Universidad de Alcalá
COMISIÓN DE ESTUDIOS OFICIALES
DE POSGRADO Y DOCTORADO

ACTA DE EVALUACIÓN DE LA TESIS DOCTORAL

Año académico 2016/17

DOCTORANDO: **LÓPEZ MARTÍNEZ, ANTONIO**
D.N.I./PASAPORTE: ******1344A**

PROGRAMA DE DOCTORADO: **D338 DOCTORADO EN INGENIERÍA DE LA INFORMACIÓN Y DEL CONOCIMIENTO**
DEPARTAMENTO DE: **Ciencias de la Computación**
TITULACIÓN DE DOCTOR EN: **DOCTOR/A POR LA UNIVERSIDAD DE ALCALÁ**

En el día de hoy 04/09/17, reunido el tribunal de evaluación nombrado por la Comisión de Estudios Oficiales de Posgrado y Doctorado de la Universidad y constituido por los miembros que suscriben la presente Acta, el aspirante defendió su Tesis Doctoral, elaborada bajo la dirección de ELADIO DOMÍNGUEZ MURILLO // JOSÉ A. GUTIÉRREZ DE MESA.

Sobre el siguiente tema: *LENGUAJE DE CONSULTAS PARA LA GESTIÓN DE ACONTECIMIENTOS*

Finalizada la defensa y discusión de la tesis, el tribunal acordó otorgar la CALIFICACIÓN GLOBAL[3] de (**no apto, aprobado, notable y sobresaliente**): SOBRESALIENTE

Alcalá de Henares, 4 de Septiembre de 2017

EL PRESIDENTE | EL SECRETARIO | EL VOCAL

Fdo.: Ángel Rosúa Francés | Fdo.: Roberto Barchino | Fdo.: José Carlos Cívica

Con fecha 14 de septiembre de 2017 la Comisión Delegada de la Comisión de Estudios Oficiales de Posgrado, a la vista de los votos emitidos de manera anónima por el tribunal que ha juzgado la tesis, resuelve:

- [X] Conceder la Mención de "Cum Laude"
- [ ] No conceder la Mención de "Cum Laude"

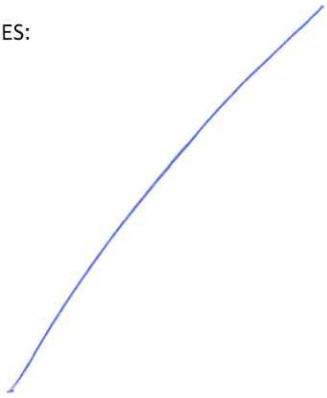
La Secretaria de la Comisión Delegada

FIRMA DEL ALUMNO

Fdo.: Antonio López

[3] La calificación podrá ser **"no apto" "aprobado" "notable" y "sobresaliente"**. El tribunal podrá otorgar la mención de "cum laude" si la calificación global es de sobresaliente y se emite en tal sentido el voto secreto positivo por unanimidad.

INCIDENCIAS / OBSERVACIONES:



Universidad de Alcalá
COMISIÓN DE ESTUDIOS OFICIALES
DE POSGRADO Y DOCTORADO

En aplicación del art. 14.7 del RD. 99/2011 y el art. 14 del Reglamento de Elaboración, Autorización y Defensa de la Tesis Doctoral, la Comisión Delegada de la Comisión de Estudios Oficiales de Posgrado y Doctorado, en sesión pública de fecha 14 de septiembre, procedió al escrutinio de los votos emitidos por los miembros del tribunal de la tesis defendida por *LÓPEZ MARTÍNEZ, ANTONIO*, el día 4 de septiembre de 2017, titulada *LENGUAJE DE CONSULTAS PARA LA GESTIÓN DE ACONTECIMIENTOS*, para determinar, si a la misma, se le concede la mención "cum laude", arrojando como resultado el voto favorable de todos los miembros del tribunal.

Por lo tanto, la Comisión de Estudios Oficiales de Posgrado **resuelve otorgar** a dicha tesis la

MENCIÓN "CUM LAUDE"

Alcalá de Henares, 21 de septiembre de 2017
EL PRESIDENTE DE LA COMISIÓN DE ESTUDIOS
OFICIALES DE POSGRADO Y DOCTORADO

Firmado digitalmente por
VELASCO PEREZ JUAN
RAMON - DNI 03087239H
Fecha: 2017.09.22
11:20:40 +02'00'

Juan Ramón Velasco Pérez

Copia por e-mail a:
Doctorando: LÓPEZ MARTÍNEZ, ANTONIO
Secretario del Tribunal: ROBERTO BARCHINO PLATA.
Directores de Tesis: ELADIO DOMÍNGUEZ MURILLO // JOSÉ A. GUTIÉRREZ DE MESA

Universidad de Alcalá

ESCUELA DE DOCTORADO
Servicio de Estudios Oficiales de Posgrado

DILIGENCIA DE DEPÓSITO DE TESIS.

Comprobado que el expediente académico de D./Dª ANTONIO LÓPEZ MARTÍNEZ reúne los requisitos exigidos para la presentación de la Tesis, de acuerdo a la normativa vigente, y habiendo presentado la misma en formato: ☐ soporte electrónico ☒ impreso en papel, para el depósito de la misma, en el Servicio de Estudios Oficiales de Posgrado, con el nº de páginas: 201 se procede, con fecha de hoy a registrar el depósito de la tesis.

Alcalá de Henares a 26 de abril de 2017

UNIVERSIDAD DE ALCALÁ SERVICIO DE ESTUDIOS OFICIALES DE PÓSGRADO

Emilia Rivero Antúnez

Fdo. El Funcionario

Universidad de Alcalá

DPTO. DE CIENCIAS DE LA COMPUTACIÓN
Escuela Politécnica Superior
Campus Universitario. Edificio Politécnico
28871 Alcalá de Henares (Madrid)
Teléfonos: 91 885 6645 Fax: 91 885 6646
e-mail: dpto.computacion@uah.es

Dr. D. Eladio Domínguez Murillo, Catedrático de Universidad del Área de Ciencia de la Computación e Inteligencia Artificial del Departamento de Informática e Ingeniería de Sistemas de la Universidad de Zaragoza,

Dr. D. José Antonio Gutiérrez De Mesa, Profesor Titular de Universidad del Área de Ciencia de la Computación e Inteligencia Artificial del Departamento de Ciencias de la Computación de la Universidad de Alcalá,

CERTIFICAN: Que una vez concluido el trabajo de Tesis Doctoral titulado "Lenguaje de Consultas para la Gestión de Acontecimientos" realizado por D. Antonio López Martínez, dicho trabajo tiene suficientes méritos teóricos, que se han contrastado adecuadamente mediante validaciones experimentales y que son altamente novedosos. Por todo ello se considera que procede su defensa pública.

Y para que conste, firman la presente en Zaragoza, a 5 de abril de 2017

Los Directores de la Tesis

Fdo.: Dr. Eladio Domínguez Murillo

Fdo.: Dr. José Antonio Gutiérrez de Meza

Universidad de Alcalá

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
Escuela Politécnica Superior
Campus Universitario. Edificio Politécnico
28871 Alcalá de Henares (Madrid)
Teléfonos: +34 91 885 66 51
Fax: +34 91 885 6646

Dra. Dña. Teresa I. Díez Folledo, Profesora Titular de Universidad del Área de Lenguajes y Sistemas Informáticos, en calidad de Directora del Departamento de Ciencias de la Computación.

CERTIFICO: Que una vez concluido el trabajo de la Tesis Doctoral titulada **"Lenguaje de Consultas para la Gestión de Acontecimientos"** realizada por D. Antonio López Martínez y dirigida por los Drs. D. Eladio Dominguez Murillo y D. José Antonio Gutiérrez de Mesa, reúne los requisitos para su presentación y defensa pública.

Y para que así conste, firmo la presente en Alcalá de Henares, a 21 de Abril de 2017

La Directora del Departamento de Ciencias de la Computación

Dra. Dña. Teresa Díez Folledo

Universidad de Alcalá

Programa de Doctorado en Ingeniería de la Información y del Conocimiento

Lenguaje de Consultas para la Gestión de Acontecimientos

Tesis doctoral presentada por

Antonio López Martínez

2017

Programa de Doctorado en Ingeniería de la Información y del Conocimiento

Lenguaje de Consultas para la Gestión de Acontecimientos

Tesis doctoral presentada por

Antonio López Martínez

Directores:
Dr. Eladio Domínguez Murillo
Dr. José Antonio Gutiérrez De Mesa

Alcalá de Henares, 2017

A mis padres.

Agradecimientos

Cuando al acercarse el final de este camino echas la vista atrás y ves que tienes tanto que agradecer, te sientes afortunado por haber conocido a tantas personas a lo largo del trayecto que lo han dado todo por ayudarte y empujarte a seguir adelante, paso a paso.

A mis directores de tesis, Eladio Domínguez y José Antonio Gutiérrez, les agradezco mucho su inestimable apoyo a lo largo de estos años y sus esfuerzos para que esta tesis llegue a ser una realidad.

Mi especial agradecimiento es para Eladio Domínguez, cuya profesionalidad, capacidad de trabajo y calidad humana me motivarán siempre. Le estaré siempre agradecido por su apoyo, su incansable espíritu de lucha y su ayuda, que me han permitido llegar hasta aquí.

Gracias a Ángel Luis Rubio, a Beatriz Pérez, a José Carlos Ciria y a María Antonia Zapata por su trabajo y dedicación, que ha sido fundamental para esta tesis. Gracias Toñi por tu dedicación y por el rigor en tus correcciones.

Gracias a todos los integrantes del Grupo de Investigación Nóesis de la Universidad de Zaragoza por sus aportaciones e investigaciones previas, pues han sido de un valor incalculable. Gracias por vuestra amistad y apoyo constante a lo largo de estos años.

A las personas que me acompañaron en Infozara, muchísimas gracias por todos los años que hemos pasado juntos. Gracias a Alberto Allué, mi compañero de fatigas con su propia tesis, lo que hemos pasado ha sido excepcional, pero seguro que lo mejor todavía está por llegar.

Agradezco finalmente a mi familia. Gracias a mi abuela Gregoria por aguantarme, tiempo atrás, en los difíciles años de juventud. Gracias a mis padres, a los que tengo tanto que agradecer que no sabría por dónde empezar, os quiero.

RESUMEN

El objetivo de esta tesis es la definición de un lenguaje de consultas para el acceso a la información almacenada en un tipo de estructura de persistencia denominado *Base de Acontecimientos*.

Una *base de acontecimientos* es una estructura de persistencia cuya unidad mínima de información es el *Acontecimiento*. El concepto *Acontecimiento* se define como *"Una pieza de información concreta, identificable e indivisible que contiene aspectos organizados de acuerdo a tres dimensiones: guía, estructura y comportamiento"*.

En el contexto de esta tesis, *estructura* refiere a los objetos que permiten representar un determinado *Universo del Discurso* (*Universe of Discourse; UoD*), *guía* hace referencia a las posibles acciones que pueden afectar a los citados objetos y, por último, *comportamiento* refiere al efecto producido sobre un objeto cuando se le aplica una determinada acción. Así pues, una *base de acontecimientos* se define como *"Un tipo de estructura de información que registra acontecimientos que han tenido lugar a lo largo del tiempo"*.

A través de esta tesis se define (1) un *framework* o sistema de conceptos y reglas que permiten representar estructuralmente, en un *Universo del Discurso*, los elementos a utilizar en la gestión de acontecimientos, (2) un *metamodelo* que normaliza los conceptos aplicados en las *estrategias de diseño* que facilitan la construcción de *bases de acontecimientos*, y (3) un *lenguaje de consultas* para facilitar el acceso a la información almacenada en *bases de acontecimientos*.

El *lenguaje*, como parte de los objetivos pretendidos, posibilita no sólo acceder a la información almacenada en una *base de acontecimientos*, sino también a la información que describe la estructura de la misma que se encuentra almacenada en un componente del *framework* denominado *Diccionario*.

El desarrollo del lenguaje de consultas ha tomado como punto de partida los siguientes requisitos:

1 Independencia de la tecnología

2 Simplicidad

3 Adaptabilidad

4 Independencia del software

La *independencia tecnológica* se logra pudiendo consultar una base de acontecimientos sin importar el sistema de persistencia en el que se encuentre implantada.

La *simplicidad* se logra por dos vías:

1 Abstrayendo al usuario final de las estructuras del sistema de persistencia, es decir, evitando especificar la localización de la información a obtener al declarar una consulta.

2 Permitiendo declarar consultas sencillas que permitan lograr el mismo resultado que

consultas declaradas en otros lenguajes que podrían llegar a ser extremadamente complejas o largas dadas las características estructurales de las bases de acontecimientos.

La *adaptabilidad* se logra posibilitando la personalización de algunas de las características (*dimensiones*) del lenguaje en función de la estrategia de diseño seleccionada para la construcción de una base de acontecimientos.

Por último, la *independencia del software* se logra cuando ciertos cambios, como por ejemplo los debidos a la evolución de una aplicación informática a lo largo del tiempo, no implican modificaciones en las consultas previamente declaradas.

ÍNDICE

I. INTRODUCCIÓN Y ANTECEDENTES

Capítulo 1. Introducción y objetivos

1.1 Introducción

La *procedencia* (*provenance*; nombre que deriva del término en francés *provenir* [61]) de un objeto refiere al conjunto de acontecimientos relativos a la existencia, naturaleza y propiedad de dicho objeto. El término surgió en el ámbito de las obras de arte y su aplicación se ha ido trasladando a lo largo del tiempo a otras áreas hasta alcanzar, en los últimos años, al ámbito de las tecnologías de la información y las ciencias de la computación. Conocer la procedencia de un objeto y mantener la traza de la misma, posibilita disponer de evidencias de la fabricación inicial o descubrimiento del objeto, así como de su historia posterior. Otro de los beneficios de conocer la procedencia de un objeto, es que a partir de ella es posible autenticarlo, es decir, asegurar que el objeto es el que debe ser.

En el ámbito de las ciencias de la computación, el término *procedencia* refiere al ciclo de vida de los datos (*data lineage*) e incluye tanto su origen como sus cambios a lo largo del tiempo desencadenados por la ejecución de diferentes procesos; en este ámbito, la procedencia es referida como *procedencia de los datos* (*data provenance*). La procedencia de los datos contempla dos aspectos fundamentales: (1) la propiedad de los datos, y (2) el uso de los datos. La propiedad de los datos refiere al responsable del origen de los mismos. El uso de los datos proporciona información sobre cómo un dato ha podido ser tratado y modificado, o incluso sobre cómo se ha utilizado. El interés científico por la procedencia de los datos ha crecido exponencialmente en los últimos años [48], de hecho, es tal, que el W3C (World Wide Web Consortium) ratificó en el año 2013 un estándar para su representación denominado *PROV* [33] que vino a incrementar la lista de sistemas de representación de la procedencia ya existentes como el *Provenance Markup Language* (PML) y el *Open Provenance Model* (OPM).

La *gestión de los procesos de negocio* (*Business Process Management*; BPM) es una disciplina que se centra en la mejora del desempeño (eficiencia y eficacia) de una organización mediante la gestión y optimización de sus procesos de negocio. Dicho de otra forma, la gestión de los procesos de negocio abarca conceptos, métodos y técnicas para dar soporte al diseño, administración, configuración, mejora y análisis de los procesos de negocio [69]. En este ámbito, los procesos de negocio se definen como secuencias de actividades o tareas que tienen por objetivo producir un determinado producto o servicio para un cliente, es decir, alcanzar un objetivo de negocio. Estas secuencias de actividades o flujos de trabajo (*workflows*), por lo general, se representan mediante diagramas de flujo (*flowcharts*).

La *monitorización* es una fase primordial dentro del ciclo de vida de la gestión de los procesos de negocio, que además se encuentra estrechamente relacionada con esta tesis. La fase de monitorización consiste en el seguimiento, valoración del desempeño y obtención de datos estadísticos de los resultados de la ejecución de cada uno de los procesos de negocio. La fase de monitorización abarca tanto el registro de la información sobre la ejecución de los

diferentes procesos de negocio, como la extracción de conocimiento que pueda facilitar la mejora de los procesos de negocio. Esta fase también es conocida a través de su nombre completo: *Monitorización de Procesos de Negocio* (*Business Process Monitoring*). La monitorización de procesos de negocio está estrechamente relacionada con la procedencia a través de su objetivo común: Extraer información valiosa de la ejecución de los diferentes procesos de negocio; entre dicha información se incluye cómo fueron generados los datos, que recursos fueron necesarios y que actividades se ejecutaron [20], es decir, información sobre su procedencia.

El *registro de información* por los sistemas de monitorización de procesos de negocio se realiza, por lo general, almacenando una secuencia de eventos que ocurren sobre los procesos de negocio [60], señalando para cada evento la fecha en la que ha sucedido, su ejecutor, así como cierta información adicional [57, 67]. Para que esto sea posible, se requiere soporte de un tipo de software denominado *Business Process Management Suite* (BPMS) que puede ser extendido por otro tipo de software denominado *Business Activity Monitoring* (BAM), especializado en la monitorización de las actividades de negocio en tiempo real. La procedencia por tanto pretende aportar una perspectiva histórica a este tipo de software [20].

Sin embargo, ante esta forma de registrar información surgen dos problemáticas:

1 Se ignora el registro de información sobre aspectos importantes del sistema: Los objetos afectados por un evento, los cambios de estado producidos en dichos objetos o los protocolos seguidos por los usuarios del sistema. Es decir, información relevante para entender el comportamiento del sistema quedaría excluida al obtener la procedencia de los datos.

2 El registro de la información se suele realizar en logs de eventos [66], los cuales suelen corresponder con estructuras de datos lineales como ficheros, tablas [67] u hojas de cálculo [30]. El principal inconveniente de esta práctica es que la extracción de conocimiento del sistema y la obtención de la procedencia de los datos requiere de complejos procesos de minería de datos (*data mining*) [63], a lo que hay que añadir la problemática indicada de que parte de la información requerida podría no estar registrada [50].

El registro de datos en este tipo de sistemas se realiza generalmente sobre bases de datos relacionales, aunque también puede existir información registrada en otro tipo de almacenes de persistencia, por ejemplo, en documentos XML [68] o como parte de la cabecera de un fichero en forma de meta-datos. Estos datos se extraen utilizando consultas SQL o lenguajes equivalentes (en caso de estar registrados en bases de datos relacionales), expresiones XPath (para el acceso a estructuras XML), etc. y, posteriormente, se procesan aplicando técnicas de minería de datos con la intención de obtener la información requerida por las tareas de monitorización de los procesos de negocio o para acceder la procedencia de los datos (*provenance dissemination*) [63]. Ante este proceso surge la tercera problemática:

3 Dependiendo de las características y tamaño de la estructura del repositorio de datos, el desarrollo de las consultas y de los procesos de extracción y tratamiento de la información puede llegar a ser altamente complejo pudiendo ser únicamente realizado por personal especializado con profundos conocimientos de la estructura del repositorio de los datos. La problemática se agrava si se tiene en cuenta que, en muchos casos, no se dispone de procesos automatizados para el registro de la información sobre la procedencia de los datos y por tanto se realiza de forma manual.

1.2 Motivación y objetivos

La motivación de esta tesis es aportar soluciones a las problemáticas descritas en el apartado anterior, concretamente a las que tienen implicaciones sobre los mecanismos para el registro y el acceso a la procedencia de los datos.

El principal objetivo de esta tesis es la definición de un lenguaje de consultas para el tratamiento de la procedencia de los datos en *Sistemas de Monitorización de Procesos de Negocio* (*Business Process Monitoring Systems*); como medio para lograrlo, ha sido necesario realizar la especificación de un *framework*, del que forma parte el lenguaje, que actúa como instrumento para facilitar la construcción de sistemas de este tipo dotados de una característica esencial: estar preparados para gestionar la procedencia de los datos [25].

Los trabajos de esta tesis se enmarcan y se apoyan en las investigaciones realizadas por el *Grupo Nóesis de la Universidad de Zaragoza* con el objetivo de solventar otra parte de las problemáticas descritas: la relativa a la arquitectura de la información, a la estructura de los repositorios para el almacenamiento de la misma y, especialmente, la relativa a su explotación y tratamiento. En estas investigaciones se propone un tipo de sistema de monitorización de procesos de negocio basado en acontecimientos, y se presenta el concepto *Acontecimiento* (*Occurrence*) *Basado en Protocolos*, definido en [24] de la siguiente forma:

> *"An occurrence is a concrete, identifiable and indivisible chunk of information which contains aspects organized according to three dimensions: guidance, structure, and behavior, given by protocol execution, object, and effect."* y considerado como *"una entidad atómica"* donde *"la terna constituida por ejecución de un 'protocolo, objeto, datos'"*

Sobre el concepto *Acontecimiento*, se asienta un tipo de estructura más avanzado denominado *Base de Acontecimientos* (*Occurrence Base*; *OcBase*) para el almacenamiento de acontecimientos; y un tipo de software denominado *Sistema de Gestión de Acontecimientos* (*Occurrence Management System*; *OcSystem*) que posibilita la gestión integral de los mismos.

Si alineamos estos conceptos con aquellos propios de la gestión de procesos de negocio, podemos decir que un acontecimiento corresponde con el resultado de la ejecución de un proceso de negocio sobre un objeto de negocio que causa un efecto o cambio sobre el mismo. Por otra parte, recordemos la definición dada al concepto *procedencia de un objeto*: *"Conjunto de acontecimientos relativos a la existencia, naturaleza y propiedad de dicho objeto"*, definición a través de la cual se establece una relación directa entre *acontecimiento* y *procedencia* (*provenance*), siendo el primer concepto parte del segundo.

De la necesidad de disponer de herramientas para realizar el tratamiento de la información almacenada en bases de acontecimientos, el mantenimiento de sus estructuras, y facilitar la construcción de sistemas de gestión de acontecimientos surge la idea de desarrollar un lenguaje de consultas específicamente ideado para estos cometidos. La denominación dada a este lenguaje de consultas, cuya primera versión ha sido publicada en [25], fue *Occurrence Query Language* (*OcQuery Language*; *OcQL*) o *Lenguaje de Consultas para la Gestión de Acontecimientos*.

El desarrollo completo del *Lenguaje de Consultas para la Gestión de Acontecimientos* es actualmente un tema abierto de investigación por parte del *Grupo Nóesis de la Universidad de Zaragoza* dado que en su extensión prevista se espera dotarlo de tres partes:

1 Acceso a acontecimientos a través del *Lenguaje de Acceso a Acontecimientos* (*Occurrence Access Language*; *OcAL*) y de su extensión, el *Lenguaje de Acceso a Información de*

Acontecimientos (*Occurrence Information Access Language*; *OcIAL*).

2 *Lenguaje para la Definición de Acontecimientos* (*Occurrence Definition Language*; *OcDL*).

3 *Lenguaje para la Modificación de Acontecimientos* (*Occurrence Modification Language*; *OcML*).

El alcance de la presente tesis, abarca la definición y desarrollo de la primera parte del lenguaje, la cual posibilita acceder a los acontecimientos y, en general, a la información almacenada en bases de acontecimientos, facilitando obtener la procedencia de los datos registrados en sistemas de monitorización de procesos de negocio basados en acontecimientos.

Para lograr que el lenguaje tuviera el alcance previsto, en primer lugar, se realizó la especificación de un *framework* al que se denominó *Occurrence Query Framework* (*OcQuery Framework; OcQF*). El framework está formado por los siguientes componentes:

1 El *Metamodelo de Bases de Acontecimientos* (*Occurrence Base Metamodel*; *OcBase Metamodel*), que contiene los conceptos que pueden utilizarse en la especificación de bases de acontecimientos.

2 El *Lenguaje de Consultas para la Gestión de Acontecimientos*.

3 Las *Consultas* concretas *para la Gestión de Acontecimientos*.

4 El *Modelo de Base de Acontecimientos* basado en el metamodelo.

5 El *Repositorio de Metadatos* o *Diccionario*, basado en la estructura del metamodelo, responsable de almacenar la información que describe la estructura de una base de acontecimientos.

6 El *Traductor de Consultas* del lenguaje de acontecimientos.

7 La *Base de Acontecimientos* resultante de implementar el *modelo de base de acontecimientos*.

Por las características con las que se le ha dotado, el lenguaje de consultas se caracteriza y aporta lo siguiente:

1 Independencia de la tecnología.

2 Abstracción al usuario final de la estructura de datos en la que se encuentra almacenada la información.

3 Simplicidad, con una sintaxis que resultará familiar a los usuarios de otros lenguajes de consulta mundialmente extendidos como SQL [42].

A través del lenguaje propuesto, es el sistema y no el usuario el que conoce de donde extraer la información solicitada; para ello las consultas OcQL son procesadas por el traductor (*OcQL Translator*) que las transforma en el lenguaje soportado por el sistema de persistencia sobre el que se encuentre implantada la base de acontecimientos (por ejemplo, un sistema gestor de bases de datos relacionales que acepta consultas SQL). Para realizar su cometido, el traductor utiliza la información registrada en el *Repositorio de Metadatos* (*Occurrence Base Metadata Repository*; *OcBase Metadata Repository*) o *Diccionario* pudiendo conocer de esta forma, la estructura de la base de acontecimientos que almacena la información a consultar.

1.3 Principales aportaciones

Las principales aportaciones que realiza esta tesis son las siguientes:

1 Un *framework* (*Occurrence Query Framework*) que proporciona componentes y una metodología para dar soporte a los procesos de construcción de *sistemas de gestión de acontecimientos*.

2 Un *metamodelo* (*Occurrence Base Metamodel*) que normaliza los conceptos aplicados en las estrategias de diseño basadas en acontecimientos y en la definición de *modelos de bases de acontecimientos*.

3 Un *lenguaje de consultas* (*Occurrence Query Language*) para facilitar el acceso a la información almacenada en *bases de acontecimientos*, independiente de la tecnología y que abstrae al usuario final de la localización de la información.

En resumen, esta tesis aporta una serie de artefactos que facilitan la construcción de un tipo de *Sistema de Monitorización de Procesos de Negocio* basado en acontecimientos al que se denomina *Sistema Gestor de Acontecimientos*, un tipo de sistemas que, de forma natural, posibilitan gestionar y extraer la *procedencia de los datos*.

1.4 Estructura de la tesis y metodología

Tanto el trabajo realizado como el resultado del mismo plasmado en este documento de tesis se ha dividido en los siguientes bloques o fases:

1 **Definición de la base teórica** en la que se apoya el desarrollo de la tesis.

Durante esta fase se realizó un estudio del estado del arte de los conceptos y tecnologías existentes tomados como punto de partida para el desarrollo de esta tesis. Las conclusiones de este estudio y las problemáticas detectadas que han motivado la elaboración de esta tesis se han presentado en este mismo capítulo.

Las investigaciones realizadas por el *Grupo Nóesis de la Universidad de Zaragoza*, en las que se fundamenta esta tesis, han sido reseñadas como contexto de la misma en el apartado 1.5 y descritas en profundidad en el Capítulo 2.

El trabajo relacionado realizado por otros autores ha sido descrito en el apartado 2.3.

El sistema informático *Arasis*, referido en diferentes ejemplos incluidos a lo largo de este documento se presenta en el apartado 2.4.

2 **Definición del framework** para la gestión de acontecimientos.

El Capítulo 3 está dedicado al framework, al que hemos denominado *Framework para la Gestión de Acontecimientos* o *Framework OcQF*; en dicho capítulo se muestra su arquitectura y se describen cada uno de los componentes que lo forman y explicando la funcionalidad que ofrecen cada uno de ellos.

Los componentes fundamentales del framework como son el *Metamodelo de Bases de Acontecimientos*, el *Traductor de Consultas* y el *Lenguaje de Consultas para la Gestión de Acontecimientos* se han explicado detalladamente en capítulos y apartados propios. El Capítulo 7 y el Capítulo 11 están dedicados a presentar dos especificaciones del *Metamodelo de Bases de Acontecimientos*, la publicada en [25] y su extensión, así como cada uno de los conceptos contemplados en las mismas. El funcionamiento del traductor y

del proceso de traducción se explican en el Capítulo 5. Del lenguaje, que es el pilar de esta tesis, hablaremos a continuación.

3 **Definición del lenguaje** de consultas para la gestión de acontecimientos.

El Capítulo 4, el Capítulo 7 y el Capítulo 11 están dedicados a presentar en profundidad el *Lenguaje de Consultas para la Gestión de Acontecimientos* con el alcance abordado en esta tesis: el desarrollo de la parte del lenguaje que posibilita definir consultas para acceder a la información almacenada en bases de acontecimientos.

En el Capítulo 4 se explican las características generales del *Lenguaje de Consultas para la Gestión de Acontecimientos* (*OcQL*) y se presentan sus diferentes partes.

En el Capítulo 7 se describe el *Lenguaje de Acceso a Acontecimientos* (*OcAL*) en sus variantes no generalizada y generalizada. OcAL posibilita obtener acontecimientos y elementos de los mismos tanto de forma independiente como formando parte de una estructura longitudinal como la *historia* o la *línea de vida*.

En el Capítulo 11 se presenta el *Lenguaje de Acceso a Información Acontecimientos* (OcIAL). OcIAL posibilita el acceso a la información completa almacenada en una base de acontecimientos, así como la obtención de información sobre su estructura, registrada en uno de los componentes del framework, el *Repositorio de Metadatos* o *Diccionario*.

4 **Validaciones** de lo realizado.

La validación del lenguaje se realizó, por una parte, a través de una prueba de concepto en la que se utilizó una implementación del traductor de consultas y, por otra parte, desarrollando la semántica formal del lenguaje la cual se ha incluido en el Capítulo 11.

La prueba del lenguaje se realizó sobre la base de acontecimientos de un sistema real, el sistema informático *Arasis* que da soporte a "Aragón Workers Health Study" (AWHS), proyecto dirigido al almacenamiento y estudio de muestras biológicas de los empleados de General Motors España [13].

La base de acontecimientos del sistema *Arasis* se encuentra implantada sobre un gestor de bases de datos relacionales, por lo que el lenguaje compatible con este último es SQL. La prueba consistió en seleccionar un conjunto de consultas del lenguaje de acontecimientos y traducirlas al lenguaje SQL; las consultas traducidas se ejecutaron sobre la base de acontecimientos del sistema *Arasis*.

Esta prueba de concepto posibilitó contrastar la validez del lenguaje, medir la complejidad de cada consulta en cada uno de los lenguajes (OcQL y SQL), y evaluar el rendimiento en términos del tiempo de ejecución necesario para obtener la información requerida. El Capítulo 8 de este documento de tesis está dedicado a presentar los detalles de esta prueba.

5 **Conclusiones y trabajo futuro**.

Este documento de tesis finaliza con el Capítulo 12, en que se exponen las conclusiones obtenidas con la realización de la tesis y se presentan las líneas de trabajo a abordar con posterioridad a la misma.

1.5 Contexto

El pilar fundamental sobre el que se ha cimentado esta tesis, han sido las investigaciones realizadas por el *Grupo Nóesis de la Universidad de Zaragoza* en los siguientes ámbitos:

- Bases de datos longitudinales.
- Bases de acontecimientos.
- Estrategias de diseño basadas en acontecimientos.
- Sistemas de gestión de acontecimientos.

Estas investigaciones han sido objeto de diversas publicaciones como [1, 2, 24, 25, 26] y se han desarrollado a través de proyectos como LISBB (TSI-020302-2008-8), QRP (TSI-020302-2009-28), THOFU (CEN-20101019) y SMOTY (IPT-2011-1328-390000) en el marco del Plan Nacional de I+D y TBBTU (INNOVA-A1-88/11), proyecto que contó con financiación del Gobierno de Aragón.

LISBB (*LISBioBank*) fue financiado por el *Ministerio de Industria, Turismo y Comercio* en la convocatoria del *Plan Avanza 2* del año 2008 (Expediente TSI-020302-2008-8). Fue un proyecto desarrollado en consorcio por *Idea Informática S. A.* (Grupo Gesfor), la *Universidad de Zaragoza* e *Infozara Consultoría Informática S. L.*, durante los años 2008 y 2009, con un presupuesto de 1.012.873 euros. En LISBB se desarrolló una herramienta informática para la gestión de biobancos cuya información queda almacenada bajo la forma de una base de datos longitudinal, y se investigó sobre la explotación de la información dirigida a facilitar el análisis longitudinal de la información almacenada.

QRP (*QualityReadyPortal: Portal para el aseguramiento de la calidad*) fue financiado por el *Ministerio de Industria, Turismo y Comercio* en la convocatoria del *Plan Avanza 2* del año 2009 (Expediente TSI-020302-2009-28). Fue un proyecto desarrollado en consorcio por *Informática Gesfor S. A.*, la *Universidad de Zaragoza* e *Infozara Consultoría Informática S. L.*, durante los años 2009 y 2010, con un presupuesto de 930.776 euros. En QRP se desarrolló una plataforma informática para el aseguramiento de la calidad en la producción de software basada en el modelo CMMI; la información generada por esta plataforma queda almacenada bajo la forma de una base de datos longitudinal.

THOFU (*Tecnologías del Hotel del Futuro*) fue financiado por el *CDTI* en la convocatoria del *Programa CENIT* del año 2009 (Expediente CEN-20101019). Fue un proyecto desarrollado por un consorcio formado por 15 entidades y 19 grupos de investigación entre los que se encontraba el *Grupo Nóesis de la Universidad de Zaragoza*, durante los años 2011 a 2013. Durante la participación en este proyecto se desarrolló el concepto Base de Acontecimientos y se investigó sobre el uso de la historia del sistema como parte de los procesos de seguridad en los sistemas informáticos de los hoteles.

SMOTY (*Un Sistema de Seguridad Basado en Inteligencia Emergente en el Internet de las Cosas*) fue financiado por el *Ministerio de Economía y Competitividad* en la convocatoria del *Subprograma INNPACTO* del año 2011 (Expediente IPT-2011-1328-390000). Fue un proyecto desarrollado en consorcio por *Lógica*, *Infozara Consultoría Informática S. L.*, *Complu Soft S. L.*, la *Universidad de Zaragoza*, la *Universidad de Alcalá* y la *Universidad Carlos III de Madrid*, durante los años 2011 a 2014, con un presupuesto de 2.340.679,80 euros. En SMOTY se investigó sobre el uso de estructuras longitudinales basadas en acontecimientos, concretamente en la *historia* para la autenticación entre dispositivos del Internet de las Cosas (*Internet of Things; IoT*) que se comunican entre sí de forma autónoma.

TBBTU (*Plataforma para la Gestión Inteligente de Muestras de Tumores en el Internet de las Cosas*) fue financiado por el *Departamento de Industria e Innovación del Gobierno de Aragón* en la convocatoria del *Programa INNOVARAGON* del año 2011 (Expediente INNOVA-A1-88/11). Fue un proyecto desarrollado en consorcio por *Infozara Consultoría Informática S. L.* y el *Grupo B90: Investigación en Anatomía Patológica Comparada de la Universidad de*

Zaragoza, durante los años 2011 y 2012, con una financiación de 120.00 euros. En TBBTU se adaptó la herramienta *TechBioBank* (el sistema informático *Arasis*) a la gestión de biobancos de tumores introduciendo, entre otras, las siguientes innovaciones: gestión de acontecimientos, implementación del origen de los datos (*data provenance*).

El autor de esta tesis ha participado en estos proyectos durante sus relaciones contractuales con la *Universidad de Zaragoza* y la empresa *Infozara Consultoría Informática S. L.* y ha sido co-autor de las publicaciones [1, 2, 25, 26] que se detallan en el siguiente apartado. Por otra parte, el resultado del Proyecto de Fin de Carrera realizado por el autor en la *Universitat Oberta de Catalunya* en el ámbito de la minería de datos fue objeto de la publicación *"Mining Navigation Patterns in a Virtual Campus"* [71].

1.6 Publicaciones

El doctorando ha participado en los siguientes artículos en los que figura como co-autor que, o bien tratan directamente los conceptos sobre los que se ha investigado, o bien han sido publicados como parte del desarrollo de nuevos conceptos e hipótesis dentro de la línea de investigación del *Grupo Nóesis de la Universidad de Zaragoza*:

- Allué, A., López, A., Ciria, J. C., Domínguez, E., Francés, Á., Zapata, M. A. (2016) *The task-oriented occurrence pattern*. In Proceedings of the 21st European Conference on Pattern Languages of Programs (p. 6). ACM. http://dx.doi.org/10.1145/3011784.3011790.

  Artículo publicado para el congreso EuroPLoP de julio de 2016, donde se describen los conceptos del patrón de evaluación de desempeño basado en un tipo concreto de estructura longitudinal basada en acontecimientos, la historia.

- Allué, A., Domínguez, E., López, A., Zapata, M. A. (2013) *QRP: A CMMI Appraisal Tool for Project Quality Management*. Procedia Technology, Volume 9, Pages 664-669, ISSN 2212-0173, http://dx.doi.org/10.1016/j.protcy.2013.12.073.

  Artículo publicado para el congreso PRojMAN de 2013, en el que se describe una aplicación de gestión de calidad siguiendo el modelo CMMI, en cuya base de datos se utilizan las estructuras longitudinales primitivas asociadas a los procesos de negocio de CMMI que dan origen al concepto de acontecimiento.

- Domínguez, E., Pérez, B., Rubio, A. L., Zapata, M. A., Allué, A., López, A. (2017) *Developing provenance-aware query systems: an occurrence-centric approach*. Knowledge and Information Systems 50, pp 661–688, DOI 10.1007/s10115-016-0950-z.

  Artículo publicado en revista JCR con factor de impacto en el tercio superior, en el que se presenta el *Lenguaje de Consultas para la Gestión de Acontecimientos* que ha sido objeto de esta tesis.

- Domínguez, E., Pérez, B., Rubio, A. L., Zapata, M. A., Allué, A., López, A. (2017) *Generating persistence structures for business process monitoring purposes: the Occurrence–centric approach*, enviado a 15th International Conference on Business Process Management (BPM 2017)

  Artículo enviado al 15th International Conference on Business Process Management (BPM 2017) con una propuesta que promueve la generación automática de estructuras de almacenamiento para la monitorización utilizando el concepto de acontecimiento.

Por otra parte, el doctorando, ha participado como co-autor en el siguiente artículo enmarcado

en el ámbito de la minería de datos y los patrones de navegación, resultante de su Proyecto de Fin de Carrera para obtener la titulación de Ingeniero en Informática por la *Universitat Oberta de Catalunya*:

- Xhafa, F.; López, A.; Caballe, S.; Kolici, V.; Barolli, L. (2012) *Mining Navigation Patterns in a Virtual Campus*. Emerging Intelligent Data and Web Technologies (EIDWT), Third International Conference on, vol., no., pp.181-189, 19-21 Sept. 2012. doi: 10.1109/EIDWT.2012.65.

Capítulo 2. Marco de referencia

En este capítulo se muestran los principales conceptos previos tomados como referencia para la elaboración de esta tesis. Según se irá indicando cuando proceda a través de las referencias correspondientes, varios de estos conceptos han sido introducidos por el Grupo Nóesis de la Universidad de Zaragoza durante la investigación en la que se enmarca esta tesis y parte de ellos han sido desarrollados en la tesis doctoral *"Modelo de Acontecimientos para la Persistencia"* presentada por Alberto Allué Gil (Universidad de Alcalá; 2017).

2.1 La aproximación basada en acontecimientos

2.1.1 El concepto Acontecimiento

Según el diccionario de la *Real Academia Española* (RAE), el término *Acontecimiento* viene de *acontecer*. Por lo tanto, teniendo esto presente, podemos decir que un acontecimiento es la percepción del resultado de un acontecer.

Por ejemplo, en el dominio de los biobancos o bancos de muestras biológicas (al que referiremos como caso de referencia a lo largo de este documento de tesis), *Registro*, *Alicuotado*, *Almacenamiento* o *Cesión* de muestras biológicas son posibles tipos de acontecimientos. Desde el punto de vista en los fenómenos que nos rodean, cuando se habla de la lluvia, se está haciendo referencia al acontecimiento resultante del proceso *Llover*.

A la vista de estos ejemplos, se puede percibir que los acontecimientos pueden clasificarse según sean acontecimientos naturales o acontecimientos artificiales. Se habla de *acontecimiento natural* para referir a acontecimientos relativos a fenómenos naturales como la lluvia o el nacimiento de una persona. Por el contrario, se habla de *acontecimientos artificiales* cuando estos se producen por fenómenos causados por la acción humana o por una máquina.

Pero la principal diferencia entre ambos tipos de acontecimiento es que, en el caso de los acontecimientos naturales, no es posible definir, al menos de forma sencilla, una estructura para registrar fielmente todos los elementos que intervienen. Por el contrario, en el caso de los acontecimientos artificiales, por lo general, es posible identificar de forma concreta los pasos seguidos que han desencadenado el acontecimiento, quien lo ha hecho, en qué momento temporal se ha producido, cual ha sido su duración, etc.

Por ejemplo, en el caso del *Alicuotado* de una muestra biológica, es posible identificar al técnico de laboratorio que ha realizado el alicuotado, el protocolo médico que ha seguido, cuándo ha realizado el alicuotado y a partir de qué muestra biológica ha extraído las alícuotas; si además se logra mantener la traza completa, por ejemplo, cómo se ha originado la muestra biológica, se podría saber de qué sujeto de estudio se extrajo, cuándo se extrajo, el personal

sanitario responsable, etc. Toda esta información puede ser registrada en sistemas de información y, en el caso que nos ocupa, en las estructuras a las que se ha denominado *Bases de Acontecimientos*.

En [24], el concepto *Acontecimiento* (*Occurrence*) se define de la siguiente forma:

> *"An occurrence is a concrete, identifiable and indivisible chunk of information which contains aspects organized according to three dimensions: guidance, structure, and behavior, given by protocol execution, object, and effect."*

Es decir, según esta definición, un acontecimiento puede entenderse por un conjunto formado por tres elementos: *Ejecución de protocolo*, *Objeto* y *Efecto*. Dicho de otra forma, podemos definir acontecimiento como el resultado de la ejecución de un protocolo que causa un efecto sobre un objeto. A este tipo de acontecimientos se le denomina *Acontecimientos Basados en Protocolos*.

Los conceptos que describen a un acontecimiento basado en protocolos pueden definirse de la siguiente manera:

- Entendemos *Proceso* como una tarea concreta, identificable y realizada de principio a fin que podemos conceptualizar como un conjunto de *Protocolos*.
- Entendemos *Protocolo* como la especificación de un conjunto estructurado y ordenado de actividades definido para llevar a cabo un objetivo.
- Cada *Protocolo*, cuando se ejecuta, actúa sobre uno o varios *Objetos* causando un *Efecto* sobre ellos; es decir, los *Objetos* reaccionan cambiado su estado y/o datos relacionados. A cada una de esas ejecuciones la denominamos *Ejecución de Protocolo*.

Más allá de la definición dada, un acontecimiento puede representarse de forma genérica como una tupla formada por tres elementos:

(A, O, E)

En esta tupla, el elemento *A* representa a la acción realizada para desencadenar el acontecimiento, el elemento *O* representa al objeto afectado por el acontecimiento, mientras que el elemento *E* representa al efecto producido sobre el objeto.

En el caso concreto de los acontecimientos basados en protocolos, la tupla anterior, queda particularizada de la siguiente manera, más cercana a la definición dada en [24]:

(Ep, O, C)

En este caso, la acción refiere a la ejecución de un protocolo (*Ep*), y el efecto producido sobre el objeto (*O*) corresponde con un cambio (*C*) en dicho objeto; por ejemplo, el cambio puede ser un cambio de estado del objeto o un cambio en sus datos.

Tomando como referencia esta tupla, podríamos representar un acontecimiento concreto de tipo *Alicuotado* registrado en un sistema informático que diera servicio a un biobanco de la siguiente forma:

('sampleAliquotingProtocolExecution002', 'bloodSample007', 'aliquotedState003')

De esta forma se estaría representando el hecho de que la ejecución de protocolo identificada en el sistema como 'sampleAliquotingProtocolExecution002' ha actuado sobre la muestra de sangre (el objeto) identificada como 'bloodSample007' causando que dicha muestra quedara en un estado identificado como 'aliquotedState003'.

2.1.2 Estructuras longitudinales

El cambio es algo habitual en el día a día pudiendo ser natural o artificial. El desarrollo de la vida de una persona es un cambio natural, mientras que la variación de los niveles de colesterol de esa persona tras aplicarle una medicación es un cambio artificial. En la actualidad, el registro de datos de forma longitudinal puede facilitar la gestión del cambio [59].

Se puede decir que un conjunto de datos es longitudinal:

Si refiere al mismo tipo de información obtenida de un conjunto de entidades a lo largo del tiempo [53].

Si es el resultado de múltiples observaciones del mismo conjunto de entidades realizado en un conjunto conocido de momentos temporales separados por intervalos de tiempo regulares o irregulares [72].

Por ejemplo, los resultados de medir anualmente el nivel de colesterol, la tensión arterial, el nivel de azúcar en sangre, etc. de los sujetos de un estudio médico (*cohorte*) durante varios años serían casos de conjuntos de datos longitudinales.

El principal beneficio de los datos longitudinales es que, a través de su análisis (*análisis longitudinal*), es posible medir el cambio. Por ejemplo, es posible estimar el efecto de varios factores correctivos en la mejora del rendimiento académico de un estudiante [53], estudiar la mejora de los niveles de colesterol de una persona tras aplicarle un tratamiento, evaluar la mejora en los tiempos de entrada a un aparcamiento tras habilitar un nuevo acceso, etc.

Hablamos de un *conjunto de datos longitudinal balanceado* cuando todas las observaciones de cada una de las entidades que forman parte de un conjunto de datos longitudinal han sido realizadas en momentos temporales separados en intervalos regulares; si, por el contrario, para cada una de las entidades se realizan observaciones en momentos temporales separados por intervalos irregulares, se han omitido observaciones de alguna entidad en ciertos momentos, etc. hablamos de un *conjunto de datos longitudinal no balanceado*. Por ejemplo, el conjunto de datos longitudinal resultante de un estudio médico en el que se ha medido el nivel de colesterol a mil sujetos en los años 2009, 2010 y 2011 sería un conjunto de datos balanceado; si por alguna circunstancia, parte de los sujetos no hubiera sido observado en alguno de los años, estaríamos hablando de un conjunto de datos no balanceado.

Los datos longitudinales pueden estructurarse tanto de forma multivariada, como de forma univariada. Tradicionalmente, las estructuras de datos para representar repetidas observaciones han seguido la forma multivariada [72]. Las estructuras de datos multivariadas se representan de forma matricial; en ellas, cada entidad siempre es referida por una única fila de datos, mientras que las diferentes observaciones realizadas se registran horizontalmente; es decir, cada una de las columnas agrupa a un conjunto de observaciones realizadas en un mismo momento temporal.

ID	PCL1	PCL2	PCL3	PCL4	Age	Female
1	66	31	58	39	27	0
2	48	56	43	43	44	1

Tabla 2.1 Resultados de estudio del trastorno por estrés postraumático [73]. Representación multivariada

Al realizar análisis de datos longitudinales sobre este tipo de estructuras surgen diversos problemas [72]: En primer lugar, en la forma multivariada, el factor tiempo suele indicarse indirectamente cuando se identifica cada columna; esto implica que, de forma natural, presentan dificultades para representar conjuntos de datos no balanceados en los que, para ciertas entidades, se realizan observaciones en diferentes intervalos de tiempo. En segundo lugar, en el análisis de datos longitudinales, algunas variables pueden tener valores cambiantes a lo largo del tiempo (por ejemplo, la edad, el estado civil, el nivel económico, la situación laboral, etc.); los errores al valorar la variabilidad en el tiempo de estas variables pueden dar resultados incorrectos o sesgados.

Esta problemática ha hecho que la mayoría de los análisis longitudinales se realicen sobre estructuras univariadas [72]. En la forma univariada, cada entidad puede estar representada por varias filas (una por cada momento temporal en el que se ha realizado una observación); en este caso, el tiempo se representa incluyendo una nueva variable a través de una columna adicional en la matriz.

ID	Time	PCL	Age	Female
1	0	66	27	0
1	1	31	27	0
1	2	58	27	0
1	3	39	27	0
2	0	48	44	1
2	1	56	44	1
2	2	43	44	1
2	3	43	44	1

Tabla 2.2 Resultados de estudio del trastorno por estrés postraumático [73]. Representación univariada

Evidentemente, frente a las estructuras multivariadas, las estructuras univariadas se caracterizan por contar con un menor número de columnas a costa de incrementar el número de filas como puede observarse si comparamos la Tabla 2.1 con la Tabla 2.2. Por ello, hay que tener presente que cuantas más mediciones se realicen a lo largo del tiempo más se incrementará la longitud de la estructura en su forma univariada.

Estructuras longitudinales basadas en acontecimientos

El *Grupo Nóesis de la Universidad de Zaragoza*, como parte de la investigación en la que se enmarca esta tesis, propone un tipo de estructura para el tratamiento de datos longitudinales al que denomina *estructura longitudinal basada en acontecimientos*.

Para dar forma a este tipo de estructuras longitudinales, se parte de la siguiente definición de dato longitudinal:

> **Definición**. Partiendo de un objeto O y de una propiedad P(O) de dicho objeto que pueda ser medida a través del tiempo, una estructura longitudinal se define como un conjunto de valores vti de P(O) tomados a través del tiempo.

Dicho de otra forma, una estructura longitudinal es una sucesión infinita potencial (v_{t1}, v_{t2}, …) de valores de P(O) tomados en una secuencia de tiempos (t_1, t_2, …).

La innovación de este tipo de estructuras longitudinales se encuentra en que cada una de las observaciones que se realizan sobre un objeto a lo largo del tiempo se representa como un acontecimiento (concepto que hemos presentado al inicio de este apartado), en vez de como un dato simple como podría ser, por ejemplo, el nivel de colesterol medido en un determinado momento temporal.

En [24] se proponen dos tipos de estructuras longitudinales basadas en acontecimientos, *Historia* y *Línea de Vida*:

> **Definición 1**. Denominamos *Historia* de un objeto al conjunto formado por todos los acontecimientos ocurridos a dicho objeto y por los acontecimientos ocurridos sobre los objetos asociados a él.
>
> **Definición 2**. Denominamos *Línea de Vida* de un objeto al conjunto de todos los acontecimientos que le han ocurrido directamente a dicho objeto.

Partiendo de estas dos definiciones, se puede decir que la línea de vida de un objeto representa a un subconjunto de los acontecimientos que forman parte de la historia de dicho objeto. Dicho de otro modo, la historia de un objeto está formada por la línea de vida del mismo y por el conjunto de líneas de vida de los objetos asociados a él. Por ejemplo, la historia de una muestra biológica almacenada en un biobanco considera no sólo los acontecimientos sucedidos a dicha muestra, sino también el conjunto de acontecimientos ocurridos al rack donde está depositado, al ultracongelador donde se ha depositado el rack, etc.

En la siguiente figura se muestra, a modo de ejemplo, la representación de la línea de vida de un sujeto que ha participado en un estudio médico:

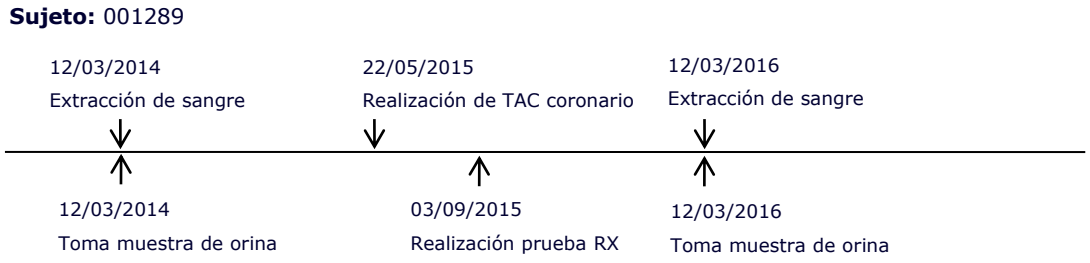


Figura 2.1 Representación gráfica de una línea de vida

La línea de vida representada en la Figura 2.1 correspondería al sujeto de estudio identificado por el código 001289 (objeto) y en ella se han representado seis acontecimientos: dos como resultado de la ejecución del protocolo 'Extracción de sangre' el 12/03/2014 y el 12/03/2016, dos como resultado de la ejecución del protocolo 'Toma muestra de orina' el 12/03/2014 y el 12/03/2016, otro como resultado de la ejecución del protocolo 'Realización de TAC coronario' el 22/05/2015 y un último acontecimiento como resultado de la ejecución del protocolo 'Realización prueba RX' el 03/09/2015.

La siguiente figura muestra una línea de vida tal y como es mostrada al usuario a través de la interfaz gráfica del sistema informático *Arasis* al que referiremos a lo largo de este documento de tesis y que presentaremos más adelante:

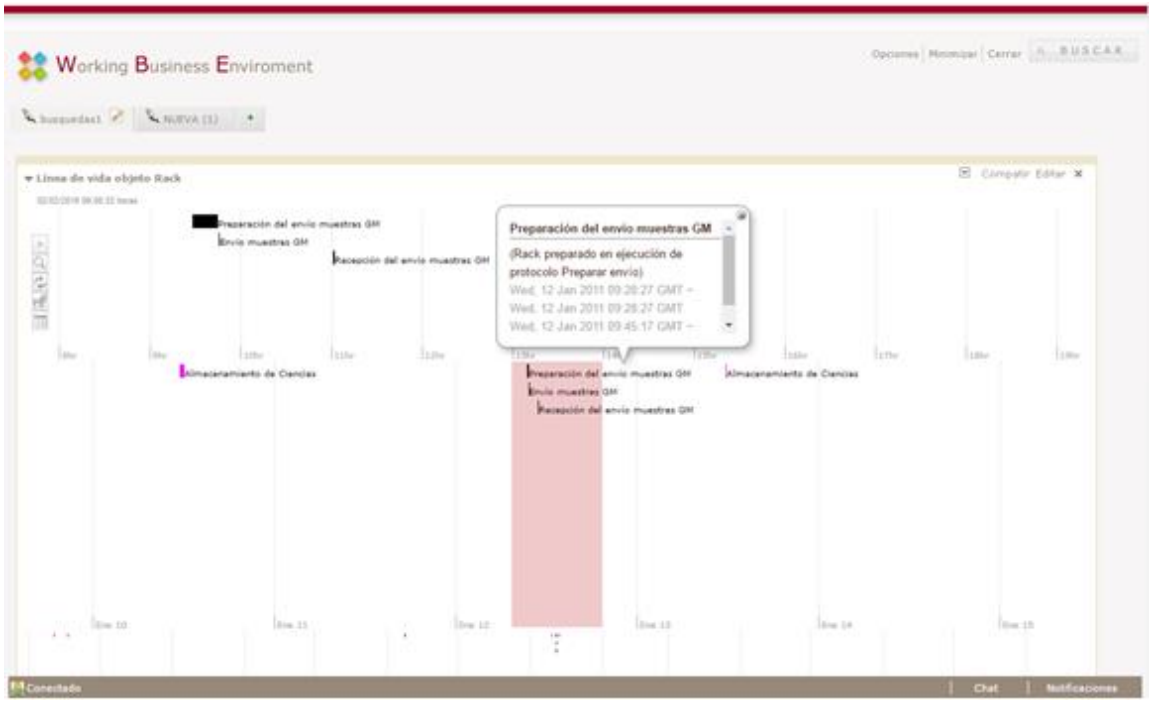


Figura 2.2 Línea de vida en el sistema informático *Arasis*

Este tipo de estructuras longitudinales es apto para representar tanto conjuntos de datos longitudinales balanceados como no balanceados y realiza una aportación fundamental a la hora de estudiar la procedencia de un objeto, hace posible seguir la traza completa de lo que le ha sucedido a un objeto a lo largo de su vida. Retomando el ejemplo que hemos puesto anteriormente, supongamos que se estropea la muestra, ¿cuál ha sido la causa?, ¿ha sido por un mal procesado?, ¿ha sido por una avería en el ultracongelador?, ¿ha sido por una mala manipulación por parte de un técnico de laboratorio?; analizando la historia de la muestra deberíamos poder determinar la secuencia de acontecimientos que ha provocado su deterioro.

Las estructuras longitudinales basadas en acontecimientos se han utilizado en diversos campos de aplicación. En los proyectos SMOTY (*SMOTY: Sistema de Seguridad Basado en Inteligencia Emergente en el Internet de las Cosas; IPT-2011-1328-390000; Subprograma INNPACTO 2011*) y THOFU (*THOFU: Tecnologías del Hotel del Futuro; CEN-20101019; Subprograma de Investigación de apoyo a Consorcios Estratégicos Nacionales de Investigación Técnica -CENIT-; 2010-2013*) se ha investigado sobre el uso de la historia para la autenticación entre dispositivos del Internet de las Cosas (*Internet of Things; IoT*) que se comunican entre sí de forma autónoma. En [2] se propone un caso de uso de la historia como parte de un patrón denominado *Task-Oriented Occurrence Pattern* que posibilita modelar las tareas que una entidad es capaz de realizar y evaluar su grado de habilidad para realizarlas con el objetivo de lograr asignarlas a los componentes de un sistema colaborativo que sean capaces de desempeñarlas de forma más óptima.

Aunque *Historia* y *Línea de Vida* son los tipos de estructura longitudinal que hemos considerado más relevantes para nuestra investigación y los que hemos utilizado con mayor frecuencia, es importante señalar que no son los únicos. Se denomina *estructura longitudinal basada en acontecimientos* al conjunto de acontecimientos que tienen una relación temporal sobre la que interesa obtener una información, por lo que se pueden definir más estructuras

de este tipo. Por ejemplo, podríamos considerar un tipo de estructura longitudinal que posibilitara obtener la huella que un objeto produce en otros objetos, que sería útil, por ejemplo, para evaluar la huella ecológica del procesado de alimentos; este tipo de estructura longitudinal, al igual que la línea de vida, es un subconjunto de la historia de un objeto dado que refiere al efecto que la existencia del objeto ha provocado en otros; es decir a los acontecimientos sucedidos por la acción directa del objeto sobre otros. O, por ejemplo, también podríamos considerar un tipo de estructura longitudinal que refiriera al conjunto de acontecimientos que configuran la ejecución completa de un proceso en el que intervinieran diferentes protocolos y objetos; en este caso la estructura longitudinal referiría a un proceso y no a un objeto y serviría, por ejemplo, para representar el flujo de trabajo realizado por varios robots trabajando conjuntamente.

2.1.3 Bases de datos longitudinales

Una *Base de Datos Longitudinal* es una base de datos estructurada de tal forma que posibilita el almacenamiento de datos a lo largo del tiempo; es decir, datos que tienen explícitamente asociado el momento temporal en el que se capturaron en el mundo real y, opcionalmente, el momento temporal en el que se produjo su registro en la base de datos. Desde este punto de vista, se pude decir que una base de datos longitudinal es una estructura longitudinal compleja a partir de la cual, aplicando procesos de tratamiento de datos, es posible extraer información en forma de estructuras longitudinales más simples.

En [8, 17, 18] se expone un punto de vista sobre las bases de datos longitudinales muy próximo al propuesto por el *Grupo Nóesis de la Universidad de Zaragoza*, aunque con diferencias relevantes. En estas tres publicaciones se indica que un dato longitudinal tiene tres dimensiones: *Entidad*, *Atributos* y *Tiempo*; es decir, que un dato longitudinal se representa por tres bloques de información: una entidad, un conjunto de datos propios de dicha entidad y un conjunto de datos temporales cuya validez es relativa a los datos de la entidad. Por el contrario, en la investigación del *Grupo Nóesis* en la que se enmarca esta tesis, se propone lo siguiente:

1 Utilizar el concepto *Objeto* en lugar de *Entidad*. Esto posibilita la construcción de sistemas que puedan utilizar todos los mecanismos del paradigma de la orientación a objeto: relaciones entre estados, herencia, agregación etc.

2 Introducir el concepto *Hecho* o *Evidencia* en lugar de trabajar con simples datos de información. La consideración de la información como *Hecho* o *Evidencia* implica que es necesario añadir información sobre lo que ha generado dicha información, es decir el procedimiento que se ha ejecutado.

   La introducción de este concepto en la investigación fue muy importante ya que ayudó a la normalización del concepto *Acontecimiento* y a la evolución hacia las *bases de acontecimientos*.

Así pues, el almacenamiento de datos longitudinales en las bases de datos longitudinales propuestas por el *Grupo Nóesis de la Universidad de Zaragoza*, implica el registro de:

1 El objeto al que corresponden los datos.

2 Los datos del procedimiento que se ha ejecutado.

3 Los datos propios del objeto generados por la ejecución del procedimiento.

4 El segmento temporal de validez de los datos.

El uso de bases de datos longitudinales en sistemas informáticos para el soporte de la investigación biomédica fue objeto del proyecto LISBioBank (*TSI-020302-2008-8; Subprograma Avanza I+D 2008*); su uso en sistemas para el soporte a la gestión de proyectos informáticos y el aseguramiento de la calidad fue objeto del proyecto QRP (*QRP: Portal para el aseguramiento de la calidad; TSI-020302-2009-28;* Subprograma Avanza I+D 2009) y publicado en [1].

