1 O modelo relacional

1.1 Sumario

2 Introdución

O modelo relacional foi desenvolvido polo investigador E.F Codd en 1970 como alternativa aos modelos existentes até ese momento. Estes modelos tiñan bastantes problemas (inseguridade, falta de concorrencia, etc.), os cales xa se comentaron en seccións anteriores.

O modelo de Codd baséase en dúas áreas das matemáticas: a teoría de conxuntos e a lóxica de predicados, polo que é un modelo moi sólido. Perseguía, entre outros, os seguintes obxectivos:

- **Independencia física dos datos**. O xeito de almacenar os datos non debe influír na súa manipulación lóxica.
- **Independencia lóxica dos datos**. Os cambios que se realicen na BBDD non deben repercutir nos programas e usuarios que acceden a ela (recorda os tres niveis da arquitectura ANSI/SPARC)
- **Flexibilidade** para presentar aos usuarios os datos da forma máis axeitada á aplicación que utilicen.
- **Sinxeleza** para a comprensión e manipulación da base de datos por parte do usuario final.

Hoxe en día o modelo relacional é o máis coñecido e soportado (aínda que non completamente) polos SXBD (MySQL, Postgress, ORACLE, INFORMIX,...).

3 Estrutura do modelo relacional

3.1 Relacións

O elemento básico do modelo relacional é a **relación** que se representa mediante unha **táboa** (en realidade un conxunto de relacións representadas mediante táboas).

Cada relación ten un **nome** e unha lista de **atributos** (as columnas da táboa) que describen un conxunto de entidades do mundo real. Cada fila (**tupla**) dunha táboa representa unha entidade do mundo real. Ademais:

- Unha relación non admite filas duplicadas.
- As filas e columnas non están ordenadas.

- A táboa é plana, é dicir, o cruce dunha fila e unha columna só pode ter un valor, por tanto, non se admiten atributos multivaluados. Por iso dise que cada valor dun atributo debe ser **atómico** (un único valor) ou nulo (que se descoñece o seu valor).

3.2 Dominios

Igual que no modelo E-R, os atributos poden tomar un conxunto de valores que se coñece como **dominio**. O dominio pode ser **xeral** (por exemplo, o código postal está formado por números naturais de cinco cifras) ou **restrinxido** (por exemplo, o atributo sexo pode tomar os valores H ou M).

3.3 Intensión e extensión

Unha relación ten un esquema ou **intensión** que é o conxunto de atributos que a forman, independentemente da orde na que estean. Represéntase da seguinte forma:

```
R(A1, ... , An)
```

Salvo que se modifique a definición do esquema este é invariable ao longo do tempo, polo que é **estático**.

Por outra banda, a **extensión** dunha relación é o conxunto de tuplas, tal que cada tupla é unha lista de valores, é dicir, son os datos que hai na táboa nun momento dado (o contido da relación).

É dinámica, pois hai insercións, borrados e actualizacións dos datos da táboa ao longo do tempo.

3.4 Grao e cardinalidade

O **grao** dunha relación é o número de atributos que ten o seu esquema.

A **cardinalidade** e o número de filas que ten a relación nun momento dado.

4 Restricións do modelo relacional

Definen as regras que debe cumprir o modelo.

4.1 Atributos

O valor de cada atributo debe ser atómico e pertencente o dominio dese atributo, ie: dom(Idade): números naturais < 150. Ademais, os valores dun atributo deben verificar (**`CHECK`**) algunha condición, ie: idade para traballar 16 ? Idade ? 65

