UADER

Facultad de Ciencia Y Tenología

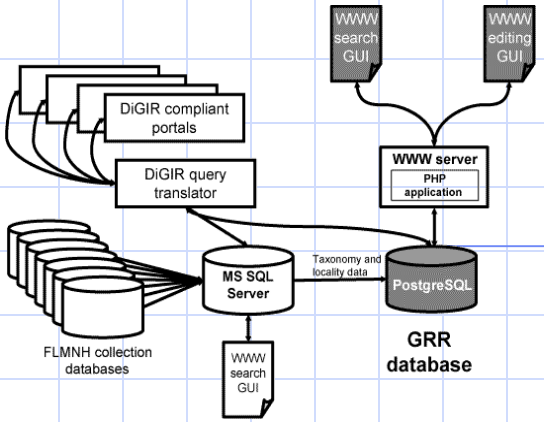
Base de Datos Avanzadas

Base de datos Avanzadas

Repaso Modelo Relacional
Metodologías asociadas
SQL DMLs

Ing. Fernando Sato
fsatopna@gmail.com

Ultima Actualización: 18/03/2019



Resumen

- Revisión de Modelo de datos y DER

Revisión del Modelo Relacional

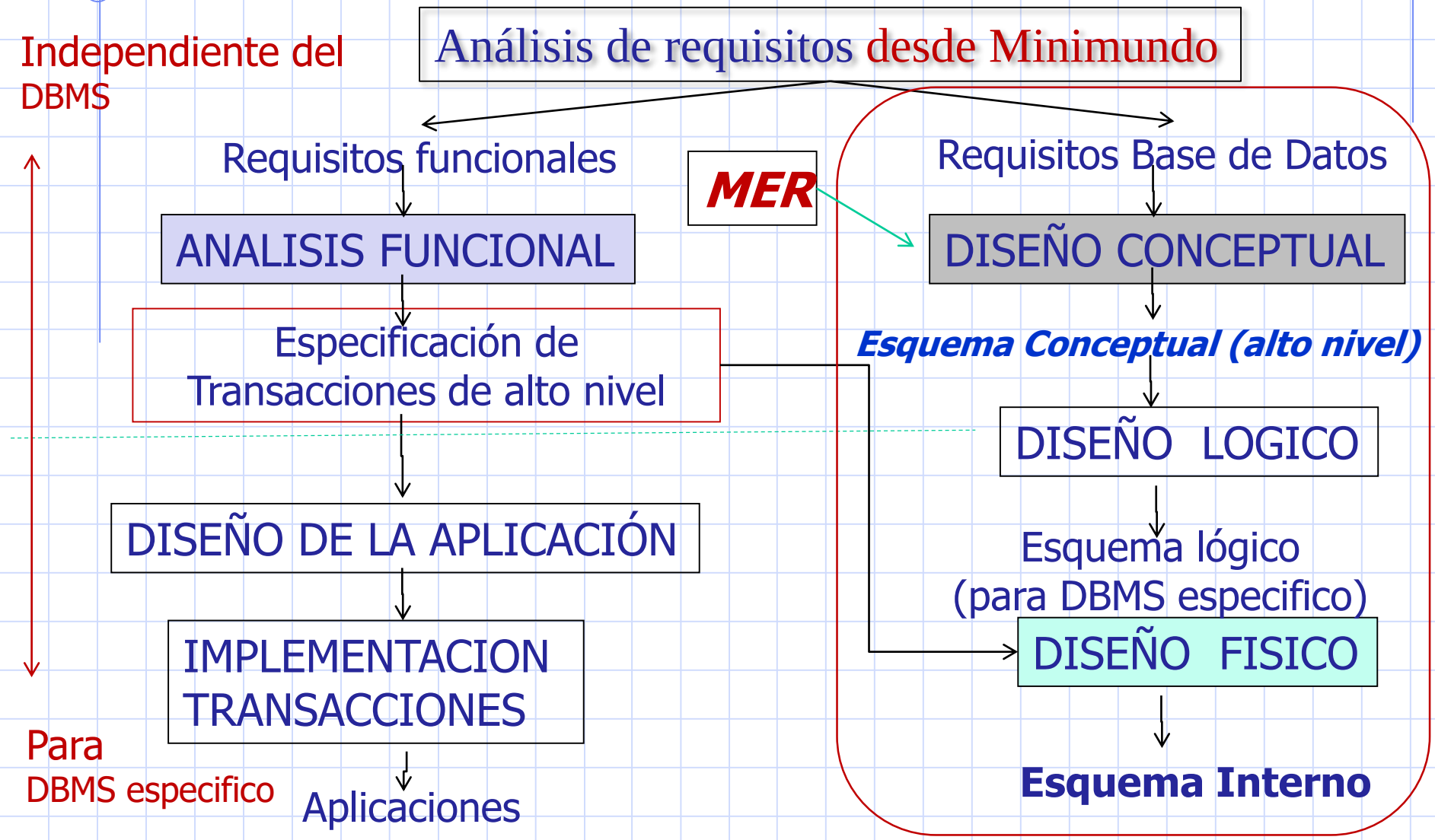
Revisión del Algebra Relacional

Normalización

DMLs de consultas

DMLs de actualización

El Proceso de Diseño ...



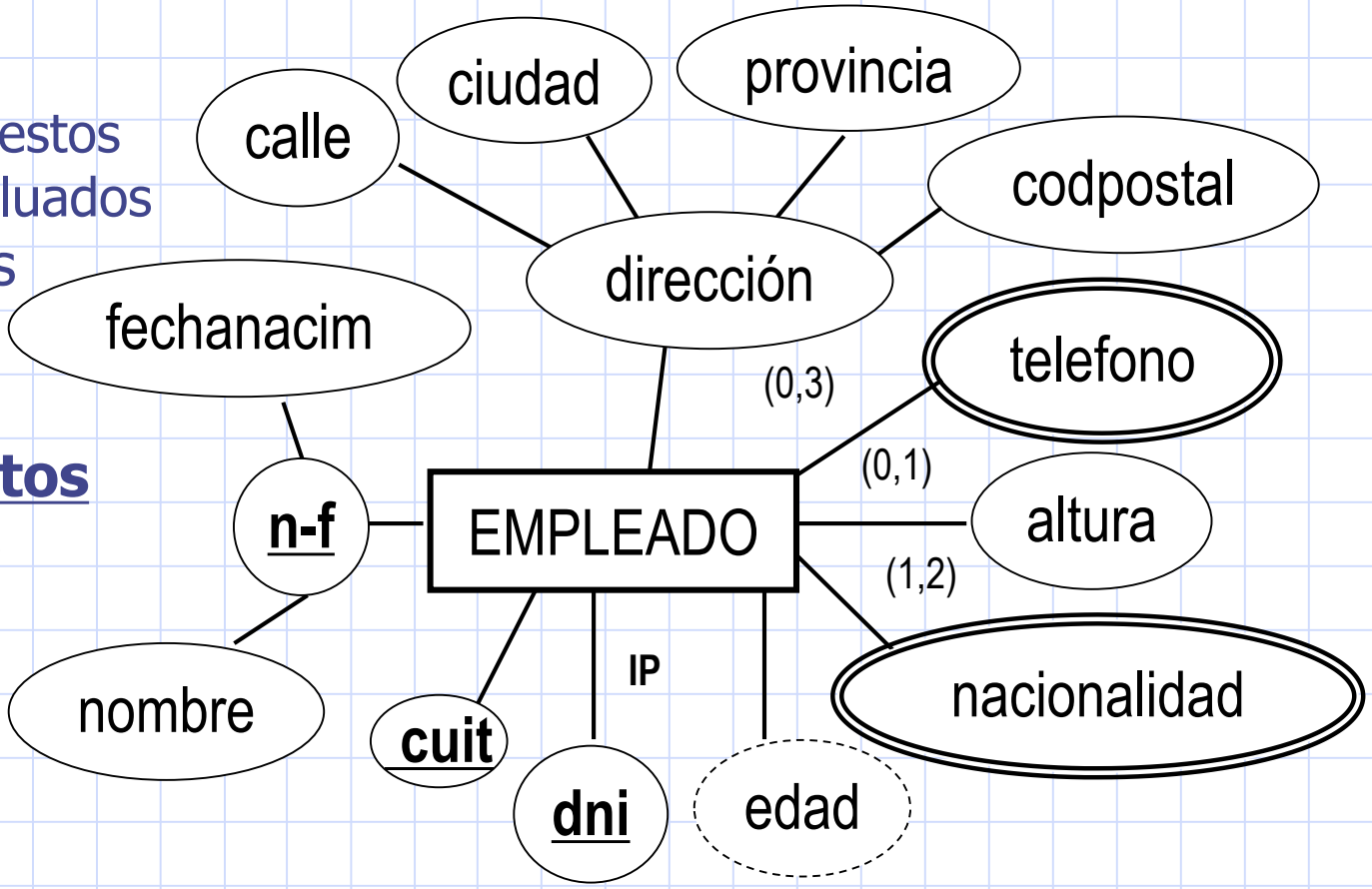
Entidades y Atributos

Tipos

- Identificador
- Simples
- Complejos:
  - Compuestos
  - Multivaluados
- Almacenados
- Derivados

Otros Conceptos

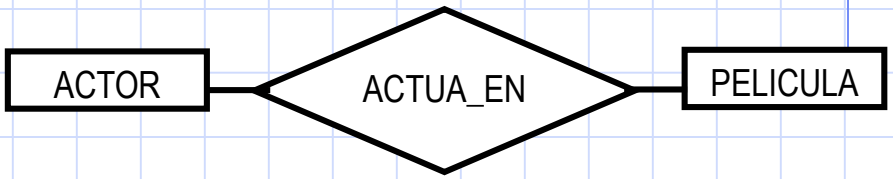
- El valor Nulo
- Cardinalidad



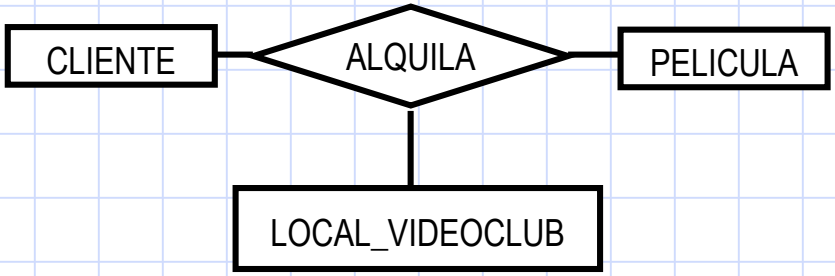
Relaciones – Concepto → Grado

Número de tipos de entidad que participan en el tipo de relación

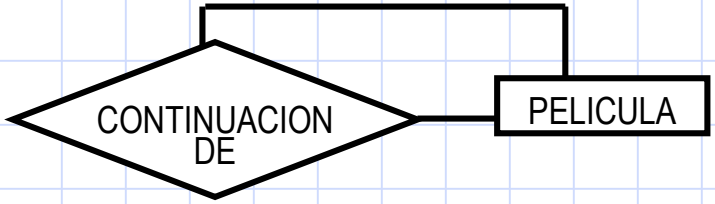
- Binaria: grado 2 (el más frecuente)



- Ternaria: grado 3

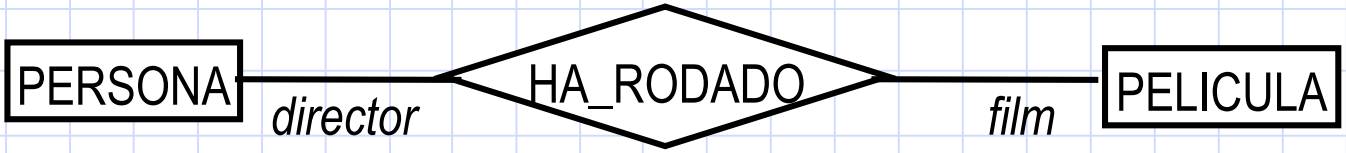


- Recursiva (o reflexiva): grado 1

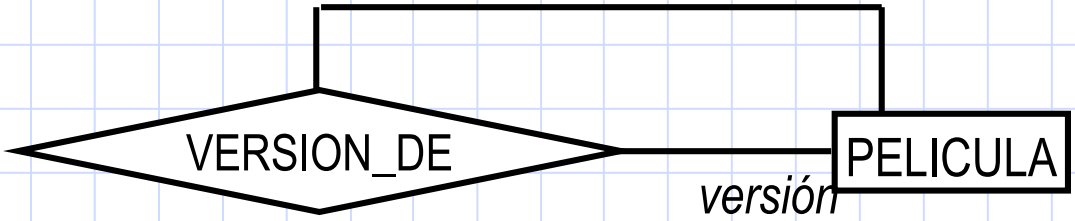


Relaciones – Nombres de Rol

- Todo tipo de entidad que participa en un tipo de relación juega un papel específico en la relación



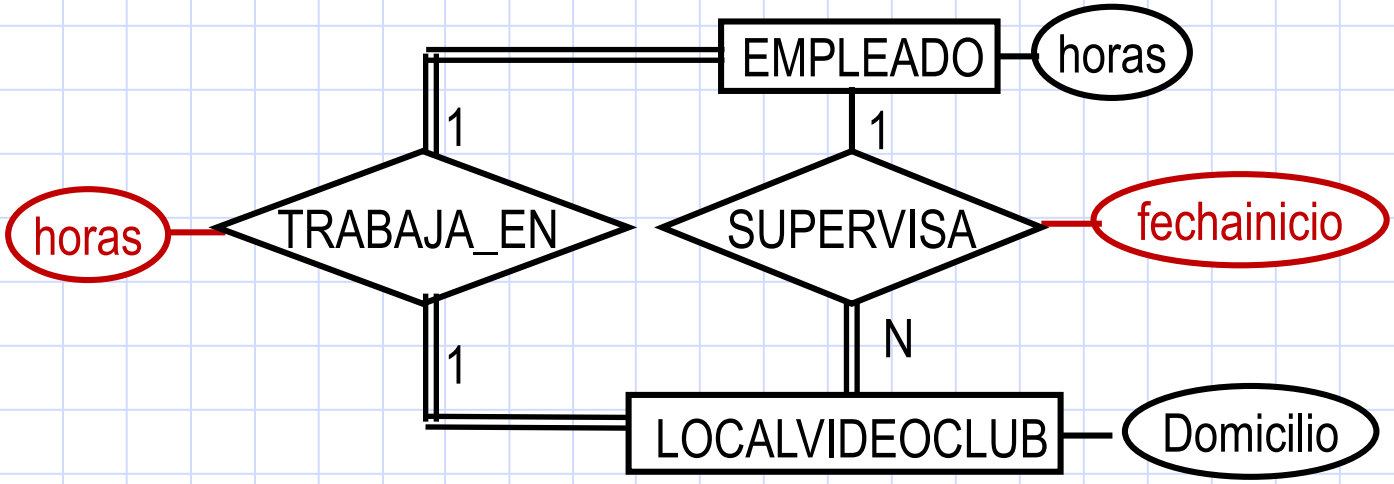
- Los nombres de rol se deben usar, sobre todo, en los tipos de relación recursivos, para evitar ambigüedad



Razón de Cardinalidad

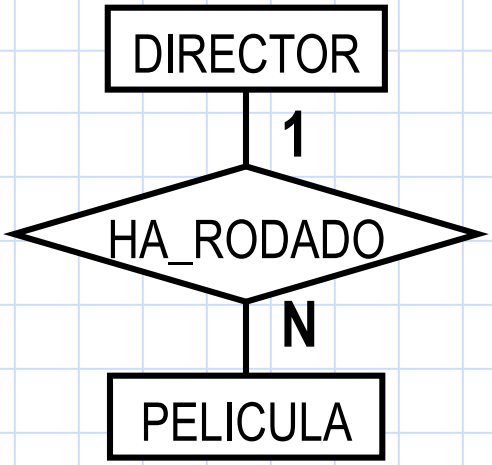
Conceptualmente pertenecen a la relación

- Un atributo de una M:N es propio de la relación
- Un atributo de una 1:1 o 1:N "se puede llevar" a uno de los tipos de entidad participantes



Razón de Cardinalidad

- **Número máximo de instancias de tipo de relación en las que puede participar una misma instancia de tipo de entidad**
  - la cardinalidad de HA_RODADO es "1 a N"
  - HA_RODADO es de tipo "1 a N"



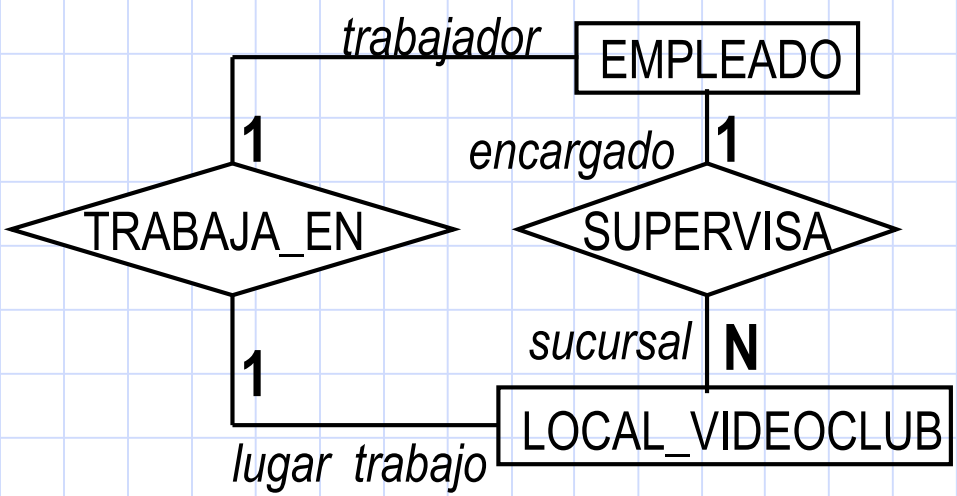
- **Notación**
  - etiqueta en la línea que une entidad y relación
  - Se lee: "un director roda varias películas" y "una película es rodada por un único director"

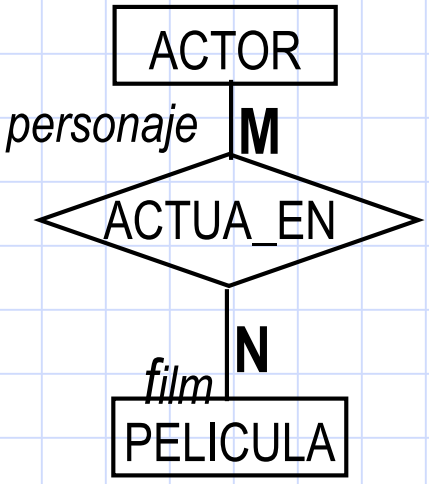
Razón de Cardinalidad

Razones de cardinalidad más comunes:

- 1:1 (“uno a uno”)
- 1:N (“uno a muchos”)
- M:N (“muchos a muchos”)

La cardinalidad con que participa una entidad en una relación es la etiqueta de la línea que la une a la relación

