



Ministerio de Producción
Presidencia de la Nación

Ministerio de Educación y Deportes

Subsecretaría de Servicios Tecnológicos y Productivos



Programa
111
mil
VOS PODÉS
SER UNO.

MODULO DE BASE DE
DATOS



Ministerio de Producción
Presidencia de la Nación

Ministerio de Educación y Deportes

Subsecretaría de Servicios Tecnológicos y Productivos



Programa
111
mil
VOS PODÉS
SER UNO.

Modelo de Entidades y
Relaciones



Ministerio de
Educación y Deportes
Presidencia de la Nación



Ministerio de Producción
Presidencia de la Nación

Agenda



Introducción

La información valiosa debe tener una representación simbólica adecuada e inteligible para que pueda ser *comunicada*.

¿Qué es un dato?

- Una pieza atómica que se utilizará para construir el concepto de información.
- Una pieza atómica de dato es un valor, que debe ser interpretado como una propiedad perteneciente a un hecho o cosa del mundo real. La interpretación es necesaria para ubicar el **dato** en **contexto**.



Modelo de datos

- **Modelado de Datos:** organizar los datos para que representen una situación del mundo real con la mayor fidelidad posible, con el objetivo de poder manejarlos computacionalmente.
- **No es posible tener un conocimiento completo del mundo real:** centrar la atención en la información relevante, ocultando o ignorando otra que **NO** sea relevante o que resulte inadecuada para el propósito requerido.
- Comprender las características de la información relevante **para determinar cómo van a ser organizados y procesados los datos:** pueden ser descriptas a través de enunciaciones generales., por ej. lenguaje natural.
- Un conjunto formal y consistente de tales enunciaciones define un **modelo conceptual de datos.**



- ❖ La correcta estructura de la base de datos se obtiene como resultado del **modelado de los datos**
- ❖ **Modelo de datos:** Es una colección de herramientas conceptuales para describir los datos, las relaciones entre ellos, la semántica que tienen en el Universo del Discurso y las restricciones referidas a sus valores
- ❖ El **modelo de datos relacional** es uno de los paradigmas más populares, subyacente a la mayoría de las BD actuales: considera los datos bajo la forma de conjuntos, que se representa habitualmente de forma tabular
- ❖ El **modelo de Entidades y Relaciones (Extendido - MERE)** brinda una representación gráfica sencilla para ver los datos, relaciones y restricciones, de acuerdo al modelo relacional.
 - Representa **Entidades, Atributos y Relaciones**
 - Reglas de transformación: esquema lógico según modelo relacional



¿Cómo construir el modelo de datos?

Etapas para encontrar una forma eficaz de representar la información en el mundo computacional:

- **Identificación de los datos de la realidad:** actores, recursos, objetos, etc. del mundo real de los cuales interesa guardar información
- **Identificación de relaciones entre datos:** detección de los vinculos significativos que se dan entre los elementos de las etapas anteriores
- **Abstracción de datos y relaciones:** representación simbólica sólo de los elementos detectados en las etapas anteriores



Entidades

Objeto real o abstracto que existe en la realidad y acerca del cual se desea almacenar información.

- Cualquier objeto (real o abstracto) que existe en la realidad y acerca del cual queremos almacenar información en la base de datos
- Algo con realidad objetiva que existe o puede ser pensado
- Una persona, lugar, cosa, concepto o suceso, real o abstracto, de interés para la empresa
- Objetos (hechos, cosas, personas,...) que tienen propiedades en común y una existencia autónoma



El elemento básico representado por el modelo entidad relación es una **entidad**, que es una cosa del mundo real con una existencia independiente.

- Una **entidad** puede ser un elemento con una existencia física (por ejemplo, una persona en particular, un coche, una casa o un empleado) o puede ser un elemento con una existencia conceptual (por ejemplo, una venta, un trabajo o un curso universitario).
- Cada entidad tiene atributos (propiedades particulares que la describen).



Una **entidad** es una abstracción de un conjunto de cosas del mundo real tal que:

- Las cosas de ese conjunto tienen las mismas características o comportamiento.
- Las cosas de ese conjunto están sujetas y conformes a las mismas reglas.

Las **entidades** que se tendrán en cuenta al modelar un sistema son aquellas que representan "cosas" de las que el sistema necesita almacenar ciertos datos.

Ejemplo: El conjunto: {ruleman 8705, tornillo 5 mm, filtro de aire, etc.} Forma la **entidad** → **REPUESTO**



Ejemplo: Todos los Clientes representan la Entidad **CLIENTE**

- Todas las entidades de un conjunto tienen el mismo conjunto de atributos que interesa modelar. Todos los ejemplares de CLIENTE, tendrán Identificador de cliente, CUIT, Nombre, Apellido y demás datos descriptores
- Cada conjunto entidad necesita un **identificador**: atributo o conjunto de ellos que permita identificar a cada uno de los ejemplares que componen el conjunto entidad. La Entidad CLIENTE tiene como atributo identificador de los ejemplares a identificador de cliente.



Identificación de Entidades

- Cosas tangibles: Artículo, Repuesto, Rodado.
- Roles desempeñados por personas u organizaciones: Cliente, Proveedor, Personal.
- Incidentes: Usado para representar la ocurrencia de un hecho (en un sistema de una compañía de seguros: Siniestros; en una empresa de transporte: Viajes).
- Interacciones: Representan alguna transacción (Compra, Pedido, Venta, Pago)

Es importante una **buena elección del nombre** dado a una entidad para la legibilidad y el entendimiento del modelo de datos.



Atributos

Un atributo es una abstracción que identifica características, propiedades que posee una entidad. Los atributos de una entidad deben ser:

- **Completos:** capturar toda la información que interesa del objeto, desde el punto de vista del sistema.
- **Plenamente elaborados:** cada atributo capture un aspecto separado de la entidad.
- **Mutuamente independientes:** cada atributo debe tomar un valor independientemente de los valores asumidos por otros atributos.



Clasificación de Atributos

- ❖ **Atributos identificadores:** el o los atributos que permiten identificar únicamente a una instancia de una entidad.
Constituyen la "clave primaria".
- ❖ **Atributos descriptivos:** son las características intrínsecas de cada instancia de la entidad; como lo dice su nombre, describen a la entidad, representan sus propiedades.
- ❖ **Atributos referenciales:** son atributos que sirven para relacionar entidades entre sí. Se denominan **REFERENCIALES** ya que hacen referencia al **ATRIBUTO IDENTIFICADOR** de la entidad con que se relacionan.



Atributos: Ejemplo

Entidad: **PELICULA**

Atributos: nombre, título original, año de estreno, disponible, duración, fecha de ingreso.

