{desafío} latam_

Modelo entidad-relación (Parte I)

Modelo conceptual

Competencias

- Identificar una entidad
- Identificar los atributos
- Identificar las relaciones entre las entidades
- Crear modelos conceptuales

Introducción

Hasta el momento, nuestro conocimiento sobre bases de datos es estrictamente procedimental: sabemos cómo implementar tablas con columnas en un motor de bases de datos, así como generar instrucciones de modificación de tablas.

El siguiente paso es saber modelar los problemas, o sea construir el modelo de datos e implementarlo en base a un problema.

La modelación de bases de datos es un proceso para definir e implementar requerimientos del usuario/empresa dentro del contexto de los sistemas de información disponibles en una organización, o sea partir de uno o mas requerimientos construiremos una base de datos que nos permita almacenar y recuperar información relevante.

Elementos básicos de una base de datos

Para entender el proceso de modelado de datos, debemos definir los elementos básicos de una base de datos. Dependiendo de la problemática a la que nos enfrentemos, deberemos reconocer sobre qué estamos tratando. Este elemento básico de un modelo de bases de datos se reconoce como **entidad**, que corresponde a un objeto del mundo real.

Ahora, si bien podemos definir una entidad, esta posee características, las cuales denominaremos **atributos** que son propias de la entidad.

Ahora nos surge la pregunta: ¿Qué puede ser considerado como una entidad o atributo?

- La definición dependerá de los requerimientos y los casos de uso.

Ejemplo

Por ejemplo, si estamos realizando una base de datos de clientes para una automotriz, probablemente queramos considerar a cada Cliente como entidad con atributos como: nombre, rut, auto comprado, año del auto, y a su vez, si la automotriz maneja un stock de autos disponibles para la venta, tendremos otra entidad de Auto, la cual definiremos caracteristicas del auto, teles como color, motor, numero de puertas, entre otros.

La importancia de la modelación

Los ejemplos mencionados buscan manifestar la importancia de la modelación, es por eso que antes de comenzar a ingresar y preprocesar nuestros datos para futuras aplicaciones, debemos modelar la naturaleza del problema y cómo las entidades se relacionan entre sí. El modelo es el paso previo a la construcción de la base de datos.

Es importante que el modelo sea el correcto, dado que puede afectar la fidelidad de los datos y complicar el desarrollo de software que lo requiera. Para efectos prácticos de esta lectura, trabajaremos con bases de datos relacionales (bases de datos representadas en tablas).

¿Cómo hacemos un modelo de datos?

El proceso de modelado de datos es una serie de 3 pasos que permiten identificar los elementos que son importantes para la creación de la base de datos, a partir de la información los datos que deberemos almacenar, hasta crear diagramas que represente completamente nuestra base de datos.

1. Modelo Conceptual
2. Modelo Lógico
3. Modelo Físico

Modelo Conceptual

Previo a empezar a crear nuestros modelos, deberemos pasar por una etapa de investigar y analizar cuales serán los datos que almacenaremos en nuestra base de datos. A esta etapa se le denomina **toma de requerimientos**, en la cual se entabla una conversacion con clientes para saber que es lo que ellos necesitan.

Para los ejercicios propuestos, se entregarán los requerimientos, por lo que no deberemos hablar con el cliente.

A partir de esta conversación, identificaremos las entidades y atributos y seguiremos siempre los mismos pasos:

1. Identificar las entidades
2. Agrupar entidades con sus atributos
3. Nombrar las relaciones entre las entidades en caso de existir.

Luego crearemos un diagrama que represente las entidades (cuadrados), sus atributos (círculos) y relaciones (rombos). Este diagrama **debe ser entendible por todos los clientes, independiente que sean programadores**.

Imagen 1. Diagrama que representa entidades, atributos y relaciones.

Ejercicio 1:

Una empresa vende productos a varios clientes. Se necesita conocer los datos personales de los clientes (nombre, apellido, dni y dirección) y llevar el registro de los productos.

Solución

Paso 1: Identificar las entidades

En este caso podemos identificar 3 candidatos.

- Empresa: en este ejemplo no nos piden registrar datos de la empresa, solo se especifica que es una empresa.
- Cliente: nos solicitan almacenar datos de nombre, apellido, dni y dirección.
- Producto: Nos piden registro de los productos.

De las entidades anteriores, como no existe requerimientos con respecto a la empresa, podemos descartarla como entidad. Por lo que sólo definimos dos entidades para este problema, Cliente y Producto.

Paso 2: Identificar los atributos

```
Cliente(nombre, apellido, dni, direccion)
Producto()
```

Paso 3: Nombrar la relación

No hay relaciones identificables, así que las omitiremos por ahora.

