Si te dijera que puedes calcular la distancia entre la Torre Eiffel en París, Francia y el Palacio de Buckingham en Londres, Inglaterra usando únicamente MySQL… ¿Me creerías? 😏

Lo creas o no, esto es completamente posible, y esto es gracias a los diferentes tipos de datos que ofrece MySQL. Seguramente has usado el clásico VARCHAR, este tipo de dato es común que lo uses para todos tus campos cuando recién aprendes bases de datos, sin embargo, a medida que vas aprendiendo te vas encontrando con nuevos tipos de datos como INTEGER, DECIMAL, DATE o incluso TIMESTAMP.

Pero… ¿Sabías que MySQL tiene tipos de datos más complejos? Ojo, el hecho de decir “complejo” no significa que sean más difíciles de usar, sino que tienen una estructura más elaborada, lo cual nos permite hacer muchas más cosas con ellos y eso es lo que vamos a ver hoy 😈.

**🧙 El tipo de dato que nos permitirá calcular distancias**

Seguramente si te preguntara qué tipo de dato puedo usar para guardar coordenadas geográficas me responderías que un tipo DECIMAL o FLOAT, ¿verdad? Esto no es del todo incorrecto, podría usar dos columnas de tipo DECIMAL y en una guardar la latitud y en otra la longitud, pero MySQL nos ofrece una mejor solución 😉.

Si quisiéramos guardar coordenadas, el mejor tipo de dato que podríamos usar es el tipo POINT, esto es básicamente un objeto que guardará tanto la latitud como la longitud en una sola columna, y nos permitirá trabajar con funciones geográficas. Este tipo de dato hace parte de las [Spatial Analysis Functions](https://dev.mysql.com/doc/refman/8.0/en/spatial-analysis-functions.html) de MySQL.

Ten en cuenta que aquí estamos guardando únicamente coordenadas geográficas, pero eso es lo único que necesitamos. ¿Qué me dices si nos ponemos manos a la obra y empezamos a calcular distancias? 😄

**🛣 ¡Midiendo distancias con MySQL!**

¡Muy bien! Ahora que ya sabes qué tipo de dato vamos a usar es hora de ponerlo en práctica. Puedes hacer esto desde cualquier cliente gráfico de bases de datos, en mi caso usaré [MySQL Workbench](https://www.mysql.com/products/workbench/).

Lo primero que debes hacer es crear una nueva base de datos y a continuación crear una tabla que contenga nuestro tipo de dato especial. En mi caso únicamente tendré las columnas id, name y coordinates:

**CREATEDATABASE** distance\_calculator CHARACTER**SET** utf8mb4 **COLLATE** utf8mb4\_general\_ci;

**USE** distance\_calculator;

**CREATETABLE** places (

**id**INT(11) **UNSIGNEDNOTNULL** AUTO\_INCREMENT,

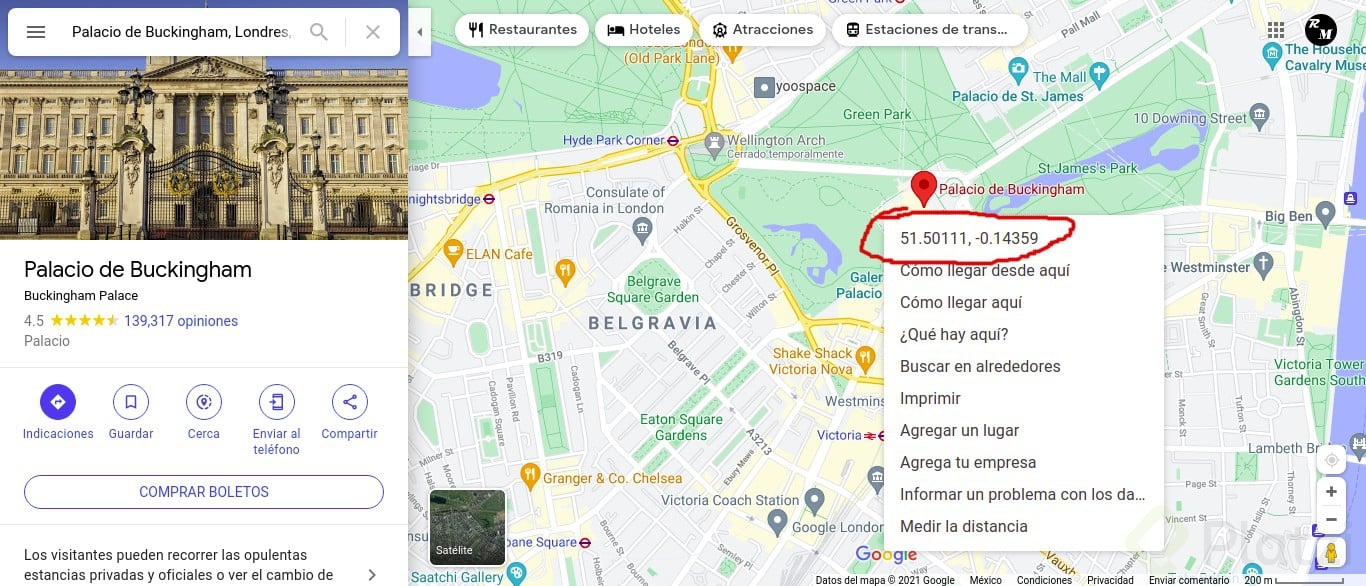
**name**VARCHAR(256) **DEFAULTNULL**,

coordinates POINT **DEFAULTNULL**,

PRIMARY **KEY** (**id**)

) **CHARSET**=utf8;

Hecho esto, solo nos queda insertar los lugares cuyas distancias queremos calcular. Para obtener las coordenadas de estos lugares podemos apoyarnos de Google Maps. Simplemente hacemos click derecho y obtendremos las coordenadas:



Como bonus, también vamos a calcular la distancia entre la Torre Eiffel y el Coliseo de Roma 😉. A continuación te dejo las sentencias SQL para insertar las coordenadas que estaremos usando:

-- Palacio de Buckingham

**INSERTINTO** places (**name**, coordinates)

**VALUES** ("Palacio de Buckingham", POINT(-0.14359, 51.50111));

-- Coliseo de Roma

**INSERTINTO** places (**name**, coordinates)

**VALUES** ("Coliseo de Roma", POINT(12.4900422, 41.8902142));

Nota que, para insertar las coordenadas, estamos haciendo uso de una función llamada POINT. Esto es muy importante, ya que se encargará de insertar correctamente nuestras coordenadas. Esta función tiene una característica particular y es que **primero debemos enviar la longitud** y luego la latitud.

Adicionalmente, ya me di a la tarea de buscar las coordenadas de la Torre Eiffel. Esta no la vamos a insertar, ya que estas son las coordenadas contra las cuales calcularemos las distancias, aunque, si tú quieres, puedes insertarlas:

* Latitud Torre Eiffel: 48.8583736
* Longitud Torre Eiffel: 2.2922926

¡Hora de calcular distancias! Para hacer esto, MySQL nos provee de una función muy interesante llamada ST\_Distance\_Sphere. Esta función comparará las coordenadas de todos los registros que estén insertados en nuestra base de datos contra las coordenadas que le pasemos y nos devolverá una nueva columna con las distancias ya calculadas, ¡probémoslo!

**SELECT**

**name**,

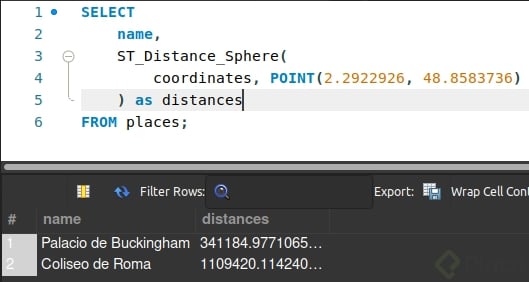
ST\_Distance\_Sphere(

coordinates, POINT(2.2922926, 48.8583736)

) **as** distances

**FROM** places;

Como puedes ver, en el primer parámetro de la función ponemos **el nombre de la columna que contiene las coordenadas de cada lugar** (en este caso es coordinates) y en el segundo parámetro (de nuevo usando la función POINT) ponemos las coordenadas del lugar contra el cual estamos calculando la distancia. En este caso, las coordenadas de la Torre Eiffel. Recuerda que esta función nos genera una nueva columna que contiene las distancias ya calculadas, por lo que mediante el uso de as estamos renombrando esa nueva columna a distances.



¡Con eso hemos calculado las distancias! Pero esta función nos devuelve el resultado en metros, ¿podemos convertirlos a kilómetros? ¡Por su puesto! Solo debemos hacer una pequeña operación matemática:

**SELECT**

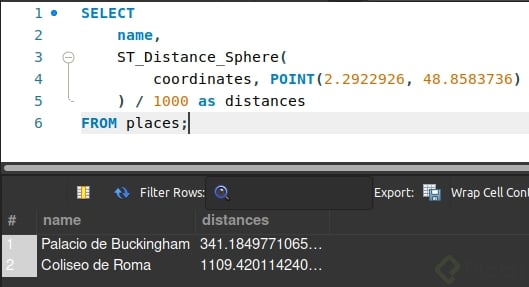
**name**,

ST\_Distance\_Sphere(

coordinates, POINT(2.2922926, 48.8583736)

) / 1000**as** distances

**FROM** places;



Únicamente dividimos el resultado de la función entre 1000 para convertirlo a Kilómetros, y de esta forma podemos ver que MySQL ya nos está dando los resultados en Kilómetros 😈.

