

Manual para Bases de Datos

Éste manual tiene como objetivo aprender los conceptos básicos sobre Bases de datos relacionales, se usarán los conceptos relacionados y derivados de las Bases de Datos relacionales, por lo tanto se usará el lenguaje *SQL*, como principal motor.

Representación

Modelo Chen o *Entidad - Relación*

Es el modelo conceptual más utilizado para el diseño de bases de datos. Fue introducido por Peter Chen en 1976. El modelo entidad relación está formado por un conjunto de conceptos que permiten describir la realidad mediante un conjunto de representaciones gráficas y lingüísticas.

Se basa en la percepción del mundo real que consiste en un conjunto de objetos básicos llamados *entidades* y de *relaciones* entre estos objetos.

A continuación se detallarán los componentes de éste modelo:

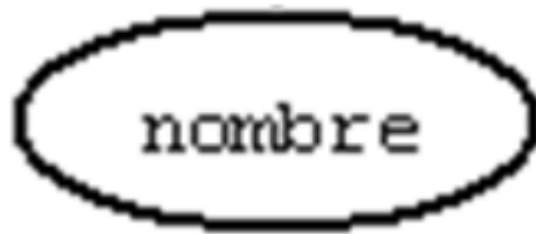
Entidad

Es un objeto real o abstracto de interés, sobre el que se recoge información y se representa gráficamente mediante un rectángulo y su nombre aparece en el interior en mayúsculas. Un nombre de entidad sólo puede aparecer una vez en el esquema conceptual. Generalmente se expresa con sustantivos.



Atributos

Es una propiedad o característica asociada a una determinada entidad o relación y por lo tanto común a todos los ejemplares. La representación grafica es por medio de una elipse etiquetada con letra en minúsculas.



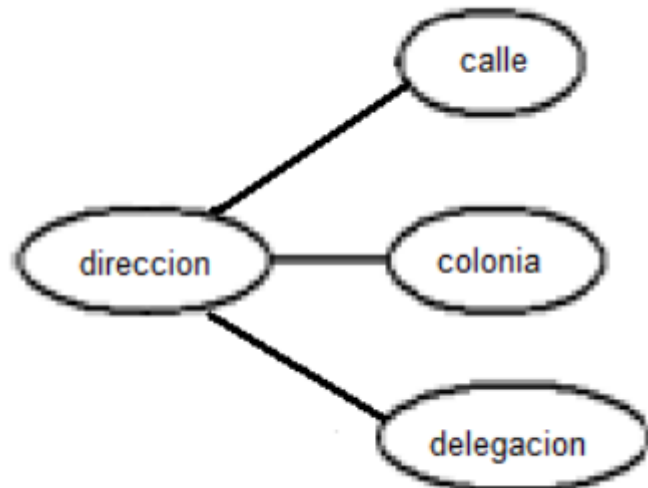
Tipos de Atributos (Opcional)

En función de las características respecto de la entidad que definen, se distinguen varios tipos de atributos

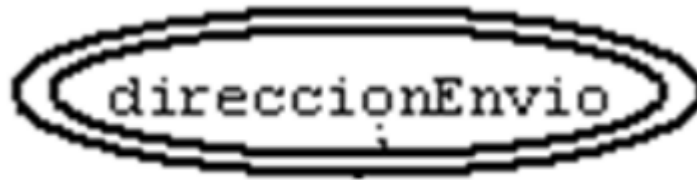
- **Simples:** No se Subdividen



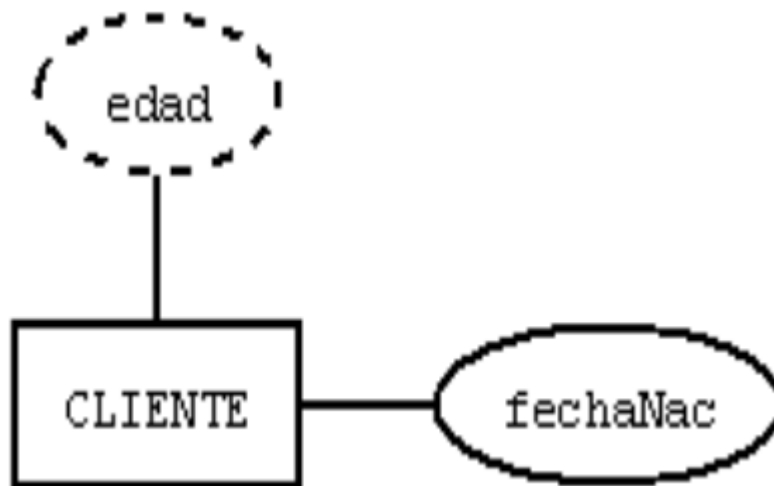
- **Compuestos:** Se dividen en otro atributos



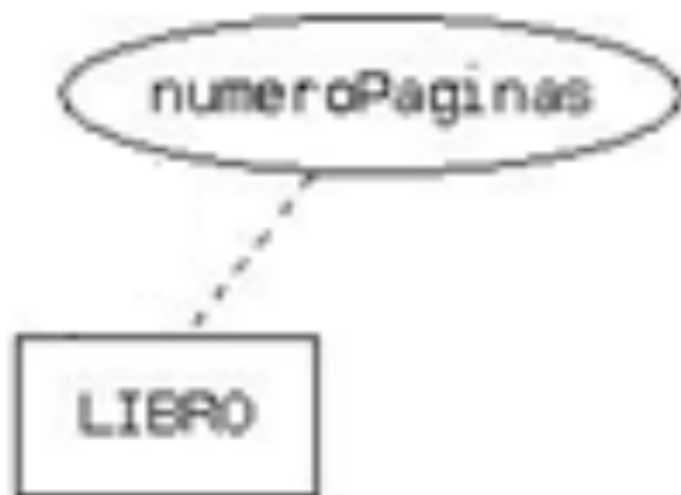
- **Multivalorados:** Tiene un conjunto de valores para una entidad concreta. Se representa con doble elipse.



