Algunas consideraciones generales para el diseño:

1) Tanto la fuente de la que procede la receta, como la ubicación del libro, cinta de vídeo, etc., con la información original son simples atributos del tipo de entidad receta. Puesto que es posible que no se tenga esta información de todas las recetas (aunque no se especifica nada en el enunciado), se opta por especificar estos atributos como opcionales.

2) Si bien en una primera aproximación se podría pensar en identificar receta con plato, se ha optado por modelar ambos conceptos de forma separada. Las razones básicas son las siguientes:

   - La información del tipo de plato, precio y del ingrediente principal está asociada a cada plato, independientemente de la receta con que se ha elaborado.
   - Se indica que los menús están formados por varios platos, sin que importe el procedimiento de elaboración de cada plato (la receta).

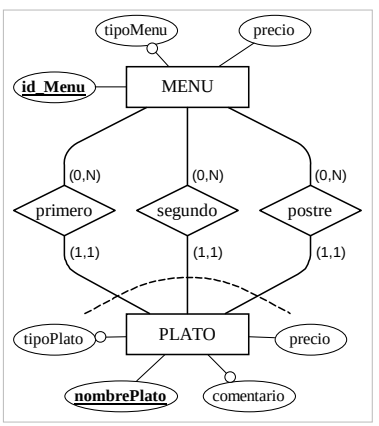
   Por otra parte, se considera que cada receta sirve para elaborar uno y sólo un plato determinado, y que de cada uno de los platos se tiene al menos una receta. Esta decisión de diseño parece acertada en el contexto del enunciado, aunque no es adecuada para el apartado a) de las modificaciones propuestas, donde una receta no tiene por qué constituir ningún plato concreto, sino que describe un "ingrediente elaborado" que puede ser empleado en otras recetas.

   Aunque lo más natural sería modelar receta como un tipo de entidad débil con respecto a plato (cada receta se identificaría con el nombre del plato y otro atributo que permitiera distinguir entre las diferentes recetas de un plato), puesto que en el enunciado se indica que las recetas se identifican con un número clave, se opta por modelar receta como un tipo de entidad fuerte (tampoco podría modelarse como débil si se permitieran recetas que no están asociadas a un plato concreto)

3) Para representar la lista de pasos de una receta se ha optado por una solución simple y eficaz, que consiste en identificar cada paso por la receta a que pertenece y un número de orden correlativo. Es decir, se ha modelado el *pasoReceta* como un tipo de entidad débil con respecto a receta, que tiene a *numPaso* como discriminador.

4) Para representar los utensilios utilizados en cada paso de una receta, dado que sólo interesa el nombre del utensilio, basta con utilizar un atributo multivaluado y opcional (no en todos los pasos tiene por qué emplearse alguno). Si se dispusiera de información adicional del utensilio (tamaño, material, etc.) sería necesario representarlo con un tipo de entidad independiente. El esquema E/R 2 ilustra este caso donde, además, se ha añadido un identificador adicional que, aunque no es necesario, permite ilustrar otros conceptos (nombre del utensilio será atributo identificador alternativo)

5) Siguiendo las pautas indicadas para los utensilios, se representan los ingredientes como un tipo de entidad, ya que hay que representar las calorías de cada ingrediente. Para justificar más esta idea se ha añadido el atributo propiedades (no necesario, pues no se menciona en el enunciado). Tanto la cantidad empleada en cada paso como las unidades empleadas en cada paso constituyen atributos de la interrelación correspondiente. Se ha especificado *unidad* como atributo opcional ya que, en algunos casos, puede no ser necesario (p.e. 2 huevos).

