Tablas del Gimnasio Normalizadas

Primero modelaremos las entidades y después las relaciones. Para el diseño y la normalización no solo se tomó en cuenta el diagrama adjunto, si no además las correciones realizadas a lo largo de las prácticas y los comentarios hechos respecto a su diseño

Persona

Esta nueva tabla se agrega para la práctica 5, pero nos servirá para evitar la redundancia de datos agregada pasada a a primera formal normal se ve como:

---

id_Persona	Nombre	Ap_Paterno	Ap_Materno	Email	Fec_Nac	Sexo	Teléfono

De esta manera su llave primaria id_Persona va a tener las siguientes dependencias funcionales:

- id_Persona → Nombre
- id_Persona → Ap_Paterno
- id_Persona →Ap_Materno
- id_Persona → Email
- id_Persona → Fec_Nacimiento
- id_Persona → Sexo
- id_Persona → Teléfono

Una vez que todas las dependencias funcionales tienen como determinante a la llave primaria vemos que está en forma normal Boyce-Codd, esto nos va a servir para poder identificar a cualquier persona relacionada con el gimnasio.

Socio

Es necesario cambiar esta tabla en más de una forma, pero para evitar que sean muchas tablas las dibujadas solo mostraremos el resultado final.

Primero hacemos que todos los atributos de la tabla sean atómicos, agregando el id_Socio (su llave primaria) podemos hacer que todos los demás atributos dependan de este, así podemos eliminar dependencias transitivas, como en el caso de que Nombre sea el determinante de Ap_Paterno y Ap_Materno, en este caso no podría haber 2 personas con el mismo nombre y apellidos, si los hacemos dependientes, cuando dependen de la llave primaria esto ya no nos importa.

Además de que cambiamos el Sistema de puntos y lo agregamos directamente a esta tabla dado que un socio es el único que puede acumular puntos.

Una vez hecho esto la tabla nos quedará de la siguiente manera:

id_Socio	Nombre	Ap_Paterno	Ap_Materno	Email	Fec_Nacimiento

Sexo	Teléfono	Puntos	Persona_Contacto

Como Persona_Contacto puede almacenar tanto el nombre, como el número o correo por el que se le desea contactar lo agregamos como otra tabla, el atributo Person_Contacto se volverá un nombre y pasará a ser llave foránea.

Para pasarlo a segunda forma normal podemos ver que existen las siguientes dependencias funcionales:

- id_Socio → Nombre
- id_Socio → Ap_Paterno (Si dependiera del nombre no podrían existir dos Socios con el mismo nombre).

- id_Socio → Ap_Materno
- id_Socio → Email
- id_Socio → Fec_Nacimiento
- id_Socio → Sexo
- id_Socio → Teléfono
- id_Socio → Puntos
- id_Socio → Persona_Contacto

Así id_Persona pasa a ser la llave primaria y es el determinante de todos los demás atributos.

Persona_Contacto

Nombre	**Ap_Paterno**	**Ap_Materno**	**Contacto**

Se agrega una nueva tabla y queda en Forma Normal de Boyce-Codd.

Tiene

Esa tabla no recibe modificaciones debido a que ya está en forma normal de Boyce-Codd. Los atributos id_Membresia e id_Socio son parte de una llave compuesta y no tienen dependencias funcionales.

Tiene

id_Membresia	**id_Socio**

Cliente

A su vez conforme mejoramos el programa notamos que también es necesario mejorar la tabla Cliente, agregando datos bastante parecidos a los de un Socio, aunque esto puede ser a gusto del cliente, ya que solo se almacenan 'por si es necesario', ya que se pueden prescindir de ellos.

id_Cliente	**Nombre**	**Ap_Paterno**	**Ap_Materno**	**Edad**	**Email**	**Sexo**	**Teléfono**

Para este caso no nos interesa saber la Persona de Contacto ni sus Puntos, dado que no es Socio ni está debidamente registrados, en versiones anteriores solo se llevaba una id para saber el numero total de clientes, por lo cual reitero que estos datos pueden ser prescindibles. En dado caso de que los ocupemos nos resultarán las siguientes dependencias funcionales:

- id_Cliente → Nombre
- id_Cliente → Ap_Paterno
- id_Cliente → Ap_Materno
- id_Cliente → Edad
- id_Cliente → Email
- id_Cliente → Sexo
- id_Cliente → Teléfono