4.2 Chaves

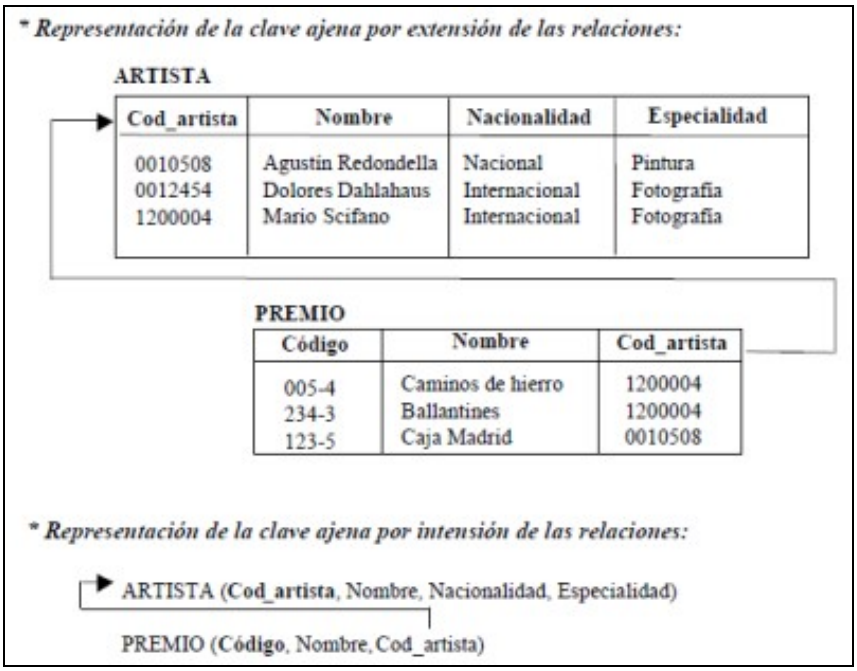
Non pode haber tuplas repetidas. Deben diferenciarse polo menos no valor dun atributo ou conxunto de atributos.

- As **chaves candidatas** son aquelas que teñen un número mínimo de atributos para diferenciar cada tupla.
- A **chave primaria** (**`PRIMARY KEY`**) ou principal é a chave seleccionada entre todas as candidatas, para identificar a cada unha das entidades. Todo atributo que compón a chave primaria non pode conter valores nulos (**`NOT NULL`**).
- Unha **chave alternativa** (**`UNIQUE`**) é cada unha das chaves candidatas que non foron seleccionadas como primaria. Toda táboa debe ter unha chave primaria.

4.3 Integridade referencial

As regras de integridade referencial céntranse en como se relacionan as táboas no modelo relacional, é dicir, ás regras que se deben seguir cando relacionamos táboas.

Para relacionar dúas táboas úsanse **chaves foráneas**, tamén chamadas chave alleas (**`FOREIGN KEY`**). Unha chave allea é un conxunto de atributos dunha táboa que son chave primaria noutra táboa. Os valores que pode tomar unha chave foránea deben existir na chave primaria da táboa correspondente ou ser nulos, é dicir, os valores dunha chave foránea non poden facer referencia a algo que non existe na táboa á que se referencia.



Que sucede na BBDD se se modifica/borra un dos valores da chave principal cod_artista? Distínguense catro casos:

- **Borrado/Modificación en cascada** (`CASCADE`): o borrado dunha tupla, ou a modificación dos valores da chave principal dunha tupla dunha relación, ocasiona o borrado ou a modificación de todas as tuplas relacionadas na outra relación.
- **Borrado/Modificación restrinxida** (`RESTRICTED`): se se quere borrar unha tupla coa súa chave primaria nunha relación e existen tuplas noutras relacións con esa chave como chave allea, non se permite levala a cabo.
- **Borrado/Modificación con posta a nulos** (`SET NULL`): o borrado dunha tupla, ou a modificación dos valores da chave principal, fai que os atributos da chave foránea tomen valores nulos, sempre e cando os teñan permitidos.
- **Borrado/Modificación con posta a valores por defecto** (`SET DEFAULT`): o borrado dunha tupla, ou a modificación dos valores da chave principal fai que os atributos da chave foránea tomen os valores por defecto, sempre e cando existan tuplas na táboa principal con eses valores por defecto na chave primaria.

