

Programación con SQL

Carmen Witsman García | 10/01/2025 | Grado en Ciencia de Datos, UNIR



A partir de un esquema previamente creado en la anterior actividad "Lenguaje de definición de datos", trabajaremos con MySQL Workbench haciendo uso de sus distintas herramientas de gestión y manipulación de datos.

ÍNDICE

- 1. Esquema
- 2. Inserción de filas
- 3. Actualización de precios
- 4. Consultas
- 5. Vista
- 6. Procedimiento almacenado
- 7. Función
- 8. Disparadores (triggers)
- 9. Cursores

1. Esquema

El esquema obtenido de la actividad 1 de la asignatura Bases de Datos es el siguiente:

```

desarrolladora(idDes, nombreDes, pais)
juego(idJue, nombreJue, pegi, idDes)
cp(idcp, ciudad, provincia)
jugador(idJug, nombreJug, direccionJug, cp, tlfJug)
tipoCompeticion(idTipo, nombreTipo)
juez(idJuez, nombreJuez)
competicion(idCom, nombreCom, idTipo, cp, fecha, importeIns, idJuez)
inscripcion(idComp, idJug, fechaInsc)

```

Donde:

- **idDes:** Es el identificador de las desarrolladoras de juegos. *
- **nombreDes:** Es el nombre de las desarrolladoras.
- **pais:** Es el país de las desarrolladoras.
- **idJue:** Es el identificador para cada juego. *
- **nombreJue:** Es el nombre de cada juego.
- **pegi:** Es el pegi de los juegos.
- **idcp:** Es el código postal de distintas ciudades. *
- **ciudad:** Es la ciudad a la que hace referencia el código postal.
- **provincia:** Es la provincia a la que pertenecen las ciudades de la relación.
- **idJug:** Es el id de cada jugador. *
- **nombreJug:** Es el nombre de cada jugador.
- **direccionJug:** Es la dirección postal de cada jugador.
- **tlfJug:** Es el teléfono móvil del jugador.
- **idTipo:** Es el identificador para cada tipo de competición. *
- **nombreTipo:** Es el nombre del tipo de competición.
- **idJuez:** Es el identificador para cada juez. *
- **nombreJuez:** Es el nombre de los jueces.
- **idCom:** Es el identificador de cada competición. *
- **nombreCom:** Es el nombre de la competición.
- **fecha:** Es la fecha en la que se celebrará la competición.

- **importeIns:** Es el importe (en euros) para inscribirse a la competición.

* : Clave primaria.

2. Inserción de filas

Se nos pide que insertemos filas de manera que haya:

- **Jugadores inscritos en competiciones y jugadores no inscritos:**
Como en la tabla original de jugadores había solo 5 personas, añadimos 2 jugadores más:

```
1 • INSERT INTO jugador(idJug,nombreJug,direccionJug,cp,tlfJug)
2 VALUES
3 ('006','Odie Lita','Calle Soleado 9','23470','698775632'),
4 ('007','Millie Bobby','Calle Strangers 44','41110','655983421');
```

Figura 1. Inserción de filas en tabla jugador en MySQL.

Así, en la tabla de inscripciones podremos hacer la inserción de filas



```
139 • INSERT INTO inscripcion(idCom,idJug,fechaInsc)
140 VALUES
141 ('C01','002','23-10-2024'),
142 ('C02','006','04-12-2024'),
143 ('C05','002','23-10-2024'),
144 ('C03','001','22-09-2024'),
145 ('C03','005','01-12-2024'),
146 ('C04','006','01-08-2025'),
147 ('C02','003','02-02-2025'),
148 ('C08','001','04-12-2025'),
149 ('C07','002','01-07-2025');
```

Figura 2. Inserción de filas en tabla inscripcion en MySQL.

Vemos el contenido de la tabla `inscripcion`:

```
153 • SELECT * FROM inscripcion;
```

Result Grid

 Filter Rows:

	idCom	idJug	fechaInsc
▶	C01	002	23-10-2024
	C02	006	04-12-2024
	C05	002	23-10-2024
	C03	001	22-09-2024
	C03	005	01-12-2024
	C04	006	01-08-2025
	C02	003	02-02-2025
	C08	001	04-12-2025
	C07	002	01-07-2025

Figura 3. Tabla inscripcion en MySQL.

En esta tabla, se encuentran todas las competiciones menos la competición 6.

Los jugadores inscritos son los de los identificadores 001, 002, 003, 005 y 006, con lo cual los jugadores 004 y 007 no estarían inscritos en ninguna competición.

- **Jugadores inscritos en una competición y otros en varias:**

Teniendo la tabla anterior (Figura 3), podemos ver que:

- El jugador 002 está inscrito en las competiciones C01, C05 y C07
- El jugador 006 está inscrito en las competiciones C02 y C04
- El jugador 001 está inscrito en las competiciones C03 y C08
- El jugador 003 está inscrito en la competición C02
- El jugador 005 está inscrito en la competición C03

- **Jueces arbitrando en competiciones y otros que no:**

Como en la tabla original de jueces había solo 5 personas, añadimos 3 jueces más:

```

INSERT INTO juez(idJuez,nombreJuez)
VALUES
('JZ06','Ester Ackerman'),
('JZ07','Tomás Reiner'),
('JZ08','Paula Campos');

```

Figura 4. Inserción de filas en tabla juez en MySQL.

