第5章

トランザクションとロック

ここでは、OracleとSQL Serverのトランザクションとロックの機能について記述します。

トランザクションの開始と終了

Oracleのトランザクションは自動的に始まり（デフォルト）、COMMITあるいはROLLBACKなどで終了します。しかし、SQL*PlusのAUTOCOMMITコマンドを使用すると、ステートメントごとにCOMMITすることも可能です。

SQL Serverでは、SET IMPLICIT_TRANSACTIONSオプションによってトランザクションの開始と終了のしかたが異なります。

- **SET IMPLICIT_TRANSACTIONS OFF**

  INSERT、UPDATE、DELETEステートメントが実行されるたびにトランザクションがCOMMITされます。SQL Serverのデフォルトは OFFです。複数のステートメントを1つのトランザクションに含めたい場合は、明示的にBEGIN TRANSACTIONを記述する必要があります。さらに、終了はCOMMIT TRANSACTIONを記述します。

  以下の例では、①および②の各ステートメントは終了時にCOMMITされますが、③と④はCOMMIT TRANSACTIONが終了するまでCOMMITされません。

```
INSERT      ……①
UPDATE      ……②
BEGIN TRANSACTION
UPDATE      ……③
INSERT      ……④
SELECT
COMMIT TRANSACTION
```

- **SET IMPLICIT_TRANSACTIONS ON**

  トランザクションは、以下のSQLステートメントが実行されると自動的に開始され、明示的にCOMMITあるいはROLLBACKされるまで終了しません。

ALTER TABLE	FETCH	REVOKE
CREATE	GRANT	SELECT
DELETE	INSERT	TRUNCATE TABLE
DROP	OPEN	UPDATE

トランザクションの中のSQLステートメント

Oracleのデータ定義言語（DDL）、データ制御言語（DCL）はそれぞれ1つのトランザクションとなり、ほかのSQLステートメントと同一のトランザクションになることはありません。

SQL Serverでは、次のステートメント以外のTransact-SQLステートメントを同一のトランザクションに含めることができます。

ALTER DATABASE	LOAD DATABASE
BACKUP LOG	LOAD TRANSACTION
CREATE DATABASE	RECONFIGURE
DISK INIT	RESTORE DATABASE
DROP DATABASE	RESTORE LOG
DUMP TRANSACTION	UPDATE STATISTICS

たとえば、ALTER TABLEなどもトランザクションに含めることができます。この場合は、トランザクション終了までテーブルにロックがかかるので注意が必要です。

データ読み取りのロック

Oracleでは、ロールバックセグメントにより読み取り一貫性が保証されているため、データの読み取り時（SELECT）にロックが（基本的には）発生しません。一方、SQL Serverにはロールバックセグメントに当たるものはなく、読み取り一貫性の保証はロックメカニズムを使用します。

つまり、Oracleではロールバックセグメントの管理とチューニングが重要であり、SQL Serverではロックメカニズムを理解したSQLステートメントの記述が重要になることがわかります（図5-1）。

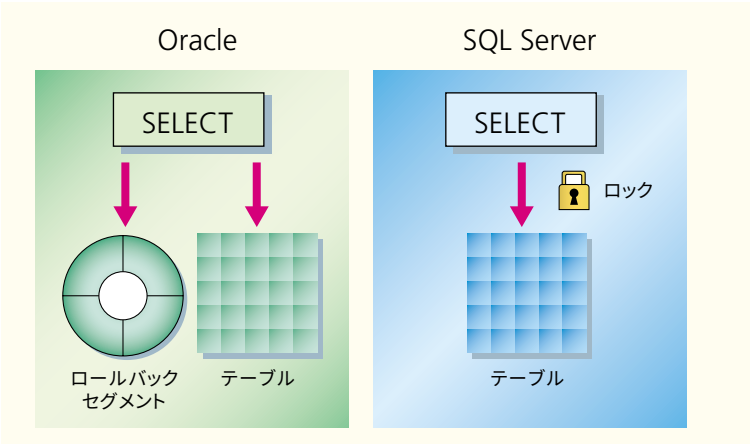

図5-1
データ読み取りのロック

ロック可能なリソース

基本的に、Oracleでは行ロックが使われ、特別なものとしてブロック、テーブルおよびパーティション単位にロックがかかります。また、ロックエスカレーションは行われません。

SQL Serverでは、最適なパフォーマンスを得るため、以下のリソースに動的なロックがかかります。

リソース	説明
RID	行識別子。テーブルの行をロックするために使用します。
キー	インデックス内の行ロック。直列化可能なトランザクションのキーの範囲を保護するために使用します。
ページ	8KBのデータページまたはインデックスページ
エクステント	8つのデータページまたはインデックスページの連続的な集合
テーブル	すべてのデータとインデックスを含むテーブル全体
DB	データベース全体

