UNIVERSIDAD DE LA EMPRESA

CAPITULO 3
MODELO RELACIONAL DE BASES DE DATOS

Se verá la forma de especificar una base de datos relacional utilizando una definición formal.

3.1 ESQUEMAS DE RELACION

El modelo relacional fue introducido en 1970, por Codd. Desde entonces ha sido usado como base teórica y práctica en estudios, investigaciones e implementaciones de sistemas de bases de datos. Sus ventajas y desventajas han sido discutidas en variadas ocasiones. Vamos a describir formalmente lo que entendemos por relación, sus esquemas y sus instancias. Definiremos el concepto de base de datos relacional y finalmente plantearemos un ejemplo de una base de datos a ser usado en el resto del texto.

Supongamos que deseamos manejar información sobre amigos. Estamos interesados en sus nombres y apellidos, ciudades y países donde viven, y teléfonos. Podemos representar la información mediante una tabla:

Tabla AGENDA DE AMIGOS

APELLIDO	NOMBRE	CIUDAD	PAIS	TELEFONO
Ullman	*Jeffrey*	*Stanford*	*EE.UU.*	*33-1-87655*
Maier	*David*	*Berkeley*	*EE.UU.*	*33-1-67443*
Albano	*Antonio*	*Udine*	*Italia*	*30-1-52108*

El contenido de esta tabla se dice que es una *instancia* de la relación. Esta instancia es única en cada momento pero puede cambiar de un instante para otro. Si tenemos nuevos amigos, serán colocados sus datos en la tabla *AMIGOS* pero no será modificado el tamaño de la tabla "a lo ancho", sus columnas no variarán, sólo sus instancias.

Definición: [Esquema de relación primitivo] Un *esquema de relación primitivo* es una tripla *PRS = (ATR, DMN, dom)* donde

- *ATR* es un conjunto finito de *atributos.* Los atributos son los cabezales de las columnas de la tabla.

- *DMN* es un conjunto finito de *dominios.* Cada dominio es un conjunto de valores los cuales pueden ser infinitos.

- *dom: ATR-->DMN* es una función que asocia a cada atributo un dominio. Para cada atributo solamente los valores del correspondiente dominio pueden aparecer en la columna que es encabezada por ese atributo.

■

Existen además otras características comunes a las instancias de una tabla, como por ejemplo en la tabla AMIGOS:

UNIVERSIDAD DE LA EMPRESA

■ El significado de las filas de la tabla: cada fila representa un amigo, y para cada amigo conocemos su apellido, nombre, ciudad y país donde vive, y su número de teléfono. Pero no conocemos la calle donde vive por ejemplo.

■ La ciudad donde un amigo está viviendo debe corresponder a una ciudad del país donde está viviendo.

■ El prefijo del teléfono debe corresponder con la característica del país y la ciudad donde está viviendo.

Definición: [Esquema relación] Un *esquema relación* es la tripla *RS = (PRS, M, SC)* donde *PRS* es un esquema de relación primitivo, *M* es el *significado* de la relación y *SC* es el *conjunto de restricciones* o condiciones de la relación.

Podemos decir que una *relación* es entonces una quíntupla *RS = (ATR, DMN, dom, M, SC).*
■

Definición: [Esquemas de relación compatibles] Dados dos esquemas de relación *RS1 = (ATR1, DMN1, dom1, M1, SC1)* y *RS2 = (ATR2, DMN2, dom2, M2, SC2),* se dice que son *compatibles* si se cumple que *ATR1 = ATR2, DMN1 = DMN2* y *dom1 = dom2.*
■

En el caso visto, la relación *AMIGOS* es entonces: *AMIGOS = (ATR, DMN, dom, M, SC)* donde tenemos que:

■ *ATR = { APELLIDO, NOMBRE, CIUDAD, PAIS, TELEFONO }*

■ *DMN = {conjunto de apellidos, conjunto de nombres, conjunto de nombres de ciudades, conjunto de nombres de países, conjunto de números válidos de teléfonos}*

■ La función *dom* asocia el conjunto de apellidos con APELLIDO, el conjunto de nombres con NOMBRE, el conjunto de nombres de ciudades con Ciudad, el conjunto de nombres de países con PAIS y el conjunto de números válidos de teléfonos con TELEFONO.

