Memoria práctica 1 Administración y gestión de bases de datos

https://github.com/Javier5Albatros/ABD

Realizado por:
Ismael Carrasco Lago
Marcos Icardo Chicote
Javier Álvarez Losada

ÍNDICE

Tabla de contenido

1. CREACIÓN Y CARGA DE LA BASE DE DATOS

1.1 Creación base de datos

Primero, creamos las sentencias SQL que nos permitían crear y destruir la Base de Datos.

```
CREATE DATABASE PracABD1;
DROP DATABASE PracABD1;
```

1.2 Creación Datafiles y Tablespaces

Seguidamente generamos las sentencias de creación y eliminación de Datafiles y Tablespaces, para las cuales tuvimos que explorar la documentación oficial.

```
CREATE TABLESPACE `ABDDBA_TBLS_cursos` ADD DATAFILE
'ABDDBA_DF_cursos.ibd' ENGINE=INNODB;
CREATE TABLESPACE `ABDDBA_TBLS_personas` ADD DATAFILE
'ABDDBA_DF_personas.ibd' ENGINE=INNODB;
CREATE TABLESPACE `ABDDBA_TBLS_matriculados` ADD DATAFILE
'ABDDBA_DF_matriculados.ibd' ENGINE=INNODB;

DROP TABLESPACE ABDDBA_TBLS_cursos;
DROP TABLESPACE ABDDBA_TBLS_personas;
DROP TABLESPACE ABDDBA_TBLS_matriculados;
```

1.3 Creación de las tablas

Una vez tenemos todo, procedimos a crear las tablas:

Tabla personas

```
CREATE TABLE IF NOT EXISTS `PracABD1`.`personas` (
  `PersonaID` INT NOT NULL,
  `Dni` VARCHAR(9) UNIQUE NOT NULL,
  `Nombre` VARCHAR(20) NOT NULL,
  `Apellidos` VARCHAR(30) NOT NULL,
  `Genero` VARCHAR(1) NULL,
  `Direccion` VARCHAR(60) NULL,
```

```
  `Localidad` VARCHAR(50) NULL,
  `Provincia` VARCHAR(30) NULL,
  `CodPostal` INT NULL,
  `Telefono` VARCHAR(9) NULL,
  `EnParo` BOOLEAN NULL,
  `Canal` INT NULL,
  `FechaNac` DATE NULL,
  `Email` VARCHAR(60) NULL,
  CONSTRAINT CHECK (Genero IN ('H', 'M')),
  CONSTRAINT CHECK (EnParo IN (0, 1)),
  CONSTRAINT CHECK (Canal IN (0, 1, 2, 3, 4)),
  PRIMARY KEY (`PersonaID`),
  UNIQUE INDEX `PersonaID_UNIQUE` (`PersonaID` ASC) VISIBLE)
  TABLESPACE ABDDBA_TBLS_personas
ENGINE = InnoDB;
```

En esta tabla definimos los constraints para los campos de Género, EnParo y Canal, y creamos el índice para la Id.

Tabla cursos

```
CREATE TABLE IF NOT EXISTS `PracABD1`.`cursos` (
  `CursoID` INT NOT NULL,
  `nombre` VARCHAR(15) NOT NULL,
  `area` VARCHAR(30) NULL,
  `edicion` INT NOT NULL,
  PRIMARY KEY (`CursoID`),
  CONSTRAINT CHECK (edicion BETWEEN 2013 AND 2020 OR edicion LIKE
1492),
  UNIQUE INDEX `CursoID_UNIQUE` (`CursoID` ASC) VISIBLE)
  TABLESPACE ABDDBA_TBLS_cursos
ENGINE = InnoDB;
```

Aquí también añadimos el índice de la clave primaria, y el constraint para la Edición, añadiendo la opción de 1492 como año no válido dentro del rango.