Entonces, podemos ver que desde la Torre Eiffel hasta el Palacio de Buckingham hay un total de 341.18 Kilómetros y de la Torre Eiffel hasta el Coliseo de Roma hay 1,109.42 Kilómetros 😉.

¡Hemos calculado la distancia usando únicamente MySQL! Esto lo podemos hacer con cualquier lugar que desees, y todo desde tu motor de bases de datos 😄.

**❗ Consideraciones al usar Spatial Analysis Functions en bases de datos**

¡Un momento! Sé que ya tienes el hype por lo que acabamos de hacer, pero debes tener en cuenta estas consideraciones al hacer uso de estas funciones.

**🌎 ¡La Tierra no es completamente redonda!**

Recuerda que la Tierra **no es completamente redonda**, sino que es una esfera desperfecta, eso significa que el radio de la Tierra puede cambiar en diferentes puntos. Esta función usa un radio por defecto de 6,370,986 metros, pero tú puedes cambiarlo pasándole un tercer parámetro con el valor del radio que quieras usar **en metros**.

Por ejemplo, podrías elegir usar el radio ecuatorial:

**SELECT**

**name**,

ST\_Distance\_Sphere(

coordinates, POINT(2.2922926, 48.8583736), 6378000

) / 1000**as** distances

**FROM** places;

**⚙ Puedes usarlo en producción, pero…**

Las Spatial Analysis Functions fueron agregadas en MySQL, MariaDB y algunos otros motores de bases de datos, pero en algunos otros están disponibles como extensiones.

Por ejemplo, si planeas usar SQLite para hacer testing de Spatial Analysis Functions, entonces necesitarás descargar la respectiva extensión de SQLite para usarlas, de lo contrario, al intentar ejecutar estas funciones obtendrás errores.

Solo debes tener en cuenta que tu motor de bases de datos de desarrollo y tu motor de bases de datos de producción las soporten, recuerda que muchas veces en los casos de hosting compartido no tienes la posibilidad de elegir tu motor y versión de base de datos 👀.

**👩‍🎓 ¡Es tu turno!**

¡Ahora que ya sabes cómo usar estas funciones y tipos de datos complejos es tu turno de ponerte a prueba! Aquí simplemente calculamos las distancias entre dos lugares, pero podemos hacer cosas más increíbles.

Algunas ideas de lo que podemos hacer con estas funciones:

* **Determinar si un lugar está dentro de un área circular**: Imagina que tienes un restaurante que entrega comida a domicilio solo a 4 kilómetros a la redonda. ¿Cómo determinarías si la ubicación de un cliente está dentro de estos 4 kilómetros usando Spatial Analysis Functions? *Spoiler: lo puedes hacer únicamente midiendo distancias* 😉.
* **Consultar los lugares disponibles dentro de un área**: Ahora supongamos que tienes diferentes restaurantes en una ciudad, y en tu sitio web ofreces un mapa desde el cual puedes visualizar todos los restaurantes que están disponibles dentro del área visible de ese mapa. ¿Cómo filtrarías los restaurantes visibles en un área usando estas funciones? *Spoiler: podrías usar la función POLYGON y la función ST\_CONTAINS para lograrlo* 😉
* **Verificar cuál lugar me queda más cerca:** Ponte en el caso de un cliente que quiere ir a uno de tus restaurantes, pero no sabe cuál le queda más cerca. ¿Cómo determinarías cuál restaurante le queda más cerca con ayuda de MySQL? *Spoiler: solo necesitas medir y comparar distancias* 😉.

Como puedes ver, medir distancias y realizar este tipo de operaciones con MySQL es muy fácil y tiene casos de usos reales. Esta es la importancia de conocer los tipos de datos de los que disponemos en nuestro motor de bases de datos y no usar VARCHAR para todo 😉.

¡Te reto a generar las consultas necesarias para cada uno de los casos de uso que te mencioné arriba! Recuerda que tenemos el [Curso de SQL y MySQL](https://platzi.com/clases/sql-mysql/) para que puedas profundizar y aprender mucho más sobre este maravilloso gestor de bases de datos, te aseguro que aprenderás algo nuevo, y por eso #NuncaParesDeAprender 😄.

Leyendo un poco mas encontre que hay dos maneras de calcular distancia entre dos puntos:

**La formula de Haversine:**  
Es muy precisa y para calcularla necesita latitud y longitud de los dos puntos a medir. se usa mucho en astronomia.

**a** = sin²(φB - φA/2) + cos φA \* cos φB \* sin²(λB - λA/2)