MANTICA: Una herramienta para métricas de Bases de Datos Relacionales

Autores: **Manuel A. Serrano, Mario Piattini, Coral Calero,**
Marcela Genero, Francisco Ruiz
Ponente: Francisco Ruiz
Institución: Escuela Superior de Informática, Universidad de Castilla – La Mancha
Dirección Postal: Ronda de Calatrava s/n. 13071 Ciudad Real
País: España
Teléfono: +34 926 29 53 00 Ext: 3715

Fax: +34 926 29 53 54
E-mail: {mserrano, mpiattin, ccalero, mgenero, fruiz}@inf-cr.uclm.es
URL: http://alarcos.inf-cr.uclm.es

Resumen:

La medida del software es un mecanismo muy efectivo de gestionar proyectos de desarrollo de software. En décadas pasadas se han desarrollado una gran cantidad de métricas, pero se han centrado principalmente en la medida de programas. Actualmente las bases de datos se han convertido en una parte muy importante de los sistemas de información (SI), por ello es necesario desarrollar métricas específicas para ellas. Además es recomendable tener una herramienta que nos ayude a recoger y procesar los valores de estas métricas. Este artículo presenta la herramienta que estamos desarrollando para conseguir este objetivo.

I. Introducción

La medida del software es una forma efectiva de gestionar proyectos de desarrollo y mantenimiento de productos software. De hecho, actualmente, se reconoce que medir el software es una forma muy adecuada de entender, controlar, predecir y mejorar el desarrollo y el mantenimiento de productos software (Briand et al., 1996). En las últimas décadas se han propuesto una gran cantidad de métricas, pero se han centrado principalmente en los programas. Desgraciadamente, no hay métricas para bases de datos (Sneed y Foshag, 1998), incluso a pesar de que las bases de datos, especialmente las relacionales se han convertido en el corazón de los sistemas de información más relevantes.

Sin embargo, el diseño de una base de datos es un proceso largo y complicado (De Miguel y Piattini, 1993), incluso aunque se basa en los sólidos fundamentos matemático (Codd, 1970).

Aunque las metodologías de diseño de bases de datos garantizan alguna forma de "calidad" (como, por ejemplo, las formas normales en el modelo relacional), es necesario desarrollar nuevas métricas para evaluar la calidad de una base de datos, ya que el tamaño y la naturaleza de una base de datos influyen en muchos aspectos del sistema de información como, por ejemplo, el esfuerzo de desarrollo (MacDonell, S.G. et al., 1997). Además, estas medidas pueden ayudar a los profesionales e investigadores a tomar las mejores decisiones (Pfleeger, 1997), y a los diseñadores a elegir entre distintos esquemas de bases de datos alternativos.

Así pues, resulta fundamental de métricas específicas para los distintos tipos de bases de datos (Relacionales, Objeto-Relacionales, Entidad/Interrelación, etc), que permitan medir características propias de cada uno de ellos. No obstante, de esta forma conseguiríamos métricas genéricas por lo que también sería conveniente adaptarlas a las características

específicas de cada Sistema Gestor de Bases de Datos (SGBD) así como los entornos de desarrollo específicos (Metodologías de trabajo, guías de especificación, preferencias personales del equipo de desarrollo, etc.).

En cualquier caso, resulta imprescindible someter a las métricas a procesos de verificación formal (Briand et al. 1996) (Zuse, 1998) y empírica que nos permitan observar su utilidad.

Además, también es necesario realizar herramientas para automatizar la adquisición, la gestión, el análisis y la presentación de los valores de las métricas para conseguir que este proceso sea más cómodo, eficiente y exacto.

En las siguientes secciones presentamos algunas métricas para bases de datos relacionales y el diseño de una herramienta para automatizar la medida de las bases de datos.

