XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa
Buenos Aires, Setiembre 1988.

Artesano : Una arquitectura de administración de bases de datos basada en vistas con enfoque funcional

c.c. Paulina Frenkel,
Lic. Nicolás Casariego,
Lic. Jorge Zarauza

Servicio de Procesamiento de Datos SECYT - CONICET
Rivadavia 1906 2do piso (1033) Buenos Aires
TE: 48-7017 , 48-2173 TX: 18052 CICYT AR

RESUMEN

Se presenta un desarrollo de una arquitectura de administración de bases de datos basada en vistas con un enfoque funcional a nivel definición, manipulación y recuperación.

Por un lado este abordaje, que distingue explícitamente el nivel externo (visión de los usuarios) del nivel conceptual (modelo lógico), hace posible su implementación interactuando a nivel físico tanto con un DBMS relacional standard o directamente sobre un file manager más un desarrollo ad-hoc.

Por otro, la modalidad elegida apunta a :

- rescatar del modelo de relación universal RU la estructura de nombres de atributos como único conocimiento del usuario respecto del modelo conceptual.

- utilizar el grafo de dependencias funcionales respecto de las claves y las vinculaciones explícitas del modelo Entidad-Relación en los algoritmos de navegación de la base de datos.

- aprovechar la semántica de objetos tratada para lograr la actualización de vistas libre de efectos colaterales.

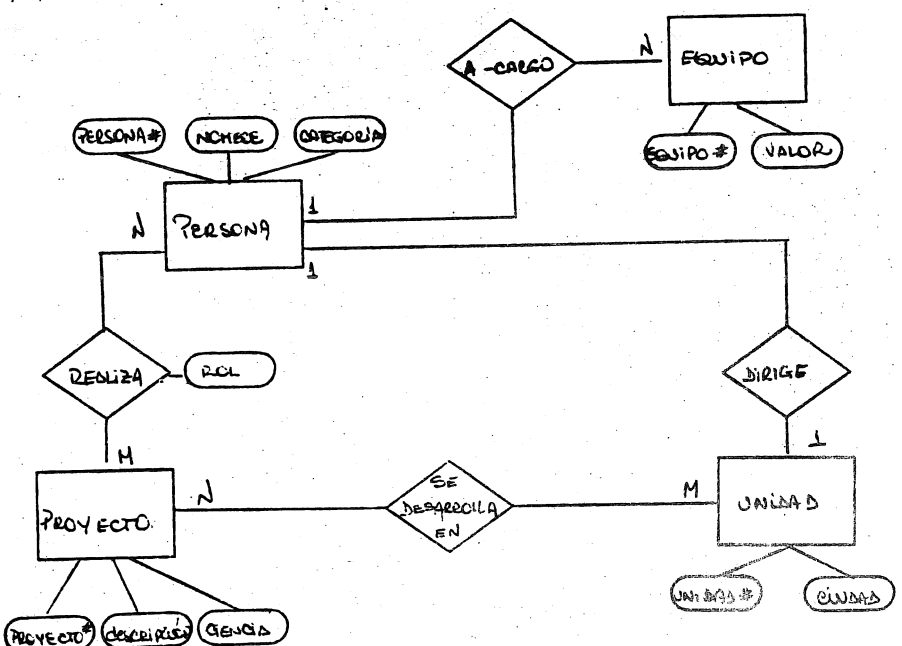
- proveer una interfase más natural de consulta.

EQUIVALENCIA DE MODELOS

Un modelo Entidad-Relación (ER) [CHE76] puede representarse mediante un conjunto de tablas en forma equivalente a un modelo relacional.
En una simplificación, para cada conjunto de entidades (CE) se puede definir una tabla (relación entidad [DAT87]), con sus atributos descriptivos, y con los atributos de identificación (identificador) como clave primaria.
Los CE débiles se representan por una tabla cuya clave primaria esta formada por la clave del CE que la sustenta más un componente discriminatorio (relación característica [DAT87]).
Las vinculaciones de 1:1 y de 1:N entre dos CE se representan incluyendo el identificador de uno, como atributo descriptivo del otro (clave foranea) y las vinculaciones de N:M como relaciones asociativas [DAT87] con la unión de los identificadores de los CE participantes, como clave primaria.
En el modelo resultante, denominado base de datos ER [BRA85], se verifican dos condiciones:

- Integridad de Entidad : ningún componente de una clave puede tener un valor nulo [DAT83].
- Integridad Referencial : una clave foranea o bien es nula o bien se corresponde con la clave primaria de una tupla de la relación foranea [DAT83].

