BASES DE DATOS

Elementos del lenguaje SQL I

Elementos del lenguaje SQL I

Capacidades

- Codifica y ejecuta los primeros scripts básicos en SQL.
- Realiza operaciones básicas con operadores y expresiones en una base de datos.

Temas

- Identificadores, expresiones y operadores
- Normalización de datos

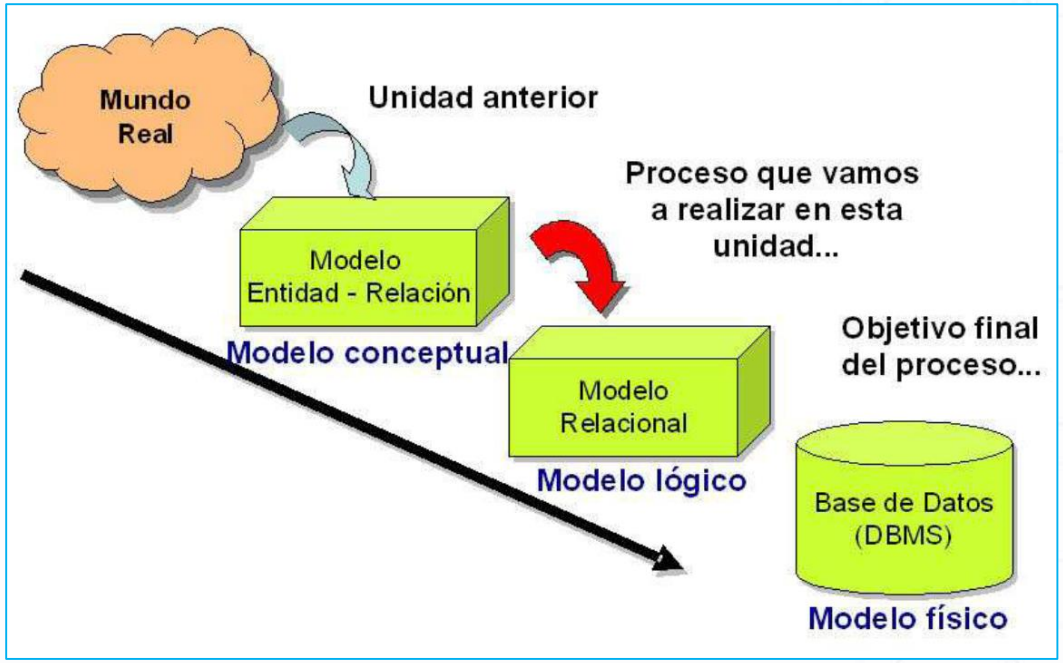
TEMA

Identificadores, expresiones y operadores

¿Qué es un modelo físico relacional?

Es uno de los modelos matemáticos más importantes y actuales para la representación de las bases de datos. Se basa en la **teoría matemática de las relaciones**, suministrándose por ello, una fundamentación teórica que permite aplicar todos los resultados de dicha teoría a problemas, tales como, el **diseño de sublenguajes de datos** y otros.

"Un modelo conceptual de datos identifica las relaciones de más alto nivel entre las diferentes entidades, mientras que un modelo lógico describe los datos con el mayor detalle posible".

