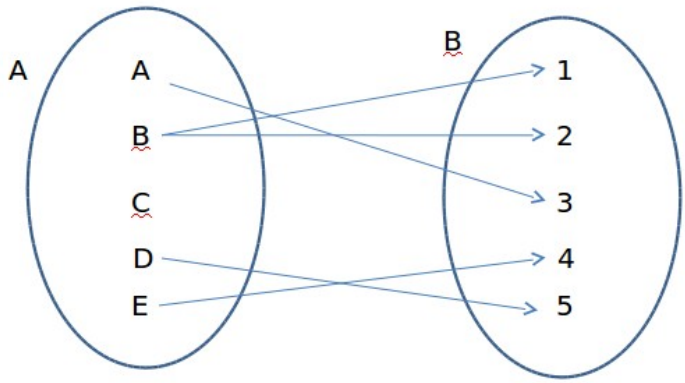
Sumario

Consultas de más de una tabla....2
Uso del Join....3
Modelo de Entidad - Relación....3
Introducción....3
Entidad....3
Clave Foránea (FOREIGN KEY)....3
Conceptos Claves....4
Super llave....4
Clave candidata....4
Relación....4
Cardinalidad de las Relaciones....4
Atributos....5
Normalización de Base de Datos....5
Dependencias....5
Dependencia funcional....5
Propiedades de la dependencia funcional....6
Dependencia funcional reflexiva....6
Dependencia funcional argumentativa....6
Dependencia funcional transitiva....6
Propiedades deducidas....7
Unión....7
Pseudo-Transitiva....7
Descomposición....7
Formas normales....7
Primera Forma Normal (1FN)....7
Segunda Forma Normal (2FN)....8
Tercera Forma Normal (3FN)....8
Forma normal de Boyce-Codd (FNBC)....8
Cuarta Forma Normal (4FN)....9
Quinta Forma Normal (5FN)....9
Reglas de Codd....9
Regla 1: La regla de la información....9
Regla 2: La regla del acceso garantizado....10
Regla 3: Tratamiento sistemático de los valores nulos....10
Regla 4: La regla de la descripción de la base de datos....10
Regla 5: La regla del sub-lenguaje integral....10
Regla 6: La regla de la actualización de vistas....11
Regla 7: La regla de insertar y actualizar....11
Regla 8: La regla de independencia física....11
Regla 9: La regla de independencia lógica....11
Regla 10: La regla de la independencia de la integridad....11
Las reglas de integridad....11
Regla 11: La regla de la distribución....12
Regla 12: Regla de la no-subversión....12

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 (Producto Relacionado).
El 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 proyectan todos los atributos de ambas tablas. Los nombres de las tablas se indican en la cláusula FROM separados con comas.



Producto Cartesiano
A1 A2 A3 A4 A5
B1 B2 B3 B4 B5
C1 C2 C3 C4 C5
D1 D2 D3 D4 D5
E1 E2 E3 E4 E5

Producto Relacionado
A3 B1 B2 D5 E4

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.
Ejemplo de composición interna:

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

Uso del Join

ANSI SQL brinda una instrucción para apoyar la unión entre tablas.

```
-- consultamos usando el comando join
select * from clientes c join facturas f on c.codigo=f.codigocliente;

-- muestra todos los elementos libres de la derecha
select * from clientes c left join facturas f on c.codigo=f.codigocliente;

-- muestra todos los elementos libres de la izquierda
select * from clientes c right join facturas f on c.codigo=f.codigocliente;
```

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.

```
CREATE TABLE Productos (
```

```
      idProducto INT UNSIGNED NOT NULL AUTO_INCREMENT,
      Nombre VARCHAR(50) NOT NULL,
      Precio DOUBLE NULL,
      Marca INT 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 NOT NULL,
      Disponible BOOLEAN NULL DEFAULT false,
      PRIMARY KEY(idProducto),
      KEY(Marca) -- se defina como "campo clave" para poder asignarle una
                 -- Foreign Key
);

CREATE TABLE Marcas(
      idMarca INT 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.

```
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.