II. Métricas para Bases de Datos Relacionales

Desde que a finales de los sesenta el Dr. Codd propusiera su modelo relacional (Codd, 1970), se ha intensificado la investigación en el campo de las bases de datos y los productos de bases de datos relacionales han generado una importante industria. Date (1995) define un sistema de gestión de base de datos relacional como "un sistema, en el que como mínimo:

- Los datos son vistos por el usuario como tablas (y sólo así) y
- Los operadores disponibles para el usuario generan nuevas tablas a partir de otras antiguas. Entre los operadores se encuentran, como mínimo la selección, (SELECT), la proyección (PROJECT) y la combinación (JOIN)".

El único indicador utilizado para medir la calidad de una base de datos relacional ha sido la teoría de la normalización, a partir de la cual Gray et al. (1991) proponen un ratio de normalidad.

En este artículo, proponemos las siguientes métricas para medir la complejidad de una base de datos relacional:

- **Número de atributos (*Number of Attributes NA*).** NA(A) es el número de atributos de la tabla A.
- **Grado de Referenciabilidad (*Referentiability Degree RD*).** RD(A) es el número de claves ajenas de la tabla A.
- **Profundidad del árbol referencial (*Depth Referential Tree DRT*).** DRT(A) se define como la longitud del máximo camino referencial de la tabla A. Los ciclos sólo se consideran una vez.

Aplicamos las métricas definidas al ejemplo de la tabla 1, tomado de Elmasri y Navathe (1997):

```
CREATE TABLE EMPLEADO (
NOMBREP VARCHAR(15) NOT NULL,
INIC CHAR,
APELLIDO VARCHAR(15) NOT NULL,
NSS CHAR(9) NOT NULL,
FECHAEN DATE,
DIRECCION VARCHAR(30),
SEXO CHAR,
SALARIO DECIMAL(10,2),
NSSSUPER CHAR(9),
ND INT NOT NULL,
CONSTRAINT CLPEMP
PRIMARY KEY (NSS),
CONSTRAINT CLESUPEREMP
FOREIGN KEY (NSSSUPER)
REFERENCES EMPLEADO(NSS)
ON DELETE SET NULL ON UPDATE
CASCADE,
CONSTRAINT CLEDEPTOEMP
FOREIGN KEY (ND) REFERENCES
DEPARTAMENTO (NUMEROD)
ON DELETE SET DEFAULT ON
UPDATE CASCADE);
CREATE TABLE DEPARTAMENTO (
NOMBRED VARCHAR(15) NOT NULL,
NUMEROD INT NOT NULL,
NSSGTE CHAR(9) NOT NULL,
FECHAINI CGTE DATE,
CONSTRAINT CLPDEPTO
```

```
CREATE TABLE LUGAR_DEPTS (
NUMEROD INT NOT NULL,
LUGARD VARCHAR(15) NOT NULL,
PRIMARY KEY (NUMEROD, LUGARD),
FOREIGN KEY (NUMEROD) REFERENCES DEPARTAMENTO
(NUMEROD)
ON DELETE CASCADE ON UPDATE CASCADE);
CREATE TABLE PROYECTO (
NOMBREPR VARCHAR(15) NOT NULL,
NUMEROPR INT NOT NULL,
LUGARPR VARCHAR(15),
NUMD INT NOT NULL,
PRIMARY KEY (NUMEROP),
UNIQUE (NOMBREP),
FOREIGN KEY (NUMD) REFERENCES DEPARTAMENTO
(NUMEROD));
CREATE TABLE TRABAJA_EN (
NSSE CHAR(9) NOT NULL,
NUMP INT NOT NULL,
HORAS DECIMAL(3,1) NOT NULL,
PRIMARY KEY (NSSE, NUMP),
FOREIGN KEY (NSSE) REFERENCES EMPLEADO (NSS),
FOREIGN KEY (NUMP) REFERENCES PROYECTO
(NUMEROP));
CREATE TABLE DEPENDIENTE (
NSSE CHAR(9) NOT NULL
NOMBRE_DEPEND VARCHAR(15) NOT NULL,
SEXO CHAR,
FECHAAN DATE,
RELACION VARCHAR(8),
PRIMARY KEY (NSSE, NOMBRE_DEPEND),
FOREIGN KEY (NSSE) REFERENCES EMPLEADO (NSS));
```

