Auxiliar 2 de Bases de Datos

Prof. Claudio Gutiérrez, Aux. Mauricio Monsalve, Ayu. Diego Díaz

Temas. *Introducción al modelamiento conceptual usando diagramas Entidad Relación. Entidades, relaciones, atributos, cardinalidad. Diferentes estilos de diagramas. Del modelo al código.*

Modelamiento conceptual

Típicamente los problemas ingenieriles pasan por una fase de diseño. Esta fase es anterior a la implementación de la solución, pues en el diseño se estudia cómo solucionar el problema (y la calidad de la solución misma).

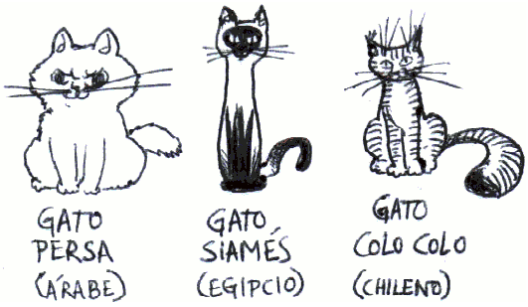
Cuando hay información de por medio, el modelamiento está orientado a la cosa conceptual. Aquí vemos el típico ejemplo de los *mapas conceptuales*, planos que relacionan conceptos diferentes. Pues bien, el modelo de datos usa un tipo de mapa conceptual bien específico: el *modelo entidad relación*.

Modelo Entidad Relación

El modelo Entidad Relación, en adelante *ER*, es un modelo que presenta distintos cuerpos de datos (*entidades*) y cómo se relacionan entre sí (*relaciones*). Se usa ER para mapear un *universo del discurso* real en un nivel abstracto, de información.

Los gatos comen ratones...

Como ejemplo introductorio, modelemos un mundo de gatos y ratones, todos con nombre. Modelaremos la información sin caer en el detalle de cada gato (Mizifú, Fido, Cucho, Pelusa, etc.) y ratón, pero considerando que son detallables.



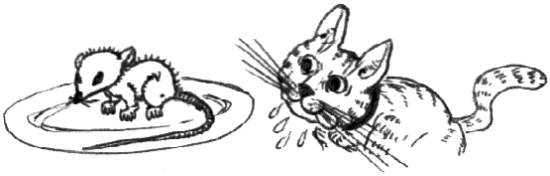
Todos son gatos. Pero son diferentes.

¿Cómo agregamos los *detalles* de cada entidad? Usando *atributos*. Por ejemplo, consideremos que los gatos tienen nombre, raza, peso, edad, etc.

Una entidad y sus atributos.

El diagrama anterior muestra cómo se presentan las relaciones y sus atributos. Las entidades se denotan como rectángulos. Los atributos, en cambio, como elipses o globos de texto. Los atributos se conectan a sus entidades mediante líneas. Es costumbre dejar a los atributos cerca de sus entidades.

Es importante mencionar aquí que el atributo subrayado, *nombre*, es una *llave*. Esto es, identifica plenamente a una *instancia de entidad*. Cuando hablamos de Mizifú, sabemos de quién hablamos: ya sabemos su peso, raza, edad, etc.



Un delicioso ratón... para lamerse los bigotes.

Pensemos en una potencial relación: *los gatos comen ratones*. A nivel de información, un gato se relaciona con un ratón que se comió. Aquí tenemos dos entidades (*gato* y *ratón*) y una relación (*come*). Esto se diagrama como sigue:

Gato (ent) – come (rel) – ratón (ent)

En la notación ER, las relaciones se representan como rombos rotados 45°. Las relaciones se conectan con sus entidades partícipes a través de líneas continuas.

Ahora bien, es posible que las relaciones, en su manera de presentarse, lleven a ciertas ambigüedades. Por ejemplo, del diagrama ER se podría leer *ratón come gato* en vez de *gato come ratón*. Para nosotros es claro que los ratones no comen gatos... ¡a menos que sean enormes y violentos huarenes!

Ratón come gato.

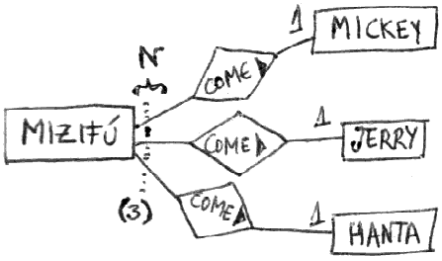
Para evitar posibles ambigüedades, se pueden usar triángulos para indicar a qué dirección apuntan las relaciones, se pueden agregar roles o se pueden incluir los nombres de las entidades en la relación.

Dos maneras de evitar ambigüedades.

La participación de las entidades en las relaciones es, muchas veces, limitada en número. Por ejemplo, un gato puede no comer ratones así como puede haberse comido muchos ratones en el pasado. Por otro lado, un ratón puede ser comido por un gato (a lo sumo) o está vivo. (Objeción: *mamá gata que reparte el ratón*) Esta caracterización de las participaciones es lo que llamamos *cardinalidad*.