Normalización de Base de Datos

La **normalización de bases de datos** es un proceso que consiste en designar y aplicar una serie de reglas a las relaciones obtenidas tras el paso del modelo entidad-relación al modelo relacional.

Las bases de datos relacionales se normalizan para:

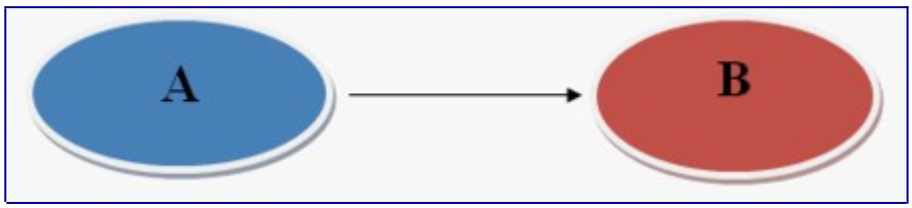
- Evitar la redundancia de los datos.
- Disminuir problemas de actualización de los datos en las tablas.
- Proteger la integridad de datos.

En el modelo relacional es frecuente llamar *tabla* a una relación; para que una tabla sea considerada como una relación tiene que cumplir con algunas restricciones:

- Cada tabla debe tener su nombre único.
- No puede haber dos filas iguales. No se permiten los duplicados.
- Todos los datos en una columna deben ser del mismo tipo.

Dependencias

Dependencia funcional



B es funcionalmente dependiente de **A**.

Una dependencia funcional es una conexión entre uno o más atributos. Por ejemplo, si se conoce el valor de *DNI(Documento Nacional de Identidad-España)* tiene una conexión con *Apellido* o *Nombre* .

Las dependencias funcionales del sistema se escriben utilizando una flecha, de la siguiente manera:

FechaDeNacimiento Edad

De la normalización (lógica) a la implementación (física o real) puede ser sugerible tener estas dependencias funcionales para lograr la eficiencia en las tablas.

Propiedades de la dependencia funcional

Existen tres axiomas de Armstrong:

Dependencia funcional reflexiva

Si "y" está incluido en "x" entonces x y

A partir de cualquier atributo o conjunto de atributos siempre puede deducirse él mismo. Si la dirección o el nombre de una persona están incluidos en el DNI, entonces con el DNI podemos determinar la dirección o su nombre.

Dependencia funcional argumentativa

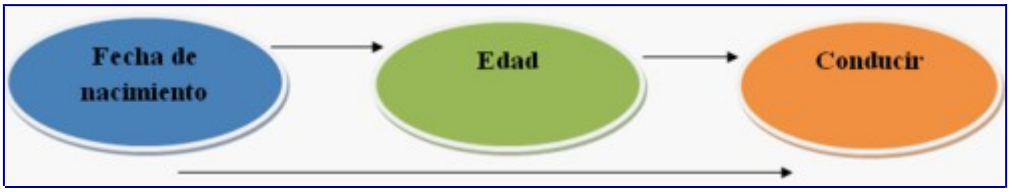
 entonces

DNI nombre

DNI,dirección nombre,dirección

Si con el DNI se determina el nombre de una persona, entonces con el DNI más la dirección también se determina el nombre y su dirección.

Dependencia funcional transitiva



Dependencia funcional transitiva.

Sean *X, Y, Z* tres atributos (o grupos de atributos) de la misma entidad. Si *Y* depende funcionalmente de *X* y *Z* de *Y*, pero *X* no depende funcionalmente de *Y*, se dice entonces que *Z* depende transitivamente de *X*. Simbólicamente sería:

X Y Z entonces *X Z*

FechaDeNacimiento Edad

Edad Conducir

FechaDeNacimiento Edad Conducir

Entonces entendemos que *FechaDeNacimiento* determina a *Edad* y la *Edad* determina a *Conducir*, indirectamente podemos saber a través de *FechaDeNacimiento* a *Conducir* (En muchos países, una persona necesita ser mayor de cierta edad para poder conducir un automóvil, por eso se utiliza este ejemplo).