Tabla matriculados_interesados

```
CREATE TABLE IF NOT EXISTS `PracABD1`.`matriculados_interesados` (
  `PersonaID` INT NOT NULL,
  `CursoID` INT NOT NULL,
  `matriculado` TINYINT NULL,
  `comentarios` VARCHAR(500) NULL,
  PRIMARY KEY (`PersonaID`, `CursoID`),
```

```
  CONSTRAINT CHECK (matriculado LIKE 1 OR NULL),
  FOREIGN KEY (PersonaID) REFERENCES personas(PersonaID),
  FOREIGN KEY (CursoID) REFERENCES cursos(CursoID))
  TABLESPACE ABDDBA_TBLS_matriculados
ENGINE = InnoDB;
```

En esta tabla añadimos las claves foráneas que referencian las dos claves anteriores, y añadimos dicho par como clave primaria de la tabla. Además, comprobamos que el campo matriculado es 1 (que está matriculado) NULL (simplemente está interesado)

Y el script para eliminar las tres tablas

```
DROP TABLE `PracABD1`.`personas`;
DROP TABLE `PracABD1`.`cursos`;
DROP TABLE `PracABD1`.`matriculados_interesados`;
```

1.4 Ejecución de los scripts

Ejecutamos todos los scripts anteriores para tener lista la infraestructura, creando la Base de Datos, los Datafiles y Tablespaces, y las tablas.

1.5 Inserción de los datos

Esta ha sido la parte más tediosa y que más problemas nos ha dado de todo el proceso.De primeras, intentamos cargar todos los datos mediante MySQL Workbench, pero tardaba mucho, llegando a estar más de 2 horas para insertar una tabla con 10.000 registros, por lo que optamos por insertarlo mediante consola. Para poder conectarnos e insertar correctamente los datos, tuvimos que usar este comando: **mysql --local-infile=1 -u root -p** También, usamos como encoding latin1.

Para pasar los datos de formato Excel a formato csv, seguimos los siguientes pasos: Archivo -> Exportar -> Cambiar el tipo de archivo -> CSV

A continuación, detallamos el proceso seguido con cada tabla para la inserción.

Tabla personas

En el campo EnParo, pusimos a 0 los valores nulos, los cuales indican que la persona no está en paro. Todo mediante excel: Seleccionar columna -> Ordenar y Filtrar -> Buscar y reemplazar.

Para los campos Canal y CodPostal repetimos el mismo procedimiento para los campos nulos.

Con el email, hicimos lo mismo, reemplazando las vocales acentuadas por su correspondiente vocal sin acento, y las ñ por n.

Para la fecha de nacimiento, usamos la función de MySQL **STR_TO_DATE** para pasarlo al formato DATE que nos interesaba.

```
LOAD DATA LOCAL INFILE 'path/to/Personas.csv' INTO TABLE personas
CHARACTER SET latin1
FIELDS TERMINATED BY ';'
LINES TERMINATED BY '\n'
(PersonaID, DNI, nombre, apellidos, genero, direccion, localidad,
provincia, codPostal, telefono, EnParo, canal, @fechaProvisional,
email)
SET FechaNac = STR_TO_DATE(@fechaProvisional, '%d/%m/%Y');
```

Tabla Cursos

Para el campo edición, para filtrar valores válidos y dar el valor 1492 a los inválidos, utilizamos la siguiente fórmula en el excel: **=SI(O(E1<2013;E1>2020);1492;E1)**. Copiamos la columna D a la E. Posteriormente introducimos la fórmula anterior en D, y a continuación copiamos D para pegarla en D otra vez, pero en esta ocasión ,seleccionando la opción de pegado por Valor. Una vez hecho eso, eliminamos la columna E, y exportamos a CSV.

```
LOAD DATA LOCAL INFILE 'path/to/Cursos.csv' INTO TABLE cursos
CHARACTER SET latin1
FIELDS TERMINATED BY ';'
LINES TERMINATED BY '\n';
```

Tabla Matriculados Interesados

En esta tabla, pusimos a 0 los valores nulos del campo Matriculado. Luego, para eliminar valores duplicados, en Excel seleccionamos las dos columnas de claves (CursoID y PersonaID), y fuimos a **Data -> Data Tools -> Remove Duplicates**. Con eso la tabla quedó lista para insertar.

```
LOAD DATA LOCAL INFILE 'path/to/Matriculados_Interesados.csv' INTO
TABLE matriculados_interesados
CHARACTER SET latin1
FIELDS TERMINATED BY ';'
LINES TERMINATED BY '\n';
```

1.6 Creación y eliminación de Claves Primarias y Foráneas

Para la creación de las claves primarias usamos estas sentencias

```
ALTER TABLE personas ADD PRIMARY KEY(PersonaID);
ALTER TABLE cursos ADD PRIMARY KEY(CursoID);
ALTER TABLE matriculados_interesados ADD PRIMARY KEY(PersonaID,
CursoID);
```

