CQA-WF: Respuestas Consistentes a Consultas Conjuntivas a través de la Semántica Well-Founded

Julia Belmar Jeniffer Cuevas Mónica Caniupán

Universidad del Bio-Bío
Departamento de Sistemas de Información
Concepción - Chile
{jbelmar,jcuevas,mcaniupa}@ubiobio.cl

Resumen

Las bases de datos pueden volverse inconsistentes con respecto a sus *restricciones de integridad* (RIs) por diversas razones, por ejemplo, luego de actualizaciones de datos. En estos casos, es posible obtener reparaciones de bases de datos, es decir, nuevas instancias que satisfacen las RIs y que se obtienen a partir de ejecutar ciertas actualizaciones a la instancia original, y que difieren de manera minimal de ésta. Una respuesta a una consulta conjuntiva es consistente si es una respuesta en toda reparación de la instancia original. Las reparaciones de bases de datos con respecto a restricciones de integridad generales pueden ser especificadas a través de programas en lógica disyuntiva, llamados programas de reparación. Estos programas pueden ser utilizados para computar respuestas consistentes a consultas de primer orden a través de la semántica de modelos estables. Sin embargo, la evaluación de respuestas consistentes a consultas a través de programas de reparación es Π_2^p-completo. En este artículo presentamos *CQA-WF* un sistema que permite computar respuestas consistentes para una clase particular de consultas conjuntivas evaluadas sobre bases de datos que no satisfacen un conjunto de dependencias funcionales (DFs), una clase particular y común de RIs. *CQA-WF* evalúa las consultas utilizando la semántica *well-founded*, semántica alternativa a la semántica de modelos estables para programas Datalog con negación y que puede ser computada en tiempo polinomial.

Palabras claves: semántica well-founded, inconsistencias, dependencias funcionales, respuestas consistentes.

1. Introducción

No es poco común que las bases de datos (BDs) se vuelvan inconsistentes con respecto a sus restricciones de integridad (RIs), las que capturan la semántica o significado de una base de datos [6]. Por ejemplo, en integración de datos cuando datos de múltiples fuentes, localmente consistentes, al ser integrados no satisfacen las restricciones de integridad globales.

Cuando una base de datos no satisface sus restricciones de integridad es posible repararla. El concepto de reparación de bases de datos relacionales fue introducido por primera vez en [1] (ver [5] para más detalles sobre reparaciones de bases de datos). Intuitivamente, una reparación para una instancia de base de datos D con respecto a un conjunto de restricciones de integridad RI es una nueva instancia D', sobre el mismo esquema y dominio de D, que satisface RI, se obtiene a partir de actualizaciones a D (inserción y/o eliminación de tuplas) y difiere de manera minimal de ésta, considerando inclusión de tuplas. En [1] también se entrega el concepto de *respuesta consistente* a consultas de primer orden. Una respuesta a una consulta $\mathcal{Q}$ es consistente, si ésta es una respuesta a $\mathcal{Q}$ en cada una de las reparaciones de la base de datos.

Se ha demostrado que el problema de obtener respuestas consistentes a consultas desde bases de datos inconsistentes con respecto a sus RIs es, en general, Π_2^p-completo [6, 5, 21]. Sin embargo, es posible utilizar programas en lógica disyuntiva para computar respuestas consistentes a consultas. El procedimiento es como sigue: (i) se genera un programa en lógica disyuntiva, denominado el programa de reparación, para especificar

las reparaciones de una instancia de base de datos D respecto de un conjunto de RIs RI, programa que se denota con $\Pi(D, RI)$. Se ha demostrado que existe una correspondencia uno-a-uno entre los modelos estables de $\Pi(D, RI)$ y las reparaciones de D [2, 3, 4, 7, 8, 13, 17]. (ii) Se genera el programa $\Pi(\mathcal{Q})$ que representa la consulta. (iii) Se evalúa el programa $\Pi(D, RI) \cup \Pi(\mathcal{Q})$ en algún razonador lógico como DLV[1] [19]. Las respuestas consistentes a la consulta $\mathcal{Q}$ se obtienen a través de un razonamiento cauteloso sobre los modelos estables de $\Pi(D, RI) \cup \Pi(\mathcal{Q})$. Se ha mostrado además que evaluar programas en lógica es ineficiente para grandes volúmenes de datos, sin embargo, en [10] se proponen optimizaciones al proceso de evaluación de respuestas consistentes a consultas vía programas de reparación.

En este artículo se presenta *CQA-WF* un sistema que computa respuestas consistentes a consultas conjuntivas, las que son evaluadas sobre instancias de bases de datos que pueden ser inconsistentes con respecto a un conjunto de dependencias funcionales (DFs), una clase particular y común de restricciones de integridad. *CQA-WF* utiliza una semántica alternativa a la semántica de modelos estables [16], denominada semántica *well-founded* [24, 20] (en adelante SWF) para computar, en tiempo polinomial, respuestas consistentes a consultas conjuntivas.

El resto del artículo se organiza de la siguiente manera: La sección 2 presenta conceptos generales incluyendo la semántica well-founded. La sección 3 describe la forma de utilizar la SWF para la computación de respuestas consistentes a consultas conjuntivas. La sección 4 presenta la arquitectura del sistema. Finalmente, la sección 5 presenta conclusiones y trabajo futuro.