"C será un dato simple (dato no primario), B, será un otro dato simple (dato no primario), A, es la llave primaria (PK). Decimos que C dependerá de B y B dependerá funcionalmente de A."

Propiedades deducidas

Unión

y entonces

Pseudo-Transitiva

y entonces

Descomposición

y está incluido en entonces

Formas normales

Las formas normales son aplicadas a las tablas de una base de datos. Decir que una base de datos está en la forma normal **N** es decir que todas sus tablas están en la forma normal **N**.

Diagrama de inclusión de todas las formas normales.

En general, las primeras tres formas normales son suficientes para cubrir las necesidades de la mayoría de las bases de datos. El creador de estas 3 primeras formas normales (o reglas) fue Edgar F. Codd.1

Primera Forma Normal (1FN)

Artículo principal: *Primera forma normal*

Una tabla está en primera forma si.

- Todos los atributos son atómicos. Un atributo es atómico si los elementos del dominio son simples e indivisibles.
- No debe existir variación en el número de columnas.
- Los campos no clave deben identificarse por la clave (dependencia funcional).
- Debe existir una independencia del orden tanto de las filas como de las columnas; es decir, si los datos cambian de orden no deben cambiar sus significados.

Esta forma normal elimina los valores repetidos dentro de una base de datos.

Segunda Forma Normal (2FN)

Artículo principal: *Segunda forma normal*

Dependencia funcional. Una relación está en 2FN si está en 1FN y si los atributos que no forman parte de ninguna clave dependen de forma completa de la clave principal. Es decir, que no existen dependencias parciales. Todos los atributos que no son clave principal deben depender únicamente de la clave principal.

En otras palabras, podríamos decir que la segunda forma normal está basada en el concepto de dependencia completamente funcional. Una dependencia funcional es completamente funcional si al eliminar los atributos A de X significa que la dependencia no es mantenida, esto es que . Una dependencia funcional es una dependencia parcial si hay algunos atributos que pueden ser eliminados de X y la dependencia todavía se mantiene, esto es .

Por ejemplo {DNI, ID_PROYECTO} HORAS_TRABAJO (con el DNI de un empleado y el ID de un proyecto sabemos cuántas horas de trabajo por semana trabaja un empleado en dicho proyecto) es completamente funcional dado que ni DNI HORAS_TRABAJO ni ID_PROYECTO HORAS_TRABAJO mantienen la dependencia. Sin embargo {DNI, ID_PROYECTO} NOMBRE_EMPLEADO es parcialmente dependiente dado que DNI NOMBRE_EMPLEADO mantiene la dependencia.

Tercera Forma Normal (3FN)

Artículo principal: *Tercera forma normal*

La tabla se encuentra en 3FN si es 2FN y si no existe ninguna dependencia funcional transitiva en los atributos que no son clave.

Un ejemplo de este concepto sería que, una dependencia funcional X Y en un esquema de relación R es una dependencia transitiva si hay un conjunto de atributos Z que no es un subconjunto de alguna clave de R, donde se mantiene X Z y Z Y.

Por ejemplo, la dependencia SSN DMGRSSN es una dependencia transitiva en EMP_DEPT de la siguiente figura. Decimos que la dependencia de DMGRSSN el atributo clave SSN es transitiva vía DNUMBER porque las dependencias SSN → DNUMBER y DNUMBER → DMGRSSN son mantenidas, y DNUMBER no es un subconjunto de la clave de EMP_DEPT. Intuitivamente, podemos ver que la dependencia de DMGRSSN sobre DNUMBER es indeseable en EMP_DEPT dado que DNUMBER no es una clave de EMP_DEPT.

Formalmente, un esquema de relación está en 3 Forma Normal Elmasri-Navâthe,2 si para toda dependencia funcional , se cumple al menos una de las siguientes condiciones:

1. es superllave o clave.
2. es atributo primo de ; esto es, si es miembro de alguna clave en .

Además el esquema debe cumplir necesariamente, con las condiciones de segunda forma normal.

