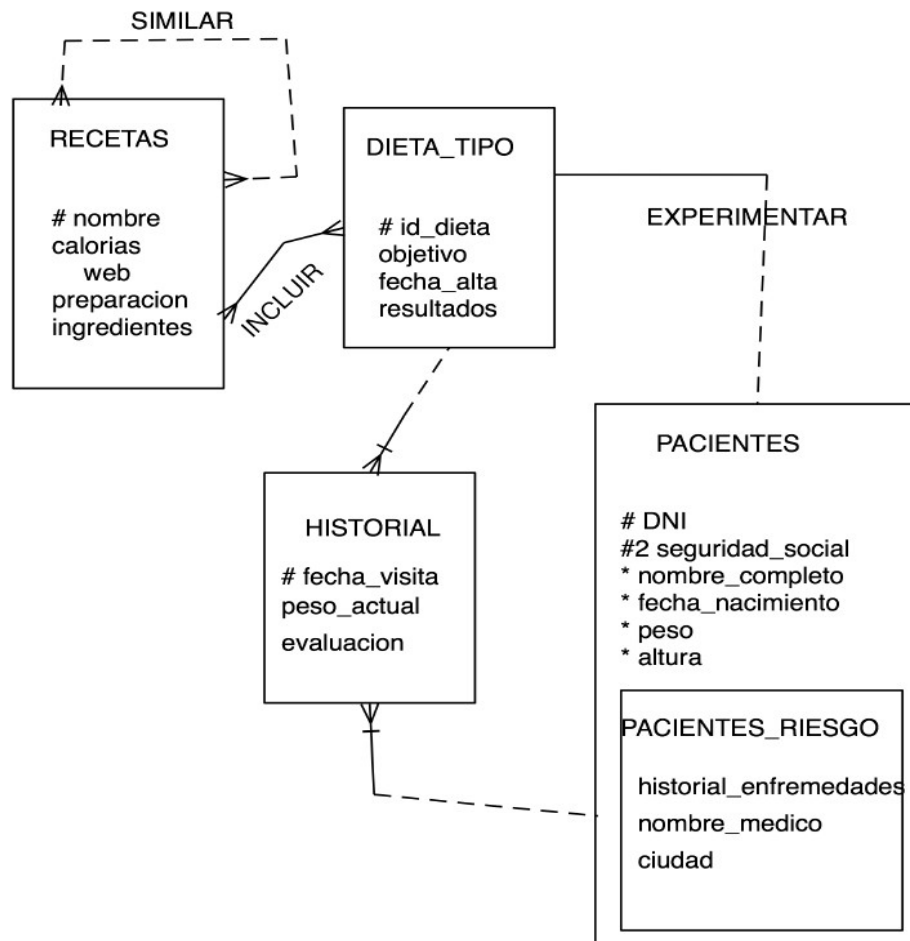


**Profesores: Manuel Enciso & Enrique Soler.**  
[www.lcc.uma.es](http://www.lcc.uma.es)

## Solución al ejercicio Pacientes de un Hospital

En esta práctica debemos modificar el esquema dado de la base de datos para adaptarlo al nuevo esquema representado por el siguiente diagrama Entidad/Relación:



Daremos una solución paso a paso. Para seguir leyendo, debes estar familiarizado con el lenguaje de consulta estructurado SQL (tanto en la manipulación de datos (DML) como en la definición de datos (DDL). Además debes haber estudiado la equivalencia entre el Modelo E/R y el Modelo


Relacional.

Usaremos el programa SQLDeveloper para realizar la práctica; este programa nos permite realizar todas las acciones necesarias de manera gráfica, pero optaremos aquí por usar las sentencias del lenguaje SQL que es un medio más adecuado para mostrar las acciones que queremos realizar. Por supuesto, está permitido que tú uses el modo gráfico en SQLDeveloper.

Para modificar el esquema antiguo primero debemos conocerlo; necesitamos información sobre las tablas existentes, sus relaciones, sus atributos y sus restricciones. La mejor manera de modificar un esquema es haciendo un pequeño diagrama de las tablas, atributos y relaciones disponibles, para así tener una visión global de la base de datos y poder apreciar mejor los cambios necesarios. Este diagrama lo podemos hacer a mano en dos minutos y será de gran ayuda durante el resto de la práctica. Además nos servirá para aprender a obtener un diagrama Entidad/Relación a partir de un esquema relacional de la base de datos. También puedes ayudarte de la herramienta SQL Data Modeler haciendo una importación desde tu esquema ayudado por el módulo de importación desde el diccionario de datos.

