Universidad del Bío-Bío
Facultad de Ciencias Empresariales
Departamento de sistemas de información

Profesor guía:
Mónica Caniupán Marileo.

IMPLEMENTACIÓN DE DATA WAREHOUSE Y REPARACIÓN DE INCONSISTENCIAS

Informe Final de Habilitación Profesional

Fecha: 6 de Abril 2009

Nombre alumno:
Alvaro Placencia Monsalves.

Resumen

Un Data Warehouse es una colección de datos orientada a un determinado ámbito; integrando información desde diversas fuentes otorga la funcionalidad de ayudar a la toma de decisiones en la entidad en que se implementa. El almacenamiento de la información en un Data Warehouse no se usa en combinación con datos de uso actual, puesto que mantienen datos históricos con el propósito de generar información que sirva de apoyo en decisiones estratégicas.

La información en los sistemas Data Warehouse se organiza de acuerdo a dimensiones y hechos. Mientras que las dimensiones proveen la forma en que los datos se organizan, los hechos almacenan tuplas que pueden verse como puntos en espacios formados por las dimensiones. Una dimensión se modela por medio de una jerarquía de categorías.

Los sistemas Data Warehouse al igual que bases de datos tradicionales pueden presentar inconsistencias; en el caso particular de los primeros se manifiesta como inconsistencias a restricciones dimensionales. Estas restricciones dimensionales se representan por las denominadas restricciones de particionado, las cuales sirven para establecer que un elemento de una categoría dada se relacione con un único elemento de una categoría padre.

Los objetivos de este proyecto son la construcción de una aplicación que implemente un Data Warehouse con su respectiva organización, y que sobre las instancias dimensionales ofrezca la funcionalidad de comprobar consistencia y reparar mediante la eliminación e inserción de arcos entre elementos.

En el Capítulo 1 se presenta la introducción al desarrollo de la aplicación, sus respectivos objetivos, alcances, límites y formas de llevar a cabo su funcionalidad principal,

es decir, la detección y reparación de instancias inconsistentes respecto de restricciones dimensionales.

El Capítulo 2 explica el concepto de Data Warehouse exponiendo cómo se subdivide la información en unidades lógicas más pequeñas organizándose en dimensiones, categorías y hechos, considerando las restricciones dimensionales, relaciones roll-up y sumarizabilidad. Todo lo necesario para comprender de qué forma ha de realizarse una reparación queda presente en este capítulo.

Capítulo 3, mediante el uso de ejemplos se describe como se produce información errónea al momento de consultar instancias dimensionales inconsistentes; estos mismos ejemplos se usan para demostrar la forma de detectar y reparar las inconsistencias.

Capítulo 4, desarrolla una completa exposición del sistema encargado de reparar las instancias de dimensión implementando programas de reparación. En este capítulo se describe la arquitectura del sistema y los algoritmos.

El Capítulo 5 muestra la interfaz gráfica del sistema reparador de instancias dimensionales, así como también el manual de usuario complementado de un seguimiento y ejemplo de cómo operar el sistema.

Finalizando, el Capítulo 6 describe las conclusiones del proyecto.

Índice

Capítulo 1 – Introducción

En el mundo de las grandes organizaciones es de vital importancia mantener información sobre sus operaciones respaldada en bases de datos. Con el objetivo de ayudar en la toma de decisiones estas organizaciones han ido incorporando bases de datos más complejas que a su vez entregan, mediante el análisis de datos, información de mayor valor para evaluar decisiones estratégicas y pronósticos sobre comportamientos futuros. Esta funcionalidad la proveen los sistemas Data Warehouse que reuniendo información desde diversas bases de datos operacionales puede responder interrogantes mucho más elaboradas que consultas sobre una base de datos tradicional. La información contenida en los Data Warehouse, así como en todas las bases de datos, no está exenta de presentar inconsistencias, que pueden significar devolver datos incorrectos cuando se realice una consulta sobre sus tablas o computar pronósticos errados.

1.1 Orígenes

Un sistema Data Warehouse (DW) según define Bill Inmon [7] se caracteriza por ser:

- *Integrado*: los datos almacenados en el Data Warehouse deben integrarse en una estructura consistente, por lo que las inconsistencias existentes entre los diversos sistemas operacionales deben ser eliminadas.
- *Temático*: sólo los datos necesarios para el proceso de generación del conocimiento del negocio se integran desde el entorno operacional. Los datos se organizan por temas para facilitar su acceso y entendimiento por parte de los usuarios finales.
- *Histórico*: el tiempo es parte implícita de la información contenida en un Data Warehouse.
- *No volátil*: el almacén de información de un Data Warehouse existe para ser leído, y no modificado.

Ejemplo 1.1

La Figura 1.1 muestra la dimensión tiempo presente en todas las implementaciones de Data Warehouse. Esta dimensión representa un esquema dimensional homogéneo el cuál indica que todos los elementos de una categoría determinada deben tener arcos con el mismo conjunto de categorías.

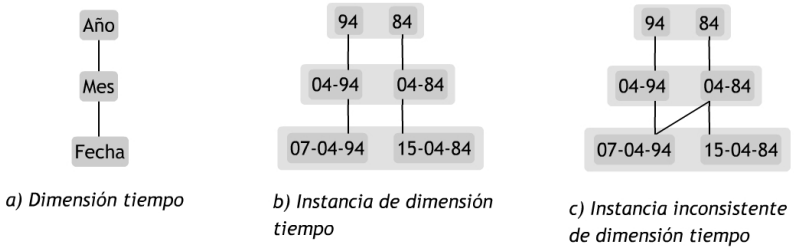


Figura 1.1 – Dimensión Tiempo

Para la Dimensión Tiempo las categorías son las siguientes: Fecha, Mes y Año (Figura 1.1 – a)).

En la Figura 1.1 b) es posible apreciar que los elementos de la instancia dimensional no presentan inconsistencias respecto de las restricciones de particionado, las cuales establecen que un elemento de una categoría hija no puede tener más de un arco con los elementos de una categoría antepasada, es decir, un elemento hijo debe tener un solo padre. La instancia dimensional de la Figura 1.1 c) es inconsistente respecto de las restricciones de particionado debido a que uno de los elementos en la categoría Fecha posee más de un arco asociándose con dos elementos de la categoría padre. □

Con el propósito de evitar que existan inconsistencias en un sistema DW se ha implementado un método basado en programación lógica para retornar la consistencia a sistemas que presenten alguna dimensión inconsistente respecto de restricciones de

particionado. Esta semántica de reparación motiva al desarrollo de una aplicación que administre sistemas DW e incorpore la detección y reparación de instancias dimensionales inconsistentes.

1.2 Objetivos

El objetivo principal se centra en el desarrollo de la aplicación que administre DWs permitiendo examinar las dimensiones de este y probando si sus instancias son consistentes respecto de restricciones de particionado. De descubrirse inconsistencias el sistema es capaz de mostrar las reparaciones posibles permitiendo al usuario seleccionar una de éstas y corregir definitivamente las inconsistencias.

Un objetivo secundario ha sido implementar el desarrollo de la aplicación sobre un lenguaje de programación con potencial WEB, para permitir su portabilidad y lograr que funcione sin inconvenientes en diferentes arquitecturas de computadores (ej. PC, Linux, MAC).

1.3 Alcances

La funcionalidad del software permite administrar DWs, detectar inconsistencias y realizar reparaciones a instancias dimensionales ingresadas al sistema. La administración contempla el ingreso, modificación y eliminación de dimensiones, categorías y elementos. Sobre una dimensión ingresada en la aplicación es posible visualizar las restricciones dimensionales que ejercen orden. Los programas de reparación detectan las inconsistencias y proponen alternativas para repararlas; luego el sistema transforma estas alternativas en reparaciones reales a realizarse sobre la instancia inconsistente, por último el usuario selecciona una de aquellas alternativas.

1.4 Límites

La aplicación contempla la detección y reparación sobre instancias de dimensión estrictamente homogéneas, esto quiere decir, que los elementos de cierta categoría deben tener antepasados en el mismo conjunto de categorías. De violarse esta restricción se está frente a instancias dimensionales heterogéneas a las cuáles el sistema no da soporte.

Capítulo 2 – Preliminares

2.1 Introducción a Data Warehouse

Un data warehouse es un repositorio de datos que integra y materializa información desde diferentes fuentes que surge como solución a las necesidades de información global de la empresa [7]; este repositorio o almacén de datos puede ser consultado desde sistemas OLAP (On-Line Analytical Processing). Por lo tanto, en una base de datos multidimensional destinada para procesamiento analítico en línea la información puede observarse como puntos en un espacio multidimensional. Específicamente los Data Warehouse principalmente se constituyen de dimensiones y hechos, las primeras reflejan la forma en que los datos se organizan, mientras que los hechos corresponden a datos cuantitativos relacionados con las dimensiones.

Usualmente, las dimensiones son consideradas la parte estática de los DWs , mientras que los hechos son considerados la parte dinámica, en el sentido que las operaciones de actualización afectan principalmente a las tablas de hechos. Sin embargo, en la práctica, la evolución de la información almacenada requiere la actualización de alguna de las dimensiones, por ejemplo, puede ocurrir que la estructura en una de esas fuentes cambie, o una nueva fuente se incorpore u otra se elimine. Alguno de estos cambios puede requerir actualizaciones a la estructura de las dimensiones agregando nuevas categorías, nuevos arcos entre ellas o nuevos elementos. También es importante señalar que las vistas multidimensionales son diseñadas según los requerimientos de los usuarios finales y una redefinición de estos puede causar una actualización dimensional [4].

Ejemplo 2.1

Una tienda de instrumentos musicales dispone de un completo catálogo de productos en cada una de las sucursales que mantiene a lo largo del país y debido a la necesidad de conocer la cantidad de ventas en cada una de ellas ha dispuesto de un DW denominado 'DW-Ventas' para generar reportes y analizar las ventas con el propósito de pronosticar comportamientos a futuro o la necesidad de abrir nuevas tiendas en una región. El repositorio considera las dimensiones *Tiempo* (a) *Ubicación* (b) y *Producto* (d) que muestra la Figura 2.1. La dimensión *Producto* representa cómo se organizan los productos dentro de cada tienda. *Producto* es el artículo en particular, por ejemplo: Stratocaster, JazzBass, Yamaha-S, SX-12, etc. *Instrumento* dice relación con el tipo de artículo, es decir si este es Guitarra, Bajo, Batería, Saxofón, Sintetizador entre otros. *Familia* corresponde a la clasificación de los productos por su funcionamiento y forma de emitir los sonidos (Aerófono, Cordófono, Membranófono, Idiófono y Electrófono). Por último Tipo representa la clasificación clásica de los instrumentos musicales: Cuerda, Viento y Percusión.

En la dimensión Producto cada Producto se asocia con una Familia que a su vez se relaciona con un Tipo, por otra parte cada Producto se relaciona con un Instrumento que termina por asociarse a un Tipo, por último cada Tipo debe asociarse con la categoría All que se encarga de mantenerlos a todos unidos.

□

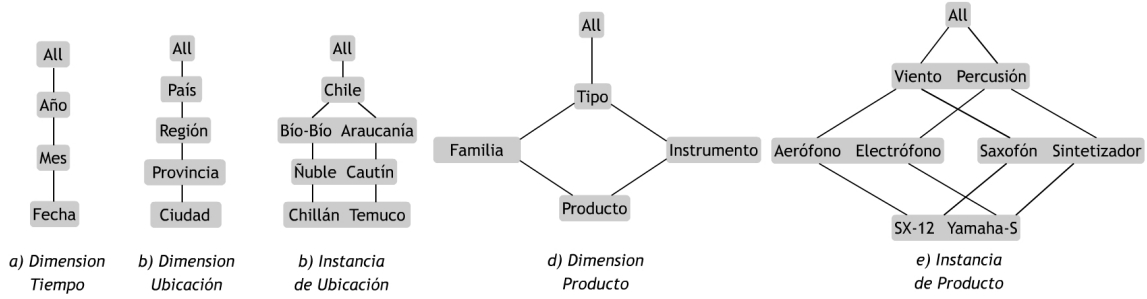


Figura 2.1 – Dimensiones Tiempo, Ubicación y Producto

Una tabla de hechos es una tabla que permite guardar datos cuantitativos asociados a las dimensiones, tales como:

- Medidas del proceso / actividad / flujo de trabajo / evento que se pretende modelar.

Estas tablas de hechos representan eventos que suceden en un determinado espacio tiempo y se caracterizan por permitir analizar los datos con el máximo de detalle.

Una tabla de hechos que relacione datos de las dimensiones del ejemplo en la Figura 2.1 podría ser la siguiente.

Ventas			
Fecha	**Producto**	**Ciudad**	**Cantidad**
01-01-09	SX-12	Chillán	2
01-01-09	Yamaha-S	Chillán	4
02-01-09	SX-12	Temuco	3
02-01-09	Yamaha-S	Temuco	7

En la tabla anterior las columnas *Producto* y *Ciudad* son las categorías base de las dimensiones relacionadas por el Data Warehouse, estas permiten visualizar con el mayor de detalle los datos organizados por dimensión. También es posible generar columnas de otras categorías de las dimensiones Producto y Ubicación. La columna Cantidad es una medida que provee la información, por ejemplo, de cuántos Productos SX-12 se vendieron en Chillán el 01-01-09.

Con la información que provee la tabla es posible responder a preguntas como: ¿Cuántos sintetizadores se vendieron en la región del Bío-Bío el 01-01-09?

2.2 Modelo de datos multidimensional

Una dimensión es un grafo direccionado acíclico que conforma un esquema $S = (C, \nearrow)$, donde C es un conjunto de categorías, y $\nearrow$ es la relación hijo/padre entre ellas, es decir, arcos en el grafo. Para un par de categorías $C_1, C_2 \in C$, se escribe $C_1 \nearrow C_2$ para denotar que C_1, C_2 forman un arco en S. $\nearrow^*$ es la clausura reflexiva y transitiva de $\nearrow$. Por simplicidad las categorías no tienen atributos y, por razones técnicas, hay una categoría superior distinguida llamada All cuyo único elemento es $\{all\}$ que es alcanzable desde todos los otros miembros en las categorías restantes vía $\nearrow^*$. La categoría en el nivel más bajo es llamada categoría inferior.

Ejemplo 2.2

La dimensión Producto representada en la Figura 2.1 d) se define por:

- Un conjunto de categorías $C = \{Producto, Familia, Instrumento, Tipo, All\}$
- Las relaciones hijo/padre $\nearrow = \{(Producto, Familia), (Producto, Instrumento),$
  $(Familia, Tipo), (Instrumento, Tipo), (Tipo, All)\}$
- $\nearrow^* = \nearrow \cup \{(Producto, Producto), (Tipo, Tipo), (Producto, Familia), ...\}$

La categoría inferior es $Producto$. $Familia$ e $Instrumento$ son antepasados directos de $Producto$. $Tipo$ y All son antepasados indirectos. □

2.3 Instancia dimensional

Una instancia dimensional se obtiene por la especificación de un conjunto de miembros por cada categoría y una relación hijo/padre entre ellos.

Una instancia sobre un esquema de dimensión $S = (C, \nearrow)$ es una tupla $(M, \angle)$ en la cual M es una colección de átomos de la forma C(a) dónde C$\in C$ y a es una constante; puntualmente C es una categoría y a es uno de sus elementos los cuales por regla deben ser distintos unos de otros. La relación $\angle$ contiene las relaciones hijo/padre entre elementos de categorías diferentes, si tenemos $a \angle b$ entonces $C_1 \nearrow C_2$ y $\mathrm{C}_1(a), \mathrm{C}_2(b)$. Denotamos con $\angle^*$ la clausura reflexiva y transitiva de $\angle$, entonces $a \angle^* b$.

Ejemplo 2.3

Instancia del esquema jerárquico Instrumentos musicales (Figura 2.3):

- $M = \{Producto(JazzBass), Producto(PrecisionBass), Producto(TB-A),$ $Familia(Cuerdófono), Familia(Membranófono), Instrumento(Bajo),$ $Instrumento(Timbal), Tipo(Cuerdas), Tipo(Percusión), All(all)\}$
- $\angle = \{(JazzBass, Cuerdófono), (PrecisionBass, Cuerdófono), (TB-A,$ $Membranófono), (JazzBass, Bajo), (PrecisionBass, Bajo), (TB-A,$ $Timbal), (Cordófono, Cuerdas), (Membranófono, Percusión), (Bajo, Cuerdas),$ $(Timbal, Percusión), (Cuerdas, all), (Percusión, all)\}$
- $\angle^* = \{(JazzBass, JazzBass), (TB-A, TB-A), (TB-A, Timbal).....\}$

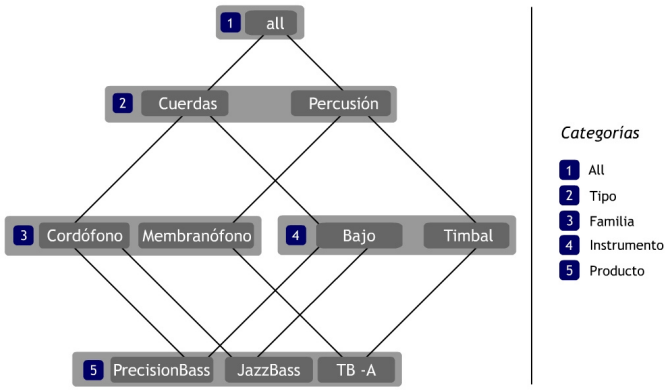


Figura 2.2 – Instancia de dimensión Producto □

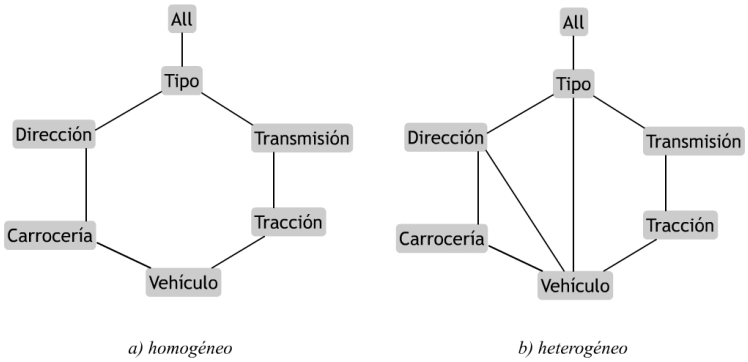
2.4 Esquemas dimensionales homogéneos y heterogéneos

El esquema dimensional homogéneo posee la restricción de que todos los arcos entre categorías son obligatorios, si se libera esta restricción, el esquema se denomina heterogéneo, el cual tiene la libertad de que los elementos tengan antepasados en distintas categorías. Los esquemas dimensionales heterogéneos son menos restrictivos y permiten una representación más natural, precisamente como son las situaciones reales. Además, en ciertos casos usan una menor cantidad de categorías para realizar un modelamiento dimensional.

