TRABAJO DE FIN DE GRADO 2019/2020

FACULTAD DE INFORMÁTICA

UNIVERSIDAD COMPLUTENSE DE MADRID

Depuración de SQL/ Debugging SQL

Realizado por:

Sergio García Rodríguez

Trabajo de fin de grado del Grado en Ingeniería del Software

Dirigido por:

Prof. Fernando Sáenz Pérez

Dpto. Ingeniería del Software e Inteligencia Artificial

RESUMEN

SQL es un lenguaje de programación declarativo que esconde muchos de sus mecanismos de resolución y por tanto no tiene un flujo de depuración claro como puedan tener otros lenguajes como C cuando se encuentra un error. Actualmente existen muy pocos métodos para hacer una depuración de una base de datos y la depuración declarativa es una de las soluciones más efectivas para esto.

DES es un programa que permite la depuración de bases de datos a través de una interfaz textual pero resulta tedioso de usar en comparación a una interfaz gráfica. En este trabajo crearemos una interfaz gráfica para la depuración de bases de datos en ACIDE, que es un IDE especializado en bases de datos que usa DES como soporte para realizar sus operaciones. En esta versión se ha añadido una ampliación de la depuración que hay en la última versión disponible de DES que permite asignar el estado de tablas y vistas que no están seleccionadas, mejorando la usabilidad del sistema. Además en este trabajo se han realizado tareas de mantenimiento y corrección de errores en ACIDE.

Palabras clave: DES, ACIDE, Depuración declarativa, IDE, SQL, Base de datos, TAPI, Consola.

ABSTRACT

SQL is a declarative programming language that hides many of its resolution mechanisms and therefore does not have a clear debugging flow as other languages such as C may have when an error is encountered. Currently there are very few methods of debugging a database and declarative debugging is one of the most effective solutions for this.

DES is a program that allows debugging databases through a textual interface but it is tedious to use compared to a graphical interface. In this work we will create a graphical interface for debugging databases in ACIDE, which is an IDE specialized in databases that uses DES as support to carry out its operations. In this version, an extension of the debugging that is in the latest available version of DES has been added that allows assigning the status of tables and views that are not selected, improving the usability of the system. In addition, this work has carried out maintenance and error correction tasks in ACIDE.

Keywords: DES, ACIDE, Declarative Debugging, IDE, SQL, Database, TAPI, Console.

ÍNDICE DE CONTENIDOS

1. INTRODUCCIÓN

La presente memoria describe el desarrollo de la interfaz de usuario para la depuración de bases de datos SQL como trabajo de fin de grado.

En ella, se presentan los fundamentos teóricos de la depuración declarativa de bases de datos SQL, el funcionamiento de esta en el sistema **DES**, el diseño de la interfaz y el funcionamiento de la misma en el sistema **ACIDE**.

Al final de la memoria se encuentra una guía de instalación destinada a los alumnos y otros interesados que quieran desarrollar nuevas funcionalidades en ACIDE.

1.1. MOTIVACIÓN

Actualmente no existen sistemas que permitan la depuración de bases de datos SQL, aunque sí que existen métodos de depuración para ejecuciones de PL/SQL. Los lenguajes como C y Java tienen un flujo de ejecución y depuración claro pero no ocurre lo mismo en lenguajes declarativos como SQL en los que se ocultan los mecanismos de resolución. Para determinar la fuente de un error se necesita aplicar la depuración declarativa, que es una técnica de depuración que permite encontrar el origen del error en un sistema basándose en las interpretaciones pretendidas del mismo.

El sistema DES [6] permite hacer una depuración textual de bases de datos SQL pero la utilidad es limitada en comparación con sistemas de depuración gráficos. ACIDE es un IDE que ayuda a nuevos programadores a crear bases de datos y se comunica con DES, por tanto el objetivo de este trabajo es la realización de una interfaz gráfica de depuración en ACIDE que aproveche las funcionalidades otorgadas por la aplicación DES.

1.2. OBJETIVOS

Este trabajo de fin de grado tiene como objetivo desarrollar una interfaz gráfica para la depuración de bases de datos SQL dentro de la aplicación ACIDE y la realización de tareas de mantenimiento y corrección de errores de la aplicación. Esta interfaz ofrece la capacidad de identificar de forma sencilla, en caso de que este exista, el origen del error en una base de datos a través de la depuración declarativa de Datalog y SQL usando la herramienta DES en un contexto en el que intervienen múltiples vistas. La interfaz gráfica proporciona una solución sencilla e intuitiva a los usuarios, especialmente a estudiantes y nuevos programadores en SQL, para averiguar el origen del problema de su base de datos en SQL.

La depuración se realiza a través de preguntas en base a los datos obtenidos de la vista o tabla, siendo las posibles respuestas: válido, no válido, tupla perdida y tupla errónea.

El funcionamiento como explicaremos en apartados posteriores de forma más extensa es el siguiente: en primer lugar, el usuario seleccionará la vista o tabla que ha detectado como incorrecta y el sistema seleccionará la siguiente vista o tabla relacionada con la anterior con más posibilidades de ser el nodo erróneo en base a la respuesta proporcionada por el usuario. El usuario también puede anotar el estado de un nodo en cualquier momento de la depuración haciendo que sea mucho más rápida. Una vez encontrado el nodo de la base de datos detectado como erróneo el usuario se dispondrá a editar la tabla o vista haciendo que la base de datos quede en el estado que el usuario quería desde un principio.

1.3. Precedentes del proyecto

ACIDE es un entorno de desarrollo integrado que puede ser utilizado fácilmente como intérprete, compilador o sistema de bases de datos para Álgebra Relacional, TRC, DRC, Datalog y SQL.

Este proyecto es una continuación en el desarrollo de la herramienta, que ha sido desarrollada por grupos anteriores para la asignatura Sistemas Informáticos:

- El primer año en el que se llevó a cabo fue en el curso académico 2006-2007 por Diego Cardiel Freire, Juan José Ortiz Sánchez y Delfín Rupérez Cañas.
- Durante el curso académico 2007-2008 el desarrollo fue continuado por Miguel Martín Lázaro.
- La siguiente iteración en el desarrollo de ACIDE fue en el curso 2010-2011 por Javier Salcedo Gómez.
- Durante el curso académico 2012-2013 el desarrollo de ACIDE siguió en manos de Pablo Gutiérrez García-Pardo, Elena Tejeiro Pérez de Ágreda y Andrés Vicente del Cura.
- En el curso académico 2013-2014 el desarrollo se realizó por Juan Jesús Marqués Ortiz, Fernando Ordás Lorente y Semíramis Gutiérrez Quintana.
- Y durante el curso 2014-2015 el desarrollo se realizó por Sergio Domínguez Fuentes quien dejó el proyecto en el estado en que me lo encontré.

Este proyecto siempre fue dirigido por Fernando Sáenz Pérez.

Estas memorias se encuentran en la sección **[1]**, **[2]**, **[3]**, **[4]** y **[5]** del capítulo Referencias.

La versión con la que empecé a desarrollar el proyecto fue la **v0.17** tenía un código fuente estandarizado y un comportamiento estable en cuanto a la gestión de proyectos, creación y edición de ficheros de texto en diferentes lenguajes de programación, conexión tanto con ODBC como con DES (Datalog Educational System) **[6]**. La versión generada a partir de este trabajo es la **v.18**.

1.4. Plan de trabajo

Al comienzo del proyecto, el director me proporcionó una especificación de los requisitos que debía cumplir el sistema y a partir de ella he realizado implementaciones de los mismos. Cuando estas implementaciones han sido realizadas se generaba una nueva versión del sistema y la enviaba al director que validaba dichas funcionalidades. De esta forma se ha refinado el sistema hasta que se ha generado una versión estable que posee todas las funcionalidades requeridas al principio del proyecto.

1.5. Plan de estructura

El resto de la memoria está estructurada de la siguiente manera:

- En el apartado 3 se define la técnica de depuración declarativa que es el fundamento sobre el uso de este proyecto.
- En el apartado 4 se expone la comunicación entre DES que posee la implementación de la depuración declarativa y ACIDE que expone la funcionalidad de DES de forma gráfica.
- En el apartado 5 se explica el diseño de la interfaz implementada en ACIDE.
- En el apartado 6 se exponen ejemplos de la depuración declarativa.
- En el apartado 7 se exponen las tareas de mantenimiento realizadas.
- En el apartado 8 se exponen las herramientas y tecnologías usadas para la realización del proyecto.
- En el apartado 9 se expone la documentación generada a lo largo del proyecto.
- En el apartado 10 se exponen las conclusiones del proyecto.
- En el apartado 11 se expone el trabajo futuro del proyecto.
- En el apartado 12 se exponen las conclusiones del proyecto en inglés.
- En el apartado 13 se expone el trabajo futuro del proyecto en inglés.
- En el apartado 14 se expone la bibliografía usada para la realización del proyecto.
- En el apartado 15 se exponen las referencias usadas en esta memoria.
- Al final de la memoria se encuentra un anexo con los pasos a realizar para poder desarrollar nuevas funcionalidades en ACIDE.

2. INTRODUCTION

This report describes the development of the user interface for debugging SQL databases as a final degree project.

In it, the theoretical foundations of the declarative debugging of SQL databases, its operation in the DES system, the interface design and its operation in the ACIDE system are presented.

At the end of the report there is an installation guide for students and other interested parties who want to develop new functionalities in ACIDE.

2.1. MOTIVATION

Currently there are no systems that allow debugging of SQL databases, although there are debugging methods for PL / SQL executions. Languages such as C and Java have a clear execution and debugging flow, but the same does not happen in declarative languages such as SQL in which the resolution mechanisms are hidden. To determine the source of an error it is necessary to apply declarative debugging, which is a debugging technique that allows finding the source of the error in a system based on the intended interpretations of it.

The DES [6] system allows textual debugging of SQL databases, but the utility is limited compared to graphical debugging systems. ACIDE is an IDE that helps new programmers to create databases and communicates with DES, therefore the objective of this work is to create a graphical debugging interface in ACIDE that takes advantage of the functionalities provided by the DES application.

2.2. OBJECTIVES

This final degree project aims to develop a graphical interface for debugging SQL databases within the ACIDE application and carrying out maintenance and error correction tasks for the application. This interface offers the ability to easily identify, if it exists, the source of the error in a database through declarative debugging of Datalog and SQL using the DES tool in a context in which they intervene multiple views. The graphical interface provides a simple and intuitive solution for users, especially students and new SQL programmers, to find out the source of their database problem in SQL.

The debugging is done through questions based on the data obtained from the view or table, the possible answers being: valid, invalid, missing tuple and wrong tuple.

The operation, as we will explain more extensively in later sections, is as follows: first, the user will select the view or table that has been detected as incorrect and the

system will select the next view or table related to the previous one with more possibilities of being the one. bad node based on the response provided by the user. The user can also annotate the status of a node at any time during debugging, making it much faster. Once the database node detected as erroneous has been found, the user will proceed to edit the table or view, making the database remain in the state that the user wanted from the beginning.

2.3. Background of the project

ACIDE is an integrated development environment that can be easily used as an interpreter, compiler or database system for Relational Algebra, TRC, DRC, Datalog and SQL.

This project is a continuation in the development of the tool, which has been developed by previous groups for "Computing Systems" subject:

- The first time ACIDE was developed was during the academic year 2006-2007 by Diego Cardiel Freire, Juan José Ortiz Sánchez and Delfín Rupérez Cañas.
- After that, during the academic year 2007-2008 the development Miguel Martín Lázaro continued with ACIDE.
- The next phase was made by Javier Salcedo Gómez during the 2010-2011 academic year.
- During the academic year 2012-2013, the development was implemented by Pablo Gutiérrez García-Pardo, Elena Tejeiro Pérez de Ágreda and Andrés Vicente del Cura
- In 2013-2014 the project was developed by Juan Jesús Marqués Ortiz, Fernando Ordás Lorente and Semíramis Gutiérrez Quintana.
- The last iteration was implemented by Sergio Domínguez Fuentes during the 2014-2015 academic year.

This project was always managed by Fernando Sáenz Pérez.

Those reports can be found in section **[1]**, **[2]**, **[3]**, **[4]** and **[5]** of the References chapter.

The version with which I started to develop the project was v0.17 it had a standardized source code and stable behavior regarding project management, creation and editing of text files in different programming languages, connection to both ODBC and with **DES** (Datalog Educational System) **[6]**. The version generated from this project is the **v.18**.

2.4. WORK PLAN

At the beginning of the project, the director provided me with a specification of the requirements that the system had to meet and based on this I have implemented them. When these implementations have been carried out, a new version of the system was generated and sent to the director who validated said functionalities. In this way, the system has been refined until a stable version has been generated that has all the functionalities required at the beginning of the project.

2.5. STRUCTURE PLAN

The rest of the report is structured as follows:

- Section 3 defines the declarative debugging technique that is the basis for the use of this project.
- In section 4 the communication between DES that has the implementation of declarative debugging and ACIDE, which exposes the functionality of DES graphically, is exposed.
- Section 5 explains the design of the interface implemented in ACIDE.
- Examples of declarative debugging are provided in Section 6.
- In section 7 the maintenance tasks performed are explained.
- In section 8 the tools and technologies used to carry out the project are exposed.
- In section 9 the documentation generated throughout the project is exposed.
- In section 10 the conclusions of the project are presented.
- In section 11 the future work of the project is exposed.
- In section 12 the conclusions of the project are presented in English.
- In section 13 the future work of the project is presented in English.
- In section 14 the bibliography used to carry out the project is exposed.
- In section 15 the references used in this specification are exposed.
- At the end of the report there is an annex with the steps to be carried out in order to develop new functionalities in ACIDE.

3. DEPURACIÓN DECLARATIVA EN SQL

Como he mencionado en la introducción, vamos a exponer qué es la depuración declarativa ya que es clave para entender el funcionamiento de este trabajo.

Este apartado es un pequeño resumen de 'Algorithmic Debugging of SQL Views' **[9]** que se encuentra en el apartado de referencias en caso de querer tener más detalles.

SQL es el lenguaje estándar más usado para consultar y actualizar bases de datos relacionales. Tiene un alto nivel de abstracción y una naturaleza declarativa que permite al usuario definir complejas operaciones.

En la mayoría de aplicaciones las consultas se vuelven demasiado complejas para codificarse en una sola declaración y generalmente se definen usando vistas. Las vistas pueden considerarse esencialmente como tablas virtuales. Están definidas por una instrucción SELECT, una instrucción que hace referencia a tablas de la base de datos, así como en otras vistas previamente definidas.

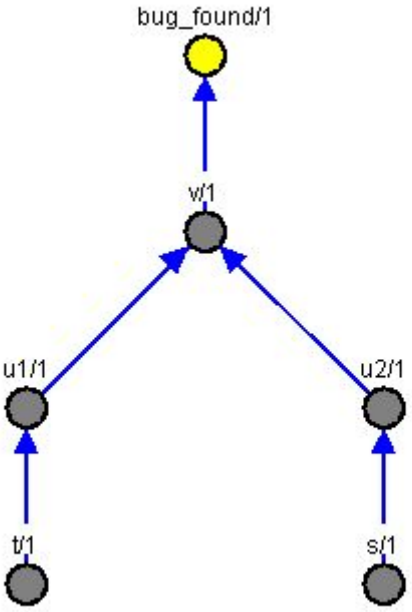
Cuando se encuentra una vista que tenga datos no esperados no podemos asegurar que dicha vista sea errónea ya que puede recibir datos incorrectos de otra vista o tabla, haciendo que el proceso de depuración sea difícil.

Para resolver este problema usaremos la depuración declarativa que tiene el siguiente proceso:

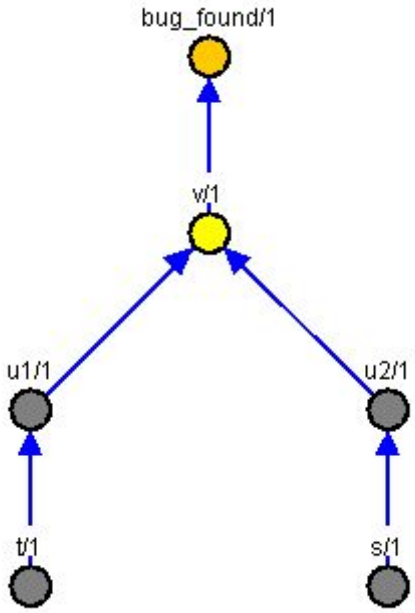
1. El proceso comienza cuando el usuario detecta datos incorrectos en una vista informando al sistema.
2. El sistema construye un grafo dirigido en el que cada dependencia de un nodo representa a una computación intermedia, que puede ser otra vista o una tabla de la que el padre obtiene la información.
3. El sistema navega a través de los nodos y el usuario proporciona información al sistema dependiendo de la información obtenida en comparación a la información esperada. En caso de que la información sea correcta el usuario informará que la vista o tabla es *válida*, en caso contrario el usuario la marcará como *no válida* pudiendo especificar más indicando que una *tupla* es *errónea* o *falta*.
4. La fase de depuración termina cuando se encuentra un nodo *no válid*o cuyas dependencias son todas *válidas*. Ese nodo se marcará como erróneo y puede ser tanto una tabla como una vista.

Esta técnica de depuración se basa en comparar la información obtenida de la vista con la información que el usuario espera obtener especificando al sistema el error que ha encontrado, de esta forma el sistema seleccionará la siguiente vista o tabla a depurar.

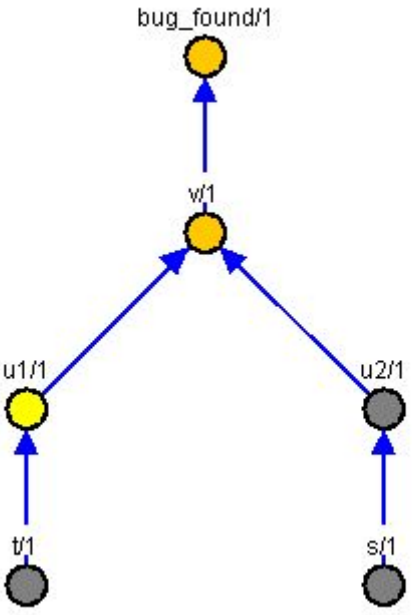
En el siguiente ejemplo vemos un grafo de dependencias de nuestra base de datos. Los nodos 't' y 'v' son tablas y el resto vistas dependientes de dichas tablas. Hemos encontrado un error en la vista 'bug_found':



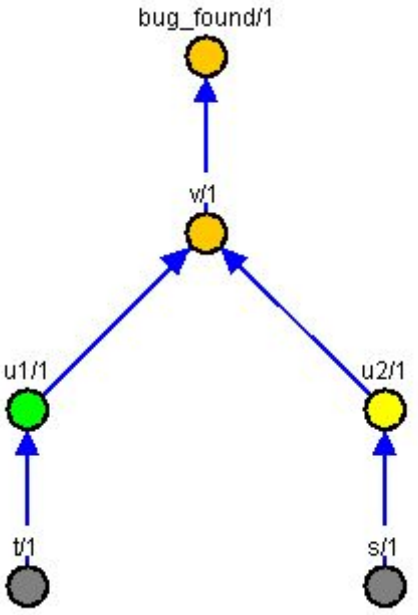
Comenzamos la depuración diciendo que el nodo 'bug_found' no es válido.



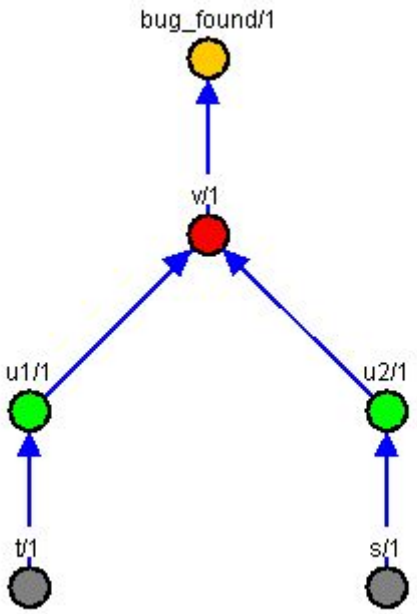
Entonces el sistema selecciona la siguiente vista que es 'v', observamos los datos y vemos que no son válidos.



A continuación observamos que la vista ‘u1’ es válida entonces seleccionamos el nodo como válido.



Vemos que el nodo ‘u2’ también es válido, por tanto el nodo (vista) que causa el problema en nuestra base de datos es el nodo ‘v’.



Ahora que sabemos cual es la técnica de depuración, voy a exponer a DES que es la aplicación textual que implementa la depuración declarativa y es usada por ACIDE para exponer sus funcionalidades.

Una de las mejoras que ofrece una interfaz gráfica es mostrar en todo momento cuáles son las dependencias entre vistas y tablas de forma clara en nuestra base de datos. Con una interfaz, un usuario puede establecer el estado de un nodo en cualquier momento mejorando la efectividad de la depuración declarativa, que es uno de los puntos clave de este proyecto.

Para conocer más detalles sobre los fundamentos teóricos de la depuración declarativa consultar la referencia **[9]**.

4. COMUNICACIÓN ENTRE ACIDE Y DES

Para entender cómo va a funcionar la interfaz de usuario y las posibilidades que tiene hay que saber cómo se aplica lo explicado en el apartado anterior.

DES es una aplicación basada en Prolog de un Sistema de bases de datos deductivas que permite el uso de Datalog, SQL, Álgebra Relacional (RA), Cálculo Relacional de Tuplas (TRC), y Cálculo Relacional de Dominios (DRC). Este sistema es el que tiene implementado el algoritmo de depuración y toda la lógica necesaria para su realización.

ACIDE es un IDE que expone las funcionalidades que posee DES de forma gráfica, y en este apartado explicaremos las funcionalidades que deben ser implementadas en ACIDE y otra información relevante.

La comunicación que se realiza entre DES y ACIDE es a través de comandos. Los comandos empiezan por el carácter '/', y se puede anteponer '/tapi' a los comandos para que sigan el protocolo TAPI (Textual Application Programming Interface) y devuelva la información estructurada. Las posibles respuestas a comandos son:

- Respuesta con éxito y sin devolver datos:

```
$success
```

- Respuesta con error:

```
$error
código
texto
...
texto
$eot
```

Donde *código* es el código del error, *texto* es el texto del error, que puede extenderse varias líneas y $eot es el fin de la transmisión de datos.

