



# Optimización de BBDD

Bernat Costa



## Factores que afectan al rendimiento

- ▶ Las consultas deben estar bien escritas y optimizadas
- ▶ Técnicas para mejorar el rendimiento
  - ▶ Índices

# Que no hacer en una consulta

Evitar el Select \*

Filtrar con el Where lo máximo posible

Ordenar condiciones en el where.

Evitar tablas temporales

Pon siempre PK a tus tablas

No uses cursores, para recorrer una tabla

Revisa los Joins

# Evitar el \* en el Select



Si queremos recuperar los emails de los clientes, no hace falta que me traiga toda la tabla, solo la columna email.



Se ahorra memoria y tiempo de procesamiento cuando tienes muchos datos.

# Filtrar Bien con el where

Si podemos filtrar, filtramos. Como menos registros nos traigamos, más rápido ira.

- Menos consumo de memoria
- Menos consumo de red.

Todo lo que podamos filtrar desde el SQL Server, mejor.

- No tiene sentido filtrar luego en un excel o una aplicación o ponernos a filtrar a mano.

# Orden de las Condiciones en el Where



Pensar bien la condición y el orden de las condiciones



Si tienes un Or, y una de las consultas SIEMPRE da true, ponerla primero.



Ejemplo:

```
select * from sales where order_id <> 309317338 or  
dbo.funcionCompleja()=1 --muy rapido
```

```
select top 100 * from sales where  
dbo.funcionCompleja()=1 or order_id <> 309317338 -- muy  
lento
```

# Crear la función compleja

```
CREATE or alter FUNCTION funcionCompleja() RETURNS int
BEGIN
    DECLARE @dtStart DATETIME=getutcdate()
    WHILE datediff(second,@dtStart,getutcdate()) < 1
    BEGIN
        set @dtStart = @dtStart
    END
    RETURN 1;
END
```



# Evitar tablas temporales





# Pon siempre Primary Keys

Las PK son una forma de indexar las tablas.

Pon SIEMPRE UNA PK en tu tabla.

## Los cursores, Fuera de la bbdd

---

Para recorrer las tablas, es mejor hacerlo desde la aplicación.

---

No tiene sentido hacerlo desde el Servidor de BBDD y están muy mal optimizados.

# Poner bien los JOINS

Los joins, pueden darnos muchos problemas si no están bien ordenados los datos.

Revisa tus joins, y asegurate que tienes indices en las columnas que pones en los ON de los joins.

¿Qué es un  
índice?



# ¿Para qué sirven?

01

Los índices son una forma de ordenar nuestras tablas

02

Sirven para hacer búsquedas y para ordenar rápidamente nuestras tablas.

03

Podremos crearlos bajo nuestro criterio según las necesidades de uso de nuestra bbdd

# Tipos de INdice

**Clustered o  
agrupados**



Se reordenan los datos de la  
tabla.



Un diccionario.

**NONCLUSTERED o no  
agrupados**



Se crea un indice de nuestra  
tabla con punteros a las partes  
donde están nuestros datos



El indice de un libro

# CLUSTERED INDEX

- ▶ La tabla se reordena por una serie de campos que tu decidas.
- ▶ Solo puede haber UNO en cada tabla.
- ▶ Funciona igual que un diccionario o listín telefónico.





# Cómo funciona un índice CLUSTERED

	name	phone	email
1	Buckminster Spence	(963) 238-7843	hymenaeos@aol.edu
2	Cameron Murray	(404) 797-2941	neque.sed.sem@aol.ca
3	Carson Tanner	(578) 381-9312	est.nunc.laoreet@outlook.com
4	Caryn Cooley	(972) 731-3133	amet.ultricies.sem@protonmail.co.uk
5	Dacey Jenkins	(544) 875-4125	taciti.sociosqu@aol.edu
6	Dai Stewart	(346) 772-2124	magnis@icloud.ca
7	Dean Foreman	1-715-785-2200	interdum.nunc@protonmail.com
8	Desirae Sherman	(820) 605-6271	semper.rutrum@protonmail.edu
9	Gabriel Whitney	(713) 515-9833	dolor.tempus@aol.ca
10	Holmes Christian	1-750-387-1145	in.aliquet.lobortis@outlook.com
11	Keegan Shaffer	1-580-683-1164	quisque.ornare.tortor@google.net
12	Lamar Burton	(417) 212-5852	iaculis.odio.nam@protonmail.com
13	Louis Berger	(328) 882-5237	nunc.mauris.morbi@google.ca
14	Virginia Marsh	1-684-531-3025	ligula@icloud.com
15	Zeph Wood	1-775-774-9877	conubia.nostra@hotmail.edu



