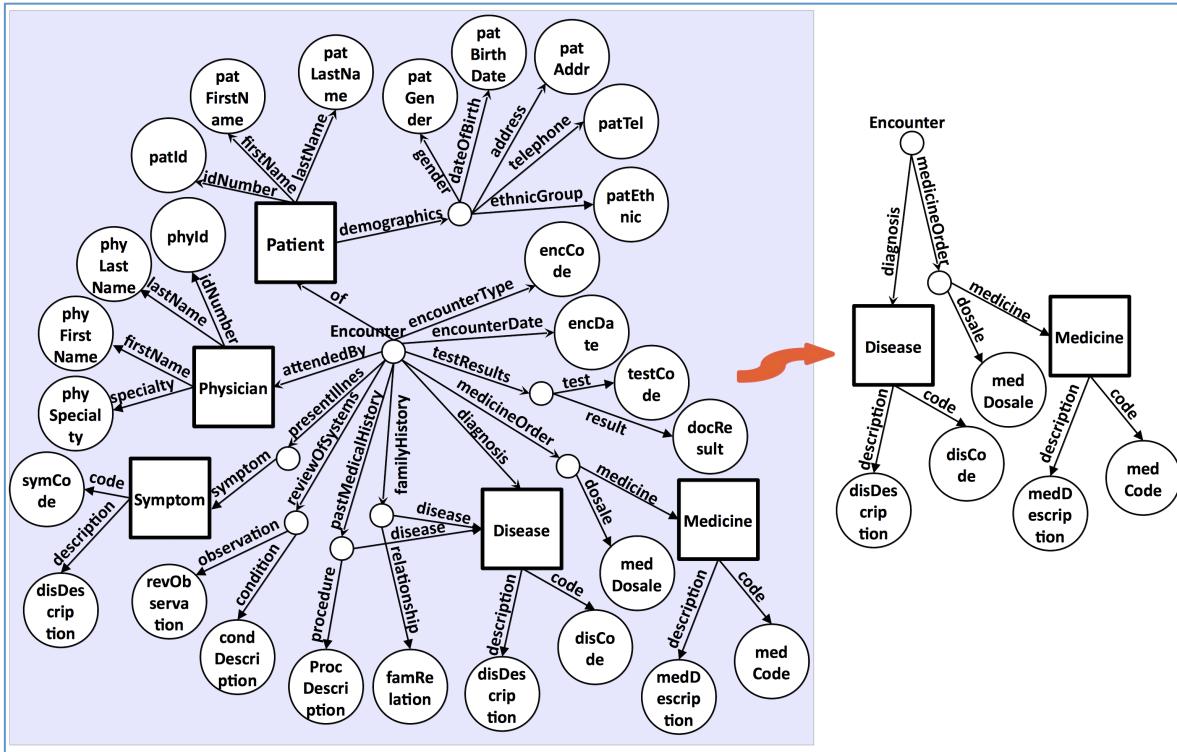
4. Preguntas sobre Transformación de Grafos

Cada ejemplo consta de un grafo inicial y uno final. El moderador muestra cada ejemplo al usuario y le pide que identifique la transformación que se realizó y explique qué datos representa el grafo transformado.

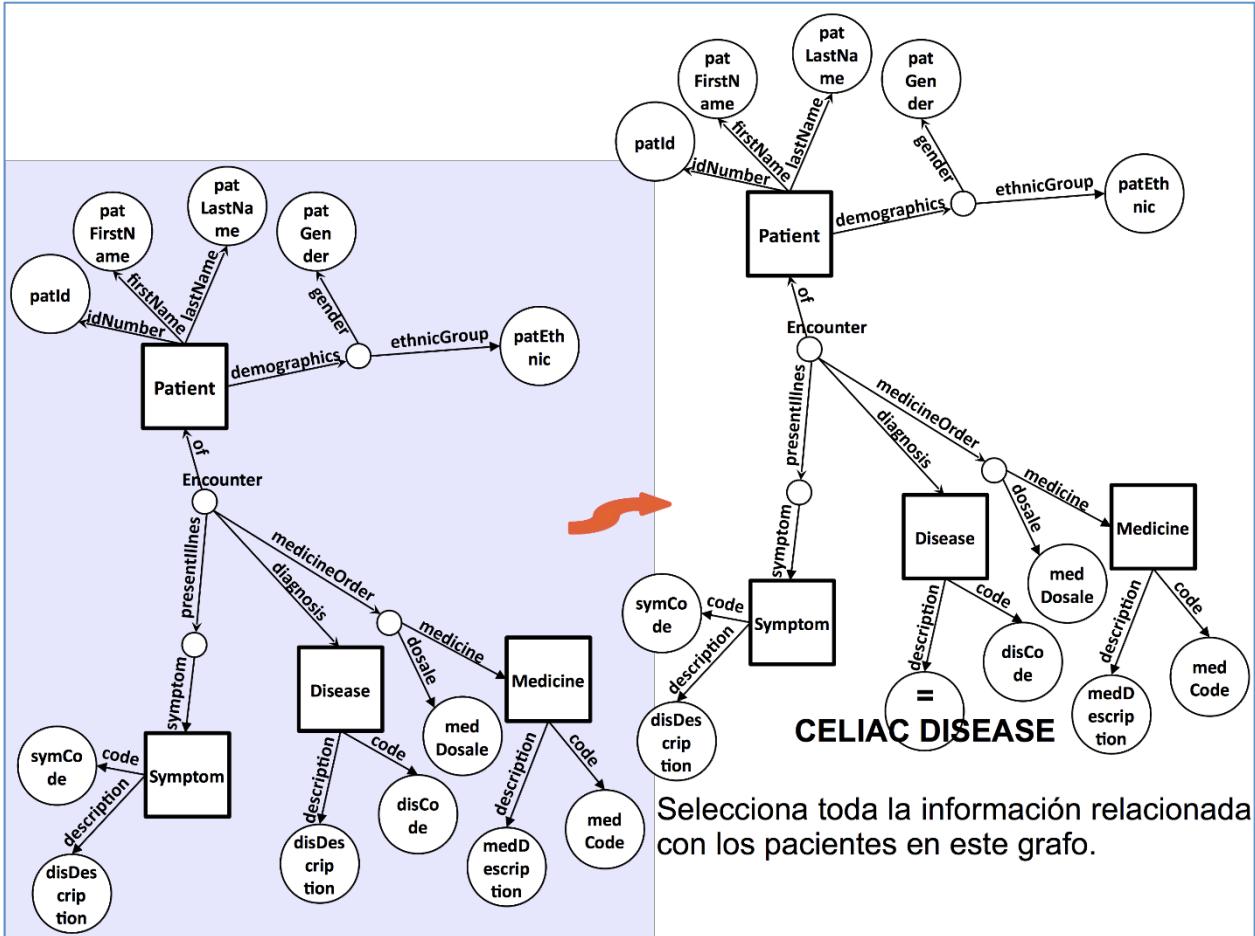
Primer ejemplo:



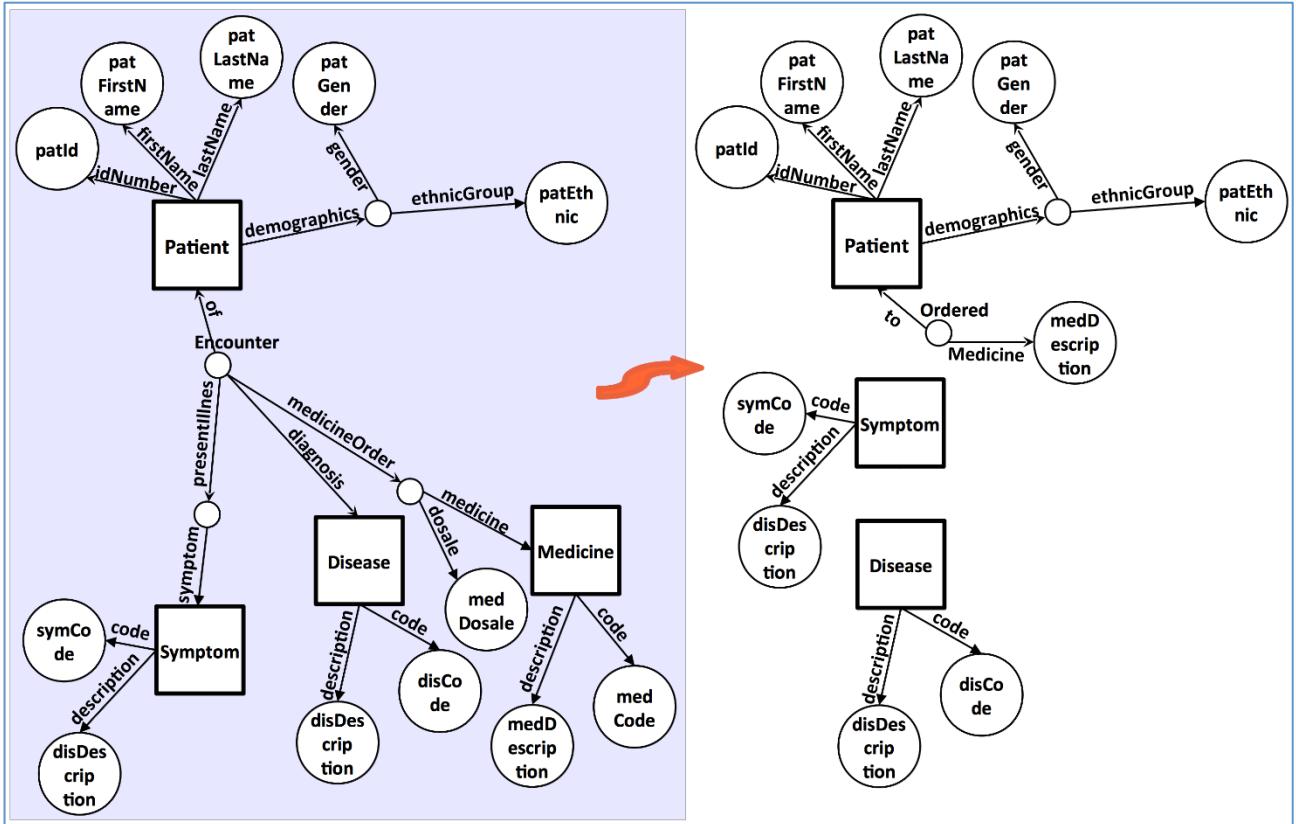
Segundo ejemplo:



Tercer ejemplo:



Cuarto ejemplo:



Preguntas:

1. ¿Alguna parte de las gráficas le parecen confusas? ¿Cuáles? ¿Cómo se podrían mejorar?
2. ¿Los dibujos que usados al lado de los textos que describen las transformaciones son útiles para dar una idea de lo cómo se transforma el grafo? ¿Si estos dibujos se usaran en la pantalla de la aplicación le recordarían al usuario cual es la transformación que ellos representan?
3. ¿Son claros los textos que describen las transformaciones del grafo? ¿Qué hace falta en esas descripciones?
4. ¿La forma de escribir las condiciones (por ejemplo " = ABEL", " = RUIZ" es clara? ¿Preferiría otra manera, por ejemplo la que se muestra en la siguiente figura?

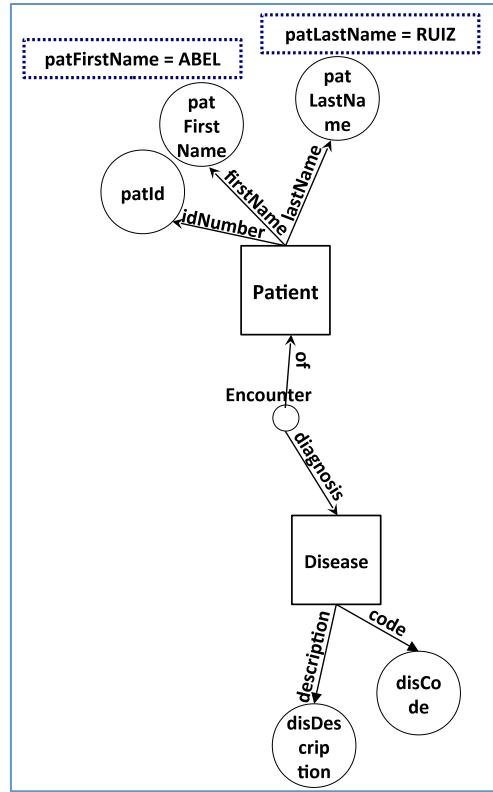


Figura 12. Segunda opción para presentación de filtros.

Anexo D. Resumen de la aplicación de la prueba exploratoria

Este anexo presenta un resumen de la aplicación de la prueba exploratoria. Los ejemplos de consultas que los participantes mencionaron durante el desarrollo de estas pruebas se incluyeron en el [Anexo B](#). Cuando se realizó esta prueba se tenía la definición del *Value Filter* (versión anterior del *Filter*) que se describió en la Sección [4.6](#). Adicionalmente, la definición de ese momento del operador *Put two objects closer* podía generar un resultado con grafos desconectados.

Primer Participante

Leyó la información que le brinda el grafo esquema. No le fue claro el rol del nodo “Encounter”, el moderador lo explicó. El participante identificó que en el grafo instancia del ejemplo hay representados varios pacientes con sus respectivos datos y leyó los datos de los pacientes 1 y 2.

En la actividad de identificación de los operadores aplicados, para el primer ejemplo señaló el tercer ícono (*Value Filter*), para el segundo ejemplo eligió el primer ícono (*Select a Portion*), para el tercer ejemplo señaló el cuarto ícono (*Class Filter*) y comentó que se parecía al primer ejemplo. En el cuarto ejemplo, el *Put Two Objects Closer* generó un resultado con grafos desconectados. Esto confundió al participante porque le pareció que había una mezcla de operaciones. El moderador aclaró que interesaba la porción del grafo donde se aplicó la contracción de caminos. Entonces, para el cuarto ejemplo eligió el segundo ícono (*Put Two Objects Closer*). Todos los ejemplos fueron identificados correctamente.

En las respuestas a las preguntas, el participante opinó que es fácil imaginar, a partir del esquema, los datos de la instancia. También resaltó la utilidad de poder realizar este tipo de operaciones sobre los datos. Opinó que los textos que explican lo que hace cada operador ayudan más que los iconos, aunque estos representan la operación que se va a realizar. Propuso un cambio para el texto del segundo ícono (*Put two objects closer*). Finalmente, consideró que era más clara para él la forma de visualización de las condiciones de filtro usadas en el ejemplo (dentro de los nodos).

Adicionalmente, durante cuando se explicaron los operadores, el participante preguntó si la herramienta daba la posibilidad de hacer algún tipo de procesamiento de la información. Por ejemplo, obtener indicadores (datos de resumen), como proporción de hombres y mujeres que cumplen la característica que se especificó.

Segundo Participante

Leyó la información que le brinda el grafo esquema e identificó en el grafo instancia del ejemplo los datos del paciente 1 y del paciente “Eve Hill”.

En la actividad de identificación de los operadores aplicados, para el primer ejemplo eligió el tercer ícono (*Value Filter*), para el segundo ejemplo eligió el primer ícono (*Select a Portion*), para el tercer ejemplo, después de un breve momento de duda pensando en elegir el tercer ícono, se decidió por el cuarto ícono (*Class Filter*), para el cuarto

ejemplo eligió el segundo ícono (*Put two objects closer*). Todos los ejemplos fueron identificados correctamente.

Con respecto a los iconos y textos que representan y describen los operadores, consideró que se entienden claramente.

Finalmente, consideró que era más clara para él la forma de visualización de las condiciones de filtro usadas en el ejemplo (dentro de los nodos).

Adicionalmente, cuando se le mostraron los grafos el participante resaltó la importancia de las variables demográficas en la caracterización de algunos tipos de enfermedad asociadas a grupos particulares de pacientes. Mencionó que los códigos de los síntomas y diagnósticos pueden ser códigos de la clasificación internacional de enfermedades o del DSM para las enfermedades mentales. También opinó que los grafos son fáciles de entender y que dan flexibilidad porque “amplifican el panorama con relación a los datos que tiene un paciente” y permiten buscar información muy concreta de un paciente o de un grupo de pacientes.

Tercer Participante

Leyó la información representada en el grafo esquema. Fue necesario que el moderador explicara el rol del nodo “*Encounter*”. El participante identificó los datos de los pacientes representados en el esquema y en el grafo instancia, los datos del paciente 2.

Una vez dada la explicación sobre los operadores, consideró que son claros y afirmó que “se entiende primero el ejercicio de simplificación del paciente y enfermedad. Es un ejercicio muy pertinente para los médicos porque en el ejercicio clínico uno no ve todo el tiempo todo sino que necesita lo concreto”. Además opina que “tener filtros y posibilidad de manejo y uso de la información es lo mas potente”. Opina que “paradójicamente” para su caso el más complejo de entender fue el primer operador (*Select a Portion*), y afirma que esto puede ser debido a que ese ícono tiene menos variaciones de color.

En la actividad de identificación de los operadores aplicados, para el primer ejemplo eligió el primer ícono (*Select a Portion*). El moderador hizo notar la condición de filtro, y el participante cambió la elección por *Value Filter*; para el segundo ejemplo eligió el segundo ícono (*Put Two Objects Closer*), para el tercer ejemplo, después de un breve momento de duda pensando en elegir el tercer ícono, se decidió por el cuarto ícono (*Class Filter*), para el cuarto ejemplo, después de dudar por un momento, eligió el segundo ícono (*Put Two Objects Closer*). Elegió de forma correcta los operadores del primer, tercer y cuarto ejemplos.

Finalmente, el participante consideró mejor la segunda opción propuesta para mostrar en el gráfico las condiciones de filtro.

Como respuesta a las preguntas, el participante opinó que “la propuesta es muy pertinente”. Afirmó que puede ser un reto grande porque muchos médicos, particularmente los mayores, no están familiarizados con la tecnología. Sugirió agregar

al grafo iconos amigables, por ejemplo, en el nodo que representa el paciente. Opinó que “las conexiones ayudan mucho a entender que estoy extractando” y que el mayor reto está en la representación visual, para que sea muy amigable. Adicionalmente, el participante sugirió cambiar el nombre del nodo “*Disease*” por “*Diagnosis*” porque el paciente puede estar sano (ej. en control de crecimiento y desarrollo, de embarazo con gestación sana, y en adulto mayor sano). En este sentido el CIE10 incluye diagnósticos para pacientes sanos. Aclaró que los tratamientos se clasifican en tres tipos: farmacológicos, no farmacológicos y quirúrgicos. También sugirió incluir en los datos la información del desenlace (ej. en hospitalización se llama epicrisis —murió, se remitió a otra unidad o se curó).

Anexo E. Diseño de la segunda prueba de evaluación

SEGUNDA PRUEBA DE EVALUACION DEL PROTOTIPO DE GRAPHTQL

Este documento contiene el diseño de la segunda prueba de evaluación del prototipo de GraphTQL, y consta de las siguientes secciones: objetivo de la prueba, preguntas que motivan este estudio, metodología (escenarios, actividades, métricas y perfil de los participantes) y plan de la prueba.

1. Objetivo

El objetivo de esta prueba es evaluar la facilidad de uso del prototipo de GraphTQL, pidiendo a los participantes formular cuatro consultas. La prueba se enfoca en la formulación de las consultas, no incluye la presentación de los resultados de las mismas.

2. Preguntas que motivan este estudio

Con este estudio se busca obtener indicios sobre la posible respuesta para cada una de las siguientes preguntas:

- ¿La visualización del grafo esquema facilita la formulación en la consulta?
- ¿Las operaciones de transformación de grafos facilitan realizar consultas que incluyen filtros y datos incompletos?
- ¿La guía para desambiguar las condiciones le ayuda al usuario a proponer consultas con filtros?

3. Metodología

La prueba incluye cuatro etapas. La primera es la introducción, donde se describe la prueba; la segunda es la presentación de la herramienta a evaluar, donde se enseña al participante como usar el lenguaje de consulta; la tercera es de evaluación, donde se pide al participante que realice cuatro consultas; y finalmente, en la etapa de cierre, donde se recogen las opiniones del participante. Durante las actividades de evaluación el usuario puede pedir la ayuda del moderador o acudir a las ayudas de la herramienta.

En la etapa de presentación de la herramienta, el moderador realiza varios ejemplos de consulta sobre un modelo con datos de películas y series de televisión (Figura 1). Para las actividades de evaluación se definió un conjunto de datos que representa información clínica de pacientes (Figura 2). Los modelos usados en estas dos etapas son diferentes porque una de las preguntas del estudio apunta a evaluar

si la representación del grafo esquema facilita la formulación de las consultas. Por lo tanto es importante observar qué ocurre cuando el participante debe empezar a usar un modelo de datos diferente.

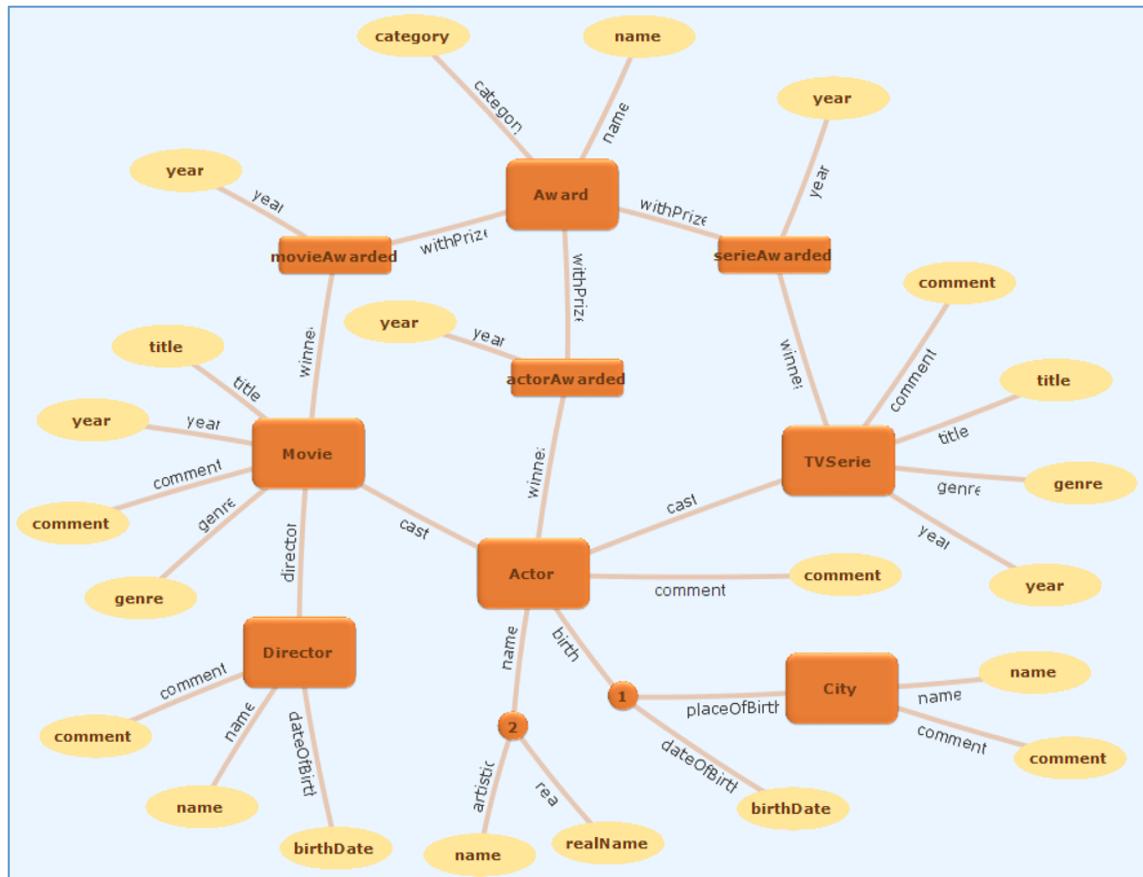


Figura 1. Esquema con datos de películas y series de televisión.

Escenarios, consultas, métricas y perfil de los participantes

Los escenarios, las consultas, las métricas y el perfil de los participantes son los mismos especificados en el diseño de la Prueba Comparativa (Anexo G). La única diferencia es que en esta versión del prototipo los operadores de filtro tenían nombre diferente: “*Portion with filter*” (para “*Filter*”) y “*Object with characteristic*” (para “*Filter+additional data*”).

