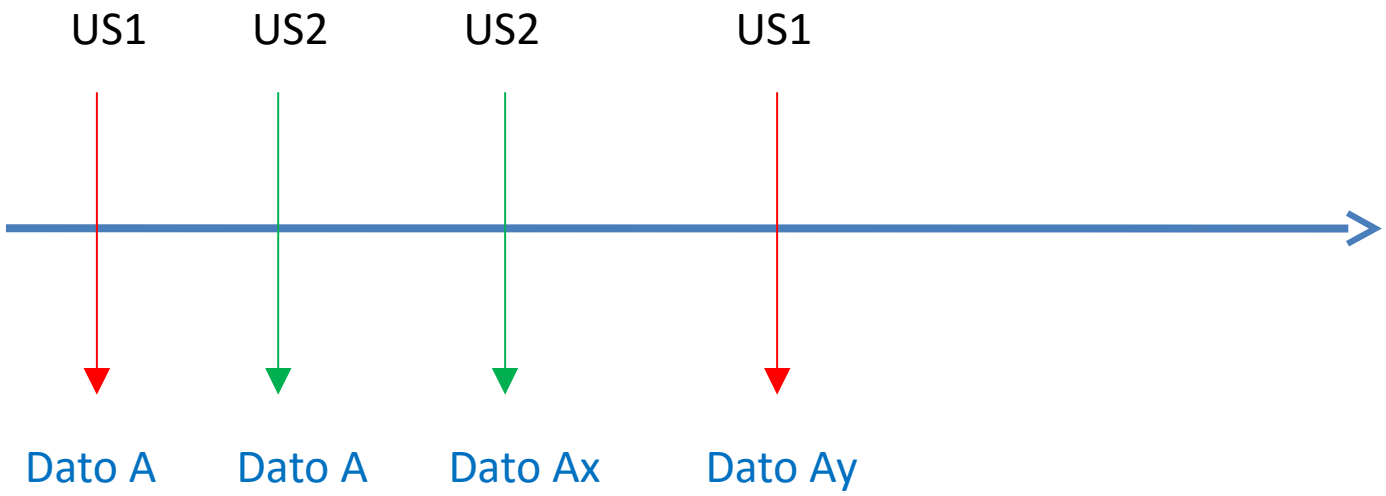
Bloqueos

En general los bloqueos son gestionados por Microsoft SQL SERVER de forma automatizada sin necesidad de gestión manual, pero existen dos factores los cuales necesitan una intervención manual, estos son en términos informáticos : BLOCKING y DEADLOCKS.

- BLOCKING.- Con este término se conoce la acción de espera de un proceso o varios por culpa de un bloqueo existente por otro proceso sobre los mismos datos. La solución para evitar mínimamente este tipo de bloqueo es optimizar el diseño de las tablas, los índices utilizados, la implementación de la sentencia sql y de la configuración hardware.
- DEADLOCK.- Con este término se conoce a el suceso de que dos procesos estén cada uno bloqueando un objeto y a la vez ambos necesitan también acceder a el objeto bloqueado por el otro. Esto crea un nudo, para lo que Microsoft SQL SERVER dependiendo del tiempo de espera en el procesador de los procesos, realiza un roll back y lanza el mensaje de error 1205 en uno de los procesos, permitiendo al otro continuar.