2. Preliminares

Una base de datos relacional tiene un esquema de la forma $\Sigma = (\mathcal{U}, \mathcal{R}, \mathcal{B})$ donde $\mathcal{U}$ es el posible dominio infinito de la base de datos, con $null \in \mathcal{U}$, $\mathcal{R}$ es un conjunto fijo de predicados de la base de datos, cada uno de ellos con un conjunto finito ordenado de atributos, y $\mathcal{B}$ es un conjunto de predicados built-in, por ejemplo $x = 5$, $x > y$, etc. $R[i]$ denota el atributo en la posición i del predicado $R \in \mathcal{R}$. Las instancias de base de datos de un esquema Σ son conexiones finitas D de átomos instanciados de la forma $R(c_1, \ldots, c_n)$, denominados *tuplas de la base de datos*, donde $R \in \mathcal{R}$ y $(c_1, \ldots, c_n)$ es una tupla de constantes. Las extensiones para los predicados built-in son fijas y posiblemente infinitas en cada instancia de base de datos. Existe un conjunto fijo de restricciones de integridad, que son sentencias en el lenguaje de primer orden $\mathcal{L}(\Sigma)$ determinado por el esquema Σ. Una dependencia funcional (DF) sobre una relación P se denota por [12]:

$$\forall \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 \bar{x}_5 . \neg [P(\bar{x}_1, \bar{x}_2, \bar{x}_4) \wedge P(\bar{x}_1, \bar{x}_3, \bar{x}_5) \wedge \bar{x}_2 \neq \bar{x}_3]. \tag{1}$$

Nosotros denotaremos una dependencia funcional sobre una relación P con $X \rightarrow Y$, donde X e Y son conjuntos de atributos de P y $X \cap Y = \emptyset$.

Para que una base de datos sea consistente, debe satisfacer todas sus DFs. Para analizar consistencia de bases de datos en presencia de valores nulos necesitamos introducir el concepto de *atributo relevante* introducido en [8] para restricciones de integridad generales. Si solamente consideramos dependencias funcionales, un atributo es relevante si aparece al menos dos veces en una DF ψ. El conjunto de atributos relevantes para una dependencia funcional ψ se denota por $\mathcal{A}$. Si tenemos un conjunto de atributos relevantes $\mathcal{A}$ y un predicado P en $\mathcal{R}$, $P^{\mathcal{A}}$ es el predicado P restringido a los atributos relevantes en $\mathcal{A}$. Correspondientemente $D^{\mathcal{A}}$ es la base de datos D restringido a los predicados $P^{\mathcal{A}}$.

Una DF $\psi : X \rightarrow Y$ sobre una relación P es satisfecha por una instancia de base de datos D, denotado por $D \models_N \psi$ si y sólo si $D^{\mathcal{A}(\psi)} \models \psi^N$, donde $\models_N$ denota satisfacción en presencia de valores nulos. Esto significa, que D satisface ψ si existe un *null* en algún atributo relevante de ψ, o si la dependencia funcional ψ se satisface de manera tradicional, es decir, si dos tuplas en P tienen los mismos valores para los atributos en X entonces también comparten el valor en Y.

Ejemplo 1. Consideremos la DF ψ: $\forall xyzmn(P(x, y, m) \wedge P(x, z, n) \rightarrow y = z)$ sobre la relación $P(x, y, z)$ y la instancia de base de datos $D = \{P(a, b, a), P(b, c, a),\ P(a, null, d)\}$. Los atributos relevantes para ψ son: x, y, z, ya que cada uno de ellos aparece dos veces en ψ. Entonces, para verificar consistencia de la base de datos basta con verificar si $\forall xyz(P^{\mathcal{A}(\psi)}(x, y) \wedge P^{\mathcal{A}(\psi)}(x, z) \rightarrow y = z)$. La instancia de base de datos D es

[1] `http://www.dbai.tuwien.ac.at/proj/dlv/`

Anotación	Átomo	La Tupla $P(\bar{a})$...
f_a	$P(\bar{a}, f_a)$	debería ser declarada falsa (eliminación)
$t^\star$	$P(\bar{a}, t^\star)$	es verdadera o se vuelve verdadera
$t^{\star\star}$	$P(\bar{a}, t^{\star\star})$	es verdadera en la reparación

Cuadro 1: Anotaciones en el programa de reparación $\Pi(D, DF)$.

consistente con respecto a ψ, puesto que a pesar de que existen dos tuplas que comparten el primer atributo, $P(a, b, a)$ y $P(a, null, d)$, la segunda tiene un valor nulo en un atributo relevante. Por el contrario, la instancia $D' = \{P(a, b, c), P(a, c, d), P(b, c, a)\}$, es inconsistente a través de las primeras dos tuplas puesto que ambas coinciden en el primer atributo pero difieren en el segundo. □

Cuando una instancia de base de datos D es inconsistente con respecto a sus dependencias funcionales, es posible repararla. Intuitivamente, una reparación para D con respecto a un conjunto de DFs DF es una nueva instancia de base de datos D' que satisface DF, comparte el esquema y dominio de D, se obtiene a partir de D vía eliminación de tuplas y difiere de manera minimal de ésta considerando inclusión de tuplas [1]. Para definir formalmente reparación, necesitamos el concepto de distancia: Sean D, D' instancias de bases de datos sobre el mismo esquema Σ y dominio $\mathcal{U}$. La distancia entre D y D' denotada con $\Delta(D, D')$ es la diferencia simétrica: $\Delta(D, D') = (D \setminus D') \cup (D' \setminus D)$ [1].

Definición 1. [8] Sean D, D', D'' instancias de base de datos, sobre un esquema Σ y dominio $\mathcal{U}$. Diremos que $D' \leq_D D''$ si y sólo si:

1. Para cada átomo $P(\bar{a}) \in \Delta(D, D')$, con $\bar{a} \in (\mathcal{U} \setminus \{null\})$, se tiene que $P(\bar{a}) \in \Delta(D, D'')$.
2. Para cada átomo $Q(\bar{a}, \bar{null}) \in \Delta(D, D')$, con $\bar{a} \in (\mathcal{U} \setminus \{null\})$, existe un $\bar{b} \in \mathcal{U}$, tal que $Q(\bar{a}, \bar{b}) \in \Delta(D, D'')$ y $Q(\bar{a}, \bar{b}) \notin \Delta(D, D')$. □

Este orden parcial es usado para definir las reparaciones de una BD inconsistente.

