Informe laboratorio 3

Grupo 51 [ Paulo Sánchez - Samuel Noble ]

# Actividad 1

## Ejercicio 1)

La consulta para determinar la cantidad de tuplas es:

select count(\*) from empresas;

Y determinó que hay **9210 tuplas**.

## Ejercicio 2)

id\_emp → nom\_emp, desc\_emp, fecha\_fund\_emp

id\_ciudad\_oper\_emp → ciudad\_oper\_emp, id\_pais\_oper\_emp

id\_pais\_oper\_emp → pais\_oper\_emp

id\_resp\_emp → nom\_resp\_emp, ap\_resp\_emp, fnac\_resp\_emp, id\_ciu\_nac\_resp\_emp

id\_ciu\_nac\_resp\_emp → ciu\_nac\_resp\_emp, id\_pais\_nac\_resp\_emp

id\_pais\_nac\_resp\_emp → pais\_nac\_resp\_emp

id\_prod → nom\_prod, desc\_prod

## Ejercicio 3)

Para identificar las violaciones de las dependencias diseñamos las siguientes consultas. En estas consultas, si el resultado contiene alguna tupla, entonces hay una violación de una dependencia funcional.

Verificación de la dependencia funcional: id\_emp → nom\_emp, desc\_emp, fecha\_fund\_emp.

Hay violación de la dependencia funcional: **id\_emp → fecha\_fund\_emp**.

select id\_emp, count(distinct nom\_emp),

from empresas  
group by id\_emp  
having count(distinct nom\_emp) > 1;

select id\_emp, count(distinct desc\_emp)  
from empresas  
group by id\_emp  
having count(distinct desc\_emp) > 1;

select id\_emp, count(distinct fecha\_fund\_emp)  
from empresas  
group by id\_emp  
having count(distinct fecha\_fund\_emp) > 1;

Verificación de la dependencia funcional: id\_ciudad\_oper\_emp → ciudad\_oper\_emp, id\_pais\_oper\_emp.

Hay violación de la dependencia funcional: **id\_ciudad\_oper\_emp → id\_pais\_open\_emp**.

select id\_ciudad\_oper\_emp, count(distinct ciudad\_oper\_emp)  
from empresas  
group by id\_ciudad\_oper\_emp  
having count(distinct ciudad\_oper\_emp) > 1;

select id\_ciudad\_oper\_emp, count(distinct id\_pais\_oper\_emp)  
from empresas  
group by id\_ciudad\_oper\_emp  
having count(distinct id\_pais\_oper\_emp) > 1;

Verificación de la dependencia funcional: id\_pais\_oper\_emp → pais\_oper\_emp.

No hay violación de dependencia funcional.

select id\_pais\_oper\_emp, count(distinct pais\_oper\_emp)  
from empresas  
group by id\_pais\_oper\_emp  
having count(distinct pais\_oper\_emp) > 1;

Verificacion de la dependencia funcional: id\_resp\_emp → nom\_resp\_emp, ap\_resp\_emp, fnac\_resp\_emp, id\_ciu\_nac\_resp\_emp.

Hay violación de la dependencia funcional: **id\_resp\_emp → nom\_resp\_emp**.

select id\_resp\_emp, count(distinct nom\_resp\_emp)  
from empresas  
group by id\_resp\_emp  
having count(distinct nom\_resp\_emp) > 1;

select id\_resp\_emp, count(distinct ap\_resp\_emp)  
from empresas  
group by id\_resp\_emp  
having count(distinct ap\_resp\_emp) > 1;

select id\_resp\_emp, count(distinct fnac\_resp\_emp)  
from empresas  
group by id\_resp\_emp  
having count(distinct fnac\_resp\_emp) > 1;

select id\_resp\_emp, count(distinct id\_ciu\_nac\_resp\_emp)  
from empresas  
group by id\_resp\_emp  
having count(distinct id\_ciu\_nac\_resp\_emp) > 1;

Verificacion de la dependencia funcional: id\_ciu\_nac\_resp\_emp → ciu\_nac\_resp\_emp, id\_pais\_nac\_resp\_emp.