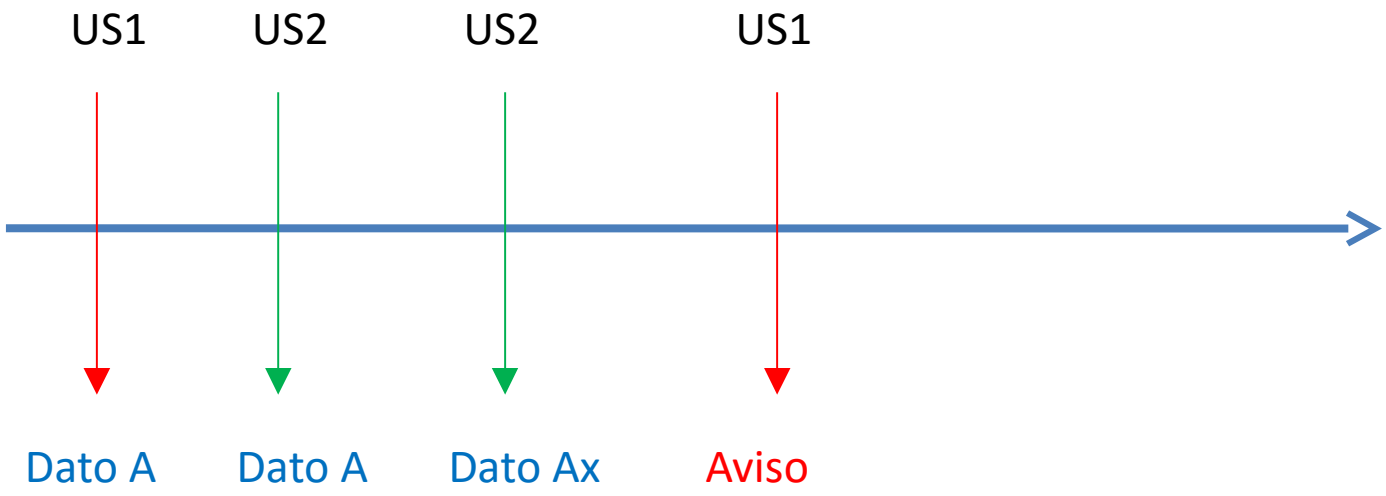
Mediante el monitor de actividad o el procedimiento sp_lock podemos ver la situación de los bloqueos.