Lo primero es obtener información sobre las tablas disponibles. La siguiente sentencia SQL nos permite conocer que tablas existen en nuestra base de datos:

```
SELECT *  
FROM user_tables;
```



Recetas  
Pacientes\_Riesgo  
Dieta\_Tipo  
Pacientes

Para mostrar el esquema de cada tabla Recetas usamos la sentencia DESCRIBE de SQLPlus (ojo, no es del SQL):

```
DESC Recetas;
```

Obtendremos los atributos de la tabla Recetas, así como el tipo de cada atributo e información de si debe o no tener un valor obligatoriamente.

```
-----  
NOMBRE          NOT NULL          VARCHAR2(100)  
CALORIAS                               NUMBER(4,1)  
WEB              VARCHAR2(100)  
PREPARACION      VARCHAR2(1000)  
INGREDIENTES     VARCHAR2(1000)  
-----
```

En esta tabla la columna *nombre* no puede tener el valor NULL, es decir, debe contener un valor obligatoriamente, no pudiéndose dejar vacío.

La información del resto de tablas la obtenemos de la misma manera; esto es,

con la sentencia:

Mostar el esquema de una tabla:

```
DESC/DESCRIBE nombre_tabla;
```

Todavía no tenemos toda la información sobre la base de datos, pues todavía no conocemos las restricciones de unicidad (claves primarias y candidatas) y las restricciones de claves ajenas (o foráneas) que nos permiten implementar relaciones las tablas. Esta información la podemos obtener con la sentencia:

```
SELECT *  
FROM user_constraints;
```

Nos devolverá los nombres de todas las restricciones, el tipo de ellas y las tablas a las que pertenecen.

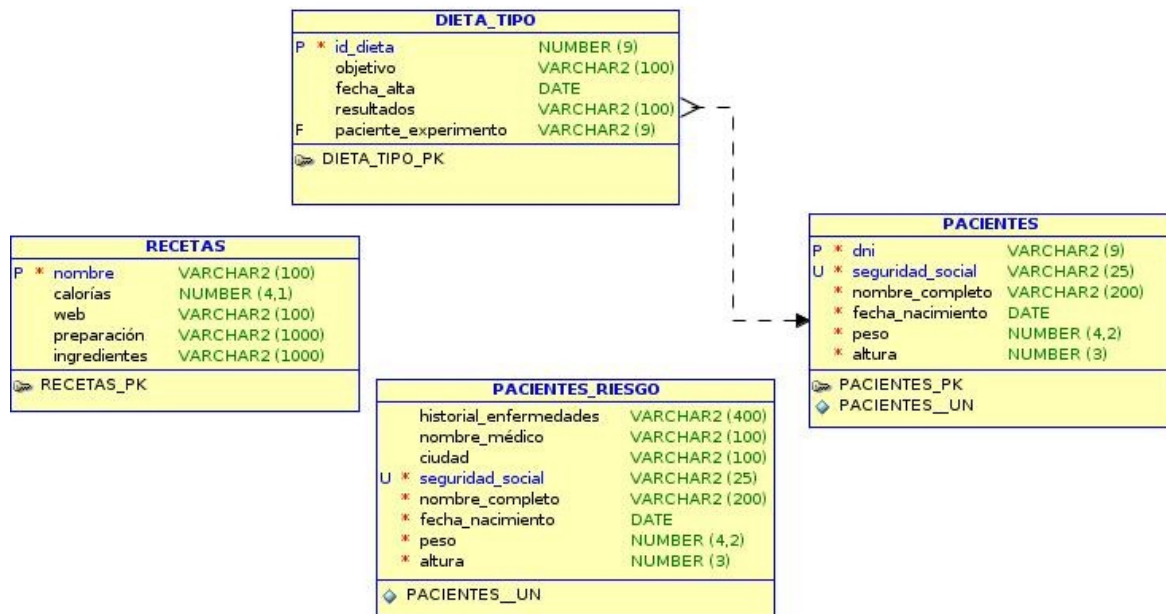
Esta información también la podemos obtener en SQL Developer haciendo click sobre la tabla correspondiente y eligiendo la pestaña "Constraints".