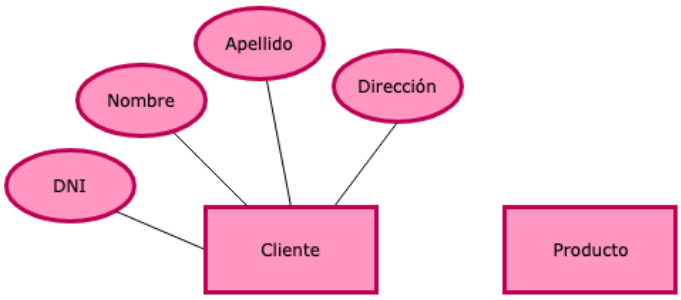

Imagen 2.Solución Ejercicio 1.

Conversación con el cliente

Al mostrarle nuestro diagrama a nuestro cliente el nos dirá que está muy bien pero que no refleja algo que nos dijo y que no anotamos en una primera etapa:

Un cliente puede comprar varios productos y un producto puede ser comprado por varios clientes y nos pide que actualicemos nuestros diagrama para reflejar esto.

Ademas nuestro cliente aprovecha la oportunidad de decirnos que los productos tienen un código.

Paso 1: Identificar las entidades

Entidades: Cliente y Producto

Paso 2: Agrupar entidades con sus atributos

Cliente(nombre, apellido, dni, direccion)
Producto(código)

Paso 3: Nombrar la relación

Relacionamos la entidad Cliente con Producto, mediante una compra.

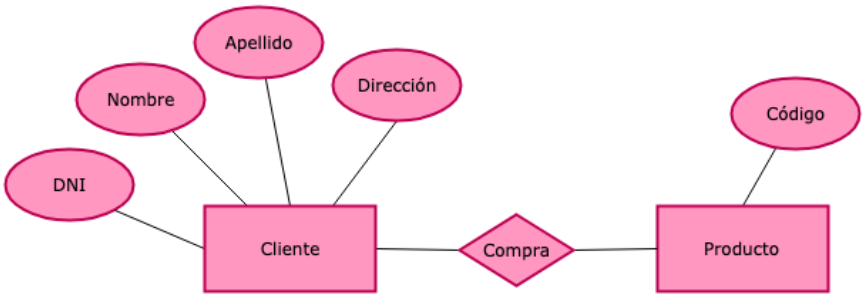

Imagen 3. Relacionar entidad Cliente-Producto.

Ejercicio 2:

Un cliente necesita construir un plataforma para llevar registro de los usuarios que se inscriben en un formulario de una página web, en el formulario llenan email, nombre y teléfono. Por otro lado el sistema tiene que ser capaz de llevar registro de cada llamado telefónico que se la ha hecho al cliente y un reporte asociado a ese llamado y que persona realizó el llamado.

Solución

Paso 1: Identificar las entidades

- Entidades: Usuario y reporte.
  Existe otro candidato a entidad, quien realiza la llamada, sin embargo no hay información de sus posibles atributos. Por ahora no lo agregaremos.

Paso 2: Agrupar entidades con sus atributos

```
Usuario (email, nombre, teléfono)
Registro (responsable, reporte)
```

Paso 3: Nombrar la relación

En este caso la relación sucede de una llamada telefónica y la podemos llamar así:

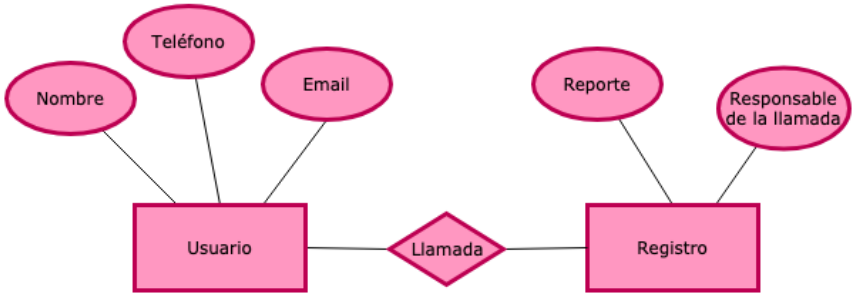

Imagen 4. Solución Ejercicio 2.

Ejercicio propuesto:

Un banco que calcula riesgo de clientes necesita llevar un registro de personas con sus nombres y números telefónicos, el país donde nacieron, sus vehículos con sus patentes, marca, modelo y año y sus propiedades con su dirección, metros cuadrados y metros cuadrados construidos. Nos piden hacer el modelo conceptual.

