東京ITスクール
TOKYO IT SCHOOL

トランザクションとロック

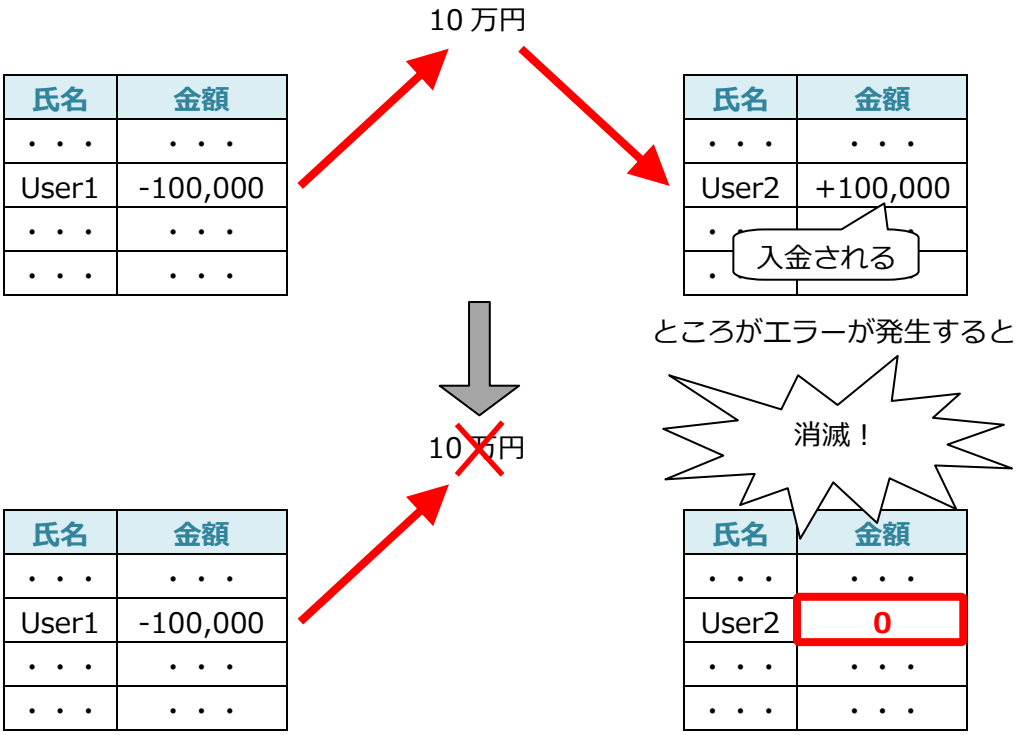
目次

1. はじめに

例えば銀行口座からの引出の際に、通帳への記帳も終わり、いざ引き出したお金が ATM から出力されるタイミングでエラーが発生したらどうなってしまうのでしょう。大切な銀行口座に不整合が発生しては大変です。何が起ろうと、整合性を保つのがデータベースの役目です。この整合性を保つための仕組みが、トランザクションそしてロックです。

2. トランザクションの必要性

トランザクションの必要性を、銀行口座の入出金処理を例に見ていきましょう。



上の図では、User1 は自分の口座にある 10 万円を、User2 の口座に振り込もうとしています。この場合、User1 の口座の残高から 10 万円を引き算し、User2 の口座の残高に 10 万円を足し算することになります。もし、User1 の口座から 10 万円引き算した時点でエラーが発生したらどうなるでしょうか。User2 の口座の残高に 10 万円足し算しないまま、User1 の口座にあった 10 万円が、この世から消滅してしまうことになります。これは、管理上あってはならない事態です。

そこで 10 万円の引き算と 10 万円の足し算を、分割出来ない 1 つの処理として扱い、もし 10 万円の足し算が失敗したら 10 万円の引き算も取り消されるようにするのです。このように、複数の処理をまとめて扱う機能がトランザクションです。

トランザクションを始めてから、結果をデータベースに反映させることをコミット（COMMMIT）、また反映せずに、元に戻すことをロールバック（ROLLBACK）といいます。

3. トランザクションにおけるロック

トランザクションとロックは密接な関係にあります。Oracle では、INSERT、UPDATE、DELETE が実行されると、該当データにはロックがかかり、他から変更することは出来なくなります。そして、そのトランザクションが終了するまでロックされた状態は続きます。

このように、Oracle のロック処理は、テーブル全体ではなく行レベルで行われます。（行レベルロック）これは Oracle の特徴の一つです。

一般的にロックされている状態では、次のようなことがいえます。

◆INSERT、UPDATE、DELETE を実行した結果は、COMMIT・ROLLBACK するまで他のユーザーの SELECT の結果に反映されない
◆INSERT、UPDATE、DELETE を実行した行は、COMMIT・ROLLBACK するまで他のユーザーは変更出来ない

また、Oracle では、次のタイミングでトランザクションが終了します。

◆COMMIT か ROLLBACK を実行したとき
◆DDL 文(ALTER CREATE DROP)を実行したとき
◆接続を切断(異常終了)したとき

このロック機能により、Oracle では、A さんが SELECT しているとき、B さんが UPDATE することによって SELECT の内容が途中で変わってしまう、あるいは後で ROLLBACK されたので、意味のないデータを SELECT してしまうといったことは起こりません。常に、整合性のある確実なデータが読み取れるという特徴、これを読み取り一貫性と呼びます。Oracle は古くから、この読み取り一貫性が強みでした。

4. 自動コミット機能

Oracle では、DML 文（SELECT、INSERT、UPDATE、DELETE）が実行されても自動的にコミットはしません。逆に、DDL 文（ALTER、CREATE、DROP）が実行された場合は自動的にコミットされます。

DML 文を実行すると自動的にコミットされてしまうことを自動コミット機能といいます。Oracle では、デフォルトではこの自動コミット機能はオフの状態になっているのですが、これをオフにして、UPDATE、INSERT、DELETE を実行すると、自動的にすぐにコミットする設定にも出来ます。

自動コミット機能のオン・オフは次のようにして設定します。

（構文）自動コミット機能

```
SET AUTOCOMMIT ON（または OFF）
```

自動コミットをオンにしておけば作業効率は向上します。しかし一瞬にして大量のデータが更新され、ロールバックが出来ない状態であるということは、常に意識する必要があります。自動コミット機能は OFF に戻しておいてください

5. セーブポイント

ROLLBACK で処理を戻す際に、どの時点の状態まで戻すかという位置を SAVEPOINT で指定することが出来ます。SAVEPOINT は、任意の位置に意図して設定することが出来ます。

セーブポイントを設定する構文は以下の通りです。

（構文）セーブポイントの設定

```
SAVEPOINT <セーブポイント名>;
```

（構文）指定するセーブポイントまでロールバック

```
ROLLBACK TO <セーブポイント名>
```