Así, en la tabla de competencias podremos hacer la inserción de filas:

```

129 • INSERT INTO competicion(idCom,nombreCom,idTipo,cp,fecha,importeIns,idJuez)
130 VALUES
131 ('C01','Picar diamantes','LOC','23470','10-12-2024',40,'JZ01'),
132 ('C02','Parkour extremo','REG','10480','30-07-2025',7,'JZ07'),
133 ('C03','Decoración de iglús','MUN','41110','01-01-2025',10,'JZ02'),
134 ('C04','La aldea duerme','MUN','10480','09-09-2025',3,'JZ05'),
135 ('C05','Wii','NAC','33150','06-02-2025',0,'JZ03'),
136 ('C06','Ping Pong','REG','23470','07-03-2025',3,'JZ07'),
137 ('C07','Juego de Sillas','NAC','41110','28-07-2025',0,'JZ02'),
138 ('C08','Batalla de gallos','LOC','10480','04-08-2025',5,'JZ04');

```

Figura 5. Inserción de filas en tabla competicion en MySQL.

Vemos el contenido de la tabla `competicion`:

	idCom	nombreCom	idTipo	cp	fecha	importeIns	idJuez
►	C01	Picar diamantes	LOC	23470	10-12-2024	40	JZ01
	C02	Parkour extremo	REG	10480	30-07-2025	7	JZ07
	C03	Decoración de iglús	MUN	41110	01-01-2025	10	JZ02
	C04	La aldea duerme	MUN	10480	09-09-2025	3	JZ05
	C05	Wii	NAC	33150	06-02-2025	0	JZ03
	C06	Ping Pong	REG	23470	07-03-2025	3	JZ07
	C07	Juego de Sillas	NAC	41110	28-07-2025	0	JZ02
	C08	Batalla de gallos	LOC	10480	04-08-2025	5	JZ04

Figura 6. Tabla competicion en MySQL.

Se puede observar en la tabla que los jueces JZ06 y JZ08 no arbitran ninguna competición, los jueces JZ01, JZ03 y JZ04 arbitran solo una competición y los jueces JZ07 y JZ02 arbitran dos competiciones.

- **Varias competiciones de cada tipo:**

Teniendo la tabla anterior (Figura 6), podemos ver que hay 2 competiciones de cada tipo:

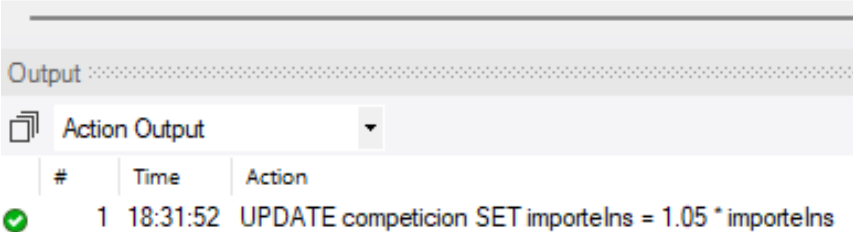
- Local: C01 y C08
- Regional: C02 Y C06
- Municipal: C03 y C04

- Nacional: C05 y C07

3. Actualización de precios

Se nos pide que el **precio** de todas las competencias **aumente un 5%**. Entonces, actualizamos el campo numérico `importeIns`, multiplicándolo por 1.05 (el 105%):

```
159 • UPDATE competicion
160 SET importeIns = 1.05 * importeIns;
```



Output

Action Output

#	Time	Action
1	18:31:52	UPDATE competicion SET importeIns = 1.05 * importeIns

Figura 7. Actualización de importeIns en tabla competicion en MySQL.

Para ver el cambio, comparamos la Figura 6 (tabla `competicion` antes de la actualización), y la tabla después del comando `UPDATE`.

	idCom	nombreCom	idTipo	cp	fecha	importeIns	idJuez	importeIns
►	C01	Picar diamantes	LOC	23470	10-12-2024	40	JZ01	44.1
	C02	Parkour extremo	REG	10480	30-07-2025	7	JZ07	7.7175
	C03	Decoración de iglús	MUN	41110	01-01-2025	10	JZ02	11.025
	C04	La aldea duerme	MUN	10480	09-09-2025	3	JZ05	3.3075
	C05	Wii	NAC	33150	06-02-2025	0	JZ03	0
	C06	Ping Pong	REG	23470	07-03-2025	3	JZ07	3.3075
	C07	Juego de Sillas	NAC	41110	28-07-2025	0	JZ02	0
	C08	Batalla de gallos	LOC	10480	04-08-2025	5	JZ04	5.5125

Figura 8. Comparación de tabla competicion antes y después de actualizar el atributo importeIns.

4. Consultas

4.1. Mostrar el nombre de cada jugador que esté inscrito en alguna competición junto con el nombre de la competición y el tipo, ordenados por

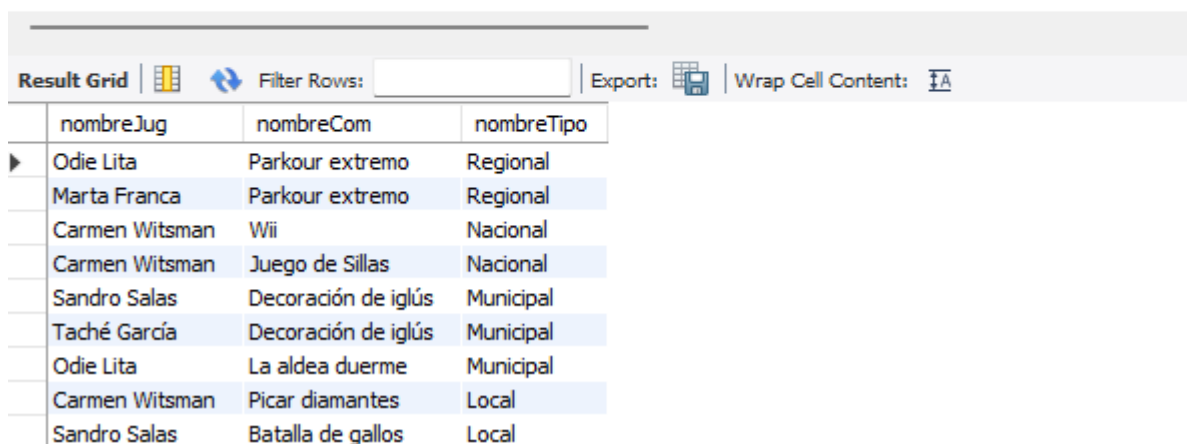
tipo de competición, descendente.