Resumiendo

El modelo conceptual nos ayuda a describir y comunicar el contenido de nuestra base de datos a un nivel alto, o sea, describirlo de forma independiente de las estructuras donde se guardarán nuestros datos. Para desarrollarlo necesitamos seguir estos 3 pasos principales:

1. Definir las entidades.
2. Agrupar las entidades con sus atributos.
3. Definir las relaciones entre las entidades.

Finalmente cabe mencionar que existen otros tipos de diagramas para realizar modelos conceptuales, el que estamos ocupando es uno de los más famosos fue diseñado por Peter Chen en 1976.

Estos diagramas pueden ser considerados innecesarios por personas con poca experiencia en el área de desarrollo, pero los modelos, ya sean conceptuales, lógicos y/o físicos son muy útiles para comunicar distintas necesidades y coordinar esfuerzos a la hora de trabajo. Es normal verlos dibujados en pizarras o impresos y colgados en las paredes de las oficinas de trabajo.

Cardinalidad

Competencias

- Identificar relaciones
- Identificar cardinalidades
- identificar identificadores

Introducción

Con el capítulo anterior ya aprendimos a identificar bajo una problemática cuáles son las entidades y sus atributos. Además se definió una relación entre estas. Las relaciones entre entidades pueden ser de variadas formas, por lo que en este capítulo aprenderemos a reconocerlas.

En el problema donde tenemos la entidad Cliente y Producto, relacionadas entre si con la relación `comprar`, esta relación nos dice que ¿un cliente compra un producto, o un producto es comprado por un cliente, o muchos productos son comprados por muchos cliente o muchos clientes compran un producto?

En el ámbito de la programación, el computador no sabe que significa Cliente o Producto, por lo que es de gran importancia que nosotros podamos definir de manera clara esta relación para que al momento de implementar nuestra base de datos no se generen incongruencias.

Cardinalidad

La cardinalidad en una base de datos indica la cantidad de veces que esta entidad puede relacionarse con otra entidad. Se define de la manera:

```
mínimo:máximo
```

Donde puede ser cualquier combinación de valores arbitrarios.

A continuación veremos las 3 combinaciones claves que definen la naturaleza de la relación:

1:1

Donde cada registro **debe** estar asociado a otro registro:

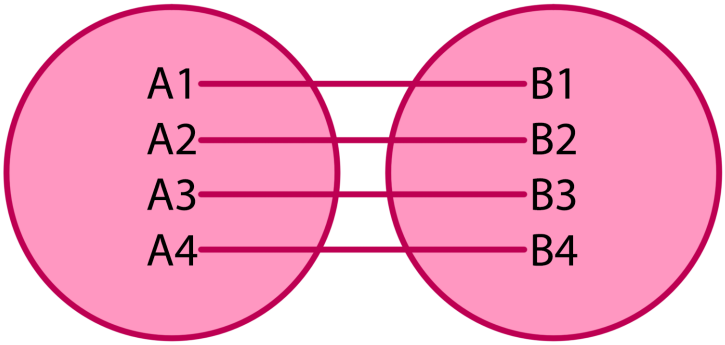

Imagen 5. Cardinalidad 1:1

0:1

A veces, incluso de uso mucho más frecuente, son las cardinalidades de 0:1, donde uno de los registros puede estar asociado a otro registro.

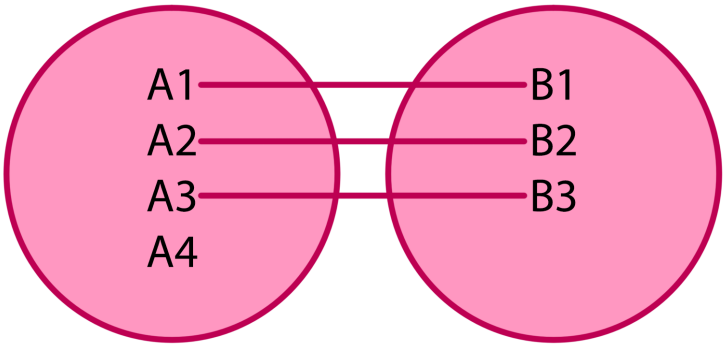

Imagen 6. Cardinalidad 0:1

Un ejemplo de esta cardinalidad sería un modelo con personas y sus cónyuges. Una persona **puede** tener una relación con otra, entonces esta sería una cardinalidad de 0:1

1:N

Donde cada entidad puede estar asociada a cualquier número de otras entidades:

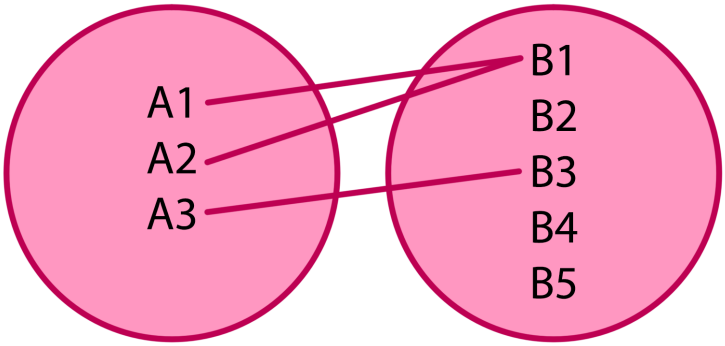

Imagen 7. Cardinalidad 1:N

Este es el tipo de cardinalidad más frecuentemente utilizada. Un ejemplo común de esta cardinalidad es el uso de **categoría**.