PRIMARY KEY (NUMEROD), **CONSTRAINT CLSDEPTO** **UNIQUE (NOMBRED),** **CONSTRAINT CLEGTEDEPTO** **FOREIGN KEY (NSSGTE)** **REFERENCES EMPLEADO(NSS)** **ON DELETE SET DEFAULT ON** **UPDATE CASCADE);**	

Tabla 1. Ejemplo

En la figura 1 se presenta el grafo relacional para el ejemplo anterior, donde las flechas indican las relaciones de integridad referencial.

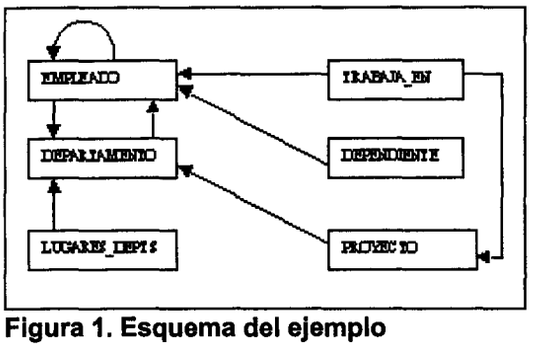

Figura 1. Esquema del ejemplo

	NA	*RD*	*DRT*
EMPLEADO	10	2	3
DEPARTAMENTO	4	1	3
LUGARES_DEPTS	2	1	4
PROYECTO	4	1	4
TRABAJA_EN	3	2	5
DEPENDIENTE	5	1	4
ESQUEMA	**28**	**8**	**5**

Tabla 2. Valores de las métricas para el ejemplo

Los valores de las métricas correspondientes al ejemplo se muestran en la tabla 2, dónde podemos observar todos los valores obtenidos para las métricas a partir de la información que nos da tanto el ejemplo como su esquema. Por ejemplo, el valor para la métrica DRT para la tabla TRABAJA_EN es cinco, que se obtiene del camino referencial siguiente (conviene recordar que, por definición, nuestra métrica DRT sólo considera los ciclos una vez):

TRABAJA_EN → PROYECTO → DEPARTAMENTO → EMPLEADO → EMPLEADO → DEPARTAMENTO.

III. Arquitectura de una herramienta para métricas de bases de datos

El objetivo de nuestra herramienta es adquirir, gestionar, procesar y representar los valores de un conjunto de métricas aplicadas al esquema de una base de datos. La herramienta debe ser independiente del modelo de datos utilizado para representar el esquema; de esta manera, podremos medir un esquema conceptual, lógico o físico. Además debe ser independiente de la plataforma de manera que podamos usar la herramienta para medir bases de datos implementadas en diferentes SGBD.

Debido a estas restricciones, nuestra herramienta está realizada de manera que el usuario pueda especificar la sintaxis de la base de datos que desea medir y las métricas que desea calcular.

La herramienta tiene un módulo para realizar la gestión de la interfaz de usuario, el cual se comunica con los otros módulos de métricas, de manera que todos los valores de las métricas se representen de la misma manera. Todos los módulos de métricas necesitan acceder a una base de datos central, que contiene información necesaria para poder calcular los valores de las métricas.

La herramienta tiene una metabase de datos para almacenar la información de los esquemas de bases de datos que queremos medir. El uso de una metabase de datos tiene bastantes beneficios:

- Podemos simplificar el proceso de medida ya que la herramienta sólo necesita procesar la información que se encuentra en su propia metabase de datos.
- Podemos realizar estudios estadísticos con un conjunto de esquemas almacenados en la metabase de datos.
- La herramienta puede medir cualquier tipo de base de datos; sólo es necesario realizar una pequeña aplicación para convertir el esquema de la base de datos que queremos medir a nuestra base de datos. (Ver figura 2).
- Es posible aplicar nuevas métricas al mismo esquema de base de datos sin tener que acceder a la base de datos real; sólo necesitamos capturar el esquema de la base de datos una vez.

