Actas de las Jenui, vol. 4. 2019.
Páginas: 95-102

DESweb: una herramienta para el aprendizaje de SQL*

Fernando Sáenz Pérez
Departamento de Ingeniería del Software e Inteligencia Artificial
Universidad Complutense de Madrid
28040 Madrid
fernan@ucm.es

Resumen

En este artículo se presenta la herramienta *on-line* DESweb para el aprendizaje del lenguaje de bases de datos SQL. Aunque en la enseñanza de la materia *Bases de datos* se emplean habitualmente sistemas gestores de bases de datos propietarios o de código abierto, resulta llamativa la limitada realimentación que proporcionan ante consultas SQL incorrectas. Sin embargo, no solo estamos interesados en aportar un mejor informe de errores sintácticos, sino también en alertar de los posibles errores semánticos. Un error semántico se produce en una instrucción sintácticamente correcta pero en cuyos resultados faltan o sobran ciertas tuplas con respecto a la interpretación pretendida por el programador. Este trabajo aborda estos aspectos de un sistema interactivo disponible a cualquier usuario, de código abierto, gratuito, y que en particular se está aplicando actualmente en distintos grupos de la asignatura *Bases de datos*.

Abstract

This paper presents the on-line tool DESweb for learning the SQL database language. Though several database management systems, either proprietary or open-source, are used in teaching *Databases*, they provide limited feedback for incorrect queries. However, we are not only interested in providing better syntactic error messages, but also semantic error warnings. A semantic error is found when a syntactically-correct statement includes extra or missing tuples with respect to its intended meaning for the programmer. This work addresses these issues in an on-line, free, open-source, and widely-available system, which is currently being applied to several groups of the *Databases* course.

Palabras clave

SQL, Bases de datos, Errores sintácticos, Errores semánticos

1. Introducción

"For better or worse, SQL is intergalactic dataspeak" [18]

Puede que esta cita no haya perdido todo su sentido desde que se enunció hace casi 30 años. En efecto, tecnologías posteriores tales como *Big Data*, NoSQL, almacenes RDF y las bases de datos XML no parecen haber desplazado al modelo relacional y sus lenguajes [3], sino más bien lo han complementado como alternativa en escenarios específicos (bases de datos de solo lectura, respuestas aproximadas en grandes bases de datos, fuentes de datos heterogéneas...). El examen de propuestas curriculares académicas como [1] permite revelar su importancia en los planes de estudio académicos. No solo las distintas universidades adoptan en sus planes estos contenidos, sino que grandes empresas como IBM, Oracle y Microsoft continúan el desarrollo y soporte de estas tecnologías.

Sin embargo, estas mismas empresas prestan una atención limitada a los usuarios programadores de SQL, proporcionando en general una realimentación débil frente a consultas incorrectas. Por ejemplo, en MySQL: *"You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax"*. También en Oracle: *"ORA-00942: table or view does not exist"* (no se indica cuál). Frente a estos mensajes, los alumnos no pueden evitar sentirse muy desatendidos por un sistema que podría proporcionarles un poco más de ayuda para localizar su error.

Nuestra intención es, pues, proporcionar a los alumnos una herramienta que les ayude en el proceso de aprendizaje del lenguaje SQL. Para ello hemos abordado varios frentes: en primer lugar, generando informes detallados de errores sintácticos, indicando no solo hasta dónde ha sido posible efectuar el análisis sin-

*Agradecemos la labor de los revisores para la mejora del artículo y el soporte proporcionado por los proyectos TIN2017-86217-R (CAVI-ART-2) del ministerio, P2018/TCS-4339 (BLOQUES-CM) de la Comunidad de Madrid y la Unión Europea e Innova-Docentia 295 de la UCM.

táctico, sino también las posibles alternativas que se esperarían y no se encuentran. En segundo lugar, analizando semánticamente las consultas sintácticamente correctas y avisando de posibles errores, como sería el caso, por ejemplo, de condiciones que se pueden demostrar falsas, ciertas o simplificables para *cualquier instancia* de la base datos. Finalmente, proporcionando un sistema *on-line* que reduzca en lo posible el coste del primer y subsiguientes accesos, evitando cualquier instalación y favoreciendo el uso de un navegador web actual sobre cualquier dispositivo (desde portátiles hasta teléfonos móviles). Por lo que sabemos, este trabajo es el primero en incorporar este nivel de descripción de errores en un sistema práctico de bases de datos.

Para conseguir este objetivo partimos del sistema DES [16] (`des.fdi.ucm.net`) (presentado en estas jornadas [15]), que ha evolucionado en gran medida desde entonces. Aunque originariamente fue concebido como un sistema para la enseñanza de bases de datos deductivas, en la actualidad incorpora muchas extensiones; en particular el lenguaje SQL para acceder a relaciones compartidas con el resto de lenguajes a los que da soporte (Datalog, cálculos relacionales TRC y DRC, y álgebra relacional). DES es un sistema muy usado internacionalmente (más de 76K descargas) en enseñanza (hay noticias de hasta 50 universidades) e investigación (`des.fdi.ucm.es/facts`). DESweb (`desweb.fdi.ucm.es`) es la interfaz web a DES, pensada para facilitar su uso y ampliar su ámbito.