pagina	Letra
1	B
2	C
5	D
9	G
10	H
11	K
12	L
14	V
15	Z

# NON CLUSTERED INDEX

---

Cuando queremos más de un índice sobre una misma tabla, no podemos ordenar la tabla por más de un índice.

---

Se crea una copia de la columna del índice y se guarda la posición. Se guarda reordenada, y luego se sacan "las iniciales".

---

Podemos crear tantos como queramos. Pero sin abusar.

# No abusar de los índices

Los índices relentizan la inserción y modificación de datos.

No hay que abusar

Ocupan espacio en disco.

# ¿Cómo creamos un Índice?

**CREATE CLUSTERED/NONCLUSTERED INDEX**  
**nombredeIndice**

**ON tabla(columna1,columna2...)**

Podemos añadirle

**INCLUDE (columna3, columna4,...)**

para que nos lo tenga en cuenta al hacer el select de  
varios campos

Las columnas del indice las podemos decidir nosotros. Puede ser 1 o n

Como vemos los indices de una tabla?

**EXEC**  
**sp\_helpindex 'nombredeletabla'**

¿Y para borrar un índice?

**DROP INDEX indx\_totalx ON  
sales**

# Importemos una BBDD grande

- ▶ (<https://eforexcel.com/wp/downloads-18-sample-csv-files-data-sets-for-testing-sales/>) Bajate el CSV de 5M de registros del teams o de la web. (Tendrás que quitarle la cabecera del de la web)
- ▶ Script bigdata.sql en Teams para importar en CSV a una tabla.
- ▶ La web del proyecto gutenbergtambién tiene un CSV de libros bastante grande.
  - ▶ <https://www.gutenberg.org/cache/epub/feeds/>
- ▶ Intenta importarlo en otra BBDD modificando el script bigdata.sql.



# Ejercicio

- ▶ Mira los índices de la tabla grande
- ▶ Prueba de hacer consultas sobre esa tabla
  - ▶ Pedidos de un país en concreto
  - ▶ Importes de mas de 10.000
  - ▶ Filtrar por un OrderID
- ▶ En azure data Studio, aprieta en Explain en lugar de RUN para ver el Plan de Ejecución de cada consulta.
- ▶ Crea un Índice, y mira en el plan de Ejecución si se usa
- ▶ Prueba con un índice con include para ver si podemos ver que se use nuestro índice en el plan de ejecución.
- ▶ Prueba con la funcion compleja el orden de las condiciones en el where con una tabla grande.

# Estadísticas de uso de IO

---

Para verlas, se pueden activar con SET STATISTICS IO ON

---

SET STATISTICS IO OFF para desactivarlas.

---

Cuando lanzas una consulta, en message te salen datos relativos a los costes de esa consulta. Sirve para comparar por ejemplo lecturas a disco

```
SET STATISTICS IO on  
use bigdata  
select country from sales where coun
```

---

Messages	Query Plan	Top Operators
45	<u>Started executing query</u> (27035 rows affected) Table 'sales'. Scan count (1 row affected) Total execution time	

# Herramientas Para Analizar la Carga de una BBDD

- ▶ SQL Server Profiler
- ▶ Herramienta que viene con el SQL Management Studio.
- ▶ Nos permite ver el log de TODO LO QUE PASA en nuestro servidor en Tiempo real.

