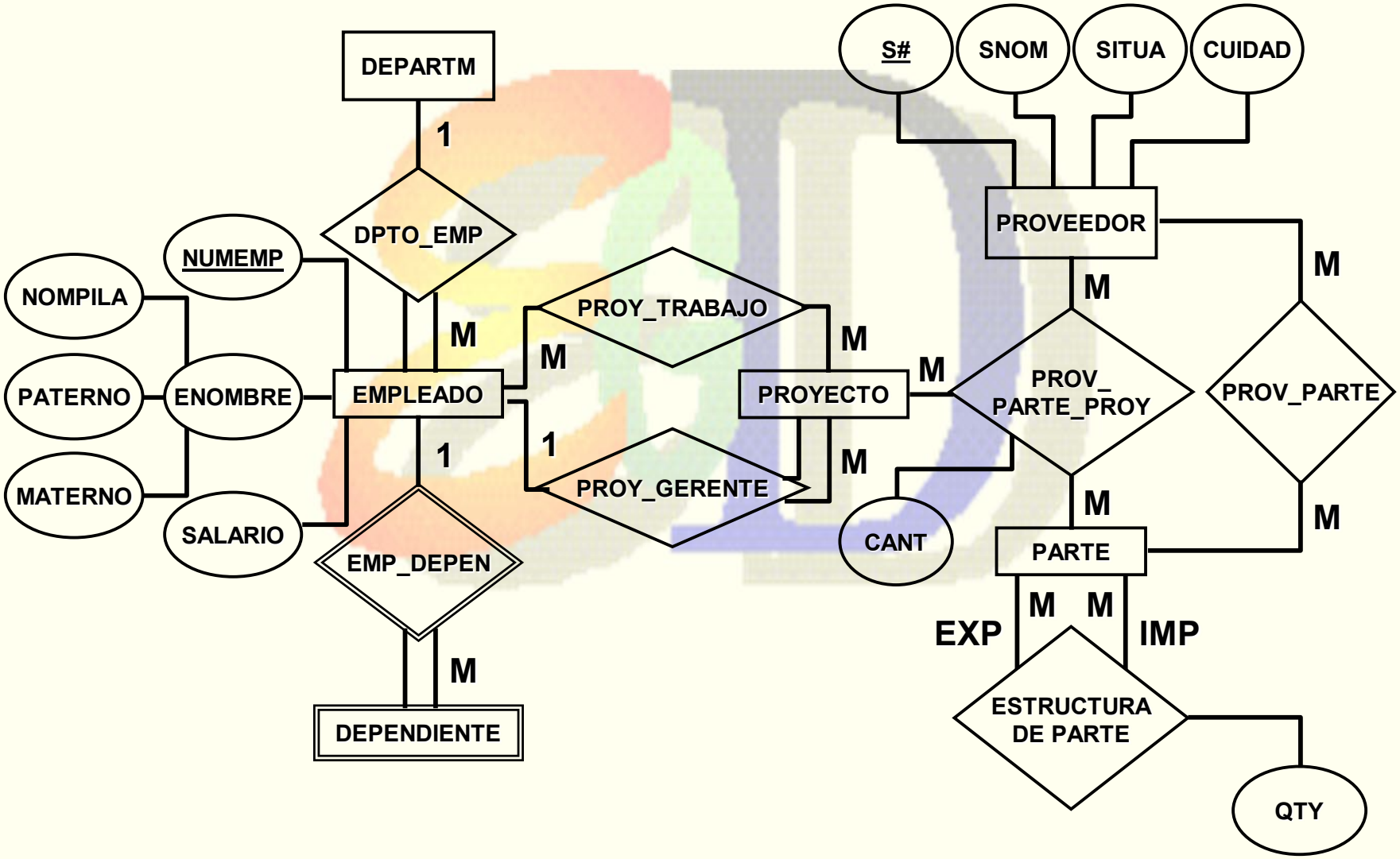
Tema 5.1 .- Tabulación: Traducción E-IR-Relacional.