Para ello, hacemos reunión natural de las tablas:

- `inscripcion` y `competicion`: a través del campo en común `idCom`
- `inscripcion` y `jugador`: a través del campo en común `idJug`
- `competicion` y `tipocompeticion`: a través del campo en común `idTipo`

Como en la tabla `inscripcion` solo se encuentran los jugadores inscritos a alguna competición, usamos esa tabla para hacer reunión natural con las otras y así obtener los nombres de los jugadores competidores, los nombres de competiciones y los nombres de los tipos de competiciones. Finalmente, la ordenamos por tipo de competición descendente.

```
167 • SELECT jugador.nombreJug, competicion.nombreCom, tipocompeticion.nombreTipo
168 FROM inscripcion, competicion, jugador, tipocompeticion
169 WHERE inscripcion.idCom = competicion.idCom
170        AND inscripcion.idJug = jugador.idJug
171        AND competicion.idTipo = tipocompeticion.idTipo
172 ORDER BY tipocompeticion.idTipo DESC;
```



	nombreJug	nombreCom	nombreTipo
▶	Odie Lita	Parkour extremo	Regional
	Marta Franca	Parkour extremo	Regional
	Carmen Witsman	Wii	Nacional
	Carmen Witsman	Juego de Sillas	Nacional
	Sandro Salas	Decoración de iglús	Municipal
	Taché García	Decoración de iglús	Municipal
	Odie Lita	La aldea duerme	Municipal
	Carmen Witsman	Picar diamantes	Local
	Sandro Salas	Batalla de gallos	Local

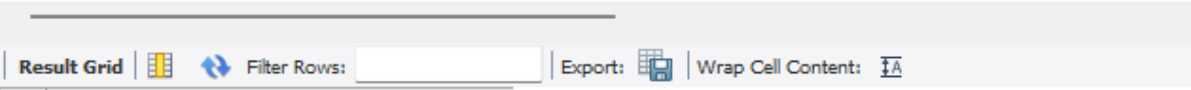
Figura 9. Consulta en MySQL y tabla de competidores y competiciones ordenada por tipo de competición.

Aparecen varias veces los mismos nombres ya que hay personas que están inscritas en múltiples competiciones.

4.2. Mostrar la competición cuyo precio de inscripción es más alto en cada uno de los tipos de competición, junto al importe medio del precio de inscripción en dicho tipo.

Para ello, usamos la función `max()` que calculará el precio máximo, `avg()` que calculará el precio medio y `GROUP BY` que agrupará la tabla por tipo de competición. Lo ordené de manera que el precio máximo mayor de cada tipo de competición sea el primero, de forma descendente.

```
176 • SELECT idTipo, max(importeIns) AS precioMásAlto, avg(importeIns) AS precioMedio
177 FROM competicion
178 GROUP BY idTipo
179 ORDER BY precioMásAlto DESC;
180
181 -- 4.3. Ciudades y provincias con competiciones nacionales o municipales
```



	idTipo	precioMásAlto	precioMedio
▶	LOC	42	23.625
	MUN	10.5	6.825000047683716
	REG	7.35	5.25
	NAC	0	0

Figura 10. Consulta en MySQL y tabla de precios máximos y medios por tipo de competición.

4.3. Mostrar las ciudades y provincias que han celebrado una competición nacional o municipal.

Utilizaré una consulta anidada en la que primero realizo reunión natural entre las tablas:

- `cp` y `competicion` : a través del campo en común `idcp`
- `tipocompeticion` y `competicion` : a través del campo en común `idTipo`

A dicha tabla la guardo temporalmente como `tabla1`, y es la que utilizo para hacer mi consulta, en la que seleccionaré únicamente a través de un `WHERE` las ciudades y provincias que tengan competiciones nacionales y/o municipales. Las grupo por provincia para que no salgan ciudades repetidas.

```
183 • SELECT ciudad, provincia
184 FROM
185 (SELECT cp.ciudad AS ciudad, cp.provincia AS provincia, tipocompeticion.nombreTipo AS tipo
186 FROM cp, competicion, tipocompeticion
187 WHERE cp.idcp = competicion.cp
188 AND tipocompeticion.idTipo = competicion.idTipo) AS tabla1
189 WHERE tipo = 'Nacional' OR tipo = 'Municipal'
190 GROUP BY ciudad, provincia;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [iA](#)

ciudad	provincia
Bollullos de la Mitación	Sevilla
Valverde de la Vera	Cáceres
Cudillero	Asturias

Figura 11. Consulta en MySQL y tabla de ciudades con competiciones municipales o nacionales.

4.4. Mostrar las ciudades y provincias que han celebrado una competición nacional y municipal.

Usamos la misma consulta solo que comprobamos con `HAVING` que la ciudad tenga más de 1 tipo diferente de competiciones y que tenga 1 de cada (nacional y municipal).

