

Capítulo 4

Modelo físico relacional de base de datos

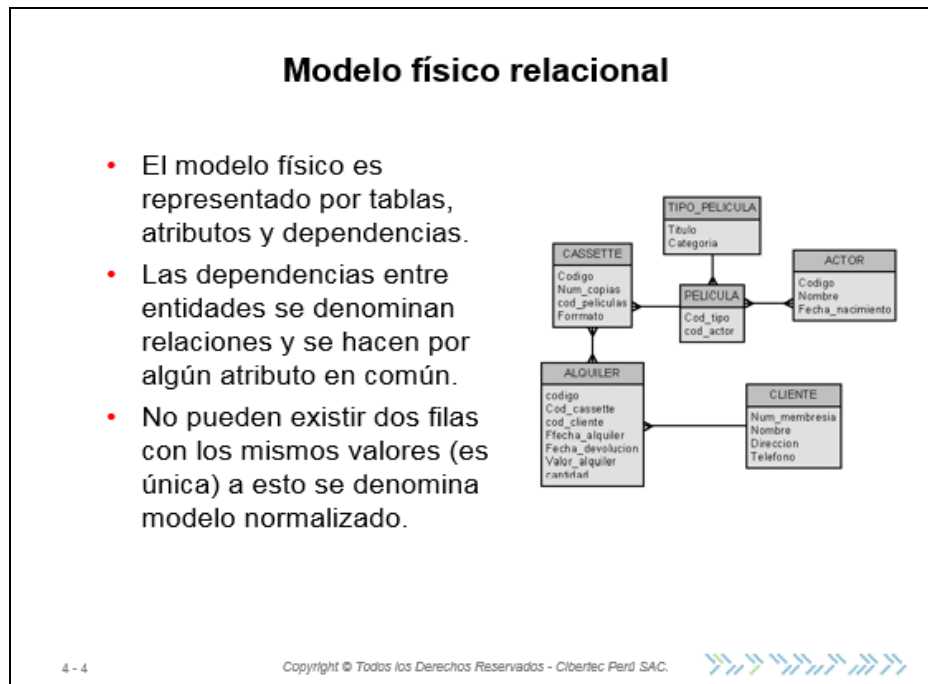
Al finalizar el capítulo, el alumno podrá:

- Reconocer el proceso del modelado lógico vs. físico.
- Aplicar las 4 formas normales de refinamiento de datos.
- Utilizar los conceptos de normalización o desnormalización.

Temas:

1. Modelo físico relacional
2. Generando el modelo físico
3. Normalización de datos

1. El modelo físico relacional



1.1 Definición

El enfoque 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 sub-lenguajes de datos y otros.

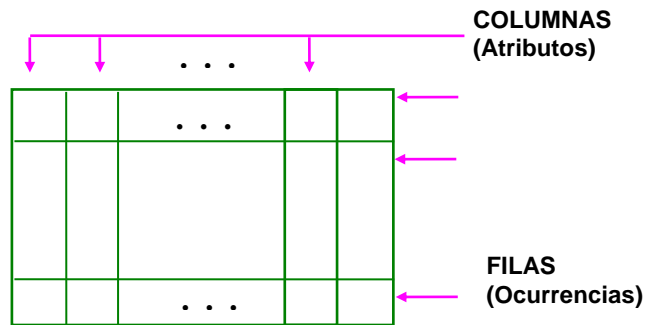
El término relación se puede definir matemáticamente, como sigue:

Relación:

Dados los conjuntos **D1, D2,..., Dn** (no necesariamente distintos), **R** es una relación sobre esos **n** conjuntos, si está constituida por un conjunto de **n-tuplos** ordenados **d1, d2,...,dn** tales que, **d1 ∈ D1, d2 ∈ D2,..., dn ∈ Dn**.

Los conjuntos **D1, D2,..., Dn** se llaman **dominios de R** y **n** constituye el grado de la relación. La cantidad de tuplas constituye la cardinalidad.

Es conveniente representar una relación como una tabla bidimensional donde cada fila representa un n-tuplo.



Aclaración del autor:

En el modelo relacional, tanto los objetos o entidades como las relaciones que se establecen entre ellos, se representan a través de **tablas**, que en la terminología relacional se denominan **relaciones**.

Cada relación está compuesta de filas (las ocurrencias de los objetos) y se les denomina, como tuplos, tuplas, uplos o uplas (en realidad, n-tuplos, pero en muchos casos se suprime la **n** cuando no existe posibilidad de confusión).

También, la relación está compuesta por columnas (los atributos o campos que toman valores en sus respectivos dominios).

No obstante, es importante considerar lo siguiente.

1. No hay dos filas (tuplas) iguales.
2. El orden de las filas no es significativo (1 y 2 se deben a que la relación es un conjunto).
El orden de las columnas sí es significativo, pues representa el orden de los dominios implicados; siempre se hace referencia a una columna por su nombre y nunca por su posición relativa.
3. El orden de las columnas no es significativo.
4. Cada valor dentro de la relación (cada valor de un atributo) es un dato atómico (o elemental), es decir, no descomponible; por ejemplo: un número, una cadena de caracteres. En otras palabras, en cada posición fila o columna existe un solo valor, nunca un conjunto de valores.

Una relación que satisface este último punto se denomina **normalizada**.

La teoría de la normalización se basa en la necesidad de encontrar una representación del conjunto de relaciones, que en el proceso de actualización sea más adecuada. Existen diferentes niveles de normalización, llamadas **formas normales**.