Y para las claves foráneas

```
ALTER TABLE matriculados_interesados
ADD CONSTRAINT cursos_fk
FOREIGN KEY (CursoID)
REFERENCES cursos(CursoID);

ALTER TABLE matriculados_interesados
ADD CONSTRAINT personas_fk
FOREIGN KEY (PersonaID)
REFERENCES personas(PersonaID);
```

Para eliminar las claves primarias usamos las siguientes sentencias

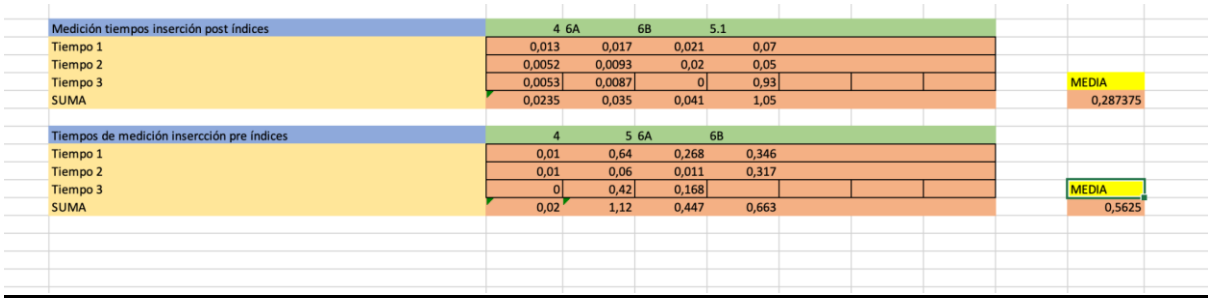
```
ALTER TABLE personas DROP PRIMARY KEY;
ALTER TABLE cursos DROP PRIMARY KEY;
ALTER TABLE matriculados_interesados DROP PRIMARY KEY;
```

Y para eliminar las foráneas,

```
ALTER TABLE matriculados_interesados DROP CONSTRAINT cursos_fk;
ALTER TABLE matriculados_interesados DROP CONSTRAINT personas_fk;
```

1.7 Medición de tiempos y justificación: ¿Índices antes o después de inserción de datos?

Procedemos a ejecutar los scripts 4,5, 6A y 6B. En la primera tabla se crean primero los índices y posteriormente se introducen los datos. En la segunda tabla, se introducen los datos primero y posteriormente se crean los índices. Los tiempos recogidos son los siguientes:



Medición tiempos inserción post índices	4	6A	6B	5.1				
Tiempo 1	0,013	0,017	0,021	0,07				
Tiempo 2	0,0052	0,0093	0,02	0,05				
Tiempo 3	0,0053	0,0087	0	0,93				MEDIA
SUMA	0,0235	0,035	0,041	1,05				0,287375

Tiempos de medición inserción pre índices	4	5	6A	6B				
Tiempo 1	0,01	0,64	0,268	0,346				
Tiempo 2	0,01	0,06	0,011	0,317				
Tiempo 3	0	0,42	0,168					MEDIA
SUMA	0,02	1,12	0,447	0,663				0,5625

Como se puede observar, si se crean los índices previamente a la inserción de datos obtenemos un tiempo medio de todos los scripts de aproximadamente 0,29s siendo la carga de datos de alrededor de 1,05s. En cambio, si se insertan los datos previamente a la creación de índices, obtenemos un tiempo medio de aproximadamente 0,56s, un tiempo prácticamente el doble que en el caso anterior, siendo la carga de datos de 1,12s, 0,07s más lenta que en el otro caso. Se observa una mejora en todas los scripts de inserción de datos creando previamente los índices a excepción de la tabla matriculados_interesados en la cual va cerca de 0,5s más lenta.
Basándonos en los datos obtenidos, sería más apropiado crear los índices previamente a la inserción de datos para obtener una mejor eficiencia.

2. Gestión de índices

2.1 Implementación de las sentencias sql más frecuentes

A. Obtener el email de un potencial alumno

Para obtener el correo de un alumno filtraremos por nombre y apellidos mediante la cláusula where.

```
SELECT DISTINCT email
FROM personas
WHERE nombre LIKE 'Ángeles'
AND apellidos LIKE 'del Nido';
```

Resultado:

```
+-----------------------------+
| email                       |
+-----------------------------+
 |AngelesdelNido1@gmail.com
+-----------------------------+
```

B. Seleccionar el email de las personas en paro

Como las personas en paro son aquellas cuyo valor del atributo enParo es igual a 0 usaremos de nuevo la sentencia where para filtrar.

```
SELECT DISTINCT email
FROM personas
WHERE enParo = 0;
```