El objeto de este artículo es presentar los resultados del esfuerzo realizado para elaborar un sistema con los objetivos enunciados, así como las primeras experiencias de su aplicación en clase con el deseo de que pueda ser útil y aplicable en otras facultades. Para ello, la sección 2 introduce el análisis sintáctico y semántico que se propone. La sección 3 incluye varios ejemplos sencillos ejecutados con el sistema bosquejado en la sección 4. La sección 5 presenta los primeros resultados de evaluación e impacto. La sección 6 recoge trabajos relacionados y la sección 7 concluye este artículo.

2. Gestor de errores

El sistema gestor de errores del análisis de sentencias SQL se aplica en dos fases: el análisis sintáctico seguido del semántico si el primero ha tenido éxito.

2.1. Análisis sintáctico

Como se ha indicado, DES ofrece soporte a varios lenguajes para el acceso a una base de datos compartida por todos ellos. A menos que se trate de un comando, un comentario o se fuerce un lenguaje en concreto, las entradas escritas en el *prompt* del sistema se analizan como si estuviesen escritas en Datalog. Si el análisis sintáctico no tuviese éxito, se probaría después con el analizador de SQL, al que seguirían de igual modo los del álgebra relacional y cálculos relacionales. Si no ha sido posible obtener un análisis sintáctico válido en ninguno de los casos, se emite el informe de error correspondiente al primero encontrado, que puede corresponder a un solo lenguaje o a varios. Por ejemplo:

```
DES> SELECT ,
Error: (DL) Invalid atom or (SQL)
Invalid SELECT list or (SQL) Expected
valid expression or (RA) Expected
valid expression near 'SELECT '
```

El informe de error indica que la entrada no es un átomo válido de Datalog (`DL`), que tampoco es una instrucción SQL (`SQL`) válida porque ni es una lista de proyección ni tampoco una expresión, y que tampoco es una expresión de filtro de la selección del álgebra relacional (`RA`). No obstante, para centrarse en un lenguaje en concreto como el que nos ocupa en este artículo, es mejor definir un modo concreto de entrada con un comando (en particular, con `/sql` para SQL).

La postura de DES frente a otros errores como el indicado en la introducción se puede ejemplificar así:

```
DES> SELECT * FROM empleado
Error: Unknown table or view 'empleado'.
Info: Possible relation (respect case):
      [empleados]
```

No solo se indica cuál es la relación que no encuentra, sino también cuáles son las posibles alternativas (aplicando reglas de encaje de patrones con las relaciones existentes).

2.2. Análisis semántico

En lugar de determinar la validez semántica de consultas con respecto a instancias de referencia proporcionadas por los expertos (como WebSQL, SQLator y varias otras; véase la sección 6), nuestro enfoque se encuentra en la línea de `sqllint` [4]. En concreto, se realiza un análisis semántico verificando ciertas propiedades en tiempo de compilación y de forma independiente de tales instancias. Por ejemplo, la condición `a>b AND b>c AND c>a` se evalúa a `false` en cualquier escenario. En un caso así, DES devuelve:

```
Warning: [Sem] Inconsistent condition.
```

Para detectar la inconsistencia de estas condiciones, traducimos la consulta SQL a un programa lógico con restricciones (CLP($\mathcal{X}$), *Constraint Logic Programming* sobre un dominio genérico $\mathcal{X}$ [7]) cuya ejecución permite determinar la inconsistencia de las condiciones en el contexto no solo de la instrucción SQL, sino también considerando las restricciones de las tablas involucradas. El programa CLP($\mathcal{X}$) en general contiene cláusulas con sucesivas restricciones que se van añadiendo al almacén de restricciones según se va

ejecutando. Como dominios de restricciones para representar a los dominios de SQL usamos: $\mathcal{FD}$ (dominio finito sobre los enteros), $\mathcal{R}$ (reales), $\mathcal{Q}$ (racionales) y el dominio de Herbrand $\mathcal{H}$ con las restricciones de desigualdad (dominios lexicográficos). Además se articula la cooperación entre dominios compatibles para aumentar la precisión del proceso deductivo. Este proceso permite también determinar simplificaciones (si una condición admite una simplificación, posiblemente no esté bien formulada) e incluso permite proponer alternativas más concisas a las consultas mediante descompilación (esto último es sujeto de trabajo futuro). La traducción de SQL a CLP aprovecha una etapa intermedia que ya se realiza en DES: la compilación de SQL a Datalog. Esto facilita la generación del programa CLP y el examen de ciertas propiedades involucradas en la identificación de errores semánticos.

Como se indica en [4], el resultado de un análisis semántico como este es un aviso de los *posibles errores* de una consulta. En efecto, la detección de una de estas situaciones no tiene por qué ser necesariamente un error de programación. Por ejemplo, uno de los errores frecuentes de los alumnos es olvidar la condición de correspondencia entre dos relaciones de la cláusula `FROM`. Sin embargo, esto no es un error si lo que se pretende es determinar todas las combinaciones de las tuplas de ambas relaciones.