Ejemplo:

El ejemplo de suministrador y producto puede ser representado fácil y claramente, mediante el modelo relacional.

Los atributos de estas dos entidades son:

- **Suministrador:** número que lo identifica, nombre, tipo y distrito donde radica.
- **Producto:** número que lo identifica, nombre, precio unitario y peso.

Además, un suministrador puede suministrar muchos productos, mientras que un producto puede ser suministrado por varios suministradores. Así mismo, se conoce la cantidad de un determinado producto que suministra un suministrador dado.

SUMINISTRADOR

SNUM	SNOM	TIPO	DIST
S1	PÉREZ	30	SAN ISIDRO
S2	RAMOS	10	SURCO
S3	ARENAS	20	SAN ISIDRO
S4	VALLE	20	LINCE
S5	LÓPEZ	15	LINCE

SP

<u>S</u>	<u>P</u>	CANT
S1	P1	3
S1	P2	2
S1	P3	4
S1	P4	2
S1	P5	1
S1	P6	1
S2	P1	3
S2	P2	4
S3	P3	4
S3	P5	2
S4	P2	2
S4	P4	3
S4	P5	4

PRODUCTO

<u>PNUM</u>	PNOM	PRECIO	PESO
P1	CLAVO	0.10	12
P2	TUERCA	0.15	17
P3	MARTILLO	3.50	80
P4	TORNILLO	0.20	10
P5	ALICATE	2.00	50
P6	SERRUCHO	4.00	90

La representación en el modelo relacional es más simple que con el modelo jerárquico y el modelo reticular, ya que con 3 tablas se tiene todo el modelo representado.

En el modelo relacional el resultado de una demanda es también una relación y las demandas simétricas (en el sentido de ser una la inversa de la otra; por ejemplo, recuperar los números de los suministradores que suministran el producto 'P4' y recuperar los números de los productos que suministra el suministrador 'S2') requieren operaciones simétricas.

Asimismo, las diversas formas de expresar las recuperaciones dan lugar a los lenguajes relacionales cuyas formas más representativas son:

- Álgebra relacional (basado en las operaciones del álgebra de relaciones)
- Cálculo relacional (basado en el cálculo de predicados)

1.2 Ventajas del modelo relacional

- Simplicidad, pues el usuario formula sus demandas en términos del contenido informativo de la BD, sin tener que atender a las complejidades de la realización del sistema, lo que implica gran independencia de los datos.
- La información se maneja en forma de tablas, lo que constituye una manera familiar de representarla.

- Al igual que en el modelo reticular, si se tienen relaciones normalizadas no surgen grandes dificultades en la actualización.

En el modelo del **Suministrador-Producto**, presentado anteriormente, un ejemplo de cada tipo de operación de actualización, sería de la siguiente manera.

- **Creación:** añadir un producto **P7**. Se agrega la nueva ocurrencia en la tabla **Producto**. Es posible hacerlo, aunque ningún suministrador lo suministre.
- **Supresión:** se puede eliminar el suministrador **S1** sin perder el producto **P6**, a pesar de que es el único que lo suministra.
- **Modificación:** se puede cambiar el precio del producto **P2** sin necesidad de búsquedas adicionales ni posibilidad de inconsistencias.

No obstante, el proceso de normalización no es suficiente.



Aclaración del autor:

Se dice que una de las desventajas del modelo relacional consiste en la dificultad de lograr productividad adecuada de los sistemas, ya que no se emplean los medios técnicos idóneos, (tales como las memorias asociativas), siendo necesario simular este proceso, pero en realidad, la eficiencia y productividad de los sistemas actuales resultan realmente muy satisfactorias.

Ejemplos de SGBD relacionales:

Query by Example (QBE) (IBM)

MS Access, Informix, Oracle, SQL Server, My SQL

1.3 Aspectos del modelo relacional

El modelo relacional está orientado hacia 3 aspectos sustanciales de los datos:

1.3.1. Estructura de datos relacional

Bajo el enfoque de la estructura, el modelo relacional permite analizar las relaciones, así como, determinar los componentes que conforman cada una de ellas. Estos componentes son:

- **Relaciones:**
 - Generalmente, se denomina tabla (aunque existe una diferencia para el modelo relacional entre tabla y relación).
 - Cada tabla es una entidad diferente.
 - Las tablas almacenadas en la base de datos se denominan **Tablas bases**.
 - Las que se crean a partir de ellas, se denominan **Tablas virtuales** (Memoria).
 - Cada relación posee un nombre que está asociado con los datos que almacena.
 - Cada relación está formada, a su vez, por **Atributos y Tuplas**.
- **Atributos:**

Es un elemento de datos que describe a la entidad representada por medio de la tabla, que, en su implementación, conforma lo que sería una columna (campo) de dicha tabla. Cada atributo posee un nombre que, normalmente, sugiere el tipo de datos que se almacenará en dicho atributo.

- **Tuplas:**

Se denomina así a cada fila de la relación (tabla). Cada tupla refleja una ocurrencia de la relación.

- **Dominio:**

Es el conjunto de valores permitidos para un atributo, validado por reglas convencionales o del negocio.

Su implementación en una base de datos, libera a la misma de los famosos “errores” de los usuarios.

Existen los dominios simples, como, por ejemplo:

- **Sexo**, cuyos valores pueden ser **Masculino o Femenino**.
- **Estado Civil**, cuyos valores pueden ser **Soltero, Casado, Divorciado o Viudo**.
- **Tipo de Empleado**, cuyos valores pueden ser **Estable o Contratado**.
- **Notas** (evaluaciones) cuyos valores van desde 0 hasta 20.