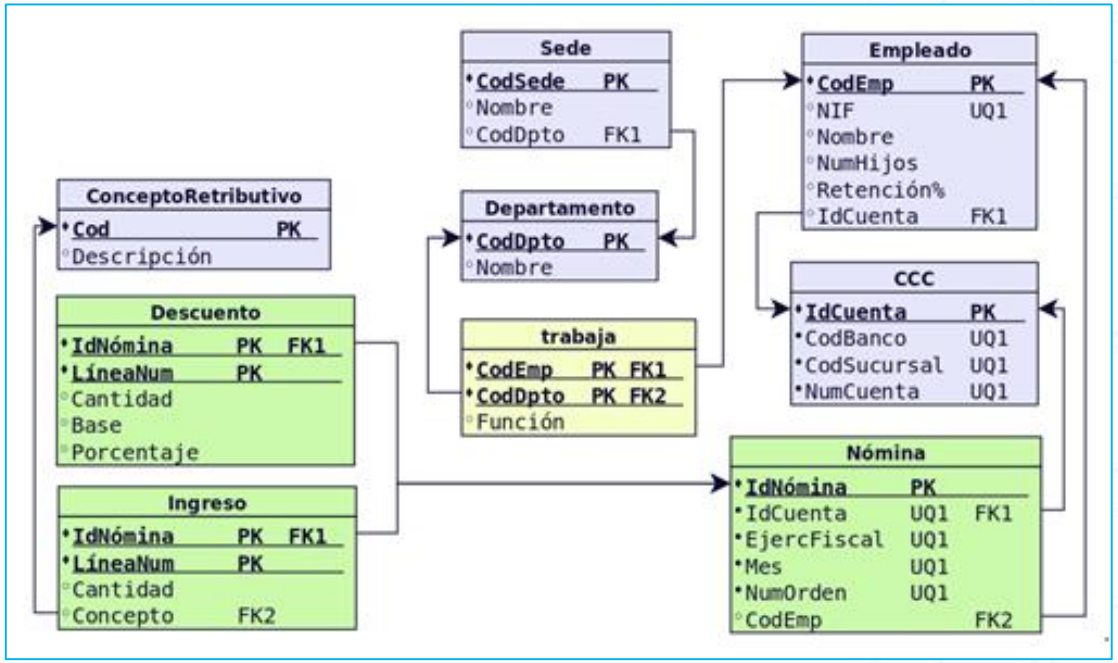


Aspectos de modelo relacional

El modelo relacional está orientado hacia dos aspectos sustanciales de los datos:

1. **Estructura de datos relacional:** Permite analizar las relaciones, así como, determinar los componentes que conforman cada una de ellas. Estos componentes son: las entidades (conocidas como tablas), los atributos (columnas de las tablas), las tuplas (cada fila de tabla), el dominio (valores de un atributo) y las relaciones (entre las tablas).
2. **Integridad de los datos:** Permite indicar ciertas restricciones propias del mundo real a la base de datos. Por ejemplo, no permitir dentro de una tabla de artículos, tuplas cuyo **atributo código sea nulo**.

El modelo relacional incluye dos reglas generales, asociados a los conceptos de **llaves primarias** y **llaves foráneas**.



Tipos de llaves en un base de datos relacional

Llave	Descripción
Primaria (*Primary key*)	Es una **columna** o **grupo de columnas** que **identifican de manera única a cada tupla de la tabla**. Por ser un identificador único, no se aceptan duplicados y su valor, generalmente, no se puede cambiar.
Alterna o candidata	Una tabla puede tener **más de una columna** o **combinación de columnas** que pueden **servir como la llave primaria de la tabla**. Por ejemplo, en una tabla de clientes se pueden establecer como llaves candidatas al código del cliente y al RUC.
Foránea (*Foreign key*)	Es una **columna** o **combinación de columnas**, dentro de una tabla que **se refieren a una llave primaria de otra tabla**. La tabla que contiene la llave foránea se le conoce como **relación referencial** y a la que contiene la llave primaria, como **relación objetivo**. La llave foránea no necesariamente puede formar parte de la llave primaria de una tabla.

Megarreglas o *constraint*

Las dos megarreglas más importantes son:

1. **Los *constraints* de integridad de entidades:** Por medio de estos se asegura que la llave primaria no puede ser **nula** ni **repetida**. Esto es lógico, ya que las tablas almacenan información proveniente del mundo real, del cual siempre existirán objetos distinguibles.
2. **Los *constraints* de integridad referencial:** Por medio de estos, una llave foránea debe coincidir con un valor de la llave primaria, es decir, el valor de la llave foránea debe ser concordante con algún valor de la llave primaria relacionada.

