

## デッドロックに関する考察

### 1-1 デッドロックとは

デッドロックとは複数のプロセスが互いに他のプロセスの完了を待ちあってしまい、永久に待ちから抜け出られない状況を指します。

データベースの排他処理をうまく行わないとデッドロックが発生する可能性があります。

簡単な例で説明しましょう。

次のようなテーブルおよびデータがあるものとします。

EMP

EMPNO	ENAME
7369	SMITH
7499	ALLEN

そして、2つのトランザクションが次のように処理を行ったとします。

処理順	トランザクション1	トランザクション2
1	7369 を更新	
2		7499 を更新
3	7499 を更新	
4		7369 を更新

要するにトランザクション1は上、下の順番で更新し、トランザクション2は下、上の順番で更新するものとします。

ロック状況もあわせて記述すると次のような状態となります。

処理順	トランザクション1	ロック状況	トランザクション2	ロック状況
1	7369 を更新	7369 をロック		
2			7499 を更新	7499 をロック
3	7499 を更新	ロック待ち		
4			7369 を更新	ロック待ち

双方が互いのロック解放を待ちあう状態になることが分かります。

この状態のことをデッドロックと呼びます。

### 1-2 Oracle での動き

多くのデータベースはデッドロックの検出機能を備えています。

Oracle を例に取り、実際の動きを見てみることにします。

Oracle のサンプルデータを用いて実験するために次の3つのSQLを用意しました。

```
/* 000.SQL */
-- データの確認

-- 接続
CONNECT SCOTT/TIGER@ORCL

SELECT EMPNO,ENAME FROM EMP
WHERE EMPNO IN (7369,7499) ORDER BY EMPNO;
```

```
/* 001.SQL */
-- デッドロックの例

-- 接続(1)
CONNECT SCOTT/TIGER@ORCL

-- ※以下を順番に貼り付ける

--- (1) -----
UPDATE EMP SET ENAME = 'TEST-1' WHERE EMPNO = 7369;

--- (3) -----
UPDATE EMP SET ENAME = 'TEST-2' WHERE EMPNO = 7499;

--- (5) -----
ROLLBACK;
```

```
/* 002.SQL */
-- デッドロックの例

-- 接続(2)
CONNECT SCOTT/TIGER@ORCL

-- ※以下を順番に貼り付ける

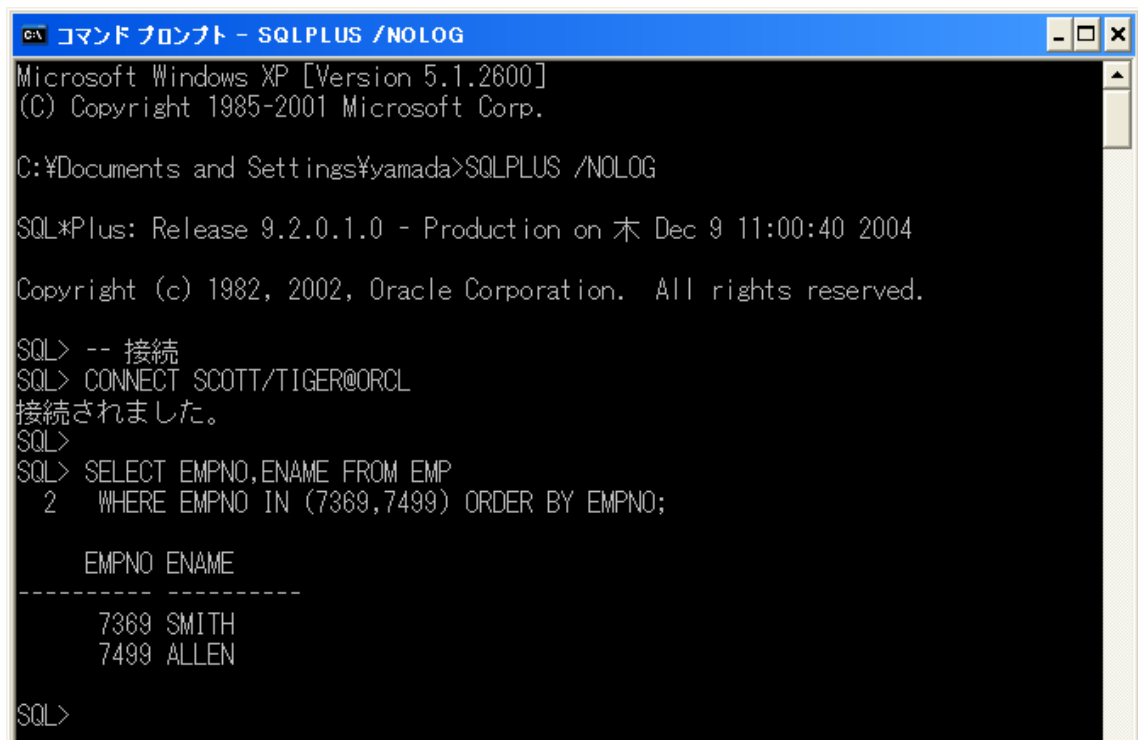
--- (2) -----
UPDATE EMP SET ENAME = 'TEST-2' WHERE EMPNO = 7499;

--- (4) -----
UPDATE EMP SET ENAME = 'TEST-1' WHERE EMPNO = 7369;

--- (6) -----
ROLLBACK;
```