■ *M* es el significado de la relación. Para cada amigo son dados el apellido, el nombre, la ciudad y el país donde vive y su número de teléfono.

■ El conjunto *SC* incluye al menos la condición de que la ciudad donde un amigo vive sea una ciudad del país donde vive, y la condición de que el prefijo del número de teléfono se corresponda con el país y la ciudad.

Ejemplo: Considérese un hotel que desea guardar información sobre sus habitaciones. Por cada habitación del hotel el gerente está interesado en su número, el número de camas que posee, si tiene o no baño privado, en qué piso del hotel se sitúa y por supuesto su precio.

Un esquema de relación que puede ser usado para este propósito es *HABITACIONES = (ATR, DMN, dom, M, SC)* donde

- *ATR = {NUM-HABITACION, CAMAS, BAÑO?, PISO, PRECIO};*
- *DMN = {conjunto de números de habitaciones, conjunto de enteros positivos, conjunto {true, false}, conjunto de números de pisos, conjunto de enteros positivos}*
- *dom* es claro.
- El significado *M* de esta relación es que la misma describe información sobre las habitaciones del hotel. Para cada habitación su número de habitación es dado, junto con su número de camas, si posee o no baño privado, en qué piso del hotel se sitúa y su precio por día.
- *SC* puede incluir diferentes condiciones tales como:

  - Cada habitación posee un número diferente de pieza.

  - Existen solamente 8 pisos y el primer dígito del número de pieza indica el piso en que se sitúa la habitación.

  - Toda habitación del piso 2 tiene baño privado.

  - Una habitación con baño privado cuesta más de U$ 20.

  - Ningún piso tiene más de 20 habitaciones.

■

3.1.1 Instancias de relación

El esquema de relación permite especificar la estructura de la relación. La información real almacenada en la relación se describe mediante *instancias de relación.* Una instancia de relación puede verse como una tabla en la que el orden de los renglones no es importante. Cada renglón tiene varias entradas, una por cada atributo en el esquema, y el valor almacenado en cada entrada debe pertenecer a un dominio adecuado.

Un renglón de la tabla se denomina *tupla.* De hecho puede verse como funciones que asocian cada atributo con su correspondiente valor. Por ejemplo, la primera tupla de la relación *AMIGOS* es una función que asocia "Ullman" con APELLIDO, "Jeffrey" con NOMBRE, etc.

□ **Definición:** [Tupla] Sea el esquema primitivo de relación *PRS = (ATR, DMN, dom)*. Una *tupla* sobre el esquema primitivo de relación *PRS* es una función $t:ATR \rightarrow U_{\delta \in DMN}{}^{\delta}$ que asocia a cada atributo con su correspondiente valor tal que para todo atributo *A e ATR, t(A) e dom(A).* Una *instancia de relación posible* del esquema de relación primitivo *PRS* es un conjunto de tuplas sobre *PRS.*

Como notación usaremos *t[A]* para expresar el valor correspondiente al atributo *A* en una tupla *t.*

■

Definición: [Restricción de relación, instancia de relación] Sea el esquema de relación *RS = (PRS, M, SC).* Una *restricción de relación* del esquema de relación *RS* se representa por una función booleana que asocia con cada instancia de relación posible de *PRS,* el valor *true* o *false.* Si esa función asocia el valor *true,* decimos que la instancia de relación posible *satisface a la restricción de relación.* Una *instancia de relación* del esquema de relación *RS* es una instancia de relación posible de *PRS,* que satisface todas las restricciones de relación de *SC.*

■

Ejemplo: Sea *prs* una instancia de relación posible sobre el esquema de relación *HABITACIONES.* Las restricciones de relación, junto con las funciones booleanas que las representan, son:

■ Cada habitación tiene un número único de habitación.

```
f1(prs) = true
iff
forall t1,t2 in prs holds
        if t1[NUM-HABITACION] = t2[NUM-HABITACION]
        then t1 = t2.
```

■ Hay sólo 8 pisos y el primer dígito del número de habitación indica el número de piso.

```
f2(prs) = true
iff
forall t in prs holds
        1 <= t[PISO] <= 8 and
        FirstDigit(t[NUM-HABITACION]) = t[PISO].
```