Constraint	Descripción
NOT NULL	Se asegura que la columna no tenga valores nulos
UNIQUE	Se asegura que cada valor en la columna no se repita
PRIMARY KEY	Es una combinación de NOT NULL y UNIQUE
FOREIGN KEY	Identifica de manera única una tupla en otra tabla
CHECK	Se asegura que el valor en la columna cumpla una condición dada
DEFAULT	Coloca un valor por defecto cuando no hay un valor especificado
INDEX	Se crea por columna para permitir búsquedas más rápidas

Tipos de datos SQL

Algunos de los tipos de datos más utilizados según su categoría son:

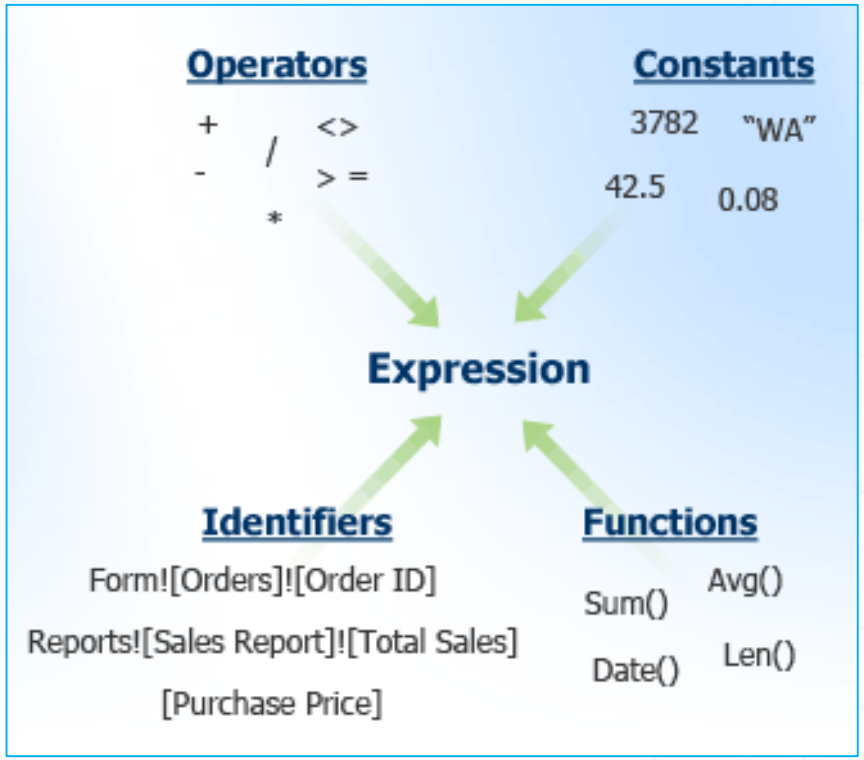
Dato	Rango	Almacenamiento
int	De -2^{31} (-2 147 483 648) a 2^{31} –1 (2 147 483 647)	4 bytes
bit	Tipo de datos entero que puede aceptar los valores 1, 0 o NULL	2 bytes
float(n)	De -1,79E+308 a -2,23E-308, 0 y de 2,23E-308 a 1,79E+308	Depende del valor de n
date	De 0001-01-01 a 9999-12-31	3 bytes
char(n)	Caracteres no unicode de longitud fija, con una longitud de n bytes. n debe ser un valor entre 1 y 8.	n bytes
varchar(n)	Caracteres no unicode de longitud variable. n indica que el tamaño de almacenamiento máximo es de 2^{31} - 1 bytes	n bytes (aprox.)
binary(n)	Datos binarios de longitud fija con una longitud de n bytes, donde n es un valor que oscila entre 1 y 8.	n bytes

Componentes de expresiones

Para crear una expresión, **se combinan identificadores** mediante **funciones**, **operadores**, **constantes** y **valores**. Cualquier expresión válida debe contener al menos una función o al menos un identificador, y también puede contener constantes u operadores. También puede usar una expresión como parte de otra, normalmente como argumento de una función. Esto se denomina **anidamiento de una expresión**. Por ejemplo, la siguiente expresión contiene componentes comunes:

Sum([Purchase Price])*0.08

- Sum() es una función integrada.
- [Purchase Price] es un identificador.
- * es un operador matemático.
- 0.08 es una constante.



