

Práctica SQL



Autor: Pablo Benayas Penas

Índice:

1. Breve introducción
2. Diseño conceptual (Modelo Entidad-Relación)
3. Diseño lógico (Model Relacional)
4. Consultas
5. Conclusiones

1.- Breve introducción

En este ejercicio se va a proceder a crear una base de datos con la que posteriormente haremos consultas. Primeramente, se explicará de manera conceptual las distintas entidades que la componen, así como un gráfico que conecte estas entidades mediante relaciones.

Más adelante, se definirán las tablas en código SQL, explicando las particularidades de cada una de ellas.

Una vez especificado el código, se mostrará el diagrama entidad-relación que devuelve SQL con ese código.

Finalmente, se realizarán las consultas.

2.- Diseño conceptual (Modelo Entidad-Relación)

Identificación de relaciones, entidades y atributos:

- **tienda_aplicaciones**
 - **tienda_aplicacion_ID** (clave principal)
 - **administrador**
 - **direccion_web**

- **empresa_desarrolladora_apps**
 - **nombre_empresa_ID** (clave principal)
 - **pais**
 - **anno_fundacion**
 - **e_mail**
 - **pagina_web**

- **trabajadores**

- **trabajador_ID** (clave principal)
- **exp_profesional**
- **direccion** -- ('direccion' a su vez tiene 3 atributos:)
 - **direccion_calle**
 - **direccion_numero**
 - **codigo_postal**
- **telefono**

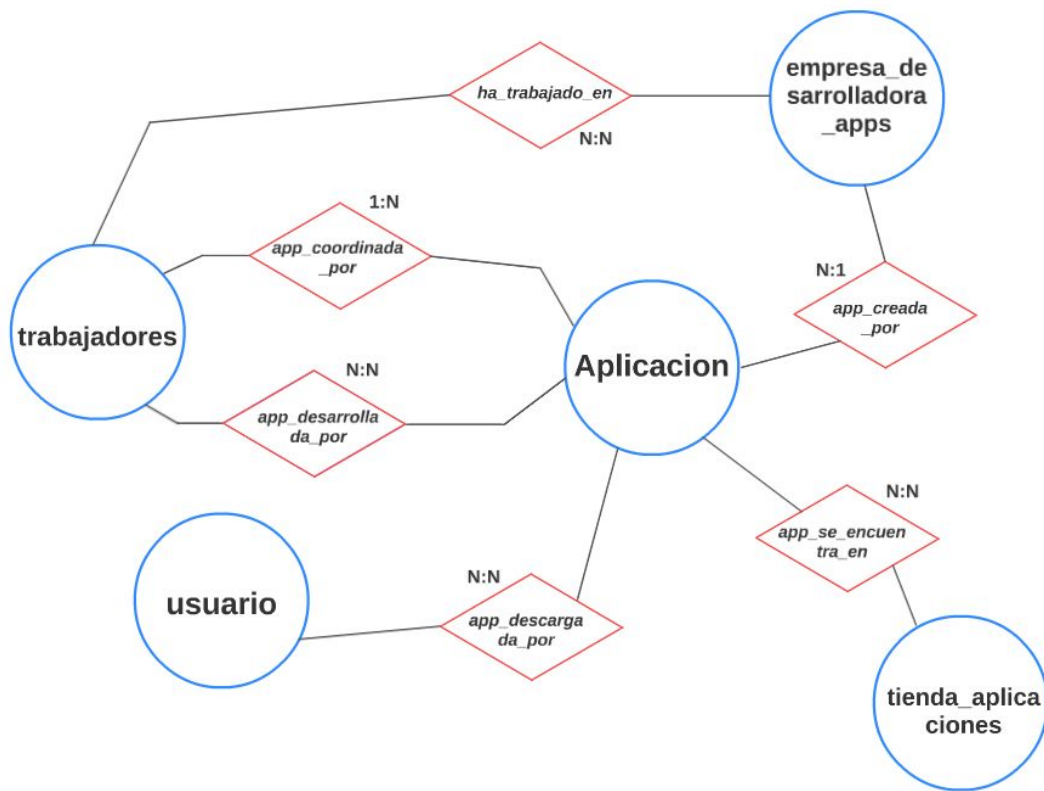
- **aplicacion**

- **aplicacion_ID** (clave principal)
- **fecha_comienzo**
- **fecha_fin**
- **nombre_aplicacion**
- **categoria**
- **espacio_de_memoria**
- **precio**

- **usuario**

- **cuenta_usuario** (clave principal)
- **nombre_usuario** (no puede ser clave principal porque un mismo usuario puede tener varias cuentas)
- **direccion**
- **descargada_por_movil** (valores: 'si' y 'no')
- **numero_de_telefono**

Gráfico:



A continuación, se va a explicar cada una de las relaciones:

- **app_descargada_por**: presenta relación N:N porque una aplicación puede ser descargada por varios usuarios y un usuario puede descargar varias aplicaciones.
- **app_desarrollada_por**: relación N:N porque una aplicación puede haber sido desarrollada por varios trabajadores y un trabajador puede haber participado en la realización de varias aplicaciones.
- **app_coordinada_por**: tiene relación 1:N porque una aplicación solo puede tener un único coordinador, mientras que un coordinador, como dice el enunciado: ‘puede dirigir varias aplicaciones’.
- **ha_trabajado_en**: tiene relación N:N porque una persona ‘puede haber trabajado en varias empresas del sector’ y una empresa desarrolladora de apps puede tener más de un único empleado.
- **app_creada_por**: tiene relación 1:N porque una aplicación solo puede ser creada por una única empresa desarrolladora (el enunciado dice textualmente ‘cada aplicación está realizada por un grupo de empleados’. Asumo que este grupo de empleados pertenece a la misma empresa).