(host)]

Window Help

TextData	ApplicationName	NTUserName	LoginName	CPU	...
	SQLQueryStress		sa		
sp_reset_connection	SQLQueryStress		sa		
sp_reset_connection	SQLQueryStress		sa		
network protocol: TCP/IP set quo...	SQLQueryStress		sa		
ect Country from sales where ...	SQLQueryStress		sa		
ATISTICS IO ON;SET STATISTICS ...	SQLQueryStress		sa		
ork protocol: TCP/IP set quo...	SQLQueryStress		sa		
ISTICS IO ON;SET STATISTICS ...	SQLQueryStress		sa		
ISTICS IO ON;SET STATISTICS ...	SQLQueryStress		sa		
ISTICS IO ON;SET STATISTICS ...	SQLQueryStress		sa		
TICS IO ON;SET STATISTICS ...	SQLQueryStress		sa		
ICS IO ON;SET STATISTICS ...	SQLQueryStress		sa		
entry from sales where ...	SQLQueryStress		sa		
entry from sales where ...	SQLQueryStress		sa		
entry from sales where ...	SQLQueryStress		sa		
entry from sales where ...	SQLQueryStress		sa		
	SQLQueryStress		sa		
et_connection	SQLQueryStress		sa		
	SQLQueryStress		sa		
_reset_connection	SQLQueryStress		sa		
ect Country from sales where ...	SQLQueryStress		sa		
network protocol: TCP/IP set quo...	SQLQueryStress		sa		
network protocol: TCP/IP set quo...	SQLQueryStress		sa		
SET STATISTICS IO ON;SET STATISTICS ...	SQLQueryStress		sa		
SET STATISTICS IO ON;SET STATISTICS ...	SQLQueryStress		sa		
SET STATISTICS IO ON;SET STATISTICS ...	SQLQueryStress		sa		
SET STATISTICS IO ON;SET STATISTICS ...	SQLQueryStress		sa		

# HERRAMIENTA PARA ESTRESAR UN SERVIDOR

## SQLQueryStress



<https://www.microsoft.com/es-es/p/sqlquerystress>



Nos permite darle una consulta y lanzarla muchas veces.



Con esta herramienta, podemos simular cargas reales de un servidor



Le indicamos el número de interacciones y de hilos



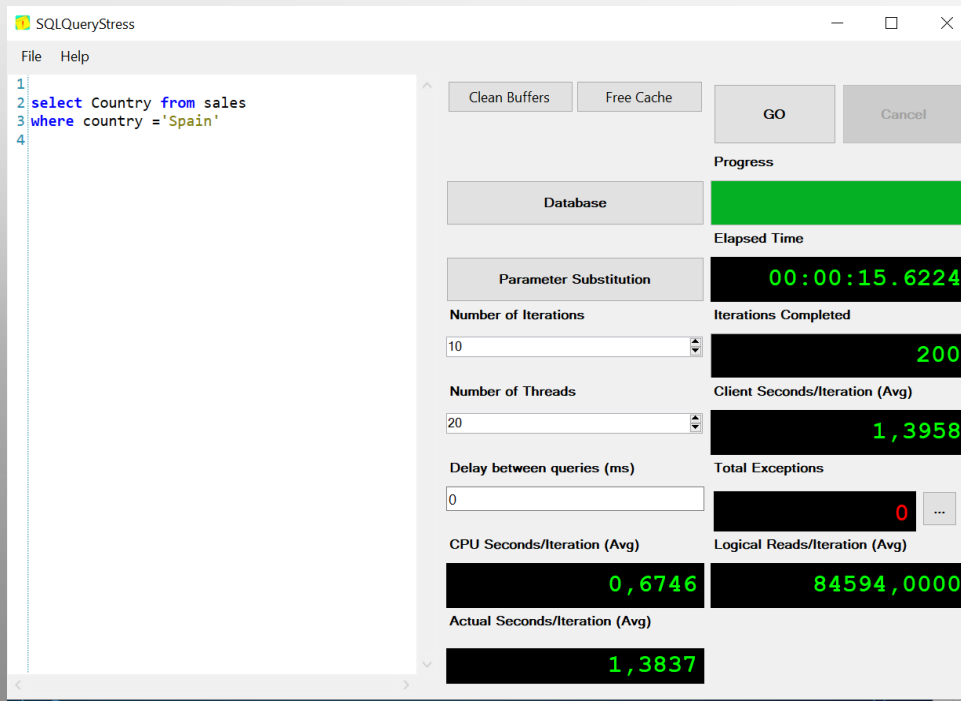
Le pasamos una consulta y nos conectamos a la bbdd.