Pelicula
anio_estreno
disponible
duracion
fecha_ingreso
nombre
titulo_original



Instancias de Entidad

Así podemos tener como ejemplo de entidades **PELICULA**, a dos instancias de película, con los siguientes atributos:

Película 1: {2013, true, 143, 11/07/2013, "El Gran Gatsby", "The Great Gatsby"}

Película 2: {2014, true, 122, 01/08/2014, "Relatos Salvajes", "Relatos Salvajes"}

A cada película de la entidad **PELICULA**, se las denomina genéricamente **instancias** de dicha entidad.



Atributo Identificador Único

Se denomina identificador a uno o más atributos que identifican unívocamente cada instancia de una entidad; es conocido también como "clave candidata".

Para elegir el atributo identificador debemos tener en cuenta dos reglas:

- Que la clave sea **mínima**: Es decir elegir la alternativa en la que se necesiten menos atributos para conformar la clave.
- Elegir el atributo **más significativo** dentro del dominio del problema que se está modelando.



El atributo A, o el conjunto de atributos, de una entidad, es un posible atributo identificador si y solo si satisface dos propiedades:

- **Unicidad:** en cualquier momento dado no existen 2 instancias con el mismo valor de A.
- **Minimidad:** Si A es compuesto (es decir el atributo identificador está formado por más de un atributo) no será posible eliminar ningún componente de A sin destruir la propiedad de unicidad.

Toda **entidad** tiene por lo menos un atributo como posible atributo identificador. El o los atributos identificadores se señalan con el símbolo "@"(arroba), o de lo contrario con la sigla PK (clave primaria)



Ejemplo de Atributo Identificador

Pelicula
anio_estreno
disponible
duracion
fecha_ingreso
nombre
titulo_original

Sin Atributo Identificador
Atributo Identificador

Pelicula
@id_pelicula
anio_estreno
disponible
duracion
fecha_ingreso
nombre
titulo_original

Con



Representación de una Entidad con su atributo identificador

Película 1: {1, 2013, true, 143, 11/07/2013, "El Gran Gatsby", "The Great Gatsby"}

Película 2: {2, 2014, true, 122, 01/08/2014, "Relatos Salvajes", "Relatos Salvajes"}

Este atributo agregado **@id_película**, no es más que un número identificador que crece secuencialmente a medida que se agregan nuevas películas: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,...n.



Atributo Referencial

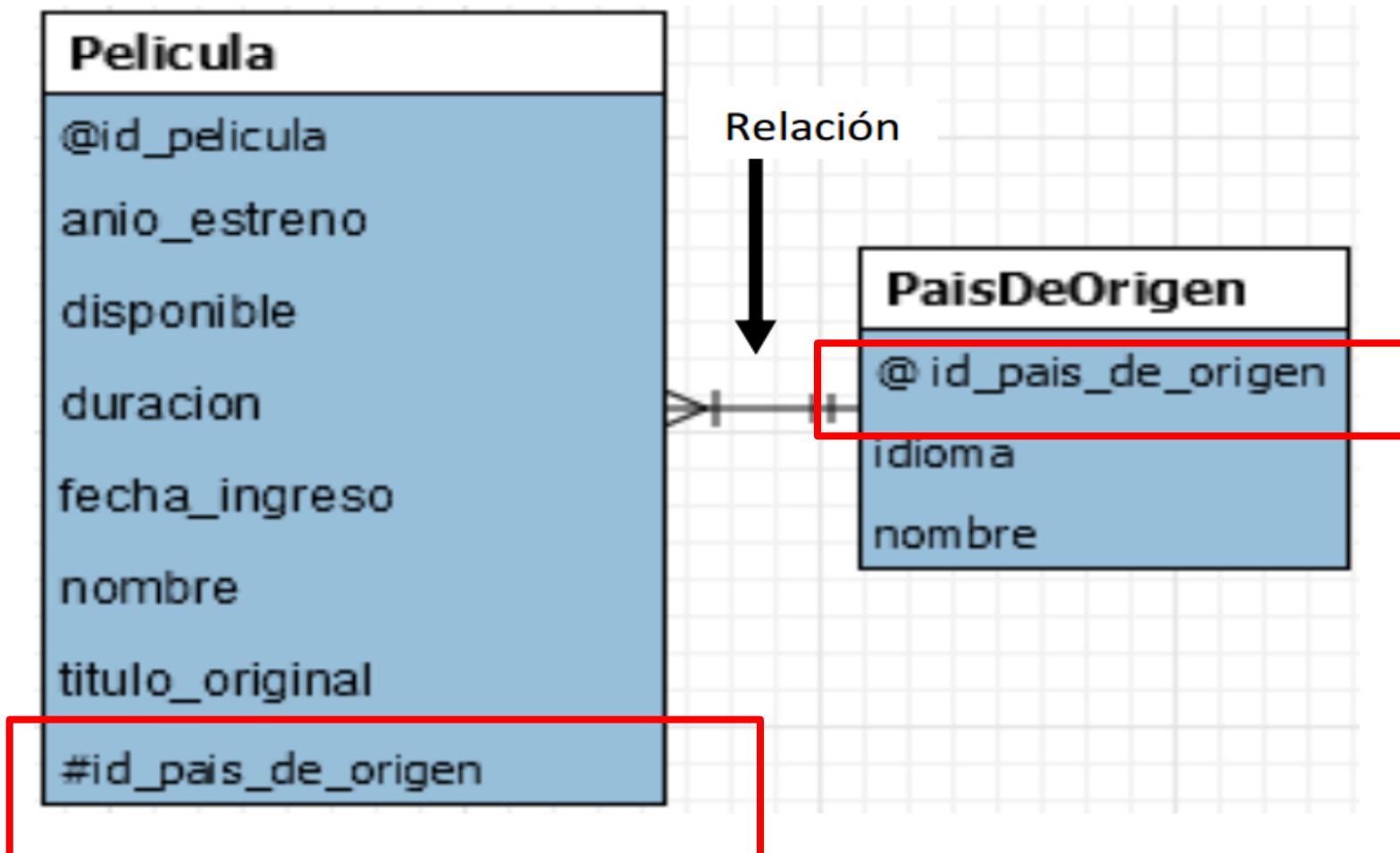
Un **atributo referencial** se utiliza para poder establecer relaciones entre diferentes entidades de un **modelo entidad-relación**. Se dice que un atributo j, ó un conjunto de atributos, de una entidad B es un atributo referencial si y sólo si satisface dos propiedades:

- Cada valor j es nulo del todo o no nulo del todo. – En caso de ser un atributo compuesto, formado por más de un atributo.
- Existe una entidad A con atributo identificador j tal que:
 - Cada valor no nulo de j es en la entidad B idéntico al valor j en alguna instancia de la entidad A. Es decir que si en B el atributo j tiene valor es porque existe ese mismo valor de j en la entidad A.

Ejemplo: **PELICULA** y **PAIS DE ORIGEN**. Se utiliza el símbolo “#” (numeral) para señalar que un **atributo es referencial**, o de lo contrario con la sigla FK (clave foránea).



