LockDebugging

Debugging locking situations

Sometimes, when multiple users or threads are accessing a database at the same time, locking situations may occur. Lock timeouts happen when a transaction has waited "long enough" for a resource that is locked by another transaction. Deadlocks occur when two or more transactions are waiting for each other, and it is impossible to get further without rolling back at least one of the transactions.

Although locking situations are not necessarily an indication of faults in your application, it is always useful to know why it happened, and possibly how to avoid it in the future.

Get educated

In order to understand how your application affects locking in the database, you should first get a basic understanding the relationship between the different types of locks, concurrency modes (isolation levels), transaction lifetimes, etc. in Derby.

The Derby manuals include lots of useful information on concurrency and locking in Derby. For example, the Derby Developer's Guide contains an entire chapter entitled "Locking, concurrency, and isolation".

Try to envision how this would work in your particular application.

Obtain debug information

Deadlocks

When a deadlock occurs, you will most likely get some output looking like this in some exception message (and/or in derby.log):

```plaintext
ERROR 40001: A lock could not be obtained due to a deadlock, cycle of locks and waiters is:
Lock : ROW, T, (1,8)
  Waiting XID : (235, S) , APP, select * from t where id = 2
  Granted XID : (238, X)
Lock : ROW, T, (1,7)
  Waiting XID : (238, S) , APP, select * from t where id = 1
  Granted XID : (235, X)
. The selected victim is XID : 235.
```

If you don't see such output, you may need to set the derby.locks.deadlockTrace property or some of the other derby.locks. properties, or upgrade to a newer version of Derby. The various properties you can set are described in the Derby Reference Manual, and some usage is described in the Developer's Guide in "Configuring deadlock detection and lock wait timeouts".

Lock timeouts

When a lock timeout occurs, a message such as the following will be printed to derby.log or be part of some exception message caught by your application:

```plaintext
java.sql.SQLTransactionRollbackException: A lock could not be obtained within the time requested.
```

If you have enabled the derby.locks.deadlockTrace property, you will probably also see a dump of the lock table at the time of the timeout, for example:
java.sql.SQLTransactionRollbackException: A lock could not be obtained within the time requested. The lockTable dump is:
2007-06-15 11:34:06.748 GMT

<table>
<thead>
<tr>
<th>XID</th>
<th>TYPE</th>
<th>MODE</th>
<th>LOCKCOUNT</th>
<th>LOCKNAME</th>
<th>STATE</th>
<th>TABLETYPE / LOCKOBJ</th>
<th>INDEXNAME / CONTAINER_ID / (MODE for LATCH only)</th>
<th>TABLENAME / CONGLOM_ID</th>
</tr>
</thead>
</table>
------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------

*** The following row is the victim ***
174 | ROW | S | 0 | (1,7) | WAIT | T | NULL | T |

*** The above row is the victim ***
171 | ROW | S | 1 | (1,8) | GRANT | T | NULL | T |
174 | ROW | S | 1 | (1,1) | GRANT | T | SQL070615133359510 | T |
171 | ROW | S | 1 | (1,1) | GRANT | T | SQL070615133359510 | T |
171 | TABLE | IX | 2 | Tablelock | GRANT | T | NULL | T |
174 | TABLE | IS | 1 | Tablelock | GRANT | T | NULL | T |
171 | TABLE | IS | 1 | Tablelock | GRANT | T | NULL | T |
171 | ROW | X | 3 | (1,7) | GRANT | T | NULL | T |

If you don't see such a lock table dump even when the deadlockTrace property has been enabled, consult the Derby Developer's Guide. See also the FAQs.

It is also possible to obtain a snapshot of the lock table of a running database by executing the following statement:

```
select * from syscs_diag.lock_table;
```

More information about the lock table can be found in the "SYSCS_DIAG.LOCK_TABLE diagnostic table" topic of the Derby Reference Manual.

### Interpreting the diagnostics

Not every aspect of the debug information mentioned above (deadlock traces or lock table dumps) is intuitive to the average user of Derby. This section is intended to provide information that may be helpful in deciphering the output.

A walk-through of a lock timeout debugging session on IRC is available in the mailing list archives for the derby-user mailing list.

Please add any other useful information you can think of to this page 😊

### Related Issues

- [DERBY-360]: Add doc info to print lock table for timeout errors
- [DERBY-666]: Enhance derby.locks.deadlockTrace to print stack traces for all threads involved in a deadlock
- [DERBY-2828]: derby.locks.deadlockTrace docs claim the property is meaningless without derby.locks.monitor=true
- [DERBY-2845]: Devguide tips on debugging deadlocks should be clearer
- [DERBY-2877]: Print the entire lock list when a deadlock occurs and deadlock tracing is on
- [DERBY-2991]: Index split deadlock