

# DEPURACIÓN DE DATALOG Y MEJORAS EN ACIDE

## DATALOG DEBUGGING AND IMPROVEMENTS IN ACIDE



TRABAJO FIN DE GRADO  
CURSO 2021-2022

AUTOR  
CRISTINA LARA LÓPEZ

DIRECTOR  
PROF. FERNANDO SÁENZ PÉREZ

GRADO EN INGENIERÍA INFORMÁTICA  
FACULTAD DE INFORMÁTICA  
UNIVERSIDAD COMPLUTENSE  
DE MADRID

*"Se te dará en función del número de obstáculos que  
seas capaz de superar."*

*- Anónimo -*

## AGRADECIMIENTOS

---

A todos los que me animaron siempre a crecer y se alegraron de mis logros. Con especial mención:

A mi amigo **Oso**, por estar desde el principio hasta el final.

Al **Universo**, por cuidar siempre de mí.

A mis **profesores**, por su conocimiento.

A la **ciencia**, por salvarme la vida.

# ÍNDICE DE CONTENIDOS

---

Resumen .....	6
Abstract .....	8
Índice de figuras .....	10
1. Introducción .....	13
1.1. Estado del arte de las bases de datos deductivas .....	14
1.1.1. Sistemas de trabajo .....	14
1.1.2. Tecnologías utilizadas .....	15
1.2. Motivación .....	16
1.3. Antecedentes del trabajo .....	17
1.4. Objetivos pretendidos .....	18
1.5. Objetivos alcanzados .....	19
1.6. Plan de trabajo .....	21
2. Introduction .....	27
2.1. State of the art of deductive databases .....	28
2.1.1. Work systems .....	28
2.1.2. Technologies used .....	29
2.2. Motivation .....	30
2.3. Work background .....	31
2.4. Intented objectives .....	32
2.5. Achieved goals .....	32
2.6. Workplan .....	34
3. Depuración de Datalog .....	41
3.1. Introducción a Datalog .....	41
3.1.1. Bases de datos deductivas .....	41
3.1.2. Lenguaje Datalog .....	41
3.1.3. Reglas .....	42
3.1.4. Negación .....	43
3.1.5. Predicados intensionales y extensionales .....	44
3.1.6. Grafo de dependencias .....	44
3.1.7. Consultas .....	48
3.1.8. Comandos básicos .....	49
3.2. Depuración Datalog en ACIDE .....	56
3.2.1. Interfaz TAPI para la depuración .....	56
3.2.2. Opciones para la depuración .....	57
3.2.3. Comandos para la depuración .....	61
3.2.4. Diseño del panel de depuración .....	65
3.2.5. Detalles técnicos del panel de depuración .....	73
3.2.6. Nociones básicas sobre una sesión de depuración .....	78

3.2.7. Depuración manual por consola .....	78
3.2.8. Depuración gráfica guiada .....	85
4. Mejoras en ACIDE .....	102
4.1. Bloqueo del panel de depuración .....	102
4.2. Cambio de idioma dinámico .....	105
4.3. Mostrar iconos de idiomas .....	108
4.4. Ajuste de la sentencia SQL a editar .....	110
4.5. Mostrar información de depuración asociada a un nodo .....	112
4.6. Colorear los nodos tabla en función de la configuración aplicada .....	113
4.7. Obtener formateada la sentencia SQL de una vista .....	114
4.8. Indicar utilidad de los elementos .....	116
4.9. Cambios en la localización a los diferentes idiomas .....	118
4.10. Control de envío de comandos .....	119
4.11. Cambiar iconos .....	120
4.12. Corrección de caracteres corruptos .....	122
4.13. Añadir cadena deshacer-rehacer .....	123
4.14. Corrección de error al arrancar ACIDE .....	126
4.15. Corrección en el procesamiento de ficheros .....	128
4.16. Desplazamiento por un lienzo con dimensiones reducidas .....	130
4.17. Ventana de depuración SQL dinámica .....	131
4.18. Conteo de tuplas .....	132
4.19. Atajos de teclado para la depuración .....	133
Conclusiones .....	135
Conclusions .....	136
Bibliografía .....	137
Apéndice – Generación del ejecutable y arranque de ACIDE .....	138

## RESUMEN

---

Las bases de datos deductivas, mediante la definición de reglas, tienen la capacidad de inferir información a partir de información que ya tienen almacenada de forma explícita.

DES es un sistema que permite trabajar a través de una consola con bases de datos deductivas.

ACIDE es un entorno de desarrollo gráfico de código abierto multiplataforma especializado en bases de datos para el Álgebra Relacional, TRC, DRC, Datalog y SQL que, en este trabajo, está configurado para trabajar con funciones específicas de DES. También puede configurarse para usarse con intérpretes o compiladores.

La depuración de programas implementados con lenguajes declarativos como Datalog (lenguaje de consulta para bases de datos deductivas), para detectar la fuente de un error, requiere herramientas específicas debido a que el flujo de ejecución de dichos programas no es claro al ocultar sus mecanismos de resolución.

Aunque el sistema DES incluye una herramienta de depuración textual para programas implementados con el lenguaje declarativo Datalog que ayuda a los programadores a encontrar errores en los mismos, la herramienta tiene una utilidad limitada:

Durante el proceso de depuración no permite deshacer acciones ejecutadas ni inspeccionar a través de la interfaz de consola el contenido de un elemento que no se esté depurando en ese momento.

También fuerza a que el usuario se deba encargar de recordar la sintaxis de los diferentes comandos necesarios para llevar a cabo una sesión de depuración.

Debido a estas limitaciones entre otras, surge la necesidad de implementar una herramienta de depuración gráfica de programas Datalog que consiga extender las funcionalidades que ya de por sí ofrece la herramienta de depuración textual del sistema DES.

Mi contribución a la versión de ACIDE que me ocupa ha sido, en primer lugar, la implementación de una herramienta de depuración gráfica de programas Datalog

en ACIDE y, en segundo lugar, la realización de mejoras en la interfaz gráfica ACIDE como las siguientes: inclusión de la cadena deshacer-rehacer durante una sesión de depuración SQL, localización dinámica a los diferentes idiomas en secciones faltantes, proporcionar más información de cada nodo durante una sesión de depuración SQL, etc.

### **Palabras clave**

DES, ACIDE, depuración declarativa, Datalog, base de datos, localización, bases de datos deductivas, predicados lógicos.

## ABSTRACT

---

Deductive databases, by defining rules, have the ability to infer information from information they already have explicitly stored.

DES is a system that allows working through a console with deductive databases.

ACIDE is a cross-platform open source graphical development environment specialized in databases for Relational Algebra, TRC, DRC, Datalog and SQL that, in this work, is configured to work with specific functions of DES. It can also be configured for use with interpreters or compilers.

Debugging programs implemented with declarative languages such as Datalog (query language for deductive databases), in order to detect the source of an error, requires specific tools because the execution flow of such programs is not clear by hiding their resolution mechanisms.

Although the DES system includes a textual debugging tool for programs implemented with the Datalog declarative language that helps programmers find bugs in them, the tool is of limited utility:

During the debugging process, it does not allow you to undo executed actions or inspect through the console interface the content of an element that is not currently being debugged.

It also forces the user to take care of remembering the syntax of the different commands necessary to carry out a debugging session.

Due to these limitations, among others, the need arises to implement a graphical debugging tool for Datalog programs that manages to extend the functionalities already offered by the textual debugging tool of the DES system.

My contribution to the version of ACIDE that concerns me has been, firstly, the implementation of a graphical debugging tool for Datalog programs in ACIDE and, secondly, the implementation of improvements to the ACIDE graphical interface, such as the following: inclusion of the undo-redo chain during an SQL debugging session, dynamic localization to different languages in missing sections, providing more information about each node during an SQL debugging session, etc.



## **Keywords**

DES, ACIDE, declarative debugging, Datalog, database, location, deductive databases, logical predicates.

## ÍNDICE DE FIGURAS

---

<a href="#">Figura I-1 Listado de versiones publicadas .....</a>	<a href="#">26</a>
<a href="#">Figure I-1 List of published versions .....</a>	<a href="#">40</a>
<a href="#">Figura D-1 Identificación de los nodos .....</a>	<a href="#">46</a>
<a href="#">Figura D-2 Arista para una dependencia recursiva .....</a>	<a href="#">47</a>
<a href="#">Figura D-3 Aristas para dependencias positivas .....</a>	<a href="#">47</a>
<a href="#">Figura D-4 Aristas para dependencias negativas .....</a>	<a href="#">48</a>
<a href="#">Figura D-5 Arista final del PDG .....</a>	<a href="#">48</a>
<a href="#">Figura D-6 Carga de un programa Datalog .....</a>	<a href="#">53</a>
<a href="#">Figura D-7 Consulta de tuplas .....</a>	<a href="#">54</a>
<a href="#">Figura D-8 Consulta específica .....</a>	<a href="#">55</a>
<a href="#">Figura D-9 Consulta reglas cargadas en BD .....</a>	<a href="#">55</a>
<a href="#">Figura D-10 Información eliminada de la BD .....</a>	<a href="#">56</a>
<a href="#">Figura D-11 Pregunta depurador .....</a>	<a href="#">63</a>
<a href="#">Figura D-12 Estadísticas de depuración .....</a>	<a href="#">64</a>
<a href="#">Figura D-13 Diseño del panel de depuración Datalog .....</a>	<a href="#">66</a>
<a href="#">Figura D-14 Ventana configuración depuración Datalog .....</a>	<a href="#">67</a>
<a href="#">Figura D-15 Grafo con etiquetas .....</a>	<a href="#">69</a>
<a href="#">Figura D-16 Grafo sin etiquetas .....</a>	<a href="#">70</a>
<a href="#">Figura D-17 Selector de predicados .....</a>	<a href="#">71</a>
<a href="#">Figura D-18 Ubicación en el editor del predicado seleccionado .....</a>	<a href="#">72</a>
<a href="#">Figura D-19 Panel de control del panel depuración Datalog .....</a>	<a href="#">73</a>
<a href="#">Figura D-20 Clases del panel depuración Datalog y su relación .....</a>	<a href="#">76</a>
<a href="#">Figura D-21 Clases de la ventana de configuración .....</a>	<a href="#">77</a>
<a href="#">Figura D-22 Clases de la ventana de depuración .....</a>	<a href="#">77</a>
<a href="#">Figura D-23 Añadir archivos .....</a>	<a href="#">86</a>
<a href="#">Figura D-24 Fichero solarsystem.dl cargado .....</a>	<a href="#">87</a>
<a href="#">Figura D-25 Procesamiento del fichero .....</a>	<a href="#">87</a>
<a href="#">Figura D-26 Configuraciones previas a la depuración .....</a>	<a href="#">88</a>
<a href="#">Figura D-27 Botón con icono Play depuración .....</a>	<a href="#">89</a>
<a href="#">Figura D-28 Pregunta 1 depuración gráfica .....</a>	<a href="#">90</a>
<a href="#">Figura D-29 Pregunta 1 depuración gráfica con respuesta .....</a>	<a href="#">90</a>
<a href="#">Figura D-30 Nodo anaranjado .....</a>	<a href="#">91</a>
<a href="#">Figura D-31 Pregunta 2 depuración gráfica .....</a>	<a href="#">91</a>
<a href="#">Figura D-32 Visualizar contenido predicado .....</a>	<a href="#">92</a>
<a href="#">Figura D-33 Contenido predicado .....</a>	<a href="#">92</a>
<a href="#">Figura D-34 Nodo válido .....</a>	<a href="#">93</a>
<a href="#">Figura D-35 Pregunta 3 depuración gráfica .....</a>	<a href="#">94</a>

<a href="#"><u>Figura D-36 Pregunta 4 depuración gráfica .....</u></a>	<a href="#"><u>95</u></a>
<a href="#"><u>Figura D-37 Deshacer en depuración .....</u></a>	<a href="#"><u>96</u></a>
<a href="#"><u>Figura D-38 Pregunta 5 depuración gráfica .....</u></a>	<a href="#"><u>96</u></a>
<a href="#"><u>Figura D-39 Pregunta 6 depuración gráfica .....</u></a>	<a href="#"><u>97</u></a>
<a href="#"><u>Figura D-40 Ventana fin depuración .....</u></a>	<a href="#"><u>98</u></a>
<a href="#"><u>Figura D-41 Estadísticas depuración .....</u></a>	<a href="#"><u>98</u></a>
<a href="#"><u>Figura D-42 Editor predicado intensional .....</u></a>	<a href="#"><u>99</u></a>
<a href="#"><u>Figura D-43 Corrección intermediate .....</u></a>	<a href="#"><u>100</u></a>
<a href="#"><u>Figura D-44 Detalles gráficos fin depuración .....</u></a>	<a href="#"><u>101</u></a>
<a href="#"><u>Figura M-1 Error procesamiento de comandos .....</u></a>	<a href="#"><u>103</u></a>
<a href="#"><u>Figura M-2 Bloqueo panel de depuración .....</u></a>	<a href="#"><u>104</u></a>
<a href="#"><u>Figura M-3 Inductor DES .....</u></a>	<a href="#"><u>105</u></a>
<a href="#"><u>Figura M-4 Cambio dinámico de idioma .....</u></a>	<a href="#"><u>106</u></a>
<a href="#"><u>Figura M-5 Localización dinámica en panel de base de datos .....</u></a>	<a href="#"><u>107</u></a>
<a href="#"><u>Figura M-6 Localización dinámica en panel de traza SQL .....</u></a>	<a href="#"><u>107</u></a>
<a href="#"><u>Figura M-7 Localización dinámica en panel de depuración SQL .....</u></a>	<a href="#"><u>108</u></a>
<a href="#"><u>Figura M-8 Menú de idiomas sin iconos .....</u></a>	<a href="#"><u>109</u></a>
<a href="#"><u>Figura M-9 Menú de idiomas con iconos .....</u></a>	<a href="#"><u>110</u></a>
<a href="#"><u>Figura M-10 Editor sentencia SQL sin ajuste de línea .....</u></a>	<a href="#"><u>111</u></a>
<a href="#"><u>Figura M-11 Diferencia opción Ajuste de línea .....</u></a>	<a href="#"><u>112</u></a>
<a href="#"><u>Figura M-12 Información de depuración asociada a un nodo .....</u></a>	<a href="#"><u>113</u></a>
<a href="#"><u>Figura M-13 Ventana configuración sesión depuración SQL .....</u></a>	<a href="#"><u>114</u></a>
<a href="#"><u>Figura M-14 Diferencia salida con y sin formato TAPI .....</u></a>	<a href="#"><u>116</u></a>
<a href="#"><u>Figura M-15 Botón Locate sin descripción .....</u></a>	<a href="#"><u>117</u></a>
<a href="#"><u>Figura M-16 Botón Locate con descripción .....</u></a>	<a href="#"><u>118</u></a>
<a href="#"><u>Figura M-17 Localización incorrecta .....</u></a>	<a href="#"><u>119</u></a>
<a href="#"><u>Figura M-18 Localización correcta .....</u></a>	<a href="#"><u>119</u></a>
<a href="#"><u>Figura M-19 Comandos no reconocidos .....</u></a>	<a href="#"><u>120</u></a>
<a href="#"><u>Figura M-20 Icono configuración poco intuitivo .....</u></a>	<a href="#"><u>121</u></a>
<a href="#"><u>Figura M-21 Icono configuración representativo .....</u></a>	<a href="#"><u>122</u></a>
<a href="#"><u>Figura M-22 Tildes mal codificadas .....</u></a>	<a href="#"><u>122</u></a>
<a href="#"><u>Figura M-23 Cadena deshacer-rehacer ventana depuración .....</u></a>	<a href="#"><u>125</u></a>
<a href="#"><u>Figura M-24 Cadena deshacer-rehacer menú contextual .....</u></a>	<a href="#"><u>126</u></a>
<a href="#"><u>Figura M-25 Inicio correcto consola DES .....</u></a>	<a href="#"><u>127</u></a>
<a href="#"><u>Figura M-26 Inicio incorrecto consola DES .....</u></a>	<a href="#"><u>127</u></a>
<a href="#"><u>Figura M-27 Procesado con botón con icono Play .....</u></a>	<a href="#"><u>129</u></a>
<a href="#"><u>Figura M-28 Dimensiones reducidas del lienzo .....</u></a>	<a href="#"><u>130</u></a>
<a href="#"><u>Figura M-29 Scroll en lienzos .....</u></a>	<a href="#"><u>131</u></a>
<a href="#"><u>Figura M-30 Scroll en tabla .....</u></a>	<a href="#"><u>132</u></a>

<a href="#"><u>Figura M-31 Conteo de tuplas .....</u></a>	<a href="#"><u>133</u></a>
<a href="#"><u>Figura M-32 Atajos de teclado [8] .....</u></a>	<a href="#"><u>134</u></a>
<a href="#"><u>Figura A-1 Opción Export Eclipse .....</u></a>	<a href="#"><u>139</u></a>
<a href="#"><u>Figura A-2 Opción jar ejecutable .....</u></a>	<a href="#"><u>140</u></a>
<a href="#"><u>Figura A-3 Exportación jar .....</u></a>	<a href="#"><u>141</u></a>
<a href="#"><u>Figura A-4 Aceptar diálogo .....</u></a>	<a href="#"><u>141</u></a>
<a href="#"><u>Figura A-5 Ubicación archivo java.exe .....</u></a>	<a href="#"><u>142</u></a>
<a href="#"><u>Figura A-6 Intérprete de comandos cmd.exe .....</u></a>	<a href="#"><u>142</u></a>
<a href="#"><u>Figura A-7 Prompt ubicado en raíz de proyecto ACIDE .....</u></a>	<a href="#"><u>143</u></a>
<a href="#"><u>Figura A-8 Comando construido .....</u></a>	<a href="#"><u>143</u></a>
<a href="#"><u>Figura A-9 Arranque ACIDE .....</u></a>	<a href="#"><u>143</u></a>

# 1. INTRODUCCIÓN

---

El sistema DES (<http://des.sourceforge.net/>), es una implementación de código abierto y multiplataforma desarrollado para poder trabajar a través de una consola con bases de datos deductivas realizando acciones en ellas como las siguientes: consultas a una base de datos deductiva o depuración de la misma.

Aunque DES originalmente fue concebido como un sistema para la enseñanza de bases de datos deductivas, actualmente incorpora muchas extensiones: lenguaje SQL para acceder a relaciones compartidas con el resto de lenguajes a los que da soporte (Datalog, cálculo relacional de tuplas – TRC: Tuple Relational Calculus – y de dominios – DRC: Domain Relational Calculus – y álgebra relacional -RA: Relational Algebra).

Es un sistema muy usado internacionalmente (alrededor de 84K descargas) en enseñanza (hay noticias de hasta 50 universidades) e investigación para fines como los siguientes: estudios de fallos de seguridad en Firefox, toma de decisiones cuantitativa en ingeniería del software, web semántica, etc. [9]

El proyecto ACIDE proporciona un entorno de desarrollo integrado gráfico, configurable, gratuito y multiplataforma que se puede configurar para ser utilizado con cualquier sistema de desarrollo, como intérpretes, compiladores y sistemas de bases de datos.

Las características de este sistema incluyen: gestión de proyectos, edición de múltiples archivos, coloreado de sintaxis, panel de consola y panel de base de datos (probado con DB2, MySQL, Oracle, PostgreSQL, SQL Anywhere y DES).

Los usuarios finales que pueden beneficiarse de este sistema incluyen: usuarios de consolas y bases de datos, investigadores que desarrollan sistemas de programación y desarrolladores. Está implementado en Java para ser independiente de la plataforma.

En esta memoria se describe la implementación de la herramienta de depuración gráfica de programas Datalog realizada en ACIDE como tarea principal en la realización de este trabajo de fin de grado detallando:

- La teoría previa que se debe conocer para poder llevar a cabo con éxito una sesión de depuración de una base de datos Datalog.
- Uso del sistema DES con Datalog.
- El diseño del panel de depuración Datalog: diagrama de clases y diseño gráfico.
- El proceso de una sesión de depuración de una base de datos Datalog.

Además, también se describen el resto de implementaciones realizadas en ACIDE con el objetivo de mejorar la experiencia de usuario.

Los apartados 3 y 4 de esta memoria proporcionan detalles sobre las funcionalidades implementadas.

Al final de la memoria, en el apartado "Apéndice – Generación del ejecutable y arranque de ACIDE", se encuentra una guía explicativa del proceso de generación del ejecutable de la aplicación así como la ejecución del mismo para futuros contribuidores al proyecto ACIDE.

## 1.1. Estado del arte de las bases de datos deductivas

Las investigaciones realizadas a lo largo de los años sobre bases de datos deductivas han permitido desarrollar sistemas capaces de trabajar con las mismas.

En concreto, a través de demostraciones automáticas de teoremas y a la programación lógica se pudo demostrar que la deducción puede llevarse a cabo sistemáticamente en un entorno de base de datos.

En los siguientes sub apartados se darán a conocer tanto algunos de los sistemas existentes hoy en día para trabajar con bases de datos deductivas como las tecnologías que se han utilizado para desarrollar este trabajo.

### 1.1.1. Sistemas de trabajo

La siguiente tabla indica algunos de los sistemas desarrollados hasta el momento que permiten trabajar con bases de datos deductivas [10,14] además de describir brevemente sus características más relevantes.

Sistema	Descripción
DEDUCE	Sistema de deducción automática implementado por IBM a mediados de los

	años 70. Soporta reglas recursivas.
Chat-80	Implementado en 1981 por Warren y Pereira. Permite realizar consultas a una base de datos deductiva que almacena información relacionada con la geografía mundial.
Sistema de Dahl	Implementado en 1982. Introduce las nociones de expresiones establecidas y relaciones distributivas y colectivas.
Proyecto LDL	Su desarrollo inició en 1984 fuera del ámbito de las universidades y originó el prototipo LDL++ cuyo desarrollo comenzó en 1990 con el objetivo de solventar los problemas que tenía el sistema LDL para trabajar con aplicaciones reales. Entre otras cosas, el sistema LDL++ soporta tanto negación estratificada como no estratificada, así como agregación.
NAIL	Su desarrollo arrancó en 1985. Algunas de sus características es que soporta negación estratificada, negación bien fundamentada y negación estratificada modularmente. Trabaja con el lenguaje GLUE que permite construcciones de bucles, procedimientos y módulos que envuelven reglas lógicas cuyo poder se equipara con el de las sentencias SQL.
LOLA	Comenzó a desarrollarse en 1988. Soporta, entre otras cosas, negación estratificada, agregación y mantenimiento de restricciones de integridad.
XSB	Fue lanzado por primera vez en 1993. Es un sistema más rápido que otros para realizar consultas y conjuntos de reglas estratificadas. Algunas de sus características es que soportar negación bien fundamentada, recursión y agregación. Además, es multiplataforma, ya que puede trabajar tanto en Unix como Windows.
DES	Fue lanzado por primera vez en el año 2004. Las características del sistema han sido descritas al comienzo del apartado 1 de esta memoria.

### 1.1.2. Tecnologías utilizadas

A continuación, se exponen las herramientas utilizadas para llevar a cabo la implementación de las tareas requeridas en este trabajo:

– IntelliJ IDEA 2020.3.2:

Es un entorno de desarrollo integrado para el desarrollo de programas informáticos.

Su utilidad en este trabajo ha sido la de servir como herramienta para codificar en el proyecto ACIDE todas las tareas requeridas en este trabajo. Además, ha permitido la exportación del ejecutable de la versión de escritorio de la aplicación.

- Sistema ACIDE versión 0.18:

El sistema ha sido utilizado para actuar como interfaz gráfica de comunicación entre el usuario final y el sistema DES.

- Sistema DES versión 6.7:

Ha sido el sistema seleccionado para que el usuario pueda trabajar con bases de datos deductivas realizando consultas a las mismas y/o depurándolas.

## 1.2. Motivación

La ausencia de sistemas que permitieran la depuración de bases de datos deductivas motivó la inclusión de una interfaz gráfica de depuración en ACIDE.

Aunque el sistema DES ofrece una interfaz de consola para realizar una depuración textual de una base deductiva donde se encuentra cargado un programa Datalog, llevar a cabo el proceso de depuración a través de dicha interfaz es un mecanismo tedioso y propenso a errores por los siguientes motivos:

- El usuario que lleva a cabo una sesión de depuración se debe encargar de consultar e introducir manualmente todos los comandos y datos requeridos durante el proceso. Esto provoca que el proceso de depuración se alargue.

Además, durante el proceso, puede introducir datos erróneos, por ejemplo, en las siguientes situaciones:

\* A la hora de especificar una tupla errónea, introduce mal los datos que componen la tupla que quiere marcar como errónea.

\* Introduce una acción para realizar el siguiente paso en la depuración que no existe.

- Durante el proceso de depuración, el usuario no tiene una visión clara de la dependencia que hay entre los elementos a depurar.

- El usuario no puede revertir decisiones tomadas. Por tanto, si llevó a cabo una acción que no era correcta como marcar como errónea una tupla válida, para corregir este error, deberá comenzar de nuevo la sesión de depuración partiendo desde cero.

- Durante la depuración no puede consultar libremente ciertos detalles como el



contenido de un elemento que en ese momento no se esté depurando, pero que forme parte de la depuración (que se haya depurado previamente o esté pendiente de depurar).

Debido a todos los inconvenientes que pueden surgir durante una sesión de depuración textual, surge la necesidad de implementar en la interfaz gráfica de ACIDE un panel de depuración Datalog que evite errores de usuario durante el proceso de depuración al aportar ventajas como las siguientes:

- En ningún momento el usuario tendrá que dedicarse a introducir datos manualmente si no que los podrá seleccionar de entre los datos sugeridos. Esto evita que el usuario introduzca datos erróneos.
- El usuario no tendrá que preocuparse de introducir manualmente comandos para realizar los siguientes pasos, ya que cuando pulse el botón asociado a la acción que quiere realizar el comando apropiado será enviado automáticamente a la consola DES.

Esto evita que el usuario introduzca comandos erróneos o tenga que invertir tiempo en buscar el comando adecuado.

- El usuario en todo momento tendrá disponible un grafo que le permite tener una visión clara de las dependencias que hay entre los elementos a depurar.
- Podrá deshacer acciones llevadas a cabo durante la depuración. Esto evita que se deba comenzar la depuración desde cero.
- Podrá consultar en todo momento el contenido o estado de cualquier elemento implicado en la depuración.
- Podrá localizar más fácilmente en el programa Datalog la definición de los elementos a depurar, ya que será resaltada.

### 1.3. Antecedentes del trabajo

Este trabajo es una evolución del proyecto ACIDE que siempre ha sido dirigido por Fernando Sáenz Pérez y que, a lo largo de los años, se ha ido desarrollando por los siguientes contribuidores:

- Durante el curso académico 2006-2007, Diego Cardiel Freire, Juan José Ortiz Sánchez y Delfín Rupérez Cañas se encargaron de realizar las versiones de la 0.1 a la 0.6 del proyecto.
- Durante el curso académico 2007-2008, Miguel Martín Lázaro se encargó de realizar la versión 0.7 del proyecto.
- Durante el curso académico 2010-2011, Javier Salcedo Gómez se encargó de

realizar la versión 0.8 del proyecto.

- Durante el curso académico 2012-2013, Pablo Gutiérrez García-Pardo, Elena Tejeiro Pérez de Ágreda y Andrés Vicente del Cura se encargaron de realizar las versiones de la 0.9 a la 0.11 del proyecto.
- Durante el curso académico 2013-2014, Semiramis Gutiérrez Quintana, Juan Jesús Marqués Ortiz y Fernando Ordás Lorente se encargaron de realizar las versiones de la 0.12 a la 0.16 del proyecto.
- Durante el curso académico 2014-2015, Sergio Domínguez Fuentes se encargó de realizar la versión 0.17 del proyecto.
- Durante el curso académico 2019-2020, Sergio García Rodríguez se encargó de realizar la versión 0.18 del proyecto.
- Durante el curso académico 2020-2021, Carlos González Torres [4] así como la autora de la presente memoria (Cristina Lara López) nos encargamos de realizar la versión 0.18.1 del proyecto.

En este curso académico 2021-2022, la autora de la presente memoria (Cristina Lara López) ha continuado el desarrollo del proyecto a partir de la versión 0.18.1. Mis contribuciones han generado la versión 0.19 del proyecto ACIDE.

## 1.4. Objetivos pretendidos

Este trabajo final de grado tenía como objetivo la implementación del panel para la depuración de bases de datos Datalog que estaba pendiente de desarrollar en ACIDE así como mejorar algunos aspectos de la interfaz:

### **Desarrollo principal**

Usando como base el depurador textual de DES disponible, implementar el panel para la depuración de bases de datos Datalog con los comandos y opciones apropiadas.

### **Mejoras relevantes**

- Bloquear el panel de depuración cuando la consola ACIDE está procesando comandos TAPI.
- Implementar el cambio de idioma dinámico en paneles de base de datos, traza y depuración.
- Mejorar el depurador SQL: proporcionar más información sobre cada nodo durante la depuración (tuplas indicadas como erróneas o faltantes), cadena de

deshacer-rehacer, atajos de teclado, conteo de tuplas, etc.

- Añadir textos descriptivos que informen de la utilidad de los diferentes componentes: botones, casillas, etc.
- Realizar cambios en la localización a los diferentes idiomas.
- Corregir errores: error en la consola durante el arranque de ACIDE, error durante el procesamiento de ficheros, etc.
- Habilitar el desplazamiento por el contenido cuando este no es mostrado por entero en el área disponible.

## 1.5. Objetivos alcanzados

Todos los objetivos que se pretendían alcanzar descritos en la sección 1.4 de esta memoria han podido ser implementados sin bloqueos relevantes:

### Desarrollo principal

Se ha desarrollado un panel de depuración Datalog que es capaz de realizar un proceso de depuración completo de una base de datos Datalog gracias a la inclusión de los siguientes detalles:

- Un selector que sugiere un listado de elementos disponibles para ser depurados pertenecientes al programa Datalog cargado en la base de datos. El usuario debe seleccionar el elemento que le interesa depurar.
- Una ventana de configuración de la depuración.
- Un lienzo en el que se dibuja el grafo con los nodos a depurar y que refleja las dependencias que tienen entre sí.
- Resalte en el editor de programas de las líneas que definen el elemento seleccionado en el grafo.
- Un botón para arrancar una sesión de depuración: abre una ventana gráfica que muestra todas las opciones disponibles para realizar el siguiente paso en la depuración.

Parte de la información que muestra la ventana gráfica de depuración es la siguiente: tuplas del elemento que se está depurando y acciones disponibles que se pueden llevar a cabo sobre el mismo incluyendo una acción de deshacer o rehacer para volver a un estado previo o posterior en la depuración, número que indica la cantidad de tuplas que contiene el elemento, etc.

- Un menú contextual disponible por cada nodo del grafo a depurar que permite realizar las siguientes acciones sobre el mismo: consultar sus tuplas, editar

su contenido, aplicar acciones para depurarlo, etc.

Los detalles de la implementación del panel de depuración Datalog se pueden consultar en los apartados 3.2.4 y 3.2.5 de esta memoria.

## **Mejoras relevantes**

Algunas de las mejoras globales implementadas en la interfaz gráfica ACIDE han sido las siguientes:

- Se ha realizado la implementación correspondiente para bloquear el panel de depuración cuando la consola DES se encuentra procesando comandos TAPI.

En concreto, si la última línea de la consola DES únicamente está compuesta del inductor DES correspondiente, se deduce que la consola ya no se encuentra procesando comandos y desbloquea el panel de depuración. En caso contrario, bloquea el panel de depuración.

- Se ha implementado el cambio de idioma dinámico para que, sin necesidad de reiniciar ACIDE, las etiquetas y descripciones de botones y resto de elementos de los paneles de base de datos, traza y depuración se actualicen al nuevo idioma aplicado.
- Se han añadido textos descriptivos que informan de la utilidad de cada botón o casilla del panel de traza o depuración cuando el ratón se posiciona sobre dichos componentes.
- Se han corregido caracteres corruptos de textos que se mostraban en diferentes ubicaciones de la interfaz gráfica ACIDE.
- Se ha corregido un error que surgía al arrancar ACIDE y que provocaba que hubiera que reiniciar manualmente su consola.
- Se ha corregido el problema de procesamiento de los ficheros (script SQL, programa Datalog, etc.).
- Se han mejorado los lienzos del panel de depuración para que permitan un desplazamiento por el grafo cargado cuando sus dimensiones son reducidas.

Las mejoras implementadas durante una sesión de depuración SQL han sido las siguientes:

- Se ha añadido una opción de "Deshacer" que permite volver a un estado anterior en la depuración.

- Se ha añadido una opción de "Rehacer" que permite volver a un estado posterior al que se había llegado durante el proceso de depuración.
- Se ha implementado una funcionalidad que proporciona información de depuración adicional a cerca de un nodo depurado (tuplas erróneas, faltantes, etc.) cuando el usuario, durante el proceso de depuración, posiciona el ratón sobre el mismo.
- Se ha mejorado la ventana de depuración SQL para que los elementos que contiene se ajusten a sus dimensiones.
- Se indica la cantidad tuplas que contiene la vista o tabla que se está depurando.
- Se han configurado atajos de teclado para las opciones de "Deshacer" y "Rehacer".

El resto de mejoras requeridas e implementadas se detallan en el apartado 4 de esta memoria.

## 1.6. Plan de trabajo

Todo el trabajo que he realizado en el proyecto ACIDE lo he llevado a cabo entre el curso anterior (2020-2021) y el actual (2021-2022).

El cronograma presentado en la siguiente página refleja el detalle de las tareas que he ido llevando a cabo para completar este trabajo así como el intervalo de tiempo en meses que dedicaba a las mismas. Las tareas de mejora en ACIDE se identifican por "(M)".

Tareas	2020				2021												2022
	S	O	N	D	E	F	M	A	M	J	J	A	S	O	N	D	E
Estudio estructura proyecto ACIDE, manual DES y estado del arte																	
Cambio de idioma dinámico (M)																	
Ajuste sentencia SQL a editar (M)																	
Colorear nodos tabla en función de la configuración aplicada (M)																	
Indicar utilidad de los elementos (M)																	
Cambios en la localización a los diferentes idiomas (M)																	
Corrección de caracteres corruptos (M)																	
Bloqueo del panel de depuración (M)																	
Mostrar iconos de idiomas (M)																	
Mostrar información de depuración asociada a un nodo (M)																	
Obtener formateada sentencia SQL de vista (M)																	
Control de envío de comandos (M)																	
Estudio lenguaje Datalog																	
Panel de depuración Datalog																	
Cambiar iconos (M)																	
Añadir cadena deshacer-rehacer (M)																	
Corrección de error al arrancar ACIDE (M)																	
Corrección en el procesamiento de ficheros (M)																	
Desplazamiento por lienzo con dimensiones reducidas (M)																	
Ventana de depuración SQL dinámica (M)																	
Conteo de tuplas (M)																	
Atajos de teclado para la depuración (M)																	

Al comienzo de mi contribución al proyecto en septiembre de 2020, su director, me facilitó el acceso a una carpeta compartida en Google Drive [2] que contenía:

- La versión 0.18 del proyecto ACIDE comprimida que incluía múltiples carpetas y archivos indispensables para poder ejecutar la última versión de ACIDE así como el código fuente sobre el que debía empezar a trabajar.
- Un documento con las tareas que tenía que realizar para implementar el panel de depuración Datalog requerido.
- Un documento que priorizaba las tareas de mejora que tenía que llevar a cabo.

Una vez me proporcionó estos accesos, entre los meses de septiembre y octubre del año 2020 me dediqué a desplegar el proyecto ACIDE en mi equipo personal para estudiar la estructura y funcionalidad de la versión 0.18. Además, estuve dedicada a:

- 1) Estudiar las secciones SQL del manual DES [1] y la memoria del TFG de Sergio García Rodríguez [5] para conocer el procedimiento que se debía seguir a la hora de realizar una sesión de depuración SQL.
- 2) Estudiar, agrupar y resumir el trabajo que se había llevado a cabo hasta el momento para trabajar con bases de datos deductivas: entornos, herramientas disponibles de depuración, etc.
- 3) Realizar diferentes pruebas en la interfaz gráfica ACIDE para estudiar su comportamiento.