De igual manera, existen los dominios compuestos, como, por ejemplo:

- **Fecha de ingreso**, compuesto por los dominios:
 - **Día**, cuyos valores van desde 1 hasta 31.
 - **Mes**, cuyos valores van desde 1 hasta 12.
 - **Año**, cuyos valores van desde 0 hasta 9999 (dependiendo del negocio).

- **Propiedades de las relaciones:**

No existen tuplas repetidas. Las tuplas y los atributos no están ordenados, “*todos los atributos tienen valores atómicos*”.

1.3.2. 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 2 reglas generales, aplicables para todas las bases de datos bajo este modelo. Estas 2 reglas están asociadas a los conceptos de llaves primarias y llaves foráneas.

- **Llave primaria (primary key - PK)**

Es una columna o grupo de columnas que identifican de manera única a cada renglón (fila) de la Tabla.

Como es un identificador único, no se aceptan duplicados en la llave primaria. El valor de la llave primaria, generalmente, no se puede cambiar.

Las llaves primarias que constan de más de una columna se llaman **Llaves compuestas**.

- **Llaves alternas o candidatas**

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. También, dentro de una tabla de empleados, se podría elegir la llave primaria entre el código del empleado o su DNI.

En resumen, las llaves candidatas deben cumplir 2 características:

- **Unicidad:** garantiza que en cualquier momento no existirán 2 tuplas con la misma Llave Primaria.
- **Minimalidad:** garantiza que, si la llave es compuesta, no se eliminará ninguno de sus componentes sin romper la propiedad de Unicidad.

Las llaves primarias son importantes, ya que el único modo que garantiza el modelo relacional para acceder a una tupla específica es por medio del valor de su llave primaria. Además, permitirán el manejo de las **Llaves foráneas**.

- **Llaves foráneas (foreign key)**

Es una columna o combinación de columnas, dentro de una tabla que se refieren a una llave primaria de otra tabla.

La relación (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 relación. En la relación **Empleados**, la llave foránea **Cargo** no es parte de la llave primaria. Sin embargo, en las relaciones producto de entidades asociativas, la llave primaria está compuesta por las llaves foráneas provenientes de las llaves primarias de las entidades que se asocian.

Las llaves foráneas sirven para hacer **"JOIN" (UNION)** de tablas y pueden ser repetidas o nulas.

- **Las megarreglas o constraints de integridad de datos**

Las dos MEGARREGLAS más importantes son:

- Los constraints de integridad de entidades.
- Los constraints de integridad referencial.

Por medio del constraint de integridad de entidades 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.

Por medio del constraint de integridad referencial, 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.

Existen 3 formas de asegurar la integridad referencial, en caso de sufrir alguna variación el valor de la llave primaria de donde proviene la respectiva llave foránea:

- Restringida
- De cascada
- Anulación

1.3.3. Operaciones con datos

- Adicionar registros.
- Actualizar registros.
- Eliminar registros.
 - Lógicamente
 - Físicamente
- Consultar registros.

2. Generando el modelo físico

Generando el modelo físico Numérico exacto		
Tipo	Descripción	Almacenamiento
tinyint	De 0 a 255	1 byte
smallint	De -2^{15} (-32.768) a $2^{15} - 1$ (32.767)	2 bytes
int	De -2^{31} (-2.147.483.648) a $2^{31} - 1$ (2.147.483.647)	4 bytes
bigint	De -263 (-9.223.372.036.854.775.808) a $2^{63} - 1$ (9.223.372.036.854.775.807)	8 bytes
decimal(p,s) o numeric(p,s)	<ul style="list-style-type: none"> p (precisión): el número total máximo de dígitos decimales que se puede almacenar, tanto a la izquierda como a la derecha del separador decimal s (escala): el número máximo de dígitos decimales que se puede almacenar a la derecha del separador decimal 	1 - 9: 5 bytes 10 - 19: 9 bytes 20 - 28: 13 bytes 29 - 38: 17 bytes
bit	Tipo de datos entero que puede aceptar los valores 1, 0 ó NULL	2 bytes
smallmoney	De - 214,7483648 a 214,7483647	4 bytes
money	Tipos de datos que representan valores monetarios o de moneda: de -922.337.203.685,4775808 a 922.337.203.685,4775807	8 bytes

4 - 7

Copyright © Todos los Derechos Reservados - Cibertec Perú SAC.



2.1 Introducción

El paso de un modelo lógico a uno físico requiere un profundo entendimiento del manejador de bases de datos que se desea emplear, incluyendo características como:

- Conocimiento a fondo de los tipos de objetos (elementos) soportados
- Detalles acerca del indexamiento, integridad referencial, restricciones, tipos de datos, etc.
- Detalles y variaciones de las versiones
- Parámetros de configuración
- Data Definition Language (DDL)

Como se comentó en el modelado lógico, el paso de convertir el modelo a tablas hace que las entidades pasen a ser tablas (más las derivadas de las relaciones) y los atributos se convierten en las columnas de dichas tablas.