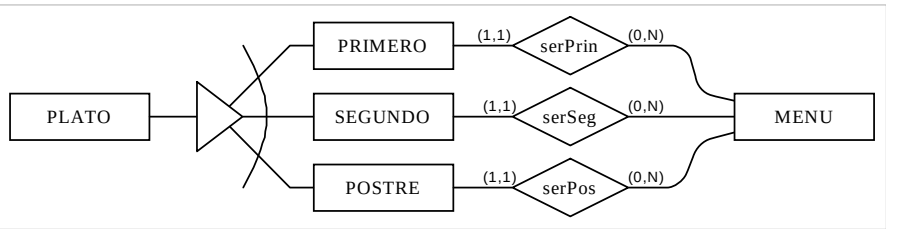


6) El modelado del menú y de su composición sugiere varias alternativas:

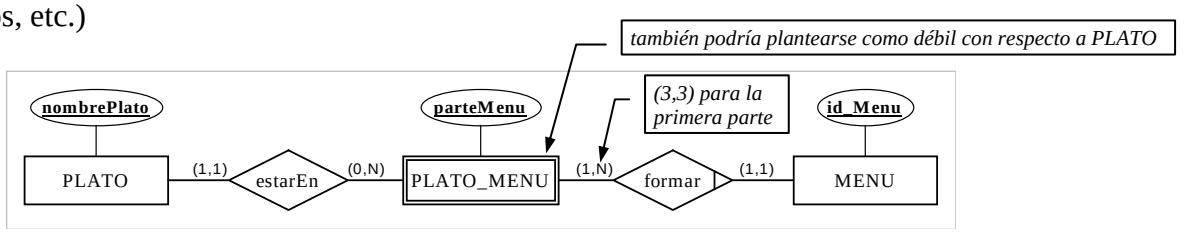
   - Si no importa qué plato constituye el primer plato, segundo o postre, se podría considerar la composición como una simple interrelación entre menú y plato con cardinalidades (0,N) y (3,3), respectivamente. Esta solución garantiza que no se puede "repetir" un plato en el menú. Para añadir la información del orden bastaría un atributo en la interrelación, pero haría falta una restricción adicional que garantice que en un menú no haya papeles repetidos (p.e. dos primeros).
   - Otra solución simple y eficaz para representar el papel de cada plato consiste en utilizar un tipo de interrelación para cada uno de ellos. El inconveniente de esta solución es que hay que añadir una restricción adicional que garantice que los platos de un menú son distintos. Esta es, quizás, la solución más eficaz y fácil de implementar en el modelo relacional.

Si un plato no puede tener diferentes papeles en los menús (siempre es primero, o segundo, o postre, o nada), basta con especificar una exclusión de las interrelaciones con plato.