Definición 2. [8] Dada una instancia D sobre un esquema Σ y un conjunto DF de DFs, una reparación de D respecto a DF es una instancia D' sobre Σ, tal que: (i) $D' \models_N DF$. (ii) D' es $\leq_D -minimal$ en la clase de instancias de bases de datos que satisfacen DF con respecto de $\models_N$, es decir, no existe una instancia D'' con $D'' <_D D'$, donde $D'' <_D D'$ significa $D'' \leq_D D'$ pero no se cumple $D' \leq_D D''$. □

Ejemplo 2. Considere la relación $P(x, y)$, la $DF : x \to y$, y la instancia de base de datos $D = \{P(b, c), P(c, a), P(b, e)\}$. La instancia D es inconsistente con respecto a DF por las tuplas $P(b, c)$ y $P(b, e)$. Existen dos reparaciones minimales: $D_1 = \{P(b, c), P(c, a)\}$ y $D_2 = \{P(b, e), P(c, a)\}$. D_1 se obtiene a partir de D eliminando la tupla $P(b, e)$. D_2 se obtiene eliminando la tupla $P(b, c)$. Por lo tanto, las distancias entre las reparaciones minimales y D corresponden a: $\Delta(D, D_1) = \{P(b, e)\}$ y $\Delta(D, D_2) = \{P(b, c)\}$. Note que la instancia $D_3 = \{P(c, a)\}$ también es una reparación para D, pero no es minimal, puesto que para obtenerla se han eliminado las dos tuplas en conflicto: $P(b, c)$ y $P(b, e)$. Por lo tanto, $\Delta(D, D_3) = \{P(b, c), P(b, e)\}$ y tenemos que $\Delta(D, D_1) \subseteq \Delta(D, D_3)$ y $\Delta(D, D_2) \subseteq \Delta(D, D_3)$. □

2.1. Programas de Reparación

Las reparaciones de una base de datos respecto de sus DFs, pueden ser especificadas con programas en lógica disyuntiva con semántica de modelos estables [16, 23]. Para una instancia de base de datos D y un conjunto DF de dependencias funcionales, se construye un programa de reparación denotado por $\Pi(D, DF)$ que utiliza las constantes definidas en el Cuadro 1. Es posible demostrar que existe una correspondencia uno a uno entre los modelos estables del programa de reparación $\Pi(D, DF)$ y las reparaciones de la BD [8].

Ejemplo 3. (Ejemplo 2 continuado) El programa de reparación $\Pi(D, DF)$ para la $DF : x \to y$ y la instancia $D = \{P(b, c), P(c, a), P(b, e)\}$ contiene las siguientes reglas y tuplas:

1. $P(b, c)$. $P(c, a)$. $P(b, e)$.

2. $P\_(x, y, f_a) \vee P\_(x, z, f_a) \leftarrow P\_(x, y, t^\star), P\_(x, z, t^\star), y \neq z, x \neq null, y \neq null, z \neq null.$
3. $P\_(x, y, t^\star) \leftarrow P(x, y).$
4. $P\_(x, y, t^{\star\star}) \leftarrow P\_(x, y, t^\star), not\ P\_(x, y, f_a).$

La regla (1) corresponde a las tuplas (hechos) de la base de datos. La regla (2) expresa que alguno de los átomos $P(x, y)$ o $P(x, z)$ debe ser eliminado para restaurar consistencia (ambos son anotados con f_a). La regla (3) define tuplas de la forma $P\_(x, y, t^\star)$ por cada tupla $P(x, y)$ en D. La regla (4) expresa que una tupla es completamente verdad cuando se tiene $P\_(x, y, t^\star)$, y no $P\_(x, z, f_a)$; es decir, se tiene $P\_(x, y, t^\star)$ y la tupla no ha sido eliminada en el proceso de reparación.

Los modelos estables de $\Pi(D, DF)$[2] son: $\mathcal{M}_1 = \{P\_(b, e, f_a), P\_(b, c, t^{\star\star}), P\_(c, a, t^{\star\star})\}$, $\mathcal{M}_2 = \{P\_(b, c, f_a), P\_(b, e, t^{\star\star}), P\_(c, a, t^{\star\star})\}$. Del modelo estable $\mathcal{M}_1$ se deriva que la consistencia de D se restaura eliminando el átomo $P(b, e)$ (ya que está anotado con f_a), por lo tanto la reparación minimal es: $D_1 = \{P(b, c), P(c, a)\}$ (ambos átomos aparecen anotados con $t^{\star\star}$). La segunda reparación se obtiene al eliminar el átomo $P(b, c)$ según lo sugerido en $\mathcal{M}_2$ y corresponde a la reparación minimal $D_2 = \{P(b, e), P(c, a)\}$. □

Las respuestas consistentes a una consulta de primer orden se definen como sigue.

Definición 3. [1] Dada una instancia de base de datos D, una tupla $\bar{t}$ es una respuesta consistente a una consulta $\mathcal{Q}(\bar{x})$ en D si y sólo si es una respuesta a la consulta $\mathcal{Q}(\bar{x})$ en cada reparación D' de D. Si la consulta es booleana (con respuesta *si* o *no*), la respuesta consistente es "*sí*" si $\mathcal{Q}$ es verdad en cada reparación D' de D; y "*no*" en caso contrario. El conjunto de respuestas consistentes a la consulta $\mathcal{Q}$ en D respecto de sus DFs en DF se denota con $ConsA(\mathcal{Q})$. □

Para poder usar los programas de reparación para computar respuestas consistentes a consultas de primer orden, éstas se expresan como programas Datalog. Es decir, dada una consulta $\mathcal{Q}$, se genera un programa Datalog $\Pi(\mathcal{Q})$ reemplazando cada literal positivo $P(\bar{x})$ por $P\_(\bar{x}, t^{\star\star})$, y cada literal negativo $not\ P(\bar{x})$ por $not\ P(\bar{x}, t^{\star\star})$, también se agrega un átomo Ans para recolectar las respuestas en cada modelo estable. Para computar las respuestas consistentes a $\mathcal{Q}$ se genera el programa $\Pi(D, DF, \mathcal{Q})$ $(\Pi(\mathcal{Q}) \cup \Pi(D, DF))$. Las respuestas consistentes se obtienen evaluando bajo la semántica cautelosa de los modelos estables [15, 16].