Mientras que por otra parte, una empresa desarrolladora puede crear más de una aplicación.

- app_se_encuentra_en: relación N:N porque una aplicación se puede encontrar en varias tiendas de apps y una tienda de apps contiene varias aplicaciones.

3.- Diseño lógico (Modelo Relacional)

Primero voy a realizar el código SQL con el que se crean las tablas de las entidades, relaciones 1:N y relaciones N:N.

Una vez creado el código, mostraré el diagrama ER que se produce al hacer 'Reverse Engineer'. Al definir las tablas y atributos no voy a usar signos de puntuación para evitar problemas al ejecutar el programa.

-- CREAR ENTIDADES:

Transformación de la entidad TIENDA_APLICACIONES

```
CREATE TABLE tienda_aplicaciones(  
  tienda_aplicacion_ID VARCHAR(20) NOT NULL UNIQUE,  
  administrador VARCHAR(20),  
  direccion_web VARCHAR(50),  
  PRIMARY KEY (tienda_aplicacion_ID)  
);
```

Transformación de la entidad EMPRESA_DESARROLLADORA_APPS

```
CREATE TABLE empresa_desarrolladora_apps(  
  nombre_empresa_ID VARCHAR(20) NOT NULL UNIQUE,  
  pais VARCHAR(20),  
  anno_fundacion INT,  
  e_mail VARCHAR(100),  
  pagina_web VARCHAR(50),  
  PRIMARY KEY (nombre_empresa_ID)  
);
```

Transformación de la entidad TRABAJADORES

```
CREATE TABLE trabajadores(  
  trabajador_ID INT NOT NULL UNIQUE,  
  exp_profesional VARCHAR(20),  
  direccion_calle VARCHAR(30),  
  direccion_numero INT,  
  codigo_postal INT,  
  telefono INT,  
  PRIMARY KEY (trabajador_ID)  
);
```

Transformación de la entidad APLICACION (la dejo en comentario porque al hacer el script de SQL, definiré la entidad una vez incluidas las dos relaciones 1:N)

```
-- CREATE TABLE aplicacion(  
--  aplicacion_ID VARCHAR(20) NOT NULL UNIQUE,  
--  fecha_comienzo VARCHAR(20),  
--  fecha_fin VARCHAR(20),  
--  nombre_aplicacion VARCHAR(30),  
--  categoria VARCHAR(20),  
--  espacio_de_memoria INT,  
--  precio INT,  
--  PRIMARY KEY (aplicacion_ID)  
-- );
```

Transformación de la entidad USUARIO

```
CREATE TABLE usuario(  
  cuenta_usuario INT NOT NULL UNIQUE,  
  nombre_usuario VARCHAR(20),  
  direccion VARCHAR(50),  
  descargada_por_movil VARCHAR(5), -- dos posibles valores: si y no  
  numero_de_telefono INT,  
  PRIMARY KEY (cuenta_usuario) -- una misma persona puede tener distintas cuentas  
);
```

-- CREAR RELACIONES 1:N

Existen dos relaciones 1:N: 'app_creada_por' y 'app_coordinada_por'. En ambas, es la entidad 'aplicacion' la que solo puede tener un único valor (un único coordinador y una única empresa que la creó, respectivamente).

Por tanto, las relaciones quedan determinadas propagando la clave primaria de 'trabajadores' y 'empresa_desarrolladora_de_apps' en 'aplicacion'.

'Delete' y 'Update' de estas dos relaciones:

- Voy a usar 'Delete Restrict' porque si el trabajador coordinador es eliminado de la entidad 'trabajadores' (porque quizás haya cambiado de empresa), quiero que la aplicación que está asociada a ese trabajador coordinador sí se mantenga y, por tanto, su tupla no sea eliminada de la entidad 'aplicacion'.
- Voy a usar 'Update Cascade' porque si la clave primaria de una tupla de 'trabajadores' es modificada, quiero que también sea modificada su clave extranjera en 'aplicacion'.

Mismo escenario con empresa desarrolladora.

Para evitar el error de 'nombre de columna es ambigua' por hacer consultas usando dos columnas con mismo nombre, voy a llamar a las claves extranjeras con un nombre distinto.

En este caso, en vez de llamar a la clave extranjera 'trabajador_ID', la voy a llamar 'trabajador_coordinador_ID'.

En vez de llamar a la otra clave extranjera 'nombre_empresa_ID', la voy a llamar 'empresa_que_desarrollo_app_ID'.

