FORMATO PARA EL DESARROLLO DE COMPONENTE FORMATIVO

PROGRAMA DE FORMACIÓN	**PROGRAMACIÓN DE APLICACIONES Y SERVICIOS PARA LA NUBE**

COMPETENCIA	220501113 Administrar base de datos de acuerdo con los estándares y requisitos técnicos	**RESULTADOS DE APRENDIZAJE**	220501113 -01 Modelar la base de datos de acuerdo a los requerimientos del sistema informático.

NÚMERO DEL COMPONENTE FORMATIVO	**1**
NOMBRE DEL COMPONENTE FORMATIVO	**Conceptos generales de bases de datos.**
BREVE DESCRIPCIÓN	En la sociedad de la información es imprescindible el empleo de tecnológicas, técnicas y procedimiento que a lo largo de las últimas décadas han consolidado un marco conceptual importe y necesario en todo proceso de administración de datos, este componente acerca estos conceptos de manera estructurada para avanzar en los procesos de gestión de la información.
PALABRAS CLAVE	Base de datos, relacional, integridad, dependencias.

AREA OCUPACIONAL	**No Aplica**
IDIOMA	**Español**

A. TABLA DE CONTENIDOS:

Introducción

1. **Conceptos generales de bases de datos**
   - 1.1. Tipos de datos y restricción de no nulidad.
   - 1.2. Tipos de bases de datos.
   - 1.3. Clasificación de bases de datos.
   - 1.4. Sistema de gestión de bases de datos.
2. **Modelo entidad relación**
   - 2.1. Relaciones entre entidades.
   - 2.2. Relaciones de uno a muchos.
   - 2.3. Relaciones de un uno a muchos.
   - 2.4. Relaciones de uno a uno
3. **Normalización**
   - 3.1. Dependencias funcionales.
   - 3.2. Diseño relacional.
   - 3.3. Reglas de integridad
   - 3.4. Lenguajes de los sistemas administradores de bases de datos
4. **Sistema gestor de base de datos**

B. DESARROLLO DE CONTENIDOS:

Introducción

Una base de datos se puede percibir como un “almacén” de información que se define y se crea una sola vez, para guardar grandes cantidades de datos de forma organizada (o estructurada), con el fin de poder encontrarla y utilizarla fácilmente. Se presentará el concepto y características de las bases de datos actuales, teniendo en cuenta que cada base de datos es diseñada para cumplir los requisitos de información de una organización o una empresa. Estos diseños están concebidos para emplear Sistemas de Gestión de Bases de Datos (en adelante SGDB), que es un sistema o servicio informático que permite a las personas definir, crear, dar soporte y mantenimiento a las bases de datos, controlando el acceso de forma segura.

1. Conceptos generales de bases de datos

El concepto general de bases de datos según la RAE es el “conjunto de datos organizado de tal modo que permita obtener con rapidez diversos tipos de información” (RAE ,2001). De acuerdo a esta definición, una hoja de cálculo de Excel puede considerarse una base datos, o un conjunto de archivos debidamente organizados, o la lista de nombres y teléfonos que está en nuestros *Smartphone*. En principio es correcto llamarle base de datos a estos ejemplos, sin embargo, en el contexto de desarrollo de *software*, se referirá a ese conjunto de información que puede ser almacenada en grandes cantidades de forma organizada y es gestionada desde a través de un **sistema de gestión de bases de datos** (SGBD).

Para poder organizar y definir la información de forma sistemática, las bases de datos también deben poder almacenar una descripción precisa de los datos que contiene conocida como **metadatos** a los que se le relaciona el tipo de información que conceptualmente guardada (es decir si se agrega una explicación de la naturaleza del dato en la empresa) se da origen a lo que conoce como **diccionario de datos** o **catálogo de datos**.

Ejemplo:

Determinar los metadatos del siguiente problema.

Una empresa tiene una base datos de llamadas telefónicas que entran y salen de su planta telefónica donde se relacionan los números de teléfonos del llamante y la extensión telefónica que recibe o hace la llamada, la fecha y la hora de la llamada, si fue o no atendida, y la duración de la misma como se muestra en la siguiente ilustración.

calldate	src	dst	duration	disposition
2021-04-10 08:48:50.0	3176964335	444	00:09:23	ANSWERED
2021-04-10 08:54:14.0	3217212347	444	00:07:47	ANSWERED
2021-04-10 08:57:17.0	3203857798	444	00:05:17	ANSWERED
2021-04-10 09:03:27.0	3226140856	666	00:04:22	ANSWERED
2021-04-10 09:14:13.0	28201400	666	00:00:36	ANSWERED
2021-04-10 09:14:45.0	3148915614	666	00:00:34	ANSWERED
2021-04-10 09:15:29.0	3148915614	666	00:02:27	ANSWERED
2021-04-10 09:18:13.0	3148915614	666	00:01:28	ANSWERED
2021-04-10 09:07:33.0	3005373259	444	00:12:12	ANSWERED
2021-04-10 09:13:19.0	3124523352	444	00:09:06	ANSWERED
2021-04-10 09:21:30.0	7006	883108608263	00:00:54	ANSWERED
2021-04-10 09:22:41.0	3125228302	666	00:03:12	ANSWERED
2021-04-10 09:28:04.0	3203498937	s	00:00:05	ANSWERED
2021-04-10 09:29:13.0	3017363892	tdial	00:00:00	FAILED
2021-04-10 09:29:23.0	3017363892	tdial	00:00:00	NO ANSWER
2021-04-10 09:30:07.0	7006	883507070131	00:00:00	NO ANSWER
2021-04-10 09:30:28.0	7006	883507070131	00:00:00	NO ANSWER
2021-04-10 09:28:35.0	3203498937	666	00:02:20	ANSWERED
2021-04-10 09:30:58.0	7006	883507070131	00:00:00	NO ANSWER
2021-04-10 09:28:32.0	3184155040	666	00:03:20	ANSWERED
2021-04-10 09:27:51.0	3235767896	666	00:04:32	ANSWERED
2021-04-10 09:28:49.0	3158628495	444	00:06:57	ANSWERED
2021-04-10 09:35:45.0	3214078916	666	00:00:52	ANSWERED
2021-04-10 09:35:18.0	3004668735	1	00:02:07	ANSWERED
2021-04-10 09:39:53.0	3226140856	666	00:02:15	ANSWERED
2021-04-10 09:37:02.0	3046032130	666	00:05:30	ANSWERED
2021-04-10 09:44:57.0	7006	883113243787	00:00:00	NO ANSWER

Ilustración 1. Base de datos de llamadas telefónicas.

NOTA IMPORTANTE: la ilustración 1 muestra una tabla de datos que tiene las características de que en cada columna se agrupa un tipo de dato particular, y cada fila es el conjunto de dato de se llama registro que comparten una relación (la relación es que cada fila corresponde a una única llamada telefónica), a esta estructura se le llama bidimensional, filas y columnas son las dos dimensiones.

Los metadatos de la lustración anterior serian la definición de cata tipo de dato como se lista a continuación:

calldate: tiene la estructura YYYY-MM-DD HH:MI:SS.Z , donde:

- YYYY sería el año un dato numérico entero de cuatro dígitos.
- MM el mes un dato numérico entero de dos dígitos entre 01 y 12.
- DD un dato numérico entero de dos dígitos entre 01 y 31
- HH un dato numérico de dos dígitos entre 00 y 23.
- MI y SS un dato numérico de dos dígitos ente 00 y 59
- Z Un dato numérico de un digito (para representar milésimas de segundo).

Como este es el conjunto de datos de la primera columna se puede que use un término que describa lo anterior con una sola palabra que es **TIMESTAMP**, de esta forma esa primera columna puede quedar correctamente descrita como:

```
calldate TIMESTAM(1)
```
Donde 1 es el número de dígitos que milisegundos (Z).

src y dst : empleado para fuente de la llamada o llamante (src) y destino de la llamada (dst) puede ser un número telefónico o de extensión telefónica, algunos tienen el código de país seguido del signo más ejemplo: 57+3155008002. Por lo tanto, se puede describir como una cadena de texto que no superará los 25 caracteres, pero puede tener menos de 25 caracteres, así el metadato iría descrito:

Comentado [OG1]: Captura del documento en Excel denominado “Datos ejemplo” hoja “Exportar datos de trabajo”.

Comentado [OG2]: Utilizar bloque de texto destacado Imagen+texto

Comentado [OG3]: Cajón texto color

Comentado [OG4]: Listado no ordenado color.

`src VARCHAR(25)` `dst VARCHAR(25)`	**De esta forma decimos que es una cadena de caracteres que varía en longitud de 0 a 25 caracteres.**

duration: tiene la estructura HH:MI:SS y como se vio antes representa el tiempo en horas, minutos y segundos:

`src TIME`	**TIME se usa para esta estructura de dato en particular**

disposition: indica si la llamada es atendida ANSWER, no atendida NO ANSWER, o si fallo la FAIL. Se puede representar con una cadena de 10 caracteres.

Por lo tanto, la solución de los metadatos de la base de datos de la ilustración 1 son:

```
calldate TIMESTAM(1),
src VARCHAR(25),
dst VARCHAR(25),
src TIME,
disposition VARCHAR(10)
```

Comentado [OG5]: Utilizar fuente consola. Courier new

Note que cada uno de ellos fue separado por un coma para que se diferencie de los demás.

Ejemplo 2:
Conociendo el objetivo de la base datos de la ilustración 1 y los metadatos cree un diccionario de dato o un catálogo de datos.

Solución:

NOMBRE	Base datos de llamadas telefónicas		
CREACIÓN	27/02/2021		
DESCRIPCIÓN	Registro de las llamadas telefónicas de la PBX de la empresa STD LTDA		
CAMPO	**TIPO DATO**	**TAMAÑO**	**DESCRIPCION**
calldate	**TIMESTAMP**	**1**	Representa el momento exacto en que entra o sale la llamada
src	**VARCHAR**	**25**	Fuente (source) de la llamada, número de quien hace la llamada telefónica
dst	**VARCHAR**	**25**	Número del destino de la llamada telefónica
duration	**TIME**	**0**	Duración de la llamada telefónica
disposition	**VARCHAR**	**10**	Estado final en que se considera quedó la llamada

Tabla 1 Diccionario de datos base de llamadas telefónica.

Comentado [OG6]: Utilizar bloque de texto destacado. Imagen+texto
La imagen sería la tabla1.

Como se puede notar en la tabla anterior, un diccionario de datos aporta información de la estructura de los datos y el uso que se da a cada dato en la empresa, organización o sistema de información, pero no sin antes darle un nombre a todo el conjunto de datos (primer fila), una fecha (segunda fila) y una descripción (tercera fila) del conjunto de los datos. En la tabla 1 se puede identificar la independencia que existe entre la lógica de los datos (representado por la columna DESCRIPCIÓN del dato) y el almacenamiento físico (estructura de almacenamiento representado por las columnas TIPO DE DATO y TAMAÑO).

1.1. Tipos de datos y restricción de no nulidad.

Existen muchos tipos de datos y varían según el SGDB, ya que cada sistema gestor define sus propios tipos de datos, aunque existen las equivalencias notables y otras no tan notables vemos una tabla que nos relacione el tipo de datos según los SGDB más comunes del mercado como *ORACLE*, *PostgeSQL*, *MySQL* y *SQLServer*.

Tipo de dato	ORACLE	PostgreSQL	MySQL	SQLServer
Cadena caracteres	**VARCHAR2**	**CHARACTER VARYING**	**VARCHAR**	**VARCHAR**
Cadena texto	**TEXT**	**TEXT**	**TEXT**	**NTEXT**
Entero pequeño	**SMALLINT**	**SMALLINT**	**SMALLINT**	**SMALLINT**
Entero	**INTEGER**	**INTEGER**	**INT**	**INT**
Fecha	**DATE**	**DATE**	**DATE**	**DATE**
Fecha y hora	**DATE**	**TIMESTAMP WITH TIME ZONE**	**DATETIME**	**DATETIME 2**
Hora	**DATE**	**TIME**	**TIME**	**TIME**
Entero con decimales	**FLOAT**	**REAL**	**FLOAT**	**FLOAT**

Tabla 2. Tipos de datos según bases de datos.

Según el tipo de sistema gestor de bases de datos cada uno define sus tipos de datos, que de alguna forma son muy parecidos entre ellos, o existen equivalencias, las diferencias son pocas y están relacionadas con aspectos técnicos del almacenamiento, se pueden ver más detalles de cada uno de ellos en los sitios oficiales de cada motor de base de datos:

Nombre	Enlace de acceso
ORACLE	https://docs.oracle.com/cd/E11882_01/server.112/e41085/sqlqr06002.htm#SQLQR959
PostgreSQL	https://www.postgresql.org/docs/13/datatype.html#DATATYPE-TABLE
MySQL	https://dev.mysql.com/doc/refman/8.0/en/data-types.html
SQLServer	https://docs.microsoft.com/en-us/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver15

Existen varios tipos de restricciones, por ahora veremos ya se ha visto la restricción de longitud, por ejemplo:

```
src VARCHAR(25)
```

En él se definen que **src**, es una columna que no puede exceder los 25 caracteres. De la misma forma existe una forma de definir, que es obligatorio registrar un dato en esa columna y se llama **restricción de no nulidad**, es decir, que ninguna fila puede no pueden tener valor nulo. De esta forma en el ejemplo de la base de datos de llamadas telefónicas si se define que todo y cada uno de los datos son obligatorios (usando las palabras **NOT NULL**), la metadata que define la estructura de los datos quedaría así:

```
calldate TIMESTAM(1) NOT NULL,
src VARCHAR(25)NOT NULL,
dst VARCHAR(25)NOT NULL,
src TIME NOT NULL,
disposition VARCHAR(10) NOT NULL
```

Algunos sistemas, para abreviar en la decisión de no nulidad acostumbra usar el acrónimo **NN** para representar **N**OT **N**ULL.

1.2. Tipos de bases de datos.

Las bases de datos han evolucionado a lo largo del tiempo, usando diferentes tipos de modelos o enfoques, se comprobado que algunos de ellos tienen mejor desempeño en distintos tipos de aplicación, La estructura en la que se configuran los datos al interior de las bases de datos es una forma de clasifican:

a. **Bases de datos con estructura jerárquica**
En las primeras bases de datos empleadas son la de estructura jerárquica. Las relaciones entre registros forman una estructura en árbol, en la actualidad un ejemplo de estas bases datos es la base da tos LDAP usada en la actualidad para sistema de directorio de datos tales como usuario, dispositivos, nombres, contraseñas, direcciones etc. dentro de una red de computadores.

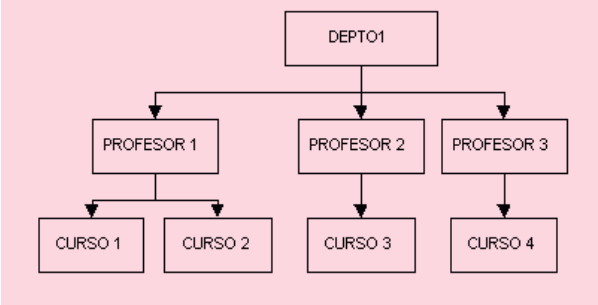


Ilustración 2. Base de datos Jerárquica (Fuente https://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_bases_de_datos#/media/Archivo:DB_jerarquica.png)

b. **Bases de datos con estructura en red**
Esta estructura es un poco más compleja que la anterior porque contiene otras relaciones que las jerárquicas admitiendo relaciones de cada registro otros de manera que puede seguir por distintos caminos para acezar la información.

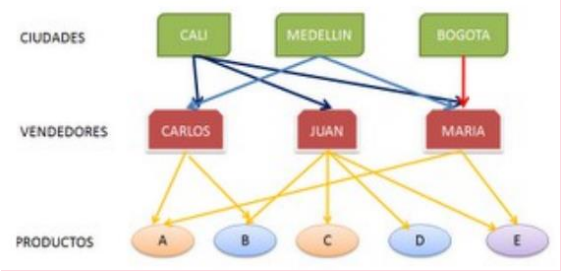


Ilustración 3. Base de datos con estructura de red. (Fuente https://gestionbasesdatos.readthedocs.io/es/latest/Tema1/Teoria.html)

c. Bases de datos con estructura relacional

Es estas bases de datos son multipropósito, lo que la hace las más extendidas en la industria y más empleada, y también la más compleja de aprender ampliamente se bordara el diseño e implementación de este tipo de bases de datos.

Se basan tablas como las que se presentó en la ilustración 1 y en la relación de los datos de unas tablas con otras.

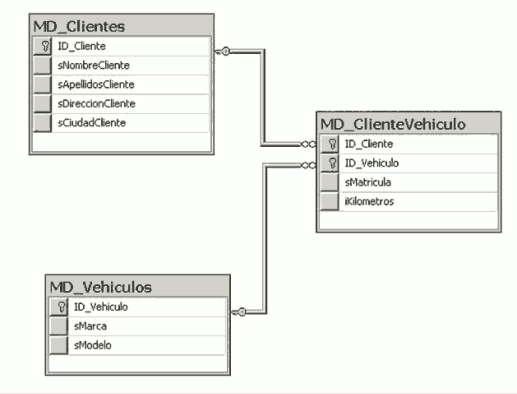


Imagen. Tablas y estructura relacional.
(Fuente https://images.app.goo.gl/HgMzrfU1qTpSY1FG6)

d. Bases de datos con estructura multidimensional

Es la extensión de las bases de datos de estructuras relacionales, como se vio en las relacionales se basan en estructuras bidimensionales (tablas), mientras la multidimensional en cubos (tres dimensiones), o más complejas N dimensiones.

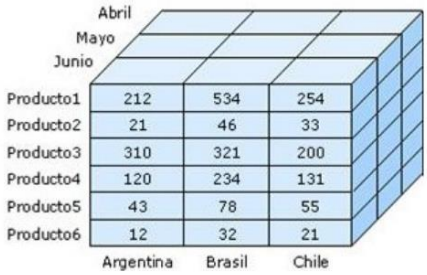


Ilustración 3. Estructura de datos de CUBO. (Fuente https://gestionbasesdatos.readthedocs.io/es/latest/Tema1/Teoria.html).

e. Bases de datos con estructura orientada a objetos

Una base de datos orientada a objetos es un sistema de gestión de base de datos mediante el cual representamos la información en forma de objetos que son utilizados en programación orientada a objetos (Kyocera, 2021).

Comentado [OG7]: Utilizar pestaña o tabs para que el aprendiz pueda ir navegando sobre los diferentes tipos de bases de datos:

a.Bases de datos con estructura jerárquica
b.Bases de datos con estructura en red
c.Bases de datos con estructura relacional
d.Bases de datos con estructura multidimensional
e.Bases de datos con estructura orientada a objetos

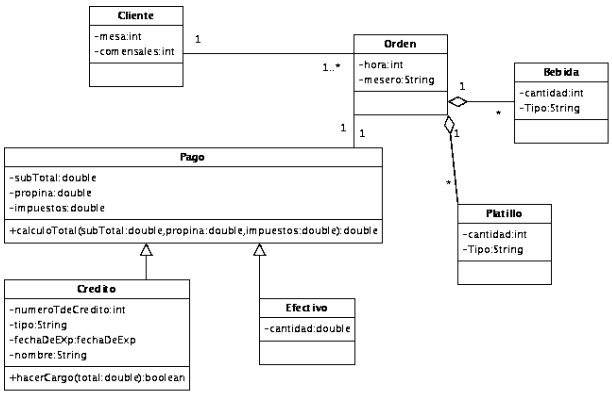


Imagen. Estructura orientada a objetos.
(Fuente https://images.app.goo.gl/JNk8HeHE4LZ1rKc59)

Cada tipo de bases de datos tiene un ámbito de aplicación en el que es mejor su desempeño. Por ejemplo, las bases de datos jerárquicas tienen un mejor desempeño en operaciones de consulta de datos puntuales, mientras que las bases de datos relacionales son precisas para garantizar la calidad de los datos y la no repetición de los mismos, aunque no muy optimizadas para hacer consultas, y finalmente las multidimensionales son muy útiles para análisis estadístico de datos históricos de volúmenes inmensos de datos. Aparte del tipo de datos existe otro tipo de clasificaciones ya no según la estructura en que se almacenan los datos sino de acuerdo a la naturaleza de los datos.

1.3. Clasificación de bases de datos.

Según la naturaleza de los datos que se almacenan en la base de datos se pueden clasificar. La siguiente tabla muestra una compilación de estas clasificaciones.

Criterio de clasificación	Clasificación	Descripción
Según la variabilidad de los datos.	Bases de datos estáticas	• Son bases de datos, cuyos datos son históricos, es decir ya no se pueden modificar, se usan comúnmente para estudiar el comportamiento de los datos a través del tiempo. • Generalmente a este tipo de bases de datos se les llama bodega de datos. • Muchas de estas bases de datos de almacenan en cubos para su análisis se las conoce como OLAP es el acrónimo en inglés de procesamiento analítico en línea (On-Line Analytical Processing).
	Bases de datos dinámicas	• Estos datos se almacenan y pueden ser modificados, agregados, borrados y consultados en cualquier momento, por ejemplo: un sistema de facturación o nuestro sistema Sofiaplus.

		• A estos sistemas se les denomina sistemas transaccionales, porque cada operación de guardar, borrar o editar se configura como una transacción. OLTP es la sigla en inglés de Procesamiento de Transacciones en Línea (*OnLine Transaction Processing*).
Según el contenido	Bases de datos documentales	• Permiten la indexación a texto completo, y en líneas generales realizar búsquedas más potentes.
	Base de datos deductivos	• Un sistema de base de datos deductivos, es un sistema de base de datos, pero con la diferencia de que permite hacer deducciones a través de inferencias. Se basa principalmente en reglas y hechos que son almacenados en la base de datos. También las bases de datos deductivas son llamadas base de datos lógica, a raíz de que se basan en lógica matemática.

Tabla 3. Clasificación de bases de datos.

Comentado [OG8]: Tabla. Leyenda: Clasificación de bases de datos.

Pueden existir otras clasificaciones generalmente más orientadas al ámbito de uso, sin embargo, se presentan las más comunes en el ejercicio del desarrollo de *software*. La base de datos relacionales es de obligatorio dominio en cualquiera de los casos, ya que los conceptos de esta son reutilizables en casi todos los otros tipos.

1.4. Sistema de gestión de bases de datos

Un sistema de gestión de la base de datos es un programa de computadora que permite definir, crear y mantener los datos de una base de datos, controlando el acceso.
Los servicios SGDB ofrece y permite los siguientes servicios:

- Permitir la definición de la base de datos usando un lenguaje de definición de datos.
- Permitir la inserción, actualización, eliminación y consulta de datos usando un lenguaje de manejo de datos.
- Proporciona un acceso controlado a la base de dato (con autenticación, roles niveles de acceso)
- Concurrencia (varios usuarios a la vez accediendo o manipulando los datos) y multitarea.
- Algunos SGDB permiten administrar le catálogo de datos.

Comentado [OG9]: Lista de color no ordenada.

Los SGBD son una herramienta muy útil. Sin embargo, se podría decir que los SGBD son un poco más complejos, ya que los usuarios ven más datos y sus relaciones de los que realmente se necesitan.

2. Modelo entidad relación

Las bases de datos relaciones inicialmente no son fáciles de modelar, por lo que se requiere de cierto nivel de abstracción, es decir se va ocultando la complejidad del diseño de la base de datos, primero analizando el requerimiento desde un punto de vista conceptual, entendido como las necesidades de la empresa o sistema de información, luego se va analizando des de un punto de vista de la lógica de los datos y finalmente se analiza físicamente como se almacenaran los datos, para ello aquí algunos conceptos.

A. ENTIDAD

El modelo de datos entidad-relación (E-R) está basado en una percepción del mundo real que consta de una colección de objetos básicos, llamados entidades, y de relaciones entre estos objetos. Una entidad es una «cosa»

u «objeto» en el mundo real que es distinguible de otros objetos. Por ejemplo, cada **persona** es una entidad, y las **cuentas** bancarias pueden ser consideradas entidades (Henry, 2002). Se representa con un rectángulo.

B. **ATRIBUTOS**

Las entidades están descritas por un conjunto de tributos por ejemplo una persona puede tener estos atributos: **nombres**, **apellidos**, **edad**, documento de identidad (**identificación**); o la cuenta bancaria puede tener los siguientes atributos: **número de cuenta**, **saldo**, **fecha de creación**. Se representa con una elipse.

C. **RELACIONES**

Una relación es una asociación entre varias entidades. Por ejemplo, una relación **titular** asocia un cliente con cada cuenta que tiene (de la que es titular). Se verá más claro un diagrama, pero antes de verlo considere los siguientes elementos para elaborar un diagrama. Se presentante con un rombo.

- Rectángulos, que representan conjuntos de entidades.
- Elipses, que representan atributos.
- Rombos, que representan relaciones entre conjuntos de entidades.
- Líneas, que unen los atributos con los conjuntos de entidades y los conjuntos de entidades con las relaciones.

Comentado [OG10]: Listado no ordenado.

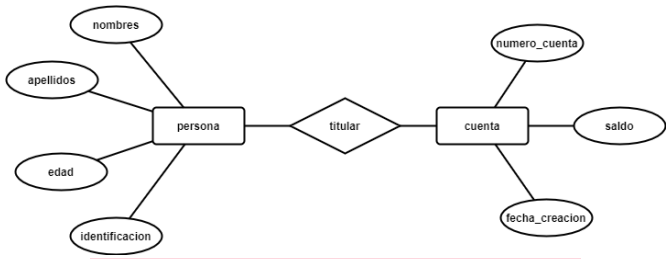


Ilustración 4. Entidad relación persona cuenta bancaria.
Fuente (https://drive.google.com/file/d/1W-ggJPezRs7noNMYjiOdAk2c5SMXfB3o/view?usp=sharing Page 1)

Comentado [OG11]: En el archivo online se encuentran relacionadas las ilustraciones que expone el experto. En el mismo se específica la hoja donde se encuentra para que pueda ser digitalizada si se requiere.

Para una base de datos de un sistema bancario consistente en clientes (es decir personas) y cuentas bancarias que tienen esos clientes. En la Ilustración 4 se muestra el diagrama Entidad Relación correspondiente. En él se indica que hay dos entidades que son **persona** y **cuenta**, con los atributos descritos para cada una anteriormente. El diagrama también representa la relación **titular** entre persona y cuenta.

Tenga en cuenta que el modelo relacional convierte las entidades en tablas por lo tanto se puede poner como ejemplo que las entidades personas se convierta en tablas como muestra la siguiente ilustración.

identificacion	nombres	apellidos	edad
8.127.346	Ana lis	Mendez	34
212.394.762	Jose Omar	Ciano	28
545.232.546	Luis Dario	Marquez	21
125.446.732	Armando	Bronca Segura	35
33.565.236.787	Vistor José	Diento Paz	24
2.343.554.768	Jose Nicolás	Timado Ruiz	19
45.456.778	Andrea	Martinez	36

a) Tabla persona

numero_cuenta	saldo	fecha_creacion
19.283.746	$ 2.589.954,00	25/08/1998
67.789.901	$ 23.465.212,00	26/08/2001
32.112.312	$ 2.343.242,00	27/08/2004
41.928.374	$ 2.343.423,00	31/08/2013
38.342.657	$ 8.965.211,00	30/08/2010
32.443.171	$ 11.172.594,33	22/06/2001
34.920.608	$ 14.483.578,83	01/09/2016
17.980.455	$ 17.794.563,33	03/09/2019
26.754.825	$ 21.105.547,83	12/04/2022

b) Tabla cuenta

Ilustración 5. Tablas de una entidad relación.

Comentado [OG12]: Las tablas reales se encuentran en los anexos el documento denominado “Datos ejemplo” y la hoja Ejemplo 1

D. **TUPLA**

Las tablas de la ilustración 5 muestran cada un conjunto de registros, la tabla persona muestra 6 registros y la tabla cuenta muestra 9 de ellos, cada uno de estos registros representa un objeto, es decir, de la tabla persona hay 6 (objetos) o personas a la agrupación de los datos de cada registro se le llama TUPLA. En términos simples una TUPLA de una base datos es el conjunto de todos los atributos (columnas) de una fila.

Comentado [OG13]: Hacer uso de tabs o pestañas para que el aprendiz pueda ir verificando los conceptos:
A.ENTIDAD
B.**ATRIBUTOS**
C.**RELACIONES**
D.**TUPLA**

2.1. Relaciones entre entidades.

Como se observa en la Ilustraciones 4 y 5 una entidad se puede representar con una tabla, sin embargo, la relación **titular** aún no se sabe cómo representarla, para ello se debe definir antes algunas propiedades de los atributos antes.

Principio de unicidad: en una tabla, nunca deben existir TUPLAS repetidas, es decir, no debe haber más de una fila con exactamente los mismos valores en cada columna que otra fila de la misma tabla. En otras palabas cada fila debe diferir al menos en un elemento (columna) con relación a las demás.

Comentado [OG14]: Cajón de texto color.

En una base de datos relacional, las relaciones permiten evitar datos redundantes. En el ejemplo una persona pudiera tener varias cuentas bancarias, pero no es deseable que, por cada una de las cuentas bancarias asociadas a la misma persona, se deban repetir lo datos de la misma persona. Sino más bien que se debe encontrar un mecanismo para relacionar la información de las dos tablas. Para lo cual se introducen algunos conceptos útiles.

- **Claves**

  Dado que en una tabla no hay filas repetidas, se pueden distinguir unas de otras, es decir, se pueden identificar de manera única (por el principio de unidad). Por lo tanto, la forma de identificarlas inequívocamente es según los valores de los atributos. De esta forma se conoce como claves a que los atributos que sirven para identificar un registro de forma inequívoca.

Ejemplo:

En la tabla persona, es sabido por todos que no existe dos personas con el mismo número de documento de identidad, por lo tanto, se puede emplear el atributo **identificación** como calve de la tabla persona. De igual forma no debe existir dos cuentas bancarias con el mismo número de cuenta, entonces, el atributo **numero_cuenta** se puede emplear como clave de la tabla cuenta. Esto se lo puede presentar

en un diagrama de entidad relación cuando a los atributos que componen una clave les marcamos subrayado el nombre del atributo como se muestra en la siguiente ilustración.

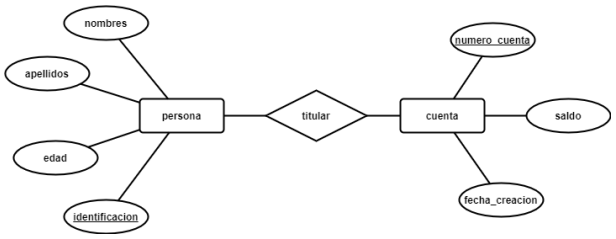


Ilustración 6. Claves de las entidades persona y cuenta

Fuente (https://drive.google.com/file/d/1W-qgJPezRs7noNMYjiOdAk2c5SMXfB3o/view?usp=sharing Page 2)

Comentado [OG15]: En el archivo online se encuentran relacionadas las ilustraciones que expone el experto. En el mismo se específica la hoja donde se encuentra para que pueda ser digitalizada si se requiere.

Notese como los atributos **identificacion** y **numero_cuenta** tiene subrayado la palabra identificadora del atributo, representado de esta forma que ese atributo es la clave (que hace que se cumpla el criterio de unicidad) de su entidad.

- **Claves cantidadata, superclase y clave primaria.**
  Suponga el caso en que el banco ahora es internacional, que es un banco moderno y solo diseñado para transacciones vía internet o virtuales, por lo tanto, el correo del cliente (persona) es necesario y obligatorio.

  En este caso ocurren un par de asuntos a analizar: al ser internacional el banco, puede existir un cliente en un país con el mismo número de identificación de otro cliente en otro país, por lo tanto, el atributo **identificación** ya no puede ser una clave de la entidad, sin embargo es importante y obligatorio saber a qué país corresponde cada cliente, el conjunto de datos **identificación** y **país** ambos si cumplen el criterio de unidad, porque no existen dos personas en el mismo país con el mismo número de identificación.

- **Clave compuesta**
  Si se escogiera el conjunto de atributos **identificación** y **país** como clave de la entidad, se está haciendo una clave compuesta porque es un conjunto de más de un atributo el que se usa para este fin.

  El otro asunto a considerar es que, se puede emplear el **correo** electrónico como clave ya que no existen dos usuarios con el mismo correo electrónico, esto es válido si se definió como regla del banco que cada cliente tenga su **correo** electrónico diferente de los demás, estos dos calos se representanta en la siguiente ilustración.

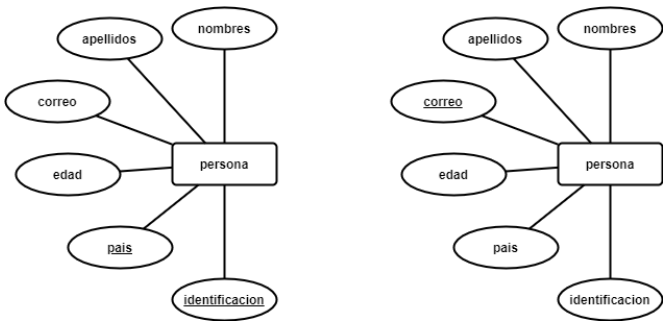


Ilustración 7. Claves Candidatas

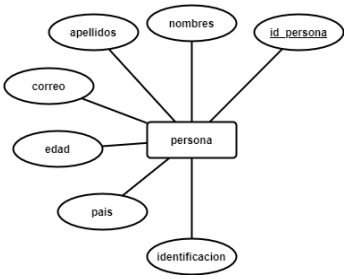
Fuente (https://drive.google.com/file/d/1W-ggJPezRs7noNMYjiOdAk2c5SMXfB3o/view?usp=sharing Page 3)

Comentado [OG16]: En el archivo online se encuentran relacionadas las ilustraciones que expone el experto. En el mismo se específica la hoja donde se encuentra para que pueda ser digitalizada si se requiere.

- **Clave candidata**

  Los dos caminos propuestos en la ilustración 7 son válidos, y se denominan claves candidatas porque con ambas se puede lograr el principio de unicidad, y es potestad de diseñador o analista de sistemas determinar cuál de las dos usa.

  Sin embargo, existe otro tipo de solución y es asignar le un atributo identificador, a la entidad persona que se encargue exclusivamente de identificar inequívocamente cada tupla o fila, a este atributo le podemos llamar **id_persona** que sería un número que el programa o software para el que estamos desarrollando la base le asignara a cada cliente cada vez que se registre por primera vez los datos de una persona (generalmente una secuencia). De esta forma tendríamos 3 claves candidatas, la ilustración 8 muestra cómo sería la entidad y la tabla que representa.



Entidad persona

id_persona	pais	identificacion	correo	nombres	apellidos	edad
1	colombia	8.127.346	ana@rmail.com	Ana lis	Mendez	34
2	colombia	212.394.762	jose@pmail.com	Jose Omar	Ciano	28
3	colombia	545.232.546	luis@jmail.com	Luis Dario	Marquez	21
4	colombia	125.446.732	arm@hmail.com	Armando	Bronca Segura	35
5	colombia	33.565.236.787	victor@umail.com	Vistor José	Diento Paz	24
6	colombia	2.343.554.768	josen@pmail.com	Jose Nicolás	Timado Ruiz	19
7	perú	45.456.778	andrea@email.com	Andrea	Martinez	36

Tabla persona

Ilustración 8. Claves Candidatas secuencia

Comentado [OG17]: En caso de requerir los editables del esquema y tabla utilizar recursos anexos. Dato ejemplo 1 y https://drive.google.com/file/d/1W-ggJPezRs7noNMYjiOdAk2c5SMXfB3o/view?usp=sharing Page 4

Este último enfoque es el más empleado y recomendado por los diseñadores o analistas del software, reduce significativamente la complejidad del diseño y facilita la relación con otra tablas o entidades. Por lo tanto, se usará cada vez que se tenga oportunidad de hacerlo.

- **Super clave**
  Es un conjunto de uno o más atributos que, tomados juntos, permiten identificar de forma única una entidad en el conjunto de entidades. Por ejemplo, el atributo **id_persona** del conjunto de entidades persona es suficiente para distinguir una persona cliente del banco de las otras. Así, **id_pesona** es una superclave. Análogamente, la combinación de **identificación** y **país** es una superclave del conjunto de entidades persona. El atributo **correo** de persona es otra superclave, pero el atributo **nombres** no lo es porque varias personas podrían tener el mismo nombre.

- **Clave primaria**
  Se empleará el término clave primaria para nombrar una clave candidata que es elegida por el diseñador o analista de la base de datos como clave principal para identificar las tuplas dentro de una entidad. Una clave (primaria, candidata y superclave) es una propiedad del conjunto de entidades, más que de las entidades individuales. Entre dos tuplas individuales en el conjunto no pueden tener el mismo valor en sus atributos clave primaria al mismo tiempo. La designación de una clave primaria es en sí una restricción que modela una característica del mundo real.

2.2. Relaciones de uno a muchos.

Una relación de uno a varios o de uno a muchos es el tipo de relación más empleada. En este tipo de relación, una fila de la tabla A puede tener muchas filas coincidentes en la tabla B. Pero una fila de la tabla B solo puede tener una fila coincidente en la tabla A. Por ejemplo, las tablas **pesona** y **cuenta** tienen una relación de uno a muchos. Es decir, cada persona puede tener más de una cuenta. Pero cada título proviene de una sola editorial.

En un diagrama de entidad relación esto se representa con lo que se denomina multiplicidad como se muestra en la siguiente ilustración:

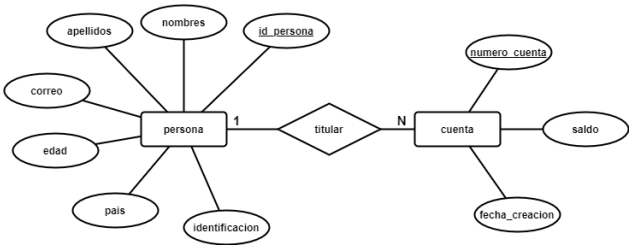


Ilustración 9. Relaciones de uno a muchos

Fuente (https://drive.google.com/file/d/1W-ggJPezRs7noNMYjiOdAk2c5SMXfB3o/view?usp=sharing Page 5)

Comentado [OG18]: En el archivo online se encuentran relacionadas las ilustraciones que expone el experto. En el mismo se específica la hoja donde se encuentra para que pueda ser digitalizada si se requiere.

A esta representación se le nomina multiplicidad **1:N** y se lee de la siguiente forma: **UNA persona** es **titular** de **N cuenta**s bancarias, de esta forma una persona puede tener **N** (que puede ser cero, uno o más de una) cuentas bancarias. Se puede leer en forma inversa que el resultado es el mismo: muchas o varias cuentas bancarias pueden tener un mismo titular.

Cuando una relación de es uno a muchos y se mapea una un modelo de base de datos relacional, la relación **titular** (rombo) se termina convirtiendo una columna de la tabla de los muchos.

id_persona	pais	identificacion	correo	nombres	apellidos	edad
1	colombia	8.127.346	ana@rmail.com	Ana lis	Mendez	34
2	colombia	212.394.762	jose@pmail.com	Jose Omar	Ciano	28
3	colombia	545.232.546	luis@jmail.com	Luis Dario	Marquez	21
4	colombia	125.446.732	arm@hmail.com	Armando	Bronca Segura	35
5	colombia	33.565.236.787	victor@umail.com	Vistor José	Diento Paz	24
6	colombia	2.343.554.768	josen@pmail.com	Jose Nicolás	Timado Ruiz	19
7	perú	45.456.778	andrea@email.com	Andrea	Martinez	36

a) Tabla persona

numero_cuenta	titular	saldo	fecha_creacion
19.283.746	7	$ 2.589.954,00	25/08/1998
67.789.901	1	$ 23.465.212,00	26/08/2001
32.112.312	5	$ 2.343.242,00	27/08/2004
41.928.374	2	$ 2.343.423,00	31/08/2013
38.342.657	6	$ 8.965.211,00	30/08/2010
32.443.171	1	$ 11.172.594,33	22/06/2001
34.920.608	5	$ 14.483.578,83	01/09/2016
17.980.455	3	$ 17.794.563,33	03/09/2019
26.754.825	4	$ 21.105.547,83	12/04/2022

b) Tabla cuenta

Ilustración 10. Tablas de un modelo relacional 1:N

Comentado [OG19]: Las tablas reales se encuentran en los anexos el documento denominado “Datos ejemplo” y la hoja Ejemplo 1

Como se puede ver en la Ilustración 10, Ana Lis Mendez, es titular de dos cuentas bancarias (67.789.901 y 32.443.171) , y todos los datos de la columna titular en la tabla cuenta se conoce como **LLAVE FORANEA** de la tabla persona; de esta forma podemos concluir.

REGLA DE MAPEO 1:N de Modelo entidad-relación a modelo relacional:

Una relación de uno a mucho (1:N) se transforma en una columna en la tabla que tiene la multiplicidad de los muchos (cuenta). Y todos los datos de esta columna existen como llave primaria en la tabla que tiene la multiplicidad uno (tabla persona).

Existe otro tipo de diagrama se usted aprenderá y que en realidad más empleado a la hora de diseñar una base datos, este diagrama se llama Diagrama Relacional, donde las entidades se presenta como tablas de una vez y las relaciones son más claras. Miremos la representación del diagrama entidad relación de la ilustración a un diagrama relacional.

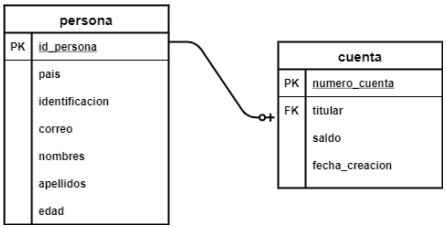


Ilustración 11. Diagrama relacional 1:N.

Fuente (https://drive.google.com/file/d/1W-ggJPezRs7noNMYjiOdAk2c5SMXfB3o/view?usp=sharing Page 6 -2)

Comentado [OG20]: En el archivo online se encuentran relacionadas las ilustraciones que expone el experto. En el mismo se específica la hoja donde se encuentra para que pueda ser digitalizada si se requiere.

En la ilustración 11 se puede ver el nombre de las entidades (persona y cuenta), los atributos haciendo especial énfasis en los que representan la llave primaria con el indicador **PK** del inglés (**P**rimary **K**ey), y también vemos la

relación titular (columna de la tabla), indicando que es una clave o llave foránea de la tabla persona lo indica con el indicador **FK** del inglés (**F**oreign **K**ey).

Algunos diagramas relacionales agregan metadatos para hacer más claro el modelo al diseñador, la naturaleza de los datos del modelo.

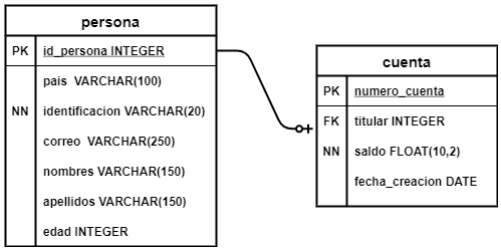


Ilustración 12. Diagrama relacional 1:N con metadatos.

Fuente (https://drive.google.com/file/d/1W-ggJPezRs7noNMYjiOdAk2c5SMXfB3o/view?usp=sharing Page 6 -2)

Comentado [OG21]: En el archivo online se encuentran relacionadas las ilustraciones que expone el experto. En el mismo se específica la hoja donde se encuentra para que pueda ser digitalizada si se requiere.

Conclusión: la relación de uno a muchos se termina convirtiendo en una columna en la tabla que tiene la multiplicidad d ellos muchos.

Comentado [OG22]: Cajón texto color.

2.3. Relaciones de un muchos a muchos.

En una relación de muchos a muchos, una fila de la tabla A puede tener relación con muchas filas en la tabla B y también en sentido contrario una fila de la tabla B con respecto de varias filas de la tabla A. Esta relación se crea definiendo una tercera tabla denominada tabla de relación. Esta nueva tabla tiene una clave principal de la tabla compuesta de las claves foráneas de la tabla A y de la B.

Ejemplo:
Supongamos que necesitamos almacenar la información de unas referencias bibliográficas o publicaciones, de libros, tesis, artículos que escriben los estudiantes de una universidad, y como se sabe un libro, tesis, artículo o cualquier publicación puede ser elaborada por más de una persona, y a su vez, una persona (docente, estudiante o investigador) puede tener más de una publicación de su autoría.

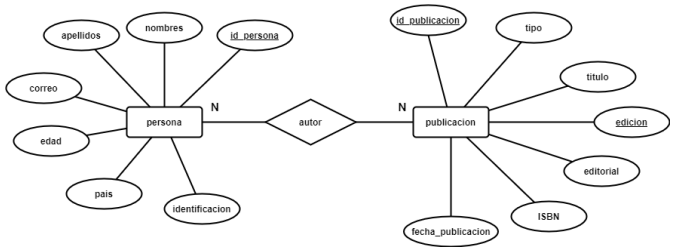


Ilustración 13. Diagrama de entidad de publicaciones bibliográficas.

Fuente (https://drive.google.com/file/d/1W-ggJPezRs7noNMYjiOdAk2c5SMXfB3o/view?usp=sharing Page 7 -2)

Comentado [OG23]: En el archivo online se encuentran relacionadas las ilustraciones que expone el experto. En el mismo se específica la hoja donde se encuentra para que pueda ser digitalizada si se requiere.

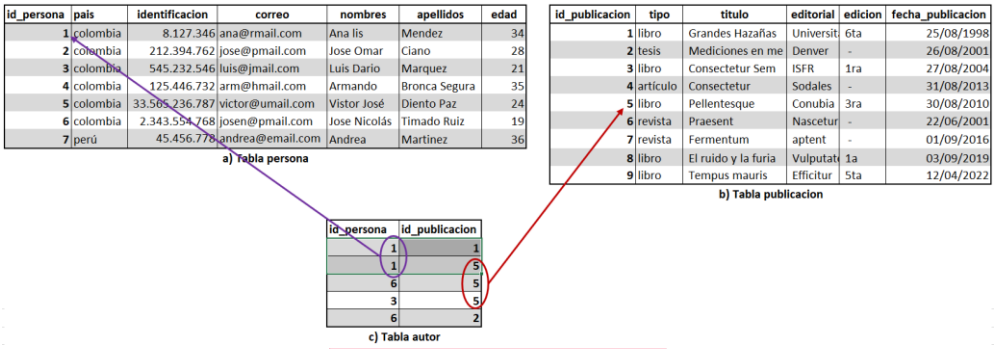
Como se puede ver en la ilustración 13, la entidad publicación tiene una llave primaria llamada identificación, y existe una relaciona llamada **autor**, y representa que unas persona pueden ser autor de una o muchas

publicaciones, y que una publicación puede ser de autoría de una o muchas personas, la multiplicidad en este caso es de muchos a muchos y se representa como N:N.

REGLA DE MAPEO N:N de Modelo entidad-relación a modelo relacional

En una relación de uno a mucho (N:N) se transforma en una tabla cuya llave primaria está compuesta por las llaves primarias de las otras dos tablas.

Como se puede ver en la ilustración 14, la tabla c) autor, nos dice que la persona Ana Lis, es autora de los libros de título Grandes hazañas y Pellentesque, también que el libro Pellentesque es escrito además de Ana Lis por Luis Dario y Jose Nicolas. Lo anterior haciendo referencia a las llaves primarias de las tablas a) **persona** y b) **publicacion**.



a) Tabla persona

id_persona	pais	identificacion	correo	nombres	apellidos	edad
1	colombia	8.127.346	ana@rmail.com	Ana lis	Mendez	34
2	colombia	212.394.762	jose@pmail.com	Jose Omar	Ciano	28
3	colombia	545.232.546	luis@jmail.com	Luis Dario	Marquez	21
4	colombia	125.446.732	arm@hmail.com	Armando	Bronca Segura	35
5	colombia	33.585.236.787	victor@umail.com	Vistor José	Diento Paz	24
6	colombia	2.343.554.768	josen@pmail.com	Jose Nicolás	Timado Ruiz	19
7	perú	45.456.778	andrea@email.com	Andrea	Martinez	36

b) Tabla publicacion

id_publicacion	tipo	titulo	editorial	edicion	fecha_publicacion
1	libro	Grandes Hazañas	Universit	6ta	25/08/1998
2	tesis	Mediciones en me	Denver	-	26/08/2001
3	libro	Consectetur Sem	ISFR	1ra	27/08/2004
4	artículo	Consectetur	Sodales	-	31/08/2013
5	libro	Pellentesque	Conubia	3ra	30/08/2010
6	revista	Praesent	Nascetur	-	22/06/2001
7	revista	Fermentum	aptent	-	01/09/2016
8	libro	El ruido y la furia	Vulputat	1a	03/09/2019
9	libro	Tempus mauris	Efficitur	5ta	12/04/2022

c) Tabla autor

id_persona	id_publicacion
1	1
1	5
6	5
3	5
6	2

Ilustración 14. Tablas de relación N:N.

La representación de la relación N:N con un diagrama relacional podemos se muestra en la ilustración 15. Donde se usan los prefijos **FK**1 y **FK**2 , para indicar Llave foránea uno y dos, para las entidades persona y publicación 2 respectivamente. También se puede ver el prefijo **UK** (**U**nique **K**ey) en la tabla publicación, en la columna ISBN, significa en toda la tabla publicación no puede existir más de una fila con el mismo ISBN es decir, nunca ninguna publicación tendrá el miso ISBN que otra publicación

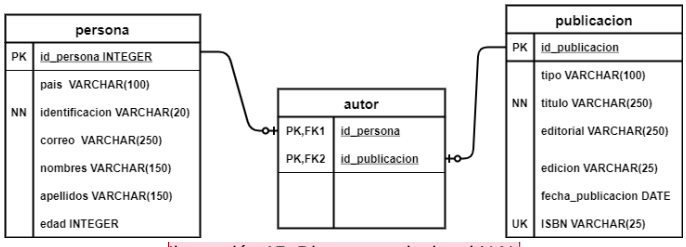


Ilustración 15. Diagrama relacional N:N.

2.4. Relaciones de uno a uno

En una relación uno a uno, una fila de la tabla A solo puede tener una fila coincidente en la tabla B y viceversa. Se crea una relación uno a uno si ambas columnas relacionadas son claves principales o tienen restricciones

Comentado [OG24]: Cajón texto color.

Comentado [OG25]: La ilustración se puede obtener del archivo en anexos “Datos ejemplo” la hoja “Ejemplo (2)”

Comentado [OG26]: El editable se puede obtener del Anexo “base de datos” Page-8

https://drive.google.com/file/d/1W-ggJPezRs7noNMYjiOdAk2c5SMXfB3o/view?usp=sharing

únicas. Este tipo de relación no es común porque la mayoría de la información que está relacionada de esta manera estaría en una tabla. Puede usar una relación uno a uno por alguno de estos motivos:

- Dividir una tabla con muchas columnas.
- Aísle parte de una tabla por motivos de seguridad.
- Almacenar datos de corta duración y que podrían eliminarse fácilmente borrando la tabla periódicamente.
- Almacenar información que solo se aplique a un subconjunto de la tabla principal.

Comentado [OG27]: Listado no ordenado color.

Viviendo al ejemplo del banco, supongamos esta vez, la empresa necesita información sobre sus empleados, que, a la vez, pueden ser clientes de banco.