Forma normal de Boyce-Codd (FNBC)

Artículo principal: *Forma normal de Boyce-Codd*

La tabla se encuentra en FNBC si cada determinante, atributo que determina completamente a otro, es clave candidata. Deberá registrarse de forma anillada ante la presencia de un intervalo seguido de una formalización perpetua, es decir las variantes creadas, en una tabla no se llegaran a mostrar, si las ya planificadas, dejan de existir.

Formalmente, un esquema de relación está en FNBC, si y sólo si, para toda dependencia funcional válida en , se cumple que

1. es superllave o clave.

De esta forma, todo esquema que cumple FNBC, está además en 3FN; sin embargo, no todo esquema que cumple con 3FN, está en FNBC.

Cuarta Forma Normal (4FN)

Artículo principal: *Cuarta forma normal*

Una tabla se encuentra en 4FN si, y solo si, para cada una de sus dependencias multivaluadas no funcionales X Y, siendo X una super-clave que, X es una clave candidata o un conjunto de claves primarias.

Quinta Forma Normal (5FN)

Artículo principal: *Quinta forma normal*

Una tabla se encuentra en 5FN si:

- La tabla está en 4FN
- No existen relaciones de dependencias de reunión (join) no triviales que no se generen desde las claves. Una tabla que se encuentra en la 4FN se dice que está en la 5FN si, y sólo si, cada relación de dependencia de reunión (join) se encuentra definida por claves candidatas. Por lo que si se aplicara una consulta entre al menos tres relaciones independientes entre sí dentro de la 4FN y se obtuvieran tuplas espurias, entonces no estaría dentro de la 5FN.

Reglas de Codd

Edgar Frank Codd se percató de que existían bases de datos en el mercado que decían ser relacionales, pero lo único que hacían era guardar la información en las tablas, sin estar literalmente normalizadas dichas tablas; entonces Codd publicó doce (12) reglas que un verdadero sistema relacional debería tener, en la práctica algunas de ellas son difíciles de realizar. Un sistema podrá considerarse "más relacional" cuanto más siga estas reglas.

Regla 1: La regla de la información

Toda la información en un RDBMS está explícitamente representada de una sola manera por valores en una tabla.

Cualquier cosa que no exista en una tabla no existe del todo. Toda la información, incluyendo nombres de tablas, nombres de vistas, nombres de columnas, y los datos de las columnas deben estar almacenados en tablas dentro de las bases de datos. Las tablas que contienen tal información constituyen el Diccionario de Datos. Esto significa que todo tiene que estar almacenado en las tablas.

Toda la información en una base de datos relacional se representa explícitamente en el nivel lógico exactamente de una manera: con valores en tablas. Por tanto los metadatos (diccionario, catálogo) se representan exactamente igual que los datos de usuario. Y puede usarse el mismo lenguaje (ej. SQL) para acceder a los datos y a los metadatos (regla 4).

Regla 2: La regla del acceso garantizado

Cada ítem de datos debe ser lógicamente accesible al ejecutar una búsqueda que combine el nombre de la tabla, su clave primaria y el nombre de la columna.

Esto significa que dado un nombre de tabla, dado el valor de la clave primaria y dado el nombre de la columna requerida, deberá encontrarse uno y solamente un valor. Por esta razón la definición de claves primarias para todas las tablas es prácticamente obligatoria.

Regla 3: Tratamiento sistemático de los valores nulos

La información inaplicable o faltante puede ser representada a través de valores nulos

Un RDBMS (Sistema Gestor de Bases de Datos Relacionales) debe ser capaz de soportar el uso de valores nulos en el lugar de columnas cuyos valores sean desconocidos.

- Se reconoce la necesidad de la existencia del valor nulo, el cual podría servir para representar, o bien una información desconocida (ejemplo, no se sabe la dirección de un empleado), o bien una información que no procede (a un empleado soltero no se le puede asignar un nombre de esposa). Así mismo, consideremos el caso de un alumno que obtiene 0 puntos en una prueba y el de un alumno que no presentó la prueba.
- Hay problemas para soportar los valores nulos en las operaciones relacionales, especialmente en las operaciones lógicas, para lo cual se considera una lógica trivaluada, con tres (no dos) valores de verdad: verdadero, falso y null. Se crean tablas de verdad para las operaciones lógicas:

```
null AND null = null

Verdadero AND null = null

Falso AND null = Falso

Verdadero OR null = Verdadero, etc.
```

Regla 4: La regla de la descripción de la base de datos

La descripción de la base de datos es almacenada de la misma manera que los datos ordinarios, esto es, en tablas y columnas, y debe ser accesible a los usuarios autorizados.