- **Derivados:** Cuando un valor puede calcularse u obtenerse a partir de otro. Se representa con una elipse con línea discontinua.



- **Opcionales:** Son usados cuando es posible desconocer el valor del atributo para cierta entidad o no se tiene un valor aplicable.



Cabe destacar que esta característica es exclusiva de esta notación.

Claves o Llaves

La clave o llave de una entidad es un atributo único e irrepetible, del cual se puede acceder facilmente a través de el.

- **Clave ó llave Primaria:** Es un atributo o conjunto de atributos que identifican en forma única a una entidad. Se representa subrayando el nombre del atributo.



- **Clave ó llave candidata:** Es un atributo en una entidad débil que la identifica junto con la clave primaria de la entidad fuerte. Se representa subrayando en forma discontinua el atributo.



Nota: Para casos especiales puede darse el concepto de llave Artificial, la cual se denota con la notación *NombreEntidad_ID*.

Relación

Es una asociación, vinculación o correspondencia entre entidades. Se representa gráficamente con un rombo etiquetado en letras minúsculas. Generalmente representadas por verbos .



Ejemplo: Compra es un tipo de relación que vincula las entidades CLIENTE Y PRODUCTO.



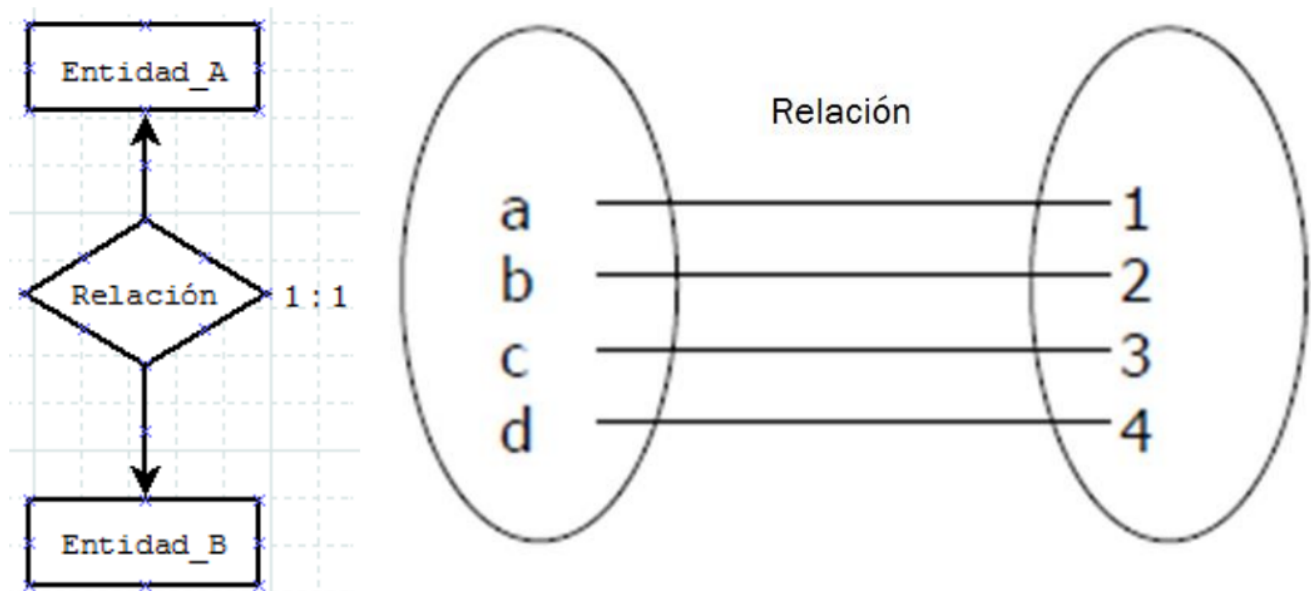
Una relación esta caracterizada por las siguientes características:

- **Nombre:** Debe de tener un nombre que la identifique unívocamente.
- **Grado:** Número de tipos de entidad sobre las que se realiza la asociación. La relación del ejemplo anterior es binaria.
- **Tipo de Correspondencia:** Número máximo de ejemplares de cada tipo de entidad que pueden intervenir en un ejemplar del tipo de relación. A esta propiedad también se le denomina *Cardinalidad*