ロックの種類

Oracleの主なロックは、共有ロックと排他ロックです。

SQL Serverではさまざまな種類のロックがあり、これらをロックモードと呼んでいます。ロックモードには、以下の種類があります。

ロックモード	説明
共有 (S)	SELECTステートメントなど、データの変更や更新を行わない読み取り専用の処理で使用します。
排他 (X)	UPDATE、INSERT、DELETEなどのデータ変更の処理で使用します。1つのリソースを同時に更新しないようにするためのロックモードです。
更新 (U)	更新可能なリソースで使用します。共通の状況で起こるデッドロックを防ぎます。デッドロックは、複数のセッションがリソースを読み取り、ロックして、あとで更新する可能性がある場合に発生します。
インテント	ロック階層を設定するのに使用します。インテントロックにはいくつか異なる種類があります。インテント共有 (IS)、インテント排他 (IX)、およびインテント排他付き共有 (SIX) です。

（続き）

ロックモード	説明
スキーマ	テーブルのスキーマに依存する処理を行うときに使用します。スキーマロックには、スキーマ安定度（Sch-S）ロックとスキーマ修正（Sch-M）ロックの2種類があります。スキーマ修正ロックは、データ定義言語（DDL）の処理（列の追加やテーブルの削除）が行われているときに使用されます。 スキーマ安定ロックは、クエリをコンパイルする間だけ使用されます。
一括更新（BU）	データを一括でテーブルにコピーするとき、あるいはTABLOCKヒントが指定されたときに使用します。

ロックモードの互換性

SQL Serverでは、ある2つのトランザクションが同じリソースに共有ロックをかけた場合は同時に実行可能です。このことを「互換性がある」と言います。しかし、同じリソースに排他ロックをかけた場合は、あとからロックをかけたトランザクションは、ほかのトランザクションがロックを解放するまで待たされます。次の表は、主なロックモードの互換性を示しています。同じリソースに対して取得されるロックが、どのロックと互換性があるのかがわかります。

要求されたモード	すでにかけられているモード					
	IS	S	U	IX	SIX	X
インテント共有（IS）	可	可	可	可	可	不可
共有（S）	可	可	可	不可	不可	不可
更新（U）	可	可	不可	不可	不可	不可
インテント排他（IX）	可	不可	不可	可	不可	不可
インテント排他付き共有（SIX）	可	不可	不可	不可	不可	不可
排他（X）	不可	不可	不可	不可	不可	不可

そのほかのロックモードとの互換性は、以下のようになります。

- スキーマ安定度（Sch-S）ロックモードは、スキーマ修正（Sch-M）ロックモード以外のすべてのロックモードと互換性があります。
- スキーマ修正（Sch-M）ロックモードは、どのロックモードとも互換性がありません。
- 一括更新（BU）ロックモードは、スキーマ安定度ロックとほかの一括更新ロックモードとしか互換性がありません。

ロックの制御

SQL Serverでは、ロックマネージャが最適なリソースとロックモードを自動的に選択します。ユーザーがロックを制御するには、以下のような方法があります。

セッションレベルでの制御

SET TRANSACTION ISOLATION LEVELを使用すると、セッションごとにロックを制御できます。SQL Serverで指定可能なオプションは、以下の4つです。

- READ UNCOMMITTED
  共有ロックは実行されず、排他ロックも使用されないことを意味します。このオプションが設定されている場合には、コミットされていないデータ、つまりダーティデータを読み取ることができます。トランザクションが終了する前にデータ内の値を変更でき、行がデータセットに現れたり消えたりします。このオプションは、Oracleには実装されていません。
- READ COMMITTED
  データが読み取られている間、ダーティリードを回避するために共有ロックが保持されるように指定します。しかし、トランザクションが終了する前にデータを変更することができるので、反復不能読み取りやファントムデータが発生する可能性があります。このオプションは、OracleおよびSQL Serverの既定値です。
- REPEATABLE READ
  クエリで使用されるすべてのデータがロックされるので、ほかのユーザーはデータを更新できません。しかし、別のユーザーが新しいファントム行をデータセットに挿入し、現在のトランザクションでの後続の読み取りにそのファントム行を挿入することは可能です。このオプションは、Oracleには実装されていません。
- SERIALIZABLE
  範囲ロックがデータセットに適用されるので、トランザクションが完了するまで、ほかのユーザーはデータセットに行を挿入したり、更新したりすることができません。

テーブルレベルでの制御

オプティマイザヒントを使うとテーブルごとにロックを制御できます。たとえば、以下のUPDATEステートメントはテーブル全体に排他ロックをかけます。さらに、トランザクション終了まで保持します。

```
UPDATE Products (TABLOCKX HOLDLOCK)
SET price = price * 0.95
WHERE price > 10000
```