Situación sin bloqueos



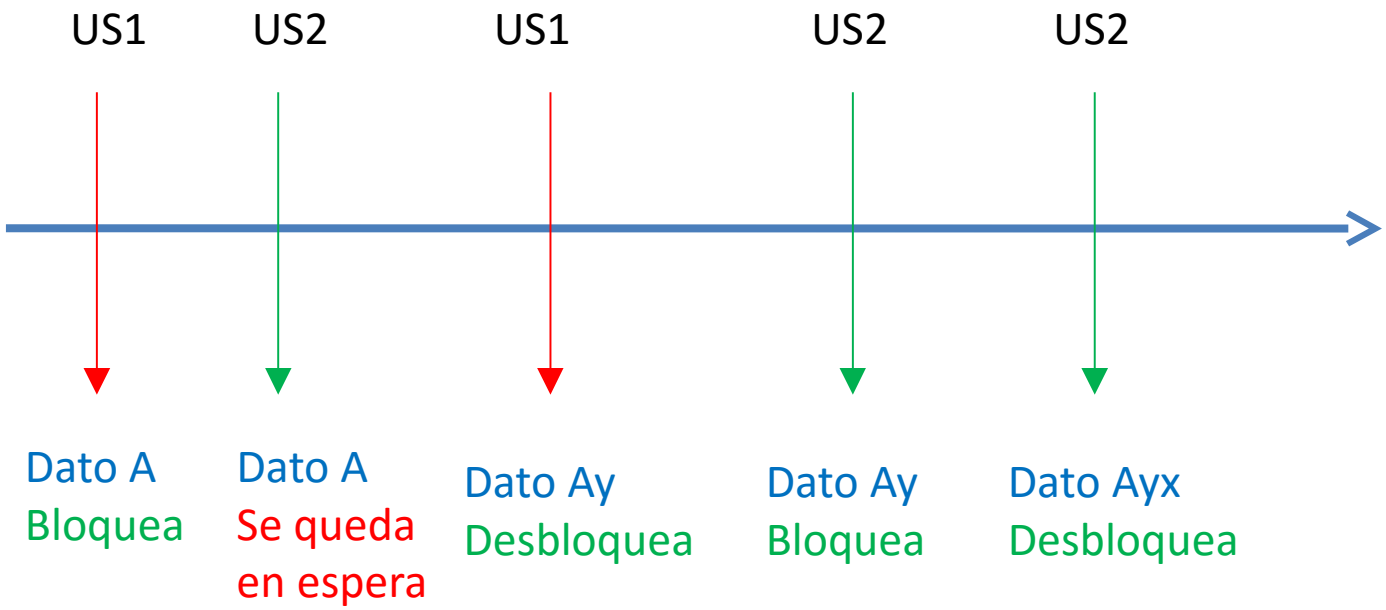
Se pierde la modificación del US2 Ax

Detección por el SGBD con mensaje de error



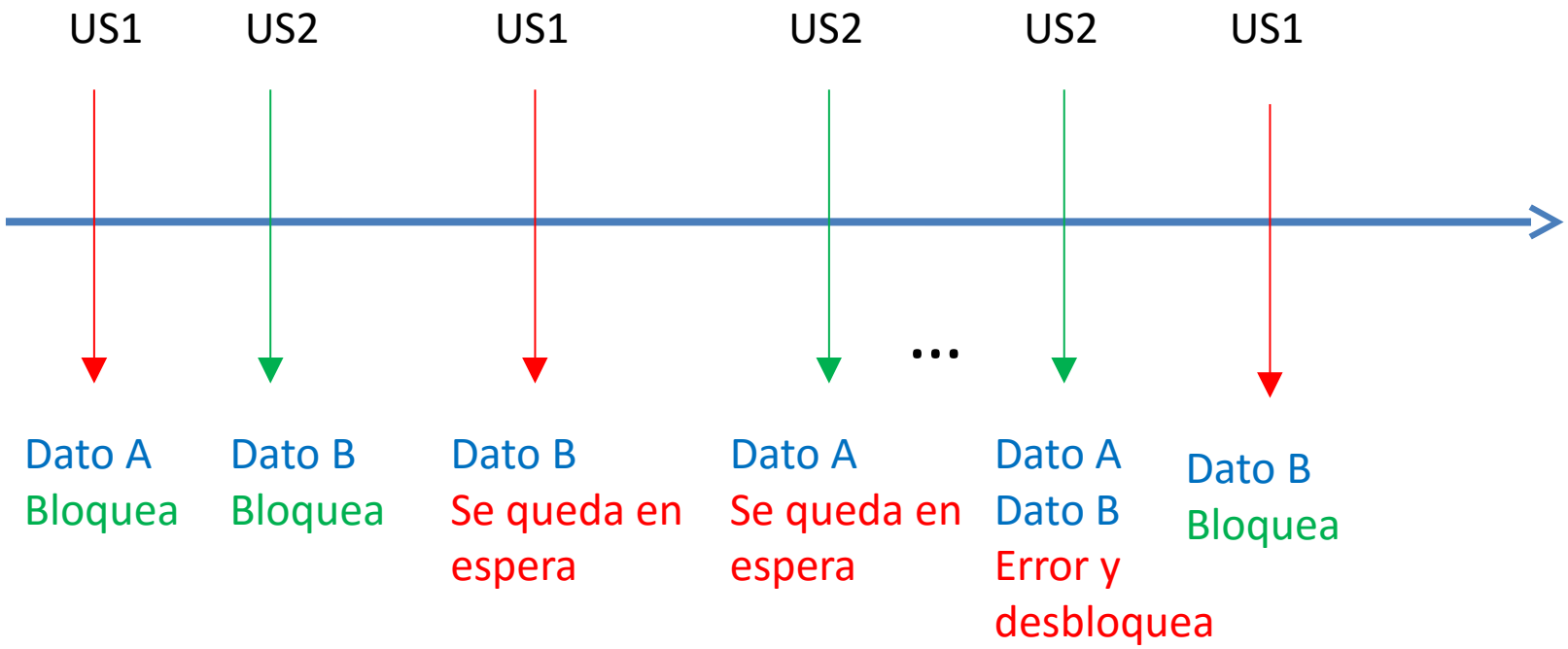
El SGBD avisa de que los datos se han modificado desde la última lectura

Situación con bloqueos



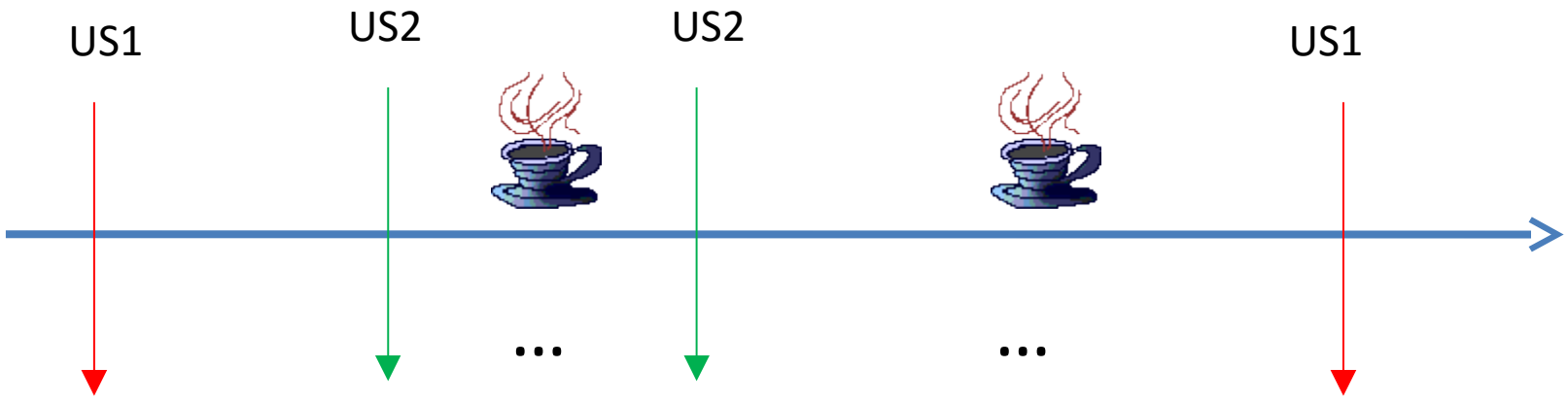
Antes de actualizar se bloquea la información y nadie tiene acceso a bloquearla para modificarla