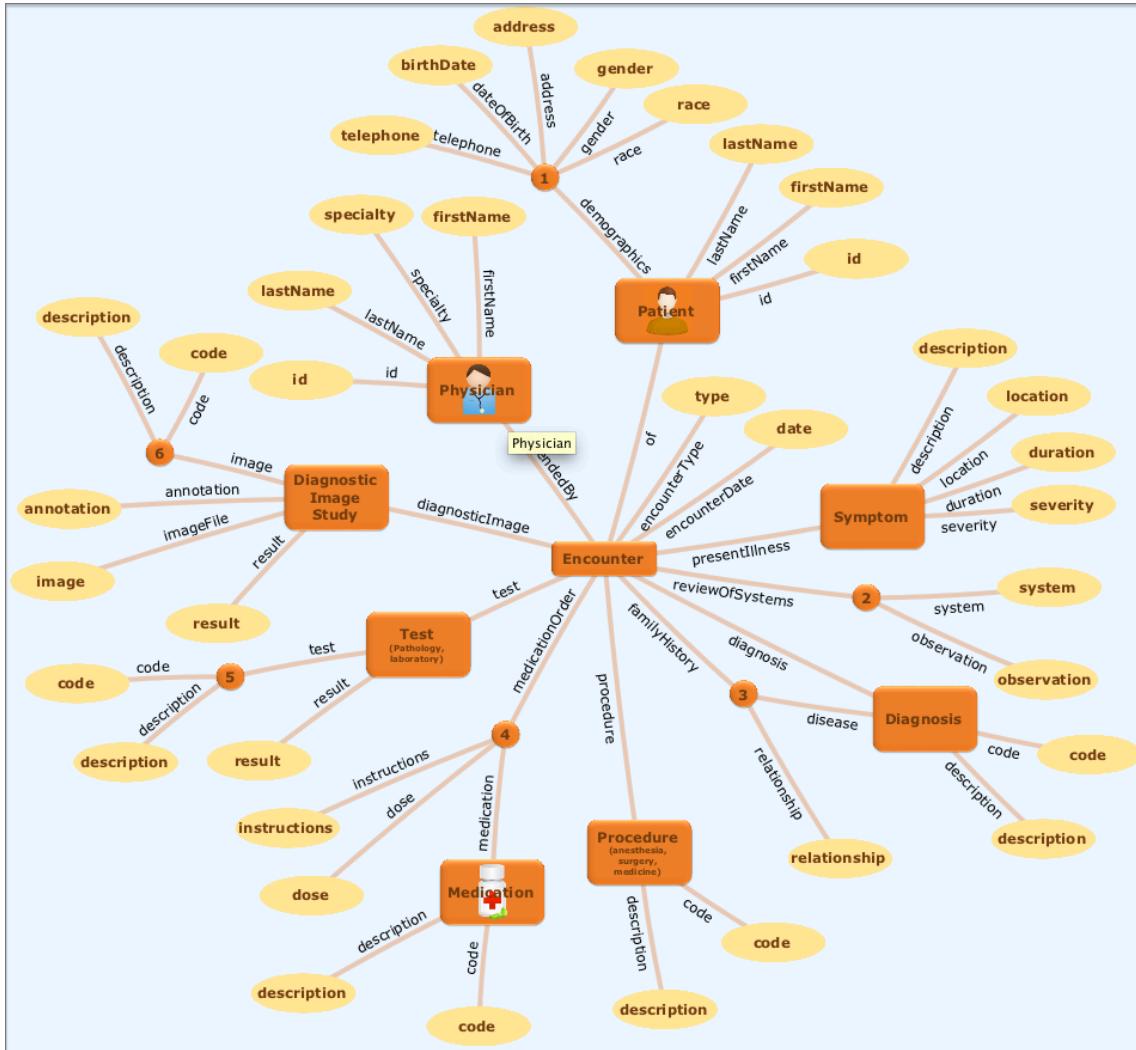


Figura 2. Esquema con datos de la historia clínica.

4. Plan de la Prueba

La prueba consta de las siguientes partes:

- **Apertura:** El moderador explica el propósito de la prueba y las partes de que consta. Se diligencia el cuestionario pre-test y el participante firma el formato de consentimiento.
- **Datos para la demostración de la herramienta.** El moderador explica al usuario el conjunto de datos que se va a usar para la presentación de la herramienta (datos de películas y series de televisión).
- **Presentación de la herramienta.** El moderador explica al usuario las opciones, botones y operaciones. El moderador desarrolla tres ejemplos.
- **Actividades.** Incluye:

-
- Presentación de los datos y escenarios: el moderador explica al participante los datos para las actividades y los dos escenarios en los que se desarrollan estas.
 - El moderador explica la técnica "Pensar en voz alta" y pide al participante que la aplique durante la prueba.
 - Evaluación: consta de 4 actividades, dos en el escenario I y dos en el escenario II. Los participantes contarán con un tiempo máximo de 15 minutos para realizar cada una de las actividades.

En cada actividad el moderador debe registrar:

- La hora de inicio y finalización.
- El número de veces que el usuario solicita las diferentes clases de ayuda.
- El número de errores que ocurren durante la realización de la actividad.

Al finalizar cada actividad el moderador debe:

- Asegurarse de que la consulta que el participante propone quede claramente grabada en el video o se capture la imagen de la aplicación.
- Registrar sus observaciones.

Tiempo asignado a cada etapa de la prueba:

Etapa	Tiempo
Apertura	3 minutos
Presentación de la herramienta	25 minutos
Presentación de escenarios	2 minutos
Evaluación	60 minutos
TOTAL	1h 30min

Anexo F. Resumen de la aplicación de la segunda prueba de evaluación

Este anexo presenta un resumen de la aplicación de la segunda prueba de evaluación. Al finalizar cada actividad, el moderador explicó a los participantes los errores que había detectado en la formulación de la consulta. La versión del prototipo el lenguaje que se usó para esta prueba tenía los siguientes nombres para los operadores: *Select a Portion*, *Portion with Filter* (para el operador *Filter*), *Object with Characteristic* (para el operador *Filter+Additional Data*) y *Put Two Objects Closer*. La única diferencia es que *Object with Characteristic* no cambiaba el grafo esquema porque recuperaba toda la información del esquema asociada a los nodos instancia de la clase de interés.

Primer Participante

- **Actividad 1:** El participante redujo el tamaño del grafo para verlo completo en la pantalla. Luego marcó los nodos *Diagnosis*, *Patient*, *id* (de *Patient*) y *description* (de *Diagnosis*). Puso un filtro en *id* (= 723628). Marcó el nodo *relationship*. Seleccionó *Portion with Filter* que generó un mensaje de error porque no había marcado la clase de interés. Dudó, desmarcó y marcó *Patient*, luego lo marcó como clase de interés y seleccionó *Portion with Filter*.

No requirió ayuda del moderador.

Errores: (a) Sobraron datos porque dejó marcado el camino $\{Encounter, diagnosis, Diagnosis\}$ que la herramienta propuso.

- **Actividad 2:** El participante marcó los nodos *Diagnosis*, *description* y *Patient*. Puso el filtro en *id* (= 723628) pero a su vez lo eligió como clase de interés (el filtro se perdió). Puso el filtro sobre *specialty* (= endocrinology). Marcó *date* (de *Encounter*), *Medication*, *description* y *dose*. Seleccionó *Portion with filter*.

No requirió ayuda del moderador.

Errores: (a) Faltó filtro en *id*. (b) Sobraron datos porque dejó marcado el camino $\{Encounter, familyHistory, \beta_3, disease, Diagnosis\}$ que la herramienta propuso.

- **Actividad 3:** El participante marcó *Diagnosis*, puso filtro en *description* (= osteomyelitis), pero luego lo marcó como clase de interés y se perdió el filtro. Puso el filtro en *date* (de *Encounter*) (= 2000). Marcó todos los datos de *Patient*, luego desmarcó los datos demográficos. Marcó *race*, *Procedure*, *description*, marcó de nuevo el filtro sobre *description*, marcó como clase de interés *Diagnosis*. Seleccionó *Object with characteristics* y en el diálogo de clarificación eligió *match ANY conditions* para *diagnosis*.

No requirió ayuda del moderador.

Errores: (a) Debía elegir *match ALL conditions* en el diálogo. (b) Error en la especificación de la condición de la fecha: dejó el operador $=$, que propuso la herramienta por defecto, y solamente escribió el año.

Comentarios: dado que el operador *Object with characteristics* no cambia el esquema el participante se desconcertó y repitió el ejercicio. Esta repetición no se tuvo en cuenta en la evaluación de la actividad.

-
- **Actividad 4:** El participante marcó *Diagnostic Image* y puso condición de filtro sobre *annotation* (= acute cholecystitis). Marcó como clase de interés *Diagnostic Image*. Luego marcó *Procedure* y, como clase de interés *Procedure*, volvió a marcar *Diagnostic Image*, puso el filtro en *description* (de *Procedure*) (= colectomy). Marcó *birthDate*, *gender* y *race*. Marcó de nuevo *Diagnostic Image* como clase de interés. Eligió *Object with characteristic* y se dió cuenta de que no era el operador que requería. Volvió a empezar la formulación y esta vez eligió *Portion with filter* y en el diálogo de clarificación, *match ALL conditions* para *Encounter*.
No requirió ayuda del moderador.
Errores: (a) Debía elegir *match ANY condition* para *Encounter*.

Segundo Participante

- **Actividad 1:** El participante puso el filtro (= 723628) en *id* (de *Patient*). Marcó *Patient* como clase de interés y marcó los nodos *Diagnosis*, *relationship* y *description* (de *Diagnosis*). Seleccionó *Portion with Filter*.
No requirió ayuda del moderador.
Errores: (a) Sobraron datos porque dejó la marca del camino $\{Encounter, diagnosis, Diagnosis\}$ que la herramienta propuso.
- **Actividad 2:** El participante marcó *Patient*, puso el filtro (= 723628) en *id*, marcó los nodos *Diagnosis* y *specialty*, eligió como clase de interés *Encounter*, puso un filtro en *specialty* (= ENDOCRINOLOGY). Marcó *date* (de *Encounter*), *Medication*, *description* y *dose*.
Eligió *Portion with Filter* que generó un falso mensaje de error (error de la aplicación).
No requirió ayuda del moderador.
La formulación fue correcta.
- **Actividad 3:** El participante marcó como clase de interés *Diagnosis* y puso la condición en *description* (= OSTEOMYELITIS). Marcó *Patient* y desmarcó el nodo β_3 para quitar el camino adicional que propuso la herramienta. Puso el filtro sobre *date* (≥ 2000). Marcó *id* (de *Patient*), *Procedure* y *description*. Seleccionó *Diagnosis* como clase de interés, pasó un período de tiempo prolongado y cambió la clase de interés por *Patient*. Eligió *Object with characteristic* y en el diálogo de clarificación, *match ALL conditions* para *Encounter*.
Recibió ayuda clase B: el moderador explicó la diferencia entre los dos filtros.
La formulación fue correcta.
- **Actividad 4:** El participante marcó *Patient*, *Diagnostic Image*, *annotation* y como clase de interés, *Patient*. Puso el filtro en *annotation* (= ACUTE CHOLECYSTITIS). Marcó *Procedure*, *emphdescription* y puso en este último el filtro (= COLECTOMY). Marcó *race*, *gender* y *birthDate*. Seleccionó *Portion with Filter* y en el diálogo de clarificación, *match ALL conditions* para *Encounter*.

No requirió ayuda del moderador.

Errores: (a) Debía elegir *match ANY condition* para *Encounter*.

Tercer Participante

- **Actividad 1:** El participante ajustó el tamaño y posición del grafo para verlo completo en la pantalla. Marcó los nodos *Patient*, *Encounter* y *id* (de *Patient*). Puso el filtro en *id* (= 723628). Marcó los nodos *relationship*, *Diagnosis* y *description*. Marcó *Diagnosis* como clase de interés. Eligió *Select a Portion*.

No requirió ayuda del moderador.

Errores: (a) Sobraron datos porque dejó la marca que la herramienta propuso para el camino *{Encounter, diagnosis, Diagnosis}*.

- **Actividad 2:** El participante marcó los nodos *Diagnosis*, *description* y *Patient*. Puso el filtro en *id* (= 723628) y en *specialty* (= endocrinology). Marcó *date* (de *Encounter*), *Medication*, *description* y *dose*. Eligió como clase de interés *Diagnosis*. Seleccionó *Portion with Filter*, en el diálogo de clarificación eligió *match ALL conditions* para *Encounter*, cuando apareció la pregunta para los caminos de *Diagnosis* se dio cuenta de que había un error. Repitió todo el ejercicio quitando la marca del nodo β_3 para quitar el camino adicional que proponía la herramienta. No requirió ayuda del moderador.

La formulación fue correcta.

- **Actividad 3:** El participante marcó los nodos *Patient*, *Encounter*, *date*. Eligió *Patient* como clase de interés. Puso el filtro en *date* (= 01-01-2000). Marcó *description* (de *Procedure*). Puso el filtro en *description* (de *Diagnosis*) (= Osteomyelitis). Seleccionó *Portion with Filter* y en el diálogo de clarificación, *match ALL conditions* para *Encounter*.

No requirió ayuda del moderador.

Errores: (a) Eligió la operación de filtro equivocada. (b) En el filtro de fecha dejó el operador = que propone la herramienta por defecto.

- **Actividad 4:** El participante marcó *Patient*, puso filtro sobre *description* (de *Diagnosis*) (= acute cholecystitis) y *description* (de *Procedure*) (= colectomy). Marcó *birthDate*, *gender* y *race*. Eligió *Portion with Filter* y en el diálogo de clarificación, *match ALL conditions* para *Encounter*.

No requirió ayuda del moderador.

Errores: (a) Debía elegir *match ANY condition* para *Encounter*. (b) Ubicó la condición de la anotación en un nodo que no corresponde.

Cuarto Participante

- **Actividad 1:** El participante marcó los nodos *Diagnosis*, *description*, *Encounter*, *relationship* y *id* (de *Patient*). Puso el filtro en *id* (= 723628). Eligió *Select a Portion*.

No requirió ayuda del moderador.

Errores: (a) Sobraron datos porque dejó la marca, que la herramienta propuso, del camino $\{Encounter, diagnosis, Diagnosis\}$. (b) Eligió la operación incorrecta.

- **Actividad 2:** El participante marcó los nodos *description* (de *Diagnosis*) y *relationship*, puso el filtro en *id* (= 723628) y en *specialty* (= ENDOCRINOLOGY). Marcó *description* (de *Medication*) y *dose*. Demoró un tiempo antes de marcar *description* (de *Diagnosis*) como clase de interés. Eligió *Portion with Filter* y en el diálogo de clarificación seleccionó *match ALL conditions* para *Encounter* y para los caminos de *Diagnosis*.

No requirió ayuda del moderador.

Errores: (a) Sobraron datos porque dejó marcado el camino $\{Encounter, familyHistory, \beta_3, diagnosis, Diagnosis\}$ que la herramienta propuso. (b) Faltó marcar la fecha del encuentro.

- **Actividad 3:** El participante marcó *Patient* y *Diagnosis*, puso el filtro en *description* (= OSTEOMYELITIS). Volvió al grafo inicial, marcó *Patient*, marcó y desmarcó repetidas veces *Diagnosis* como clase de interés. Agregó de nuevo el filtro en *description* y el filtro de *date* (> 2000). Marcó *id* (de *Patient*) y *description* (de *Procedure*). Eligió *Portion with Filter* y en el diálogo de clarificación seleccionó *match ANY conditions* para *Encounter* y para *Patient*, *match ALL conditions*.

No requirió ayuda del moderador.

Errores: (a) Eligió la operación de filtro equivocada. (b) Hubo error en el filtro por el formato de la fecha de encuentro. (c) Sobraron datos porque no quitó la marca del camino $\{Encounter, familyHistory, \beta_3, diagnosis, Diagnosis\}$ que la herramienta propuso.

- **Actividad 4:** El participante marcó *Diagnostic Image*, puso filtro sobre *description* (de *Procedure*) (= COLECTOMY) y sobre *annotation* (= ACUTE CHOLECYSTITIS). Marcó *birthDate*, *gender*, *race* y como clase de interés, *Patient*. Seleccionó *Object with Characteristic* y en el diálogo de clarificación, *match ALL conditions* para *Encounter*.

No requirió ayuda del moderador.

Errores: (a) Debía elegir *Portion with Filter*. (b) En el diálogo de clarificación debía elegir *match ANY condition* para *Encounter*.

Anexo G. Diseño de la prueba comparativa

DISEÑO DE LA PRUEBA COMPARATIVA DE USABILIDAD

Este documento contiene el diseño de la prueba de usabilidad comparativa del prototipo de GraphTQL, y consta de las siguientes secciones: objetivo de la prueba, preguntas que motivan este estudio, hipótesis, metodología (escenarios, actividades, métricas y perfil de los participantes), plan de la prueba y materiales para la realización de la prueba.

1. Objetivo

El objetivo de esta prueba es comparar una herramienta gráfica de consulta sobre grafos y el prototipo de la herramienta desarrollada, para evaluar si estas le ayudan al usuario final a formular consultas que involucran filtros y datos incompletos. La prueba se enfoca en la formulación de la consulta y no incluye la presentación de los resultados de las mismas.

2. Preguntas que motivan este estudio

Con este estudio se busca obtener indicios sobre la posible respuesta para cada una de las siguientes preguntas:

- ¿La visualización del grafo esquema facilita la formulación de la consulta y contribuye a reducir errores?
- ¿Las operaciones de transformación de grafos facilitan la formulación de consultas que requieren filtros e incluyen datos incompletos?
- ¿La guía para desambiguar las condiciones le ayuda al usuario a proponer consultas con filtros y reduce los errores en su formulación?

3. Hipótesis

Las siguientes características de un sistema de consulta facilitan la formulación de consultas sobre un modelo de datos basado en grafos:

- Visualización de grafo esquema. Cuando no hay visualización de un grafo esquema el usuario debe explorar los datos para conocer cuál es el modelo de datos subyacente. Esta tarea puede resultar engorrosa. Por otro lado, un grafo esquema es más fácil de interpretar que un grafo de datos, aun cuando éste incluya solo una porción de los datos, y facilita la formulación de la consulta porque el usuario no debe construir, desde cero, los patrones, sino que selecciona las áreas de interés en el grafo esquema. Además, facilita identificar las clases y

los atributos y encontrar caminos entre ellos. También evita errores en la formulación de la consulta generados por el uso de arcos con dirección incorrecta o porque no se especifica el tipo de clase de los nodos.

- Operadores basados en la transformación del grafo, cuya definición tiene en cuenta que los datos son incompletos y permiten pensar a nivel de objetos y sus relaciones (mayor nivel de abstracción). Estos operadores actúan sobre una representación uniforme de los datos (nodos y arcos del grafo), mientras que los lenguajes basados en *pattern matching* (i.e. SPARQL) tienen operaciones sobre el grafo ---*pattern matching*--- y operaciones sobre conjuntos de mapeos ---*optional*, *union*, *join*. Además, las transformaciones del grafo esquema se reflejan en la interfaz, dando una retroalimentación al usuario en cada paso de la formulación de la consulta.
- Diálogo para desambiguar las condiciones. Este diálogo permite especificar de manera más precisa las condiciones y reducir el número de errores que se presentan cuando se requiere combinar operaciones (ej. en SPARQL combinación de grupos de patrones, *optional*, *filter* y *union*). La combinación de estas operaciones exige al usuario un mayor conocimiento del lenguaje de consulta, y de la semántica de sus construcciones.

4. Metodología

Esta es una prueba de usabilidad, exploratoria comparativa con dos herramientas de consulta de datos representados en grafos, la primera es *Gruff*¹ y la segunda, el prototipo de GraphTQL, el lenguaje propuesto. Se aplicará un estudio *between-subjects* y se realizará una prueba piloto.

El sistema de consulta propuesto pretende ayudar a los usuarios finales, con poco conocimiento sobre lenguajes de consulta. En esta prueba se busca comparar esta herramienta con otra herramienta de consulta gráfica, que tenga el poder expresivo para formular consultas que involucran filtros y datos incompletos. Las herramientas actuales que más se acercan a esta descripción son las *graphical query builders* para SPARQL². Entre ellas se eligió *Gruff*, porque facilita hacer *browsing* sobre los datos y formular consultas SPARQL de manera gráfica. *Gruff* ofrece una representación gráfica para la mayoría de las operaciones de SPARQL³. Además, otros *graphical query builders* y lenguajes visuales basados en SPARQL operan de

¹ <http://franz.com/agraph/gruff/>

² <http://www.w3.org/TR/sparql11-query/>

³ <http://www.w3.org/TR/sparql11-query/>

forma similar a *Gruff*. El prototipo propuesto representa los datos mediante GDM⁴ (*Graph Data Model*), mientras que para *Gruff* usa una representación de los datos en RDF-S⁵.

La prueba incluye cuatro etapas. La primera es la introducción, donde se describe la prueba; la segunda es la presentación de la herramienta a evaluar, donde se enseña al participante cómo usar la herramienta; la tercera es de evaluación, donde se pide al participante que formule cuatro consultas; y finalmente, la cuarta, la etapa de cierre, recoge las opiniones de los participantes.

Durante las actividades de evaluación el usuario puede pedir la ayuda del moderador o acudir a las ayudas de la herramienta. Estas ayudas se clasificaron, de acuerdo con el nivel de impacto que pueden tener en la eficiencia del desarrollo de la actividad, con el fin de tener en cuenta este impacto en las métricas.

- **Ayudas Clase A:** Con mucho impacto en la efectividad. Estas son las ayudas en las que el moderador da pistas o genera preguntas que llevan al participante a re-pensar la solución que está proponiendo.
- **Ayudas Clase B:** Con poco impacto en la efectividad. Son ayudas en las que el moderador aclara o explica de nuevo la semántica de las operaciones.
- **Ayudas Clase C:** Sin impacto en la efectividad. Estas incluyen la consulta de manuales o ayudas de la herramienta y las ayudas en las que el moderador:
 - Explica el funcionamiento de la herramienta, diferente a las operaciones, i.e. use clic derecho, haga *drag*, en que menú aparece una opción.
 - Aclara lo que se busca recuperar en la consulta de la actividad.

Se considera que las ayudas clase C afectan la eficiencia (tiempo que toma realizar la actividad) mas no la efectividad (capacidad de terminar la actividad con éxito) ya que esta información no afecta la solución que el usuario plantea, sino el cómo representar esa solución en la herramienta.

Las consultas de la etapa de presentación de las herramientas se definieron sobre un conjunto de datos sobre películas y series de televisión (Figura 1). Para las actividades de evaluación se definió un conjunto de datos que representa información clínica de pacientes (Figura 2). Los modelos usados en estas dos etapas son diferentes porque una de las preguntas del estudio se orienta a evaluar si la representación del grafo esquema facilita la formulación de las consultas. Por lo tanto, es importante observar qué ocurre cuando el participante debe empezar a usar un modelo de datos diferente.

⁴ Hidders, J. Typing Graph-Manipulation Operations. In *Proc. of the 9th Intl. Conf. on Database Theory*. Springer-Verlag, 2002.

⁵ <http://www.w3.org/TR/rdf-schema/>

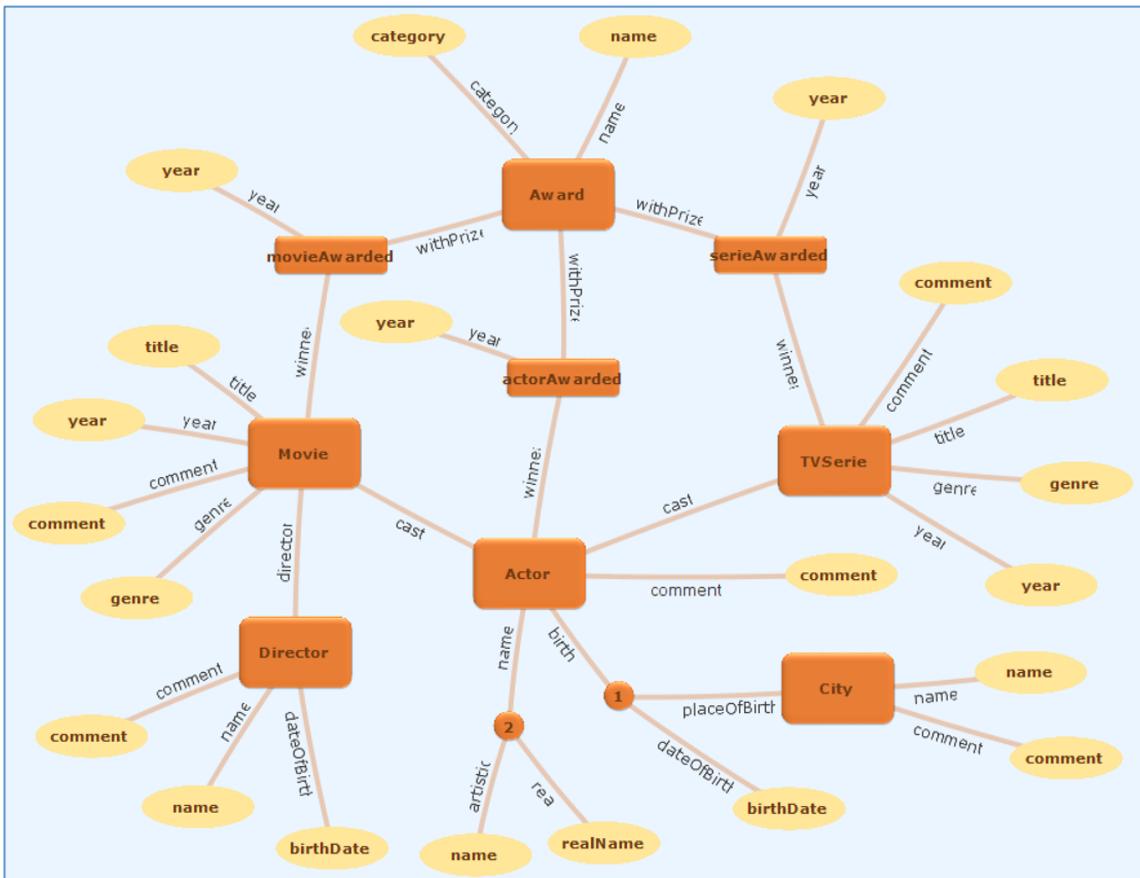


Figura 1. Esquema de los datos de películas y series de televisión.

Dado que *Gruff* está desarrollado en inglés, se usará una versión también en inglés de los modelos, los datos, y los manuales de las herramientas. El moderador ofrecerá la traducción de los textos y mensajes cuando el participante lo solicite. Las consultas de los ejemplos y las actividades se seleccionaron cuidadosamente para responder a las preguntas del estudio y reducir la sobrecarga en la formulación de las mismas (ej. por la cantidad de nodos requeridos).

