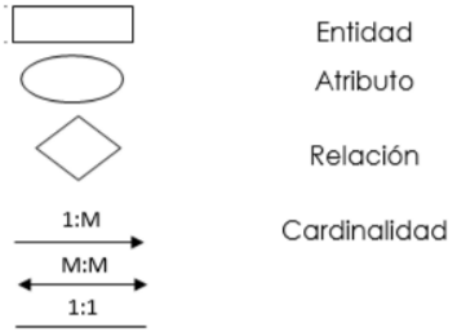
Bases de datos II

El diseño de bases de datos es un proceso demorado que permite representar un sistema complejo en un sistema de relaciones, de modo que cada parte del sistema esté dentro de la organización del mismo. Para esto se siguen algunos pasos en los que se describen ciertas lógicas. El diseño conceptual más usado es el de diagrama Entidad Relación, que es el que se explicará.

Diseño conceptual

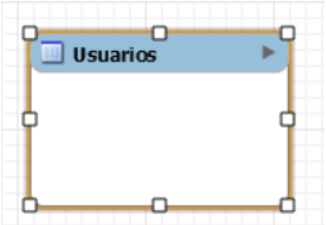
Es una etapa de análisis que incluye el conocer todos los requerimientos empresariales. Esto se logra luego de varias entrevistas y análisis. Lo que se busca es representar la realidad con un modelo descriptivo o semántico aproximado.

El diagrama de entidad relación utiliza formas para representar entidades, atributos y relaciones, las cuales se muestran a continuación :



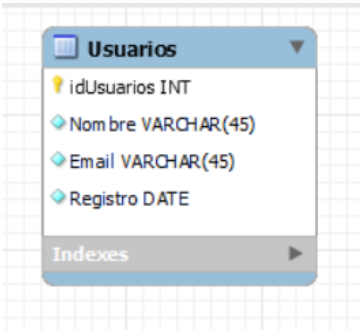
Entidad: Objeto que tiene sentido propio con su sola existencia, es decir no necesita más objetos para describirse, pero sí necesita atributos diferentes

En la etapa de **diseño lógico**, una entidad se representaría así:

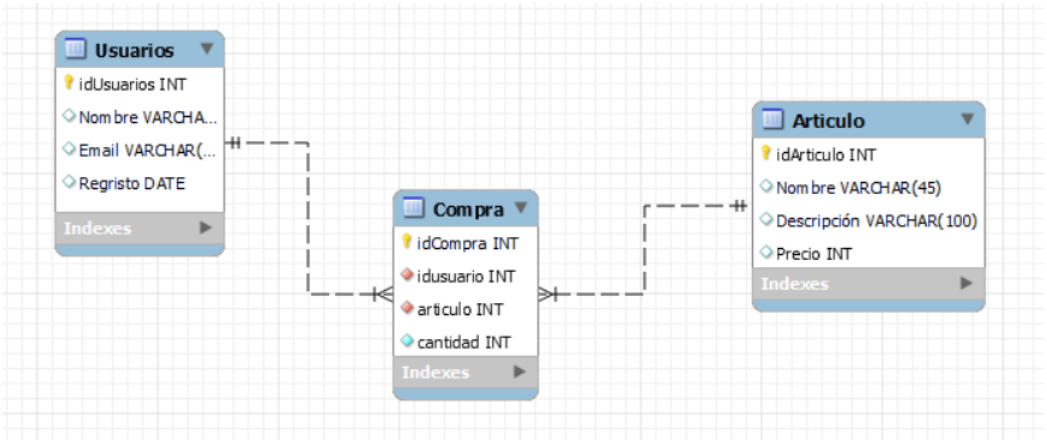


Atributos: Son las características de un objeto. Por ejemplo, el nombre, el email, su fecha de registro son atributos de un Usuario.

Por ejemplo en el siguiente ejemplo vemos una tabla con atributos de Usuario.