■ Cada habitación del piso 2 tiene baño.

```
f3(prs) = true
iff
forall t in prs holds
        if t[PISO] = 2
        then t[BANO?] = true.
```

■ Una habitación con baño privado cuesta más de U$ 25.

```
f4(prs) = true
iff
```

UNIVERSIDAD DE LA EMPRESA

```
forall t in prs holds
        if t[BANO?] = true
        then t[PRECIO] > 25.
```

■ Ningún piso tiene más de 20 habitaciones.

```
f5(prs) = true
iff
forall nropiso, 1 <= nropiso <= 8 holds
        Card( {t in prs | t[PISO] = nropiso} ) <= 20.
```

■

Ejemplo: Las siguientes son, respectivamente, una instancia válida y una instancia no válida de la relación *HABITACIONES.*

NUM-HABITACION	CAMAS	BAÑO?	PISO	PRECIO
201	*3*	*true*	*2*	*30*
202	*2*	*true*	*2*	*32*
203	*2*	*true*	*2*	*25*
303	*1*	*false*	*3*	*15*

NUM-HABITACION	CAMAS	BAÑO?	PISO	PRECIO
201	*1*	*false*	*2*	*10*
302	*2*	*true*	*3*	*20*
505	*2*	*true*	*3*	*25*

La tabla previa *no es una instancia válida* de la relación *HABITACIONES* por las siguientes razones:

■ La última fila dice que hay una habitación con número 505 en el piso 3, lo que viola la segunda restricción de integridad.

■ La primera fila dice que hay una habitación en el piso 2 sin baño. Esto viola la tercera restricción de integridad.

■ La segunda fila dice que hay una habitación con baño privado por solamente U$ 20. Esto viola la cuarta restricción de integridad.

UNIVERSIDAD DE LA EMPRESA

3.1.2 Esquemas e instancias de bases de datos

Definición: [Esquema primitivo de una base de datos] Sea *PDS = { RSi = (ATRi, DMNi, domi, Mi, SCi) | i ∈ I },* donde *RSi* son esquemas de relación tales que para todo *i,j ∈ I*, y *A ∈ ATR, A e ATRi ∩ ATRj* implica que *domi(A) = domj(A).* Todas las funciones *domi* pueden ser extendidas a una función *dom:* $U_{i \in I}$ *ATRi --> DMNi.*
■
Observaciones: Esta definición nos coloca en el supuesto de que si el mismo atributo es usado en dos esquemas de relaciones diferentes dentro del mismo esquema de base de datos, entonces estos dos atributos, deben tener el mismo significado (semántica) y por lo tanto deben tener el mismo dominio asociado.
■

Definición: [Esquema de la base de datos] El *esquema de la base de datos* es la tripla *DS = (PDS, DM, SDC)* siendo *PDS* un esquema primitivo de la base de datos, *DM* el significado de la base de datos y *SDC* el conjunto de las restricciones de la base de datos.
■

Definición: [Instancia posible de una base de datos] Una *instancia posible de una base de datos* es el conjunto *pds* de instancias de relaciones tal que hay exactamente una instancia de relación en *pds* por cada esquema en el esquema primitivo de la base de datos (PDS).
■

Definición: [Restricciones de la base de datos] Las restricciones de la base de datos son representadas por funciones booleanas que asocian *true* o *false* a cada posible instancia de la base de datos. Si asocia *true* decimos que la instancia de la base de datos satisface las restricciones.

3.2 ESQUEMAS DINAMICOS

Un esquema de relación tiene en cada momento una única instancia de relación. Esta instancia cambia regularmente. Si queremos seguir la pista de estos cambios o evoluciones, debemos considerar la secuencia de instancias consecutivas asociadas con el esquema de la relación en el transcurso del tiempo. Podría suceder que esta secuencia deba cumplir ciertas condiciones extras: condiciones que deben verificarse sobre la actualización de cada instancia. En el ejemplo del hotel podríamos pedir que nunca un baño sea removido de una habitación. Por lo tanto no podría haber instancias consecutivas donde una habitación posee baño en la primer instancia y no lo posee en la segunda instancia de la misma. Este tipo de restricciones son llamadas *restricciones dinámicas* de las relaciones. Formalmente ellas son funciones booleanas sobre el conjunto de secuencias de instancias del esquema de la relación.