Teniendo presente lo que acabamos de exponer, y recordando el concepto *Acontecimiento* presentado al inicio de este capítulo, se puede entender cómo esta visión de las bases de datos longitudinales derivó de forma natural hacia un nuevo tipo de estructura denominado *Base de Acontecimientos* del que vamos a hablar a continuación.

2.1.4 Bases de acontecimientos

De forma análoga a la estructura de información conocida como *Base de Datos*, en la que el *Dato* es considerado la unidad mínima de información, en [24] se propone un nuevo tipo de estructura denominada *Base de Acontecimientos* (*Occurrence Base*; *OcBase*) cuya principal característica es que su unidad mínima de información es el *Acontecimiento*.

> **Definición**. Una *Base de Acontecimientos* es una estructura de persistencia que registra acontecimientos sucedidos a lo largo del tiempo [2, 24].

Más concretamente, una *Base de Acontecimientos* es un tipo de estructura de información que registra los acontecimientos que han tenido lugar a lo largo del tiempo como resultado de un conjunto de ejecuciones de protocolos, así como los cambios de estado de los objetos afectados por las mismas.

Se considera que una *Base de Acontecimientos* es un tipo de estructura longitudinal dado que la información que almacena, los acontecimientos, llevan asociado explícitamente el momento temporal en el que sucedieron. De este tipo de estructuras es posible extraer subconjuntos de información en forma de otros tipos de estructuras longitudinales, por ejemplo, en forma de *Historia* o *Línea de Vida*.

La última de las características fundamentales de las bases de acontecimientos refiere a la eliminación de información de las mismas. Las bases de acontecimientos son estructuras de información en las que la eliminación de cualquier acontecimiento es una acción que, de forma natural está prohibida, incluso si el registro se ha realizado con o por error [24]. En caso de error, el acontecimiento afectado es marcado como no válido registrándose a la par la correspondiente incidencia. Únicamente se admite la eliminación de acontecimientos en casos excepcionales como por ejemplo para dar cumplimiento a requerimientos legales que exijan la eliminación de cierta información, por ejemplo, para dar cumplimiento a la *Ley Orgánica de Protección de Datos* (*LOPD*), o en bases de acontecimientos en las que, por requisito, los acontecimientos sucedidos con anterioridad a determinada fecha dejan de tener relevancia.

Base de acontecimientos frente a base de datos

Los *Sistemas Gestores de Bases de Datos Relacionales* (*Relational Database Management System; RDBMS*), o más concretamente las *Base de Datos* mantienen puntos en común con las *Bases de Acontecimientos* y unos mismos objetivos (el registro, el almacenamiento y la recuperación de información), pero existe una clara distinción entre ambas estructuras [24]

tal y como indicamos a continuación:

1 Las bases de acontecimientos son estructuras diseñadas según las especificaciones marcadas por un modelo de base de acontecimientos, específicamente preparadas para la persistencia de acontecimientos sucedidos a lo largo del tiempo.

2 La unidad mínima de información en las bases de datos es el *Dato*, mientras que el *Acontecimiento*, concepto que hemos presentado en el apartado 2.1, es la unidad equivalente en una base de acontecimientos.

3 En una base de acontecimientos, la eliminación de cualquier acontecimiento está prohibido de forma natural, incluso si el registro se ha realizado con o por error. En este caso, el acontecimiento es marcado como no válido registrándose a la par la correspondiente incidencia. Debe recordarse que el fin último que se pretende es la mejora de los procesos de negocio, por lo que esta forma de proceder puede aportar un valioso conocimiento a lo largo del tiempo como puede ser la causa de los errores durante el registro.

   A este respecto, es importante señalar que únicamente se admite la eliminación de acontecimientos ante casos excepcionales como por ejemplo para dar cumplimiento a requerimientos legales que exijan la eliminación de cierta información.

De forma natural, las bases de acontecimientos pueden utilizarse tanto para el almacenamiento de información como para su explotación [25]. Los procesos de explotación de la información almacenada en este tipo de estructuras se inician, bien con la extracción de uno o varios acontecimientos de forma aislada, bien recuperando un conjunto de los mismos como parte de otras estructuras longitudinales basadas en acontecimientos más simples, por ejemplo, en forma de *Historia* o de *Línea de Vida*.

2.2 Una estrategia de diseño basada en acontecimientos (OcBaseMD)

Al igual que las bases de datos son gestionadas a través de sistemas gestores de bases de datos; las bases de acontecimientos son gestionadas a través de lo que denominamos *Sistemas de Gestión de Acontecimientos* (*Occurrence Management Systems*; *OcSystems*) que en [24] se definen como:

> **Definición**. Un *Sistema de Gestión de Acontecimientos* es cualquier sistema que posibilite la gestión integral de acontecimientos, tanto su almacenamiento en una base de acontecimientos, como su explotación (de los acontecimientos y del conocimiento derivado de los mismos) a través de software específico.

Un *Sistema de Gestión de Acontecimientos* incluye una base de acontecimientos y posibilita, entre otras cosas, la recuperación de acontecimientos almacenados en la misma a través de estructuras longitudinales como pueden ser las que hemos citado en el apartado anterior: *Historia* y *Línea de Vida*. A través de un sistema gestor de acontecimientos es posible, por ejemplo, generar un conjunto de líneas de vida que refieran a una serie de objetos interrelacionados. Por las características inherentes a las estructuras longitudinales, podemos decir que un *Sistema Gestor de Acontecimientos* es capaz de proporcionar información sobre lo sucedido a un objeto en un determinado instante de tiempo o durante un periodo de tiempo y por ello, podemos definir un *Sistema Gestor de Acontecimientos* como un sistema longitudinal.

La estrategia de diseño

La construcción de sistemas de gestión de acontecimientos no sería posible sin la existencia de una estrategia de diseño.

Se entiende *Estrategia de Diseño* como una serie de elementos (arquitecturas, modelos, métodos, etc.) que guían el proceso a seguir para llevar a cabo un diseño.

En [24] se propone una estrategia de diseño que se ha utilizado para desarrollar entre otros, el sistema informático *Arasis*.

La estrategia de diseño, a la que se denomina *OcBase Model Driven Design Strategy* (*OcBaseMD Design Strategy*), está basada en la metodología conocida como *Ingeniería Dirigida por Modelos* o *Model Driven Engineering* (*MDE*) descrita en [5] y su aplicación se realiza de la forma descrita brevemente a continuación:

1 Definición del perfil y los patrones de acontecimientos

En este apartado se describe la infraestructura que posibilita la especificación de modelos de bases de acontecimientos recurriendo a modelos UML. Concretamente se utilizan:

- Diagramas de actividades para representar protocolos.
- Diagramas de estado para representar los estados de los objetos.
- Diagramas de clase para representar las estructuras de persistencia de las bases de acontecimientos.

La definición de un perfil UML (*UML profile*) posibilita personalizar y extender la sintaxis y semántica de UML [54], adaptando los diagramas de clases a las necesidades concretas de los modelos de bases de acontecimientos permitiendo modelar el concepto *Acontecimiento* descrito al inicio de este capítulo. Por lo general, la forma de extender el vocabulario de UML es mediante el uso de estereotipos de la forma que se mostrará más adelante.

Al perfil definido se le denomina *Perfil de Acontecimiento* (*Ocurrence Profile*) y su diseño puede verse en la siguiente figura:

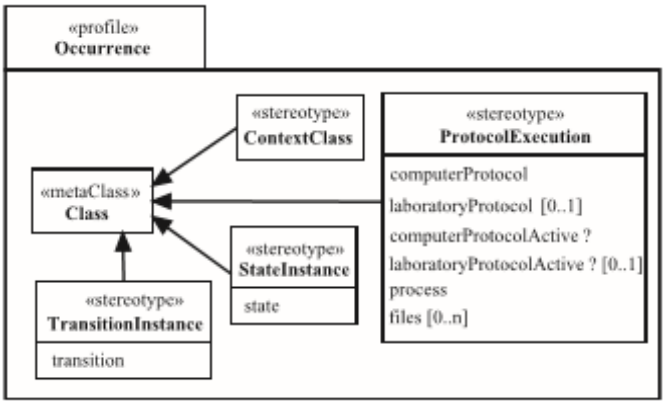


Figura 2.3 Perfil de acontecimiento particularizado al concepto de acontecimiento basado en protocolos

Este perfil define como estereotipos los cuatro elementos que dan forma al concepto de acontecimiento:

- El estereotipo <<ContextClass>> permite representar el concepto *Objeto*.
- El estereotipo <<StateInstance>> permite representar los estados por los que podrán pasar los objetos.
- El estereotipo <<TransitionInstance>> permite representar los cambios de estado.
- El estereotipo <<ProtocolExecution>> permite representar las ejecuciones de protocolo.

Haciendo uso de este perfil se definen dos patrones para representar acontecimientos, los cuales se denominan *Patrones de Acontecimiento* (*Occurrence Patterns*).

El concepto *Patrón* fue definido por Christopher Alexander en [3] de la siguiente forma:

> *"Each pattern describes a problem that occurs over and over again in our environment, and then describes the core solution to that problem, in such a way that you can use the solution a million times over, without ever doing it the same way twice"*.

Un patrón evita desarrollar una nueva solución cada vez que surge un problema aplicando una solución ya conocida al mismo. Este concepto formulado inicialmente en [3] para referir a patrones en ciudades y edificios se ha aplicado en el ámbito de la ingeniería del software [29].

Los patrones más relevantes definidos a través de esta estrategia de diseño, son los siguientes:

- Patrón Cambio de Estado (*Change State Occurrence Pattern*)

  Este patrón, definido en forma de diagrama de clases UML, especifica los componentes necesarios para representar un acontecimiento que implique, como efecto, el cambio de estado de un objeto. El patrón, representado en la siguiente figura, es aplicado durante el diseño de una base de acontecimientos.

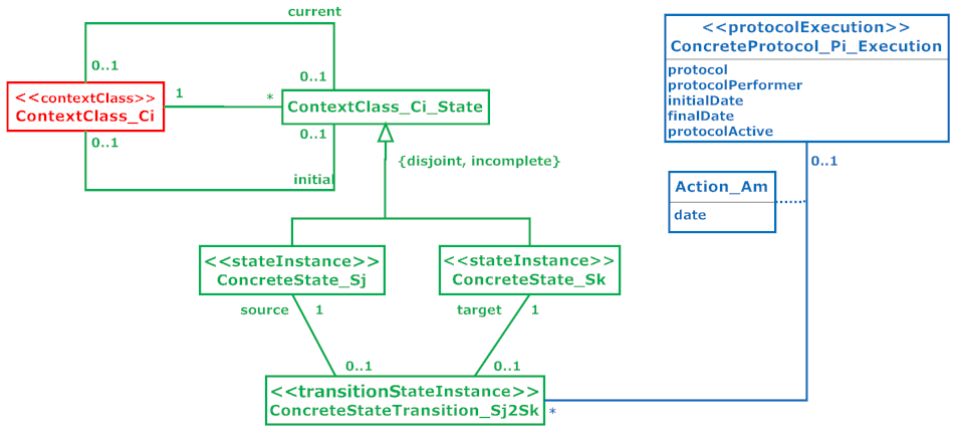


Figura 2.4 Patrón Cambio de Estado

- Patrón Creación de Objeto (*Object Creation Occurrence Pattern*)

  Este patrón, definido también como diagrama de clases UML, especifica los componentes necesarios para representar la creación de un objeto como resultado de la ejecución de un protocolo. La representación de este patrón puede verse en la siguiente figura:

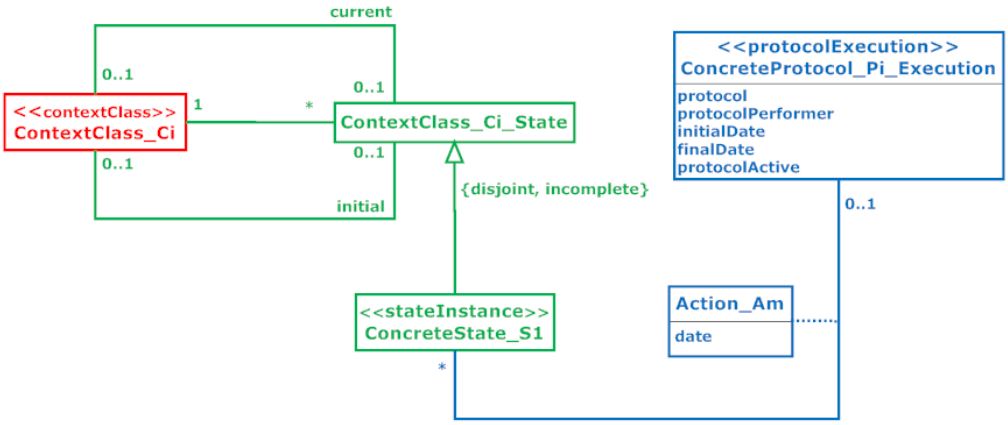


Figura 2.5 Patrón Creación de Objeto

2 Desarrollo de la metodología para la construcción de bases de acontecimientos

La metodología para la construcción de bases de acontecimientos se desarrolla aplicando la metodología MDA (*Model Driven Architecture*).

MDA o *Arquitectura Dirigida por Modelos* es una metodología para el desarrollo del software promovida por el OMG (*Object Management Group*) que propone realizar la especificación de un sistema en forma de modelos. De esta forma, la funcionalidad de un sistema se define en primer lugar como un modelo independiente de la plataforma (PIM) que posteriormente se trasforma a un modelo específico de la plataforma (PSM) sobre el que se realizará la implementación del sistema [55].

Así pues, para desarrollar la metodología, se trabaja en tres niveles utilizando respectivamente para cada uno de ellos:

- Diagramas de clase UML estereotipados para la especificación del modelo independiente de la plataforma (*Platform Independent Models*; *PIM*).
- Diagramas de clase UML no estereotipados para la especificación de los modelos específicos de la plataforma (*Platform Specific Models*; *PSM*).
- SQL para la especificación del código fuente de la base de acontecimientos.

Para la definición del PIM se aplican los patrones de acontecimientos definidos previamente como pueden ser los dos indicados: el patrón *Cambio de Estado* y el patrón *Creación de Objeto*. En particular, durante esta fase se define el *Patrón PIM para el Diseño de Bases de Acontecimientos* (*OcBase PIM Pattern*) que aplica los dos patrones mencionados de forma reiterada para modelar los objetos, protocolos y estados de la base de acontecimientos.

Una vez se ha definido el PIM, este se transforma en modelos más cercanos al sistema de persistencia sobre el que se implantará la base de acontecimientos, obteniendo así los PSM. Para ello, de forma equivalente al Patrón PIM, se define el *Patrón PSM para el Diseño de Bases de Acontecimientos* (*OcBase PSM Pattern*).

Finalmente, a partir del PSM se aplica una nueva transformación para generar el código fuente de la base de acontecimientos. Por ejemplo, si la implantación de la base de acontecimientos se realizara sobre sobre un sistema gestor de bases de datos relacionales, se generaría el conjunto de instrucciones SQL necesarias.

3 Definición de la arquitectura del Sistema de Gestión de Acontecimientos

En [24] se propone una arquitectura basada en el patón Modelo-Vista-Controlador (*Model-View-Controller; MVC*) como puede observarse en la siguiente figura:

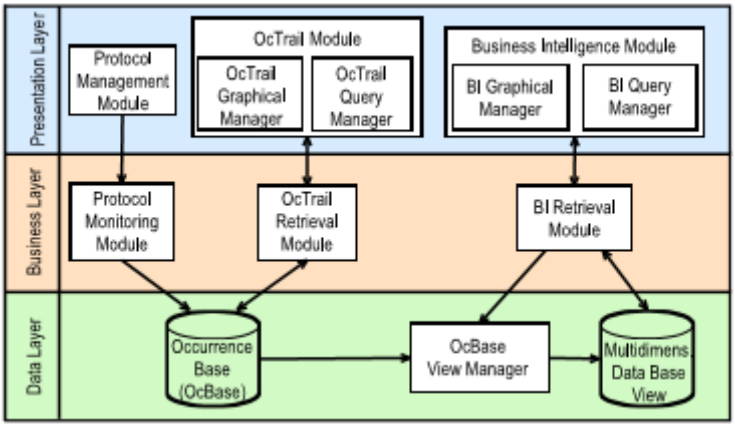


Figura 2.6 Arquitectura de un Sistema de Gestión de Acontecimientos

Cada uno de los componentes que implementan la funcionalidad del Sistema Gestor de Acontecimientos, se ubican en cada una de las capas definidas por este patrón: *Presentation Layer* o Capa de Presentación (Vista; View), *Business Layer* o Capa de Negocio (Controlador; Controller) y *Data Layer* o Capa de Persistencia (Modelo; Model).

Además de agruparse en 'horizontal', a través de las tres capas definidas por el patrón MVC, los componentes que forman parte de esta arquitectura pueden agruparse en 'vertical' en forma de tres subsistemas:

1 Subsistema de gestión y monitorización de protocolos

   A través de este subsistema se registra y monitoriza la ejecución de aquellos protocolos implementados en el sistema que forman parte de los procesos de negocio definidos por una organización para dirigir su trabajo diario. Esta información es almacenada y extraída (monitorización) de la base de acontecimientos.

2 Subsistema de OcTrail

   A través de este subsistema es posible extraer información en relación a los acontecimientos que han ocurrido a un objeto y a todos los objetos relacionados con él; es decir, obtener la historia, la línea de vida, etc.

3 Subsistema de inteligencia de negocio (Business Intelligence; BI)

   A través de este subsistema es posible extraer conocimiento derivado de la ejecución de los procesos de negocio.

Dado que entre los objetivos de esta tesis no se encuentra la construcción de un Sistema de Gestión de Acontecimientos, no consideramos necesario detallar la funcionalidad concreta que ofrece cada uno de los componentes de esta arquitectura.

Un caso concreto: el modelo de base de acontecimientos de Arasis

El sistema informático *Arasis*, que presentaremos un poco más adelante, se enmarca en el dominio de los biobancos y se construyó para dar servicio a "Aragón Workers Health Study" (AWHS), un proyecto dirigido al almacenamiento y estudio de muestras biológicas de los empleados de General Motors España [13].

Siguiendo la estrategia de diseño que se acaba de presentar, en primer lugar, se definió el modelo independiente de la plataforma (PIM) de la base de acontecimientos del sistema *Arasis*; se muestra en la siguiente figura tal y como se presentó en [24]:

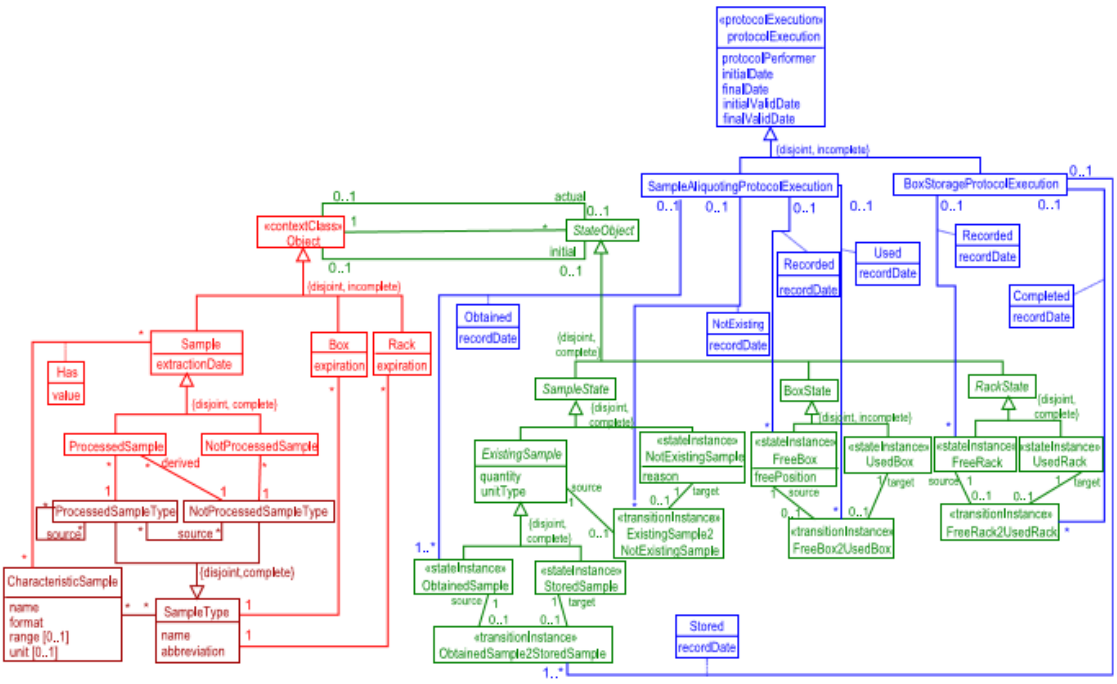


Figura 2.7 Extracto del PIM del modelo de la base de acontecimientos del sistema *Arasis* publicado en [24]

- Las clases *Sample*, *Box* y *Rack* se utilizaron para representar respectivamente a tres de los tipos de objeto contemplados en el sistema: las muestras tratadas en el biobanco, las gradillas en las que se almacenan las muestras, y los racks en los que se almacenan las alícuotas.
- Las clases *SampleAliquotingProtocolExecution* y *BoxStorageProtocolExecution* se utilizaron para representar a dos de los posibles tipos de ejecución de protocolo.
- Los posibles tipos de estado activo que pueden producirse se representaron a través de las clases *ExistingSample*, *ObtainedSample*, *StoredSample* y *NonExistingSample* para los objetos de tipo *Sample*, *FreeBox* y *UsedBox* para los objetos de tipo *Box*, y *FreeRack* y *UsedRack* para los objetos de tipo *Rack*.

  Podemos observar también cómo se representaron cuatro tipos de transición disparada que desencadenan cambios de estado, se hizo a través de las clases identificadas como *ObtainedSample2StoredSample*, *ExistingSample2NonExistingSample*, *FreeBox2UsedBox* y *FreeRack2UsedRack*.

  Por último, los conceptos *Source* y *Target* del metamodelo también fueron contemplados en el modelo representado en la Figura 2.7; por ejemplo, podemos observar en el tipo de

transición *FreeBox2UsedBox*, que al dispararse cambia el estado activo de un objeto de tipo *Box*; la clase *FreeBox* representa al estado origen (source), mientras que la clase *UsedBox* representa al estado destino (target).

Como hemos podido observar, el modelo de una base de acontecimientos debe implementar los diferentes diagramas de estados por los que pueden pasar los diferentes tipos de objeto contemplados en el mismo; por ejemplo, en la siguiente figura podemos observar el diagrama de estados del tipo de objeto *Sample* implementado en el modelo mostrado:

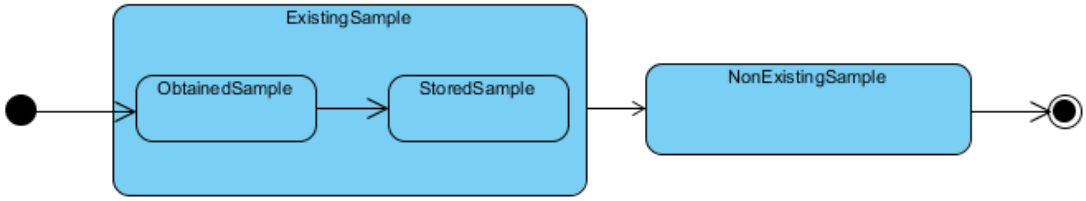


Figura 2.8 Diagrama de estados del tipo de objeto Sample

Sistema de Gestión de Acontecimientos en el ámbito de esta tesis

A través de esta tesis se pretende dar continuidad a las investigaciones publicadas en [24] aportando nuevos elementos que faciliten y normalicen la construcción de sistemas de gestión de acontecimientos.

A través del *Framework OcQF* (*Occurrence Query Framework*; *OcQuery Framework*) del que hablaremos en el siguiente capítulo, se proporcionan componentes y una metodología para dar soporte a los procesos de construcción de sistemas de gestión de acontecimientos.

A través del *Metamodelo de Bases de Acontecimientos* (*Occurrence Base Metamodel*; *OcBase Metamodel*) se normalizarán los conceptos aplicados en la estrategia de diseño durante la definición del perfil y los patrones, afectando de esta forma a la definición de los modelos de bases de acontecimientos que los aplican.

Un Sistema de Gestión de Acontecimientos podría implementar el *Lenguaje de Consultas para la Gestión de Acontecimientos* (*Occurrence Query Language*; *OcQuery Language*; *OcQL*), que presentaremos en el Capítulo 4, el Capítulo 7 y el Capítulo 11, para realizar el tratamiento de los acontecimientos almacenados en la base de acontecimientos así como su recuperación como parte de una estructura longitudinal (historia, línea de vida, etc.).

2.3 Trabajo relacionado

Finalizamos el recorrido por el marco de referencia de esta tesis presentando el trabajo relacionado de otros autores.

Existe mucha literatura sobre la procedencia (*provenance*), la cual ha dado origen a múltiples

estudios sobre la materia [27, 48]. Por ejemplo, como Moreau demostró en sus trabajos [48], desde el inicio del presente milenio, el número de estudios publicados en relación a la procedencia se ha incrementado significativamente. Sin lugar a dudas, una de las causas de este incremento es la transversalidad de esta materia, lo cual se hace evidente en los numerosos y variados usos y tipos de procedencia que existen tal y como se indica en [63]. Por ejemplo, los tipos de procedencia pueden analizarse en función del nivel que ocupen en la estructura de capas del software *(application stack*); se pueden encontrar ejemplos de procedencia a nivel de sistema operativo, a nivel de aplicación o incluso en escenarios de *big data* [12].

Esta diversidad de tipos y usos de la procedencia han motivado la necesidad de definir estándares. Las propuestas más relevantes en este ámbito son *Open Provenance Model* (OPM) [49] y *PROV Data Model* [51]. Si resumimos las diferentes aproximaciones existentes en la literatura, podemos destacar dos modelos de procedencia: la procedencia de los datos (*data provenance*) y la procedencia de los flujos de trabajo (*workflow provenance*). A este respecto, existen opiniones de que ambos modelos representan a las dos caras de la misma moneda, como en [10], donde se reconoce que *"debe existir algún tipo de convergencia entre estos dos modelos"*; en las investigaciones en las que se enmarca esta tesis se considera que esto es así y ambos modelos se consideran como uno solo a través del concepto *Acontecimiento*.

Otra forma de comparar las diferentes aproximaciones existentes en la literatura puede encontrarse en forma de listas en las que se exponen las características deseables que deberían tener los sistemas de procedencia (*provenance systems*). Glavic y Dittrich [31] categorizan los sistemas de procedencia en función de las funcionalidades que soportan. De forma similar, en [21] se presenta una lista de veintiuna recomendaciones que debería tener el sistema de procedencia ideal. Estas recomendaciones se encuentran agrupadas en cinco categorías: (1) Modelización, (2) captura, (3) almacenamiento, (4) seguridad, y (5) consulta y visualización.

Tal y como se ha indicado anteriormente, uno de los modelos básicos de procedencia es la procedencia de los flujos de trabajo (*workflow provenance*), por tanto, es lógico que existan numerosas propuestas en esta área. Por ejemplo, en [39] se propone un sistema de modelado de flujos de trabajo (*workflow modeling system*) para capturar la procedencia de los datos. Otro ejemplo de un sistema que gestiona la procedencia desde la perspectiva de un flujo de trabajo es *Vistrails* [62]. También es importante mencionar el artículo [28], en el cual se analiza el sistema *Swift* desde la perspectiva de OPM [49].

Existen también estudios en los que se aborda la relación entre monitorización de procesos y procedencia. Por ejemplo, el trabajo de Curbera et al. [20] contribuye significativamente a los aspectos que refieren a la monitorización de procesos de negocio utilizando tecnología preparada para tratar la procedencia. Con un objetivo similar en mente, Brauer y Hasselbring [7] proponen capturar la procedencia integrando un framework de monitorización en los sistemas basados en flujos de trabajo. A pesar de estas propuestas, en [22] se afirma que *"la procedencia en los sistemas de monitorización de procesos de negocio ha recibido una atención insignificante a pesar de su relevancia"*. Este estudio también habla de la importancia, desde el punto de vista de la procedencia, del software para la monitorización desarrollado bajo una arquitectura orientada a servicios (*Service Oriented Architecture*; *SOA*). Una aproximación algo diferente se presenta en [15], donde se realiza un análisis retrospectivo de la procedencia, a través de técnicas de minería de datos, empleando un método específico para la representación temporal. El framework que proponemos está diseñado para posibilitar la construcción de sistemas (*Sistemas de Gestión de*

Acontecimientos; *Occurrence Management Systems*; *OcSystems*) en los cuales la monitorización de procesos de negocio está totalmente vinculada a la procedencia.

Finalmente, diversos autores han estudiado qué modelos de datos y lenguajes son los idóneos para gestionar la procedencia. Por ejemplo, Holland et al. [34] hacen énfasis en que, de forma natural, la estructura para dar soporte a la procedencia tiene forma de grafo dirigido, lo cual coincide con la propuesta de OPM [49] y con la visión de Freire et al. [27]. En particular, esto quiere decir que el modelo de bases de datos relacionales no es el ideal para la gestión de la procedencia. Sin embargo, en [34], los autores reconocen que los sistemas de bases de datos relacionales son ampliamente utilizados para gestionar la procedencia al estar fácilmente disponibles para su uso. Esta circunstancia hace que lo apropiado sea establecer procedimientos para gestionar la procedencia de la información sobre sistemas gestores de bases de datos relacionales (*Database Management Systems*; *DBMS*). El estudio [32] sigue esta misma línea de investigación y propone el uso de SQL para consultar de forma eficiente la información sobre la procedencia. Otras aproximaciones definen lenguajes específicos o utilizan lenguajes existentes adaptados en función de la tecnología utilizada. Los lenguajes pSQL [16], TriSQL [70] o ProQL [43] son ejemplos de estas aproximaciones. En este último estudio, junto con el lenguaje, se presenta un proceso para traducir expresiones ProQL a SQL. Esta aproximación es similar a la utilizada en [14], aunque en este caso, el lenguaje de partida es SPARQL. A este respecto, el lenguaje (OcAL) que proponemos posibilita a los usuarios realizar consultas a nivel de instancia, de forma parecida a otros lenguajes con el mismo propósito. Sin embargo, OcAL pretende ir más allá, dado que su versión generalizada (*Generalized OcAL*) también posibilita que los usuarios declaren consultas sobre tipos de objetos e incluso que hagan lo propio con consultas que combinan tanto instancias como tipos. Esta característica es especialmente útil para obtener información aumentada o enriquecida sobre la procedencia. Por otra parte, a diferencia de otras aproximaciones, el framework que proponemos no fija el sistema de persistencia que debe ser utilizado. El responsable de la implementación del framework es libre de elegir el sistema de persistencia que considere más apropiado en cada caso.

2.4 El sistema informático Arasis

A lo largo de este documento de tesis se referirá a un sistema informático, *Arasis*, que ha sido utilizado como una de las referencias fundamentales durante las investigaciones en las que se enmarca la presente tesis.

El sistema informático *Arasis*, enmarcado en el ámbito de los biobancos y de la investigación biomédica, entró en funcionamiento en el año 2009 con el objetivo de dar servicio al proyecto "Aragón Workers Health Study" (AWHS).

El proyecto AWHS es un proyecto de investigación financiado por el CNIC, el Centro Nacional de Investigación Cardiovascular, y el Gobierno de Aragón, que se puso en marcha con el objetivo de identificar aspectos genéticos y de hábitos vitales relacionados con factores de riesgo cardiovascular [13]. AWHS es un estudio de cohorte para el cual se seleccionó como muestra a un amplio grupo de trabajadores de la factoría de General Motors España ubicada en Figueruelas (Zaragoza). Como resultado del proyecto se espera poder proporcionar información novedosa sobre la distribución y trayectorias de los factores de riesgo cardiovascular.

Para comprender el alcance funcional del sistema informático *Arasis*, es importante señalar que uno de los objetivos del proyecto AWHS fue el desarrollo de un biobanco para el almacenamiento de las muestras biológicas (de sangre, plasma, ADN, etc.) extraídas a los sujetos el estudio, los trabajadores de General Motors España. Estas muestras biológicas se procesan en varias alícuotas (*Alicuotar* es el proceso por el cual una muestra biológica se divide en varias partes que se depositan en unos recipientes llamados *alícuotas*), las cuáles son almacenadas en contenedores adecuados, como cajas, racks o congeladores de muestras biológicas en el biobanco.

Para gestionar este trabajo, se desarrollaron varios protocolos de laboratorio que definen las actividades que han de realizarse para la correcta gestión de las muestras biológicas, desde su obtención y almacenamiento en el biobanco, hasta su posterior uso con fines científicos. Entre los tipos de uso se contempló tanto el procesamiento de las muestras para el estudio, como la cesión de muestras a otros laboratorios, o su envío entre las diferentes sedes del biobanco (una ubicada en Madrid y otra en Zaragoza) en las que se replican las muestras almacenadas por seguridad.

El sistema informático *Arasis* fue desarrollado por la empresa Infozara S. L. implementando los diferentes protocolos de laboratorio en forma de protocolos informáticos posibilitando así el registro de la información generada durante las actividades realizadas durante el día a día en los laboratorios. Esta información se almacena en la base de acontecimientos del sistema posibilitando así explotar el conocimiento almacenado en la misma a través de la realización de los procesos de análisis de datos requeridos para el estudio.

II. UN MARCO DE TRABAJO PARA LA GESTIÓN DE ACONTECIMIENTOS

Capítulo 3. El Framework OcQF para la gestión de acontecimientos

3.1 Introducción

Framework (cuya traducción aproximada sería *marco de trabajo*) es un término inglés utilizado para referir a un conjunto estandarizado de conceptos, prácticas y criterios para abordar un determinado tipo de problemática que sirve como referencia, para la resolución de otros problemas similares. Este término es muy utilizado en el mundo del desarrollo de software para referir a un conjunto de artefactos o componentes de software personalizables utilizado como base para el desarrollo de nuevo software; en este ámbito, los objetivos principales que persigue un framework son (1) acelerar el proceso de desarrollo, (2) la reutilización de código y (3) promover buenas prácticas de desarrollo.

En el ámbito de esta tesis, denominamos *Framework OcQF* (*Occurrence Query Framework; OcQuery Framework*) al sistema de conceptos y reglas que permiten representar estructuralmente, en un Universo del Discurso (*Universe of Discourse; UoD*) o dominio, los elementos a utilizar en la gestión de acontecimientos, proporcionando, al mismo tiempo, una metodología de trabajo para dar soporte a los procesos de construcción de *Sistemas de Gestión de Acontecimientos*. Los conceptos se distribuyen en una estructura formada por distintas partes que constituyen su arquitectura.

En este capítulo describiremos en detalle cada uno de los componentes que forman parte del *Framework OcQF*, comenzaremos mostrando la arquitectura del mismo y seguidamente pasaremos a explicar cada uno de ellos. Los componentes del framework que son pilares de esta tesis como el *Metamodelo de Bases de Acontecimientos* y el *Lenguaje de Consultas para la Gestión de Acontecimientos* cuentan con capítulos propios donde se habla de ellos en profundidad.

3.2 Arquitectura del Framework OcQF

La arquitectura del *Framework OcQF* (*Occurrence Query Framework; OcQuery Framework*), publicada en [25], es la que se describe en la siguiente figura:

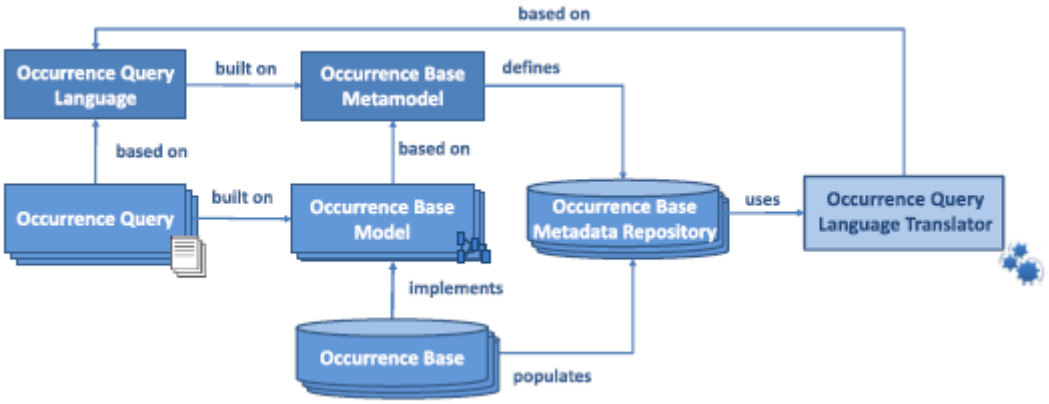


Figura 3.1 Arquitectura del Occurrence Query Framework

Los componentes mostrados en la Figura 3.1 se clasifican en dos grupos según la forma en la que son utilizados al realizar una implementación del framework:

1 Componentes predefinidos

1.1 Lenguaje de Consultas para la Gestión de Acontecimientos (*Occurrence Query Language; OcQL*)

1.2 Metamodelo de Bases de Acontecimientos (*Occurrence Base Metamodel*)

Estos componentes únicamente pueden ser modificados por un meta-modelizador, es decir, por aquellas personas responsables de realizar la especificación de referencia del *Framework OcQF* a partir de la cual se realizarán las diferentes implementaciones del mismo. En este documento de tesis mostraremos dos casos en los que se proponen variaciones en la especificación de referencia, tanto del metamodelo (Capítulo 6 y Capítulo 9), como del lenguaje (Capítulo 7 y Capítulo 11).

2 Componentes configurables

Agrupa a los componentes adaptables a los requisitos de cada implementación del framework:

2.1 Consultas para la Gestión de Acontecimientos (*Occurrence Queries*)

2.2 Modelo de Base de Acontecimientos (*Occurrence Base Model*)

2.3 Repositorio de Metadatos de base de acontecimientos (*Occurrence Base Metadata Repository*)

2.4 Traductor de Consultas OcQL (*Occurrence Query Language Translator*)

2.5 Base de Acontecimientos (*Occurrence Base*)

A continuación, vamos a realizar una presentación de las distintas partes y sus relaciones, las más relevantes para los objetivos de la tesis serán descritas de forma más detallada en los siguientes capítulos.

3.2.1 Metamodelo de Bases de Acontecimientos

Denominamos *Metamodelo de Bases de Acontecimientos* (*Occurrence Base Metamodel*;

OcBase Metamodel) al conjunto de conceptos que pueden ser utilizados en la especificación y explotación de una base de acontecimientos.

En base a estos conceptos se construye el *Lenguaje de Consultas para la Gestión de Acontecimientos* (*Occurrence Query Language*; *OcQL*) debiéndose construir un repositorio de los metadatos (*Occurrence Base Metadata Repository*) o *Diccionario* que describe la estructura de una base de acontecimientos concreta (*Occurrence Base*).

Como ejemplo, en el siguiente diagrama de clases representamos los conceptos que componen un metamodelo muy sencillo:

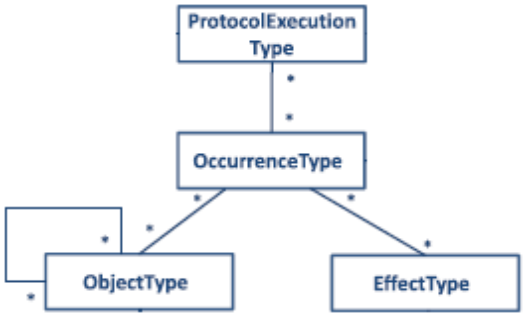


Figura 3.2 Caso de especificación de Metamodelo de Bases de Acontecimientos

En el Capítulo 7 y el Capítulo 11 presentaremos dos especificaciones del *Metamodelo de Bases de Acontecimientos* y cada uno de los conceptos contemplados en las mismas, por lo que no nos extenderemos más en este apartado.

3.2.2 Lenguaje de Consultas para la Gestión de Acontecimientos

El *Lenguaje de Consultas para la Gestión de Acontecimientos* (*Ocurrence Query Language*, *OcQuery Language*; *OcQL*) posibilita el tratamiento de la información almacenada en una base de acontecimientos (*Occurrence Base*; *OcBase*) posibilitando así el análisis de su procedencia.

Por lo general, los lenguajes de consultas para la gestión de información almacenada en sistemas de persistencia (como por ejemplo sistemas basados en bases de datos, estructuras basadas en documentos XML, etc.) requieren, además de indicar la información que debe procesarse, especificar la localización exacta de la misma (por ejemplo, a través de consultas XPath -XML- o de cláusulas FROM en consultas SQL -bases de datos relacionales-) dentro de las estructuras (tabla, nodo, etc.) del sistema de persistencia. Dicho de otro modo, dado que el sistema no es capaz de indicar la localización de la información, es el usuario final el que debe disponer del conocimiento necesario para indicarla al construir la consulta. En el lenguaje que se propone (*OcQL*), el usuario final no indica dónde está localizada la información, sino que es el sistema el que sabe localizarla, liberando así de trabajo al usuario final y no exigiéndole conocimientos de la estructura de persistencia correspondiente.

El lenguaje propuesto posibilita la declaración de consultas tecnológicamente independientes del sistema de persistencia. Esta independencia tecnológica posibilita que las consultas OcQL puedan ser traducidas a diferentes lenguajes de consulta como SQL, OQL o XQuery, lo cual es posible gracias a la existencia de un proceso de traducción que transforma las consultas OcQL

en consultas interpretables por el sistema de persistencia (por ejemplo, en consultas SQL).

El *Lenguaje de Consultas para la Gestión de Acontecimientos* cuenta con tres capítulos propios en este documento de tesis, el Capítulo 4, el Capítulo 7 y el Capítulo 11, en los que hablaremos de él en profundidad. Del funcionamiento del proceso de traducción vamos a hablar en el siguiente apartado.

3.2.3 Traductor de Consultas OcQL

El *Traductor de Consultas OcQL* (*Occurrence Query Language Translator*; *OcQL Translator*), como su nombre indica, es el responsable de traducir automáticamente cada consulta OcQL en una consulta compatible con el sistema de persistencia en el que se encuentra implantada una base de acontecimientos. Por ejemplo, si el sistema de persistencia correspondiera con un gestor de bases de datos relacionales (SGBD), el traductor generaría consultas SQL a partir de las consultas OcQL proporcionadas.

Durante proceso que describiremos en detalle más adelante, cada consulta OcQL (*Occurrence Query*) es procesada por el traductor (*Occurrence Query Language Translator*) utilizando los datos almacenados en el repositorio de metadatos (*Occurrence Base Metadata Repository*). El traductor, aplicando un algoritmo de traducción, genera automáticamente una consulta compatible con el sistema de persistencia (*Persistence System Supported Query*) sobre el que esté implantada la base de acontecimientos (*Occurrence Base*). La ejecución de la consulta resultante deriva en la extracción de la información deseada (*Provenance Information*).

El *Repositorio de Metadatos* es un pilar fundamental durante el proceso de traducción dado que, tal y como veremos más adelante, contiene información descriptiva de la base de acontecimientos (*Occurrence Base*) que posibilita que la traducción sea posible.

En el Capítulo 5 hablaremos con mayor profundidad tanto del traductor como del proceso de traducción.

3.2.4 Modelo de Base de Acontecimientos

Un modelo para la construcción de una base de acontecimientos o *Modelo de Base de Acontecimientos* (*Occurrence Base Model*) es el resultado de utilizar el *Metamodelo de Bases de Acontecimientos* para definir una estructura específica que modela un determinado *Universo del Discurso* (*Universe of Discourse; UoD*) o *dominio* siguiendo una estrategia de diseño.

Esto quiere decir que, en función del alcance que se quiera lograr y de la complejidad del sistema a desarrollar, el *Framework OcQF* posibilita que no recojan la totalidad de los conceptos definidos en el metamodelo. Por ejemplo, como veremos más adelante, uno de los conceptos que podrían recogerse en el metamodelo es *Tipo de Ejecutor*; sin embargo, es posible definir un modelo de base de acontecimientos siguiendo una estrategia de diseño que no recoja este concepto. Evidentemente, este tipo de decisiones tienen implicaciones durante la implementación del framework; por ejemplo, si el concepto *Tipo de Ejecutor* del metamodelo no es recogido en un modelo, no será posible declarar ni traducir consultas OcQL que refieran a dicho concepto y tampoco será posible registrar este tipo de información en las bases de acontecimientos resultantes de implementar el modelo.

La que acabamos de describir es uno de los aspectos en relación al metamodelo que hay que

tener presentes al definir un modelo de base de acontecimientos y que vienen determinados por la estrategia de diseño seleccionada tal y como hemos podido ver en el apartado 2.2.

Tal y como describiremos en el siguiente apartado, la implementación de un modelo de base de acontecimientos como resultado una base de acontecimientos.

Una vez hemos introducido el *Metamodelo de Bases de Acontecimientos*, el *Modelo de Base de Acontecimientos* e introducido el concepto *Base de Acontecimientos*, antes de continuar, presentamos en la siguiente figura las relaciones entre ellos:

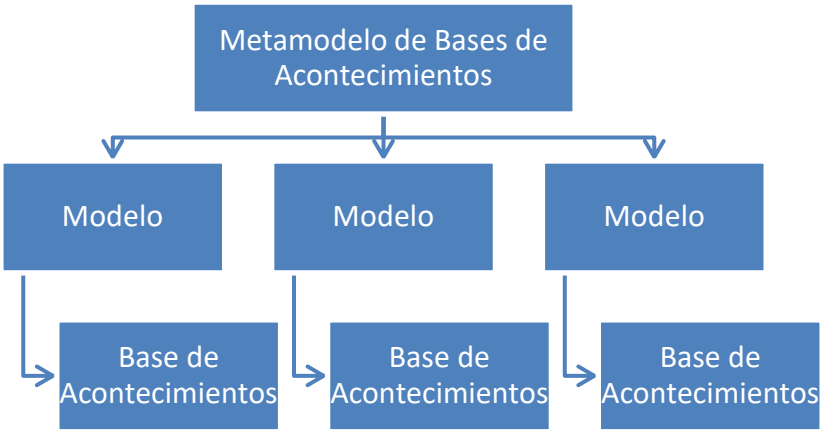


Figura 3.3 Relación entre metamodelo, modelo y base de acontecimientos

En la Figura 3.3 podemos observar como a partir de un único *Metamodelo de Bases de Acontecimientos* pueden definirse múltiples *Modelos de Base de Acontecimientos* representando diferentes dominios o universos del discurso (UoD), y cada uno de estos últimos, implementarlos en una *Base de Acontecimientos*.

3.2.5 Base de Acontecimientos

Una *Base de Acontecimientos* (*Occurrence Base*; *OcBase*) es una estructura de almacenamiento que cumple un doble cometido:

1 Facilitar la realización de estudios longitudinales a través de la explotación de la información en forma de estructuras longitudinales más simples como la *Historia* o la *Línea de Vida*.

2 Facilitar la obtención de la procedencia (*provenance*) de la información.

Puede decirse también que una *Base de Acontecimientos* es una estructura de persistencia que registra acontecimientos sucedidos a lo largo del tiempo que nace como resultado de la implementación concreta de un determinado modelo de base de acontecimientos (*Occurrence Base Model*) sobre un sistema para la persistencia de la información como puede ser, por ejemplo, un sistema gestor de bases de datos relacionales (*Relational Database Management System; RDBMS*).

3.2.6 Repositorio de Metadatos de Base de Acontecimientos

Llamamos *Repositorio de Metadatos de Base de Acontecimientos* (*Occurrence Base Metadata Repository*; *OcBase Metadata Repository*) al componente del framework que posibilita almacenar el conocimiento expresado en los modelos de base de acontecimientos, así como el de las instancias de los mismos en forma de bases de acontecimientos. Es decir, el *Repositorio de Metadatos* almacena información que describe la estructura de una base de acontecimientos, de los elementos que la componen (por ejemplo, tablas, campos, vistas, etc. en el caso de una base de datos relacional) y de las relaciones entre ellos. El *Repositorio de Metadatos* también puede ser referido simplemente como *Diccionario*.

La idea de repositorio de metadatos que proponemos se basa en el concepto *Diccionario de Datos* o *Catálogo de Datos* que, en el *Dictionary of IBM & Computing Terminology* [36] se define de la siguiente manera:

> *"A centralized repository of information about data such as meaning, relationships to other data, origin, usage, and format. It assists management, database administrators, system analysts, and application programmers in planning, controlling, and evaluating the collection, storage, and use of data."*

Este concepto también es definido por Oracle [56] de la siguiente manera:

> *"(…) a read-only set of tables that provides information about the database."*

Y también es contemplado en SQL de Server de Microsoft siendo denominado en este caso *SQL Server System Catalog* [47].

Al realizar una implementación del *Framework OcQF*, la implementación del *Repositorio de Metadatos* no tiene por qué realizarse sobre el mismo tipo de sistema de persistencia sobre el que se implante la *Base de Acontecimientos*. Aunque tal vez esta sería la decisión natural, el framework posibilita que esta elección pueda realizarse libremente.

Por otra parte, al realizar una implementación del *Repositorio de Metadatos*, debe tenerse en cuenta que su esquema debe recoger los conceptos incluidos en el *Metamodelo de Bases de Acontecimientos*. Del mismo modo, al poblar de información dicho esquema, debe tenerse en cuenta que la información que lo ha de poblar es dependiente de la *Base de Acontecimientos* a la que describe.

El *Repositorio de Metadatos* es un pilar fundamental en el proceso de traducción dado que la información contenida en el *Repositorio de Metadatos* es utilizada por el *Traductor de consultas OcQL* para conocer la estructura de la *Base de Acontecimientos* y poder así generar las consultas compatibles con el sistema de persistencia a partir de las correspondientes consultas OcQL.

A lo largo de los siguientes capítulos de este documento de tesis se irá refiriendo según corresponda al *Repositorio de Metadatos* y se irán viendo ejemplos de la forma en la que se utiliza y su rol durante el proceso de traducción.

3.2.7 Consultas OcQL

Como se ha indicado al inicio de este capítulo, el *Lenguaje de Consultas para la Gestión de Acontecimientos* (*Occurrence Query Language*; *OcQL*) es uno de los dos componentes del *Framework OcQF* que no puede modificarse cuando se realiza una implementación del mismo. No obstante, esta afirmación no implica que OcQL no sea adaptable, al contrario, el lenguaje

deja abiertas las siguientes dimensiones del mismo para que puedan ser definidas en función de la estrategia de diseño seleccionada:

1 Los tipos de acontecimientos contemplados.

2 Los tipos de estructuras longitudinales utilizadas.

3 Los tipos de elementos de los acontecimientos que pueden ser seleccionados en las consultas sobre estructuras longitudinales.

4 Los tipos de elementos de los acontecimientos que pueden ser seleccionados en las consultas sobre acontecimientos.

De estas decisiones dependen las *Consultas OcQL* (que forman parte del componente *Occurrence Query* representado en la Figura 3.1) que pueden definirse e interpretarse por el traductor.

Por ejemplo, adelantando algunos conceptos que presentaremos más adelante, no sería posible traducir consultas OcQL que hicieran referencia a conceptos como *Tipo de Efecto* (EffectType) o *Tipo de Ejecución de Protocolo* (ProtocolExecutionType) si estos no son contemplados en el metamodelo (ver Figura 3.2), considerados por la estrategia de diseño seleccionada, implementados en el modelo de base de acontecimientos derivado de la misma y, finalmente, contemplados en el traductor.

3.3 Etapas de la implementación del framework

Una vez que hemos presentado los componentes que forman parte del *Framework OcQF* vamos a describir las etapas que posibilitan realizar una implementación del mismo.

Tal y como hemos indicado al inicio de este capítulo, los componentes del *Framework OcQF* se clasifican en dos grupos según vengan predefinidos por los meta-modelizadores o sean adaptables a los requisitos de cada una de las implementaciones que se realicen del framework. Recordemos que los componentes predefinidos son (1) el *Metamodelo de Bases de Acontecimientos* y (2) el *Lenguaje de Consultas para la Gestión de Acontecimientos* (OcQL), mientras que los componentes adaptables son (1) las *Consultas para la Gestión de Acontecimientos*, (2) el *Modelo de Base de Acontecimientos*, (3) el *Repositorio de Metadatos*, (4) el *Traductor* del lenguaje de acontecimientos y (5) la *Base de Acontecimientos*.

En el escenario que vamos a describir, reflejamos el punto de vista de los responsables de realizar una implementación del framework; es decir, partimos de la situación en la que las especificaciones de referencia del metamodelo y del lenguaje han sido realizadas por los meta-modelizadores y por lo tanto vienen dados para toda implementación que se realice a partir las mismas.

Los componentes predefinidos por los meta-modelizadores actúan como marco de referencia para la particularización de los componentes adaptables obteniéndose, como resultado de la misma, una implementación concreta del framework.

Por otra parte, tal y como se ha podido observar en la arquitectura mostrada en la Figura 3.1, todos los componentes del framework se encuentran relacionados de la siguiente manera:

1 El lenguaje de consultas se basa en los conceptos descritos en el metamodelo.

2 La totalidad o parte de los conceptos del metamodelo son recogidos en la estrategia de

diseño elegida para definir los modelos de base de acontecimientos.

3 Las consultas para la gestión de acontecimientos (consultas OcQL) quedan delimitadas por el conjunto de conceptos del metamodelo recogido en la estrategia de diseño.

4 La base de acontecimientos corresponde con una implementación del modelo y sus estructuras se describen en el repositorio de metadatos.

5 El traductor accede a la información que describe a la base de acontecimientos para realizar la traducción de las consultas OcQL a un lenguaje soportado por el sistema de persistencia seleccionado para la implantación de la base de acontecimientos.

Este apartado vamos a presentar de forma teórica cómo se particularizan estos componentes y cómo entran en juego sus relaciones durante la realización de una implementación del *Framework OcQF*; estos conceptos, y los que se irán presentando a lo largo de este documento de tesis, serán aplicados en el caso práctico que se presentará en el Capítulo 8 que describe una implementación concreta del framework probada sobre la base de acontecimientos del sistema informático *Arasis*.

Antes de proseguir, es necesario describir a los participantes en los procesos de diseño y desarrollo. Consideramos que en estos procesos de la implementación del framework intervienen como mínimo participantes que pueden tomar alguno de estos dos roles:

1 Desarrollador

Los participantes con este rol son responsables de la configuración, preparación y desarrollo de los componentes adaptables del framework.

2 Modelador

Los participantes con este rol son responsables de la definición del modelo de datos.

Como puede observarse, a efectos de no añadir complejidad a la explicación y lograr centrar los aspectos importantes, hemos simplificado los roles que de forma natural intervienen en los procesos de construcción de software.

A continuación, pasamos a describir en qué consiste cada una de las etapas para realizar una implementación del *Framework OcQF* y la forma en la que intervienen cada uno de los tipos de participante.

3.3.1 Primera etapa: Desarrollo del software

Los desarrolladores son los encargados de llevar a buen término la primera etapa de la implementación del framework, durante la misma, su principal responsabilidad es desarrollar el traductor de consultas cuya función, recordemos, es transformar cada consulta OcQL de entrada para devolver una consulta equivalente definida en el lenguaje de persistencia seleccionado, por ejemplo, SQL.

Para realizar la implementación del traductor, los desarrolladores deben tener en cuenta y conocer lo siguiente:

1 La estrategia de diseño orientada a acontecimientos seleccionada para el desarrollo de la base de acontecimientos.

2 El sistema de persistencia sobre el que se implantará la base de acontecimientos.

3 El lenguaje de consultas del sistema de persistencia.

4 La sintaxis y semántica de OcQL.

5 La forma en que se va a implementar el repositorio de metadatos.

Así pues, es fundamental que la estrategia de diseño, el sistema de persistencia y el repositorio de metadatos sean conocidos antes de comenzar el proceso de desarrollo del traductor.

El sistema de persistencia

La elección del sistema de persistencia es clave dado que de esta decisión depende el lenguaje al que serán traducidas las consultas OcQL proporcionadas al traductor y las características del entorno sobre el que se implantará la base de acontecimientos.

Recordemos que el sistema de persistencia no tiene porqué limitarse a un sistema gestor de bases de datos relacionales y al lenguaje SQL, sino que, en función de las necesidades, es posible optar por ejemplo por sistemas basados en documentos XML y el lenguaje XPath.

La estrategia de diseño

Si bien es posible optar por diferentes estrategias de diseño, en el ámbito de la investigación en la que se enmarca esta tesis, y especialmente para dar forma a la prueba de concepto que presentaremos en el Capítulo 8, se ha seleccionado la estrategia de diseño orientada a acontecimientos propuesta en [24] y descrita en el apartado 2.2. A esta estrategia de diseño, utilizada para la definición de modelos de bases de acontecimientos, la hemos denominado *OcBase Model Driven Design Strategy* (*OcBaseMD Design Strategy*).

Tal y como hemos explicado en el apartado 2.2, esta estrategia de diseño está basada en la metodología conocida como *Ingeniería Dirigida por Modelos* o *Model Driven Engineering* (*MDE*) descrita en [5], y propone partir de un modelo independiente de la plataforma (PIM) en la que se implantará el sistema y transformarlo en un modelo específico de la plataforma (PSM) para generar la especificación del código fuente (en este caso el necesario para la implantación de la base de acontecimientos).

El repositorio de metadatos

Para desarrollar el traductor se debe conocer información específica sobre las estructuras de almacenamiento de la base de acontecimientos y sobre las relaciones entre ellas; a esta información descriptiva se le conoce usualmente como *metadatos*.

El *Framework OcQF* cuenta con el componente denominado *Repositorio de Metadatos* (*Occurrence Base Metadata Repository*; *OcBase Metadata Repository*) que hemos presentado en el apartado 3.2.6. El *Repositorio de Metadatos* es el componente del framework en el que se registra la información sobre las estructuras de almacenamiento de la base de acontecimientos y sobre las relaciones entre ellas.

Durante esta etapa, y como paso previo al desarrollo del traductor, los desarrolladores deben definir e implementar el esquema del repositorio de metadatos de acuerdo con la implementación realizada del metamodelo. En relación a esto, es importante recordar que, aunque lo natural es que el repositorio de metadatos se implante sobre el mismo tipo de sistema de persistencia que la base de acontecimientos, el framework posibilita que esta elección pueda realizarse libremente.

3.3.2 Segunda etapa: Modelización de un contexto

La segunda etapa se inicia una vez que los desarrolladores han realizado la implementación del traductor, y con ella comienza la participación de los modeladores.

Para cada universo del discurso o dominio específico, los modeladores deben crear el correspondiente modelo de base de acontecimientos, y deben hacerlo de acuerdo con el metamodelo y siguiendo la estrategia de diseño seleccionada. El modelo de base de acontecimientos será implementado en función del sistema de persistencia seleccionado dando como resultado una base de acontecimientos.

El siguiente paso, una vez implementada la base de acontecimientos, es poblar el repositorio de metadatos con la información concreta sobre las estructuras de almacenamiento de la base de acontecimientos y sus dependencias de acuerdo al esquema implementado por los desarrolladores. En la siguiente figura podemos ver, como ejemplo, el esquema de un repositorio de metadatos basado en el metamodelo representado en la Figura 3.2, poblado con información descriptiva de la base de acontecimientos de un biobanco:

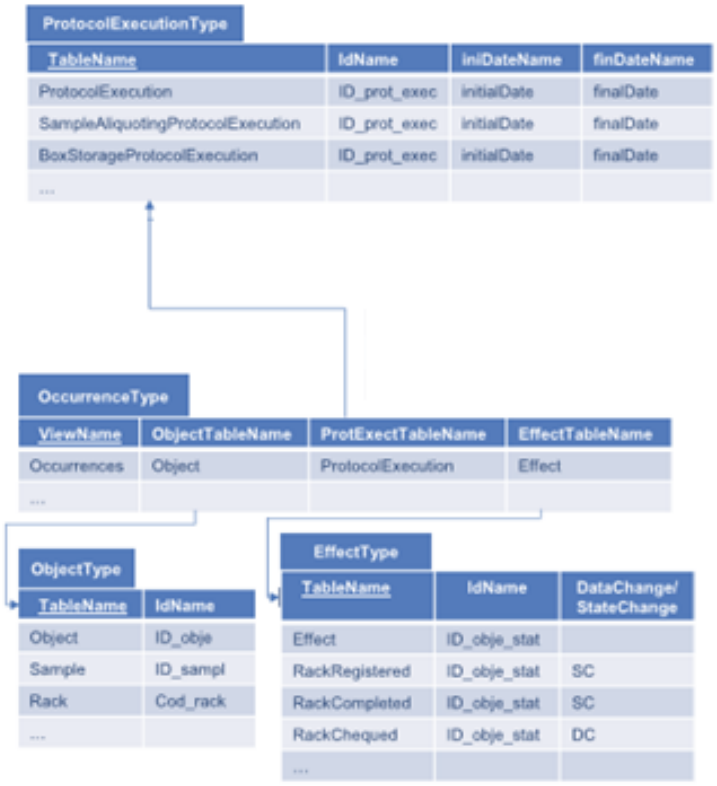


Figura 3.4 Caso de implementación del Repositorio de Metadatos

3.3.3 Tercera etapa: Explotación

Una vez que los desarrolladores y los modeladores han implementado los diferentes artefactos, se inicia la etapa de explotación.