```

194 • SELECT ciudad, provincia
195 FROM
196 (SELECT cp.ciudad AS ciudad, cp.provincia AS provincia, tipocompeticion.nombreTipo AS tipo
197 FROM cp, competicion, tipocompeticion
198 WHERE cp.idcp = competicion.cp
199 AND tipocompeticion.idTipo = competicion.idTipo) AS tabla1
200 GROUP BY ciudad, provincia
201 HAVING count(tipo) > 1
202 AND sum(tipo = 'Nacional') > 0
203 AND sum(tipo = 'Municipal') > 0;

```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
ciudad	provincia		
Bollullos de la Mitación	Sevilla		

Figura 12. Consulta en MySQL y tabla de ciudades con competencias municipales y nacionales.

4.5. Mostrar las ciudades y provincias que han celebrado una competición nacional y no mundial.

Esta vez, en la cláusula **HAVING** especificamos que haya al menos una competición nacional en la ciudad y que haya 0 competencias municipales.

```

206 • SELECT ciudad, provincia
207 FROM
208 (SELECT cp.ciudad AS ciudad, cp.provincia AS provincia, tipocompeticion.nombreTipo AS tipo
209 FROM cp, competicion, tipocompeticion
210 WHERE cp.idcp = competicion.cp
211 AND tipocompeticion.idTipo = competicion.idTipo) AS tabla1
212 GROUP BY ciudad, provincia
213 HAVING sum(tipo = 'Nacional') > 0
214 AND sum(tipo = 'Municipal') = 0;

```

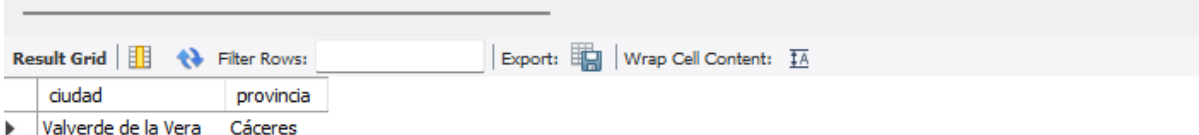
Result Grid	Filter Rows:	Export:	Wrap Cell Content:
ciudad	provincia		
Cudillero	Asturias		

Figura 13. Consulta en MySQL y tabla de ciudades con competencias nacionales y no municipales.

4.6. Mostrar las ciudades y provincias que no han celebrado una competición nacional pero sí mundial.

El proceso es el mismo que en el punto anterior:

```
219 • SELECT ciudad, provincia
220 FROM
221 (SELECT cp.ciudad AS ciudad, cp.provincia AS provincia, tipocompeticion.nombreTipo AS tipo
222 FROM cp, competicion, tipocompeticion
223 WHERE cp.idcp = competicion.cp
224 AND tipocompeticion.idTipo = competicion.idTipo) AS tabla1
225 GROUP BY ciudad, provincia
226 HAVING sum(tipo = 'Nacional') = 0
227 AND sum(tipo = 'Municipal') > 0;
```



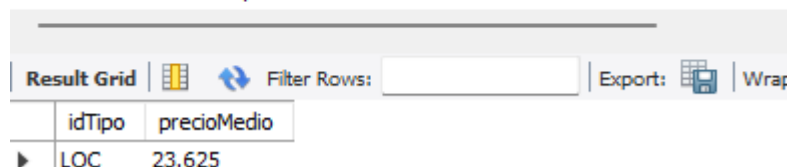
ciudad	provincia
Valverde de la Vera	Cáceres

Figura 14. Consulta en MySQL y tabla de ciudades con competiciones municipales y no nacionales.

4.7. Mostrar el importe de inscripción medio de los tipos de competición cuya media es mayor que 20 €

De la tabla `competicion`, seleccionamos el tipo y el precio medio del importe, lo agrupamos por tipo y con la cláusula `HAVING` establecemos que nos muestre únicamente los que tengan media mayor a 20.

```
231 • SELECT idTipo, avg(importeIns) AS precioMedio
232 FROM competicion
233 GROUP BY idTipo
234 HAVING precioMedio > 20;
```



idTipo	precioMedio
LOC	23.625

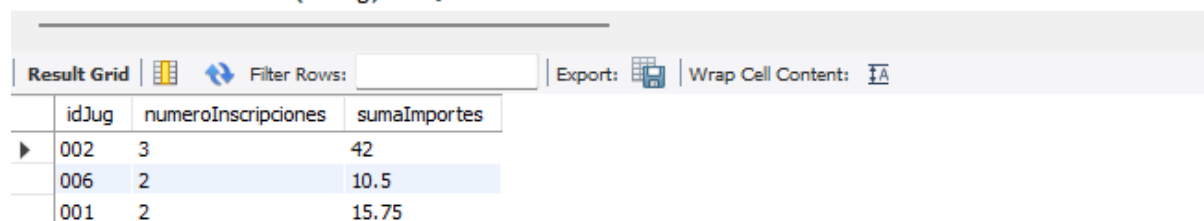
Figura 15. Consulta de medias de importes de inscripción mayores a 20 por tipo de competición en MySQL.

4.8. Mostrar la suma de los importes pagados por los jugadores que se han inscrito en más de una competición.

Hacemos reunión natural de la tabla `inscripcion` y `competicion` con el atributo común `idCom`, así podemos recuperar los jugadores y los importes de las inscripciones. De esa tabla que obtenemos, seleccionamos los identificadores de los jugadores, el número de inscripciones, y la suma de los importes de cada inscripción.

Para comprobar que el jugador está inscrito en más de una competición, usamos la cláusula `HAVING` en la que con `count()` contaremos el número de veces que aparece cada jugador en la tabla.

