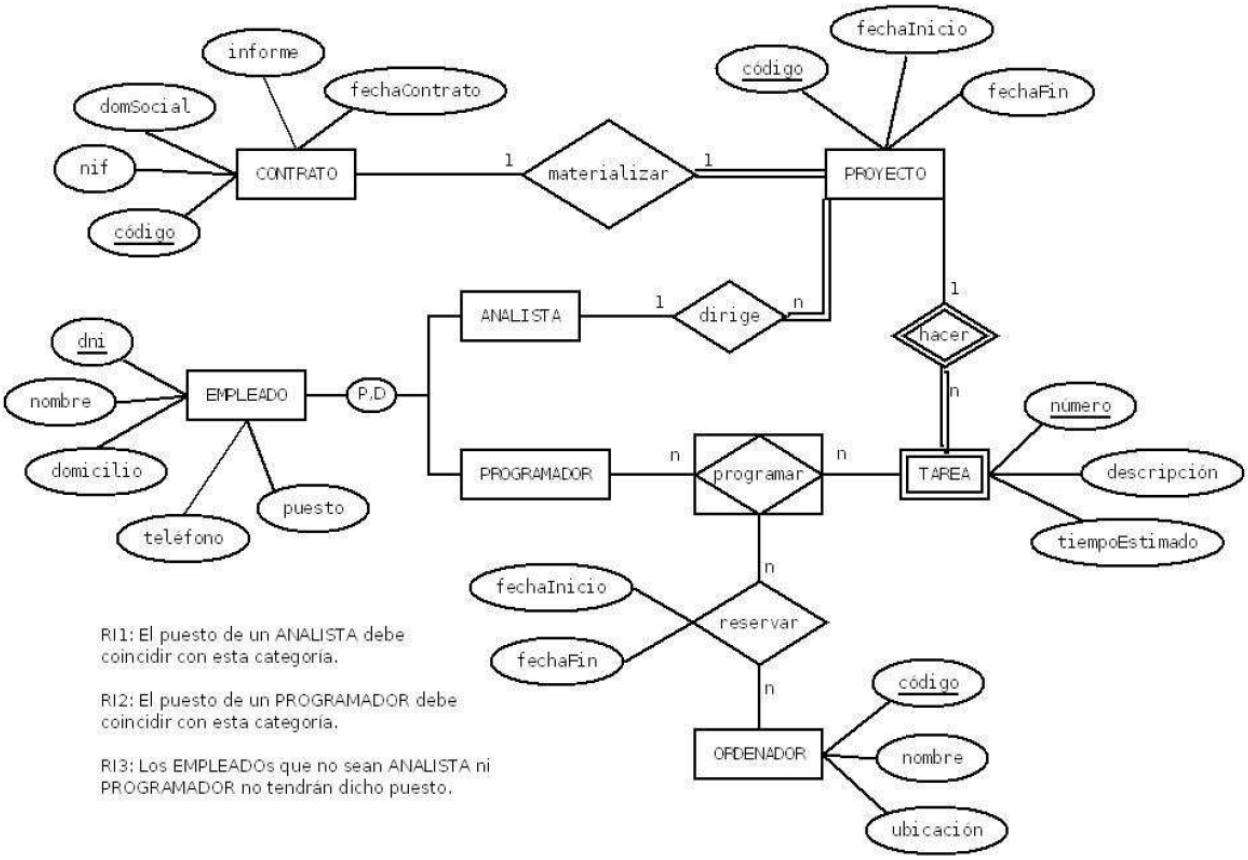
Empresa SW



1) Transformamos las **entidades fuertes** y sus atributos.

```
CONTRATO (codigo, nif, domSocial, informe, fechaContrato)
CP: {codigo}

EMPLEADO (dni, nombre, domicilio, telefono, puesto)
CP: {dni}

PROYECTO (codigo, fechaInicio, fechaFin)
CP: {codigo}

ORDENADOR (código, nombre, ubicacion)
CP: {codigo}
```

2) Añadimos las **restricciones de integridad** (RI) detectadas en la fase de análisis.

```
RI1: El puesto de un ANALISTA debe coincidir con esta categoría.
RI2: El puesto de un PROGRAMADOR debe coincidir con esta categoría.
RI3: Los EMPLEADOs que no sean ANALISTA ni PROGRAMADOR no tendrán
dicho puesto.
```

3) Transformamos las especializaciones de **EMPLEADO**.

```
ANALISTA (dni)
CP: {dni}
CAjena: {dni} -> EMPLEADO.dni

PROGRAMADOR (dni)
```

CP: {dni}
CAjena: {dni} -> EMPLEADO.dni

Ahora nos fijamos en el **tipo de la especialización**:
- Parcial: este tipo de especialización no necesita ninguna RI adicional.
- Disjunta: necesitamos la siguiente RI adicional.