Diagrama E/IR



Traducción de un Diagrama E/R a una definición de Base de Datos Relacional

ENTIDADES REGULARES

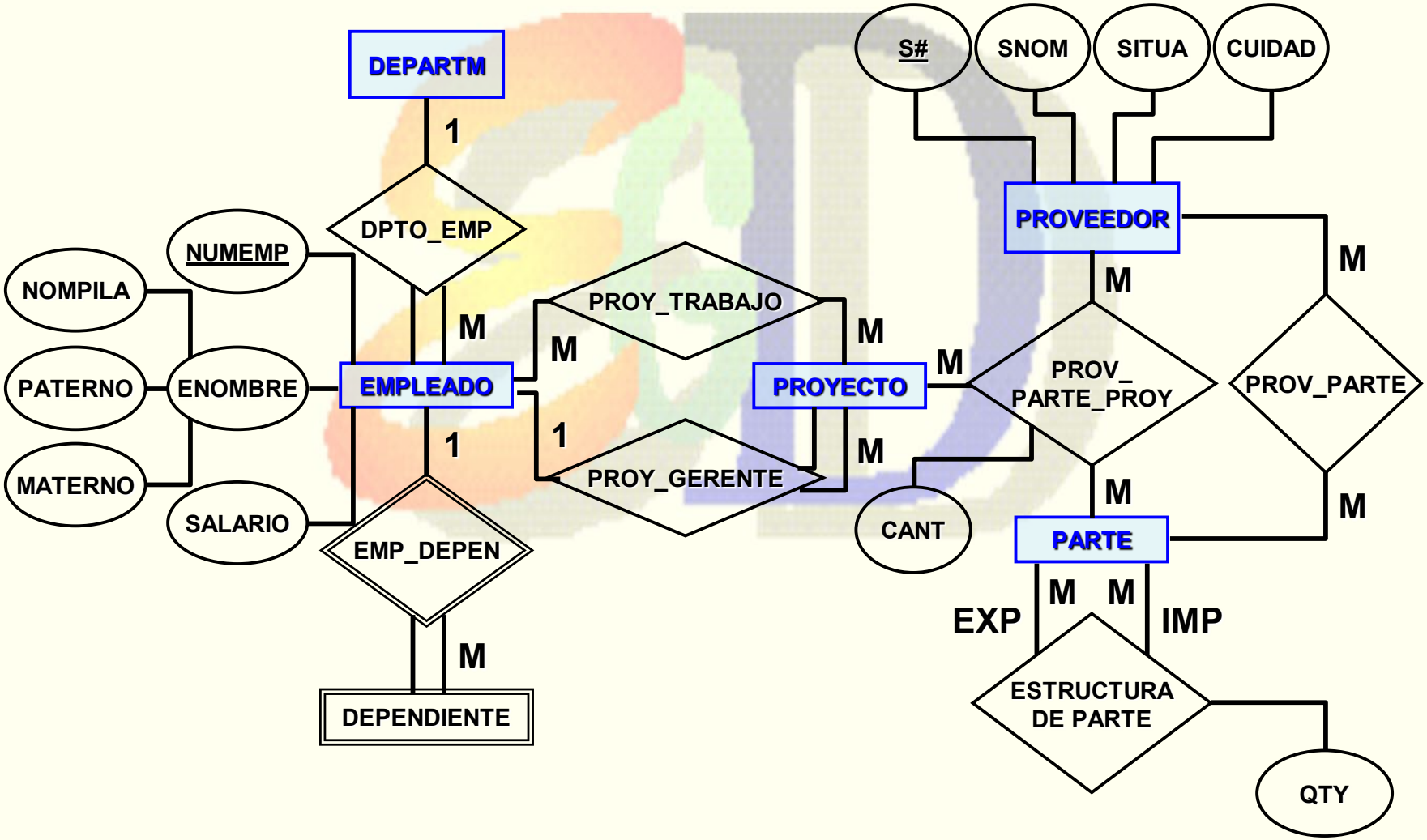
- **Cada tipo de entidad regular corresponde a una relación base**.

  Por lo tanto, la base de datos incluirá cinco relaciones base:

  - **DEPTO.**
  - **EMP.**
  - **S.**
  - **P.**
  - **J.**

  correspondientes a las entidades regulares **departamento, empleado, proveedor, parte** y **proyecto**.

Traducción de un Diagrama E/R a una definición de Base de Datos Relacional



Entidades Regulares

- Por supuesto, cada una de estas relaciones base tendrá una clave primaria:
  - **NUMDEPTO.**
  - **NUMEMP.**
  - **S#.**
  - **P#.**
  - **J#.**

  correspondientes a las propiedades clave identificadas en el diagrama E/R.

Entidades Regulares

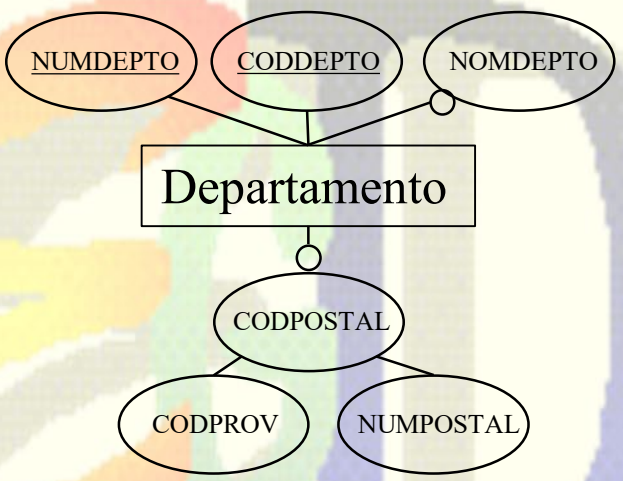
- Estos hechos pueden documentarse escribiendo un conjunto apropiado de sentencias de definición de datos de SQL.
- Esta sugerencia de utilizar SQL (o un pseudoSQL) para registrar las decisiones de diseño **no indica que sea la única forma de realizarlo**, **pero el formalismo alternativo que se utilice deberá ser equivalente en lo funcional**.