Ejemplo 4. (Ejemplo 3 continuado) Consideremos la consulta $\mathcal{Q}(x) : Ans(x) \leftarrow P(x, y)$, $\Pi(\mathcal{Q})$ es: $Ans(x) \leftarrow P\_(x, y, t^{\star\star})$. Los siguientes son los modelos estables del programa $\Pi(D, DF, \mathcal{Q})$: $\mathcal{M}_1 = \{P\_(b, e, f_a), P\_(b, c, t^{\star\star}), P\_(c, a, t^{\star\star}), Ans(b), Ans(c)\}$, $\mathcal{M}_2 = \{P\_(b, c, f_a), P\_(b, e, t^{\star\star}), P\_(c, a, t^{\star\star}), Ans(b), Ans(c)\}$. Por lo tanto, la respuesta consistente para $\mathcal{Q}$ es $\{b, c\}$, ya que $Ans(b)$ y $Ans(c)$ son comunes en ambos modelos estables. □

2.2. Semántica Well-Founded

La semántica well-founded para programas Datalog normales fue introducida en [24], y luego fue extendida para programas con disyunción en [20, 22]. Es una semántica alternativa a la semántica de modelos estables [15, 16, 23] y puede ser computada en tiempo polinomial. De hecho, en [15] se demostró que si un programa normal tiene un modelo well-founded total, éste coincide con su único modelo estable. En lo que sigue de este artículo, se considera la semántica well-founded tal como fue introducida en [20].

La interpretación well-founded (IWF) de un programa Π, denotada por $W_\Pi = \langle W^+, W^-, W^u \rangle$, está compuesta por tres subconjuntos de átomos: (i) W^+, que corresponde al conjunto de átomos con valor de verdad *verdadero*. (ii) W^-, que corresponde al conjunto de átomos con valor de verdad *falso*. Este conjunto también se conoce como conjunto no-fundamentado (del inglés "unfounded set"). (iii) W^u es el conjunto de átomos con valor de verdad *desconocido*.

La semántica de modelos estables trata de encontrar modelos alternativos para un programa, dándoles a los átomos un valor verdadero o falso, en los distintos modelos. Así, un programa normal puede tener varios modelos estables, pero sólo una IWF. Los modelos estables pueden computarse a partir de la IWF, partiendo de los átomos que son verdaderos o falsos en la interpretación, se trata de dar distintos valores a los átomos no determinados. De lo anterior se concluye que el conjunto de átomos que son verdaderos en la IWF están contenidos en cada modelo estable del programa respectivo, y el conjunto de átomos falsos es un subconjunto de los átomos que son falsos en cada modelo estable [24][20].

[2] Para simplificar la presentación, los modelos estables se despliegan restringidos a los átomos anotados con f_a y $t^{\star\star}$.

La IWF se define como el menor punto fijo de un operador $\mathcal{W}_\Pi$ que mapea interpretaciones de la forma $I = \langle I^+, I^-, I^u \rangle$, con I^+, I^-, I^u, conjuntos disjuntos de átomos instanciados (ground) que, en conjunto, cubren toda la base de Herbrand del programa [15]. En este caso, I^+ es el conjunto de átomos positivos en I, I^- es el conjunto de átomos negativos en I, es decir, $not.\{not\ A \mid not\ A \in I\}$[3] y I^u denota al conjunto de literales que contiene a los átomos que no están ni en I^+ ni en I^-. La base para definir la IWF es el conjunto de átomos no-fundamentado.

Definición 4. [20] Sea I una interpretación para un programa disyuntivo Π. Un conjunto $X \subseteq B_\Pi$ de átomos instanciado es un conjunto no-fundamentado para Π con respecto a I si para cada $a \in X$ (un átomo no fundamentado en X), para cada regla $r \in \Pi$ (la versión instanciada de Π), tal que $a \in H(r)$ (la cabeza de r), al menos una de las siguientes condiciones se cumple:
1. $B(r) \cap not.I \neq \emptyset$, es decir, el cuerpo de r es falso con respecto a I.
2. $B^+(r) \cap X \neq \emptyset$, es decir, algún literal positivo del cuerpo pertenece a X.
3. $(H(r) \setminus X) \cap I \neq \emptyset$, es decir, un átomo en la cabeza de r, distinto de a y otros elementos en X, es verdad con respecto a I. □

La unión de todos los conjuntos de átomos no-fundamentados para Π con respecto a I se denomina el *super conjunto de átomos no-fundamentados* y se denota por $GUS_\Pi(I)$. Para programas Datalog normales el GUS es también un conjunto no-fundamentado, pero para programas disyuntivos esto no necesariamente se cumple [20]. Sin embargo, para nuestros programas de reparación se puede garantizar que GUS es siempre no-fundamentado [9].

Definición 5. [20] Dado un programa disyuntivo Π, el *operador well-founded* denotado por $\mathcal{W}_\Pi$ se define, sobre una interpretación I para la cual $GUS_\Pi(I)$ es no-fundamentado por:

$$\mathcal{W}_\Pi(I) := \Gamma_\Pi(I) \cup not.GUS_\Pi(I) \tag{2}$$

donde $\Gamma_\Pi(I)$ es el operador que declara un átomo A verdad con respecto a I, si existe una regla en Π, tal que A está en la cabeza de la regla, el cuerpo de dicha regla es verdad con respecto a I y los otros átomos en la cabeza de la regla (si ésta fuese disyuntiva) son falsos con respecto a I. □

La IWF de un programa disyuntivo instanciado Π está definida por el punto fijo W_Π de las interpretaciones definidas por: $W_0 := \emptyset$, $W_{k+1} := \mathcal{W}_\Pi(W_k)$. W_Π se computa en tiempo polinomial [20].

