Depurador de SQL y Datalog para DESweb
SQL and Datalog Debuggers for DESweb

Trabajo de Fin de Grado
Curso 2022–2023

Autores
Laura Buquerín Arias
David Stetco

Director
Fernando Sáenz Pérez

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Depurador de SQL y Datalog para DESweb
SQL and Datalog Debuggers for DESweb

Trabajo de Fin de Grado en Ingeniería Informática

Autores
Laura Buquerín Arias
David Stetco

Director
Fernando Sáenz Pérez

Convocatoria: *Junio 2023*

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

29 de mayo de 2023

Resumen

Depurador de SQL y Datalog para DESweb

En el mundo de la programación una de las tareas más importantes para la correcta realización del trabajo es resolver los errores que van surgiendo en el código de la manera más eficiente posible. Esto se lleva a cabo a través de lo que se conoce en informática como la fase de depuración. Para ello, la mayoría de entornos de desarrollo integrados (Integrated Development Environment o IDE) nos ofrecen una interfaz de usuario intuitiva y cómoda de utilizar con la mayor parte de los lenguajes de programación más utilizados hoy en día.

DES (Datalog Educational System) es un sistema de bases de datos deductivas de código abierto, multiplataforma y con implementación basada en Prolog. Permite trabajar a través de una consola e incluye los lenguajes de consulta Álgebra Relacional, TRC, DRC, Datalog y SQL. Haciendo uso de esta herramienta, DESweb nace como una aplicación web que ofrece a los usuarios un entorno gráfico de desarrollo mucho más accesible que facilita la comprensión y enseñanza de estos tipos de bases de datos.

Pese a que DES proporciona un proceso de depuración para nuestras bases de datos, este se lleva a cabo mediante el uso de la consola, lo que puede llegar a resultar en una tarea tediosa debido a la poca claridad que ofrece una interacción a base de comandos con el depurador. Esto puede desembocar en una mala realización de la fase de depuración y posterior pérdida de eficiencia debido al tiempo que supone corregir los errores. Además, dado que DESweb también es usado con propósitos educativos, puede dificultar y ralentizar la comprensión de los usuarios.

El objetivo de este proyecto es implementar en DESweb una interfaz de depuración SQL y Datalog utilizando como referencia la ya existente en el IDE de ACIDE (IDE de código abierto y multiplataforma encargado de presentar las funcionalidades del sistema DES), aportando más claridad en el proceso de depuración convirtiéndola en una tarea intuitiva y eficiente.

Palabras clave

DES, DESweb, SQL, Datalog, Depuración declarativa, IDE, Bases de datos, Aplicación web.

Abstract

SQL and Datalog Debuggers for DESweb

In the world of programming, one of the most important tasks for the correct performance of the work is to resolve the errors that arise in the code in the most efficient way possible. This is done through what is known in computer science as the *debugging phase*. For this purpose, most integrated development environments (IDE) offer an intuitive and easy-to-use user interface with most of the programming languages used today.

DES (Datalog Educational System) is an open source, cross-platform, deductive database system with a Prolog-based implementation. It allows working through a console and includes the query languages Relational Algebra, TRC, DRC, Datalog and SQL. Making use of this tool, DESweb is born as a web application that offers users a much more accessible graphical development environment that facilitates the understanding and teaching of these types of databases.

Although DES provides a debugging process for our databases, this is carried out through the use of the console, which can become a tedious task due to the lack of clarity offered by a command-based interaction with the debugger. This can lead to poor performance of the debug phase and subsequent loss of efficiency due to the time required to correct errors. In addition, since DESweb is also used for educational purposes, it can hinder and slow down the understanding of the users.

The objective of this project is to implement in DESweb a SQL and Datalog debugging interface using as a reference the one already existing in the ACIDE IDE (open source and multi-platform IDE in charge of presenting the functionalities of the DES system), providing more clarity in the debugging process making it an intuitive and efficient task.

Keywords

DES, DESweb, SQL, Datalog, Declarative Debugging, IDE, Databases, Web Application.

Índice

Índice de figuras

Capítulo 1

Introducción

"El futuro mostrará los resultados y juzgará a cada uno de acuerdo a sus logros"
— Nikola Tesla

En este documento se va proceder a exponer de manera detallada la realización del proyecto consistente en la implementación de una interfaz de usuario que sirva de ayuda en la tarea de depuración de bases de datos SQL y Datalog en la aplicación DESweb.

Se explicará en qué consiste el método de la depuración declarativa, del cual se hace uso en el sistema DES, así como la metodología utilizada para llevar a cabo este proceso en cuanto a la comunicación con DES se refiere y, por último, se describirán las diferentes partes de la interfaz de depuración agregadas y sus funcionalidades correspondientes.

1.1. Motivación

La tarea de depuración, también conocida como *debug*, se trata de un momento crítico a la hora de programar cualquier tipo de aplicación. Es por ello que hoy en día encontramos múltiples herramientas que proporcionan una manera eficiente de llevar a cabo este proceso. Lenguajes populares como C#, Java y Python se basan en el paradigma de la programación imperativa, donde el flujo de ejecución es claro y facilita de manera considerable la implementación de herramientas de depuración. Sin embargo, en el caso de lenguajes como SQL o Datalog, cambiamos a un paradigma de programación declarativo, donde se opera a un nivel de abstracción más alto que en los anteriores, resultando en una problemática importante a la hora de trazar un plan para encontrar errores.

DES hace uso de lo que se conoce como depuración declarativa para paliar este problema. Esta técnica, a diferencia de técnicas de depuración tradicionales, abstrae los detalles de ejecución para centrarse en los resultados, ofreciendo una manera práctica para conseguir un resultado eficiente en esta tarea. Pese a que DESweb

permite utilizar esta función mediante una comunicación a través de la consola, esta vía acaba resultando tediosa y poco intuitiva para el usuario que va a realizar una depuración de su base de datos, ya que no se tiene una referencia visual más allá de la que la consola de comandos puede llegar a ofrecer. Debido a esto, uno puede llegar a perderse entre los pasos de depuración y no conseguir su objetivo principal: encontrar y corregir el origen del problema que causa un mal funcionamiento en el código de la manera más eficiente posible.

Con esta problemática en mente, este proyecto ha surgido de la motivación por ofrecer a los usuarios de DESweb una interfaz que sirva de puente con el sistema de depuración declarativa que DES ofrece, mostrando distintos elementos como nodos y relaciones contenidos en grafos, los cuales se utilizan para representar el contexto de la base de datos. También se han implementado varias ventanas modales con información acerca de las preguntas y respuestas realizadas en cada fase de la depuración, así como las distintas tuplas contenidas en cada vista o tabla SQL, o en los predicados Datalog. Cabe destacar el uso como referencia de la aplicación ACIDE para la construcción del panel de depuración en su versión web.

1.2. Antecedentes

La aplicación ACIDE posee una gran relación con el proyecto DESweb, tratándose básicamente este último de una versión web de la misma. ACIDE es un proyecto que ha sido desarrollado por múltiples contribuidores y ha sido dirigido siempre por Fernando Sáenz Pérez, tutor de este trabajo. La cronología del desarrollo de ACIDE es la siguiente:

- Curso académico 2006-2007: versión 0.1-0.6, realizada por Diego Cardiel Freire, Juan José Ortiz Sánchez y Delfín Rupérez Cañas.
- Curso académico 2007-2008: versión 0.7, realizada por Miguel Martín Lázaro.
- Curso académico 2010-2011: versión 0.8, realizada por Javier Salcedo Gómez.
- Curso académico 2012-2013: versión 0.9-0.11, realizada por Pablo Gutiérrez García-Pardo, Elena Tejeiro Pérez de Ágreda y Andrés Vicente del Cura.
- Curso académico 2013-2014: versión 0.12-0.16, realizada por Semiramis Gutiérrez Quintana, Juan Jesús Marqués Ortiz y Fernando Ordás Lorente.
- Curso académico 2014-2015: versión 0.17, realizada por Sergio Domínguez Fuentes.
- Curso académico 2019-2020: versión 0.18, realizada por Sergio García Rodríguez.
- Curso académico 2020-2021: versión 0.19, realizada por Carlos González Torres y Cristina Lara López.

La versión 0.18 de ACIDE, realizada por Sergio García Rodríguez, implementó la interfaz de depuración SQL [5]. Posteriormente, en la versión 0.19, Cristina Lara López se encargó de implementar esta misma interfaz en su variante Datalog [6]. Estos dos trabajos han servido como referencia para este proyecto.

La aplicación DESweb se trata fundamentalmente del resultado en la especialización para el sistema DES de la herramienta *Asistente de validación del proyecto TIN2013-44742-C4-3-R Validación asistida de programas mediante análisis, anotaciones, demostraciones matemáticas y pruebas* [8, 7]. Durante este proyecto se ha modificado la interfaz y se ha añadido el panel de bases de datos con sus distintas pestañas. La interfaz general de DESweb se expondrá con mayor detalle en el capítulo 2 en la sección correspondiente.

1.3. Objetivos

Este trabajo de fin de grado posee como objetivo implementar una interfaz gráfica de depuración SQL y Datalog para la aplicación DESweb. Esta interfaz hará uso de múltiples herramientas para ofrecer a los usuarios un proceso de depuración más intuitivo y eficiente, facilitando de gran manera la localización de errores en el código.

Se añadirá un panel adicional de depuración a los ya existentes en la aplicación. Este panel podrá encontrarse oculto (comportamiento por defecto al iniciar una sesión en DESweb) y hacerse visible cuando un usuario desee comenzar una sesión de depuración. El panel contará con una representación gráfica del estado actual de la base de datos a través de un grafo formado por nodos (tablas, vistas o predicados) y sus conexiones dirigidas, correspondientes a las dependencias entre nodos. Este grafo estará ordenado jerárquicamente al igual que se muestra en la aplicación de ACIDE, colocando el nodo que se está depurando en la parte superior del panel y sus relaciones debajo en forma de niveles. Adicionalmente, se le permitirá al usuario desplazar los nodos a su antojo si desea alterar la colocación de estos según se muestra por defecto.

En la parte inferior del panel estarán situadas las distintas herramientas disponibles para la fase de depuración. Entre estas podemos encontrar algunas como la función de recargar el grafo, seleccionar la vista objeto de la depuración, configurar las opciones de depuración y el botón encargado de empezar la sesión de depuración.

Este nuevo panel formará parte de la distribución ya existente en la actual interfaz de DESweb, haciendo uso de un estilo de diseño similar y siendo posible editar sus dimensiones, tal y como se puede hacer con los otros paneles, así como cambiar entre los temas de colores existentes en la aplicación, permitiendo al usuario adecuar el estilo de la página a su gusto.

1.4. Plan de trabajo

En cuanto al plan de trabajo que se ha seguido durante este curso académico 2022-2023 se puede organizar de la siguiente forma (figura 6.1)):

- En septiembre el tutor ha proporcionado acceso a una carpeta de Google Drive donde se encontraba el código de la aplicación ACIDE, proyecto que ha servido como referencia para implementar posteriormente la interfaz de depuración en la aplicación de DESweb. Este primer mes se ha dedicado a estudiar la metodología utilizada a la hora de realizar la depuración en el sistema DES, así como la sintaxis utilizada para comunicarse a través de la API textual mediante la consola.

- En octubre el tutor ha proporcionado acceso a la carpeta de Google Drive de la versión 1.7 de DESweb, versión sobre la cual se han realizado las modificaciones necesarias para implementar la nueva interfaz de depuración. También se han estudiado los documentos referentes al manual de uso de DESweb así como la información referente a la estructura del proyecto.

- El periodo comprendido desde el mes de noviembre hasta diciembre se ha utilizado para montar el proyecto DESweb en local y realizar pruebas de funcionamiento para conocer la aplicación. Por otro lado, también ha sido necesario invertir notables cantidades de tiempo para dedicarlos al estudio del código utilizado en el proyecto, ya que la interfaz de la aplicación está implementanda utilizando el lenguaje Prolog mediante una técnica para generar HTML dinámico denominada Termerized HTML.

- Una vez comprendido el funcionamiento y código del proyecto, en enero se ha empezado a desarrollar la interfaz de depuración, utilizando de referencia la ya existente en ACIDE junto a las directrices del tutor.

- Por último, al finalizar el desarrollo de la interfaz de depuración, cubriendo por completo el flujo de depuración, se ha comenzado con la realización de la memoria del proyecto y aplicado las correcciones sugeridas por el tutor. Esto se ha llevado a cabo durante la última mitad del mes de abril y el mes de mayo.

A lo largo de todo el desarrollo del proyecto se han ido realizando consultas con el tutor a través del correo electrónico cuando la duda podía ser resuelta de esta manera. También se han realizado reuniones presenciales en la facultad cuando era necesario aclarar dudas más complejas o revisar el estado del proyecto, así como tutorías por videoconferencia.

Tareas	2022				2023				
	Sept.	Oct.	Nov.	Dic.	Enero	Febrero	Marzo	Abril	Mayo
Pruebas en ACIDE a través de la consola y primer contacto con la interfaz de depuración									
Estudio de la estructura del código de DESweb, montaje del proyecto en local y pruebas de funcionamiento									
Implementación del panel de depuración en la interfaz con un contenedor vacío y las herramientas correspondientes sin funcionalidad									
Implementación del grafo de depuración en el panel									
Creación de la ventana modal de depuración y sus elementos									
Implementación de la depuración completa y su impacto en el aspecto visual del grafo									
Implementación de las herramientas del grafo: zoom, actualizar, navegar por los nodos, etc.									
Implementación del menú contextual									
Redacción y posterior corrección de la memoria del proyecto									

Figura 1.1: Diagrama de Gantt

Capítulo 2

Depuración declarativa en SQL y Datalog

En este capítulo se va a realizar una pequeña introducción a los lenguajes de programación SQL y Datalog para entender su estructura básica. Una vez realizado esto, se procederá a explicar en qué consiste la técnica de depuración declarativa utilizada en el proyecto, así como sus posibles problemas y sus respectivas soluciones.

2.1. SQL

El lenguaje SQL, de sus siglas en inglés Structured Query Language [10], se trata de uno de los más utilizados a la hora de trabajar con bases de datos. Basado en álgebra y cálculo relacional, está formado por un lenguaje de definición de datos (Data Definition Language o DDL), un lenguaje de manipulación de datos (Data Manipulation Language o DML) y un lenguaje de control de datos (Data Control Language o DCL). Debido a su naturaleza declarativa y gran nivel de abstracción permiten al usuario realizar una gran variedad de operaciones.

A través del DDL se permite modificar, borrar o definir las tablas a partir de las cuales la base de datos con la que se trabaja ordenará su información. Para este cometido se proporcionan tres operaciones básicas: CREATE, ALTER y DROP.

Por otro lado, el DML se encarga de consultar, insertar, modificar o eliminar la correspondiente información de la base de datos, proporcionando para ello otras cuatro instrucciones: SELECT, INSERT, UPDATE y DELETE.

Por último, se encuentra el DCL, el cual se encarga de manejar el control de acceso a la información almacenada en la base de datos, concediendo o retirando privilegios a los usuarios de esta. Algunos de los comandos que forman parte de este sublenguaje son GRANT y REVOKE. El primero permite conceder a usuarios específicos la posibilidad de realizar tareas específicas, mientas que el segundo permite limitar la accesibilidad de un usuario de la base de datos a los objetos de esta. Cabe destacar que el sistema DES no cuenta actualmente con soporte para DCL.

2.2. Datalog

Datalog es un lenguaje de consulta de bases de datos deductivas basado en Prolog. Dichas bases de datos tienen la capacidad de definir reglas en un lenguaje declarativo y, gracias a determinados mecanismos de deducción, permiten obtener información adicional a la guardada explícitamente en la base de datos.

En Datalog una regla sigue la sintaxis:

```
cabeza :- cuerpo
```

Donde:

- `cabeza`: es un único átomo.
- `cuerpo`: puede estar compuesto por uno o varios átomos. Si el cuerpo de una regla está vacío, se denomina *hecho*.

Cuando una regla no tiene ningún átomo en su cuerpo se habla de un hecho, el cual está representado por la siguiente estructura:

$$X(p_1, \ldots, p_N)$$

Donde:

- `X`: nombre del predicado.
- p_i: parámetro número i.

La aridad de un predicado es el número de parámetros que recibe y se representa de la forma:

```
Predicado/Aridad
```

Por ejemplo, en el predicado $X(p_1,p_2)$ la aridad sería 2 y la representación del predicado vendría dada por `X/2`.

Pongamos ahora un ejemplo de cómo se podría inferir información a partir de reglas:

Supongamos que tenemos el predicado `mother(X,Y)` y que en la base de datos almacenamos la siguiente tupla:

```
mother(grace,amy)
```

Por otro lado tenemos el predicado `father(X,Y)` con las siguiente tupla:

```
father(tom,amy)
```

Estas tuplas están almacenadas de forma explícita en la base de datos. Ahora vamos a suponer la siguiente regla:

```
parent(X,Y) :- father(X,Y); mother(X,Y).
```

Esta regla expresa que, si se cumple `father(X,Y)` **o bien** se cumple `mother(X,Y)`, entonces se cumplirá `parent(X,Y)`.

Usando el ejemplo anterior:

- `Grace` es madre de `Amy`, por lo que la tupla `(grace,amy)` es válida para el predicado `parent(X,Y)`.
- `Tom` es padre de `Amy`, por lo que la tupla `(tom,amy)` es válida para el predicado `parent(X,Y)`.

De esta manera hemos conseguido inferir información a partir de las tuplas guardadas en la base de datos y podemos saber que las siguientes tuplas de deducen del programa:

- `parent(grace,amy).`
- `parent(tom,amy).`

Por lo que, como se ha podido ver en el ejemplo, en Datalog existen dos tipos de predicados:

- **Predicado extensional**: este que no tiene cuerpo (y, por lo tanto, es considerado un hecho). Sus tuplas se almacenan explícitamente en la base de datos.
- **Predicado intensional**: este compuesto por una o más reglas y del que se infiere la información.

También cabe destacar que Datalog permite hacer uso de la recursividad, tal y como se muestra en el siguiente predicado:

```
ancestor(X,Y) :- parent(X,Y).

ancestor(X,Y) :- parent(X,Z), ancestor(Z,Y).
```

En esta regla se indica que `X` es ancestro de `Y` si se cumple que:

- `X` es progenitor de `Y`.
- `X` es progenitor de `Z` y `Z` a su vez es un ancestro de `Y`.

2.3. Depuración declarativa

La siguiente sección es un pequeño resumen realizado a partir de los trabajos *Algorithmic Debugging of SQL Views* [4] y *Declarative Debugging of Wrong and Missing Answers for SQL Views* [3].

Cuando el usuario se encuentra con una vista que posee un dato erróneo, comúnmente denominado como *bug*, debe empezar una ardua tarea de revisión sobre todas las vistas y tablas de las cuales esta vista inicial depende para obtener su información, consumiendo de esta manera una gran cantidad de tiempo y esfuerzo.

Aquí entra en juego lo que se conoce como la depuración declarativa, técnica implementada por DES para proporcionar una herramienta de revisión de vistas erróneas en nuestros programas. Se basa en el siguiente flujo:

- El proceso comienza con la identificación de un error correspondiente al resultado inesperado de una vista definida por el usuario.
- El depurador automáticamente construye un grafo de tipo árbol donde cada nodo correspondiente a una tabla o vista que sirve como computación intermedia para obtener el resultado final del nodo al cual se le ha detectado en un primer momento el error.
- Este grafo es recorrido a medida que el usuario compara cada resultado obtenido con el resultado esperado asociado a la relación. Si estos coinciden se declara ese nodo como nodo válido. En el caso contrario en el cual se detecte una anomalía se marca como no válido.
- Este algoritmo termina cuando nos topamos con un nodo no válido cuyos nodos hijos se tratan de nodos válidos. En este caso, se denomina este nodo como *buggy* y el depurador finalizará indicando el nombre de esta vista o tabla, origen del error en el programa.

La depuración declarativa está basada en la comparación entre las respuestas esperadas y las respuestas reales obtenidas a la hora de realizar las consultas SQL sobre las distintas tablas o vistas de la base de datos. Estas respuestas se obtienen a partir de una fuente externa confiable que puede almacenar la información correcta del sistema o, en el caso de DESweb, el usuario de la aplicación. Este se encargará de responder preguntas de la forma *"¿Es la siguiente respuesta (...) la esperada para la vista V?"* realizadas por el depurador, navegando el grafo de árbol hasta que se obtenga toda la información necesaria para localizar el nodo erróneo.