Ejemplos

- Tenemos artículos y cada uno puede tener una categoría; por lo mismo en categoría hay diversos artículos.
- Personas y sus países de orígenes, una persona solo puede tener un país de origen, cada país tiene asociado N personas.
  Otras variantes típicas son **0:N** y fijando un N por ejemplo **1:4**

N:N

Este tipo de cardinalidad representa situaciones donde cada entidad puede estar asociadas a múltiples entidades de otro tipo y al mismo tiempo una entidad del otro grupo:

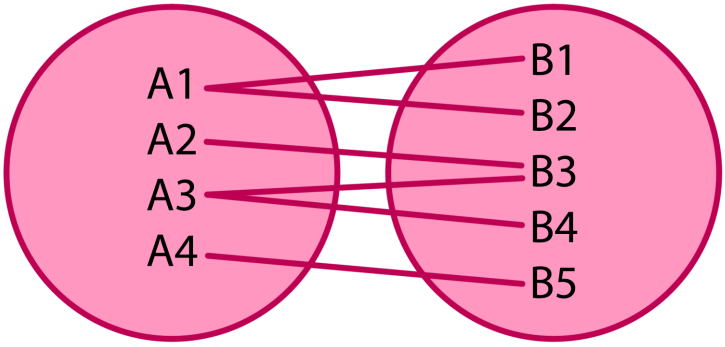

Imagen 8. Cardinalidad N:N

Un ejemplo típico de esta cardinalidad se da entre artículos y tags. Un artículo puede tener múltiples tag y cada tag puede estar asociado a múltiples artículos.
Un detalle importante como ambas cardinalidades pueden ser distintas a este tipo de cardinalidad a veces se denomina **N:M**.

Análisis de cardinalidades

- Si decimos que la cardinalidad de una relación entre clientes y cuentas bancarias es de 1 a N, esto quiere decir que un cliente puede tener varias cuentas bancarias y que una cuenta bancaria solo puede pertenecer a un cliente.
- Si decimos que es de 1 a 1, estamos diciendo que un cliente solo puede tener una cuenta bancaria y esa cuenta bancaria le debe pertenecer a un único cliente.
- Si volvemos a nuestro ejercicio donde guardábamos el registro de llamadas telefónicas de diversos clientes, veremos que a cada cliente podemos hacer varios llamadas que resulten en registros, y un registro de llamada telefónica solo puede pertenecer a un cliente.

Identificadores (ID)

El identificador es un **atributo** o **conjunto de atributos** que determina de modo único la ocurrencia de una entidad.

Analicemos algunos casos:

Trabajadores en una oficina

Tenemos un registro de trabajadores de una oficina, de los cuales tenemos nombre, email corporativo y fecha de nacimiento ¿Cuál debería ser el identificador?

Imagen 9. Ejemplo Identificadores.

El nombre y la fecha de nacimiento pueden repetirse con otros trabajadores, por lo que email corporativo es el mejor candidato.

¿Qué pasaría si por algún motivo dos personas ocupan el mismo email, por ejemplo: finanzas@corporación.com?

En ese caso, otra posibilidad sería utilizar un **atributo compuesto**, por ejemplo: mail más nombre, pero en una empresa grande podría darse que eso tampoco sea único.

Podríamos agregar la fecha de nacimiento o incluso podríamos crear un atributo nuevo que sea un código trabajador para identificar únicamente a cada trabajador.

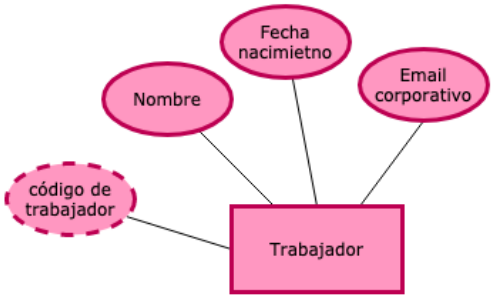

Imagen 10. Agregar atributo.

Suele suceder que en el proceso de modelamiento aparecen estas necesidades para las empresas y proyectos y por eso se debe diseñar esta etapa con cuidado. Eventualmente los identificadores serán nuestras **claves primarias**.

