



Laboratorio de Bases de Datos (EBB)

Unidad III – Consultas

Departamento de Electricidad, Electrónica y Computación
Facultad de Ciencias Exactas y Tecnología
Universidad Nacional de Tucumán



Primer Cuatrimestre 2024



Contenido

01

Consultas
básicas

02

Múltiples tablas

03

Técnicas
avanzadas

Consultas básicas



- Inserción de datos
- Recuperación de datos
- Modificación de datos
- Borrado de datos

Inserción de datos

Problema:

- Se quiere llevar información sobre los distintos productos que se venden
- Inicialmente, de los productos se conoce su nombre e identificador (número entero que identifica unívocamente al producto)

Inserción de datos

- Para agregar filas a una tabla se usa la sentencia INSERT

```
INSERT [INTO] tabla [(lista_columnas)]  
VALUES (lista_valores)
```

Inserción de datos

Consideraciones:

- Usar la lista de columnas para forzar el orden de los valores
- Especificar los datos a ingresar luego de la palabra VALUES (el orden y tipo de datos deben coincidir con los de la tabla)
- La sentencia INSERT falla si se violan las restricciones

Inserción de datos

Consideraciones:

- MySQL permite insertar múltiples filas, separando cada grupo de valores por una coma
- Si la columna tiene valores por defecto o acepta nulos, se puede omitir el nombre en la sentencia INSERT
- Especificar el valor nulo escribiendo NULL (sin comillas)

Inserción de datos

Consideraciones:

- Si la columna tiene un valor por defecto también se puede emplear la cláusula DEFAULT
- Si la columna no admite nulos ni tiene valor por defecto, falla la sentencia
- No se puede usar para columnas AUTO_INCREMENT

Recuperación de datos

Sentencia SELECT:

- Se usa para recuperar los datos

```
SELECT [ALL | DISTINCT] <lista_selección>  
FROM {<tabla_fuente>} [...n]  
WHERE <predicado_búsqueda>
```

Recuperación de datos

Sentencia **SELECT**:

- La lista de selección puede contener:
 - Columnas
 - Expresiones
 - Variables
 - Constantes

Recuperación de datos

Sentencia **SELECT**:

- Sobre la lista de selección:
 - Devuelve los resultados en el orden listado
 - Las columnas se separan con comas
 - El (*) devuelve todas las columnas (evitar su uso)

Recuperación de datos

Sentencia **SELECT**:

- Sobre la cláusula **WHERE**:
 - Limita el número de filas devueltas según la condición de búsqueda: sólo se devuelven las filas que cumplen con los criterios de la cláusula
 - Para cadenas y fechas se emplean comillas simples o dobles
 - Conviene emplear condiciones de búsqueda positivas

Recuperación de datos

Sentencia **SELECT**:

- En una cláusula WHERE se pueden emplear:
 - Comparadores
 - Operadores lógicos
 - Rango de valores
 - Lista de valores
 - Valores desconocidos

Recuperación de datos

Comparadores

- =, >, <, >=, <=, <>, LIKE, NOT LIKE
- Comparan columnas, expresiones, variables y constantes
- Conviene usar condiciones positivas

Recuperación de datos

Comparadores

- Para comparar cadenas y fechas se puede usar LIKE:
 - %: Cualquier cadena de 0 o más caracteres
 - _ : Cualquier carácter
 - LIKE 'BR%': cadenas que empiecen con “BR”
 - LIKE '_er': cadenas de 3 letras terminadas en “er”

Recuperación de datos

Comparadores

- También se pueden usar expresiones regulares mediante el uso de la cláusula REGEXP

Recuperación de datos

Operadores lógicos

- NOT, AND, OR (en ese orden)
- Se pueden combinar expresiones
- Admiten paréntesis

Recuperación de datos

Rango de valores

- Para obtener filas cuyos valores de búsqueda estén dentro de un rango se usa el operador BETWEEN
- La inclusión de los extremos del intervalo depende del SGBDR
- Es mejor usar BETWEEN que \leq y \geq
- Evitar el uso de NOT BETWEEN
- En las fechas hay que tener en cuenta la hora

Recuperación de datos

Lista de valores

- Para obtener filas cuyos valores de búsqueda pertenezcan a un conjunto dado se usa la condición IN
- Se pueden reemplazar por series de expresiones OR
- No incluir el valor nulo en el conjunto (resultados inesperados)
- El uso de NOT IN disminuye el rendimiento