  Por ejemplo, esta sería la proposición para la relación **DEPTO** (a grandes rasgos):

```
create table DEPTO
        (NUMDEPTO . . . ,
                    . . . , )
        primary key (NUMDEPTO);
```

Entidades Regulares

- Si la entidad departamento tuviese estas propiedades:




```
create table DEPTO (NUMDEPTO . . . ,
                    CODDEPTO . . . not null,
                    NOMDEPTO . . . not null,
                    CODPROV . . . not null,
                    NUMPOSTAL . . . not null)
primary key (NUMDEPTO)
alternative key (CODDEPTO);
```

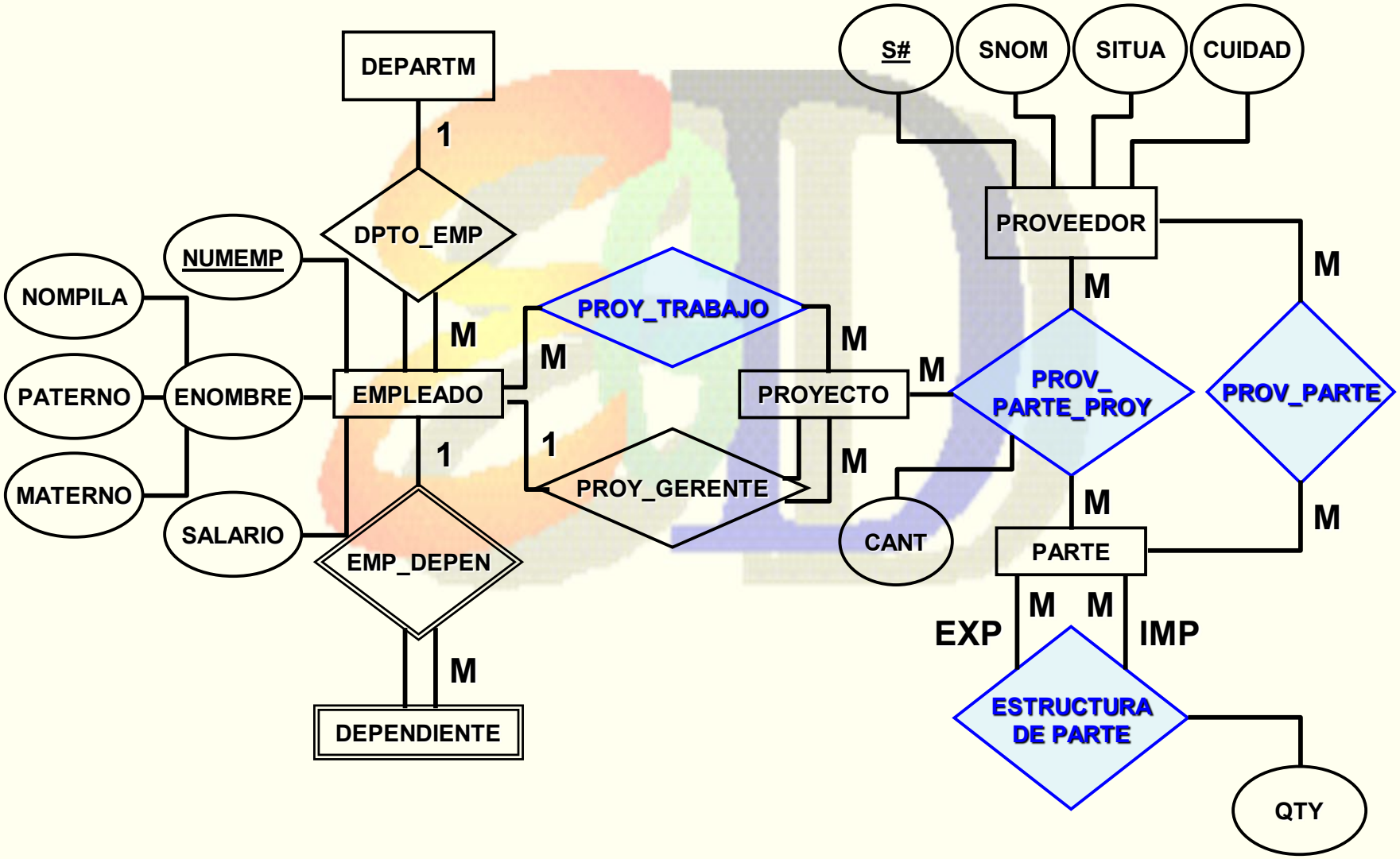
Interrelaciones de Muchos a Muchos

- Las que aparecen en el ejemplo (binarias o no), son las siguientes:
  - **PROY_TRABAJO.**
  - **PROV_PARTE.**
  - **PROV_PARTE_PROY.**
  - **ESTRUCTURA DE PARTE.**
- Cada una de ellas se corresponde con una relación base. Por ello, se introducen otras cuatro relaciones base.

  Por ejemplo, para la interrelación **PROV_PARTE** puede ser la relación base **SP**.

  Dejando a un lado por el momento la cuestión de la clave primaria de esta relación podemos centrarnos en el asunto de las claves ajenas, **ya que son necesarias para identificar a los participantes de la interrelación**.

