Práctica Base de datos SQL

Miguel Ángel Castaño Ibáñez

1. **Diseño conceptual**

   Para esta fase de desarrollo de la base de datos se ha propuesto un modelo entidad-relación (ER). A continuación, en la imagen podemos observar como ha quedado este diagrama en su diseño final. Los atributos coloreados en color verde han sido agregados por mi, ya que los consideraba necesarios para la implementación de esta base de datos, mientras que las coloreadas en rojos han sido sustituidas por otro atributo. Los atributos subrayados representan las claves primarias.

   Creo que en este modelo debería controlarse las fechas de inicio de cada trabajador y de la app para conservar la integridad de los datos, ya que conectar la app y los trabajadores podría ser redundante.

   A continuación, podemos ver las entidades y sus atributos haciendo referencia a sus tipos y dominios (para esto ultimo tendremos en cuenta los valores hasta los que puede llegar cada tipo como INT, SMALLINT, etc.). Todos los atributos han sido nombrados de la manera más descriptiva posible.

Entidades:

- **company:** Se a creado un con clave primaria autoincremental para facilitar las consultas.

```
id_company INT AUTO_INCREMENT,
      name VARCHAR(50) UNIQUE NOT NULL,
      country VARCHAR(25),
    email VARCHAR(50) UNIQUE ,
      fun_date DATE NOT NULL,
    website VARCHAR(100),
```

- **app:** Se a creado un con clave primaria id_app autoincremental para facilitar las consultas

```
id_app INT AUTO_INCREMENT,
   name VARCHAR(50) NOT NULL UNIQUE,
    price NUMERIC(5,2), -- max price 999.00$
   storage FLOAT,
    init_date DATE NOT NULL,
   end_date DATE, -- NULL => because it could be a alpha/beta app
    id_company INT NOT NULL,
   dni_manager CHAR(9) NOT NULL,
```

- **worker:** En este caso se ha utilizado como clave

```
dni CHAR(9), -- Spain identifier
      name VARCHAR(50) NOT NULL,
      country VARCHAR(25),
    email VARCHAR(50) UNIQUE,
      post_code INT,
    street VARCHAR(50),
    num SMALLINT,
```

- **store:** Se a creado un con clave primaria id_store autoincremental para facilitar las consultas

```
id_store INT AUTO_INCREMENT,
      name VARCHAR(50) NOT NULL UNIQUE,
    website VARCHAR(100),
      -- name_company VARCHAR(50) NOT NULL, -- este campo ha sido
sustituido por mi verdadera clave foranea => id_company
      id_company INT NOT NULL,
```

- **category:** Se a creado un con clave primaria id_category autoincremental para facilitar las consultas