Este algoritmo es muy eficiente e intuitivo ya que emula el procedimiento que una persona seguiría para analizar y encontrar un error a la hora de obtener una respuesta inesperada. Pese a esto, se pueden encontrar algunas críticas frente a este método de depuración entre las cuales encontramos las siguientes:

- Una vez terminada la depuración se indica la vista errónea, sin embargo, no se indica qué parte asociada a la consulta es la causante de la respuesta inesperada. Pese a esto, el hecho de encontrar la vista culpable en un contexto

complejo de un sistema con docenas de computaciones intermedias en forma de vistas disminuye notablemente el tiempo y esfuerzo empleado en la tarea de depuración.

- El depurador puede llegar a formular una gran cantidad de preguntas, lo que supondría un importante inconveniente en contextos de programación imperativa, donde los árboles pueden llegar a contener miles de nodos. Sin embargo, en el caso de sistemas SQL, la estrategia de Divide y Vencerás requiere una media de $\log_2 n$ preguntas, donde n es el número de preguntas necesarias. Por ejemplo, en un árbol que contiene 100 vistas, hará falta una media de 7 preguntas antes de encontrar el nodo erróneo.

- Algunas de las preguntas pueden ser difíciles de contestar para el usuario del sistema, punto importante teniendo en cuenta el alto número de filas en instancias de bases de datos reales. Para paliar este problema, el sistema DES posee una herramienta capaz de generar pequeñas instancias de la base de datos que permiten al usuario revisar varias vistas relacionadas dentro del sistema.

Como herramienta de apoyo a la depuración declarativa, se ha desarrollado una interfaz de depuración en DESweb que aporte información adicional al usuario a la hora de entrar en una sesión de depuración.

Capítulo 3

Comunicación con DES

En este capítulo se explicará el método de funcionamiento de la comunicación de la aplicación DESweb con el sistema DES. Para esto, se ha utilizado información encontrada en el propio manual de DES [9].

Como ya se ha mencionado anteriormente en este documento, DES se trata de un sistema deductivo desarrollado con la intención de ser utilizado para enseñar el lenguaje de Datalog. Basado en Prolog, permite trabajar con bases de datos relacionales con los lenguajes Álgebra Relacional (Relational Algebra o RA), Cálculo Relacional de Tuplas (Tuple Relational Calculus o TRC), Cálculo Relacional de Dominios (Domain Relational Calculus o DRC) y Datalog. Además de un IDE (Integrated Development Environmet) gráfico y configurable denominado ACIDE, también hay habilitado un dominio *front-end* online denominado DESweb, aplicación sobre la que se ha desarrollado la interfaz de depuración gráfica objeto de este proyecto.

Cabe mencionar que, pese a que DES ha sido desarrollado para funciones pedagógicas, también ha sido utilizado para desarrollar algunas extensiones novedosas mencionadas en el manual de DES [9].

La comunicación entre el sistema DES y DESweb se realiza a través de servicios web implementados en Prolog. Estos servicios se encargan de realizar las distintas peticiones añadiendo los parámetros necesarios y obteniendo así la información del contexto de la base de datos que está siendo utilizada por el usuario en ese momento. DES proporciona una API textual (TAPI) con este propósito, la cual permite conectarse a las aplicaciones externas como ACIDE o, en este caso, DESweb.

Esta TAPI entrega información estructurada de dos posibles formas:

- Respuesta exitosa:

```
$success
```

- Respuesta errónea:

```
$error
```

```
code
text
$eot
```

El usuario puede comunicarse con el depurador y progresar en la sesión de depuración introduciendo comandos en la consola de DESweb. Los servicios web implementados durante el desarrollo del proyecto hacen uso de estos comandos para realizar el intercambio de peticiones y respuestas de la misma manera en la que un usuario lo realizaría en la consola. La nueva interfaz de depuración proporciona una abstracción frente a estos comandos para llevar a cabo este intercambio, aprovechando elementos visuales como el grafo y los diálogos.

3.1. Depuración de SQL

A continuación, se van a presentar los distintos comandos utilizados en una sesión de depuración SQL para conocer sus utilidades y posibilidades. Todas estas facetas están representadas de una manera u otra en la interfaz.

Para comenzar con al depuración se hace uso del siguiente comando:

```
/debug_sql View [Options]
```

Donde:

- `View`: indica la vista donde el usuario ha detectado un error.
- `Options`: establece las opciones a tener en cuenta por el depurador a la hora de comenzar la sesión de depuración.

Dentro del parámetro `Options` pueden venir especificadas las siguientes opciones:

```
Options =   [Answer]
            [trust_tables([yes|no])][trust_file(FileName)]
            [oracle_file(FileName)] [debug([full|plain])]
            [order([cardinality|topdown])]
```

Donde:

- `Answer`: permite iniciar la depuración añadiendo una respuesta inicial que indica el error localizado en la vista.
- `trust_tables`: permite indicar si las tablas existentes en la base de datos objeto de depuración se consideran como válidas.
- `trust_file`: permite indicar si las vistas definidas en el archivo proporcionado se consideran como válidas.

- `oracle_file`: permite utilizar un fichero *oracle automaton*, donde estarán definidas las especificaciones para depurar automáticamente.
- `debug`: especifica el modo de depuración empleado por el depurador, siendo la opción *full* habilitadora de las opciones de tupla faltante o errónea.
- `order`: especifica el orden de recorrido de los nodos del grafo de depuración.

Las opciones por defecto son: `Answer` vacío, `trust_tables(yes)`, `trust_file` vacío, `oracle_file` vacío, `debug(full)` y `order(cardinality)`.

Entre las posibles respuestas proporcionadas por el usuario al depurador se encuentran las siguientes:

```
Answer =    abort                 |
            valid                 |
            nonvalid              |
            missing(PartialTuple) |
            wrong(TotalTuple)
```

Donde:

- `abort`: permite al usuario abortar la sesión de depuración.
- `valid`: indica que el nodo es válido.
- `nonvalid`: indica que el nodo no es válido.
- `missing`: indica que el nodo no es válido, agregando como causa del error una tupla faltante. Esta tupla debe ser parcial.
- `wrong`: indica que el nodo no es válido, agregando como causa del error una tupla errónea. Esta tupla debe ser total.

El sistema continuará realizando preguntas al usuario acerca del estado de los distintos nodos relacionados con la vista inicial. Las preguntas realizadas por el depurador pueden tener siguiente forma:

```
Question =  all(RelationName)                   |
            subset(RelationName1,RelationName2) |
            in(Tuple,RelationName)
```

Donde:

- `all`: cuestiona si la información contenida en `RelationName` es correcta. Cualquier tipo de respuesta es válida.
- `subset`: cuestiona si las tuplas contenidas en `RelationName1` deberían ser un subconjunto de las tuplas contenidas en `RelationName2`. Solo admite las respuestas `valid` y `nonvalid`.

- `in`: cuestiona si `Tuple` debería estar contenida en `RelationName`. Solo admite las respuestas `valid` y `nonvalid`.

La interfaz está preparada para representar estos posibles formatos de pregunta a través de la ventana de depuración de la aplicación web, exponiendo frente al usuario toda la información disponible respecto a las tuplas contenidas en la vista o tabla que se encuentra depurando dentro del recorrido en el grafo de depuración. El usuario hará uso de las herramientas de la interfaz para responder a la cuestión planteada por el sistema.

Una vez comenzada la sesión de depuración mediante la llamada `/debug_sql`, se deberá seguir respondiendo al sistema con la información solicitado por el depurador para encontrar el nodo erróneo. Para ello, se hace uso del siguiente comando:

```
/debug_sql_answer Question Answer
```

Donde:

- `Question`: representa la pregunta formulada por el depurador a la cual se está respondiendo actualmente. Puede tomar cualquier valor de los expuestos anteriormente.
- `Answer`: contiene la información referente a la respuesta proporcionada por el usuario a la pregunta actual del depurador. Puede tomar cualquier valor de los expuestos anteriormente.

En cualquier momento se puede hacer uso del siguiente comando para obtener la última pregunta formulada por el depurador:

```
/debug_sql_current_question
```

Una vez que el usuario ha respondido a las preguntas que el depurador necesita para calcular el origen del error en la base de datos, el sistema devuelve el nodo erróneo marcando su estado como erróneo, finalizando la sesión de depuración. Finalizada la depuración se muestra en la ventana el nombre de la vista o tabla errónea así como información adicional sobre la sesión. Para obtener está información se hace uso del siguiente comando:

```
/debug_sql_statistics
```

A continuación, se van a mencionar algunos comandos adicionales que pueden resultar útiles a la hora de realizar una depuración.

Es posible establecer de manera directa el estado de un nodo si se conoce con certeza sin estar actualmente depurando este una vez comenzada la depuración. Para ello, se hace uso del siguiente comando:

```
/debug_sql_set_node NodeName state
```

Donde:

- `NodeName`: indica el nombre de la vista o tabla sobre la cual se va a establecer el estado.
- `state`: indicar el estado que se va a establecer al nodo objeto del comando. El valor del estado puede ser válido o no válido.

Para conocer el recorrido calculado por el depurador para recorrer el grafo de nodos de la vista que se quiere depurar se puede utilizar el siguiente comando:

```
/trace_sql NodeName
```

Donde:

- `NodeName`: indica el nombre de la vista o tabla origen del recorrido.

3.2. Depuración de Datalog

En este apartado se recogen los comandos utilizados durante una sesión de depuración de Datalog. Cada comando puede ejecutarse a través de la nueva interfaz gracias a las distintas funcionalidades que se han implementado.

Lo primero que necesitaremos será conocer los diferentes nodos para poder crear el grafo de dependencias *PDG*. Para ello usaremos el comando:

```
/pdg
```

Este comando nos devuelve dos listas: una con los predicados y sus aridades y otra con las diferentes relaciones entre los nodos.

Una vez tengamos la información de los diferentes predicados, podremos decidir cuál queremos que sea el nodo inicial para la depuración.

Para obtener las tuplas correspondientes a un predicado deberemos escribir en la consola el nombre del predicado seguido de el número de parámetros indicado en la aridad. Por ejemplo, si tenemos el predicado *ancestor/2* podremos obtener sus tuplas con:

```
ancestor(X,Y).
```

Para iniciar una sesión de depuración de Datalog se utiliza el comando:

```
/debug_dl nombrePredicado/Aridad [Options]
```

Donde:

- `nombrePredicado/Aridad`: indica el predicado sobre el que se va a llevar a cabo la depuración.
- `Options`: establece las opciones a tener en cuenta por el depurador a la hora de comenzar la sesión de depuración.

Dentro del parámetro `Options` pueden venir especificadas las siguientes opciones:

```
Options =   [Answer]
            [trust_extension([yes|no])]
            [file(FileName)]
```

Donde:

- `Answer`: permite dar una respuesta de antemano a la primera pregunta que se realiza en la depuración.
- `trust_extension`: permite indicar si se debe confiar en los predicados extensionales.
- `file`: nombre del fichero Datalog que contiene el programa a depurar. El nombre debe ir seguido de la extensión *.dl*.

Las opciones por defecto son: `Answer` vacío, `trust_extension(no)` y `file` vacío.

Entre las posibles respuestas proporcionadas por el usuario al depurador se encuentran las siguientes:

```
Answer =    abort                 |
            valid                 |
            nonvalid              |
            missing(PartialTuple) |
            wrong(TotalTuple)
```

Donde:

- `abort`: permite al usuario abortar la sesión de depuración.
- `valid`: indica que el nodo es válido.
- `nonvalid`: indica que el nodo no es válido.
- `missing`: indica que el nodo no es válido, agregando como causa del error una tupla faltante. Esta tupla debe ser parcial.
- `wrong`: indica que el nodo no es válido, agregando como causa del error una tupla errónea. Esta tupla debe ser total.

El sistema continuará realizando preguntas al usuario acerca del estado de los distintos nodos relacionados con el predicado inicial. Las preguntas realizadas por el depurador pueden tener cualquiera de las siguientes estructuras:

```
Question =  all(Predicado/Aridad)                          |
            subset(Predicado1/Aridad,Predicado2/Aridad)    |
            empty(Predicado/Aridad)                        |
            nonempty(Predicado/Aridad)
```

Donde:

- `all`: cuestiona si las tuplas del predicado `Predicado` son correctas. Cualquier tipo de respuesta es válida.
- `subset`: cuestiona si las tuplas contenidas en `Predicado1` deberían ser un subconjunto de las tuplas contenidas en `Predicado2`. Solo admite las respuestas `yes` y `no`.
- `empty`: pregunta si el predicado `Predicado` debería estar vacío y no contener tuplas. Solo admite las respuestas `yes` y `no`.
- `nonempty`: pregunta si el predicado `Predicado` no debería estar vacío y, por lo tanto, debería contener alguna tupla. Solo admite las respuestas `yes` y `no`.

La interfaz gráfica implementada permite mostrar las distintas preguntas que el depurador realiza al usuario, así como tablas con las tuplas contenidas en los predicados. Para responder a estas el usuario cuenta con diferentes botones que le permiten establecer cada una de las respuestas.

Una vez comenzada la sesión de depuración mediante la llamada `/debug_dl`, el sistema continuará haciendo preguntas al usuario y procesando las diferentes respuestas hasta dar con el nodo erróneo.

Dichas respuestas se envían al sistema a través del siguiente comando:

```
/debug_dl_answer Question Answer
```

Donde:

- `Question`: pregunta que está realizando el depurador al usuario.
- `Answer`: respuesta que proporciona el usuario a la pregunta actual realizada por el depurador.

Para obtener la pregunta actual que el depurador plantea al usuario se utiliza el comando:

```
/debug_dl_current_question
```

Cuando el depurador encuentra el nodo erróneo gracias a las respuestas del usuario, devuelve una respuesta en la que marca el nodo con el estado *erroneous*, finalizando así la depuración. En este momento el sistema mostrará al usuario una ventana con el nombre del nodo erróneo, así como información sobre la sesión de la depuración y los diferentes errores encontrados durante la misma. Dicha información se obtiene mediante el comando:

```
/debug_dl_statistics
```

Existen otros comandos que pueden ser de utilidad para el usuario durante la depuración. Por ejemplo, es posible establecer de manera directa el estado de un nodo mediante el comando:

```
/debug_dl_set_node Predicado/Aridad State
```

Donde:

- `Predicado/Aridad`: indica el nombre del predicado sobre el cuál se va a establecer el estado y su aridad.
- `State`: estado que se quiere establecer para el predicado indicado. Entre los posibles estados encontramos: `valid`, `nonvalid`, `missing(PartialTuple)` y `wrong(TotalTuple)`.

También es posible consultar el estado de los nodos en cada momento de la depuración con el comando:

```
/debug_dl_node_state
```

Capítulo 4

Diseño de la interfaz de depuración

En este capítulo se va a proceder a exponer la interfaz desarrollada a lo largo del proyecto. Primero, se va a realizar una pequeña introducción a la interfaz utilizada como referencia de la aplicación ACIDE, mostrando los elementos claves que forman parte de la sesión de depuración. Después de esta explicación, se expondrán las tecnologías utilizadas así como el resultado de la implementación de la interfaz en la aplicación DESweb, indicando posibles cambios realizados y su razón respecto a los elementos presentes en la versión de ACIDE.

4.1. ACIDE

Toda la información expuesta en esta sección ha sido extraída del manual de ACIDE el cual se puede obtener directamente de la página de la misma aplicación [1].

Al igual que DESweb, ACIDE (A Configurable IDE) se trata de un entorno de desarrollo integrado, de código abierto y multiplataforma el cual ofrece a los usuarios una interfaz gráfica parametrizada y especializada para el sistema DES. La implementación de la aplicación está principalmente realizada bajo el lenguaje de programación Java, utilizando también HTML y XML para la presentación y representación de datos.

En la parte superior de la aplicación se localiza la barra de menú, la cual alberga funcionalidades generales referentes al manejo de los ficheros, acciones útiles de la aplicación como rehacer y deshacer cambios, el buscador de términos y más funciones referentes al manejo de proyectos entre otras. La aplicación cuenta con múltiples paneles en la vista principal que exponen información relevante durante toda la sesión de un usuario. Entre alguno de los paneles que podemos encontrar se encuentran el panel de la consola, el explorador de archivos, el panel de base de datos, etc. Este documento se va a centrar en mostrar principalmente el panel de depuración debido a la relevancia que tiene con el proyecto (figura 4.1).

El panel de depuración alberga cuatro pestañas referentes a la funcionalidad de

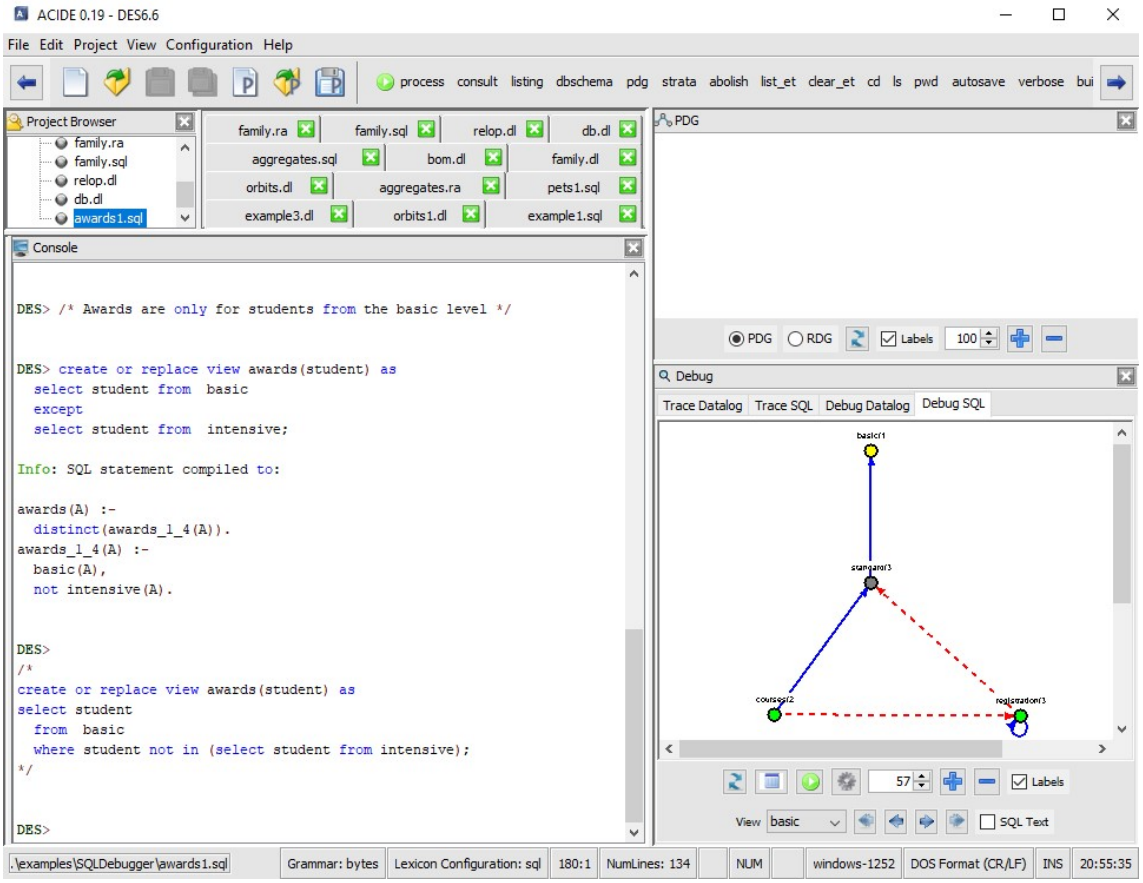

Figura 4.1: Vista principal de ACIDE al iniciar la aplicación y cargar una base de datos

traza y depuración de bases de datos SQL y Datalog.

Las pestañas Trace Datalog (figura 4.4) y Trace SQL (figura 4.2) muestran un grafo navegable de dependencias de la base de datos para un predicado Datalog o vista SQL respectivamente. Se pueden utilizar las herramientas presentes en el panel para actualizar la interfaz, acercar o alejar el zoom, mostrar u ocultar las etiquetas de cada nodo, trazar un predicado o vista a través del botón de *query* que abre una ventana, consultar las tuplas de la vista o predicado seleccionado, y avanzar en el recorrido ya sea hacia adelante o hacia atrás.