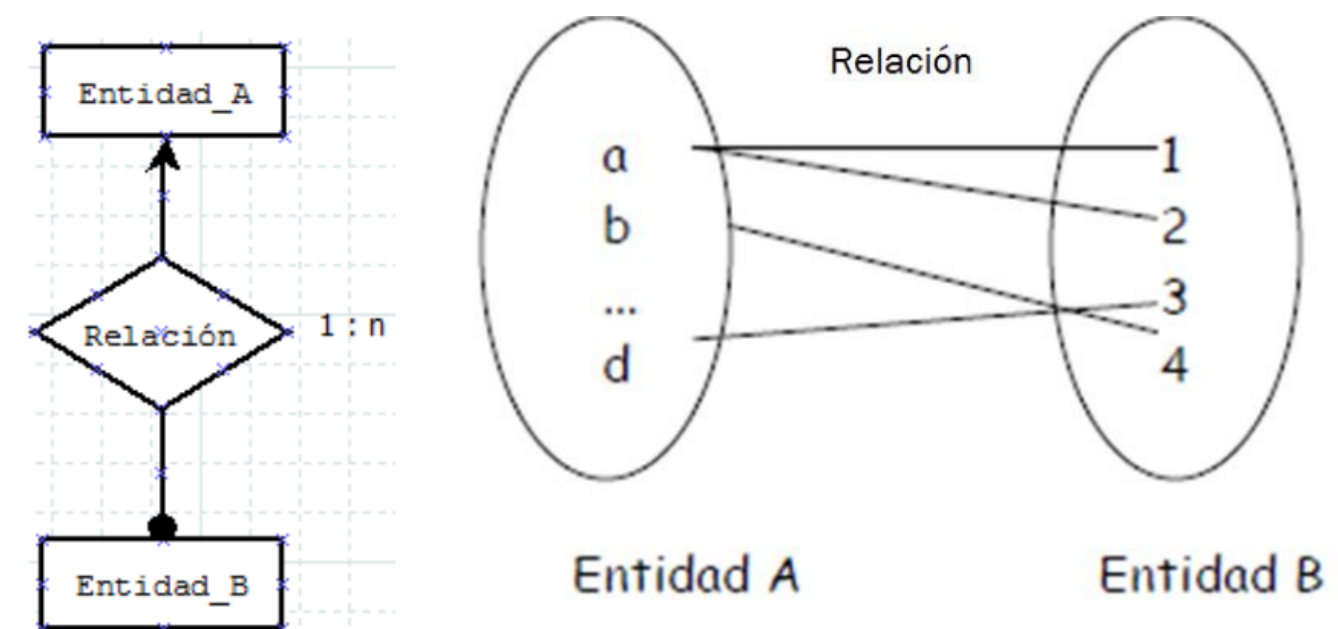
Cardinalidad

Número de ejemplares de una entidad asociadas a otro ejemplar de una entidad o de la misma. Para este punto, existen 3 tipos de cardinalidad.

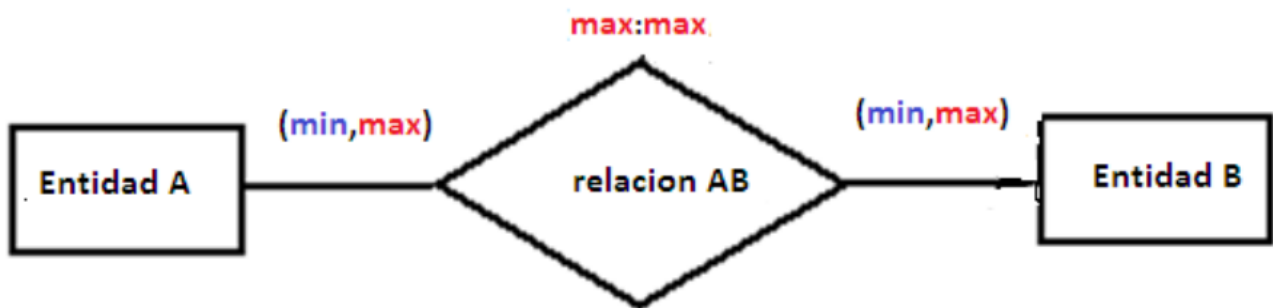
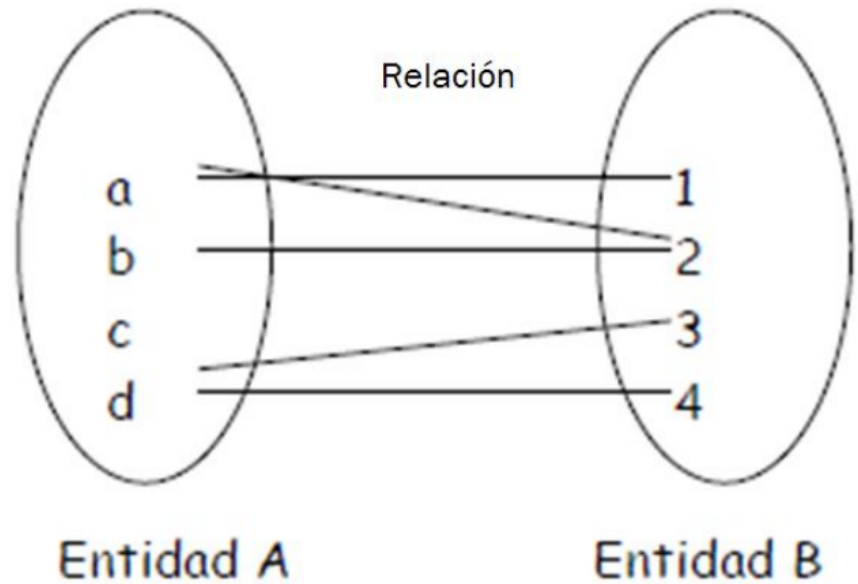
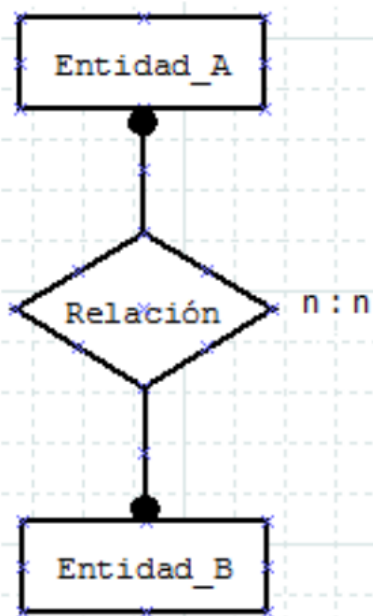
- $(1 : 1)$ ó $(1 \text{ a } 1)$ = Relación Uno a Uno, donde a cada entidad A le corresponde una y solo una entidad B