Transformaciones:

-- Transformación de la relación "app_coordinada_por" entre APLICACION y

-- TRABAJADORES

-- Transformación de la relación "app_creada_por" entre APLICACION y

-- EMPRESA_DESARROLLADORA_APPS

Código:

```
CREATE TABLE aplicacion(  
  aplicacion_ID VARCHAR(20) NOT NULL UNIQUE,  
  fecha_comienzo VARCHAR(20),  
  fecha_fin VARCHAR(20),  
  nombre_aplicacion VARCHAR(30),  
  categoria VARCHAR(20),  
  espacio_de_memoria INT,
```

```

precio INT,
trabajador_coordinador_ID INT NOT NULL UNIQUE,
empresa_que_desarrollo_app_ID VARCHAR(20) NOT NULL UNIQUE,
PRIMARY KEY (aplicacion_ID),
FOREIGN KEY (trabajador_coordinador_ID) REFERENCES trabajadores(trabajador_ID)
ON DELETE RESTRICT
ON UPDATE CASCADE,
FOREIGN KEY (empresa_que_desarrollo_app_ID) REFERENCES
empresa_desarrolladora_apps(nombre_empresa_ID)
ON DELETE RESTRICT
ON UPDATE CASCADE
);

```

-- CREAR RELACIONES N:N

Las relaciones N:N requieren crear una nueva tabla para que la relación quede definida.
En esta nueva tabla hay que añadir la clave primaria de ambas entidades.

‘Delete’ y ‘Update’ de estas relaciones:

- Voy a usar ‘delete cascade’ porque si elimino un valor de la clave primaria de la entidad, quiero que ese valor también sea borrado en la clave extranjera, la cual se encuentra en la nueva tabla creada.
Por ejemplo, en la relación ‘app_se_encuentra_en’ que tienen las entidades ‘aplicacion’ y ‘tienda_aplicaciones’, si una tienda de aplicaciones deja de ser operativa, simplemente la relación ‘app_se_encuentra_en’ elimina todas las tuplas que contienen el nombre de sea tienda.
La entidad ‘aplicacion’, sin embargo, no ha tenido ninguna modificación en este proceso.
- Voy a usar ‘Update Cascade’ porque si la clave primaria de una tupla es modificada, quiero que también sea modificada su clave extranjera en la relación.
- Mientras que la eliminación de una clave primaria implica la eliminación de la tupla entera, la modificación solamente cambia el campo de la clave.

(Aclaración: aunque el nombre de la clave primaria y extranjera sea distinto, ambas columnas guardan exactamente los mismos valores.)

Para evitar el error de ‘nombre de columna es ambigua’, también voy a llamar a las claves extranjeras de las relaciones N:N con un nombre distinto a la de la clave primaria original.

Transformación de la relación “app_se_encuentra_en” entre APLICACION y TIENDA_APLICACIONES

```
CREATE TABLE app_se_encuentra_en(  
  aplicacion_ID_1 VARCHAR(20),  
  tienda_aplicacion_ID_1 VARCHAR(20),  
  PRIMARY KEY (aplicacion_ID_1, tienda_aplicacion_ID_1),  
  FOREIGN KEY (aplicacion_ID_1) REFERENCES aplicacion(aplicacion_ID)  
  ON DELETE CASCADE  
  ON UPDATE CASCADE,  
  FOREIGN KEY (tienda_aplicacion_ID_1) REFERENCES  
  tienda_aplicaciones(tienda_aplicacion_ID)  
  ON DELETE CASCADE  
  ON UPDATE CASCADE  
);
```

Transformación de la relación “app_descargada_por” entre APLICACION y USUARIO

(En esta relación se va a incluir los siguientes atributos: ‘fecha_de_descarga’, ‘pais_donde_se_realizo_descarga’, ‘comentarios’ y ‘puntuacion’).

```
CREATE TABLE app_descargada_por(  
  cuenta_usuario_1 INT,  
  aplicacion_ID_2 VARCHAR(20),  
  fecha_de_descarga VARCHAR(20), -- (then, convert to DATETIME)  
  pais_donde_se_realizo_descarga VARCHAR(20),  
  comentarios VARCHAR(150),  
  puntuacion INT,  
  PRIMARY KEY (cuenta_usuario_1, aplicacion_ID_2),  
  FOREIGN KEY (cuenta_usuario_1) REFERENCES usuario(cuenta_usuario)
```

```
ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (aplicacion_ID_2) REFERENCES aplicacion(aplicacion_ID)
ON DELETE CASCADE
ON UPDATE CASCADE
);
```