En la figura 3 podemos observar el aspecto que presenta nuestra herramienta.

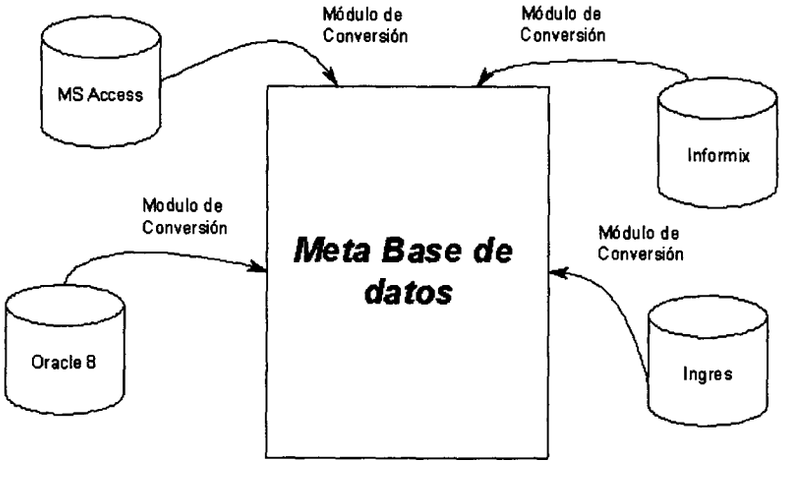

Figura 2. Proceso de conversión

Figura 3. Pantalla de la herramienta

IV. Metabase de datos

El diseño de la metabase de datos permite representar cualquier tipo de esquema de base de datos. Cualquier objeto del esquema es un "elemento" de un "tipo de elemento" concreto, que tiene relaciones con otros elementos. Además cada elemento tiene definidas un conjunto de "métricas" y tiene unos resultados para estas métricas (Ver figura 4). Con este diseño podemos representar cualquier esquema de bases de datos, así como los valores de las métricas que ese esquema ha obtenido para propósitos relacionales.

Para calcular los valores de las métricas de un esquema específico, necesitamos usar algunas vistas de la base de datos y convertir los datos almacenados en la base de datos a una estructura en memoria en forma de grafo, ya que la mayoría de las métricas necesitan utilizar la teoría de grafos para calcular sus valores.

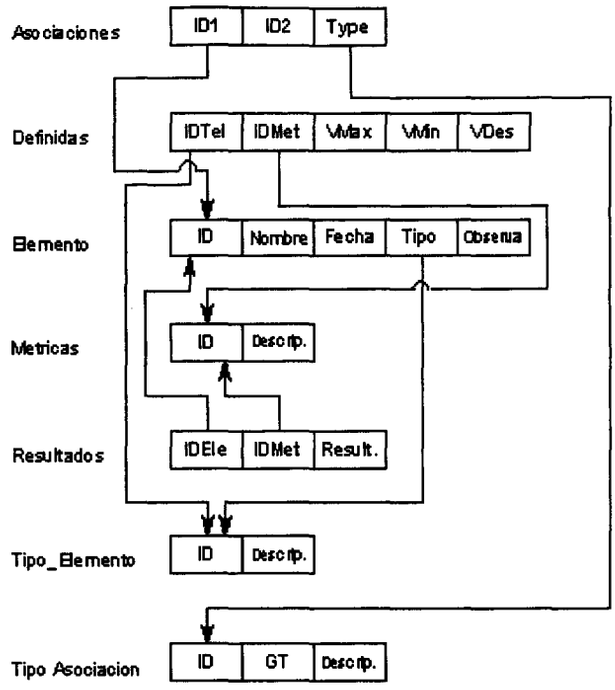

Figura 4. Esquema Relacional de la metabase

V. Representación de un esquema relacional en la metabase de datos

Para representar un esquema relacional en nuestra base de datos, tenemos que añadir registros según se muestra en la siguiente tabla:

Tabla Elementos	Tabla Asociaciones
Se introduce un registro por cada: • Esquema BDR (EBDR) • Tabla (T) • Columna (C) • Clave primaria (PK) • Clave ajena (FK)	Se introduce un registro por cada: • Tabla que pertenece a un Esquema (TEBDR) • Columna que pertenece a una Tabla (CT) • Clave primaria que pertenece a una Tabla (PKT) • Clave ajena que pertenece a una Tabla (FKT) • Columna que pertenece a una Clave primaria (CPK) • Columna que pertenece a una Clave ajena (CFK) • Clave ajena que apunta a una Tabla (FKT) • Columna que apunta a un Columna (CCFK)

Tabla 3. Reglas para almacenar elementos en la metabase

Una vez que tenemos almacenada la información, las vistas de nuestro esquema relacional se consiguen mediante las siguientes consultas:

Esquemas

Muestra todos los Esquemas Relacionales almacenados en la MetaBD

```
SELECT
Elemento.ID_ele AS ID,
Elemento.Nombre_ele AS Nombre,
Elemento.Fecha_ele AS Fecha

FROM
Elemento

WHERE
(((Elemento.Tipo_ele)='EBDR'));
```

Tablas

Muestra todas las tablas de un Esquema Relacional

```
SELECT
Elemento.ID_ele AS ID,
Elemento.Nombre_ele AS Nombre,
Elemento.Fecha_ele AS Fecha

FROM Elemento
INNER JOIN (Asociaciones
```

```
INNER JOIN Elemento AS Elemento_1
ON Asociaciones.ID2_aso = Elemento_1.ID_ele)
ON Elemento.ID_ele = Asociaciones.ID1_aso

WHERE
(((Elemento.Tipo_ele)='T')
AND ((Asociaciones.ID2_aso)=[N° Esquema?])
AND ((Asociaciones.Tipo_aso)='TEBDR')
AND ((Elemento_1.Tipo_ele)='EBDR'));
```

Columnas por Tabla

Muestra todas las columnas de todas las tablas de un Esquema Relacional

```
SELECT Elemento.ID_ele AS ID,
Elemento_1.Nombre_ele AS Tabla,
Elemento.Nombre_ele AS Columna,
Elemento.Fecha_ele AS Fecha

FROM Elemento
INNER JOIN (((Asociaciones
INNER JOIN Elemento AS Elemento_1
ON Asociaciones.ID2_aso = Elemento_1.ID_ele)
INNER JOIN Asociaciones AS Asociaciones_1
ON Elemento_1.ID_ele = Asociaciones_1.ID1_aso)
INNER JOIN Elemento AS Elemento_2
ON Asociaciones_1.ID2_aso =
Elemento_2.ID_ele)
ON Elemento.ID_ele = Asociaciones.ID1_aso

WHERE
(((Elemento.Tipo_ele)='C')
AND ((Asociaciones.Tipo_aso)='CT')
AND ((Elemento_1.Tipo_ele)='T')
AND ((Asociaciones_1.Tipo_aso)='TEBDR')
AND ((Asociaciones_1.ID2_aso)=[N° Esquema?])
AND ((Elemento_2.Tipo_ele)='EBDR'))

ORDER BY
Elemento_1.Nombre_ele;
```