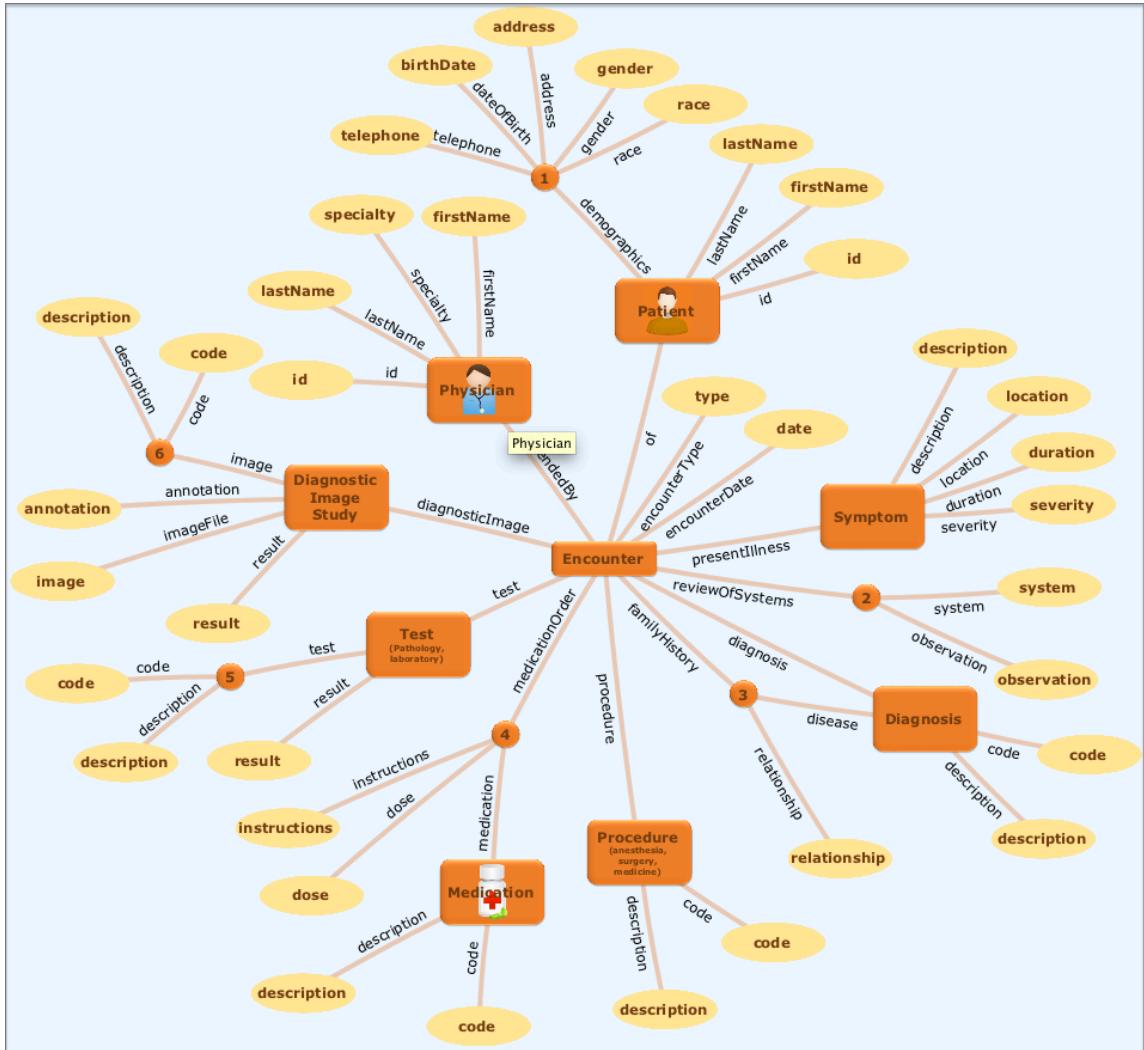


Figura 2. Esquema de los datos clínicos.

a. Escenarios y clasificación de las consultas

Las consultas de esta prueba están ambientadas en dos escenarios:

- Escenario 1: Un médico está en su consultorio atendiendo la consulta de un paciente.
- Escenario 2: Un médico está buscando historias clínicas de pacientes que cumplen ciertas características, para iniciar una investigación.

Estos escenarios se eligieron entre un grupo de posibles escenarios que se identificaron en las pruebas de usabilidad realizadas. Durante el desarrollo de estas

pruebas, los profesionales de la salud participantes dieron ejemplos de consultas, de los cuales se recopilaron 31 consultas. De estas 31 consultas, 11 requieren funciones de agregación y 4 requieren establecer comparaciones entre variables (comparaciones entre nodos del grafo). Estos dos tipos de pregunta no se incluyeron en esta prueba dado que las comparaciones entre variables no se han definido completamente para la herramienta y por tanto no están implementadas en la interfaz, además de que las funciones agregadas hacen parte de trabajo futuro del sistema de consulta propuesto. Las 16 preguntas restantes, que se pueden especificar en la versión prototipo actual del sistema de consulta, se clasificaron de acuerdo con los niveles de complejidad que se presentan en la Tabla 1. De acuerdo con esta clasificación hay 5 consultas de nivel 1, 8 de nivel 2, y 3 de nivel 3. Los niveles de complejidad de las consultas (Tabla 1), se propusieron de manera similar a la clasificación propuesta por Bell y Rowe⁶ y Owei et al.⁷, teniendo en cuenta los operadores de transformación de grafos y la hipótesis de este estudio.

Tabla 1. Clasificación de las consultas según su complejidad.

Nivel	La consulta incluye
1	<i>Pattern matching</i> con una condición de filtro, con o sin datos opcionales
2	<i>Pattern matching</i> con más de una condición de filtro, con o sin datos opcionales
3	<i>Pattern matching</i> con condiciones de filtro y recuperando todos los datos conectados a una clase seleccionada
4	<i>Pattern matching</i> con condiciones de filtro y otras operaciones (i.e. unión)

Las consultas seleccionadas para las actividades de la prueba son de mediana complejidad, en el sentido de que incluyen patrones formados por varias tripletas. En esas consultas los filtros no tienen solamente impacto local (i.e. para afectar solamente el tipo de nodo o la variable involucrada en la expresión de filtro), sino que también inciden sobre otros nodos. En estas consultas se requiere encontrar

⁶ Bell, J. E., & Rowe, L. A. An exploratory study of ad hoc query languages to databases. In *Eight Intl. Conf. on Data Engineering*, 1992. IEEE Comput. Soc. Press.

⁷ Owei, V., Navathe, S. B., & Rhee, H.-S. An abbreviated concept-based query language and its exploratory evaluation. *Journal of Systems and Software*, 63(1), 2002.

objetos de una clase A que están conectados por medio de un camino en particular con objetos de la clase B que cumple un filtro. Por ejemplo, en la consulta "Encuentre los pacientes que han recibido un diagnóstico de diabetes", el filtro se especifica sobre "diagnóstico"; sin embargo, la consulta pide los pacientes que están conectados con ese diagnóstico. Dado que todos los patrones de las consultas de la prueba tienen este nivel de complejidad, los niveles planteados en la Tabla 1 no tienen en cuenta si el patrón de consulta es simple (una o dos tripletas) o complejo (varias tripletas).

b. Consultas para la presentación de la herramienta y para las actividades

En total se seleccionaron 7 consultas: 3 ejemplos para ilustrar el uso de la herramienta y 4 para las actividades de evaluación. Estas consultas, que se muestran en las Tablas 2 y 3, son de nivel 1, 2 o 3, debido a que un alto porcentaje de los ejemplos dados por los médicos corresponden a estos niveles.

Tabla 2. Consultas para los ejemplos.

Consulta	Nivel
Se requiere encontrar los títulos de las películas galardonadas con la categoría “ <i>Best Picture Academy Award</i> ” a partir del año 2000. Además, se desea incluir el nombre de los actores y el género de la película, si están disponibles.	2
De los actores que han ganado un premio en la categoría “ <i>Outstanding Lead Actor</i> ”, se requieren todos los datos disponibles de: títulos de las películas en que han actuado, categoría y año de los premios que han recibido. (Note que estos datos incluyen los otros premios, además de “ <i>Outstanding Lead Actor</i> ”).	3
Se requiere los nombres de los actores que han trabajado en películas de género “ <i>Comedy</i> ” y también han trabajado bajo la dirección de “ <i>Robert Zemeckis</i> ”. (Observe que pudieron ser películas diferentes, pero el actor cumple con ambas condiciones).	2

Tabla 3. Consultas para las actividades.

Consulta	Nivel	Escenario
Se requieren las enfermedades (incluyendo su descripción) registradas en la historia familiar del paciente con identificación 723628; incluir la relación familiar si está disponible.	1	I
Se requiere encontrar los diagnósticos (incluyendo su descripción) dados al paciente con identificación 723628 por médicos con especialidad en " <i>ENDOCRINOLOGY</i> " y, si está disponible, las fechas de las consultas y las medicinas que le recetaron en ese momento (incluyendo descripción y dosis).	2	I
De los pacientes que han tenido un diagnóstico de " <i>OSTEOMYELITIS</i> " desde el año 2000 (desde 01-01-2000), se requieren todos los datos de: identificación del paciente, diagnósticos recibidos, y procedimientos que se le han realizado (incluir la descripción de los diagnósticos y procedimientos). (Note que estos datos incluyen los otros diagnósticos, además de <i>OSTEOMYELITIS</i> , y todos los datos que se piden son independientes de la fecha en que se registraron).	3	II
Se requiere encontrar los pacientes que han tenido un estudio de imágenes diagnósticas con anotación " <i>GALLSTONES</i> " y se les ha realizado el procedimiento " <i>COLECTOMY</i> ". De estos pacientes se requiere la fecha de nacimiento, el género y la raza, si están disponibles. (Observe que estos resultados pudieron realizarse en encuentros diferentes, pero el paciente cumple con ambas condiciones).	2	II

Las consultas se formularon teniendo en cuenta además otras consideraciones, por ejemplo, cuando se usa Gruff se requiere un tiempo para dibujar la consulta, por esto se limitó la cantidad de nodos que requiere la consulta. Adicionalmente, y teniendo en cuenta que en Gruff no hay un equivalente al *path contraction*, la formulación de las consultas seleccionadas no necesitan de esta operación. Tampoco se incluyeron ejemplos que requirieran otras operaciones de SPARQL (por ejemplo *union*), para evitar que los participantes tengan que aprender estos conceptos.

La Tabla 4 muestra la relación entre las consultas y las preguntas que motivan este estudio. En particular la tercera consulta de los ejemplos, y la tercera consulta de la evaluación, presentan un caso particular que podría mostrar la facilidad que la desambiguación provee al momento de especificar los filtros de la consulta.

Tabla 4. Relación de las consultas con las preguntas del estudio.

Preguntas del estudio	Consultas
¿La visualización del grafo esquema facilita la formulación de la consulta y contribuye a reducir errores?	Todas
¿Las operaciones de transformación de grafos facilitan la formulación de consultas que requieren filtros e incluyen datos incompletos?	Todas
¿La guía para desambiguar las condiciones le ayuda al usuario a proponer consultas con filtros y reduce los errores en su formulación?	Consultas de nivel 2, 3 y 4

c. Métricas

Este estudio mide dos aspectos de la experiencia de usuario: rendimiento y satisfacción⁸. Las métricas de rendimiento para cada actividad son:

- **Efectividad:** mide el grado con que la consulta que propone el usuario, retorna información correcta. Se definen dos medidas de efectividad: (a) número de consultas correctamente formuladas, y (b) número de errores cometidos en la formulación de las consultas. Dado que en una misma actividad se pueden presentar varios errores de un mismo tipo, en cada actividad se contabiliza el número de tipos de error diferentes. Por ejemplo, si en una consulta hay tres arcos que se debían marcar como opcionales, pero no se marcaron, se contabiliza un error.

Algunos tipos de error que podrían ocurrir en la formulación son los siguientes:

- a. Error en los caminos que conectan los nodos. En *Gruff* esto puede ocurrir por una definición errada de los arcos (tripletas), porque el camino no existe en la base de datos, o porque el camino existe pero representa una relación diferente, faltan o sobran arcos (tripletas) o porque la dirección del arco (tripleta) está invertida. En GraphTQL puede ocurrir que se seleccione un camino, entre los nodos, diferente al requerido.

⁸ Tullis, T., & Albert, W. Measuring the user experience. 2008.

-
- b. Uso errado de un elemento u operador. En *Gruff* el usuario debe especificar si las tripletas son o no opcionales. En GraphTQL el usuario debe escoger uno de los cuatro operadores.
 - c. Hacen falta nodos y arcos (tripletas) necesarias para la especificación del patrón del filtro (aplica igual en ambas herramientas).
 - d. Las condiciones se ubican en nodos diferentes (aplica igual en ambas herramientas).
 - e. Error al establecer la conjunción/disyunción de las condiciones del filtro. En *Gruff* esto puede estar relacionado con el patrón en sí mismo o con la especificación del filtro. En GraphTQL ocurre cuando la respuesta a las preguntas de desambiguación no corresponden con lo solicitado.
 - f. Ausencia de datos requeridos en la consulta que no son parte del patrón de filtro. Nótese que este error tiene un impacto diferente al error del numeral c., porque en este caso el patrón de filtro puede haber sido especificado correctamente, con lo cual se recupera al menos una parte del resultado correcto.
- **Eficiencia en la formulación de la consulta:** mide el tiempo (minutos y segundos) requerido para formular cada consulta.
 - **Número de veces que recurre a la ayuda:** se registra clasificada según la clase de ayuda.

Para medir la percepción de los participantes con respecto a la facilidad de uso del software se usará el cuestionario SUS⁹ (*System Usability Scale*), con las modificaciones propuestas por Bangor et al.¹⁰.

d. Perfil de los participantes

La prueba se realizará con participantes con los siguientes perfiles:

- **Profesionales de la salud.** Médicos, enfermeros o profesiones afines, para quienes puede ser de interés la información clínica de los pacientes. Deben tener algún grado de familiaridad con el manejo de historias clínicas en sistemas de información. Pueden ser, o no, especialistas en cualquier área.
- **Estudiantes de medicina.** Se buscará la participación de estudiantes de 6º semestre en adelante, de la Universidad del Valle o la Universidad Javeriana Seccional Cali.

⁹ Brooke, J. SUS - A quick and dirty usability scale. In Usability evaluation in industry. 1996.

¹⁰ Bangor, A., Kortum, P., & Miller, J. An Empirical Evaluation of the System Usability Scale. Intl. Journal of Human-Computer Interaction, 24(6), 2008.

Todos los participantes deben ser personas que habitualmente usen el computador como herramienta en sus labores profesionales o de estudio.

La edad, el género o la raza no son criterio de selección de los participantes.

La mitad de los participantes de cada grupo evaluará GraphTQL, mientras que la otra mitad evaluará *Gruff*.

e. Lugar y equipo para la prueba

Las pruebas se realizarán en una oficina que puede ser la del participante u otra destinada a esta actividad.

Para la prueba se usará un computador portátil, una máquina virtual con S.O Linux y las herramientas de consulta. *Gruff* se ejecuta sobre la máquina virtual.

5. Plan de la Prueba

La prueba consta de las siguientes partes:

- **Apertura:** El moderador explica el propósito de la prueba y las partes que la integran. Se diligencia el cuestionario pre-test y el participante firma el formato de consentimiento.
- **Datos para la demostración de la herramienta.** El moderador explica al usuario el conjunto de datos que se va a usar para la presentación de la herramienta (datos de películas y series de televisión).
- **Presentación de la herramienta que le corresponde al usuario evaluar.** El moderador explica al usuario las opciones, botones y operaciones que puede realizar. En el caso de *Gruff* solamente se presentarán las opciones y conceptos necesarios para resolver las actividades de la prueba. El moderador desarrolla tres ejemplos (Tabla 2).
- **Actividades.** Incluye:
 - Presentación de los datos y escenarios: el moderador explica al participante los datos para las actividades y los dos escenarios en los que estas se desarrollan.
 - Evaluación: consta de 4 actividades, dos en el escenario I y dos en el escenario II. Los participantes contarán con un tiempo máximo de 15 minutos para realizar cada una de las actividades.

En cada actividad el moderador debe registrar:

- La hora de inicio y finalización.
- El número de veces que el usuario solicita las diferentes clases de ayuda.
- El número de errores que ocurren durante la realización de la actividad.

Al finalizar cada actividad el moderador debe:

-
- Asegurarse de que la consulta que el participante propone quede claramente grabada en el video o se capture la imagen de la aplicación.
 - Registrar sus observaciones.
- **Cuestionario post-test.**

Tiempo asignado a cada etapa de la prueba:

Etapa	Tiempo
Apertura	3 minutos
Presentación de la herramienta	35 minutos
Presentación de escenarios	2 minutos
Evaluación	60 minutos
Post-test	5 minutos
TOTAL	1h 45min

6. Disposición de los materiales y las herramientas para la prueba

A continuación presentan los materiales que se requieren para la prueba, estos son:

- *Screener questionnaire.*
- Guión de la prueba para la herramienta 1 (*Gruff*).
- Guión de la prueba para la herramienta 2 (GraphTQL).
- Formato para registro por parte del moderador.
- Cuestionario post-test.

Disposición de la herramienta *Gruff*:

- Conectar a la base de datos *Actors* para la etapa de demostración, y a la base de datos *Clinical* para la fase de evaluación.
- *Select current predicates* (menú *Global Options*): seleccionar todos los predicados, buscarlos con ejemplos de datos. De esta manera se agiliza el proceso de formulación de las consultas.
- Cambiar los valores por defecto de los siguientes parámetros en la herramienta:
 - *Maximum simple triples to display* (menú *Visual Graph Options – Inclusion Options*): 60
 - *Path finding timeout* (menú *Visual Graph Options – Finding Paths Between Nodes*): 40

-
- *Maximum paths to display* (menú *Visual Graph Options – Finding Paths Between Nodes*): 20
 - *Select or Create a text index* (menú *Text Search*): seleccionar todos los predicados, indexar los objetos.

PRUEBA COMPARATIVA - HERRAMIENTAS DE CONSULTA SOBRE MODELOS DE GRAFOS

Introducción

Mi nombre es María Constanza Pabón, soy estudiante del doctorado en ingeniería de la Universidad del Valle y profesora en la Pontificia Universidad Javeriana. Estoy realizando una investigación que tiene como propósito el desarrollo de herramientas gráficas que faciliten la consulta de datos. El caso de estudio particular tiene aplicación en el dominio médico, específicamente en consultas de datos clínicos de pacientes. En particular en el dominio médico, el fácil acceso a los datos clínicos de los pacientes es importante en procesos como la atención al paciente, la investigación, la enseñanza y la vigilancia epidemiológica, entre otros.

Como parte de este trabajo requiero involucrar usuarios potenciales de las herramientas para evaluar la facilidad de uso de las aplicaciones que se están desarrollando.

Por ello lo estamos invitando a participar en un estudio de comparación de dos herramientas gráficas de consulta de datos representados en grafos, en una sesión que dura aproximadamente 1h 45m. Durante la sesión se presenta y explica una de las herramientas al participante, luego se le pide que realice algunas actividades (consultas) y que finalmente responda un cuestionario con el que evalúa la herramienta.

Agradecemos la colaboración que nos pueda brindar en este estudio.

Datos del participante

Nombre: _____

Género: F M

¿Usted nos permitiría grabar la sesión en video? Si No

Usted es:

- Estudiante de medicina, enfermería, o carreras afines
- Profesional en medicina, enfermería, o su labor profesional se relaciona con campos de la salud
- Otro

-
- ¿Cuál es su profesión/ Que carrera estudia? _____
-

- Especialista en / Otros títulos alcanzados:
-
-

¿Uso usted el computador para realizar labores de estudio/trabajo? Si No

¿Para qué usa típicamente el computador (correo, sistemas de información,...)?

Tiene experiencia en el uso de lenguajes de consulta sobre bases de datos? Si No

Su rango de edad es:

- Menor de 18 años (finalizar) 18 a 29 años 30 a 39 años
 40 a 49 años 50 a 59 años 60 años o más

FORMULARIO DE CONSENTIMIENTO

Estudio: Prueba comparativa - Herramientas gráficas de consulta sobre modelos de grafos

El propósito de este estudio es comparar la facilidad de formulación de consultas en dos herramientas gráficas de consulta de datos representados en grafos.

Durante la sesión, que dura aproximadamente 1h 45m, se presenta y explica el uso de una herramienta, luego se le pide al participante que realice algunas actividades (consultas) y que finalmente responda un cuestionario con el que evalúa la herramienta.

La participación en este estudio es voluntaria y anónima. La prueba podría ser video-grabada. Se garantiza que los datos personales y las grabaciones son confidenciales y se usarán solamente para estudiar las características de la herramienta, en cumplimiento de la ley 1581 de 2012 y el decreto 1377 de 2013 de la República de Colombia sobre protección de datos personales.

En esta prueba se evalúa la herramienta (la tecnología), no la capacidad o formación de los participantes.

El participante podrá dar por terminada la sesión en el momento que lo desee.

Yo _____
con _____

identificación _____ acepto
participar _____

en este estudio y entiendo los términos del mismo.

Firma del Participante: _____

Fecha: _____

Moderador: _____

Firma del moderador: _____

Guion de la Prueba para herramienta 1.

El propósito de esta prueba es tener la opinión del usuario sobre el uso de una herramienta de consulta de datos. Esta prueba hace parte de un test en el que se comparan dos herramientas. La mitad de los participantes evalúan la primera herramienta y la otra mitad, la segunda herramienta. No hay respuestas correctas o incorrectas, todo lo que el participante haga y opine ayudará a mejorar el diseño de la herramienta.

La prueba incluye: a) Registro de los datos del participante y firma del consentimiento para realizar la prueba, b) Presentación de la herramienta a evaluar, c) Realización de 4 actividades. El participante dispondrá de un tiempo máximo de 15 minutos para realizar cada actividad. El participante puede usar el manual y la ayuda de la herramienta, también solicitar ayuda al moderador o hacer preguntas sobre las actividades. El participante puede solicitar al moderador la traducción de los textos de la herramienta.

1. Registro de los datos del participante (*screener questionnaire*) y firma el consentimiento para realizar la prueba.

2. Espacio para que los participantes expresen sus dudas e inquietudes con respecto a la prueba.

3. Datos para la demostración de la herramienta

Para mostrar el funcionamiento de la herramienta se usará una base de datos con información de películas y series de televisión. Esta información incluye los directores de las películas o series, los actores del reparto, la ciudad de nacimiento de los actores, y los premios recibidos por las películas, las series o los actores. Algunos de estos datos son reales (ej. Los títulos de las películas, los nombres de los actores) y otros son ficticios (ej. Las fechas, los géneros y los premios).

La información que se tiene es:

- Las películas son de tipo (*type*) *Movie*, y tienen: título (*title*), el año en que se estrenó (*year*), el género (*genre*), un comentario (*comment*), quién la dirigió (*directedBy*), y el reparto (*cast*).
- Las series son de tipo (*type*) *TVSerie*, y tienen: título (*title*), el año en que se estrenó (*year*), el género (*genre*), comentario (*comment*), y el reparto (*cast*).
- Los directores son de tipo (*type*) *Director* y tienen: nombre (*name*), fecha de nacimiento (*dateOfBirth*), y comentario (*comment*).

- Los actores son de tipo (*type*) *Actor* y tienen: nombre (*name*, este es el nombre artístico), nombre real (*realName*), fecha de nacimiento (*dateOfBirth*), lugar de nacimiento (*placeOfBirth*) y un comentario (*comment*).
- Los premios son de tipo (*type*) *Award* y tienen: un nombre (*name*) y una categoría (*category*). El nombre, es el nombre del premio, por ejemplo "*Academy Awards*", o "*Emy Awards*". Cada premio se da en varias categorías, por ejemplo: "*Best Actress in a Leading Role*", o "*Best Picture Academy Award*". Dado que el mismo premio y categoría se da a diferentes actores en diferentes años, el tipo *ActorAwarded* tiene la información de que premio (*withPrize*) se entregó a que actores (*winner*) y en qué año (*year*). De la misma manera, el tipo *MovieAwarded* y *SerieAwarded* tienen la información de que premio (*withPrize*) se otorgó a que película o serie (*winner*) y en qué año (*year*).
- Las ciudades son de tipo (*type*) *City* y tienen: nombre (*name*), y comentario (*comment*).

Adicionalmente, las películas, las series, los directores, los actores, las ciudades y los premios tienen el dato *label*, que es igual al nombre en el caso de los directores, actores, ciudades y premios; e igual al título en el caso de las series y las películas. Este dato se agregó porque la herramienta tiene una opción para buscar por el valor del *label*.

Además, se debe tener en cuenta que los datos de una película o serie particular pueden estar incompletos. Por ejemplo, no todas las películas tienen género, o han sido galardonadas con un premio, o tienen registrado un director. Lo

mismo sucede con los datos de los actores, directores y premios.

Los datos se representan con grafos donde los nodos (rectángulos) representan un objeto, una característica o atributo de un objeto; y los arcos (líneas que conectan los nodos) representan una relación entre los nodos. En la Figura 1, se muestra parte de la pantalla de la aplicación, en el lado izquierdo se

identifican (por colores) los tipos de objeto que se encuentran en el lado derecho y los nombres de las relaciones (o de los atributos).

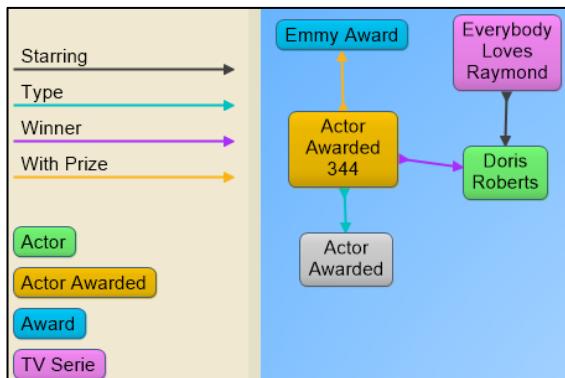


Figura 1. Datos representados en grafo.