Por ahora diremos que los email corporativos son únicos y cada persona en la empresa tiene el suyo propio.

Registro telefónico

En el caso de los registros no tenemos un buen candidato.

- El reporte es un texto largo que no tenemos seguridad de que no sean iguales entre ellos.
- Resulta que una persona puede llamar múltiples veces.
- La combinación de estos tampoco hay seguridad de que sea única.

Tenemos dos opciones interesantes:

- agregar la hora de la llamada
- agregar un número para esta llamada

Debemos consultar con nuestros clientes antes de agregar entidades y atributos.

Por el bien del ejercicio, supongamos que nuestro cliente nos dio el visto bueno para agregar un identificador a la tabla de registros de llamados. De esta forma, subrayando los identificadores nuestro modelo conceptual quedaría de la siguiente forma:

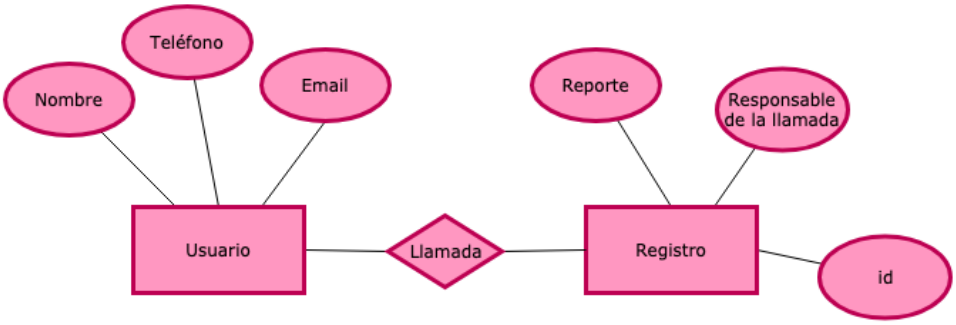

Imagen 11. Ejemplo identificadores.

¿Por qué son importantes los identificadores?

El concepto más importante asociado a bases de datos es el de **integridad**: que nuestros registros sean correctos y completos. Sin identificadores no tendríamos certeza que estamos modificando el dato correcto. Por ejemplo, si cambiamos el teléfono de un usuario con nombre Juan, pero hay cincuenta usuarios con ese nombre, cambiaríamos los 50 teléfonos perdiendo nuestros datos anteriores.

Los identificadores nos permiten a encontrar de forma única nuestros datos. Sobre estos construiremos nuestras claves primarias mas adelante en el modelado lógico.

Un detalle importante: es perfectamente posible tener una tabla sin identificadores, pero esto puede ser problemático por los motivos anteriormente expuestos. Un buen modelado conceptual requiere de definir identificadores para cada entidad.

El modelo lógico

Competencias

- Generar modelos lógicos
- Identificar la propagación de la clave primaria

Introducción

Mientras el modelo conceptual tiene como objetivo describir la información de forma independiente del motor de base de datos, el modelo lógico describe el modelo de datos en términos del tipo de motor. En esta unidad, trabajaremos con un motor relacional.

Modelo conceptual a modelo lógico

Para transformar un modelo conceptual en lógico lo que tenemos que hacer es:

1. Transformar todas las entidades en tablas, agregar los atributos como columnas de la tabla.
2. Transformar todas las relaciones del tipo N:N en tablas nuevas.
3. Propagar la clave primaria de las tablas en las relaciones 1:N desde el lado de las 1 al lado de las N.

Realicemos el ejemplo con nuestro modelo conceptual:

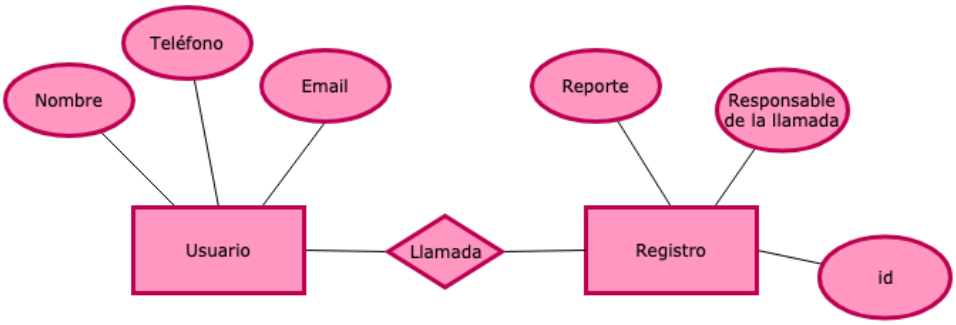

Imagen 12. Ejemplo de Modelo Conceptual.

