GENERALITAT VALENCIANA
Conselleria d'Educació, Investigació, Cultura i Esport

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE

IES ANTONIO SEQUEROS ALMORADÍ

Unión Europea
Fondo Social Europeo
El FSE invierte en tu futuro

U.T. 3: Bases de Datos Relacionales.

I.E.S. ANTONIO SEQUEROS

ADMINISTR. DE SISTEMAS INFORMÁTICOS EN RED

GESTIÓN DE BASES DE DATOS

CURSO 2022/2023

Índice

1. INTRODUCCIÓN: EL MODELO PROPUESTO POR CODD.....1
1.1. LAS 12 REGLAS DE CODD.....2
2. ESTRUCTURA DE UNA B.D. RELACIONAL.....3
2.1. RELACIÓN Y TUPLA.....3
2.2. DOMINIO Y ATRIBUTO.....5
2.3. RESTRICCIONES DEL MODELO.....6
2.4.ESQUEMAS RELACIONALES.....12
2.5. EL MODELO RELACIONAL Y LA ARQUITECTURA ANSI.....13

Bibliografía

DE MIGUEL y PIATTINI (1999). *Fundamentos y Modelos de Bases de Datos*. Ed. Ra-Ma
DE MIGUEL y PIATTINI (1993). *Concepción y Diseño de Bases de Datos*. Ed. Ra-Ma
KORTH Y SILBERSCHATZ(1998). *Fundamentos de Bases de Datos.* Ed. Mc Graw-Hill.

1. INTRODUCCIÓN: EL MODELO PROPUESTO POR CODD.

Las desventajas de los modelos jerárquicos y en red, condujeron a un intenso interés en el nuevo modelo de datos *relacional* propuesto por Codd en 1970. Este modelo constituye un intento de simplificar las estructuras de bases de datos.

A finales de los años sesenta, Codd introdujo la teoría matemática de las relaciones en el campo de las bases de datos, suministrando un sólido fundamento teórico para el desarrollo de nuevos sistemas, dentro de este enfoque relacional.

El documento de Codd propone un modelo de datos basado en la teoría *de las* relaciones, en donde los datos se estructuran lógicamente en forma de relaciones -tablas-, siendo un objetivo fundamental del modelo mantener la independencia de esta estructura lógica respecto al modo de almacenamiento y a otras características de tipo físico.

El trabajo publicado por Codd presentaba un nuevo modelo de datos que perseguía una serie de objetivos, muchos de ellos comunes a otros modelos, que se pueden resumir en los siguientes:

- **Independencia física:** El modo en que se almacenan los datos no debe influir en su manipulación lógica y, por tanto. los usuarios que acceden a esos datos no han de modificar sus programas por cambios en el almacenamiento físico.
- **Independencia lógica:** Añadir, eliminar o modificar cualquier elemento de la base de datos no debe repercutir en los programas y/o usuarios que están accediendo a subconjuntos parciales de los mismos (vistas).
- **Flexibilidad:** En el sentido de poder ofrecer a cada usuario los datos de la forma que este prefiera.
- **Uniformidad:** Las estructuras lógicas de los datos presentan un aspecto uniforme (tablas), lo que facilita la concepción y manipulación de la base de datos por parte de los usuarios.
- **Sencillez:** Las características anteriores. así como unos lenguajes de usuario muy sencillos, producen como resultado que el modelo de datos relacional sea fácil de comprender y de utilizar por parte del usuario final.

Es necesario insistir en la importancia que Codd concede al tema de la independencia de la representación lógica de los datos respecto a su almacenamiento interno (independencia de ordenación, independencia de indización e independencia de los caminos de acceso).

Para conseguir los objetivos citados, Codd introduce el concepto de relación -tabla- como estructura básica del modelo. Todos los datos de una base de datos se representan en forma de relaciones cuyo contenido varía en el tiempo. Una relación, en terminología relacional, es un conjunto de filas –tuplas- con unas determinadas características.

Con respecto a la parte dinámica del modelo, se propone un conjunto de operadores que se aplican a las relaciones. Algunos de estos operadores son clásicos de la teoría de conjuntos - no hay que olvidar que una relación se define matemáticamente como un conjunto -, mientras que otros fueron introducidos específicamente para el modelo relacional. Todos ellos conforman el *álgebra relacional* definida formalmente por

Codd (1972) donde además se compara el álgebra relacional con el *cálculo relacional*, otro lenguaje también propuesto por Codd (1970).

La *teoría de la normalización*, cuyas tres primeras formas normales fueron introducidas por Codd desde sus primeros trabajos. elimina dependencias entre atributos que originan anomalías en la actualización de la base de datos y proporciona una estructura más regular en la representación de relaciones, constituyendo el soporte para el diseño de bases de datos relacionales.

1.1. LAS 12 REGLAS DE CODD.

Ante la discusión sobre el mejor modelo de datos, y la proliferación de SGBD comerciales en los que se incluían características relacionales en productos no relacionales en su origen (sistemas renacidos, en palabras de Codd),impulsan a Codd (1985) a publicar sus famosas 12 Reglas que debe cumplir un SGBD para poder denominarse relacional y con las que evaluó algunos de estos sistemas comerciales.