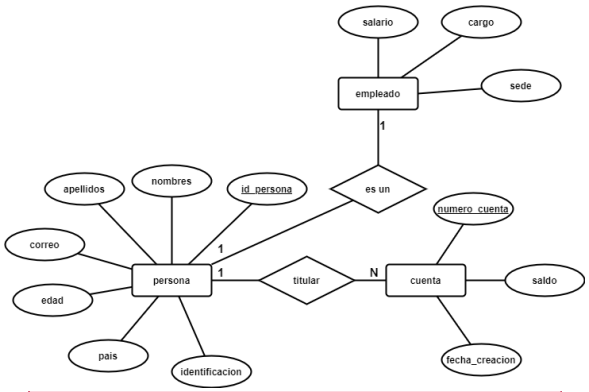


Ilustración 16. Diagrama entidad relación persona-empleado

Comentado [OG28]: El editable se puede obtener del Anexo "base de datos" Page-9

https://drive.google.com/file/d/1W-ggJPezRs7noNMYjiOdAk2c5SMXfB3o/view?usp=sharing

REGLA DE MAPEO 1:1 de Modelo entidad-relación a modelo relacional

En el momento de convertir la relaciono con multiplicidad 1:1 a tablas la relación uno a uno se convierte en una columna de la tabla empleado con restricción de unicidad (**U**nique **K**ey, o **P**rimary **K**ey) como se muestra en el siguiente diagrama relacional.

Comentado [OG29]: Cajón texto color.

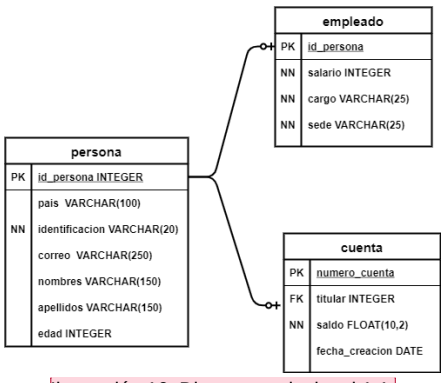


Ilustración 16. Diagrama relacional 1:1.

Comentado [OG30]: El editable se puede obtener del Anexo "base de datos" Page-6

https://drive.google.com/file/d/1W-ggJPezRs7noNMYjiOdAk2c5SMXfB3o/view?usp=sharing

Como se puede inferir de la ilustración 16, se crea una tabla donde se relacionan los datos de las personas que son empleados del banco, a través de una columna que se llama **id_persona**, pero es común entre algunos diseñadores ponerle el nombre que lo relaciones con la tabla a la que perteneces es decir **id_empleado**, pero esto queda al criterio del diseñador.

3. Normalización

La normalización es el procedimiento mediante el cual se aplican las reglas de mapeo o conversión de un modelo entidad-relación a un modelo relacional, es decir a tablas y relaciones, los principios rectores de este proceso son: No redundancia de datos (que no se repitan los datos) y la dependencia de los datos dependencia incoherente (una separación lógica de datos en tablas).

Los datos redundantes desperdician espacio en disco y crean problemas de mantenimiento. Por ejemplo, en la ilustración 16 se ve que los datos comunes de los clientes y los bancos están en la tabla persona, mientras que los datos que pertenecen exclusivamente a los empleados está en otra tabla que se relaciona con la de personas.

¿Qué es una "dependencia incoherente"? Aunque resulta intuitivo para un usuario buscar en la tabla de persona el correo de un cliente en particular, es posible que no tenga sentido buscar en esa columna, el salario del empleado que se relaciona con esos datos de la tabla persona. El salario del empleado está relacionado con la tabla empleado o depende de él. Las dependencias incoherentes pueden dificultar el acceso a los datos, ya que la ruta para encontrar los datos puede faltar o estar dañada.

Existen unas reglas básicas de normalización las más comunes son las 3 formas normales se listan a continuación y se irán desarrollando con un ejemplo más complejo.

Forma Normal	Principios
Primera forma normal	● Eliminar grupos de repetición en tablas individuales. ● Cree una tabla independiente para cada conjunto de datos relacionados. ● Identifique cada conjunto de datos relacionados con una clave primaria.
Segunda forma normal	● Crear tablas independientes para conjuntos de valores que se aplican a varios registros. ● Relacione estas tablas con una clave foránea.
Tercera forma normal	● Eliminar los campos que no dependen de la clave.

Tabla 4. Formas normales.

Hay una cuarta forma normal, también llamada forma normal de *Boyce Codd* (BCNF), y una la quinta forma normal, pero rara vez se consideran en un diseño práctico, solo agregan complejidad al sistema y no aportan un valor funcional que importe a la solución del problema por ello no será tratada.

3.1. Dependencias funcionales.

Una dependencia funcional es un tipo de restricción que termina construyendo una generalización del concepto de clave, como se estudió en el modelo E-R y el modelo relacional. Pero no es tan fácil localizar las dependencias, ya que necesitan de un análisis de los atributos (columnas) o, con más precisión, de las interrelaciones entre atributos, y frecuentemente la intuición no es suficiente a la hora de encontrar y clasificar todas las dependencias.

Aunque existe una teoría matemática para realizar este análisis, nos puede ayudar un ejemplo sencillo nos puede enseñar como analizar estas dependencias de manera intuitiva como se verá a continuación.

Ejemplo:
Un cliente le pide que desarrolle un software para llevar el inventario de una ferretería, los productos, los proveedores, el precio que cada proveedor nos vende cada producto, para lo cual nos da la siguiente tabla, y debemos identificar las dependencias funcionales para poder realizar nuestro modelo relacional.

nit_proveedor	correo	nombres	telefono	codigo_producto	producto	precio
8127346	ana@rmail.com	Ana Lis Mendez	398589854	MS-028	Masillas FFC	$ 25.890,00
212394762	jose@pmail.com	Jose Omar	5475698	PS-012	Pistola de silicona	$ 14.300,00
545232546	luis@jmail.com	Luis Dario	3984558	CT-003	Cutter RDS	$ 8.700,00
8127346	ana@rmail.com	Ana Lis Mendez	398589854	LG-021	Lija C20	$ 820,00
212394762	jose@pmail.com	Jose Omar	5475698	MS-028	Masillas FFC	$ 26.000,00
8127346	ana@rmail.com	Ana Lis Mendez	398589854	LG-021	Lija C20	$ 7.900,00
545232546	luis@jmail.com	Luis Dario	36584114	PS-012	Pistola de silicona	$ 12.990,00
8127346	ana@rmail.com	Ana Lis Mendez	398589854	LG-021	Lija C20	$ 830,00
212394762	jose@pmail.com	Jose Omar	5475698	CT-003	Cutter RDS	$ 9.200,00

Tabla 5. Proveedores, productos y precios.

Comentado [OG31]: La ilustración se puede obtener del archivo en anexos “Datos ejemplo” la hoja 6.

El principio de no repetición de datos dice que debe identificar todos los datos que se repiten y se puede en la siguiente figura, tenido en cuenta los colores de sombreado como se muestra, en la ilustración 17, note que el nit_proveedor son iguales en todos registros que tienen el mismo valor de correo, nombres, y teléfono.

nit_proveedor	correo	nombres	telefono	codigo_producto	producto	precio
8127346	ana@rmail.com	Ana Lis Mendez	398589854	MS-028	Masillas FFC	$ 25.890,00
212394762	jose@pmail.com	Jose Omar	5475698	PS-012	Pistola de silicona	$ 14.300,00
545232546	luis@jmail.com	Luis Dario	3984558	CT-003	Cutter RDS	$ 8.700,00
8127346	ana@rmail.com	Ana Lis Mendez	398589854	LG-021	Lija C20	$ 820,00
212394762	jose@pmail.com	Jose Omar	5475698	MS-028	Masillas FFC	$ 26.000,00
8127346	ana@rmail.com	Ana Lis Mendez	398589854	LG-021	Lija C20	$ 7.900,00
545232546	luis@jmail.com	Luis Dario	36584114	PS-012	Pistola de silicona	$ 12.990,00
8127346	ana@rmail.com	Ana Lis Mendez	398589854	LG-021	Lija C20	$ 830,00
212394762	jose@pmail.com	Jose Omar	5475698	CT-003	Cutter RDS	$ 9.200,00

Ilustración 17. Dependencia de nit_proveedor con datos del proveedor. (GIF 1 ANIMADO)

Comentado [OG32]: La ilustración corresponde al GIF 1. En caso de ajustar la imagen GIF se comparte el PPT generado para la creación de la animación.

Note también en la ilustración 18 como la columna código_producto es el mismo para cada columna producto, pero difiere del precio que cada proveedor ofrece del mismo:

nit_proveedor	correo	nombres	telefono	codigo_producto	producto	precio
8127346	ana@rmail.com	Ana Lis Mendez	398589854	MS-028	Masillas FFC	$ 25.890,00
212394762	jose@pmail.com	Jose Omar	5475698	PS-012	Pistola de silicona	$ 14.300,00
545232546	luis@jmail.com	Luis Dario	3984558	CT-003	Cutter RDS	$ 8.700,00
8127346	ana@rmail.com	Ana Lis Mendez	398589854	LG-021	Lija C20	$ 820,00
212394762	jose@pmail.com	Jose Omar	5475698	MS-028	Masillas FFC	$ 26.000,00
8127346	ana@rmail.com	Ana Lis Mendez	398589854	LG-021	Lija C20	$ 7.900,00
545232546	luis@jmail.com	Luis Dario	36584114	PS-012	Pistola de silicona	$ 12.990,00
8127346	ana@rmail.com	Ana Lis Mendez	398589854	LG-021	Lija C20	$ 830,00
212394762	jose@pmail.com	Jose Omar	5475698	CT-003	Cutter RDS	$ 9.200,00

Ilustración 18. Dependencia de código_producto_ con datos del producto. (GIF 2 ANIMADO)

Comentado [OG33]: La ilustración corresponde al GIF 2. En caso de ajustar la imagen GIF se comparte el PPT generado para la creación de la animación.

La dependencia funciona código_producto no se cumple para precio del producto, por lo tanto, dice que a cada proveedor y producto debe existir un precio distinto como se ve la siguiente ilustración.

nit_proveedor	correo	nombres	telefono	codigo_producto	producto	precio
8127346	ana@rmail.com	Ana Lis Mendez	398589854	MS-028	Masillas FFC	$ 25.890,00
212394762	jose@pmail.com	Jose Omar	5475698	PS-012	Pistola de silicona	$ 14.300,00
545232546	luis@jmail.com	Luis Dario	3984558	CT-003	Cutter RDS	$ 8.700,00
8127346	ana@rmail.com	Ana Lis Mendez	398589854	LG-021	Lija C20	$ 820,00
212394762	jose@pmail.com	Jose Omar	5475698	MS-028	Masillas FFC	$ 26.000,00
8127346	ana@rmail.com	Ana Lis Mendez	398589854	LG-021	Lija C20	$ 7.900,00
545232546	luis@jmail.com	Luis Dario	36584114	PS-012	Pistola de silicona	$ 12.990,00
8127346	ana@rmail.com	Ana Lis Mendez	398589854	LG-021	Lija C20	$ 830,00
212394762	jose@pmail.com	Jose Omar	5475698	CT-003	Cutter RDS	$ 9.200,00

Ilustración 19. Dependencia de nit_proveedor, código_producto_ con precio del producto. (GIF 3 ANIMADO)

Comentado [OG34]: La ilustración corresponde al GIF 3. En caso de ajustar la imagen GIF se comparte el PPT generado para la creación de la animación.

En la siguiente tabla se observan las dependencias detectadas:

Atributo	Es dependencia funcional de:
nit_proveedor	• correo • nombres • teléfono
codigo_pruducto	• producto
nit_proveedor, codigo_producto	• precio

Tabla 5. Dependencias funcionales

Definición formal DEPENDENCIA FUNCIONAL

Se dice que un atributo X de una relación «depende funcionalmente» de otro atributo o conjunto de tributos Y de la relación si a todo valor (o valores del conjunto) Y le corresponde siempre el mismo valor de X.

Comentado [OG35]: Cajón texto color.

Las dependencias funcionales de la tabla 5 se pueden representar en modelo entidad relación, teniendo en cuanta que un proveedor, puede proveedor muchos productos y a su vez un producto puede ser proveedor por más de un proveedor, y que por cada producto que es proveedor por un proveedor existe un precio.

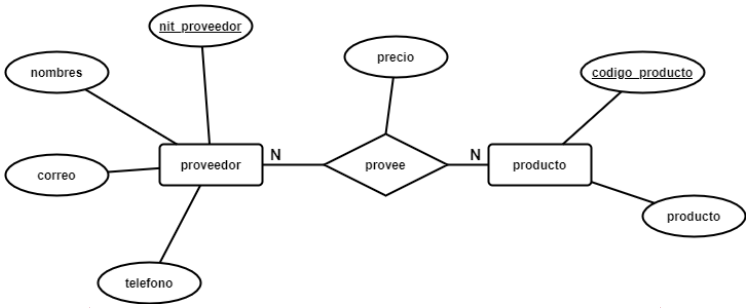


Ilustración 19. Diagrama entidad relación dependencia funcional.

Como se puede notar en la ilustración 19, la relación provee, cuando es mapeado a un diagrama relacional, se genera una tabla intermedia provee que debe tener un atributo adicional “precio”.

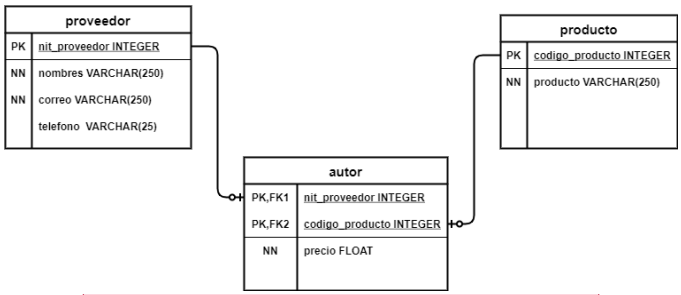


Ilustración 19. Diagrama relacional dependencia funcional.

3.2. Diseño relacional.

Hasta este punto se han examinado problemas concretos de las formas normales muy básicas y de la normalización. Ahora se estudiará el modo en que se encaja la normalización en proceso general de diseño de bases de datos se puede hacer de una de esas formas:

1. Convirtiendo un diagrama entidad relación a un diagrama relacional y de esa forma obtener las tablas.
2. Puede tener una tabla con todos los datos que se procesan en una aplicación e identificado las dependencias funcionales de manera que nos permite identificar las tablas y las llaves o claves primarias de cada una.
3. Se puede hacer un diseño ad hoc (para que cumpla el requerimiento) y se debe comprobar luego que satisface la forma normal deseada.

En el mundo practico y en la medida que se tiene más experticia las opciones 2 y 3 son las más empleadas.

Comentado [OG36]: El editable se puede obtener del Anexo “base de datos” Page-11

https://drive.google.com/file/d/1W-ggJPezRs7noNMYjiOdAk2c5SMXfB3o/view?usp=sharing

Comentado [OG37]: El editable se puede obtener del Anexo “base de datos” Page-12

https://drive.google.com/file/d/1W-ggJPezRs7noNMYjiOdAk2c5SMXfB3o/view?usp=sharing

Comentado [OG38]: Listado ordenado cuadro color.

Veamos un ejemplo completo de cómo abordar un problema de almacenamiento de datos y transformarlo en un modelo relacional, recogiendo los conceptos vistos hasta ahora, pasado de un diagrama entidad relación a un modelo relacional que es en todo caso el objetivo general de un diseño de base de datos:

Problema:
Se necesita de gestionar una biblioteca escolar, y nuestro cliente quiere tener ciertas herramientas a su disposición para controlar publicaciones tales como libros, revistas, folletos; también llevar un registro de los estudiantes y préstamos. Adicionalmente se necesita un control de los ejemplares de cada libro o publicación, su ubicación y su estado, registrar la retirada de libros por préstamo y restitución de ejemplares, para esto necesita información sobre editoriales datos como dirección, teléfono y nombre de la editorial.

De cada publicación se pueden tener varios ejemplares, por ejemplo: del libro “nacho lee” la biblioteca tiene 5 ejemplares, también de cada publicación se debe conocer, su autor o autores, el tema que trata la publicación (geografía, fisca, matemática, etc..), la editorial. Cada libro, tiene un título, un idioma, y formato (digital web, impreso, digital CD), el código ISBN y fecha de publicación.

Cada ejemplar tiene un código de barras, cuando un ejemplar ya está muy descargado se debe poder dar de baja es decir que ya se puede botar y no usarlo más.

Tanto el estudiante, como el auto de un libro tiene los siguientes datos: nombres, apellidos, correo. Adicional a un estudiante tiene los datos de documento de identidad, teléfono, dirección, grado escolar.

De cada préstamo interesa saber que ejemplar se prestó, fecha en la que se prestó, la fecha en que se devolvió y alguna nota o comentario.

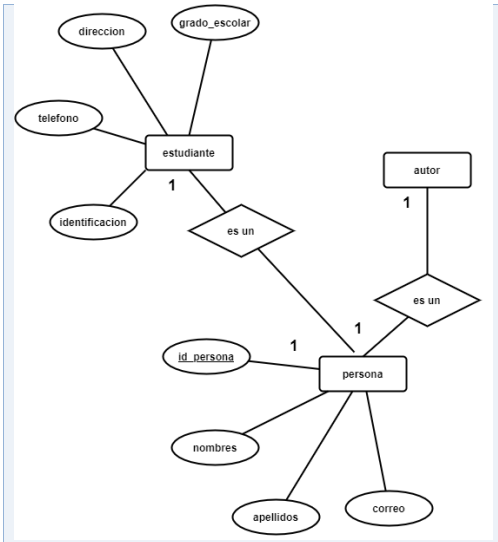


Ilustración 20. Análisis de relacionar autor

En este diagrama se presenta una posible solución analicémosla.
La relación entre persona y estudiante está muy bien corresponde al caso estudiado en las relaciones de uno a uno (entre persona y empleado del banco) de la ilustración 16.

El tema se pone complejo porque debemos analizar si aplica lo mismo para autor:

Considere que al mapear la estas entidades a diagrama relacional me resultan 3 tablas (estudiante, persona y autor), pero autor solo tendría una única columna y al no tener atributos adicionales relacionados con el autor es prácticamente lo mismo que tener la tabla persona, por lo tanto no es conveniente dejarlo de esta forma y más bien relacionar la publicación directamente con la entidad persona.

Estudie detenidamente el diagrama entidad relación, por varios minutos, asegúrese que comprende el porqué de cada elemento y su multiplicidad. Antes de continuar, realice a mano alzada diagrame el modelo relacional:

Comentado [OG39]: El editable se puede obtener del Anexo “base de datos” Page-15

https://drive.google.com/file/d/1W-ggJPezRs7noNMYjiOdAk2c5SMXfB3o/view?usp=sharing

Recuerde:
Una relación de muchos a muchos (N:N) se convierte en una tabla cuya llave primaria es computara por las llaves foráneas de las otras dos tablas, Una relación de uno a muchos (1:N) se convierte en una columna (llave foránea) de la tabla que tiene los muchos, y finalmente una relación de uno a uno (1:1) se convierte una columna en cualquier (llave foránea) de las dos tablas.

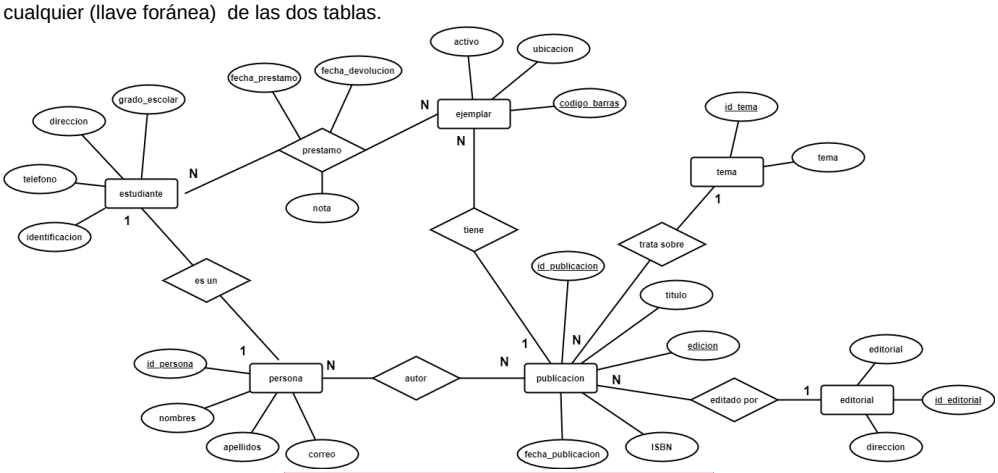


Ilustración 21. Modelo entidad relación biblioteca

Comentado [OG40]: El editable se puede obtener del Anexo "base de datos" Page-7

https://drive.google.com/file/d/1W-ggJPezRs7noNMYjiOdAk2c5SMXfB3o/view?usp=sharing

En la siguiente ilustración se presenta el modelo en un diagrama relacional, este diagrama es importante ya que hace parte de uno de los subproductos que por petición del cliente muchas veces es el un entregable en un proyecto de software, no es así con el diagrama entidad relación ya que este tipo de diagrama requiere más formas y contiene menos tecnicismos.

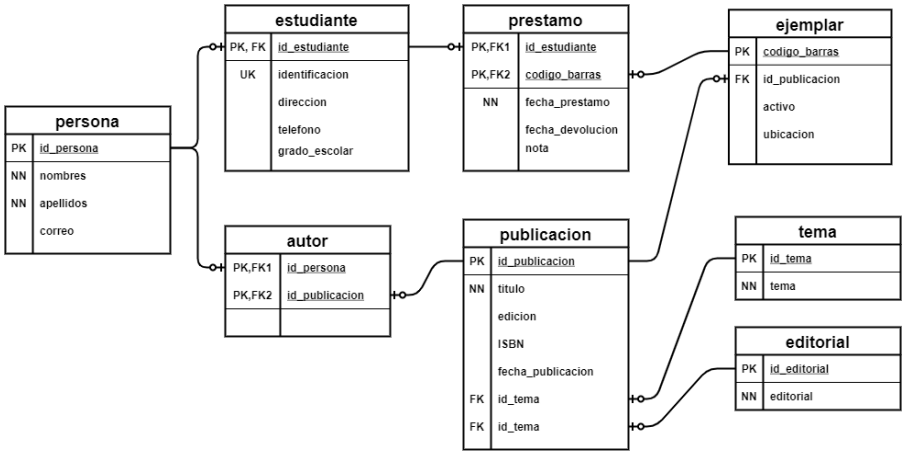


Ilustración 22. Modelo relacional

Comentado [OG41]: El editable se puede obtener del Anexo “base de datos” Page-13

https://drive.google.com/file/d/1W-ggJPezRs7noNMYjiOdAk2c5SMXfB3o/view?usp=sharing

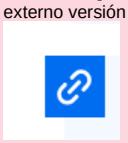
Hasta ahora se ha visto cómo distribuir el almacenamiento de los datos en tablas, de manera que no permita la duplicidad de datos y procurando una estructura lógica, que presente la información del mundo real. Pero aún falta analizar el problema desde el punto de vista que permita garantizar que los datos que se van a insertar son correctos. Para lo cual es importante conocer las restricciones de integridad que pude tener una base de datos.

3.3. Reglas de integridad.

Cuando se define un atributo si saber lo estamos escogiendo para él un dominio es decir un conjunto de valores que puede tener. Por ejemplo: en MySQL cuando escogemos un atributo de tipo entero (INT) los posibles valores que puede tener esa columna van desde -2147483648 hasta 2147483647, es decir valores positivo u negativos, de manera que si queremos solo valores positivos debemos definirlo como entero sin signo (UNSIGNED INT) por lo tanto los valores van desde 0 hasta 4294967295, vea un dato completo en el siguiente enlace

Tipos de enteros: https://dev.mysql.com/doc/refman/8.0/en/integer-types.html

Comentado [OG42]: Llamado a la acción de recurso externo versión simplificada.



La calidad de los datos garantiza que los datos almacenados en su base de datos cumplan con los estándares y requisitos de la organización. En otras palabras, asegura el mantenimiento de la integridad de los datos en una base de datos. Al hacerlo, aplica un conjunto de reglas a un conjunto de datos completo o específico y lo almacena en la base de datos de destino por lo consiguiente para garantizar la integridad se debe tener en cuenta:

- **Criterio de NULIDAD**

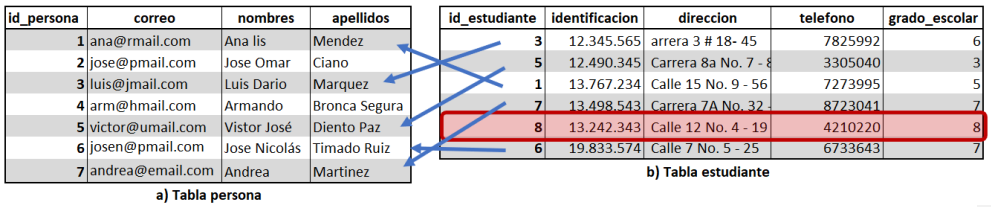
Cuando en una fila un atributo (columna) es desconocido, se dice que es nulo. Un nulo no representa el valor cero (0) ni una cadena vacía (“”) ya que éstos son valores tienen un significado o valor. El nulo significa ausencia de información, bien porque al insertar la fila se desconocía el valor del atributo, o tal vez porque para dicha fila el atributo no tiene valor. Debido a que los nulos no son valores, deben tratarse de forma particular, lo que causa problemas de implementación. Actualmente casi todos los SGDB soportan valores nulos.

- **Integridad de entidad**

Ninguno de los atributos o columnas que componen una llave primaria debe ser NULO. Es decir, una clave prima es irreducible para identificar de modo único una fila, el que sea irreducible significa que ningún subconjunto de la clave primaria sirve para identificar inequívocamente la dial. Si se permitirá valor nulo en una columna de hace parte de la llave primaria, esta estaría contradiciendo la irreductibilidad de una llave primaria. Tenga en cuenta que esto no aplica para claves alternativas y solo para la clave primaria.

- **Integridad referencia.**

Analice el modelo de la ilustración 22. En particular la relación entre las tablas persona y estudiante. Como se sabe, la tabla estudiante tiene una llave primaria que a la vez es una referencia de la tabla persona, mire un ejemplo en la siguiente ilustración.



id_persona	correo	nombres	apellidos
1	ana@rmail.com	Ana lis	Mendez
2	jose@pmail.com	Jose Omar	Ciano
3	luis@jmail.com	Luis Dario	Marquez
4	arm@hmail.com	Armando	Bronca Segura
5	victor@umail.com	Vistor José	Diento Paz
6	josen@pmail.com	Jose Nicolás	Timado Ruiz
7	andrea@email.com	Andrea	Martinez

a) Tabla persona

id_estudiante	identificacion	direccion	telefono	grado_escolar
3	12.345.565	arrera 3 # 18- 45	7825992	6
5	12.490.345	Carrera 8a No. 7 - 8	3305040	3
1	13.767.234	Calle 15 No. 9 - 56	7273995	5
7	13.498.543	Carrera 7A No. 32 -	8723041	7
8	13.242.343	Calle 12 No. 4 - 19	4210220	8
6	19.833.574	Calle 7 No. 5 - 25	6733643	7

b) Tabla estudiante

Ilustración 23. Error de integridad referencial

Como siguiente la ilustración 23, por cada estudiante (tabla estudiante) existe uno y solo un elemento en la tabla persona, de tal forma que un estudiante es la conjunción de ambas tablas por medio de la llave foránea (que a la vez es llave primaria) id_estudiante, ¿pero ¿qué tal que no existiera?, eso significa que la base de datos es inconsistente o que tiene datos basura en la tabla estudiante. A afortunadamente los SGDB permite definir reglas que impidan que esto ocurra en las tablas, es decir que no permita registrar un estudiante con id_estudiante con un valor que no existe en la columna id_persona de la tabla persona.

Comentado [OG43]: Realizar acordeón con las reglas de integridad:
- **Criterio de NULIDAD (+)**
- **Integridad de entidad (+)**
- **Integridad referencia. (+)**

Pregunta de análisis de datos

Ahora pregúntese lo siguiente: ¿qué pasaría si se borra el registro de la tabla persona que tiene id_persona con valor tres (3) sufrirá algún problema los datos de la tabla estudiante?, o que pasaría si se cambia el valor de tres (3) en la columna id_persona, por otro valor como por ejemplo nueve (9). antes de continuar tome unos minutos para pensarlo.

Comentado [OG44]: Bloque de texto destacado.

RESPUESTA

La respuesta a las cuestiones planeadas es que en cualquiera de las operaciones de borrado o actualización del dato se incurrirá en una violación de la integridad referencial de la tabla estudiante, ya que no habrá manera de conocer los datos de complementarios del estudiante que se encuentran en la tabla persona.

Para evitar estos problemas de insistencia los sistemas gestores de bases de datos permiten definir unas reglas cuando se borra, o actualiza un dato que es referencia foránea de otra tabla.

Regla de borrado: define el comportamiento a la pregunta ¿Qué ocurre si se intenta borrar la fila referenciada por la clave foránea de otra tabla? Para lo cual el SGBD implementa una de las siguientes opciones:

- Restringir: no se permite borrar la fila referenciada.
- Cascada: se borra la fila referenciada y también se borran las filas que la referencian mediante la clave foránea en la otra tabla.
- Poner null: se borra la fila referenciada y las filas que la referenciaban ponen a nulo la clave foránea (sólo si acepta nulos).
- Valor por defecto: se borra la fila referenciada y las filas que la referenciaban ponen en la clave foránea el valor por defecto defina la regla.

Regla de edición: define el comportamiento a la pregunta: ¿Qué sucede si se intenta modificar el valor de la clave primaria de la fila referenciada por la clave foránea por otra tabla?

- Restringir: no se permite editar la fila referenciada.
- Cascada: se actualiza la fila referenciada y también se actualiza las filas que la referencian mediante la clave foránea en la otra tabla.

- Poner null: se actualiza la fila referenciada y las filas que la referenciaban ponen a nulo la clave foránea (sólo si acepta nulos).
- Valor por defecto: se edita la fila referenciada y las filas que la referenciaban ponen en la clave foránea el valor por defecto defina la regla.

Comentado [OG45]: Usar tarjetas o dos tabs para resaltar las dos reglas:
✓**Regla borrado**
✓**Regla edición**

De esta forma una vez definir la estructura del modelo de datos de datos, se debe determinar el comportamiento de estas operaciones, lo cual generalmente responde a requerimiento del sistema de información.

3.4. Lenguajes de los sistemas administradores de bases de datos.

Los sistemas gestores de bases de datos emplean un lenguaje que se denomina SQL que corresponde al nombre en inglés (**S**tructured **Q**uery **L**anguage), Generalmente, cuando un SGBD relacional implementa el lenguaje SQL, todas las acciones, y operaciones se llevan a cabo en el sistema mediante sentencias de este lenguaje. Dentro de SQL hay varios tipos de sentencias que se agrupan en cuanto conjuntos:

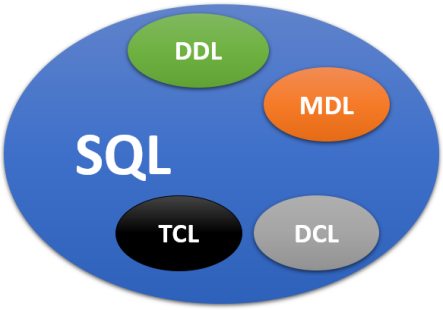


Imagen. Conjunto bases de datos SQL

Comentado [OG46]: Rediseñar imagen. https://images.app.goo.gl/Nu6rPypE2MnAhUm79

- ✓ **Sentencias de definición de datos (DDL):** son las sentencias que permiten crear tablas, alterar su definición y eliminarlas. En una base de datos relacional existen otros tipos de objetos además de las tablas, como las vistas, los índices y los disparadores. Las sentencias para crear, alterar y eliminar vistas e índices también pertenecen a este conjunto de sentencia se les denomina **DDL** del inglés **D**ata **D**efinition **L**anguage.

Ejemplo DDL

Para crear la tabla persona de la ilustración 22. Quedaría de la siguiente forma:

```
CREATE TABLE persona (
     id_persona   INT NOT NULL ,
     nombres      VARCHAR(250) NOT NULL ,
     apellidos      VARCHAR(250) NOT NULL ,
     correo          VARCHAR(250) NOT NULL ,
      PRIMARY KEY (id_persona)
  );
```

Note lo siguiente: cada atributo se define en una línea separado por una coma, se agregan restricciones de no nulidad (NOT NULL) al final la sentía termina con punto y coma. Ahora mire como se borraría esa misma tabla:

```
DROP TABLE persona;
```

Comentado [OG47]: Utilizar acordeón para ocultar el ejemplo y que el aprendiz interactúe para visualizarlo.

Ejemplo DDL (+)

- ✓ **Sentencias de manejo de datos (DML):** etas sentencias que permiten insertar datos en las tablas, consultarlos, editarlos y borrarlos, se denomina **DML** del inglés **D**ata **M**anipulation **L**anguage.

Ejemplo DML

Veamos cómo se inserta una fila a la tabla persona, luego como se actualiza la fila insertada y posteriormente como se borra:

```
INSERT INTO persona (id_persona, nombres, apellidos, correo)
VALUES (1, 'Ana lis','Mendez','ana@rmail.com');
```

```
UPDATE persona SET nombres = 'Ana Lis' WHERE id_persona = 1;
```

```
DELETE FROM persona WHERE id_persona = 1;
```

Comentado [OG48]: Utilizar acordeón para ocultar el ejemplo y que el aprendiz interactúe para visualizarlo.

Ejemplo DML (+)

- ✓ **Sentencias de control (DCL):** son las sentencias empleadas por los administradores de la base de datos para realizar tareas como, por ejemplo, crear usuarios y concederles o revocar los privilegios. Se usa el término **DCL** del inglés **D**ata **C**ontrol **L**anguage.

Ejemplo DCL

En el ejemplo se le da permisos a un usuario llamado **userdb** de la base de datos de nombre biblioteca_db para que pueda insertar, actualizar, borrar y consultar datos de la tabla persona desde el mismo computador donde está instalada la base de datos (localhost).

```
GRANT INSERT, UPDATE, DELETE, SELECT ON biblioteca_db.persona TO
userdb@'localhost';
```

Comentado [OG49]: Utilizar acordeón para ocultar el ejemplo y que el aprendiz interactúe para visualizarlo.

Ejemplo DCL (+)

- ✓ **Sentencias de control de transacciones (TCL):** un pequeño grupo de sentencias que permiten procesar en bloque operaciones **DML** garantizando que se efectúen todas y cada una de las operaciones o ninguna. **TCL** del inglés **T**ransaction **C**ontrol **L**anguage.

4. Sistema gestor de base de datos

En el mercado existen muchos sistemas gestores de bases de datos y a continuación se usará MySQL para realizar las prácticas, pero ello no significa ni que sea el mejor, o el más completo del mercado, sino más bien el más usado para aplicaciones web, de pequeña, mediana y algunos casos de gran complejidad. Es un buen punto de partida y todos los conceptos en MySQL vistos son reutilizables en PosgresSQL, Oracle, u otro motor de bases de datos relacional.

A continuación, podrá visualizar el paso a paso de la descarga e instalación del gestor MySQL Workbench y Web XAMPP:

VIDEO DESCARGA E INSTALACIÓN de herramienta CASE MySQL Workbench.

Comentado [OG50]: Los vídeos se disponen en la carpeta anexos y lo ideal es generar el intro y outro y agregar los datos del experto que genera los vídeos:
Peter Emerson Pinchao Solis
Experto temático
Regional Cauca, Centro de teleinformática y producción industrial

Si es posible crear carousel con los dos vídeos generados por el experto para la descarga e instalación de **Workbech y XAMPP** de lo contrario dos modals para la visualización y reproducción.

VIDEO DESCARGA E INSTALACIÓN de entorno desarrollo WEB XAMPP.

Finalmente revise cuidadosamente cómo usar MySQL Workbench para diseñar la base de datos del Modelo relacional de la biblioteca del numeral 3.2 y la ilustración 22:

VIDEO DISEÑO DE UNA BASE DATOS con MySQL Workbench relación de uno a uno

VIDEO DISEÑO DE UNA BASE DATOS con MySQL Workbench relación de uno a muchos

VIDEO DISEÑO DE UNA BASE DATOS con MySQL Workbench relación de muchos a muchos

Comentado [OG51]: Los vídeos se disponen en la carpeta anexos y lo ideal es generar el intro y outro y agregar los datos del experto que genera los vídeos:
Peter Emerson Pinchao Solis
Experto temático
Regional Cauca, Centro de teleinformática y producción industrial

Si es posible crear carousel o modal con los tres vídeos generados por el experto para el uso del gestor Workbench en la relación:
- ✓**Uno a uno**
- ✓**Uno a muchos**
- ✓**Muchos a muchos**

C. ACTIVIDADES DIDÁCTICAS (OPCIONALES SI SON SUGERIDAS)

N/A

D. MATERIAL COMPLEMENTARIO:

Referencia APA del Material	Tipo de material (Video, capítulo de libro, articulo, otro)	Enlace del Recurso o Archivo del documento o material
Espinosa, A. P. (2014, Nov. 16). L6 1 Nociones y modelos de bases de datos. Youtube.	Vídeo	https://www.youtube.com/watch?v=pPATLxijDfw
Silberschatz, A., Korth, H. F., Sudarshan, S., Pérez, F. S., Santiago, A. I., & Sánchez, A. V. (2006). Fundamentos de bases de datos.	PDF	http://ceneval.isi.uson.mx/CENEVAL/4Implantacion%20de%20infraestructura/Fundamentos%20de%20base%20de%20datos.pdf

E. GLOSARIO:

TERMINO	SIGNIFICADO
Base datos	Conjunto ordenado y finito de operaciones que permite hallar la solución de un problema.
Metadatos	Conjunto unitario de instrucciones que permite a una computadora realizar funciones diversas, como el tratamiento de textos, el diseño de gráficos, la resolución de problemas matemáticos, el manejo de bancos de datos
Diccionario de datos	Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora.

F. REFERENCIAS BIBLIOGRÁFICAS:

Asale, R.-, & Rae. (2021). base | Diccionario de la lengua española. https://dle.rae.es/base#CiiosqO

Kyocera, I. Conceptos sobre base de datos orientada a objetos. (2021). https://rb.gy/5deqdh
López Juan Carlos, 2009, Algoritmos y programación (guía para docentes). Eduteka http://eduteka.icesi.edu.co/articulos/GuiaAlgoritmos
Silberschatz, A., Korth, H. F., Sudarshan, S., Pérez, F. S., Santiago, A. I., & Sánchez, A. V. (2006). Fundamentos de bases de datos. http://ceneval.isi.uson.mx/CENEVAL/4Implantacion%20de%20infraestructura/Fundamentos%20de%20base%20de%20datos.pdf
Resnick, Mitchel (2007): Sembrando las semillas para una sociedad más creativa. Laboratorio de medios de MIT, Massachussets. Eduteka. http://www.eduteka.org/ScratchResnickCreatividad.php
Maryqiu1987. (2021). Descripción de normalización de base de datos. https://docs.microsoft.com/es-es/office/troubleshoot/access/database-normalization-description

G. CONTROL DEL DOCUMENTO

	Nombre	**Cargo**	**Dependencia** *(Para el SENA indicar Regional y Centro de Formación)*	**Fecha**
Autor (es)	Henry Eduardo Bastidas Paruma	Experto temático	Regional Cauca, Centro de teleinformática y producción industrial	Mayo 2021
	Peter Emerson Pinchao Solis	Experto temático	Regional Cauca, Centro de teleinformática y producción industrial	Mayo 2021
	Oscar Absalón Guevara	Diseñador instruccional	Centro de Gestión Industrial	Mayo 2021
	Rafael Neftalí Lizcano Reyes	Asesor Pedagógico Ecosistema RED	Centro Industrial del Diseño y la Manufactura - Regional Santander	

H. CONTROL DE CAMBIOS

(Diligenciar únicamente si realiza ajustes a la Unidad Temática)

	Nombre	**Cargo**	**Dependencia**	**Fecha**	**Razón del Cambio**
Autor (es)					

VUELTA CICLISTA

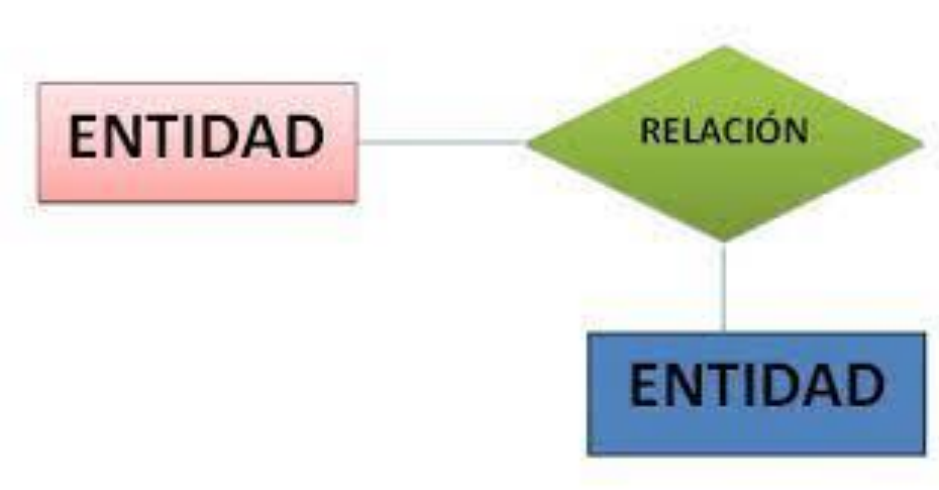
Se requiere diseñar una base de datos sobre una vuelta ciclista. En la vuelta participan una serie de equipos, que están formados por los ciclistas y un director deportivo. Cada ciclista tiene asignado un dorsal, y nos gustaría almacenar en la base de datos su nombre, edad y nombre del equipo al que pertenece. Un ciclista sólo puede pertenecer a un equipo. Los ciclistas disputan una serie de etapas. Cada etapa logra ganarla sólo un ciclista. De estas etapas nos interesa conocer su número (las etapas se numeran consecutivamente: 1, 2, los kilómetros que se recorren, el nombre de la población de salida y el de la población donde está la llegada o meta de la etapa.

Las etapas pueden tener puertos de montaña. Un puerto no puede aparecer en más de una etapa. De los puertos nos interesa conocer su nombre, el porcentaje que indica la pendiente media del puerto, la altura máxima y la categoría (primera, especial, etc.). Los ciclistas pueden ganar puertos, y un puerto sólo lo puede ganar un ciclista, que será el primero que lo alcance.

También nos interesa saber los maillots que puede llevar un ciclista en una etapa. Estos maillots otorgan una serie de premios a los ciclistas que los llevan al finalizar la vuelta. Las características más importantes que nos interesan de un maillot son: su código, tipo o clasificación que premia ese maillot (general, montaña, etc.). color de la camiseta asociada e importe del premio que corresponde al ciclista que termine la vuelta llevando el maillot. Los ciclistas pueden llevar uno o varios maillots (aunque sólo se pongan uno) en una misma etapa.

GC-F-004v.04

Modelo Entidad Relación E-R



Instructor: Gustavo Salazar Escobar

Análisis y Desarrollo de Software ADSO

1- Competencia 220501095 - Diseñar la solución de software de acuerdo con procedimientos y requisitos técnicos.

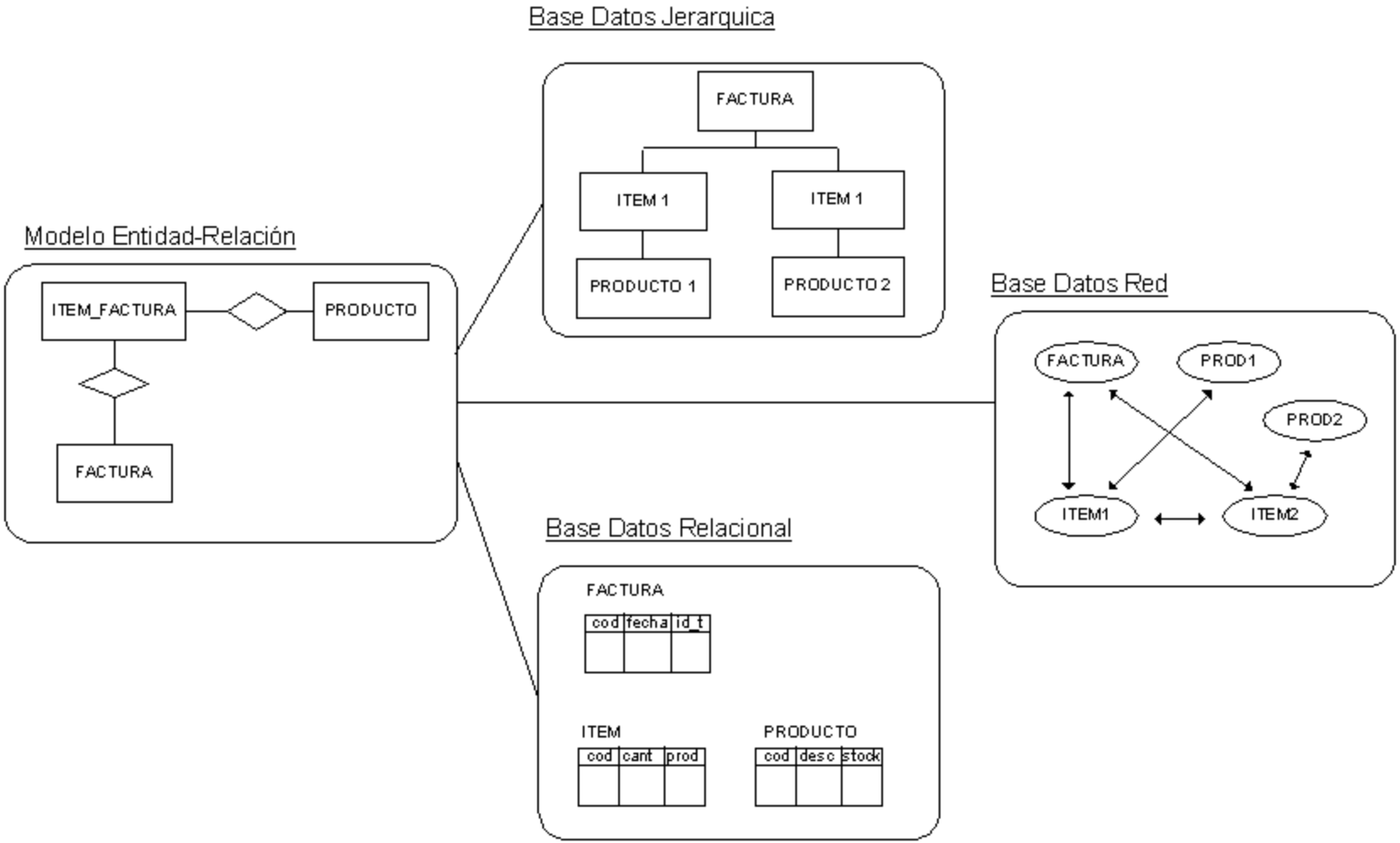
Actividad de Aprendizaje GA4-220501093-AA1 Diseñar la estructura de la base de datos del sistema.

Análisis y Desarrollo de Software ADSO

ACTIVIDAD DE APRENDIZAJE	RESULTADO DE APRENDIZAJE (RAP)	Evidencia a Aprobar	Fecha de Entrega
GA4-220501093- AA1 Diseñar la estructura de la base de datos del sistema. ***GA4-220501095-AA1 Diseñar la estructura de la base de datos del sistema***	**220501095-02 -** Estructurar el modelo de datos del *software* de acuerdo con las especificaciones del análisis.	**GA4-220501093-AA1- EV01 Identificar entidades y relaciones planteadas en el caso de estudio** **GA4-220501095-AA1- Diseñar el EV01 - modelo conceptual y lógico de acuerdo al caso de estudio** **GA4-220501095-AA1- conceptual EV02 - Modelos y lógico para el proyecto desarrollo de *software***	06/09/2022 a 27/09 /2022

MER (Modelo Entidad Relación)

El modelado de datos es independiente del hardware o software usado para su implementación. Un modelo Entidad - Relación, puede ser implementado en bases de datos jerárquicas, red o relacionales.


Base Datos Jerarquica
FACTURA
ITEM 1
ITEM 1
PRODUCTO 1
PRODUCTO 2
Modelo Entidad-Relación
ITEM_FACTURA
PRODUCTO
FACTURA
Base Datos Red
FACTURA
PROD1
PROD2
ITEM1
ITEM2
Base Datos Relacional
FACTURA
cod
fecha
id_t
ITEM
cod
cant
prod
PRODUCTO
cod
desc
stock

Conceptos claves



Entidades

- **Una entidad es un objeto que tiene significado o importancia, cuya información se necesita conocer.**
- **Un objeto de interés al negocio**
- **Una clase o categoría de algo**
- **El nombre de una cosa**

Entidades Ejemplo:

- En el contexto de administración de personal de una empresa las entidades pueden ser:

- Empleado

- Departamento

- Proyecto

Atributos

Describen las entidades y son las piezas específicas de información que se desea conocer

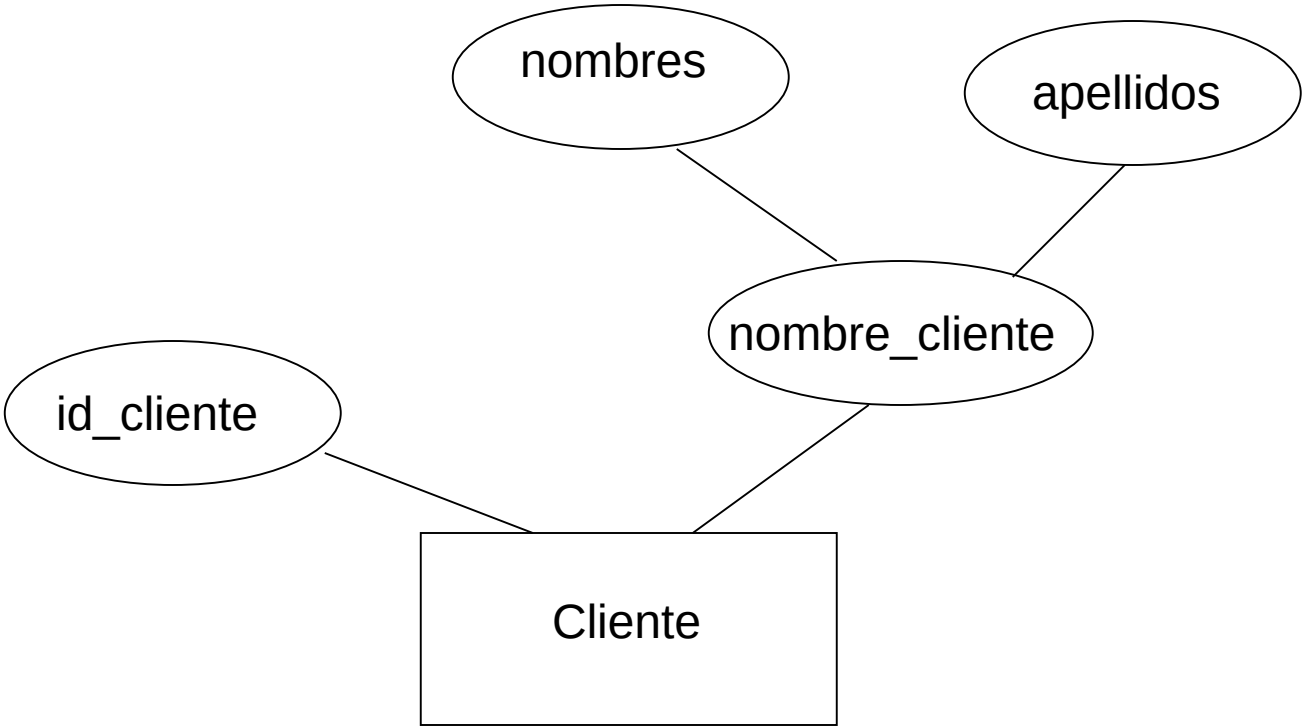
- **Por ejemplo, posibles atributos para la entidad empleado serían:**
- **-Nombre**
- **-Fecha nacimiento**
- **-Sueldo**
- **-Rut**

Atributos

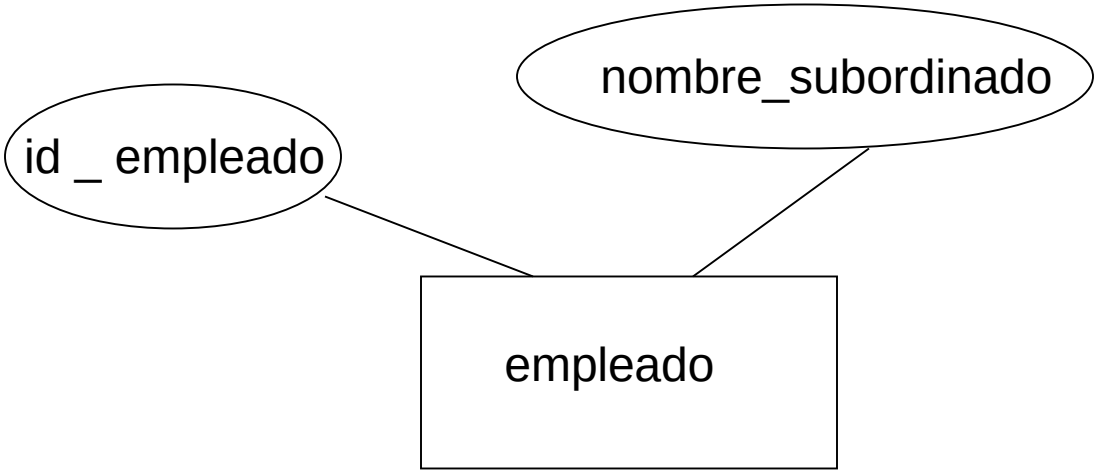
Para cada atributo hay un conjunto de valores permitidos, llamados el dominio o el conjunto de valores de ese atributo.
Tipos de atributos:

Simples y compuestos
Univalorados y multivalorados
Nulos
Derivados

Atributos simples y compuestos



Atributos univalorados y multivalorados



id_empleado es univalorado

nombre_subordinado es multivalorado

Atributos Nulos

Se usan cuando una entidad no tiene valor para un atributo

Puede indicar que el valor del atributo es desconocido

Un atributo desconocido puede ser bien perdido (el valor existe pero no se dispone de la información) o no conocido (no se sabe realmente si el valor existe o no)

Atributos Derivados

Son atributos que se obtienen a partir de otros atributos.