Posteriormente, entre los meses de noviembre de 2020 y marzo de 2021 me dediqué a implementar las mejoras solicitadas en la interfaz ACIDE relacionadas con los siguientes aspectos:

Mejoras en los paneles de traza y depuración SQL y mejoras genéricas como corrección de caracteres corruptos en los textos informativos de la interfaz.

Por cada mejora o conjunto de mejoras implementadas, generaba un ejecutable de la aplicación.

El nuevo ejecutable de la aplicación generado lo subía a la carpeta compartida de Google Drive [2] con la que trabajaba para que pudiera ser probado por el director del TFG y diera su posterior validación.

Una vez había terminado de implementar las mejoras anteriores, antes de comenzar a implementar el panel de depuración Datalog, durante los meses de abril, mayo y junio de 2021 me dediqué a estudiar el lenguaje Datalog resumiendo los siguientes conceptos:

- 1) Conceptos básicos del lenguaje Datalog. La información la obtuve de la documentación disponible en Internet [6].
- 2) Uso del sistema DES con el lenguaje Datalog. La información la obtuve de la documentación proporcionada por el director del TFG [7]. Además, mientras realizaba el resumen, practiqué con el sistema DES siguiendo los tutoriales de la documentación.
- 3) Comandos y opciones disponibles para llevar a cabo una sesión de depuración básica de programas Datalog. Esta información se refleja con detalle en las secciones del manual del sistema DES [1] que hacen referencia a la depuración con el lenguaje Datalog.

Para practicar los conceptos adquiridos, iba realizando sesiones de depuración de programas Datalog en la consola DES.

Posteriormente, una vez estructurados todos los conceptos relacionados con el lenguaje Datalog, en julio de 2021 comencé a desarrollar el panel de depuración Datalog y finalicé en enero de 2022. En paralelo iba implementando mejoras pendientes de realizar que afectaban tanto a funcionalidades SQL como Datalog.

Durante el proceso de pruebas de las funcionalidades implementadas, si detectaba una salida incorrecta tras la ejecución de comandos por la consola, se lo notificaba al director del TFG para que sacara una versión mejorada del sistema DES. Para las mejoras, se aplicó el mismo procedimiento de validación que para el panel de depuración Datalog.

Todas las tareas implementadas están alojadas en el siguiente repositorio de mi cuenta GitHub que mantiene un control de los cambios e implementaciones que he ido realizando hasta alcanzar la versión 0.19 del proyecto:

<https://github.com/crislara/ACIDE>

A finales de marzo de 2021, tras haber implementado varias tareas de mejora, se procedió a publicar una versión intermedia a la actual del proyecto ACIDE que fue la versión 0.18.1. Esta versión de la aplicación está disponible en el apartado de




descargas de la página web dedicada a ACIDE [3]:

<http://www.fdi.ucm.es/profesor/fernan/ACIDE/html/download.html>

En la página web anterior se encuentran:

- Las diferentes versiones de ACIDE publicadas disponibles para ser descargadas.
- Una guía que indica cómo configurar la versión descargada en el sistema operativo que interese (Windows, Linux, etc.) así como la configuración necesaria para interpretar la sintaxis del lenguaje utilizado para crear los programas con los que se vaya a trabajar.
- Un manual de usuario de uso del sistema DES [1].
- Un fichero "releasenotesACIDE.txt" que describe los cambios implementados en la última versión publicada de ACIDE.



[Home](#)  
[Download](#)  
[Screenshots](#)  
[Release Notes](#)  
[License](#)  
[News](#)  
[Contact](#)  
[Credits](#)  
[Publications](#)  
[Statistics](#)  
[Pre-built Bundles](#)

## Download

**Warnings:**

This software comes with no warranty (see [License](#)) and it is in the development stage (alpha version) so that many expected functionalities will not work.

**Current Version:**

[ACIDE](#) version 0.18.1 (released on March, 28th, 2021)  
[Sources](#)  
[User Manual](#) (PDF)  
[Release Notes](#) (.txt)  
Look also for [Pre-built Bundles](#)

**Getting Started:**

Download ACIDE alpha version from above, decompress it, and execute the jar file. As released, this system is preconfigured with a sample project for MS Windows and its command shell (`cmd.exe`).

Typically, to configure ACIDE to work with any given system:

**First:** Configure the shell (Configuration -> Shell) and select the executable of the system shell (interpreter, compiler, database, ...) (e.g., `C:\Windows\system32\cmd.exe`), its working directory (if any), the quit command (e.g., `exit`), and whether the shell echoes user inputs (as, e.g., for the Windows operating system shell).

**Second:** Configure the toolbar (Configuration -> Toolbar) for adding new commands that can be executed from the icons in the toolbar. Give a toolbar icon name (e.g., `type`), the command that will be executed (e.g., `type %activeFile%`, which stands for executing the command type followed by the name of the selected file in the multifile editor), a help text if desired (e.g., Command for typing the current file), a file image (if desired). The icons in a toolbar can be saved as a list (Save list) and loaded afterwards (Load list).

**Third:** Configure or load the (programming) language for syntax colouring (Configuration -> Lexicon), and file extension associations (e.g., `.bat`). Currently, parsing is included but it cannot be applied.

**Fourth:** Try to open a file and check whether the syntax is highlighted and whether you can submit your commands to the shell clicking your buttons (e.g., `type`).

More details can be found in the [User Manual](#) above.

**Previous Versions:**

[ACIDE](#) version 0.18 (released on January, 25th, 2021). [Sources](#)  
[ACIDE](#) version 0.17 (released on December, 26th, 2015). [Sources](#)  
[ACIDE](#) version 0.16 (released on August, 14th, 2014). [Sources](#)  
[ACIDE](#) version 0.15 (released on April, 29th, 2014). [Sources](#)  
[ACIDE](#) version 0.14 (released on March, 12th, 2014)  
[ACIDE](#) version 0.13 (released on February, 3rd, 2014). [Sources](#)  
[ACIDE](#) version 0.12 (released on December, 18th, 2013). [Sources](#)  
[ACIDE](#) version 0.11 (released on July, 8th, 2013). [Sources](#)  
[ACIDE](#) version 0.10 (released on April, 24th, 2013). [Sources](#)  
[ACIDE](#) version 0.9 (released on December, 20th, 2012)  
[ACIDE](#) version 0.8 (released on July, 8th, 2011). [Sources](#)  
[ACIDE](#) version 0.7 (released on November, 5th, 2008). [Sources](#)  
[ACIDE](#) version 0.6 (released on August, 20th, 2007). [Sources](#)  
[ACIDE](#) version 0.5 (released on June, 4th, 2007)  
[ACIDE](#) version 0.4 (released on May, 18th, 2007)  
[ACIDE](#) version 0.3 (released on April, 25th, 2007)  
[ACIDE](#) version 0.2 (released on April, 16th, 2007)  
[ACIDE](#) version 0.1 (released on April, 9th, 2007)  
[ACIDE](#) version 0.0 (launched on March, 2007)

Figura I-1 Listado de versiones publicadas

## 2. INTRODUCTION

---

The DES system (<http://des.sourceforge.net/>), is an open source and multiplatform implementation developed to be able to work through a console with deductive databases performing actions on them such as: querying a deductive database or debugging it.

Although DES was originally conceived as a system for teaching deductive databases, it currently incorporates many extensions: SQL language to access relations shared with the rest of the languages it supports (Datalog, relational calculation of tuples - TRC: Tuple Relational Calculus - and domains - DRC: Domain Relational Calculus - and relational algebra -RA: Relational Algebra).

It is a system widely used internationally (around 84K downloads) in teaching (there are reports from up to 50 universities) and research for purposes such as the following: studies of security flaws in Firefox, quantitative decision making in software engineering, semantic web, etc. [9]

The ACIDE project provides a graphical, configurable, free, cross-platform integrated development environment that can be configured for use with any development system, such as interpreters, compilers and database systems.

Features of this system include: project management, multi-file editing, syntax colouring, console panel and database panel (tested with DB2, MySQL, Oracle, PostgreSQL, SQL Anywhere and DES).

End users who can benefit from this system include: console and database users, researchers developing programming systems and developers. It is implemented in Java to be platform independent.

This report describes the implementation of the Datalog graphical program debugging tool carried out at ACIDE as the main task in the completion of this thesis, detailing:

- The prior theory that must be known in order to successfully conduct a Datalog database debugging session.
- Use of the DES system with Datalog.

- The design of the Datalog debug panel: class diagram and graphical design.
- The process of a Datalog database debugging session.

In addition, the rest of the implementations carried out in ACIDE with the aim of improving the user experience are also described.

Sections 3 and 4 of this report provide details on the implemented functionalities.

At the end of the report, in the section "Appendix - ACIDE executable generation and start-up", there is an explanatory guide to the process of generating the application executable and its execution for future contributors to the ACIDE project.

## 2.1. State of the art of deductive databases

Research on deductive databases over the years has led to the development of systems capable of working with databases.

In particular, through automatic theorem proving and logic programming, it was possible to demonstrate that deduction can be carried out systematically in a database environment.

In the following sub-sections, some of the systems that exist today for working with deductive databases and the technologies that have been used to develop this work will be presented.

### 2.1.1. Work systems

The following table lists some of the systems developed so far that allow working with deductive databases [10,14] and briefly describes their most relevant characteristics.

System	Description
DEDUCE	Automatic deduction system implemented by IBM in the mid-1970s. Supports recursive rules.
Chat-80	Implemented in 1981 by Warren and Pereira. It allows queries to be made against a deductive database that stores information related to world geography.
Dahl's system	Implemented in 1982. Introduces the notions of set expressions and distributive

	and collective relations.
LDL Project	Its development started in 1984 outside the university environment and originated the LDL++ prototype whose development started in 1990 with the aim of solving the problems that the LDL system had in working with real applications. Among other things, the LDL++ system supports both stratified and unstratified denial, as well as aggregation.
NAIL	Its development started in 1985. Some of its features are that it supports layered negation, well-founded negation and modularly layered negation. It works with the GLUE language which allows loop, procedure and module constructs involving logical rules whose power is on par with that of SQL statements.
LOLA	It started to be developed in 1988. It supports, among other things, layered denial, aggregation and integrity constraint maintenance.
XSB	It was first launched in 1993. It is a faster system than others for performing queries and layered rule sets. Some of its features are that it supports well-founded negation, recursion and aggregation. In addition, it is cross-platform, as it can work on both Unix and Windows.
DES	It was first launched in 2004. The characteristics of the system have been described at the beginning of section 1 of this report.

### 2.1.2. Technologies used

The tools used to carry out the implementation of the tasks required in this work are described below:

- IntelliJ IDEA 2020.3.2:

This is an integrated development environment for software development.

Its usefulness in this work has been to serve as a tool for coding all the tasks required in this work in the ACIDE project. In addition, it has allowed the export of the executable of the desktop version of the application.

- ACIDE system version 0.18:

The system has been used to act as a graphical communication interface between the end user and the DES system.

- DES system version 6.7:

It has been the system of choice for the user to work with deductive databases by querying and/or debugging them.

## 2.2. Motivation

The absence of systems that allow the debugging of deductive databases motivated the inclusion of a graphical debugging interface in ACIDE.

Although the DES system offers a console interface to perform textual debugging of a deductive database where a Datalog program is loaded, carrying out the debugging process through this interface is a tedious and error-prone mechanism for the following reasons:

- The user conducting a debugging session must manually query and enter all commands and data required during the debugging process. This causes the debugging process to be lengthy.

In addition, during the process, erroneous data may be entered, for example, in the following situations:

- \* When specifying an erroneous tuple, you enter the wrong data for the tuple you want to mark as erroneous.

- \* You enter an action to perform the next step in debugging that does not exist.

- During the debugging process, the user does not have a clear view of the dependency between the elements to be debugged.
- The user cannot reverse decisions made. Therefore, if he performed an action that was not correct, such as marking a valid tuple as erroneous, to correct this error, he must start the debugging session again from scratch.
- During debugging, you cannot freely consult certain details such as the content of an element that is not currently being debugged, but that is part of the debugging (that has been previously debugged or is pending debugging).

Due to all the inconveniences that can arise during a textual debugging session, the need arises to implement a Datalog debugging panel in the ACIDE graphical interface that avoids user errors during the debugging process by providing advantages such as the following:

- At no time does the user have to enter data manually, but can select from the

suggested data. This prevents the user from entering erroneous data.

- The user will not have to worry about manually entering commands to perform the following steps, as when he/she presses the button associated to the action he/she wants to perform, the appropriate command will be automatically sent to the DES console.

This prevents the user from entering wrong commands or having to spend time searching for the right command.

- The user will at all times have a graph available that allows him to have a clear view of the dependencies between the elements to be debugged.

- It is possible to undo actions carried out during debugging. This avoids having to start debugging from scratch.

- You can consult at any time the content or status of any element involved in the debugging.

- You will be able to locate more easily in the Datalog program the definition of the elements to be debugged, as it will be highlighted.

## 2.3. Work background

This work is an evolution of the ACIDE project, which has always been directed by Fernando Sáenz Pérez and which, over the years, has been developed by the following contributors:

- During the academic year 2006-2007, Diego Cardiel Freire, Juan José Ortiz Sánchez and Delfín Rupérez Cañas were in charge of carrying out versions 0.1 to 0.6 of the project.

- During the 2007-2008 academic year, Miguel Martín Lázaro was in charge of the 0.7 version of the project.

- During the 2010-2011 academic year, Javier Salcedo Gómez was in charge of the 0.8 version of the project.

- During the 2012-2013 academic year, Pablo Gutiérrez García-Pardo, Elena Tejeiro Pérez de Ágreda and Andrés Vicente del Cura were in charge of versions 0.9 to 0.11 of the project.

- During the 2013-2014 academic year, Semiramis Gutiérrez Quintana, Juan Jesús Marqués Ortiz and Fernando Ordás Lorente were in charge of versions 0.12 to 0.16 of the project.

- During the 2014-2015 academic year, Sergio Domínguez Fuentes was in charge of the 0.17 version of the project.

- During the academic year 2019-2020, Sergio García Rodríguez was in charge of the 0.18 version of the project.



- During the 2020-2021 academic year, Carlos González Torres [4] and the author of this report (Cristina Lara López) were in charge of the 0.18.1 version of the project.

In this academic year 2021-2022, the author of this report (Cristina Lara López) has continued the development of the project from version 0.18.1. My contributions have generated version 0.19 of the ACIDE project.

## 2.4. Intended objectives

The objective of this final degree project was to implement the Datalog database debugging panel that was pending development at ACIDE and to improve some aspects of the interface:

### **Main development**

Using the available DES textual debugger as a basis, implement the Datalog database debugging panel with the appropriate commands and options.

### **Relevant improvements**

- Lock the debug panel when the ACIDE console is processing TAPI commands.
- Implement dynamic language switching in database, trace and debug panels.
- Improve the SQL debugger: provide more information about each node during debugging (tuples indicated as erroneous or missing), undo-redo chain, keyboard shortcuts, tuples count, etc.
- Add descriptive texts informing about the usefulness of the different components: buttons, boxes, etc.
- Make changes in the localisation to the different languages.
- Fix errors: error in the console during the start-up of ACIDE, error during file processing, etc.
- Enable scrolling through the content when the content is not displayed in its entirety in the available area.

## 2.5. Achieved goals

All the intended objectives described in section 1.4 of this report could be



implemented without relevant blockages:

## **Main development**

A Datalog debugging panel has been developed which is able to perform a complete debugging process of a Datalog database by including the following details:

- A selector that suggests a list of available elements to be debugged belonging to the Datalog program loaded in the database. The user must select the element to be debugged.
- A debug configuration window.
- A canvas on which the network with the nodes to be debugged is drawn, reflecting their dependencies on each other.
- Highlight in the program editor the lines that define the selected element in the network.
- A button to start a debugging session: opens a graphical window showing all the options available to perform the next step in the debugging.

Some of the information displayed in the graphical debug window is the following: tuples of the element being debugged and available actions that can be performed on it including an undo or redo action to return to a previous or later state in the debug, number indicating the number of tuples contained in the element, etc.

- A context menu available for each node of the network to be debugged that allows the following actions to be performed on it: consult its tuples, edit its content, apply actions to debug it, etc.

The details of the implementation of the Datalog debugging panel can be found in sections 3.2.4 and 3.2.5 of this report.

## **Relevant improvements**

Some of the global improvements implemented in the ACIDE graphical interface include the following:

- A corresponding implementation has been made to lock the debug panel when the DES console is processing TAPI commands.

Specifically, if the last line of the DES console consists of only the corresponding

DES inductor, it follows that the console is no longer processing commands and unlocks the debug panel. Otherwise, it locks the debug panel.

- Dynamic language switching has been implemented so that, without the need to restart ACIDE, the labels and descriptions of buttons and other elements in the database, trace and debug panels are updated to the new language applied.
- Descriptive texts have been added to inform about the usefulness of each button or box in the trace or debug panel when the mouse is positioned over those components.
- Fixed corrupted text characters displayed in different locations in the ACIDE GUI.
- Fixed a bug when starting ACIDE that caused ACIDE to manually restart its console.
- The problem with file processing (SQL script, Datalog program, etc.) has been corrected.
- Debug panel canvases have been improved to allow scrolling through the loaded graph when their dimensions are small.

Improvements implemented during a SQL debugging session are as follows:

- An "Undo" option has been added to allow a return to an earlier state in debugging.
- A "Redo" option has been added to return to a later state that was reached during the debugging process.
- A functionality has been implemented that provides additional debugging information about a debugged node (wrong tuples, missing tuples, etc.) when the user, during the debugging process, hovers the mouse over it.
- The SQL debug window has been improved so that the elements it contains fit its dimensions.
- The number of tuples contained in the view or table being debugged is displayed.
- Keyboard shortcuts have been configured for the "Undo" and "Redo" options.

The rest of the improvements required and implemented are detailed in section 4 of this report.

## 2.6. Workplan

All the work I have done in the ACIDE project has been carried out between the

previous academic year (2020-2021) and the current one (2021-2022).

The chronogram presented on the following page reflects the detail of the tasks that I have been carrying out to complete this work as well as the time interval in months that I dedicated to them. Improvement tasks in ACIDE are identified by "(M)".

Tasks	2020				2021												2022
	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J
ACIDE project structure study, DES manual and state of the art																	
Dynamic language change (M)																	
Adjust SQL statement to be edited (M)																	
Colour table nodes according to the applied configuration (M)																	
Indicate usefulness of elements (M)																	
Changes in localisation to different languages (M)																	
Correction of corrupted characters (M)																	
Debug panel lock (M)																	
Show language icons (M)																	
Show debugging information associated with a node (M)																	
Get formatted SQL statement from view (M)																	
Command send control (M)																	
Datalog language study																	
Datalog debug panel																	
Change icons (M)																	
Add undo-redo chain (M)																	
Fix for error when booting ACIDE (M)																	
Correction in file processing (M)																	
Canvas scrolling with reduced dimensions (M)																	
Dynamic SQL debugging window (M)																	
Tuple counting (M)																	
Keyboard shortcuts for debugging (M)																	

At the beginning of my contribution to the project in September 2020, its director gave me access to a shared folder on Google Drive [2] containing:

- The zipped version 0.18 of the ACIDE project that included multiple folders and files essential to be able to run the latest version of ACIDE as well as the source code on which I had to start working.
- A document with the tasks I had to perform in order to implement the required Datalog debug panel.
- A document prioritising the enhancement tasks I had to carry out.

Once I was provided with these accesses, between September and October 2020 I devoted myself to deploy the ACIDE project in my personal team to study the structure and functionality of version 0.18. In addition, I was dedicated to:

- 1) Studying the SQL sections of the DES manual [1] and the TFG memory of Sergio García Rodríguez [5] to know the procedure to follow when performing a SQL debugging session.
- 2) Study, group and summarise the work that had been carried out so far to work with deductive databases: environments, available debugging tools, etc.
- 3) Carry out different tests on the ACIDE graphical interface to study its behaviour.

Subsequently, between November 2020 and March 2021 I worked on implementing the requested improvements in the ACIDE interface related to the following aspects:

Improvements in the SQL trace and debugging panels and generic improvements such as correction of corrupted characters in the informative texts of the interface.

For each implemented enhancement or set of enhancements, an application executable was generated.

The newly generated application executable was uploaded to the shared Google

Drive [2] folder I was working with so that it could be tested by the director of the TFG and later validated.

Once I had finished implementing the previous improvements, before starting to implement the Datalog debugging panel, during the months of April, May and June 2021 I dedicated myself to study the Datalog language summarising the following concepts:

- 1) Basic concepts of the Datalog language. The information was obtained from the documentation available on the Internet [6].
- 2) Use of the DES system with the Datalog language. The information was obtained from the documentation provided by the director of the TFG [7]. In addition, while doing the abstract, I practised with the DES system by following the tutorials in the documentation.
- 3) Commands and options available to carry out a basic debugging session of Datalog programs. This information is reflected in detail in the sections of the DES system manual [1] that refer to debugging with the Datalog language.

In order to practice the concepts acquired, I was carrying out debugging sessions of Datalog programs in the DES console.

Subsequently, once all the concepts related to the Datalog language were structured, in July 2021 I started to develop the Datalog debugging panel and finished in January 2022. In parallel, I was implementing pending improvements affecting both SQL and Datalog functionalities.

During the testing process of the implemented functionalities, if an incorrect output was detected after the execution of commands through the console, the director of the TFG was notified so that an improved version of the DES system could be released. For the enhancements, the same validation procedure was applied as for the Datalog debugging panel.

All the implemented tasks are hosted in the following repository in my GitHub account that keeps track of the changes and implementations I have made until reaching the 0.19 version of the project:


<https://github.com/crislara/ACIDE>

At the end of March 2021, after having implemented several improvement tasks, an intermediate version of the ACIDE project was published: version 0.18.1. This version of the application is available in the download section of the ACIDE website [3]:

<http://www.fdi.ucm.es/profesor/fernan/ACIDE/html/download.html>

On the above web page you will find:

- The different published versions of ACIDE available for download.
- A guide indicating how to configure the downloaded version in the operating system of interest (Windows, Linux, etc.) as well as the necessary configuration to interpret the syntax of the language used to create the programmes to be worked with.
- A user manual for the DES system [1].
- A file "releasenotesACIDE.txt" describing the changes implemented in the latest published version of ACIDE.



- Home
- Download
- Screenshots
- Release Notes
- License
- News
- Contact
- Credits
- Publications
- Statistics
- Pre-built Bundles

## Download

**Warnings:**

This software comes with no warranty (see [License](#)) and it is in the development stage (alpha version) so that many expected functionalities will not work.

**Current Version:**

[ACIDE](#) version 0.18.1 (released on March, 28th, 2021)  
[Sources](#)  
[User Manual](#) (PDF)  
[Release Notes](#) (.txt)  
 Look also for [Pre-built Bundles](#)

**Getting Started:**

Download ACIDE alpha version from above, decompress it, and execute the jar file. As released, this system is preconfigured with a sample project for MS Windows and its command shell (`cmd.exe`).

Typically, to configure ACIDE to work with any given system:

**First:** Configure the shell (Configuration -> Shell) and select the executable of the system shell (interpreter, compiler, database, ...) (e.g., `C:\Windows\system32\cmd.exe`), its working directory (if any), the quit command (e.g., `exit`), and whether the shell echoes user inputs (as, e.g., for the Windows operating system shell).

**Second:** Configure the toolbar (Configuration -> Toolbar) for adding new commands that can be executed from the icons in the toolbar. Give a toolbar icon name (e.g., `type`), the command that will be executed (e.g., `type %activeFile%`, which stands for executing the command type followed by the name of the selected file in the multifile editor), a help text if desired (e.g., Command for typing the current file), a file image (if desired). The icons in a toolbar can be saved as a list (Save list) and loaded afterwards (Load list).

**Third:** Configure or load the (programming) language for syntax colouring (Configuration -> Lexicon), and file extension associations (e.g., `.bat`). Currently, parsing is included but it cannot be applied.

**Fourth:** Try to open a file and check whether the syntax is highlighted and whether you can submit your commands to the shell clicking your buttons (e.g., `type`).

More details can be found in the [User Manual](#) above.

**Previous Versions:**

[ACIDE](#) version 0.18 (released on January, 25th, 2021). [Sources](#)  
[ACIDE](#) version 0.17 (released on December, 26th, 2015). [Sources](#)  
[ACIDE](#) version 0.16 (released on August, 14th, 2014). [Sources](#)  
[ACIDE](#) version 0.15 (released on April, 29th, 2014). [Sources](#)  
[ACIDE](#) version 0.14 (released on March, 12th, 2014)  
[ACIDE](#) version 0.13 (released on February, 3rd, 2014). [Sources](#)  
[ACIDE](#) version 0.12 (released on December, 18th, 2013). [Sources](#)  
[ACIDE](#) version 0.11 (released on July, 8th, 2013). [Sources](#)  
[ACIDE](#) version 0.10 (released on April, 24th, 2013). [Sources](#)  
[ACIDE](#) version 0.9 (released on December, 20th, 2012)  
[ACIDE](#) version 0.8 (released on July, 8th, 2011). [Sources](#)  
[ACIDE](#) version 0.7 (released on November, 5th, 2008). [Sources](#)  
[ACIDE](#) version 0.6 (released on August, 20th, 2007). [Sources](#)  
[ACIDE](#) version 0.5 (released on June, 4th, 2007)  
[ACIDE](#) version 0.4 (released on May, 18th, 2007)  
[ACIDE](#) version 0.3 (released on April, 25th, 2007)  
[ACIDE](#) version 0.2 (released on April, 16th, 2007)  
[ACIDE](#) version 0.1 (released on April, 9th, 2007)  
[ACIDE](#) version 0.0 (launched on March, 2007)

Figura I-1 List of published versions



## 3. DEPURACIÓN DE DATALOG

---

### 3.1. Introducción a Datalog

Antes de poder comprender el proceso que se sigue para realizar una sesión depuración Datalog así como todos los elementos que intervienen, se debe tener una base del lenguaje Datalog. Por este motivo, el objetivo de esta sección es proporcionar al lector unos conocimientos básicos del lenguaje Datalog.

#### 3.1.1. Bases de datos deductivas

Las bases de datos deductivas incrementan la capacidad expresiva de las bases de datos relacionales, ya que tienen la capacidad de, a través de reglas, deducir información a partir de información que ya hay almacenada de forma explícita en la base de datos. Por ejemplo:

Partiendo de que se sabe que María es madre de Juan y Pedro se deduce que Juan y Pedro son hermanos, ya que ambos provienen de la misma madre.

#### 3.1.2. Lenguaje Datalog

Datalog es un lenguaje de consulta de bases de datos deductivas basado en el lenguaje de programación lógica Prolog.

Una regla en Datalog tiene la siguiente estructura:

**cabeza :- cuerpo.**

Donde:

- **"cabeza"**: compuesta de un solo átomo.
- **"cuerpo"**: compuesto de ninguno, uno o varios literales. Un literal puede ser un átomo, un átomo negado o una metallamada. Si el cuerpo de una regla no está compuesto de ningún literal, la regla será un "hecho".

Toda regla siempre debe terminar en punto "." para indicar el fin de su definición.

La definición de un hecho es la siguiente:

$$p(T_1, \dots, T_n).$$

Donde:

- "p": es el símbolo que hace referencia al nombre del predicado.
- "T<sub>i</sub>": es el argumento n° "i" que recibe. Dicho argumento puede ser una variable o constante. Las variables comienzan por mayúscula.

Luego un hecho es una regla sin cuerpo.

Un conjunto de reglas define un predicado.

La aridad de un predicado es la cantidad de argumentos que recibe. La sintaxis que se utiliza para indicar la aridad de un predicado es la siguiente:

$$\langle \text{nombre\_predicado} \rangle / \langle \text{aridad} \rangle$$

Por ejemplo, la aridad del predicado "madre(X,Y)" es 2, ya que recibe dos argumentos (X e Y) -> madre/2

### 3.1.3. Reglas

Como se indicó en el apartado 3.1.2, una regla en Datalog tiene la siguiente estructura:

$$p \text{ :- } q_1, \dots, q_n.$$

Donde:

- "p": es el nombre del predicado que compone la cabeza de la regla.
- "q<sub>1</sub>, ..., q<sub>n</sub>": es un listado de literales que componen el cuerpo de la regla. Los literales del cuerpo de una regla pueden estar negados como luego se explicará en el apartado 3.1.4. Si n=0, la lista de literales estará vacía y, por tanto, "p" será un hecho.

Un predicado contiene tuplas de datos que deben cumplir la relación definida para el mismo. Los argumentos que se pasan al predicado siguen un orden e indican cómo se relacionan los datos. Si los datos de todas las tuplas de un predicado

respetan la relación, se considerará que ese predicado es cierto/válido.

Por tanto, si todos los literales del cuerpo de una regla son ciertos, es decir, almacenan tuplas de datos correctas, entonces, el predicado de la cabeza de la regla será cierto.

A continuación, mediante un ejemplo, se muestra cómo a partir de una regla se deduce información:

Supongamos que se tiene definido el predicado "madre(X,Y)" que significa que "X" es madre de "Y".

Además, el predicado "madre(X,Y)" está compuesto de las siguientes tuplas:

- Una tupla que indica que "María" es madre de "Juan" -> madre(María,Juan).
- Una tupla que indica que "María" es madre de "Pedro" -> madre(María, Pedro).

La información de dichas tuplas es información almacenada explícitamente en la base de datos.

Después, se define la regla "hermano(Y,Z) :- madre(X,Y), madre(X,Z)". El predicado "hermano(Y,Z)" significa que "Y" es hermano de "Z".

La regla anterior deduce la siguiente información a partir de las tuplas del predicado "madre(X,Y)":

Si la persona "X" es madre de la persona "Y" y también de la persona "Z", entonces, se deduce implícitamente que la persona "Y" es hermano de la persona "Z".

Por tanto, haciendo uso de la regla anterior, se deduce implícitamente que si María es madre de Juan y María es madre de Pedro, entonces, Juan es hermano de Pedro.

Por último, en Datalog se distinguen dos tipos de reglas:

- Reglas extensionales: aquellas que no tienen cuerpo (hechos).
- Reglas intensionales: aquellas compuestas de cabeza y cuerpo.

### 3.1.4. Negación

A veces puede resultar útil expresar una negación en el cuerpo de una regla. Para expresar una negación en el cuerpo de una regla se utiliza el operador "not". En concreto, lo que se niega en el cuerpo de una regla es un átomo del cuerpo al que precede el operador not como prefijo para negarlo. Por ejemplo:

Si se quiere negar el átomo "madre(X,Y)", le debe preceder el operador "not" como prefijo -> not madre(X,Y)

### 3.1.5. Predicados intensionales y extensionales

En Datalog, un predicado es:

- Extensional si sus reglas son hechos.

Por tanto, para definir un predicado extensional previamente se han de haber definido los hechos como reglas sin cuerpo.

- Intensional si incluye reglas que no sean hechos.

La definición de un predicado intensional viene dada por una regla o conjunto de reglas. Por ejemplo:

- El predicado "madre(X,Y)" es extensional, ya que previamente se han definido hechos como los siguientes: Madre(María,Juan)., etc.
- El predicado "hermano(Y,Z)" es intensional, ya que es definido a partir de la siguiente regla que no es un hecho al disponer de cuerpo:

$$\text{hermano}(Y,Z) \text{ :- madre}(X,Y), \text{madre}(X,Z).$$

### 3.1.6. Grafo de dependencias

Todo predicado (y su contenido (tuplas)) que se infiere a partir de una o varias reglas y que, por tanto, es un predicado intensional, dependen de otros predicados. Sin embargo, un predicado extensional no depende de otros predicados.

Para construir el grafo de dependencias (PDG: Predicate Dependency Graph) de un predicado intensional "p" se realiza el siguiente procedimiento:

1) El grafo estará formado por los siguientes nodos:

- Un nodo que representa al predicado intensional "p".
- Un nodo por cada literal distinto que hay en el cuerpo de la regla o conjunto de reglas que infieren ese predicado intensional "p".

2) Se dibujará una arista del átomo "q" al predicado intensional "p" si el átomo "q" aparece en el cuerpo de alguna de las reglas que infieren al predicado intensional "p". Por tanto, saldrá una arista de "q" a "p" si existe una regla del siguiente tipo:

$$p :- a, b, c, \dots, q, \dots$$

3) Se dibujará una arista que sale del predicado intensional "p" y vuelve a entrar al mismo cuando existe una regla donde "p" aparece tanto en su cabeza como en su cuerpo. Por tanto, una misma arista saldrá del predicado "p" y volverá a entrar a él si existe una regla del siguiente tipo:

$$p :- a, b, c, p, \dots, q, \dots$$

En esta situación, se dice que el predicado "p" es recursivo.

Además, el sistema se encargará de calcular el cierre transitivo de la relación que implementa el PDG. El concepto de transitividad se refiere a que, si existe un camino que une el nodo "p" con el nodo "q", no importará que para llegar desde "q" hasta "p" haya que visitar otros nodos del grafo.

A continuación, mediante el siguiente ejemplo, se mostrará cómo se construye un grafo de dependencias de un programa Datalog:

Supongamos el siguiente programa Datalog "orbits.dl" con la siguiente definición de hechos y reglas intensionales:

```
star(sun) .
orbits(earth, sun) .
% Primera regla
orbits(moon, earth) .
orbits(X, Y) :-
    orbits(X, Z) ,
    orbits(Z, Y) .
% Segunda regla
```

```

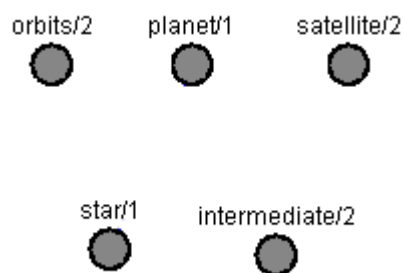
satellite(X,Y) :-
    orbits(X,Y),
    not(intermediate(X,Y)),
    not(star(Y)).
% Tercera regla
planet(X) :-
    orbits(X,Y),
    star(Y),
    not(intermediate(X,Y)).
% Cuarta regla
intermediate(X,Y) :-
    orbits(X,Z),
    orbits(Z,Y).

```

A continuación se explica el procedimiento que el sistema realiza para construir el grafo de dependencias de predicados a partir de los hechos y reglas intensionales definidos en el programa:

1) Identifica los diferentes predicados tanto extensionales como intensionales que aparecen en el programa, ya que cada uno de ellos será representado por un nodo distinto en el grafo:

- star
- orbits
- satellite
- planet
- intermediate



*Figura D-1 Identificación de los nodos*

2) En la primera regla (`orbits(X,Y) :- orbits(X,Z),orbits(Z,Y).`), como "orbits" está tanto en la cabeza como en el cuerpo de la regla, se traza una arista que salga y entre al mismo. En esta situación, queda reflejado que "orbits" es recursivo.

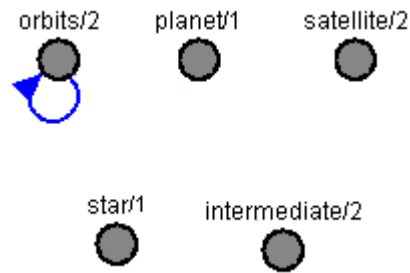


Figura D-2 Arista para una dependencia recursiva

3) En la tercera regla (`planet(X) :- orbits(X,Y), star(Y), not intermediate(X,Y).`), como "orbits" y "star" se encuentran en el cuerpo de la regla sin negar y "planet" en la cabeza, se trazan dos aristas de "orbits" y "star" a "planet" para representar la dependencia positiva (arista azul) que existe por no haber negación.

La misma arista de dependencia positiva traza de "orbits" a "intermediate" por encontrarse sin negar en el cuerpo de la cuarta regla que define "intermediate":

`intermediate(X,Y) :- orbits(X,Z), orbits(Z,Y).`

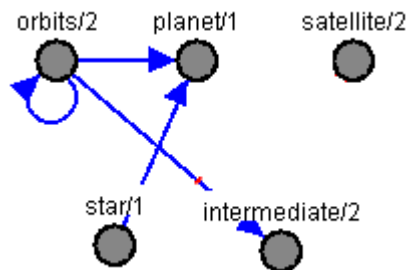


Figura D-3 Aristas para dependencias positivas

4) En la segunda regla (`satellite(X,Y) :- orbits(X,Y), not intermediate(X,Y), not star(Y).`), como tanto "intermediate" como "star" están en el cuerpo de la regla negados y "satellite" en la cabeza de la regla que lo define, traza dos aristas de "intermediate" y "star" a "satellite" que reflejan la dependencia negativa que tienen (arista roja) debido a la negación.