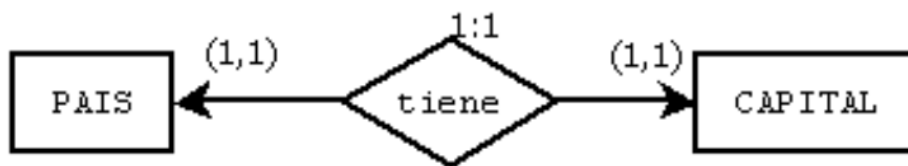
- $(1 : M)$ ó $(1 : *)$ = Relación Uno a Muchos, donde a la entidad A, le corresponde muchos ejemplares de la entidad B



- $(M : M)$ ó $(* : *)$ = Relación Muchos a Muchos, donde muchos ejemplares de la Entidad A se asocian con muchos ejemplares de la entidad B.



Ejemplos: Un país tiene una capital y una capital pertenece a un país



Un cliente tiene uno o más pedidos, pero un pedido sólo pertenece a un cliente



Un avión va a varios aeropuertos, y un aeropuerto recibe varios aviones.



Ejemplos Entidad Relación

1. **Entidades Reportaje y Periodista:** Un reportaje puede ser desarrollado en equipos de hasat 4 Periodistas (1 : M). A su vez, un periodista puede participar en el desarrollo de varios reportajes (1 : M). Se debe guardar el porcentaje de participación del periodista en cada reportaje. **Importante:** Considerar que no todos los periodistas hacen reportajes

A. Tipo de relación: (M:N)

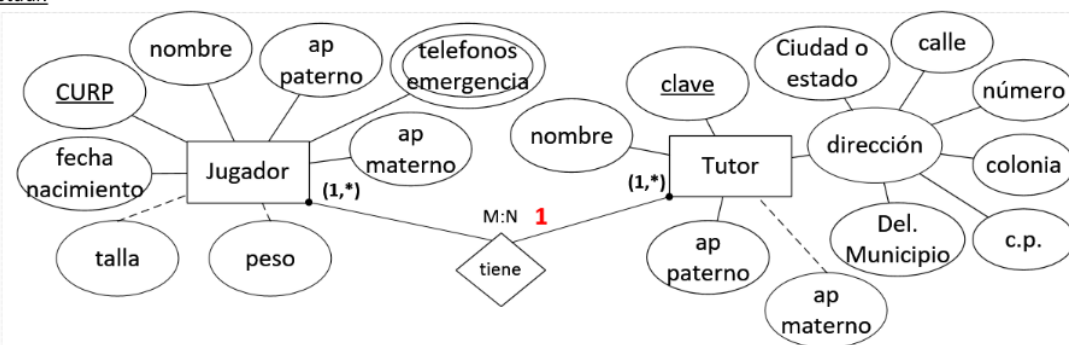
Modelo E/R



Modelo relacional.

2. **Juadores y Tutores:** Se requiere para un equipo de futbol, almacenar la información de los jugadores y tutores, de los jugadores nos interesa conocer el nombre, apellido paterno, telefonos, fecha de nacimiento, apellido materno, talla y peso. Y de los tutores nos interesa conocer, el nombre, clave, ciudad o estado, calle, número, colonia, C.P, Apellido paterno, apellido materno y municipio. *Nota:* Un jugador puede tener uno o mas tutores, pero un tutor puede tener uno, muchos o ningún jugador.

Diseño conceptual:


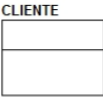


Modelo Físico


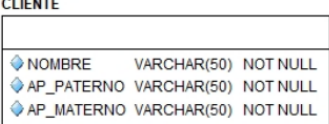
Llamado También Formato Relacional (Relation Format), Formato IE (International Engineering Format), Formato de Martín (Martin's Format), Modelo Pata de gallo (Crow's foot Format), Formato IDEF1X.

Originalmente desarrollado por "The Computer System Laboratory of the National Institute of Standards and Technology" en diciembre del 1993.