Ejemplo 2.4

En la Figura 2.3 se muestran los posibles esquemas para una misma dimensión. El esquema a) de la dimensión Vehículos es homogéneo debido a que cada uno de los arcos en el esquema es obligatorio para cada uno de los elementos en las categorías, es decir, todos los elementos de la categoría *Vehículo* individualmente tienen un arco a un elemento en *Carrocería*, luego todos los elementos de *Carrocería* poseen un arco a un elemento en *Dirección*, los elementos en *Dirección* tienen un arco a un elemento en la categoría *Tipo* y así para todos los elementos en las otras categorías relacionadas por un arco. El esquema b) a diferencia del primero posee dos arcos adicionales, uno entre *Vehículo* y *Dirección* y otro

entre *Vehículo* y *Tipo*. Los elementos en la categoría *Vehículos* pueden tener directamente un arco a *Dirección* o a *Tipo* sin tener antepasados en *Carrocería*.



a) homogéneo *b) heterogéneo*

Figura 2.3 – Tipos de esquemas dimensionales

□

Con el propósito de simplificar la explicación de las soluciones a los ejemplos, los capítulos siguientes así como hasta ahora solo se trabajará con esquemas dimensionales homogéneos.

2.5 Relación Roll-up

Formalmente una relación roll-up entre elementos de dos categorías C_i y C_j se define por :

$$R_{C_i}^{C_j}(D) = \{(x, y) \mid C_i(x) \wedge C_j(y) \wedge x \angle^* y\}$$

donde D es la instancia dimensional sobre la cual se aplica la relación roll-up.

Las relaciones roll-up son fundamentales para computar agregación de datos; para garantizar que éstas sean funcionales las instancias dimensionales deben satisfacer un conjunto de restricciones de particionado.

Ejemplo 2.5

Las siguientes son relaciones roll-up entre las categorías en la Figura 2.2:

$$R^{Tipo}_{Producto}(D) = \{(PrecisionBass, Cuerdas), (JazzBass, Cuerdas), (TB-A, Percusión)\}$$

$$R^{Familia}_{Producto}(D) = \{(JazzBass, Cordófono)(PrecisionBass, Cordófono), (TB-A, Membranófono)\}$$

$$R^{Instrumento}_{Producto}(D) = \{(JazzBass, Bajo), (PrecisionBass, Bajo), (TB-A, Timbal)\}$$

$$R^{Tipo}_{Familia}(D) = \{(Cuerdófono, Cuerdas), (Membranófono, Percusión)\}$$

$$R^{Tipo}_{Instrumento}(D) = \{(Bajo, Cuerdas), (Timbal, Percusión)\}$$

□

2.6 Restricciones dimensionales

Las características que definen si un esquema es homogéneo o heterogéneo es posible especificarlas por medio de restricciones dimensionales.

Una ***restricción de particionado*** se define por:

$$\forall xyz(C_i(x) \wedge C_j(y) \wedge C_j(z) \wedge x \angle^* y \wedge x \angle^* z) \rightarrow (y = z))$$

Esta establece que un elemento de una categoría debe asociarse mediante un arco con un único elemento en una categoría antepasada, de aquí que la restricción el elemento *y* debe ser igual a *z*.

Sobre la instancia dimensional de la Figura 2.2, una ***restricción de particionado*** sería:

$$\forall xyz((Producto(x) \wedge Tipo(y) \wedge Tipo(z) \wedge x \angle^* y \wedge x \angle^* z) \rightarrow (y = z))$$

esta establece que cada miembro de la categoría *Producto* se asocia con un único miembro en la categoría *Tipo*, sin importar las categorías intermedias en las que cada elemento de Producto tenga antepasados, las que particularmente para la instancia dimensional de la Figura 2.2 son las categorías *Familia* e *Instrumento*.

Las restricciones dimensionales son una combinación boleana de dos tipos de átomos: átomos de camino y átomos de igualdad.

Por ejemplo, para la Figura 2.2

- Átomo de igualdad : $\exists x(Producto(x) \wedge x = JazzBass)$
- Átomo de camino : $\forall x(Producto(x) \rightarrow \exists y(Intrumento(y) \wedge x \angle y))$

Los átomos de igualdad establecen la existencia de ciertos elementos en determinadas categorías.

Los átomos de camino permiten especificar las condiciones que ejerce el esquema jerárquico sobre la instancia dimensional, por lo tanto, sirven para modelar el camino que deben seguir los elementos desde la categoría inferior hasta la categoría superior.

Considerando la dimensión *Producto* representada en la Figura 2.2, las siguientes son las restricciones de camino que establecen las vías obligatorias para los elementos de las distintas categorías.

- $\forall x(Producto(x) \rightarrow \exists y(Intrumento(y) \wedge x \angle y))$
- $\forall x(Producto(x) \rightarrow \exists w(Familia(w) \wedge x \angle w))$
- $\forall y(Instrumento(y) \rightarrow \exists z(Tipo(z) \wedge y \angle z))$
- $\forall w(Familia(w) \rightarrow \exists z(Tipo(z) \wedge w \angle z))$

- $\forall x(Producto(x) \rightarrow \exists z(Tipo(z) \wedge x \angle z))$
- $\forall z(Tipo(z) \rightarrow \exists q(All(q) \wedge z \angle q))$
- $\forall x(Producto(x) \rightarrow \exists q(All(q) \wedge x \angle q))$
- $\forall y(Instrumento(y) \rightarrow \exists q(All(q) \wedge y \angle q))$
- $\forall w(Familia(w) \rightarrow \exists q(All(q) \wedge w \angle q))$

2.7 Sumarizabilidad

Una estrategia clave para acelerar el procesamiento de los consultas de agregación es la reutilización de otras consultas de agregación que hayan sido computadas y materializadas anteriormente. Esto se logra reescribiendo una consulta de agregación como una nueva consulta que haga referencia a otra consulta ya procesada y cuyos resultados han sido materializados. En el mundo OLAP encontrar estas reescrituras se conoce como *navegación de agregación* y representa al concepto de sumarizabilidad.

Consultas de agregación

En DWs las consultas de agregación son computadas desde instancias dimensionales a través de las relaciones roll-up y las tablas de hechos. Estas relaciones son tratadas como tablas relacionales. Por ejemplo, la relación roll-up $R_{Producto}^{Instrumento}$ puede verse como una tabla relacional de la forma $R(Producto, Instrumento)$.

Se usarán consultas de agregación con declaraciones group-by, de la siguiente forma:

$$\begin{aligned} &SELECT\ A_j, ..., A_m, f(A) \\ &FROM\ T, R_i, ..., R_m \\ &WHERE\ condiciones \\ &GROUP\ BY\ A_J, ..., A_n \end{aligned}$$

donde $A_j,....,A_n$ son atributos de la tabla de hechos T o de las funciones roll-up $R_i,....,R_m$ y f es $min, max, count, sum, avg$ aplicado al atributo A con $A \cap \{A_j,....,A_n\} = 0$.

Ejemplo 2.6

Considere la relación roll-up sobre la instancia dimensional de la Figura 2.2 :

$$R^{Tipo}_{Producto}(D) = \{(JazzBass, Cuerdas), (PrecisionBass, Cuerdas), (TB-A, Percusión)\}$$

Se puede, mediante la propiedad sumarizabilidad *(SUMM)*, computar esta relación roll-up usando la categoría intermedia *Instrumento*, esto porque todo elemento de la categoría *Producto* posee un arco a un elemento en *Instrumento* y luego a *Tipo* desde *Instrumento* (Figura 2.2). Si se tiene dos vistas materializadas con las relaciones *Producto-Instrumento* e *Instrumento-Tipo* , aplicando SUMM se obtiene:

$SUMM_A$
$R^{Tipo}_{Producto}(D) = R^{Instrumento}_{Producto}(D) \textbf{\textit{join}} R^{Tipo}_{Instrumento}(D)$

desde las relaciones roll-up materializadas:

- $R^{Instrumento}_{Producto}(D) = \{(JazzBass, Bajo), (PrecisionBass, Bajo), (TB-A, Timbal)\}$
- $R^{Tipo}_{Instrumento}(D) = \{(Bajo, Cuerdas), (Timbal, Percusión)\}$

Ahora, sobre este mismo esquema de dimensión representado por la Figura 2.1 d), si se tuviera materializada las relaciones *Producto-Familia* y *Familia-Tipo* de la instancia en

la Figura 2.2 se obtendría el mismo resultado que la intersección *Producto-Instrumento* e *Instrumento-Tipo* como se vio anteriormente.

$SUMM_B$
$R^{Tipo}_{Producto}(D) = R^{Familia}_{Producto}(D)$ ***join*** $R^{Tipo}_{Familia}(D)$

desde las relaciones roll-up materializadas:

- $R^{Familia}_{Producto}(D) = \{(PrecisionBass, Cuerdófono), (JazzBass, Cuerdófono), (TB-A, Membranófono)\}$
- $R^{Tipo}_{Familia}(D) = \{(Cuerdófono, Cuerdas), (Membranófono, Percusión)\}$

Por conclusión se tiene que $SUMM_A$ y $SUMM_B$ son equivalentes.

□

Capítulo 3 – Consistencia y reparaciones

3.1 Consistencia

Una instancia dimensional es consistente si satisface las restricciones de particionado sobre la dimensión. Por lo tanto, si una instancia no satisface alguna de las restricciones de particionado se denomina *inconsistente.*

Ejemplo 3.1

Las restricciones de particionado sobre el esquema de la dimensión Producto representada en la Figura 2.1 d) son:

1. $\forall xyz((Producto(x) \wedge Instrumento(y) \wedge Instrumento(z) \wedge x\angle^* y \wedge x\angle^* z) \rightarrow (y = z))$
2. $\forall xyz((Producto(x) \wedge Familia(y) \wedge Familia(z) \wedge x\angle^* y \wedge x\angle^* z) \rightarrow (y = z))$
3. $\forall xyz((Producto(x) \wedge Tipo(y) \wedge Tipo(z) \wedge x\angle^* y \wedge x\angle^* z) \rightarrow (y = z))$
4. $\forall xyz((Producto(x) \wedge All(y) \wedge All(z) \wedge x\angle^* y \wedge x\angle^* z) \rightarrow (y = z))$
5. $\forall xyz((Instrumento(x) \wedge Tipo(y) \wedge Tipo(z) \wedge x\angle^* y \wedge x\angle^* z) \rightarrow (y = z))$
6. $\forall xyz((Instrumento(x) \wedge All(y) \wedge All(z) \wedge x\angle^* y \wedge x\angle^* z) \rightarrow (y = z))$
7. $\forall xyz((Familia(x) \wedge Tipo(y) \wedge Tipo(z) \wedge x\angle^* y \wedge x\angle^* z) \rightarrow (y = z))$
8. $\forall xyz((Familia(x) \wedge All(y) \wedge All(z) \wedge x\angle^* y \wedge x\angle^* z) \rightarrow (y = z))$
9. $\forall xyz((Tipo(x) \wedge All(y) \wedge All(z) \wedge x\angle^* y \wedge x\angle^* z) \rightarrow (y = z))$

Estas restricciones establecen que a través de arcos directos o transitivamente los elementos de cada una de las categorías deben asociarse solamente con un elemento en una categoría antepasada.

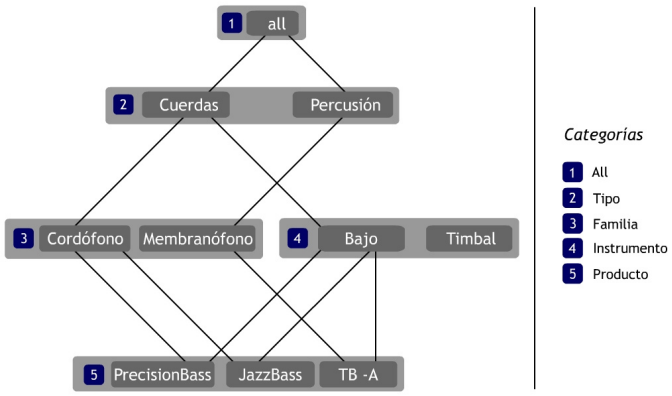


Figura 3.1 – Instancia dimensional Producto inconsistente

La instancia de la Figura 3.1 es inconsistente respecto a la restricción de particionado Nº 3 presentada anteriormente debido a que el elemento $TB-A$ posee un roll-up a *Percusión* vía *Membranófono* en *Familia* y también a *Cuerdas* vía *Bajo* en *Instrumento*.

Como consecuencia de lo anterior la siguiente relación roll-up no es funcional:

$$R^{Tipo}_{Producto}(D) = \{(JazzBass, Cuerdas), (PrecisionBass, Cuerdas), (TB\text{-}A, Percusión), (TB-A, Cuerdas)\}$$

Ahora, para exponer las consecuencias de tener instancias dimensionales inconsistentes se considerará la siguiente tabla de hechos:

Ventas		
Fecha	**Producto**	**Total**
01-01-09	PrecisionBass	100.000
01-01-09	JazzBass	100.000
01-01-09	TB-A	200.000

Considerando las tuplas de la tabla de hechos *Ventas*, la relación roll-up materializada $R^{Tipo}_{Producto}$ representada por la tabla relacional $R(Producto, Tipo)$ sobre la instancia dimensional inconsistente de la Figura 3.1 y la siguiente consulta de agregación que pregunta por el total de ventas para cada uno de los elementos en Producto agrupándolos por Tipo se tiene:

$$
\begin{aligned}
&SELECT\ \ R.Tipo, SUM(V.Total)\\
&FROM\ \ Ventas\ V, R\\
&WHERE\ \ R.Producto = V.Producto\\
&GROUP\ \ BY\ \ R.Tipo
\end{aligned}
$$

R	
Producto	**Tipo**
PrecisionBass	Cuerdas
JazzBass	Cuerdas
TB-A	Percusión
TB-A	Cuerdas

Las respuestas a la consulta son $(Cuerdas, 400.000)$, $(Percusión, 200.000)$. Los resultados presentan una anomalía debido a que la respuesta $(Cuerdas, 400.000)$ considera 3 productos: *PrecisionBass*, *JazzBass* y *TB-A* , estos con los totales de 100.000, 100.000 y 200.000 respectivamente, lo que suma la cifra de 400.000; aquí se aprecia claramente un error debido a que el elemento *TB-A* se está contando dos veces, relacionándose en *Tipo* tanto con el elemento *Percusión* como con *Cuerdas.*

La consulta materializada $R^{Tipo}_{Producto}$ se obtuvo mediante la propiedad de sumarizabilidad sobre la instancia dimensional inconsistente (Figura 3.1), de cualquier forma en que se haya obtenido $R^{Tipo}_{Producto}$, sobre la instancia inconsistente las relaciones roll-up $R^{Familia}_{Producto}$, $R^{Tipo}_{Producto}$, $R^{Tipo}_{Instrumento}$ y $R^{Tipo}_{Familia}$ presentarán alguna inconsistencia.

□

3.2 Reparación de instancias dimensionales

Si se tiene una instancia de dimensión inconsistente, entonces es lógico pensar en reparar o restaurar la consistencia de esta instancia. Para ello, es posible implementar una semántica de reparación, la cual es presentada en el artículo “Consistent Query Answering in Data Warehouses” [3].

La consistencia se restaura eliminando e insertando arcos entre elementos de distintas categorías, relacionadas por un arco en el esquema de dimensión, de tal manera que se reestablece la consistencia de la instancia.

Una ***reparación minimal*** es aquella que reestablece la consistencia con el menor número de eliminaciones e inserciones de arcos entre elementos.

Ejemplo 3.2

Gráficamente las reparaciones a la instancia dimensional inconsistente de la Figura 3.1 consistirían en la eliminación de uno de los arcos que entran en conflicto seguido de la inserción de otro arco que relacione elementos distintos entre las mismas categorías.

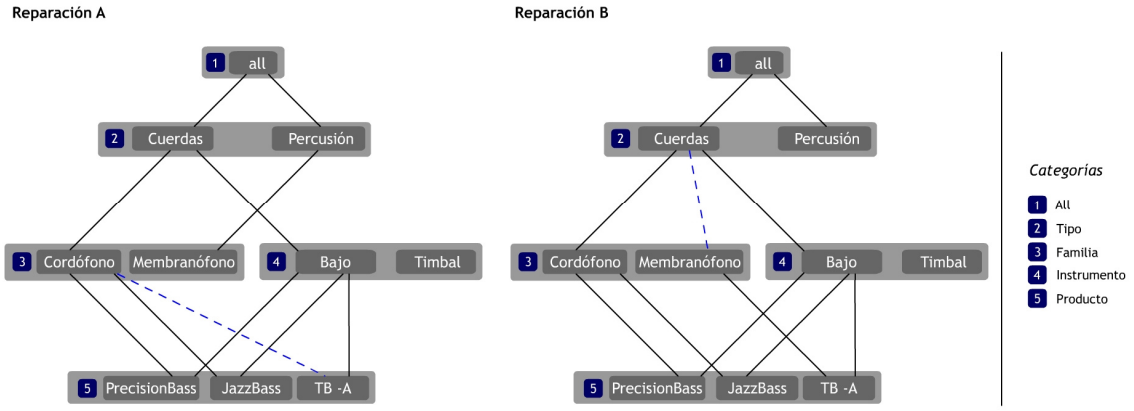


Figura 3.2 – Reparaciones sobre instancia dimensional inconsistente.

Las reparaciones de la Figura 3.2 se obtienen realizando una eliminación y una inserción.

- Reparación A : Eliminación del arco (TB-A,Membranófono) e Inserción del arco (TB-A,Cordófono).
- Reparación B : Eliminación del arco (Membranófono,Percusión) e Inserción del arco (Membranófono, Cuerdas).