2.2 Tipos de datos

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

Numérico exacto

Tipo de Dato	Intervalo	Almacenamiento
tinyint	De 0 a 255	1 byte
smallint	De -2^{15} (-32.768) a $2^{15} - 1$ (32.767)	2 bytes
int	De -2^{31} (-2.147.483.648) a $2^{31} - 1$ (2.147.483.647)	4 bytes
bigint	De -2^{63} (-9.223.372.036.854.775.808) a $2^{63} - 1$ (9.223.372.036.854.775.807)	8 bytes
decimal(p,s) o numeric(p,s)	<ul style="list-style-type: none"> p (precisión): el número total máximo de dígitos decimales que se puede almacenar, tanto a la izquierda como a la derecha del separador decimal. La precisión debe ser un valor comprendido entre 1 y la precisión máxima de 38. La precisión predeterminada es 18 s (escala): el número máximo de dígitos decimales que se puede almacenar a la derecha del separador decimal. La escala debe ser un valor comprendido entre 0 y p. Sólo es posible especificar la escala si se ha especificado la precisión. La escala predeterminada es 0 	Precisión 1 - 9: 5 bytes Precisión 10 - 19: 9 bytes Precisión 20 - 28: 13 bytes Precisión 29 - 38: 17 bytes
bit	Tipo de datos entero que puede aceptar los valores 1, 0 ó NULL	2 bytes
smallmoney	De - 214,7483648 a 214,7483647	4 bytes
money	Tipos de datos que representan valores monetarios o de moneda: de - 922.337.203.685,4775808 a 922.337.203.685,4775807	8 bytes

Numérico aproximado

Tipo de Dato	Intervalo	Almacenamiento
float	De $-1,79E+308$ a $-2,23E-308$, 0 y de $2,23E-308$ a $1,79E+308$	Depende del valor de <i>n</i>
real	De $-3,40E + 38$ a $-1,18E - 38$, 0 y de $1,18E - 38$ a $3,40E + 38$	4 bytes

Fecha y hora

Tipo de Dato	Intervalo	Almacenamiento
date	De 0001-01-01 a 9999-12-31	3 bytes
datetime2	Intervalo de fecha De 0001-01-01 a 9999-12-31 Intervalo de hora De 00:00:00 a 23:59:59.9999999	8 bytes
datetime	Intervalo de fecha De 1753-01-01 a 9999-12-31 Intervalo de hora De 00:00:00 a 23:59:59.997	8 bytes

datetimeoffset	Intervalo de fecha De 0001-01-01 a 9999-12-31 Intervalo de hora De 00:00:00 a 23:59:59.9999999 Intervalo de zona horaria De -14:00 a +14:00	10 bytes
smalldatetime	Intervalo de fecha De 1900-01-01 a 2079-06-06 Intervalo de hora De 00:00 a 23:59	4 bytes
time	De 00:00:00.0000000 a 23:59:59.9999999	5 bytes

Cadena de carácter

Tipo de Dato	Intervalo	Almacenamiento
char (n)	Caracteres no Unicode de longitud fija, con una longitud de n bytes. n debe ser un valor entre 1 y 8.000	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.)
text	En desuso, sustituido por varchar. Datos no Unicode de longitud variable con una longitud máxima de $2^{31} - 1$ (2.147.483.647) caracteres	max bytes (aprox.)

Carácter unicode

Tipo de Dato	Intervalo	Almacenamiento
nchar (n)	Datos de carácter Unicode de longitud fija, con n caracteres. n debe estar comprendido entre 1 y 4.000	2 * n bytes
nvarchar (n)	Datos de carácter Unicode de longitud variable. n indica que el tamaño máximo de almacenamiento es $2^{31} - 1$ bytes	2 * n bytes
ntext (n)	En desuso, sustituido por nvarchar. Datos Unicode de longitud variable con una longitud máxima de $2^{30} - 1$ (1.073.741.823) caracteres	2 * n bytes

Cadenas binarias

Tipo de Dato	Intervalo	Almacenamiento
binary (n)	Datos binarios de longitud fija con una longitud de n bytes, donde n es un valor que oscila entre 1 y 8.000	n bytes
varbinary (n)	Datos binarios de longitud variable. n indica que el tamaño de almacenamiento máximo es de $2^{31} - 1$ bytes	n bytes
Image	En desuso, sustituido por varbinary. Datos binarios de longitud variable desde 0	n bytes