UNIVERSIDAD DE LA EMPRESA

Definición: [Esquema de relación dinámico] Un *esquema de relación dinámico* es el par ordenado *DYDS = (RS, SDYC)* donde *RS* es un esquema de relación y *SDYC* es un *conjunto de restricciones dinámicas de la relación.* Cada una de estas restricciones está representada por una función que asocia *true* o *false* con cada secuencia de instancias de la relación RS.

Decimos que la secuencia de instancias de la relación *satisface una restricción dinámica de la relación* si la función que la representa asocia *true* para esa secuencia.
■

Definición: [Evolución] La *evolución de un esquema dinámico de relación* es una secuencia *rs0, rs1, rs2, ...* de instancias de la relación para las cuales cada restricción dinámica de la relación en SDYC es satisfecha.
■

Definición: [Esquema dinámico de una base datos] Un *esquema dinámico de una base de datos* es un par ordenado *DYDS = (DS,SDYDC)* donde *DS* es el esquema de la base de datos y *SDYDC* es el conjunto de las restricciones dinámicas de la base de datos. Cada una de estas restricciones está representada por una función que asocia *true* o *false* con cada secuencia de instancias de la base de datos *DS.*

Decimos que la secuencia de instancias de la base de datos satisface una restricción dinámica de la base de datos si la función que la representa retorna *true* para esa secuencia.
■

Definición: [Evolución de un esquema dinámico de la base de datos] La *evolución de un esquema dinámico de la base de datos* es una secuencia *ds0, ds1, ds2, ...* de instancias de la base de datos para las cuales cada restricción dinámica de la base de datos en *SDYDC* es satisfecha.
■

Ejemplo: En el ejemplo del hotel, vamos a suponer que no es posible remover un baño de una habitación, aunque las habitaciones pueden ser anuladas. El esquema dinámico de la relación es *DHABIT = (HABITACIONES, SDYC)* donde *SDYC* consiste solamente de la restricción dinámica que dice que ningún baño puede ser removido de una habitación. Esta restricción es representada por una función booleana *NoRemover.*

La función *NoRemover:DS x DS --> {true,false}* se aplica a dos instancias de la base de datos *ds1* y *ds2.* Esta función retorna *true* cuando para cada tupla de *HABITACIONES* en la instancia *ds1*, que tenga baño, la tupla correspondiente (si existe) en la instancia *ds2* también tiene baño.
Vemos que *NoRemover* retorna *false* para la siguiente secuencia de instancias de la base de datos, debido a que la habitación *101* en la primera instancia tiene baño y en la siguiente no lo tiene.

NUM-HABITACION	CAMAS	BAÑO?	PISO	PRECIO
101	*2*	*true*	*1*	*25*
201	*3*	*true*	*2*	*30*
202	*2*	*true*	*2*	*32*
203	*2*	*true*	*2*	*25*
303	*1*	*false*	*3*	*15*

NUM-HABITACION	CAMAS	BAÑO?	PISO	PRECIO
101	*2*	*false*	*1*	*25*
201	*3*	*true*	*2*	*30*
202	*2*	*true*	*2*	*32*
203	*2*	*true*	*2*	*15*
303	*1*	*false*	*3*	*25*

■

3.2.1 Clasificación de restricciones

Llegados a este punto podemos entonces clasificar las restricciones en distintos tipos:

- restricciones dinámicas de la base de datos
- restricciones dinámicas de la relación
- restricciones de la base de datos
- restricciones de la relación
- restricciones de la tupla

En forma genérica, vamos a denominar a todas estas restricciones como *restricciones de integridad.*

Las restricciones de relación determinan cuáles conjuntos de tuplas son instancias válidas de la relación. Para determinar ésto debemos ver la relación en su totalidad. Algunas veces es posible verificar tupla por tupla, en este caso decimos que es una *restricción de tupla.* Del ejemplo del hotel, las restricciones de relación:

- hay solamente 8 pisos y el primer digito del número de habitación indica el piso,
- toda habitacion del piso 2 tiene baño, y
- una habitacion con baño privado cuesta más de U$ 20,

son restricciones de tupla.

