

Re: transaction demarction

Source: <http://coding.derkeiler.com/Archive/Java/comp.lang.java.programmer/2006-11/msg01230.html>

- *From:* "Rhino" <no.offline.contact.please@xxxxxxxxxx>
 - *Date:* Fri, 10 Nov 2006 02:14:33 -0500
-

"gk" <srcjnu@xxxxxxxxxx> wrote in message
news:1163135217.271071.43820@xx

hi, what does it mean by transaction demarcation?(in EJB)

you see, they are saying "demarcation" ? do they mark something ?
what they mark ? how thye mark ? who marks ?

transactions are unit of statements whcih begins and ends .

i am not sure about they keyword "demarcation".....this is confusing.

can you explain with an example ?

You are 'gbattine', aren't you?

I don't use EJBs and I don't know much of the theory behind them but I
suspect that they are talking about "units of work".

Programs are often broken down into units of work. The many things a complex
program can do are grouped together into chunks of code that do logically
related things.

For instance, a banking transaction that moves money from one bank account
into another may also report the balances in both accounts that were
involved in the money transfer.

The work of transferring the money from the first account to the second
would usually be logically grouped together into the same logical chunk of
code: the debit of the first account and the credit of the same amount in
the second account would have to both succeed or both fail as a UNIT. You
would NOT want the debit to succeed but have the credit fail; that would
throw the accounts out of balance: if that happened, either the bank would
get unearned money or you would. Either way, auditors would not be happy.

Re: transaction demarcation

Therefore, the debit and credit have to both succeed or both fail. They get grouped together into the same "unit of work".

By the same token, the reporting of the final balances in the two accounts would likely be grouped together into a unit of work, although likely a different unit of work than the transfer.

The job of the application designer is to think of the best place to put the boundaries on the units of work: what work gets done in each unit of work. A complex program might have several units of work while a simple one might have only one. The boundaries between units of work can also be called "demarcations". In our banking example, there were two units of work: the one that did the transfer and the one that reported the balances. There was a "demarcation" between the two units of work.

In practice, the first unit of work begins the first time you do anything that involves database access and ends when you either COMMIT or ROLLBACK. The COMMIT or ROLLBACK ends the unit of work or, in other words, is the demarcation that shows where the unit of work ends. The final unit of work ends when the program issues a COMMIT or ROLLBACK and begins when the previous unit of work ends. All other units of work begin after the previous COMMIT or ROLLBACK and end on the next COMMIT or ROLLBACK.

The other critical point is that some events can cause COMMIT or ROLLBACK to take place, even if COMMIT or ROLLBACK aren't coded in that spot in the program. For example, a normal completion of a program usually causes a COMMIT, even if the program doesn't include a COMMIT statement and an abnormal termination of a program usually causes a ROLLBACK, even if the program doesn't include a ROLLBACK statement. You **HAVE** to know this, otherwise, you will never understand why specific work experienced a COMMIT or ROLLBACK. You need to learn exactly which situations cause these implicit COMMITs and ROLLBACKs: the specific situations which cause these depend on the specific database and language you are using and should be documented in the manuals for the database. Then again, sometimes the documentation is incomplete and you simply have to ask someone what causes implicit COMMIT and ROLLBACK in your database.

In the case of MySQL, I don't know if they have documented what causes implicit COMMIT or ROLLBACK; you probably need to look in their documentation for the appropriate version of MySQL and ask them (via the MySQL mailing lists) if you can't find this information in the documentation.

--

Rhino

.