Problema: interbloqueo



El SGBD lo detecta tras un tiempo definido y le da error a uno de los usuarios

Problema: bloqueo pendiente (tomar café)



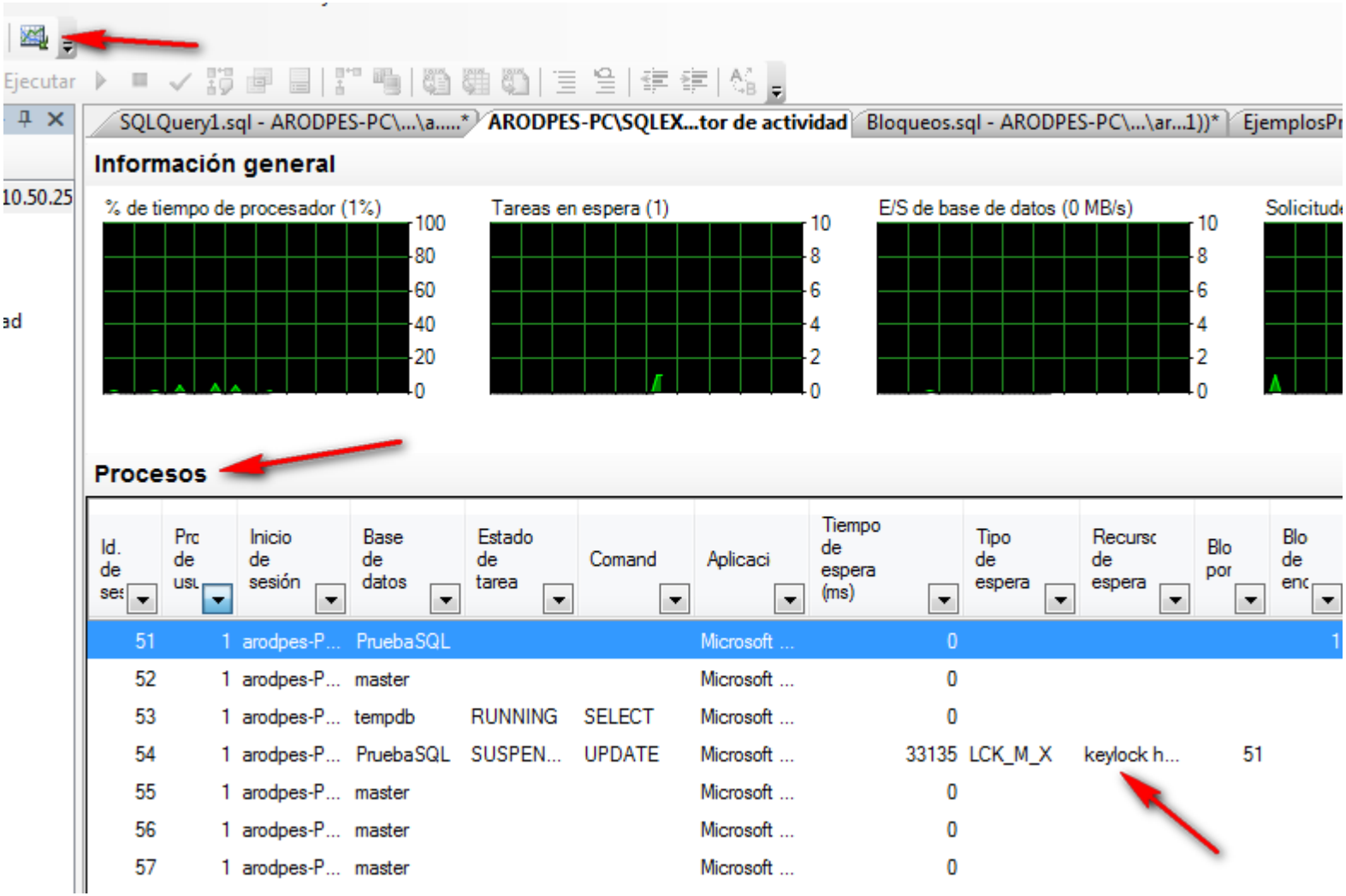
Dato A
Bloquea
Se produce E/S
y queda
pendiente de
respuesta