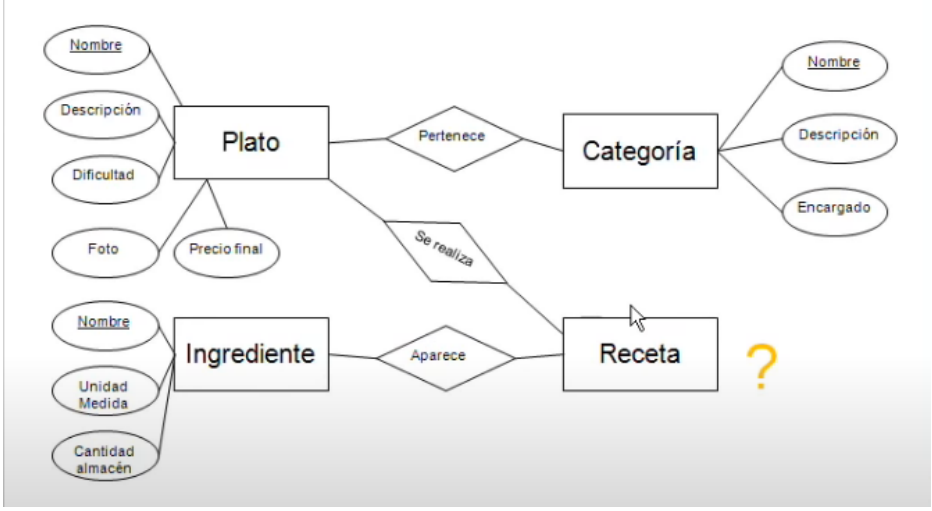
Relación: Una relación es una acción de correspondencia entre dos entidades. Por ejemplo que un usuario compre un articulo.

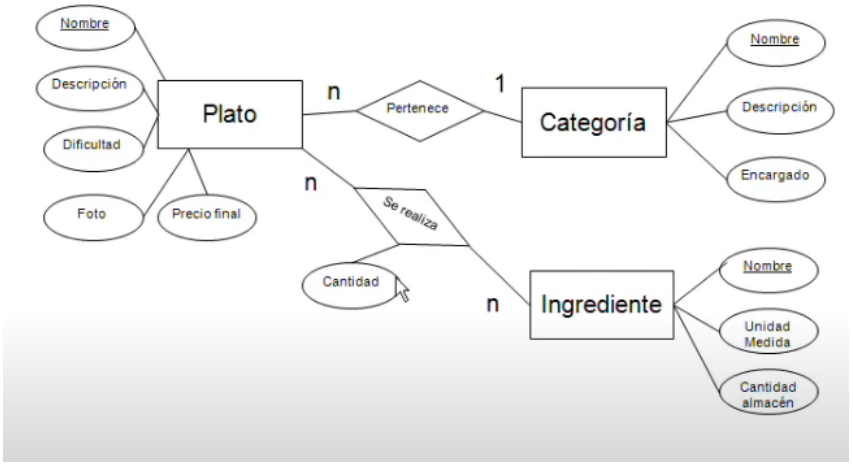


La relación también es una tabla, que se construye en base a dos entidades. Por eso algunas veces las tablas también son llamadas tablas relación.

Cardinalidad : Implica la relación entre 2 entidades, cuántos elementos del primero pueden relacionarse con cuántos del segundo. Por ejemplo, muchos clientes pueden comprar muchos artículos, por lo que la relación es M a M.

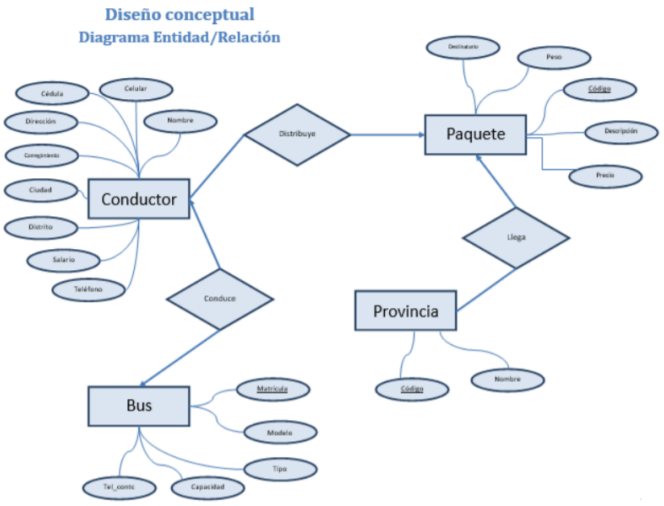
Ejemplo : Se desea construir una base de datos para un almacén de un restaurante. Para cada plato, se desea su nombre, descripción, nivel de dificultad, elaboración, una foto y el precio final para el cliente. Cada plato pertenece a una categoría. Las categorías se caracterizan por su nombre, una breve descripción y el nombre encargado. Además, de los platos se desea conocer la receta para su realización, con la lista de ingredientes necesarios, aportando la cantidad requerida, las unidades de medida (Gramos, litros, etc…) y cantidad actual en el almacén





Ejemplo 2 :