En [5] se recoge una lista enumerada y bastante exhaustiva de posibles errores semánticos en consultas SQL. De ellos, incorporamos en DES un número importante (sobre todo por su relevancia al ser errores comunes que cometen los alumnos), con implementaciones específicas que en general se benefician de las deducciones derivadas de la ejecución del programa lógico con restricciones. De todos los posibles casos de cada error, indicamos en la siguiente lista (con los mismos códigos de [5]) los que implementamos.

- Error 1: Condición inconsistente (`Inconsistent condition`). Si falla la evaluación del programa CLP, se emite un aviso como se ha indicado en un ejemplo anterior. Obviamente, detectar fallo al imponer una determinada restricción durante la ejecución del programa CLP no significa que esa restricción sea la causante del fallo de la red de restricciones impuesta hasta el momento, pero sí se puede emitir el informe de inconsistencia en el contexto de toda la red.
- Error 2: `DISTINCT` innecesario (`Unnecessary DISTINCT`). Se avisa si se puede determinar que la consulta no devuelve duplicados pero aun así incluye este modificador. Una primera aproximación a otra más general se basa en las claves primarias de las relaciones implicadas en la consulta.
- Error 3: Columna de salida constante (`Constant output column`). Como consecuencia de la evaluación del programa CLP, si alguna variable de salida se liga a un valor constante, significa que la consulta siempre devolverá esa constante, lo cual es síntoma de una mala formulación.
- Error 4: Columnas duplicadas (`Duplicated column values`). Si dos o más columnas de salida del programa CLP se asignan a la misma variable lógica, significa que contendrán los mismos valores.
- Error 5: Variable de tupla sin usar (`Unused tuple variable`). Se trata de una relación que aparece en la lista `FROM` pero a la que no se accede en la consulta raíz (el error 27 captura el resto de casos para relaciones que no se usan).
- Error 6: Reunión innecesaria (`Unnecessary join`). Se avisa si no se usa ninguna columna de una relación que aparezca en una reunión de relaciones ligadas por integridad referencial.
- Error 7: Variables de tupla idénticas (`Tuple variables are always identical`). Se avisa si dos o más relaciones producen las mismas tuplas, lo cual se puede comprobar en el programa CLP detectando si se produce más de una vez la misma llamada a un predicado con las mismas ligaduras de variables.
- Error 8: Condición implicada o tautológica (`Implied or tautological condition`). Una condición que se puede demostrar trivialmente cierta. Se comprueba deduciendo el fallo del complemento de la condición usando la misma técnica aplicada en el error 1.
- Error 9: Comparación con `NULL` (`Comparison with NULL`). Usar un operador de comparación (`=`, `<>`, ...) con el valor `NULL` siempre devolverá un valor lógico falso. Su detección se reduce simplemente a examinar el árbol sintáctico de SQL.
- Error 11: Operador de comparación genérico innecesario (`Unnecessary general comparison operator`). Se avisa si aparece `LIKE '%'` en alguna condición, lo cual es equivalente a `IS NOT NULL`. Para su detección se emplea el mismo procedimiento del error anterior.
- Error 12: `LIKE` sin comodines (`LIKE without wildcards`). De nuevo, este error se detecta por el examen del árbol sintáctico de SQL.
- Error 13: `SELECT` innecesariamente complicada en una subconsulta `EXISTS` (`Unnecessarily complicated SELECT in EXISTS subquery`). Se detectan usos diferentes de `SELECT *` como raíz de la subconsulta existencial examinando el árbol sintáctico de SQL.
- Error 16: Modificador `DISTINCT` innecesario en una función de agregación (`Unnecessary DISTINCT in aggregation function`). Se avisa si se usa `MIN` o `MAX` con un argumento

modificado con DISTINCT, o si se usa cualquier otra función de agregación con DISTINCT sobre columnas clave. En ambos casos se examina la traducción a Datalog para su determinación.

- Error 17: Argumento innecesario en COUNT (Unnecessary argument of COUNT). Usando los metadatos, avisa si se aplica COUNT a un argumento que no puede ser nulo debido a una clave.
- Error 27: Condición de reunión ausente (Missing join condition). Avisa si al menos dos de las relaciones de una cláusula FROM no se reúnen bajo ningún criterio. Esto incluye al error 5 (variable de tupla sin usar).
- Error 32: Cláusula HAVING extraña (Strange HAVING. Avisa si una consulta con esta cláusula HAVING no incluye una cláusula GROUP BY, por lo que esta condición se aplicaría a una única tupla resultado de la agrupación.
- Error 33: SUM(DISTINCT ...) o AVG(DISTINCT ...). Avisa si se incluye descarte de duplicados para el argumento de SUM o de AVG. El descarte es sospechoso porque en general los duplicados son relevantes en estos agregados.

3. Ejemplos de uso

Esta sección incluye algunos ejemplos de uso de la herramienta acompañados con los informes que produce[1]. Como se subraya en [4], estos informes advierten de posibles usos incorrectos de consultas sintácticamente correctas, y no indican necesariamente un error. Por ello, la herramienta emite informes de avisos (*warnings*) sin rechazar la consulta.