Podemos comprobar la eficiencia de varios servidores o de distintas configuraciones

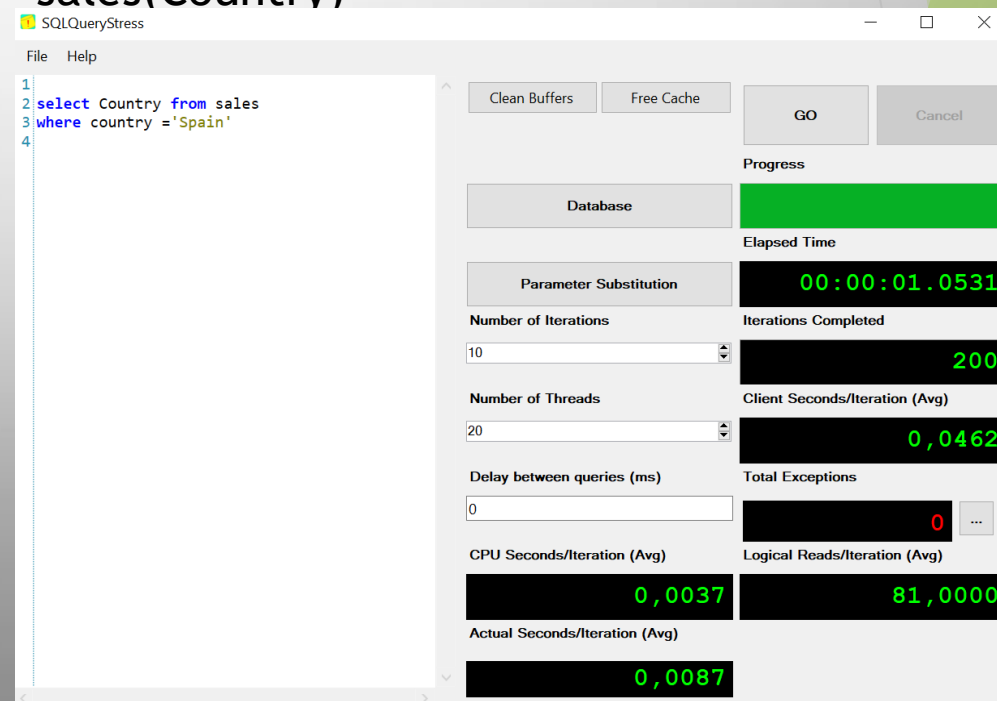
# SQLQueryStress

Sin  
INDICE



Con INDICE

create nonclustered index indx\_country on  
sales(Country)



# La importancia de los tipos de datos

- Los tipos de datos de nuestra tabla serán determinantes para la optimización
- No es lo mismo un VARCHAR(100), que un CHAR(100) que un NVARCHAR(100)
- O un INT, que un TinyINT que un Decimal(18,2)



A Newton's cradle with several silver spheres hanging from thin wires. The background is a mix of green and grey geometric shapes.

## El tipo de datos, influirá en el tamaño

### ► Cadenas:

- CHAR(100) -> reserva el espacio para 100 caracteres
- VARCHAR(100) -> reserva el espacio solo para los caracteres usados.
- NVARCHAR(100) -> Reserva el doble de espacio que un varchar



# Enteros

Data type	Range	Storage
bigint	$-2^{63}$ (-9,223,372,036,854,775,808) to $2^{63}-1$ (9,223,372,036,854,775,807)	8 Bytes
int	$-2^{31}$ (-2,147,483,648) to $2^{31}-1$ (2,147,483,647)	4 Bytes
smallint	$-2^{15}$ (-32,768) to $2^{15}-1$ (32,767)	2 Bytes
tinyint	0 to 255	1 Byte

# Ajustar los tipos de datos

- ▶ Como más ajustemos los tipos de datos, menor será el uso de disco.
- ▶ Con tablas con 5 Millones de registros, la diferencia puede ser notable
- ▶ No solo ganamos en espacio en disco. Los costes de un Table SCAN se disparan si la tabla "pesa" más en disco ya que hay que leer mas info.
- ▶ Procedimiento para ver cuando ocupa una tabla en disco:
  - ▶ EXEC sp\_spaceused 'sales'

# Ejercicio

---

Con el script bigdata, creemos las tablas salestinny y salesbig.