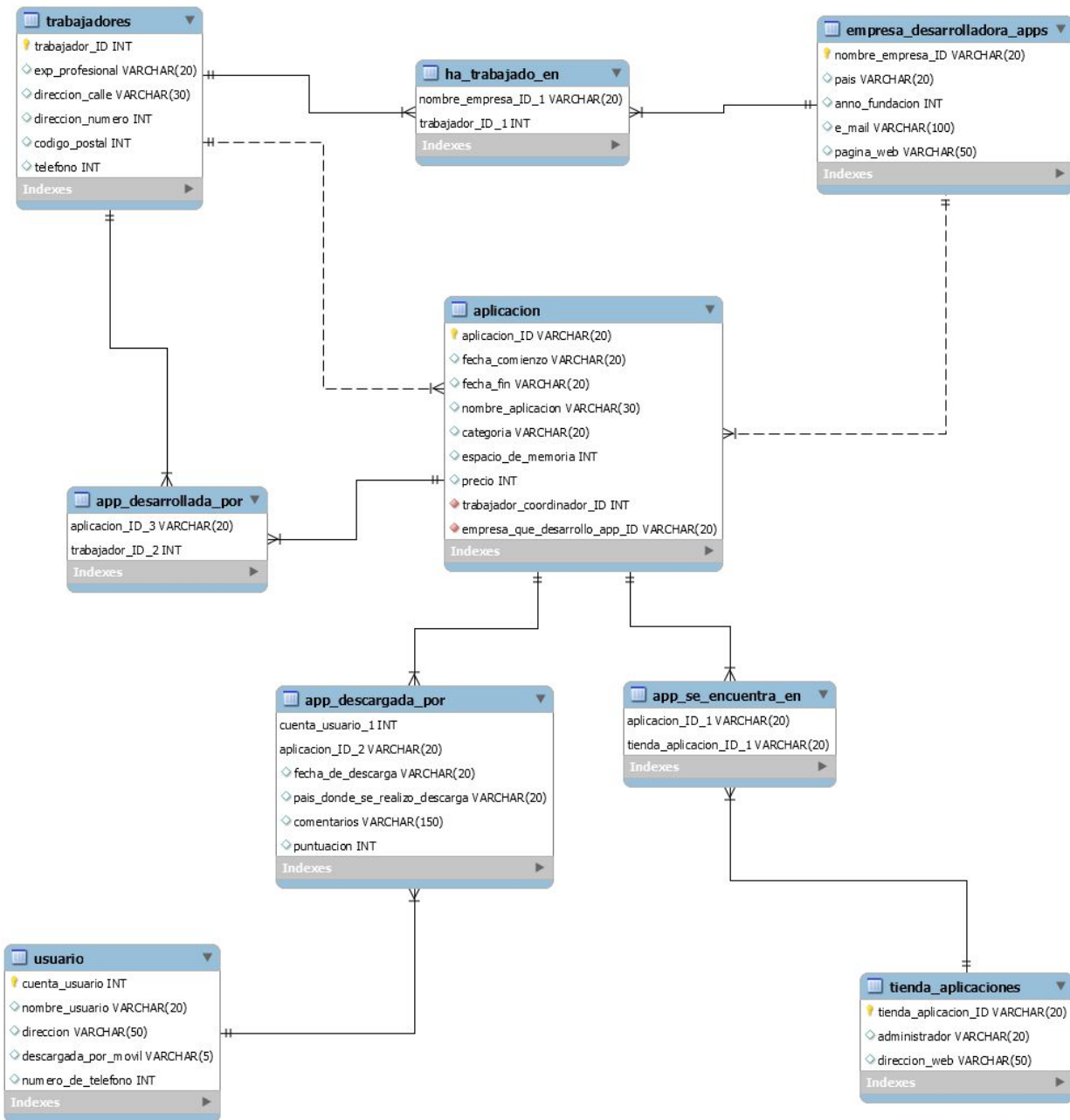
Transformación de la relación “ha_trabajado_en” entre TRABAJADORES y EMPRESA_DESARROLLADORA_APPS

```
CREATE TABLE ha_trabajado_en(
nombre_empresa_ID_1 VARCHAR(20),
trabajador_ID_1 INT,
PRIMARY KEY (nombre_empresa_ID_1, trabajador_ID_1),
FOREIGN KEY (nombre_empresa_ID_1) REFERENCES
empresa_desarrolladora_apps(nombre_empresa_ID)
ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (trabajador_ID_1) REFERENCES trabajadores(trabajador_ID)
ON DELETE CASCADE
ON UPDATE CASCADE
);
```

Transformación de la relación “app_desarrollada_por” entre APLICACION y TRABAJADORES

```
CREATE TABLE app_desarrollada_por(
aplicacion_ID_3 VARCHAR(20),
trabajador_ID_2 INT,
PRIMARY KEY (aplicacion_ID_3, trabajador_ID_2),
FOREIGN KEY (aplicacion_ID_3) REFERENCES aplicacion(aplicacion_ID)
ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (trabajador_ID_2) REFERENCES trabajadores(trabajador_ID)
ON DELETE CASCADE
ON UPDATE CASCADE
);
```

Diagrama Entidad-Relación usando 'Reversed Engineer':



4.- Consultas

Una vez creada la base de datos, se va a proceder a realizar consultas:

Pero antes, es buena práctica visualizar todas las tablas creadas (ver inserción de valores en el archivo SQL)

```
select * from app_se_encuentra_en;
```

aplicacion_ID_1	tienda_aplicacion_ID_1
airbnb_AA	App Store
Spotify_01	App Store
wechat_alpha	App Store
whatsapp_alpha	App Store
airbnb_AA	AppStore
Spotify_01	AppStore
whatsapp_alpha	AppStore
airbnb_AA	Google Play Store
UBER_01	Google Play Store
wechat_alpha	Google Play Store
whatsapp_alpha	Google Play Store
UBER_01	Xiaomi App Store
wechat_alpha	Xiaomi App Store
whatsapp_alpha	Xiaomi App Store
NULL	NULL

```
select * from app_descargada_por;
```

