



Bisoños Usuarios de GNU/Linux de Mallorca y Alrededores | Bergantells Usuaris de GNU/Linux de Mallorca i Afegitons

Diagramas Entidad-Relación en web. (147196 lectures)

Per Joan Miquel, [Joanmi](http://www.mallorcaweb.net/joanmiquel) (<http://www.mallorcaweb.net/joanmiquel>)

Creado el 30/03/2004 21:10 modificado el 30/03/2004 21:10



Si alguna vez os habéis visto en la necesidad de diseñar y mantener una base de datos relacional, sabréis lo importante que es una buena documentación de todo el proceso de diseño, desde el modelo Entidad-Relación hasta el código SQL final. Y sobre todo lo mucho que cuesta mantener esta documentación al día cada vez que hacemos modificaciones a la base de datos y, muy especialmente en proyectos de software libre.

Existen varias herramientas gráficas que permiten representar el modelo Entidad-Relación de una base de datos. Algunas de ellas incluso pueden, con mayor o menor acierto, generarnos el modelo relacional a partir del diagrama E-R. Todas están muy bien para modelos pequeños como la típica biblioteca o sistema de alquiler de películas. Pero cuando se trata de proyectos más complejos, representar el modelo de datos de una manera legible resulta cada vez más difícil. Y las modificaciones y ampliaciones resultan cada vez más costosas.

Dividir los esquemas en bloques más pequeños puede ayudar a facilitar la comprensión, pero multiplica el trabajo y dificulta el estudio. Al final, cansado de tantas "(f)eines"* gráficas, he optado por implementarme mi propia solución. Y creo que el resultado ha valido la pena :-)

(*) En catalan: *eines* = herramientas y *feines* = trabajos.

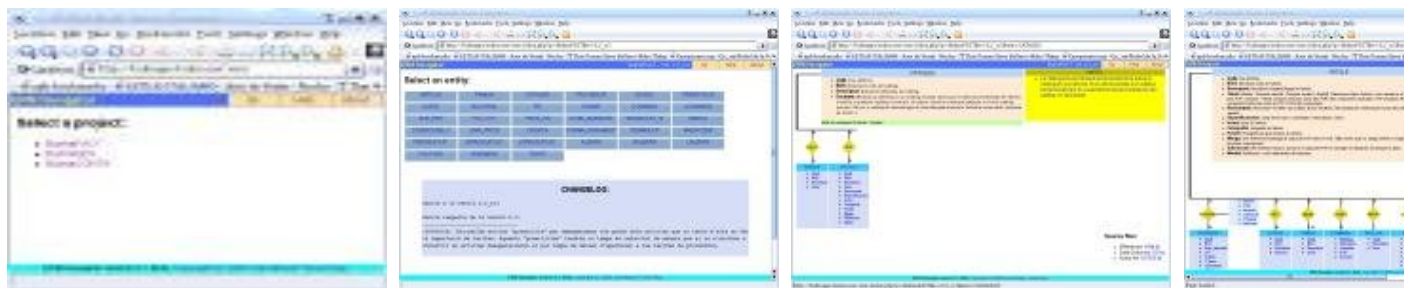
([Versió original en català](#)⁽¹⁾)

Índice:

1. Para los impacientes.
2. Origen de la idea.
3. Instalación.
4. Como generar los diagramas.
5. Funcionalidades futuras.
6. Como colaborar.

Para los impacientes:

Para los impacientes, aqui tenéis un par de screenshots:



(6)

...O también lo podáis ver "en acción" en:

<http://bulmages.bulma.net/erm/>⁽⁷⁾

Origen de la idea:

Hace poco he sido "llamado a filas" al proyecto Bulmagés porque se estaba pensando en arrancar el módulo de facturación y TPV (actualmente Bulmagés sólo es contabilidad) y como que, como por muchos es sabido, yo estaba interesado y, de hecho ya estaba haciendo cosas por mi cuenta, me invitaron a colaborar en el proyecto.