Evidentemente, toda restricción de tupla es una restricción de relación. Más aún, toda restricción de relación de un esquema relación definido en un esquema de base de datos, es equivalente a una restricción de base de datos sobre dicho esquema de base de datos.

La restricción de relación sólo especifica las instancias de relación de la relación dada, y no dice nada acerca de las instancias de relación de las demás relaciones de la base de datos.

En resumen, las restricciones de tupla son equivalentes a un subconjunto de las restricciones de relación; y éstas son equivalentes a un subconjunto de las restricciones dinámicas de la relación y a otro subconjunto de las restricciones de la base de datos. Las restricciones de la base de datos son equivalentes a un subconjunto de las restricciones dinámicas de la base de datos.

3.3 EL HOTEL GUILLERMO TELL

En esta sección se plantea en forma completa y detallada a la base de datos utilizada en el Hotel Guillermo Tell.

El esquema de la base de datos es *HOTELDB = (SRS, DM, SDC),* siendo *SRS = {HABITACIONES, MUCAMAS, VISITANTES, ESTADIA, CUENTA-TELEFONO, EMPLEADOS}.* El significado *DM* expresa que todas las relaciones involucradas en la base de datos, contienen información concerniente al mismo hotel.

SDC es un conjunto conteniendo 5 restricciones:

- Hay una mucama responsable por cada habitación del hotel.
- Las habitaciones donde permanecen los visitantes son habitaciones del hotel.
- Los números de habitaciones que aparecen en la relación *CUENTA-TELEFONO* son números de habitaciones válidas del hotel.
- Si hay una llamada telefónica de una habitación, entonces esa habitación está ocupada en esa fecha.
- Cada mucama en la instancia de la relación *MUCAMAS* es un empleado en la instancia de *EMPLEADOS,* cuyo trabajo es "mucama."

El conjunto de las restricciones de integridad dinámicas contiene a las siguientes restricciones:

UNIVERSIDAD DE LA EMPRESA

- No debe ser borrado un visitante si hay todavía información sobre él o ella en la relación *ESTADIA.*

- Todas las cuentas de teléfono de un visitante deben ser pagadas cuando el visitante abandona el hotel.

- Un baño no debe ser removido de una habitación, sin embargo las habitaciones pueden ser anuladas.

- Solamente la estadía con fecha de partida más vieja puede ser removida de la relación *ESTADIA.*

A continuación se describe cada una de las relaciones de la base de datos.

HABITACIONES = (ATR_R, DMN_R, domR, MR, SCR)

- *ATR_ R = {NUM-HABITACION, CAMAS, BAÑO?, PISO, PRECIO}*

- *DMN_ R = {conjunto de número de habitaciones, conjunto de enteros positivos, {true,false}, conjunto de números de pisos, conjuntos de enteros (precios)}*

- *domR* es obvio.

- *MR* es el significado de la relación.

SCR puede incluir diferentes condiciones como por ejemplo:

- Cada habitación tiene un único número de habitación.

- Hay solamente 8 pisos y el primer dígito del número de la habitación indica el piso.

- Cada habitación del piso 2 tiene baño privado.

- Una habitación con baño privado cuesta más de U$ 20.

- Ningún piso tiene más de 20 habitaciones.

MUCAMAS = (ATR_M, DMN_M, domM, MM, SCM)

- *ATR_M = {NUM-MUCAMA, NUM-HABITACION}*

- *DMN_M = {conjunto de números de empleados, conjunto de números de habitaciones}*

- *domM* es obvio.

UDE
UNIVERSIDAD DE LA EMPRESA

- *MM*: cada tupla de la relación indica el número de empleado de una mucama y el número de habitacion que está a su cargo.

Las únicas dos restricciones de integridad de la relación en *SCM* son: cada mucama es responsable por cuatro habitaciones y no hay dos mucamas que sean responsables por la misma habitación.

VISITANTES = (ATR_V, DMN_V, domV, MV, SCV)

- *ATR_V = {NUM-VIS, NOMBRE-VIS, DIRECC-VIS, CIUDAD-VIS, PAIS-VIS}*

- *DMN_V* y *domV* son obvios.