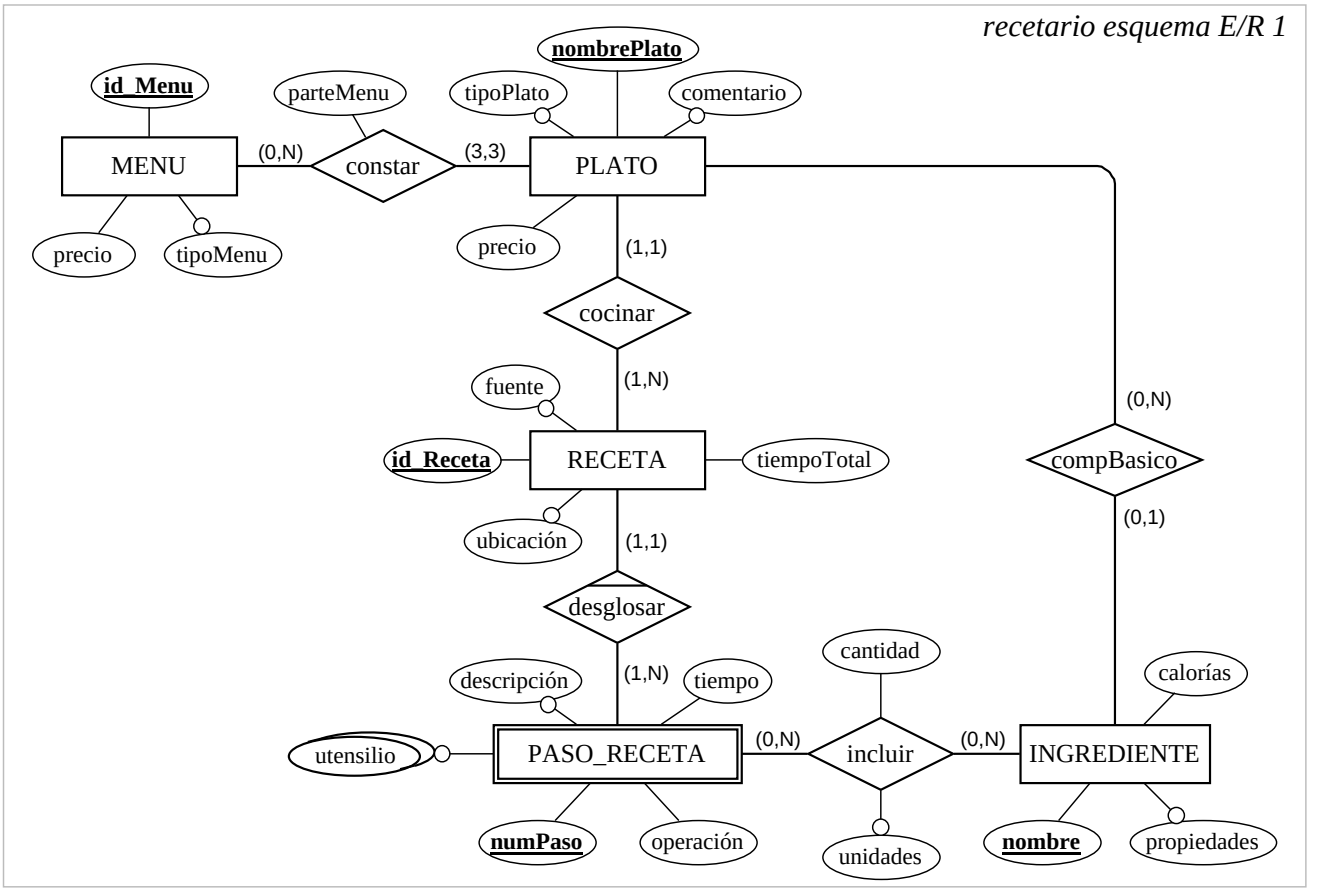
- Otra solución adecuada, que incluye de forma natural esta última situación, consiste en representar los platos mediante una especialización con exclusión entre los subtipos.
  Aunque este esquema es más elegante y general (permite representar más información), puede complicar un poco más la representación de forma innecesaria, por lo que no se utiliza.



- Otra solución bastante interesante consiste en representar cada plato del menú con un tipo de entidad débil con respecto a menú, que tiene como atributo discriminador *parteMenu*, para describir la parte del menú que representa (primero, segundo, postre). La principal ventaja de esta solución es que permite configurar menús más complejos de un modo más fácil (p.e. menús con entrantes, aperitivos, etc.)