Las pestañas correspondientes a los paneles de depuración SQL (figura 4.3) y Datalog (figura 4.5) presentan un aspecto similar a los anteriores, diferenciándose en la adición de funcionalidad correspondiente a una depuración. Mantienen los botones de actualizar el panel, configurar el zoom o las etiquetas y navegar en el recorrido de nodos.

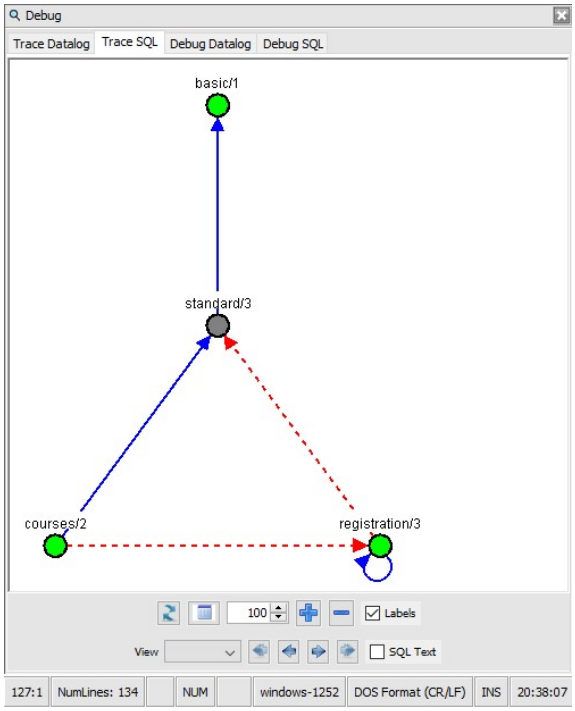

Figura 4.2: Pestaña correspondiente al panel de traza de SQL

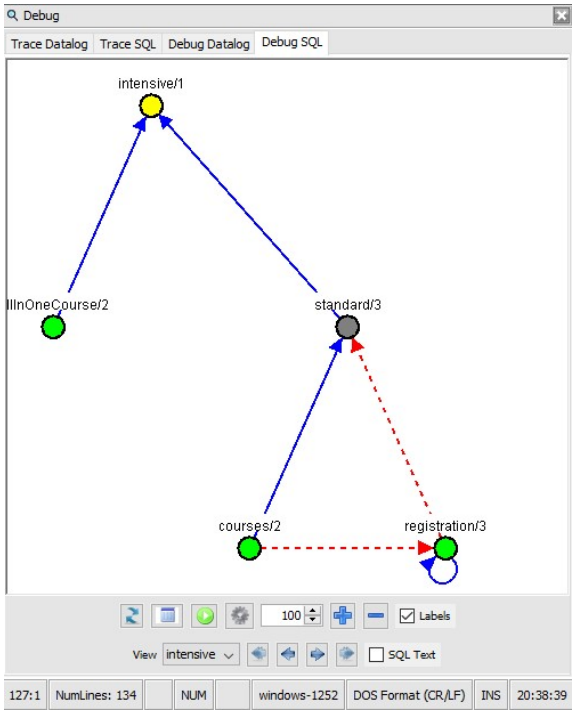

Figura 4.3: Pestaña correspondiente al panel de depuración de SQL

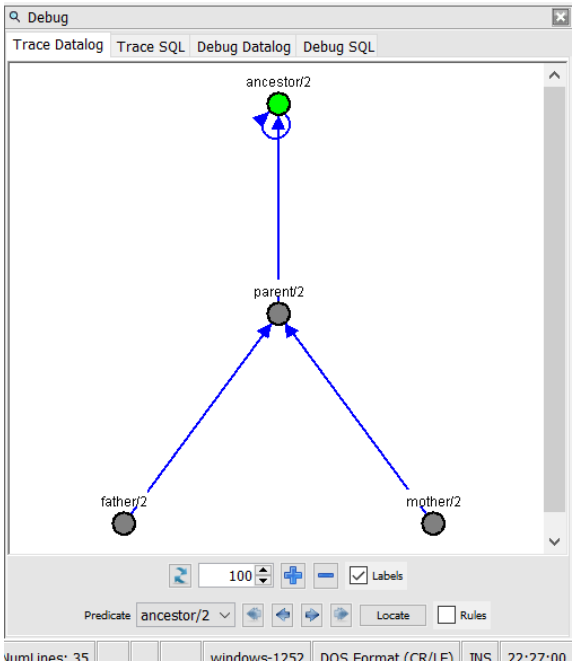

Figura 4.4: Pestaña correspondiente al panel de traza de Datalog

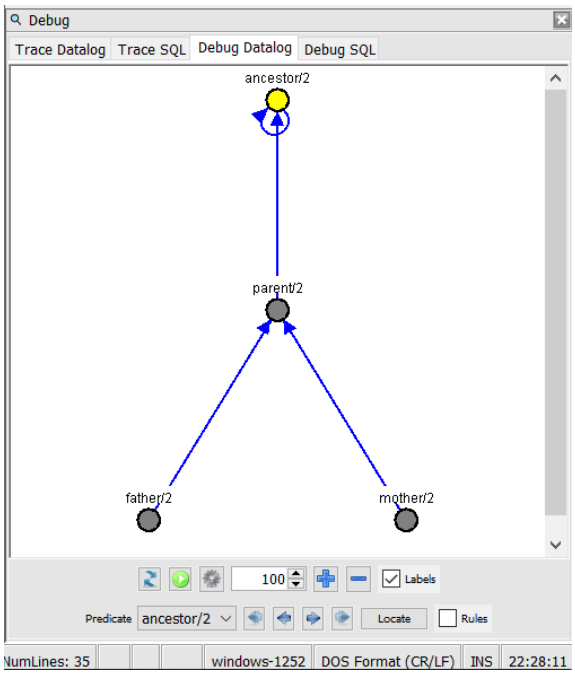

Figura 4.5: Pestaña correspondiente al panel de depuración de Datalog

Entre los nuevos botones se encuentran aquellos que manejan la depuración, como la configuración inicial que se va a utilizar en la sesión de depuración y el propio botón de inicio de esta.

Una vez iniciada la depuración se muestra la ventana de depuración que, aparte de exponer la pregunta del depurador al usuario, muestra información relevante sobre la vista o predicado actual, como el número de tuplas y su contenido, y los errores capturados a lo largo de la sesión de depuración (figura 4.6).

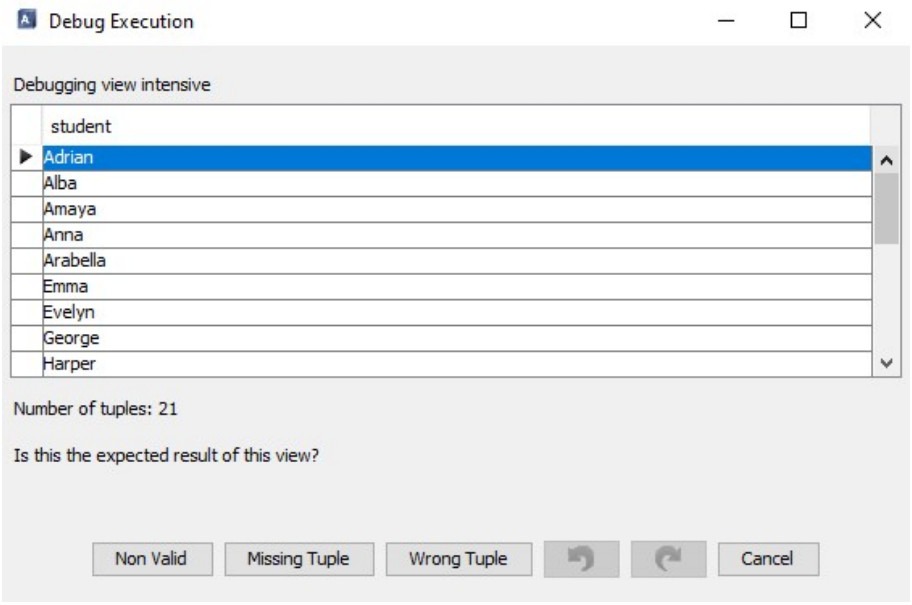

Figura 4.6: Ventana de depuración una vez iniciado el proceso

Una vez terminada la depuración, la ventana indica la vista o predicado que se ha detectado como erróneo con la información proporcionada a lo largo de la sesión (figura 4.7).

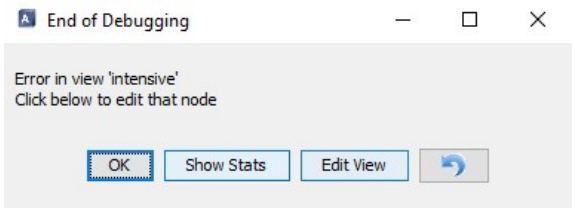

Figura 4.7: Ventana final de la sesión de depuración

En esta ventana se pueden realizar diferentes acciones:

- El usuario puede terminar la depuración pulsando el botón OK.
- Mediante la opción Show Stats el usuario puede consultar información relevante de la sesión de depuración (figura 4.8).
- También proporciona la posibilidad de editar la vista o predicado erróneo para solucionar el error encontrado (figura 4.9).

- Por último, se permite deshacer el último paso en la depuración para proporcionar otra respuesta si se cree que la solución a la que se ha llegado no es correcta.

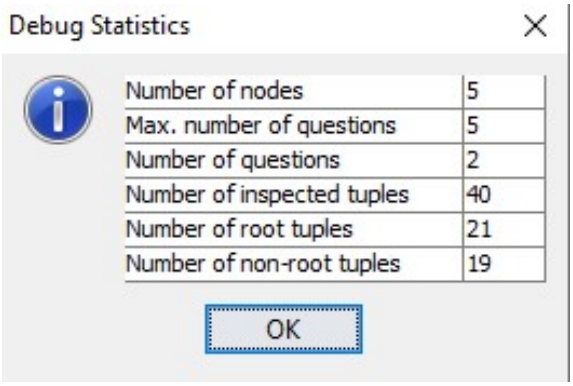

Figura 4.8: Ventana de estadísticas de la depuración

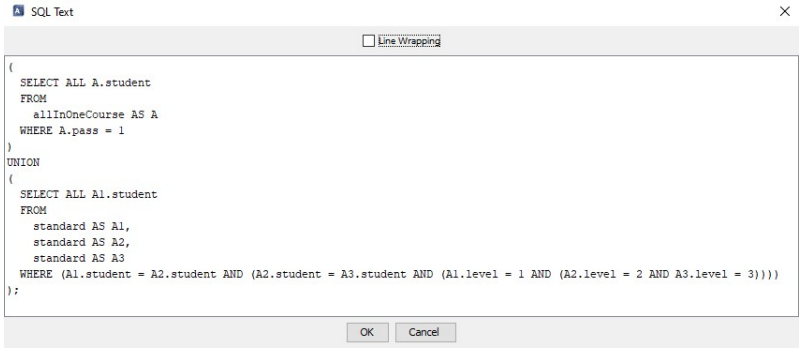

Figura 4.9: Ventana que permite editar el predicado o vista erróneo

4.2. DESweb

Las secciones correspondientes a la explicación del concepto y utilización de servicio web y HTML Termerized son una pequeña introducción a la información presente en el documento de investigación *Designing a Validation Assistant for Multi-Language Formal Verification* [2].

DESweb utiliza el sistema de programación lógica SWI-Prolog (Simple, Powerful, and Portable Prolog System). Basado en el lenguaje de programación de alto nivel Prolog, se trata de una tecnología de código abierto y ampliamente utilizada, la cual proporciona compatibilidad con múltiples plataformas, incluyendo Windows, macOS y Linux, haciéndolo accesible a una gran variedad de sistemas operativos. Cabe destacar que es la única herramienta necesaria en el equipo para poder instalar esta aplicación.

4.2.1. Servicio Web

Para el intercambio de datos entre el servidor y el cliente, el concepto de servicio web, comúnmente conocido como *web service*, es el estándar adoptado por la W3C (World Wide Web Consortium). SWI-Prolog permite utilizar un lenguaje de programación de alto nivel y gran abstracción, reduciendo en gran parte los costes de producción, tiempo y mantenimiento para este cometido. Sus herramientas y paquetes para el desarrollo web incorporan marcos de servidores web, servicios web y el lenguaje de programación declarativo Prolog. Otro punto a favor a destacar sobre la utilización de esta tecnología es el hecho de que no requiere un servidor HTTP como puede ser Apache, ya que sus librerías incluyen el servicio HTTP.

SWI-Prolog ofrece varios canales de comunicación pero, debido a las necesidades de DESweb, se han utilizado principalmente dos de ellos.

- WebSockets: protocolo orientado a mensajes ligeros montados encima de flujos TCP/IP. Se utiliza comúnmente como una mejora de una conexión HTTP para proporcionar una comunicación bidireccional vía flujos, en particular, flujos Prolog arbitrarios. Esta aproximación permite a DESweb realizar comunicaciones interactivas.

- JSON/AJAX. JSON (JavaScript Object Notation): se trata de un formato de estándar abierto para la comunicación cliente-servidor con información construida a partir de pares clave-valor. Nació con el objetivo de sustituir XML y deriva del lenguaje JavaScript. El conjunto de herramientas AJAX (Asynchronous JavaScript and XML) permite crear aplicaciones web asíncronas. Esta aproximación permite a DESweb realizar comunicaciones no interactivas.

4.2.2. Llamadas a los servicios web

A continuación se detalla una lista de las llamadas realizadas desde el cliente a los servicios web del cliente DES para la comunicación cliente-servidor. Se especifica el endpoint del servicio, los parámetros enviados, una breve descripción del mismo y un ejemplo de cómo utilizarlo en JavaScript. Los posibles valores de los parámetros indicados corresponden a los descritos en el capítulo 3.

- `/get_rdg_json`: Obtener el gráfico de dependencia de la relación en formato JSON.

  - Parámetros:

    - `user`: usuario actual.

  - Ejemplo en navegador:

```
$.getJSON(location.protocol+"//" + location.host
+ "/get_rdg_json", {user:"test"});
```

- `/get_tables_json`: Obtener los nombres de tabla en formato JSON.

  - Parámetros:
    - `user`: usuario actual.
  - Ejemplo en navegador:

```
$.getJSON(location.protocol+"//" + location.host
+ "/get_tables_json", {user:"test"});
```

- `/debug_sql_node_state`: Visualización de los estados de los nodos de depuración SQL.
  - Parámetros:
    - `user`: usuario actual.
  - Ejemplo en navegador:

```
$.getJSON(location.protocol+"//" + location.host
+ "/debug_sql_node_state", {user:"test"});
```

- `/get_current_db_json`: Obtener la conexión actual y el DBMS como un término JSON.
  - Parámetros:
    - `user`: usuario actual.
  - Ejemplo en navegador:

```
$.getJSON(location.protocol+"//" + location.host
+ "/get_current_db_json", {user:"test"});
```

- `/debug_sql_current_question`: Devuelve la pregunta de depuración SQL actual.
  - Parámetros:
    - `user`: usuario actual.
  - Ejemplo en navegador:

```
$.getJSON(location.protocol+"//" + location.host
+ "/debug_sql_current_question", {user:"test"});
```

- `/debug_sql_statistics`: Devuelve las estadísticas de una sesión de depuración SQL.
  - Parámetros:
    - `user`: usuario actual.
  - Ejemplo en navegador:

```
$.getJSON(location.protocol+"//" + location.host
+ "/debug_sql_statistics", {user:"test"});
```

- `/debug_sql`: Iniciar una nueva sesión de depuración SQL.
  - Parámetros:

    - `user`: usuario actual.
    - `view`: nombre de la vista.
    - `trust_tables`: indica si se confía en las tablas de nuestra base de datos, es decir, en caso de confiar en las tablas sus nodos tendrán el estado 'valid'.
  - Ejemplo en navegador:

```
$.getJSON(location.protocol+"//" + location.host
+ "/debug_sql", {user:"test"});
```

- `/get_relation_json`: Obtiene las vistas y tablas de un programa SQL y sus relaciones.
  - Parámetros:
    - `user`: usuario actual.
    - `connection`: nombre de la base de datos.
    - `name`: nombre de la vista.
  - Ejemplo en navegador:

```
$.getJSON(location.protocol+"//" + location.host
+ "/get_relation_json", {user:"test", connection:"$des",
name:"allInOneCourse"});
```

- `/debug_sql_answer`: Responder a la pregunta actual del depurador SQL.
  - Parámetros:
    - `user`: usuario actual.
    - `question`: pregunta actual.
    - `answer`: respuesta a la pregunta.
  - Ejemplo en navegador:

```
$.getJSON(location.protocol+"//" + location.host
+ "/debug_sql_answer", {user:"test", question:"all(basic)",
answer:"valid"});
```

- `/trace_sql`: Iniciar una nueva sesión de rastreo SQL.
  - Parámetros:
    - `user`: usuario actual.
    - `view`: nombre de la vista.
    - `order`: especifica el orden de recorrido de los nodos del grafo de depuración.
  - Ejemplo en navegador:

```
$.getJSON(location.protocol+"//" + location.host
+ "/trace_sql", {user:"test", view:"awards", order:""});
```

- `/debug_sql_set_node`: Establece el estado de un nodo.

  - Parámetros:
    - `user`: usuario actual.
    - `node`: nombre del nodo.
    - `state`: estado del nodo.
  - Ejemplo en navegador:

```
$.getJSON(location.protocol+"//" + location.host
+ "/debug_sql_set_node", {user:"test", node:"standard",
state:"valid"});
```

- `/get_pdg_json`: Obtener el gráfico de dependencias de un programa SQL.
  - Parámetros:
    - `user`: usuario actual.
  - Ejemplo en navegador:

```
$.getJSON(location.protocol+"//" + location.host
+ "/get_pdg_json", {user:"test"});
```

- `/debug_dl_node_state`: Visualización de los estados de los nodos de depuración Datalog.
  - Parámetros:
    - `user`: usuario actual.
  - Ejemplo en navegador:

```
$.getJSON(location.protocol+"//" + location.host
+ "/debug_dl_node_state", {user:"test"});
```

- `/debug_dl`: Iniciar una nueva sesión de depuración Datalog.
  - Parámetros:
    - `user`: usuario actual.
    - `name_arity`: nombre del predicado y su aridad.
    - `trust_extension`: indica si se debe confiar los predicados extensionales.
  - Ejemplo en navegador:

```
$.getJSON(location.protocol+"//" + location.host
+ "/debug_dl", {user:"test", name_arity:"p/1",
trust_extension:"no"});
```

- `/debug_dl_current_question`: Devuelve la pregunta actual de la depuración Datalog.
  - Parámetros:
    - `user`: usuario actual.
  - Ejemplo en navegador:

```
$.getJSON(location.protocol+"//" + location.host
+ "/debug_dl_current_question", {user:"test"});
```

- `/get_dl_query_solutions_json`: Devuelve las soluciones a una consulta Datalog.
  - Parámetros:
    - `user`: usuario actual.
    - `query`: consulta Datalog.
  - Ejemplo en navegador:

```
$.getJSON(location.protocol+"//" + location.host +
"/get_dl_query_solutions_json", {user:"test", query:"p(X)"});
```

- `/debug_dl_answer`: Responder a la pregunta actual de una sesión de depuración Datalog.
  - Parámetros:
    - `user`: usuario actual.
    - `question`: pregunta actual.
    - `answer`: respuesta a la pregunta actual.
  - Ejemplo en navegador:

```
$.getJSON(location.protocol+"//" + location.host
+ "/debug_dl_answer", {user:"test", question:"all(s/1)",
answer:"valid"});
```

- `/debug_dl_node_state`: Visualización de los estados de los nodos de depuración Datalog.
  - Parámetros:
    - `user`: usuario actual.
  - Ejemplo en navegador:

```
$.getJSON(location.protocol+"//" + location.host
+ "/debug_dl_node_state", {user:"test"});
```

- `/debug_dl_solutions`: Obtención de las tuplas de una subconsulta en Datalog.
  - Parámetros:
    - `user`: usuario actual.
  - Ejemplo en navegador:

```
$.getJSON(location.protocol+"//" + location.host
+ "/debug_dl_solutions", {user:"test"});
```

- `/trace_datalog`: Iniciar una nueva sesión de rastreo Datalog.