Interrelaciones de Muchos a Muchos



Interrelaciones de Muchos a Muchos

```
create table SP
        (S#    . . . ,
         P#    . . . ,
               . . . )
        foreign key (S#) references S
        foreign key (P#) references P
```

- Es evidente que la relación **SP** **debe incluir 2 claves ajenas**: **S#** y **P#**, **correspondientes a los 2 participantes: proveedores** y **partes** y por ello **hacen referencia a las relaciones participantes S** y **P**.
- Además, es preciso especificar un **conjunto apropiado de reglas para cada clave ajena**.

Reglas para Claves Ajenas

- Es importante darse cuenta que la regla de integridad referencial se formula en términos de estados de la base de datos.

> **La base de datos no debe contener valores no nulos de claves ajenas para los cuales no exista un valor concordante de la clave primaria en la relación objetivo pertinente.**

- **Cualquier estado de la base de datos que no satisfaga la regla será incorrecto por definición**, pero **¿cómo evitar esos estados incorrectos?** La regla en si no lo dice.
- Una posibilidad es **que el sistema rechace cualquier operación que produzca estados ilegales**.

Reglas para Claves Ajenas

- Una alternativa preferible en algunos casos será que el sistema:
  - **Acepte la operación.**
  - **En caso necesario, realice operaciones compensatorias que garanticen un estado legal como resultado final.**

    Por ejemplo, si el usuario solicita eliminar el proveedor S1 de la relación S, también debería ser posible hacer que el sistema también elimine de la relación SP las referencias a dicho proveedor sin necesidad de acciones adicionales por parte del usuario.

    Todo ello suponiendo que el efecto de eliminación en cascada constituye lo deseado.
- En consecuencia, para cualquier base de datos, el diseñador podrá especificar:
  - **Qué operaciones habrán de rechazarse.**
  - **Cuáles habrán de aceptarse.**
    - **Qué operaciones de compensación habrá de realizar el sistema.**

Reglas para Claves Ajenas

- La línea fundamental es que, **para cada clave ajena, será necesario responder a dos preguntas**:

  1) **¿Qué deberá suceder si hay un intento de eliminar el objetivo de una referencia de clave ajena?**

  2) **¿Qué deberá suceder si hay un intento de modificar la clave primaria del objetivo de una referencia de clave ajena?**

Eliminación de Referenciados

PREGUNTA

¿QUÉ DEBERÁ SUCEDER SI HAY UN INTENTO DE ELIMINAR EL OBJETIVO DE UNA REFERENCIA DE CLAVE AJENA?

- Por ejemplo: **Un intento de eliminar un proveedor del cual existe por lo menos una oferta.**
- Considerando este caso de forma explícita, existen al menos tres posibilidades:
  - **RESTRICCIÓN:** La operación está restringida al caso en que no existen tales ofertas y se rechaza en caso contrario.
  - **PROPAGACIÓN:** La operación se propaga en cascada, eliminando también las ofertas correspondientes.
  - **ANULACIÓN:** Se asignan nulos a la clave ajena en todas las ofertas implicadas y a continuación se elimina el proveedor.

    Por supuesto, este caso no es aplicable si la clave ajena no puede aceptar nulos.

Modificación de la Clave Primaria Referenciada

PREGUNTA

¿QUÉ DEBERÁ SUCEDER SI HAY UN INTENTO DE MODIFICAR LA CLAVE PRIMARIA DEL OBJETIVO DE UNA REFERENCIA DE CLAVE AJENA?

- Por ejemplo: **Un intento de modificar el número de un proveedor del cual existe por lo menos una oferta.**
- Considerando este caso de forma explícita, existen al menos las mismas tres posibilidades que en el caso de la eliminación:
  - **RESTRICCIÓN:** La operación está restringida al caso en que no existen tales ofertas y se rechaza en caso contrario.
  - **PROPAGACIÓN:** La operación se propaga en cascada, modificando también la clave ajena de las ofertas correspondientes.
  - **ANULACIÓN:** Se asignan nulos a la clave ajena en todas las ofertas implicadas y a continuación se modifica el número del proveedor.

    Por supuesto, este caso no es aplicable si la clave ajena no puede aceptar nulos.

Ejemplos de Interrelaciones Muchos a Muchos

- Volviendo al ejemplo de la interrelación muchos a muchos **PROV_PARTE** representada por la tabla SP, ahora se pueden **especificar sus reglas para claves ajenas**.

  Adviértase que las reglas de borrado en este ejemplo y en los subsiguientes de interrelaciones que no son débiles:

  - **Son solo una posibilidad.**
  - **No las fija el hecho de tratarse de una interrelación m:m, sino la semántica real que se está representando.**

```
create table SP
    (S# . . . ,
     P# . . . ,
     . . . )
foreign key (S#) references S
    delete of S restricted
    update of S.S# cascade
foreign key (P#) references P
    delete of P restricted
    update of P.P# cascade;
```

Ejemplos de Interrelaciones Muchos a Muchos

```
create table SP
    (S# . . . ,
     P# . . . ,
        . . . )
    foreign key (S#) references S
        delete of S restricted
        update of S.S# cascade
    foreign key (P#) references P
        delete of P restricted
        update of P.P# cascade;
```

- Las referencias a ambos participantes en la interrelación muchos a muchos presentan las mismas reglas de claves ajenas:
  - Se impide la eliminación de un proveedor o una parte que estén referenciados en la interrelación.