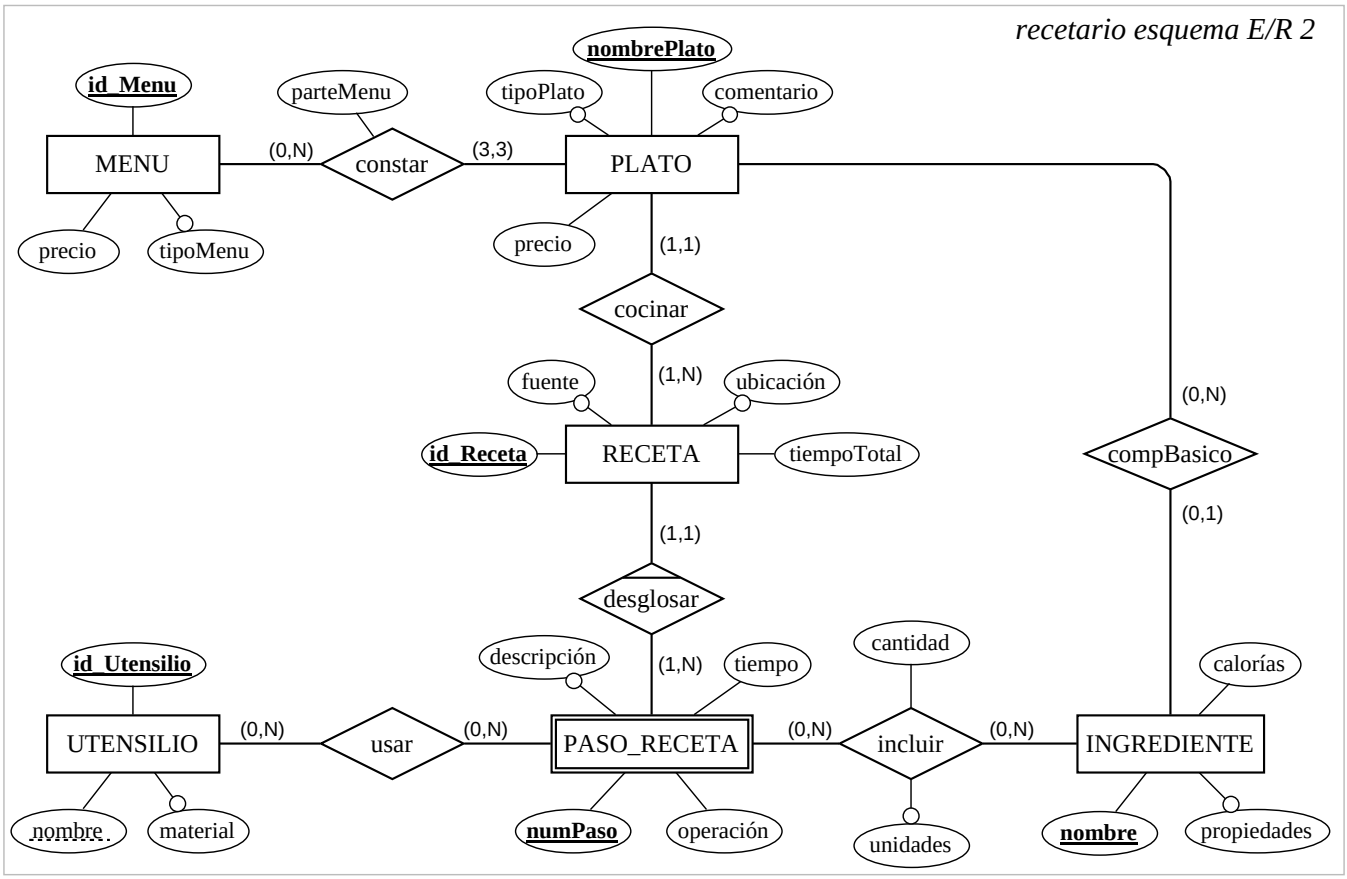


Teniendo en cuenta las consideraciones efectuadas, se puede plantear el siguiente esquema E/R