Observamos que por ejemplo la tabla Recetas tiene una Primary Key (clave primaria) formada por el atributo *nombre*.

La tabla Dieta\_Tipo tiene una Foreign Key (clave foránea) que referencia al atributo DNI (la clave primaria) de la tabla Pacientes, y que nos es obligatoria (puede tener el valor NULL). Esta es la única relación que tenemos actualmente en nuestra base de datos.

Nótese que la tabla Pacientes\_Riesgo no tiene clave primaria y además dispone de varios atributos que también se encuentran en la tabla Pacientes. Por lo que ya podemos ir apreciando que la tabla Pacientes\_Riesgo puede ser en realidad una subentidad de Pacientes, y efectivamente, así es como viene modelada en el nuevo esquema.

Ya tenemos una visión global de la base de datos, la siguiente imagen muestra un esquema relacional con las tablas y los atributos. Notar que no se trata de un diagrama Entidad/Relación tal y como lo hemos estudiado, sino que es simplemente un diagrama relacional que nos permite representar más información (como el tamaño de los campos de texto o la precisión en los valores numéricos) y que nos ayudará a tener más claros las modificaciones a realizar sobre la base de datos. Aunque obtener un diagrama Entidad/Relación a partir de este es tarea sencilla. A nosotros nos bastará con este.



Ahora estamos en condiciones de modificar nuestra base de datos. Podemos empezar modificando cualquier parte del esquema siempre que conservemos todos los datos.

Empezaremos por la tabla más simple de modificar, la tabla Recetas, ya que se mantienen todos los atributos en el nuevo modelo, así que no hay que agregar ni eliminar ninguno. Lo que si debemos diseñar son las dos relaciones que tiene esta tabla. La primera se trata de una relación reflexiva de muchos a muchos. Esta relación se puede leer como: *"una o varias recetas son similares a otra u otras recetas."*

Recordad que las relaciones muchos a muchos (m:m) se implementan creando una nueva tabla con dos atributos que referencian a las dos tablas relacionadas. La clave de esta nueva tabla será una clave compuesta por los dos atributos.

La relación m:m dará lugar a una nueva tabla: Similares. El hecho de que sea reflexiva significa que ambos atributos referencian a la misma tabla Recetas.

```
CREATE TABLE Similares (
    receta1 REFERENCES Recetas,
    receta2 REFERENCES Recetas,
    PRIMARY KEY (receta1, receta2)
);
```

La segunda relación entre la tabla Recetas y la tabla Dieta\_Tipo es otra

relación m:m, lo que dará lugar a una nueva tabla tal y como acabamos de ver con la relación anterior. La diferencia es que esta relación es obligatoria por ambas partes. De esta manera: *"una o varias recetas deben estar incluidas en una dieta\_tipo, y una o varias dietas\_tipos deben incluir una o varias recetas"*. Aunque como veremos a continuación, esta diferencia no se aprecia en el modelo relacional resultante.

Llamaremos a esta nueva tabla Incluir.

```
CREATE TABLE Incluir (  
    recetas REFERENCES Recetas,  
    dietas REFERENCES Dieta_Tipo,  
    PRIMARY KEY (recetas, dietas)  
);
```

Notar que la obligatoriedad de la relación m:m que modelamos no es distinguible. ¿Cuál es la diferencia con la relación anterior que era opcional?. No existe diferencia, porque el aspecto de la obligatoriedad de las relaciones sólo es posible plasmarlo cuando existe una restricción de clave foránea sobre la que descansa la relación. Esto sólo se da en las relaciones M:1 o 1:1. Por ello, este aspecto se dejará para su posterior implementación con Disparadores (Triggers).

El siguiente paso será modificar la relación "experimental" entre las tablas Dieta\_Tipo y Pacientes. Esta relación ya existe, pero es m:1 y opcional. En el nuevo diseño queremos que sea 1:1 y obligatoria.

Antes de poder modificar esta relación, debemos tener en cuenta que el campo *dni* de los pacientes es una cadena de caracteres, y en el nuevo diseño queremos que sea un campo numérico de 9 dígitos.

Por lo que tenemos que realizar este cambio en la tabla Pacientes. Pero no nos dejará hacer el cambio si existen referencias a esta tabla.