1. **Representación de la información:** Toda información en una base de datos relacional debe representarse explícitamente a nivel lógico, y de manera única, por medio de valores en tablas.
2. **Acceso garantizado:** Todo dato (valor atómico) debe ser accesible mediante una combinación de un nombre de tabla, un valor de su clave primaria y el nombre de una columna.
3. **Tratamiento de valores nulos:** Los valores nulos (distinto de la cadena de caracteres vacía o de una cadena de caracteres en blanco y distinta del cero o de cualquier otro número) se soportan en los SGBD completamente relacionales para representar la falta de información y la información inaplicable de un modo sistemático e independiente del tipo de datos.
4. **Diccionario de datos basado en el modelo relacional: (LDD)** La descripción de la base de datos (metadatos) se representa a nivel lógico del mismo modo que los datos ordinarios, de forma que los usuarios autorizados puedan aplicar el mismo lenguaje relacional que aplican a los datos normales.
5. **Sublenguaje de datos completo:** Debe existir al menos un lenguaje que permita un completo manejo de la base de datos (definición de datos, definición de vistas, manipulación de datos-interactiva y por programa-, restricciones de integridad, autorizaciones y gestión de transacciones-comienzo, cumplimentación y vuelta atrás-).
6. **Actualización de vistas**: Toda vista teóricamente actualizable debe poder ser actualizada por el sistema.
7. **Inserciones, modificaciones y eliminaciones de alto nivel:** Todas las operaciones de manipulación de datos (consulta, inserción, modificación y borrado) deben operar sobre conjuntos de filas (lenguaje no navegacional). Los sistemas existentes hasta el momento en el que surge el modelo relacional actuaban registro a registro obligando al programador de una base de datos a *navegar* por la misma.

8. **Independencia física de los datos:** Los programas de aplicación y las actividades terminales de los usuarios, es decir, el acceso lógico a los datos permanece inalterado incluso cuando cambien los métodos de acceso o la forma de almacenamiento.
9. **Independencia lógica de los datos:** Los programas de aplicación y las actividades terminales de los usuarios, no deben verse afectados por cambios realizados en las tablas que estén permitidos teóricamente y que preserven la información.
10. **Independencia de la integridad**: Las reglas de integridad de una base de datos deben ser definibles por medio del sublenguaje de datos relacional y habrán de almacenarse en el diccionario de datos de la base de datos, no en los programas de aplicación.
11. **Independencia de la distribución:** Debe existir un sublenguaje de datos que pueda soportar bases de datos distribuidas sin alterar los programas de aplicación cuando se distribuyan los datos por primera vez o se redistribuyan éstos posteriormente.
12. **Regla de la no subversión**: Si un SGBD soporta un lenguaje de bajo nivel (un solo registro cada vez), éste no puede utilizarse para saltarse las reglas de integridad y las restricciones expresadas por medio del lenguaje de más alto nivel (múltiples registros a la vez).

Hay que advertir que no todas las reglas tienen igual importancia, y que los SGBD suelen cumplir, como máximo, 8 de ellas. Según Codd, sólo si se satisfacen las 12 reglas se aprovecharían todos los beneficios potenciales que ofrece el modelo relacional.

2. ESTRUCTURA DE UNA B.D. RELACIONAL.

2.1. RELACIÓN Y TUPLA.

La relación es el elemento básico del modelo relacional, y se puede representar como una *tabla*. En ella podemos distinguir un conjunto de columnas, denominadas *atributos*, que representan propiedades de la misma y que están caracterizadas por un nombre, y un conjunto de filas llamadas *tuplas*, que son las ocurrencias de la relación. El número de filas de una relación se denomina *cardinalidad*, mientras que el número de columnas es el *grado*. Existen también *dominios* de donde los atributos toman sus valores.

Se vuelve a insistir en que una relación se puede representar en forma de tabla, pero que tiene una serie de elementos característicos que la diferencian del concepto típico de tabla:

No puede haber filas duplicadas, es decir, todas las tuplas tiene que ser distintas.