    Mientras existan ofertas correspondientes no deben eliminarse ni el proveedor ni la parte implicada.

Ejemplos de Interrelaciones Muchos a Muchos

```
create table SP
    (S# . . . ,
     P# . . . ,
        . . . )
    foreign key (S#) references S
        delete of S restricted
        update of S.S# cascade
    foreign key (P#) references P
        delete of P restricted
        update of P.P# cascade;
```

- Se permite la **actualización de la clave de las entidades participantes**, de forma que **será factible reflejar**, por ejemplo, **el cambio del código identificativo de un proveedor**, reflejándose este cambio adecuadamente en la interrelación.

Ejemplos de Interrelaciones Muchos a Muchos

- **¿Qué sucede con la clave primaria de la relación que representa a la interrelación muchos a muchos.**
- **La combinación de las claves ajenas que identifican a los participantes siempre ofrecerá:**
  - **Unicidad:** no puede existir mas de una ocurrencia de las misma interrelación que asocie a los mismos participantes
  - **y no nulidad:** el grado de la interrelación es el mismo para todas sus ocurrencias

En el ejemplo de la relación SP, sería la combinación **(S# , P#)**.

- Otra posibilidad sería **introducir un nuevo atributo, no compuesto, que funcione como clave primaria de la interrelación**. Por ejemplo, **número de oferta**.
  - **El diseñador de la base de datos puede tener objeciones con respecto a las claves primarias compuestas** (el grado de la interrelación puede ser alto).

Ejemplos de Interrelaciones Muchos a Muchos

- A efectos del ejemplo podríamos decidirnos por la primera opción, añadiendo a la proposición **create table** la cláusula **primary key (S# , P#)**.

```
create table SP
    (S#  . . . ,
     P#  . . . ,
         . . . )
primary key (S# , P#)
foreign key (S#) references S
        delete of S restricted
        update of S.S# cascade
foreign key (P#) references P
        delete of P restricted
        update of P.P# cascade;
```

Ejemplos de Interrelaciones Muchos a Muchos

- En lo que respecta a la interrelación **PROY_TRABAJO**:
  - Suponiendo que las relaciones **EMP** y **J** representan a las entidades participantes empleado y proyecto, respectivamente.
  - Suponiendo que las claves primarias de ambas relaciones son, respectivamente, **NUMEMP** y **J#**.

Se puede representar esta interrelación con una relación base **JEMP** con la siguiente definición:

```
create table JEMP
    (NUMEMP  . . . ,
     J#              . . . ,
                     . . . )
    primary key (NUMEMP , J#)
    foreign key (NUMEMP) references EMP
        delete of EMP . . .
        update of EMP.NUMEMP . . .
    foreign key (J#) references J
        delete of J . . .
        update of J.J# . . .
```

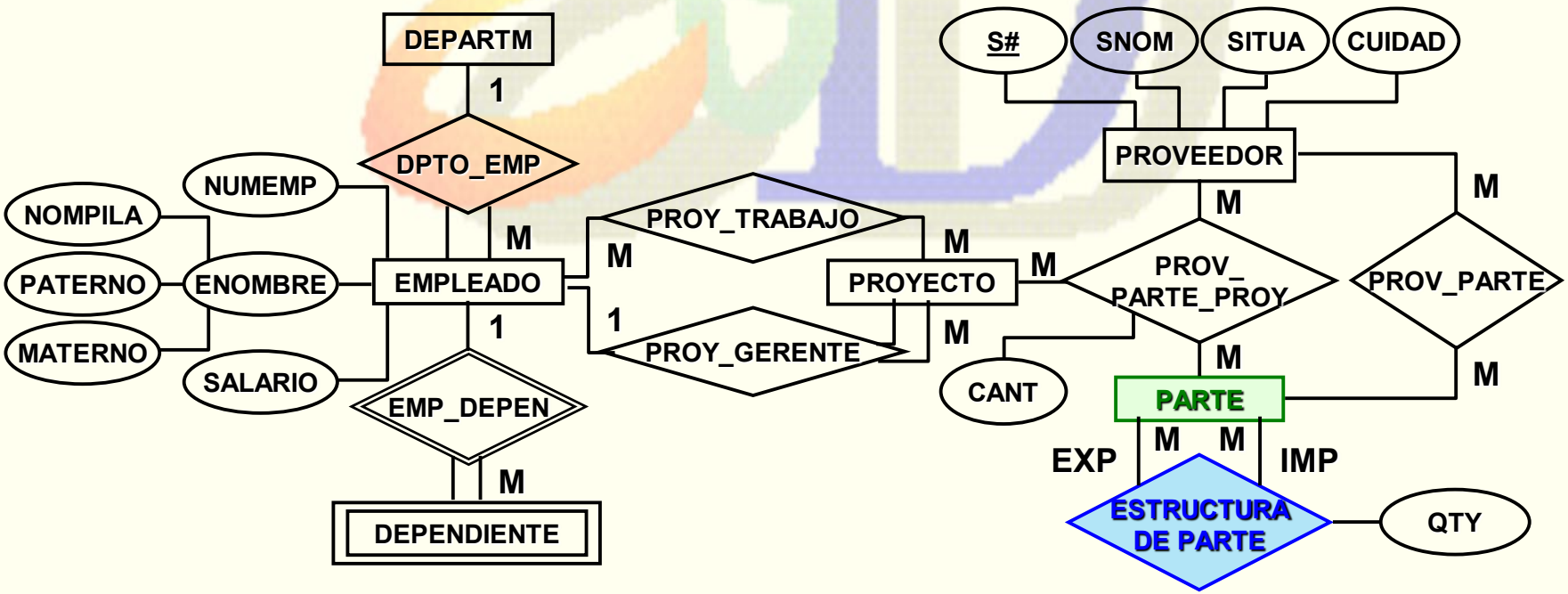
Ejemplos de Interrelaciones Muchos a Muchos