4. Demostración de la herramienta

Se presentarán solamente las opciones de la herramienta necesarias para realizar las actividades.

El modo *Graph View* (menú *View*)

Permite seleccionar datos y mostrarlos en una representación de grafo. Puede ser usado para explorar la organización de los datos.

Los datos se buscan por medio de las opciones de menú que se describen a continuación. Una vez hay datos seleccionados, estos se despliegan la pantalla, cuya distribución se muestra en la Figura 2. En la parte izquierda de la pantalla, el *Legend Panel* muestra los nombres de los arcos (identificados por color y tipo de línea), los tipos de objeto, y un *overview* del grafo. A la derecha se despliega el grafo con los datos seleccionados. Este modo tiene las siguientes opciones de menú:

- Menú *Display*
 - *Display some sample triples*: muestra un ejemplo de los datos. Algunas acciones que se pueden realizar sobre el resultado son:
 - Agregar datos dando clic derecho en un nodo y eligiendo “*Display linked nodes from menus*” o “*Display linked nodes from a tree*”.
 - *Zoom* del grafo (*scroll* del mouse).
 - Desplazarse en el grafo moviendo la ventana marcada en el *overview* (clic y arrastrar).
 - *Display a node by label*: permite buscar (o incluir en el grafo desplegado) un nodo dado el label (título de la película o serie de TV, nombre del actor, nombre del director,...). El label que se escribe debe coincidir con el guardado en la base de datos (incluyendo mayúsculas, minúsculas, espacios, y debe escribirse completo).
 - *Select an instance node by type*: permite buscar un objeto particular en una lista de objetos del tipo seleccionado. La lista está en orden alfabético.
- Menú *Text Search*
 - *Find and Display Nodes*: lista los nodos que están relacionados (tienen un arco) con un nodo que contiene la(s) palabra(s) buscadas.
- Menú *Link*
 - *Display linked nodes for the current predicates*: agrega los nodos relacionados con el nodo seleccionado (nodos conectados por un arco).
 - *Display only linked nodes for the current predicates*: despliega solamente el nodo seleccionado y los nodos relacionados con él (nodos conectados por un arco). Se borran los demás nodos.

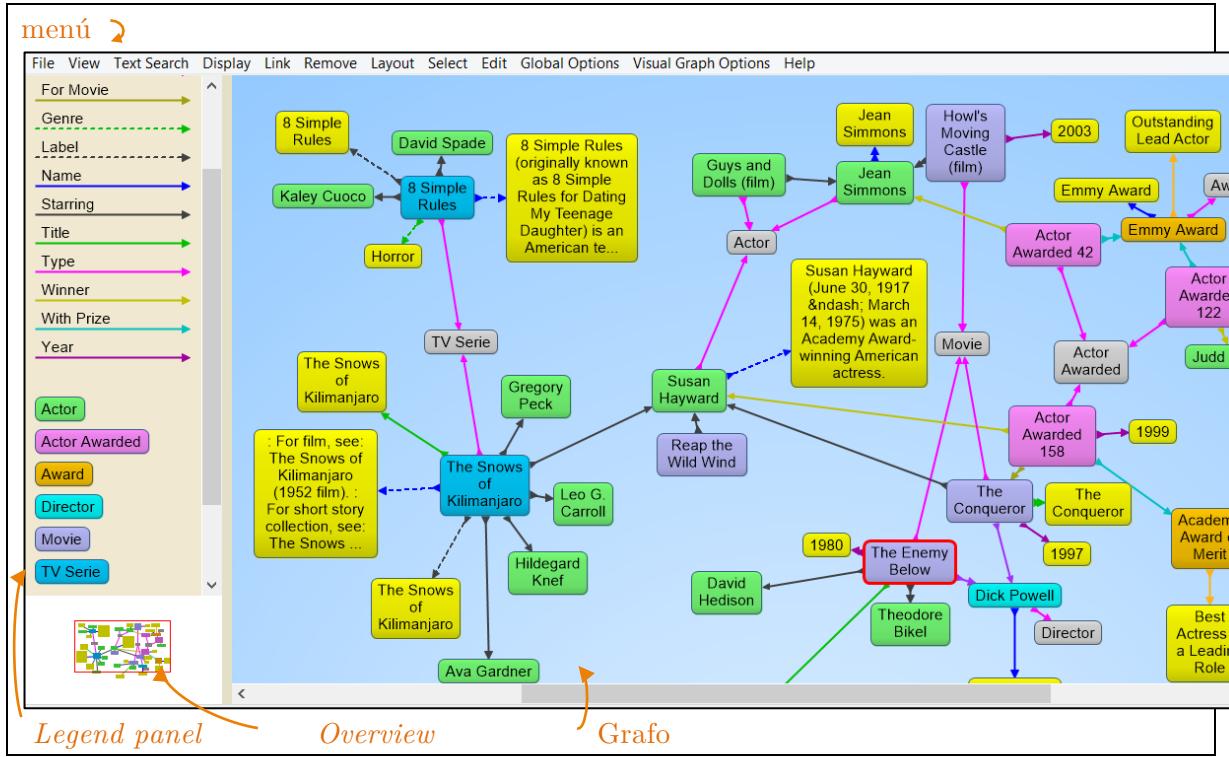


Figura 2. Pantalla de la herramienta.

- Menú *Remove*
 - *Remove selected node*
 - *Exclude selected node*: borra el nodo seleccionado, el nodo no vuelve a aparecer como resultado de otros comandos.
 - *Remove selected link*
 - *Remove all nodes*

El modo *Graphical Query View* (menú *View*)

Permite formular consultas sobre los datos. Las consultas se formulan construyendo un patrón de búsqueda, éste es un grafo que describe las características de los datos que se quieren recuperar. El grafo se compone de nodos y arcos. Los nodos pueden ser no-variables (nodos de la base de datos, datos particulares) o pueden ser variables (son nodos que no conocemos o que estamos buscando). Los arcos representan relaciones entre nodos o atributos. La respuesta a la consulta son los datos que se ajustan al patrón (ver Figura 3).

Ejemplo:

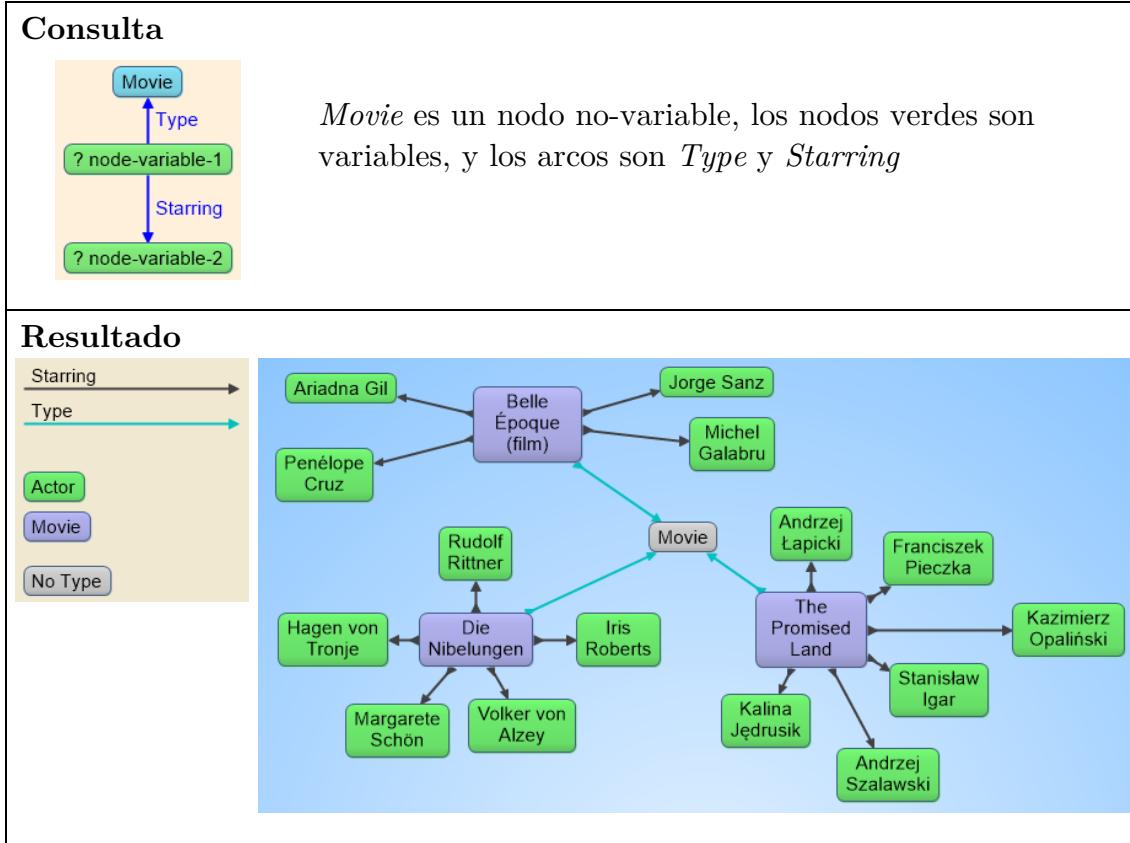


Figura 3. Ejemplo de consulta.

La Figura 4 muestra la aplicación en modo *Graphical Query View*. El patrón de consulta se construye en la ventana del lado derecho. Para agregar un nodo se hace clic derecho y eligiendo del menú “*Add variable node*” o “*Add non-variable node*” según el caso. Para los nodos variables se especifica un nombre, para los no-variables se busca el nodo en la base de datos. Este nodo puede ser buscado por el *label* (*Node by label*), por el tipo de dato (*Instance node by type*), buscando un texto (*Subject node by text search*), o si el nodo está en el *graph view* se puede copiar (ctrl-c) y pegar en el *graphical query view* (ctrl-v).

Generalmente se requiere especificar el tipo de los nodos variables. Para ello se hace clic-derecho sobre el nodo, en el menú que aparece se elige la opción “*Specify Node Filter*”, luego la opción “*type*”, finalmente el tipo de objeto al que hace referencia ese nodo.

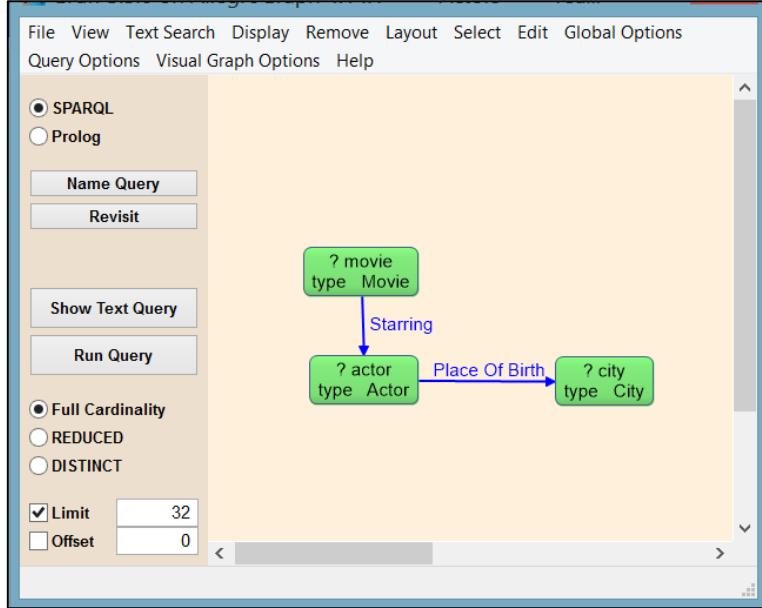


Figura 4. Modo *Graphical Query View*.

Para crear un arco entre los nodos se hace clic derecho sobre uno de los nodos y se elige “Add Predicate Link” (o “Add Predicate Variable Link”, o “Add Filter Link”). Cuando se agrega un arco no variable, aparece un menú con diferentes opciones para buscar el nombre de arco que se requiere. Para estos ejercicios se recomienda usar “All Predicates” que despliega todos los nombres de los arcos de la base de datos y permite seleccionar el que se requiere. Para los arcos con variable, se puede especificar el nombre de la variable. Para los arcos con filtros, la aplicación permite especificar una condición sobre el arco.

Dado que la base de datos tiene información incompleta (i.e. podrían haber películas sin la información de los actores de su elenco), se puede dar la opción de que una parte del patrón sea opcional. Por ejemplo, en la Figura 5 se buscan nodos de tipo *Movie* y opcionalmente los actores de la película (*Starring*). Por esto, en el resultado aparece la película “*Knight Moves*”, que no tiene actores registrados. Si el arco no fuese opcional, esta película no aparecería en el resultado, pues la consulta estaría requiriendo recuperar las películas que tuviesen actores.

Para seleccionar un arco como opcional (*optional*), se da clic derecho sobre el arco y se elige la opción “Toggle Optional”. Si se requiere marcar como *optional* varios nodos encadenados, se debe usar un agrupador *optional*: dar clic derecho sobre la ventana, elegir “Add a Grouper” y luego “Add an Optional Grouper”. Esto crea un cuadro de agrupamiento donde se ubican los nodos y arcos opcionales. Además se hace necesario duplicar, dentro del agrupador, un nodo de afuera del

agrupador para hacer el enlace. Este nodo que se duplica debe tener el mismo nombre, para que se enlacen los nodos de afuera del agrupador con los de adentro.

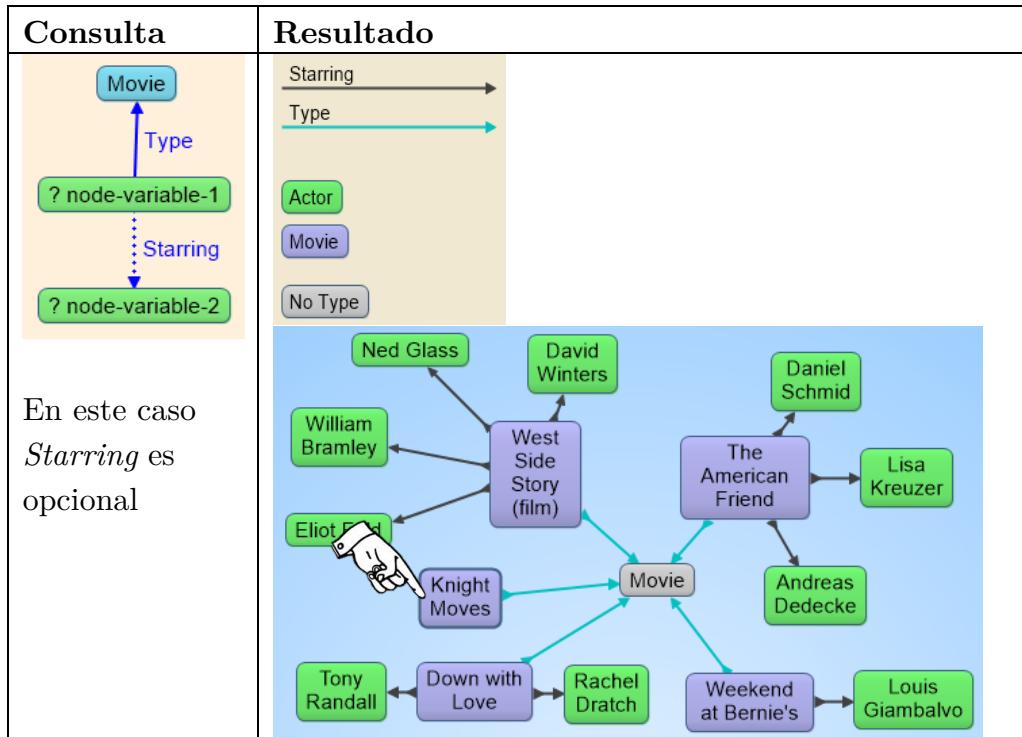


Figura 5. Ejemplo de consulta con *optional*.

El clic derecho sobre los nodos da las opciones de: agregar arcos, especificar el filtro sobre el nodo, renombrar el nodo (si es variable), cambiar la variable, cambiar de variable a no variable (y viceversa), y borrar el nodo. El clic derecho sobre un arco da las opciones de: cambiar el tipo de arco (variable, no variable, con filtro), agregar características, convertirlo en opcional (y viceversa), cambiar la dirección y borrarlo. Finalmente, el clic derecho sobre una porción vacía de la ventana da las opciones de: agregar variable, borrar nodos o arcos, agregar un agrupador, agregar un filtro general, y especificar las variables que se entregan en el resultado y las variables por las que se ordena el resultado.

Finalmente, cuando la consulta está creada se presiona el botón “Run Query”.

Ejemplos

Primera consulta: *Se requiere encontrar los títulos de las películas galardonadas con la categoría “Best Picture Academy Award” a partir del año*

2000. Además, se desea incluir el nombre de los actores y el género de la película, si están disponibles.

En el *Graphical Query View* (ver Figura 6):

- Identificar los tipos de objeto que hay en la consulta: películas, premios, y actores. Crear un nodo variable para cada uno, y especificar como filtro del nodo el tipo correspondiente.
- Identificar los datos que interesan de cada objeto: el título de las películas, la categoría de los premios, el año del premio, y el nombre de los actores. Agregar un nodo para cada uno de ellos.
- Agregar los arcos entre los nodos. Para identificar los nombres y dirección de los arcos se puede consultar el *graph view*. Hacer notar que para unir película y categoría de premio es necesario agregar otros nodos.
- Identificar las condiciones sobre los datos: la categoría “*Best Picture Academy Award*”, el año del premio mayor a 2000.
- Marcar los nodos y arcos que son opcionales, en este caso el actor y su nombre, y el género de la película. El actor y nombre de actor deben ir dentro de un agrupador.
- Ejecutar la consulta (*Run Query*).

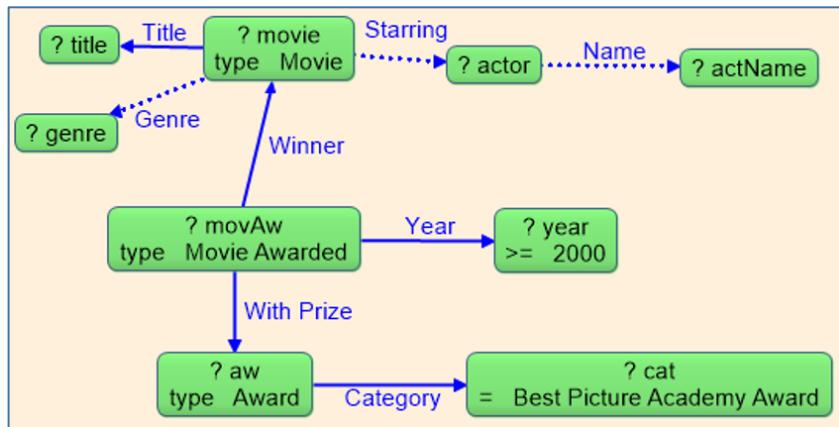


Figura 6. Primera consulta de ejemplo.

Segunda consulta: De los actores que han ganado un premio en la categoría “Outstanding Lead Actor”, se requieren todos los datos disponibles de: títulos de las películas en que han actuado, categoría y año de los premios que han recibido. (Note que estos datos incluyen los otros premios, además de “Outstanding Lead Actor”).

Hacer notar que este caso es diferente al primer ejemplo porque aquí se requieren los datos de los otros premios, además del premio de la condición. Por lo cual se necesita encontrar los actores que ganaron el premio y agregar, de nuevo, los premios (los otros).

En el *Graphical Query View* (ver Figura 7):

- Para encontrar los actores:
 - Identificar los tipos de objeto que hay en la condición: actores y premios. Crear un nodo variable para cada uno, y especificar como filtro del nodo el tipo correspondiente.
 - Identificar las condiciones sobre los datos: categoría = “*Outstanding Lead Actor*”
- Agregar los datos opcionales: título de las películas, categoría y año de los premios. Si son comunes a los de la condición, agregarlos dos veces, con nombres de variables diferentes. Agregar agrupadores si es necesario.
- Agregar los arcos entre los nodos. Para identificar que arcos usar, consultar el *graph view*. Si es necesario agregar otros nodos.
- Ejecutar la consulta (*Run Query*).

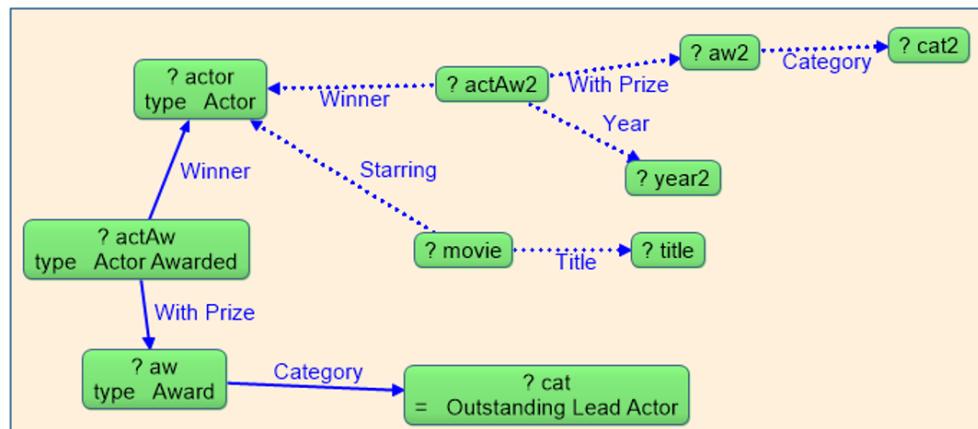


Figura 7. Segunda consulta de ejemplo.

Tercera consulta: Se requiere los nombres de los actores que han trabajado en películas de género "Comedy" y también han trabajado bajo la dirección de "Robert Zemeckis". (Observe que pudieron ser películas diferentes, pero el actor cumple con ambas condiciones).

Hacer notar que este caso es diferente al primer ejemplo porque el actor debe cumplir las dos condiciones, pero pueden ser películas diferentes para cada

condición. Por lo tanto, en el patrón se debe representar la posibilidad de que sean dos películas diferentes.

En el *Graphical Query View* (ver Figura 8):

- Identificar los tipos de objeto que hay en la consulta: actores, películas, y directores. Crear un nodo variable para cada uno, y especificar como filtro del nodo el tipo correspondiente.
- Identificar los datos que interesan de cada objeto: el nombre de los actores.
- Identificar las condiciones sobre los datos: el género de las películas y el nombre del director. **En este caso se deben crear dos nodos tipo *Movie***, uno por cada condición, ya que las condiciones del actor se pueden cumplir, cada una, con una película diferente.
- Agregar los arcos entre los nodos. Para identificar que arcos usar, consultar el *graph view*. Agregar otros nodos si es necesario.
- Ejecutar la consulta (*Run Query*).

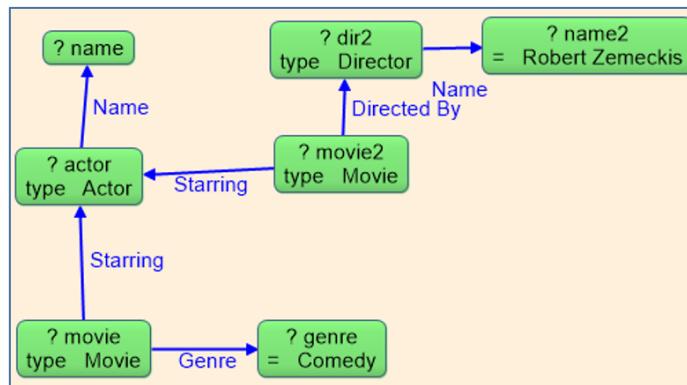


Figura 8. Tercera consulta de ejemplo.

5. Actividades

Presentación de los datos

Para realizar las actividades se usará una base de datos con información de la historia clínica de pacientes. Esta información incluye los pacientes, los médicos, los encuentros entre pacientes y médicos (i.e. consulta externa, consulta en urgencias, cirugía...), los síntomas, los diagnósticos realizados, las formulas médicas recetadas, la historia familiar registrada, resultados de exámenes, y los procedimientos realizados. Estos datos son ficticios, fueron generados aleatoriamente, por lo tanto no encontrarán en ellos coherencia desde el punto de vista clínico (por ejemplo, no habrá correspondencia lógica entre los medicamentos recetados y el diagnóstico).

Los datos incluyen:

- Los pacientes son de tipo (*type*) *Patient*, y tienen: identificación (*id*), nombre (*firstName*), apellido (*lastName*), y datos demográficos que incluyen género (*gender*), fecha de nacimiento (*dateOfBirth*), dirección (*address*), teléfono (*telephone*), y grupo étnico (*ethnicGroup*).
- Los médicos son de tipo (*type*) *Physician*, y tienen: identificación (*id*), nombre (*firstName*), apellido (*lastName*), y especialidad (*specialty*).
- Los encuentros son de tipo (*type*) *Encounter* y tienen: tipo de encuentro (*encounterType*), fecha de encuentro (*encounterDate*), paciente (*of*), médico que atendió (*attendedBy*), enfermedad actual (*presentIllness*), revisión de sistemas (*reviewOfSystems*), historia médica anterior (*medicalHistory*), historia médica familiar (*familyHistory*), diagnóstico (*diagnosis*), procedimientos (*procedure*), orden de medicinas (*medicationOrder*), exámenes de laboratorio (*Test*), y exámenes de imágenes diagnósticas (*diagnosticImage*).
- La descripción de la enfermedad actual (*presentIllness*) incluye los síntomas (*type Symptom*). Los síntomas tienen: código (*code*), descripción (*description*), ubicación (*location*), duración (*duration*), y severidad (*severity*).
- La revisión de sistemas (*type reviewOfSystems*) incluye el sistema (*system*), y la observación (*observation*).
- La historia médica anterior (*type medicalHistory*) incluye las enfermedades diagnosticadas (*disease*) y procedimientos previos (*procedure*).
- La historia médica familiar (*type familyHistory*) incluye la relación familiar con la persona que presentó la enfermedad (*relationship*) y la enfermedad que presentó (*disease*).
- El diagnóstico (*type Diagnosis*) hace referencia al ICD10, y tiene un código (*code*) y una descripción (*description*).
- La orden de medicinas (*type medicationOrder*) tiene la medicina (*type Medication*), la dosis (*dose*), y las instrucciones (*instructions*).
- La medicina (*type Medication*) tiene el código (*code*) y la descripción (*description*).
- Los procedimientos (*type Procedure*) tienen código (*code*), descripción (*description*), y resultados (*result*).
- Los exámenes de laboratorio (*type Test*) tienen código (*code*), descripción (*description*), y resultados (*result*).
- Los exámenes de imágenes diagnósticas (*type diagnosticImageStudy*) tienen código (*code*), descripción (*description*), resultados (*result*), y las imágenes (*type Image*) con el archivo de la imagen (*imageFile*) y las anotaciones sobre la

misma (*annotations*). Las anotaciones son texto (palabras clave) que identifican los resultados encontrados en la imagen.