5 Operacións

Para poder traballar co modelo relacional (crear relacións, introducir datos, etc.) Codd definiu a **álxebra relacional**. Os SXBD utilizan linguaxes baseadas na álxebra relacional.

A linguaxe aceptada por todos os sistemas de bases de datos comerciais é o SQL, linguaxe estruturada de consulta que permite crear, manipular e obter datos de todos os obxectos que constitúen a base de datos. Esta linguaxe foi normalizado pola Organización Internacional de Estándares (ISO) en 1992 e sofre, como todos os estándares, revisións periódicas que fan que continuamente se editen anexos da norma (actualmente chámase **SQL:2008**). No entanto e como case todas as normas, non está soportada 100% por ningún produto comercial, é dicir, cada SGBD incorpora o seu propio SQL que recolle, dependendo de cada sistema, o máis significativo do estándar.

6 Transformación do modelo E-R ao modelo relacional

Unha vez obtido o esquema conceptual mediante o modelo E-R entramos na fase de deseño lóxico de datos. As regras principais para transformar un modelo E-R a un esquema relacional son as que se detallan a continuación.

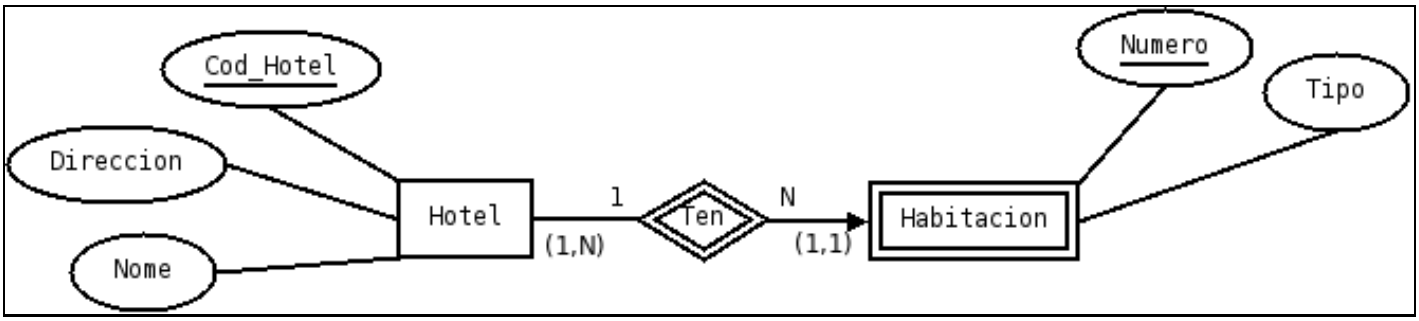
6.1 Entidades e atributos

Toda **entidade** transfórmase nunha táboa, elixindo unha das claves candidatas como clave primaria.

Todo **atributo** transfórmase nunha columna dentro dunha táboa.

6.2 Entidades débiles

As **entidades débiles** transfórmanse directamente en táboas. A súa clave primaria obtense como combinación da clave parcial da entidade débil e da clave da entidade forte.

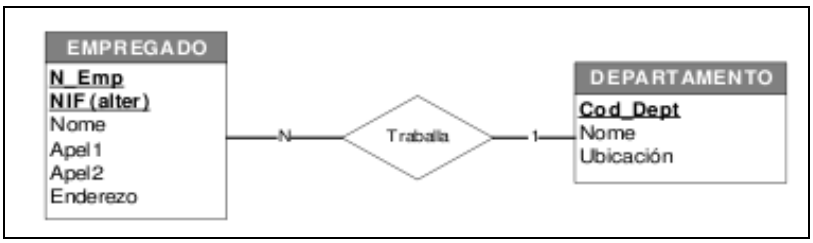


O diagrama anterior transformaríase da seguinte forma:

```
Hotel(Cod_Hotel, Nome, Direccion)
Habitacion(Cod_Hotel, Numero, Tipo)
```