Traza la misma arista de dependencia negativa de "intermediate" a "planet" por encontrarse negado en el cuerpo de la tercera regla que define "planet":

$\text{planet}(X) \text{ :- orbits}(X,Y), \text{star}(Y), \text{not intermediate}(X,Y).$

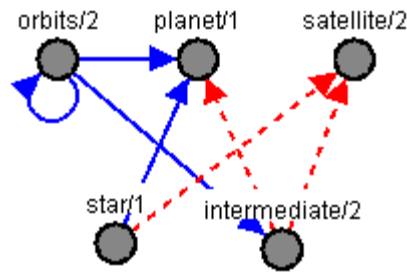


Figura D-4 Aristas para dependencias negativas

5) Por último, el sistema traza una arista de dependencia positiva de "orbits" a "satellite" (arista azul) debido a que "orbits" (sin negarse) interviene en la definición de "satellite" en la cuarta regla.

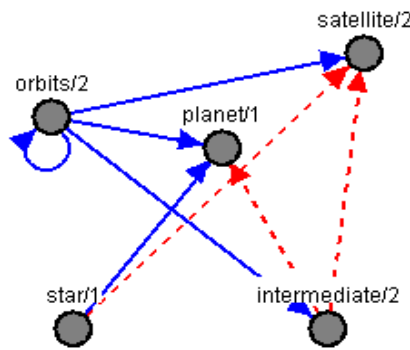


Figura D-5 Arista final del PDG

Con este quinto paso se ha terminado de crear el grafo de dependencias entre predicados.

Luego, por todo lo explicado hasta el momento, el estado de una base de datos deductiva viene definido por:

- Un conjunto de hechos.
- Reglas intensionales que infieren información.

### 3.1.7. Consultas

La siguiente expresión permite consultar las tuplas de un predicado cargado en la base de datos deductiva del sistema DES:



$$p(X_1, \dots, X_n)$$

Como se ve en la expresión anterior, para realizar la consulta, se debe introducir el nombre del predicado seguido de tantas variables como su aridad.

Las variables pueden ser sustituidas por constantes si lo que interesa es obtener aquellas tuplas del predicado que en esa posición contienen el mismo dato constante.

Cada vez que el usuario realiza una llamada a un predicado, DES resuelve la consulta usando las reglas cargadas en la BD y aplicando el método de resolución. De este modo, se obtiene como salida las tuplas del predicado consultado.

Los resultados de las resoluciones se almacenan en una tabla extensional (ET: Extension Table).

### 3.1.8. Comandos básicos

A continuación se describen los comandos básicos que hay que conocer para poder trabajar con un programa Datalog en el sistema DES:

Comando	Utilidad
<p><b>/help</b></p> <p><b>/help comando</b></p>	<p>Permite consultar todos los comandos que se pueden usar en la consola DES. Los agrupa/clasifica por categorías.</p> <p>Se debe introducir el número de la categoría para consultar sus comandos.</p> <p>Junto al comando /help también se puede especificar un comando para conocer sus diferentes opciones de uso.</p>
<p><b>/consult nombreProgramaDatalog</b></p> <p><b>/c nombreProgramaDatalog</b></p>	<p>Carga en la base de datos deductiva del sistema DES el programa Datalog especificado (incluyendo hechos y reglas intensionales).</p> <p>Antes de realizar la carga, elimina toda la información previa que ya hubiera cargada en la base de datos.</p> <p>La extensión ".dl" de los programas Datalog</p>

	puede ser omitida a la hora de especificar el programa Datalog que se quiere cargar.
<b>/list_et</b> <b>/list_et nombrePredicado</b> <b>/list_et nombrePredicado/aridad</b>	<p>Permite consultar las tuplas resultado de la última resolución. Se mostrará la información almacenada hasta el momento en la tabla extensional.</p> <p>Junto al comando /list_et también se puede especificar el nombre de un predicado (y opcionalmente su aridad) consultado previamente.</p> <p>De este modo, únicamente se obtiene la información de la tabla extensional relacionada con dicho predicado.</p>
<b>/clear_et</b>	Permite eliminar todas las tuplas resultado de la última resolución.
<b>/assert regla</b>	Permite añadir a la base de datos deductiva del sistema DES la regla especificada junto al comando "/assert".
<b>/retract regla</b>	Permite eliminar de la base de datos deductiva del sistema DES la regla especificada junto al comando "/retract".
<b>/abolish</b> <b>/abolish nombrePredicado</b> <b>/abolish nombrePredicado/aridad</b>	<p>Elimina todas las reglas (sean extensionales o intensionales) que han sido cargadas en la base de datos deductiva del sistema DES.</p> <p>Adicionalmente, borra todo el contenido de la tabla de extensión almacenado hasta el momento.</p> <p>Junto al comando /abolish se puede especificar el nombre del predicado (y opcionalmente su aridad) cuyas reglas se desean eliminar de la base de datos.</p> <p>Adicionalmente, borra el contenido de la tabla de extensión almacenado hasta el momento relacionado con el predicado especificado.</p>
<b>/listing</b> <b>/listing nombrePredicado</b> <b>/listing nombrePredicado/Aridad</b>	Muestra las reglas (sean extensionales o intensionales) almacenadas en la base de datos deductiva del sistema DES que definen los predicados.

	<p> Junto al comando /listing se puede especificar el nombre del predicado (y opcionalmente su aridad) para consultar su información: tuplas y reglas que lo definen.</p>
--	---

A continuación, mediante un ejemplo, se mostrará el modo de uso y utilidad de algunos de los comandos mencionados en este apartado:

Se desea cargar el programa Datalog "family.dl" en la base de datos deductiva del sistema DES. El código del programa Datalog "family.dl" es el siguiente:

```
%
% Family
%
% Datalog Formulation

% Optionally declare types
:-type(father(father:string,child:string)).
father(tom,amy).
father(jack,fred).
father(tony,carolII).
father(fred,carolIII).

% Optionally declare types
:-type(mother(mother:string,child:string)).
mother(grace,amy).
mother(amy,fred).
mother(carolI,carolII).
mother(carolII,carolIII).

parent(X,Y) :-
    father(X,Y)
    ;
    mother(X,Y).
% The above clause for parent is equivalent to:
% parent(X,Y) :-
%     father(X,Y).
% parent(X,Y) :-
%     mother(X,Y).

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

La interpretación del código del programa Datalog es la siguiente:

Los comentarios realizados en el programa van precedidos del carácter "%".

Aunque en Datalog los datos con los que se trabaja no deben ser obligatoriamente de un tipo concreto (int, String, etc.), opcionalmente se ha declarado de qué tipo es cada argumento que recibe una tupla tanto del predicado extensional "mother" como del predicado extensional "father" utilizando la siguiente especificación:

`:-type(nombrePredicado(argumento1:tipo,argumento2:tipo,...,argumenton:tipo)).`

Por tanto, debido a la especificación establecida, el primer y segundo argumento que recibe tanto el predicado "mother" como el predicado "father" serán de tipo String.

El predicado "father(X,Y)" establece la siguiente relación: "X" es padre de "Y".

En concreto, las tuplas del predicado "father(X,Y)" indican que:

- "fred" es padre de "carolIII".
- "jack" es padre de "fred".
- "tom" es padre de "amy".
- "tony" es padre de "carolIII".

El predicado "mother(X,Y)" establece la siguiente relación: "X" es madre de "Y".

En concreto, las tuplas del predicado "mother(X,Y)" indican que:

- "amy" es madre de "fred".
- "carolI" es madre de "carolII".
- "carolII" es madre de "carolIII".
- "grace" es madre de "amy".

El predicado "parent(X,Y)" establece la siguiente relación: "X" es padre o madre de "Y". Las tuplas del predicado "parent(X,Y)" se infieren a partir de las siguientes reglas definidas de forma abreviada utilizando la disyunción lógica (;):

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

Sin hacer uso de la disyunción lógica para abreviar, la definición de las reglas sería la siguiente:

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

El predicado "ancestor(X,Y)" establece las siguientes relaciones: "X" es antepasado de "Y" si "X" es padre o madre de "Y" o "X" es padre o madre de "Z" y "Z" es un antepasado de "Y".

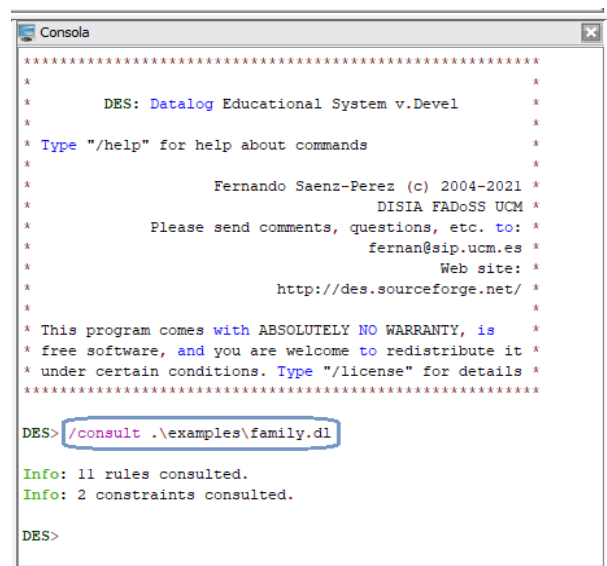
El predicado "ancestor" es recursivo, ya que en una de sus reglas (ancestor(X,Y) :- parent(X,Z),ancestor(Z,Y).) se invoca a sí mismo.

Las tuplas del predicado "ancestor(X,Y)" se infieren a partir de las siguientes dos reglas:

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

Asumiendo que el programa Datalog se encuentra almacenado en la carpeta "examples" que hay en la raíz del proyecto ACIDE, para realizar la carga del programa Datalog en la base de datos deductiva del sistema DES desde la consola de ACIDE, se debe introducir el siguiente comando:

```
/consult .\examples\family.dl
```



```
*****  
*  
*      DES: Datalog Educational System v.Devel  
*  
* Type "/help" for help about commands  
*  
*      Fernando Saenz-Perez (c) 2004-2021  
*      DISIA FADoSS UCM  
*      Please send comments, questions, etc. to:  
*      ferman@sip.ucm.es  
*      Web site:  
*      http://des.sourceforge.net/  
*  
* This program comes with ABSOLUTELY NO WARRANTY, is  
* free software, and you are welcome to redistribute it  
* under certain conditions. Type "/license" for details  
*****  
  
DES> /consult .\examples\family.dl  
  
Info: 11 rules consulted.  
Info: 2 constraints consulted.  
  
DES>
```

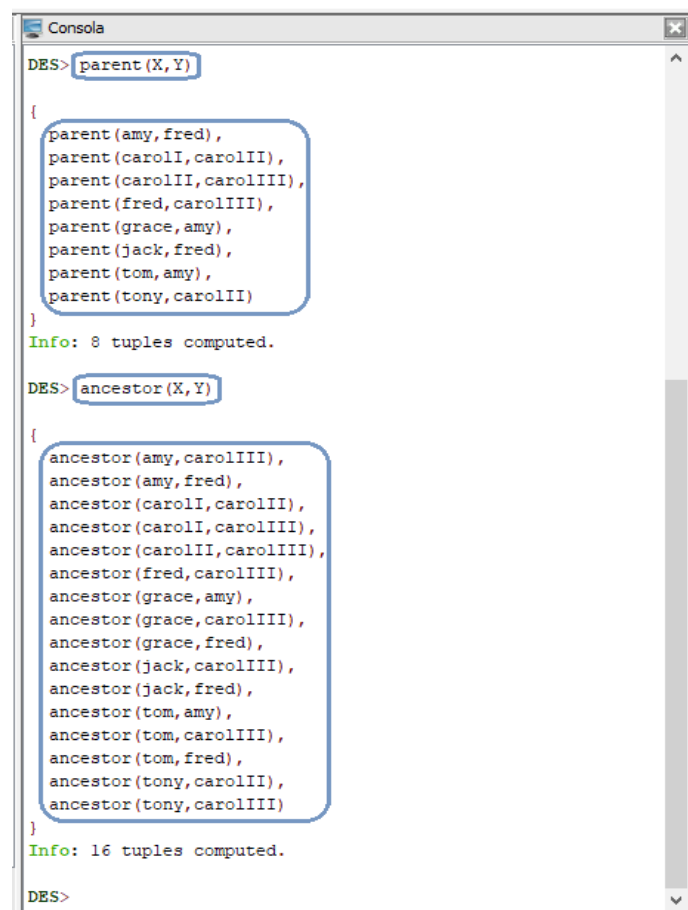
Figura D-6 Carga de un programa Datalog

Para consultar las tuplas de los predicados "parent(X,Y)" y "ancestor(X,Y)" se deben introducir las siguientes llamadas por la consola de ACIDE:

parent(X,Y)  
ancestor(X,Y)

Como muestra la siguiente imagen, tras introducir las dos llamadas anteriores, se obtienen todas las personas tanto del sexo femenino como masculino que tienen hijos así como todos los antepasados de una persona. Por ejemplo:

Un antepasado de la persona llamada "fred" es "grace" (ancestor(grace,fred)), ya que "grace" dio a luz a "amy" (parent(grace,amy)) y "amy" tuvo a "fred" (parent(amy,fred)).



```
Consola
DES> parent(X,Y)
{
  parent(amy,fred),
  parent(carolI,carolIII),
  parent(carolIII,carolIII),
  parent(fred,carolIII),
  parent(grace,amy),
  parent(jack,fred),
  parent(tom,amy),
  parent(tony,carolIII)
}
Info: 8 tuples computed.
DES> ancestor(X,Y)
{
  ancestor(amy,carolIII),
  ancestor(amy,fred),
  ancestor(carolI,carolIII),
  ancestor(carolI,carolIII),
  ancestor(carolII,carolIII),
  ancestor(carolIII,carolIII),
  ancestor(fred,carolIII),
  ancestor(grace,amy),
  ancestor(grace,carolIII),
  ancestor(grace,fred),
  ancestor(jack,carolIII),
  ancestor(jack,fred),
  ancestor(tom,amy),
  ancestor(tom,carolIII),
  ancestor(tom,fred),
  ancestor(tony,carolIII),
  ancestor(tony,carolIII)
}
Info: 16 tuples computed.
DES>
```

Figura D-7 Consulta de tuplas

Adicionalmente, si se quiere realizar una consulta más específica, será necesario sustituir ciertas variables de la consulta por constantes. Por ejemplo, si se quieren consultar todos los antepasados de la persona llamada "fred", se debe introducir la

siguiente llamada por la consola DES:

ancestor(X,fred)

Como se puede observar en la llamada anterior, la variable "Y" ha sido sustituida por la constante "fred". De este modo, tal y como muestra la siguiente imagen obtenemos que los antepasados de "fred" son: amy, grace, jack y tom.

```
DES> ancestor(X,fred)

{
  ancestor(amy,fred),
  ancestor(grace,fred),
  ancestor(jack,fred),
  ancestor(tom,fred)
}
Info: 4 tuples computed.

DES>
```

*Figura D-8 Consulta específica*

Para consultar las reglas (tanto extensionales como intensionales) que definen a los predicados almacenadas en la base de datos del sistema DES se debe introducir el siguiente comando:

/listing

```
DES> /listing

ancestor(X,Y) :-
  parent(X,Z),
  ancestor(Z,Y).
ancestor(X,Y) :-
  parent(X,Y).
father(fred,carolIII).
father(jack,fred).
father(tom,amy).
father(tony,carolII).
mother(amy,fred).
mother(carolI,carolII).
mother(carolII,carolIII).
mother(grace,amy).
parent(X,Y) :-
  father(X,Y)
;
mother(X,Y).

Info: 11 rules listed.

DES>
```

*Figura D-9 Consulta reglas cargadas en BD*

Para eliminar todas las reglas del predicado `ancestor(X,Y)` de la base de datos así como su información almacenada hasta el momento en la tabla extensional, se debe introducir el siguiente comando:

`/abolish ancestor`

Si tras introducir el comando anterior se listan de nuevo las reglas almacenadas en la base de datos del sistema DES introduciendo el comando `/listing`, se puede apreciar como han sido eliminadas de la base de datos las reglas que definían el predicado `ancestor`.

```
DES> /abolish ancestor

DES> /listing

father(fred,carolIII).
father(jack,fred).
father(tom,amy).
father(tony,carolII).
mother(amy,fred).
mother(carolI,carolIII).
mother(carolII,carolIII).
mother(grace,amy).
parent(X,Y) :-
    father(X,Y)
;
    mother(X,Y).

Info: 9 rules listed.

DES> |
```

*Figura D-10 Información eliminada de la BD*

Como muestra la imagen anterior, las siguientes reglas ya no se encuentran almacenadas en la base de datos del sistema DES:

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

## 3.2. Depuración Datalog en ACIDE

### 3.2.1. Interfaz TAPI para la depuración

DES proporciona una API textual (TAPI: Textual Application Programming Interface) para que aplicaciones externas se puedan comunicar con el sistema DES.

Por tanto, el depurador completo de Datalog que ofrece el sistema DES se puede automatizar y conectar a otras interfaces a través de la API textual.



Esta API textual es la que está permitiendo a los usuarios interactuar con el sistema DES a través de la aplicación ACIDE.

Cuando una aplicación externa se deba comunicar con DES, los comandos que se envíen para la comunicación deben ir precedidos por el comando `/tapi`. Por ejemplo:

`/tapi /listing`

La salida obtenida como respuesta tras emitir comandos a la interfaz textual API se encontrará formateada para facilitar su manipulación. Las posibles respuestas a comandos emitidos precedidos del comando `"/tapi"` son las siguientes:

- Respuesta informando del éxito de la ejecución del comando:

`$success`

- Respuesta informando del error que se ha producido al intentar ejecutar el comando:

`$error`

`código`

`texto`

`...`

`texto`

`$eot`

Donde:

- `$error`: indica el inicio de la transmisión de datos que informan del error.
- `código`: es el código del error.
- `texto`: es el texto que describe el error que se ha producido.
- `$eot`: indica el fin de la transmisión de datos.

Las siguientes secciones explican la interfaz TAPI para los comandos del depurador Datalog.

### 3.2.2. Opciones para la depuración

A continuación se listan las posibles opciones que se pueden pasar como argumentos a determinados comandos de la depuración Datalog para cumplir distintos fines así como su sintaxis:

- Estados de un nodo: cada nodo en el PDG puede recibir uno de los siguientes estados:

Status :=  
valid                   |  
nonvalid               |  
erroneous

Luego los posibles estados son: valid, nonvalid o erroneous.

El usuario no podrá asignar el estado erroneous a un nodo pero sí el sistema.

- Definición de tuplas: las tuplas que especifica el usuario deben seguir la siguiente sintaxis:

Tuple :=  
Relation(Value<sub>1</sub>,...,Value<sub>n</sub>)

Donde:

- Relation: el nombre del predicado.
- Value<sub>1</sub>,...,Value<sub>n</sub>: listado de valores que recibe como argumento el predicado y que conforman una tupla del mismo.

Si el nombre de la relación (del predicado) contiene espacios en blanco o comienza con una letra que no es minúscula, deberá ser encerrada entre comillas simples.

- Respuestas del usuario: las posibles respuestas que puede dar el usuario ante una pregunta que le esté realizando el depurador durante la sesión de depuración, son las siguientes:

Answer :=  
abort                               |  
valid                               |  
nonvalid                           |  
missing(PartialTuple)           |

`wrong`(TotalTuple)

Luego las posibles respuestas son:

- `abort`: provoca que la sesión finalice.
- `valid`
- `nonvalid`
- `missing`(PartialTuple)

“PartialTuple” es la tupla que se quiere marcar como ausente en el predicado que actualmente se está depurando. La sintaxis de “PartialTuple” es la siguiente:

$$\text{Relation}(\text{Value}_1, \dots, \text{Value}_n)$$

Donde:

- `Relation`: el nombre del predicado con una tupla faltante que se está depurando actualmente.
- `Value1, ..., Valuen`: son los valores que debe contener la tupla faltante en el predicado “Relation”.

Si uno de los valores que componen la tupla faltante no debe tomar un valor concreto, en dicho argumento, se debe establecer un guion bajo (`_`). Por ejemplo:

Si se establece la opción “`missing(t(_a))`” se estará indicando que al predicado “`t`” le falta una tupla que tenga como segundo valor “`a`” pero como primer valor cualquiera.

- `wrong`(TotalTuple)

“TotalTuple” es la tupla que se quiere marcar como errónea en el predicado que actualmente se está depurando. La sintaxis de “TotalTuple” es la siguiente:

$$\text{Relation}(\text{Value}_1, \dots, \text{Value}_n)$$

Donde:

- `Relation`: el nombre del predicado con una tupla errónea que se está depurando actualmente.
- `Value1, ..., Valuen`: son los valores que componen la tupla errónea en el

predicado "Relation".

A continuación se listan las distintas opciones que puede utilizar el depurador para comunicarse con el usuario durante una sesión de depuración Datalog:

- Preguntas al usuario: las posibles preguntas que el depurador puede realizar al usuario durante una sesión de depuración Datalog son las siguientes:

Question :=

<code>all</code> (NombrePredicado/Aridad)	
<code>subset</code> (NombrePredicado1/Aridad,NombrePredicado2/Aridad)	
<code>empty</code> (NombrePredicado/Aridad)	
<code>nonempty</code> (NombrePredicado/Aridad)	

Donde:

- NombrePredicado: es el nombre del predicado que se desea depurar.
- Aridad: es un número que indica la cantidad de argumentos que recibe el predicado a depurar.

Luego las posibles preguntas que puede realizar el depurador son:

- `all`(NombrePredicado/Aridad)

Pregunta si todas las tuplas que contiene el predicado "NombrePredicado" son las que debe contener.

- `subset`(NombrePredicado1/Aridad,NombrePredicado2/Aridad)

Pregunta si todas las tuplas del predicado "NombrePredicado1" deben pertenecer al predicado "nombrePredicado2".

- `empty`(NombrePredicado/Aridad)

Pregunta si el predicado "NombrePredicado" debe estar vacío, es decir, si no debe contener ninguna tupla.

- `nonempty`(NombrePredicado/Aridad)

Pregunta si el predicado "NombrePredicado" debe contener al menos una tupla.

En lo que se refiere a las preguntas que realiza el depurador durante una sesión de depuración y las respuestas del usuario a las preguntas del depurador es relevante saber que cualquier pregunta de las anteriores admite las siguientes respuestas: “abort”, “valid”, “nonvalid”, “missing(PartialTuple)” o “wrong(TotalTuple)”.

### 3.2.3. Comandos para la depuración

A continuación se describen los comandos básicos que hay que conocer para poder llevar a cabo una sesión de depuración de un programa Datalog:

Comando	Utilidad
<b>/debug_dl nombrePredicado/Aridad</b> Opciones	<p>Permite iniciar una sesión de depuración para el predicado "nombrePredicado/Aridad".</p> <p>A continuación se detalla el significado de los argumentos que recibe el comando:</p> <ul style="list-style-type: none"> <li>– nombrePredicado: es el nombre del predicado que se desea depurar.</li> <li>– Aridad: es un número que indica la cantidad de argumentos que recibe el predicado a depurar.</li> <li>– Opciones: permite establecer un listado de las siguientes opciones separadas por espacios:</li> </ul> <p>+ <b>trust_extension</b>([yes no]): si se acompaña el comando junto a la opción "trust_extension(yes)", durante la depuración se confiará en los predicados extensionales. Por defecto, el depurador no confía en los predicados extensionales.</p> <p>+ <b>file</b>(File): dentro de los paréntesis de la opción "file" se debe poner entre comillas simples el nombre del fichero con extensión ".dl" que contienen el programa Datalog a depurar.</p> <p>+ <b>respuesta</b>: permite de antemano dar la respuesta a la primera pregunta que realizará el depurador durante la sesión.</p> <p>Las posibles respuestas son las siguientes: abort, valid, nonvalid, missing(PartialTuple) o wrong(TotalTuple).</p> <p>Si se utiliza la interfaz TAPI para realizar la sesión de</p>

	<p>depuración, es decir, si emite el comando <code>/debug_dl</code> precedido del comando <code>/tapi</code> y la ejecución del comando es exitosa, la respuesta tendrá el siguiente formato:</p> <pre> NodeName State ... NodeName State \$eot </pre> <p>Donde:</p> <ul style="list-style-type: none"> <li>– <code>NodeName</code>: hace referencia al nombre de uno de los predicados que ya han sido depurados.</li> <li>– <code>State</code>: indica el estado del nodo "<code>NodeName</code>" que le precede tras haber sido depurado.</li> <li>– <code>\$eot</code>: indica el fin de la transmisión de datos.</li> </ul> <p>En caso de error a la hora de iniciar la sesión de depuración, la respuesta de error tendrá el formato especificado para la misma en el apartado 3.2.1.</p>
<p><b><code>/debug_dl_set_node</code></b>  <b>nombrePredicado/Aridad Estado</b></p>	<p>En cualquier momento, es posible etiquetar/marcar cualquier nodo del grafo de dependencias de predicados con el estado "valid", "nonvalid", "missing(PartialTuple)" o "wrong(TotalTuple)".</p> <p>Establecer estados en los diferentes nodos acelera la depuración porque, durante la sesión, el depurador no se parará a analizar dichos nodos, ya que se conoce su estado.</p> <p>A continuación se detalla el significado de los argumentos que recibe el comando:</p> <ul style="list-style-type: none"> <li>– <code>nombrePredicado/Aridad</code>: es el nombre del predicado cuyo estado se desea establecer acompañado de la cantidad de argumentos que recibe.</li> <li>– <code>Estado</code>: es el estado que se quiere establecer para el predicado "nombrePredicado". Se puede establecer alguno de los siguientes estados: "valid", "nonvalid", "missing(PartialTuple)" o "wrong(TotalTuple)".</li> </ul>
<p><b><code>/debug_dl_current_question</code></b></p>	<p>Permite obtener la pregunta actual que el depurador está realizando al usuario.</p> <p>Tal y como se indicó en el apartado 3.2.2, el depurador</p>

	<p>puede realizar los siguientes tipos de preguntas al usuario:</p> <ol style="list-style-type: none"> <li>1) <b>all</b>(nombrePredicado/Aridad)</li> <li>2) <b>subset</b>(nombrePredicado1/Aridad,nombrePredicado2/Aridad)</li> <li>3) <b>empty</b>(nombrePredicado/Aridad)</li> <li>4) <b>nonempty</b>(nombrePredicado/Aridad)</li> </ol> <p>Por ejemplo, si se envía el comando <code>/debug_dl_current_question</code> a la consola DES y como salida se muestra la pregunta <code>"all(t/2)"</code>, el depurador estaría preguntando al usuario si todas las tuplas que contiene el predicado <code>"t"</code> son las que se espera que contenga.</p> <p>Si se utiliza la interfaz TAPI a la hora de intentar obtener la pregunta que realiza el depurador y se obtiene un error, la respuesta de error tendrá el formato especificado para la misma en el apartado 3.2.1.</p>
<div> <div>/debug_dl_answer</div> <div>Pregunta</div> <div>Respuesta</div> </div>	<p>A continuación se detalla el significado de los argumentos que recibe el comando:</p> <ul style="list-style-type: none"> <li>– Pregunta: es la pregunta que el depurador está realizando al usuario. Se puede obtener enviando a consola el comando <code>/debug_dl_current_question</code> y pegando la salida en esta sección.</li> <li>– Respuesta: es la respuesta a la pregunta que el depurador está realizando al usuario. Las respuestas permitidas son las siguientes: <code>abort</code>, <code>valid</code>, <code>nonvalid</code>, <code>missing(PartialTuple)</code> o <code>wrong(TotalTuple)</code>. Por ejemplo:</li> </ul> <ol style="list-style-type: none"> <li>1) Primero se obtiene y se copia de consola la pregunta actual que está realizando el depurador al usuario mediante el envío del comando <code>/debug_dl_current_question</code> a consola.</li> </ol> <pre> /debug_dl_current_question all(t/2) </pre> <p><i>Figura D-11 Pregunta depurador</i></p> <p>Como muestra la imagen anterior, el depurador está preguntando al usuario si todas las tuplas que contienen el predicado <code>"t"</code> son las que debe contener.</p>

	<p>2) Después, se copia la pregunta del punto 1) y se responde a dicha pregunta ("all(t/2)") enviando a consola el comando <code>/debug_dl_answer</code> junto con los siguientes argumentos para indicar que el contenido del predicado "t" es válido:</p> <p style="text-align: center;"><code>/debug_dl_answer all(t/2) valid</code></p> <p>La opción "valid" pasada como segundo argumento al comando <code>/debug_dl_answer</code> indica que el contenido del predicado "t" es el esperado.</p>
<code>/debug_dl_statistics</code>	<p>Permite obtener las estadísticas de la última sesión de depuración que se realizó.</p> <pre> Info: Buggy relation found: t Info: Debug Statistics: Info: Number of questions      : 1 Info: Number of inspected tuples: 2 Info: Number of root tuples    : 2 Info: Number of non-root tuples: 0 </pre> <p style="text-align: center;"><i>Figura D-12 Estadísticas de depuración</i></p> <p>Si se utiliza la interfaz TAPI para consultar las estadísticas de la última depuración realizada, es decir, si emite el comando <code>/debug_dl_statistics</code> precedido del comando <code>/tapi</code> y la ejecución del comando es exitosa, la respuesta tendrá el siguiente formato:</p> <pre> Item1 Value1 ... ItemN ValueN \$eot </pre> <p>Donde:</p> <ul style="list-style-type: none"> <li>— Itemi: cadena de texto que describe una de las medidas utilizadas para obtener las estadísticas de la depuración.</li> </ul> <p>Por ejemplo, "Item1" podría tomar como valor la cadena de texto "Number of questions" para hacer referencia a la medida que contabiliza la cantidad de preguntas que en total el depurador le ha hecho al usuario a lo largo de la depuración.</p>



	<ul style="list-style-type: none"> <li>– Valuei: indica el resultado numérico obtenido tras aplicar la medida “Itemi”.</li> </ul> <p>Por ejemplo, si “Item1” toma el valor “Number of questions” y “Value1” toma el valor “3” indicará que, en la última depuración, únicamente ha sido necesario realizar tres preguntas al usuario para obtener un resultado que resuelva la depuración.</p> <ul style="list-style-type: none"> <li>– \$eot: indica el fin de la transmisión de datos.</li> </ul> <p>En caso de error a la hora de intentar consultar las estadísticas de la última depuración, la respuesta de error tendrá el formato especificado para la misma en el apartado 3.2.1.</p>
--	--

Los siguientes comandos definidos en la tabla anterior serán útiles durante una sesión de depuración únicamente cuando la comunicación entre ACIDE y DES se esté realizando a través de la interfaz TAPI:

```
/debug_dl_set_node nombrePredicado/Aridad Estado
/debug_dl_current_question
/debug_dl_answer Pregunta Respuesta
```

Luego la emisión de los comandos debe ir precedida del comando /tapi como se muestra a continuación:

```
/tapi /debug_dl_set_node nombrePredicado/Aridad Estado
/tapi /debug_dl_current_question
/tapi /debug_dl_answer Pregunta Respuesta
```

En el apartado 3.2.7, mediante la realización de una sesión de depuración de ejemplo, se irán utilizando los distintos comandos de depuración definidos en este apartado para ver su utilidad de manera práctica.

### 3.2.4. Diseño del panel de depuración

En esta nueva versión del proyecto (versión 0.19), se ha incorporado a la interfaz gráfica ACIDE un nuevo panel de depuración para programas Datalog.

Hay dos zonas claramente diferenciadas en el panel de depuración Datalog

implementado:

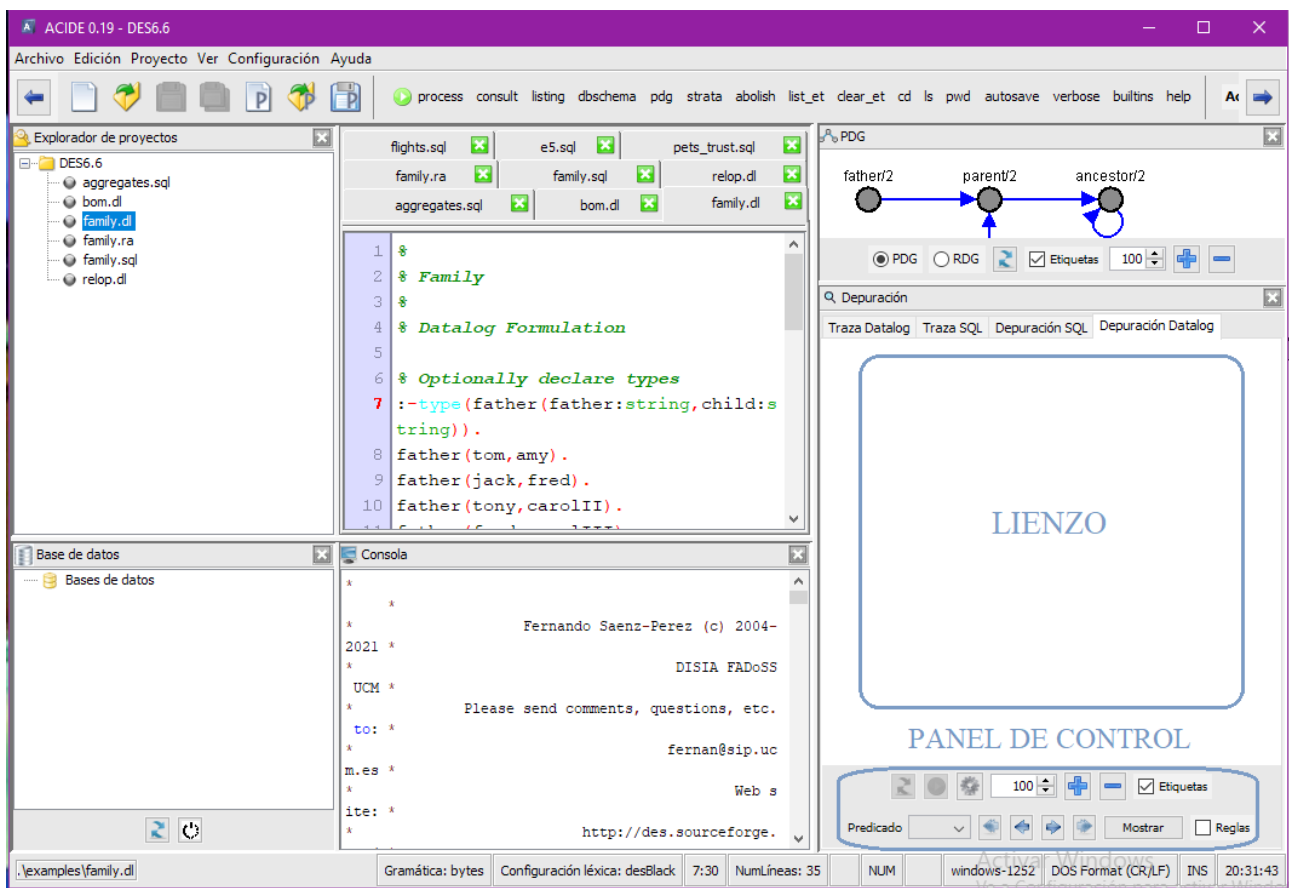


Figura D-13 Diseño del panel de depuración Datalog

Mientras que la zona superior del panel es un lienzo destinado a representar gráficamente el grafo de dependencias entre predicados que se debe depurar, la zona inferior se ha destinado a ubicar el conjunto de componentes (botones, casillas, etc.) que permiten realizar las siguientes acciones:

- Actualizar:

Cuando el usuario pulsa el botón de actualización (1), el grafo de dependencias de predicados cargado en el lienzo se vuelve a dibujar y su zoom se restablece al 100%.

Este botón únicamente estará disponible si el usuario ha seleccionado un predicado a depurar y la consola no está procesando comandos.

- Iniciar la depuración:

Cuando el usuario pulsa el botón de inicio (2), comienza una sesión de depuración

Datalog.

Este botón únicamente estará disponible si el usuario ha seleccionado un predicado a depurar y la consola no está procesando comandos.

– Configurar la depuración:

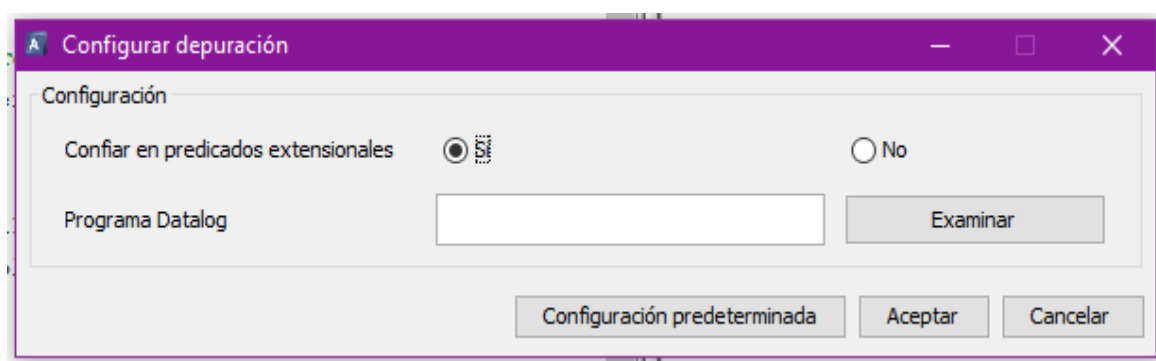
Cuando el usuario pulsa el botón de configuración (3), le aparecerá una ventana emergente en la que podrá establecer la siguiente configuración:

\* Podrá indicar si desea que, por defecto, se confíe en los predicados extensionales.

Si indica que, por defecto, se debe confiar en los predicados extensionales seleccionando la opción "Sí", durante una sesión de depuración Datalog, el depurador no inspeccionará las tuplas de los predicados extensionales que forman parte del grafo de dependencias de predicados a depurar, ya que confiará en que contienen datos correctos.

\* Podrá seleccionar un programa Datalog a depurar distinto de los programas Datalog de ejemplo que ofrece el proyecto.

En cualquier momento, podrá aplicar y guardar la configuración que ha realizado pulsando el botón "Aceptar". También podrá restaurar la configuración por defecto del depurador Datalog que es la de no confiar en los predicados extensionales seleccionando el botón "Configuración predeterminada".



*Figura D-14 Ventana configuración depuración Datalog*

Posteriormente, cuando arranque una sesión de depuración Datalog, se aplicará la configuración que ha establecido.

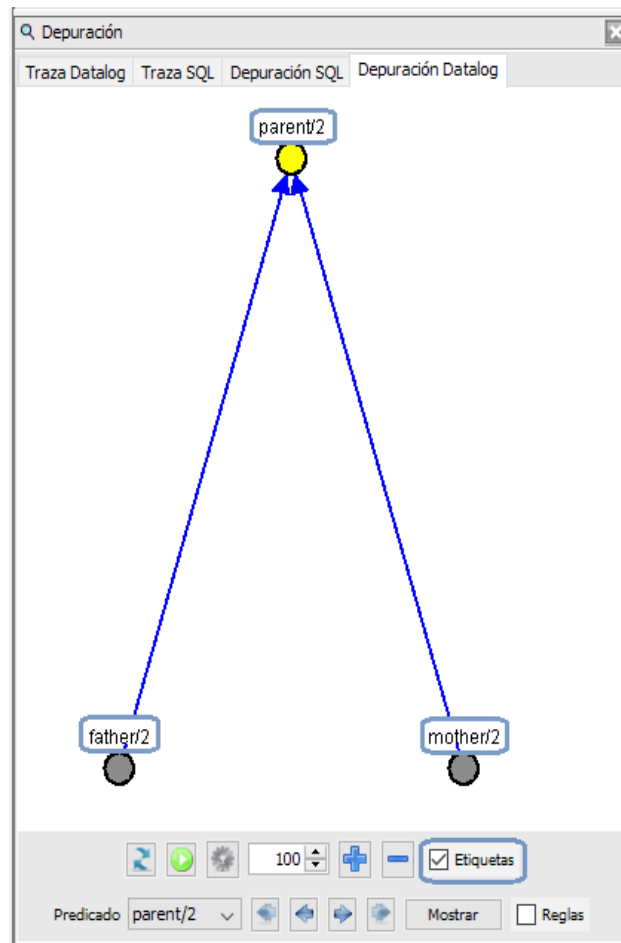
- Realizar zoom:

Cuando hay un grafo dibujado en el lienzo, el usuario podrá agrandar (+) o reducir (-) el tamaño del grafo ya sea pulsando los botones de zoom correspondientes o utilizando el componente para ajustar el zoom manualmente (4).

- Visualizar etiquetas:

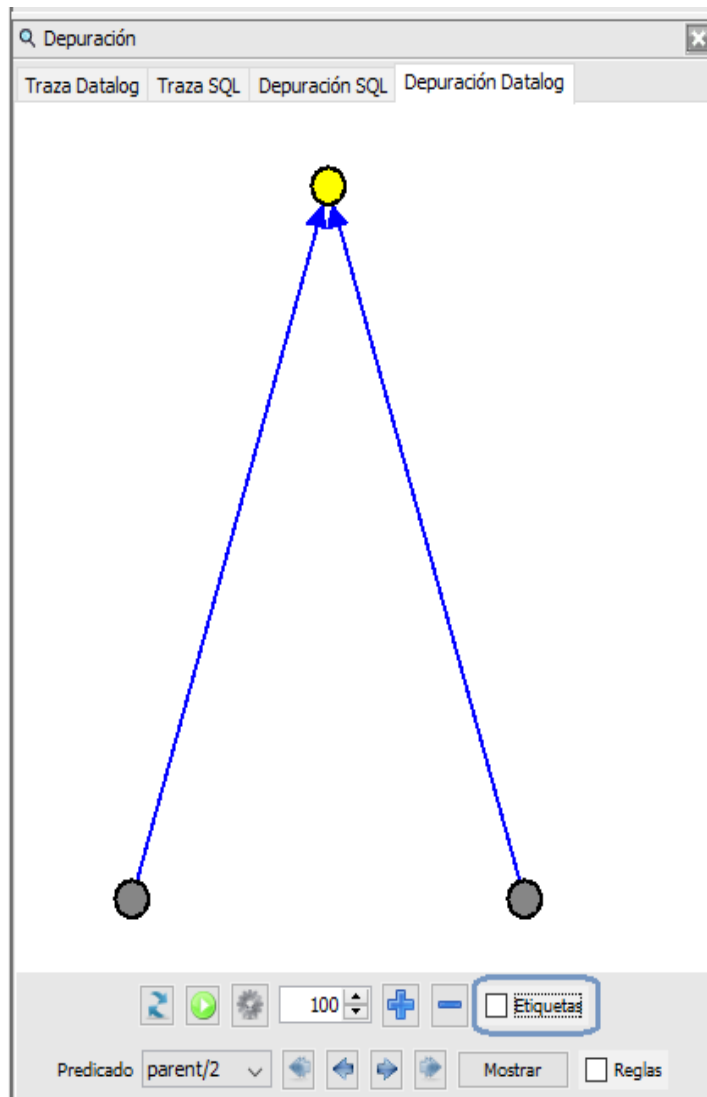
Cuando hay un grafo dibujado en el lienzo, el usuario podrá:

\* Marcar la casilla "Etiquetas" (5) si desea que, junto a cada nodo del grafo, se indique el nombre del predicado que representa más su aridad.



*Figura D-15 Grafo con etiquetas*

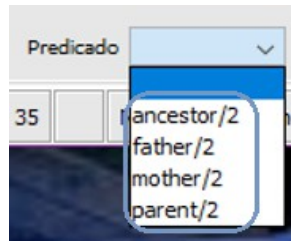
\* Desmarcar la casilla "Etiquetas" (5) si no desea que, junto a cada nodo del grafo, se indique el nombre del predicado que representa más su aridad.



*Figura D-16 Grafo sin etiquetas*

- Seleccionar un predicado a depurar:

Cuando un programa Datalog ha sido cargado en la base deductiva del sistema DES, el selector de predicados (6) sugerirá al usuario el listado de predicados del programa Datalog cargado. El usuario, de entre el listado de predicados sugeridos, únicamente podrá seleccionar uno.



*Figura D-17 Selector de predicados*

– Seleccionar nodos:

Cuando hay un grafo dibujado en el lienzo, un conjunto de cuatro botones (7) permiten al usuario desplazarse por los distintos nodos del grafo para ir seleccionándolos:

\* El primer botón comenzando por la izquierda, selecciona en el grafo el primer nodo.

El primer nodo es el nodo asociado al predicado que el usuario quiere depurar y que ha seleccionado en el selector de predicados.

\* El segundo botón comenzando por la izquierda, selecciona en el grafo el nodo anterior al seleccionado.

El nodo anterior al seleccionado es aquel nodo asociado al literal que va antes en el listado de literales del cuerpo de una regla o se encuentra en la cabeza y el asociado al nodo seleccionado en el cuerpo. Por ejemplo:

Dada la siguiente regla, "parent" se considera anterior a "father", ya que "parent" se encuentra en la cabeza y "father" en el cuerpo.

$$\text{parent}(X,Y) \text{ :- father}(X,Y).$$

\* El tercer botón comenzando por la izquierda, selecciona en el grafo el nodo siguiente al seleccionado.

El siguiente nodo al seleccionado es aquel nodo asociado al literal que va después en el listado de literales del cuerpo de una regla o se encuentra en el cuerpo de una regla y el asociado al nodo seleccionado en la cabeza. Por ejemplo:

Dada la siguiente regla, "father" sería posterior a "parent", ya que "father" se encuentra en el cuerpo y "parent" en la cabeza.

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

\* El cuarto botón comenzando por la izquierda, selecciona en el grafo el último nodo.

El último nodo del grafo es el nodo asociado al último literal en el listado de literales del cuerpo de una regla o la última regla.

– Mostrar definición en el editor:

Cuando hay un grafo dibujado en el lienzo, hay un nodo seleccionado en él y no está seleccionada la casilla "Reglas" (9), si el usuario pulsa el botón "Mostrar" (8), en el editor de ficheros se realizará un desplazamiento automático a las líneas del fichero del programa Datalog cargado que definen el predicado seleccionado quedando estas al inicio del editor.

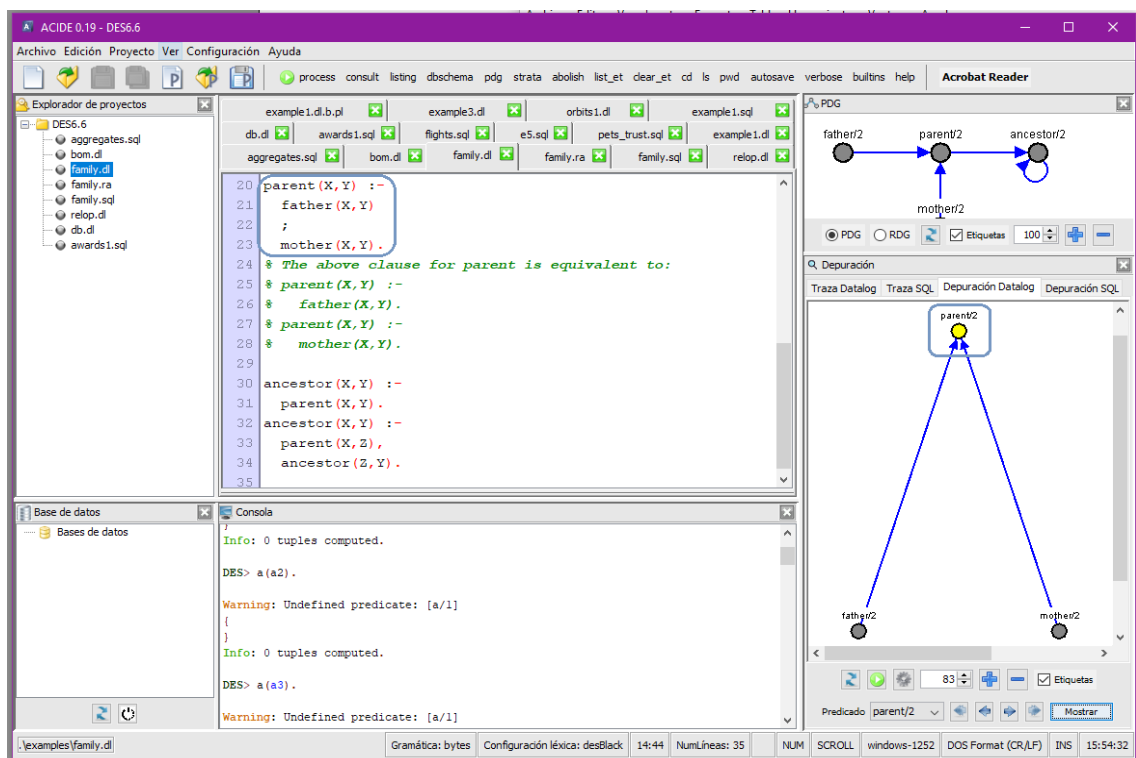


Figura D-18 Ubicación en el editor del predicado seleccionado

– Resaltar reglas:



Cuando hay un grafo dibujado en el lienzo y hay un nodo seleccionado en él, si el usuario marca la casilla "Reglas" (9), en el editor de ficheros se realizará un desplazamiento automático a las líneas del fichero del programa Datalog cargado que definen el predicado seleccionado.

Además, las líneas que definen el predicado seleccionado se resaltarán en verde fosforito.

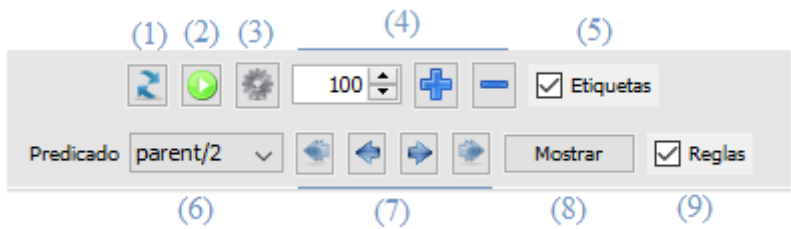


Figura D-19 Panel de control del panel depuración Datalog

### 3.2.5. Detalles técnicos del panel de depuración

En este apartado se darán detalles técnicos sobre el desarrollo principal: clases implicadas en la implementación del panel de depuración Datalog y un diagrama de clases de sus componentes principales.

Lo primero que se implementó fue el **panel de depuración Datalog** que aporta una funcionalidad básica: carga de predicados procesados, representación del grafo de depuración, opciones de desplazamiento por el grafo, etc.

Seguidamente, se desarrolló la **ventana de configuración de la depuración** que permite tanto cargar un programa Datalog específico a depurar como indicar si se debe confiar o no en los predicados extensionales.

Por último, se creó la **ventana de depuración** que permite realizar una depuración gráfica de un predicado.

La siguiente tabla resume la funcionalidad que ofrece cada una de las clases implementadas para el desarrollo del panel de depuración Datalog:

Clase	Funcionalidad ofrecida
Clases principales	
AcideDebugDatalogPanel.java	Implementa el panel de depuración Datalog.

AcideDebugDatalogConfiguration.java	Parte lógica de la configuración de la depuración: carga y guardado de configuración, etc.
AcideDebugDatalogConfigurationWindow.java	Implementa la ventana gráfica de configuración de la depuración.
AcideDebugDatalogDebugWindow.java	Implementa la ventana gráfica de depuración.
<b>Clases auxiliares en las que se apoyan las clases principales</b>	
AcideDebugDatalogPanelConfigureDebugListener.java	Muestra la ventana de configuración de la depuración.
AcideDebugDatalogPanelEditIntensionalPredicateListener.java	Permite editar de forma gráfica las reglas intensionales que definen un predicado.
AcideDebugDatalogPanelFirstNodeListener.java	Remarca el nodo principal del grafo de depuración.
AcideDebugDatalogPanelLastNodeListener.java	Remarca el último nodo del grafo de depuración.
AcideDebugDatalogPanelLocateListener.java	Ubica la definición de un predicado en el editor.
AcideDebugDatalogPanelMissingNodeListener.java	Registra una tupla faltante.
AcideDebugDatalogPanelNextNodeListener.java	Avanza al siguiente nodo del grafo de depuración.
AcideDebugDatalogPanelNonValidNodeListener.java	Marca un nodo como no válido.
AcideDebugDatalogPanelPreviousNodeListener.java	Retrocede al nodo anterior del grafo de depuración.
AcideDebugDatalogPanelRedoListener.java	Permite rehacer una acción aplicada durante la depuración.
AcideDebugDatalogPanelRefreshListener.java	Refresca el grafo de depuración cargado en el panel.
AcideDebugDatalogPanelShowErrorsListener.java	Muestra los errores detectados en la depuración.
AcideDebugDatalogPanelShowLabelsListener.java	Muestra las etiquetas de los nodos del grafo de depuración.
AcideDebugDatalogPanelShowPredicateListener.java	Muestra el contenido (tuplas) de un predicado.
AcideDebugDatalogPanelShowRulesListener.java	Remarca en el editor la definición de un predicado.
AcideDebugDatalogPanelStartDebugListener.java	Arranca una sesión de depuración gráfica.
AcideDebugDatalogPanelUndoListener.java	Permite deshacer una acción aplicada durante

	la depuración.
AcideDebugDatalogPanelValidNodeListener.java	Marca un nodo como válido.
AcideDebugDatalogPanelViewBoxListener.java	Actualiza el panel según el nuevo predicado seleccionado a depurar.
AcideDebugDatalogPanelWrongNodeListener.java	Registra una tupla errónea.
AcideDebugDatalogPanelZoomInListener.java	Amplia las dimensiones del grafo de depuración en el panel.
AcideDebugDatalogPanelZoomOutListener.java	Reduce las dimensiones del grafo de depuración en el panel.

A continuación se muestra el diagrama de clases tanto del panel de depuración Datalog como de su ventana de configuración y ventana de depuración. Para apreciar los detalles de los distintos diagramas es necesario incrementar el zoom de la página. Al hacer zoom, se verán todos los detalles con nitidez.

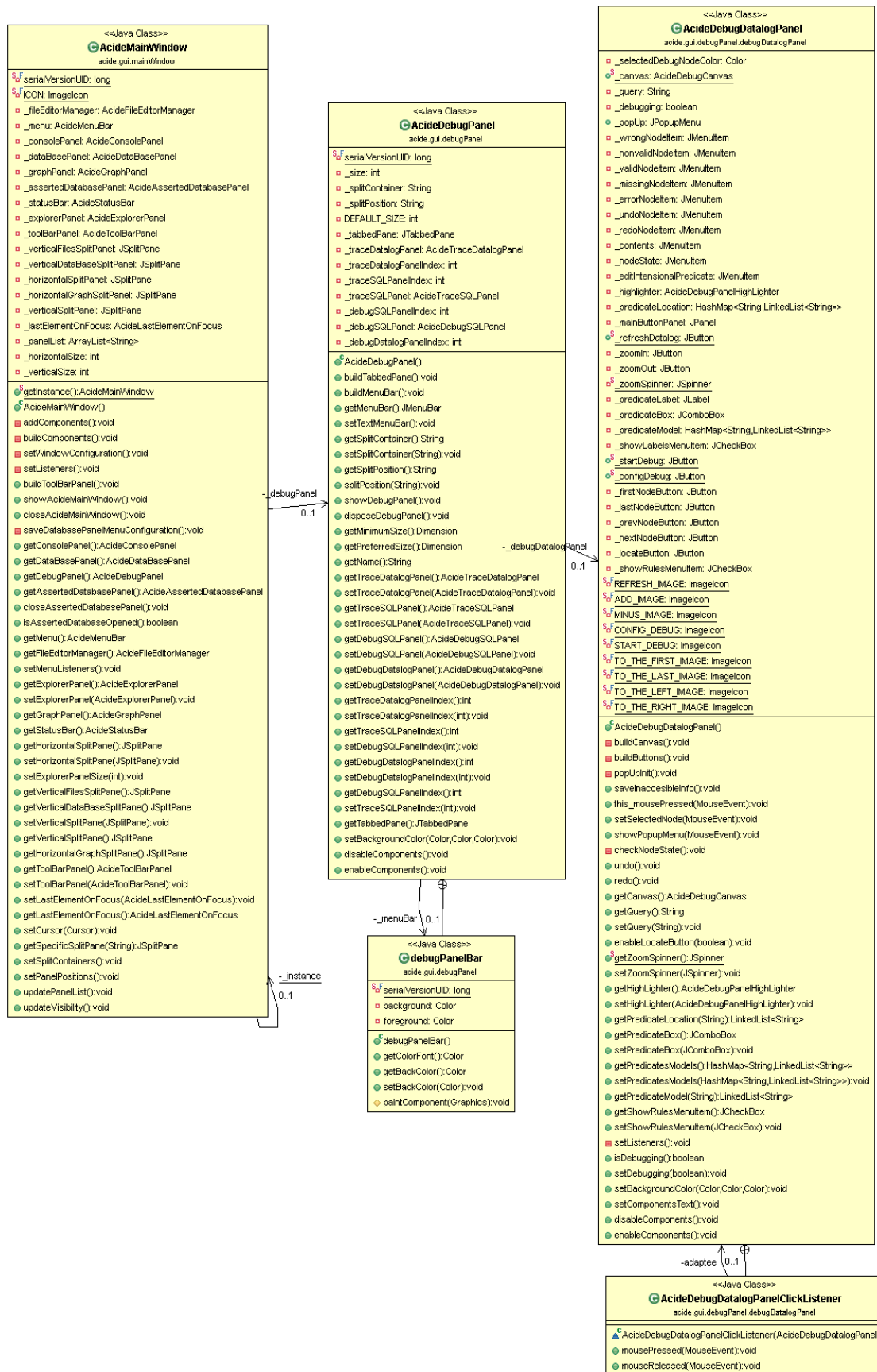
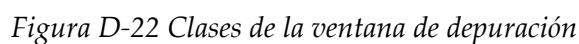
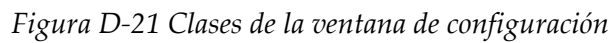


Figura D-20 Clases del panel depuración Datalog y su relación



### 3.2.6. Nociones básicas sobre una sesión de depuración

El objetivo de llevar a cabo una sesión de depuración Datalog es detectar qué está provocando que un predicado no contenga la información esperada (tuplas que se espera que contenga).

Por tanto, en cuanto un usuario detecte que un predicado no es válido debido a que no está devolviendo el resultado esperado, puede comenzar una sesión de depuración Datalog para depurarlo y encontrar la causa del error.

A la hora de iniciar la sesión de depuración, no debe marcar como válido el predicado a depurar, ya que se estaría indicando que es válido y, por tanto, no tendría sentido iniciar la sesión de depuración en busca de errores.

Una vez comenzada la sesión de depuración, el depurador irá recorriendo el grafo. Se irá analizando el siguiente predicado del grafo a depurar hasta detectar el predicado causante del problema.

El depurador no se detendrá a analizar aquellos predicados extensionales en los que se confía debido a que, por defecto, asumirá que su contenido es correcto y, por tanto, no hará falta depurarlos.

### 3.2.7. Depuración manual por consola

En esta sección, mediante un ejemplo, se llevará a cabo una sesión de depuración Datalog desde la consola sin hacer uso de la interfaz gráfica de depuración implementada para Datalog en ACIDE.

Posteriormente en el apartado 3.2.8 de esta memoria se realizará el mismo proceso de depuración pero a través de la interfaz gráfica.

De este modo, se podrán observar las ventajas de la depuración gráfica guiada frente a la depuración manual por consola.

El programa Datalog a depurar será el siguiente:

```
% Orbits of cosmological objects  
star(sun).  
orbits(earth, sun).
```

```

orbits(moon, earth).
orbits(X,Y) :-
    orbits(X,Z),
    orbits(Z,Y).

satellite(X,Y) :-
    orbits(X,Y),
    not(intermediate(X,Y)),
    not(star(Y)).

planet(X) :-
    orbits(X,Y),
    star(Y),
    not(intermediate(X,Y)).

intermediate(X,Y) :-
    orbits(X,Z),
    orbits(Y,Z).

```

Como se puede apreciar en el código anterior:

- El hecho "star(sun)." indica que el Sol (sun) es una estrella.
- La regla intensional "orbits(X,Y):- orbits(X,Z), orbits(Z,Y)." define que:

"X" orbitará/girará alrededor de "Y" si existe un "Z" tal que "X" orbita alrededor de él y "Z" a su vez orbita alrededor de "Y":

- El hecho "orbits(earth,sun)." indica que la Tierra (earth) gira alrededor del Sol (sun).
- El hecho "orbits(moon,earth)." indica que la Luna (moon) gira alrededor de la Tierra (earth). Por ejemplo:

La Luna (X) orbita alrededor del Sol (Y), ya que la Luna (X) orbita alrededor de la Tierra (Z) y la Tierra (Z) a su vez orbita alrededor del Sol (Y).

- La regla intensional "satellite(X,Y) :- orbits(X,Y), not intermediate(X,Y), not star(Y)." define que:

"X" será satélite de "Y" si "X" orbita alrededor de "Y", "Y" no es una estrella y, además, no hay un objeto intermedio (planeta, etc.) entre "X" e "Y" que también orbite alrededor de "Y". Por ejemplo:

La Luna (X) es un satélite de la Tierra (Y), ya que orbita alrededor de ella y, además, la Tierra no es una estrella y entre la Luna y la Tierra no hay un objeto intermedio que también orbite alrededor de la Tierra.

– La regla intensional "planet(X) :- orbits(X,Y), star(Y), not intermediate(X,Y)." define que:

"X" será un planeta si "X" orbita alrededor de "Y", "Y" es una estrella y, además, no hay un objeto intermedio (planeta, etc.) entre "X" e "Y" que también orbite alrededor de "Y". Por ejemplo:

La Tierra (X) es un planeta, ya que orbita alrededor del Sol (Y) que es una estrella y, además, no hay un objeto intermedio entre la Tierra y el Sol que también esté orbitando alrededor del Sol.

– La regla intensional "intermediate(X,Y) :- orbits(X,Z), orbits(Y,Z)." define que:

Entre "X" e "Y" existe un objeto intermedio (planeta, etc.) si tanto "X" como "Y" orbitan alrededor de "Z". Por ejemplo:

Según la definición de la regla anterior, entre la Luna (X) y la Tierra (Y) existe un objeto intermedio (planeta, etc.), ya que la Luna orbita alrededor del Sol (Z) y la Tierra también orbita alrededor del Sol.

El programa Datalog anterior se ha guardado en un fichero con nombre "solarsystem.dl". El fichero "solarsystem.dl" ha sido almacenado en la carpeta "examples" de la raíz del proyecto ACIDE.

Posteriormente, se ha emitido a la consola DES el siguiente comando para procesar el fichero "solarsystem.dl" y almacenar las reglas del programa Datalog que contiene en la base de datos del sistema:

```
/consult ./examples/solarsystem.dl
```

Ahora el sistema ha inferido información (tuplas) a partir de las reglas que se han cargado en su base de datos.

Un astrónomo ha consultado los diferentes planetas que componen el sistema solar cargado en la base de datos del sistema DES y se ha llevado las manos a la cabeza cuando el sistema le ha informado que la Luna (moon) es uno de los planetas que lo componen, ya que la Luna es un satélite:



Consulta realizada	DES> <b>planet(X)</b>
Salida obtenida	<pre>{   planet(earth),   planet(moon) }</pre> Info: 2 tuples computed.

Ahora es nuestra misión depurar el predicado planet(X) para averiguar qué ha ocurrido. El procedimiento que realizamos para depurar el predicado planet(X) y encontrar la causa del error es el siguiente:

Se emite el siguiente comando para iniciar la depuración del predicado "planet/1", ya que contiene un objeto que no es un planeta (moon):

Comando emitido	DES> <b>/debug_dl planet/1</b>
Salida obtenida	Info: Loading and transforming file... <pre>{   1-planet(earth),   2-planet(moon) }</pre> Input: Is this the expected answer for planet/1? (y/n/mT/wN/a/h) [n]:

El sistema nos pregunta si el contenido del predicado "planet/1" es el esperado. Indicamos que no especificando que la segunda tupla (2-planet(moon)) es errónea mediante la combinación "w2", ya que la Luna (moon) no es un planeta.

Cuanto más detalles demos durante la depuración, antes se resolverá la causa del error. Por ejemplo, no es lo mismo indicar que el predicado planet(X) no es válido introduciendo como respuesta "n" que indicar que una tupla específica es errónea.

Respuesta emitida	Input: Is this the expected answer for planet/1? (y/n/mT/wN/a/h) [n]: <b>w2</b>
Salida obtenida	<pre>{   1-star(sun) }</pre> Input: Is this the expected answer for star/1? (y/n/mT/wN/a/h) [n]:

Ahora el sistema nos pregunta si el contenido del predicado "star/1" es el esperado. Indicamos que sí introduciendo el carácter "y", ya que todas las estrellas que contiene (todas sus tuplas que únicamente es una) son estrellas: el Sol (sun) es una estrella.

Respuesta emitida	Input: Is this the expected answer for star/1? (y/n/mT/wN/a/h) [n]: y
Salida obtenida	<pre>{   1-intermediate(moon,sun) }</pre> Input: Do you expect that NONE of these tuples be in intermediate/2? (y/n/mT/wN/a/h) [n]:

En este punto, el sistema nos pregunta si lo esperado es que ninguna de las tuplas mostradas en la salida (1-intermediate(moon,sun)) estén en el predicado "intermediate/2".

Indicamos que no introduciendo el caracter "n" porque la tupla mostrada en la salida sí debe estar en "intermediate/2", ya que entre la Luna (moon) y el Sol (sun) sí hay un objeto intermedio que es la Tierra debido a que la Tierra orbita alrededor del Sol.

Respuesta emitida	Input: Do you expect that NONE of these tuples be in intermediate/2? (y/n/mT/wN/a/h) [n]: n
Salida obtenida	<pre>{   1-orbits(earth,sun),   2-orbits(moon,earth),   3-orbits(moon,sun) }</pre> Input: Is this the expected answer for orbits/2? (y/n/mT/wN/a/h) [n]:

Ahora el sistema nos pregunta si el contenido del predicado "orbits/2" es el esperado. Indicamos que sí introduciendo el caracter "y", ya que todos los objetos que indica que orbitan alrededor de otros (todas sus tuplas) realmente sí orbitan:

La Tierra orbita alrededor del Sol (earth,sun), la Luna orbita alrededor de la Tierra (moon, earth) y, además, la Luna orbita alrededor del Sol.

Respuesta emitida	Input: Is this the expected answer for orbits/2? (y/n/mT/wN/a/h) [n]: y
Salida obtenida	<pre>{   1-intermediate(earth,earth),   2-intermediate(earth,moon),   3-intermediate(moon,earth),   4-intermediate(moon,moon) }</pre> Input: Is this the expected answer for intermediate/2? (y/n/mT/wN/a/h)

[n]:

En este punto, el sistema nos pregunta si el contenido del predicado "intermediate/2" es el esperado. Indicamos que no especificando que la segunda tupla (2-intermediate(earth,moon)) es errónea mediante la combinación "w2", ya que entre la Tierra (earth) y la Luna (moon) no hay un objeto intermedio que orbite alrededor de la Luna.

Respuesta emitida	Input: Is this the expected answer for intermediate/2? (y/n/mT/wN/a/h) [n]: w2
Salida obtenida	{ 1-orbits(earth,sun), 2-orbits(moon,sun) } Input: Do you expect that ALL these tuples be in orbits/2? (y/n/mT/wN/a/h) [n]:

Ahora el sistema nos pregunta si lo esperado es que todas las tuplas mostradas en la salida estén en el predicado "orbits/2".

Indicamos que sí introduciendo el caracter "y", ya que las tuplas mostradas en la salida sí deben estar en "orbits/2" debido a que tanto la Tierra orbita alrededor del Sol (earth, sun) como la Luna orbita alrededor del Sol (moon,sun).

Respuesta emitida	Input: Do you expect that ALL these tuples be in orbits/2? (y/n/mT/wN/a/h) [n]: y
Salida obtenida	Info: Bug source: [buggy(intermediate,erroneous program rule, [(earth,moon),_intermediate1])] Info: Debug Statistics: Info: Number of questions : 6 Info: Number of inspected tuples: 13 Info: Number of root tuples : 2 Info: Number of non-root tuples : 11

La salida obtenida en este último punto nos indica que se ha llegado al fin de la depuración debido a que nos informa de los siguientes dos puntos:

- Que se ha encontrado la fuente del error y es la siguiente:

La regla que se ha definido para "intermediate" es errónea.

Bug source: [buggy(intermediate,erroneous program rule,[(earth,moon),\_intermediate1])]

- Las estadísticas de la depuración que, entre otras cosas nos informan que el sistema ha necesitado realizar hasta seis preguntas al usuario para poder encontrar la causa del error.

Ahora, sabiendo la causa del error, es nuestra responsabilidad corregir la regla definida para "intermediate" que se encuentra almacenada en la base de datos del sistema DES. Para ello, emitiremos los siguientes comandos:

- Un comando que elimina de la base de datos del sistema la regla errónea que define "intermediate":

```
/retract intermediate(X,Y) :- orbits(X,Z), orbits(Y,Z).
```

- Un comando que inserta en la base de datos del sistema la regla correcta que debe definir "intermediate":