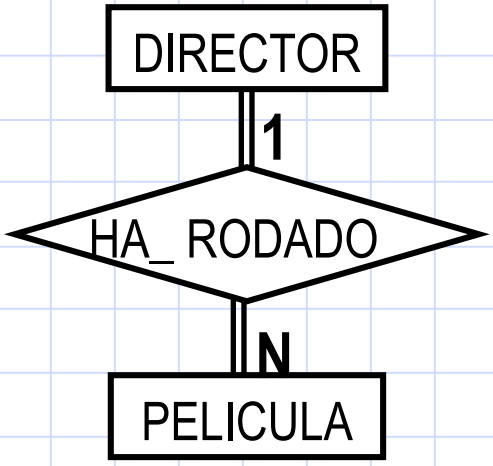


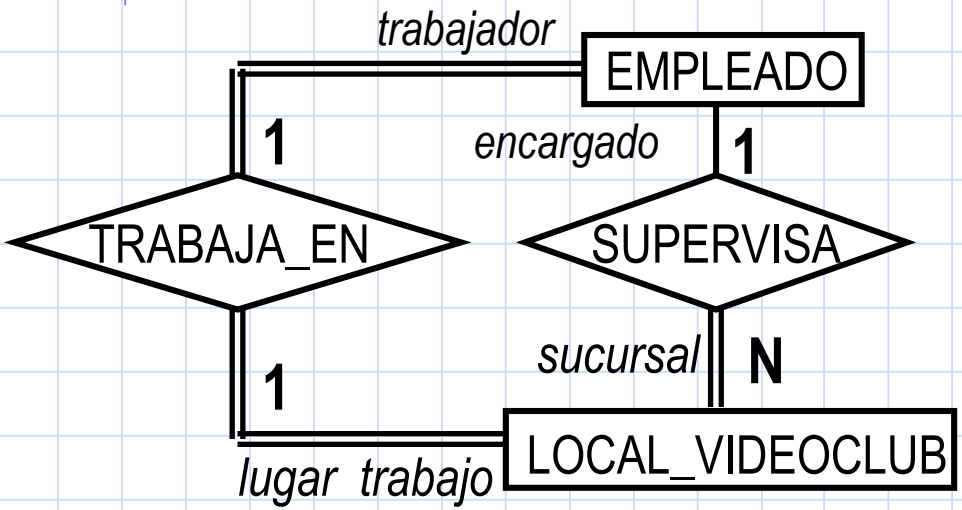


Razón de Participación

Notación

- Líneas dobles o simples

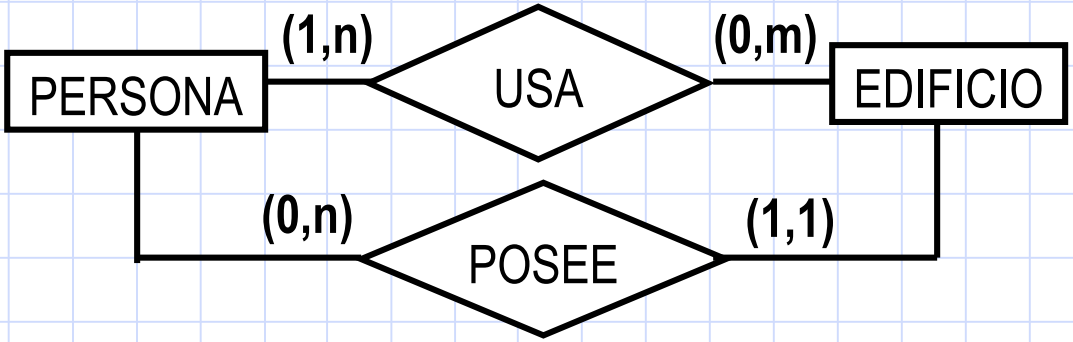




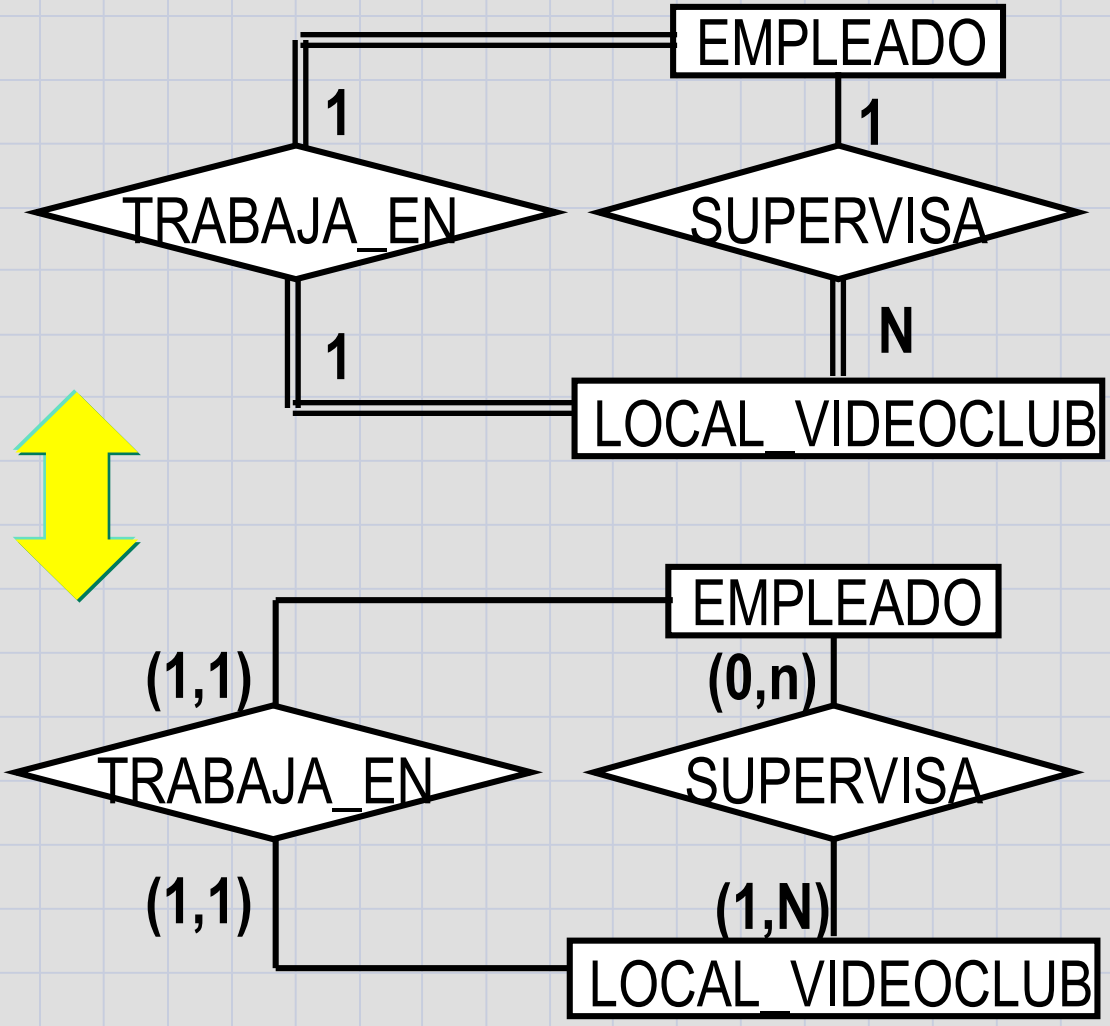
Cardinalidad Mínima / Máxima

- Números mínimo y máximo de instancias del tipo de relación en las que puede intervenir una instancia del tipo de entidad

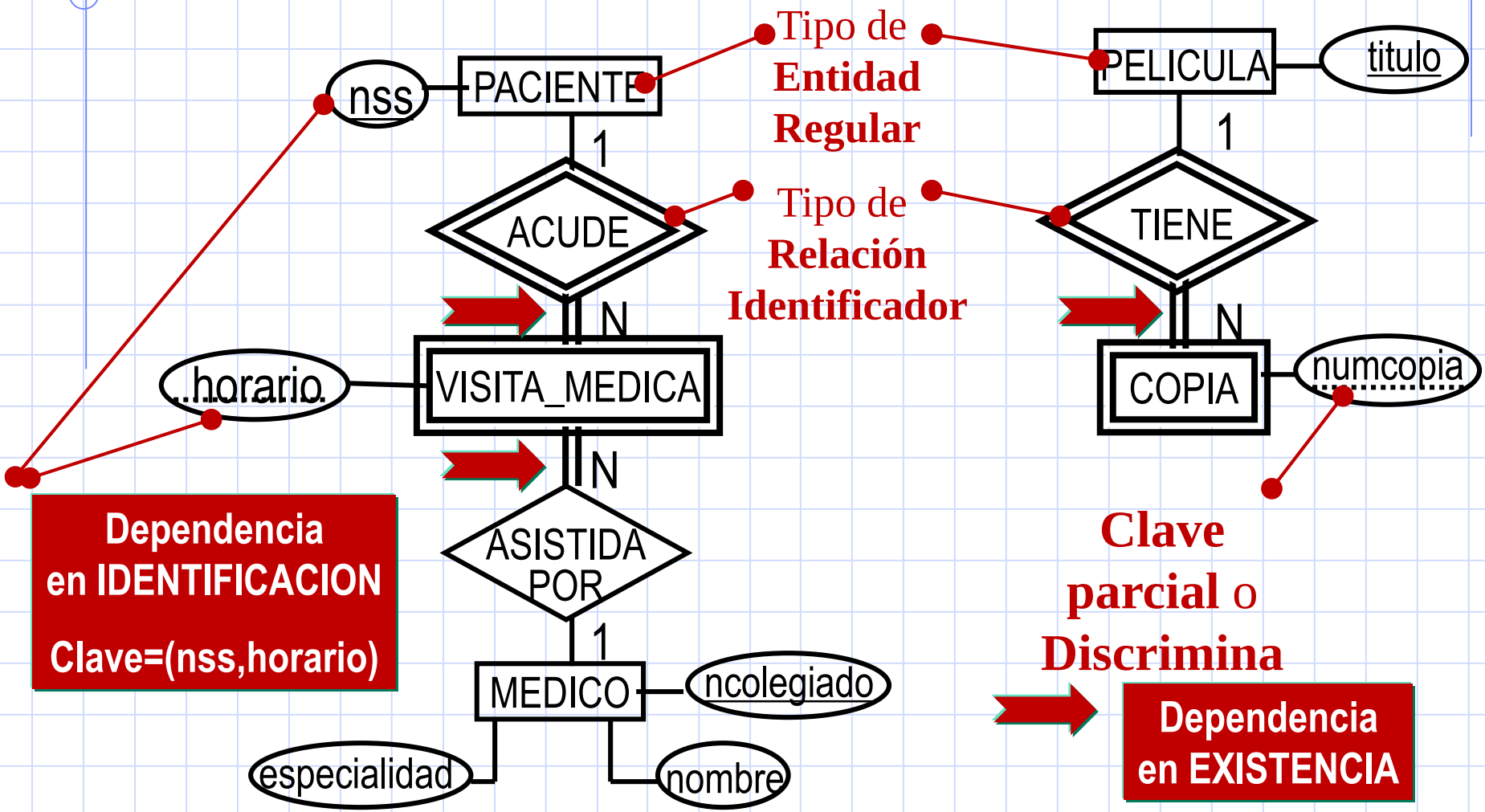
- Notación
  - `(min, max)` en la línea que une entidad y relación



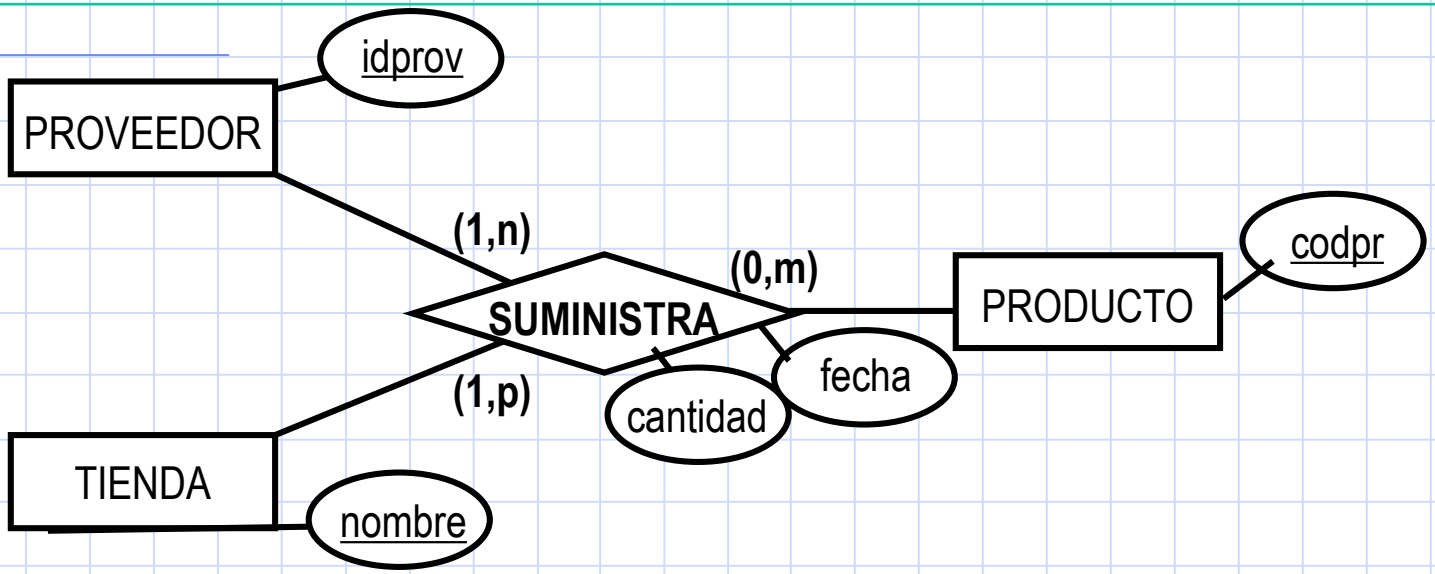
Cardinalidad Vs Participación

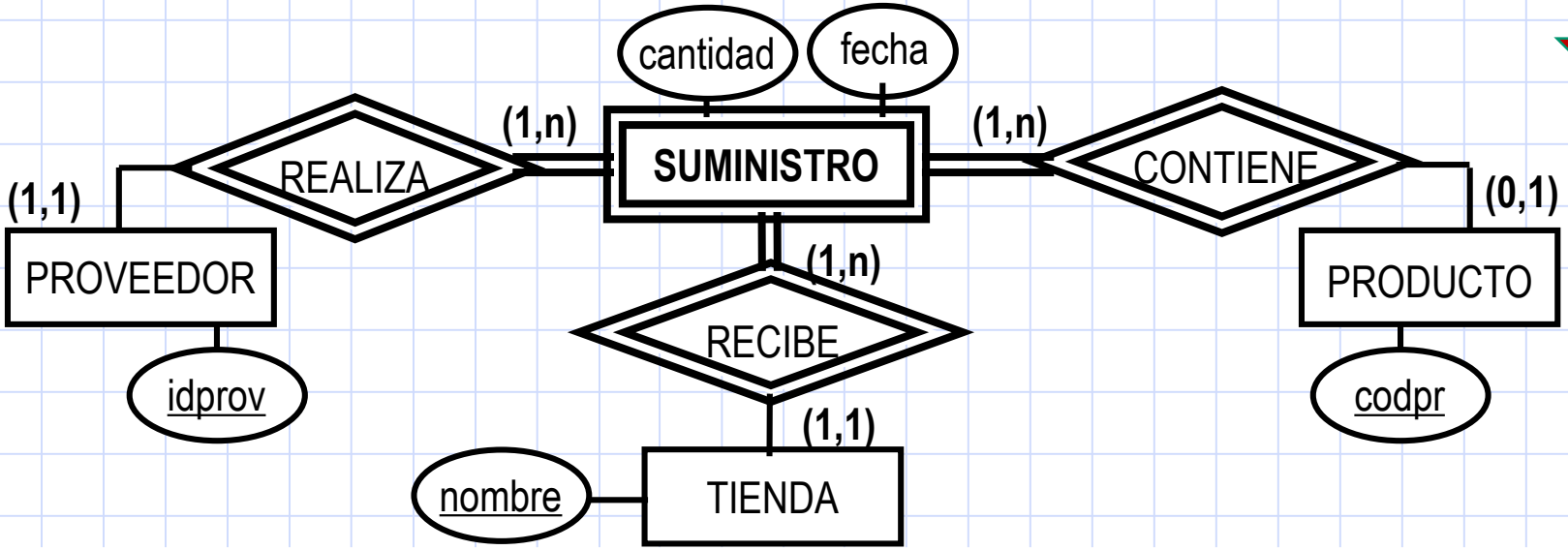


Entidades Debiles



Ternarias → Binarias





Resumen

- Revisión del Modelo de datos y DER
- Revisión del Modelo Relacional
- Revisión del Algebra Relacional
- Normalización
- DMLs de consultas
- DMLs de actualización

Modelo de Datos

Relaciones Base

Son los repositorios naturales de información.
Para SQL son las **Tablas**.

Relaciones Virtuales:

No poseen información referencian directa o indirectamente a un relación base.
Para SQL son las **Vistas**.

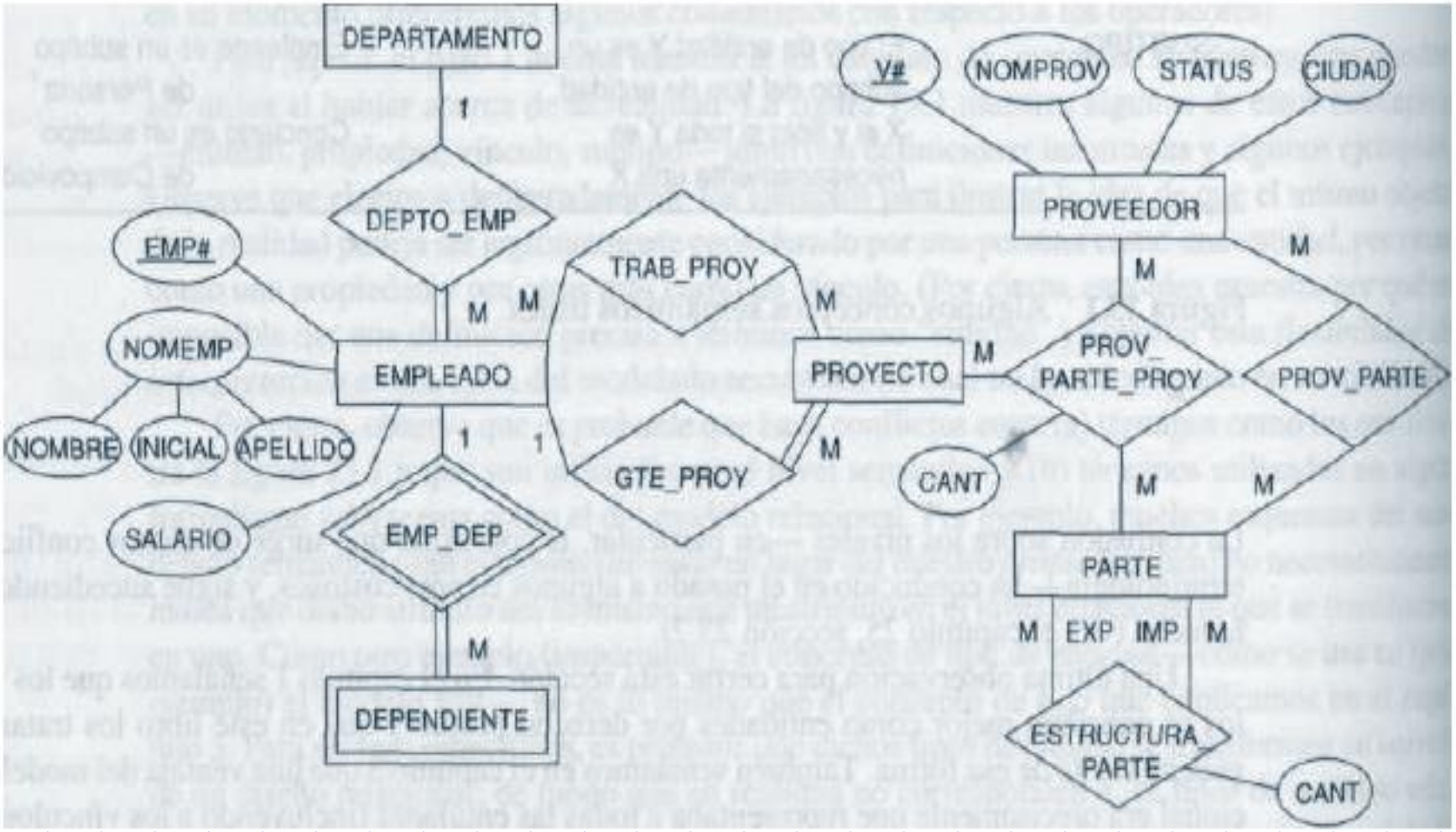
Relaciones Resultado:

Toda operación relacional genera una nueva relación.

```
Ej: Select dni, nombre from personas
```

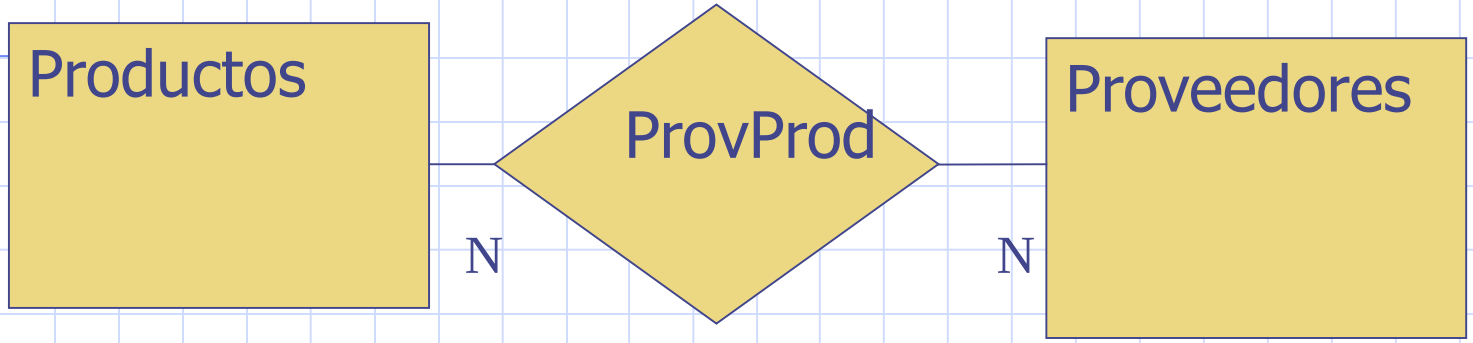
Es una nueva relación que se puede usar por ej como tabla o relación derivada.

DER – Visión Bibliografía.



Tomado de: *Fundamentos de Sistemas de Base de Datos*
Elmasri / Navathe

DER – Visión Practica



Productos	Proveedores_Productos	Proveedores
Codigo	producto_codigo	Cuit
Nombre	cuit	Nombre
Abreviatura	fecha_inicio	Codpos
	fecha_fin	

Productos

Codigo	Nombre	Abrev
10	CDr Teltron	CDrT
20	DVDr Tel	DVDrT

Proveedores

Cuit	Nombre	Codpos
20170446667	Garcia Pedro	3100
27170445134	Salas Marina	3000

Proveedores_Productos

Producto_codigo	Cuit	Fecha_inici	Fecha_fin
10	20170446667	20/08/2009	
30	20170446667	10/07/2009	

Caracteristicas del Modelo

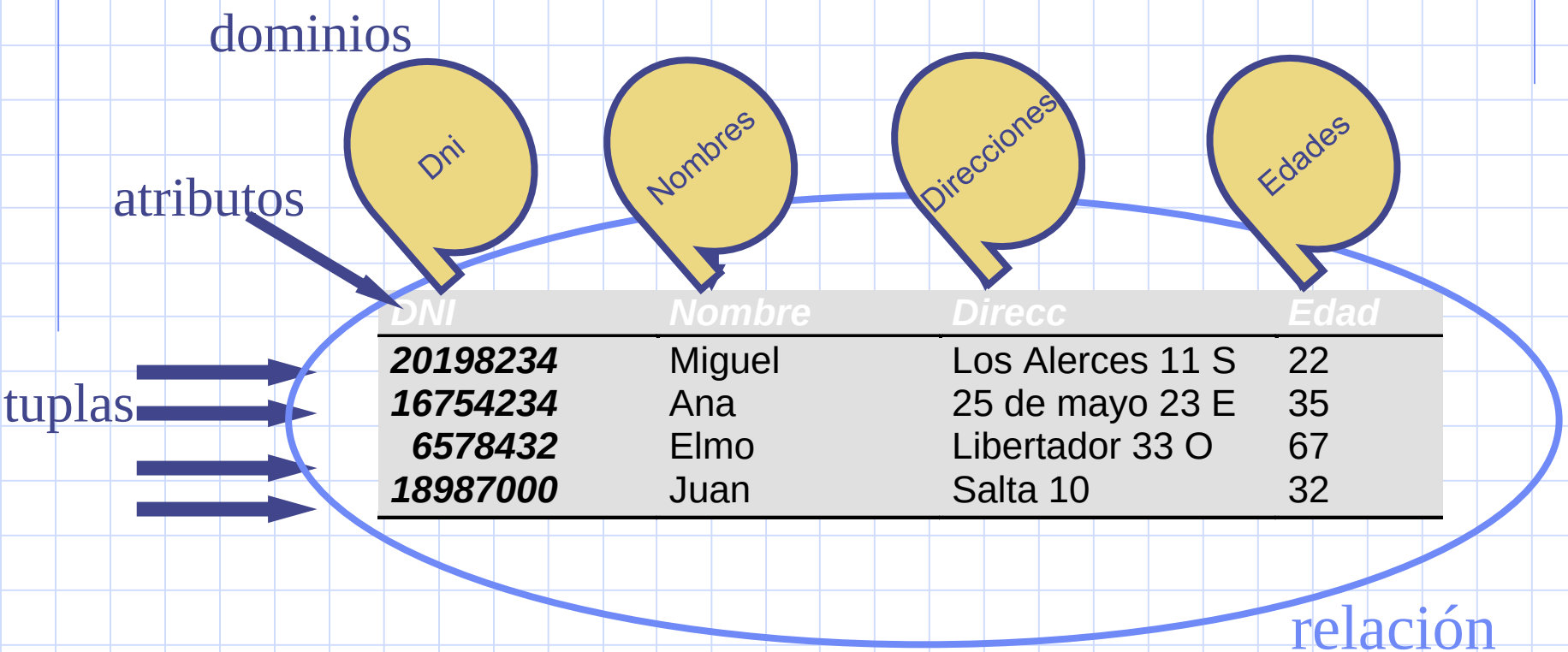
- La base de datos es vista como una colección de ***relaciones***.
- Una relación puede ser vista como una ***tabla***, con filas llamadas ***tuplas*** y con cabeceras de columnas llamadas ***atributos***.