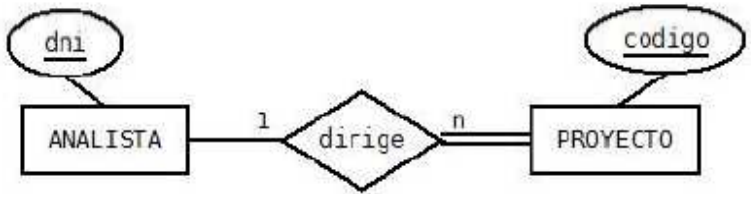
RI4: Los ANALISTA no pueden ser PROGRAMADORes ni viceversa.

4) Transformamos las **entidades débiles** y sus atributos. Un PROYECTO puede desglosarse en muchas TAREAs pero una TAREA solo pertenecerá a un PROYECTO.
El valor del atributo TAREA.numero puede repetirse para distintos PROYECTOs pero no habrá dos TAREAs dentro del mismo PROYECTO con el mismo valor en dicho atributo.

TAREA (proyecto, numero, descripcion, tiempoEstimado)
CP: {proyecto, numero}
CAjena: {proyecto} -> PROYECTO.codigo

5) Transformamos las **relaciones** entre entidades y sus atributos.

a) Relación binaria DIRIGE 1:N



PROYECTO (codigo, fechaInicio, fechaFin, **dniAnalista**)
CP: {ódigo}
CAjena: {dniAnalista} -> ANALISTA.dni
VNN: {dniAnalista}
NOTA: Si obligamos a informar siempre el atributo **dniAnalista**, nos aseguramos de que todos los PROYECTOs tienen un ANALISTA asignado que los dirige.

b) Relación binaria MATERIALIZAR 1:1

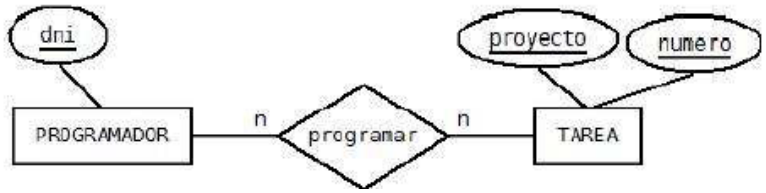
PROYECTO (codigo, fechaInicio, fechaFin, dniAnalista, **codContrato**)
CP: {ódigo}
CAjena: {dniAnalista} **->** ANALISTA.dni
VNN: {dniAnalista}
CAjena: {codContrato} -> CONTRATO.codigo
UNICO: {codContrato}
NOTA: Un valor CONTRATO.codigo solo puede aparecer una vez, como máximo, en PROYECTO.codContrato. Eso quiere decir que un mismo CONTRATO no estará materializado/asociado a más de un PROYECTO.

Podíamos haber añadido la transformación de MATERIALIZAR en CONTRATO en vez de en PROYECTO pero como tenemos una RE en PROYECTO es más fácil de captar si lo hacemos de esta forma.

c) Relación binaria HACER 1:N

La relación binaria HACER se ha transformado al realizar la transformación de la entidad débil TAREA y no hay ninguna restricción especial que haga falta añadir, así que ya la tenemos transformada.

d) Relación binaria PROGRAMAR N:N

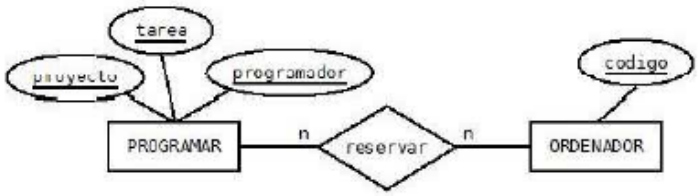


PROGRAMAR (programador, proyecto, tarea)
CP: { programador, proyecto, tarea }
CAjena: {programador} -> PROGRAMADOR.dni
CAjena: {proyecto, tarea} -> {TAREA.proyecto, TAREA.numero}
NOTA: Estaría también bien si lo expresases así:
CAjena: {proyecto, tarea} ->TAREA

e) Relación binaria RESERVAR N:N
En esta relación participa la agregación PROGRAMAR y su clave principal se corresponde con la clave principal de su relación interna.