De todas las reparaciones posibles sobre la instancia dimensional inconsistente de la Figura 3.1, las reparaciones A y B son mínimas debido a que la eliminación e inserción de una sola relación entre elementos de distintas categorías retorna la consistencia a la instancia realizando el número mínimo de cambios posible, es decir, A y B son las reparaciones que menos cambios realizan sobre la instancia dimensional original entre otras posibles.

La Figura 3.3 no es una reparación minimal, esto ya que se obtiene eliminando dos arcos e insertando otros dos, esto genera una instancia consistente pero alterando demasiado la información lo cual puede significar nefastas consecuencias al momento de obtener informes sobre ventas.

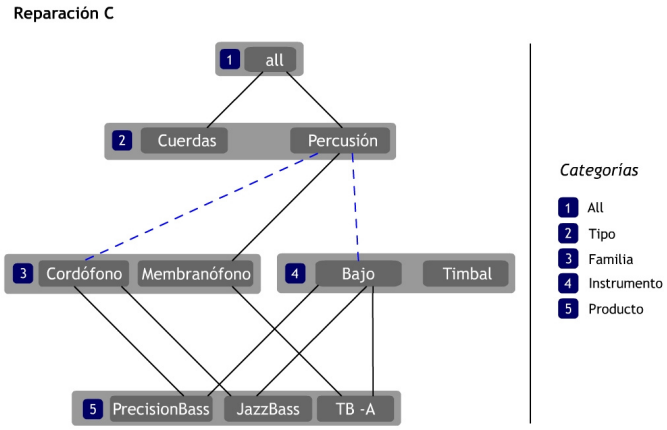


Figura 3.3 – Reparación C no mínima.

□

3.3 Programas de reparación

Las reparaciones de una instancia inconsistente de dimensión pueden implementarse por programas en lógica. Por lo tanto, las reparaciones minimales es posible especificarlas por medio de un programa Datalog con negación que busca los mejores modelos para reparar la instancia inconsistente.

Los átomos usados en el programa son los siguientes:

Átomo	***Significado***
• $C(a)$	El elemento a pertenece a una categoría C.
• $R(a,b,C_1,C_2)$	Representa un arco entre el elemento a perteneciente a la categoría C_1 y el elemento b perteneciente a la categoría C_2.
• $R'(a,b,C_1,C_2)$	Representa un arco insertado producto de la reparación.

• $RT'(a,b,C_1,C_2)$	Clausura Transitiva de R'
• $Ins(a,b,C_1,C_2)$	Es el arco (a,b) con $C_1(a)$ y $C_2(b)$ que se inserta cuando se repara una instancia.
• $Del(a,b,C_1,C_2)$	Es el arco (a,b) con $C_1(a)$ y $C_2(b)$ eliminado cuando se repara una instancia.

Tabla Elementos del programa de reparación. □

El programa de reparación hace uso de dos constructores especiales:

- Operador ***choice*** [8].
- Restricciones débiles (***weak constraint***) [9].

Un programa de reparación para una instancia dimensional $D(M,\angle)$ definida sobre un esquema de dimensión $S=(C,\nearrow)$ contiene las siguientes reglas:

Reglas
1. $C(a)$. Por cada categoría $C \in C$ y cada elemento perteneciente a C.
2. $R(a,b,C(a),C(b))$. Por cada arco existente en la instancia dimensional.
3. $R'(x,y',C_i,C_j) \leftarrow R(x,y,C_i,C_j), x \neq y,\ C_j(y').\ choice((x,C_j)(y'))$.
4. $RT'(x,y,n_1,n_2) \leftarrow R'(x,y,n_1,n_2)$
5. $RT'(x,z,n_1,n_3) \leftarrow R'(x,y,n_1,n_2), R'(y,z,n_2,n_3)$
6. $\leftarrow RT'(x,y,n_1,n_2), RT'(x,z,n_1,n_2), y \neq z$
7. $Ins(x,y,n_1,n_2) \leftarrow R'(x,y,n_1,n_2), not\ R(x,y,n_1,n_2)$
8. $Del(x,y,n_1,n_2) \leftarrow R(x,y,n_1,n_2), not\ R'(x,y,n_1,n_2)$
9. $\Leftarrow Ins(x,y,n_1,n_2)[1:1]$

10. $\Leftarrow Del(x, y, n_1, n_2)[1:1]$

1. La regla en (1) obtiene todos los átomos presentes en la instancia de dimensión $(M, \angle)$.
2. Esta regla se encarga de mostrar cada relación roll-up existente entre los elementos de la instancia.
3. Esta regla establece que debe R' almacena una nueva relación roll-up en la cual los elementos de la categoría hijo C_i son asociados con un nuevo elemento en la categoría padre C_j. Este nuevo elemento se obtiene usando el operador ***choice*** que elige aleatoriamente un elemento del dominio C_j. El operador *choice* puede ser sustituido por reglas tradicionales de Datalog bajo la semántica de modelo estable mediante el reemplazo de cada una de las reglas en el punto 3 por las siguientes:
   - $R'(x, y', C_i, C_j) \leftarrow R(x, y, C_i, C_j), x \neq y, C_j(y'), chosen_{C_j}(x, C_j, y').$
   - $chosen_{C_j}(x, C_j, y') \leftarrow R(x, y, C_i, C_j), x \neq y, C_j(y'), not\ diffChoice(x, C_j, y').$
   - $diffChoice_{C_j}(x, C_j, y') \leftarrow chosen_{C_j}(x, C_j, y'), C_j(y), y \neq y'$
4. y 5. Computa las relaciones roll-up transitivas entre categorías hijas y sus antepasadas.
6. Esta regla (sin cabeza) establece que no puede existir un elemento en una categoría n_1 asociado a más de un elemento en una categoría padre n_2 (restricción de particionado).
7. Establece la inserción de un arco que no existe actualmente en la instancia.
8. Establece la eliminación de un arco que existe en la instancia original.
9. y 10. Son restricciones débiles de la forma $\varphi\ [peso : nivel]$. Si φ es verdadera en el modelo, la restricción débil es violada y el peso aumenta al costo del nivel. Los modelos estables que minimizan la suma de los pesos de las restricciones débiles violadas se denominan los 'mejores modelos' del programa. Ambas restricciones débiles del programa de reparación poseen un nivel=1 y un peso=1, es decir, al

momento de reparar la instancia ambas poseen el mismo nivel de prioridad, esto significa que insertar y eliminar tienen el mismo costo.

Ejemplo 3.3

El programa de reparación para la instancia dimensional inconsistente representada por la Figura 3.1 es el siguiente:

$$R'(x, y', z, Familia) \leftarrow R(x, y, z, Familia), Familia(y'), choice((x, Familia)(y')), x \neq y.$$

$$R'(x, y', z, Instrumento) \leftarrow R(x, y, z, Instrumento), Instrumento(y'), choice((x, Instrumento)(y')), x \neq y.$$

$$R'(x, y', z, Tipo) \leftarrow R(x, y, z, Tipo), Tipo(y'), choice((x, Tipo)(y')), x \neq y.$$

$$RT'(x, y, n_1, n_2) \leftarrow R'(x, y, n_1, n_2).$$

$$RT'(x, z, n_1, n_3) \leftarrow R'(x, y, n_1, n_2), R'(y, z, n_2, n_3).$$

$$\leftarrow RT'(x, y, n_1, n_2), RT'(x, z, n_1, n_2), y \neq z.$$

$$Ins(x, y, n_1, n_2) \leftarrow R'(x, y, n_1, n_2), not\ R(x, y, n_1, n_2).$$

$$Del(x, y, n_1, n_2) \leftarrow R(x, y, n_1, n_2), not\ R'(x, y, n_1, n_2).$$

$$\Leftarrow Ins(x, y, n_1, n_2)[1:1].$$

$$\Leftarrow Del(x, y, n_1, n_2)[1:1].$$

Una vez computado este programa de reparación se obtienen dos mejores modelos:

$$M_1 = \{Del(TB-A, Membranófono, Producto, Familia), Ins(TB-A, Cordófono, Froducto, Familia)\}$$
$$Cost([peso:nivel]) :< [2:1] >$$

$$M_2 = \{Del(Membranófono, Percusión, Familia, Tipo), Ins(Membranófono, Cuerdas, Familia, Tipo)\}$$
$$Cost([peso:nivel]) :< [2:1] >$$

Estos modelos representan las reparaciones minimales posibles de realizarse sobre la instancia dimensional *Producto* inconsistente. Es importante señalar que estos modelos están restringidos a los átomos Ins y Del representados por las restricciones débiles presentadas anteriormente; es decir, son un extracto del modelo estable completo, sólo muestra cómo se reestablece la consistencia insertando un arco (Ins) y eliminando otro (Del).

En el siguiente Capítulo se comenzará a desarrollar los tópicos sobre la forma en que la aplicación se encarga de resolver los problemas de inconsistencias. Primero se analizará la arquitectura del sistema, luego los algoritmos para finalizar con la exploración de los menús y pantallas.

Capítulo 4 – Arquitectura

La aplicación encargada de reparar físicamente una instancia de dimensión inconsistente en un DW es un sistema denominado *Data Warehouse* ***fixer*** (Reparador de DW) el cuál administra dimensiones, sus respectivas categorías, arcos entre categorías, elementos de cada categoría y arcos entre elementos. Además de administrar completamente un DW el sistema incorpora la funcionalidad para detectar y reparar instancias dimensionales que violen alguna restricción de particionado.

El sistema es una aplicación WEB desarrollada en los lenguajes HTML[1] y PHP[2] versión 5.2.8 que interactúan con el motor de base de datos MySQL[3] 5.1.30 instalado sobre un servidor Apache versión 2.2.10. Los programas de reparación expuestos en el Capítulo 3 se ejecutan en DLV versión [build BEN/Oct 11 2007 gcc 3.4.5 (mingw special)] que obtiene los modelos estables reducidos a nivel de inserción y eliminación de arcos (Best models), con los cuales es posible implementar una reparación en el DW por medio de PHP.

La aplicación *Data Warehouse fixer* ha sido desarrollada en un lenguaje de programación potenciado para servir como sistema WEB, por lo tanto, funciona instalado sobre un servidor proveído por WAMPSERVER 2.0 en su versión para Windows XP que además de servidor contiene aplicaciones de administración tales como PHPmyadmin y SQLlite Manager que facilitan el trabajo de sistemas WEB. *Data Warehouse fixer* instalado sobre Linux (distribución Ubuntu 8.04) funciona con los mismos parámetros, pero esta vez, con otro paquete de aplicaciones denominado XAMPP 1.6.2 que provee un servidor Apache, y las aplicaciones de administración antes señaladas.

[1] HyperText Markup Language (lenguaje de marcas de hypertexto)
[2] **P**HP **H**ypertext **P**re-processor
[3] Sistema de gestión de base de datos relacional, multihilo y multiusuario.

Además de las tecnologías descritas, la aplicación en gran parte de sus módulos funciona validando el ingreso de información y redireccionando páginas mediante JavaScript, el cual es un lenguaje de programación interpretado que no necesita compilación.

Data Warehouse fixer fue programado usando el editor de aplicaciones WEB Dreamweaver MX 2004 instalado en Windows Xp sobre los siguientes componentes de hardware: Procesador Intel Pentium Dual Core 1.83 Ghz, 2 Gb de memoria ram, Disco duro de 250 Gb a 5400 rpm; para su instalación sólo necesita de 5 mb de espacio en disco. Para su óptimo funcionamiento requiere de al menos los siguientes componentes de hardware:

- Procesador de sexta generación (Dos núcleos de ejecución).
- 1 Gb de memoria ram.
- Disco duro con 5400 rpm de velocidad.

El modelo de datos sobre el cual se implementa la aplicación se compone de una base de datos relacional bajo un esquema no tradicional en la implementación de DW; esto es, una base de datos con tablas normalizadas sin una tabla central de hechos que represente la interacción de todas las dimensiones sobre el DW.

4.1 Arquitectura del sistema

La arquitectura del sistema se compone de siete módulos que interactúan entre si además de interactuar con el usuario y con sistemas externos a la aplicación.

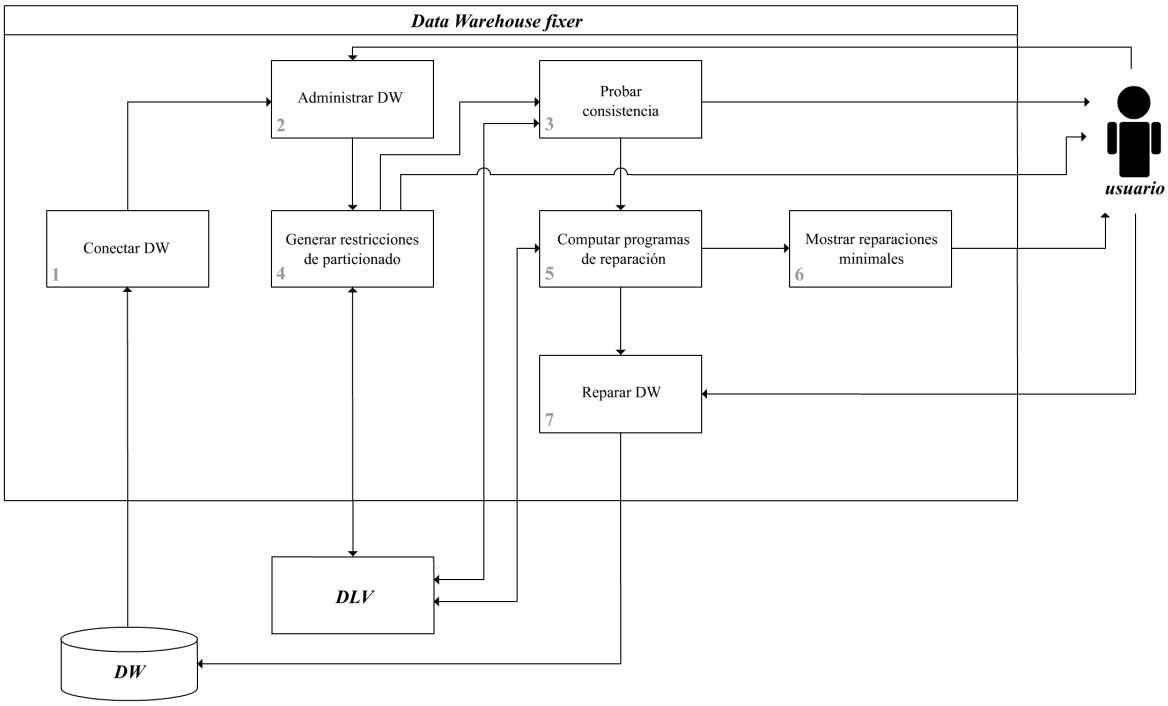


Figura 4.1 – Arquitectura de *Data Warehouse fixer*.

Módulos de la arquitectura

1. Conectar DW. Este módulo se encarga de realizar la conexión con un Data Warehouse y obtener las instancias de dimensión. La conexión puede establecerse sobre distintos DW, a su vez, este módulo administra los DW que han sido ingresados.
2. Administrar DW. Provee la funcionalidad necesaria para ingresar, modificar, eliminar y listar toda la información sobre cada instancia de dimensión del DW.

3. Generar restricciones de particionado. Se conecta al sistema DLV externo y mediante la ejecución de un programa genera un listado con las restricciones de particionado para una instancia dimensional específica.
4. Probar consistencia. Prueba la consistencia de una instancia de dimensión. Para su funcionamiento debe interactuar con el módulo 5 que obtiene las respuestas desde la computación de los programas de reparación en DLV.
5. Computar programas de reparación. Este módulo interactúa con DLV computando el programa de reparación analizado en el Capítulo 3, en este programa se cambia el operador ***choice*** por las reglas propias de Datalog con negación que funciona correctamente en DLV[4]. Una vez computado el programa devuelve los modelos estables que servirán para reparar en caso que la instancia analizada sea inconsistente con respecto a las restricciones de particionado.
6. Mostrar reparaciones minimales. Muestra al usuario un extracto de los modelos estables retornados por el módulo 5 indicando qué operaciones de reparación deben ejecutarse para devolver la consistencia; el usuario puede escoger uno de estos modelos y reparar la instancia.
7. Reparar DW. Con los modelos estables y la decisión del usuario sobre qué modelo usará en la reparación, el módulo reparar DW se conecta con la base de datos y realiza tareas de eliminación e inserción de arcos entre elementos.

4.2 Algoritmos

El sistema se compone de un conjunto de archivos PHP que interactúan entre si mediante HTML. Gráficamente éstos se comportan como archivos HTML pero incorporan código PHP para realizar internamente el manejo de cadenas de texto, arreglos de variables y consultas a la base de datos.

A continuación se explicará una serie de algoritmos que explican las funcionalidades más importantes de la aplicación. El desarrollo de cada uno de ellos

[4] Capítulo 3, Sección 3.3 – Programas de reparación

permite comprender cómo se realizan los procesos de administración, intercambio de información e interacción con el ejecutable externo DLV. Por administración se entiende el ingreso, modificación y eliminación de información.

Algoritmo 4.2.1 – Generar restricciones de particionado

Las restricciones de particionado se generan a partir de los arcos entre categorías sobre una dimensión determinada. Para generar estas restricciones es necesario interactuar con el sistema DLV mediante la ejecución de un programa que devuelve las restricciones que una dimensión consistente debe respetar.

Programa en DLV que genera restricciones de particionado

Reglas	***Explicación***
1. $pc(X,Z):-arco(X,Y), pc(Y,Z$	Genera todas las restricciones de particionado transitivamente.
2. $pc(X,Y):-arco(X,Y).$	Establece que cada arco entre categorías en la dimensión representa una restricción de particionado.
3. $arco(C_1,C_2).$	Representa un arco entre las categorías C_1 y C_2 .
4. $pc(X,Y)?$	Pregunta por todas las restricciones de particionado.

