Relaciones exclusivas

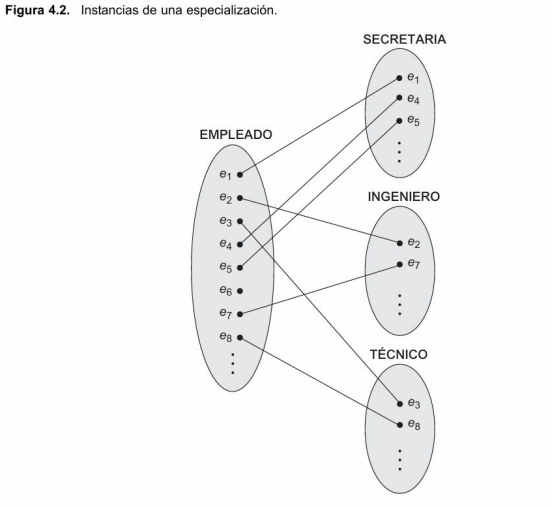
Bases de datos
Tema 4: Diseño lógico de una base de datos
Presentación por: Molina Véjar Aarón Gael

Introducción: Modelo ER (mejorado o extendido)

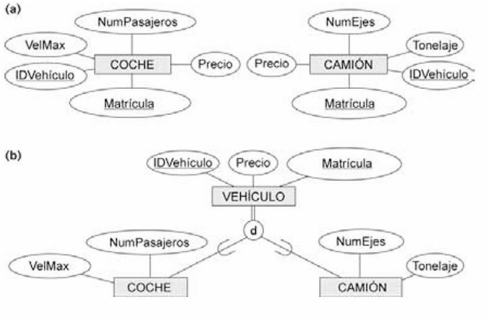
Los conceptos del modelo entidad-relación básico son suficientes para modelar la mayoría de escenarios en los que se desea construir una base de datos. Sin embargo a el deseo de construir aplicaciones que reflejen de un modo más preciso las propiedades y restricciones de los datos han llevado a la construcción de un modelo con conceptos adicionales que se pueden usar para modelar escenarios más específicos.

Algunos conceptos que se añaden en el modelo EER

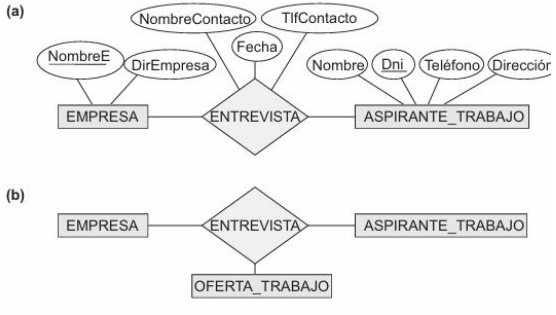
Especialización



Generalización



Agregación



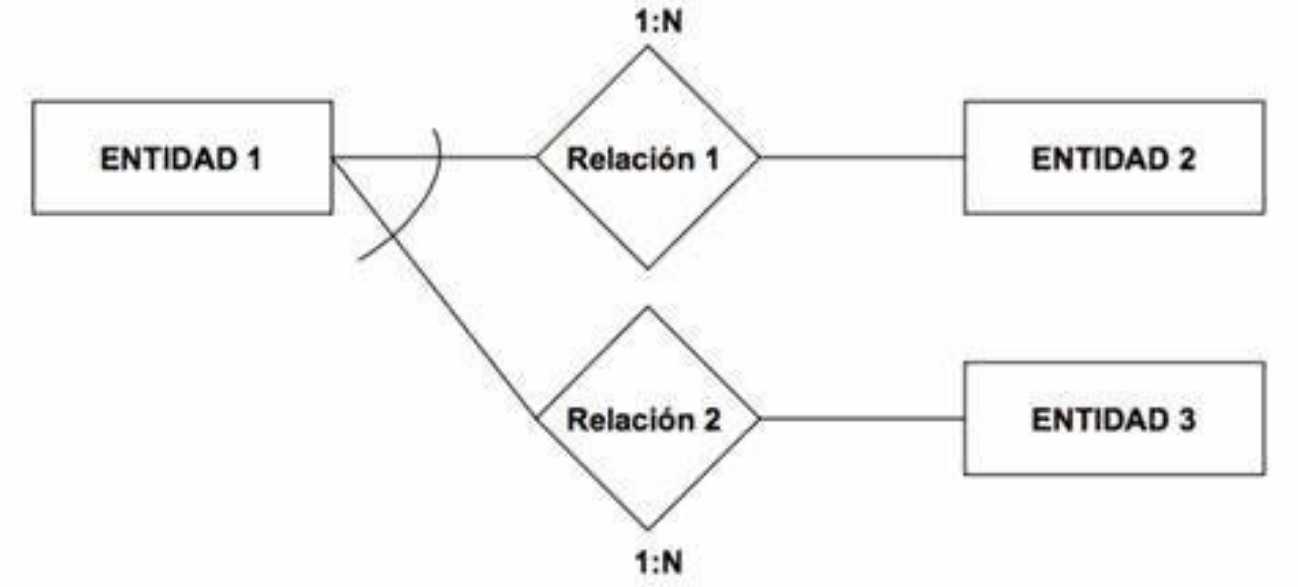
Concepto de exclusividad y relaciones exclusivas

En general el concepto de exclusividad se aplica como una restricción a las relaciones comunes y corrientes que se manejan en el modelo entidad-relación.