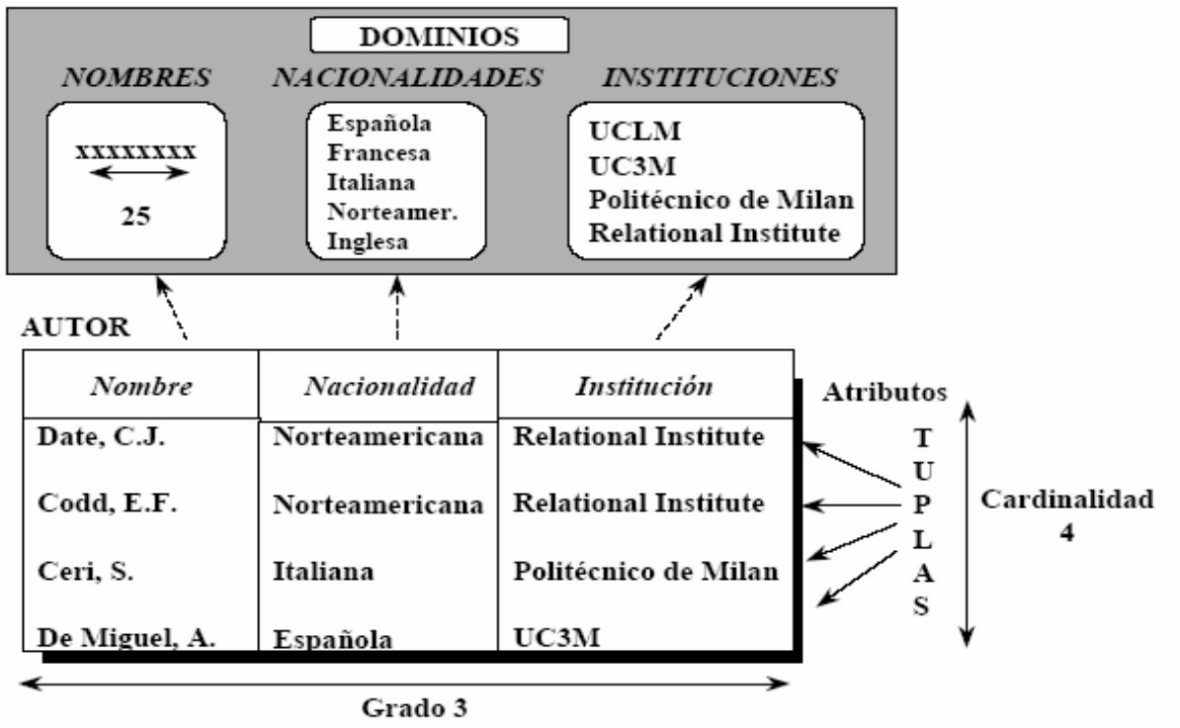
El orden de las filas es irrelevante.

La tabla es plana, es decir, en el cruce de una fila y una columna sólo puede haber un valor (no se admiten atributos multivaluados).

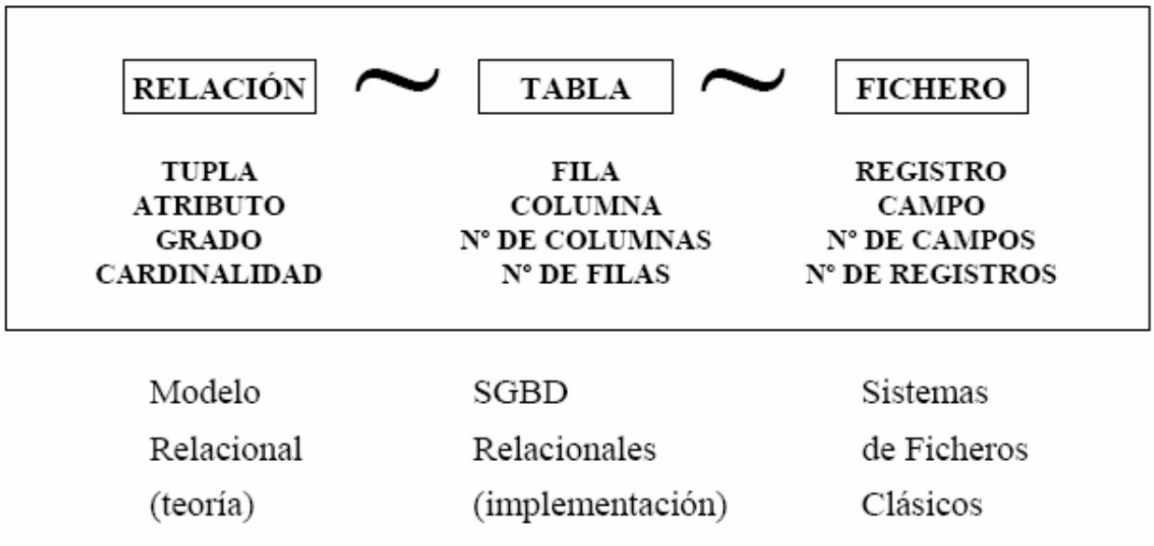
Se trata de restricciones inherentes al modelo que más adelante se analiza.

Una relación siempre tiene un nombre, y en ella es posible distinguir una cabecera (*esquema de relación o intensión*) que define la estructura de la tabla; es decir, sus atributos con los dominios subyacentes, y un cuerpo, *extensión*, que está formado por un conjunto de tuplas que varían en el tiempo.

Esta representación de la relación como una tabla ha sido el origen de que los productos relacionales y los usuarios utilicen habitualmente el nombre de tabla (en principio ajeno a la teoría relacional) para denominar las relaciones y , como consecuencia de ellos, se llame filas a las tuplas y columnas a los atributos.



Si hacemos una comparación de terminología entre relación, tabla y fichero obtenemos:



2.2. DOMINIO Y ATRIBUTO.

Un *dominio* D es un conjunto finito de valores homogéneos y atómicos V_1, V_2,.. V_n, caracterizado por un nombre. Por ejemplo, el dominio *sexo* tiene los valores: *varón, mujer*, que, como ocurre con cualquier conjunto son todos del mismo tipo.