```
/assert intermediate(X,Y) :- orbits(X,Z), orbits(Z,Y).
```

La regla anterior es correcta, ya que define correctamente la situación en la cual hay un objeto intermedio (Z) entre un objeto "X" que orbita alrededor de un objeto "Y" alrededor del cual también orbita el objeto "Z". Por ejemplo:

Se puede declarar que hay un objeto (planeta, etc.) intermedio entre la Luna (X) y el Sol (Y) si la Luna orbita alrededor de la Tierra (Z) y la Tierra orbita alrededor del Sol.

Tras realizar estas modificaciones, solicitamos al astrónomo que vuelva a consultar los planetas y satélites del sistema solar cargado en la base de datos. El astrónomo consulta de nuevo los planetas y, como novedad, también los satélites del sistema solar.

Consulta realizada	DES> <b>planet(X)</b>
Salida obtenida	{ planet(earth) } Info: 1 tuple computed.
Consulta realizada	DES> <b>satellite(X,Y)</b>

Salida obtenida	<pre>{   satellite(moon,earth) }</pre> <p>Info: 1 tuple computed.</p>
-----------------	---

Ahora ya puede respirar aliviado. La Luna ya no se considera uno de los planetas del sistema solar pero sí uno de sus satélites. ¡Buen trabajo!

Por último, si analizamos detenidamente el proceso de depuración llevado a cabo, algunas de las desventajas que se pueden observar son las siguientes:

- El usuario tiene que encargarse de recordar la sintaxis de los comandos, ya que trabaja directamente con ellos: comando para arrancar la sesión de depuración, comando para eliminar y añadir nuevas reglas, etc.  
Como tiene que introducir los comandos manualmente, debe tener cuidado de que estén correctamente definidos y sus argumentos sean los esperados.
- Durante la depuración, no puede consultar libremente ciertos detalles como el contenido de otros predicados.
- No puede revertir decisiones tomadas en caso de equivocarse al dar una respuesta. Esto provoca que tenga que realizar la depuración de nuevo partiendo desde cero.

### 3.2.8. Depuración gráfica guiada

Como se ha visto en el apartado anterior, el proceso manual de depuración por consola tiene claras desventajas. Por este motivo, para facilitar el proceso de depuración al usuario, se ha implementado una interfaz gráfica de depuración de programas Datalog que, entre otras cosas, ofrece ventajas como las siguientes:

- Permite deshacer o rehacer acciones llevadas a cabo durante el proceso de depuración.
- El usuario no tendrá que preocuparse de recordar la sintaxis de los comandos ni trabajar con ellos directamente. Por tanto, el sistema siempre emitirá el comando correcto junto a las opciones adecuadas.
- En todo momento se puede consultar el contenido de cualquier predicado depurado o pendiente de depurar.
- En cualquier momento de la depuración se puede establecer el estado de uno de los nodos pendientes de depurar agilizando así el fin de la depuración, ya que el depurador ya no necesitará analizar el contenido de dicho nodo, puesto que ya se ha indicado su estado.

Al igual que en el apartado 3.2.7, el predicado a depurar en este apartado volverá a ser el predicado "planet(X)" del programa Datalog definido en el fichero "solarsystem.dl".

Una de las opciones para cargar el contenido del programa Datalog del fichero "solarsystem.dl" en la base de datos del sistema DES es añadir el archivo "solarsystem.dl" al proyecto DES6.6 que hay abierto en el panel de explorador de proyectos de ACIDE y procesarlo. Para ello, realizamos los siguientes pasos:

1) Damos clic derecho sobre la carpeta del proyecto "DES6.6" ubicado en el panel de explorador de proyectos de ACIDE y seleccionamos la opción "Añadir archivo(s)".

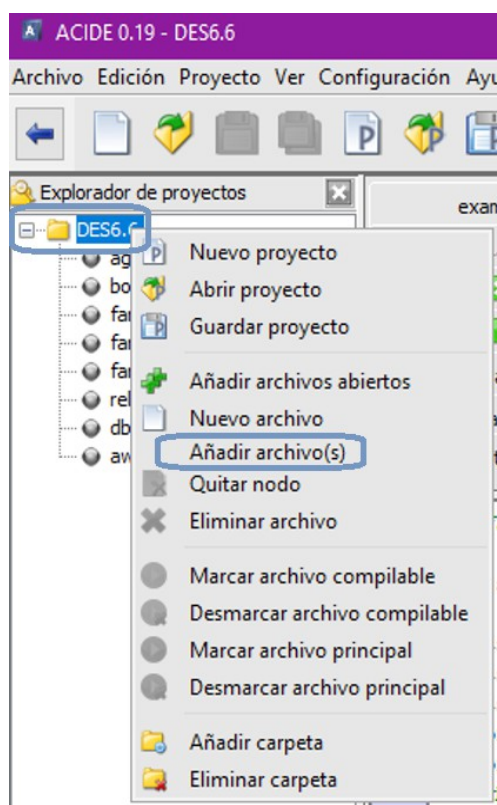


Figura D-23 Añadir archivos

2) Nos ubicamos dentro de la carpeta "examples" de la raíz del proyecto ACIDE y seleccionamos el fichero "solarsystem.dl". Tras realizar esta acción, el archivo "solarsystem.dl" habrá sido añadido al proyecto DES6.6 abierto en ACIDE.

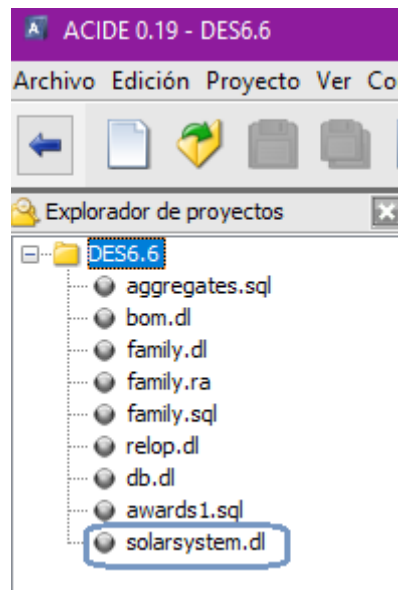


Figura D-24 Fichero solarsystem.dl cargado

Nos situamos sobre el fichero "solarsystem.dl" y lo procesamos seleccionando alguno de los siguientes botones disponibles en la barra de herramientas de la interfaz gráfica:

- Botón con icono Play.
- Botón con texto "consult".

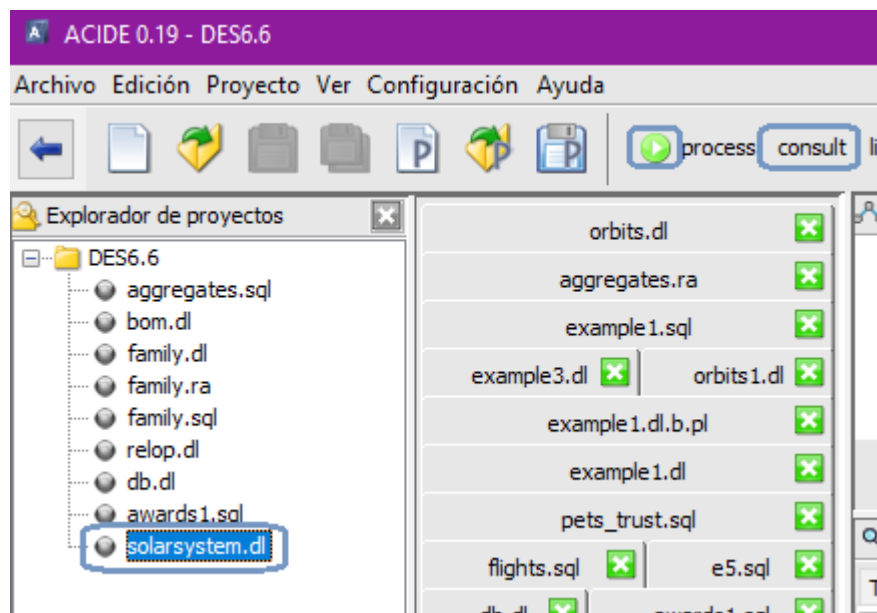


Figura D-25 Procesamiento del fichero

Tras procesar el fichero, las reglas del programa Datalog se habrán cargado en la base de datos del sistema DES.

A continuación nos dirigimos al panel de depuración Datalog y realizamos las siguientes configuraciones previas a la depuración:

- En el campo "Predicado" del panel de control, seleccionamos el predicado a depurar: "planet/1".

Tras seleccionar el predicado "planet/1", automáticamente se dibujará el grafo de predicados que contiene tanto el predicado detectado como erróneo como aquellos predicados a partir de los cuales se construye.

El nodo coloreado de amarillo será el primer nodo a depurar.

- Marcamos la casilla "Reglas" para que, durante la depuración, en el fichero "solarsystem.dl" del editor, se vayan iluminando en verde fosforito las distintas reglas que definen el predicado que actualmente se está depurando.

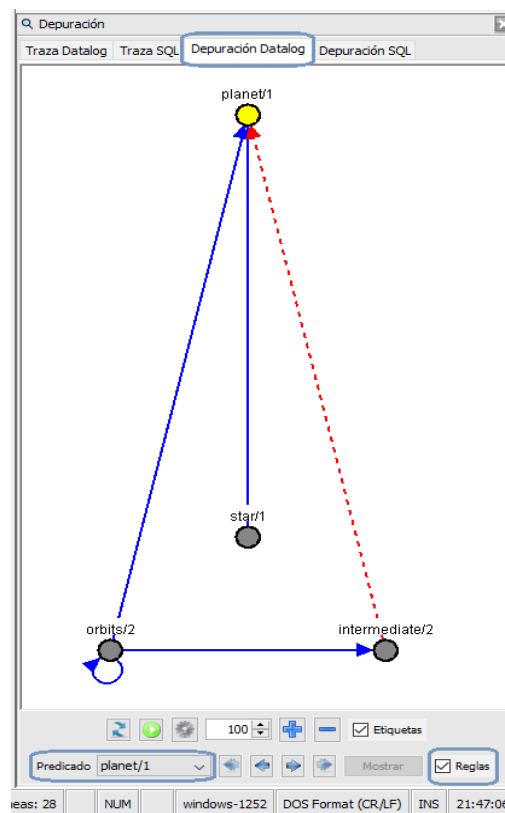
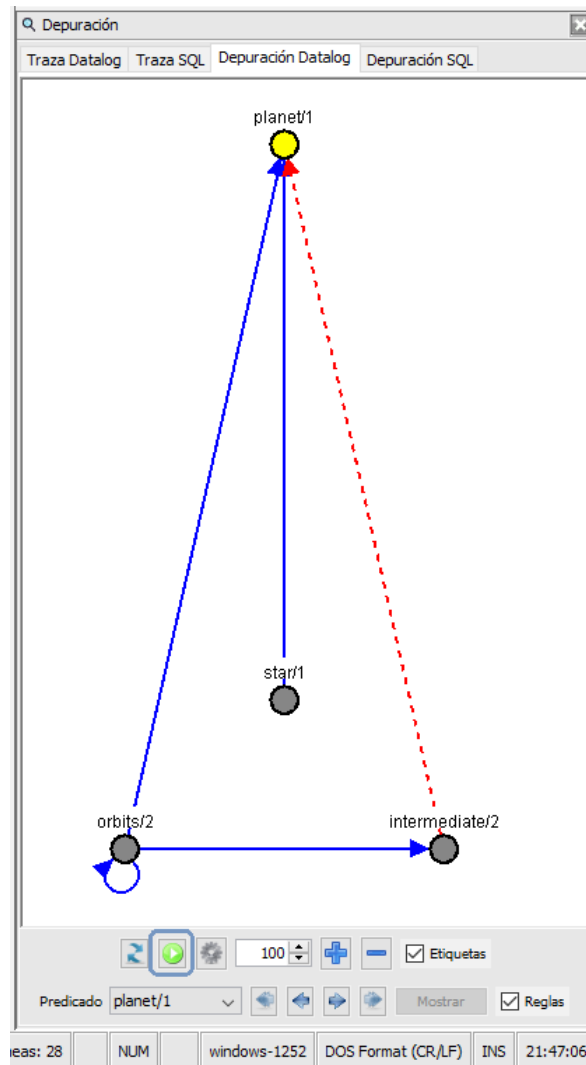


Figura D-26 Configuraciones previas a la depuración



El procedimiento que realizamos para depurar el predicado planet(X) y encontrar la causa del error es el siguiente:

Se selecciona el botón con icono Play que hay en el panel de control del panel de depuración Datalog para arrancar la depuración.



*Figura D-27 Botón con icono Play depuración*

Tras seleccionar el botón con icono Play, comienza la sesión de depuración y, como muestra la siguiente imagen, lo primero que nos pregunta el depurador es si el contenido del predicado "planet" es el esperado:

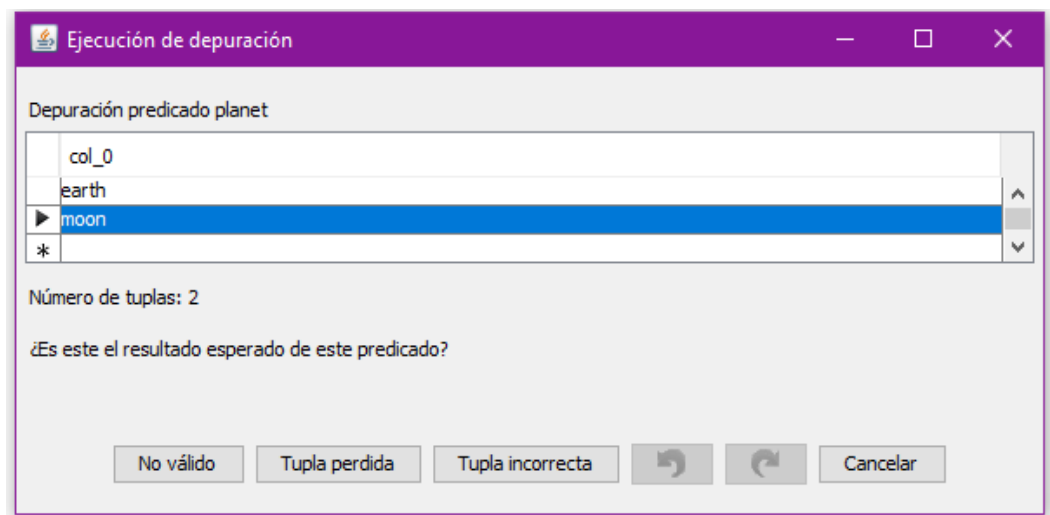


Figura D-28 Pregunta 1 depuración gráfica

Indicamos que no seleccionando la segunda tupla (moon) y pulsando el botón "Tupla incorrecta", ya que la Luna (moon) no es un planeta. El sistema realizará una actualización en el grafo y coloreará de naranja el nodo con etiqueta "planet/1" para indicar que no es válido.

Como bien dijimos en el apartado 3.2.7, cuantos más detalles demos durante la depuración, antes se encontrará la causa del error. Por ejemplo, no es lo mismo indicar que el predicado planet(X) no es válido que indicar que una de sus tuplas es incorrecta, ya que el depurador se enfocará en resultados concretos.

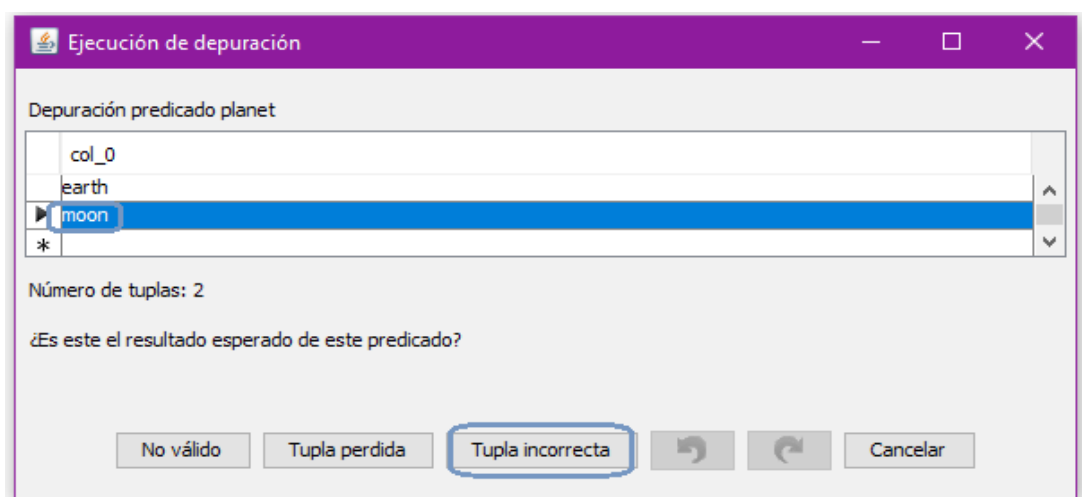


Figura D-29 Pregunta 1 depuración gráfica con respuesta

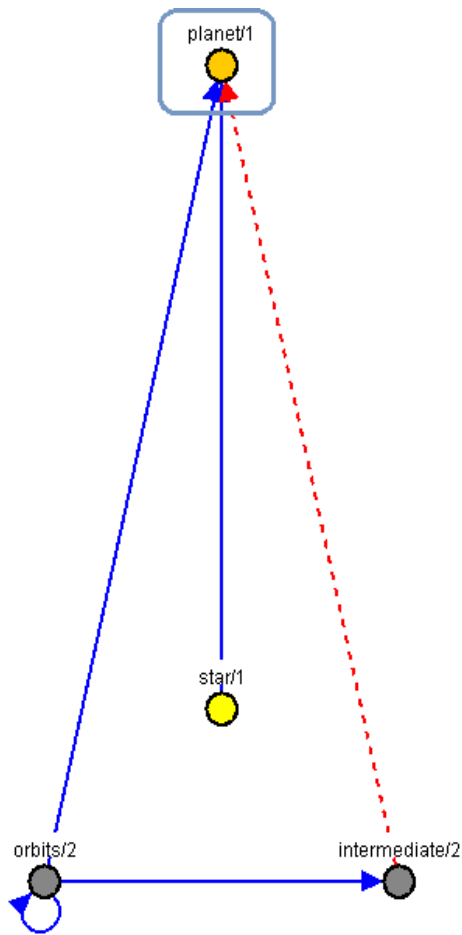


Figura D-30 Nodo anaranjado

Como muestra la siguiente imagen, la segunda pregunta que nos realiza el depurador es si el contenido del predicado "star" es el esperado:

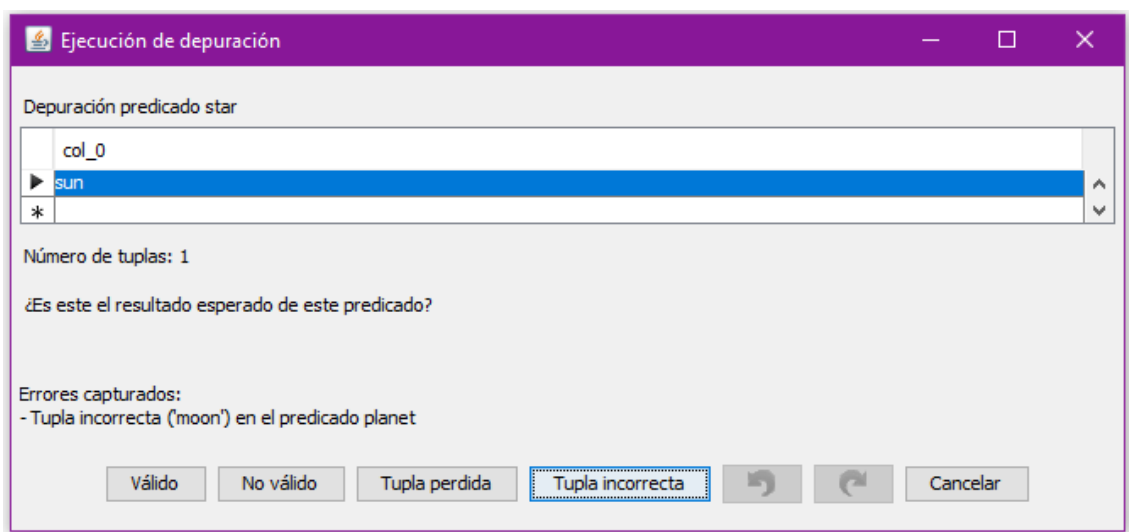


Figura D-31 Pregunta 2 depuración gráfica

Antes de responder a la segunda pregunta, para facilitar el trabajo al depurador, vamos a consultar el contenido del predicado "orbits/2" y, si su contenido es el esperado, lo marcaremos como válido estableciendo su estado a "Valid".

Esto hará que el depurador no se pare a inspeccionar el contenido completo del predicado "orbits/2" debido a que se ha indicado que es válido.

Para ello, seleccionamos las siguientes opciones implementadas y disponibles al dar clic derecho con el ratón sobre un nodo del grafo:

1) Damos clic derecho sobre el nodo con etiqueta "orbits/2" y, en el menú contextual que aparece, seleccionamos su opción "Abrir".

Tras realizar la acción anterior, aparecerá una ventana emergente con una tabla que nos mostrará el contenido/tuplas del predicado "orbits".

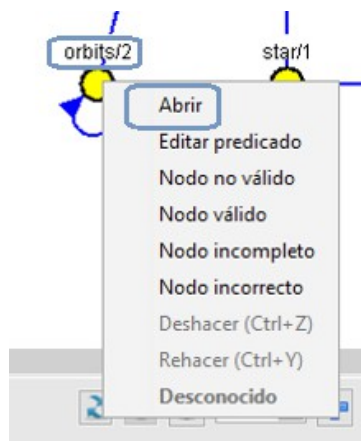
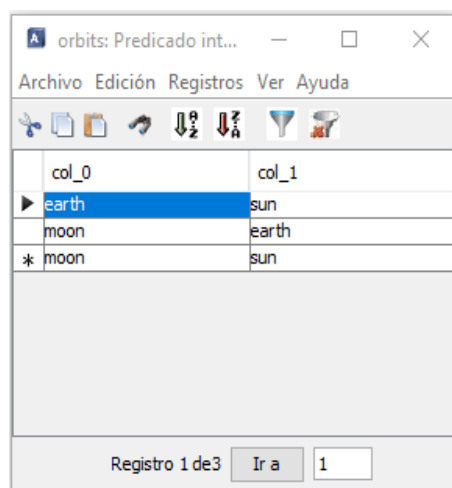


Figura D-32 Visualizar contenido predicado



col_0	col_1
earth	sun
moon	earth
* moon	sun

Registro 1 de 3   Ir a   1

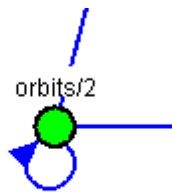
*Figura D-33 Contenido predicado*

Tras inspeccionar el contenido del predicado "orbits" determinamos que es correcto, ya que:

- La Tierra (earth) orbita alrededor del Sol (sun): la primera tupla (earth,sun) es correcta.
- La Luna (moon) orbita alrededor de la Tierra (earth): la segunda tupla (moon, earth) es correcta.
- La Luna (moon) orbita alrededor del Sol (sun): la tercera tupla (moon, sun) es correcta.

Todas las tuplas del predicado "orbits" son correctas. Además, no se espera que contenga tuplas adicionales. El predicado puede ser marcado como válido.

2) Nuevamente damos clic derecho sobre el nodo con etiqueta "orbits/2" y seleccionamos su opción "Nodo válido" para marcarlo como válido. La aplicación actualizará el color del nodo a verde fosforito para indicar que es un nodo con contenido válido.



*Figura D-34 Nodo válido*

Estas dos acciones que acabamos de llevar a cabo no sería posible realizarlas durante una depuración manual por consola a no ser que se utilizara la interfaz TAPI para la comunicación. Por tanto, como vemos, estas son algunas de las ventajas que la depuración gráfica ofrece.

Es importante mencionar que el menú contextual de los nodos que hay disponible al pulsar clic derecho sobre los mismos es una funcionalidad implementada muy potente que también permite llevar a cabo de forma completa la depuración de un predicado concreto, ya que proporciona las siguientes ventajas:

- Proporciona libertad a la hora de ir depurando los nodos del grafo.

No obliga a seguir el camino de depuración que marca el sistema, ya que en cualquier momento se puede aplicar un cambio en el estado de cualquier nodo pendiente de depurar.

- Simplifica la gestión de elementos al usuario debido a que no hace uso de una ventana de depuración.

Retomamos a la segunda pregunta que nos está realizando el depurador y seleccionamos el botón "Válido" para responder a la segunda pregunta, ya que el contenido del predicado "star" es el esperado, puesto que la tupla que contiene indica que el Sol (sun) es una estrella y este hecho es correcto. Además, no esperamos que contenga más estrellas.

Tras responder a la segunda pregunta, como se muestra en la siguiente imagen, el depurador nos pregunta si la tupla "sun" debe pertenecer al predicado "star".

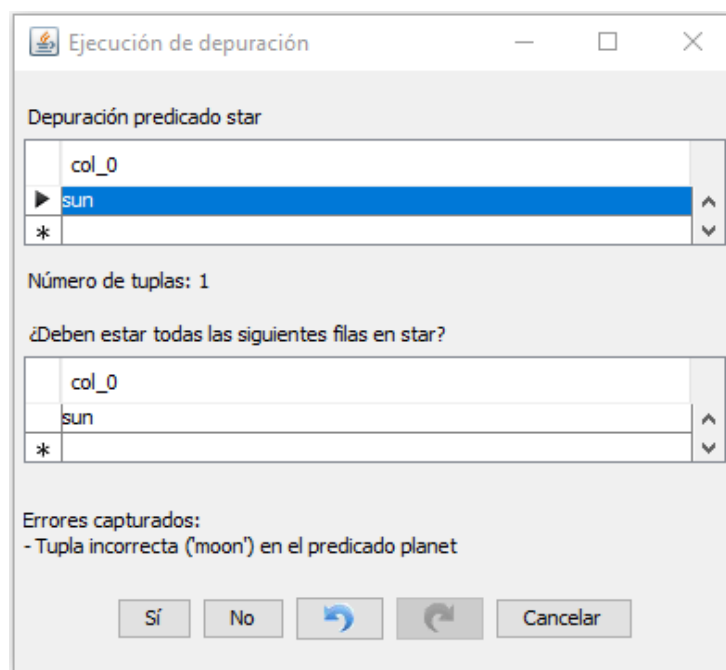


Figura D-35 Pregunta 3 depuración gráfica

Tal y como hemos explicado anteriormente, seleccionamos el botón "Sí" para responder a la tercera pregunta e indicar que "sun" sí debe pertenecer a "star" debido a que se considera una estrella.

Tras responder a la tercera pregunta, como se muestra en la siguiente imagen, el depurador nos pregunta si la tupla "moon,sun" debe pertenecer al predicado "intermediate".

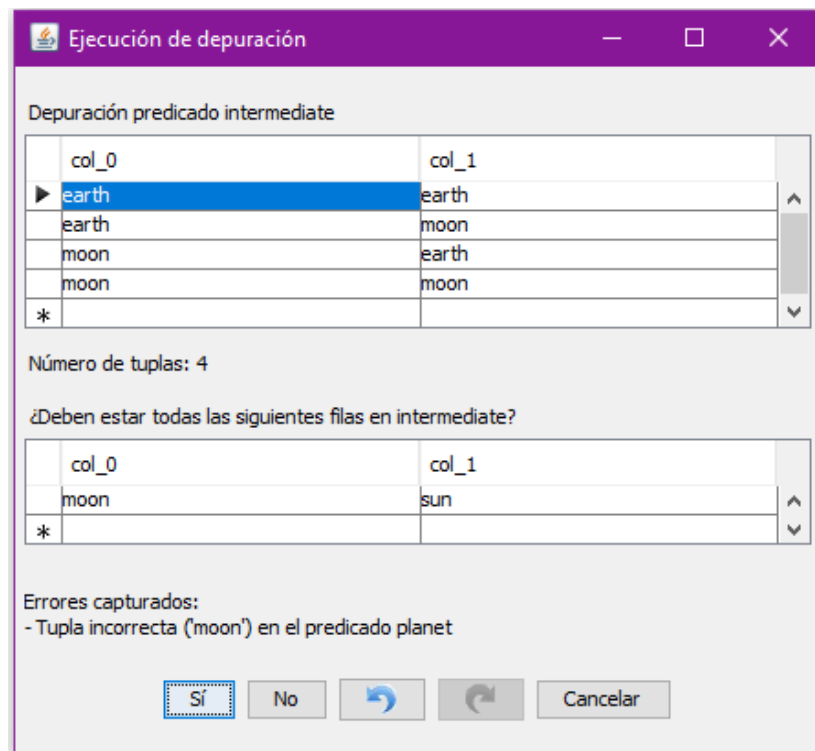


Figura D-36 Pregunta 4 depuración gráfica

Durante la depuración recibimos una llamada de teléfono que nos despista y para responder a esta cuarta pregunta seleccionamos el botón "No".

Después, recapacitamos y nos damos cuenta de que hemos respondido erróneamente a la cuarta pregunta, ya que "moon,sun" sí debe pertenecer a "intermediate" debido a que entre la Luna (moon) y el Sol (sun) hay un objeto intermedio (la Tierra (earth)) que también orbita alrededor del Sol.

Para corregir la respuesta errónea introducida para la cuarta pregunta, seleccionamos el botón "deshacer" disponible y habilitado en la ventana emergente de depuración que realiza la siguiente pregunta. También se puede pulsar el atajo de teclado "Ctrl + Z + Z" para aplicar la acción de "deshacer". Al revertir la acción, retornamos a la cuarta pregunta. Esta es otra ventaja que la depuración gráfica ofrece.

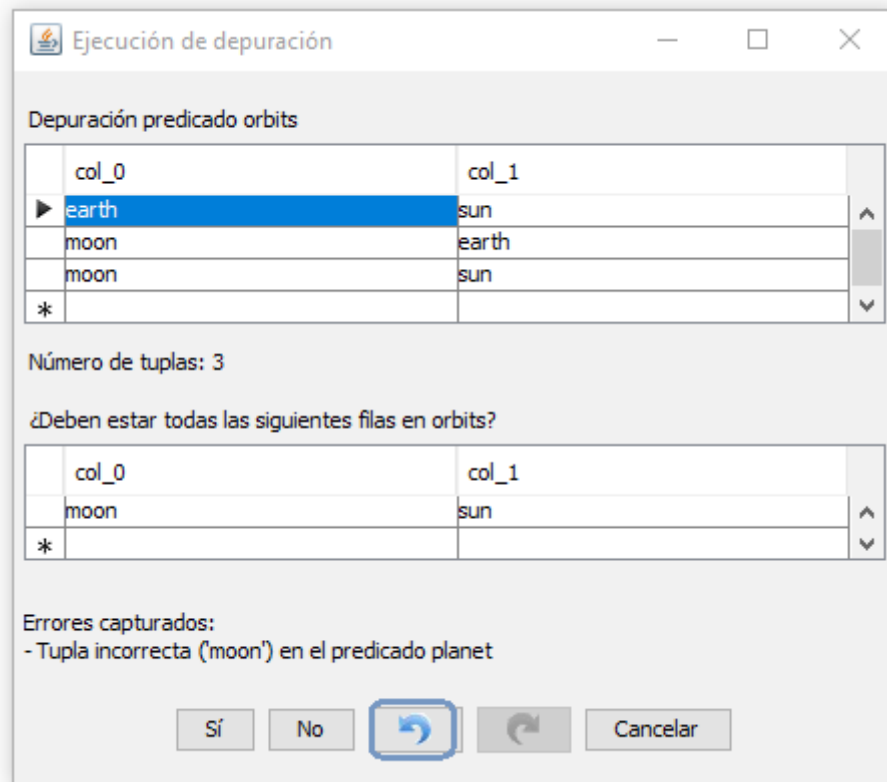


Figura D-37 Deshacer en depuración

Tras responder a la cuarta pregunta de forma correcta, tal y como se muestra en la siguiente imagen, el depurador nos pregunta si el contenido del predicado "intermediate" es el esperado.

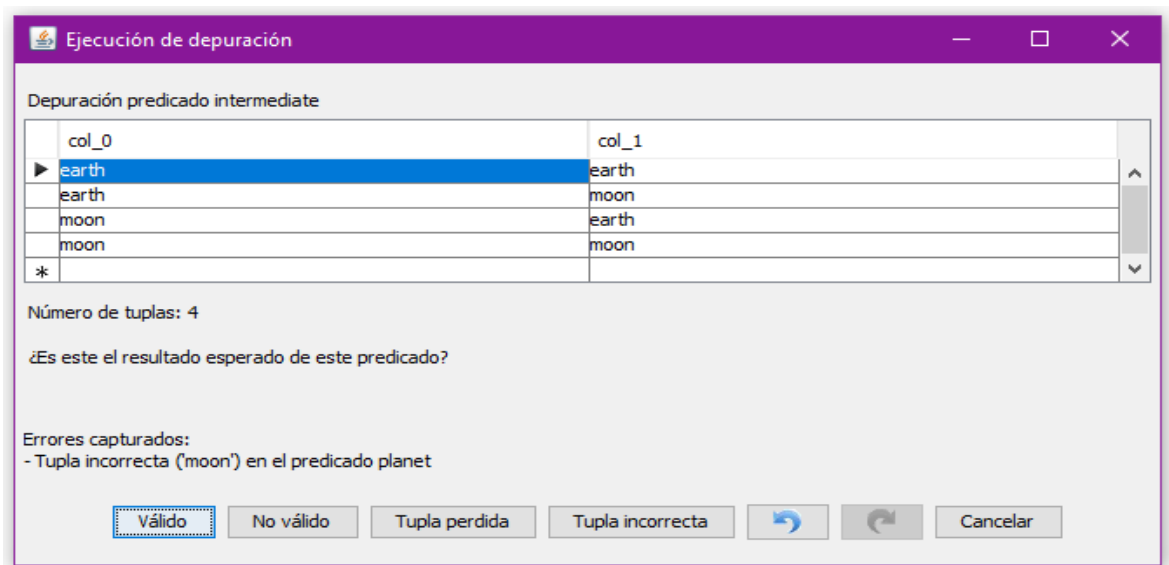


Figura D-38 Pregunta 5 depuración gráfica



Respondemos a esta quinta pregunta seleccionando la segunda tupla "earth,moon" y pulsando el botón "Tupla incorrecta".

De este modo indicamos que "earth,moon" no debe pertenecer a "intermediate", ya que entre la Tierra (earth) y la Luna (moon) no hay un objeto intermedio que esté siendo orbitado por la Tierra y a su vez que ese objeto esté orbitando alrededor de la Luna. Además, la Tierra no orbita alrededor de la Luna.

Una vez respondida a la quinta pregunta, como se muestra en la siguiente imagen, el depurador sigue formulando preguntas y, en este caso, nos pregunta si las tuplas "earth,sun" y "moon,sun" deben pertenecer al predicado "orbits".

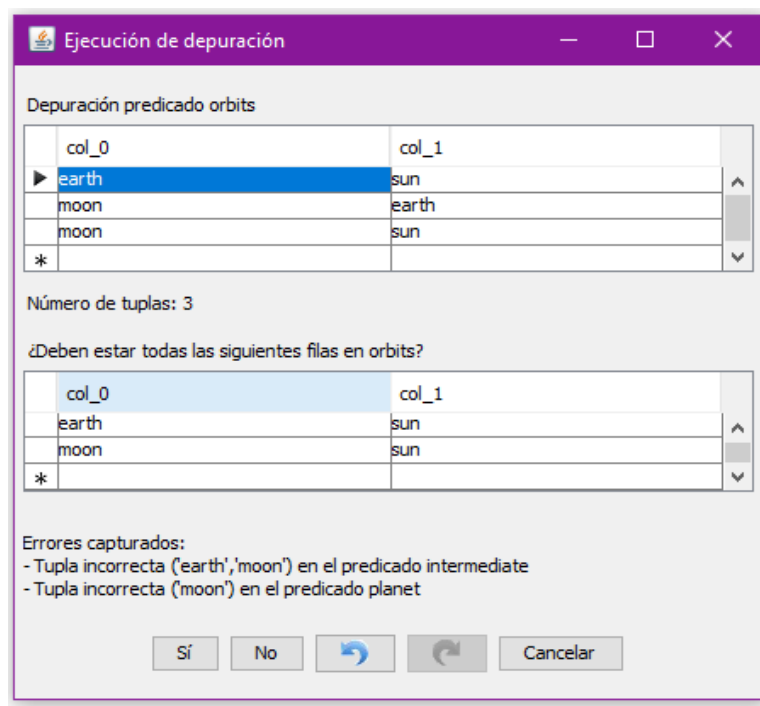


Figura D-39 Pregunta 6 depuración gráfica

Respondemos a esta sexta pregunta seleccionando el botón "Sí", ya que:

- La Tierra (earth) orbita alrededor del Sol (sun): la primera tupla (earth,sun) es correcta.
- La Luna (moon) orbita alrededor del Sol (sun): la segunda tupla (moon, sun) es correcta.

Tras responder a esta sexta pregunta, aparece una ventana emergente que nos indica la siguiente información:

- El fin de la depuración en su título.
- Que se ha encontrado el predicado erróneo. En concreto, indica que el predicado erróneo es "intermediate".
- Los errores detectados y marcados durante la depuración.

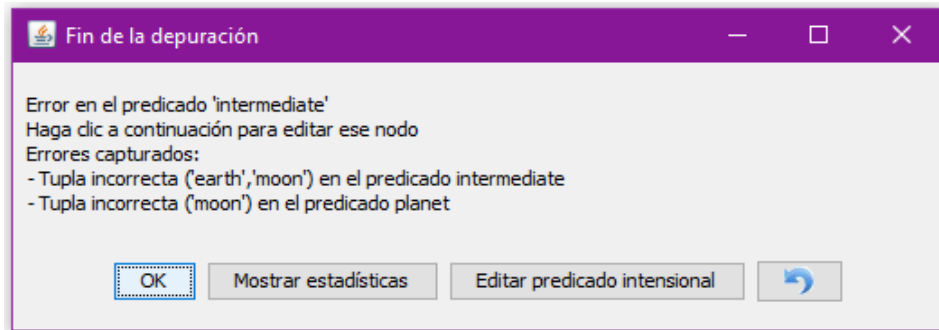


Figura D-40 Ventana fin depuración

La ventana de fin de depuración anterior nos permite realizar las siguientes acciones:

- 1) Aceptar la resolución pulsando el botón "OK" en cuyo caso no se actualizará la base de datos del sistema para corregir información errónea almacenada en la misma referente al predicado detectado como erróneo.
- 2) Consultar las estadísticas de la depuración seleccionando el botón "Mostrar estadísticas".

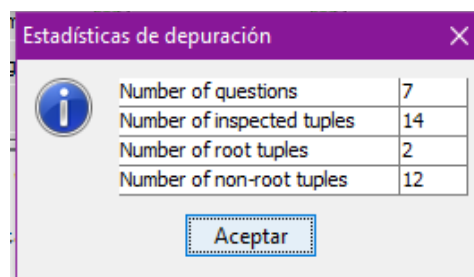


Figura D-41 Estadísticas depuración

Las estadísticas nos indican que el depurador ha tenido que realizar hasta siete preguntas al usuario (Number of questions) hasta dar con la causa del error.

A simple vista este hecho nos puede parecer que la depuración ha sido menos eficiente que la depuración manual por consola que solo ha requerido realizar seis preguntas al usuario para encontrar el origen del error, pero esto tiene una

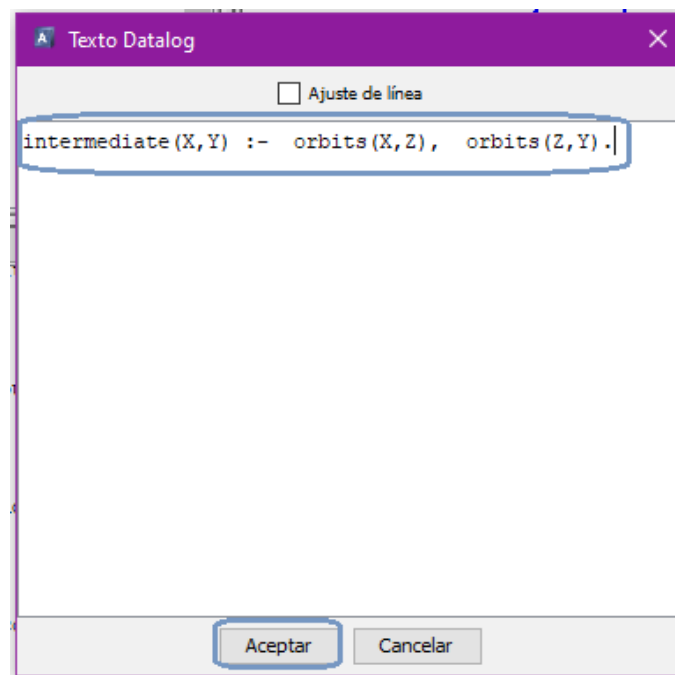
explicación:

La pregunta adicional contabiliza al marcar el nodo con etiqueta "orbits/2" como válido a través de su menú contextual. Por tanto, esta cuenta adicional puede ser ignorada.

3) Editar en la base de datos la definición del predicado intensional detectado como erróneo (intermediate) seleccionando el botón "Editar predicado intensional".

En concreto, permite modificar la regla que define "intermediate" desde una ventana de edición. Al realizar la modificación de la regla y pulsar el botón "Aceptar" dos veces para guardar los cambios, se eliminará de la base de datos del sistema la regla antigua errónea que definía "intermediate" y se añadirá la nueva regla:

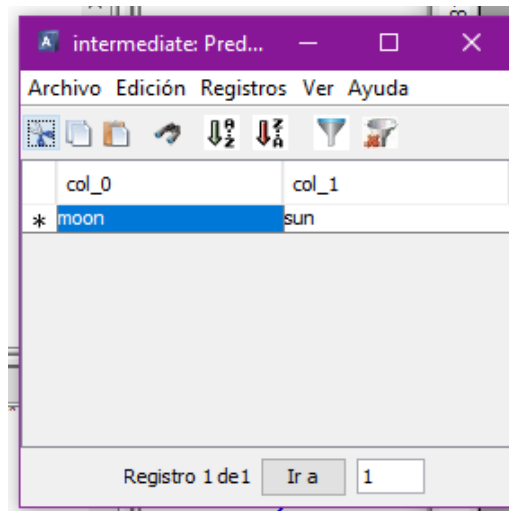
Regla antigua	intermediate(X,Y) :- orbits(X,Z), orbits(Y,Z).
Nueva regla	intermediate(X,Y) :- orbits(X,Z), orbits(Z,Y).



*Figura D-42 Editor predicado intensional*

Tras esta actualización de la base de datos del sistema, si damos clic derecho sobre el nodo "intermediate/2" y seleccionamos la opción "Abrir" de su menú contextual para consultar su contenido, podremos observar que únicamente está compuesto de tuplas correctas y las tuplas erróneas ya no pertenecen al mismo:

- La Luna (moon) orbita alrededor del Sol (sun) y, además, entre la Luna y el Sol hay un objeto intermedio que es la Tierra, ya que la Luna orbita alrededor de la Tierra y esta última también orbita alrededor del Sol. Por tanto, la tupla "moon,sun" debe pertenecer a "intermediate".
- Las tuplas "earth,earth", "earth,moon" y "moon,moon" no debían formar parte del contenido de "intermediate", ya que no hay objetos intermedios orbitados por el primer elemento de la tupla que a su vez estén orbitando alrededor del segundo elemento de la tupla.



*Figura D-43 Corrección intermediate*

4) Revertir la última acción aplicada y retornar a la pregunta número seis que nos realizaba el depurador seleccionando el botón "Deshacer" disponible y habilitado en esta ventana de fin de la depuración.

Adicionalmente, antes de aceptar la resolución pulsando el botón "OK" de la ventana de fin de la depuración:

- Podemos posicionarnos con el ratón sobre cualquier nodo del grafo para tener acceso a los detalles de su depuración: estado final del nodo y, cuando aplique, errores detectados en el mismo.
- Podemos observar el color de los diferentes nodos del grafo para conocer su estado final tras su depuración.

Estado	Color
Desconocido	Gris
Seleccionado	Amarillo

Rojo	Erróneo
Verde	Válido
Naranja	No válido

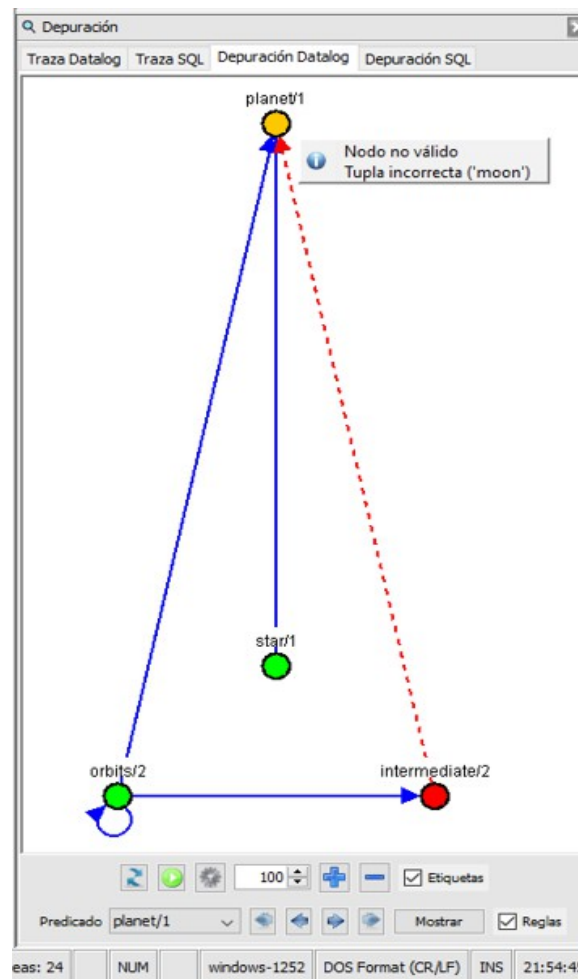


Figura D-44 Detalles gráficos fin depuración

Como hemos visto, durante el proceso de depuración se ha podido apreciar de forma práctica las ventajas que ofrece la depuración gráfica.

## 4. MEJORAS EN ACIDE

---

En las siguientes secciones del apartado actual detallo las mejoras que he implementado en ACIDE.

### 4.1. Bloqueo del panel de depuración

#### Necesidad que motiva la realización de la tarea

Mientras la consola DES se encontraba procesando comandos, los paneles de traza y depuración se podían utilizar. Esta situación provocaba conflictos como el siguiente:

Supongamos que un usuario que se encuentra trabajando con ACIDE procesa un script SQL. Por ejemplo, procesa el script de ejemplo SQL "family.sql" mediante el envío a la consola DES del comando `"/process .\examples\family.sql"`.

Posteriormente, introduce por consola el comando `"/debug_sql ventas_cruzadas"` para comenzar una sesión de depuración SQL de la vista "ventas\_cruzadas".

Si el usuario se dirige al panel de depuración SQL para seleccionar una de las vistas del script de ejemplo "family.sql" que ha procesado, en lugar de sugerirle el listado de vistas cargado, se acoplará un mensaje de error y la información de depuración que actualmente se está mostrando por consola.

Esto es debido a que, cuando el usuario despliega el selector de vistas, internamente se envía el comando `"/tapi /list_views"` a la consola DES para obtener las vistas creadas lo cual provoca un error por el siguiente motivo:

En ese momento, en la consola DES, hay un proceso activo esperando una respuesta relacionada con la sesión de depuración iniciada y el comando `"/tapi /list_views"` no es una respuesta válida que se admita para la sesión de depuración.

En la siguiente imagen se ha reproducido el escenario que se acaba de describir para mostrar gráficamente el error que se produce.

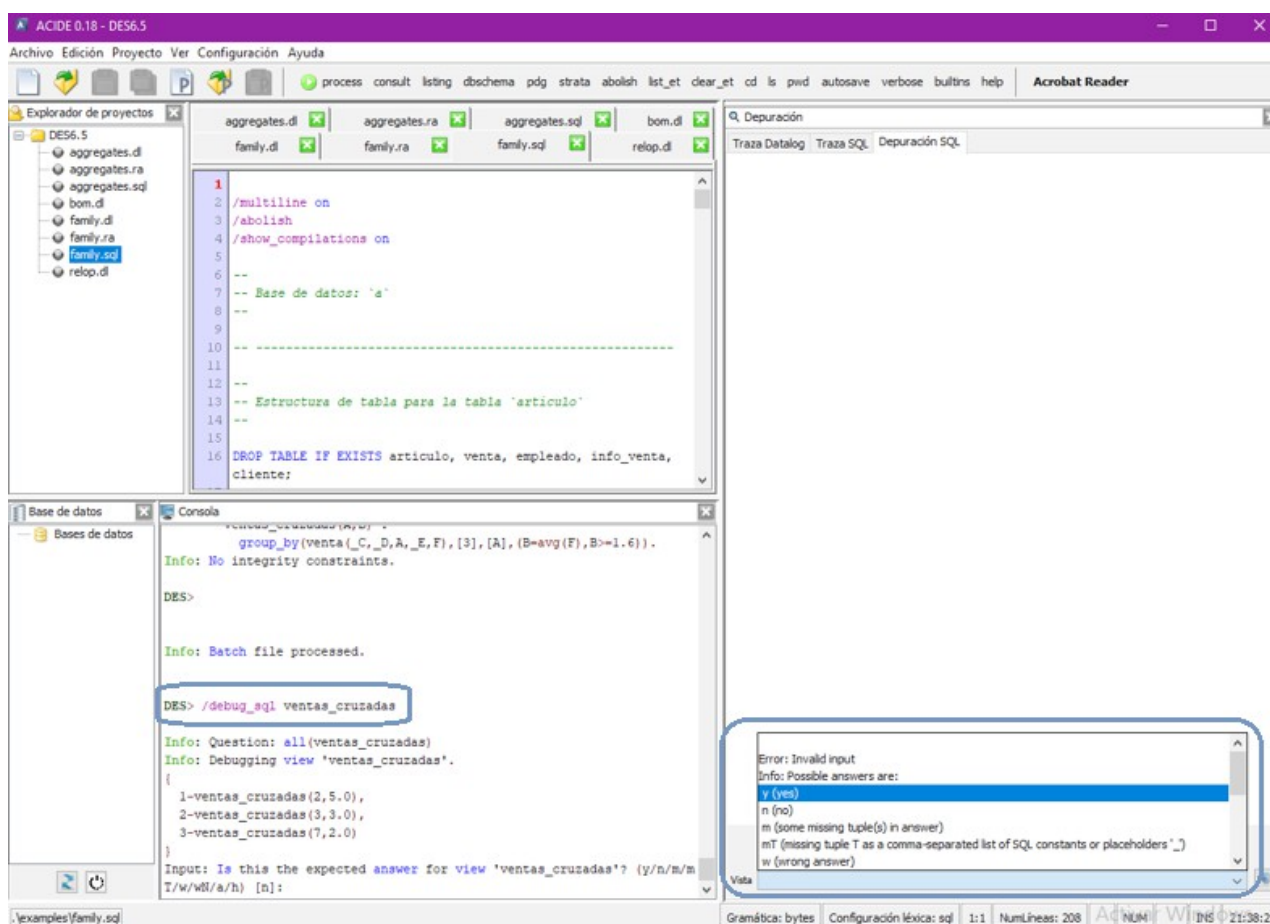


Figura M-1 Error procesamiento de comandos

## Solución implementada

Para evitar conflictos como el que se acaba de describir, se determina que la mejor solución es bloquear el panel de depuración en su totalidad mientras haya un proceso activo (comandos procesándose) en la consola DES.

Para implementar esta solución, cada vez que la consola DES se encuentra procesando un comando, se bloquea tanto el panel de control (cada botón, casilla, selector, etc.) que hay en cada panel de traza y depuración como el lienzo/canvas donde se dibuja el grafo a depurar.

La lógica aplicada para deducir que la consola DES se encuentra procesando un comando es analizar cuál es el contenido de la última línea de la consola y, si no es equivalente a los inductores "DES>", "DES-SQL>", "DES-RA>", "DES-TRC>", "DES-DRC>" o "FDES>", se deduce que se encuentra procesando un comando.

Además, si el usuario ubica el cursor del ratón sobre el panel de depuración, el

icono del cursor cambiará a una rueda de espera para informar al usuario de forma gráfica que las funcionalidades del panel de depuración, en ese momento, no están disponibles.

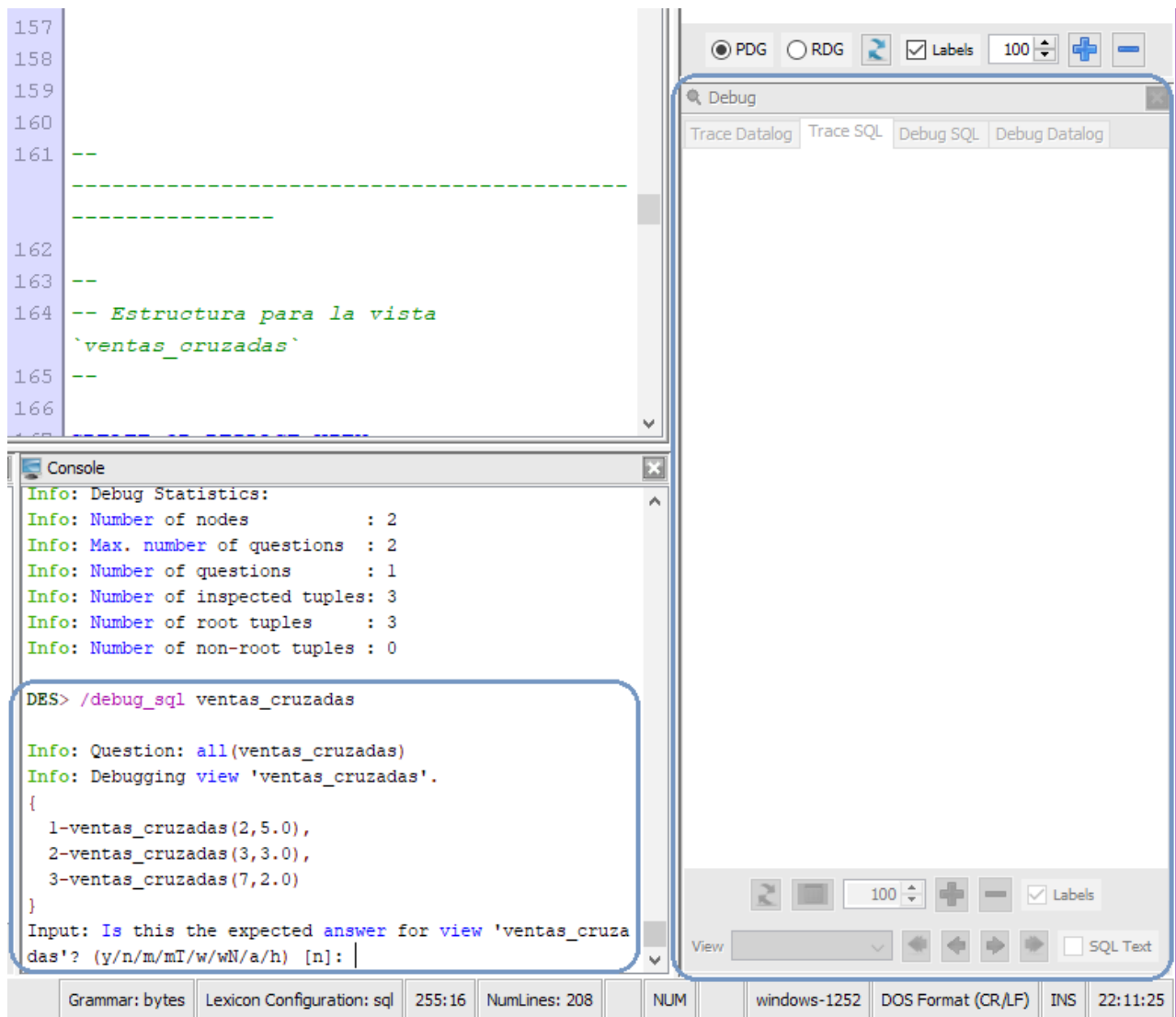


Figura M-2 Bloqueo panel de depuración

Los elementos de cada panel de traza y depuración (panel de control y lienzo) se desbloquearán cuando la última línea de la consola únicamente contenga alguno de los siguientes inductores, ya que se asumirá que la consola ya no se encuentra procesando ningún comando:

**DES>, DES-SQL>, DES-RA>, DES-TRC>, DES-DRC> o FDES>**



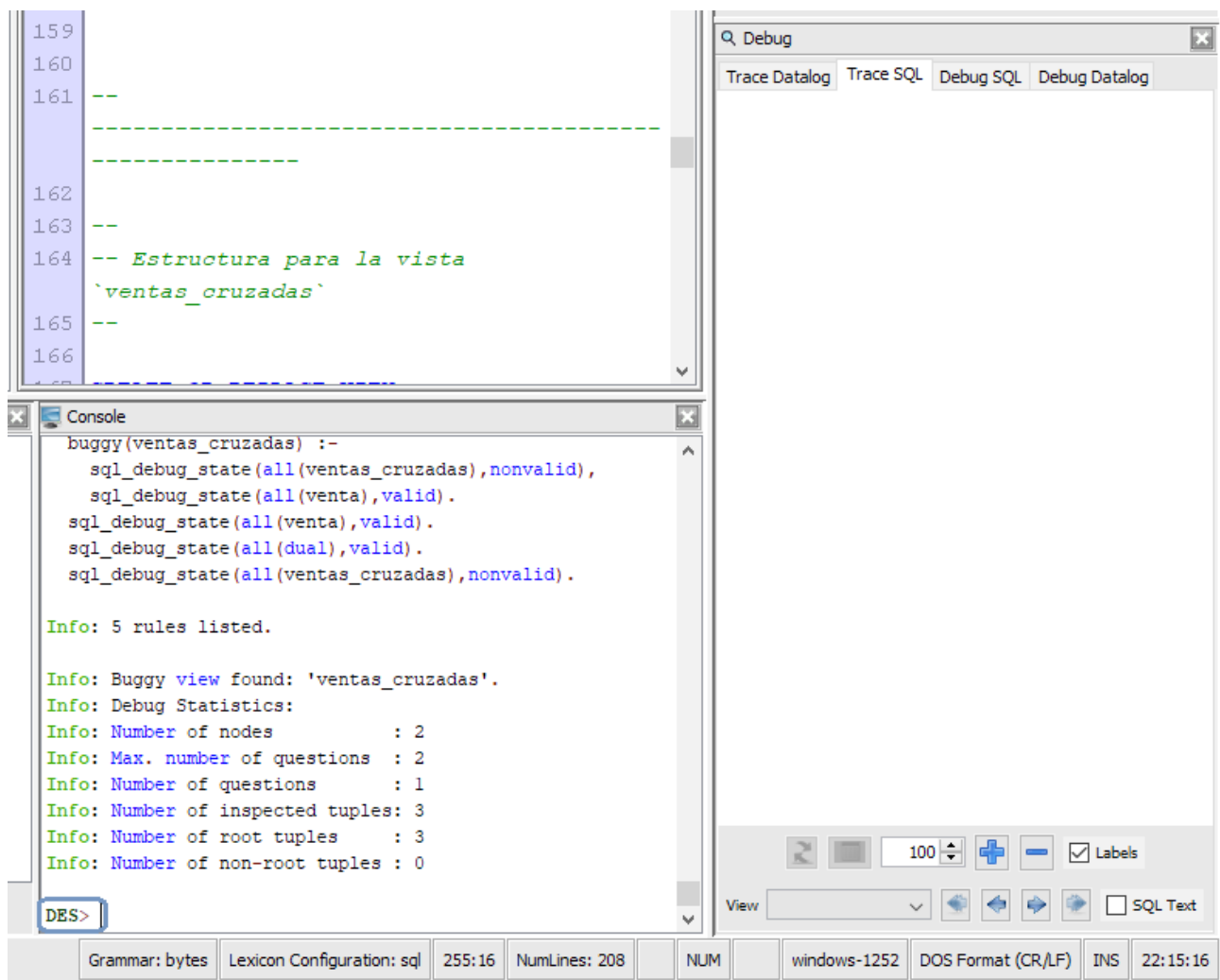


Figura M-3 Inductor DES

## 4.2. Cambio de idioma dinámico

### Necesidad que motiva la realización de la tarea

Durante la ejecución de ACIDE, si se cambia de idioma (Inglés, Francés o Español), el idioma de las etiquetas descriptivas de los elementos (botones, etc.) de los paneles de traza, base de datos y depuración no se actualiza al nuevo idioma y es necesario reiniciar ACIDE para que se apliquen los cambios en el idioma. Por ejemplo:

Como muestra la siguiente imagen, aunque se ha efectuado el cambio de idioma de "Español" a "Inglés" con ACIDE en ejecución, la localización al idioma Inglés no se ha aplicado en la etiqueta (texto descriptivo) asociada al botón de refresco del

panel de base de datos (Actualizar) pero sí al título del panel (Database).

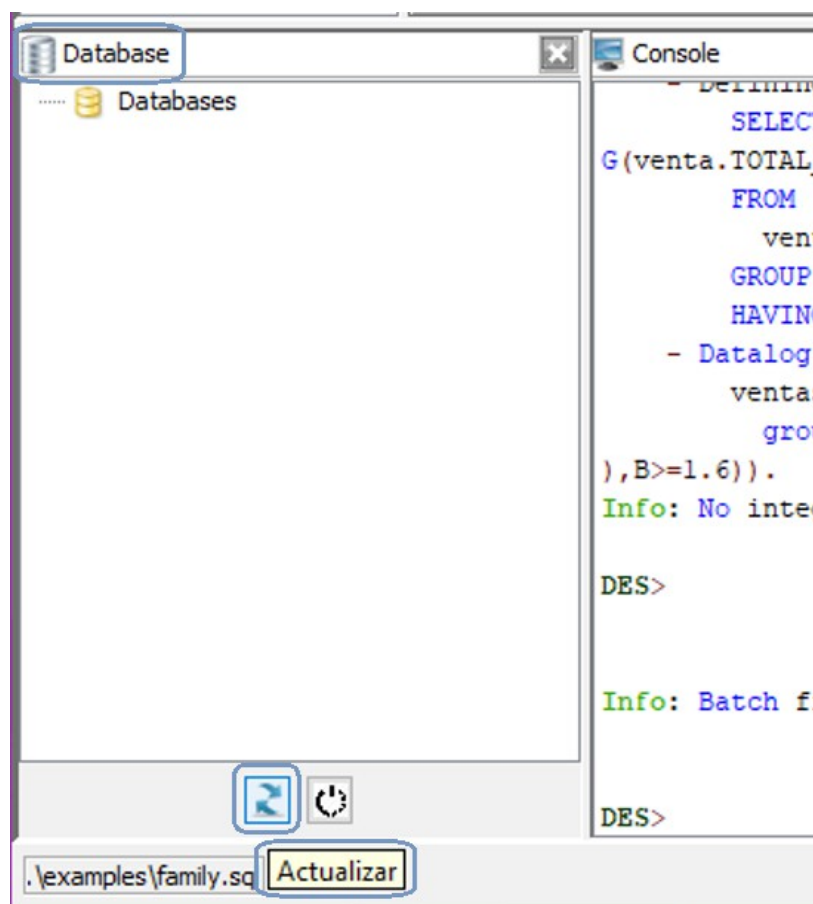


Figura M-4 Cambio dinámico de idioma

### Solución implementada

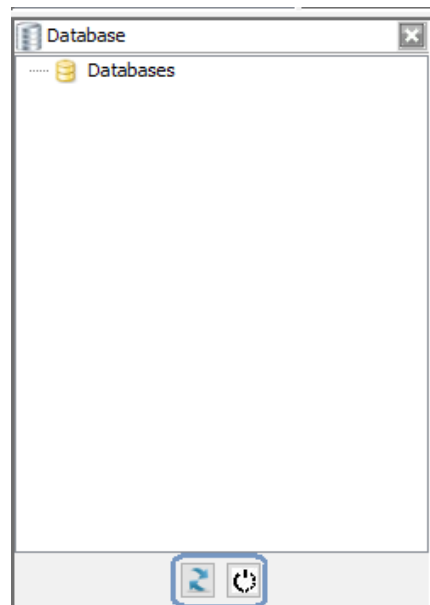
Se han añadido más actualizaciones de idiomas a aplicar cada vez que se lanza el evento encargado de realizar la localización de idioma a nivel global en ACIDE por haber realizado un cambio de idioma desde la siguiente sección de ACIDE:

Pestaña "Configuración" -> apartado "Idioma"

En concreto, se ha aplicado la localización dinámica a los siguientes elementos:

- Panel de base de datos

Se ha aplicado la localización dinámica a las etiquetas asociadas a los botones de refresco y reinicio de dicho panel.



*Figura M-5 Localización dinámica en panel de base de datos*

- Panel de traza SQL

Se ha aplicado la localización dinámica a las etiquetas asociadas a los siguientes elementos: selector de una vista, casilla que permite indicar si se quieren visualizar o no las etiquetas asociadas a cada nodo y la casilla que permite indicar si se quiere visualizar o no la sentencia SQL que define una vista.

Se ha aplicado la localización dinámica a las etiquetas descriptivas asociadas a los botones de refresco y de mostrar vista seleccionada.

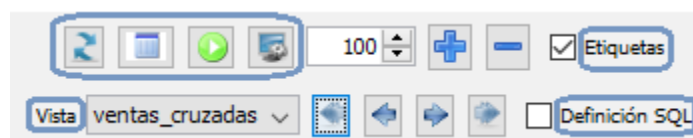


*Figura M-6 Localización dinámica en panel de traza SQL*

- Panel de depuración SQL

Se ha aplicado la localización dinámica a las etiquetas asociadas a los siguientes elementos: selector de una vista, casilla que permite indicar si se quieren visualizar o no las etiquetas asociadas a cada nodo y la casilla que permite indicar si se quiere visualizar o no la sentencia SQL que define una vista.

Se ha aplicado la localización dinámica a las etiquetas descriptivas asociadas a los botones de refresco, mostrar vista seleccionada, inicio y configuración de una sesión de depuración.



*Figura M-7 Localización dinámica en panel de depuración SQL*

### 4.3. Mostrar iconos de idiomas

#### Necesidad que motiva la realización de la tarea

Cuando el proyecto ACIDE se ubica en una ruta formada por carpetas donde al menos una de ellas, en su nombre, contiene espacios, cuando se ejecute la aplicación "acide.jar" contenida en el proyecto ACIDE, si el usuario se dirige a la siguiente ubicación no verá los iconos asociados a cada idioma (inglés, español y francés):

Pestaña "Configuration" -> apartado "Language"

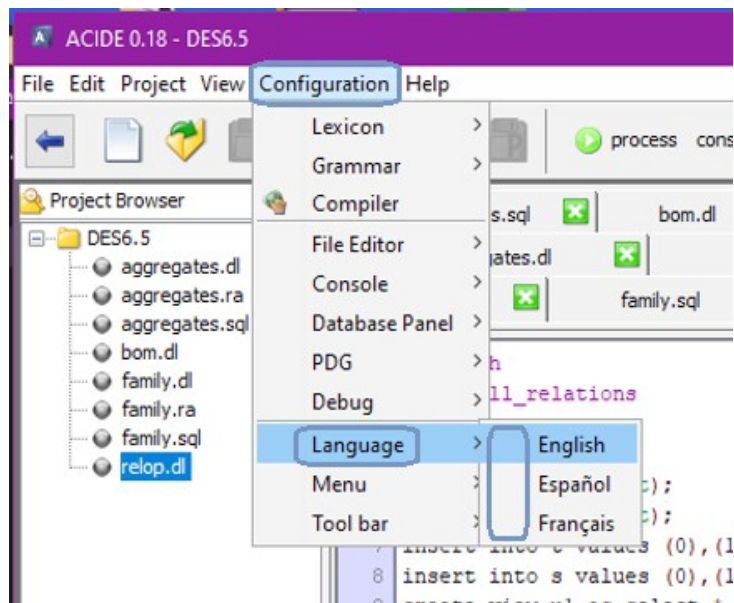


Figura M-8 Menú de idiomas sin iconos

Un ejemplo de una ruta que contiene espacios sería la siguiente:

"C:\Users\Admin\Desktop\Proyecto 2\Acide"

Como se observa en la ruta anterior, la carpeta llamada "Proyecto 2" contiene un espacio entre la palabra "Proyecto" y el número "2".

#### Solución implementada

Se ha modificado el código del proyecto ACIDE para que se cree el icono a partir de la ruta que lleva a la imagen asociada al idioma.

Esta implementación siempre obtiene la imagen porque se trabaja con la clase "ImageIcon" que es capaz de encontrar ficheros cuyo nombre contiene espacios. Se considerará que la imagen ha sido encontrada si el ancho del icono es mayor a "0", ya que significará que el icono ha podido ser creado.

Tras aplicar esta implementación, aunque el proyecto ACIDE se ubique en una ruta con espacios, se mostrará el icono asociado a cada idioma.

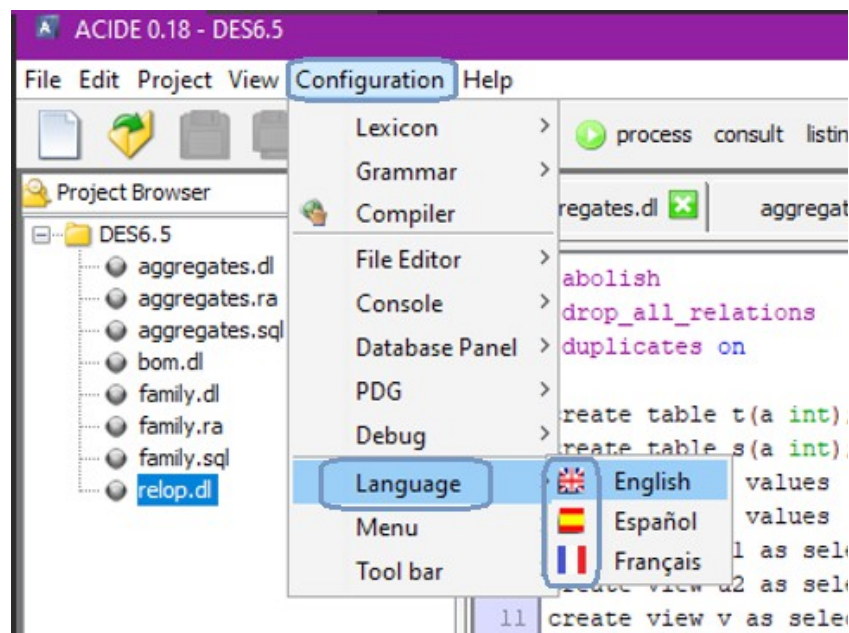


Figura M-9 Menú de idiomas con iconos

#### 4.4. Ajuste de la sentencia SQL a editar

##### Necesidad que motiva la realización de la tarea

La sentencia SQL que crea una vista no se ajusta a los límites del editor de texto desde el cual se puede modificar cuando este es redimensionado. En esta situación, el usuario tendrá que realizar un desplazamiento horizontal para visualizar la sentencia SQL al completo. Por ejemplo:

Supongamos que una vista llamada "ventas\_cruzadas" se forma a partir de la siguiente sentencia SQL:

```
CREATE OR REPLACE VIEW ventas_cruzadas(ID_EMPLEADO, VENTA_CRUZADA) AS select  
'venta'.ID_EMPLEADO AS ID_EMPLEADO, avg('venta'.TOTAL ARTICULOS) AS  
'VENTA_CRUZADA' from 'venta' group by 'venta'.ID_EMPLEADO having  
(avg('venta'.TOTAL ARTICULOS) >= 1.6);
```

Si el usuario selecciona la opción "Edit View" al dar clic derecho sobre el nodo asociado a la vista "ventas\_cruzadas" para editar su sentencia SQL y redimensiona el editor de la ventana emergente que aparece, la sentencia no se ajustará

dinámicamente a los nuevos límites del editor.

En esta situación, tendrá que realizar un desplazamiento horizontal para visualizarla al completo.

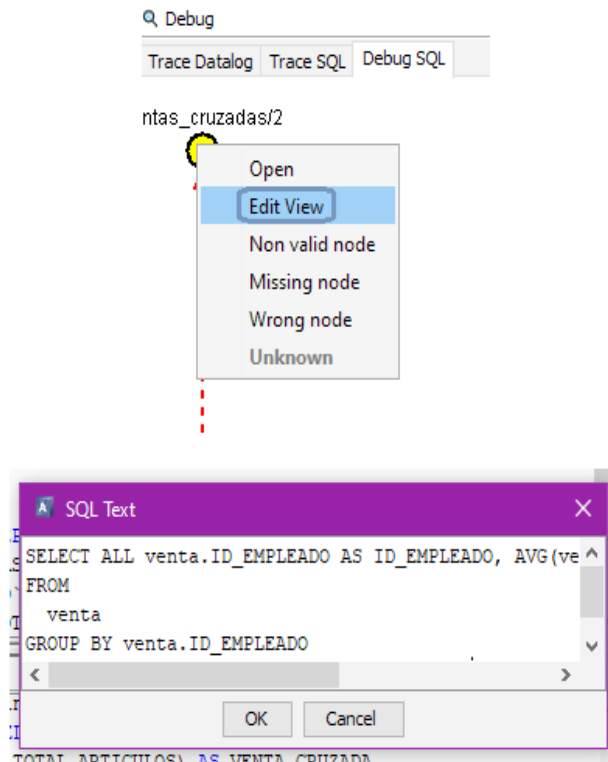


Figura M-10 Editor sentencia SQL sin ajuste de línea

### Solución implementada

Para implementar esta tarea, en el editor de texto de una sentencia SQL, se ha añadido una casilla de verificación llamada "Ajuste de línea" (Word Wrap).

Si el usuario selecciona la casilla de verificación "Ajuste de línea", la sentencia SQL se ajustará a las dimensiones del editor y no tendrá que realizar un desplazamiento horizontal para visualizarla al completo.

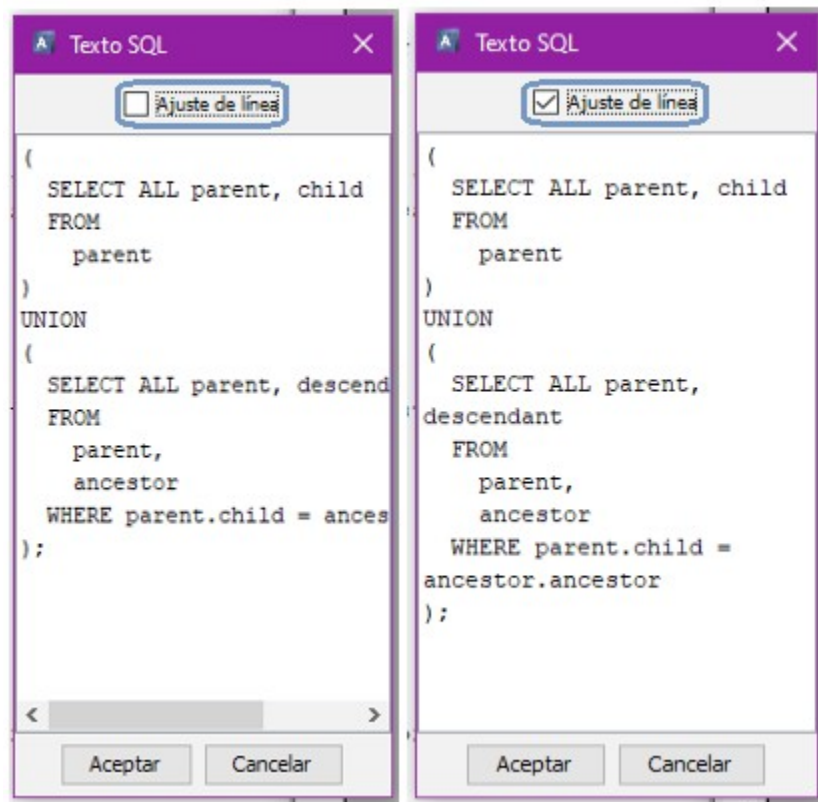


Figura M-11 Diferencia opción Ajuste de línea

## 4.5. Mostrar información de depuración asociada a un nodo

### Necesidad que motiva la realización de la tarea

Durante una sesión de depuración SQL, antes de que finalice, el usuario no dispone de una opción que le permita consultar el resultado de la depuración de un nodo concreto: estado con el que ha sido marcado (válido, no válido o erróneo), si se ha indicado que contenía una tupla errónea, etc.

### Solución implementada

Se ha construido un componente que actúa como una descripción emergente (tooltip). Cuando el usuario, durante una sesión de depuración SQL, se posiciona con el ratón sobre un nodo depurado, se mostrará dinámicamente la siguiente información del nodo:

- Su estado tras la depuración: válido, no válido o erróneo.
- La tupla faltante en caso de haberla introducido o la tupla errónea en caso de haberla marcado.



Como se muestra en la siguiente imagen, si el usuario se posiciona sobre el nodo depurado que representa la vista "parent", se le indica que el nodo ha sido marcado como erróneo y que, en él, se ha encontrado que la tupla ('amy','fred') es incorrecta.

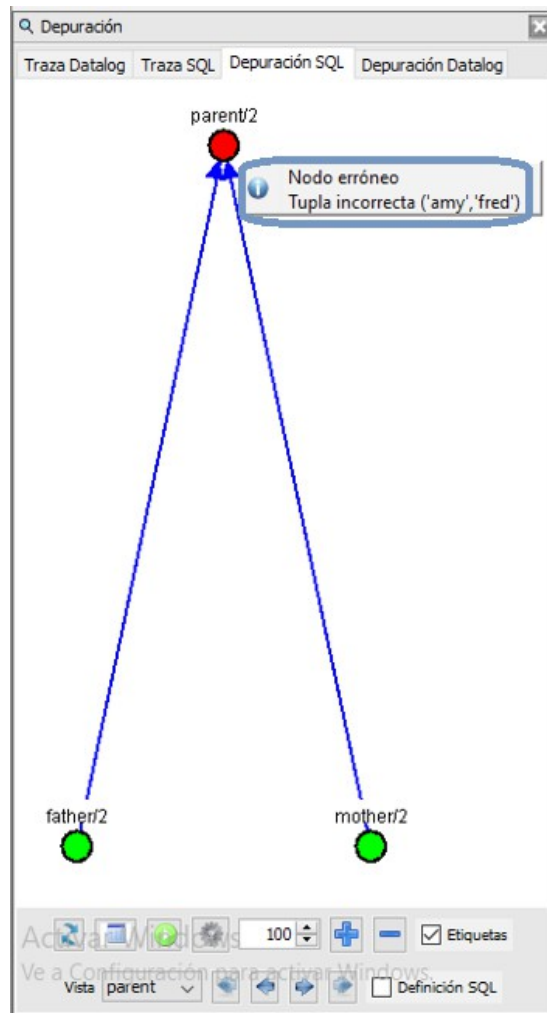


Figura M-12 Información de depuración asociada a un nodo

## 4.6. Colorear los nodos tabla en función de la configuración aplicada

### Necesidad que motiva la realización de la tarea

Cuando el usuario configura las opciones de una sesión de depuración SQL, si marca la opción de "confiar en las tablas", el color de los nodos tabla del grafo cargado no se actualiza a verde para indicar que, por defecto, se confía en su contenido (tuplas con datos correctos).

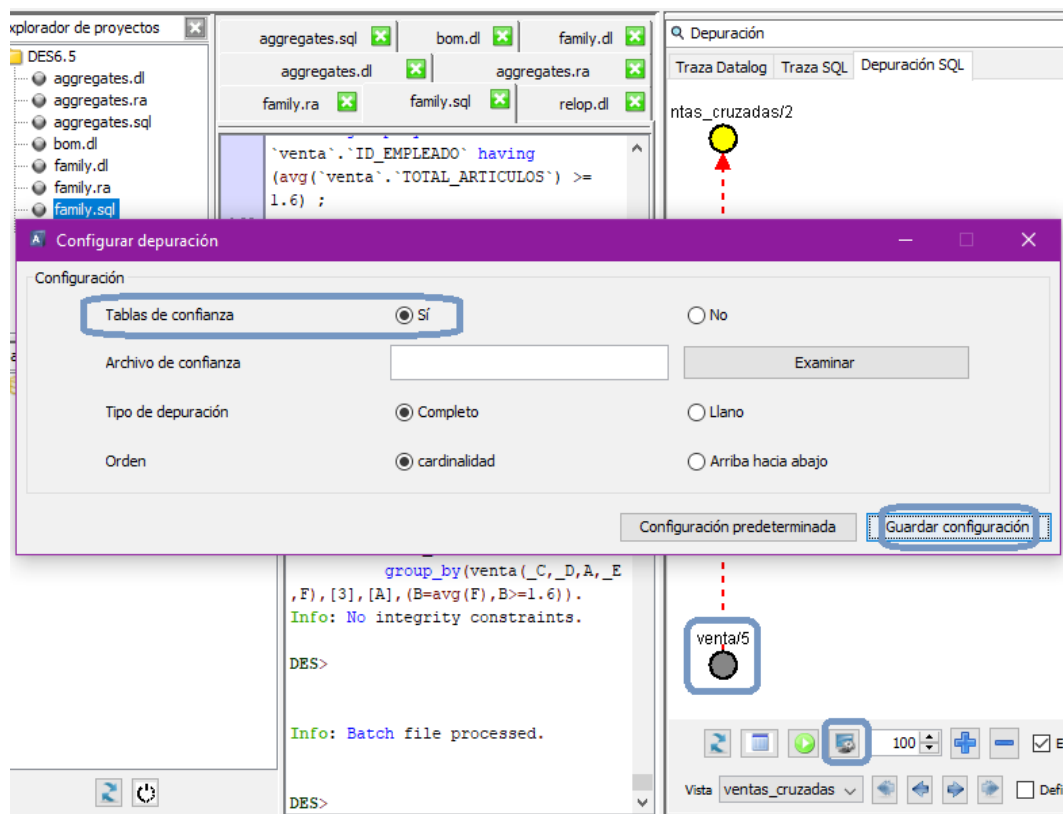


Figura M-13 Ventana configuración sesión depuración SQL

### Solución implementada

Se ha actualizado el código del listener que captura el evento que se lanza cuando el usuario selecciona el botón "Guardar configuración" de la ventana de configuración de depuración.

Si el usuario, en la ventana de configuración de la depuración SQL, selecciona el botón de radio "Sí" para la opción "Tablas de confianza" y pulsa el botón "Guardar configuración", automáticamente el color de los nodos tabla del grafo se actualizará a verde.

## 4.7. Obtener formateada la sentencia SQL de una vista

### Necesidad que motiva la realización de la tarea

Cuando se procesa un script SQL y, en el selector del panel de traza o depuración se selecciona una vista del script SQL procesado, si la casilla "Definición SQL" se

encuentra marcada o se marca, la sentencia SQL de la vista seleccionada se obtiene mediante el envío a la consola DES del siguiente comando:

```
/dbschema <nombre_vista>
```

Sin embargo, el comando `"/dbschema <nombre_vista>"` obtiene la sentencia SQL sin aplicarle un formato que facilite su análisis y manejo en el código Java del proyecto ACIDE.

Por este motivo, surge la necesidad de obtener la sentencia SQL de una vista a través de la API Textual que proporciona DES (TAPI), ya que la salida que genera como respuesta a los comandos que recibe sí está formateada para facilitar su análisis y manejo en Java.

#### Solución implementada

Se ha modificado el código del proyecto ACIDE para que, en lugar de enviar a DES el comando `"/dbschema <nombre_vista>"` para obtener la sentencia SQL de una vista, se envíe el siguiente comando que obtiene la sentencia SQL formateada para facilitar su tratamiento:

```
/tapi /dbschema "<nombre_vista>"
```

Como se muestra en la siguiente imagen, se quiere obtener la sentencia SQL de la vista llamada "buenas\_ventas".

Si se envía el comando `"/dbschema"` directamente a la consola DES sin pasar a través de la API Textual (TAPI), la salida obtenida no tiene aplicado un formato (resultado de la izquierda).

Sin embargo, si el comando se envía a través de la API Textual que proporciona la consola DES (`/tapi /dbschema`), la salida obtenida estará formateada (resultado de la derecha).

```
DES> /dbschema buenas_ventas
```

```
Info: Database '$des'
```

```
Info: View:
```

```
A buenas_ventas(ID_EMPLEADO:int,NUMERO_VENTAS:int)
```

```
- Defining SQL statement:
```

```
    SELECT ALL venta.ID_EMPLEADO AS ID_EMPLEADO, COUNT(0) AS NUMERO_VENTAS  
    FROM  
        venta  
    WHERE venta.TOTAL_VENTA >= 50  
    GROUP BY venta.ID_EMPLEADO;
```

```
- Datalog equivalent rules:
```

```
buenas_ventas(A,B) :-  
    group_by((venta(_C,_D,A,E,_F),E>=50), {A},B=count(0)).
```

```
DES> /tapi /dbschema "buenas_ventas"
```

```
$view
```

```
$sql
```

```
buenas_ventas  
ID_EMPLEADO  
int  
NUMERO_VENTAS  
int
```

```
$
```

```
SELECT ALL venta.ID_EMPLEADO AS ID_EMPLEADO, COUNT(0) AS NUMERO_VENTAS  
FROM  
    venta  
WHERE venta.TOTAL_VENTA >= 50  
GROUP BY venta.ID_EMPLEADO;
```

```
$
```

```
buenas_ventas(A,B) :-  
    group_by((venta(_C,_D,A,E,_F),E>=50), {A},B=count(0)).
```

```
$
```

```
$
```

```
$
```

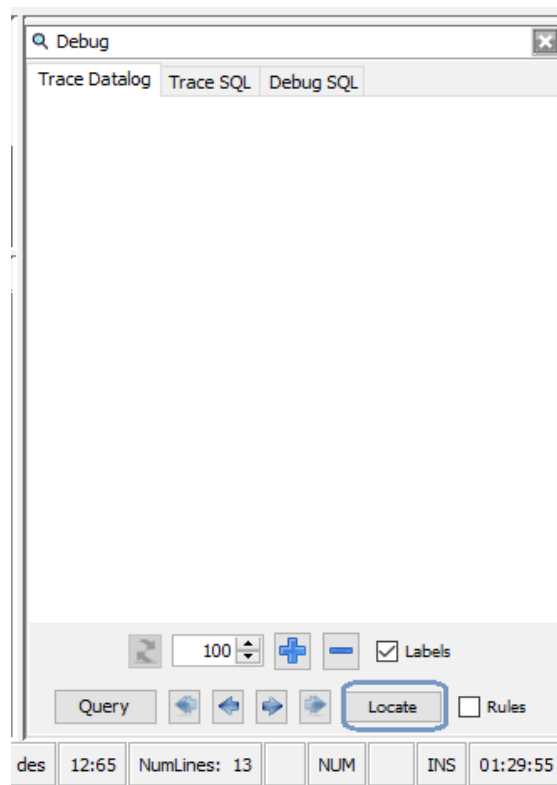
```
$
```

```
$
```

```
$
```

```
$get
```

Si no se aclara del algún modo la utilidad del botón "Locate" al usuario, su utilidad quedará confusa.

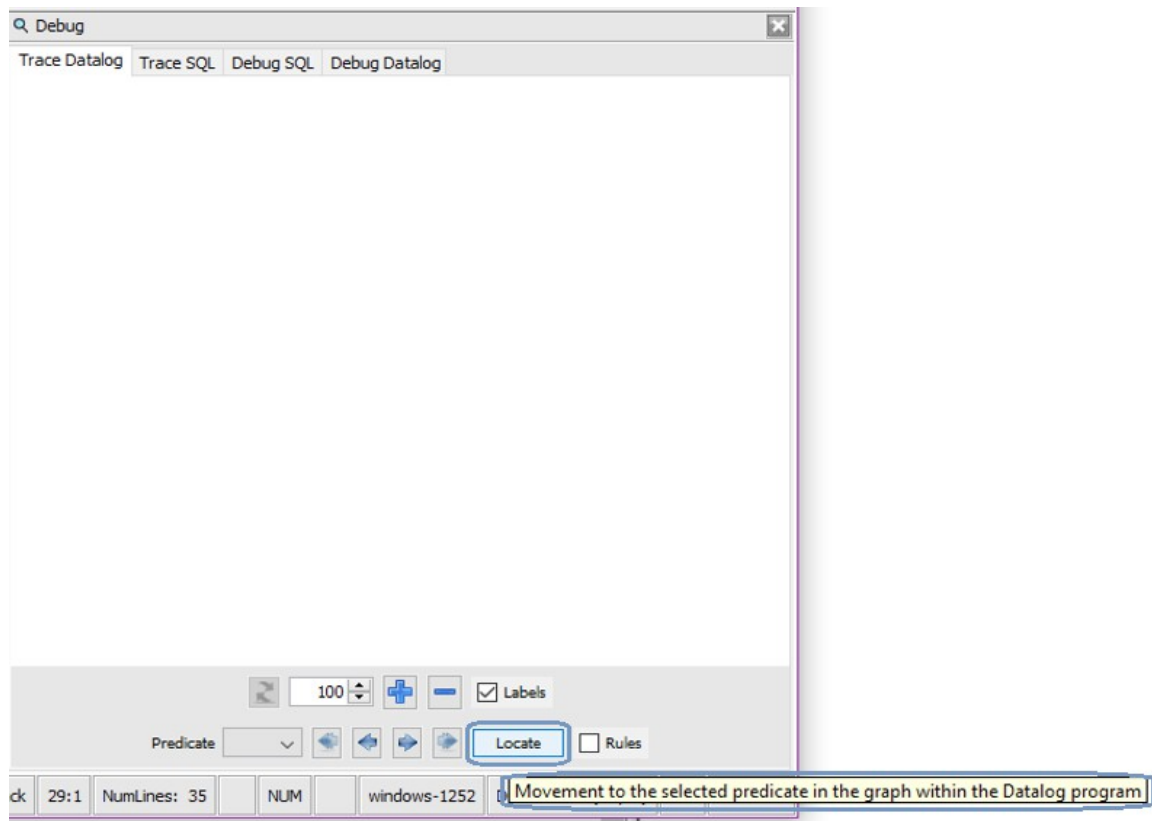


*Figura M-15 Botón Locate sin descripción*

### Solución implementada

Por cada elemento que hay en los paneles de traza y depuración (botones, casillas, etc.), se ha añadido una etiqueta emergente que describe su utilidad.

Cuando el usuario posicione el ratón sobre uno de estos elementos, aparecerá una etiqueta emergente que dará detalles sobre su utilidad.



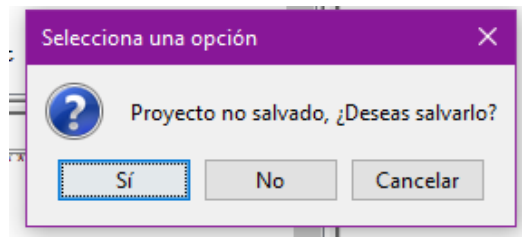
*Figura M-16 Botón Locate con descripción*

## 4.9. Cambios en la localización a los diferentes idiomas

### Necesidad que motiva la realización de la tarea

Los textos descriptivos que se utilizan a lo largo de ACIDE contienen palabras poco adecuadas para el contexto. Por ejemplo:

A la hora de cerrar ACIDE, si durante su ejecución se cambió de idioma, antes de finalizar ACIDE, aparecerá un cuadro de diálogo que preguntará si se desea salvar el proyecto con la nueva configuración aplicada (cambio de idioma). En esta situación, la pregunta adecuada sería si se desea guardar el proyecto, no salvar.

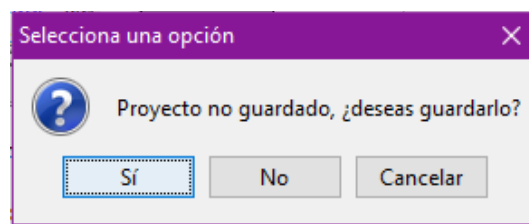


*Figura M-17 Localización incorrecta*

### Solución implementada

Para implementar esta tarea se han modificado los ficheros de idiomas del proyecto ACIDE para realizar los siguientes cambios:

- La cadena de texto "Fichero no salvado, ¿quieres salvarlo?" ha sido reemplazada por "Fichero no guardado, ¿quieres guardarlo?".
- La cadena de texto "Proyecto no salvado, ¿deseas salvarlo?" ha sido reemplazada por "Proyecto no guardado, ¿desea guardarlo?".



*Figura M-18 Localización correcta*

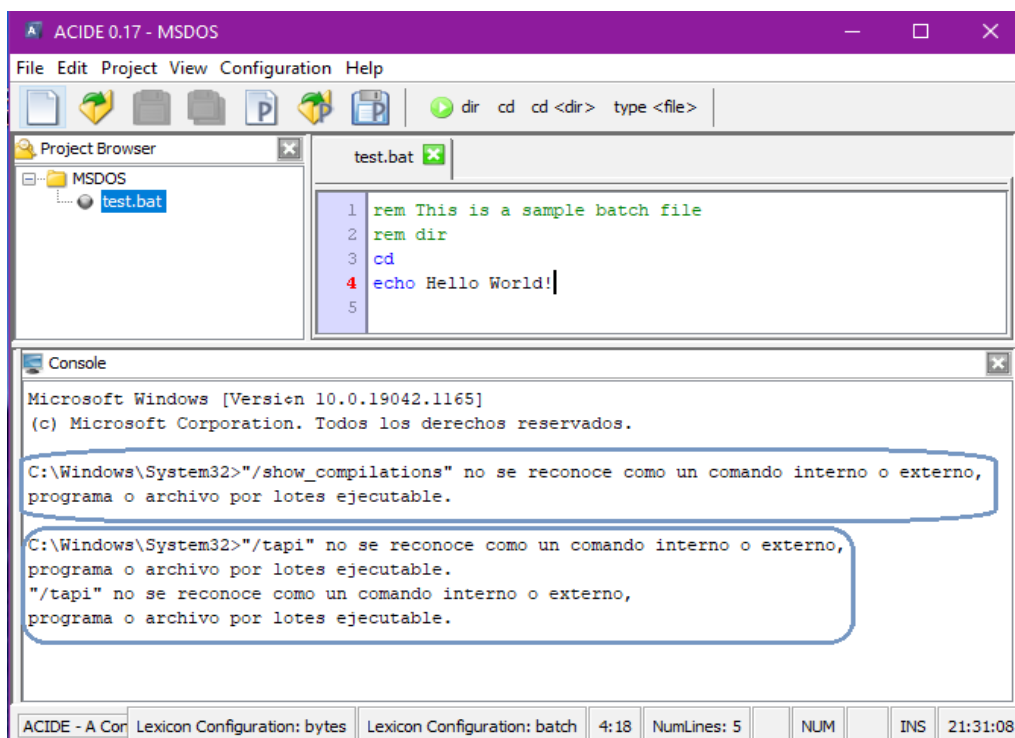
- La cadena de texto "Ventana de salvado de recursos creada" ha sido reemplazada por "Ventana de guardado de recursos creada".
- La cadena de texto "Ventana de salvado de recursos" ha sido reemplazada por "Ventana de guardado de recursos".
- La cadena de texto "Selecciona los recursos a salvar" ha sido reemplazada por "Selecciona los recursos a guardar".

## 4.10. Control de envío de comandos

Necesidad que motiva la realización de la tarea

Se emiten comandos `"/pdg"` y `"/tapi"` cuando ACIDE no usa la consola DES. Al ser `"/pdg"` y `"/tapi"` comandos específicos de DES, esto provoca que una consola que no es DES no sea capaz de reconocer dichos comandos e informe del error. Por ejemplo:

Cuando ACIDE ha sido configurado para utilizar la consola MSDOS de Windows, durante su uso, a la consola MSDOS se pueden enviar comandos DES (`"/pdg"` y `"/tapi"`) provenientes de otros paneles. En esta situación, la consola MSDOS informa que no ha sido capaz de interpretar/reconocer los comandos recibidos.



*Figura M-19 Comandos no reconocidos*

### Solución implementada

Se ha modificado la clase que representa el panel de la consola en ACIDE: no se podrán enviar comandos específicos de DES (`/pdg`, `/tapi`, etc.) a la consola configurada en ACIDE si esta no es DES (`des.exe` (en Window) y `des` (en Linux)).

## 4.11. Cambiar iconos

### Necesidad que motiva la realización de la tarea



Algunos elementos de los paneles tienen establecidos iconos poco intuitivos. Por ejemplo:

El botón que abre una ventana para configurar una sesión de depuración SQL, tiene establecido un icono que no deja claro al usuario su utilidad.



*Figura M-20 Icono configuración poco intuitivo*

### Solución implementada

Se han buscado iconos más representativos, se han ajustado sus dimensiones a los límites permitidos y se han añadido al proyecto ACIDE para que sustituyan a los viejos iconos.

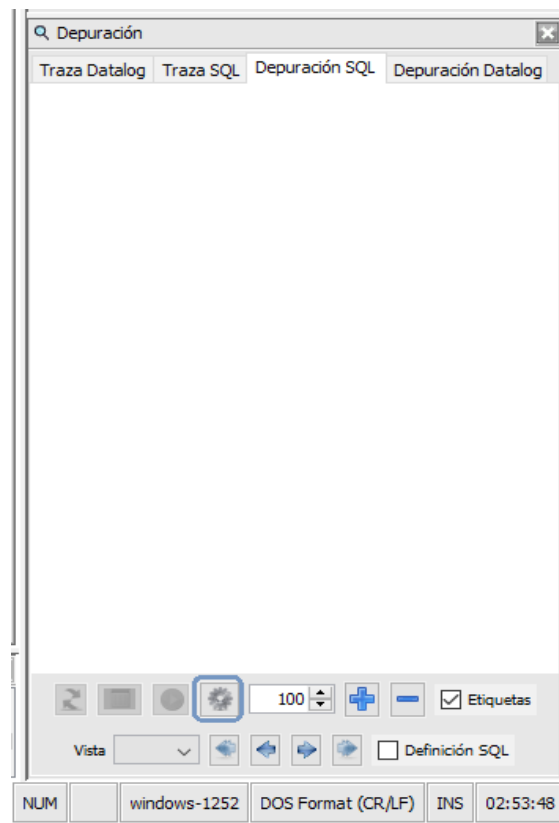


Figura M-21 Icono configuración representativo

## 4.12. Corrección de caracteres corruptos

### Necesidad que motiva la realización de la tarea

En ciertos textos descriptivos utilizados en las diferentes secciones de ACIDE hay caracteres corruptos. Por ejemplo:

Tras finalizar una sesión de depuración SQL, en la ventana emergente que aparece y que permite consultar los resultados de la depuración, hay palabras que contienen caracteres corruptos como la palabra "estadísticas" debido a que, en este caso, no se ha codificado adecuadamente la tilde de la vocal "i".

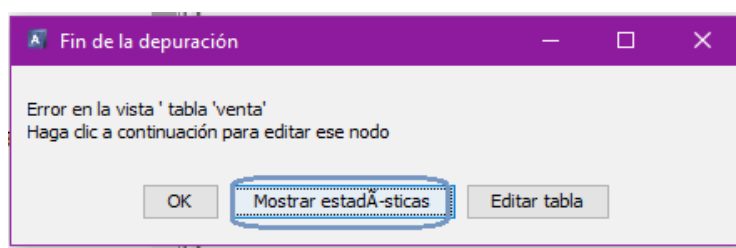


Figura M-22 Tildes mal codificadas

#### Solución implementada

Se han modificado los ficheros de idiomas del proyecto ACIDE para aplicar una codificación unicode. Por ejemplo:

El carácter "í" fue codificado a "\u00ed".

### 4.13. Añadir cadena deshacer-rehacer

#### Necesidad que motiva la realización de la tarea

Durante una sesión de depuración de una vista, el usuario no dispone de una opción que le permita revertir acciones previas que ha aplicado durante la misma: marcar una vista como no válida, introducir una tupla faltante, etc.

Así mismo tampoco dispone de una opción que le permita rehacer acciones que ya aplicó durante la depuración cuando ha verificado que sus acciones previas eran correctas.

Al disponer de opciones tanto para revertir como para rehacer acciones realizadas durante una sesión de depuración de una vista, se obtienen las siguientes ventajas:

- Opción de deshacer: Evita que la sesión deba comenzar desde el inicio, ya que se puede retornar a puntos previos donde se aplicó una acción incorrecta.
- Opción de rehacer: Permite volver al punto actual tras revisar que las acciones previas que se aplicaron fueron correctas.
- Ambas opciones reducen el tiempo que el usuario tarda en depurar una vista.

#### Solución implementada

Se ha implementado tanto una opción de "deshacer" como una opción de "rehacer" tanto en la ventana interactiva de depuración como en las opciones disponibles en el menú contextual de cada nodo:

- Ventana interactiva de depuración:

Como muestra la siguiente imagen, se han añadido los siguientes dos botones

adicionales a la ventana interactiva de depuración:

- Botón de "deshacer" una acción realizada (1):

Cuando el usuario pulsa el botón de "deshacer", se revierte la depuración del último nodo recorrido y este pasa a ser el siguiente nodo pendiente de depurar por el usuario.

La opción de deshacer nunca permitirá al usuario poder volver hasta el punto durante la depuración en el cual se debía depurar el nodo raíz, ya que desde el inicio se conoce su invalidez. Recordar que el nodo raíz, en el grafo de dependencias es aquel que representa la vista que se está depurando durante la sesión de depuración.

Nodos posteriores al nodo raíz que ya han sido depurados siempre permite alcanzarlos para volverlos a depurar.

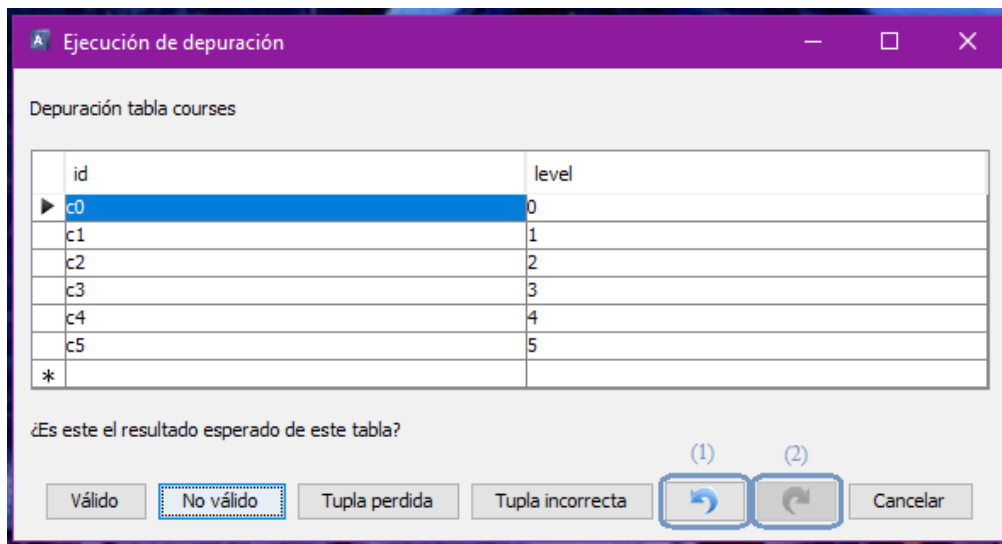
Así mismo también permite, una vez finalizada la sesión de depuración, volverla a restaurar para revertir acciones aplicadas durante la misma y, de este modo, cambiar los resultados finales de la depuración.

- Botón de "rehacer" una acción realizada (2):

Cuando el usuario pulsa el botón de "rehacer", automáticamente se volverá a aplicar la acción que anteriormente se aplicó al nodo actual que se está depurando.

Cuando el usuario ha deshecho acciones que aplicó y, en un determinado nodo recorrido con anterioridad, a la hora de depurarlo de nuevo, aplica una nueva acción, la opción de "rehacer" dejará de estar disponible, ya que el curso de la depuración cambia debido a que se podrán obtener resultados finales distintos por aplicar nuevas acciones.

Así mismo también permite volver hasta el punto en que se había finalizado la sesión de depuración con los mismos resultados finales.



*Figura M-23 Cadena deshacer-rehacer ventana depuración*

- Menú contextual de un nodo:

En el menú contextual de cada nodo también se han implementado las opciones de "deshacer" y "rehacer". Cuando el usuario se encuentra depurando un nodo, podrá acceder a estas dos opciones si da clic derecho con el ratón sobre el mismo.

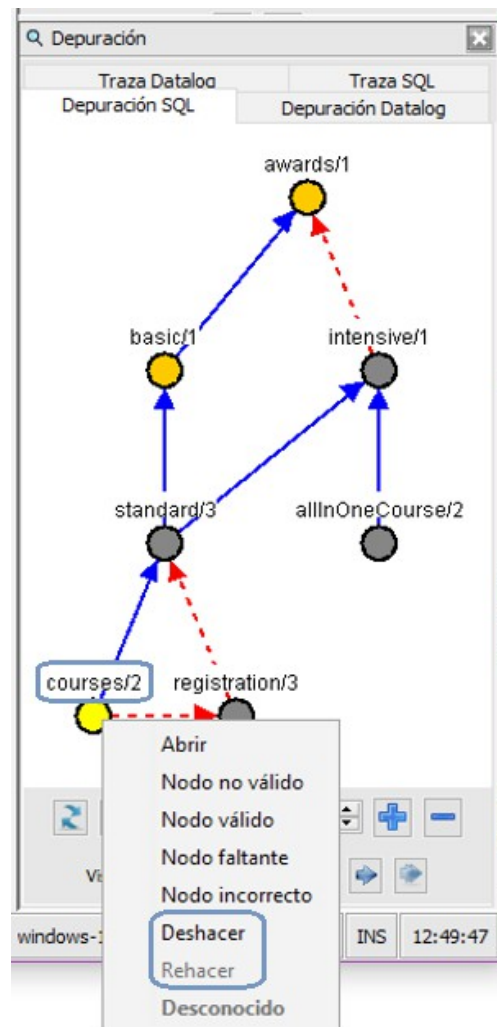


Figura M-24 Cadena deshacer-rehacer menú contextual

La disponibilidad de dichas opciones en cada paso de la depuración será la misma que se ha explicado en el punto anterior para la ventana interactiva de depuración.

Tras finalizar la sesión de depuración, como ocurría para la ventana interactiva de depuración, podrá volverla a restaurar (si su interés es aplicar nuevas acciones) dando clic derecho con el ratón sobre el nodo marcado como erróneo (nodo coloreado de rojo) y seleccionando su opción de "deshacer".

#### 4.14. Corrección de error al arrancar ACIDE

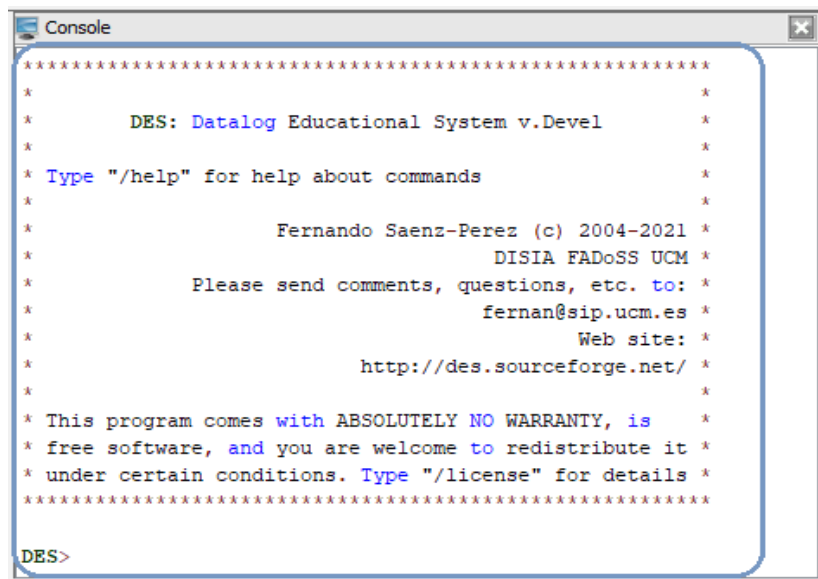
##### Necesidad que motiva la realización de la tarea

En ciertas ocasiones, al arrancar ACIDE, cuando está configurado para utilizar la consola DES, la consola de la interfaz no arranca correctamente, ya que en lugar de

mostrar el mensaje de bienvenida al sistema seguido del inductor DES muestra únicamente el resultado de la ejecución de un comando (salida \$success).

Esto provoca que haya que reiniciar la consola manualmente para que el inductor DES pase a estar disponible y se puedan enviar comando a consola a través de él.

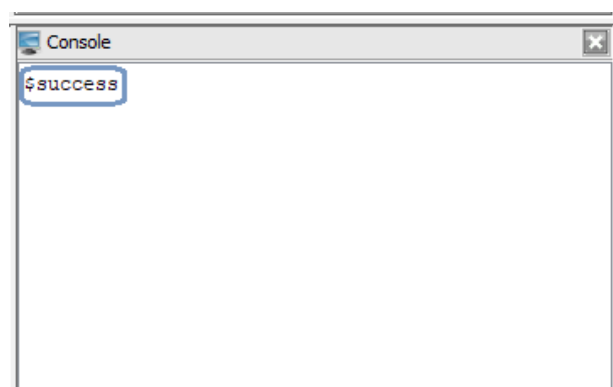
La siguiente imagen muestra la información que se espera que al arrancar muestre la consola de la interfaz ACIDE.



