

Manual para Bases de Datos

Este 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*

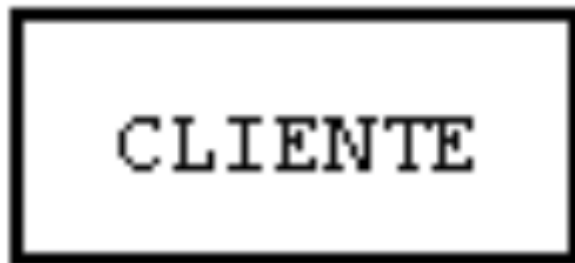
Es el modelo conceptual más utilizado para el diseño de bases de datos. Fue introducido por Peter Chen en 1976. El modelo entidad relación está formado por un conjunto de conceptos que permiten describir la realidad mediante un conjunto de representaciones gráficas y lingüísticas.

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:

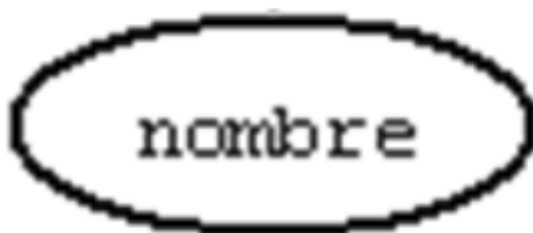
Entidad

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.



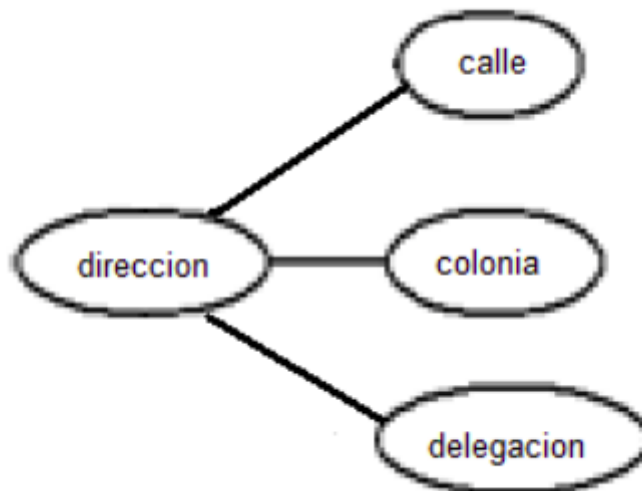
Tipos de Atributos (Opcional)

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

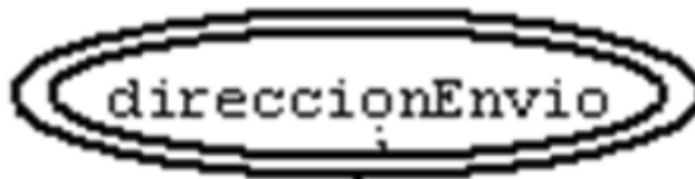
- **Simples:** No se Subdividen



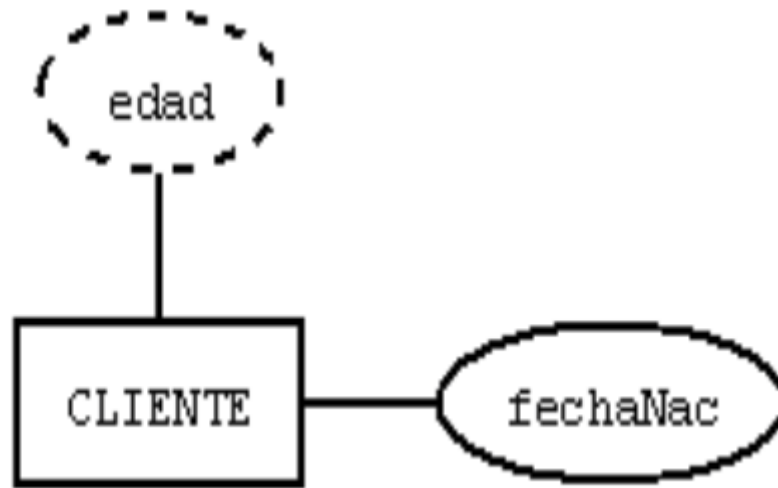
- **Compuestos:** Se dividen en otros atributos



