

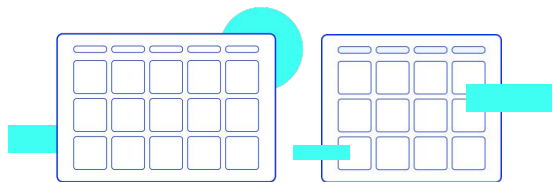
# Bootcamp Java Developer

Fase 1 - Java Analyst  
Módulo 6

# Consultas relacionadas

## Consultas de más de una tabla

Es posible consultar datos desde varias tablas en la misma sentencia **SELECT**. Esto permite realizar otras dos operaciones de álgebra relacional: el **producto cartesiano** y la **composición interna**.



### **Producto cartesiano**

Se obtiene mencionando las **dos tablas en una consulta sin ninguna restricción** en la cláusula **WHERE**.

El **producto cartesiano** de dos tablas son **todas las combinaciones de todas las filas de las dos tablas**. Usando una sentencia **SELECT** se deben listar todos los campos de ambas tablas. Los nombres de las tablas se indican en la cláusula **FROM** separados con comas.

## Producto cartesiano

Tabla: Productos							
idProducto	Nombre	Precio	Marca	Categoría	Presentación	Stock	Disponible
1	iPhone 6	499.99	1	Smartphone	16GB	500	SI
2	iPad Pro	599.99	1	Smartphone	128GB	300	SI
3	Nexus 7	299.99	4	Smartphone	32GB	250	NO
4	Galaxy S7	459.99	2	Smartphone	64GB	200	SI
5	Impresora T23	489.99	8	Impresoras	Color	180	NO
6	Impresora T33	399	8	Impresoras	Color	200	NO
7	Lavarropa 7000	1679	4	Lavarropas	Automático	100	SI
8	Camara Digital 760	649	9	Fotografía	Sin detalle	150	NO
9	Notebook CQ40-300	2999	7	Notebooks	Intel Core i3	100	SI

Tabla: Marcas	
idMarca	Nombre
1	Apple
2	Samsung
3	Huawei
4	LG
5	Motorola
6	Google
7	HP
8	Epson
9	Kodak

Analizando el panorama, podrás observar que en la tabla **Productos**, el campo **idProducto** es **Clave Primaria** y, por ende, no puede tener valores repetidos. Pero, en el campo **Marca**, el valor numérico hace referencia a la marca a la que pertenece el producto. **Una marca podría no tener ningún producto asociado, uno o muchos.**

Sintaxis:

```
SELECT * FROM Productos, Marcas;
```

**Nota:** Vale aclarar que la **concordancia lógica de los datos** jugará un **rol importante** a la hora de ejecutar este tipo de consultas.



Así, se obtiene la **combinación** de todos los registros de la primera tabla con todos los registros de la segunda tabla:

idProducto	Nombre	Precio	Marca	Categoría	Presentación	Stock	Disponible	idMarca	Nombre
1	iPhone 6	499.99	1	2	16GB	500	SI	1	Apple
1	iPhone 6	499.99	1	2	16GB	500	SI	2	Samsung
1	iPhone 6	499.99	1	2	16GB	500	SI	3	Huawei
1	iPhone 6	499.99	1	2	16GB	500	SI	4	LG
1	iPhone 6	499.99	1	2	16GB	500	SI	5	Motorola
1	iPhone 6	499.99	1	2	16GB	500	SI	6	Google
1	iPhone 6	499.99	1	2	16GB	500	SI	7	HP
1	iPhone 6	499.99	1	2	16GB	500	SI	8	Epson
1	iPhone 6	499.99	1	2	16GB	500	SI	9	Kodak

## Composición interna

La *composición interna* se trata de un **producto cartesiano restringido** en donde las **tuplas** (conjunto de nombres de atributos relacionados) que se emparejan deben **cumplir una condición determinada**.

Ejemplo:

```
SELECT * FROM Productos, Marcas  
WHERE Productos.Marca = Marcas.idMarca;
```



# Modelo *Entidad - Relación*

## Introducción

Cuando se utiliza una base de datos para gestionar información, se está plasmando una parte del mundo real en una serie de **tablas**, **registros** y **campos** ubicados en un dispositivo, lo cual crea un modelo parcial de la realidad. Antes de crear físicamente estas tablas, se debe realizar un **modelo de datos**.

El modelo ***Entidad-Relación (E-R)*** es uno de los varios modelos conceptuales existentes para el diseño de bases de datos.

Se suele cometer el error de ir creando nuevas tablas a medida que se van necesitando, haciendo así el modelo de datos y la construcción física de las tablas simultáneamente.

En el modelo ***E-R*** se parte de una **situación real** a partir de la cual se definen entidades y relaciones entre dichas entidades.



## Entidad


Una entidad es cualquier "objeto" discreto sobre el que se tiene información. Cada ejemplar de una entidad se denomina **instancia**. Las entidades son modeladas en la base de datos como tablas.




## Integridad referencial

La integridad referencial es un mecanismo que **garantiza la integridad de datos** en tablas relacionadas, ya que la misma **evita la existencia** de los llamados **registros huérfanos** (registros hijos sin su correspondiente registro padre).

Para establecer la integridad referencial es necesario crear en una **tabla hija**, una **clave externa o foránea** que esté relacionada a una clave primaria de la tabla padre.