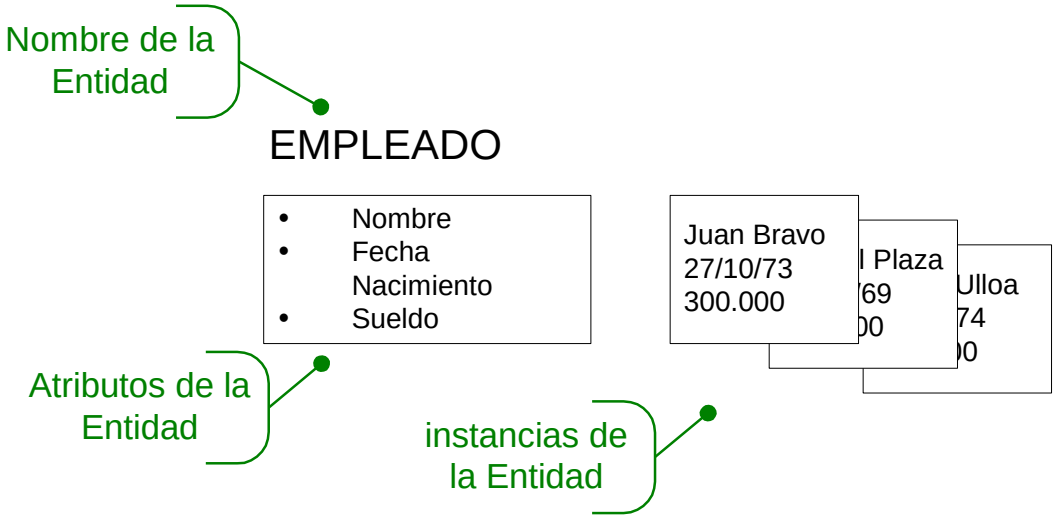
Ejemplo:

edad_empleado

antigüedad_cliente

Instancias

Cada entidad debe tener múltiples ocurrencias o instancias. Por ejemplo, la entidad Empleado, tiene una ocurrencia (o instancia) por cada empleado en la empresa.



Tuplas

Las tablas de la ilustración 5 muestran cada un conjunto de registros, la tabla persona muestra 6 registros y la tabla cuenta muestra 9 de ellos, cada uno de estos registros representa un objeto, es decir, de la tabla persona hay 6 (objetos) o personas a la agrupación de los datos de cada registro se le llama TUPLA. En términos simples una TUPLA de una base datos es el conjunto de todos los atributos (columnas) de una fila.

identificacion	nombres	apellidos	edad
8.127.346	Ana lis	Mendez	34
212.394.762	Jose Omar	Ciano	28
545.232.546	Luis Dario	Marquez	21
125.446.732	Armando	Bronca Segura	35
33.565.236.787	Vistor José	Diento Paz	24
2.343.554.768	Jose Nicolás	Timado Ruiz	19
45.456.778	Andrea	Martinez	36

a) Tabla persona

numero_cuenta	saldo	fecha_creacion
19.283.746	$ 2.589.954,00	25/08/1998
67.789.901	$ 23.465.212,00	26/08/2001
32.112.312	$ 2.343.242,00	27/08/2004
41.928.374	$ 2.343.423,00	31/08/2013
38.342.657	$ 8.965.211,00	30/08/2010
32.443.171	$ 11.172.594,33	22/06/2001
34.920.608	$ 14.483.578,83	01/09/2016
17.980.455	$ 17.794.563,33	03/09/2019
26.754.825	$ 21.105.547,83	12/04/2022

b) Tabla cuenta

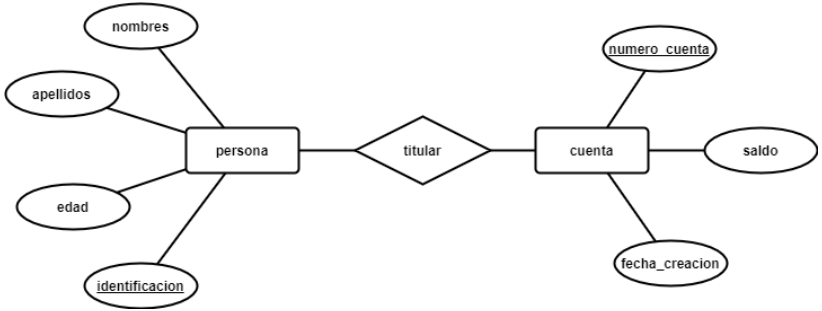
Llaves o Claves

Dado que en una tabla no hay filas repetidas, se pueden distinguir unas de otras, es decir, se pueden identificar de manera única (por el principio de unidad). Por lo tanto, la forma de identificarlas inequívocamente es según los valores de los atributos. De esta forma se conoce como claves a que los atributos que sirven para identificar un registro de forma inequívoca.

Llaves o Claves

- Ejemplo:

En la tabla persona, es sabido por todos que no existe dos personas con el mismo número de documento de identidad, por lo tanto, se puede emplear el atributo **identificación** como calve de la tabla persona. De igual forma no debe existir dos cuentas bancarias con el mismo número de cuenta, entonces, el atributo **numero_cuenta** se puede emplear como clave de la tabla cuenta. Esto se lo puede presentar en un diagrama de entidad relación cuando a los atributos que componen una clave les marcamos subrayado el nombre del atributo como se muestra en la siguiente ilustración.

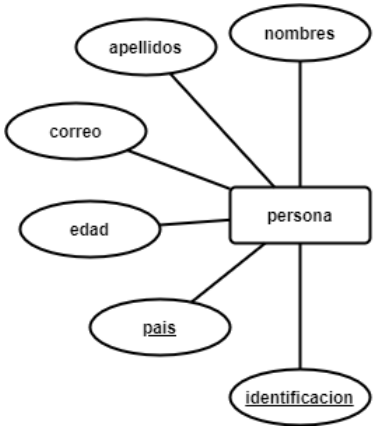


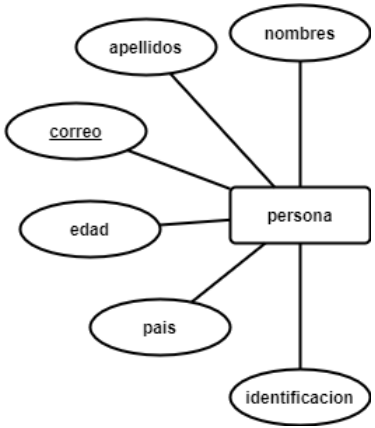
Llaves Primaria

Se empleará el término clave primaria para nombrar una clave candidata que es elegida por el diseñador o analista de la base de datos como clave principal para identificar las tuplas dentro de una entidad. Una clave (primaria, candidata y superclave) es una propiedad del conjunto de entidades, más que de las entidades individuales. Entre dos tuplas individuales en el conjunto no pueden tener el mismo valor en sus atributos clave primaria al mismo tiempo. La designación de una clave primaria es en sí una restricción que modela una característica del mundo real.

Llaves Compuesta

Si se escogiera el conjunto de atributos **identificación** y **país** como clave de la entidad, se está haciendo una clave compuesta esta porque es un conjunto de más de un atributo el que se usa para este fin.





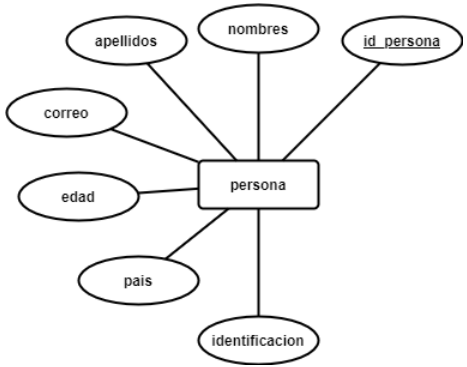
El otro asunto a considerar es que, se puede emplear el **correo** electrónico como clave ya que no existen dos usuarios con el mismo correo electrónico, esto es válido si se definió como regla del banco que cada cliente tenga su **correo** electrónico diferente de los demás, estos dos calos se representanta en la siguiente ilustración.

Llave Candidata

Los dos caminos propuestos en la ilustración 7 son válidos, y se denominan claves candidatas porque con ambas se puede lograr el principio de unicidad, y es potestad de diseñador o analista de sistemas determinar cuál de las dos usa.

Sin embargo, existe otro tipo de solución y es asignar le un atributo identificador, a la entidad persona que se encargue exclusivamente de identificar inequívocamente cada tupla o fila, a este atributo le podemos llamar **id_persona** que sería un número que el programa o software para el que estamos desarrollando la base le asignara a cada cliente cada vez que se registre por primera vez los datos de una persona (generalmente una secuencia). De esta forma tendríamos 3 claves candidatas, la ilustración 8 muestra cómo sería la entidad y la tabla que representa.

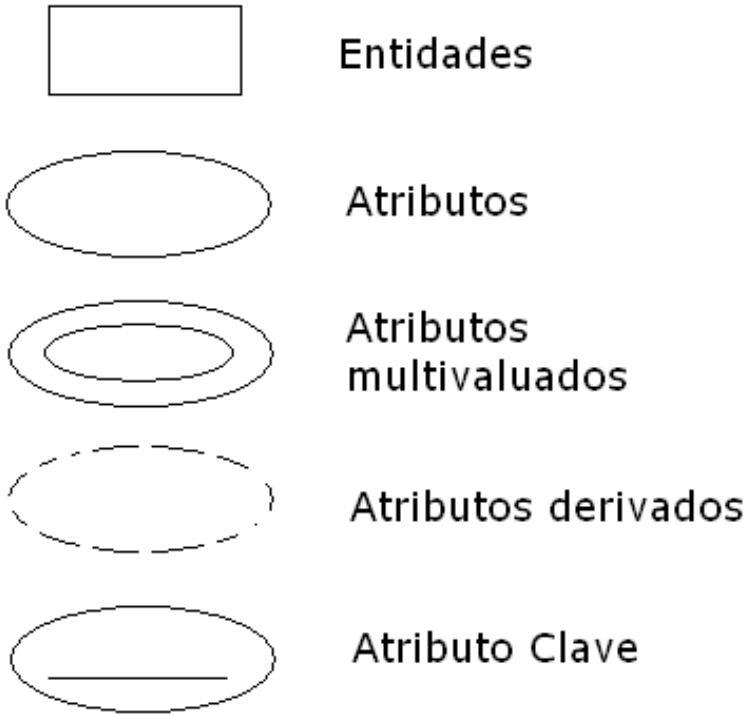
Llave Candidata



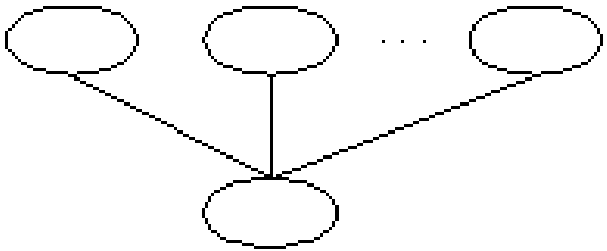
id_persona	pais	identificacion	correo	nombres	apellidos	edad
1	colombia	8.127.346	ana@rmail.com	Ana lis	Mendez	34
2	colombia	212.394.762	jose@pmail.com	Jose Omar	Ciano	28
3	colombia	545.232.546	luis@jmail.com	Luis Dario	Marquez	21
4	colombia	125.446.732	arm@hmail.com	Armando	Bronca Segura	35
5	colombia	33.565.236.787	victor@umail.com	Vistor José	Diento Paz	24
6	colombia	2.343.554.768	josen@pmail.com	Jose Nicolás	Timado Ruiz	19
7	perú	45.456.778	andrea@email.com	Andrea	Martinez	36

Tabla persona

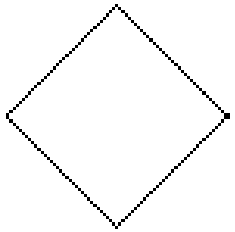
Notación Diagramas E-R



Notación Diagramas E-R



Atributo Compuesto



Relaciones

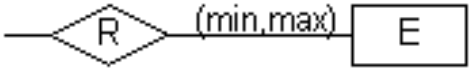
Notación Diagramas E-R



Participación total de E2 en R

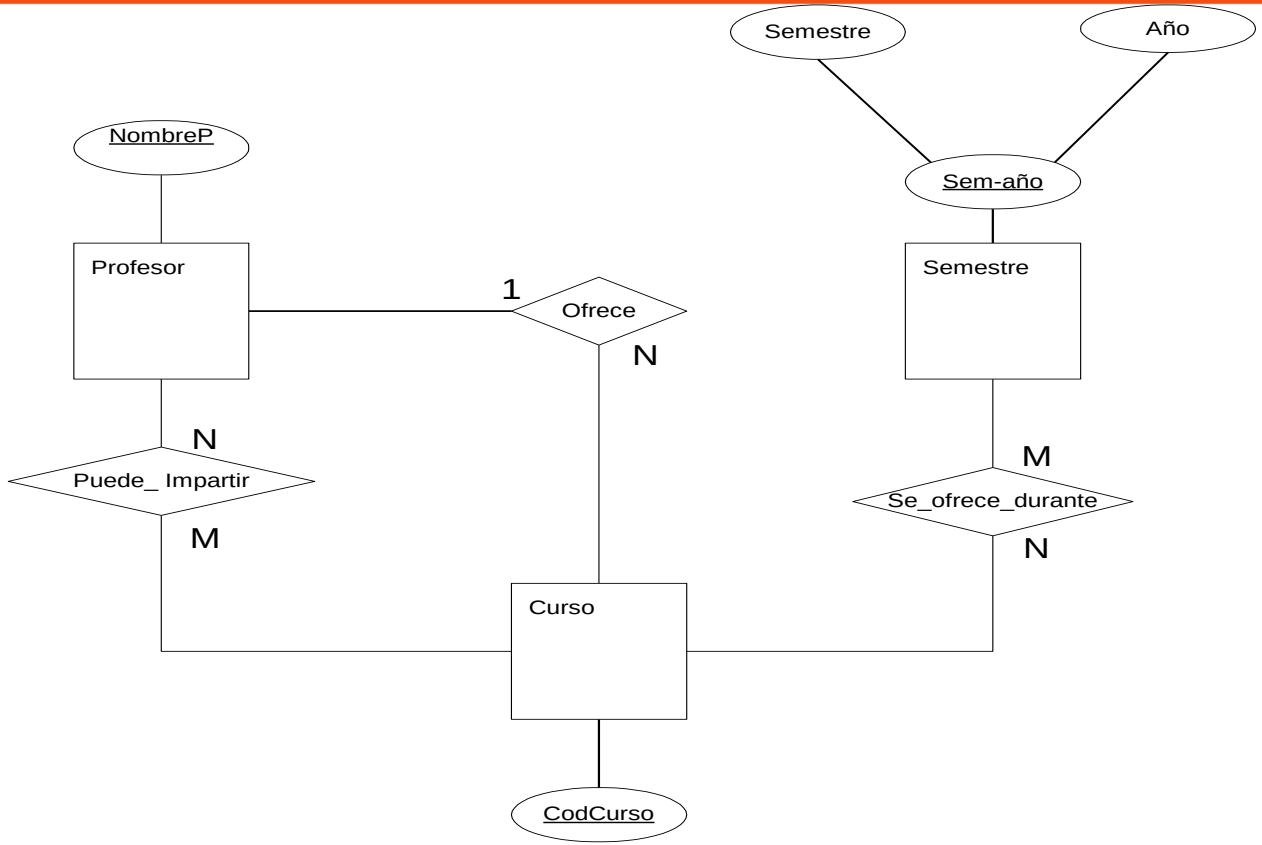


Razón de cardinalidad 1:N para E1:E2 en R



Restricción estructural (min,max) de la participación de E en R

Notación Diagramas E-R



Relaciones

Una relación es una asociación bi-direccional entre dos entidades, o entre una entidad consigo misma.

Ejemplo:

la relación entre las entidades INSTRUCTOR y CURSO puede ser:

Cada CURSO puede ser impartido por sólo un INSTRUCTOR.

Cada INSTRUCTOR puede ser asignado a uno a más cursos.

Principio de unicidad: en una tabla, nunca deben existir TUPLAS repetidas, es decir, no debe haber más de una fila con exactamente los mismos valores en cada columna que otra fila de la misma tabla. En otras palabas cada fila debe diferir al menos en un elemento (columna) con relación a las demás.

Relaciones

Grado de opcionalidad:

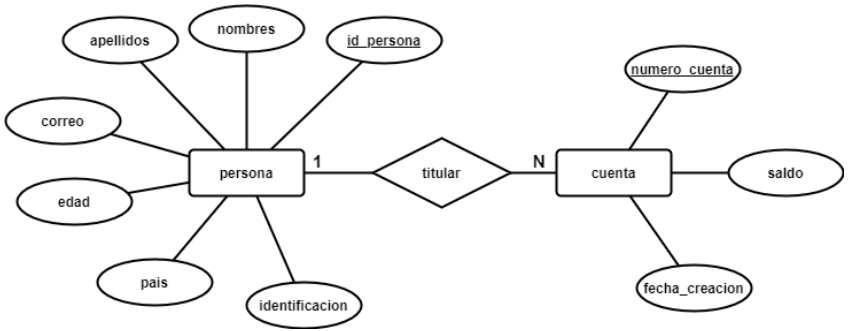
- Opcional: Puede ser.
- Obligatorio: Debe ser

Grado de cardinalidad:

- uno o muchos
- uno a uno

Relaciones de uno a muchos.

Una relación de uno a varios o de uno a muchos es el tipo de relación más empleada. En este tipo de relación, una fila de la tabla A puede tener muchas filas coincidentes en la tabla B. Pero una fila de la tabla B solo puede tener una fila coincidente en la tabla A. Por ejemplo, las tablas **pesona** y **cuenta** tienen una relación de uno a muchos. Es decir, cada persona puede tener más de una cuenta. Pero cada título proviene de una sola editorial.



Relaciones de uno a muchos.

A esta representación se le nomina multiplicidad **1:N** y se lee de la siguiente forma: **UNA persona** es **titular** de **N cuenta**s bancarias, de esta forma una persona puede tener **N** (que puede ser cero, uno o más de una) cuentas bancarias. Se puede leer en forma inversa que el resultado es el mismo: muchas o varias cuentas bancarias pueden tener un mismo titular.

Cuando una relación de es uno a muchos y se mapea una un modelo de base de datos relacional, la relación **titular** (rombo) se termina convirtiendo una columna de la tabla de los muchos.

id_persona	pais	identificacion	correo	nombres	apellidos	edad
1	colombia	8.127.346	ana@rmail.com	Ana lis	Mendez	34
2	colombia	212.394.762	jose@pmail.com	Jose Omar	Ciano	28
3	colombia	545.232.546	luis@jmail.com	Luis Dario	Marquez	21
4	colombia	125.446.732	arm@hmail.com	Armando	Bronca Segura	35
5	colombia	33.565.236.787	victor@umail.com	Vistor José	Diento Paz	24
6	colombia	2.343.554.768	josen@pmail.com	Jose Nicolás	Timado Ruiz	19
7	perú	45.456.778	andrea@email.com	Andrea	Martinez	36

a) Tabla persona

numero_cuenta	titular	saldo	fecha_creacion
19.283.746	7	$ 2.589.954,00	25/08/1998
67.789.901	1	$ 23.465.212,00	26/08/2001
32.112.312	5	$ 2.343.242,00	27/08/2004
41.928.374	2	$ 2.343.423,00	31/08/2013
38.342.657	6	$ 8.965.211,00	30/08/2010
32.443.171	1	$ 11.172.594,33	22/06/2001
34.920.608	5	$ 14.483.578,83	01/09/2016
17.980.455	3	$ 17.794.563,33	03/09/2019
26.754.825	4	$ 21.105.547,83	12/04/2022

b) Tabla cuenta

Relaciones de uno a muchos.

REGLA DE MAPEO 1:N de Modelo entidad-relación a modelo relacional:

Una relación de uno a mucho (1:N) se transforma en una columna en la tabla que tiene la multiplicidad de los muchos (cuenta). Y todos los datos de esta columna existen como llave primaria en la tabla que tiene la multiplicidad uno (tabla persona).

id_persona	pais	identificacion	correo	nombres	apellidos	edad
1	colombia	8.127.346	ana@rmail.com	Ana lis	Mendez	34
2	colombia	212.394.762	jose@pmail.com	Jose Omar	Ciano	28
3	colombia	545.232.546	luis@jmail.com	Luis Dario	Marquez	21
4	colombia	125.446.732	arm@hmail.com	Armando	Bronca Segura	35
5	colombia	33.565.236.787	victor@umail.com	Vistor José	Diento Paz	24
6	colombia	2.343.554.768	josen@pmail.com	Jose Nicolás	Timado Ruiz	19
7	perú	45.456.778	andrea@email.com	Andrea	Martinez	36

a) Tabla persona

numero_cuenta	titular	saldo	fecha_creacion
19.283.746	7	$ 2.589.954,00	25/08/1998
67.789.901	1	$ 23.465.212,00	26/08/2001
32.112.312	5	$ 2.343.242,00	27/08/2004
41.928.374	2	$ 2.343.423,00	31/08/2013
38.342.657	6	$ 8.965.211,00	30/08/2010
32.443.171	1	$ 11.172.594,33	22/06/2001
34.920.608	5	$ 14.483.578,83	01/09/2016
17.980.455	3	$ 17.794.563,33	03/09/2019
26.754.825	4	$ 21.105.547,83	12/04/2022

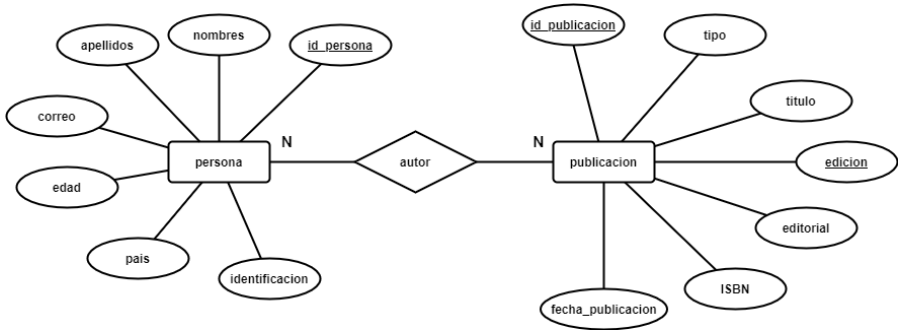
b) Tabla cuenta

Relaciones de muchos a muchos.

En una relación de muchos a muchos, una fila de la tabla A puede tener relación con muchas filas en la tabla B y también en sentido contrario una fila de la tabla B con respecto de varias filas de la tabla A. Esta relación se crea definiendo una tercera tabla denominada tabla de relación. Esta nueva tabla tiene una clave principal de la tabla compuesta de las claves foráneas de la tabla A y de la B.

Relaciones de muchos a muchos.

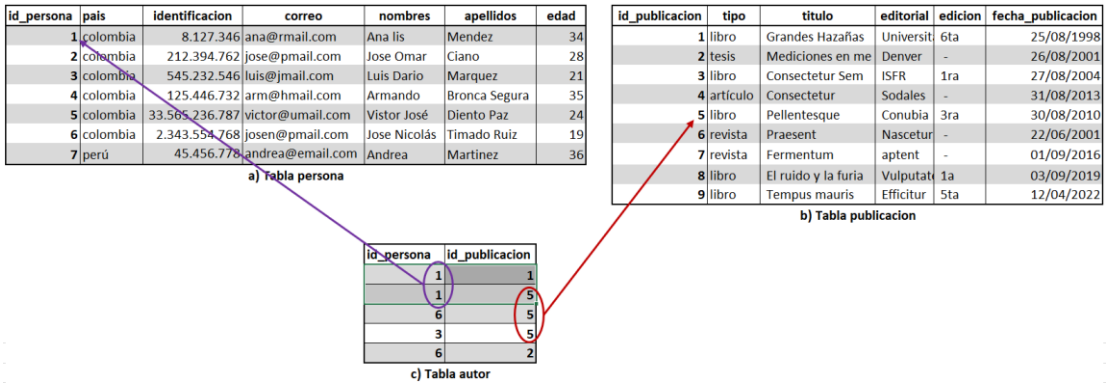
Ejemplo:
Supongamos que necesitamos almacenar la información de unas referencias bibliográficas o publicaciones, de libros, tesis, artículos que escriben los estudiantes de una universidad, y como se sabe un libro, tesis, artículo o cualquier publicación puede ser elaborada por más de una persona, y a su vez, una persona (docente, estudiante o investigador) puede tener más de una publicación de su autoría.



Relaciones de muchos a muchos.

REGLA DE MAPEO N:N de Modelo entidad-relación a modelo relacional: En una relación de uno a mucho (N:N) se transforma en una tabla cuya llave primaria está compuesta por las llaves primarias de las otras dos tablas.

Como se puede ver en la ilustración, la tabla c) autor, nos dice que la persona Ana Lis, es autora de los libros de título Grandes hazañas y Pellentesque, también que el libro Pellentesque es escrito además de Ana Lis por Luis Dario y Jose Nicolas. Lo anterior haciendo referencia a las llaves primarias de las tablas a) **persona** y b) **publicacion**.



id_persona	pais	identificacion	correo	nombres	apellidos	edad
1	colombia	8.127.346	ana@rmail.com	Ana lis	Mendez	34
2	colombia	212.394.762	jose@pmail.com	Jose Omar	Ciano	28
3	colombia	545.232.546	luis@jmail.com	Luis Dario	Marquez	21
4	colombia	125.446.732	arm@hmail.com	Armando	Bronca Segura	35
5	colombia	33.565.236.787	victor@umail.com	Vistor José	Diento Paz	24
6	colombia	2.343.554.768	josen@pmail.com	Jose Nicolás	Timado Ruiz	19
7	perú	45.456.778	andrea@email.com	Andrea	Martinez	36

a) Tabla persona

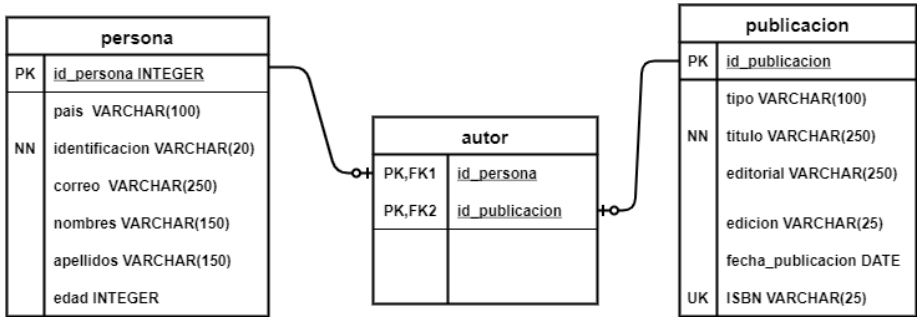
id_publicacion	tipo	titulo	editorial	edicion	fecha_publicacion
1	libro	Grandes Hazañas	Universit	6ta	25/08/1998
2	tesis	Mediciones en me	Denver	-	26/08/2001
3	libro	Consectetur Sem	ISFR	1ra	27/08/2004
4	artículo	Consectetur	Sodales	-	31/08/2013
5	libro	Pellentesque	Conubia	3ra	30/08/2010
6	revista	Praesent	Nascetur	-	22/06/2001
7	revista	Fermentum	aptent	-	01/09/2016
8	libro	El ruido y la furia	Vulputat	1a	03/09/2019
9	libro	Tempus mauris	Efficitur	5ta	12/04/2022

b) Tabla publicacion

id_persona	id_publicacion
1	1
1	5
6	5
3	5
6	2

c) Tabla autor

Relaciones de muchos a muchos.



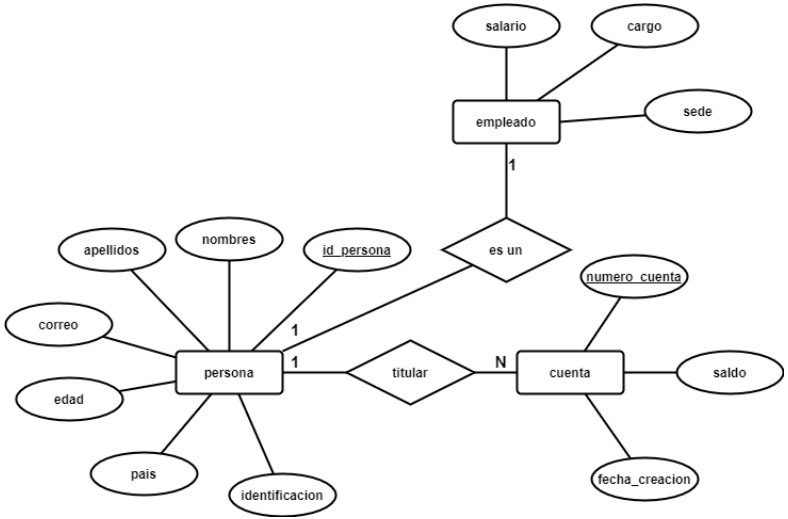
Relaciones de uno a uno

En una relación uno a uno, una fila de la tabla A solo puede tener una fila coincidente en la tabla B y viceversa. Se crea una relación uno a uno si ambas columnas relacionadas son claves principales o tienen restricciones únicas. Este tipo de relación no es común porque la mayoría de la información que está relacionada de esta manera estaría en una tabla. Puede usar una relación uno a uno por alguno de estos motivos:

- Dividir una tabla con muchas columnas.
- Aísle parte de una tabla por motivos de seguridad.
- Almacenar datos de corta duración y que podrían eliminarse fácilmente borrando la tabla periódicamente.
- Almacenar información que solo se aplique a un subconjunto de la tabla principal.

Relaciones de uno a uno

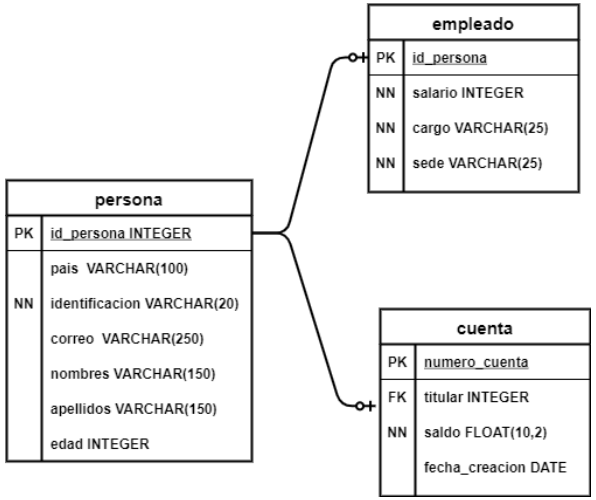
Viviendo al ejemplo del banco, supongamos esta vez, la empresa necesita información sobre sus empleados, que, a la vez, pueden ser clientes de banco.



Relaciones de uno a uno

REGLA DE MAPEO 1:1 de Modelo entidad-relación a modelo relacional

En el momento de convertir la relaciono con multiplicidad 1:1 a tablas la relación uno a uno se convierte en una columna de la tabla empleado con restricción de unicidad (**U**nique **K**ey, o **P**rimary **K**ey) como se muestra en el siguiente diagrama relacional.





G R A C I A S

Línea de atención al ciudadano: 018000 910270
Línea de atención al empresario: 018000 910682

@SENAcomunica

www.sena.edu.co

EJERCICIOS

1. Se desea calcular la distancia recorrida (m) por un móvil que tiene velocidad constante (m/s) durante un tiempo T (Sg), considerar que es un MRU (Movimiento Rectilíneo Uniforme)

2. Se necesita obtener el promedio simple de un estudiante a partir de sus tres notas parciales.

3. Elaborar un algoritmo que solicite el número de respuestas correctas, incorrectas y en blanco, correspondientes a postulantes, y muestre su puntaje final considerando, que por cada respuesta correcta tendrá 4 puntos, respuestas incorrectas tendrá -1 y respuestas en blanco tendrá 0.

4. Elaborar un algoritmo que permita ingresar el número de partidos ganados, perdidos y empatados, por algún equipo en el torneo apertura, se debe de mostrar su puntaje total, teniendo en cuenta que por cada partido ganado obtendrá 3 puntos, empatado 1 punto y perdido 0 puntos.

5. Se requiere el algoritmo para elaborar la planilla de un empleado. Para ello se dispone de sus horas laboradas en el mes, así como de la tarifa por hora.

6. De un triángulo se tiene la longitud de la base y la longitud de la altura. Determine el valor de su área.

7. Se requiere llenar un tanque que tiene una capacidad de 50 metros cúbicos. Haga un algoritmo que imprima las horas que tarda en llenarse dicho tanque con una manguera que tiene una capacidad de L litros de agua por minuto.

8. Un amigo suyo acaba de iniciar un negocio de venta de zapatos. Por ahora sólo vende tres tipos de zapatos: sandalias, tenis y mocasines. Cada tipo de zapato lo adquiere a un costo distinto y para venderlos, supone una ganancia del 55%. Cuando un cliente llega debe comprar de los tres tipos de zapatos y la cantidad que desee de cada uno de ellos. El cliente tiene derecho a un 8% de descuento sobre la compra que realiza. Ayúdele a su amigo a crear un programa que, para un cliente dado, muestre su nombre, el valor de la venta sin descuento, el descuento, valor de la venta con descuento y valor de la venta incluyendo IVA (venta neta final).

Tecnólogo de análisis y desarrollo de software ficha técnica 2455274

Fundamentos de programación estructurada y estructuras cíclicas GA3-220501093-AA2-EV01

LUZ ANGELICA SANCHEZ RAMIREZ
Aprendiz

Jesus Andres Silva Plazas
Instructor

Centro agroempresarial y desarrollo pecuario del Huila

2022

1 | 0

INTRODUCCION

Hemos leído el componente informativo donde tenemos que tener el análisis, diseño e implementación de cada problema para tener claro para poder desarrollarlo de la forma efectiva para que sea resuelto, fácil y entendido de la mejor forma, vamos a plantear los siguientes ejercicios:

1. **Algoritmo que a partir de una fecha de nacimiento y una fecha actual determine la edad en años actual de una persona**.

Análisis:

- **Formular el problema:** consiste desde la fecha de nacimiento y fecha actual en edad en años actuales
- **Resultados esperados:** se espera que el algoritmo funcione sin ningún error al ser ejecutado
- **Datos disponibles:** son como fecha de nacimiento como día, mes y año y con la fecha actual igualmente como mes día y año
- **Restricciones:** ninguna
- **Procesos necesarios:** solicitar que el usuario indique los respectivos datos como día, mes y año

Diseño:

Algoritmo Años_Actual_Persona

1 | 0

```
        Definir Edad, DiaActual, MesActual,
    AñoActual Como Entero
        Definir DiaNacim, MesNacim,
AñoNacim Como Entero
    Escribir "Por favor escriba el dia de la fecha de
nacimiento "
    leer DiaNacim
    Escribir "Por favor escriba el mes de la fecha
de nacimiento "
    leer MesNacim
    Escribir "Por favor escriba el año de la fecha
de nacimiento "
    leer AñoNacim
    Escribir "Por favor escriba la fecha en dia
actual"
    leer DiaActual
    Escribir "Por favor escriba la fecha  mes actual
"
    leer MesActual

    Escribir "Por favor escriba la fecha  año
actual"
    leer AñoActual

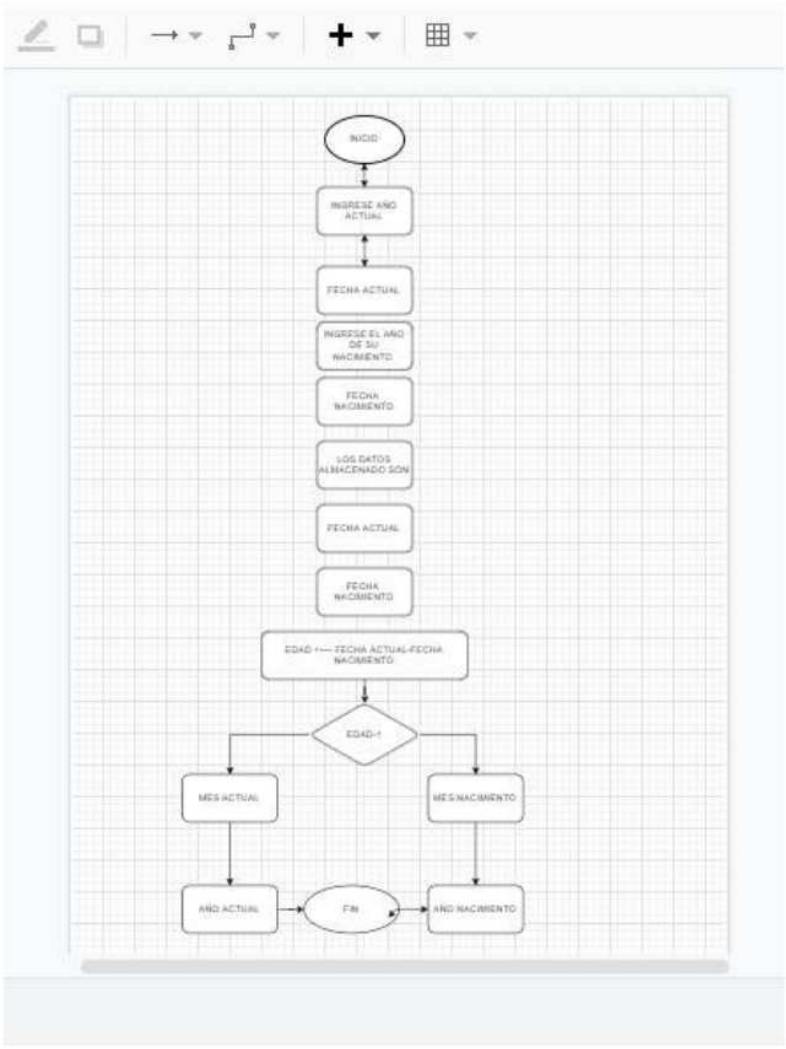
    Edad = AñoActual - AñoNacim

    si MesNacim > MesActual Entonces
        Edad = Edad-1
    SiNo
```


1 | 0

```
                    si MesNacim == MesActual
            Entonces
                        si DiaNacim > DiaActual
Entonces
                    Edad = Edad-1
                FinSi
                si DiaNacim == DiaActual Entonces
                    Escribir "¡Oh!Felicidades por tu
cumpleaños"
                FinSi
            FinSi
        FinSi
        Escribir "Tu edad actual es:  ",edad, "años"
FinAlgoritmo
```


1 | 0



2. Algoritmo que permita determinar si un año indicado es o no un año bisiesto.

Análisis:

- **Formular el problema:** consiste diferenciar que año es Bisiesto y que año no es bisiesto

1 | 0

- **Resultados esperados:** se espera que el algoritmo funcione sin ningún error al ser ejecutado y sin ningún problema

- **Datos disponibles:** año actual, años pasados o futuros
- **Restricciones:** ninguna
- **Procesos necesarios:** solicitar que el usuario indique los respectivos datos como año sea vigente, pasado o futuro para obtener la información clara

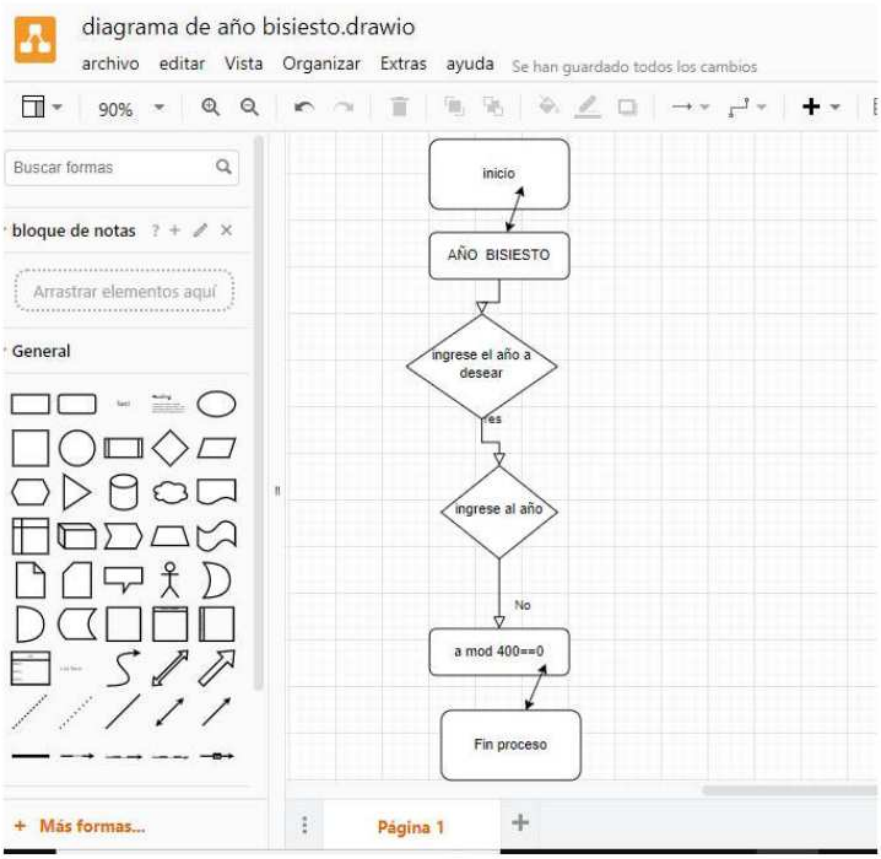
Diseño:

```
Algoritmo Año_Bisiesto
        Escribir " Por favor dígite si el año es Bisiesto o No
Bisiesto"
        Leer Año

        si (Año mod 4=0 Y Año mod 100<>0) Entonces
                Escribir "El Año " Año " Es Bisiesto"
        sino
                si (Año mod 400 =0 y Año mod 100=0)
                        Escribir  "El Año " Año "Es Bisiesto"
                sino
                        Escribir  "El Año " Año " No es Bisiesto"

                FinSi
        FinSi
FinAlgoritmo
```


1 | 0


diagrama de año bisiesto.drawio
archivo editar Vista Organizar Extras ayuda Se han guardado todos los cambios
90%
Buscar formas
bloque de notas
Arrastrar elementos aquí
General
+ Más formas...
inicio
AÑO BISIESTO
ingrese el año a desear
es
ingrese al año
No
a mod 400==0
Fin proceso
Página 1

1 | 0

PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL
FORMATO GUÍA DE APRENDIZAJE

1. IDENTIFICACIÓN DE LA GUÍA DE APRENDIZAJE

- **Denominación del programa de formación**: Análisis y Desarrollo de Software
- **Código del programa de formación**: 228118
- **Nombre del proyecto:** Construcción de software integrador de tecnologías orientadas a servicios
- **Fase del proyecto**: Planeación
- **Actividad de proyecto:** Estructurar procesos lógicos para la construcción de algoritmos.
- **Competencias**:

  Técnicas:
  - 220501093- Evaluar requisitos de la solución de software de acuerdo con metodologías de análisis y estándares.

  Claves:

  Transversales:

- **Resultados de aprendizaje a alcanzar**

  Técnicos:
  - 220501093-03 - Desarrollar procesos lógicos a través de la implementación de algoritmos.

  Claves:

  Transversales:

- **Duración de la Guía**: 240 horas
  Técnicas: 96 horas.
  Claves: 48 horas.
  Transversales: 96 horas.

2. PRESENTACIÓN

Estimado aprendiz, el SENA extiende una cordial bienvenida a este proceso de formación, en el cual se desarrollarán los temas que fundamentan todo lo correspondiente a la programación de software, es decir las bases algorítmicas relacionadas con las estructuras secuenciales, condicionales y cíclicas sobre la que se sustentan la construcción de la lógica que se usará para dar soluciones a problemas utilizando lenguajes de programación.

Permanece atento, y ¡**APROVECHA ESTA OPORTUNIDAD DE APRENDIZAJE!**

3. FORMULACIÓN DE LAS ACTIVIDADES DE APRENDIZAJE

En este apartado se describirán las actividades de aprendizaje planteadas para el desarrollo de la competencia, así como la indicación respectiva para que logre la elaboración de las evidencias en un total de 240 horas.

De este programa de formación, en cada una de las actividades se detallan las evidencias a entregar. La entrega de las evidencias se realizará durante (6) semanas planeadas para esta guía. En el siguiente cuadro se observa para la competencia, el momento y las evidencias a entregar según las actividades de aprendizaje asociadas a la misma.

Competencia	Semana 1	Semana 2	Semana 3	Semana 4	Semana 5	Semana 6
	Evidencias					
220501093 – Evaluar requisitos de la solución de acuerdo con metodologías de análisis y estándares.	Evidencia de conocimiento: Bases conceptuales de lógica proposicional. GA3-220501093-AA1-EV01 Evidencia de producto: Taller de resolución de problemas de algoritmos en pseudocódigo y diagramas de flujo. GA3-220501093-AA1-EV02	Evidencia de conocimiento: Fundamentos de programación estructurada y estructuras cíclicas. GA3-220501093-AA2-EV01	Evidencia de desempeño: Foro temático: Estructuras de programación. GA3-220501093-AA2-EV02	Evidencia de Producto: Taller aplicando funciones y procedimientos en la solución de algoritmos. GA3-220501093-AA2-EV03	Evidencia de conocimiento: Bases teóricas de estructuras de almacenamiento en memoria. GA3-220501093-AA3-EV01	Evidencia de desempeño: Resolución a problemas algorítmicos aplicando estructuras de almacenamiento. GA3-220501093-AA3-EV02
Horas por semana	**16**	**16**	**16**	**16**	**16**	**16**

El proceso de formación se centra en el aprendizaje basado en problemas y posibilitará la realización de una gran cantidad de ejercicios prácticos, los cuales permitirán desarrollar habilidades de análisis para asociar las características de los problemas con las estructuras algorítmicas requeridas para el desarrollo de una solución.

3.1. Actividades de aprendizaje de la competencia: - Evaluar requisitos de la solución de software de acuerdo con metodologías de análisis y estándares (220501093).

Esta competencia se centra en la realización de actividades de evaluación de las mejores estrategias que permitan abordar cada uno de los requisitos de la solución de software de una manera organizada y efectiva, específicamente para esta guía se abordarán las bases requeridas para la construcción de algoritmos los cuales son la esencia de la programación

3.1.1. Actividad de aprendizaje GA3-220501093 - AA1: Realizar actividades para el desarrollo del pensamiento algorítmico.

En esta actividad se estudiarán los conceptos básicos asociados al concepto de algoritmo, se abordará la definición formal de algoritmo, las características del pensamiento algorítmico y cómo esto nos ayuda a plantear soluciones a problemas reales y problemas asociados a la programación. Luego se abordan cada uno de los componentes requeridos para abordar el problema e identificar cada uno de los elementos de este.

Duración: 14 horas

Materiales de formación a consultar: para el desarrollo de esta actividad es importante la lectura y análisis del material de formación: "**Introducción a la algoritmia"**

Evidencias: a continuación, se describen las acciones y las correspondientes evidencias que conforma la actividad de aprendizaje**:**

- **Evidencia GA3-220501093-AA1-EV01**: **Bases conceptuales de lógica proposicional.**

Esta evidencia se centra en consolidar los conceptos relacionados con los componentes de lógica matemática indispensables para el desarrollo de estructuras de control algorítmicas

Para su desarrollo es importante la lectura del componente formativo. Se resolverá un cuestionario que se emplea como técnica de evaluación. Las preguntas del cuestionario pueden ser:

- ✓ De carácter formativo, las cuales permiten determinar el nivel alcanzado de desarrollo de la competencia durante la formación.
- ✓ De carácter evaluativo, que determinan el dominio y el nivel máximo alcanzado en la competencia.

Lineamientos generales para la entrega de la evidencia:

- **Duración:** 30 minutos.
- Para responder el cuestionario remítase al área de la actividad correspondiente y acceda al espacio: **Bases conceptuales de lógica proposicional. GA3-220501093-AA1-EV01**.

- **Evidencia GA3-220501093-AA1-EV02**: **Taller de resolución de problemas de algoritmos en pseudocódigo y diagramas de flujo.**

Teniendo en cuenta las recomendaciones para el análisis de problemas desarrollado durante el componente formativo, en esta actividad se plantean algunos problemas los cuales deberá leer detenidamente para luego identificar los elementos de entrada y los resultados esperados. Adicionalmente se introduce el tema de diagramas de flujo para lo cual deberá realizar una investigación corta respondiendo algunos lineamientos establecidos en esta actividad

Elementos para tener en el desarrollo del taller:

Sección 1 Taller

Teniendo en cuenta los siguientes problemas identificar los posibles elementos de entrada y resultados esperados de cada uno:

- Se desea elaborar un algoritmo que permita identificar la cantidad de dólares equivalentes a una cantidad de pesos colombianos.
- Se desea elaborar un algoritmo que permita determinar la temperatura equivalente en grados centígrados a la cantidad de grados Fahrenheit actuales en la ciudad de New York.
- Suponiendo que nos encontramos descansando en una nuestra casa en una ciudad de Colombia requiero hacer un plan detallado para llegar a mi sitio de trabajo a tiempo el día siguiente.
- Suponiendo que tengo habilidades en la elaboración de comida necesito elaborar un arroz con pollo para 5 personas.

