◆ ロック

ページ数の関係で紙面に掲載できなかったパターンを、ここで解説します。

暗黙的ロック(Implicit Lock)

アプリケーションプログラマは細心の注意を払って、ロック機構を利用しなければなりません。ロック獲得を忘れれば読み込んだデータが最新のものである保証は無くなり、ロック解放を忘れればシステムはだんだんと動きが悪くなり、最後には止まってしまうでしょう。困ったことにオフラインロック機構はテストで品質を担保するのが困難です。アプリケーションは特定の条件やタイミングが合致した時にだけロックを不適切に扱うかもしれません。例えば例外が発生した時にだけロックを解放し忘れるかもしれません。もしも可能であればプログラマではなく、フレームワークなどの基盤となるソフトウェア層で自動的にロックを管理できれば安心です。これが暗黙的ロックです。

楽観的ロックを暗黙的に行う場合は、データの取り出しで自動的にバージョンを読み出して保持し、データの更新の際にバージョンチェックとバージョンのインクリメントを自動的に行うことになるでしょう。実際、例えば Hibernate のような O-R マッピングツールは設定さえしておけば、アプリケーションプログラムが特に何もコードを書かなくても、こうした処理を自動的に行ってくれる機能を持っています。

悲観的ロックを暗黙的に行う場合、データの取り出しで読み込みロックを自動的に取得し、ビジネストランザクション終了時に獲得した全てのロックを自動的に解放するような機構を提供することになるでしょう。なお、悲観的ロックを暗黙的に行う場合には、書き込みロックの獲得は自動的に行うべきではないという点に注意してください。これには 2 つの理由があります。まず第一に、どの時点で書き込みロックを自動的に獲得すべきかを考えてみましょう。それは恐らく読み込んだオブジェクトの更新処理の開始時点になると考えられます。しかしこの時点というのはビジネストランザクションの開始時点ではないでしょう。ここでロックを失敗させても「ビジネストランザクションの開始時点で衝突を検出して、ユーザの 2 度打ち作業を防ぐ」という悲観的ロックの長所を発揮することができません。第二に、自動的に書き込みロックを獲得すると、不必要にロック期間が長くなりアプリケーションの活性が下がってしまう恐れがあります。以上の理由から、書き込みロックの獲得を自動的に行うのは困難ですが、書き込みロックに関して暗黙的に行えることもあります。それは、更新の反映時に書き込みロックが取得済みであることをチェックする処理です。ロックが取得されていなければ、それは明らかにプログラミングミスなので、実行時例外をスローする機構を用意するのが良いでしょう。