4.1. OPCIONES DE DEPURACIÓN

A continuación se van a listar las posibles operaciones y configuraciones que se van a usar en los comandos para la depuración de bases de datos.

- Estados de los nodos:

```
State :=
  valid     |
  nonvalid  |
  erroneous
```

- Configuración de la depuración:

- `trust_tables([yes|no])`: esta opción indica si se confía en las tablas de nuestra base de datos, es decir, en caso de confiar en las tablas sus nodos tendrán el estado 'valid'.
- `trust_file(FileName)`: esta opción indica que se confíen en las vistas definidas en el archivo indicado.
- `debug([full|plain])`: esta opción indica si se pueden o no usar las opciones de tupla faltante y errónea.
- `order([cardinality|topdown])`: esta opción indica el orden de selección de nodos.

- Tuplas:

```
Tuple :=
  Relation(Value,...,Value)
```

Los valores se encierran entre comillas simples para valores de tipo `string`.

- Preguntas al usuario:

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

- Respuestas del usuario:

```
Answer :=
    abort                    | % Para todas las preguntas
    valid                    | % Para todas las preguntas
    nonvalid                 | % Para todas las preguntas
    missing(PartialTuple)    | % Menos para in y subset
    wrong(TotalTuple)        | % Menos para in
```

4.2. Comandos

En este apartado vamos a exponer los comandos de depuración y sus funcionalidades.

- Comenzar la depuración:

```
/tapi /debug_sql ViewName Options
```

Donde *ViewName* es el nombre de la vista a depurar y *Options* es la lista de opciones de depuración comentada anteriormente separada por espacios.

- Marcar nodo con su estado:

```
/tapi /debug_sql_set_node NodeName State
```

- Pregunta al usuario:

```
/tapi /debug_sql_current_question
```

- Respuesta del usuario:

```
/tapi /debug_sql_answer Question Answer
```

- Estadísticas de la depuración:

```
/tapi /debug_sql_statistics
```

4.3. FUNCIONAMIENTO DE LA DEPURACIÓN

Cuando el usuario detecta que una vista no devuelve los resultados esperados (no es válida), puede iniciar la sesión de depuración emitiendo el siguiente comando:

```
/tapi /debug_sql ViewName Options
```

Como respuesta, DES devolverá una lista de nodos y sus estados si no hay error. En esta lista de nodos que devuelve DES es posible que se encuentre el nodo causante del problema. En este caso el nodo está identificado como erróneo: `erroneous` (por ejemplo cuando se selecciona una vista que solo depende de tablas y se confía en sus contenidos). En este caso habría terminado la depuración. Si devuelve un error (respuesta estándar de error como se describe en el punto 6) la depuración habría terminado.

Las opciones (*Options*) son las definidas en el apartado 6.1. Dependiendo de la respuesta del usuario, *Answer* tomará distintos valores:

- No válida (Non-valid): Al elegir esta opción se emite el comando:
  `/tapi /debug_sql ViewName nonvalid Options`
- Tupla errónea (Wrong Tuple): Para elegir esta opción se debe seleccionar la tupla errónea en la rejilla de datos de *ViewName* (la tupla es obviamente total). Al elegir esta opción se emite el comando:
  `/tapi /debug_sql ViewName wrong(Tuple) Options`
- Tupla faltante (Missing Tuple): Para elegir esta opción se deben escribir los valores de la tupla en una rejilla. Alguno de los valores puede estar vacío (es decir, la tupla puede ser parcial, como se describe en el apartado 1). Al elegir esta opción se emite el comando:
  `/tapi /debug_sql ViewName missing(Tuple) Options`

Después el sistema genera una pregunta y selecciona el siguiente nodo que debe ser depurado. El tipo de pregunta que se espera que responda el usuario se determina con el comando:

```
/debug_sql_current_question
```

que devuelve ese tipo de pregunta, que se detallan a continuación.

Para dicha pregunta se pueden tener las siguientes respuestas:

- Válida (Valid): Al elegir esta opción se responde con:
  `/tapi /debug_sql_answer all/subset/in(ViewName) valid`
- No válida (Non-valid): Al elegir esta opción se responde con:
  `/tapi /debug_sql_answer all/subset/in(ViewName) nonvalid`
- Tupla errónea (Wrong Tuple): Para elegir esta opción se deben escribir los valores de la tupla. Alguno de los valores puede estar vacío.
  `/tapi /debug_sql_answer all/subset(ViewName) wrong(Tuple)`
- Tupla faltante (Missing Tuple): Para elegir esta opción se deben escribir los valores de la tupla. Alguno de los valores puede estar vacío.
  `/tapi /debug_sql_answer all(ViewName) missing(Tuple)`
- Abortar (Abort): Aborta la depuración. Al elegir esta opción se responde con:
  `/tapi /debug_sql_answer all(ViewName) abort`

En cualquier momento el usuario puede marcar los nodos de los que no se conozca aún su estado con el siguiente comando:

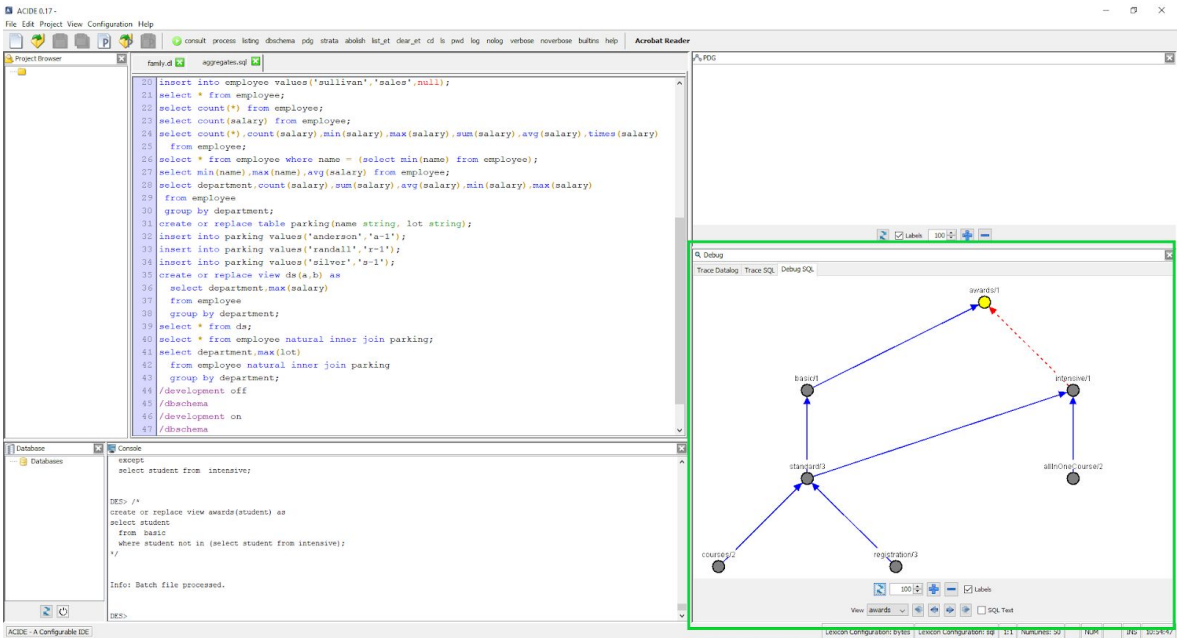
```
/tapi /debug_sql_set_node NodeName state
```

La depuración termina porque la cancela el usuario o bien porque se devuelve un nodo erróneo (`erroneous`, resultado del comando `/tapi /debug_sql_answer`). En este último caso se indica el nodo erróneo y se informa de las estadísticas, que se obtienen con el comando `tapi /debug_sql_statistics` explicado en el apartado 6.2.

En el siguiente apartado se muestra cómo se ha diseñado la interfaz gráfica en base a estos comandos.

5. DISEÑO DE LA INTERFAZ

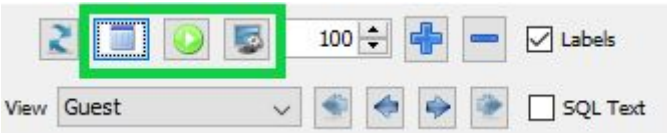
Como hemos comentado en la introducción, ACIDE es un IDE desarrollado por varios grupos de estudiantes en años anteriores. En el momento de comenzar este proyecto ya estaba creado el panel de depuración dentro del sistema. Éste tenía un comportamiento estable en cuanto a la obtención de las vistas existentes en la base de datos en uso y la generación de grafos que muestran la relación entre vistas y tablas, como puede visualizarse en la siguiente imagen.



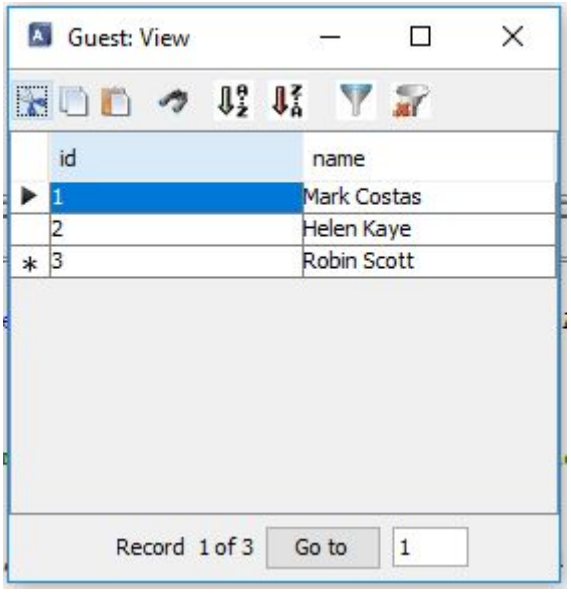
El principal objetivo de la interfaz tiene que ser la depuración de forma rápida e intuitiva para que cualquier usuario que no tenga experiencia previa con la aplicación **ACIDE** pueda encontrar los errores que pueda tener su base de datos.

Las características principales que tiene que tener la interfaz es de ser clara para el usuario y por tanto en cada paso de depuración el sistema debe exponer toda la información disponible. Para ello el panel 'Debug SQL' de **ACIDE** muestra el grafo de computación con los estados de los nodos actualizados y una ventana con la información del nodo que está siendo depurado actualmente.

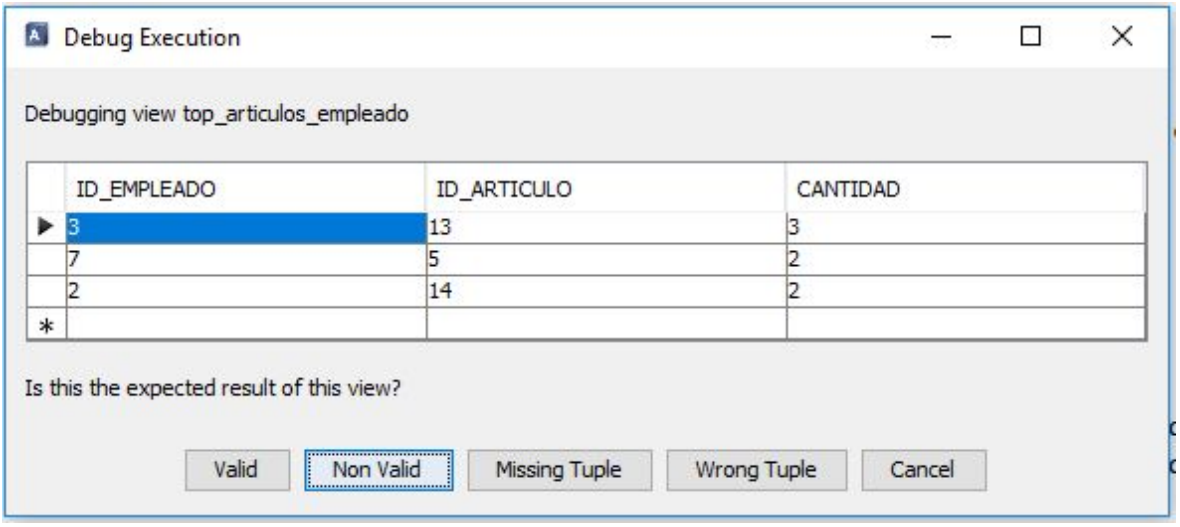
En este panel se han incluido 3 botones nuevos con el objetivo de mostrar la información de la vista seleccionada, para comenzar la depuración y para configurar la depuración como se muestra en la siguiente imagen.



El primer botón muestra la información obtenida de la vista seleccionada que es la vista que el usuario ha detectado como errónea. Esto permite que el usuario pueda comparar en cualquier momento esa vista con otras vistas o tablas relacionadas con ella, teniendo un mayor contexto sobre cuál puede ser el origen del problema. Pulsando este botón se obtiene una nueva ventana como la siguiente:

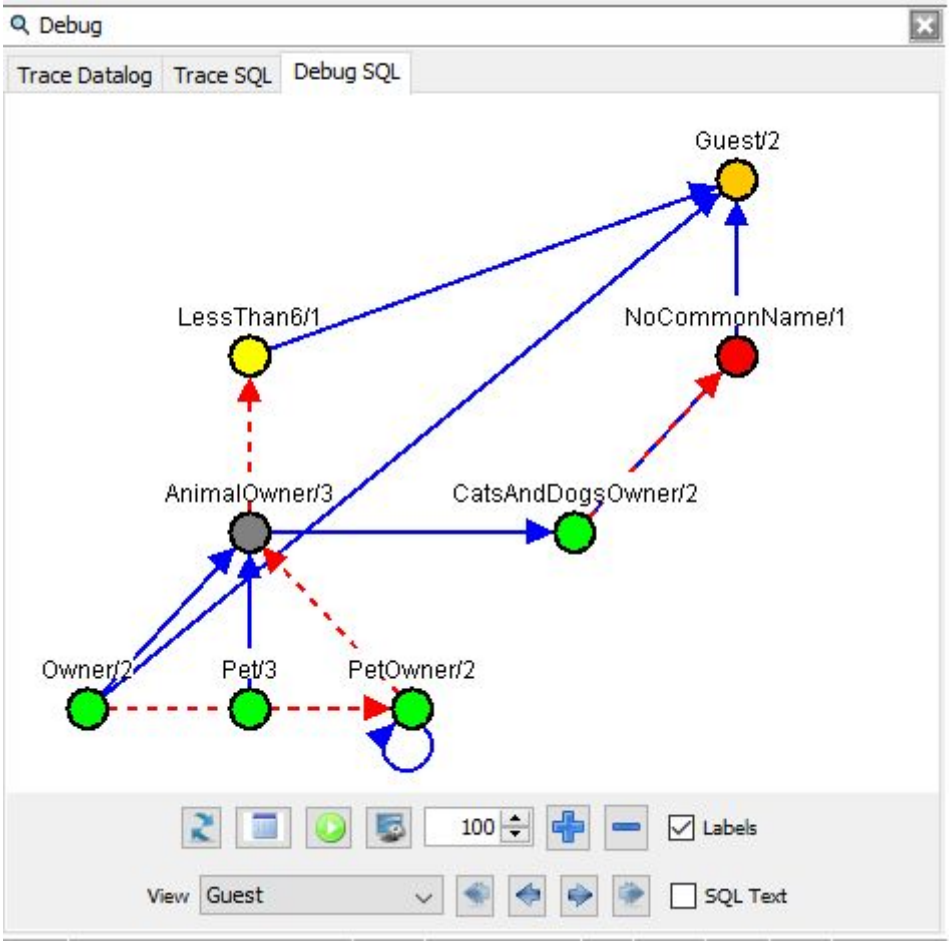


En caso de pulsar en el botón de comenzar la depuración se abrirá una nueva ventana con la información de la vista o tabla que estemos depurando en ese momento con las opciones disponibles. La tabla de datos siempre tendrá su última fila vacía y el usuario la rellenará en caso de querer informar al sistema que hay una tupla faltante en dicha vista o tabla. En caso de querer indicar que hay una tupla incorrecta, se informará al sistema enviando la tupla completa que esté seleccionada en ese momento.

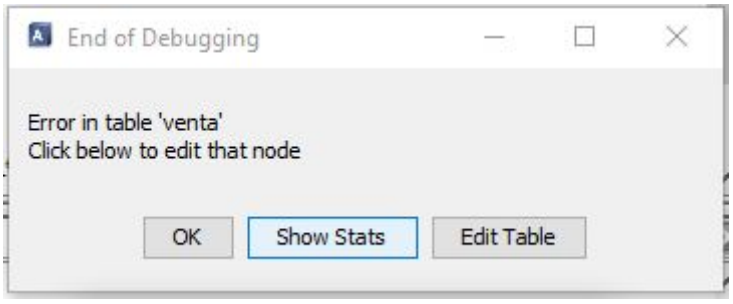


En cada paso de depuración se actualizan los colores de los nodos del grafo de depuración con la información que tiene actualmente el sistema, los colores de los nodos son los siguientes:

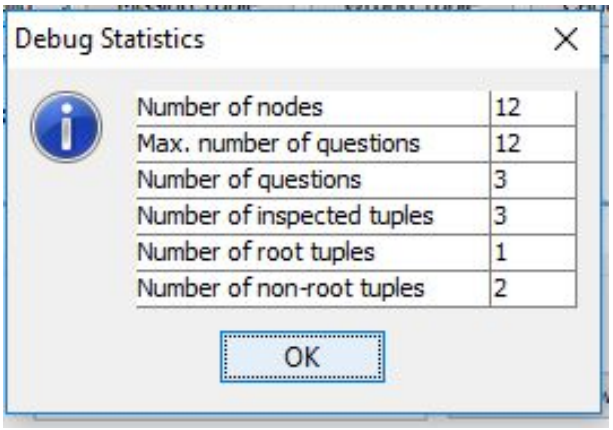
- Gris: en caso de que dicho nodo no haya sido explorado.
- Amarillo: indica que dicho nodo es el seleccionado actualmente.
- Verde: en caso de que el nodo haya sido evaluado como válido.
- Naranja: en caso de que el nodo haya sido evaluado como no válido.
- Rojo: indica que dicho nodo es el origen del error en la base de datos.



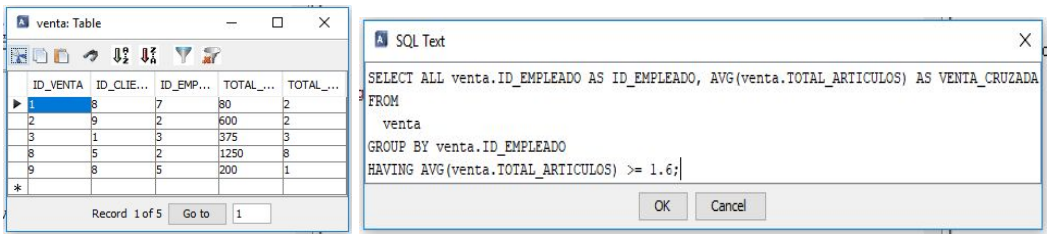
Al finalizarse el proceso de depuración, se muestra cuál era el nodo que causaba el problema, permitiendo ver posteriormente las estadísticas de la depuración y editar dicho nodo:



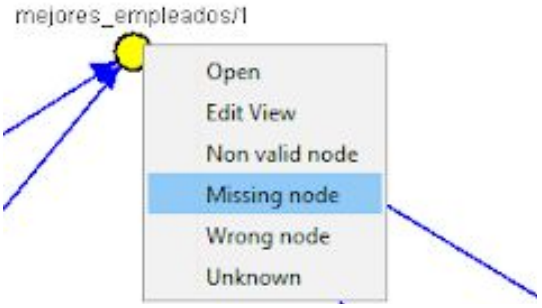
La muestra de estadísticas tiene la información de el número de nodos, el máximo número de preguntas, el número de preguntas, el número de tuplas inspeccionadas, el número de tuplas raíz y el número de tuplas que no son raíz:



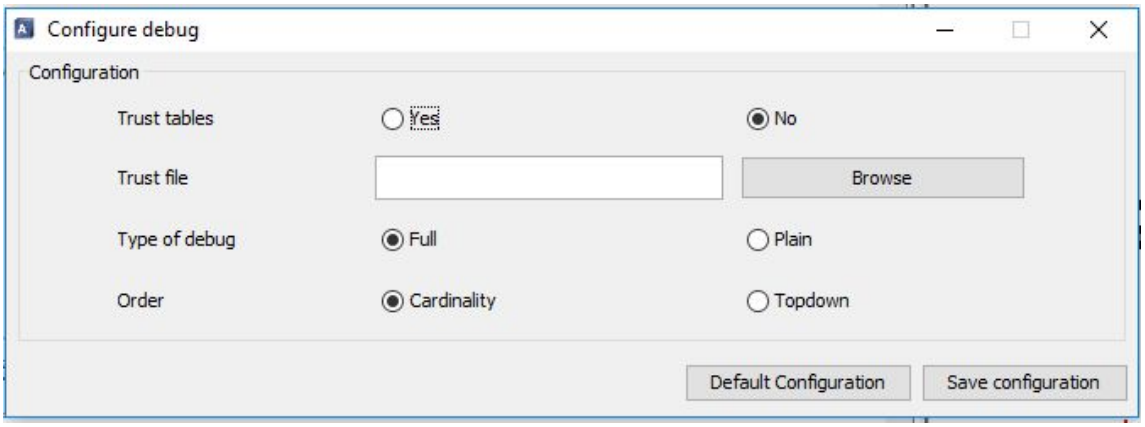
La opción de edición del nodo abrirá una tabla en la que se puede editar la información en caso de ser tabla o la edición de la formulación en SQL de la definición de la vista en caso de serlo:



También se han añadido opciones al grafo de depuración con menús contextuales en cada nodo haciendo clic derecho sobre el mismo con las diferentes opciones que dispone el nodo. De esta forma se puede hacer una depuración más rápida aún ya que permite al usuario elegir el nodo que quiere depurar dando una mayor libertad al usuario de poder depurar sus bases de datos. Estas opciones no se encuentran disponibles todavía en la última versión de DES publicada.