Respecto a los problemas planteados:

- Para precisar los elementos de entrada de cada una de las situaciones planteadas no olvide las preguntas guía:
    - ¿Qué información es importante y necesaria para resolver el problema?
    - ¿Qué información no es importante y se puede prescindir?
    - ¿Cuáles son los datos de entrada conocidos?
    - ¿Cuál es la incógnita o que debemos calcular?
    - ¿Los datos se pueden agrupar en categorías?
    - ¿Qué información adicional hace falta para resolver el problema?
- Para precisar los resultados esperados se recomienda utilizar las preguntas guía:
    - ¿Qué información están solicitando?
    - ¿En qué formato se debe entregar esta información?

Sección 2 Taller

Elabore una investigación corta usando los materiales disponibles en la biblioteca o internet respecto a los fundamentos para la resolución de problemas con algoritmos, selecciones por lo menos tres fuentes que le permitan resolver las siguientes preguntas:

- Definición de diagrama de flujo.
- Símbolos mas importantes.
- Seleccionar uno de los problemas de la sección 1 y representarlo en su equivalente diagrama de flujo.

Respecto a los diagramas de flujo:

- Referenciar las fuentes de información utilizadas para resolver cada una de las preguntas utilizando formato APA.

Lineamientos generales para la entrega de la evidencia:

- **Productos para entregar:** Un documento con la solución del taller propuesto.
- **Extensión:** Libre.

- **Formato:** PDF o Word.
- Para hacer el envío de la evidencia remítase al área de la actividad correspondiente y acceda al espacio: **Taller de resolución de problemas de algoritmos en pseudocódigo y diagramas de flujo. GA3-220501093-AA1-EV02**.

3.1.2. Actividad de aprendizaje GA3-220501093-AA2 - Desarrollar soluciones algorítmicas a través de estructuras de control.

En esta actividad se estudiarán algunas formas de representación de algoritmos y se introducen algunos conceptos clave como: constantes, variables, acumuladores, contadores, operadores, jerarquía de los operadores. Finalmente se abordan las estructuras de control secuencial y herramientas existentes para la creación y pruebas de algoritmos.

Duración: 40 horas

Materiales de formación a consultar: para el desarrollo de esta actividad es importante la lectura y análisis del material de formación: "**Análisis y solución de problemas aplicando algoritmos"**

Evidencias: a continuación, se describen las acciones y las correspondientes evidencias que conforma la actividad de aprendizaje**:**

- **Evidencia GA3-220501093-AA2-EV01**: **Fundamentos de programación estructurada y estructuras cíclicas.**

Esta evidencia se centra en consolidar los conceptos relacionados con las metodologías de algoritmos y sus tres fases: análisis, diseño e implementación. Respecto a la implementación se aborda los pseudocódigos y diagramas de flujo.

Para su desarrollo es importante la lectura del componente formativo. Se resolverá un cuestionario que se emplea como técnica de evaluación. Las preguntas del cuestionario pueden ser:

- ✓ De carácter formativo, las cuales permiten determinar el nivel alcanzado de desarrollo de la competencia durante la formación.
- ✓ De carácter evaluativo, que determinan el dominio y el nivel máximo alcanzado en la competencia

Lineamientos generales para la entrega de la evidencia:

- **Duración:** 30 minutos.
- Para responder el cuestionario remítase al área de la actividad correspondiente y acceda al espacio: **Fundamentos de programación estructurada y estructuras cíclicas. GA3-220501093-AA2-EV01**.

- **Evidencia GA3-220501093-AA2-EV02**: **Foro temático: Estructuras de programación.**

Teniendo en cuenta las diferentes estructuras de control y su respectiva forma de representación en diagrama de flujo y pseudocódigo, realizar su participación en el foro argumentando cual de las dos

representaciones le resulta más conveniente para cada una de las estructuras de control trabajadas. Tome en cuenta las siguientes preguntas orientadoras:

- ✓ ¿Qué beneficios tiene una representación en comparación con la otra?
- ✓ ¿Respecto a las herramientas existentes, cuál tiene más representación?
- ✓ ¿Cuál sería las más fácil para entender las estructuras algorítmicas?
- ✓ ¿Cuál permitirá una transición más fácil hacia la codificación?

Realice seguimiento al foro y responda a un comentario crítico-reflexivo de acuerdo con la intervención que hayan realizado sus compañeros demostrando construcción de conocimiento frente a los planteamientos que cada uno proponga.

Lineamientos generales para la entrega de la evidencia:

- **Productos para entregar:** participación en el foro.
- Para hacer el envío de la evidencia remítase al área de la actividad correspondiente y acceda al espacio: **Foro temático: Estructuras de programación. GA3-220501093-AA2-EV02**.

- **Evidencia GA3-220501093-AA2-EV03: Taller aplicando funciones y procedimientos en la solución de algoritmos.**

En esta actividad deberá aplicar todos los conocimientos adquiridos a lo largo del componente formativo para dar solución a problemas utilizando la notación de pseudocódigo y diagramas de flujo usando las herramientas establecidas por el instructor. Tenga en cuenta los diferentes tipos de estructuras de control básicas: secuenciales, condicionales y repetitivas.

La descripción de cada uno de los enunciados a resolver se encuentra en el ***Anexo 1.***

Elementos para tener para la solución de este taller:

- Cada problema se debe solucionar usando las dos notaciones (pseudocódigo y diagrama de flujo).
- Debe utilizar la herramienta o el modelo de representación de cada notación sugerida por el instructor.

Lineamientos generales para la entrega de la evidencia:

- **Productos para entregar:** Un único archivo comprimido con el nombre del estudiante donde se encuentre todas las soluciones a cada uno de los ejercicios presentados en el anexo 1.
- **Formato:** ZIP.
- Para hacer el envío de la evidencia remítase al área de la actividad correspondiente y acceda al espacio: **Taller aplicando funciones y procedimientos en la solución de algoritmos. GA3-220501093-AA2-EV03**.

3.1.3. Actividad de aprendizaje GA3-220501093-AA3 - Solucionar planteamientos aplicando el concepto de arreglos (vectores y matrices).

Esta actividad se centra en la introducción a los lenguajes de programación abordando conceptos básicos, características de los lenguajes, sus tipos y el proceso de estación. Luego se identifican los tipos de datos, operadores, comentarios y expresiones admitidas por el lenguaje de programación seleccionado.

Finalmente se repasan cada una de las estructuras de control y se incluye adicionalmente las estructuras de datos tipo vector y matriz.

Duración: 42 horas

Materiales de formación a consultar: para el desarrollo de esta actividad es importante la lectura y análisis del material de formación: " **Codificación de Algoritmos"**

Evidencias: a continuación, se describen las acciones y las correspondientes evidencias que conforma la actividad de aprendizaje**:**

- **Evidencia GA3-220501093-AA3-EV01**: **Bases teóricas de estructuras de almacenamiento en memoria.**

Esta evidencia se centra en consolidar los conceptos básicos relacionados con los lenguajes de programación, entornos de codificación e instalación y la sintaxis del lenguaje de JavaScript.

Para su desarrollo es importante la lectura del componente formativo. Se resolverá un cuestionario que se emplea como técnica de evaluación. Las preguntas del cuestionario pueden ser:

- ✓ De carácter formativo, las cuales permiten determinar el nivel alcanzado de desarrollo de la competencia durante la formación.
- ✓ De carácter evaluativo, que determinan el dominio y el nivel máximo alcanzado en la competencia

Lineamientos generales para la entrega de la evidencia:

- **Duración:** 30 minutos.
- Para responder el cuestionario remítase al área de la actividad correspondiente y acceda al espacio: **Bases teóricas de estructuras de almacenamiento en memoria. GA3-220501093-AA3-EV01**.

- **Evidencia GA3-220501093-AA3-EV02**: **Resolución a problemas algorítmicos aplicando estructuras de almacenamiento.**

En esta actividad deberá aplicar todos los conocimientos adquiridos a lo largo del componente formativo para dar solución a problemas utilizando todas las estructuras de control requeridas y el lenguaje JavaScript.

La descripción de cada uno de los enunciados a resolver se encuentra en el ***Anexo 2.***

Elementos para tener para la solución de este taller:

- Cada problema se debe solucionar usando el lenguaje JavaScript.
- Si bien existen muchas formas posibles de resolver un problema se recomienda utilizar las estructuras que permitan el desarrollo de acciones de una manera simple y eficiente.

Lineamientos generales para la entrega de la evidencia:

- **Productos para entregar:** Un único archivo comprimido con el nombre del estudiante donde se encuentre todas las soluciones a cada uno de los ejercicios presentados en el anexo 2.
- **Formato:** ZIP.

- Para hacer el envío de la evidencia remítase al área de la actividad correspondiente y acceda al espacio: **Resolución a problemas algorítmicos aplicando estructuras de almacenamiento. GA3-220501093-AA3-EV02**.

4. ACTIVIDADES DE EVALUACIÓN

Tome como referencia la técnica e instrumentos de evaluación citados en la guía de Desarrollo Curricular

Evidencias de Aprendizaje	Criterios de Evaluación	Técnicas e Instrumentos de Evaluación
Evidencia de conocimiento: Bases conceptuales de lógica proposicional. GA3-220501093-AA1-EV01	Soluciona problemas de lógica proposicional incorporando habilidades propias en el oficio como programador.	**Cuestionario:** IE- GA3-220501093-AA1-EV01
Evidencia de producto: Taller de resolución de problemas de algoritmos en pseudocódigo y diagramas de flujo. GA3-220501093-AA1-EV02		**Lista de chequeo:** IE- GA3-220501093-AA1-EV02
Evidencia de conocimiento: Fundamentos de programación estructurada y estructuras cíclicas. GA3-220501093-AA2-EV01	Crea soluciones a problemas mediante algoritmos que incluyen estructuras secuenciales, condicionales y cíclicas.	**Cuestionario:** IE- GA3-220501093-AA2-EV01
Evidencia de desempeño: Foro temático: Estructuras de programación. GA3-220501093-AA2-EV02		**Lista de chequeo:** IE- GA3-220501093-AA2-EV02
Evidencia de Producto: Taller aplicando funciones y procedimientos en la solución de algoritmos. GA3-220501093-AA2-EV03		**Lista de chequeo:** IE- GA3-220501093-AA2-EV03

Evidencia de conocimiento: Bases teóricas de estructuras de almacenamiento en memoria. GA3-220501093-AA3-EV01	Crea funciones y procedimientos en la solución de algoritmos para ordenar y simplificar los códigos. Manipula arreglos en diferentes dimensiones para dar solución a problemas reales.	**Cuestionario:** IE- GA3-220501093-AA3-EV01
Evidencia de desempeño: Resolución a problemas algorítmicos aplicando estructuras de almacenamiento. GA3-220501093-AA3-EV02	Administra la información de los usuarios por medio de archivos, permitiendo el ingreso, modificación y eliminación de los datos.	**Lista de chequeo:** IE- GA3-220501093-AA3-EV02

5. GLOSARIO DE TÉRMINOS

Archivo ejecutable: archivo en código binario que se genera a partir de la compilación de un programa y el cual representa el programa a ser usado por un cliente.
***JavaScript*:** Es un lenguaje de programación interpretado orientado a objetos.
Metodología: ciencia que consta de métodos y técnicas, que se aplican sistemáticamente durante un proceso de investigación o para solucionar una problemática.

6. REFERENTES BIBLIOGRÁFICOS

- Aguilar, J. L. (2008). *Fundamentos de programaci}n (Spanish Edition)* (4th ed.). McGraw-Hill Interamericana de España S.L.
- M. (2021a). *Introducción a la programación estructurada en C (Spanish Edition)* (#REF! ed.). PEARSO.
- Sierra, F. J. C. (2000). *C/C++ curso de programación*. Alianza Editorial.
- Vásquez, J. (2012). Análisis y diseño de algoritmos. Red tercer milenio, Tlalnepantla, Estado de México. ISBN: 978-607-733-053-0

7. CONTROL DEL DOCUMENTO

	Nombre	Cargo	Dependencia	Fecha
Autor (es)	Jonathan Guerrero Astaiza	Experto temático	Centro de Teleinformática y Producción Industrial	Noviembre 2021

8. CONTROL DE CAMBIOS (diligenciar únicamente si realiza ajustes a la guía)

	Nombre	Cargo	Dependencia	Fecha	Razón del Cambio
Autor (es)					

INTRODUCCIÓN A LA ALGORITMIA

Estructura de contenidos

INTRODUCCIÓN 3

1. CONCEPTO DE ALGORITMOS 3

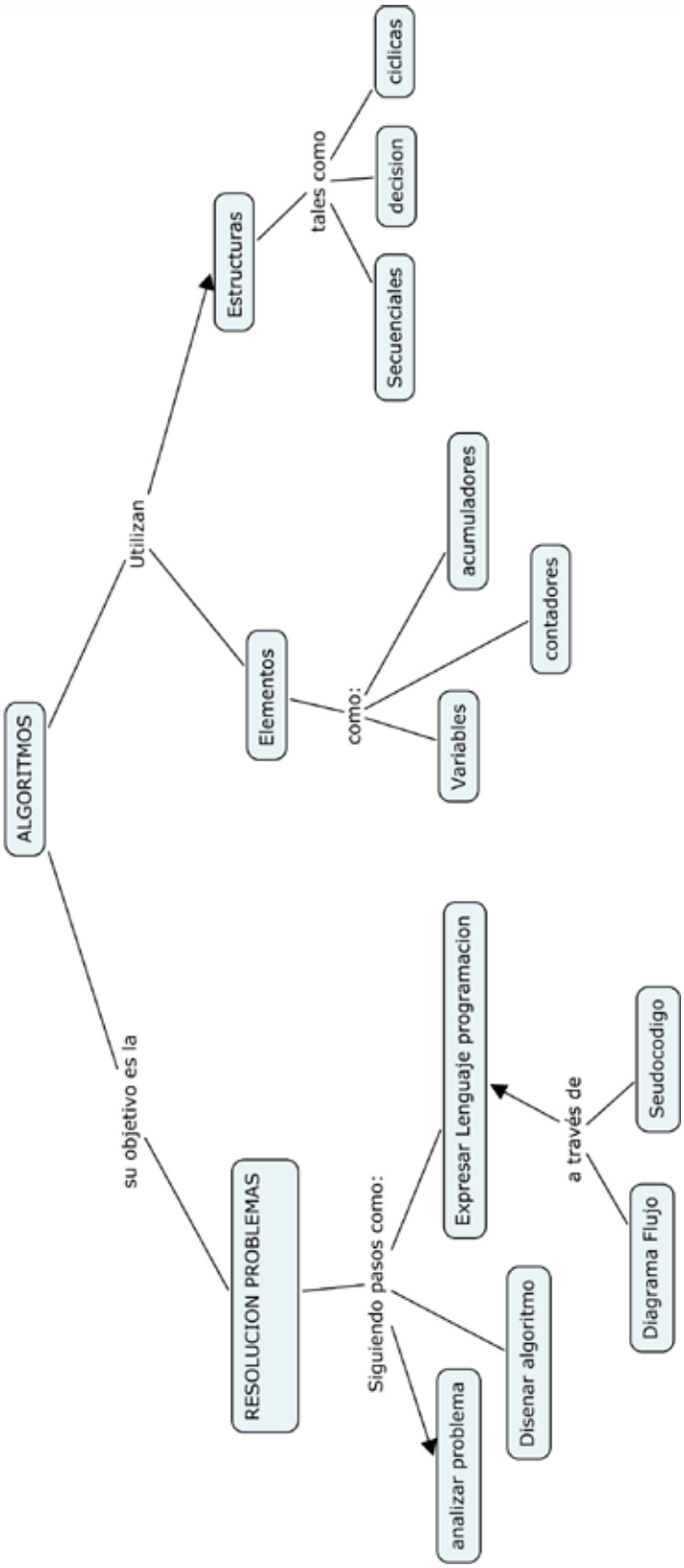
2. UN POCO DE HISTORIA 3

3. RESOLUCIÓN DE PROBLEMAS CON ALGORITMOS..... 4

4. SOLUCIÓN DE PROBLEMAS CON DIAGRAMAS DE FLUJO Y LENGUAJE NATURAL (SEUDOCÓDIGO)..... 14

5. ESTRUCTURAS ALGORÍTMICAS O DE PROGRAMACION..... 18

Mapa conceptual
INTRODUCCIÓN A LA ALGORITMIA



INTRODUCCIÓN

Nuestro actuar como personas se basa en obedecer por medio de ordenes Secuenciales y lógicas a nuestro cerebro; hoy en día un computador realiza tareas y maneja datos obedeciendo secuencias de pasos lógicos para lo cual ha sido programado a través de algoritmos, los cuales permiten solucionar problemas por medio de la escritura secuencial (paso a paso) de lenguaje natural y organizado, para luego llevarlos a un computador mediante un lenguaje de programación.

Es muy importante que usted como aprendiz logre desarrollar y despertar su mentalidad algorítmica, ya que durante todo su proceso estará inmerso en aprender a solucionar problemas durante la etapa lectiva y esto se logra por medio del diseño, creación e implementación de un algoritmo.

1. CONCEPTO DE ALGORITMOS

Un algoritmo se puede definir como un conjunto de instrucciones que conducen a la solución de un problema determinado, las cuales deben estar relacionadas lógica y ordenadamente.

2 UN POCO DE HISTORIA

Los algoritmos fueron creados con la necesidad de resolver problemas matemáticos, sus inicios se dan con la creación de las reglas de sumar, multiplicar y dividir números decimales (Mohammed al-Khowarizmi-matemático persa), de ahí nace el nombre de la "Algoritmo", Su apellido fue traducido al latín como Algorismus y posteriormente paso al español como Algoritmo; más adelante Euclides, gran matemático griego utilizo algoritmos cuando invento el método para encontrar el máximo común divisor de dos números; y por último se puede decir que el gran padre de la algoritmia es Niklaus Wirth el cual tuvo el gran invento de Pascal.



Los algoritmos fueron creados con el fin de utilizarlos para resolver problemas, para luego llevarlos a un computador mediante un lenguaje

de programación; para realizar un algoritmo se debe establecer una metodología de desarrollo, una de estas es la llamada resolución de problemas, la cual permite realizar un diseño organizado del algoritmo.

3 RESOLUCIÓN DE PROBLEMAS CON ALGORITMOS

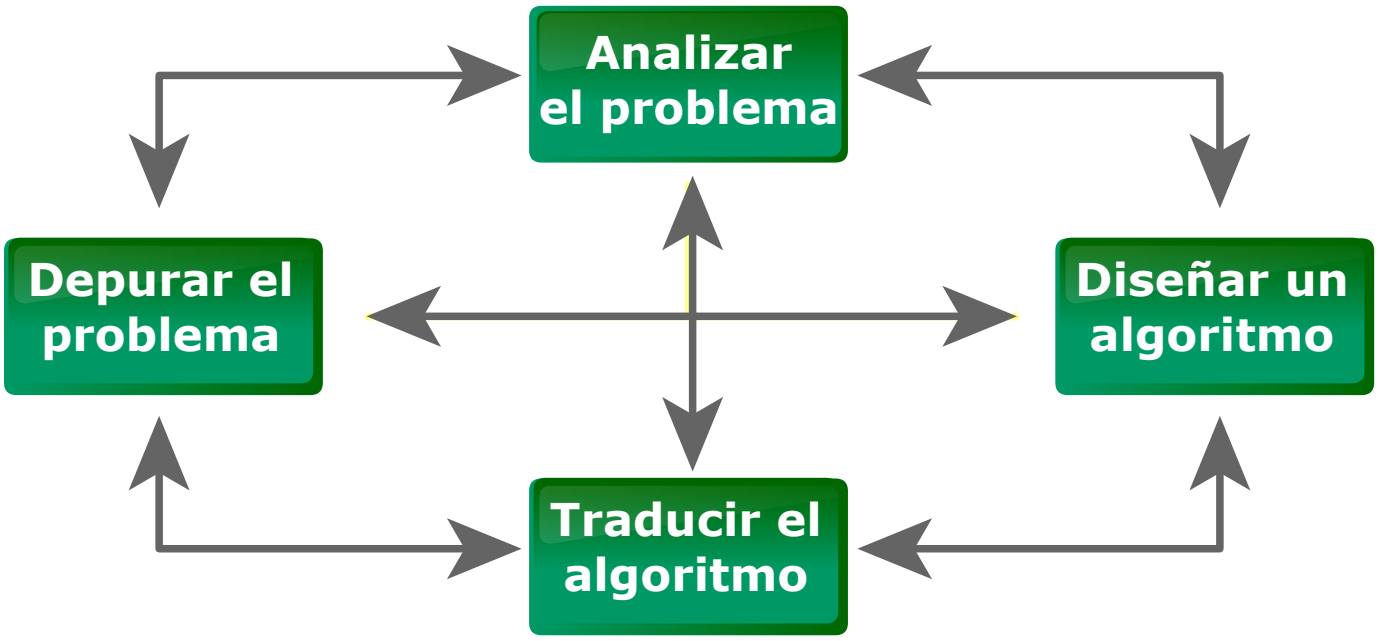
Para solucionar un problema mediante un algoritmo es necesario seguir un orden, uno de los primeros pasos es el diseño previo de un algoritmo, con la resolución de problemas podemos seguir este orden de la siguiente manera:

a. Analizar el problema (entender el problema)

b. Diseñar el algoritmo: En este paso se describe la secuencia ordenada de pasos que conduce a la solución del problema citado (diagrama de flujo o seudocódigo).

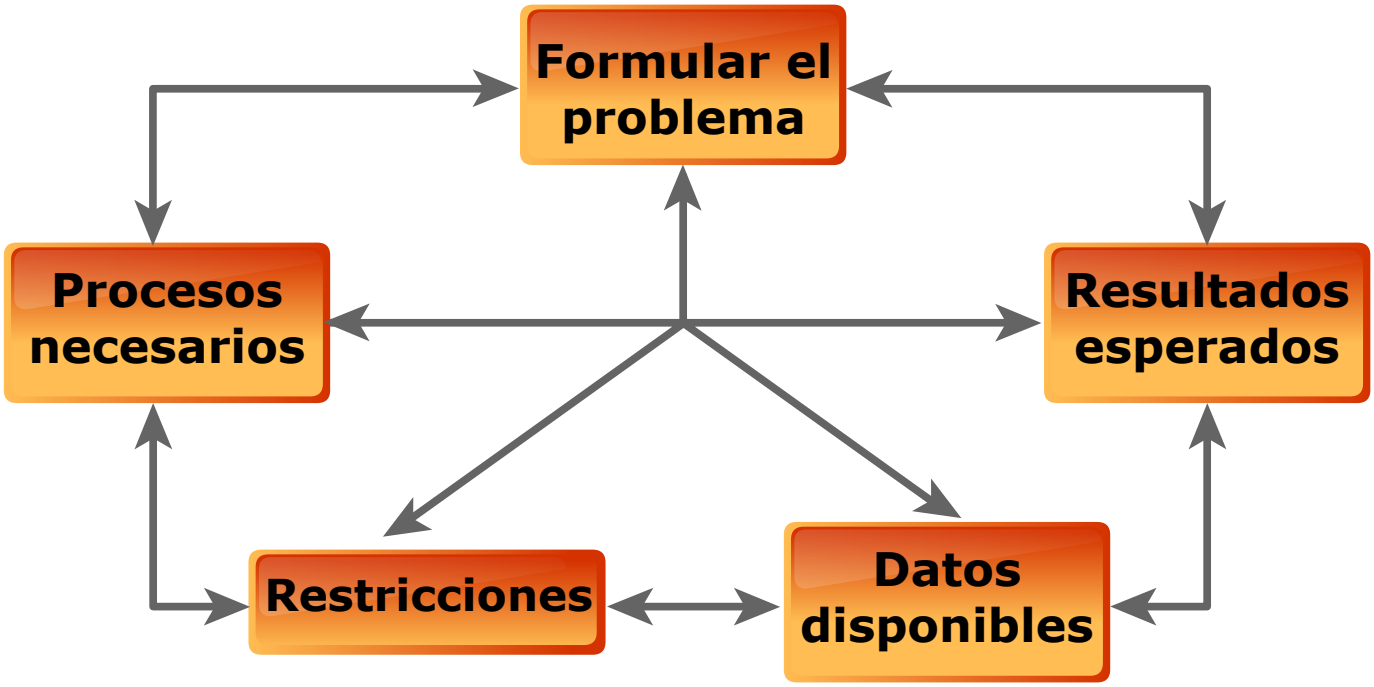
c. Expresar el algoritmo: el algoritmo se debe expresar como un programa en un lenguaje de programación adecuado. (Fase de codificación.)

d. Ejecución y validación: Se pone en ejecución el programa realizado en un computador.



3.1 Analizar el problema

La finalidad como ya se ha mencionado de un algoritmo es resolver un problema, lo primero y esencial es analizar el problema para lograr un entendimiento claro



Reformular el problema: Si el problema que se ha planteado no se encuentra bien formulado, redactado o no se entiende al momento de leerse, debe reformularlo de tal manera que usted lo entienda y tenga claro la situación problema a solucionar).

Resultados Esperados: Debe especificar, describir y/o escribir los resultados que espera, por ejemplo (cuál es el producto final que se quiere tener para dar solución al problema, cual es la información a la que se necesita llegar, que se espera del problema citado)

Datos Disponibles: Identificar la información disponible se resuelve haciéndose las siguientes preguntas: que información es importante o relevante para solucionar el problema? Cuáles son los datos de entrada?, cual es la incógnita?, que información me falta para resolver el problema?,

es decir todos los datos que tengo disponibles que me permitan solucionar el problemas.

Restricciones: Determina las condiciones que plantea el problema para lograr el resultado, lo que está permitido, lo prohibido.

Procesos necesarios: En esta fase debe definir los procesos para poder convertir la información disponible, en resultados esperados que den solución al problema ya que se determinan los procesos que se necesitan, las formulas a utilizar y el orden de lo que se debe realizar.

Ejemplo 1

Carlos Andrés, se encuentra ahorrando para comprar una patineta que vale 55.000 pesos. Su papá le ha dado para sus gastos 5.000 pesos durante 7 semanas. Por lavar el auto de su tío tres veces recibió 8.000 pesos. Su hermano Juan Antonio ganó 10.000 pesos por hacer los mandados de su mamá y 4.000 por sacar a pasear el perro. *¿Carlos Andrés tiene ahorrado el dinero suficiente para comprar la patineta o aún le falta?*

Formular el problema: Ya se encuentra claramente planteado, pero si no entiende la redacción puede redactarlo a su manera, hasta es posible realizar un resumen de lo planteado.

Resultados esperados: Saber si Carlos Andrés tiene o no tiene ahorrado el dinero para comprar su patineta, la cual cuesta 55.000 pesos.

Datos disponibles: Los ingresos de Carlos Andrés 5.000 pesos por 7semanas + 8.000 pesos, los datos irrelevantes serian: Los 10.000 y 4.000 pesos qué ganó el hermano ya que no aportan información para la solución de este problema y se pueden omitir.

Restricciones: no se encuentra ninguna.



Procesos: Calcular el valor ahorrado por Carlos para saber si le alcanza para comprar la patineta.

ValorAhorrado = 5000 + 8000 = 13.000

Es decir no le alcanza para la patineta.

Ejemplo 2

Se necesita calcular el área de un triángulo rectángulo cuya Base mide 3 cm, la Altura 4 cm y la Hipotenusa 5 cm.

Formular el problema: Ya se encuentra claramente planteado, pero si no entiende la redacción puede redactarlo a su manera, hasta es posible realizar un resumen de lo planteado.

Resultados esperados: El área de un triángulo rectángulo.

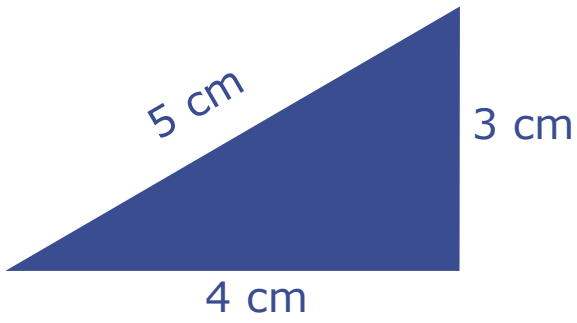
Datos disponibles: Base, Altura, Hipotenusa, tipo de triángulo.

La incógnita es el área y todos los valores son constantes. El valor de la hipotenusa se puede omitir.

El aprendiz debe preguntarse si sus conocimientos actuales de matemáticas le permiten resolver este problema; de no ser así, debe plantear una estrategia para obtener los conocimientos requeridos.

Determinar las restricciones: Utilizar las medidas dadas, y saber que se debe aplicar la fórmula del área de un triángulo rectángulo.

Procesos necesarios: Guardar en dos variables los valores de Base y Altura; Guardar en una constante el divisor 2; aplicar la fórmula área=base*altura/2; comunicar el resultado (área).



3.2 Diseño de un algoritmo

El diseño de un algoritmo se puede realizar mediante un diagrama de flujo o mediante seudocódigo. Los algoritmos tienen las siguientes características

Características de los algoritmos

Un algoritmo debe:

1. Ser preciso e indicar el orden de realización de cada paso.

2. Estar definido. Si se sigue un algoritmo dos veces, se debe obtener el mismo resultado cada vez.

3. Ser finito. Si se sigue un algoritmo, se debe terminar en algún momento.

Un algoritmo de contener como mínimo las siguientes partes:

```
Entrada
Proceso
Salida.
```

Por ejemplo para realizar una receta de comida por medio de un algoritmo, cada parte podrá estar determinada así:

```
Entrada: ingredientes y utensilios empleados.
Proceso: elaboración de la receta en la cocina.
Salida: terminación del plato (por ejemplo, cordero).
```

3.2.1 Diseño de un algoritmo mediante diagrama de flujo

Los algoritmos pueden representarte de varias maneras, entre ellas está la representación de diagramas de flujo. La representación en diagramas de flujo tiene como objetivo seguir paso a paso la solución de un problema mediante gráficos

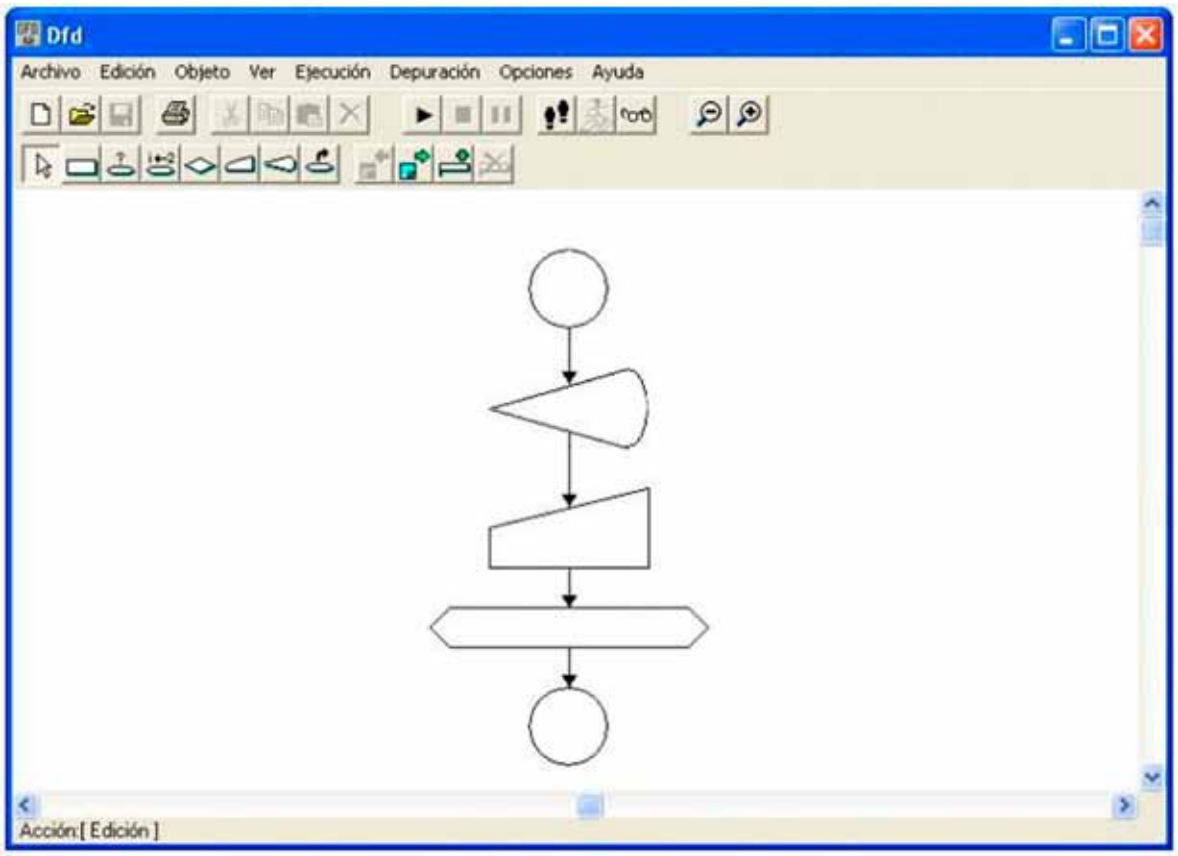
3.2.1.1 Definición Diagrama de flujo

Un diagrama de flujo es un conjunto secuencial de figuras geométricas standard conectadas lógicamente entre sí para dar solución a un problema específico, cada figura tiene un significado propio .

La secuencia lógica se da por medio de flechas llamadas líneas de flujo que indican el flujo lógico del algoritmo. Al ser un diagrama grafico facilita la visión de la ejecución del algoritmo.

La simbología utilizada en estos diagramas ha sido estandarizada por las organizaciones ANSI(American National Institute) y por ISO (International Estándar Organization).

Existen herramientas software que permiten realizar los gráficos mencionados anteriormente, una de las más utilizadas es DFD, el cual es de fácil uso y libre acceso en la web, algunos beneficios de utilizar estos son:



- Almacenar digitalmente los diagramas construidos.
- Introducirles modificaciones fácilmente.
- Imprimir copias de los diagramas para compartirlos con compañeros o documentar sus trabajos.
- Exportarlos en varios formatos gráficos para utilizarlos en otros programas.
- Alinear y organizar los símbolos automáticamente

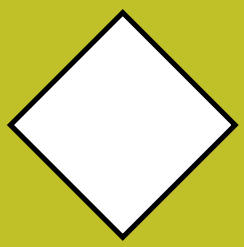
Simbología Estándar de un diagrama de flujo



Para iniciar y terminar un diagrama



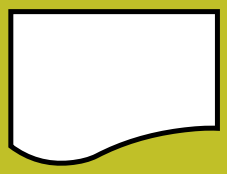
Para representar un proceso



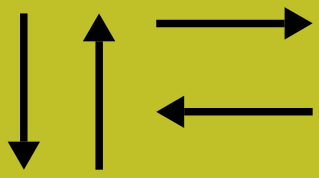
Para toma de decisiones:
utilizado también en estructuras cíclicas



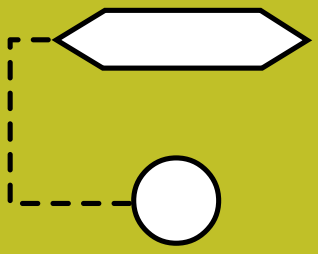
Para representar entrada de datos



Para representar salida de datos



Flechas conectoras



Iteración: Indica que una instrucción o un grupo de instrucciones deben ejecutarse repetida-mente.

3.2.1.2 Reglas para la elaboración de un diagrama de flujo:

1. Los diagramas se deben realizar de arriba hacia abajo y de izquierda hacia derecha.

2. Los símbolos de inicio y final deben aparecer solo una vez

3. La ejecución de un programa siempre empieza en la parte superior del programa.

4. La dirección del flujo se debe representar por medio de flechas.

5. Todas las líneas de flujo deben llegar a un símbolo o a otra línea.

6. Se deben inicializar las variables que se utilicen o permitir la asignación de valores mediante la consulta a un usuario.

3.2.2 Diseño de un algoritmo mediante seudocódigo

La representación de seudocódigo sigue paso a paso la solución de un problema con lenguaje natural, pero recuerde que debe realizar la metodología resolución de problemas para poder iniciar su algoritmo con seudocódigo. Podría recordarse la imagen de resolución de problemas explicada anteriormente.

```
Paso 1: Inicio
Paso 2: ….
Paso 3…
Paso n…Fin
```

Ejemplo1:

Se retoma el ejemplo de Carlos Andrés para llevarlo en forma de algoritmo:

Carlos Andrés, se encuentra ahorrando para comprar una patineta que vale 55.000 pesos. Su papá le ha dado para sus gastos 5.000 pesos durante 7 semanas. Por lavar el auto de su tío tres veces recibió 8.000 pesos. Su hermano Juan Antonio ganó 10.000 pesos por hacer los mandados de su mamá y 4.000 por sacar a pasear el perro. ¿Carlos Andrés tiene ahorrado el dinero suficiente para comprar la patineta o aún le falta?

```
Paso 1: Inicio
Paso 2: Asignarle el valor del dinero ahorrado por Carlos Andres a la Variable ValorAhorrado = 5.000+8000
Paso 3: Mostrar el ValorAhorrado, indicando si le alcanza o no para la patineta.
Paso 4: Fin.
```

Ejemplo 2

Se retoma el ejemplo anterior también por continuidad.

Se necesita calcular el área de un triángulo rectángulo cuya Base mide 3 cm, la Altura 4 cm y la Hipotenusa 5 cm.

```
Paso1:Inicio
Paso2:Indicar que la variable Base tiene un valor de 3cm
Paso3: Indicar que la variable Altura es de 4cm
Paso4: Calcular el área a=( Bases*Altura )/2
Paso 5 : Mostar el área
Paso 6 Fin.
```

Ejemplo 3

También se pueden resolver problemas cotidianos: por medio de seudocódigo escriba un algoritmo para poder pasarse los semáforos, una vez usted se encuentra esperando pasar como peatonal.

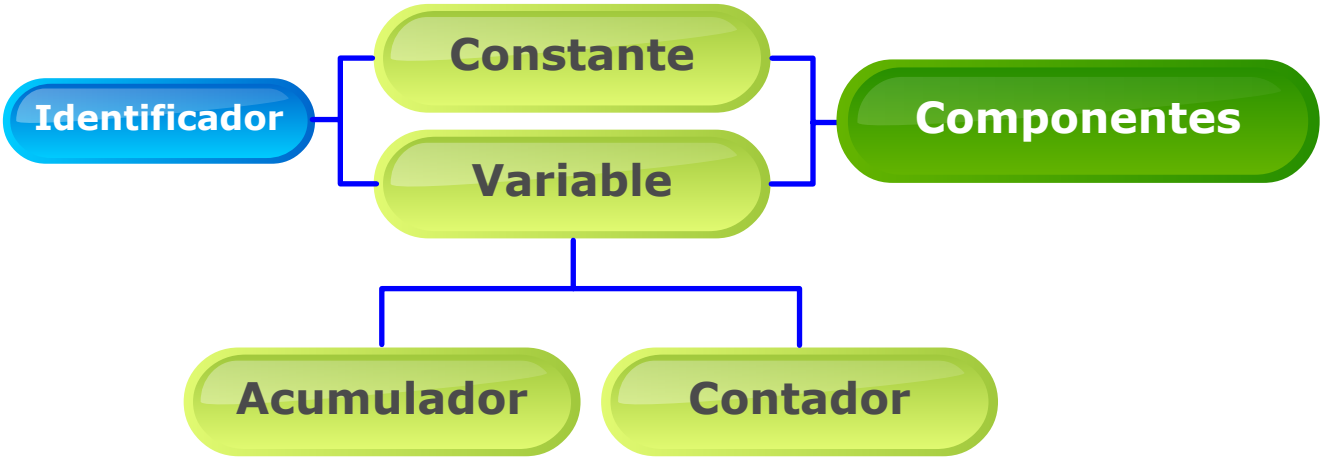
```
Paso1:Inicio
Paso2:Ver el color del semáforo
Paso3: Si el semáforo esta en rojo : hay que detenerse
Paso 4: Mostar mensaje Hay que detenerse.
Paso 5: Si el semáforo esta en amarillo: alistarse para pasar
Paso 6: Mostar mensaje alistarse.
Paso7: Si el semáforo esta en verde: pasarse la calle.
Paso 8: Mostar mensaje puede pasarse y la persona se pasa la calle.
Paso 0 Fin.
```

3.3 Expresar el algoritmo en un lenguaje de programación:

Una vez el algoritmo este diseñado en diagrama de flujo o en lenguaje natural (seudocódigo), se pone en marcha la "traducción" de este a un lenguaje de programación especifico.

Existen diferentes tipos de lenguajes que permiten "traducir" el algoritmo para que pueda ser entendido por el computador entre los cuales son java, c#, visual basic, entre otros, los cuales son tratados específicamente más adelante.

4 SOLUCIÓN DE PROBLEMAS CON DIAGRAMAS DE FLUJO Y LENGUAJE NATURAL (SEUDOCÓDIGO).



Recuerde cuando se plantee un problema ya sabe que puede resolverlo por medio de diagrama de flujo o por lenguaje natural.

Es importante conocer y aprender a utilizar los siguientes conceptos que están inmersos al momento de realizar algoritmos.

4.1.1 Variables:

Recordemos que cuando se inicia con el análisis del problema se identifican los datos iniciales, estos datos se estructuran o se definen como variables;

en ellas se pueden almacenar valores y son nombradas con identificadores, es decir nombres para poder identificarlas dentro del algoritmo. Por ejemplo, si en el problema de calcular el área de un triángulo tengo como datos iniciales la base y la altura, estas dos anteriores serian llamadas variables y sus nombres serán "BASE" y "ALTURA".

Un aspecto importante de las variables es que pueden cambiar su valor durante la ejecución del algoritmo.

Se debe tener en cuenta que una variable puede ser declarada, asignada o solicitada mediante un algoritmo para poder ser utilizada.

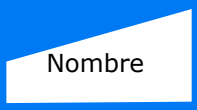
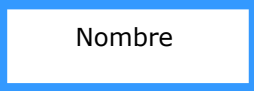
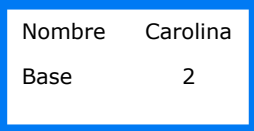
Declaración de una variable: Para utilizar una variable tanto en seudocódigo como en diagrama de flujo es necesario siempre declararla, es decir indicarle al algoritmo que va a utilizar una variable por ejemplo nombre , quiere decir que usted necesita una variable que almacene los nombres de las personas.

Asignación de valor a una variable: Luego de declararla puede usted necesitar que esa variable inicie por defecto con un valor por ejemplo Saldo = 2000, quiere decir que utilizara la variable saldo pero que esta inicializada con un valor de 2000, puede inicializar una variable según el valor que necesite para solucionar el problema.

Asignarle un valor a una variable, constante, acumulador o contador, expresiones complejas o simples, por ejemplo.

```
Variable = expresión
Por ejemplo: Nombre = Carmen
Saldo = 2000
Área = (Base*altura)
```

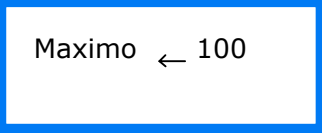
Solicitar una variable: Cuando se necesita que el valor de la variable pueda ser ingresado por un usuario al computador, entonces debe solicitarse este valor de la variable.

Diagrama de flujo	Uso	Seudocódigo
Solicitar una variable 	Cuando se necesita que el valor de la variable pueda ser ingresado por un usuario al computador.	Solicitar una variable cont=0 Leer Nombre Conocer Nombre
Declarar una variable 	Indica al algoritmo que va a utilizar una variable por ejemplo nombre, esta variable almacena nombres.	Declarar una variable Declarar Variable Nombre
Asignarle valor a una variable 	Para asignarle un valor a una variable, constante, acumulador o contador, expresiones complejas o simples.	Asignarle valor a una variable Nombre= Carolina Base = 2

4.1.2 Constantes:

Almacenan datos al igual que las variables, pero su gran particularidad es que sus datos no cambian durante la ejecución del algoritmo, es decir siempre el valor de la constante va ser el mismo.

Las constantes se deben declarar e inicializar.

Diagrama de flujo	Seudocódigo
Declarar Una constante e inicializarla 	Declarar Una constante e inicializarla Constante Máximo =100

4.1.3 Contadores:

Permiten almacenar valores con el objetivo de llevar a cabo un conteo en las secuencias de ejecución.

Como su palabra lo dice permiten contar, para poder utilizar un contador es necesario inicializarlo en un valor y luego incrementar su valor de una manera constante para permitir realizar el conteo.

Diagrama de flujo	Seudocódigo
Declarar e iniciar un contador cont ← 0 Utlizar contador cont ← cont+1	Declarar e inicializar un contador cont=0 Utilizar un contador cont=cont+1

4.1.4 Acumuladores:

Permiten almacenar el valor de una variable "A", sumando el valor de ella más el valor de una variable "B"

Como su palabra lo dice permiten acumular el valor de una variable, para poder utilizar un acumulador es necesario inicializarlo en un valor y luego iniciar con la acumulación del valor.

Diagrama de flujo	Seudocódigo
Declarar e iniciar un contador cont ← 0 Utlizar contador cont ← cont+1	Declarar e inicializar un contador acum=0 Utilizar un contador acum=acum+suma

4.1.5 Identificadores:

Los identificadores son nombres que se dan a las variables, constantes, acumuladores y contadores para así poder diferenciarlos. Para asignar los nombres se debe tener en cuenta lo siguiente:

- Los nombres pueden estar formados por una combinación de letras y números (saldoMes, salario, fecha2, baseTriángulo, etc).
- El primer carácter de un nombre debe ser una letra.
- La mayoría de los lenguajes de programación diferencian las mayúsculas de las minúsculas.
- Los nombres deben ser nemotécnicos, con solo leerlos se puede entender lo que contienen. Deben ser muy descriptivos; no utilizar abreviaturas, a menos que se justifique plenamente.
- No utilizar caracteres reservados (%, +, /, >, etc).
- No utilizar palabras reservadas por los lenguajes de programación.
- Para cumplir con convenciones ampliamente utilizadas (Jiménez, 2002), los nombres de procedimientos, variables y constantes deben empezar con minúscula. Ejemplo, fecha, suma, etc. Si es un nombre compuesto por varias palabras, cada una de las palabras (con excepción de la primera) debe empezar con mayúscula. Ejemplo: fechaInicial, baseTriángulo, etc.

5. ESTRUCTURAS ALGORÍTMICAS O DE PROGRAMACION

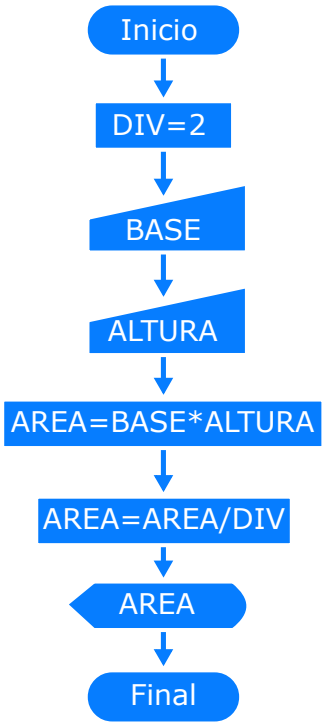
Las estructuras algorítmicas están creadas para orientarnos en la forma de diseñar algoritmos, cada una de ellas representa cierto concepto que permite lograr encontrar la solución del problema, entre ellas tenemos:

5.1 Estructura secuencial

Es una de las estructuras más sencilla, conocida también como estructura lineal y se compone de instrucciones que deben ejecutarse secuencialmente.

Diagrama de flujo	Seudocódigo
acción 1 ↓ acción 2 ↓ acción n	**Inicio** <accion1> <accion2> * * * <accionN> **Fin**

Correspondencia de seudocódigo a diagrama de flujo



Paso 1: Inicio

Paso 2: Asingar el numero 2 a la constante DIV

Paso 3: Saber la base del triángulo y guardarlo en la variable "BASE"

Paso 4: Saber la altura del triángulo y guardarla en la variable "ALTURA"

Paso 5: Guardar en la variable "AREA" el valor de BASE*ALTURA

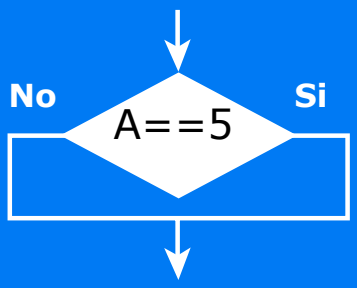
Paso 6: Guardar en la variable "AREA" la división entre el "AREA " sobre la variable "DIV"

Paso 7: Mostar el valor de área.

Paso 8: Fin

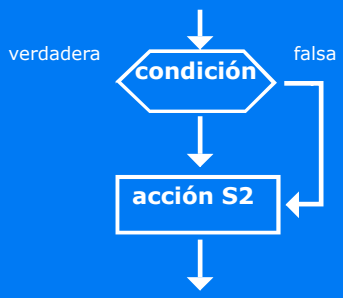
5.2 Estructuras de decisión

Son utilizadas para tomar decisiones lógicas, llamadas también estructuras selectivas o alternativas, en ellas se evalúa una condición y en función del resultado de la misma se realiza una opción u otra.

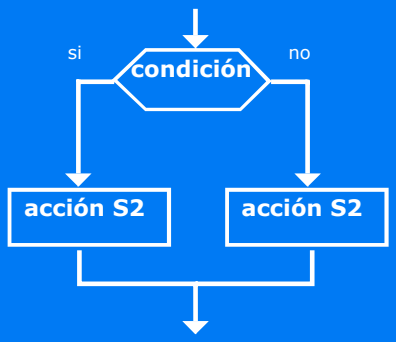
Diagrama de flujo	Seudocódigo
	**Inicio** Si condición entonces ... . **Fin-si** **Else condición**

Las estructuras de decisión pueden ser:

Simples: (llamada si-entonces), ejecuta una determinada acción cuando se cumple una determinada condición. La selección si-entonces evalúa la condición y si la condición es verdadera, entonces ejecuta la acción; si la condición es false entonces no hace nada.

Diagrama de flujo	Seudocódigo
	**Inicio** Si <condición> <acción> else fin-si

Dobles: La estructura anterior es muy limitada y normalmente se necesitará una estructura que permita elegir entre dos opciones o alternativas posibles, en función del cumplimiento o no de una determinada condición

Diagrama de flujo	Seudocódigo
	Si <condición> entonces <accion S1> si_no <accion S2> fin-si

5.3 Estructuras cíclicas y/o repetitivas

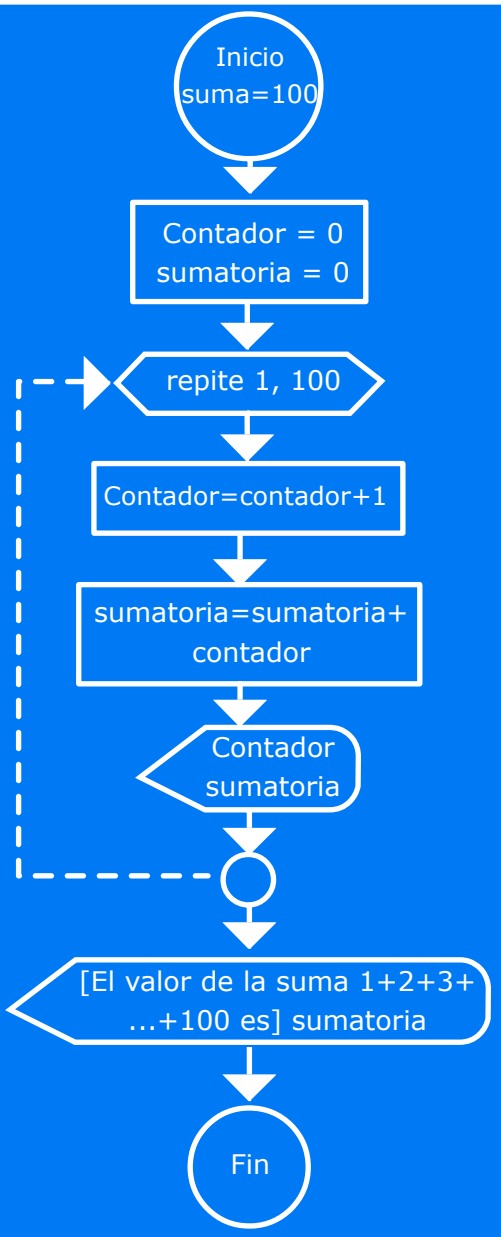
Un tipo muy importante de estructura es el algoritmo necesario para repetir una o varias acciones un número determinado de veces. Las estructuras que repiten una secuencia de instrucciones un número determinado de veces se denominan bucles, y se llama iteración al hecho de repetir la ejecución de una secuencia de acciones.