Relación

- Única estructura del modelo
- Estructura elemental, atómica

Carscteristicas del Modelo (II)

Gráficamente son → Tablas



DNI	Nombre	Direcc	Edad
20198234	Miguel	Los Alerces 11 S	22
16754234	Ana	25 de mayo 23 E	35
6578432	Elmo	Libertador 33 O	67
18987000	Juan	Salta 10	32

Diseño Lógico

Persona (dni, nombre, direcc, edad) ← *se pude poner* ***pk (dni)***

Caracteristicas del Modelo (III)

- **Dominio:** Un dominio D es un conjunto finito o infinito de valores homogéneos y atómicos que un atributo puede tomar.

  Se pueden definir por

  comprensión 0 <= entero <= 10

  o por **extensión** ('Licenciado','Ingeniero','Técnico')

- **Atributo**: Representa una característica importante de la relación (relevante), un atributo toma los valores de un dominio determinado.

  Nota: 0 <= enteros <= 10

  Titulo: ('Licenciado','Ingeniero','Técnico')

Caracteristicas del Modelo (IV)

Formalización de las Relaciones en SQL

Cabecera →

Cuerpo

DNI	*Nombre*	*Direcc*	*Edad*
20198234	Miguel	Los Alerces 11 S	22
16754234	Ana	25 de mayo 23 E	35
6578432	Elmo	Libertador 33 O	67
18987000	Juan	Salta 10	32

Cabecera R{(A1:D1),(A2:D2),……,(Am:Dm)}

R es una relación de grado m, su primer atributo se denomina A1 y es del dominio D1, luego tiene A2 del dominio D2, etc.

Esto tiene relación directa con SQL

```
CREATE TABLE R ( A1 D1, A2 D2,…..,Am Dm);
```

Para el ejemplo

```
CREATE TABLE Persona ( DNI int, nombre char(20),… );
```

Caracteristicas del Modelo (V)

Formalización de las Relaciones en SQL

Cabecera →

Cuerpo

DNI	*Nombre*	*Direcc*	*Edad*
20198234	Miguel	Los Alerces 11 S	22
16754234	Ana	25 de mayo 23 E	35
6578432	Elmo	Libertador 33 O	67
18987000	Juan	Salta 10	32

Cuerpo R {(A1:vi1),(A2:vi2),……,(Am:vim)} donde i=1,2,…,n

El cuerpo esta formado por un conjunto de tuplas que varían en el tiempo, cada tupla esta formada por pares atributo-valor {(A1:vi1),(A2:vi2),……,(Am:vim)} donde vij pertence al dominio J

La relación es de grado m y cardinalidad n

¿Cuánto vale m y cuanto n? ¿Qué representan?

Propiedades de las Relaciones

No existen tuplas duplicadas:
Esto se debe a que el cuerpo de la relación es un conjunto matemático. Por esto existe una diferencia entre tabla(admite duplicados) y relación (como dijimos no admite).

No existe orden entre las tuplas:
Mismas razones que la propiedad anterior.

No existe orden en los atributos:
R{(A1:D1)(A2:D2)} es equivalente a R{(A2:D2)(A1:D1)}

Todos los valores tomados para los atributos en las relaciones son atómicos:
No es posible que asuman multivalores o vectores.

Tipos de Relaciones

Relaciones Base

Son los repositorios naturales de información.
Para SQL son las **Tablas**.

Relaciones Virtuales:

No poseen información referencian directa o indirectamente a un relación base.
Para SQL son las **Vistas**.

Relaciones Resultado:

Toda operación relacional genera una nueva relación.

```
Ej: Select dni, nombre from personas
```

Es una nueva relación que se puede usar por ej como tabla o relación derivada.

Tipos de Relaciones (I)

Relaciones Intermedias

Surgen de una suboperacion de una operación relacional.

```
Ej: Select dni,
           nombre
      from personas where dni in
         (select dni  from  deudores)
```

Primeramente el motor resuelve select dni from deudores y con esto crea una relación intermedia.

Relaciones Instantaneas (Snapshot):

Son las relaciones que se arman desde una relación en un instante determinado de tiempo. (Foto)

```
Ejemplo stock al 30-junio-2002.
```

Tipos de Relaciones (III)

Relaciones Temporales

Son relaciones que se usan en un programa o procedimiento almacenado con el fin hacer calculos u operaciones intermedias. Cuando el objeto que la creo "muere" el motor elimina estas relaciones.

```
Ej: create temporary table (dni int);
```

Resumen

- Revisión del Modelo Relacional
- Revisión de DER
- Revisión del Algebra Relacional
- Normalización
- DMLs de consultas
- DMLs de actualización

Trabajando con Relaciones

Proceso de consulta de una base de datos relacional:
Toda consulta a una SBDR genera como resultado una relación.

Existen dos mecanismos formales para consultar un SBDR con fuerte fundamento teórico:

el Calculo Relacional

el Algebra Relacional.

Calculo Relacional, una idea

Basado en el calculo de Predicados de Primer Orden.

No Procedural o Procedimental.
Tiene la estructura `Relación: predicado`

Ej: Dada la relación:

`ESTUDIANTE(DNI, NOM, EDAD, DIR)`

Seleccionar tuplas de estudiantes llamados Pepe:

`ESTUDIANTE: NOM='PEPE'`

Seleccionar estudiantes que viven en Bilbao y tienen más de 23 años:

`ESTUDIANTE: DIR='Bilbao' AND EDAD>23`

Algebra Relacional

El resultado se da por una aplicación sucesiva de operaciones del motor de resolución.

Cada ***operación*** toma una o mas ***relaciones como operandos*** y produce una ***relación resultado***.

Hay 8 (ocho) operadores básicos los cuales se pueden combinar en cualquier orden en operaciones compuestas:

Operadores Unarios:

Selección, Proyección

Operadores Binarios:

Union, Diferencia, Intersección, Producto Cartesiano, Reunión, División

Operador Proyección

Subconjunto vertical de la relación origen, es decir grado <= al grado de R y cardinalidad igual a R.

Genera otro esquema de relación que contiene solamente los atributos de R especificados en la lista de atributos

ID	DNI	NOMBRE	DOMCALLE	DOMNRO
1	29848842	Coronel, David Augusto	Misiones	718
2	30552687	Faifer, María Alfonsina	San Luis	651
3	31706580	Filipuzzi, Luciano	Pasteur	0
4	29121246	Ielpo, Constanza Soledad	Galan	1307
5	27833326	Piedrabuena, María de las Mercedes	Belgrano	530
6	29447659	Zamboni, Jorgelina Esther	Diamante	60
7	29620470	Caisso, Mónica Gabriela	Santiago del Estero	339
8	30808960	Ibarra, Leandro Ariel	Saavedra	555
9	21423660	Moia, Maria Del Rosario	Francisco del Miranda Nº 790	790
10	21912673	Wiszniovski, Gustavo Juan M.	Urquiza	658

```
Select
  dni, nombre
from personas
```

Notación: $\pi_{<\text{lista de atributos}>}(R)$

Ejemplo: $\pi_{<\text{dni,nombre}>}(\text{Personas})$

Operador Selección

- Subconjunto horizontal de la relación origen, es decir grado = al grado de R y cardinalidad <= a R.
- Genera otra relación cuyo esquema es el mismo de R y, en cuanto a la extensión, posee todas las tuplas de R que satisfacen la condición de selección

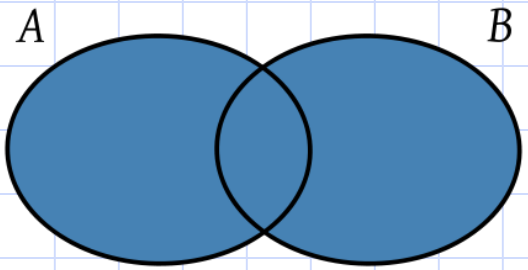
```
Select * from personas where edad < 25
```

ID	DNI	NOMBRE	DOMCALLE	DOMNRO	DOMPISO	EDAD
1	29848842	Coronel, David Augusto	Misiones	718	PB	33
2	30552687	Faifer, María Alfonsina	San Luis	651		23
3	31706580	Filipuzzi, Luciano	Pasteur	0	PB	44
4	29121246	Ielpo, Constanza Soledad	Galan	1307		51
5	27833326	Piedrabuena, María de las Mercedes	Belgrano	530		21
6	29447659	Zamboni, Jorgelina Esther	Diamante	60	8	20
7	29620470	Caisso, Mónica Gabriela	Santiago del Estero	339	3	54
8	30808960	Ibarra, Leandro Ariel	Saavedra	555		53
9	21423660	Moia, Maria Del Rosario	Francisco del Miranda Nº 790	790	2	40
10	21912673	Wiszniovski, Gustavo Juan M.	Urquiza	658		51

Notación: $\sigma_{\text{<selection condition>}}(R)$

Ej: $\sigma_{\text{<edad < 25>}}(\text{Personas})$

Operador Unión



Genera un esquema de relación que posee el mismo conjunto de atributos de R1 y R2, e incluye las tuplas que pertenecen a R1 o a R2 o a ambas.

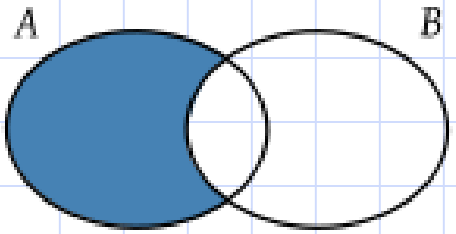
```
Select * from personas union select * from deudores;
```

Notación: R1 **U** R2

Restricción: Relaciones Unión Compatibles o Compatibles con la Unión

Dos relaciones, R(A1, A2, ..., An) y S(B1, B2, ..., Bn), serán unión compatibles si tienen el mismo tipo de tuplas. Es decir, si ambas tienen grado n y si dom(Ai)=dom(Bi) para $1 \leq i \leq n$.

Operador Diferencia



- Genera un esquema de relación que posee el mismo conjunto de atributos de R1 y R2, e incluye las tuplas que pertenecen a R1 y no pertenencen a R2.

```
Select * from personas Minus select * from deudores;
```

o

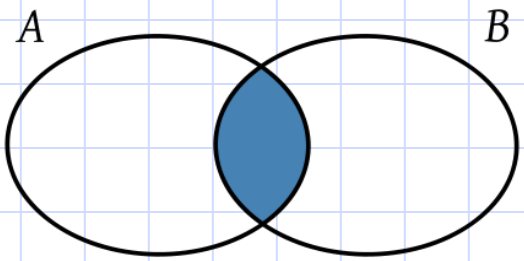
```
Select * from personas where dni not in (select dni from deudores);
```

Notación: R1 **minus** R2 *(except en PostgreSQL)*

Restricción: Idem Union, relaciones R1 y R2 deben ser compatibles.

Operador no conmutativo A minus B es distinto que B minus A

Operador Intersección



Genera un esquema de relación que posee el mismo conjunto de atributos de R1 y R2, e incluye las tuplas que pertenecen a R1 y a R2

```
Select * from personas intersect select * from deudores;
o
Select p.* from personas p, deudores d where p.dni = d.dni;
```

Notación: $R1 \cap R2$

Restricción:

Idem Union, R1 y R2 deben ser compatibles.

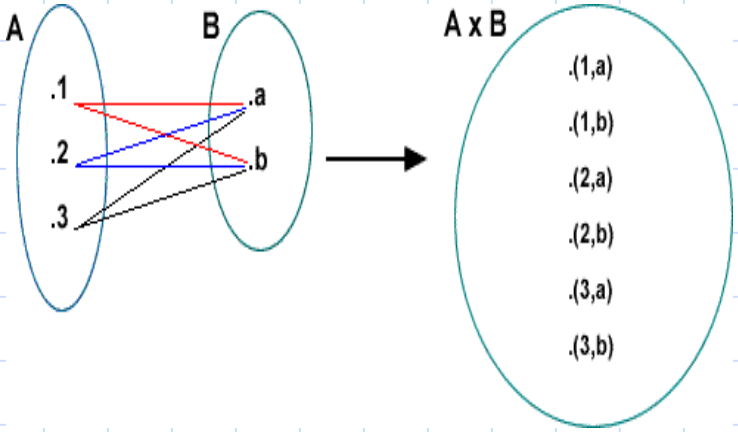
Operador Producto Cartesiano

Genera un esquema de relación que posee la unión de atributos de R1 y R2, y como tuplas todas las combinaciones posibles de las tuplas de R1 y R2

```
Select * from personas, deudores;
```

Notación: R1 *x* R2

Restricción: No existen, aplicable
A cualquier R1 y R2, incluso R x R.



Operador Reunión

Genera un esquema de relación que posee la unión de atributos de R1 y R2, (n+m), y como tuplas todas las combinaciones posibles de R1 y R2 que satisfacen la condición de reunión.

```
Select * from personas p, localidades l where l.cp = p.cp;
o
Select p.* from personas p join localidades l on l.cp =
p.cp;
```

Notación: $R1 \bowtie_{<r1.cp=r2.cp>} R2$

Restricción:

Deben tener uno o mas atributos con el mismo significado y dominio.

Operador Reunión (II)

Personas

Dni	Nombre	Cp
17	Juan	3000
21	Maria	3100
24	Pedro	2000
27	Luis	3100

Localidades

Cp	Nombreloc	Provincia
3100	Paraná	ER
2000	Rosario	Santa Fe

Producto Cartesiano Personas x Localidad

Personas Reunion <p.cp=l.cp>Localidad

Dni	Nombre	CP	Cp2	Nombreloc	Provincia
17	Juan	3000	3100	Paraná	ER
21	Maria	3100	3100	Paraná	ER
24	Pedro	2000	3100	Paraná	ER
27	Luis	3100	3100	Paraná	ER
17	Juan	3000	2000	Rosario	Santa Fe
21	Maria	3100	2000	Rosario	Santa Fe
24	Pedro	2000	2000	Rosario	Santa Fe
27	Luis	3100	2000	Rosario	Santa Fe

Operador División

La idea básica es "Que todos los elementos cumplan con una propiedad", es equivalente a "Que el conjunto de elementos que no la cumplen este vacio".

Notación: R1 ÷ R2_____ *Operador no conmutativo*

Ventas

Vendedor	Provincia	MontoVendido
Juan	Entre Rios	700000.00
Luis	Santa Fe	300000.00
Maria	Cordoba	384000.00
Pedro	Cordoba	2600000.00
Pedro	Entre Rios	800000.00
Pedro	Santa Fe	200000.00

Provincias

Provincia
Cordoba
Entre Rios
Santa Fe

Ventas % Provincias

Vendedor
Pedro

Para el caso planteado arriba Ventas % Provincias Representa que vendedor vendió en todas las provincias.

Operador División con SQL?

No contamos con un operador especifico para este fin, usando **Not exists** y rompiendonos la cabeza podemos implementarlo.

Se desea conocer? Que vendedor vendió en todas las provincias.

Ventas

Vendedor	Provincia	MontoVendido
Juan	Entre Rios	700000.00
Luis	Santa Fe	300000.00
Maria	Cordoba	384000.00
Pedro	Cordoba	2600000.00
Pedro	Entre Rios	800000.00
Pedro	Santa Fe	200000.00

Provincias

Provincia
Cordoba
Entre Rios
Santa Fe

?

Como sería la solución SQL de → *Ventas % Provincias*

Operador División con SQL?

```
SELECT DISTINCT vendedor
  FROM ventas v1      WHERE NOT EXISTS
        (SELECT *  FROM provincia p
       WHERE NOT EXISTS
         (SELECT *  FROM ventas v2
          WHERE   p.nombre = v2.provincia
            AND   v2.vendedor = v1.vendedor))
```

Provincias en la que el vendedor X NO Vendio

Ventas

Vendedor	Provincia	MontoVendido
Juan	Entre Rios	700000.00
Luis	Santa Fe	300000.00
Maria	Cordoba	384000.00
Pedro	Cordoba	2600000.00
Pedro	Entre Rios	800000.00
Pedro	Santa Fe	200000.00

Provincias

Provincia
Cordoba
Entre Rios
Santa Fe

Ventas % Provincias

Vendedor
Pedro

NOTA: Se muestra solamente a modo de ejemplo del potencial, este caso se resuelve en forma mas sencilla con logica procedural (Stored Procedure).

Ventajas del Modelo

Potencia combinatoria:
Como todas las operaciones toman como entradas relaciones y generan una nueva relación, esto nos permite combinar todos los operadores en una misma sentencia.

Propuesta "100% Declarativa":
Las operadores permiten hacer los requerimientos planteando ***que*** es lo que queremos obtener y **no *el como*** se debe hacer.

En contraposición al enfoque procedural donde se plantea **como** realizarlo. *"Piense en el algoritmo de como realizar una diferencia o reunión".*

Ejercicios Propuestos

Contexto de los ejercicios

Relaciones:

BD2015 BDA2015 Comunicaciones y Redes

Cada relación contiene el esquema:

Legajo	*Int*
ApellidoNombres String	*Varchar(80)*
Sexo String	*Varchar(9)*
Dni Entero	*Int*

Ejercicios Propuestos

Relaciones

Relación BD2015

Legajo	ApellidoNombres	Sexo	Dni
13938	Ariza, Sebastián	Masculino	37080079
13627	Bazan, Joaquín Ernesto	Masculino	36406311
12739	Botteri, Diego Andres	Masculino	23975402
13890	Carfagna, Lautaro	Masculino	36269072
13357	Haliburtón, José María	Masculino	36208541
13292	Militello, Lucas Gonzalo	Masculino	34680097
14206	Narvaez, Germán Simon	Masculino	37080115
14214	Poli, Leandro Hernán Jesús	Masculino	37080584

Relación BDA2015

Legajo	ApellidoNombres	Sexo	Dni
14650	Aguero, Florencia Guadalupe	Femenino	37703384
14156	Brown, Cristhian Jonathan	Masculino	34680475
13920	Cabrera, Rodrigo Emanuel	Masculino	33033886
11521	Castillon, Fernando Ezequiel	Masculino	27087840
14658	De Giusto, Ciro	Masculino	37290343
14319	Falico, Cesar Alejandro	Masculino	37562168
14671	Fenés, Pablo Rodrigo	Masculino	32830747
13893	Fontana, Juan Esteban	Masculino	32016092
14167	Gerfau, Luis Gabriel	Masculino	34014231
14506	Gutierrez, Matias Emmanuel	Masculino	37381136
13357	Haliburtón, José María	Masculino	36208541
14484	Mistrorigo. Bianca Micaela	Femenino	39580654

Ejercicios Propuestos

Relaciones

Relación RedesyComunicaciones2015

Legajo	ApellidoNombres	Sexo	Dni
14650	Aguero, Florencia Guadalupe	Femenino	37703384
13627	Bazan, Joaquín Ernesto	Masculino	36406311
14156	Brown, Cristhian Jonathan	Masculino	34680475
13920	Cabrera, Rodrigo Emanuel	Masculino	33033886
11521	Castillon, Fernando Ezequiel	Masculino	27087840
13893	Fontana, Juan Esteban	Masculino	32016092
14167	Gerfau, Luis Gabriel	Masculino	34014231
14506	Gutierrez, Matias Emmanuel	Masculino	37381136
13357	Haliburtón, José María	Masculino	36208541
14197	Herrmann, Lucas Emilio	Masculino	35296844
14484	Mistrorigo. Bianca Micaela	Femenino	39580654

Ejercicios Propuestos

Se Pide

1. Que Operador Relacional usaría para formar una nueva relación tomando como origen BDA2015 que contenga solo alumnos del sexo femenino.
   Realice el equivalente Sql.

2. Que Operador usaría para redefinir la relación bd2015 de manera que contenga la misma cardinalidad de bd2015 y que contenga solo los atributos legajo y apellidoNombres.
   Realice equivalente Sql.

3. Que Operador usaría para formar una nueva relación con los alumnos de bda2015 y que además sean alumnos en bd2015.
   Realice equivalente Sql.

4. Que Operador usaría para formar una nueva relación con los alumnos que cursan bda2015 y no cursan bd 2015.
   Realice equivalente Sql.

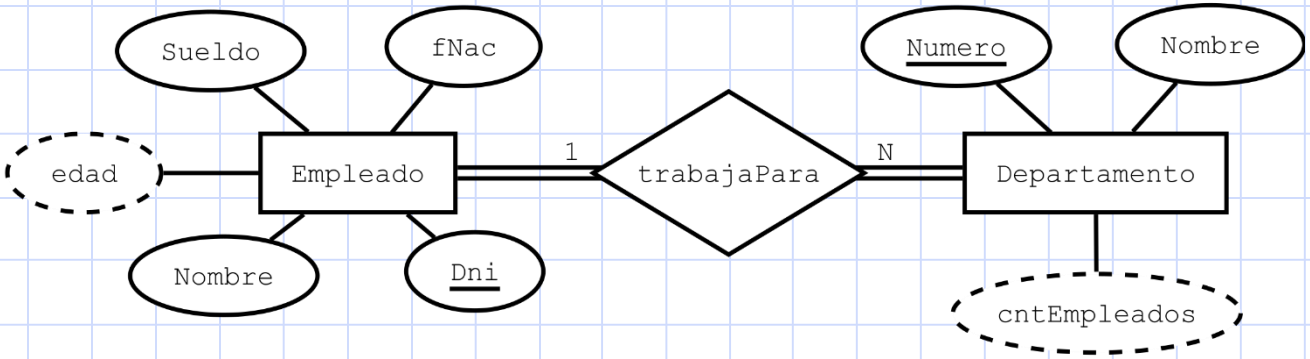
Ejercicios Propuestos

Se Pide

5. Suponga ahora que se posee una nueva relación denominada, **Alumnos**, que contiene**: Legajo, ApellidoNombres, Carrera y FechaInscripcion,**

   Genere un nueva relación con la información de bd2015 reunida con alumnos vinculando estas por su legajo.