Adicionalmente, los pacientes, médicos, diagnósticos, síntomas, medicinas, exámenes y procedimientos tienen el dato *label*, que es igual al nombre en el caso de los pacientes, médicos y medicinas; y es igual a las descripciones en los otros casos.

Se debe tener en cuenta que los datos suelen estar incompletos. Por ejemplo, en un encuentro se registra la realización de un procedimiento, y en otro se registran síntomas y diagnóstico.

Actividades de evaluación

Las actividades que se van a realizar en esta prueba se ambientan en dos escenarios:

- Escenario I: Usted es un médico y está en su consultorio atendiendo la consulta de un paciente. Requiere consultar datos de la historia clínica de ese paciente.
- Escenario II: Usted es un médico que va a iniciar una investigación y está buscando historias clínicas de pacientes que cumplen ciertas características.

Las actividades son:

1. Se requieren las enfermedades (incluyendo su descripción) registradas en la historia familiar del paciente con identificación 723628; incluir la relación familiar si está disponible. (*Por favor, recuerde iniciar la grabación antes de empezar a formular la consulta*)
2. Se requiere encontrar los diagnósticos (incluyendo su descripción) dados al paciente con identificación 723628 por médicos con especialidad en "*ENDOCRINOLOGY*" y, si está disponible, las fechas de las consultas y las medicinas que le recetaron en ese momento (incluyendo descripción y dosis). (*Por favor, recuerde iniciar la grabación antes de empezar a formular la consulta*)
3. De los pacientes que han tenido un diagnóstico de "*OSTEOMYELITIS*" desde el año 2000 (desde 01-01-2000), se requieren todos los datos de: identificación del paciente, diagnósticos recibidos, y procedimientos que se le han realizado (incluir la descripción de los diagnósticos y procedimientos). (Note que estos datos incluyen los otros diagnósticos, además de *OSTEOMYELITIS*, y todos los

datos que se piden son independientes de la fecha en que se registraron). (*Por favor, recuerde iniciar la grabación antes de empezar a formular la consulta*)

4. Se requiere encontrar los pacientes que han tenido un estudio de imágenes diagnósticas con anotación "*GALLSTONES*" y se les ha realizado el procedimiento "*COLECTOMY*". De estos pacientes se requiere la fecha de nacimiento, género y raza, si están disponibles. (Observe que estos resultados pudieron realizarse en encuentros diferentes, pero el paciente cumple con ambas condiciones). (*Por favor, recuerde iniciar la grabación antes de empezar a formular la consulta*)

6. **Cuestionario.** Solicitar al participante responder el cuestionario post-test.

7. **Espacio para que los participantes expresen sus dudas e inquietudes con respecto a la prueba.**

Guion de la Prueba para herramienta 2.

El propósito de esta prueba es tener la opinión del usuario sobre el uso de una herramienta de consulta de datos. Esta prueba hace parte de un test en el que se comparan dos herramientas, la mitad de los participantes evalúan la primera herramienta y la otra mitad, la segunda herramienta. No hay respuestas correctas o incorrectas, todo lo que el participante haga y opine nos ayudará a mejorar el diseño de nuestra herramienta.

La prueba incluye: a) Registro de los datos del participante y firma del consentimiento para realizar la prueba, b) Presentación de la herramienta a evaluar, c) Realización de 4 actividades. El participante dispondrá de un tiempo máximo de 15 minutos. El participante puede usar el manual y la ayuda de la herramienta, también solicitar ayuda al moderador o hacer preguntas sobre las actividades. El participante puede solicitar al moderador la traducción de los textos de la herramienta.

A continuación se registran los datos del participante y se firma el consentimiento para realizar la prueba.

- 1. Registro de los datos del participante (*screener questionnaire*) y firma el consentimiento para realizar la prueba.**
- 2. Espacio para que los participantes expresen sus dudas e inquietudes con respecto a la prueba.**

3. Datos para la demostración de la herramienta

Para mostrar el funcionamiento de la herramienta se usará una base de datos con información de películas y series de televisión. Esta información incluye los directores de las películas o series, los actores del reparto, las ciudades de nacimiento de los actores, y los premios recibidos por las películas, las series o los actores. Algunos de estos datos son reales (ej. Los títulos de las películas, los nombres de los actores) y otros son ficticios (ej. Las fechas, los géneros y los premios).

La información que se tiene es:

- Las películas (*Movie*) tienen: título (*title*), el año en que se estrenó (*year*), el género (*genre*), un comentario (*comment*), quien la dirigió (*directedBy*), y el reparto (*cast*).
- Las series de televisión (*TVSerie*) tienen: título (*title*), el año en que se estrenó (*year*), el género (*genre*), un comentario (*comment*), y el reparto (*cast*).

-
- Los directores (*Director*) tienen: nombre (*name*), fecha de nacimiento (*dateOfBirth*), y comentario (*comment*).
 - Los actores (*Actor*) tienen: nombre (*name*, este es el nombre artístico), nombre real (*realName*), fecha de nacimiento (*dateOfBirth*), lugar de nacimiento (*placeOfBirth*) y un comentario (*comment*).
 - Los premios (*Award*) tienen: nombre (*name*) y categoría (*category*). El nombre, es el nombre del premio, por ejemplo "Academy Awards", o "Emy Awards". Cada premio se da en varias categorías, por ejemplo: "*Best Actress in a Leading Role*", o "*Best Picture Academy Award*". Dado que el mismo premio y categoría se da a diferentes actores en diferentes años, *ActorAwarded* tiene la información de que premio (*withPrize*) se entregó a que actores (*winner*) y en qué año (*year*). De la misma manera, *MovieAwarded* o *SerieAwarded* tienen la información de que premio (*withPrize*) se otorgó a que película o serie (*winner*) y en qué año (*year*).
 - Las ciudades (*City*) tienen nombre (*name*) y comentario (*comment*).

Además, se debe tener en cuenta que los datos de una película o serie particular pueden estar incompletos. Por ejemplo, no todas las películas tienen género, o han sido galardonadas con un premio, o tienen registrado un director. Lo mismo sucede con los datos de los actores, directores y premios.

Los datos se representan con grafos, los nodos con forma de rectángulo representan objetos; los nodos con forma de óvalo, una característica o atributo de un objeto; los nodos con forma de círculo agrupan datos; y los arcos (líneas que conectan los nodos) representan una relación entre los nodos. La Figura 1 muestra un ejemplo de un grafo con datos donde, por ejemplo, la película 1 (Movie_1) tiene como título "*The King's Speech*" y es del año 2010.

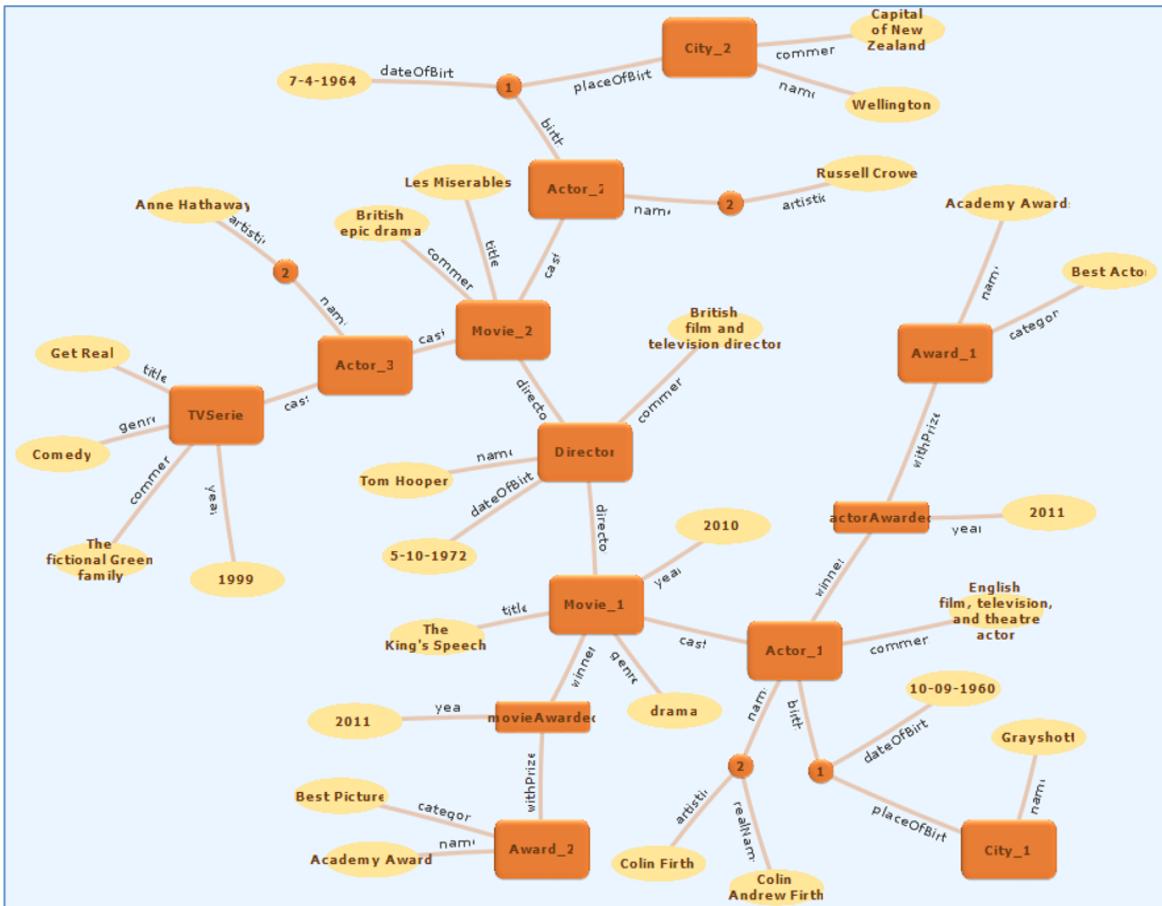


Figura 1. Ejemplo de un grafo con datos

4. Demostración de la herramienta

Este es un prototipo, la herramienta no está completamente implementada, por esto no entrega los resultados de las consultas.

La Figura 2 muestra la herramienta. La **vista principal** muestra la organización general de los datos, estos no son los datos particulares de un paciente, un médico, o una medicina, sino una descripción de los datos que puede tener objeto de ese tipo. La **vista general** permite hacer desplazamiento y zoom de la vista principal. Los **operadores** permiten transformar el grafo para seleccionar la porción de datos que interesa. Los **controles** permiten limpiar la selección, devolver un paso atrás, devolver el grafo al estado inicial, hacer zoom y ejecutar la consulta. El historial despliega la lista de operaciones que se han realizado sobre el grafo.

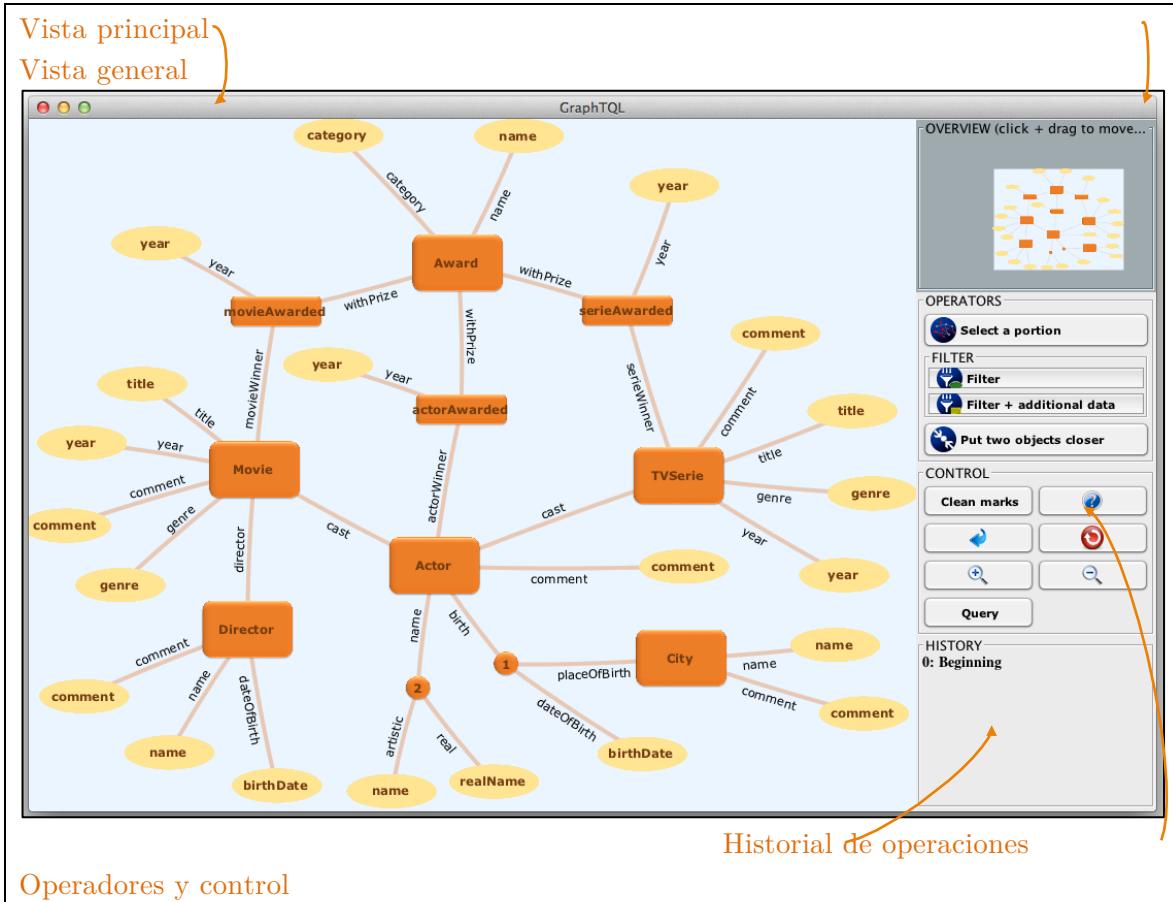


Figura 2. Pantalla de la herramienta.

Cuando se necesita recuperar información almacenada, generalmente se requieren partes específicas de los datos. En el caso de los grafos, una manera de encontrar esos datos es transformar el grafo, reduciéndolo para dejar solamente los datos que interesan.

■ **Operación de la vista principal:**

- Para seleccionar nodos y arcos basta con hacer clic sobre el mismo (el objeto u arco se torna rojo). Si se hace clic sobre un nodo o arco ya seleccionado, se quitar la marca.
- El clic derecho sobre un nodo rectangular permite marcarlo como clase de interés. Clic derecho sobre el nodo de interés lo marca como nodo seleccionado.
- El clic derecho sobre un nodo ovalado permite seleccionarlo como clase de interés o definir los filtros sobre el nodo.

- **Operación de la vista general (*overview*):**
 - Con clic se arrastra la región azul, movimiento que se replica en la vista principal.
 - Con el *scroll* se amplía (o reduce) el tamaño del grafo.

Operadores

- **Select a portion** (Seleccionar porción): Elegir una porción del grafo que tiene los datos que interesan. Por ejemplo, si interesa la fecha de nacimiento, el nombre y el nombre real de los actores, las categorías de los premios, y la fecha en que se han otorgado los premios a los actores, se marcan los nodos y arcos entre ellos (Figura 3), se da clic en el botón "Select a portion", y se obtiene la porción que aparece en Figura 4-(a). Si la consulta se ejecutara sobre los datos de la Figura 1, el resultado sería la Figura 4-(b).

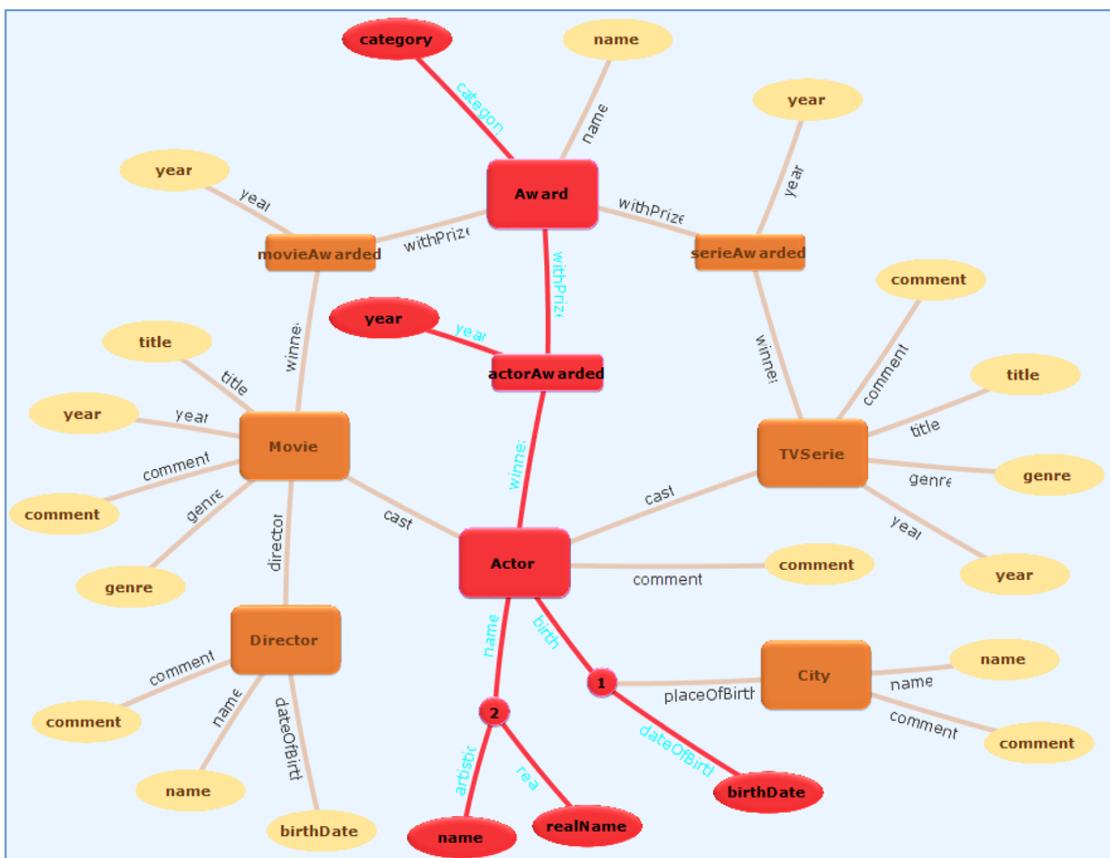


Figura 3. Nodos marcados para *Select a portion*.

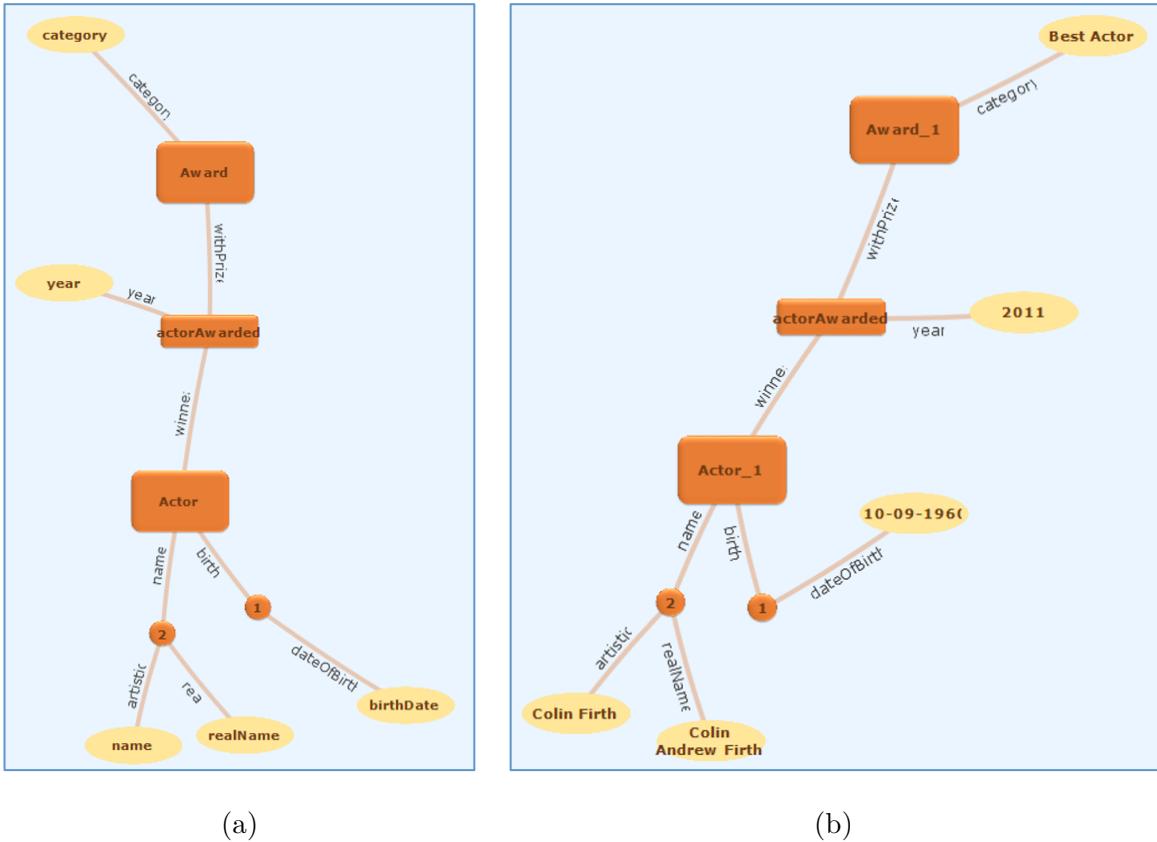
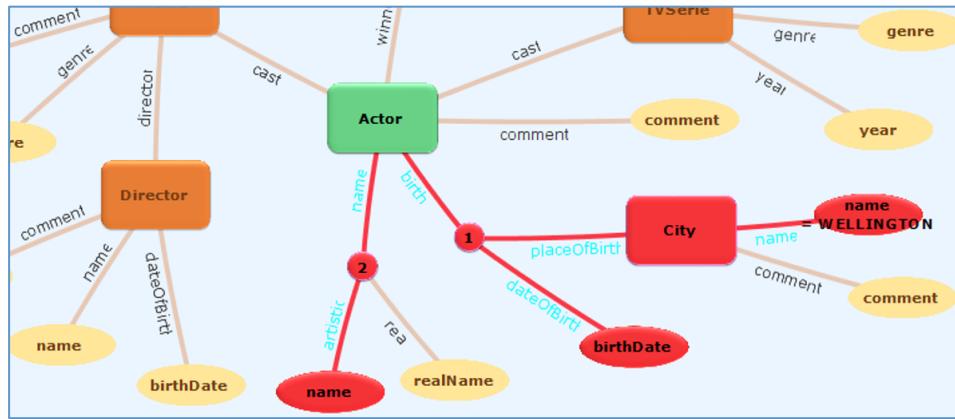
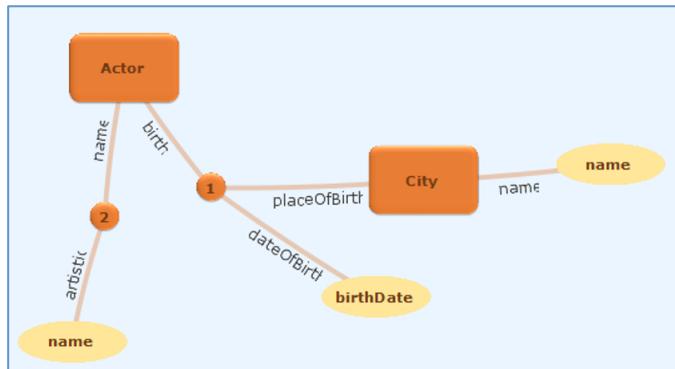


Figura 4. Resultado *Select a portion*.

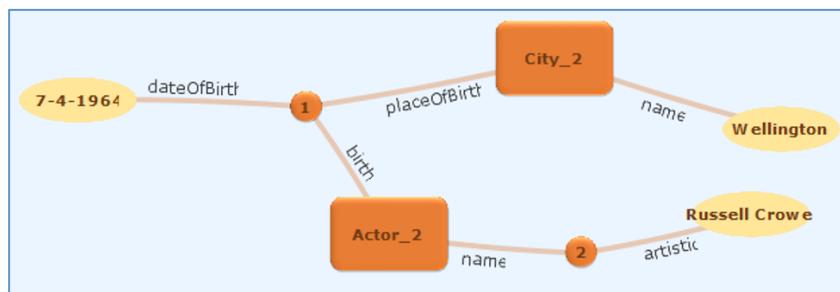
- **Filter (Filtro):** Seleccionar los objetos de cierto tipo que cumplen una condición específica, por ejemplo si interesa el nombre y fecha de nacimiento de los actores nacidos en *New York*. Para ello se selecciona un objeto de interés (actor), la condición que debe cumplir ese objeto (nacido en "Wellington") y los datos que se requieren (nombre y fecha de nacimiento). Estos últimos datos aparecen en el resultado si están disponibles. (Ver: Figura 5-(a), formulación de la consulta; Figura 5-(b), resultado-grafo general; Figura 5-(c), ejemplo datos de resultado).