1. Tenemos 2 entidades por lo que quedarán dos tablas.
2. No tenemos relaciones N a N así que no se crearán tablas nuevas.
3. Propagamos la clave primaria desde el lado de las 1 a las N

Para realizar el diagrama del modelo lógico hay distintos tipos de notaciones.

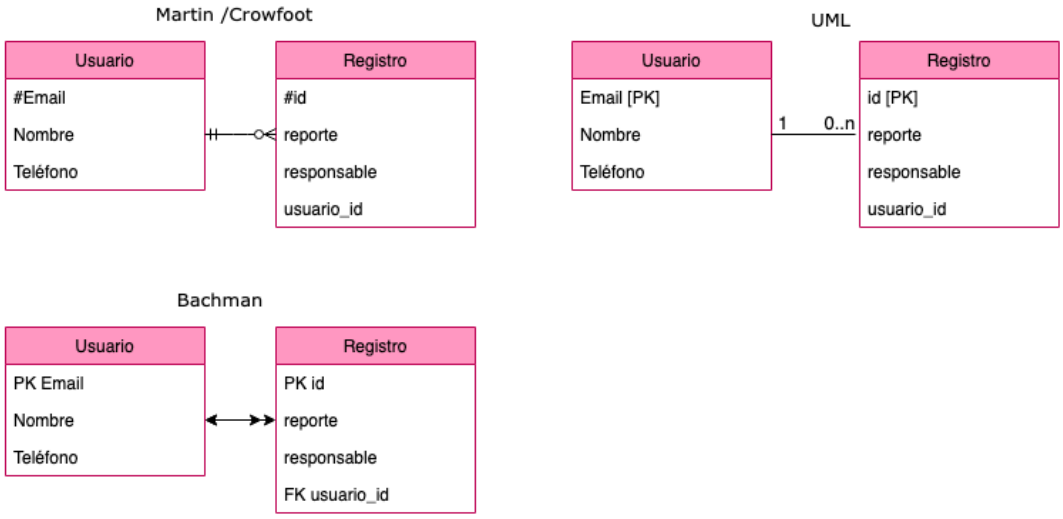

Imagen 13. Diagrama Modelo Lógico.

La diferencia entre los tipos de notaciones principalmente cambian la forma de representar la cardinalidad y como anotar la clave primaria, foránea y atributos obligatorios.

Existen estándares para representar cada uno de los diagramas, sin embargo, es poco habitual que los programas para realizar estos diagramas se rijan 100% en función de estos estándares. A lo largo de esta unidad, utilizaremos la notación de UML. En esta la cardinalidad de la relación se anota como `mínimo..máximo`, un único número indica qué la cantidad es obligatoria. El nombre de la relación va indicado sobre la línea para indicar la relación de izquierda a derecha y bajo la línea para indicarla de derecha a izquierda.

¿Cómo recordar la propagación de clave?

En lugar de memorizar reglas sobre cuál tabla propaga a cuál, es mejor visualizar cómo sería el resultado.
Supongamos que tenemos la tabla de artículos y la tabla de comentarios. Un artículo puede tener muchos comentario, y un comentario le pertenece a un artículo ¿En qué lado va la clave foránea?

Artículo

id	url	título	contenido
1	/articulo1	Artículo 1	Lorem ipsum ..
2	/articulo2	Artículo 2	Lorem ipsum ..
3	/articulo3	Artículo 3	Lorem ipsum ..
4	/articulo4	Artículo 4	Lorem ipsum ..

Comentario

id	contenido
1	Primer comentario
2	Segundo comentario
3	Tercer comentario
4	Cuarto comentario

Imagen 14. Ejemplo de artículos y comentarios.

Hay solo dos opciones:

- en el lado de artículo.
- en el de comentarios.

Si ponemos la clave de artículo en los comentarios quedaría:

Artículo

id	url	título	contenido
1	/articulo1	Artículo 1	Lorem ipsum ..
2	/articulo2	Artículo 2	Lorem ipsum ..
3	/articulo3	Artículo 3	Lorem ipsum ..
4	/articulo4	Artículo 4	Lorem ipsum ..

Comentario

id	contenido	articulo_id
1	Primer comentario	1
2	Segundo comentario	1
3	Tercer comentario	2
4	Cuarto comentario	1

Imagen 15. Clave del artículo en los comentarios.

En cambio si lo hiciéramos al revés obtendríamos algo así:

Artículo

id	url	título	contenido	contenido1_id	contenido1_id	contenido1_id	contenido1_id
1	/articulo1	Artículo 1	Lorem ipsum ..	1	2		4
2	/articulo2	Artículo 2	Lorem ipsum ..			3	
3	/articulo3	Artículo 3	Lorem ipsum ..				
4	/articulo4	Artículo 4	Lorem ipsum ..				