Asumimos que los aspectos del mundo real que se modelan en la base de datos, han sido representados en un diagrama ER bien formado, previo a su traducción en un conjunto de tablas.
Tomemos como ejemplo el siguiente diagrama ER con vinculaciones binarias que representa la interacción entre personas que desarrollan proyectos en unidades de investigación. Estas personas son responsables de determinado equipamiento científico y pueden dirigir estas unidades y/o participar en los proyectos.



La base de datos ER queda compuesta por las tablas:

```
tablas
PERSONA          (persona#, nombre, categoria)
EQUIPO           (equipo#, valor, responsable#)
PROYECTO         (proyecto#, descripción, ciencia)
UNIDAD           (unidad#, ciudad, director#)
REALIZA          (realizador#, actividad#, rol)
SE-DESARROLLA-EN (desarrollo#, lugar#)
```

donde cada atributo representa el uso de un dominio dentro de cada tabla.
En nuestro caso, veamos qué atributos están asociados a un mismo dominio:

```
dominios
PERSONA-ID  (persona#, director#, realizador#, responsable#)
PROYECTO-ID (proyecto#, actividad#, desarrollo#)
UNIDAD-ID   (unidad#, lugar#)
EQUIPO-ID   (equipo#)
IMPORTE     (valor)
CODIGO      (rol, ciencia, grado, categoria)
TEXTO       (nombre, descripción, ciudad)
```

A partir del diagrama ER pueden explicitarse las dependencias funcionales entre atributos. La integridad de entidad garantiza la existencia de una dependencia funcional (DF) de los atributos respecto del identificador de un CE o lo que es equivalente de la clave de la tabla a la que pertenecen.

```
persona# -> nombre                equipo# -> valor
         -> categoria                     -> responsable#

proyecto# -> descripción          unidad#  -> ciudad
          -> ciencia                       -> director#

realizador#,actividad# -> rol
```

Tanto persona#, como responsable#, director# y realizador# son nombres distintos de atributos con valores pertenecientes a un mismo dominio: persona-id; al igual que lugar# y unidad# con respecto a unidad-id. Estos sinónimos los usamos cuando incluimos los atributos de una tabla en otra (claves foraneas restringidas por la integridad referencial), en un modelo de datos con exigencia de nombres únicos.
El uso de estos sinónimos, más las propiedades de la DF (reflexividad, aumentación y transitividad) nos definen otras dependencias funcionales tales como:

```
persona# -> persona#              director# -> persona#
         -> realizador#                     -> realizador#
         -> director#                       -> responsable#
         -> responsable#                    -> director#
                                            -> nombre
```

-> categoría

desarrollo#,lugar# -> desarrollo#
-> lugar#

En nuestro enfoque la DF se representa como una función denominada primitiva, en la que el atributo queda determinado genéricamente por el dominio al que pertenece la clave primaria de la cual depende [KEN81]; por ej.: Si el nombre depende funcionalmente de persona#, decimos que:

nombre(persona-id)

es la representación funcional de dicha dependencia, donde el cuerpo de la función es el atributo en cuestión y el parámetro de la función es el dominio al cual pertenece la clave primaria. Entonces a partir de un valor del dominio persona-id queda determinada una instancia del atributo nombre.
Esta función primitiva es la misma para:

persona# -> nombre
director# -> nombre
realizador# -> nombre
responsable# -> nombre

Las siguientes son funciones primitivas del ejemplo dado:

persona#(persona-id)
nombre(persona-id)
categoría(persona-id)

proyecto#(proyecto-id)
descripción(proyecto-id)
ciencia(proyecto-id)

realizador#(persona-id,proyecto-id)
actividad#(persona-id,proyecto-id)
rol(persona-id,proyecto-id)

desarrollo#(proyecto-id,unidad-id)
lugar#(proyecto-id,unidad-id)

equipo#(equipo-id)
valor(equipo-id)
responsable#(equipo-id)

unidad#(unidad-id)
ciudad(unidad-id)
director#(unidad-id)

En un esquema con nombres de atributos únicos, el conjunto de DF implementado como funciones primitivas, contiene la información suficiente para que baste la mención de esos nombres para identificarlos.