Y así sucesivamente para cada número de la lista. En otras palabras, el algoritmo repite muchas veces las acciones.

Tales opciones repetidas se denominan bucles o lazos. La acción (o acciones) que se repite en un bucle se denomina iteración. Las dos principales preguntas a realizarse en el diseño de un bucle son: ¿qué contiene el bucle? y ¿cuántas veces se debe repetir?

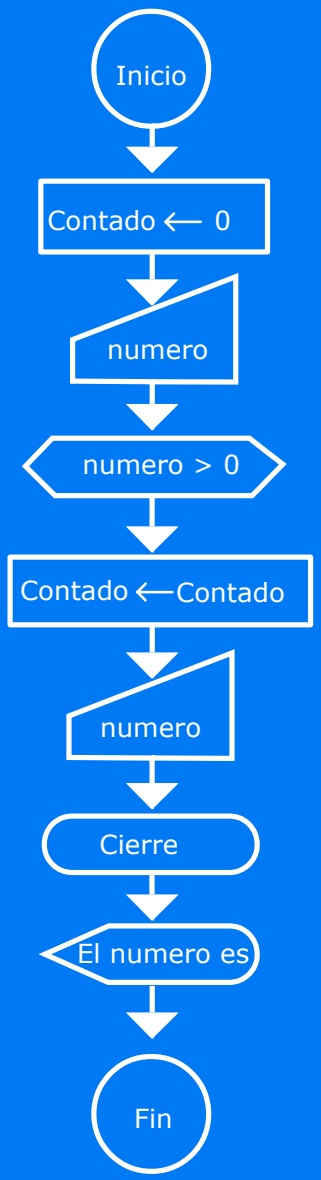
Se manejan entonces 3 tipos de estructuras cíclicas que permiten dar respuesta a los anteriores interrogantes

5.3.1 Estructura para

Diagrama de flujo	Seudocódigo
	Inicio suma=100 contador=100 sumatoria=0 Para 1 hasta 100 contador=contador=1 sumatoria=sumatoria+contador Mostrar contador Mostrar sumatoria fin-para Mostrar "el valor de la suma 1+2+3+...+100 es" sumatoria Fin

En muchas ocasiones se conoce de antemano el número de veces que se desean ejecutar las acciones de un bucle. Esta estructura permite ejecutar una o varias instrucciones un determinado de veces finita, fija.

5.3.2 Estructura mientras

Diagrama de flujo	Seudocódigo
	Inicio contador=0 leer numero mientras numero>0 hacer contador = contador+1 leer numero fin-mientras escribir "El número de enteros positivos es" contador Fin

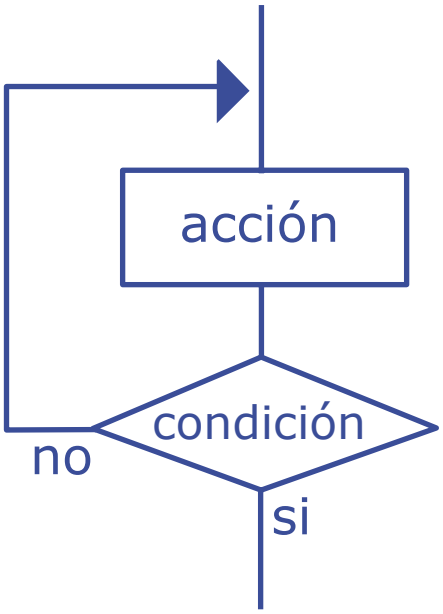
Se llama Mientras a la estructura algorítmica que se ejecuta mientras la condición evaluada resulte verdadera. Se evalúa la expresión booleana y, si es cierta, se ejecuta la instrucción especificada, llamada el cuerpo del bucle. Entonces se vuelve a evaluar la expresión booleana, y si todavía

es cierta se ejecuta de nuevo el cuerpo. Este proceso de evaluación de la expresión booleana y ejecución del cuerpo se repite mientras la expresión sea cierta.

Cuando se ejecuta la instrucción mientras, la primera cosa que sucede es que se evalúa la condición (una expresión booleana). Si se evalúa falsa, ninguna acción se toma y el programa prosigue en la siguiente instrucción del bucle. Si la expresión booleana es verdadera, entonces se ejecuta el cuerpo del bucle, después de lo cual se evalúa de nuevo la expresión booleana. Este proceso se repite una y otra vez mientras la expresión booleana (condición) sea verdadera.

5.3.3 Estructura repita

La estructura cíclica REPITA, al igual que la estructura cíclica mientras, se ejecuta un número indeterminado de veces, estas dos estructuras tienen un comportamiento similar, presentando su principal diferencia en el lugar de estructura donde se evalúa la condición, dado que la estructura MIENTRAS evalúa la condición del ciclo al inicio del mismo y la estructura REPITA lo hace al final del mismo, de este modo, en la estructura cíclica REPITA, el programador garantiza que el ciclo se ejecuta al menos una vez.



GLOSARIO

ALGORITMO: Es un conjunto de instrucciones que conducen a la solución de un problema determinado, las cuales deben estar relacionadas lógica y ordenadamente.

ANSI (American National Standards Institute): El Instituto Nacional Estadounidense de Estándares, es una organización sin ánimo de lucro que supervisa el desarrollo de estándares para productos, servicios, procesos y sistemas en los Estados Unidos.

ISO (International Organization for Standardization): La Organización Internacional para la Estandarización, es una federación mundial que agrupa a representantes de cada uno de los organismos nacionales de estandarización (como lo es el ICONTEC en Colombia), y que tiene como objeto desarrollar estándares internacionales que faciliten el comercio internacional entre otros.

SEUDOCODIGO: Es una descripción informal de alto nivel de un algoritmo, que utiliza las convenciones estructurales de un lenguaje de programación verdadero, pero que está diseñado para la lectura.

BIBLIOGRAFÍA

[1]. JOYANES Luis. Fundamentos de programación, Algoritmos y estructura de datos. McGraw – Hill

[2] Lopes Garcia, Juan Carlos. Algoritmos Y Programación (Guía Para Docentes), Segunda Edición, 2007, 2009.

Control de documento Construcción Objeto de Aprendizaje Introducción a la Algoritmia	
Desarrollador de contenido Experto temático	Leydy Carolina Muñoz
Asesor pedagógico	Rafael Neftalí Lizcano Reyes Claudia Milena Hernández
Producción Multimedia	Luis Fernando Botero Mendoza Victor Hugo Tabares
Programadores	Daniel Eduardo Martínez
Líder expertos temáticos	Ana Yaqueline Chavarro Parra
Líder línea de producción	Santiago Lozada Garcés



RETO JUEGO DE BLACKJACK

En esta actividad deberán programar un juego básico de Blackjack. Las soluciones que se pueden aportar son muchísimas y queda abierto a la imaginación de cada programador. El juego debe contener lo siguiente:

1. El juego deberá permitir barajar las 52 cartas aleatoriamente
2. El juego debe permitir repartir cartas (las primeras 2 cartas del maso barajado son para el jugador) para 1 jugador que va a jugar contra la casa o el Crupier en un casino.
3. Se debe verificar si desde la partida el jugador ya hizo Blackjack (sumar 21 puntos).
4. Se debe permitir al jugador tomar cartas de la baraja y añadirlas a su mano, sebe contar cuantas cartas va tomando.
5. El juego debe permitir que una vez Jugador ya ha terminado y le toca el turno al Croupier, salvo que previamente el Jugador hubiera tenido Blackjack o se hubiera pasado, en cuyo caso el juego finaliza.

Aquí el resultado de una posible solución.



Se debe consultar para tener en cuenta las reglas básicas del Blackjack.

Análisis y Desarrollo de Software

ADSO

El empleo es de todos | Mintrabajo

Evidencias de Conocimientos:

GA3-220501093-AA1-EV01: Bases conceptuales de lógica proposicional.

GA3-220501093-AA1-EV02: Taller de resolución de problemas de algoritmos en pseudocódigo y diagramas de flujo.

Por:
Gustavo Salazar
Instructor Técnico

Centro De Comercio y Servicios
Regional CAUCA

Análisis y Desarrollo de Software ADSO

GA3-220501093-AA1-EV01:

Bases conceptuales de lógica proposicional.

GA3-220501093-AA1-EV02:

Taller de resolución de problemas de algoritmos en pseudocódigo y diagramas de flujo



Análisis y Desarrollo de Software

AGENDA

- BIENVENIDA
- Socialización Evidencia **GA3-220501093-AA1-EV01** y **GA3-220501093-AA1-EV02**
- Resolución de inquietudes
- Conclusiones

Análisis y Desarrollo de Software ADSO

1- Competencia 220501093 - Evaluar requisitos de la solución de software de acuerdo con metodologías de análisis y estándares

Actividad e Aprendizaje GA3-220501093-AA1: Realizar actividades para el desarrollo del pensamiento algorítmico

Análisis y Desarrollo de Software ADSO

ACTIVIDAD DE APRENDIZAJE	RESULTADO DE APRENDIZAJE (RAP)	Evidencia a Aprobar	Fecha de Entrega
GA3-220501093-AA1: Realizar actividades para el desarrollo del pensamiento algorítmico.	**220501093-03** - Desarrollar procesos lógicos a través de la implementación de algoritmos.	**GA3-220501093-AA1-EV01**: Bases conceptuales de lógica proposicional. **GA3-220501093-AA1-EV02**: Taller de resolución de problemas de algoritmos en pseudocódigo y diagramas de flujo.	06/09/2022 a 04/10 /2022

Análisis y Desarrollo de Software ADSO

Evidencias de Conocimiento

GA3-220501093-AA1-EV01

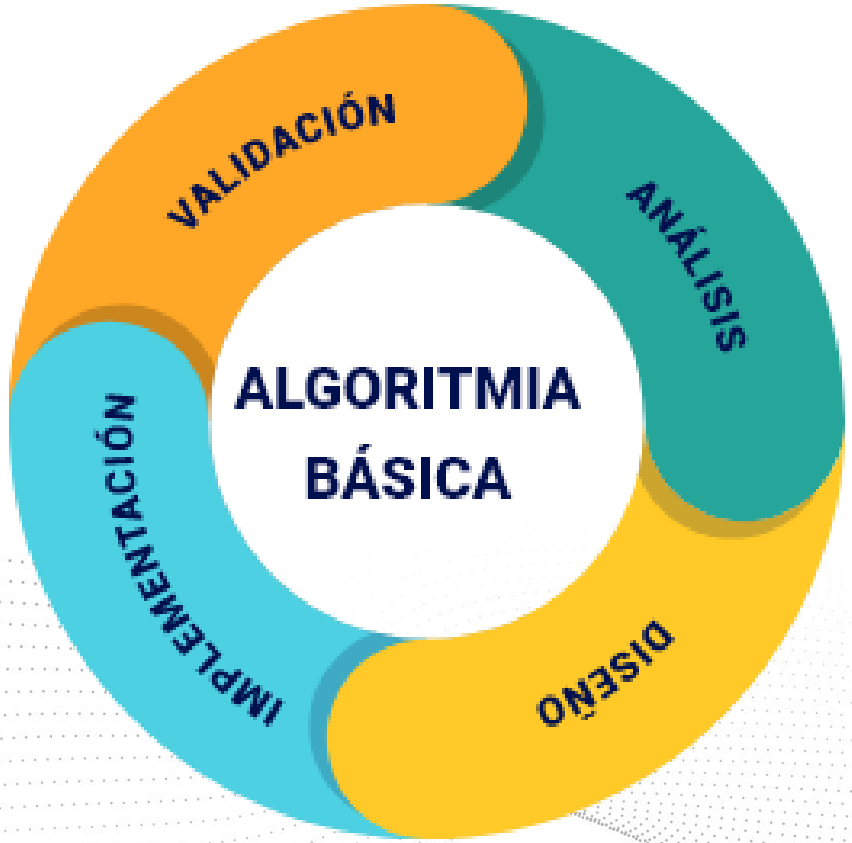
GA3-220501093-AA1-EV02



Materiales de formación: para el desarrollo de estas actividades, es importante la lectura y el análisis del componente formativo

“**Introducción a la algoritmia**”.

1. Introducción a la algoritmia



Es la aplicación apropiada de una serie de pasos detallados que garantizan una solución correcta de un problema;

La programación de computadores o construcción de software se basa en esta práctica.

Definición de algoíitmo

Un algoritmo está definido como un “**conjunto ordenado y finito de operaciones que permite hallar la solución de un problema**”.

Computacionalmente las máquinas están en capacidad de hacer unas pocas operaciones matemáticas muy básicas y sencillas, Se considera que computan enormes cantidades de estas operaciones en unidades de tiempo muy corto.



1. Introducción a la algoritmia

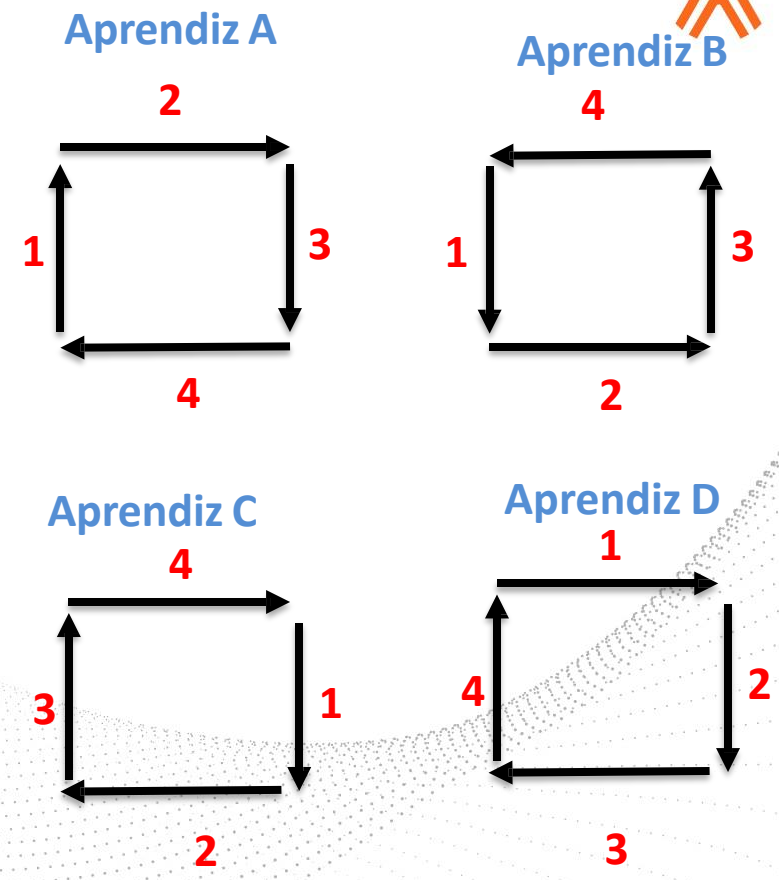


1. Introducción a la algoritmia

Pensamiento algorítmico

Un problema puede ser abordado de vaíias maneías y **apoítaí la misma solución**, peío lo que siempíe tendíán en común es que paía íesolveílo usaíon una **secuencia de pasos oídenados**.

Por ejemplo, si se pasa una **hoja de papel cuadriculado** a los aprendices de una clase y se les pide que **dibujen un cuadrado** sin levantar el lápiz, habrá quienes lo hagan de diferente manera y obtengan el mismo resultado como lo muestra la siguiente figura:



Existe una gran cantidad de soluciones al mismo problema.

1. Introducción a la algoritmia

En un problema más complejo se concluye que existen tantas soluciones como el enfoque o lógica empleada por cada aprendiz o persona. Pero lo que sí es común en todos los casos es que se realizaron **unas fases mentales secuencialmente ordenadas**, las cuales son:

1. Entender el problema enunciado por el instructor (**entender el problema**).
2. Escoger una ruta a seguir del rectángulo (**hacer un plan**).
3. Plasmar en el papel cuadriculado ese plan (**ejecutar el plan**).
4. Revisar el resultado y ver que se alcanzó el objetivo (**revisar**).



Cuando el problema es muy complejo de resolver, muchas veces toca **volver y analizar el problema**, o trazar un nuevo plan o plasmar ese nuevo plan una y otra vez, revisando los resultados y volviendo a analizar hasta encontrar la solución del problema.

Solución de problemas y programación

"Los algoritmos son una herramienta que permite descíibií claíamente un conjunto finito de instíucciones"



"En el ámbito de la computación **los algoritmos son una herramienta** que permite describir claramente **un conjunto finito de instrucciones ordenadas secuencialmente y libres de ambigüedad**, que debe llevar a cabo un computador para lograr un resultado previsible. Vale la pena recordar que un programa de computador consiste de una serie de instrucciones muy precisas y escritas en un lenguaje de programación que el computador entiende (Logo, Java, Pascal, etc.)"

1. Introducción a la algoritmia

Ejemplo 1: se propone diseñar un algoritmo para apagar una computadora.

```
Algoritmo: apagar computadora.
INICIO
1.-Dar clic en el icono INICIO.
  2.-Dar clic en el icono que dice APAGAR.
    3.-Esperar que se apague completamente.
      4.-Desconectar los cables del enchufe.
FIN
```

Ejemplo 2: se propone diseñar un algoritmo para cargar un celular.

```
Algoritmo: cargar un celular.
INICIO
1. Buscar el cargador.
  2. Agarrar el cargador.
    3. Agarrar el celular.
      4. Poner el enchufe en el celular.
        5. Conectarlo al tomacorriente.
          6. Esperar a que se ponga a cargar.
            7. Desconectarlo del tomacorriente.
              8. Ponerlo en un lugar seguro.
FIN
```

1. Introducción a la algoritmia



algoritmo debe ser



Realizable:
El **proceso algorítmico** debe terminar después de una cantidad finita de pasos. Se dice que **un algoritmo es inaplicable** cuando se ejecuta con un conjunto de datos iniciales y **el proceso resulta infinito** o durante la ejecución se encuentra con un obstáculo insuperable sin arrojar un resultado.

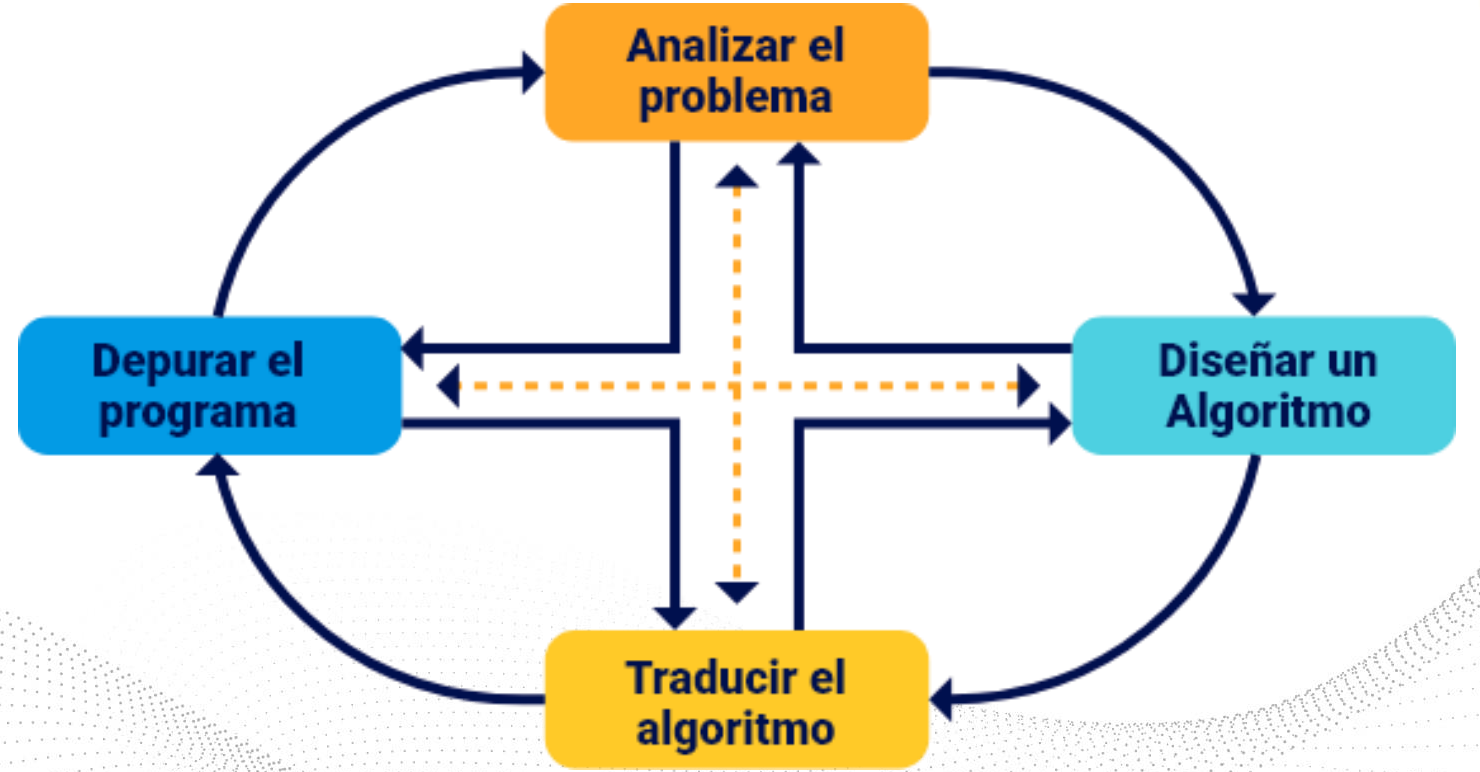
Comprensible:
Debe ser claro lo que hace, de forma que quien ejecute los pasos (ser humano o máquina) sepa **qué, cómo y cuándo hacerlo**. Debe existir un procedimiento que determine el proceso de ejecución.



Preciso:
El orden de ejecución de las instrucciones debe estar perfectamente indicado. Cuando se ejecuta varias veces, con los mismos datos iniciales, **el resultado debe ser el mismo siempre**. La precisión implica determinismo

1. Introducción a la algoritmia

Fases para elaborar un programa de computador



El proceso de diseño de algoritmos de programas informáticos, están basados en un algoritmo que posteriormente es escrito en un lenguaje de programación

1. Introducción a la algoritmia

Anfilisis del píoblema (entendeílo)

Como los programas de computador tienen por objetivo **resolver problemas muy puntuales**, lo primero que se debe realizar para resolverlos es lograr una mejor comprensión posible de estos.

Paía íealizaí esta actividad se debe:

1. **Formular claramente el problema.**
2. **Especificar los resultados que se desean obtener con la solución.**
3. **Identificar la información disponible en los datos.**



1. Formular el problema

A continuación, algunas preguntas previas a la formulación del problema:

- ✓ ¿Es posible definir de forma más clara el problema?
- ✓ ¿Qué palabras no son conocidas o se desconoce su significado?
- ✓ ¿Se ha resuelto antes algún problema similar?
- ✓ ¿Qué información es importante para resolver el problema?
- ✓ ¿Qué información no es importante y se puede omitir?

1. Introducción a la algoritmia

Ejemplo
Don Juan necesita decidir cómo comprar un teléfono celular que cuesta **$870.000 de contado** o **$980.000 a crédito**; él tiene $630.000 pesos en efectivo.

Luego de revisar las preguntas previas se puede concluir:

Como el efectivo que tiene don Juan no le alcanza para comprar el teléfono celular de contado, entonces él tiene dos opciones**: comprarlo completamente a crédito** o también **pagar una parte de contado (como cuota inicial) y el resto diferirlo a crédito**.

Para poder resolver este problema se debe conocer **el número de cuotas en el que difiere si desea pagarlo totalmente a crédito** o conocer el número de cuotas y el valor total del celular si se da una cuota inicial de **$630.000 pesos**.



2. Precisar elementos de entrada

Otro punto importante en la fase de análisis del problema es determinar cuál es la información disponible, por lo tanto:

- ✓ ¿Qué información es importante y necesaria para resolver el problema?
- ✓ ¿Qué información no es importante y se puede prescindir?
- ✓ ¿Cuáles son los datos de entrada conocidos?
- ✓ ¿Cuál es la incógnita o qué se debe calcular?
- ✓ ¿Los datos se pueden agrupar en categorías?
- ✓ ¿Qué información adicional hace falta para resolver el problema?

3. Precisar resultados esperados

Para determinar con claridad cuál es el resultado final esperado (o producto) que se quiere obtener es necesario determinar o definir qué resultados se solicitan y, en qué tipo o formato deben estar (mostrados en pantalla, diagramación, orden, etc.). Para ello, es importante tener en cuenta:

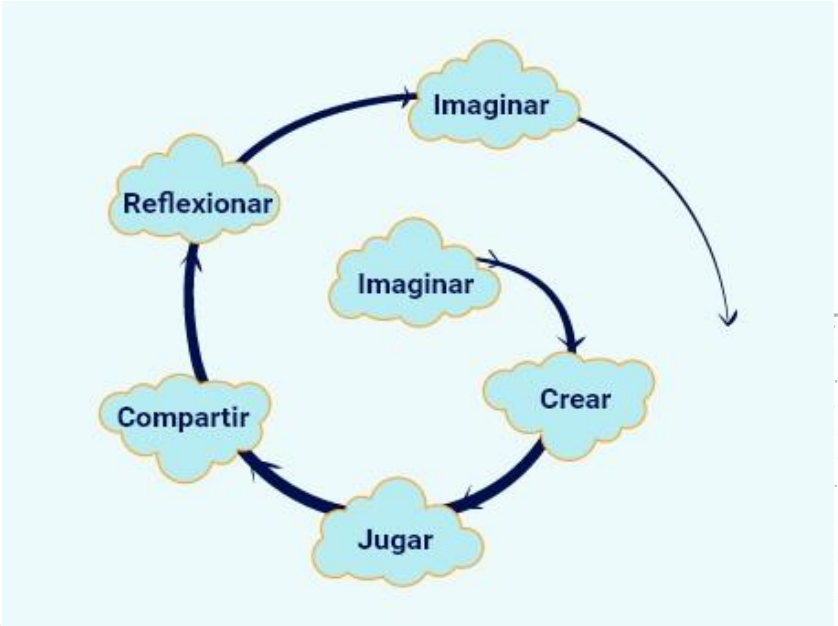
- ✓ ¿Qué información están solicitando?
- ✓ ¿En qué formato se debe entregar esta información?¿Qué información adicional hace falta para resolver el problema?

1. Introducción a la algoritmia

Desarrollo de la creatividad: elementos, modelos, fases y sus objetivos

Resolver algoritmos es una actividad que requiere en la mayoría de los casos **creatividad por parte del programador**, esta es una habilidad que se va fortaleciendo en la medida que se desempeña más y más en procesos de desarrollo de software.

Para estimular "el pensamiento creativo propuesto por Resnick (2007) **el programador debe imaginar qué quiere hacer**; crea un proyecto basado en sus ideas; juega y comparte sus ideas y creaciones con otros para reflexionar sobre sus experiencias y permitirse imaginar nuevas ideas y nuevos proyectos. La espiral genera **un proceso indefinido de mejoramiento continuo**"



Espiral del Pensamiento Creativo diseñada por el Dr. Mitchel Resnick

En conclusión

Un algoritmo es la definición de una **secuencia lógica de pasos que permiten solucionar un problema**



Formular algoritmos es la base de los grandes adelantos en las ciencias matemáticas y la tecnología, porque se pueden aplicar a cualquier área del conocimiento.

2. Lógica matemática

La lógica matemática es la disciplina que trata sobre los métodos de razonamiento. En un nivel elemental, la lógica proporciona reglas y técnicas para determinar si es o no válido un argumento dado.

El razonamiento lógico se emplea en matemáticas para demostrar teoremas; en ciencias de la computación para verificar si son o no correctos los programas; en las ciencias física y naturales, para sacar conclusiones de experimentos; y en las ciencias sociales y en la vida cotidiana, para resolver una multitud de problemas. Ciertamente se usa en forma constante el razonamiento lógico para realizar cualquier actividad.



2. Lógica matemática



2. Lógica matemática

Componentes de la lógica matemática

Todo razonamiento lógico está compuesto por una serie de elementos que se interrelacionan entre sí, y la correcta interpretación de estos componentes y de sus relaciones corresponde al análisis deductivo. **Estos componentes son las preposiciones y conectores lógicos**



2. Lógica matemática

Proposiciones

Es una oración que puede ser **falsa o verdadera**, pero no ambas a la vez.

Las proposiciones pueden **ser simples o compuestas**. Las primeras están formadas por una sola proposición, por el contrario, las proposiciones compuestas están conformadas por dos o más proposiciones unidas por un conector.

proposiciones o Enunciados

p: la tierra es plana.

q: -17 + 38 = 21

r: hola ¿cómo estás?

S: lava el carro por favor.

p y **q** son proposiciones simples válidas porque pueden tomar un valor verdadero o falso; sin embargo, los enunciados r y s son proposiciones no válidas, porque no pueden tomar un valor de falso o verdadero, uno de ellos es un saludo y el otro es una orden.

2. Lógica matemática

Conectores lógicos y proposiciones compuestas

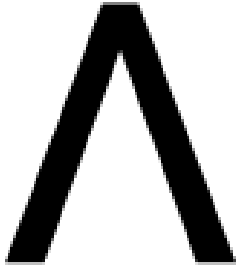
Existen las proposiciones compuestas que son las que se componen de varias proposiciones, las cuales utilizan para su conexión **operadores o conectores lógicos**

Operador AND (y) conjunción

Conecta dos proposiciones que se deben unir para que se pueda obtener un resultado verdadero, se representa por el siguiente símbolo ∧.

Ejemplos:

- ✓ Nuestra bandera es de color amarillo, azul **y** rojo.
- ✓ El número 9 es mayor que 7 **y** menor que 12.
- ✓ El carro enciende cuando tiene corriente eléctrica **y** gasolina en el tanque.



2. Lógica matemática

Conectores lógicos y proposiciones compuestas

Existen las proposiciones compuestas que son las que se componen de varias proposiciones, las cuales utilizan para su conexión **operadores o conectores lógicos**

Operador OR (o) disyunción

Conecta dos proposiciones y se obtiene una proposición compuesta verdadera cuando por lo menos unas de las proposiciones son verdaderas., se representa por el siguiente símbolo **V**

Ejemplo:

- ✓ Puedo ir a Bogotá en avión **o** en bus.
- ✓ Puedo entrar al estadio si compro mi boleta **o** si tengo un pase de cortesía.
- ✓ Voy a desayunar con huevos revueltos **o** huevos pericos.

V

2. Lógica matemática

Conectores lógicos y proposiciones compuestas

Existen las proposiciones compuestas que son las que se componen de varias proposiciones, las cuales utilizan para su conexión **operadores o conectores lógicos**

Operador NOT (no) negación

Su propósito es negar una proposición, es decir, que si tenemos una proposición verdadera al aplicarle este tipo de operador se obtiene la negación, es decir, es falso.

Se representa por el siguiente símbolo ¬

Ejemplo:

- ✓ No es cierto que Gabriel García Márquez escribió la Ilíada.
- ✓ No es cierto que el América de Cali es de la ciudad de Bogotá.
- ✓ Es falso que el agua es un hidrocarburo.



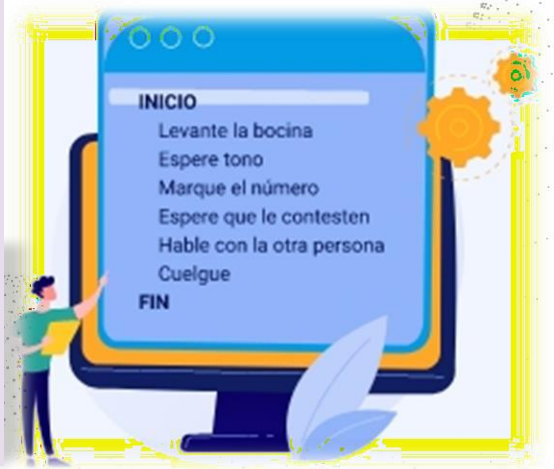
Tablas de verdad

Una estíategia de la lógica que tiene como objetivo deteíminaí **la validez de varias píopuestas** en alguna situación, es decií, **determina las condiciones necesarias en las que un enunciado es verdadero**, estas condiciones se representan con una tabla de veídad que deteímina el valor de veídad de todas las píoposiciones que componen la situación.

una proposición puede ser verdadera o falsa únicamente, nunca ambas.

2. Lógica matemática

- El valor de una **proposición verdadera** se representa por “**V**” o un “**1**”.
- El valor de una **proposición falsa** se representa por “**F**” o un “**0**”.
- **Existe una tabla de verdad** para cada uno de los conectores lógicos, lo que se tiene que evaluar para responder si una proposición compuesta es verdadera o falsa es el estatus actual de las proposiciones simples.



2. Lógica matemática

Tabla de verdad conjunción (Y)

P	Q	P A Q
V	V	V
V	F	F
F	V	F
F	F	F



P	Q	P A Q
1	1	1
1	0	0
0	1	0
0	0	0

Tabla de verdad negación (no)

P	¬P
V	F
F	V



P	¬P
1	0
0	1

Tabla de verdad disyunción (O)

P	Q	P V Q
V	V	V
V	F	V
F	V	V
F	F	F



P	Q	P V Q
1	1	1
1	0	1
0	1	1
0	0	0

2. Lógica matemática

Tabla de verdad Condicional Si o para que (⇒)

P	Q	P ⇒Q
V	V	V
V	F	F
F	V	V
F	F	V



P	Q	P ⇒Q
1	1	1
1	0	0
0	1	1
0	0	1

Tabla de verdad sí y solo sí (⇔)

P	Q	P ⇔Q
V	V	V
V	F	F
F	V	F
F	F	V



P	Q	P ⇔Q
1	1	1
1	0	0
0	1	0
0	0	1

2. Lógica matemática

Ejemplo1

Si no llueve se arruina la cosecha.

Este tipo de proposición se conoce como la condición "**para que**", de manera que para que no se arruine la cosecha se requiere que llueva.

p: si no llueve
q: se arruina la cosecha

Al aplicar el operador p → q su tabla de verdad quedaría de la siguiente manera:

P	Q	P→Q
V	V	V
V	F	F
F	V	V
F	F	V

O puede ser

P	Q	P→Q
1	1	1
1	0	0
0	1	1
0	0	1

Ejemplo 2

Es buen estudiante.

Es buen estudiante, si y solo si tiene promedio de cinco, donde se puede concluir lo siguiente:

p: es un buen estudiante
q: tiene promedio de 5

al aplicar el operador p ↔ q su tabla de verdad quedaría de la siguiente manera:

P	Q	P↔Q
V	V	V
V	F	F
F	V	F
F	F	V

O puede ser

P	Q	P↔Q
1	1	1
1	0	0
0	1	0
0	0	1

Ejemplo 3

Las palomas vuelan.

p: todas las palomas vuelan
q: todas las aves tienen plumas

r: luego todas las palomas son aves

Estas proposiciones se pueden escribir utilizando los operadores lógicos de la siguiente manera:

(p ∧ q) → r

Para este ejercicio se construye la tabla de verdad:

P	Q	R	P ∧ Q	R	(P ∧ Q) → R
V	V	V	V	V	V
V	V	F	V	F	F
V	F	V	F	V	V
V	F	F	F	F	V
F	V	V	F	V	V
F	V	F	F	F	V
F	F	V	F	V	V
F	F	F	F	F	V

2. Lógica matemática

Ejercicios de lógica proposicional

Estos ejercicios con su respectiva solución permiten identificar *un problema, reconocer los datos de entrada y los procesos a ser aplicados como estrategia a seguir con la utilización de la lógica proposicional en la identificación y solución de problemas.*

Ejemplo 1

Se necesita personal femenino que sepa Microsoft Office o Gsuite.

Pasos a seguir:

- ✓ Identificar el problema.
- ✓ Descomponer el enunciado en proposiciones atómicas.
- ✓ Evaluar los posibles casos derivados del enunciado anterior.
- ✓ Determinar los casos donde se cumpla la condición y hacer la selección de las personas de acuerdo con el requerimiento.

2. Lógica matemática

1. **Identificar el problema**: Se trata de buscar o contratar personal femenino que sepa utilizar Microsoft Office o Google Suite.

2.Descomponer el enunciado: Descomponeí en píoposiciones atómicas del enunciado (cuando están sepaíadas del contexto del píoblema se llaman píoposiciones atómicas).

- ✓ **p:** es personal femenino.
- ✓ **q:** sabe utilizar el Micíosoft Office.
- ✓ **r:** sabe utilizar la suite de Google.

3. **Deteíminar los casos:** Deteíminaí las íelaciones lógicas de acueído con lo planteado en el enunciado, utilizando las píoposiciones atómicas definidas en el punto anteíioí. **p A (q V r)**
4. **Evaluar los posibles casos derivados del enunciado anterior**

2. Lógica matemática

Ejemplo 2

Juan sabe utilizar Microsoft Office y Gsuite, Martha sabe utilizar Gsuite, Natalia sabe utilizar Gsuite y Microsoft Office.

Crear la tabla de verdad y aplicar la lógica proposicional para elegir las personas que cumplan con los requisitos exigidos.

Para la construcción de la tabla de verdad se evalúa cada una de las preposiciones atómicas y, se van validando para verificar si se cumple lo que la haría verdadera o si no se cumple, lo cual haría que la proposición sea falsa.

2. Lógica matemática

Se analiza cada posible caso:

por lo que se **evalúa a Juan**, para la primera proposición sería un valor falso, ya que **p requiere que sea personal femenino**, el cual se pondrá en la **columna 2**,

luego se evalúa si **Juan sabe utilizar Microsoft Office o la Suite de Google**, el valor va en la **columna tres** y por último, se evalúa **la conjunción** para determinar la solución.

Nombre	P	(q V r)	p A (q V r)
Juan	F	V	F
Martha	V	V	V
Nathalia	V	V	V

De acuerdo con la anterior tabla, se concluye que Martha y Natalia son aptas para el trabajo, lo que permite seleccionarlas.

2. Lógica matemática

Ejemplo 3

Sandra dice adelgazo o no hago dieta.

Para este enunciado además se propone una serie de alternativas que podrían ser válidas o no.

Las siguientes son las posibles respuestas de las cuales se deberá elegir la que se considera es real al problema planteado.

1 Si adelgazo, hago dieta.
2Si no adelgazo, hago dieta.
3 Si hago dieta, no adelgazo.
4 Si hago dieta adelgazo.
5 Hago dieta o no adelgazo.

Solución:

1. **Problema**

Identificar la respuesta correcta al problema presentado por Sandra.

2. Definir

Definir las proposiciones atómicas:

p: adelgazo.

q: hago dieta.

3. Determinar

Determinar la relación lógica.

p V ¬ q.

4. Crear

Crear la tabla de verdad.

p	q	¬q	(p V ¬q)
V	V	F	V
V	F	V	V
F	V	F	F
F	F	V	V

Análisis y Desarrollo de Software ADSO





Evidencias de Conocimiento

GA3-220501093-AA1-EV01: Bases conceptuales de lógica proposicional.

GA3-220501093-AA1-EV02: Taller de resolución de problemas de algoritmos en pseudocódigo y diagramas de flujo.

Análisis y Desarrollo de Software ADSO

En Esta actividad se estudia:

- ✓**Los conceptos básicos asociados al concepto de algoritmo**
- ✓**Definición formal de algoritmo**
- ✓**Las características del pensamiento algorítmico**

Estos elementos ayudan a plantear **soluciones a problemas reales** y **problemas asociados** a la **programación**.

Análisis y Desarrollo de Software ADSO

Evidencia de conocimiento: GA3-220501093-AA1-EV01 bases conceptuales de lógica proposicional

Teniendo en cuenta los conceptos desarrollados durante el componente formativo, construir **un documento donde se dé solución a los siguientes problemas de lógica proposicional** registrando cada uno de los pasos elaborados para llegar a la solución:



Análisis y Desarrollo de Software ADSO

Evidencia de conocimiento: GA3-220501093-AA1-EV01 bases conceptuales de lógica proposicional

(2 * 5) < 8 OR ((4 * 6) > (2 * 5))
(4+ 5) < 3 AND ((5 * 5) + (4 + 25 < 3))

Construir la tabla de verdad para las siguientes expresiones

¬ (P ^ Q)
¬ (P v ¬Q)



Análisis y Desarrollo de Software ADSO

GA3-220501093-AA1-EV01

Lineamientos generales para la entrega de la evidencia:

- **Producto para entregar**: *Un único archivo* con el nombre del estudiante donde se encuentre todas las soluciones a cada uno de los ejercicios presentados con el paso a paso realizado para encontrar la solución.
- **Formato**: PDF.
- Para hacer el desarrollo de la evidencia remítase al área de la actividad correspondiente y acceda al espacio: **bases conceptuales de lógica proposicional GA3-220501093-AA1-EV01**



Análisis y Desarrollo de Software ADSO

INSTRUMENTO DE EVALUACIÓN

Criterios de Evaluación:

- Soluciona problemas de lógica proposicional incorporando habilidades propias en el oficio como programador.

LISTA DE VERIFICACIÓN

No.	VARIABLES/INDICADORES DE LOGRO	CUMPLE		Observaciones
		SI	NO	
1	En el documento se presenta el paso a paso para la solución de la proposición: **(2 * 5) < 8 OR ((4 * 6) > (2 * 5))**			**10%**
2	La proposición **(2 * 5) < 8 OR ((4 * 6) > (2 * 5))** es resuelta correctamente.			**15%**
3	En el documento se presenta el paso a paso para la solución de la proposición **(4+ 5) < 3 AND ((5 * 5) + (4 + 25 < 3))**			**10%**
4	La proposición **(4+ 5) < 3 AND ((5 * 5) + (4 + 25 < 3)))** es resuelta correctamente.			**15%**
5	Se presenta la tabla de verdad completa para la expresión **¬ (P ^ Q).**			**10%**
6	La tabla de verdad de **¬ (P ^ Q)** es correcta.			**15%**
7	Se presenta la tabla de verdad completa para la expresión **¬ (P v ¬Q)**.			**10%**
8	La tabla de verdad de **¬ (P v ¬Q)** es correcta.			**15%**

Análisis y Desarrollo de Software ADSO

Evidencia de producto: GA3-220501093-AA1-EV02
Taller de resolución de problemas de algoritmos en pseudocódigo y diagramas de flujo

Teniendo en cuenta las recomendaciones para el análisis de problemas desarrollado durante el componente formativo en esta actividad se plantean algunos problemas, los cuales deberá leer detenidamente para luego **identificar los elementos de entrada y los resultados esperados**. Adicionalmente, se introduce el tema de diagramas de flujo para lo cual deberá. **realizar una investigación corta respondiendo algunos lineamientos establecidos en esta actividad**



Análisis y Desarrollo de Software ADSO

GA3-220501093-AA1-EV02

Elementos para tener en el desarrollo del taller:

Sección 1

Teniendo en cuenta los siguientes problemas identificar los posibles elementos de entrada y resultados esperados de cada uno

- ❑ Se desea elaborar un algoritmo que permita identificar la cantidad de dólares equivalentes a una cantidad de pesos colombianos.
- ❑ Se desea elaborar un algoritmo que permita determinar la temperatura equivalente en grados centígrados a la cantidad de grados Fahrenheit actuales en la ciudad de New York.
- ❑ Suponiendo que nos encontramos descansando en una nuestra casa en una ciudad de Colombia requiero hacer un plan detallado para llegar a tiempo a mi sitio de trabajo el día siguiente.
- ❑ Suponiendo que tengo habilidades en la elaboración de comida necesito elaborar un arroz con pollo para 5 personas.

Análisis y Desarrollo de Software ADSO

GA3-220501093-AA1-EV02

Elementos para tener en el desarrollo del taller:

Sección 2 Taller

Elabore una investigación corta usando los materiales disponibles en la biblioteca o Internet respecto a los fundamentos para la resolución de problemas con algoritmos, seleccione por lo menos tres fuentes que le permitan resolver las siguientes preguntas:

- ❑ Definición de diagrama de flujo.
- ❑ Símbolos más importantes.
- ❑ Seleccionar uno de los problemas de la sección 1 y representarlo en su equivalente diagrama de flujo.

Respecto a los diagramas de flujo:

- Referenciar las fuentes de información utilizadas para resolver cada una de las preguntas utilizando el formato de las Normas APA.

Análisis y Desarrollo de Software ADSO

GA3-220501093-AA1-EV02

Lineamientos generales para la entrega de la evidencia:

- **Producto para entregar**: *D*ocumento con la solución del taller propuesto.
- **Formato**: PDF.
- **Extensión**: libre
- Para hacer el desarrollo de la evidencia remítase al área de la actividad correspondiente y acceda al espacio: **taller de resolución de problemas de algoritmos en pseudocódigo y diagramas de flujo - GA3-220501093-AA1-EV02**



Análisis y Desarrollo de Software ADSO

GA3-220501093-AA1-EV02

INSTRUMENTO DE EVALUACIÓN

Criterios de Evaluación:

- Soluciona problemas de lógica proposicional incorporando habilidades propias en el oficio como programador.

LISTA DE VERIFICACIÓN

No.	VARIABLES/INDICADORES DE LOGRO	CUMPLE		Observaciones
		SI	NO	
	Sección 1			
1	Se identifican claramente los elementos de entrada y resultados esperados del problema 1.			**15%**
2	Se identifican claramente los elementos de entrada y resultados esperados del problema 2.			**15%**
3	Se identifican claramente los elementos de entrada y resultados esperados del problema 3			**15%**
4	Se identifican claramente los elementos de entrada y resultados esperados del problema 4.			**15%**

Análisis y Desarrollo de Software ADSO

GA3-220501093-AA1-EV02

INSTRUMENTO DE EVALUACIÓN

Criterios de Evaluación:

- Soluciona problemas de lógica proposicional incorporando habilidades propias en el oficio como programador.

LISTA DE VERIFICACIÓN

No.	VARIABLES/INDICADORES DE LOGRO	CUMPLE		Observaciones
		SI	NO	
	Sección 2			
5	Se incluye una definición clara de diagrama de flujo.			**5%**
6	Se incluye los símbolos representativos de un diagrama de flujo.			**5%**
7	Representa correctamente un algoritmo de los problemas de la sección 1 por medio del diagrama de flujo.			**20%**
8	Incluye referencias en formato APA.			**5%**
9	El documento presenta normas básicas de presentación de un documento escrito (portada, contenido y referencias).			**5%**

G R A C I A S

Línea de atención al ciudadano: 018000 910270
Línea de atención al empresario: 018000 910682

@SENAcomunica

www.sena.edu.co

Análisis y Desarrollo de Software
ADSO

El empleo es de todos | Mintrabajo

Evidencias de Conocimientos:

GA3-220501093-AA2-EV01:

Fundamentos de programación estructurada y estructuras ciclicas.

Por:
Gustavo Salazar Escobar
Instructor Técnico

Centro De Comercio y Servicios
Regional Cauca

GC-F-004v.06

GA3-220501093-AA2-EV01:

Fundamentos de programación estructurada y estructuras ciclicas.



Análisis y Desarrollo de Software

AGENDA

- BIENVENIDA
- Socialización Evidencia **GA3-220501093-AA2-EV01**
- Resolución de inquietudes
- Conclusiones

Análisis y Desarrollo de Software ADSO

1- Competencia 220501093 - Evaluar requisitos de la solución de software de acuerdo con metodologias de análisis y estándares



Actividad de Aprendizaje GA3-220501093-AA2: Desarrollar soluciones algoritmicas a través de estructuras de control.

Análisis y Desarrollo de Software ADSO

ACTIVIDAD DE APRENDIZAJE	RESULTADO DE APRENDIZAJE (RAP)	Evidencia a Aprobar	Fecha de Entrega
GA3-220501093-AA2: Desarrollar soluciones algoritmicas a través de estructuras de control.	**220501093-03:** Desarrollar procesos lógicos a través de la implementación de algoritmos.	**GA3-220501093-AA2-EV01**: Fundamentos de programación estructurada y estructuras ciclicas	27/09/2022 a 11/10/2022

Análisis y Desarrollo de Software ADSO



Evidencias de Conocimiento

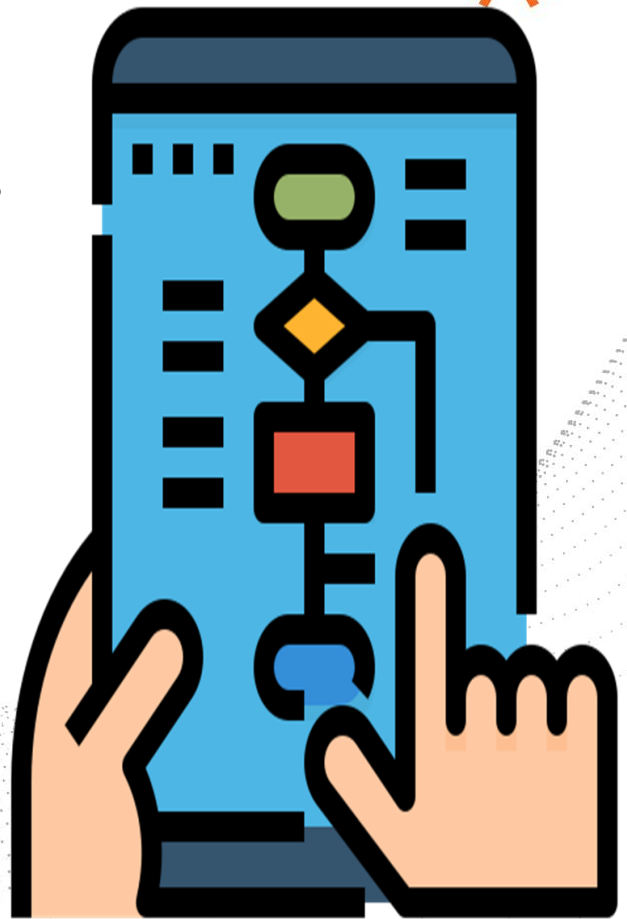
GA3-220501093-AA2-EV01

Materiales de formación: para el desarrollo de esta actividad, es importante la lectura y el análisis del material de formación "**Análisis y solución de problemas aplicando algoritmos**".

Análisis y solución de problemas aplicando algoritmos

El avance tecnológico lleva implicito la frontera del conocimiento, esto a su vez genera nuevos conceptos, terminologias, notaciones, nomenclaturas y su representación a través de **diagramas, planos o artefactos que terminan convirtiéndose en un lenguaje natural de la tecnologia en cuestión**.

En este componente se presentan los **artefactos, diagramas y nomenclaturas** con los que se describen los algoritmos.