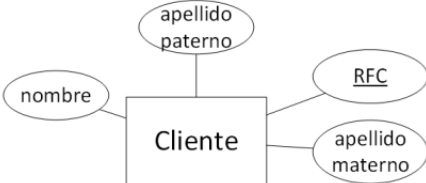
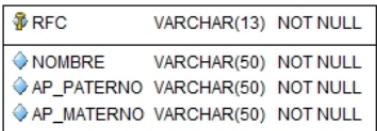
Representación de Entidades.

Diseño conceptual	Diseño lógico Crow's Foot e IDEF1X
	

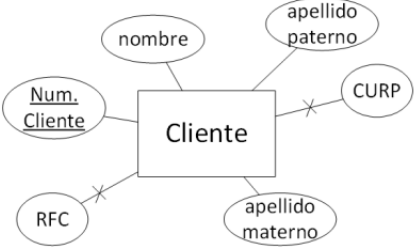

Representación general de atributos.

Diseño conceptual	Diseño lógico Crow's Foot e IDEF1X
	

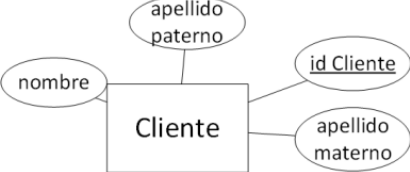
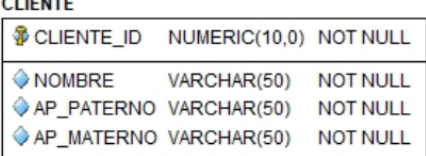
Atributos clave y llaves primarias

Diseño conceptual	Diseño lógico Crow's Foot e IDEF1X
	


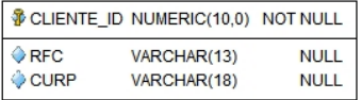
Claves candidatas y llaves primarias candidatas.

Diseño conceptual	Diseño lógico Crow's Foot e IDEF1X
	

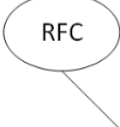
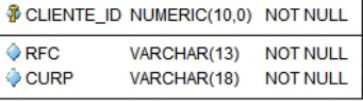
Clave artificial y llave primaria artificial

Diseño conceptual	Diseño lógico Crow's Foot e IDEF1X
	

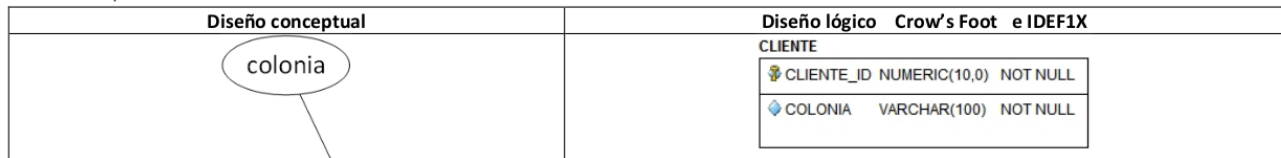
Atributo opcional.

Diseño conceptual	Diseño lógico Crow's Foot e IDEF1X
	

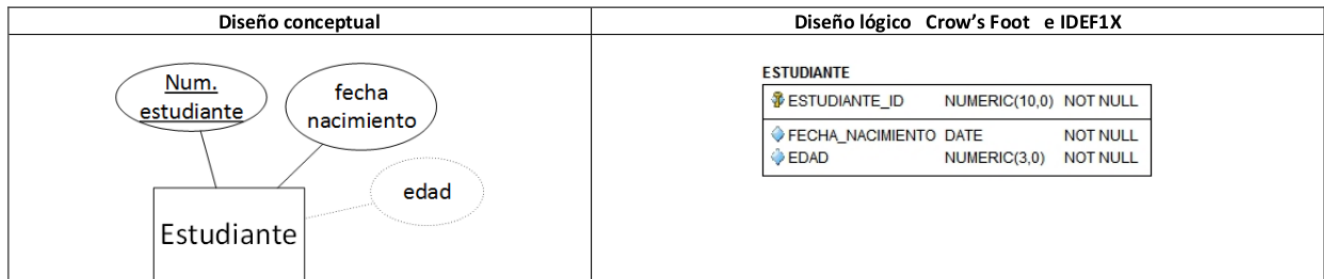
Atributo requerido.

Diseño conceptual	Diseño lógico Crow's Foot e IDEF1X
	

Atributo simple



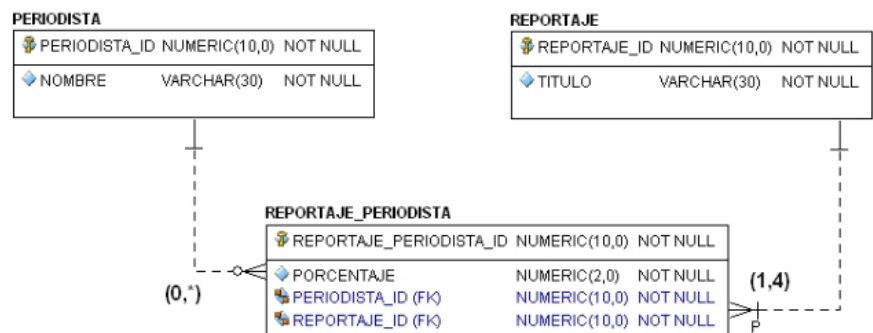
Atributos derivados.