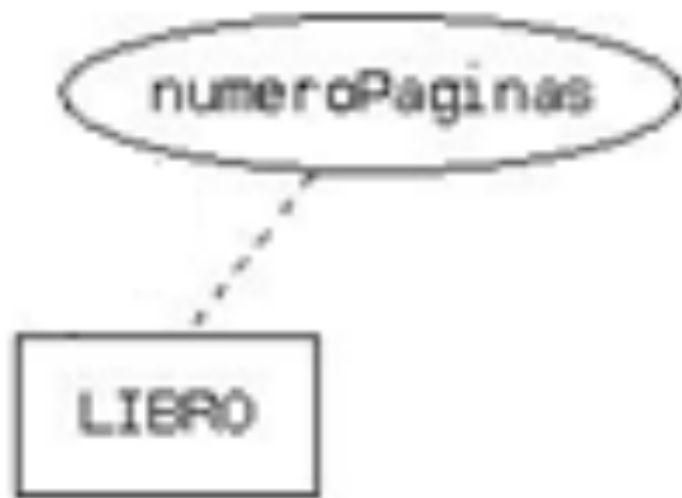
- **Multivalorados:** Tiene un conjunto de valores para una entidad concreta. Se representa con doble elipse.



- **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.



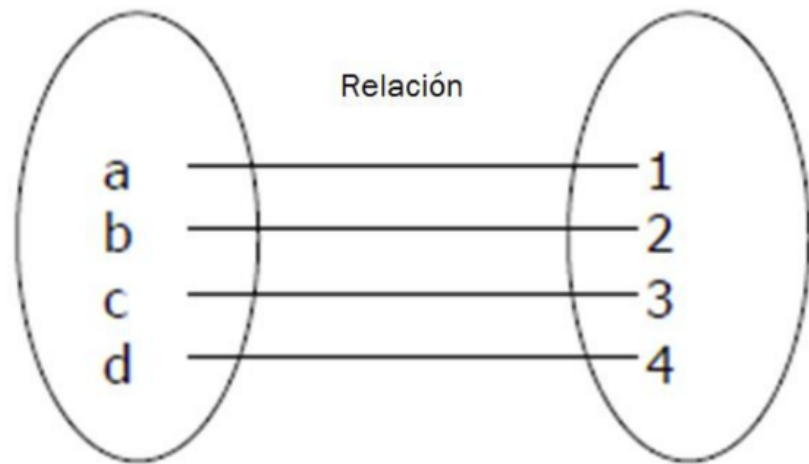
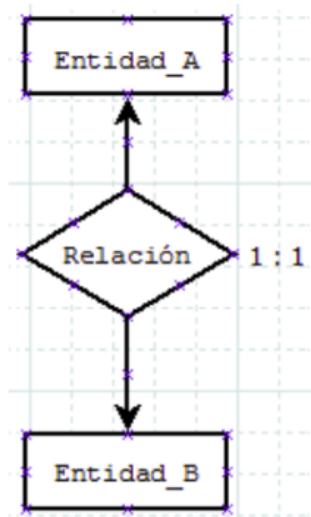
Una relación esta caracterizada por las siguientes características:

- **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*

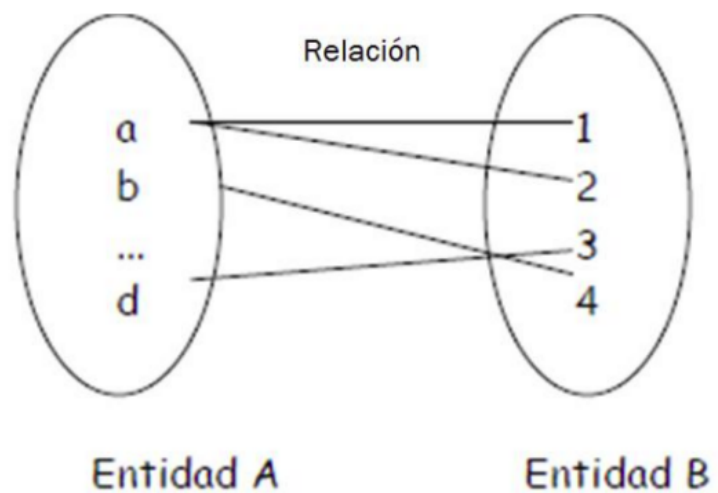
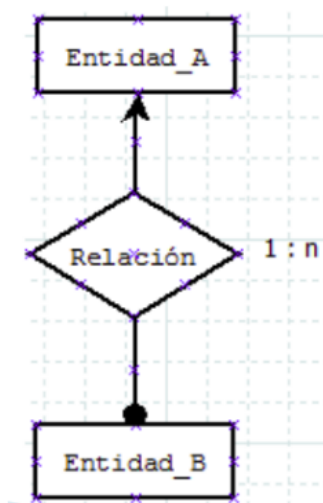
Cardinalidad

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.

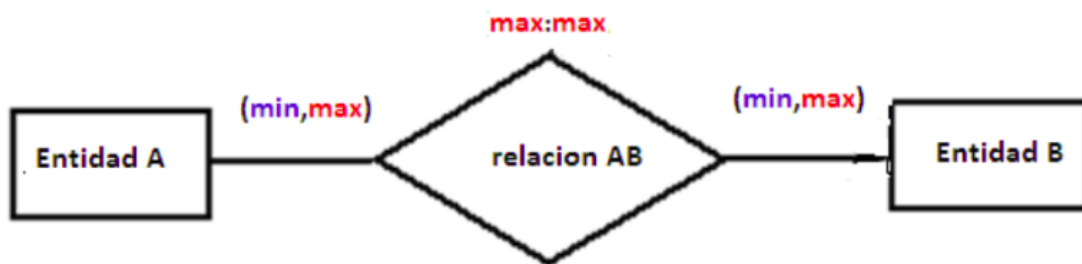
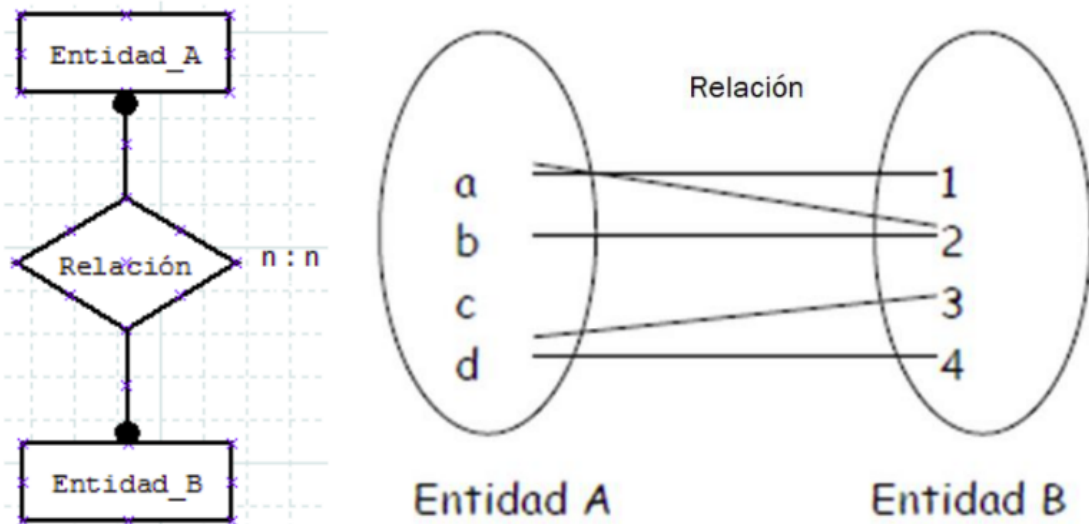
- (1 : 1) ó (1 a 1) = Relación Uno a Uno, donde a cada entidad A le corresponde una y solo una entidad B



- $(1 : M)$ 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 hasta 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.