Todo dominio ha de tener un nombre, por el cual nos podemos referir a él, y un tipo de datos; así el tipo de datos del dominio *sexo* es una tira de caracteres de longitud cinco. También se le puede asociar una unidad de medida, como metros, kilos, etc. y ciertas restricciones (como un rango de valores).

Los dominios pueden definirse por intensión o por extensión. Por ejemplo, el dominio *edad* de los alumnos se define por intensión como entero de longitud 3 comprendido entre 12 y 130, mientras que una posible definición por extensión del dominio *día semana* puede ser {lunes, martes, miércoles, jueves, viernes, sábado, domingo}.

El universo del discurso de una base de datos relacional, representado por U, está compuesto por un conjunto finito y no vacío de atributos, A_1 , A_2 , ... , A_n estructurados en relaciones; cada atributo toma sus valores de un único dominio y varios atributos pueden tener el mismo dominio.

Es habitual dar el mismo nombre al atributo y al dominio. En el caso de que sean varios los atributos de una misma tabla definidos sobre el mismo dominio, habrá que darles nombres distintos, ya que una tabla no puede tener dos atributos con el mismo nombre (aunque sí pueden existir en tablas distintas).

Además de los dominios y atributos simples introducidos inicialmente (aquellos que presentan valores atómicos e indivisibles), en posteriores trabajos se introdujo el concepto de *dominio compuesto*.

Un dominio compuesto se puede definir como una combinación de dominios simples que tiene un nombre y a la que se pueden aplicar ciertas restricciones de integridad. Por ejemplo, un usuario puede necesitar manejar, además de los tres dominios *Día, Mes y Año*, un domino compuesto denominado *Fecha* que sería la combinación de los tres primeros, y al que podríamos aplicar las adecuadas restricciones de integridad a fin de que no aparecieran valores no válidos para la fecha; algo análogo ocurre con el nombre y los apellidos, que , según las aplicaciones, puede ser conveniente tratar en conjunto o por separado.

De la misma forma, es posible definir un atributo compuesto *Fecha* que tomaría sus valores del dominio compuesto de igual nombre.

Tanto los atributos compuestos como los dominios compuestos pueden ser tratados, si así lo precisa el usuario, como piezas únicas de información, es decir, como valores atómicos.

2.3. RESTRICCIONES DEL MODELO.

En el modelo relacional, al igual que en otros modelos, existen restricciones, es decir, estructuras u ocurrencias no permitidas, siendo preciso distinguir entre restricciones inherentes y restricciones de usuario. Los datos almacenados en la base han de adaptarse a las estructuras impuestas por el modelo (por ejemplo, no tener tuplas duplicadas) y han de cumplir las restricciones de usuario para constituir una ocurrencia válida del esquema.

Aunque este punto se va a estudiar también en el tema dedicado al diseño de bases de datos relacionales, lo abordamos ahora con un tratamiento ligeramente distinto al del tema citado.

Para ello dividiremos el punto en tres subapartados, en los cuales veremos en primer lugar una definición de los distintos tipos de clave que nos podemos encontrar al trabajar con el modelo relacional. En segundo lugar observaremos las restricciones que se desprenden de la propia definición del modelo, Por último trabajaremos con las restricciones del usuario (también llamadas semánticas) en las que nos centraremos en la integridad referencial y todo lo que lleva asociado.

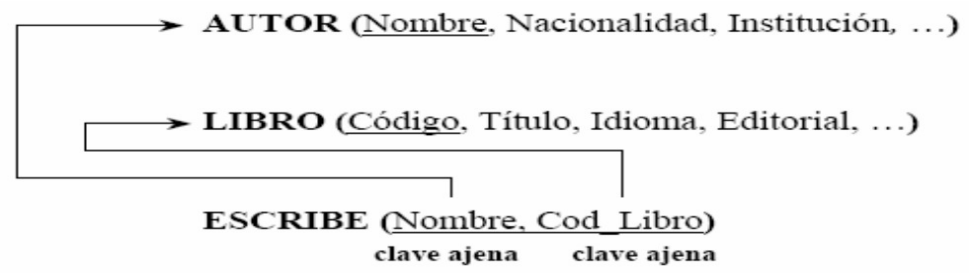
2.3.1. INTRODUCCIÓN AL CONCEPTO DE CLAVE.

Una ***clave candidata*** de una relación es un conjunto no vacío de atributos que identifican unívoca y mínimamente cada tupla. Por la propia definición de relación, siempre hay, al menos, una clave candidata, ya que al ser cada relación un conjunto, no pueden existir dos tuplas repetidas y, por tanto, el conjunto de todos los atributos identificará unívocamente a las tuplas.

En la definición de una clave candidata, si no se cumpliera la condición de minimalidad se eliminarían aquellos atributos que lo impidiesen. Una relación puede tener más de una clave candidata, entre las cuales se debe distinguir:

- ***Clave primaria*** es aquella clave candidata que el usuario escogerá, por consideraciones ajenas al modelo relacional, para identificar las tuplas de la relación.
- ***Claves alternativas*** son aquellas claves candidatas que no han sido escogidas como clave primaria.