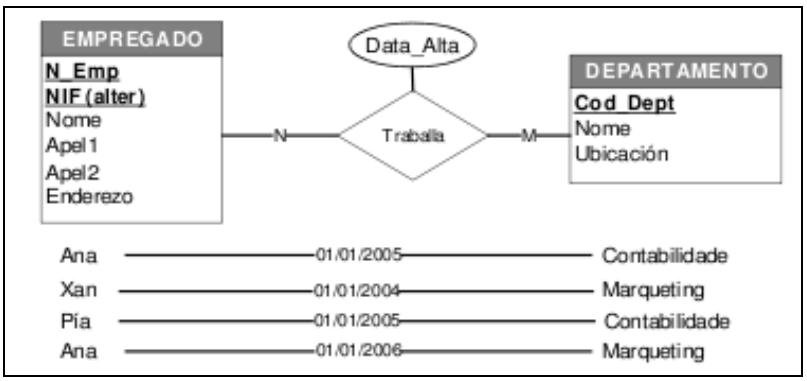
6.3 Transformación de relacións binarias

- **Relacións binarias 1:N**. Transfórmanse introducindo a clave do lado 1 como foránea no lado N. No caso de que a relación teña atributos propios é aconsellable levalos para a táboa asociada coa entidade do lado N. Tamén se pode crear unha táboa, conxugando as claves das entidades que relaciona e incorporando os atributos da relación binaria.



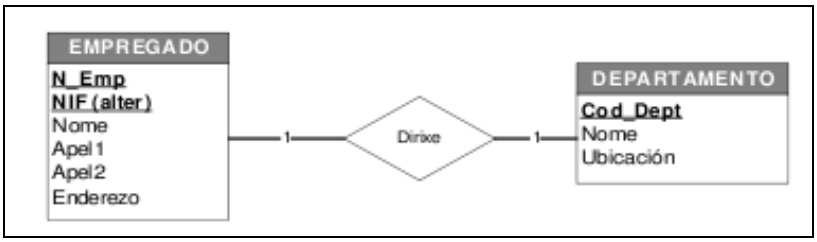

```
EMPREGADO (N_Emp, NIF, Nome, Apel1, Apel2, Enderezo, Cod_Dept)
DEPARTAMENTO (Cod_Dept, Nome, Ubicación)
```

- **Relacións binarias N:M**. Xeran sempre unha táboa adicional na que a clave é a combinación das claves das entidades que relaciona. Se existen atributos na relación tamén se incorporan á nova táboa.




```
EMPREGADO (N_Emp, NIF, Nome, Apel1, Apel2, Enderezo)
DEPARTAMENTO (Cod_Dept, Nome, Ubicación)
TRABALLA (N_Emp, Cod_Dept, Data_Alta)
```

- **Relacións binarias 1:1**. Resólvense introducindo unha clave foránea nunha das entidades relacionadas. Normalmente, propágase a chave primaria do lado parcial (cardinalidade (0,1)) como chave foránea no lado total da relación (cardinalidade (1,1)) xunto cos atributos desta. No caso de que a relación teña atributos propios hai que propagalos para calquera das táboas asociadas coas entidades, a ser posible para aquela que teña participación total da relación, é dicir, a que teña cardinalidade (1,1). Tamén se podería crear unha nova táboa conxugando as chaves das entidades e os atributos da relación. No seguinte exemplo estase supoñendo que un empregado obrigatoriamente ten que traballar nun departamento, pero un departamento pode non ter empregados traballando nel.




```
EMPREGADO (N_Emp, NIF, Nome, Apel1, Apel2, Enderezo, Cod_Dept)
DEPARTAMENTO (Cod_Dept, Nome, Ubicación)
```

6.4 Transformación de relacións reflexivas ou recursivas

6.4.1 Relacións recursivas 1:N

Procédese do mesmo xeito que no caso das relacións tipo binarias 1:N, onde a clave primaria do lado 1 se introducía no lado N da relación como clave foránea, neste caso a entidade do lado 1 é a mesma que a do lado N. Se a relación ten atributos tamén se inclúen na táboa. Por exemplo:

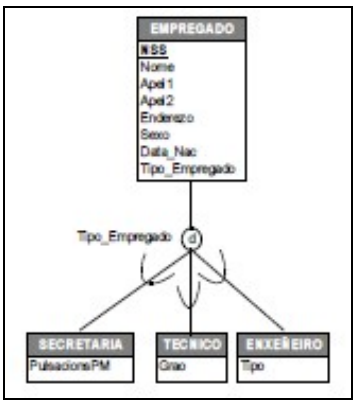
```
EMPREGADO(NSS, Nome, Apel1, Apel2, Enderezo, Sexo, Data_Nac, NSS_Supervisor)
```

6.4.2 Relacións recursivas N:M

Crear unha nova táboa cos atributos da relación e dúas veces os atributos que conforman a chave primaria da entidade que relaciona. Por exemplo, se supoñemos que un empregado pode ter varios xefes distintos na mesma data:

```
EMPREGADO (NSS, Nome, Apel1, Apel2, Enderezo, Sexo, Data_Nac)
TEN (NSS_Subordinado, NSS_Xefe, data)
```

6.5 Transformación de xerarquías



Partiremos do seguinte exemplo de xerarquía total e sen solapar. Para pasar estas relacións ao modelo relacional temos varias opcións que se verán a continuación.

6.5.1 Eliminar os subtipos

- **Transformación**.

1. Transfírense os atributos dos subtipos ao supertipo.
2. Transfírense tamén as relacións dos suptipos ao supertipo e mantéñense as relacións nas que intervén o supertipo.
3. Existe un atributo t (se non existe xa na superclase un atributo discriminador) que indica a que subclase pertence cada tupla da táboa.
4. A clave primaria da única táboa será a da superclase.

```
EMPREGADO (NSS, Nome, Apel1, Apel2, Enderezo, Sexo, Data_Nac, Tipo_Empregado, PulsacionsPM, Grao, Tipo)
```

- **Inconvenientes**.

- Prodúcense valores nulos nos atributos que proveñen dos suptipos, ademais de acceder a datos non requiridos se se desexa coñecer información propia dun subtipo concreto.

- Se a xerarquía non é total, é dicir, hai empregados que poden non ser secretarios, técnicos ou enxeñeiros, habería que poñer valores nulos nos atributos Tipo_empregado, PulsacionsPM, Grao e Tipo.

- **Cando se utiliza**.

- Non se pode usar esta opción en xerarquías solapadas pois cun só atributo discriminador non se poderían indicar as múltiples subclases ás que pode pertencer unha entidade.

- É unha boa opción se os subtipos teñen poucos atributos.

6.5.2 Eliminar o supertipo

- **Transformación**.

1. Os atributos do supertipo pasan a cada un dos suptipos.
2. As relacións nas que apareza o supertipo aplícanse a cada un dos suptipos.
3. A clave primaria de cada táboa será a da superclase.

```
SECRETARIA (NSS, Nome, Apel1, Apel2, Enderezo, Sexo, Data_Nac, Tipo_Empregado, PulsacionsPM)
TECNICO (NSS, Nome, Apel1, Apel2, Enderezo, Sexo, Data_Nac, Tipo_Empregado, Grao)
ENXEÑEIRO (NSS, Nome, Apel1, Apel2, Enderezo, Sexo, Data_Nac, Tipo_Empregado, Tipo)
```

- **Inconvenientes**.

- Prodúcese redundancia.

- Aumenta o número de relacións.

- É máis lento xa que se require o acceso a varias entidades para recuperar a información común.

- **Cando se utiliza**.

- Só funciona con xerarquías totais (perderíanse aquelas entidades da superclase que non pertencesen a ningunha subclase) e sen solapamento.

- Aplícase cando o supertipo ten poucos atributos e participa en poucas relacións.