A. Tipo de relación: (M:N)

Modelo E/R



Modelo relacional.

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
	CLIENTE RFC VARCHAR(13) NOT NULL NOMBRE VARCHAR(50) NOT NULL AP_PATERNO VARCHAR(50) NOT NULL AP_MATERNO VARCHAR(50) NOT NULL

Claves candidatas y llaves primarias candidatas.

Diseño conceptual	Diseño lógico Crow's Foot e IDEF1X
	CLIENTE NUM_CLIENTE NUMERIC(10,0) NOT NULL RFC VARCHAR(13) NOT NULL CURP VARCHAR(18) NOT NULL NUM VARCHAR(30) NOT NULL NOMBRE VARCHAR(50) NOT NULL AP_PATERNO VARCHAR(50) NOT NULL AP_MATERNO VARCHAR(50) NOT NULL

Clave artificial y llave primaria artificial

Diseño conceptual	Diseño lógico Crow's Foot e IDEF1X
	CLIENTE CLIENTE_ID NUMERIC(10,0) NOT NULL NOMBRE VARCHAR(50) NOT NULL AP_PATERNO VARCHAR(50) NOT NULL AP_MATERNO VARCHAR(50) NOT NULL

Atributo opcional.

Diseño conceptual	Diseño lógico Crow's Foot e IDEF1X
	CLIENTE CLIENTE_ID NUMERIC(10,0) NOT NULL RFC VARCHAR(13) NULL CURP VARCHAR(18) NULL

Atributo requerido.

Diseño conceptual	Diseño lógico Crow's Foot e IDEF1X
	CLIENTE CLIENTE_ID NUMERIC(10,0) NOT NULL RFC VARCHAR(13) NOT NULL CURP VARCHAR(18) NOT NULL

Atributo simple

Diseño conceptual	Diseño lógico Crow's Foot e IDEF1X
	CLIENTE CLIENTE_ID NUMERIC(10,0) NOT NULL COLONIA VARCHAR(100) NOT NULL

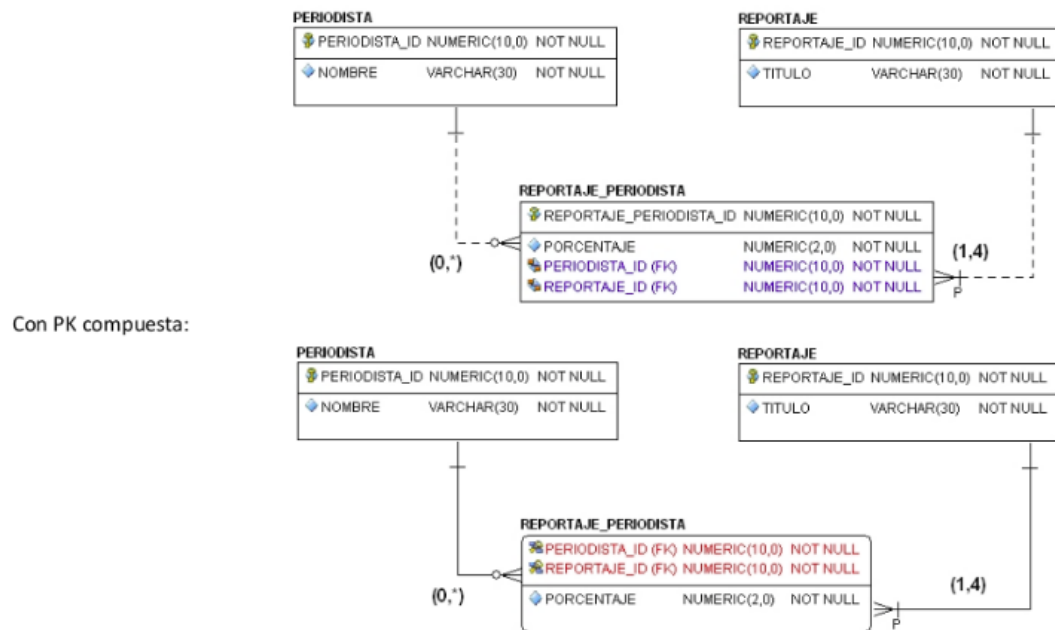
Atributos derivados.