Claves primarias por Tabla

Muestra las columnas que forman la clave primaria de cada una de las tablas de un esquema relacional

```
SELECT
Elemento.ID_ele AS ID,
Elemento_1.Nombre_ele AS Tabla,
Elemento.Nombre_ele AS PK,
Elemento.Fecha_ele AS Fecha

FROM Elemento
INNER JOIN ((((Asociaciones
INNER JOIN Elemento AS Elemento_2
ON Asociaciones.ID2_aso = Elemento_2.ID_ele)
INNER JOIN (Asociaciones AS Asociaciones_1
INNER JOIN Elemento AS Elemento_1
ON Asociaciones_1.ID2_aso =
Elemento_1.ID_ele)
ON Elemento_2.ID_ele = Asociaciones_1.ID1_aso)
```

```
INNER JOIN Asociaciones AS Asociaciones_2
ON Elemento_1.ID_ele =
Asociaciones_2.ID1_aso)
INNER JOIN Elemento AS Elemento_3
ON Asociaciones_2.ID2_aso =
Elemento_3.ID_ele)
ON Elemento.ID_ele = Asociaciones.ID1_aso

WHERE
(((Elemento.Tipo_ele)='C') AND ((Asociaciones.Tipo_aso)='CPK') AND
((Asociaciones_1.Tipo_aso)='PKT')
AND ((Elemento_1.Tipo_ele)='T')
AND ((Elemento_2.Tipo_ele)='PK')
AND ((Asociaciones_2.Tipo_aso)='TEBDR')
AND ((Elemento_3.Tipo_ele)='EBDR')
AND ((Elemento_3.ID_ele)=[N° Esquema?]))

ORDER BY
Elemento_1.Nombre_ele;
```

Claves ajenas por Tabla

Muestra las columnas que forman cada una de las claves ajenas de cada una de las tablas de un esquema relacional

```
SELECT
Elemento.ID_ele AS ID,
Elemento_1.Nombre_ele AS Tabla,
Elemento.Nombre_ele AS FK,
Elemento.Fecha_ele AS Fecha

FROM Elemento
INNER JOIN ((((Asociaciones
INNER JOIN Elemento AS Elemento_2
ON Asociaciones.ID2_aso = Elemento_2.ID_ele)
INNER JOIN (Asociaciones AS Asociaciones_1
INNER JOIN Elemento AS Elemento_1
ON Asociaciones_1.ID2_aso =
Elemento_1.ID_ele)
ON Elemento_2.ID_ele = Asociaciones_1.ID1_aso)
INNER JOIN Asociaciones AS Asociaciones_2
ON Elemento_1.ID_ele =
Asociaciones_2.ID1_aso)
INNER JOIN Elemento AS Elemento_3
ON Asociaciones_2.ID2_aso =
Elemento_3.ID_ele)
ON Elemento.ID_ele = Asociaciones.ID1_aso

WHERE
(((Elemento.Tipo_ele)='C')
AND ((Asociaciones.Tipo_aso)='CFK')
AND ((Asociaciones_1.Tipo_aso)='FKT')
AND ((Elemento_1.Tipo_ele)='T')
AND ((Elemento_2.Tipo_ele)='FK')
AND ((Asociaciones_2.Tipo_aso)='TEBDR')
AND ((Elemento_3.Tipo_ele)='EBDR')
AND ((Elemento_3.ID_ele)=[N° Esquema?]))

ORDER BY
Elemento_1.Nombre_ele;
```