La explotación se realiza directamente por parte de los usuarios finales que, en función de sus necesidades, simplemente deben definir las consultas OcQL para procesar la información de la base de acontecimientos.

Por ejemplo, el usuario final de una implementación del traductor podría ser un técnico de laboratorio que desea consultar información almacenada en la base de datos que da servicio al biobanco en el que trabaja; en este caso, el usuario podría estar interesado en obtener distintas informaciones sobre la procedencia de los datos como pueden ser: (1) La línea de vida de las muestras biológicas extraídas a un determinado sujeto de estudio, (2) la historia del almacenamiento de las alícuotas obtenidas de las muestras, o (3) los acontecimientos relacionados con un determinado protocolo de procesamiento de muestras.

3.4 Resumen de conceptos

Hasta el momento, en el presente documento de tesis hemos presentado o referido a los múltiples conceptos que se han tenido en cuenta durante las investigaciones realizadas y hemos tratado de describir la forma en la que se relacionan y el efecto que uno tiene sobre los otros. Antes de continuar, en este apartado vamos a centrar y resumir todo este conocimiento.

Los conceptos fundamentales que, hasta el momento, hemos presentado o referido en este documento pueden verse representados en el siguiente diagrama de conceptos:

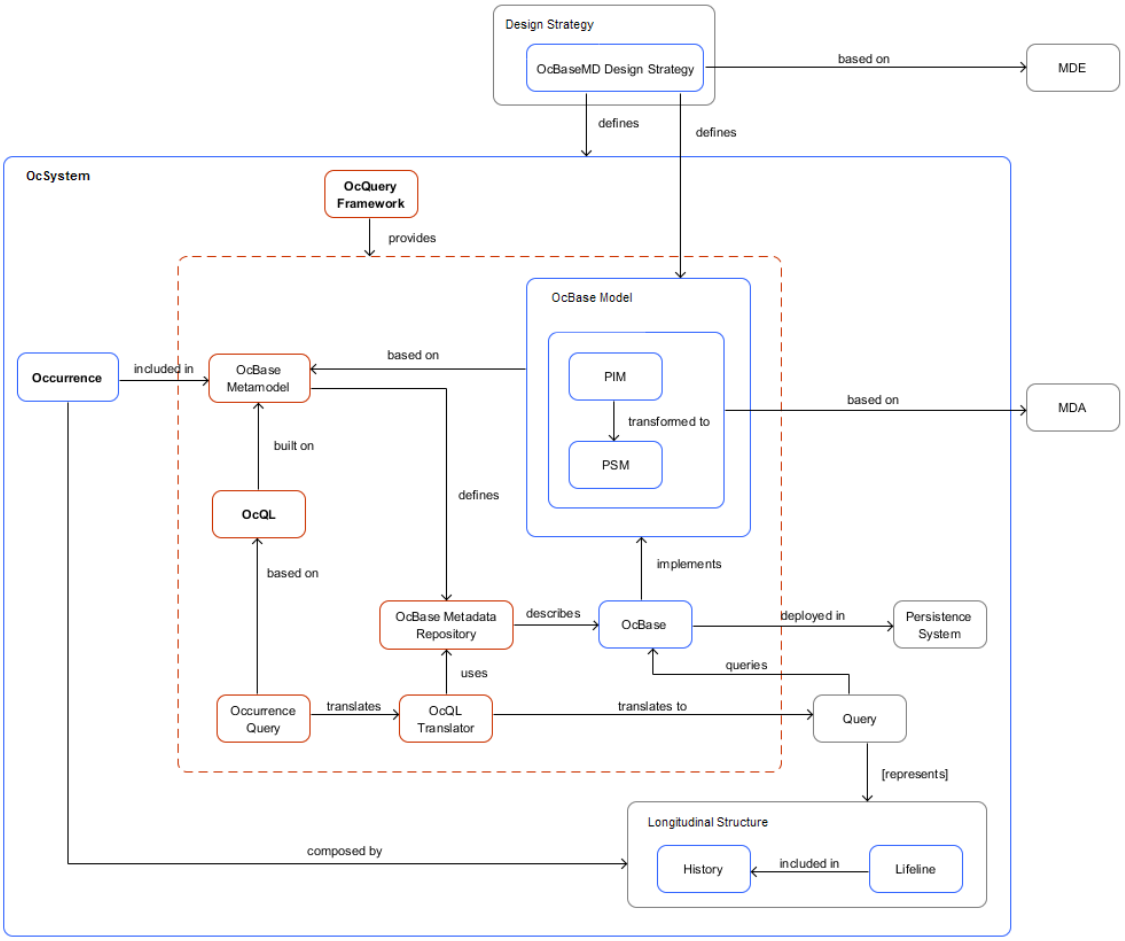


Figura 3.5 Diagrama de conceptos presentados o referidos en la tesis

Para mayor claridad, los conceptos de esta figura se han representado siguiendo el siguiente código de color: (1) Rojo para señalar los conceptos que han sido desarrollados como parte de esta tesis, (2) azul para señalar los conceptos que han sido recogidos de las investigaciones del *Grupo Nóesis de la Universidad de Zaragoza*, y (3) gris para señalar conceptos existentes que han sido aplicados de forma directa o indirecta en la tesis. En el caso de los conceptos coloreados en azul, es importante señalar que bien pueden ser propios, bien haber sido adecuados en las investigaciones correspondientes tomando como referencia conceptos existentes.

En la Figura 3.5 podemos observar en un primer nivel dos conceptos fundamentales: (1) El *Framework OcQF* (*Occurrence Query Framework*; *OcQuery Framework*), y (2) el *Sistema de Gestión de Acontecimientos* (*Occurrence Management System*; *OcSystem*).

El *Framework OcQF* proporciona un conjunto de componentes y una metodología para dar soporte a los procesos de construcción de *Sistemas de Gestión de Acontecimientos*.

Los principales componentes que proporciona el *Framework OcQF* son los siguientes: (1) El *Metamodelo de Bases de Acontecimientos* (*Occurrence Base Metamodel*; *OcBase Metamodel*), (2) el *Lenguaje de Consultas Para la Gestión de Acontecimientos* (*Occurrence Query Language*; *OcQL*), (3) el *Modelo de Base de Acontecimientos* (*Occurrence Base Model*; *OcBase Model*), (4) la *Base de Acontecimientos* (*Occurrence Base*; *OcBase*), (5) el *Traductor de*

Consultas OcQL (Occurrence Query Language Translator; OcQL Translator), y (6) el *Repositorio de Metadatos de Bases de Acontecimientos* (*Occurrence Base Metadata Repository*; *OcBase Metadata Repository*).

Pero el concepto fundamental en el que se apoyan el resto es *Acontecimiento* (*Occurrence*). Este concepto es contemplado en el *Metamodelo de Bases de Acontecimientos* y es aplicado en la definición de *Modelos de Bases de Acontecimientos* y en las *Bases de Acontecimientos* resultantes de la implementación de los modelos. Es más, el metamodelo contempla no sólo este concepto, sino que lo complementa con un conjunto de conceptos que lo definen y que son esenciales para dar forma a la aproximación basada en acontecimientos que es fundamental en las investigaciones realizadas.

El metamodelo y el lenguaje dependen de la especificación de referencia que realicen los meta-modelizadores, los restantes componentes son personalizables en cada implementación realizada del *Framework OcQF*. Esto quiere decir, por ejemplo, que es posible definir un *Modelo de Base de Acontecimientos* que no contemple algunos de los conceptos del metamodelo.

La especificación de un *Sistema de Gestión de Acontecimientos* se realiza siguiendo una *Estrategia de Diseño* (*Design Strategy*) basada en acontecimientos [24] en función de la cual es posible definir un *Modelo de Base de Acontecimientos*. El modelo resultante depende de la estrategia de diseño seleccionada en la que se determinarán los conceptos del metamodelo que se usarán y de qué modo. Aunque es posible optar por diferentes estrategias de diseño, en las investigaciones realizadas hemos aplicado una estrategia (*OcBase Model Driven Design Strategy*; *OcBaseMD Design Strategy*) que sigue la metodología conocida como *Ingeniería Dirigida por Modelos* (*Model Driven Engineering*; *MDE*) y usa la metodología *MDA* o *Arquitectura Dirigida por Modelos* (*Model Driven Architecture*), lo que quiere decir que la especificación de un *Sistema de Gestión de Acontecimientos* se realiza mediante la definición de modelos. Concretamente, aplicando esta estrategia de diseño, una *Base de Acontecimientos* es el resultado de definir un *Modelo Independiente de la Plataforma* (*Platform Independent Model*; *PIM*), que posteriormente es transformado en un *Modelo Específico de la Plataforma* (*Platform Specific Model*; *PSM*) a partir del cual se genera el código fuente de la *Base de Acontecimientos*. En este contexto, denominamos *Plataforma* al *Sistema de Persistencia* (Persistence System) sobre el que se encuentra implantada una *Base de Acontecimientos*. Para lograr definir estos modelos, como parte de la *Estrategia de Diseño*, es necesario haber definido previamente una serie de *Perfiles* y *Patrones*; en esta etapa, los conceptos del metamodelo que sean contemplados como parte de la *Estrategia de Diseño* afectará a cómo sean definidos dichos perfiles y patrones.

Con la introducción del *Metamodelo de Bases de Acontecimientos* se logra normalizar el conjunto de conceptos que puede utilizar una determinada *Estrategia de Diseño* para la definición de *Modelos de Base de Acontecimientos*.

El *Lenguaje de Consultas Para la Gestión de Acontecimientos* (*Occurrence Query Language*; *OcQL*) posibilita el tratamiento de la información almacenada en una base de acontecimientos (*Occurrence Base*; *OcBase*), para que esto sea posible, una *Consulta OcQL* (*Occurrence Query*) es procesada por el Traductor (Occurrence Query Language Translator; *OcQL Translator*) y traducida a una consulta (*Query*) compatible con el *Sistema de Persistencia* en el que se encuentra implantada la *Base de Acontecimientos*. El proceso de traducción de consultas OcQL se apoya en el *Repositorio de Metadatos* (*Occurrence Base Metadata Repository*; *OcBase Metadata Repository*), el cual contiene información descriptiva de la *Base de Acontecimientos*. Una vez traducidas, las consultas son ejecutadas sobre la *Base de*

Acontecimientos para realizar la correspondiente operación de tratamiento de la información. En el caso de las consultas de acceso a acontecimientos, su ejecución posibilita obtener acontecimientos y elementos de los mismos, información de acontecimientos y de sus elementos, así como acontecimientos formando parte de una *Estructura Longitudinal (Longitudinal Structure*), como por ejemplo la *Historia* (*History*) o la *Línea de Vida* (*Lifeline*).

En los siguientes capítulos de este documento de tesis profundizaremos, desarrollaremos o referiremos a los conceptos que acabamos de explicar.

Capítulo 4. El Lenguaje de Consultas para la Gestión de Acontecimientos (OcQL)

4.1 Introducción

El principal objetivo de las investigaciones desarrolladas y de la tesis que se presenta en este documento ha sido, junto con la especificación del *Framework OcQF*, la definición de un lenguaje de consultas para el tratamiento de la información almacenada en bases de acontecimientos (*Occurrence Base – OcBase*). A este lenguaje se le ha denominado *Lenguaje de Consultas para la Gestión de Acontecimientos* (*Ocurrence Query Language*; *OcQuery Language*; *OcQL*) siendo una parte del mismo, junto con el *Framework OcQF* y el metamodelo, objeto de la siguiente publicación [25]:

> *Domínguez, E., Pérez, B., Rubio, A. L., Zapata, M. A., Allué, A., López, A. (2017) Developing provenance-aware query systems: an occurrence-centric approach. Knowledge and Information Systems 50, pp 661–688, DOI 10.1007/s10115-016-0950-z.*

Este capítulo está dedicado a presentar las características generales del lenguaje.

4.2 Características del lenguaje

El *Lenguaje de Consultas para la Gestión de Acontecimientos* (OcQL) se caracteriza por lo siguiente:

1 Independencia de la tecnología

2 Simplicidad

3 Adaptabilidad

4 Independencia del software

Independencia de la tecnología

Las consultas OcQL son tecnológicamente independientes del sistema de persistencia; esto quiere decir que a través de ellas es posible obtener información registrada en una base de acontecimientos independientemente del sistema de persistencia en el que se encuentre implantada y, por otra parte, que dichas consultas serían válidas, aunque en un momento dado dicho sistema de persistencia cambiara.

La mencionada independencia tecnológica se logra gracias a la existencia de un proceso de traducción que transforma las consultas OcQL en consultas declaradas en un lenguaje

compatible con el sistema de persistencia en el que se encuentre implantada la base de acontecimientos correspondiente (por ejemplo, en consultas SQL, OQL, XQuery, etc.). En el Capítulo 5 se explicará en qué consiste el proceso de traducción.

Simplicidad

Por lo general, los lenguajes de consultas para la gestión de información almacenada en sistemas de persistencia (como sistemas gestores de bases de datos relacionales -RDBMS- o estructuras basadas en documentos XML) requieren, además de indicar la información que debe procesarse, especificar la localización de la misma (por ejemplo, a través de consultas XPath -XML- o de cláusulas FROM en consultas SQL -bases de datos-) dentro de las estructuras (tabla, nodo, etc.) del sistema de persistencia. Dicho de otro modo, dado que el sistema no es capaz de indicar la localización de la información, es el usuario el que debe disponer del conocimiento necesario para especificarla al construir la consulta.

Las consultas OcQL se declaran especificando únicamente la información que se desea extraer, evitando referir a su localización, y utilizando conceptos propios de la *Procedencia* (*Provenance*) como *Acontecimiento*, *Historia* o *Línea de vida*.

Dadas las características de las bases de acontecimientos, las consultas OcQL evitan el tener que escribir consultas complejas en el lenguaje soportado por el sistema de persistencia en el que se encontrara implantada la correspondiente base de acontecimientos. En el Capítulo 8 mostraremos varios casos en los que se puede observar como a través de una consulta OcQL de no más de cinco líneas se puede lograr el mismo resultado que a través de una consulta SQL equivalente de varias decenas de líneas.

Adaptabilidad

OcQL puede ser adaptado a diferentes estrategias de diseño basadas en acontecimientos. Varias de las dimensiones del lenguaje están abiertas para ser definidas en función de la estrategia de diseño elegida para desarrollar las bases de acontecimientos, concretamente son las siguientes:

1 Los tipos de acontecimientos contemplados.

2 Los tipos de estructuras longitudinales utilizadas.

3 Los tipos de elementos de los acontecimientos que pueden ser seleccionados en las consultas sobre estructuras longitudinales.

4 Los tipos de elementos de los acontecimientos que pueden ser seleccionados en las consultas sobre acontecimientos.

Por ejemplo, haciendo uso de la estrategia de diseño descrita en el apartado 2.2, para la prueba de concepto que presentaremos en el Capítulo 8, las dimensiones del lenguaje se definieron de la siguiente manera:

1 Se consideró un tipo de acontecimiento.

2 Los tipos de estructura longitudinal contemplados fueron *Historia* y *Línea de Vida*.

3 El tipo de elemento considerado en las consultas sobre estructuras longitudinales fue *Objeto*, de tal forma que los tipos de estructura longitudinal contemplados (*Historia* y *Línea de Vida*) hicieran referencia a acontecimientos ocurridos a este tipo de elementos.

4 Los tipos de elementos considerados en las consultas sobre acontecimientos fueron objetos, ejecuciones de protocolo, estados activos, efectos y ejecutores.

Independencia del software

Hay cambios que no afectan a OcQL como la evolución del software o de los sistemas de persistencia. Tal y como se indica en [65], en la mayoría de las aplicaciones informáticas, los servicios de consulta de información se proporcionan a los usuarios finales a través de interfaces basadas en formularios. El problema de esta práctica, es que cualquier cambio en el tipo de sistema persistencia, por ejemplo, la sustitución de un sistema gestor de bases de datos por otro, por un sistema de persistencia basado en documentos XML, por un sistema de persistencia basado en objetos JSON, etc. requiere también la modificación de las consultas sobre las que se apoyan los formularios y, por tanto, de la propia aplicación. Sin embargo, gracias a OcQL, cuando el sistema de persistencia evoluciona, las consultas traducidas se obtienen automáticamente sin necesidad de modificar la aplicación.

4.3 Partes del lenguaje

Tal y como hemos indicado en el capítulo introductorio de este documento de tesis, el desarrollo del *Lenguaje de Consultas para la Gestión de Acontecimientos* (OcQL) es actualmente un tema abierto de investigación por parte del *Grupo Nóesis de la Universidad de Zaragoza* dado que en su extensión prevista comprende tres partes:

1 Acceso a acontecimientos:

  1.1 *Lenguaje de Acceso a Acontecimientos* (*Occurrence Access Language*; *OcAL*).

  1.2 *Lenguaje de Acceso a Información de Acontecimientos* (*Occurrence Information Access Language*; *OcIAL*).

2 *Lenguaje para la Definición de Acontecimientos* (*Occurrence Definition Language*; *OcDL*).

3 *Lenguaje para la Modificación de Acontecimientos* (*Occurrence Modification Language*; *OcML*).

El alcance de la presente tesis comprende el desarrollo de la primera parte estando previsto el desarrollo de los otros dos como parte del trabajo futuro.

En este apartado vamos a describir brevemente en qué consiste cada uno de los lenguajes que forman parte de OcQL según el alcance logrado y el previsto.

Lenguaje de Acceso a Acontecimientos

El *Lenguaje de Acceso a Acontecimientos* (*Occurrence Access Language*; *OcAL*) está dirigido a facilitar la obtención de acontecimientos almacenados en una base de acontecimientos. OcAL contempla consultas tanto para obtener acontecimientos incluidos en una estructura longitudinal como, por ejemplo, una *Línea de Vida*, como para obtener acontecimientos o elementos de los mismos que no constituyen una estructura longitudinal. Los elementos de acontecimientos que pueden obtenerse son aquellos que los definen; es decir, objeto, ejecución de protocolo y estado.

El *Lenguaje de Acceso a Acontecimientos* se divide en dos variantes. La primera variante, a la que denominamos *Lenguaje No Generalizado de Acceso a Acontecimientos* (*Non Generalized Occurrence Access Language; Non Generalized OcAL*), está dirigida a posibilitar el acceso a los acontecimientos registrados en una base de acontecimientos sin necesidad de tener conocimiento de los conceptos definidos en el *Metamodelo de Bases de Acontecimientos*; la segunda, a la que denominamos *Lenguaje Generalizado de Acceso a Acontecimientos* (*Generalized Occurrence Access Language; Generalized OcAL*) incorpora referencias a los conceptos definidos en el metamodelo posibilitando obtener información registrada en el *Repositorio de Metadatos*.

Lenguaje de Acceso a Información de Acontecimientos

El *Lenguaje de Acceso a Información de Acontecimientos* (*Occurrence Information Access Language*; *OcIAL*) está dirigido a facilitar la extracción de conocimiento de una base de acontecimientos, bien de la información almacenada en la misma, bien sobre su estructura.

Para ello, el *Lenguaje de Acceso a Información de Acontecimientos* (*OcIAL*) contempla dos tipos de consultas: (1) Las consultas del primer tipo, a las que denominamos *Consultas Sobre la Base de Acontecimientos* (*Occurrence Base Queries*) posibilitan obtener la información almacenada en una base de acontecimientos, por ejemplo información sobre objetos, procesos ejecutados, estados, etc.; (2) las consultas del segundo tipo, a las que denominamos *Consultas Sobre el Diccionario* (*Dictionary Queries*), posibilitan acceder al diccionario o repositorio de metadatos para obtener información sobre la estructura de una base de acontecimientos como por ejemplo los tipos de objetos, los tipos de procesos que pueden ejecutarse, los tipos de estado, etc. contemplados en la misma.

Lenguaje para la Definición de Acontecimientos

Actualmente, como parte del trabajo de investigación futuro, está previsto que el *Lenguaje para la Definición de Acontecimientos* (*Occurrence Definition Languaje*; *OcDL*) posibilite la creación, actualización y borrado de las estructuras de una base de acontecimientos que posibilitan almacenar acontecimientos, información de los mismos y de sus elementos.

Salvando las distancias, un caso equiparable a la funcionalidad que se espera ofrecer a través de OcDL sería el del *Lenguaje de Definición de Datos* (*Data Definition Language*; *DDL*) de SQL, el cual comprende entre otras, las consultas para la creación (CREATE TABLE), modificación (ALTER TABLE) y eliminación (DROP TABLE) de tablas.

Lenguaje para la Modificación de Acontecimientos

Junto con el *Lenguaje para la Definición de Acontecimientos*, como parte del trabajo de investigación futuro, está previsto el desarrollo del *Lenguaje para la Modificación de Acontecimientos* (*Occurrence Modification Languaje*; *OcML*) con un alcance que posibilite el registro, modificación y borrado de acontecimientos en una base de acontecimientos.

Un caso equiparable a la funcionalidad que se espera ofrecer a través de OcML sería el del *Lenguaje de Manipulación de Datos* (*Data Manipulation Language*; *DML*) de SQL, el cual comprende entre otras, las consultas para la inserción (INSERT), actualización (UPDATE) y borrado (DELETE) de datos.

4.4 Elementos del lenguaje

El *Lenguaje de Consultas para la Gestión de Acontecimientos* (OcQL) utiliza diferentes tipos de elementos, concretamente los siguientes:

- **Cláusulas** (*Clauses*), que, en conjunto, dan forma a las consultas del lenguaje OcQL; por ejemplo, `SELECT OBJECT OF TYPE`. En ciertos casos, como iremos viendo, son opcionales.
- **Expresiones** (*Expressions*), que, como resultado, producen valores. Por ejemplo, `Volume.quantity * 0.1`.
- **Predicados** (*Predicates*), los cuales especifican condiciones que deben ser evaluadas para dar un valor booleano (true o false). Por ejemplo, `Volume.quantity > 40`.
- **Consultas** (*Queries*), formadas por un conjunto de cláusulas, cuya función es posibilitar la recuperación de información de una base de acontecimientos en función de las condiciones definidas; por ejemplo, *consultar los objetos con un volumen mayor a 40*.

4.5 Palabras reservadas del lenguaje

Las palabras reservadas son términos que no pueden utilizarse libremente por los usuarios al definir consultas OcQL salvo que refieran a ellas declarándolas entre comillas simples como cadenas de caracteres.

Como en otros lenguajes de consultas como SQL, en el *Lenguaje de Consultas para la Gestión de Acontecimientos* (*OcQL*), todo término que forma parte de la sintaxis del lenguaje que veremos en los siguientes capítulos es considerado una palabra reservada; por ejemplo, `SELECT`, `RELATED TO`, `RESTRICTED BY`, etc. Pero, a diferencia de otros lenguajes para la definición de consultas, OcQL contempla la existencia de un conjunto de palabras reservadas adicionales que no forman parte de la sintaxis del lenguaje dado que su uso depende de la estrategia de diseño seleccionada.

Como hemos indicado anteriormente, una de las premisas de OcQL es que pueda ser adaptado a diferentes estrategias de diseño basadas en acontecimientos; tal y como ya se ha explicado, esto se logra especificando las siguientes dimensiones del lenguaje:

1 Los tipos de acontecimientos contemplados.

2 Los tipos de estructuras longitudinales utilizadas.

3 Los tipos de elementos de los acontecimientos que pueden ser seleccionados en las consultas sobre estructuras longitudinales.

4 Los tipos de elementos de los acontecimientos que pueden ser seleccionados en las consultas sobre acontecimientos.

Una vez que se ha seleccionado la estrategia de diseño y se han adaptado a la misma las dimensiones del lenguaje, es posible utilizar la totalidad o parte de las siguientes palabras reservadas al declarar consultas OcQL. A continuación, mostramos algunas de ellas:

1 Palabras reservadas para referir a elementos de los acontecimientos:

`OBJECT`, `ACTIVE STATE`, `CURRENT STATE`, `INITIAL STATE`, `PROTOCOL EXECUTION`.

2 Palabras reservadas para referir a otros elementos contemplados en el metamodelo:

`PROCESS`, `PROTOCOL`, `PERFORMER`, `EFFECT`, `DATA CHANGE`, `STATE CHANGE`, `FIRED TRANSITION`.

3 Palabras reservadas para referir a tipos de estructuras longitudinales.

`LIFELINE`, `HISTORY`.

Como hemos podido ver, para lograr el objetivo de que OcQL sea altamente personalizable, se han contemplado una serie de elementos utilizados en la definición de consultas que no se han incluido como parte de la sintaxis del lenguaje sino que se han considerado palabras reservadas; esto posibilita, por ejemplo, realizar implementaciones del *Framework OcQF* que no recojan ciertos conceptos del metamodelo como por ejemplo *Tipo de Proceso*, *Tipo de Protocolo*, etc.; en estos dos casos, el uso de las palabras reservadas `PROCESS` y `PROTOCOL` no sería posible al declarar consultas OcQL.

El *Diccionario de Palabras Reservadas* se encuentra definido en el *Repositorio de Metadatos* y su contenido, del mismo modo que las dimensiones del lenguaje, es particularizable a las decisiones tomadas. Así pues, si como parte de una estrategia de diseño, se contemplara un tipo de estructura longitudinal diferente, además de realizar su implementación correspondiente, habría que incluir en el diccionario la palabra reservada con la que referirlo. Por ejemplo, en la estrategia de diseño para construir un sistema de gestión de producción alimentaria, se podría decidir contemplar la inclusión de la *Traza* como uno de los tipos de estructura longitudinal; en este caso, el diccionario de palabras reservadas debería incluir una en particular para referir a este tipo de estructura, por ejemplo, `TRACE`. Del mismo modo, en la estrategia de diseño descrita en el apartado 2.2, se decidió contemplar los tipos de estructura longitudinal *Historia* y *Línea de Vida* y, por ello, en el diccionario se incluyeron respectivamente `HISTORY` y `LIFELINE` como palabras reservadas.

Capítulo 5. El proceso de traducción

5.1 Introducción

El proceso de traducción es un pilar fundamental para que el *Lenguaje de Consultas para la Gestión de Acontecimientos* (OcQL) sea posible, pues, a través de este proceso, es posible traducir una consulta OcQL a una consulta declarada en un lenguaje compatible con el sistema de persistencia en el que se encuentre implantada una base de acontecimientos. Como hemos indicado, este proceso es el que dota al lenguaje de su característica de independencia de la tecnología.

En el proceso de traducción intervienen varios de los componentes del *Framework OcQF* tal y como puede observarse en el siguiente diagrama de flujo:

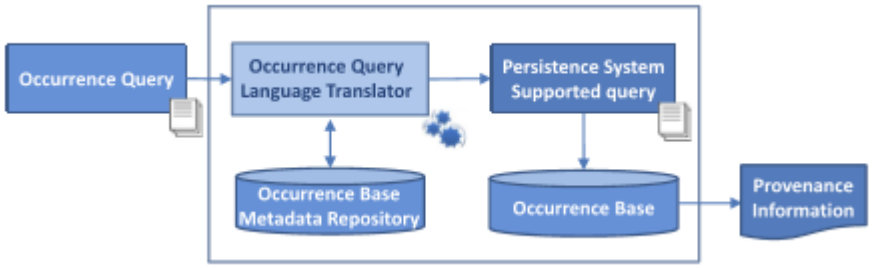


Figura 5.1 Componentes que intervienen en el proceso de traducción de una consulta OcQL

El diagrama de flujo de la Figura 5.1 muestra no sólo los componentes del Framework OcQF que intervienen en el proceso de traducción, sino las relaciones que se establecen entre ellos durante el mismo.

Concretamente, el flujo del proceso de traducción transcurre de la siguiente manera:

1 Una *Consulta OcQL* (componente *Occurrence Query*) es proporcionada al *Traductor de Consultas OcQL* (componente *Occurrence Query Language Translator*) como un parámetro de entrada.

2 El *Traductor de Consultas OcQL*, en el que se encuentra implementado el algoritmo de traducción, procesa la *Consulta OcQL* y la traduce apoyándose en la información almacenada en el *Repositorio de Metadatos* (componente *Occurrence Base Metadata Repository*). El algoritmo de traducción, tal y como veremos más adelante, requiere esta información para conocer la estructura de la base de acontecimientos y así construir una consulta que pueda ser ejecutada sobre la misma.

3 El traductor produce como salida una consulta compatible con el sistema de persistencia (*Persistence System Supported Query*) sobre el que esté implantada la *Base de Acontecimientos* (componente *Occurrence Base*).

4 La ejecución de la consulta resultante sobre la *Base de Acontecimientos* deriva en la extracción de la información deseada.

Más específicamente, el proceso de traducción de una consulta OcQL para obtener una consulta en el lenguaje del sistema de persistencia sobre el que está implantada una base de acontecimientos se implementa en el traductor a través de un algoritmo principal que coordina la traducción y la delega en diferentes bloques en función de cláusula de la consulta OcQL a traducir. La estructura del algoritmo de traducción depende de la sintaxis del lenguaje de consultas del sistema de persistencia y varía en cada caso.

En el siguiente apartado profundizaremos en la *pieza* fundamental que interviene en el proceso de traducción: el *Traductor de Consultas OcQL*.

5.2 El Traductor de Consultas OcQL

El *Traductor de Consultas OcQL*, denominado *Occurrence Query Language Translator* (*OcQL Translator*), es uno de los componentes del *Framework OcQF* que participan en el proceso de traducción, concretamente es el que tiene el rol más importante dado que es el responsable de traducir automáticamente las consultas OcQL que se le proporcionan como información de entrada en consultas compatibles con el sistema de persistencia sobre el que está implantada la base de acontecimientos de la que se pretende obtener información.

El *Traductor de Consultas OcQL* es uno de los componentes que pueden adaptarse al realizar una implementación del *Framework OcQF* y para ello deben tenerse presentes los siguientes aspectos:

1 La estrategia de diseño orientada a acontecimientos seleccionada para el desarrollo de la base de acontecimientos.

   Como parte de la estrategia de diseño se deciden los conceptos que se recogen del *Metamodelo de Bases de Acontecimientos* y la forma en la que se personalizan las dimensiones abiertas del lenguaje: Tipos de acontecimiento, tipos de elementos propios del concepto *Acontecimiento*, y tipos de estructuras longitudinales contempladas. Estas decisiones completan el lenguaje y por tanto tienen impacto en el traductor. Por ejemplo, el traductor debe detectar si una consulta OcQL está refiriendo a un concepto del metamodelo que no ha sido recogido por la estrategia de diseño seleccionada y avisar del error.

2 El sistema de persistencia sobre el que se implantará la base de acontecimientos.

   Este aspecto es fundamental dado que de esta decisión dependerá el lenguaje al que serán traducidas las consultas OcQL y las características del entorno sobre el que se implantará la base de acontecimientos. Si, por ejemplo, el sistema de persistencia pertenece al ámbito de las bases de datos relacionales, este correspondería con un sistema gestor de bases de datos como podría ser Oracle Database, SQL Server, PostgreSQL, MySQL, etc.

3 El lenguaje de consultas del sistema de persistencia.

   El lenguaje al que serán traducidas las consultas OcQL dependerá del sistema de

persistencia seleccionado. Por ejemplo, si el sistema de persistencia corresponde con un sistema gestor de bases de datos relacionales, el lenguaje de las consultas obtenidas será SQL, si el sistema de persistencia está basado en documentos XML, el lenguaje será XPath, etc.

4 La sintaxis y semántica de OcQL.

Tal y como veremos más adelante, una de las responsabilidades del traductor es validar las consultas OcQL que le llegan como información de entrada. Las consultas OcQL proporcionadas no sólo deben ser válidas frente a la sintaxis del lenguaje, sino también semánticamente válidas; por ejemplo, una consulta OcQL que refiere a un determinado tipo de objeto (por ejemplo, *Sample*) puede ser sintácticamente válida, pero ser incorrecta semánticamente si dicho tipo de objeto no ha sido contemplado en el modelo de la base de acontecimientos y mapeado en el *Repositorio de Metadatos*.

5 La forma en que se ha implementado el *Repositorio de Metadatos*.

Este aspecto es fundamental dado que el *Repositorio de Metadatos* es el mecanismo del que dispone el traductor para conocer la estructura de una base de acontecimientos durante el proceso de traducción de las consultas OcQL. Por ejemplo, no es lo mismo disponer de un repositorio de metadatos que haya sido implementado sobre una base de datos relacional, que disponer de uno implementado sobre una base de datos NoSQL, o de uno implementado utilizando documentos XML

.

III. EL LENGUAJE DE ACCESO A ACONTECIMIENTOS

Capítulo 6. El Metamodelo de Bases de Acontecimientos

6.1 Introducción

Denominamos *Metamodelo de Bases de Acontecimientos* (*Occurrence Base Metamodel*; *OcBase Metamodel*) al conjunto de conceptos que pueden ser utilizados en la especificación de modelos de base de acontecimientos. Recordemos que metamodelo y *Modelo de Base de Acontecimientos* (*Occurrence Base Model*) son tres de los componentes que forman parte del *Framework OcQF* que hemos descrito en el Capítulo 3.

Además de conceptos, el *Metamodelo de Bases de Acontecimientos* también especifica las relaciones entre ellos y las multiplicidades de dichas relaciones. Las multiplicidades de cada relación entre los conceptos del metamodelo actúan como restricciones mínimas que debe cumplir todo modelo especificado a partir del metamodelo. En este aspecto, el metamodelo propuesto pretende ser muy general, posibilitando que sean los propios modelos de bases de acontecimientos, en función de la estrategia de diseño seleccionada, los que fijen sus restricciones específicas para adaptarse a las necesidades particulares del dominio a modelar. Por ejemplo, *Tipo de Ejecutor* es un concepto del metamodelo que posibilita diseñar modelos en los que se pueda conocer el responsable de la ejecución de un determinado protocolo; es posible especificar un modelo que no recoja este concepto y, por tanto, que la base de acontecimientos derivada de dicho modelo tampoco lo haga.

La introducción del *Metamodelo de Bases de Acontecimientos* es una novedad publicada en [25]; a través de él se logra normalizar el conjunto de conceptos que pueden ser recogidos por una determinada estrategia de diseño, como puede ser la descrita en [24], para definir modelos de bases de acontecimientos.

El *Metamodelo de Bases de Acontecimientos* se describe en el nivel M2 siguiendo el esquema de cuatro niveles propuesto por UML [54]. El nivel M2 se sitúa por encima del nivel M1 en el que se describen los *Modelos de Bases de Acontecimientos* (*Occurrence Base Models*) que, a su vez, está por encima del nivel M0, en el que se describen las *Bases de Acontecimientos* (*Occurrence Bases*). Por otra parte, el nivel M2 se encuentra por debajo del nivel M3, nivel de los meta-metamodelos en el que se establecen la estructura y semántica de los metamodelos. En la siguiente figura se muestran los niveles de UML:

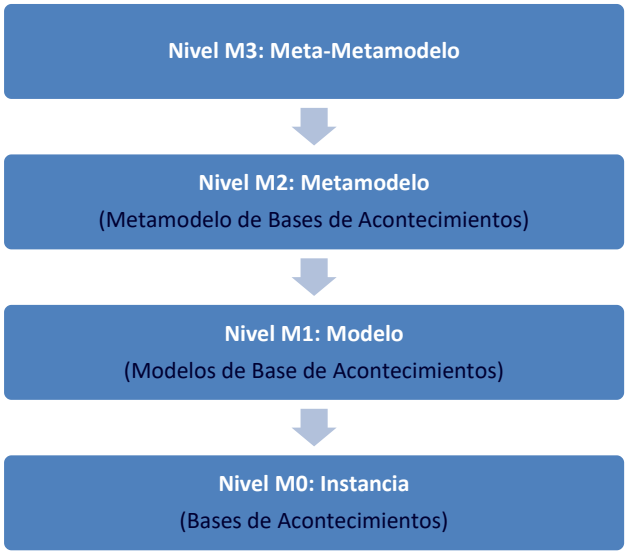


Figura 6.1 Niveles de UML y posicionamiento en los mismos de los componentes del framework

En este capítulo comenzaremos presentando la estructura del *Metamodelo de Bases de Acontecimientos* a través de un diagrama de conceptos y continuaremos describiendo cada uno de los conceptos que forman parte del mismo.

6.2 Diagrama de conceptos

El sistema de conceptos que definen el *Metamodelo de Bases de Acontecimientos* publicado en [25] se describe a través del siguiente diagrama de clases:

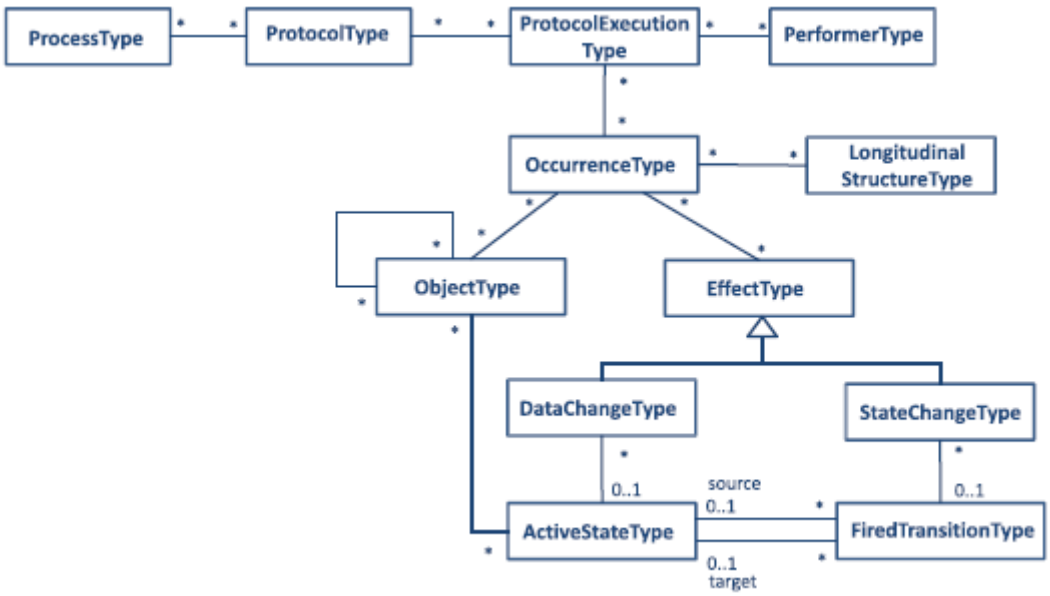


Figura 6.2 Metamodelo de Bases de Acontecimientos

A continuación, se describen dichos conceptos y las relaciones mostradas.

6.3 Tipo de Acontecimiento

El concepto que es pilar o base del metamodelo propuesto es *Tipo de Acontecimiento* (OccurrenceType), dicho concepto refiere a todas las clases de acontecimiento contempladas en un modelo concreto de bases de acontecimientos. Dicho de otra forma, las instancias de la meta-clase OccurrenceType declaradas en un determinado modelo de base de acontecimientos representan a cada uno de los tipos de acontecimiento que se quieren contemplar en dicho modelo.

Podemos definir el concepto *Tipo de Acontecimiento* como una familia de acontecimientos que puede ser referida genéricamente por una expresión del lenguaje natural. Por ejemplo: 'lluvia', 'alicuotado', 'envío', 'pago de factura', etc.

Por ejemplo, en el contexto del sistema informático *Arasis* que, recordemos, se enmarca en el dominio de los biobancos [13], algunos de los tipos de acontecimientos que se han contemplado son *Registro*, *Alicuotado*, *Almacenamiento* o *Cesión* de muestras biológicas que, a su vez, refieren a acontecimientos concretos registrados en la base de acontecimientos (por ejemplo, al alicuotado de la muestra de sangre extraída el 01/10/2015 al empleado nº. 45621 de la General Motors, al almacenamiento del tubo de orina O389761 en la posición (1, 1) del rack R4218, etc.).

Al igual que otros aspectos contemplados en la representación de un determinado universo del discurso (*universe of discourse* – *UoD*), los tipos de acontecimiento dependen del dominio a modelizar. Por ejemplo, si en lugar del dominio de un biobanco, se tratara de modelar el dominio de una institución educativa, algunos de los tipos de acontecimientos serían *Calificación* o *Diligenciado* de actas académicas, *Convocatoria* de tribunales de evaluación, etc.

Como es sabido, a la hora de representar un universo del discurso o dominio a través de un modelo es primordial identificar y comprender los aspectos importantes para dicho dominio; en el caso de los modelos para la construcción de bases de acontecimientos, esta afirmación se traduce en que lo más importante es identificar y comprender los tipos de acontecimientos que formarán parte del sistema a modelar puesto que de ellos dependerán gran parte de los conceptos representados en el modelo tal y como se explicará más adelante.

Una prueba para la afirmación realizada en el párrafo anterior es que la estructura que se considera para el concepto *Tipo de Acontecimiento* se ha definido en un primer nivel de descripción por los conceptos *Tipo de Ejecución de Protocolo* (ProtocolExecutionType), *Tipo de Objeto* (ObjectType) y *Tipo de Efecto* (EffectType) que se describen en los siguientes apartados.

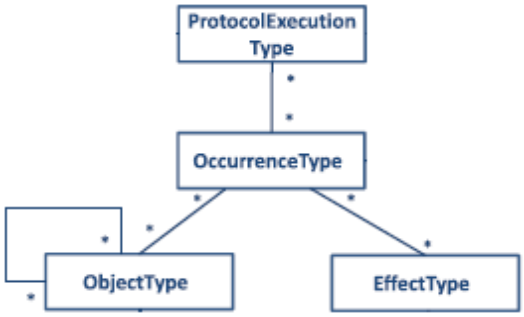


Figura 6.3 El concepto Tipo de Acontecimiento y sus conceptos relacionados

En relación a lo anterior, es importante señalar que la forma de representar y relacionar estos conceptos en el metamodelo no es trivial y se fundamenta en la definición de acontecimiento basado en protocolo dada en [24] y referida en el apartado 2.1; obsérvese que los tres primeros conceptos enumerados corresponden respectivamente con cada uno de los tres aspectos referidos en dicha definición: protocolo ejecutado, objeto afectado y efecto producido sobre el objeto.

Un caso particular: Acontecimientos no protocolizados

En el apartado 2.1 hemos presentado el concepto *Acontecimiento* y lo hemos definido como una terna formada por (1) el protocolo ejecutado, (2) el objeto afectado, y (3) el efecto producido sobre dicho objeto. Por otra parte, al inicio de este capítulo, hemos indicado que el *Metamodelo de Bases de Acontecimientos* contempla una serie de conceptos que pueden contemplarse o no en todo modelo de base de acontecimientos basado en él.

Ahora, supongamos que un determinado modelo de base de acontecimientos no recoge los conceptos *Tipo de Protocolo*, *Tipo de Ejecución de Protocolo* y *Tipo de Ejecutor* del metamodelo. Lo que acabamos de comentar es una cuestión muy interesante dado que uno los temas abiertos en la investigación en la que se enmarca esta tesis pretende dar solución al concepto de *Acontecimiento no protocolizado*.

Los acontecimientos naturales son los casos más evidentes de acontecimientos que no están basados en protocolos. Un ejemplo de acontecimiento natural es el nacimiento, otro ejemplo puede ser la lluvia, etc.; si en un sistema quisiéramos representar el hecho de que ha llovido,

no tendría sentido representar el proceso *Llover*.

El caso de la lluvia es todavía más interesante dado que no puede decirse que este acontecimiento haya sucedido sobre un objeto; puede decirse "Ha llovido en Alcalá", pero no podemos considerar Alcalá un objeto como tal. Si quisiéramos registrar el hecho de que ha llovido en una base de acontecimientos, se podría valorar introducir la distinción entre *Objetos Abiertos* y *Objetos Cerrados*. Hablamos de *Objetos Cerrados* cuando podemos decir que sus límites son conocidos, por ejemplo, muestra biológica, alícuota, rack, etc.; como contrapunto, hablamos de *Objetos Abiertos* cuando los límites de los mismos no pueden definirse fácilmente, al menos en relación a los acontecimientos que ocurren sobre ellos; por ejemplo, al decir "Ha llovido en Alcalá", quizás lo que ha ocurrido realmente es que ha llovido en el casco histórico. Así pues, si se quisiera registrar la información sobre la lluvia en zonas, se tendría que especificar el papel que juega una zona como objeto afectado por la lluvia.

Los acontecimientos naturales se distinguen de los artificiales (dentro de los que se encuentran los *Acontecimientos basados en protocolos*), en que no existe un sujeto que sea el responsable de la realización del acontecimiento; es decir, no existe un ejecutor.

A este respecto, aunque hemos pretendido dar unas pinceladas en relación al concepto *Acontecimiento no protocolizado* para intentar mostrar hacia donde se encamina la investigación, es importante señalar que dicho concepto no entra dentro del alcance de esta tesis, alcance dentro del cual sí que entran los *Acontecimientos basados en protocolos*. Desde este punto de vista, el metamodelo propuesto, tal y como se indica en [25], está pensado para sistemas cuya dinámica se encuentra dirigida por un conjunto de procesos que se realizan mediante la ejecución de diferentes protocolos.

6.4 Tipo de Ejecución de Protocolo

El concepto *Tipo de Ejecución de Protocolo* o, más concretamente las instancias de la meta-clase ProtocolExecutionType realizadas en un determinado modelo de base de acontecimientos, representa a las clases de ejecución de protocolo que se quieren contemplar en dicho modelo.

Por ejemplo, en el modelo de la base de acontecimientos de un biobanco, se podría especificar una clase denominada *SampleRecordingProtocolExecution* que es una instancia de la clase ProtocolExecutionType del metamodelo. A través de esta clase, en toda base de acontecimientos que implemente el modelo, es posible incluir una entidad para el registro de las ejecuciones del protocolo *Registrar muestra*.

A continuación, se muestra la especificación del ejemplo anterior a través de un diagrama en el que se agrupan las clases correspondientes en función del nivel de UML al que pertenecen:

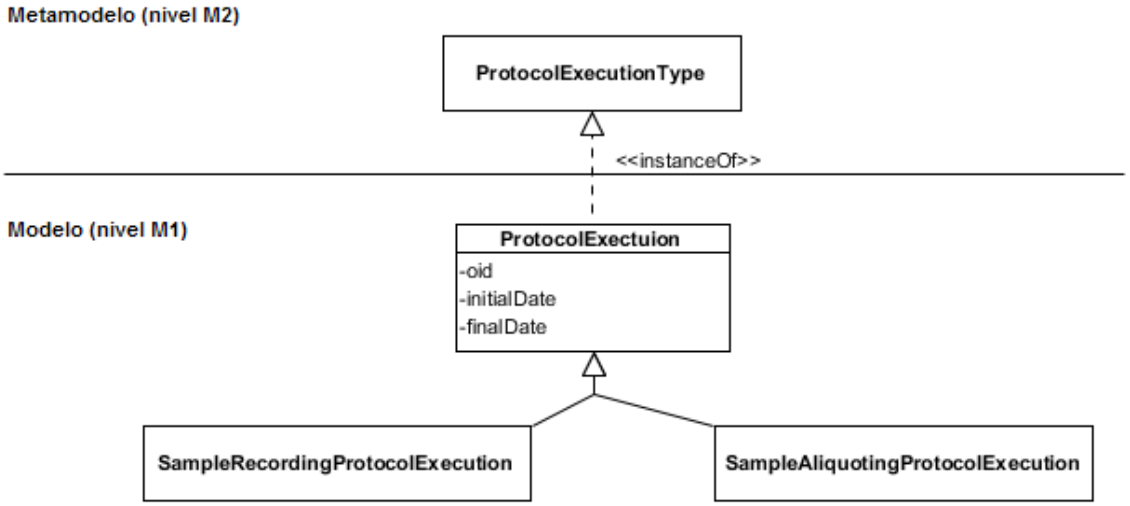


Figura 6.4 El concepto Tipo de Ejecución de Protocolo. Diagrama de clases por niveles UML

Así pues, cuando por ejemplo decimos "Registro de la muestra S159283 registrado el 01/10/2015, realizado entre las 09:00 y las 09:05" estamos refiriendo a la ejecución del protocolo *Registrar muestra* que se realizó en la fecha indicada.

En el metamodelo, el concepto *Tipo de Ejecución de Protocolo* se encuentra relacionado con los conceptos *Tipo de Protocolo* (ProtocolType) y *Tipo de Ejecutor* (PerformerType) que describiremos en los siguientes apartados. Con la introducción de estos dos conceptos, se pretende resolver una de las problemáticas referidas en la introducción de este documento de tesis a la que pretenden dar solución las investigaciones realizadas: la ausencia en muchos casos de información relevante para el estudio de la procedencia de los datos.

6.5 Tipo de Protocolo y Tipo de Proceso

El concepto *Tipo de Protocolo* que hemos introducido en el apartado anterior está representado en el metamodelo a través de la clase ProtocolType de tal forma que las instancias de la misma declaradas en un determinado modelo de base de acontecimientos corresponden con las clases de protocolo que se quieren contemplar en dicho modelo.

Por ejemplo, en el modelo de base de acontecimientos de un biobanco, podrían contemplarse dos tipos de protocolo: (1) *De ordenador*, y (2) *De laboratorio* representados respectivamente por las clases ComputerProtocol y LaboratoryProtocol. A través de estas clases sería posible distinguir respectivamente entre aquellos protocolos cuya ejecución y traza completa (usuario ejecutor, fechas de ejecución, objetos afectados, etc.) queda registrada en la base de acontecimientos del sistema, de aquellos ejecutados por los técnicos del laboratorio en el mundo real que quedan reseñados en la base de acontecimientos. En el contexto de este ejemplo es importante señalar que se considera que un protocolo de ordenador puede tener correspondencia con un determinado protocolo de laboratorio.

En la siguiente figura se especifica el ejemplo anterior a través de un diagrama en el que se

agrupan las clases correspondientes en función del nivel de UML al que pertenecen:

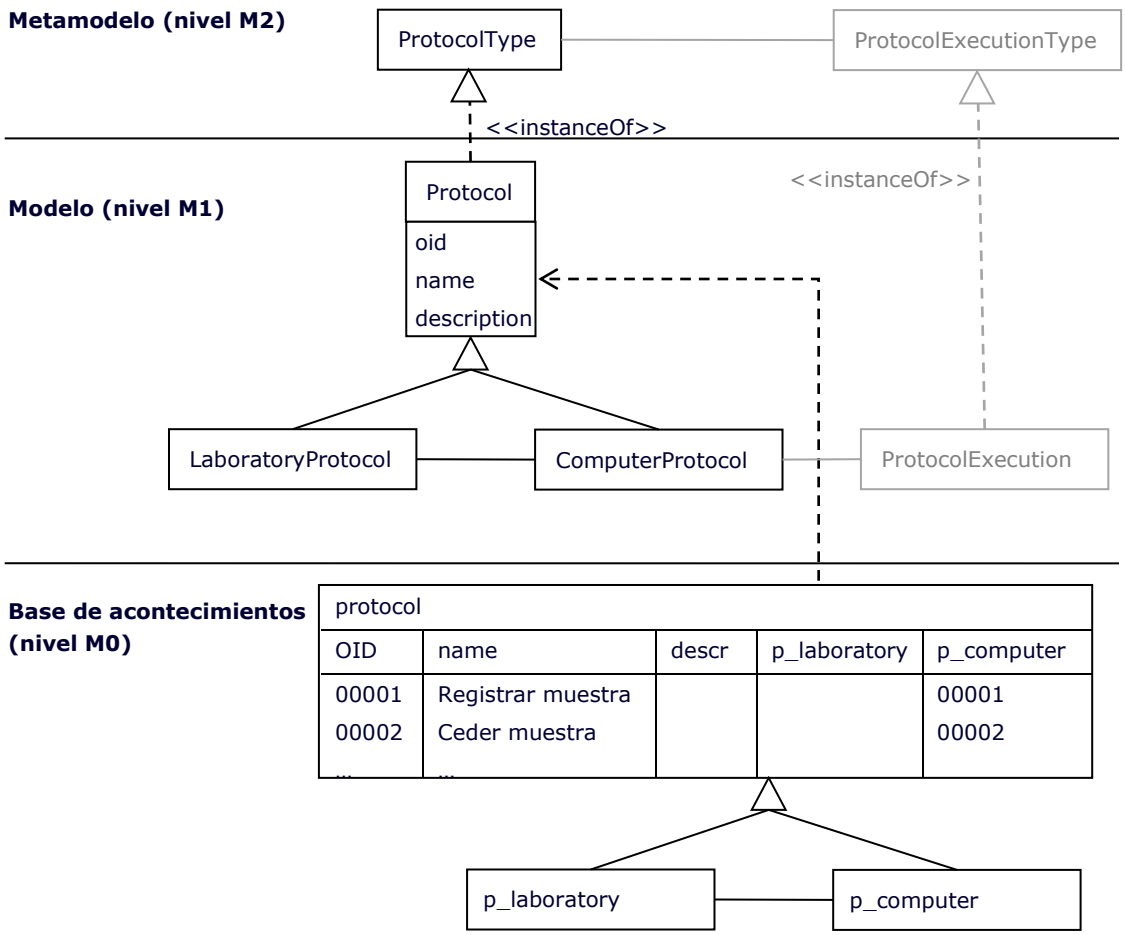


Figura 6.5 El concepto *Tipo de Protocolo*. Diagrama de clases por niveles UML

Tipo de Protocolo se asocia con el concepto *Tipo de Proceso* (ProcessType) materializando así, como ya se ha comentado, la relación entre los conceptos *Protocolo* y *Proceso*:

Definición 1. Un *Proceso* es una tarea concreta, identificable y realizada de principio a fin que podemos conceptualizar como un conjunto de *Protocolos*.

Definición 2. Un *Protocolo* es la especificación de un conjunto estructurado y ordenado de actividades definido para llevar a cabo un objetivo.

Así pues, podemos decir, que un protocolo es la especificación de un conjunto estructurado y ordenado de actividades y que un proceso está formado por un conjunto de protocolos. Dicho de otra forma, un protocolo define los pasos a seguir para realizar correctamente un proceso.

Figura 6.6 Los conceptos Tipo de Proceso y Tipo de Protocolo en el metamodelo

Si bien en el sistema *Arasis* se registran las diferentes ejecuciones de protocolos realizadas y sus resultados, actualmente no se encuentra implementado el diagrama de actividades; es decir no se registra cómo se desarrolla un proceso. De existir esta implementación, podríamos, por ejemplo, tener un proceso de recogida de muestras que tuviera asociado los protocolos de alicuotado de muestras, de almacenamiento de muestras, etc. que se ejecutarían en este orden.

Cuando el concepto *Tipo de Proceso* se instancia en un modelo de acontecimientos, es posible conocer el proceso al que pertenecen o han pertenecido las ejecuciones de protocolo registradas en una determinada base de acontecimientos.

En la siguiente figura se muestran las relaciones ente los conceptos *Tipo de Protocolo* y *Tipo de Proceso* tanto en el metamodelo como en un modelo basado en el mismo:

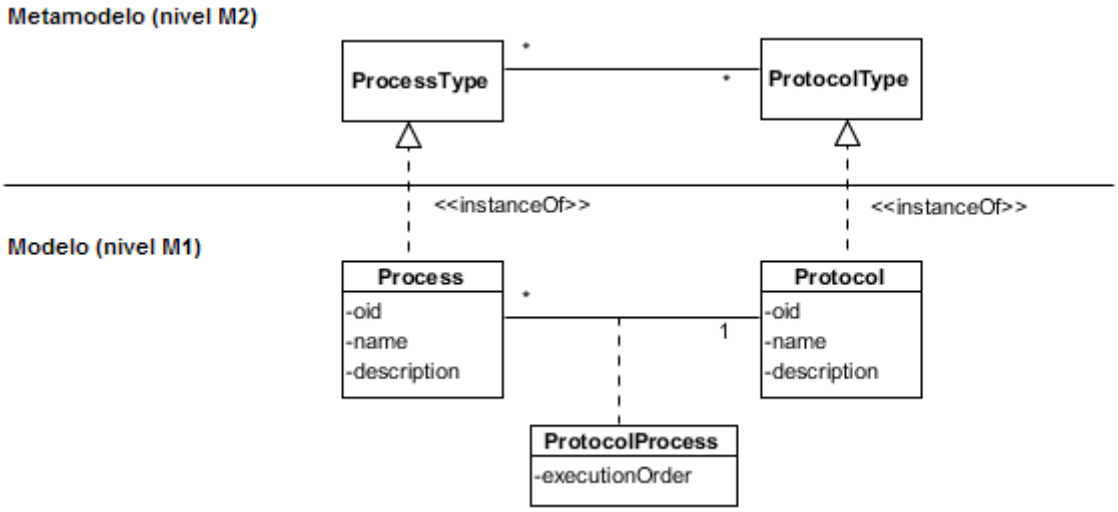


Figura 6.7 Los conceptos Tipo de Protocolo y Tipo de Proceso. Diagrama de clases por niveles UML

Sobre acontecimientos, procesos y protocolos

Para cerrar este apartado, a continuación, vamos a aclarar un poco más los conceptos que acabamos de presentar y más particularmente las relaciones que se establecen entre ellos.

Un acontecimiento es algo que se percibe como el resultado de un acontecer, de algo que ha ocurrido. Por ejemplo: 'se ha obtenido una alícuota', 'se ha realizado el envío de un pedido', 'se ha pagado una factura', etc. Todos estos casos de tipos de acontecimientos tienen asociado un proceso. Por ejemplo: 'alicuotar', 'enviar', 'pagar', etc.

Un proceso puede desarrollarse de forma natural, sin la intervención de un humano o de una máquina, o por la actuación de uno o más agentes que intervienen para que el proceso se

desarrolle, produciendo el acontecimiento que corresponda.

En los procesos que se desarrollan por la intervención de un humano o una máquina, consideramos la existencia de uno o más protocolos que guían la forma en cómo se lleva a cabo un proceso. Cada protocolo es una unidad de ejecución que da como resultado una o varios acontecimientos, pudiendo darse casos de que estos acontecimientos formen parte de un acontecimiento más general resultado de la ejecución completa de un proceso.

6.6 Tipo de Ejecutor

El concepto *Tipo de Ejecutor* (PerformerType) se introduce con el objetivo de poder registrar información para identificar al responsable de la ejecución de un determinado protocolo en aquellas bases de acontecimientos resultantes de implementar un modelo que recoja este concepto del metamodelo. Esta información se considera importante como parte de la procedencia de los datos de un sistema.

Por ejemplo, en el modelo de base de acontecimientos de un biobanco, podrían contemplarse los siguientes tipos de ejecutores: (1) *Usuario*, (2) *Sistema*, y (3) *Protocolo*, representados respectivamente por las clases *UserPerformer*, *SystemPerformer* y *ProtocolPerformer*. A través de estas clases es posible distinguir si un protocolo ha sido ejecutado por un usuario autorizado del sistema, por una tarea automatizada del sistema, o por otro protocolo. A continuación, se especifica este ejemplo a través de un diagrama en el que se agrupan las clases correspondientes en función del nivel de UML al que pertenecen:

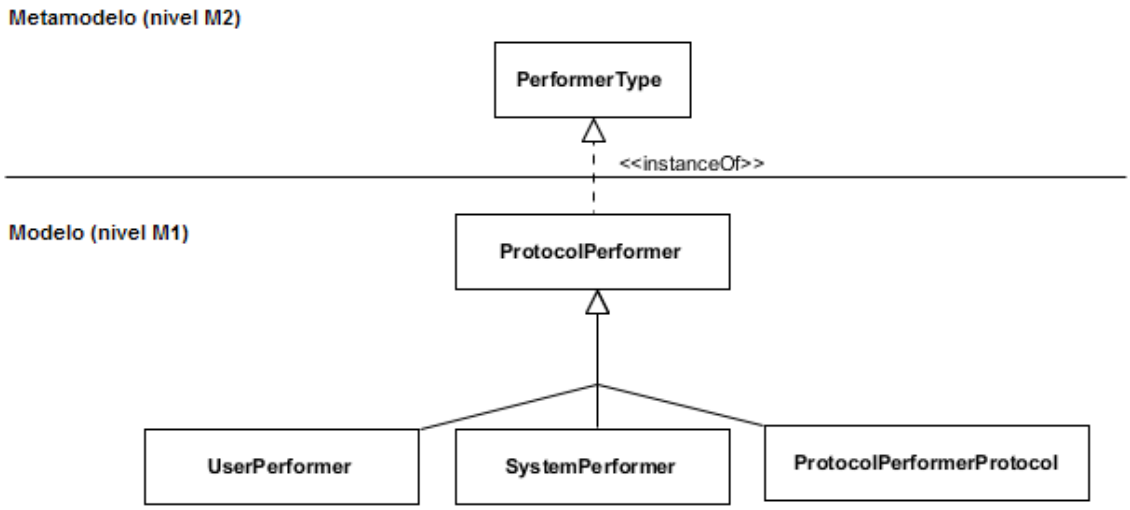


Figura 6.8 El concepto Tipo de Ejecutor. Diagrama de clases por niveles UML

Así pues, cuando por ejemplo decimos "Registro de la muestra S159283 registrado el 01/10/2015 a las 09:00 por el usuario U15398" estamos refiriendo a la ejecución del protocolo *Registrar muestra* por parte de un ejecutor de tipo *Usuario*.