A	B	C	D	E	F
cuenta_usuario	aplicacion_ID_2	fecha_de_descarga	pais_donde_se_realizo_descarga	comentarios	puntuacion
72	Spotify_01	2017-06-23	Iran	NULL	4
121	wechat_alpha	2019-06-05	Portugal	Me voy a China y esta aplicacion es muy util alli	4
124	wechat_alpha	2016-07-21	China	NULL	3
256	Spotify_01	2016-10-23	Angola	NULL	4
312	airbnb_AA	2016-01-08	Iran	NULL	1
404	Spotify_01	2018-03-10	Rusia	NULL	2
500	airbnb_AA	2017-03-01	Canada	Es un timo!!! No te devuelven el dinero	1
2218	Spotify_01	2016-03-08	Japon	NULL	1
7001	wechat_alpha	2018-07-19	India	NULL	4
9001	whatsapp_alpha	2016-04-02	Iran	NULL	4
10045	UBER_01	2016-12-22	China	NULL	3
12412	airbnb_AA	2015-11-22	Portugal	NULL	4
20101	wechat_alpha	2017-07-16	Angola	Esta version no es la nueva que han sacado	2
41937	whatsapp_alpha	2017-02-13	Rusia	NULL	4
45412	UBER_01	2015-01-16	Rusia	NULL	5
72651	wechat_alpha	2016-01-08	India	NULL	4
81237	UBER_01	2015-02-06	Japon	NULL	5
84359	UBER_01	2015-10-22	India	Los conductores de uber en Bangalore ponen la musica muy a	2
90477	whatsapp_alpha	2019-02-14	Gabon	NULL	3
123777	whatsapp_alpha	2019-11-27	Vaticano	NULL	4
182979	Spotify_01	2015-03-21	India	NULL	5
182979	UBER_01	2015-03-21	India	NULL	5
182979	wechat_alpha	2015-03-21	India	NULL	5
196712	UBER_01	2016-04-07	China	NULL	4
197873	whatsapp_alpha	2018-11-28	China	NULL	3
259159	Spotify_01	2016-09-30	Iran	NULL	4
313176	airbnb_AA	2018-01-13	Rusia	NULL	2
341218	whatsapp_alpha	2018-07-25	China	NULL	4
393732	airbnb_AA	2016-07-22	Rusia	NULL	4
393732	Spotify_01	2016-07-21	Rusia	NULL	1
393732	UBER_01	2016-07-30	Rusia	NULL	5
393732	wechat_alpha	2016-07-02	Rusia	NULL	3
393732	whatsapp_alpha	2016-07-30	Rusia	NULL	2
5001245	Spotify_01	2015-09-12	Vaticano	NULL	4

select * from ha_trabajado_en;

A	B	C
nombre_em	trabajador_ID_1	
Fluper	43006	
Intellectsoft	111104	
TechAhead	123305	
Fueled	123400	
OpenXcell	123402	
Fluper	123410	
TechAhead	162347	
Fueled	234512	
OpenXcell	324411	
Fluper	345615	
Intellectsoft	420004	
TechAhead	456126	
OpenXcell	623453	
Fluper	663413	
TechAhead	663413	
Intellectsoft	723451	
OpenXcell	723451	
OpenXcell	773459	
Fueled	893401	
Intellectsoft	893412	
Fueled	956128	
OpenXcell	993403	

select * from app_desarrollada_por;

A	B	C
aplicacion_ID	trabajador_ID_2	
UBER_01	43006	
airbnb_AA	111104	
Spotify_01	123305	
wechat_alph	123400	
airbnb_AA	123402	
whatsapp_al	123410	
airbnb_AA	162347	
wechat_alph	234512	
Spotify_01	324411	
UBER_01	345615	
airbnb_AA	420004	
whatsapp_al	456126	
Spotify_01	623453	
airbnb_AA	663413	
airbnb_AA	723451	
Spotify_01	773459	
wechat_alph	893401	
Spotify_01	893412	
whatsapp_al	956128	
wechat_alph	993403	

select * from tienda_aplicaciones;

tienda_aplicacion_ID	administrador	direccion_web
App Store	Apple	https://www.apple.com/ios/app-store/
AppStore	Amazon	https://www.amazon.com/mobile-ap...
Google Play Store	Google Andorid	https://www.apple.com/ios/app-store/
Xiaomi App Store	Xiaomi	https://xiaomi-pedia.com/apps
NULL	NULL	NULL

select * from empresa_desarrolladora_apps;

nombre_empresa_ID	pais	anno_fundacion	e_mail	pagina_web
Fluper	India	2013	enquiry@fluper.com	https://www.fluper.com
Fueled	EE.UU/UK	2008	nyc@fueled.com	https://fueled.com
Intellectsoft	EE.UU.	2007	info@intellectsoft.net	https://www.intellectsoft.net/
OpenXcell	India	NULL	hr@openxcell.com	https://www.openxcell.com/
TechAhead	EE.UU/India	2009	sales@techaheadcorp.com	https://www.techaheadcorp.com
NULL	NULL	NULL	NULL	NULL

select * from trabajadores;