Esto esta fundamentado por la teoría de la Relación Universal RU [ULL83] [KEN81] [M&U82] [M&S88] donde un esquema queda definido por un conjunto finito de objetos, más un conjunto de dependencias funcionales respecto de la clave.
Un objeto es un conjunto de atributos, y se asume que todos los objetos son un subconjunto de un conjunto universal de atributos. Podemos asimilar el concepto de objeto del esquema RU al de una relación que representa un CE o vinculaciones entre éstos, donde cada atributo tiene asociado un dominio de igual forma que en el modelo relacional.

Artesano define estos objetos a través de un mecanismo de vistas

(tablas "virtuales" derivadas de proyecciones, juntas y/o uniones de las tablas básicas [DAT83]) con el conocimiento de las dependencias entre atributos expresadas en las funciones primitivas.
En el modelo relacional la existencia de vistas otorga mayor independencia de datos [DAT83] [COD79] al pertenecer a un nivel por encima del conceptual, logrando inmunidad ante los cambios tales como crecimiento y reestructuración del esquema.
Por otro lado, las vistas permiten al usuario tener una percepción más simplificada del universo modelado, al tiempo que preparan una vía para implementar la seguridad en cuanto a qué datos pueden ser accedidos por qué usuarios.
Artesano está concebido para interactuar con un esquema lógico preexistente haciendo posible su aplicabilidad tanto sobre un administrador de bases de datos standard, como sobre un "file manager" más un desarrollo ad-hoc.
En esta arquitectura basada en vistas, la correspondencia entre las actualizaciones sobre estas con actualizaciones sobre las tablas resultan complejas debido a que las vistas que tienen atributos asociados a un mismo dominio provenientes de tablas distintas, ocasionan actualizaciones interdependientes.
Según la complejidad de la vista puede ocurrir que una modificación de un atributo desencadene una sucesión de inserciones, modificaciones y/o eliminaciones en las diferentes tablas involucradas.
En efecto, se espera que la actualización de vistas actúe como si fuera sobre los objetos básicos de la base de datos, esto es, libre de efectos colaterales.
Para que esto ocurra es necesario restringir el tipo de vistas que se permiten definir. Hay varios enfoques estudiados por distintos autores [M&SBB], que van desde el más estricto, que consiste en definir vistas coincidentes con tablas, hasta el más flexible, por medio del cual se pueden definir vistas que garanticen el "mapping" de actualizaciones (inserción y eliminación), pasando por aquellas vistas que sólo garantizan el "mapping" de eliminaciones.
Artesano, en su primer versión, permite definir y manipular vistas que involucran atributos de varias tablas, con la restricción que cada atributo sea accesible a través de un camino de longitud uno dentro del grafo que representa la clausura [SAL86] respecto a la clave de la vista.
De esta manera es posible actualizar las vistas y respetar la integridad de la base de datos.

ENFOQUE FUNCIONAL

Una vista se crea, definiéndola mediante una expresión funcional que contiene un conjunto de atributos (como funciones primitivas), literales y/o vistas preexistentes. Los parámetros de la función determinan la clave de la vista, resultando de la unión de todos los parámetros de las funciones primitivas y/o vistas incluidas.

```
⟨definición⟩   ::= ⟨declaración⟩ = ⟨término⟩ [+⟨término⟩ ...]
⟨declaración⟩  ::= nombre-vista( dominio [,dominio ...] )
⟨término⟩      ::= literal /
                   atributo( ⟨argumento⟩ [,⟨argumento⟩ ...] ) /
                   vista( ⟨argumento⟩ [,⟨argumento⟩ ...] )
```

```
⟨argumento⟩    ::= dominio /
                   ⟨término⟩
```

Una vista se recupera mediante la invocación de dicha función usando como argumento efectivo un valor del dominio que tiene asociado la clave de la vista.

```
⟨invocación⟩   ::= nombre-vista( ⟨argumento⟩ [,⟨argumento⟩ ...] )
⟨argumento⟩    ::= valor  /
                   ↕
```

Siguiendo con nuestro ejemplo, explicitaremos la semántica del enfoque funcional mediante sentencias en SQL standard [ORA86] que producen un efecto equivalente, tanto en la definición como en la recuperación de las vistas.