  - Parámetros:
    - `user`: usuario actual.
    - `goal`: nombre del predicado y sus parámetros.
    - `order`: especifica el orden de recorrido de los nodos del grafo de depuración.
  - Ejemplo en navegador:

```
$.getJSON(location.protocol+"//" + location.host
+ "/trace_datalog", {user:"test", goal:"awards(X)", order:""});
```

- `/debug_dl_statistics`: Obtener las estadísticas de una sesión de depuración Datalog.
  - Parámetros:
    - `user`: usuario actual.
  - Ejemplo en navegador:

```
$.getJSON(location.protocol+"//" + location.host
+ "/debug_dl_statistics", {user:"test"});
```

- `/debug_dl_cleanup`: Limpiar la última sesión de depuración Datalog.
  - Parámetros:
    - `user`: usuario actual.
  - Ejemplo en navegador:

```
$.getJSON(location.protocol+"//" + location.host
+ "/debug_dl_cleanup", {user:"test"});
```

- `/debug_dl_set_node`: Establecer el estado de un nodo.
  - Parámetros:
    - `user`: usuario actual.
    - `node`: nombre del nodo y su aridad.
    - `state`: estado que se quiere establecer.
  - Ejemplo en navegador:

```
$.getJSON(location.protocol+"//" + location.host
+ "/debug_dl_set_node", {user:"test", node:"r/1",
state:"valid"});
```

4.2.3. Tecnologías utilizadas

Para el desarrollo de la interfaz o fachada de la aplicación se ha utilizado el lenguaje de programación declarativa Prolog, mediante el uso de la técnica HTML Termerized explicada en el siguiente apartado. Prolog (Programming in Logic) se trata de un lenguaje de programación de alto nivel basado en la lógica formal. Se

basa en el paradigma de programación lógica, donde los programas se definen en términos de relaciones lógicas entre hechos y reglas.

También se ha utilizado el lenguaje de hojas de estilo CSS (Cascading Style Sheets) para describir el aspecto y formato del contenido HTML generado a partir del código Prolog. CSS se utiliza para controlar la presentación visual de los documentos web, incluyendo el diseño, los colores, las fuentes y otros aspectos visuales. Permite separar el contenido y la presentación, proporcionando una mayor flexibilidad y mantenibilidad del código.

Por otro lado, se ha utilizado el lenguaje de programación JavaScript para desarrollar las funcionalidades correspondientes al cliente de la aplicación DESweb. JavaScript se trata de un lenguaje de alto nivel y orientado a objetos que se utiliza principalmente en el desarrollo web. Algunas de las ventajas que presenta este lenguaje incluyen una sintaxis fácil de aprender, un tipado dinámico, permite interactuar con el DOM y posee una gran cantidad de bibliotecas para facilitar el desarrollo.

4.2.4. HTML Termerized

El lenguaje de programación declarativo Prolog es capaz de producir contenido HTML basado en los términos estructurados de Prolog, abstrayendo los detalles de más bajo nivel del desarrollo web. Adicionalmente, es bien sabido que la utilización de Prolog para la creación dinámica de contenido produce la reducción del tamaño del programa y permite a los programadores razonar en un nivel lógico de gran abstracción.

DESweb hace uso de la librería HTML proporcionada por SWI-Prolog para la creación de contenido HTML dinámico, haciendo uso de las gramáticas DCG (Definite Clause Grammar), que es una notación declarativa usada para describir y traducir estructuras sintácticas a Prolog, a modo de plantillas. Esta técnica recibe el nombre de *Termerized HTML*. Esto permite mantener gran parte del procesamiento en la parte del servidor.

SWI-Prolog proporciona el predicado `reply_html_page/1` para la generación dinámica de contenido. En el ejemplo de la figura 4.10 se puede observar a la izquierda el manejador Prolog encargado de generar el contenido HTML de la imagen central, mostrado en el navegador tal y como está representado en la parte derecha. Cualquier etiqueta HTML puede ser especificada como un término Prolog, evitando la representación HTML y sustituyendo esta por una especificación Prolog más limpia. Una de las ventajas que esta técnica ofrece es la posibilidad de realizar un cambio en caliente sin tener que reiniciar la totalidad de la aplicación.

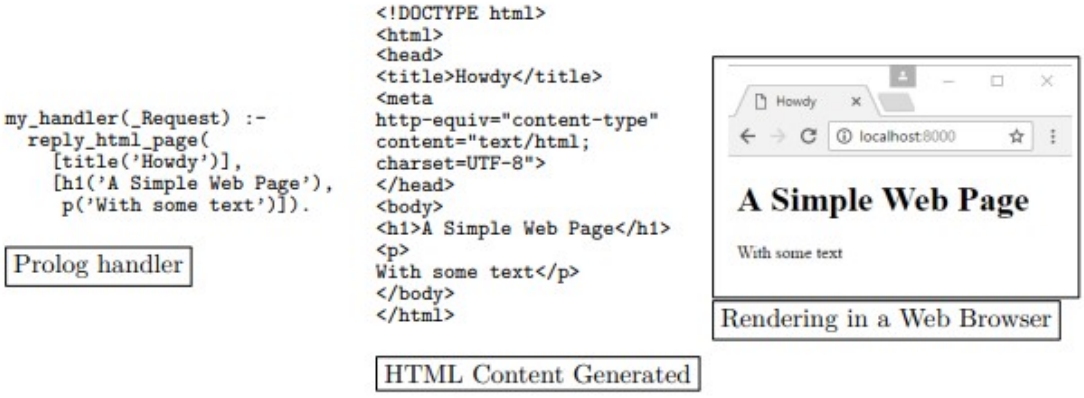

Figura 4.10: Una página web simple generada y proporcionada por SWI-Prolog [2]

4.2.5. Interfaz

Para el desarrollo de la nueva interfaz de depuración de DESweb se ha utilizado como referencia la aplicación ACIDE. Esta última ya cuenta con su propia interfaz de depuración desarrollada por alumnos durante años anteriores. Se han emulado las características principales añadiendo, cuando ha sido posible, el mismo número de botones y utilizando sus respectivos iconos. Todo esto con el objetivo de presentar frente al usuario una interfaz lo más familiar e intuitiva posible.

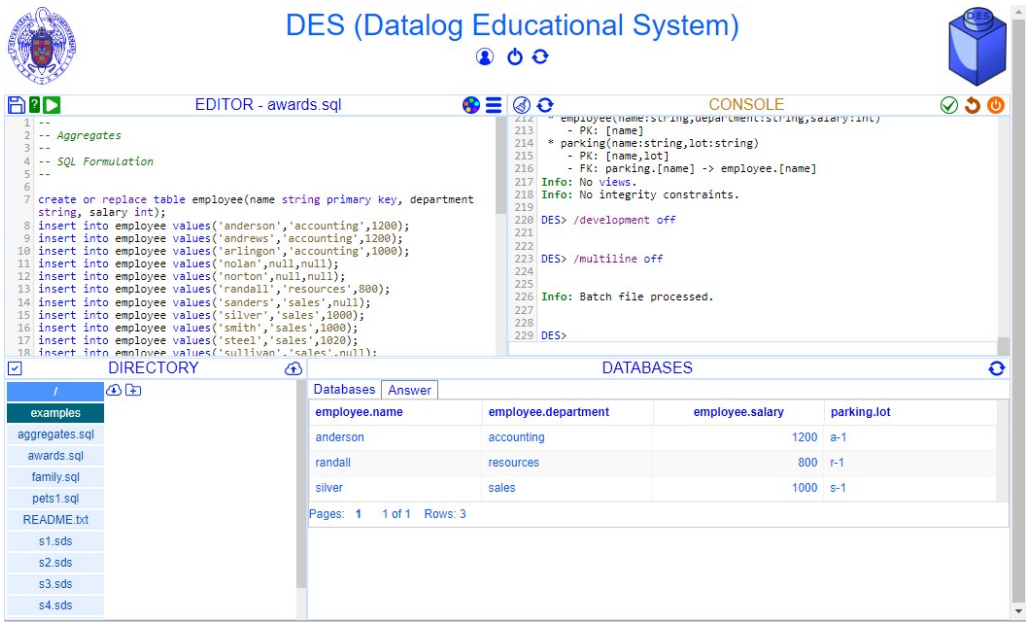

Figura 4.11: Vista una vez iniciada una sesión en DESweb con el panel de depuración oculto

El panel de depuración se mantiene oculto al entrar por primera vez como usuario

de DESweb (Figura 4.11). Podemos activar el modo depuración a través del menú desplegable situado en el editor de la aplicación. Una vez activado, el panel de depuración se mostrará en la parte inferior derecha de la aplicación (figura 4.12). El usuario será capaz de modificar las dimensiones del panel según crea conveniente.

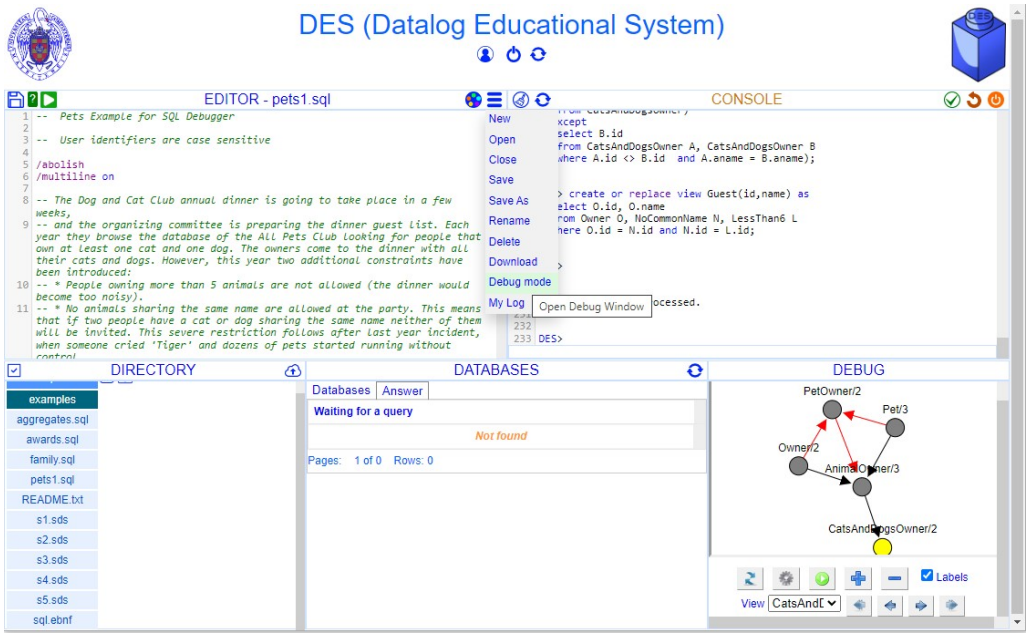

Figura 4.12: Vista una vez iniciada una sesión en DESweb con el panel de depuración

El panel de depuración está formado por dos secciones principales:

- Un grafo de depuración, el cual se encuentra situado en la parte superior del panel.

- Una barra de herramientas, la cual se encuentra situada en la parte inferior del panel.

El grafo de depuración se encarga de representar el estado de las distintas vistas, tablas o predicados a lo largo de la sesión de depuración. Cada vista, tabla o predicado está representado por un nodo en el grafo, el cual se puede identificar gracias a la etiqueta situada en la parte superior de este. A su vez, los nodos se encuentran conectados a través de arcos que representan las dependencias entre cada uno de ellos. Estas dependencias pueden ser de dos tipos: negativas o positivas, representadas mediante el color negro o rojo respectivamente. En el extremo de cada arco se puede observar una flecha encargada de indicar la dirección de cada relación.

Figura 4.13: De izquierda a derecha y de arriba hacia abajo: *Actualizar, Configuración inicial, Comenzar depuración, Ampliar zoom, Alejar zoom, Etiquetas, Vista seleccionada, Primer nodo, Nodo anterior, Siguiente nodo y Último nodo*

En la barra de herramientas se encuentran las múltiples funcionalidades disponibles a la hora de empezar una sesión de depuración (figura 4.13).

- Actualizar: botón encargado de actualizar la información mostrada en el grafo de depuración.
- Configuración inicial: botón encargado de permitir al usuario modificar la configuración inicial que se utilizará al comenzar la depuración (figura 4.14). Cuando el usuario presiona este botón se le muestra un diálogo con todas las posibles opciones que se pueden configurar en el depurador. Entre las opciones SQL se encuentran las siguientes:
  - Opción de confiar o no en las tablas de la base de datos.
  - Tipo de depuración que se quiere seguir. Los posibles valores son Completa o Básica.
  - Tipo de orden que se quiere seguir a la hora de recorrer el grafo de depuración. Los posibles valores son Cardinal o de arriba a abajo (Top to Bottom).
  - Opción de confiar en las vistas contenidas en el archivo seleccionado.

  Entre las opciones Datalog se encuentran las siguientes:
  - Opción de confiar o no en los predicados extensionales.
  - Opción de confiar en los predicados contenidos en el archivo seleccionado.
- Comenzar depuración: botón encargado de dar comienzo a la sesión de depuración. Permanece deshabilitado mientras no haya una vista seleccionada.
- Ampliar zoom: botón encargado de acercar la vista del grafo de depuración.
- Alejar zoom: botón encargado de alejar la vista del grafo de depuración.
- Etiquetas: botón encargado de permitir mostrar o no las etiquetas de los nodos en el grafo de depuración.
- Nodo seleccionado: selector encargado de indicar la vista o predicado objeto de la sesión de depuración. Permanece vacío mientras no haya vistas/predicados disponibles.

- Primer nodo: botón encargado de mostrar en el grafo de depuración el primer nodo del recorrido.
- Nodo anterior: botón encargado de mostrar en el grafo de depuración el anterior nodo del recorrido.
- Siguiente nodo: botón encargado de mostrar en el grafo de depuración el siguiente nodo del recorrido.
- Último nodo: botón encargado de mostrar en el grafo de depuración el último nodo del recorrido.

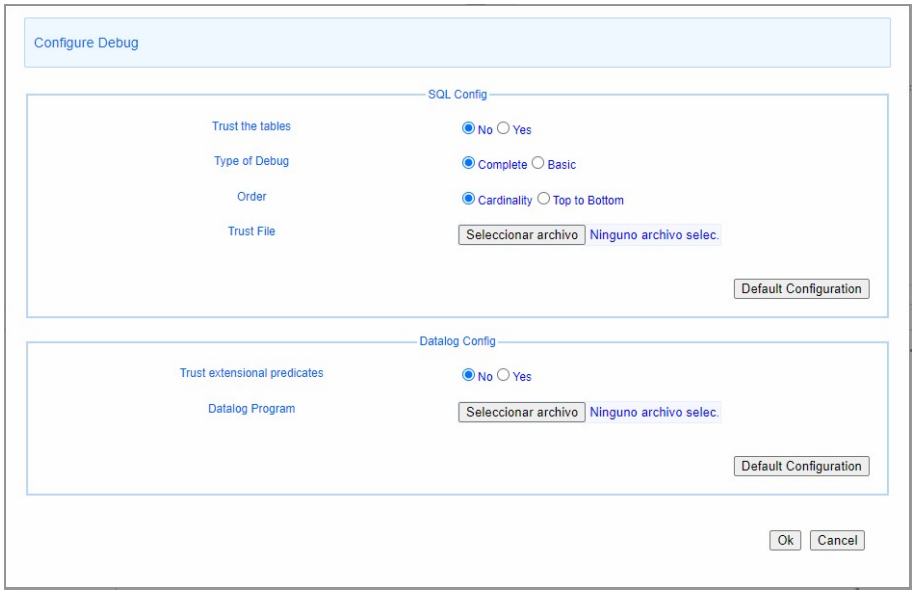

Figura 4.14: Panel para elegir la configuración inicial de la sesión de depuración

Cabe destacar la fusión de los dos paneles de depuración SQL y Datalog en uno solo frente al diseño original de ACIDE que presentaba estos paneles en pestañas separadas. Este cambio se ha realizado para hacer más clara la depuración de cualquier base de datos al usuario, no teniendo este que cambiar de pestañas según el tipo de base de datos que se encuentre manejando. También se ha eliminado la posibilidad de realizar *zoom* indicando un valor numérico y, aumentando o disminuyendo este por unidades. Esta funcionalidad ha sido sustituida por la posibilidad de hacer *zoom* para empequeñecer o agrandar el tamaño el grafo de depuración con la rueda del ratón. Esto se puede observar comparando las vista de los dos paneles (figura **??** y figura 4.15).

Una vez cargado el contexto de la base de datos y seleccionada la vista que se desea depurar aparecerá en el panel el grafo de depuración correspondiente. En él se pueden observar los nodos y sus relaciones donde se representan a través de colores los distintos estados en los que se encuentran (figura 4.15).

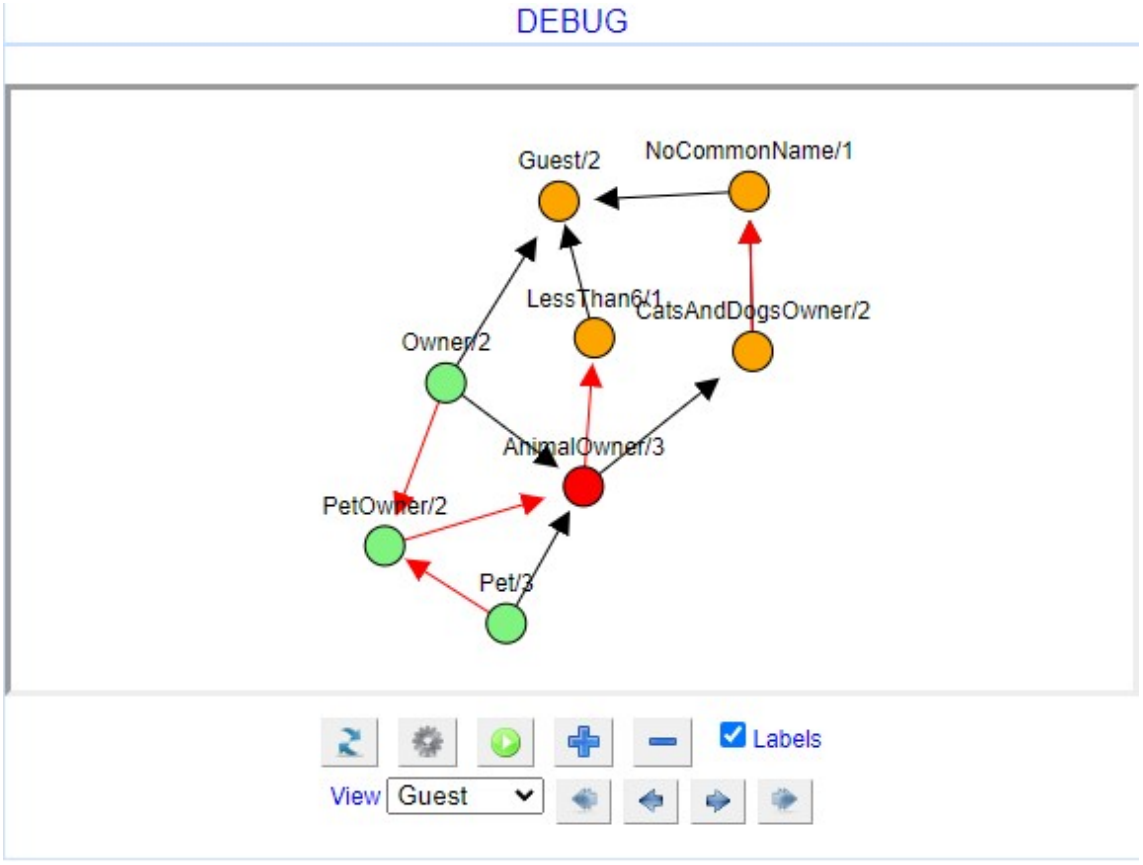

Figura 4.15: Estado del grafo después de terminar una sesión de depuración

Por un lado, los nodos pueden aparecer coloreados por uno de los siguientes colores, indicando el correspondiente estado del mismo:

- Verde : indica que este nodo se trata de una vista, tabla o predicado válida/o.
- Rojo: indica que este nodo se trata de la vista, tabla o predicado origen del error de la sesión de depuración.
- Amarillo: indica que este nodo se trata del nodo actual del recorrido en el grafo.
- Naranja: indica que este nodo se trata de una vista, tabla o predicado evaluada/o como no válida por el usuario.
- Gris: indica que este nodo se encuentra sin explorar y, por tanto, con estado desconocido.

Por otro lado, como ya se ha comentado anteriormente, los arcos encargados de representar las dependencias entre los distintos nodos del grafo pueden ser de dos colores, negro o rojo, indicando respectivamente si se trata de una relación positiva o negativa.