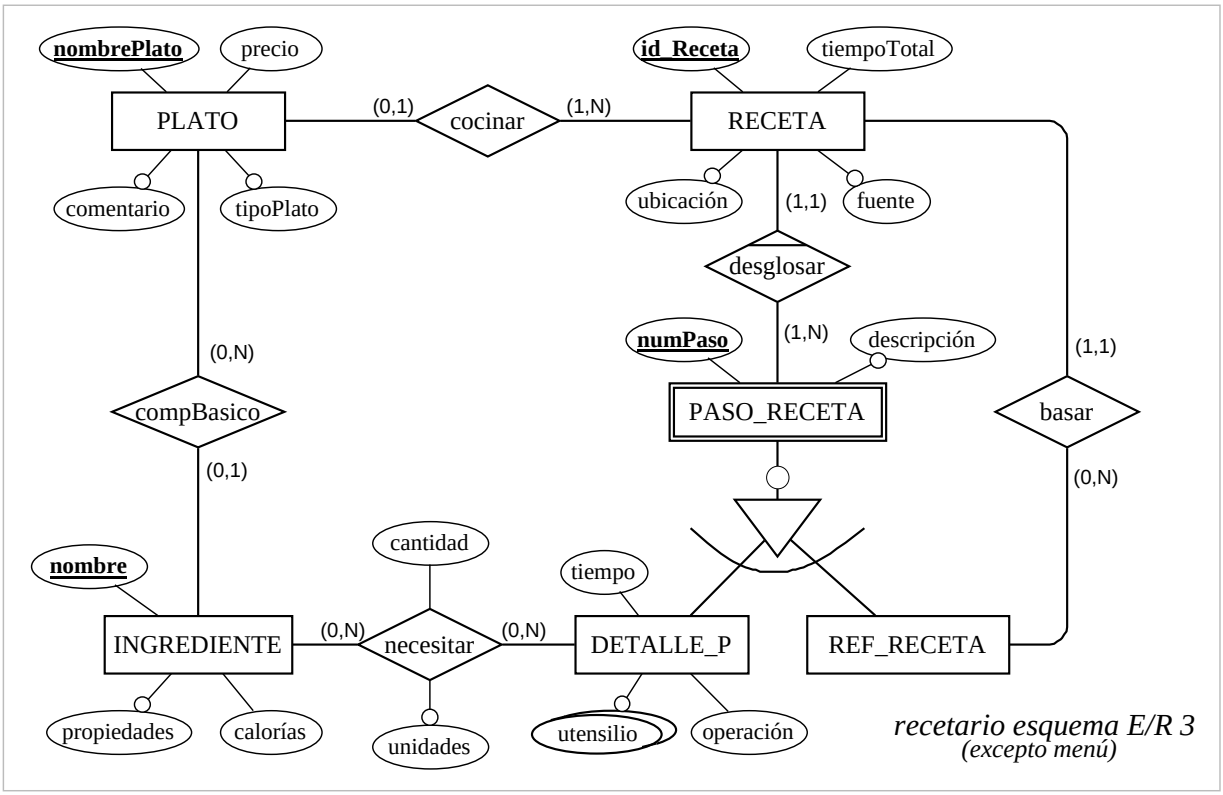


- ☛ *todo ingrediente básico de un plato debe formar parte de la lista de ingredientes de algún paso de las recetas*
- ☛ *verificar que* parteMenu *no se repite para los platos de un mismo menú*
- ☛ • • •

Si deseara tener más información (material, tamaño, etc.) de los utensilios empleados, se podría plantear el siguiente esquema, con las mismas restricciones del esquema anterior (es el que se transformará al modelo relacional, salvo la parte del menú):



7) La posibilidad de que un paso de una receta consista en la realización de una receta determinada se puede modelar mediante una especialización del tipo de entidad *pasoReceta* en los subtipos *refReceta* y *detalleP*. Además hay que cambiar la cardinalidad de la interrelación plato con receta a (0,1), puesto que una receta no tiene por qué constituir un plato.



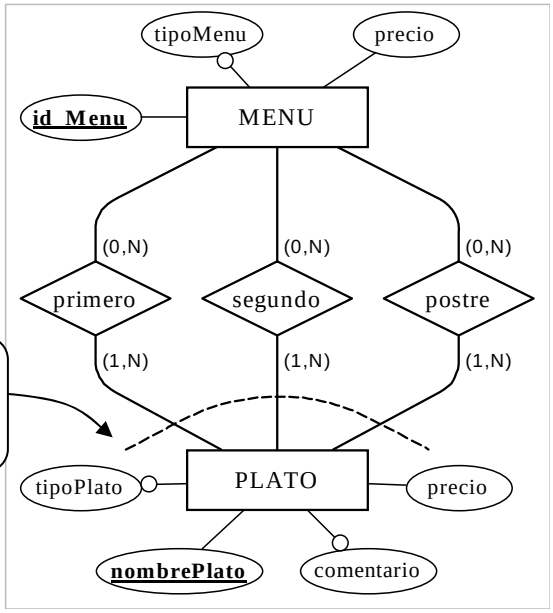
☛ *una receta no puede estar basada en sí misma.*

Nota: En todos los esquemas E/R se ha utilizado la notación de cardinalidad de la interrelación.

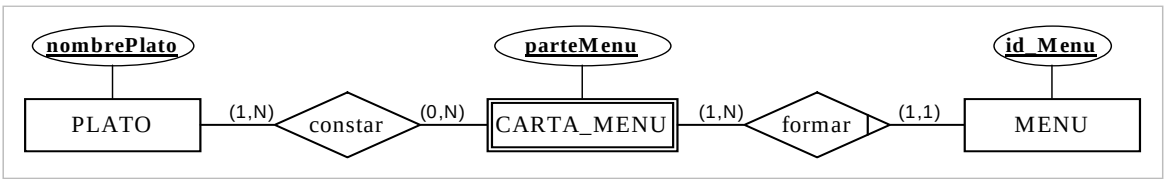
8) La posibilidad de que un menú disponga de varios platos para elegir se puede modelar cambiando simplemente la cardinalidad máxima de las interrelaciones entre menú y plato.