000.SQLはデータの存在を確認するものです。通常、図 1のような結果が得られます。

図 1 テストデータの存在確認



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\yamada>SQLPLUS /NOLOG

SQL*Plus: Release 9.2.0.1.0 - Production on 木 Dec 9 11:00:40 2004
Copyright (c) 1982, 2002, Oracle Corporation. All rights reserved.

SQL> -- 接続
SQL> CONNECT SCOTT/TIGER@ORCL
接続されました。
SQL>
SQL> SELECT EMPNO,ENAME FROM EMP
       2  WHERE EMPNO IN (7369,7499) ORDER BY EMPNO;

      EMPNO ENAME
-----
       7369 SMITH
       7499 ALLEN

SQL>
```

次に 2 つの SQLPLUS を起動し、001.SQL、002.SQL の SQL を順番に貼り付けていきます。  
2 つの SQLPLUS が区別しやすいように背景色を変更しておきます。

図 2 最初の SQL (トランザクション 1)



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\yamada>SQLPLUS /NOLOG

SQL*Plus: Release 9.2.0.1.0 - Production on 木 Dec 9 11:05:34 2004
Copyright (c) 1982, 2002, Oracle Corporation. All rights reserved.

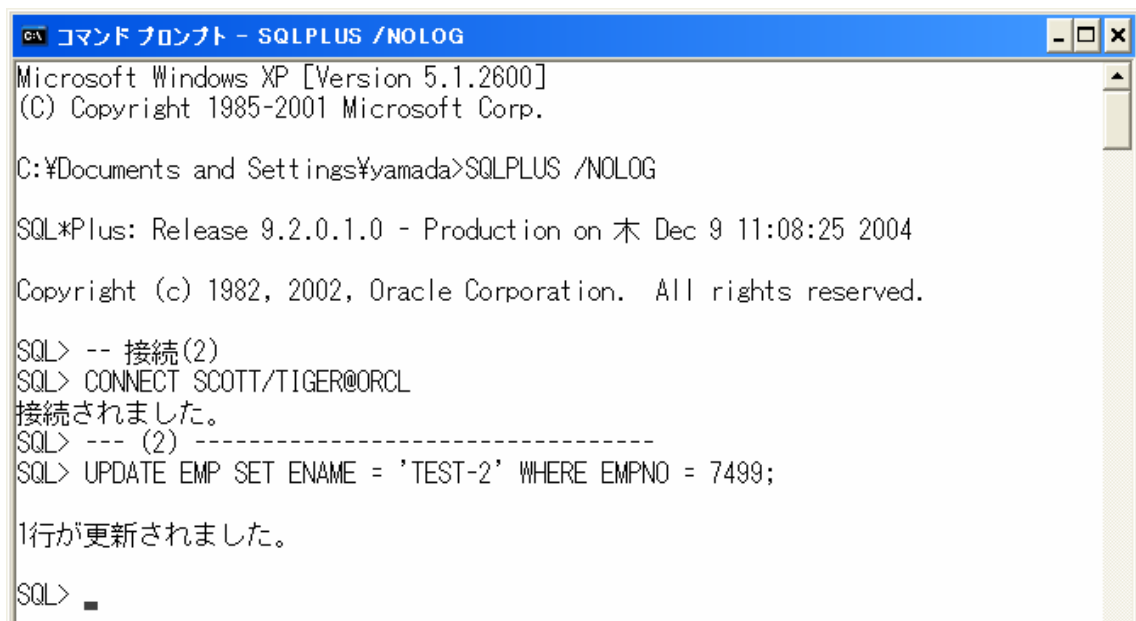
SQL> -- 接続(1)
SQL> CONNECT SCOTT/TIGER@ORCL
接続されました。
SQL> --- (1) -----
SQL> UPDATE EMP SET ENAME = 'TEST-1' WHERE EMPNO = 7369;

1行が更新されました。

SQL> ■
```