Se desea automatizar parte de la gestión de Expreso Veragüense, empresa que en un segmento de sus operaciones reparte paquetes de Santiago a Panamá y viceversa. Los encargados de llevar los paquetes son los conductores de los buses, de los que se quiere guardar el número de cédula, nombre, teléfono, dirección, corregimiento, ciudad, celular, distrito, salario. De los paquetes transportados interesa conocer el código de paquete, descripción, peso, destinatario y dirección del destinatario, precio de envío. Un conductor distribuye muchos paquetes, y un paquete sólo puede ser distribuido por un conductor. De las provincias a las que llegan los paquetes interesa guardar el código de provincia y el nombre. Un paquete sólo puede llegar a una provincia. Sin embargo, a una provincia pueden llegar varios paquetes. De los buses que utilizan los conductores, interesa conocer la matrícula, modelo, tipo y capacidad de pasajeros, teléfono para contrato del bus. Un conductor puede conducir diferentes buses en fechas diferentes, y un bus puede ser conducido por varios conductores



Diseño lógico

En esta etapa se parte del resultado del diseño conceptual, que se transforma al tipo de base de datos que vamos a utilizar. Más concretamente, es preciso que se ajuste al modelo del SGBD con el que se desea implementar la base de datos.

El proceso de normalización que se aplica en esta etapa consiste en una serie de reglas que deben cumplir las tablas y relaciones obtenidas tras el paso del modelo entidad relación al modelo relacional, para entonces ser un modelo lógico.

Las bases de datos relacionales se normalizan básicamente para: evitar la redundancia de los datos, evitar problemas de actualización de los datos en las tablas, proteger la integridad de los datos. Existen varios niveles de normalización de base de datos, en este caso aplicaremos las tres primeras formas normales que se describen a continuación:

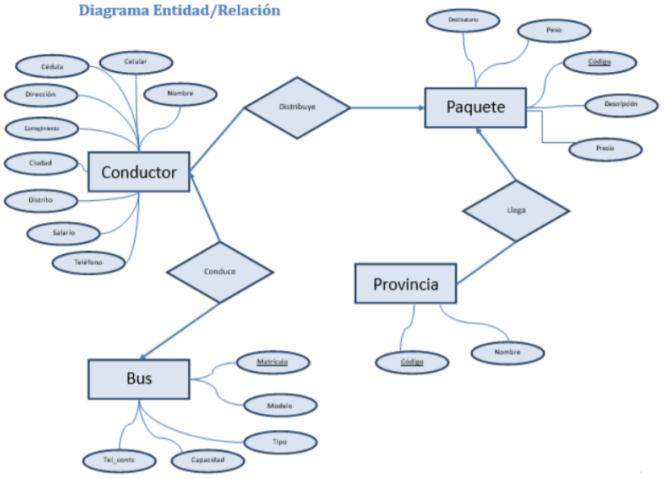
Primera forma normal (1FN) Se eliminan todos los campos o atributos repetidos Se asegura la atomicidad de los campos, en caso de existir atomicidad, se evalúa la creación de una nueva tabla Cada tabla debe tener una llave primaria Se asegura una dependencia funcional respecto a la llave primaria.

Segunda forma normal (2FN) Debe cumplir la primera forma normal No deben existir dependencias parciales: todos los campos no llaves deben depender solo de la llave primaria.

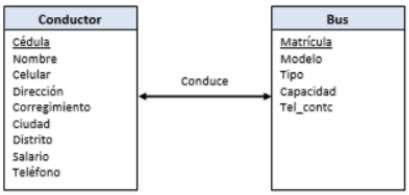
Tercera forma normal (3FN) Debe cumplir con la segunda forma normal No deben existir dependencias transitivas: ningún campo debe depender de un campo no llave