Hay violación de la dependencia funcional: **id\_ciu\_nac\_resp\_emp → id\_pais\_nac\_resp\_emp**.

select id\_ciu\_nac\_resp\_emp, count(distinct ciu\_nac\_resp\_emp)  
from empresas  
group by id\_ciu\_nac\_resp\_emp  
having count(distinct ciu\_nac\_resp\_emp) > 1;

select id\_ciu\_nac\_resp\_emp, count(distinct id\_pais\_nac\_resp\_emp)  
from empresas  
group by id\_ciu\_nac\_resp\_emp  
having count(distinct id\_pais\_nac\_resp\_emp) > 1;

Verificacion de la dependencia funcional: id\_pais\_nac\_resp\_emp → pais\_nac\_resp\_emp.

No hay violación de dependencia funcional.

select id\_pais\_nac\_resp\_emp, count(distinct pais\_nac\_resp\_emp)  
from empresas  
group by id\_pais\_nac\_resp\_emp  
having count(distinct pais\_nac\_resp\_emp) > 1;

Verificacion de la dependencia funcional: id\_prod → nom\_prod, desc\_prod.  
No hay violación de dependencia funcional.

select id\_prod, count(distinct nom\_prod)  
from empresas  
group by id\_prod  
having count(distinct nom\_prod) > 1;

select id\_prod, count(distinct desc\_prod)  
from empresas  
group by id\_prod  
having count(distinct desc\_prod) > 1;

## Ejercicio 4)

Sean los conjuntos de atributos:

* A = {rubro\_emp}  
  *Los que no están en ninguna df*
* B = { id\_emp, id\_ciudad\_oper\_emp, id\_resp\_emp, id\_prod}   
  *Los que no están a la derecha de ninguna df*
* C = { nom\_emp, desc\_emp, ciudad\_oper\_emp, pais\_oper\_emp, fecha\_fund\_emp, nom\_resp\_emp, ap\_resp\_emp, fnac\_resp\_emp, ciu\_nac\_resp\_emp, pais\_nac\_resp\_emp, nom\_prod, desc\_prod }  
  *Los que no están a la izq de ninguna df*

A unión B es el conjunto de atributos que están obligado a estar en todas las claves, si resulta que la clausura de este conjunto es superclave, entonces es clave, debido a que no puedo quitar atributos, ya que estos son obligatorios.

Como (A unión B)+ = (rubro\_emp, id\_emp, id\_ciudad\_oper\_emp, id\_resp\_emp, id\_prod )+ y esta clausura genera todos los atributos del esquema empresas entonces (rubro\_emp, id\_emp, id\_ciudad\_oper\_emp, id\_resp\_emp, id\_prod) es clave.

## Ejercicio 5)

Para detectar violaciones que son generadas por nulos en las claves usamos la siguiente consulta:

select rubro\_emp, id\_emp, id\_ciudad\_oper\_emp, id\_resp\_emp, id\_prod

from empresas

where rubro\_emp is null or id\_emp is null

or id\_ciudad\_oper\_emp is null or id\_resp\_emp is null

or id\_prod is null or rubro\_emp = '' or id\_emp = ''

or id\_ciudad\_oper\_emp = '' or id\_resp\_emp = ''

or id\_prod = '';

En este caso el resultado nos va a devolver todas las tuplas en las que por lo menos uno de los atributos de la clave en nulo. En total son **8743 tuplas**.

Para detectar el caso en que la violación es generada por más de una tupla con la misma clave, generamos la consulta:

select count(\*), rubro\_emp, id\_emp, id\_ciudad\_oper\_emp, id\_resp\_emp, id\_prod

from empresas

group by rubro\_emp, id\_emp, id\_ciudad\_oper\_emp, id\_resp\_emp, id\_prod

having count(\*) > 1;

En este caso el resultado nos muestra en cada tupla, qué claves están duplicadas en la tabla, y cuantas veces aparece duplicada.

## Ejercicio 6)

### Parte a)

CREATE TABLE public.empresas2

(

id\_emp character varying NOT NULL,

nom\_emp character varying,

desc\_emp text,

rubro\_emp character varying,

id\_ciudad\_oper\_emp character varying,

ciudad\_oper\_emp character varying,

id\_pais\_oper\_emp character varying,

pais\_oper\_emp character varying,

fecha\_fund\_emp integer,

id\_resp\_emp character varying,

nom\_resp\_emp character varying,

ap\_resp\_emp character varying,

fnac\_resp\_emp date,

id\_ciu\_nac\_resp\_emp character varying,

ciu\_nac\_resp\_emp character varying,

id\_pais\_nac\_resp\_emp character varying,

pais\_nac\_resp\_emp character varying,

id\_prod character varying,

nom\_prod character varying,

desc\_prod character varying

)