Diseño conceptual	Diseño lógico Crow's Foot e IDEF1X
	ESTUDIANTE ESTUDIANTE_ID NUMERIC(10,0) NOT NULL FECHA_NACIMIENTO DATE NOT NULL EDAD NUMERIC(3,0) NOT NULL

Ejemplos Modelo Físico

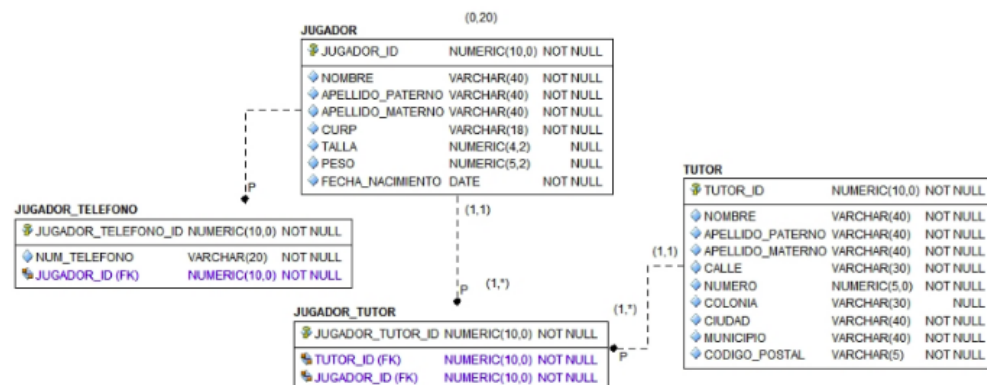
Nota: Se realizarán los ejemplos resueltos en el modelo E - R

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

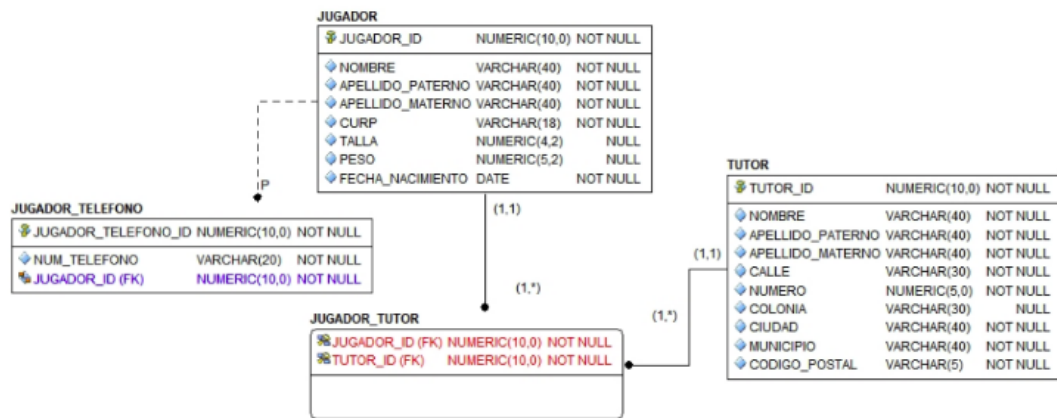


2. **Juadores y Tutores:** Se requiere para un equipo de futbol, almacenar la información de los jugadores y tutores, de los jugadores nos interesa conocer el nombre, apellido paterno, telefonos, fecha de nacimiento, apellido materno, talla y peso. Y de los tutores nos interesa conocer, el nombre, clave, 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*, este 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 tenngan 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