Para dar comienzo a una sesión de depuración se debe tener seleccionada la vista en el caso de una base de datos SQL, o el predicado en el caso de Datalog, el cual el usuario ha detectado como nodo erróneo. A continuación se debe pulsar en el botón correspondiente a la funcionalidad de comienzo de la depuración. Una vez pulsado, aparecerá una ventana de depuración de aspecto similar a la utilizada en la versión de ACIDE. En dicha ventana se expondrá al usuario la pregunta correspondiente que formula el depurador DES para detectar el origen del error (figura 4.16). También se mostrará información relevante de la vista que se está depurando.

Entre la información que se muestra se pueden encontrar el nombre de la vista o predicado actual en la depuración, el número de tuplas que contiene así como el contenido de esta misma vista/predicado en forma de tabla. En la parte inferior del diálogo estarán localizados los diferentes botones encargados de proporcionar al depurador DES la respuesta del usuario a la pregunta formulada. Entre estas posibles opciones encontramos las siguientes:

- Valid: marca la vista o predicado actual con el estado válido. Esta respuesta solo está disponible una vez el usuario ha respondido a la primera pregunta de la depuración, ya que el primer nodo de la sesión de depuración se trata del nodo que el usuario ha detectado como no válido.

- Non Valid: marca el nodo actual con estado no válido.

- Missing Tuple: responde al depurador marcando el nodo como no válido debido a una tupla faltante. Esta tupla se debe indicar introduciendo sus valores a través de la última fila de la tabla del diálogo. Si se intenta elegir esta respuesta sin haber proporcionado la información necesaria la aplicación mostrará una alerta e impedirá continuar con el flujo de depuración.

- Wrong Tuple: responde al depurador marcando el nodo como no válido debido a una tupla errónea. Esta tupla se debe indicar seleccionando una tupla existente en la tabla del diálogo. Si se intenta elegir esta respuesta sin haber proporcionado la información necesaria la aplicación mostrará una alerta e impedirá continuar con el flujo de depuración.

- Cancel: esta respuesta permite al usuario cancelar la actual sesión de depuración cerrando con ello el diálogo.

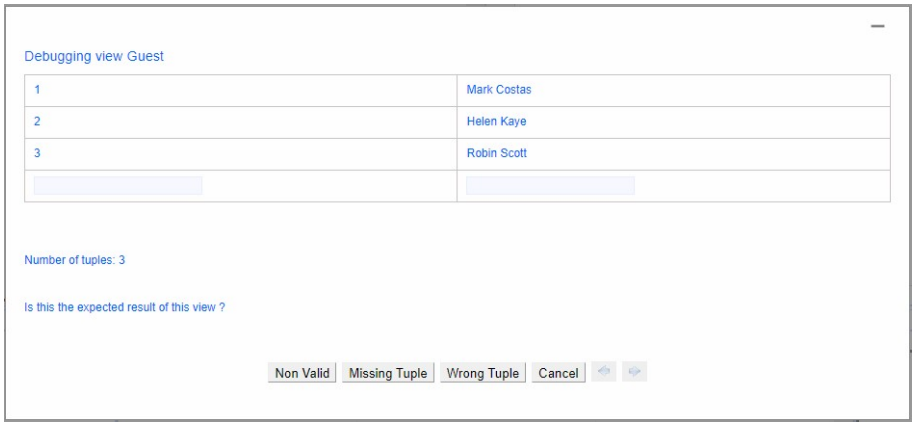

Figura 4.16: Ventana de depuración al comienzo de la sesión de depuración

A medida que el usuario avance en la sesión el depurador se le irán realizando las preguntas que el depurador crea necesarias para localizar el error en la base de datos. De manera simultánea, el grafo de depuración actualizará los estados de sus nodos para proporcionar al usuario la información del estado actual de la sesión.

Una vez respondida la primera pregunta, en la venta de depuración se añadirá la posibilidad de marcar el nodo con el estado válido (figura 5.39). También se activarán las opciones situadas en la parte inferior derecha para permitir al usuario volver hacia atrás si cree que hay alguna pregunta que ha respondido de manera errónea o desea consultar cuestiones anteriores. De igual manera, se permite al usuario avanzar en el flujo de depuración deshaciendo los cambios a la hora de retroceder en este, volviendo al estado inicial de la sesión de depuración.

Figura 4.17: Ventana de depuración con las opciones para establecer el estado del nodo a válido así como la posibilidad de deshacer o rehacer el flujo de depuración

Para responder al depurador estableciendo el estado del nodo actual en el flujo a no válido utilizando la opción de tupla faltante será necesario añadir la información de esta tupla en la respuesta (figura 4.18). Para ello, se debe hacer uso de la fila

introducida en la tabla en la parte inferior de esta. En este lugar el usuario deberá especificar todos los datos referentes a las distintas columnas de la vista, tabla o predicado actual. Si alguno de estos campos quedará vacío, se informa al usuario de este problema a través de una alerta, evitando enviar la respuesta al sistema hasta introducir la información necesaria.

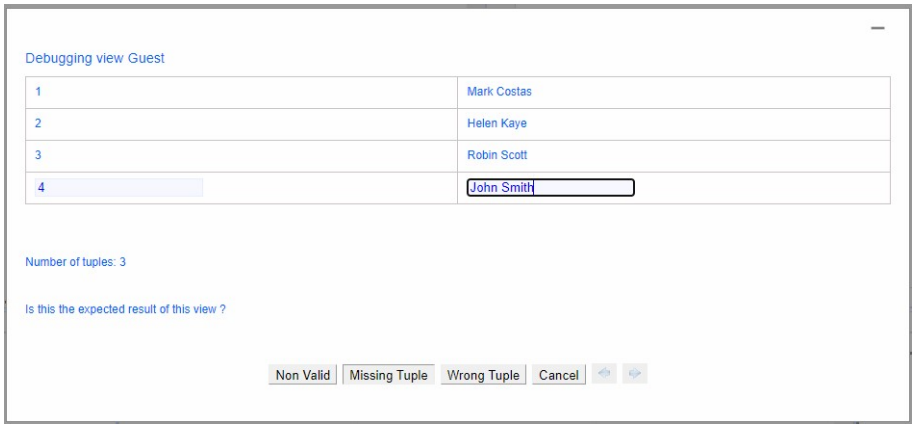

Figura 4.18: Ventana de depuración haciendo uso de la respuesta de nodo no válido con tupla faltante

Por otro lado, la opción de tupla errónea tendrá un funcionamiento similar. El usuario deberá seleccionar una de las filas correspondiente a la tupla que considera como errónea en la tabla de la ventana de depuración (figura 4.19). Esto se debe llevar a cabo antes de pulsar el botón correspondiente a la opción de tupla errónea. De no ser así, al no haber una tupla seleccionada e intentar enviar una respuesta de este tipo, se mostrará una alerta informado del problema, evitando que el usuario envía una respuesta de este tipo con información necesaria ausente.

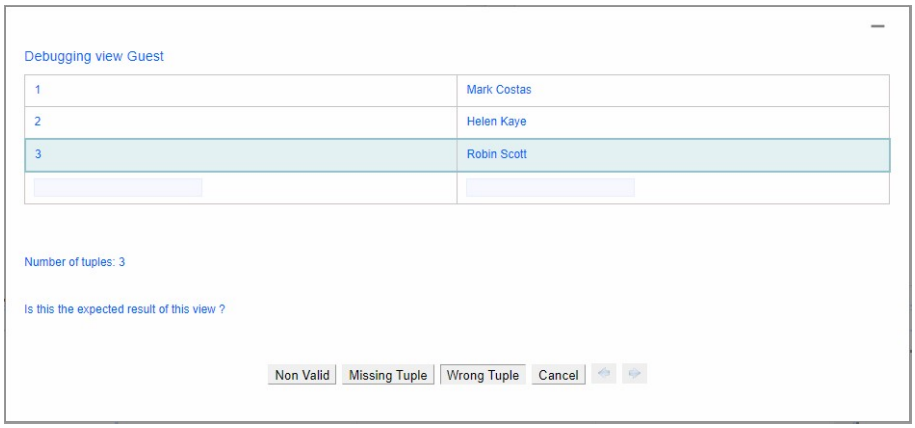

Figura 4.19: Ventana de depuración haciendo uso de la respuesta de nodo no válido con tupla errónea

Por último, al terminar la sesión de depuración una vez que el sistema ha recibido

la información suficiente como para encontrar el nodo erróneo, se mostrará en la aplicación una última ventana exponiendo el nombre de la vista, tabla o predicado (figura 4.20).

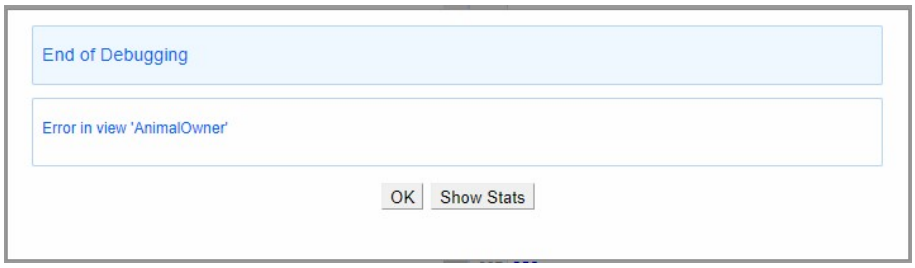

Figura 4.20: Ventana de depuración una vez terminada la sesión de depuración

En este momento, se le presentarán al usuario la posibilidad de mostrar información acerca de la sesión que se ha llevado a cabo (figura 4.23). Entre esta información se encuentra la siguiente:

- Número de nodos
- Máximo número de preguntas posibles
- Número de preguntas realizadas
- Número de tuplas inspeccionadas
- Número de tuplas raíz
- Número de tuplas no-raíz

Figura 4.21: Ventana de depuración mostrando las estadísticas de la sesión de depuración

Cabe destacar que en, cualquier momento del flujo de depuración, el usuario tiene la posibilidad de mover la ventana de depuración a su antojo haciendo clic con el ratón en ella y arrastrando, así como ocultar por completo esta sin detener la depuración utilizando el botón correspondiente situado en la parte superior derecha del modal. Esto permite consultar información relevante a la depuración sin detener esta por completo. Al ocultar la ventana aparecerá una pestaña en el panel de depuración que permitirá al usuario abrirla de nuevo una vez haya consultado la información necesaria.

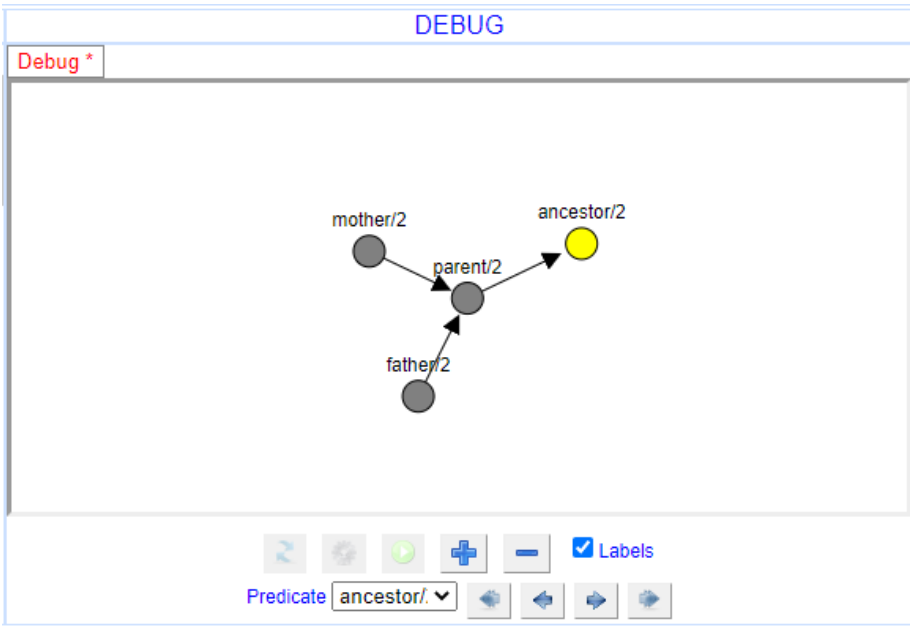

Figura 4.22: Pestaña creada al ocultar la ventana de depuración

Asimismo, también es posible contestar a la pregunta que el depurador hace al usuario, mostrar las tuplas de una vista o predicado y avanzar o retroceder en la depuración mediante un menú contextual que se muestra al hacer clic derecho sobre alguno de los nodos del grafo.

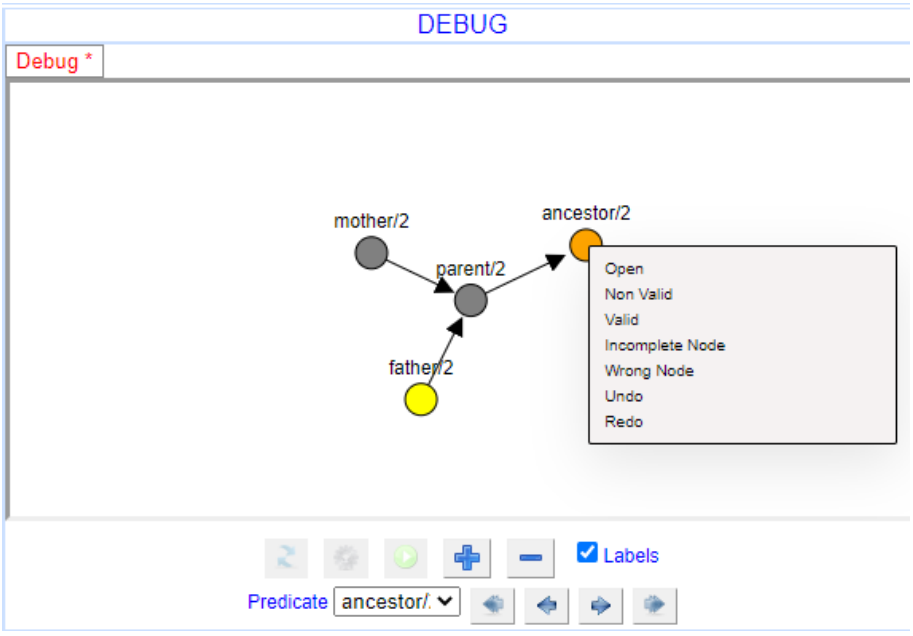

Figura 4.23: Menú contextual de los nodos

Capítulo 5

Ejemplos de depuración

En esta sección vamos a mostrar un ejemplo completo de depuración paso a paso, explicando los diferentes servicios utilizados y sus parámetros, así como las respuestas obtenidas.

Para cada llamada se ha utilizado el usuario de pruebas `david1` en el entorno de pruebas alojado en `localhost:9000`.

En estos ejemplos se puede ver el protocolo de comunicación cliente-servidor.

5.1. Ejemplo de depuración Datalog

Comenzamos tomando como ejemplo el programa family.dl:

```
father(tom,amy).
father(jack,fred).
father(tony,carolII).
father(fred,carolIII).

mother(grace,amy).
mother(amy,fred).
mother(carolI,carolII).
mother(carolII,carolIII).

parent(X,Y) :- father(X,Y); mother(X,Y).

ancestor(X,Y) :- parent(X,Y).
ancestor(X,Y) :- parent(X,Z), ancestor(Z,Y).
```

Lo primero que haremos será obtener la información de los nodos y sus respectivas relaciones (comando `/pdg` por consola):

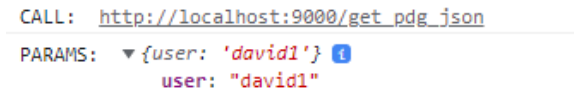

Figura 5.1: Llamada al servicio para obtener el grafo de dependencias

Esto nos devuelve la siguiente respuesta:


```
{success: 'true', message: 'success', nodes: Array(4), arcs: Array(4)}
  arcs: Array(4)
    0:
      arc: Array(3)
        0: {kind: 'pos'}
        1: {from: 'ancestor/2'}
        2: {to: 'ancestor/2'}
        length: 3
        [[Prototype]]: Array(0)
      [[Prototype]]: Object
    1:
      arc: Array(3)
        0: {kind: 'pos'}
        1: {from: 'parent/2'}
        2: {to: 'ancestor/2'}
        length: 3
        [[Prototype]]: Array(0)
      [[Prototype]]: Object
    2:
      arc: Array(3)
        0: {kind: 'pos'}
        1: {from: 'father/2'}
        2: {to: 'parent/2'}
        length: 3
        [[Prototype]]: Array(0)
      [[Prototype]]: Object
    3:
      arc: Array(3)
        0: {kind: 'pos'}
        1: {from: 'mother/2'}
        2: {to: 'parent/2'}
        length: 3
        [[Prototype]]: Array(0)
      [[Prototype]]: Object
    length: 4
    [[Prototype]]: Array(0)
  message: "success"
  nodes: Array(4)
    0: {node: 'ancestor/2'}
    1: {node: 'father/2'}
    2: {node: 'mother/2'}
    3: {node: 'parent/2'}
    length: 4
    [[Prototype]]: Array(0)
  success: "true"
```


Figura 5.2: Respuesta con el grafo de dependencias

Una vez obtenida esta información nos será posible dibujar el grafo de dependencias.

Vamos ahora a comenzar la sesión de depuración. Lo primero que hacemos es seleccionar el nodo inicial y hacer clic en el botón de inicio de depuración. Para este ejemplo seleccionaremos el nodo `ancestor/2`.

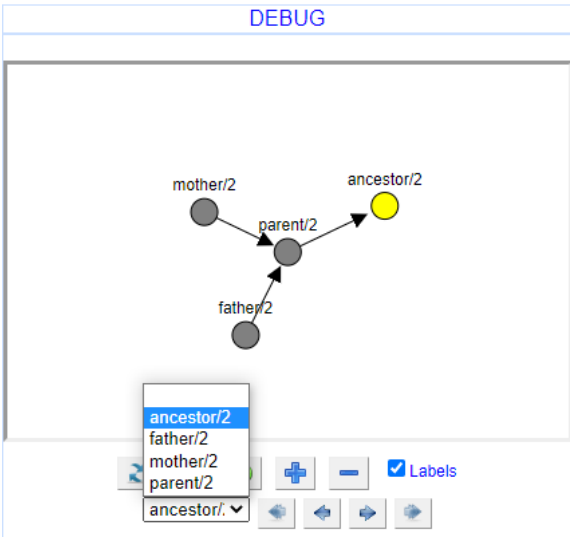

Figura 5.3: Selección de predicado inicial

Esto abrirá la ventana de depuración, donde se mostrarán las tuplas del nodo seleccionado.