Ejemplo 4.1

El programa generador de restricciones de particionado para la dimensión Producto representada por la Figura 2.1 d)

$$pc(X,Z) :- arco(X,Y),\ pc(Y,Z).$$
$$pc(X,Y) :- arco(X,Y).$$
$$arco(Producto, Familia).$$
$$arco(Producto, Instrumento).$$
$$arco(Familia, Tipo).$$
$$arco(Instrumento, Tipo).$$
$$arco(Tipo, All).$$
$$pc(X,Y)?$$

Las respuestas que representan las restricciones obtenidas de ejecutar el programa anterior en DLV son las siguientes:

Producto, Familia
Producto, Instrumento
Producto, Tipo
Producto, All
Familia, Tipo
Familia, All
Instrumento, Tipo
Instrumento, All
Tipo, All

□

Para generar las restricciones de particionado en la aplicación primero ha de realizarse una consulta SQL que pregunte por todos arcos entre categorías de una dimensión determinada, luego mediante un arreglo y archivos externos “txt” es posible computar el programa y obtener resultados, a continuación se detallan los pasos de este algoritmo.

Pseudo código:

1. *Consultar a la base de datos todos los arcos entre categorías para una determinada dimensión.*

2. *Mientras (exista una arco)*
   {Extraer categoría hija y categoría padre, copiar ambas en una variable ***arcos*** *considerando regla* $arco(C_1, C_2)$ *del programa DLV}*
3. *Al no quedar más arcos por extraer copiar reglas* $pc(X,Z) :- arco(X,Y), pc(Y,Z)$ y $pc(X,Y) :- arco(X,Y)$ *en una variable* ***programa*** *adjuntando la variable* ***arcos*** *y la regla* $pc(X,Y)?$.
4. *Ejecutar variable* ***programa*** *en sistema DLV.*
5. *Leer datos devueltos por sistema DLV copiando resultados en una variable* ***respuesta****.*
6. *Extraer restricciones de particionado desde variable* ***respuesta*** *y mostrar en pantalla.*

Algoritmo 4.2.2 – Revisar consistencia en instancia dimensional

Revisar la consistencia para una instancia dimensional consiste en determinar si los arcos entre elementos de esta instancia respetan las restricciones de particionado establecidas sobre una dimensión específica. Por lo tanto, primero es necesario consultar por todos los arcos sobre la instancia dimensional y luego comprobar si estos vulneran alguna de las restricciones de particionado. Para llevar a cabo esta revisión es necesario ejecutar un programa en DLV que determine por cada arco si se cumplen las restricciones. De existir una violación se devuelve los elementos que generan la inconsistencia junto con la restricción de particionado vulnerada. De ser consistente la instancia dimensional, el programa devuelve una respuesta vacía, sin resultados.

Programa en DLV que revisa consistencia

Reglas	***Explicación***
1. $Inc(X,Y,Z,W) :- R2(X,Y,Z,W), R2(X,Y2,Z,W), Y != Y2.$	Esta regla comprueba si un elemento hijo de una categoría hija llega a dos

	elementos padres en la misma categoría padre. De existir tal elemento lo almacena como inconsistencia.
2. $R2(X,W,C1,C3):-R(X,Y,C1,C2),$ $R2(Y,W,C2,C3).$ $R2(E1,E2,C1,C2):-R(E1,E2,C1,C2).$	Genera todos los arcos R transitivos de la instancia dimensional.
3. $R(E1,E2,C1,C2).$	Representa un arco entre el elemento E1 de C1 y E2 de C2.
4. $Inc(X,Y,Z,W)?$	Pregunta por todas las inconsistencias.

Ejemplo 4.2

El programa DLV anterior para la instancia dimensional Producto inconsistente representada por la Figura 3.1 es el siguiente:

```
INC(X,Y,Z,W):-R2(X,Y,Z,W),R2(X,Y2,Z,W),Y!=Y2.
R2(X,W,C1,C3):-R(X,Y,C1,C2),R2(Y,W,C2,C3).
R2(E1,E2,C1,C2):-R(E1,E2,C1,C2).

R(Jazzbass,Cordofono,Producto,Familia).
R(Precisionbass,Cordofono,Producto,Familia).
R(Tb_a,Membranofono,Producto,Familia).
R(Jazzbass,Bajo,Producto,Instrumento).
R(Precisionbass,Bajo,Producto,Instrumento).
R(Cordofono,Cuerdas,Familia,Tipo).
R(Membranofono,Percusion,Familia,Tipo).
R(Bajo,Cuerdas,Instrumento,Tipo).
R(Tb_a,Bajo,Producto,Instrumento).

INC(X,Y,Z,W)?
```

Las respuestas producto de la ejecución del programa son las siguientes:

Tb_a,Cuerdas,Producto,Tipo

Tb_a,Percusion,Producto,Tipo

Esta respuesta informa que los elementos *Tb_a* y *Cuerdas* vulneran la restricción de particionado *Producto-Tipo*, esto debido a que el elemento *Tb_a* además de poseer una relación transitivamente al elemento *Cuerdas* en *Tipo* también posee una relación con *Percusion* en la misma categoría.

□

De forma similar al Algoritmo 4.2.1 para revisar la consistencia de una instancia dimensional la aplicación necesita primero ejecutar una consulta en SQL y a partir de los resultados obtenidos generar el programa DLV encargado de obtener las posibles inconsistencias.

Pseudo código:

1. *Consultar a la base de datos todos los arcos entre elementos de distintas categorías para una determinada dimensión.*
2. *Mientras (exista una arco)*
   {Extraer elemento hijo más su respectiva categoría y elemento padre más la categoría a la cual pertenece, luego copiar datos en una variable ***arcos*** *considerando regla* $R(E1, E2, C1, C2)$ *del programa DLV}*
3. *Luego de extraer todos los arcos, copiar reglas*
   $Inc(X, Y, Z, W) :- R2(X, Y, Z, W), R2(X, Y2, Z, W), Y != Y2.$ y
   $R2(X, W, C1, C3) :- R(X, Y, C1, C2), R2(Y, W, C2, C3).$ *junto con*
   $R2(E1, E2, C1, C2) :- R(E1, E2, C1, C2).$ *en una variable* ***programa*** *adjuntando la variable* ***arcos*** *y la regla* $Inc(X, Y, Z, W)?$.
4. *Ejecutar variable* ***programa*** *en sistema DLV.*
5. *Leer datos devueltos por sistema DLV copiando resultados en una variable* ***respuesta****.*
6. *Extraer las tuplas desde variable* ***respuesta*** *y mostrar en pantalla los datos. Si* ***respuesta*** *no contiene información es porque la instancia dimensional es*

consistente y se muestra por pantalla la información respecto a ello. De haber inconsistencias se extrae las tuplas desde la variable ***respuesta*** *y se muestra la información ordenando primero los elementos y luego las restricciones de particionado que han sido vulneradas.*

Algoritmo 4.2.3 – Reparar instancia dimensional

De existir inconsistencias producto de la ejecución del programa en DLV descrito en el Algoritmo 4.2.2 la aplicación permite el ingreso al módulo de reparación. Este módulo funciona ejecutando el programa de reparación en DLV visto en el Capítulo 3. A continuación se detalla los pasos para obtener los modelos estables reducidos usados para generar una reparación a una instancia dimensional inconsistente, se asume la existencia de un DW con dimensiones, categorías, elementos, arcos entre categorías y arcos entre elementos.

Pseudo código:

1. *Consultar a la base de datos todos los arcos entre elementos de distintas categorías para una determinada dimensión.*
2. *Mientras (exista una arco)*
   {Extraer elemento hijo más su respectiva categoría y elemento padre más la categoría a la cual pertenece, luego copiar datos en una variable ***arcos*** *considerando regla* $R(a,b,C(a),C(b))$ *del programa DLV}*
3. *Una vez terminado el proceso de extraer los arcos se adjuntan todas las reglas del programa de reparación a una variable* ***programa*** *incluyendo la variable* ***arcos****.*
4. *Ejecutar variable* ***programa*** *en sistema DLV.*
5. *Leer datos devueltos por sistema DLV copiando resultados en una variable* ***respuesta****.*
6. *De existir modelos estables en la variable* ***respuesta*** *mostrar cada uno de ellos por separado adjuntando el botón que permite reparar finalmente la instancia.*

7. *Cada uno de los modelos existentes en la variable **respuesta** retornan la consistencia a la instancia dimensional, por lo tanto, seleccionado uno de los ellos y pulsando el **botón reparar** se realizan las tareas de inserción y eliminación indicadas por cada modelo.*
8. *Una vez completadas las tareas de reparación se muestra el listado completo de arcos en la instancia dimensional reflejando los cambios realizados.*

4.2.4. Algoritmos de administración.

a) Ingreso de información

Esta serie de pasos conduce al ingreso de información desde la aplicación a la base de datos y permite su uso dentro de la misma. El siguiente ejemplo muestra los pasos necesarios para ingresar un arco entre elementos de distintas categorías de una dimensión específica en un Data Warehouse.

Pseudo código:

Ingresar arco entre elementos
{
Conectar a Data Warehouse ( )
{
SI(EXISTE DATA WAREHOUSE)
{Seleccionar desde la lista desplegable un Data Warehouse;}
DE LO CONTRARIO
{Ingresar nuevo Data Warehouse y volver a conectar;}
}

Desplegar pantalla principal de la aplicación con menús de acceso;

Mientras(No se haya terminado de ingresar elementos)
{
Ingresar Elemento ( )
{
SI(EXISTE DIMENSION)
{Abrir Ingresar Elemento desde el menú principal;
Seleccionar dimensión; }
DE LO CONTRARIO
{ Abrir Ingresar Dimensión desde el menú principal;
Ingresar nombre para dimensión;
//Luego de ingresar una dimensión automáticamente se ingresa la
//categoría 'All' (superior) y el elemento único 'all' de esta categoría
Regresar a Ingresar Elemento ( ); }

SI(EXISTE CATEGORIA)
{Seleccionar categoría;
Ingresar nombre de elemento para la categoría seleccionada;
//Luego de ingresar el elemento, si este pertenece a una
//categoría relacionada con la categoría superior 'All'
//automáticamente se ingresa un arco entre este elemento
//y el elemento 'all' de la categoría superior}
DE LO CONTRARIO
{ Abrir Ingresar Categoría desde el menú principal;
Seleccionar dimensión;
Ingresar nombre de categoría para la dimensión seleccionada;
Regresar a Ingresar Elemento ( );}
}// Fin ingresar elemento
}//Fin mientras

```
Ingresar arco-elementos ( )
{
        Seleccionar dimensión;
        Seleccionar categoría origen; // Categoría hija

        SI(EXISTE RELACION ENTRE CATEGORIAS)
        {
        Seleccionar categoría destino; // Categoría padre
        Mientras(no se haya seleccionado elemento hijo y elemento padre)
        {
                Seleccionar elemento( )
                {
                        SI(EXISTE ELEMENTO EN CATEGORIA)
                        {Seleccionar elemento; // hijo o padre}
                        DE LO CONTRARIO
                        {
                                //Aplicación informa que una de las categorías
                                //no posee elementos
                                Regresar a Ingresar Elemento ( );
                        }
                }

        }// Fin mientras
        Ingresar arco entre elementos;
        }
        DE LO CONTRARIO
        {
        Abrir Ingresar relación desde el menú principal;
        Seleccionar dimensión;
        Seleccionar categoría hija;
```

Seleccionar categoría padre;
Ingresar arco entre categorías;
//Luego de ingresar un arco entre categorías, si la categoría padre
//es la categoría superior 'All' automáticamente se ingresa un arco
//por cada uno de los elementos de la categoría hija (si es que estos existen) y el
//elemento 'all' de la categoría superior.
Regresar a Ingresar arco-elementos ( );
}
}// Fin ingresar arco-elementos
}// Fin Ingresar arco entre elementos

b) Modificación de información

La aplicación considera la modificación de los nombres para cualquiera de los siguientes: Data Warehouse, Dimensión, Categoría y Elemento. La modificación de arcos, tanto entre categorías como entre elementos se realiza primero eliminando un arco existente y luego insertando uno nuevo; ambas operaciones son de forma manual, esto con el propósito de que no exista información con referencias perdidas dentro de la base de datos, por ejemplo: modificar un arco entre categorías, si no se considera la eliminación como primer paso es posible encontrarse con arcos entre elementos que no dependen de un arco entre categorías. Modificar información como se dijo anteriormente solo está habilitada para *nombres*, cada DW, dimensión, categoría y elemento además de su respectivo nombre poseen un identificador numeral que se auto incrementa luego de cada ingreso, este valor no puede modificarse debido a ser la clave primaria con la cual es posible establecer las relaciones entre DW, Dimensiones, Categorías, Elementos, arcos entre categorías y arcos entre elementos. El siguiente pseudo código consiste en la modificación de una categoría; se asume para ello la existencia de un DW y de una dimensión.

Pseudo código:

Modificar categoría

{

Conectar a Data Warehouse ( )

{

Seleccionar Data Warehouse desde la lista desplegable;

}

Desplegar pantalla principal de la aplicación con menús de acceso;

Abrir Listar categorías;

Seleccionar dimensión;

Seleccionar categoría;

SI(PROCEDER MODIFICACIÓN)

{

Sobrescribir nombre de categoría actual;

Modificar categoría;

//Los arcos en que participa esta categoría y arcos entre elementos de esta

//categoría permanecen intactos.

}

DE LO CONTRARIO

{

Cancelar modificación dejando el mismo nombre;

Regresar a Listar categorías;

}

}// Fin modificar categoría

c) Eliminación de información

Eliminar información desde la aplicación es un procedimiento sumamente importante que considera eliminaciones simples y anidadas; lo último permite que no exista por ejemplo, relaciones entre elementos de categorías que han sido eliminadas. Antes de realizar cualquier eliminación que implique eliminar otros datos, la aplicación informa al usuario advirtiendo de que lo que está pronto a ocurrir si decide continuar. El siguiente pseudo código muestra los pasos necesarios para eliminar una dimensión; se asume la existencia de un Data Warehouse del cual se eliminará ésta.

Pseudo código:

Eliminación dimensional
{
Conectar a Data Warehouse ( )
{
Seleccionar Data Warehouse desde la lista desplegable;
}

Desplegar pantalla principal de la aplicación con menús de acceso;

Eliminar dimensión ( )
{
Abrir Listar dimensiones desde el menú principal;
Seleccionar dimensión a eliminar;
La aplicación despliega advertencia: "Eliminar una dimensión implica eliminar la dimensión y todas las categorías, elementos y relaciones asociadas a ella";

SI(PROCEDER ELIMINACIÓN)
{
Consultar categorías asociadas a la dimensión;

Mientras(existan categorías asociadas a la dimensión)
{
Consultar elementos asociados a la categoría;
Mientras(existan elementos asociados a la categoría)
{
Eliminar elemento ( );
}
Eliminar categoría ( );
}
Consultar arcos entre categorías asociadas a la dimensión;
Mientras(existan arcos entre categorías asociadas a la dimensión)
{
Consultar arcos entre elementos de categorías asociadas a la dimensión;
Mientras(existan arcos entre elementos de categorías asociadas a la dimensión)
{
Eliminar arco entre elementos ( );
}
Eliminar arco entre categorías ( );
}
Eliminar dimensión ( );
}// Fin SI
DE LO CONTRARIO
{ Regresar y abrir Listar dimensiones; }
}// Fin Eliminar dimensión
}//Fin Eliminación dimensional

4.3 Base de datos

El modelo de datos multidimensional puede almacenarse usando un conjunto de relaciones.

La base de datos del sistema WEB *Data Warehouse fixer* en su modelo físico se representa por la Figura 4.3

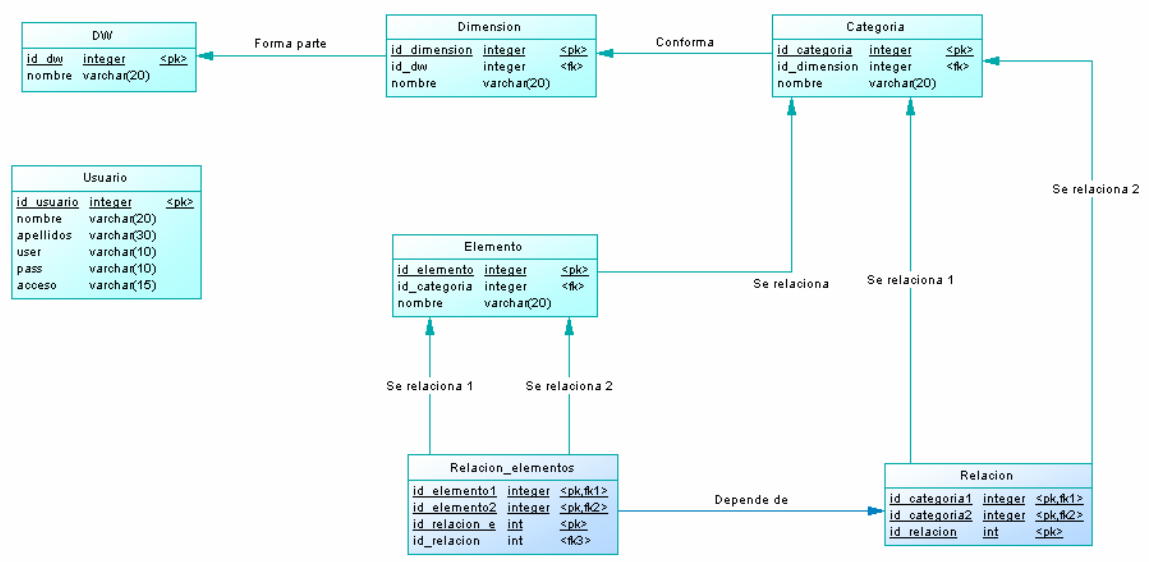


Figura 4.3 – Base de datos sistema *Data Warehouse fixer*

1. Tabla **DW**. Almacena el identificador (***id_dw***) del DW asignado automáticamente por el sistema gestor de base de datos que además es su clave primaria y el atributo ***nombre*** definido por el usuario.

2. Tabla **Dimensión**. Sus atributos son ***id_dimension*** (identificador/clave primaria) y ***nombre*** de la dimensión. Junto con sus atributos básicos del modelo conceptual se incorpora el atributo ***id_dw*** como clave foránea proveniente de la tabla DW.

3. Tabla **Categoría**: Dispone de ***id_categoría*** (identificador/clave primaria), ***id_dimension*** (clave foránea proveniente de la tabla Dimensión) y ***nombre*** definido por el usuario.