si no se especifica la exclusión de las interrelaciones con plato (un plato podría pertenecer a partes distintas en diferentes menús), entonces hay que:

☛ *verificar que los platos de un menú son distintos.*



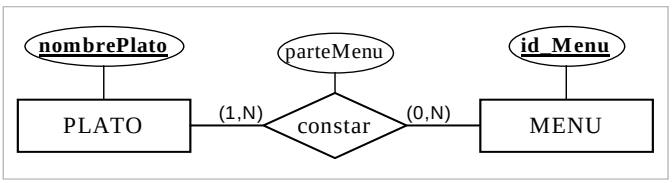
También se puede usar la solución basada en la representación explícita de las partes del menú:



No obstante, dado que en este problema la mayor ventaja de esta solución es que permite que haya platos repetidos en un menú (en diferentes partes), que no es necesario, resulta más simple y eficaz modelarlo como en la primera propuesta. Además hay que

☛ *verificar que* ∀MENU ∃*ocurrencias en* constar *con todas* parteMenu

Obsérvese que en el modelo relacional la solución anterior conduce a esta última.



Nota: Para simplificar la presentación, en los esquemas E/R se ha omitido la descripción de los dominios, incluyéndose solamente en el esquema relacional (son los mismos).

Esquema relacional de la Base de Datos "Recetario"

DOMINIOS

```
tpNombre = cadena[32];                tpTexto  = cadena[255];
tpPlato  = ( carne, pescado, huevos, verdura, ... );
tpUnidad = ( gramo, pizca, cucharada, ... );
tpParteMenu = ( primero, segundo, postre );
```

ESQUEMAS DE RELACION *(esquema E/R 2)*

Receta (
- **id-Receta** : *natural*, ***clave primaria***;
- fuente : *tpNombre*;
- ubicacion : *tpNombre*;
- tiempoTotal : *tpTiempo*, ***NO NULO***;
- nombrePlato : *tpNombre*, ***NO NULO***, ***clave ajena de*** *Plato*);

☛ *verificar que toda receta tiene al menos un paso ≡∀* id_receta, *∃ ocurrencia en* PasoReceta

PasoReceta (
- **id-Receta** : *natural*, ***clave ajena de*** *Receta*;
- **numPaso** : *natural*;
- tiempo : *tpTiempo*, ***NO NULO***;
- descripcion : *tpTexto*;
- operación : *tpTexto*, ***NO NULO***;
- (id-Receta, numPaso) ***clave primaria***);

Utensilio (*{ sobraría en ER1 (*Utensilio *atributo multivaluado), quedando sólo* usoUtensilios(id-Receta, numPaso, nombUtensilio) *}*
- **id-Utensilio** : *natural*, ***clave primaria***;
- nombUtensilio: *tpNombre*, ***NO NULO, UNICO***;
- material : *tpNombre*);

usoUtensilios (
- **id-Receta** : *natural*;
- **numPaso** : *natural*;
- **id-Utensilio** : *natural*, ***clave ajena de*** *Utensilio*;
- (id-Receta, numPaso, id-Utensilio) ***clave primaria***;
- (id-Receta, numPaso) ***clave ajena de*** *pasoReceta*);

Ingrediente (
- **nombIngred** : *tpNombre*;
- calorias : *natural*, ***NO NULO***;
- propiedades : *tpTexto*);

usoIngredientes (
- **id-Receta** : *natural*;
- **numPaso** : *natural*;
- **nombIngred** : *tpNombre*, ***clave ajena de*** ingrediente;
- cantidad : *natural*, ***NO NULO;***
- unidades : *tpUnidad;*
- (id-Receta, numPaso, id-Ingred) ***clave primaria***;
- (id-Receta, numPaso) ***clave ajena de*** *pasoReceta*);

Plato (
- **nombrePlato** : *tpNombre*, ***clave primaria***;
- tipoPlato : *tpPlato*;
- comentario : *tpTexto*;
- precio : *natural*, ***NO NULO;*** *{en céntimos de euro, o redondeado a euros;, si no, nº real con 2 decimales}*
- ingredBasico: *tpNombre,* ***clave ajena de*** *Ingrediente*);