Recuperación de datos

Valores desconocidos

- Una columna tiene el valor nulo (NULL) si no se ingresó valor alguno
- Un valor nulo no equivale a la cadena vacía ni a 0
- Para devolver filas cuyos valores de búsqueda sean nulos se usa IS NULL
- Usar IS NOT NULL para devolver filas con valores conocidos

Recuperación de datos

Orden de los datos

- Para ordenar las filas en orden ascendente (ASC) o descendente (DESC) se emplea ORDER BY
- Si no se lo usa no se garantiza un orden determinado
- Por defecto se ordena ascendentemente (ASC)
- Las columnas por las que se ordenan no necesitan aparecer en la lista de selección
- Se puede ordenar por columnas o expresiones y referenciarlas por su número de orden

Recuperación de datos

Eliminación de duplicados

- Para eliminar filas duplicadas se puede usar DISTINCT
- La combinación de valores en la lista de selección determina la desigualdad entre filas: sólo se muestra un elemento de todas las filas que tienen la misma combinación de valores de la lista de selección
- DISTINCT no ordena, para ello se usa ORDER BY

Recuperación de datos

Cambio de nombres a las columnas

- Por defecto se muestran los nombres de las columnas tal cual fueron definidas al momento de crear la tabla
- Para hacer más legibles sus nombres se les puede asignar un alias
- También se pueden definir alias para expresiones
- Se puede omitir la cláusula AS y dejar un espacio entre la expresión y el alias

Recuperación de datos

Uso de constantes

- Las constantes o literales son letras, números o símbolos

Recuperación de datos

CASE

- Evalúa secuencialmente una lista de condiciones y se detiene en la primera que cumpla la condición
- Se puede utilizar en cualquier sentencia que permita una expresión válida (SELECT, UPDATE, DELETE y SET, cláusulas IN, WHERE, ORDER BY, etc)
- No se puede usar para controlar el flujo de ejecución de bloques de instrucciones, funciones, procedimientos almacenados e instrucciones SQL
- Tiene 2 formatos

Recuperación de datos

Función COALESCE

- Forma abreviada de un tipo especial de CASE:

CASE

 WHEN (expr1 IS NOT NULL) THEN expr1

 WHEN (expr2 IS NOT NULL) THEN expr2

 ...

 ELSE exprN

END

COALESCE (expr1, expr2 [, exprN])

Recuperación de datos

Función IFNULL

- Reemplaza NULL con el valor especificado

IFNULL (expresión , valor)

Recuperación de datos

Función NULLIF

- Devuelve NULL si las 2 expresiones son iguales

NULLIF (expresión1, expresión2)

Modificación de datos

Sentencia UPDATE:

- Las filas a modificar se especifican en la cláusula WHERE
- Los valores nuevos se especifican con la cláusula SET
- Se pueden cambiar los datos de una tabla a la vez
- Los valores ingresados deben ser del mismo tipo de las columnas a modificar y no deben violar las restricciones
- Se pueden usar expresiones basadas en columnas de la tabla u otras tablas, variables y constantes

Borrado de datos

Para borrar filas de una tabla se puede usar:

- DELETE:
 - Borra **una o más** filas de una tabla
 - Se pueden limitar las filas a borrar (WHERE)
 - Las filas borradas se registran en el registro de transacciones

Borrado de datos


Para borrar filas de una tabla se puede usar:

- TRUNCATE:
 - Borra **todas** las filas de una tabla
 - Si hay una columna AUTO_INCREMENT, resetea la semilla al valor inicial
 - Se ejecuta mucho más rápido que DELETE ya que no se registra en el registro de transacciones



Múltiples tablas



- Combinación de múltiples tablas
 - Combinación de varios resultsets
 - Creación de tabla desde un resultset
- 

Combinación de múltiples tablas

- Para producir un resultado que incorpore filas de 2 o más tablas se utiliza el operador JOIN:

SELECT <lista_selección>

FROM T1 [INNER | LEFT | RIGHT | OUTER] JOIN T2

ON <condición>

- JOIN: especifica las tablas que se combinan y cómo
- ON: especifica las columnas en común de las tablas