4. Tabla **Elemento**: Almacena ***id_elemento*** (identificador/clave primaria), como clave foránea incorpora el atributo ***id_categoría*** que establece su dependencia con una y solo una categoría y ***nombre*** ingresado por el usuario.

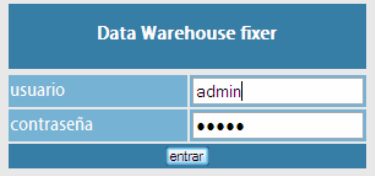
5. Tabla **Relación**: Esta tabla establece la relación entre categorías de una misma dimensión. Su clave primaria se compone de tres atributos, el primero de ellos es el identificador de la relación (***id_relacion***), los dos siguientes representan a los identificadores de ambas categorías relacionadas (***id_categoria1***,***id_categoria2***).

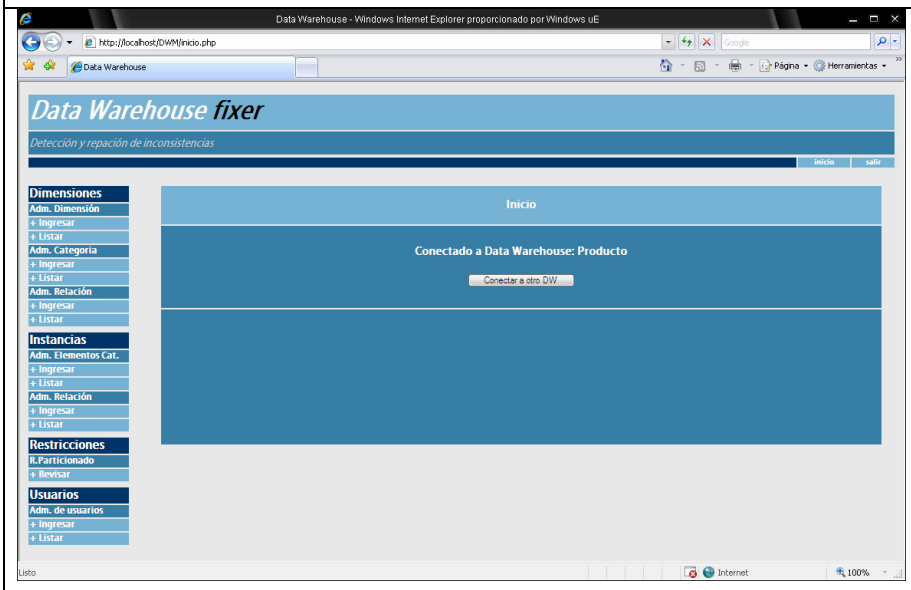
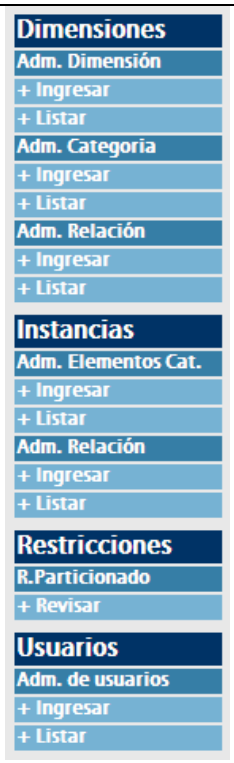
6. Tabla **Relación_elementos**: Establece la relación entre elementos de distintas categorías en una misma dimensión. La clave primaria se compone de tres atributos, el primero de ellos es el identificador de relación (***id_relacion_e***) y luego siguen los identificadores primarios de cada uno de los elementos relacionados (***id_elemento1***, ***id_elemento2***). Por último se incorpora un atributo foráneo de la tabla Relación (***id_relacion***) que sirve para mantener la dependencia de existencia, esto quiere decir que si no existe la relación entre categorías no debería existir relación entre elementos de estas categorías.

7. Tabla **Usuarios**: Esta tabla no mantiene dependencia con ninguna otra tabla en el sistema; su función es la de almacenar los usuarios que administran la aplicación, su información para ingresar en el sistema y su nivel de importancia denotado por el acceso. Su clave primaria es ***rut***, los demás atributos son: ***nombre***, ***apellidos***, ***user***, ***pass*** y ***acceso***.

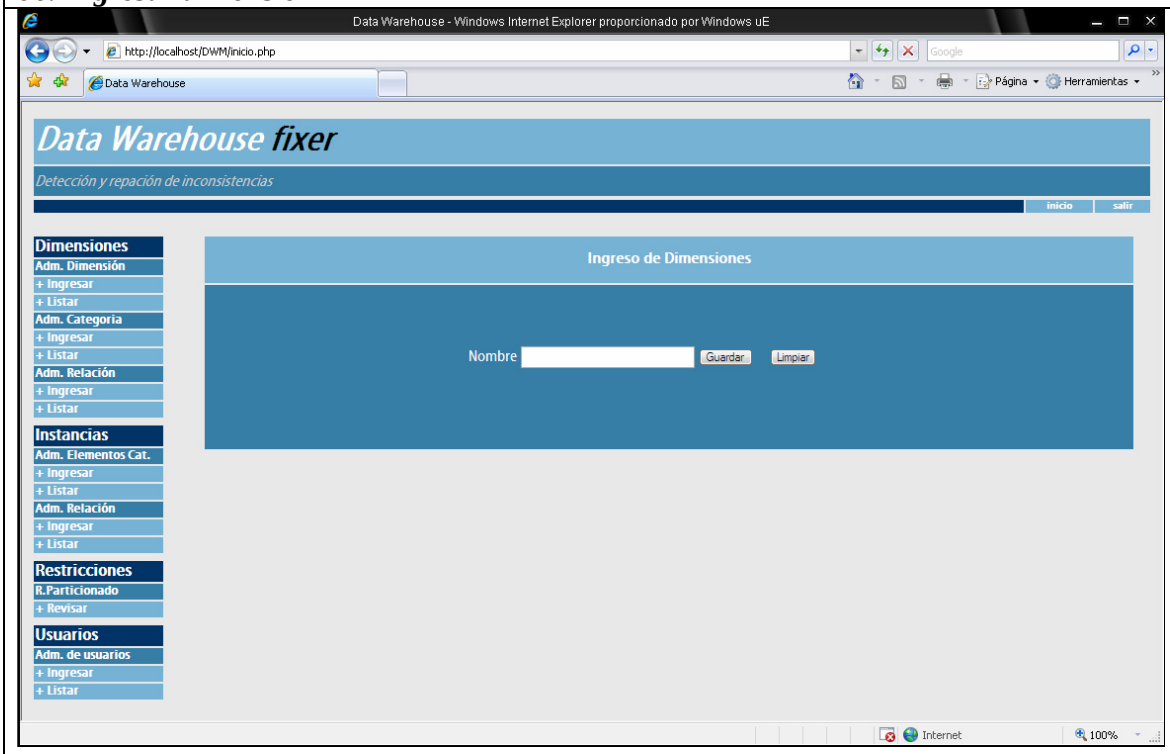
Capítulo 5 - Interfaz

La interfaz de la aplicación Data Warehouse fixer se compone de una serie de menús que administran los datos de forma clara y estructurada. Este capítulo mostrará acompañado de un ejemplo las pantallas y a su vez el manual de usuario de la aplicación, finalizando con los pasos necesarios para llevar a cabo la instalación de ésta en los sistemas Microsoft Windows Xp y Linux Ubuntu 8.04.

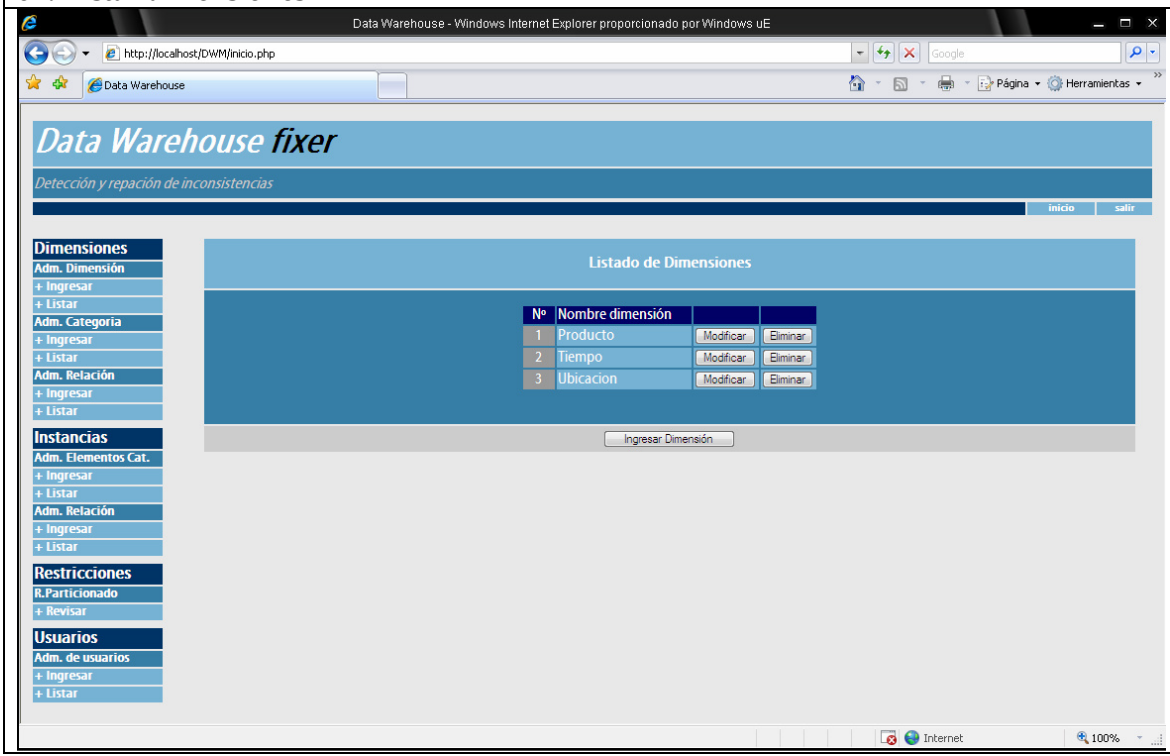
01. Ingresar al sistema	***02. Conectar con DW***	***03. Ingresar DW***
		
Primera pantalla del sistema que muestra el formulario de ingreso. Para ingresar por primera vez el usuario y contraseña son ***admin***.	Luego de validar los datos del usuario es necesario conectarse a un DW desde la lista desplegable, si es la primera vez que se llega a esta pantalla habrá que pulsar el botón ***Nuevo / Administrar DWs***. De existir DWs en el sistema solo es necesario seleccionar uno de ellos desde la lista para acceder a la pantalla principal.	Si se ha pulsado el botón Nuevo / Administrar DWs se despliega el formulario de ingreso seguido del listado de DWs ingresados con anterioridad. Desde aquí también es posible Modificar sus nombres o eliminarlos.

04. Pantalla Principal	***05. Menú***
	
La pantalla principal se compone de tres partes: la primera es el bloque superior que muestra el nombre de la aplicación y bajo este al costado derecho los botones de inicio y salir. El botón inicio así como también el nombre de la aplicación conducen a la pantalla principal (home). El botón salir cierra la sesión actual y retorna a la pantalla Nº 1 *Ingresar al sistema*. El segunda parte es el bloque menú ubicado al costado izquierdo de la aplicación y mostrado en detalle en la pantalla Nº 5 Menú. Por último está el bloque central donde se despliegan las pantallas seleccionadas desde el menú. Cuando se está en el home este bloque informa el nombre del DW al cual se ha conectado, también ofrece la posibilidad de conectar a otro mediante el botón bajo el nombre. Tras pulsar este botón se regresa a la pantalla Nº 2 *Conectar con DW*.	

06. Ingresar dimensión

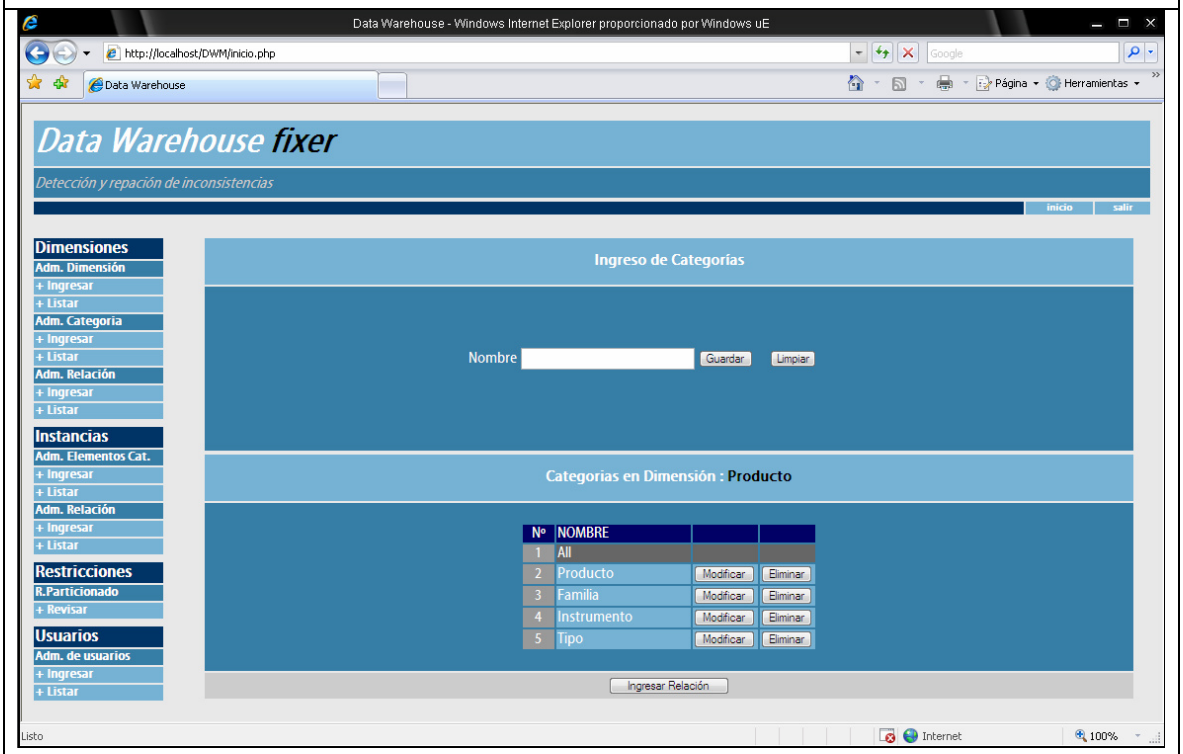


07. Listar dimensiones



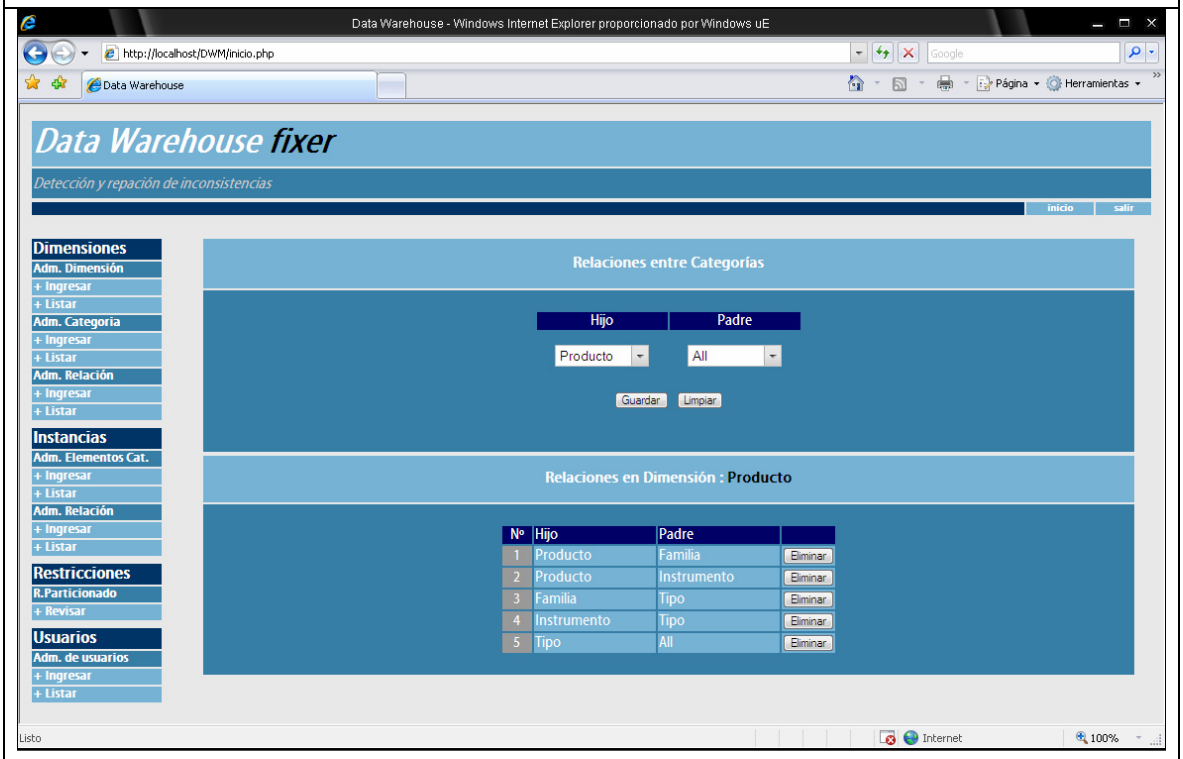
Seleccionando Ingresar desde el menú en el apartado Dimensiones – Adm. Dimensión se entra a la pantalla Nº 6 Ingresar dimensión. El sistema validará que el nombre ingresado no exista previamente en el DW seleccionado desde la pantalla Nº 2 Conectar con DW. El botón ***guardar*** ingresa el nombre a la base de datos y a continuación despliega la pantalla Nº 7 Listar dimensiones. El botón ***limpiar*** resetea la campo de texto dejándolo vacío.

