DISEÑO DE BASES DE DATOS RELACIONALES:

MODELADO LÓGICO: EL MODELO RELACIONAL

Tema 3

Manuel Ramos Cabrer

Índice

Ejemplo de una relación

ID	*nombre*	*nombre_dpto*	*salario*
10101	Mariano	Telemática	35000
22222	Ana	Física	45000
12121	Felipe	Economía	40000
32343	Juan	Historia	30000
45565	María	Telemática	35000
98345	José	Señal y Comun.	40000
76766	Asunción	Biología	32000
58583	Loreto	Historia	50000
83821	Pedro	Telemática	47000
15151	Luis	Música	33000
33456	Paula	Física	40000
76543	Lucía	Economía	42000

(a) Relación *profesores*

Tipos de atributo

- Cada atributo de una relación tiene un nombre
- El conjunto de valores permitidos para cada atributo se denomina **dominio** del atributo
- Los valores de los atributos deben ser (normalmente) **atómicos**, es decir, indivisibles
  - P.e. atributos con valores multivaluados no son atómicos
  - P.e. atributos con valores compuestos no son atómicos
- El valor especial *null* pertenece a cualquier dominio
- El valor nulo complica la definición de algunos operadores

Esquema e Instancia de un relación

- A_1, A_2, …, A_n son *atributos*

- $R = (A_1, A_2, \ldots, A_n)$ es un **esquema de relación**

  P.e. *Docente*=
  (*ID, nombre, nombre_dpto, salario*)

- Formalmente, dados los conjuntos D_1, D_2, …. D_n una **relación *r*** es un subconjunto de

  $$D_1 \times D_2 \times \ldots \times D_n$$

  - Es decir, una relación es un conjunto de n-tuplas $(a_1, a_2, \ldots, a_n)$ donde cada $a_i \in D_i$

- Los valores que almacene en cada momento (**instancia de la relación**) se especifican en forma tabular.
  - Un elemento *t* de *r* es una **tupla** y se representa por una fila.

Las relaciones no tienen orden

- El orden de las tuplas no es relevante (las tuplas se pueden almacenar en un orden arbitrario)
- P.e. la relación *docente* con tuplas no ordenadas

ID	*nombre*	*nombre_dpto*	*salario*
22222	Ana	Física	45000
12121	Felipe	Economía	40000
32343	Juan	Historia	30000
45565	María	Telemática	35000
98345	José	Señal y Comun.	40000
76766	Asunción	Biología	32000
10101	Mariano	Telemática	35000
58583	Loreto	Historia	50000
83821	Pedro	Telemática	47000
15151	Luis	Música	33000
33456	Paula	Física	40000
76543	Lucía	Economía	42000

(a) Relación *profesores*

Base de datos

- Una base de datos está formada por un conjunto de relaciones
- La información sobre una organización se divide en partes y cada relación almacena una parte de la información

  P.e.: *docente*
  alumno
  tutor

- Mal diseño: almacenar toda la información en una sola relación:

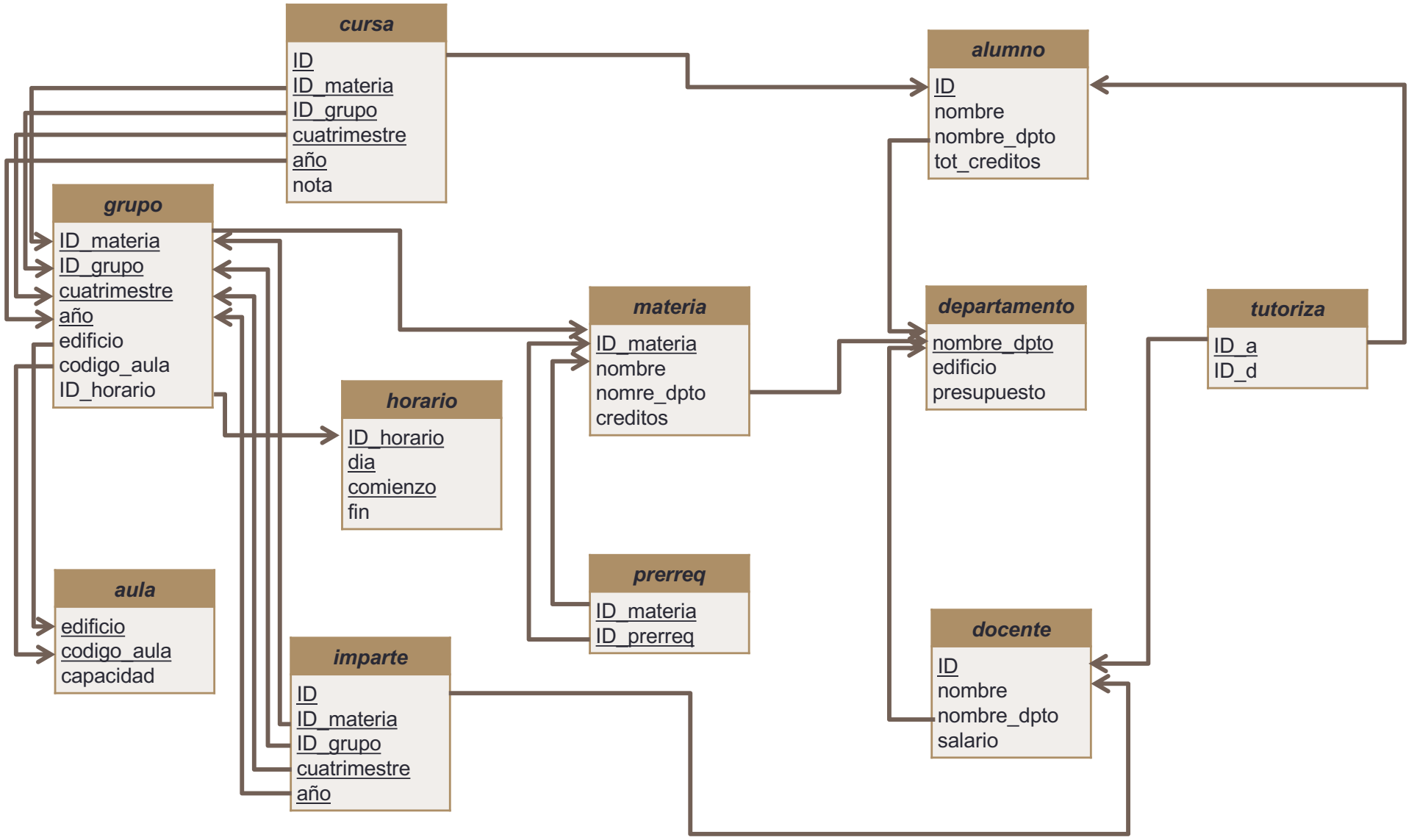
  univ(ID_docente, nombre, nombre_dpto, salario, ID_alumno, …)

  da lugar a
  - información repetida (p.e. dos alumnos tienen el mismo docente)
  - necesidad de valores nulos (p.e. información sobre un alumno sin tutor)
- La teoría de la normalización se encarga de como diseñar esquemas relacionales correctos

Claves

- Dado K $\in$ R
- *K* es una ***superclave*** de *R* si los valores de *K* son suficientes para identificar cada una de las tuplas de cada relación posible *r(R)*
  - por "posible *r*" indicamos una relación *r* que pueda existir en la organización que estamos modelando.
  - Ejemplo: {*ID*} y {*ID*, *nombre*}
    son ambas superclaves de *Docente*, si consideramos que dos docentes no pueden tener el mismo nombre.
- *K* es una ***clave candidata*** si *K* es mínima
  - Ejemplo: {ID} es una clave candidata de Docente.
- Una de las claves candidatas se elige como ***clave primaria***.
  - ¿Cuál?
- Restricción de ***clave foránea***: Valor de una relación que debe aparecer en otra.
  - Relación referenciada.
  - Relación que referencia.

Diagrama del esquema para una BD de universidad



Conversión de esquemas E-A a relaciones

- Los conjuntos entidad y los conjuntos asociación se pueden representar uniformemente como *esquemas de relación* que representan los contenidos de la base de datos.
- Una base de datos que sigue el esquema E-A se puede representar mediante un conjunto de esquemas de relación.
- Para cada conjunto entidad y cada conjunto asociación existe un esquema de relación único al que se le asigna el nombre del conjunto entidad o conjunto asociación correspondiente.
- Cada esquema tiene un conjunto de columnas (normalmente una por atributo), que tienen nombres únicos.
- Convertir un diagrama E-A a esquemas de relación es la base para conseguir un diseño relacional a partir de ese diseño E-A

Representación de conjuntos entidad con atributos simples

- Un conjunto entidad fuerte se transforma en un esquema con los mismos atributos.
  - P.e.: *alumno(ID, nombre, tot_creditos)*
- Un conjunto entidad débil se convierte en un esquema que incluye una columna para la clave primaria del conjunto entidad fuerte que la identifica.
  - P.e. *grupo(id_materia, id_grupo, cuatrimestre, año)*

Atributos compuestos y multivalorados

- Los atributos compuestos se eliminan creando un nuevo atributo para cada uno de los campos componentes
  - P.e. dado el conjunto entidad *docente* con atributo compuesto *nombre* con atributos componentes *nombre_comun* y *primer_apellido, la relación* correspondiente al conjunto entidad tendrá dos columnas
    nombre_nombre_comun y *nombre_primer_apellido*
    - *El prefijo se puede omitir si no hay ambiguedad*
- Un atributo multivalorado M de una entidad E se representa mediante una nueva relación EM
  - La relación EM tendrá como columnas la clave primaria de E y un atributo que se corresponderá con el atributo multivalorado M
  - P.e. El atributo multivalorado *telefonos* de *empleado* se representa mediante la relación
    docente_telefonos (*ID, numero_telefono*)
  - Cada valor de un atributo multivalorado se corresponde con una fila diferente de la relación EM
    - P.e., una entidad docente con clave primaria “22222” y teléfonos “123456” y “234567” se corresponde con dos filas:
      (22222, 123456) y(22222, 234567)

Representación de conjuntos asociación

- Un conjunto asociación varios a varios se representa con un esquema con atributos para las claves primarias de los dos conjuntos entidad participantes, y también para los atributos descriptivos del conjunto asociación.
- P.e.: esquema para el conjunto asociación *tutoriza:*

  tutoriza(id_a, id_d)

Representación de conjuntos asociación

- Un conjunto asociación uno a varios o varios se puede representar añadiendo un atributo adicional al lado de “varios”, que sea clave foránea y referencia la clave primaria del lado de “uno”.
- P.e.: en vez de crear un esquema para el conjunto entidad *docente_dpto*, podemos añadir una clave foránea *nombre_dpto* al conjunto entidad *docente*.

Representación de conjuntos asociación

- Para un conjunto asociación uno a uno cualquiera de los extremos puede jugar el papel de “varios”.
  - Es decir, el atributo adicional (clave foránea) se puede añadir a cualquiera de los esquemas correspondientes a los dos conjuntos entidad.
- En la utilización de claves foráneas, si la participación es parcial en el lado de “varios”, puede dar lugar a la aparición de valores nulos (*“null”*)
- En el esquema correspondiente a un conjunto asociación, el enlace entre la entidad débil y su entidad fuerte identificadora es redundante.
  - P.e. El esquema *grupo* ya contiene los atributos que deberían aparecer en el esquema grupo_*materia*.

Determinación de claves a partir de conjuntos E-A

- **Conjunto entidad fuerte**. La clave primaria del conjunto entidad pasa a ser la clave primaria del esquema.
- **Conjunto entidad débil**. La clave primaria del esquema está formada por la unión de la clave primaria del conjunto entidad fuerte y el discriminador del conjunto entidad débil.
- **Conjunto asociación**. La unión de las claves primarias de los conjuntos entidad participantes es una superclave del esquema.
  - Para conjuntos asociación varios-a-varios, la unión de las claves primarias pasa a ser la clave primaria de las relación.
  - Para conjuntos asociación varios-a-uno (si decidimos resolverlas mediante un esquema y no mediante una clave foránea), la clave primaria del conjunto entidad “varios” pasa a ser la clave primaria de la relación.
  - Para conjuntos asociación uno-a-uno (si decidimos resolverlas mediante un esquema y no mediante una clave foránea), la clave primaria de la relación puede ser la de cualquiera de los conjuntos entidad.

Representación de especializaciones con esquemas

- Método 1:
  - Crear un esquema para la entidad de nivel alto
  - Crear un esquema para cada conjunto entidad de nivel bajo, que incluirá la clave primaria del conjunto entidad de nivel alto y los atributos locales

esquema	atributos
persona	*ID, nombre, calle, ciudad*
alumno	*ID, tot_creditos*
empleado	*ID, salario*

  - Problema: obtener información sobre, por ejemplo, *empleados* requiere acceder a dos relaciones, una correspondiente al esquema de alto nivel y otra correspondiente al esquema de bajo nivel.

Representación de especializaciones con esquemas (Cont.)

- Método 2:
  - Crear un esquema para cada conjunto entidad con todos los atributos locales y heredados

esquema	atributos
persona	*ID, nombre, calle, ciudad*
alumno	*ID, nombre, calle, ciudad, tot_creditos*
empleado	*ID, nombre, calle, ciudad, salario*

  - Si la especialización es total, el esquema para la entidad generalizada (*persona*) no tiene que almacenar información
    - Se puede definir como una relación “vista” que contenga la unión de las relaciones de especialización
    - Pero aún puede ser necesario un esquema para restricciones de tipo clave foránea
  - Problema: la información sobre nombre, calle y ciudad puede almacenarse de manera redundante para personas que son a la vez alumos y empleados

Esquemas correspondientes a agregaciones

- Para representar una agregación, se crea un esquema conteniendo:
  - la clave primaria de la asociación agregada,
  - la clave primaria del conjunto entidad asociado
  - cualquier atributo descriptivo

Esquemas correspondientes a agregaciones (Cont.)

- P.e. para representar la asociación *evalua* entre la asociación *supervisa_proy* y el conjunto entidad *director*, creamos un esquema

evalua (ID_a, ID_proyecto, ID_d, ID_evaluacion)

- El esquema supervisa_proy es redundante **siempre que** permitamos almacenar valores nulos en el atributo *ID_evaluación* en la relación *evalua*.

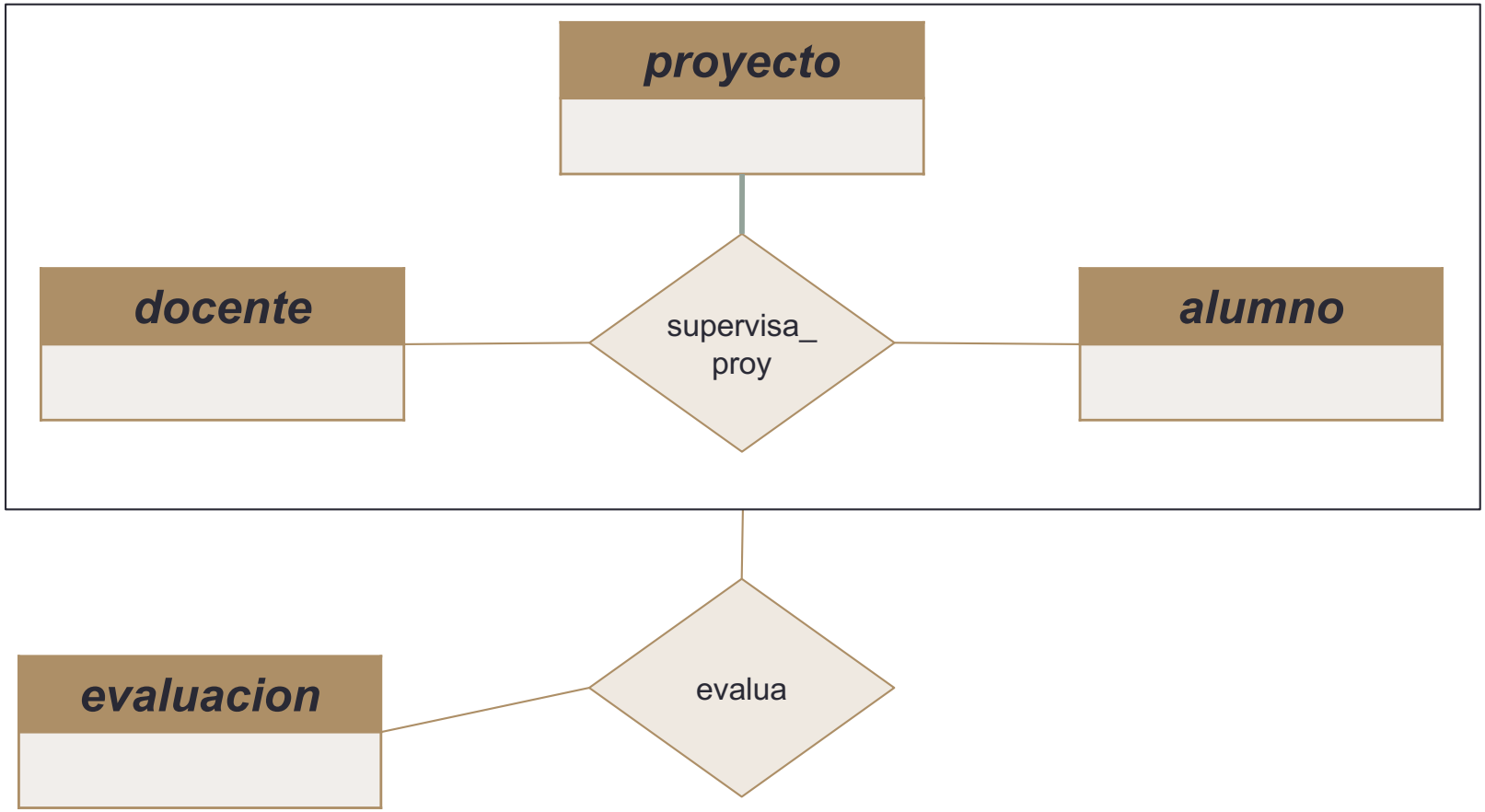


Diagrama E-A para una universidad

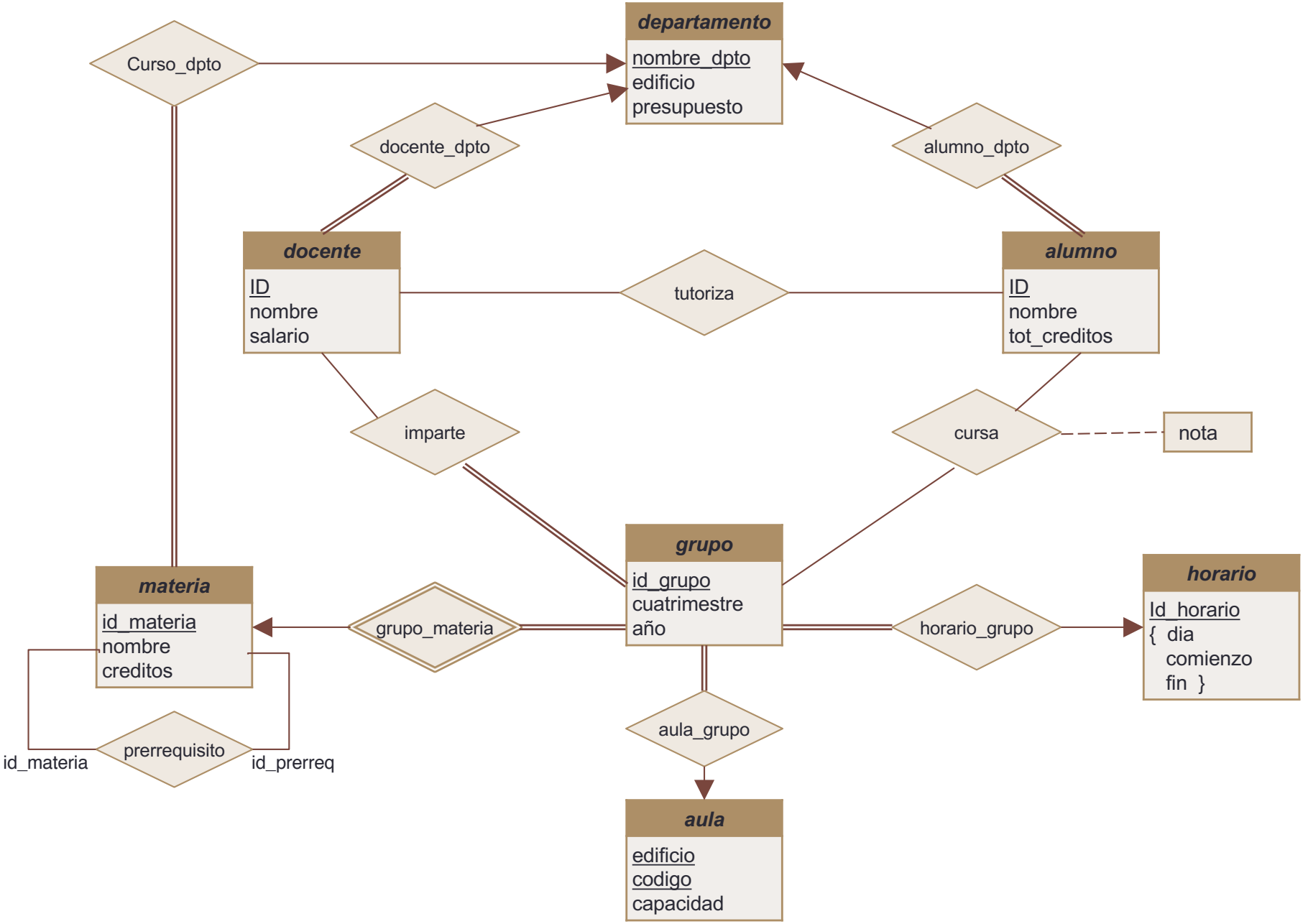
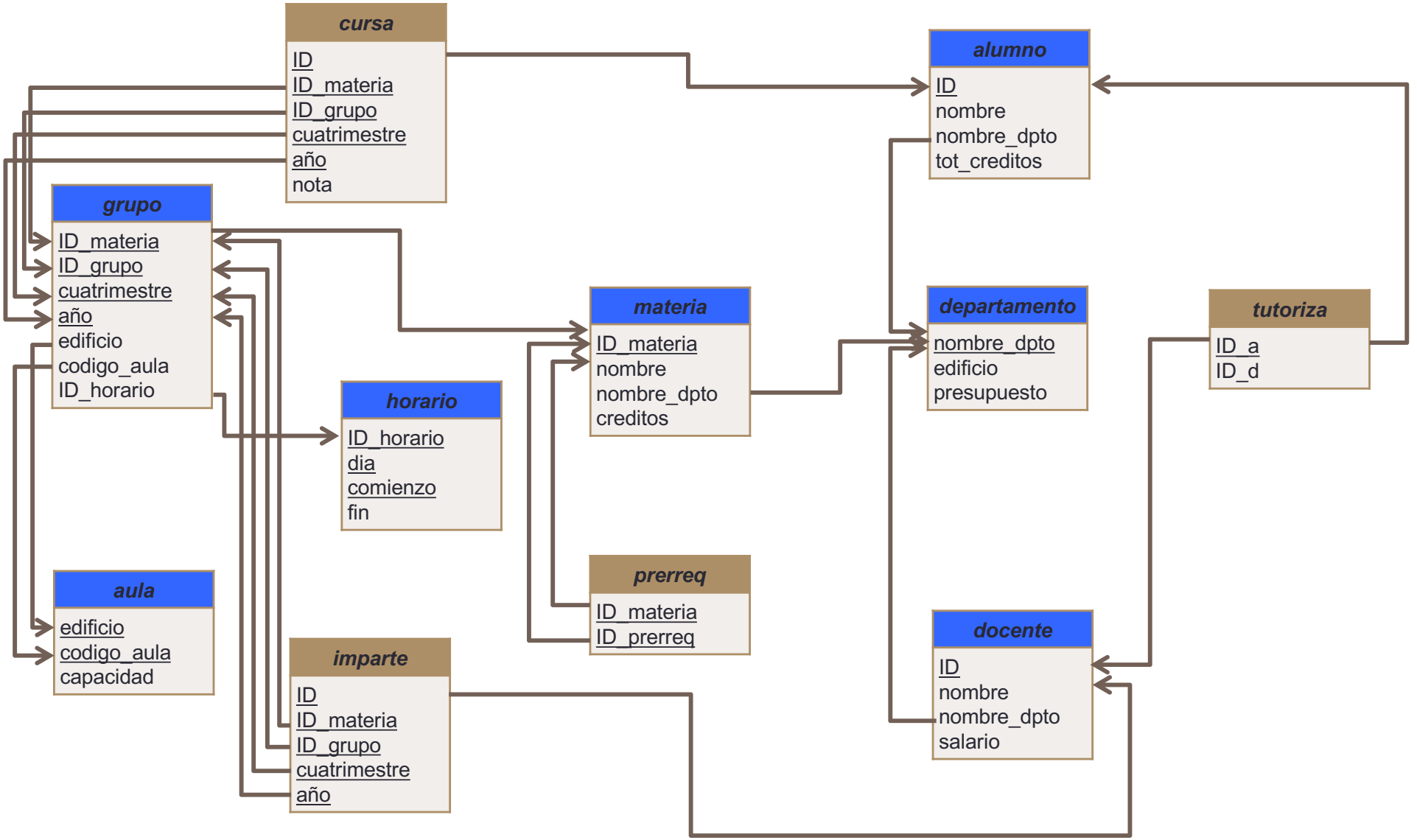


Diagrama del esquema para una BD de universidad



Restricciones de dominio

- Las restricciones de integridad nos protegen ante daños accidentales en la base de datos, asegurando que los cambios autorizados en la base de datos no van a producir una pérdida de consistencia en los datos
- Las restricciones de dominios son la forma más elemental de restricciones de integridad.
- Comprueban los valores insertados en la base de datos, y comprueban las consultas para asegurar que las comparaciones tienen sentido.
- Se pueden crear nuevos dominios a partir de los tipos de datos existentes

  P.e. **create domain** *Euros* **numeric**(12, 2)
  create domain *Libras* **numeric**(12,2)

- No se puede asignar o comparar un valor de tipo *Euros* con un valor de tipo *Libras*.
  - No obstante, se pueden convertir tipos:
    (**cast** *r.A* **as** *Libras*)

  (Se debería también multiplicar por la conversión euro-a-libra)

Integridad referencial

- Asegura que un valor que aparece en una relación para un conjunto de atributos determinado también aparece en un conjunto de atributos de otra relación.
  - Ejemplo: Si "Ingeniería Telemática" es un nombre de departamento que aparece en una de las tuplas de la relación *docente*, entonces existe una tupla en la relación *departamento* para el departamento "Ingeniería Telemática".
- Definición formal
  - Dadas las relaciones $r_1(R_1)$ y $r_2(R_2)$ con claves primarias K_1 y K_2 respectivamente.
  - El subconjunto α de R_2 es una ***clave foránea*** referenciando K_1 en la relación r_1, si para cada t_2 en r_2 debe haber una tupla t_1 en r_1 tal que $t_1[K_1] = t_2[\alpha]$.
  - Las restricciones de integridad referencial también se denominan dependencias de subconjunto ya que se pueden expresar como

$$\prod_{\alpha} (r_2) \subseteq \prod_{K1} (r_1)$$

Comprobación de integridad referencial durante una modificación

- Se deben realizar las siguientes comprobaciones con el fin de preservar la siguiente restricción de integridad referencial:

$$\prod_{\alpha}(r_2) \subseteq \prod_{K1}(r_1)$$

- **Insertar.** Si una tupla t_2 se inserta en r_2, el sistema se debe asegurar de que hay una tupla t_1 en r_1 tal que $t_1[K] = t_2[\alpha]$. Es decir

$$t_2[\alpha] \in \prod_{K}(r_1)$$

- **Eliminar.** Si se elimina una tupla t_1 de r_1, el sistema debe hallar el conjunto de tuplas de r_2 que referencianσ t_1:

$$\sigma_{\alpha = t1[K]}(r_2)$$

  Si el conjunto no es vacío
  - o bien se rechaza el comando como un error,
  - o bien se deben eliminar las tuplas que referencian a t_1 (se permiten *eliminaciones en cascada*)

Modificaciones de la base de datos (Cont.)

- **Actualizaciones.** Hay dos casos:
  - Si se actualiza una tupla t_2 en la relación r_2 y la actualización modifica los valores de la clave foránea α,entonces se debe hacer un test similar al caso de inserción:
    - Si t_2' denota el nuevo valor de la tupla t_2, el sistema se debe asegurar de que

      $$t_2'[\alpha] \in \prod_K(r_1)$$
  - Si se actualiza una tupla t_1 en r_1, y la actualización modifica el valor de la clave primaria (*K*), entonces se debe realizar un test similar a la del caso de eliminación:
    1. El sistema debe calcular

       $$\sigma_{\alpha = t1[K]}(r_2)$$

       utilizando el valor anterior de t_1 (el valor antes de hacer la actualización).
    2. Si el conjunto no es vacío
       1. la actualización se puede rechazar como un error, o
       2. La actualización se puede hacer en cascada sobre las tuplas del conjunto, o
       3. Las tuplas del conjunto se pueden eliminar.

Lenguajes de consulta relacionales

- Lenguajes con los que el usuario obtiene información almacenada en la base de datos.
- Tipos de lenguajes
  - procedimentales
  - No procedimentales (declarativos)
- Lenguajes “puros”:
  - Álgebra relacional
  - Cálculo relacional de tuplas
  - Cálculo relacional de dominios
- Los lenguajes puros son la base de los lenguajes que se utilizan habitualmente

Álgebra relacional

- Lenguaje procedimental
- Seis operadores básicos
  - selección
  - proyección
  - unión
  - diferencia de conjuntos
  - producto cartesiano
  - renombrar
- Los operadores se aplican sobre dos o más relaciones y dan como resultado una nueva relación.

Operación de selección

- Notación: $\sigma_p(r)$
- *p* se denomina predicado de la selección
- Se define como:

$$\sigma_p(\boldsymbol{r}) = \{t \mid t \in r \ \ \mathbf{y}\ p(t)\}$$

donde *p* es una fórmula en calculo proposicional formada por términos unidos por : $\wedge$ (y), $\vee$ (**o**), $\neg$ (**no**)
Cada término tiene la forma:

<atributo> *op* <atributo> o <constante>

donde *op* es: =, ≠, >, ≥, <, ≤

- Ejemplo de selección:

$\sigma_{\textit{nombre\_dpto=“Ingeniería Telemática”}}(\textit{departamento})$

Operación de selección – Ejemplo

- Relación *r*

A	*B*	*C*	*D*
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

- $\sigma_{A=B \wedge D > 5}(r)$

A	*B*	*C*	*D*
α	α	1	7
β	β	23	10

Operación de proyección

- Notación:

$$\prod_{A1, A2, \ldots, Ak} (r)$$

donde A_1, A_2 son nombres de atributos y r en un nombre de relación.

- El resultado es la relación de k columnas que se obtiene eliminando las columnas no listadas
- Las filas duplicadas del resultado se eliminan, ya que las relaciones son conjuntos
- P.e. Eliminar el atributo nombre-sucursal de *cuentas*

$$\prod_{numero\text{-}cuenta,\ saldo} (\text{cuenta})$$

Operación de proyección – Ejemplo

- Relación *r*:

A	*B*	*C*
α	10	1
α	20	1
β	30	1
β	40	2

$\prod_{A,C}(r)$

A	*C*
α	1
α	1
β	1
β	2

=

A	*C*
α	1
β	1
β	2

Operación de unión

- Notación: $r \cup s$
- Se define como:

$$r \cup s = \{t \mid t \in r \text{ o } t \in s\}$$

- Para que $r \cup s$ sea válida.

1. r, s deben tener el mismo número de atributos.
2. Los dominios de los atributos deben ser *compatibles* (p.e., la 2ª columna de r contiene el mismo tipo de valores que la 2ª columna de s)

- P.e. encontrar los nombre de todas las personas de la universidad

$$\prod_{nombre} (alumno) \cup \prod_{nombre} (docente)$$

Operación de unión – Ejemplo

- Relaciones *r, s:*

A	*B*
α	1
α	2
β	1

r

A	*B*
α	2
β	3

s

r ∪ s:

A	*B*
α	1
α	2
β	1
β	3

Operación diferencia de conjuntos

- Notación $r - s$
- Se define como:

$$r - s = \{t \mid t \in r \ \mathbf{y}\ \mathrm{t} \notin s\}$$

- La diferencia de conjuntos se debe realizar entre relaciones *compatibles*.
  - *r* y *s* deben tener el mismo número de atributos
  - Los dominios de los atributos de *r y s* deben ser compatibles

Operación diferencia de conjuntos – Ejemplo

- Relaciones *r, s:*

A	*B*
α	1
α	2
β	1

r

A	*B*
α	2
β	3

s

r – s:

A	*B*
α	1
β	1

Operación producto cartesiano

- Notación *r* x *s*
- Se define como:

$$r \times s = \{t\, q \mid t \in r \ \textbf{y}\ q \in s\}$$

- Se asume que los atributos de r(R) y s(S) son disjuntos. (Es decir, $R \cap S = \varnothing$).
- Si los atributos de *r(R)* y *s(S)* no son disjuntos, se debe utilizar la operación de renombrar.

Operación producto cartesiano - Ejemplo

Relaciones *r, s*:

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

r x *s*:

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Combinación de operaciones

- Se pueden construir expresiones utilizando varias operaciones
- Ejemplo: $\sigma_{A=C}(r \times s)$
- $r \times s$

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

- $\sigma_{A=C}(r \times s)$

A	B	C	D	E
α	1	α	10	a
β	2	β	20	a
β	2	β	20	b

Operación de renombrado

- Permite nombrar, y por tanto referirnos a, los resultados de las expresiones de álgebra relacional.
- Permite referirse a una relación con más de un nombre.

  Ejemplo:

$$\rho_X (E)$$

  devuelve la expresión *E* con el nombre *X*

- Si la expresión *E* en álgebra relacional tiene un orden *n*, entonces

$$\rho_{X\ (A1,\ A2,\ \ldots,\ An)} (E)$$

  devuelve la expresión *E* con el nombre *X*, y con los atributos renombrados a *A1, A2, …., An*.

Definición formal

- Una expresión básica en álgebra relacional puede ser:
  - Una relación de la base de datos
  - Una relación constante
- Dadas dos expresiones en álgebra relacional E_1 y E_2 también son expresiones en álgebra relacional:
  - $E_1 \cup E_2$
  - $E_1 - E_2$
  - $E_1 \times E_2$
  - $\sigma_p (E_1)$, P es un predicado sobre los atributos de E_1
  - $\prod_s (E_1)$, S es una lista que contiene algunos atributos de E_1
  - $\rho_x (E_1)$, x es el nuevo nombre del resultado de E_1

Otras operaciones

- Podemos definir más operaciones que no proporcionan nueva funcionalidad al álgebra relacional pero que simplifican consultas habituales.
  - Intersección de conjuntos
  - Reunión (*join*) natural
  - División
  - Asignación

Operación intersección de conjuntos

- Notación: $r \cap s$
- Se define como:

$$r \cap s = \{ t \mid t \in r \textbf{ and } t \in s \}$$

- Asumiendo:
  - r, s tienen el mismo número de atributos
  - Los atributos de r y s con compatibles
- Nota: $r \cap s = r - (r - s)$

Operación intersección de conjuntos - Ejemplo

- Relaciones r, s:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- r ∩ s

A	B
α	2

Operación *join* natural

- Notación: $r \bowtie s$

- Dadas dos relaciones *r* y *s* sobre los esquemas *R* y *S* respectivamente.
  Entonces, $r \bowtie s$ es una relación sobre el esquema $R \cup S$ que se obtiene de la siguiente manera:
  - Se considera cada par de tuplas t_r de *r* y t_s de *s*.
  - Si t_r y t_s tienen el mismo valor para cada uno de los atributos en $R \cap S$, se añade una tupla *t* al resultado, donde
    - *t* tiene el mismo valor que t_r sobre *r*
    - *t* tiene el mismo valor que t_s sobre *s*

- Ejemplo:
  $R = (A, B, C, D)$
  $S = (E, B, D)$
  - Esquema de relación = (A, B, C, D, E)
  - $r \bowtie s$ se define como:

$$\Pi_{r.A,\ r.B,\ r.C,\ r.D,\ s.E}\ (\sigma_{r.B = s.B \wedge r.D = s.D}\ (r \times s))$$

Operación *join* natural - Ejemplo

- Relaciones r, s:

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ε

s

$r \bowtie s$

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

Operación división

$$r \div s$$

- Adecuada para consultas que incluyan la expresión “para todos”.
- Dadas las relaciones *r* y *s* sobre los esquemas R y S respectivamente, donde
  - $R = (A_1, \ldots, A_m, B_1, \ldots, B_n)$
  - $S = (B_1, \ldots, B_n)$

  El resultado de r ÷ s es una relación sobre el esquema

  $R - S = (A_1, \ldots, A_m)$

$$r \div s = \{t \mid t \in \prod_{R-S}(r) \wedge \forall u \in s \ (tu \in r)\}$$

Operación división - Ejemplo

Relaciones *r, s*:

A	*B*
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
ε	6
ε	1
β	2

r

B
1
2

s

r ÷ *s*:

A
α
β

Otro ejemplo de división

Relaciones *r, s*:

A	*B*	*C*	*D*	*E*
α	a	α	a	*1*
α	a	γ	a	*1*
α	a	γ	b	*1*
β	a	γ	a	*1*
β	a	γ	b	*3*
γ	a	γ	a	*1*
γ	a	γ	b	*1*
γ	a	β	b	*1*

r

D	*E*
a	*1*
b	*1*

s

r ÷ *s*:

A	*B*	*C*
α	a	γ
γ	a	γ

Operación asignación

- La operación asignación (←) facilita un modo conveniente de expresar consultas complejas.
  - Escribir consultas como un programa secuencial consistente en
    - un conjunto de asignaciones
    - seguido por una expresión cuyo valor se muestre como el resultado de la consulta.
  - La asignación siempre se debe realizar a una variable relación temporal.
- Ejemplo: $r \div s$ se puede expresar como

$$temp1 \leftarrow \Pi_{R-S}(r)$$
$$temp2 \leftarrow \Pi_{R-S}((temp1 \times s) - \Pi_{R-S,S}(r))$$
$$result = temp1 - temp2$$

  - El resultado de la expresión a la derecha de ← se asigna a la variable relación de la izquierda de ←.
  - La variable se puede utilizar en las expresiones que van a continuación.

Operaciones del Álgebra Relacional Extendida

- Proyección generalizada
- Reunión (*join*) externa
- Funciones agregadas

Proyección generalizada

- Extiende la operación de proyección permitiendo funciones aritméticas en la lista de proyección.

$$\Pi_{F1, F2, \ldots, Fn}(E)$$

- *E* es cualquier expresión en álgebra relacional
- $F_1, F_2, \ldots, F_n$ son expresiones aritméticas que incluyen constantes y atributos del esquema de *E*.
- P.e. Dada la relación *info_producto(nombre, precio_compra, precio_venta),* averiguar cuánto se gana en cada producto:

$$\Pi_{nombre,\ precio\_venta - precio\_compra}\ (info\_producto)$$

Funciones y operaciones agregadas

- Las **funciones de agregación** toman como argumentos un conjunto de valores y devuelven un valor simple como resultado.

  avg: valor medio
  min: valor mínimo
  max: valor máximo
  sum: suma de valores
  count: número de valores

- **Operación agregada** en álgebra relacional

$$_{G1, G2, \ldots, Gn}\, g_{F1(A1), F2(A2), \ldots, Fn(An)} (E)$$

  - E es cualquier expresión en álgebra relacional
  - $G_1, G_2 \ldots, G_n$ es una lista de atributos sobre los que agrupar (puede estar vacía)
  - Cada F_i es una función agregada
  - Cada A_i es un nombre de atributo

Operación agregada - Ejemplo

- Relación *r*:

A	*B*	*C*
α	α	7
α	β	7
β	β	3
β	β	10

$$g_{\text{sum(c)}}(r)$$

sum(c)
27

Operación agregada - Ejemplo

- Relación *cuenta* agrupada por *nombre-sucursal*:

nombre-sucursal	*numero-cuenta*	*saldo*
Vigo	A-102	400
Vigo	A-201	900
Madrid	A-217	750
Madrid	A-215	750
Pontevedra	A-222	700

$$nombre\text{-}sucursal\ \mathcal{G}\ sum(saldo)\ (cuenta)$$

nombre-sucursal	*saldo*
Vigo	1300
Madrid	1500
Pontevedra	700

Funciones agregadas (Cont.)

- El resultado de la agregación no tiene nombre
  - Se puede utilizar la operación de renombrado para darle un nombre
  - Por conveniencia, se permite el renombrado como parte de la operación de agregación

$$nombre\text{-}sucursal\ \mathcal{G}\ \textbf{sum}(saldo)\ \textbf{as}\ sum\text{-}saldo\ (cuenta)$$

Join externo

- Es una extensión de la operación de join que evita la pérdida de información.
- Calcula el join y después añade las tuplas de una relación que no coinciden con las tuplas de la otra relación al resultado del join.
- Utiliza valores *null*:
  - *null* significa que el valor es desconocido o no existe
  - Todas la comparaciones en las que participa un valor *null* son **falsas** por definición.

Join externo – Ejemplo

- Relación *materia*

ID_materia	*nombre_materia*	*creditos*
170	Biología	3
230	Geografía	4
260	Matemáticas	6

- Relación *matriculado*

nombre_alumno	*ID_materia*
Juan López	L-170
Ana Vázquez	L-230
José García	L-155

Join externo – Ejemplo

- **Join interno**

materia ⋈ *matriculado*

ID_materia	*nombre_materia*	*creditos*	*nombre_alumno*
170	Biología	3	Juan López
230	Geografía	4	Ana Vázquez

- **Join externo izquierdo**

materia ⟕ *matriculado*

ID_materia	*nombre_materia*	*creditos*	*nombre_alumno*
170	Biología	3	Juan López
230	Geografía	4	Ana Vázquez
260	Matemáticas	6	*null*

Join externo – Ejemplo

- **Join externo derecho**

$materia \rtimes\!\!\!\!\subset matriculado$

ID_materia	*nombre_materia*	*creditos*	*nombre_alumno*
170	Biología	3	Juan López
230	Geografía	4	Ana Vázquez
155	*null*	*null*	José García

- **Join externo total**

$materia \supset\!\!\rtimes\!\!\subset matriculado$

ID_materia	*nombre_materia*	*creditos*	*nombre_alumno*
170	Biología	3	Juan López
230	Geografía	4	Ana Vázquez
260	Matemáticas	6	*null*
155	*null*	*null*	José García

Valores nulos

- Las tuplas pueden contener valores nulos, denotados por *null*, en algunos de sus atributos
- *null* significa valor desconocido o que el valor no existe.
- El resultado de cualquier expresión aritmética en la que participe *null* es *null.*
- Las funciones agregadas ignoran los valores *null*
  - Es una decisión arbitraria. Alternativamente se podría haber devuelto como resultado *null*.
  - Seguimos la semántica de SQL respecto al manejo de valores nulos
- Para eliminación de duplicados y agrupamientos, *null* recibe el mismo tratamiento que cualquier otro valor, y se asume que dos nulos son iguales
  - Alternativa: asumir que cada nulo es distinto de los demás
  - Ambas son decisiones arbitrarias. Nosotros seguimos SQL

Valores nulos

- Las comparaciones con valores *null* devuelven un valor especial de verdad denominado *desconocido*
  - Si se usa *falso* en vez de *desconocido*, entonces *not (A < 5)* no sería equivalente a *A >= 5*
- Lógica trivalorada utilizando el valor de verdad *desconocido*:
  - OR: (*desconocido* **or** verdad) = verdad,
    (*desconocido* **or** *falso*) = *desconocido,*
    (*desconocido* **or** *desconocido)* = *desconocido*
  - AND: *(verdad* **and** *desconocido)* = *desconocido,*
    (falso **and** *desconocido)* = *falso,*
    (desconocido **and** *desconocido)* = *desconocido*
  - NOT*:* *(***not** *desconocido)* = *desconocido*
  - En SQL "*P* **es desconocido**" se evalúa a verdad si el predicado *P* se evalúa a *desconocido*
- El resultado de un predicado de selección se trata como *falso si se evalúa como desconocido*

Modificación de la base de datos

- El contenido de la base de datos se puede modificar utilizando las siguientes operaciones:
  - Borrado
  - Inserción
  - Actualización
- Todas estas operaciones se expresan mediante el operador de asignación.

Borrado

- Una petición de borrado se expresa de manera similar a una consulta, excepto que, en vez de mostrar las tuplas al usuario, las tuplas seleccionadas se eliminan de la base de datos.
- Sólo se pueden eliminar tuplas completas; no se pueden eliminar solo determinados atributos
- Un borrado se expresa en álgebra relacional como:

$$r \leftarrow r - E$$

donde r es una relación y E es una consulta en álgebra relacional.

Ejemplo de borrado

- Borrar todas los alumnos con menos de 50 créditos cursados.

$$alumno \leftarrow alumno - \sigma_{tot\_creditos < 50}(alumno)$$

Inserción

- Para insertar datos en una relación podemos:
  - o bien especificar la tupla a insertar
  - o bien escribir una consulta cuyo resultado esté formado por las tuplas a insertar
- En álgebra relacional, una inserción se expresa:

$$r \leftarrow r \cup E$$

  donde r es una relación y E es una expresión en álgebra relacional.
- La inserción de una sola tupla se realiza cuando E es una relación constante que contiene una tupla.

Ejemplo de inserción

- Insertar información en la base de datos sobre un nuevo alumno.

$$alumno \leftarrow alumno \cup \{(\text{“76767”}, \text{“Felipe Gómez”}, \text{“Ingeniería Telemática”}, 0)\}$$

Actualización

- Permite cambiar el valor de una tupla sin cambiar *todos* los valores de la tupla
- Para ello se utiliza la operación de proyección generalizada

$$r \leftarrow \Pi_{F1, F2, \ldots, Fl,} (r)$$

- Cada F_i es
  - el atributo *i* de *r*, si el atributo *i* no se quiere actualizar, o,
  - si se va a actualizar el atributo *i*, F_i es una expresión en la que intervienen solamente constantes y los atributos de *r*, que proporciona el nuevo valor del atributo

Ejemplo de actualización

- Añadir 5 créditos al alumno con ID “22222”.

$$cuenta \leftarrow \Pi_{ID,\ nombre,\ nombre\_dpto,\ tot\_creditos\ +\ 5}(\sigma_{ID=\text{“22222”}}(alumno))$$