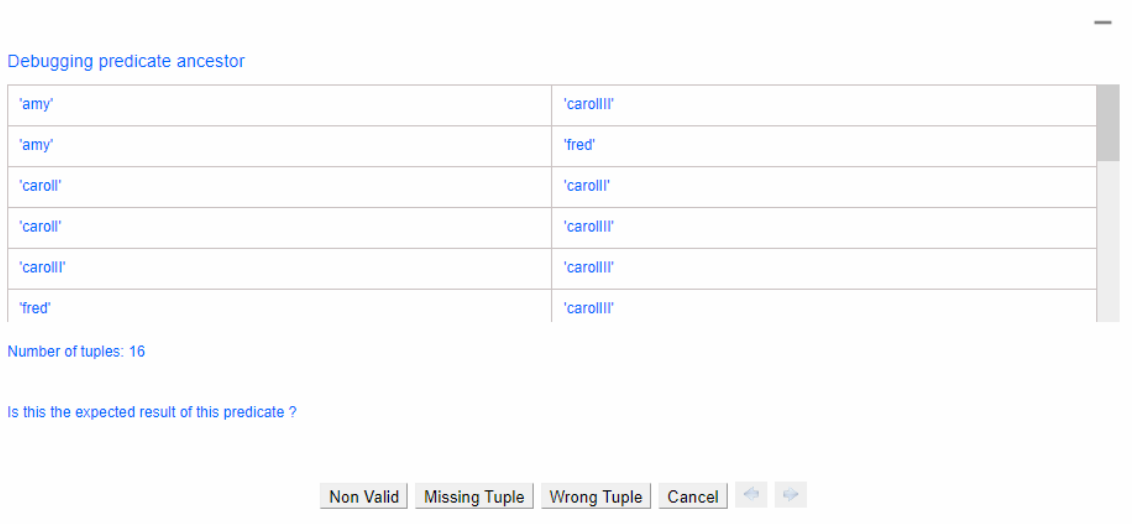

Figura 5.4: Tuplas del predicado inicial

Para obtener estas tuplas utilizamos el siguiente servicio (comando `ancestor(X,Y)` por consola):

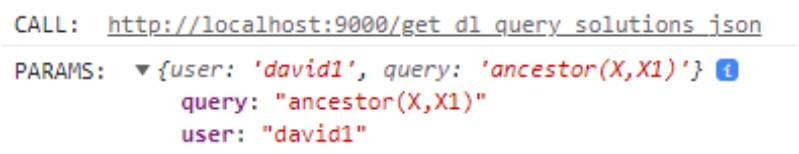

Figura 5.5: Llamada al servicio para obtener las tuplas de `ancestor/2`

El sistema nos devuelve una lista con las tuplas correspondientes al predicado enviado como parámetro:

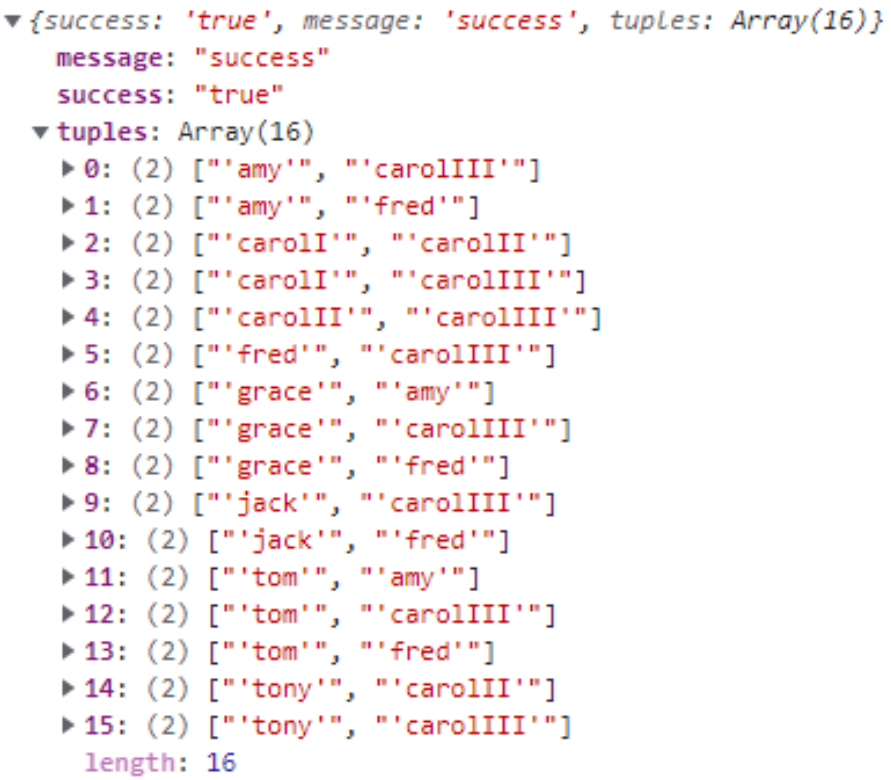

Figura 5.6: Respuesta con las tuplas del predicado `ancestor/2`

A continuación podemos indicar el estado de las tuplas. En el momento en el que se seleccione una de ellas comenzará la sesión de depuración. Para iniciar la sesión se hará uso del servicio `/debug_dl`.

A continuación se detallan las posibles llamadas:

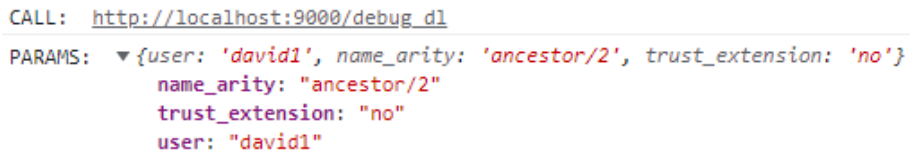

Figura 5.7: Envío de respuesta "Non Valid"

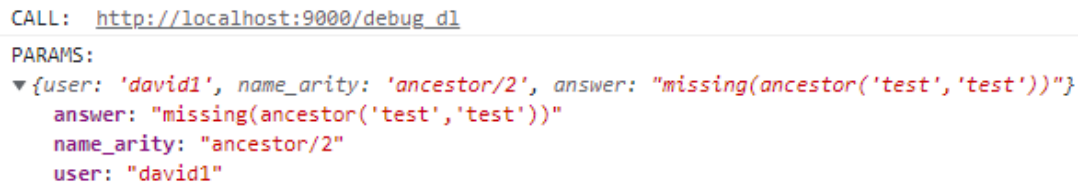

Figura 5.8: Envío de respuesta "Missing Tuple"

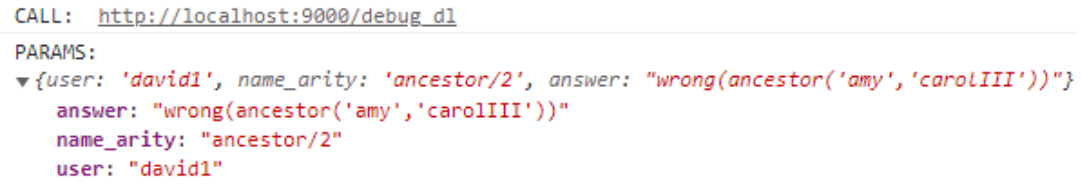

Figura 5.9: Envío de respuesta "Wrong Tuple"

El sistema nos devolverá una respuesta indicando si la sesión de depuración ha sido iniciada con éxito y, en caso afirmativo, una lista con los estados conocidos de los nodos:

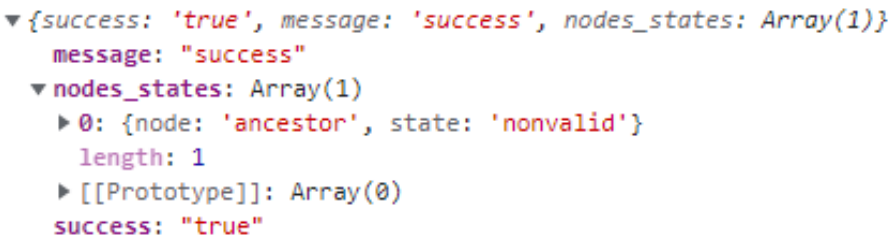

Figura 5.10: Respuesta del sistema al iniciar una sesión de depuración

Ahora debemos obtener la siguiente pregunta que nos hace el sistema. Para ello debemos realizar la siguiente llamada (comando `/debug_dl_current_question` por consola):

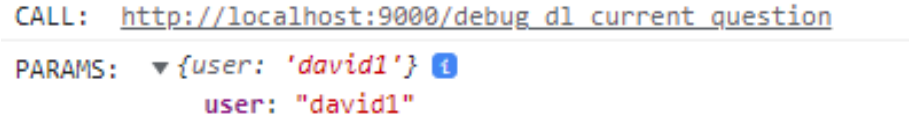

Figura 5.11: Llamada para obtener la pregunta actual

El sistema nos devolverá la pregunta de la siguiente manera:

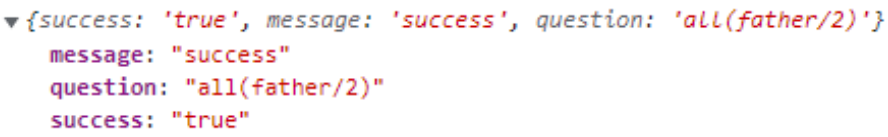

Figura 5.12: Respuesta del sistema con la pregunta actual

Vemos que la pregunta ahora es `all(father/2)`. Por cada pregunta que nos haga el sistema deberemos hacer la llamada `/debug_get_dl_query_solutions_json` explicada anteriormente para obtener las tuplas del predicado sobre el que se está preguntando y mostrarlas por pantalla.

Figura 5.13: Tuplas del predicado `father/2`

Una vez iniciada la depuración, el resto de respuestas utilizarán el servicio `/debug_dl_answer`:

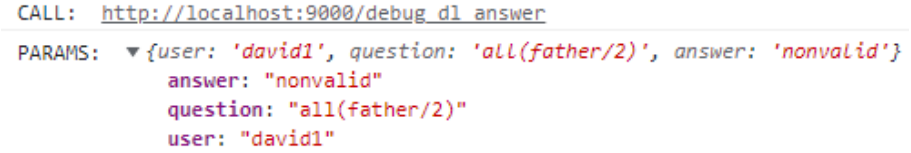

Figura 5.14: Respuesta a la pregunta `all(father/2)`

El usuario continuará respondiendo las preguntas del sistema hasta que este indique que uno de los nodos es erróneo. En este caso la respuesta a la llamada anterior devuelve la siguiente información:

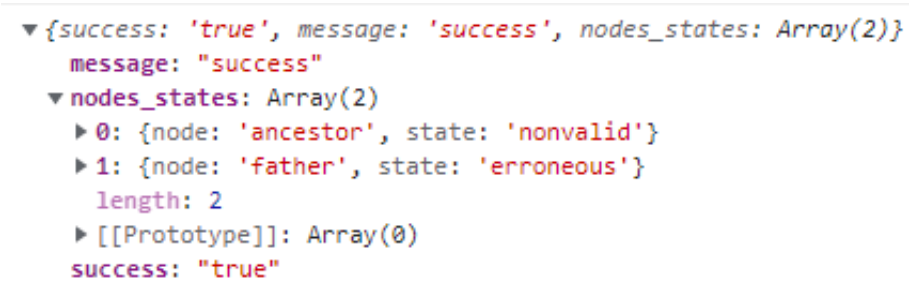

Figura 5.15: Respuesta del sistema indicando un nodo erróneo

Como se puede observar en la respuesta anterior se nos indica que el nodo `father/2` es erróneo. Cuando se recibe esta respuesta la ventana de depuración se cierra y en su lugar se muestra un modal indicando el fin de la depuración y el nodo problemático.

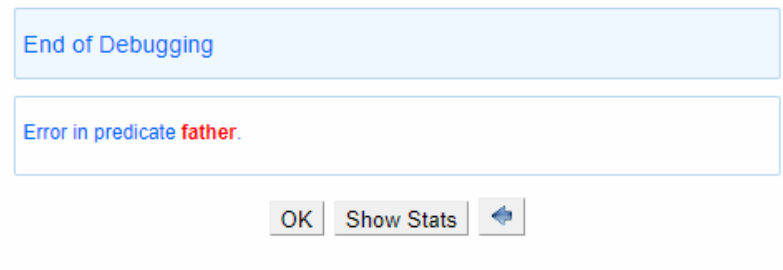

Figura 5.16: Fin de la depuración

También es posible ver información adicional de la sesión de depuración haciendo clic en el botón "Show Stats".

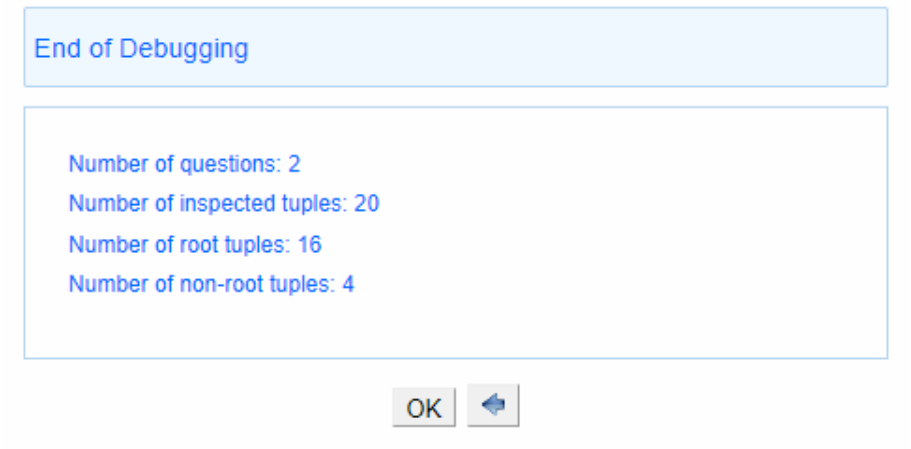

Figura 5.17: Estadísticas de la depuración

5.2. Ejemplo de depuración SQL

Comenzamos tomando como ejemplo el programa awards1.sql:

```
/* Tables */
DROP TABLE IF EXISTS registration;
DROP TABLE IF EXISTS courses;
DROP TABLE IF EXISTS allInOneCourse;

CREATE TABLE allInOneCourse (
  student varchar(45) NOT NULL,
  pass integer,
  PRIMARY KEY (student)
);

CREATE TABLE courses (
  id varchar(11) NOT NULL,
  level int(11) DEFAULT NULL,
  PRIMARY KEY (id)
);

CREATE TABLE registration (
  student varchar(45) NOT NULL,
  course varchar(11) NOT NULL,
  pass integer,
  FOREIGN KEY(course) references courses(id),
  PRIMARY KEY (student,course)
);

INSERT INTO allInOneCourse(student,pass)  VALUES ('Adrian',1);
INSERT INTO allInOneCourse(student,pass)  VALUES ('Alba',1);
INSERT INTO allInOneCourse(student,pass)  VALUES ('Alisha',0);
INSERT INTO allInOneCourse(student,pass)  VALUES ('Amaya',1);
INSERT INTO allInOneCourse(student,pass)  VALUES ('Arabella',1);
INSERT INTO allInOneCourse(student,pass)  VALUES ('Ava',0);
INSERT INTO allInOneCourse(student,pass)  VALUES ('Dexter',0);
INSERT INTO allInOneCourse(student,pass)  VALUES ('Emma',1);
INSERT INTO allInOneCourse(student,pass)  VALUES ('Evelyn',1);
INSERT INTO allInOneCourse(student,pass)  VALUES ('Gabriel',0);
INSERT INTO allInOneCourse(student,pass)  VALUES ('Gavin',0);
INSERT INTO allInOneCourse(student,pass)  VALUES ('George',1);
INSERT INTO allInOneCourse(student,pass)  VALUES ('Harper',1);
INSERT INTO allInOneCourse(student,pass)  VALUES ('Henry',1);
INSERT INTO allInOneCourse(student,pass)  VALUES ('James',1);
INSERT INTO allInOneCourse(student,pass)  VALUES ('Jennifer',1);
```

```
INSERT INTO allInOneCourse(student,pass)  VALUES ('Maggy',0);
INSERT INTO allInOneCourse(student,pass)  VALUES ('Miguel',1);
INSERT INTO allInOneCourse(student,pass)  VALUES ('Nicolas',1);
INSERT INTO allInOneCourse(student,pass)  VALUES ('Noah',1);
INSERT INTO allInOneCourse(student,pass)  VALUES ('Olivia',1);
INSERT INTO allInOneCourse(student,pass)  VALUES ('Owen',1);
INSERT INTO allInOneCourse(student,pass)  VALUES ('Paul',1);
INSERT INTO allInOneCourse(student,pass)  VALUES ('Rocco',0);
INSERT INTO allInOneCourse(student,pass)  VALUES ('Shane',0);
INSERT INTO allInOneCourse(student,pass)  VALUES ('Stella',1);
INSERT INTO allInOneCourse(student,pass)  VALUES ('Tosca',1);
INSERT INTO allInOneCourse(student,pass)  VALUES ('Violet',1);

INSERT INTO courses(id,level)  VALUES ('c1',  1);
INSERT INTO courses(id,level)  VALUES ('c2',  2);
INSERT INTO courses(id,level)  VALUES ('c3',  3);
INSERT INTO courses(id,level)  VALUES ('c4',  4);
INSERT INTO courses(id,level)  VALUES ('c5',  5);
INSERT INTO courses(id,level)  VALUES ('c0',  0);

INSERT INTO registration(student,course,pass) VALUES  ('Alba','c1',0);
INSERT INTO registration(student,course,pass) VALUES  ('Alba','c2',1);
INSERT INTO registration(student,course,pass) VALUES  ('Anna','c0',1);
INSERT INTO registration(student,course,pass) VALUES  ('Anna','c1',1);
INSERT INTO registration(student,course,pass) VALUES  ('Anna','c2',1);
INSERT INTO registration(student,course,pass) VALUES  ('Anna','c3',0);
INSERT INTO registration(student,course,pass) VALUES  ('Carla','c0',1);
INSERT INTO registration(student,course,pass) VALUES  ('James','c0',1);
INSERT INTO registration(student,course,pass) VALUES  ('James','c1',1);
INSERT INTO registration(student,course,pass) VALUES  ('James','c2',1);
INSERT INTO registration(student,course,pass) VALUES  ('James','c3',1);
INSERT INTO registration(student,course,pass) VALUES  ('Juan','c1',1);
INSERT INTO registration(student,course,pass) VALUES  ('Juan','c2',0);
INSERT INTO registration(student,course,pass) VALUES  ('Juan','c5',1);
INSERT INTO registration(student,course,pass) VALUES  ('Mica','c1',1);
INSERT INTO registration(student,course,pass) VALUES  ('Mica','c2',0);
INSERT INTO registration(student,course,pass) VALUES  ('Miguel','c2',1);
INSERT INTO registration(student,course,pass) VALUES  ('Paul','c0',1);
INSERT INTO registration(student,course,pass) VALUES  ('Pedro','c2',1);

/* Views */

/* Students, level of their courses and  pass/fail flag */
create or replace view standard(student, level, pass) as
   select R.student, C.level, R.pass
   from courses C, registration R
```

```
   where C.id = R.course;

/* Students from the basic course (level = 0) */
create or replace view basic(student) as
   select S.student
   from standard S
   where S.level = 0 and S.pass=1;

/* Intensive students */
create or replace view intensive(student) as
   (select A.student
    from allInOneCourse A  where A.pass=1)
   union
   (select A1.student
    from standard A1, standard A2, standard A3
    where A1.student = A2.student and A2.student = A3.student
          and
          A1.level = 1 and A2.level = 2 and A3.level = 3);

/* Awards are only for students from the basic level */
create or replace view awards(student) as
  select student from  basic
  except
  select student from  intensive;
```

Lo primero que haremos será obtener la información de los nodos y sus respectivas relaciones (comando `/rdg` por consola):

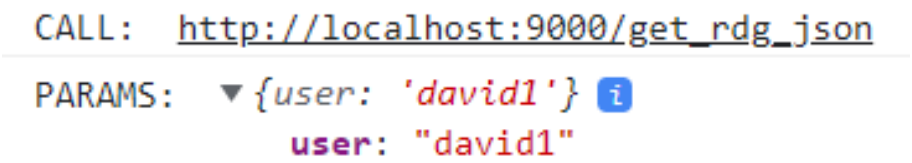

Figura 5.18: Llamada al servicio para obtener el grafo de dependencias

Esto nos devuelve la siguiente respuesta:

Figura 5.19: Respuesta con el grafo de dependencias parte 1

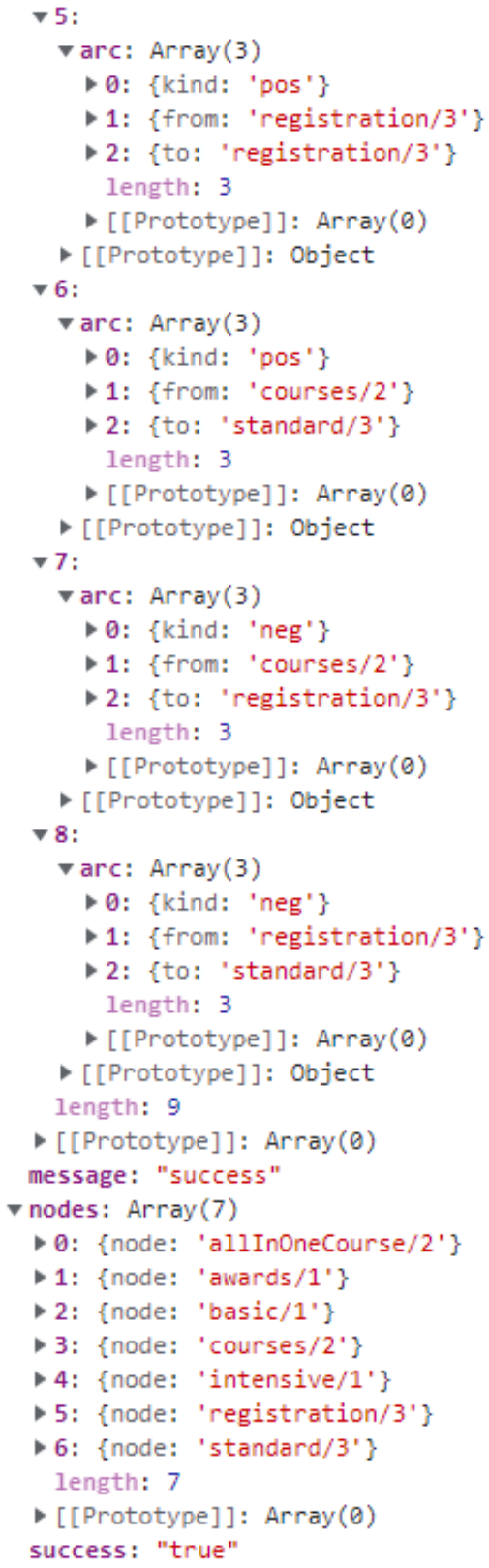

Figura 5.20: Respuesta con el grafo de dependencias parte 2

Una vez obtenida esta información nos será posible dibujar el grafo de dependencias.