Comentario

id	contenido	articulo_id
1	Primer comentar	1
2	Segundo coment	1
3	Tercer comentari	2
4	Cuarto comentar	1

Imagen 16. Clave del comentario en los artículos.

Aquí necesitaríamos infinitas columnas para poder guardar las asociaciones, por lo mismo la única opción viable es agregar la clave foránea en el lado de los N. Si olvidamos la regla siempre podemos hacer este ejercicio de visualización.

Modelo Físico

Competencias

- Crear modelo físico
- Crear tablas en la base a partir del modelo físico.

Introducción

El modelo físico es el responsable de representar cómo se construirá finalmente el modelo en nuestra base de datos. Incluye la estructuras de las tablas, definiendo cada uno de sus atributos y tipo de dato para cada atributo, las restricciones de las columnas, las claves primarias, claves foráneas y las relaciones entre las tablas.

En síntesis, el modelo físico es la versión final del modelo.

Traspasando el modelo físico a SQL

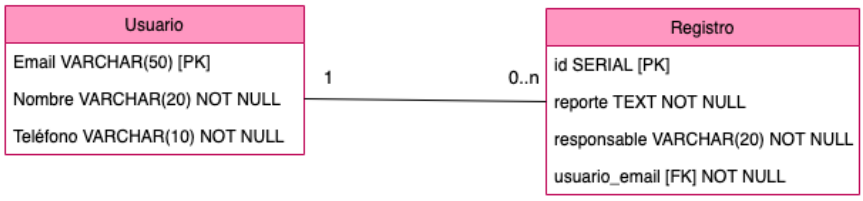

Imagen 17. Modelo Físico a SQL.

Con el modelo físico podemos pasar construir nuestras tablas y agregar datos.

```
CREATE TABLE usuario(
email VARCHAR(50),
nombre VARCHAR(20) NOT NULL,
telefono VARCHAR(15) NOT NULL,
PRIMARY KEY (email)
);
CREATE TABLE registro(
id CHAR(10),
reporte VARCHAR2(1000) NOT NULL,
responsable VARCHAR(50) NOT NULL,
usuario_email VARCHAR(50) REFERENCES usuario (email),
PRIMARY KEY (id)
);
```

Al traspasar los datos a SQL nos damos cuenta que nuestro modelo físico tiene toda la información necesaria.
Existen programas que nos permiten generar SQL a partir de modelos físicos y otros que nos permiten generar los diagramas a partir de SQL.

Finalmente agregamos datos para probar nuestro modelo.

```
INSERT INTO usuario(email, nombre, telefono) VALUES ('usuario1@gmail.com',
'Juan', '12345678');
INSERT INTO usuario(email, nombre, telefono) VALUES ('usuario2@gmail.com',
'Francisca', '12345679');
INSERT INTO registro (id, reporte, responsable, usuario_email) values (1, 'El usuario
presenta problemas para realizar el pago online', 'Javiera',
'usuario1@gmail.com');
INSERT INTO registro (id, reporte, responsable, usuario_email) values (2, 'El usuario
presenta problemas para ingresar a la plataforma', 'Javiera',
'usuario2@gmail.com');
```

Luego si queremos seleccionar todos los registros con sus usuarios realizaremos un join entre ambas tablas.

```
select *
from registro
join usuario on usuario.email = registro.usuario_email;
```

Caso con relaciones N a N

Un caso muy común al momento de modelar consiste el de relaciones N a N. Estudiemos el siguiente caso:

Un empleado puede trabajar en diversos proyectos de una empresa, los empleados para entrar a la plataforma necesitan identificarse con un correo corporativo, password y su nombre, de cada proyecto se tiene el nombre y una descripción.

Se pueden dar situaciones donde en un proyecto no trabaje ningún empleado y un empleado no trabaje en ningún proyecto.

Modelo conceptual

Recordemos nuestros pasos importantes:

1. **Identificar entidades**: en este caso empleado y proyecto
2. **Agrupar entidades con sus atributos:**

```
Empleado (email, password, nombre)
Proyecto (nombre, descripción)
```

3. **Identificamos relaciones y sus cardinalidades**
   Un empleado puede trabajar en 0 a n proyectos, en un proyecto. La relación es "participación"
4. **Identificamos candidatos únicos**
   En el caso de usuario el email es único, en el caso de proyecto el nombre podría serlo pero eventualmente dos proyectos podrían quedar con el mismo nombre, agregaremos el atributo **id**.