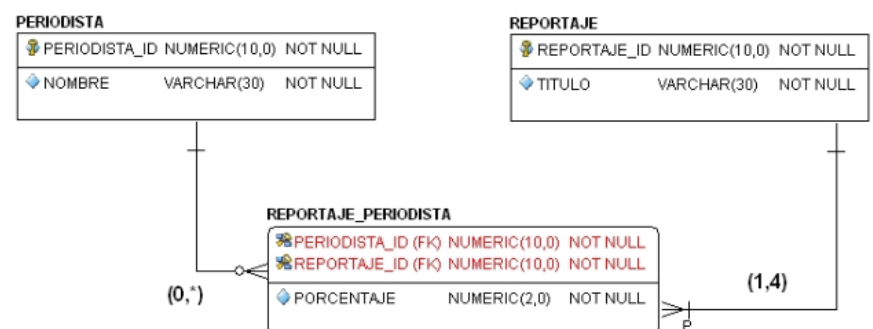
Ejemplos Modelo Físico

Nota: Se realizarán los ejemplos resueltos en el modelo E - R

1. **Entidades Reportaje y Periodista:** Un reportaje puede ser desarrollado en equipos de hasat 4 Periodistas (1 : M). A su vez, un periodista puede participar en el desarrollo de varios reportajes (1 : M). Se debe guardar el porcentaje de participación del periodista en cada reportaje. **Importante:** Considerar que no todos los periodistas hacen reportajes

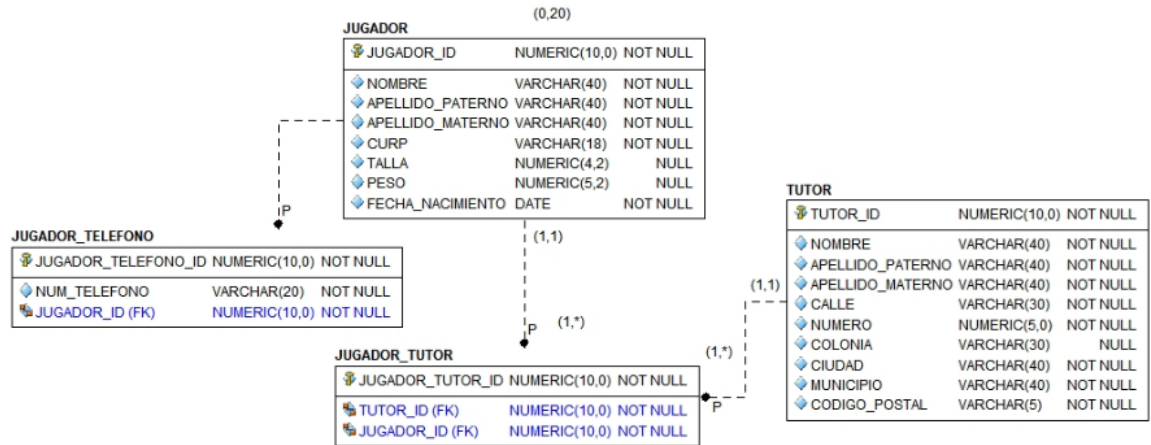


Con PK compuesta:

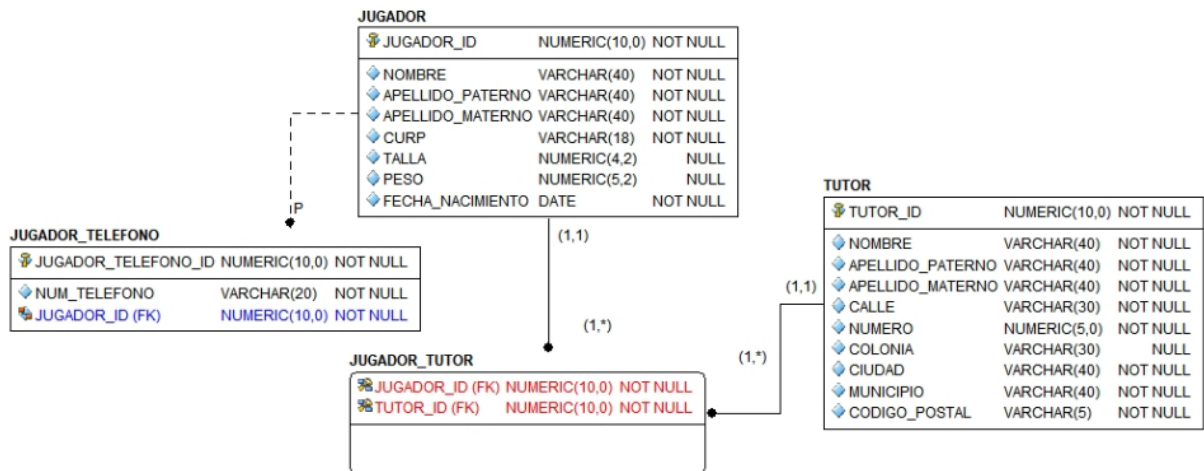


2. **Juadores y Tutores:** Se requiere para un equipo de futbol, almacenar la información de los jugadores y tutores, de los jugadores nos interesa conocer el nombre, apellido paterno, telefonos, fecha de nacimiento, apellido materno, talla y peso. Y de los tutores nos interesa conocer, el nombre, clave, ciudad o estado, calle, número, colonia, C.P, Apellido paterno, apellido materno y municipio. *Nota:* Un jugador puede tener uno o mas tutores, pero un tutor puede tener uno, muchos o ningún jugador.