Combinación de múltiples tablas

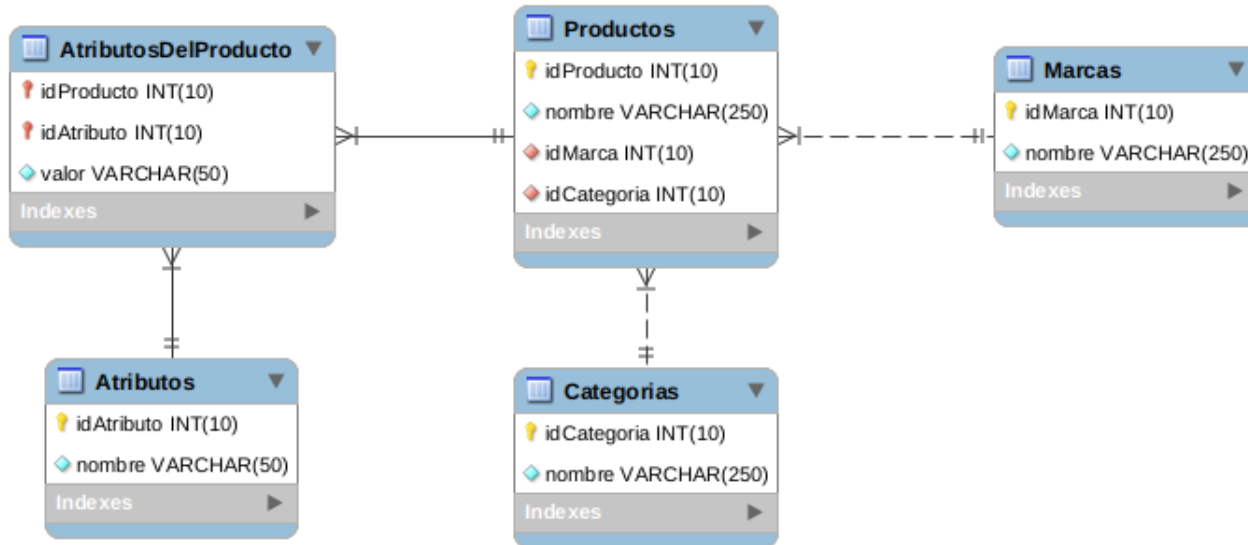
Consideraciones:

- Especificar la condición según las PKs y FKs
- Si la clave es compuesta, **se deben referenciar todas** las columnas luego de la cláusula ON
- Si 2 o más tablas tienen los mismos nombres de columnas, se debe anteponer el nombre de la tabla
- Limitar el número de tablas ya que aumenta el tiempo de proceso de la consulta

Combinación de múltiples tablas

Cambio en el problema:

- Los productos también tienen una marca, una categoría y una serie de atributos



Combinación de múltiples tablas

INNER JOIN:

- Combina tablas comparando valores en las columnas comunes. Devuelve las filas que cumplan la condición
- Consideraciones:
 - Es el JOIN por defecto (se puede omitir INNER)
 - Si se incluye WHERE se restringe la cantidad de filas devueltas
 - Evitar el uso de NULL como condición

Combinación de múltiples tablas

- Consideraciones:
 - No se garantiza un orden en el resultado
 - $A \text{ JOIN } B \text{ JOIN } C = (A \text{ JOIN } B) \text{ JOIN } C$
 - $A \text{ JOIN } B \text{ JOIN } C = (B \text{ JOIN } C) \text{ JOIN } A$
 - Las tablas deben estar relacionadas con FKs
 - Los JOINS son conmutativos

Combinación de múltiples tablas

OUTER JOIN:

- Hay 2 tipos:
 - **LEFT:** ídem INNER JOIN más las filas de la tabla izquierda que no cumplen la condición del JOIN
 - **RIGHT:** ídem INNER JOIN más las filas de la tabla derecha que no cumplen la condición del JOIN

Combinación de múltiples tablas

OUTER JOIN:

- Consideraciones:
 - $A \text{ LEFT JOIN } B = B \text{ RIGHT JOIN } A$
 - Evitar el uso de NULL como condición
 - Usarlo entre 2 tablas preferentemente
 - Se puede omitir OUTER

Combinación de múltiples tablas

CROSS JOIN:

- Realiza el producto cartesiano de 2 tablas (no requiere columnas en común)
- Generalmente se lo usa para poblar rápidamente tablas
- $\text{Card}(A \text{ CROSS JOIN } B) = \text{Card}(A) * \text{Card}(B)$

Combinación de múltiples tablas

Cambio en el diseño:

- Para los atributos del producto, en lugar de las tablas **Atributos** y **AtributosDelProducto** se puede agregar a la tabla **Productos** una columna tipo JSON



Combinación de múltiples tablas

Sintaxis JSON:

- Una colección desordenada de pares clave:valor, entre {}
- Una lista ordenada de valores (vector), entre []
- Ejemplos:
 - {}
 - {"user": "Sammy", "online": true, "followers": 987}
 - []
 - [1, 2, true, false]

Combinación de múltiples tablas

Sintaxis JSON:

- Un valor puede ser:
 - Una cadena entre comillas dobles
 - Un número
 - true/false
 - null
 - Otro objeto JSON
 - Un vector

Combinación de múltiples tablas

Sintaxis JSON (objetos anidados):

```
{
  "sammy" : {
    "username" : "SammyShark",
    "followers" : 987
  },
  "jesse" : {
    "username" : "JesseOctopus",
    "followers" : 432
  }
}
```

Combinación de múltiples tablas

Sintaxis JSON (vectores anidados):

```
{
  "username" : "SammyShark",
  "websites" : [
    {
      "desc" : "work", "URL" : "https://digit.com/"
    },
    {
      "desc" : "tutorial", "URL" : "https://digit.com/t"
    }
  ]
}
```

Combinación de múltiples tablas

Sintaxis JSON (atributos multivaluados):

```
{  
  "username" : "SammyShark",  
  "followers" : 987,  
  "websites" : ["https://www.sitio1.com/",  
                "https://www.sitio2.com/",  
                "https://www.sitio3.com/",  
                "https://www.sitio4.com/"  
              ]  
}
```

Combinación de múltiples tablas

Inserción en una columna JSON:

- Armar el objeto JSON manualmente
- Función **JSON_OBJECT()**: evalúa una lista de pares clave:valor y devuelve un objeto JSON con los mismos
- Función **JSON_MERGE_PRESERVE()**: combina 2 o más documentos JSON y devuelve el resultado combinado

Combinación de múltiples tablas

Consulta de una columna JSON:

- Función **JSON_EXTRACT()**: devuelve los datos de un objeto JSON
- Operador ->
- En ambos casos se debe especificar el camino, el cual siempre empieza con \$

Combinación de múltiples tablas

Camino JSON:

```
{  
  "a": 1,  
  "b": 2,  
  "c": [3, 4],  
  "d": {  
    "e": 5,  
    "f": 6  
  }  
}
```

Camino JSON:

`$.a`: devuelve 1

`$.c`: devuelve [3, 4]

`$.c[0]`: devuelve 3

`$.c[1]`: devuelve 4

`$.d.e`: devuelve 5

Combinación de múltiples tablas

Modificación de una columna JSON:

- Función **JSON_REPLACE()**: modifica el valor de la clave sólo si existe la misma
- Función **JSON_INSERT()**: agrega una clave y su valor sólo si la misma no existe
- Función **JSON_SET()**: si existe la clave, la modifica. Si no existe la agrega

Combinación de múltiples tablas

Borrado en una columna JSON:

- Función **JSON_REMOVE()**: si existe la clave, la borra junto con su valor

Combinación de múltiples tablas

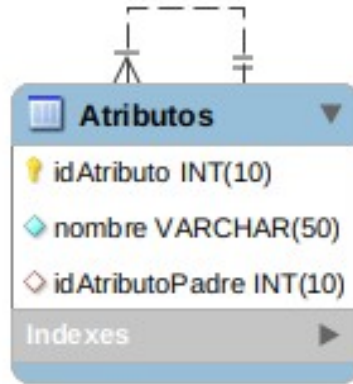
Extracción de datos de una columna JSON:

- Función **JSON_TABLE()**: extrae datos de un objeto JSON como un conjunto de filas y columnas

Combinación de múltiples tablas

Cambio en el problema:

- Ahora un atributo puede tener, a su vez, otros atributos
- Diseño relacional: hacer el JOIN de la tabla **Atributos** con sí misma



Combinación de múltiples tablas

JOIN de una tabla con sí misma:

- Al hacer el JOIN de la tabla **Atributos** con sí misma:
 - Se quiere mostrar de cada atributo su nombre y el de su padre
 - Hay que pensar como si hubiera 2 tablas iguales con el mismo nombre, para lo cual se deberán usar alias para referenciar “las 2 tablas”