---

Modifiquemos los tipos de datos, las cadenas que ocupen mas o menos según nuestra tabla.

---

Importemos los 5M de registros del CSV a las dos tablas.

---

Comparemos los datos del sp\_spaceused

---

Comparemos el plan de ejecución, mirando el tiempo estimado del tablescan en cada una de las tablas.

---

Lanza un select sobre las 3 tablas con el SQLQueryStress para ver la diferencia.

# PLAN DE EJECUCIÓN

SSMS o Azure Data Studio nos dan la opción de ver el plan de ejecución de una consulta.

Entendiendo ese plan, podemos plantear mejoras.

Esas mejoras, solo se notan con BBDD muy grandes. Con pocos registros, no se nota la diferencia con los ordenadores actuales.

Nuestras BBDD deben poder escalar. Por lo tanto DEBEMOS preocuparnos de esos planes de ejecución.

# Analizar el plan de Ejecución

```
select Country from sales where country = 'Spain'
```

100 %

Results Messages Execution plan

Query 1: Query cost (relative to the b  
SELECT [Country] FROM [sales] WHERE [c  
Missing Index (Impact 99.8282): CREATE

Table Scan

Scan rows from a table.

Physical Operation	Table Scan
Logical Operation	Table Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Number of Rows Read	5000000
Actual Number of Rows for All Executions	27035
Actual Number of Batches	0
Estimated I/O Cost	62,6647
Estimated Operator Cost	63,5814 (99%)
Estimated Subtree Cost	63,5814
Estimated CPU Cost	0,91668
Estimated Number of Executions	1
Number of Executions	12
Estimated Number of Rows for All Executions	54679,8
Estimated Number of Rows Per Execution	54679,8
Estimated Number of Rows to be Read	5000000
Estimated Row Size	19 B
Actual Rebinds	0
Actual Rewinds	0

Query executed successfully.

# Plan de ejecución

Sin indices de una select con un where.

# Plan de Ejecución con índice

```
create nonclustered index indx_country on sales(Country)

select Country from sales where country ='Spain'
```

Query 1: Query cost (relative to query plan) 100.00 %

SELECT [Country] FROM [sales]

Index Seek (NonClustered)

Cost: 0.005s

27035 of 27035 (100%)