```
*****
*
*      DES: Datalog Educational System v.Devel
*
* Type "/help" for help about commands
*
*      Fernando Saenz-Perez (c) 2004-2021
*      DISIA FADoSS UCM
*      Please send comments, questions, etc. to:
*      fernan@sip.ucm.es
*      Web site:
*      http://des.sourceforge.net/
*
* This program comes with ABSOLUTELY NO WARRANTY, is
* free software, and you are welcome to redistribute it
* under certain conditions. Type "/license" for details
*****
DES>
```

*Figura M-25 Inicio correcto consola DES*

Sin embargo, en ciertas ocasiones, al arrancar, muestra la salida "\$success".



*Figura M-26 Inicio incorrecto consola DES*

### Solución implementada

Tras analizar la implementación del proyecto ACIDE, se detectó que, al arrancar

ACIDE, se emitían dos comandos a consola que provocaban que la consola de ACIDE no arrancara correctamente.

La salida por consola "\$success" era el resultado de haber emitido el comando `"/tapi /test_tapi"` al arrancar. En concreto, se emitían los siguientes comandos:

- Comando `"/tapi /test_tapi"`: se emitía al inicio para comprobar si ACIDE estaba configurado para utilizar la consola DES y, en caso afirmativo, cargar el panel de base de datos de la interfaz.

Se estudió la situación y se vio que no era necesario emitir el comando `"/tapi /test_tapi"` para realizar dicha comprobación, ya que también se podía revisar obteniendo la configuración de consola de ACIDE.

- Comando `"/pdg"`: se emitía al inicio innecesariamente, ya que al arrancar no hay que construir ningún grafo en el panel PDG debido a que aún no se ha procesado ningún nuevo programa a depurar o analizar.

Una vez se eliminó toda emisión de comandos a la consola DES al arrancar ACIDE y se sustituyó la función del comando `"/tapi /test_tapi"` por una equivalente que no requería la emisión de comandos a consola, se pudo comprobar que la consola iniciaba correctamente y no había que reiniciarla.

Se determinó que, para evitar que en un futuro volviera a surgir el mismo problema, en futuras implementaciones, no se debía seguir añadiendo funcionalidad al proyecto que provocara que al arrancar ACIDE se emitieran comandos a la consola DES.

## 4.15. Corrección en el procesamiento de ficheros

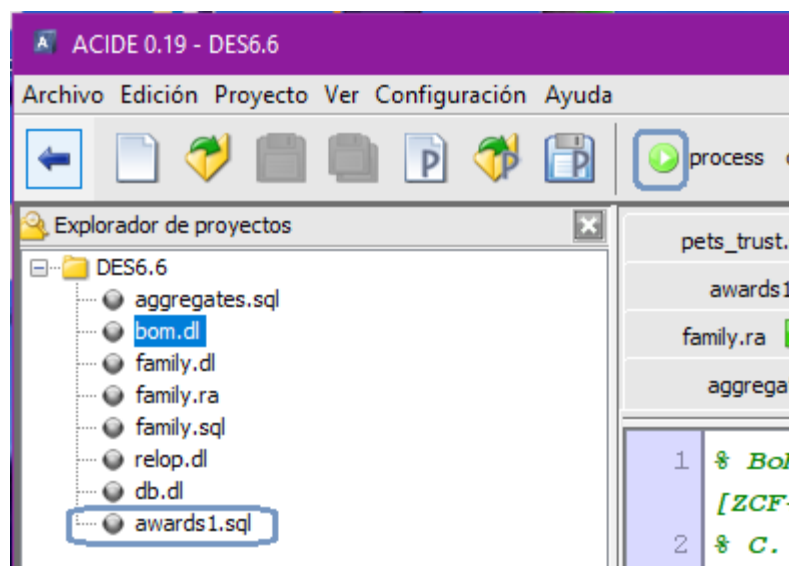
### Necesidad que motiva la realización de la tarea

En ciertas ocasiones, cuando se pulsa el botón con icono Play de la barra de comandos para procesar un fichero que contiene, por ejemplo, un script SQL que se quiere cargar en la base de datos, durante el procesamiento del mismo, se produce una excepción.

Dicha excepción notifica al usuario que la inserción que se ha intentado realizar en ese punto no es válida y continúa el procesamiento del contenido del fichero hasta



su terminación.



*Figura M-27 Procesado con botón con icono Play*

En esta situación, cuando el fichero termine de procesarse, se habrá cargado en la base de datos un script SQL incompleto, ya que no contendrá todas las sentencias SQL que lo componen.

Si la excepción se produce cuando, por ejemplo, se está intentando insertar una sentencia SQL de creación de una vista, esta situación sería un fallo crítico, pues el script final cargado no contendría dicha vista y no podría consultarse o depurarse si se quisiera.

#### Solución implementada

Tras investigar el motivo del fallo, la excepción se producía cuando la posición desde la cual se debía insertar el primer carácter de la siguiente cadena del script SQL que se debía añadir a la base de datos no era una posición correcta.

Para solucionar este problema, se ha implementado un mecanismo que resetea periódicamente el buffer de la consola a la cual se están enviando las sentencias SQL durante el procesamiento.

De este modo se consigue que, durante el procesamiento del fichero, se vuelva a insertar desde el inicio del buffer evitando así el riesgo de llegar a posiciones no válidas para la inserción.

## 4.16. Desplazamiento por un lienzo con dimensiones reducidas

### Necesidad que motiva la realización de la tarea

Cuando las dimensiones de los lienzos de los paneles de traza y depuración se reducen, no existe un mecanismo que facilite tanto el desplazamiento horizontal como vertical por su contenido haciendo uso del ratón. Esta situación dificulta la visualización completa del grafo cargado en los diferentes lienzos.

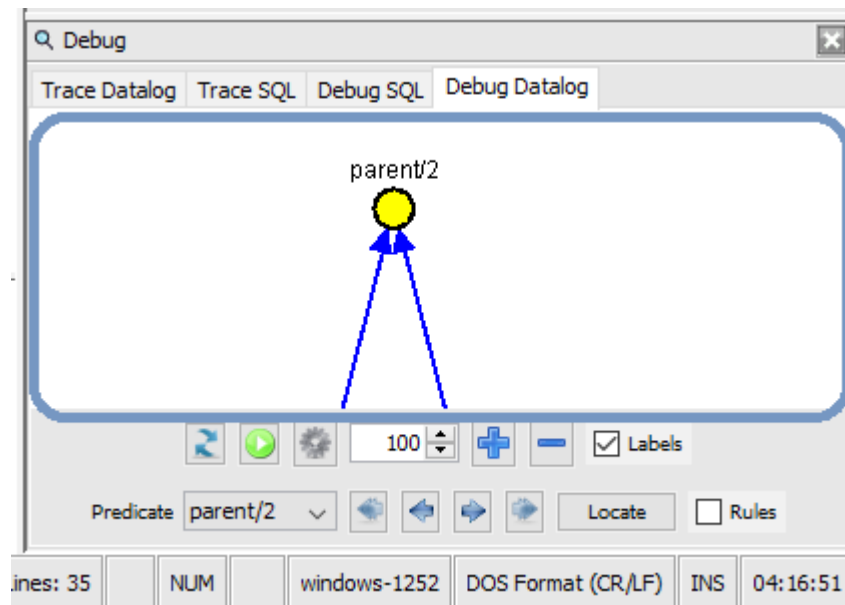
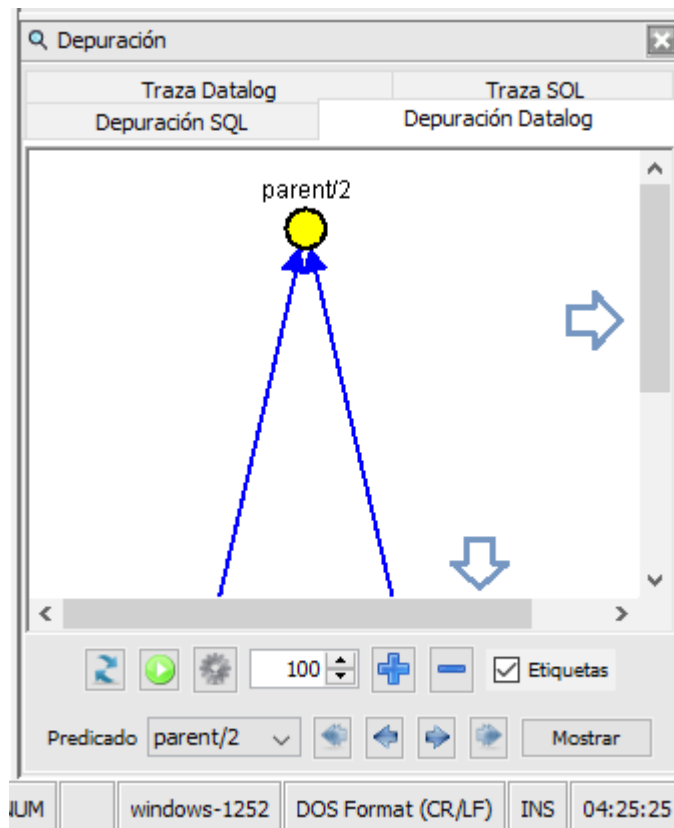


Figura M-28 Dimensiones reducidas del lienzo

### Solución implementada

Se ha implementado una funcionalidad que permite realizar un desplazamiento con el ratón tanto horizontal como vertical dentro del área de cada lienzo (tanto en los lienzos de traza como depuración) cuando las dimensiones de cada uno de ellos se reducen.



*Figura M-29 Scroll en lienzos*

## 4.17. Ventana de depuración SQL dinámica

### Necesidad que motiva la realización de la tarea

Si la vista o tabla que se está depurando contiene más tuplas de las que se pueden mostrar a lo largo de la pantalla, no se podrán visualizar todas de forma simultánea.

### Solución implementada

Se ha modificado la implementación para que los elementos que contiene la ventana de depuración SQL como la tabla que contiene las tuplas de la vista o tabla que se está depurando, etc. se ajusten a las dimensiones de la ventana cada vez que esta es redimensionada.

Por tanto, si, por ejemplo, el tamaño de la ventana de depuración SQL se reduce y la vista o tabla que se está depurando contiene más tuplas de las que se pueden mostrar en las nuevas dimensiones, se producirá el siguiente ajuste automático:

La tabla se ajustará ofreciendo una barra de desplazamiento vertical para que se pueda navegar por ella y ver la totalidad de las tuplas que contiene.

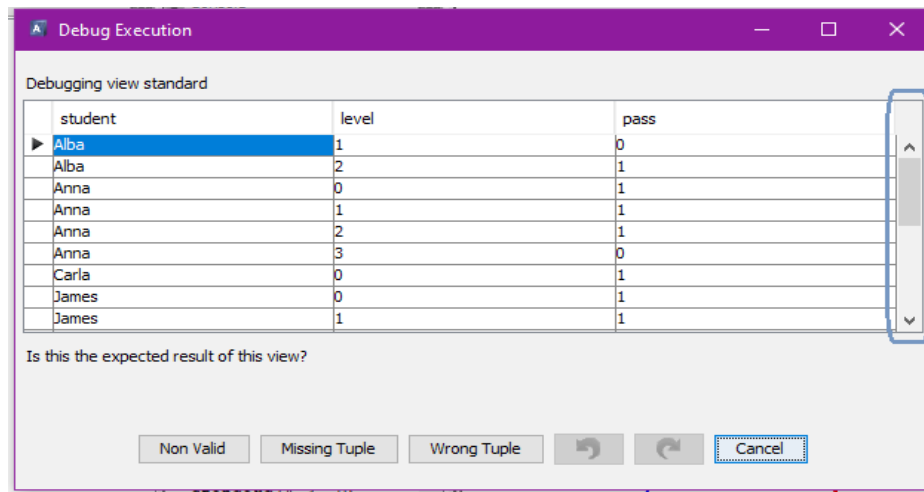


Figura M-30 Scroll en tabla

## 4.18. Conteo de tuplas

### Necesidad que motiva la realización de la tarea

Surgió la necesidad de contabilizar la cantidad de tuplas que contiene la vista o tabla que se está depurando y mostrar el resultado en la ventana de depuración SQL por los siguientes motivos:

- Una vista o tabla puede contener un gran número de tuplas y, por tanto, ser imposible mostrar todas ellas en la pantalla.
- Para facilitar la depuración e identificar rápidamente si la tabla o vista que se está depurando es válida al ver cuántas tuplas se han calculado para el mismo, que pueden o no coincidir con el número esperado.

### Solución implementada

Se ha añadido bajo la tabla de tuplas de la ventana de depuración SQL la cantidad de tuplas que tiene la vista o tabla que se está depurando.

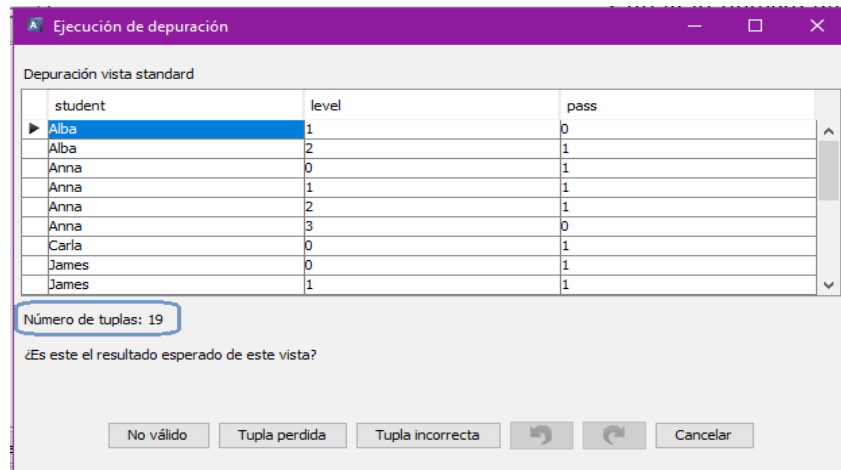


Figura M-31 Conteo de tuplas

## 4.19. Atajos de teclado para la depuración

### Necesidad que motiva la realización de la tarea

Para agilizar y facilitar el proceso de depuración al usuario, se planteó asociar las siguientes acciones a diferentes atajos de teclado:

- La acción de "deshacer" disponible en el proceso de depuración desde el menú contextual de los diferentes nodos del grafo a depurar a la combinación de teclas:

Ctrl + Z + Z

El usuario deberá mantener pulsada la tecla Ctrl mientras lleva a cabo una doble pulsación de la tecla Z.

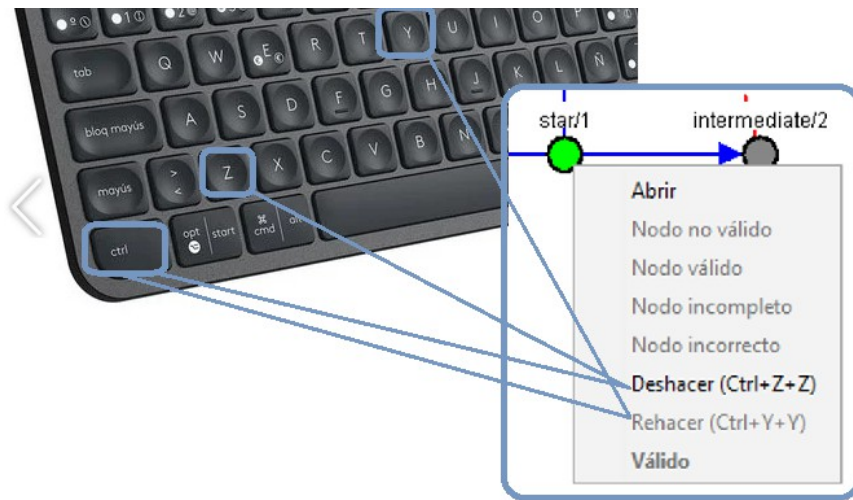
- La acción de "rehacer" disponible en el proceso de depuración desde el menú contextual de los diferentes nodos del grafo a depurar a la combinación de teclas:

Ctrl + Y + Y

El usuario deberá mantener pulsada la tecla Ctrl mientras lleva a cabo una doble pulsación de la tecla Y.

De este modo, si el usuario necesita deshacer o rehacer un paso realizado en la depuración, con una simple combinación de teclas podrá revertir o rehacer el paso

realizado sin la necesidad de invertir tiempo en dirigirse al panel de depuración, dar clic derecho sobre el nodo a depurar y elegir la acción correspondiente.



*Figura M-32 Atajos de teclado [8]*

### Solución implementada

Se han realizado tanto configuraciones en el código del manejador de eventos de teclado como en el código del panel de depuración Datalog y SQL.

Además, se ha actualizado el texto de las acciones deshacer y rehacer del menú contextual de los nodos.

## CONCLUSIONES

---

Una vez concluida mi contribución en ACIDE, puedo manifestar con confianza que se han cumplido los objetivos principales: la última versión de la aplicación (0.19) ya dispone de un panel para llevar a cabo la depuración de programas Datalog.

Además de cumplir con los objetivos principales, se han resuelto otras necesidades: se han solventado errores que dificultaban la interacción con la interfaz ACIDE y se han realizado mejoras en la misma.

Las contribuciones de años anteriores han sido de gran utilidad en el desarrollo, ya que me han servido como referente para poder cumplir con los estándares de código acordados así como para implementar respetando el diseño de años anteriores y, de esta forma, seguir conservando la uniformidad del proyecto.

Tras analizar el proceso llevado a cabo durante todos estos meses para desarrollar la interfaz gráfica de depuración Datalog en ACIDE he podido concluir que:

ACIDE es un entorno de desarrollo que facilita enormemente las implementaciones de código, ya que dispone de gran variedad de librerías que permiten crear todo tipo de componentes ajustados a las necesidades.

Sin embargo, he podido detectar que, durante las implementaciones llevadas a cabo, ha sido necesario mejorar o añadir nueva funcionalidad en el sistema DES: se tuvo que terminar de desarrollar en DES la función encargada de establecer el nuevo estado de un nodo del grafo a depurar mediante el comando `"/debug_dl_set_node"`.

Por tanto, el sistema DES se encuentra en continuo desarrollo y tiene la particularidad de ser muy robusto, ya que permite llevar a cabo procesos complejos como una sesión de depuración.

## CONCLUSIONS

---

Now that my contribution to ACIDE has been completed, I can confidently state that the main objectives have been met: the latest version of the application (0.19) now has a panel for debugging Datalog programs.

In addition to meeting the main objectives, other needs have been addressed: bugs that made it difficult to interact with the ACIDE interface have been fixed and improvements have been made to the interface.

The contributions from previous years have been very useful in the development, as they have served as a reference for me to comply with the agreed code standards and to implement respecting the design of previous years and, in this way, continue to maintain the uniformity of the project.

After analysing the process carried out during all these months to develop the Datalog debugging graphical interface in ACIDE, I have been able to conclude that:

ACIDE is a development environment that greatly facilitates code implementations, as it has a wide variety of libraries that allow the creation of all kinds of components adjusted to the needs.

However, I have been able to detect that, during the implementations carried out, it has been necessary to improve or add new functionality in the DES system: the function in charge of establishing the new state of a node of the network to be debugged by means of the command `"/debug_dl_set_node"` had to be completed in DES.

The DES system is therefore under continuous development and has the particularity of being very robust, as it allows complex processes such as a debugging session to be carried out.



## BIBLIOGRAFÍA

---

- [1] Manual de usuario DES versión 6.7:  
<https://www.fdi.ucm.es/profesor/fernan/des/html/manual/manualDES.html>
- [2] Google Drive: <https://drive.google.com/>
- [3] Página web oficial DES: <https://www.fdi.ucm.es/profesor/fernan/des>
- [4] Memoria TFG Carlos González Torres 2020-2021 - Gestión de Temas y Mejoras en ACIDE:  
[https://eprints.ucm.es/id/eprint/66830/1/GONZ%C3%81LEZ%20TORRES%202083082\\_CARLOS\\_GONZALEZ\\_TORRES\\_GESTION\\_DE\\_TEMAS\\_Y\\_MEJORAS\\_EN\\_A\\_CIDE\\_784051\\_1481962891.pdf](https://eprints.ucm.es/id/eprint/66830/1/GONZ%C3%81LEZ%20TORRES%202083082_CARLOS_GONZALEZ_TORRES_GESTION_DE_TEMAS_Y_MEJORAS_EN_A_CIDE_784051_1481962891.pdf)
- [5] Memoria TFG Sergio García Rodríguez 2019-2020 – Depuración de SQL:  
[https://eprints.ucm.es/id/eprint/68251/1/GARCIA\\_RODRIGUEZ\\_Depuracion\\_de\\_SQL\\_4398578\\_1134765792.pdf](https://eprints.ucm.es/id/eprint/68251/1/GARCIA_RODRIGUEZ_Depuracion_de_SQL_4398578_1134765792.pdf)
- [6] Conceptos básicos Datalog: <https://1library.co/document/z1l88o8q-datalog-bases-de-datos-deductivas.html>
- [7] Teoría Datalog y uso del sistema DES con el lenguaje Datalog:  
<http://bioinfo.uib.es/~joemiro/aenui/procJenui/Jen2007/sadesu.pdf>
- [8] Imagen de un teclado: <https://www.pccomponentes.com/logitech-mx-keys-teclado-inalambrico-avanzado-grafito>
- [9] F. Sáenz-Pérez, DESweb: una herramienta para el aprendizaje de SQL, XXV Edición de las Jornadas sobre la Enseñanza Universitaria de la Informática, Murcia, Spain, July, 2019.
- [10] Google Scholar: [https://scholar.google.es/scholar?hl=es&as\\_sdt=0%2C5&as\\_vis=1&q=deductive+database+systems&btnG=](https://scholar.google.es/scholar?hl=es&as_sdt=0%2C5&as_vis=1&q=deductive+database+systems&btnG=)
- [11] A survey of deductive database systems:  
<https://www.sciencedirect.com/science/article/pii/0743106694000399>
- [12] An introduction to deductive database systems:  
<https://www.acs.org.au/content/dam/acs/50-years/journals/acj/ACJ-V15-N02-198305.pdf#page=16>
- [13] SQUALID A Deductive DBMS: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.37.6724&rep=rep1&type=pdf>
- [14] XSB: Extending Prolog with Tabled Logic Programming:  
<https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.294.9440&rep=rep1&type=pdf>

## APÉNDICE – GENERACIÓN DEL EJECUTABLE Y ARRANQUE DE ACIDE

---

A continuación se pasarán a detallar los pasos que se deben seguir para llevar a cabo las siguientes tareas:

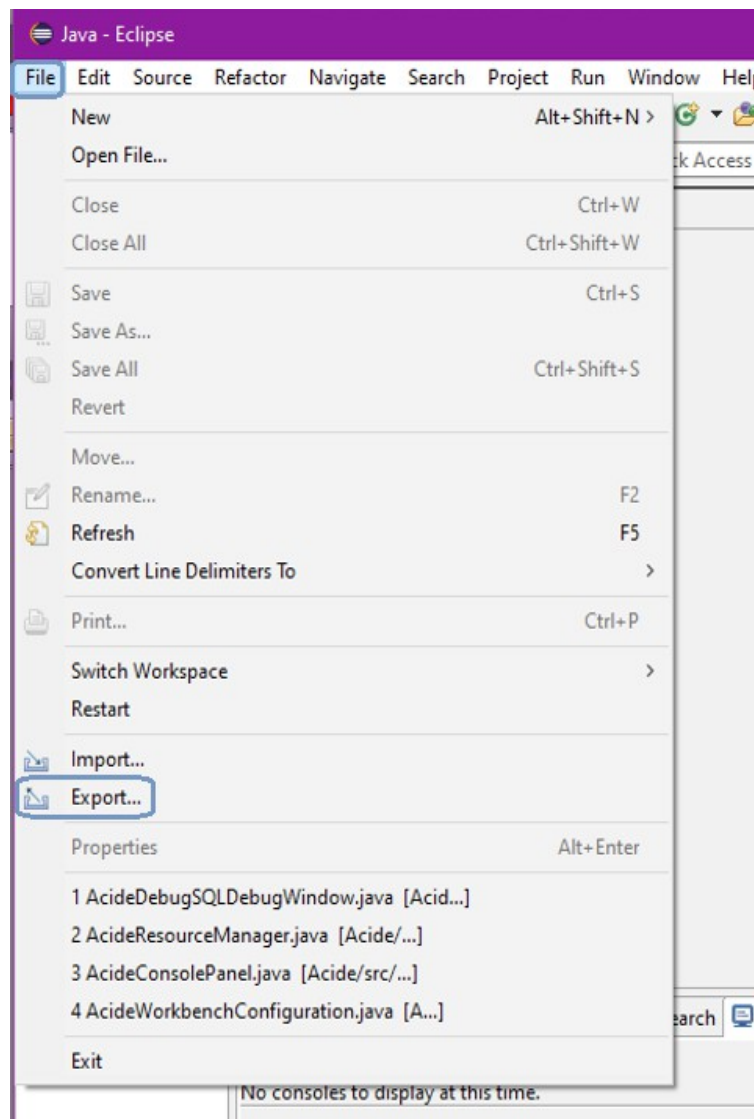
- Generar el archivo jar que permite arrancar ACIDE.
- Arrancar ACIDE a través del intérprete de comandos "cmd.exe" mediante la ejecución del archivo jar generado.

En esta sección no nos ocuparemos de detallar el proceso de configuración del entorno debido a que se dispone de una explicación detallada de este asunto en el anexo de la memoria del curso académico 2019-2020 [5].

Sin embargo, sí que describiremos el proceso a realizar para generar el archivo jar y ejecutarlo desde el intérprete de comandos cmd de Windows debido a que algunos compañeros tuvieron problemas para realizar estas acciones.

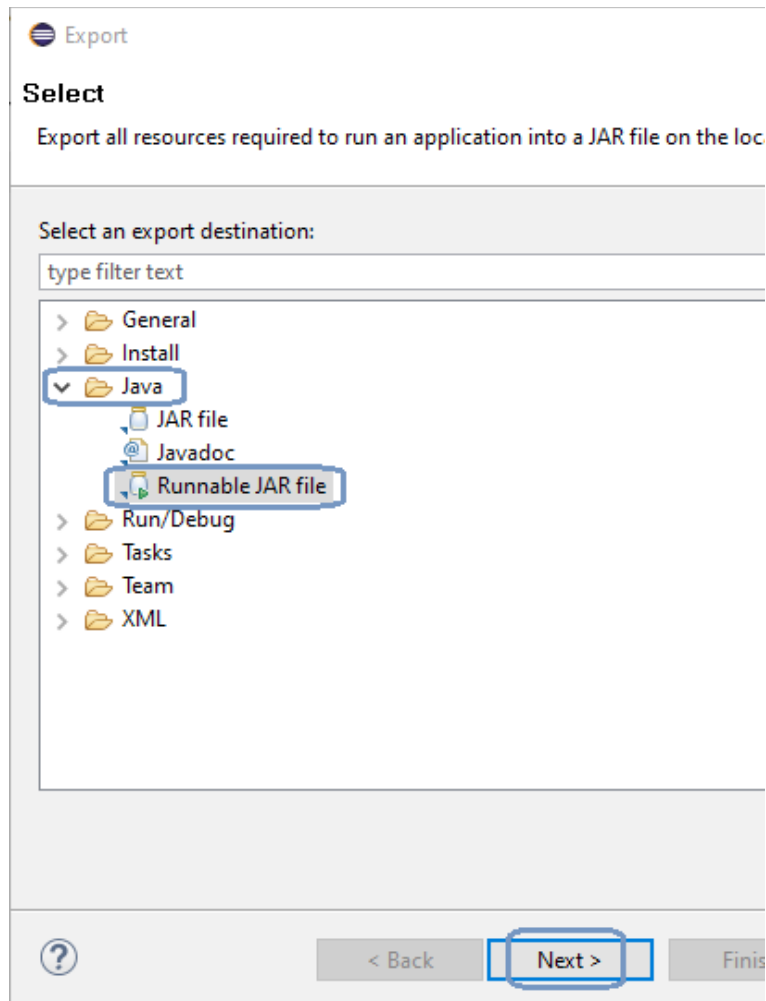
Para generar el archivo jar, se deben realizar los siguientes pasos:

- 1) Abrir Eclipse e importar el proyecto ACIDE.
- 2) Seleccionar el apartado "Export" de la pestaña "File" de Eclipse:



*Figura A-1 Opción Export Eclipse*

En la ventana emergente que aparece, desplegar la carpeta llamada "Java", seleccionar la opción "Runnable JAR file" y el botón "Next >".

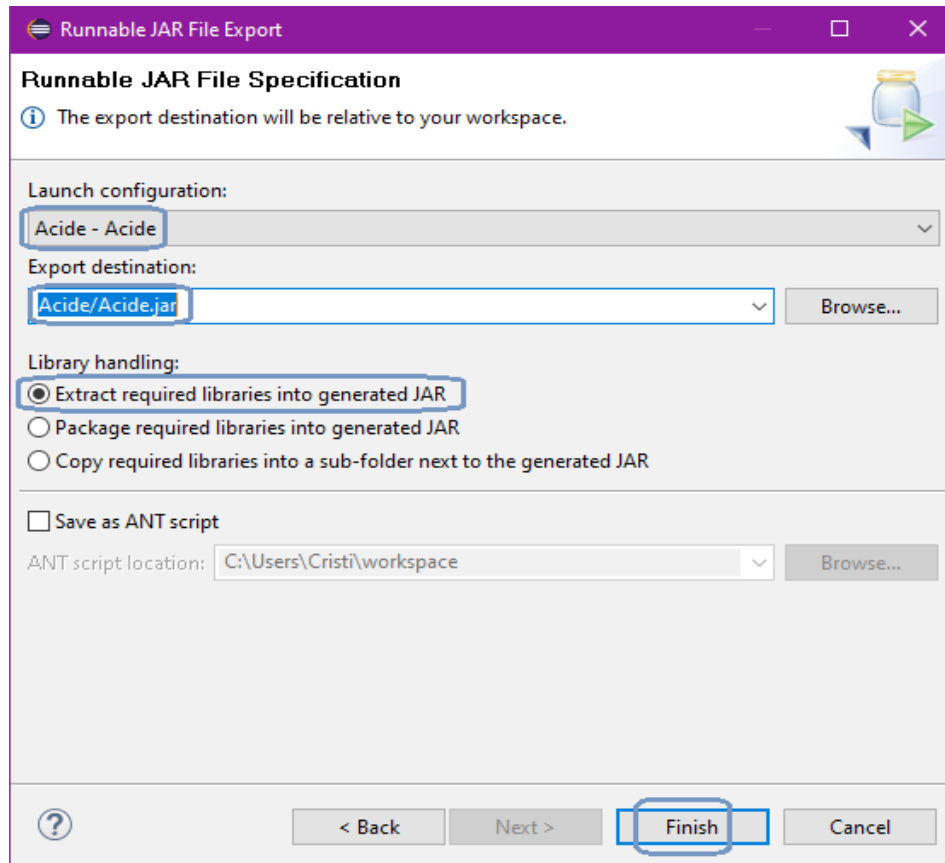


*Figura A-2 Opción jar ejecutable*

En los campos de la siguiente ventana emergente se deben seleccionar las siguientes opciones:

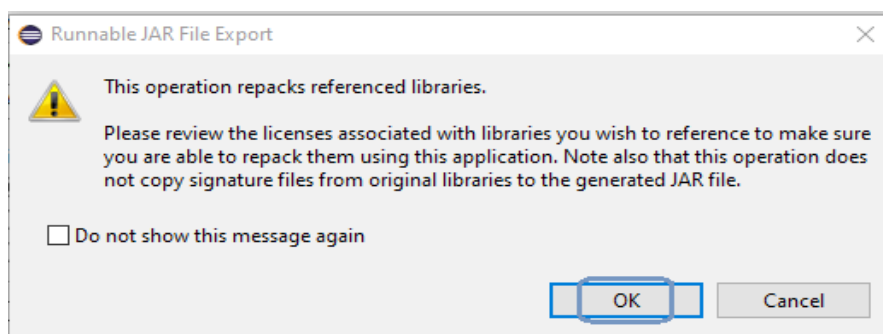
- En el campo "Launch configuration" seleccionar el lanzador de ACIDE configurado en Eclipse.
- En el campo "Export destination", introducir la ruta al proyecto ACIDE.
- En el campo "Library handling" seleccionar la opción "Extract required libraries into generated JAR".

Por último, seleccionar el botón "Finish".



*Figura A-3 Exportación jar*

Seleccionar el botón "OK" en el cuadro de diálogo que aparece.

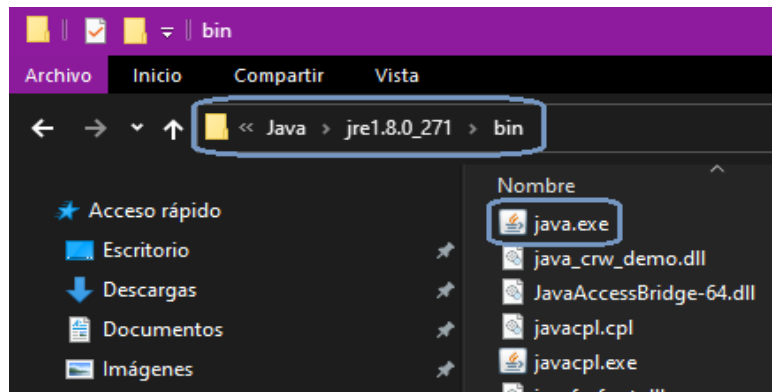


*Figura A-4 Aceptar diálogo*

Si la versión Java predeterminada del sistema es la versión 1.8, una vez generado el archivo jar, para arrancar ACIDE a partir del jar generado, se deberá pulsar dos veces sobre el mismo desde el explorador de archivos. En caso contrario, se deben realizar los siguientes pasos:

1) Asumiendo que el path de java no está configurado en el sistema, habrá que localizar y copiar la ruta de nuestro sistema al ejecutable **java.exe**. En mi caso, se ubica en la siguiente ruta:

C:\Program Files\Java\jre1.8.0\_271\bin

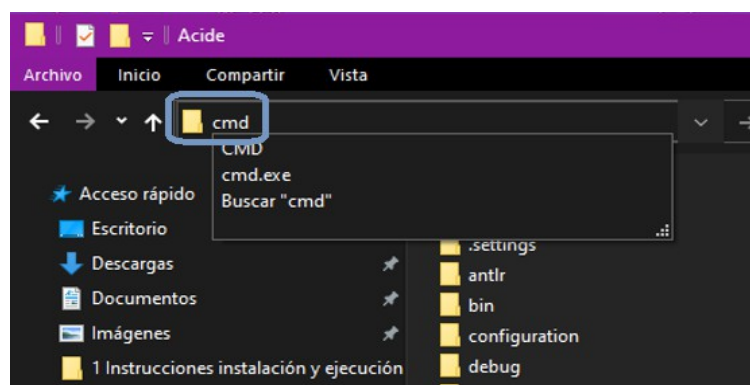


*Figura A-5 Ubicación archivo java.exe*

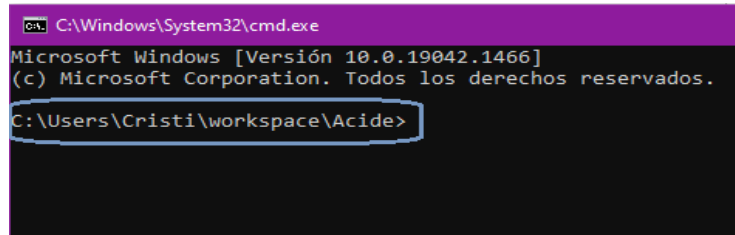
Agregar al final de la ruta anterior el nombre del archivo ejecutable java -> java.exe

C:\Program Files\Java\jre1.8.0\_271\bin\java.exe

2) Después, una vez copiada la ruta completa al archivo ejecutable java, dirigirse a la raíz del proyecto ACIDE y en la barra de rutas del explorador de archivos introducir la palabra "cmd" para que se abra el intérprete de comandos **cmd** y el prompt se ubique automáticamente dentro de la raíz del proyecto ACIDE.



*Figura A-6 Intérprete de comandos cmd.exe*



*Figura A-7 Prompt ubicado en raíz de proyecto ACIDE*

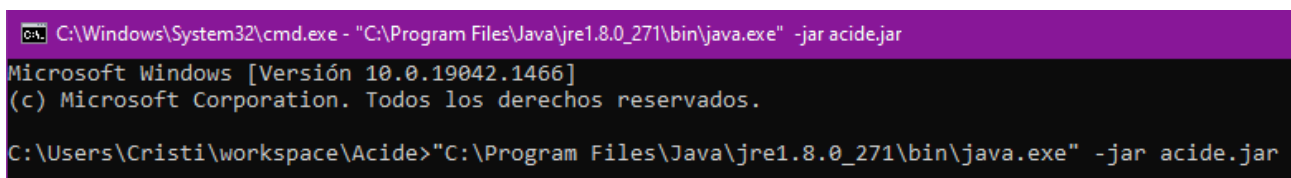
3) Por último, en el intérprete de comandos cmd, se deben introducir comillas dobles en el prompt y, entre ellas pegar la ruta a java.exe copiada anteriormente.

Una vez pegada la ruta a java.exe se debe pasar el argumento **-jar** y, tras el argumento jar introducir el nombre del archivo jar.

Tras realizar las acciones anteriores, el comando construido a ejecutar en cmd será el siguiente:

`"C:\Program Files\Java\jre1.8.0_271\bin\java.exe" -jar acide.jar`

Por último, se debe pulsar intro para ejecutar el comando anterior y arrancar ACIDE.



*Figura A-8 Comando construido*



*Figura A-9 Arranque ACIDE*