Entrenador

Una vez que se fué mejorando la tabla se creo una super clase para modelar a las personas relacionadas con el gimnasio, por eso existen varias tablas con atributos similares, salvo que nos interesa también saber la direccion del entrenador, pero tomando como base el ejemplo dado para esta práctica, convertiremos su direccion en atributos atómicos al momento en el que agregamos mas atributos a la tabla.

id_Empleado	**Nombre**	**Ap_P**	**Ap_M**	**Email**	**Edad**	**Sexo**	**Telefono**	**id_Direccion**

Manejamos las dirección como un id que es una llave foránea y nos quedan las siguientes dependencias funcionales:

- id_Empleado → Nombre
- id_Empleado → Ap_P
- id_Empleado → Ap_M
- id_Empleado → Email
- id_Empleado → Edad
- id_Empleado → Sexo
- id_Empleado → Telefono
- id_Empleado → id_Direccion

Membresía

Membresía originalmente solo contenía el valor de su id que se pasaba como llava foránea de la especialización de los tipos de membresia y su costo, pero para darle mas sentido a esta entidad se le asigna tambien el id del socio como una llave foránea, que a su vez se utiliza para crear una llave compuesta.

id_Membresia	**id_Socio**	**Costo**

Nos queda la dependencia funcional id_Membresia → Costo

Membresía Básica

Contiene la información respecto a los beneficios de la membresía básica y su identificador único.

id_Membresia	**Areas_Entrenamiento**	**Regaderas**

Nos quedan las dos dependencias funcionales:

- id_Membresia → Areas_Entrenamiento
- id_Membresia → Regaderas

Sus beneficios de todos los tipos de membresias depeden del id porque este a su vez nos define el tipo de membresia que puede ser.

Membresía Plus

id_Membresia	**Areas_Entrenamiento**	**Regaderas**	**Casillero**	**Entrenador**

Con las depdendencias funcionales:

- id_Membresia → Areas_Entrenamiento
- id_Membresia → Regaderas
- id_Membresia → Casillero

- id_Membresia → Entrenador

Membresía Premium

id_Membresia	Areas_Entrenamiento	Regaderas	Casillero	Entrenador	Sauna	Especialista	Puntos

Esta vez nos importa almacenar tambien si es valido el sistema de puntos o no, pero estos se guardan en la tabla Socio que interactua con las clases y sus puntos, nos resultan las siguientes dependendias funcionales:

- id_Membresia → Areas_Entrenamiento
- id_Membresia → Regaderas
- id_Membresia → Casillero
- id_Membresia → Entrenador
- id_Membresia → Sauna
- id_Membresia → Especialista
- id_Membresia → Puntos

Clase

Para las clases solo es necesario agregar otra tabla para representar los días que son impartidos las clases, y estos se relacionan mediante una llave foránea.

Nombre_Clase	Instructor	Costo	id_Dias	Hora_Inicio	Hora_Fin	Puntos

Todas las dependencias quedan en base a su llave primaria:

- Nombre_Clase → Instructor
- Nombre_Clase → Costo
- Nombre_Clase → id_Dias
- Nombre_Clase → Hora_Inicio
- Nombre_Clase → Hora_Fin
- Nombre_Clase → Puntos

Dias_Clase

id_Dias	Lunes	Martes	Miercoles	Jueves	Viernes	Sabado	Domingo

Area

Agregamos una nueva entidad para modelar las áreas disponibles del gimnasio

id_Area	Nombre_Area	Visitas	Es_Permitida

Producto

Como todos los atributos ya eran atómicos solo nos encargamos de eliminar sus dependencias y dirigirlas a su llave primaria compuesta:

Nombre_Producto	Presentación	Descripción	Existencias	Marca

Tener

La relación Tener nos muestra como un Socio puede tener una membresía, pero por el diseño de la base de datos podemos prescindir de una tabla, pero la relación ejemplificandola en el diagrama seguiría siendo útil.

Tomar_Clase_Area

Se agrega una nueva relación para las clases y las áreas donde son impartidas, pero las relaciones no llevan atributos

Tomar_Clase_Socio

Nombre_Clase	Entrenador	id_Socio

No tiene dependencias funcionales

Tomar_Clase_Cliente

Nombre_Clase	Entrenador	id_Cliente

No tiene dependencias funcionales

Comprar_Socio

Nombre_Producto	Presentación	id_Socio

No tiene dependencias funcionales

Comprar_Cliente

Nombre_Producto	Presentación	id_Cliente

No tiene dependencias funcionales.

Impartir_Clase

Nombre_Clase	Entrenador	id_Entrenador

No tiene dependencias funcionales.