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)

Descomposición de 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
((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;
```

- Para el esquema original (empresas):

```
SELECT COUNT(*) FROM public.empresas;
```

Resultando que en las tablas del ejercicio 11 tenemos **18687 tuplas** mientras que en la original (empresas) tenemos **9215 tuplas**.