	hasta 231 - 1 (2.147.483.647) bytes	
--	-------------------------------------	--

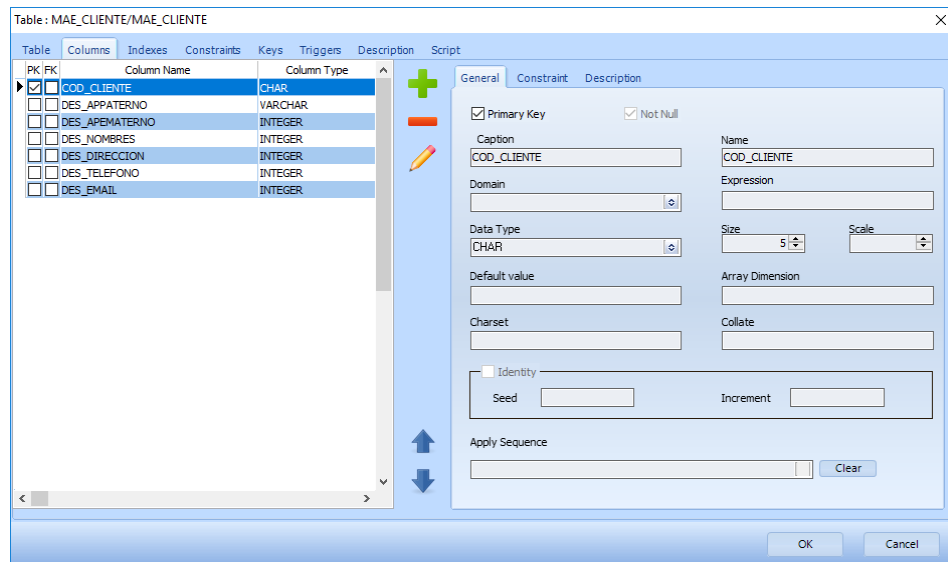
Otros tipos de datos

Tipo de Dato	Intervalo	Almacenamiento
cursor	Tipo de datos para las variables o para los parámetros de resultado de los procedimientos almacenados que contiene una referencia a un cursor. Las variables creadas con el tipo de datos cursor aceptan NULL.	
timestamp	Tipo de datos que expone números binarios únicos generados automáticamente en una base de datos. El tipo de datos timestamp es simplemente un número que se incrementa y no conserva una fecha o una hora.	8 bytes
sql_variant	Tipo de datos que almacena valores de varios tipos de datos aceptados en SQL Server, excepto text, ntext, image, timestamp y sql_variant.	
uniqueidentifier	Es un GUID (Globally Unique Identifier, Identificador Único Global)	16 bytes
table	Es un tipo de datos especial que se puede utilizar para almacenar un conjunto de resultados para su procesamiento posterior. table se utiliza principalmente para el almacenamiento temporal de un conjunto de filas devuelto como el conjunto de resultados de una función con valores de tabla.	
XML	Almacena datos de XML. Puede almacenar instancias de XML en una columna o una variable de tipo XML	

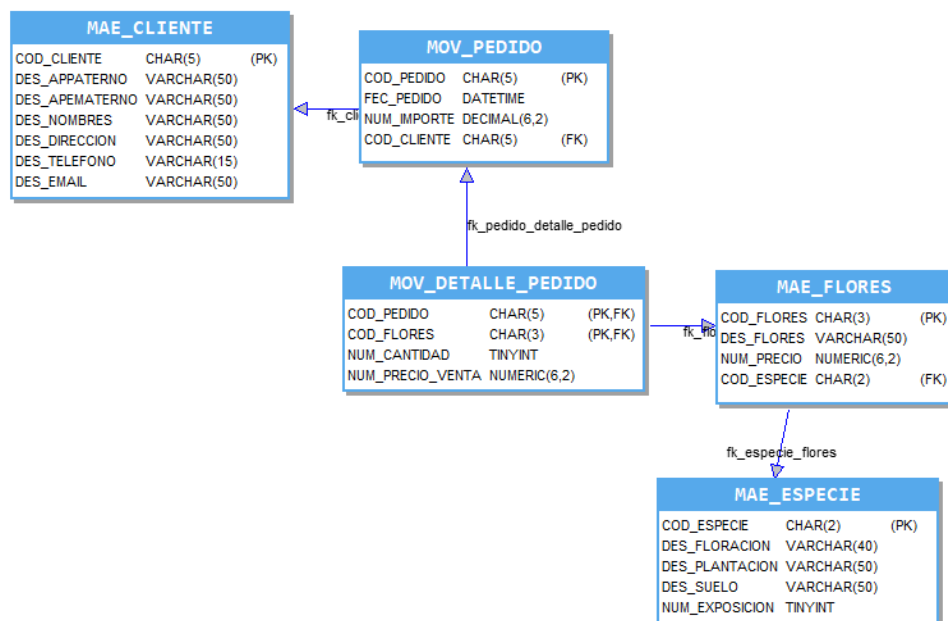
A continuación, se indicará como definir los tipos de datos para el modelo físico elaborado con ER/One Data Modeler.

PASO A: redefinición de los tipos de datos para el modelo físico

1. Hacer doble clic sobre la entidad y en el cuadro de diálogo seleccionar la ficha **Columns**.
2. Dar clic en la columna donde se va a realizar la redefinición del tipo de dato.
3. Hacer clic en el ícono del lápiz  para poder editar el atributo.
4. En la ficha **General**, en el control **Data Type**, modificar el tipo de dato que va a tener la columna. Por ejemplo, columna **cod_cliente** colocar como **CHAR** y en control **Size** colocar el 5, que quiere decir que solo va a recibir un carácter fijo de 5 posiciones.
5. Dar clic en el ícono check , para confirmar el cambio.



6. Redefinir los tipos de datos de cada atributo de la entidad, especificando el tamaño del mismo.



7. Al finalizar, hacer clic en **OK**.

PASO B: creación de la base de datos

En este punto, con la ayuda del SQL Server Managment Studio, crear una base de datos llamada **FLORISTERIA**, para el ejemplo.