Para que nos resulte más sencilla la transformación de la relación RESERVAR podemos tratar la agregación PROGRAMAR **como si fuese** una ENTIDAD.

Podemos imaginarnos que el E/R queda así:

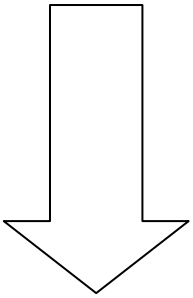


Ahora solo tenemos que transformar la relación N:N como siempre.
RESERVAR (proyecto, tarea, programador, ordenador)
CP: { proyecto, tarea, programador, ordenador }
CAjena: {proyecto, tarea, programador} -> PROGRAMAR
CAjena: {ordenador} -> ORDENADOR

El esquema lógico relacional final es el siguiente:

```
CONTRATO (codigo, nif, domSocial, informe, fechaContrato)
CP: {codigo}

EMPLEADO (dni, nombre, domicilio, telefono, puesto)
CP: {dni}

ORDENADOR (código, nombre, ubicacion)
CP: {codigo}
RI1: El puesto de un ANALISTA debe coincidir con esta categoría.
RI2: El puesto de un PROGRAMADOR debe coincidir con esta categoría.
RI3: Los EMPLEADOs que no sean ANALISTA ni PROGRAMADOR no tendrán
dicho puesto.

ANALISTA (dni)
CP: {dni}
CAjena: {dni} -> EMPLEADO.dni

PROGRAMADOR (dni)
CP: {dni}
CAjena: {dni} -> EMPLEADO.dni
RI4: Los ANALISTA no pueden ser PROGRAMADORes ni viceversa.

TAREA (proyecto, numero, descripcion, tiempoEstimado)
CP: { proyecto, numero }
CAjena: {proyecto} -> PROYECTO.codigo

PROYECTO (codigo, fechaInicio, fechaFin, dniAnalista, codContrato)
CP: {codigo}
CAjena: {dniAnalista} -> ANALISTA.dni
VNN: {dniAnalista}
CAjena: {codContrato} -> CONTRATO.codigo
UNICO: {codContrato}

PROGRAMAR (programador, proyecto, tarea)
CP: { programador, proyecto, tarea }
CAjena: {programador} -> PROGRAMADOR.dni
CAjena: {proyecto, tarea} -> (TAREA.proyecto, TAREA.numero}

RESERVAR (proyecto, tarea, programador, ordenador)
CP: { proyecto, programador, proyecto, tarea }
CAjena: {proyecto, tarea, programador} -> PROGRAMAR
CAjena: {ordenador} -> ORDENADOR
```

MAS



¿CLAVES PRIMARIAS COMPUESTAS POR DEMASIADOS ATRIBUTOS?

Cuando lleguemos a la fase de creación/implementación de la base de datos nos daremos cuenta de que no es muy útil tener claves principales compuestas por más de dos o tres atributos.

La solución es inventarnos una **clave principal artificial** formada por un solo atributo y poner nuestra clave principal compuesta como una clave alternativa. De esta forma el acceso a la tabla es mucho más eficiente.

Podemos esperar hasta la fase de implementación para solucionarlo o hacerlo en esta fase.

Si lo solucionásemos ahora, las relaciones PROGRAMAR y RESERVAR quedarían así:

```
PROGRAMAR (codProgramar, programador, proyecto, tarea)
CP: {codPRogramar}
UNICO: {programador, proyecto, tarea}
CAjena: {programador} -> PROGRAMADOR.dni
CAjena: {proyecto, tarea} -> {TAREA.proyecto, TAREA.numero}

RESERVAR (codReservar, proyecto, tarea, programador, ordenador)
CP: { codReservar }
UNICO: {proyecto, tarea, programador, ordenador}
CAjena: {proyecto, tarea, programador} -> PROGRAMAR
CAjena: {ordenador} -> ORDENADOR
```