WITH (

OIDS=FALSE

);

### Parte b)

ALTER TABLE public.empresas2 ADD PRIMARY KEY (rubro\_emp, id\_emp, id\_ciudad\_oper\_emp, id\_resp\_emp, id\_prod);

### Parte c)

insert into public.empresas2

(

id\_emp, nom\_emp, desc\_emp, rubro\_emp, id\_ciudad\_oper\_emp, ciudad\_oper\_emp, id\_pais\_oper\_emp, pais\_oper\_emp, fecha\_fund\_emp, id\_resp\_emp, nom\_resp\_emp, ap\_resp\_emp, fnac\_resp\_emp, id\_ciu\_nac\_resp\_emp, ciu\_nac\_resp\_emp, id\_pais\_nac\_resp\_emp, pais\_nac\_resp\_emp, id\_prod, nom\_prod, desc\_prod

)

select

coalesce(nullif(id\_emp,''), 'SIN INFORMACION DE EMPRESA'), nom\_emp, desc\_emp, coalesce(nullif(rubro\_emp,''), 'SIN INFORMACION DE RUBRO'), coalesce(nullif(id\_ciudad\_oper\_emp,''), 'SIN INFORMACION DE CIUDAD'), ciudad\_oper\_emp, id\_pais\_oper\_emp, pais\_oper\_emp, fecha\_fund\_emp, coalesce(nullif(id\_resp\_emp,''), 'SIN INFORMACION DE RESPONSABLE'), nom\_resp\_emp, ap\_resp\_emp, fnac\_resp\_emp, id\_ciu\_nac\_resp\_emp, ciu\_nac\_resp\_emp, id\_pais\_nac\_resp\_emp, pais\_nac\_resp\_emp, coalesce(nullif(id\_prod,''), 'SIN INFORMACION DE PRODUCTO'), nom\_prod, desc\_prod

from public.empresas

ON CONFLICT (rubro\_emp, id\_emp, id\_ciudad\_oper\_emp, id\_resp\_emp, id\_prod) DO NOTHING;

Para asegurarnos que no se ha perdido ninguna empresa generamos la siguiente consulta, la cual busca cualquier empresa que esté en la tabla original (empresas) y no esté en empresas2.

select distinct id\_emp

from empresas e

where id\_emp not in

(select id\_emp from empresas2 e2 where e2.id\_emp = e.id\_emp);

### Parte d)

Para el caso en el que la violación es por valores nulos en la clave, decidimos reemplazar los nulos o textos vacíos que encontramos en los atributos de la clave, por textos descriptivos relacionados a ese atributo (por ejemplo: SIN INFORMACION DE RUBRO cuando rubro\_emp en la tupla es nulo o vacío).

Para el caso en el que la violación es por duplicidad de clave, decidimos quedarnos con la primera de las tuplas encontradas, lo que resultó en no realizar ninguna acción al momento de encontrar una tupla que tenga los mismos valores en los atributos de la clave. Esta acción se realiza con la instrucción “do nothing”.

## Ejercicio 7)

### Parte a)

CREATE TABLE public.empresas3

(

id\_emp character varying NOT NULL,

nom\_emp character varying,

desc\_emp text,

rubro\_emp character varying,

id\_ciudad\_oper\_emp character varying,

ciudad\_oper\_emp character varying,

id\_pais\_oper\_emp character varying,

pais\_oper\_emp character varying,

fecha\_fund\_emp integer,

id\_resp\_emp character varying,

nom\_resp\_emp character varying,

ap\_resp\_emp character varying,

fnac\_resp\_emp date,

id\_ciu\_nac\_resp\_emp character varying,

ciu\_nac\_resp\_emp character varying,

id\_pais\_nac\_resp\_emp character varying,

pais\_nac\_resp\_emp character varying,

id\_prod character varying,

nom\_prod character varying,

desc\_prod character varying

)

WITH (

OIDS=FALSE

);

ALTER TABLE public.empresas3

ADD PRIMARY KEY

(rubro\_emp, id\_emp, id\_ciudad\_oper\_emp, id\_resp\_emp, id\_prod);

### Parte b)

DROP FUNCTION IF EXISTS public.processingNewLine() CASCADE;

CREATE FUNCTION public.processingNewLine() RETURNS trigger AS $$

BEGIN

IF NEW.id\_emp is NULL OR NEW.id\_emp = '' THEN

NEW.id\_emp := 'SIN INFORMACION DE LA EMPRESA';