A	B	C	D	E	F	G
trabajador_ID	exp_profesional	direccion_calle	direccion_numero	codigo_postal	telefono	
43006	alta	Harvey Hayes street	37	26020	939678373	
111104	media	Bobby Garcia Avenue	75	26018	966328235	
123305	media	David Walton Avenue	998	26019	972691179	
123400	NULL	5th Avenue	121	26014	924514136	
123402	media	James Hansen street	77	26016	961825490	
123410	baja	Mark Perez Avenue	100	26010	911883966	
162347	media	Joan Trotter Avenue	4	26007	986301127	
234512	NULL	Mable Hughes Avenue	122	26002	964437347	
324411	alta	Amanda Balowski street	12	26011	934787678	
345615	NULL	Roger Lehman Avenue	727	26005	961521305	
420004	baja	213rd Avenue	1002	26004	NULL	
456126	alta	Brian Wells street	80	26006	949274407	
623453	baja	Paul Bowen street	50	26003	913786826	
663413	NULL	Juan Perez street	212	26013	937613218	
723451	alta	Jonathan Miller Avenue	11	26001	911773891	
773459	alta	Janine Tidwell square	46	26009	NULL	
893401	NULL	Dominic Delgado street	1	26015	918829793	
893412	alta	Danny Faust street	312	26012	945621065	
956128	media	412nd Avenue	33	26008	989855261	
993403	baja	Katharine Peterson street	144	26017	954543431	

select * from aplicacion;

A	B	C	D	E	F	G	H	I
aplicacion_ID	fecha_comienzo	fecha_fin	nombre_aplicacion	categoria	espacio_de_memoria	precio	trabajador_coordinador_ID	empresa_que_desarrollo_app_ID
airbnb_AA	2017-03-19	2019-12-21	airbnb	alquileres	32	0	420004	Intellectsoft
Spotify_01	2018-11-13	2019-09-26	Spotify	musica	50	0	893412	Fueled
UBER_01	2016-05-01	2019-07-06	uber	transportes	62	0	43006	Fluper
wechat_alpha	2016-10-18	2019-10-30	wechat	mensajeria	10	0	993403	OpenXcell
whatsapp_alpha	2016-02-04	2019-04-13	whatsapp	mensajeria	5	0	456126	TechAhead

select * from usuario;

A	B	C	D	E
cuenta_usuario	nombre_usuario	direccion	descargada_por_movil	numero_de_telefono
72	Concepcion Lewis	963, 39th Avenue, NY	Si	675351408
121	Elvia Brown	1567, 30th Avenue, NY	No	NULL
124	Samuel Roberts	1281, 20th Avenue, NY	No	NULL
256	Ying Hobbins	1099, 15th Avenue, NY	Si	636154822
312	John Vang	429, 7th Avenue, NY	Si	603832652
404	Alvin Baldwin	1604, 37th Avenue, NY	Si	697796511
500	Juan Elizarraras	1831, 21th Avenue, NY	Si	689542664
2218	Jim Partin	1759, 41th Avenue, NY	Si	661807953
7001	Gary Oestmann	1984, 33th Avenue, NY	Si	605406879
9001	Jeffrey Turner	919, 56th Avenue, NY	Si	668277074
10045	Pei Villagomez	847, 34th Avenue, NY	Si	610917317
12412	Mark Johnson	1042, 52th Avenue, NY	Si	617183074
20101	John Lyons	1565, 47th Avenue, NY	Si	666256422
41997	Mark Johnson	1555, 48th Avenue, NY	Si	619322935
45412	Timothy Pena	472, 14th Avenue, NY	Si	687220307
72651	Henry Byrne	1603, 15th Avenue, NY	Si	613721891
81237	Mark Johnson	1027, 13th Avenue, NY	Si	694025222
84359	Felicia Giraldo	1246, 63th Avenue, NY	Si	682815918
90477	Cortez Bartels	1505, 7th Avenue, NY	No	NULL
123777	Bruce Trujillo	1708, 32th Avenue, NY	Si	675076749
182979	Jennifer Hays	578, 15th Avenue, NY	Si	687509540
196712	Patricia Duckworth	1024, 58th Avenue, NY	No	NULL
197873	Kelly Hagen	1579, 33th Avenue, NY	Si	673208663
259159	Mark Johnson	220, 37th Avenue, NY	Si	633425707
313176	Ruth Winters	74, 51th Avenue, NY	Si	632071038
341218	Tekashi McDonald	313, 11th Avenue, NY	Si	617951989
393732	Shane Sanders	1104, 14th Avenue, NY	Si	608268009
5001245	Elaine Heckman	48, 11th Avenue, NY	Si	663282031

CONSULTAS:

-- 1) Número de personas que descargaron aplicación con móvil

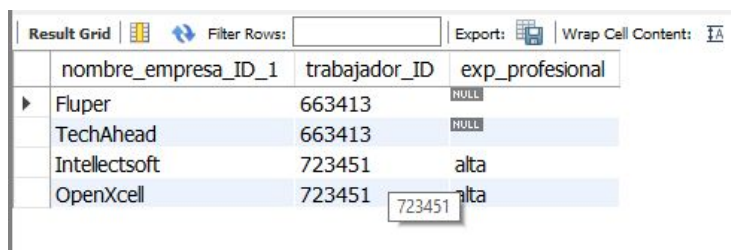
```
SELECT COUNT(*) FROM usuario where descargada_por_movil = 'Si';
```



COUNT(*)
24

-- 2) Nombre de la empresa, trabajador Id y experiencia profesional de los empleados que han
-- trabajado en más de una empresa desarrolladora de aplicaciones

```
SELECT nombre_empresa_ID_1, trabajador_ID, exp_profesional
FROM trabajadores as t
RIGHT JOIN ha_trabajado_en as h
ON t.trabajador_ID = h.trabajador_ID_1
where trabajador_ID_1
IN (SELECT trabajador_ID_1
FROM trabajadores as t
RIGHT JOIN ha_trabajado_en as h
ON t.trabajador_ID = h.trabajador_ID_1
group by trabajador_ID
having count(*) > 1);
```



nombre_empresa_ID_1	trabajador_ID	exp_profesional
Fluper	663413	NULL
TechAhead	663413	NULL
Intellectsoft	723451	alta
OpenXcell	723451	alta

-- 3) Pais donde más descargas se han realizado

```
select pais_donde_se_realizo_descarga, count(*) as frecuencia
from app_descargada_por
group by pais_donde_se_realizo_descarga
order by frecuencia DESC
```


limit 1;

Result Grid		Filter Rows:	Export:	Wrap Cell Content:	Fetc
	pais_donde_se_realizo_descarga	frecuencia			
▶	Rusia	9			

-- 4) Empresa desarrolladora de aplicaciones con oficina central en la India

select * from empresa_desarrolladora_apps where pais like '%ndia';

Result Grid		Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
	nombre_empresa_ID	pais	anno_fundacion	e_mail	pagina_web
▶	Fluper	India	2013	enquiry@fluper.com	https://www.fluper.com
	OpenXcell	India	NULL	hr@openxcell.com	https://www.openxcell.com/
	TechAhead	EE.UU/India	2009	sales@techaheadcorp.com	https://www.techaheadcorp.com
*	NULL	NULL	NULL	NULL	NULL

-- 5) Empresa desarrolladora con sucursales en más de un país.

select * from empresa_desarrolladora_apps where pais like '%/%';

Result Grid		Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
	nombre_empresa_ID	pais	anno_fundacion	e_mail	pagina_web
▶	Fueled	EE.UU/UK	2008	nyc@fueled.com	https://fueled.com
	TechAhead	EE.UU/India	2009	sales@techaheadcorp.com	https://www.techaheadcorp.com
*	NULL	NULL	NULL	NULL	NULL

-- 6) media de las puntuaciones de los usuarios que no pusieron comentario frente
-- a los que si.

```
select 'con_comentario' as 'tipo de usuario', avg(puntuacion)
from app_descargada_por where comentarios is not null
UNION
select 'sin_comentario', avg(puntuacion)
from app_descargada_por where comentarios is null;
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
tipo de usuario	avg(puntuacion)		
con_comentario	2.2500		
sin_comentario	3.5333		3.5333

-- 7) e-mail de empresas desarrolladoras de apps cuyas apps no están relacionadas con mensajería.

```
SELECT e_mail
FROM empresa_desarrolladora_apps as e
INNER JOIN aplicacion as a
ON e.nombre_empresa_ID = a.empresa_que_desarrollo_app_ID
where categoria != 'mensajeria';
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
e_mail			
info@intellectsoft.net			
nyc@fueled.com			
enquiry@fluper.com			

-- 8) ID de aplicación y fecha de aquellas que fueron comenzadas en el 2017

```
select aplicacion_ID, fecha_comienzo from aplicacion
where fecha_comienzo >= '2017-01-01' and fecha_comienzo <= '2017-12-31';
```

Result Grid	Filter Rows:	Edit:	Export/Import:
aplicacion_ID	fecha_comienzo		
airbnb_AA	2017-03-19		
NULL	NULL		

-- 9) usuarios que al menos han descargado 3 aplicaciones

```
select cuenta_usuario_1, count(*)
from app_descargada_por
group by cuenta_usuario_1
```

```
having count(*) >= 3
order by count(*) desc;
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
cuenta_usuario_1	count(*)		
393732	5		
182979	3		

```
-- 10) puntuacion media de la aplicacione desarrollada por la empresa donde
-- trabajan/han trabajado más empleados
```