6.7 Tipo de Objeto

La introducción del concepto *Tipo de Objeto* (ObjectType) en el metamodelo posibilita la definición de modelos de bases de acontecimientos que contemplen diferentes clases de objeto que permitan modelar un determinado dominio o universo del discurso.

Objeto es uno de los conceptos fundamentales que dan forma a un acontecimiento; tal y como hemos indicado en el apartado 2.1, es el elemento propio de un acontecimiento que sufre cambios cuando se desencadena la ejecución de un protocolo que produce un efecto sobre el mismo; por otra parte, las estructuras longitudinales basadas en acontecimientos son la representación de un conjunto de acontecimientos ocurridos sobre objetos a lo largo del tiempo.

Los tipos de objeto especificados en cada modelo dependerán exclusivamente del dominio concreto que se requiera modelar; por ejemplo, en el sistema informático *Arasis* cuyo modelo representa el dominio de un biobanco, algunos de los tipos de objeto especificados han sido *Muestra* (biológica), *Alícuota*, *Gradilla*, *Rack*, *Sujeto* (de estudio), *Analítica*, etc.

Cuando hablamos de acontecimientos basados en protocolos, un objeto siempre se origina en el sistema por la acción de un protocolo. Por ejemplo, una factura nace cuando se realiza una venta, un acta cuando un profesor califica a los alumnos de una asignatura, etc.

Por otra parte, los objetos pueden relacionarse entre sí cuando se especifican en un modelo. Por ejemplo, un objeto *Venta* no tiene sentido sin un objeto *Artículo*, un objeto *Acta* no tiene sentido si no existe un objeto *Alumno* y un objeto *Asignatura*, etc.

Aunando todos los conceptos del metamodelo presentados hasta el momento, cuando decimos "Registro de la muestra S159283 registrado el 01/10/2015 a las 09:00 por parte del usuario U1298" estamos refiriendo respectivamente a la ejecución del protocolo *Registrar muestra* sobre un objeto de tipo *Muestra*, por parte de un ejecutor de tipo *Usuario*.

A continuación, se muestra un extracto con algunos de los tipos de objeto contemplados en *Arasis* a través de un diagrama en el que se agrupan las clases correspondientes en función del nivel de UML al que pertenecen:

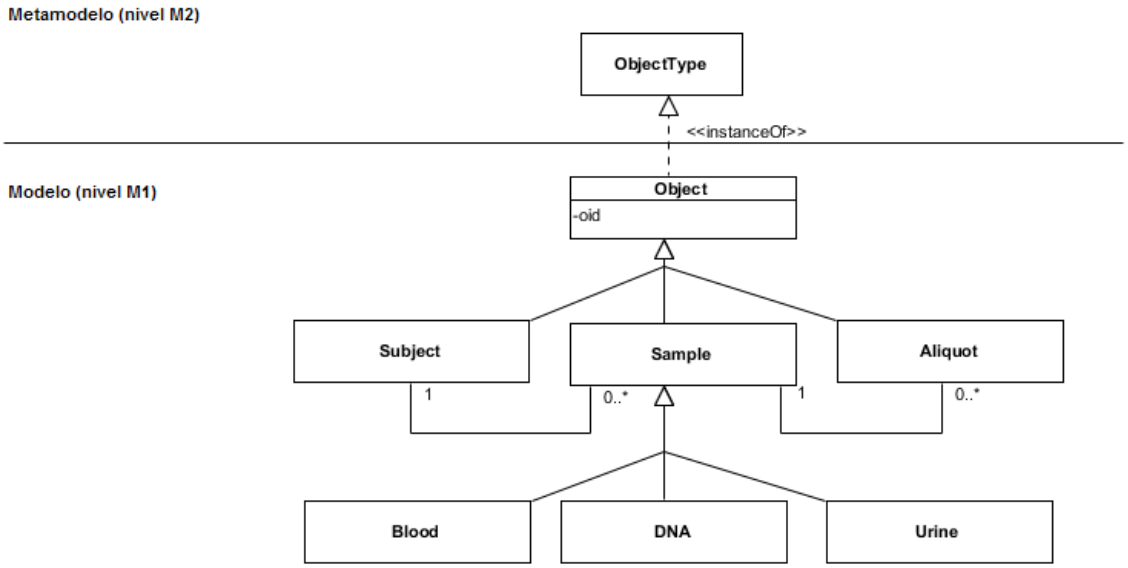


Figura 6.9 El concepto Tipo de Objeto. Diagrama de clases por niveles UML

Sobre los datos propios de un objeto

Como su nombre indica, el objetivo de todo *Sistema de Información* es almacenar y facilitar la gestión de información; para que esto sea posible, los objetos contemplados en dicho sistema deben tener un conjunto de datos propios de los mismos de los que se extrae parte del conocimiento del sistema. Por lo general, cuando se habla de "datos propios de un objeto", inmediatamente se piensa en aquellos datos directamente relacionados con dicho objeto.

Desde nuestro punto de vista, cuando hablamos de "datos propios de un objeto", nos referimos no sólo a los datos directamente relacionados con él, sino también a aquellos datos relacionados con alguno de los estados por los que puede pasar dicho objeto a lo largo de su vida en el sistema; recordemos que hemos hablado de la relación entre objetos y estados en el apartado 2.2. La principal diferencia entre ambos casos es que, cuando los datos están directamente relacionados con un objeto, no es posible seguir de manera natural su evolución temporal, algo que por el contrario si es posible cuando los datos se encuentran relacionados con los estados de un objeto tal y como veremos en el siguiente apartado. En capítulos posteriores retomaremos este concepto.

6.8 Tipo de Efecto

El concepto *Tipo de Efecto* (EffectType) se introduce en el metamodelo para posibilitar la especificación de los cambios producidos en un objeto por la ejecución de un protocolo que actúa sobre él.

Cuando en el apartado 2.1 hemos mostrado cómo se entiende el concepto *Acontecimiento* desde el punto de vista de esta tesis, hemos indicado que un acontecimiento es el resultado de la ejecución de un protocolo que causa un determinado efecto sobre un objeto. Por otra

parte, en función de la estrategia de diseño seleccionada, podemos considerar que los datos que son propios de un objeto (1) puedan estar directamente relacionados con él, y/o (2) que puedan estar relacionados con alguno de sus estados. Teniendo esto presente, consideramos que podrían ocurrir hasta cuatro posibles tipos de efecto:

1 Un **cambio en los datos que no produce un cambio de estado** del objeto. En este caso, el cambio afecta a los datos propios del objeto, el cual se mantiene en el mismo estado en el que se encontraba.

2 Un **cambio en los datos que** produce un cambio de estado. En este caso, los datos corresponden con los relacionados con el estado afectado.

3 Un **cambio de estado que produce un cambio en los datos** vinculados al mismo.

4 Un **cambio de estado que no produce cambios en los datos**.

Los tres últimos casos están representados a través del concepto *Cambio de Estado*, mientras que el primero se encuentra representado a través del concepto *Cambio en los Datos*. El segundo y el tercer caso son un tanto peculiares, de alguna forma podría decirse que son casos híbridos dado que, si bien se entienden como un *Cambio de Estado*, a su vez, desencadenan un *Cambio en los Datos*.

El segundo caso podría darse, por ejemplo, en la base de acontecimientos que diera soporte a un sistema de gestión de un aparcamiento; supongamos que en dicha base de acontecimientos hay un tipo de objeto *Aparcamiento* con dos tipos de estado, *Disponible* y *Ocupado*; en un momento dado, un aparcamiento 'ap1' con una capacidad de 500 plazas, se encuentra en un estado 'est1' (*Disponible*) al tener 499 plazas ocupadas; si en ese momento aparcara un coche, el aparcamiento pasaría a tener las 500 plazas ocupadas, lo que produciría un nuevo estado 'est2' (*Ocupado*).

El tercer caso podría darse, por ejemplo, en la base de acontecimientos que diera soporte a un sistema de gestión académica, al determinar por ejemplo el punto de corte en un proceso de solicitud de becas en función del número de alumnos que han solicitado una beca, de sus calificaciones y del número de becas disponibles; supongamos que en dicha base de acontecimientos hay un tipo de objeto *Beca* con dos tipos de estado, *No concedida* y *Concedida*; la información sobre la calificación de corte depende del tipo de estado *Concedida*. El ejecutarse el protocolo de concesión de becas, la beca 'bec1' pasa de un estado 'eb1' (*No concedida*) a un estado 'bec2' (Concedida), lo cual ocasiona un cambio en los datos de la misma, en este caso la introducción de la calificación de corte con la que se ha concedido.

El *Metamodelo de Bases de Acontecimientos* contempla los cuatro casos que acabamos de mencionar agrupándolos en dos posibles tipos de efecto:

1 Un **cambio del estado activo** (ActiveStateType) del objeto afectado (StateChangeType).

2 Un **cambio en los datos** propios del objeto afectado (DataChangeType).

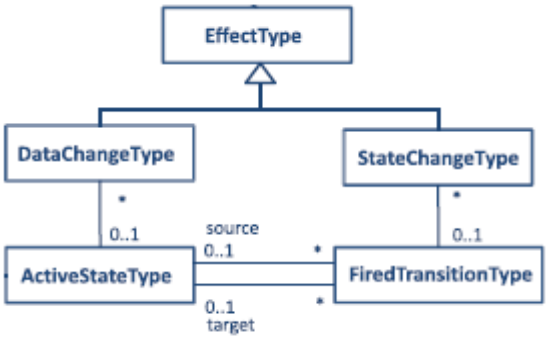


Figura 6.10 El concepto Tipo de Efecto en el metamodelo

Antes de continuar, es importante aclarar algunos de los conceptos que hemos ido introduciendo: *Estado*, *Estado Activo* y *Estado Actual*. Los dos primeros conceptos se han referido en [25] donde, tal y como se indica, no es lo mismo hablar de *Estado* que de *Estado Activo*:

> *"A state represents the possibility of an object of being in it, and an active state represents the actual time an object remains in a particular state."*

Es decir; un *Estado* representa la posibilidad de que un objeto se encuentre en él, mientras que *Estado Activo* refiere al estado en el que un objeto se ha encontrado en un momento temporal dado (pasado o presente). *Estado Actual* refiere al estado en el que un objeto se encuentra en el momento que consideramos como actual.

La clase ActiveStateType del metamodelo se instancia en un modelo de base de acontecimientos para poder disponer de bases de acontecimientos en las que se puedan almacenar todos los estados de un objeto que han sido estados actuales a lo largo del tiempo, ahora y en el pasado. A este respecto, es importante señalar que, para cada objeto, en cada instante únicamente hay un estado actual.

Cuando el tipo de efecto corresponde con un **cambio en el estado activo**, el metamodelo contempla la especificación de:

1 El **tipo de transición** (FiredTransitionType) que ha producido dicho cambio de estado.

   La clase FiredTransitionType refiere a las clases que sirven para almacenar el cambio de un estado a otro. Un cambio de estado se produce siempre por el disparo de una transición.

   Llegados a este punto, definimos los conceptos *Transición* y *Transición Disparada* de la siguiente forma: Una *Transición* que refiere a la posibilidad de que un objeto pase de un estado a otro; por otra parte, a la transición que se lleva a cabo en un momento dado de modo que un objeto cambia su estado actual, pasando de uno a otro, la denominamos *Transición Disparada*.

   Así pues, la clase FiredTransitionType se instancia en un modelo de base de acontecimientos para poder disponer de bases de acontecimientos en las que se puedan almacenar todas las transiciones que se han disparado a lo largo del tiempo desencadenando cambios en el estado de los objetos.

2 El **estado de partida** (Source). Es decir, el estado activo en el momento previo al disparo de la transición.

3 El **estado al que se ha llegado** (Target). Es decir, el estado que ha pasado a ser el nuevo estado activo una vez se ha completado la transición.

4 Los posibles **cambios en los datos** (DataChangeType) que se hayan podido producir, los cuales se encontrarán relacionados con el estado al que se ha llegado.

Por ejemplo, en el sistema *Arasis*, la ejecución de un protocolo de almacenamiento de muestras (tipo BoxStorageProtocolExecution), desencadena una transición de tipo ObtainedSample2StoredSample que produce un cambio estado, concretamente provoca que se pase de un estado que representa que una muestra ha sido obtenida (ObtainedSample), a otro que representa que la muestra ha sido almacenada (StoredSample). En la siguiente figura se muestra la especificación de este ejemplo sobre un extracto del modelo de la base de acontecimientos del sistema *Arasis*:

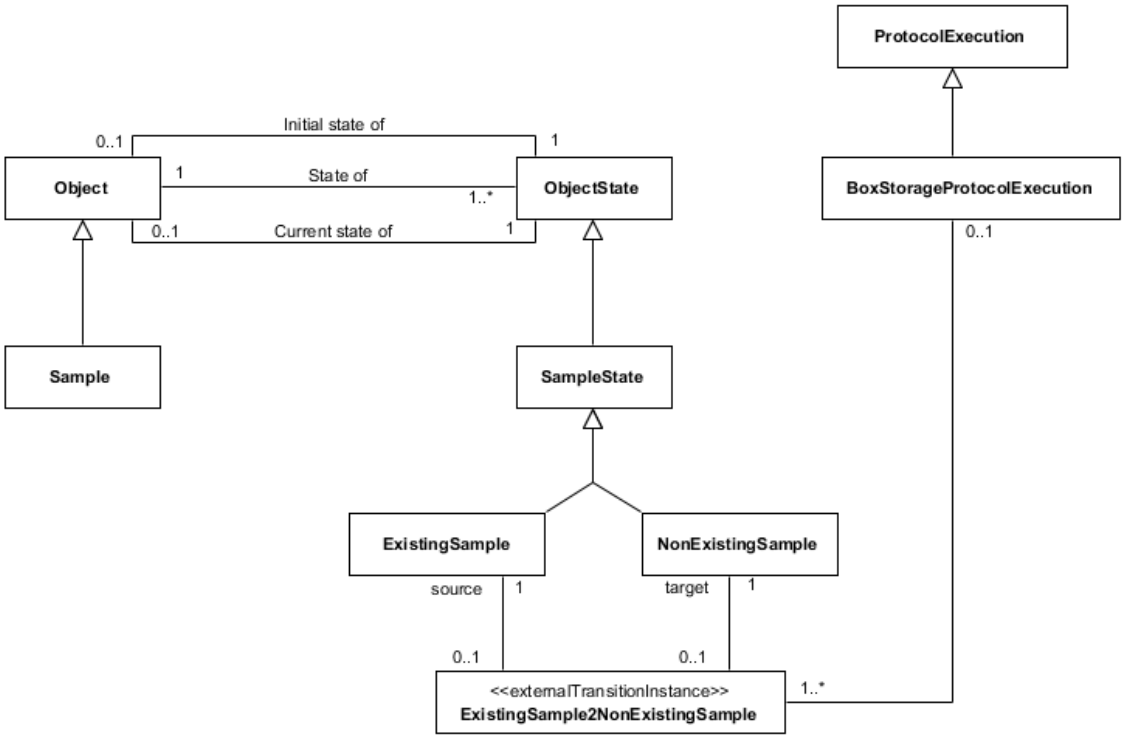


Figura 6.11 Extracto del PIM de una base de acontecimientos de *Arasis*

En la estrategia de diseño seguida para especificar el modelo de la base de acontecimientos del sistema *Arasis* se ha contemplado el patrón *Cambio de Estado* (ver apartado 2.2); la aplicación de este patrón ha posibilitado que la base de acontecimientos de este sistema pueda registrar la información necesaria como para obtener la traza completa de cambios de estado producidos en un determinado objeto.

El patrón *Cambio de Estado* contempla la existencia de un estado inicial, un estado actual (o activo durante un periodo temporal dado) y un histórico de los estados por los que ha pasado un objeto a lo largo de su vida. El estado inicial es necesario para el *Provenance*, pues el modo de enlazar 'la vida anterior' de un objeto con el resto de estados una vez su estado inicial ha dejado de ser el estado actual, además de facilitar la trazabilidad del objeto en el

sistema. Este patrón corresponde con el representado en la Figura 2.4.

Cuando el *Tipo de Efecto* corresponde con un **cambio en los datos** propios de un objeto, el metamodelo contempla la especificación del estado activo (ActiveStateType) en el que se ha podido producir dicho cambio. El cambio en los datos de un objeto que se encuentra en un estado dado se puede representar por medio de una transición interna al estado, puesto que, tal como se describe en UML, una transición interna *"Implies that the Transition, if triggered, occurs without exiting or entering the source State (i.e., it does not cause a state change)"* [54].

Respecto a este *Tipo de Efecto*, es importante señalar que, en un *Modelo de Base de Acontecimientos*, los datos pueden estar vinculados bien directamente a un objeto, bien a uno de sus estados tal y como hemos indicado. En el metamodelo que describimos en esta tesis no hemos considerado las situaciones en las que el cambio se produce en los datos vinculados directamente al objeto. La forma óptima de modelizar estos cambios, manteniendo la traza, es actualmente un tema abierto en la investigación en la que se enmarca esta tesis.

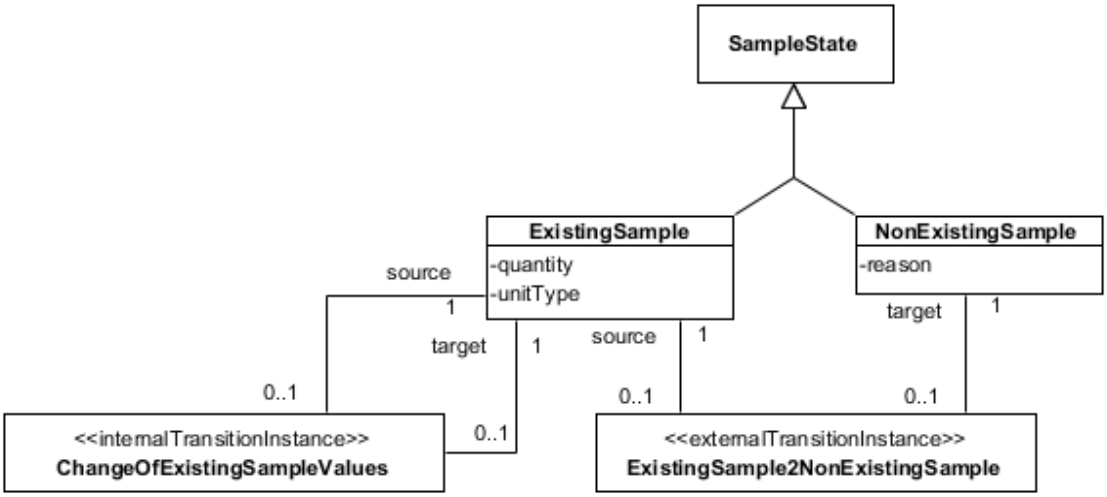


Figura 6.12 Modelización del cambio en los datos de un objeto

En la Figura 6.12 aparece un ejemplo de cómo se ha modelizado un cambio en los datos asociados a un estado. En concreto se puede observar como los tipos de estado *ExistingSample* y *NonExistingSample* tienen una serie de atributos específicos de cada uno de ellos. Un objeto de tipo *Sample* cuyo estado activo sea del tipo *ExistingSample* tendrá registrada una determinada cantidad (quantity) expresada en un tipo de unidad (unitType). Si por la acción de un determinado protocolo, se viera afectado uno de estos dos datos, se desencadenaría la transición interna de tipo *ChangeOfExistingSampleValues* generándose un nuevo estado activo de tipo *ExistingSample* que contendría los nuevos valores. Si en lugar de este efecto se desencadenara la transición de tipo *ExistingSample2NonExistingSample*, el estado de tipo *NonExistingSample* pasaría a ser el estado actual.

6.9 Tipo de Estructura Longitudinal

Finalizamos el recorrido por los diferentes conceptos contemplados en el metamodelo de bases de acontecimientos con la presentación, como parte del mismo, del concepto *Tipo de Estructura Longitudinal* (LongitudinalStructureType). Recordemos que en el Capítulo 2 hemos realizado una exposición sobre el estado del arte del concepto *Estructura Longitudinal*, así como a su interpretación en la investigación en la que se enmarca esta tesis.

El concepto *Tipo de Estructura Longitudinal* (LongitudinalStructureType) se incluye en el metamodelo para hacer posible considerar diferentes tipos de estructuras longitudinales a través de las cuales es posible representar acontecimientos que han tenido lugar a lo largo del tiempo. Este tipo de estructuras, entre las que destacan *Historia* y *Línea de vida* de un objeto propuestas en [24], son fundamentales a la hora de extraer conocimiento y obtener la procedencia de los datos.

Como veremos en el Capítulo 7, uno de los tipos de consultas contemplado en el *Lenguaje de Acceso a Acontecimientos*, denominado *Longitudinal Structure Queries*, posibilita la recuperación de acontecimientos incluidos en estructuras longitudinales como pueden ser los tipos *Historia* y *Línea de Vida* propuestos en [24].

Para concluir este apartado, es importante señalar que, aunque en el ámbito de esta tesis se ha profundizado en el uso de los dos tipos de estructuras longitudinales basadas en acontecimientos, *Historia* y *Línea de Vida*, propuestos en [24] y referidos en el apartado 2.1, el metamodelo admite contemplar otros tipos de estructuras longitudinales que posibiliten realizar diferentes tipos actividades de explotación de la información como análisis de calidad, análisis de eficiencia, procesos de evaluación de la sostenibilidad, etc. Este último caso aplicado, por ejemplo, a la industria alimentaria a través de la inclusión de una traza de la producción, posibilitaría evaluar la *huella ecológica* del procesado de los alimentos.

Capítulo 7. Lenguaje de Acceso a Acontecimientos

7.1 Introducción

En el Capítulo 4 hemos presentando en términos generales el *Lenguaje de Consultas para la Gestión de Acontecimientos* (*Ocurrence Query Language*, *OcQuery Language*; *OcQL*) como una herramienta para posibilitar el tratamiento de la información almacenada en una base de acontecimientos (*Occurrence Base*; *OcBase*). En este capítulo vamos a presentar el *Lenguaje de Acceso a Acontecimientos* (*Occurrence Access Language*; *OcAL*), que ha sido objeto de esta tesis y publicado en [25]. El *Lenguaje de Acceso a Acontecimientos* está dirigido a facilitar el acceso y recuperación de acontecimientos almacenados en una base de acontecimientos.

Como parte del *Lenguaje de Acceso a Acontecimientos* distinguimos entre la variante a la que referimos como 'no generalizada' y la variante generalizada, a las cuales dedicaremos este capítulo. La primera variante, a la que denominamos *Lenguaje No Generalizado de Acceso a Acontecimientos* (*Non Generalized Occurrence Access Language; Non Generalized OcAL*), está dirigida a posibilitar el acceso a los acontecimientos registrados en una base de acontecimientos sin necesidad de tener conocimiento de los conceptos definidos en el *Metamodelo de Bases de Acontecimientos* del que hemos hablado en el Capítulo 6. A diferencia de la primera variante, la segunda, a la que denominamos *Lenguaje Generalizado de Acceso a Acontecimientos* (*Generalized Occurrence Access Language; Generalized OcAL*) incorpora también referencias a los conceptos definidos en el metamodelo permitiendo obtener información acerca de los mismos registrada en el *Repositorio de Metadatos*. El *Repositorio de Metadatos* es uno de los componentes del framework al que hemos dedicado el apartado 3.2.6 y del cual mostraremos un caso de implementación en el apartado 8.5.

En este capítulo explicaremos detalladamente las dos variantes del *Lenguaje de Acceso a Acontecimientos*; en primer lugar, presentaremos el contexto del que se obtendrán los ejemplos mostrados en este capítulo, continuaremos exponiendo los aspectos generales de OcAL y de su sintaxis, y finalizaremos dedicando un apartado a cada una de las variantes del lenguaje. En cada uno de los dos apartados dedicados a describir las variantes del lenguaje, comenzaremos explicando detalladamente la especificación formal de la sintaxis de cada una, y continuaremos realizando su descripción textual poniendo ejemplos en cada caso.

7.2 Sobre los ejemplos de este capítulo

Para acompañar a las explicaciones sobre *Lenguaje de Acceso a Acontecimientos* que realizaremos en este capítulo, a lo largo del mismo incluiremos a modo de ejemplo diferentes consultas y mostraremos los resultados que se obtendrían al ejecutarlas. Estos ejemplos se enmarcan en el contexto de una base de acontecimientos que da servicio a un banco de

muestras biológicas (también denominado biobanco), por lo que corresponden con consultas que podrían ejecutarse sobre la misma; a este respecto, es importante señalar que en este caso estamos hablando de una base de acontecimientos teórica, no sobre la base de acontecimientos del sistema *Arasis* a la que hemos referido en ocasiones a lo largo de este documento de tesis.

En este apartado vamos a realizar una breve especificación de la base de acontecimientos y del repositorio de metadatos en los que se basan los ejemplos de este capítulo.

Acontecimientos

En primer lugar, vamos a mostrar el conjunto de acontecimientos al que referiremos en los ejemplos de este capítulo:

Occurrence IDO	Occurrence	Occurrence Date
occurrence001	(‘bloodSample007’, ‘sampleRecordingProtocolExecution001’, ‘sampleState001’)	2016-03-14 09:55:00
occurrence002	(‘bloodSample007’, ‘sampleStoringProtocolExecution002’, ‘sampleState002’)	2016-03-14 10:00:00
occurrence003	(‘box001’, ‘sampleStoringProtocolExecution002’, ‘boxState002’)	2016-03-14 10:00:01
occurrence004	(‘bloodSample008’, ‘sampleRecordingProtocolExecution002’, ‘sampleState003’)	2017-03-15 09:45:00
occurrence005	(‘bloodSample008’, ‘sampleStoringProtocolExecution003’, ‘sampleState004’)	2017-03-15 09:50:00
occurrence006	(‘box001’, ‘sampleStoringProtocolExecution003’, ‘boxState003’)	2017-03-15 09:50:01
occurrence007	(‘bloodSample008’, ‘sampleAliquotingProtocolExecution005’, ‘sampleState005’)	2017-03-16 10:20:00
occurrence008	(‘bloodSample007’, ‘sampleDestroyingProtocolExecution004’, ‘sampleState006’)	2017-03-22 12:30:00
occurrence009	(‘box001’, ‘sampleDestroyingProtocolExecution004’, ‘boxState004’)	2017-03-22 12:30:01

Tabla 7.1 Conjunto de acontecimientos de ejemplo

En relación a esta tabla, es importante señalar que se contenido es únicamente una forma de representar un conjunto de acontecimientos que no corresponde con la forma en la que se almacenan en una base de acontecimientos.

Recordemos que, según lo indicado en el Capítulo 2, un acontecimiento puede representarse como una tupla de la siguiente forma: (O, Pe, C). Cada uno de los elementos de esta tupla refiere respectivamente a:

1 El objeto O afectado por el acontecimiento.

2 La ejecución de protocolo (Protocol Execution; Pe) que ha afectado al objeto.

3 El cambio C producido sobre el objeto por la ejecución del protocolo.

En el caso de los acontecimientos que hemos mostrado, el cambio correspondería con un cambio de estado del objeto. Para representarlo, en la Tabla 7.1 mostramos para cada acontecimiento el estado destino de la transición.

Teniendo esto presente, la Tabla 7.1 representa a un conjunto de acontecimientos que han afectado a los objetos de tipo *Sample* 'bloodSample007', 'bloodSample008' y al objeto de tipo *Box* 'box001' en unas fechas determinadas (que corresponden con las fechas de registro de los protocolos correspondientes) de la siguiente manera:

- La ejecución de protocolo 'sampleRecordingProtocolExecution001' ha producido la creación del objeto 'bloodSample007' en el estado 'sampleState001'.
- La ejecución de protocolo 'sampleStoringProtocolExecution002' ha producido que la muestra 'bloodSample007' cambie su estado activo al estado 'sampleState002' para señalar que ha quedado almacenada (*stored*).
- La ejecución de protocolo 'sampleStoringProtocolExecution002' también ha producido que el objeto 'box001' cambie su estado activo de 'boxState001' a 'boxState002' debido a la variación del número de huecos libres al almacenar la muestra 'bloodSample007' señalando que el objeto 'box001' ha pasado de estar libre a estar siendo utilizado.
- La ejecución de protocolo 'sampleRecordingProtocolExecution002' ha producido la creación del objeto 'bloodSample008' en el estado 'sampleState003'.
- La ejecución de protocolo 'sampleStoringProtocolExecution003' ha producido que la muestra 'bloodSample008' cambie su estado activo al estado 'sampleState004' para señalar que ha quedado almacenada (*stored*).
- La ejecución de protocolo 'sampleStoringProtocolExecution003' también ha producido que el objeto 'box001' cambie su estado activo de 'boxState002' a 'boxState003' debido a la variación del número de huecos libres al almacenar la muestra 'bloodSample008'. En este caso se partía de un estado en el que se señalaba que el objeto 'box001' estaba siendo utilizado y se llega a un estado equivalente.
- La ejecución de protocolo 'sampleAliquotingProtocolExecution005' ha producido que el objeto 'bloodSample008' cambie su estado activo de 'sampleState004' a 'sampleState005' debido a la preparación de alícuotas a partir de dicha muestra.
- La ejecución de protocolo 'sampleDestroyingProtocolExecution004' ha producido que el objeto 'bloodSample007' cambie su estado activo de 'sampleState002' a 'sampleState006' debido a la destrucción de dicha muestra (recordemos que en una base de acontecimientos debe tratar de que quede evidencia de todo lo sucedido y, por lo tanto, en este caso, la destrucción de una muestra no implica la eliminación de su información de la base de acontecimientos).
- La ejecución de protocolo 'sampleDestroyingProtocolExecution004' también ha producido que el objeto 'box001' cambie su estado activo de 'boxState003' a 'boxState004' debido a la variación del número de huecos libres al destruir la muestra 'bloodSample008'.

En cada uno de los acontecimientos de ejemplo hemos representado los cambios que se han producido en los datos correspondientes:

- Los estados activos de los objetos de tipo *Box* contemplan como dato el número de huecos libres para almacenar muestras.
- Los estados activos de los objetos de tipo *Sample* no contemplan cambios en los datos.

Es importante señalar que consideramos que los acontecimientos aquí referidos han sido producidos por ejecuciones de protocolo realizadas por el usuario 'laboratoryAssistant001'.

Diagramas de estados

Mostramos a continuación los diagramas de estados de los tipos de objeto que aparecen en los ejemplos de este capítulo (*Sample* y *Box*):

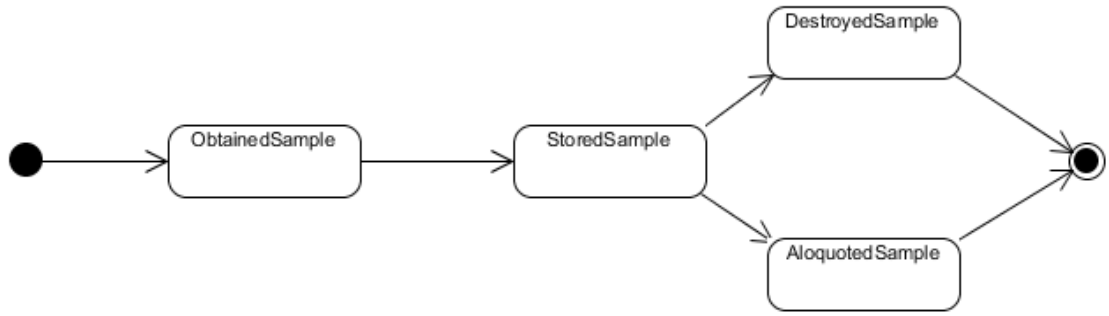


Figura 7.1 Diagrama de estados del tipo de objeto Sample

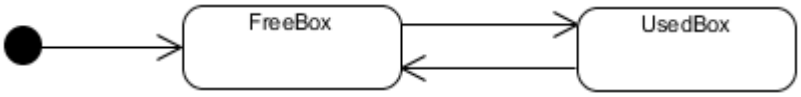


Figura 7.2 Diagrama de estados del tipo de objeto Box

Repositorio de metadatos

Una vez presentados y descritos los acontecimientos que pueblan nuestra base de acontecimientos de ejemplo vamos a mostrar algunos datos que están almacenados en el repositorio de metadatos que describe la estructura de la misma. Recordemos que el *Repositorio de Metadatos* es el componente del framework que almacena la información que describe la estructura de una base de acontecimientos.

La siguiente tabla muestra una parte de los datos almacenados en el repositorio de metadatos que hemos tomado como referencia en los ejemplos que presentaremos para explicar la variante generalizada del *Lenguaje de Acceso a Acontecimientos*:

OBJECT TYPE	ACTIVE STATE TYPE	PROTOCOL EXECUTION TYPE	PERFORMER TYPE
Object	*SampleState*	*ProtocolExecution*	*Performer*
Sample	ObtainedSample	SampleRecordingProtocolExecution	UserPerformer
Box	StoredSample	SampleStoringProtocolExecution	SystemPerformer
	AliquotedSample	SampleAliquotingProtocolExecution	
	DestroyedSample	SampleDestroyingProtocolExecution	
	BoxState	RackStoringProtocolExecution	

	FreeBox		
	UsedBox		

Tabla 7.2 Representación en forma tabular del repositorio de metadatos de ejemplo

La primera columna de la siguiente tabla muestra los tipos de elemento registrados en el repositorio de metadatos, para cada uno de ellos, en la segunda columna, se muestran los identificadores de las instancias de los mismos registradas en la base de acontecimientos; como puede observarse, estos identificadores han sido utilizados en la representación de los acontecimientos mostrados en la Tabla 7.1:

Tipo de elemento	**Identificador de instancia**
Sample	bloodSample007
	bloodSample008
Box	box001
ObtainedSample	sampleState001
StoredSample	sampleState002
	sampleState004
AliquotedSample	sampleState005
DestroyedSample	sampleState006
FreeBox	boxState001
UsedBox	boxState002
	boxState003
	boxState004
SampleRecordingProtocolExecution	sampleRecordingProtocolExecution001
	sampleRecordingProtocolExecution002
SampleStoringProtocolExecution	sampleStoringProtocolExecution002
	sampleStoringProtocolExecution003
SampleAliquotingProtocolExecution	sampleAliquotingProtocolExecution005
SampleDestroyingProtocolExecution	sampleDestroyingProtocolExecution004
UserPerformer	laboratoryAssistant001

Tabla 7.3 Tipos de elemento y sus instancias

7.3 Sintaxis general del lenguaje

La sintaxis del *Lenguaje de Acceso a Acontecimientos* introduce diferentes variaciones en función de si ésta refiere a la variante no generalizada o a la variante generalizada.

En este apartado, vamos a presentar los elementos que son comunes en la sintaxis de las dos variantes del *Lenguaje de Acceso a Acontecimientos*; en el caso de elementos no comunes, indicaremos donde corresponda de que variante del lenguaje son propios. La sintaxis formal

de cada una de las variantes del lenguaje será mostrada en los siguientes apartados.

7.3.1 Estructura de una consulta

Todas las consultas del *Lenguaje de Acceso a Acontecimientos* tienen una estructura común que puede esquematizarse de la siguiente forma:

1 Se inician con una cláusula `SELECT`, para referir a la información que se desea recuperar de la base de acontecimientos.

Por ejemplo, a través de esta cláusula es posible indicar que se desea obtener información propia de objetos, estados, ejecuciones de protocolo, etc. o indicar que se desea representar un determinado tipo de estructura longitudinal como la *Historia* o la *Línea de Vida* de acontecimientos ocurridos a un determinado objeto.

2 La cláusula `SELECT` puede ir seguida opcionalmente por una cláusula `RELATED TO` para concretar el tipo de información a recuperar.

Por ejemplo, a través de esta cláusula es posible indicar que se desea recuperar la información generada por una determinada ejecución de protocolo, por un determinado usuario que ha realizado diversas ejecuciones de protocolos, que es propia de un determinado *Tipo de Objeto*, etc.

3 A continuación, opcionalmente, se pueden incluir las cláusulas `BETWEEN`, `ON`, `BEFORE` y `AFTER` que posibilitan definir condiciones para delimitar el conjunto de información a recuperar a un periodo de tiempo (`BETWEEN`) entre dos fechas, a una determinada fecha (`ON`), o a un periodo de tiempo anterior (`BEFORE`) o posterior (`AFTER`) a un determinado acontecimiento.

4 En función del subtipo de consulta, pueden cerrarse con una cláusula `ORDER BY` para definir la forma de ordenación, bien ascendente (`ASC`) o descendente (`DESC`).

En notación BNF (*Backus–Naur Form*; *Backus Normal Form*), esta sintaxis puede representarse de la siguiente forma:

```
<OcAL_generic_query> ::=
SELECT <select_clause_expression>
[RELATED TO <related_to_clause_expression>]
[BETWEEN <date> AND <date> | ON <date>]
[BEFORE <before_clause_expression> | AFTER <after_clause_expression>]
[ORDER BY <order_by_clause_expression>];
```

7.3.2 Condiciones

El *Lenguaje de Acceso a Acontecimientos*, a través de las siguientes cláusulas, permite introducir condiciones para delimitar el conjunto de información a recuperar:

Cláusula	Descripción	Ejemplo
BETWEEN …	Obtener información entre un rango de fechas	BETWEEN '2016-01-01 00:00:00' AND

AND	(inclusive).	'2016-12-31 23:59:59'
ON	Obtener información en una fecha determinada.	ON '2016-08-16'
BEFORE	Obtener información previa a un acontecimiento.	BEFORE 'occurrence010'
AFTER	Obtener información posterior a un acontecimiento.	AFTER 'occurrence010'

Tabla 7.4 Condiciones de las consultas OcAL

Las fechas incluidas en las cláusulas `BETWEEN … AND` y `ON` refieren a la fecha de ocurrencia de los acontecimientos. Por lo general, la fecha de ocurrencia de un acontecimiento suele corresponder con las fechas de ejecución del protocolo que lo ha generado, aunque esta es una de las cuestiones que pueden verse afectadas en función de la estrategia de diseño seleccionada.

7.3.3 Operadores lógicos

El *Lenguaje de Acceso a Acontecimientos* contempla tres operadores lógicos. Los operadores `AND` y `OR` se utilizan como parte de la cláusula `RELATED TO` para poder referir a múltiples elementos. El operador `NOT` se utiliza para negar condiciones definidas tanto en la cláusula `RELATED TO` como en las cláusulas `BETWEEN … AND` y `ON`.

Operador	Descripción	Ejemplo
AND	Dadas dos condiciones separadas por el operador AND, es necesario que se cumplan ambas.	RELATED TO PROTOCOL EXECUTION 'sampleAliquotingProtocolExecution002' AND PERFORMER 'userPerformer001'
OR	Dadas dos condiciones separadas por el operador OR, es suficiente con que se cumpla una de las dos.	RELATED TO PROTOCOL EXECUTION 'sampleAliquotingProtocolExecution002' OR PERFORMER 'userPerformer001'
NOT	Niega la condición a la que precede si se utiliza como parte de la cláusula RELATED TO. Niega a la cláusula a la que precede cuando se declara antes de las cláusulas BETWEEN y ON.	RELATED TO PROTOCOL EXECUTION NOT 'sampleAliquotingProtocolExecution002' NOT BETWEEN '2016-01-01 00:00:00' AND '2016-12-31 23:59:59' NOT ON '2016-08-16'

Tabla 7.5 Operadores lógicos de las consultas OcAL

Como se ha podido observar, el uso del operador `NOT` no se ha contemplado como complemento de las cláusulas `BEFORE` y `AFTER`. El motivo es la simplicidad del lenguaje ya que la cláusula `BEFORE` puede utilizarse como negación de la cláusula `AFTER` y viceversa.

7.3.4 Instancias y tipos

Como veremos más adelante, las diferentes cláusulas de una consulta del *Lenguaje de Acceso a Acontecimientos* pueden referir tanto a instancias de elementos como a clases o tipos de elementos. Hablamos de *Clase* o *Tipo* cuando referimos a un elemento especificado en un

modelo de base de acontecimientos; hablamos de *Instancia* de un elemento para referir a información concreta de un determinado tipo registrada en una base de acontecimientos.

Por ejemplo, podríamos decir *Sample* para referir a una clase declarada en el modelo de la base de acontecimientos de un biobanco, y *bloodSample007* para referir a una instancia de dicha clase.

Cuando en una consulta se refiere a clases declaradas en un determinado modelo de base de acontecimientos, estas se declaran sin entrecomillar, por ejemplo: *Sample*. Por otra parte, cuando en una consulta se refiere a instancias de clases, estas son referidas a través de su identificador en el sistema, por ejemplo: 'bloodSample007'.

7.3.5 Ordenación

En relación a la ordenación de los resultados, en el *Lenguaje de Acceso a Acontecimientos* podemos encontrar dos posibles comportamientos en función del tipo de consulta ejecutada:

1 Si la consulta posibilita recuperar acontecimientos incluidos en una estructura longitudinal, los resultados siempre se obtendrán ordenados cronológicamente dado que el tiempo es la unidad de ordenación natural en este tipo de estructuras. A este tipo de consultas las denominamos *Longitudinal Structure Queries*, tal y como veremos más adelante.

2 Si la consulta se utiliza para recuperar acontecimientos que no constituyen una estructura longitudinal o elementos propios de los mismos, los resultados se mostrarán por defecto sin seguir un criterio concreto de ordenación. A este tipo de consultas las denominamos *Occurrences Queries*, tal y como veremos más adelante.

Para este último caso, el *Lenguaje de Acceso a Acontecimientos*, dispone de la cláusula `ORDER BY` junto con dos posibles palabras reservadas: `ASC` para obtener los resultados ordenados de forma ascendente en función del criterio que se indique, y `DESC` para obtener los resultados ordenados de forma descendente. El uso de las palabras reservadas `ASC` y `DESC` es opcional, por tanto, si no se especifica ninguna de ellas como parte de la cláusula `ORDER BY`, se aplica el criterio de ordenación por defecto que es ascendente.

7.4 Lenguaje no generalizado de acceso a acontecimientos

El *Lenguaje No Generalizado de Acceso a Acontecimientos* (*Non Generalized Occurrence Access Language; Non Generalized OcAL*) es un subconjunto de consultas dedicado a posibilitar el acceso a los acontecimientos almacenados en una base de acontecimientos sin necesidad de tener conocimiento de los conceptos definidos en el metamodelo; es decir se restringen a los niveles M0 y M1 de UML representados en la siguiente figura:

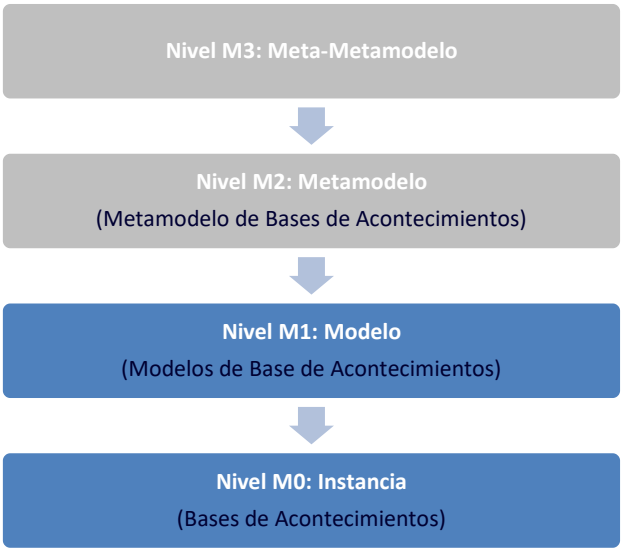


Figura 7.3 Niveles de UML abarcados por OcAL no generalizado

Dentro de esta variante del lenguaje entraría, por ejemplo, una consulta para obtener la línea de vida de un objeto, o una consulta para obtener los estados implicados en la ejecución de un protocolo.

7.4.1 Introducción a la sintaxis

En este apartado vamos a introducir la sintaxis del *Lenguaje No Generalizado de Acceso a Acontecimientos* refiriendo en cada caso a la sintaxis que corresponda utilizando la notación BNF. La sintaxis completa se incluye en el apartado 7.4.4 y la forma de uso de la notación BNF en el Anexo A.

La sintaxis del *Lenguaje No Generalizado de Acceso a Acontecimientos* tiene una estructura base en la que se introducen variaciones en función de los dos tipos de consultas contemplados que explicaremos con más detalle en los apartados 7.4.2 y 7.4.3. La forma de especificar esto en notación BNF es la siguiente:

```
<OccurrenceQL> ::= <longitudinal_structure_query> | <occurrences_query>;
```

1 Consultas para obtener acontecimientos incluidos en una estructura longitudinal (*Longitudinal Structure Queries*).

A continuación, mostramos una versión simplificada de la sintaxis de este tipo de sentencias, en el apartado 7.4.4 completaremos la definición:

```
<longitudinal_structure_query> ::=
SELECT [INITIAL OCCURRENCE OF | FINAL OCCURRENCE OF]
 <longitudinal_structure_type>
[RELATED TO <structure_element> <structure_element_instance> [OF TYPE
<type_name>]]
[BETWEEN <date> AND <date> | ON <date>]
[BEFORE <occurrence_instance> | AFTER <occurrence_instance>];
```

2 Consultas para obtener acontecimientos o elementos de acontecimientos que no

constituyen una estructura longitudinal (*Occurrences Queries*).

A continuación, mostramos una versión simplificada de la sintaxis de este tipo de sentencias, en el apartado 7.4.4 completaremos la definición:

```
<occurrences_query> ::=
SELECT [INITIAL | FINAL] OCCURRENCE [OF TYPE <occurrence_type>] |
<occurrence_element> [OF TYPE <type_name>]
[RELATED TO <occurrence_element> <occurrence_element_instance> [OF TYPE
<type_name>]]
[BETWEEN <date> AND <date> | ON <date>]
[BEFORE <occurrence_instance> | AFTER <occurrence_instance>]
[ORDER BY <occurrence_elements_with_order_type>];
```

La variación más sustancial en la sintaxis de ambos tipos de consulta se encuentra en la especificación de la cláusula `SELECT`, a través de la cual se indica la información que se desea obtener:

- En las consultas del primer tipo, a través de la cláusula `SELECT`, se refiere al tipo de estructura longitudinal a representar (`<longitudinal_structure_type>`) y, opcionalmente a los acontecimientos inicial o final de la misma utilizando respectivamente las palabras reservadas `INITIAL OCCURRENCE OF` y `FINAL OCCURRENCE OF`.

  Los tipos de estructura longitudinal contemplados en una determinada implementación del *Framework OcQF* dependen de la estrategia de diseño seleccionada y de cómo se defina la dimensión del lenguaje que refiere a las mismas tal y como hemos indicado en el apartado 4.5.

- En las consultas del segundo tipo es posible referir a acontecimientos y a sus elementos.

  La expresión `[INITIAL | FINAL] OCCURRENCE [OF TYPE <occurrence_type>]` especifica la forma en la que es posible referir a acontecimientos como parte de la cláusula `SELECT`. La expresión alternativa `<occurrence_element> [OF TYPE <type_name>]` especifica cómo referir a elementos de un acontecimiento (*Objeto*, *Efecto*, *Ejecución de Protocolo*). A través del símbolo `<occurrence_element>` estamos introduciendo la posibilidad de referir a elementos propios de un acontecimiento utilizando las palabras reservadas del lenguaje (`OBJECT`, `ACTIVE STATE` y `PROTOCOL EXECUTION`) siempre que la estrategia de diseño seleccionada lo posibilite; a través del símbolo `<type_name>`, tal y como indicaremos más adelante, estamos introduciendo la posibilidad de indicar de qué clase de las especificadas en un modelo de base de acontecimientos es el elemento `<occurrence_element>`.

Esta diferenciación en la cláusula `SELECT` viene motivada por el hecho de que las estructuras longitudinales basadas en acontecimientos están dirigidas a mostrar acontecimientos ocurridos a un determinado objeto o conjunto de objetos, por lo que entendemos que no es procedente referir a información parcial de las mismas.

Siguiendo con las consultas del segundo tipo, a través de la cláusula `RELATED TO` se establecen los criterios de selección de los acontecimientos sobre los que se pide la información. Por ejemplo, a través de esta cláusula podemos especificar que la información a obtener debe referir a acontecimientos producidos sobre objetos de un determinado tipo, a acontecimientos generados por un determinado tipo de ejecución de protocolo, etc. Como parte de esta cláusula, la expresión `<occurrence_element> <occurrence_element_instance> [OF TYPE <type_name>]` especifica cómo hay que referir a los elementos propios de un acontecimiento; el símbolo `<occurrence_element>` especifica al

elemento al que se referirá utilizando las palabras reservadas del lenguaje (`OBJECT`, `ACTIVE STATE` y `PROTOCOL EXECUTION`); el símbolo `<type_name>` se introduce para permitir especificar de qué clase declarada en un modelo de base de acontecimientos es el elemento indicado; el símbolo `<occurrence_element_instance>` se introduce para permitir referir a una instancia concreta de la clase indicada a través del símbolo `<type_name>`.

Como hemos podido ver, las cláusulas `SELECT` y `RELATED TO` de una consulta pueden referir tanto a instancias de elementos como a clases o tipos de elementos. Hablamos de *Clase* o *Tipo* cuando referimos a un elemento especificado en un modelo de base de acontecimientos y para ello hemos incluido el símbolo `<type_name>` en la sintaxis del lenguaje; como contraposición, hablamos de *Instancia* de un elemento para referir a información concreta registrada en una base de acontecimientos. En las expresiones `<occurrence_element> [OF TYPE <type_name>]` y `<occurrence_element> [NOT] <occurrence_element_instance> [OF TYPE <type_name>]` se contemplan respectivamente ambos conceptos. Por ejemplo, en una consulta OcAL, el símbolo `<type_name>` puede tomar valor *Sample* para referir a una clase declarada en el modelo de la base de acontecimientos de un biobanco; así pues, al declarar `RELATED TO OBJECT 'bloodSample007' OF TYPE Sample`, estaríamos refiriendo a una instancia de la clase *Sample* identificada como '*bloodSample007'*; al declarar `SELECT OBJECT OF TYPE Sample`, estaríamos indicando que deseamos obtener los objetos de la clase declarada en el modelo como *Sample*.

Veamos a continuación algunas consultas de ejemplo para mostrar lo que hemos descrito hasta ahora.

Ejemplo:

La siguiente consulta permite obtener los objetos de tipo *Sample* afectados por la ejecución de protocolo 'sampleStoringProtocolExecution002' de tipo *SampleAliquotingProtocolExecution*:

```
SELECT OBJECT OF TYPE Sample
RELATED TO PROTOCOL EXECUTION 'sampleStoringProtocolExecution002'
    OF TYPE SampleAliquotingProtocolExecution
```

Consulta 7.1

Ejemplo:

La siguiente consulta permite obtener las ejecuciones de protocolo de tipo *SampleAliquotingProtocolExecution* que han afectado a la muestra 'bloodSAmple007':

```
SELECT PROTOCOL EXECUTION OF TYPE SampleAliquotingProtocolExecution
RELATED TO OBJECT 'bloodSample007 OF TYPE Sample
```

Consulta 7.2

Prosigamos hablando de la cláusula `RELATED TO`; al inicio de este apartado hemos podido observar que al definir la sintaxis de esta cláusula para las consultas del primer tipo se ha utilizado la expresión `<structure_element> [NOT] <structure_element_instance> [OF TYPE <type_name>]`, mientras que para la sintaxis en el caso de las consultas del segundo tipo se ha utilizado la expresión `<occurrence_element> [NOT] <occurrence_element_instance> [OF TYPE <type_name>]`. El motivo es que, tal y como hemos indicado, las consultas del primer tipo van dirigidas a obtener información de estructuras longitudinales y para ello es necesario que refieran a elementos propios de las mismas (`<structure_element_xxxxxxxx>`). Por el contrario, las consultas del segundo tipo

van dirigidas a obtener información de acontecimientos, por lo que sus símbolos los especificamos como <occurrence_element_xxxxxxx> al referir a elementos propios de los mismos. Veamos a continuación un ejemplo de consulta del primer tipo:

Ejemplo:

La siguiente consulta permite obtener la línea de vida de la instancia de un contenedor de muestras identificado como 'box001':

SELECT LIFELINE
RELATED TO OBJECT 'box001' OF TYPE Box
Consulta 7.3

La cláusula BETWEEN … AND y ON es común a los dos tipos de consulta indicados, a través de ella es posible definir condiciones para acotar los resultados obtenidos a un determinado periodo temporal o a un momento temporal dado respectivamente. El símbolo <date> incluido como parte de la especificación de esta cláusula hace referencia a la fecha de ocurrencia de los acontecimientos. Por lo general, la fecha de ocurrencia de un acontecimiento suele corresponder con las fechas de ejecución del protocolo que lo ha generado, aunque esta es una de las cuestiones que pueden verse afectadas en función de la estrategia de diseño seleccionada.

Las cláusulas BEFORE y AFTER también son comunes a ambos tipos de consulta y, a través de ellas, es posible acotar los resultados obtenidos a acontecimientos ocurridos con anterioridad o posterioridad al acontecimiento indicado. El símbolo <occurrence_instance> incluido como parte de la especificación de estas cláusulas hace referencia a la instancia del acontecimiento que se desea establecer como cota de la información a obtener.

Por último, la cláusula ORDER BY, que es específica de las consultas para obtener acontecimientos que no constituyen una estructura longitudinal, posibilita establecer los criterios de ordenación (ascendente -ASC- y descendente -DESC-) de los resultados. La sintaxis no contempla el uso de esta cláusula como parte de las consultas para obtener acontecimientos incluidos en una estructura longitudinal dado que la ordenación natural de este tipo de estructuras es el tiempo, por lo que los resultados siempre se obtienen ordenados de forma cronológica.

Sobre la optimización y el rendimiento

Como se ha podido observar en la sintaxis que acabamos de mostrar, la declaración de algunos elementos en una consulta del *Lenguaje de Acceso a Acontecimientos* es opcional (lo cual, en notación BNF, se denota encerrando el correspondiente símbolo o expresión entre corchetes), esto posibilita construir consultas más simples.

Ejemplo:

En la siguiente consulta que posibilita obtener los estados activos de los acontecimientos producidos por las ejecuciones de protocolos realizadas por el usuario 'laboratoryAssistant001' se ha omitido la palabra reservada OF TYPE en la cláusula RELATED TO:

SELECT ACTIVE STATE
RELATED TO PERFORMER 'laboratoryAssistant001'
Consulta 7.4

Al omitir la referencia al tipo de ejecutor (a la clase declarada en el modelo de la que es instancia 'laboratoryAssistant001') en la consulta anterior se ha logrado una consulta más simple; no obstante, hay que tener presente que la ejecución de dicha consulta producirá resultados de forma menos eficiente dado que:

1 El traductor deberá determinar los diferentes tipos de ejecutor definidos en el *Repositorio de Metadatos*.

2 La consulta resultante del proceso de traducción, en función de la estructura de la base de acontecimientos, podría llegar a ser más compleja y por tanto más costosa de ejecutar por el sistema de persistencia dado que a través de ella se deberá determinar de qué tipo de ejecutor es instancia 'laboratoryAssistant001'.

Ejemplo:

La siguiente consulta es el resultado de modificar la Consulta 7.4 para mejorar el rendimiento incluyendo la palabra reservada `OF TYPE` e indicando el tipo de ejecutor de protocolo:

```
SELECT ACTIVE STATE
RELATED TO PERFORMER 'laboratoryAssistant001' OF TYPE UserPerformer
```

Consulta 7.5

Acabamos de introducir algunos conceptos sobre el rendimiento del lenguaje, aspecto que se retomará más adelante.

7.4.2 Consultas para obtener acontecimientos incluidos en una estructura longitudinal

Este tipo de consultas (*Longitudinal Structure Queries*) se utiliza para recuperar acontecimientos como parte de una estructura longitudinal como pueden ser la *Historia* y la *Línea de Vida*.

Al declarar consultas de este tipo, es necesario especificar tanto el tipo de estructura longitudinal a reproducir (por ejemplo, *Historia* o *Línea de Vida*) como las instancias relacionadas con la estructura cuya información se desea obtener (por ejemplo, una instancia podría ser el objeto identificado como 'bloodSample007'). Opcionalmente, también se puede indicar el tipo de instancia ('bloodSample007', como veremos en los ejemplos que iremos mostrando, sería una instancia de un objeto de tipo *Sample*).

La cláusula SELECT

La cláusula `SELECT` se utiliza para indicar la información que se desea obtener; en el caso de las consultas para obtener acontecimientos como parte de una estructura longitudinal, en ella se refiere al tipo de estructura longitudinal que se quiere reproducir.

En los ejemplos que vamos a ir mostrando a continuación, la cláusula `SELECT` irá acompañada bien de la palabra reservada `HISTORY` cuando declaremos una consulta para obtener una estructura longitudinal de tipo *Historia*, bien de la palabra reservada `LIFELINE` cuando declaremos una consulta para obtener una estructura longitudinal de tipo *Línea de Vida*.

Aunque ya hemos hablado de ello previamente, puede llamar la atención que ni `HISTORY` ni `LIFELINE` hayan sido especificadas como parte de la sintaxis del lenguaje; esto sucede porque

los tipos de estructura longitudinal son elementos cuyo uso depende de la estrategia de diseño seleccionada y de la adecuación de una de las dimensiones del lenguaje (la que refiere a los tipos de estructuras longitudinales utilizadas) que se han dejado abiertas. Esto quiere decir que, en otros casos, podrían seleccionarse estructuras longitudinales diferentes que habría que implementar de una forma específica y que deberían ser referidas utilizando sus propias palabras reservadas. Una de las premisas del *Lenguaje de Consultas para la Gestión de Acontecimientos* (*OcQL*) es la compatibilidad con cualquier estrategia de diseño basada en acontecimientos y, por ello, `HISTORY` y `LIFELINE` son términos tratados como palabras reservadas del lenguaje tal y como hemos podido ver en el apartado 4.5.

Ejemplo

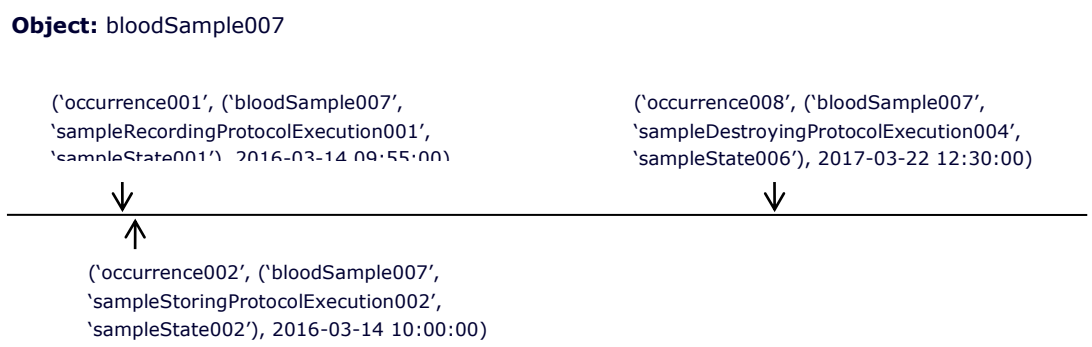
La siguiente consulta incluye la palabra reservada `LIFELINE` para obtener la línea de vida de la instancia de una muestra identificada como 'bloodSample007' registrada en la base de acontecimientos que da servicio a un biobanco:

SELECT LIFELINE
RELATED TO OBJECT 'bloodSample007' OF TYPE Sample
Consulta 7.6

Tomando como referencia el conjunto de acontecimientos mostrados en la Tabla 7.1, esta consulta permitiría obtener una línea de vida formada por los siguientes acontecimientos:

LIFELINE
OCURRENCE
('occurrence001', ('bloodSample007', 'sampleRecordingProtocolExecution001', 'sampleState001'), 2016-03-14 09:55:00)
('occurrence002', ('bloodSample007', 'sampleStoringProtocolExecution002', 'sampleState002'), 2016-03-14 10:00:00)
('occurrence008', ('bloodSample007', 'sampleDestroyingProtocolExecution004', 'sampleState006'), 2017-03-22 12:30:00)

La información devuelta por la consulta anterior se podría representar gráficamente como una línea de vida de la siguiente manera:



Recordemos, según hemos ido explicando en los capítulos anteriores, que al referir a "instancia de una muestra" en la descripción de la Consulta 7.6, hemos pretendido indicar que la muestra 'bloodSample007' es una instancia de la clase que se ha denominado *Sample* en el modelo de la base de acontecimientos que, a su vez, es una instancia de la clase ObjectType definida en el metamodelo (aunque en este caso, como hemos indicado, no corresponda enunciarlo explícitamente en la consulta).