Representación de Entidades y su forma de relacionarse, con su atributo referencial





Si tuviéramos las siguientes entidades en PaísDeOrigen:

País de Origen 1: {1, “Español”, Argentina}

País de Origen 1: {2, “Inglés”, Estados Unidos}

País de Origen 1: {3, “Francés”, Francia}

Entonces la entidad película, tendría en el **atributo referencial id_pais_de_origen**, el valor 1, que referencia a la instancia Argentina de la **entidad PaisDeOrigen**, como se ve a continuación:

Película 2: { 1, 2014, true, 122, 01/08/2014, "Relatos Salvajes", "Relatos Salvajes",1}

Reglas de Integridad



Reglas de Integridad: Integridad de Entidades

Ningún componente del atributo identificador en una entidad aceptará NULOS (nulo se considera que es inexistente, es decir, ausencia de valor).

Ejemplo:

Película: {**NULL**, 2014, true, 122, 01/08/2014, "Relatos Salvajes", "Relatos Salvajes",1} -> **Mal**

Película: { **2**, 2014, true, 122, 01/08/2014, "Relatos Salvajes", "Relatos Salvajes",1} -->**Bien**



Reglas de Integridad: Integridad Referencial

Un modelo de datos no debe contener valores en sus atributos referenciales para los cuales no exista un valor concordante en el (ó los) atributos identificadores en la entidad objetivo pertinente.

Ejemplo:

Película: { 2, 2014, true, 122, 01/08/2014, "Relatos Salvajes",
"Relatos Salvajes", 4 }

No existe en la tabla PaisDeOrigen una instancia con
id_pais_de_origen = 4.

Película: { 2, 2014, true, 122, 01/08/2014, "Relatos Salvajes",
"Relatos Salvajes", 1 }



Relaciones

- ★ Asociación o vinculación entre **entidades**.
- ★ Una relación es la abstracción de un conjunto de asociaciones que existen entre las instancias de dos **entidades**, por ejemplo, existe una relación entre Película y PaísDeOrigen
- ★ Las relaciones tienen sentido bidireccional.
- ★ Las relaciones existen ya que las **entidades** representan aspectos del mundo real y en este mundo los componentes no están aislados, sino que se relacionan entre sí; es por esto que es necesario que existan las relaciones entre las **entidades**.



Cardinalidad y Opcionalidad de las relaciones

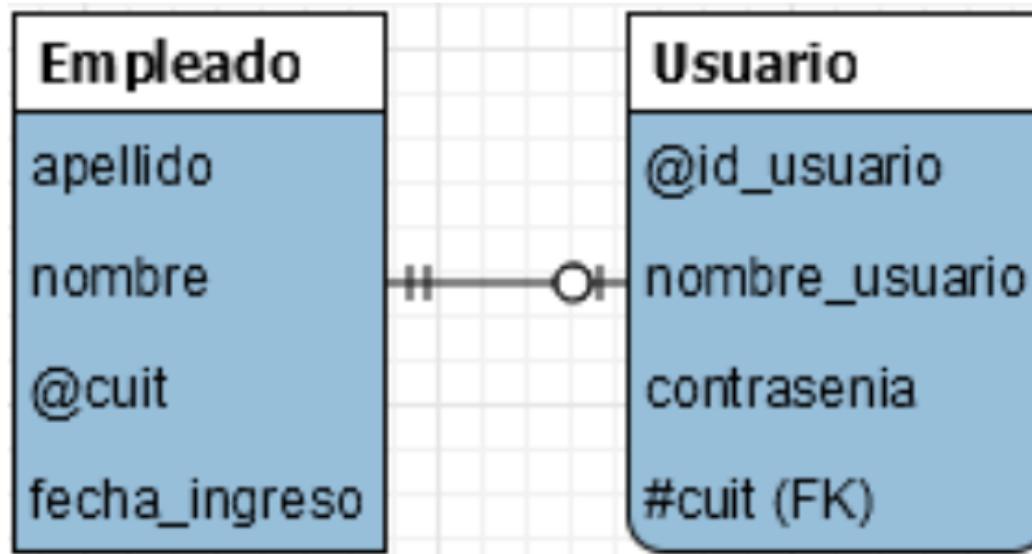
Cardinalidad: Indica para una instancia de una entidad A con cuántas instancias de la entidad B se relaciona. Las posibilidades son: 0, 1 o muchos.

Opcionalidad: Indica para una instancia de una entidad A, si la relación con instancias de la entidad B, es opcional u obligatoria. Las posibilidades son: 0 o 1.



Caso Uno a Uno

Un empleado puede tener o no un usuario y si ese usuario existe, es para un único empleado.



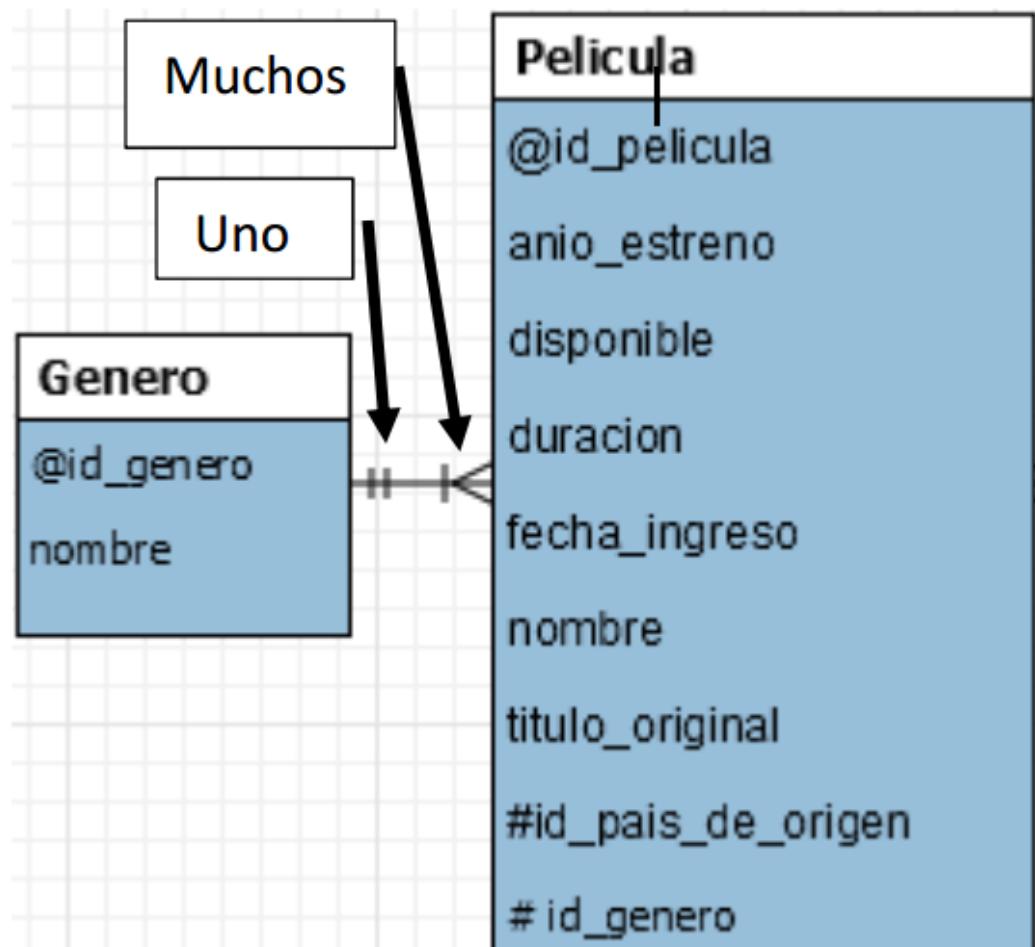
Un empleado puede tener opcionalmente un único usuario o no tener ningún usuario asociado, un usuario está asociado de manera obligatoria a un único empleado.