Resultado:

email
YolandaSagrarioHarinero3@gmail.com
YolandaPizarroFraile@gmail.com
YolandaPardoPortillo4@yahoo.com
YolandaMenaValenciano4@gmail.com
YolandaLazaroMuelas2@gmail.com
YolandaGuijarroVicente1@yahoo.com
VictorTrigueroHernando@gmail.com
VictorTorcazCampillo3@gmail.com
VictorIglesiasMayo1@gmail.com

C. Obtener el número de alumnos de una determinada provincia

Haremos uso de la sentencia count filtrando con where por provincia.

```
SELECT COUNT(*)
FROM personas
WHERE provincia LIKE 'Lugo';
```

Resultado:

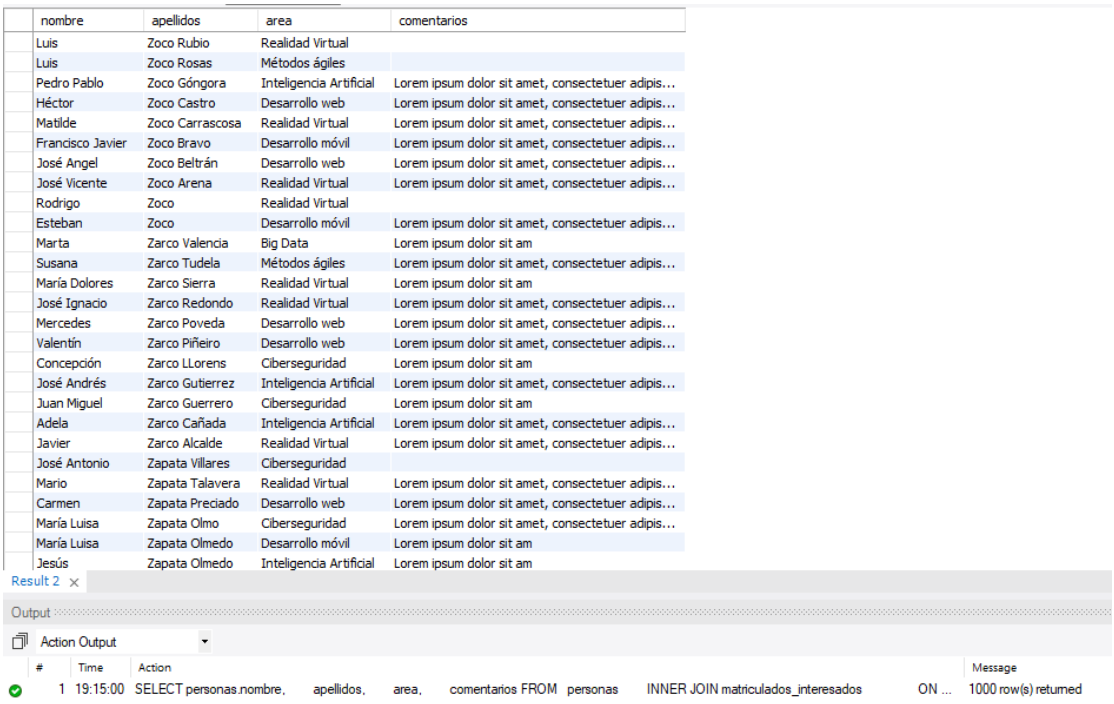
```
+----------+
| COUNT(*) |
+----------+
|       20 |
+----------+
1 row in set (0,04 sec)
```

D. Obtener nombre completo, curso matriculado, área del curso y sus comentarios. Deben ser de una determinada provincia y año. Ordenar por apellidos.

En este caso, usaremos un inner join para juntar las dos tablas mediante la clave foránea de personaId en matriculados_interesados y posteriormente con where filtraremos como hemos hecho en las queries anteriores. Finalmente para ordenar por apellidos usaremos la sentencia “ORDER BY” apellidos desc y se mostrarán de la z en adelante.

```
SELECT personas.nombre,
       apellidos,
       area,
       comentarios
FROM   personas
       INNER JOIN matriculados_interesados
               ON matriculados_interesados.personaid =
personas.personaid
       INNER JOIN cursos
               ON cursos.cursoid =
matriculados_interesados.cursoid
WHERE  provincia LIKE 'Madrid'
       AND edicion = 2018
ORDER  BY apellidos DESC
```