El primer ejemplo trata sobre la creación de tablas con restricciones CHECK:

```
CREATE TABLE departments(
dept  VARCHAR(10) PRIMARY KEY,
dname VARCHAR(20),
budget INT CHECK (budget > 0));

CREATE TABLE employees(
ename VARCHAR(20) PRIMARY KEY,
dept  VARCHAR(10) REFERENCES departments,
salary INT CHECK (salary BETWEEN 5000 AND
2000));
Warning: [Sem] Inconsistent condition.
```

Si bien la primera sentencia es correcta, el informe indica para la segunda una condición inconsistente (siempre falsa), por lo que ninguna tupla con un valor de sueldo distinto de nulo podría insertarse en la tabla. El origen del error en este caso puede haber sido desde un error tipográfico hasta usar incorrectamente el orden de los valores. Si en este mismo ejemplo se usase la condición salary >= 2000 OR salary <= 5000, el informe resultante sería: Warning: [Sem] Tautological condition: (salary > 2000 OR salary < 5000), lo cual denota una condición tautológica (siempre cierta) debido al uso incorrecto del operador OR en lugar de AND.

Siguiendo este ejemplo y una vez corregido este error, la herramienta detecta otros en consultas como la siguiente, que trata de devolver los empleados de dos departamentos en concreto:

```
SELECT ename FROM employees WHERE
dept='IT' AND dept='HR';
Warning: [Sem] Inconsistent condition.
answer(employees.ename:varchar(20)) ->
{ }
Info: 0 tuples computed.
```

El origen del error se encuentra de nuevo en un uso inadecuado de la conectiva lógica y, por tanto, no se devuelve ninguna tupla en el resultado.

Otro error habitual que cometen los alumnos es olvidar la condición de correspondencia entre relaciones[2]:

```
SELECT ename, dname FROM employees,
departments WHERE budget=0;
Warning: [Sem] Inconsistent condition.
Warning: [Sem] Missing join condition for
[employees,departments].
```

Nótese cómo el gestor de errores informa de tantos posibles errores como encuentra.

También resulta frecuente que los alumnos escriban consultas con relaciones innecesarias, como en:

```
SELECT ename FROM employees NATURAL INNER
JOIN departments;
Warning: [Sem] Unnecessary join with
"departments". There is a foreign key
relating this with "employees", and
non-key attributes are not accessed.
```

Aunque se avisan de determinados usos innecesarios de DISTINCT como en el siguiente ejemplo, aún hay espacio de mejora para ampliar su detección.

```
SELECT DISTINCT ename FROM employees
WHERE dept='IT' OR dept='HR';
Warning: [Sem] Unnecessary DISTINCT
because of primary key in [employees].
```

A continuación se muestra un ejemplo donde se detecta un uso incorrecto más sutil. Se trata de una tabla de productos industriales gaseosos para los que la suma de porcentajes de sus componentes debe ser 100:

```
CREATE TABLE gas_products( name
VARCHAR(20) PRIMARY KEY, butane FLOAT,
propane FLOAT, olefins FLOAT, diolefins
FLOAT, CHECK (butane BETWEEN 0 AND 100
AND propane BETWEEN 0 AND 100 AND olefins
```

[1] Pese a que por motivos de espacio se ilustran ejemplos muy básicos, la herramienta maneja consultas de la complejidad que se espera en estas enseñanzas, incluyendo subconsultas, correlaciones, operadores [NOT] IN/EXISTS, recursión con WITH, etc.

[2] A partir de aquí se omitirán los resultados de las consultas.

```
AND diolefins BETWEEN 0 AND 100 AND
butane+propane+olefins+diolefins = 100));
```

En la siguiente consulta se detecta que la condición se puede simplificar porque, dadas las restricciones de la tabla y la condición de la consulta, se obtiene un sistema de ecuaciones determinado. Sus soluciones son 45 y 35 para las columnas de butano y propano respectivamente. Así, es un síntoma de un posible error.

```
SELECT butane, propane FROM
gas_products WHERE butane-propane=10
AND butane+propane=80;
Warning: [Sem] Constant output column
"butane" with value "45.0".
Warning: [Sem] Constant output column
"propane" with value "35.0".
```

4. El sistema DESweb

4.1. Interfaz de usuario

La fachada al sistema DESweb permite el acceso de tres modos distintos: como usuario no registrado, usuario normal provisto de identificador y contraseña, y usuario administrador. En este artículo nos centraremos en la interfaz de usuario registrado y dejaremos el resto de detalles disponibles en un apéndice *on-line*[3].

Mientras los usuarios registrados mantienen su sesión y sus archivos en el servidor de forma ilimitada, a los invitados se les asigna una sesión temporal durante 50 minutos (periodo que pueden ampliar a su conveniencia). La interfaz de usuario (figura 1) está dividida en cinco paneles identificados en la figura con números en rojo. El panel de usuario (**1**) permite modificar la contraseña o cerrar la sesión de un usuario.