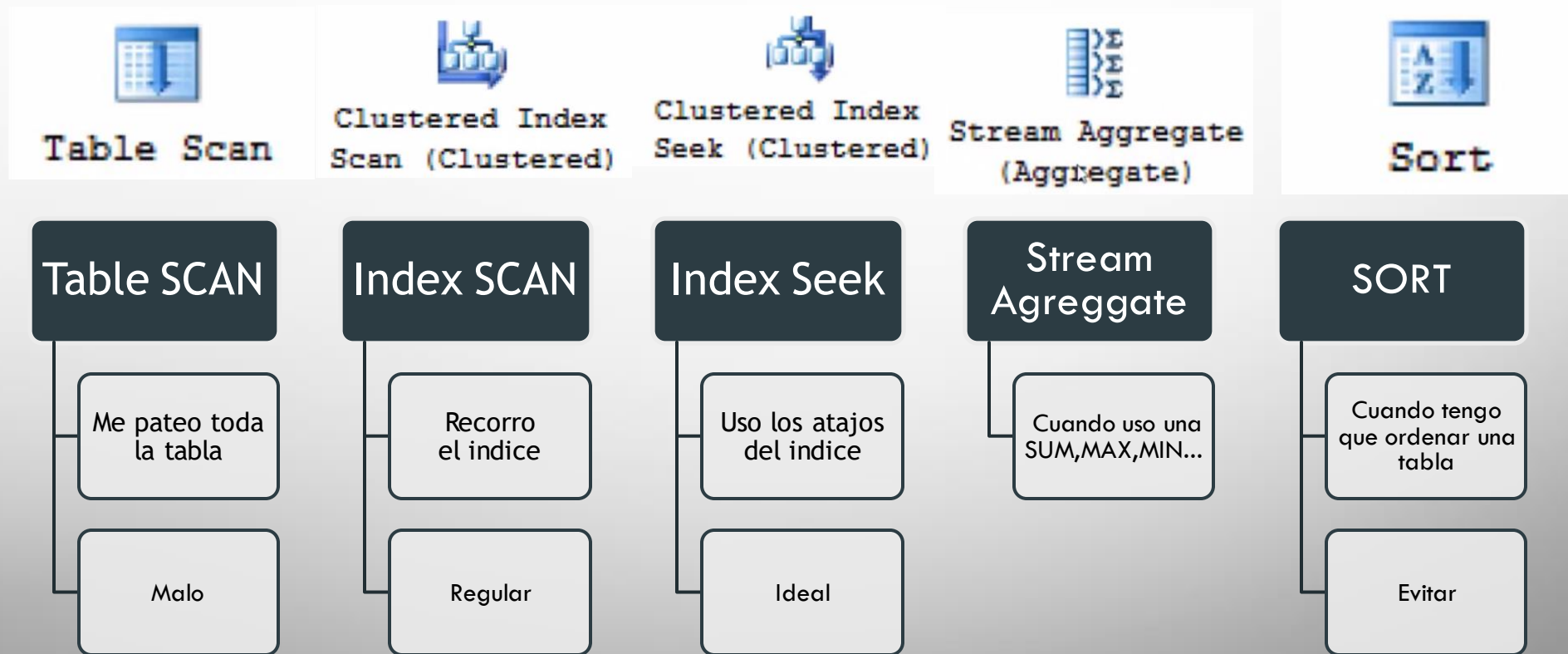
Query executed successfully.

Index Seek (NonClustered)	
Scan a particular range of rows from a nonclustered index.	
Physical Operation	Index Seek
Logical Operation	Index Seek
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Number of Rows Read	27035
Actual Number of Rows for All Executions	27035
Actual Number of Batches	0
Estimated Operator Cost	0,0989501 (100%)
Estimated I/O Cost	0,0690546
Estimated Subtree Cost	0,0989501
Estimated CPU Cost	0,0298955
Estimated Number of Executions	1
Number of Executions	1
Estimated Number of Rows for All Executions	27035
Estimated Number of Rows to be Read	27035
Estimated Number of Rows Per Execution	27035
Estimated Row Size	19 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	True
Node ID	0

- ▶ Agregamos un índice
- ▶ create nonclustered index  
indx\_country on sales(Country)
- ▶ El plan y los costes mejoran



# Iconos En el Plan de Ejecución



# Usar una función evita usar índices

Si usamos por ejemplo la función YEAR(fechar), no usaremos el index seek sino un index scan (en caso de tener un índice sobre la fecha).

```
CREATE NONCLUSTERED INDEX INDX_DATE ON  
SALES(ORDER_DATE)
```

```
SELECT TOP 100 order_date FROM  
sales WHERE YEAR(order_date) = 2018
```

```
SELECT TOP 100 order_date FROM sales  
WHERE Order_date < '01/01/2019' and  
Order_date > '01/01/2018'
```

## Con YEAR

- Poco eficiente

## Sin Year

- Muy Eficiente

# Ejercicio

Comprueba las  
dos consultas  
con la fecha con  
el índice creado.

Mira el plan de  
ejecución

Mira los  
resultados con  
el  
SqlQueryStress

# JOINS

Los Joins, SIEMPRE son más efectivos que una subconsulta.

Las subconsultas, si se ponen y se puede, SIEMPRE entre el FROM y el Where.

Ni en el Select, ni el en Where

En el FROM o en un Inner.