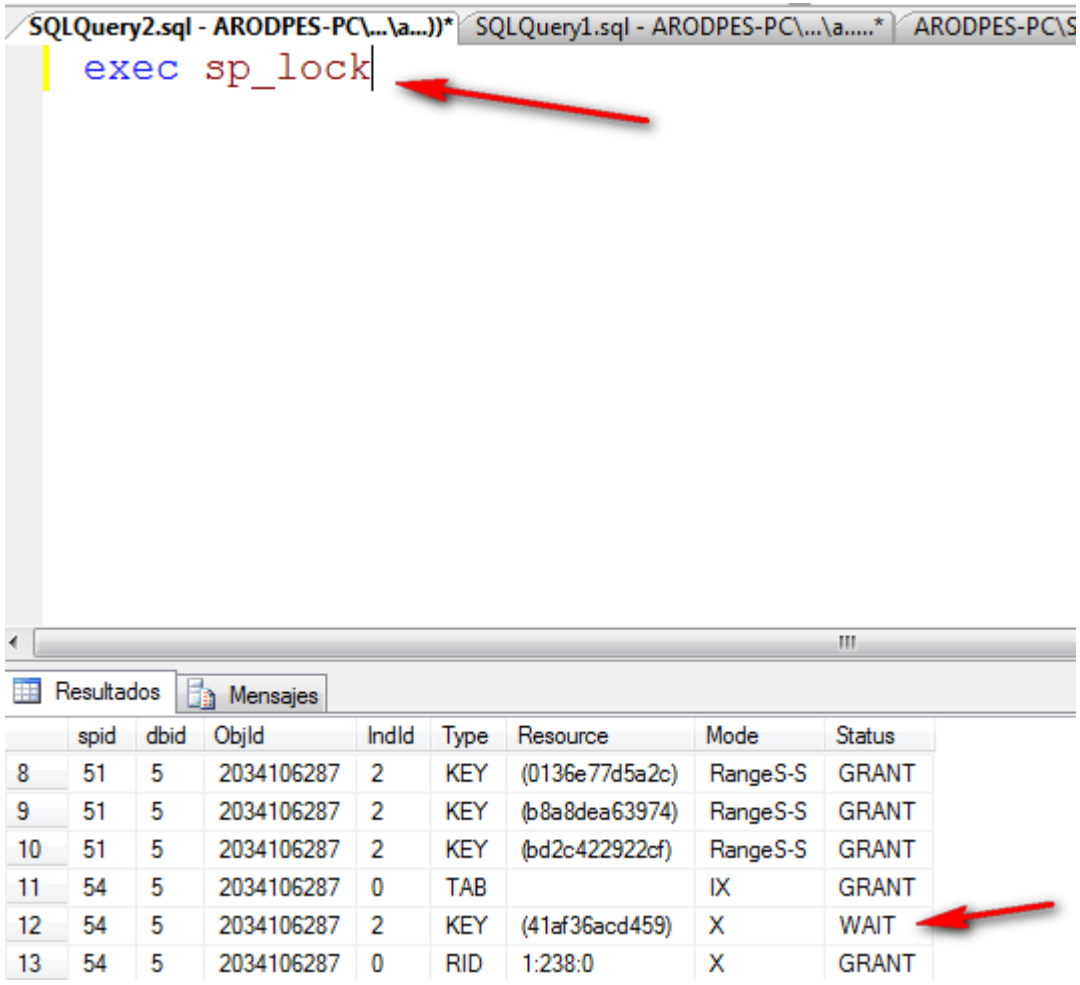
Dato A
Se queda en
espera

Dato A
Error por
mucho tiempo
en espera

Dato A
Desbloquea

Queda pendiente el US2 hasta que el US1 termine la E/S y desbloquee la información.
Puede dar error a US2 si pasa mucho tiempo en espera. También puede detectar que pasa mucho tiempo bloqueado y dar error al US1