- Igualmente, se propagan las actualizaciones de los códigos que actúan como claves primarias de las relaciones que representan a las entidades participantes.
- La novedad aquí consiste en relajar la restricción de las eliminaciones de participantes, ya que **se puede establecer que la eliminación de, por ejemplo, un empleado, debe aceptarse y propagarse automáticamente a todas sus implicaciones de la empresa**.

```
create table JEMP
    (NUMEMP     . . . ,
     J#         . . . ,
                . . . )
    primary key (NUMEMP , J#)
    foreign key (NUMEMP) references EMP
        delete of EMP cascade
        update of EMP.NUMEMP cascade
    foreign key (J#) references J
        delete of J cascade
        update of J.J# cascade;
```

Ejemplos de Interrelaciones Muchos a Muchos

- En la interrelación **estructura_de_parte** aparece la misma entidad (**parte**) participando doblemente.
- Se puede representar con una relación PP **con dos claves ajenas a EP# e IP#, que referencian ambas a las partes en sus facetas de explosión de partes e implosión de partes, respectivamente**.



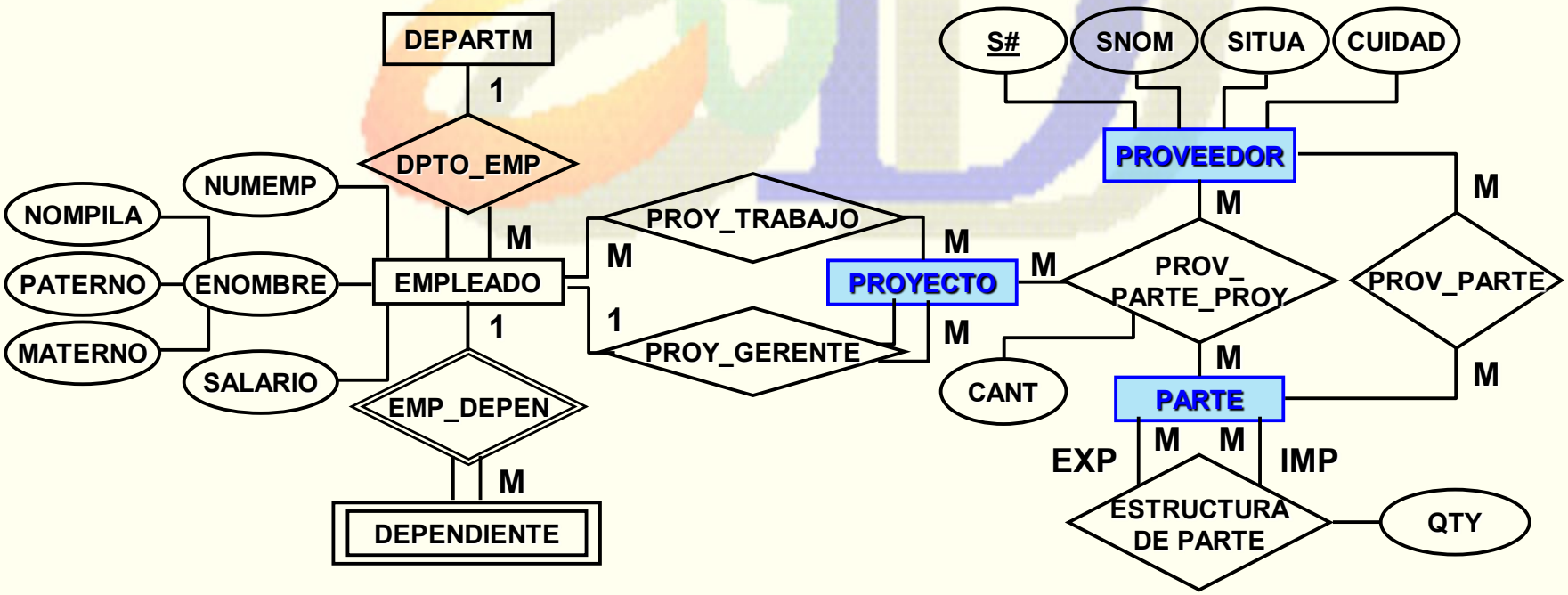
Ejemplos de Interrelaciones Muchos a Muchos

- La clave primaria de la relación PP puede ser la combinación **(EP# , IP#)**.
- La definición de la relación PP podría expresarse como se indica en la tabla siguiente.
  - Se pueden aplicar aquí comentarios similares a los de los ejemplos anteriores, referentes a la regla de actualización.
- Nótese sin embargo, cómo parece lógico pensar, **que no es admisible la eliminación sin más de una parte que puede ser componente de otras**.
- Mientras que al eliminar una parte no hay inconveniente en borrar su descomposición.

```
create table PP
    (EP#      . . . ,
     IP#      . . . ,
              . . . )
primary key (EP# , IP#)
foreign key (EP#) references P
        delete of P cascade
        update of P.P# cascade
foreign key (IP#) references P
        delete of P restricted
        update of P.P# cascade;
```

Ejemplos de Interrelaciones Muchos a Muchos

- En la interrelación **prov_parte_proy** aparecen **tres participantes** y podemos representarla con la **relación SPJ**.
- Obviamente, los comentarios sobre la clave primaria de SPJ son similares a todos los anteriores.
- Aparecerán **tres claves ajenas**: **S#**, **P#**, **J#**, que referencian a **proveedores**, **partes** y **proyectos**, respectivamente.



Ejemplos de Interrelaciones Muchos a Muchos

```
create table SPJ
    (S#  . . . ,
     P#  . . . ,
     J#  . . . ,
         . . . )
    primary key (S# , P# , J#)
    foreign key (S#) references S
        delete of S restricted
        update of S.S# cascade
    foreign key (P#) references P
        delete of P restricted
        update of P.P# cascade
    foreign key (J#) references J
        delete of J cascade
        update of J.J# cascade;
```

- **Debe imposibilitarse la desaparición** (sin más tramite) **de los proveedores y de las partes que estén participando en un proyecto**.

Al menos se puede pensar que habrán de ser sustituidos/das por otros/as de similares características.

Por ello **la regla de eliminaciones de los proveedores y la de las partes están restringidas**.

Ejemplos de Interrelaciones Muchos a Muchos

```
create table SPJ
    (S#  . . . ,
     P#  . . . ,
     J#  . . . ,
         . . . )
    primary key (S# , P# , J#)
    foreign key (S#) references S
        delete of S restricted
        update of S.S# cascade
    foreign key (P#) references P
        delete of P restricted
        update of P.P# cascade
    foreign key (J#) references J
        delete of J cascade
        update of J.J# cascade;
```

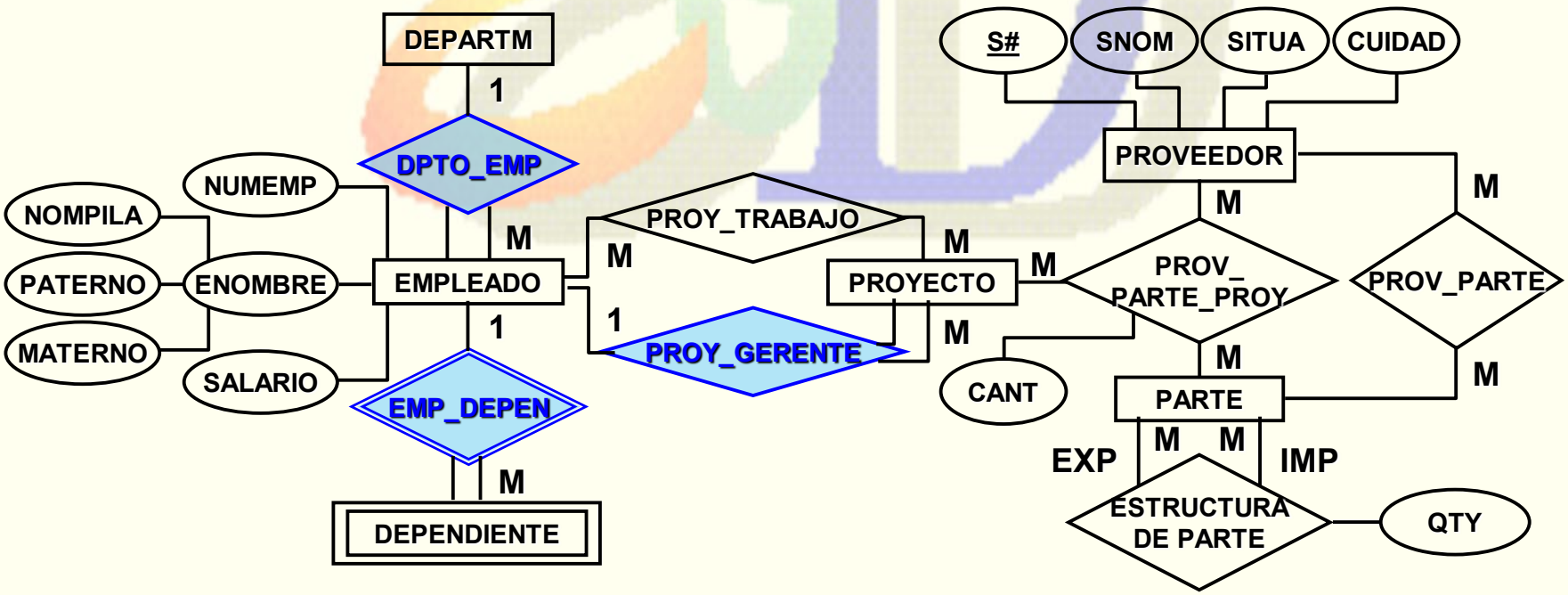
- Sin embargo, la eliminación de un proyecto debe aceptarse, propagándose este eliminación a todos los participantes en el mismo.