```
238 • SELECT idJug, count(idJug) AS numeroInscripciones, sum(importeIns) AS sumaImportes
239 FROM
240 (SELECT inscripcion.idJug, competicion.importeIns
241 FROM inscripcion, competicion
242 WHERE inscripcion.idCom = competicion.idCom) AS tabla1
243 GROUP BY idJug
244 HAVING count(idJug) > 1;
```



	idJug	numeroInscripciones	sumaImportes
▶	002	3	42
	006	2	10.5
	001	2	15.75

Figura 16. Consulta de sumas de importes de inscripciones de jugadores inscritos a más de una competición en MySQL.

5. Vista

Crearemos una **vista que sea el resultado de una consulta que incluya, al menos, una reunión natural, una función de agregación y una cláusula *having*.**

A esta vista la llamaré `pago_clientes`, en la que podremos **ver información sobre jugadores inscritos a competiciones** (nombre, dirección, teléfono), **así como el total que deben pagar**, por todas sus inscripciones. En esta vista no se incluirán aquellos jugadores que se inscribieron a competiciones gratuitas.

La **utilidad** de esta vista es que, en el caso de que una empresa tenga que gestionar los pagos de forma global, le interesará conocer el teléfono o la vivienda para poder reclamar en caso de impago, y el importe para solicitar que sea pagado. De igual forma, no les interesará aquellos jugadores que no deben pagar nada, así los procesos de búsqueda serán más eficientes.

La consulta que implementé fue la siguiente:

```

249 • DROP VIEW pago_clientes;
250 • CREATE VIEW pago_clientes AS(
251     SELECT j.idJug AS id,
252           j.nombreJug AS nombre,
253           j.direccionJug AS direccion,
254           j.tlfJug AS telefono,
255           sum(c.importeIns) AS total_pagar
256     FROM jugador j, competicion c, inscripcion i
257     WHERE j.idJug = i.idJug AND
258           i.idCom = c.idCom
259     GROUP BY j.idJug
260     HAVING sum(c.importeIns > 0)
261 );
262
263 • SELECT * FROM pago_clientes;

```

Result Grid					
		Filter Rows:		Export:	Wrap Cell Content:
	id	nombre	direccion	telefono	total_pagar
▶	002	Carmen Witsman	Calle Angelsoni 3	695432878	42
	006	Odie Lita	Calle Soleado 9	698775632	10.5
	001	Sandro Salas	Calle Carámbola 20	657893222	15.75
	005	Taché García	Calle Don Limpio 11	627554213	10.5
	003	Marta Franca	Calle Pisalobos 15	722454398	7.349999904632568

Figura 17. Vista de clientes inscritos a competencias no gratuitas y el total a pagar en My SQL.

6. Procedimiento almacenado

Crearemos un procedimiento almacenado que permita hacer una inscripción en una competición. Para ello es necesario que se soliciten todos los datos necesarios al usuario del sistema.

Para proceder con el procedimiento, deberemos introducir el nombre del jugador, su identificador, su dirección, su código postal, su teléfono, el nombre de la competición y la fecha de inscripción.

Primero, almacenamos en una tabla temporal el identificador de la competición a partir de la tabla

`competicion` y la variable del procedimiento `v_nombreCom`. Es decir, introducimos el nombre de la competición y mediante reunión natural almacenamos el identificador correspondiente.

Luego, se comprueba si dicha competición existe. Si es así, se sigue con el procedimiento, si no, se lanza un mensaje de error.

En el caso de que el jugador constase previamente en la base de datos, solo se introducen los datos en la tabla

`inscripcion`. De forma contraria, si el jugador no estaba registrado previamente en la base de datos, primero se inserta en la tabla `jugador`, y luego en la tabla `competicion`.

El código implementado es el siguiente:

```
268 DELIMITER $$
269
270 CREATE PROCEDURE inscribir(IN v_nombreJug varchar(60),
271                             IN v_idJug char(3),
272                             IN v_direccion varchar(60),
273                             IN v_cp char(5),
274                             IN v_tlfJug char(9),
275                             IN v_nombreCom varchar(60),
276                             IN v_fechaIns char(10))
277 BEGIN
278     DROP TEMPORARY TABLE IF EXISTS compe;
279     CREATE TEMPORARY TABLE compe (
280         idcompeticion char(3));
281
282     INSERT INTO compe
283     SELECT idCom
284     FROM competicion
285     WHERE v_nombreCom = nombreCom;
286
287     -- La competición no existe
288     IF NOT EXISTS (SELECT 1 FROM competicion WHERE v_nombreCom = competicion.nombreCom)
289     THEN SIGNAL SQLSTATE '45000'
290          SET MESSAGE_TEXT = 'La competición no existe';
291
292     END IF;
```

```

294      -- El jugador ya estaba inscrito
295      IF EXISTS (SELECT 1 FROM compe c, inscripcion i
296                  WHERE v_idJug = i.idJug
297                        AND i.idCom = c.idcompeticion)
298      THEN SIGNAL SQLSTATE '45001'
299      SET MESSAGE_TEXT = 'El jugador ya está inscrito en la competición';
300
301      END IF;
302
303      -- El jugador ya constaba en la base de datos
304
305      IF EXISTS (SELECT 1 FROM jugador WHERE v_nombreJug = nombreJug
306                AND v_idJug = idJug)
307      THEN
308      INSERT INTO inscripcion (idJug, idCom, fechaInsc)
309      SELECT v_idjug, idcompeticion, v_fechaIns
310      FROM compe;
311
312      END IF;
313
314      END $$
315
316      DELIMITER ;

```

Figura 18. Procedimiento almacenado de inscripción de jugadores en competiciones en MySQL.

Probamos el procedimiento inscribiendo a tres personas a competiciones: una persona que ya estaba inscrita a dicha competición, otra que constaba en la base de datos, y otra que no constaba en la base de datos.

1. Jugador ya inscrito en competición (Error)