Ejecutar
SQLQuery1.sql - ARODPES-PC\...\a.....*
ARODPES-PC\SQLEX...tor de actividad
Bloqueos.sql - ARODPES-PC\...\ar...1))*
EjemplosPr
Información general
% de tiempo de procesador (1%)
100
80
60
40
20
0
Tareas en espera (1)
10
8
6
4
2
0
E/S de base de datos (0 MB/s)
10
8
6
4
2
0
Solicitud
Procesos
Id. de ses
Prc de usu
Inicio de sesión
Base de datos
Estado de tarea
Comand
Aplicaci
Tiempo de espera (ms)
Tipo de espera
Recurso de espera
Blo por
Blo de enc
51 1 arodpes-P... PruebaSQL Microsoft ... 0 1
52 1 arodpes-P... master Microsoft ... 0
53 1 arodpes-P... tempdb RUNNING SELECT Microsoft ... 0
54 1 arodpes-P... PruebaSQL SUSPEN... UPDATE Microsoft ... 33135 LCK_M_X keylock h... 51
55 1 arodpes-P... master Microsoft ... 0
56 1 arodpes-P... master Microsoft ... 0
57 1 arodpes-P... master Microsoft ... 0



	spid	dbid	ObjId	IndId	Type	Resource	Mode	Status
8	51	5	2034106287	2	KEY	(0136e77d5a2c)	RangeS-S	GRANT
9	51	5	2034106287	2	KEY	(b8a8dea63974)	RangeS-S	GRANT
10	51	5	2034106287	2	KEY	(bd2c422922cf)	RangeS-S	GRANT
11	54	5	2034106287	0	TAB		IX	GRANT
12	54	5	2034106287	2	KEY	(41af36acd459)	X	WAIT
13	54	5	2034106287	0	RID	1:238:0	X	GRANT

Preparamos el ejemplo de bloqueo, creando una tabla y su índice, cargándola además con datos.

```
USE pruebasql;
GO
-- Crear tabla e índice de prueba.
if object_id('t_lock_ci') is not null
  drop index libros.t_lock_ci;
if object_id('t_lock') is not null
  drop table t_lock;
CREATE TABLE t_lock
    (c1 int, c2 int);
GO
CREATE INDEX t_lock_ci on t_lock(c1);
GO
-- Insertar valores en la tabla de prueba
INSERT INTO t_lock VALUES (1,1),(2,2),(3,3),(4,4),(5,5),(6,6);
GO
select c1,c2 from t_lock
go
```

Abrimos una sesión (Nueva consulta) para ejecutar lo siguiente:

```
-- Session 1 se realizan en instancias de consulta diferentes
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
--configuramos de manera que las sentencias no pueden
--leer datos modificados hasta que se hace commit
BEGIN TRAN
    SELECT c1
        FROM t_lock
        WITH(holdlock, rowlock);--bloqueamos el select
```

Abrimos otra sesión con un intento de actualización, que queda pendiente de ejecución, pendiente de que finalice la ejecución de la sentencia de la otra sesión.

```
BEGIN TRAN
    UPDATE t_lock SET c1 = 10;
```

Se puede detectar que existe el bloqueo.

Desbloqueamos la primera sesión, si pasa mucho tiempo esto se realiza automáticamente

```
-- Session 1
ROLLBACK;
GO
```

En la segunda sesión aparece el mensaje de Comandos completados correctamente.

Decidimos si se efectúa o no el trabajo de la segunda sesión.

```
    -- Session 2
ROLLBACK;
GO
```

Estrategias de bloqueo

Sin bloqueo: la ejecución sin el valor de bloqueo es la más rápida. Si utiliza datos de sólo lectura, es posible que no necesite el bloqueo.
Bloqueo pesimista: se gestiona en el programa, adquiere bloqueos sobre entradas y luego mantiene los bloqueos hasta que se realiza la confirmación. Esta estrategia de bloqueo proporciona una mayor coherencia a costa del rendimiento.
Bloqueo optimista: se encarga el SGBD y no el programa, toma una imagen anterior de cada registro que toca la transacción y compara la imagen con los valores de entrada actuales cuando se confirma la transacción. Si los valores de entrada cambian, la transacción se retrotrae. No se mantiene ningún bloqueo hasta el momento de la confirmación. Esta estrategia de bloqueo proporciona una mejor concurrencia que la estrategia pesimista, con el riesgo de que la transacción se retrotraiga y el coste de memoria de realizar una copia adicional de la entrada.