Combinación de múltiples tablas

JOIN de una tabla con sí misma:

Atributos (A1)		
idAtributo	nombre	idAtributoPadre
1	Pantalla	NULL
2	Resolución	NULL
3	Puertos	NULL
4	Parlantes	NULL
5	HDMI	3

Atributos (A2)		
idAtributo	nombre	idAtributoPadre
1	Pantalla	NULL
2	Resolución	NULL
3	Puertos	NULL
4	Parlantes	NULL
5	HDMI	3

Combinación de múltiples tablas

JOIN de una tabla con sí misma:

- ¿Cómo sería la consulta para obtener esta salida?

idProducto	nombre	Marca	Categoría	Atributo	Valor
1	Prime	Samsung	Televisión	Pantalla	50 pulgadas
1	Prime	Samsung	Televisión	Resolución	2048 x 1152 pixels
2	Prime	Samsung	Televisión	Puertos	NULL
2	Prime	Samsung	Televisión	HDMI	1
...

Combinación de múltiples tablas

Atributos con atributos en diseño JSON:

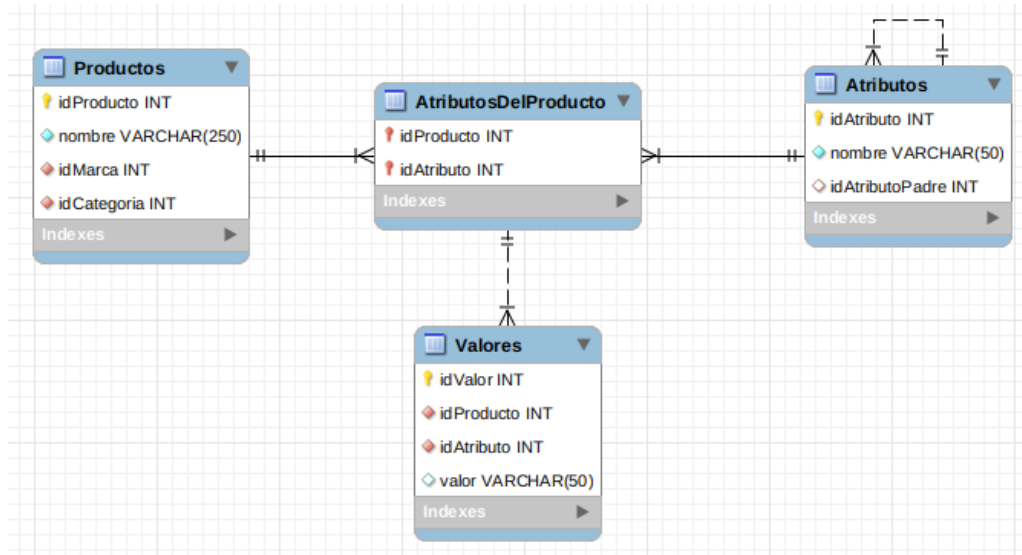
- ¿Y la consulta para el diseño JSON?

idProducto	nombre	Marca	Categoría	Atributo	Valor
1	Prime	Samsung	Televisión	Pantalla	50 pulgadas
1	Prime	Samsung	Televisión	Resolución	2048 x 1152 pixels
2	Prime	Samsung	Televisión	Puertos	NULL
2	Prime	Samsung	Televisión	HDMI	1
...

Combinación de múltiples tablas

Cambio en el problema:

- Ahora un atributo puede tener, a su vez, varios valores:



Combinación de varios resultsets

Resultset:

- Conjunto de filas de una BD, más los metadatos sobre la consulta (nombres de columnas, tipos de datos, etc)
- Para combinar los resultados de 2 o más sentencias SELECT en un solo resultset se usa el operador UNION:

```
SELECT ... UNION [ALL | DISTINCT] SELECT ...
```

Combinación de varios resultsets

Consideraciones:

- Los nombres de las columnas de la combinación son los nombres del primer SELECT
- Los resultados deben tener el mismo tipo de datos, número y orden de columnas en la lista de selección
- Automáticamente se remueven las filas duplicadas, a menos que se emplee la cláusula ALL

Combinación de varios resultsets

Consideraciones:

- Si se quieren ordenar sólo los resultados de uno de los SELECT, el mismo debe ir entre paréntesis
- El uso de ORDER BY para uno solo de los SELECT no implica un orden para la combinación final
 - Por defecto UNION genera una salida desordenada