(a)



(b)



(c)

Figura 5. Ejemplo Filter.

- **Filter+additional data** (Filtro+datos adicionales): Seleccionar **todos** los datos marcados de un objeto que cumple una condición. Por ejemplo, en la Figura 6, seleccionar todos los datos marcados de los directores que han trabajado con el actor “*Colin Firth*”. “**Todos los datos**” incluyen otros actores con los que ha trabajado el director. Este operador se diferencia de *Filter* en que recupera todos los datos de los objetos que cumplen la condición, esto es: datos diferentes a la condición (ej. otros actores). Un ejemplo del resultado se observa en la Figura 7.

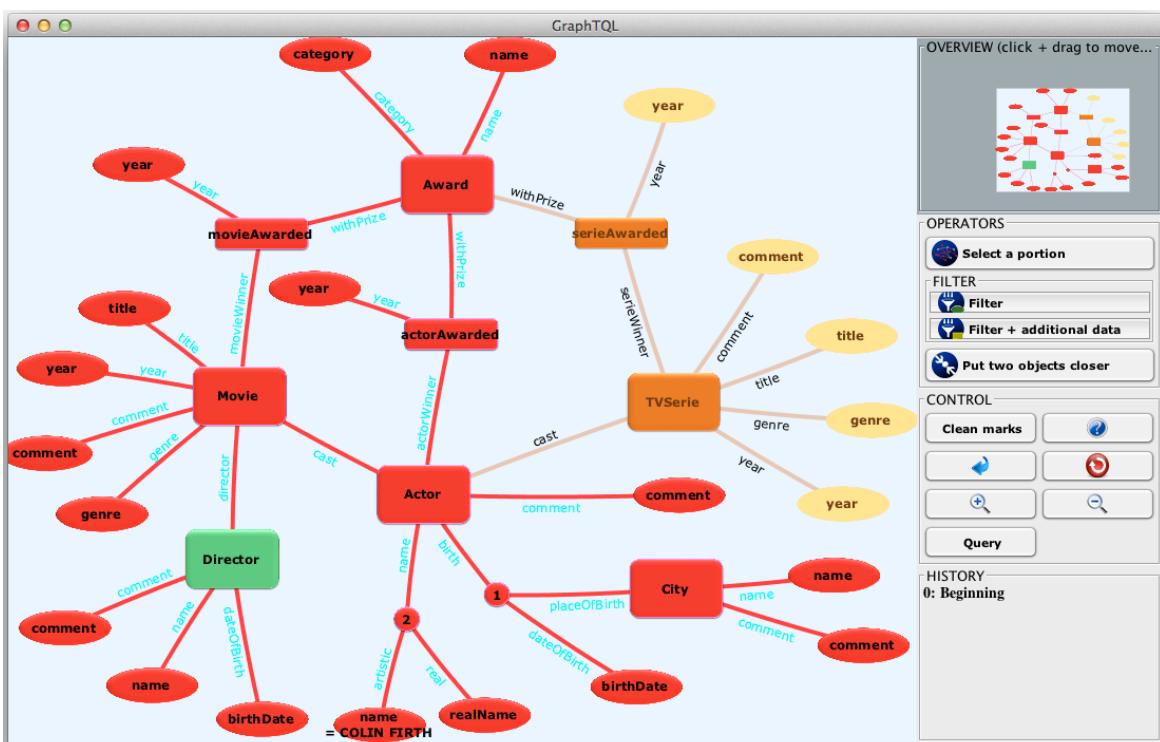


Figura 6. Ejemplo *Filter+additional data*.

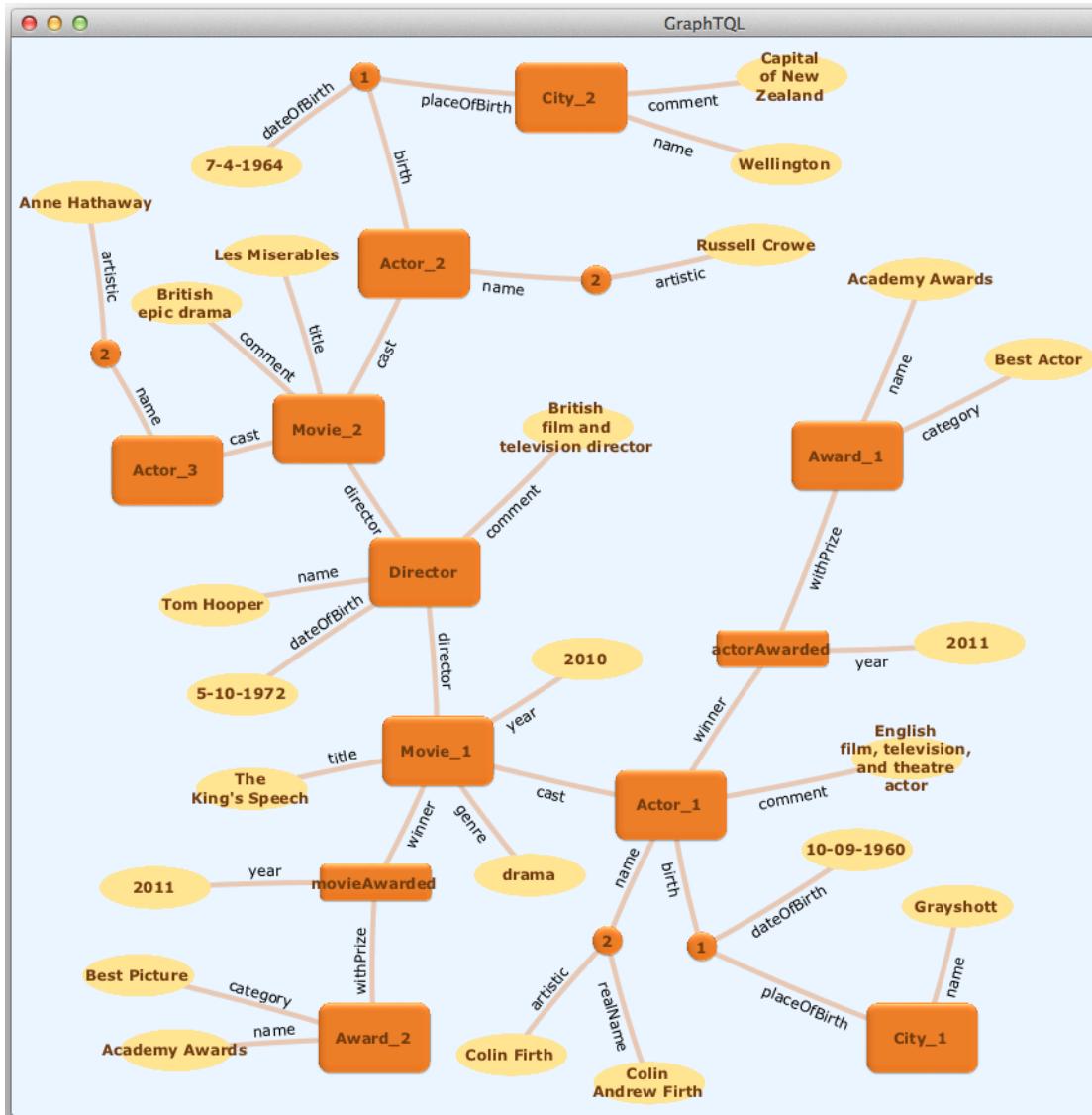
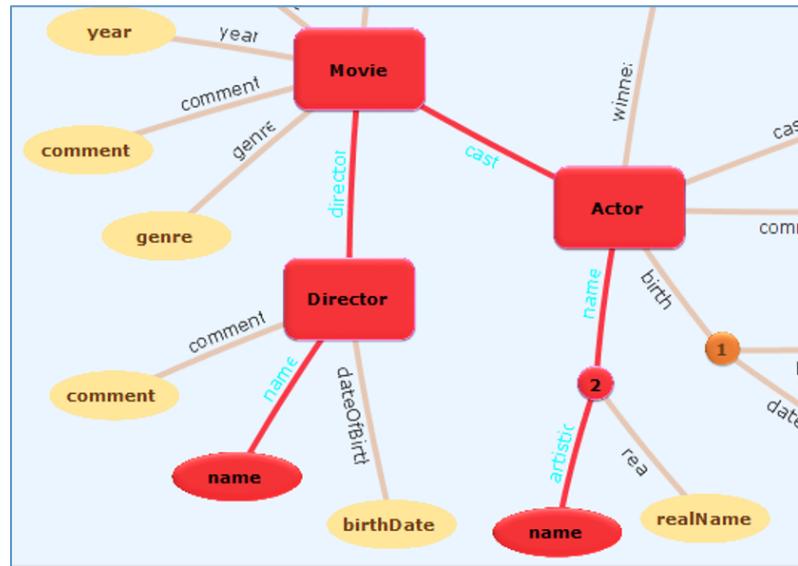
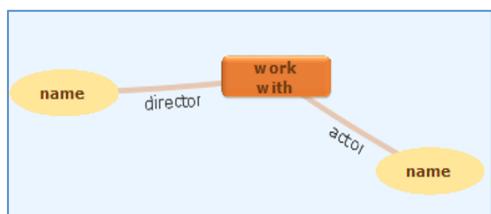


Figura 7. Resultado *Filter+additional data*.

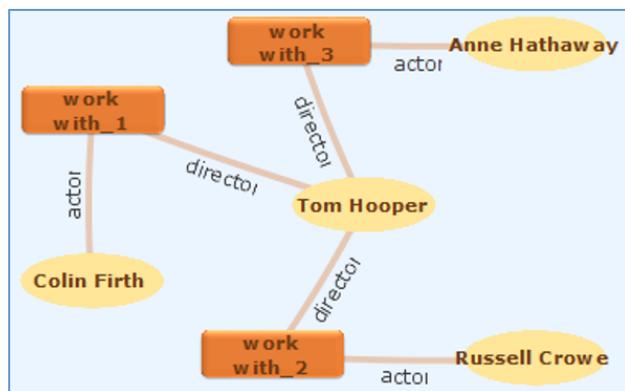
- ***Put two objects closer*** (Acercar objetos): Acercar dos nodos en el grafo reemplazando el camino que los une. En este caso no hay interés en los datos del camino, éstos desaparecen del grafo. Por ejemplo si se quiere acercar el nombre del director con el nombre de los actores que ha dirigido. (ver: Figura 8-(a), formulación de la consulta; Figura 8-(b), resultado-grafo general; Figura 8-(c), ejemplo datos de resultado).



(a)



(b)



(c)

Figura 8. Ejemplo *Put two objects closer*.

Consultas de Ejemplo

Primera consulta: Se requiere encontrar los títulos de las películas galardonadas con la categoría “BEST PICTURE” del premio “ACADEMY AWARD” a partir del año 2000. Además, se desea incluir el nombre de los actores y el género de la película, si están disponibles.

- Marcar el objeto de interés: título de la película.
 - Especificar la condición: la categoría “*BEST PICTURE*”, el nombre del premio *ACADEMY AWARD*, y el año del premio mayor a 2000.
 - Elegir el camino que especifica películas ganadoras.
 - Marcar otros nodos que interesa en la respuesta: género de la película y nombre de los actores.
 - Seleccionar el operador *Filter* y responder las preguntas de desambiguación.
(Ver Figura 9)

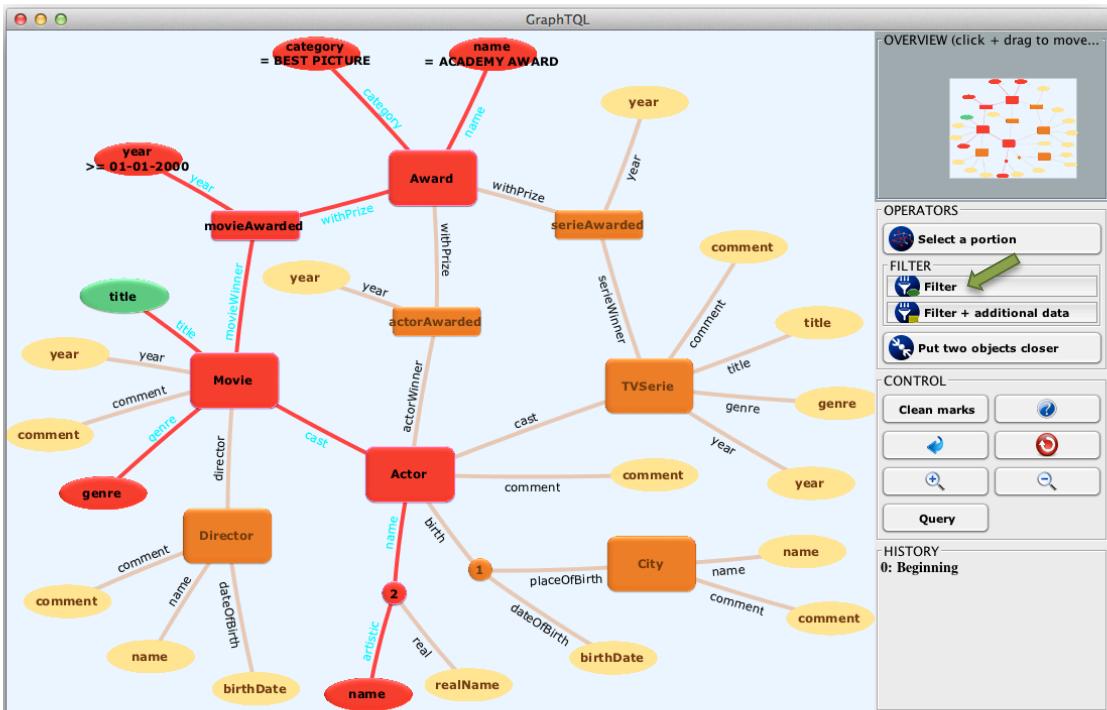


Figura 9. Primera consulta de ejemplo.

Segunda consulta: De los actores que han ganado un premio en la categoría "OUTSTANDING LEAD ACTOR", se requieren todos los datos disponibles de: títulos de las películas en que han actuado, categoría y año de los premios que han recibido. (Note que estos datos incluyen los otros premios, además de "OUTSTANDING LEAD ACTOR").

Hacer notar que este caso es diferente al primer ejemplo porque aquí se requieren los datos de los otros premios, además del premio de la condición. Esto se logra con un *Filter+additional data*.

- Marcar la clase de interés: actor.
- Especificar la condición: ganadores de un premio en la categoría = "OUTSTANDING LEAD ACTOR". Elegir el camino que representa los premios para actores.
- Marcar otros nodos que interesa en la respuesta: título de la película, categoría y año de los premios recibidos por el actor.
- Seleccionar el operador *Filter+additional data* y responder las preguntas de desambiguación.

(Ver Figura 10)

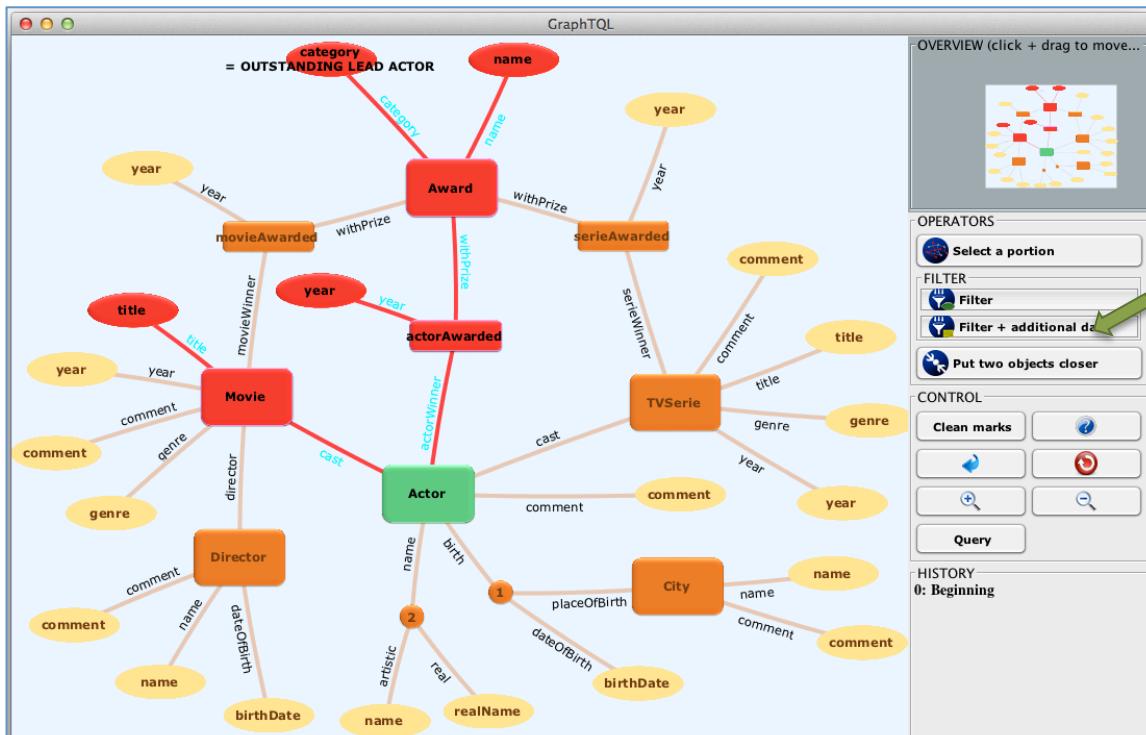


Figura 10. Segunda consulta de ejemplo

Tercera consulta: Se requieren los nombres de los actores que han trabajado en películas de género "COMEDY" y también han trabajado bajo la dirección de "ROBERT ZEMECKIS". (Observe que pudieron ser películas diferentes, pero el actor cumple con ambas condiciones).

Hacer notar que este caso es diferente al primer ejemplo porque el actor debe cumplir las dos condiciones, pero pueden ser películas diferentes para cada condición. Esto se debe tener en cuenta para responder a las preguntas de desambiguación del filtro.

- Marcar la clase de interés: nombre de actor.
 - Especificar las condiciones: actores que han trabajado en películas de género "COMEDY" y que también han trabajado bajo la dirección de "ROBERT ZEMECKIS".
 - Elegir el camino que representa que un actor trabajó en una película.
 - Seleccionar el operador *Filter* y responder las preguntas de desambiguación.
- (Ver Figura 11)

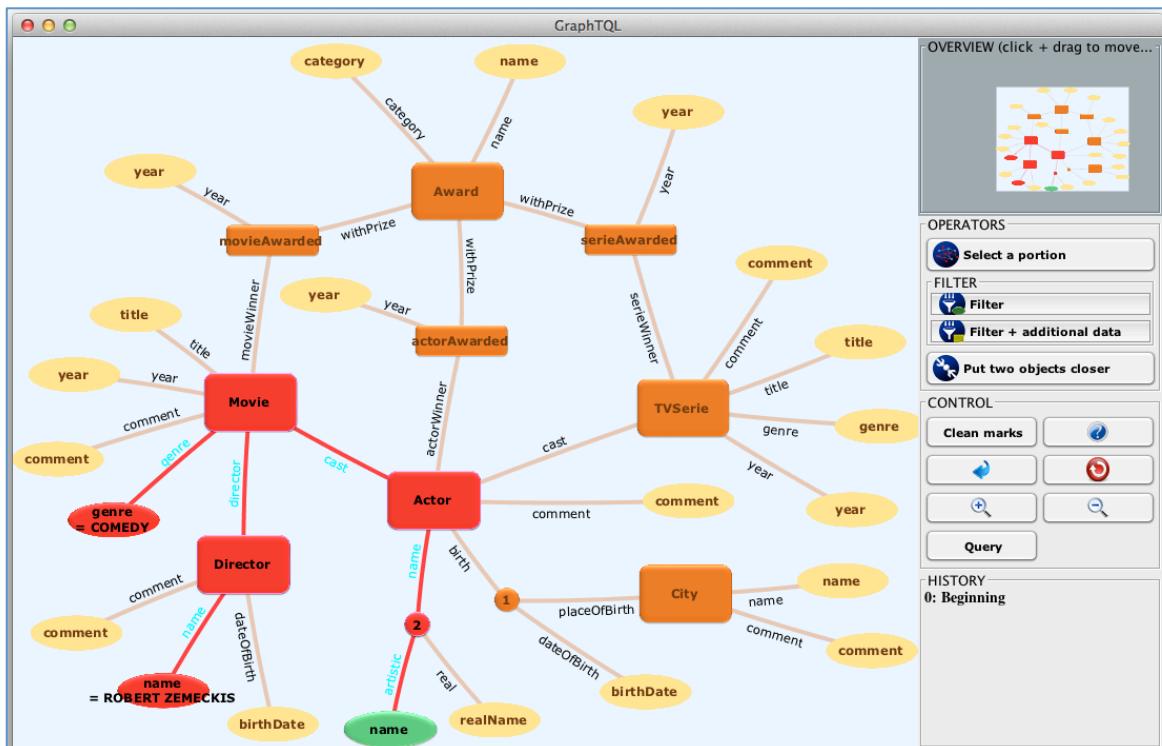


Figura 11. Tercera consulta de ejemplo.

5. Actividades

Presentación de los datos y escenarios

Para realizar las actividades se usará una base de datos con información de la historia clínica de pacientes.

Esta información incluye:

- Los pacientes (*Patient*) tienen: identificación (*id*), nombre (*firstName*), apellido (*lastName*), y datos demográficos que incluyen género (*gender*), fecha de nacimiento (*dateOfBirth*), dirección (*address*), teléfono (*telephone*), y raza (*race*).
- Los médicos (*Physician*) tienen: identificación (*id*), nombre (*firstName*), apellido (*lastName*), y especialidad (*specialty*).
- Los encuentros (*Encounter*) incluyen las consultas en consultorio, por urgencias, visitas domiciliarias, hospitalización, etc. Estos tienen: tipo de encuentro (*encounterType – Ambulatory, inpatient, outpatient,...*), fecha del encuentro (*encounterDate*), el paciente (*of*), el médico que atendió (*attendedBy*), la enfermedad actual (*presentIllnes*), la revisión de sistemas (*reviewOfSystems*), la historia médica familiar (*familyHistory*), el diagnóstico (*diagnosis*), las órdenes de medicinas (*medicationOrder*), los exámenes de laboratorio (*Test*), los procedimientos (*Procedure*), y los exámenes de imágenes diagnósticas (*Diagnostic Image Study*).
- La descripción de la enfermedad actual (*presentIllnes*) incluye los síntomas (*Symptom*) que tienen: código (*code*), descripción (*description*), ubicación (*location*), duración (*duration*), y severidad (*severity*).
- La revisión de sistemas (*reviewOfSystems*) incluye el sistema (*system*) y la observación (*observation*).
- La historia médica familiar (*familyHistory*) incluye la relación familiar con la persona que presentó la enfermedad (*relationship*) y la enfermedad que presentó (*disease*).
- El diagnóstico (*Diagnosis*) tiene un código (*code*) y una descripción (*description*).
- La orden de medicinas (*type medicationOrder*) tiene la medicina (*Medication*), la dosis (*dose*), y las instrucciones (*instructions*).
- La medicina (*Medication*) tiene el código (*code*) y la descripción (*description*).
- Los procedimientos (*Procedure*) tienen código (*code*), descripción (*description*), y resultados (*result*).
- Los exámenes de laboratorio (*Test*) tienen código (*code*), descripción (*description*), y resultados (*result*).

-
- Los exámenes de imágenes diagnósticas (*Diagnostic Image Study*) tienen código (*code*), descripción (*description*), resultados (*result*), y las imágenes (*Image*) con el archivo de la imagen (*imageFile*) y las anotaciones sobre la misma (*annotations*). Las anotaciones son texto (palabras clave) que identifican los resultados encontrados en la imagen.

Se debe tener en cuenta que los datos suelen estar incompletos. Por ejemplo, en un encuentro se registra la realización de un procedimiento, y en otro se registran los síntomas y el diagnóstico.

Actividades de evaluación

Las actividades que se van a realizar en esta prueba se ambientan en dos escenarios:

- Escenario I: Usted es un médico y está en su consultorio atendiendo la consulta de un paciente. Requiere consultar datos de la historia clínica de ese paciente.
- Escenario II: Usted es un médico que va a iniciar una investigación y está buscando historias clínicas de pacientes que cumplen ciertas características.

Las actividades son:

1. Se requieren las enfermedades (incluyendo su descripción) registradas en la historia familiar del paciente con identificación 723628; incluir la relación familiar si está disponible. (*Por favor, recuerde iniciar la grabación antes de empezar a formular la consulta*)
2. Se requiere encontrar los diagnósticos (incluyendo su descripción) dados al paciente con identificación 723628 por médicos con especialidad en "*ENDOCRINOLOGY*" y, si está disponible, las fechas de las consultas y las medicinas que le recetaron en ese momento (incluyendo descripción y dosis). (*Por favor, recuerde iniciar la grabación antes de empezar a formular la consulta*)
3. De los pacientes que han tenido un diagnóstico de "*OSTEOMYELITIS*" desde el año 2000 (desde 01-01-2000), se requieren todos los datos de: identificación del paciente, diagnósticos recibidos, y procedimientos que se le han realizado (incluir la descripción de los diagnósticos y procedimientos). (Note que estos datos incluyen los otros diagnósticos, además de *OSTEOMYELITIS*, y todos los

datos que se piden son independientes de la fecha en que se registraron). (*Por favor, recuerde iniciar la grabación antes de empezar a formular la consulta*)

4. Se requiere encontrar los pacientes que han tenido un estudio de imágenes diagnósticas con anotación "*GALLSTONES*" y se les ha realizado el procedimiento "*COLECTOMY*". De estos pacientes se requiere la fecha de nacimiento, género y raza, si están disponibles. (Observe que estos resultados pudieron realizarse en encuentros diferentes, pero el paciente cumple con ambas condiciones). (*Por favor, recuerde iniciar la grabación antes de empezar a formular la consulta*)

6. **Cuestionario.** Solicitar al participante responder el cuestionario post-test.