En caso de pulsar el botón de configurar la depuración se mostrará una nueva ventana con las opciones descritas en el apartado anterior:



Con todo esto descrito anteriormente el funcionamiento es el siguiente: el usuario selecciona la vista que ha detectado como incorrecta en el desplegable de vistas que hay en el panel 'Debug SQL' y después comienza la depuración a través del botón 'Start Debug' del panel de 'Debug SQL' o a través del menú contextual de dicho nodo.

En caso de que haya pulsado el botón 'Start Debug' comenzará la depuración guiada y mostrará la información de los nodos y las opciones disponibles de dicho nodo.

En caso de que el usuario haya seleccionado las opciones del menú contextual se realizará la depuración libre en la que el usuario seguirá seleccionando las opciones del menú contextual.

En cualquier momento el usuario puede cambiar de modo de depuración que le convenga más en ese momento.

Cuando el proceso de depuración termine obtendrá la ventana que muestra cual es el nodo incorrecto y las opciones de mostrar estadísticas y la de edición del nodo erróneo.

A continuación mostraremos en la práctica todo lo explicado anteriormente para ver la utilidad y eficiencia de realizar una depuración gráfica en contraposición de realizarlo con una interfaz textual.

6. DEPURACIÓN DECLARATIVA EN PRÁCTICA

En este apartado mostraremos en la práctica el funcionamiento de la depuración declarativa. Para realizarlo, vamos a exponer

Para este ejemplo de depuración vamos a definir una base de datos de una tienda de moda 'Parfuá' que contiene 4 tablas: empleados, artículos, ventas e info_venta.

En esta tienda les gusta recompensar a sus mejores dependientes ya que participan activamente durante el proceso de venta para aconsejar e intentar vender todo lo posible. Para hacer la distinción de sus mejores dependientes tienen que superar una serie de requisitos en base a la información almacenada es sus bases de datos a lo largo del mes. Estos requisitos suelen ser muy exigentes ya que a los que superan todos los filtros 3 meses seguidos tienen la posibilidad de ascenso y una semana de vacaciones pagadas a cargo de la empresa.

Las restricciones impuestas para la obtención de mejores empleados son las siguientes:

- Los dependientes deben tener una media de ventas cruzadas mayor a 1,6. Las ventas cruzadas corresponden a la cantidad de productos vendidos en un mismo ticket, e.g. un dependiente vende 2 productos a un cliente y después vende otro producto a un cliente distinto, las ventas cruzadas corresponden a 1,5 ya que ha vendido 3 productos a 2 clientes.
- Tienen que haber participado activamente en el top de productos más vendidos por la tienda.
- Tienen que tener ventas con un importe mayor a 50 euros.
- Tienen que tener un rendimiento de sueldo mayor a 5, es decir, las ventas que hace a lo largo del mes tienen que ser al menos 5 veces su sueldo.

En la figura X.X se muestra las tablas de la base de datos Parfuá. Las claves primarias se muestran subrayadas. La figura Y.Y muestra las vistas creadas para la selección de los mejores empleados de nuestra tienda. La primera vista es *buenas_ventas(id_empleado, numero_ventas)* que obtiene el id de los empleados que han hecho ventas con importes superiores a 50 euros y el número de veces que lo han conseguido. La segunda vista *top_artículos(id_artículo, cantidad)* muestra los artículos más vendidos de la tienda y sus unidades vendidas. La tercera vista muestra *top_articulo_empleado(id_empleado, id_articulo)* recoge el artículo más vendido de cada empleado. La cuarta vista *empleado_en_top(id_empleado)* muestra los empleados cuyo producto más vendido esté en la vista *top_artículos.* La quinta vista *ventas_cruzadas(id_empleado, venta_cruzada)* obtiene los ids de los empleados cuya venta cruzada media sea mayor que 1,6 junto a su venta cruzada. La sexta vista *sueldos_rentables(id_empleado, sueldo)* obtiene los id de los empleados y sueldo de aquellos cuyas ventas sean al menos 5 veces superiores a su sueldo. La séptima vista *mejores_empleados(id_empleado)* recoge a los empleados que están en *empleado_en_top*, *buenas_ventas*, *ventas_cruzadas* y *sueldos_rentables*.

```
   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) ;

   CREATE OR REPLACE VIEW buenas_ventas(ID_EMPLEADO, NUMERO_VENTAS)
AS
     select `venta`.`ID_EMPLEADO` AS `ID_EMPLEADO`,count(0) AS
`NUMERO_VENTAS`
     from `venta`
     where (`venta`.`TOTAL_VENTA` >= 50)
     group by `venta`.`ID_EMPLEADO` ;

   CREATE OR REPLACE VIEW mejores_articulos(ID_ARTICULO, CANTIDAD)
AS
     select `info_venta`.`ID_ARTICULO` AS
`ID_ARTICULO`,sum(`info_venta`.`CANTIDAD`) AS `CANTIDAD`
     from `info_venta`
     group by `info_venta`.`ID_ARTICULO`
     order by sum(`info_venta`.`CANTIDAD`) desc;
     --LIMIT 5;

   CREATE OR REPLACE VIEW top_articulos(ID_ARTICULO) AS
     select mejores_articulos.ID_ARTICULO
     from mejores_articulos
     ORDER BY mejores_articulos.CANTIDAD DESC
     FETCH FIRST 5 ROWS ONLY;
```

```
CREATE OR REPLACE VIEW top_articulos_empleado(ID_EMPLEADO,
ID_ARTICULO, CANTIDAD)  AS
    select `venta`.`ID_EMPLEADO` AS
`ID_EMPLEADO`,`info_venta`.`ID_ARTICULO` AS
`ID_ARTICULO`,sum(`info_venta`.`CANTIDAD`) AS `CANTIDAD`
    from (`venta` join `info_venta`)
    where  (`venta`.`ID_VENTA` = `info_venta`.`ID_VENTA`)
    group by `ID_ARTICULO`, `ID_EMPLEADO`
    order by sum(`info_venta`.`CANTIDAD`) desc;

CREATE OR REPLACE VIEW ventas_empleado(ID_EMPLEADO, TOTAL_VENTA)
AS
    select venta.ID_EMPLEADO AS ID_EMPLEADO, sum(venta.TOTAL_VENTA)
as TOTAL_VENTA
    from venta
    group by ID_EMPLEADO;

CREATE OR REPLACE VIEW sueldos_rentables(ID_EMPLEADO, SUELDO)  AS
    select empleado.ID AS ID, empleado.SUELDO AS SUELDO
    from (empleado join ventas_empleado)
    where empleado.ID = ventas_empleado.ID_EMPLEADO and
(ventas_empleado.TOTAL_VENTA >= (empleado.SUELDO * 5));

CREATE OR REPLACE VIEW empleado_en_top(ID_EMPLEADO)  AS
    select `top_articulos_empleado`.`ID_EMPLEADO` AS `ID_EMPLEADO`
    from `top_articulos_empleado`,`top_articulos`
    where (`top_articulos_empleado`.`ID_ARTICULO` =
`top_articulos`.`ID_ARTICULO`) ;
```

```
CREATE OR REPLACE VIEW mejores_empleados(ID_EMPLEADO)  AS
  select `empleado_en_top`.`ID_EMPLEADO` AS `ID_EMPLEADO`
  from (((`empleado_en_top` join `sueldos_rentables`) join
`buenas_ventas`) join `ventas_cruzadas`)
  where ((`empleado_en_top`.`ID_EMPLEADO` =
`sueldos_rentables`.`ID_EMPLEADO`)
  and (`sueldos_rentables`.`ID_EMPLEADO` =
`buenas_ventas`.`ID_EMPLEADO`)
  and (`buenas_ventas`.`ID_EMPLEADO` =
`ventas_cruzadas`.`ID_EMPLEADO`)) ;
```

Ha ocurrido un escándalo, uno de nuestros dependientes ha salido como mejor empleado en el mes de agosto y resulta que estaba de vacaciones, vamos a tener que depurar nuestra base de datos para averiguar qué ha ocurrido.

6.1. Depuración Declarativa SQL en DES

A continuación vamos a exponer una depuración a través de DES, para demostrar cómo sería una depuración a través de la aplicación que podemos descargar actualmente desde https://sourceforge.net/projects/des/files/latest/download.

Los comandos para realizar la depuración a través de DES son:

- /process archivo.sql para procesar el archivo y alimentar el sistema con la información que queremos depurar.
- /debug_sql nombre_vista

Comenzamos la depuración con el comando /debug_sql seleccionando la vista que hemos detectado como no válida, en este caso 'mejores_empleados' poniendo la opción de no confiar en las tablas ya que creemos que puede haber un fallo en las mismas.

```
DES> /debug_sql mejores_empleados trust_tables(no)

Info: Question: all(mejores_empleados)
Info: Debugging view 'mejores_empleados'.
{
 1-mejores_empleados(2)
```

}

Decimos que la tupla 1 es incorrecta como habíamos visto anteriormente.

Input: Is this the expected answer for view 'mejores_empleados'? (y/n/m/mT/w/wN/a/h) [n]: w1

Info: 15 rules listed.

Info: Question: all(sueldos_rentables)

Info: Debugging view 'sueldos_rentables'.

{

1-sueldos_rentables(2,300)

}

Seleccionamos que la vista no es válida ya que sabemos que el empleado con ID 2 no puede estar dentro de la vista sueldos_rentables.

Input: Is this the expected answer for view 'sueldos_rentables'? (y/n/m/mT/w/wN/a/h) [y]: n

Info: 16 rules listed.

Info: Question: subset(buenas_ventas_slice1,buenas_ventas)

Info: Debugging view 'buenas_ventas'.

La siguiente pregunta y las consecutivas nos pregunta por la inclusión de la tupla que tiene empleado con ID 2. En este caso en concreto respondemos que no puesto que sabemos que la vista buenas_ventas está relacionada con ventas y el empleado con ID 2 no debería tenerlas.

Input: Should 'buenas_ventas' include the tuple '2,1'? (y/n/a) [y]: n

Info: processAnswer(subset(buenas_ventas_slice1,buenas_ventas),wrong(buenas_ventas(2,1)))

Info: 18 rules listed.

Info: Question: subset(empleado_en_top_slice1,empleado_en_top)

Info: Debugging view 'empleado_en_top'.

En este caso ocurre lo mismo, decimos que no es válida.

Input: Should 'empleado_en_top' include the tuple '2'? (y/n/a) [y]: n

Info: processAnswer(subset(empleado_en_top_slice1,empleado_en_top),wrong(empleado_en_top(2)))

Info: 21 rules listed.

Info: Question: subset(sueldos_rentables_slice1,sueldos_rentables)

Info: Debugging view 'sueldos_rentables'.

Para que un empleado tenga un sueldo rentable debe tener unas ventas 5 veces superior a su sueldo por lo tanto no es válido.

Input: Should 'sueldos_rentables' include the tuple '2,300'? (y/n/a) [y]: n

Info: processAnswer(subset(sueldos_rentables_slice1,sueldos_rentables),wrong(sueldos_rentables(2,300)))

Info: 23 rules listed.

Info: Question: subset(empleado_slice1,empleado)

Info: Debugging table 'empleado'.

Input: Should 'empleado' include the tuple '2,Laura,300'? (y/n/a) [y]: y

Info: processAnswer(subset(empleado_slice1,empleado),valid)

Info: 24 rules listed.

Info: Question: subset(top_articulos_empleado_slice1,top_articulos_empleado)

Info: Debugging view 'top_articulos_empleado'.

Sucede lo mismo, en el caso de top_articulos_empleado el empleado tiene que tener ventas por lo tanto la tupla que contiene un empleado con ID 2 no debe estar incluida en esta vista.

Input: Should 'top_articulos_empleado' include the tuple '2,14,2'? (y/n/a) [y]: n

Info: processAnswer(subset(top_articulos_empleado_slice1,top_articulos_empleado),wrong(top_articulos_empleado(2,14,2)))

Info: 26 rules listed.

Info: Question: subset(top_articulos_slice1,top_articulos)

Info: Debugging view 'top_articulos'.

En este caso sabemos que el artículo con ID 14 sí que es uno de nuestros artículos más vendidos este mes, decimos que debe estar incluido en esta vista.

Input: Should 'top_articulos' include the tuple '14'? (y/n/a) [y]: y

Info: processAnswer(subset(top_articulos_slice1,top_articulos),valid)

Info: 27 rules listed.

Info: Question: subset(ventas_cruzadas_slice1,ventas_cruzadas)

Info: Debugging view 'ventas_cruzadas'.

Para que un empleado tenga ventas cruzadas debe tener ventas, así que decimos que no puede estar en la vista ventas_cruzadas.

Input: Should 'ventas_cruzadas' include the tuple '2,5.0'? (y/n/a) [y]: n

Info: processAnswer(subset(ventas_cruzadas_slice1,ventas_cruzadas),wrong(ventas_cruzadas(2,5.0)))

Info: 29 rules listed.

Info: Question: subset(ventas_empleado_slice1,ventas_empleado)

Info: Debugging view 'ventas_empleado'.

Al igual que en la pregunta anterior, no se debe incluir la tupla que tiene el empleado con ID 2 en esta vista.

Input: Should 'ventas_empleado' include the tuple '2,1850'? (y/n/a) [y]: n

Info:
processAnswer(subset(ventas_empleado_slice1,ventas_empleado),wrong(ventas_empleado(2,1850)))

Info: 31 rules listed.

Info: Question: all(top_articulos)

Info: Debugging view 'top_articulos'.

{

1-top_articulos(5),

2-top_articulos(7),

3-top_articulos(9),

4-top_articulos(13),

5-top_articulos(14)

}

Según hemos visto en la información mostrada anteriormente, la vista top_articulos es correcta puesto que sus tuplas son las esperadas.

Input: Is this the expected answer for view 'top_articulos'? (y/n/m/mT/w/wN/a/h) [y]: y

Info: processAnswer(all(top_articulos),valid)

Info: 32 rules listed.

Info: Question: all(venta)

Info: Debugging table 'venta'.

{

1-venta(1,8,7,80,2),

2-venta(2,9,2,600,2),

3-venta(3,1,3,375,3),

4-venta(8,5,2,1250,8),

5-venta(9,8,5,200,1)

}

Como ya sospechábamos anteriormente, la tabla ventas no debería de estar bien puesto que las vistas que dependían de esta eran erróneas. En este caso sabemos que la segunda y cuarta tupla tiene una venta con el ID de empleado 2, en este caso

respondemos que la tabla no es válida y con esto descubrimos el origen de nuestro problema.

Input: Is this the expected answer for table 'venta'? (y/n/m/mT/w/wN/a/h) [y]:
n

Info: processAnswer(all(venta),nonvalid)

Info: 33 rules listed.

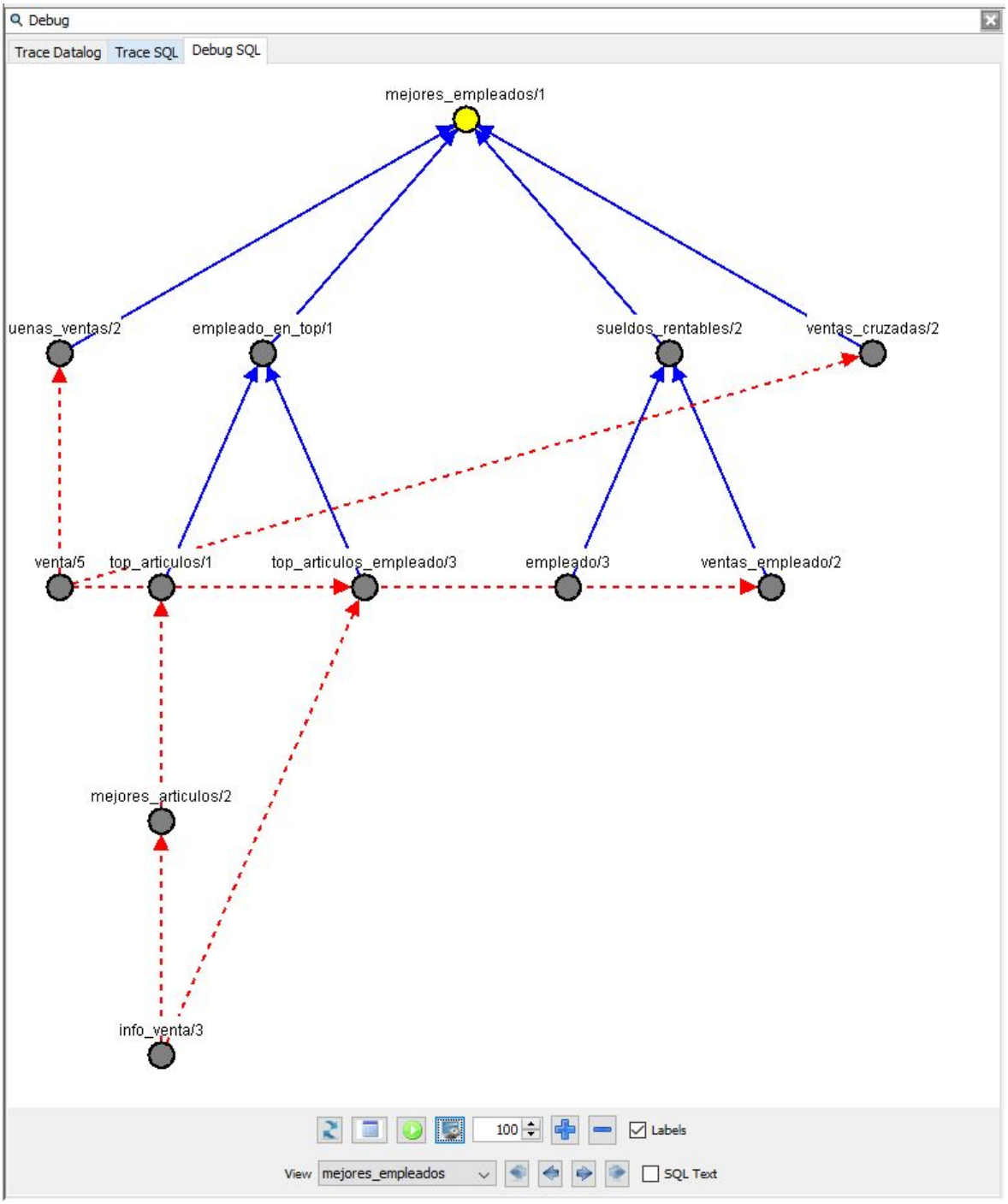
Info: Buggy table found: 'venta'.
Info: Debug Statistics:
Info: Number of nodes : 12
Info: Max. number of questions : 12
Info: Number of questions : 12
Info: Number of inspected tuples: 20
Info: Number of root tuples : 1
Info: Number of non-root tuples : 19

El problema de esta depuración no es solo que el proceso es muy extenso ya que además se han eliminado en la sesión de depuración anterior el listado de reglas mostrado anterior a las líneas 'Info: x rules listed.' , sino que también se puede volver confuso para los usuarios que intentan hacer una depuración de sus bases de datos ya que no pueden ver de forma clara la información que están depurando. En nuestro caso sabíamos que el empleado con ID 2 no podía estar en nuestras vistas de mejor empleado porque estaba de vacaciones pero en caso de haber sido por un fallo en el cálculo de alguna vista al no mostrar escasa información habría sido más complicado para el usuario detectar como válidas o no válidas las vistas.

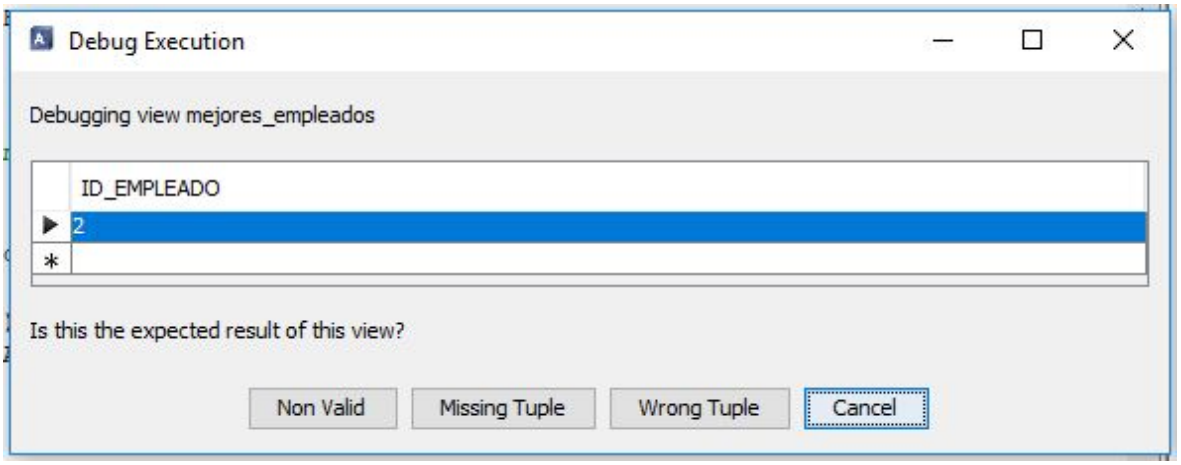
6.2. Depuración Declarativa SQL en ACIDE

Como hemos visto en el apartado anterior el proceso de depuración puede ser bastante tedioso, por eso el objetivo de este TFG es la creación de una interfaz gráfica en la que se pueda hacer una depuración de una base de datos de una forma más rápida e intuitiva.

Únicamente tenemos que abrir ACIDE y procesar nuestro archivo SQL como haríamos en DES. Una vez procesado podemos seleccionar la vista que queremos depurar para mostrar el grafo de dependencias entre vistas y tablas y comenzar la depuración pulsando el botón play.