Problema – Desde DER → MR



Empleado(dni, fNac, Nombre, Sueldo)
vEmpleado(dni, fNac, Nombre, Sueldo, Edad)

```
CREATE TABLE empleado (
    dni integer,
    nombre character varying(50),
    sueldo numeric(10,2),
    fnac date,
    PRIMARY KEY (dni)
)
```

```
create view vEmpleado
as
  select empleado.*,
    age(fnac) as edad
  from empleado
```

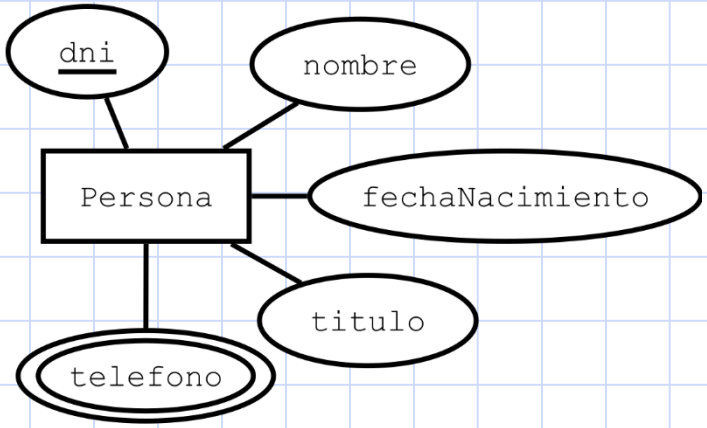
Diferencias en modelos

- El concepto de Relación es distinto en ambos, además no existe el concepto de entidad.
- MR no admite atributos multivalorados.
- MR no admite atributos derivados.
- MER describe el conjunto de claves candidatas y no existe el concepto de clave principal.

Algoritmo de pasaje MER ➔ MR

❑ **Caso 1: Mapeo de Entidades Regulares**

Por cada entidad (fuerte) regular E del esquema ER, cree una relación R que incluya todos los atributos simples de E. Incluya únicamente los atributos simples que conforman un atributo compuesto. Seleccione uno de los atributos clave de E como clave principal para R. Si la clave elegida de E es compuesta, entonces el conjunto de los atributos simples que la forman constituirán la clave principal de R.



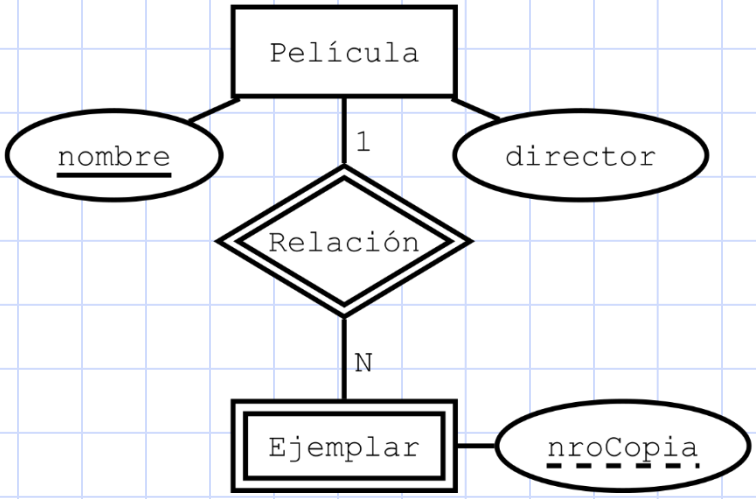
Persona(dni, nombre, fecnac, titulo)

Nota: Si hubiera varias claves candidatas para E, las secundarias constituirán restricción **unique.**

Algoritmo de pasaje MER ➔ MR

- **Caso 2: Mapeo de Entidades Débiles**

Por cada tipo de entidad débil W del esquema ER con el tipo de entidad propietario E, cree una relación R e incluya todos los atributos simples (o componentes simples de los atributos compuestos) de W como atributos de R. Además, incluya como atributos de la foreign key de R, el(los) atributo(s) de la o las relaciones que correspondan al o los tipos de entidad propietarios; esto se encarga de identificar el tipo de relación de W. La clave principal de R es la combinación de la(s) clave(s) principal(es) del o de los propietarios y la clave parcial del tipo de entidad débil W, si la hubiera.

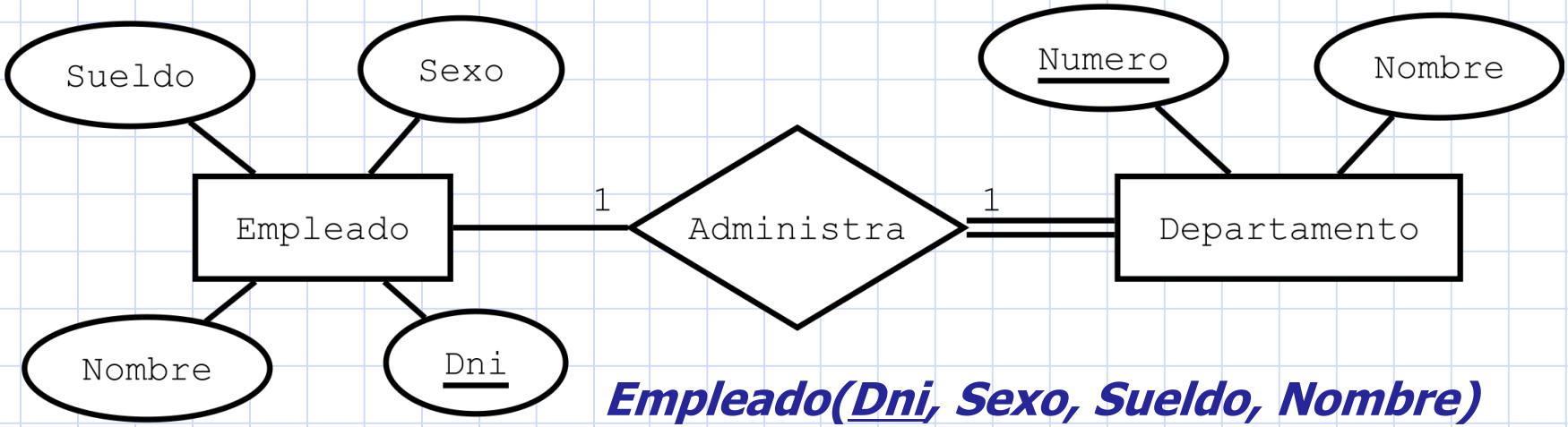


Ejemplar(nombrePel, nroCopia)

Algoritmo de pasaje MER ➔ MR

❑ **Caso 3: Mapeado de los tipos de relación 1:1 binaria. "Foreing Key"**

Seleccione una de las relaciones (por ejemplo, S) e incluya como *foreign key* en S la clave principal de *T.* Lo mejor es elegir un tipo de entidad con *participación total* en *R* en el papel de S. Incluya todos los tributos simples (o los componentes simples de los atributos compuestos) del tipo de relación 1: 1 *R* como atributos de S.

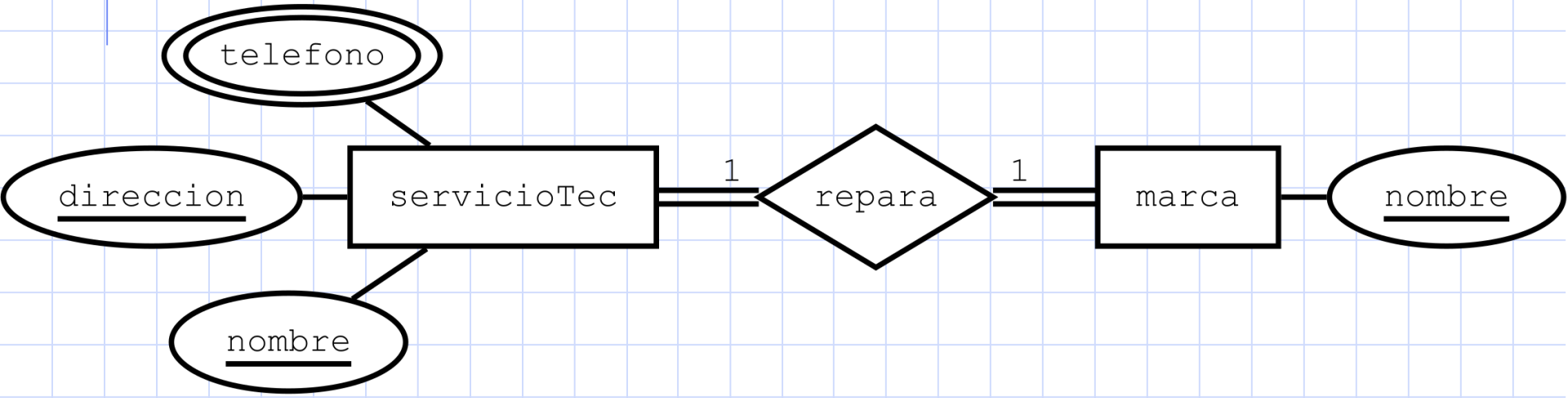


Empleado(Dni, Sexo, Sueldo, Nombre)
Departamento(Numero, Nombre, DniAdm)
FK(DniAdm) reference Empleado(Dni)

Algoritmo de pasaje MER ➔ MR

❑ **Caso 3: Mapeado de los tipos de relación 1:1 binaria. "Relación Mezclada"**

Una asignación alternativa de un tipo de relación 1: 1 es posible al mezclar los dos tipos de entidad y la relación en una sola relación. **Esto puede ser apropiado cuando *las dos participaciones son totales.***

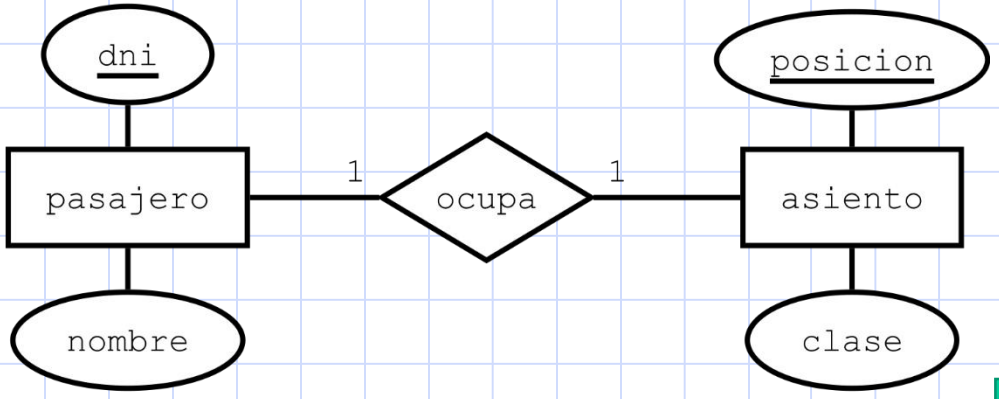


ServicioTecMarca(direc, nombreSrv, marcaNombre)

Algoritmo de pasaje MER ➔ MR

❑ **Caso 3: Mapeado de los tipos de relación 1:1 binaria. "Relación de relación"**

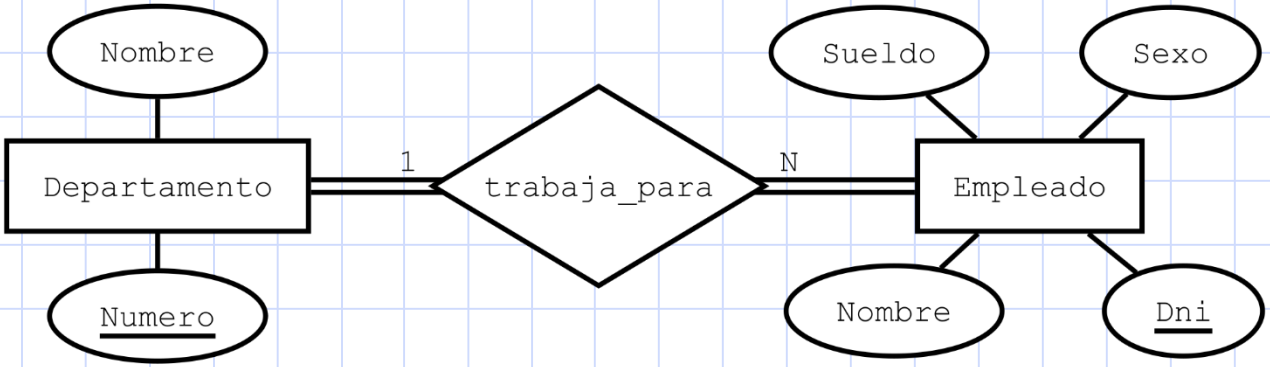
Consiste en configurar una tercera relación *R* con el propósito de crear una referencia cruzada de las claves principales de las relaciones S y *T* que representan los tipos de entidad. Como veremos, esta metodología es necesaria para las relaciones M:N binarias. La relación *R* se denomina relación de relación (y, en algunas ocasiones, tabla de búsqueda), porque cada tupla de *R* representa una instancia de relación que relaciona una tupla de S con otra de *T.*



Pasajero(dni, nombre)
Asiento(posición,clase)
Ocupa(posición,dni)

Algoritmo de pasaje MER ➔ MR

❑ **Caso 4: Mapeado de tipos de relaciones 1:N binarias.**
Identifique la relación S que representa el tipo de entidad participante en el lado *N* del tipo de relación. Incluya *como foreign key* en S la clave principal de la relación *T* que representa el otro tipo de entidad participante en *R;* Incluya todos los atributos del tipo de relación l:N como atributos de S.

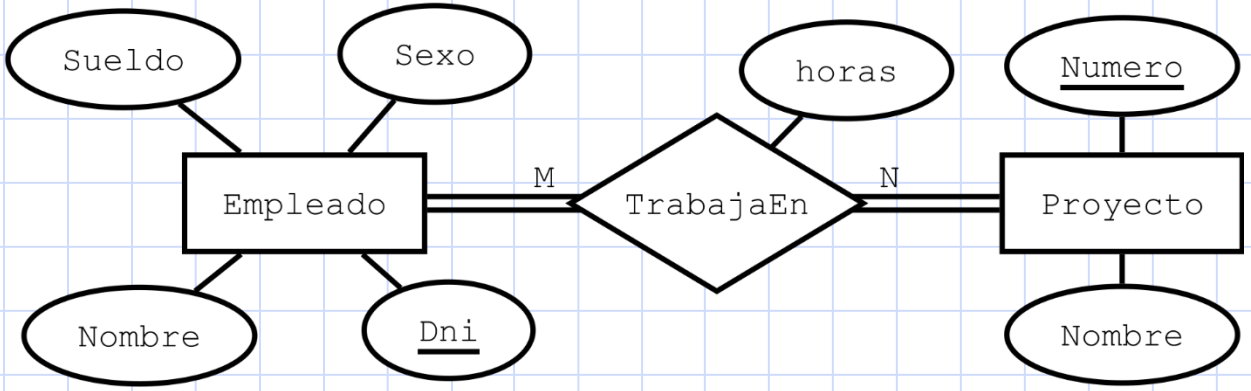


Empleado(Dni, Sexo, Sueldo, Nombre, nroDepto)
FK(nroDepto) reference Departamento(Numero)
Departamento(Numero, Nombre)

Algoritmo de pasaje MER ➔ MR

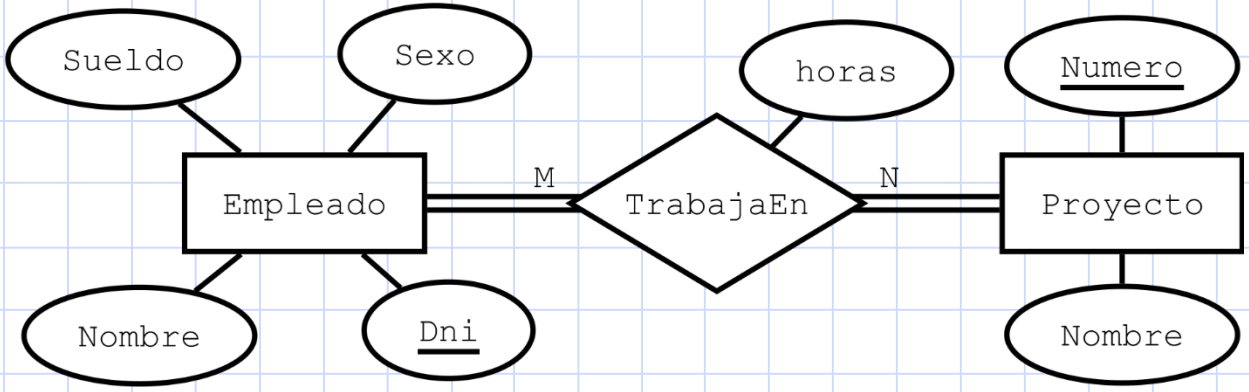
- **Caso 5: Mapeado de tipos de relaciones M:N binarias**

Por cada tipo de relación M:N binaria *R,* cree una nueva relación S para representar a *R.* Incluya como atributos de *la foreign key* en S las claves principales de las relaciones que representan los tipos de entidad participantes; su combinación formará la clave principal de S. Incluya también cualesquiera atributos simples del tipo de relación M:N (o los componentes simples de los atributos compuestos) como atributos de S. No podemos representar un tipo de relación M:N con un atributo de *foreign key* en una de las relaciones participantes (como hicimos para los tipos de relación 1: 1 o 1 :N) debido a la razón de cardinalidad M:N; debemos crear una *relación de relación* S separada.



Algoritmo de pasaje MER ➔ MR

- **Caso 5: Mapeado de tipos de relaciones M:N binarias**

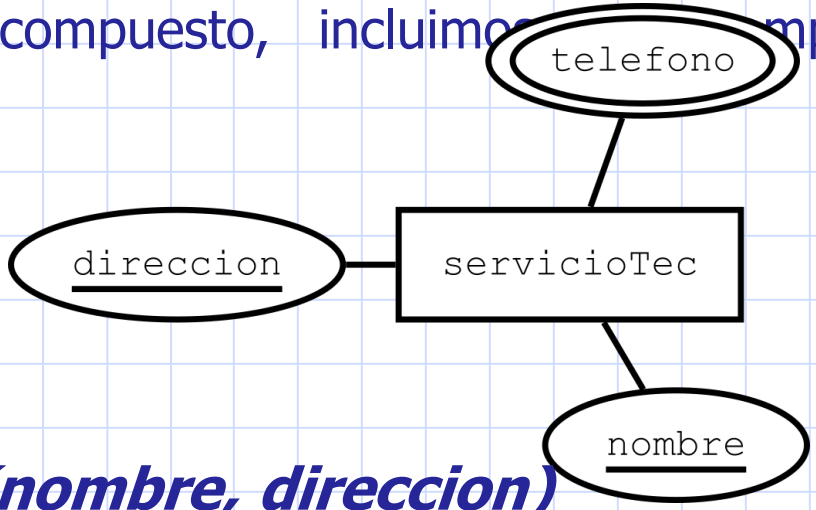


Empleado(Dni, Sexo, Sueldo, Nombre)
Proyecto(Numero, Nombre)
ProyectoEmpleado(Dni, Proyecto, horas)
FK(Dni) reference Empleado(Dni)
FK(Proyecto) reference Proyecto(Numero)

Algoritmo de pasaje MER ➔ MR

❑ **Caso 6: Mapeado de atributos multivalor.**

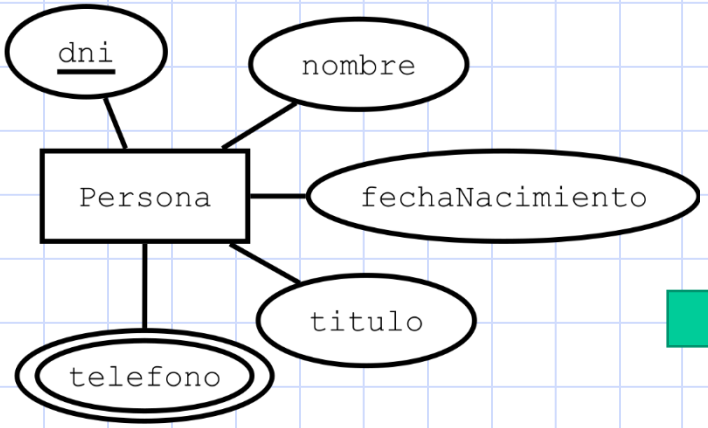
Por cada atributo multivalor *A,* cree una nueva relación *R.* Esta relación incluirá un atributo correspondiente a *A,* más el atributo clave principal *K* (como *foreign key* en *R)* de la relación que representa el tipo de entidad o tipo de relación que tiene *A* como un atributo. La clave principal de *R* es la combinación de *A* y *K.* Si el atributo multivalor es compuesto, incluimos [illegible]mponentes simples.



servicioTec(nombre, direccion)
telefonoSrv(nombreSrv, numero)
FK(nombreSrv) reference servicioTec(nombre)

Algoritmo de pasaje MER ➔ MR

- **Caso 6': Mapeado de atributos multivalor.**
  Aplicado del caso a `Persona.telefono`.