END IF;

IF NEW.rubro\_emp is NULL OR NEW.rubro\_emp = '' THEN

NEW.rubro\_emp := 'SIN INFORMACION DEL RUBRO';

END IF;

IF NEW.id\_ciudad\_oper\_emp is NULL OR NEW.id\_ciudad\_oper\_emp = '' THEN

NEW.id\_ciudad\_oper\_emp := 'SIN INFORMACION DE LA CIUDAD';

END IF;

IF NEW.id\_resp\_emp is NULL OR NEW.id\_resp\_emp = '' THEN

NEW.id\_resp\_emp := 'SIN INFORMACION DEL RESPONSABLE';

END IF;

IF NEW.id\_prod is NULL OR NEW.id\_prod = '' THEN

NEW.id\_prod := 'SIN INFORMACION DEL PRODUCTO';

END IF;

IF EXISTS(SELECT 1 FROM public.empresas3

WHERE id\_emp = NEW.id\_emp AND id\_ciudad\_oper\_emp = NEW.id\_ciudad\_oper\_emp

AND rubro\_emp = NEW.rubro\_emp AND id\_prod = NEW.id\_prod

AND id\_resp\_emp = NEW.id\_resp\_emp) THEN

RETURN NULL;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER processing\_before\_inserting

BEFORE INSERT ON public.empresas3

FOR EACH ROW

EXECUTE PROCEDURE public.processingNewLine();

Y la sentencia para cargar empresas3 con los datos de empresas:

insert into public.empresas3

(

id\_emp, nom\_emp, desc\_emp, rubro\_emp, id\_ciudad\_oper\_emp, ciudad\_oper\_emp, id\_pais\_oper\_emp, pais\_oper\_emp, fecha\_fund\_emp, id\_resp\_emp, nom\_resp\_emp, ap\_resp\_emp, fnac\_resp\_emp, id\_ciu\_nac\_resp\_emp, ciu\_nac\_resp\_emp, id\_pais\_nac\_resp\_emp, pais\_nac\_resp\_emp, id\_prod, nom\_prod, desc\_prod

)

select

id\_emp, nom\_emp, desc\_emp, rubro\_emp, id\_ciudad\_oper\_emp, ciudad\_oper\_emp, id\_pais\_oper\_emp, pais\_oper\_emp, fecha\_fund\_emp, id\_resp\_emp, nom\_resp\_emp, ap\_resp\_emp, fnac\_resp\_emp, id\_ciu\_nac\_resp\_emp, ciu\_nac\_resp\_emp, id\_pais\_nac\_resp\_emp, pais\_nac\_resp\_emp, id\_prod, nom\_prod, desc\_prod

from public.empresas;

Para asegurarnos que no se ha perdido ninguna empresa generamos la siguiente consulta, la cual busca cualquier empresa que esté en la tabla original (empresas) y no esté en empresas3.

select distinct id\_emp

from empresas e

where id\_emp not in

(select id\_emp from empresas3 e3 where e3.id\_emp = e.id\_emp);

# Actividad 2)

## Ejercicio 8)

### Parte a)

Comprobamos que la tabla empresas no estaba en 3NF. Procederemos a

descomponer la tabla empresas en 3NF con join sin pérdida.

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre: EMPRESA (R1)  Dependencias funcionales: { id\_emp → nom\_emp, desc\_emp, fecha\_fund\_emp } | | | |
| Id\_emp | nom\_emp | desc\_emp | fecha\_fund\_emp |

|  |  |  |
| --- | --- | --- |
| Nombre: CIUDAD DONDE OPERA (R2)  Dependencias funcionales: { id\_ciudad\_oper\_emp → ciudad\_oper\_emp, id\_pais\_oper\_emp } | | |
| id\_ciudad\_oper\_emp | ciudad\_oper\_emp | id\_pais\_oper\_emp |

|  |  |
| --- | --- |
| Nombre: PASI DONDE OPERA (R3)  Dependencias funcionales: { id\_pais\_oper\_emp → pais\_oper\_emp } | |
| id\_pais\_oper\_emp | pais\_oper\_emp |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre: RESPONSABLES EMPRESA (R4)  Dependencias funcionales: { id\_resp\_emp → nom\_resp\_emp, ap\_resp\_emp, fnac\_resp\_emp, id\_ciu\_nac\_resp\_emp } | | | | |
| id\_resp\_emp | nom\_resp\_emp | ap\_resp\_emp | fnac\_resp\_emp | id\_ciu\_nac\_resp\_emp |