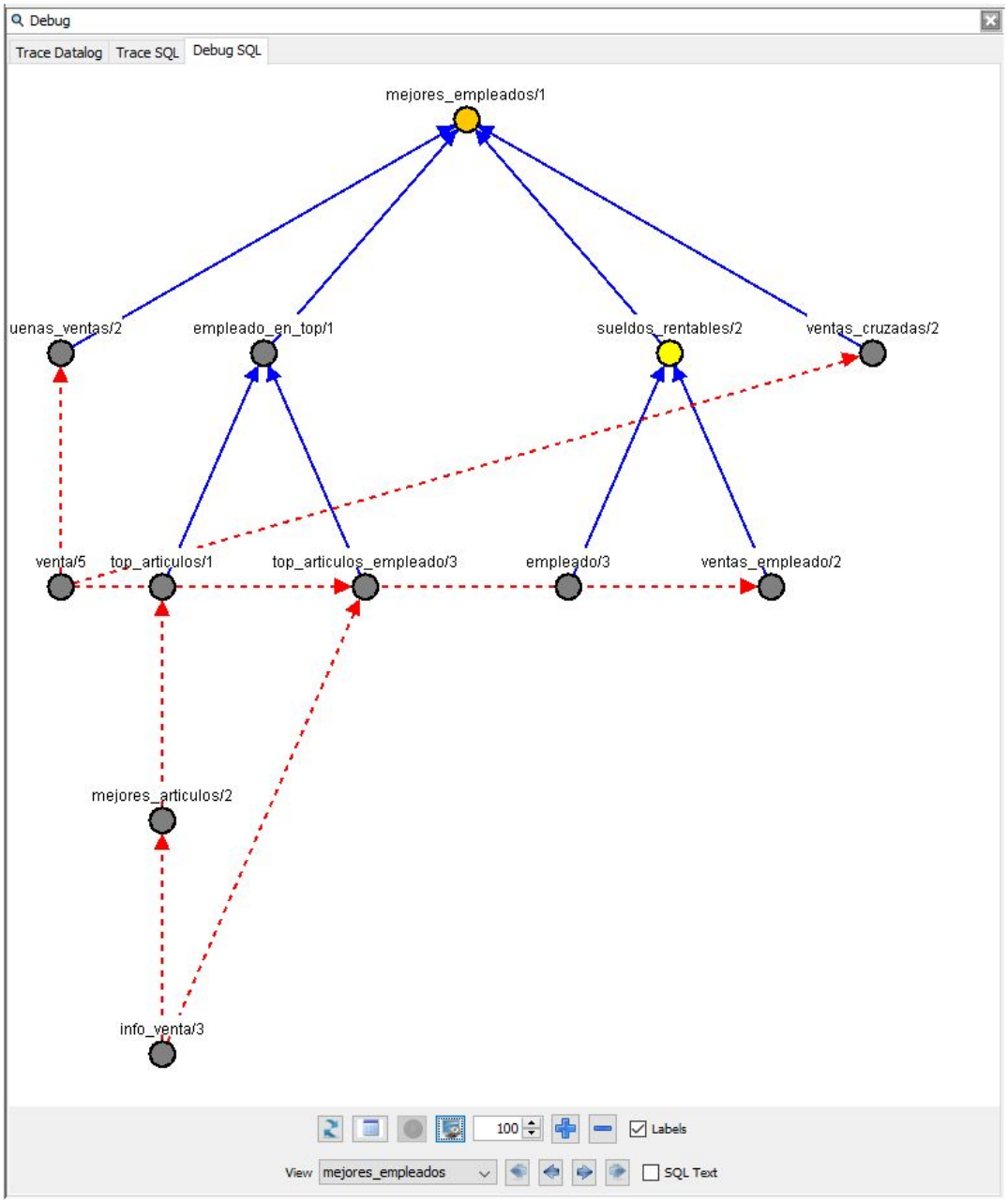


Al pulsarlo se muestra el contenido de la vista y las posibles opciones de depuración.

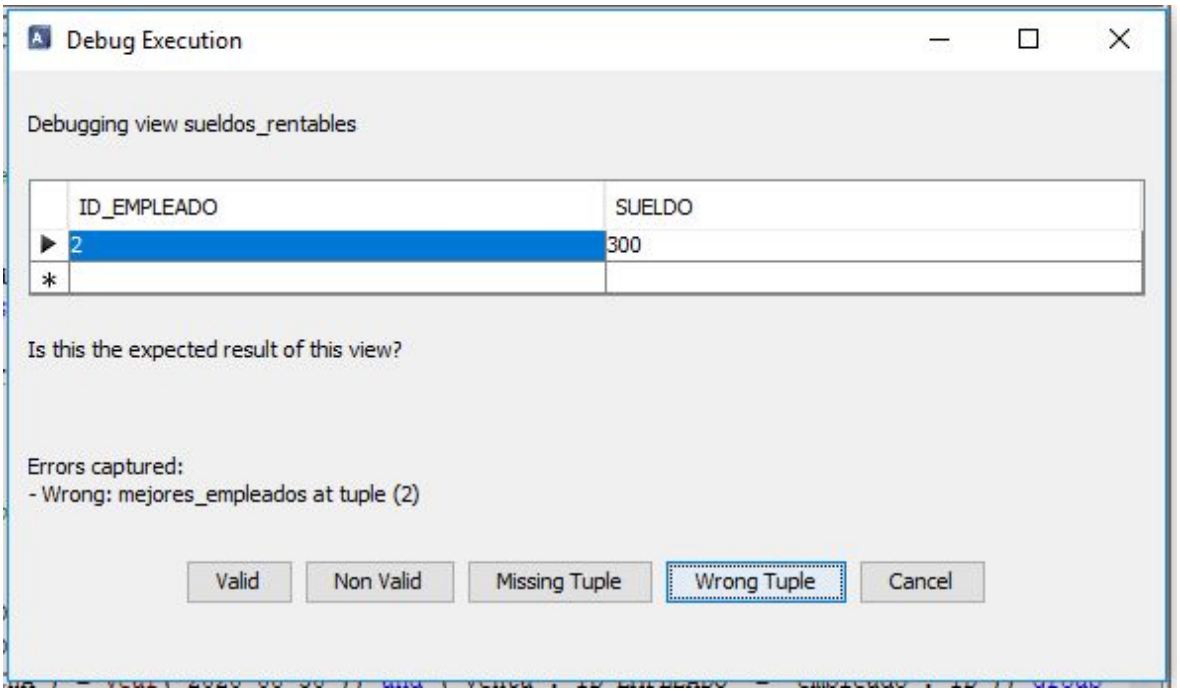


En nuestro caso sabemos que el empleado con ID 2 no debería de estar en la tabla así que pulsaremos el botón 'Wrong Tuple'.

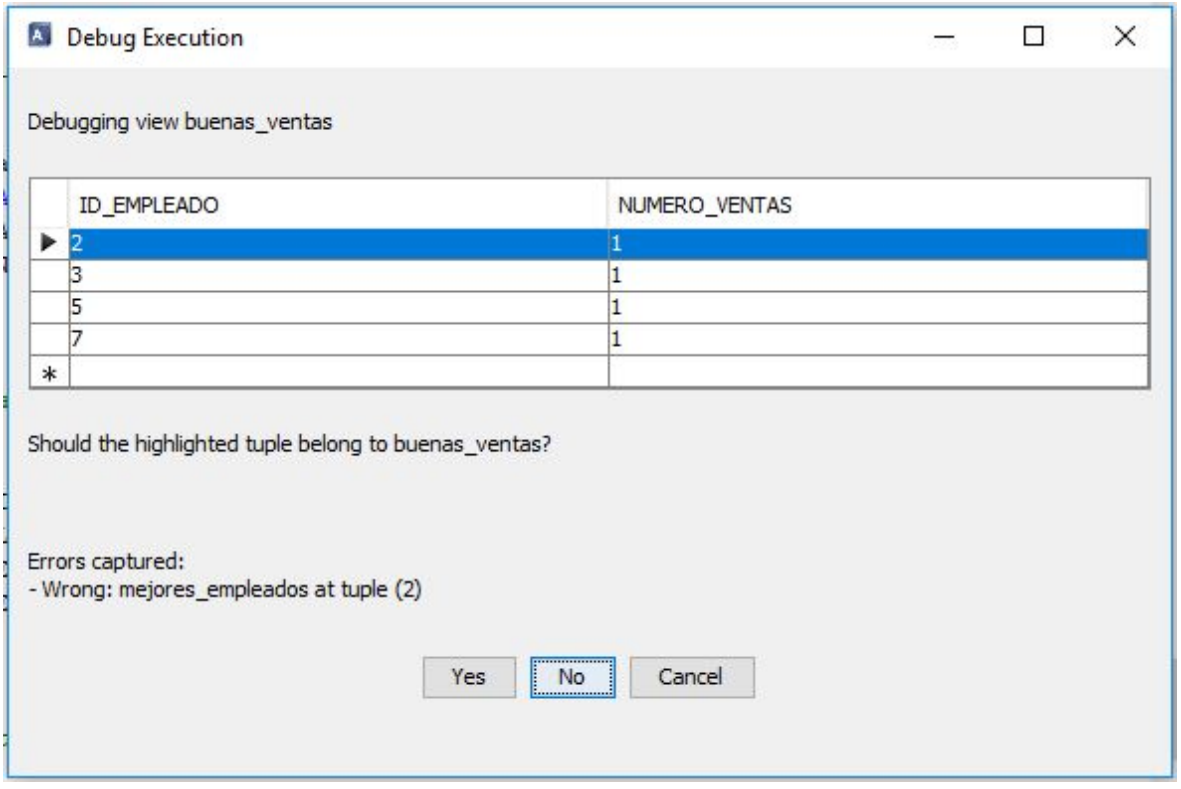
Con cada paso de depuración se marca un nuevo nodo del grafo y se marca con diferentes colores el estado de los nodos explorados. Los colores posibles pueden ser: verde para los nodos válidos, naranja para los nodos no válidos y rojo para los nodos erróneos.



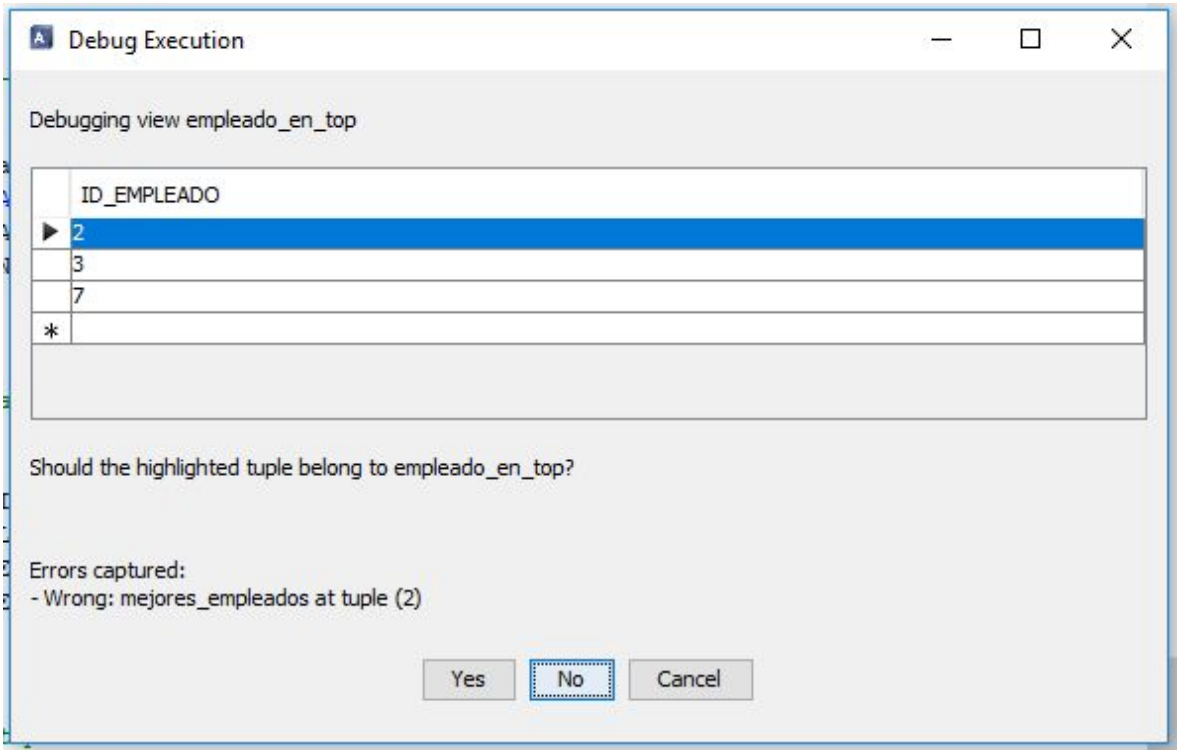
Ahora procedemos a depurar a vista sueldos_rentables, que tampoco es una vista válida, por lo tanto pulsamos el botón 'Non Valid'.



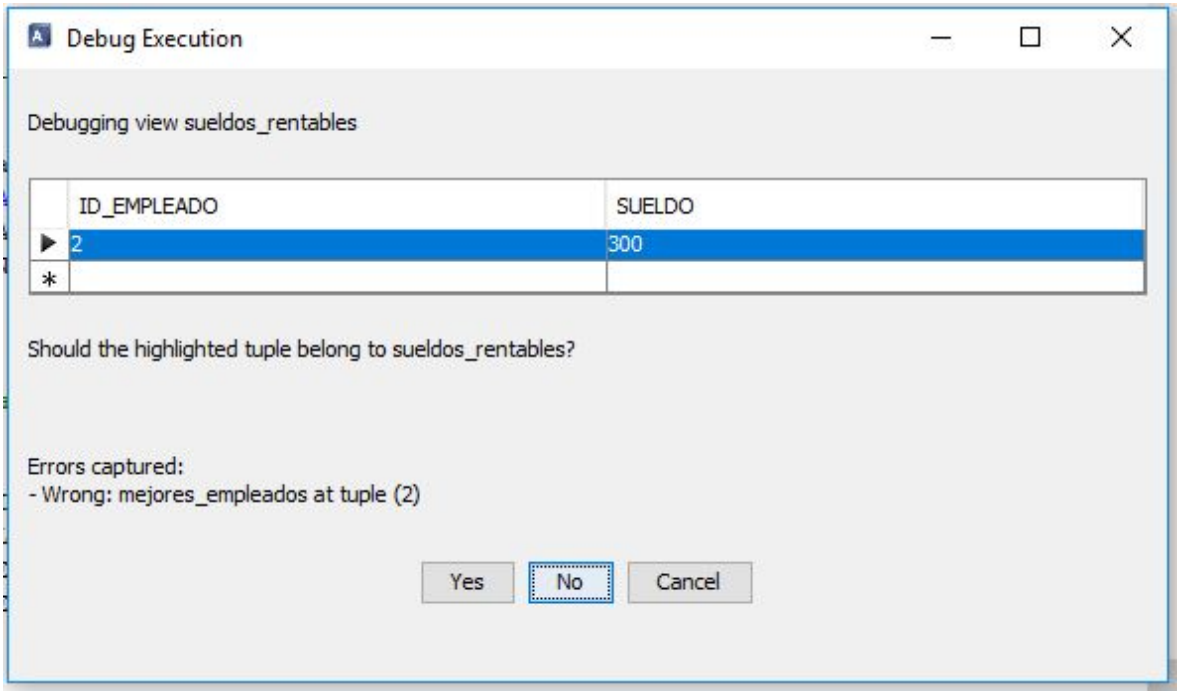
La vista buenas_ventas posee el id 2 por lo tanto no debe pertenecer a esta.



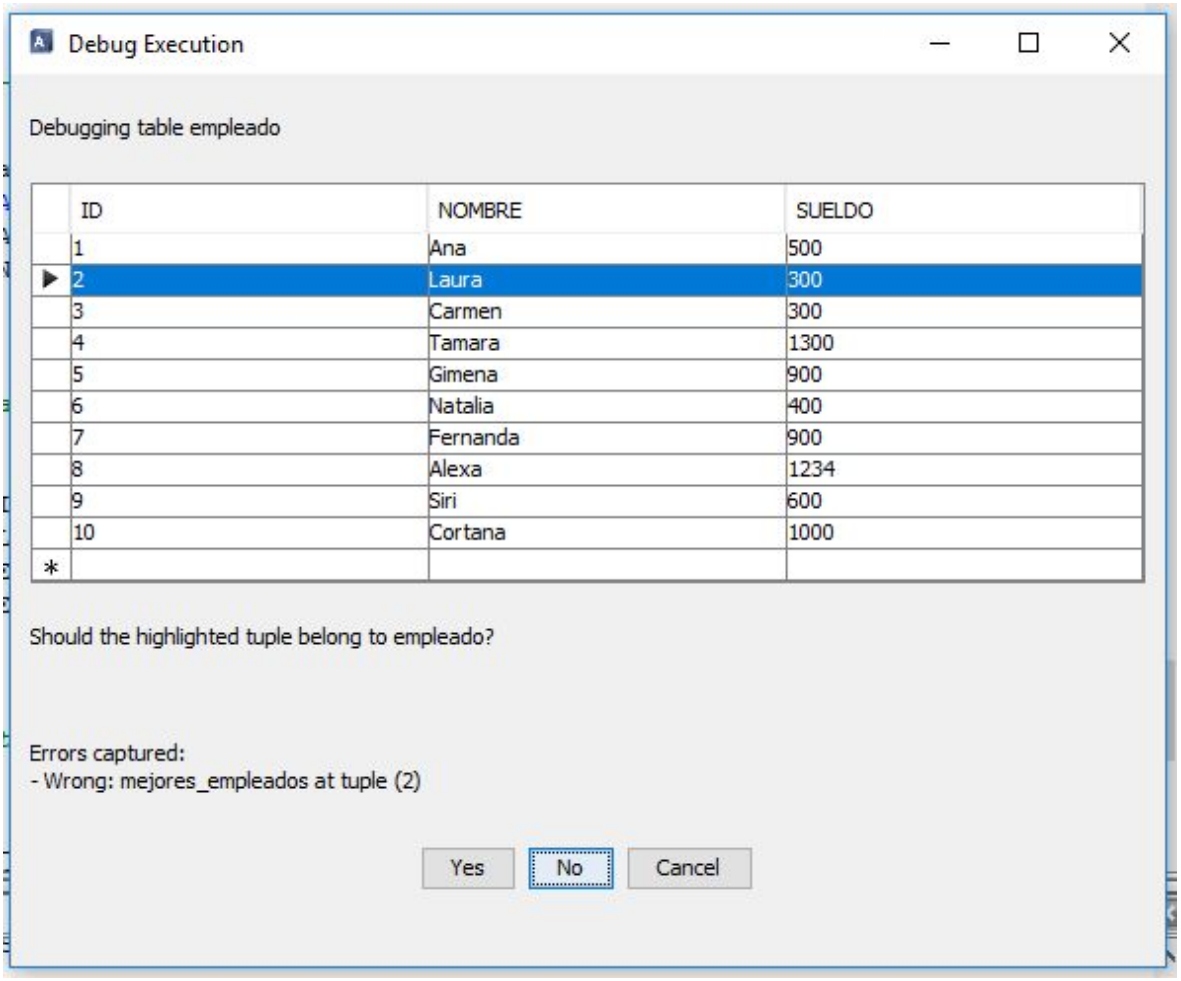
La vista empleado_en_top tiene el id 2 por lo tanto tampoco debería de estar en esta.



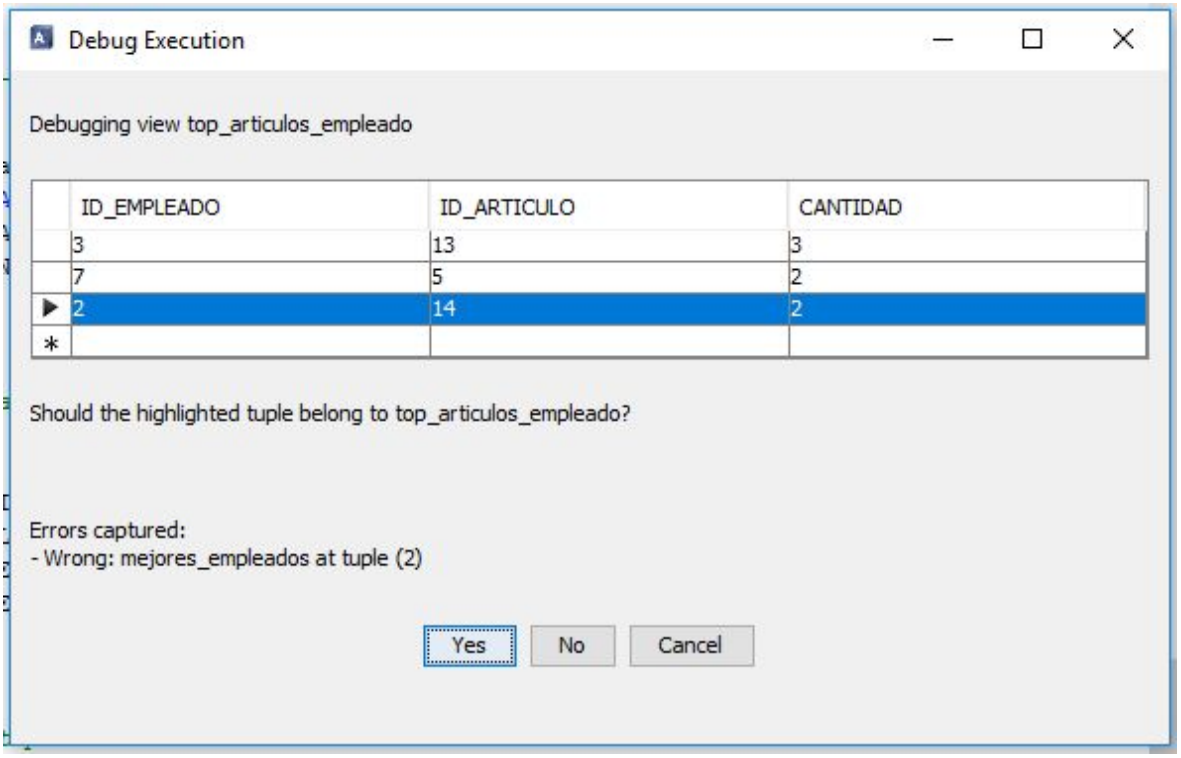
La vista sueldos_rentables no debe tener la tupla con id 2.