La idea era empezar con un TPV porque hay una persona a quien le interesa y está dispuesta a currar por tenerlo listo en poco tiempo. Va bien marcarse objetivos y este es uno fácilmente asumible.

Pero todos coincidimos en que hacer un TPV sin antes tener definida la Base de Datos de toda la gestión supondría que al ponernos a trabajar en ésta, seguramente nos veríamos obligados a hacer cambios serios en la implementación del TPV, cosa que es duplicar trabajo.

Por esta razón nos basamos en la documentación de la "media gestión" que yo tengo funcionando en la empresa en la que trabajo para hacer el diseño de la base de datos de la futura BulmaFACT. Y el primer problema con que nos encontramos fué que aquel diagrama no se correspondía al 100% con la base de datos actual porque ésta ha sufrido modificaciones que no siempre he dispuesto de tiempos de actualizar al modelo Entidad-Relación y hacerlo cuadrar con el relacional y el SQL de la Base de datos.

Por lo tanto, tuve que partir del modelo ER que tenía y *repescar* las mejoras de notas sueltas y de consultas directas sobre la BD en producción. Y, lógicamente, es previsible que el resultado sea objeto de modificaciones futuras puesto que se pretende que todo aquel que esté interesado pueda participar en el desarrollo del proyecto.

Además ésto añade una complicación extra: Si muchas personas tienen que participar en el diseño del modelo de datos, es difícil que todos entiendan a la perfección la finalidad de entidades o campos concretos colocados por otras personas. Si esto ya es un problema cuando el diseño de la base de datos la hace una sola persona y hemos de ir tomando nota de nuestros propios pensamientos para luego ser capaces de entenderlos en el futuro, imagináros cuando tienen que ser entendidos por muchas personas.

Resumiendo, necesitábamos un modelo ER:

1. Fácilmente modificable.
2. Rápido de consultar.
3. Fácilmente comprensible:
 - ◆ Tanto por el autor...
 - ◆ ...como por terceras personas que lo tengan que consultar.
4. Y si no es mucho pedir que modificando una sola fase del diseño, el resto se puedan generar automáticamente.

Creo que los tres primeros puntos los hemos conseguido. Y el cuarto está en el TO-DO (el proyecto es joven todavía ;-)) pero creo que no será demasiado difícil de conseguir.



Para entender la idea, si pensamos en la naturaleza de un diagrama *ER, veremos que consta de dos partes:

- Una lista de entidades con la correspondiente enumeración de los campos que la componen y con indicación de cual es el campo o campos que forman la clave primaria.
- Un esquema gráfico dónde se representan estas entidades y las relaciones que hay entre ellas (que, como sabemos, a veces podan contener algún campo de datos).

La lista de entidades es fácil y se puede hacer con cualquiera editor de texto plano (vi comes for help! :-)). Pero cuanto más grande es el cociente entre el número de relaciones y el de entidades, más difícil es conseguir una representación gráfica clara y legible (incluso con lápiz y papel) de las relaciones existentes entre las entidades. Por eso es tan difícil encontrar una herramienta CAD adecuada.

Como que esto suponía un gran contratiempo decidí, en lugar de dibujar el diagrama, redactar también una lista de las relaciones indicando las entidades que relacionan, con qué cardinalidad y, de haberlos, los campos de datos pertinentes. Y que cada cual se dibuje, si quiere, su propio diagrama de todo o de la parte que el interese, pero que el formato "oficial" del modelo ER seria el texto plano.

Con esto se solucionaba el punto 1 puesto que, por ejemplo, para añadir una relación al diagrama nada más rápido que escribirla. Y así surgió la idea que solucionaría el punto 2, porque Crear una aplicación capaz de leer e indexar toda esta información era relativamente trivial y, si bien plasmarla toda de golpe y que quedase legible es, como hemos dicho, poco menos que imposible, representar una entidad y, al lado, todas las entidades con que se relaciona y dibujar las relaciones, también era un problema relativamente sencillo.