Resultado:



nombre	apellidos	area	comentarios
Luis	Zoco Rubio	Realidad Virtual	
Luis	Zoco Rosas	Métodos ágiles	
Pedro Pablo	Zoco Góngora	Inteligencia Artificial	Lorem ipsum dolor sit amet, consectetuer adipis...
Héctor	Zoco Castro	Desarrollo web	Lorem ipsum dolor sit amet, consectetuer adipis...
Matilde	Zoco Carrascosa	Realidad Virtual	Lorem ipsum dolor sit amet, consectetuer adipis...
Francisco Javier	Zoco Bravo	Desarrollo móvil	Lorem ipsum dolor sit amet, consectetuer adipis...
José Angel	Zoco Beltrán	Desarrollo web	Lorem ipsum dolor sit amet, consectetuer adipis...
José Vicente	Zoco Arena	Realidad Virtual	Lorem ipsum dolor sit amet, consectetuer adipis...
Rodrigo	Zoco	Realidad Virtual	
Esteban	Zoco	Desarrollo móvil	Lorem ipsum dolor sit amet, consectetuer adipis...
Marta	Zarco Valencia	Big Data	Lorem ipsum dolor sit am
Susana	Zarco Tudela	Métodos ágiles	Lorem ipsum dolor sit amet, consectetuer adipis...
María Dolores	Zarco Sierra	Realidad Virtual	Lorem ipsum dolor sit am
José Ignacio	Zarco Redondo	Realidad Virtual	Lorem ipsum dolor sit amet, consectetuer adipis...
Mercedes	Zarco Poveda	Desarrollo web	Lorem ipsum dolor sit amet, consectetuer adipis...
Valentín	Zarco Piñeiro	Desarrollo web	Lorem ipsum dolor sit amet, consectetuer adipis...
Concepción	Zarco LLorens	Ciberseguridad	Lorem ipsum dolor sit am
José Andrés	Zarco Gutierrez	Inteligencia Artificial	Lorem ipsum dolor sit amet, consectetuer adipis...
Juan Miguel	Zarco Guerrero	Ciberseguridad	Lorem ipsum dolor sit am
Adela	Zarco Cañada	Inteligencia Artificial	Lorem ipsum dolor sit amet, consectetuer adipis...
Javier	Zarco Alcalde	Realidad Virtual	Lorem ipsum dolor sit amet, consectetuer adipis...
José Antonio	Zapata Villares	Ciberseguridad	
Mario	Zapata Talavera	Realidad Virtual	Lorem ipsum dolor sit amet, consectetuer adipis...
Carmen	Zapata Preciado	Desarrollo web	Lorem ipsum dolor sit amet, consectetuer adipis...
María Luisa	Zapata Olmo	Ciberseguridad	Lorem ipsum dolor sit amet, consectetuer adipis...
María Luisa	Zapata Olmedo	Desarrollo móvil	Lorem ipsum dolor sit am
Jesús	Zapata Olmedo	Inteligencia Artificial	Lorem ipsum dolor sit am