```
select avg(puntuacion) from app_descargada_por
where aplicacion_ID_2
in (select aplicacion_ID from aplicacion, ha_trabajado_en
    where empresa_que_desarrollo_app_ID
    in (select nombre_empresa_ID_1 -- , count(*)
        from ha_trabajado_en
        group by nombre_empresa_ID_1
        having count(*) > 4));
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
avg(puntuacion)			
3.5714			

```
-- 11) código de los empleados (los cuales trabajan o han trabajado en empresa) y nombre
-- de la empresa que desarrolló la app mas descargada en India.
```

```
select empresa_que_desarrollo_app_ID as
primero_nombre_empresa_luego_IDs_de_sus_empleados
from aplicacion
where aplicacion_ID
in (select aplicacion_ID_2 from app_descargada_por
    where pais_donde_se_realizo_descarga = 'India'
    group by aplicacion_ID_2
    having count(*) > 2)
union
select trabajador_ID_1 from ha_trabajado_en
```

```

where nombre_empresa_ID_1
in (select empresa_que_desarrollo_app_ID from aplicacion
    where aplicacion_ID
    in (select aplicacion_ID_2 from app_descargada_por
        where pais_donde_se_realizo_descarga = 'India'
        group by aplicacion_ID_2
        having count(*) > 2));

```

-- DETALLE: 'limit 1' no es una válida subconsulta en esta versión de mySQL. Si no, en vez de usar 'having count(*) > 2))' ordenaría de mayor a menor y usaría 'limit 1' para seleccionar el máximo

primero_nombre_empresa_luego_IDs_de_sus_empleados	
OpenXcell	
123402	
324411	
623453	
723451	
773459	
993403	

-- 12) Agrupar por países a los usuarios que han descargado la app que lleva más tiempo
-- en el mercado

```

select aplicacion_ID_2, pais_donde_se_realizo_descarga,
count(*) as numero_de_usuarios
from app_descargada_por
where aplicacion_ID_2
in (select aplicacion_ID from aplicacion
    where fecha_fin in (select min(fecha_fin) from aplicacion))
group by pais_donde_se_realizo_descarga
order by numero_de_usuarios desc;

```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
aplicacion_ID_2	pais_donde_se_realizo_descarga	numero_de_usuarios	
whatsapp_alpha	Rusia	2	
whatsapp_alpha	China	2	
whatsapp_alpha	Iran	1	
whatsapp_alpha	Gabon	1	
whatsapp_alpha	Vaticano	1	

-- 13) ID, direccion y telefono de los trabajadores de los que no se conoce su
-- experiecia profesional

```
select trabajador_ID, direccion_calle, direccion_numero, codigo_postal, telefono
from trabajadores
where exp_profesional is null;
```

Result Grid	Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
trabajador_ID	direccion_calle	direccion_numero	codigo_postal	telefono
123400	5th Avenue	121	26014	924514136
234512	Mable Hughes Avenue	122	26002	964437347
345615	Roger Lehman Avenue	727	26005	961521305
663413	Juan Perez street	212	26013	937613218
893401	Dominic Delgado street	1	26015	918829793
NULL	NULL	NULL	NULL	NULL

-- 14) Devolver el nombre de todas las empresas desarrolladoras que no hayan tenido puntuación
-- uno por parte de los usuarios.

```
select empresa_que_desarrollo_app_ID from aplicacion
where aplicacion_ID
not in (select distinct aplicacion_ID_2 from app_descargada_por where puntuacion = 1);
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
empresa_que_desarrollo_app_ID			
Fluper			
OpenXcell			
TechAhead			

-- 15) ID de empleados que han desarrollado la aplicación que en mas tiendas

-- se encuentra

```
select trabajador_ID_1 from ha_trabajado_en
where nombre_empresa_ID_1
in (select empresa_que_desarrollo_app_ID from aplicacion
    where aplicacion_ID
    in (select aplicacion_ID_1 from app_se_encuentra_en
        group by aplicacion_ID_1
        having count(*) > 3));
```



The screenshot shows a database query result grid. At the top, there is a toolbar with options like 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'. Below the toolbar, the first column is labeled 'trabajador_ID_1'. The data rows contain the following values: 123305, 162347, 456126, and 663413. The rows with 162347 and 663413 are highlighted in blue.

trabajador_ID_1
123305
162347
456126
663413

5.- Conclusiones:

- SQL fue creado para almacenar bases de datos de manera eficiente.
- Las relaciones nos permiten conectar dos entidades sin haber tenido que fusionar todos los campos de las tablas. De esta manera se evita redundancias y duplicidades al hacer modificaciones en algún campo, además de ahorrar mucha memoria.
- Dependiendo de las particularidades de la relación, la forma de definir dicha relación en SQL será diferente. Por ejemplo, la relación 'app_coordinada_por' es de tipo 1:N y se define propagando la clave primaria de 'trabajadores' en 'aplicacion'. Sin embargo, 'app_se_encuentra_en' es de tipo N:N y hay que crear una nueva tabla con las claves primarias de las dos entidades que componen la relación.
- Para entender qué conexión tienen las tablas de una misma base de datos, es especialmente útil usar el diagrama de Entidad-Relación.
- SQL dispone de una gran variedad de comandos y operadores para hacer consultas. Sin embargo, el uso de subconsultas fue especialmente útil para aquellas consultas que requerían usar más de una tabla.

Este es el final. ¡Muchas gracias!