Se denomina ***clave ajena*** de una relación R_2 a un conjunto no vacío de atributos cuyos valores han de coincidir con los valores de la clave primaria de una relación R_1 (R_1 y R_2 pueden ser la misma relación). Cabe destacar que la clave ajena y la correspondiente clave primaria han de estar definidas sobre los mismos dominios.



2.3.2. RESTRICCIONES INHERENTES AL MODELO RELACIONAL.

En el modelo relacional cabe mencionar como restricciones inherentes, además de las derivadas de la definición matemática de relación, la *integridad de entidad*.

- De la definición matemática de relación se deducen inmediatamente una serie de características propias de una relación que se han de cumplir obligatoriamente, por lo que se trata de restricciones inherentes y que, como ya se ha señalado, diferencian una relación de una tabla:
  - No hay dos tuplas iguales.
  - El orden de las tuplas no es significativo.
  - El orden de los atributos (columnas) no es significativo.
  - Cada atributo sólo puede tomar un único valor del dominio, no admitiéndose por tanto los atributos multivaluados.
- La regla de integridad de entidad establece que: "Ningún atributo que forme parte de la clave primaria de una relación puede tomar un valor nulo"; esto es, un valor desconocido o inexistente. Esta restricción debería aplicarse también a las claves alternativas, pero el modelo no lo exige.

2.3.3. RESTRICCIONES DE USUARIO O SEMANTICAS.

Podemos considerar la restricción de usuario, dentro del contexto relacional, como un predicado definido sobre un conjunto de atributos, de tuplas o de dominios, que debe ser verificado por los correspondientes objetos para que estos constituyan una ocurrencia válida del esquema.

Otra definición podría ser: son facilidades que el modelo ofrece a los usuarios para que puedan reflejar en el esquema, lo más fielmente posible, la semántica del mundo real.

Los tipos de restricciones permitidos en el Modelo Relacional son:

- Clave Primaria (PRIMARY KEY),
- Unicidad (UNIQUE),
- Obligatoriedad (NOT NULL),
- Integridad Referencial (FOREIGN KEY),
- Restricciones de Rechazo :
  - Verificación (CHECK), y
  - Aserción (ASSERTION).
- Disparador (trigger).
- Dependencia (se estudiará en otro tema).

A continuación se ven todos los tipos de restricciones nombrados anteriormente excepto el último de ellos:

- ***Clave primaria (PRIMARY KEY)***. Permite declarar un atributo o un conjunto de atributos como clave primaria de una relación, por lo que sus valores no se podrán repetir ni se admitirán los nulos (o valores "ausentes"). La obligatoriedad de la clave primaria es una restricción inherente del modelo relacional; sin embargo, la declaración de un atributo como clave primaria de una relación es una restricción semántica que responde a la necesidad del usuario de imponer que los valores del conjunto de atributos que constituyen la clave primaria no se repitan en la relación ni tampoco tomen valores nulos.

  Debemos distinguir entre la restricción inherente de obligatoriedad de la clave primaria y la restricción semántica que le permite al usuario indicar qué atributos forman parte de la clave primaria.

- ***Unicidad (UNIQUE):*** Los valores de un conjunto de atributos (uno o más) no pueden repetirse en una relación. Esta restricción permite la definición de claves alternativas .

- ***Obligatoriedad (NOT NULL):*** Los valores de un conjunto de atributos (uno o más) no admite valores nulos.

- Dentro de las restricciones de usuario destaca la ***restricción de integridad referencial (FOREIGN KEY)*** que se expresa de la siguiente manera:"Si una relación R_2 (relación que referencia) tiene un descriptor (subconjunto de atributos) que es la clave primaria de la relación R_1 (relación referenciada y no necesariamente diferente de R_2), todo valor de dicho descriptor debe concordar con un valor de la clave primaria de R_1, o ser nulo".

  El descriptor es, por tanto, una clave ajena de la relación R_2. También podemos inferir que la clave ajena puede ser parte (o la totalidad) de la clave primaria de R_2 .Por último, deducimos que la clave ajena puede admitir nulos o tener restricción de obligatoriedad (NOT NULL).

  En la figura que aparece en el comienzo de la página siguiente podemos observar un ejemplo de integridad referencial entre el atributo *Editorial* de la relación *LIBRO* (relación que referencia) y el atributo *Nombre_e* de la relación *EDITORIAL* (relación referenciada) cumpliéndose que *Nombre_e* es la clave primaria de la relación *EDITORIAL* (cumpliéndose lo establecido en la definición de restricción de integridad referencial)

EDITORIAL

Nombre_e	*Dirección*	*País*	*Ciudad*
Universal Books	Brown Sq. 23	EEUU	Los Angeles
Rama	Canillas, 144	España	Madrid
Mc Graw-Hill	Basauri 17	España	Madrid
Paraninfo	Virtudes 7	España	Madrid

LIBRO

Código	*Título*	...	*Editorial*
00345D7	Int. Artificial		Paraninfo
1022305	Concep. y Dis.		Rama
4939H2	Turbo C++		Mc Graw-Hill
0045307	Virus Informát.		Nulo
01123J3	Sist. Informac.		Rama

Ejemplo de Integridad Referencial

La integridad referencial está plenamente vinculada con el concepto de *interrelación* del modelo Entidad-Relación de Chen. De hecho, la forma en que el modelo relacional implementa las interrelaciones es a través de la propagación de las claves principales o alternativas como claves ajenas.