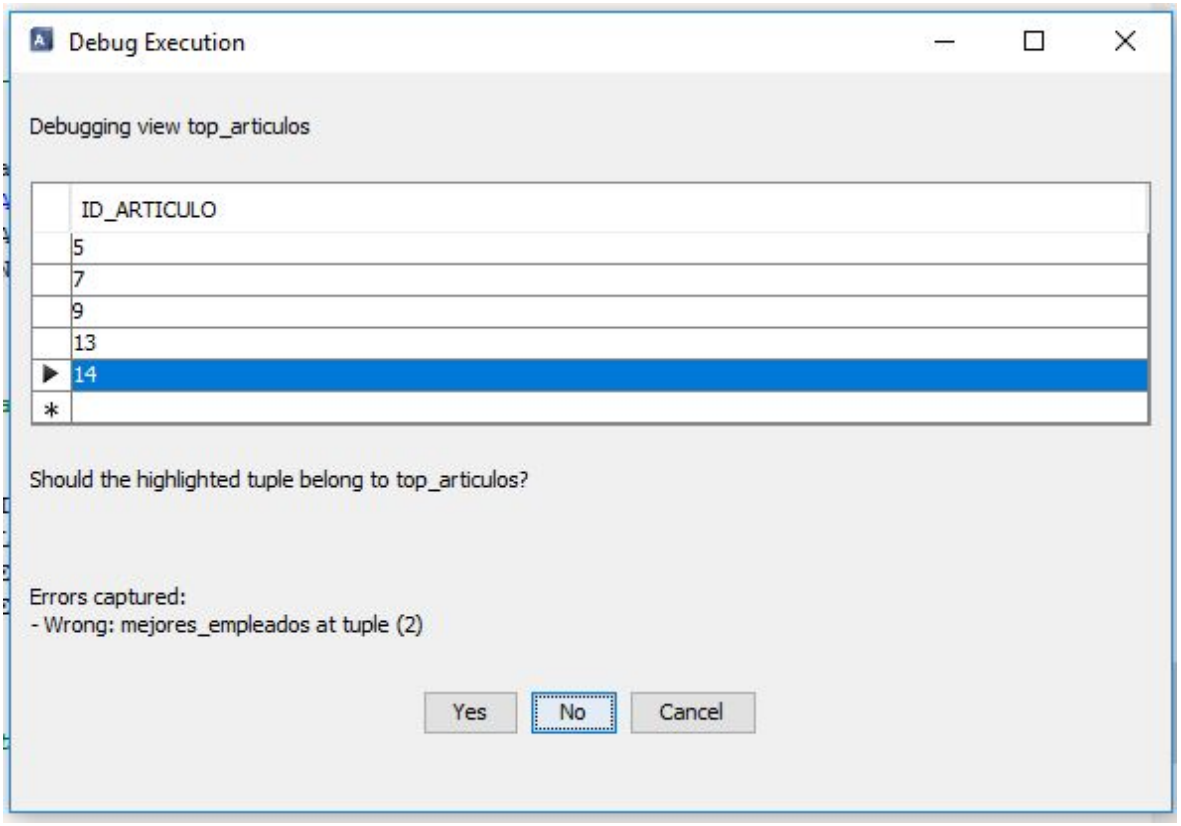
La información de la vista empleados está bien así que pulsamos 'Yes'.



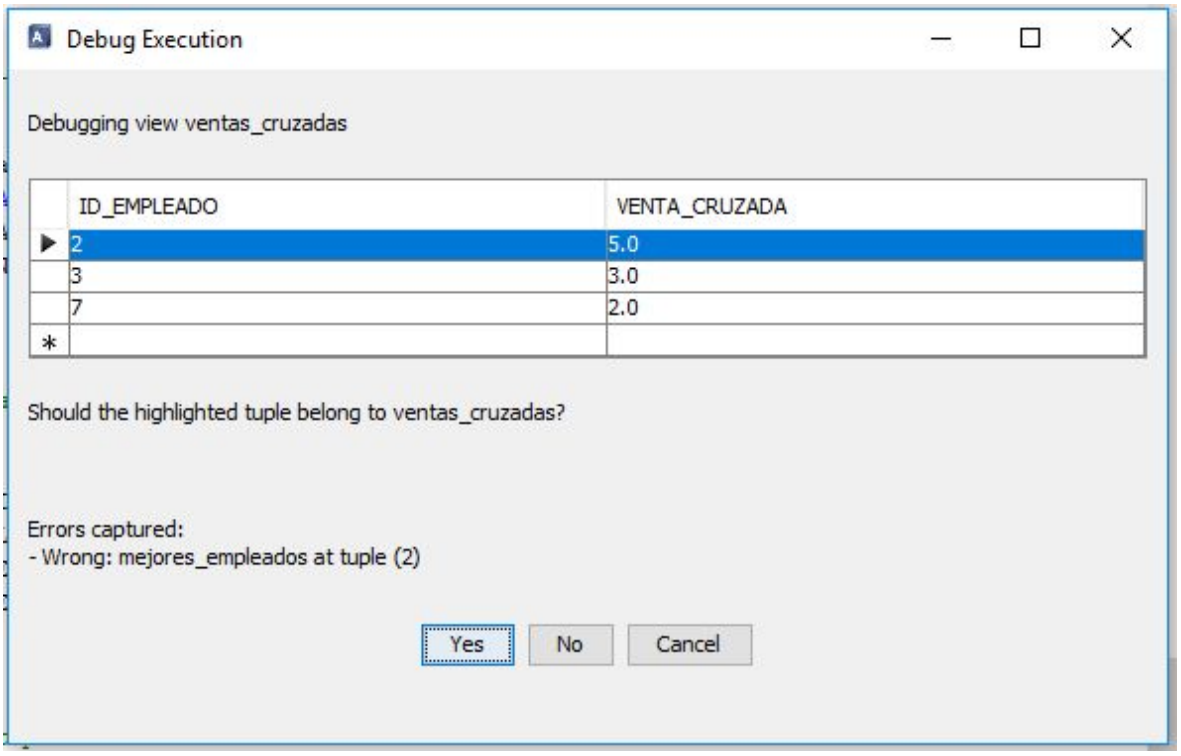
La tupla con ID_EMPLEADO 2 no debería de estar en top_articulos_empleado.



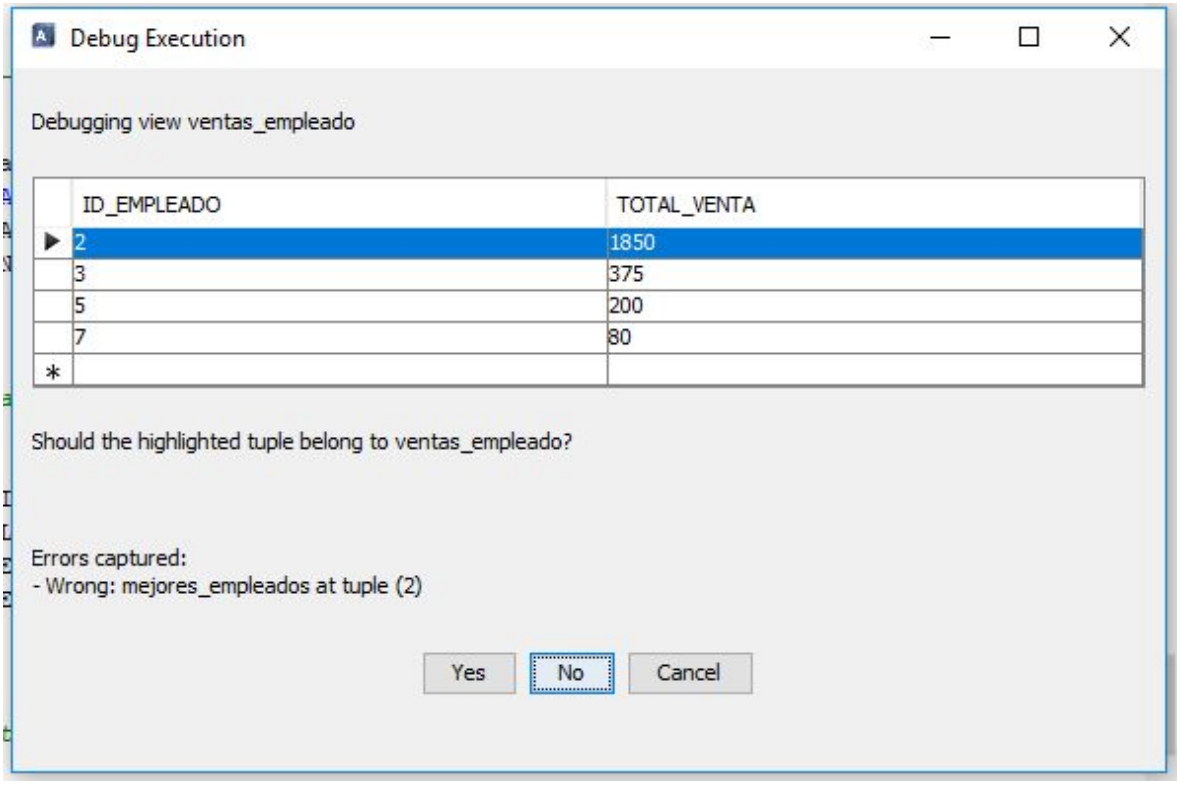
La vista de top_articulos es correcta.



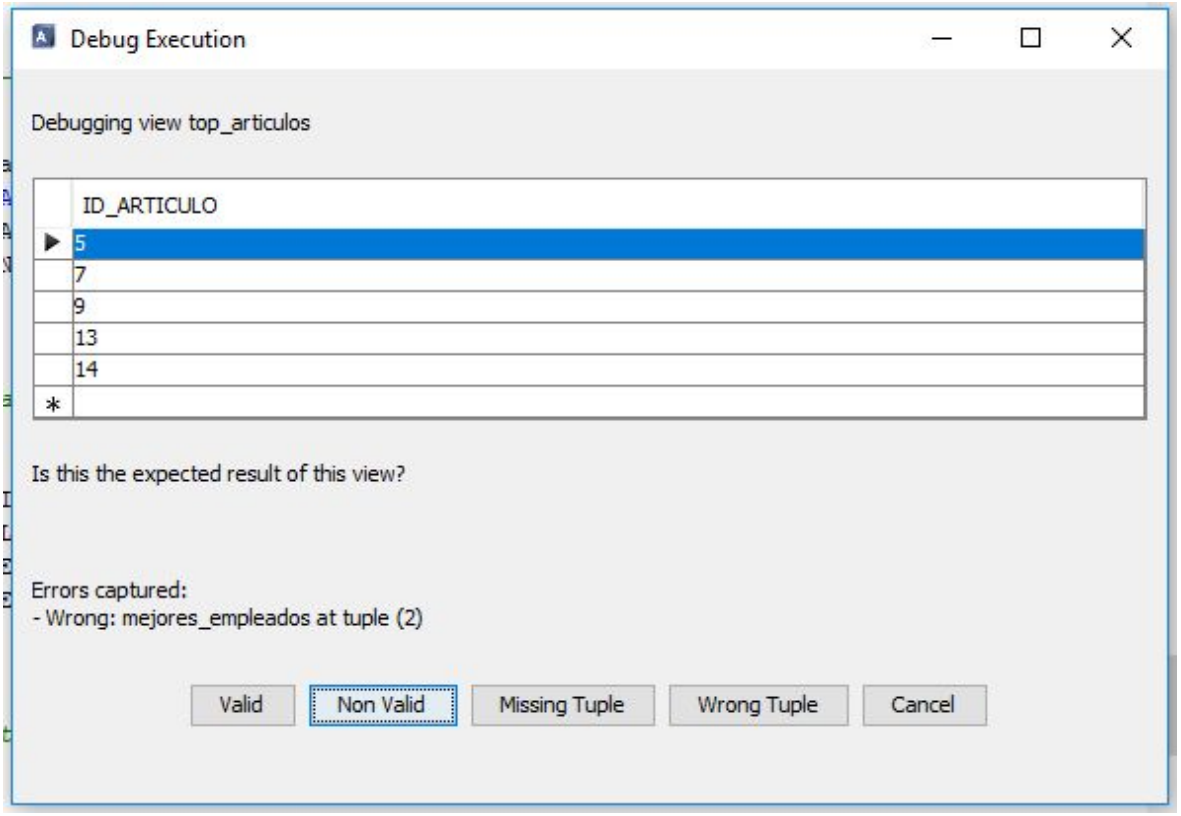
La vista ventas_cruzadas es errónea, no debería tener la tupla con ID_EMPLEADO 2.



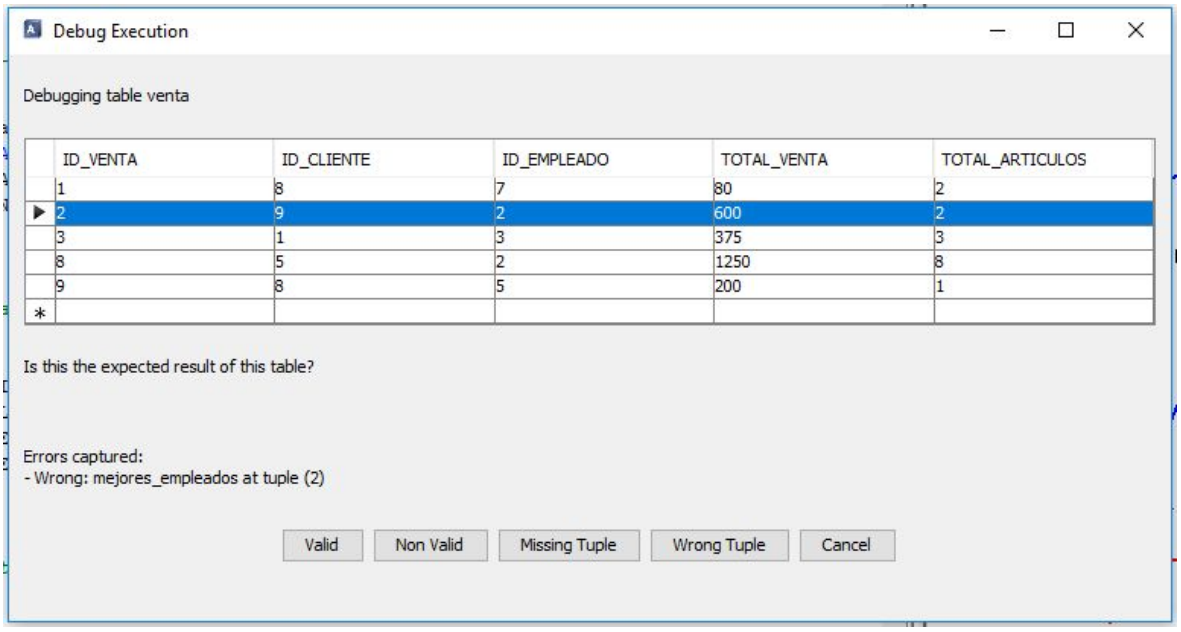
Tampoco tiene que tener la tupla con ID_EMPLEADO 2 la vista ventas_empleado.



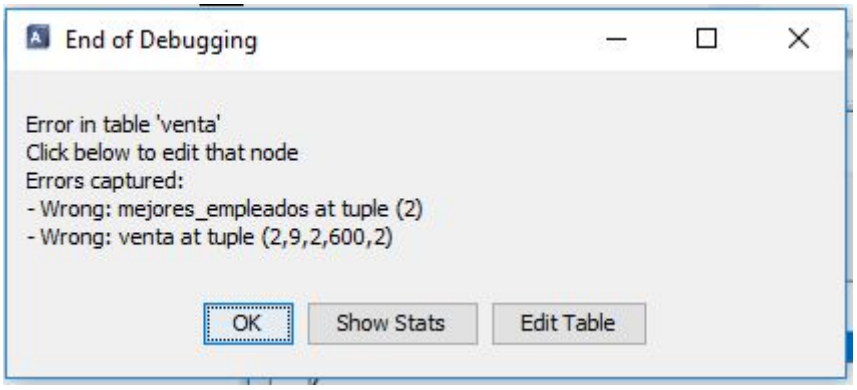
La vista top_articulos es correcta ya que esos artículos se han vendido estos días.



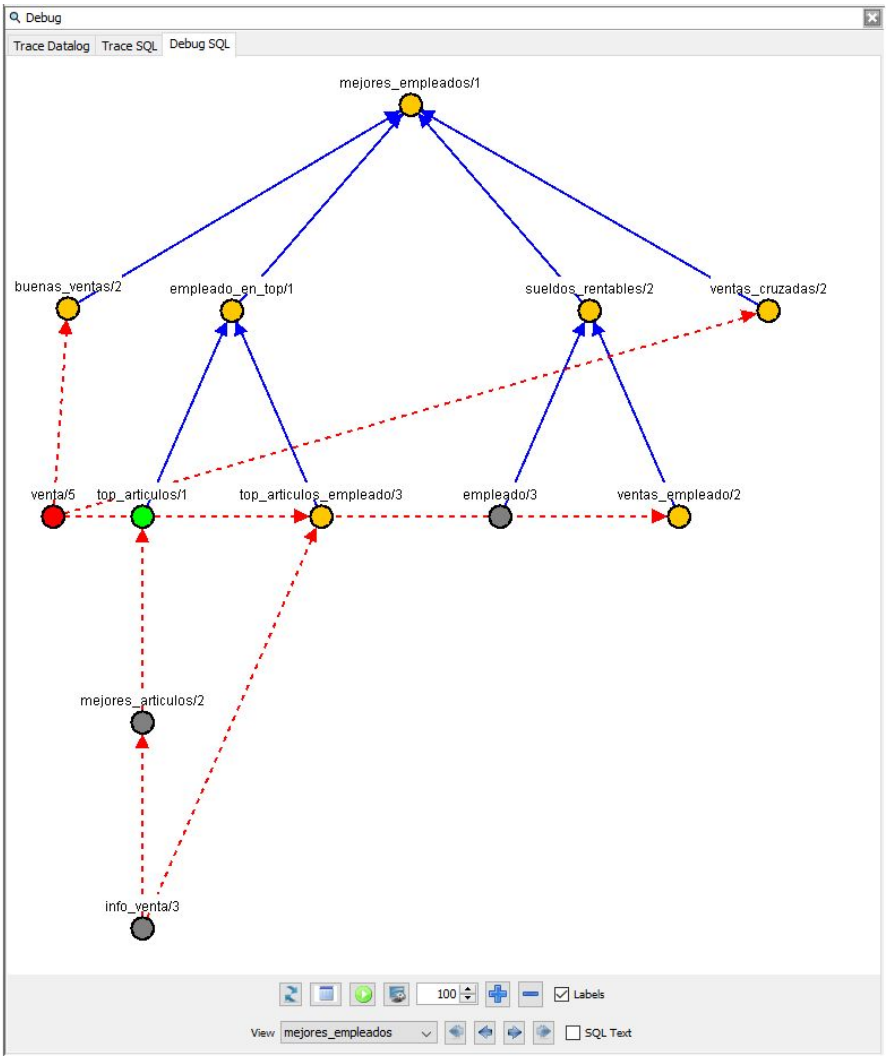
La tupla con ID_EMPLEADO 2 es incorrecta, así que pulsamos el botón 'Wrong Tuple'.



Nuestro proceso de depuración termina porque hemos encontrado cuál era el origen de nuestro problema: se ha registrado una venta en nuestra base de datos que no es posible.



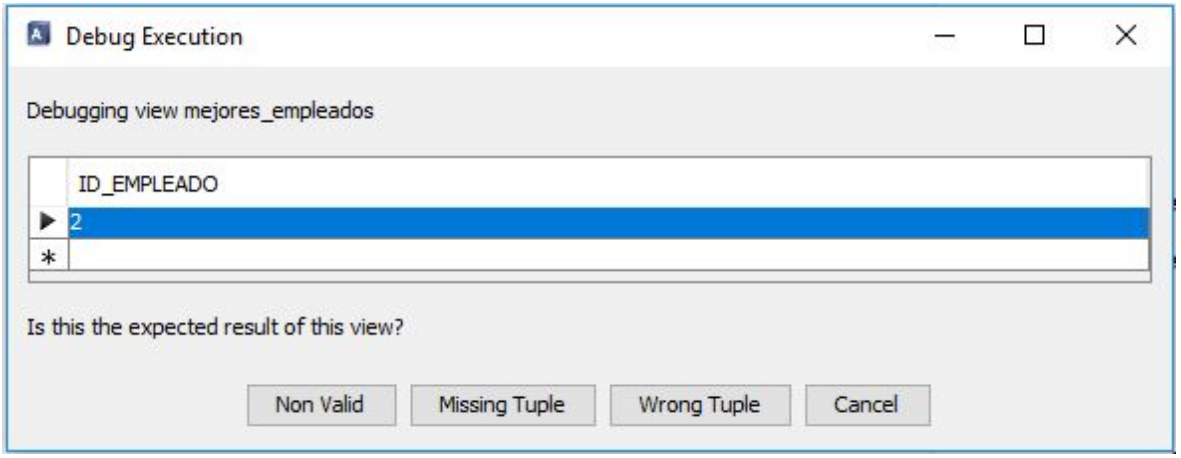
Y nuestro grafo de depuración muestra el estado de los nodos.



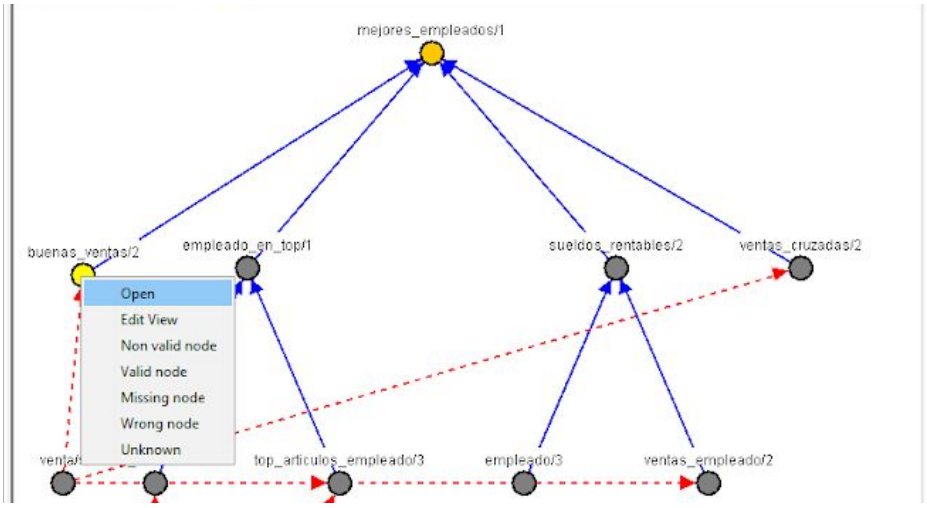
Resulta que durante la primera semana de agosto los sistemas informáticos de la empresa estaban colapsados y los empleados apuntaban sus ventas en un excel y el empleado con ID 3 había registrado mal su mejor venta poniendo el ID 2.