Caso Uno a Muchos

Una película tiene un único género, pero un género como “Drama” puede estar asignado a muchas películas:

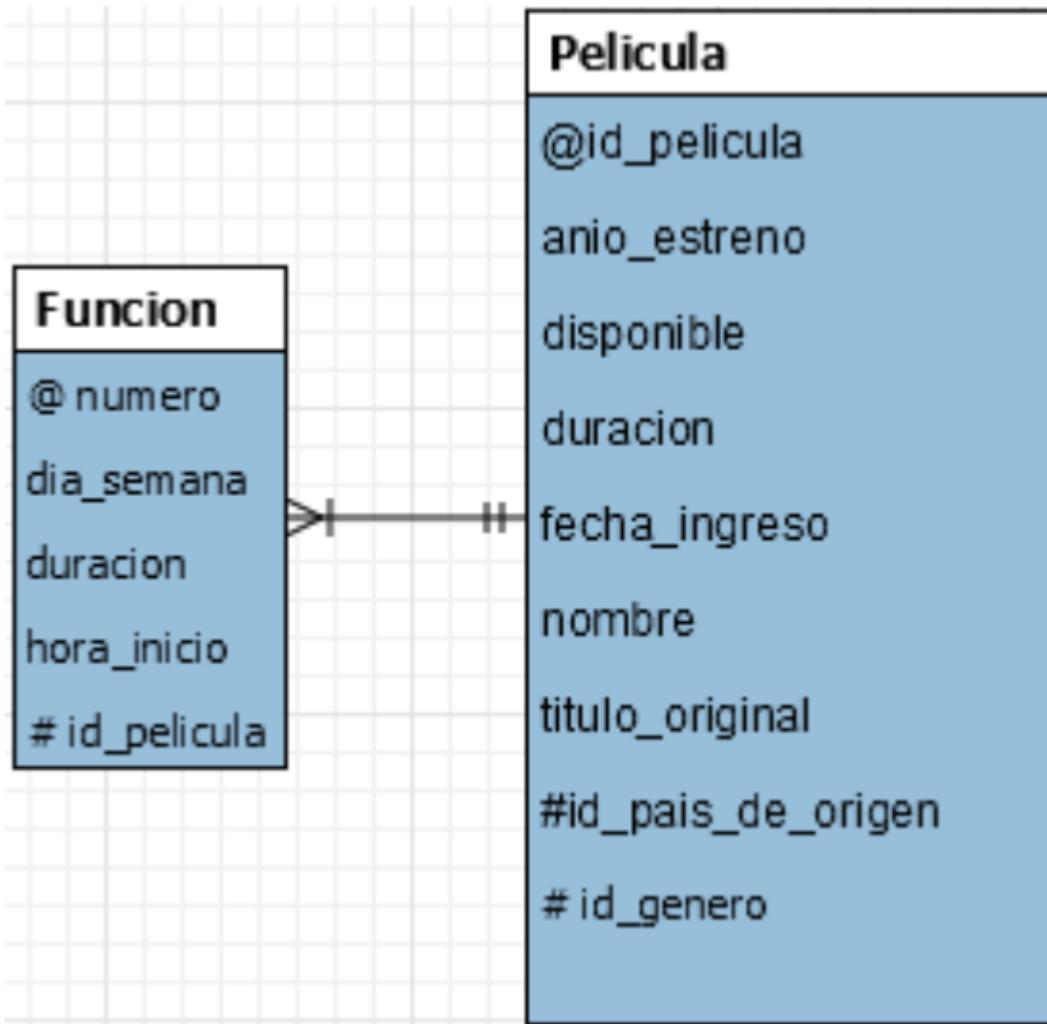
“Un género puede estar asignado a muchas películas, pero una película tiene un único género”





Caso Muchos a Uno

Una película tiene muchas funciones asignadas, pero una función es para una única película.





Valores atómicos en cada celda

Caso 1: Relación entre Función y Película, con Función referenciando a Película. En este caso la función tiene una única película asociada porque se proyecta una película por función.

Caso 2: Relación entre Función y Película, con Película referenciando a Función. En este caso como la película puede proyectarse en muchas funciones, deberíamos poner en la columna identificada como **#id-funcion**, más de un atributo referencias, rompiendo la propiedad de atomicidad de los atributos:



Caso 1

ENTIDAD FUNCIÓN

@numero	dia_semana	duración	Hora_inico	#id_pelicula
1	3	150	22:30	2
2	4	150	18:15	2

ENTIDAD PELICULA

@id_pelicula	Nombre	duración	#id_pais_de_origen	#id_genero
2	Relatos Salvajes	122	1	1