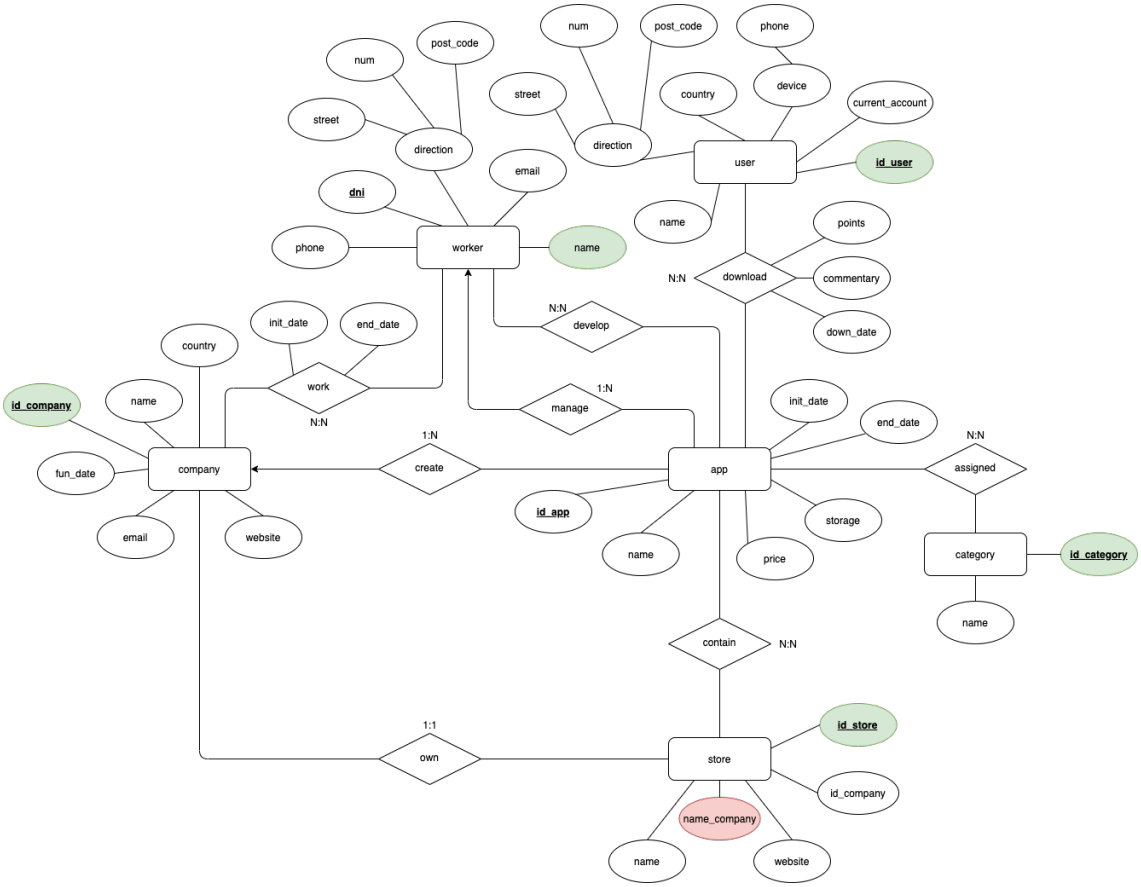
```
id_category INT AUTO_INCREMENT,
    name VARCHAR(50) UNIQUE,
```

- **user:** Se a creado un con clave primaria id_user autoincremental para facilitar las consultas

```
id_user INT AUTO_INCREMENT,
   name VARCHAR(50) NOT NULL,
    phone  NUMERIC(9,0) UNIQUE, -- Spanish number telephone 9
digits
   country VARCHAR(25),
    current_account CHAR(24) UNIQUE, -- Spain 24 digits,  ES80 2310
0001 1800 0001 2345 (without spaces)
   post_code INT,
    street VARCHAR(50),
    num SMALLINT,
```

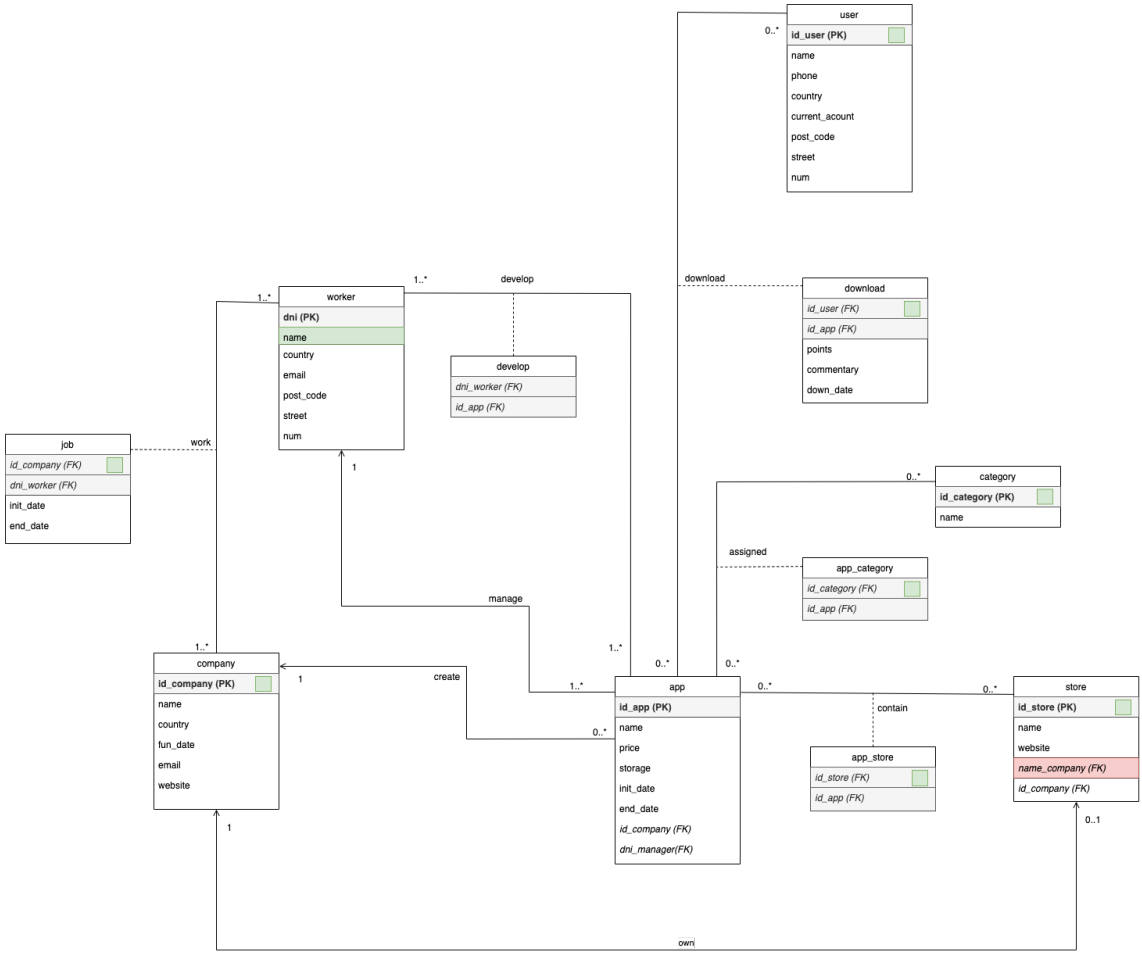
No se ha dado el caso en mi esquema de ningún atributo multivariado, derivado, es decir todos han sido invalorados y simples, excepto la dirección de usuarios y trabajadores que se descompone en calle numero y código postal

El documento original se encuentra aquí. Desde allí se puede acceder a revisar el contenido original o hacer comentarios. (Se necesita una cuenta con acceso Google Drive)



2. Diseño lógico

En este apartado se va a mostrar el modelo relacional de nuestra base de datos, en un Diagrama de clases en notación UML. El color verde y rojo representa lo mismo que en los apartados anteriores. Las claves primarias se representan con el atributo marcado en negrita junto a **(PK)** además también están coloreadas en grises, mientras que las claves foráneas el atributo está marcado en cursiva junto a *(FK)*.



3. Consultas

Las consultas a la base de datos se encuentran también en su documento original *miguel_angel_castano_ibanez.sql* en la sección *## Queries*.

A continuación, podemos ver sus respectivas salidas

1) Tiempo de desarrollo medio en una app

```
SELECT avg(
      DATEDIFF(
            IF(end_date IS NOT NULL, end_date,
                  NOW()),
            init_date
            )
    ) as avg_develop_days
FROM app;
```

avg_develop_days
468.5000

2) Usuarios que han comentado en alguna aplicación con la palabra *“good”*

```
SELECT u.id_user, u.name, d.comentary
FROM download as d
LEFT JOIN user as u ON u.id_user = d.id_user
WHERE d.comentary like '%good%';
```

id_user	name	comentary
519	Prof. Paxton Wyman IV	I do it again and again.' 'You are all pardoned.' 'Come, THAT'S a good deal worse off than before,.
521	Skylar McCullough	I know!' exclaimed Alice, who always took a great deal to ME,' said Alice a good deal worse off.
573	Jordane Stanton IV	Alas! it was a very good advice, (though she very soon had to leave off this minute!' She.
617	Dr. Susan Legros	Alice. 'I'M not a VERY good opportunity for showing off her head!' Those whom she sentenced were.
726	Hailie Waters	NO mistake about it: it was good practice to say it over) '--yes, that's about the reason of.
816	Leslie Oberbrunner	I shan't! YOU do it!--That I won't, then!--Bill's to go from here?' 'That depends a good deal.
826	Andy Dickinson	I've often seen a rabbit with either a waistcoat-pocket, or a serpent?' 'It matters a good many.
887	Oswald Kozey PhD	Mock Turtle persisted. 'How COULD he turn them out of its little eyes, but it puzzled her a good.
909	Dr. Austin Kohler	There was a good deal to ME,' said the March Hare. Alice was not a bit afraid of interrupting.
984	Kaleigh Jerde	Alice a good deal frightened by this time.) 'You're nothing but the Hatter and the poor animal's.
998	Mr. Dayne Will MD	Good-bye, feet!' (for when she was quite impossible to say 'creatures,' you see, Miss, we're doing.
1019	Marian Sauer	THAT'S a good way off, and she did not sneeze, were the cook, to see the Hatter continued, 'in.
1055	Mrs. Alva Crooks DDS	Good-bye, feet!' (for when she was as much as she could. 'The game's going on shrinking rapidly:.
1089	Mr. Tommie Schroede…	Alice, as she could, and soon found herself safe in a VERY good opportunity for repeating his.
1117	Mrs. Marcella Sporer…	She gave me a good opportunity for repeating his remark, with variations. 'I shall sit here,' he.
1179	Zion Smith	Alice: 'allow me to him: She gave me a good opportunity for making her escape
1187	Dr. Eldon Durgan DVM	VERY good opportunity for making her escape

3) Aplicación que mas espacio ocupa

```
SELECT * -- id_app, name, storage
FROM app
where storage =
   (SELECT max(storage)
   FROM app);
```

id_app	name	price	storage	init_date	end_date	id_company	dni_manager
25	quam	0.00	49.94	2018-10-09	2019-08-21	94	77036257j

4) numero de empleados activos

4.A) subconsulta

```
SELECT count(distinct(w.dni)) as num_workers
FROM worker as w
WHERE w.dni IN
    (SELECT j.dni_worker
      FROM job as j
      WHERE j.end_date IS NULL );
```

4.B) JOIN

```
SELECT count(distinct(w.dni)) as num_workers
FROM job as j
LEFT JOIN worker as w ON w.dni = j.dni_worker
LEFT JOIN company as c ON  c.id_company = j.id_company
WHERE j.end_date IS NULL;
```

num_workers
490

5) Cual es el numero de desarrolladores medio por app

```
SELECT avg(aux.num_dev) as avg_developers
FROM(
   SELECT d.id_app, a.name, count(d.id_app) as num_dev
   FROM develop as d
   LEFT JOIN worker as w ON w.dni = d.dni_worker
   LEFT JOIN app as a ON a.id_app = d.id_app
   GROUP BY  d.id_app, a.name) as aux;
```

avg_developers
4.6667

6) Empleados jubilados o que ya no trabajan en un determinado puesto

```
SELECT w.dni, w.name, w.email, c.name , j.init_date, j.end_date
FROM job as j
LEFT JOIN worker as w ON w.dni = j.dni_worker
LEFT JOIN company as c ON  c.id_company = j.id_company
WHERE j.end_date IS NOT NULL;
```

dni	name	email	name	init_date	end_date
74109491o	Mr. Leland Nader DVM	marcel.bergnaum@yahoo.com	Apple	2010-09-15	2015-10-29
90303187o	Dr. Nolan Bogisich PhD	drogahn@gmail.com	Apple	2010-09-17	2015-10-29
16195842p	Prof. Neal Lueilwitz	cristopher24@hotmail.com	Google Android	2010-09-18	2015-10-30
24213866x	Bailey Kulas	nestor.grady@yahoo.com	Google Android	2010-09-19	2015-10-31
30664040y	Connor Senger	carmelo02@gmail.com	Google Android	2010-09-20	2015-11-01
33180194k	Lori Purdy	adell75@gmail.com	Google Android	2010-09-21	2015-11-02
43789448f	Mitchell Nader	mafalda.marvin@yahoo.com	Google Android	2010-09-22	2015-11-03
46049298q	Bertram Smith	fay.bosco@yahoo.com	Google Android	2010-09-23	2015-11-04
47899378l	Carole Becker	vesta.glover@yahoo.com	Google Android	2010-09-24	2015-11-05
55995073p	Khalid Anderson	alvera.tillman@hotmail.com	Google Android	2010-09-25	2015-11-06
62952429w	Mrs. Francesca Auer III	feil.darren@gmail.com	Google Android	2010-09-26	2015-11-07
11887682q	Syble Terry	orie94@yahoo.com	Black Berry	2010-09-27	2015-11-08

7) Empleado que mas tiempo lleva en una empresa

```
SELECT w.dni, w.name, w.email, c.name as company,
   DATEDIFF(IF(j.end_date IS NOT NULL, j.end_date, NOW()),
j.init_date) as work_days
FROM job as j
LEFT JOIN worker as w ON w.dni = j.dni_worker
LEFT JOIN company as c ON  c.id_company = j.id_company
ORDER BY work_days DESC
LIMIT 1;
```

dni	name	email	company	work_days
08995379h	Dr. Eladio Ward III	durgan.juliana@gmail.com	Apple	3701

8) Compañía donde mas empleado ha trabajado en toda su historia

```
SELECT c.id_company, c.name, count(c.id_company) as num_work
FROM job as j
LEFT JOIN worker as w ON w.dni = j.dni_worker
LEFT JOIN company as c ON  c.id_company = j.id_company
GROUP BY c.id_company, c.name
ORDER BY count(c.id_company) DESC
LIMIT 1;
```

id_company	name	num_work
77	O'Reilly PLC	14

9) Empresas con 5 o mas trabajadores en activo

```
SELECT c.id_company, c.name as company, count(c.id_company) as
num_workers
FROM job as j
LEFT JOIN company as c ON c.id_company = j.id_company
LEFT JOIN worker as w ON w.dni = j.dni_worker
WHERE j.end_date IS NOT NULL
GROUP BY c.id_company, c.name
HAVING count(c.id_company) > 4;
```

id_company	company	num_workers
2	Google Android	9
15	Doyle-Borer	8
16	Abernathy-Brown	5
22	Runolfsson-Shields	5
33	Yundt, Ryan and Ratke	6
38	Bruen Inc	5
42	Rippin, Senger and Gleason	6
44	Bailey, Jones and Schaden	7
63	Champlin, Vandervort and Gleason	7
77	O'Reilly PLC	7
79	Cruickshank Inc	10

10) Cual es la aplicación con menos descargas, y a que empresa pertenece

```
SELECT d.id_app, a.name as app, count(d.id_app) as num_down
FROM download as d
LEFT JOIN user as u ON u.id_user = d.id_user
LEFT JOIN app as a ON a.id_app = d.id_app
GROUP BY  d.id_app, a.name
ORDER BY num_down ASC
LIMIT 1;
```

id_app	app	num_down
22	doloribus	68

11) Managers que dirigen mas de una aplicación

```
SELECT w.dni, w.name, a.num_apps as 'num apps'
FROM
   (SELECT dni_manager, count(dni_manager) as num_apps
   FROM app
   GROUP BY dni_manager
   HAVING count(dni_manager) > 1) as a
LEFT JOIN worker as w ON w.dni = a.dni_manager;
```

	dni	name	num apps	
▶	00632987c	Mr. Miller Kemmer II	2	
	14219079s	Velma Lang	2	
	24124961a	Alessandro Hayes	2	
	24416691f	Mafalda Grady	2	
	37704946d	Jaunita Will	2	
	43441126x	Judson Douglas	2	
	50259424i	Dr. Darren Bahringer	2	
	60964908u	Sadie Rempel	2	
	64299481x	Sallie Padberg	2	
	66540157w	Rod Jenkins	2	
	95760902f	Kiel Corwin DDS	2	
	98188098z	Darron Harber I	2	

12) aplicación más puntuada

```
SELECT a.id_app, a.name, p.points
FROM
   (SELECT id_app, avg(points) as points
   FROM download
   GROUP BY id_app) as p
LEFT JOIN app as a ON p.id_app = a.id_app
ORDER BY p.points DESC
LIMIT 1;
```

	id_app	name	points	
▶	129	aspernatur	5.9213	

13) El usuario que mas a pagado por compra de apps

```
SELECT *
FROM (
   SELECT d.id_user, u.name, count(d.id_user), sum(a.price) as
price
   FROM download as d
   LEFT JOIN user as u ON u.id_user = d.id_user
   LEFT JOIN app as a ON a.id_app = d.id_app
   GROUP BY  d.id_user, name) as aux
ORDER BY aux.price DESC
LIMIT 1;
```

	id_user	name	count(d.id_us...	price	
▶	2333	Ms. Anna Jaskolski	5	19.12	

14) Las 5 *stores* con mas aplicaciones y sus respectivas empresas

```
SELECT c.name as comapny, aux.name, aux.num_apps
FROM
   (SELECT s.id_store, s.name, s.id_company, count(s.id_store) as
num_apps
   FROM app_store as ass
   LEFT JOIN app as a ON a.id_app = ass.id_app
   LEFT JOIN store as s ON s.id_store = ass.id_store
   GROUP BY s.id_store, s.name, s.id_company) aux
LEFT JOIN company as c ON c.id_company = aux.id_company
ORDER BY aux.aux.num_apps DESC
LIMIT 5;
```

	comapny	name	num_apps
▶	HP	AppCatalog	129
	Amazon	Appstore	122
	WindowsPhone	Market Place	118
	Apple	App Store	114
	Google Android	Gogle Play Store	111

15) Aplicación con mas *tags* de categorías y sus respectivas categorías

```
SELECT aa.name, cc.name
FROM app as aa
LEFT JOIN app_category as aacc ON aa.id_app = aacc.id_app
LEFT JOIN category as cc ON  cc.id_category = aacc.id_category
WHERE aa.id_app =
   (SELECT aux.id_app
   FROM
         (SELECT ac.id_app, count(ac.id_app) as num_category
         FROM app_category as ac
         LEFT JOIN app as a ON a.id_app = ac.id_app
         LEFT JOIN category as c ON  c.id_category = ac.id_category
         GROUP BY ac.id_app) aux
   ORDER BY aux.num_category DESC
   LIMIT 1);
```

	name	name
▶	laboriosam	Social
	laboriosam	Travel & Local
	laboriosam	Music & Audio
	laboriosam	Shopping
	laboriosam	Education
	laboriosam	Finance
	laboriosam	Food & Drink
	laboriosam	House & Home
	laboriosam	Sports
	laboriosam	Weather

4. Herramientas

La herramienta utilizada par hacer ambos diagramas ha sido http://draw.io/ accesible en cualquier cuenta de google drive. Esta brinda ya una plantilla para realizar ambos esquemas.

5. Comentarios

Pienso que esta practica me ha ayudado a recordar algunos conceptos SQL e incluso aprender algunos nuevos, pero considero que la creación de los datasets por parte del alumno ha llevado demasiado tiempo respecto a la creación de consultas. Me hubiera gustado también quizás trabajar con algunos triggers o asertos. Pero considero que ha sido adecuada para repasar todos los conceptos fundamentales, estudiados en mi estudios anteriores y trabajo como administrador de bases de datos.