Ejemplo

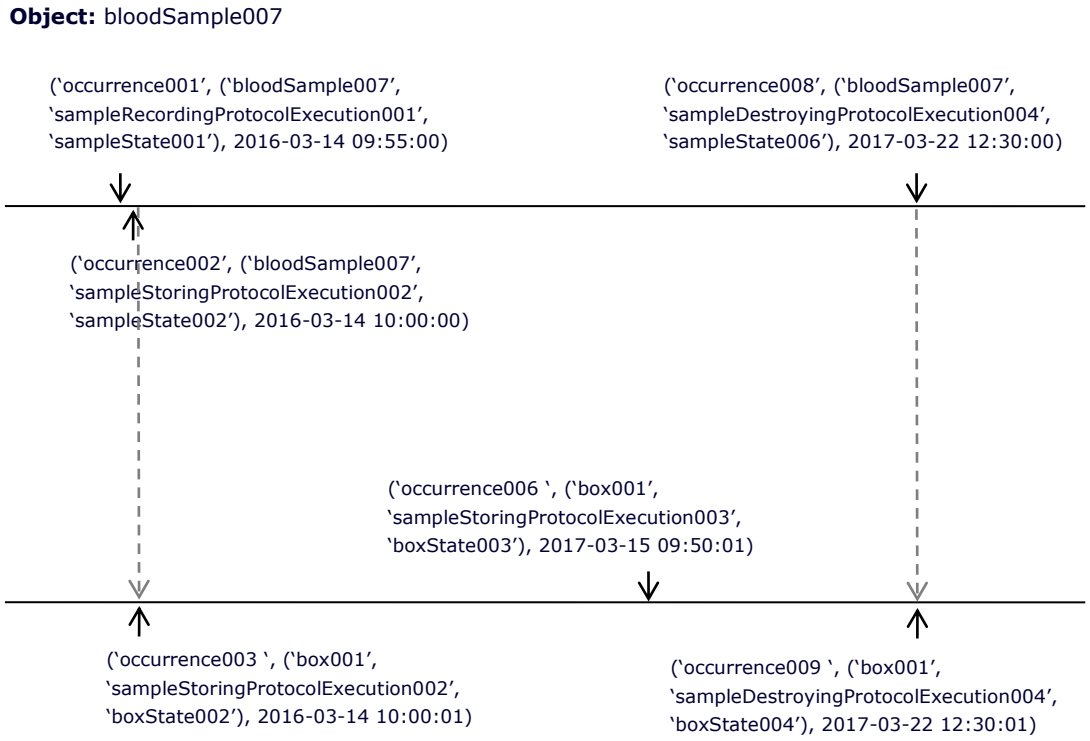
La siguiente consulta incluye la palabra reservada `HISTORY` para obtener la historia de la instancia de una muestra identificada como 'bloodSample007' registrada en la base de acontecimientos que da servicio a un biobanco:

SELECT HISTORY
RELATED TO OBJECT 'bloodSample007' OF TYPE Sample
Consulta 7.7

Al ejecutar la consulta anterior obtendríamos los siguientes resultados:

HISTORY
OCURRENCE
('occurrence001', ('bloodSample007', 'sampleRecordingProtocolExecution001', 'sampleState001'), 2016-03-14 09:55:00)
('occurrence002', ('bloodSample007', 'sampleStoringProtocolExecution002', 'sampleState002'), 2016-03-14 10:00:00)
('occurrence003 ', ('box001', 'sampleStoringProtocolExecution002', 'boxState002'), 2016-03-14 10:00:01)
('occurrence006 ', ('box001', 'sampleStoringProtocolExecution003', 'boxState003'), 2017-03-15 09:50:01)
('occurrence008', ('bloodSample007', 'sampleDestroyingProtocolExecution004', 'sampleState006'), 2017-03-22 12:30:00)
('occurrence009 ', ('box001', 'sampleDestroyingProtocolExecution004', 'boxState004'), 2017-03-22 12:30:01)

La información devuelta por la consulta anterior se podría representar gráficamente de la siguiente manera:



Tal y como hemos indicado en el apartado 2.1, la *Historia* de un objeto refiere al conjunto formado por todos los acontecimientos ocurridos a dicho objeto y por los acontecimientos ocurridos sobre los objetos asociados a él; es decir, sabiendo que la muestra 'bloodSample007' se encuentra almacenada en la caja 'box001', la Consulta 7.7 también muestra los acontecimientos ocurridos a la caja 'box001'.

Para este tipo de consultas, opcionalmente, es posible definir condiciones específicas de búsqueda que posibilitan referir únicamente al acontecimiento inicial o al acontecimiento final del tipo de estructura longitudinal indicado, para ello se utilizan respectivamente las palabras reservadas `INITIAL OCCURRENCE OF` y `FINAL OCCURRENCE OF` como parte de la cláusula `SELECT`.

Ejemplo

La siguiente consulta, resultante de modificar la Consulta 7.6 para incluir la palabra reservada `INITIAL OCCURRENCE OF`, posibilita obtener el acontecimiento que representa el momento en el que se originó la muestra 'bloodSample007' en el sistema (es decir, el acontecimiento inicial de la línea de vida del objeto de tipo *Sample* identificado como 'bloodSample007'):

```
SELECT INITIAL OCCURRENCE OF LIFELINE
RELATED TO OBJECT 'bloodSample007' OF TYPE Sample
```

Consulta 7.8

La ejecución de la consulta anterior permitiría obtener el siguiente acontecimiento que forma parte de la línea de vida del objeto 'bloodSample007':

LIFELINE
OCURRENCE
(‘occurrence001’, (‘bloodSample007’, ‘sampleRecordingProtocolExecution001’, ‘sampleState001’), 2016-03-14 09:55:00)

Ejemplo

En la siguiente consulta se incluyen las palabras reservadas `FINAL OCCURRENCE OF` e `HISTORY` para poder obtener el último acontecimiento del que hay constancia en el sistema que ha afectado a la muestra ‘bloodSample7’ (es decir, el acontecimiento final de la historia del objeto de tipo *Sample* identificado como ‘bloodSample007’):

SELECT FINAL OCCURRENCE OF HISTORY
RELATED TO OBJECT ‘bloodSample007’ OF TYPE Sample
Consulta 7.9

Al ejecutar esta consulta, se obtendría el último acontecimiento sucedido en la historia de la muestra ‘bloodSample007’:

HISTORY
OCURRENCE
(‘occurrence009 ‘, (‘box001’, ‘sampleDestroyingProtocolExecution004’, ‘boxState004’), 2017-03-22 12:30:01)

Debido al comportamiento de la estructura longitudinal de tipo *Historia* que hemos mencionado anteriormente, puede llamar la atención que el resultado obtenido no refiera al objeto ‘bloodSample007’; esto se debe a que el último acontecimiento de la *Historia* de dicho objeto ha ocurrido no sobre el mismo, sino sobre uno de los objetos asociados a él, en este caso la caja ‘box001’.

La cláusula RELATED TO

A través de la cláusula `RELATED TO` es posible establecer los criterios de selección de los acontecimientos a mostrar en el tipo de estructura longitudinal indicada en la cláusula `SELECT`.

En los ejemplos mostrados hasta ahora hemos podido observar el uso de la cláusula `RELATED TO` para lograr representar estructuras longitudinales formadas por acontecimientos ocurridos sobre instancias de objetos de un determinado tipo; pero el *Lenguaje No Generalizado de Acceso a Acontecimientos* contempla establecer otro tipo de criterios si así se determina a través de la estrategia de diseño.

Ejemplo

La siguiente consulta mostraría las líneas de vida de los objetos afectados por las ejecuciones de protocolo realizadas por el usuario ‘laboratoryAssistant001’:

SELECT LIFELINE

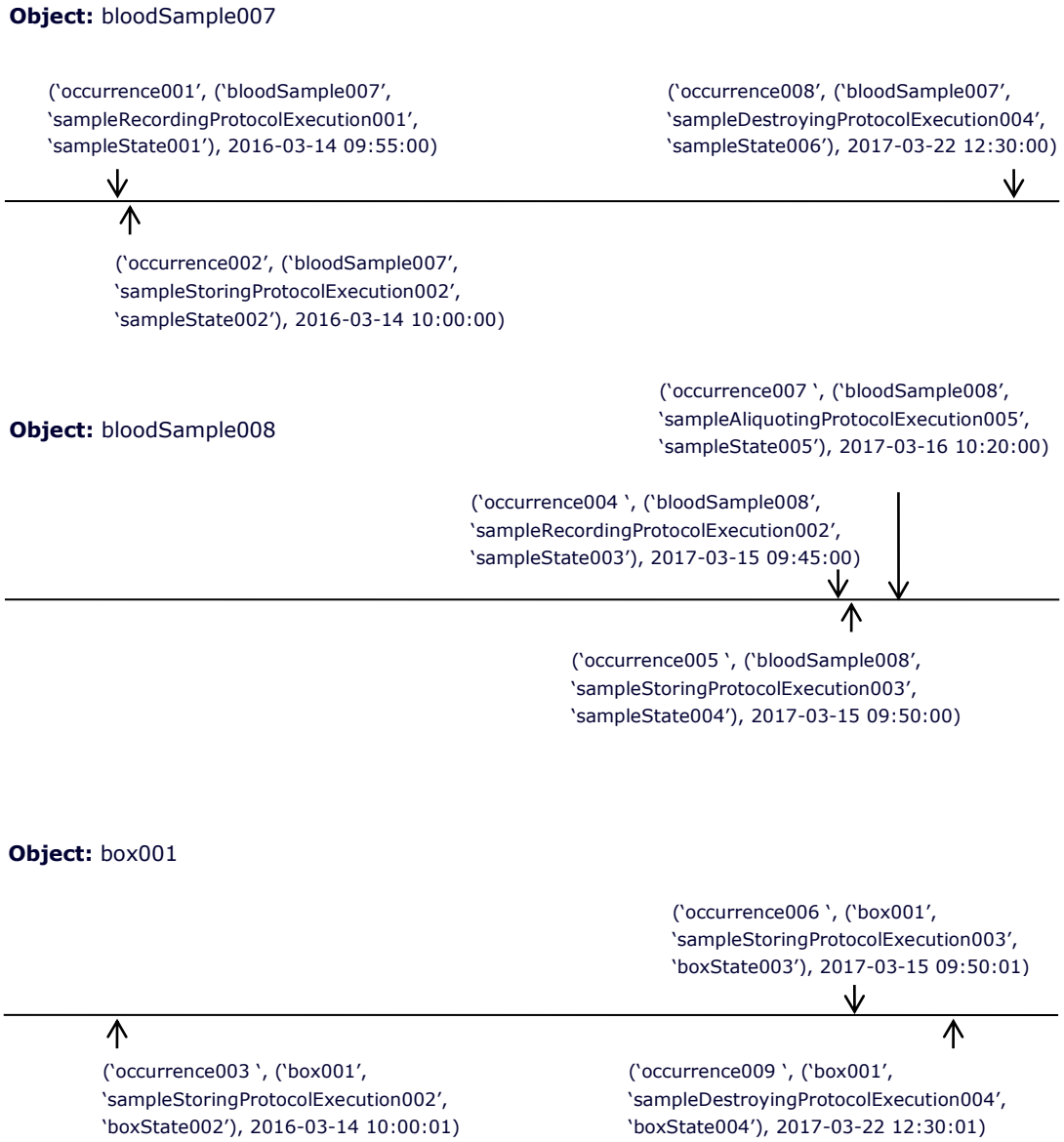
RELATED TO PERFORMER ‘laboratoryAssistant001’ OF TYPE UserPerformer

Consulta 7.10

La ejecución de la consulta anterior permitiría obtener un conjunto de líneas de vida formadas por los siguientes acontecimientos:

LIFELINE
OCURRENCE
(‘occurrence001’, (‘bloodSample007’, ‘sampleRecordingProtocolExecution001’, ‘sampleState001’), 2016-03-14 09:55:00)
(‘occurrence002’, (‘bloodSample007’, ‘sampleStoringProtocolExecution002’, ‘sampleState002’), 2016-03-14 10:00:00)
(‘occurrence008’, (‘bloodSample007’, ‘sampleDestroyingProtocolExecution004’, ‘sampleState006’), 2017-03-22 12:30:00)
(‘occurrence004 ‘, (‘bloodSample008’, ‘sampleRecordingProtocolExecution002’, ‘sampleState003’), 2017-03-15 09:45:00)
(‘occurrence005 ‘, (‘bloodSample008’, ‘sampleStoringProtocolExecution003’, ‘sampleState004’), 2017-03-15 09:50:00)
(‘occurrence007 ‘, (‘bloodSample008’, ‘sampleAliquotingProtocolExecution005’, ‘sampleState005’), 2017-03-16 10:20:00)
(‘occurrence003 ‘, (‘box001’, ‘sampleStoringProtocolExecution002’, ‘boxState002’), 2016-03-14 10:00:01)
(‘occurrence006 ‘, (‘box001’, ‘sampleStoringProtocolExecution003’, ‘boxState003’), 2017-03-15 09:50:01)
(‘occurrence009 ‘, (‘box001’, ‘sampleDestroyingProtocolExecution004’, ‘boxState004’), 2017-03-22 12:30:01)

El conjunto de resultados que acabamos de mostrar, se podría representar gráficamente de la siguiente manera:



Si, como parte de la cláusula `RELATED TO` definimos varias condiciones separadas por el operador lógico `OR` del que hemos hablado en el apartado 7.3.3, podríamos obtener varias líneas de vida. Por ejemplo, las líneas de vida de diferentes objetos.

Ejemplo

La siguiente consulta mostraría las líneas de vida de los objetos 'bloodSample007' y 'bloodSample008':

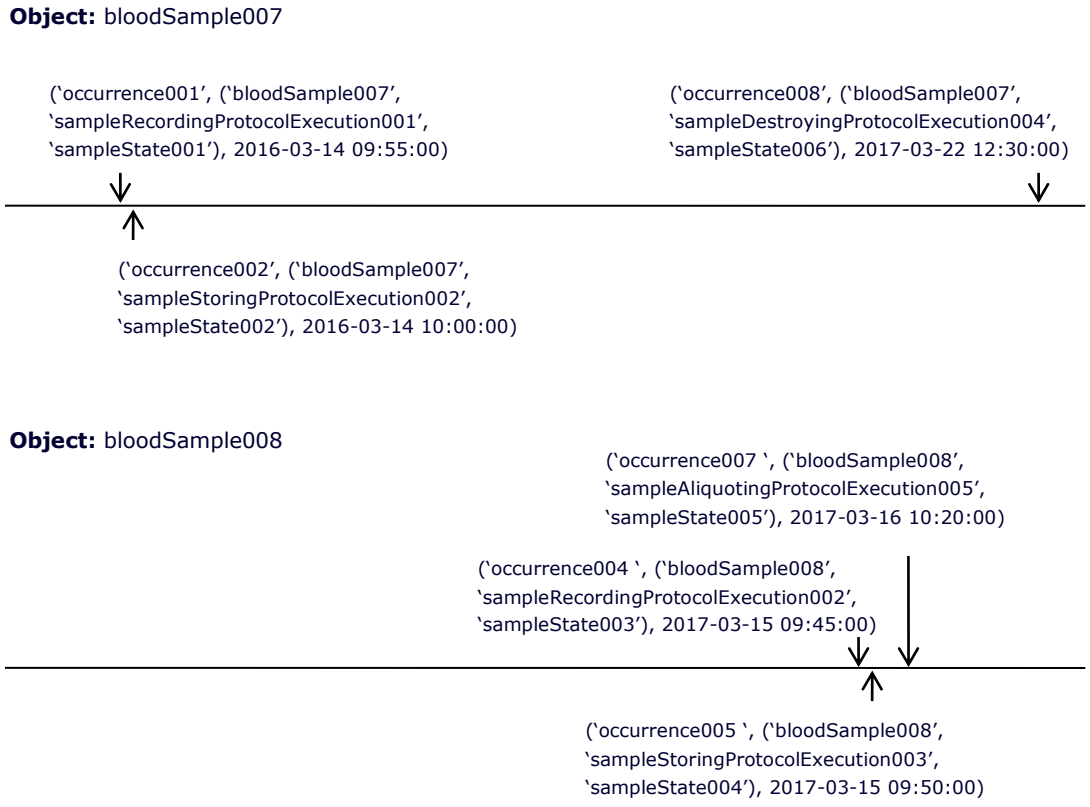
```
SELECT LIFELINE
RELATED TO
    OBJECT 'bloodSample007' OF TYPE Sample
    OR
    OBJECT 'bloodSample008' OF TYPE Sample
```

Consulta 7.11

La ejecución de la consulta anterior permitiría obtener las líneas de vida formadas por los siguientes acontecimientos:

LIFELINE
OCURRENCE
(‘occurrence001’, (‘bloodSample007’, ‘sampleRecordingProtocolExecution001’, ‘sampleState001’), 2016-03-14 09:55:00)
(‘occurrence002’, (‘bloodSample007’, ‘sampleStoringProtocolExecution002’, ‘sampleState002’), 2016-03-14 10:00:00)
(‘occurrence008’, (‘bloodSample007’, ‘sampleDestroyingProtocolExecution004’, ‘sampleState006’), 2017-03-22 12:30:00)
(‘occurrence004 ‘, (‘bloodSample008’, ‘sampleRecordingProtocolExecution002’, ‘sampleState003’), 2017-03-15 09:45:00)
(‘occurrence005 ‘, (‘bloodSample008’, ‘sampleStoringProtocolExecution003’, ‘sampleState004’), 2017-03-15 09:50:00)
(‘occurrence007 ‘, (‘bloodSample008’, ‘sampleAliquotingProtocolExecution005’, ‘sampleState005’), 2017-03-16 10:20:00)

Así pues, el conjunto de resultados que acabamos de mostrar, se podría representar gráficamente de la siguiente manera como dos líneas de vida:



Condiciones

El *Lenguaje de Acceso a Acontecimientos* posibilita establecer condiciones para delimitar el conjunto de información a recuperar utilizando las siguientes cláusulas tal y como hemos indicado en el apartado 7.3:

- La cláusula `BETWEEN … AND` para restringir la búsqueda a un determinado periodo de tiempo.

  Ejemplo

  La siguiente variación de la Consulta 7.6, permite obtener los acontecimientos de línea de vida de la instancia de una muestra identificada como 'bloodSample007' sucedidos a lo largo del año 2016:

  SELECT LIFELINE
  RELATED TO OBJECT 'bloodSample007' OF TYPE Sample
  BETWEEN '2016-01-01' AND '2016-12-31'
  Consulta 7.12

LIFELINE
OCURRENCE
('occurrence001', ('bloodSample007', 'sampleRecordingProtocolExecution001', 'sampleState001'), 2016-03-14 09:55:00)
('occurrence002', ('bloodSample007', 'sampleStoringProtocolExecution002', 'sampleState002'), 2016-03-14 10:00:00)

  Tal y como hemos indicado en el apartado 7.3, la cláusula `BETWEEN` puede ir precedida del operador lógico `NOT` para negar la condición definida en la misma.

  Ejemplo

  Si modificamos la consulta anterior para incluir el operador `NOT`, podemos obtener los acontecimientos de línea de vida de la instancia de una muestra identificada como 'bloodSample007' que no han sucedido durante el año 2016:

  SELECT LIFELINE
  RELATED TO OBJECT 'bloodSample007' OF TYPE Sample
  NOT BETWEEN '2016-01-01' AND '2016-12-31'
  Consulta 7.13

LIFELINE
OCURRENCE
('occurrence008', ('bloodSample007', 'sampleDestroyingProtocolExecution004', 'sampleState006'), 2017-03-22 12:30:00)

- La cláusula `ON` para restringir la búsqueda a un determinado instante temporal.

Ejemplo

La siguiente consulta permite obtener los acontecimientos de línea de vida de la instancia de una muestra identificada como 'bloodSample007' sucedidos el 14/03/2016 a las 10:00:00:

```
SELECT LIFELINE
RELATED TO OBJECT 'bloodSample007' OF TYPE Sample
ON '2016-03-14 10:00:00'
```

Consulta 7.14

LIFELINE
OCURRENCE
('occurrence002', ('bloodSample007', 'sampleStoringProtocolExecution002', 'sampleState002'), 2016-03-14 10:00:00)

Recordemos que, tal y como hemos indicado en el apartado 7.3, la cláusula `ON`, al igual que la cláusula `BETWEEN … AND`, puede ir precedida del operador lógico `NOT` para negar la condición definida en la misma.

- La cláusula `BEFORE` para obtener información referente a acontecimientos ocurridos antes del acontecimiento indicado, y la cláusula `AFTER` para solicitar información referente a acontecimientos ocurridos después del acontecimiento indicado.

  Este tipo de condiciones son interesantes, por ejemplo, para analizar lo sucedido antes o después de un incidente o de un evento que requiere especial atención.

  Ejemplo

  La siguiente consulta permite obtener los acontecimientos de línea de vida de la instancia de una muestra identificada como 'bloodSample007' sucedidos antes del acontecimiento 'occurrence003':

```
SELECT LIFELINE
RELATED TO OBJECT 'bloodSample007' OF TYPE Sample
BEFORE 'occurrence003'
```

Consulta 7.15

LIFELINE
OCURRENCE
('occurrence001', ('bloodSample007', 'sampleRecordingProtocolExecution001', 'sampleState001'), 2016-03-14 09:55:00)
('occurrence002', ('bloodSample007', 'sampleStoringProtocolExecution002', 'sampleState002'), 2016-03-14 10:00:00)

Ejemplo

La siguiente consulta permite obtener los acontecimientos de línea de vida de la instancia de una muestra identificada como 'bloodSample007' sucedidos después del acontecimiento 'occurrence003':

SELECT LIFELINE
RELATED TO OBJECT 'bloodSample007' OF TYPE Sample
AFTER 'occurrence003'
Consulta 7.16

LIFELINE
OCURRENCE
('occurrence008', ('bloodSample007', 'sampleDestroyingProtocolExecution004', 'sampleState006'), 2017-03-22 12:30:00)

Ordenación

Por último, tal y como se ha indicado en el apartado 7.4.1, para este tipo de consultas no se contempla la cláusula `ORDER BY` para modificar el tipo de ordenación por defecto que se aplicará por considerarse que la ordenación natural (y única) en las estructuras longitudinales es el tiempo.

Esto quiere decir, que los resultados de cualquier consulta para obtener acontecimientos incluidos en una estructura longitudinal, son devueltos ordenados por fecha de ocurrencia de dichos acontecimientos, empezando por los más antiguos y terminando por los más recientes. Por lo general, la fecha de ocurrencia de un acontecimiento suele corresponder con las fechas de ejecución del protocolo que lo ha generado.

7.4.3 Consultas para obtener acontecimientos o elementos acontecimientos que no constituyen una estructura longitudinal

Este tipo de consultas (*Occurrences Queries*) se utiliza para obtener acontecimientos o elementos propios del concepto *Acontecimiento* (objeto, efecto y ejecución de un protocolo) que cumplen ciertas condiciones y no constituyen una estructura longitudinal.

La cláusula SELECT

A diferencia de las consultas para obtener acontecimientos incluidos en una estructura longitudinal, este tipo de consultas permite obtener acontecimientos y elementos de acontecimientos en función de las palabras reservadas que se declaren como parte de la cláusula `SELECT`.

Podemos obtener acontecimientos incluyendo, como parte de la cláusula `SELECT`, la palabra reservada `OCCURRENCE` que, opcionalmente, puede ir precedida de las palabras reservadas `INITIAL` o `FINAL` para especificar que deseamos obtener respectivamente el acontecimiento inicial o final relacionado con los elementos que especifiquemos en la cláusula `RELATED TO` de la que hablaremos más adelante.

Ejemplo

A través de la siguiente consulta, es posible determinar los acontecimientos producidos por la

ejecución del protocolo ‘sampleAliquotingProtocoloExecution002’:

```
SELECT OCCURRENCE
RELATED TO PROTOCOL EXECUTION ‘sampleStoringProtocolExecution002’
    OF TYPE SampleAliquotingProtocolExecution
```

Consulta 7.17

La ejecución de esta consulta daría los siguientes resultados:

OCURRENCE
(‘occurrence002’, (‘bloodSample007’, ‘sampleStoringProtocolExecution002’, ‘sampleState002’), 2016-03-14 10:00:00)
(‘occurrence003 ‘, (‘box001’, ‘sampleStoringProtocolExecution002’, ‘boxState002’), 2016-03-14 10:00:01)

Por otra parte, también es posible declarar consultas para obtener elementos de un acontecimiento como el objeto al que refiere la ejecución de protocolo que lo ha producido, el efecto causado por dicha ejecución de protocolo, etc. en lugar de acontecimientos completos; para ello se utilizan las palabras reservadas `OBJECT`, `ACTIVE STATE` y `PROTOCOL EXECUTION`. Tal y como hemos indicado, la posibilidad de usar la totalidad o parte de las palabras reservadas depende de la estrategia de diseño seleccionada.

Es posible seleccionar varios elementos de los acontecimientos refiriendo a ellos a través de las correspondientes palabras reservadas, una detrás de otra y separadas por comas. Por ejemplo, a través de la siguiente cláusula `SELECT`, podríamos obtener una lista de acontecimientos mostrando la terna de elementos que los definen (objeto, estado y ejecución de protocolo): `SELECT OBJECT, ACTIVE STATE, PROTOCOL EXECUTION`.

Ejemplo

Podemos modificar la Consulta 7.17 para obtener un punto de vista diferente, aunque equivalente:

```
SELECT OBJECT, ACTIVE STATE, PROTOCOL EXECUTION
RELATED TO PROTOCOL EXECUTION ‘sampleStoringProtocolExecution002’
    OF TYPE SampleAliquotingProtocolExecution
```

Consulta 7.18

La ejecución de esta consulta daría los siguientes resultados:

OBJECT	ACTIVE STATE	PROTOCOL EXECUTION
bloodSample007	sampleState002	sampleStoringProtocolExecution002
box001	boxState002	sampleStoringProtocolExecution002

Si comparamos los resultados de esta consulta con los de la Consulta 7.17, podemos observar que la información obtenida es equivalente; como parte de los resultados podemos ver los estados activos resultantes del cambio de estado producido por la ejecución del protocolo ‘sampleStoringProtocolExecution002’, así como las propias ejecuciones de protocolo y los

objetos afectados.

El resultado obtenido en esta última consulta viene condicionado por el tipo de información solicitada en la cláusula `RELATED TO`; si, por ejemplo, hubiéramos declarado `RELATED TO OBJECT 'bloodSample007' OF TYPE Sample`, hubiéramos obtenido todos los acontecimientos ocurridos sobre dicho objeto y por tanto todos los estados activos por los que ha pasado. Hablaremos con más detalle de la cláusula `RELATED TO` y de cómo se utiliza para condicionar los resultados un poco más adelante.

Una de las posibilidades que ofrece el lenguaje a través de la cláusula `SELECT` es restringir el tipo de resultados que se desea obtener utilizando la palabra reservada `OF TYPE` como parte de dicha cláusula. La palabra reservada `OF TYPE` puede referir tanto a uno de los tipos de acontecimiento contemplados en el modelo cuando va precedida de la palabra reservada `OCCURRENCE`, como a una de las clases declaradas en el modelo que refieren a elementos de un acontecimiento si va precedida de las palabras reservadas `OBJECT`, `ACTIVE STATE` y `PROTOCOL EXECUTION`.

Ejemplo

A través de la siguiente consulta, es posible seleccionar un conjunto de objetos de tipo *Sample* (muestra), afectados por la ejecución del protocolo 'sampleStoringProtocoloExecution002' (que es de tipo *SampleStoringProtocolExecution*) a lo largo del año 2016:

```
SELECT OBJECT OF TYPE Sample
RELATED TO PROTOCOL EXECUTION 'sampleStoringProtocolExecution002'
    OF TYPE SampleAliquotingProtocolExecution
BETWEEN '2016-01-01' AND '2016-12-31'
```

Consulta 7.19

Al ejecutar la consulta anterior obtendríamos como resultado el objeto 'bloodSample007' sobre el que ha ocurrido uno de los acontecimientos representados en la Tabla 7.1. Recordemos que la ejecución del protocolo 'sampleStoringProtocolExecution002' ha generado otro acontecimiento, aunque este no sería mostrado al ejecutar la consulta anterior ya que hace referencia a un objeto de tipo *Box* que ha quedado excluido por la condición `OBJECT OF TYPE Sample` definida en la cláusula `SELECT`.

Así pues, al declarar la palabra reservada `OBJECT` en la consulta anterior, estamos indicando que, de los diferentes elementos que definen al concepto *Acontecimiento*, queremos recuperar únicamente los objetos; es decir, que, a través de esta consulta, estamos listando acontecimientos, pero mostrando únicamente una parte de la información que los define, concretamente los objetos. Además, al restringir el tipo de objeto a la clase *Sample* utilizando la palabra reservada `OF TYPE` como parte de la cláusula `SELECT`, estamos indicando que únicamente queremos obtener los acontecimientos sucedidos sobre muestras.

Ejemplo

La siguiente consulta incluye la palabra reservada `OF TYPE` para permitir seleccionar los objetos de tipo *Sample* y las ejecuciones de protocolo realizadas por el usuario 'laboratoryAssistant001' durante el año 2016 que han afectado a dichos objetos:

```
SELECT OBJECT OF TYPE Sample, PROTOCOL EXECUTION
RELATED TO PERFORMER 'laboratoryAssistant001' OF TYPE UserPerformer
```

BETWEEN ‘2016-01-01’ AND ‘2016-12-31’
Consulta 7.20

De esta forma, al ejecutar la consulta anterior obtendríamos los siguientes resultados:

OBJECT	PROTOCOL EXECUTION
bloodSample007	sampleRecordingProtocolExecution001
bloodSample007	sampleStoringProtocolExecution002

En el ejemplo anterior hemos podido ver como el uso de la palabra reservada `OF TYPE` tiene repercusiones no sólo sobre el elemento al que hace referencia, sino también sobre los elementos que lo acompañan como parte de la cláusula `SELECT`. Por este motivo, la siguiente consulta no produciría resultados:

SELECT OBJECT OF TYPE Sample, PROTOCOL EXECUTION OF TYPE RackStoringProtocolExecution
RELATED TO PERFORMER ‘laboratoryAssistant001’ OF TYPE UserPerformer
BETWEEN ‘2016-01-01’ AND ‘2016-12-31’
Consulta 7.21

La cláusula RELATED TO

A través de la cláusula `RELATED TO` es posible especificar a qué instancias de elementos de un acontecimiento hace referencia la información que se desea obtener.

La palabra reservada `OF TYPE`, precedida de las palabras reservadas `OBJECT`, `ACTIVE STATE` o `PROTOCOL EXECUTION`, se utiliza para referir a clases declaradas en el modelo que refieren a elementos de un acontecimiento.

Cuando, como parte de la cláusula `RELATED TO` especificamos un identificador entre comillas simples después de una de las palabras reservadas, nos estamos refiriendo a una instancia de la clase indicada después de la palabra reservada `OF TYPE`; por ejemplo, al declarar ‘sampleStoringProtocoloExecution002’ en la Consulta 7.19, hemos referido a una instancia concreta de la clase *SampleStoringProtocolExecution* definida en el modelo de la base de acontecimientos que, a su vez, es una implementación de la clase *ProtocolExecutionType* definida en el metamodelo.

Ejemplo

Tal y como hemos indicado al explicar la cláusula `SELECT`, procedamos a modificar la cláusula `RELATED TO` de la Consulta 7.18 y observemos los resultados obtenidos:

SELECT OBJECT, ACTIVE STATE, PROTOCOL EXECUTION
RELATED TO OBJECT ‘bloodSample007’ OF TYPE Sample
Consulta 7.22

La ejecución de esta consulta permitiría obtener los siguientes resultados y, tal y como hemos indicado, ver, como parte de los mismos, los estados activos por los que ha pasado el objeto ‘bloodSample007’:

OBJECT	ACTIVE STATE	PROTOCOL EXECUTION
bloodSample007	sampleState001	sampleRecordingProtocolExecution001
bloodSample007	sampleState002	sampleStoringProtocolExecution002
bloodSample007	sampleState006	sampleDestroyingProtocolExecution004

A la hora de declarar los elementos de la cláusula `RELATED TO`, el lenguaje posibilita referir en una misma consulta a múltiples elementos o instancias separados por los operadores lógicos `AND` u `OR`.

Ejemplo

La siguiente consulta permite seleccionar el conjunto de acontecimientos producido por la ejecución del protocolo 'sampleAliquotingProtocoloExecution002' realizada por parte del usuario 'laboratoryAssistant001' (que es instancia de la clase *UserPerformer*):

```
SELECT OBJECT, ACTIVE STATE
RELATED TO
    PROTOCOL EXECUTION 'sampleStoringProtocolExecution002'
    AND
    PERFORMER 'laboratoryAssistant001' OF TYPE UserPerformer
```

Consulta 7.23

Al ejecutar esta consulta obtendríamos un conjunto de acontecimientos de los que únicamente se mostraría el objeto afectado y el estado activo en el que quedó dicho objeto al ejecutarse el protocolo 'sampleStoringProtocolExecution002' por parte del usuario identificado como 'laboratoryAssistant001':

OBJECT	ACTIVE STATE
bloodSample007	sampleState002
box001	boxState002

Al referir a la Consulta 7.23 hemos indicado que posibilita obtener los acontecimientos producidos por las ejecuciones de protocolo realizadas por el usuario 'laboratoryAssistant001' y para ello hemos utilizado la palabra reservada `PERFORMER`, que no corresponde con ninguno de los elementos propios del concepto *Acontecimiento*, sino que hace referencia a un elemento relacionado con uno de ellos (con la ejecución de protocolo).

Por último, en relación a la Consulta 7.23, también puede observarse que en una de las sentencias no hemos incluido la palabra reservada `OF TYPE`; recordemos que, tal y como hemos indicado en el apartado 7.4.1, esta palabra reservada es opcional, por lo que dicha consulta dará los mismos resultados que obtendríamos si hubiéramos usado dicha palabra reservada, aunque con impacto en el rendimiento tal y como hemos indicado también en dicho apartado.

Ejemplo

Si modificamos la Consulta 7.23 para especificar la palabra reservada `OF TYPE`, obtenemos la siguiente consulta:

```
SELECT OBJECT, ACTIVE STATE
RELATED TO
    PROTOCOL EXECUTION 'sampleStoringProtocolExecution002'
    OF TYPE SampleStoringProtocolExecution
    AND
    PERFORMER 'laboratoryAssistant001' OF TYPE UserPerformer
```

Consulta 7.24

El *Lenguaje de Acceso a Acontecimientos* contempla realizar negaciones sobre las afirmaciones declaradas como parte de la cláusula `RELATED TO` incluyendo el operador lógico `NOT` antes de referir a la instancia del elemento.

Ejemplo

La siguiente consulta, que incluye el operador lógico `NOT` como parte de una de las afirmaciones definidas en la cláusula `RELATED TO`, posibilita obtener todos los acontecimientos producidos por ejecuciones de protocolo realizadas por el usuario 'laboratoryAssistant001' que no han afectado a la muestra 'bloodSample007':

```
SELECT OCCURRENCE
RELATED TO
    OBJECT NOT 'bloodSample007' OF TYPE Sample
    AND
    PERFORMER 'laboratoryAssistant001' OF TYPE UserPerformer
```

Consulta 7.25

Al ejecutar esta consulta obtendríamos los siguientes acontecimientos:

OCURRENCE
('occurrence004 ', ('bloodSample008', 'sampleRecordingProtocolExecution002', 'sampleState003'), 2017-03-15 09:45:00)
('occurrence005 ', ('bloodSample008', 'sampleStoringProtocolExecution003', 'sampleState004'), 2017-03-15 09:50:00)
('occurrence007 ', ('bloodSample008', 'sampleAliquotingProtocolExecution005', 'sampleState005'), 2017-03-16 10:20:00)
('occurrence003 ', ('box001', 'sampleStoringProtocolExecution002', 'boxState002'), 2016-03-14 10:00:01)
('occurrence006 ', ('box001', 'sampleStoringProtocolExecution003', 'boxState003'), 2017-03-15 09:50:01)
('occurrence009 ', ('box001', 'sampleDestroyingProtocolExecution004', 'boxState004'), 2017-03-22 12:30:01)

Condiciones

Al igual que en el caso de las consultas para obtener acontecimientos incluidos en una estructura longitudinal, para este tipo de consultas, el lenguaje también permite establecer condiciones de búsqueda para obtener exclusivamente la información generada dentro de un determinado rango de fechas (`BETWEEN … AND`) o en un instante temporal dado (`ON`), o la información anterior (`BEFORE`) o posterior (`AFTER`) a un acontecimiento.

Ejemplo

A través de la siguiente consulta es posible seleccionar los objetos afectados por las ejecuciones de protocolo realizadas por el usuario 'laboratoryAssistant001' durante el año 2016:

SELECT OBJECT, PROTOCOL EXECUTION
RELATED TO PERFORMER 'laboratoryAssistant001' OF TYPE UserPerformer
BETWEEN '2016-01-01' AND '2016-12-31'
Consulta 7.26

Al ejecutar esta consulta obtendríamos los siguientes resultados:

OBJECT	PROTOCOL EXECUTION
bloodSample007	sampleRecordingProtocolExecution001
bloodSample007	sampleStoringProtocolExecution002
box001	sampleStoringProtocolExecution002

Tal y como hemos indicado en el apartado 7.3, el lenguaje contempla la posibilidad de introducir negaciones sobre las afirmaciones realizadas en la cláusula `BETWEEN … AND` y en la cláusula `ON`.

Ejemplo

A través de la siguiente consulta es posible seleccionar los objetos afectados por las ejecuciones de protocolo realizadas por el usuario 'laboratoryAssistant001' durante cualquier año excepto el 2016:

SELECT OBJECT, PROTOCOL EXECUTION
RELATED TO PERFORMER 'laboratoryAssistant001' OF TYPE UserPerformer
NOT BETWEEN '2016-01-01' AND '2016-12-31'
Consulta 7.27

Al ejecutar esta consulta obtendríamos los siguientes resultados:

OBJECT	PROTOCOL EXECUTION
bloodSample008	sampleRecordingProtocolExecution002
bloodSample008	sampleStoringProtocolExecution003
box001	sampleStoringProtocolExecution003
bloodSample008	sampleAliquotingProtocolExecution005
bloodSample007	sampleDestroyingProtocolExecution004
box001	sampleDestroyingProtocolExecution004

Ejemplo

A través de la siguiente consulta es posible seleccionar los acontecimientos ocurridos con anterioridad al acontecimiento identificado como 'occurrence004':

SELECT OCCURRENCE

RELATED TO PERFORMER ‘laboratoryAssistant001’ OF TYPE UserPerformer
BEFORE ‘occurrence004’
Consulta 7.28

Al ejecutar esta consulta obtendríamos los siguientes resultados:

OCURRENCE
(‘occurrence001’, (‘bloodSample007’, ‘sampleRecordingProtocolExecution001’, ‘sampleState001’), 2016-03-14 09:55:00)
(‘occurrence002’, (‘bloodSample007’, ‘sampleStoringProtocolExecution002’, ‘sampleState002’), 2016-03-14 10:00:00)
(‘occurrence003 ‘, (‘box001’, ‘sampleStoringProtocolExecution002’, ‘boxState002’), 2016-03-14 10:00:00)

Ordenación

A diferencia de las consultas para obtener acontecimientos incluidos en estructuras longitudinales, la sintaxis de este tipo de consultas sí que admite especificar uno o varios criterios de ordenación a través de la cláusula `ORDER BY`.

Ejemplo

La siguiente consulta posibilita obtener los estados activos de un conjunto de acontecimientos que hayan sido generados por las ejecuciones de protocolo realizadas por el usuario ‘laboratoryAssistant001’, obteniendo los resultados ordenados por identificador de estado activo de manera ascendente (recordar, según hemos indicado en el apartado 7.3.5, que si no se incluyen las palabras reservadas `ASC` o `DESC` se aplica el criterio de ordenación establecido por defecto, que es ascendente):

SELECT OBJECT, ACTIVE STATE
RELATED TO PERFORMER ‘laboratoryAssistant001’ OF TYPE UserPerformer
ORDER BY ACTIVE STATE
Consulta 7.29

Al ejecutar esta consulta obtendríamos los siguientes resultados:

OBJECT	ACTIVE STATE
box001	boxState002
box001	boxState003
box001	boxState004
bloodSample007	sampleState001
bloodSample007	sampleState002
bloodSample008	sampleState003
bloodSample008	sampleState004
bloodSample008	sampleState005
bloodSample007	sampleState006

Es posible especificar varios criterios de ordenación separando cada uno de ellos por comas.

Ejemplo

La siguiente consulta posibilita obtener los estados activos de un conjunto de acontecimientos que hayan sido generados por las ejecuciones de protocolo realizadas por el usuario 'laboratoryAssistant001', obteniendo los resultados ordenados por identificador de objeto de manera ascendente y por identificador de ejecución de protocolo de manera descendente:

SELECT OBJECT, ACTIVE STATE, PROTOCOL EXECUTION
RELATED TO PERFORMER 'laboratoryAssistant001' OF TYPE UserPerformer
ORDER BY OBJECT ASC, PROTOCOL EXECUTION DESC
Consulta 7.30

Al ejecutar esta consulta obtendríamos los siguientes resultados:

OBJECT	ACTIVE STATE	PROTOCOL EXECUTION
box001	boxState003	sampleStoringProtocolExecution003
box001	boxState002	sampleStoringProtocolExecution002
box001	boxState004	sampleDestroyingProtocolExecution004
bloodSample007	sampleState002	sampleStoringProtocolExecution002
bloodSample007	sampleState001	sampleRecordingProtocolExecution001
bloodSample007	sampleState006	sampleDestroyingProtocolExecution004
bloodSample008	sampleState004	sampleStoringProtocolExecution003
bloodSample008	sampleState003	sampleRecordingProtocolExecution002
bloodSample008	sampleState005	sampleAliquotingProtocolExecution005

7.4.4 Sintaxis completa

La sintaxis completa en notación BNF del *Lenguaje No Generalizado de Acceso a Acontecimientos* es la siguiente:

```
<OccurrenceQL> ::= <longitudinal_structure_query> | <occurrences_query>;
(* Longitudinal Structure Queries *)
<longitudinal_structure_query> ::=
   SELECT [INITIAL OCCURRENCE OF | FINAL OCCURRENCE OF]
    <longitudinal_structure_type>
   [RELATED TO <structure_element_instances>]
   [[NOT] BETWEEN <date> AND <date> | [NOT] ON <date>]
   [BEFORE <occurrence_instance> | AFTER <occurrence_instance>];
<structure_element_instances> ::=
```

```
   <structure_element> [NOT] <structure_element_instance> [OF TYPE <type_name>]
   [(AND | OR) <structure_element_instances>];
(* Occurrences Queries *)
<occurrences_query> ::=
   SELECT <occurrence_related_elements>
   [RELATED TO <occurrence_element_instances>]
   [[NOT] BETWEEN <date> AND <date> | [NOT] ON <date>]
   [BEFORE <occurrence_instance> | AFTER <occurrence_instance>]
   [ORDER BY <occurrence_elements_with_order_type>];
<occurrence_related_elements> ::=
   [INITIAL | FINAL] OCCURRENCE [OF TYPE <occurrence_type>] |
   <occcurrence_elements_with_type>;
<occurrence_elements> ::=
   <occurrence_element> [, <occurrence_elements>];
<occurrence_elements_with_type> ::=
   <occurrence_element> [OF TYPE <type_name>]
   [, <occurrence_elements_with_type>];
<occurrence_element_instances> ::=
   <occurrence_element>    [NOT]    <occurrence_element_instance>    [OF    TYPE
   <type_name>] [(AND | OR) <occurrence_element_instances>];
<occurrence_elements_with_order_type> ::=
   <occurrence_element> [<order_type>]
   [, <occurrence_elements_with_order_type>];
<order_type> ::= ASC | DESC;
```

Listado 7.1 Sintaxis del Lenguaje No Generalizado de Acceso a Acontecimientos

7.5 Lenguaje generalizado de acceso a acontecimientos

El *Lenguaje Generalizado de Acceso a Acontecimientos* (*Generalized Occurrence Access Language; Generalized OcAL*) posibilita la declaración de consultas que refieran a los conceptos definidos en el *Metamodelo de Bases de Acontecimientos* como *Tipo de Objeto*, *Tipo de Ejecución de Protocolo*, etc.; es decir, se incorpora el nivel M2 de UML a los niveles M0 y M1 abarcados por el lenguaje no generalizado tal y como presentamos en la siguiente figura:

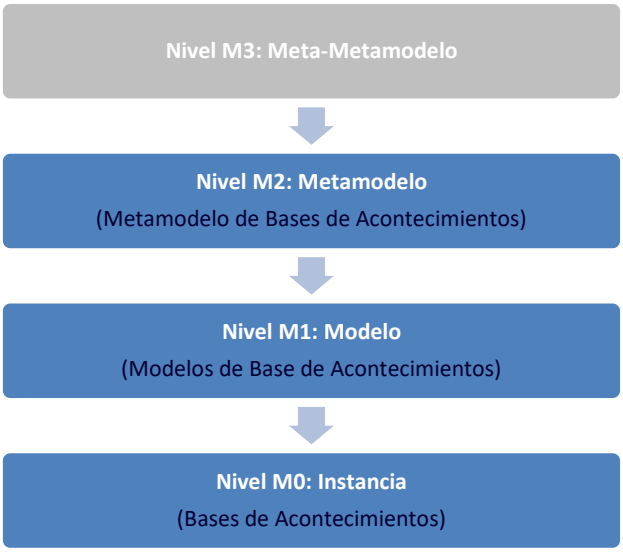


Figura 7.4 Niveles de UML abarcados por OcAL generalizado

Por ejemplo, usando el *Lenguaje Generalizado de Acceso a Acontecimientos* se podría hacer una consulta para obtener las líneas de vida de un tipo de objeto, o una consulta para obtener los tipos de estado activo afectados por un tipo de ejecución de protocolo.

7.5.1 Introducción a la sintaxis

En este apartado vamos a introducir la sintaxis del *Lenguaje Generalizado de Acceso a Acontecimientos* refiriendo en cada caso a la sintaxis que corresponda utilizando la notación BNF. La sintaxis completa se incluye en el apartado 7.5.3 y la forma de uso de la notación BNF en el Anexo A.

La sintaxis del *Lenguaje Generalizado de Acceso a Acontecimientos*, al igual que sucede en el caso de la variante no generalizada, tiene una estructura base en la que se introducen variaciones en función de los dos tipos de consultas que hemos explicado en el apartado 7.4.2 y en el apartado 7.4.3. La forma de especificar esto en notación BNF es la siguiente:

```
<GOccurrenceQL> ::= <G_longitudinal_structure_query> | <G_occurrences_query>;
```

1 Consultas para obtener acontecimientos incluidos en una estructura longitudinal (*Longitudinal Structure Queries*).

A continuación, mostramos una versión simplificada de la sintaxis de este tipo de sentencias, en el apartado 7.5.3 completaremos la definición:

```
<G_longitudinal_structure_query> ::=
SELECT [INITIAL OCCURRENCE OF | FINAL OCCURRENCE OF]
<longitudinal_structure_type>
[RELATED TO <structure_element> <structure_element_instance> [OF TYPE
<type_name>] | <structure_element> TYPE <type_name>]
[BETWEEN <date> AND <date> | ON <date>]
[BEFORE <ocurrence_instance> | AFTER <ocurrence_instance>];
```

2 Consultas para obtener acontecimientos o elementos de acontecimientos que no constituyen una estructura longitudinal (*Occurrences Queries*).

A continuación, mostramos una versión simplificada de la sintaxis de este tipo de sentencias, en el apartado 7.5.3 completaremos la definición:

```
<Goccurrences_query> ::=
SELECT [INITIAL | FINAL] OCCURRENCE [OF TYPE <occurrence_type>]
| <occurrence_element> [OF TYPE <type_name>] | <occurrence_element> TYPE
[RELATED TO <occurrence_element> <occurrence_element_instance> [OF TYPE
<type_name>] | <occurrence_element> TYPE <type_name>]
[BETWEEN <date> AND <date> | ON <date>]
[BEFORE <ocurrence_instance> | AFTER <ocurrence_instance>]
[ORDER BY <occurrence_elements_with_order_type>
| ORDER BY <occurrence_element_types_with_order_type>];
```

Pero los cambios más importantes se pueden observar al comparar la sintaxis del *Lenguaje Generalizado de Acceso a Acontecimientos* con la sintaxis de la variante no generalizada a la que amplía. La sintaxis del *Lenguaje Generalizado de Acceso a Acontecimientos* hace referencia no sólo a elementos propios del concepto *Acontecimiento* y a sus instancias, sino también a los tipos de estos elementos, que corresponden con conceptos contemplados en el metamodelo como *Tipo de Ejecución de Protocolo*, *Tipo de Objeto*, etc. Las expresiones `<structure_element> TYPE` y `<occurrence_element> TYPE` se introducen para especificar cómo hacer referencia a dichos conceptos; la forma de hacerlo es declarar una de las palabras reservadas del lenguaje como `OBJECT`, `PROTOCOL EXECUTION`, etc. (cualquiera de las permitidas por la estrategia de diseño seleccionada) en lugar de `<structure_element>` o `<occurrence_element>`, seguida de la palabra reservada `TYPE` (`OBJECT TYPE`, `PROTOCOL EXECUTION TYPE`, etc.). Recordemos que en el apartado 4.5 hemos hablado sobre las palabras reservadas del lenguaje.

La distinción entre elementos de un acontecimiento y sus tipos de la que acabamos de hablar está contemplada no sólo en la cláusula `SELECT`, sino también en la cláusula `RELATED TO` para posibilitar establecer los criterios de selección de los acontecimientos sobre los que se pide la información, y en la cláusula `ORDER BY`, para declarar los criterios de ordenación.

No nos vamos a extender mucho más en este subapartado dado que el resto de conceptos y el uso de cada una de las cláusulas del lenguaje ha sido previamente introducido en los apartados 7.3 y 7.4.1.

7.5.2 Descripción del lenguaje

El *Lenguaje Generalizado de Acceso a Acontecimientos* posibilita obtener el tipo especificado en el metamodelo al que pertenecen las instancias declaradas en un modelo de base de acontecimientos. Por ejemplo, dada la muestra identificada como 'bloodSample007', a través de una consulta, podríamos llegar a saber que es instancia de un objeto de tipo *Sample* (muestra) que a su vez es instancia de la clase ObjectType del metamodelo.

Por otra parte, el *Lenguaje Generalizado de Acceso a Acontecimientos* posibilita que los resultados puedan condicionarse para lograr relacionarlos con las instancias derivadas de un tipo específico; es decir, es posible evitar especificar las instancias, indicando solamente el tipo. Por ejemplo, especificando el tipo de objeto *Sample*, podríamos declarar una consulta OcAL para obtener todas las instancias del mismo.

La combinación de estas dos posibilidades aplicándolas sobre las cláusulas `SELECT` y `RELATED TO` da lugar a los tres tipos de consultas especificados en la siguiente tabla:

	OcAL no generalizado	**OcAL generalizado**		
		Primer tipo	**Segundo tipo**	**Tercer tipo**
SELECT	Instancias	Instancias	Tipos	Tipos
RELATED TO	Instancias	Tipos	Instancias	Tipos

Tabla 7.6 Tipos de consulta del lenguaje OcAL generalizado

Primer tipo

El primer tipo de consultas de la variante generalizada del lenguaje refiere a las consultas declaradas para obtener instancias de la base de acontecimientos relacionadas con tipos específicos de instancias.

Ejemplo

La siguiente consulta utiliza la palabra reservada `OBJECT` seguida de la palabra reservada `TYPE` como parte de la cláusula `RELATED TO` para obtener el acontecimiento inicial de las líneas de vida de todos los objetos de tipo *Sample*:

```
SELECT INITIAL OCCURRENCE OF LIFELINE
RELATED TO OBJECT TYPE Sample
```

Consulta 7.31

LIFELINE
OCURRENCE
(‘occurrence001’, (‘bloodSample007’, ‘sampleRecordingProtocolExecution001’, ‘sampleState001’), 2016-03-14 09:55)
(‘occurrence004 ‘, (‘bloodSample008’, ‘sampleRecordingProtocolExecution002’, ‘sampleState003’), 2017-03-15 09:45)

La ejecución de la consulta anterior daría como resultado dos acontecimientos, que corresponden con los acontecimientos iniciales de las líneas de vida de los objetos ‘bloodSample007’ y ‘bloodSample008’, que son instancias de la clase *Sample* o, dicho de otra forma, son objetos de tipo *Sample*. En dicha consulta, se puede observar un caso de uso del concepto *Tipo de Objeto* (ObjectType) especificado en el metamodelo.

Dentro del *Lenguaje Generalizado de Acceso a Acontecimientos*, las consultas para obtener acontecimientos incluidos en una estructura longitudinal (*Longitudinal Structure Queries*), son específicas del primer tipo dado que su intencionalidad es posibilitar la obtención de instancias de acontecimientos que, en este caso, forman parte de una estructura longitudinal, por lo que no están contempladas en los otros dos tipos de consultas de la variante generalizada del lenguaje de los que vamos a hablar a continuación.

Ejemplo

La siguiente consulta posibilita obtener cada muestra (las instancias de objeto de tipo *Sample*) asociada al tipo de ejecución de protocolo *SampleStoringProtocolExecution* (en lugar de a una ejecución de protocolo concreta como en los ejemplos mostrados en el apartado 7.4) a lo largo del año 2016:

SELECT OBJECT OF TYPE Sample, PROTOCOL EXECUTION
RELATED TO PROTOCOL EXECUTION TYPE SampleStoringProtocolExecution
BETWEEN '2016-01-01' AND '2016-12-31'
Consulta 7.32

Al ejecutar esta consulta obtendríamos los objetos de tipo *Sample* y las ejecuciones de protocolo que han afectado a dichos objetos de los acontecimientos ocurridos durante el año 2016 generados por ejecuciones de protocolo de tipo *SampleStoringProtocolExecution*; concretamente la consulta anterior daría los siguientes resultados:

OBJECT	PROTOCOL EXECUTION
bloodSample007	sampleStoringProtocolExecution002

Si observamos el conjunto de acontecimientos definido en la Tabla 7.1, podemos ver que el objeto 'bloodSample007', durante el año 2016 se ha visto afectado únicamente por la ejecución de protocolo identificada como 'sampleStoringProtocolExecution002', la cual es instancia de la clase *SampleStoringProtocolExecution*.

Ejemplo

A través de la siguiente consulta, es posible determinar los acontecimientos producidos por la ejecución de los protocolos de tipo *SampleStoringProtocolExecution* ejecutados por usuarios (*UserPerformer*):

SELECT OCCURRENCE
RELATED TO
    PROTOCOL EXECUTION TYPE SampleStoringProtocolExecution
    AND
    PERFORMER TYPE UserPerformer
Consulta 7.33

OCURRENCE
('occurrence002', ('bloodSample007', 'sampleStoringProtocolExecution002', 'sampleState002'), 2016-03-14 10:00)
('occurrence003 ', ('box001', 'sampleStoringProtocolExecution002', 'boxState002'), 2016-03-14 10:00)
('occurrence005 ', ('bloodSample008', 'sampleStoringProtocolExecution003', 'sampleState004'), 2017-03-15 09:50)
('occurrence006 ', ('box001', 'sampleStoringProtocolExecution003', 'boxState003'), 2017-03-15 09:50)

Segundo tipo

El segundo tipo de consultas de la variante generalizada del lenguaje refiere a las consultas

declaradas para obtener el tipo de las instancias relacionadas con instancias específicas de la base de acontecimientos.

Este tipo de consultas (y las consultas del tercer tipo de las que hablaremos más adelante), a diferencia de las consultas del primer tipo que acabamos de explicar y de las consultas de la variante no generalizada del lenguaje, no devuelven acontecimientos, sino que posibilitan obtener la información descriptiva de una base de acontecimientos almacenada en el *Repositorio de Metadatos*.

Ejemplo

La siguiente consulta posibilita obtener los tipos de estado activo por los que ha ido pasando la muestra 'bloodSample007':

```
SELECT ACTIVE STATE TYPE
RELATED TO OBJECT 'bloodSample007' OF TYPE Sample
```

Consulta 7.34

La consulta anterior presenta un caso de uso del concepto *Tipo de Estado Activo* (ActiveStateType) contemplado en el metamodelo. Al ejecutar esta consulta, obtendríamos los tipos de estado activo por los que ha pasado el objeto 'bloodSample007'; si observamos la Tabla 7.1, los estados por los que ha pasado dicha muestra han sido 'sampleState001', 'sampleState002' y 'sampleState006'; si observamos la Tabla 7.3,el primero de dichos estados es instancia de *ObtainedSample*, el segundo es instancia de *StoredSample*, mientras que el tercero es instancia de *DestroyedSample*; por tanto, los resultados obtenidos serían los siguientes:

ACTIVE STATE TYPE
ObtainedSample
StoredSample
DestroyedSample

Ejemplo

La siguiente consulta posibilita obtener los tipos de objeto y los tipos de estado activo de los acontecimientos producidos por la ejecución del protocolo 'sampleStoringProtocolExecution002' ejecutado por el usuario 'laboratoryAssistant001':

```
SELECT OBJECT TYPE, ACTIVE STATE TYPE
RELATED TO
    PROTOCOL EXECUTION 'sampleStoringProtocolExecution002'
    AND
    PERFORMER 'laboratoryAssistant001' OF TYPE UserPerformer
```

Consulta 7.35

Observando la Tabla 7.1, podemos ver que la ejecución del protocolo 'sampleStoringProtocolExecution002' ha producido dos acontecimientos que han afectado a los objetos 'bloodSample007' y 'box001' que son instancias de los tipos de objeto *Sample* y *Box* respectivamente; por otra parte, en la Tabla 7.3, podemos observar que los estados 'sampleState002' y 'boxState002' son respectivamente instancias de los tipos de estado *StoredSample* y *UsedBox*; por tanto, al ejecutar consulta anterior, obtendríamos los siguientes

resultados:

OBJECT TYPE	ACTIVE STATE TYPE
Sample	StoredSample
Box	UsedBox

Tercer tipo

El tercer tipo de consultas de la variante generalizada del lenguaje refiere a las consultas declaradas para obtener el tipo de las instancias que refieren a tipos específicos de instancias.

Este tipo de consultas, al igual que las consultas del segundo tipo, no devuelven acontecimientos, sino que obtienen información almacenada en el *Repositorio de Metadatos*.