08. Ingresar categoría



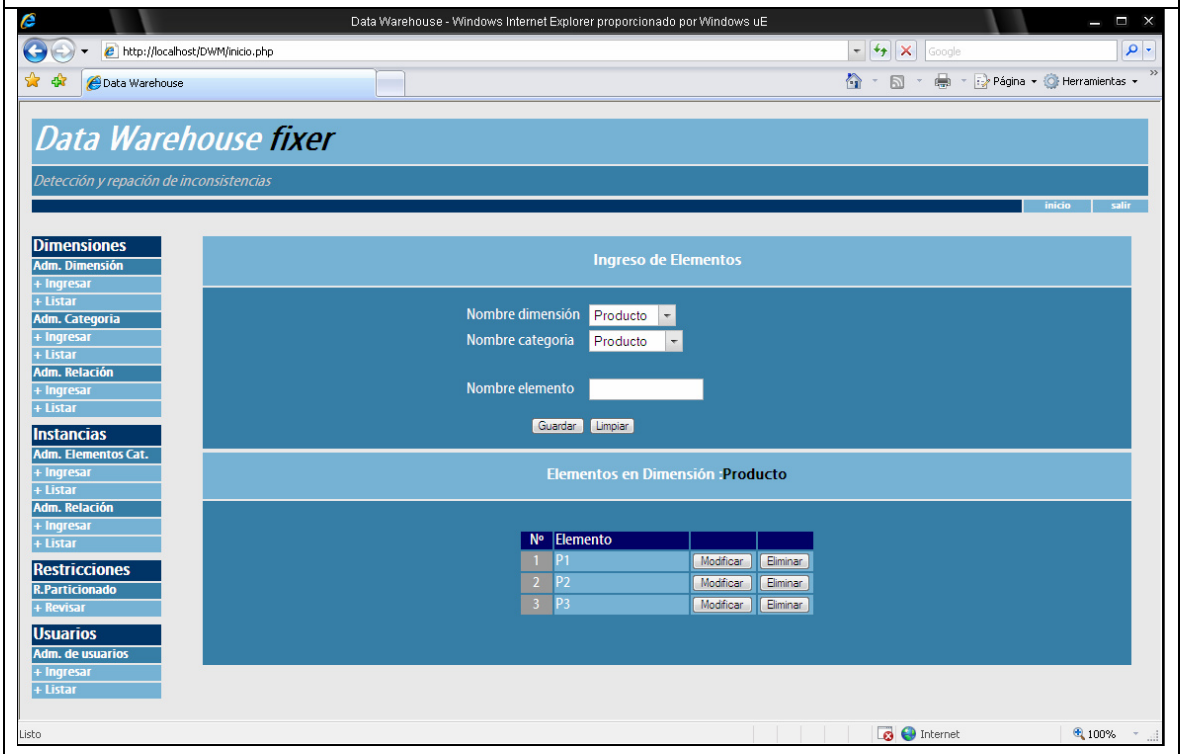
Seleccionando Ingresar desde el menú en el apartado Dimensiones – Adm. Categoría se muestra una pantalla con una lista desplegable que contiene las dimensiones, seleccionado una de ellas se entra a la pantalla Nº 8 *Ingresar categoría*. En esta pantalla se dispone de un campo de texto para ingresar el nombre de una categoría, luego de presionar el botón ***guardar*** se ingresa el nombre en la base de datos y se actualiza la pantalla mostrando el listado de categorías ingresadas. En el menú, la opción Listar en el apartado Dimensiones – Adm. Categoría, luego de escoger una dimensión despliega el mismo listado de la pantalla Nº 8 pero sin incluir el formulario de ingreso.

09. Ingresar relaciones entre categorías



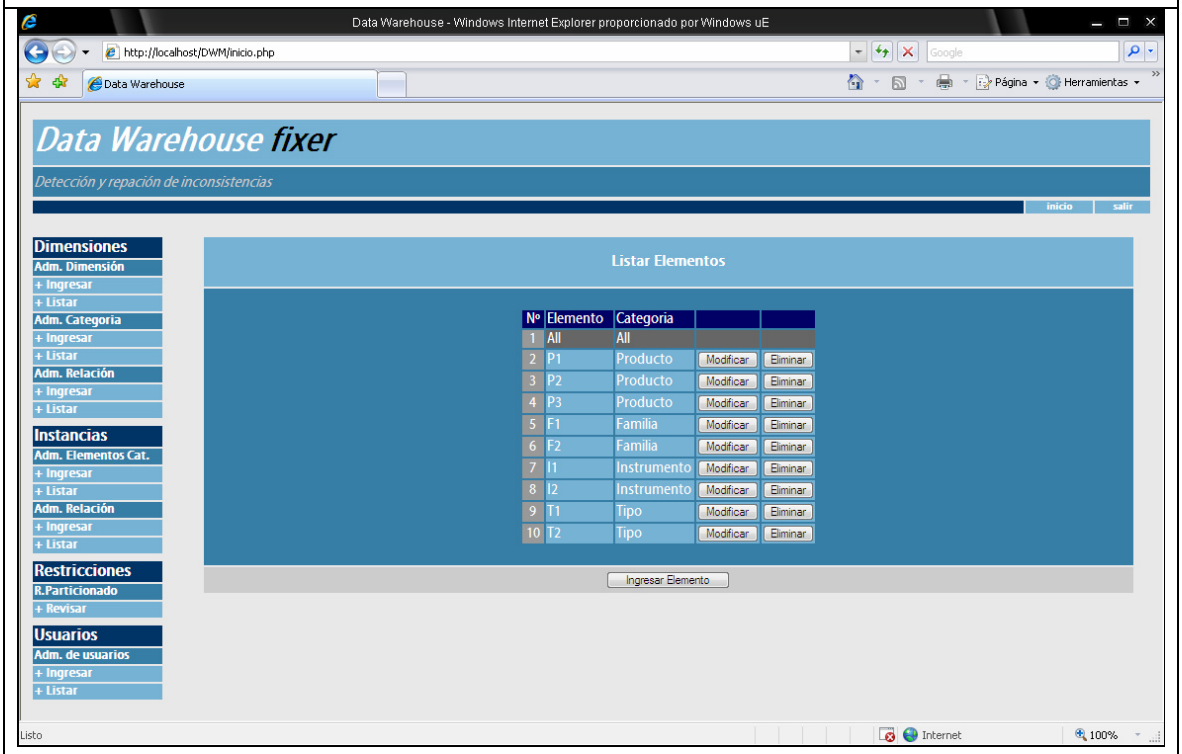
Seleccionando Ingresar desde el menú en el apartado Dimensiones – Adm. Relación se muestra una pantalla con una lista desplegable que contiene las dimensiones, seleccionado una de ellas se entra a la pantalla Nº 9 *Ingresar relaciones entre categorías*. En esta pantalla se dispone de dos listas desplegables con las categorías de la dimensión seleccionada. Escogiendo una categoría como hija y otra como padre luego de presionar el botón guardar se ingresa este arco en la base de datos que permitirá más adelante ingresar arcos entre los elementos de estas categorías. Bajo el formulario de ingreso para las relaciones se muestra el listado con todas las relaciones (arcos en el esquema dimensional) entre categorías y seguido de cada una la opción de eliminación. En el menú, la opción Listar en el apartado Dimensiones – Adm. Relación, luego de escoger una dimensión despliega el mismo listado de la pantalla Nº 9 pero sin incluir el formulario de ingreso.

10. Ingresar elementos



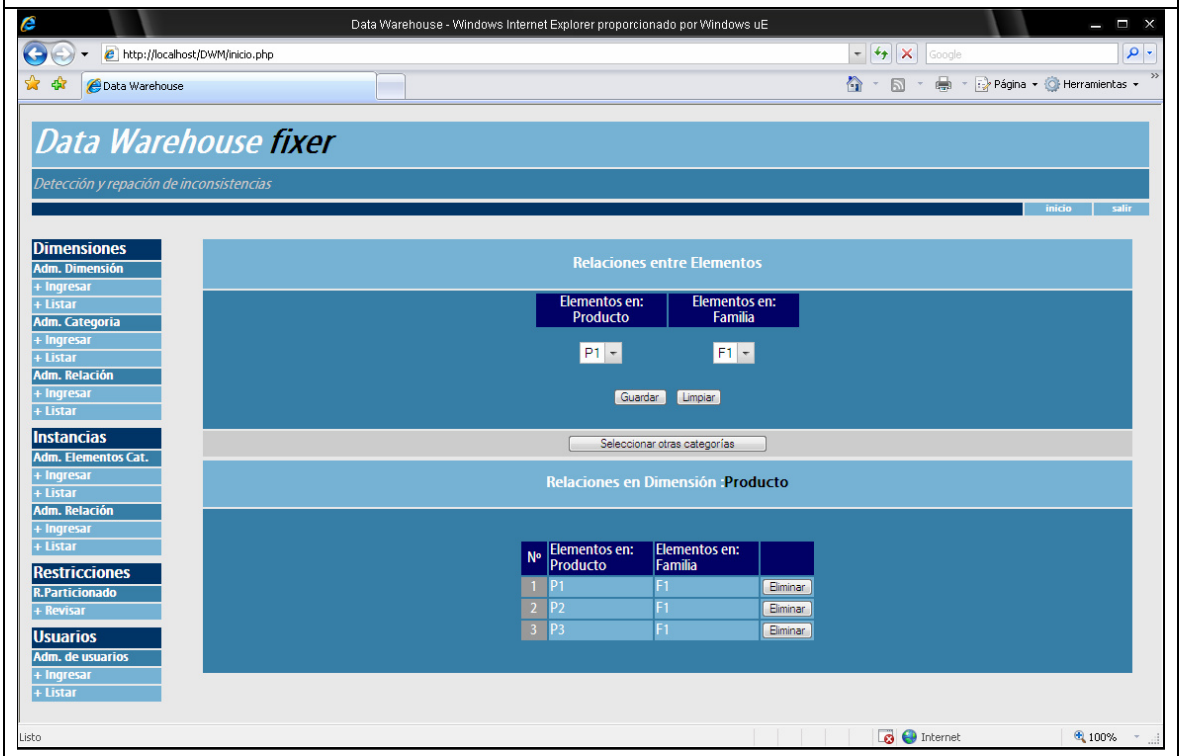
Seleccionando Ingresar desde el menú en el apartado Instancias – Adm. Elementos Cat. se muestra una pantalla con una lista desplegable que contiene las dimensiones, seleccionado una de ellas se despliega otra pantalla donde ha de seleccionarse la categoría en la cual se desea ingresar un elemento. Una vez escogida la dimensión y la categoría se despliega la pantalla Nº 10 *Ingresar elementos*, aquí es posible ingresar un elemento desde un formulario; bajo éste se muestra un listado con los elementos ingresados en la categoría seleccionada adjuntando las opciones de modificar y eliminar. Este listado también es accesible desde Listar en el apartado de Adm. Elementos Cat. seleccionando *Listar por categoría*. Es importante señalar que para ingresar un elemento en una categoría distinta a la seleccionada solo debe escogerse una diferente desde la lista desplegable en el formulario, del mismo modo es posible cambiar la dimensión en el formulario.

11. Listado completo de elementos



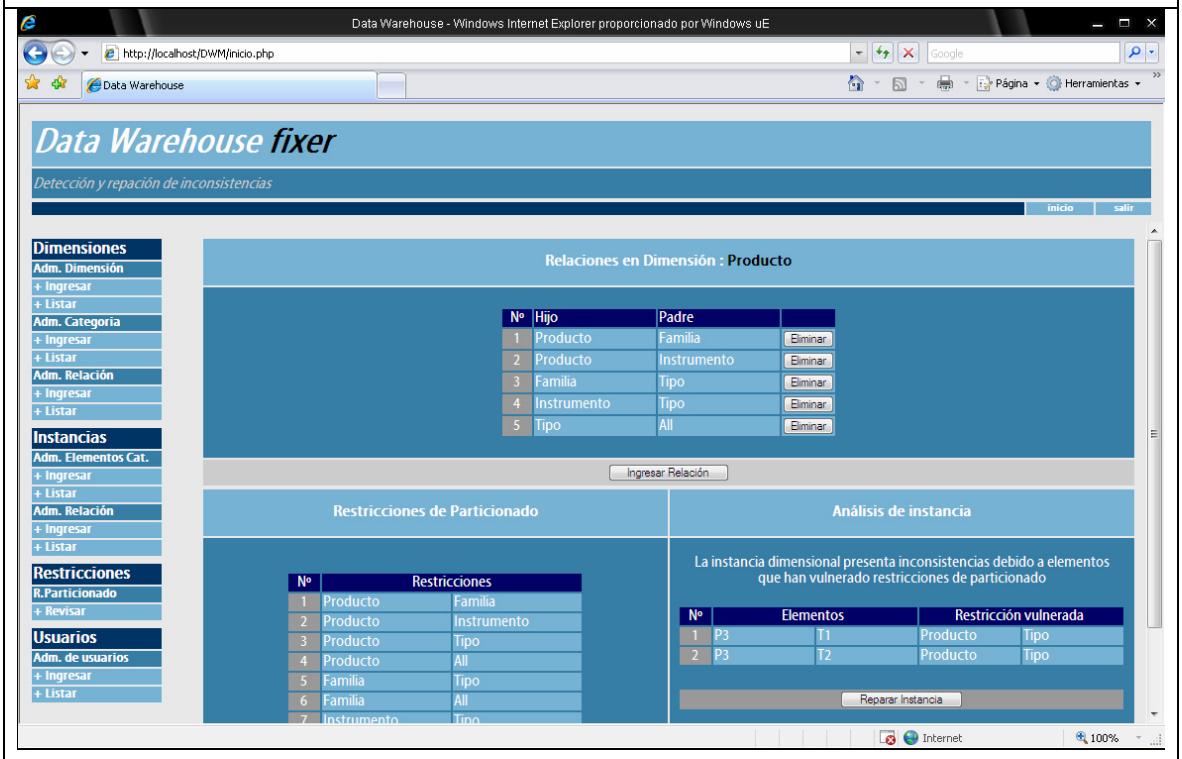
Seleccionando Listar desde el menú en el apartado de Instancias – Adm. Elementos Cat. y a continuación escogiendo *Listado completo* se muestra la pantalla Nº 11 *Listado completo de elementos*, este incluye todos los elementos que se han ingresado en cada una de las categorías de la dimensión seleccionada.

12. Ingresar relaciones entre elementos



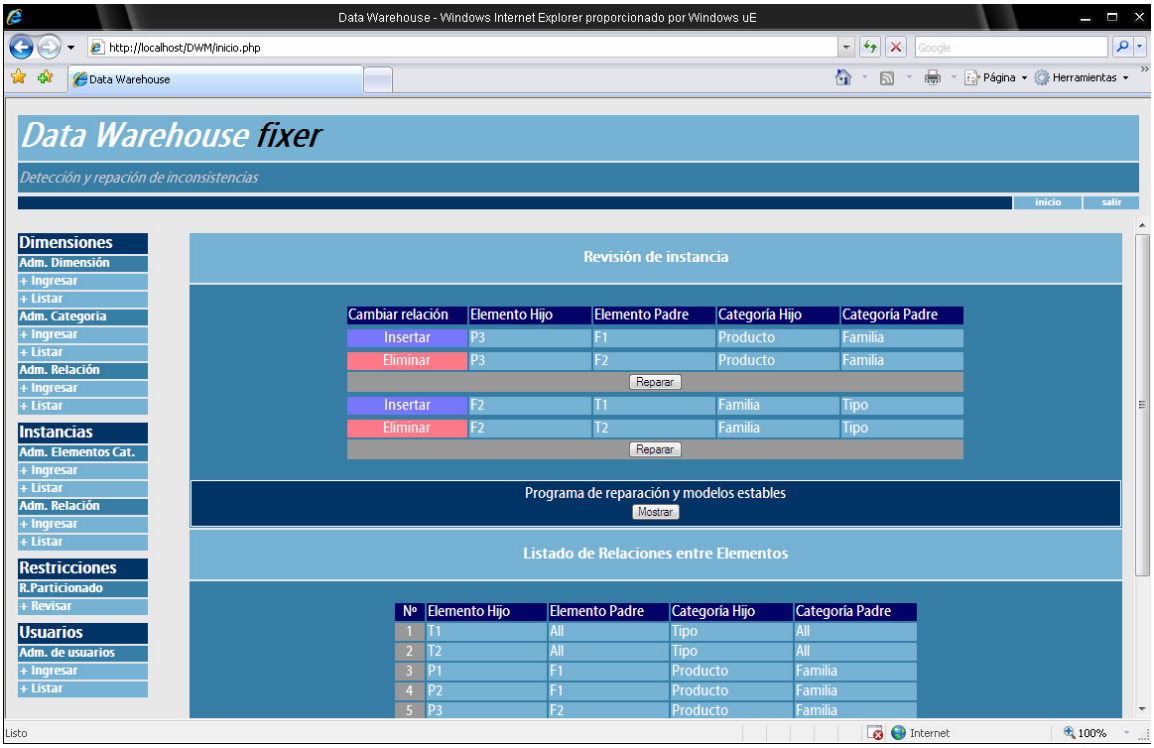
Seleccionando Ingresar desde el menú en el apartado Instancias – Adm. Relación se muestra una pantalla con una lista desplegable que contiene las dimensiones, seleccionado una de ellas se despliega una segunda pantalla que incluye un listado con las categorías pertenecientes a la dimensión, se indica que seleccionando una de ellas se mostrará a continuación una nueva pantalla listado las categorías padres de la primera. Una vez escogidas dimensión, categoría hija y categoría padre se muestra la pantalla Nº 12 *Ingresar relaciones entre elementos*. Aquí se ven dos listas desplegables en un formulario, la primera tiene los elementos de la categoría hija y la segunda los elementos de la categoría padre. En la parte inferior se despliega el listado de los arcos entre elementos ingresados para las categorías seleccionadas. También es posible acceder a este listado desde Listar en el menú ubicado el apartado de Instancias – Adm. Relación seleccionando *Listar por categoría-categoría*. Si se desea un listado completo de todos los arcos entre elementos de categorías para una dimensión específica ha de seleccionarse *Listado completo*.