El panel EDITOR (**2**) muestra el contenido del archivo abierto como en un editor de texto simple. Incluye coloreado sintáctico sensible a la extensión del archivo (`.sql`, `.ra`, ...) para los distintos lenguajes que soporta. También incluye botones habituales para guardar (Save y Save As) y cerrar el archivo abierto (Close). El cambio de tema de colores de la interfaz con el botón Colors se añadió a petición de algunos alumnos que preferían fondos oscuros (como el del tema Enter The Matrix). Finalmente, el botón Run ejecuta los contenidos del panel de edición como si se tratase de un *script* SQL, RA o Datalog (lenguaje determinado a partir de la extensión del archivo).

El panel CONSOLE (**3**) muestra la salida del sistema DES y, como complemento al botón Run, permite la entrada de comandos e instrucciones en el cuadro de texto inferior (debajo del *prompt* DES>). Integra historial de comandos, y la pulsación de la tecla Intro provoca el envío de la entrada al servidor para su procesamiento remoto. Se ha configurado un límite de tiempo para la ejecución de estas entradas (si se sobrepasa, se muestra el mensaje de error `Timeout exceeded`). Junto al título de este panel aparecen una serie de botones: Clear (borra la salida de la consola), Manual (accede al manual completo de DES), Refresh (actualiza los contenidos de la consola), Commit (compromete los cambios que se hayan aplicado sobre la base de datos), Rollback (retrocede los cambios hasta el último punto de salvaguarda, sea este automático o manual) y Restart (reinicia la consola).

El panel DIRECTORY (**4**) contiene la jerarquía de directorios y los archivos de usuario en el sistema de archivos del servidor. Al pulsar el nombre de un archivo, se carga en el editor. También es posible descargar un archivo (Download), eliminarlo (Remove) o subir un archivo local al servidor (Upload).

Finalmente, el panel MESSAGE (**5**) muestra los mensajes que emite el sistema DESweb (no los propios de DES, que se muestran en la propia consola) referidos a las acciones de la interfaz web y los posibles errores que se puedan producir.

Además de esta interfaz de usuario, está disponible la de administrador para la gestión habitual de cuentas de usuario. En particular también permite observar el tamaño de los registros de sesiones, la actividad de los usuarios y estadísticas de uso. El apéndice indicado al principio de esta sección ofrece todos los detalles.

4.2. Descripción del sistema

DES nació de la necesidad de la enseñanza de sistemas de bases de datos deductivas con Datalog, y de ahí que actualmente se hereden ciertas características no habituales en las bases de datos relacionales. Por ejemplo, está orientado a conjuntos (como el álgebra relacional original propuesta por Codd) y, si bien esto es útil para trabajar con este lenguaje formal y los cálculos relacionales, se puede configurar para trabajar con duplicados al usar SQL. Además, su sistema de tipos es estricto, por lo que es posible obtener errores de tipos al mezclar distintos tipos numéricos. No obstante, esto se puede relajar aplicando coherción automática de tipos. Los identificadores de usuario (tablas, columnas, ...) distinguen mayúsculas de minúsculas, aunque las palabras clave de SQL no. Al iniciar sesión en DESweb como usuario normal o invitado, el sistema aparece configurado para trabajar en SQL (con duplicados, coherción automática de tipos y entrada multilínea).

Aunque DES es una base de datos en memoria (*In-memory DB*), se ha configurado para que disponga de almacenamiento persistente. Así, al usarlo a través de DESweb, cualquier modificación del esquema de la base de datos o de sus contenidos se mantiene para posteriores sesiones. Como es natural, cada usuario accede a una instancia propia de la base de datos.