  Por ello, la regla de eliminación para los proyectos aparece relajada, de forma que se propaga.

- Por último, a las reglas de actualización de las claves primarias de las relaciones que representan a los participantes se le aplican, en este caso, similares comentarios que en los anteriores.

Ejemplos de Interrelaciones Muchos a Uno

- En el ejemplo existen tres interrelaciones de este tipo:
  - **PROY_GERENTE:** Los proyectos indican a sus gerentes.
  - **DEPTO_EMP:** Los empleados señalan a sus departamentos.
  - **EMP_DEPEN:** Los dependientes señalan a los empleados.
- La última implica un tipo de entidad débil, dependiente.
- Las otras dos implican solo tipos de entidades regulares.



Interrelaciones Muchos a Uno

- Estas interrelaciones **no provocan la introducción de nuevas relaciones**.
- En su lugar, **basta con introducir una clave ajena en la relación correspondiente al lado de muchos en la interrelación que haga referencia a la relación del lado de uno**.
- De esta forma, en el caso de la interrelación **DEPTO_EMP**:

```
create table EMP
    (NUMEMP      . . . ,
     NUMDEPTO    . . .  not null,
                 . . . )
    primary key (NUMEMP)
    foreign key (NUMDEPTO) references DEPTO
        delete of DEPTO restricted
        update of DEPTO.NUMDEPTO cascade;
```

Bajo la **<u>suposición</u>** de que **no podremos eliminar un departamento que tenga asignado empleados, necesitándose reasignar previamente los empleados a otro departamento para, a continuación, proceder a la eliminación**.

Interrelaciones Muchos a Uno

- Aunque **también cabría suponer** que **podrían eliminarse los empleados de un departamento al desaparecer este (regla de eliminación en cascada en lugar de restringida) si así lo establece la política de la empresa**:

```
create table EMP
    (NUMEMP      . . . ,
     NUMDEPTO    . . .  not null,
                 . . . )
    primary key (NUMEMP)
    foreign key (NUMDEPTO) references DEPTO
        delete of DEPTO cascade
        update of DEPTO.NUMDEPTO cascade;
```

- En cualquiera de los dos casos **representaremos fielmente la participación total de EMPLEADO en la relación DPTO_EMP**
  - **mediante la no nulidad de la clave ajena.**

Ejemplos de Interrelaciones Muchos a Uno

```
create table EMP
    (NUMEP      . . . ,
     NUMDEPTO   . . . ,
                . . . )
    primary key (NUMEMP)
    foreign key (NUMDEPTO) references DEPTO
        delete of DEPTO set null
        update of DEPTO.NUMDEPTO cascade;
```

- **Esta tabulación sería válida si la participación de EMPLEADO en DPTO_EMP fuese parcial.**
- Para un empleado dado:
  - podríamos desconocer (al menos temporalmente) cuál es su departamento
    - Por ejemplo, durante un proceso de reestructuración de la empresa en el que desaparecen algunos departamentos. Por ello, **pueden ser lógicos los valores que representan sus reglas de nulos y de eliminación**.
  - o bien podría no ser aplicable un departamento para ese empleado

Ejemplos de Interrelaciones Muchos a Uno

- La interrelación **PROY_GERENTE** se representa **incluyendo en la relación J** (que representa a los proyectos) **la clave ajena NUMEMP** (que referencia al empleado que actúa como gerente del proyecto:

```
create table J
      (J#              . . . ,
       NUMEMP          . . .  not null,
                       . . . )
      primary key (J#)
      foreign key (NUMEMP) references EMP
            delete of EMP restricted
            update of EMP.NUMEMP cascade;
```

Ejemplos de Interrelaciones Muchos a Uno

```
create table J
    (J#          . . . ,
     NUMEMP      . . . not null,
                 . . . )
    primary key (J#)
    foreign key (NUMEMP) references EMP
        delete of EMP restricted
        update of EMP.NUMEMP cascade;
```

- Es obvio que, en este caso, pueden comentarse las reglas para claves ajenas de forma parecida a las del ejemplo anterior:
  - La **restricción parece apropiada** dado que **no es lógico pensar que la eliminación de un gerente cause la desaparición del proyecto que maneja**.
  - La **anulación está contraindicada por la participación total de PROYECTO** en **PROY_GERENTE**.

    Aunque quizás convendría redefinir dicha participación como **parcial**, dado que alguna reasignación de personal podría dejar a un proyecto sin gerente temporalmente.