「1 行が更新されました」とメッセージが出ていますが、まだコミットしていないので、レコードはロックされた状態になっています。

図 3 2 番目の SQL (トランザクション 2)



```
C:\> コマンド プロンプト - SQLPLUS /NOLOG
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\yamada>SQLPLUS /NOLOG

SQL*Plus: Release 9.2.0.1.0 - Production on 木 Dec 9 11:08:25 2004

Copyright (c) 1982, 2002, Oracle Corporation. All rights reserved.

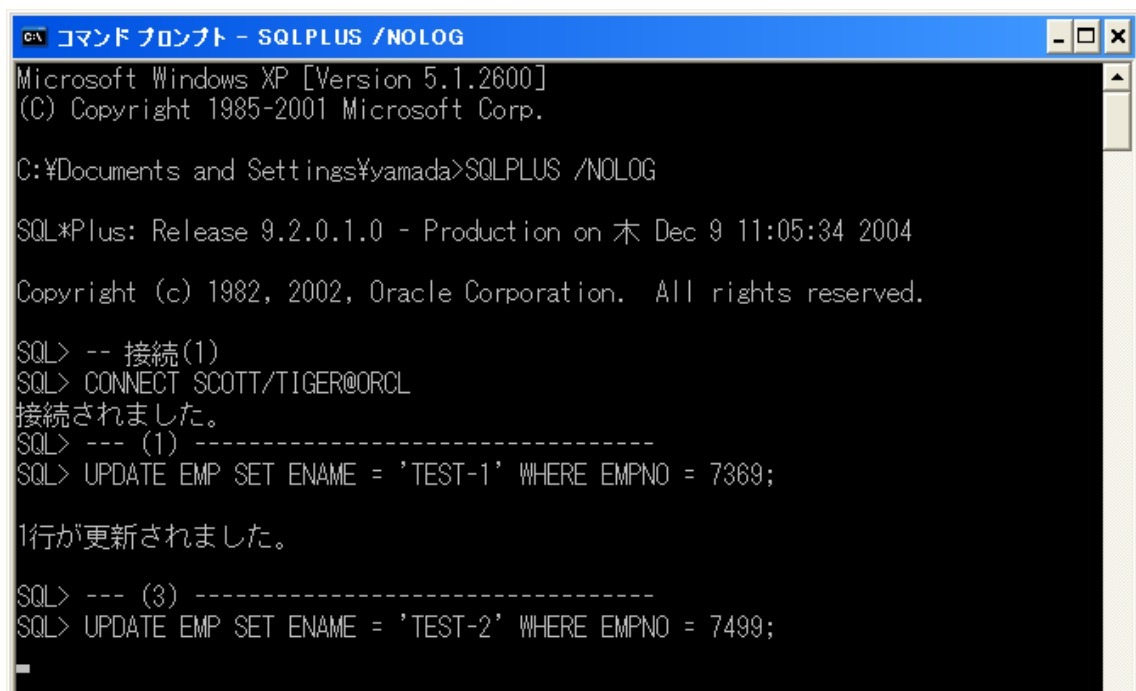
SQL> -- 接続(2)
SQL> CONNECT SCOTT/TIGER@ORCL
接続されました。
SQL> --- (2) -----
SQL> UPDATE EMP SET ENAME = 'TEST-2' WHERE EMPNO = 7499;

1行が更新されました。

SQL> _
```