Ejemplo

A través de la siguiente consulta podemos obtener los diferentes tipos de estado activo por los que han pasado las instancias de la clase *Box*:

```
SELECT ACTIVE STATE TYPE
RELATED TO OBJECT TYPE Box
```

Consulta 7.36

La consulta anterior mostraría los tipos de estado activo por los que han pasado las instancias de la clase *Box*:

ACTIVE STATE TYPE
UsedBox

Ejemplo

A través de la siguiente consulta podemos obtener los diferentes tipos de estado activo y los tipos de objeto relacionados con dichos tipos de estado activo resultantes de las ejecuciones de protocolos de tipo *SampleStoringProtocolExecution*:

```
SELECT OBJECT TYPE, ACTIVE STATE TYPE
RELATED TO PROTOCOL EXECUTION TYPE SampleStoringProtocolExecution
```

Consulta 7.37

Al ejecutar la consulta anterior obtendríamos los siguientes resultados:

OBJECT TYPE	ACTIVE STATE TYPE
Sample	StoredSample
Box	UsedBox

7.5.3 Sintaxis completa

La sintaxis completa en notación BNF del *Lenguaje Generalizado de Consultas de Acontecimientos* es la siguiente:

```
<GOccurrenceQL> ::= <G_longitudinal_structure_query> | <G_occurrences_query>;

(* Longitudinal Structure Queries *)

<G_longitudinal_structure_query> ::=
   SELECT [INITIAL OCCURRENCE OF | FINAL OCCURRENCE OF]
   <longitudinal_structure_type>
   [RELATED TO <structure_instances_related_elements>]
   [[NOT] BETWEEN <date> AND <date> | [NOT] ON <date>]
   [BEFORE <ocurrence_instance> | AFTER <ocurrence_instance>];

<structure_instances_related_elements> ::=
   <structure_element_instances> | <structure_element_type_instances>;

<structure_element_instances> ::=
   <structure_element> [NOT] <structure_element_instance> [OF TYPE <type_name>]
   [AND | OR <structure_element_instances>];

<structure_element_type_instances> ::=
   <structure_element_type> [NOT] <type_name> [(AND | OR)
   <structure_element_types_instances>];

<structure_element_type> ::=
   <structure_element> TYPE;

(* Occurrences Queries *)

<Goccurrences_query> ::=
   SELECT <ocurrence_related_elements>
   [RELATED TO <occurrence_instances_related_elements>
   [[NOT] BETWEEN <date> AND <date> | [NOT] ON <date>]
   [BEFORE <ocurrence_instance> | AFTER <ocurrence_instance>]
   [ORDER BY <occurrence_elements_with_order_type>
   | ORDER BY <occurrence_element_types_with_order_type>];

<occurrence_related_elements> ::=
   [INITIAL | FINAL] OCCURRENCE [OF TYPE <occurrence_type>]
   | <occurrence_elements_with_type> | <occurrence_element_types>;

<occurrence_elements> ::=
   <occurrence_element> [, <occurrence_elements>];

<occurrence_elements_with_type> ::=
   <occurrence_element> [OF TYPE <type_name>]
   [, <occurrence_elements_with_type>];

<occurrence_element_types> ::=
```

```
   <occurrence_element_type> [, <occurrence_element_types>];
<occurrence_element_type> ::=
   <occurrence_element> TYPE;
<occurrence_instances_related_elements> ::=
   <occurrence_element_instances> | <occurrence_element_type_instances>;
<occurrence_element_instances> ::=
   <occurrence_element>    [NOT]    <occurrence_element_instance>    [OF    TYPE
   <type_name>] [(AND | OR) <occurrence_element_instances>];
<occurrence_element_type_instances> ::=
   <occurrence_element_type> [NOT] <type_name> [(AND | OR)
   <occurrence_element_type_instances>];
<occurrence_elements_with_order_type> ::=
   <occurrence_element> [<order_type>][,<occurrence_elements_with_order_type>];
<occurrence_element_types_with_order_type> ::=
   <occurrence_element_type> [<order_type>]
   [, <occurrence_element_types_with_order_type>];
<order_type> ::= ASC | DESC;
```

Listado 7.2 Sintaxis del Lenguaje Generalizado de Acceso a Acontecimientos

Capítulo 8. Prueba de concepto

8.1 Introducción y objetivos

En este capítulo se presentan los resultados de la prueba de concepto del *Lenguaje de Acceso a Acontecimientos* validada sobre una versión de la base de acontecimientos del sistema informático *Arasis*.

Los objetivos que se fijaron para la prueba de concepto fueron los siguientes:

1 Verificar la validez de los resultados de la investigación.

2 Medir la complejidad de las consultas del lenguaje frente a las consultas resultantes de traducir las primeras a otro lenguaje de consultas.

3 Evaluar el rendimiento del lenguaje en términos del tiempo de ejecución necesario para obtener la información requerida.

8.2 Pasos previos

Para la realización de la prueba de concepto se siguieron los siguientes pasos:

1 Fijar las dimensiones del lenguaje.

2 Traducción de consultas del lenguaje.

La prueba de concepto se validó sobre la base de acontecimientos del sistema *Arasis* mediante la realización de las siguientes actividades:

1 Definición del esquema del repositorio de metadatos de la base de acontecimientos del sistema *Arasis* poblándolo con información descriptiva de la misma.

2 Definición de un conjunto de consultas de complejidad variable.

3 Traducción de las consultas del lenguaje de acontecimientos.

La traducción de las consultas del lenguaje de acontecimientos se realizó a consultas SQL debido a que la base de acontecimientos del sistema *Arasis* se encuentra implantada sobre un sistema gestor de bases de datos relacionales, concretamente sobre PostgreSQL.

4 Contraste de los resultados obtenidos:

4.1 Comparación de la longitud y complejidad de la consulta OcQL original con la consulta SQL resultante de su traducción.

4.2 Medición en términos de tiempo de ejecución de las consultas SQL resultantes.

8.3 Aplicación de la estrategia de diseño

Para la realización de la prueba de concepto fue necesario tener presente la estrategia de diseño orientada a acontecimientos descrita en el apartado 2.2 y publicada en [24].

Esta estrategia de diseño, propone seguir la metodología *Model Driven Engineering* (*MDE*) descrita en [5] consistente en partir de un modelo independiente de la plataforma (*Platform Independent Model*; PIM) transformándolo para obtener un modelo específico de la plataforma (*Platform Specific Model*; PSM) y, finalmente, la especificación del código fuente del sistema de persistencia. En el contexto de esta tesis, a esta estrategia de diseño la hemos denominado *Estrategia de diseño OCBaseMD* (*OcBase Model-Driven design strategy*; *OcBaseMD design strategy*).

Esta estrategia de diseño, esencialmente establece tanto las clases generales como las específicas que un modelo de base de acontecimientos debe definir, así como sus atributos y asociaciones. Recordemos que las clases de un modelo de base de acontecimientos corresponden con instancias de las clases definidas en el *Metamodelo de Bases de Acontecimientos* presentado en el Capítulo 6. Concretamente, los aspectos de esta estrategia de diseño que se han tenido en cuenta son los presentados en la siguiente tabla:

Aspectos de la estrategia de diseño	
Clases	Clases generales: *ProtocolExecution*, *Object*, *Effect*, *Performer* Una clase específica por cada tipo de objeto, ejecución de protocolo, efecto y ejecutor que hereda de la correspondiente clase general. Una clase por cada tipo de estado activo y de transición desencadenada. Una clase por cada relación entre tipos de objetos.
Atributos	Cada clase específica define sus correspondientes atributos. Cada clase específica hereda los atributos, incluyendo el identificador de objeto, de la correspondiente clase general. Excepcionalmente, las clases específicas relativas a tipos de objetos, definen su propio identificador, mientras que la clase general *Object* mantiene una referencia con cada clase específica.
Asociaciones	Las asociaciones entre las clases corresponden con las establecidas en el metamodelo. Cada clase general se encuentra asociada con sus correspondientes clases específicas.

Tabla 8.1 Principales aspectos de la estrategia de diseño OCBaseMD

Tomando como referencia esta estrategia, una de las decisiones fue cómo adaptar las dimensiones abiertas del lenguaje (hemos hablado de ellas en el apartado 4.2), las cuales finalmente quedaron fijadas de la siguiente manera:

1 Se consideró un tipo de acontecimiento.

2 Se contemplaron dos tipos de estructuras longitudinales: *Historia* y *Línea de vida*.

3 El tipo de elemento considerado en las consultas sobre estructuras longitudinales fue *Objeto*; es decir, que este tipo de consultas se orientó a obtener los acontecimientos de un objeto que forma parte de este tipo de estructuras.

4 Los tipos de elementos considerados en las consultas sobre acontecimientos fueron los siguientes:
   - Objetos
   - Ejecuciones de protocolo
   - Estados activos
   - Efectos
   - Ejecutores de protocolos

La estrategia de diseño que acabamos de describir se debe tener en cuenta por el traductor de consultas (*Occurrence Query Languaje Translator*) y para la definición del esquema del repositorio de metadatos (*Occurrence Base Metadata Repository*; *OcBase Metadata Repository*), válido para describir cualquier base de acontecimientos definida siguiendo esta estrategia de diseño. Hablaremos de estas dos actividades en los siguientes apartados de este capítulo.

8.4 Consideraciones sobre el sistema Arasis

El sistema informático *Arasis* inició su explotación en el año 2009; esto implica que la validación de la prueba de concepto se realizó sobre una base de acontecimientos que en origen no incorporaba algunas de las innovaciones obtenidas en la investigación durante estos años. Así pues, es importante tener en cuenta los siguientes aspectos:

1 En la época en la que se diseñó la base de acontecimientos de *Arasis*, los acontecimientos no se consideraban una entidad del sistema, sino información derivada de las ejecuciones de protocolos, los objetos y los efectos sobre ellos.

2 Las estructuras longitudinales historia y línea de vida son generadas dinámicamente.

Para solventar estas dos cuestiones y alinear la estructura de la base de acontecimientos de *Arasis* con los avances logrados en la investigación se realizaron las siguientes adecuaciones:

1 Se generó una vista a la que se denominó `Occurrence` (acontecimiento) para representar la siguiente información: los identificadores en el sistema (OIDs) de las diferentes ejecuciones de protocolos, los identificadores en el sistema de los objetos y los identificadores en el sistema de los efectos sobre los objetos, todos ellos registrados en las correspondientes tablas de la base de acontecimientos.

2 Se generó una vista a la que se denominó `Lifeline` (línea de vida) para recopilar, a partir del contenido de la vista `Occurrence`, los acontecimientos sucedidos a cada objeto, incorporando la fecha inicial y la fecha final de cada uno de ellos.

3 Se implementó una función a la que se denominó `History` que devuelve, para cada identificador de objeto proporcionado como parámetro de entrada, el conjunto de acontecimientos sucedidos sobre el objeto en sí mismo y sobre sus objetos relacionados.

8.5 Definición del esquema del repositorio de metadatos

En el apartado 3.2.6 hemos explicado la funcionalidad que ofrece el componente del

Framework OcQF al que hemos denominado *Repositorio de Metadatos* (*Occurrence Base Metadata Repository*; *OcBase Metadata Repository*), al que también referimos simplemente como *Diccionario*. En este apartado vamos a presentar, a modo de ejemplo, una implementación del mismo.

Recordemos que el *Repositorio de Metadatos* es el componente del framework que es responsable de almacenar la información que describe la estructura de una base de acontecimientos resultante de implementar un determinado modelo de base de acontecimientos.

El esquema del *Repositorio de Metadatos* se define tomando como referencia la estructura del *Metamodelo de Bases de Acontecimientos* tal y como podremos ver en este apartado, mientras que su estructura física varía en función del sistema de persistencia en el que se encuentre implantado; por ejemplo, si el *Repositorio de Metadatos* se encontrara implantado sobre un gestor de bases de datos relacionales, su estructura estaría formada por tablas, o si por el contrario se encontrara implantado sobre un gestor NoSQL, su estructura estaría formada por documentos XML o documentos JSON.

Por otra parte, el contenido del *Repositorio de Metadatos* depende del sistema de persistencia en el que se encuentre implantada la base de acontecimientos a la que describe; por ejemplo, si la base de acontecimientos estuviera implantada sobre un gestor de bases de datos relacionales, la información almacenada en el *Repositorio de Metadatos* referiría a tablas, campos, vistas, relaciones entre tablas, etc. de la base de datos.

Caso de implementación y de aplicación

En la figura que mostramos a continuación, puede verse el esquema del *Repositorio de Metadatos* publicado en [25] definido en la prueba de concepto; concretamente, la figura representa una parte del esquema del repositorio de metadatos poblado, en este caso, con información que describe la base de acontecimientos del sistema *Arasis*, la cual se encuentra implantada sobre un sistema gestor de bases de datos relacionales, al igual que el propio repositorio de metadatos.

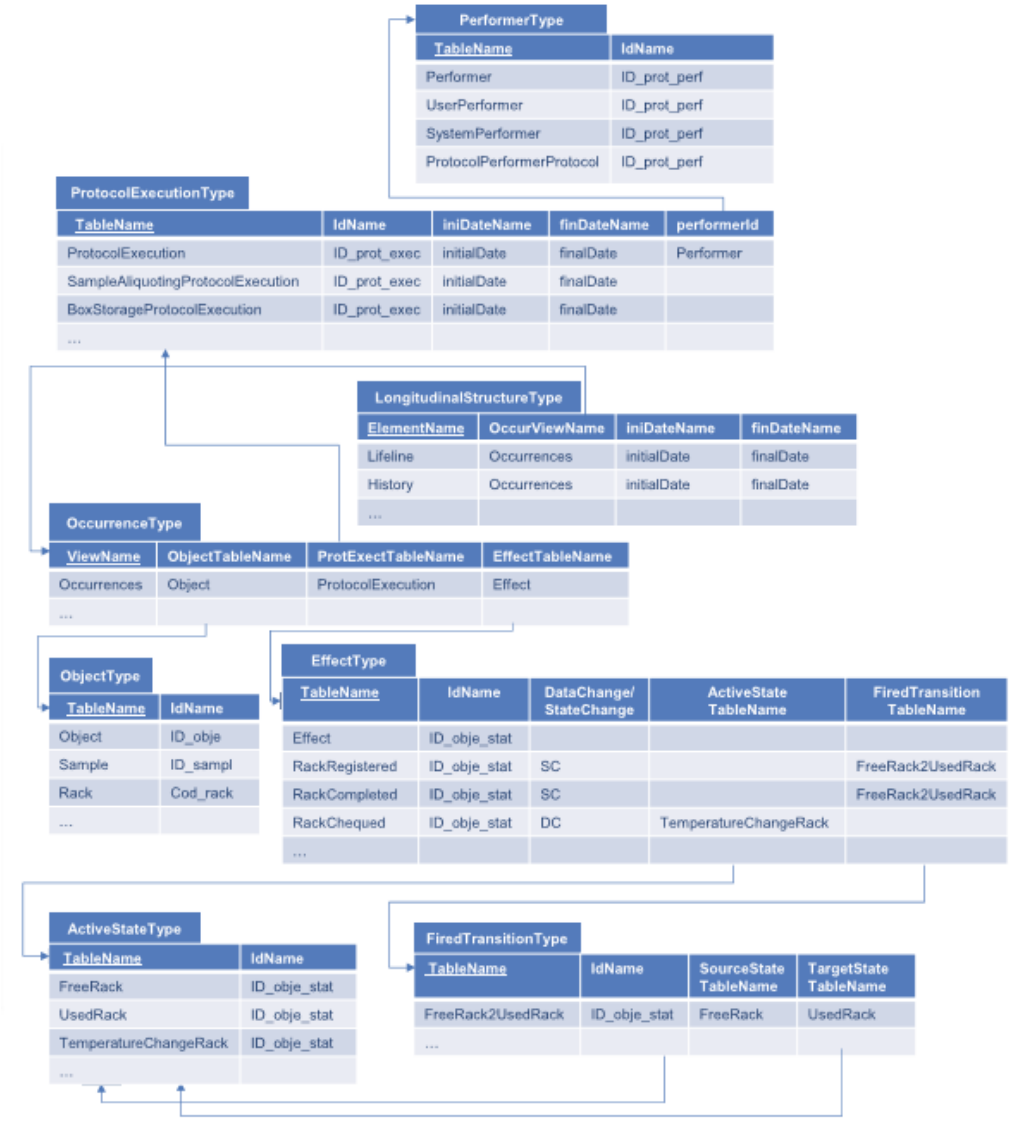


Figura 8.1 Repositorio de metadatos poblado con datos de la base de acontecimientos de *Arasis*

Como puede observarse, la estructura del *Repositorio de Metadatos* está basada en el diagrama de conceptos del metamodelo que hemos representado en la Figura 6.2.

Si comparamos la estructura relacional mostrada en la Figura 8.1 con los conceptos presentados en la Figura 6.2 podemos ver como ambas representaciones se encuentran alineadas, pues en la primera podemos observar como conceptos tales como *Tipo de Ejecución de Protocolo*, *Tipo de Acontecimiento*, *Tipo de Objeto*, etc. han sido recogidos respectivamente como las tablas *ProtocolExecutionType*, *OccurrenceType*, *ObjectType*, etc.

Además, comparando ambas figuras, también puede observarse que el contenido del *Repositorio de Metadatos* es dependiente de la *Base de Acontecimientos* a la que describe.

Como hemos podido ver, el *Repositorio de Metadatos*, además de los conceptos del metamodelo, contempla, como parte de sus estructuras, los campos necesarios para registrar la información de la *Base de Acontecimientos*, concretamente, en el caso de la Figura 8.1:

- La entidad **OccurrenceType** dispone de los campos *ViewName*, *ObjectTableName*, *ProtExectTableName* y *EffectTableName* que almacenan respectivamente el nombre de la estructura de la base de acontecimientos que contiene la información sobre los acontecimientos, el nombre de la tabla de referencia que almacena los objetos afectados por los acontecimientos (Object), el nombre de la tabla de referencia que almacena las ejecuciones de protocolo realizadas (ProtocolExecution), y el nombre de la tabla de referencia que almacena los efectos producidos sobre cada objeto (Effect); es decir información sobre la terna de elementos que definen el concepto *Acontecimiento*.

  Hablamos de *tablas de referencia* debido a que, la base de acontecimientos de *Arasis*, se diseñó contemplando jerarquías de tablas denominadas *relaciones ISA* [9, 35], existiendo como parte de la misma las denominadas tablas *padre*, a las que en el párrafo anterior hemos denominado tablas de referencia (Performer, Object, ProtocolExecution, etc.), así como unas tablas *hija* de las mismas (UserPerformer, SampleAliquotingProtocolExecution, Sample, etc.).
- La entidad **LongitudinalStructureType** dispone de los campos *ElementName*, *OccurViewName*, *IniDateName* y *FinDateName*. El campo *ElementName* almacena el nombre con el que es conocida en el sistema la estructura que representa a cada tipo de estructura longitudinal, el campo *OccurViewName* refiere al nombre de la estructura de la base de acontecimientos que contiene la información sobre los acontecimientos, mientras que los campos *IniDateName* y *FinDateName* contienen los nombres de los campos que almacenan las fechas de inicio y de fin de cada acontecimiento propio de cada tipo de estructura longitudinal.
- La entidad **PerformerType** dispone de los campos *TableName* e *IdName* en los que se registran respectivamente el nombre de las tablas que almacenan información sobre los ejecutores de protocolo y el nombre de los campos de las mismas que actúan como claves primarias.
- La entidad **ProtocolExecutionType**, que refiere al concepto *Ejecución de Protocolo* del concepto *Acontecimiento*, dispone de los campos *TableName*, *IdName*, *IniDateName*, *FinDateName* y *PerformerId* para permitir almacenar respectivamente el nombre de las tablas que registran ejecuciones de protocolos, el nombre de los campos que actúan como clave primaria de dichas tablas, los nombres de los campos que registran las fechas de ejecución, así como el nombre del campo que relaciona las ejecuciones de protocolo con sus ejecutores.
- La entidad **ObjectType**, que refiere al concepto *Objeto* del concepto *Acontecimiento*, dispone de los campos *TableName* e *IdName* en los que se registran respectivamente el nombre de las tablas que almacenan información sobre objetos y el nombre de los campos de las mismas que actúan como claves primarias.
- La entidad **EffectType**, que refiere al concepto *Efecto* del concepto *Acontecimiento*, dispone de los campos *TableName*, *IdName*, *DataStateChange*, *ActiveStateTableName* y *FiredTransitionTableName*. El campo *TableName* almacena los nombres de las tablas en las que se registran los efectos producidos sobre cada uno de los objetos, el campo IdName almacena el nombre de los campos de dichas tambas que actúan como claves primarias, el campo *DataStateChange* se utiliza para registrar el *Tipo de Efecto* al que refiere cada una de las tablas (cambio de estado o cambio en los datos), el campo *ActiveStateTableName* almacena el nombre de las tablas en las que se registra información sobre los estados activos en los que se han producido cambios en los datos; por último, el campo *FiredTransitionTableName* almacena, en el caso de los cambios de estado, el nombre de las tablas en las que se registra información sobre la transición entre estados. En el apartado

6.8 se han explicado cada uno de los conceptos aquí presentados.

- La entidad **ActiveStateName** dispone de los campos *TableName* e *IdName* en los que se registran respectivamente el nombre de las tablas que almacenan información sobre los estados activos en los que pueden encontrarse cada uno de los objetos, y el nombre de los campos de las mismas que actúan como claves primarias.
- Por último, la entidad **FiredTransitionType**, dispone de los campos *TableName*, *IdName*, *SourceStateTableName* y *TargetStateTableName*. El campo *TableName* almacena los nombres de las tablas que registran información sobre las transiciones entre estados, el campo *IdName*, almacena el nombre de los campos de dichas tablas que actúan como claves primarias; por último, los campos *SourceStateTableName* y *TargetStateTableName* refieren respectivamente a aquellas tablas que almacenan los estados activos en los que pueden encontrarse o haberse encontrado los objetos, que actúan como estado origen (*source*) y estado destino (*target*) de la transición entre estados.

  Por ejemplo, en la base de acontecimientos descrita por el repositorio de metadatos que estamos explicando, los objetos de tipo *Rack* pueden encontrarse en dos posibles tipos de estado activo, *FreeRack* o *UsedRack*; la tabla que registra la transición entre estos dos tipos de estado activo es *FreeRack2UsedRack*; cuando se dispara una transición de tipo *FreeRack2UsedRack*, el tipo de estado del que se parte es *FreeRack*, mientras que el tipo de estado al que se llega es *UsedRack*.

8.6 Traducción de consultas OcAL en consultas SQL

El desarrollo de un traductor de consultas OcAL debe tener en cuenta los siguientes aspectos:

1 El lenguaje de consultas nativo para acceder a una base de acontecimientos implantada sobre un gestor de bases de datos relacionales. En este caso, el lenguaje seleccionado fue SQL.

  Por ejemplo, la base de acontecimientos de *Arasis* se encuentra implantada sobre un gestor de bases de datos relacionales, concretamente PostgreSQL, por lo que el lenguaje de consultas compatible con el mismo es SQL.

2 La forma en la que se implementó el esquema del *Repositorio de Metadatos*, la cual hemos explicado en el apartado anterior.

3 La sintaxis y semántica del *Lenguaje de Acceso a Acontecimientos* (OcAL) particularizada en función de cómo quedaron fijadas las dimensiones abiertas del lenguaje. Recordemos que de ello hemos hablado en el apartado 8.3 de este mismo capítulo.

En los siguientes apartados vamos a presentar la traducción de algunos ejemplos de consultas sobre la base de acontecimientos de *Arasis* para cada uno de los tipos de consulta del *Lenguaje no generalizado de Acceso a Acontecimientos* que hemos descrito en el capítulo anterior: (1) Consultas sobre estructuras longitudinales (*Longitudinal Structure Queries*) y (2) Consultas sobre acontecimientos (*Occurrences Queries*). Estas consultas han sido traducidas de forma automática con un algoritmo de traducción que se ha implementado en el seno del *grupo Nóesis* y que se describe en [25]. A través de estos ejemplos se quiere hacer patente la potencia del lenguaje, al comparar las sentencias OcAL y las sentencias SQL generadas.

8.6.1 Traducción de consultas sobre estructuras longitudinales

Durante el proceso de traducción de consultas sobre estructuras longitudinales (*Longitudinal Structure Queries*) es importante tener presente que la consulta SQL resultante deberá acceder a la vista `Lifeline` y a la función `History`, las cuales, debe ser referidas como parte de la cláusula `FROM` de dicha consulta.

La traducción de una consulta de este tipo resulta en una consulta SQL por cada palabra reservada `INITIAL OCCURRENCE` o `FINAL OCCURRENCE`; esto quiere decir que las consultas OcAL con ambas palabras reservadas se traducirían en dos consultas SQL unidas por una cláusula `UNION` (una correspondiente a la traducción de la consulta SQL derivada del uso de la palabra reservada `INITIAL OCCURRENCE` y otra correspondiente a la traducción de la consulta derivada del uso de la palabra reservada `FINAL OCCURRENCE`), mientras que las consultas con sólo una de las cláusulas se traducirían en una única sentencia SQL.

A continuación, se muestra una consulta sobre estructuras longitudinales que permite recuperar el acontecimiento inicial en la línea de vida de una muestra identificada como 'bloodSample7'. Debajo se muestra la consulta SQL generada aplicando el algoritmo de traducción:

```
SELECT INITIAL OCCURRENCE OF LIFELINE
RELATED TO OBJECT 'bloodSample7' OF TYPE Sample
```

```
1  SELECT o.*
2  FROM Lifeline o,
3      Sample trwl,
4      OBJECT ob
5  WHERE o.ID_obje = 'bloodSample7'
6      AND ob.ID_sampl = trwl.ID_sampl
7      AND o.ID_obje = ob.ID_obje
8      AND o.initialDate = (SELECT MIN(initialDate)
                            FROM Lifeline WHERE ID_object = 'bloodSample7')
```

Consulta 8.1 Ejemplo de traducción de consulta OcAL sobre estructuras longitudinales

En este caso, dado que la consulta contiene las palabras reservadas `LIFELINE` e `INITIAL OCCURRENCE`, se generan las instrucciones de las líneas 1, 2 y 8 de la consulta SQL. De este modo se selecciona toda la información de la vista `Lifeline` cuyo valor en el campo `initialDate` corresponda con el valor mínimo que toma dicho campo en la línea de vida del objeto 'bloodSample7'.

Además, como en la cláusula `RELATED TO` aparece el objeto 'bloodSample7' (`RELATED TO Object`), se incluye la línea 5 de la instrucción SQL.

Por último, dado que la consulta OcAL incluye la partícula `OF TYPE Sample`, como parte de la cláusula `RELATED TO`, se introducen las instrucciones que podemos ver en las líneas 3, 4, 6 y 7 de la Consulta 8.1.

8.6.2 Traducción de consultas sobre acontecimientos

Durante el proceso de traducción de consultas sobre acontecimientos (*Occurrences Queries*), el elemento clave es la vista `Occurrence`; en este caso, la estructura de la consulta SQL resultante del proceso de traducción dependerá de los elementos que hayan sido referidos en las cláusulas `SELECT` y `RELATED TO`.

Las consultas OcAL que incluyan la palabra reservada `ACTIVE STATE`, ya sea como parte de la cláusula `SELECT` o como parte de la cláusula `RELATED TO`, se traducen como un conjunto de consultas SQL separadas por la cláusula `UNION`, mientras que las consultas que no incluyan esta palabra reservada se traducen como una única consulta SQL.

A continuación, se muestra una consulta sobre acontecimientos que permite seleccionar un conjunto de objetos de tipo *Sample* (muestra), afectados por la ejecución del protocolo identificado como 'sampleAliquotingProtocoloExecution002' de tipo SampleAliquotingProtocolExecution en caso de que esta se haya realizado a lo largo del año 2016. Debajo de ella se muestra la consulta SQL generada aplicando el algoritmo de traducción:

```
SELECT OBJECT OF TYPE Sample
RELATED TO PROTOCOL EXECUTION 'sampleAliquotingProtocolExecution002'
    OF TYPE SampleAliquotingProtocolExecution
BETWEEN '01/01/2016' AND '31/12/2016'
```

```
1  SELECT o.ID_obje
2  FROM Occurrence o,
3      Sample ts1,
4      OBJECT ob,
5      SampleAliquotingProtocolExecution trw1
6  WHERE o.ID_obje = ob.ID_obje
7      AND ob.ID_sampl = ts1.ID_sampl
8      AND o.ID_prot_exec = trw1.ID_prot_exec
9      AND o.ID_prot_exec = 'sampleAliquotingProtocolExecution002'
10     AND o.initialDate >= '01/01/2016' AND o.finalDate <= '31/12/2016'
```

Consulta 8.2 Ejemplo de traducción de consulta OcAL sobre acontecimientos

Como tercer ejemplo, mostramos una consulta sobre acontecimientos que incluye la palabra reservada `ACTIVE STATE`; concretamente, esta consulta posibilita seleccionar los estados activos a los que se ha llegado por ejecuciones de protocolo realizadas por el ejecutor identificado como 'laboratoryAssistant001':

```
SELECT ACTIVE STATE
RELATED TO PERFORMER 'laboratoryAssistant001'
```

```
1  SELECT s.ID_obje_stat
2  FROM Occurrence o, TemperatureChangeRack s, RackChequed e, ProtocolExecution pe
```

```
3  WHERE s.ID_obje_stat = e.ID_obje_stat AND e.ID_obje_stat = o.ID_obje_stat
4      AND o.ID_prot_exec = pe.ID_prot_exec AND pe.ID_prot_perf = 'laboratoryAssistant001'
5  …
6  UNION
7  SELECT t.ID_prev_obje_stat
8  FROM Occurrence o, FreeRack s, FreeRack2UsedRack t, RackRegistered e,
9      ProtocolExecution pe
10 WHERE s.ID_obje_stat = t.ID_obje_stat AND t.ID_obje_stat = e.ID_obje_stat
11     AND e.ID_obje_stat = o.ID_obje_stat
12     AND o.ID_prot_exec = pe.ID_prot_exec AND pe.ID_prot_perf = 'laboratoryAssistant001'
13 UNION
14 SELECT t.ID_obje_stat
15 FROM Occurrence o, UsedRack s, FreeRack2UsedRack t, RackCompleted e,
16     ProtocolExecution pe
17 WHERE s.ID_obje_stat = t.ID_obje_stat AND t.ID_obje_stat = e.ID_obje_stat
18     AND e.ID_obje_stat = o.ID_obje_stat
19     AND o.ID_prot_exec = pe.ID_prot_exec AND pe.ID_prot_perf = 'laboratoryAssistant001'
19 …
20
```

Consulta 8.3 Ejemplo de traducción de consulta OcAL con la palabra reservada ACTIVE STATE

Como hemos podido observar, el resultado de la traducción de consultas OcAL que incluyen la palabra reservada `ACTIVE STATE` consiste en un conjunto de consultas SQL relacionadas a través de la cláusula `UNION`.

Antes de continuar, recordemos que, entre los diferentes tipos de estado activo que pueden asociarse con un ejecutor, están aquellos que desencadenan un cambio en los datos y aquellos que desencadenan un cambio de estado, los cuales corresponden con los tipos de efecto representados respectivamente por las clases *DataChangeType* y *StateChangeType* del metamodelo. Estos conceptos han sido presentados en el apartado 6.8.

La traducción de consultas sobre acontecimientos que incluyen la palabra reservada `ACTIVE STATE` transcurre de la siguiente forma:

1 Se genera una consulta SQL diferente por cada tipo de estado activo con el objetivo de poder seleccionar aquellos estados activos de cada tipo con los que el ejecutor está relacionado.

2 Se relaciona cada una de las consultas obtenidas a través de la cláusula SQL `UNION`.

En la Consulta 8.3, la tabla *TemperatureChangeRack* almacena información de un tipo de estado activo que desencadena un cambio en los datos (líneas de la 1 a 4), mientras que las tablas *FreeRack* y *UsedRack* representan a tipos de estado activo que desencadenan cambios de estado y actúan respectivamente como estados origen y destino de la transición *FreeRack2UsedRack* que se desencadena (líneas de la 7 a la 12, y de la 14 a la 19). En relación a esto, es importante señalar que estas tablas corresponden con las registradas en el repositorio de metadatos cuyo esquema se ha representado en la Figura 8.1. También es importante señalar que la consulta SQL que hemos mostrado es un extracto de la que se obtendría realmente la cual referiría a todas las tablas que almacenan información de cada uno de los tipos de estado activo contemplados en el sistema.

8.7 Análisis y evaluación de la implementación

En este apartado se exponen las fortalezas y debilidades de las propuestas que hemos presentado hasta ahora:

1 En primer lugar, se exponen las propiedades del *Framework OcQF* para posibilitar la consulta de la procedencia en *Sistemas de Gestión de Acontecimientos* (*Occurrence Management Systems*; *OcSystems*).

2 En segundo lugar, con el objetivo de evaluar la efectividad de la traducción, se muestran los resultados del análisis de algunas de las propiedades (longitud y tiempo de ejecución) de las consultas SQL generadas por el algoritmo de traducción.

Sobre el framework

Desde el punto de vista estructural, una de las ventajas de la propuesta realizada reside en los dos componentes predefinidos que forman parte del framework que recordemos son:

1 El *Lenguaje de Consultas para la Gestión de Acontecimientos* (*Occurrence Query Language; OcQL*.

2 El *Metamodelo de Bases de Acontecimientos* (*Occurrence Base Metamodel*).

Por una parte, está demostrado que el uso de metamodelos mejora el rigor y la comunicación entre desarrolladores y facilita la integración de sistemas y la interoperabilidad entre ellos [45, 46]. Por otra parte, el hecho de que el lenguaje esté definido estrictamente de acuerdo a los conceptos establecidos en el metamodelo propuesto, promueve su correcto uso en sistemas que se definan en base a dicho metamodelo.

A través de los componentes predefinidos del framework propuesto, es posible distinguir entre diferentes tipos de participantes (por ejemplo, desarrolladores, modeladores y usuarios finales) con roles claramente definidos, lo que posibilita que cada tipo de participante se centre en sus responsabilidades específicas y se eviten solapamientos de tareas tal y como hemos podido ver en el apartado 3.3. En esta misma línea, es importante señalar que una vez que se ha establecido la estrategia de diseño para la definición de bases de acontecimientos, los responsables del desarrollo únicamente necesitan implementar un traductor para realizar la traducción de las consultas al lenguaje que corresponda con el sistema de persistencia sobre el que está implantada la base de acontecimientos (por ejemplo, SQL). Con esto queremos decir que un mismo traductor puede ser utilizado para todas las bases de acontecimientos definidas siguiendo la misma estrategia de diseño.

Un potencial inconveniente (que depende de la estrategia de diseño seleccionada para definir la base de acontecimientos) de la propuesta realizada, es que en un determinado modelo de base de acontecimientos puede definirse un número elevado de clases de ejecución de protocolo, clases de objeto y clases de efecto y, muy probablemente, cada una de ellas acabe siendo implementada, por ejemplo, como una tabla en la base de acontecimientos. Esta situación podría resultar en consultas excesivamente grandes como resultado de la traducción de las consultas del lenguaje de acontecimientos. Sin embargo, dado que los usuarios finales no necesitan trabajar con las consultas generadas por el traductor, esta cuestión no debería suponer una merma en la usabilidad y manejabilidad del framework.

En relación a la funcionalidad, el principal objetivo del *Framework OcQF* y su principal ventaja, es que posibilita definir sistemas preparados para gestionar la procedencia de los datos

(*Provenance-aware Systems*) basados en el concepto *Acontecimiento*. A través del framework, los usuarios finales pueden fácilmente formular potentes consultas que serían extremadamente complejas de formular en otro tipo de sistemas. Por ejemplo, como apoyo a las investigaciones del proyecto AWHS [13], es posible obtener la línea de vida completa de las muestras biológicas extraídas a empleados de General Motors España, así como la historia del almacenamiento de las alícuotas generadas a partir de las mismas en tubos, cajas, racks y crioviales.

Adicionalmente, la propuesta realizada logra liberar a los usuarios finales tanto de la dependencia de los administradores del sistema de persistencia correspondiente (por ejemplo, de los administradores de bases de datos) que habitualmente son responsables de facilitar el acceso y recuperar los datos, como de los formularios prediseñados que únicamente permiten obtener un conjunto predeterminado de datos.

La filosofía que hay detrás de la definición del lenguaje de consultas, ofrece potenciales beneficios:

1 El lenguaje se ha definido a nivel conceptual, lo que tiene por objetivo elevar el nivel de abstracción en los sistemas de consultas existentes.

2 El lenguaje utiliza una sintaxis independiente de la tecnología, de tal forma que los usuarios finales no tengan que estar familiarizados con los detalles de la implementación del sistema de persistencia sobre el que se encuentre implantada la correspondiente base de acontecimientos.

3 El lenguaje utiliza expresiones fáciles de utilizar (`SELECT`, `RELATED TO`, `BETWEEN`, etc.) junto con términos para referir a los conceptos y funciones que son utilizados de forma habitual en su dominio.

4 La variante no generalizada del lenguaje posibilita que las consultas sean realizadas a nivel de instancia (de dato), mientras que la variante generalizada posibilita la definición de consultas sobre los tipos de elemento (tipo de objeto, tipo de ejecución de protocolo, etc.) que serán creados, incluyendo consultas que combinan y relacionan tanto tipos como instancias, ofreciendo una forma de obtener información enriquecida sobre la procedencia.

Evaluación del rendimiento de las traducciones

La evaluación de las traducciones se ha realizado de la siguiente forma:

1 Se ha comparado la longitud y complejidad de un conjunto de consultas de acontecimientos con la longitud de las consultas SQL resultantes de su traducción.

2 Se han realizado diversos experimentos para medir el rendimiento en términos de tiempo de ejecución de las consultas SQL generadas.

En relación al primer punto, a través de los ejemplos que se han ido mostrando a lo largo de este texto, ha podido observarse como, claramente, la longitud de las consultas del lenguaje de acontecimientos es mucho menor que la longitud de las consultas SQL equivalentes. Es especialmente reseñable el caso de las consultas que incluyen la palabra reservada `ACTIVE STATE`; recordemos que, en este caso, las consultas SQL resultantes del proceso de traducción están formadas por múltiples cláusulas `SELECT` unidas por la cláusula `UNION`. Por ejemplo, la traducción completa de la Consulta 8.3 genera una consulta SQL formada por 54 cláusulas `SELECT`. En general, se puede decir que el lenguaje posibilita a los usuarios obtener la

procedencia de la información de una forma extremadamente concisa en comparación con SQL. Esto posibilita que los usuarios finales cometan un menor número de errores durante la especificación de las consultas.

En relación al segundo punto, en los siguientes apartados se presentan los resultados de los tres experimentos que se han realizado con el objetivo de analizar el rendimiento en términos de tiempo de ejecución de las consultas SQL generadas. Estos experimentos se realizaron sobre el sistema gestor de bases de datos *PostgreSQL* en su versión 9.3, instalado sobre un ordenador de sobremesa *HP Compaq 6200 Pro* equipado con un procesador *Intel i5-2400* 3.1 GHz y 12 GB de RAM.

8.7.1 Primer experimento

El primer experimento consistió en la ejecución de una consulta definida para obtener la historia de un conjunto de empleados:

```
SELECT HISTORY
RELATED TO OBJECT TYPE Subject
```

Consulta 8.4

Se seleccionó este tipo de consulta por una parte porque se observó la construcción de la historia es la actividad que más tiempo consume y, por otra parte, porque la historia de un empleado implica, ya sea directa o indirectamente, a todos los tipos de objeto.

Para realizar este experimento se seleccionaron seis empleados con un número variable de muestras (de una a veinte en el momento de la realización del experimento), con el objetivo de comparar el tiempo de respuesta de la consulta generada cuando el número de instancias implicadas aumenta.

La siguiente tabla muestra los resultados de este experimento, mostrando, para cada empleado (identificado por su código anonimizado de empleado), el número de muestras, el número de objetos derivados (por ejemplo, alícuotas, tubos, cajas, racks o crioviales) implicados en su historia, el número de protocolos ejecutados, el número de resultados obtenidos de la base de datos y, finalmente, el tiempo de respuesta (en milisegundos).

Código de empleado	Nº. de muestras	Nº. de objetos derivados	Nº. de protocolos ejecutados	Nº. de resultados obtenidos	Tiempo (en ms)
865393	1	2	11	13	230
570090	4	36	38	133	520
701941	8	50	59	167	1221
781547	12	48	61	142	1872
512200	16	81	122	260	3106
929570	20	133	176	451	3526

Tabla 8.2 Resultados obtenidos del proceso de construcción de la historia

Por otra parte, la siguiente figura representa el tiempo de respuesta con respecto al número de muestras, incluyendo la línea de tendencia (*Trend Line*):

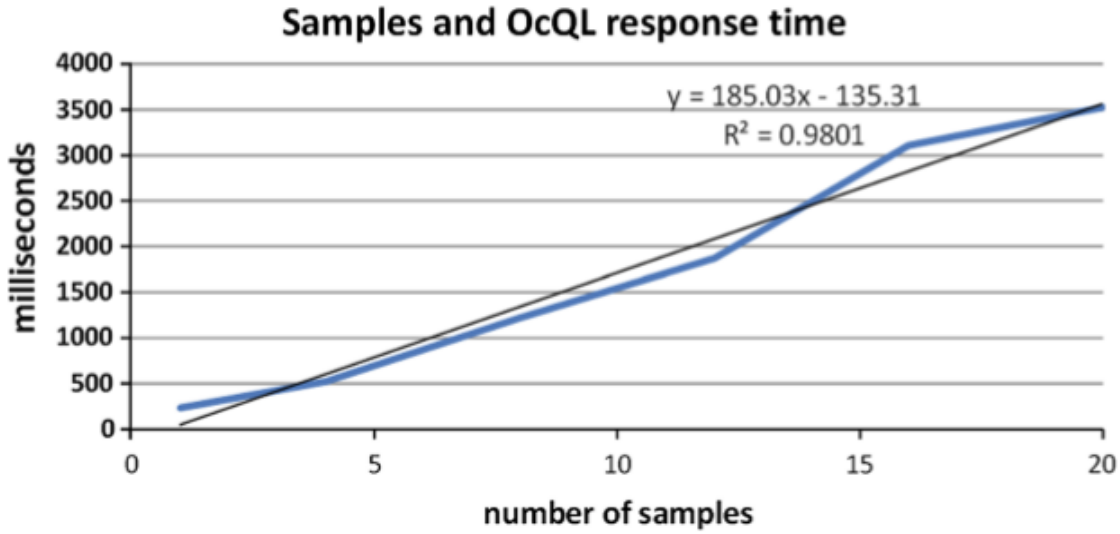


Figura 8.2 Relación entre el número de muestras y el tiempo de respuesta

Obsérvese que el valor del coeficiente de determinación (R^2) es 0,98, lo que significa un buen alineamiento de la línea con los datos. Este resultado permite concluir que el tiempo de respuesta es linealmente dependiente a la complejidad de la historia, lo que puede ser considerado un buen resultado en términos de rendimiento.

8.7.2 Segundo experimento

El segundo experimento consistió en la ejecución de una consulta definida para obtener las ejecuciones del protocolo realizadas por diferentes ejecutores de tipo *UserPerformer*:

```
SELECT PROTOCOL EXECUTION
RELATED TO PERFORMER TYPE UserPerformer
```

Consulta 8.5

Se seleccionó esta consulta por el elevado número de ejecuciones de protocolo que devuelve (de 18.159 a 157.919 en el momento de la realización del experimento).

La siguiente figura representa el tiempo de respuesta de la consulta generada con respecto al número de ejecuciones de protocolo, incluyendo la línea de tendencia:

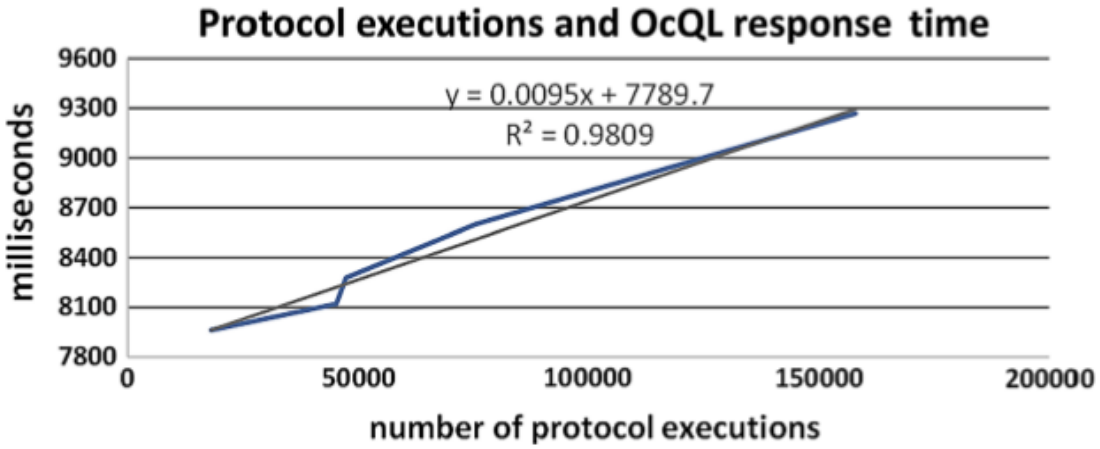


Figura 8.3 Relación entre el número de ejecuciones de protocolo y el tiempo de respuesta

Obsérvese que, al igual que en el experimento anterior, se ha obtenido un coeficiente de determinación (R^2) de 0,98, pudiendo considerarse este un buen resultado en términos de rendimiento.

8.7.3 Tercer experimento

El tercer experimento se dedicó a la obtención de los objetos afectados por un conjunto de ejecuciones de protocolo de tipo *SampleAliquotingProtocolExecution*:

```
SELECT OBJECT
RELATED TO PROTOCOL EXECUTION TYPE SampleAliquotingProtocolExecution
```

Consulta 8.6

Se seleccionó esta consulta por el elevado número de objetos que devuelve.

La siguiente figura representa el tiempo de respuesta de la consulta generada con respecto al número de ejecuciones de protocolo, incluyendo la línea de tendencia:

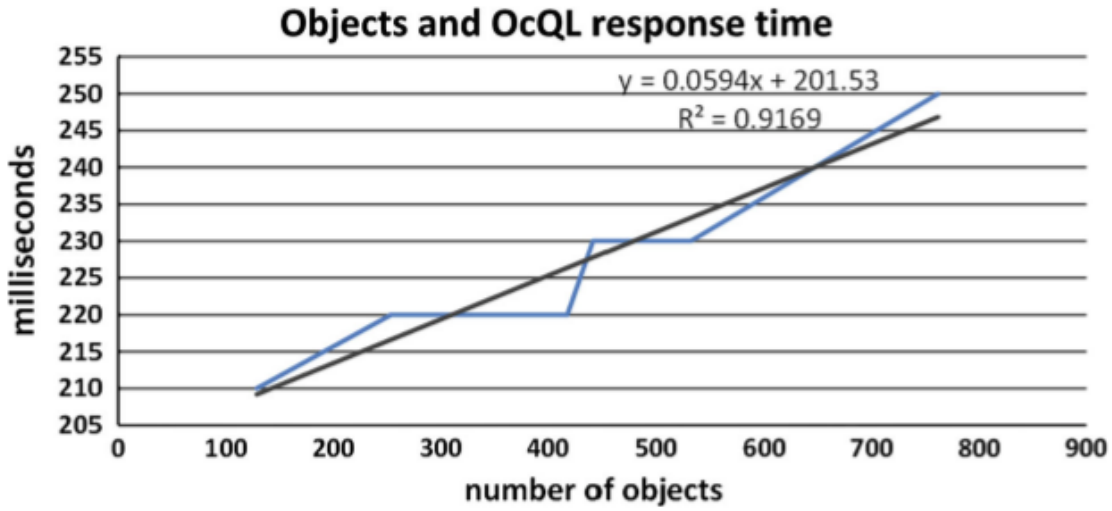


Figura 8.4 Relación entre el número de objetos y el tiempo de respuesta

En este caso, al igual que en los dos experimentos anteriores, se ha obtenido un alto coeficiente de determinación (R^2), concretamente 0,91, quedando demostrado de esta forma el buen rendimiento de los resultados obtenidos.

IV. El Lenguaje de Acceso a Información de Acontecimientos

Capítulo 9. El Metamodelo Extendido de Bases de Acontecimientos

9.1 Introducción

El objetivo de este capítulo es describir una extensión del *Metamodelo de Base de Acontecimientos* (*Occurrence Base Metamodel*; *OcBase Metamodel*), presentado en el Capítulo 6, extendiendo su expresividad dirigida a la gestión de la información de contexto que forme parte de los acontecimientos.

Esta nueva versión que denominamos *Metamodelo Extendido de Base de Acontecimientos* (*Occurrence Base Extended Metamodel*; *OcBase Extended Metamodel*) está especialmente dirigida a disponer de un lenguaje de acceso a información propia de acontecimientos cercano, en lo posible, a un lenguaje natural que permita comunicar expresiones en el marco de las bases de acontecimientos.

El lenguaje asociado, que denominaremos *Lenguaje de Acceso a Información Acontecimientos*, distinguirá entre el subconjunto de expresiones dirigidas a extraer información del diccionario de una base de acontecimientos, del subconjunto de expresiones dirigidos propiamente a la explotación de la información contenida en una base de acontecimientos.

El metamodelo se presenta progresivamente a través de varios apartados mostrando aspectos que, en cierto modo, constituyen un proceso de extensión propio del metamodelo en función de una perspectiva particular sobre la percepción de los acontecimientos.

- En primer lugar, presentamos los conceptos del metamodelo considerados como básicos.
- En segundo lugar, los que pueden ser utilizados para modelar los tipos de cambio de estado que pueden afectar a cada tipo de objeto.
- En tercer lugar, aquellos que posibilitan la expresión de información propia del contexto.
- Por último, presentamos los que posibilitan anclar en el tiempo los acontecimientos.

En cada subapartado se muestra un diagrama de clases que describe el correspondiente sistema de conceptos, añadiendo la descripción de su contenido, las novedades frente al metamodelo no extendido. y aquellos aspectos o elementos no expresados en el diagrama; la no inclusión de estos estos últimos conceptos como parte del diagrama es debida a la limitación de la expresividad de la herramienta utilizada para el dibujo del diagrama o esquema UML.

9.2 Diagrama básico

El metamodelo presentado en el Capítulo 6 de este documento de tesis utiliza una noción de acontecimiento basada en protocolo, donde, como se definió en [24]:

> *"A protocol is the concrete specification of the structured, logical order of activities designed to achieve the goal of a process."*

De ese modo, entonces un acontecimiento se produce por ejecución de un protocolo sobre un objeto, por cuya acción se produce un cambio de estado del objeto.

Esta perspectiva es adecuada en los casos en que el protocolo está siempre definido y es estable en el Universo del Discurso.

Sin embargo, hay situaciones en las que el protocolo es desconocido o varía con frecuencia, por lo que, en estos casos, sería más interesante partir de una noción de acontecimiento basada en procesos, considerando, la siguiente definición:

> *"A process is a concrete, identifiable, end–to– end task performed for a given purpose, conceptualized as a cohesive set of protocols."*

Esta es la perspectiva que se considerará en adelante, el analista-diseñador podrá elegir una u otra según considere cuál es la situación más adecuada a su caso.

El concepto Acontecimiento Basado en Procesos

Concretamente, definimos:

> Un acontecimiento basado en procesos como una pieza de información concreta, identificable e indivisible que contiene aspectos organizados de acuerdo a tres dimensiones: guía, estructura y comportamiento, dados por la ejecución de un proceso, un objeto y un efecto, en el que el proceso al actuar sobre el objeto produce como un cambio de estado de este.

Usos

Desde esta perspectiva, la de acontecimiento basada en procesos, si, por ejemplo, se pretendiera construir un sistema informático que posibilitara el registro de acontecimientos que no han sido producidos por la ejecución de un proceso (por ejemplo, fenómenos naturales como la lluvia), se aplicaría una estrategia de diseño que no recogiera los conceptos *Tipo de Proceso Ejecutado*, *Tipo de Proceso* y *Tipo de Protocolo*.

Por otra parte, si, por ejemplo, se pretendiera construir un sistema donde no se considerara necesario identificar los protocolos utilizados al ejecutar un proceso, no se contemplaría el concepto *Tipo de Protocolo*. Esta flexibilidad no era posible con la versión original del metamodelo, donde la eliminación del concepto *Tipo de Protocolo* obligaba a hacer lo propio con el concepto *Tipo de Proceso*.

El diagrama

En la Figura 9.1 mostramos un diagrama UML que presenta la parte del metamodelo que contiene los conceptos básicos.

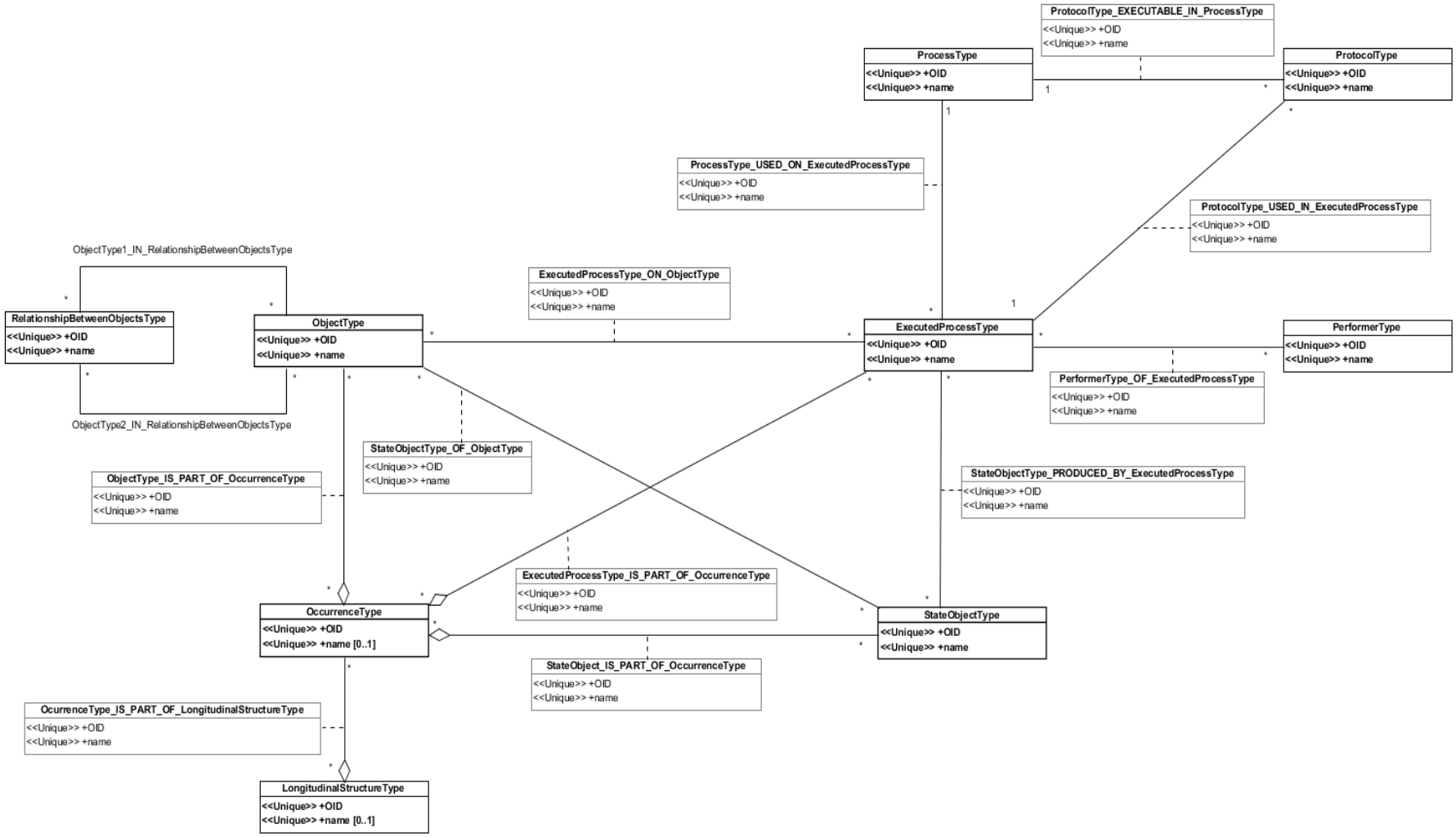


Figura 9.1 Conceptos básicos del Metamodelo Extendido de Bases de Acontecimientos

A continuación, señalamos los cambios que se han realizado en relación al metamodelo no extendido.

Incorporación de los OID y name

En todas las clases y en determinadas asociaciones se han incorporado explícitamente los atributos *OID* (*Object Identifier*; *Identificador de Objeto* o *IDO*) y *name* con el siguiente sentido:

OID es un identificador de objeto en el sistema de cada una de las instancias tipo expresadas en el diagrama anterior.

Por otra parte, *name* refiere a un nombre descriptivo de una instancia de un tipo que debe ser único dentro del conjunto de instancias de ese tipo.

Por ejemplo, una instancia de *ObjectType* puede ser una cuyo nombre es 'alícuota' y su identificador de objeto '11132'. Ello significa que '11132' identifica a *alícuota*, dentro del sistema, y que *alícuota* es el nombre de un tipo de objeto.

La intención de uso de estos elementos es la identificación de los elementos y asociaciones referidas por el metamodelo, no sólo en el marco de la clase del tipo en el que están explícitos sino también en el ámbito global de todas las posibles instancias. La intención de aplicación de esta regla es disponer en el diccionario; en un contexto de aplicación, de las identificaciones de objeto y descriptiva de los metadatos correspondientes.

Este hecho está reflejado en el diagrama del metamodelo por el uso de los estereotipos

<<Unique>> +OID

<<Unique>> +name

Finalmente es interesante señalar que la incorporación de estos atributos en la especificación del metamodelo facilita la comunicación en el desarrollo de aplicaciones que faciliten su gestión. Por ejemplo, al haber declarado estos atributos, podríamos comunicar la estructura de un determinado modelo de base de acontecimientos ya no sólo gráficamente (por ejemplo, a través de un diagrama de clases UML), sino también textualmente; veamos, por ejemplo, una posible especificación en formato XML del ejemplo que hemos dado más arriba:

```
<model>
   <objectTypes>
      <objectType>
         <oid>11132</oid>
         <name>alícuota</name>
      </objectType>
   </objectTypes>
</model>
```

Materialización del tipo de asociación entre ObjectType

Para la realización de consultas que relacionan acontecimientos entre objetos, es esencial conocer las asociaciones particulares que hay en el sistema entre los tipos de objetos. Por ejemplo, si deseamos relacionar los resultados de análisis clínicos de 'sujetos de estudio' con

las 'alícuotas' existentes en un biobanco sobre dichos sujetos, será necesario conocer el tipo de asociación existentes entre ellos y, para ello, será necesario disponer de ese conocimiento explícito en el diccionario.

Por ello se ha considerado necesario disponer de la referencia a ese tipo de asociaciones en el metamodelo y, en consecuencia, considerar su materialización como clase de objetos bajo el nombre

RelationshipBetweenObjectsType

Bajo dicha perspectiva los diagramas de objetos se consideran tipados por los propios objetos y por los tipos de asociaciones entre ellos. Las relaciones del diagrama son las relaciones de estructura entre un tipo de asociación y los tipos de objetos asociados.

Los tipos de acontecimientos y los tipos de estructura longitudinal como agregaciones

Para facilitar la expresividad de solicitud de información de contexto sobre acontecimientos y estructuras longitudinales se ha considerado procedente considerarlos como estructuras explícitas de agregación tal como se señala en el diagrama anterior.

Un cambio en la perspectiva

Para fortalecer la característica evolutiva del uso del metamodelo, se ha considerado procedente intercambiar el posicionamiento de *ProcessType* y *ProtocolType*. Pudiera ser que en un sistema concreto de aplicación se deseara registrar el proceso general (por ejemplo, *alicuotar*) pero no registrar el protocolo concreto que en cada momento se ha utilizado.

En ese caso, la solución que se propone permite tomar esa decisión y, en un momento dado, evolucionar el sistema para integrar en el mismo los protocolos en uso sin realizar ningún cambio en los acontecimientos previamente registrados.

En relación a los cambios de estado, aunque se incluyen en otra parte posterior del metamodelo, no se han considerado como básicos por razones análogas a las que acabamos de expresar.

Alguna restricción en la expresividad

Para limitar el ámbito de aplicación, no se ha considerado los *DataChangeState* y, por tanto, no se ha considerado la abstracción *EffectType* que tiene en común con *StateChangeType*.

Nominación de las relaciones entre elementos del metamodelo

Otra novedad que incorporamos en el metamodelo extendido, y que destacamos por su importancia, es dotar de nombres a las asociaciones que están especificadas en el diagrama.

La intención de uso de dichos nombres es aumentar la expresividad del lenguaje de acontecimientos para referir de forma unívoca a las asociaciones entre tipos de elementos.

Dichos nombres estarán sujetos a las siguientes restricciones:

- Cada nombre es único en el conjunto de los nombres de las asociaciones del metamodelo.

- El nombre debe referir a los nombres de los elementos asociados y contener el vínculo lingüístico que proporciona la significación de la asociación.

  Por ejemplo, en la siguiente figura

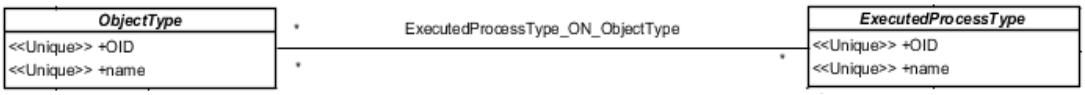


Figura 9.2 Caso de nominación de las relaciones entre elementos del metamodelo

  la asociación especificada está nominada por

  ExecutedProcessType_ON_ObjectType.

  El entorno permite distinguir los nombres de los elementos asociados del vínculo lingüístico que señala a la asociación.

- En el caso de especificaciones textuales es necesario distinguir el nombre de los elementos asociados, utilizando los paréntesis para su discriminación sintáctica, tal como se indica a continuación

  (ExecutedProcessType)_ON_(ObjectType)

9.3 Diagramas de estados

En esta extensión se señalan explícitamente la expresividad para la incorporación explícita de los cambios de estado, señalando en particular los tipos de estado inicial y de estado final, tal y como puede verse en el diagrama que se muestra a continuación.