Persona(dni, nombre, fecnac, titulo)

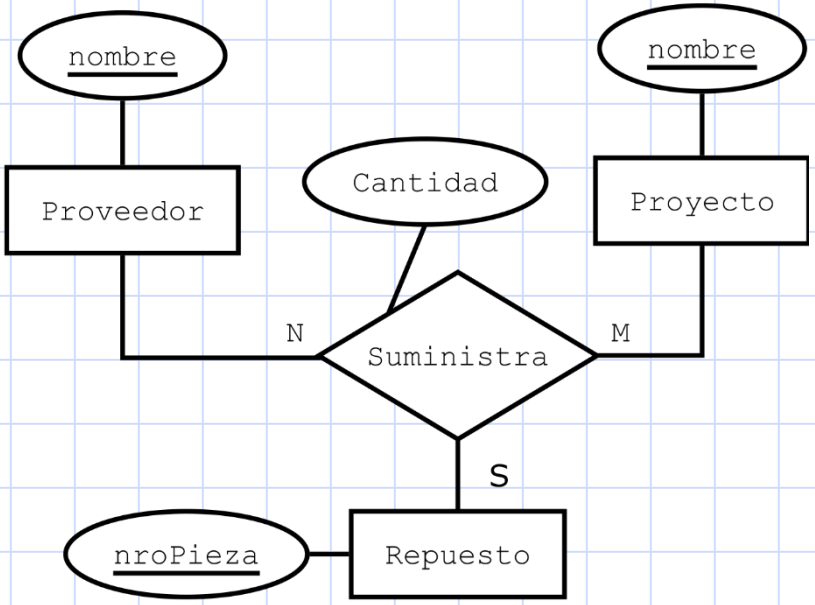
PersonaTelefono(dni, telefono)

Algoritmo de pasaje MER ➔ MR

❑ **Caso 7: Mapeado de los tipos de relación *n-ary.***

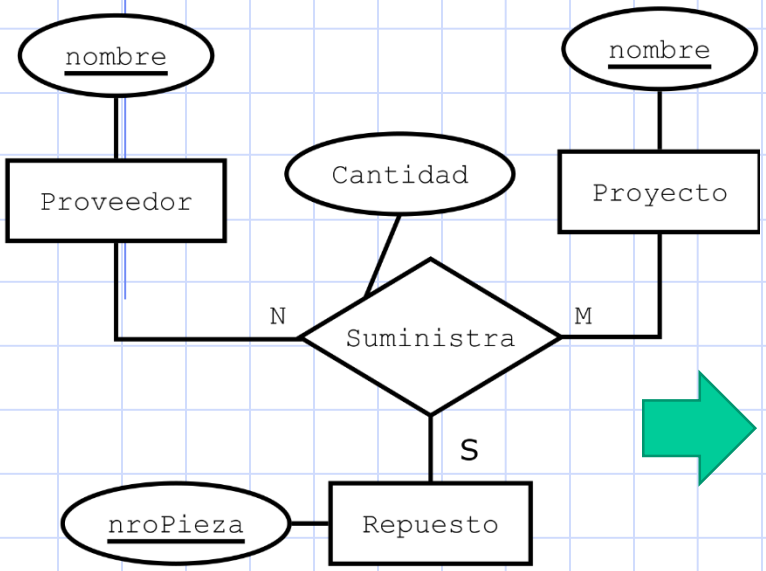
Por cada tipo de relación n-ary *R,* donde *n* > 2, cree una nueva relación S para representar *R.* Incluya como atributos de *la foreign key* en S las claves principales de las relaciones que representan los tipos de entidad participantes. Incluya también cualesquiera atributos simples del tipo de relación n-my como atributos de S.

Normalmente, la clave principal de S es una combinación de todas *lasforeign keys* que hacen referencia a las relaciones que representan los tipos de entidad participantes.



Algoritmo de pasaje MER ➔ MR

- **Caso 7: Mapeado de los tipos de relación *n-ary.***

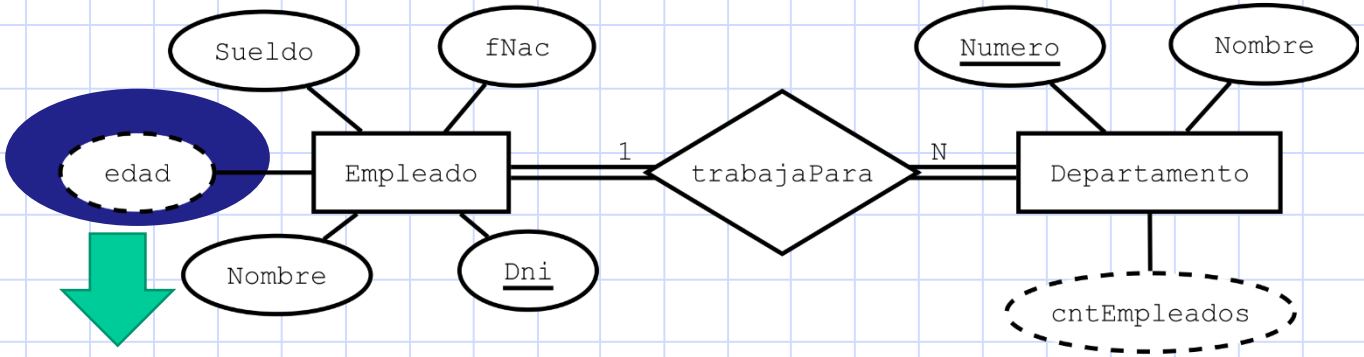


Proveedor(<u>nombre</u>)
Proyecto(<u>nombre</u>)
Repuesto(<u>nroPieza</u>)
provProySum(<u>nProv</u>, <u>nPry</u>, <u>idPieza</u>, cnt)

FK(nProv) reference Proveedor(nombre)
FK(nPry) reference Proyecto(nombre)
FK(idPieza) reference Repuesto(nroPieza)

Algoritmo de pasaje MER ➔ MR

- **Alternativa mapeo atributos derivados**



Empleado(dni, fNac, Nombre, Sueldo) *(relación base)*

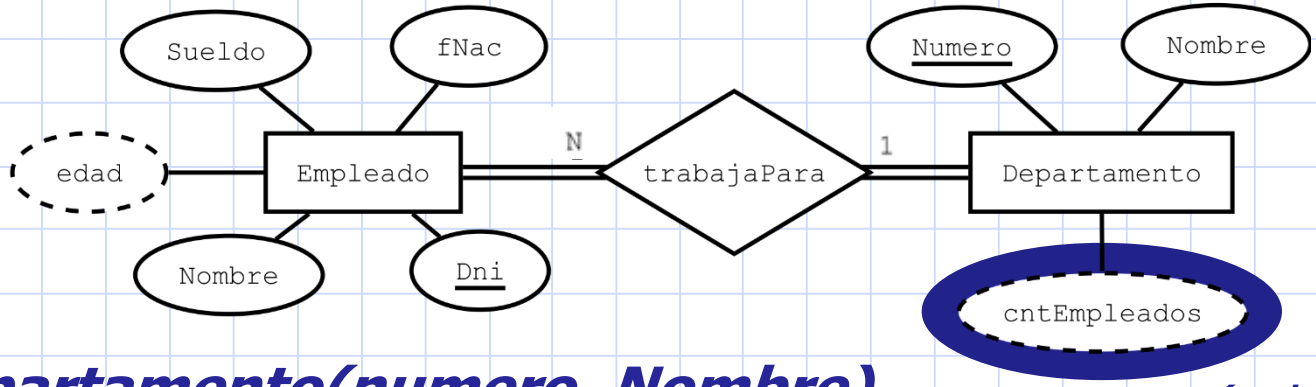
vEmpleado(dni, fNac, Nombre, Sueldo, Edad) *(relación virtual)*

```
CREATE TABLE empleado (
    dni integer,
    nombre character varying(50),
    sueldo numeric(10,2),
    fnac date,
    PRIMARY KEY (dni)
)
```

```
create view vEmpleado
as
  select empleado.*,
    age(fnac) as edad
  from empleado
```

Algoritmo de pasaje MER ➔ MR

- **Alternativa mapeo atributos derivados**



Departamento(numero, Nombre) *(relación base)*

vDeptartamento(numero ,Nombre, cntEmpleados)

```
create table departamento (
  nombre varchar (50),
  numero integer,
  primary key (numero)
)
alter table empleado
  add column dptoNro integer;
```

```
create view
vDepartamvirtual)
ento as
  select d.*, (select
count(*) from empleado e
 where e.dptoNro=d.numero)
    as cntEmpleados
 from departamento d
```

Resumen

- Revisión de Modelado y DER
- Revisión del Modelo Relacional
- Revisión del Algebra Relacional
- Normalización
- DMLs de consultas
- DMLs de actualización

Diseño de Bases de Datos

¿Qué es el Diseño de Bases de Datos Relacionales?

Informalmente es la agrupación de atributos para formar esquemas de relaciones de "calidad".

Hay dos niveles de Esquemas de Relación

- El nivel lógico "vistas de usuario" →Relaciones Virtuales.
- El nivel de almacenamiento "tablas" →Relaciones Base.

*El Diseño se refiere principalmente a las **Relaciones Base**, Como objetivo secundario las vistas completan el esquema externo, lo visible de la Base de Datos para los distintos tipos de Usuarios.*

Diseño de Bases de Datos

Existen pautas informales para el diseño.

- Asegurarse de que la semántica de los atributos esté clara
- Reducir la información redundante en tuplas
- Reducir valores NULL en tuplas
- No permitir la posibilidad de generar tuplas falsas

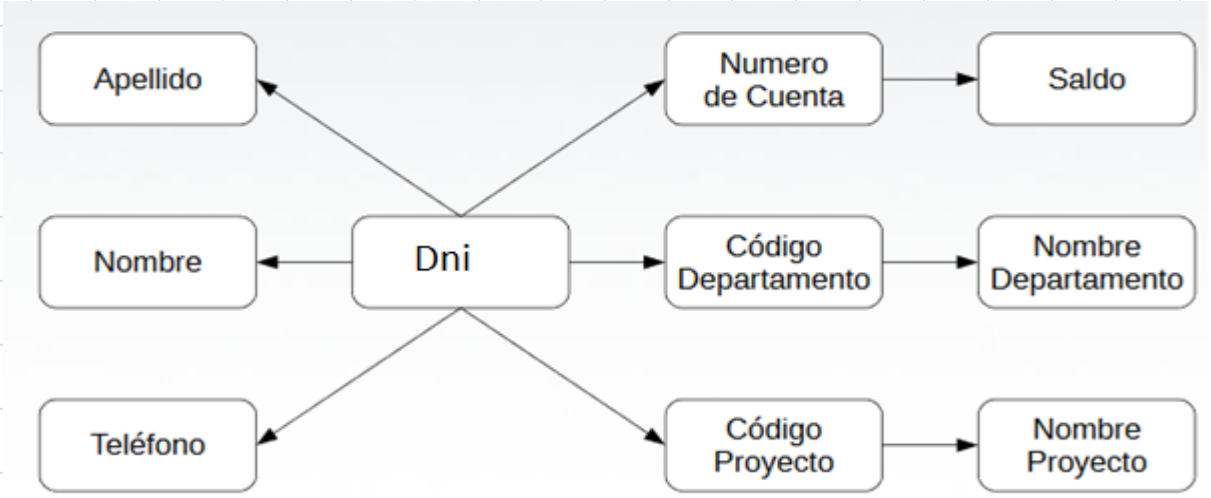
Repasamos los conceptos formales de dependencias funcionales y formas normales:

- 1NF (primera forma normal)
- 2NF (segunda forma normal)
- 3NF (tercera forma normal) Codd 1970

- BCNF (forma normal de Boyce-Codd)
  Boyce/Codd 1974

Dependencias Funcionales (DF)

Concepto: Las DF son restricciones de integridad que **nos permiten conocer que interrelaciones existen entre dos o más atributos** del mundo real.



Son inherentes al contenido semántico de los datos, que se han de cumplir para cualquier ocurrencia del esquema de relación.

Dependencias Funcionales (DF)

Supongamos que nuestro esquema de BD relacional tiene n atributos A1, A2, …, An,, pensemos que la BD completa está descrita por un:

único esquema universal R(A1, A2, …, An,).

Definición: Una DF, denotada por X →, entre dos conjuntos de atributos X e Y que son subconjuntos de R, especifica una restricción en las posibles tuplas que pueden formar un estado de relación r de R. La restricción dice que dos tuplas t1 y t2 en r que cumplen que t1[X]= t2[X] deben cumplir también que t1[Y]= t2[Y].

Dep Func(DF) – Caso de Estudio

Se cuenta con información de Proyectos de Investigación de una universidad, caracterizada por el numero y nombre de cada proyecto y la facultad donde se desarrolla y los rrhh que intervienen (dni, nombre y horas de afectación al proyecto).

EMP_PROY		Proyectos				
Dni	NombreE	NumProyecto	NombreP	Facultad	FacDirección	Horas

Restricciones detectadas:

- EMP_PROY cuenta con una ocurrencia por empleado
- Un empleado participa en 0,1 o mas proyectos
- Un proyecto se desarrolla en una y solo una facultad

Dep Func(DF) – Caso de Estudio

EMP_PROY		Proyectos				
Dni	NombreE	NumProyecto	NombreP	Facultad	FacDirección	Horas

EMP_PROY (Ocurrencias)

Dni	NombreE	NumProyecto	NombreP	Facultad	FacDirección	Horas
12345678	Pérez, José	1 2	Auto Solar Algebra Lineal	Paraná Santa Fe	Almafuerte 1033 Lavaisse 610	32 8
17044666	Ojeda, Fernando	5	Acción del Viento	C.Uruguay	Ing Pereyra 676	20
…..	….	…				…
29541266	Pajares, Luis	1 3	Auto Solar Casa Social	Paraná Santa Fe	Almafuerte 1033 Lavaisse 610	10 5

Dep Func(DF)

En el caso de estudio

EMP_PROY		Proyectos				
Dni	NombreE	NumProyecto	NombreP	Facultad	FacDirección	Horas

El Dni y nro de proyecto determinan las horas por semana que el empleado trabaja en el proyecto:

- (Dni, NumProyecto) →Horas X=(Dni,NumProy) Y=Horas

El nro de proyecto determina su nombre y la facultad donde se desarrolla:

- NumProyecto → (Nombrep,Facultad)
  X=NumProy Y=(Nombrep,Facultad)

Dep Func(DF)

En el caso de estudio

EMP_PROY		Proyectos				
Dni	NombreE	NumProyecto	NombreP	Facultad	FacDirección	Horas

- (Dni, NumProyecto) →Horas X=(Dni,NumProy) Y=Horas

Decimos:

- **X** se denomina **determinante**, también **lado izquierdo**.
- **Y** se denomina **dependiente**, también **lado derecho**.
- **Y** depende funcionalmente de **X**.
- Cada valor de **X** tiene asociado siempre el mismo valor de **Y** en una relación **R**, que contiene a **X** e **Y** como atributos o conjuntos de atributos.
- Si **K** es una clave de **R, K** determina funcionalmente todos los atributos de **R**.

DF – concepto de **Clave**

Una **Superclave** de una relación R(A1,A2,…,An), es un conjunto de uno o más atributos
$S \subseteq R$ que nos permiten identificar inequívocamente una instancia de **R**.

Personas

Nombre	Dni	FechaNac	Sexo	Cuil

- Conjunto1(Dni,Cuil)
- Conjunto2(Nombre,Dni)
- Conjunto3(Dni)
- ~~Conjunto4(Fecha,Fexo)~~ **NO es una POSIBLE CLAVE**
- Conjunto5(Cuil)

DF – Concepto de **Clave**

Una **clave K** de una relación **R**, es una superclave de **R**, con la propiedad adicional de que la eliminación de cualquier atributo **A** de **K** provocará que **K** deje de ser una superclave.

Personas

Nombre	Dni	FechaNac	Sexo	Cuil

¿Cuál de estas sería una clave?

- SuperClave1(Dni,Cuil)
- SuperClave2(Nombre,Dni)
- SuperClave3(Dni)
- SuperClave4(Cuil)

← Será clave si eliminamos cualquiera de sus componentes y no sigue siendo clave.

La diferencia entre una clave y superclave es que la clave tiene que ser **mínima**.

DF – Concepto de **Clave**

Claves candidatas: cada clave es una clave candidata (a ser clave primaria).

Personas

Nombre	**Dni**	FechaNac	Sexo	Cuil

Claves candidatas
Clave1(Dni)
Clave2(Cuil)

Clave Primaria, se representa subrayando todos sus atributos.
¿Como se representan las secundarias?

Clave primaria: Es una de las claves candidata que el diseñador elige.

El resto de estas claves seran claves secundarias.

Por Ejemplo: **ClavePrimaria(Dni)** **ClaveSecundaria(Cuil)**

DF – Concepto de **Clave**

Atributo Primo es un atributo de R que es miembro de alguna clave candidata.

Personas

Nombre	**Dni**	FechaNac	Sexo	Cuil

Claves candidatas
Clave1(Dni)
Clave2(Cuil)

Atributo No Primo es un atributo de R que no es miembro de una clave candidata.

Para este caso, tanto **CUIL** como **DNI** son atributos primos.

Primera Forma Normal

- La **1FN** es una restricción que se basa en **NO PERMITIR** atributos multivaluados ni compuestos, es decir, relaciones dentro de relaciones.
- Considerado como parte de la definición de Relación.
- La 1FN establece que los dominios de los atributos deben incluir valores atómicos.

Ejemplo

EMP_PROY (Dni, NombreE,
{PROYS(NumProy,NombreP,Facultad,FacDirección,Horas)})

multivalores: *Se representan entre llaves*

Primera Forma Normal

(a)

EMP_PROY

		Proys				
Dni	NombreE	NumProyecto	NombreP	Facultad	FacDirección	Horas

(b)

EMP_PROY

Dni	NombreE	NumProyecto	NombreP	Facultad	FacDirección	Horas
12345678	Pérez, José	1	Auto Solar	Paraná	Almafuerte 1033	32
		2	Algebra Lineal	Santa Fe	Lavaisse 610	8
17044666	Ojeda, Fernando	5	Acción del Viento	C.Uruguay	Ing Pereyra 676	20
.....	....	...				...
29541266	Pajares, Luis	1	Auto Solar	Paraná	Almafuerte 1033	10
		3	Casa Social	Santa Fe	Lavaisse 610	5

(c)

EMP_PROY

Dni	NombreE	NumProyecto	NombreP	Facultad	FacDirección	Horas
12345678	Pérez, José	1	Auto Solar	Paraná	Almafuerte 1033	32
12345678	Pérez, José	2	Algebra Lineal	Santa Fe	Lavaisse 610	8
17044666	Ojeda, Fernando	5	Acción del Viento	C.Uruguay	Ing Pereyra 676	20
.....	....	...	...	...	...	...
29541266	Pajares, Luis	1	Auto Solar	Paraná	Almafuerte 1033	10
29541266	Pajares, Luis	3	Casa Social	Santa Fe	Lavaisse 610	5

Referencia Figura

Normalización 1FN.

(a) Una Relación que no esta en 1FN.

(b) Ejemplo de intancias de la relación.

(c) Versión a 1FN de la misma relación con redundancia.

Primera Forma Normal

(a)

EMP_PROY

		Proys				
Dni	NombreE	NumProyecto	NombreP	Facultad	FacDirección	Horas

EMP_PROY (Dni, NombreE, {PROYS(NumProy, NombreP, Fac, FacDir, Horas)})

(b)

EMP_PROY1

Dni	NombreE

EMP_PROY1 (Dni, NombreE)

EMP_PROY2

Dni	NumProyecto	NombreP	Facultad	FacDirección	Horas

EMP_PROY2 (Dni, NumProy, NombreP, Facultad, FacDirección, Horas)

Primera Forma Normal

(a)

EMP_PROY

Dni	NombreE	NunProyecto	NombreP	Facultad	FacDirección	Horas
12345678	Pérez, José	1	Auto Solar	Paraná	Almafuete 1033	32
		2	Algebra Lineal	Santa Fe	Lavaisse 610	8
17044666	Ojeda, Fernando	5	Acción del Viento	C.Uruguay	Ing. Pereyra 676	20
.....	....	...	...	...	...	...
29541266	Pajares, Luis	1	Auto Solar	Paraná	Almafuete 1033	10
		3	Casa Social	Santa Fe	Lavaisse 610	5

Referencia de Datos
Normalización 1FN.
(a) Relación **no** en 1FN.
(b) Relaciones en 1FN.

(b)

EMP_PROY1

Dni	NombreE
12345678	Pérez, José
17044666	Ojeda, Fernando
.....	....
29541266	Pajares, Luis

EMP_PROY2

Dni	NumProyecto	NombreP	Facultad	FacDirección	Horas
1	1	Auto Solar	Paraná	Almafuerte 1033	32
2	2	Algebra Lineal	Santa Fe	Lavaisse 610	8
5	5	Acción del Viento	C.Uruguay	Ing Pereyra 676	20
...	...		...		...
5	1	Auto Solar	Paraná	Almafuerte 1033	10
3	3	Casa Social	Santa Fe	Lavaisse 610	5

Dni es parte de la Clave Primaria(PK), tambien FK de EMP_PROY1

Clave Primaria (PK)

Segunda Forma Normal

- Una relación R está en la segunda forma normal (2NF) si esta en 1FN y además cada atributo **no primo** A en R depende completa y funcionalmente de la clave primaria.
- R se puede descomponer en relaciones 2NF a través del proceso de normalización 2NF que analizaremos a continuación.

Segunda Forma Normal

- Utiliza los conceptos clave primaria y de DF.
- Recordamos

  Atributo primo: un atributo que es miembro de la clave primaria K.
  DF Completa: un DF Y → Z donde la eliminación de cualquier atributo de Y significa que la DF ya no es válido

Ejemplos:

- {Dni, NumProyecto} → Horas es un **DF completo** ya que ni Dni → Horas ni NumProyecto → Horas se mantienen quitando NumProy y Dni alternativamente.
- {Dni, NumProyecto} → NombreP **NO ES** un DF completo (tiene dependencia parcial) ya que NumProyecto → NombreP es una FD.

¿Facultad y FacDir?