Claves ajenas por Tabla por Destino

Devuelve la correspondencia de columnas de cada uno de los FKs de un esquema relacional

```
SELECT
Elemento_5.ID_ele AS ID_FK,
Elemento_1.Nombre_ele AS Tabla,
Elemento.ID_ele AS ID_Campo,
Elemento.Nombre_ele AS FK,
Elemento_3.Nombre_ele AS TDestino,
Elemento_2.ID_ele AS ID_Destino,
Elemento_2.Nombre_ele AS Destino,
Elemento.Fecha_ele AS Fecha

FROM ((((Asociaciones AS Asociaciones_2
INNER JOIN Elemento AS Elemento_2
ON Asociaciones_2.ID2_aso = Elemento_2.ID_ele)
INNER JOIN Elemento
ON Asociaciones_2.ID1_aso = Elemento.ID_ele)
INNER JOIN Asociaciones AS Asociaciones_3
ON Elemento_2.ID_ele = Asociaciones_3.ID1_aso)
INNER JOIN Elemento AS Elemento_3
ON Asociaciones_3.ID2_aso = Elemento_3.ID_ele)
INNER JOIN (Asociaciones
INNER JOIN (Elemento AS Elemento_5
INNER JOIN (((Asociaciones AS Asociaciones_1
INNER JOIN Elemento AS Elemento_1
ON Asociaciones_1.ID2_aso = Elemento_1.ID_ele)
INNER JOIN Asociaciones AS Asociaciones_4
ON Elemento_1.ID_ele = Asociaciones_4.ID1_aso)
INNER JOIN Elemento AS Elemento_4
ON Asociaciones_4.ID2_aso =
Elemento_4.ID_ele)
ON Elemento_5.ID_ele = Asociaciones_1.ID1_aso)
ON Asociaciones.ID2_aso = Elemento_5.ID_ele)
ON Elemento.ID_ele = Asociaciones.ID1_aso

WHERE
(((Elemento.Tipo_ele)='C')
AND ((Asociaciones.Tipo_aso)='CFK')
AND ((Asociaciones_1.Tipo_aso)='FKT')
AND ((Elemento_1.Tipo_ele)='T')
AND ((Asociaciones_2.Tipo_aso)='CCFK')
AND ((Asociaciones_3.Tipo_aso)='CT')
AND ((Elemento_3.Tipo_ele)='T')
AND ((Asociaciones_4.Tipo_aso)='TEBDR')
AND ((Elemento_4.Tipo_ele)='EBDR')
AND ((Elemento_4.ID_ele)=[N° Esquema?])
AND ((Elemento_5.Tipo_ele)='FK'))

ORDER BY
Elemento_1.Nombre_ele;
```

A continuación veremos un ejemplo de cómo representar un esquema relacional en nuestra metabase de datos.

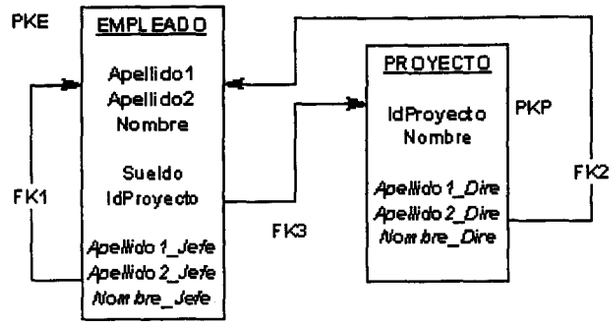

Figura 5. Esquema relacional de ejemplo

La representa
sería:

Tipo Elemento	
ID_tel	Descripción_tel
C	Columna
EBDR	Esquema BDR
FK	Clave ajena
PK	Clave primaria
T	Tabla

Tipo Asociacion		
ID_tas	TC_tas	Descripción_tas
CCFK	S	Columna apunta a Columna (FK)
CFK	A	Columna pertenece a Clave ajena
CPK	A	Columna pertenece a Clave primaria
CT	A	Columna pertenece a Tabla
FK-T	S	Clave ajena apunta a Tabla
FKT	A	Clave ajena pertenece a Tabla
PKT	A	Clave primaria pertenece a Tabla
TEBDR	A	Tabla Pertenece a Esquema BDR

Elemento

Id_ele	Nombre_ele	Fecha_ele	Tipo_ele
1	Empleado	28/07/99	T
2	Proyecto	28/07/99	T
3	Apellido1	28/07/99	C
4	Apellido2	28/07/99	C
5	Nombre	28/07/99	C
6	Sueldo	28/07/99	C
7	Apellido1_Jefe	28/07/99	C
8	Apellido2_Jefe	28/07/99	C
9	Nombre_Jefe	28/07/99	C
10	Id_Proyecto	28/07/99	C
11	Nombre	28/07/99	C
12	Apellido1_Dir	28/07/99	C
13	Apellido2_Dir	28/07/99	C
14	Nombre_Dir	28/07/99	C
15	PKE	28/07/99	PK
16	PKP	28/07/99	PK
17	FK1	28/07/99	FK
18	FK2	28/07/99	FK
19	FK3	28/07/99	FK
20	Id_Proyecto	28/07/99	C
21	Esquema BDR Ejemplo 1	28/07/99	EBDR