|  |  |  |
| --- | --- | --- |
| Nombre: CIUDAD DE NACIMIENTO RESPONSABLE (R5)  Dependencias funcionales: { id\_ciu\_nac\_resp\_emp → ciu\_nac\_resp\_emp, id\_pais\_nac\_resp\_emp } | | |
| id\_ciu\_nac\_resp\_emp | ciu\_nac\_resp\_emp | id\_pais\_nac\_resp\_emp |

|  |  |
| --- | --- |
| Nombre: PAÍS DE NACIMIENTO RESPONSABLES EMPRESA (R6)  Dependencias funcionales: { id\_pais\_nac\_resp\_emp → pais\_nac\_resp\_emp } | |
| id\_pais\_nac\_resp\_emp | pais\_nac\_resp\_emp |

|  |  |  |
| --- | --- | --- |
| Nombre: PRODUCTO (R7)  Dependencias funcionales: { id\_prod → nom\_prod, desc\_prod } | | |
| id\_prod | nom\_prod | desc\_prod |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre: CLAVE (R8)  Dependencias funcionales: { } | | | | |
| rubro\_emp | Id\_emp | Id\_ciudad\_oper\_emp | Id\_resp\_emp | id\_prod |

No hay pérdida de dependencias funcionales debido a que el algoritmo visto en el curso ya preserva las dependencias funcionales.

### Parte b)

Unificamos los siguientes esquemas:

|  |  |  |
| --- | --- | --- |
| Nombre tabla: CIUDAD DONDE OPERA.  Dependencias funcionales: { id\_ciudad\_oper\_emp → ciudad\_oper\_emp, id\_pais\_oper\_emp } | | |
| id\_ciudad\_oper\_emp | ciudad\_oper\_emp | id\_pais\_oper\_emp |

|  |  |  |
| --- | --- | --- |
| Nombre tabla: CIUDAD DE NACIMIENTO RESPONSABLE  Dependencias funcionales: { id\_ciu\_nac\_resp\_emp → ciu\_nac\_resp\_emp, id\_pais\_nac\_resp\_emp } | | |
| id\_ciu\_nac\_resp\_emp | ciu\_nac\_resp\_emp | id\_pais\_nac\_resp\_emp |

A los atributos les cambiaremos su nombre para que no se confundan:

* **Id\_ciudad\_oper\_emp** y **id\_ciu\_nac\_resp\_emp** renombrado a: **id\_ciudad**.
* **ciudad\_oper\_emp** y **ciu\_nac\_resp\_emp** renombrado a: **ciudad**.
* **Id\_pais\_oper\_emp** y **id\_pais\_nac\_resp\_emp** renombrado a: **id\_pais**.

Concluimos con el esquema:

|  |  |  |
| --- | --- | --- |
| Nombre: CIUDADES (R2N)  Dependencias funcionales: { id\_ciudad → ciudad, id\_pais } | | |
| id\_ciudad | ciudad | id\_pais |

Análogamente los siguientes esquemas pasarán a estar unificados en el esquema pais:

|  |  |
| --- | --- |
| Nombre tabla: PAIS DONDE OPERA  Dependencias funcionales: { id\_pais\_oper\_emp → pais\_oper\_emp } | |
| id\_pais\_oper\_emp | pais\_oper\_emp |

|  |  |
| --- | --- |
| Nombre tabla: PAÍS DE NACIMIENTO RESPONSABLES EMPRESA  Dependencias funcionales: { id\_pais\_nac\_resp\_emp → pais\_nac\_resp\_emp } | |
| id\_pais\_nac\_resp\_emp | pais\_nac\_resp\_emp |

A los atributos les cambiaremos su nombre para que no se confundan:

* **id\_pais\_oper\_emp** y **id\_pais\_nac\_resp\_emp** renombrado a: **id\_pais**.
* **pais\_oper\_emp** y **pais\_nac\_resp\_emp** remobrado a: **pais**.

|  |  |
| --- | --- |
| Nombre tabla: PAÍS (R3N)  Dependencias funcionales: { id\_pais → pais } | |
| id\_pais | pais |

El nuevo conjunto de esquemas se mantiene en 3NF ya que cada esquema tiene dependencias funcionales (X→Y) en las que X es superclave.