El problema hubiera sido hacerlo a mano (repetición de trabajo) o consultarlo (ir hacia delante y hacia atrás por un montón de páginas para seguir el trazado del esquema). Pero la idea, era hacer esta misma representación en formato web de manera que, hiperenlazando las entidades, esta navegación se convertiría en instantánea.

Ahora sólo nos quedaba el punto 3: Hacer que el diagrama fuera fácilmente comprensible. En primer lugar por el propio autor, tarea que, como todos sabemos, corresponde a un documento aparte del modelo ER denominado Diccionario de Datos y que se debe hacer al mismo tiempo que se diseña el modelo ER porque si no después no nos acordamos de porqué eran la mitad de cosas. Pero que a la práctica todo el mundo lo hacía después por lo engorroso que resulta pasar constantemente del ratón al teclado y viceversa (ahora podemos, por ejemplo, tener dos pestañas del konsole con vim para editar simultáneamente MER y DD sin tener que utilizar el ratón para nada. Además, tampoco hace falta consultarlo "aparte", sino que las descripciones de los campos nos salen al lado de éstos para la entidad que estamos consultando y en un cuadro emergente al pasar el ratón por encima (anchor con 'title') para los campos de las entidades hijas y las relaciones.

...y en segundo lugar, para terceras personas, caso en el cual sólo queda desear que el autor haya tomado buenas notas de lo que ha hecho y de porqué lo ha hecho si no queremos tener que preguntarle (o si él no quiere que le pregunten :-P).

Pero aun así, para quien no las ha escrito, puede ser complicado encontrar una nota en el momento en que la necesita. Por esta razón, la aplicación también interpreta ciertos marcadores que podamos utilizar dentro el fichero de notas para:

- Delimitar secciones (con jerarquía).
- Marcar las notas como "TO-DO notes" (cosas a hacer) que salen en rojo, o "Recordatorios o cosas hechas".
- "Enganchar" la nota a una o varias entidades o relaciones determinadas.

Finalmente, si existe un fichero denominado CHANGELOG.txt, en la vista general del diagrama sale reproducido debajo de manera que si tenemos diferentes versiones podamos ir anotando los cambios que vamos realizando en comparación con la versión anterior.

Instalación:

La instalación es mucho sencilla. *Vos podáis *devallar la última versión a:

<http://bulma.net/~joanmi/projectes/erm/>⁽⁸⁾

Para instalarlo sólo tenéis que descomprimir el fichero en un directorio accesible vía web:



```
#tar xvzf nombrefichero.tgz
```

Esto os creará un directorio denominado 'erm' y dentro este directorio, además de la aplicación en si, encontraréis un directorio denominado 'PROJECTS'. Cada directorio que creéis dentro de 'PROJECTS' será un nuevo proyecto y cada subdirectorio una nueva versión.

Aquí tenéis que crear tres ficheros (o cuatro) de texto:

- **ERM.txt:** Que contendrá el modelo Entidad-Relación.
- **DD.txt:** Por el diccionario de datos.
- **NOTES.txt:** Por las anotaciones.
- y, opcionalmente, **CHANGELOG.txt** por el historial de cambios.

La sintaxis a utilizar en cada uno de los ficheros (excepto el CHANGELOG.txt que es libre) se explica en el siguiente apartado, aunque también podeis coger como referencia los ficheros de [BulmaFACT](#)⁽⁹⁾ que salen enlazados al pie de cada página. Os remito concretamente a los de BulmaFACT y no de BulmaGES en general porque estos son hechos míos y hacen uso, al menos en un lugar u otro, de todas las funcionalidades del sistema. En cambio el resto los ha hecho Tomeu Borrás cogiendo estos como modelo (prueba de que no es difícil* de entender sin documentación :-))

(*) Ostras! Dicho así parece una indirecta :-P. Pero ya me entendéis ;-)

Como generar los diagramas:

Los diagramas, como hemos comentado, se generan a partir de tres ficheros. El más importante es ERM.txt que contiene el modelo E-R. DD.txt contiene el diccionario de datos y, como su nombre indica, NOTES.txt las anotaciones, comentarios, recordatorios, etc...