7. **Espacio para que los participantes expresen sus dudas e inquietudes con respecto a la prueba.**

Registro de Sesión

PRUEBA COMPARATIVA - HERRAMIENTAS DE CONSULTA SOBRE MODELOS DE GRAFOS

Moderador _____

Fecha: _____ Hora: _____

Registro de actividades

Consulta	Hora Inicio	Hora Fin	Número de Errores	Número de veces que solicita ayuda		
				Clase A	Clase B	Clase C
1						
2						
3						
4						

Otras observaciones

Actividad 1:

Actividad 2:

Actividad 3:

Actividad 4:

Errores: retorna a un paso anterior, cambia una acción aplicada antes.

Ayudas. **Clase A:** El moderador da pistas o genera preguntas que llevan al participante a repensar la solución que está proponiendo. **Clase B:** El moderador aclara o explica de nuevo la semántica de las operaciones. **Clase C:** Consulta de manuales o ayudas de la herramienta y las ayudas en las que el moderador explica el funcionamiento de la herramienta (diferente a las operaciones) o aclara lo que se busca recuperar en la consulta de la actividad.

Cuestionario *post-test*

Lea cuidadosamente las siguientes afirmaciones y califique si está de acuerdo o en desacuerdo con ellas, según la escala que se ofrece.

	Totalmente de acuerdo 5	4	3	2	Totalmente en desacuerdo 1
Me gustaría usar este producto frecuentemente	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Me parece que la herramienta es innecesariamente compleja	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Pienso que la herramienta es fácil de usar	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Creo que voy a necesitar el apoyo de un técnico para poder usar la herramienta	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Me parece que las funciones de esta herramienta están bien integradas	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Pienso que hay muchas inconsistencias en la herramienta	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Diría que la mayoría de las personas aprenderían a usar la herramienta muy rápido	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Me pareció que la herramienta es difícil de usar	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Me sentí muy confiado usando la herramienta	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Tendría que aprender muchas cosas antes de poder usar esta herramienta	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Comentarios adicionales sobre esta experiencia de uso de la herramienta					

Anexo H. Resumen de la prueba comparativa

Este anexo presenta un resumen de la aplicación de la prueba comparativa. Al finalizar cada actividad, el moderador explicó a los participantes los errores que había detectado en la formulación de la consulta de esa actividad.

Evaluación de *Gruff*

Primer Participante

Este participante es estudiante de medicina y también es programador de *software*.

- **Actividad 1:** El participante usó 4 minutos y 44 segundos para formular la consulta.

Construyó el patrón { ?pac ?id ?id . ?pac ?of ?enc . ?enc ?FamilyHistory ?his . ?his ?relationship ?rel filter { ?id = 723628 } }

El moderador le brindó al participante 4 veces ayuda sobre las opciones de menú (Clase C).

Errores: (a) Dirección incorrecta del arco (?pac ?of ?enc). (b) Faltaron datos requeridos (enfermedad y descripción). (c) No marcó *relationship* como opcional.

- **Actividad 2:** El participante formuló la consulta en 9 minutos y 32 segundos.

Construyó el patrón { ?pac of ?en . ?pac id ?id . ?enc attendedBy ?med . ?med specialty ?espe . ?enc encounterDate ?date . optional { ?enc diagnosis ?diag } optional { ?enc medicationOrder ?medior } optional { ?medior Medication ?medi } optional { ?medior Dose ?dos } optional { ?medi Description ?descr } filter { ?id = 6801840 } filter { ?espe = ENDOCRINOLOGY } }

No requirió ayuda del moderador.

Errores: (a) Dirección incorrecta en arco (?pac ?of ?enc). (b) Faltaron datos requeridos (descripción de diagnóstico). (c) No marcó *encounterDate*, *medicationOrder*, *medication*, *description* y *dose* como opcionales.

- **Actividad 3:** El participante usó 8 minutos y 42 segundos para formular la consulta.

Construyó el patrón { ?encou Of ?pacientes . ?pacientes Id ?id . ?encou Diagnosis ?diag . ?encou EncounterDate ?date . ?diag Description ?des4 . ?diag Procedure ?proce . ?proce Description ?des1 . ?encou2 of ?pacientes . ?encou2 Diagnosis ?diag2 . ?diag2 Description ?des3 . ?diag2 Procedure ?proce2 . ?proce2 description ?des2 . filter { ?diag = OSTEOMYELITIS } filter { ?date >= 2000 } }

No requirió ayuda del moderador.

Errores: (a) Usó un camino incorrecto para conectar ?encou y ?proce, ?encou2 y ?proce2. (b) Faltaron datos requeridos (descripción de diagnosis). (c) Sobraron datos no marcados comoopcionales (?proce y ?des1). (d) No marcó *encounterDate*, *medicationOrder*, *medication*, *description* y *dose* como opcionales. (e) Ubicó la condición de diagnóstico sobre el objeto, no sobre la descripción.

- **Actividad 4:** El participante formuló la consulta en 11 minutos y 43 segundos.

Construyó el patrón { ?enco Of ?pac . OPTIONAL { ?pac Race ?raz } OPTIONAL { ?pac Gender ?gen } OPTIONAL { ?pac DateOfBirth ?nac } . ?enco DiagnositicImage ?imagdia Image ?image . ?image Annotation ?anota . ?enco2 Of ?pac . ?enco2 Procedure ?proce . ?proce Description ?desc Filter {?anota = INFLAMMATION OF GALLBLADDER} Filter {?desc = COLECTOMY} }

El moderador ayudó al participante a encontrar en el modelo el nodo anotación (Clase A).

Errores: (a) Usó un camino incorrecto para conectar ?pac con ?raz, ?gen y ?nac.

Segundo Participante

- **Actividad 1:** El participante usó 13 minutos y 46 segundos para formular la consulta.

Construyó el patrón { ?Patient ?id ?idPatient . ?Patient ?relationship ?famHistory . ?Encounter ?FamilyHistory ?famHistory . ?famHistory ?disease ?DiabetesMiellitus . ?DiabetesMiellitus ?descripcion ?DiabetesMiellitusTipoII filter { idPatient = 723628 } }

El moderador brindó ayuda en varias ocasiones para encontrar datos en el modelo (Clase B) y sobre las opciones de los menús (Clase C).

Errores: (a) Usó un arco inexistente entre Patient y famHistory. (b) No marcó relationship como opcional.

- **Actividad 2:** El participante formuló la consulta en 15 minutos y 47 segundos.

Construyó el patrón { ?Encounter001 of ?patient . ?Patient id ?id723628 . ?Encounter001 attendedBy ?Medico . ?Medico specialty ?endocrinologo . ?Encounter001 encounterDate ?19enero2015 . ?Encounter001 diagnosis ?DiabetesMiellitus . ?Encounter002 of ?Patient . ?Encounter002 attendedBy ?Medico2 . ?Medico2 specialty ?endocrinologo . ?Encounter002 medicationOrder ?medicationOrder001 . ?medicationOrder001 medication ?insulinaCristalina . ?Encounter002 diagnosis ?DiabetesMiellitus }

El moderador ayudó al participante tres veces, para encontrar en el modelo las conexiones de médico y paciente, el diagnóstico y la medicina (Clase A). El participante dedicó bastante tiempo a tratar de encontrar un encuentro con una orden médica. El moderador le recordó que puede buscar un nodo de tipo orden médica y desde allí buscar las relaciones con los otros nodos.

Errores: (a) Nombró las variables con datos y termina confundiendo variables y filtros, por esto no especificó filtros, puso esos datos en el nombre de la variable. Esto pudo ser debido a que copia el patrón de la vista de exploración del grafo de datos. (b) Crea dos variables para la clase encounter, lo que no era necesario en este caso. (c) No incluyó la descripción del diagnóstico, la descripción y dosis de las medicinas. (d) No marcó los datos opcionales.

- **Actividad 3:** El participante usó 10 minutos y 30 segundos para formular la

consulta.

Construyó el patrón { ?Encounter001 of ?paciente1 . ?paciente1 id ?c76319060 . ?Encounter001 diagnosis ?LiliasisRenal . ?Encounter001 diagnosis ?osteomielitis . ?Encounter001 encounterDate ?2enero2000 . ?Paciente2 diagnosis ?osteomielitis . ?paciente2 id ?25274616 . ?osteomielitis medicationOrder ?ordenMedica2 . ?ordenMedica2 medication ?vancomicina10mgcada6 filter { ?25274616 = 132} }

No requirió ayuda del moderador.

Errores: (a) Nombró las variables con datos, confundió variables y filtros. Puso los filtros solicitados en el nombre de la variable. (b) Puso un filtro adicional en *id* del *paciente2* (=132). (c) Crea dos variables para la clase *patient* y *diagnosis*. (d) Debía crear dos variables para la clase *encounter*. (e) No incluyó procedimiento y su descripción. (f) Incluyó *medication order* y *medication*, que no se habían solicitado, y no los marcó como opcionales. (g) Usó una relación inexistente entre *patient* y *diagnosis*. (h) No marcó los datos opcionales.

■ **Actividad 4:** El participante formuló la consulta en 12 minutos y 52 segundos.

Construyó el patrón { ?patient1 race ?mestizo . ?patient1 dateOfBirth ?5marzo1974 . ?patient1 gender ?masculino . ?encounter6 of ?patient1 . ?encounter15 of ?patient1 . ?encounter12 of ?patient1 . ?encounter12 diagnosticImage ?imagenDiagnostica . ?imagenDiagnostica result ?Gallstone . ?encounter200 result ?imagenDiagnostica . ?encounter200 of ?paciente2 . ?encounter106 of ?paciente2 . ?encounter106 of ?paciente2 . ?encounter230 of ?paciente2 . ?paciente2 gender ?femenino . ?paciente2 id ?19diciembre1977 . ?encounter230 procedure ?colectomy . ?encounter15 procedure ?colectomy }

No requirió ayuda del moderador.

Errores: (a) El participante incluyó 2 pacientes y 6 encuentros, la formulación de la consulta requería un paciente con dos encuentros y cada encuentro con una condición. (b) Ubicó los filtros como nombre de las variables. (c) Usó relaciones inexistentes para conectar el *patient1* con la raza, género y fecha de nacimiento. También para conectar *Encounter200* con *imagenDiagnostica*. (d) No incluyó la anotación de la imagen diagnóstica. No incluyó la descripción del procedimiento. (e) Incluyó el resultado de la imagen diagnóstica, que no se requería en la consulta, y no lo marcó como opcional. (f) No marcó los datos opcionales.

Tercer Participante

■ **Actividad 1:** El participante usó 12 minutos y 22 segundos para formular la consulta.

Construyó el patrón { ?paciente ? id ?id . ?paciente ?of ?encounter . ?encounter Diagnosis ?diagnosis . ?diagnosis Description ?descripcion . ?encounter ?familyHistory ?familia . ?familia Diagnosis ?diagnosticos OPTIONAL { ?familia ?relationship ?relacion } filter { ?id = 723628 } }

Durante la formulación el moderado explicó de nuevo las opciones de menú cada vez que el participante iba a usar una opción por primera vez (Clase C) y las estrategias para navegar en los datos y encontrar sus relaciones (Clase C).

Errores: (a) Dirección incorrecta del arco (`?paciente ?of ?encounter`). (b) Sobraron datos, agregó diagnósticos dados al paciente y no los marcó como opcionales. (c) Faltaron datos, no incluyó la descripción de la enfermedad asociada a la historia familiar.

- **Actividad 2:** El participante formuló la consulta en 14 minutos y 58 segundos. Construyó el patrón `{ ?paciente id ?id . ?encuentro ?of ?paciente . ?encuentro Diagnosis ?diagnostico . ?diagnostico Description ?descripcion . ?encuentro AttendedBy ?doctor . ?doctor Specialty ?especialidad optional { ?encounterDate ?fecha } optional { ?encuentro medicationOrder ?ordenmed } ?ordenmed Medication ?medicamento . ?ordenmed Dose ?dosis . ?medicamento ?description filter { ?id = 723628 } filter { ?especialidad = ENDOCRINOLOGY }` Durante la formulación el moderado explicó de nuevo algunas opciones de menú (Clase C), menos que en la primera actividad.

Error: (a) Usó el nombre `?descripcion` para la descripción del diagnóstico y para la del medicamento. Esto en la consulta significa que ambos deben ser iguales. (b) Faltó marcar como opcionales el medicamento, su descripción y dosis.

- **Actividad 3:** El participante usó 11 minutos y 38 segundos para formular la consulta.

Construyó el patrón `{ ?paciente id ?id . ?encuentro ?of ?paciente . ?encuentro EncounterDate ?fecha2 . ?encuentro Diagnosis ?diagnostico . ?encuentro2 ?of ?paciente OPTIONAL { ?encuentro EncounterDate ?fecha3 } ?encuentro2 Of ?Procedimiento . ?Procedimiento Description ?descripcion . ?encuentro2 Diagnosis ?diagnosticos . ?diagnosticos Description ?descripcion . filter { ?fecha2 >= 01-01-2000 } filter { ?diagnostico = OSTEOMYELITIS } }`

No requirió ayuda del moderador.

Errores: (a) Ubicó la condición de filtro en el nodo diagnóstico (el objeto) no sobre descripción (el atributo). (b) Usó una relación inexistente entre `?encuentro` y `?Procedimiento (Of)`. (c) Usó el nombre `?descripcion` para la descripción del Procedimiento y para la del Diagnóstico, esto en la consulta significa que ambos deben ser iguales. (d) Faltó marcar como opcionales el diagnóstico y el procedimiento asociados al `?encuentro2`, sino la consulta exige que ambos ocurran en el mismo encuentro.

Comentarios: No borró la consulta anterior para dejarla como modelo de referencia y evitar ir al *graph view* para explorar los datos.

- **Actividad 4:** El participante formuló la consulta en 6 minutos y 58 segundos. Construyó el patrón `{ OPTIONAL { ?paciente Demographics ?demog } ?demog DateOfBirth ?fechanac . ?demog Race ?raza . ?demog Gender ?genero ?encuentro2 ?of ?paciente . ?encuentro2 Procedure ?procedimiento . ?procedimiento De-`

```
scription ?descripcion . ?encuentro ?of ?paciente . ?encuentro DiagnosticImage
?imagen_dx . ?imagen_dx Image ?imagen . ?imagen Annotation ?anotacion filter
{ ?descripcion=COLECTOMY } filter { ?anotacion=GALLSTONES } }
```

No requirió ayuda del moderador.

Errores: (a) Faltó marcar como opcionales el genero, la raza y la fecha de nacimiento.

Cuarto Participante

- **Actividad 1:** El participante usó 13 minutos y 56 segundos para formular la consulta.

Construyó el patrón { ?a Id ?723628 . ?encounter Of ?a . ?encounter FamilyHistory ?famil_h . ?famil_h Disease ?descr . ?descr Description ?des . ?famil_h Relationship ?rel filter { ?723628 = 723628 } } .

El moderador explicó de nuevo algunas opciones de los menus y las estrategias para navegar los datos (Clase C). También hizo algunas aclaraciones sobre el modelo de datos (Clase B).

Errores: (a) Faltó marcar como opcional la relación familiar.

- **Actividad 2:** El participante formuló la consulta en 12 minutos y 6 segundos.

Construyó el patrón { ?pac id ?id . ?encounter ?of ?pac OPTIONAL { ?encounter EncounterDate ?FECHA } ?encounter Diagnosis ?dx . ?encounter AttendedBy ?med . ?med Specialty ?esp OPTIONAL { ?encounter medicationOrder ?med_or } OPTIONAL { ?med_or Medication ?med } filter { ?id = 723628 } filter { ?esp = ENDOCRINOLOGY } }

El moderador brindó ayuda para encontrar algunos datos (Clase B).

Errores: (a) Usó el nombre ?med para la medicina y para el médico. En la consulta esto significa que ambos deben ser iguales. (b) Faltó incluir la descripción del diagnóstico y la descripción y dosis de la medicina.

- **Actividad 3:** El participante usó 12 minutos y 21 segundos para formular la consulta.

Construyó el patrón { ?e Of ?id . ?e EncounterDate ?FECHA . ?encuentro Diagnosis ?diagnostico . ?e Procedure ?PROCED . ?e Diagnosis ?DX_OTRO . ?e Diagnosis ?diagnosi filter { ?FECHA >= 01-01-2000 } filter { ?diagnosi = OSTEOMYELITIS } }

No requirió ayuda del moderador.

Errores: (a) Ubicó el filtro del diagnóstico en el objeto (Diagnosis) y no en el atributo (description). (b) Faltó incluir la descripción de los diagnósticos y los procedimientos. (c) Faltó incluir una segunda variable para encuentros, para agregar encuentros que tienen los otros diagnósticos y los procedimientos.

- **Actividad 4:** El participante formuló la consulta en 10 minutos y 31 segundos.

Construyó el patrón { ?ENCUENTRO Of ?PACIENTE OPTIONAL { ?PA-

```
CIENTE Demographics ?DEMO } OPTIONAL { ?DEMO Race ?RAZA } OPTIONAL { ?DEMO DateOfBirth ?NAC } OPTIONAL { ?DEMO Gender ?GEN } ?ENCUENTRO Procedure ?PROCED . ?PROCED Description ?COL . ?ENCUENTRO DiagnosticImage ?IMAGEN_DX . ?IMAGEN_DX Result ?RESULTADO filter { ?COL = COLECTOMY } filter { ?RESULTADO = GALLSTONES }
```

No requirió ayuda del moderador.

Errores: (a) Ubicó el filtro de anotación en el atributo resultado. (b) Especificó las condiciones conectadas al mismo encuentro, debían estar en encuentros diferentes.

Evaluación de GraphTQL

Los dos primeros participantes evaluaron una versión del prototipo de GraphTQL en la que el operador *Filter* aparece en el botón *Simple Filter* y *Filter+Additional Data* en el botón *All-Data Filter*.

Primer Participante

- **Actividad 1:** El participante usó 8 minutos y 28 segundos para formular la consulta.

Eligió *diagnosis* como clase de interés. Puso filtro en *id* (de *patient*) (= 723628). Marca *relationship*. Marcó el camino { *Encounter*, *familyHistory*, β_3 , *disease*, *Diagnosis* }. Seleccionó *Simple filter*.

El moderador explicó los caminos que propone la herramienta cuando el participante marca los nodos *diagnosis* y *relationship*, porque estos caminos resultan un poco extraños (Clase B).

La formulación fue correcta.

- **Actividad 2:** El participante formuló la consulta en 3 minutos y 22 segundos.

Marcó *Diagnosis* como clase de interés. Agregó filtro sobre *id* (de *patient*) (= 723628), *specialty* (de *physician*) (=ENDOCRINOLOGY). Marcó *date* (de *encounter*), *description* (de *medication*), y *dose*. Eligió el camino { *Encounter*, *diagnosis*, *Diagnosis* }. Marcó *All-data filter* y *match all conditions*.

No requirió ayuda del moderador.

Errores: (a) Aplicó la operación incorrecta, debía ser *Simple filter*.

- **Actividad 3:** El participante usó 4 minutos y 25 segundo para formular la consulta.

Marcó clase de interés *Diagnosis*. Puso filtro en *description* (de *diagnosis*) (=OSTEOMYELITIS) y *date* (de *encounter*) ($>=2000$). Marcó *description* (de *procedure*) y todos los atributos de *patient* (aclaró que para ella todos identifican al paciente). Eligió *All-Data filter* y *match all conditions*

El moderador mostró que se pueden mover las ventanas de los diálogos (Clase C).

La formulación fue correcta.

- **Actividad 4:** El participante formuló la consulta en 3 minutos y 13 segundos. Marcó como clase de interés *Patient*. Marca *race*, *gender* y *birthdate*. Agregó un filtro en *image* (=ACUTE CHOLECYSTITIS) y en *description* (de *procedure*) (=COLECTOMY). Marcó *All-data filter* y en el diálogo de desambiguación *all-condition* para *encounter*.

No requirió ayuda del moderador.

Errores: (a) Ubicó el filtro de anotación en *image*. (b) Faltó marcar *id* de *patient*. (c) Aplicó la operación incorrecta, debía ser *Simple filter*. (d) Debió elegir *Any-condition* para *encounter*.

Comentarios: el participante manifestó que tuvo dificultad para elegir el operador y la condición de *encounter*.

Segundo Participante

- **Actividad 1:** El participante formuló la consulta en 2 minutos y 52 segundos. Marcó *description* (de *Diagnosis*) como clase de interés. Agregó filtro en *id* (de *patient*) (=723628). Marcó el camino { *Encounter*, *familyHistory*, β_3 , *disease*, *Diagnosis* }. Seleccionó *relationship*. Seleccionó *Simple filter*. El moderador aclaró qué se requería en la consulta (Clase C). La consulta se formuló correctamente.
- **Actividad 2:** El participante usó 2 minutos y 44 segundo para formular la consulta. Marcó *description* (de *Diagnosis*) como clase de interés. Agregó filtro sobre *id* (de *patient*) (= 723628), *specialty* (de *physician*) (=ENDOCRINOLOGY). Marcó *date* (de *encounter*), *description* (de *medication*), y *dose*. Eligió el camino { *Encounter*, *diagnosis*, *Diagnosis* }. Marcó *Simple filter* y en el diálogo de clarificación *match all conditions* para *encounter*. El moderador aclaró qué se requería en la consulta (Clase C). La formulación fue correcta.
- **Actividad 3:** El participante formuló la consulta en 3 minutos y 35 segundos. Marcó clase de interés *Patient*. Puso filtro en *description* (de *diagnosis*) (=OSTEOMYELITIS) y *date* (de *encounter*) ($>=2000$). Marcó *description* (de *procedure*) y *id* (de *patient*). Eligió *All-Data filter* y *match all conditions*. El moderador aclaró qué se requería en la consulta (Clase C). La formulación fue correcta.
- **Actividad 4:** El participante usó 1 minutos y 58 segundo para formular la consulta. Marcó como clase de interés *Patient*. Marcó *race*, *gender* y *birthdate*. Agregó un filtro en *description* (de *Diagnostic Image*) (=ACUTE CHOLECYSTITIS) y en *description* (de *procedure*) (=COLECTOMY). Marcó *All-data filter* y en el diálogo

de desambiguación *Any-condition* para *encounter* y *All-conditions* para *Patient*. El moderador aclaró qué se requería en la consulta (Clase C).
Errores: (a) Ubicó filtro de anotación en *description*. (b) Aplicó la operación incorrecta, debía ser *Simple filter*.

Tercer Participante

- **Actividad 1:** El participante usó 1 minuto y 56 segundos para formular la consulta.
Marcó *description* (de *Diagnosis*) como clase de interés. Marcó *description* de *Diagnosis* Agregó filtro en *id* (de *patient*) (=723628). Marcó el camino { *Encounter*, *familyHistory*, β_3 , *disease*, *Diagnosis* }. Seleccionó *Filter*. No requirió ayuda del moderador.
La formulación fue correcta.
- **Actividad 2:** El participante formuló la consulta en 4 minutos y 8 segundos.
Marcó *Diagnosis* como clase de interés. Agregó filtro sobre *id* (de *patient*) (= 723628), *specialty* (de *physician*) (=ENDOCRINOLOGY). Marcó *date* (de *encounter*), *description* y *code* (de *medication*), y *dose*. Eligió el camino { *Encounter*, *diagnosis*, *Diagnosis* }. Marcó *Filter* y en el diálogo de clarificación *match all conditions* para *encounter*.
No requirió ayuda del moderador.
La formulación fue correcta.
- **Actividad 3:** El participante formuló la consulta en 3 minutos y 32 segundos.
Marcó clase de interés *Patient*. Pone filtro en *description* (de *diagnosis*) (=osteomyelitis) y *date* (de *encounter*) ($>=01-01-2000$). Marcó *description* (de *procedure*) y *id* (de *patient*). Eligió *Filter+Additional Data* y *match all conditions*. No requirió ayuda del moderador.
La formulación fue correcta.
- **Actividad 4:** El participante formuló la consulta en 3 minutos y 48 segundos.
Marcó como clase de interés *Patient*. Marca *id*, *race*, *gender* y *birthdate*. Agregó un filtro en *annotation* (=gallstones) y en *description* (de *procedure*) (=colectomy). Marcó *Filter+Additional Data* y en el diálogo de desambiguación *Any-condition* para *encounter* y *Any-condition* para *Patient*.
No requirió ayuda del moderador.
Errores: (a) Aplicó la operación incorrecta, debía ser *Filter*. (b) Debió seleccionar *All-conditions* para *Patient* en el diálogo de clarificación.

Cuarto Participante

- **Actividad 1:** El participante usó 3 minutos y 28 segundos para formular la consulta.
Marcó *Patient* como clase de interés. Agregó filtro en *id* (de *patient*) (=723628).

Marcó *code* y *description* de *Diagnosis* y *relationship*. Marcó el camino {*Encounter*, *familyHistory*, β_3 , *disease*, *Diagnosis*}. Seleccionó *Filter*.

El moderador explicó porqué en el modelo las enfermedades se representan con el nodo *Diagnosis* (Clase C).