Debido a los procesos de renombre, las dependencias funcionales quedan de la siguiente manera:

id\_emp → nom\_emp, desc\_emp, fecha\_fund\_emp

id\_ciudad → ciudad, id\_pais

id\_pais → pais

id\_resp\_emp → nom\_resp\_emp, ap\_resp\_emp, fnac\_resp\_emp, id\_ciudad

id\_prod → nom\_prod, desc\_prod

## Ejercicio 9)

Debido a la realidad planteada en la letra, sabemos que una misma empresa no está asociada a un rubro específico, ni a una ciudad en específico, ni a un responsable en específico, ni produce un producto en específico.

Debido a que no encontramos una relación entre los atributos (rubro\_emp, id\_ciudad, id\_resp\_emp, id\_prod) que son los utilizados para indicar la información descrita arriba, entonces inferimos las siguientes dmv:

id\_emp ->> rubro\_emp

id\_emp ->> id\_ciudad

id\_emp ->> id\_resp\_emp

id\_emp ->> id\_prod

Estas dmv son no triviales.

## Ejercicio 10)

El esquema R8 es:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre: CLAVE (R8)  Dependencias funcionales: { }  Dependencias multivaluadas: { id\_emp ->> rubro\_emp, id\_emp ->> id\_ciudad, id\_emp ->> id\_resp\_emp, id\_emp ->> id\_prod } | | | | |
| rubro\_emp | Id\_emp | Id\_ciudad | Id\_resp\_emp | id\_prod |

Encontramos que el esquema R8 no cumple 4NF, dado que, por ejemplo, la dependencia multivaluada id\_emp ->> rubro\_emp ya descrita, es no trivial, y id\_emp no es superclave de R8.

Por lo tanto aplicaremos en primera instancia el algoritmo de descomposición en 4NF sobre la dependencia multivaluada (id\_emp ->> rubro\_emp). Este algoritmo devuelve dos nuevos esquemas.

Ahora verificamos cual de estos esquemas viola 4NF, y en caso de existir una violación aplicaremos nuevamente el algoritmo. Este proceso culmina con los siguientes esquemas.

|  |  |
| --- | --- |
| Nombre: RUBROS DE EMPRESAS (R81)  Dependencias funcionales: { }  Dependencias multivaluadas: { id\_emp ->> rubro\_emp} | |
| Id\_emp | rubro\_emp |

|  |  |
| --- | --- |
| Nombre: CIUDADES DE EMPRESAS (R82)  Dependencias funcionales: { }  Dependencias multivaluadas: {id\_emp ->> id\_ciudad} | |
| Id\_emp | Id\_ciudad |

|  |  |
| --- | --- |
| Nombre: RESPONSABLES DE EMPRESAS (R83)  Dependencias funcionales: { }  Dependencias multivaluadas: { id\_emp ->> id\_resp\_emp } | |
| Id\_emp | Id\_resp\_emp |

|  |  |
| --- | --- |
| Nombre: PRODUCTOS DE EMPRESAS (R84)  Dependencias funcionales: { }  Dependencias multivaluadas: { id\_emp ->> id\_prod } | |
| Id\_emp | id\_prod |

No hay pérdida de dependencias funcionales ya que el esquema que se modificó para llevar a 4NF no tenía dependencias funcionales.

## Ejercicio 11)

### Parte a)

Se generan las tablas:

CREATE TABLE public.R1

(

id\_emp character varying NOT NULL,

nom\_emp character varying,

desc\_emp text,

fecha\_fund\_emp integer

)

WITH (

OIDS=FALSE

);

CREATE TABLE public.R2N

(

id\_ciudad character varying,

ciudad character varying,

id\_pais character varying

)

WITH (

OIDS=FALSE

);

CREATE TABLE public.R3N

(

id\_pais character varying,

pais character varying

)

WITH (

OIDS=FALSE

);

CREATE TABLE public.R4

(

id\_resp\_emp character varying,

nom\_resp\_emp character varying,

ap\_resp\_emp character varying,

fnac\_resp\_emp date,

id\_ciudad character varying

)

WITH (

OIDS=FALSE

);

CREATE TABLE public.R7

(

id\_prod character varying,

nom\_prod character varying,

desc\_prod character varying

)

WITH (

OIDS=FALSE

);

CREATE TABLE public.R81

(

id\_emp character varying NOT NULL,

rubro\_emp character varying

)