```

326 • CALL inscribir(
327     'Carmen Witsman',
328     '002',
329     'Calle Angelsoni 3',
330     '29400',
331     '695432878',
332     'Picar Diamantes',
333     '10-11-2024');

```

Output

Action Output

#	Time	Action	Message
1	18:28:37	CALL inscribir...	Error Code: 1644. El jugador ya está inscrito en la competición

Figura 19. Llamada a procedimiento de inscripción en MySQL.

2. Jugador existente en la BBDD

```
SELECT * FROM inscripcion;
CALL inscribir(
'Sandro Salas',
'001',
'Calle Carámbola 20',
'41110',
'657893222',
'Wii',
'11-11-2024');
SELECT * FROM inscripcion;
```

Figura 20. Llamada a procedimiento de inscripción 2 en MySQL

idCom	idJug	fechaInsc
C01	002	23-10-2024
C02	006	04-12-2024
C05	002	23-10-2024
C03	001	22-09-2024
C03	005	01-12-2024
C04	006	01-08-2025
C02	003	02-02-2025
C08	001	04-12-2025
C07	002	01-07-2025

Figura 21. Tabla inscripción antes de procedimiento 2.

idCom	idJug	fechaInsc
C01	002	23-10-2024
C02	006	04-12-2024
C05	002	23-10-2024
C03	001	22-09-2024
C03	005	01-12-2024
C04	006	01-08-2025
C02	003	02-02-2025
C08	001	04-12-2025
C07	002	01-07-2025
C05	001	11-11-2024

Figura 22. Tabla inscripción después de procedimiento 2

Vemos que se añade el jugador 001 a la competición C05 (última fila de la tabla de Figura 22).

3. Jugador no existente en BBDD

```
CALL inscribir(
'Garbanzo Loco',
'009',
'Calle Guam 34',
'41110',
'698666547',
'Parkour extremo',
'22-10-2024');
SELECT * FROM inscripcion;
```

Figura 23. Llamada a procedimiento de inscripción 3 en MySQL

idCom	idJug	fechaInsc
C01	002	23-10-2024
C02	006	04-12-2024
C05	002	23-10-2024
C03	001	22-09-2024
C03	005	01-12-2024
C04	006	01-08-2025
C02	003	02-02-2025
C08	001	04-12-2025
C07	002	01-07-2025
C05	001	11-11-2024

Figura 22. Tabla inscripción después de procedimiento 2

idCom	idJug	fechaInsc
C01	002	23-10-2024
C02	006	04-12-2024
C05	002	23-10-2024
C03	001	22-09-2024
C03	005	01-12-2024
C04	006	01-08-2025
C02	003	02-02-2025
C08	001	04-12-2025
C07	002	01-07-2025
C05	001	11-11-2024
C02	009	22-10-2024

Figura 24. Tabla inscripción después de procedimiento 3

Vemos que se añade el jugador 009 a la competición C02 (última fila de la tabla de Figura 24).

7. Función

Crearemos una **función que permita saber el importe total recaudado por las inscripciones de una competición**, que se pasará como argumento.

En nuestra función, pasamos el nombre de la competición como argumento, y la función hace una reunión natural de los nombres para encontrar en la tabla las coincidencias, y así poder sumar con la función de agregación `sum()` todos los importes correspondientes a la competición pasada por parámetro.

El código implementado es el siguiente:

```
359  DELIMITER //
360  • CREATE FUNCTION recaudado (v_competicion varchar(60)) RETURNS float
361  DETERMINISTIC
362  BEGIN
363  RETURN (SELECT sum(importeIns)
364          FROM competicion
365          WHERE v_competicion = competicion.nombreCom);
366  END //
367  DELIMITER ;
```

Figura 25. Función para conocer importe recaudado en competiciones en MySQL.

Hacemos una prueba para conocer el importe total recaudado en la competición 'Decoración de iglús':

369 • `SELECT recaudado('Decoración de iglús');`

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	recaudado('Decoración de iglús')			
▶	10.5			

Figura 26. Llamada a función para conocer importe recaudado en la competición 'Decoración de iglús' en MySQL.

8. Disparadores (triggers)

8.1. Crearemos un disparador (*trigger*) que, cuando se realice una inscripción a una competición, escriba en la tabla *inscripciones_log* indicando la fecha de realización de la operación, la competición y jugador afectados y el usuario que ha realizado la operación.

El primer paso será crear la tabla `inscripciones_log`:

```
CREATE TABLE inscripciones_log (
    id INT AUTO_INCREMENT PRIMARY KEY,
    fecha_operacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    id_competicion CHAR(3),
    id_jugador CHAR(3),
    usuario_operacion VARCHAR(255)
);
```

Figura 27. Creación de tabla `inscripciones_log` en MySQL.

Los campos de esta nueva tabla serán:

- `id INT AUTO_INCREMENT PRIMARY KEY`: Esta columna `id` es una clave primaria que se incrementa automáticamente con cada nuevo registro en la tabla. Esto asegura que cada registro tiene un identificador único.

- `fecha_operacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP` : La columna `fecha_operacion` almacena la fecha y hora en que se realizó la operación. La función `CURRENT_TIMESTAMP` se usa para guardar automáticamente la fecha y hora actuales.
- `id_competicion CHAR(3)` : Esta columna almacena el identificador de la competición afectada por la inscripción.
- `id_jugador CHAR(3)` : Esta columna almacena el identificador del jugador que se inscribió.
- `usuario_operacion VARCHAR(255)` : Esta columna almacena el nombre del usuario que realizó la operación. La función `CURRENT_USER()` se usa para obtener esta información.

Ahora, crearemos el trigger, que “saltará” después de que se inserte una fila en la tabla `inscripcion` :