Tipos de componentes

Componente	Formato general	Descripción
Identificador	[Collection name]![Object name].[Property name]	Solo se tiene que especificar suficientes elementos de un identificador para que sea único en el contexto de la expresión.
Función	Function(argument, argument)	Uno de los argumentos suele ser un identificador o una expresión. Algunas funciones no requieren argumentos.
Operador	Identifier operator Identifier	Hay excepciones a este formato.
Constante	Identifier comparison_operator Constant	-
Valor	-	Los valores pueden aparecer en varias lugares de una expresión.

Identificadores

Cuando usa un objeto, colección o propiedad en una expresión, se hace referencia a ese elemento mediante un **identificador**. Este incluye el **nombre del elemento que se está identificando** y también el **nombre del elemento al que pertenece**. Por ejemplo, el identificador de un campo incluye el nombre del campo y el nombre de la tabla a la que pertenece el campo. Un ejemplo de este tipo de identificador se encuentra en:

[Customers]![BirthDate]

Hay tres operadores que se pueden usar en un identificador:

- El operador de **signo de exclamación (!)**
- El operador de **punto (.)**
- El operador de **corchetes ([ ])**

```
CREATE DATABASE _NombreBdNuevo;
CREATE DATABASE #NombreBdNuevo;
GO

USE _NombreBdNuevo;

CREATE TABLE tablaNueva#@1(
 columna1$ VARCHAR(10)
 , #columna2@ VARCHAR(10)
 , columna3 VARCHAR(10)
 , CONSTRAINT nombreRestriccion$ PRIMARY KEY ( columna1$ )
)
```

Funciones

Una función es un procedimiento que se puede usar en una expresión. Algunas funciones, como **getdate()**, no necesitan ninguna entrada para que funcionen. Pero la mayoría de funciones requieren una entrada, que se denomina argumentos. Las funciones se pueden dividir en dos grupos (existen muchas mas, que **dependen del sistema de bases de datos** que se utilice):

- **Funciones agregadas SQL**, devuelven un solo valor, calculado con los valores de una columna. Por ejemplo, Avg(), Count(), Max(), Min(), Sum(), entre otras.
- **Funciones escalares SQL**, devuelven un solo valor basándose en el valor de entrada. Por ejemplo, Ucase(), Lcase(), Mid(), Len(), Format(), entre otras.

Operadores

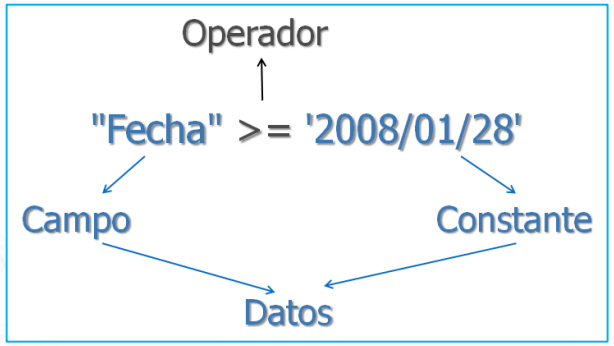
Un operador es una palabra o símbolo que indica una relación aritmética o lógica específica entre los demás elementos de una expresión. Los operadores pueden ser:

- **Aritmético**, como el signo más (+).
- **De comparación**, como el signo igual (=).
- **Lógico**, como Not.
- **De concatenación**, como &.
- **Especial**, como Like.