[3] https://www.fdi.ucm.es/profesor/fernan/desweb/manual_es_1.2.pdf

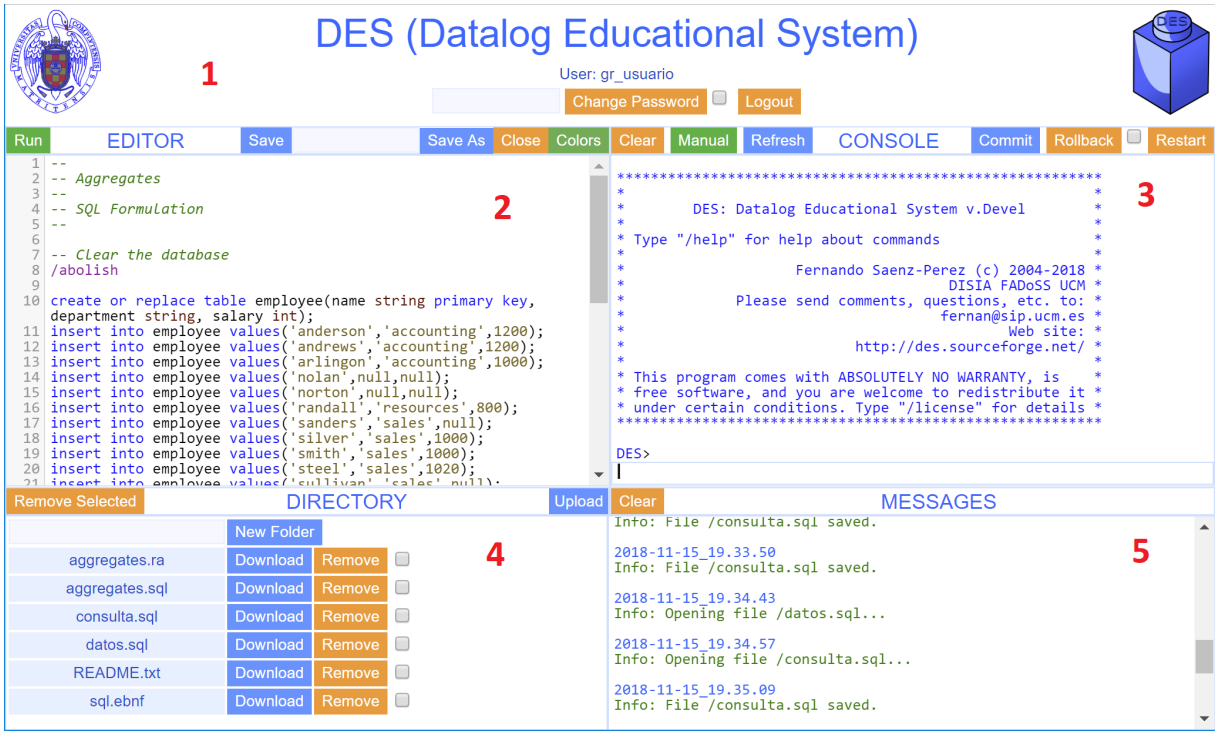

Figura 1: Interfaz de usuario registrado.

DESweb se implementó original y exclusivamente en SWI-Prolog [19], el cual dispone de un conjunto de bibliotecas para el desarrollo web para generar rápida y fácilmente un servidor sin necesidad de otras tecnologías extra como Apache o IIS.

Aunque es posible dejar toda la carga de procesamiento en el servidor (para el que no se prevé un gran número de transacciones por segundo), la experiencia de usuario (acostumbrado a otras interfaces) se ve perjudicada cuando se recarga la página en cada petición. Para mejorar esta experiencia se han ido trasladando al cliente las operaciones más frecuentes con JavaScript.

La generación de páginas HTML es dinámica: se ha aplicado la tecnología *Termerized HTML* (`library(html)`) [11], que permite especificar este código con estructuras de datos Prolog y sin usar código HTML nativo. Este enfoque es más flexible que el uso de HTML estático: por un lado, simplemente con reconsultar los predicados generadores de código se puede actualizar el servidor; por otro lado, permite la generación dinámica de código al vuelo para configurar las páginas de forma paramétrica.

La instalación del servidor requiere cualquier sistema operativo habitual (Ubuntu, Windows, MacOS, ...) al que SWI-Prolog ofrezca soporte. Deben estar instalados SWI-Prolog 7.*x* y algunas herramientas de GNU. El servidor se puede descargar de `desweb.sourceforge.net` e instalarlo siguiendo las instrucciones del archivo `README.txt`. El cliente web solo requiere HTML 5 y JavaScript.

5. Evaluación e impacto

Durante el curso actual estamos llevando a cabo un proyecto de innovación docente (programa InnovaDocencia) en el que se enmarca este trabajo. Usamos la herramienta en varios grupos de la asignatura *Bases de datos* de los grados *Ingeniería Informática* e *Ingeniería del Software* y los dobles grados *Ingeniería Informática - Matemáticas* y *Administración y Dirección de Empresas - Ingeniería Informática*. Se trata de una asignatura de carácter obligatorio, de 6 créditos, con un temario introductorio a las bases de datos relacionales que en particular incluye álgebra relacional y SQL (además de diseño, transacciones y disparadores). Para su evaluación se realizan pruebas de desarrollo (controles), prácticas en grupo y examen final. El número de alumnos matriculados en la asignatura es de entre 30 y 70 dependiendo del grupo. Se les han creado más de 200 cuentas de usuario, habiéndose contabilizado más de 600 accesos con cuentas de invitado de un total de más de 3.000 inicios de sesión.

DESweb ha sido usado tanto por el profesor en la exposición de las clases teóricas como por los alumnos en las prácticas cerradas y en la resolución de los ejercicios planteados. En primer lugar, hemos aplicado la herramienta a las prácticas sobre álgebra relacional, aprovechando que el sistema DES ofrece soporte a este lenguaje y se puede configurar para trabajar sobre conjuntos sin duplicados. En segundo lugar, la hemos aplicado a las prácticas de SQL (objeto principal del

proyecto de innovación) activando los duplicados y la coherción de tipos. Las prácticas planteadas al usar la herramienta no han diferido en esencia de las de cursos anteriores. De hecho, al ser una asignatura coordinada entre varios departamentos y con múltiples grupos, cada profesor ha sido libre de usarla o no en sus clases.

Como beneficio del uso de la herramienta en las clases teóricas, observamos que ayuda a captar la atención de los alumnos (apreciable por su lenguaje corporal e intervenciones). Además, acostumbrados a un mundo muy interactivo, los alumnos aprecian la realimentación que produce la herramienta cuando les explicamos los errores comunes que cometen. Ejemplos de estos errores son olvidar la condición de correspondencia entre relaciones (además del aviso semántico, pueden ver la gran cantidad de tuplas en la respuesta) y añadir descarte de duplicados cuando no es necesario (lo que implica degradación del rendimiento). Todos estos avisos aparecen resaltados en color en la herramienta y animamos a los alumnos a que los tengan en cuenta.