Ejemplo 5. (Ejemplo 3 continuado) La IWF para el programa de reparación es[4]: (i) $W^+ = \{P\_(c, a, t^{\star\star})\}$. (ii) $W^u = \{P\_(b, e, f_a), P\_(b, e, t^{\star\star}), P\_(b, c, f_a), P\_(b, c, t^{\star\star})\}$. (iii) W^- está formado por combinaciones de átomos de la forma $P\_(c_1, c_2, f_a)$ y $P\_(c_1, c_2, t^{\star\star})$ que son falsos. □

3. Interpretación Well-Founded y Respuestas Consistentes a Consultas

La IWF de los programas de reparación puede utilizarse para evaluar consultas conjuntivas de la forma (3) y (4) las que no permiten joins entre relaciones [12]:

$$Ans(w_1 \ldots w_m) \leftarrow \exists z_1 \ldots z_n (P_1(\bar{x}_1), \ldots P_n(x_n)), \tag{3}$$

donde $w_1 \ldots w_m$, $z_1, \ldots z_n$ son todas las variables que aparecen en los átomos del cuerpo de la consulta, cada $\bar{x}_i$ corresponde a la aridad de P_i y las variables $w_1 \ldots w_m$ son variables libres de la consulta. Diremos que la consulta es *simple* si no hay constantes ni símbolos repetidos en ella. Usualmente, los cuantificadores existenciales se encuentran implícitos en una consulta, sin embargo en este artículo se escribirán explícitamente en la consulta.

$$Ans \leftarrow \exists z_1 \ldots z_n (P_1(\bar{x}_1), \ldots P_n(\bar{x}_n)), \tag{4}$$

donde $z_1, \ldots z_n$ son variables que aparecen en los átomos de la consulta, y $\bar{x}_1 \ldots \bar{x}_n$ son variables y/o constantes.

[3] Para un conjunto de literales $\mathcal{L}$, $not.\mathcal{L}$ denota al conjunto de literales que son complementarios a $\mathcal{L}$.
[4] Por simplificación, en el resto del artículo se desplegará la IWF restringida a átomos anotados con $t^{\star\star}$, con f_a o átomos Ans.

Definición 6. [9] Para una instancia D, un conjunto DF de DFs, y una consulta conjuntiva $\mathcal{Q}$ de la forma (3), una tupla $\bar{t}$ es una *respuesta well-founded* a $\mathcal{Q}$ si $Ans(\bar{t})$ está en $W^+ \cup W^u$ de la IWF del programa $\Pi(D, DF, \mathcal{Q})$. El conjunto de respuestas well-founded a $\mathcal{Q}$ es denotado por $WFA^{+u}(\mathcal{Q})$. □

Diferenciaremos los atributos de una relación de acuerdo a la siguiente definición [9]: Dada una relación P, con $DF : X \rightarrow Y$, los atributos en P se dividen en: (i) atributos en el antecedente (atributos en X). (ii) Atributos en el consecuente (atributos en Y). (iii) Atributos simples, los cuales no están presentes ni en X ni en Y. El primer caso a analizar corresponde a consultas donde los atributos proyectados son atributos que están en el antecedente de alguna DF.

Definición 7. [9] Sea D una instancia de base de datos, DF un conjunto de DFs (a lo más una DF o clave primaria por relación), si las variables libres en la consulta $\mathcal{Q}$ de la forma (3) corresponden a atributos en el antecedente de una DF, entonces las respuestas consistentes para $\mathcal{Q}$ con respecto a DF, coinciden con las respuestas well-founded para $\mathcal{Q}$. □

En este caso, el subconjunto W^u de la IWF de $\Pi(D, DF, \mathcal{Q})$ puede usarse directamente para responder consistentemente. Esto se debe a que, aún cuando existen tuplas anotadas con $t^{\star\star}$, que se refieren a tuplas inconsistentes con respecto a las DFs, dichas tuplas comparten el mismo valor para el atributo proyectado.

Ejemplo 6. Sea la relación de alumnos $A(id, nombre, fecha\_nac, edad)$ y la siguiente instancia $D = \{A(1, juan, 2/10/00, 11), A(1, felipe, 2/11/91, 20), A(2, felipe, 3/11/91, 20)\}$, la $DF : id \rightarrow nombre$. La consulta $\mathcal{Q} : Ans(x) \leftarrow \exists yzwA(x, y, z, w)$ consulta por los valores id de los alumnos, el cual, es un atributo en el antecedente de la DF. Los conjuntos W^+ y W^u de la IWF del programa $\Pi(D, DF, \mathcal{Q})$ corresponden a:

- $W^+ = \{A\_(2, felipe, 3/11/91, 20, t^{\star\star}), \underline{Ans(2)}\}$.
- $W^u = \{A\_(1, juan, 2/10/00, 11, t^{\star\star}), A\_(1, felipe, 2/11/91, 20, t^{\star\star}), \underline{Ans(1)}\}$.

La respuesta well-founded es $WFA^{+u}(\mathcal{Q}) = \{1, 2\}$, que corresponde a la respuesta consistente para $\mathcal{Q}$. □

En algunos casos es necesario realizar una reescritura de consultas para utilizar el conjunto W^u de la IWF del programa $\Pi(D, DF)$. Esto se ilustra en el siguiente ejemplo.