Dadas las siguientes tablas:

PERSONA

persona#	nombre	categoria
100	paulina	1
200	nicolás	3
300	jorge	3

EQUIPO

equipo#	valor	responsable#
55	1000	300
56	500	200

UNIDAD

unidad#	ciudad	director
2111	buenos aires	300

PROYECTO

proyecto#	descripción	ciencia
AA	base de datos	MC
BB	lenguajes	MC

REALIZA

realizador#	actividad#	rol
100	AA	02
100	BB	01
200	BB	02

SE-DESARROLLA-EN

desarrollo#	lugar#
BB	CNEA
AA	CAICYT

En la definici^["^\n funcional vemos que no es necesario incluir dentro de la vista, los atributos que forman parte de la clave; ni mencionar la tabla de donde provienen todos los atributos.
En esta version de Artesano la recuperaci^["^\n o bien es de todo el conjunto o bien es de una instancia.

Definición:

vista1(persona-id) = nombre(persona-id) + categoria(persona-id)

CREATE VIEW vista1 AS

```
SELECT nombre, categoría, persona#
FROM   persona ;

Recuperación:

vista1(200) = nicolás , 3
SELECT nombre, categoría FROM vista1 WHERE persona# = 200 ;

vista1(100) = paulina , 1
SELECT nombre, categoría FROM vista1 WHERE persona# = 100 ;

vista1(*)   = paulina , 1
              nicolás , 3
              jorge   , 3
SELECT nombre, categoría FROM vista1 ;

-----------------------------------------------------------------

vista2(equipo-id)   = nombre(responsable#(equipo-id)) +
                      categoría(responsable#(equipo-id))

CREATE VIEW vista2 AS
SELECT nombre, categoría, equipo#
FROM   persona, equipo
WHERE  persona# = responsable# ;

vista2(55)  = jorge , 3
SELECT nombre, categoría FROM vista2 WHERE equipo# = 55 ;

-----------------------------------------------------------------
```

Dentro de la definici^["^\n de una vista se puede incluir una vista ya existente.

```
-----------------------------------------------------------------

vista2'(equipo-id)  = vista1(responsable#(equipo-id))

CREATE VIEW vista2' AS
SELECT nombre, categoría, equipo#
FROM   vista1, equipo
WHERE  persona# = responsable# ;

vista2'(55) = jorge , 3
SELECT nombre, categoría FROM vista2  WHERE equipo# = 55 ;

-----------------------------------------------------------------
```

Los atributos que intervienen en vistas con clave multi-dominio pueden depender de cualquiera de los dominios componentes o cualquier combinaci^["^\n de ellos.

```
-----------------------------------------------------------------

vista3(persona-id,proyecto-id) = rol(persona-id,proyecto-id) +
                                 nombre(persona-id) +
                                 descripción(proyecto-id)
```

```
CREATE VIEW vista3 AS
SELECT rol, nombre, descripción, persona#, proyecto#
FROM   realiza, persona, proyecto
WHERE  persona#  = realizador#
AND    proyecto# = actividad# ;


vista3(100,AA) = 02 , paulina , base de datos
SELECT rol, nombre, descripción FROM vista3
WHERE persona# = 100 AND proyecto# = 'AA' ;

vista3(100,*)  = 02 , paulina , base de datos
                 01 , paulina , lenguajes
SELECT rol, nombre, descripción FROM vista3 WHERE persona# = 100 ;

vista3(*,BB)   = 01 , paulina , lenguajes
                 02 , nicolás , lenguajes
SELECT rol, nombre, descripción FROM vista3 WHERE proyecto# = 'BB';
```

Los literales se utilizan en forma equivalente a una constante en el cuerpo de una funci^["^\n genérica, pudiendo ser usados como componentes de una vista o para instanciar una funci^["^\n primitiva.

```
vista4(unidad-id)   = nombre(director#(unidad-id)) +
                      'es el director de la unidad' +
                      unidad#(unidad-id)

CREATE VIEW vista4 (nombre,texto,unidad#) AS
SELECT nombre, 'es el director de la unidad', unidad#
FROM   persona, unidad
WHERE  persona# = director# ;

vista4(2111) = jorge es el director de la unidad 2111
SELECT nombre, texto, unidad# FROM vista4 WHERE unidad# = 2111 ;
```