Caso 2

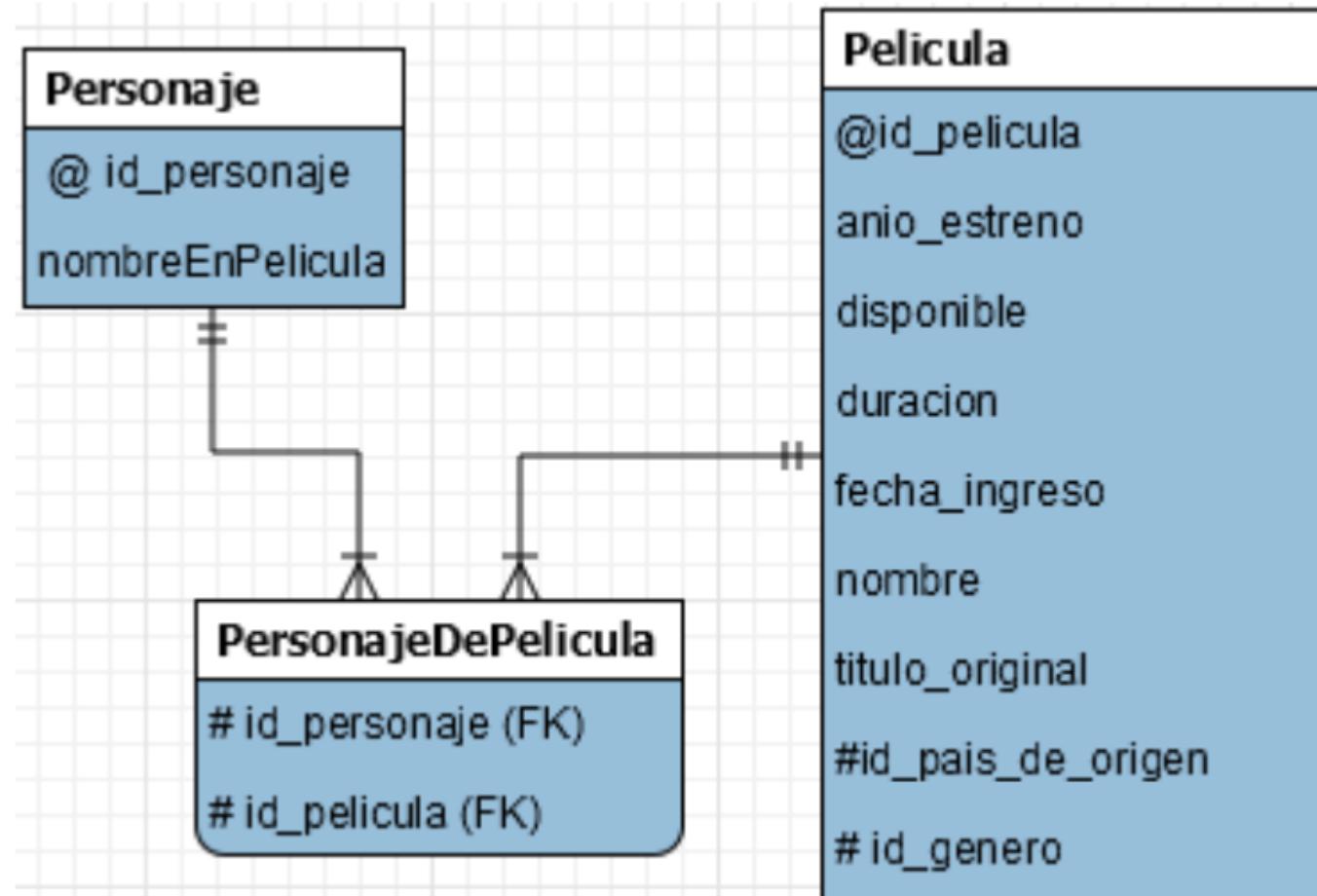
ENTIDAD FUNCIÓN			
@numero	dia_semana	duración	Hora_inico
1	3	150	22:30
2	4	150	18:15

ENTIDAD PELICULA					X
@id_pelicula	Nombre	duración	#id_pais_de_orginen	#id_genero	#id_funcion
2	Relatos Salvajes	122	1	1	1,2



Casos Muchos a Muchos

Un personaje puede pertenecer a varias películas, por ejemplo: Harry Potter y la piedra filosofal, Harry Potter y la cámara secreta, Harry Potter y el prisionero de Azkaban... y una película puede tener varios personajes.





Entidades de Personajes:

{1, “Harry Potter”}

{2, “Ron Weasley”},

{3, “Hermione Granger”}

Instancias de la Entidad Película:

{5, 2001, true, 152, 03/03/2001, “Harry Potter y la piedra filosofal”,

“Harry Potter and the Philosopher's Stone”, 1, 3}

{6, 2002, true, 152, 03/06/2002, “Harry Potter y la cámara secreta”,

“Harry Potter and the Chamber of Secrets”, 1, 3}

Instancias de la Entidad PersonajeDePelicula:

{1,5},{ 2,5},{3,5},{1,6},{ 2,6},{3,6}



Simbología en Diagramas de Entidad - Relación

Tipo de Cardinalidad / Opcionalidad	Simbología	Descripción de Finalidad y Observaciones
0 ó 1	 (0,1)	Ninguna o una única ocurrencia servirá de asociación entre dos entidades cualquiera. Si la relación existe, será una única ocurrencia de la Entidad.
1 y 1	 (1,1)	Una y solo una ocurrencia estará asociando una Entidad A con una Entidad B.
0 a N	 (0, N)	Ninguna, una o varias ocurrencias asocian dos entidades del modelo de datos. Si existe podrá ser efectivizado por una o varias ocurrencias de la Entidad.
1 a N	 (1, N)	Una o varias ocurrencias servirán siempre de asociaciones entre dos entidades cualquiera del modelo de Datos.



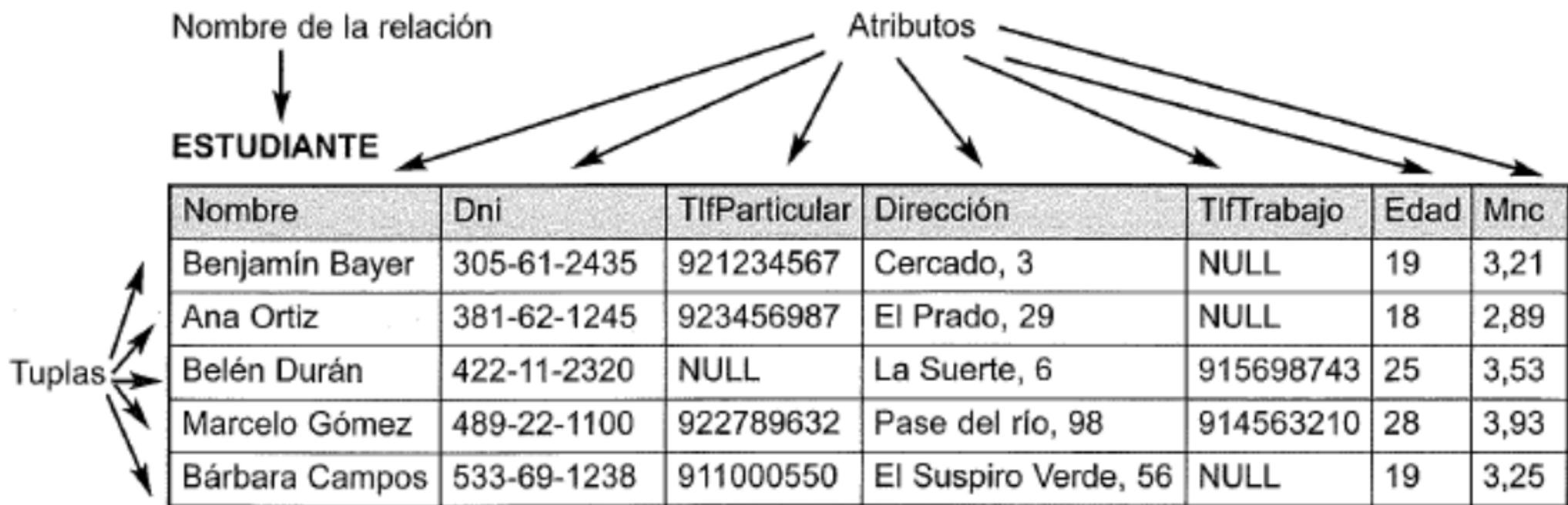
Modelo Lógico Relacional

- El Modelo Lógico Relacional es un modelo de datos conceptual de alto nivel.
- Este modelo y sus variaciones se utilizan con frecuencia para el diseño conceptual de las aplicaciones de base de datos.
- Este modelo es una forma de representar los datos (mediante tablas), y la manera para manipular esa representación (utilizando operadores).
- Se ocupa de tres aspectos de los datos: su **estructura**, su **integridad** y su **manipulación**.