13. Restricciones de particionado



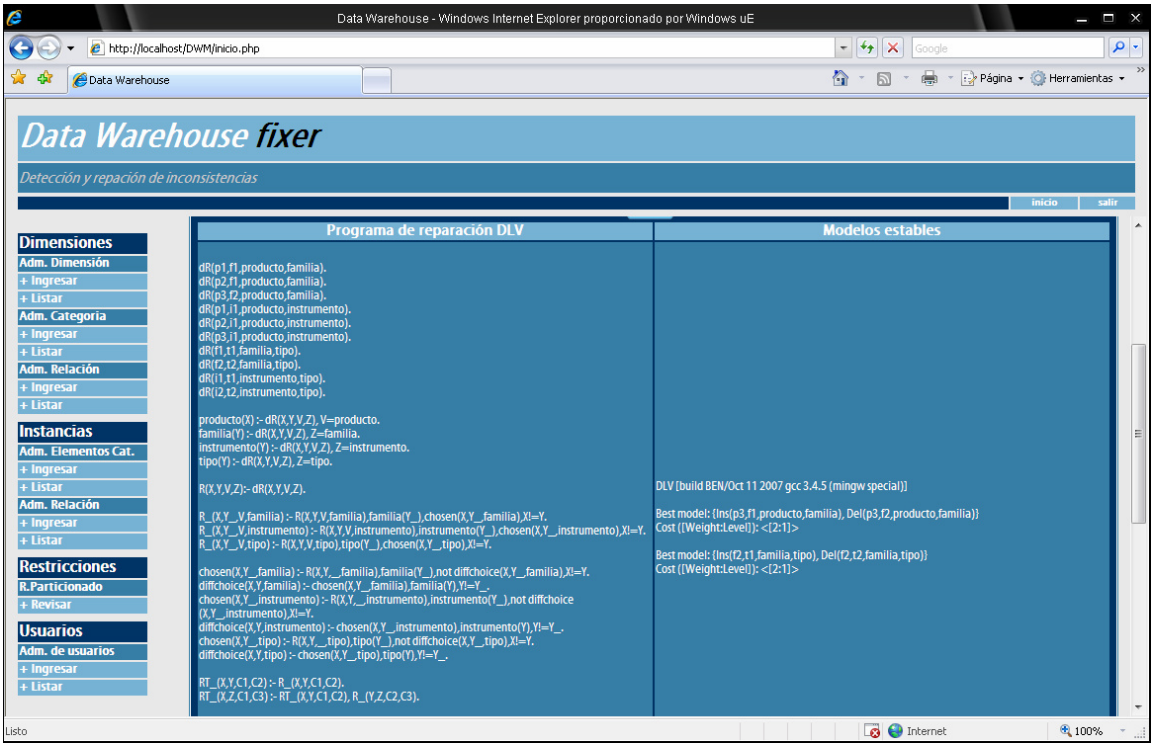
Seleccionando Revisar desde el menú en el apartado Restricciones – R.Particionado se muestra una pantalla con una lista desplegable que contiene las dimensiones, seleccionado una de ellas se entra a la pantalla Nº 13 Restricciones de particionado. Esta pantalla muestra las restricciones de particionado calculadas sobre el listado de arcos en el esquema para la dimensión seleccionada. En la parte superior de esta pantalla se despliega el mismo listado de relaciones de la pantalla Nº 9 adjuntando la opción de ingresar una nueva relación lo que actualizaría las restricciones de particionado. Al costado inferior derecho se muestra si la instancia dimensional es consistente con respecto de las restricciones de particionado. De vulnerarse alguna de las restricciones se muestra los elementos que participan en esta violación y las restricciones de particionado vulneradas; a continuación se muestra el botón que permite la reparación de la instancia. Más abajo es posible revisar el programa en DLV que revisa la consistencia de la instancia.

14. Reparar Instancia



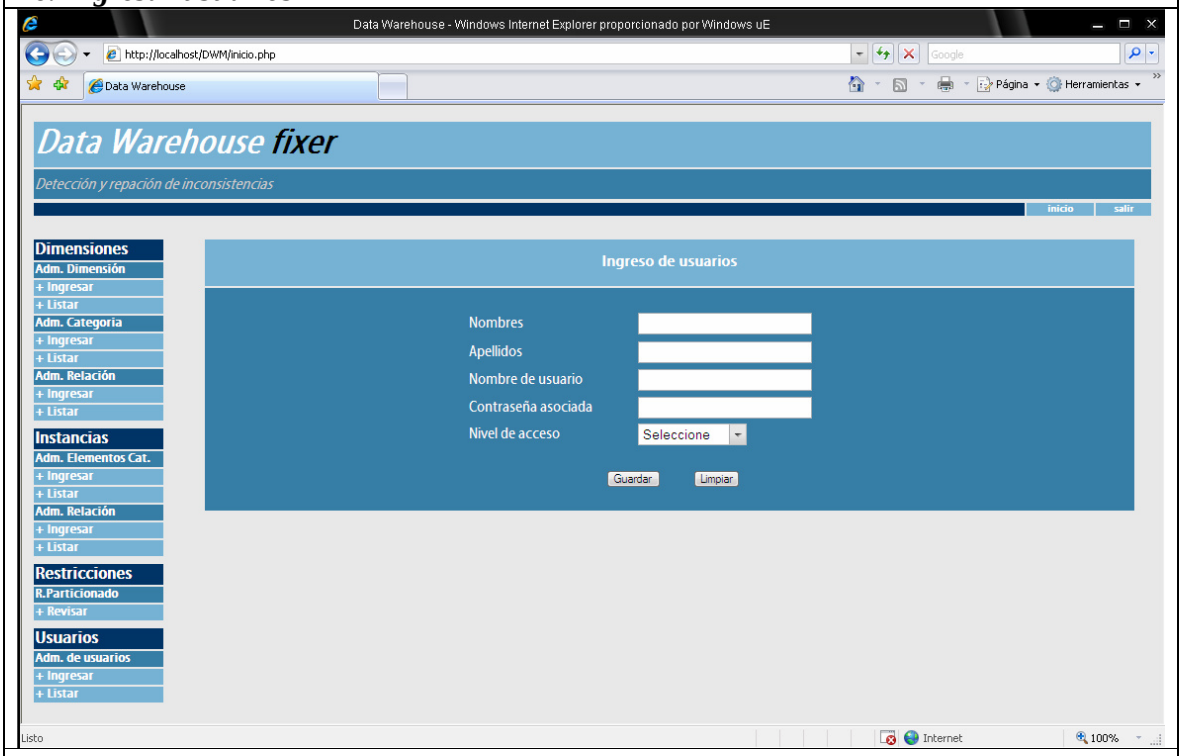
Si se presiona el botón reparar de la pantalla Nº13 – Restricciones de particionado se ingresa a esta pantalla. En la parte superior se muestran las reparaciones posibles de realizarse para reestablecer la consistencia en la instancia. Para esto se indica qué arco debe eliminarse y cuál insertarse en cada una de las reparaciones. En la sección intermedia es posible abrir una pantalla extra que muestra el programa de reparación ejecutado en DLV y la porción de los modelos estables reducida a inserciones y eliminaciones de arcos. Esta información es posible de apreciar en la pantalla Nº 15 *Programas de reparación y modelos estables*.

15. Programas de reparación y modelos estables

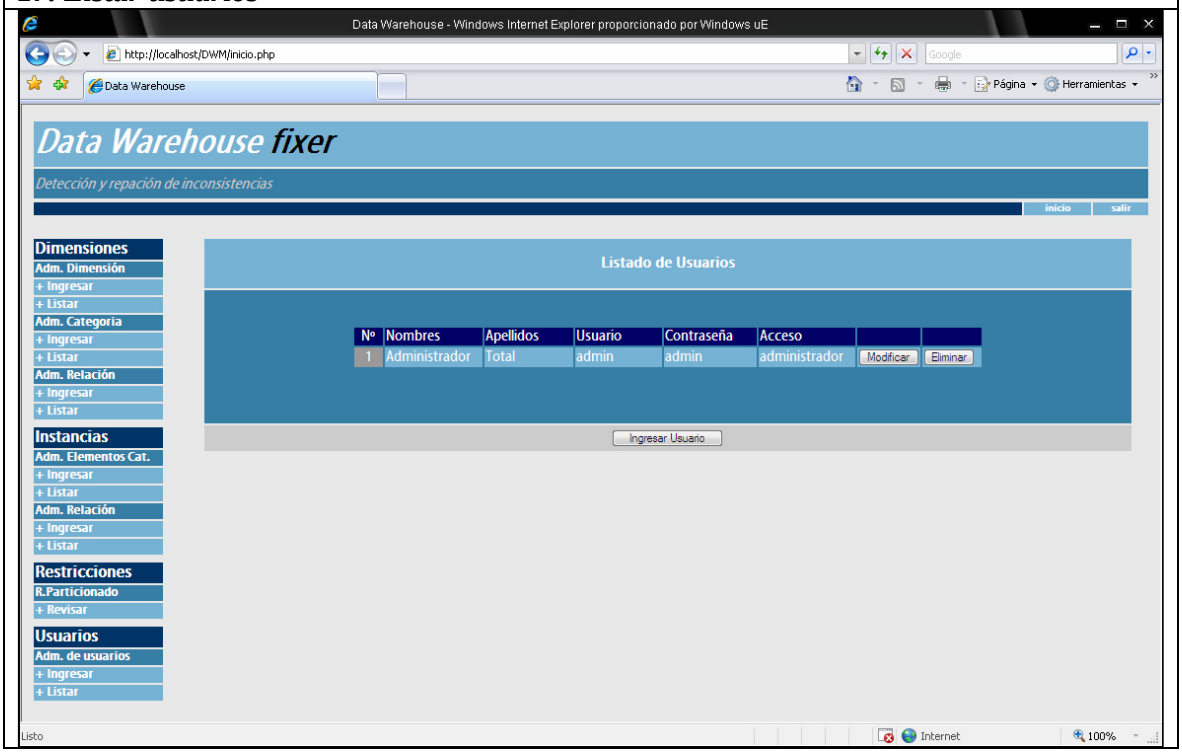


Las siguientes pantallas, Nº 16 y 17 permiten el ingreso y la actualización de usuarios respectivamente, para acceder a ellas se ha de ingresar en el menú de Usuarios.

16. Ingresar usuarios



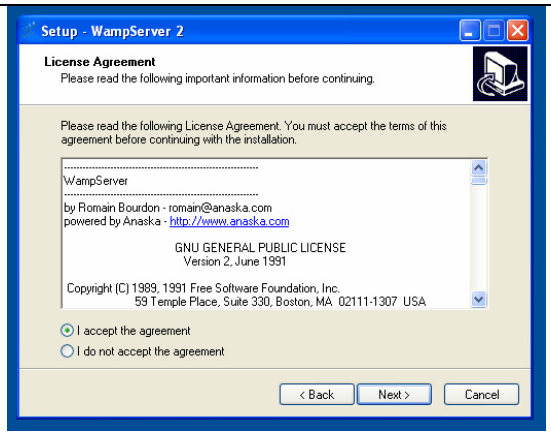
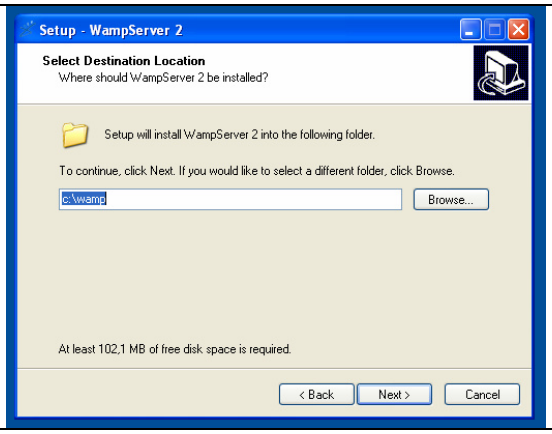
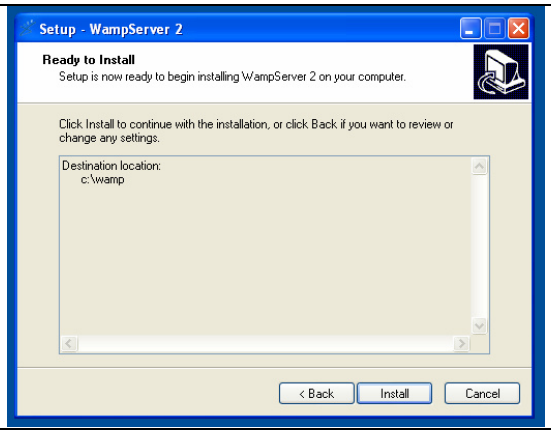
17. Listar usuarios

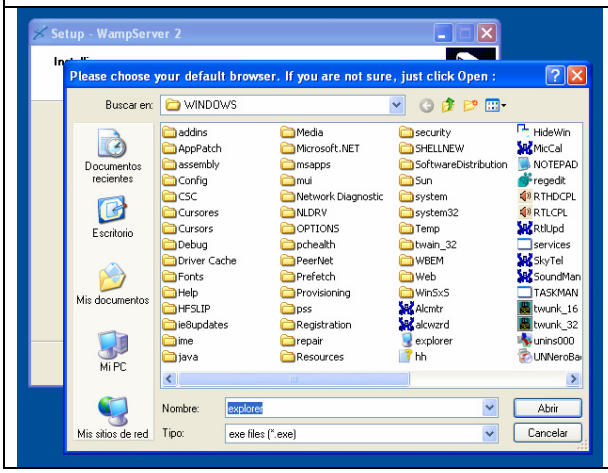
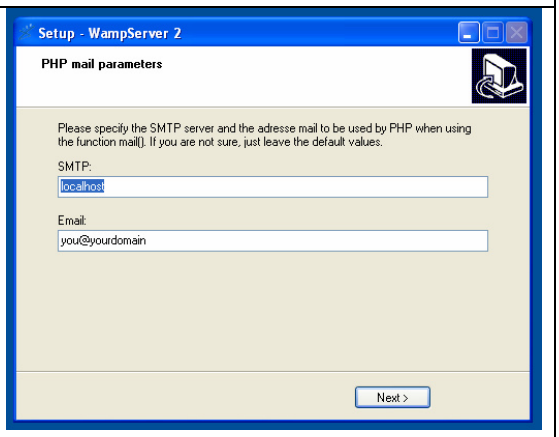


5.1 Instalación en Microsoft Windows Xp

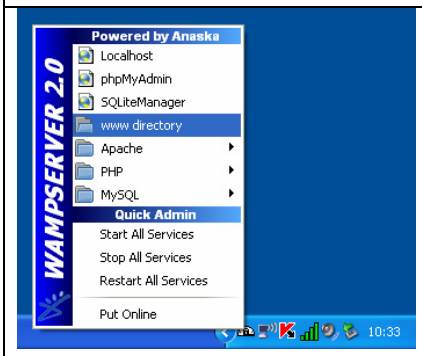
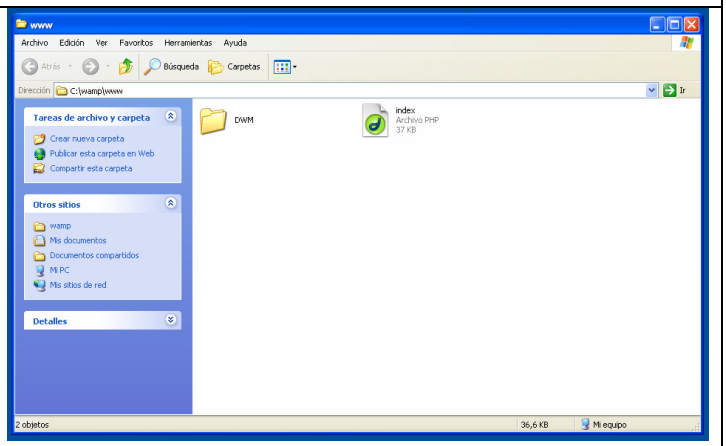
Los pasos para realizar la instalación de *Data Warehouse fixer* son los siguientes:

1. Instalar WAMPSERVER 2.0

1.1 Doble click instalador Wampserver	***1.2 Aceptar***
	
1.3 Seleccionar directorio de instalación	***1.4 Instalar***
	

1.5 Seleccionar navegador	*1.6 Seleccionar dominio*
	

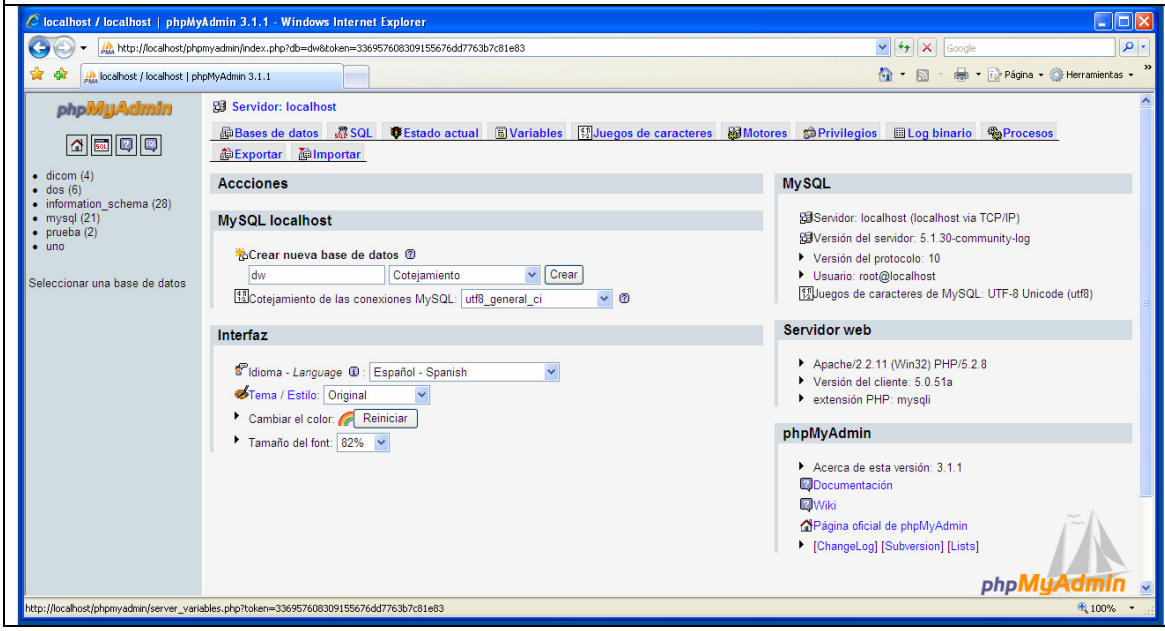
2. Iniciar WAMPSERVER (Inicio-Programas-WampServer-Start WampServer)
3. Se abre icono de WAMPSERVER en el inicio rápido, click derecho con el mouse y seleccionar ***www directory*** , copiar dentro del directorio la carpeta DWM (*Sistema Data Warehouse fixer*)