Diseño lógico



Otra manera de responder el ejemplo anterior, es de la siguiente forma, creando una entidad que solo contenga las llaves de la entidad *Jugador_telefono* y *Tutor*, esta forma se crea la entidad *Jugador_Tutor*, la cual tiene una llave compuesta, la cual es la llave de la entidad *Jugador_telefono* y *Tutor*.



SQL (Codificando la Base)

SQL *Structured Query Language*, actualmente Database Language Query, es un lenguaje de base de datos empleado para:

- Creación y manipulación de las estructuras de una base de datos.
- Administración de los datos
- Ejecución de sentencias complejas diseñadas para transformar los datos almacenados en información útil.
- Diseñado para trabajar con conjunto de datos.
- Portable. Existencia de estándares regulados que permiten el uso del lenguaje relativamente independiente al manejador que se utilice.
- Existencia de extensiones empleadas para evitar los problemas que implica la falta de sentencias de control y estructuras que proporciona un lenguaje procedimental. En este caso cada manejador define sus propias extensiones del SQL:
 - PL-SQL en Oracle

- Transact SQL en SQL Server (Microsoft)
- SQL-PL En DB2 (IBM)

Categorías

- **DDL (Data Definition Language):** Lenguaje de definición de datos. Es el lenguaje encargado de la creación, modificación y eliminación de la estructura de los objetos de la base de datos (tablas, índices, vistas, etc).
- **DML (Data Manipulation Language):** Lenguaje de manipulación de datos. Es el lenguaje que permite realizar las tareas de consulta, modificación y eliminación de los datos almacenados en una base de datos.
- **DCL (Data Control Language):** Lenguaje de control de datos. Es el lenguaje encargado de configurar y establecer el control de acceso a la base de datos. Incluye instrucciones para definir accesos y privilegios a los distintos objetos de la base de datos.
- **DQL (Data Query Language):** Lenguaje de consulta de datos. Algunos autores clasifican a la instrucción SELECT como el único elemento de una cuarta categoría del lenguaje SQL Data Query Language (DQL).
- **Transaction Control:** Control de transacciones. Es el lenguaje empleado para crear, y administrar transacciones aplicadas a un conjunto de sentencias DML principalmente.

Lenguaje	Clausulas básicas
DDL (Data Definition Language)	create alter drop rename truncate comment
DML (Data Manipulation Language)	insert update delete merge
DCL (Data Control Language)	grant revoke
DQL (Data Query Language)	select
Transaction Control	commit rollback savepoint

Creación de Tablas (Sentencia Create)

Ejemplo:

Crear siguiente tabla empleado empleando sintaxis SQL estándar y en Oracle
En SQL estándar:

EMPLEADO	
◆ NOMBRE	VARCHAR(40)
◆ APELLIDO_PATERNO	VARCHAR(40)
◆ APELLIDO_MATERNO	VARCHAR(40)
◆ FECHA_NACIMIENTO	DATE
◆ TIPO_EMPLEADO	CHAR(1)
◆ SUELDO_BASE	DECIMAL(8,2)
◆ FOTO	BLOB
◆ TITULADO	BOOLEAN

```
CREATE TABLE empleado (
    nombre varchar2(40),
    apellido_paterno VARCHAR2(40),
    apellido_materno VARCHAR2(40),
    fecha_nacimiento DATE,
    tipo_empleado CHAR(1),
    sueldo_base number(8,2),
    foto BLOB,
    titulado NUMBER(1,0)
);
```

NOTA: Usualmente se recomienda que las tablas tengan una llave primaria o una clave, sin embargo esto no es un requisito obligatorio.

Ejemplo: Crear la siguiente tabla a partir del modelo relacional

EMPLEADO_SIMPLE			
◆ EMPLEADO_ID	NUMERIC(10,0)	NOT NULL	
◆ NOMBRE	VARCHAR(40)	NOT NULL	

```
CREATE TABLE empleado_simple(
    empleado_id NUMBER(10,0) NOT NULL,
    nombre VARCHAR2(40) NOT NULL
);
```

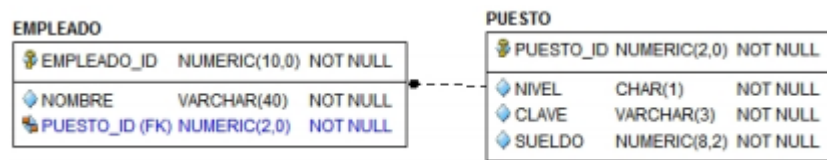
Ejemplo: Crear la siguiente tabla a partir del modelo relacional, tomar en cuenta la creación de la llave primaria

PUESTO			
◆ PUESTO_ID	NUMERIC(2,0)	NOT NULL	
◆ NIVEL	CHAR(1)	NOT NULL	
◆ CLAVE	VARCHAR(3)	NOT NULL	
◆ SUELDO	NUMERIC(8,2)	NOT NULL	

```
CREATE TABLE puesto(
    puesto_id NUMERIC(2,0) PRIMARY KEY,
    nivel CHAR(1) NOT NULL,
    clave VARCHAR2(3) NOT NULL,
    sueldo NUMERIC(8,2) NOT NULL
);
```

```
CREATE TABLE puesto(
    puesto_id NUMERIC(2,0) CONSTRAINT puesto_id_pk PRIMARY KEY,
    nivel CHAR(1) NOT NULL,
    clave VARCHAR2(3) NOT NULL,
    sueldo NUMERIC(8,2) NOT NULL
);
```

Ejemplo: Crear la siguiente tabla a partir del modelo relacional, tomar en cuenta la creación de la llave foránea que una a ambas tablas.