```
-- Trigger
DELIMITER //
CREATE TRIGGER after_inscripcion_insert
AFTER INSERT ON inscripcion
FOR EACH ROW
BEGIN
    INSERT INTO inscripciones_log (id_competicion, id_jugador, usuario_operacion)
    VALUES (NEW.idCom, NEW.idJug, CURRENT_USER());
END //
DELIMITER ;
```

Figura 28. Creación de trigger para inserciones en la tabla `inscripcion` en MySQL.

- `AFTER INSERT ON inscripcion` : Especifica que el disparador se ejecutará después (`AFTER`) de que se inserte una fila en la tabla `inscripcion` .
- `FOR EACH ROW` : Indica que el disparador se ejecutará para cada fila que se inserte en la tabla `inscripcion` .
- `BEGIN ... END` : Delimita el bloque de código que se ejecutará cuando se active el disparador.
- `INSERT INTO inscripciones_log (id_competicion, id_jugador, usuario_operacion) VALUES (NEW.idcompeticion, NEW.idJug, CURRENT_USER())` : Esta línea inserta un nuevo registro en la tabla `inscripciones_log` con los siguientes valores:

- `NEW.idCom` : El valor del campo `idCom` de la nueva fila insertada en `inscripcion`.
- `NEW.idJug` : El valor del campo `idJug` de la nueva fila insertada en `inscripcion`.
- `CURRENT_USER()` : El usuario actual que ejecutó la operación.

Hacemos una nueva inscripción con el procedimiento que creamos en el punto 6 y vemos como varía la tabla `inscripciones_log` :

```
SELECT * FROM inscripciones_log;
CALL inscribir(
    'Carmen Witsman',
    '002',
    'Calle Angelsoni 3',
    '29400',
    '695432878',
    'Ping Pong',
    '10-11-2024');
SELECT * FROM inscripciones_log;
```

Figura 29. Llamada a procedimiento de inscripción 4 en MySQL

id	fecha_operacion	id_competicion	id_jugador	usuario_operacion
NULL	NULL	NULL	NULL	NULL
1	2025-01-11 19:34:33	C06	002	root@localhost
NULL	NULL	NULL	NULL	NULL

Figura 30. Antes y después de la tabla `inscripciones_log` tras llamada a procedimiento de inscripción 4 en MySQL

8.2. Crearemos un disparador (*trigger*) que impida rebajar el importe de las inscripciones.

Seguimos el procedimiento del apartado anterior, pero esta vez el disparador se ejecutará antes de la consulta. Se comprobará si el precio nuevo de las inscripciones es menor y, en caso de ser así, nos aparecerá un mensaje de error.

El código implementado es:

```

DELIMITER //
CREATE TRIGGER no_rebajar
BEFORE UPDATE ON competicion
FOR EACH ROW
BEGIN
    IF NEW.importeIns < OLD.importeIns
    THEN SIGNAL SQLSTATE '45009'
    SET MESSAGE_TEXT = 'No se puede rebajar el importe de la inscripción';
    END IF;
END //
DELIMITER ;

```

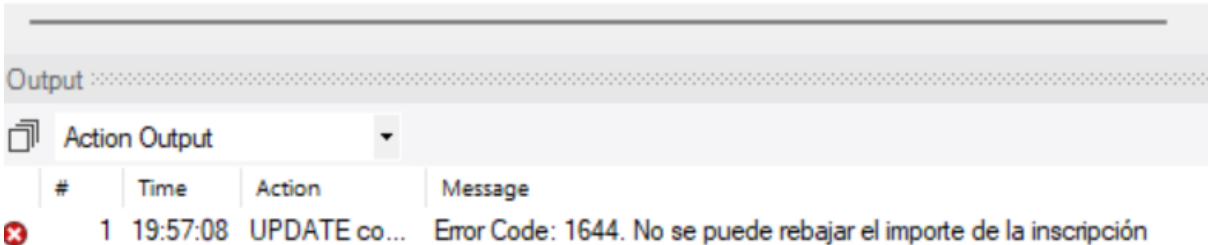
Figura 32. Disparador que impide rebajar precios de inscripciones de la tabla competicion en MySQL.

Si queremos rebajar un 5% los precios de las inscripciones, “salta” el mensaje de error del disparador:

```

420 • UPDATE competicion
421 SET importeIns = 0.95 * importeIns;

```



#	Time	Action	Message
1	19:57:08	UPDATE co...	Error Code: 1644. No se puede rebajar el importe de la inscripción

Figura 33. Actualización fallida de precio de inscripciones en MySQL.

9. Cursores

Crearemos un **procedimiento que incluya, al menos, un cursor, operaciones que combinen dos o más tablas y actualización de una tabla.**

Este procedimiento `informacion_competicion_provincia` recibe como parámetro el nombre de una provincia y realiza las siguientes tareas:

1. **Obtención de códigos postales:** Utiliza un cursor (`cur1`) para obtener todos los códigos postales (`idcp`) asociados a la provincia proporcionada.

2. **Recorrido de competencias:** Utiliza un segundo cursor (`cur2`) para recorrer todas las competencias que se celebran en cada uno de los códigos postales obtenidos.
3. **Cálculo de datos:** Durante el recorrido de las competencias, se cuentan el número total de competencias (`v_numComp`), el importe total recaudado (`v_totalRecaudado`) y el número total de inscripciones (`v_totalInscripciones`).
4. **Cálculo del precio medio:** Calcula el precio medio de las inscripciones si hay competencias en la provincia.
5. **Actualización de la tabla `resumenProvincia`:** Inserta o actualiza la tabla `resumenProvincia` con los datos calculados.
6. **Devolución de resultados:** Devuelve los resultados de la tabla `resumenProvincia` específica para la provincia proporcionada.

El código implementado es el siguiente:

```
CREATE TABLE resumenProvincia (  
    provincia VARCHAR(50) PRIMARY KEY,  
    totalCompeticiones INT,  
    totalJugadores INT,  
    precioMedio FLOAT  
);  
  
DELIMITER //  
CREATE PROCEDURE informacion_competicion_provincia(IN nombreProvincia VARCHAR(50))  
BEGIN  
    DECLARE v_idcp CHAR(5);  
    DECLARE v_idCom CHAR(3);  
    DECLARE v_numComp INT DEFAULT 0;  
    DECLARE v_totalRecaudado FLOAT DEFAULT 0.0;  
    DECLARE v_totalImporte FLOAT DEFAULT 0.0;  
    DECLARE v_totalInscripciones INT DEFAULT 0;  
    DECLARE done INT DEFAULT 0;  
  
    -- Cursor para obtener los idcp de la provincia  
    DECLARE cur1 CURSOR FOR  
        SELECT idcp  
        FROM cp  
        WHERE provincia = nombreProvincia;
```

```

-- Cursor para recorrer las competencias en la provincia
DECLARE cur2 CURSOR FOR
    SELECT idCom, importeIns
    FROM competicion
    WHERE cp = v_idcp;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

-- Recorrer los códigos postales de la provincia
OPEN cur1;

read_cps: LOOP
    FETCH cur1 INTO v_idcp;
    IF done THEN
        LEAVE read_cps;
    END IF;

    -- Recorrer las competencias en cada código postal
    OPEN cur2;

    read_comps: LOOP
        FETCH cur2 INTO v_idCom, v_totalImporte;
        IF done THEN
            LEAVE read_comps;
        END IF;

        -- Recorrer las competencias en cada código postal
        OPEN cur2;

        read_comps: LOOP
            FETCH cur2 INTO v_idCom, v_totalImporte;
            IF done THEN
                LEAVE read_comps;
            END IF;

            -- Contar competencias
            SET v_numComp = v_numComp + 1;

            -- Calcular el total recaudado
            SET v_totalRecaudado = v_totalRecaudado + v_totalImporte;

            -- Contar las inscripciones
            SET v_totalInscripciones = v_totalInscripciones +
                (SELECT COUNT(*)
                 FROM inscripcion
                 WHERE idCom = v_idCom);

        END LOOP;
    CLOSE cur2;
END LOOP;
CLOSE cur1;

```

```

-- Calcular el precio medio si hay competiciones
IF v_numComp > 0 THEN
    SET v_totalImporte = v_totalRecaudado / v_numComp;
ELSE
    SET v_totalImporte = 0.0;
END IF;

-- Insertar o actualizar la tabla resumenProvincia
INSERT INTO resumenProvincia (provincia, totalCompeticiones, totalJugadores, precioMedio)
VALUES (nombreProvincia, v_numComp, v_totalInscripciones, v_totalImporte)
ON DUPLICATE KEY UPDATE
    totalCompeticiones = v_numComp,
    totalJugadores = v_totalInscripciones,
    precioMedio = v_totalImporte;
-- Devolver los resultados finales
SELECT * FROM resumenProvincia WHERE provincia = nombreProvincia;
END //

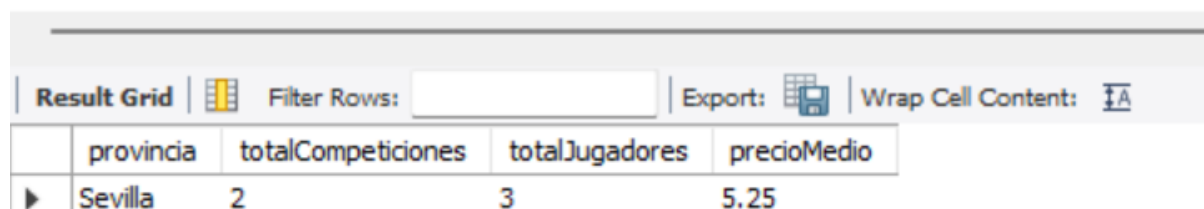
DELIMITER ;

```

Figura 34. Procedimiento almacenado con cursores para obtener resúmenes de competiciones por provincia en MySQL.

Probamos el procedimiento:

```
511 • CALL informacion_competicion_provincia('Sevilla');
```



	provincia	totalCompeticiones	totalJugadores	precioMedio
▶	Sevilla	2	3	5.25

Figura 35. Llamada a procedimiento almacenado para obtener resumen de competiciones en Sevilla en MySQL.

Este procedimiento es de gran importancia, ya que proporciona una comprensión clara y detallada de las competiciones que se llevan a cabo en una provincia. Permite:

- Identificar el número total de competiciones y jugadores inscritos en la provincia.
- Calcular el precio medio de las inscripciones en relación con el rendimiento financiero.

- Facilitar la gestión y planificación de eventos deportivos.

Esta información resulta fundamental para la administración eficiente y la optimización de los recursos y actividades deportivas.