Además de definir las claves ajenas, hay que determinar las consecuencias que pueden tener ciertas operaciones (borrado y modificación) realizadas sobre tuplas de las relación referenciada; se pueden distinguir, en principio, las siguientes opciones:

- ***Operación restringida (no action)***: Esto es, el borrado o la modificación de tuplas de la relación que contiene la clave primaria referenciada sólo se permite si no existen tuplas con dicha clave en la relación que contiene la clave ajena. Esto nos llevaría, por ejemplo, a que para poder borrar un cliente de nuestra base de datos no tendría que haber ninguna factura de dicho cliente (suponiendo que CLIENTE y FACTURA son dos tablas a las que pertenecen las respectivas tuplas de cada tipo), en caso contrario el sistema impediría el borrado.
- ***Operación con transmisión en cascada (cascade)***: Esto es, el borrado o modificación de tuplas de la relación que contiene la clave primaria referenciada lleva consigo el borrado o modificación en cascada de las tuplas de la relación que contiene la clave ajena. En nuestro ejemplo, equivaldría a decir que al modificar el código de un cliente en la relación CLIENTE (suponiendo que dicho código fuese la clave referenciada), se tendría que modificar también el código en todas las facturas de nuestra base de datos realizadas al cliente.

- ***Operación con puesta a nulos (set null)***: Esto es, el borrado o modificación de tuplas de la relación que contiene la clave primaria referenciada lleva consigo poner a nulos los valores de las claves ajenas de la relación que referencia. Esto nos llevaría a que cuando se borra un cliente, a las facturas que se han generado para dicho cliente y que se encuentran en la relación FACTURA se les coloque el atributo código del cliente a nulos. Esta opción, obviamente, sólo es posible cuando el atributo que es clave ajena admite el valor nulo.
- ***Operación con puesta a valor por defecto (set default)***: Esto es, el borrado o modificación de tuplas de la relación que contiene la clave primaria referenciada lleva consigo poner un valor por defecto a la clave ajena de la relación que referencia; valor por defecto que debe haber sido definido al crear la tabla correspondiente.

La opción seleccionada en caso de borrado es independiente de la de modificación, es decir, cada uno tomará una de las cuatro opciones por separado.

A continuación vemos un ejemplo de SQL con las restricciones semánticas marcadas en negrita:

```
CREATE TABLE editorial(
        nombre_e CHAR(20) PRIMARY KEY,
        dirección CHAR(50) NOT NULL,
        ciudad CHAR(15),
        pais CHAR(15));

CREATE TABLE libro(
        código CHAR(3),
        título CHAR(50) UNIQUE,
        idioma CHAR(25),
        nombre_e CHAR(20),
        PRIMARY KEY (código),
        FOREING KEY (nombre_e) REFERENCES editorial
                ON DELETE SET NULL,
                ON UPDATE CASCADE);
```

Restricciones de Rechazo: El usuario formula una condición mediante un predicado definido sobre un conjunto de atributos, tuplas o dominios, que debe ser verificado en toda operación de actualización para que el nuevo estado constituya una ocurrencia válida del esquema. Existen dos clases, la verificación (CHECK) y la aserción (ASSERTION):

Verificación (CHECK): Comprueba, en toda operación de actualización, si el predicado es cierto o falso y, en el segundo caso, rechaza la operación. La restricción de verificación se define sobre un único elemento (dentro de un CREATE TABLE) y puede o no tener nombre. Por ejemplo:

CHECK (N_HORAS > 30) en Tabla CURSO_DOCTORADO

- **Aserción (ASSERTION)**: Actúa de forma idéntica a la anterior, pero se diferencia de ella en que puede afectar a varios elementos (por ejemplo, a dos tablas distintas). Por tanto, su definición no va unida a la de un determinado elemento del esquema y siempre ha de tener un nombre. La aserción es un elemento más del esquema. Por ejemplo:

```
CREATE ASSERTION CONCEDE_ SOLICITA AS
CHECK (SELECT Cod_ Estudiante, Cod_ Beca FROM CONCEDE) IN
(SELECT Cod_Estudiante, Cod_Beca FROM SOLICITA));
```

Disparador (trigger): Restricción en la que el usuario pueda especificar libremente la respuesta (acción) ante una determinada condición. Así como las anteriores reglas de integridad son declarativas, los disparadores son procedimentales, siendo preciso que el usuario escriba el procedimiento que ha de aplicarse en caso de que se cumpla la condición. Por ejemplo: si una beca es solicitada por más de 50 alumnos, se introduce un texto en una tabla de mensajes para que la persona que gestiona las becas considere si es necesario ofrecer más becas;

```
CREATE TRIGGER Comprobar_Matriculados
AFTER INSERT ON SOLICITA
DECLARE
NUM_SOLICITUDES Number;
BEGIN
SELECT COUNT(*) INTO NUM_SOLICITUDES FROM SOLICITA;
IF NUM_SOLICITUDES > 50 THEN
INSERT INTO MENSAJES VALUES (' Hay más de 50 solicitudes');
END IF;
END Comprobar_Matriculados;
```

2.3.4. VALORES NULOS.

Valor nulo: Señal utilizada para representar información desconocida, inaplicable, inexistente, no válida, no proporcionada, indefinida, etc. Es una marca que indica que el dato asociado a un atributo de una entidad es desconocido en un momento dado.