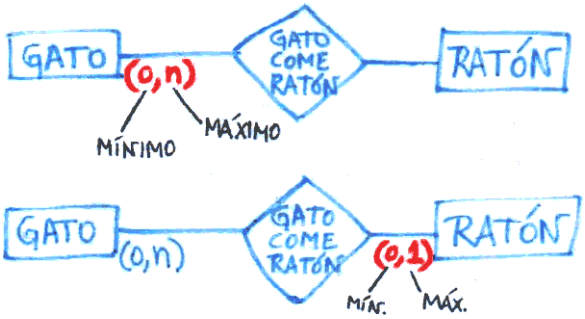
No todos los gatos tienen los mismos gustos...

En nuestra notación, denotamos la cardinalidad como un par (*mín*, *máx*) de cotas. Y anotamos la participación en la relación en el lado de cada entidad. Tratamos de sintetizar algo como:



Relación N:1.

Usando *n* para designar la multiplicidad, las cardinalidades del ejemplo quedan como:



Un gato puede no comer ratones hasta comer muchos ratones: (0,n). Por otra parte, un ratón es comido, a lo sumo, por un gato: (0,1).

Los símbolos que usamos para los límites son: (0,1), (1,1), (0,*n*), (1,*n*) y (*n*,*m*). (**¿existe (0,0)?**) También es posible usar términos como (2,5) cuando la situación lo amerite.

Choque de notaciones

La notación de cardinalidad usada en el curso es detallada. Desafortunadamente, se suele usar una notación práctica sólo para relaciones binarias y que denota las cardinalidades al revés:

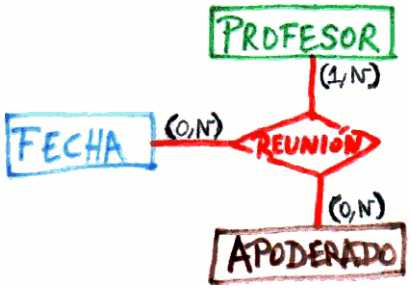
*Un gato come **N** ratones.*
*Un ratón es comido por **1** gato.*

Las desventaja de la notación anterior aparece en las relaciones ternarias o *n*-arias en general, con $n>2$. En cambio, nuestra notación tiene significado completo para cualquier número de entidades participantes.

Pensemos, por ejemplo, en la relación ternaria *reunión*, que representa una reunión de curso. En una reunión de curso, participa el profesor jefe y los apoderados, en una fecha determinada.

- *Mi hijo es el primero del curso.*
- *Y el mío el segundo. (¡Hasta que un accidente deje a mi hijo como el primero! Muaja, ¡muajaja!, ¡¡MUAJAJAJAJA!!)*

Las tres entidades relacionadas por reunión son: *profesor* (jefe), *apoderado* y *fecha* (que bien podría ser un atributo *compuesto*).



Relación ternaria "reunión"

El diagrama anterior representa los hechos que: un profesor jefe siempre asistirá, al menos, a una reunión, pero deberá asistir a varias; hay apoderados que jamás asisten a las reuniones mientras que otros van muchas veces o siempre; y en una fecha determinada puede no haber reuniones de curso como pueden haber varias (típico final de mes).

La relación anterior sería cuaternaria si se considera la entrega del *boletín de notas* de un alumno (cuarta y quinta entidades). Este tipo de situación es completamente inconveniente de modelar con la técnica usual de notación de cardinalidades, a menos que se violen conceptualmente las reglas del modelamiento ER.

Del modelo al código

El uso de diagramas ER es parte de una técnica más grande que consiste en generar código a partir de un diseño. Esto es, en parte, la filosofía de las *metodologías dirigidas por el diseño* (*model driven* etc.) y de los *procesos unificados* (como RUP). La idea general es partir de un diseño bien trabajo y **convertir automáticamente el diseño en código**. Esto reduce el riesgo de las etapas intermendias, y deja todo el peso en la etapa del diseño. Es, en parte, un proceso en cascada. Pero permite alterar el código cambiando sólo el diseño, lo que no es lento. Esto es un proceso iterativo.

Si se parte con un diagrama ER, el siguiente paso es convertirlo a relacional. Ahí se puede optimizar un poco utilizando procedimientos de *normalización*. Luego, el modelo relacional [normalizado] se convierte a tablas usando SQL. Hay multitud de programas que hacen todo este trabajo, salvo por lo de normalizar. ¡Pero pasar de un dibujo al código es algo bastante bienvenido! ¡Y reduce el riesgo del desarrollo de software!

En otros casos, como en RUP, se utiliza UML. A partir de los casos de uso, se traspasa *semi automáticamente* cada caso a un modelo de análisis usando algún patrón (como View/Control/Model). Junto al modelo del universo del discurso (de clases), se genera un diagrama de clases orientado a la aplicación, y se genera código (esto es similar a ER). Asimismo, es posible detallar el código a través de diagramas de interacción (secuencia, colaboración) y máquinas de estado, lo que permite generar código mucho más terminado. Técnicas como esta son bienvenidas en ingeniería de software puesto que permiten saltar del diseño a un prototipo de software casi funcional en poco tiempo ¡y con poco riesgo! Claro, las habilidades deben centrarse en el modelamiento.