Combinación de varios resultsets

Consideraciones:

- Para ordenar toda la combinación, cada SELECT debe ir entre paréntesis, y la cláusula ORDER BY al final:

```
(SELECT ColA, ColB FROM Tabla1)  
UNION  
(SELECT ColC, ColD FROM Tabla2)  
ORDER BY ColA;
```

Combinación de varios resultsets

Consideraciones:

- En el caso anterior, ORDER BY no puede llevar el nombre de una tabla (sólo la columna). Para especificar una tabla se debe proporcionar un alias a toda la unión:

```
SELECT ColB FROM  
( (SELECT ColA, ColB FROM Tabla1)  
  UNION  
  (SELECT ColC, ColD FROM Tabla2) ) AS T  
ORDER BY T.ColA;
```

Creación de tabla desde un resultset

- Se puede crear una nueva tabla en base al resultado de una consulta usando CREATE SELECT
- Al crear la tabla se puede especificar que la misma sea temporal
- También se puede clonar la estructura de una tabla en otra

Técnicas avanzadas



- Subconsultas
- Subconsultas correlacionadas
- Modificación de datos

Subconsultas

Subconsulta:

- Consulta anidada dentro de otra

```
SELECT * FROM t1  
WHERE col1 = (SELECT col1 FROM t2);
```

- `SELECT * FROM t1`: consulta externa
- `(SELECT col1 FROM t2)`: subconsulta (anidada dentro de la consulta externa)

Subconsultas

Consideraciones:

- Siempre van entre paréntesis
- Se pueden anidar dentro de otras
- Pueden devolver un valor, una fila/columna o una tabla
- Pueden contener casi cualquier cláusula
- No pueden modificar y seleccionar la misma tabla

Subconsultas

Ventajas:

- Permiten aislar cada parte de una sentencia
- Alternativas para realizar operaciones que requerirían uniones y combinaciones (JOIN) complejas. Ejemplo: Mostrar las filas de T1 donde el valor de col1 sea igual al máximo valor de col2 en T2:

```
SELECT * FROM T1  
WHERE col1 = (SELECT MAX(col2) FROM T2);
```

Subconsultas

EXISTS y NOT EXISTS:

- Si una subconsulta devuelve al menos una fila, EXISTS devuelve TRUE y NOT EXISTS FALSE

Subconsultas correlacionadas

Subconsulta correlacionada:

- Subconsulta que utiliza los datos de la consulta externa (depende de la misma)
- Una subconsulta se puede ejecutar sola, no depende de la consulta externa. Una subconsulta correlacionada no se puede ejecutar sola, depende de la consulta externa.
- Se evalúa una vez por cada fila de la consulta externa

Subconsultas correlacionadas

Subconsulta correlacionada:

- Ejemplo:

```
SELECT *  
FROM T1  
WHERE col1 = (SELECT col1 FROM T2 WHERE T2.col2 = T1.col2)
```

- La subconsulta tiene una referencia a una columna de T1, aunque la cláusula FROM no la mencione. El SGBDR busca entonces fuera de la subconsulta, y la encuentra en la consulta externa

Modificación de datos

- A una tabla se le pueden insertar, borrar y/o modificar filas basadas en otras tablas
- Para la inserción se puede emplear INSERT ... SELECT:
 - Se agregan las filas que satisfagan el SELECT
 - Las columnas de la tabla destino deben tener tipos de datos compatibles a la lista de selección
 - Siempre se agregan filas a una única tabla

Modificación de datos

- Para el borrado y modificación de filas basadas en otras tablas, se puede:
 - Emplear subconsultas
 - Emplear JOINS

Resumen

- Inserción de datos
- Recuperación de datos
- Modificación de datos
- Borrado de datos
- Combinación de múltiples tablas
- Combinación de varios resultsets
- Creación de tabla desde un resultset
- Subconsultas
- Subconsultas correlacionadas
- Modificación de datos

Otros recursos

JSON:

- <https://scotch.io/tutorials/working-with-json-in-mysql>
- <https://dev.mysql.com/doc/refman/8.0/en/json.html>
- <https://ed.team/blog/como-trabajar-con-json-en-mysql>
- <http://www.mysqltutorial.org/mysql-json/>
- <https://www.sitepoint.com/use-json-data-fields-mysql-databases/>
- <https://www.compose.com/articles/mysql-for-your-json/>