PASO C: Creación de los objetos de la base de datos

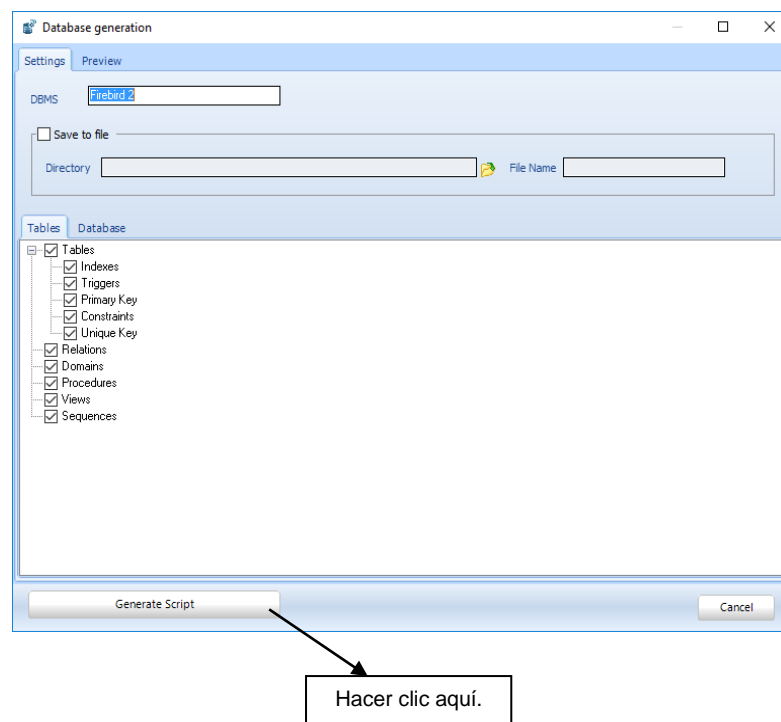
Para crear los objetos de la base de datos, ER/One Data Modeler crea un procedimiento que contiene todas las sentencias SQL que deben ejecutarse para crear los objetos definidos en el modelo físico. Por ello, se puede:

- Previsualizar el script de la creación de los objetos de base de datos.

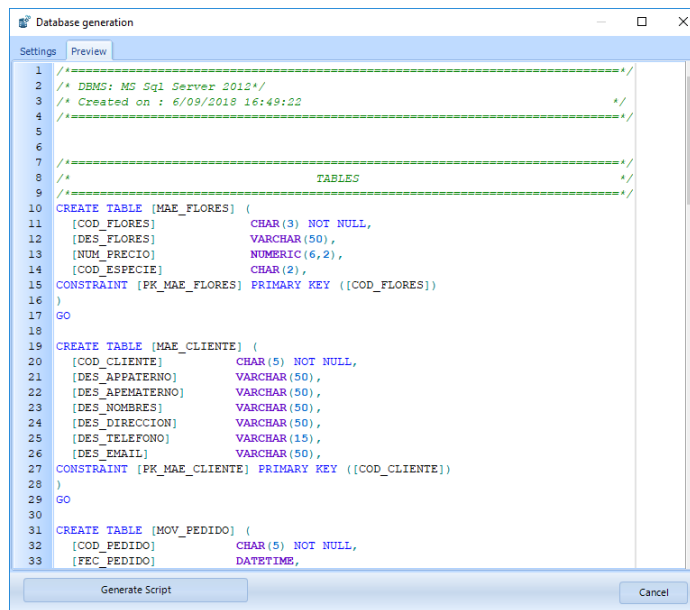
- Generar un archivo script (programa) que guarde el procedimiento que crea los objetos de la base de datos y ejecutar posteriormente, el procedimiento desde el cliente SQL Server.

Para revisar el procedimiento que crea los objetos de la base de datos


1. En el menú **Tools**, ejecutar la opción **Generate DLL Script** o dar clic en el siguiente ícono .
2. En el cuadro de diálogo **Database generation** se podrán ver todas las instrucciones que se ejecutarán para crear los objetos definidos en el modelo físico.
3. Dar clic en el botón **Generate Script** para previsualizar el script SQL