La restricción utilizada en las relaciones exclusivas se conoce como “restricción de disyunción” (disjointness) y esta es definida como una de las restricciones que definen el concepto de supertipo y subtipo.

La restricción de disyunción especifica que las subclases de la especificación deben estar separadas, es decir que deben ser como máximo, miembro de una de las subclases de la especialización.

Representación de una relación exclusiva



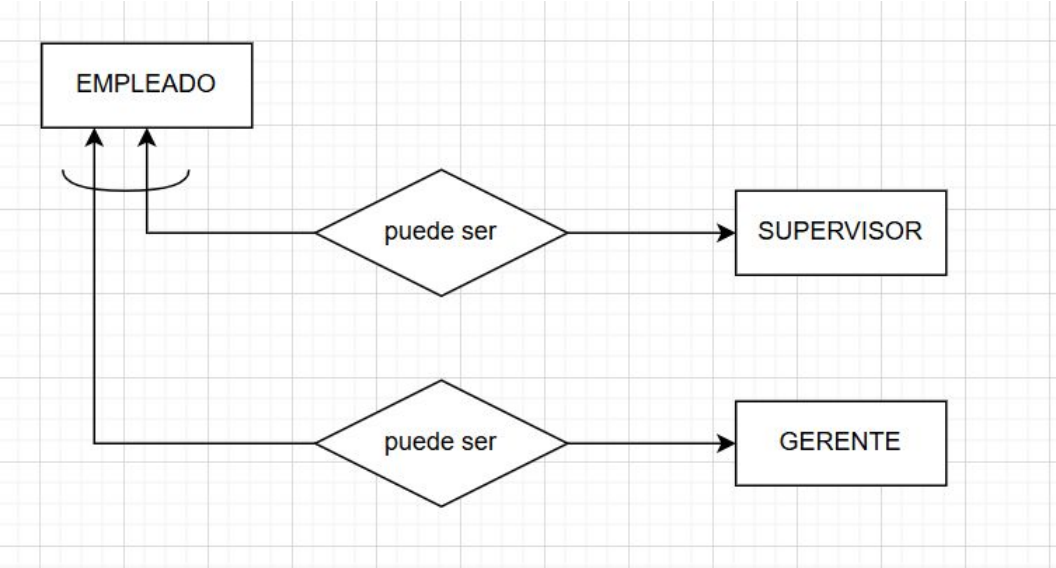
Ejemplo de un escenario con una relación exclusiva

Entidades:

- -Gerente
- -Empleado
- -Supervisor

Exclusividad: Un empleado puede ser un gerente o un supervisor, pero no ambos.

Representación



Implementación en modelo relacional

- Claves foráneas y restricción de unicidad: Se crean tablas independientes por cada entidad y se relaciona como si se tratara de una relación normal a la hora de relacionar claves foráneas entre entidades.
- Restricción check ó bloques de código: Para indicar y asegurar la restricción de disyunción se puede optar por considerar una restricción check para cada entidad, sin embargo lo más recomendable es manejarlo con bloques de programación para tener un control más explícito y seguro.

Modelo relacional con el ejemplo Empleado (restricción por check)

EMPLEADO= {num_empleado int PK, nombre varchar(50), calle varchar(40), puesto varchar(40), …}

SUPERVISOR= {num_supervisor int PK, fecha_ingreso date,..., num_empleado int FK C}

GERENTE= {num_gerente int PK, fecha_ingreso date, …, num_empleado int FK C}

Modelo relacional con el ejemplo Empleado (restricción por bloque de código o trigger)

EMPLEADO= {num_empleado int PK, nombre varchar(50), calle varchar(40), puesto varchar(40), …}

SUPERVISOR= {num_supervisor int PK, fecha_ingreso date,..., num_empleado int FK}

GERENTE= {num_gerente int PK, fecha_ingreso date, …, num_empleado int FK}

Trigger en SQL

```
CREATE TRIGGER exclusividad_gerente_supervisor
BEFORE INSERT OR UPDATE ON Gerente
FOR EACH ROW
BEGIN
   IF EXISTS (SELECT 1 FROM Supervisor WHERE num_supervisor = NEW.num_gerente) THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'El empleado no puede ser Gerente y Supervisor al mismo
tiempo';
   END IF;
END;

CREATE TRIGGER exclusividad_supervisor_gerente
BEFORE INSERT OR UPDATE ON Supervisor
FOR EACH ROW
BEGIN
   IF EXISTS (SELECT 1 FROM Gerente WHERE num_gerente = NEW.num_supervisor) THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'El empleado no puede ser Gerente y Supervisor al mismo
tiempo';
   END IF;
END;
```

Referencias

- Elmasri, R., & Navathe, S. B. (2016). **Fundamentals of Database Systems** (7th ed.). Pearson.
- Coronel, C., Morris, S., & Rob, P. (2016). **Database Systems: Design, Implementation, and Management** (12th ed.). Cengage Learning.