Diseño conceptual

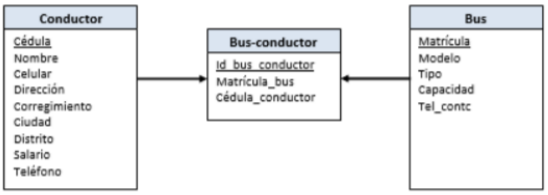
Diagrama Entidad/Relación



Relación muchos a muchos

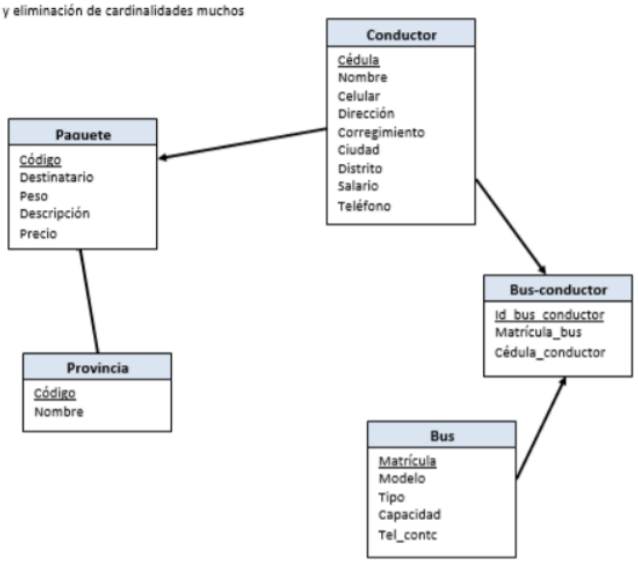


Solución de la relación muchos a muchos



Diseño Lógico

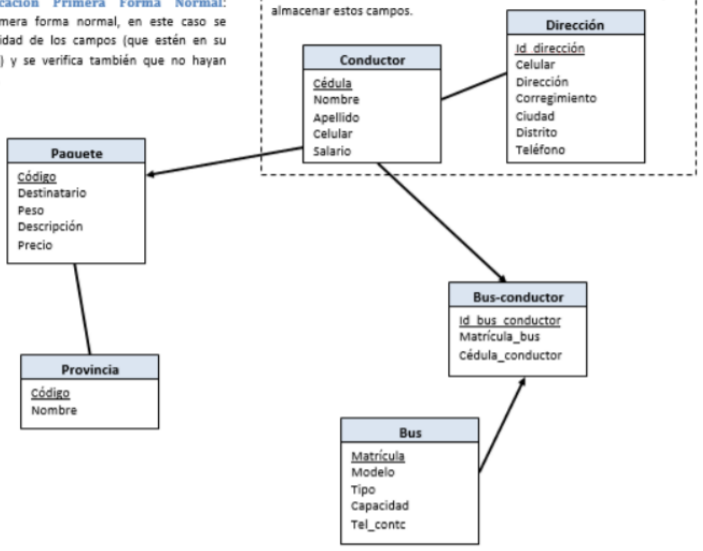
Fase #1, Entidades a tablas: Transformación de entidades a tablas y eliminación de cardinalidades muchos a muchos.



Diseño Lógico

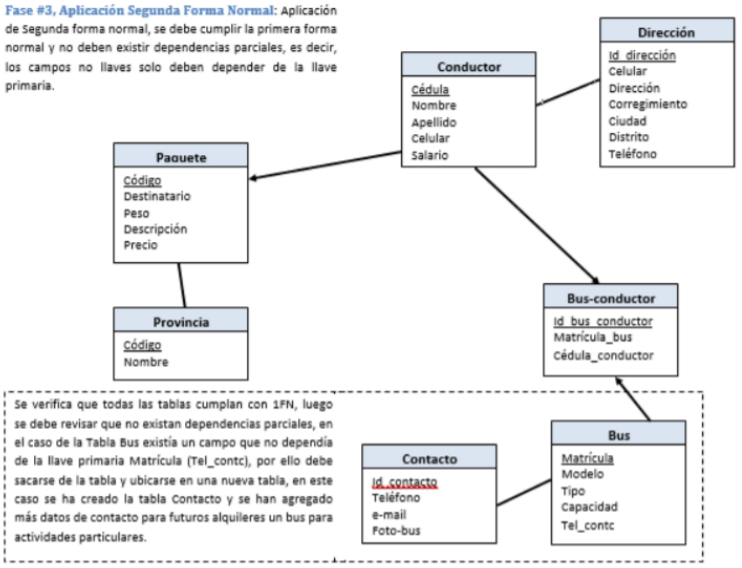
Fase #2, Aplicación Primera Forma Normal: Aplicación de Primera forma normal, en este caso se verifica la atomicidad de los campos (que estén en su mínima expresión) y se verifica también que no hayan campos repetidos.

Se verifica atomicidad de los campos de Conductor, en este caso se aprecia que el campo Nombre no es atómico, por lo tanto se crea un nuevo campo llamado Apellido; debido a que la tabla Conductor almacenaba muchos campos relacionados a la dirección del conductor, se crea una nueva tabla llamada Dirección con cardinalidad uno a uno para almacenar estos campos.