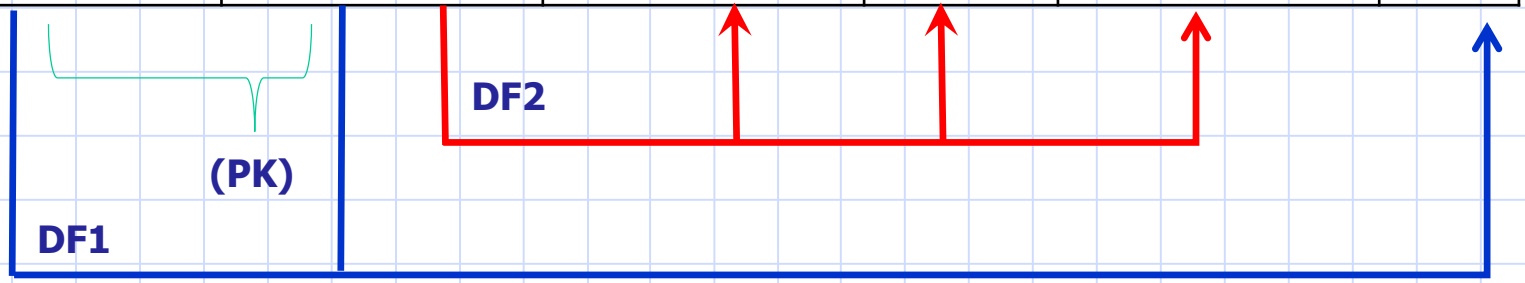
EMP_PROY2

Dni	NumProyecto	NombreP	Facultad	FacDirección	Horas

Segunda Forma Normal

EMP_PROY2

Dni	NumProyecto	NombreP	Facultad	FacDirección	Horas
1	1	Auto Solar	Paraná	Almafuerte 1033	32
2	2	Algebra Lineal	Santa Fe	Lavaisse 610	8
5	5	Acción del Viento	C.Uruguay	Ing Pereyra 676	20
…	…		…		…
1	1	Auto Solar	Paraná	Almafuerte 1033	10
3	3	Casa Social	Santa Fe	Lavaisse 610	5



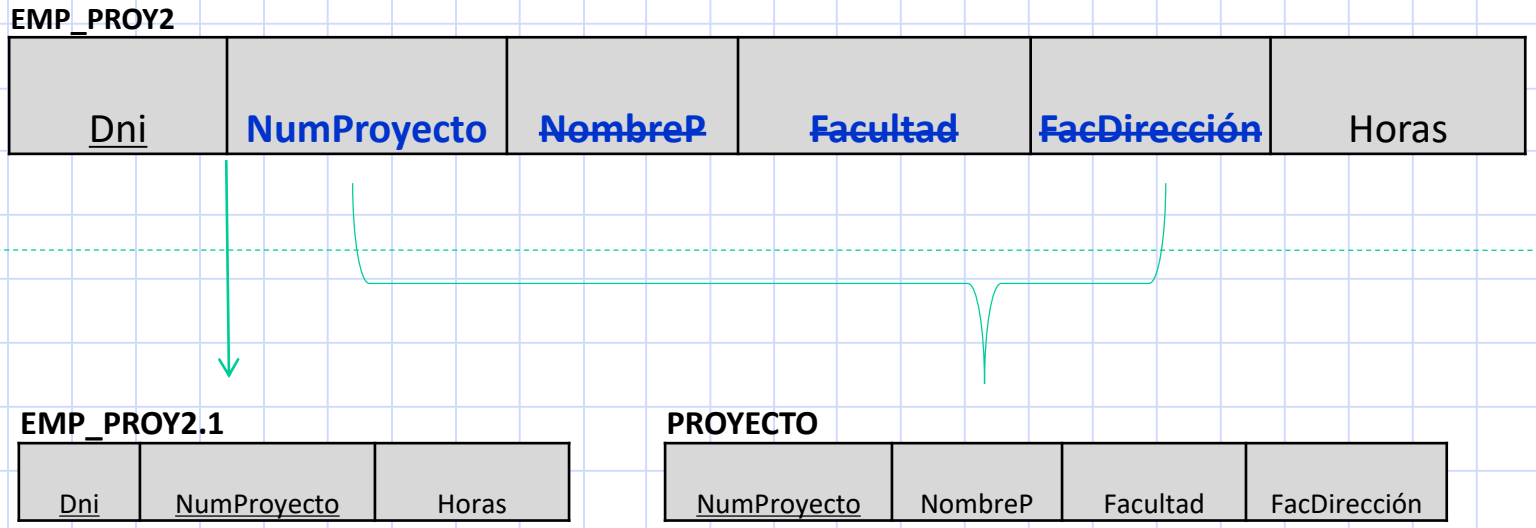
DF2: determina una dependencia parcial, ya que NombreP, Facultad y Dirección de Facultad no dependen de DNI, solo de Numero de Proyecto.

→ EMP_PROY2 no esta en 2FN

EMP_PROY2

Dni	NumProyecto	~~NombreP~~	~~Facultad~~	~~FacDirección~~	Horas

Segunda Forma Normal



Segunda Forma Normal

EMP_PROY2 (1FN)

Dni	NumProyecto	NombreP	Facultad	FacDirección	Horas
1	1	Auto Solar	Paraná	Almafuerte 1033	32
2	2	Algebra Lineal	Santa Fe	Lavaisse 610	8
5	5	Acción del Viento	C.Uruguay	Ing Pereyra 676	20
...	...		...		...
1	1	Auto Solar	Paraná	Almafuerte 1033	10
3	3	Casa Social	Santa Fe	Lavaisse 610	5

EMP_PROY2_1 (2FN)

Dni	NumProyecto	Horas
1	1	32
2	2	8
5	5	20
...	...	
1	1	10
3	3	5

PROYECTOS (2FN)

NumProyecto	NombreP	Facultad	FacDirección
1	Auto Solar	Paraná	Almafuerte 1033
2	Algebra Lineal	Santa Fe	Lavaisse 610
5	Acción del Viento	C.Uruguay	Ing Pereyra 676
3	Casa Social	Santa Fe	Lavaisse 610

EMP_PROY2 por DF2 se descompone en:

- PROYECTOS
- EMP_PROY2.1

Tercera Forma Normal

- Una relación R está en la tercera forma normal (3FN) si esta en 2FN y además ningún atributo **no primo** A en R es transitivamente dependiente de la clave principal.

Dicho de otra forma, una relación R esta en 3FN si esta en 2FN y además, no existen dependencias funcionales entre atributos no primos.

- R se puede descomponer en relaciones 3NF a través del proceso de normalización 3NF que analizaremos a continuación.

Tercera Forma Normal

PROYECTO

NumProyecto	NombreP	Facultad	FacDirección
1	Auto Solar	Paraná	Almafuerte 1033
2	Algebra Lineal	Santa Fe	Lavaisse 610
5	Acción del Viento	C.Uruguay	Ing Pereyra 676
3	Casa Social	Santa Fe	Lavaisse 610

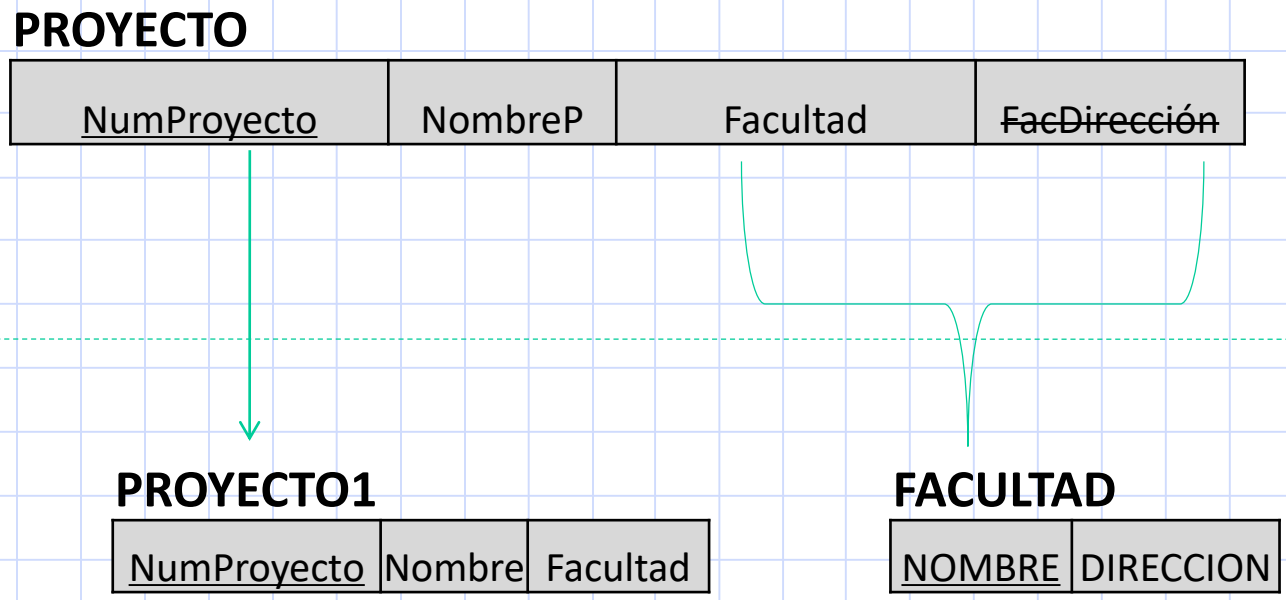
DF3: determina una dependencia funcional entre Facultad y su ubicación (FacDirección). → PROYECTO no esta en 3FN

PROYECTO1

NumProyecto	NombreP	Facultad	~~FacDirección~~

Tercera Forma Normal

FD3 determina que → PROYECTO no esta en 3FN



Tercera Forma Normal

PROYECTO

NumProyecto	NombreP	Facultad	FacDirección
1	Auto Solar	Paraná	Almafuerte 1033
2	Algebra Lineal	Santa Fe	Lavaisse 610
5	Acción del Viento	C.Uruguay	Ing Pereyra 676
3	Casa Social	Santa Fe	Lavaisse 610

PROYECTO1

NumProyecto	NombreP	Facultad
1	Auto Solar	Paraná
2	Algebra Lineal	Santa Fe
5	Acción del Viento	C.Uruguay
3	Casa Social	Santa Fe

FACULTAD

Nombre	Dirección
Paraná	Almafuerte 1033
Santa Fe	Lavaisse 610
C.Uruguay	Ing Pereyra 676

PROYECTO por DF3 se descompone en:

- PROYECTO1
- FACULTAD

RESUMEN DEL PROCESO a 3FN

(a)

EMP_PROY

		Proyectos				
Dni	NombreE	NumProy	NombreP	Facultad	FacDirección	Horas

(b)

EMPLEADO

Dni	Nombre

(c)

PROYECTO

Numero	Nombre	Facultad

(d)

FACULTAD

Nombre	Dirección

(e)

EMP_PROY

Dni	NumProyecto	Horas

Referencias:

(a) Esquema de Relación Universal de partida.

(b) , (c), (d), (e): Relaciones en 3FN que representan el esquema de partida.

Parafraseando a KENT

La normalizacion

Debe cumplir
"Que cada atributo dependa":

- ✓ de la clave (1FN)
- ✓ de toda la clave (2FN)
- ✓ y nada mas que de la clave (3FN)

Resumen

- Revisión de Modelado y DER
- Revisión del Modelo Relacional
- Revisión del Algebra Relacional
- Normalización
- DMLs de consultas
- DMLs de actualización

Introducción a SQL

- **SQL** (Structured Query Language)**.**
- Es el Lenguaje mas importante en servidores relacionales.
- Es totalmente Declarativo.
- Standard de Mercado.

Sub Lenguajes

Se compone de 3 partes:

DDL (Data Definition Language).

DML (Data Manipulation Language).

DCL (Data Control Language).

Extensiones y Restricciones

SQL implementa SELECT con un conjunto de Extensiones y Restricciones.

Las Restricciones nos permiten saber que podemos utilizar concretamente en:

- Proyección
- From
- Restricción

Las Extensiones permiten realizar agrupamientos, expandir las reuniones, y potenciar aún mas cada parte integrante de un Query.

Extensiones de Proyección

Proyección

Select *lista_proyección*

from *lista_relaciones*

Lista_relaciones

Tablas (Relaciones Base)

Vistas (Relaciones Virtuales)

Tablas Derivadas (Relaciones Resultado)

Procedimientos (Stored Procedure)

Nota: Todas las listas se separan por coma.

Extensiones de Proyección (II)

Proyección

Select *lista_proyección*

from *lista_relaciones*

Lista_proyección

Columnas (de algún miembro lista de relaciones)

Constantes

Operadores (+, - , *, /) (|| , SUBSTRING)

Funciones Internas (Cast. Current_DATEs, CASE, etc)

UDF SUBSELECT de card 1 grado 1

Funciones agregación

Extensiones de Proyección (III)

- Funciones Internas y operador de renombre
- Case: Función Similar a switch de java

  case when condicion1 **then** proy 1[**when** condicion2 **then** proy2] **else** proy3 **end**

- Cast: Función de cambio de tipo

  cast(columna **as** tipo_de_dato**)**

- Extract:

  extract(month **from** columna_tipo_fecha**)** use: month or day or year

- Current's:

  current_date, current_time, current_timestamp, current_user

- Renombrado de atributos: operador **AS**

Extensiones Proyección (IV)

Proyección con filas únicas

Select [distinct] *lista_proyección*

from *lista_relaciones*

Lista_proyección

Elimina filas repetidas de la proyección.

Debe estar solo una vez, después de la cláusula select.

Extensiones de Restricción

Restricción

Select *lista_proyección*
from *lista_relaciones*
where

Condición

Operadores Lógicos: <, >, <=, >=, =, !=,

Compuestas not, and, or between

Operador especial like(%,_) similar to(%,_,+,?,*,[],etc) ver http://es.wikipedia.org/wiki/Expresiones_regulares

Operador de conjunto in (comprueba si es miembro del conjuto)

Se pueden emplear casi los mismos elementos que se proyectan, con excepción de Agregación.

Group By – Fxs agregate

Agrupamiento

Select *lista_proyección*
from *lista_relaciones*
[where *condicion*]
[group by *lista_de_grupo*]

Funciones de agregación:

Count count(distinct columna) Sum *para numéricos*

Min max Avg *para numéricos*

Las funciones de agregación sin group by toman un solo grupo (toda las filas involucradas)

Solo se pueden proyectar: lista de grupo y funciones de agregación.

Group By – Fxs agregate

Agrupamiento

Select *lista_proyección*
from *lista_relaciones*
[where *condicion*]
[group by *lista_de_grupo*]

Funciones de agregación:

Count count(distinct columna) Sum *para numéricos*

Min max Avg *para numéricos*

Las funciones de agregación sin group by toman un solo grupo (toda las filas involucradas)

Solo se pueden proyectar: lista de grupo y funciones de agregación.

Group By – Fxs agregate

Agrupamiento

Select *lista_proyección*
from *lista_relaciones*
[where *condiciones*]
[group by *lista_de_grupo*]
[having *condición_con_agregación*]

Having:

Restringe el conjunto resultado. Equivalente a where para select, pero solo se puede usar con group by.

Select – Order by

Orden

Select *lista_proyección*
from *lista_relaciones*
[where *condiciones*]
[group by *lista_de_grupo*]
[having *condición_con_agregación*]
[order by *lista_de_ordenacion*]

Limitar la salida:

limit n	al final limita el conjunto resultado a n filas
offset n	descarta las primeras n filas

Select – tratamiento valor NULL

Definición: Representa un valor desconocido, incierto.

Concepto planteado por Codd → uso la letra ω (omega).

Tratamiento en:

En Comparaciones, Condicionales.

Use IS "*en lugar de* ="

Funciones de agregación.

Función coalesce.

- Coalesce(columna1, [columna2, columna3,…], columna_o_cte)

Tratamiento NULL - comparaciones

Sabemos que: cualquier operación con NULL,

Da como resultado Null, sea del tipo que sea aritmetica logica, etc.

Condicionales

Igualdad → use IS — *Where campo is NULL*

*Desi*gualdad → use IS NOT — *Where campo is not NULL*

O *Where not campo is NULL*

Ejemplos:

```
select * from productos2 where precio is null
select * from productos2 where precio is not null
select * from productos2 where not precio is null
```

Tratamiento NULL - Transformaciones

Coalesce

Devuelve el primero de sus argumentos que no es nulo. A menudo se utiliza para sustituir un valor por defecto para los valores nulos cuando se recuperan los datos de pantalla, por ejemplo:

Ejemplos:

```
select
 coalesce(descripcion_corta, descripcion,'sin descripción')
```

Si descripción_corta es no nulo, la función devolvera su valor

Caso contrario

Si descripcion es no nulo, devolvera descripcion

Caso contrario devolvera la constante 'Sin Descripcion'

Generalmente el último argumento es siempre una constante.

Tratamiento NULL – en Fxs Agregación

Count, Sum y Avg → Descartan estos valores:

Ejemplos:

```
SELECT  COUNT(*) AS COUNTASTERISCO,
        COUNT(CODPOS) AS COUNTCODPOS
FROM PROVEEDORES
```

CUIT	NOMBRE	CODPOS	VENTASULTIMOANIO
10	JUAN	3000	30000000
12	PEDRO	3100	10000
15	MARIA	null	30000
18	MARTA	3000	null

Resultado:

COUNTASTERISCO	COUNTCODPOS

Tratamiento NULL – en GROUP BY

- NULL en GROUP BY se toma como un valor mas, es decir otro grupo mas:

Ejemplos:

```
SELECT  CODPOS, COUNT(*) CANTIDAD FROM PROVEEDORES
        GROUP BY CODPOS
```

CUIT	NOMBRE	CODPOS	VENTASULTIMOANIO
10	JUAN	3000	30000000
12	PEDRO	3100	10000
15	MARIA	null	30000
18	MARTA	3000	null

Resultado:

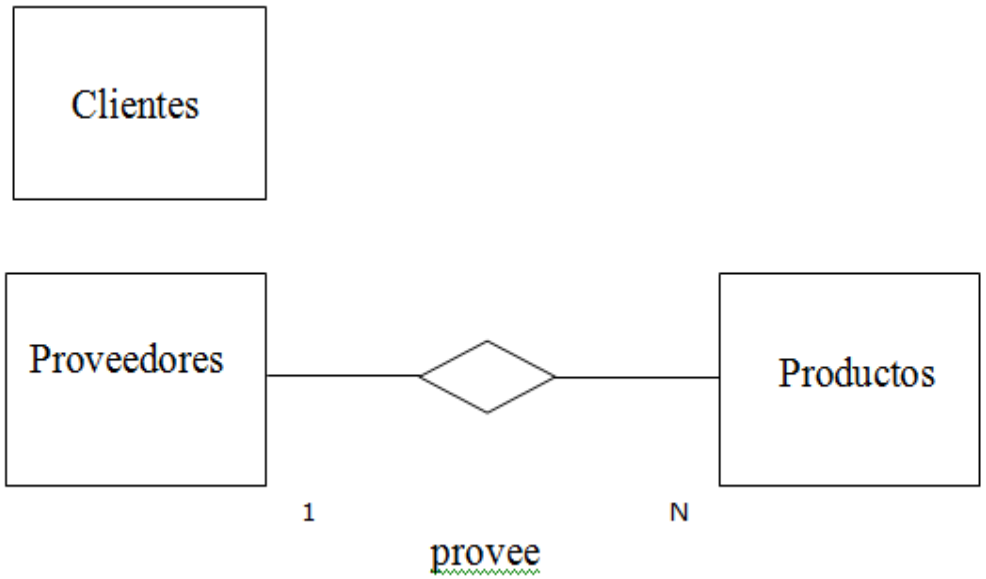
CODPOS	CANTIDAD
NULL	1
3000	2
3100	1

Mas Extensiones

Extensiones Varias

* Reuniones Internas
* Otras Reuniones: externas y completas (outer y full join)
* Operador Exists
* Subselect en proyección, restricción y from (tablas derivadas)

Preparación Material de Trabajo




```
CREATE TABLE CLIENTES
( CUIT VARCHAR(11)
        PRIMARY KEY,
NOMBRE VARCHAR(50),
CODPOSCLI INTEGER);
```

```
CREATE TABLE PROVEEDORES
( CUIT VARCHAR(11) PRIMARY KEY,
  NOMBRE VARCHAR(50),
  CODPOS INTEGER,
  VENTASULTIMOANIO DECIMAL(12,2)
);
```

```
CREATE TABLE PRODUCTOS (
   CODIGO INTEGER,
   NOMBRE VARCHAR(50),
   PRECIO DECIMAL(5,2),
   PROVEEDOR_CUIT VARCHAR(11),
   PRIMARY KEY (CODIGO),
   FOREIGN KEY (PROVEEDOR_CUIT)
REFERENCES PROVEEDORES(CUIT));
```

Select – Reuniones (I)

Una reunión es una relación binaria que requiere un atributo o atributos en comun entre ambas relaciones.

Consiste en restringir un producto cartesiano según la condición de igualdad del elementos o elementos comunes.

Por Ej, podemos reunir Productos con los datos de su proveedor:

Prod Cartesiano → `SELECT * FROM PRODUCTOS, PROVEEDORES`
Restringido por Att común → `WHERE CUIT = PROVEEDOR_CUIT`

Select – Reuniones (II)

Prod Cartesiano → `SELECT * FROM PRODUCTOS, PROVEEDORES`

Restringido por Att común → `WHERE CUIT = PROVEEDOR_CUIT`

codigo	nombre	precio	proveedor_cuit	cuit	nombre	codpos	ventasultimoanio
50	P1	10.00	10	10	JUAN	3000	30000000.00
50	P1	10.00	10	12	PEDRO	3100	10000.00
50	P1	10.00	10	15	MARIA		30000.00
50	P1	10.00	10	18	MARTA	3000	
50	P1	10.00	10	17	CAPPELLETTI Carlos	3000	
60	P2	5.00	10	10	JUAN	3000	30000000.00

Select –Reuniones Tipos

Existen 3 casos de Reuniones:

Internas: → [Inner] Join

Externas:

→ Izquierdas Left [outer] Join

→ Derechas Right [outer] Join

Completas: → Full Join

Select –Reuniones Internas

SQL Provee una estructura sintactica especifica para estos casos

[Inner] Join

Select *lista_proyección*

from *relacion1 r1* **[inner] join** *relacion2 r2*

on *r1.columna = r2.columna [{and|or} ....]*

Ejemplo:

```
SELECT * FROM PRODUCTOS
INNER JOIN PROVEEDORES
 ON CUIT = PROVEEDOR_CUIT
```

La nueva relación incluye solo las filas que cumplen con la condición de combinación o reunión.

Select –Reuniones Internas

Proveedores

Cuit	Nombre	CodigoPostal
11111111	Garcia Juan	3100
33333333	Sacks Pedro	3100
44444444	Re Adrian	3000
55555555	Torres Maria	3000

Clientes

Cuit	Nombre	CodigoPostal
22222222	Huck Luis	3100
33333333	Sacks Pedro	3100
44444444	Re Adrian	3000
88000000	Sanz Armando	3100
88888888	Sain Daniel	3100

```
SELECT * FROM proveedores p
  JOIN clientes c
   ON p.cuit = c.cuit
```

¿Que resultado obtendríamos?