Otro ejemplo que podríamos haber seguido que se añade a versiones anteriores de DES es la posibilidad de asignar un estado a una vista directamente a través de menús contextuales de los nodos del grafo, de esta forma podemos hacer una depuración completa sin tener que seguir necesariamente el camino que nos recomienda DES, y en casos como este que podemos tener sospechas de ciertas vistas o tablas se pueden hacer depuraciones mucho más rápidas.

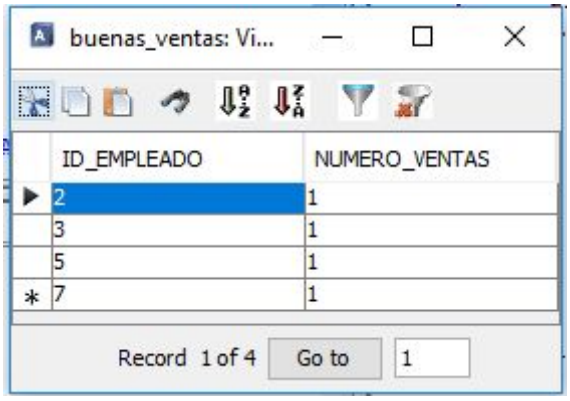
Ahora vamos a hacer la depuración de nuestro sistema a través de los menús contextuales de los nodos del grafo. En este caso empezamos como en el caso anterior, es decir, pulsando el botón 'Start Debug' y definimos que la vista es 'Non Valid'.

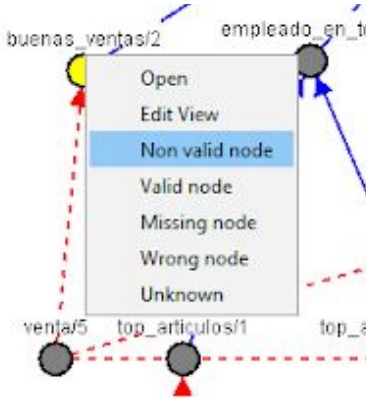


Para que un empleado sea considerado como mejor empleado (en la vista *mejores_empleados*) tiene que tener unas ventas excepcionales, por tanto vamos a enfocarnos más en las vistas relacionadas con la tabla *ventas* como puede ser el caso de *buenas_ventas* y abrimos la vista.

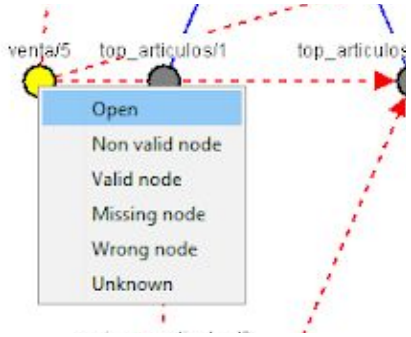


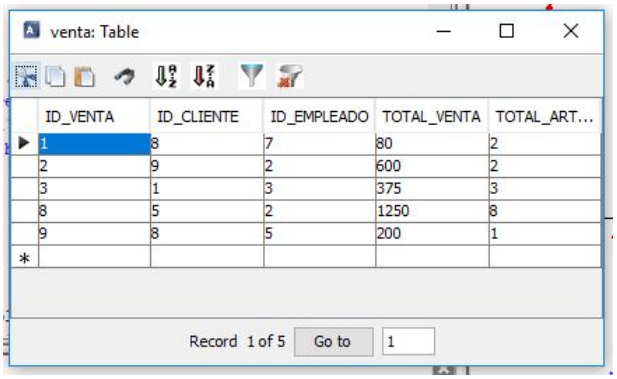
Podemos ver que el empleado con ID 2 no debería de estar así que volvemos al menú contextual y asignamos el estado del nodo a 'Non valid node'.



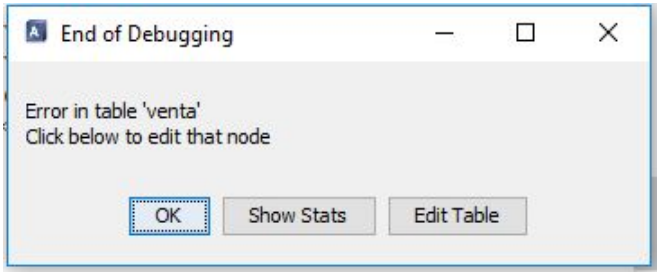


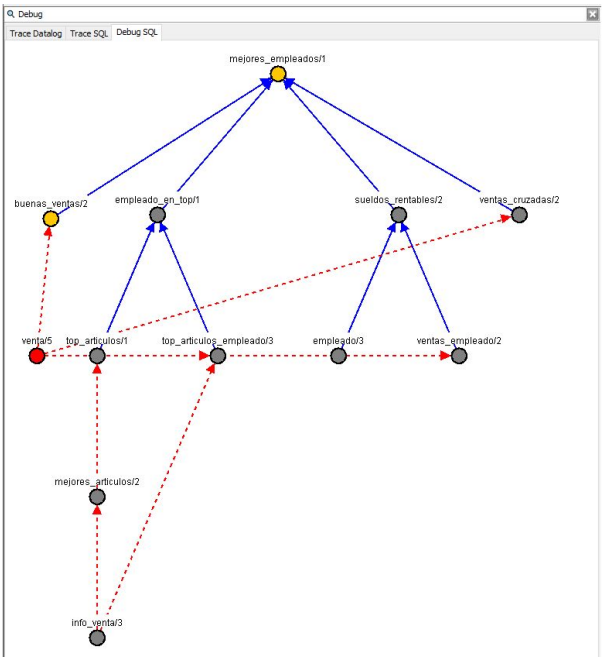
Ahora vamos a ver directamente la tabla *ventas*, ya que un empleado no puede estar dentro de *mejores_empleados* sin tener ventas y la abrimos.





Observamos que la tabla ventas efectivamente no estaba bien así que pulsamos 'Non valid node' y vemos que el origen del error es la tabla *ventas*.





Esta es una forma de depuración mucho más rápida aún que la anterior permitiendo identificar rápidamente el origen del problema de nuestra base de datos.

Esto es una mejora introducida en la aplicación DES por Fernando Sáenz Pérez, el director de este TFG.

De esta forma queda demostrada la eficiencia de realizar depuraciones de bases de datos de forma gráfica ya que el usuario tiene a su disposición toda la información de su base de datos durante su depuración y tiene un mayor control sobre su proceso. Esto es muy importante ya que normalmente una base de datos es mucho más grande que lo expuesto en el ejemplo anterior facilitando un proceso que de otra forma sería muy tedioso y consumiría mucho tiempo.

En caso de querer explorar más ejemplos se pueden ver los ejemplos incluidos en la documentación oficial de **DES** [7].

7. TAREAS DE MANTENIMIENTO REALIZADAS

El objetivo principal de este TFG ha sido la elaboración de la interfaz de usuario para la depuración de bases de datos en ACIDE pero además se ha realizado grandes esfuerzos por hacer mantenimiento de la aplicación y resolución de bugs de años anteriores. A continuación se van a exponer las tareas de mantenimiento y resolución de bugs más notables que se han realizado:

- Homogeneizar los estilos de dibujo de todos los grafos de **ACIDE** haciendo que el nodo raíz se mantenga en la parte superior.
- Creación de un grafo nuevo **RDG** dentro del panel **PDG** y mostrar el grafo al abrir este panel.
- Mantener la configuración léxica de la consola de **ACIDE** al reiniciarla.
- En el panel 'Database' actualizar las vistas antes de mostrarlas, antes mantenían los contenidos del procesamiento del archivo SQL pero no las modificaciones realizadas desde la aplicación.
- Realizar control de errores a la hora de realizar edición de una vista (modificación de la consulta SQL). Esto corrompía al sistema.
- Evitar la modificación de datos de las vistas durante su visualización.
- Añadir la funcionalidad al checkbox 'SQL Text' de abrir y seleccionar el nodo del árbol en el panel 'Database' y centrar dicho nodo en el panel para que el usuario pueda verlo.
- Incrementar el número de comandos entre **ACIDE** y **DES** usen el protocolo **TAPI** para una mejor mantenibilidad de la comunicación entre los dos sistemas y eliminación de comandos duplicados.

- Arreglada la modificación de inserción de datos en las tablas en tuplas que tenían más de 1 columna y la funcionalidad de borrado de tupla.

- Arreglada la visualización de vistas y tablas para las bases de datos externas a **DES.**

- Arreglada la visualización de texto SQL en bases de datos externas a **DES** para la edición de vistas. Estos 2 últimos puntos han sido imprescindibles para el correcto funcionamiento de la depuración en bases de datos externas.

8. HERRAMIENTAS Y TECNOLOGÍAS

La herramienta con la que se ha llevado a cabo el control de versiones ha sido **GitHub [13],** debido a mi experiencia de uso de esta herramienta y al extenso uso de esta, de forma que futuros grupos puedan seguir utilizando este repositorio ya configurado para futuros desarrollos sobre **ACIDE**.

El IDE utilizado para el desarrollo de código ha sido **Intellij** por su gran sistema de depuración y mi experiencia de uso del mismo.

Para el uso de bases de datos externas a través de **ODBC** he usado la herramienta **XAMPP** que tiene como base de datos **MySQL.**

Para el desarrollo de la interfaz he utilizado **Java Swing** ya que este proyecto era una continuación de trabajos de años anteriores y **ACIDE** está desarrollado con esta tecnología.

Para la transferencia de archivos y ejecutables entre alumno y director hemos usado **Google Drive[21]**.

9. DOCUMENTACIÓN

Todos los archivos del proyecto, tanto los archivos de documentación como los archivos de código son objeto de control de la gestión de la configuración. Se ha seguido con la configuración de la gestión descrita en las memorias **[1]**, **[2]**, **[3], [4]**, **[5] y [6].**

En la comunicación entre alumno y director durante la realización del proyecto, se ha llevado a cabo el seguimiento del documento de tareas llamado **"tareas_XX-XX-XXXX"** en la carpeta de entregas de **Google Drive[21]** donde **XX-XX-XXXX** es la fecha de entrega. Este documento de tareas se actualizaba según se realizaban las tareas y se hacían nuevas iteraciones sobre el proyecto.

En este documento, las tareas podían estar en 3 estados posibles

- **PENDIENTE**: Tarea requerida por parte del director para ser implementada.
- **REVISAR**: Implementación realizada por el alumno, pero pendiente de aprobación.
- **OK:** Implementación realizada y aprobada por el director.

Los estándares aplicados en este documento han sido los mismos que en años anteriores para mantener uniformidad de las documentaciones, para más información revisar las memorias referenciadas en **[1]**, **[2]**, **[3], [4]**, **[5] y [6]**.

Con cada iteración en el desarrollo se ha entregado una nueva versión de la aplicación al director que consistía en un archivo ZIP, el archivo ejecutable jar y una actualización en el documento de tareas de **Google Drive [21]**. Dentro del archivo ZIP se encontraba el ejecutable del proyecto, los archivos de configuración, el código del proyecto y archivos SQL para ser procesados por la aplicación. Cada envío se guardaba en una carpeta con la fecha de entrega. De esta forma se almacenan las diferentes versiones de **ACIDE** teniendo un control sobre los avances del proyecto.

10. CONCLUSIONES

Con la finalización del proyecto se han cumplido los objetivos propuestos al inicio del mismo al mismo tiempo que se han resuelto errores que tenía la herramienta y se han realizado otras mejoras mencionadas en el apartado anterior.

Se ha creado una interfaz gráfica afín a la ya existente en **ACIDE.** Se han mantenido todos los estándares de código y documentación establecidos en años anteriores. Esto es muy importante ya que **ACIDE** es un programa de código abierto que se usa en universidades de todo el mundo y probablemente se realizarán trabajos de fin de grado en un futuro relacionados con este sistema.

Hay que tener en cuenta que **ACIDE** es una aplicación bastante grande en la que han intervenido muchos programadores y por tanto los primeros meses de trabajo me ha supuesto un gran esfuerzo para empaparme del funcionamiento de la aplicación y de la infraestructura interna del mismo.

Se pueden procesar todos los ejemplos reflejados en la manual oficial de **DES [7]** tanto en el modo de depuración guiada como en el modo de depuración de los menús contextuales de los nodos de los grafos.

La aplicación **ACIDE** es un **IDE** robusto que puede ser de gran ayuda para nuevos programadores pero puede mejorarse por tanto en el siguiente apartado se exponen posibles mejoras que pueden implementarse en el sistema para una mejora en la experiencia de usuario y mantenibilidad.

11. TRABAJO FUTURO

En esta sección se listan las mejoras más notables que mejorarían el uso de la aplicación:

- Introducir en el fichero de idioma los atajos de teclado, de este modo los atajos quedarían asociados al idioma cargado en la aplicación.
- Incluir un panel en la pantalla principal la base de datos asertada en vez de mostrarla en una nueva ventana.
- Permitir el uso de perspectivas como en **Intellij o Eclipse[10]**. Debido a la gran cantidad de paneles y funcionalidades que posee **ACIDE** sería interesante que se pudieran configurar perspectivas como otros IDE mejorando la experiencia de los usuarios.
- Añadir la funcionalidad de abrir ficheros en el editor de texto de **ACIDE** cuando un usuario pulsa y arrastra un archivo dentro de este.
- Incluir test unitarios y de integración, de esta forma se asegura una mayor mantenibilidad de la aplicación cuando se modifiquen funcionalidades o se crean nuevas versiones de **DES** asegurando la completa compatibilidad entre ambas aplicaciones.

Los puntos anteriores son los que considero más importantes para una mejora de la aplicación pero pueden encontrarse más mejoras en el documento **TODO_ACIDE.doc.** Este documento se encuentra en la carpeta de Google Drive del proyecto.

12. CONCLUSIONS

With the completion of the project, the objectives proposed at the beginning of the project have been met, at the same time that errors in the tool have been resolved and other improvements mentioned in the previous section have been made.

An interface has been implemented according to already implemented within **ACIDE** and all the code and documentation standards established in previous years have been maintained, this is very important since **ACIDE** is an open source program that is used in universities around the world and there will probably be final degree projects in the future related to this system.

Keep in mind that **ACIDE** is a fairly large application in which many programmers have intervened and therefore the first months of work have involved a great effort to soak up the operation of the application and its internal infrastructure.

All the examples reflected in the official **DES** manual **[7]** can be processed both in the guided debugging mode and in the debugging mode of the contextual menus of the graph nodes.

The **ACIDE** application is a robust **IDE** that can be of great help for new programmers but it can be improved, therefore, the next section presents possible improvements that can be implemented in the system for an improvement in the user experience and maintainability.

13. FUTURE WORK

This section lists the most notable improvements that would improve the use of the application:

- Enter the keyboard shortcuts in the language file, in this way the shortcuts would be associated with the language loaded in the application.
- Include a panel on the main screen of the asserted database instead of displaying it in a new window.
- Allow the use of perspectives as in **Intellij** or **Eclipse [10]**. Due to the large number of panels and functionalities that **ACIDE** has, it would be interesting if it could be configured like other IDEs to which users are already accustomed and can configure it to their own liking.
- Add the functionality to open files in the **ACIDE** text editor when a user clicks and drags a file into it.
- Include unit and integration tests, in this way, greater maintainability of the application is ensured when functionalities are modified or new versions of **DES** are created, ensuring complete compatibility between both applications.

The previous points are the ones that I consider most important for an improvement of the application but more improvements can be found in the **TODO_ACIDE.doc** document

14. BIBLIOGRAFÍA

Para desarrollar el presente proyecto se han seguido las siguientes fuentes:

- **Explicación del funcionamiento de los comandos de DES.**
  - *Datalog Educational System V6.5 User's Manual. Fernando Sáenz Pérez. Universidad Complutense de Madrid. 2020.*
- **Teoría sobre la depuración declarativa en SQL:**
  - *Declarative Debugging of Wrong and MissingAnswers for SQL Views. Rafael Caballero, Yollanda García-Ruíz, y Fernando Sáenz Pérez. Universidad Complutense de Madrid. 2013.*
  - *Algorithmic Debugging of SQL Views.R. Caballero, Y. García-Ruiz and F. Sáenz-Pérez. Universidad Complutense de Madrid. 2011.*

15. REFERENCIAS

[1] D. Cardiel Freire, J. J. Ortiz Sánchez y D. Rupérez Cañas. ACIDE: A Configurable IDE. Universidad Complutense de Madrid. 2007.

[2] M. Martín Lázaro. ACIDE 0.2: a configurable integrated development environment. Universidad Complutense de Madrid. 2008.

[3] J. Salcedo Gómez. ACIDE: A Configurable IDE. DES GUI Front-end. Universidad Complutense de Madrid. 2011.

[4] P. Gutiérrez García-Pardo, E. Tejeiro Pérez de Ágreda, A. Vicente del Cura: A Configurable IDE. DES GUI Front-end. Universidad Complutense de Madrid. 2013

[5] J. J. Marqués Ortiz, F. Ordás Lorente y S. Gutiérrez Quintana: A Configurable IDE. DES GUI Front-end. Universidad Complutense de Madrid. 2014

[6] S. Domínguez Fuentes: Acide: a configurable IDE: ACIDE debugging. Universidad Complutense de Madrid. 2015

[7] Manual oficial de DES:
https://www.fdi.ucm.es/profesor/fernan/des/html/manual/manualDES.html

[8] R. Caballero, Y. García-Ruiz, and F. Sáenz-Pérez, Declarative Debugging of Wrong and Missing Answers for SQL Views. Universidad Complutense de Madrid. 2013.

[9]R. Caballero, Y. García-Ruiz and F. Sáenz-Pérez. Algorithmic Debugging of SQL Views. Universidad Complutense de Madrid. 2011.

[10] Página oficial de Eclipse: https://www.eclipse.org/

[11] Página oficial de Google drive: https://drive.google.com/

[12] Página oficial de Intellij: https://www.jetbrains.com/es-es/idea/

[13] Página oficial de GitHub: https://github.com/

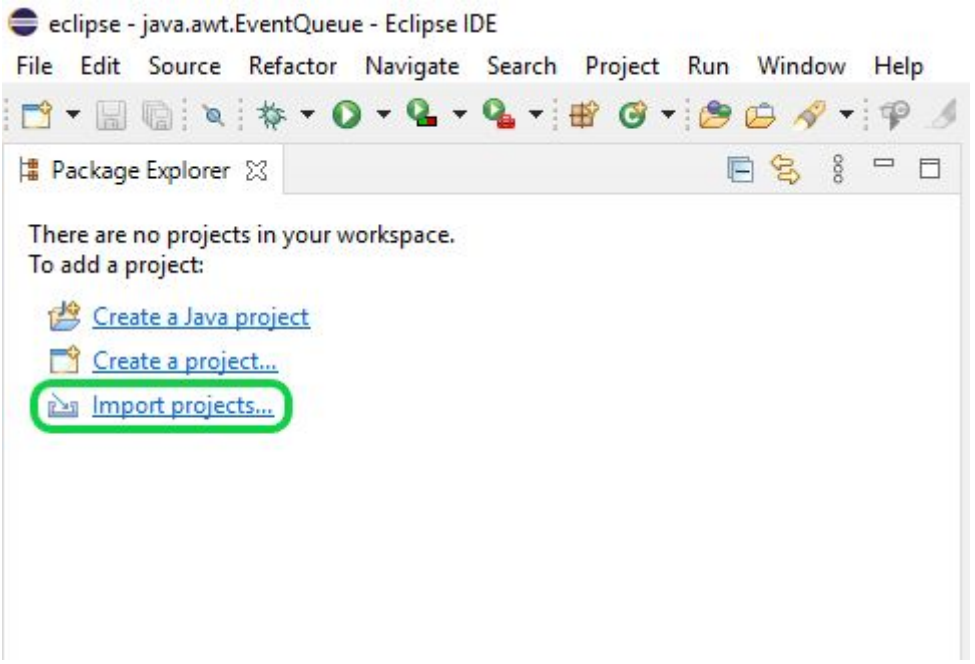
ANEXO: PREPARACIÓN DEL ENTORNO DE DESARROLLO

Para la preparación del entorno vamos a necesitar tener instalados los siguientes componentes:

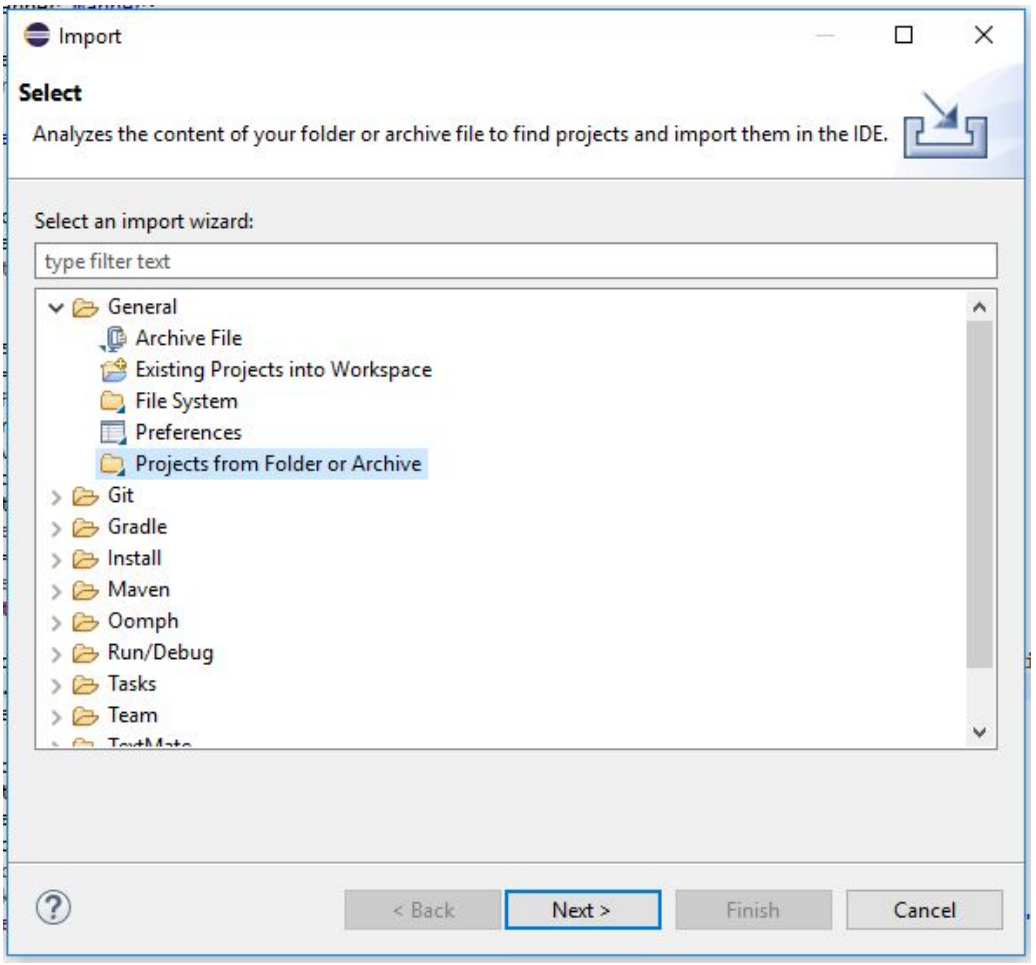
- Java Virtual Machine (JVM).
- JRE 1.8+
- Un IDE que soporte java, recomendados Intellij y Eclipse.
- En caso de usar un sistema operativo **UNIX** hay que dar permisos de ejecución a los ejecutables **DES**, esto es muy importante ya que si no damos permisos de ejecución a los ejecutables no funcionará nuestro sistema. Para hacerlo hay que emitir el siguiente comando en la raíz del proyecto: **sudo chmod +x des des_start**