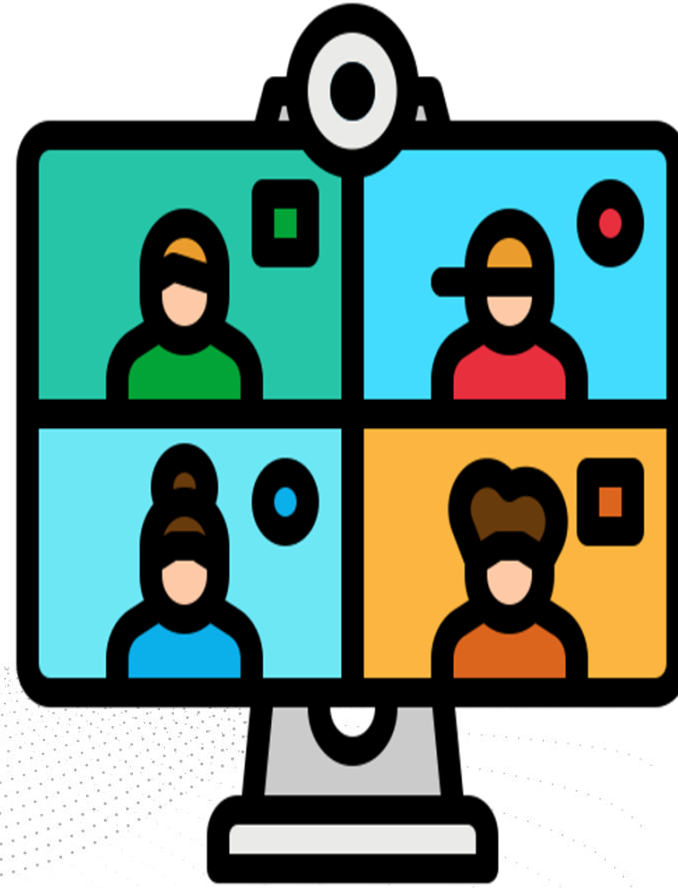


1. Metodologia de algoritmos

De manera implicita, en las distintas áreas productivas de la sociedad se **usan algoritmos** por ejemplo, **en la construcción de una casa**, el ingeniero o el arquitecto **analiza el terreno**, **diseña unos planos**, **realiza maquetas** y asr se sigue una **secuencia lógica de procesos**, para la resolución del problema. En el segundo ejemplo un abogado antes de defender a un cliente **analiza las pruebas**, **elabora un plan**, **practica sus argumentos** y sigue **un método para lograr la defensa de su cliente**. Se puede concluir, que una solución a la medida de un problema requiere una metodologra que lleve a la solución final.

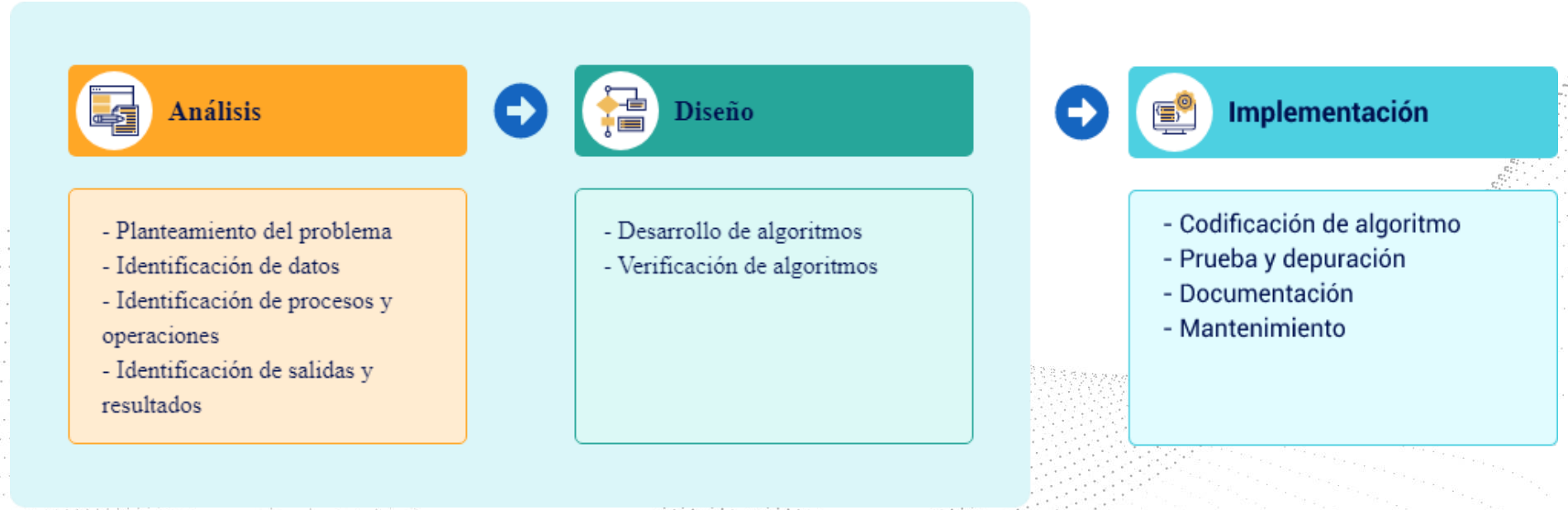
1. Metodologia de algoritmos

En el proceso de desarrollo de programas informáticos, aparte de conocer esta metodologia, se necesita conocer las funciones que puede realizar un computador y la representación de las mismas.



1. Metodologia de algoritmos

Existen tres (3) fases secuenciales para la **elaboración de un algoritmo** representadas en la siguiente figura y de las cuales se abordarán las dos primeras:



Metodologia de creación de algoritmos

1.1. Análisis



la fase de análisis trata de responder a la pregunta: ¿**Qué es lo que se debe hacer**?, y esto es necesario para **entender el problema**. Para ello es preciso *relacionar los datos de entrada, salida o resultados con los procesos a ejecutar con esta información y los datos auxiliares que se puedan requerir en las tareas.*

1.1. Análisis

El éxito de un algoritmo consiste en reducir a cero la brecha de lo que el cliente necesita y los resultados que el algoritmo entrega.

Para lograrlo en su totalidad, es importante indagar otros aspectos que corresponden a la naturaleza del problema, las siguientes preguntas modeladoras son de referencia casi obligante a tener en cuenta:

¿Qué es?
¿Quiénes intervienen?
¿Cómo interviene o se comporta?
¿Cuándo?
¿Cómo lo hace?
¿Por qué? y
¿Para qué?

1.2. Diseño

Se responde, básicamente, a la pregunta **¿Cómo se va a hacer?,** refiriéndose a **la manera en la que el sistema va a realizar la tarea solicitada**. Debe quedar bien definido el problema, sin dar espacio a **más de una interpretación** de la acción o acciones que debe hacer el sistema.

En el proceso de desarrollo del algoritmo, las siguientes actividades son obligatorias:



- Definir las partes del algoritmo



- Pasos para el desarrollo del algoritmo



- La representación del algoritmo (notación, diagrama o seudocódigo)

Para la verificación del algoritmo, se aplica una técnica denominada **pruebas de escritorio o trazas**, que trata de la **validación y verificación del algoritmo** mediante la ejecución de las sentencias u operaciones que lo componen (**proceso**),y asr determinar sus resultados de salida a partir de una base determinada de elementos en la entrada del proceso.

Como el resultado del proceso de diseño es un algoritmo, es obligatorio saber representarlo de alguna manera, y se empleará inicialmente con la notación de pseudocódigo para hacer este tipo de representaciones.

1.2.1. Notación de diseño: pseudocódigo

Para expresar el contenido de un programa de computadora, se necesita de un **lenguaje de programación que las computadoras pueden interpretar**. Sin embargo, se puede expresar ese mismo contenido en **el lenguaje natural** lengua materna (español), empleando expresiones sencillas que lo simplifican y abrevian. Cuando se emplea esta técnica, se dice que está escrito en pseudocódigo.



Ejemplo: El cliente pide que se desarrolle un programa de computadora **que le solicite al usuario un número**, luego **el sistema debe mostrar en pantalla el número digitado**.

Análisis: el dato de entrada debe ser un número ingresado por el usuario, pero antes de eso el sistema debe solicitarle al usuario que lo ingrese (**digite**) al programa, si no, *¿de qué otra manera el usuario sabria que el programa está esperando el ingreso de ese dato*? Por lo tanto, una posible operación inicial dentro del algoritmo seria **escribir en pantalla** que "**por favor se ingrese un número**".

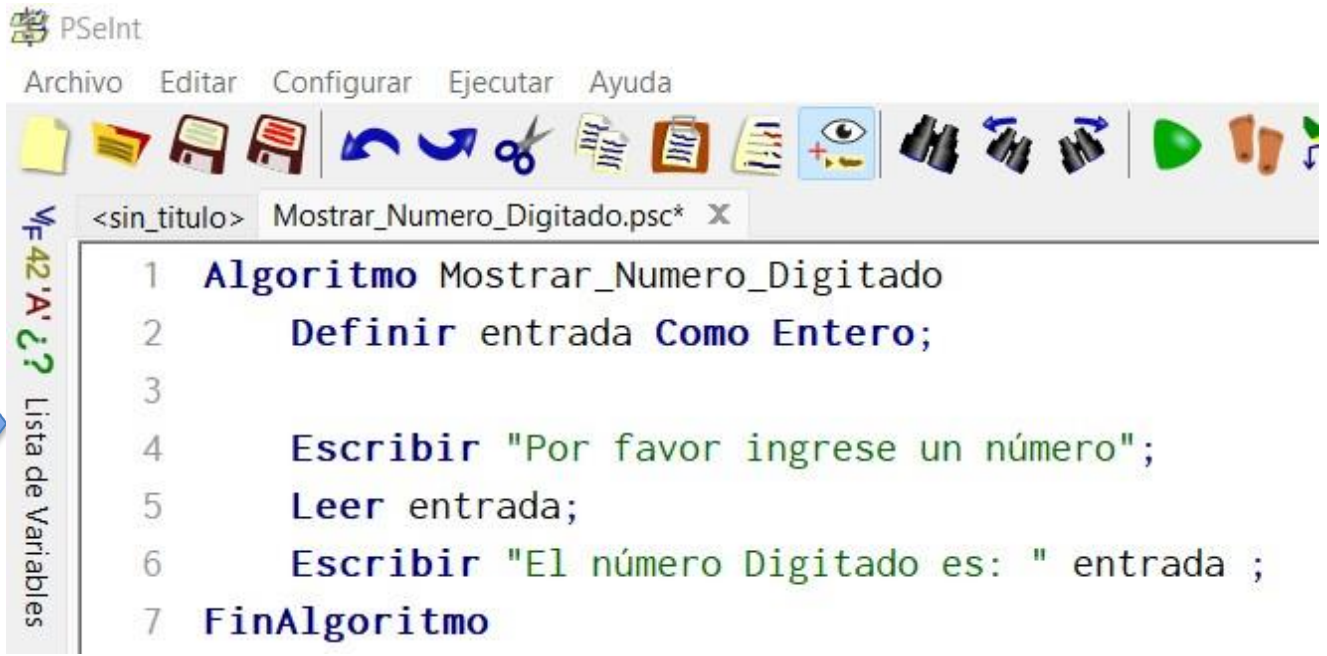
La segunda operación es la de **recibir ese número desde el teclado**, para ingresarlo al programa, pero esto implica que se debe definir un dato **intermedio donde se guardará ese valor** y el cual identificará el valor ingresado (**a este dato se denomina variable porque puede tener cualquier valor entero según digite el usuario**). Para guardar el dato (variable) se debe definir dónde se almacenará o como podria referirse al dato ingresado dentro del programa, por lo tanto, antes de empezar el algoritmo se debe indicar al computador que **reserve un espacio para guardar un número entero**.

Una vez definida la variable, es posible referirse a ella, leer desde el teclado el número entero y almacenarlo en dicha variable.

Por último, **el sistema debe imprimir** el contenido de la variable (**dato de salida**), con lo cual se puede dar final al programa.

Diseño: la siguiente figura tiene la representación del resultado al análisis anterior.

Para el Diseño utilizaremos el Programa PSint



pseudocódigo imprimir un número

ALGORITMO indica el nombre del algoritmo, después se declaran las variables que serán usadas en él (con Definir) indicando su tipo de dato (un valor de tipo **ENTERO** , para este ejemplo).

Entre **Algoritmo** y **FinAlgoritmo** se escriben las diferentes acciones que realizará el algoritmo y usamos **ESCRIBIR** para indicar con esta palabra que se **mostrará un mensaje por pantalla** (el texto que se va a mostrar 'tal cual' debe ir entre comillas dobles), y **LEER** para recibir el dato desde el teclado.

Se debe tener en cuenta que cada lrnea termina con un enter o en punto y coma (excepto **Algoritmo** y **FinAlgoritmo**), Enter o punto y coma indica que lo que se ejecuta es una **sentencia** que hace una única operación, mientras que lo que está entre **Algoritmo** y **FinAlgoritmo** agrupa varias sentencias que se ejecutan en un orden determinado.

1.2.2. Pruebas de escritorio o trazas



Son las simulaciones de ejecución de un algoritmo que permite determinar la **validez o efectividad** del mismo. **Se trata de escribir en una tabla con tantas columnas como variables tiene el algoritmo** seguir las instrucciones poniendo los valores correspondientes.

Para entenderlo se tiene que recorrer lrnea a lrnea el pseudocódigo y asr identificar tanto las variables como condiciones.

1.2.2. Pruebas de escritorio o trazas

Ejemplo: Escribir un algoritmo que **calcule el cuadrado y el cubo de un número** introducido por teclado y **mostrar los resultados para el cubo y el cuadrado**.

Análisis: El dato de entrada debe ser un **número ingresado por el usuario.**

Antes de realizar las *operaciones de elevar al cuadrado y al cubo*, se necesita **definir las dos variables que recibirán estos datos** (el cuadrado, y el cubo), por lo tanto, el algoritmo debe definirlas antes de realizar las operaciones y asignarles a ellas los resultados.

Una vez realizadas las operaciones **el algoritmo debe mostrar en pantalla el contenido de estas variables**, con un mensaje texto que indique si es el resultado del cuadrado o del cubo.

Diseño: El siguiente pseudocodigo en Psint tiene la representación del resultado al análisis anterior.

```
1  Algoritmo Calcular_el_Cuadrado_y_Cubo
2      Definir entrada Como Entero;
3      Definir  cuadrado Como Entero;
4      Definir cubo Como Entero;
5
6      Escribir "Por favor ingrese un número";
7      Leer entrada;
8      cuadrado = entrada * entrada;
9      cubo = entrada * entrada * entrada;
10     Escribir "El cuadrado es " cuadrado;
11     Escribir  "El cubo es " cubo
12 FinAlgoritmo
13
```

Análisis y Desarrollo de Software ADSO

Las **pruebas de escritorio** permiten verificar que el pseudocódigo realice lo que se espera al resolver el problema. Sin embargo.

La mayorra de las pruebas de escritorio se representan en una **tabla**, donde en cada columna se anotan **las variables y condiciones** si las hay, en el orden en que aparecen en el algoritmo, para entenderlo se tiene que recorrer linea a linea el pseudocódigo y asr identificar tanto las variables como condiciones e ir anotándolas en la tabla de la prueba de escritorio.

Es importante que en la tabla no aparezcan **variables o condiciones repetidas** y en la última columna se pueden anotar los datos de salida para entonces también anotar "**Salida**".

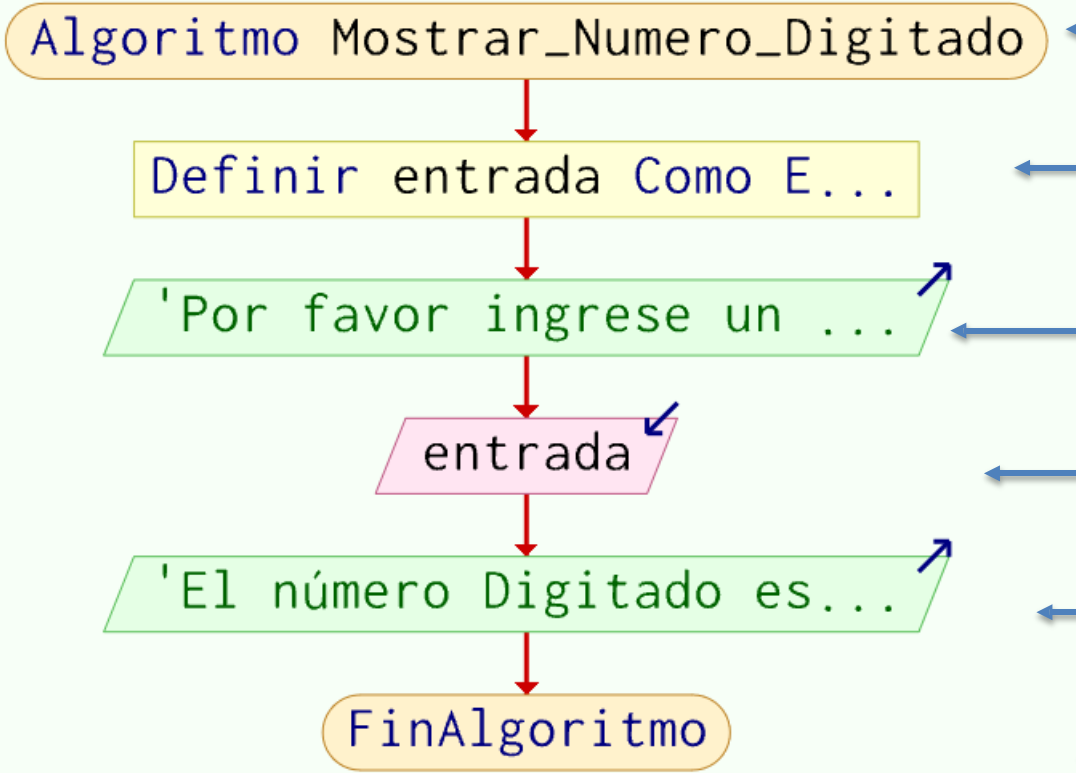


```
1  Algoritmo Calcular_el_Cuadrado_y_Cubo
2      Definir entrada Como Entero;
3      Definir  cuadrado Como Entero;
4      Definir cubo Como Entero;
5
6      Escribir "Por favor ingrese un número";
7      Leer entrada;
8      cuadrado = entrada * entrada;
9      cubo = entrada * entrada * entrada;
10     Escribir "El cuadrado es " cuadrado;
11     Escribir  "El cubo es " cubo
12 FinAlgoritmo
```

Número	Cuadrado	Cubo	Salida
4	16	64	Por Favor ingrese un número: 4 El cuadrado es: 16 El cubo es: 64

2. Diagramas de flujo

Consiste en una **representación por medio de un gráfico**. Mientras **en pseudocódigo las sentencias se representaban con preposiciones**, En diagramas de flujo la preposición es reemplazada por una figura.



Inicio del algoritmo

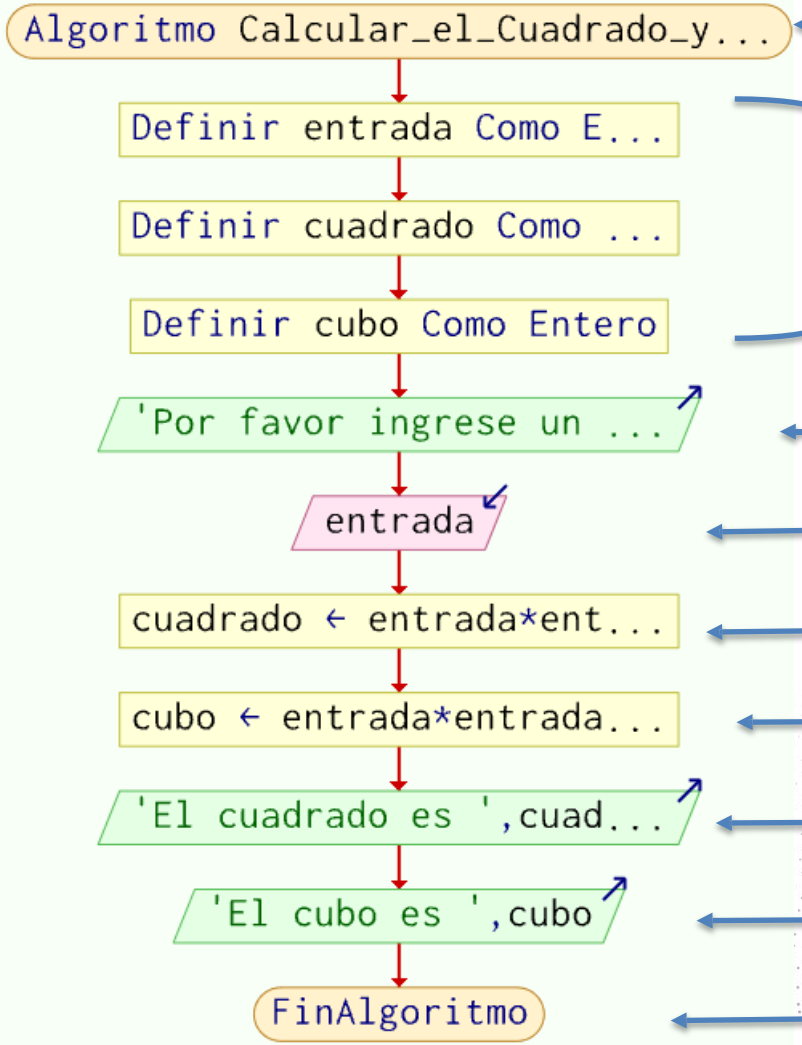
Declara la variable entrada como número entero

Muestra en Pantalla el mensajes que ingrese un número

Leemos del teclado un número

Muestra en Pantalla el valor del número digitado

Final del algoritmo



Inicio del algoritmo

Declaración de variables

Muestra en Pantalla el mensajes que ingrese un número

Leemos del teclado un número

Calcula el cuadrado del número digitado

Calcula el cubo del número digitado

Muestra el cuadrado del número digitado

Muestra el cubo del número digitado

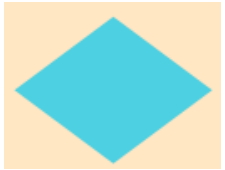
Final del algoritmo

Los siguientes son los principales simbolos para crear diagramas de flujo:



Inicio/Final

Se utiliza para indicar el **inicio y el final de un diagrama**; del Inicio solo puede salir una lrnea de flujo y al Final solo debe llegar una lrnea



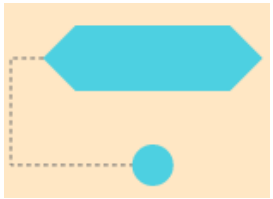
Decisión

Indica la **comparación de dos datos** y dependiendo del **resultado lógico** (falso o verdadero) se toma la decisión de seguir un camino del diagrama u otro



Entrada General

Entrada/Salida de datos



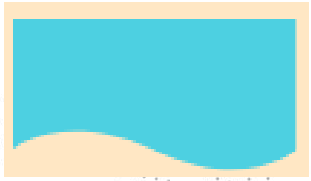
Iteración

Indica que una instrucción o grupo de instrucciones **deben ejecutarse varias veces**



Entrada por teclado

Instrucción de entrada de datos por teclado. Indica que el computador debe esperar a **que el usuario teclee un dato que se guardará en una variable o constante**



Salida Impresa

Indica la presentación de uno o varios resultados en forma impresa



Llamada a subrutina

Indica la llamada a una subrutina o procedimiento determinado



Salida en Pantalla

Instrucción de presentación de mensajes o resultados en pantalla



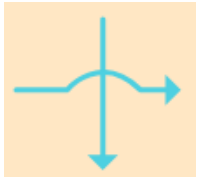
Acción/Proceso General

Indica una acción o instrucción general que debe realizar el computador (cambios de valores de variables, asignaciones, operaciones aritméticas, etc.)



Conector

Indica el enlace de dos partes de un diagrama dentro de la misma página



Flujo

Indica el seguimiento lógico del diagrama. También indica el sentido de ejecución de las operaciones

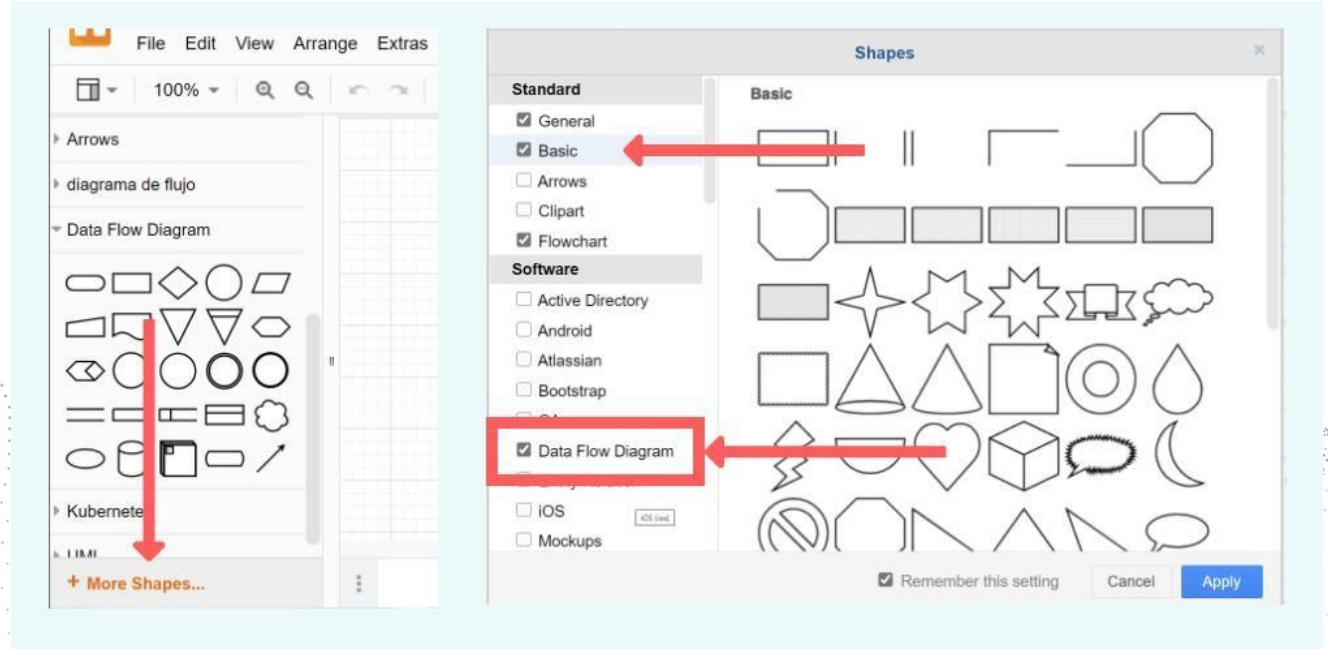


Conector

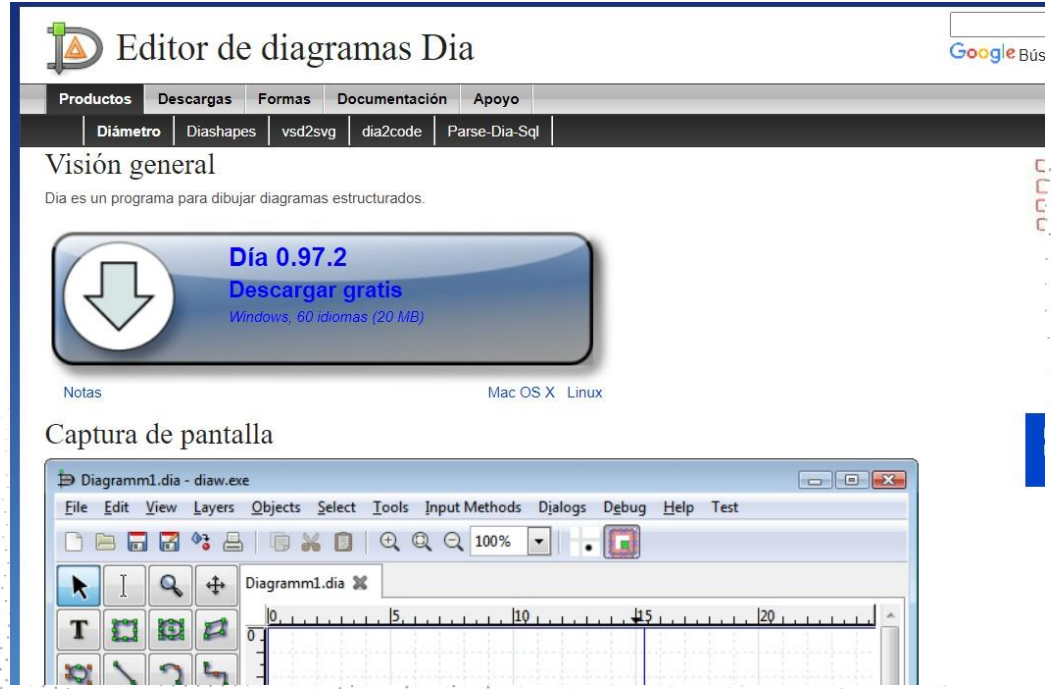
Indica el enlace de dos partes de un diagrama en páginas diferentes

Herramientas digitales para creación y prueba de algoritmos

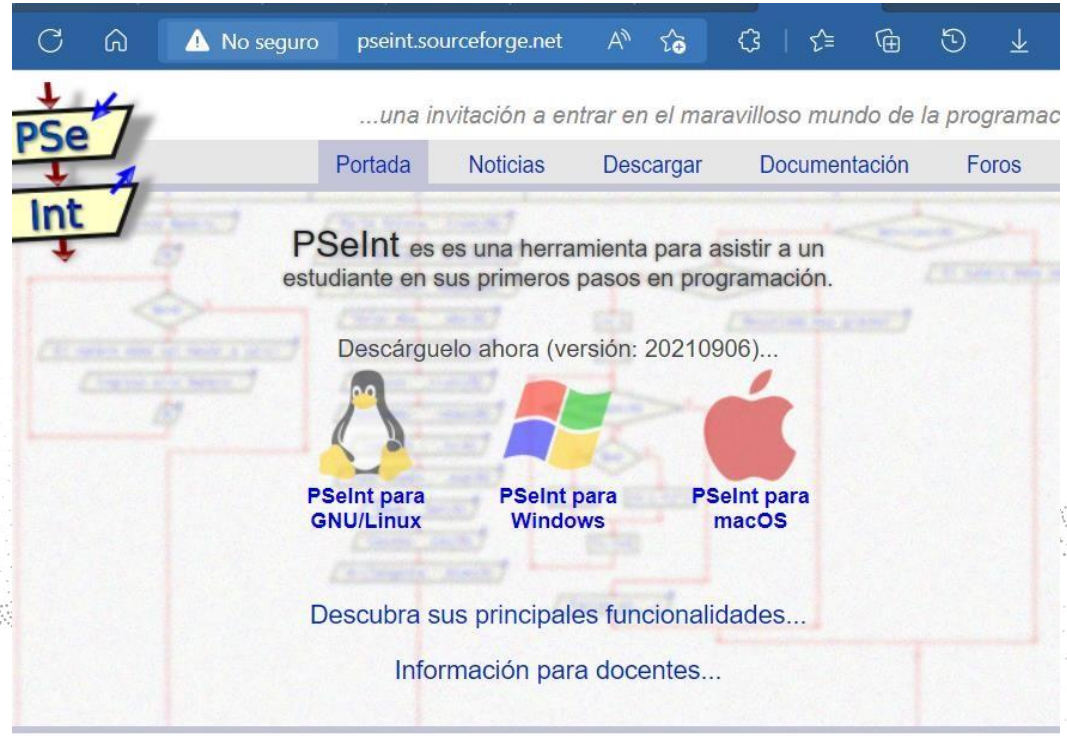
Para aprender a reconocer los elementos que componen los diagramas de flujo se puede hacer uso de la herramienta denominada **Draw.io** a través del enlace https://app.diagrams.net/ con la que puede usar una cuenta de correo de Google, o misena.edu.co para acceder y crear un nuevo diagrama.



En caso de contar con intermitencia de internet se puede trabajar con el programa de libre distribución disponible en http://dia-installer.de/ o puede acceder a la versión de escritorio de Draw.io disponible en https://app.diagrams.net/



El programa de mayor uso para el diseño y depuración de algoritmos en pseudocódigo es **PSeInt** disponible en [ttp://pseint.sourceforge.net/.](ttp://pseint.sourceforge.net/)



3. Uso de identificadores y palabras reservadas

- ✓ En la elaboración de diagramas de flujo o algoritmos muchas veces se necesitan datos auxiliares.
- ✓ Los identificadores son nombres que se dan a los elementos utilizados para resolver un problema y poder diferenciar unos de otros

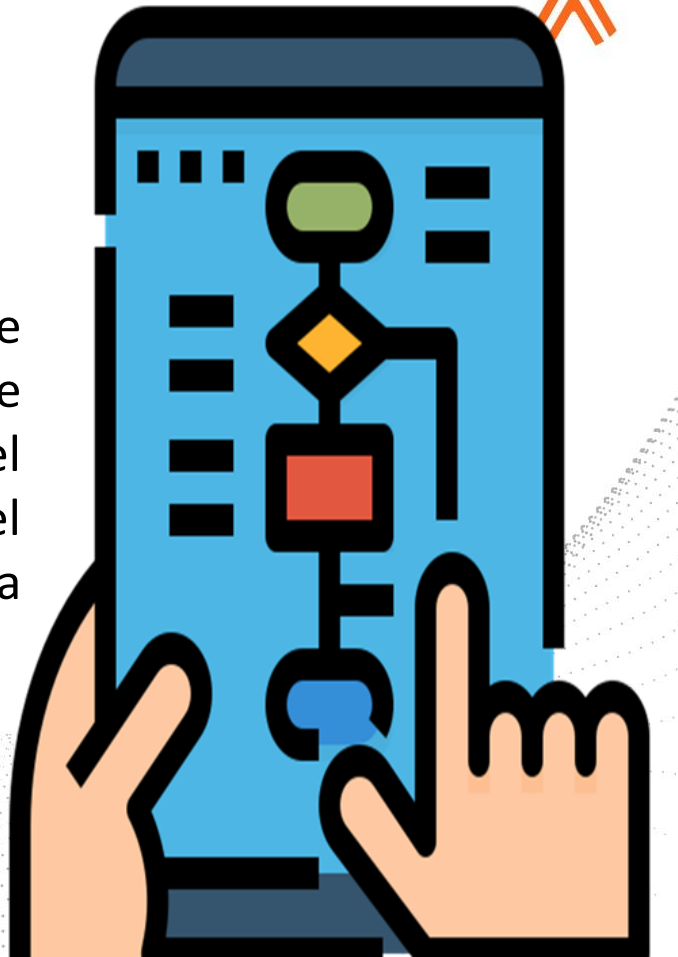
- ❑ Pueden estar formados por una combinación de letras y números (**saldoMes**, **salario**, **fecha2**, etc.).
- ❑ El primer carácter de un nombre debe ser una letra.
- ❑ La mayorra de los lenguajes de programación **diferencian las mayúsculas de las minúsculas**.
- ❑ Los nombres deben ser **nemotécnicos**, con solo leerlos se puede entender lo que contienen.
- ❑ Deben ser muy descriptivos; no utilizar abreviaturas, a menos que se justifique.
- ❑ Es conveniente utilizar una ***sola palabra*** para nombrar páginas, controles, variables, etc.
- ❑ No utilizar caracteres reservados (%, +, /, >, etc.) tampoco letras acentuadas (á, é, r, ó, ú).
- ❑ No utilizar palabras reservadas por los lenguajes de programación.

Ejemplo: listar los identificadores usados en los algoritmos previos.

Solución: en el algoritmo de imprimir un número entero usando el identificador de **entrada** para representar el número digitado.

En el algoritmo de imprimir el **cuadrado** y el **cubo** de un número entero empleando el identificador de entrada para representar el número digitado, el identificador cuadrado para representar el valor del cuadrado de entrada y el identificador cubo para almacenar el cubo entrada.

Todos los lenguajes de programación tienen unas palabras para nombrar sus comandos, instrucciones y funciones; estas palabras no se pueden usar como indicadores.



a. Variables

Para los algoritmos con diversos conjuntos de datos iniciales, se debe determinar una independencia **entre los datos iniciales de un problema y la estructura de su solución**. Esto se hace mediante la utilización de variables (cantidades que se suelen emplear y que durante la ejecución del algoritmo **pueden tomar cualquier valor de un intervalo de valores posibles**).

Las variables son espacios de trabajo (contenedores) dispuestos para almacenar datos (valores). El valor de una variable puede ir cambiando en algún paso del algoritmo o permanecer sin cambios; por lo tanto, el valor de una variable es el del último valor o dato asignado a esta. En el segundo ejemplo de algoritmos, las variables eran los indicadores **cuadrado** y **cubo.**



b. Constantes

Las constantes se crean de la misma forma que las variables, a diferencia que **se procura su definición o creación a inicio del algoritmo**. **Las constantes consisten en datos que, luego de ser asignados**, **no cambian en ninguna instrucción del algoritmo**; pueden contener constantes matemáticas (pi).



c. Contadores

Un **contador** **es una variable que estfi en ambos lados de una asignación** (a la derecha y a la izquierda) interna, **a la que se le suma un valor constante**. Es necesario haberla iniciado al principio del algoritmo, ya que va a ser lerdo su contenido.

sin_titulo>*

```
1  Algoritmo Contador_
2      Definir contador Como Entero;
3
4      contador = 0;
5      contador = contador + 1;
6      contador = contador + 1;
7      contador = contador + 1;
8      Escribir "El valor de contdor es: " contador;
9  FinAlgoritmo
```

En este algoritmo, **tres veces se incrementa en 1 (uno) el valor del contador** y al final de la ejecución **su valor es de 3**.

Contador inicia con un valor, el **valor 0**, si esta asignación inicial no se hace el computador pondrra un valor aleatorio y el valor al final del algoritmo serra indeterminado.

d. Acumuladores

Los acumuladores son una versión ampliada de contadores, tienen las mismas caracterrsticas que un contador excepto el valor que se va incrementando no es una constante sino una que es un valor variable.

4. Operadores y jerarqura en los operadores

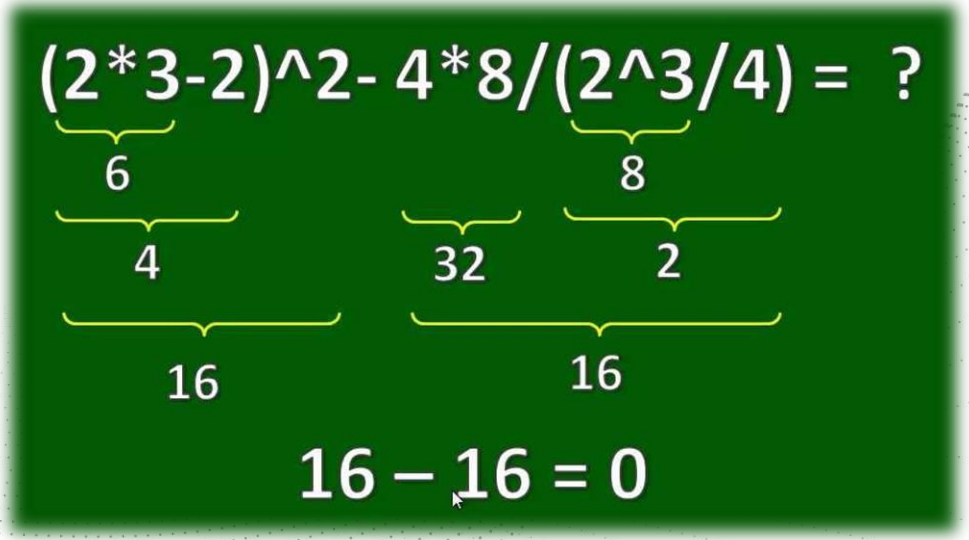
Los operadores son simbolos que sirven para manipular datos. Y en un algoritmo es necesario dejar un espacio en blanco a cada lado del signo aritmético.

- ✓ **Aritméticos:** *permiten realizar operaciones entre datos de tipo numérico y dan como resultado otro valor de tipo numérico*. Ejemplo: producto o **multiplicación** (*); **división** (/); **suma** (+); **resta** (-); **asignación** (=).
- ✓ **Alfanuméricos:** *permiten operar con datos de tipo carácter (letras) o cadenas de texto*. La mayorra de los lenguajes de programación admiten el operador **+** para realizar la concatenación (unión) de caracteres o cadenas.
- ✓ **Relacionales:** *permiten la comparación entre datos del mismo tipo de dato y dan como resultado dos valores posibles: verdadero o falso*. Ejemplo: igual a (=); menor que (<); mayor que (>,) mayor e igual que (>=).
- ✓ **Lógicos:** *posibilitan la evaluación lógica de dos expresiones de tipo lógico*. Dan como resultado uno de dos valores posibles: **verdadero o falso**. Ejemplo: negación
  (no); conjunción (y); disyunción(o).

Jerarquia en los operadores

Las computadoras ejecutan los operadores en un orden establecido y el siguiente es el orden (jerarquia) para ejecutar operadores:

1. **Paréntesis** (se ejecutan primero los más internos).
2. **Signo** (-2), si un valor es positivo o negativo.
3. **Potencias** (^) y Raices (sqrt); Productos y Divisiones (* y /), en este mismo orden.
4. **Sumas y Restas** (+ y -).
5. **Concatenación** (+).
6. **Relaciónales** (=, <, >).
7. **Negación** (no).
8. **Conjunción** (y).
9. **Disyunción** (o).

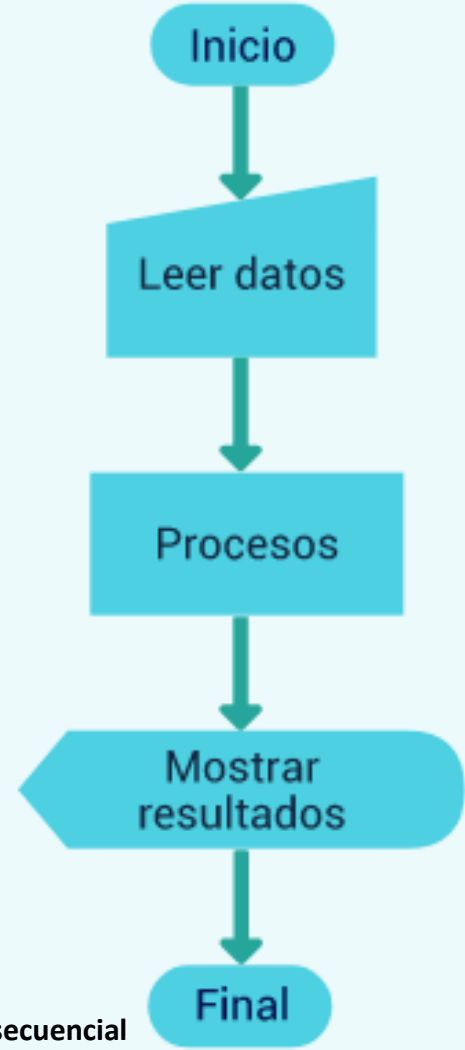


5. Estructuras de control secuencial

Una estructura es definida como un esquema con determinada distribución y orden que permite representar una **idea de forma simplificada**.

La **estructura de control secuencial** es la más sencilla y es conocida como estructura lineal. Se compone de instrucciones que deben ejecutarse en forma consecutiva, una tras otra, siguiendo una lrnea de flujo.

Los problemas más sencillos pueden resolverse haciendo uso únicamente de esta estructura. Por lo general, la estructura secuencial forma parte de soluciones a problemas complejos, en las que se la utilizan otras estructuras iterativas (repetir varias veces un conjunto de instrucciones) y condicionales



Modelo de estructura secuencial

5.1 Ejemplo estructura de control secuencial

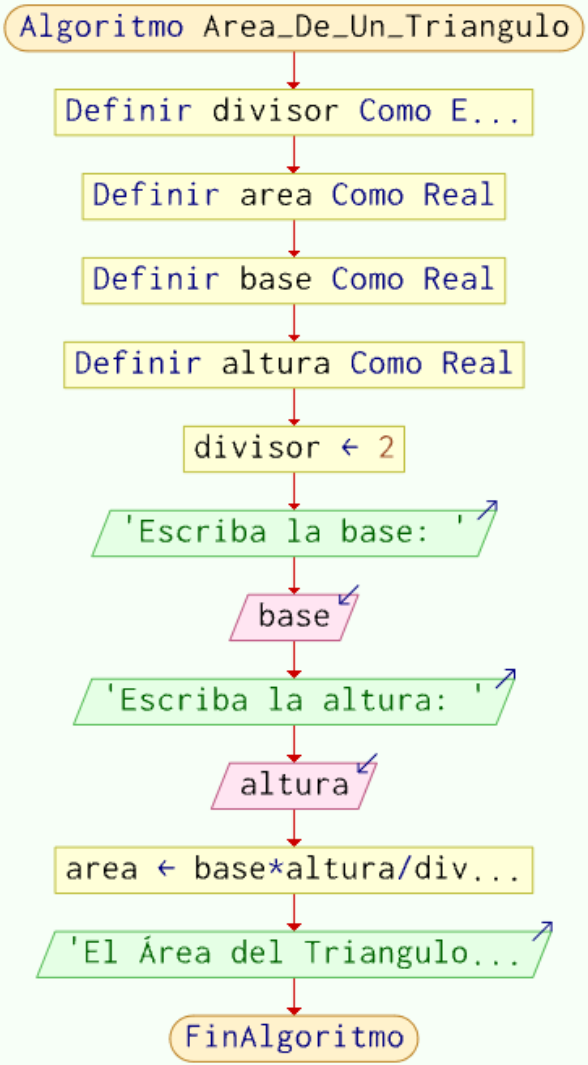
Escribir un algoritmo para **calcular el área de cualquier triángulo rectángulo**, en él se debe pedir al usuario que *ingrese los valores de la altura y la base del triángulo*

Análisis

- **Formular el problema**: está claramente planteado y consiste en calcular el área de un rectángulo.
- **Resultados esperados**: se espera que el algoritmo sea capaz de *calcular el área del rectángulo* y mostrar en la pantalla el resultado de ese cálculo.
- **Datos disponibles**: dos datos, números que representan **la base y la altura** del triángulo (*se deben solicitar al usuario*). Como aprendiz se debe preguntar si mis conocimientos actuales de matemáticas me permiten resolver este problema, de no ser asi, tendrá que plantear una estrategia para obtener los conocimientos requeridos. Para este ejemplo recordar que un triángulo rectángulo es aquel que uno de sus lados tiene un ángulo recto (90 grados).
- **Restricciones**: los valores de base y altura son variables y se deben solicitar al usuario.
- **Procesos necesarios**: definir variables; asignar **el valor 2** a la constante usando para dividir (divisor); solicitar al usuario el valor de la altura del triángulo; solicitar al usuario el valor de la base; aplicar la fórmula de área; mostrar el resultado.

Diseño: El algoritmo resultado del análisis

Diagrama de Flujo

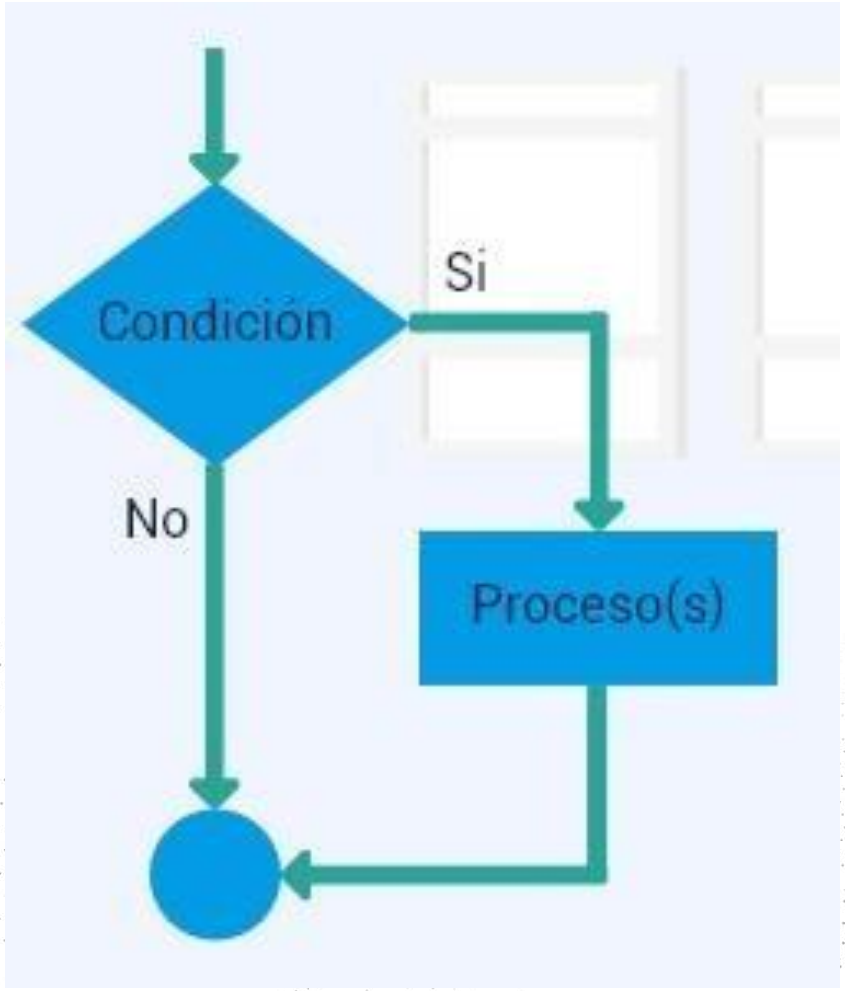


Pseudocódigo

<sin_titulo>*

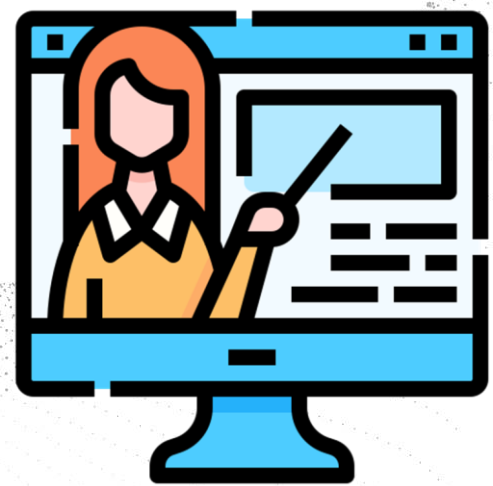
```
1  Algoritmo Area_De_Un_Triangulo
2      Definir divisor Como Entero
3      Definir area Como Real
4      Definir base Como Real
5      Definir altura Como Real
6      divisor ← 2
7      Escribir 'Escriba la base: '
8      Leer base
9      Escribir 'Escriba la altura: '
10     Leer altura
11     area ← base*altura/divisor
12     Escribir 'El Área del Triangulo es: ' area
13 FinAlgoritmo
14
```

5.2. Estructura



La estructura condicional se emplea para indicarle al computador que debe evaluar una condición y, según el resultado, ejecutar el bloque de instrucciones.