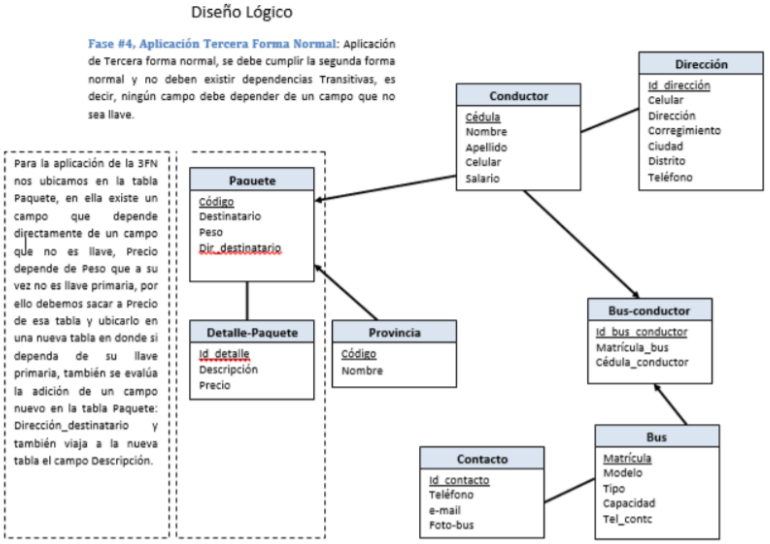


Diseño Lógico

Fase #3, Aplicación Segunda Forma Normal: Aplicación de Segunda forma normal, se debe cumplir la primera forma normal y no deben existir dependencias parciales, es decir, los campos no llaves solo deben depender de la llave primaria.



Se verifica que todas las tablas cumplan con 1FN, luego se debe revisar que no existan dependencias parciales, en el caso de la Tabla Bus existía un campo que no dependía de la llave primaria Matrícula (Tel_contc), por ello debe sacarse de la tabla y ubicarse en una nueva tabla, en este caso se ha creado la tabla Contacto y se han agregado más datos de contacto para futuros alquileres un bus para actividades particulares.



Método en partes:

1. Mapear todas las entidades normales : Con todos los atributos simples
2. Mapear todas las relaciones 1 : 1 : Incluir a un lado un llave foránea en favor de la participación total.
3. Mapear todas las relaciones 1:N : Incluye en lado de 1 al lado de N.
4. Mapear todas las relaciones binarias N:M : Se crea una nueva tabla compuesta que es la combinación de ambas entidades. También incluye la combinación de ambas.

Exportar base de datos:

Database + Forward Engineer + Next + Close

Asignar una llave foránea:

Para asignar una llave foránea tenemos dos opciones:

FOREING KEY (Atributo Rn) REFERENCES Rn-1(Key) ON DELETE SET NULL

FOREING KEY (Atributo Rn) REFERENCES Rn-1(Key) ON DELETE SET CASCADE

ON DELET SET NULL se refiere a que al eliminar una fila de referencia, quedará nulo y ON DELETE SET CASCADE se refiere a que al eliminar una fila de referencia, se eliminará la fila de relación.

ORDER BY : Para ordenar de acuerdo un atributo, va luego del select. Tiene los métodos ASC y DESC.

SELECT * FROM + R + ORDER BY + ATRIBUT

Wildcards

% = cualquier # caracteres, _ = un caracter

SELECT * FROM R WHERE ATRIBUTE LIKE '%DAN%'

Union

SELECT ATRIBUTE1 FROM R1 + UNION + SELECT ATRIBUTE2 FROM R2

Join

Sirve para unir 2 tablas

SELECT R.ATRIBUTE1, R2.ATRIBUTE2
FROM R
JOIN R2
ON CONDITION ENTRE AMBAS

Ejemplo

```
SELECT employee.emp_id, employee.first_name, branch.branch_na
FROM employee
JOIN branch
ON employee.emp_id = branch.mgr_id;
```

Se puede especificar LEFT JOIN para que aparezcan todas las filas de la tabla de la izquierda. con RIGHT JOIN aparecen todas las filas de la tabla de la derecha.

Solicitudes Unidas (Nested Queries)

Para unir solicitudes más complejas se puede hacer uso de las solicitudes unidas; por ejemplo:

https://www.geeksforgeeks.org/nested-queries-in-sql/

Disparadores

Se ejecutan desde la línea de cliente de Mysql

DELIMITER $$

```
CREATE
        TRIGGER my_trigger before insert
        on usuarios
        FOR EACH ROW BEGIN
                INSERT INTO trigger_test Values(‘se añadió un nuevo usuario’);
        END$$
DELIMITER;
```