CONSULTAS RELACIONADAS

Consultas de más de una tabla

Es posible hacer consultas usando varias tablas en la misma sentencia **SELECT**.

Esto permite realizar otras dos operaciones de álgebra relacional: el **producto cartesiano** y la **composición interna**. 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 proyectan todos los atributos de ambas tablas. Los nombres de las tablas se indican en la cláusula **FROM** separados con comas.

Ejemplo de **producto cartesiano**:

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

En este caso, la visualización de registros resultantes estará compuesta por el producto entre la totalidad de registros de una tabla y la totalidad de registros de la otra tabla.

Otro ejemplo puede comprobar la cantidad de registros de la misma consulta mediante la función **count()**:

```
SELECT COUNT(*) FROM Productos, Marcas;
```

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 determinada condición.

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

Ejemplo:

Tabla: Productos							
idProducto	**Nombre**	**Precio**	**Marca**	**Categoria**	**Presentacion**	**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, se ve 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 qué Marca pertenece el producto. Una marca puede tener ningún producto asociado, o uno o muchos.

Ejemplo de Producto Cartesiano:

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

idProducto	Nombre	Precio	Marca	Categoria	Presentacion	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

Así se obtiene la combinación de todos los registros de la primera tabla con todos los registros de la segunda tabla. Vale aclarar que la concordancia lógica de los datos jugará un rol importante a la hora de ejecutar este tipo de consultas

Ejemplo de Composición Interna:

Observando la tabla resultante del producto cartesiano se puede ver que los registros válidos son aquellos en que los valores de los campos ID son iguales.

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

Resultado:

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

Modelo de 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 ordenador; creándose 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.

El modelo de datos más extendido es el denominado ENTIDAD/RELACIÓN (E/R) 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.

Clave Foránea (FOREIGN KEY)

La Clave Foránea referencia a la clave primaria de una tabla. Esta 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.

Sintaxis del FOREIGN KEY

```
ALTER TABLE NombreTabla ADD FOREIGN KEY(Campo1) REFERENCES TablaCategorizante (idTabla) ON DELETE
[CASCADE NO ACTION RESTRICT] ON UPDATE [CASCADE NO ACTION RESTRICT]
```

Ejemplo:
1) Crear la tabla Productos a sabiendas que hay/existirá una tabla Marcas en donde estarán registradas varios registros que representarán las marcas existentes.

Entonces, la tabla Productos tendrá un campo "categorizante" (en este caso, Marca) al cual se le asignarán valores que coincidan con los IDs existentes de la tabla Marcas.

```sql
CREATE TABLE Productos (
	idProducto INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,
	Nombre VARCHAR(50) NOT NULL,
	Precio DOUBLE NULL,
	Marca INT(11) UNSIGNED NOT NULL, -- Coincide en tipo/longitud con el campo al que será relacionado
	Categoria VARCHAR(30) NOT NULL,
	Presentacion VARCHAR(30) NOT NULL,
	Stock INT(6) NOT NULL,
	Disponible BOOLEAN NULL DEFAULT false,
	PRIMARY KEY (idProducto),
	KEY (Marca) -- Debe ser definido primero como "campo clave" para poder asignarle una Foreign Key
);
```

```sql
CREATE TABLE Marcas (
	idMarca INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,
	Nombre VARCHAR(30) NOT NULL,
	PRIMARY KEY (idMarca) -- Al definir una Primary Key esta podrá tener como enlace una Foreign Key
);
```

La relación se podrá entender de la siguiente forma: **La Tabla Productos, en su campo Marca, se "alimenta" de la Tabla Marcas a traves de su campo idMarca.**

```sql
ALTER TABLE Productos ADD FOREIGN KEY (Marca) REFERENCES Marcas(idMarca) ON DELETE CASCADE ON UPDATE CASCADE;
```

Conceptos Claves

Super llave

Es un conjunto de uno o más atributo que "juntos" identifican de manera única a una entidad.

Es un conjunto de uno o más atributos que, tomados colectivamente, permiten identificar de forma única un registro en el conjunto de registros. 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.

Clave candidata

Llave 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:

- Relaciones **de uno a uno**: una instancia de la entidad A se relaciona con una y solamente una de la entidad B.
- Relaciones **de uno a muchos:** cada instancia de la entidad A se relaciona con varias instancias de la entidad B.
- Relaciones **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.
- Considerando cuales son los problemas que hay que salvar y las tareas que los usuarios deberán completar

- Crear Bases de Datos Normalizadas.
- Prever innecesariamente información duplicada, inconsistencias en la base de datos, anomalías y problemas de pérdida de la información.