Para la preparación del entorno en Eclipse hay que seguir los siguientes pasos:

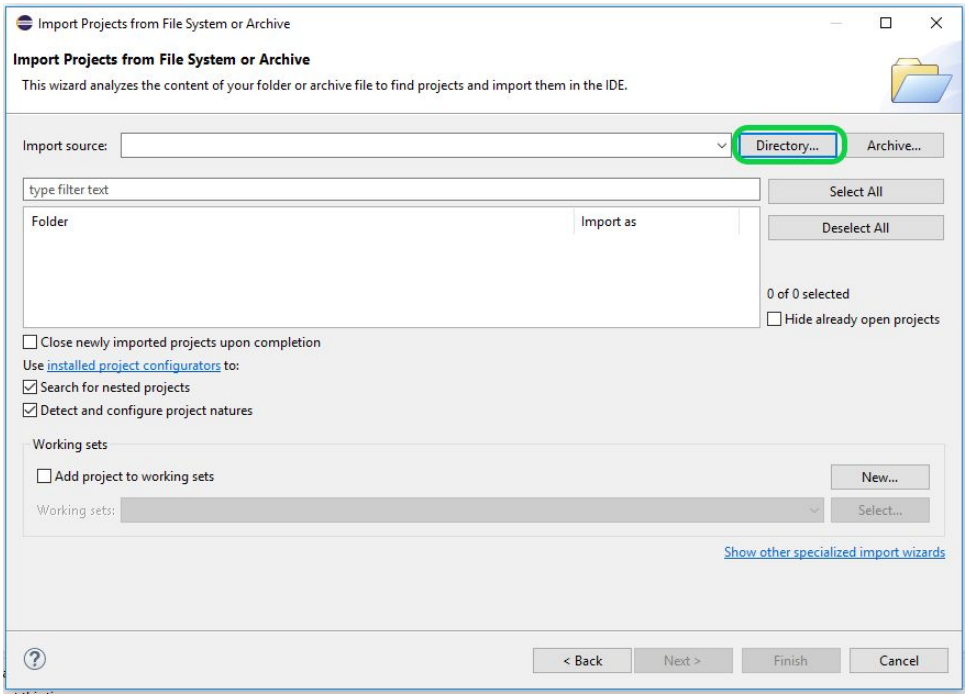
Dentro de Eclipse seleccionamos la opción de importar proyectos.

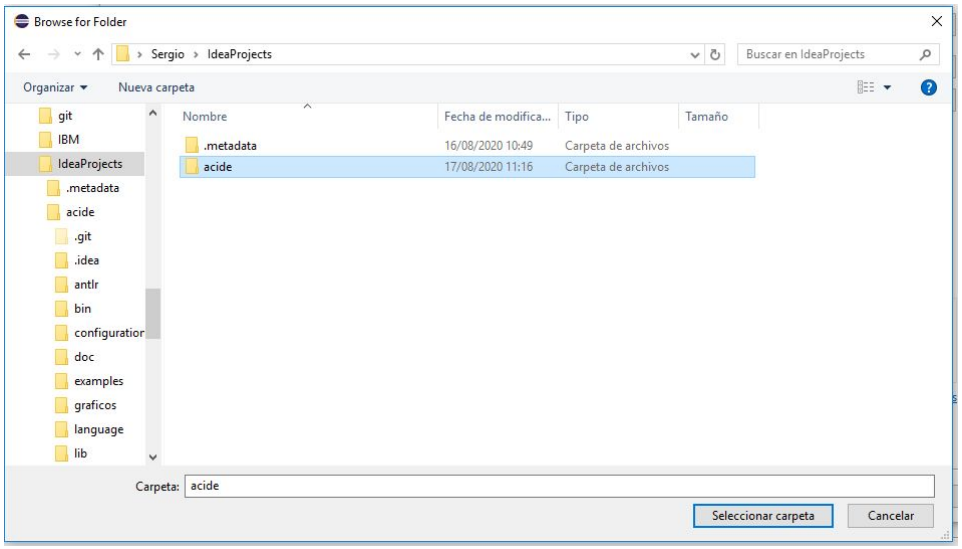


Después tenemos 2 opciones posibles, importar desde archivo existente.

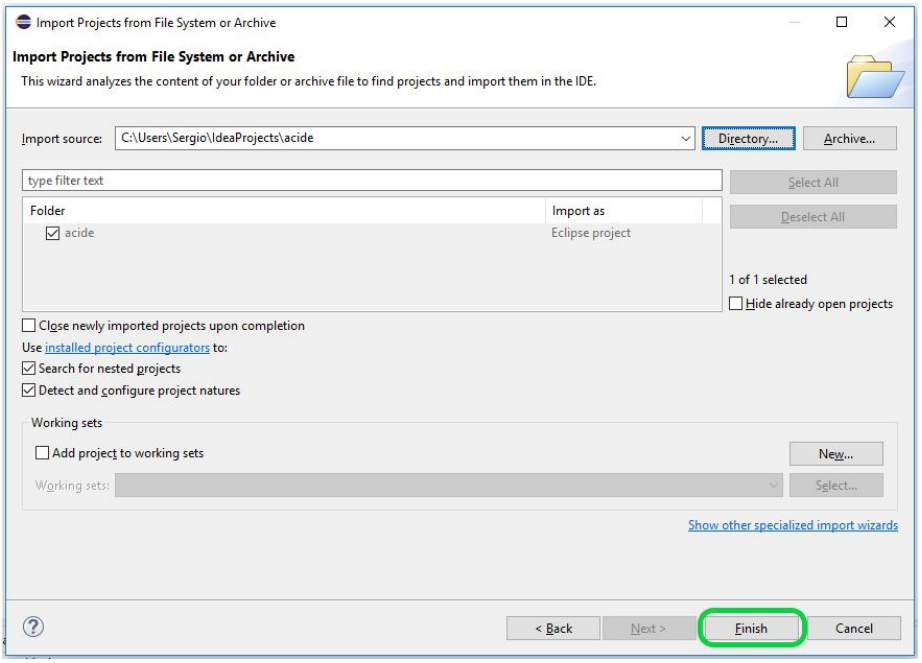


En caso de importar desde un archivo descargado nos encontraremos la siguiente imagen y pulsamos en directorio para seleccionar la raíz del archivo.

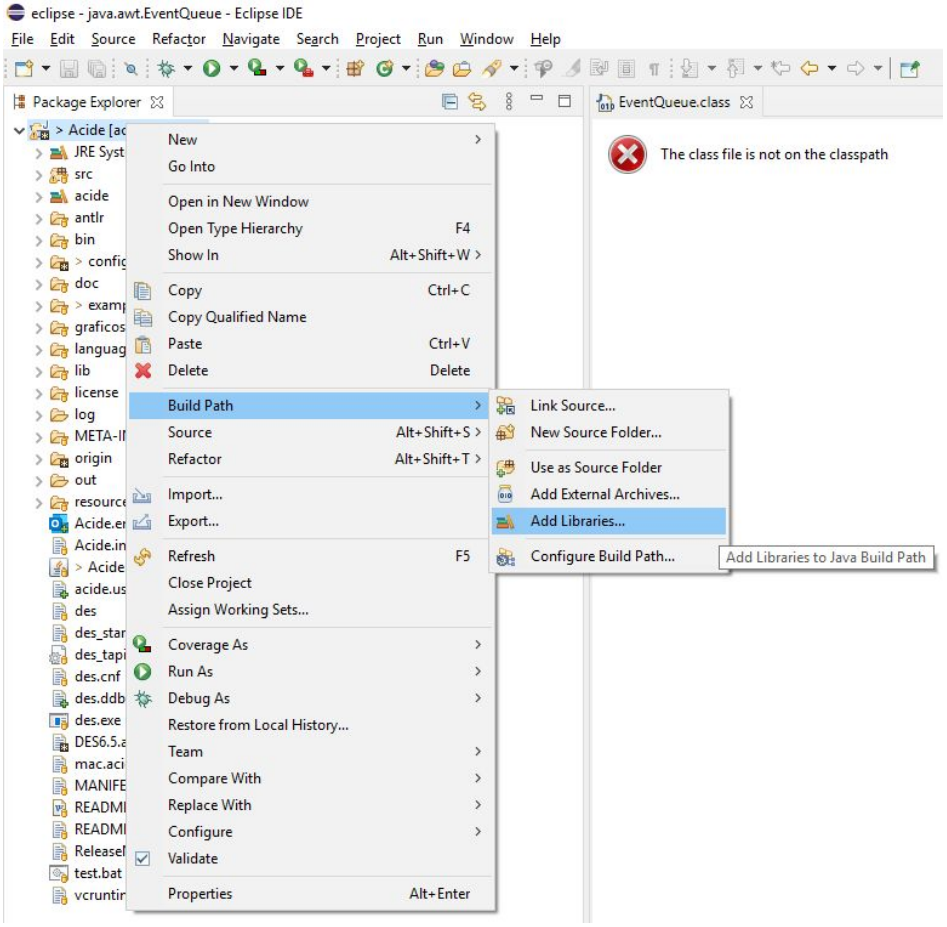




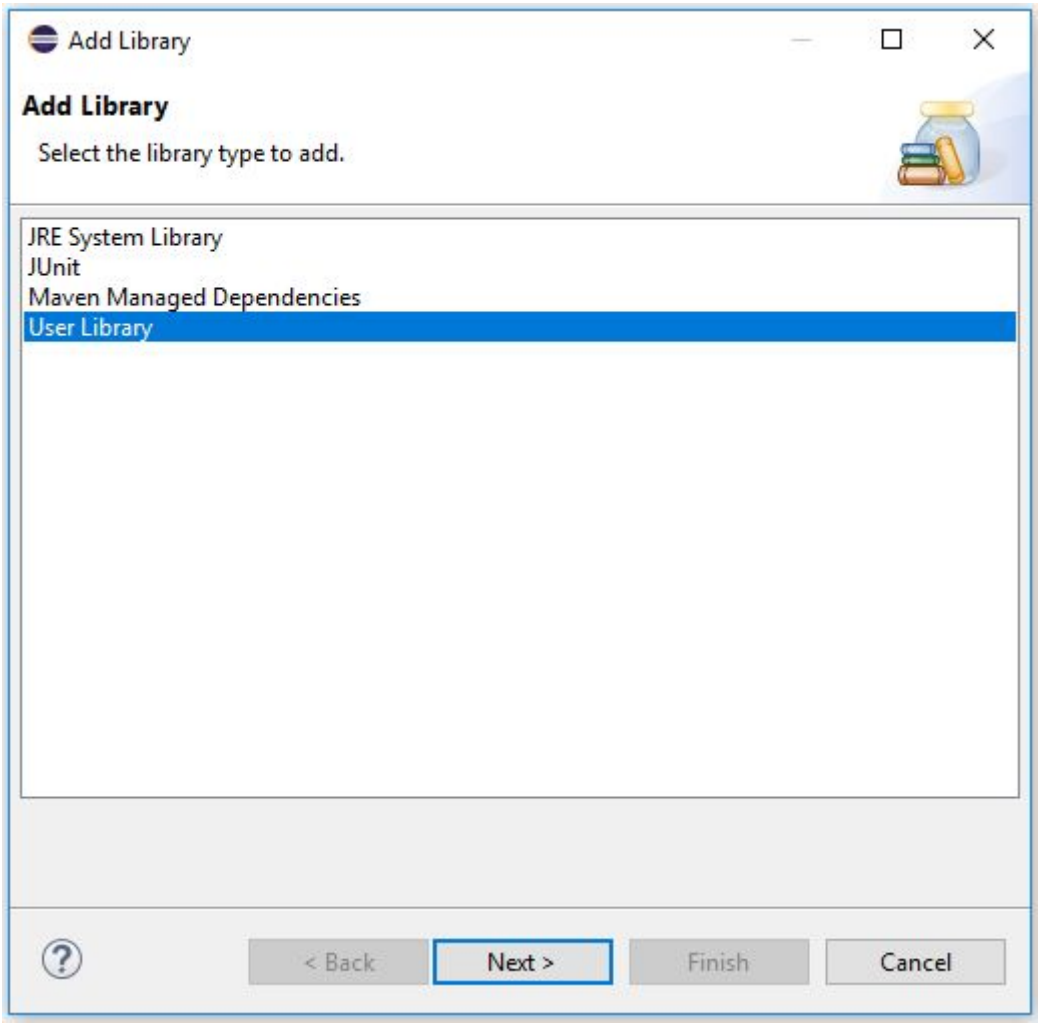
Y pulsamos en finalizar.

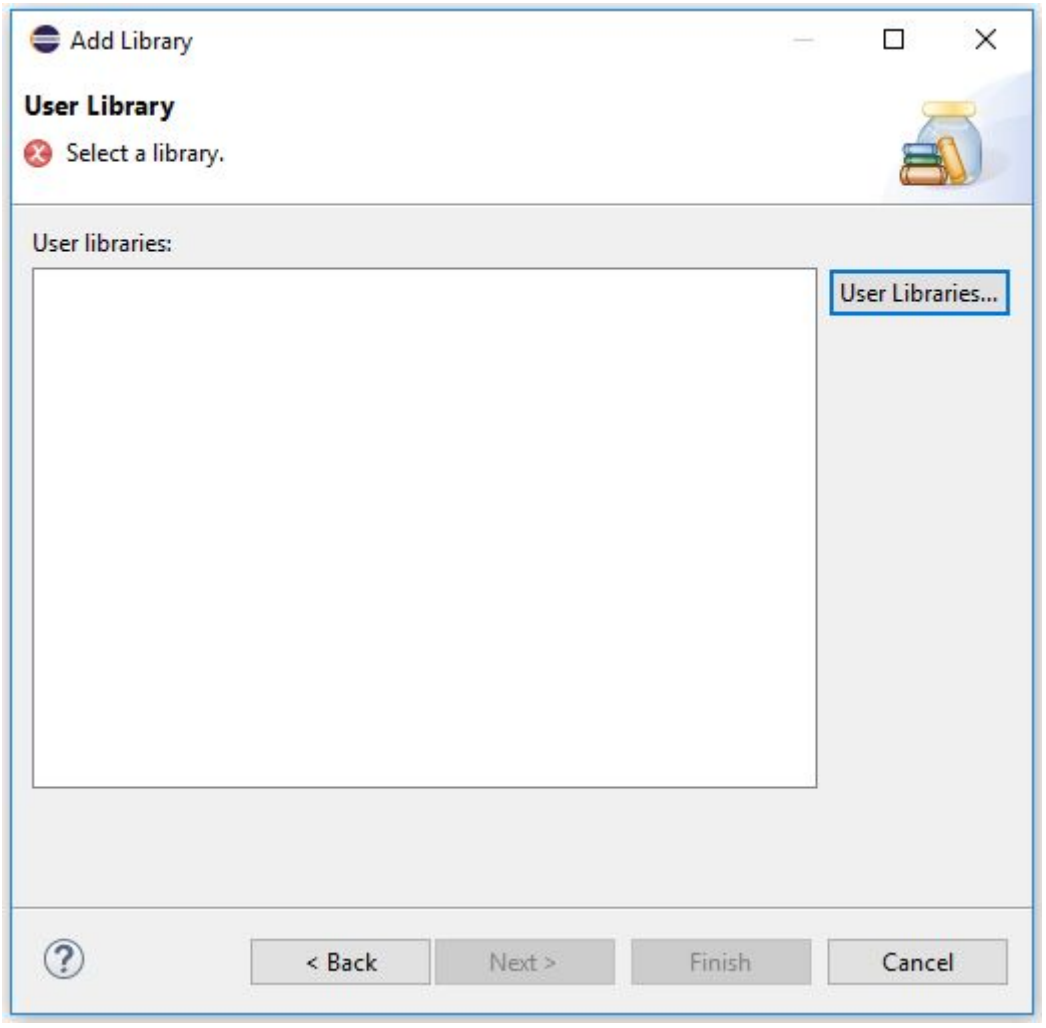


Añadimos las librerías de ACIDE a 'Build Path':

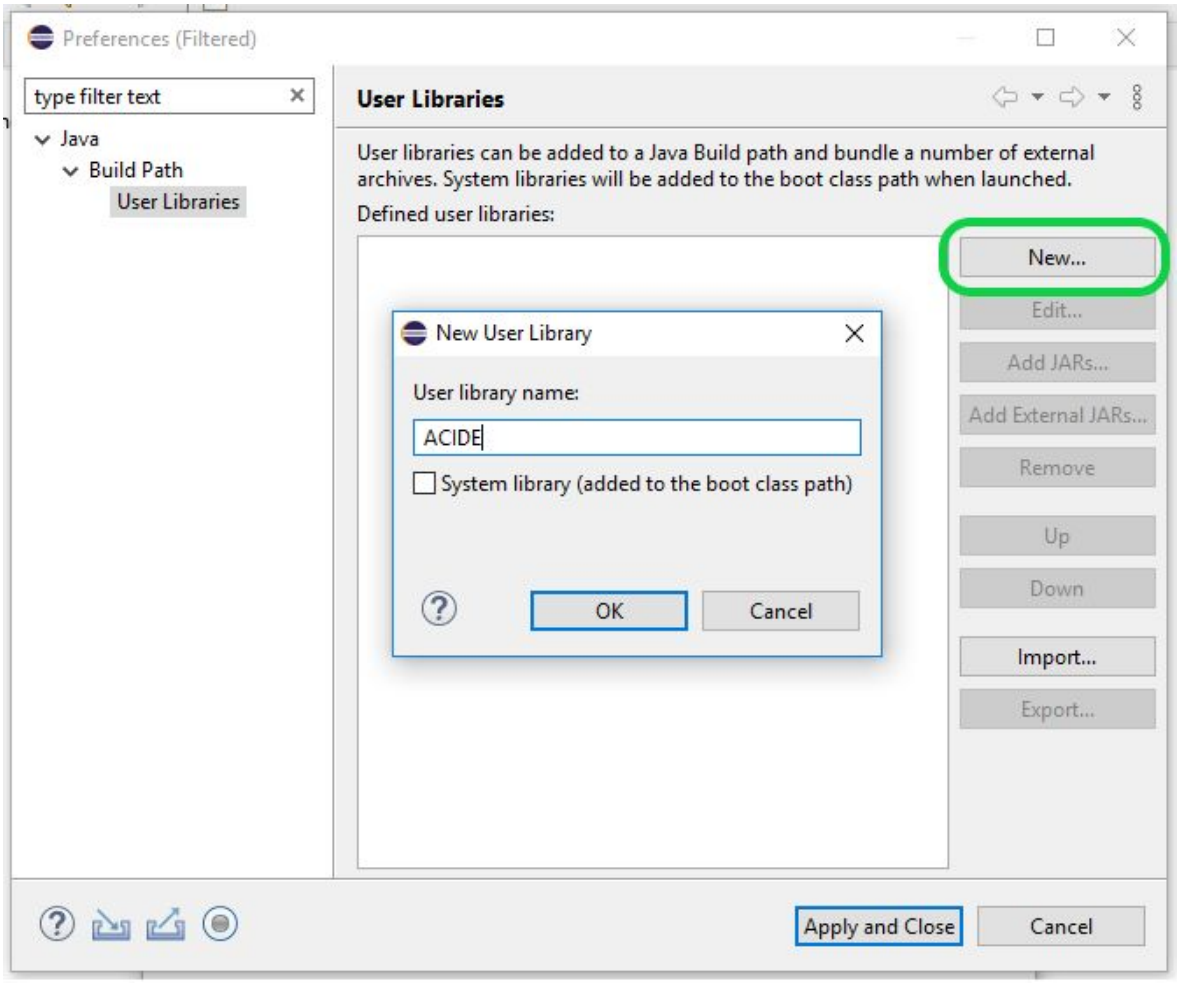


Creamos una nueva librería de usuario:

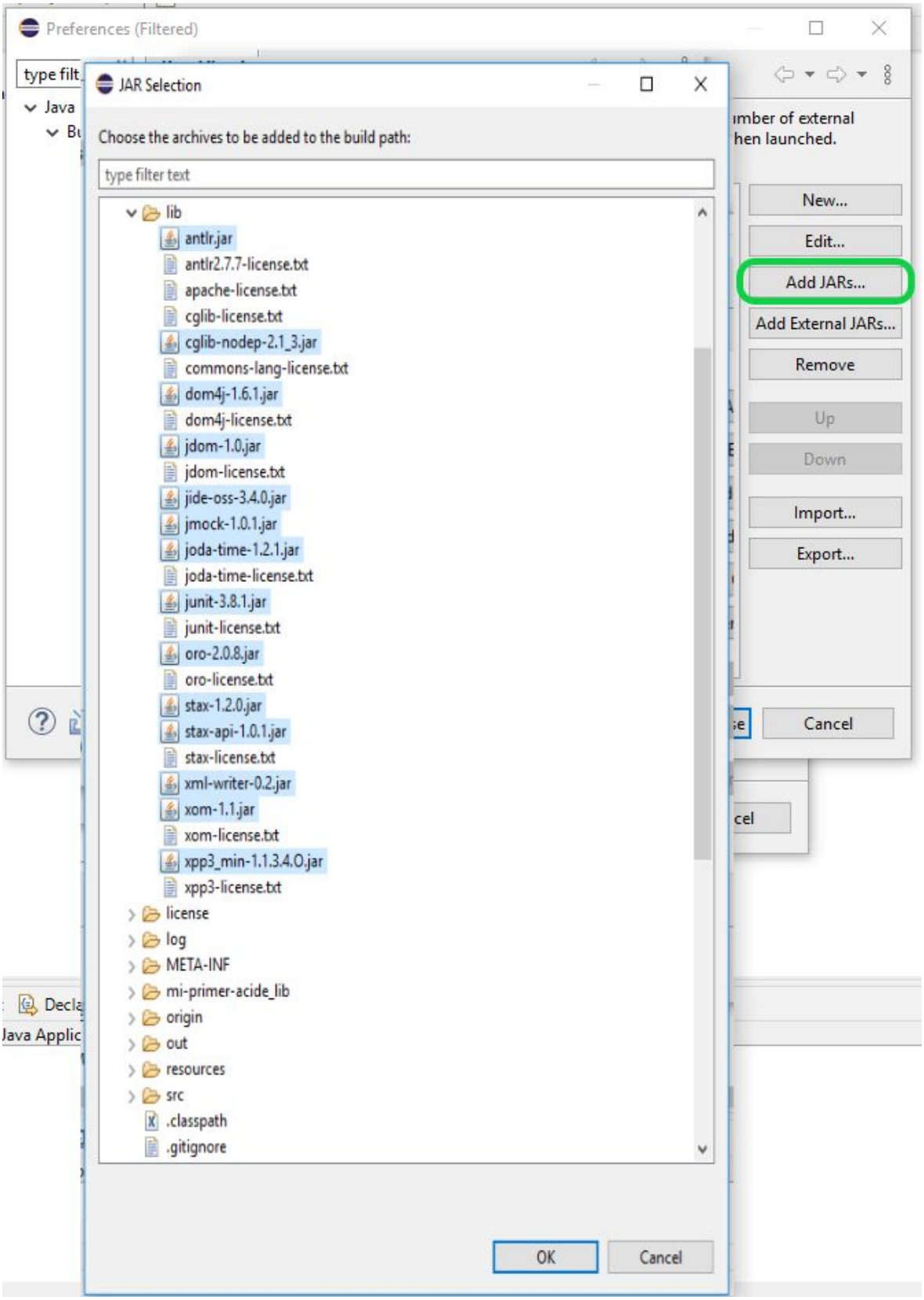

Add Library
Add Library
Select the library type to add.
JRE System Library
JUnit
Maven Managed Dependencies
User Library
< Back
Next >
Finish
Cancel



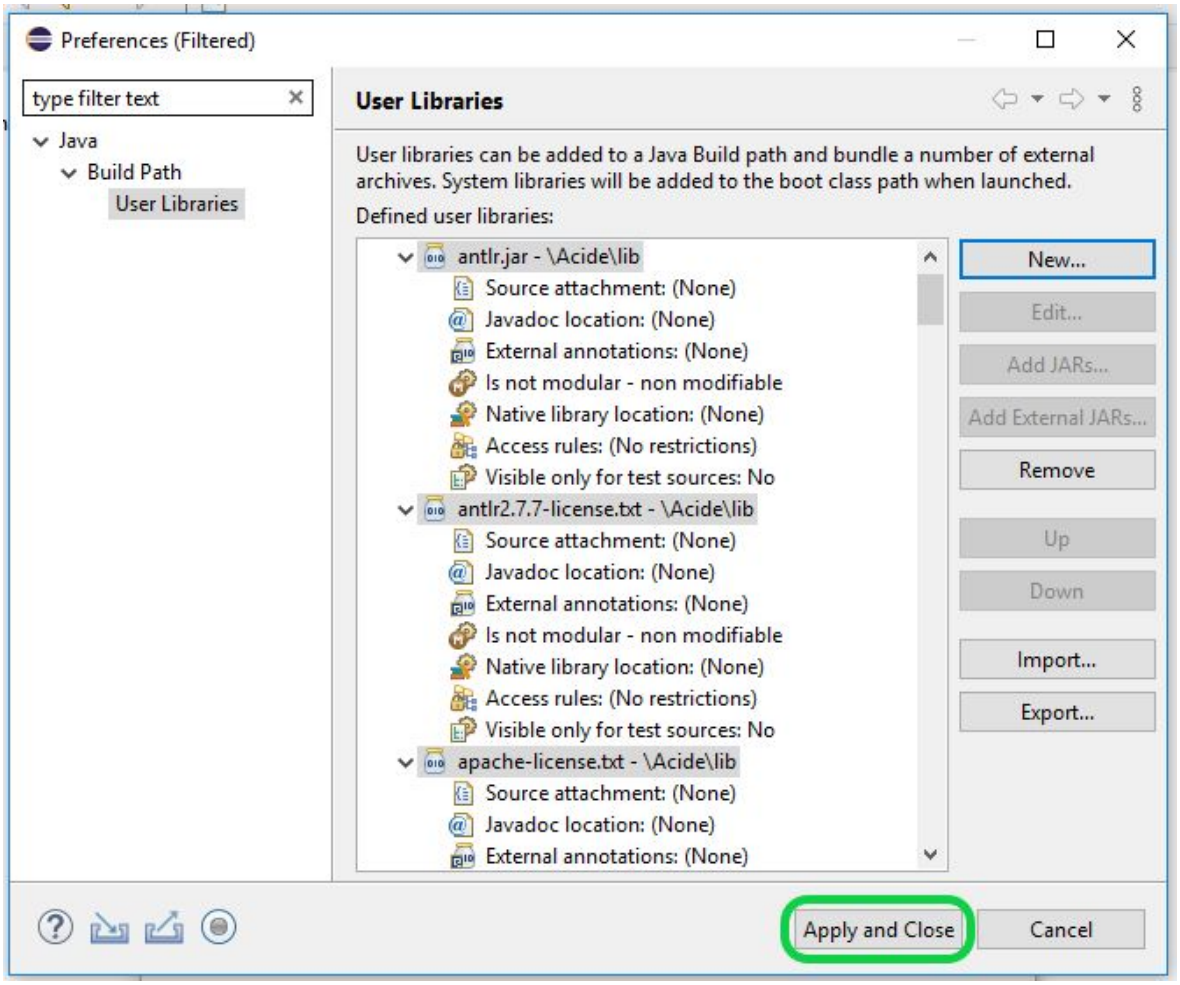
Es recomendable asignarle el nombre de ACIDE a dicha librería:



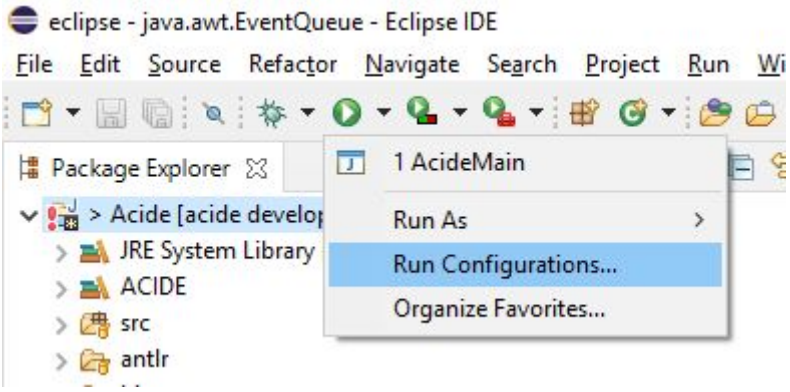
Añadimos **únicamente los archivos .jar** que están dentro de la carpeta lib:



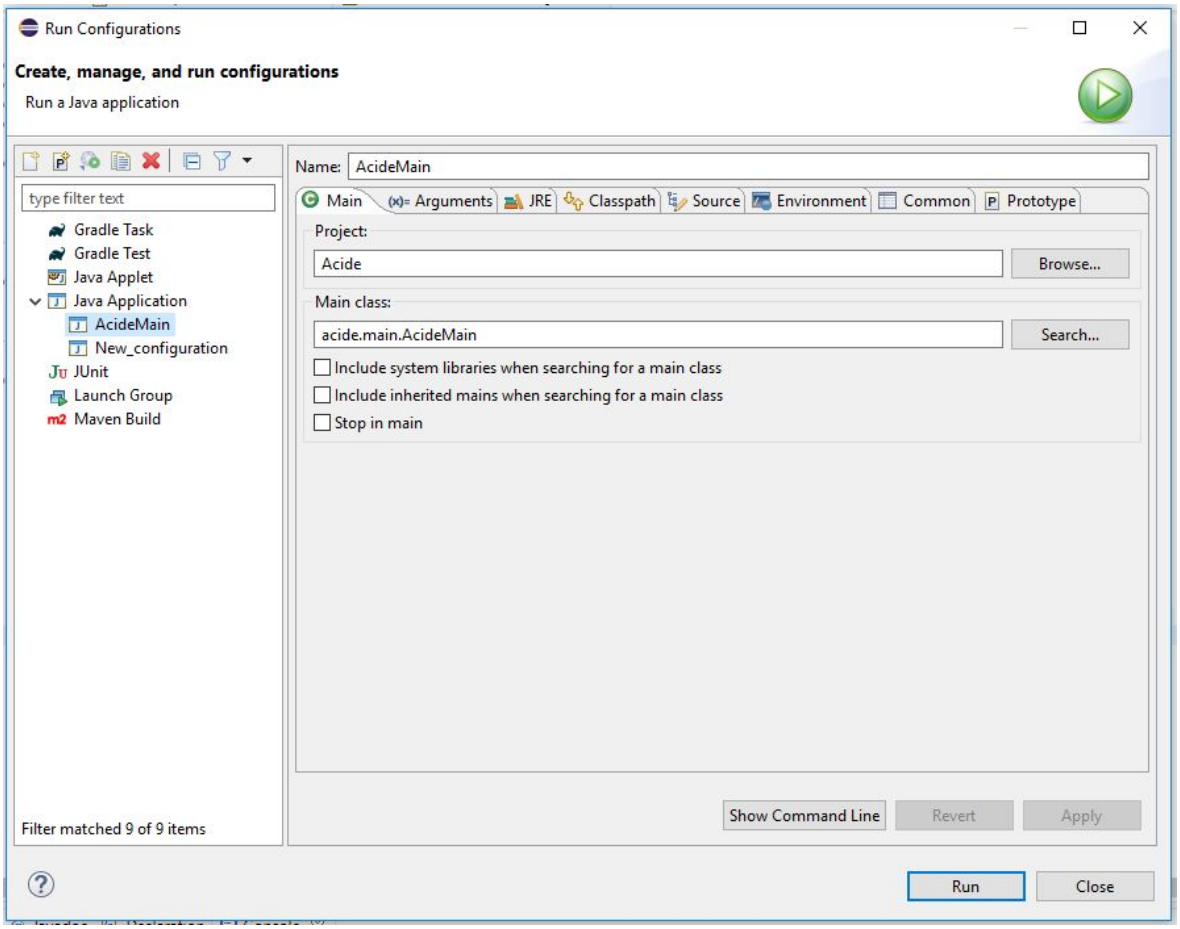
Aplicamos los cambios:



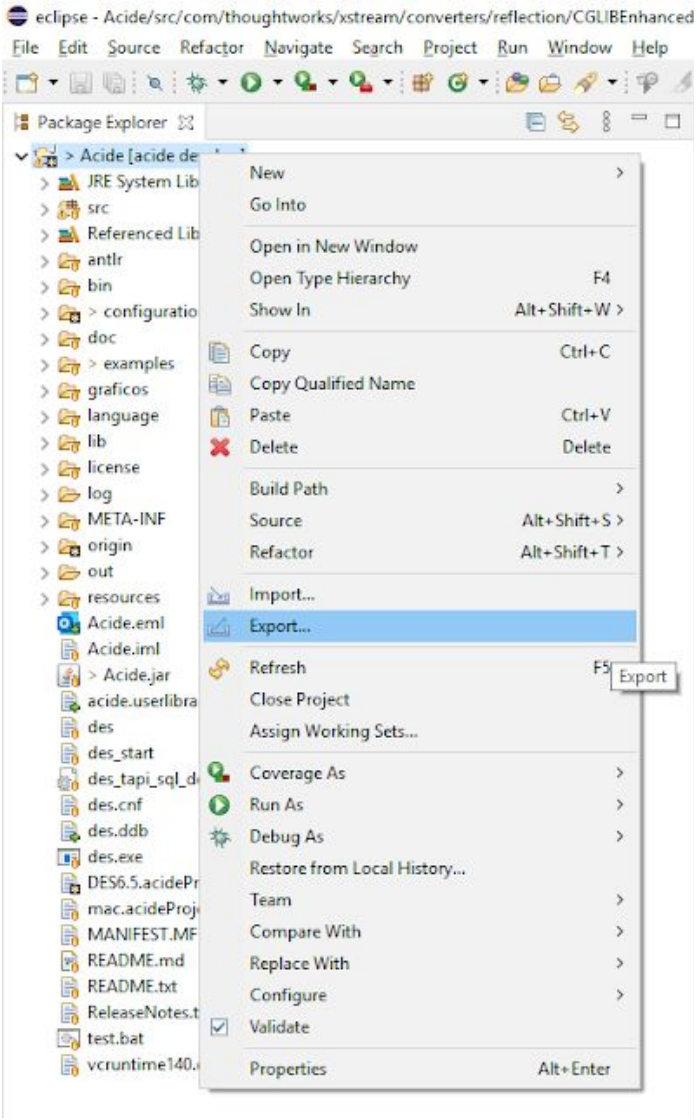
Ahora tenemos que añadir un nuevo perfil de ejecución, en el botón de iniciar pulsamos en el desplegable y seleccionamos la opción de 'Run Configurations':



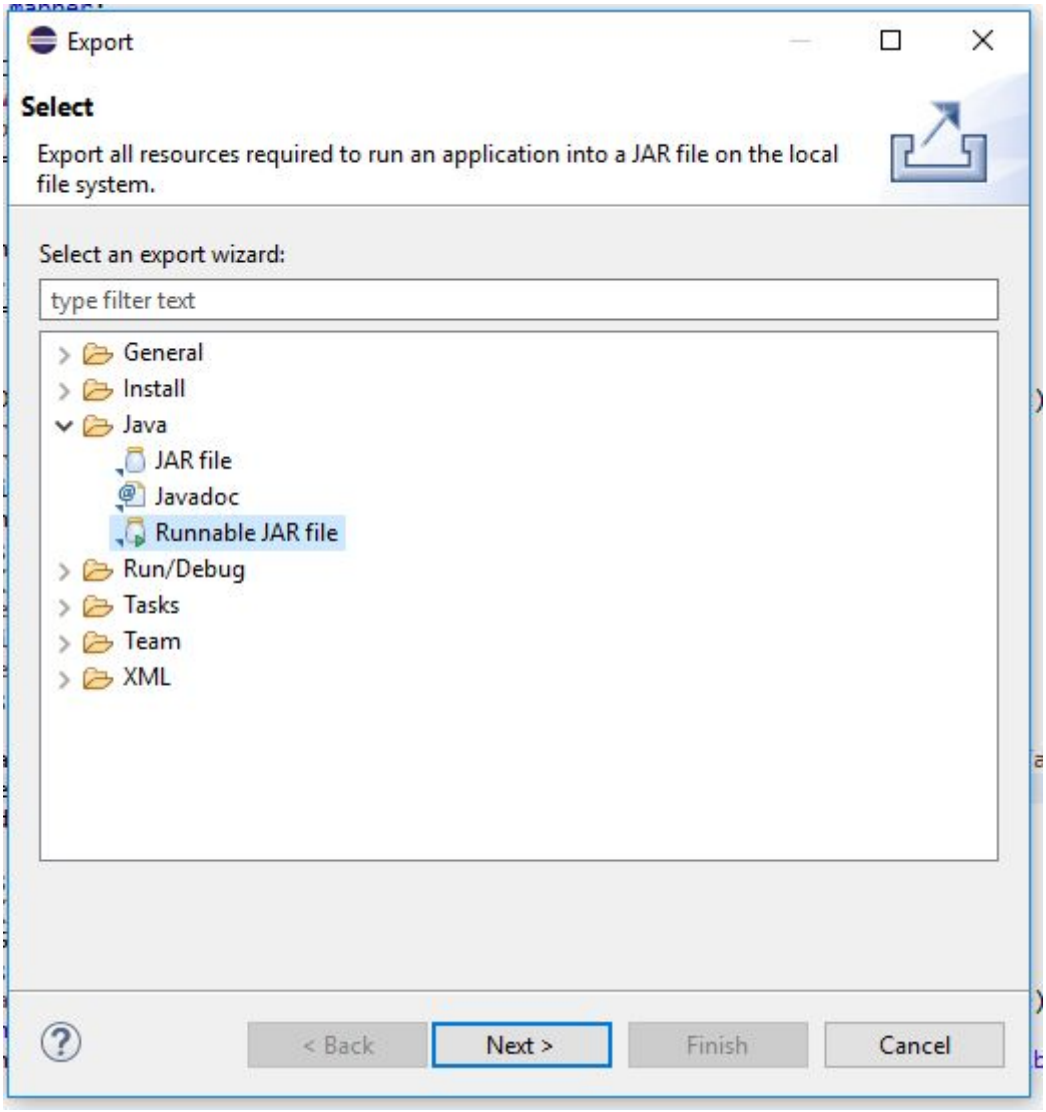
Añadimos una nueva configuración dentro de 'Java Application' y seleccionamos como la clase principal 'AcideMain':



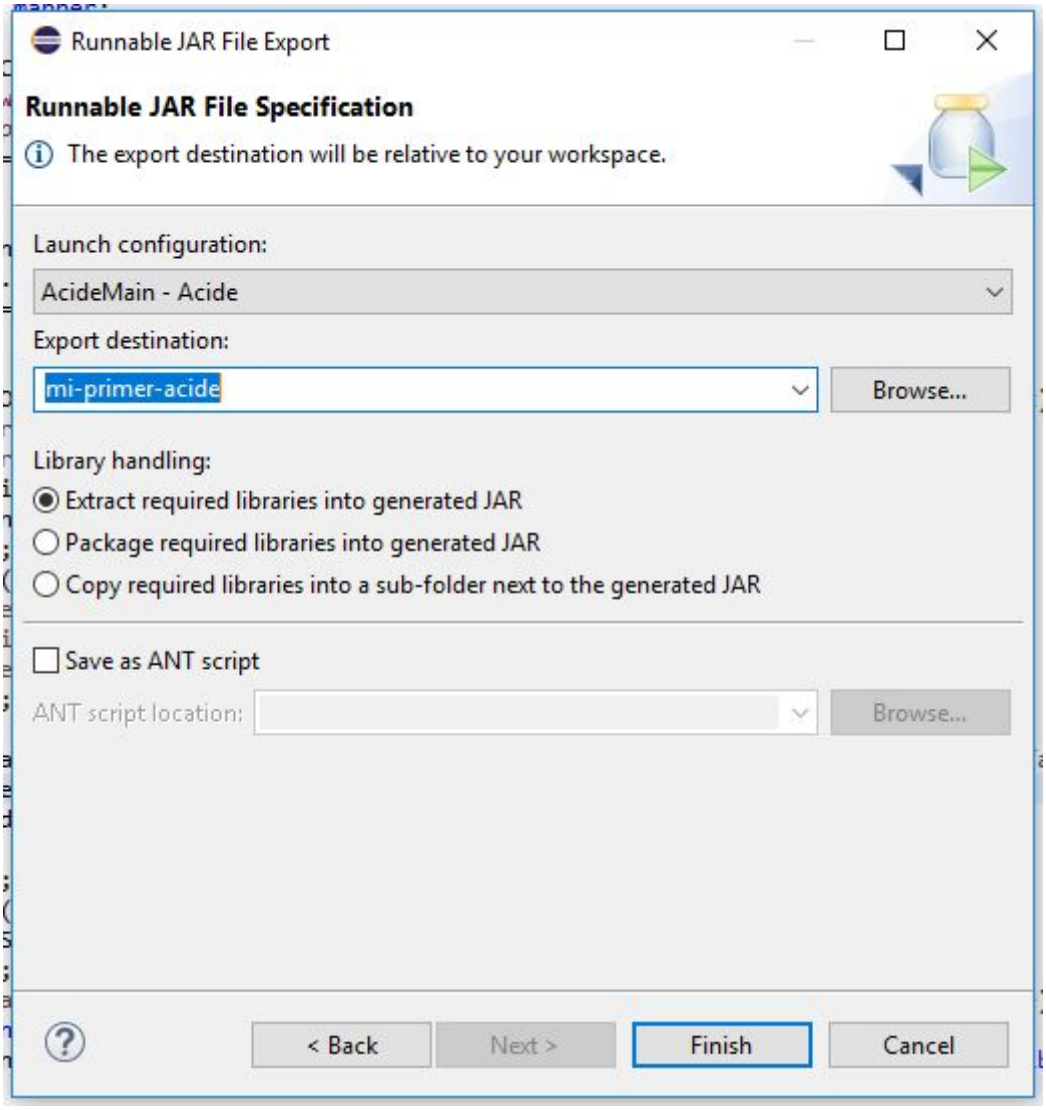
Ahora ya somos capaces de correr la aplicación, lo único que nos queda es generar nuestros ejecutables, para ello hacemos clic derecho en el proyecto y seleccionamos la opción de exportar:



Una vez hayamos seleccionado la opción pulsaremos la opción 'Runnable JAR file' dentro de la carpeta 'Java'.



Después seleccionaremos configuración de ejecución generada anteriormente y damos nombre a nuestro ejecutable y nos aseguramos de que esté en la raíz del proyecto puesto que para ejecutarse necesita los archivos de configuración.



Con esto ya tenemos todas las opciones que necesitamos como desarrolladores.