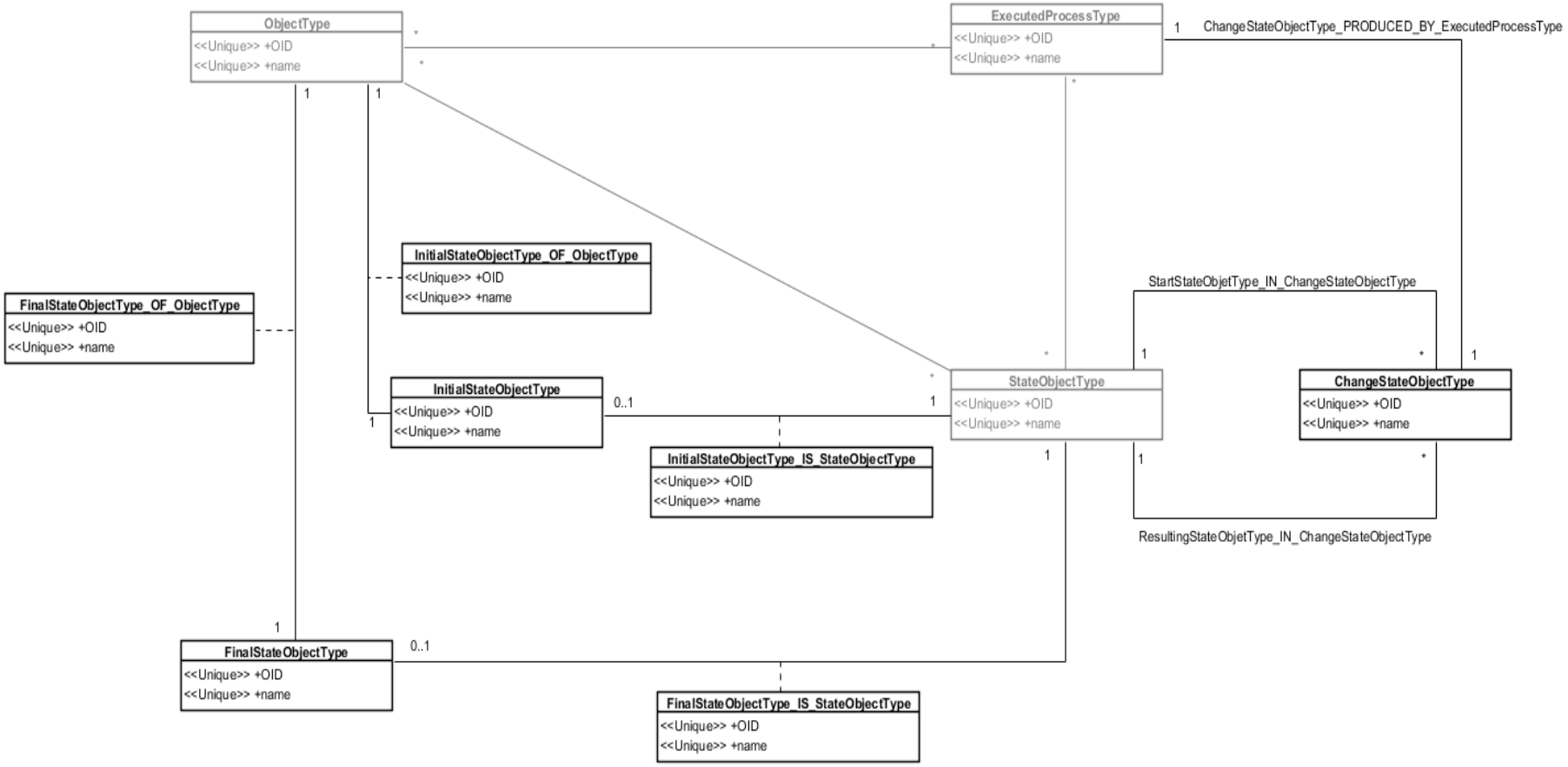


Figura 9.3 Conceptos del Metamodelo Extendido de Bases de Acontecimientos para representar diagramas de estados

9.4 Información dependiente del contexto

Esta parte del metamodelo se orienta a especificar la estructura de los tipos de información que serán dependientes del contexto particular de aplicación.

Por ejemplo, en el caso de una investigación biomédica, 'sujeto de estudio' puede ser un tipo de objeto de la base de acontecimientos, instancia del elemento ObjectType.

Los sujetos de estudio concretos tendrán información, cuya estructura y valor, dependerá del contexto particular de aplicación.

En esta parte se establecen las formas de esos tipos de información que son admisibles, bajo la perspectiva del metamodelo que se presenta y que se describen en el siguiente diagrama.

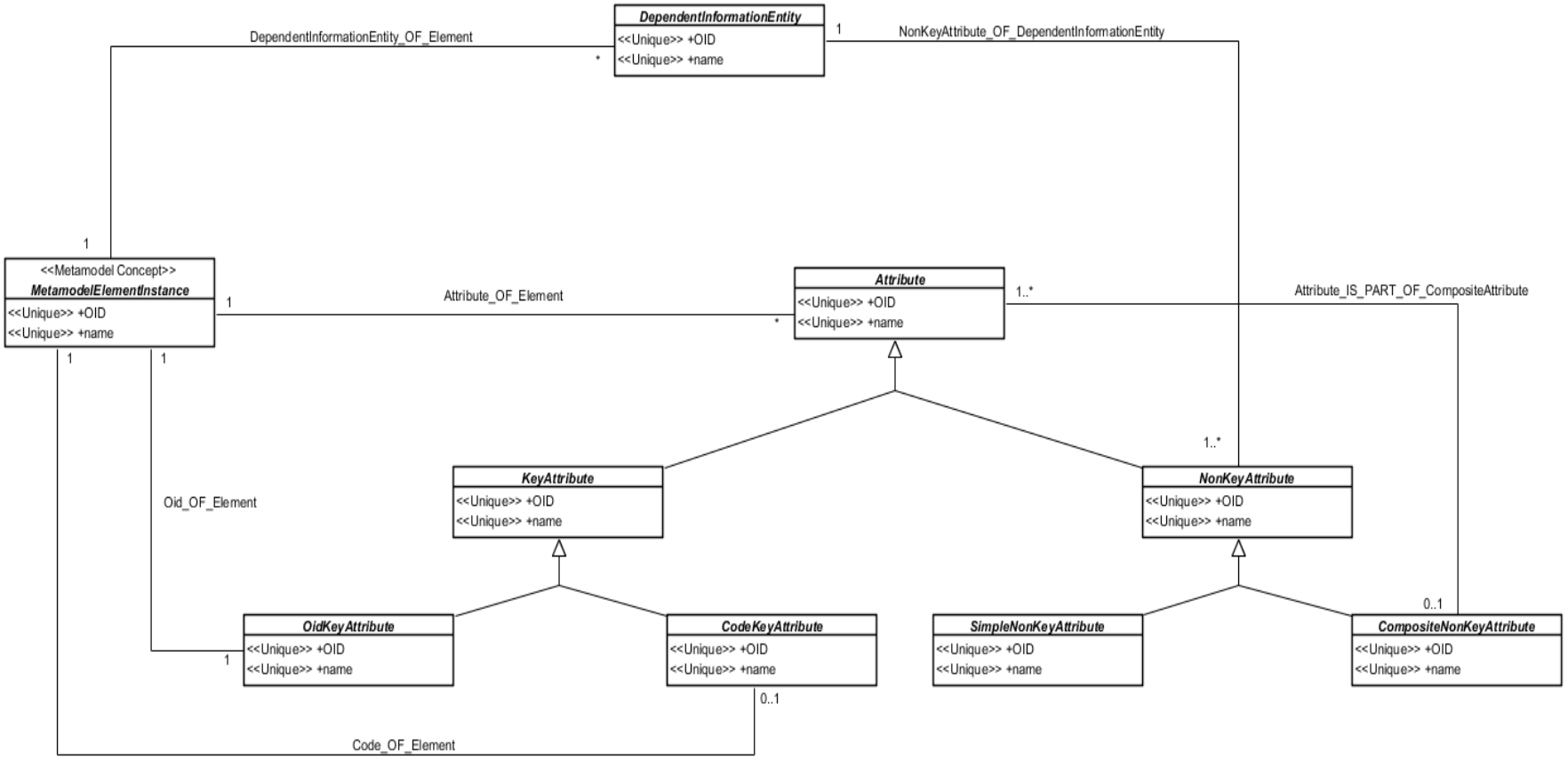


Figura 9.4 Información dependiente del contexto

La instancia de un elemento cualquiera del metamodelo definido en los diagramas anteriores está representada por la clase *MetamodelElementInstance*.

El diagrama anterior representa la estructura de atributos que puede tener uno de dichos elementos.

Los identificadores de objeto (OID) y los identificadores semánticos (name) cumplen las mismas reglas y usos que en los casos expuestos anteriormente. Quizás cabe señalar aquí la siguiente:

Si el sistema de explotación obliga a distinguir una clave como preferente, esta debe ser la instancia de *OidKeyAttribute*.

Vamos a detallar un ejemplo de instancia en relación al anterior diagrama.

Podríamos considerar

('111345', 'sujeto_de_estudio'),

como instancia de ObjectType, instancia que podría tener los siguientes atributos en algún caso de aplicación:

('111000', 'IDO_sujeto'),

como instancia de *OidKeyAttribute*

('111010', 'codigo_sujeto'),

como instancia de *CodeKeyAttribute*

y

('247890', 'datos_generales'),

como instancia de *DependentInformationEntity*

que, a su vez, podría tener el siguiente conjunto de atributos, instancias de *NonKeyAttribute*

('999120', 'nombre_de_sujeto'),

('765000', 'fecha_de_nacimiento')

9.5 Anclaje en el Tiempo

Por último, en este apartado mostramos un caso simple de cómo los acontecimientos se pueden anclar en el tiempo a través de:

Fecha de ejecución del protocolo

Fecha de registro del acontecimiento

Fecha de inicio de un estado de objeto.

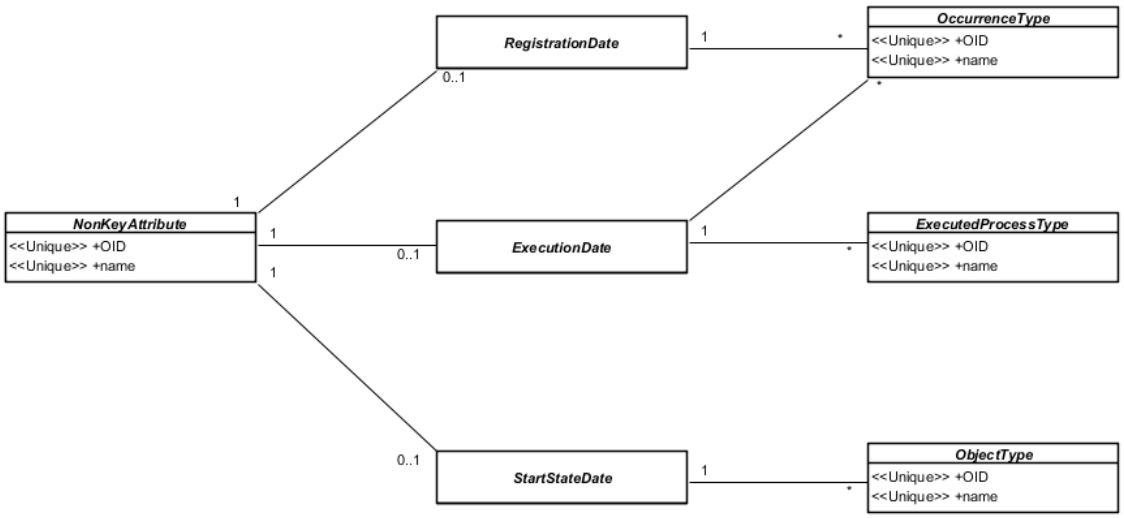


Figura 9.5 Anclaje en el tiempo del metamodelo extendido

Capítulo 10. El Diccionario

10.1 Introducción

El *Diccionario* o *Repositorio de Metadatos*, tal y como lo hemos definido en el apartado 3.2.6, es el componente del *Framework OcQF* que posibilita almacenar el conocimiento expresado en los modelos de bases de acontecimientos, así como el de las instancias de los mismos en forma de bases de acontecimientos. Concretamente, el *Diccionario* almacena información descriptiva de la estructura de una base de acontecimientos resultante de la implementación de un modelo de base de acontecimientos, de los elementos que la componen y de las relaciones entre ellos.

Como vamos a ver en este capítulo, para realizar la implementación del *Diccionario* se ha aplicado la metodología MDA (*Model Driven Architecture*) partiendo de un modelo independiente de la plataforma que utiliza los conceptos y estructura del *Metamodelo de Bases de Acontecimientos* que hemos presentado en el capítulo anterior. A lo largo de este capítulo vamos a mostrar un caso de implementación del *Diccionario* obtenida tras aplicar sucesivas transformaciones sobre el modelo inicial.

Recordemos que la metodología MDA (*Model Driven Architecture*) o *Arquitectura Dirigida por Modelos* propone realizar la especificación de un sistema en forma de modelos; según esta técnica, la funcionalidad de un sistema se define en primer lugar como un modelo independiente de la plataforma (*Platform Independent Model*; *PIM*) que posteriormente se trasforma a un modelo específico de la plataforma (*Platform Specific Model*; *PSM*) sobre el que se realizará la implementación del sistema [55]. Esta técnica fue empleada para desarrollar la *Metodología para la Construcción de Bases de Acontecimientos* que hemos mostrado en el apartado 2.2 y que fue publicada en [24].

10.2 Definición del modelo independiente de la plataforma

Para definir el modelo independiente de la plataforma (PIM) se partió de tres de los sistemas de conceptos del *Metamodelo de Bases de Acontecimientos* que hemos presentado en el capítulo anterior: (1) Del sistema de conceptos básicos (Figura 9.1), (2) del sistema de conceptos para representar cambios de estado (Figura 9.3), y (3) del sistema de conceptos para representar información dependiente del contexto (Figura 9.4). Sobre ellos se realizó lo siguiente:

1 Se fijaron las multiplicidades de los roles de cada relación entre las clases.

   Recordemos que el metamodelo proporciona una serie de conceptos que pueden ser utilizados para la especificación de modelos de bases de acontecimientos; la estructura de

estos modelos se basa en la estrategia de diseño seleccionada en cada caso. Sin embargo, para la definición del modelo independiente de la plataforma del *Diccionario*, la estructura del metamodelo se mantiene y, por tanto, es necesario fijar las multiplicidades de las relaciones entre cada una de las clases al existir una libertad en el metamodelo que, para este caso, es necesario restringir.

2 Se materializaron las relaciones entre las clases.

La materialización es necesaria para asegurar que la independencia perceptiva de las clases asociadas se conserve en los niveles inferiores de abstracción.

3 Aplicando los conceptos del metamodelo para representar información dependiente del contexto tal y como se ha descrito en el apartado 9.4 se especificaron los atributos de cada una de las clases del diccionario.

Concretamente se incluyeron dos atributos: (1) Un atributo para posibilitar el registro de un identificador de cada elemento y asociación de la base de acontecimientos descrita en el diccionario, y (2) un atributo *name* para registrar el nombre de dichos elementos.

4 Se fijaron una serie de reglas para asegurar la consistencia del modelo resultante.

Reglas

Para la especificación del modelo independiente de la plataforma se fijaron las siguientes reglas:

1 Todas las clases que refieren a conceptos del metamodelo, así como ciertas asociaciones que interesaba identificar, se especificaron incluyendo dos atributos: un identificador de objeto (*OID*; *Object Identifier*) declarado como *OID_<nombre_elemento>* u *OID_<asociacion>*, y un nombre (*name*).

El identificador de objeto se utiliza para identificar internamente a los elementos genéricos representados por los conceptos recogidos del metamodelo, así como a las asociaciones.

El nombre (name) se utiliza para identificar en el UoD (Universo del Discurso) al elemento genérico o asociación.

El objetivo sería, por ejemplo, poder registrar en la entidad *ObjectType* del diccionario un tipo de objeto con nombre "Sample" identificado de forma unívoca en el mismo como "0001".

2 Excepcionalmente, aquellos elementos que son utilizados como señales para la localización no incluyen el atributo *name*. Es por ejemplo el caso de las clases que identifican los tipos de estado de objetos que pueden ser iniciales o finales: *InitialStateObjectType* y *FinalStateObjectType*.

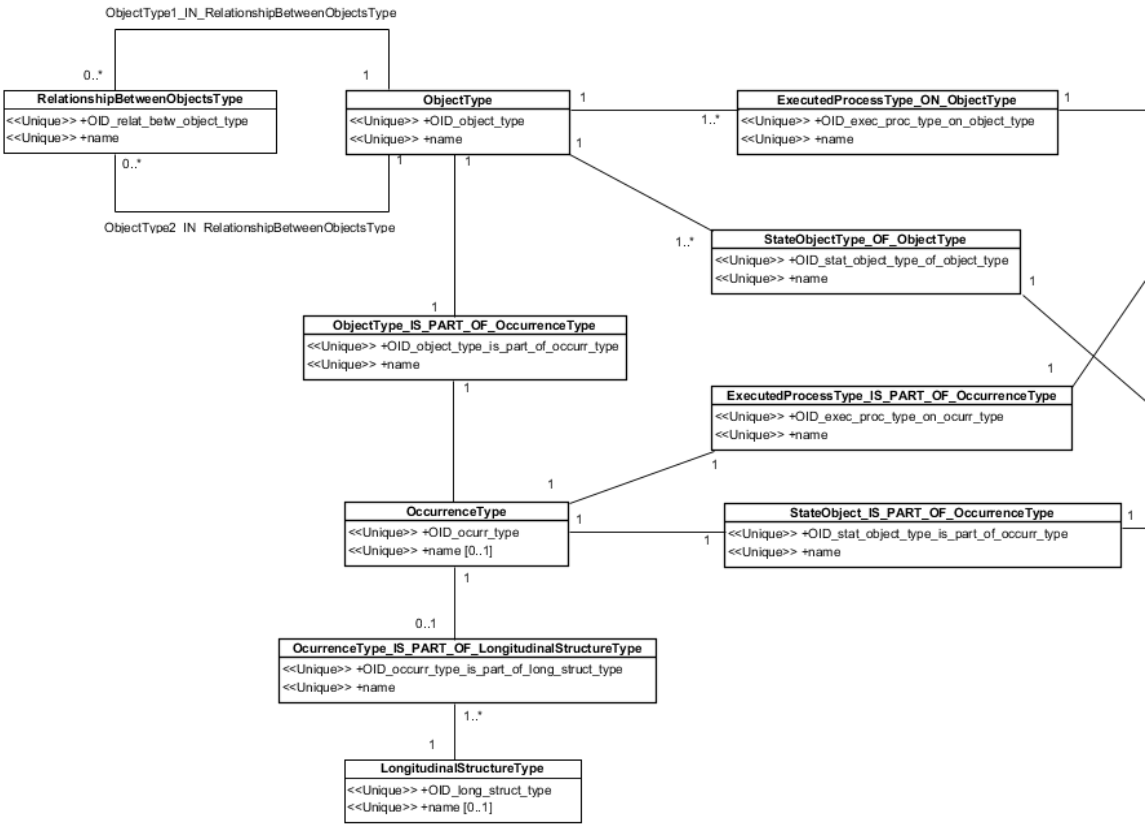
3 El valor del OID de un elemento genérico tiene que identificarlo entre todos los elementos genéricos registrados en una implementación del diccionario; para reflejar esto, los atributos *OID_<nombre_elemento>* se han estereotipado como <<Unique>>. Esto es aplicable también a los correspondientes atributos de las asociaciones.

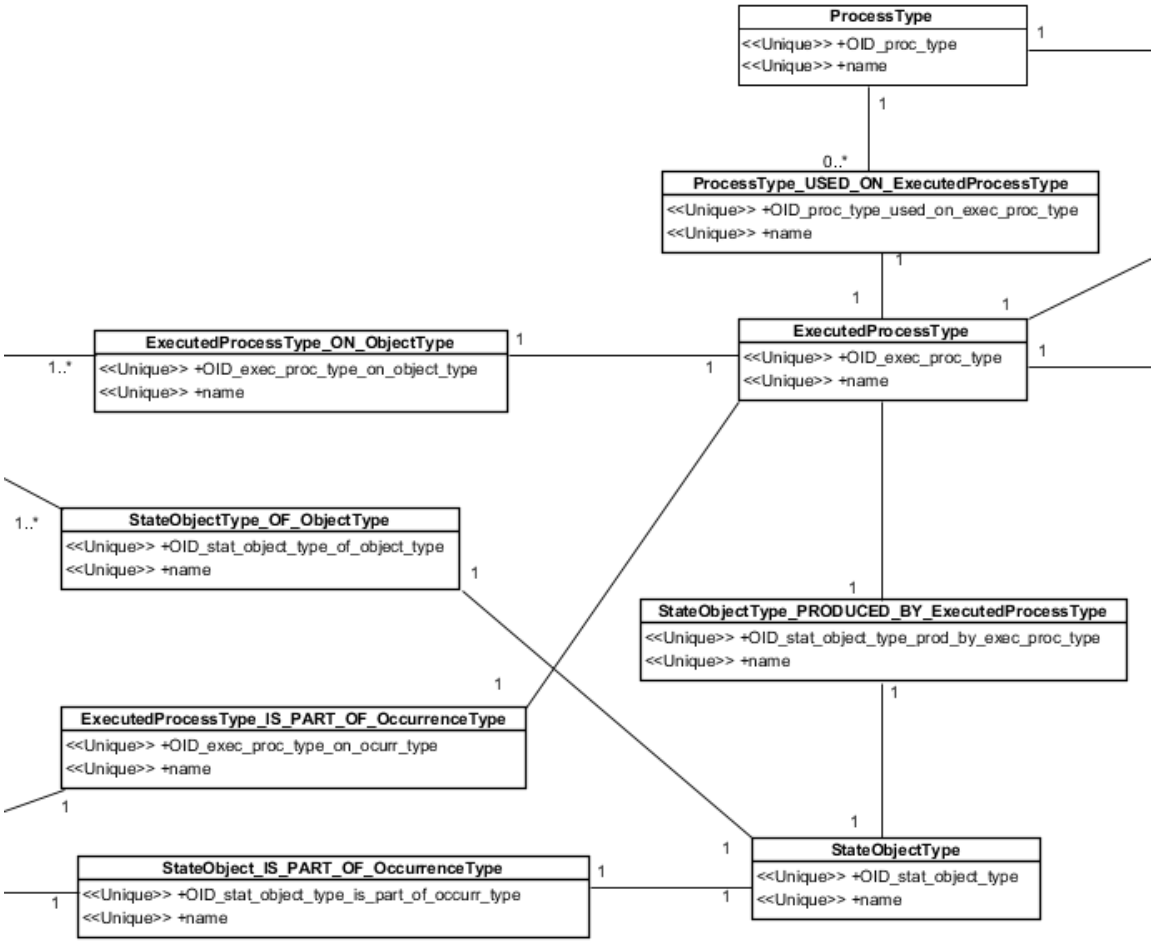
4 El valor del atributo *name* debe ser único entre los nombres de los elementos de un determinado tipo y entre todos los demás elementos registrados en una implementación del diccionario; para reflejar esto, los atributos *name* se han estereotipado como <<Unique>>. Esto es aplicable también a los correspondientes atributos de las asociaciones.

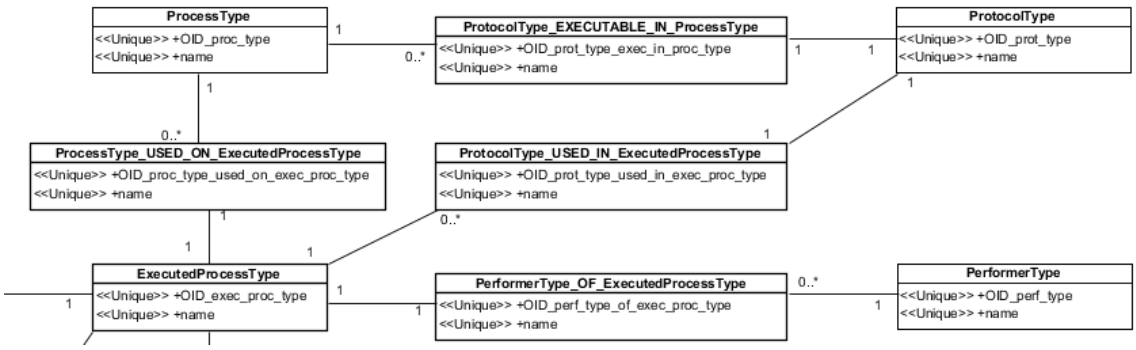
Por ejemplo, si en una implementación del diccionario hubiera registrada una instancia de tipo de objeto (ObjectType) con nombre "Sample", este nombre no podría tenerlo la instancia de un tipo de estado de objeto (StateObjectType); o si "Stored" es un nombre de la instancia de un tipo de estado de objeto, entonces no puede existir una instancia de tipo de proceso ejecutado (ExecutedProcessType) que se llame así: o si "SampleStorage" es el nombre de una instancia de tipo de proceso ejecutado, no puede existir una instancia de tipo de proceso (ProcessType) que se llame así.

El modelo resultante

Tras realizar este proceso, el modelo independiente de la plataforma (PIM) que se obtuvo fue el representado en el siguiente diagrama de clases (lo mostramos dividido en partes para facilitar su legibilidad):




ProcessType
<<Unique>> +OID_proc_type
<<Unique>> +name
ProcessType_USED_ON_ExecutedProcessType
<<Unique>> +OID_proc_type_used_on_exec_proc_type
<<Unique>> +name
ExecutedProcessType_ON_ObjectType
<<Unique>> +OID_exec_proc_type_on_object_type
<<Unique>> +name
ExecutedProcessType
<<Unique>> +OID_exec_proc_type
<<Unique>> +name
StateObjectType_OF_ObjectType
<<Unique>> +OID_stat_object_type_of_object_type
<<Unique>> +name
StateObjectType_PRODUCED_BY_ExecutedProcessType
<<Unique>> +OID_stat_object_type_prod_by_exec_proc_type
<<Unique>> +name
ExecutedProcessType_IS_PART_OF_OccurrenceType
<<Unique>> +OID_exec_proc_type_on_ocurr_type
<<Unique>> +name
StateObject_IS_PART_OF_OccurrenceType
<<Unique>> +OID_stat_object_type_is_part_of_occurr_type
<<Unique>> +name
StateObjectType
<<Unique>> +OID_stat_object_type
<<Unique>> +name


ProcessType
<<Unique>> +OID_proc_type
<<Unique>> +name
ProtocolType_EXECUTABLE_IN_ProcessType
<<Unique>> +OID_prot_type_exec_in_proc_type
<<Unique>> +name
ProtocolType
<<Unique>> +OID_prot_type
<<Unique>> +name
ProcessType_USED_ON_ExecutedProcessType
<<Unique>> +OID_proc_type_used_on_exec_proc_type
<<Unique>> +name
ProtocolType_USED_IN_ExecutedProcessType
<<Unique>> +OID_prot_type_used_in_exec_proc_type
<<Unique>> +name
ExecutedProcessType
<<Unique>> +OID_exec_proc_type
<<Unique>> +name
PerformerType_OF_ExecutedProcessType
<<Unique>> +OID_perf_type_of_exec_proc_type
<<Unique>> +name
PerformerType
<<Unique>> +OID_perf_type
<<Unique>> +name

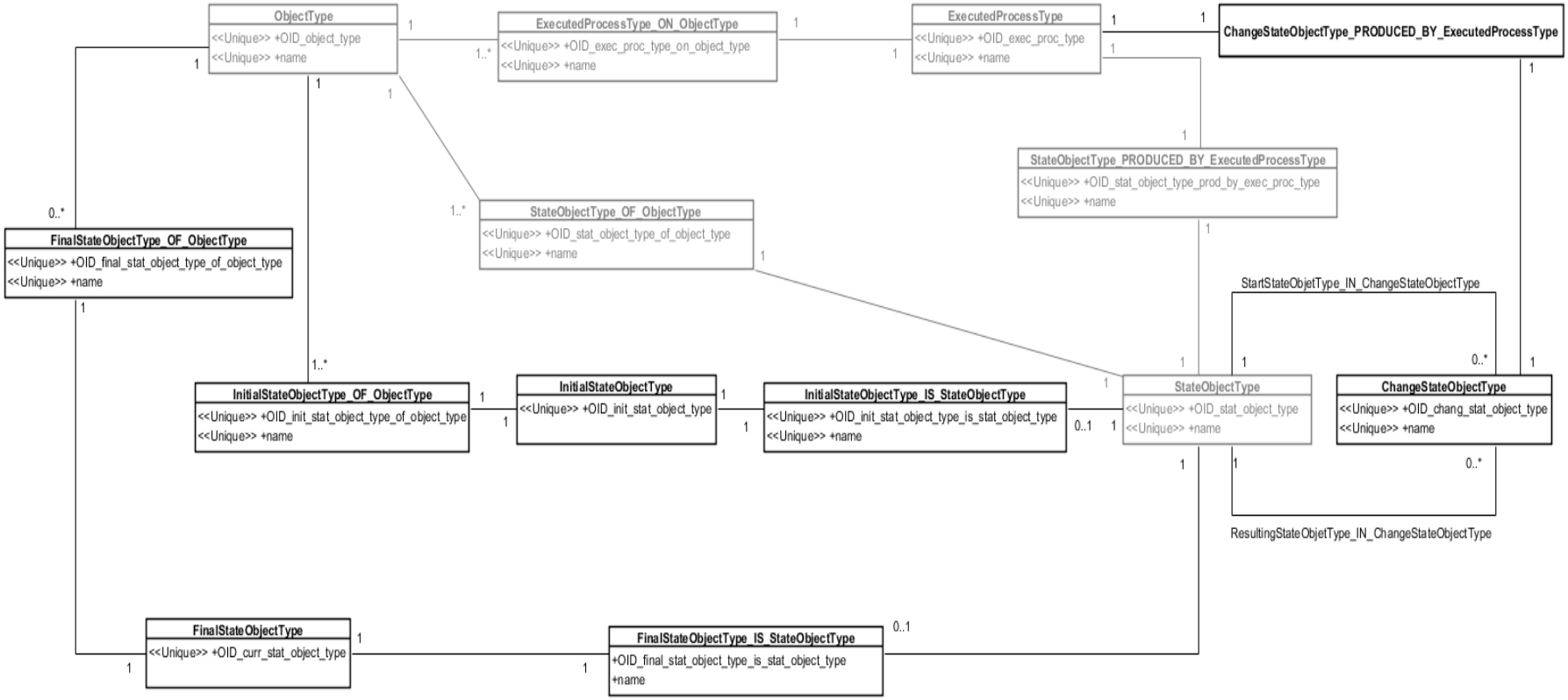


Figura 10.1 El modelo independiente de la plataforma (PIM) del Diccionario

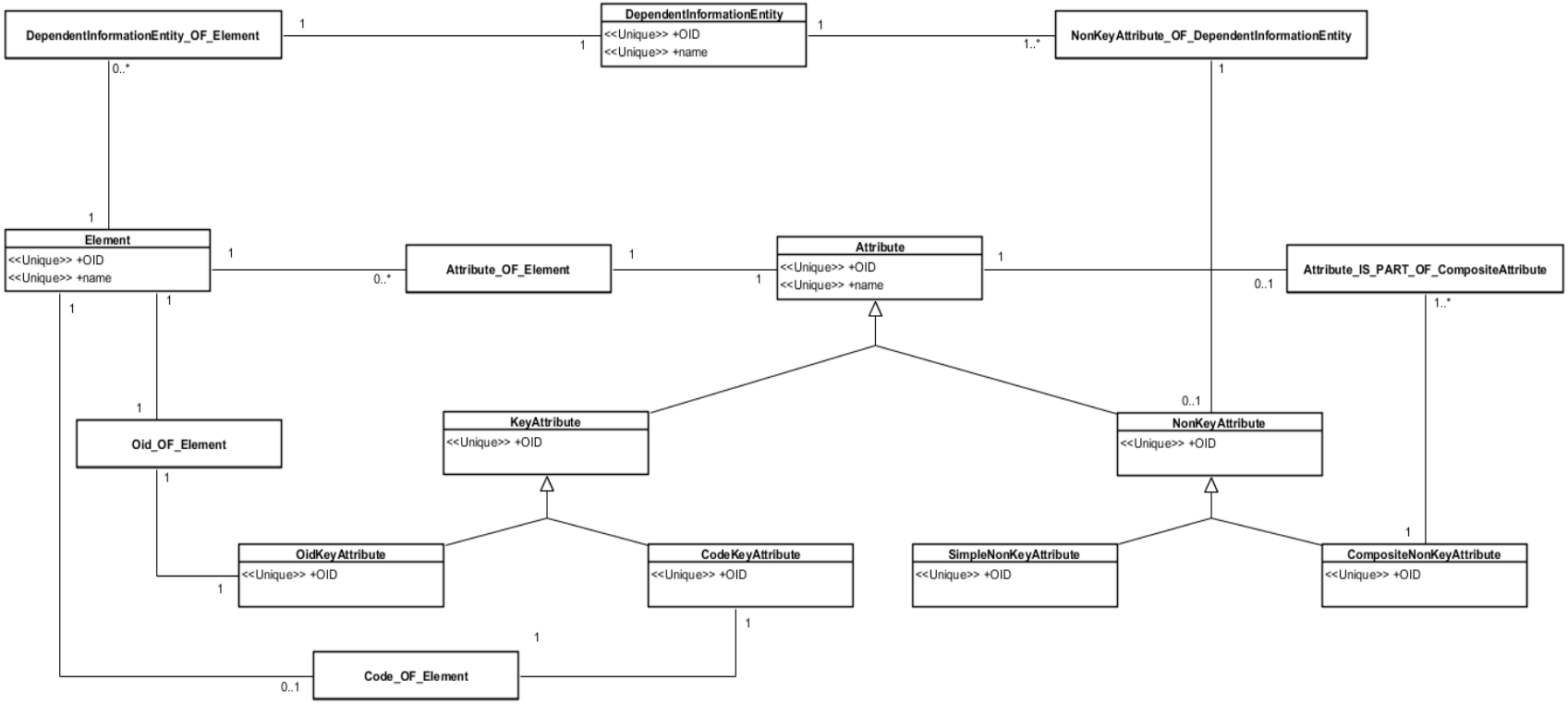


Figura 10.2 El modelo independiente de la plataforma (PIM) del Diccionario. Objetos de información

Objetos de información

Cuando hablemos de *Información de Acontecimientos* utilizaremos en ocasiones el concepto *Objeto de Información*; dado que el término *Objeto* se utiliza con asiduidad es importante diferenciar este concepto de otros conceptos que hacen referencia a objetos.

Para referir a los objetos que forman parte de los acontecimientos recurrimos al concepto *Objeto de Acontecimiento*; dicho de otro modo, el concepto *Objeto de Acontecimiento* se utiliza para referir a aquellos objetos afectados por acontecimientos; este concepto de objeto es al que hemos referido en numerosas ocasiones a lo largo de este texto.

Para referir a otros tipos de objeto, que pueden contemplarse en un modelo de base de acontecimientos (y por tanto en el *Diccionario*), y dan información sobre un *Objeto de Acontecimiento* o sobre los otros elementos de un acontecimiento como proceso ejecutado y estado, recurrimos al concepto *Objeto de Información*.

10.3 Definición del modelo específico de la plataforma

El modelo específico de la plataforma (PSM) depende, como su nombre indica, del sistema de persistencia (plataforma) sobre el que se vaya a implementar el *Diccionario*.

En este apartado vamos a mostrar un caso concreto de implementación sobre un sistema gestor de bases de datos relacionales, y por tanto vamos a mostrar el resultado de la transformación del modelo independiente de la plataforma (*PIM*) a un modelo específico de la plataforma (*PSM*) de tipo relacional.

El resultado de esta transformación es el diagrama entidad – relación del cual presentamos un extracto a continuación:

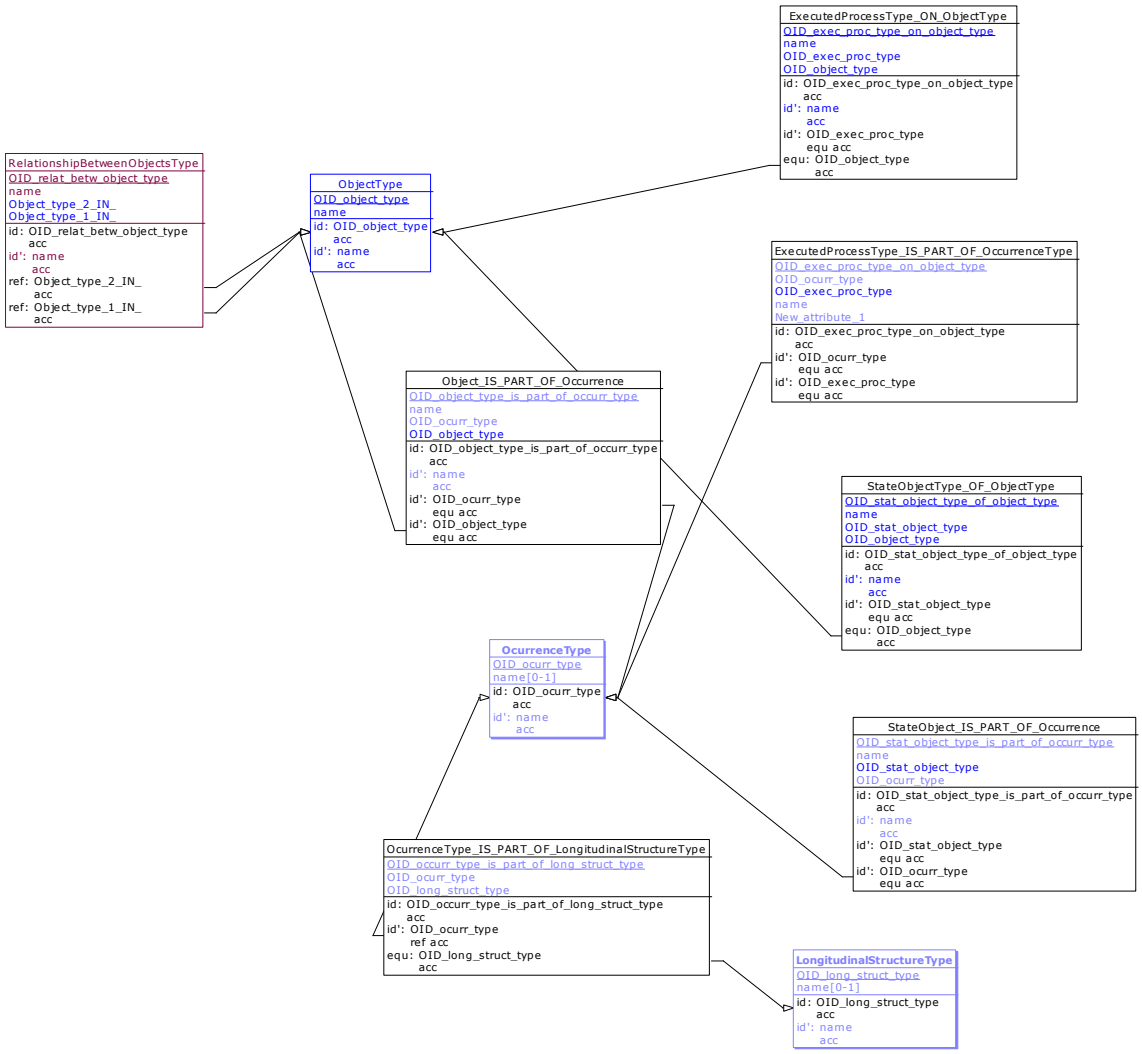


Figura 10.3 Diccionario. Extracto de un modelo específico (PSM) de la plataforma

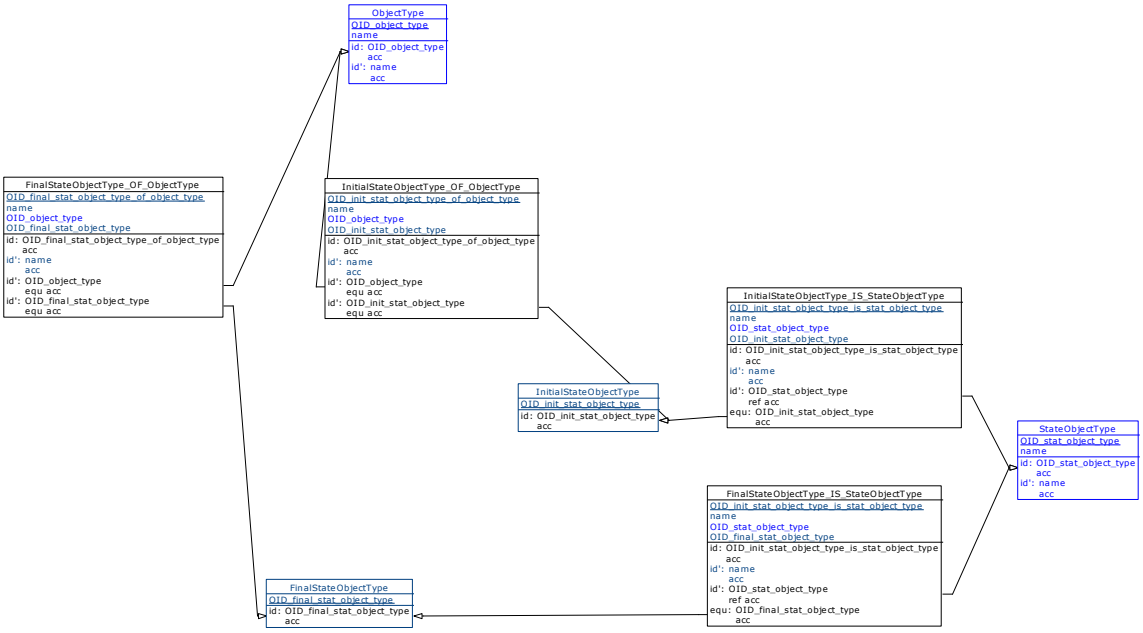


Figura 10.4 Diccionario. Extracto de un modelo específico (PSM) de la plataforma. Estados inicial y final

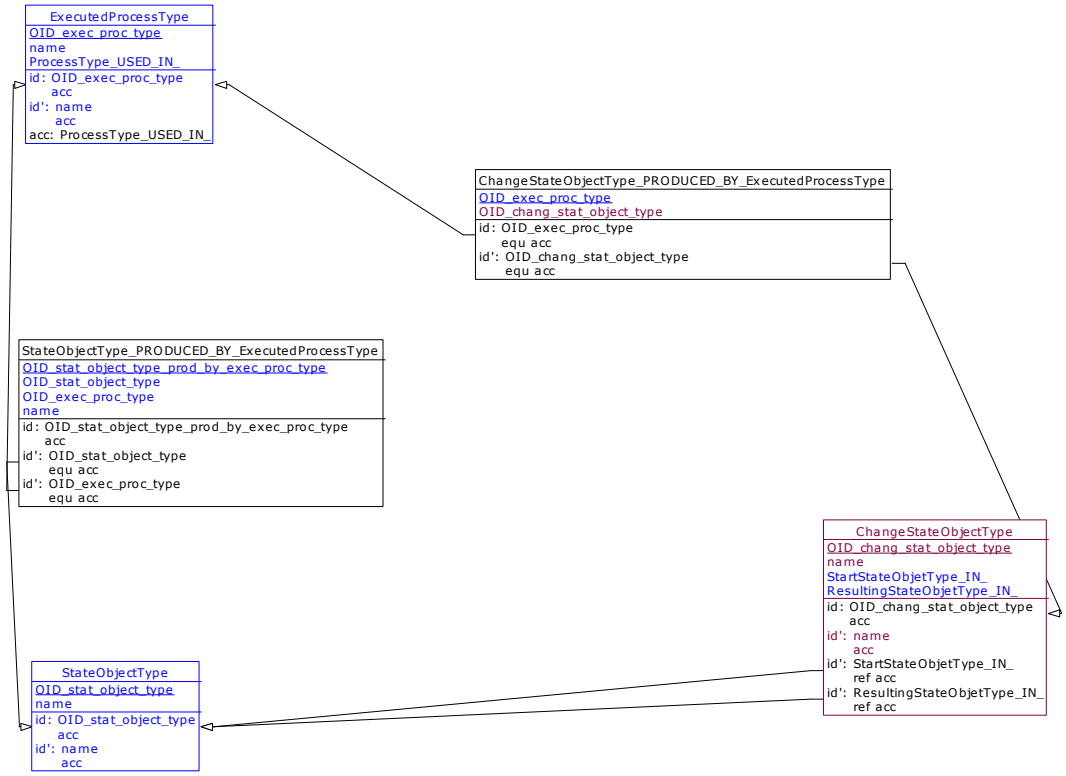


Figura 10.5 Diccionario. Extracto de un modelo específico (PSM) de la plataforma. Cambios de estado

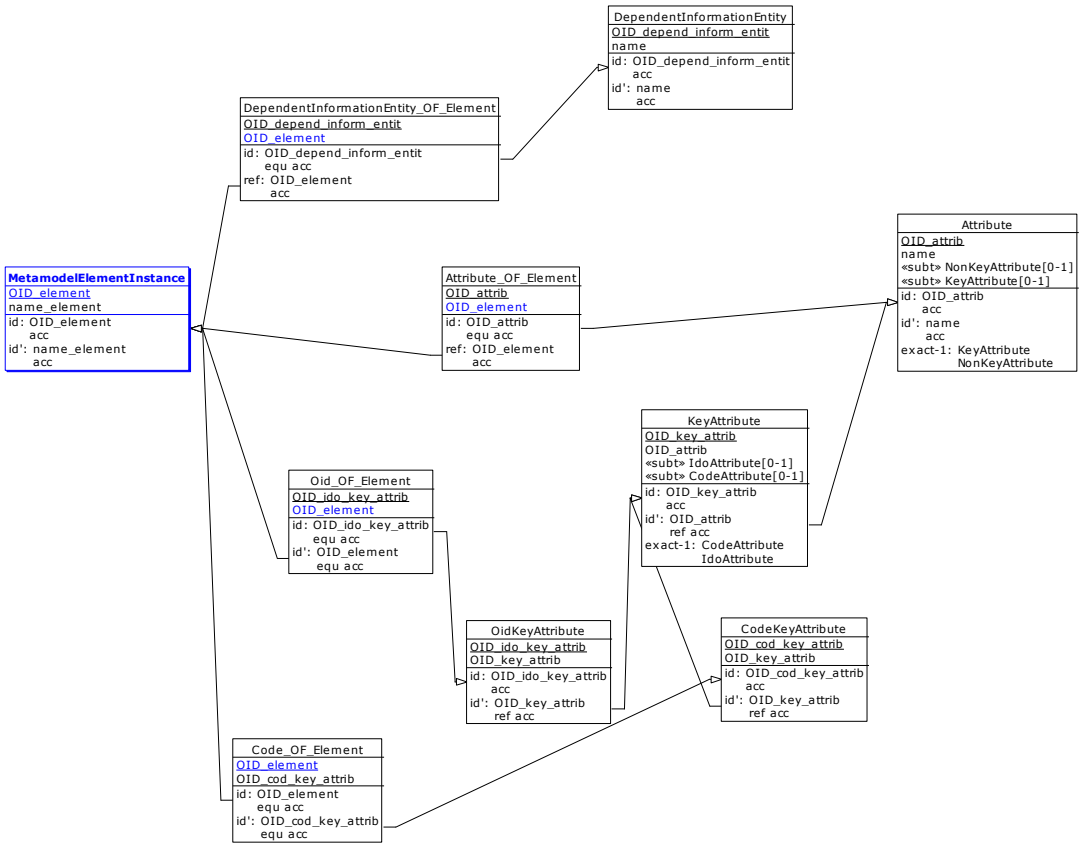


Figura 10.6 Diccionario. Extracto de un modelo específico (PSM) de la plataforma. Objetos de información

Capítulo 11. Lenguaje de Acceso a Información de Acontecimientos

11.1 Introducción

En el Capítulo 7 hemos presentado la sintaxis y forma de uso tanto del *Lenguaje no Generalizado de Acceso a Acontecimientos* como del *Lenguaje Generalizado de Acceso a Acontecimientos*, ambos publicados en [25], los cuales son parte del *Lenguaje de Acceso a Acontecimientos* (*OcAL*).

En este capítulo vamos a presentar el *Lenguaje de Acceso a Información de Acontecimientos* (*Occurrence Information Access Language*; *OcIAL*) como extensión del *Lenguaje de Acceso a Acontecimientos*.

El *Lenguaje de Acceso a Información de Acontecimientos* está formado por dos tipos de consultas, por una parte, aquellas que posibilitan obtener información almacenada en una base de acontecimientos y, por otra parte, aquellas que posibilitan obtener información almacenada en el diccionario o repositorio de metadatos.

Por ejemplo, una consulta para obtener el volumen de los objetos de tipo *Sample* registrados en la base de acontecimientos de un biobanco, o una consulta para obtener el número de huecos disponibles en un momento dado en cada uno de los objetos de tipo *Box* registrados en dicha base de acontecimientos correspondería con consultas del primer tipo. Por otra parte, una consulta para obtener la lista de tipos de objeto registrados en el diccionario, o una consulta para obtener la lista de tipos de procesos ejecutados que han podido actuar sobre un determinado tipo de objeto corresponderían con consultas del segundo tipo.

Al igual que el *Lenguaje de Acceso a Acontecimientos*, el uso de OcIAL se fundamenta en los siguientes pilares:

1 En el uso de una expresividad semántica cercana, en lo posible, a los términos de acontecimientos.

2 Que dicha expresividad no señale a ninguna forma de persistencia; tan sólo a las relaciones semánticas entre los términos utilizados.

3 En la existencia de un *Diccionario* que almacena información descriptiva de la estructura semántica de la base de acontecimientos, al que también hemos referido como *Repositorio de Metadatos*.

4 En la existencia de un algoritmo de traducción que, consultando la información almacenada en el *Diccionario* posibilita la traducción de las consultas OcIAL en consultas compatibles con el sistema de persistencia en el que se encuentra implantada la base de acontecimientos correspondiente.

11.2 Sobre los ejemplos de este capítulo

Para complementar y facilitar la comprensión de las explicaciones sobre *Lenguaje de Acceso a Información de Acontecimientos* que realizaremos a lo largo de este capítulo, mostraremos diferentes consultas de ejemplo, así como los resultados que se obtendrían con cada una de ellas. Dichos ejemplos corresponden con consultas OcIAL que podrían ejecutarse sobre una base de acontecimientos que da servicio a un biobanco.

En este apartado vamos a realizar una breve especificación de la base de acontecimientos y el diccionario en los que se basan los ejemplos de este capítulo.

Diccionario

La siguiente tabla muestra los nombres (*name*) de algunos elementos del diccionario:

ObjectType	ExecutedProcessType	PerformerType
sample	sample_recording	technician
aliquot	aliquoting	
subject	subject_consent_recording	
analytic		

Tabla 11.1 Representación del diccionario de ejemplo

A continuación, mostramos los objetos de información y atributos definidos en el diccionario que hemos tomado como referencia en los ejemplos que mostraremos en este capítulo. Este tipo de información corresponde con la que se registra en la estructura que hemos representado en la Figura 10.1 (PIM) y en la Figura 10.3 (PSM).

Entidad del diccionario	Elemento o instancia	Entidad dependiente de información	Atributo
ObjectType	sample		OID
			code
	aliquot		OID
			code
		volume	quantity
			unit
	subject		OID
			code
		subject_data	name
			first_surname
	analytic		OID
			code
ExecutedProcess	sample_recording		OID

Type		sample_recording_data	execution_date
	aliquoting		OID
		aliquoting_data	execution_date
	subject_consent_recording		OID
		subject_consent_recording_data	execution_date
PerformerType	technician		OID
		performer_data	name
			first_surname

Tabla 11.2 Representación de objetos de información y atributos en el diccionario

11.3 Sobre la sintaxis del lenguaje

En este apartado vamos a presentar las normas de sintaxis que son específicas o exclusivas del *Lenguaje de Acceso a Información de Acontecimientos*.

11.3.1 Estructura de una consulta

Las consultas del *Lenguaje de Acceso a Información de Acontecimientos* tienen una estructura común que puede esquematizarse de la siguiente forma:

1 Se inician con una cláusula `SELECT`, para referir a la información que se desea recuperar, ya sea de la base de acontecimientos o del diccionario.

Por ejemplo, en el caso de las consultas sobre la base de acontecimientos, a través de esta cláusula es posible indicar que se desea obtener información propia de objetos, estados, procesos ejecutados, etc.

En el caso de las consultas sobre el diccionario es posible, por ejemplo, indicar que se desea obtener información sobre los tipos de objetos, tipos de estados, etc. contemplados en una determinada base de acontecimientos.

2 La cláusula `SELECT` va seguida por la cláusula `RELATED BY` cuando se solicita información sobre dos o más elementos.

Por ejemplo, esta cláusula se incluiría para solicitar información sobre tipos de objetos y los tipos de proceso que pueden ejecutarse que podrían afectarles.

3 La cláusula `SELECT` puede ir seguida de la cláusula `RELATED TO` en caso de pretender pedir información de algún elemento relacionado con otro a través de una asociación semántica. El lenguaje posibilita la declaración de varias cláusulas `RELATED TO` una detrás de otra de tal forma que la siguiente establece la relación con el elemento de la anterior.

Por ejemplo, esta cláusula se incluiría para solicitar información sobre tipos de proceso cuya ejecución podría afectar a algún tipo de objeto.

4 A continuación, puede declararse la cláusula `RESTRICTED BY` para especificar las restricciones que se quieren aplicar sobre los resultados a obtener.

Por ejemplo, a través de esta cláusula es posible especificar que se desea obtener las

muestras con un determinado volumen, los procesos ejecutados durante un determinado intervalo de fechas, etc.

En el caso de las consultas sobre el diccionario es posible indicar, por ejemplo, que únicamente se desea obtener la información sobre un determinado tipo de objeto.

5 En caso de tratarse de una consulta de grupo podrá incluirse la cláusula `FOR EACH` para posibilitar agrupar los resultados según el tipo de elemento que se especifique.

Incluyendo esta cláusula es posible construir, por ejemplo, una consulta para obtener el número de analíticas realizadas a cada sujeto de estudio o, en el caso de las consultas sobre el diccionario, el número de tipos de estado por los que puede pasar cada tipo de objeto.

6 La cláusula `FOR EACH` puede ir seguida de la cláusula `SUCH THAT` la cual posibilita definir restricciones sobre el elemento por el que se desea agrupar.

Por ejemplo, en una consulta en la que se solicitara un recuento de los tipos de procesos cuyas ejecuciones podrían afectar a cada tipo de objeto, esta cláusula se podría indicar que sólo se desean obtener aquellos tipos de objeto que no pueden verse afectados por más de cinco tipos de proceso ejecutados.

En notación BNF (*Backus–Naur Form*; *Backus Normal Form*), esta sintaxis puede representarse de la siguiente forma:

```
<OcIAL_query> ::=
SELECT <information_to_obtain>
[{(RELATED BY | RELATED TO) <related_ elements>}]
[RESTRICTED BY <restricting_conditions>]
[FOR EACH <element_in_select_clause>
[SUCH THAT <restricting_conditions>]];
```

11.3.2 Operadores de comparación

El *Lenguaje de Acceso a Información de Acontecimientos* contempla el uso de los siguientes operadores de comparación como parte de los predicados declarados en las cláusulas `RESTRICTED BY` y `SUCH THAT`:

Operador	Descripción	Ejemplo
=	Compara la igualdad de dos expresiones.	volume.quantity = 40
<>	Compara dos expresiones para determinar si la expresión de la izquierda no es igual que la expresión de la derecha.	code <> '001-001'
>	Compara dos expresiones para determinar si la expresión de la izquierda tiene un valor mayor que el de la expresión de la derecha.	execution_date > '2016-08-16'
<	Compara dos expresiones para determinar si la expresión de la izquierda tiene un valor menor que el de la expresión de la	execution_date < '2016-08-16'

	derecha.	
>=	Compara dos expresiones para determinar si la expresión de la izquierda tiene un valor mayor o igual que el de la expresión de la derecha.	execution_date >= '2016-08-16'
<=	Compara dos expresiones para determinar si la expresión de la izquierda tiene un valor menor o igual que el de la derecha.	volume.quantity <= 40

Tabla 11.3 Operadores de comparación de las consultas OcIAL

La palabra reservada BETWEEN … AND

Aunque no es propiamente un operador de comparación, incluimos en este apartado la referencia a la palabra reservada `BETWEEN … AND` dado que su uso es similar al de un operador de comparación; en su caso, posibilita obtener información entre un rango de fechas (inclusive).

Por ejemplo, en este caso, estaríamos estableciendo una condición por la cual el atributo *initialDate* debe corresponder con una fecha del año 2016:

```
execution_date BETWEEN '2016-01-01 00:00:00' AND '2016-12-31 23:59:59'
```

11.3.3 Operadores lógicos

El *Lenguaje de Acceso a Información de Acontecimientos* contempla dos operadores lógicos. Los operadores `AND` y `OR` se utilizan como parte de las cláusulas `RELATED TO`, `RELATED BY` y `RESTRICTED BY` para poder establecer múltiples restricciones.

Operador	Descripción	Ejemplo
AND	Dadas dos condiciones separadas por el operador AND, es necesario que se cumplan ambas.	RESTRICTED BY volume.quantity > 40 AND volume.unit = 'ml'
OR	Dadas dos condiciones separadas por el operador OR, es suficiente con que se cumpla una de las dos.	RESTRICTED BY name = 'Miguel' OR name = 'Lope'
NOT	Niega la cláusula o el predicado al que precede.	NOT RELATED TO NOT TO InitialStateObjectType

Tabla 11.4 Operadores lógicos de las consultas OcIAL

11.3.4 Funciones de agregación

Las funciones de agregación son propias de las consultas de grupo. El *Lenguaje de Acceso a Información de Acontecimientos* contempla las siguientes funciones de agregación:

Función	Descripción	Ejemplo
COUNT(elemento)	Obtiene el n.º de ocurrencias del elemento al que hace referencia.	COUNT(OID)
MIN(elemento)	Obtiene el valor mínimo del elemento indicado.	MIN(volume.quantity)
MAX(elemento)	Obtiene el valor máximo del elemento indicado.	MAX(volume.quantity)
AVG(elemento)	Obtiene el valor medio del elemento indicado.	AVG(age)
SUM(elemento)	Obtiene la suma de los diferentes valores del elemento indicado.	SUM(free_positions)

Tabla 11.5 Funciones de agregación contempladas en OcIAL

11.3.5 Instancias y tipos

Como veremos más adelante, las diferentes cláusulas de una consulta del *Lenguaje de Acceso a Información de Acontecimientos* pueden referir tanto a instancias de elementos como a clases o tipos de elementos. Hablamos de *Clase* o *Tipo* cuando referimos a un elemento especificado en un modelo de base de acontecimientos; hablamos de *Instancia* de un elemento para referir a información concreta registrada en una base de acontecimientos.

Por ejemplo, podríamos decir *Sample* para referir a una clase declarada en el modelo de la base de acontecimientos de un biobanco, y *bloodSample007* para referir al identificador de una instancia de dicha clase.

Cuando en una consulta OcIAL se refiere a clases declaradas en un determinado modelo de base de acontecimientos, estas se declaran sin entrecomillar, por ejemplo: *Sample*. Por otra parte, las instancias de clases pueden ser referidas en una consulta OcIAL a través de cualquiera de los atributos propios de las mismas registrados en el *Diccionario*.

11.4 Consultas sobre el diccionario

Las *Consultas Sobre el Diccionario* (*Dictionary Queries*) posibilitan obtener información descriptiva de una base de acontecimientos almacenada en el *Repositorio de Metadatos* o *Diccionario*; para lograrlo, su sintaxis recoge referencias a conceptos propios del *Metamodelo de Bases de Acontecimientos*. Por lo tanto, al igual que ocurría con el *Lenguaje Generalizado de Acceso a Acontecimientos* descrito en el apartado 7.5, las consultas de esta variante del lenguaje abarcan los niveles M0, (*Base de Acontecimientos*), M1 (*Modelo de Base de Acontecimientos*) y M2 (metamodelo) de UML (ver Figura 7.4).

Sobre el Diccionario

Dado que será un concepto al que refiramos continuamente a lo largo de este apartado, antes de continuar, recordemos que el *Diccionario*, al que también hemos denominado *Repositorio de Metadatos*, es el responsable almacenar el conocimiento expresado en un modelo de base

de acontecimientos, así como el de las instancias del mismo en forma de bases de acontecimientos

Por otra parte, la estructura del *Diccionario*, así como las entidades que lo forman, está basada en el *Metamodelo de Bases de Acontecimientos*, por lo que necesariamente los elementos que aparecen en el primero deben encontrarse definidos en el segundo.

11.4.1 La cláusula SELECT

Mediante la cláusula `SELECT` se especifica un conjunto de elementos y, sobre cada uno de ellos, la información que se requiere.

Téngase en cuenta que sólo se consideran elementos aquello que en el metamodelo está especificado como una clase.

Caso de solicitud de datos de información de un elemento

La cláusula `SELECT` posibilita solicitar información sobre los tipos de elementos del diccionario escribiendo su nombre y a continuación lo que se solicita en forma de lista bajo la sintaxis siguiente:

```
SELECT <entidad_del_diccionario> GIVING (<lista_atributos_entidad>)
```

La respuesta a dicha consulta es la familia de las instancias de la entidad del diccionario referido y, para cada una de ellas, la lista de las instancias de los atributos que se solicitan.