こちらはまだレコードロック状態です。

図 4 トランザクション 1 で 7499 の更新を実行



```
C:\> コマンド プロンプト - SQLPLUS /NOLOG
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\yamada>SQLPLUS /NOLOG

SQL*Plus: Release 9.2.0.1.0 - Production on 木 Dec 9 11:05:34 2004

Copyright (c) 1982, 2002, Oracle Corporation. All rights reserved.

SQL> -- 接続(1)
SQL> CONNECT SCOTT/TIGER@ORCL
接続されました。
SQL> --- (1) -----
SQL> UPDATE EMP SET ENAME = 'TEST-1' WHERE EMPNO = 7369;

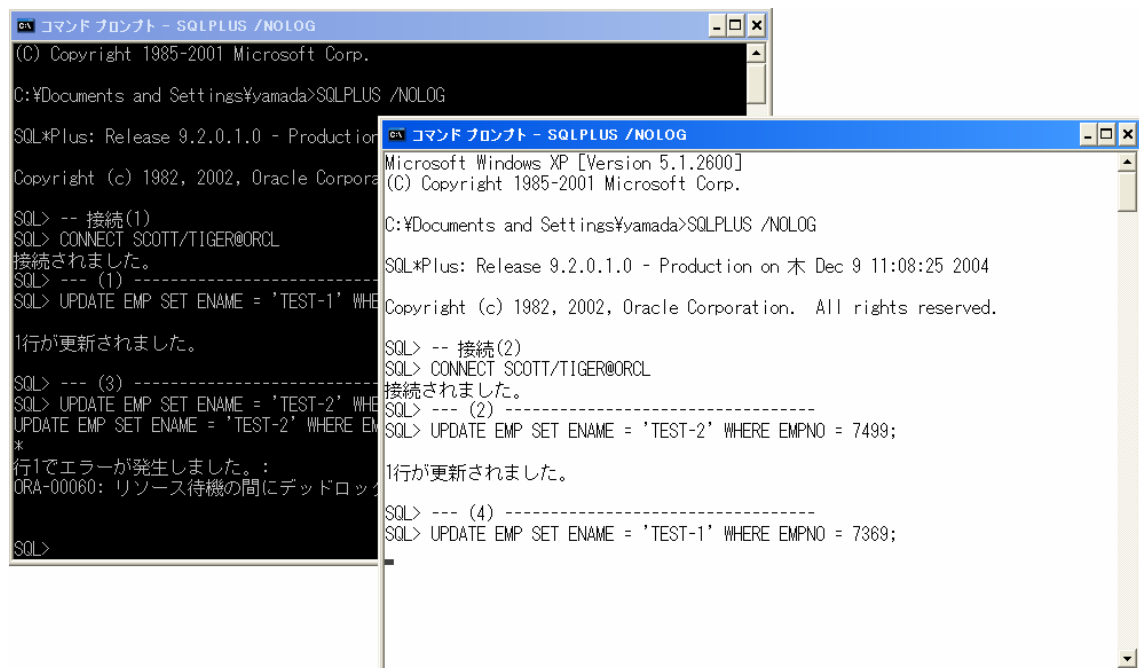
1行が更新されました。

SQL> --- (3) -----
SQL> UPDATE EMP SET ENAME = 'TEST-2' WHERE EMPNO = 7499;

_
```

7499 はトランザクション 2 がロックしているので、ロック待ち状態となっています。

図 5 トランザクション 2 で 7369 の更新を実行



デッドロックが検出され、トランザクション 1 にエラーが表示されているのが分かります。

トランザクション 1 をロールバックすると、トランザクション 2 のロック待ちが解除され、更新が実行されます。

### 1-3 デッドロックの回避方法（理論と実験）

デッドロックを回避するためには次のような方法があります。

- ・ 1 つのトランザクションで複数レコードを更新しない。
- ・ 複数レコードを更新するときは更新順序を規定しておく。

前者の対応は確実ではありますが、複数レコード間の整合性維持というトランザクションの意味がなくなるので、採用はできません。

しかし「なるべく複数レコード更新を行わない」という努力目標とすることができます。具体的には、マスター更新などで複数レコードを画面に表示させ、1 度に更新を行うユーザーインターフェースは避けるのが無難でしょう。

後者の対応がデッドロック回避の一般的手段となっています。

これは次の 2 つの SQL で確認できます。

```
/* 003.SQL */  
-- 順序良く更新すればよい  
  
-- 接続(1)  
CONNECT SCOTT/TIGER@ORCL  
  
-- ※以下を順番に貼り付ける  
  
--- (1) -----
```

```
UPDATE EMP SET ENAME = 'TEST-1' WHERE EMPNO = 7369;
```

```
--- (3) -----
```

```
UPDATE EMP SET ENAME = 'TEST-2' WHERE EMPNO = 7499;
```

```
--- (5) -----
```

```
ROLLBACK;
```

```
/* 004.SQL */
```

```
-- 順序良く更新すればよい
```

```
-- 接続(2)
```

```
CONNECT SCOTT/TIGER@ORCL
```

```
-- ※以下を順番に貼り付ける
```

```
--- (2) -----
```

```
UPDATE EMP SET ENAME = 'TEST-1' WHERE EMPNO = 7369;
```

```
--- (4) -----
```

```
UPDATE EMP SET ENAME = 'TEST-2' WHERE EMPNO = 7499;
```

```
--- (6) -----
```

```
ROLLBACK;
```