3.1 Abrir www directory	*3.2 Directorio*
	

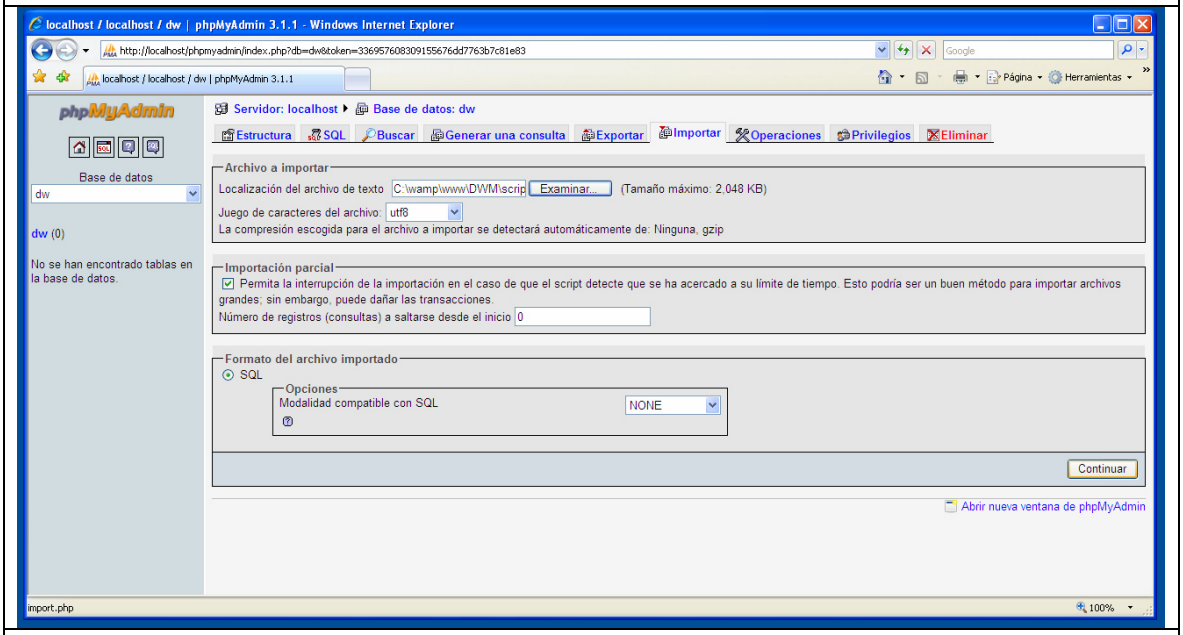
4. Sobre el icono de WAMPSERVER, click derecho con el mouse y seleccionar phpMyAdmin, crear base de datos ingresando el nombre *dw* en Crear base de datos, luego presionar botón 'crear'. Seleccionar **Importar** y por último en *Localización del archivo de texto* seleccionar script_dwm.txt ubicado en la carpeta DWM del paso anterior.

4.1 Abrir phpMyAdmin

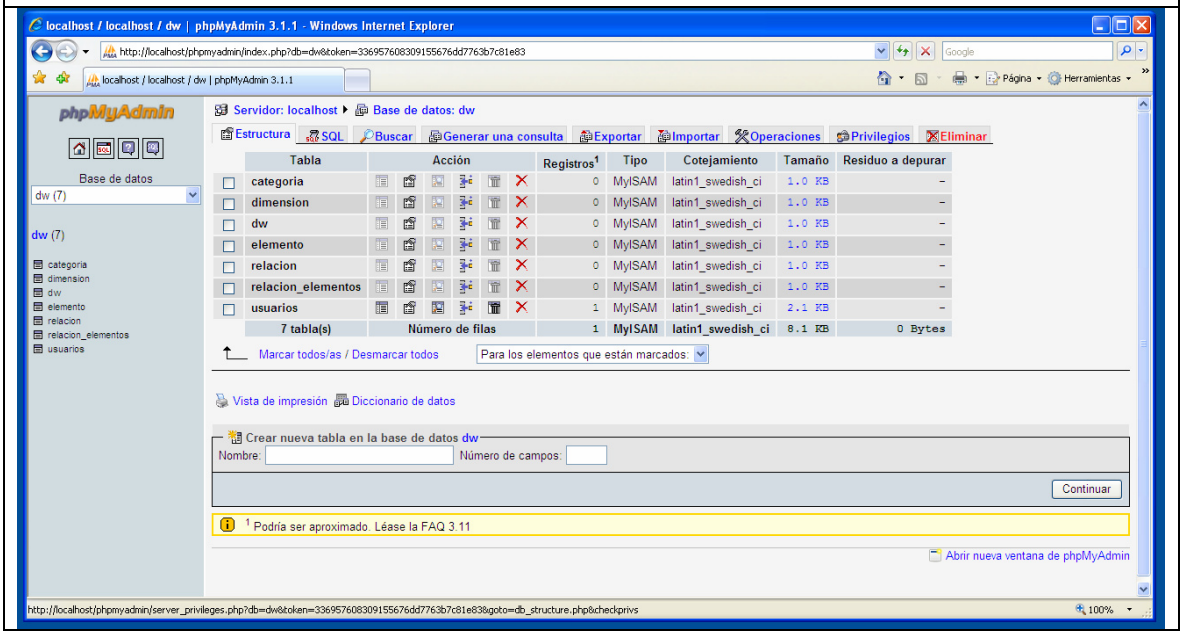
4.2 Crear base de datos



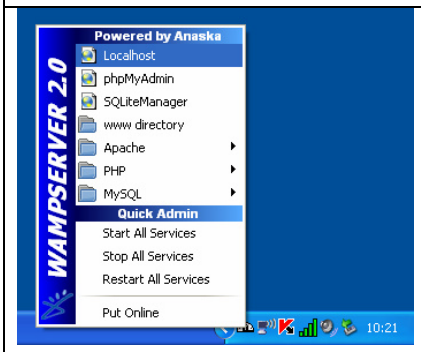
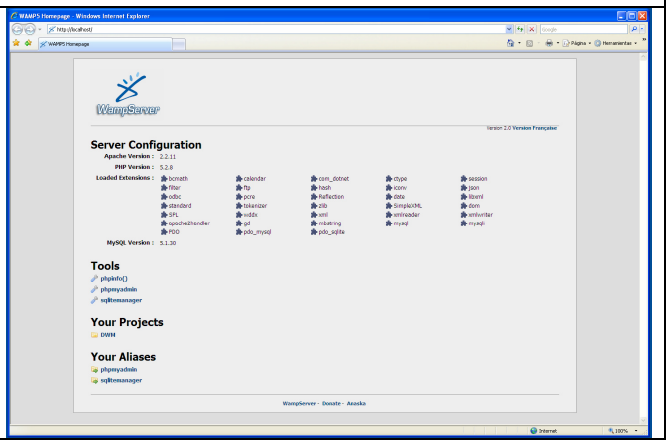
4.3 Importar script base datos



4.4 Base de datos creada



5. Seleccionar Localhost y a continuación en Your Proyects seleccionar ‘DWM’

5.1 Seleccionar localhost	*5.2 Pantalla localhost*
	

6. Una vez iniciado el servidor Apache también es posible ingresar a la aplicación escribiendo en cualquier navegador de Internet la dirección http://localhost/DWM/

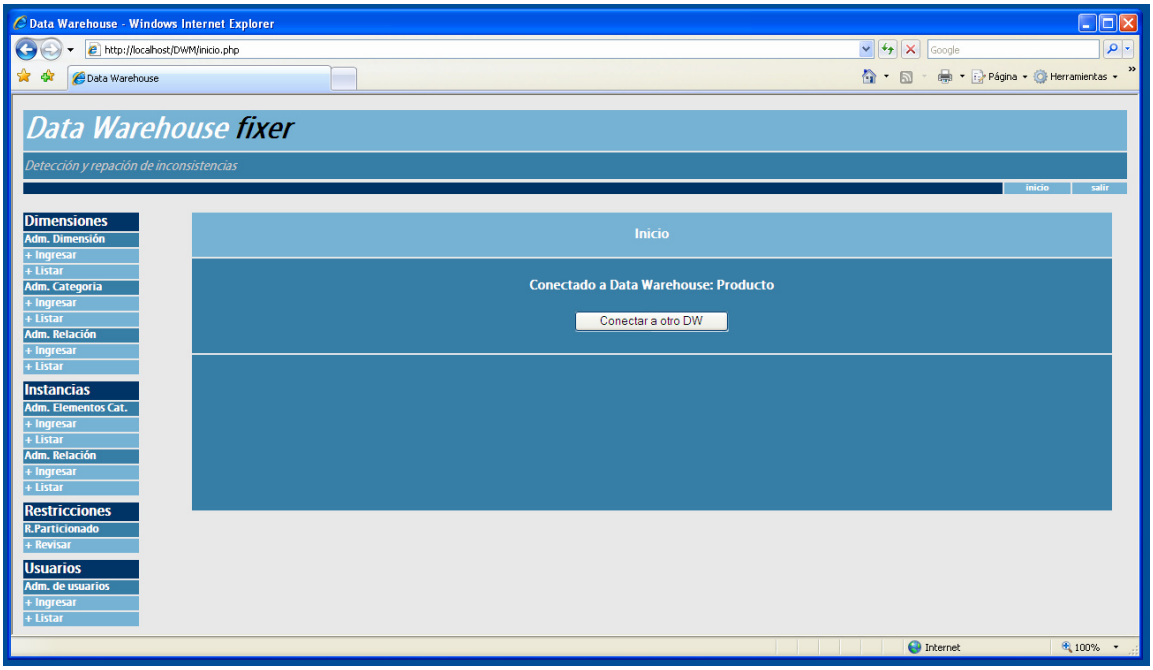


Figura 5.1 *Data Warehouse fixer* funcionando sobre Microsoft Windows Xp.

5.2 Instalación en Linux Ubuntu 8.04

1. Descargar XAMPP 1.6.2
2. Escribir en la Terminal

    - **cd /opt**
    - **sudo tar -zxvf xampp-linux-1.6.2.tar.gz**

3. Para iniciar XAMPP escribir en la terminal: **sudo /opt/lampp/lampp start**
4. Ubicar la ruta **opt/lampp/htdocs** y copiar en el directorio la carpeta DWM (*Sistema Data Warehouse fixer*)
5. Abrir en cualquier navegador de Internet la dirección http://localhost/phpmyadmin , luego seleccionar Importar y por último en *Localización del archivo de texto* seleccionar script_dwm.txt ubicado en la carpeta DWM.
6. Para ingresar al sistema abrir un navegador de Internet e ir a la dirección http://localhost/DWM

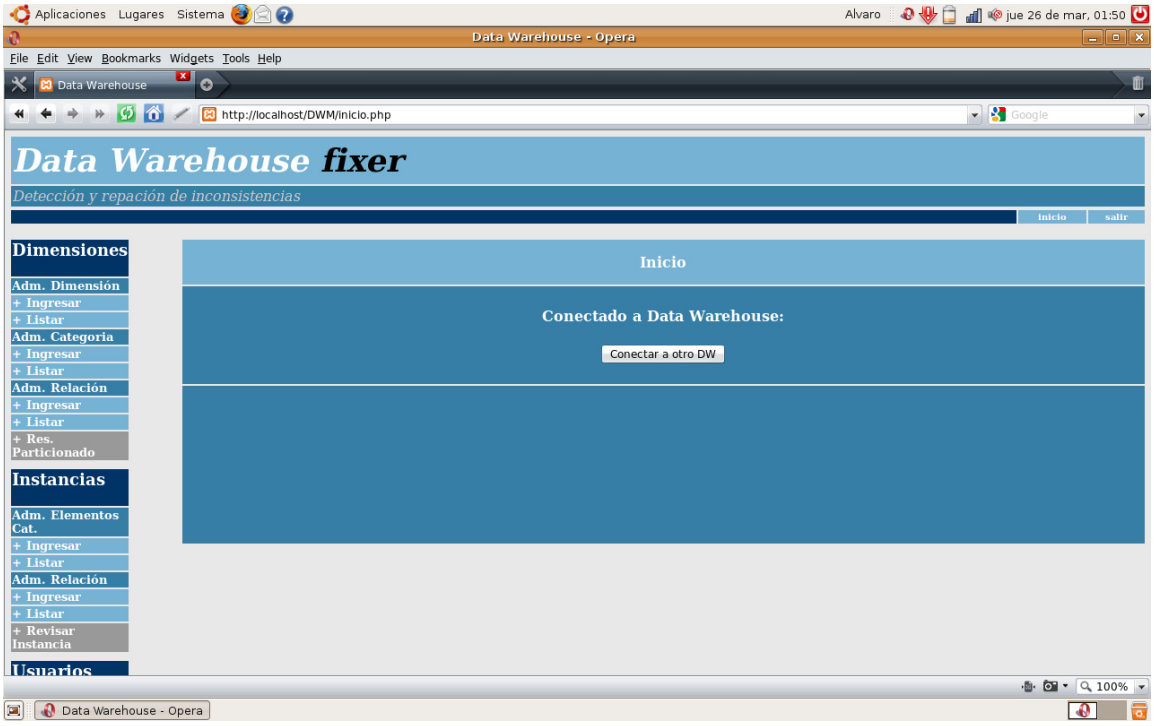


Figura 5.2 – *Data Warehouse fixer* (versión previa) funcionando sobre Linux Ubuntu 8.04.

Capítulo 6 – Conclusiones

En el presente documento ha dejado una clara explicación sobre la composición de los sistemas Data Warehouse, así como también su funcionalidad y forma de administración. Exponiendo mediante ejemplos la posibilidad de encontrar violaciones a restricciones dimensionales, y usando una semántica de reparación se ha mostrado cómo detectar inconsistencias y cómo dar solución a este inconveniente. Haciendo uso de los programas de reparación por medio de eliminaciones e inserciones de relaciones entre elementos, se logra retornar la consistencia a instancias de dimensión con inconsistencias. Por último se ha explorado la arquitectura de la aplicación WEB encargada de implementar la administración de sistemas DWM y tanto la detección como la reparación de inconsistencias a través de la computación de programas de reparación ejecutados externamente en DLV (Datalog con negación).

6.1 Investigación

6.1.1 Investigación del tema.

La investigación realizada en el Proyecto de Título ha sido de fundamental apoyo para poder comprender a totalidad los conceptos tratados en el entorno de los sistemas Data Warehouse. En este informe de Habilitación profesional se incorporó gran parte de la información recolectada en la primera etapa dejando solo los contenidos de mayor valor para sentar las bases del desarrollo de la aplicación encargada de reparar instancias dimensionales.

6.1.2 Investigación para desarrollo del software.

Para llevar a cabo el desarrollo de la aplicación Data Warehouse fixer se ha tenido que investigar sobre el manejo de ficheros y variables en el lenguaje de programación PHP,

esto con el fin de interactuar externamente con archivos de texto generados desde PHP y luego de ejecutar programas volver a leerlos obteniendo consigo respuestas necesarias para computar las reparaciones. La investigación también ha significado el estudio de programación lógica, específicamente Datalog con negación mediante el uso de DLV, programa de distribución libre de uso académico no comercial, encargado de computar los programas de reparación generados desde PHP.

Los conocimientos adquiridos durante la formación académica han significado terminar la implementación de una aplicación que considera múltiples factores, desde la programación de código en distintos lenguajes de programación, pasando por el diseño de sistemas, bases de datos, seguridad y administración de los mismos.

6.2 Desarrollo posterior

Este informe de Habilitación profesional junto con su respectivo software podrá servir para otros profesionales que investiguen en el área de Bases de datos y que necesiten realizar experimentos sobre sistemas Data Warehouse sin necesidad de adquirir software propietario o realizar intensas labores de instalación.

Agradecimientos

A mis padres y familia.

Y al proyecto FONDECYT número #11070186 que financió este trabajo.

Bibliografía

[1] Carlos Hurtado y Alberto Mendelzon. *OLAP Dimension Constraints. Symposium on Principles of Database Systems (PODS), Madison, USA, June 2002.* Páginas: 01-05,06,07.

[2] Mónica Caniupán *Optimizing and Implementing repair programs for consistent query answering in databases, Ottawa-Carleton Institute for Computer Science School of Computer Science at CARLETON UNIVERSITY Ottawa, Notario March, 2007. © Copyright by Mónica Caniupán, 2007. Capítulo Nº 8: A Repair Semantics for Multidimensional Databases.* Páginas: 160-180.

[3] Bertossi, L., Bravo, L., Caniupán, M. *Consistent Query Answering in Data Warehouses*. To appear in The Alberto Mendelzon Workshop on Foundations of Data Management. Arequipa, Perú, May 12-15, 2009. Páginas: 01-06.

[4] Carlos A. Hurtado, Alberto O. Mendelzon y Alejandro A. Vaisman. *Updating OLAP dimensions. In Proc. 2nd IEEE-DOLAP Workshop. 1999.* Páginas: 60-64.

[5] Carlos Hurtado, Claudio Gutiérrez (Universidad de Chile) y Alberto Mendelzon (University of Toronto). *Capturing Summarizability with Integrity Constraints in OLAP*. Páginas: 854-886.

[6] Carlos Hurtado, Alberto O. Mendelzon y Alejandro Vaisman. Maintaining Data Cubes under Dimension Updates. In Proc. 15th IEEE-ICDE Conference, Sydney, Australia,1999. Páginas: 346-357.

[7] W. H. Inmon, Wiley e hijos, *Building the Data Warehouse, 4th Edition.*

[8] Giannotti, F., Greco, S., Saccà. D., y Zaniolo, C. 1997. *Programming with Non-determinism in Deductive Databases*. Annals of Mathematics and Artifical Inteligence 19, 1-2, 97-125.

[9] Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gotttlob, G., Koch, C., Mateis, C., Perri, S., y Scarcello, F. 2006. The DLV System for Knowlodge Representation and Reasoning. ACM Transactions on Computational Logic 7,3, 499-562.