WITH (

OIDS=FALSE

);

CREATE TABLE public.R82

(

id\_emp character varying NOT NULL,

id\_ciudad character varying

)

WITH (

OIDS=FALSE

);

CREATE TABLE public.R83

(

id\_emp character varying NOT NULL,

id\_resp\_emp character varying

)

WITH (

OIDS=FALSE

);

CREATE TABLE public.R84

(

id\_emp character varying NOT NULL,

id\_prod character varying

)

WITH (

OIDS=FALSE

);

Se generan todas las claves:

ALTER TABLE public.R1 ADD PRIMARY KEY (id\_emp);

ALTER TABLE public.R2N ADD PRIMARY KEY (id\_ciudad);

ALTER TABLE public.R3N ADD PRIMARY KEY (id\_pais);

ALTER TABLE public.R4 ADD PRIMARY KEY (id\_resp\_emp);

ALTER TABLE public.R7 ADD PRIMARY KEY (id\_prod);

ALTER TABLE public.R81 ADD PRIMARY KEY (id\_emp, rubro\_emp);

ALTER TABLE public.R82 ADD PRIMARY KEY (id\_emp, id\_ciudad);

ALTER TABLE public.R83 ADD PRIMARY KEY (id\_emp, id\_resp\_emp);

ALTER TABLE public.R84 ADD PRIMARY KEY (id\_emp, id\_prod);

Se generan todas las foreign keys:

ALTER TABLE public.R2N ADD FOREIGN KEY (id\_pais) REFERENCES public.R3N (id\_pais);

ALTER TABLE public.R4 ADD FOREIGN KEY (id\_ciudad) REFERENCES public.R2N (id\_ciudad);

ALTER TABLE public.R81 ADD FOREIGN KEY (id\_emp) REFERENCES public.R1 (id\_emp);

ALTER TABLE public.R82 ADD FOREIGN KEY (id\_emp) REFERENCES public.R1 (id\_emp);

ALTER TABLE public.R82 ADD FOREIGN KEY (id\_ciudad) REFERENCES public.R2N (id\_ciudad);

ALTER TABLE public.R83 ADD FOREIGN KEY (id\_emp) REFERENCES public.R1 (id\_emp);

ALTER TABLE public.R83 ADD FOREIGN KEY (id\_resp\_emp) REFERENCES public.R4 (id\_resp\_emp);

ALTER TABLE public.R84 ADD FOREIGN KEY (id\_emp) REFERENCES public.R1 (id\_emp);

ALTER TABLE public.R84 ADD FOREIGN KEY (id\_prod) REFERENCES public.R7 (id\_prod);

### Parte b)

Se cargarán todas las tablas en el siguiente orden:

* Los países donde operan las empresas.
* Los países donde nacieron los responsables.
* Las ciudades donde operan las personas.
* Las ciudades donde nacieron los responsables.
* Las empresas.
* Los responsable de empresas.
* Los productos.
* Las relaciones entre empresas y sus rubros.
* Las relaciones entre empresas y sus ciudades.
* Las relaciones entre empresas y sus responsables.
* Las relaciones entre empresas y sus productos.

INSERT INTO public.R3N (id\_pais, pais)

SELECT id\_pais\_oper\_emp id\_pais, pais\_oper\_emp pais FROM public.empresas

where id\_pais\_oper\_emp is not null and id\_pais\_oper\_emp <> ''

ON CONFLICT (id\_pais) DO NOTHING;

INSERT INTO public.R3N (id\_pais, pais)

SELECT id\_pais\_nac\_resp\_emp id\_pais, pais\_nac\_resp\_emp pais FROM public.empresas

where id\_pais\_nac\_resp\_emp is not null and id\_pais\_nac\_resp\_emp <> ''

ON CONFLICT (id\_pais) DO NOTHING;

INSERT INTO public.R2N (id\_ciudad, ciudad, id\_pais)

SELECT id\_ciudad\_oper\_emp id\_ciudad, ciudad\_oper\_emp ciudad, id\_pais\_oper\_emp id\_pais FROM public.empresas

where id\_ciudad\_oper\_emp is not null and id\_ciudad\_oper\_emp <> ''

ON CONFLICT (id\_ciudad) DO NOTHING;

INSERT INTO public.R2N (id\_ciudad, ciudad, id\_pais)