Constantes

Una constante es un valor conocido que no cambia y que se puede usar en una expresión. Hay cuatro constantes que se usan habitualmente:

- **True:** Indica algo que es lógicamente verdadero.
- **False:** Indica algo que es lógicamente falso.
- **Null:** Indica la falta de un valor conocido.
- **"" (empty string):** Indica un valor que se sabe que está vacío.



Valores

Puede utilizar los valores literales en las expresiones, como por ejemplo, el **número** 1.254 o la **cadena** "Introduzca un número entre 1 y 10". También puede usar valores numéricos, que pueden ser una serie de dígitos, incluyendo un signo y un punto decimal, si es necesario. En ausencia de un signo, SQL supone un valor positivo. Para **crear un valor negativo**, incluya el signo menos (**-**). También puede usar **notación científica**. Para ello, incluya "E" o "e" y el signo del exponente (por ejemplo, 1.0E-6).

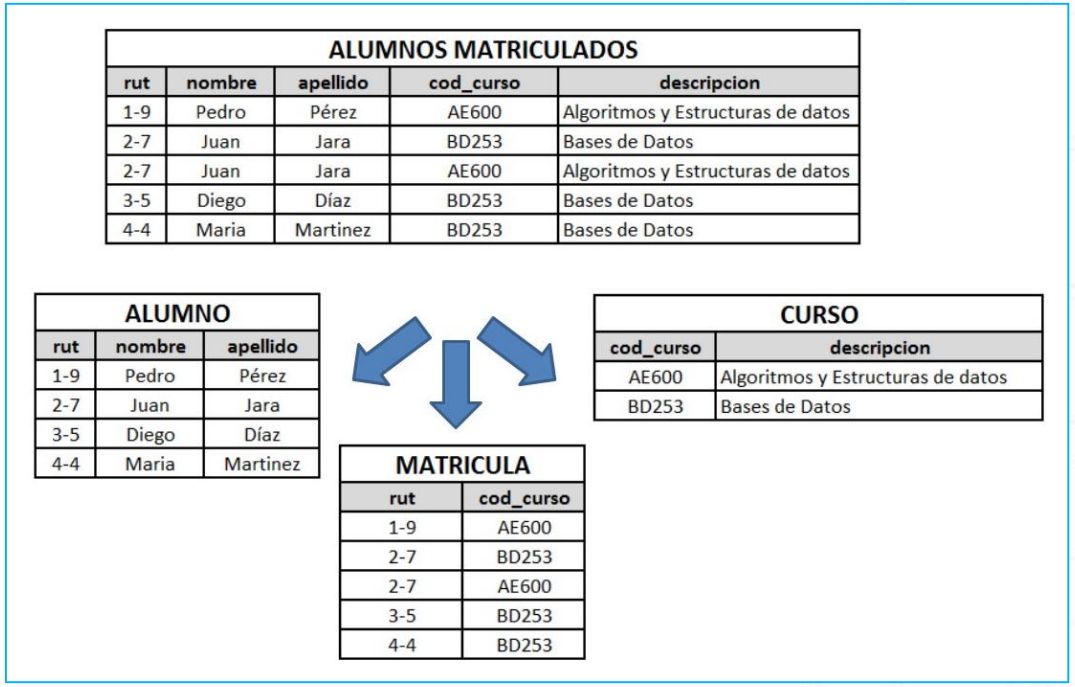
TEMA

Normalización de datos

¿En qué consiste la normalización de datos?

La normalización es la expresión formal del **modo de realizar un buen diseño**, pues provee los medios necesarios para describir la estructura lógica de los datos en un sistema de información. La **teoría de la normalización** se ha desarrollado para obtener estructuras de datos eficientes que **eviten las anomalías de actualización** además de **mejorar la independencia de los datos** permitiendo realizar extensiones de la base de datos, afectando muy poco o nada, a los sistemas de gestión de datos.

"El concepto de normalización fue introducido por E. D. Codd y fue pensado para aplicarse a sistemas relacionales; sin embargo, tiene aplicaciones más amplias".