Ejemplo 7. Consideremos la misma relación $A(id, nombre, fecha\_nac, edad)$ y la $DF : fecha\_nac \rightarrow edad$, la instancia $D = \{A(1, juan, 2/10/00, 11), A(1, juan, 2/10/00, 15), A(2, felipe, 2/10/00, 11)\}$ es inconsistente con respecto a DF. Existen dos reparaciones minimales: $D_1 = \{A(1, juan, 2/10/00, 11), A(2, felipe, 2/10/00, 11)\}$, $D_2 = \{A(1, juan, 2/10/00, 15), A(2, felipe, 2/10/00, 11)\}$. Los conjuntos W^+ y W^u de la IWF para $\Pi(D, DF, \mathcal{Q})$ con $\mathcal{Q} : Ans(z, w) \leftarrow \exists xyA(x, y, z, w)$, corresponden a:

- $W^+ = \{A\_(2, felipe, 2/10/00, 11, t^{\star\star}), \underline{Ans(2/10/00, 11)}\}$
- $W^u = \{A\_(1, juan, 2/10/00, 11, t^{\star\star}), A\_(1, juan, 2/10/00, 15, t^{\star\star}), \underline{Ans(2/10/00, 15)}, \underline{Ans(2/10/00, 11)}\}$.

Por lo tanto, $WFA^{+u}(\mathcal{Q}) = \{(2/10/00, 11), (2/10/00, 15)\}$. Pero la respuesta consistente es $\langle 2/10/00, 11\rangle$.

La respuesta correcta (consistente) se obtiene al reescribir $\mathcal{Q}$ como:$\mathcal{Q}' : Ans(z, w) \leftarrow \exists xyA\_(x, y, z, w, t^{\star\star}) \wedge (\forall x'y'w'(A\_(x', y', z, w', t^{\star\star}) \rightarrow w' = z))$. La nueva consulta $\mathcal{Q}'$ se evalúa sobre W^u de la IWF de $\Pi(D, DF)$, y se obtiene una respuesta vacía, por lo tanto, la unión de respuestas desde $W^+ \cup W^u$ es $\langle 2/10/00, 11\rangle$, la cual coincide con la respuesta consistente a $\mathcal{Q}$. □

Con respecto a consultas conjuntivas de la Forma (4):

Definición 8. [9] Sea D una instancia de base de datos y DF un conjunto de DFs (a lo más una DF o clave primaria por relación), la respuesta consistente a una consulta $\mathcal{Q}$ del tipo (4) con respecto a DF es "sí" si:

- Ans está en W^+ de la IWF de $\Pi(D, DF, \mathcal{Q})$, o
- Hay respuesta positiva para $\mathcal{Q}'$ evaluada en el conjunto W^u de la IWF de $\Pi(D, DF)$, donde $\mathcal{Q}'$ es la reescritura de $\mathcal{Q}$.

En caso contrario la respuesta consistente es "*no*". □

El Cuadro 2 resume el uso de la IWF para obtener respuestas consistentes a consultas.

Tipo Consulta	Reescritura	Respuesta a $\mathcal{Q}$
Consultas de la forma (3), $\bar{w}$ se refiere sólo a atributos antecedentes de alguna DF	No	La respuesta se encuentra en $W^+ \cup W^u$ de la IWF de $\Pi(D, DF, \mathcal{Q})$
Consultas de la forma (3), $\bar{w}$ contiene al menos un atributo consecuente de alguna DF	Si, se obtiene $\mathcal{Q}'$	Las respuestas consistentes se obtienen mediante la unión de las respuestas a $\mathcal{Q}$ obtenidas desde W^+ de la IWF de $\Pi(D, DF, \mathcal{Q})$, más las respuestas obtenidas para $\mathcal{Q}'$ evaluadas sobre el conjunto W^u de la IWF de $\Pi(D, DF)$
Consultas de la forma (4)	Si $Ans \notin W^+$ de la IWF de $\Pi(D, DF, \mathcal{Q})$	Si $Ans \in W^+$ de la IWF de $\Pi(D, DF, \mathcal{Q})$ la respuesta consistente es *si*. En caso contrario, se evalúa $\mathcal{Q}'$ en W^u de la IWF de $\Pi(D, DF)$

Cuadro 2: Cuadro resumen del uso de la IWF de programas.

4. Descripción del Sistema

CQA-WF es un sistema Web para computar respuestas consistentes a consultas en base a la interpretación well-founded de programas. Para el procesamiento de consultas, *CQA-WF* utiliza un algoritmo implementado en Lenguaje PROLOG[5]. Este algoritmo computa la IWF (conjuntos W^+ y W^u solamente) para programas Datalog normales, es decir, programas con negación pero sin disyunción[6]. Sin embargo, los programas de reparación restringidos a DFs pueden transformarse en programas Datalog con negación, sin disyunción [9]. Por lo tanto, *CQA-WF* genera programas de reparación sin disyunción que son equivalentes a los programas definidos en la Sección 2 (ver [9] para mayores detalles).

CQA-WF computa respuestas consistentes para consultas conjuntivas de la forma (3) y (4). La Figura 1 muestra la arquitectura y el flujo de información del sistema. *CQA-WF* permite editar instancias de bases de datos, dependencias funcionales, e ingresar consultas. El sistema valida la sintaxis de las consultas, genera el programa de reparación $\Pi(D, DF)$ para una instancia D y conjunto de DFs DF, el programa $\Pi(D, DF, \mathcal{Q})$ y la reescritura de consultas si corresponde.

El sistema se compone de ocho módulos codificados en lenguaje *C* [18].

1. *Editar BD:* permite editar la base de datos ya sea a nivel de tuplas o de esquema. Además este módulo permite visualizar las tuplas en cada relación de la base de datos.

2. *Ingresar DFs:* permite ingresar DFs. Este módulo se conecta con la base de datos para obtener la información de las relaciones y sus atributos. Luego, las DFs son validadas y almacenadas en el archivo "DF".

3. *Ingresar Consulta:* permite ingresar las consultas a ser evaluadas por el usuario.

4. *Preparar Datos:* obtiene el nombre de cada relación de la base de datos junto con su aridad, datos que son luego utilizados para generar el programa de reparación.