Estructura de Datos Relacional

En la sección anterior se presentaron los conceptos de **entidad** y de **relación** como conceptos para el **modelado de datos reales**. En el **modelo relacional**, cada **fila de la tabla** representa un hecho que, por lo general, se corresponde con **una instancia de la entidad**.





Terminología en el Modelo Lógico

- Una **relación** corresponde a lo que conocemos como **tabla**, que se utiliza para representar los datos que queremos almacenar en nuestra base de datos. En el modelado de entidad relación es la **Entidad**.
- Un **atributo** corresponde a una **columna** o **campo**. El número de atributos se denomina **grado**.
- Una **tupla** corresponde a una **fila** o **registro** de esa **tabla**. En el modelado de entidad relación es lo que denominamos **instancia** de la Entidad. El número de tuplas de una tabla se denomina **cardinalidad**.
- La **clave primaria** es un identificador único para la tabla. Nunca existen dos filas de la tabla con el mismo valor en esa columna o combinación de columnas.
- Un **dominio** es una colección de valores, de los cuales uno o más atributo (columnas) obtienen sus valores reales.



Propiedades de las Relaciones

- No existen tuplas repetidas
- Las tuplas no tienen que estar ordenadas, necesariamente, (de arriba hacia abajo)
- Los atributos no tienen que estar ordenados, necesariamente, (de izquierda a derecha)
- Todos los valores de los atributos son atómicos (esto significa que debe tener un único valor, por ejemplo si es un atributo que contendrá un número de teléfono, solo puede contener un número de teléfono)



Integridad en las Bases de Datos Relacionales

- El objetivo de las reglas de integridad es informar al sistema administrador de Bases de Datos (DBMS), de ciertas restricciones del mundo real, (por ejemplo, los pesos de las piezas no pueden ser negativos).
- Se debe vigilar las operaciones de inserción y modificación y rechazar cualquier entrada que no cumpla con las especificaciones.
- El Modelo Relacional incluye dos reglas generales de integridad: *Reglas de Integridad para Clave Primaria* y *Reglas de Integridad para Clave Foránea*.



Reglas de integridad para Clave Primaria

Un atributo a (posiblemente compuesto) de la relación R es una clave candidata de R, sí y solo sí satisface las siguientes propiedades:

- **Unicidad:** no existen dos tuplas de R con el mismo valor de a, en un momento dado.
- **Minimalidad:** si a es un atributo compuesto, no puedo eliminar un componente de a sin destruir la propiedad de unicidad.

En el caso de que existan varias claves candidatas, debemos escoger una y las demás serán claves Alternativas (claves candidata que no son clave primaria).



¿Por qué son importantes las Claves Primarias?

- Son importantes porque constituyen el mecanismo de direccionamiento a nivel de tuplas, básico en un sistema relacional.
- Es el único modo garantizado por el sistema para localizar una tupla específica.
- Ningún componente de la clave primaria de una relación base puede aceptar nulos. Se entiende por nulo a un valor o representación que, por convención, no representa valor real del atributo aplicado.



En una base de datos relacional, nunca registraremos información de algo que no podamos identificar. Consideraciones para esta regla:

- Para las claves primarias compuestas: cada valor individual de la clave primaria debe ser no nulo en su totalidad.
- Esta regla se aplica a las relaciones base únicamente.
- Se aplica sólo a las claves primarias.

En caso de que la clave primaria fuera compuesta, se utiliza la abreviación PK para cada atributo que forma parte de la clave primaria.



Reglas de integridad para Clave Foránea

La **clave ajena** (FOREIGN KEY) es un atributo de una **relación R2** cuyos valores deben concordar con los de la clave primaria de alguna otra **relación R1** (no se requiere el caso inverso).

La **integridad referencial** se formula en términos de estados de la base de datos. para cada clave ajena es necesario responder tres preguntas :

1. ¿Puede aceptar nulos esa clave ajena?
2. ¿Qué deberá suceder si hay un intento de eliminar el objetivo de una referencia de clave ajena?
3. ¿Qué deberá suceder si hay un intento de modificar la clave primaria del objetivo de una referencia?



Ministerio de
Educación y Deportes
Presidencia de la Nación



Ministerio de Producción
Presidencia de la Nación

Lenguaje de Consulta Creación de Esquemas



Introducción a SQL

El nombre **SQL** significa Lenguaje de consulta estructurado (Structured Query Language).

SQL es un lenguaje de bases de datos global: cuenta con sentencias para definir datos, consultas y actualizaciones.

SQL utiliza los términos tabla, fila y columna para los términos relación, tupla y atributo del modelo relacional formal, respectivamente.

Sintaxis para:

- Lenguaje de definición de datos: **DDL** (Data Definition Language).
- Lenguaje de manipulación de datos: **DML** (Data Manipulation Language)



Lenguaje de definición de datos - DDL

Permite crear y definir nuevas bases de datos, campos e índices.

- o **CREATE**: Crea nuevas tablas, campos e índices.
- o **DROP**: Elimina tablas e índices.
- o **ALTER**: Modifica las tablas agregando campos o cambiando la definición de los campos.



Lenguaje de manipulación de datos - DML

Permite generar consultas para ordenar, filtrar y extraer datos de la base de datos.

- o **SELECT**: Consulta registros de la base de datos que satisfagan un criterio determinado.
- o **INSERT**: Carga lotes de datos en la base de datos en una única operación.
- o **UPDATE**: Modifica los valores de los campos y registros especificados.
- o **DELETE**: Elimina registros de una tabla de una base de datos.



Sintaxis para sentencias SQL

{Alternativas}, entre llaves se colocarán las palabras que tienen opciones o alternativas en la sentencia a la que pertenecen.

[Opcional], entre corchetes se colocarán las palabras que son opcionales en la sentencia, es decir que pueden colocarse o pueden obviarse.