Por tanto, el beneficio obvio que esperamos cuando los alumnos usen la herramienta es que aprovechen esta realimentación de errores (tanto sintácticos como semánticos) para que sean conscientes de ellos antes de entregar prácticas y ejercicios. Aunque algunos aprovechan esto en las prácticas cerradas, también es cierto que observamos ciertos problemas. Otros alumnos prefieren preguntar al profesor en lugar de examinar los informes de errores, lo que quizás evidencie una cierta pereza. También parecen tener dificultades al interpretar estos informes (tanto de DESweb como de intérpretes de consultas SQL como Oracle). Un posible motivo es que estén en inglés, por lo que una mejora de la herramienta sería su localización a distintos idiomas.

A lo largo de este primer cuatrimestre se ha ido mejorando la herramienta para adaptarla a los alumnos (cambiando la interfaz de usuario, ampliando la funcionalidad del lenguaje SQL soportado y corrigiendo errores), por lo que su realimentación ha sido muy valiosa en este aspecto. Si bien la etapa de paso a producción debería haber sido sencilla, hemos encontrado inconvenientes en el entorno institucional (reglas de acceso en la capa de seguridad, dificultad en conseguir el certificado digital y configuración del entorno de virtualización, entre otros), lo que ha provocado en ocasiones inconvenientes en el acceso al servidor.

De momento hemos recibido realimentación directa de los alumnos en las prácticas supervisadas y a través tanto del campus virtual como del correo electrónico. La impresión general (que intentaremos perfilar a finales de las asignaturas –actualmente en curso como la de segundo cuatrimestre– con cuestionarios) es de buena acogida. Un motivo obvio ha sido la inmediatez de acceso al sistema[4], sobre todo comparándolo con el otro sistema que hemos usado: SQLDeveloper sobre Oracle, cuyo acceso estaba limitado a los puestos de laboratorio. Si un alumno quería hacer las prácticas de Oracle fuera del laboratorio, debía instalarse el servidor y el cliente (más de uno necesitó ayuda para ello a pesar de las instrucciones que les proporcionamos). Por otro lado, comentaban las dificultades para escribir consultas y procedimientos sintácticamente correctos debido a los mensajes de error que recibían[5].

Otra fuente muy valiosa para evaluar la utilidad del sistema es el examen de los archivos de registro de sesión (*session logs*) que se almacenan automáticamente. Estos archivos contienen para cada usuario el volcado de su interacción con el sistema en la consola. En ellos se pueden observar las reacciones de los alumnos al arreglar sus consultas en respuesta a los mensajes de error emitidos por el sistema. Por tanto, esto ofrecerá una indicación de la utilidad de los informes de error emitidos en los futuros análisis que realicemos.

6. Trabajos relacionados

Existen múltiples herramientas para el aprendizaje de SQL que se centran en las respuestas de las consultas con respecto a instancias concretas de la base de datos proporcionadas por expertos (*e.g.,* ACME [17], SQLator [13], WebSQL [2], AsseSQL [12], SQLZoo[6], SQLJudge [9] y QueryViz [8]). Otro tipo de herramienta, como SQL Tutor [10], permite la selección automática de problemas planteados a cada alumno en términos de su modelo. SQL-LTM [6] es un módulo de tutorización que se basa en consultas de referencia con respecto a las que se comparan las de los alumnos.

Las herramientas que se basan en la comparación de la semántica pretendida de referencia con la real de la consulta se pueden considerar ortogonales a DESweb. En efecto, se aplican después de la ejecución de las consultas, en lugar de antes. Por lo tanto, cualquiera de ellas podría complementarse con los resultados que proporciona DESweb: su análisis semántico sería una primera etapa que deben superar las consultas antes de poder confrontar sus resultados con los de referencia.

Tan solo tenemos constancia de una herramienta similar a DESweb para el análisis semántico en tiempo de compilación: `sqllint` [4] (que de hecho inspiró este trabajo), pero que resulta decididamente insuficiente para su implantación en el aula. En primer lugar, tan solo se proporcionan mensajes de error para el análisis semántico, y ni trabaja con un motor de ejecución

[4]Esto no es en sí mismo un motivo para elegir DESweb (existen otros sistemas *on-line* como SQL Fiddle: `sqlfiddle.com`) sino más bien la realimentación que produce frente a errores.

[5]Varios alumnos preguntaron si no era posible hacer las prácticas de Oracle en DESweb (las dedicadas a procedimientos almacenados y disparadores, a los que DES no ofrece soporte).