La formulación fue correcta.

- **Actividad 2:** El participante formuló la consulta en 2 minutos y 51 segundos. Marcó *Patient* como clase de interés. Agregó filtro sobre *id* (de *patient*) (= 723628), *specialty* (de *physician*) (=ENDOCRINOLOGY). Marcó *date* (de *encounter*), *description* (de *medication*), y *dose*. Eligió el camino {*Encounter*, *diagnosis*, *Diagnosis*}. Marcó *Filter* y en el diálogo de clarificación *match any condition* para *encounter*.

No requirió ayuda del moderador.

Errores: (a) Debió seleccionar *All-conditions* para *Patient* en el diálogo de clarificación.

- **Actividad 3:** El participante usó 4 minutos 45 segundos para formular la consulta. Marcó clase de interés *Diagnosis*. Puso filtro en *description* (de *diagnosis*) (=OSTEOMYELITIS) y *date* (de *encounter*) (>=01-01-2000). Marcó *description* (de *procedure*) y *id* (de *patient*). Eligió *Filter+Additional Data* y en el diálogo de clarificación *match all conditions* para *encounter*.

El moderador aclaró la diferencia entre el código y el nombre del diagnóstico (Clase C) y aclaró lo que se requiere en la consulta porque la redacción de la actividad resulta confusa debido a la observación sobre las fechas (Clase C).

La consulta se formuló correctamente.

- **Actividad 4:** El participante formuló la consulta en 1 minuto y 21 segundos. Marcó como clase de interés *Patient*. Marca *race*, *gender* y *birthdate*. Agregó un filtro en *annotation* (=GALLSTONES) y en *description* (de *procedure*) (=COLECTOMY). Marcó *Filter* y en el diálogo de desambiguación *All-conditions* para *encounter*.

No requirió ayuda del moderador.

Errores: (a) Debió seleccionar *Any-condition* para *encounter*.

Anexo I. Consultas de la prueba de implementación

En este anexo se presentan las consultas usadas en la prueba de implementación, expresadas en SPARQL y con los operadores de nivel lógico de GraphTQL. Al final del anexo se muestran las consultas GraphTQL expresadas en XML.

Consulta 1

Se requieren las enfermedades (incluyendo su descripción) registradas en la historia familiar del paciente con identificación 723628; incluir la relación familiar si está disponible

Expresión SPARQL:

```
CONSTRUCT {  
    ?enc <http://example.com/property/of> ?pat .  
    ?pat <http://example.com/property/id> ?id .  
    ?enc <http://example.com/property/familyHistory> ?fh .  
    ?fh <http://example.com/property/disease> ?diag .  
    ?diag <http://example.com/property/description> ?desc .  
    ?fh <http://example.com/property/relationship> ?rel  
}  
WHERE {  
    ?enc <http://example.com/property/of> ?pat.  
    ?pat <http://example.com/property/id> ?id .  
    ?enc <http://example.com/property/familyHistory> ?fh .  
    ?fh <http://example.com/property/disease> ?diag .  
    OPTIONAL {?diag <http://example.com/property/description> ?desc}  
    OPTIONAL {?fh <http://example.com/property/relationship> ?rel}  
    FILTER (?id = 20000002)  
}
```

Expresión GraphTQL:

```
LogicLFilter(<S, I>, Diagnosis,  
(id_{Diagnosis,disease,\beta_3,familyHistory,Encounter,of, Patient,id,id} = 20000002))  
→ <S', I'>  
SelUnion(<S, I>, <S', I'>, Diagnosis, {{description}}) → <S'', I''>  
SelUnion(<S, I>, <S'', I''>, \beta_3, {{relationship}}) → <S''', I'''>
```

Consulta 2
<p>Se requiere encontrar los diagnósticos (incluyendo su descripción) dados al paciente con identificación 723628 por médicos con especialidad en “ENDOCRINOLOGY” y, si está disponible, las fechas de las consultas y las medicinas que le recetaron en ese momento (incluyendo descripción y dosis)</p> <p>Expresión SPARQL:</p> <pre>CONSTRUCT { ?enc <http://example.com/property/of> ?pat . ?pat <http://example.com/property/id> ?id . ?enc <http://example.com/property/attendedBy> ?phy . ?phy <http://example.com/property/specialty> ?spe . ?enc <http://example.com/property/diagnosis> ?diag . ?diag <http://example.com/property/description> ?dgDesc . ?enc <http://example.com/property/encounterDate> ?enD . ?enc <http://example.com/property/medicationOrder> ?mo . ?mo <http://example.com/property/medication> ?me . ?me <http://example.com/property/description> ?meDesc . ?enc <http://example.com/property/medicationOrder> ?mo1 . ?mo1 <http://example.com/property/dose> ?dose . } WHERE { ?enc <http://example.com/property/of> ?pat. ?pat <http://example.com/property/id> ?id . ?enc <http://example.com/property/attendedBy> ?phy . ?phy <http://example.com/property/specialty> ?spe . ?enc <http://example.com/property/diagnosis> ?diag . OPTIONAL {?diag <http://example.com/property/description> ?dgDesc} OPTIONAL {?enc <http://example.com/property/encounterDate> ?enD} OPTIONAL { ?enc <http://example.com/property/medicationOrder> ?mo . ?mo <http://example.com/property/medication> ?me . ?me <http://example.com/property/description> ?meDesc} OPTIONAL { ?enc <http://example.com/property/medicationOrder> ?mo1 . ?mo1 <http://example.com/property/dose> ?dose} FILTER (?id = 20000002) FILTER (str(?spe) = "ENDOCRINOLOGY")}</pre>

```
CONSTRUCT {
?enc <http://example.com/property/of> ?pat .
?pat <http://example.com/property/id> ?id .
?enc <http://example.com/property/attendedBy> ?phy .
?phy <http://example.com/property/specialty> ?spe .
?enc <http://example.com/property/diagnosis> ?diag .
?diag <http://example.com/property/description> ?dgDesc .
?enc <http://example.com/property/encounterDate> ?enD .
?enc <http://example.com/property/medicationOrder> ?mo .
?mo <http://example.com/property/medication> ?me .
?me <http://example.com/property/description> ?meDesc .
?enc <http://example.com/property/medicationOrder> ?mo1 .
?mo1 <http://example.com/property/dose> ?dose .
}
WHERE {
?enc <http://example.com/property/of> ?pat.
?pat <http://example.com/property/id> ?id .
?enc <http://example.com/property/attendedBy> ?phy .
?phy <http://example.com/property/specialty> ?spe .
?enc <http://example.com/property/diagnosis> ?diag .
OPTIONAL {?diag <http://example.com/property/description> ?dgDesc}
OPTIONAL {?enc <http://example.com/property/encounterDate> ?enD}
OPTIONAL {
?enc <http://example.com/property/medicationOrder> ?mo .
?mo <http://example.com/property/medication> ?me .
?me <http://example.com/property/description> ?meDesc}
OPTIONAL {
?enc <http://example.com/property/medicationOrder> ?mo1 .
?mo1 <http://example.com/property/dose> ?dose}
FILTER (?id = 20000002) FILTER (str(?spe) = "ENDOCRINOLOGY")}
```

Expresión GraphTQL:

```
LogicLFilter(<S, I>, Encounter,  
    (id_{Encounter,of, Patient,id,id} = 20000002) AND  
    (specialty_{Encounter,attendedBy,Physician,specialty,specialty} =  
        "ENDOCRINOLOGY")) → <SSF, ISF>  
LogicLFilter(<S, I>, Diagnosis,  
    (Encounter_{Encounter,diagnosis,Diagnosis} inSubgraph <SSF, ISF>))  
    → <S', I'>  
SelUnion(<S, I>, <S', I'>, Diagnosis, {{description}}) → <S'', I''>  
SelUnion(<S, I>, <S'', I''>, Encounter,  
    {{Encounter, encounterDate, date}, {Encounter, medicationOrder,  
        β4, medication, Medication, description, description},  
        {Encounter, medicationOrder, β4, dose, dose}}) → <S''', I'''>
```

Consulta 3

De los pacientes que han tenido un diagnóstico de “OSTEOMYELITIS” desde el año 2000, se requieren todos los datos de: identificación del paciente, diagnósticos recibidos, y procedimientos que se le han realizado (incluir la descripción de los diagnósticos y procedimientos). Note que estos datos incluyen los otros diagnósticos, además de OSTEOMYELITIS y eventos que se dieron en fechas diferentes a la del diagnóstico de OSTEOMYELITIS.

Expresión SPARQL:

```
CONSTRUCT {  
    ?enc <http://example.com/property/of> ?pat .  
    ?enc <http://example.com/property/encounterDate> ?enD .  
    ?enc <http://example.com/property/diagnosis> ?diag .  
    ?diag <http://example.com/property/description> ?dgDesc .  
    ?pat <http://example.com/property/id> ?id .  
    ?enc4 <http://example.com/property/of> ?pat .  
    ?enc4 <http://example.com/property/encounterDate> ?encD2 .  
    ?enc2 <http://example.com/property/of> ?pat .  
    ?enc2 <http://example.com/property/diagnosis> ?diag2 .  
    ?diag2 <http://example.com/property/description> ?dgDesc2 .  
    ?enc3 <http://example.com/property/of> ?pat .  
    ?enc3 <http://example.com/property/procedure> ?pro .  
    ?pro <http://example.com/property/description> ?prDesc .  
}  
WHERE {  
    ?enc <http://example.com/property/of> ?pat .  
    ?enc <http://example.com/property/encounterDate> ?enD .  
    ?enc <http://example.com/property/diagnosis> ?diag .  
    ?diag <http://example.com/property/description> ?dgDesc .  
    OPTIONAL { ?pat <http://example.com/property/id> ?id }  
    OPTIONAL { ?enc4 <http://example.com/property/of> ?pat .  
        ?enc4 <http://example.com/property/encounterDate> ?encD2 }  
    OPTIONAL { ?enc2 <http://example.com/property/of> ?pat .  
        ?enc2 <http://example.com/property/diagnosis> ?diag2 .  
        ?diag2 <http://example.com/property/description> ?dgDesc2 }  
    OPTIONAL { ?enc3 <http://example.com/property/of> ?pat .  
        ?enc3 <http://example.com/property/procedure> ?pro .  
        ?pro <http://example.com/property/description> ?prDesc }  
    FILTER (str(?enD) >= ‘2000-01-01’)  
    FILTER (str(?dgDesc) = “OSTEOMYELITIS”)  
}
```

Expresión GraphTQL:

```
LogicLFilter(<S, I>, Encounter,  
    (date_{Encounter, encounterDate, date} = "2000-01-01") AND  
    (description_{Encounter, diagnosis, Diagnosis, description, description} =  
        "OSTEOMYELITIS")) → <SSF, ISF>  
LogicLFilter(<S, I>, Patient,  
    (Encounter_{Encounter, of, Patient} inSubgraph <SSF, ISF>))  
    → <S', I'>  
SelUnion(<S, I>, <S', I'>, Patient,  
    {{Patient, of, Encounter, encounterDate, date},  
     {Patient, of, Encounter, procedure, Procedure, description},  
     {Patient, of, Encounter, diagnosis, Diagnosis, description,  
      description}}) → <S'', I''>
```

Consulta 4

Se requiere encontrar los pacientes que han tenido un estudio de imágenes diagnósticas con anotación “GALLSTONES” y se les ha realizado el procedimiento “COLECTOMY”. De estos pacientes se requiere la fecha de nacimiento, género y raza, si están disponibles. Estos resultados pudieron realizarse en encuentros diferentes, pero el paciente cumple con ambas condiciones

Expresión SPARQL:

```
CONSTRUCT {  
    ?enc1 <http://example.com/property/of> ?pat.  
    ?enc1 <http://example.com/property/diagnosticImage> ?dImg .  
    ?dImg <http://example.com/property/annotation> ?ann .  
    ?enc2 <http://example.com/property/of> ?pat .  
    ?enc2 <http://example.com/property/procedure> ?pro .  
    ?pro <http://example.com/property/description> ?prDesc .  
    ?pat <http://example.com/property/demographics> ?dem1 .  
    ?dem1 <http://example.com/property/race> ?race .  
    ?pat <http://example.com/property/demographics> ?dem2 .  
    ?dem2 <http://example.com/property/gender> ?gender .  
    ?pat <http://example.com/property/demographics> ?dem3 .  
    ?dem3 <http://example.com/property/dateOfBirth> ?db .  
}  
WHERE {  
    ?enc1 <http://example.com/property/of> ?pat .  
    ?enc1 <http://example.com/property/diagnosticImage> ?dImg .  
    ?dImg <http://example.com/property/annotation> ?ann .  
    ?enc2 <http://example.com/property/of> ?pat .  
    ?enc2 <http://example.com/property/procedure> ?pro .  
    ?pro <http://example.com/property/description> ?prDesc  
OPTIONAL {  
    ?pat <http://example.com/property/demographics> ?dem1 .  
    ?dem1 <http://example.com/property/race> ?race }  
OPTIONAL {  
    ?pat <http://example.com/property/demographics> ?dem2 .  
    ?dem2 <http://example.com/property/gender> ?gender }  
OPTIONAL {  
    ?pat <http://example.com/property/demographics> ?dem3 .  
    ?dem3 <http://example.com/property/dateOfBirth> ?db }  
FILTER (str(?ann) = “GALLSTONES”)  
FILTER (str(?prDesc)=“COLECTOMY”)  
}
```

Expresión GraphTQL:

```
LogicLFilter(<S, I>, Patient,
  (annotation_{Patient,of,Encounter,diagnosticImage,DiagnosticImageStudy,
    annotation,annotation} = "GALLSTONES") AND
  (description_{Patient,of,Encounter,procedure,Procedure,description,description} =
    "COLECTOMY"))
→ <S', I'>
SelUnion(<S, I>, <S', I'>, Patient,
  {{Patient, demographics, β₁, race, race},
   {Patient, demographics, β₁, gender, gender},
   {Patient, demographics, β₁, dateOfBirth, birthDate}}))
→ <S'', I''>
```

Consulta 5

Se requiere encontrar a los pacientes que han tenido una imagen diagnóstica “CAROTID PULSE TRACING WITH ECG LEAD” y a los pacientes a los que se les ha realizado el procedimiento “INSERTION OF PALATAL IMPLANT”. Recuperar además la identificación de estos pacientes, si está disponible.

Expresión SPARQL:

```
CONSTRUCT {  
    ?enc <http://example.com/property/of> ?pat .  
    ?enc <http://example.com/property/diagnosticImage> ?dImg .  
    ?dImg <http://example.com/property/image> ?img .  
    ?img <http://example.com/property/description> ?descImg .  
    ?enc <http://example.com/property/procedure> ?pro .  
    ?pro <http://example.com/property/description> ?prDesc .  
    ?pat <http://example.com/property/id> ?id .  
}  
WHERE {  
    { ?enc <http://example.com/property/of> ?pat .  
        ?enc <http://example.com/property/diagnosticImage> ?dImg .  
        ?dImg <http://example.com/property/image> ?img .  
        ?img <http://example.com/property/description> ?descImg .  
        FILTER (str(?descImg) = “CAROTID PULSE TRACING  
                WITH ECG LEAD” ) }  
    UNION  
    { ?enc <http://example.com/property/of> ?pat .  
        ?enc <http://example.com/property/procedure> ?pro .  
        ?pro <http://example.com/property/description> ?prDesc .  
        FILTER (str(?prDesc) = “INSERTION OF PALATAL IMPLANT” ) }  
    OPTIONAL { ?pat <http://example.com/property/id> ?id }  
}
```

Expresión GraphTQL:

```
LogicLFilter(<S, I>, Patient,  
  (description_{Patient,of,Encounter,diagnosticImage,DiagnosticImageStudy,  
    image,diagnosticImage,description,description}  
   = “CAROTID PULSE TRACING WITH ECG LEAD”) OR  
  (description_{Patient,of,Encounter,procedure,Procedure,description,description} =  
   “INSERTION OF PALATAL IMPLANT”)) → <S’, I’>  
SelUnion(<S, I>, <S’, I’>, Patient, {{Patient, id, id}}) → <S”, I”>
```

Consultas de GraphTQL expresadas en XML

Consulta 1

```
<?xml version="1.0" encoding="UTF-8"?>
<query>
    <operation>
        <operator>FILTER</operator>
        <result>
            <schema>S-1</schema>
            <instance>I-1</instance>
        </result>
        <parameters>
            <schema>S-0</schema>
            <instance>I-0</instance>
            <interestClass>Diagnosis</interestClass>
            <expression>
                <classes>
                    <class>paId</class>
                </classes>
                <paths>
                    <path>
                        <edge>fhDi</edge>
                        <edge>enFH</edge>
                        <edge>enOf</edge>
                        <edge>paID</edge>
                    </path>
                </paths>
                <ops>
                    <op>=</op>
                </ops>
                <literals>
                    <literal>20000002</literal>
                </literals>
            </expression>
        </parameters>
    </operation>
    <operation>
        <operator>SELUNION</operator>
        <parameters>
            <schema>S-0</schema>
            <instance>I-0</instance>
            <schema>S-1</schema>
            <instance>I-1</instance>
            <interestClass>Diagnosis</interestClass>
            <paths>
                <path>
                    <edge>dgDe</edge>
                </path>
            </paths>
        </parameters>
    </operation>
```

```
<operation>
  <operator>SELUNION</operator>
  <parameters>
    <schema>S-0</schema>
    <instance>I-0</instance>
    <schema>S-1</schema>
    <instance>I-1</instance>
    <interestClass>famHistory</interestClass>
    <paths>
      <path>
        <edge>fhRe</edge>
      </path>
    </paths>
  </parameters>
</operation>
</query>
```

Consulta 2

```
<?xml version="1.0" encoding="UTF-8"?>
<query>
    <operation>
        <operator>SUBFILTER</operator>
        <result>
            <subFId>SF-0</subFId>
        </result>
        <parameters>
            <schema>S-0</schema>
            <instance>I-0</instance>
            <filterResult>
                <schema>S-1</schema>
                <instance>I-1</instance>
            </filterResult>
            <interestClass>Encounter</interestClass>
            <expression>
                <booleanOps>
                    <bop>AND</bop>
                </booleanOps>
                <classes>
                    <class>paID</class>
                    <class>phSpecialty</class>
                </classes>
                <paths>
                    <path>
                        <edge>en0f</edge>
                        <edge>paID</edge>
                    </path>
                    <path>
                        <edge>enAB</edge>
                        <edge>phSp</edge>
                    </path>
                </paths>
                <ops>
                    <op>=</op>
                    <op>=</op>
                </ops>
                <literals>
                    <literal>20000002</literal>
                    <literal>ENDOCRINOLOGY</literal>
                </literals>
            </expression>
        </parameters>
    </operation>
```

```
<operation>
  <operator>FILTER</operator>
  <result>
    <schema>S-1</schema>
    <instance>I-1</instance>
  </result>
  <parameters>
    <schema>S-0</schema>
    <instance>I-0</instance>
    <interestClass>Diagnosis</interestClass>
    <expression>
      <classes>
        <class>Encounter</class>
      </classes>
      <paths>
        <path>
          <edge>enDg</edge>
        </path>
      </paths>
      <ops>
        <op>inSubG</op>
      </ops>
      <literals>
        <literal>SF-0</literal>
      </literals>
    </expression>
  </parameters>
</operation>
<operation>
  <operator>SELUNION</operator>
  <parameters>
    <schema>S-0</schema>
    <instance>I-0</instance>
    <schema>S-1</schema>
    <instance>I-1</instance>
    <interestClass>Diagnosis</interestClass>
    <paths>
      <path>
        <edge>dgDe</edge>
      </path>
    </paths>
  </parameters>
</operation>
```

```
<operation>
  <operator>SELUNION</operator>
  <parameters>
    <schema>S-0</schema>
    <instance>I-0</instance>
    <schema>S-1</schema>
    <instance>I-1</instance>
    <interestClass>Encounter</interestClass>
    <paths>
      <path>
        <edge>enDa</edge>
      </path>
      <path>
        <edge>enM0</edge>
        <edge>moMe</edge>
        <edge>meDe</edge>
      </path>
      <path>
        <edge>enM0</edge>
        <edge>moDo</edge>
      </path>
    </paths>
  </parameters>
</operation>
</query>
```

Consulta 3

```
<?xml version="1.0" encoding="UTF-8"?>
<query>
    <operation>
        <operator>SUBFILTER</operator>
        <result>
            <subFId>SF-0</subFId>
        </result>
        <parameters>
            <schema>S-0</schema>
            <instance>I-0</instance>
            <filterResult>
                <schema>S-1</schema>
                <instance>I-1</instance>
            </filterResult>
            <interestClass>Encounter</interestClass>
        <expression>
            <booleanOps>
                <bop>AND</bop>
            </booleanOps>
            <classes>
                <class>enDate</class>
                <class>dgDescription</class>
            </classes>
            <paths>
                <path>
                    <edge>enDa</edge>
                </path>
                <path>
                    <edge>enDg</edge>
                    <edge>dgDe</edge>
                </path>
            </paths>
            <ops>
                <op>>=</op>
                <op>=</op>
            </ops>
            <literals>
                <literal>2000-01-01</literal>
                <literal>OSTEOMYELITIS</literal>
            </literals>
        </expression>
    </parameters>
</operation>
```

```

<operation>
    <operator>FILTER</operator>
    <result>
        <schema>S-1</schema>
        <instance>I-1</instance>
    </result>
    <parameters>
        <schema>S-0</schema>
        <instance>I-0</instance>
        <interestClass>Patient</interestClass>
        <expression>
            <classes>
                <class>Encounter</class>
            </classes>
            <paths>
                <path>
                    <edge>enOf</edge>
                </path>
            </paths>
            <ops>
                <op>inSubG</op>
            </ops>
            <literals>
                <literal>SF-0</literal>
            </literals>
        </expression>
    </parameters>
</operation>
<operation>
    <operator>SELUNION</operator>
    <parameters>
        <schema>S-0</schema>
        <instance>I-0</instance>
        <schema>S-1</schema>
        <instance>I-1</instance>
        <interestClass>Patient</interestClass>
        <paths>
            <path>
                <edge>paID</edge>
            </path>
            <path>
                <edge>enOf</edge>
                <edge>enDa</edge>
            </path>
            <path>
                <edge>enOf</edge>
                <edge>enPr</edge>
                <edge>prDe</edge>
            </path>
            <path>
                <edge>enOf</edge>
                <edge>enDg</edge>
                <edge>dgDe</edge>
            </path>
        </paths>
    </parameters>
</operation>
</query>

```

Consulta 4

```
<?xml version="1.0" encoding="UTF-8"?>
<query>
    <operation>
        <operator>FILTER</operator>
        <result>
            <schema>S-1</schema>
            <instance>I-1</instance>
        </result>
        <parameters>
            <schema>S-0</schema>
            <instance>I-0</instance>
            <interestClass>Patient</interestClass>
            <expression>
                <booleanOps>
                    <bop>AND</bop>
                </booleanOps>
                <classes>
                    <class>diAnnotation</class>
                    <class>prDescription</class>
                </classes>
                <paths>
                    <path>
                        <edge>enOf</edge>
                        <edge>enDI</edge>
                        <edge>imAn</edge>
                    </path>
                    <path>
                        <edge>enOf</edge>
                        <edge>enPr</edge>
                        <edge>prDe</edge>
                    </path>
                </paths>
                <ops>
                    <op>==</op>
                    <op>=</op>
                </ops>
                <literals>
                    <literal>GALLSTONES</literal>
                    <literal>MYRINGOTOMY WITH INSERTION OF TUBE</literal>
                </literals>
            </expression>
        </parameters>
    </operation>
```

```
<operation>
  <operator>SELUNION</operator>
  <parameters>
    <schema>S-0</schema>
    <instance>I-0</instance>
    <schema>S-1</schema>
    <instance>I-1</instance>
    <interestClass>Patient</interestClass>
    <paths>
      <path>
        <edge>paDe</edge>
        <edge>paRa</edge>
      </path>
      <path>
        <edge>paDe</edge>
        <edge>paGe</edge>
      </path>
      <path>
        <edge>paDe</edge>
        <edge>paDB</edge>
      </path>
    </paths>
  </parameters>
</operation>
</query>
```

Consulta 5

```
<?xml version="1.0" encoding="UTF-8"?>
<query>
    <operation>
        <operator>FILTER</operator>
        <result>
            <schema>S-1</schema>
            <instance>I-1</instance>
        </result>
        <parameters>
            <schema>S-0</schema>
            <instance>I-0</instance>
            <interestClass>Patient</interestClass>
            <expression>
                <booleanOps>
                    <bop>OR</bop>
                </booleanOps>
                <classes>
                    <class>diDescription</class>
                    <class>prDescription</class>
                </classes>
                <paths>
                    <path>
                        <edge>enOf</edge>
                        <edge>enDI</edge>
                        <edge>diIm</edge>
                        <edge>diDe</edge>
                    </path>
                    <path>
                        <edge>enOf</edge>
                        <edge>enPr</edge>
                        <edge>prDe</edge>
                    </path>
                </paths>
                <ops>
                    <op>==</op>
                    <op>==</op>
                </ops>
                <literals>
                    <literal>CAROTID PULSE TRACING WITH ECG LEAD</literal>
                    <literal>INSERTION OF PALATAL IMPLANT</literal>
                </literals>
            </expression>
        </parameters>
    </operation>
    <operation>
        <operator>SELUNION</operator>
        <parameters>
            <schema>S-0</schema>
            <instance>I-0</instance>
            <schema>S-1</schema>
            <instance>I-1</instance>
            <interestClass>Patient</interestClass>
            <paths>
                <path>
                    <edge>paID</edge>
                </path>
            </paths>
        </parameters>
    </operation>
</query>
```