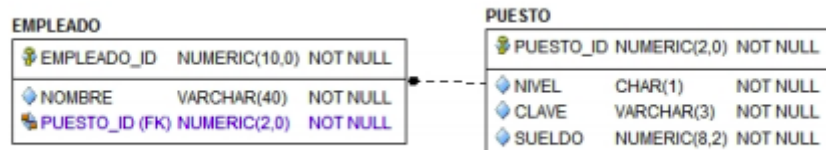
PUESTO

	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
);
```

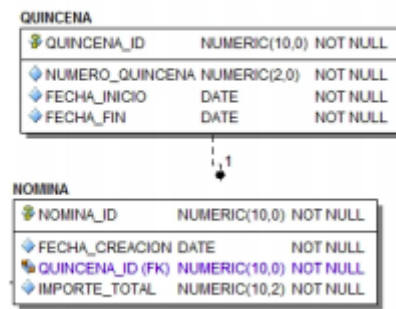
Ejemplo: Crear la siguiente tabla a partir del modelo relacional, tomar en cuenta la creación de la llave foránea que una a ambas tablas.



```
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_PATERNO	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', 'Barcenas', '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

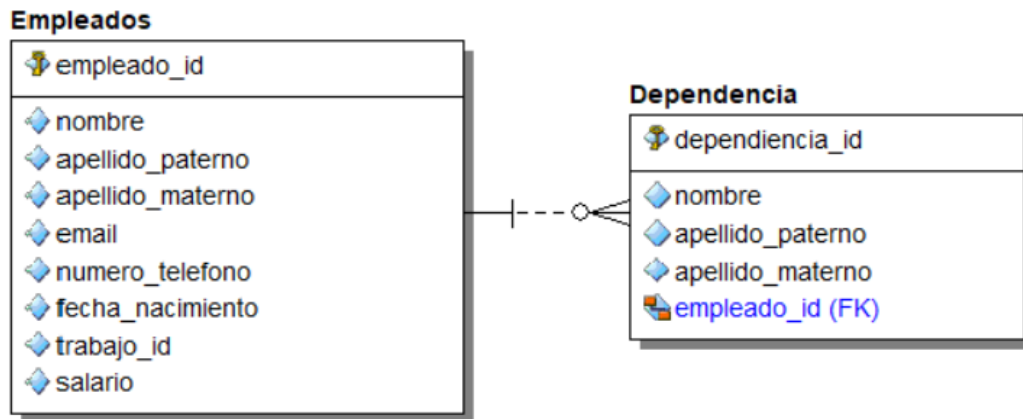
La sentencia *update* permite actualizar los valores de uno o más campos asociados a uno o mas registros de una tabla

Sintaxis Básica

```
UPDATE table_name  
SET column = value1  
    column2 = value2  
WHERE = condition
```

- Se debe indicar la tabla que se desea actualizar con la cláusula *UPDATE*
- Se debe especificar las columnas que se desea modificar con la cláusula *SET*. **NOTA:** Las columnas que no figuren en la cláusula *SET* conservarán sus valores originales
- Se debe especificar las filas a actualizar en la cláusula *WHERE*

Ejemplo: Tome como referencia el siguiente modelo relacional



Suponga que la identificación del empleado 192 Sarah Bell Cambió su apellido de *Bell* a *López* y usted necesita actualizar su registro en la tabla *Empleados*. En este caso se debe usar la instrucción *UPDATE* y su syntaxis es la siguiente:

```
UPDATE empleados
SET
    last_name = 'Lopez'
WHERE
    employee_id = 192;
```