Vamos a ver en que se fundamenta la necesidad de los valores nulos en BD:

- ✓ Crear tuplas (filas) con ciertos atributos cuyo valor es desconocido en ese momento, por ejemplo el año de edición de un libro.
- ✓ Añadir un nuevo atributo a una relación existente; atributo que, en el momento de añadirse, no tendría ningún valor para las tuplas de la relación.
- ✓ Atributos inaplicables a ciertas tuplas, por ejemplo, la editorial para un artículo (ya que un artículo no tiene editorial) o la profesión de un menor.

El tratamiento de valores nulos exige redefinir las operaciones de comparación, aritméticas, algebraicas y de agregación de forma específica para el caso en que un operando tome valor nulo. Para ello usamos la **lógica trivaluada**, en la cual se utilizan tres valores: C (Cierto), F (Falso) y Q (Quizás).

Además, se incluyen nuevos operadores:

- ✓ **MAYBE** (ES POSIBLE): devuelve cierto si el argumento vale quizás y falso en otro caso.
- ✓ **IS NULL** (ES NULO), que toma el valor cierto si el operando es nulo y falso en caso contrario.

Se redefinen las operaciones aritméticas: se considera *nulo* el resultado de una suma, resta, multiplicación o división si alguno de los operandos toma valor nulo.

En cuanto a la presencia de valores nulos en claves, etc. se ha de cumplir lo siguiente:

- ✓ Una clave candidata no puede presentar valores nulos (este es el motivo por el que en muchos SGBDR es innecesario acompañar una sentencia PRIMARY KEY o UNIQUE con el cualificador NOT NULL).
- ✓ Un atributo utilizado para el cómputo de un índice sí puede presentar valores nulos aunque se haya definido como único.
- ✓ Una clave ajena puede contener valores nulos.

En cualquier caso la recomendación a aplicar es ser muy restrictivos a la hora de permitir la presencia de nulos limitando esta posibilidad preferentemente a los casos en los que la semántica del universo del discurso pueda presentar valores de este tipo (por ejemplo claves ajenas que aún no han podido cambiar, fechas que todavía no tienen sentido,...).

2.4.ESQUEMAS RELACIONALES.

Ahora podemos dar una definición más completa de ***esquema de una relación*** como R(A:D,S) siendo R el nombre de la relación, A la lista de atributos, D los dominios sobre los que están definidos los atributos, y S las restricciones de integridad intraelementos (afectan a atributos y/ o tuplas de una única relación).

Y el ***esquema de una base de datos relacional*** será definido como E < $\{R_i\}$, $\{I_i\}$ > siendo E el nombre del esquema relacional, { R_i} el conjunto de esquemas de relación, y $\{I_i\}$ el conjunto de restricciones de integridad interelementos (afectan a más de una relación y/ o dominio).

En términos de implementación en SQL, un esquema E tendrá la siguiente forma E <R, D, T, V> siendo R el conjunto de esquemas de relación (CREATE TABLE), D el conjunto de definiciones de dominios (CREATE DOMAIN), T el conjunto de restricciones interrelación y sobre dominios (CREATE ASSERTION, CREATE TRIGGER, ...), y V el conjunto de vistas (CREATE VIEW).

Se puede definir una Base de Datos Relacional como una ocurrencia válida de un esquema relacional.