- *MV* indica el número de visitante con su nombre, dirección, ciudad y país de procedencia.

SCV contiene dos restricciones de integridad, expresando que cada visitante tiene un número diferente y que si dos visitantes provienen de la misma ciudad entonces provienen del mismo país.

ESTADIA = (ATR_E, DMN_E, domE, ME, SCE)

- *ATR_E = {NUMERO-VIS, FECHA-ARRIBO, FECHA-PARTIDA, HABIT-ESTADIA, Número-ACOMPAÑANTES, CUENTA}*

- *DMN_E* y *domE* son obvios.

- *ME* indica el número de visitantes con sus fechas de arribo, fecha de partida, la habitación que ocuparon durante su estadía, el número de acompañantes y la cuenta que pagaron. Si la estadía no ha terminado aún el valor de la fecha de partida y de la cuenta a pagar son "desconocidos."

SCE contiene dos restricciones de integridad indicando que:

- La fecha de partida es siempre posterior a la fecha de arribo de un visitante.

- Un visitante no puede volver a arribar al hotel mientras esté vigente su estadía en el hotel.

CUENTA-TELEFONO = (ATR_T, DMN_T, domT, MT, SCT)

- *ATR_T = {NUM-HABITACION, FECHA, HORA, DESTINO, CUENTA, PAGA?}*

- *DMN_T* y *domT* son obvios.

UNIVERSIDAD DE LA EMPRESA

- *MT* indica las llamadas telefónicas para una habitación, en una fecha y hora determinadas, a un destino y cuál es el valor de la cuenta. Se indica también si la cuenta ya fue pagada o no.

SCT contiene solamente una restricción de integridad: no existen dos tuplas diferentes que sean iguales en el número de la habitación, la fecha y la hora juntas.

EMPLEADOS = (ATR_EMP, DMN_EMP, domEMP, MEMP, SCEMP)

- *ATR_EMP = { Número-EMPL, NOMBRE-EMPL, TRABAJO, SALARIO}*
- *DMN_EMP* y *domEMP* son obvios.
- *MEMP* indica los empleados del hotel con sus números, nombre, trabajo y sueldo.

SCEMP sólo contiene una única restricción: cada empleado del hotel tiene un número único de empleado.

UNIVERSIDAD DE LA EMPRESA

3.4 BASE DE DATOS ACMEBD

CLIENTES = (ATR_CLI, DMN_CLI, domCLI, MCLI, SCCLI)

■ *ATR_CLI = { NRCLIENT, NOMBRE, APELLIDO, COMPAÑIA, DIRECC1, DIRECC2, CIUDAD, ESTADO, CODPOST, TELEFONO, FOTO}*

■ *DMN_CLI = { números, strings, fotografías}*

■ *dom_CLI = { NRCLIENT → números, NOMBRE → strings, APELLIDO → strings, COMPAÑIA → strings, DIRECC1→ strings, DIRECC2→ strings, CIUDAD → strings, ESTADO → strings, CODPOST → strings, TELEFONO → strings, FOTO → fotografía }*

■ *MCLI* Este esquema mantiene los datos de los clientes. Los elementos de *ATR_CLI* tienen el siguiente significado:

NRCLIENT	: Número identificatorio del cliente.
NOMBRE	: Nombre propio del cliente.
APELLIDO	: Apellido del cliente.
COMPAÑIA	: Nombre de la compañia del cliente.
DIRECC1	: Dirección del cliente (primera línea).
DIRECC2	: Dirección del cliente (segunda línea).
CIUDAD	: Nombre de la ciudad del cliente.
ESTADO	: Código del estado al que pertenece la ciudad.
CODPOST	: Código postal del cliente.
TELEFONO	: Teléfono del cliente.
FOTO	: Fotografía del cliente o logotipo de su compañia.

UNIVERSIDAD DE LA EMPRESA

■ *SCCLI*

NRCLIENT : Es único.
NOMBRE, APELLIDO y COMPAÑIA : No pueden ser nulos

NOTASCLI = (ATR_NCLI, DMN_NCLI, domNCLI, MNCLI, SCNCLI)

■ *ATR_NCLI = { NRCLIENT, TEXTONOT}*

■ *DMN_NCLI = { números, texto}*

■ *dom_NCLI = { NRCLIENT → números, TEXTONOT → texto }*

■ *MNCLI* Este esquema mantiene el detalle de los clientes. Contiene varias anotaciones realizadas sobre el cliente. Los elementos de *ATR_CLI* tienen el siguiente significado:

NRCLIENT : Número del cliente a quien corresponde la anotación realizada.
TEXTONOT : Texto de la anotación realizada.

■ *SCNCLI {∅}*

FABRICNT = (ATR_FAB, DMN_FAB, domFAB, MFAB, SCFAB)

■ *ATR_FAB = { CODFABR, NOMFABR}*

■ *DMN_FAB = { números, texto}*

■ *dom_FAB = { CODFABR → números, NOMFABR → texto }*

■ *MFAB* Este esquema mantiene los datos de fabricantes de los artículos. Los elementos de *ATR_FAB* tienen el siguiente significado:

CODFABR : Código que identifica al fabricante.
NOMFABR : Nombre del fabricante.

■ *SCFAB*

CODFABR es único y no puede ser nulo
NOMFABR no puede ser nulo

ARTICS = (ATR_ART, DMN_ART, domART, MART, SCART)

■ *ATR_ART = { NRORDEN, NRSTOCK, CODFABR, CANTIDAD, PRECIO}*

■ *DMN_ART = { números, strings}*

■ dom*_ART = { NRORDEN → números, NRSTOCK → números, CODFABR → strings,CANTIDAD → números , PRECIO → números }*

■ *MART* Este esquema mantiene las líneas de detalle de cada orden que pertenece al esquema ORDENES u ORDENESC. Los elementos de *ATR_ART* tienen el siguiente significado:

NRORDEN : Número de la orden a la que pertenece la línea de detalle.
NRSTOCK : Número de stock del artículo ordenado.
CODFABR : Código de fabricante del artículo.
CANTIDAD : Cantidad ordenada del artículo.
PRECIO : Precio del artículo ordenado.

■ *SCART*

NRORDEN, NRSTOCK,CODFABR : Son únicos en conjunto.
CANTIDAD, PRECIO : No pueden ser nulos.
PRECIO se calcula *CANTIDAD* por el precio unitario (*PRUNIDAD*) disponible en esquema stock.

STOCK = (ATR_STO, DMN_ STO, dom STO, M STO, SC STO)

■ *ATR_STO = {NRSTOCK, CODFABR, DESCRIP, PRUNIDAD}*

■ *DMN_STO = { números, strings}*

■ *dom_STO = { NRSTOCK → números, CODFABR → strings, DESCRIP → strings, PRUNIDAD → números }*

■ *MSTO* Este esquema mantiene las líneas de detalle de cada orden que pertenece al esquema ORDENES u ORDENESC. Los elementos de *ATR_STO* tienen el siguiente significado:

NRSTOCK : Número de stock del artículo.
CODFABR : Código de fabricante del artículo.
DESCRIP : Descripción del artículo.
PRUNIDAD : Precio unitario del artículo en U$.

■ *SCSTO*

NRSTOCK y CODFABR son únicas a la vez.
DESCRIP no puede ser nulo.
PRUNIDAD no puede ser nulo.

ORDENES = (ATR_ORD, DMN_ORD, domORD, MORD, SCORD)

■ *ATR_ORD = { NRORDEN, FECHAORD, NCLIENT, CLIEPO, FECHAENV, TASAENV, FECHAPAG, MONTO }*

■ *DMN_ORD = { números, fechas, strings}*

■ dom_*ORD* = { *NRORDEN* → *números, FECHAORD* → *fechas, NCLIENT* → *números, CLIEPO* → *strings, FECHAENV* → *fechas, TASAENV* → *números , FECHAPAG* → *fechas, MONTO* → *números* }

■ *MORD* Este esquema corresponde al registro de todas las órdenes de artículos realizados por los clientes. Por cada orden, hay varias líneas de detalle, una por artículo ordenado, en el esquema *ARTICS*. Cuando una orden es pagada, la tupla correspondiente debe de sacarse de ORDENES y pasarse a ORDENESC. Los elementos de *ATR_ORD* tienen el siguiente significado:

NRORDEN : Número identificatorio de la orden
FECHAORD : Fecha de ingreso de la orden
NCLIENT : Número del cliente que hizo la orden
CLIEPO : Código que indica la forma de pago de la orden
FECHAENV : Fecha de envío de los artículos ordenados, es nula si aún no se ha realizado el envío.
TASAENV : Tarifa de envío en U$.
FECHAPAG : Fecha en la que el cliente realizó el pago, es nula si la orden aún no se ha realizado.
MONTO : Valor total de la orden (sin tasa de envío) obtenido de la suma de los productos de precios unitariospor cantidad de los artículos incluídos en la orden, más la aplicación de impuestos agregados al valor.

■ *SCORD*

NRORDEN : es único.
FECHAORD <= *FECHAENV*.
FECHAORD <= *FECHAPAG*.
FECHAORD, NCLIENT, MONTO : no puede ser nulo

ORDENESC = (ATR_ORDC, DMN_ORDC, domORDC, MORDC, SCORDC)

■ *ATR_ORDC = { NRORDEN, FECHAORD, NCLIENT, CLIEPO, FECHAENV, TASAENV, FECHAPAG, MONTO }*

■ *DMN_ORDC = { números, fechas, strings}*

■ dom_*ORDC = { NRORDEN → números, FECHAORD → fechas, NCLIENT → números, CLIEPO → strings, FECHAENV → fechas, TASAENV → números , FECHAPAG → fechas, MONTO → números }*

■ *MORDC* Este esquema corresponde al registro de todas las órdenes de artículos realizados por los clientes y que ya están pagas. Los elementos de *ATR_ORDC* tienen el siguiente significado:

NRORDEN : Número identificatorio de la orden
FECHAORD : Fecha de ingreso de la orden
NCLIENT : Número del cliente que hizo la orden
CLIEPO : Código que indica la forma de pago de la orden
FECHAENV : Fecha de envío de los artículos ordenados, es nula si aún no se ha realizado el envío.
TASAENV : Tarifa de envío en U$.
FECHAPAG : Fecha en la que el cliente realizó el pago, es nula si la orden aún no se ha realizado.
MONTO : Valor total de la orden (sin tasa de envío) obtenido de la suma de los productos de precios unitarios por cantidad de los artículos incluídos en la orden, más la aplicación de impuestos agregados al valor.

■ *SCORDC* igual a *SCORD*

3.5 NOCION MATEMATICA DEL MODELO RELACIONAL

El *modelo de datos relacional* está basado en el concepto matemático de *relación* utilizado en la teoría de conjuntos. En esta teoría, una *relación* es un subconjunto de un producto cartesiano de una lista de *dominios.* Un *dominio* es simplemente un conjunto de valores. El producto cartesiano de los dominios *D1, D2, ..., Dk* (que se escribe como *D1 x D2 x ... x Dk*), es el conjunto de todas las *k-tuplas* <*v1, v2, ..., vk*> tales que *vi* $\in$ *Di,* para todo *i=1...k.* El valor de *k* debe ser mayor que 0.

Una *relación* (en sentido matemático) es cualquier subconjunto no estricto del producto cartesiano de uno o más dominios. En lo que respecta a las bases de datos, asumiremos que una relación es *finita.* Es decir, una instancia de una relación tiene una cantidad finita de tuplas.

Cada relación subconjunto del producto cartesiano *D1 x D2 x ... x Dk* se dice que tiene *aridad o grado k.* Una tupla <*v1, v2, ...,vk*> tiene *k* componentes; la i-sima componente es v_i.

Podemos ver a una relación como una *tabla* donde cada *renglón* es una tupla y cada *columna* se corresponde a una componente. A menudo se le dan nombres a las columnas, a los que se llaman *atributos.* Si colocamos nombres de atributo en las columnas de una relación, entonces el orden de las columnas carece de importancia. En términos matemáticos vemos a las tuplas como asociaciones entre los nombres de atributos y valores de los dominios de los atributos.

UNIVERSIDAD DE LA EMPRESA

Bibliografía.

[Paredaens 1989]. Paredaens, J., De Bra, P., Gyssens, M., Van Gucht, D. 1989. *The Structure of the Relational Data Model,* Springer-Verlag.

---oOo---