Creación de Base de Datos

CREATE {DATABASE | SCHEMA} [IF NOT EXISTS]

nombre_base_datos

- Esta sentencia sirve para crear una base de datos con un nombre específico.
- Para poder crear una base de datos, el usuario que la crea debe tener privilegios de creación asignados.
- IF NOT EXISTS significa: SI NO EXISTE, por lo tanto, esto es útil para validar que la base de datos sea creada en caso de que no exista, si la base de datos existe y se ejecuta esta sentencia, se genera error.
- CREATE SCHEMA o CREATE DATABASE son sinónimos.

CREATE DATABASE IF NOT EXISTS complejo_de_cine



Creación de Tablas

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] nombre_de_tabla  
nombre_de_columna tipo_de_dato [NOT NULL | NULL] [DEFAULT  
valor_por_defecto][AUTO_INCREMENT] [UNIQUE | [PRIMARY] KEY  
[CONSTRAINT [nombre_relación] FOREIGN KEY (nombre_columna)
```

REFERENCES

```
nombre_de_tabla (nombre_columna)]  
[ON DELETE opciones_de_referencia]  
[ON UPDATE opciones_de_referencia]
```



Tipo de Datos más comunes

BIT[(longitud)]

INT[(longitud)]

BIGINT[(longitud)]

DOUBLE[(longitud,decimales)]

DATE **

TIME **

DATETIME **

CHAR[(longitud)] [BINARY] **

VARCHAR(longitud) [BINARY] **



Opciones de Referencias

Las opciones de referencia sirven para establecer que se hará en casos de que se elimine o se actualice una fila de la tabla primaria que está siendo referenciada por una fila de la tabla secundaria.

CASCADE: Eliminar o actualizar la fila de la tabla primaria, y automáticamente eliminar o actualizar las filas coincidentes en la tabla secundaria.

SET NULL: Eliminar o actualizar la fila de la tabla primaria, y establecer la columna de clave externa (Foreign key) de la tabla secundaria a NULL. Si se especifica una SET NULL, hay que asegurarse que no se haya declarado la columna de la tabla secundaria como NOT NULL.

RESTRICT: Rechaza la operación de eliminación o actualización en la tabla primaria.

SET DEFAULT: Para una operación de eliminación o actualización en la tabla primaria se establecerá un valor por defecto para la tabla secundaria.



Definición de Opcionalidades

Las opciones **NOT NULL | NULL**, sirven para especificar si dicha columna puede aceptar valores nulos: NULL, o si no puede guardar valores nulos: NOT NULL.

También se puede de manera opcional, indicar un valor por defecto **DEFAULT** para una columna.

AUTO_INCREMENT: se refiere a un valor AUTO INCREMENTAL; sirve para aquellas columnas con valores que numéricos enteros donde se necesita que dicho valor se incremente en uno por cada fila insertada en la tabla. Se utiliza muy frecuentemente en las claves primarias.

UNIQUE: sirve para indicar que una columna en la tabla no puede tener valores repetidos, debe ser UNICA. No pueden existir dos filas en la tabla que tengan el mismo valor para un atributo definido como UNIQUE.

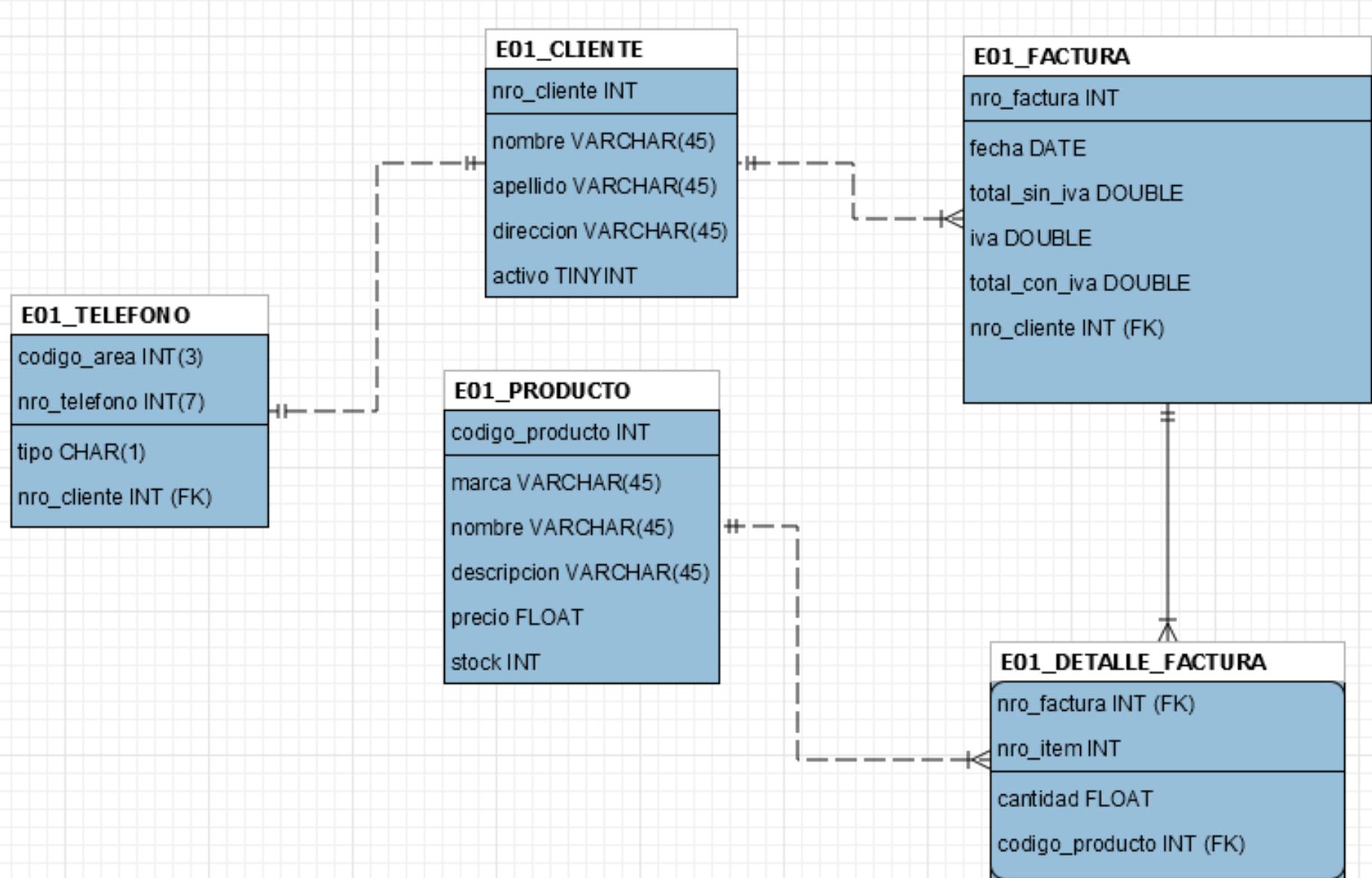


PRIMARY KEY: sirve para especificar que una columna es clave primaria de una tabla. Si una columna es clave primaria implica que sus valores no deben repetirse (UNIQUE).

CONSTRAINT: significa RESTRICCIÓN y se le asigna un nombre para identificarla; dicho nombre funciona como clave que identifica únicamente a cada CONSTRAINT que exista en la base de datos, por lo que toda CONSTRAINT debe tener un nombre no repetido; luego se indica con la palabra FOREIGN KEY, cuál es la columna de la tabla que funciona como clave foránea, indicando a continuación a cuál tabla y a cuál columna de dicha tabla hace referencia la clave foránea con la palabra REFERENCES.



Ejemplo Facturación





Creación de Tablas del Trabajo Práctico