# Joins EFicientes

El plan de Ejecución, nos diferencia 3 tipos de Joins,

## Merge Join

- Los dos campos del ON están indexados.



Merge Join  
(Inner Join)

## Nested Join

- Uno de los dos campos del join, tiene un Índice



Nested Loops  
(Inner Join)

## Hash Join

- Ninguno de los campos del ON está indexado



Hash Match  
(Inner Join)

# CÓMO MEJORAMOS NUESTROS JOINS?



Debemos intentar evitar los hashjoins y que todos sean MergeJoin, o como mínimo Nested.



Creando Indices en las Fks



Repensando nuestros Joins.

# Ejercicio

- ▶ Analiza los Joins que hicimos el trimestre pasado.
- ▶ Empleados de California con el AdventureWorks2017.
- ▶ Analiza el plan de ejecución de la consulta.
- ▶ Intenta mejorarla con Índices o cambiando el orden de los Joins.
- ▶ Usa el SQLQueryStress para notar las diferencias



# Borrando el log de transacciones



# Limpiar el log de transacciones

Las bbdd SQL Server son dos ficheros

El MDF son los datos



El LDF es el log de transacciones.

No tiene sentido guardar el log si tienes un backup completo reciente.



El log suele crecer con el tiempo.

# ¿Como lo recortamos?

Mirar el tamaño de los dos ficheros de la bbdd bigdata:

 bigdata.mdf	19/01/2022 21:29	Archivo MDF	1.395.456 KB
 bigdata_log.ldf	19/01/2022 21:29	Archivo LDF	2.301.952 KB

```
ALTER DATABASE bigdata
SET RECOVERY SIMPLE;
GO
DBCC SHRINKFILE(bigdata_log, 1);
GO
ALTER DATABASE bigdata
SET RECOVERY FULL;
GO
```

 bigdata.mdf	19/01/2022 21:38	Archivo MDF	1.296.704 KB
 bigdata_log.ldf	19/01/2022 21:39	Archivo LDF	3.976 KB

# Que es el SET RECOVERY SIMPLE y FULL

Simple

Nuestra BBDD  
no almacena  
las transacciones.

Full

Se guardan  
las transacciones  
en el log.

# Crea una Bateria de pruebas

- ▶ Recopila en un script, todas las pruebas que hemos hecho en este tema.
  - ▶ Importar CSV 5M
  - ▶ Consultas con indice y sin indice, Con indices Clustered y nonclustered. Con Primary Key y Sin Primary Key...
  - ▶ Filtros con YEAR y sin YEAR
  - ▶ Cambiar tipos de datos en la tabla.
  - ▶ Consultas con Inner Joins y Subconsultas
  - ▶ Algun Group by con un MAX
- ▶ Quiero una bateria de pruebas que testee todo lo hablado y que en los planes de pruebas aparezcan todos los inconos mencionados en el PDF.

# ¿Qué documentamos de la batería de pruebas?

- ▶ Plan de ejecución
- ▶ SET STATISTICS IO ON
- ▶ Tiempos de ejecución del plan de ejecución
- ▶ TEST con SQLQUERYStress
- ▶ SQL Server Profiler

# Ejercicio Con Docker

- ▶ Lanza la bateria de pruebas mirando el plan de ejecución y probando el SQLQueryStress.
- ▶ Mira el SQL Server Profiler.
- ▶ Documenta como ha funcionado Docker con esta bateria de prueba

# Ejercicio Con Raspberry Pi u otra máquina virtual

- ▶ Instalar y configurar una Raspberry Pi
- ▶ Instalar docker en ella
- ▶ Levantar SQL Server
- ▶ Lanzar las baterias de prueba que hemos hecho en nuestro equipo en la Raspberry Pi
- ▶ Usar SQLQueryStress
- ▶ Probar con indices y sin indices
- ▶ Comparar resultados con vuestro equipo

# Ejercicio Con Virtual Box

- ▶ Instala SQL Server en un windows Server sobre Virtual Box
- ▶ Lanza la bateria de test sobre el virtual box y compara resultados con docker y Raspberry Pi