Ejemplo: El empleado *Angela Ramirez Luna* se ha divorciado, se requiere reflejar los cambios en la base de datos. Adicionalmente se requiere corregir su fecha de nacimiento al 11-01-1980

```
UPDATE empleado SET conyuge_empleado_id = null,
    fecha_nacimiento = to_date('11-01-1980', 'dd-mm-yyyy')
WHERE nombre='angela'
    AND apellido_paterno='ramirez'
    AND apellido_materno='luna';
```

Usando Subconsulta

A veces es necesario actualizar los valores buscando algo en específico, en estos casos se puede usar una búsqueda más avanzada, en el caso de SQL a este procedimiento de búsqueda sobre búsqueda se le conoce como *SubConsulta* y se puede usar en una actualización de datos o también llamada *Update*.

Ejemplo: Actualizar el puesto del empleado ANGEL JUAREZ AGUIRRE de director general (DG) a jefe de departamento (JD)

```
update empleado set puesto_id =
    (select puesto_id
     from puesto
      where clave_puesto = 'jd')
where nombre='angel'
    and apellido_paterno='juarez'
    and apellido_materno='aguirre';
```

Sentencia DELETE

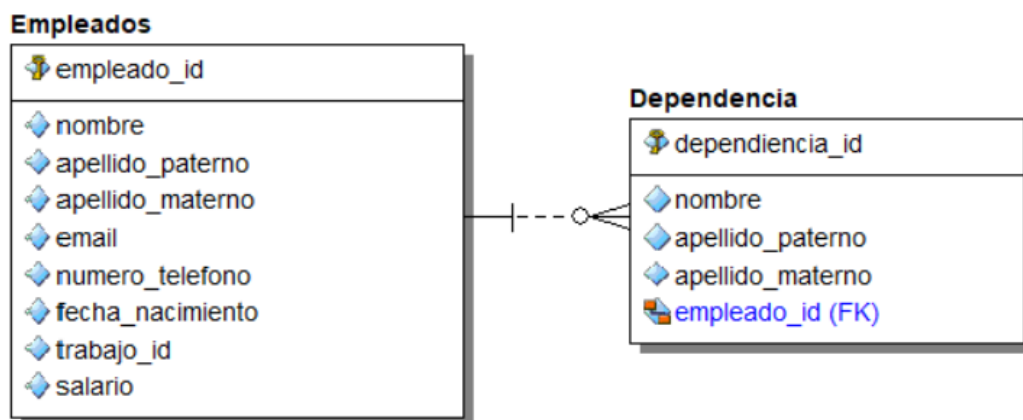
Para eliminar una o más filas de una tabla se usa la instrucción *DELETE*. La sintaxis general para la instrucción *DELETE* es la siguiente:

```
DELETE
FROM
    table_name
where
    condition;
```

- Se debe proporcionar el nombre de la tabla de donde se desea remover los registros
- Se debe especificar la condición en la cláusula *WHERE*, para identificar las filas que deben eliminarse
- La Instrucción *DELETE* no devuelve un conjunto de resultados, sin embargo devuelve el número de filas eliminadas.

NOTA: Tener cuidado con el uso de *DELETE*, ya que, si no se especifician las condiciones correctas en la cláusula *WHERE*, pueden eliminarse registros no deseados o inclusive el contenido completo de la tabla

Ejemplo: Teniendo el siguiente modelo relacional, borrar de la tabla dependientes, al usuario que tenga el ID = 16



```
DELETE FROM dependets
WHERE
    dependet_id = 16;
```

Usando Subconsulta

Al igual que en la sentencia *UPDATE*, en *DELETE*, también se puede usar una Subconsulta para hacer mas preciso la búsqueda del dato que queremos borrar.

Ejemplo: Eliminar todos los registros asociados a los hijos del empleado *Juan Martinez López*.


```
delete from hijo_empleado
where empleado_id = (
    select empleado_id
    from empleado
    where nombre='juan'
        and apellido_paterno='martinez'
        and apellido_materno='lopez');
```

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.