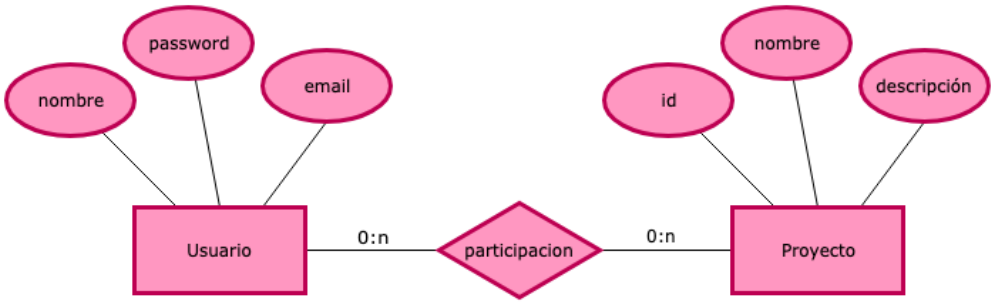

Imagen 18. Ejemplo Modelo Conceptual.

Modelo lógico

Para traspasar el modelo a lógico recordemos nuestros pasos:

1. Transformar todas las entidades en tablas, agregar los atributos como columnas de la tabla.
2. Transformar todas las relaciones del tipo N:N en tablas
3. Propagar la clave primaria de las tablas en las relaciones 1:N desde el lado de las 1 al lado de las N.

En este caso tenemos dos entidades mas la entidad de la relación **N a N**.

Al propagar las claves quedaremos con el siguiente diagrama.

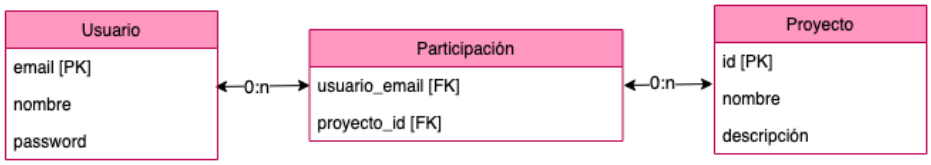

Imagen 19. Ejemplo Modelo Lógico.

Pero todavía nos queda hacernos unas preguntas importantes:

- ¿Puede un usuario trabajar dos veces en el mismo proyecto?
- ¿Cómo evitamos que esto suceda?

Para evitar los grupos repetitivos o sea que el usuario con email 123@123 trabaje en el **proyecto 1** varias veces, podemos agregar una **clave primara compuesta**. De esta forma nos aseguramos que los trabajos siempre tengan ambos campos y además que estos existan asegurando la integridad referencial.

```
CREATE TABLE usuario(
email VARCHAR(50),
nombre VARCHAR(20),
password VARCHAR(255),
PRIMARY KEY (email)
);

CREATE TABLE proyecto(
id CHAR(10),
nombre VARCHAR(50),
descripcion VARCHAR(1000),
PRIMARY KEY (id)
);
-- La tabla intermedia se crea al último para poder agregar las referencias
CREATE TABLE participacion(
usuario_email VARCHAR(50) REFERENCES usuario (email),
proyecto_id CHAR(10) REFERENCES proyecto (id),
PRIMARY KEY(usuario_email, proyecto_id)
);
```

Insertamos datos para probar nuestra consulta

```
INSERT INTO usuario(email, nombre, password) VALUES ('usuario1@gmail.com',
'Juan', '12345678');
INSERT INTO usuario(email, nombre, password) VALUES ('usuario2@gmail.com',
'Francisca', 'asdfghi');
INSERT INTO proyecto(id, nombre, descripcion) VALUES (1, 'Proyecto1', 'Proyecto
secreto');
INSERT INTO proyecto(id, nombre, descripcion) VALUES (2, 'Proyecto2', 'Proyecto
público');
INSERT INTO participacion(usuario_email, proyecto_id) values
('usuario1@gmail.com', 1);
INSERT INTO participacion(usuario_email, proyecto_id) values
('usuario1@gmail.com', 2);
INSERT INTO participacion(usuario_email, proyecto_id) values
('usuario2@gmail.com', 1);
```

Para seleccionar todos los usuarios con sus proyectos respectivos necesitamos hacer dos `JOIN`, usuarios con participación y participación con proyecto.

```
SELECT * FROM usuario
INNER JOIN participacion ON email = usuario_email
INNER JOIN proyecto ON proyecto.id = participacion.proyecto_id;
```

email	nombre	password	usuario_email	proyecto_id	id	nombre	descripcion
'usuario1@gmail.com'	'Juan'	'12345678'	'usuario1@gmail.com'	1	1	"Proyecto1	'Proyecto secreto'
'usuario1@gmail.com'	'Juan'	'12345678'	'usuario1@gmail.com'	2	2	"Proyecto2	'Proyecto público'
'usuario2@gmail.com'	'Francisca'	'asdfghi'	'usuario2@gmail.com'	1	1	"Proyecto1	'Proyecto secreto'

Imagen 20. Tabla.