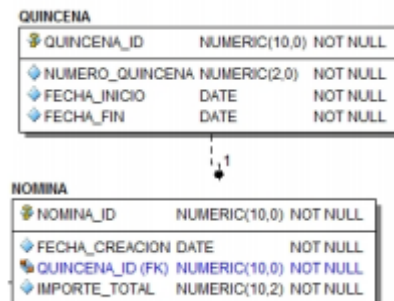


```

CREATE TABLE puesto(
    puesto_id NUMERIC(2, 0) PRIMARY KEY,
    nivel CHAR(1) NOT NULL,
    clave VARCHAR2(3) NOT NULL,
    sueldo NUMERIC(2) NOT NULL
);

CREATE TABLE empleado(
    empleado_id NUMERIC(10,0) PRIMARY KEY,
    nombre VARCHAR2(40) NOT NULL,
    puesto_id NOT NULL CONSTRAINT puesto_id_fk REFERENCES puesto(puesto_id)
);
  
```

Ejemplo: Crear la siguiente tabla a partir del modelo relacional, tomar en cuenta la creación de la llave foránea que una a ambas tablas.



```

CREATE TABLE quincena(
    quincena_id NUMERIC(10,0) CONSTRAINT quincena_pk PRIMARY KEY,
    numero_quincena NUMERIC(2,0) NOT NULL,
    fecha_inicio DATE NOT NULL,
    fecha_fin DATE NOT NULL
);

CREATE TABLE nomina(
    nomina_id NUMERIC(10,0) CONSTRAINT nomina_pk PRIMARY KEY,
    fecha_creacion DATE NOT NULL,
    quincena_id NOT NULL CONSTRAINT
quincena_id_fk REFERENCES
quincena(quincena_id)
);
  
```

Inserción de Datos en SQL (Sentencia Insert, Update y Delete)

Permite agregar, modificar, o eliminar la información almacenada en una base de datos. Esta categoría del SQL está integrada por las siguientes instrucciones:

- **UPDATE:** Nos permite hacer cambios en la tabla con respecto a la información antes contenida, es decir hace una actualización o cambio en los datos de la tabla
- **DELETE:** Como su nombre nos indica, nos permite borrar datos que estén contenidos en la tabla
- **INSERT:** Nos permite Ingresar nuevos valores a la tabla

Sentencia INSERT

Ejemplo: Usar la sentencia *Insert* para llenar con datos la siguiente tabla

EMPLEADO	
NOMBRE	VARCHAR(40)
APELLIDO_PATERO	VARCHAR(40)
APELLIDO_MATERNO	VARCHAR(40)
FECHA_NACIMIENTO	DATE
TIPO_EMPLEADO	CHAR(1)
SUELDO_BASE	DECIMAL(8,2)
FOTO	BLOB
TITULADO	BOOLEAN

```
INSERT INTO empleado VALUES('Edgar Daniel', 'Barcenaz', 'Martínez', to_date('1997/01/15
10:40:00', 'yyyy/mm/dd hh24:mi:ss'), 'C', 1000);
```

```
INSERT INTO empleado VALUES('Berenice', 'Medel', 'Sánchez', to_date('1997/01/10
10:40:00', 'yyyy/mm/dd hh24:mi:ss'), 'C', 1600, NULL, NULL);
```

Nota: La notación presentada anteriormente, es la notación corta, una clara desventaja es que se debe conocer de antemano el orden en el que se definieron los campos al crear la tabla, es propensa a errores ya que en la secuencia no se ve de forma clara a que campo pertenece cada valor.

Adicionalmente, se deben especificar todos los valores de los campos, aunque estos sean nulos, por ejemplo, en el caso del campo conyuge_id, si no se cuenta con el valor, se tiene que escribir la palabra "null" para indicarle al manejador la ausencia de dicho valor.

La forma recomendada es la siguiente:

```
INSERT INTO empleado(nombre, apellido_paterno, apellido_materno, fecha_nacimiento,
tipo_empleado, sueldo_base) VALUES('Isaura', 'Ramírez', 'Salazar', to_date('1997/07/29
10:40:00', 'yyyy/mm/dd hh24:mi:ss'), 'C', 1000);
```

```
INSERT INTO empleado(nombre, apellido_paterno, apellido_materno, fecha_nacimiento,
tipo_empleado, sueldo_base) VALUES('Andrea', 'García', 'Ruiz', to_date('1997/09/08
10:40:00', 'yyyy/mm/dd hh24:mi:ss'), 'C', 700);
```

Sentencia UPDATE

Sentencia DELETE

Bibliografía

Este Manual se realizó con la ayuda de los apuntes de los profesores:

- Jorge A. Rodríguez Campos: jorgerdc@gmail.com
- Lucila Patricia Arellano Mendoza: claseslpam@gmail.com

A los cuales se les agradece por brindar su extraordinario conocimiento.