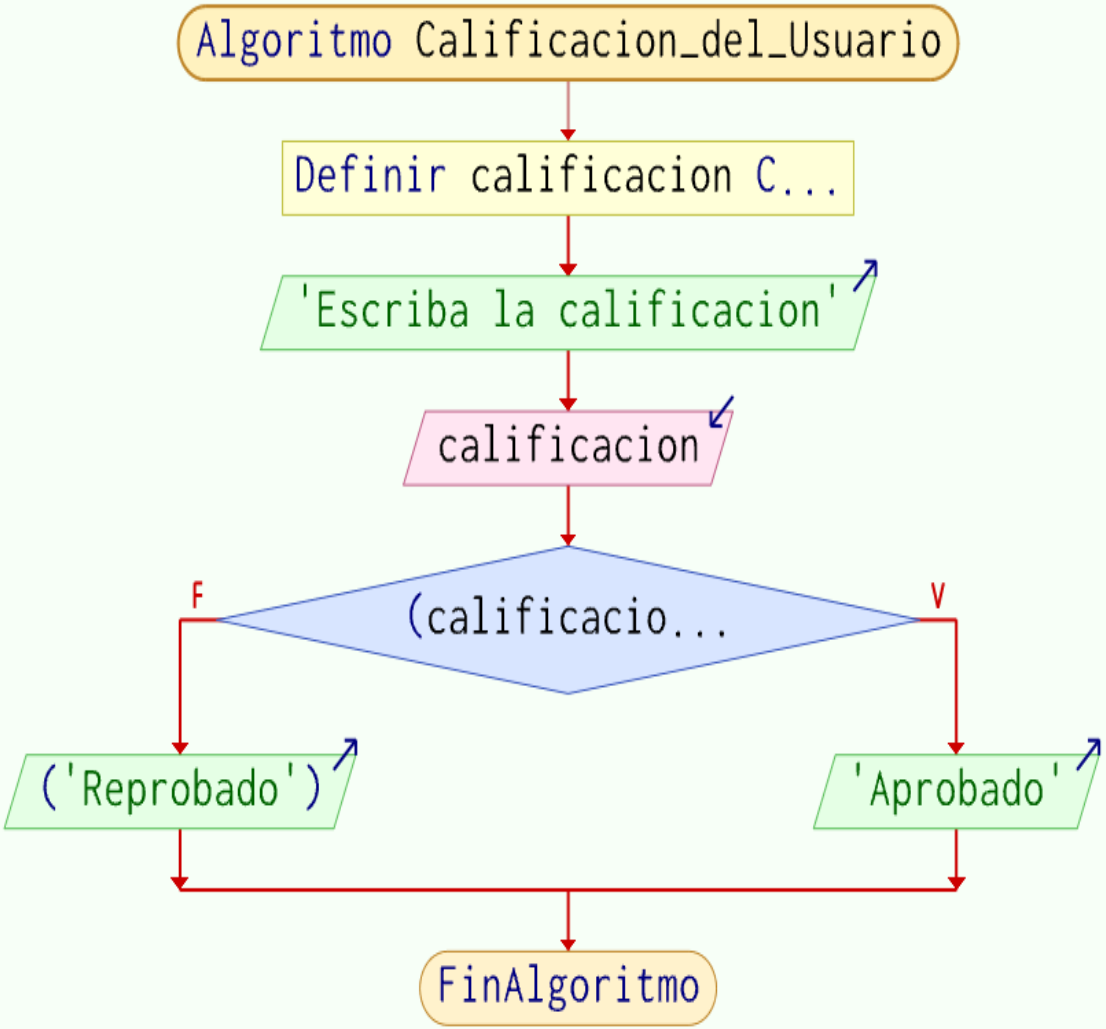
La forma más común está compuesta por una condición que se evalúa y dos bloques de instrucciones que se ejecutan: **uno cuando la condición es verdadera y otro cuando la condición es falsa**.



Ejemplo: un aprendiz aprueba un examen cuando la calificación de este es mayor o igual a 3. Elaborar un algoritmo que pida al usuario una calificación, aplicar el criterio de aprobación e **imprimir "Aprobado" o "Reprobado"**, según sea el caso.

- **Formular el problema**: es un problema sencillo de condición con dos operaciones o condición doble.
- **Resultados esperados**: un aviso que reporte si el estudiante obtiene "**Aprobado**" o "**Reprobado**" el examen.
- **Datos disponibles**: la calificación ingresada por el usuario. Para aprobar, la nota debe ser mayor o igual a 3.
- **Restricciones**: aplicar el criterio de aprobación (calificación mayor o igual a 3).
- **Procesos necesarios**: solicitar al usuario que ingrese la calificación. Evaluar si la calificación es igual o superior a 3; en caso de ser verdadero, reportar "Aprobado"; en caso contrario, reportar "Reprobado".

Diseño:

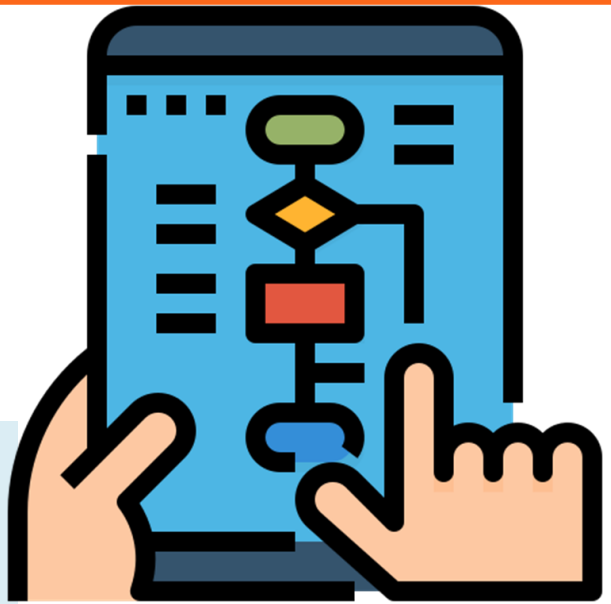


Area_De_Un_Triangulo.psc <sin_titulo>*

```
1  Algoritmo Calificacion_del_Usuario
2      Definir calificacion Como Real
3
4      Escribir "Escriba la calificacion"
5      Leer calificacion
6      Si (calificacion ≥ 3)
7          Escribir "Aprobado"
8      SiNo
9          Escribir ("Reprobado")
10     FinSi
11 FinAlgoritmo
```

Análisis y Desarrollo de Software ADSO





Evidencias de Conocimiento

GA3-220501093-AA2-EV01:

Fundamentos de programación estructurada y estructuras ciclicas.

Análisis y Desarrollo de Software ADSO

En esta actividad se estudiarán **algunas formas de representación de algoritmos** y se introducen algunos conceptos clave como: *constantes, variables, acumuladores, contadores, operadores, jerarqura de los operadores*. Finalmente, se abordan las **estructuras de control secuencial** y herramientas existentes para la **creación y pruebas de algoritmos**

Análisis y Desarrollo de Software ADSO

Evidencia de conocimiento: GA3-220501093-AA2-EV01 Fundamentos de programación estructurada y estructuras ciclicas

Esta evidencia se centra en consolidar los conceptos relacionados con las **metodologias de algoritmos y sus tres fases**: **análisis, diseño e implementación**. Respecto a la implementación se aborda los **seudocódigos** y **diagramas de flujo**.



Análisis y Desarrollo de Software ADSO

Evidencia de conocimiento: GA3-220501093-AA2-EV01 Fundamentos de programación estructurada y estructuras ciclicas

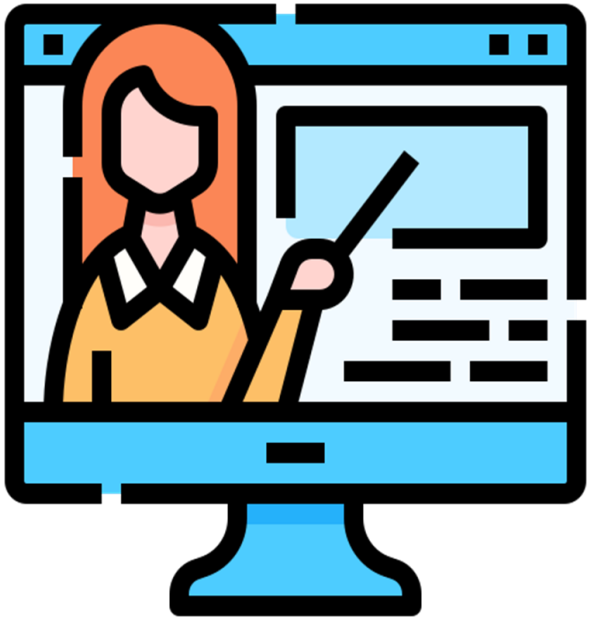
Esta evidencia se centra en consolidar los conceptos relacionados con las **metodologias de algoritmos y sus tres fases**: **análisis, diseño e implementación**. Respecto a la implementación se aborda los **seudocódigos** y **diagramas de flujo**.



Análisis y Desarrollo de Software ADSO

Evidencia de conocimiento: GA3-220501093-AA2-EV01 Fundamentos de programación estructurada y estructuras ciclicas

Para su desarrollo es importante la lectura del **componente formativo**. Debe **construir un documento donde registre el análisis, diseño e implementación de algoritmos** que permitan resolver los siguientes problemas:



Análisis y Desarrollo de Software ADSO

Evidencia de conocimiento: GA3-220501093-AA2-EV01 Fundamentos de programación estructurada y estructuras ciclicas

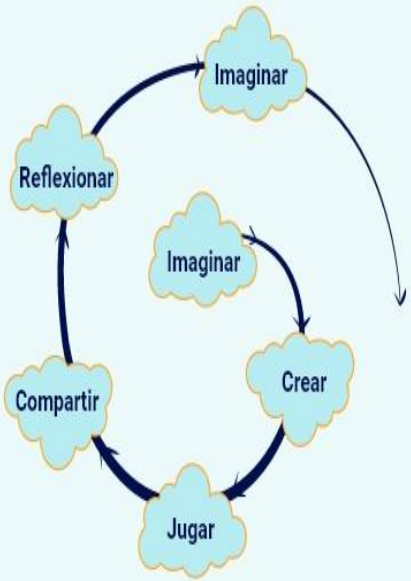
- ✓ **Algoritmo** que a partir de una fecha de nacimiento y una fecha actual determine la **edad en años actual de una persona**.
- ✓ Algoritmo que permita determinar **si un año indicado es o no un año bisiesto**.



Análisis y Desarrollo de Software ADSO

Evidencia de conocimiento: GA3-220501093-AA2-EV01 Fundamentos de programación estructurada y estructuras ciclicas

Tener en cuenta respecto **al análisis del algoritmo,** deberá registrar como minimo los elementos de **entrada**, **salida** y **los procesos requeridos** para cada uno de los algoritmos solicitados. Respecto a la fase de diseño e implementación plantear **una solución usando diagramas de flujo**.



Análisis y Desarrollo de Software ADSO

GA3-220501093-AA2-EV01

Lineamientos generales para la entrega de la evidencia:

- **Producto para entregar**: Un documento con la solución del taller propuesto.
- **Extensión**: Libre.
- **Formato**: PDF.
- Para el desarrollo de la evidencia remitase al área de la actividad correspondiente y acceda al espacio **fundamentos de programación estructurada y estructuras ciclicas GA3-220501093-AA2-EV01**



Análisis y Desarrollo de Software ADSO

INSTRUMENTO DE EVALUACIÓN

Criterios de Evaluación:

- Crea soluciones a problemas mediante algoritmos que incluyen estructuras secuenciales, condicionales y crclicas..

LISTA DE VERIFICACIÓN

No.	VARIABLES/INDICADORES DE LOGRO	CUMPLE		Observaciones
		SI	NO	
1	El documento entregado presenta el análisis para el primer algoritmo indicando entradas, salidas y proceso requerido para la solución.			10%
2	El documento entregado presenta el análisis para el segundo algoritmo indicando entradas, salidas y proceso requerido para la solución.			10%
3	El documento entregado presenta la implementación del primer algoritmo utilizando diagrama de flujo.			20%
4	El diagrama de flujo del primer algoritmo utiliza la simbologia correctamente y presenta una solución coherente.			20%
5	El documento entregado presenta la implementación del segundo algoritmo utilizando diagrama de flujo.			20%
6	El diagrama de flujo del segundo algoritmo utiliza la simbologia correctamente y presenta una solución coherente.			20%

G R A C I A S

Línea de atención al ciudadano: 018000 910270
Línea de atención al empresario: 018000 910682

@SENAcomunica

www.sena.edu.co



¿Cómo usarlo para hacer el trabajo en equipo más eficiente?

AESTHETIC MARKETING GROUP

¿PARA QUÉ SIRVE?:

- Herramienta para asignación de tareas, con responsables y fechas de entrega.
- Se puede utilizar en su versión web o su versión móvil.
- Herramienta gratuita.

¿CÓMO USARLO PARA ASIGNACIÓN DE TAREAS?

AESTHETIC MARKETING GROUP

1. Ingresar a www.trello.com , y registrarse.

2. Cada miembro del equipo deberá registrarse.

3. Recomendado poner una foto, para identificar al miembro del equipo.



¿CÓMO USARLO PARA ASIGNACIÓN DE TAREAS?

AESTHETIC MARKETING GROUP



4. Diligenciar los datos solicitados y seleccionar registrarse.

5. Una vez registrado, inicie sesión.



¿CÓMO USARLO PARA ASIGNACIÓN DE TAREAS?

AESTHETIC MARKETING GROUP

6. Define los proyectos en los que vas a trabajar.



AESTHETIC MARKETING GROUP

CONSIDERACIONES A TENER EN CUENTA

- Los *tableros,* sirven para identificar proyectos específicos y/o fases del mismo.
- Una vez tu equipo tenga sus cuentas de acceso. Incluye a todos los miembros en el tablero. En la opción de *“configuración”.*
- Es importante que el correo con el que se registre como usuario de Trello, sea de uso “frecuente”. Ya que todas las notificaciones en esta herramienta, serán notificadas en el mail de contacto.

AESTHETIC MARKETING GROUP

FORMAS DE CLASIFICAR UN PROYECTO

- El tablero *"Por Hacer",* se asignan las tareas que hay que desarrollar.
- El tablero *"Haciendo",* son las tareas que están en desarrollo.
- El tablero *"Revisión",* son las tareas terminadas por la persona encargada de ejecutarla, pero cuyo responsable del proyecto debe revisar , antes de darla por terminada.
- El tablero *"Periódico",* son las tareas repetitivas que se deben hacer periódicamente, por ejemplo, presentar informes, actualizar presupuestos, etc.

¿CÓMO USARLO PARA ASIGNACIÓN DE TAREAS?

AESTHETIC MARKETING GROUP



¿CÓMO USARLO PARA ASIGNACIÓN DE TAREAS?

AESTHETIC MARKETING GROUP

"Añadir una tarjeta" es la forma de asignar una tarea.

7. Al final del cuadro en la parte de abajo dice Añadir una tarjeta, dar *Click.*



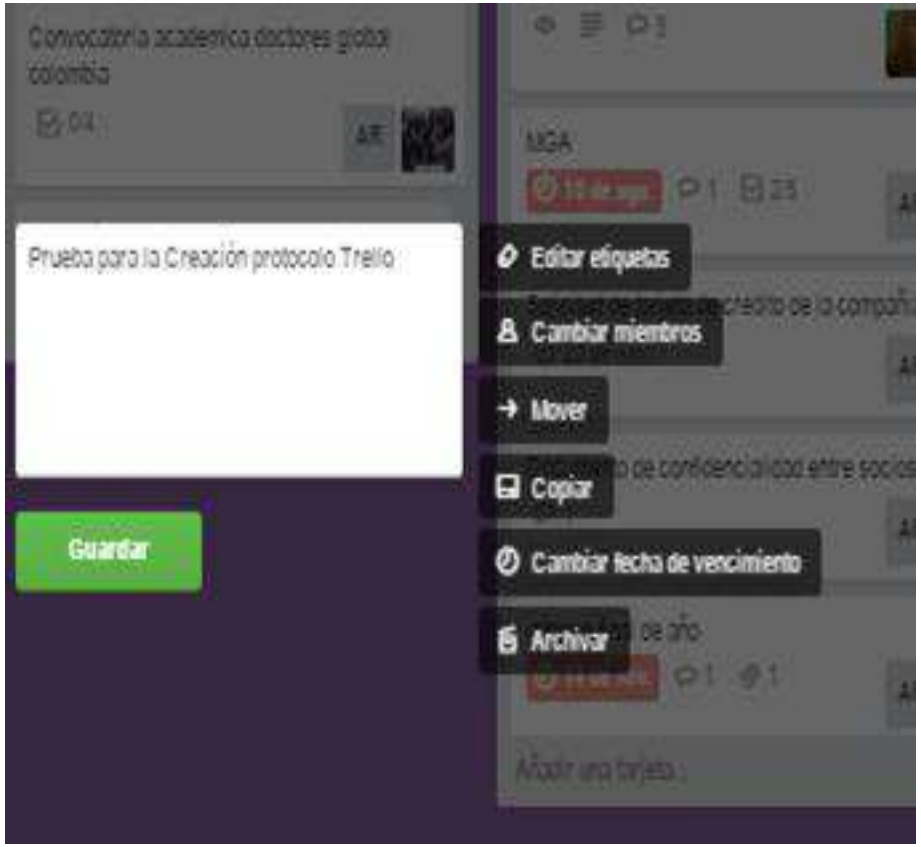
8. Selecciona un título a la tarea, para luego seleccionar *"el lapiz"*



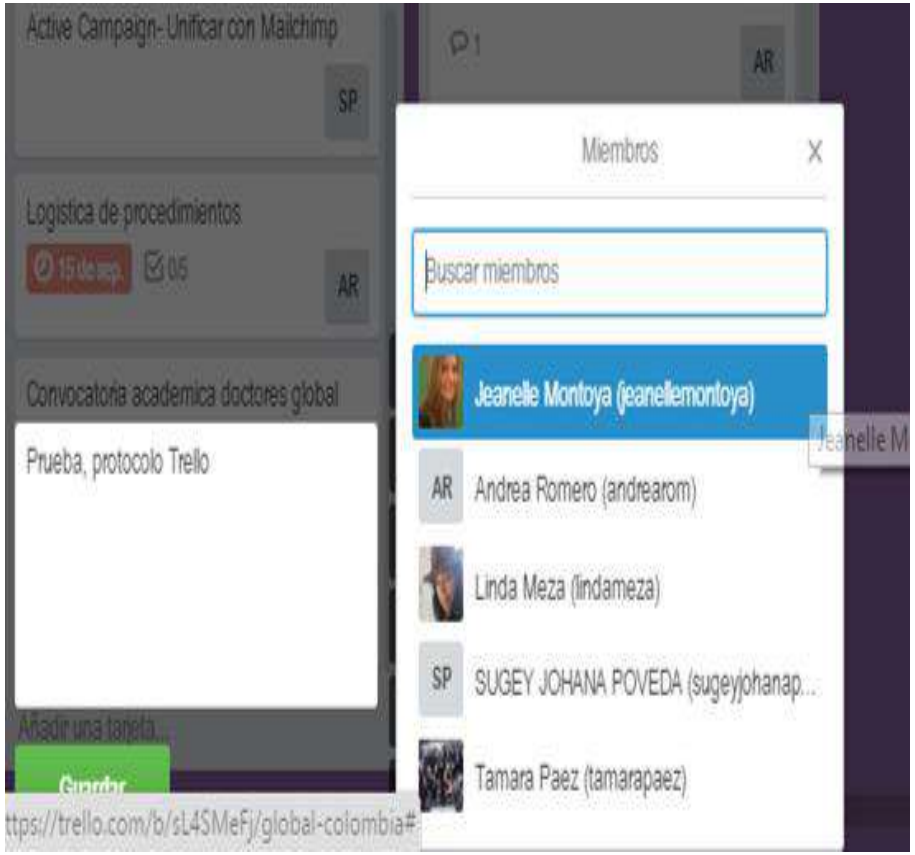
¿CÓMO USARLO PARA ASIGNACIÓN DE TAREAS?

AESTHETIC MARKETING GROUP

9. Puedes definir, cuando debe estar entregada, puedes incluir archivos; y lo más importante, asignar a la persona del equipo que debe llevarla a cabo.



¿CÓMO USARLO PARA ASIGNACIÓN DE TAREAS?

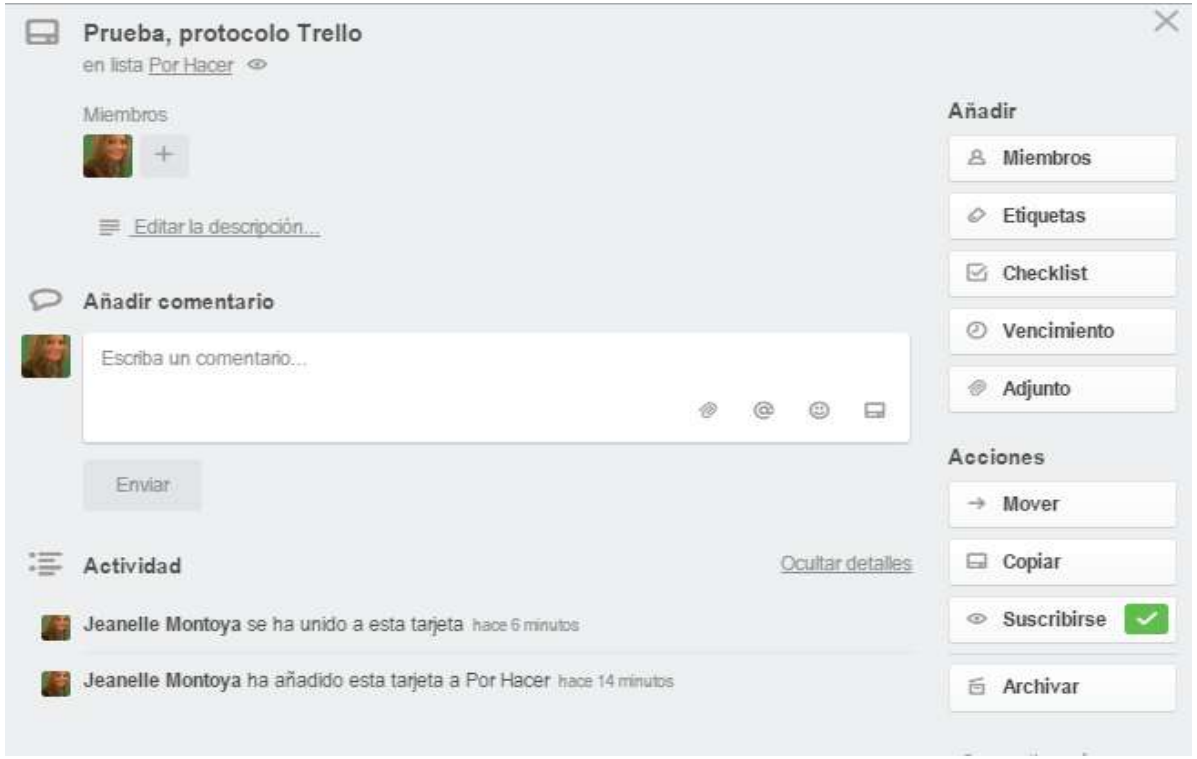


10. Una vez asignada la tarea,
a la persona le llegará
Una notificación a su correo.

¿CÓMO USARLO PARA ASIGNACIÓN DE TAREAS?

AESTHETIC MARKETING GROUP

11. Al seleccionar la tarea recién creada, esta se abre para añadir los comentarios respectivos, o si es necesario hacer un Check list, o adjuntar algún archivo, entre otras opciones.



¿CÓMO USARLO PARA ASIGNACIÓN DE TAREAS?

AESTHETIC MARKETING GROUP

- La tarea se traslada del tablero *"Por Hacer"* a *"Haciendo"* cuando se inicia a ejecutar la tarea.
- Cuando se termina la tarea, se traslada al tablero *"Revisión"*.
- Únicamente quien crea la tarea es quien la archiva, por lo tanto el proceso va hasta *"Revisión"*.

- En el tablero *"Periódico"*: este tablero tiene asignadas las funciones que se hacen de forma periódica y su respectivo responsable.



CONVERSIÓN MER A MODELO RELACIONAL

Estructura de contenidos

1. INTRODUCCIÓN 3

2. TRANSFORMACIÓN MER A MODELO RELACIONAL..... 3

3. REGLAS GENERALES DE TRANSFORMACIÓN..... 5

3.1 Transformación para las entidades y atributos..... 5

3.2 Transformación para las Relaciones..... 7

4. TRANSFORMACIÓN MER- EXTENDIDO A RELACIONAL..... 11

5. CONSTRUCCIÓN TRANSFORMACIÓN MER- EXTENDIDO A RELACIONAL..... 14

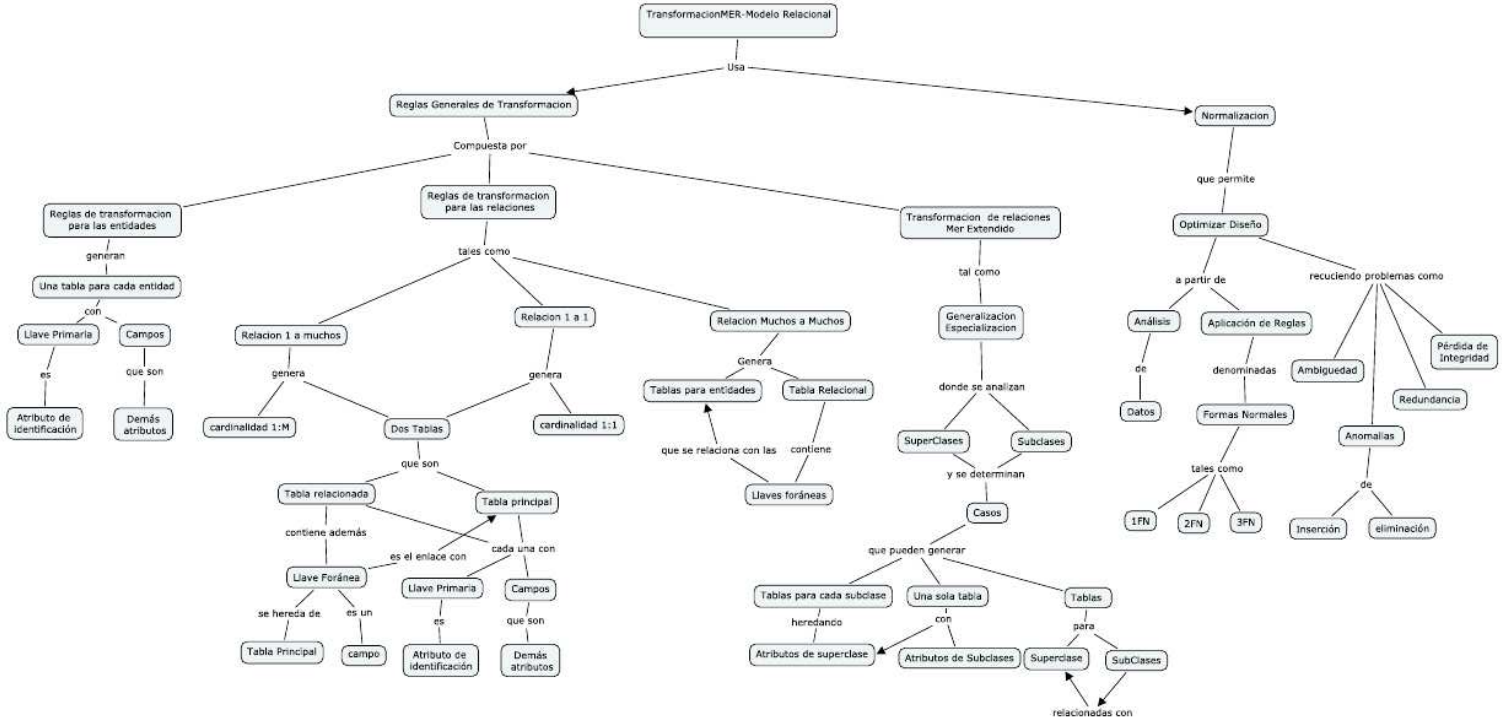
5.1 Relación especialización y generalización..... 15

5.2 Relación Recursiva 20

6. NORMALIZACIÓN..... 23

6.1 Grados de Normalizacion..... 24

Mapa conceptual
CONVERSIÓN MER A MODELO RELACIONAL



1. INTRODUCCIÓN

Almacenar información es uno de los usos más frecuentes que una organización o empresa realiza día a día en un computador. La construcción e implementación de bases de datos permite que la organización pueda almacenar la información relevante del negocio a través de la construcción de una base de datos que se realiza por medio del modelado de datos. En este proceso están identificados tres modelos:

1. Modelo Entidad Relación (MER)
2. Modelo Relacional
3. Modelo Físico

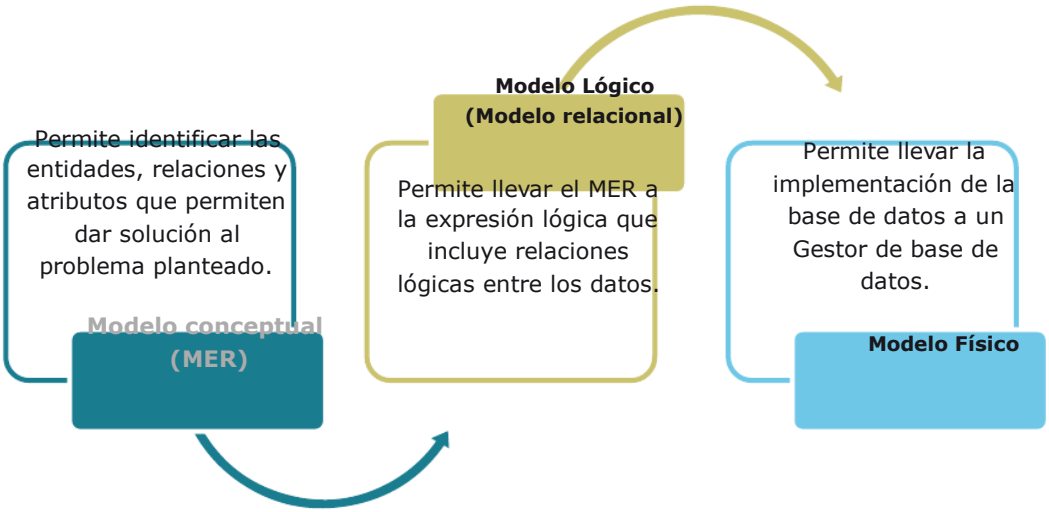
En este material de consulta se presentará el proceso de construcción de un modelo relacional a partir de un MER, lo cual se denomina: **"Transformación del MER a Modelo Relacional".**

La intención es que el aprendiz logre desarrollar y despertar su interés en el modelado de datos, utilizando dicho modelo para la creación e implementación de una base de datos relacional.

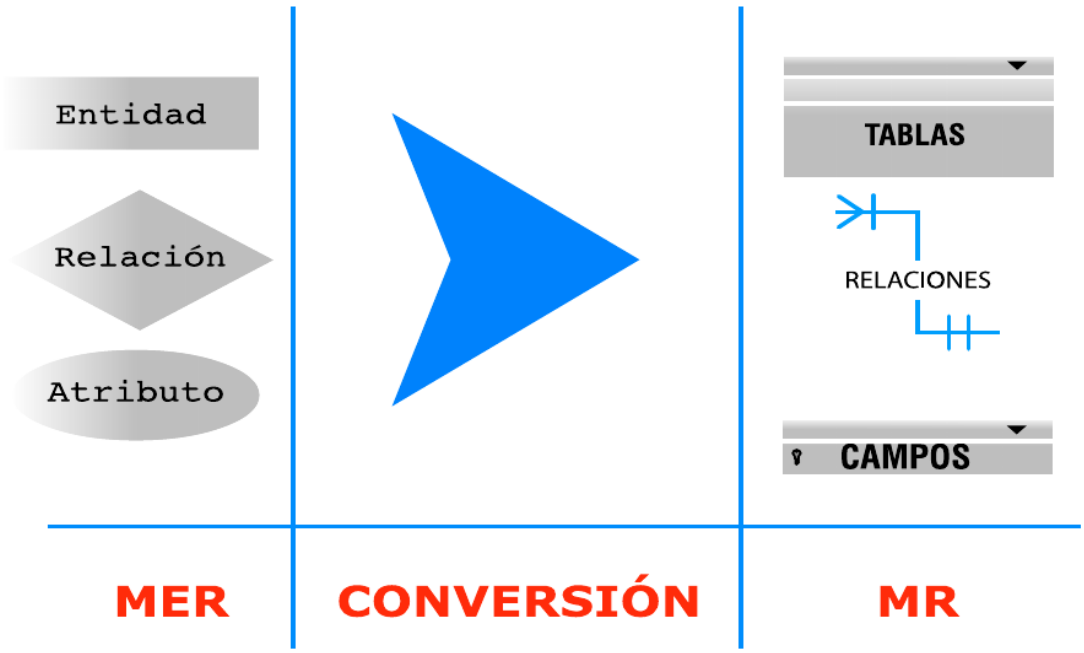
2. TRANSFORMACIÓN MER A MODELO RELACIONAL

Uno de los pasos iniciales para la construcción de una base de datos es el modelado de ellos por medio de un Modelo Entidad Relación (MER), modelo que permite por medio de entidades, atributos y relaciones acercarse a una representación preliminar del diseño de una base de datos.

Luego de terminar un MER debe entrarse en el diseño de un modelo lógico, el cual es llamado modelo relacional. La forma como se logra desarrollar una base de datos relacional es a través del modelado y la metodología de diseño de una base de datos puede estar enfocada en los siguientes 3 pasos:

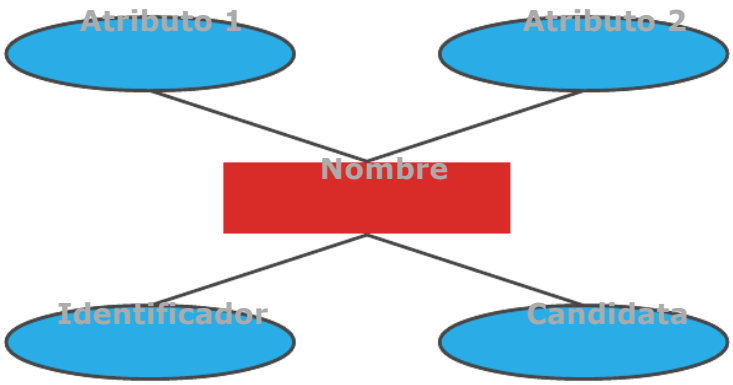


En estos momentos se abordará el proceso de la construcción de un modelo relacional a partir de un MER.



3. REGLAS GENERALES DE TRANSFORMACIÓN

La transformación de un Modelo Entidad Relación a Relacional, se puede expresar mediante un esquema relacional o un modelo relacional.



El esquema relacional consta de palabras y relaciones mediante flechas que indican la asociación de las entidades; el modelo relacional es un dibujo que representa las tablas, atributos, llaves primarias y llaves foráneas según la transformación realizada.

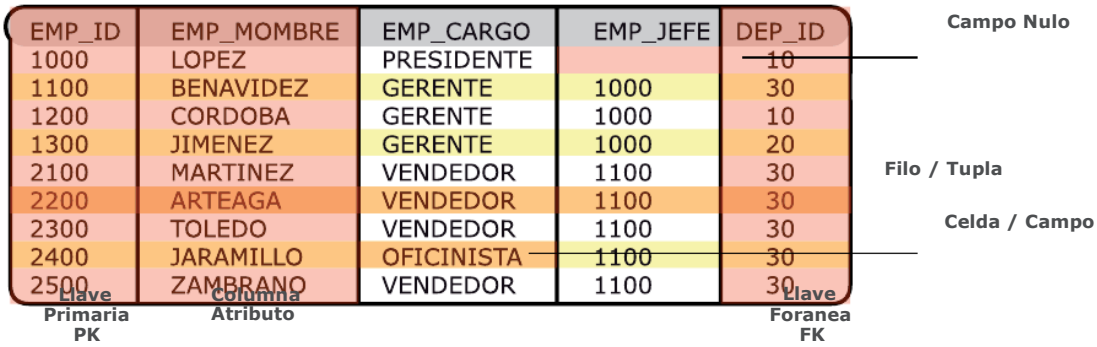
A continuación se explica detalladamente la transformación de un MER según las reglas para las entidades y se expresa en esquema relacional y modelo relacional.

NombreTabla(Identificador, Atributo1, Atributo2, Candidata)

3.1 Transformación para las entidades y atributos

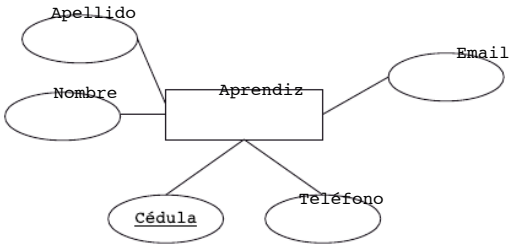
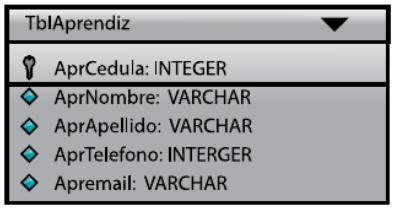
- Entidades. Las entidades pasan a ser tablas
- Atributos. Los atributos pasan a ser columnas o atributos de la tabla.
- Identificadores principales. Pasan a ser claves primarias
- Identificadores candidatos. Pasan a ser claves candidatas.

La siguiente imagen ilustra los anteriores elementos:



Por ejemplo:

Si se tiene la siguiente entidad Aprendiz se convierte entonces:

MER	
Esquema Relacional	TblAprendiz(AprCedula,AprNombre,AprApellido, ,AprTelefono,AprEmail) En este esquema la llave primaria se identifica porque se encuentra subrayado el atributo que será llave primaria
Modelo Relacional	 *La tabla se llama TblAprendiz y la llave primaria se identifica porque el nombre puede estar: subrayado, tener al frente el estereotipo <<pk>> o una llave en color amarillo.*

Nota: para garantizar unicidad de estilos recuerde que debe tener una estandarización para nombrar las tablas y los atributos. Por ejemplo para este caso la entidad aprendiz se convierte en la tabla aprendiz y es llamada como TblAprendiz, el estilo de nombrado es iniciar con las tres primeras letras Tbl para indicar que es tabla, seguido del nombre de la entidad e iniciando con la primera letra en mayúscula.

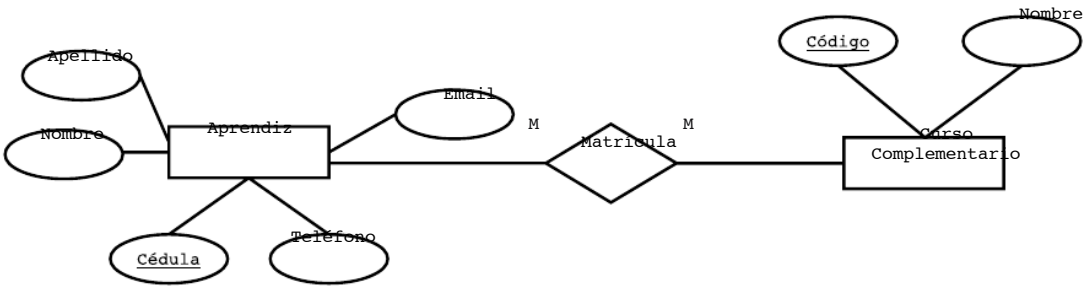
Para el atributo Apellido, en el modelo relacional se convierte en AprApellido indicando Apr que viene de la tabla Aprendiz y seguido del nombre del atributo cada uno iniciando con Mayuscula la primera letra.

3.2 Transformación para las Relaciones

3.2.1 Relación Muchos a Muchos (M..M):

Esta relación se distingue en un MER porque su cardinalidad entre las dos entidades esta designada por (M..M) o (N..M).

Ejemplo:



Transformación Relación Muchos a Muchos (M..M):

Cuando se tiene una relación de este tipo se debe crear una nueva relación (tabla) la relación se transforma en una tabla cuyos atributos son: los atributos de la relación y las claves de las entidades relacionadas (que pasarán a ser claves externas). La clave de la tabla la forman todas las claves externas.

Transformación a Esquema Relacional

Cuando se hace la transformación a esquema relacional se debe tener un esquema de nombrado para las tablas que se generaran, por ejemplo para este caso la entidad aprendiz la tabla generada se denomina TblAprendiz, anteponiéndole tres letras Tbl que indican que es una tabla; de la misma manera para los atributos de esta entidad se le antepone las tres letras Apr para indicar que son atributos de la tabla Aprendiz. Así se continuara el estilo de nombrado para todos los casos de estudio y ejemplos de este material.

TblAprendiz(**AprCedula**,AprNombre, AprApellido , ,AprTelefono,AprEr

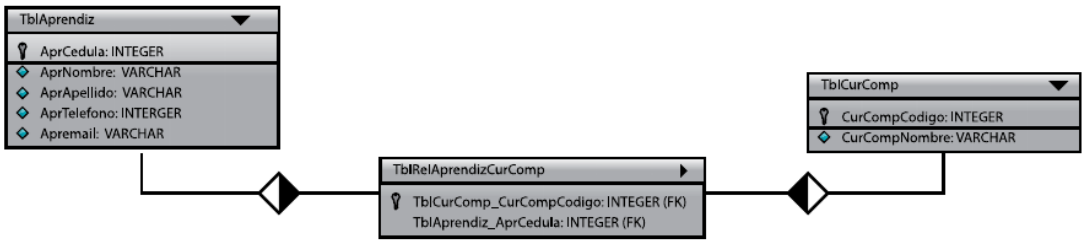
TblCursoComp(**CurCompCodigo**,CurcompNombre)

TblRelAprCurso(**AprCedula**, **CurCompCodigo**)

Transformación a Modelo Relacional

Para realizar las transformaciones a modelo relacional puede utilizar herramientas case, para estos ejemplos se ha utilizado DbDesigner, el uso de ésta última se detallará en los videotutoriales que acompañan el laboratorio correspondiente.

Se visualiza la nueva tabla donde las llaves primarias son a la vez llaves foráneas, nótese que la cardinalidad de la relación está representada por los rombos en blanco (1) o sombreados (M)



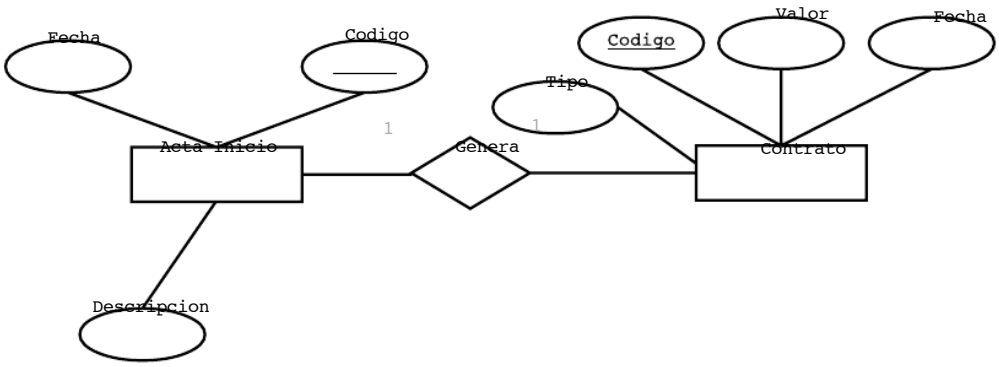
La nueva tabla creada es llamada TblRelAprendizCurComp, que tiene incluida las dos llaves primarias de la tabla TblAprendiz y TblCurComp. Estas llaves en la nueva tabla son llamadas llaves foráneas y son representadas por un <fk> en el modelo.

Nota: las llaves primarias pasada a la nueva tabla se llaman llaves foráneas.

3.2.2 Relación 1 a 1 (1..1):

Esta relación se distingue en un MER porque su cardinalidad entre las dos entidades esta designada por (1..1).

Ejemplo:



Transformación Relación 1 a 1 (1..1):

Cuando se tiene una relación de este tipo se envía la llave primaria de una tabla a la otra tabla que participa de la relación, se envía la llave primaria que se considere sea más relevante para enviar a la otra tabla.

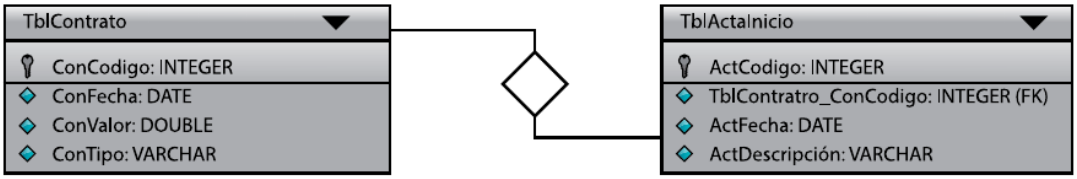
Esquema Relacional

TblContrato(***ConCodigo***,ConValor,ConFecha,ConTipo).

TblActaInicio(***ActCodigo***,ActFecha,ActDescricpción,TblContratoConCodigo)

Para este ejemplo específico se ha pasado la llave primaria del contrato (ConCodigo) a la tabla ActaInicio.

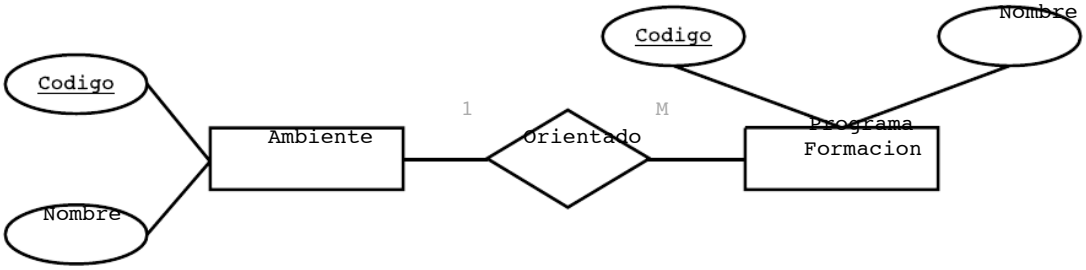
Transformación a Modelo Relacional



En la tabla TblActaInicio se incluye la clave primaria de la tabla TblContrato (ConCodigo) quedando como llave foránea.

3.2.3 Relación 1 a Muchos (1..M):

Esta relación se distingue en un MER porque su cardinalidad entre las dos entidades esta designada por (1..M).



Transformación Relación 1 a Muchos (1..M):

Cuando se tiene una relación de este tipo se debe colocar o enviar la llave primaria de la tabla que tiene en cardinalidad 1 a la tabla que tiene cardinalidad M

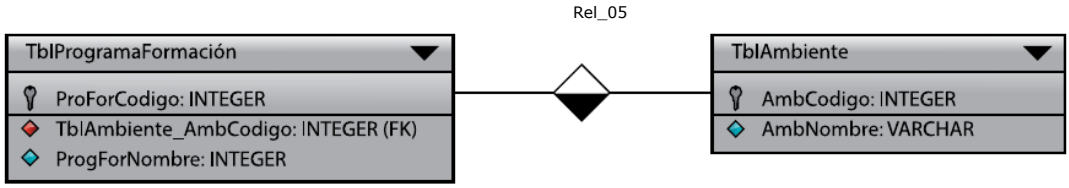
Esquema relacional

TblAmbiente(**AmbCodigo**, AmbNombre)

TblProgramaFormacion(**PrgForCodigo**,PrgForNombre,**AmbCodigo**)

Visualice que La tabla TblAmbiente tiene como cardinalidad 1, por la tanto la clave primaria de esta tabla es pasada a la tabla TblProgramaFormacion.

Transformación a Modelo Relacional



Note que la clave de la tabla TblAmbiente es pasada a la Tabla TblProgramaFormacion la cual tiene su estereotipo <fk>.

4. TRANSFORMACIÓN MER- EXTENDIDO A RELACIONAL

A continuación se presenta un ejemplo de transformación de modelo entidad relación a Relacional, que aplica las reglas descritas. Para este ejemplo se toma una situación problémica y su modelo conceptual, para lo cual se presenta la descripción del problema, el MER y el modelo relacional resultante.

Descripción del Problema

En el SENA Regional Cauca del Centro de Teleinformática y Producción industrial el coordinador requiere un sistema de información que permita gestionar los horarios de los instructores pertenecientes a un centro de formación que orientan formación profesional integral, tanto de planta como de contrato, tenga en cuenta que un instructor de contrato puede tener su relación de contrato por horas o asignación mensual.

Para lo cual es necesario conocer: Un instructor orienta formación profesional (clase) en un programa de formación (el programa de formación se refiere a las tecnologías o tecnólogos que ofrece el SENA), o en un curso complementario (los cuales son cursos de duración de 2 a 3 meses y solicitados por empresas del sector para capacitar a sus empleados).

Ambos: Programa de formación y complementarios son identificados por un código.

Cuando un aprendiz se matricula a un programa de formación o curso complementario tiene que durante toda su etapa lectiva (clase), cursar competencias laborales.

Una competencia podría asemejarse como una asignatura que un aprendiz debe cursar, pero para este caso siempre manejaremos el término competencia.

Asignación de Competencias: Para asignar una competencia a un instructor es necesario que esté legalmente contratado por el centro o que tenga resolución de nombramiento y según su especialidad se le asignan las competencias que va a orientar.

Ambientes de formación: Es el sitio específico (salón) donde se orientaran las competencias asignadas. Cada ambiente tiene una codificación asignada.

Reglas del negocio: Un instructor de contrato puede iniciar labores una vez tenga el acta de inicio.

Tenga en cuenta que un instructor puede laborar de lunes a viernes de 7:00 a 10:00 p.m. de la noche, sábados y domingos de 8:00 a 6:00 p.m.

Un instructor puede orientar diferentes competencias en diferentes programas de formación y cursos complementarios.

Un instructor no puede orientar una competencia en diferentes ambientes.

Dos o más instructores pueden orientar una o más competencias en un mismo ambiente.