La información de tablas, vistas, permisos de acceso de usuarios autorizados, etc, debe ser almacenada exactamente de la misma manera: En tablas. Estas tablas deben ser accesibles igual que todas las tablas, a través de sentencias de SQL (o similar).

Regla 5: La regla del sub-lenguaje integral

Debe haber al menos un lenguaje que sea integral para soportar la definición de datos, manipulación de datos, definición de vistas, restricciones de integridad, y control de autorizaciones y transacciones.

Esto significa que debe haber por lo menos un lenguaje con una sintaxis bien definida que pueda ser usado para administrar completamente la base de datos.

Regla 6: La regla de la actualización de vistas

Todas las vistas que son teóricamente actualizables, deben ser actualizables por el sistema mismo.

La mayoría de las RDBMS permiten actualizar vistas simples, pero deshabilitan los intentos de actualizar vistas complejas.

Regla 7: La regla de insertar y actualizar

La capacidad de manejar una base de datos con operandos simples se aplica no sólo para la recuperación o consulta de datos, sino también para la inserción, actualización y borrado de datos'.

Esto significa que las cláusulas para leer, escribir, eliminar y agregar registros (SELECT, UPDATE, DELETE e INSERT en SQL) deben estar disponibles y operables, independientemente del tipo de relaciones y restricciones que haya entre las tablas o no.

Regla 8: La regla de independencia física

El acceso de usuarios a la base de datos a través de terminales o programas de aplicación, debe permanecer consistente lógicamente cuando quiera que haya cambios en los datos almacenados, o sean cambiados los métodos de acceso a los datos.

El comportamiento de los programas de aplicación y de la actividad de usuarios vía terminales debería ser predecible basados en la definición lógica de la base de datos, y éste comportamiento debería permanecer inalterado, independientemente de los cambios en la definición física de ésta.

Regla 9: La regla de independencia lógica

Los programas de aplicación y las actividades de acceso por terminal deben permanecer lógicamente inalteradas cuando quiera que se hagan cambios (según los permisos asignados) en las tablas de la base de datos.

La independencia lógica de los datos especifica que los programas de aplicación y las actividades de terminal deben ser independientes de la estructura lógica, por lo tanto los cambios en la estructura lógica no deben alterar o modificar estos programas de aplicación.

Regla 10: La regla de la independencia de la integridad

Todas las restricciones de integridad deben ser definibles en los datos, y almacenables en el catálogo, no en el programa de aplicación.

Las reglas de integridad

1. Ningún componente de una clave primaria puede tener valores en blanco o nulos (ésta es la norma básica de integridad).
2. Para cada valor de clave foránea deberá existir un valor de clave primaria concordante. La combinación de estas reglas aseguran que haya integridad referencial.

Regla 11: La regla de la distribución

El sistema debe poseer un lenguaje de datos que pueda soportar que la base de datos esté distribuida físicamente en distintos lugares sin que esto afecte o altere a los programas de aplicación.

El soporte para bases de datos distribuidas significa que una colección arbitraria de relaciones, bases de datos corriendo en una mezcla de distintas máquinas y distintos sistemas operativos y que esté conectada por una variedad de redes, pueda funcionar como si estuviera disponible como en una única base de datos en una sola máquina.

Regla 12: Regla de la no-subversión

Si el sistema tiene lenguajes de bajo nivel, estos lenguajes de ninguna manera pueden ser usados para violar la integridad de las reglas y restricciones expresadas en un lenguaje de alto nivel (como SQL).

Algunos productos solamente construyen una interfaz relacional para sus bases de datos No relacionales, lo que hace posible la subversión (violación) de las restricciones de integridad. Esto no debe ser permitido.