実際に実験してみましょう。

図 6 トランザクション 1



```
C:\> コマンド プロンプト - SQLPLUS /NOLOG
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\yamada>SQLPLUS /NOLOG

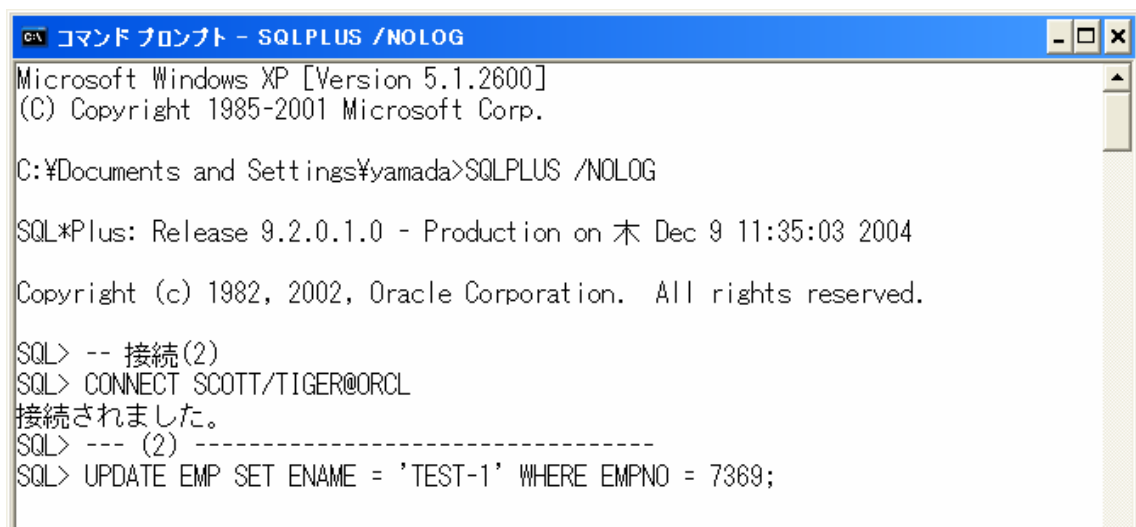
SQL*Plus: Release 9.2.0.1.0 - Production on 木 Dec 9 11:33:20 2004
Copyright (c) 1982, 2002, Oracle Corporation. All rights reserved.

SQL> -- 接続(1)
SQL> CONNECT SCOTT/TIGER@ORCL
接続されました。
SQL> --- (1) -----
SQL> UPDATE EMP SET ENAME = 'TEST-1' WHERE EMPNO = 7369;

1行が更新されました。

SQL>
```