Usuarios: Los usuarios que interactúan con la base de datos

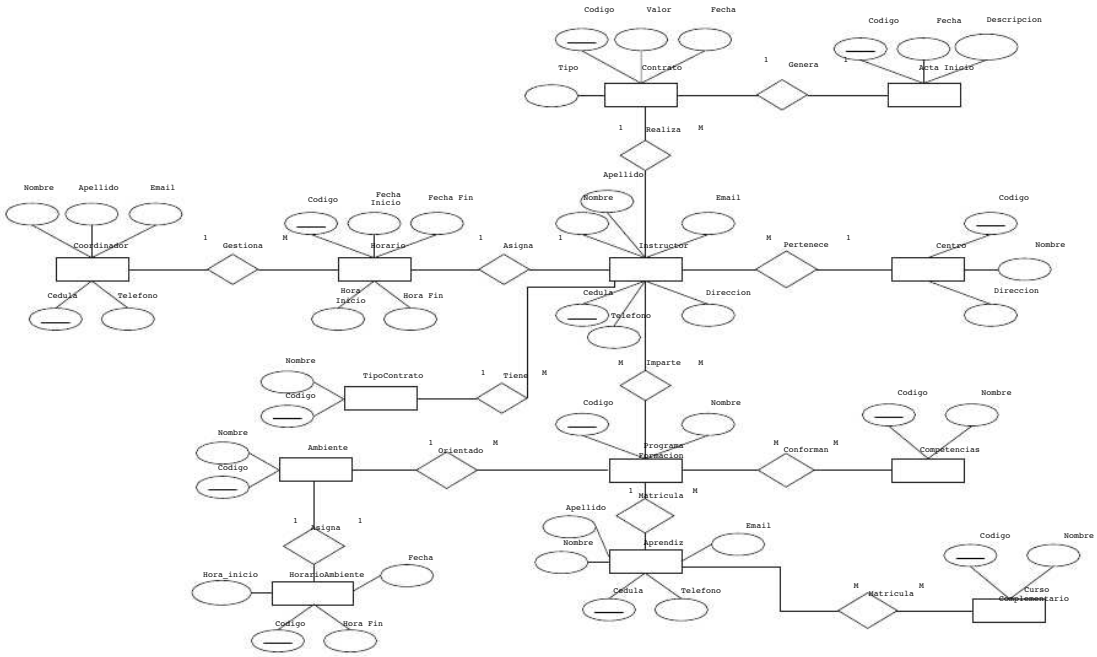
Coordinador académico

Instructores del SENA centro de Teleinformática y PI Regional Cauca

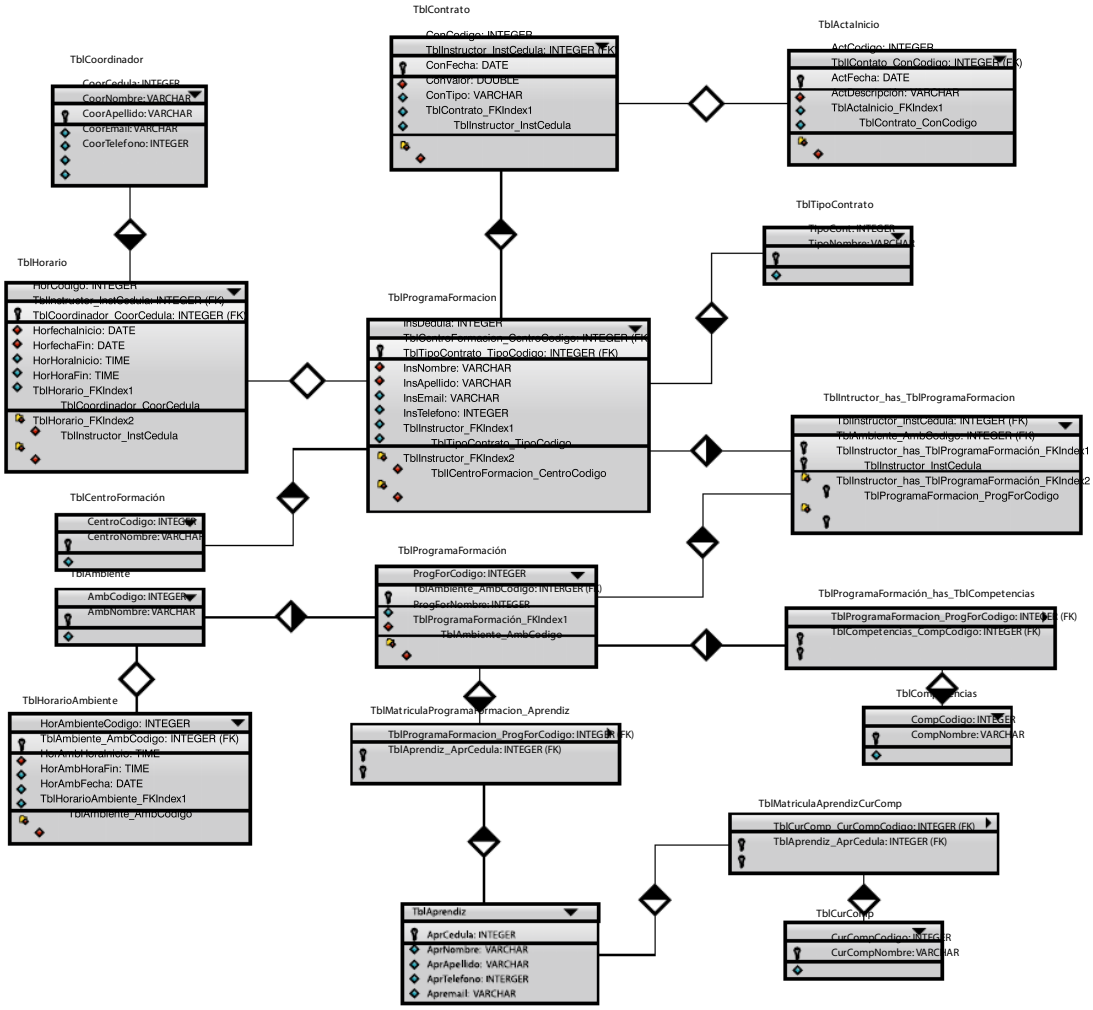
Informes por Usuario: Los informes necesarios para mostrar a los usuarios identificados son:

- Visualizar el horario de todos los instructores de planta.
- Visualizar el horario de todos los instructores de contrato
- Visualizar el horario de un solo instructor.
- Visualizar el horario de todos los ambientes
- Visualizar el horario de un solo ambiente.
- Visualizar el horario de todos los programas de formación.

Modelo Entidad Relación



Modelo Relacional

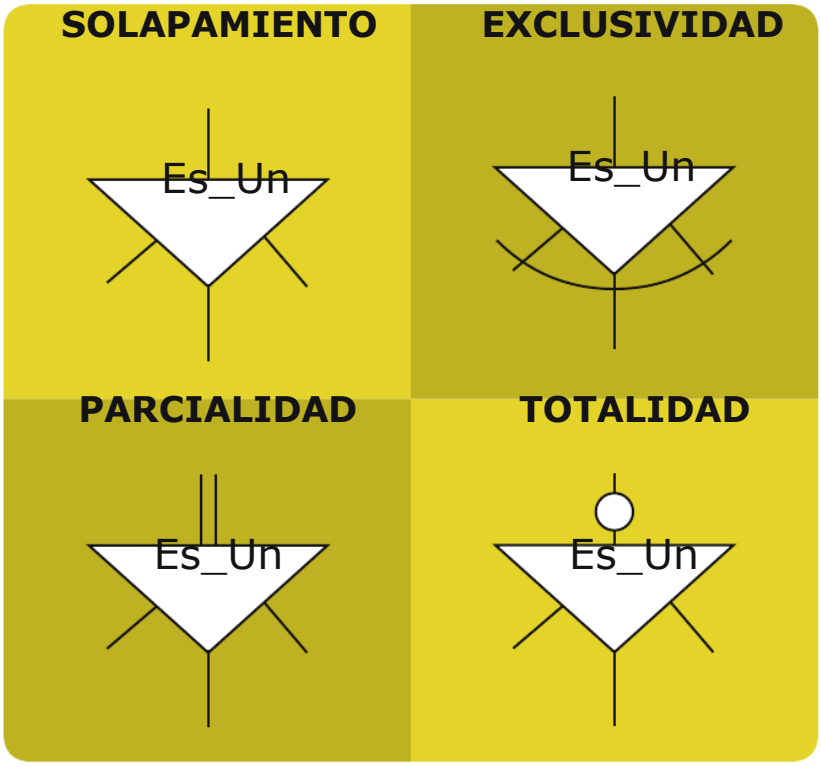


5. CONSTRUCCIÓN TRANSFORMACIÓN MER-EXTENDIDO A RELACIONAL

El MER-Extendido ha sido creado para modelar otros tipos de relaciones que no contempla el MER tradicional, los cuales son la generalización y especialización, recursivas, inclusividad y exculsividad. A continuación se muestra como se transforma cada una de estas relaciones a un modelo relacional.

5.1 Relación especialización y generalización

Recordemos que en un MER-Extendido la generalización puede identificarse como solapamiento, exclusividad, parcialidad o totalidad Figura 2,en estas relaciones se encuentran entidad superclase o supertipos y entidades subclase o subtipos,

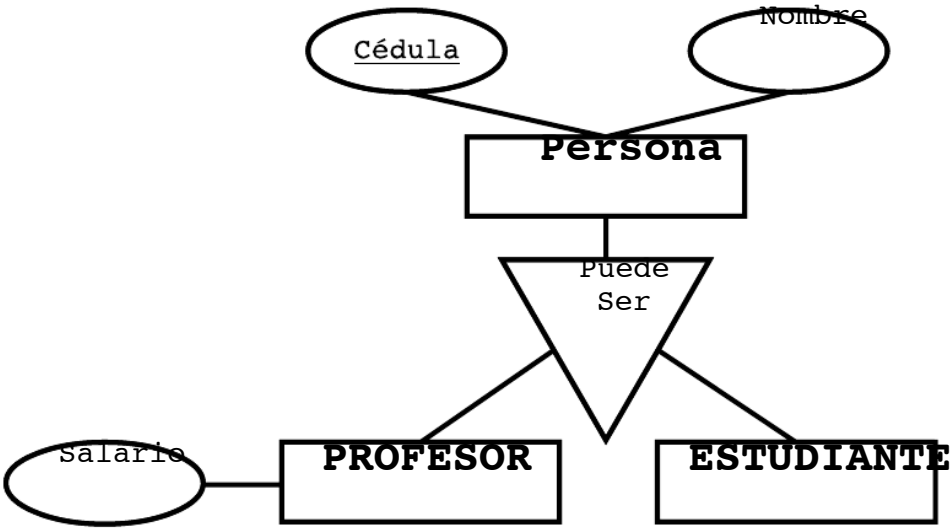


Tipos de relación Generalización en un Mer Extendido

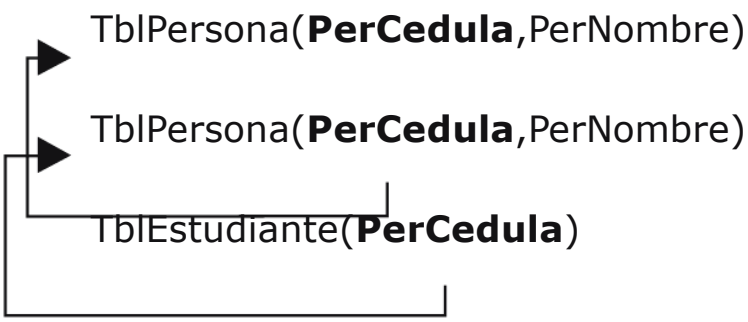
Existen cuatro opciones para la transformación de estos tipos de relaciones

1. Si es solapada y parcial: Crear una relación **(tabla)** para la superclase, con sus atributoscorrespondientes y una relación **(tabla)** para cada subclase con sus atributos incluyendo la clave primaria de la superclase.

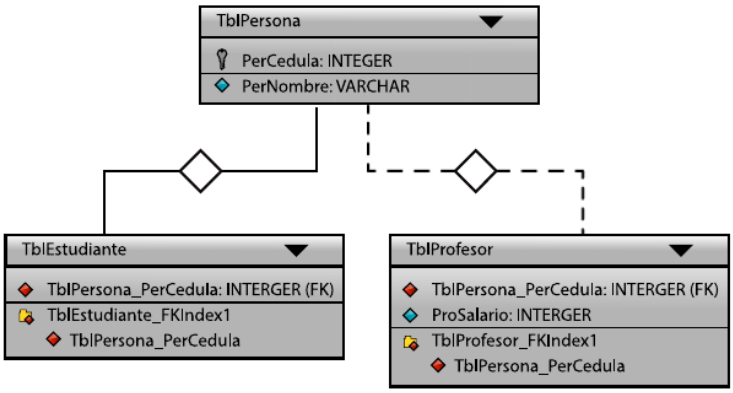
Generalización: esta representación es solapada y parcial, es decir una persona puede ser profesor y estudiante a la vez y no es obligatorio que asuma un subtipo.



Transformación Esquema Relacional



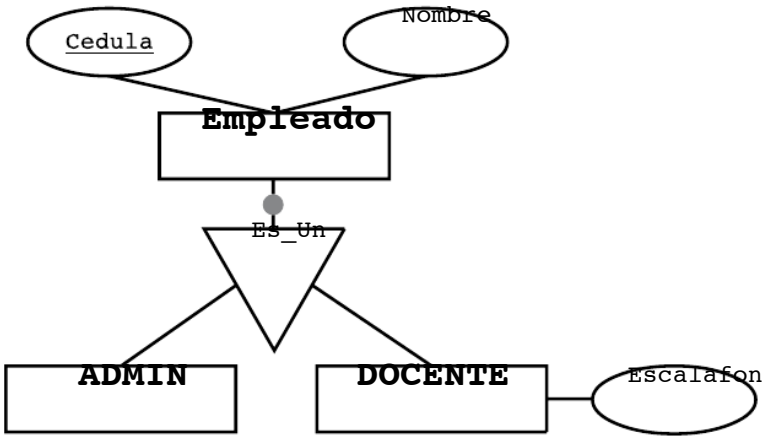
Transformación Modelo Relacional



Observe que se creó la tabla persona con sus campos y se crearon las tablas profesor y estudiante que heredan la llave primaria de la tabla persona (PerCedula) y se convierte en una llave foránea.

2. Si es total: Se crea una relación (tabla) para las entidades subtipo, donde cada una tiene los atributos del supertipo. No se crea una relación para el supertipo.

Relación de totalidad: Obligatoriamente el empleado tiene que tomar algún subtipo o administrador y/o docente.



Transformación Esquema Relacional

TblAdmin(AdminCedula,AdminNombre)

TblDocente(DocCedula,DocNombre)

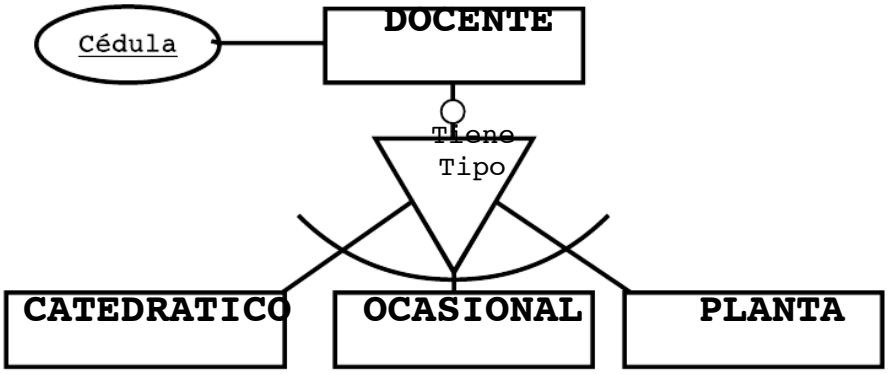
Transformación Modelo Relacional



Observe que quedan dos tablas y se suprime la superclase (empleado), cada una de las tablas hereda los atributos de la superclase y continúan con los atributos propios.

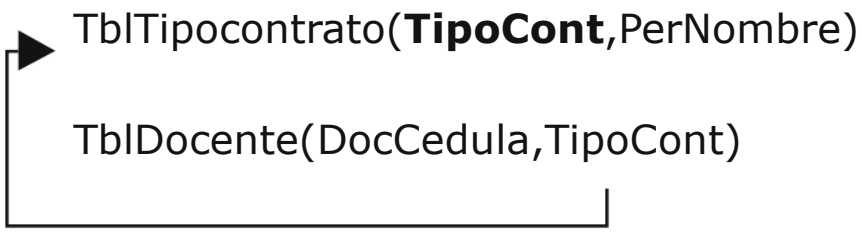
3. Crear una relación (tabla) para almacenar los subtipos y la relación (tabla) superclase. Teniendo en cuenta si es parcial, total, solapado o de exclusividad; Miremos los dos casos que pueden darse en este sentido:

Caso 1: Si la descripción del problema indica que obligatoriamente **pertenece a un solo tipo** es decir la relación es (total y exclusiva), entonces se crea una relación para la SuperClase y otra para el tipo y se le pasa la llave primaria de la tabla tipo a la tabla superclase.



En esta representación el docente puede tener un solo contrato (ocacional, catedrático y planta) y obligatoriamente tiene que tener asociado un contrato.

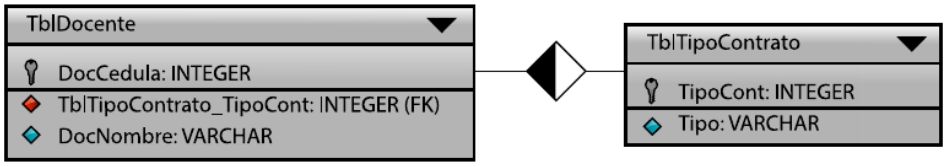
Transformación esquema relacional



Observe que se crea una relación (tabla) que almacena los diferentes tipos de contrato que un docente puede tener y adicionalmente a la tabla TblDocente se la pasa la llave primaria de la tabla TipoContrato.

De esta manera se garantiza que cada docente obligatoriamente tenga un solo tipo de contrato.

Transformación modelo relacional

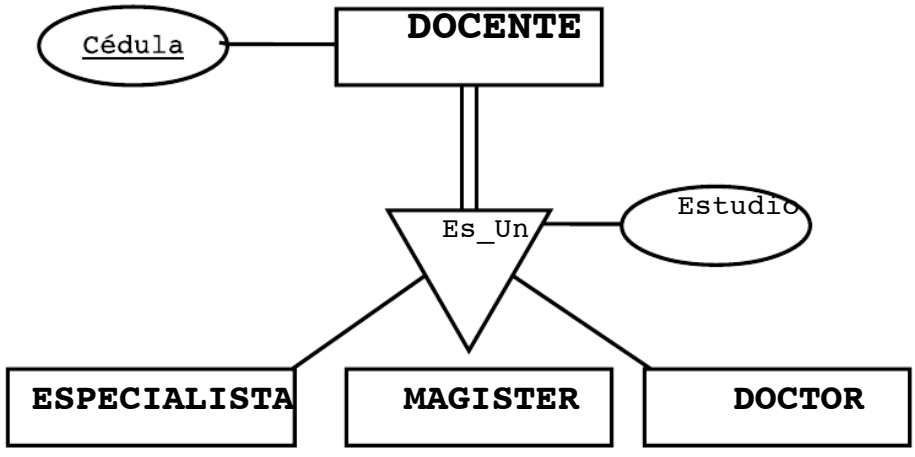


Observe que se genera una nueva tabla llamada TblTipCont que almacenara los diferentes tipos de contrato que puede tener un docente, para mayor explicación esta tabla se almacenara con los siguientes datos.

TblTipCont

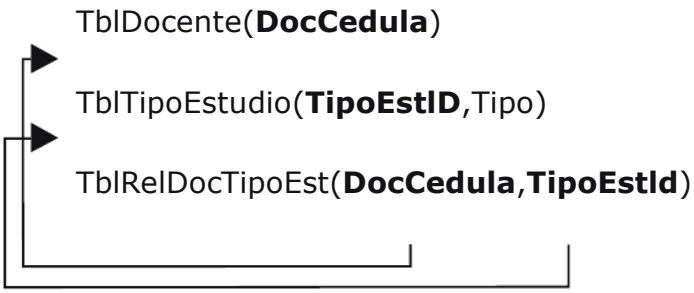
TipoContID	Tipo
1	Catedratico
2	Ocasional
3	Planta

Caso 2: Si la descripción del problema indica que puede **pertenecer a varios tipos** es decir la relación es (parcial y solapada), entonces se crea una relación para la SuperClase, otra para el tipo y se trata la relación superclase y Tipo como una relación de muchos a muchos.

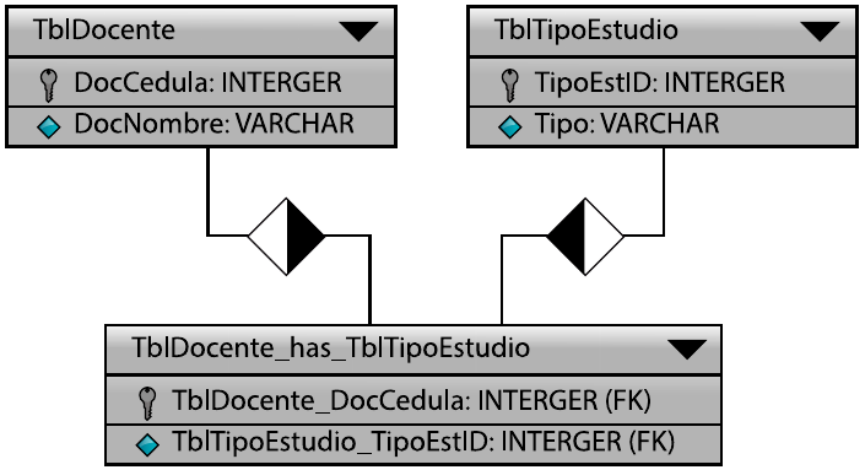


En esta representación un docente puede tener o no un tipo de estudio y puede tenerlos todos a la vez.

Transformación Esquema Relacional



Observe que se crea una relacion (tabla) para la superclase y otra para el tipo y se tratan estas dos relaciones como muchos a muchos, por lo tanto se crea una tabla intermedia que contenga las llaves primarias de las 2 tablas creadas.

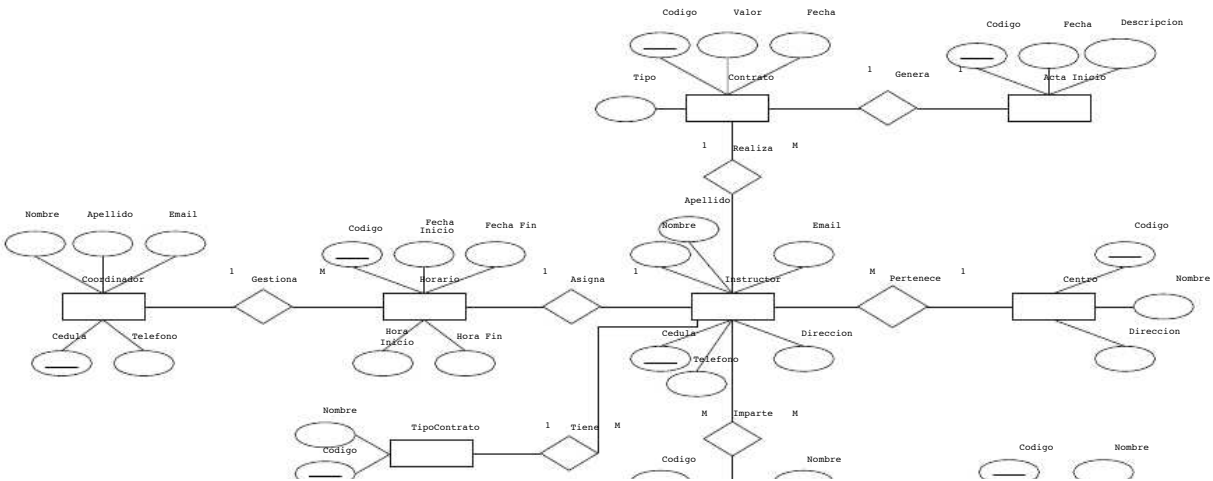


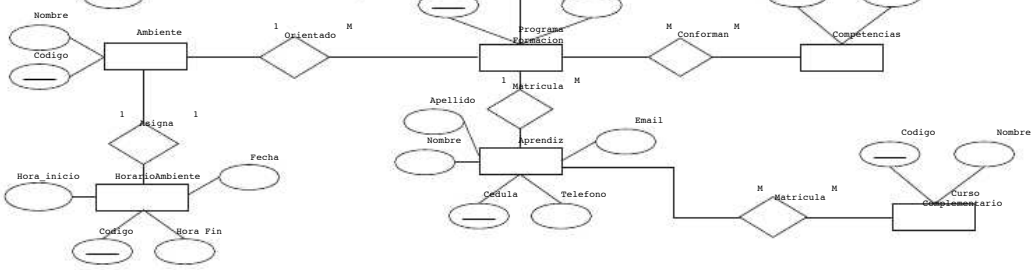
5.2 Relación Recursiva

Las relaciones recursivas se tratan de la misma forma que las otras, sólo que hay que imaginar que la tabla se divide en dos, una por cada rol. Teniendo en cuenta eso, la solución es idéntica a lo ya resuelto en los casos anteriores.

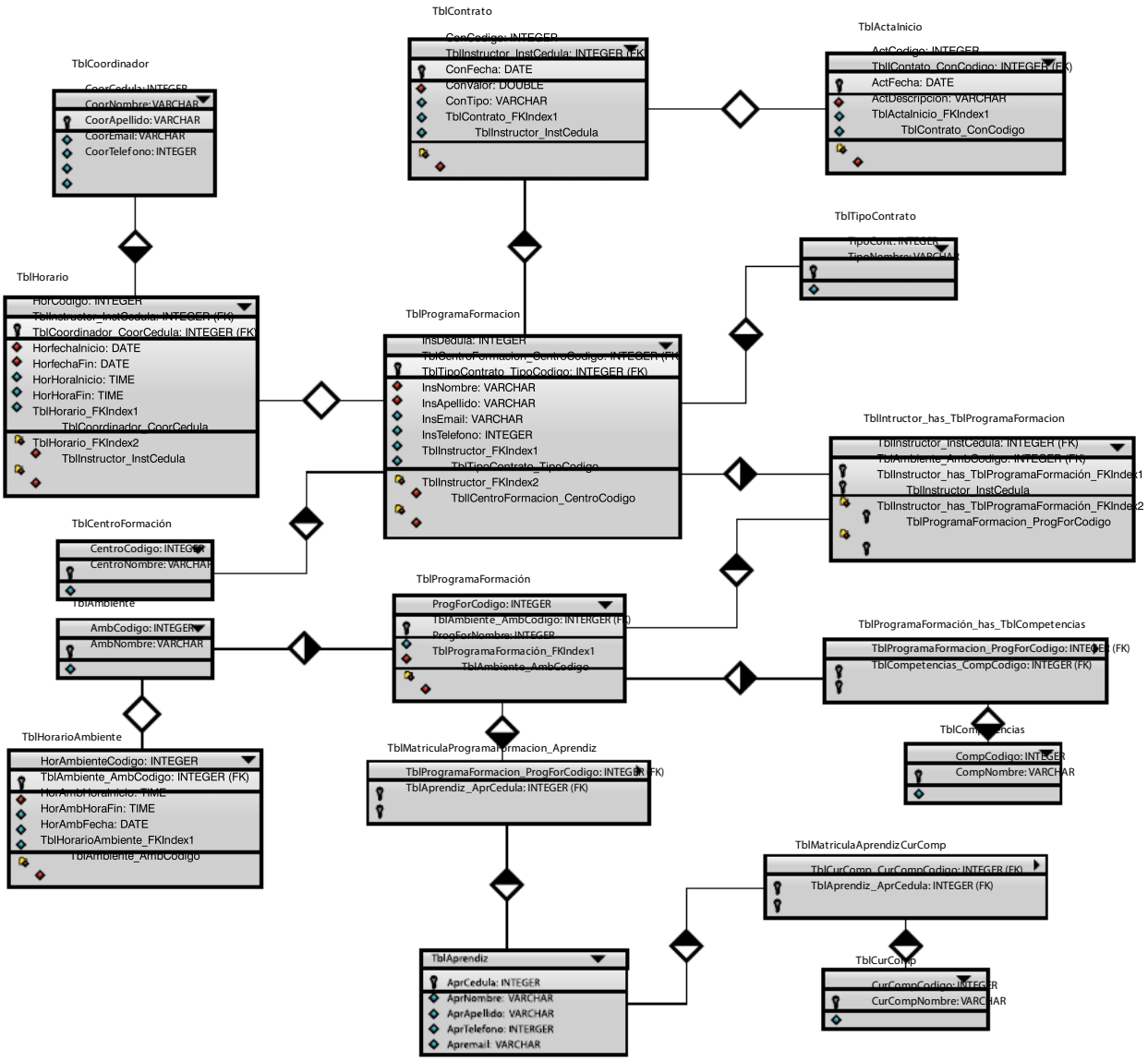
- Visualizar el horario de todos los instructores de planta.
- Visualizar el horario de todos los instructores de contrato
- Visualizar el horario de un solo instructor.
- Visualizar el horario de todos los ambientes
- Visualizar el horario de un solo ambiente.
- Visualizar el horario de todos los programas de formación.

Modelo Entidad Relación




Nombre
Codigo
Ambiente
1
Orientado
M
Programa Formacion
M
Conforman
M
Competencias
1
Matricula
M
Apellido
Nombre
Aprendiz
Email
Cedula
Telefono
Codigo
Nombre
Curso Complementario
M
Matricula
M
1
Asigna
1
Hora_inicio
HorarioAmbiente
Fecha
Codigo
Hora Fin

Modelo Relacional



5. CONSTRUCCIÓN TRANSFORMACIÓN MER-EXTENDIDO A RELACIONAL

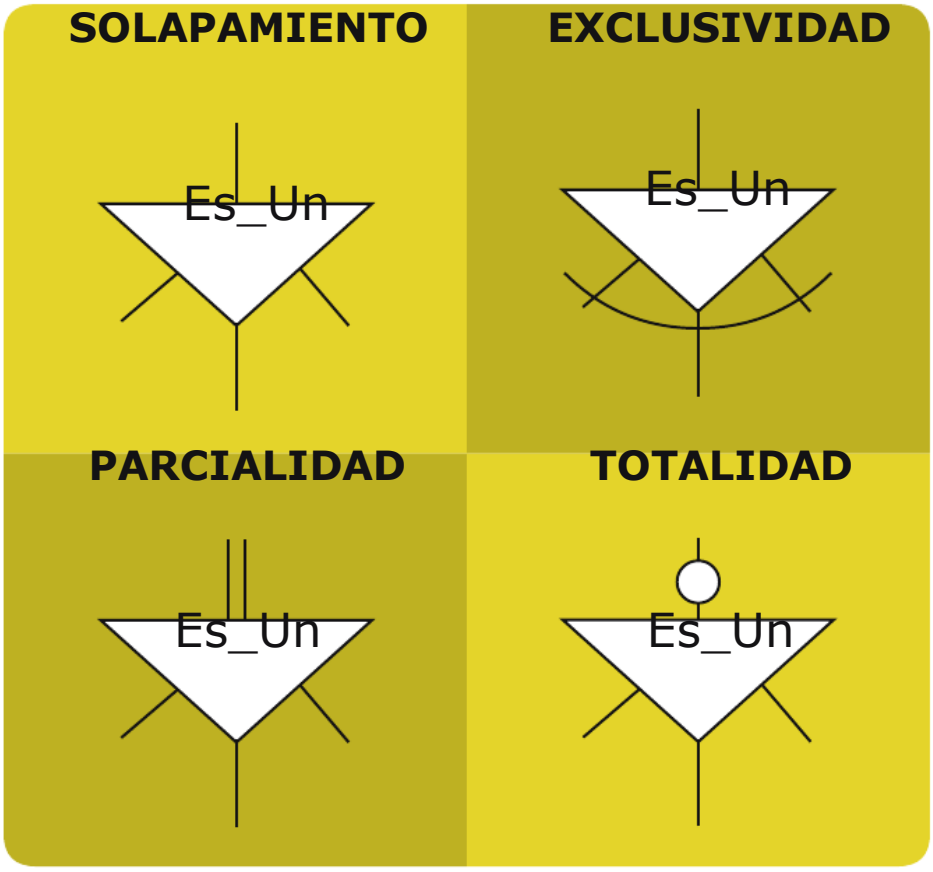
El MER-Extendido ha sido creado para modelar otros tipos de relaciones que no contempla el MER tradicional, los cuales son la generalización y especialización, recursivas, inclusividad y exculsividad. A continuación se muestra como se transforma cada una de estas relaciones a un modelo relacional

relacional.

5.1 Relación especialización y generalización

5.1 Relación especialización y generalización

Recordemos que en un MER-Extendido la generalización puede identificarse como solapamiento, exclusividad, parcialidad o totalidad Figura 2,en estas relaciones se encuentran entidad superclase o supertipos y entidades subclase o subtipos,



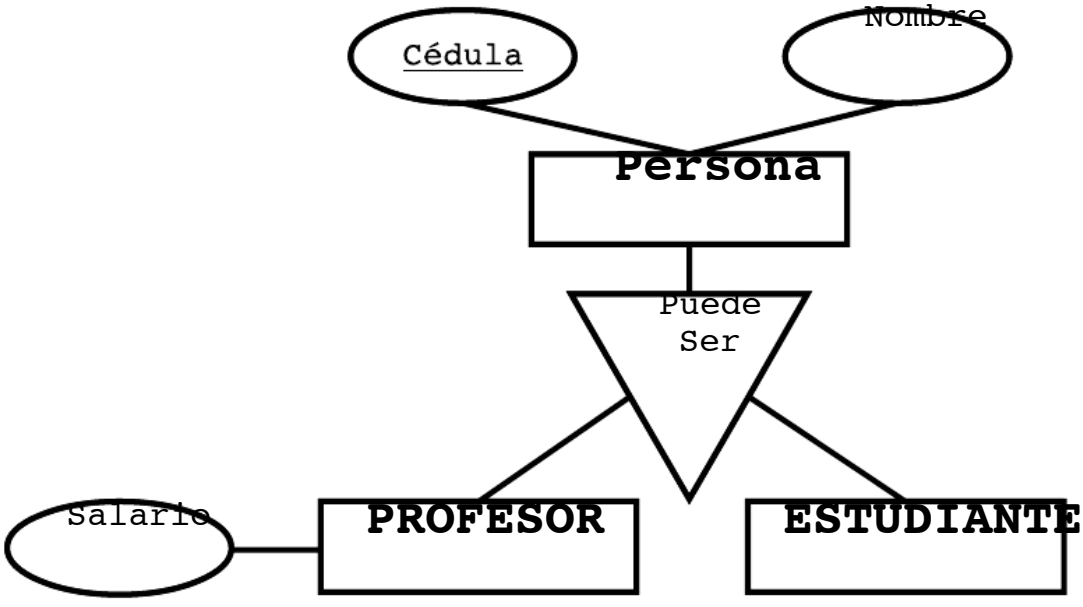
Tipos de relación Generalización en un Mer Extendido

Existen cuatro opciones para la transformación de estos tipos de relaciones

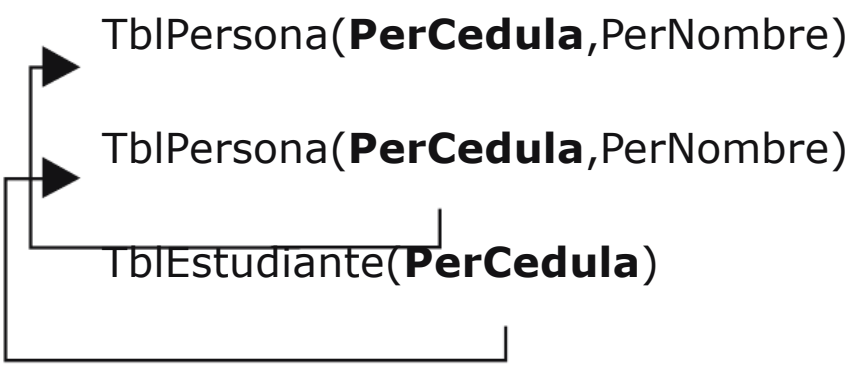
1. Si es solapada y parcial: Crear una relación **(tabla)** para la superclase, con sus atributoscorrespondientes y una relación **(tabla)** para cada subclase con sus atributos incluyendo la clave primaria de la superclase.

Generalización: esta representación es solapada y parcial, es decir una persona puede ser profesor y estudiante a la vez y no es obligatorio que asuma un subtipo.

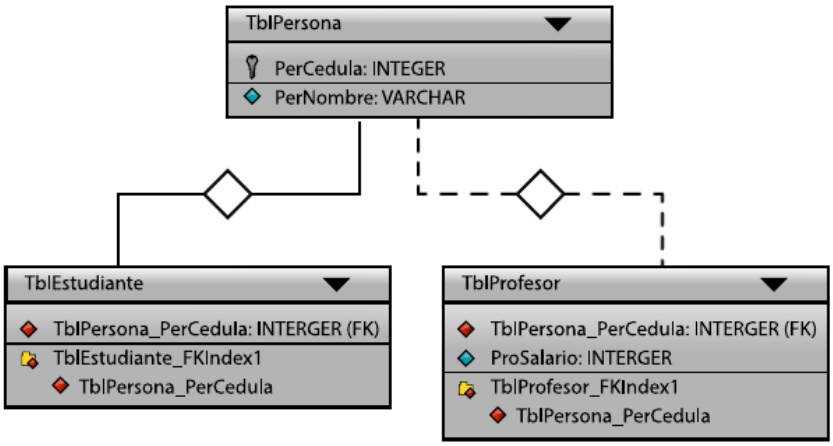
Conversión Mer a Modelo Relacional



Transformación Esquema Relacional



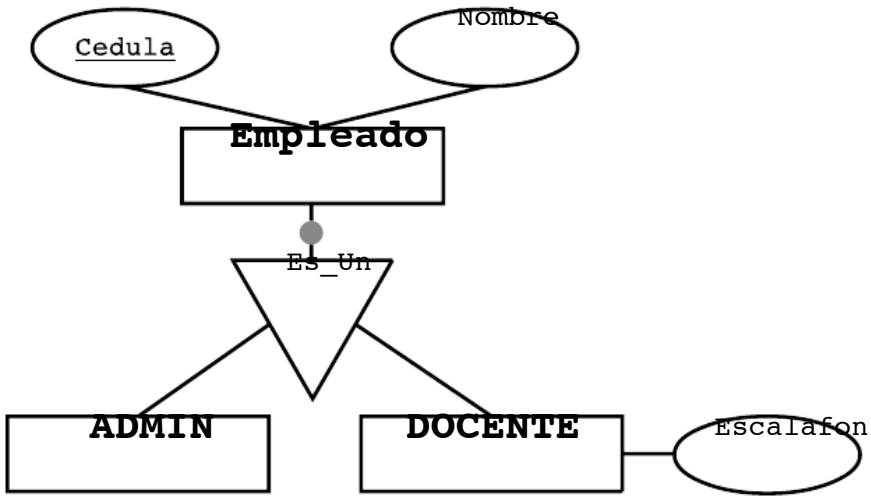
Transformación Modelo Relacional



Observe que se creó la tabla persona con sus campos y se crearon las tablas profesor y estudiante que heredan la llave primaria de la tabla persona (PerCedula) y se convierte en una llave foránea.

2. Si es total: Se crea una relación (tabla) para las entidades subtipo, donde cada una tiene los atributos del supertipo. No se crea una relación para el supertipo.

Relación de totalidad: Obligatoriamente el empleado tiene que tomar algún subtipo o administrador y/o docente.



Transformación Esquema Relacional

TblAdmin(AdminCedula,AdminNombre)

TblDocente(DocCedula,DocNombre)

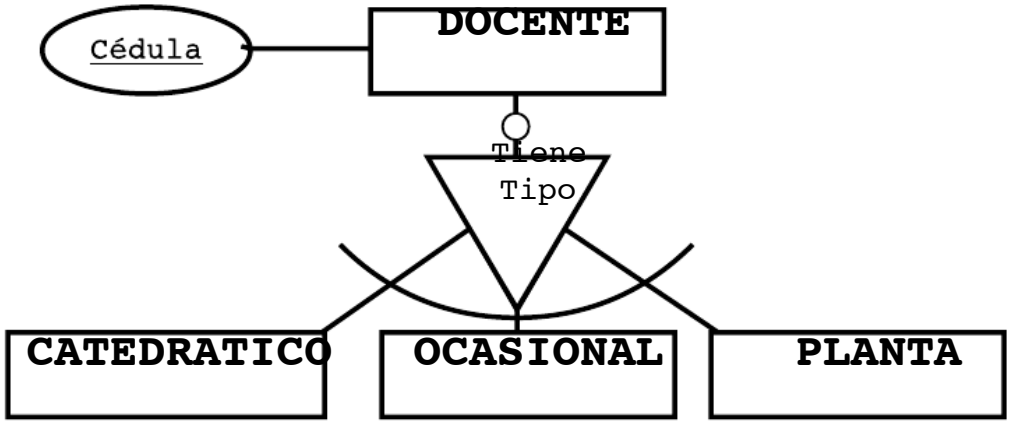
Transformación Modelo Relacional



Observe que quedan dos tablas y se suprime la superclase (empleado), cada una de las tablas hereda los atributos de la superclase y continúan con los atributos propios.

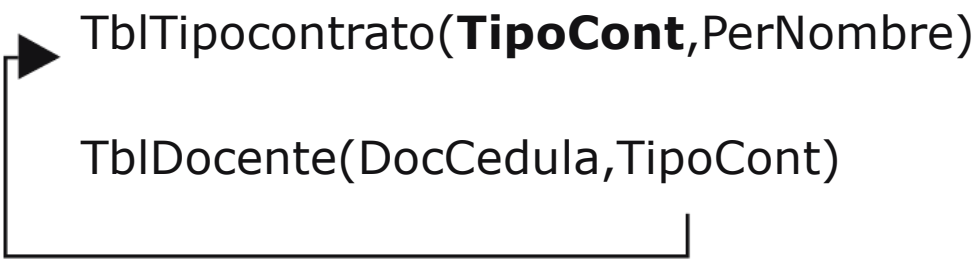
3. Crear una relación (tabla) para almacenar los subtipos y la relación (tabla) superclase. Teniendo en cuenta si es parcial, total, solapado o de exclusividad; Miremos los dos casos que pueden darse en este sentido:

Caso 1: Si la descripción del problema indica que obligatoriamente **pertenece a un solo tipo** es decir la relación es (total y exclusiva), entonces se crea una relación para la SuperClase y otra para el tipo y se le pasa la llave primaria de la tabla tipo a la tabla superclase.



En esta representación el docente puede tener un solo contrato (ocacional, catedrático y planta) y obligatoriamente tiene que tener asociado un contrato.

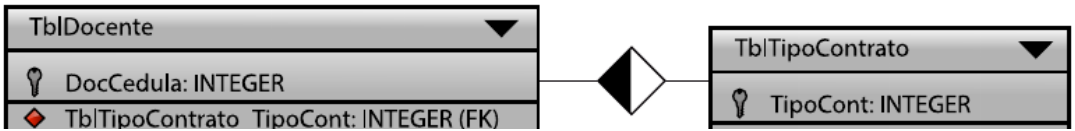
Transformación esquema relacional



Observe que se crea una relación (tabla) que almacena los diferentes tipos de contrato que un docente puede tener y adicionalmente a la tabla TblDocente se la pasa la llave primaria de la tabla TipoContrato.

De esta manera se garantiza que cada docente obligatoriamente tenga un solo tipo de contrato.

Transformación modelo relacional





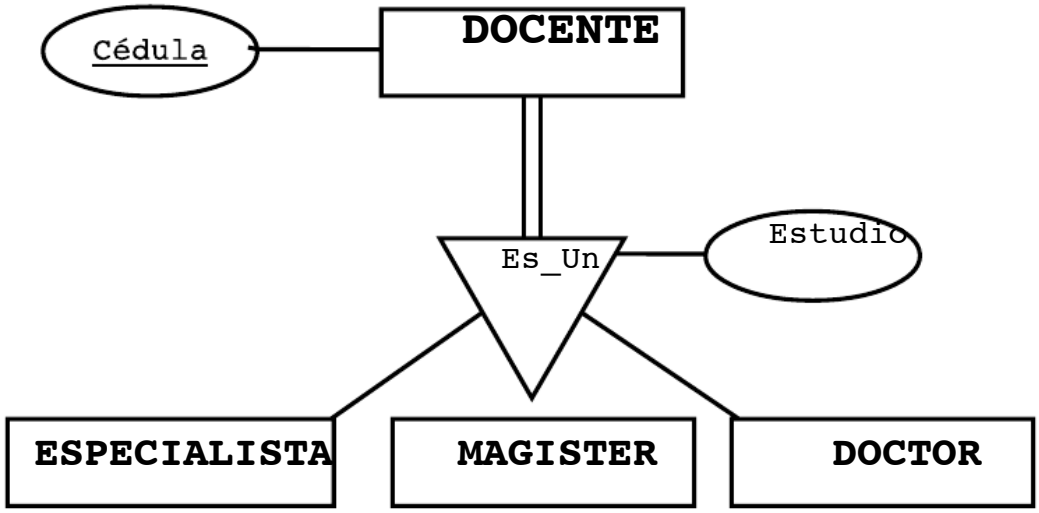


Observe que se genera una nueva tabla llamada TblTipCont que almacenara los diferentes tipos de contrato que puede tener un docente, para mayor explicación esta tabla se almacenara con los siguientes datos.

TblTipCont

TipoContID	Tipo
1	Catedratico
2	Ocasional
3	Planta

Caso 2: Si la descripción del problema indica que puede **pertenecer a varios tipos** es decir la relación es (parcial y solapada), entonces se crea una relación para la SuperClase, otra para el tipo y se trata la relación superclase y Tipo como una relación de muchos a muchos.



En esta representación un docente puede tener o no un tipo de estudio y puede tenerlos todos a la vez.

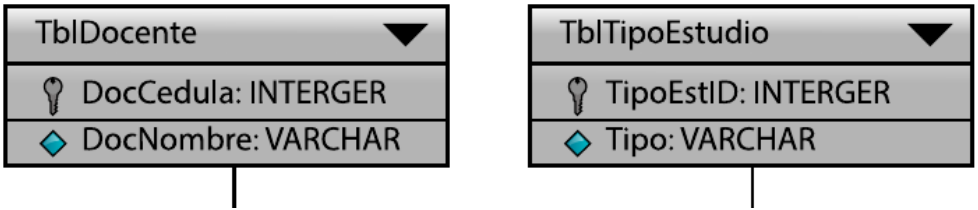
Transformación Esquema Relacional

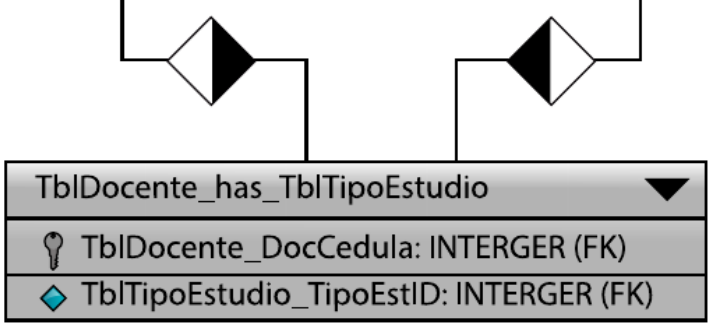
TblDocente(**DocCedula**)

TblTipoEstudio(**TipoEstID**,Tipo)

TblRelDocTipoEst(**DocCedula**,**TipoEstId**)

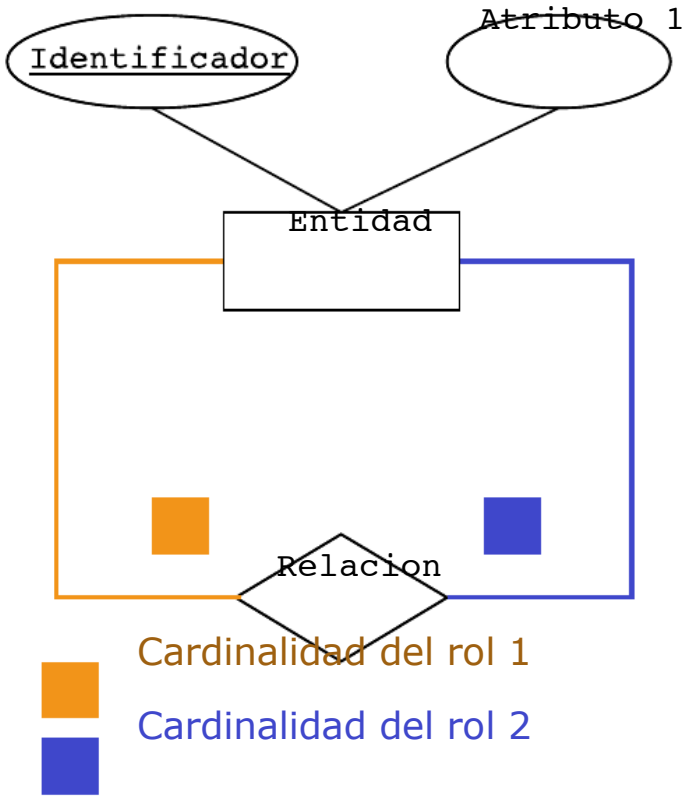
Observe que se crea una relacion (tabla) para la superclase y otra para el tipo y se tratan estas dos relaciones como muchos a muchos, por lo tanto se crea una tabla intermedia que contenga las llaves primarias de las 2 tablas creadas.



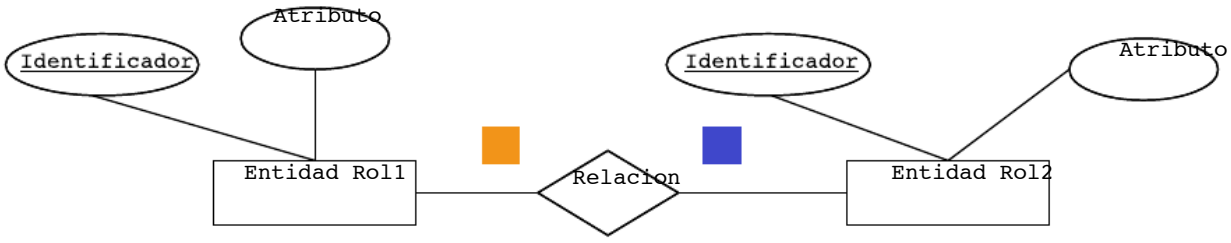


5.2 Relación Recursiva

Las relaciones recursivas se tratan de la misma forma que las otras, sólo que hay que imaginar que la tabla se divide en dos, una por cada rol. Teniendo en cuenta eso, la solución es idéntica a lo ya resuelto en los casos anteriores.



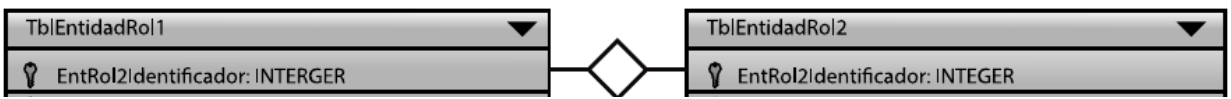
La forma de imaginarse las relaciones recursivas para resolverlas es:



Donde se "duplica la entidad" y se conservan las cardinalidades, luego dependiendo de ésta, se aplica la regla de transformación correspondiente teniendo las siguientes posibilidades:

1 caso: si la cardinalidad de la relación es de 1 a 1:

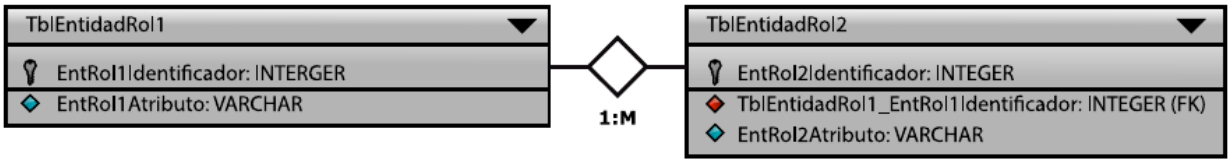
Conversión Mer a Modelo Relacional



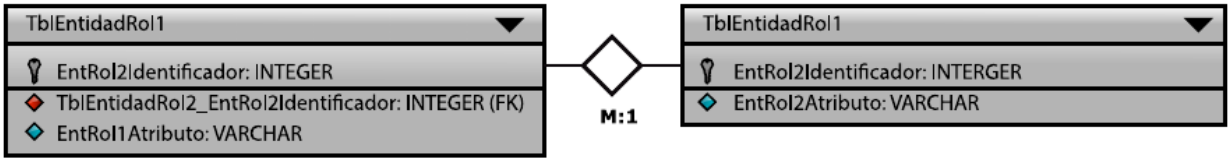


La entidad con Rol débil hereda el atributo de identificación de la otra entidad y pasa a ser su llave foránea

2 caso: si la cardinalidad de la relación es de 1 a M :

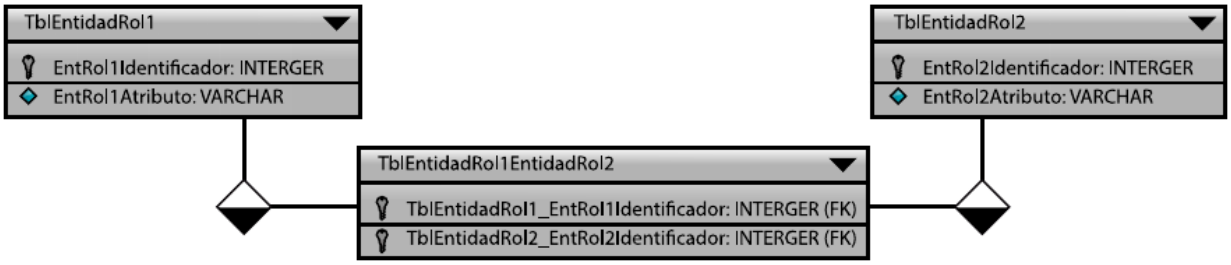


O de M a 1:



En cualquiera de los casos, la entidad con relación muchos hereda el atributo de identificación de la otra y esta se convierte en la llave foránea

3 caso: si la cardinalidad de la relación es de M a M:



Se genera en este caso una tercera tabla que relaciona las dos entidades, está compuesta por los atributos de identificación que hereda de cada entidad, más los atributos propios de la relación (si existieran).