表：トランザクションの分離レベル

分離レベル	分離性に関する問題		
	ダーティリード	反復不能読み取り	ファントムリード
READ UNCOMMITTED	発生する	発生する	発生する
READ COMMITTED	発生しない	発生する	発生する
REPEATABLE READ	発生しない	発生しない	発生する
SERIALIZABLE	発生しない	発生しない	発生しない

ロックのヒントには、以下のものがあります。

ロックのヒント	説明
NOLOCK	READ UNCOMMITEDと同じです。
READUNCOMMITTED	NOLOCKと同じです。
READCOMMITTED	READ COMMITTEDと同じです。SQL Serverのデフォルトです。
REPEATABLEREAD	REPEATABLE READと同じです。
HOLDLOCK	SERIALIZABLEと同じです。
SERIALIZABLE	SERIALIZABLEの分離レベルで動作するトランザクションと同じロックセマンティクスでスキャンを実行します。HOLDLOCKに相当します。
ROWLOCK	行レベルのロックを使用します。
PAGLOCK	ページレベルのロックを使用します。
TABLOCK	テーブルレベルのロックを使用します。
TABLOCKX	テーブルに排他ロックを使用します。
READPAST	ロックされた行をスキップします。このオプションを指定すると、対象のトランザクションは、ほかのトランザクションが行ロックを解除するまでブロックされることはなくなり、ほかのトランザクションによってロックされた、通常は結果セットに含まれる行をスキップします。READPASTロックヒントは、READ COMMITTEDの分離レベルで動作するトランザクションにだけ適用され、このロックヒントは行レベルのロックを読み飛ばします。SELECTステートメントにだけ適用されます。

（続き）

ロックのヒント	説明
UPDLOCK	テーブルの読み出しの間、共有ロックの代わりに更新ロックを使用し、ステートメントが終了するまで、あるいはトランザクション終了まで保持します。UPDLOCKには、ほかのユーザーがデータを読むことをブロックせずにデータを読み取ることができます。また、読み取り以降にデータが変更されていないという保証があるため、あとでそのデータを更新できるという利点があります。
XLOCK	ステートメントで処理されるすべてのデータについて、トランザクションの終了まで保持される排他ロックを使用します。このロックは、PAGLOCKまたはTABLOCKで指定できます。この場合、排他ロックは適切な粒度に適用されます。

ロックタイムアウト

SQL Serverでは、別のトランザクションがすでにリソースに対してロックしていると、そのリソースに関する互換性のないロックをトランザクションに対して与えることができません。これがブロックです。このブロックの時間が長時間に及ぶと、デッドロックの発生原因となります。デッドロックが発生した場合、トランザクションは自動的にロールバックされます。これを未然に防ぐために、ブロックが起きたときにタイムアウトを設定することができます。以下の設定は、ブロックが起きたときに10秒間待つ指定です。

```
SET LOCK_TIMEOUT 10000
```

これはセッションごとに指定可能で、デフォルトの値は -1で無限に待つことになります。現在の値を調べるには、@@LOCK_TIMEOUT関数を使用します。タイムアウトが発生した場合は、アプリケーションにエラー番号1222を返します。

ロック情報の表示

SQL Serverでは、現在のロック状況をSQL Server Enterprise Managerあるいはsp_lockシステムストアドプロシージャなどで表示できます。図5-2の画面は、ID59番のプロセスがID58番のプロセスのロックを待っている状態を示しています。これは、ID58が「ブロックしている」ケースとなっています。

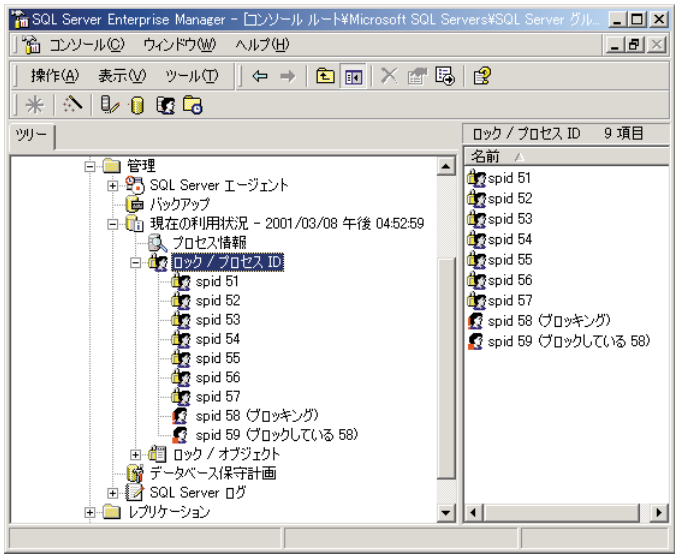

図 5-2
ロックの表示
(SQL Server Enterprise Manager)