También tendremos que preguntar al sistema cuáles de los nodos corresponden a tablas. Para ello usamos la siguiente llamada (comando `/list_tables` por consola):

Figura 5.21: Llamada para obtener las tablas

La respuesta a esta llamada será de la forma:

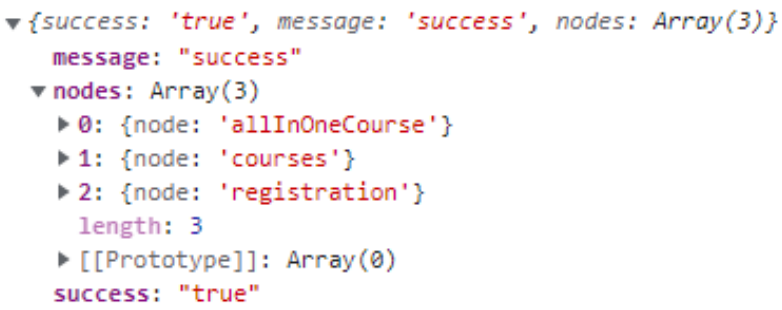

Figura 5.22: Respuesta con las correspondientes tablas

Vamos ahora a comenzar la sesión de depuración. Lo primero que hacemos es seleccionar el nodo inicial y hacer clic en el botón de inicio de depuración. Para este ejemplo seleccionaremos el nodo `basic`.

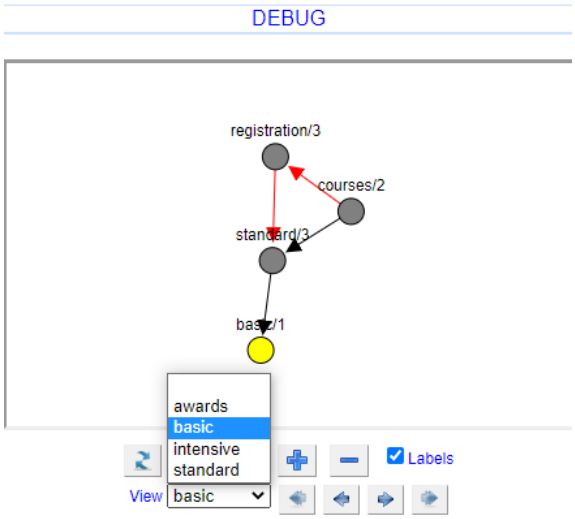

Figura 5.23: Selección de la vista inicial

Esto abrirá la ventana de depuración, donde se mostrarán las tuplas del nodo seleccionado.

Figura 5.24: Tuplas de la vista inicial

Para obtener dichas tuplas primero deberemos obtener información de la base de datos a consultar (comando `/current_db` por consola).

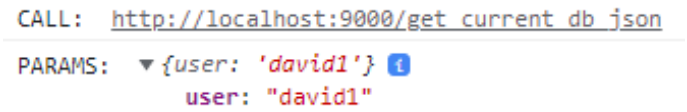

Figura 5.25: Llamada al servicio para obtener la base de datos

La respuesta obtenida será:

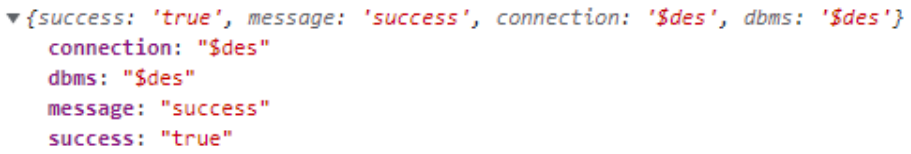

Figura 5.26: Respuesta indicando la base de datos

Esta información la usaremos en las siguientes llamadas. Para obtener las tuplas del nodo inicial utilizamos el siguiente servicio (comando `select * from basic` por consola):

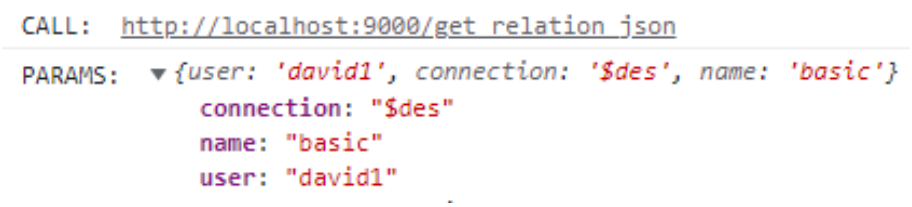

Figura 5.27: Llamada al servicio para obtener las tuplas de `basic`

El sistema nos devuelve una lista con las tuplas correspondientes a la vista o tabla enviada como parámetro:

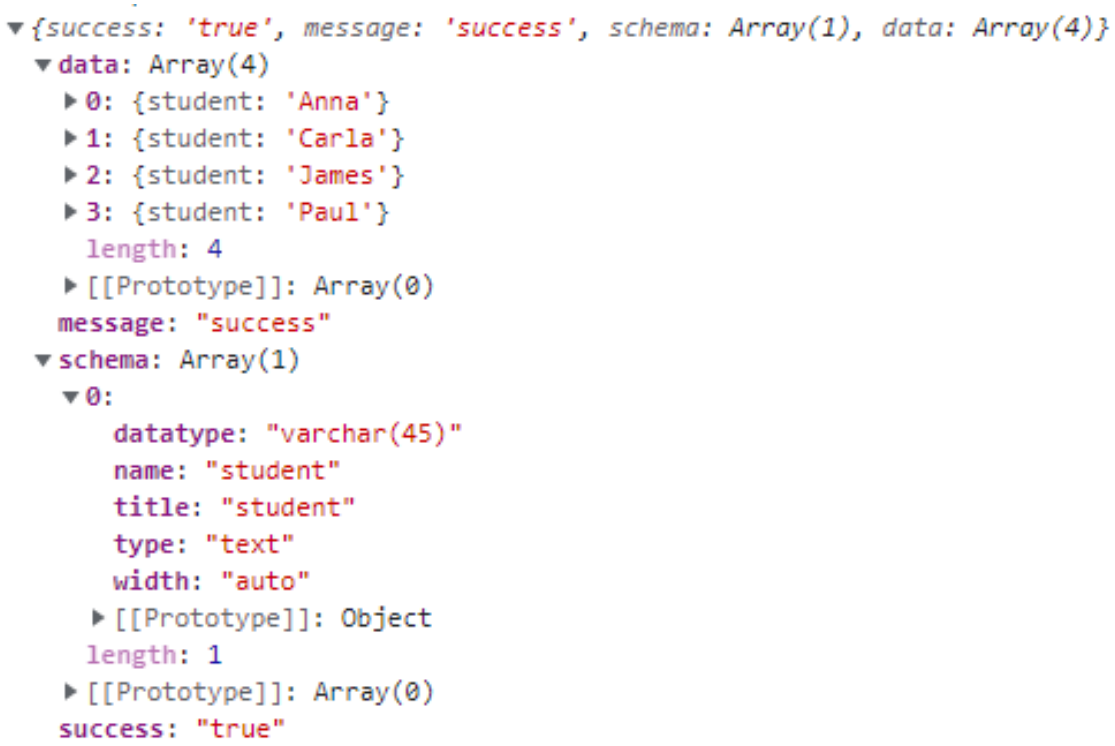

Figura 5.28: Respuesta con las tuplas de la vista `basic`

A continuación podemos indicar el estado de las tuplas. En el momento en el que se seleccione una de ellas comenzará la sesión de depuración. Para iniciar la sesión se hará uso del servicio `/debug_sql`.

A continuación se detallan las posibles llamadas:

Figura 5.29: Envío de respuesta "Non Valid"

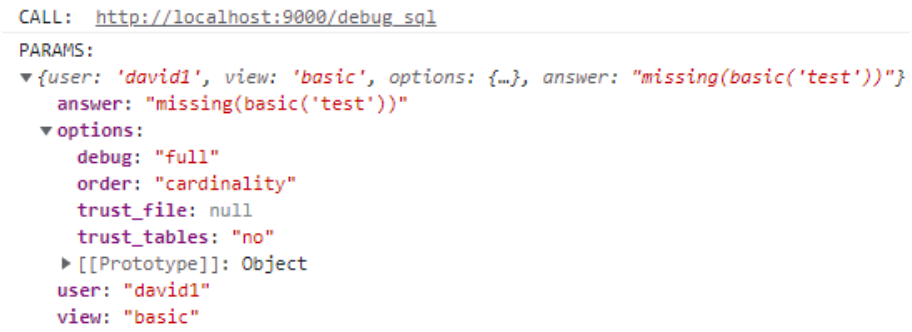

Figura 5.30: Envío de respuesta "Missing Tuple"

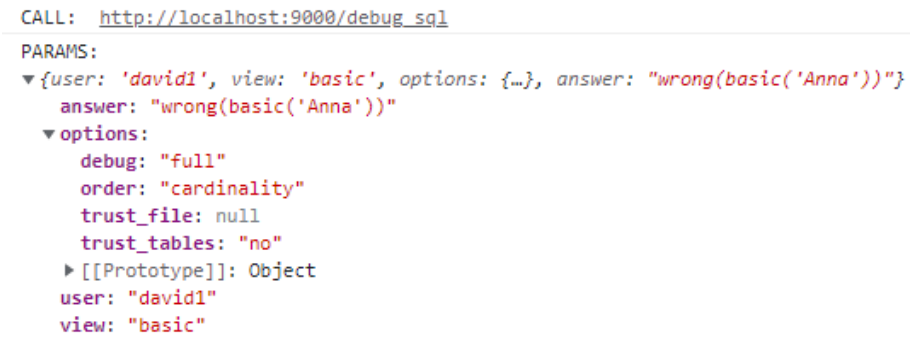

Figura 5.31: Envío de respuesta "Wrong Tuple"

El sistema nos devolverá una respuesta indicando si la sesión de depuración ha sido iniciada con éxito y, en caso afirmativo, una lista con los estados conocidos de los nodos:

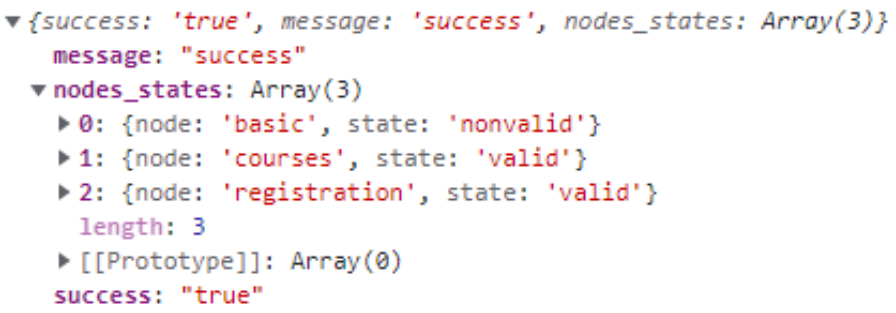

Figura 5.32: Respuesta del sistema al iniciar una sesión de depuración

Ahora debemos obtener la siguiente pregunta que nos hace el sistema. Para ello debemos realizar la siguiente llamada (comando `/debug_sql_current_question` por consola):

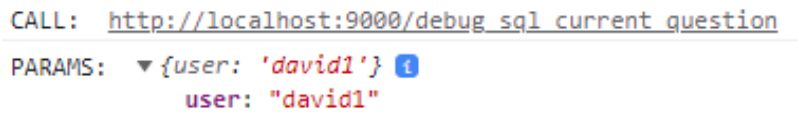

Figura 5.33: Llamada para obtener la pregunta actual

El sistema nos devolverá la pregunta de la siguiente manera:

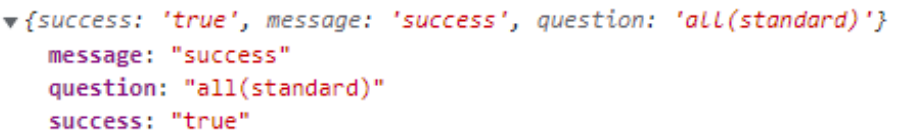

Figura 5.34: Respuesta del sistema con la pregunta actual

Vemos que la pregunta ahora es `all(standard)`. Por cada pregunta que nos haga el sistema deberemos hacer la llamada `/get_relation_json` explicada anteriormente para obtener las tuplas del predicado sobre el que se está preguntando y mostrarlas por pantalla.

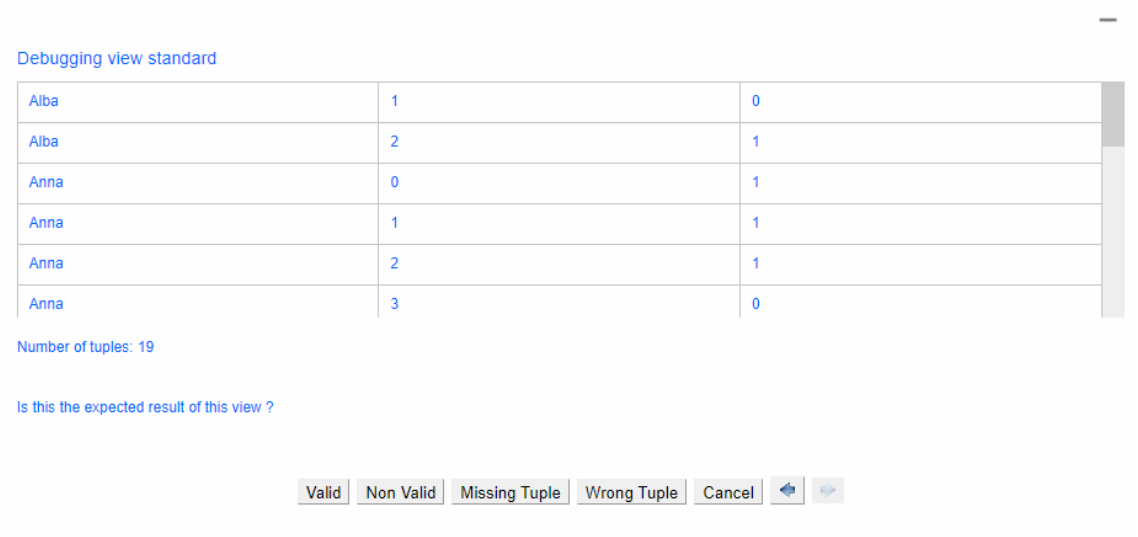

Figura 5.35: Tuplas de la vista `standard`

Una vez iniciada la depuración, el resto de respuestas utilizarán el servicio `/debug_sql_answer`:

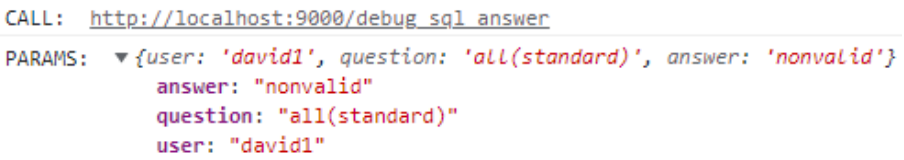

Figura 5.36: Respuesta a la pregunta `all(standard)`

El usuario continuará respondiendo las preguntas del sistema hasta que este indique que uno de los nodos es erróneo. En este caso la respuesta a la llamada anterior devuelve la siguiente información:

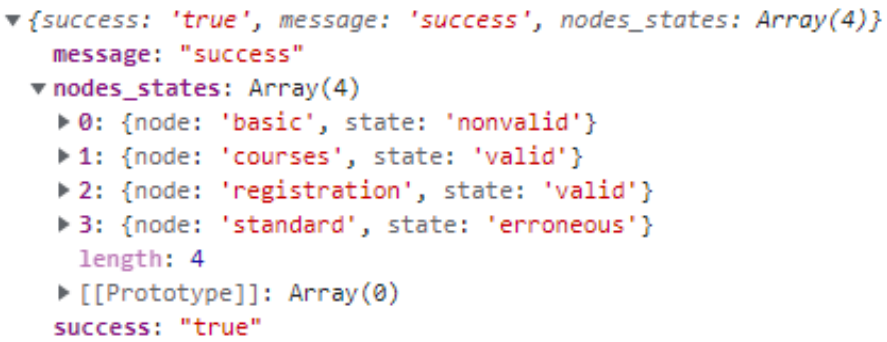

Figura 5.37: Respuesta del sistema indicando un nodo erróneo

Como se puede observar en la respuesta anterior se nos indica que el nodo

`standard` es erróneo. Cuando se recibe esta respuesta la ventana de depuración se cierra y en su lugar se muestra un modal indicando el fin de la depuración y el nodo problemático.

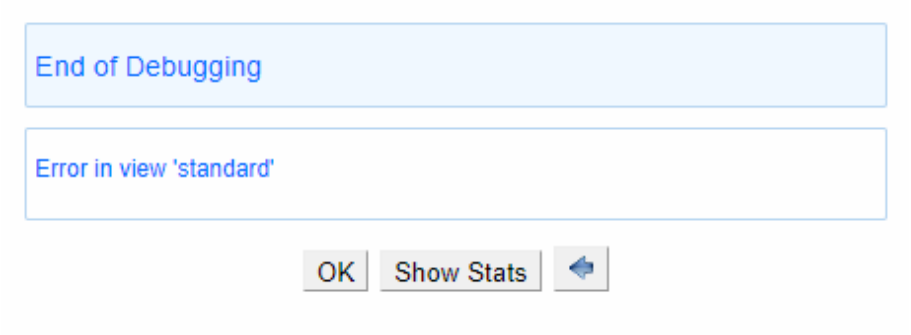

Figura 5.38: Fin de la depuración

También es posible ver información adicional de la sesión de depuración haciendo clic en el botón "Show Stats".

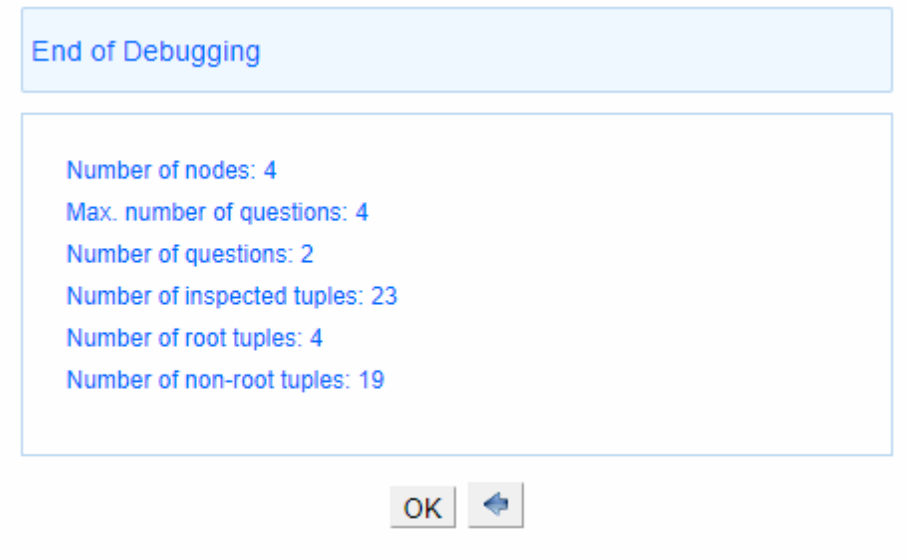

Figura 5.39: Estadísticas de la depuración

Capítulo 6

Conclusiones y trabajo futuro

El desarrollo e implementación de una interfaz gráfica para los lenguajes SQL y Datalog en la aplicación DESweb, objetivo principal del proyecto, ha sido realizado de manera satisfactoria. Esta interfaz cubre todo el flujo que se lleva a cabo en una sesión de depuración, lo que permite al usuario abstraerse de realizarla a través de la consola.

La interfaz desarrollada ha sido implementada en la estructura existente de DESweb de la manera menos disruptiva posible, manteniendo el sistema de paneles que utiliza la página para mostrarse al usuario, siendo posible ocultar este nuevo panel por completo si no se requiere su utilización.