Result 2 ×

Output

Action Output

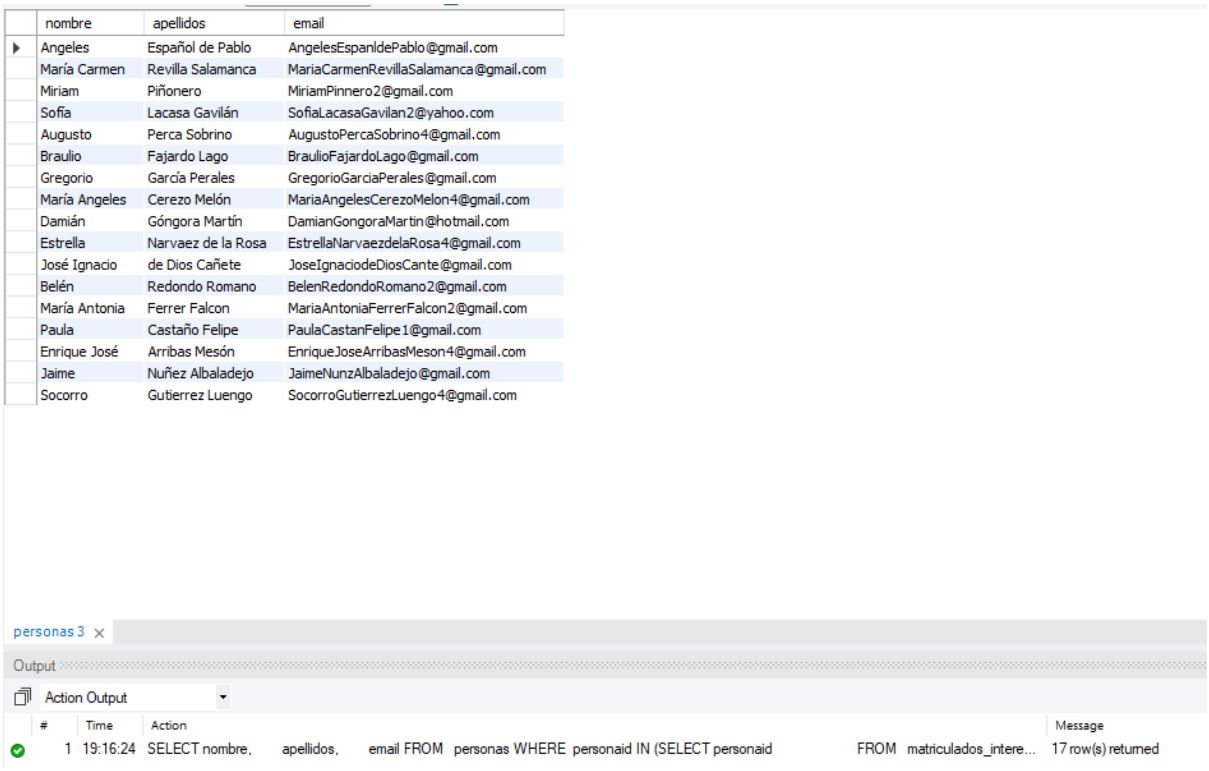
#	Time	Action	Message
1	19:15:00	SELECT personas.nombre, apellidos, area, comentarios FROM personas INNER JOIN matriculados_interesados ON ...	1000 row(s) returned

E. Obtener nombre completo y email de las personas matriculadas en un determinado curso.

En este caso, vamos a hacer uso de una subconsulta para obtener los ids de las personas matriculadas en un determinado curso y luego filtrar en la tabla personas por esos ids.

```
SELECT nombre,
       apellidos,
       email
FROM   personas
WHERE  personaid IN (SELECT personaid
                     FROM   matriculados_interesados
                     WHERE  cursoid = 256);
```

Resultado:



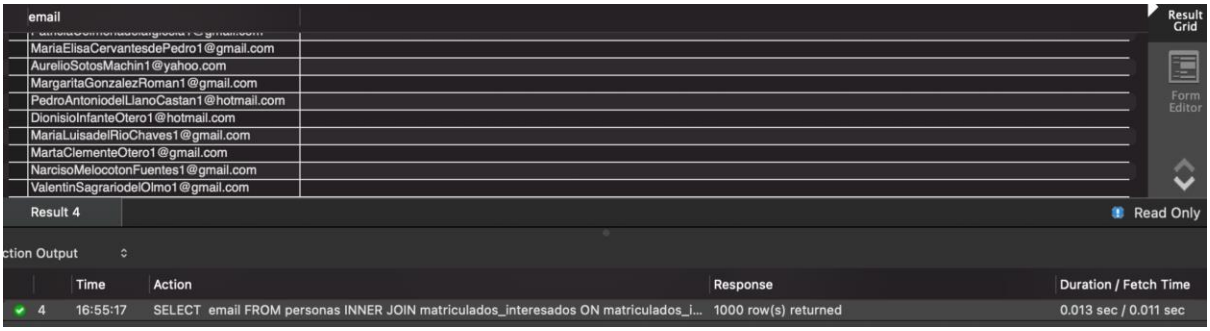
	nombre	apellidos	email
▶	Angeles	Español de Pablo	AngelesEspanldePablo@gmail.com
	María Carmen	Revilla Salamanca	MariaCarmenRevillaSalamanca@gmail.com
	Miriam	Piñonero	MiriamPinnero2@gmail.com
	Sofía	Lacasa Gavilán	SofiaLacasaGavilan2@yahoo.com
	Augusto	Perca Sobrino	AugustoPercaSobrino4@gmail.com
	Braulio	Fajardo Lago	BraulioFajardoLago@gmail.com
	Gregorio	García Perales	GregorioGarciaPerales@gmail.com
	María Angeles	Cerezo Melón	MariaAngelesCerezoMelon4@gmail.com
	Damián	Góngora Martín	DamianGongoraMartin@hotmail.com
	Estrella	Narvaez de la Rosa	EstrellaNarvaezdelaRosa4@gmail.com
	José Ignacio	de Dios Cañete	JoseIgnaciodeDiosCante@gmail.com
	Belén	Redondo Romano	BelenRedondoRomano2@gmail.com
	María Antonia	Ferrer Falcon	MariaAntoniaFerrerFalcon2@gmail.com
	Paula	Castaño Felipe	PaulaCastanFelipe1@gmail.com
	Enrique José	Arribas Mesón	EnriqueJoseArribasMeson4@gmail.com
	Jaime	Nuñez Albaladejo	JaimeNunzAlbaladejo@gmail.com
	Socorro	Gutierrez Luengo	SocorroGutierrezLuengo4@gmail.com

F. Consulta Interés 1: Obtener el email de aquellas personas matriculadas en un curso en 2020 las cuales lo hicieron a través de la web

Haremos uso de un “INNER JOIN” para juntar las tablas matriculados_interesados y personas mediante la clave foránea de la primera haciendo referencia a personaId y posteriormente filtraremos por canal y edición para obtener el deseado.

```
SELECT email
FROM   personas
       INNER JOIN matriculados_interesados
               ON matriculados_interesados.personaid =
personas.personaid
       INNER JOIN cursos
               ON cursos.cursoid =
matriculados_interesados.cursoid
WHERE  canal = 1
       AND edicion = 2020;
```

Resultado:



G. Consulta Interés 2: Obtener el teléfono de aquellas personas que se han matriculado alguna vez en un curso.

Para esta última consulta se usa el mismo planteamiento que la anterior pero filtrando por Matriculado el cual debe estar a 1 indicando que está matriculado y no tenga un teléfono nulo, en nuestro caso al ser un string vacío y no expresamente un NULL, usaremos “NOT LIKE ‘ ’ ”.

```
SELECT telefono
FROM   personas
       INNER JOIN matriculados_interesados
               ON matriculados_interesados.personaid =
personas.personaid
WHERE  matriculado = 1
       AND telefono NOT LIKE '';
```

Resultado:

```
| 995440565 |
| 995440565 |
| 995440565 |
| 995440565 |
| 994550836 |
| 994550836 |
| 995440565 |
| 994143575 |
| 995440565 |
| 990188107 |
| 995440565 |
| 990188107 |
| 990823786 |
| 990188107 |
| 990823786 |
| 990823786 |
| 990823786 |
| 995440565 |
| 990823786 |
| 998868133 |
| 995440565 |
| 990188107 |
| 993881373 |
+-----------+
15885 rows in set (0,06 sec)
```

2.2 Creación índices oportunos para las consultas y medición de tiempos

Tiempos de medición sin índices	A	B	C	D	E	F1	F2		MEDIA
Tiempo 1	0,05	0,08	0,05	27,07	0,09	2,42	0,08		
Tiempo 2	0,06	0,08	0,05	26,3	0,09	2,38	0,1		
Tiempo 3	0,06	0,08	0,05	26,12	0,09	2,43	0,1		
Media	0,05666667	0,08	0,05	26,4966667	0,09	2,41	0,09333333		4,18238095

Tiempos de medición Clave Primaria	A	B	C	D	E	F1	F2	
Tiempo 1	0,04	0,06	0,03	0,35	0,03	0,07	0,06	
Tiempo 2	0,05	0,06	0,03	0,22	0,03	0,06	0,06	
Tiempo 3	0,06	0,07	0,03	0,22	0,03	0,07	0,06	
Media	0,05	0,06333333	0,03	0,26333333	0,03	0,06666667	0,06	0,08047619

Tiempos de medición Claves fóraneas y primarias	A	B	C	D	E	F1	F2	
Tiempo 1	0,06	0,07	0,04	0,09	0	0,03	0,06	
Tiempo 2	0,06	0,07	0,04	0,07	0	0,04	0,06	
Tiempo 3	0,03	0,06	0,04	0,07	0	0,03	0,07	
Media	0,05	0,06666667	0,04	0,07666667	0	0,03333333	0,06333333	0,04714286

Tiempos de medición Claves FP e índice 1 en Apellidos(10)	A	B	C	D	E	F1	F2	
Tiempo 1	0	0,08	0,04	0,09	0	0,03	0,05	
Tiempo 2	0	0,08	0,04	0,07	0	0,03	0,06	
Tiempo 3	0	0,07	0,04	0,07	0	0,03	0,07	
Media	0	0,07666667	0,04	0,07666667	0	0,03	0,06	0,04047619

Tiempos de medición Claves FP e índice 2 en Provincia	A	B	C	D	E	F1	F2	
Tiempo 1	0,05	0,07	0	0,07	0	0,04	0,07	
Tiempo 2	0,05	0,07	0	0,05	0	0,03	0,06	
Tiempo 3	0,05	0,07	0	0,06	0	0,02	0,06	
Media	0,05	0,07	0	0,06	0	0,03	0,06333333	0,03904762

Tiempos de medición Claves FP e índice 3 en Edicion	A	B	C	D	E	F1	F2	
Tiempo 1	0,06	0,08	0,05	0,06	0	0,03	0,05	
Tiempo 2	0,05	0,07	0,04	0,06	0	0,03	0,06	
Tiempo 3	0,06	0,06	0,04	0,06	0	0,03	0,06	
Media	0,05666667	0,07	0,04333333	0,06	0	0,03	0,05666667	0,0452381

Tiempos de medición Claves FP y todos los índices Apellidos Provincia Edicion	A	B	C	D	E	F1	F2	
Tiempo 1	0	0,08	0	0,06	0	0,02	0,06	
Tiempo 2	0	0,07	0	0,06	0	0,03	0,06	
Tiempo 3	0	0,07	0	0,05	0	0,03	0,05	
Media	0	0,07333333	0	0,05666667	0	0,02666667	0,05666667	0,03047619

Teniendo en cuenta las consultas más utilizadas, observamos que se solía realizar un filtrado por provincia y edición. Siendo una int y la otra un varchar relativamente pequeño (30), decidimos establecer dichas columnas como índices para mejorar las máximas consultas posibles. Posteriormente decidimos establecer apellidos como índice utilizando solo los 10 primeros caracteres para mejorar en especial la primera query.

Con todos estos índices se observa una gran mejora, obteniendo 3 consultas con un tiempo cercano a los 0s y el resto no superando ninguna los 0,075s. A su vez si comparamos las medias, existe una diferencia enorme entre el tiempo medio de medición sin índices ni claves primarias y foráneas a la medición con claves primarias y foráneas más índices, siendo la primera de alrededor de 4,18s a 0,03s al final.
Los scripts de creación y eliminación de índices son los siguientes:

```
CREATE INDEX Personas_Index ON personas (Apellidos(10));
DROP INDEX Personas_Index ON personas;

CREATE INDEX Provincia_Index ON personas(Provincia);
DROP INDEX Provincia_Index ON personas;

CREATE INDEX Edicion_Index ON cursos(edicion);
DROP INDEX Edicion_Index ON cursos;
```

3.Conclusiones del cambio de tamaño de página.

En xampp en mysql dar a config al no tener el archivo innodb_page_size se crea y se le pone los valores de 4K, 16K, 64K y se modifica al igual el innodb_buffer_size a 4M,16M,64M y se guarda, a continuación se va a C:xampp/mysql/data se modifica data dejando solo my.ini y la carpeta mysql repetir el proceso para cada tamaño se inicia de nuevo todo y una vez estemos en phpMyAdmin ir a C:xampp/phpMyAdmin/sql ejecutar el script create_table.sql,creas la base de datos e importar todos los archivos de ella .
Cuando tenemos un tamaño de página pequeño de 4K el tamaño de la base de datos es más grande que la de 16K y 64K pero si aumentamos excesivamente el tamaño deja de ser eficiente y aumenta el tamaño de la base de datos como ocurre al pasar de 16K a 64K que aunque disminuyamos el número de páginas ocupa más espacio en la base de datos.
Con respecto a los tiempos, cuanto más grande es el tamaño de página disminuye el tiempo que tarda en realizar las consultas puesto que tiene menos páginas donde

consultar.

Tamaño	cursos	matriculados	personas
4K	1,6MB	20,1MB	20,6MB
16K	1,6MB	19,2MB	19,5MB
64K	832KB	20,1MB	20,6MB

	TIEMPO EN SEGUNDOS		
Consultas	4K	16K	64K
2.a	0,133	0,0737	0,0281
2.b	0,0106	0,0011	0,0022
2.c	0,1511	0,0179	0,0179
2.d	2,0558	0,3150	0,2008
2.e	0,0171	0,013	0,0016
2.f	0,0829	0,0392	0,0024
2.g	0,0131	0,0029	0,0018