Por ejemplo

```
SELECT ExecutedProcessType GIVING (OID_exec_proc_type, name)
```

devolvería, tal como se muestra en la siguiente tabla, el número de identificación y el nombre de los tipos de procesos que se admiten en la estructura como posiblemente ejecutados.

ExecutedProcessType	
OID_exec_proc_type	**name**
0000000001	sample_recording
0000000002	aliquoting
0000000003	subject_consent_recording

También se podría solicitar el nombre de todos los tipos de asociaciones existentes entre tipos de objetos a través de la consulta

```
SELECT RelationshipBetweenObjectsType GIVING (name)
```

Solicitud de toda la información

Como es usual, se puede solicitar toda la información existente sobre el elemento escribiendo * detrás de `GIVING`.

Renombre local de un elemento

Se puede renombrar un elemento a los únicos efectos de la consulta de forma usual utilizando la partícula `AS` entre el elemento y su renombre.

Caso de solicitud de datos de información de varios elementos

Se solicita información sobre cada uno de los elementos, separando la especificación por comas

SELECT
    <entidad1_del_diccionario> GIVING (<lista_atributos_entidad1>),
    <entidad2_del_diccionario> GIVING (<lista_atributos_entidad2>),
    ...

Por ejemplo

SELECT ExecutedProcessType GIVING (name), ObjectType GIVING (name)

devolvería una familia de pares de información sin ninguna relación entre ellos, cada elemento del par proporcionaría el nombre del elemento correspondiente.

ExecutedProcessType	**ObjectType**
name	**name**
sample_recording	sample
sample_recording	aliquot
sample_recording	subject
aliquoting	sample
aliquoting	aliquot
aliquoting	subject
subject_consent_recording	sample
subject_consent_recording	aliquot
subject_consent_recording	subject

Este tipo de consultas no es útil en nuestro contexto pues la información que se proporciona no está relacionada semánticamente. En todo caso, la mostramos para señalar el hecho de la falta de semántica.

De alguna forma estamos señalando que la operación matemática del producto cartesiano no tiene interés semántico en sí misma, su interés es tener un contexto para señalar algo en él que tenga sentido semántico.

11.4.2 La semántica formal del select

Una sentencia constituida sólo por una cláusula `SELECT`

SELECT e_1 GIVING (a_{11}, a_{12}...a_{1n1}), e_2 GIVING (a_{21}, a_{22}...a_{2n2})...e_p GIVING (a_{P1}, a_{P2}...a_{2np})

puede interpretarse que representa a un conjunto de informaciones del mismo tipo (tipadas

por su expresión) que constituyen la respuesta a lo solicitado y que cada una de ellas se pueden expresar de la siguiente forma

$((v(a_{11}), v(a_{12})... v(a_{1n1}))$, $(v(a_{21}), v(a_{22})... v(a_{2n2}))$,... $(v(a_{p1}), v(a_{p2})... v(a_{pnp}))$

donde

$\{i_{e1}, i_{e2},...i_{ep}\}$ es un conjunto de instancias $\{i_{e1}, i_{e2},...i_{ep}\}$ de elementos del metamodelo $\{e_1, e_2,...e_p\}$

y

$(v(a_{i1}), v(a_{i2})... v(a_{ini}))$ es una lista de valores de los atributos $(a_{i1}, a_{i2}... a_{ini})$ de una instancia i_{ei} elemento del metamodelo e_i

por tanto, es un elemento del producto cartesiano, $D(a_{i1}) \times D(a_{i2}) \times ...D(a_{eini})$, de los dominios de los correspondientes atributos.

De este modo, la respuesta a la consulta es un subconjunto del siguiente producto cartesiano

$$R= (\{e_1\} \times ((\{a_{11}\} \times D(a_{11})) \times ... \times (\{a_{1n1}\} \times D(a_{1n1})))) \times ... \times (\{e_p\} \times ((\{a_{p1}\} \times D(a_{p1})) \times ... \times (\{a_{pnp}\} \times D(a_{pnp}))))$$

Se hace notar que cada conjunto puntual representa la nominación de los elementos del conjunto con el que directamente se multiplica; así

e_i nomina a cada elemento de $((\{a_{i1}\} \times D(a_{i1})) \times ... \times (\{a_{ini}\} \times D(a_{ini})))$

a_{ij} nomina a cada elemento $D(a_{ij})$

Obsérvese que, si se olvidan las nominaciones, existe una proyección canónica a los subconjuntos del producto cartesiano de los dominios de los atributos que aparecen en la consulta

$D(a_{11}) \times ... \times D(a_{1n1}) \times ... \times D(a_{p1}) \times ... \times D(a_{pnp})$

De ello se concluye que, si nos olvidamos de la expresividad semántica, el soporte matemático de la consulta se proyecta de forma unívoca sobre el soporte matemático de la consulta relacional

SELECT $a_{11}, a_{12}...a_{1n1}, a_{21}, a_{22}...a_{2n2}...e_{p,...}a_{P1}, a_{P2}...a_{2np}$
FROM $e_1, e_2,...e_p$

Comparación con el SQL relacional

En el lenguaje SQL relacional es necesario especificar en la cláusula `FROM` las tablas en las que se encuentran los atributos que se solicitan.

En nuestro caso (*OcIAL*), esas tablas son elementos que se especifican en el `SELECT` señalando directamente los atributos que sobre ellos se solicitan. Esta decisión es natural debido a que en lo cotidiano la información se solicita sobre algo. Por otra parte, como valor añadido, se obtiene que la localización del atributo por parte del sistema se restringe a comprobar que un atributo lo es del elemento señalado.

Obsérvese que, como hemos señalado anteriormente, existe una proyección natural de lo que se solicita en el lenguaje (*OcIAL*) sobre lo que se solicitaría en el caso relacional, olvidando los nombres de los elementos y uniendo las listas en una sola.

11.4.3 La cláusula RELATED BY

En nuestro caso, siempre que se solicite información de dos o más elementos, de forma natural existirá una relación semántica entre los elementos que se especificará con una cláusula de tipo `RELATED`.

La cláusula `RELATED BY` especifica el tipo de relación entre los elementos sobre los que se pide información

La estructura de la cláusula `RELATED BY` puede describirse de la siguiente forma:

```
RELATED BY <nombre_de_la_asociación>
```

A la hora de indicar el nombre de la asociación como parte de la cláusula `RELATED BY` debe cumplirse lo siguiente:

- El nombre de la asociación debe tener una sintaxis válida:

  Si observamos el diseño del diccionario mostrado en la Figura 10.1, los nombres de las asociaciones siguen siempre un mismo patrón:

  `entidad_origen_DESCRIPTOR_entidad_destino`

  Por ejemplo, el nombre de la relación entre las entidades *PerformerType* y ExecutedProcessType es `PerformerType_OF_ExecutedProcessType`.

  Para lograr que el sistema pueda reconocer las entidades asociadas por una asociación, el nombre de las asociaciones referidas a través de la cláusula `RELATED BY` debe indicarse de la siguiente manera:

  `(entidad_origen)_DESCRIPTOR_(entidad_destino)`
- El nombre de la asociación debe corresponder con una asociación declarada en el diccionario.

Ejemplo

Si deseamos el nombre de los tipos de procesos que pueden ejecutarse junto con el nombre de los tipos de objeto a los que han afectado, basta especificar

```
SELECT ExecutedProcessType GIVING (name), ObjectType GIVING (name)
RELATED BY (ExecutedProcessType)_HAS_PRODUCED_(ObjectType)
```

que devolvería, en nuestro ejemplo, la información contenida en la siguiente tabla:

ExecutedProcessType	**ObjectType**
name	**name**
sample_recording	sample
aliquoting	aliquot
subject_consent_recording	subject

Caso de varias asociaciones

Cuando es necesario especificar varias relaciones, se puede realizar bajo dos formas distintas semánticamente equivalentes:

- Utilizando tantas cláusulas `RELATED BY` como asociaciones.

- O bien, utilizando una única cláusula `RELATED BY` especificando a continuación todas las asociaciones separadas por la conectiva `AND`.

Por ejemplo, la consulta en la que se solicita los tipos de asociaciones junto con el par de tipos de objetos asociados puede realizarse con una de las dos expresiones que siguen:

```
SELECT
    ObjectType AS ObjectType1 GIVING (name),
    RelationshipBetweenObjectsType GIVING (name),
    ObjectType AS ObjectType2 GIVING (name)
RELATED BY (ObjectType1)_IN_(RelationshipBetweenObjectsType)
RELATED BY (ObjectType2)_IN_(RelationshipBetweenObjectsType)
```

```
SELECT
    ObjectType AS ObjectType1 GIVING (name),
    RelationshipBetweenObjectsType GIVING (name),
    ObjectType AS ObjectType2 GIVING (name)
RELATED BY
    (ObjectType1)_IN_(RelationshipBetweenObjectsType) AND
    (ObjectType2)_IN_(RelationshipBetweenObjectsType)
```

11.4.4 La semántica formal del RELATED BY

Las instancias de una asociación del tipo

(<Elem1>)_<VINCULO>_(<Elem2>)

pueden representarse por los pares de valores de identificadores de objeto de las instancias de los elementos asociados.

Por tanto, si una asociación de ese tipo forma parte de una cláusula

RELATED BY (<Elem1>)_<VINCULO>_(<Elem2>)

podemos considerar el subconjunto del producto cartesiano de los dominios de los respectivos identificadores de objetos

D(<Elem1>) x D(<Elem1>)

Y en él, el subconjunto de parejas de identificadores que señalan que las instancias representadas están asociadas por la especificada.

Llamamos **dominio de la asociación al subconjunto anterior**.

Por otra parte, recordemos que un elemento genérico devuelto por una cláusula `SELECT` es del tipo

$(\mathbf{(}v(a_{11}), v(a_{12})... v(a_{1n1})\mathbf{)}, \mathbf{(}v(a_{21}), v(a_{22})... v(a_{2n2})\mathbf{)},... \mathbf{(}v(a_{p1}), v(a_{p2})... v(a_{pnp})\mathbf{)})$

$= (v_i, v_2,...v_p)$

y que una de las tuplas

$(v(a_{i1}), v(a_{i2})... v(a_{ini}))$

es el conjunto de valores, de los atributos solicitados, de una instancia i_{ei} del elemento e_i. Recordemos también que una instancia está identificada por un valor del identificador de objeto de e_i. Por tanto, una instancia respuesta de la cláusula `SELECT` tiene unívocamente

asociada una tupla

$(ido_i, ido_2 ... ido_p)$

de valores de identificadores de objeto de las correspondientes instancias.

Por tanto, la cláusula

RELATED BY (e_i)_<VINCULO>_(e_l)

define el siguiente predicado

(ido_i, ido_j) forma parte del dominio de la asociación (e_i)_<VINCULO>_(e_l)

predicado que establece formalmente la respuesta de sentencia `SELECT` en presencia de una cláusula `RELATED BY`.

De lo expuesto en los últimos apartados se observa la existencia de una semántica formal de las expresiones que estamos definiendo. Por tanto, para agilizar la descripción del lenguaje las obviaremos entendiendo que se pueden deducir sin especial dificultad extendiendo las técnicas que hemos utilizado en estos dos últimos apartados.

11.4.5 La cláusula RELATED TO

La cláusula `RELATED TO` permite pedir información de algún elemento que está relacionado con otro a través de una asociación semántica que se especifica a continuación de la partícula `TO`.

Ejemplo

Si deseamos el nombre de los tipos de elementos que posiblemente serán procesos ejecutados relacionados de forma natural con algún tipo de objeto, escribiríamos

```
SELECT ExecutedProcessType GIVING (name)
RELATED TO ObjectType BY (ExecutedProcessType)_HAS_PRODUCED_(ObjectType)
```

lo cual devolvería, en nuestro ejemplo, la información contenida en la siguiente tabla

ExecutedProcessType
name
sample_recording
aliquoting
subject_consent_recording

La diferencia que presenta la consulta anterior sin considerar la cláusula `RELATED TO` es que

```
SELECT ExecutedProcessType GIVING (name)
```

podría dar los nombres de los tipos de procesos posiblemente ejecutados que no están especificados a ningún tipo de objeto. Estos dos tipos de consultas son necesarias para los controles de calidad.

Como en el caso anterior se pueden concatenar varias cláusulas tal como se señala en el siguiente ejemplo en el que se solicita el nombre de los tipos de procesos que están relacionados de forma natural con un tipo de objeto y con un tipo de ejecutor,

simultáneamente:

```
SELECT ExecutedProcessType GIVING (name)
RELATED TO ObjectType BY (ExecutedProcessType)_HAS_PRODUCED_(ObjectType)
RELATED TO PerformerType BY (PerformerType)_OF_(ExecutedProcessType)
```

Como en el caso de la cláusula `RELATED BY`, se admite una expresión sintáctica equivalente utilizando sólo una expresión de tipo `RELATED TO` tal como se indica a continuación

```
SELECT ExecutedProcessType GIVING (name)
RELATED TO
    ObjectType BY (ExecutedProcessType)_HAS_PRODUCED_(ObjectType) AND TO
    PerformerType BY (PerformerType)_OF_(ExecutedProcessType)
```

11.4.6 Las cláusulas RELATED como proposiciones

En su forma más general se admite que las cláusulas `RELATED` se concatenen también con las conectivas `OR` (disyunción) y `NOT` (negación) constituyendo una expresión proposicional.

Por ejemplo, la consulta listar los nombres de los tipos de estados posibles de objetos, que están asociados a un tipo de objeto posible, que no pueden ser estado inicial ni estado final, se puede expresar como:

```
SELECT StateObjectType GIVING (name)
RELATED TO
    ObjectType BY (StateObjectType)_OF_(ObjectType) AND
    NOT
        (
        TO InitialStateObjectType BY (InitialStateObjectType)_IS_(StateObjectType) OR
        TO (FinalStateObjectType BY (FinalStateObjectType)_IS_(StateObjectType)
        )
```

o de forma equivalente

```
SELECT StateObjectType GIVING (name)
RELATED TO
    ObjectType BY StateObjectType_OF_ObjectType AND
    NOT TO InitialStateObjectType BY (InitialStateObjectType)_IS_(StateObjectType) AND
    NOT TO FinalStateObjectType BY (FinalStateObjectType)_IS_(StateObjectType)
```

La forma NOT RELATED TO

También se admite la forma `NOT RELATED TO` bajo la forma de uso que se muestra en el ejemplo

```
SELECT StateObjectType GIVING (name)
NOT RELATED TO
    ObjectType BY (StateObjectType)_OF_(ObjectType)
```

con el sentido de que se buscan los nombres de los tipos posibles de estados de objetos que no están relacionados con un tipo posible de objeto.

11.4.7 La cláusula RESTRICTED BY

Las cláusulas `RELATED TO` y `RELATED BY` establecen relaciones entre los tipos de elementos y, por lo tanto, en ese sentido, podrían considerarse como restricciones a la libertad de la percepción de las instancias de dichos tipos de elementos.

Sin embargo, en nuestra propuesta, reservamos el tipo explícito de restricción a aquellas que se establecen a través de valores de las características o atributos de los tipos de elementos mediante un predicado, restringiendo la respuesta a los elementos de información que hacen verdad el predicado.

La sintaxis de estos predicados es la usual. La cláusula se puede especificar detrás de una cláusula `SELECT` o después de una cláusula `RELATED`.

Ejemplos

e1) Nombre del tipo de proceso ejecutado identificado por '0000000002'.

```
SELECT ExecutedProcessType GIVING (name)
RESTRICTED BY OID_exec_proc_type = '0000000002'
```

La ejecución de esta consulta daría como resultado 'aliquoting' dado que es el tipo de proceso ejecutado identificado por el valor indicado.

e2) Nombre del tipo de proceso ejecutado identificado por '0000000002' que puede ser ejecutado por un tipo de ejecutor.

```
SELECT ExecutedProcessType GIVING (name)
RESTRICTED BY OID_exec_proc_type = '0000000002'
RELATED BY (PerformerType)_OF_(ExecutedProcessType)
```

La ejecución de esta consulta daría como resultado 'aliquoting' dado que es el tipo de proceso ejecutado identificado por el valor indicado y, además, en el diccionario, se ha establecido que es ejecutado por un tipo de ejecutor. En caso de que el tipo de proceso referido en la consulta no hubiera tenido asociado un tipo de ejecutor, la consulta no hubiera devuelto resultados.

Ambigüedad en los nombres de los atributos

La ambigüedad en el uso de un atributo en relación al elemento atribuido se resuelve como es usual anteponiendo el nombre del tipo al que atribuye, separando los términos mediante un punto, como se realiza en el siguiente ejemplo.

Ejemplo

e3) Nombres de los tipos de proceso ejecutados que pueden ser utilizados por los usuarios del sistema.

```
SELECT ExecutedProcessType GIVING (name)
RELATED BY (PerformerType)_OF_(ExecutedProcessType)
RESTRICTED BY PerformerType.name = 'technician'
```

La ejecución de esta consulta devolvería los siguientes resultados:

ExecutedProcessType
name
sample_recording

aliquoting
subject_consent_recording

Ejemplo

e4) Información de los tipos de objeto sobre los que pueden actuar ejecutores que corresponden con usuarios del sistema.

```
SELECT ObjectType GIVING *
RELATED BY
    (ExecutedProcessType)_HAS_PRODUCED_(ObjectType) AND
    (PerformerType)_OF_(ExecutedProcessType)
RESTRICTED BY PerformerType.name = 'technician'
```

La ejecución de esta consulta devolvería los siguientes resultados:

ObjectType	
OID_object_type	**name**
0000000001	sample
0000000002	aliquot
0000000003	subject

El tipo de objeto *analytic* no tiene referencia a tipos de procesos ejecutados y por tanto a un tipo de ejecutor, por lo que no se ha representado como parte de los resultados devueltos por la consulta de este ejemplo.

11.4.8 Consultas anidadas

Como parte de las consultas sobre el diccionario es posible definir consultas anidadas utilizando el vínculo predicativo `IN` como parte de la cláusula `RESTRICTED BY` de forma semejante al caso SQL.

Por ejemplo, para obtener los nombres de los tipos posibles de proceso ejecutados que han sido utilizados como nombres de tipos posibles de estado de objetos se puede utilizar la siguiente consulta:

```
SELECT ExecutedProcessType GIVING (name)
RESTRICTED BY name IN
    (
    SELECT StateObjectType GIVING (name)
    )
```

11.4.9 Funciones de agregación

El lenguaje admite expresiones para extraer información de grupos de elementos construidos por igualdad de uno o más valores de los atributos de los elementos.

Como es usual la información se puede extraer considerando la totalidad de lo referido como grupo, utilizando funciones de agregación que realizan cálculos o bien sobre grupos

explícitamente definidos, de forma que para cada grupo se solicitan funciones de agregación.

El *Lenguaje de Acceso a Información de Acontecimientos* contempla las funciones de agregación indicadas en la siguiente tabla.

Función
COUNT()
MIN()
MAX()
AVG()
SUM()

Tabla 11.6 Funciones de agregación

Ejemplos

e1) Obtener el número de tipos posibles de proceso ejecutados y el número posible de tipos de objetos.

```
SELECT
    ExecutedProcessType GIVING (COUNT(OID_exec_proc_type)),
    ObjectType GIVING (COUNT(OID_object_type))
```

OBSERVACIÓN

La semántica procedural de esta consulta es la que se desprende de forma natural del significado que se aprecia a través del lenguaje natural: Se calculan por separado las instancias del tipo de procesos ejecutados y las del tipo de objetos, contando en cada conjunto el número de identificadores OID.

La respuesta es la misma que daría la expresión, en el lenguaje relacional,

```
SELECT COUNT(DISTINCT OID_exec_proc_type), (COUNT(DISTINCT OID_object_type)
FROM ExecutedProcessType, ObjectType
```

Pero es importante señalar que la semántica procedural de este último es distinta, pues, primero se debe realizar el producto cartesiano y después, en cada columna, contar el número de identificadores distintos.

e2) Obtener el número de tipos posibles de procesos ejecutados y el número posible de tipos de objetos tales que los tipos posibles de procesos son aplicables a los tipos posibles de objetos.

```
SELECT
    ExecutedProcessType GIVING (COUNT(OID_exec_proc_type)),
    ObjectType GIVING (COUNT(OID_object_type))
RELATED BY (ExecutedProcessType)_ON_(ObjectType)
```

El significado procedural de esta consulta es que se toman las instancias de cada uno de los elementos señalados y se restringen a aquellos que están relacionados por la asociación establecida en la cláusula `RELATED`. A continuación, se cuentan los identificadores.

En este caso el resultado de dicha consulta es el mismo que la siguiente sentencia escrita en lenguaje relacional

```
SELECT COUNT(DISTINCT OID_exec_proc_type), (COUNT(DISTINCT OID_object_type)
```

```
FROM ExecutedProcessType, ObjectType, ExecutedProcessType_ON_ObjectType
WHERE
    ExecutedProcessType.IDO_exec_proc_type =
        (ExecutedProcessType)_ON_(ObjectType).IDO_exec_proc_type AND
    ObjectType.IDO_object_type =
        (ExecutedProcessType)_ON_(ObjectType).IDO_object_type
```

cuyo significado procedural es distinto al caso anterior pues se realiza el producto cartesiano, sobre él se restringe y posteriormente se cuentan los identificadores distintos de cada componente.

11.4.10 Cláusula FOR EACH

La cláusula `FOR EACH` refiere a los elementos para los que cada instancia define un grupo sobre el que se solicita, de forma genérica, una información.

Se puede referir a un elemento del diccionario como en el siguiente ejemplo:

e1) Mostrar el número de tipos de procesos ejecutados que pueden actuar sobre cada tipo de objeto:

```
SELECT
    ObjectType GIVING (name),
    ExecutedProcessType GIVING (COUNT(OID_exec_proc_type))
RELATED BY (ExecutedProcessType)_ON_(ObjectType)
FOR EACH ObjectType
```

La semántica procedural es la siguiente: Para cada instancia de tipo posible de objeto, se consideran los tipos posibles de procesos ejecutados sobre dicho tipo de objeto, y se cuenta su número a través del identificador de objeto

La ejecución de esta consulta daría los siguientes resultados:

ObjectType	**ExecutedProcessType**
name	**COUNT(OID_exec_proc_type)**
sample	1
aliquot	1
subject	1

Si nos restringimos al conjunto de datos de muestra presentados en la Tabla 11.2, en este caso, la función de agregación `COUNT()` nos devuelve la referencia a los tipos de procesos que pueden ejecutarse que afectan a cada tipo de objeto, concretamente

sample_recording, *aliquoting* y *subject_consent_recording.*

El tipo de objeto *analytic* no tiene referencia a tipos de procesos ejecutados, por lo que no se ha representado como parte de los resultados devueltos por la consulta de este ejemplo.

Agrupaciones sobre dos o más elementos

Se admiten con la sintaxis natural que señala el siguiente ejemplo:

```
SELECT
    ObjectType GIVING (name), ExecutedProcessType GIVING (name),
    COUNT(ExecutedProcessType_ON_ObjectType)
RELATED BY (ExecutedProcessType)_ON_(ObjectType)
FOR EACH ObjectType, ExecutedProcessType
```

Proporciona para cada par de tipo posible de objeto y tipo posible de proceso ejecutado sobre el anterior, el número de asociaciones por ejecución establecidas entre ellos

11.4.11 Cláusula SUCH THAT

La cláusula `SUCH THAT` establece las restricciones de los grupos definidos. Su contenido es un predicado que sea valorable sobre cada uno de los grupos especificados.

Ejemplo

```
SELECT
    ObjectType GIVING (name),
    ExecutedProcessType GIVING (COUNT(OID_exec_proc_type))
RELATED BY (ExecutedProcessType)_ON_(ObjectType)
FOR EACH ObjectType
SUCH THAT COUNT(OID_exec_proc_type) >= 5
```

```
SELECT
    ObjectType GIVING (name),
    ExecutedProcessType GIVING (COUNT(OID_exec_proc_type))
RELATED BY (ExecutedProcessType)_ON_(ObjectType)
FOR EACH ObjectType
SUCH THAT
    COUNT(OID_exec_proc_type) >= 5 AND
    ObjectType.name <> 'sample'
```

11.4.12 Información sobre tipos de acontecimientos

Los tipos de acontecimiento (OccurrenceType) están definidos en nuestro metamodelo como agregación de los tipos de objeto (ObjectType), los tipos de procesos ejecutados (ExecutedProcessType) y los tipos de estados de objetos (StateObjectType).

En primer lugar, el uso en las consultas se puede realizar sin más utilizando la sintaxis que se ha definido hasta el momento.

Ejemplos

e1) Listar los identificadores de los tipos de acontecimiento posibles que están asociados a un tipo de objeto posible.

```
SELECT OccurrenceType GIVING (OID_ocurr_type)
RELATED TO ObjectType BY (Object)_IS_PART_OF_(Occurrence)
```

e2) Listar los identificadores de los tipos de acontecimiento posibles que no están asociados a un tipo de objeto posible.

```
SELECT OccurrenceType GIVING (OID_ocurr_type)
NOT RELATED TO ObjectType BY (Object)_IS_PART_OF_(Occurrence)
```

Pero también se admiten formas que permiten explotar el conocimiento de la estructura de agregación, como mostraremos a continuación

El uso de la estructura de agregación

Una instancia de *OcurrenceType* es la agregación de una instancia de *ObjectType*, de *StateObjectType* y de *ExecutedProcessType*. Por tanto, podemos considerar que *OcurrenceType* establece una relación ternaria entre sus partes.

En ese sentido anterior podemos establecer proposiciones que señalen que una o más partes del conjunto

P = {ObjectType, StateObjectType, ExecutedProcessType}

están relacionadas por OcurrenceType, para lo cual se utilizará una cláusula `RELATED BY`.

Por ejemplo:

e1) Listar los identificadores de los tipos posibles de procesos ejecutados y los nombres de los tipos posibles de objetos que posiblemente formen parte de algún tipo de acontecimiento.

```
SELECT ExecutedProcessType GIVING (OID_exec_proc_type), ObjectType GIVING (name)
RELATED BY OccurrenceType
```

Obsérvese la diferencia de dicha consulta con la del siguiente ejemplo.

e2) Listar los nombres de los tipos posibles de procesos y los nombres de las ejecuciones posibles de los mismos de forma que los primeros formen parte de un posible tipo de acontecimiento y, análogamente, los segundos formen parte de un posible tipo de acontecimiento (no necesariamente el mismo para ambos).

```
SELECT ProcessType GIVING (name), ExecutedProcessType GIVING (name)
RELATED BY
   (ProcessType)_USED_ON_(ExecutedProcessType) AND
RELATED ProcessType BY OcurrenceType AND
RELATED ExecutedProcessType BY OcurrenceType
```

11.4.13 Consultas sobre tipos de información de contexto

De forma natural la estructura de lenguaje definida en los apartados anteriores se aplica a las partes del metamodelo que especifican el tipo de información de contexto.

Para realizarlo sólo es necesario aplicar algunas propiedades usuales de las relaciones de tipo ISA que concretamos a continuación.

1 Unicidad de camino ISA

   Entre dos elementos del metamodelo, sólo puede existir como máximo un camino de relaciones ISA.

2 Propiedad asociativa

   El sistema reconoce el único camino que existe entre dos elementos relacionados por un camino de relaciones ISA. De ese modo, si se especifica una relación del tipo:

   (<subelemento>)_IS_A_(<elemento>)

   se señala que existe un camino de relaciones ISA entre los elementos especificados.

3 Herencia

Todo elemento e, que está relacionado con un elemento e' a través de un camino de relaciones ISA, hereda todas las propiedades establecidas para todos los elementos que forman parte del único camino de relaciones ISA que existe entre e y e'.

Presentamos a continuación un ejemplo de aplicación.

e1) Para cada tipo de objeto extraer el nombre de su clave primaria.

SELECT
ObjectType GIVING (name),
CodeKeyAttribute GIVING (name)
RELATED BY
(CodeKeyAttribute)_IS_A_(Attribute)
(Attribute)_OF_(ObjectType)
FOR EACH ObjectType

11.4.14 Sintaxis

La sintaxis en notación BNF de las consultas sobre el diccionario es la siguiente:

<DOccurrenceQL> ::= <D_occurrences_query>;

<D_occurrences_query> ::=
SELECT <dictionary_entities_information>
[{RELATED BY <association_name>} | {[NOT] RELATED TO <related_to_elements>}]
[RESTRICTED BY <restricting_conditions>]
[FOR EACH <dicitionary_entities>
[SUCH THAT <group_restricting_conditions>]];

<dictionary_entities_information> ::=
(<dictionary_entity_information> [, <dicitionary_entities_information>])
| "*";

<dicitionary_entity_information> ::=
<dicitionary_entity> GIVING "("<information_object_elements>")" [AS
<alias_name>;

<information_object_elements> ::=
<information_object_attributes> | <aggregation_functions>;

<information_object_attributes> ::=
<information_object_attribute> [, <information_object_attributes>];

<information_object_attribute> ::=
[<information_object>.]<attribute>;

<aggregation_functions> ::=
<aggregation_function> [, <aggregation_functions>];

<aggregation_function> ::=
COUNT"("<information_object_attribute>")"

```
   | MIN”(“<information_object_attribute>”)”
   | MAX”(“<information_object_attribute>”)”
   | SUM”(“<information_object_attribute>”)”
   | AVG”( <information_object_attribute>”)”;
<association_name> ::=
   ”(“<source_dictionary_entity>”)”_<association_descriptor>_”(“<target_dictiona
   ry_entity>”)”
<related_to_elements> ::=
   <dicitionary_entity> BY <association_name>
<restricting_conditions> ::=
   (<condition_by_information_object_attribute>
   | <information_object_attribute> IN “(“<D_occurrences_query>”)”)
   [(AND | OR) <restricting_conditions>];
<condition_by_information_object_attribute> ::=
   <information_object_attribute> <comparison_operator> <value>;
<dictionary_entities> ::=
   <dictionary_entity > [, <dicitionary_entities>];
<group_restricting_conditions> ::=
   (<condition_by_information_object_attribute> | <aggregation_function>)
   [(AND | OR) <group_restricting_conditions>];
```

Listado 11.1 Sintaxis de las consultas sobre el diccionario

11.5 Consultas sobre la base de acontecimientos

Las *Consultas Sobre la Base de Acontecimientos* (*Occurrence Base Queries*) posibilitan obtener información almacenada en una base de acontecimientos como por ejemplo información sobre objetos, procesos ejecutados, estados, etc. concretos.

La sintaxis de las *Consultas Sobre la Base de Acontecimientos* es del mismo tipo que el definido para el caso de las *Consultas Sobre el Diccionario* realizando el siguiente cambio

<elemento>Type

por

<elemento> OF TYPE

tal como señala explícitamente en la siguiente tabla.

Elemento primitivo	Expresión
ObjectType	OBJECT OF TYPE <name>
RelationshipBetweenObjectsType	RELATIONSHIP OBJECTS OF TYPE <name>

StateObjectType	STATE OBJECT OF TYPE <name>
InitialStateObjectType	INITIAL STATE OBJECT OF TYPE <name>
CurrentStateObjectType	CURRENT STATE OBJECT OF TYPE <name>
ExecutedProcessType	EXECUTED PROCESS OF TYPE <name>
PerformerType	PERFORMER OF TYPE <name>
ProcessType	PROCESS OF TYPE <name>
ProtocolType	PROTOCOL OF TYPE <name>

Tabla 11.7 Elementos sobre los que se puede preguntar a través de la cláusula SELECT

11.5.1 Sobre el tipo de información

Recordemos que, según se establece en el metamodelo, la información de contexto de un elemento del metamodelo puede estar especificada a través de atributos directamente asociados o bien a través de entidades dependientes de información, que tienen asociados atributos.

Por ejemplo, un objeto de tipo 'aliquot' puede tener una información de tipo 'volume' definida como entidad dependiente que, a su vez, tenga como atributos 'quantity' (cantidad) y 'unit' (unidad de medida).

Cuando se solicita dicho tipo de información se puede utilizar la sintaxis natural

```
GIVING (volume(quantity, unit))
```

o bien se puede utilizar la forma plana

```
GIVING (volume.quantity, volume.unit))
```

Sin embargo, cuando se utiliza el dato como parte de un predicado, es necesario utilizar siempre la forma plana.

Ejemplo

Lista de volúmenes de todas aquellas instancias de objetos de tipo 'aliquot' registradas en la base de acontecimientos que tienen un volumen igual a 40.

```
SELECT OBJECT OF TYPE aliquot GIVING (volume(quantity, unit))
RESTRICTED BY volume.quantity = 40
```

Un ejemplo más general

La siguiente consulta proporciona información de las alícuotas que han sido obtenidas por ejecución del proceso alicuotado entre las fechas especificadas y cuyo volumen es mayor estrictamente que 0.

```
SELECT
    OBJECT OF TYPE aliquot GIVING (code, volume.quantity, volume.unit),
    EXECUTED PROCESS OF TYPE aliquoting GIVING (OID, execution_date),
    PERFORMER OF TYPE technician GIVING (code, name, first_surname)
RELATED BY
    (aliquot)_PRODUCED_BY_(aliquoting) AND
```

(technician)_WAS_PERFORMER_OF_(aliquoting)
RESTRICTED BY
execution_date BETWEEN '2017-01-01' AND ‘'2017-12-31' AND
volume.quantity > 0

OBSERVACIONES

1 (aliquot)_PRODUCED_BY_(aliquoting)

es una instancia de

ExecutedProcessType_ON_ObjectType

donde

aliquot, aliquoting

son instancias respectivas de

ObjectType, ExecutedProcessType

2 (technician)_WAS_PERFORMER_OF_(aliquoting)

es una instancia de

PerformerType_OF_ExecutedProcessType

donde

technician, aliquoting

son instancias respectivas de

PerformerType, ExecutedProcessType

11.5.2 Predicado OCCURRENCE

Llamamos predicado de tipo `OCCURRENCE` a un predicado del tipo

(<instancia_elemento>)_IS_PART_OF_(OCCURRENCE)

que toma el valor verdad (true) si `<instancia_elemento>` forma parte de un acontecimiento; dicho de otro modo, si la expresión anterior es una instancia de una de las siguientes asociaciones

Object_IS_PART_OF_Occurrence

ExecutedProcessType_IS_PART_OF_OccurrenceType

StateObject_IS_PART_OF_Occurrence

para alguna instancia de *OccurrenceType*.

Extendemos de forma natural la definición anterior a un predicado que expresa si dos o tres instancias de elementos forman parte simultáneamente de una instancia de *OccurrenceType*.

Por ejemplo, podemos considerar los siguientes predicados

(aliquot)_IS_PART_OF_(OCCURRENCE)

(aliquot, aliquoting)_IS_PART_OF_(OCCURRENCE)

(aliquot, aliquoting, aliquoted)_IS_PART_OF_(OCCURRENCE)

para establecer si las instancias expresadas a la izquierda forman parte de un mismo acontecimiento.

Bajo esta sintaxis, podemos escribir la consulta del ejemplo anterior bajo la siguiente forma

equivalente:

```
SELECT
    OBJECT OF TYPE aliquot GIVING (code, volume.quantity, volume.unit),
    EXECUTED PROCESS OF TYPE aliquoting GIVING (OID, execution_date),
    PERFORMER OF TYPE technician GIVING (code, name, first_surname)
RELATED BY
    (aliquot, aliquoting)_IS_PART_OF_(OCCURENCE) AND
    (technician)_PERFORMER_OF_(aliquoting)
RESTRICTED BY
    execution_date BETWEEN '2017-01-01' AND '2017-12-31' AND
    volume.quantity > 0
```

V. CONCLUSIONES Y TRABAJO FUTURO

Capítulo 12. Conclusiones y trabajo futuro

En este último capítulo del presente documento de tesis presentamos las principales conclusiones que se derivan de los trabajos realizados en la tesis, y se señalan las posibles líneas de investigación que podrían seguirse en el futuro o se están siguiendo.

12.1 Conclusiones

12.1.1 Lo realizado

En esta tesis se ha presentado un lenguaje de consultas para el acceso y explotación de la información contenida en una base de acontecimientos, utilizando una expresividad cercana al lenguaje natural en términos propios de los elementos que forman parte de un acontecimiento.

Para la definición de dicho lenguaje se han utilizado procesos de abstracción del lenguaje SQL relacional de modo que **el lenguaje** pueda ser utilizado en **un nivel puramente conceptual**.

Para **garantizar la utilidad** del lenguaje se ha realizado una prueba de concepto que garantice la traducción de sus expresiones como expresiones del lenguaje relacional; esto se ha realizado, y publicado en [25], para la versión que accede a conceptos (no se considera las informaciones que pudieran existir sobre ellos) y ha sido presentado en el Capítulo 8.

Esta prueba, junto con las formalizaciones de la semántica de las consultas definidas en la segunda parte de la tesis (basadas en el álgebra relacional), garantizan la posibilidad de la implementación del lenguaje extendido sobre un gestor de bases de datos relacionales.

Todas las pruebas han sido realizadas sobre la base de acontecimientos del sistema *Arasis*, que corresponde con una base de datos relacional que implementa un *modelo de base de acontecimientos* basado en procesos para la gestión de información biomédica sobre una muestra de sujetos de la que existe información longitudinal (información extraída a través del tiempo).

12.1.2 El alcance sintáctico logrado

Se ha logrado definir un lenguaje conceptual de acceso a una base de acontecimientos construida sobre un diagrama de objetos, que considera las asociaciones entre ellos como única estructura de abstracción, de forma que cada objeto tiene asociado su correspondiente diagrama de estados posibles, así como los posibles procesos que pueden dar lugar a los

cambios de estados.

Se hace notar que las estructuras de acontecimientos se establecen como estructuras de agregación, así como las estructuras longitudinales.

Restricciones útiles del alcance

El metamodelo y su lenguaje, además del alcance expresado, puede ser utilizado sin más en las siguientes situaciones:

1 Gestión estática de objetos, sin tratamiento explícito de los cambios de estado ni de los acontecimientos.

2 Gestión dinámica de objetos, sin tratamiento explícito de los procesos que originan los cambios de estado.

3 Gestión dinámica de objetos dirigida por procesos, sin tratamiento explícito de los ejecutores de los procesos y sin tratamiento explícito de los acontecimientos.

4 Gestión dinámica de objetos dirigida por procesos, con tratamiento explícito de los ejecutores de los procesos, pero sin tratamiento explícito de los acontecimientos

5 Gestión dinámica de objetos dirigida por procesos con o sin tratamiento explícito de los protocolos utilizados.

12.1.3 El alcance semántico logrado: Object Provenance

Si el metamodelo y el lenguaje es utilizado en la extensión definida, para cada objeto, sin ningún tratamiento específico, se tiene la siguiente información:

1 Su origen.

2 Su historia propia a través de los acontecimientos que han sucedido sobre él.

3 Los lugares, las custodias, las propiedades y cualquier otra situación en la que se ha encontrado el objeto, si se han considerado los procesos correspondientes.

4 La relación con el contexto; es decir, su relación con otros objetos y sus diferentes situaciones, relaciones que pueden extraerse de la existencia de asociaciones entre los objetos, de procesos relacionados que han actuado sobre los mismos, ejecutores que han ejecutado procesos sobre los dos objetos...

Por tanto, se dispone de una estructura que proporciona implícitamente con gran extensión y profundidad los elementos del *Provenance* de cualquier objeto.

12.1.4 Evolución simple: Incorporación de otras abstracciones

De lo expuesto en la tesis es fácil evolucionar el modelo de lenguaje a la consideración de otras estructuras de abstracción en el diagrama de objetos, como es la inclusión en el diagrama de objetos de:

1 Las relaciones ISA como asociaciones entre objetos; su inclusión en el lenguaje sería semejante a la que se ha realizado con las relaciones ISA que se han considerado en el diagrama de información dependiente del contexto representado en la Figura 9.4.

2 Objetos agregados que se definen como agregación de un conjunto finito de otros objetos. La incorporación al lenguaje de estos elementos es semejante a la que se ha realizado para los acontecimientos.

12.2 Trabajo en desarrollo

12.2.1 Lenguajes de definición y modificación de acontecimientos

El trabajo que ya se ha iniciado es el de la definición de las partes del lenguaje dirigidas a la definición y modificación de acontecimientos. Concretamente se está desarrollando:

1 El *Lenguaje de Definición de Acontecimientos*, como lenguaje que expresa los cambios de estructura en la base de acontecimientos, en particular también incorpora expresividad sobre la gestión de las restricciones de integridad y, más generalmente, las reglas de negocio; por ejemplo, considerar un nuevo objeto en el diagrama de objetos, o modificar la estructura de los posibles procesos ejecutados, o cambiar la obligatoriedad de una información.

2 El *Lenguaje de Modificación de Acontecimientos*, como lenguaje que expresa los cambios de información sobre acontecimientos (en particular, la gestión de los propios acontecimientos, una vez establecidas las estructuras); por ejemplo, registrar la información de un nuevo acontecimiento cuya estructura ya está establecida en el sistema.

12.2.2 Nuevas estructuras longitudinales: Dato longitudinal

El *dato longitudinal* se refiere a la evolución de los valores de un metadato a través del tiempo. Esta información, incluida en la estructura de bases de acontecimientos, tiene un especial valor en la investigación biomédica. La base de acontecimientos, además de proporcionar explícitamente la estructura longitudinal del mismo (es decir, su evolución a través del tiempo) proporcionaría su *provenance*.

Esta investigación se encuentra en estado muy avanzado en cuanto a la estructura y el lenguaje para su incorporación expresiva. Su incorporación implicaría disponer de un *Data Provenance* relacionado con el *Object Provenance* del que actualmente se dispone. Es decir, se podría relacionar la historia de un dato con la historia de otros datos que se encontraran en el contexto del anterior; algo esencial y básico para la investigación biomédica.

12.3 Otros objetivos del grupo de investigación

El doctorando ha participado en alguna medida en otros aspectos evolutivos de las *Bases de Acontecimientos*, aspectos que se encuentran en diferentes estados de evolución.

12.3.1 Incorporación del Workflow

En la estructura de bases de acontecimientos definida, los procesos son elementos atómicos sin ninguna relación explícita entre ellos; tan sólo se puede apreciar alguna relación a través de los diagramas de estados de los objetos, al estar cada estado asociado a un proceso ejecutado.

El objetivo es incluir relaciones entre los procesos de modo que definan un *workflow*.

Pruebas parciales se han realizado con aplicación a la gestión de incidencias y problemas.

12.3.2 Gestión del cambio de dato

Para la gestión de la calidad y la mejora de procesos es esencial incorporar la gestión del cambio de un dato, cualquiera que sea la causa del cambio.

Ya se dispone de una metodología e implementación simple en bases de acontecimientos implementadas relacionalmente. Es necesario analizar y estudiar su incorporación conceptual en el modelo que se ha definido.

12.3.3 Técnicas de Business Intelligence

El dato longitudinal se refiere a la evolución de los valores de un metadato a través del tiempo. Esta información, incluida en la estructura de bases de acontecimientos, tiene un especial valor en la investigación biomédica. La base de acontecimientos, además de proporcionar explícitamente la estructura longitudinal del mismo (es decir, su evolución a través del tiempo) proporcionaría su *provenance*.

Anexo A Especificación de la sintaxis del lenguaje. La notación BNF

Para la especificación de la sintaxis del lenguaje nos hemos basado en un metalenguaje (es decir, un lenguaje que se utiliza para especificar otros lenguajes) denominado *Notación de Backus-Naur* o notación *BNF* (*Backus–Naur Form* o *Backus Normal Form* en su denominación inglesa) [4]. La notación BNF se utiliza comúnmente para describir la sintaxis de lenguajes de programación; por ejemplo, la sintaxis de SQL se encuentra descrita utilizando la notación BNF [37].

Una especificación en notación BNF está formada por un conjunto de reglas de derivación con la siguiente estructura:

```
<símbolo> ::= <expresión>
```

Partiendo de esta estructura podemos definir cada una de sus partes:

- `<símbolo>` es un elemento no terminal; es decir, puede ser descompuesto en varios elementos.
- `<expresión>` está formada por una o varias secuencias de símbolos o expresiones separadas por el carácter `|` que representa una opción a elegir. Cada una de las opciones es una posible sustitución de `<símbolo>`. Así pues, la siguiente regla:

  ```
  <símbolo> ::= <secuencia1> | <secuencia2>
  ```

  Indica que `<símbolo>` puede ser sustituido bien por `<secuencia1>`, bien por `<secuencia2>`.

  Los símbolos que forman parte de una expresión que nunca aparecen a la izquierda de una regla, se consideran elementos no terminales y los representamos sin encerrar entre `<…>`. Por ejemplo, en esta secuencia de reglas, `símbolo1` y `símbolo2` serían elementos terminales:

  ```
  <regla> ::= <regla1> | <regla2>
  <regla1> ::= símbolo1
  <regla2> ::= símbolo2
  ```

- `::=` indica que el símbolo de la izquierda debe ser sustituido por la expresión de la derecha.

Por otra parte, con el objetivo de lograr mayor expresividad en la sintaxis del lenguaje, hemos incorporado algunos elementos de la notación BNF extendida (*Extended Backus–Naur form*; *Extended BNF*), la cual ha sido adoptada como un estándar ISO (ISO/IEC 14977) [38].

De esta notación extendida recogemos la posibilidad de agrupar expresiones y símbolos encerrándolos entre paréntesis `(…)`, la forma de representación de expresiones y símbolos opcionales encerrándolas entre corchetes `[…]`, la forma de denotar repeticiones utilizando

llaves `{…}`, y el uso del carácter punto y coma como terminador.

Por ejemplo, en la siguiente regla, `símbolo3` seguido bien de `símbolo4` o `símbolo5` sería un elemento opcional:

```
<regla2> ::= símbolo2 [símbolo3 (símbolo4 | símbolo5)];
```

La notación BNF extendida también contempla el uso de comentarios encerrándolos entre `(* ... *)`. Por ejemplo: `(* Esto es un comentario *)`.

Por último, para no añadir complejidad a la sintaxis, hemos optado por no entrecomillar los términos y caracteres que son parte de la misma salvo casos de ambigüedad con elementos propios de la notación BNF indicados en este apartado. Así pues, los caracteres coma, punto, etc. los representaremos sin entrecomillar, mientras que caracteres como paréntesis, corchetes y el punto y coma los encerraremos entre comillas dobles.

Símbolo	**Descripción**
::=	Definición
|	Alternancia
(…)	Agrupación
[…]	Opcionalidad
{ … }	Repetición
(* … *)	Comentario
;	Terminación

Tabla 12.1 Tabla resumen de los símbolos utilizados

Bibliografía

1 Allué, A., Domínguez, E., López, A., Zapata, M. A. (2013) QRP: A CMMI Appraisal Tool for Project Quality Management. Procedia Technology, Volume 9, Pages 664-669, ISSN 2212-0173, http://dx.doi.org/10.1016/j.protcy.2013.12.073

2 Allué, A., López, A., Ciria, J. C., Domínguez, E., Francés, Á., Zapata, M. A. (2016) The task-oriented occurrence pattern. In Proceedings of the 21st European Conference on Pattern Languages of Programs (p. 6). ACM. http://dx.doi.org/10.1145/3011784.3011790

3 Alexander, C. (1977) A Pattern Language: Towns, Buildings, Construction. Oxford University Press, USA. p. 1216. ISBN 0-19-501919-9

4 Backus, J. W. (1959) The Syntax and Semantics of the Proposed International Algebraic Language of the Zurich ACM-GAMM Conference. Proceedings of the International Conference on Information Processing, UNESCO, pp.125-132. http://www.softwarepreservation.org/projects/ALGOL/paper/Backus-Syntax_and_Semantics_of_Proposed_IAL.pdf/view

5 Bézivin, J. (2006) Model driven engineering: an emerging technical space. In: Proceedings of GTTSE'05, Springer, pp 36–64

6 Bloesch, A. C., Halpin, T. A. (1996) ConQuer: a conceptual query language. In: Thalheim B (ed) ER. Lecture Notes in Computer Science, vol 1157. Springer, pp 121–13

7 Brauer, P. C., Hasselbring, W. (2012) Capturing provenance information with a workflow monitoring extension for the kieker framework. In: Proceedings of the 3rd international workshop on semantic web in provenance management, CEUR-WS

8 Brenzler, J., Clark, S. J. (2000) Longitudinal Database Design. The INDEPTH Annual General Meeting

9 Burleson Consulting. Database Objects and Inheritance. http://www.dba-oracle.com/t_object_database_inheritance.htm. Último acceso en febrero de 2017

10 Buneman, P., Davidson, S. B. (2010) Data provenance - The foundation of data quality. http://www.sei.cmu. edu/measurement/research/upload/Davidson.pdf. Last visited on March 2017

11 Campanile, F., Coppolino, L., Giordano, S., Romano, L. (2008) A business process monitor for a mobile phone recharging system. J Syst Archit 54(9):843–848

12 Carata, L., Akoush, S., Balakrishnan, N., Bytheway, T., Sohan, R., Selter, M., Hopper, A. (2014) A primer on provenance. Commun ACM 57(5):52–60

13 Casasnovas, J. A., Alcalde, V., Civeira, F., Guallar, E., Ibanez, B., Jimenez-Borreguero, J., Laclaustra, M., Leon, M., Ordovas, J. M., Pocovi, M., Sanz, G., Fuster, V. (2012) Aragon workers' health study - design and cohort description. BMC Cardiovascular Disorders

12(1), 45

14 Chebotko, A., Lu, S., Fei, X., Fotouhi, F. (2010) Rdfprov: a relational rdf store for querying and managing scientific workflow provenance. Data Knowl Eng 69(8):836–865

15 Chen, P., Plale, B., Aktas, M. S. (2014) Temporal representation for mining scientific data provenance. Future Gener Comput Syst 36:363–378

16 Chiticariu, L., Tan, W-C., Vijayvargiya, G. (2005) Dbnotes: a post-it system for relational databases based on provenance. In: Proceedings of the 2005 ACM SIGMOD international conference on management of data, ACM, pp 942–944

17 Clark, S. J. (2007) An introduction to the General Temporal Data Model and the Structured Population Event History Register. Scandinav Journal of Public Heath, 21-25

18 Clark, S. J. (2006) A temporal data model and the structured population event history register. Demographic Research 15, 181-252

19 Costello, C., Molloy, O. (2008) Towards a semantic framework for business activity monitoring and management. In: AAAI spring symposium: AI meets business rules and process management, pp 17–27

20 Curbera, F., Doganata, Y., Martens, A., Mukhi, N. K., Slominski, A. (2008) Business provenance–a technology to increase traceability of end-to-end operations, vol 5331. In: OTM Conferences (1). Lecture Notes in Computer Science, vol 5331. Springer, pp 100–119

21 Curcin, V., Miles, S., Danger, R., Chen, Y., Bache, R., Taweel, A. (2014) Implementing interoperable provenance in biomedical research. Future Gener Comput Syst 34:1–16

22 da Cruz, S. M. S., Costa, R. M., Manhães, M., Zavaleta, J. (2013) Monitoring soa-based applications with business provenance. In: Proceedings of the 28th anual ACM symposium on applied computing, ACM, pp 1927– 1932

23 DeFee, J., Harmon, P. (2005) Workflow handbook. In: Future strategies, chapter business activity monitoring and simulation, pp 53–74

24 Domínguez, E., Pérez, B., Rubio, A. L., Zapata, M. A., Lavilla, J., Allué, A. (2014) Occurrence-Oriented Design Strategy for Developing Business Process Monitoring Systems. IEEE Trans. Knowl. Data Eng. 26(7), 1749–1762

25 Domínguez, E., Pérez, B., Rubio, A. L., Zapata, M. A., Allué, A., López, A. (2017) Developing provenance-aware query systems: an occurrence-centric approach. Knowledge and Information Systems 50, pp 661–688, DOI 10.1007/s10115-016-0950-z. http://link.springer.com/article/10.1007%2Fs10115-016-0950-z

26 Domínguez, E., Pérez, B., Rubio, A. L., Zapata, M. A., Allué, A., López, A. (2017) Generating persistence structures for business process monitoring purposes: the Occurrence–centric approach, enviado a 15th International Conference on Business Process Management (BPM 2017)

27 Freire, J., Koop, D., Santos, E., Silva, C. T. (2008) Provenance for computational tasks: a survey. Comput Sci Eng 10(3):11–21

28 Gadelha, L. M. Jr., Clifford, B., Mattoso, M., Wilde, M., Foster, I. (2011) Provenance management in swift. Future Gener Comput Syst 27(6):775–780

29 Gamma, E., Helm, R. Johnson, R.; Vlissides, J. (1995) Design Patterns: Elements of

Reusable Object-Oriented Software. Addison-Wesley. ISBN 0-201-63361-2

30 Gerede, C. E., Bhattacharya, K., Su, J. (2007) Static analysis of business artifact-centric operational models. In: Proceedings of SOCA'07, pp 133–140

31 Glavic, B., Dittrich, K. R. (2007) Data provenance: a categorization of existing approaches. In: Datenbanksysteme in business, technologie und web (BTW'07), pp 227–241

32 Glavic, B., Miller, R. J., Alonso, G. (2013) Using sql for efficient generation and querying of provenance information. In: Tannen, V., Wong, L., Libkin, L., Fan, W., Tan, W-C., Fourman, M. P. (eds) In search of elegance in the theory and practice of computation. Springer, Berlin, pp 291–32

33 Groth, P., Moreau, L. (2013) An Overview of the PROV Family of Documents. W3C Note. Available at: https://www.w3.org/TR/prov-overview/. Ultimo acceso en mayo de 2016

34 Holland, D. A., Braun, U. J., Maclean, D., Muniswamy-Reddy, K-K., Seltzer, M. I. (2008) Choosing a data model and query language for provenance. In: Proceedings of the 2nd international provenance and annotation workshop

35 Hull, R., King, R. (1987) Semantic database modeling: survey, applications, and research issues. ACM Comput. Surv. 19, 3, 201-260. DOI=http://dx.doi.org/10.1145/45072.45073

36 IBM. Dictionary of IBM & Computing Terminology. http://www-03.ibm.com/ibm/history/documents/pdf/glossary.pdf. Último acceso en febrero de 2017

37 ISO/IEC 9075 Information technology — Database languages — SQL (Draft). Whitemarsh Information Systems Corporation. http://www.wiscorp.com/sql200n.zip

38 ISO/IEC 14977. Information technology - Syntactic metalanguage - Extended BNF. http://standards.iso.org/ittf/PubliclyAvailableStandards/s026153_ISO_IEC_14977_1996(E).zip

39 Joglekar, G. S., Giridhar, A., Reklaitis, G. (2014) A workflow modeling system for capturing data provenance. Comput Chem Eng 67:148–158

40 Kang, B., Kim, D., Kang, S-H. (2012) Real-time business process monitoring method for prediction of abnormal termination using KNNI-based LOF prediction. Expert Syst Appl 39(5):6061–6068

41 Kang, D., Lee, S., Kim, K., Lee, J. Y. (2009) An OWL-based semantic business process monitoring framework. Expert Syst Appl 36(4):7576–7580

42 Karsai, G., Krahn, H., Pinkernell, C., Rumpe, B., Schneider, M., Völkel, S. (2009) Design guidelines for domain specific languages. In: Proceedings of the 9th OOPSLA workshop on domain-specific modeling (DSM'09), pp 7–13

43 Karvounarakis, G., Ives, Z. G., Tannen, V. (2010) Querying data provenance. In: Proceedings of SIGMOD'10, ACM, pp 951–962

44 Ko, R. K. (2009) A computer scientist's introductory guide to business process management (BPM). Crossroads 15(4):4:11–4:18

45 Kobryn, C. (2000) Architectural patterns for metamodeling. In: Evans A, Kent S, Selic B (eds) UML'00 — the unified modeling language. LNCS, vol 1939. Springer, Berlin, pp 265–277 31

46 Lucia, A. D., Deufemia, V., Gravino, C., Risi, M. (2009) Design pattern recovery through visual language parsing and source code analysis. J Syst Softw 82(7):1177–1193

47 Microsoft. Querying the SQL Server System Catalog. https://technet.microsoft.com/en-us/library/ms189082(v=sql.105).aspx. Último acceso en febrero de 2017

48 Moreau, L. (2010) The foundations for provenance on the web. Found Trends Web Sci 2(2–3):99–241

49 Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., Plale, B., Simmhan, Y., Stephan, E., den Bussche, J. V. (2011) The open provenance model core specification (v1. 1). Future Gener Comput Syst 27(6):743–756

50 Moreau, L., Groth, P., Miles, S., Vazquez-Salceda, J., Ibbotson, J., Jiang, S., Munroe, S., Rana, O., Schreiber, A., Tan, V., Varga, L. (2008) The provenance of electronic data. Commun ACM 51(4):52–58

51 Moreau, L., Missier, P. (2013) PROV-DM: The PROV data model, technical report, World Wide Web Consortium. http://www.w3.org/TR/prov-dm/

52 Mukhi, N. K. (2010) Monitoring unmanaged business processes. In: Proceedings of the 2010 international conference on the move to meaningful internet systems—volume part I', OTM'10, Springer, pp 44–59

53 National Center for Analysis of Longitudinal Data in Education Research (CALDER). What are Longitudinal Data? http://www.caldercenter.org/what-are-longitudinal-data. Último acceso en diciembre de 2016

54 OMG (2015) OMG Unified Modeling Language (OMG UML). http://www.omg.org/spec/UML/2.5/PDF/. Último acceso en marzo de 2017

55 OMG (2013) OMG Model Driven Architecture. http://www.omg.org/mda. Último acceso en diciembre de 2016

56 Oracle. The Data Dictionary. http://docs.oracle.com/cd/B10501_01/server.920/a96524/c05dicti.htm. Último acceso en febrero de 2017

57 Rebuge, A., Ferreira, D. R. (2012) Business process analysis in healthcare environments: A methodology based on process mining. Inform. Syst., vol. 37, no. 2, pp. 99–116

58 Reichert, M., Bassil, S., Bobrik, R., Bauer, T. (2010) The proviado access control model for business process monitoring components. Enterp Model Inf Syst Archit Int J 5(3):64–88

59 Rogosa, D., Brandt, D,; Zimowski, M, (1982) A growth curve approach to the measurement of change. Psychological Bulletin, Vol 92(3), 726-748. http://dx.doi.org/10.1037/0033-2909.92.3.726

60 Rozinat, A., van der Aalst, W. M. P. (2008) Conformance checking of processes based on monitoring real behavior. Inform. Syst., vol. 33, no. 1, pp. 64–95

61 Satya S. S., Amit, P. S. (2009) Provenir Ontology: Towards a Framework for eScience Provenance Management. http://works.bepress.com/amit_sheth/281/. Ultimo acceso en mayo de 2016

62 Scheidegger, C., Koop, D., Santos, E., Vo, H., Callahan, S., Freire, J., Silva, C. (2008) Tackling the provenance challenge one layer at a time. Concurr Comput Pract Exp 20(5):473–483

63 Simmhan, Y., Plale, B., Gannon, D. (2005) A survey of data provenance in e-science. ACM Sigmod Rec 34(3):31–36

64 ter Hofstede, A. H. M., van der Aalst, W. M. P., Adams, M., Russell, N. (2010) Modern business process automation: YAWL and its support environment. Springer, Heidelberg

65 Tian, H., Sunderraman, R., Yian, H. (2007) A domain-specific conceptual data modeling and querying methodology. In: Proceedings of the 1st international conference on information systems, technology and management, New Delhi, India

66 van der Aalst, W. M. P. (2007) Exploring the CSCW spectrum using process mining. Adv Eng Inf 21(2):191– 199

67 van der Aalst W., Weijter, A., Maruster, L. (2003) Workflow mining: discovering process models from event logs. IEEE Trans Knowl Data Eng 16(9):1128–1142

68 W3C (World Wide Web Consortium). Extensible Markup Language (XML). https://www.w3.org/XML/. Último acceso en diciembre de 2016

69 Weske, M. (2012) Business Process Management: Concepts, Languages, Architectures. Second Edition. Springer

70 Widom, J. (2008) Trio: a system for data, uncertainty, and lineage. In: Aggarwal CC (ed) Managing and mining uncertain data. Springer, Berlin, pp 113–148

71 Xhafa, F.; López, A.; Caballe, S.; Kolici, V.; Barolli, L. (2012) Mining Navigation Patterns in a Virtual Campus. Emerging Intelligent Data and Web Technologies (EIDWT), Third International Conference on, vol., no., pp.181-189, 19-21 Sept. 2012. doi: 10.1109/EIDWT.2012.65. Available at: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6354739

72 Xian, L. (2015) Introduction to longitudinal data analysis in psychiatric research. Shanghai Arch Psychiatry. Aug 25; 27(4): 256–259. doi: 10.11919/j.issn.1002-0829.215089. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4621293/

73 Xian, L. (2016) Methods and Applications of Longitudinal Data Analysis. Elsevier. ISBN: 978-0-12-801342-7