ALUMNOS MATRICULADOS

rut	nombre	apellido	cod_curso	descripcion
1-9	Pedro	Pérez	AE600	Algoritmos y Estructuras de datos
2-7	Juan	Jara	BD253	Bases de Datos
2-7	Juan	Jara	AE600	Algoritmos y Estructuras de datos
3-5	Diego	Díaz	BD253	Bases de Datos
4-4	Maria	Martinez	BD253	Bases de Datos

ALUMNO

rut	nombre	apellido
1-9	Pedro	Pérez
2-7	Juan	Jara
3-5	Diego	Díaz
4-4	Maria	Martinez

MATRICULA

rut	cod_curso
1-9	AE600
2-7	BD253
2-7	AE600
3-5	BD253
4-4	BD253

CURSO

cod_curso	descripcion
AE600	Algoritmos y Estructuras de datos
BD253	Bases de Datos

Formas normales

La normalización involucra varias fases que se realizan en orden. La realización de la segunda fase supone que se ha concluido la primera y así sucesivamente. Generalmente, existen las siguientes formas normales:

- **Primera forma normal (1FN)**: Las columnas repetidas deben eliminarse y colocarse en tablas separadas. Las celdas de la tabla deben poseer valores simples y **no se permiten grupos ni arreglos repetidos como valores**. Todos los ingresos en cualquier columna deben ser del mismo tipo. Asimismo, cada columna debe tener un nombre único, el orden de las columnas en la tabla no es importante. Dos hileras en una tabla no deben ser idénticas, aunque el orden de las hileras no es importante.
- **Segunda forma normal (2FN):** Se debe tener la 1FN. Todas las dependencias parciales se deben eliminar y separar dentro de sus propias tablas (una **dependencia parcial** es un término que describe a aquellos datos que no dependen de la llave primaria de la tabla para identificarlos). Además, la 2FN solo se aplica para **llaves compuestas**.
- **Tercera forma normal (3FN):** Hay que eliminar y separar cualquier dato que no sea clave. El valor de esta columna debe depender de la llave. Todos los valores deben identificarse únicamente por la llave. Del mismo modo, **elimina cualquier dependencia transitiva** (aquella en la cual, las columnas que no son llave son dependientes de otras columnas que tampoco lo son).

Ejemplo de normalización de datos

A continuación, mostraremos el proceso de normalización de datos a una pequeña base de datos sobre órdenes de pedidos que cuenta con 6 registros y 11 atributos que incluye a la llave principal (**ID_ORDEN**).

A	B	C	D	E	F	G	H	I	J	K
TB_ORDENES										
ID_ORDEN	FECHA	ID_CLIENTE	NOM_CLIENTE	ESTADO	NUM_ARTICULO	NOM_ARTICULO	PRECIO_ACTUAL	PRECIO_VENTA	CANTIDAD	SUBTOTAL
2301	23/02/2011	101	MARTIN	CARACAS	3786	RED	35.00	35.00	3	105.00
2301	23/02/2011	101	MARTIN	CARACAS	4011	RAQUETA	65.00	65.00	6	390.00
2301	23/02/2011	101	MARTIN	CARACAS	9132	PAQ-3	4.75	4.75	8	38.00
2302	25/02/2011	107	HERNAN	CORO	5794	PAQ-6	5.00	5.00	4	20.00
2303	27/02/2011	110	PEDRO	MARACAY	4011	RAQUETA	65.00	65.00	2	130.00
2303	27/02/2011	110	PEDRO	MARACAY	3141	FUNDa	10.00	10.00	2	20.00

Primera forma normal (1FN)

Separamos a los datos iniciales en dos tablas: una que contiene una llave principal simple (ID_ORDEN) y otra una llave principal compuesta (ID_ORDEN + NUM_ARTICULO). Eliminamos valores repetidos de ID_ORDEN en la primera tabla.