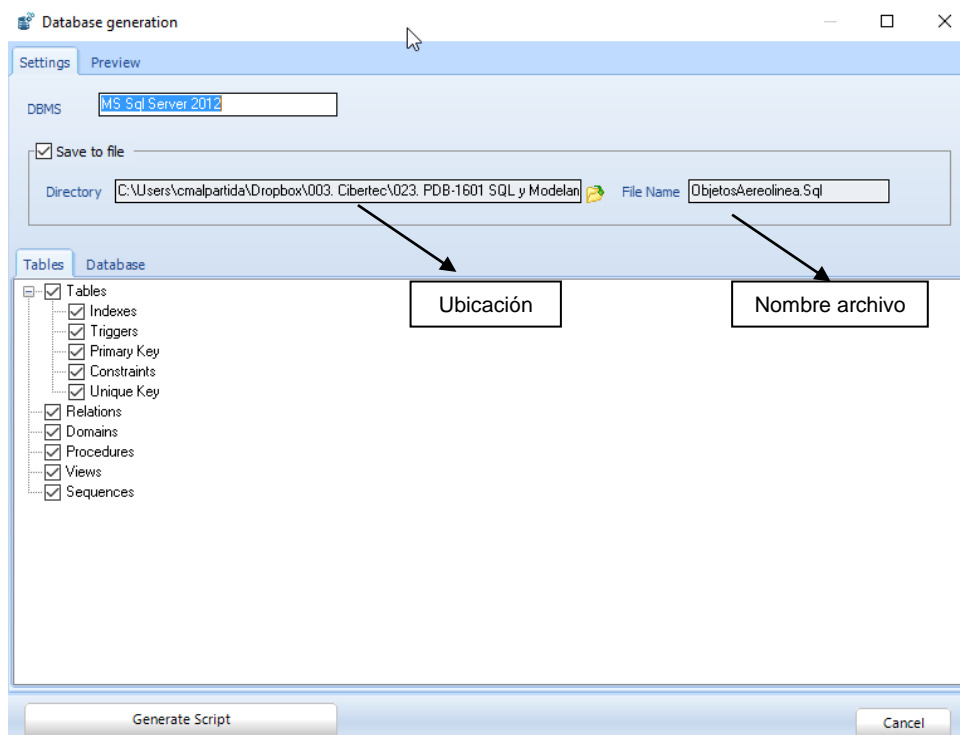


4. Hacer clic en el botón **Cancel**, al finalizar la revisión.



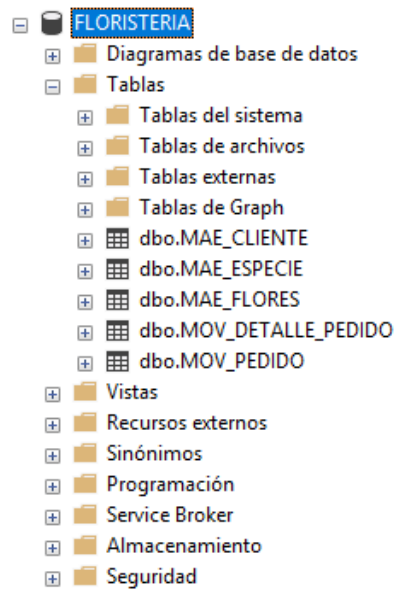
Para crear un script SQL que puede ser editado y ejecutado, posteriormente

1. En el menú **Tools**, ejecutar la opción **Generate DLL Script** o dar clic en el siguiente icono .
2. En el cuadro de diálogo **Database generation** se podrán ver todas las instrucciones que se ejecutarán para crear los objetos definidos en el modelo físico.
3. Dar clic en la opción **Save to File** y especificar la ubicación y el nombre del archivo que almacenará el script (guardarlo con extensión **sql**). Por ejemplo, **ObjetosAereolinea.sql**
4. Finalmente, dar clic en el botón **Generate Script**.



Revisión del servidor SQL para comprobar la creación de los objetos

1. Ingresar al **SQL Management Studio**; en **Authentication**, utilizar **SQL Server Authentication**, luego, utilizar **sa** para **Login** y **Password**.
2. Hacer clic sobre la base de datos **FLORISTERIA** utilizando el botón secundario del ratón.
3. Del menú contextual, ejecutar la opción **Actualizar**.
4. Expandir **FLORISTERIA** y hacer doble clic sobre **Tablas**.



3. Normalización de datos

Normalización de datos

Definición

- Propuesto por E.D. Codd para base de datos relacionales.
- Permite obtener estructuras eficientes y rápidas.
- Representa la expresión formal de un buen diseño.
- Mitiga las anomalías durante las actualizaciones de datos.
- Mejora la independencia de los datos.
- Permite un crecimiento controlado de los datos.

ID	Last Name	First Name
007	Bond	James
024	Bauer	Jack
123	Smith	Jane

ID	Cust_ID	Item	Qty
1	007	Fancy Gadget	4
2	024	Hand Gun	1
3	024	Bullet-proof vest	1

4 - 15
Copyright © Todos los Derechos Reservados - Cibertec Perú S.A.C.

3.1. Definición

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.

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

3.2. Ventajas de la normalización

- Evita anomalías en la actualización.
- Mejora la independencia de los datos permitiendo realizar extensiones de la BD, afectando muy poco o nada, a los programas de aplicación existentes que acceden la base de datos.

3.3. Las formas normales

La normalización involucra varias fases que se realizan en orden. La realización de la 2da. fase supone que se ha concluido la 1ra. y así sucesivamente.

Tras completar cada fase, se dice que la relación está en:

- Primera forma normal (1FN)
- Segunda forma normal (2FN)
- Tercera forma normal (3FN)
- Forma normal de Boyce-Codd (FNBC)

Existen, además, la cuarta (4FN) y la quinta (5FN) formas normales.

- **Primera forma normal (1FN):** establece que 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 (atributo) 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):** para que una tabla esté en 2NF tiene que cumplir con lo siguiente:
 - Estar en 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).
 - La 2FN solo se aplica para llaves compuestas.
- **La tercera forma normal (3FN):** señala que 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 aplicación:

A través de este ejercicio, se intenta afirmar los conocimientos de normalización, con un ejemplo simplificado de una base de datos para una pequeña biblioteca.

CodLibro	Título	Autor	Editorial	NombreLector	FechaDev
1001	Variable compleja	Murray Spiegel	McGraw Hill	Pérez Gómez, Juan	15/04/2005
1004	Visual Basic 2005	E. Petroustos	Anaya	Ríos Terán, Ana	17/04/2005
1005	Estadística	Murray Spiegel	McGraw Hill	Roca, René	16/04/2005
1006	Oracle University	Autor1: Nancy Greenberg Autor 2: Priya Nathan	Oracle Corp.	García Roque, Luis	20/04/2005
1007	Power Builder 10.0	Ramalho	McGraw Hill	Pérez Gómez, Juan	18/04/2005

Esta tabla no cumple el requisito de la primera forma normal (1NF) de solo tener campos atómicos, pues el nombre del lector es un campo que puede (y conviene) descomponerse en apellido paterno, apellido materno y nombres. Tal como se muestra en la siguiente tabla.

1NF

CodLibro	Título	Autor	Editorial	Paterno	Materno	Nombres	FechaDev
1001	Variable compleja	Murray Spiegel	McGraw Hill	Pérez	Gómez	Juan	15/04/2005
1004	Visual Basic 2005	E. Petroustos	Anaya	Ríos	Terán	Ana	17/04/2005
1005	Estadística	Murray Spiegel	McGraw Hill	Roca		René	16/04/2005
1006	Oracle University	Nancy Greenberg	Oracle Corp.	García	Roque	Luis	20/04/2005
1006	Oracle University	Priya Nathan	Oracle Corp.	García	Roque	Luis	20/04/2005
1007	Power Buiden 10.0	Ramalho	McGraw Hill	Pérez	Gómez	Juan	18/04/2005

Como se puede ver, hay cierta redundancia característica de 1NF.

La segunda forma normal (2NF) pide que no existan dependencias parciales o dicho de otra manera, todos los atributos no clave, deben depender por completo de la clave primaria.

Por ejemplo, el título es completamente identificado por el código del libro, pero el nombre del lector en realidad no tiene dependencia de este código, por tanto, estos datos deben ser trasladados a otra tabla.

2NF

CodLibro	Título	Autor	Editorial
1001	Variable compleja	Murray Spiegel	McGraw Hill
1004	Visual Basic 2005	E. Petroustos	Anaya
1005	Estadística	Murray Spiegel	McGraw Hill
1006	Oracle University	Nancy Greenberg	Oracle Corp.
1006	Oracle University	Priya Nathan	Oracle Corp.
1007	Power Buiden 10.0	Ramalho	McGraw Hill

La nueva tabla solo contendrá datos del lector.

CodLector	Paterno	Materno	Nombres
501	Pérez	Gómez	Juan
502	Ríos	Terán	Ana
503	Roca		René
504	García	Roque	Luis

Se ha creado una tabla para contener los datos del lector y también se creó la columna **CodLector** para identificar unívocamente a cada uno. Sin embargo, esta nueva disposición de la base de datos necesita que exista otra tabla para mantener la información de qué libros están prestados y a qué lectores.

CodLibro	CodLector	FechaDev
1001	501	15/04/2005
1004	502	17/04/2005
1005	503	16/04/2005
1006	504	20/04/2005
1007	501	18/04/2005

Para la tercera forma normal (3NF) la relación debe estar en 2NF y además, los atributos no clave, deben ser mutuamente independientes y dependientes por completo, de la clave primaria. Esto significa que las columnas en la tabla deben contener solamente información sobre la entidad definida por la clave primaria y, por tanto, las columnas en la tabla deben contener datos acerca de una sola cosa.

En el ejemplo en 2NF, la primera tabla conserva información acerca del libro, los autores y editoriales, por lo cual, se deberán crear nuevas tablas para satisfacer los requisitos de 3NF.

3NF

CodLibro	Título
1001	Variable compleja
1004	Visual Basic 2005
1005	Estadística
1006	Oracle University
1007	Power Builder 10.0

CodAutor	Autor
801	Murray Spiegel
802	E. Petroustsos
803	Nancy Greenberg
804	Priya Nathan
806	Ramalho

CodEditorial	Editorial
901	McGraw Hill
902	Anaya

CodEditorial	Editorial
903	Oracle Corp.

Aunque se han creado nuevas tablas para que cada una tenga información acerca de una entidad, también se ha perdido la información acerca de qué autor ha escrito qué libro y las editoriales correspondientes, por ende, se deberán crear otras tablas que relacionen cada libro con sus autores y editoriales.

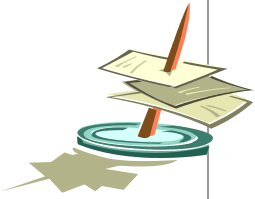
CodLibro	codAutor
1001	801
1004	802
1005	801
1006	803
1006	804
1007	806

CodLibro	codEditorial
1001	901
1004	902
1005	901
1006	903
1007	901

El resto de las tablas no necesitan modificación.

CodLector	Paterno	Materno	Nombres
501	Pérez	Gómez	Juan
502	Ríos	Terán	Ana
503	Roca		René
504	García	Roque	Luis

CodLibro	CodLector	FechaDev
1001	501	15/04/2005
1004	502	17/04/2005
1005	503	16/04/2005
1006	504	20/04/2005
1007	501	18/04/2005

**Normalización vs. Desnormalización:**

Cómo sería si en vez de normalizar las tablas de las bases de datos, se desnormalizarán; suena incoherente para alguien que ha seguido el dogma de diseño de bases de datos.

Sin entrar en grandes detalles, uno de los resultados casi directos de la normalización tiende a ser la obtención de varias tablas relacionadas unas con otras de una manera física.

El normalizar una base de datos, depende de la aplicación que se desarrolle. Si el diseño solo tiene una clase persistente, generalmente, se debería tener una tabla representándola.

Uno de los principales mitos respecto a las consecuencias de normalizar una BBDD es pensar que en un motor relacional es costoso procesar un join.

Esto incide en el hecho de tener tres tablas (resultado de una normalización) y luego, realizar una consulta sobre las tres tablas, entonces se tendrán que hacer 2 joins.

El tiempo de respuesta es menor si solo se consulta una tabla. En una aplicación pequeña esto pasa desapercibido, pero en una aplicación que maneja grandes volúmenes de acceso a datos, de seguro que si se toma en cuenta. Sin embargo, las consultas a una BBDD son solo una pequeña parte de lo que una aplicación hace con ellas.

Por otro lado, si se quiere eliminar un registro en una tabla, y esa tabla está relacionada con otras, se tendrá que eliminar también el registro de esas tablas.