☛ *verificar que de todo plato hay al menos una receta.*
☛ *verificar que si* ingredBasico *es* no nulo, *se usa en todas las recetas del plato.*

Menu (*{corresponde al primer esquema E/R presentado, con 3 interrelaciones binarias}*
- **id-Menu** : *natural*, ***clave primaria***;
- tipoMenu : *tpNombre*;
- precio : *natural*, ***NO NULO***;
- primero : *tpNombre*, ***NO NULO***, ***clave ajena de*** *Plato*;
- segundo : *tpNombre*, ***NO NULO***, ***clave ajena de*** *Plato*;
- postre : *tpNombre*, ***NO NULO***, ***clave ajena de*** *Plato*);

☛ *verificar que* primero, segundo *y* postre *son distintos.*

el esquema correspondiente al tipo de entidad débil PlatoMenu (apartado 6) quedaría:

```
Plato_Menu (
     id-Menu   : natural,  clave ajena de Menu;
     parteMenu : tpParteMenu;
     refPlato  : tpNombre, NO-NULO, clave ajena de Plato;
     (id-Menu, refPlato) UNICO;
     (id-Menu, parteMenu) clave primaria);
```

garantiza que no se repiten platos en un menú

☛ *verificar que* ∀ id-Menu, ∃ *ocurrencia en* PlatoMenu

Obsérvese que queda más compleja que la solución anterior.

*apartado **a)***

hay que modificar PasoReceta, *añadiendo* refReceta *y quitando la restricción de NO-NULO al atributo* tiempo

```
PasoReceta (
     id-Receta   : natural;
     numPaso     : natural;
     refReceta   : natural, clave ajena de Receta;
     tiempo      : tpTiempo;
     descripcion : tpTexto;
     (id-Receta, numPaso) clave primaria);
```

☛ *verificar que si* refReceta *es nulo*, tiempo *es NO-NULO, y que si* refReceta *es NO-NULO*, tiempo *es NULO.*

☛ *verificar que una receta no está basada en sí misma*

además, en las relaciones `usoUtensilios y usoIngredientes`, *hay que*

☛ *verificar que en el* PasoReceta *referenciado,* refReceta *es NULO.*

*apartado **b)***

las referencias a platos que hay en `menu` *se transforman en relaciones:*

```
Menu (
     id-Menu   : natural, clave primaria;
     tipoMenu  : tpNombre;
     precio    : natural, NO NULO);

cartaPrimeros (
     id-Menu   : natural,  clave ajena de Menu;
     refPlato  : tpNombre, clave ajena de Plato;
     (id-Menu, refPlato) clave primaria);

cartaSegundos (
     id-Menu   : natural,  clave ajena de Menu;
     refPlato  : tpNombre, clave ajena de Plato;
     (id-Menu, refPlato) clave primaria);

cartaPostres (
     id-Menu   : natural,  clave ajena de Menu;
     refPlato  : tpNombre, clave ajena de Plato;
     (id-Menu, refPlato) clave primaria);
```

☛ *verificar que todo menú tiene al menos un primero, un segundo, y un postre*
≡∀ id_Menú *en* Menú ∃ *ocurrencia en* cartaPrimeros, cartaSegundos, y en cartaPostres

otra solución más simple, correspondiente al último esquema E/R sería la siguiente:

```
Carta_Menu (
     id-Menu   : natural,  clave ajena de Menu;
     refPlato  : tpNombre, clave ajena de Plato;
     parteMenu : tpParteMenu, NO-NULO;
     (id-Menu, parteMenu) UNICO;     {≡sólo si un menú sólo tuviera 3 platos (uno por cada parte del menú)}
     (id-Menu, refPlato) clave primaria);
```

☛ *verificar que* ∀ id_Menú *en* Menú, y ∀parteMenu ∃ *ocurrencia en* carta_Menu *{≡todo menú tiene 3 platos}*

Observaciones:

1) *Para facilitar la lectura, la especificación de clave primaria se ha puesto de forma redundante (no sería necesario)*
2) *En la especificación de claves ajenas que no se indique lo contrario se sobreentiende que las op. de borrado y modificación están "restringidas"*