5. *Generar Programa de Reparación:* este módulo genera el programa de reparación y lo almacena en el archivo "p_reparacion.pl".

6. *Computar Interpretación Well Founded:* este módulo realiza una llamada a la aplicación que obtiene la IWF del programa de reparación, se genera el archivo "salida", que almacena los conjuntos W^+ y W^u de la IWF de $\Pi(D, DF)$.

7. *Computar Respuestas Consistentes:* este módulo obtiene la respuesta well-founded para la consulta ingresada, para ello computa de manera incremental la IWF del programa $\Pi(D, DF, \mathcal{Q})$.

[5] `http://www.swi-prolog.org/`
[6] Algoritmo reportado en *"A Tiny Interpreter for Datalog with Well-Founded"*, Bertram Ludäsher, 1995.

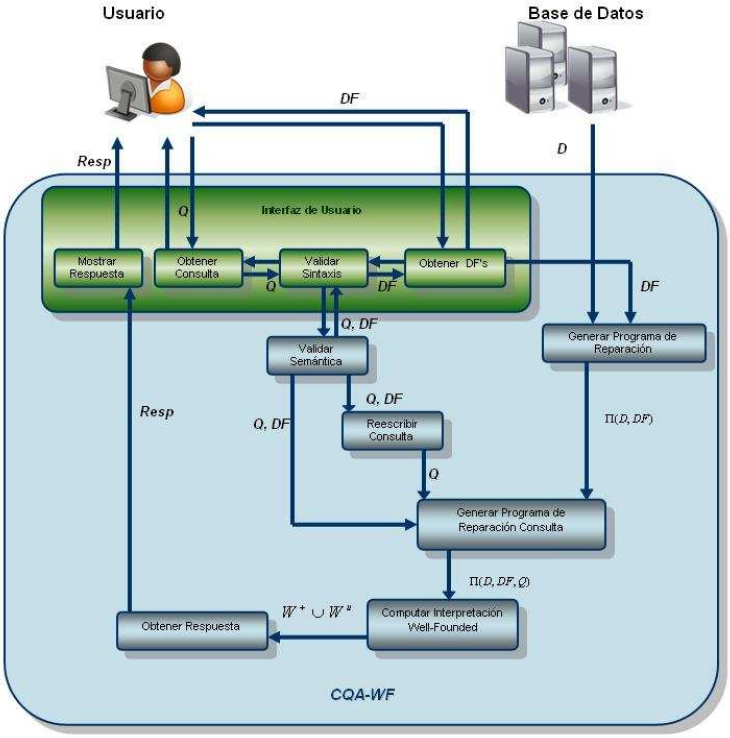

Figura 1: Arquitectura sistema *CQA-WF*.

8. *Procesamiento:* este módulo actúa como nexo entre la aplicación Web y el resto de los programas en implementados en C y PROLOG. Este módulo además controla los diversos eventos del sistema.

Por restricciones de espacio solamente destacamos el Algoritmo 1 que procesa consultas de la forma (3). El algoritmo recibe como entrada una consulta con proyección, además de las respectivas dependencias funcionales y la instancia de base de datos D. El algoritmo primero verifica si las variables referentes a la proyección en la consulta corresponden a variables presentes sólo en el antecedente de las DFs (función $SiVariablesSoloAntecedente()$). Si es el caso, las respuestas consistentes a la consulta se obtienen directamente de los conjuntos W^+ y W^u de la IWF de $\Pi(D, DF, \mathcal{Q})$ (función $ExtraerTuplasenW^+ yW^u()$). Si las variables proyectadas aparecen en el consecuente de las DFs, o corresponden a variables (atributos) simples (función $SiVariablesenConsecuenteoSimples()$), se realiza la reescritura de la consulta para obtener las respuestas utilizando ambos conjuntos W^+ y W^u de la IWF de los respectivos programas (función $ExtraerTuplasenW^+ yW^u Filtradas()$).

5. Conclusiones

Presentamos *CQA-WF* un sistema Web que permite computar respuestas consistentes a consultas conjuntivas con respecto a la semántica well-founded. Esta semántica ha sido utilizada como alternativa a la semántica de modelos estables para programas en lógica con negación. La interpretación well-founded de un programa puede ser computada en tiempo polinomial, por lo que su uso en el computo de consultas es relevante, dado el alto costo que conlleva computar respuestas consistentes a consultas.

CQA-WF es el primer sistema que implementa programas de reparación con semántica well-founded para obtener respuestas consistentes a consultas conjuntivas. En [10] se presenta un sistema más general que implementa programas de reparación con semántica de modelos estables. Este sistema permite obtener respuestas consistentes a consultas de primer orden evaluadas sobre bases de datos inconsistentes con respecto a restricciones de integridad universales, referenciales y de no-nulidad. Otros sistemas que permiten evaluar consultas

Algoritmo 1 Respuestas consistentes well-founded a consultas de la forma (3)

Entrada: Instancia D, conjunto DF de DFs, consulta $\mathcal{Q}$
Salida: Respuestas Consistentes a $\mathcal{Q}$
Variables: $VERDAD$, $RESPUESTAS$
  $VERDAD$=$SiVariablesSoloAntecedente(\mathcal{Q}$,DF)
  Si $VERDAD$ es $VERDADERO$ **Entonces**
    $RESPUESTAS$=$ExtraerTuplasenW^{+}yW^{u}(\mathcal{Q})$
  Sino
    $VERDAD$=$SiVariablesenConsecuenteoSimples(\mathcal{Q}$,DF)
    Si $VERDAD$ es $VERDADERO$ **Entonces**
      $RESPUESTAS$= $ExtraerTuplasenW^{+}yW^{u}Filtradas(\mathcal{Q})$
    Fin Si
  Fin Si
  Retornar $RESPUESTAS$

sobre bases de datos inconsistentes en tiempo polinomial son: (i) el sistema *Queca* [11] que implementa reescritura de consultas y que puede ser utilizado con restricciones de integridad universales y consultas conjuntivas sin proyección. (ii) El sistema *Hippo* [12] que implementa reescritura de consultas de primer orden basada en métodos de grafos. Este sistema también funciona para un conjunto restringido de restricciones de integridad y consultas conjuntivas libres de proyección. (iii) El sistema *ConQuer* [14] permite obtener respuestas consistentes con respecto a restricciones de llave primaria y una clase no trivial de consultas conjuntivas con proyección. Dejamos como trabajo futuro la comparación en términos de rendimiento entre alguno de los sistemas reportados y *CQA-WF*.

