Modelo Relacional

Introducción

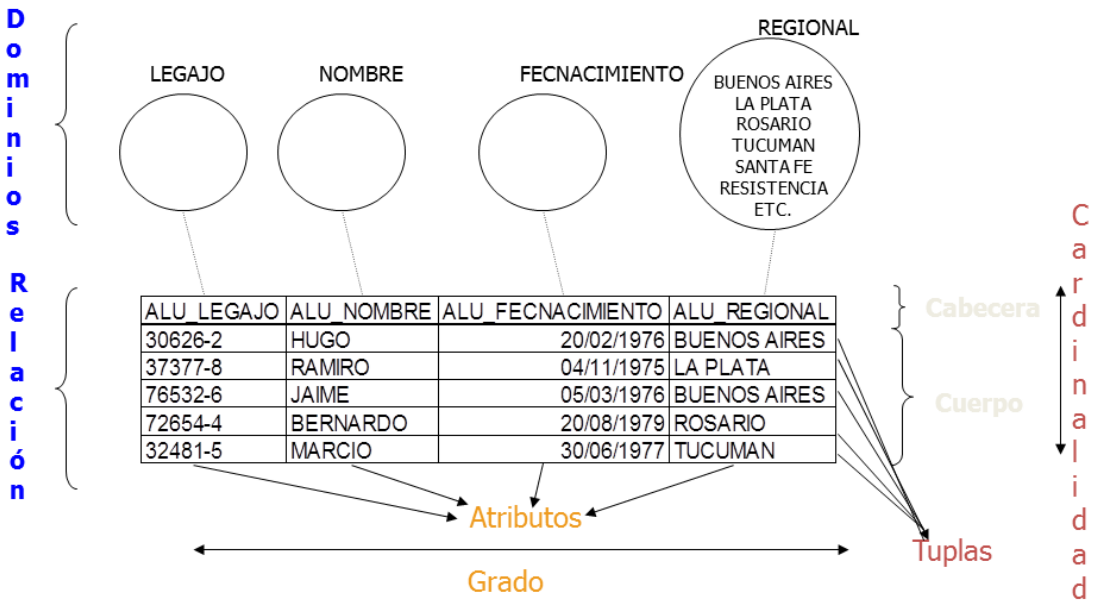
El modelo relacional es actualmente el modelo de datos de mayor uso en las organizaciones.

El Dr. Edgar F. Codd en el año 1970 publicó un paper conocido con el título “A Relational model of Data for Large Shared Data Banks” (“Un modelo relacional de datos para grandes bancos de datos compartidos”), este documento planteaba las bases del modelo relacional.
El modelo relacional logró la adhesión inicial de otros expertos, porque apuntaba a resolver el problema de grandes bases de datos compartidas.

El concepto fundamental en el que se basa el modelo relacional es que los datos se representan de una sola manera, como una estructura tabular, conformada por filas y columnas.

A esta estructura se la definió formalmente como relación, aunque de manera informal y en sus implementaciones se la conoce con el nombre de tabla.

El siguiente gráfico es una ayuda para la comprensión de la estructura de una relación.



Término relacional	“Equivalente” informal
Relación	Tabla
Tupla	Fila o registro
Cardinalidad	Numero de filas
Atributo	Columna o campo
Grado	Número de columnas
Clave primaria	Identificador único o clave primaria también

Dominio	Conjunto de valores posibles, etc.

Dominios

Una colección de valores, de los cuales los atributos obtienen sus valores reales.
Los dominios son la menor unidad de semántica de información desde el punto de vista del modelo, son atómicos, o sea que no se pueden "descomponer". Por supuesto que al decir "desde el punto de vista modelo", quiere decir que no es válido descomponer, por ejemplo un número de legajo válido en dígitos decimales. En este caso se pierde el significado.
En el ejemplo del punto anterior, el dominio "NOMBRE" es el conjunto de todos los nombres posibles, por lo que el conjunto de valores que aparecen en el atributo "ALU_NOMBRE" de la relación "ALUMNO", es un subconjunto del dominio sobre el cual se define.

En consecuencia un dominio es un conjunto de valores escalares, todos del mismo tipo.
Observación: como nulo no es un valor, los dominios no contienen nulos. Una relación con "null" en uno de sus atributos (o sea una tabla con un campo NULL), implica que la relación representada por esa tupla no involucra al dominio sobre el cual se carece valor.
Importancia de los dominios: Los dominios nos permiten restringir las comparaciones.

Por ejemplo, supongamos las siguientes consultas SQL:

```
SELECT ALUMNO.NOMBRE
  FROM ALUMNO, CURSO, MATERIA
 WHERE CURSO.ALU_LEGAJO = ALUMNO.ALU_LEGAJO
   AND CURSO.MAT_CODIGO = MATERIA.MAT_CODIGO
   AND MATERIA.MAT_CODIGO = '85-1346';

SELECT ALUMNO.*
  FROM ALUMNO, CURSO
 WHERE CURSO.MAT_CODIGO = ALUMNO.ALU_LEGAJO
   AND CURSO.COD_CURSO = '3052';
```

En el primer ejemplo, la consulta está bien formulada. ¿Por qué?. Pues porque los atributos de las distintas tablas que intervienen en las comparaciones están definidos sobre los mismos dominios subyacentes. Tanto ALU_LEGAJO de la tabla CURSO, como ALU_LEGAJO de la tabla ALUMNO, se definen sobre el dominio "LEGAJO", que representa a todos distintos los valores de legajo posibles. Lo mismo ocurre para los atributos definidos sobre el dominio de códigos de materia. En el segundo ejemplo, la consulta (si bien está sintácticamente bien escrita) no tiene sentido alguno ya que comprar códigos de materias con números de legajos, es decir, atributos definidos sobre dominios distintos.
Sin embargo, los motores de bases de datos relacionales comerciales, al menos los más conocidos hasta el momento, no realizan ningún tipo de chequeos que permitan arrojar al usuario un error del tipo "existe incompatibilidad entre dominios de atributos comparados". Esto se debe a que estos sistemas no incorporan dentro de su funcionalidad el manejo de dominios.

Por supuesto que no siempre es inválida la incompatibilidad de dominios; puede haber casos en los que el usuario requiera una consulta donde los atributos sean en apariencia "incompatibles", pero que realmente sean semánticamente compatibles.
Los chequeos que realizan los motores se refieren a los tipos de datos de los campos comparados. Sin embargo, por lo general se efectúa automáticamente una conversión de alguno de los tipos de datos siempre que sea posible. Por ejemplo al comparar un dato numérico con otro de caracteres, la cadena de caracteres es convertida a su equivalente numérico y a continuación se efectúa la comparación.
Si los motores permitieran incluir definiciones de dominios, y los atributos debieran definirse sobre ellos, esto resultaría especialmente útil, ya que en un sistema estructurado de esta forma, una pregunta del tipo: "¿Cuales relaciones contienen información referente a provincias?" se resolvería mediante una simple consulta al catálogo del sistema.
Resulta de buena práctica imponer el uso de ciertas "convenciones de nombre" para los atributos de las relaciones, de manera que siempre posean el mismo nombre cuando estén definidos sobre el mismo dominio (siempre que sea posible, ya que si existen dos atributos de una misma relación definidos sobre el mismo dominio no se puede utilizar el mismo nombre).

Las últimas versiones de motores me permiten definir "tipos de datos del usuario". Por ejemplo, definir el tipo de datos "COD_PROVINCIA", que es numérico, entero, de dos posiciones, valores definidos entre 1 y 24. Luego al definir los campos en las tablas se puede hacer referencia a estos tipos de dato. Este es el equivalente más cercano al concepto de ***dominio***.

No se debe olvidar que el concepto de dominio es mucho más amplio que el de tipo de datos de un DBMS o de un lenguaje de programación. Ejemplos:
- No todos los operadores de comparación son válidos aunque dos tipos de datos pertenezcan al mismo dominio. Si existen dos atributos que se refieren al dominio "estado civil", tiene sentido preguntar por igual o distinto. No así en el caso de "mayor que" o "menor que".
- Existen reglas semánticas que ante atributos pertenecientes a ciertos dominios operados entre sí arrojan como resultado un valor correspondiente a un tercer dominio. Ej: ACELERACION * TIEMPO = INCREMENTO EN LA VELOCIDAD.

Relaciones

Definición: una relación R se define sobre un conjunto de dominios D_1,D_2, …, D_n y se compone de una *cabecera* y un *cuerpo*.

Cabecera: está formada por un conjunto **fijo** de atributos, o mejor dicho pares atributo-dominio, tal que a cada atributo le corresponde un dominio.
Cabecera = { (A_1:D_1), (A_2:D_2), …, (A_n:D_n). }
Cuerpo: un conjunto de tuplas que varía con el tiempo. Cada tupla a su vez está formada por un 0conjunto de pares atributo-valor:
$Tupla_i$= { (A_1:v_{i1}) … (A_n:v_{in}) } con $1 \leq i \leq m$, siendo m cardinalidad de la relación, n grado de la relación.

En el ejemplo anterior de la tabla de alumnos:
1. La tabla tiene cuatro dominios subyacentes: el dominio de números de legajo (LEGAJO), el de nombres (NOMBRE), el de fechas de nacimiento

(FECNACIMIENTO) y el de facultades regionales (REGIONAL). Siempre que representemos una tabla, aunque no representemos los dominios, debemos entender que lógicamente "están allí".

2. La tabla tiene una fila de cabecera: (ALU_LEGAJO, ALU_NOMBRE, ALU_FECNACIMIENTO y ALU_REGIONAL).
   Esa fila de cabecera representa al conjunto de pares ordenados: (LEGAJO, ALU_LEGAJO) (NOMBRE, ALU_NOMBRE) (FECNACIMIENTO, ALU_FECNACIMIENTO) (REGIONAL, ALU_REGIONAL). El primer componente de cada par ordenado es el nombre del atributo, y el segundo es el nombre del dominio correspondiente sobre el cual está definido.
3. El resto de la tabla esta formado por un conjunto que constituye el cuerpo de la relación y que varía con el tiempo (se insertan, modifican, o eliminan filas). Por lo tanto, el conjunto de filas representados en el figura de la tabla de alumnos, es solo una vista temporal de los valores que aparecen en un momento dado en la base.
   Sea la fila: (30628-2; HUGO; 20/02/1976; BUENOS AIRES)
   En realidad esta fila representa al siguiente conjunto de pares ordenados:
   (ALU_LEGAJO, 30628-2)
   (ALU_NOMBRE, HUGO)
   (ALU_FECNACIMIENTO, 20/02/1976)
   (ALU_REGIONAL, BUENOS AIRES)
   En consecuencia esta fila, al igual que las otras, representa una tupla de la relación.

De 1, 2, y 3 concluimos que la tabla que graficamos en una representación "razonable" de una relación, debido a que convenimos una forma de lectura en la cual pudimos determinar: que existen los dominios subyacentes, que los atributos se definen sobre esos dominios y los pares ordenados constituyen la cabecera, que hay un conjunto de filas que varía con el tiempo que representan a las tuplas de la relación.

Una relación NO ES una tabla. La tabla es una representación de una relación. Es muy ventajoso en términos de uso y comprensión (esta una de las mayores ventajas del modelo relacional) que una relación tenga un equivalente informal tan simple de poner en papel como una "tabla". Sin embargo, una relación no sigue un orden determinado, como por ejemplo lo hace la tabla con sus columnas o sus filas.
Como ya se dijo, el valor n se llama grado (cantidad de atributos de la relación). Los atributos de una relación no varían con el tiempo. Si hacemos un ALTER TABLE en SQL para agregar una columna, modificamos la tabla, pero en lo que se refiere al modelo relacional, implica la creación de una nueva relación de grado n+1 a partir de una de grado n existente.

El valor m, es la cardinalidad de la relación. La cardinalidad es un concepto dinámico, varía con el tiempo. Es por ello básicamente que un dominio y una relación unaria no son la misma cosa.

Propiedades de una Relación

Estas propiedades fijan limitaciones en el modelo y diferencian a la definición formal de Relación con su implementación la tabla.
Fueron definidas por Codd en su publicación A Relational Model of Data for Large Data Banks.

Dada una relación:

1. **No existen en ella tuplas repetidas**
   El cuerpo de una relación es un conjunto y como tal no acepta elementos duplicados. Una tabla en un DBMS podría tener filas duplicadas si no se toman las medidas necesarias al definirlas. Pero se debe recordar que el modelo relacional **no admite filas repetidas**. En consecuencia toda relación tiene clave primaria.

2. **Las tuplas no están ordenadas**
   Los elementos de un conjunto matemático no se encuentran ordenados. No existe el concepto de direccionamiento por posición o por adyacencia (equivalente al acceso directo/acceso secuencial). Las filas de las tablas de los DBMS si tienen un orden, debido a la necesidad de orden físico en disco.

3. **Los atributos no están ordenados**
   La cabecera es un conjunto y como tal no está ordenado. No se puede hablar del "tercer atributo" o del "segundo atributo después de este otro". Las tablas de los DBMS requieren que sus columnas sigan un orden determinado.

4. **Todos los valores de los atributos son atómicos.**
   Como consecuencia de que los atributos están definidos sobre los dominios, y estos a su vez son atómicos, los valores de los atributos serán a su vez atómicos. Visto de manera simple, por cada posición de fila y columna dentro de una tabla, siempre existe un solo valor, nunca una lista de valores. En consecuencia las relaciones no contienen grupos repetitivos, o sea que está “normalizada" (en lo que se conoce como primera forma normal).

Reglas de Codd

Estas reglas definidas por Codd buscan fijar conceptos para asegurar que un “DBMS - Data Base Manager System” (“Sistema de Gestión de Bases de Datos”) fuera relacional (RDBMS – Relational Data Base Manager System).
En 1985 Codd publicó dos artículos en la revista Computer World “Is your DBMS Really Relational?” y “Does your DBMS run by the rules?” en donde escribió las mencionadas reglas.

Regla 0 – Regla de fundación.
Un DBMS debe gestionar sus datos almacenados utilizando solo sus capacidades relacionales.

Regla 1 – Regla de la información.

Toda información en la base de datos tiene que estar representada de una única forma, como valores de una relación.

Regla 2 – Regla de acceso garantizado.
El acceso a cada uno de los datos está garantizado utilizando una combinación del nombre de la relación, valor de clave primaria y nombre de la columna asociada al dato requerido.

Regla 3 – Tratamiento sistemático de valores nulos.
Los valores nulos (ausencia de valor) deben ser soportados por el RDBMS para poder representar la información faltante en una manera sistémica.

Regla 4 – Catálogo dinámico en línea basado en el modelo relacional.
La representación de la base de datos está representada igual que los datos ordinarios, o sea en relaciones tabulares.

Regla 5 – Regla del sublenguaje de datos completa.
Un DBMS puede admitir varios lenguajes, sin embargo debe haber un lenguaje cuyas sentencias puedan expresar por una sintaxis bien definida lo siguiente:

- *Definición de datos*
- *Definición de vista*
- *Manipulación de datos*
- *Restricciones de integridad*
- *Autorización*
- *Límites de la transacción*

Regla 6 – Regla de actualización de vistas.
A través del sistema se deben actualizar todas las vistas que son teóricamente actualizables.

Regla 7 – Inserción, actualización y borrado de alto nivel
La capacidad de manejar una relación base, se aplica no solo a la recuperación de datos, sino también a la inserción, actualización y borrado de datos.

Regla 8 – Independencia física de datos
Los programas de aplicación permanecerán lógicamente inalterables, siempre que se realicen cambios en las representaciones de almacenamiento o de métodos de acceso.

Regla 9 – Independencia lógica de datos
Los programas de aplicación permanecerán lógicamente inalterables, siempre que se realicen cambios en las definiciones de datos, en tanto y en cuanto los cambios preserven la información vital para dichos probramas.

Regla 10 – Independencia de integridad
Las reglas de integridad, específicas de las bases de datos relacionales, se deben poder definir en un sublenguaje de datos y deben ser almacenadas en el catálogo o diccionario de datos, no en los programas de aplicación.

Regla 11 – Independencia de distribución

Los valores nulos (ausencia de valor) deben ser soportados por el RDBMS para poder representar la información faltante en una manera sistémica.

Regla 12 – Regla de no subversión
Los valores nulos (ausencia de valor) deben ser soportados por el RDBMS para poder representar la información faltante en una manera sistémica.

Tipos de Relaciones

Estos son algunos de los tipos de relaciones que manejan los RDBMS.

1. Relaciones base: o relaciones reales, son el ejemplo de las tablas. Dada su importancia en el modelo de datos (debido a su necesidad en las distintas aplicaciones), el diseñador de la base le escogió un nombre y tiene existencia permanente ya que es parte de la base de datos en sí (son los datos).
2. Vistas: Son relaciones virtuales, que se definen en términos de otras relaciones. Tienen un nombre definido.
3. Instantáneas o Snapshots: también son relaciones derivadas, que tienen un nombre. La diferencia es que los datos son almacenados (copiados). La instantánea puede definir también si es actualizable y cada cuanto tiempo.
4. Resultados de consultas (queries). No tienen existencia permanente dentro de la base, pero pueden ser nombradas.
5. Resultados intermedios de consultas (por ejemplo los "subqueries"). No tienen existencia permanente dentro de la base y no pueden ser nombrados.
6. Relaciones temporales: es nombrada, como 1,2, o 3, pero se destruye en forma automática en algún momento apropiado. Por ejemplo, se crea al iniciar una sesión con la base, es utilizada por la aplicación, y después se destruye automáticamente.

Base de Datos Relacional

"Es una base de datos percibida por el usuario como una colección de relaciones normalizadas de diversos grados que varía con el tiempo"
Al decir que es percibida por el usuario implica que todas las ideas deben ser aplicables en los niveles externo y conceptual de la arquitectura, no importa cual fuera la representación interna de los datos. Es importante también tener en cuenta el hecho de que es modificable a lo largo del tiempo, o sea que conforme se modifiquen las tuplas de las relaciones, la BD es la misma; no así al cambiar la definición de un dominio o de una relación.

Atributos Clave

Clave Candidata
Un atributo o conjunto de atributos de una relación pertenece al conjunto de claves candidatas si se cumplen las siguientes condiciones:

- **Unicidad:** Indica la no repetición del valor de un atributo o conj. de atributos en la vida de una relación.

- **Minimalidad:** indica que será la mínima combinación de atributos posibles que cumpla con la condición de unicidad.

Definición

El atributo K, simple o compuesto de una relación R, es clave candidata de R ⇔ se cumplen, independientemente del tiempo, las siguientes propiedades:

a- Unicidad

$\forall\ t_i, t_j \in R / i \neq j \Rightarrow K_i \neq K_j$

b- Minimalidad

Si K es compuesta de r elementos, cualquier clave compuesta de un subconjunto de r-1 atributos de K, perderá la propiedad de unicidad.

Clave Primaria

La clave primaria, pertenece al conjunto de claves candidatas, y es aquella que el diseñador elige para identificar a cada entidad que se almacenará en una base de datos relacional.
La decisión de cuál clave candidata se elegirá como primaria, dependerá del criterio y experiencia del diseñador y del negocio en la cual se utilice la base de datos.

Clave Alterna o Alternativa

Son aquellas claves, pertenecientes al conjunto de claves candidatas, que no se eligieron como primaria.

Ejemplo de claves candidatas, primaria y alternas.

Sea una relación **cliente**, con los siguientes atributos: **cod_cliente, nombre, nro_cuit, tipo_documento, nro_documento, domicilio** y **saldo**. Tanto **cod_cliente** como **nro_cuit** y la cobinación de **tipo_documento y nro_documento** tienen valores únicos para cada cliente.

Entonces, la relación **cliente** tiene varias claves ***candidatas***.

De todas ellas, escojo como clave ***primaria*** *el atributo* ***cod_cliente***, y a las demás las llamamos ***alternas***.

Para entender el concepto de minimalidad, si tomásemos la clave compuesta ***(cod_cliente ; nro_cuit)***, la misma no es candidata, ya que sacando el atributo ***nro_cuit*** de la clave sigo teniendo *unicidad*.

Como ninguna relación acepta "filas" duplicadas, entonces toda relación tiene al menos una clave candidata, de a lo sumo n atributos (n grado de la relación). De allí escogemos una y solo una clave primaria.

Como consecuencia toda relación tiene una y solo una clave primaria. Esta es la clave que cobra importancia para la definición de las reglas de integridad de las entidades. Las demás son solo una consecuencia del método para la elección de una clave primaria. El concepto de clave primaria no implica la existencia de un índice o cualquier otra estructura de acceso a los datos.

Las claves primarias son el único modo garantizado por el sistema de direccionamiento de una fila específica (tupla) dentro de una tabla (relación). Siempre que se solicite a un sistema relacional por las tuplas con un valor de clave primaria determinado, producirá a lo sumo una tupla.

Clave Foránea

La clave foránea es un atributo o combinación de atributos, que permite la combinación de datos de las distintas relaciones del sistema.
Es un atributo o combinación de atributos de la relación R2 cuyos valores deben coincidir con los de la clave primaria de alguna relación R1. El valor de esta clave foránea nos da una referencia a la tupla donde se encuentra el valor de clave primaria. La relación donde se encuentra la clave primaria es la "*relación referenciada*" o "*relación objetivo*".

Definición

El atributo K, simple o compuesto de una relación R_2, es clave foránea de R_2 $\Leftrightarrow$ se cumplen, independientemente del tiempo, las siguientes propiedades:

1. Cada valor K_i correspondiente a la tupla ti, es totalmente nulo o totalmente no nulo (o bien todos los atributos que la componen son nulos, o bien son todos no nulos).
2. $\exists$ R_1 , P / P es clave primaria de R_1 y se satisface: $\forall$ $K_i \in R_2$, Ki no nulo $\exists$ $P_j \in R_1$ / $K_i=P_j$

- La concordancia de las claves primaria en R_1 y foránea en R_2 viene dada al estar definidas sobre el mismo dominio.
- La clave foránea puede o no ser parte de la clave primaria de R_2, no importa para esta definición.
- R_1 y R_2 no son necesariamente distintas. Si $R_1=R_2$, se dice que existe una *relación autorreferencial*.
- Las claves foráneas deben en ciertas ocasiones aceptar nulos. Por ejemplo, una relación autorreferencial para el caso de la primer tupla que ingrese al conjunto.

Ejemplo de Claves Foráneas.

Tomemos como ejemplo un modelo con las siguientes relaciones:

Alumnos

Nro_legajo	Nombre	Fec_nacimiento	Regional
30628-2	HUGO	20/02/1976	BUENOS AIRES
37377-8	RAMIRO	04/11/1975	LA PLATA
76532-6	JAIME	05/03/1976	BUENOS AIRES
72654-4	BERNARDO	20/08/1979	ROSARIO
32481-5	MARCIO	30/06/1977	TUCUMAN

Cursos

Cod_curso	Nro_legajo	Cod_materia
3052	30628-2	85-1346
3052	37377-8	85-4002
3051	76532-6	85-1346
3051	72654-4	85-4002
2053	32481-5	95-2285

Materias

Cod_materia	Nombre	Plan	Especialidad
85-1346	PROGRAMACION I	1985	K
85-4002	ALGEBRA II	1985	K
95-2285	INGENIERIA Y SOCIEDAD	1985	M
95-1003	ALGEBRA Y GEOM. ANALITICA	1985	M

En la relación Cursos se hace evidente que solo se permiten valores para los atributos Nro_legano que se encuentren en la relación Alumnos. De la misma manera, solo se permiten valores para el atributo Cod_Materia de la relación Cursos que existan en la relación Materias. Es absurdo que un alumno se inscriba a una materia que no existe. Estos dos campos son ejemplos de ***claves foráneas***.

Reglas de Integridad en el Modelo Relacional

El modelo relacional define tres reglas de integridad aplicables a cualquier base de datos: **la regla de integridad de las entidades**, **la regla de integridad referencial** y **las reglas de negocios o comerciales**.

Las reglas de integridad formarán parte del modelo representado en la base y como tales deben ser definidas sobre las relaciones base, es decir, aquellas relaciones nombradas de existencia permanente en la base (las tablas).

Regla de Integridad de las Entidades

"Ningún componente de la clave primaria de una relación base puede aceptar nulos"

Nulo es la ausencia de valor para un atributo que pertenezca al conjunto dominio sobre el cual está definido. No es solamente el "NULL". Por ejemplo, si el número de legajo es tipo CHAR, una cadena en blanco es un valor nulo para la presente definición.

La justificación lógica de esta regla de integridad es que: una relación representa a una entidad del mundo real, por lo que una tupla es un elemento de ese conjunto. Las entidades son distinguibles (es decir, se las puede identificar), y las claves primarias son el mecanismo de identificación en el modelo relacional. Un elemento cualquiera que quiero representar en la base tiene una forma de ser identificado en el mundo real; en consecuencia tiene que poseer valores de atributo que lo identifique. Una base de datos relacional no admite registrar información acerca de algo que no se puede identificar.

Regla de Integridad Referencial

"Cada valor de una clave foránea debe existir como vaor en la clave primaria de otra relación o ser desconocido"

La lógica de la regla es simple: si X hace referencia a Y, entonces Y debe existir. La existencia de Y viene dada por la regla de integridad de entidades, que es la existencia de clave primaria.

Mecanismos para la implementación de la integridad referencial

La regla de integridad referencial habla de mantener en un estado consistente de la base de datos. En consecuencia, cualquier operación que no deje la base en estado consistente será incorrecta. Para estas operaciones se deberá definir algún tipo de acción para mantener la integridad de los datos. Por ejemplo, si se quiere eliminar el alumno con nombre "JAIME", ¿qué hacemos con las materias a las que se inscribió?. Una opción será eliminar todos los registros correspondientes en la relación CURSOS. Otra opción es no permitir el borrado.

El analista a cargo del diseño de la base debe definir la forma en la que el DBMS manejará la integridad referencial. Los aspectos a definir son:

a) Si la clave foránea acepta nulos. Por ejemplo una entidad de cursos ya abiertos a inscripción, pero sin docente asignado, podría aceptar nulos en una clave foránea compuesta por el legajo del docente. Por el contrario, en el ejemplo de la entidad CURSOS precedente, no es válido un registro que no posea código de materia correspondiente.
b) La acción a llevar a cabo si se intenta eliminar un registro con una clave primaria referenciada por una clave foránea de otra relación. Existen tres opciones:
    1. RESTRICT: no se permite la eliminación del registro "padre".
    2. CASCADE: se eliminan también los registros que la referencian. Por ejemplo se deja de dar una materia y se eliminan todas las inscripciones de la tabla "hija".

3. ANULACION: se le asigna nulo a todas las claves foráneas (la clave foránea debe permitir nulos)

c) La acción a llevar a cabo si se intenta modificar la clave primaria de un registro referenciado. También existen tres opciones:
   1. RESTRICT: no se permite la modificación del registro "padre".
   2. CASCADE: se modifican también las claves foráneas que la referencian. Por ejemplo se cambia el código de una materia y automáticamente se actualizan los registros de inscripciones.
   3. ANULACION: se le asigna nulo a todas las claves foráneas (la clave foránea debe permitir nulos)

Por último, debemos considerar que los mecanismos presentados son los más comunes: mediante la utilización de TRIGGERS, el diseñador de la base puede realizar operaciones más complejas que verifiquen que la base se encuentre en un estado consistente. O por ejemplo, guardar un log con las modificaciones. Inclusive aquellas que no involucran el concepto de integridad referencial. Por ejemplo, si se intenta eliminar un "item" de una tabla de ítems de facturas, que el campo "total_factura" de la cabecera se recalcule automáticamente.

Reglas de negocios o comerciales

Cada organización define sus reglas de negocio en función a los objetivos que persigue, por lo que son específicas de cada empresa.

Independientemente que en las reglas de negocio se definen en el Modelo de Dominio de la Aplicación, los RDBMS permiten aplicar parte de las misma y de esta manera asegurar que no habrá datos inválidos en la Base de Datos.

ALGEBRA RELACIONAL

Los OCHO operadores relacionales:

Restricción, Select	σ	Unario	Relacional especial	Primitivo
Proyección, Proyect	π	Unario	Relacional especial	Primitivo
Producto, Producto cartesiano	X	Binario	Derivado de conjuntos	Primitivo
Union	U	Binario sobre R compatibles	Derivado de conjuntos	Primitivo
Intersección	$\cap$	Binario sobre R compatibles	Derivado de conjuntos	No primitivo
Diferencia	-	Binario sobre R compatibles	Derivado de conjuntos	Primitivo
Reunión o JOIN	><	Binario, compatibilidad de atributos	Relacional especial	No primitivo
División	%	Binario, compatibilidad de atributos	Relacional especial	No primitivo

Por su utilidad, las operaciones no primitivas conviene manejarlas en forma directa.

Cuando la condición de la relación no es de igualdad, o los campos comparados no se definen sobre el mismo dominio, hablamos de un JOIN THETA o JOIN c/SELECCIÓN.

OPERADORES ADICIONALES

AMPLIACION o EXTEND: efectuar cálculos con los datos
RESUME o SUMARIZACION: (operaciones como SUM, MIN, COUNT, etc)
OUTER JOINS
DIVISION GENERALIZADA