TB_ORDENES

ID_ORDEN	FECHA	ID_CLIENTE	NOM_CLIENTE	ESTADO
2301	23/02/2011	101	MARTIN	CARACAS
2302	25/02/2011	107	HERNAN	CORO
2303	27/02/2011	110	PEDRO	MARACAY

TB_DET_ORDENES

ID_ORDEN	NUM_ARTICULO	NOM_ARTICULO	PRECIO_ACTUAL	PRECIO_VENTA	CANTIDAD	SUBTOTAL
2301	3786	RED	35.00	35.00	3	105.00
2301	4011	RAQUETA	65.00	65.00	6	390.00
2301	9132	PAQ-3	4.75	4.75	8	38.00
2302	5794	PAQ-6	5.00	5.00	4	20.00
2303	4011	RAQUETA	65.00	65.00	2	130.00
2303	3141	FUNDa	10.00	10.00	2	20.00

Segunda forma normal (2FN)

De acuerdo a la 1FN, TB_ORDENES tiene una llave compuesta, por lo cual podemos separarla en dos (TB_ORDENES y TB_CLIENTE), además, la tabla TB_DET_ORDENES también contiene una llave compuesta por lo cual podemos separarla (TB_DER_ORDENES y TB_PRODUCTO).

TB_ORDENES

ID_ORDEN	FECHA	ID_CLIENTE
2301	23/02/2011	101
2302	25/02/2011	107
2303	27/02/2011	110

TB_CLIENTE

ID_CLIENTE	NOM_CLIENTE	ESTADO
101	MARTIN	CARACAS
107	HERNAN	CORO
110	PEDRO	MARACAY

TB_DET_ORDENES

ID_ORDEN	NUM_ARTICULO	PRECIO_VENTA	CANTIDAD	SUBTOTAL
2301	3786	35.00	3	105.00
2301	4011	65.00	6	390.00
2301	9132	4.75	8	38.00
2302	5794	5.00	4	20.00
2303	4011	65.00	2	130.00
2303	3141	10.00	2	20.00

TB_PRODUCTO

NUM_ARTICULO	NOM_ARTICULO	PRECIO_ACTUAL
3786	RED	35.00
4011	RAQUETA	65.00
9132	PAQ-3	4.75
5794	PAQ-6	5.00
3141	FUNDa	10.00

Tercera forma normal (3FN)

Por último, en la tabla TB_DET_ORDENES el atributo SUBTOTAL puede obtenerse multiplicando el PRECIO_VENTA por CANTIDAD, así que podemos eliminarlo.

TB_ORDENES

ID_ORDEN	FECHA	ID_CLIENTE
2301	23/02/2011	101
2302	25/02/2011	107
2303	27/02/2011	110

TB_CLIENTE

ID_CLIENTE	NOM_CLIENTE	ESTADO
101	MARTIN	CARACAS
107	HERNAN	CORO
110	PEDRO	MARACAY

TB_DET_ORDENES

ID_ORDEN	NUM_ARTICULO	PRECIO_VENTA	CANTIDAD
2301	3786	35.00	3
2301	4011	65.00	6
2301	9132	4.75	8
2302	5794	5.00	4
2303	4011	65.00	2
2303	3141	10.00	2

Se eliminó la columna SUBTOTAL

TB_PRODUCTO

NUM_ARTICULO	NOM_ARTICULO	PRECIO_ACTUAL
3786	RED	35.00
4011	RAQUETA	65.00
9132	PAQ-3	4.75
5794	PAQ-6	5.00
3141	FUNDa	10.00

¿Qué aprendimos?

Conclusiones

- Se definieron a los identificadores, expresiones y operadores en el lenguaje SQL, pero previamente se recordó los aspectos más importantes del modelo relacional.
- Se explicó el proceso de normalización de datos mediante un ejemplo.

Operador
( ) Parentesis
*, /, % (Multiplicación, División, Modulo)
+, - (Añadir / Positivo / Concatenar, Substracción / Negativo)
=, <, >, <=, >=, <>, !=, !>, !< (Comparación)
NOT
AND
BETWEEN, IN, LIKE, OR
= (Asignación)