A los tres ficheros, las líneas que empiecen por punto y coma (;), *son entendidas como comentarios. El resto se explica acto seguido.

ERM.txt

En este fichero cualquier línea que no esté en blanco o empiece por (;) se entenderá como una entidad o una relación. El texto inicial hasta el primer paréntesis (()) (entidades) o llave ({} (relaciones) es el nombre de la entidad o relación. Si es entidad, entre los paréntesis y separados por comas irán los nombres de los campos de la entidad y los que estén encerrados entre corchetes ([y]), se entenderan como parte de la clave primaria.

Ejemplo:

```
FACTURA ([Numero], Fecha)
LINEAFACTURA ([Numero], Cantidad, P.V.P., Descuento)
```

Las relaciones también pueden tener parámetros y, si los tienen, se indicaran entre paréntesis del mismo modo que a las entidades, pero antes y entre llaves ({ y }) se indicarán, separadas por coma, las dos entidades que relacionan y la cardinalidad correspondiente a cada una entre paréntesis.

Ejemplo:

```
Detalle {FACTURA(1), LINEAFACTURA(1,N)}
; Si tuviese campos de datos, los indicariamos al final de la forma (Campo1, Campo2...)
```

No hay ningún problema con definir entidades y relaciones mezcladas (salvo el buen gusto :-P), pero para definir una relación se necesario que antes se definan las entidades que relaciona. Si no la aplicación se quejará. Igual que pasará si nos equivocamos con el nombre de la entidad. Alerta que los nombres de las entidades y relaciones son sensibles a mayúsculas/minúsculas.

DD.txt



En este fichero, encabezamos las entidades con una línea consistente en su nombre entre corchetes (para las entidades) y llaves (para las relaciones). Después se pone la descripción de la entidad o relación y las descripciones de los campos en las líneas que empiecen con '- '.

Ejemplo:

```
[Factura]
Factura a cliente.
- Numero: Numero de factura.
```

Como veis, no es obligatorio describir todos los campos. Si nos resulta mas claro, aunque no los describimos, también podemos mencionarlos, pero sin los dos puntos (:) que marcan el inicio de la descripción. La descripción de la entidad (o relación) en sí también se puede omitir.

También podemos definir constantes por ahorrarnos escribir descripciones repetitivas. Se podan definir en cualquiera parte del fichero pero sólo serán válidas a partir del momento en que se definen. Las constantes se reconocen por ir encerradas entre los símbolos (< y >).

Ejemplo:

```
<NLI> = Numero de línea.
[LINEAFACTURA]
- Numero: <NLI>
- Cantidad.
- P.V.P.: Precio sin IVA.
- Descuento.
```

NOTAS.txt

Finalmente, en el fichero de notas, las líneas que empiezan en (*) Indican el inicio de un nuevo punto a recordar. Pero si, en cambio, empiezan con (-) quiere decir que son notas de cosas que están por hacer. Por ejemplo, verificaciones (o 'constraints') que han de ser implementadas pero que, obviamente, no lo podemos hacer todavía y queremos acordarnos cuándo sea el momento.

Las notas las podemos enganchar a una (o también varias) entidades o relaciones determinadas, de manera que aparezcan en el 'post-it' que hay habilitado al efecto. Una línea con una lista de entidades separadas por comas entre corchetes indica que la última nota se ha de enganchar a las entidades mencionadas. Y lo mismo vale (en líneas separadas) para relaciones si en vez de corchetes se utilizan llaves.

Ejemplo:

```
* Esta nota saldrá enganchada al visualizar las entidades FACTURA, LINEAFACTURA o la relacion Detalle.
[FACTURA, LINEAFACTURA]
{Detalle}

- Esta otra nos recuerda una cosa pendiente de hacer y saldrá sólo en FACTURA.
[FACTURA]

//ol
```

Además, en cualquiera momento, podemos definir secciones poniendo el nombre de las mismas entre los símbolos (< y >) en una línea independiente. Las secciones son anidables y al encontrar una línea con (<>) (Sin nombre) se entenderá que acaba la última sección definida.