CONCLUSIONES

Hacia un Lenguaje de Consulta más Natural

Una de las formas más simples de implementar una interfase de consulta y recuperación basada en un lenguaje "cuasi-natural" con un vocabulario controlado consiste en el reconocimiento de palabras clave en el contexto adecuado, descartando la "cáscara" sintáctica.
En nuestro enfoque, la disponibilidad de una estructura de nombres de atributos con un alto contenido semántico, preparan el camino para la construcción de interfases más amigables y simples

con el usuario.
La interpretación de un requerimiento expresado en un subconjunto del castellano, es uno de los próximos objetivos a encarar en la implementación de Artesano, luego, la definición de la vista surgiría de la interpretación del requerimiento.

Ejemplo:

- Requerimiento:

" quiero la descripción y la ciencia del proyecto 'AA', más el nombre del director de las unidades donde se desarrolla ".

- Interpretación como vista:

```
vista(proyecto-id,unidad-id) =
    descripción(proyecto-id) +
    ciencia(proyecto-id) +
    nombre(director#(lugar#(proyecto-id,unidad-id)))
```

- Recuperación:

```
vista(AA,?)
```

IMPLEMENTACION

Este trabajo está parcialmente implementado con la Base de Datos de Recursos Científico-Tecnológicos del Servicio de Procesamiento de Datos SECYT -CONICET.
Paralelamente, alumnos de la ESLAI en una pasantía dirigida por los autores " Diseño de un módulo de manejo de visiones y acceso a datos ", están construyendo, con un enfoque equivalente, un prototipo para interactuar con un file-manager.

ANEXO

Propiedades de la Dependencia Funcional [SAL86]

```
NOMENCLATURA : atributos                (minúsculas)
               conjunto de atributos    (mayúsculas)
               ->                       (determina funcionalmente)
               U                        (unión de conjuntos)

REFLEXIVIDAD o "Dependencia Trivial" :
     (x,y) -> x
     (x,y) -> y

AUMENTACION :
     X -> Y  implica  X U Z -> Y U Z

TRANSITIVIDAD :
     si  X -> Y  y  Y -> Z  entonces  X -> Z

"Atributos Extra en Parte Izquierda" :
     X -> Y  implica  X U Z -> Y

"Misma Parte Izquierda" :
     X -> Y  y  X -> Z  implica  X -> Y U Z

"Unico Atributo en Parte Derecha" :
     Z = { z1, z2, ... zn }
     X -> Z  si y solo si     X -> z1
                              X -> z2
                              ...
                              X -> zn

CLAUSURA de X :
     X+ = { y / X -> y }
```

REFERENCIAS BIBLIOGRAFIAS

[CHE76] P.P. CHEN
" The entity-relationship model - Towards a unified view of data "
ACM, Transactions on Databases System (March 1976).

[COD79] E.F. CODD
" Extending the database relational model to capture more meaning "
ACM Transactions on Database Systems Vol.4 N.4 (1979)

[KEN81] W. KENT
" Consequences of assuming a universal relation "
ACM Transactions on Database Systems Vol.6 N.4 (1981)

[M&U82] R. FAGIN & A.O. MENDELZON & J.D. ULLMAN
" A Simplified universal relation and its properties "
ACM Transactions on Database Systems Vol.7 N.3 (1982)

[ULL83] J.D. ULLMAN
" Universal relation interfases for databases systems "
Information Processing, IFIP (1983) North Holland.

[DAT83] C.J. DATE
" Introduction to Database Systems "
Addison-Wesley (1983)

[BRA85] L.I. BRADY
" A Universal relation assumption based on entities and relationships "
IEEE (1985).

[SAL86] B.J. SALZBERG
" An introduction to data base design "
Academic Press (1986).

[ORA86] J. SACHS
" Oracle: SQL*Plus user's guide " version 2.0
Oracle Corporation, California USA (1986)

[DAT87] C.J. DATE
" Relational Database "
Addison-Wesley Cap.19 (1987)

[M&S88] V.M. MARKOWITZ & A. SHOSHANI
" Abbreviated query interpretation in entity relationship oriented databases "
Computed Science Research Dept. Berkeley (1988).