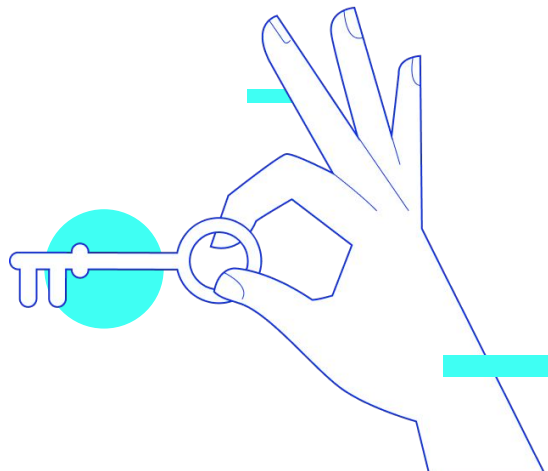
Es posible establecer el comportamiento de los registros en la tabla hija, cuando se producen actualizaciones de datos en la clave primaria de la tabla padre, o se eliminan registros en la tabla padre a través de la definición de **operaciones en cascada: ON UPDATE, ON DELETE**.



## Clave Foránea (*Foreign Key*)

La *Clave Foránea* de una tabla es aquella que **referencia a la *Clave Primaria*** de una tabla. Ésta puede referenciar a la *Clave Primaria* de la misma tabla o de otra. Ante una consulta SQL, se valida la legitimidad de los datos almacenados en una *Clave Foránea* y se fuerza la *integridad referencial*.

La *Clave Foránea* debe tener el **mismo tipo de datos que el campo al cual hace referencia**, es decir, la *Clave Primaria*.



Sintaxis de la **Clave Foránea**:

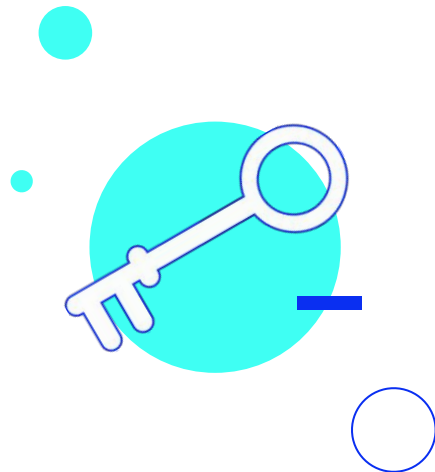
```
CREATE TABLE Facturas(Letra char NOT NULL, Numero int,  
id_articulo int UNSIGNED NOT NULL, -- Coincide en tipo/longitud con el campo al que será relacionado  
PRIMARY KEY (Letra, Numero),  
);
```

```
CREATE TABLE Articulos(id_articulo INT UNSIGNED NOT NULL AUTO_INCREMENT,  
Nombre VARCHAR(30) NOT NULL,  
PRIMARY KEY (id_articulo) -- Al definir una Primary Key, ésta podrá tener como enlace una Foreign Key  
);  
  
ALTER TABLE Facturas ADD FOREIGN KEY(id_articulo) REFERENCES Articulos(id_articulo) ON DELETE CASCADE  
ON UPDATE CASCADE;
```

## Súper llave

Es un conjunto de uno o más **atributos** que "juntos" permiten **identificar de forma única un registro en el conjunto de registros**. En pocas palabras: es un conjunto de atributos mediante los cuales es posible reconocer a un registro.

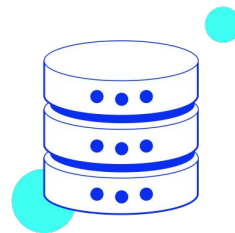
Este tipo de llaves contiene comúnmente **atributos ajenos**, es decir, atributos que no son indispensables para llevar a cabo el reconocimiento del registro.



## Conceptos clave

- **Clave candidata:** es una *súper llave* mínima. Una tabla puede tener varias llaves candidatas, pero solo una es elegida como *llave primaria*.
- **Relación:** una relación describe cierta interdependencia (de cualquier tipo) entre una o más entidades. Esta no tiene sentido sin las entidades que relaciona. Las relaciones son definidas con *claves primarias* y *claves foráneas* y mantienen la integridad referencial.
- **Cardinalidad de las Relaciones:** una relación describe cierta interdependencia (de cualquier tipo) entre una o más entidades. Las relaciones pueden ser:
  - **De uno a uno:** una instancia de la entidad A se relaciona con una y solamente una de la entidad B.
  - **De uno a muchos:** cada instancia de la entidad A se relaciona con varias instancias de la entidad B.
  - **De muchos a muchos:** cualquier instancia de la entidad A se relaciona con cualquier instancia de la entidad B.

- **Atributos:** las entidades tienen atributos. Un atributo de una entidad representa alguna propiedad que nos interesa almacenar en el *modelo de Bases de Datos*; los atributos son almacenados como columnas o campos de una tabla.
- **Consideraciones en el planeamiento del *Diseño Lógico de la Base de Datos*:**
  - Determinar el negocio y las necesidades del usuario.
  - Considerar cuáles son los problemas que hay que resolver y las tareas que los usuarios deberán completar.
  - Crear *Bases de Datos Normalizadas*.
  - Evitar el almacenamiento de información duplicada, inconsistencias en la base de datos, anomalías y problemas de pérdida de la información.



**¡Sigamos  
trabajando!**