図 7 トランザクション 2



```
C:\> コマンド プロンプト - SQLPLUS /NOLOG
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

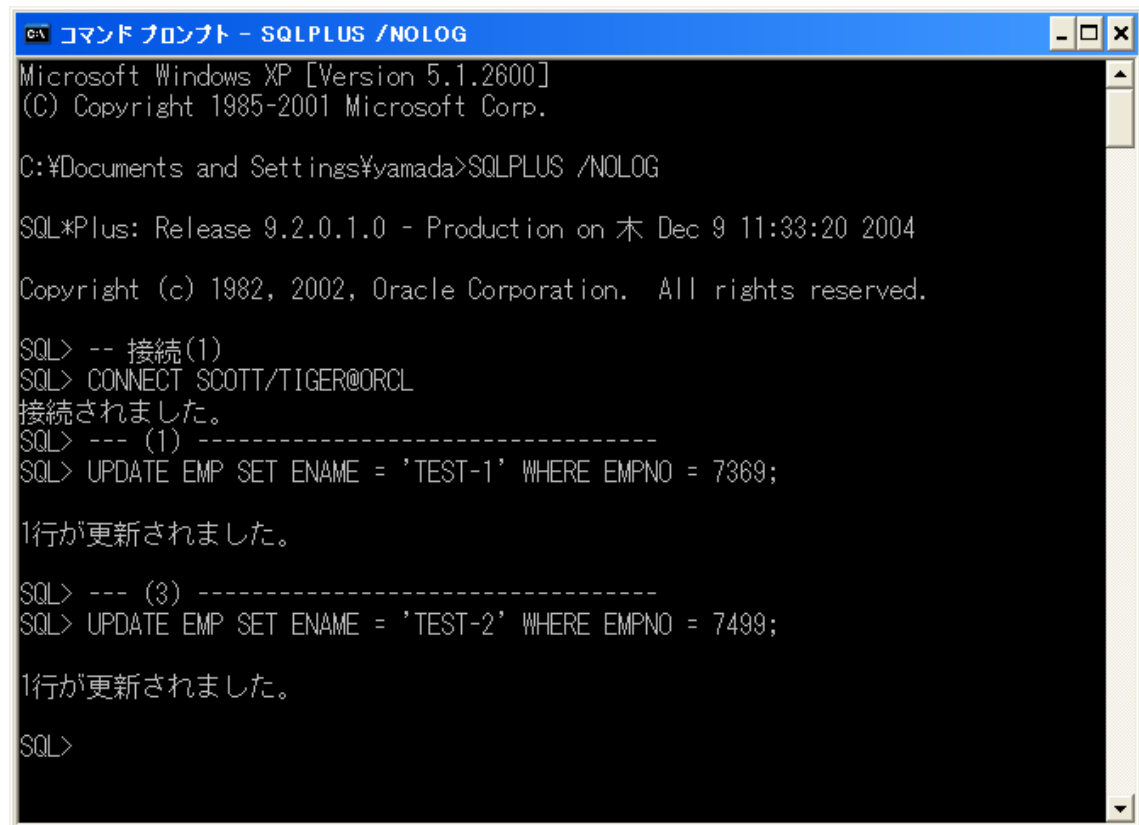
C:\Documents and Settings\yamada>SQLPLUS /NOLOG

SQL*Plus: Release 9.2.0.1.0 - Production on 木 Dec 9 11:35:03 2004
Copyright (c) 1982, 2002, Oracle Corporation. All rights reserved.

SQL> -- 接続(2)
SQL> CONNECT SCOTT/TIGER@ORCL
接続されました。
SQL> --- (2) -----
SQL> UPDATE EMP SET ENAME = 'TEST-1' WHERE EMPNO = 7369;
```

トランザクション 2 はロック待ち状態です。

図 8 トランザクション 1 で 2 行目を更新



```

C:\> コマンド プロンプト - SQLPLUS /NOLOG
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\yamada>SQLPLUS /NOLOG

SQL*Plus: Release 9.2.0.1.0 - Production on 木 Dec 9 11:33:20 2004

Copyright (c) 1982, 2002, Oracle Corporation. All rights reserved.

SQL> -- 接続(1)
SQL> CONNECT SCOTT/TIGER@ORCL
接続されました。
SQL> --- (1) -----
SQL> UPDATE EMP SET ENAME = 'TEST-1' WHERE EMPNO = 7369;

1行が更新されました。

SQL> --- (3) -----
SQL> UPDATE EMP SET ENAME = 'TEST-2' WHERE EMPNO = 7499;

1行が更新されました。

SQL>

```

問題なく更新されることが分かります。トランザクション 2 は相変わらずロック待ちなので 2 行目の更新に移ることができません。トランザクション 1 をコミットまたはロールバックすることによりロックが解除され、更新が行われるようになります。2 行目の更新も他でロックされていないので、問題なく実行されます。

#### 1-4 デッドロックの回避方法（実装方法）

ここで実験したような、1 トランザクションで 1 つのテーブルの複数レコードを更新する例はあまり多くありません。一般的には複数のテーブルに対する更新が 1 つのトランザクションとして実行されます。このときに更新順序を規定するのがプログラム規約となりますが、確実なのはストアードプロシージャ化でしょう。ストアードプロシージャ化が難しい場合は、共通モジュールなどに収めるのが良いでしょう。

#### 1-5 回避不能なデッドロックについて

実際の業務では回避不能なデッドロックも存在します。

次のような例が考えられます。

- ・ 製品入庫時に、子製品の出庫を自動で行う。
- ・ 棚卸実差を修正するために実差のある製品在庫を一気に更新する。

まったく別の処理が製品在庫に対し複数更新を行うので、デッドロック回避は偶然に頼るしかありません。このようなときは「棚卸実差修正を夜間バッチにする」「棚卸実差修正時は製品入庫処理を行わないように運用する」などの対応が必要になるでしょう。後者の運用の場合、プログラム間の排他処理を実装する方法も考えられます。