El aspecto y herramientas del panel de depuración se han implementado teniendo en cuenta en gran medida la aplicación ACIDE, utilizando los mismos elementos visuales cuando ha sido posible concediendo así familiaridad a los usuarios provenientes de esta aplicación. Esto se puede observar en el aspecto visual de las herramientas así como en la distribución de los elementos del panel y las distintas ventanas de depuración.

También se han añadido mejoras en ciertos aspectos de la interfaz de DESweb con respecto a ACIDE. Por ejemplo, se ha llevado a cabo la unión de los paneles de depuración para SQL y Datalog en uno solo. También se han suprimido algunas herramientas como el zoom numérico optando por una aproximación más intuitiva como es hacer zoom a través de la rueda del ratón en el grafo de depuración. Estos cambios otorgan simplicidad a la interfaz, lo que acelera el proceso de acomodación de los usuarios a estos nuevos elementos.

Cabe destacar que la implementación de este nuevo panel en la interfaz ha sido realizada desde cero sobre el código existente en DESweb. Esto ha solicitado una gran cantidad de tiempo de investigación y estudio para la comprensión de la estructura del proyecto con objetivo de desarrollar código robusto y mantenible, que sirva de guía para futuros trabajos realizados sobre esta aplicación.

Los objetivos expuestos al principio de este documento los cuales no ha sido posible implementar debido a la falta de tiempo han sido los siguientes:

- Desplazamiento de los nodos del grafo de depuración mediante la utilización del ratón.
- Implementación de los diferentes temas de DESweb que permiten cambiar el aspecto visual de la página por completo.
- Presentación jerárquica de los nodos del grafo de depuración, ordenando estos en distintos niveles según la profundidad en la relación con la vista o predicado que se ha seleccionado para depurar.

Estos objetivos pueden servir como trabajo futuro para los siguientes proyectos sobre DESweb.

Introduction

In this document we will proceed to explain in detail the implementation of a user interface to assist in the task of debugging SQL and Datalog databases in the DESweb application.

It will be explained what is the method of declarative debugging, which is used in the DES system, as well as the methodology used to carry out this process in terms of communication with DES and, finally, the different parts of the added debugging interface and its corresponding functionalities will be described.

6.1. Motivation

The debugging task, also known as *debug*, is a critical moment when programming any type of application. That is why today we find multiple tools that provide an efficient way to carry out this process. Popular languages such as C#, Java and Python are based on the imperative programming paradigm, where the execution flow is clear and facilitates considerably the implementation of debugging tools. However, in the case of languages such as SQL or Datalog, we switch to a declarative programming paradigm, where we operate at a higher level of abstraction than in the previous ones, resulting in a major problem when it comes to mapping out a plan to find errors.

DES makes use of what is known as declarative debugging to alleviate this problem. This technique, unlike traditional debugging techniques, abstracts the execution details to focus on the results, offering a practical way to achieve an efficient result in this task. Although DESweb allows using this function by means of a communication through the console, this way ends up being tedious and unintuitive for the user who is going to perform a debugging of its database, since there is no visual reference beyond the one that the command console can offer. Because of this, one can get lost among the debugging steps and not achieve its main goal: to find and correct the source of the problem that causes a malfunction in the code in the most efficient way possible.

With this problem in mind, this project has arisen from the motivation to offer DESweb users an interface that serves as a bridge to the declarative debugging

system that DES offers, showing different elements such as nodes and relationships contained in graphs, which are used to represent the context of the database. Several modal windows have also been implemented with information about the questions and answers made in each phase of the debugging, as well as the different tuples contained in each view or SQL table, or in the Datalog predicates. It is worth mentioning the use of the ACIDE application as a reference for the construction of the debugging panel in its web version.

6.2. Background

The ACIDE application is closely related to the DESweb project, the latter being basically a web version of it. ACIDE is a project that has been developed by multiple contributors and has always been directed by Fernando Sáenz Pérez, tutor of this work. The chronology of the development of ACIDE is as follows:

- Academic year 2006-2007: version 0.1-0.6, carried out by Diego Cardiel Freire, Juan José Ortiz Sánchez and Delfín Rupérez Cañas.
- Academic year 2007-2008: version 0.7, by Miguel Martín Lázaro.
- Academic year 2010-2011: version 0.8, by Javier Salcedo Gómez.
- Academic year 2012-2013: version 0.9-0.11, by Pablo Gutiérrez García-Pardo, Elena Tejeiro Pérez de Ágreda and Andrés Vicente del Cura.
- Academic year 2013-2014: version 0.12-0.16, by Semiramis Gutiérrez Quintana, Juan Jesús Marqués Ortiz and Fernando Ordás Lorente.
- Academic year 2014-2015: version 0.17, by Sergio Domínguez Fuentes.
- Academic year 2019-2020: version 0.18, by Sergio García Rodríguez.
- Academic year 2020-2021: version 0.19, by Carlos González Torres and Cristina Lara López.

ACIDE version 0.18, developed by Sergio García Rodríguez [5], implemented the SQL debugging interface. Later, in version 0.19, Cristina Lara López was in charge of implementing this same interface in its Datalog variant [6]. These two works have served as reference for this project.

The DESweb application is essentially the result of the specialization for the DES system of the *Validation assistant of the TIN2013-44742-C4-3-R project Assisted program validation through analysis, annotations, mathematical demonstrations and proofs* tool. [8, 7]. During this project, the interface has been modified and the database panel with its various tabs has been added. The general DESweb interface will be discussed in more detail in Chapter 2 in the corresponding section.

6.3. Objectives

The objective of this work is to implement a graphical SQL and Datalog debugging interface for the DESweb application. This interface will make use of multiple tools to offer users a more intuitive and efficient debugging process, greatly facilitating the localization of errors in the code.

An additional debugging panel will be added to the existing ones in the application. This panel will be hidden (default behavior when logging into DESweb) and will be visible when a user wants to start a debugging session. The panel will have a graphical representation of the current state of the database through a graph formed by nodes (tables, views or predicates) and their directed connections, corresponding to the dependencies between nodes. This graph will be hierarchically ordered as shown in the ACIDE application, placing the node being debugged at the top of the panel and its relationships below in the form of levels. Additionally, the user will be allowed to move the nodes at will if he/she wishes to alter the placement of the nodes as shown by default.

At the bottom of the panel will be located the different tools available for the debugging phase. Among these we can find some such as the function to reload the network, select the view to be debugged, configure the debugging options and the button to start the debugging session.

This new panel will be part of the layout already existing in the current DESweb interface, using a similar design style and being possible to edit its dimensions, as it is possible to do with the other panels, as well as to switch between the existing color themes in the application, allowing the user to adapt the page style to his taste.

6.4. Work plan

Regarding the work plan followed during this academic year 2022-2023 it can be organized as follows (figure 6.1)):

- In September the tutor has provided access to a Google Drive folder where the code of the ACIDE application was located, a project that has served as a reference to later implement the debugging interface in the DESweb application. This first month has been dedicated to study the methodology used when debugging in the DES system, as well as the syntax used to communicate through the textual API via the console.

- In October the tutor has provided access to the Google Drive folder of DESweb version 1.7, version on which the necessary modifications have been made to implement the new debugging interface. We have also studied the documents concerning the DESweb user manual as well as the information concerning the structure of the project.

- The period from November to December was used to set up the DESweb project locally and to carry out functional tests to get to know the application. On the other hand, it has also been necessary to invest considerable amounts of time to study the code used in the project, since the application interface is implemented using the Prolog language through a technique for generating dynamic HTML called Termerized HTML.
- Once the operation and code of the project was understood, in January we started to develop the debugging interface, using as a reference the one already existing in ACIDE together with the tutor's guidelines.
- Finally, after finishing the development of the debugging interface, covering completely the debugging flow, we started with the realization of the project memory and applied the corrections suggested by the tutor. This has been carried out during the last half of April and May.

Throughout the development of the project, consultations have been made with the tutor via e-mail when the doubt could be solved in this way. There have also been face-to-face meetings at the faculty when it was necessary to clarify more complex doubts or review the status of the project, as well as tutorials by videoconference.

Tareas	2022				2023				
	Sept.	Oct.	Nov.	Dic.	Enero	Febrero	Marzo	Abril	Mayo
Pruebas en ACIDE a través de la consola y primer contacto con la interfaz de depuración									
Estudio de la estructura del código de DESweb, montaje del proyecto en local y pruebas de funcionamiento									
Implementación del panel de depuración en la interfaz con un contenedor vacío y las herramientas correspondientes sin funcionalidad									
Implementación del grafo de depuración en el panel									
Creación de la ventana modal de depuración y sus elementos									
Implementación de la depuración completa y su impacto en el aspecto visual del grafo									
Implementación de las herramientas del grafo: zoom, actualizar, navegar por los nodos, etc.									
Implementación del menú contextual									
Redacción y posterior corrección de la memoria del proyecto									

Figure 6.1: Gantt Diagram

Conclusions and Future Work

The development and implementation of a graphical interface for SQL and Datalog languages in the DESweb application, the main objective of the project, has been successfully completed. This interface covers all the flow that takes place in a debugging session, allowing the user to abstract from performing it through the console

The developed interface has been implemented in the existing DESweb structure in the least disruptive way possible, keeping the panel system that the page uses to show itself to the user, being possible to hide this new panel completely if its use is not required.

The look and feel and tools of the debug panel have been implemented with the ACIDE application in mind, using the same visual elements where possible, thus granting familiarity to users coming from this application. This can be seen in the visual appearance of the tools as well as in the layout of the panel elements and the different debugging windows.

Improvements have also been added to certain aspects of the DESweb interface with respect to ACIDE. For example, the debug panels for SQL and Datalog have been merged into one. Some tools such as the numerical zoom have also been removed, opting for a more intuitive approach such as zooming through the debug network using the mouse wheel. These changes provide simplicity to the interface, which speeds up the process of user's accommodation to these new elements.

It should be noted that the implementation of this new panel in the interface has been done from scratch on top of the existing DESweb code. This has required a great deal of research and study time to understand the structure of the project in order to develop robust and maintainable code that will serve as a guide for future work on this application.

The objectives stated at the beginning of this document which have not been possible to implement due to lack of time have been the following:

- Displacement of the nodes of the debugging network by using the mouse.
- Implementation of the different DESweb themes that allow to change the visual aspect of the page completely.

- Hierarchical presentation of the nodes of the debug network, ordering them in different levels according to the depth in the relation with the view or predicate that has been selected for debugging.

These objectives can serve as future work for the following projects on DESweb.

Contribuciones personales

En esta sección se van a exponer la contribución realizada por cada uno de los integrantes al proyecto a lo largo de la realización de este. Cabe destacar que, aunque en un primer momento este desarrollo se planteó como dos trabajos independientes, uno para SQL y otro para Datalog, el gran número de funcionalidades que ambos comparten, la necesidad de simplificar el código para no duplicar trabajo, así como la gran cooperación entre los dos alumnos hizo que finalmente se haya optado por presentar como un único trabajo.

Dado que tanto la depuración de SQL como la variante de Datalog utilizan en común gran parte de las funcionalidades creadas, gran parte del desarrollo ha sido realizado por los dos integrantes del grupo de manera indistinta, siendo ambas contribuciones de igual importancia y envergadura. Asimismo, muchas de las funcionalidades descritas previamente en este trabajo se han desarrollado de una forma genérica, para que puedan ser usadas por ambos lenguajes.

Las tareas realizadas por Laura Buquerín Arias son las siguientes:

- Tareas referentes a la investigación y diseño del proyecto:
  - Comunicación con el tutor y asistencia a tutorias.
  - Investigación del funcionamiento de ACIDE y estudio de las técnicas de depuración declarativa.
  - Investigación sobre el lenguaje Datalog.
  - Investigación sobre la implementación de DESweb y realización de pruebas de depuración mediante consola para comprender el funcionamiento de la herramienta.
- Tareas referentes al propio desarrollo del proyecto:
  - Implementación del panel de depuración en la interfaz de DESweb y la sección de herramientas de depuración.
  - Agregación de imágenes e iconos utilizados en ACIDE para los botones incluidos en la sección de herramientas de depuración.
  - Implementación del selector de vistas Datalog del panel de depuración.

- Implementación de la ventana utilizada para la sesión de depuración.
- Implementación de la ventana de configuración para la depuración de SQL y Datalog.
- Gestión de las tuplas mostradas en la ventana de depuración una vez que se ha iniciado una sesión de depuración Datalog así como cada vez que se responde al depurador.
- Implementación de un método que no permita al usuario enviar una respuesta del tipo *Missing Tuple* si no se ha indicado ninguna tupla.
- Implementación de un método que no permita al usuario enviar una respuesta del tipo *Wrong Tuple* si no se ha seleccionado ninguna tupla.
- Implementación del cambio de color de una fila de la tabla de tuplas cuando esta es seleccionada.
- Implementación de la segunda tabla que se muestra al enviar una respuesta del tipo *Wrong Tuple* en Datalog.
- Implementación de los botones con las distintas respuestas posibles a las preguntas realizadas durante la depuración.
- Implementación del botón de cancelación de la depuración.
- Implementación de los botones rehacer/deshacer.
- Implementación del menú contextual de los nodos al hacer clic derecho sobre uno de ellos.
- Visualización de las tuplas de una vista o predicado al hacer clic sobre un nodo.
- Implementación de la función de minimizar y mostrar de nuevo la ventana de depuración.
- Implementación de la función de arrastrado de la ventana de depuración.
- Implementación de la ventana de estadísticas de depuración.
- Visualización de los errores encontrados durante la sesión de depuración.

- Tareas referentes a la redacción de la presente memoria:

  - Redacción del capítulo 2: Depuración declarativa en SQL y Datalog, sección 2.2. Datalog.
  - Redacción del capítulo 3: Comunicación con DES, sección 3.2. Depuración de Datalog.
  - Redacción del capítulo 3: Comunicación con DES, sección 3.3. Ejemplo de depuración en Datalog.
  - Correcciones sugeridas por el tutor del proyecto sobre la presente memoria a lo largo de su realización.

Por otro lado, las tareas realizadas por David Stetco son las siguientes:

- Tareas referentes a la investigación y diseño del proyecto:
  - Comunicación con el tutor a través del correo o de forma presencial cuando esto fuera necesario.
  - Investigación del funcionamiento de ACIDE y posteriores pruebas para mejorar la comprensión de la técnica de depuración declarativa.
  - Investigación, montaje y testeo de la aplicación DESweb para comprender el funcionamiento de la herramienta.
- Tareas referentes al propio desarrollo del proyecto:
  - Implementación inicial del panel de depuración en la interfaz de DESweb, añadiendo el contenedor del grafo de depuración y las herramientas.
  - Obtención y agregación a la interfaz de los elementos visuales provenientes de ACIDE utilizados para representar las herramientas del panel de depuración.
  - Búsqueda e investigación de librerías Javascript que proporcionen soporte a la hora de dibujar el grafo de depuración en el panel. Finalmente, se optó por utilizar la librería *D3 v7.8.4* ya que cuenta con una comunidad muy activa y proporciona muchas opciones para futuras mejoras.
  - Implementación del selector de vistas SQL del panel de depuración.
  - Implementación del grafo de depuración en el panel, haciendo uso de la librería *D3* para dibujar los nodos y sus arcos.
  - Posterior generalización de la función encargada de dibujar el grafo para incluir esta en un controlador que gestione ambos tipos de bases de datos, tanto SQL como Datalog.
  - Gestión de la correcta visualización del grafo de depuración a la hora de mostrar la información cuando se carga información en el sistema.
  - Gestión de la información referente a los estados de cada nodo y arco de una base de datos SQL, así como su correcta representación visual en el grafo de depuración.
  - Implementación de las funciones de zoom en el grafo de depuración mediante las herramientas proporcionadas por la librería *D3*.
  - Desarrollo de la función del controlador encargada de gestionar los servicios web implementados por el tutor del proyecto para comunicarse con el sistema DES.
  - Implementación de las herramientas disponibles en el panel de depuración para recorrer los nodos del grafo de una base de datos SQL.
  - Gestión de las tuplas mostradas en la ventana de depuración una vez que se ha iniciado una sesión de depuración SQL así como cada vez que se responde al depurador.
  - Implementación de la fila extra en la tabla que contiene las tuplas en la ventana de depuración que se utiliza para proporcionar la información en una respuesta de tipo tupla faltante.

    - Implementación de las funciones encargadas de gestionar el correcto redimensionado del panel de depuración cuando se modificar su tamaño, el tamaño los demás paneles existentes en la interfaz o el tamaño de la propia ventana de la aplicación web.

- Tareas referentes a la redacción de la presente memoria:

    - Importación de la plantilla de *Latex* en la aplicación web Overleaf.
    - Redacción del resumen.
    - Redacción del capítulo 1: Introducción.
    - Redacción del capítulo 2: Depuración declarativa en SQL y Datalog, secciones 2.1. SQL y 2.3. Depuración declarativa.
    - Redacción del capítulo 3: Comunicación con DES, sección 3.1. Depuración de SQL.
    - Redacción del capítulo 4: Diseño de la interfaz de depuración, secciones 4.1. Acide y 4.2. DESweb, subsecciones 4.2.1. Servicio Web, 4.2.3. Tecnologías utilizadas, 4.2.4. HTML Termerized y 4.2.5. Interfaz.
    - Redacción del capítulo 6: Conclusiones y trabajo futuro.
    - Correcciones sugeridas por el tutor del proyecto sobre la presente memoria a lo largo de su realización.

Bibliografía

Y así, del mucho leer y del poco dormir, se le secó el celebro de manera que vino a perder el juicio.

Miguel de Cervantes Saavedra

[1] ACIDE, A Configurable IDE V0.18 User's Manual. https://www.fdi.ucm.es/profesor/fernan/ACIDE/html/download.html, 2021.

[2] Caballero, R., García Ruiz, Y., López Fraguas, F., Montenegro, M., Susana, N., Peña, R., Sáenz Pérez, J., Fernando amd Sánchez, Segura, C. y Blázquez Saborido, J. Designing a Validation Assistant for Multi-Language Formal Verification. 2023.

[3] Caballero, R., García Ruiz, Y. y Sáenz Pérez, F. Declarative Debugging of Wrong and Missing Answers for SQL Views. Disponible en https://www.fdi.ucm.es/profesor/fernan/FSP/CGS13a.pdf.

[4] Caballero, R., García Ruiz, Y. y Sáenz Pérez, F. Algorithmic Debugging of SQL Views. Disponible en https://federwin.sip.ucm.es/sic/investigacion/publicaciones/pdfs/SIC-3-11.pdf.

[5] García Rodriguez, S. Depuración de SQL. Disponible en https://eprints.ucm.es/id/eprint/68251/1/GARCIA_RODRIGUEZ_Depuracion_de_SQL_4398578_1134765792.pdf.

[6] Lara López, C. Depuración de Datalog y mejoras en ACIDE. Disponible en https://eprints.ucm.es/id/eprint/70740/1/Memoria_TFG_2021-2022%20Cristina%20Lara%20L%C3%B3pez.pdf.

[7] López Fraguas, F. J., Peña Marí, R. V., Nieva Soto, S., Sáenz Pérez, F., Sánchez Hernández, J. y Segura Díaz, C. M. TIN2017-86217-R. Disponible en https://produccioncientifica.ucm.es/proyectos/48206/detalle.

[8] Peña Marí, R. V., Caballero Roldán, R., García Ruiz, Y., López Fraguas, F. J., Montenegro Montes, M., Nieva Soto, S., Sáenz Pérez, F., Sánchez Hernández, J. y Segura Díaz, C. M. TIN2013-44742-C4-3-R. Disponible en https://produccioncientifica.ucm.es/proyectos/38933/detalle.

[9] Sáenz Pérez, F. Datalog Educational System V6.7 User's Manual. https://www.fdi.ucm.es/profesor/fernan/des/html/manual/manualDES.html, 2021.

[10] Wikipedia, T. F. E. SQL. https://en.wikipedia.org/wiki/SQL, 2022. Recuperado en mayo 21, 2023.

–¿Qué te parece desto, Sancho? – Dijo Don Quijote –
Bien podrán los encantadores quitarme la ventura,
pero el esfuerzo y el ánimo, será imposible.

Segunda parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes

–Buena está – dijo Sancho –; fírmela vuestra merced.
–No es menester firmarla – dijo Don Quijote–,
sino solamente poner mi rúbrica.

Primera parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes