Carrera: Técnico Superior en Programación
Materia: Laboratorio de Computación III
Tema: Modelo relacional

Modelo relacional

Relaciones

Como mencionamos en anteriores apuntes, cada entidad está representada lógicamente en la base de datos mediante una tabla. Y cada registro o tupla es una instancia de esa entidad. Esto quiere decir que si en una tabla de Ciudades tenemos cinco registros, tendremos cinco instancias de ciudades distintas, que pueden o no pertenecer al mismo país, que pueden o no ser la capital de algún país, etc. Lo importante dentro de una base de datos relacional es poder asegurar que en una tabla **no se almacenen registros duplicados.**

En la mayoría de los casos, con una sola tabla no se puede representar el problema a resolver. Es por eso que en nuestra en base de datos tendremos una colección de tablas, y cada una de ellas representará una de las entidades de nuestro problema. Si bien distintas tablas representan elementos distintos, es muy común que existan elementos en común entre algunas de ellas. Éstos elementos en común relacionan las tablas entre sí. Es por eso que podemos decir que una relación es el vínculo existente entre dos o más entidades. Generalmente describe algún tipo de interacción entre las mismas. Por ejemplo, una relación entre las entidades Alumno y Universidad podría llamarse "estudia en" ya que un Alumno estudia en una Universidad.

Tablas

La información se almacenan en nuestra base de datos dentro de tablas. Las mismas se caracterizan por representarse gráficamente como objetos rectangulares conformados por filas y columnas. Cada columna almacenará información sobre una propiedad determinada de la tabla (atributo) y cada fila almacenará una instancia de la tabla en cuestión conformada por la información de cada uno de sus atributos.

atributo 1	**atributo 2**	**atributo N**
valor 1 fila 1	valor 2 fila 1	valor N fila 1
valor 1 fila 2	valor 2 fila 2	valor N fila 2
.	.	.
valor 1 fila M	valor 2 fila M	valor N fila M

Dominios

Los dominios determinan cada uno de los posibles valores válidos para un atributo. Cada dominio tiene un nombre y una posible definición del mismo. Ejemplos:

Edad: Número entero entre 1 y 120. Ejemplos: (1, 50, 30, 68)

Nacionalidad: Texto de 50 caracteres. Ejemplos: (Argentina, Chile, España, Italia)

Fecha de nacimiento: Fecha (día, mes y año). Ejemplos: (02/10/1986)

Claves

Existen distintos tipos de claves dentro de un modelo relacional. Las claves nos dan la pauta si dicho valor representa únivocamente al registro dentro de la tabla o si representa un registro de manera únivoca pero en otra tabla.

Las claves más comunes que utilizaremos son:

Clave primaria (PK): La clave primaria es aquella columna (o conjunto de columnas) que representa de manera unívoca a todo el registro. Es un identificador que será siempre único en toda la tabla. No acepta valores nulos.

Clave foránea (FK): La clave foránea es una columna (o conjunto de columnas) de una tabla, la cual está relacionada y es dependiente de una clave primaria (o alternativa) en otra tabla. Puede aceptar valores nulos.

Otros tipos de claves:

Clave candidata: En una tabla, puede ocurrir que existe más de una columna que representa unívocamente a la tabla. De manera que se deberá elegir una para designarla como clave primaria y las demás serán claves candidatas.

Clave alternativa: La clave alternativa es cualquier clave candidata que no haya sido seleccionada como clave primaria.

Clave compuesta: Una clave compuesta es una clave que está conformada por más de una columna.

Nulos

Es común que en ocasiones queramos indicar que el contenido de nuestro atributo equivale a 'ningún valor'. El problema que surge aquí es que la ausencia de valor es difícil de representar entre los distintos tipos de dato. Es necesario poder identificar dicha ausencia de valor en atributos de tipo entero, real, cadena de texto, booleano, fecha, etc. Es por eso que surge el concepto de nulo (en inglés NULL). De ésta manera podemos indicar que el contenido de un atributo no tiene ningún valor.

En claves foráneas, los valores nulos sirven para indicar que el registro actual no está relacionado con ninguno. En otros atributos, indica que dicho valor podría no ser rellenado por alguna razón.

Restricciones

En base de datos se puede definir a una restricción como cierta condición que debe cumplirse para poder incorporar los datos.

Los tipos de restricciones más comunes son:

Clave primaria: Los valores definidos como clave primaria no pueden repetirse.

Único: No permite que los atributos definidos de esta forma puedan repetirse.

Obligatoriedad (NOT NULL): Prohíbe que el atributo definido de esa manera no tenga 'ningún valor'.

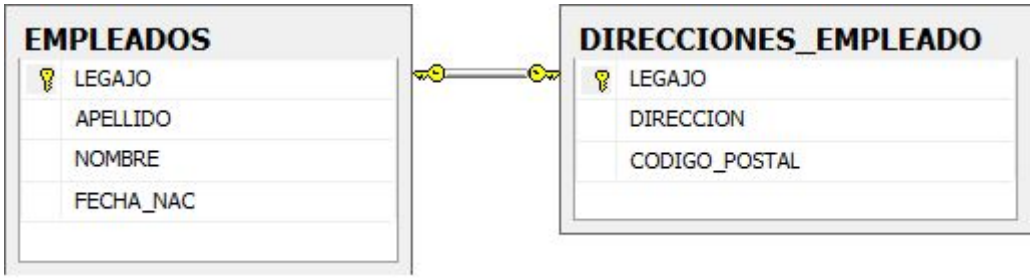
Integridad referencial: No permite ingresar valores en el atributo que no se encuentren reflejados en la tabla donde dicho atributo es clave primaria o alternativa.

Regla de validación (CHECK): Condición que se establece y que se debe de cumplir para que un dato sea ingresado/actualizado.

Tipos de relaciones

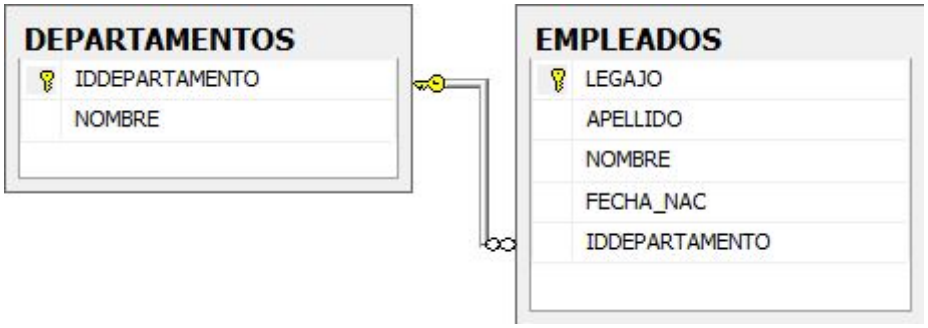
A continuación veremos los tipos de relaciones que pueden existir entre diferentes tablas dentro de una base de datos. Para ello utilizaremos como ejemplo una base de datos de cierta empresa que almacenará la información de sus empleados, departamentos, direcciones, obras sociales e idiomas.

Relación de uno a uno (1 : 1)



Se dice que una tabla A tiene una relación de uno a uno con una tabla B, cuando una fila de la tabla A no puede tener más de una fila coincidente en la tabla B y viceversa. Se crea una relación uno a uno si las dos columnas relacionadas son claves principales o son definidas como atributos únicos (UNIQUE) (en este caso se ha decidido que sólo se almacenará una dirección del empleado).

Relación de uno a varios (1 : n)



Se dice que una tabla A tiene una relación de uno a varios con una tabla B, cuando una fila de la tabla A puede corresponderse con muchas filas de la tabla B. Pero sólo una fila de la tabla B puede corresponderse con otra de la tabla A. Se crea mediante una relación si una de las dos columnas relacionadas es clave principal y la otra se define como clave foránea.

Relación de varios a varios (n : n)

Ena una relación de varios a varios, una fila de la tabla A puede tener muchas filas coincidentes en la tabla B y viceversa. Este tipo de relación se crea definiendo una tercer tabla, denominada tabla de unión. La clave principal de la tabla de unión es la combinación de las claves principales de las tablas A y B, que en esta tabla deberán ser clave foráneas.

Creación de tablas por código

Supongamos que queremos crear la tabla Empleados que figura en el gráfico de la página anterior. Para ello escribiremos el siguiente código:

```
CREATE TABLE EMPLEADOS(
  LEGAJO BIGINT NOT NULL PRIMARY KEY,
  APELLIDO VARCHAR(50) NOT NULL,
  NOMBRE VARCHAR(50) NOT NULL,
  FECHA_NAC DATETIME NULL,
  IDDEPARTAMENTO BIGINT NOT NULL FOREIGN KEY REFERENCES
DEPARTAMENTOS(IDDEPARTAMENTO),
)
```

En el código anterior podemos notar como luego de indicar el comando para crear la tabla Empleados debemos indicar cada una de las columnas con sus características. También habrá que indicar para cada columna un nombre y un tipo de dato. Luego podremos establecer las características que vimos anteriormente:

NULL: Acepta que la columna contenga ningún valor .
NOT NULL: Prohíbe que la columna contenga ningún valor.
UNIQUE: No permite que los valores de esta columna puedan repetirse.
PRIMARY KEY: Los valores definidos como clave primaria no pueden repetirse.
FOREIGN KEY: No permite ingresar valores en el atributo que no se encuentren reflejados en la tabla donde dicho atributo es clave primaria (salvo que acepte nulo).
IDENTITY(inicio, incremento): Las columnas identity o autonuméricas se caracterizan por ser atributos numéricos que automáticamente incrementan su valor sin la necesidad que el usuario ingrese nada. Los parámetros necesarios para definirla son (valor inicial, valor incremental). Siendo el valor inicial el valor numérico entero desde donde comenzará a contar y el valor incremental el valor numérico entero que aumentará de registro en registro.
DEFAULT: La columna tiene un valor por defecto en el caso de que no se le asigne uno de forma explícita.

Cabe destacar que se puede establecer las restricciones de tipo PRIMARY KEY y FOREIGN KEY en

la definición de la columna siempre que no sean claves compuestas. A las restricciones de tipo Foreign Key, además se deberá indicar a qué tabla y campo hace referencia. Es obligatorio que dicha tabla y columna hayan sido previamente creadas en la base de datos. La columna deberá ser una clave primaria en dicha tabla.

Alternativamente se puede crear la tabla Empleados de las siguientes maneras:

```
CREATE TABLE EMPLEADOS(
  LEGAJO BIGINT NOT NULL,
  APELLIDO VARCHAR(50) NOT NULL,
  NOMBRE VARCHAR(50) NOT NULL,
  FECHA_NAC DATETIME NULL,
  IDDEPARTAMENTO BIGINT NOT NULL,
  PRIMARY KEY (LEGAJO),
  FOREIGN KEY (IDDEPARTAMENTO) REFERENCES DEPARTAMENTOS (IDDEPARTAMENTO)
)
```

En este ejemplo vemos como se pueden definir las restricciones del tipo PK y FK luego de definir cada una de las columnas. Será necesario ahora indicar entre paréntesis a la columna de la tabla a la se está definiendo dicha restricción.

```
CREATE TABLE EMPLEADOS(
  LEGAJO BIGINT NOT NULL,
  APELLIDO VARCHAR(50) NOT NULL,
  NOMBRE VARCHAR(50) NOT NULL,
  FECHA_NAC DATETIME NULL,
  IDDEPARTAMENTO BIGINT NOT NULL,
)
GO
ALTER TABLE EMPLEADOS
ADD CONSTRAINT PK_EMPLEADO PRIMARY KEY (LEGAJO)
GO
ALTER TABLE EMPLEADOS
ADD CONSTRAINT FK_EMPLEADO_DEPARTAMENTO FOREIGN KEY (IDDEPARTAMENTO) REFERENCES
EMPLEADOS (IDDEPARTAMENTO)
```

En este ejemplo, se puede notar que debemos ejecutar un conjunto de instrucciones DDL para poder definir todas las características de nuestra tabla. Primero se genera la tabla con la definición de cada uno de las columnas. Luego se altera la estructura de la tabla mediante la consulta del tipo ALTER TABLE para agregar la restricción (ADD CONSTRAINT) que corresponda. Nótese la necesidad de asignarle un nombre representativo a la restricción PK_EMPLEADO y FK_EMPLEADO_DEPARTAMENTO para la clave primaria y foránea respectivamente.

El uso de la instrucción GO permite poder ejecutar el conjunto de instrucciones como un proceso por lotes.

Claves compuestas

Supongamos que queremos representar los datos de empleados de una empresa, pero ésta en particular tiene una peculiar forma de identificar a los empleados. Utiliza legajos, pero al tener diferentes sucursales, incorpora también el código de sucursal para poder identificar al empleado. Ya que los legajos se repiten entre distintas sucursales.

Ejemplo:

LEGAJO	*SUCURSAL*	*APELLIDO y NOMBRE*
1000	San Isidro	Pérez, Julio
2000	Martinez	Cáceres, Julieta
1000	Martinez	Mercado, Yamila

Esto incorpora una complejidad mayor a nuestra estructura de tabla. Ya conocemos que por proceso de normalización y evitar la redundancia de información, la sucursal deberá ser un código que la identifique y que exija integridad referencial con una clave primaria en una tabla que podríamos llamar Sucursales.
Luego, la estructura de nuestra tabla Empleados quedaría de la siguiente manera:

LEGAJO	*IDSUCURSAL*	*APELLIDO y NOMBRE*
1000	1	Pérez, Julio
2000	2	Cáceres, Julieta
1000	2	Mercado, Yamila

Sin embargo, el atributo Legajo que generalmente podría surgir como clave candidata aquí no lo es. Ya que no nos asegura la representación unívoca de un empleado en la entidad. Sin embargo y siguiendo con la lógica de la empresa, la clave candidata será la combinación de Legajo y Sucursal. Es decir, no podrá haber más de un empleado con el mismo legajo en la misma sucursal pero sí podría haber empleados con mismo número de legajo en distintas sucursales.