Se ha mostrado que la semántica well-founded se puede utilizar para computar una clase especial de consultas conjuntivas. Sin embargo, ésta podría ser utilizada para computar respuestas aproximadas a consultas más generales como se reporta en [9]. La implementación para este tipo de consultas se deja para trabajo futuro.

Referencias

[1] Marcelo Arenas, Leopoldo Bertossi, and Jan Chomicki. Consistent Query Answers in Inconsistent Databases. In *Proceedings of the ACM Symposium on Principles of Database Systems(PODS'99)*, pages 68–79, 1999.

[2] Marcelo Arenas, Leopoldo Bertossi, and Jan Chomicki. Answer Sets for Consistent Query Answering in Inconsistent Databases. *Theory and Practice of Logic Programming*, 3(4-5):393–424, 2003.

[3] Pablo Barceló and Leopoldo Bertossi. Logic Programs for Querying Inconsistent Databases. In *Proceedings of the International Symposium on Practical Aspects of Declarative Languages (PADL'03)*, Springer LNCS 2562, pages 208–222, 2003.

[4] Pablo Barceló, Leopoldo Bertossi, and Loreto Bravo. Characterizing and Computing Semantically Correct Answers from Databases with Annotated Logic and Answer Sets. In *Semantics in Databases*, Springer LNCS 2582, pages 1–27, 2003.

[5] Leopoldo Bertossi. Consistent Query Answering in Databases. *ACM Sigmod Record*, 35(2):68–76, 2006.

[6] Leopoldo Bertossi and Jan Chomicki. Query Answering in Inconsistent Databases. In *Logics for Emerging Applications of Databases*, Springer LNCS 1973, pages 43–83, 2003.

[7] Loreto Bravo and Leopoldo Bertossi. Consistent Query Answering under Inclusion Dependencies. In Hanan Lutfiyya, Janice Singer, and Darlene A. Stewart, editors, *14th Annual IBM Centers for Advanced Studies Conference (CASCON'04)*, pages 202–216. IBM, 2004.

[8] Loreto Bravo and Leopoldo Bertossi. Semantically Correct Query Answers in the Presence of Null Values. In *Proceedings of the EDBT WS on Inconsistency and Incompleteness in Databases (IIDB'06)*, Springer LNCS 4254, pages 336–357, 2006.

[9] Mónica Caniupán. *Optimizing And Implementing Repair Programs For Consistent Query Answering In Databases*. PhD thesis, School of Computer Science, Carleton University, 2007.

[10] Mónica Caniupán and Leopoldo E. Bertossi. The consistency extractor system: Answer set programs for consistent query answering in databases. *Data Knowl. Eng.*, 69(6):545–572, 2010.

[11] Alexander Celle and Leopoldo Bertossi. Querying Inconsistent Databases: Algorithms and Implementation. In *Proceedings of the First International Conference on Computational Logic*, pages 942–956. Springer-Verlag, 2000.

[12] Jan Chomicki and Jerzy Marcinkowski. Minimal-Change Integrity Maintenance using Tuple Deletions. *Information and Computation*, 197(1-2):90–121, 2005.

[13] Thomas Eiter, Michael Fink, Gianluigi Greco, and Domenico Lembo. Efficient Evaluation of Logic Programs for Querying Data Integration Systems. In *Proceedings of the International Conference on Logic Programming (ICLP'03)*, Springer LNCS 2916, pages 163–177, 2003.

[14] Ariel Fuxman, Elham Fazli, and Renée J. Miller. ConQuer: Efficient Management of Inconsistent Databases. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 155–166, 2005.

[15] Michael Gelfond and Vladimir Lifschitz. The Stable Model Semantics for Logic Programming. In *Proceedings of the International Conference on Logic Programming (ICLP'88)*, pages 1070–1080, 1988.

[16] Michael Gelfond and Vladimir Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9(3/4):365–386, 1991.

[17] Gianluigi Greco, Sergio Greco, and Ester Zumpano. A Logical Framework for Querying and Repairing Inconsistent Databases. *IEEE Transactions on Knowledge and Data Engineering*, 15(6):1389–1408, 2003.

[18] Brian Kernighan and Dennis Ritchie. *The C Programming Language*. Prentice Hall, Inc., 1988.

[19] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Christoph Koch, Cristinel Mateis, Simona Perri, and Francesco Scarcello. The DLV System for Knowledge Representation and Reasoning. *ACM Transactions on Computational Logic*, 7(3):499–562, 2006.

[20] Pasquale Rullo Nicola Leone and Francesco Scarcello. Disjunctive stable models: unfounded sets, fixpoint semantics, and computation. *Inf. Comput.*, 135:69–112, 1997.

[21] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[22] Teodor C. Przymusinski. Well-founded semantics coincides with three-valued stable semantics. *Fundam. Inf.*, 13:445–463, 1990.

[23] Teodor C. Przymusinski. Stable Semantics for Disjunctive Programs. *New Generation Computing*, 9:401–424, 1991.

[24] Allen Van Gelder, Kenneth Ross, and John S. Schlipf. Unfounded Sets and Well-Founded Semantics for General Logic Programs. In *Proceedings of the ACM Symposium on Principles of Database Systems(PODS'88)*, pages 221–230, 1988.