Select –Reuniones Internas

Proveedores

Cuit	Nombre	CodigoPostal
11111111	Garcia Juan	3100
33333333	Sacks Pedro	3100
44444444	Re Adrian	3000
55555555	Torres Maria	3000

Clientes

Cuit	Nombre	CodigoPostal
22222222	Huck Luis	3100
33333333	Sacks Pedro	3100
44444444	Re Adrian	3000
88000000	Sanz Armando	3100
88888888	Sain Daniel	3100

```
SELECT * FROM proveedores p
  JOIN clientes c
   ON p.cuit = c.cuit
```

obtendríamos

Proveedores Join Clientes on Cuit

Cuit	Nombre	CodigoPostal	Cuit	Nombre	CodigoPostal
33333333	Sacks Pedro	3100	33333333	Sacks Pedro	3100
44444444	Re Adrian	3000	44444444	Re Adrian	3000

Select –Reuniones externas left

SQL Provee una estructura sintactica especifica para estos casos

Left [outer] Join

Select *lista_proyección*

from *relacion1 r1* ***left*** **join** *relacion2 r2*

on *r1.columna = r2.columna [{and|or} ....]*

Ejemplo:

```
SELECT * FROM PRODUCTOS
LEFT JOIN PROVEEDORES        ON CUIT = PROVEEDOR_CUIT
```

La nueva relación incluye las filas filas del inner mas las filas de relación 1 que no tienen combinación con relacion 2,
para estos casos los atributos correspondientes a relación 2 quedarán en null

LEFT Join Ejemplo

Proveedores

Cuit	Nombre	CodigoPostal
11111111	Garcia Juan	3100
33333333	Sacks Pedro	3100
44444444	Re Adrian	3000
55555555	Torres Maria	3000

Clientes

Cuit	Nombre	CodigoPostal
22222222	Huck Luis	3100
33333333	Sacks Pedro	3100
44444444	Re Adrian	3000
88000000	Sanz Armando	3100
88888888	Sain Daniel	3100

```
SELECT * FROM proveedores p
  LEFT JOIN clientes c
   ON p.cuit = c.cuit
```

Proveedores Left Join Clientes on Cuit

Cuit	Nombre	CodigoPostal	Cuit	Nombre	CodigoPostal
11111111	Garcia Juan	3100			
33333333	Sacks Pedro	3100	33333333	Sacks Pedro	3100
55555555	Torres Maria	3000			
44444444	Re Adrian	3000	44444444	Re Adrian	3000

Select –Reuniones externas right

SQL Provee una estructura sintactica especifica para estos casos

Right [outer] Join

Select *lista_proyección*
from *relacion1 r1* ***right*** **join** *relacion2 r2*
on *r1.columna = r2.columna [{and|or} ....]*

Ejemplo:

```
SELECT * FROM PRODUCTOS
RIGTH JOIN PROVEEDORES
 ON CUIT = PROVEEDOR_CUIT
```

La nueva relación incluye las filas filas del inner mas las filas de relación 2 que no tienen combinación con relacion 1, para estos casos los atributos correspondientes a relación 1 quedarán en null

RIGHT Join Ejemplo

Proveedores

Cuit	Nombre	CodigoPostal
11111111	Garcia Juan	3100
33333333	Sacks Pedro	3100
44444444	Re Adrian	3000
55555555	Torres Maria	3000

Clientes

Cuit	Nombre	CodigoPostal
22222222	Huck Luis	3100
33333333	Sacks Pedro	3100
44444444	Re Adrian	3000
88000000	Sanz Armando	3100
88888888	Sain Daniel	3100

```
SELECT * FROM proveedores p
  RIGHT JOIN clientes c
   ON p.cuit = c.cuit
```

¿Que resultado obtendríamos?

RIGHT Join Ejemplo

Proveedores

Cuit	Nombre	CodigoPostal
11111111	Garcia Juan	3100
33333333	Sacks Pedro	3100
44444444	Re Adrian	3000
55555555	Torres Maria	3000

Clientes

Cuit	Nombre	CodigoPostal
22222222	Huck Luis	3100
33333333	Sacks Pedro	3100
44444444	Re Adrian	3000
88000000	Sanz Armando	3100
88888888	Sain Daniel	3100

```
SELECT * FROM proveedores p
  RIGHT JOIN clientes c
   ON p.cuit = c.cuit
```

Obtendríamos

Proveedores Right Join Clientes on Cuit

Cuit	Nombre	CodigoPostal	Cuit	Nombre	CodigoPostal
44444444	Re Adrian	3000	44444444	Re Adrian	3000
33333333	Sacks Pedro	3100	33333333	Sacks Pedro	3100
			22222222	Huck Luis	3100
			88888888	Sain Daniel	3100
			88000000	Sanz Armando	3100

Select –Reuniones externas full

SQL Provee una estructura sintactica especifica para estos casos

Full [outer] Join

Select *lista_proyección*
from *relacion1 r1* ***full*** **join** *relacion2 r2*
on *r1.columna = r2.columna [{and|or} ....]*

Ejemplo:

```
SELECT * FROM PRODUCTOS
FULL JOIN PROVEEDORES
 ON CUIT = PROVEEDOR_CUIT
```

La nueva relación incluye las filas filas del inner mas las filas de relación 2 que no tienen combinación con relacion 1, mas las filas de relacion 1 que no tienen combinación con relación 2.

FULL Join Ejemplo

Proveedores

Cuit	Nombre	CodigoPostal
11111111	Garcia Juan	3100
33333333	Sacks Pedro	3100
44444444	Re Adrian	3000
55555555	Torres Maria	3000

Clientes

Cuit	Nombre	CodigoPostal
22222222	Huck Luis	3100
33333333	Sacks Pedro	3100
44444444	Re Adrian	3000
88000000	Sanz Armando	3100
88888888	Sain Daniel	3100

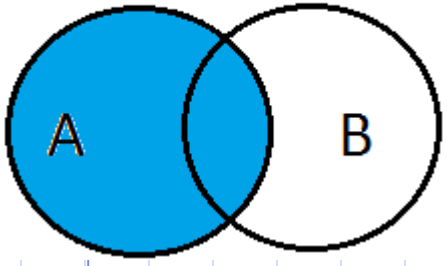
```
SELECT * FROM proveedores p
  FULL JOIN clientes c
   ON p.cuit = c.cuit
```

Proveedores Full Join Clientes on Cuit

Cuit	Nombre	CodigoPostal	Cuit	Nombre	CodigoPostal
11111111	Garcia Juan	3100			
33333333	Sacks Pedro	3100	33333333	Sacks Pedro	3100
55555555	Torres Maria	3000			
44444444	Re Adrian	3000	44444444	Re Adrian	3000
			88000000	Sanz Armando	3100
			22222222	Huck Luis	3100
			88888888	Sain Daniel	3100

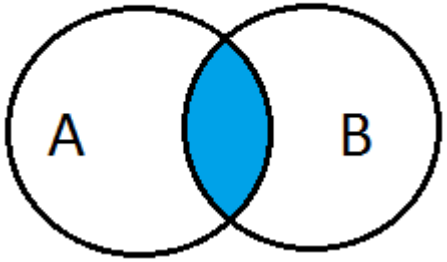
TODOS

LEFT JOIN



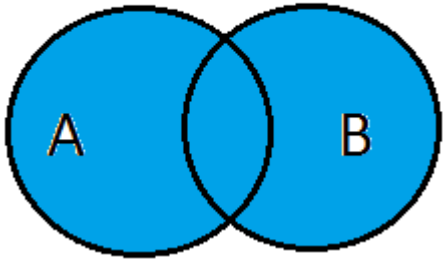
SELECT *lista_proy*
FROM *TablaA A*
LEFT JOIN *TablaB B*
ON *A.key = B.key*

INNER



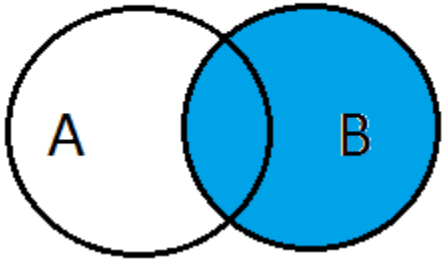
SELECT *lista_proy*
FROM *TablaA A*
JOIN *TablaB B*
ON *A.key = B.key*

FULL JOIN



SELECT *lista_proy*
FROM *TablaA A*
FULL JOIN *TablaB B*
ON *A.key = B.key*

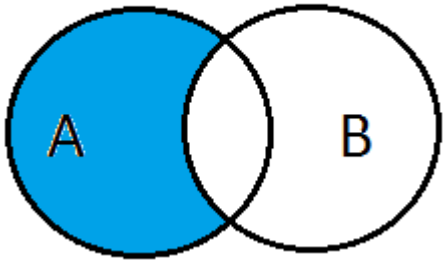
RIGHT JOIN



SELECT *lista_proy*
FROM *TablaA A*
RIGHT JOIN *TablaB B*
ON *A.key = B.key*

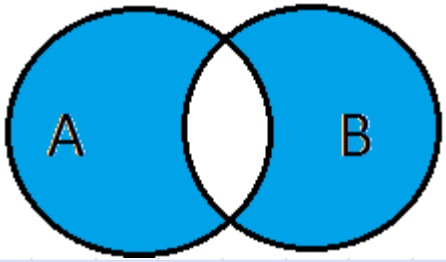
Tratando con JOINs

LEFT minus INNER



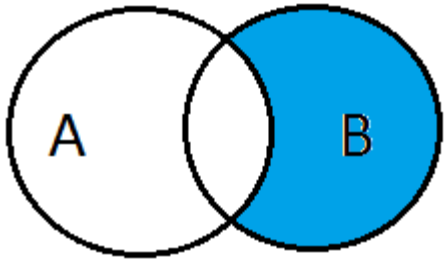
SELECT *lista_proy*
FROM *TablaA A*
LEFT JOIN *TablaB B*
ON *A.key = B.key*
?????????

FULL minus INNER



SELECT *lista_proy*
FROM *TablaA A*
FULL JOIN *TablaB B* ***ON*** *A.key = B.key*
AND A.key IS NULL OR B.key IS NULL
?????????

RIGHT minus INNER

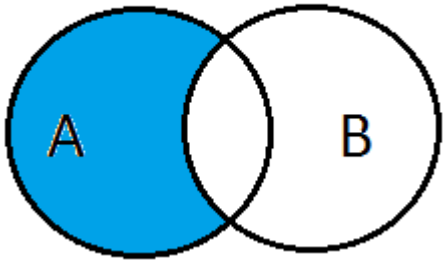


SELECT *lista_proy*
FROM *TablaA A*
RIGHT JOIN *TablaB B*
ON *A.key = B.key*
????????

¿Que agregaría a las sentencias para obtener el resultado ilustrado?

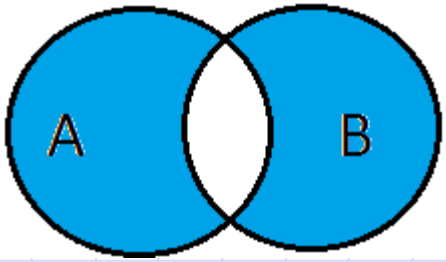
Alternativas mas directas

LEFT minus INNER



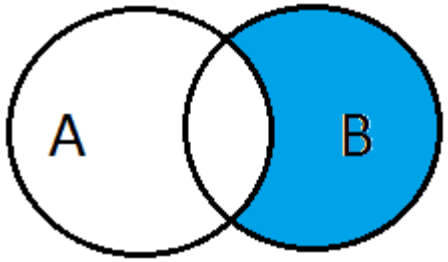
?????????

FULL minus INNER



?????????

RIGHT minus INNER



?????????

¿Que operador o combinación representa cada figura?

Select – Sub select

- Podemos usar sub select en 3 casos:
  - En Proyección
  - En Restricción
  - En lista de relaciones (Tablas Derivadas)

Select – Sub select en Proyección

- En Proyección – *Solo restringido a salidas de grado 1 y cardinalidad 1.*

```
SELECT CODIGO, NOMBRE, PROVEEDOR_CUIT,
  ( SELECT NOMBRE
      FROM PROVEEDORES WHERE CUIT =   PROVEEDOR_CUIT     )
   FROM PRODUCTOS
```

Para entender la idea lo podemos interpretar,

El motor realiza el select de productos (codigo, nombre, prov_cuit) y luego por cada fila de este busca el nombre del proveedor.

Select – Sub select en Restricción

- En Restricción
  - *Usar = Solo si esta garantizado que la salida es de grado 1 y cardinalidad 1.*
  - *Caso contrario usar in*

```
SELECT * FROM PROVEEDORES
 WHERE CUIT IN
   (  SELECT DISTINCT PROVEEDOR_CUIT FROM PRODUCTOS  )
```

Para entender la idea lo podemos interpretar,

El motor realiza el select de productos y luego con esta lista de cuit realiza la selección de proveedores usando el operador de conjunto IN.

Select – Sub select en From

En From (en algunos textos figura como tabla anidada o tabla derivada)

```
SELECT *
        FROM PROVEEDORES,
       (SELECT CODPOS, SUM(VENTASULTIMOANIO) AS VENTAS
                  FROM PROVEEDORES
                 GROUP BY CODPOS ) TOTALVENTAS
WHERE TOTALVENTAS.CODPOS = PROVEEDORES.CODPOS
```

Para entender la idea lo podemos interpretar,

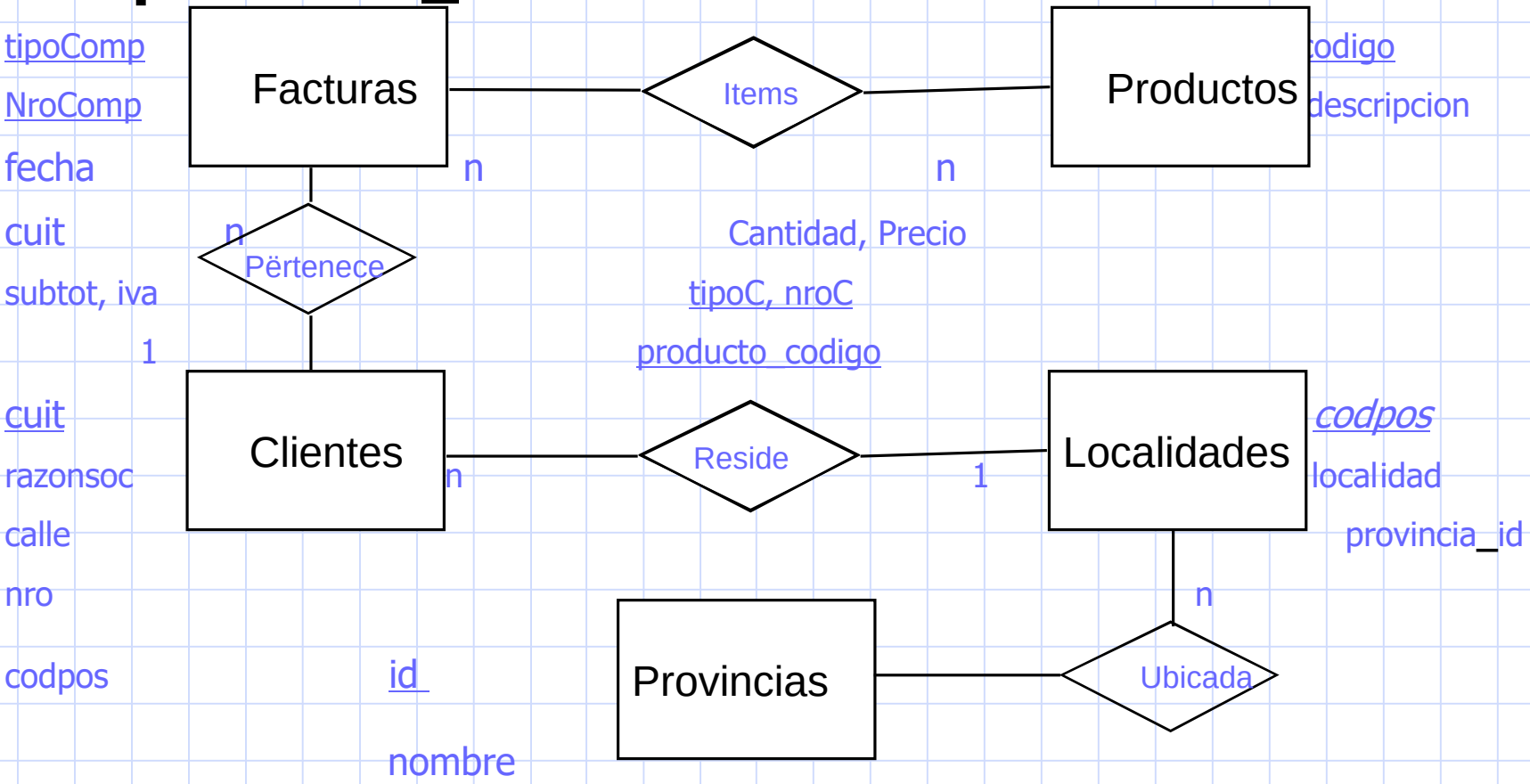
El motor realiza el select de proveedores creando la relación intermedia totalventas y luego hace un join con proveedores.

***Nota:** La tabla anidada debe renombrarse obligatoriamente.*

Caso de Estudio - Resuelva usando Subselect

1. Obtenga un listado completo de clientes con la localidad.
2. Obtenta un listado de facturas con la cantidad de Items.
3. Obtenga un reporte de clientes reunido con la cantidad de IVA tributado en el año 2015 usando subselect en FROM.

Comprobantes_Facturación



Select – UNION – INTERSECT - EXCEPT

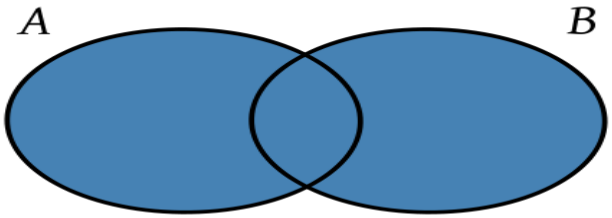
- Operadores Relacionales Binarios de relaciones de conjuntos homogeneos.

  Union: une 2 select de conjuntos homogéneos.

  Intersect: intersecta 2 select de conjuntos homogéneos.

  Except: Diferencia de 2 select de conjuntos homogéneos.

Select – UNION (I)



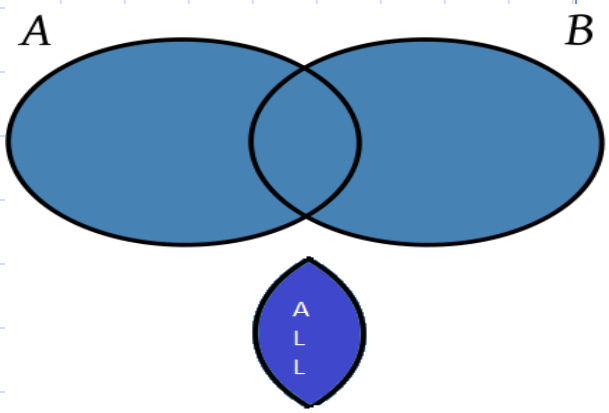
Genera un esquema de relacion que posee el mismo conjunto de atributos de R1 y R2, e incluye las tuplas que pertenecen a R1 o a R2 o a ambas.

```
Select * from clientes
 union
select * from deudores;
```

Restriccion: ambas relaciones deben ser compatibles

Dos relaciones, R(A1, A2, , An) y S(B1, B2, , Bn), son compatibles si tienen el mismo tipo de tuplas. Es decir, si ambas tienen grado n y si dom(Ai)=dom(Bi) para $1 \leq i \leq n$.

Select – UNION (II)



Existen 2 tipos de uniones

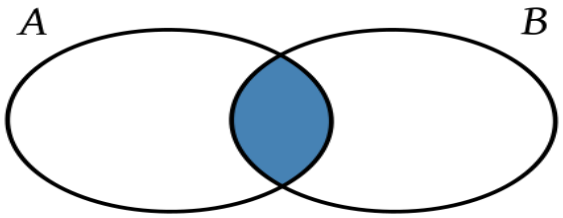
Puras: eliminan filas repetidas

```
Select * from clientes
     union select * from deudores;
```

Completas: repreentan tambien las filas repetidas

```
Select * from clientes
     Union all select * from deudores;
```

Select – INTERSECCIÓN



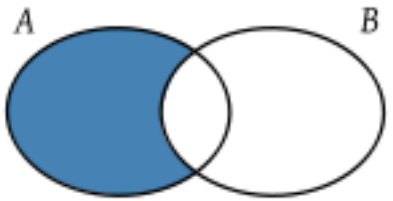
Genera un esquema de relacin que posee el mismo conjunto de atributos de R1 y R2, e incluye las tuplas que pertenecen a R1 y tambien a R2.

```
Select * from clientes
 intersect
select * from deudores;
```

Restriccion: ambas relaciones deben ser compatibles

Dos relaciones, R(A1, A2, , An) y S(B1, B2, , Bn), son compatibles si tienen el mismo tipo de tuplas. Es decir, si ambas tienen grado n y si dom(Ai)=dom(Bi) para $1 \leq i \leq n$.

Select – DIFERENCIA



Genera un esquema de relacin que posee el mismo conjunto de atributos de R1 y R2, e incluye las tuplas que pertenecen a R1 y no pertenecen a R2.

```
Select * from clientes
 Except
select * from deudores;
```

Restriccion: ambas relaciones deben ser compatibles

Dos relaciones, R(A1, A2, , An) y S(B1, B2, , Bn), son compatibles si tienen el mismo tipo de tuplas. Es decir, si ambas tienen grado n y si dom(Ai)=dom(Bi) para $1 \leq i \leq n$.

Caso de Estudio

4. Obtenga los alumnos recursantes entre 2015 y 2016.
5. Obtenga los alumnos que cursaron en el 2016 y no en el 2015.
6. Obtenta la nomina total de alumnos que cursaron en 2015 y 2016 orden:nombre.
7. Obtenga el año que cursaron mas alumnos.
8. Obtenga la cantidad de alumnos que cursaron por año.
9. Obtenga la cantidad de alumnos no recursantes.
10. Obtenga la cantidad un reporte como el siguiente:

```
      Grupo                    Cantidad
-----------------------------------
Inscriptos 2015                  xxxx
Inscriptos 2016                  xxxx
Recursantes                      xxxx
No recursantes                   xxxx
Varones                          xxxx
```

Cursado de Base de Datos

Legajo
Nombre
sexo

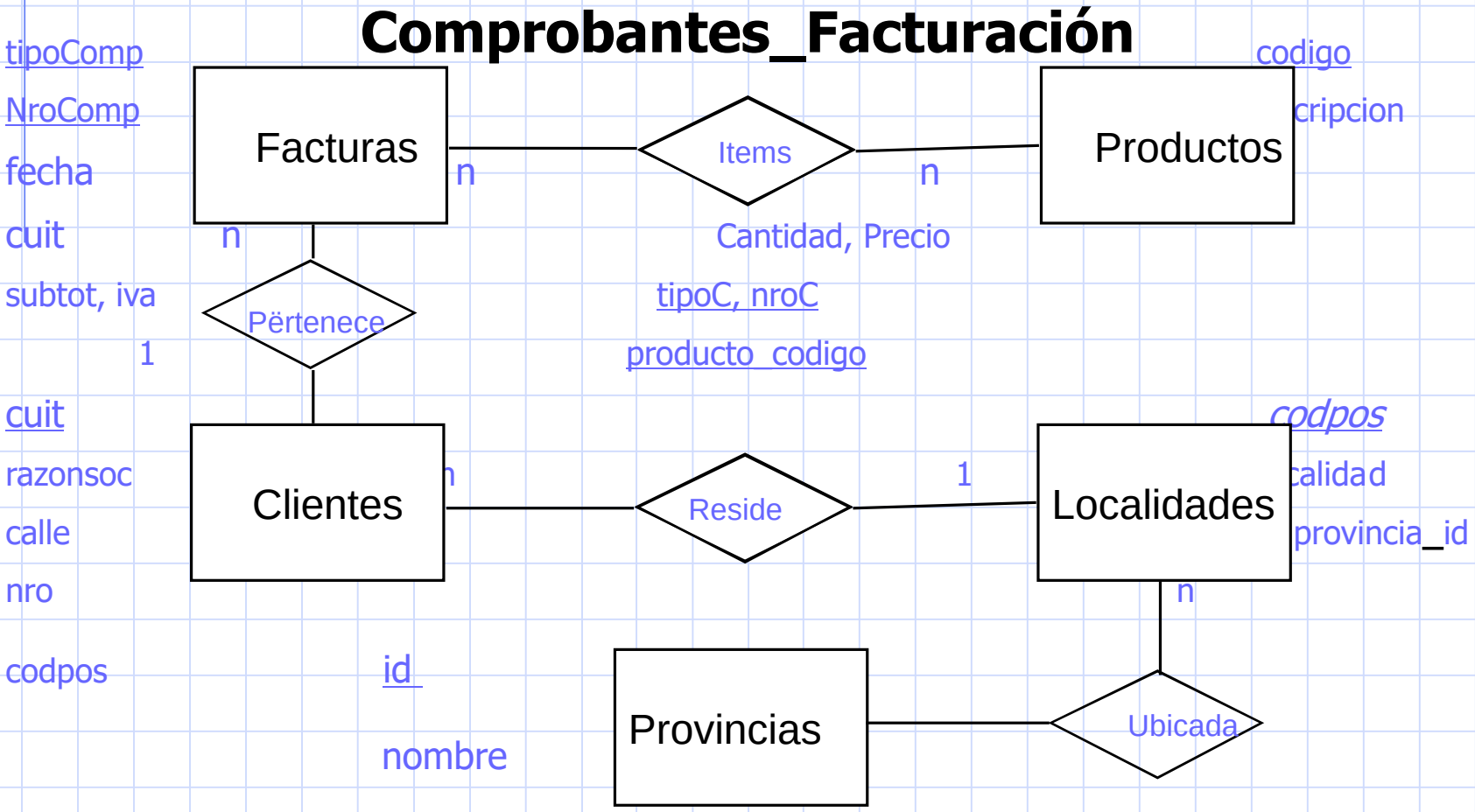
Bd2015

Bd2016

Legajo
Nombre
Sexo

Caso de Estudio

11. Si se tiene una nueva rel 'ProductosComptencia' de estructura simil a productos.
    a) Liste los productos que provee la competencia y nosostros no.
    b) Liste los productos que vende la competencia y nosotros.
    c) Liste la union de ambas relaciones, con y sin filas repetidas.
    d) que sucede si nosotros tenemos Producto: 05 'Rem t1' y la comp 05 'REM T1'



Select – Desigualdades {ANY / ALL}

Al igual que podiamos usar in como operador de conjunto de igualdades podemos unsar ANY (SOME) o ALL como modificadores, los cuales nos permiten tambien comparar con conjuntos de registros de igualdades o desigualdades:

<expresion> {= / < / > /!= /...} {ANY / ALL} Subconsulta

```
-- ANY o SOME
Lista las filas si al menos una fila de la subconsulta
   cumple con la restricción de igualdad o desigualdad


-- ALL
Lista las filas si todas las filas de la subconsulta cumple
   con la restricción de igualdad o desigualdad
```

Select – Desigualdades ANY / ALL

```
-- ANY o SOME
SELECT * FROM VENTAS
WHERE IMPORTE > ANY
(SELECT VENTA_MINIMA FROM SUCURSAL)
```

Lista las filas de ventas si al menos una sucursal posee venta minima menor

```
-- ALL
SELECT * FROM VENTAS
WHERE IMPORTE > ALL
(SELECT VENTA_MINIMA FROM SUCURSAL)
```

Lista las filas de ventas cuyo importe son superiores a todas las sucursales (considerando venta mínima).

Select – Exists

El operador exists detecta si el conjunto que le sigue no esta vacio.

```
Select * from clientes c Where
 Exists
( select * from proveedores p on
p.cuit = c.cuit)
)
```

Lista los clientes que tambien son proveedores.

`Not Exists` detecta si el conjunto esta vacio.

Caso de Estudio

4. Obtenga los productos cuyo precio son inferiores a al menos un competidor.
5. Obtenga los productos cuyo precio son mas baratos que toda la competencia.
6. Obtenga los productos sin competencia.

Comparativa Precios Competencia

Productos	ProdCompetencia
codigo	codigo
Nombre	competidor
Precio	Nombre
	preciocomp

Select – Sinopsys PostgreSQL

```
SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
    * | expression [ [ AS ] output_name ] [, ...]
    [ FROM from_item [, ...] ]
    [ WHERE condition ]
    [ GROUP BY expression [, ...] ]
    [ HAVING condition [, ...] ]
    [ {UNION | INTERSECT | EXCEPT} [ ALL | DISTINCT ]
    select ]
    [ ORDER BY expression [ ASC | DESC ]
                [ NULLS { FIRST | LAST } ] [, ...] ]
    [ LIMIT { count | ALL } ]
    [ OFFSET start [ ROW | ROWS ] ]
```

http://www.postgresql.org/docs/9.4/static/sql-select.html

Resumen

Revisión del Modelado y DER

Revisión del Modelo Relacional

Normalización

Revisión del Algebra Relacional

DMLs de consultas

- DMLs de actualización

DML de Actualización de Información

Las sentencias de actualización son:

Insert: Agrega filas.

Delete: Elimina filas.

Update: Modifica filas.

Los objetos destino de una actualización "directa" son:

Tablas.

Vistas (Con características determinadas).

DML – Insert (I)

Insert: Agrega una fila a una tabla o vista.

Insert into *relación* *[(lista_de_columnas)]*
values *(lista_valores);*
[returning *(lista_de_columnas)];*

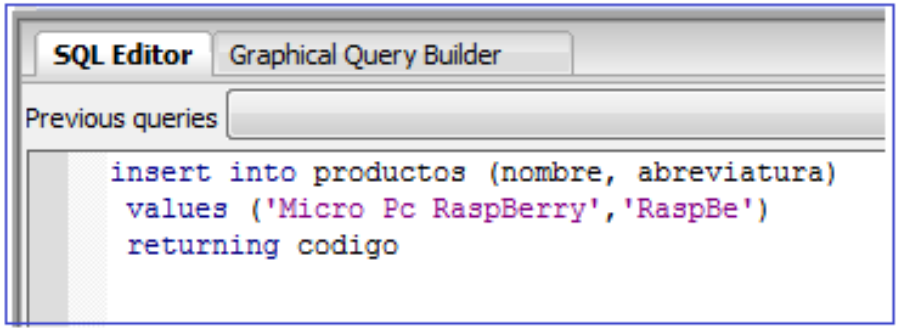
Que sucede con las columnas no incluidas?

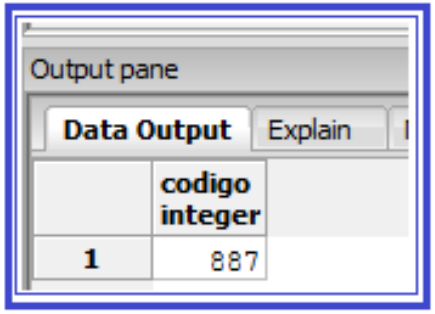
Nota: Luego del análisis sintáctico se evalúan las ***constraint*** de la relación y por ultimo la seguridad.

DML – Insert (II)

Ejemplo: Agregando una fila a productos.

Insert into *productos (nombre, abreviatura)*
values (‘Micro Pc RaspBerry’,’RaspBe’);
returning *codigo;*





Returning: el modificador opcional ***“Returning codigo”*** hace que se retorne a modo de ***select*** el código asignado por el *serial*.

DML – Insert from Select (III)

Insert … select: Inserta las filas proveniente de un select compatible.

Insert into *relación [values (lista_columnas)]*
select …..
[**returning** *lista de atributos*];

Ej: con

```
CREATE TABLE PROVEEDORES_BACK
(   CUIT        CHAR(11) NOT NULL,
    NOMBRE  VARCHAR(50),
    CODPOS          INTEGER,
    FECHA_BACK TIMESTAMP,
    USR_BACK     VARCHAR(32),
   PRIMARY KEY (CUIT,, FECHA_BACK) );
```

DML – Insert from Select (IV)

Ejemplo: Como copiarían las filas de PROVEEDORES a PROVEEDORES_BACK

DML – Insert from Select (IV)

Ejemplo: Como copiarían las filas de PROVEEDORES a PROVEEDORES_BACK

Insert into *proveedores_back*
(cuit, nombre, fecha_back,usr_back)
Select *cuit, nombre, current_timestamp, current_user*
from *proveedores;*

DML – Delete (I)

Delete: Elimina la/s fila/s de una tabla o vista.

Delete from *relación*
[where *condición]*
[returning *(lista_de_columnas)];*

Elimina todas las filas de relación que cumplen con "condición".

Condición: Idem sintáctica y semánticamente a la vista en "select".

Nota: Si no se incluye condición todas las filas de la relación serán eliminadas.

DML – Delete (II)

Ejemplo : Como Eliminaría las filas de Proveedores_productos con fecha fin no nula y menor a la fecha actual.

DML – Delete (II)

Ejemplo : Como Eliminaría las filas de Proveedores_productos con fecha fin no nula y menor a la fecha actual.

Delete From *proveedores_productos*
Where *fecha_fin < current_date;*

¿y los nulos?

DML – Truncate

Truncate: DDL que vacia todas las filas.

Truncate table *nombre_tabla;*

Nota: Al ser una DDL es Autocommit (en todas las implementaciones).

DML – Update (I)

Update: Actualiza fila/s de una tabla o vista.

Update *relación*

set *columna = valor [, columna = valor]*

[where *condición];*

[returning *(lista_de_columnas)];*

Actualiza todas las filas de relación que cumplen con "condición". Las columnas afectadas son las que figuran en SET.

Condición: Idem sintáctica y semánticamente a la vista en "select".

Nota: Si no se incluye condición todas las filas de la relación serán actualizadas.

DML – Update (II)

Ejemplo: Como cambiaría el nombre del proveedor 20-17044666-7 por “Nuevo Nombre”

DML – Update (III)

Ejemplo: Como cambiaría el nombre del proveedor 20-17044666-7 por "Nuevo Nombre"

Update *proveedores*

set *nombre = 'Nuevo Nombre'*

where *cuit = '20-17044666-7'*

DML – Update Caso Especial (ii)

La sentencia **Update** admite el uso de otras tabla para determinar las filas a actualizar. *Idem para Delete.*

Agrega FROM y Update [Only].

Update [Only] *relación*
set *columna =valor [, columna = valor]*
[from *lista_tablas]*
[where *condición];*

Ej: **update only** *productos d set precio = precio * 1.12*
from *proveedores_productos pp, proveedores v*
where *pp.cuit = v.cuit and pp.producto_codigo = d.codigo*
and v.codpos = 3000

DML – Update Caso Especial (i)

La sentencia **Update** admite **subselect** en el valor del set y en restricción.

Update *relación*

set *columna = (SubSelect1) [, columna = valor]*

[where *columna = (SubSelect2) ....];*

Idem para Delete pero solo con Restricción.

Que características debe tener subselect1?

Que características debe tener subselect2?

Ejercicios Propuestos

1) Realice los siguientes agregados:
   - El producto "Teclado Genérico 101 Esp" abreviando el nombre como crea conveniente.
   - El proveedor "Romero Hnos" de Paraná con Cuit a determinar luego.
   - Una relación para el proveedor "20-17044666-7" y el codigo de producto 13, iniciandose la relación hoy.

2) Realice el sig Update utilizando solo subselect sin from: **update only** *productos d set precio = precio * 1.12*

   from *proveedores_productos pp, proveedores v*

   where *pp.cuit = v.cuit*

   and pp.producto_codigo = d.codigo

   and v.codpos = 3000

Resumen

Revisión del Modelo Relacional

Revisión de DER y Normalización

Revisión del Algebra Relacional

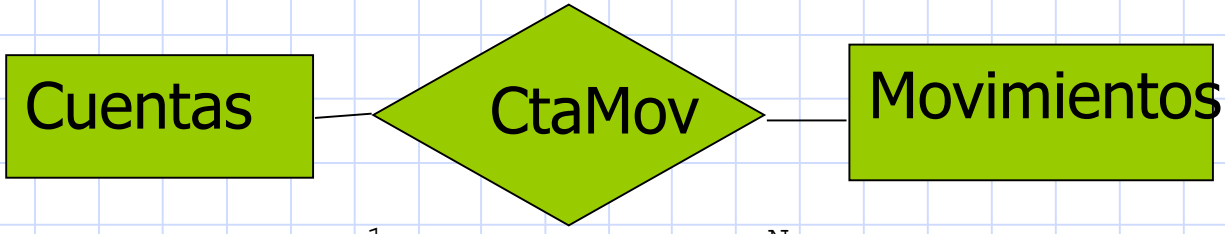
DMLs de consultas

DMLs de actualización

- Transacciones ???

Problema

Una transferencia bancaria de la cuanta A a la B:



NroCta PK

Nombre

Saldo

NroMovimiento PK

NroCta FK sobre Cuentas.Nrocta

Fecha

Debito_Credito

Importe

Involucra las siguientes acciones

Ingresar un movimiento de debito a la cuenta A.

Ingresar un movimiento de crédito a la cuenta B..

Modificar el saldo de A.

Modificar el saldo de B.

Transacción N

Problema corte en este punto

Solución *<<Transacción de BD>>*

Como primera aproximación podríamos definir:

transacción en el ámbito de una base de datos como una unidad de trabajo que se desarrolla en un contexto atómico.

La función básica de las transacciones es:

garantizar que las operaciones que realizamos contra nuestra base de datos no ponen en peligro la *coherencia y consistencia* de la información que almacenamos en ella.

Definición Formal

Una transacción es un conjunto de operaciones consecutivas de la misma sesión que se aceptan (*Commit*) o rechazan (*Rollback*) como una sola.

Implementación de un Motor Transaccional

Un sistema administrador de base de datos transaccional debe cumplir con las propiedades ACID.

Propiedades de las Transacciones

- Atomicity (*Atomicidad*): Se ejecuta en su totalidad o no se ejecuta nada.

- Consistency (*Consistencia*): Las transacciones siempre llevan a la base de un estado consistente a otro también consistente.

- Isolation (*aIslamiento*): Cada transacción se ejecuta en la base de datos como si estuviera sola.

- Durability (*Durabilidad* o *persistencia*): Una vez que una transacción se acepta los cambios deben quedar fijos en la base de datos aún en caídas posteriores.

Sentencias asociadas

Commit: Confirma transacciones.

Rollback: Deshace transacciones.

Begin: Inicia una Transacción.

Comienzo y fin de transacciones

Una transacción comienza cuando se produce alguno de los siguientes enventos:

- **Inicio Sesión**
- **Begin**
- **Commit**
- **Rollback**

Una transacción finaliza cuando se produce alguno de los siguientes enventos

- **Commit**
- **Rollback**
- **End** Cierre de una transacción

Los clientes oficiales de sistemas SQLSERVER y POSTGRES trabajan con AUTOCOMIT

Algunos Ej Postgres (sin Atocommit)

```
-- Transferencia de la cuenta A a la cuenta B
-- Inicio de la transacción
BEGIN;

-- Insertando el movimiento de la cuenta A.
INSERT INTO movs (nombre_cta, fecha, importe,tipo) VALUES ('A',current_date,120,'D');

-- Insertando el movimiento de la cuenta B.
INSERT INTO movs (nombre_cta, fecha, importe,tipo) VALUES ('B',current_date,120,'C');

SAVEPOINT cargos_hechos;

-- Abonando la cuenta B.
UPDATE cuentas SET saldo = saldo - 120 WHERE nombre_cta = 'H';

-- Error: La cuenta a modificar era A, no la H.
ROLLBACK TO cargos_hechos;

-- Actualizando la cuenta correcta.
UPDATE cuentas SET saldo = saldo - 120 WHERE nombre_cta = 'A';
UPDATE cuentas SET saldo = saldo + 120 WHERE nombre_cta = 'B';

-- Escribimos los cambios
COMMIT;
```

Algunos Ej Postgres (sin Atocommit)(II)

```
-- Transferencia de la cuenta A a la cuenta B

-- Inicio de la transacción
BEGIN;

-- Insertando el movimiento de la cuenta A.
INSERT INTO movs (nombre_cta, fecha, importe,tipo) VALUES ('A',current_date,120,'D');

-- Insertando el movimiento de la cuenta B.
INSERT INTO movs (nombre_cta, fecha, importe,tipo) VALUES ('B',current_date,120,'C');

SAVEPOINT cargos_hechos;

-- Abonando la cuenta B.
UPDATE cuentas SET saldo = saldo - 120 WHERE nombre_cta = 'H';

-- Error: La cuenta a modificar era A, no la H.
ROLLBACK TO cargos_hechos;

-- Actualizando la cuenta correcta.
UPDATE cuentas SET saldo = saldo - 120 WHERE nombre_cta = 'A';
UPDATE cuentas SET saldo = saldo + 120 WHERE nombre_cta = 'B';

-- Otra forma de confirmar con END similar a commit
END;
```

Algunos Ej Postgres (sin Atocommit)(III)

```
-- Transferencia de la cuenta A a la cuenta B

-- Inicio de la transacción
BEGIN;

-- Insertando el movimiento de la cuenta A.
INSERT INTO movs (nombre_cta, fecha, importe,tipo) VALUES ('A',current_date,120,'D');

-- Insertando el movimiento de la cuenta B.
INSERT INTO movs (nombre_cta, fecha, importe,tipo) VALUES ('B',current_date,120,'C');

SAVEPOINT cargos_hechos;

-- Abonando la cuenta B.
UPDATE cuentas SET saldo = saldo - 120 WHERE nombre_cta = 'H';

-- Error: La cuenta a modificar era A, no la H.
ROLLBACK TO cargos_hechos;

-- Actualizando la cuenta correcta.
UPDATE cuentas SET saldo = saldo - 120 WHERE nombre_cta = 'A';
UPDATE cuentas SET saldo = saldo + 120 WHERE nombre_cta = 'B';

-- Se cancela todo por que hay un error en las fechas
ROLLBACK;
```

Algunos Ej SqlServer (sin Atocommit)

```
-- Transferencia de la cuenta A a la cuenta B
-- Inicio de la transacción
BEGIN TRANSACTION;

-- Insertando el movimiento de la cuenta A.
INSERT INTO movs (nombre_cta, fecha, importe,tipo) VALUES ('A',current_date,120,'D');

-- Insertando el movimiento de la cuenta B.
INSERT INTO movs (nombre_cta, fecha, importe,tipo) VALUES ('B',current_date,120,'C');

-- Actualizando la cuenta correcta.
UPDATE cuentas SET saldo = saldo - 120 WHERE nombre_cta = 'A';
UPDATE cuentas SET saldo = saldo + 120 WHERE nombre_cta = 'B';

-- Escribimos los cambios
COMMIT;
```

Algunos Ej SqlServer (sin Atocommit) (II)

```
-- Transferencia de la cuenta A a la cuenta B
-- Inicio de la transacción
BEGIN TRANSACTION;

-- Insertando el movimiento de la cuenta A.
INSERT INTO movs (nombre_cta, fecha, importe,tipo) VALUES ('A',current_date,120,'D');

-- Insertando el movimiento de la cuenta B.
INSERT INTO movs (nombre_cta, fecha, importe,tipo) VALUES ('B',current_date,120,'C');

-- Actualizando la cuenta correcta.
UPDATE cuentas SET saldo = saldo - 120 WHERE nombre_cta = 'A';
UPDATE cuentas SET saldo = saldo + 120 WHERE nombre_cta = 'B';

-- Se cancela todo cambio
ROLLBACK;
```

Niveles de Aislamiento I

- Serializable: Este es el nivel de aislamiento más alto. Especifica que todas las transacciones ocurran de modo aislado, o dicho de otro modo, como si todas las transacciones se ejecutaran de modo serie (una tras otra). La sensación de ejecución simultánea de dos o más transacciones que perciben los usuarios sería una ilusión producida por el SGBD.

- read commited (lectura de lo aceptado): si estamos en una transacción con este nivel de aislación, podremos ver los cambios de los demás *cuando ellos terminen su transacción con commit* (nosotros tendremos que repetir la consulta).

- snapshot (lectura repetible): se garantiza que el usuario que esté en una transacción de este tipo verá siempre los mismos datos, aunque otros usuarios hagan cambios y los acepten. No veremos los cambios hasta que cerremos nuestra transacción y comencemos una nueva. Ideal para reportes.

Niveles de Aislamiento II

- **Dirty read (lectura sucia):** Hay un nivel más de aislamiento que es común en los sistemas de bases de datos locales. En este nivel se pueden ver los cambios inmediatamente. Esto trae problemas podemos leer datos que no han sido confirmados.

  Este nivel no cumple con las propiedades ACID.
- No implementado en firebird ni postgreSQL

Bloqueos

Los bloqueos impiden que se haga una segunda modificación sobre los mismos datos mientras todavía no se han aceptado o rechazado los primeros.

Hay 2 Tipos:

Pesimista: Asumen que es muy probable que en un rdbms 2 transacciones accedan a los mismos datos. Entonces bloquean al inicio. (Cuando se edita un registro).

Optimista: Asumen que es muy poco probable que un rdbms 2 transacciones accedan a los mismos datos.
Entonces bloquean cuando se realizado una modificación.

Fuentes

Libro: Introducción a los SISTEMAS DE BASES DE DATOS
Autor: C.J. Date
Editorial: Addison Wesley

Libro: Fundamentos de Bases de Datos
Autor: Silberschatz / Korth / Sudarshan
Editorial: Mc Graw Hill

Documentación Oficial PostgreSQL
http://www.postgresql.org/docs/9.1/static/