**Ejercicio 2**

Solución teórica de problemas de rendimiento de consultas.  
Dado un proyecto que utiliza una BD relacional para almacenar datos, tenemos una tabla  
que ha aumentado mucho su volumen de registros y las consultas empiezan a ir cada vez  
más lentas. Al analizar el caso nos encontramos principalmente con dos tipos de  
peticiones:

**● Desde la web del proyecto se pide regularmente acceder a los datos filtrando por  
las columnas A y B.**

En este caso, sin tener más información al respecto, habría que revisar los índices creados en la tabla.

Si las columnas A y B se consultan independientemente crearíamos índices para cada una de ellas

ALTER TABLE table\_name ADD index table\_name\_A (A);

ALTER TABLE table\_name ADD index table\_name\_B (B);  
  
Si además se consultan simultáneamente, revisaríamos el orden en que se consulta ej:  
SELECT fields FROM table\_name WHERE A=? AND B=?  
  
Y en consecuencia crearíamos un índice para ambos campos

ALTER TABLE table\_name ADD index table\_name\_B (A,B);  
  
Referencia: <https://dev.mysql.com/doc/refman/8.0/en/multiple-column-indexes.html>  
  
Estoy planteando el caso más básico, si las consultas que realizamos son sobre campos numéricos y/o de texto (haciendo consultas de comparación sencilla). En caso de que habláramos de otro tipo de consultas, como por ejemplo búsquedas de texto parciales plantearía crear índices de texto (FULLTEXT en mysql) que requieren el uso de funciones de búsqueda especiales en la consulta.

Referencia: https://dev.mysql.com/doc/refman/8.0/en/fulltext-search.html

Si habláramos aun de mayor complejidad de búsquedas de texto, habría que contemplar la posibilidad de usar un indexador como Sphinx o ElasticSearch.

**● Una vez por día se lanza desde una tarea cron un proceso en batch con consultas  
bastante intensivas.**

Sin cambiar de tecnología, explica cómo analizarías el caso y cuáles serían tus propuestas  
de mejora al problema.  
  
Igualmente dependería que las particularidades del caso, pero como ejercicio mental voy a platear la situación en la que sobre la tabla realizan diversas consultas para obtener los datos del día y obtener unas estadísticas diarias. En tal caso, por ejemplo, es conveniente disponer de una tabla aparte a la tabla origen, donde las queries se produzcan sobre un numero limitado de registros, en este caso, los del día.  
  
En mysql/maria DB no se dispone de ello y se deben usar tablas temporales, pero en Postgres existen las vistas materializadas.  
  
  
  
En el ejemplo que planteaba:  
  
CREATE MATERIALIZED VIEW pedidos\_diarios AS

SELECT \* FROM pedidos  
  
WHERE fecha < CURRENT\_DATE AND fecha > CURRENT\_DATE + INTERVAL '1 day';  
  
  
Incluso podemos aprovechar en la vista materializada para crear campos denormalizados, como podrían ser sumas de datos agrupados. Así no es necesario calcularlos en las consiguientes queries que realizará el proceso.

CREATE MATERIALIZED VIEW pedidos\_diarios AS

SELECT pedidos.\*, count(id\_producto as productos) as num\_productos, sum(productos.importe) as importe  
FROM pedidos inner join productos ON productos.id\_pedido = pedidos.id  
WHERE fecha < CURRENT\_DATE AND fecha > CURRENT\_DATE + INTERVAL '1 day'

GROUP BY pedidos.id

podemos crear índices sobre la vista materializada si los campos son usados para comparaciones en las queries del proceso.  
  
  
  
En el proceso nocturno debemos refrescar la vista materializada.  
  
REFRESH MATERIALIZED VIEW pedidos\_diarios;

Otras técnicas conocidas podrían ser mantener las denormalizaciones en tiempo real.  
  
Esto se puede conseguir con triggers, pero no soy muy partidiario de los mismos porque considero que tiene un efecto muy perverso que haya cierta lógica de negocio en la BBDD que invisible a nuestro repositorio de código. Dicho en llano: “Luego nadie sabe por qué pasan las cosas”.  
  
O mantener esas denormalizaciones realizando los cálculos ente cualquier creación, edición del modelo etc. Por ejemplo en el caso planteado, mantener num\_productos del pedido cada vez que al pedido se le asocia un producto o se elimina. En casos muy extremos conviene incluso que estos cálculos se hagan en tiempo real asíncronamente, lanzando una petición a alguna cola de trabajo.