Asociaciones

Id1_aso	Id2_aso	Tipo_aso
1	21	TEBDR
2	21	TEBDR
3	1	CT
3	15	CPK
4	1	CT
4	15	CPK
5	1	CT
5	15	CPK
6	1	CT
7	1	CT
7	3	CCFK
7	17	CFK
8	1	CT

8	4	CCFK
8	17	CFK
9	1	CT
9	5	CCFK
9	17	CFK
10	2	CT
10	16	CPK
11	2	CT
12	2	CT
12	3	CCFK
12	18	CFK
13	2	CT
13	4	CCFK
13	18	CFK
14	2	CT
14	5	CCFK
14	18	CFK
15	1	PKT
16	2	PKT
17	1	FK-T
17	1	FKT
18	1	FK-T
18	2	FKT
19	1	FKT
19	2	FK-T
20	1	CT
20	10	CCFK
20	19	CFK

VI. Conclusiones y trabajo futuro

Las bases de datos, especialmente las relacionales, se han convertido en el corazón de los sistemas de información actuales más relevantes.

La medida del software es una de las mejores formas de controlar y gestionar el desarrollo de productos software. Desgraciadamente la mayoría de las métricas propuestas se han centrado en los programas, dejando de lado la medida de las bases de datos.

Necesitamos crear métricas para bases de datos, de manera que podamos ayudar en la gestión de proyectos de desarrollo de sistemas de información. Por lo tanto es necesaria una mayor investigación en el campo de las métricas para bases de datos, para que podamos abordar distintos modelos y distintos SGBD.

Pero, no basta con proponer métricas, es necesario validarlas formalmente para asegurarnos que las métricas son correctas. No obstante, también es necesario validarlas empíricamente para comprobar que realmente tienen una utilidad práctica.

Una vez que tenemos métricas, es recomendable tener una herramienta que nos ayude en la extracción de los valores de las métricas, su gestión, representación y análisis.

Por supuesto, también necesitamos mejorar la herramienta para que sea capaz de distintos modelos de bases de datos y distintos SGBD, de manera que podamos tener una herramienta más versátil, flexible y útil.

Referencias

- Briand, L. Morasca, S. Y Basili, V. (1996) "Property-Based Software Engineering Measurement". *IEEE Transactions on Software Engineering,* Vol. 22, No 1, January, 1996.

- Codd, E. F. (1970). *A Relational Model of Data for Larged Shared Data Banks.* CACM, 13 (6), 377-387.

- Date, C. J. (1995). *An Introduction to Database Systems.* 6th. Addison-Wesley, Reading, Massachusetts.

- De Miguel, A. Y Piattini, M. (1993), *Concepción y diseño de bases de datos relacionales,* Ed. Addison-Wesley.

- Elsmasri, R. Y Navathe, S. (1997). *Database Systems.* Addison-Wesley. Massachussets.

- MacDonell, S. G., Shepperd, M. J. Y Sallis, P.J. (1997). *Metrics for Database Systems: An Empirical Study.* Proc. Fourth International Software Metrics Symposium – Metrics' 97, Alburquerque. IEEE Computer Society, pp. 99-107.

- Pfleeger, S. L. (1997). *Assessing Software Measurement.* IEEE Software. Marzo/Abril pp.25-26.

- Sneed, H. M. Y Foshag, O. (1998). Measuring Legacy Databases Structures. Proc of *The European Software Measurement Conference FESMA 1998,* Antwerp, May 6-8, Coombes, Van Huysduyen and Peeters (eds.), 199-211.

- Zuse, H. (1998). *A framework of Software Measurement.* Berlin, Walter de Gruyter.

Agradecimientos

Este articulo forma parte del proyecto MANTICA, soportado parcialmente por CICYT y la Unión Europea (1FD97-0168).

Manuel Angel Serrano disfruta actualmente de una beca de investigación financiada por el proyecto MANTEMA (ATYCA, ATOS-ODS).