[6]`https://sqlzoo.net`

de consultas ni con instancias de tablas. Generalmente, los alumnos en las prácticas y ejercicios cuentan no solo con la descripción de los metadatos, sino también con instancias concretas con las que probar sus consultas. Sin darles esta posibilidad, el uso de `sqllint` está muy limitado. En segundo lugar, el espectro de errores que cubre es bastante inferior al cubierto con DES (si bien es cierto que trata alguno que DES no): de 16 tipos de errores cubiertos por DES, solo 9 lo son por `sqllint` (no obstante, el tratamiento de los errores concretos puede ser diferente). El sistema `sqllint` también limita severamente el tipo de consultas que se pueden tratar (por ejemplo, no se pueden incluir expresiones o ciertas subconsultas), lo que implica que una gran parte de las consultas no pasen ni siquiera el análisis sintáctico (que por otra parte tampoco informa de cuáles son los errores). Tampoco dispone de interfaz gráfica de usuario. Aun siendo un sistema interesante, estas limitaciones descartan a `sqllint` como una herramienta efectiva para la docencia. En cambio, DES da soporte a un gran espectro del estándar ANSI/ISO de SQL, y como interfaz aporta tanto DESweb como un entorno gráfico de escritorio escrito en Java [14] (no descrito aquí) similar a sistemas comerciales.

7. Conclusiones

La conclusión fundamental de este trabajo es haber experimentado con éxito la aplicabilidad de DESweb en la asignatura *Bases de datos*. Con respecto a novedad y aportaciones, hemos construido un sistema *práctico* para el aprendizaje de SQL con realimentación semántica y con técnicas no empleadas antes para este análisis, como son el uso de resolutores de restricciones y su cooperación.

No obstante, este es un trabajo en evolución y esta conclusión se debe fundamentar en estudios de campo mucho más amplios. Por ello, uno de los objetivos no es solo aportar nuestra experiencia limitada, sino proporcionar esta herramienta a la comunidad docente con la intención de recibir una mayor realimentación y poder mejorar el sistema. Por ejemplo, además de mejorar la interfaz, aumentar la precisión de los análisis y capturar otro tipo de errores semánticos, también sería útil aportar mensajes de estilo o normativa de código.

Referencias

[1] ACM/IEEE-CS Joint Task Force on Computing Curricula. Computer science curricula 2013. Technical report, ACM Press and IEEE Computer Society Press, 2013.

[2] G. N. Allen. WebSQL: An Interactive Web Tool for Teaching Structured Query Language. En *Proceedings of AMCIS*, 2000.

[3] P. Atzeni *et al*. The Relational Model is Dead, SQL is Dead, and I Don'T Feel So Good Myself. *SIGMOD Record*, 42(2):64–68, 2013.

[4] S. Brass y C. Goldberg. Proving the safety of SQL queries. En *Proceedings of QSIC*, pp. 197–204, 2005.

[5] S. Brass y C. Goldberg. Semantic Errors in SQL Queries: A Quite Complete List. *The Journal of Systems and Software*, 79(5):630–644, 2006.

[6] R. Dollinger. SQL Lightweight Tutoring Module - Semantic Analysis of SQL Queries based on XML Representation and LINQ. En *Proceedings of ED-MEDIA*, pp. 3323–3328, 2010.

[7] T. Früwirth y S. Abdennadher. *Essentials of Constraint Programming*. Springer-Verlag, 2003.

[8] W. Gatterbauer. Databases will Visualize Queries too. *Proceedings of the VLDB Endowment*, 4(12):1498–1501, 2011.

[9] L. G. Gutiérrez, F. J. Martínez y P. Vega. Evaluador de sentencias de bases de datos en la formación de ingenieros. *ANFEI Digital*, (5):1–9, 2016.

[10] A. Mitrovic. Fifteen years of constraint-based tutors: what we have achieved and where we are going. *User Modeling and User-Adapted Interaction*, 22(1):39–72, 2012.

[11] A. Ogborn. Creating Web Applications in SWI-Prolog, 2015. `www.pathwayslms.com/swipltuts/html/index.html`.

[12] J. R. Prior. AsseSQL: An Online, Browser-based SQL Skills Assessment Tool. En *Proceedings of ITiCSE*, pp. 327–327, 2014.

[13] S. Sadiq, M. Orlowska, W. Sadiq y J. Lin. SQLator: An Online SQL Learning Workbench. *SIGCSE Bulletin*, 36(3):223–227, 2004.

[14] F. Sáenz-Pérez. ACIDE: An Integrated Development Environment Configurable for LaTeX. *The PracTeX Journal*, 2007(3), 2007.

[15] F. Sáenz-Pérez. DES: un recurso para el aprendizaje de bases de datos deductivas. En *Actas de JENUI*. Thomson, 2007.

[16] F. Sáenz-Pérez. DES: A Deductive Database System. *Electronic Notes on Theoretical Computer Science*, 271:63–78, 2011.

[17] J. Soler, F. Prados, I. Boada y J. Poch. A Web-based tool for teaching and learning SQL. En *Proceedings of ITHET*, 2006.

[18] M. Stonebraker *et al.* Third-Generation Database System Manifesto - The Committee for Advanced DBMS Function. *SIGMOD Record*, 19(3):31–44, 1990.

[19] J. Wielemaker, T. Schrijvers, M. Triska y T. Lager. SWI-Prolog. *Theory and Practice of Logic Programming*, 12(1-2):67–96, 2012.