Veamos como quedaría la definición de la tabla Empleados con la clave concatenada, asumiendo la clave Legajo + IDSucursal

```
CREATE TABLE EMPLEADOS(
  LEGAJO BIGINT NOT NULL,
  IDSUCURSAL BIGINT NOT NULL,
  APELLIDO VARCHAR(50) NOT NULL,
  NOMBRE VARCHAR(50) NOT NULL,
  PRIMARY KEY (LEGAJO, IDSUCURSAL),
  FOREIGN KEY (IDSUCURSAL) REFERENCES SUCURSALES (IDSUCURSAL)
)
```

Como podemos ver, creamos nuestra tabla empleados definiendo las columnas y luego definimos las restricciones del tipo PK y FK. Para la restricción de clave primaria debemos indicar que la misma está conformada por dos columnas (legajo e idsucursal). Luego, aún debemos indicar que la columna IdSucursal además es una clave foránea en la tabla Sucursales.

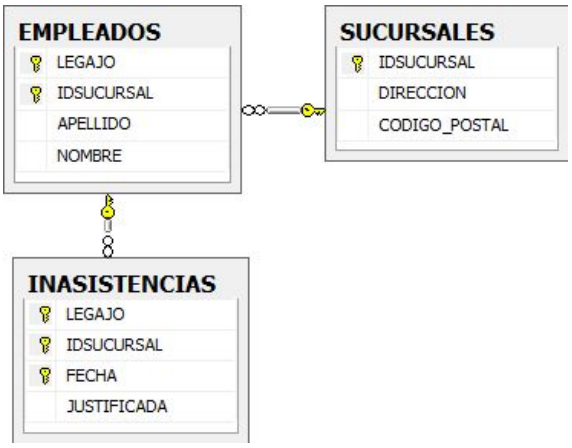
Veamos qué pasa ahora si la empresa desea registrar las inasistencias de los empleados. Para ello debemos registrar una inasistencia para cada empleado en la fecha que faltó,

eventualmente podríamos indicar si la ausencia fue o no justificada.
Diseñamos la tabla Inasistencias y vemos que, nuevamente, tendremos que crear una clave compuesta. La clave principal de la tabla inasistencias debería ser el empleado y la fecha en la que faltó. Esto se debe a que un empleado puede faltar en diferentes fechas y en una fecha pueden faltar distintos empleados pero jamás puede ocurrir que un empleado falte dos veces el mismo día.

Lo complejo de la situación aquí es que cuando nos referimos a "el empleado" en la tabla de Inasistencias, claramente estamos haciendo referencia a una clave foránea. Pero, como vimos anteriormente, la clave principal del empleado en la tabla de Empleados estaba compuesta por el Legajo y el IdSucursal. De modo que aquí debemos realizar una clave foránea compuesta exigiendo integridad referencial sobre la combinación de las dos columnas.

Veamos como queda la tabla Inasistencias con las restricciones que definimos anteriormente:

```
CREATE TABLE INASISTENCIAS(
  LEGAJO BIGINT NOT NULL,
  IDSUCURSAL BIGINT NOT NULL,
  FECHA DATETIME NOT NULL,
  JUSTIFICADA BIT NOT NULL DEFAULT (0),
  PRIMARY KEY (LEGAJO, IDSUCURSAL, FECHA),
  FOREIGN KEY (LEGAJO, IDSUCURSAL) REFERENCES EMPLEADOS (LEGAJO, IDSUCURSAL)
)
```



Podemos observar una clave principal conformada por tres columnas (legajo, idsucursal y fecha) y una clave foránea conformada por dos columnas (legajo e idsucursal). Esta clave foránea exigirá integridad referencial por una ocurrencia de ambos valores. De modo que deberá existir un registro que tenga la combinación de ambos valores en dichas columnas para que se permita el ingreso de datos. Tiene que coincidir el Legajo y el IdSucursal.

La especificación DEFAULT en una columna nos permite indicar cuál será el valor por defecto por si no se ingresa ningún valor en dicha columna. En el caso del campo booleano Justificada, el valor por defecto es False.