```
CREATE TABLE IF NOT EXISTS `bd111mil`.`E01_CLIENTE` (
  `nro_cliente` INT NOT NULL,
  `nombre` VARCHAR(45) NOT NULL,
  `apellido` VARCHAR(45) NOT NULL,
  `direccion` VARCHAR(45) NOT NULL,
  `activo` TINYINT NOT NULL,
  PRIMARY KEY (`nro_cliente`))
ENGINE = InnoDB;
```



```
CREATE TABLE IF NOT EXISTS `bd111mil`.`E01_TELEFONO` (
  `codigo_area` INT(3) NOT NULL,
  `nro_telefono` INT(7) NOT NULL,
  `tipo` CHAR(1) NOT NULL,
  `nro_cliente` INT NOT NULL,
  PRIMARY KEY (`codigo_area`, `nro_telefono`),
  INDEX `fk_E01_TELEFONO_E01_CLIENTE1_idx` (`nro_cliente` ASC),
  UNIQUE INDEX `E01_CLIENTE_nro_cliente_UNIQUE` (`nro_cliente` ASC),
  CONSTRAINT `fk_E01_TELEFONO_E01_CLIENTE1`
    FOREIGN KEY (`nro_cliente`) REFERENCES `bd111mil`.`E01_CLIENTE`(`nro_cliente`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
```



```
CREATE TABLE IF NOT EXISTS `bd111mil`.`E01_FACTURA` (
    `nro_factura` INT NOT NULL,
    `fecha` DATE NOT NULL,
    `total_sin_iva` DOUBLE NOT NULL,
    `iva` DOUBLE NOT NULL,
    `total_con_iva` DOUBLE GENERATED ALWAYS AS (total_sin_iva+iva)
    VIRTUAL,
    `nro_cliente` INT NOT NULL,
    PRIMARY KEY (`nro_factura`),
    INDEX `IDX_E01_FK_FACTURA_CLIENTE` (`nro_cliente` ASC),
    CONSTRAINT `FK_E01_FACTURA_CLIENTE`
        FOREIGN KEY (`nro_cliente`) REFERENCES `bd111mil`.`E01_CLIENTE`(`nro_cliente`)
        ON DELETE NO ACTION ON UPDATE NO ACTION)
```



```
CREATE TABLE IF NOT EXISTS `bd111mil`.`E01_PRODUCTO` (
  `codigo_producto` INT NOT NULL,
  `marca` VARCHAR(45) NOT NULL,
  `nombre` VARCHAR(45) NOT NULL,
  `descripcion` VARCHAR(45) NOT NULL,
  `precio` FLOAT NOT NULL,
  `stock` INT NOT NULL,
  PRIMARY KEY (`codigo_producto`))
```



```
CREATE TABLE IF NOT EXISTS `bd111mil`.`E01_DETALLE_FACTURA` (
  `nro_factura` INT NOT NULL,
  `nro_item` INT NOT NULL,
  `cantidad` FLOAT NOT NULL,
  `codigo_producto` INT NOT NULL,
  PRIMARY KEY (`nro_factura`, `nro_item`),
  INDEX `fk_E01_DETALLE_FACTURA_E01_PRODUCTO1_idx`(`codigo_producto` ASC),
  INDEX `fk_E01_DETALLE_FACTURA_E01_FACTURA1_idx`(`nro_factura` ASC),
  CONSTRAINT `fk_E01_DETALLE_FACTURA_E01_PRODUCTO1`
    FOREIGN KEY(`codigo_producto`)
    REFERENCES `bd111mil`.`E01_PRODUCTO`(`codigo_producto`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_E01_DETALLE_FACTURA_E01_FACTURA1`
    FOREIGN KEY(`nro_factura`)
    REFERENCES `bd111mil`.`E01_FACTURA`(`nro_factura`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
```



Alteración de Tabla

Las acciones que se pueden hacer sobre una **Tabla** mediante la alteración son:

ADD: COLUMN, INDEX, KEY, CONSTRAINT, FULLTEXT

CHANGE: COLUMN

MODIFY: COLUMN

DROP: COLUMN, PRIMARY KEY, INDEX, KEY, FOREING KEY

DISABLE: KEYS

ENABLE: KEYS

RENAME: INDEX, KEY, TO, AS

ORDER BY



Ejemplos de Alteraciones

Para cambiar el nombre de una tabla de t1 a t2

ALTER TABLE t1 RENAME t2;



Para cambiar los tipos de datos de sus atributos, suponiendo que tenemos dos atributos a es INTEGER y b es CHAR con longitud 10: CHAR (10), cambiaremos a como TINYINT sin aceptar nulos y cambiaremos b por el nombre c y con longitud 20:

**ALTER TABLE t2 MODIFY a TINYINT NOT NULL,
CHANGE b c CHAR(20);**



Para agregar una nueva columna d, de tipo DATETIME:

ALTER TABLE t2 ADD d DATETIME;

Para agregar un nuevo índice en una columna d

ALTER TABLE t2 ADD INDEX (d);



Para eliminar la columna c:

ALTER TABLE t2 DROP COLUMN c;

**Para agregar una nueva columna auto incremental
y clave primaria, de tipo entera**

llamada e:

**ALTER TABLE t2 ADD e INT NOT NULL
AUTO_INCREMENT,
ADD PRIMARY KEY (e);**



Borrar Tabla

DROP TABLE remueve una o más tablas. Para poder eliminar tablas, el usuario que ejecuta la sentencia debe tener privilegios de **DROP** para cada tabla que quiera eliminar.

Con este comando se elimina todos los datos de la tabla, tanto la definición como las filas de datos de la misma. Si no existe la tabla que se desea borrar MySQL devuelve error, por eso es útil utilizar la opción IF EXISTS en caso de que exista la tabla la elimina.

DROP [TEMPORARY] TABLE [IF EXISTS]
nombre_tabla [,nombre_tabla] ...



Creación de Índices

```
CREATE INDEX nombre_indice
ON nombre_tabla (col_name [(length)] [ASC | 
DESC])
[opcion_algoritmo | opcion_bloqueo] ...
```