SELECT id\_ciu\_nac\_resp\_emp id\_ciudad, ciu\_nac\_resp\_emp ciudad, id\_pais\_nac\_resp\_emp id\_pais FROM public.empresas

where id\_ciu\_nac\_resp\_emp is not null and id\_ciu\_nac\_resp\_emp <> ''

ON CONFLICT (id\_ciudad) DO NOTHING;

INSERT INTO public.R1 (id\_emp, nom\_emp, desc\_emp, fecha\_fund\_emp)

SELECT id\_emp, nom\_emp, desc\_emp, fecha\_fund\_emp FROM public.empresas

WHERE id\_emp is not null and id\_emp <> ''

ON CONFLICT (id\_emp) DO NOTHING;

INSERT INTO public.R4 (id\_resp\_emp, nom\_resp\_emp, ap\_resp\_emp, fnac\_resp\_emp, id\_ciudad)

SELECT id\_resp\_emp, nom\_resp\_emp, ap\_resp\_emp, fnac\_resp\_emp, id\_ciu\_nac\_resp\_emp id\_ciudad FROM public.empresas

where id\_resp\_emp is not null and id\_resp\_emp <> ''

ON CONFLICT (id\_resp\_emp) DO NOTHING;

INSERT INTO public.R7 (id\_prod, nom\_prod, desc\_prod)

SELECT id\_prod, nom\_prod, desc\_prod FROM public.empresas

where id\_prod is not null and id\_prod <> ''

ON CONFLICT (id\_prod) DO NOTHING;

INSERT INTO public.R81 (id\_emp, rubro\_emp)

SELECT id\_emp, rubro\_emp FROM public.empresas

where id\_emp is not null and rubro\_emp is not null

and id\_emp <> '' and rubro\_emp <> ''

ON CONFLICT (id\_emp, rubro\_emp) DO NOTHING;

INSERT INTO public.R82 (id\_emp, id\_ciudad)

SELECT id\_emp, id\_ciudad\_oper\_emp id\_ciudad FROM public.empresas

where id\_emp is not null and id\_ciudad\_oper\_emp is not null

and id\_emp <> '' and id\_ciudad\_oper\_emp <> ''

ON CONFLICT (id\_emp, id\_ciudad) DO NOTHING;

INSERT INTO public.R83 (id\_emp, id\_resp\_emp)

SELECT id\_emp, id\_resp\_emp FROM public.empresas

where id\_emp is not null and id\_resp\_emp is not null

and id\_emp <> '' and id\_resp\_emp <> ''

ON CONFLICT (id\_emp, id\_resp\_emp) DO NOTHING;

INSERT INTO public.R84 (id\_emp, id\_prod)

SELECT id\_emp, id\_prod FROM public.empresas

where id\_emp is not null and id\_prod is not null

and id\_emp <> '' and id\_prod <> ''

ON CONFLICT (id\_emp, id\_prod) DO NOTHING;

## Ejercicio 12)

Para calcular la cantidad de tuplas de ambos esquemas, desarrollamos las siguientes consultas:

* Para el esquema en 4NF del ejercicio 11:

SELECT COUNT(\*) FROM public.R1 = 4712 tuplas.

SELECT COUNT(\*) FROM public.R2N = 810 tuplas.

SELECT COUNT(\*) FROM public.R3N = 160 tuplas.

SELECT COUNT(\*) FROM public.R4 = 364 tuplas.

SELECT COUNT(\*) FROM public.R7 = 1669 tuplas.

SELECT COUNT(\*) FROM public.R81 = 5807 tuplas.

SELECT COUNT(\*) FROM public.R82 = 1618 tuplas.

SELECT COUNT(\*) FROM public.R83 = 379 tuplas.

SELECT COUNT(\*) FROM public.R84 = 3168 tuplas.

SELECT

((SELECT COUNT(\*) FROM public.R1)

+(SELECT COUNT(\*) FROM public.R2N)

+(SELECT COUNT(\*) FROM public.R3N)

+(SELECT COUNT(\*) FROM public.R4)

+(SELECT COUNT(\*) FROM public.R7)

+(SELECT COUNT(\*) FROM public.R81)

+(SELECT COUNT(\*) FROM public.R82)

+(SELECT COUNT(\*) FROM public.R83)

+(SELECT COUNT(\*) FROM public.R84)) ts; = 18687 tuplas

* Para el esquema original (empresas):

SELECT COUNT(\*) FROM public.empresas; = 9210 tuplas.