2.5. EL MODELO RELACIONAL Y LA ARQUITECTURA ANSI.

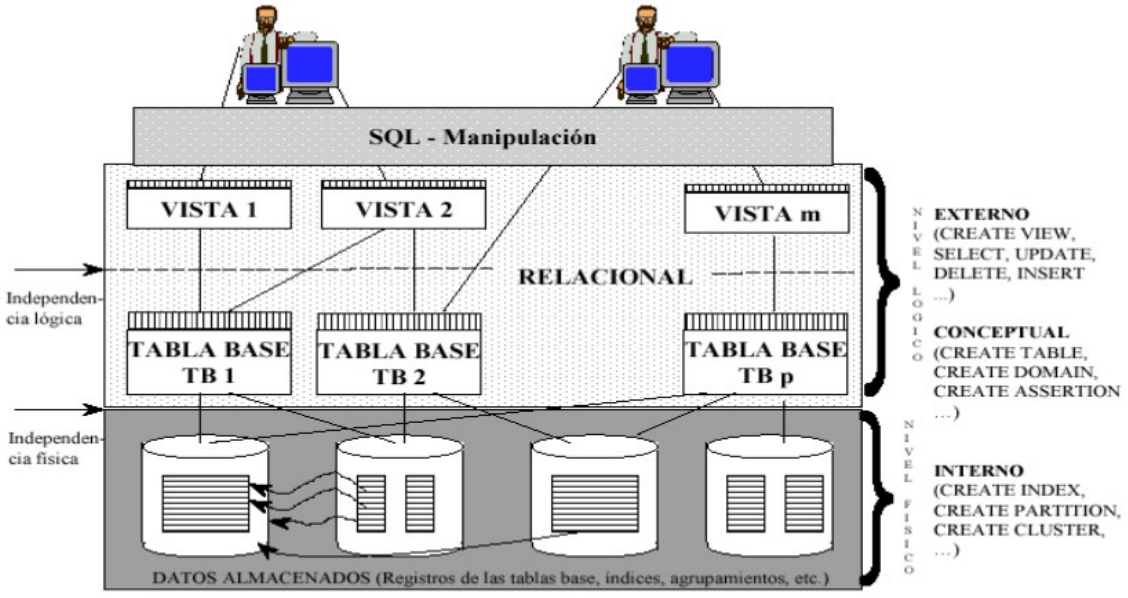
El modelo relacional puede examinarse en el marco de la arquitectura ANSI a tres niveles tal y como se explica en el capítulo sobre SGBD.

Todos los objetos que se han visto hasta el momento, esto es, los dominios, relaciones, claves y restricciones constituyen el esquema conceptual de la arquitectura ANSI. Las relaciones que podríamos llamar primarias (cuyo esquema se define en SQL mediante una sentencia CREATE TABLE) se denominan *tablas base o reales*, ya que tienen una representación directa en el almacenamiento interno.

Existe otro tipo de tablas, denominadas *tablas virtuales o vistas*, que se definen sobre una o más tablas base. Las vistas son ventanas sobre tablas reales, de las que sólo se almacena su definición, y no tienen, por tanto, representación directa en el almacenamiento; equivalen al esquema externo de la arquitectura ANSI o a los subesquemas del modelo Codasyl.

Por lo que respecta al esquema interno, el modelo relacional no especifica absolutamente nada puesto que se trata de un modelo de nivel lógico. En general, cada registro del esquema interno se corresponde con una tupla de las relaciones base, pero no tendría por qué haber una correspondencia biunívoca ya que un registro podría estar constituido por varias tuplas de distintas relaciones, o viceversa, esto es, una tupla podría dividirse en varios registros. Además, cada producto ofrece los elementos internos, como índices, agrupamientos (clusters), tablas hash, particiones, etc., que el diseñador considera más oportunos con el fin de mejorar el rendimiento del sistema.

En la figura que aparece a continuación se da una visión global del modelo relacional dentro del marco definido por la arquitectura ANSI, mostrando además ciertas sentencias del lenguaje SQL que ayudan a definir los elementos correspondientes en los tres niveles. Las sentencias que atañen al nivel interno variarán de un producto a otro y no están estandarizadas.



Vemos, por tanto, que el modelo relacional teórico se adapta bastante bien a la arquitectura ANSI, con las siguientes excepciones:

- ✓ Si dispone de las oportunas autorizaciones, a cualquier usuario se le permite "ver", tanto las relaciones base como las vistas, mientras que en la arquitectura ANSI para un usuario la base de datos está limitada al esquema externo –vistas- ya que el esquema conceptual –relaciones base- son exclusivamente responsabilidad del administrador y sólo pueden ser definidas y manejadas por éste.
- ✓ Aunque las vistas se corresponden con los esquemas externos de ANSI, en el modelo relacional no todas las vistas son actualizables.

En la práctica, muchos productos no responden a la arquitectura a tres niveles, ya que las definiciones del esquema conceptual y del esquema interno no están claramente diferenciadas.

EJERCICIOS DE REPASO

1. Elige una regla de Codd y explica en qué consiste.
2. Explica mediante un ejemplo los términos cardinalidad y grado.
3. Compara la terminología del modelo relacional, los sistemas gestores de bases de datos relacionales y los sistemas de ficheros clásicos.
4. ¿Qué relación existe entre dominio y atributo?
5. ¿De qué formas puede definirse un dominio? Pon un ejemplo de cada tipo.
6. ¿Qué diferencia existe entre claves candidatas, primaria y alternativas?
7. Pon dos ejemplos de claves ajenas.
8. ¿Qué relación existe entre la restricción PRIMARY KEY y la combinación de las restricciones UNIQUE y NOT NULL?
9. ¿Qué modificadores para los borrados y las modificaciones se pueden poner en una restricción de integridad referencial?
10. ¿Cuáles son las Restricciones de Rechazo y qué diferencia existe entre ellas?
11. ¿Qué es un TRIGGER y para qué sirve?
12. ¿Cuáles son las restricciones inherentes al modelo relacional?
13. ¿Qué diferencia existe entre las tablas reales y las vistas?
14. Asocia las siguientes instrucciones SQL con el nivel de la arquitectura ANSI correspondiente: CREATE VIEW, CREATE INDEX, CREATE TABLE, SELECT, UPDATE, DELETE e INSERT.