6.5.3 Eliminación da xerarquía

- **Transformación**.

1. A relación xerárquica transfórmase en tantas relacións binarias como subtipos haxa.
2. Os subtipos transfórmanse en entidades débiles do supertipo.
3. Mantéñense as relacións e atributos do supertipo e dos subtipos.

```
EMPREGADO (NSS, Nome, Apel1, Apel2, Enderezo, Sexo, Data_Nac, Tipo_Empregado)
SECRETARIA (NSS, PulsacionsPM)
TECNICO (NSS, Grao)
ENXEÑEIRO (NSS, Tipo)
```

- **Inconvenientes**. Produce un esquema máis complexo.
- **Cando se utiliza**. É a máis utilizada.

7 Normalización do modelo relacional

A normalización de bases de datos consiste en aplicar unha serie de regras ás relacións obtidas tras o paso do modelo entidade-relación ao modelo relacional. Cando consturímos un modelo relacional a partir dun modelo entidade-relación o habitual é que o modelo xa estea normalizado ata a 3ª forma normal (3FN). Por iso, úsase moito cando se deseña directamente o modelo relacional.

As bases de datos relacionales normalízanse para:

- Evitar a redundancia dos datos.
- Evitar problemas de actualización dos datos nas táboas.
- Protexer a integridade dos datos.

Vexamos un exemplo:

N_MATRICULA	Data_Merca	Prezo	MODELO	MARCA	COR	POTENCIA
C-1111-C	01/01/2002	10.000	Golf TDI - 91	VolksWagen	Azul	90
C-2222-C	01/01/2004	11.000	Golf TDI - 91	VolksWagen	Amarelo	90
C-3333-C	01/01/2003	10.000	320d - 98	BMW	Azul	100
C-4444-C	01/01/2003	5.000	2CV	Citroën	Branco	25

Na táboa anterior coñecendo a matrícula dun coche, pódense coñecer os demais datos asociados a ese coche: que modelo é, a que marca pertence, que cor ten e que potencia desenvolve.

Por outra banda pódese observar que para saber a potencia que ten un coche ou a que marca pertence, chega con coñecer o modelo do coche independentemente de se hai ou non algún exemplar matriculado.

Tomando en consideración o anterior a relación/táboa anterior presenta os problemas que se coñecen co nome xeral de **Anomalías de actualización**.

- **Anomalías de Borrado**. Sucede cando ao borrar unha tupla pérdese toda a información, neste caso, dun coche. Por exemplo se C-4444-C é o único coche que queda na táboa con modelo 2CV. Se se borra esa tupla, a información sobre un coche modelo 2CV (marca: Citroën e potencia: 2) perderase, ata que alguén matricule outro coche modelo 2CV.
- **Anomalías de Inserción**. Sucede cando non se pode rexistrar información nunha táboa ata que se dea unha situación extraordinaria. Por exemplo, se sae ó mercado o coche VOLKSWAGEN POLO-00 cunha potencia de 60CV. Este coche non poderá ser dado de alta na táboa anterior ata que alguén compre un.
- **Anomalías de modificación**. Se se ten un atributo que depende doutro que non é clave, e se rexistramos os dous ao mesmo tempo sucede que, se temos que modificar o valor do segundo atributo temos que facelo en tódalas tuplas. Isto carrexa un problema de potencial inconsistencia da BD, pois podemos esquecernos de modificar unha tupla, e ademais temos que ir tupla por tupla. Por exemplo, POTENCIA depende de MODELO. Cada vez que metamos un GOLF, por exemplo, temos que poñerlle a súa potencia de 90CV. E se hai que modificar esta potencia temos que ir tupla por tupla.

Para resolver estes problemas débese normalizar a táboa cando menos ate 3FN, deste xeito obtéñense dúas novas táboas que serían Taxi e Modelo. **De aí a importancia de facer un bo modelo ER**.

--Arribi 11:20 21 oct 2009 (BST)