Así que lo primero será eliminar todas las referencias a la tabla Pacientes; en este caso sólo se trata de la propia relación "experimental".

```
ALTER TABLE Dieta_Tipo  
    DROP CONSTRAINT SYS_C004372;
```

El nombre de la restricción puede variar.

Para eliminar una restricción de una tabla usamos la sentencia:

```
ALTER TABLE tabla  
    DROP CONSTRAINT nombre_restricción;
```

Ahora si podemos cambiar el tipo del campo *dni*, no sin antes quitar la restricción de clave primaria que tiene.

```
ALTER TABLE Pacientes  
    DROP CONSTRAINT SYS_C004363;
```

```
ALTER TABLE Pacientes
    MODIFY DNI NUMBER(9);
```

Modificación de columnas: ALTER TABLE tabla MODIFY columna atributos;
---

Volvemos a poner dni como clave primaria:

```
ALTER TABLE Pacientes
    ADD CONSTRAINT Pacientes_PK PRIMARY KEY (dni)
    ENABLE;
```

Por último, cambiamos también el tipo del campo *paciente\_experimento* de la tabla Dieta\_Tipo, ya que como nos indican, todas las claves foráneas han de ser simples y de tipo NUMBER.

```
ALTER TABLE Dieta_Tipo
    MODIFY paciente_experimento NUMBER(9);
```

```
ALTER TABLE Dieta_Tipo
    ADD CONSTRAINT Dieta_Tipo_Pacientes_FK1
        FOREIGN KEY (PACIENTE_EXPERIMENTO)
            REFERENCES PACIENTES (DNI)
    ENABLE;
```

Con la siguiente sentencia cambiamos la obligatoriedad de la relación:

```
ALTER TABLE Dieta_Tipo
    MODIFY paciente_experimento NUMBER(9) NOT NULL;
```

y a continuación cambiamos el grado:

```
ALTER TABLE Dieta_Tipo
    ADD CONSTRAINT Dieta_Tipo_UK1 UNIQUE (paciente_experimento)
    ENABLE;
```

Pasemos ahora a modificar la tabla Pacientes\_Riesgo. Como dijimos antes, esta tabla tiene muchos atributos idénticos a los de la tabla Pacientes, por lo que será una subentidad de dicha tabla (relación “es-un”).

La tabla Pacientes no cambia en el nuevo esquema, salvo el tipo del campo *dni* que acabamos de modificar.

Para transformar la tabla Pacientes\_Riesgo en una subtabla de la tabla Pacientes, creamos una nueva columna que referencie a la clave primaria de Pacientes. Esta nueva columna a su vez será la clave primaria (clave prestada) de Pacientes\_Riesgo.

Podemos añadir columnas a una tabla existente con la sentencia:

```
ALTER TABLE tabla
    ADD (nombre_columna1 tipo [restricciones],
        nombre_columna2 tipo [restricciones],
        ...);
```

```
ALTER TABLE Pacientes_Riesgo
    ADD (dni NUMBER(9) REFERENCES PACIENTES PRIMARY KEY);
```

Eliminamos la restricción de unicidad del campo *seguridad\_social* para poder eliminar a continuación dicha columna de la tabla Pacientes\_Riesgo.

```
ALTER TABLE Pacientes_Riesgo
    DROP CONSTRAINT SYS_C004370;
```

```
ALTER TABLE Pacientes_Riesgo
    DROP COLUMN seguridad_social;
```

Igualmente eliminamos las columnas repetidas o innecesarias.

```
ALTER TABLE Pacientes_Riesgo
    DROP COLUMN nombre_completo;
```

```
ALTER TABLE Pacientes_Riesgo
    DROP COLUMN fecha_nacimiento;
```

```
ALTER TABLE Pacientes_Riesgo
    DROP COLUMN peso;
```

```
ALTER TABLE Pacientes_Riesgo
    DROP COLUMN altura;
```

Para terminar de modificar nuestro esquema solo nos falta crear la tabla Historial; Historial es una entidad débil de Dieta\_Tipo y Pacientes\_Riesgo, por lo que tomará prestada las claves de ambas tablas.

```
CREATE TABLE Historial (
    fecha_visita DATE,
    dieta REFERENCES DIETA_TIPO,
    paciente REFERENCES PACIENTES,
    PRIMARY KEY (fecha_visita, paciente, dieta),
    peso_actual NUMBER(4,2),
    evaluacion VARCHAR2(100)
);
```