行ロックを行っているセッションの特定と対処法

ロック発生時の対応

対応手順１.

ロック発生セッションとロック元セッションの関係調査

対応手順２.

ロック発生 SQL 文とロック元 SQL 文の調査

対応手順３.

ロック元オブジェクトが路地しているロック実行の対象リソース（オブジェクト）

V$LOCK ビューの主な列値

列　　名	説　　明
sid	セッション ID
type	ロックの種別 TX：　トランザクション同士のロック競合 （レコード・ロック） TM：　SQL がロックをかけてしまい以降の処理が止まっている　（テーブルロック） UL：　アプリケーションにて、明示的に DBMS_LOCK パッケージを使ってのロック制御呼出し （並行処理防止のためのロック機構） CF：　制御ファイルトランザクションエンキュー （D/B 内部システムによるロック） JQ：　ジョブキュー関連のエンキュー （D/B 内部システムによるロック） ST：　領域管理トランザクションエンキュー （D/B 内部システムによるロック） SQ：　シーケンス関連のエンキュー （D/B 内部システムによるロック） .
lmode	保持しているロックのモード
request	要求しているロックのモード **ロック未獲得** 他処理のロック解放を待ち状態の可能性あり もしくは、今まさにロック取得する瞬間
block	lmode の保持しているロックが、他のセッションの SQL 文の処理を邪魔（待機状態）しているかどうかを表す 0：他のどんな処理も邪魔していない （待機状態にさせていない） 1：他の処理を**待機させてしまっている**

lmode 列と request 列の値の種類とその意味

列 値	記号／旧記号 名称	ロックの説明	発行 SQL
0	なし	なし	なし
1	NULL／ NULL ロック	なし	Select
2	RS／SS 行共有ロック	表の定義変更や行の UPDATE 待ちを行わせるロック（緩い）	Select　FOR UPDATE lock table　ROW　SHARE　MODE
3	RX／SX 行排他ロック	行更新などの操作の後、他者に対しては、更新前の時点の一貫性あるデータを参照させる 他者の更新には、行排他ロック解除まで待たせるロック	insert update delete lock table　ROW　EXCLUSIVE　MODE
4	S 共有ロック	Create INDEX 操作中のロック 他者のロック操作や更新操作を待機させるロック	lock table　IN　SHARE　MODE
5	SRX／SSX 共有行排他ロック	他者の Select 以外の表へのアクセス待機を行わせるロック	lock table　IN　SHARE　ROW EXCLUSIVE　MODE
6	X 排他ロック	他者への表へのアクセス待機を行わせるロック	lock table　IN　EXCLUSIVE　MODE

ロック状態の調査方法

```
SELECT　SE.sid ,　LK.type ,　LK.lmode ,　LK.request ,　LK.block ,　SE.program
　FROM　V$SESSION　SE ,　V$LOCK　LK
　WHERE　SE.sid　=　LK.sid　;
```

レコード・ロックだけの状況を調査する場合は、

```
　　AND　LK.type　=　'TX'
```

TX: トランザクション同士のロック競合（レコード・ロック）

実行例）

SID	TYPE	LMODE	REQUEST	BLOCK
101	TX	6	0	1
103	TX	6	0	1
103	TX	0	6	0
108	TX	0	6	0

セッション 103 は、行ロックを１つ獲得していると同時に
１つの行ロックを待つ待機イベント中である
この場合は、V$LOCK では、２レコードに分かれて出力される

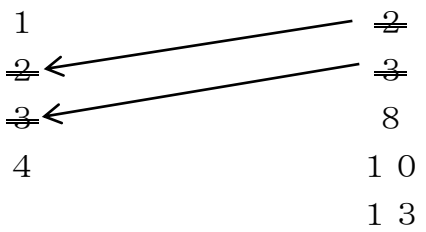
【ロック待ち待機イベントのロック元セッションの調査方法】

※個別のセッションに対するロック元セッションの調査については、このドキュメントの１２ページ目を参照のこと

(1). ロックを保持して、他のセッションをブロックしているセッションの SID 一覧を作成する

(2). ロックを**要求中（待機）**しているセッションの SID 一覧を作成する

(3). （1）のロックを保持している SID 一覧から（2）ロックを要求中（待機）しているセッションのを消し込むと、ロックを保持して Active なセッションが残る
これが、原因元のセッションである



保持のレコード要素の中で、要求中のレコード要素で消込みが出来なかった SID 値は、他にロックを行わせ自身は、ロック待無しで Active 状態にある

調査例）

(1).保持＆ブロックのセッション

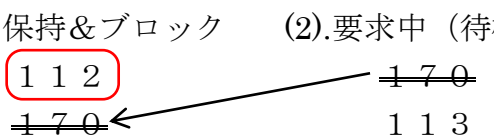
```
SELECT   SE.sid ,   LK.type ,   LK.lmode ,   LK.request ,   LK.block ,
         SE.program
  FROM   V$SESSION   SE ,   V$LOCK   LK
 WHERE   SE.sid   =   LK.sid
   AND   LK.lmode   <>   0
   AND   LK.block     !=   0
 ORDER   BY   SE.sid ;
```

SID	TYPE	LMODE	REQUEST	LOCK
112	TX	6	0	1
170	TX	6	0	1

(2).要求中（待機）のセッション

```
SELECT   SE.sid ,   LK.type ,   LK.lmode ,   LK.request ,   LK.block
         SE.program
FROM   V$SESSION   SE ,   V$LOCK   LK
 WHERE   SE.sid   =   LK.sid
   AND   LK.request   !=   0
 ORDER   BY   SE.sid ;
```

SID	TYPE	LMODE	REQUEST	LOCK
170	TX	0	6	0
113	TX	0	6	0



この結果、SID＝１２０が、ロック元のセッションと考えられる

参考

ロック保持はしているが他のセッションには影響しない（ブロックはしていない）

セッション情報

```
SELECT  SE.sid ,  LK.type ,  LK.lmode ,  LK.request ,  LK.block ,
        SE.program
  FROM  V$SESSION  SE ,  V$LOCK  LK
 WHERE  SE.sid  =  LK.sid
  AND   LK.lmode   <>  0
  AND   LK.request  =  0
  AND   LK.block    =  0
 ORDER  BY  SE.sid ;
```

SID	TYPE	LMODE	REQUEST	LOCK
113	AE	4	0	0
125	XR	1	0	0
159	CF	2	0	0
159	RS	2	0	0

排他制御ロックの種類とコード

ロックの種類

（1）DML ロック（UPDATE、INSERT、DELETE、SELECT FOR UPDATE）

（2）DDL ロック（CREATE、ALTER、DROP オブジェクト）

（3）ラッチと内部ロック

内部データベース構造とメモリ構造を保護する目的のロック、ユーザーが意識する必要なし

次ページ

DDL ロック一覧

ロックの種類	内　　容
排他 DDL ロック	オブジェクトに対する定義内容の同時変更を防ぎます このロックは、他者からの定義内容の参照にもロック解除待ちが発生します 例）２者によるテーブル列の同時追加防止
共有 DDL ロック	DDL 文実行時に、その定義を変更されることを防ぎます このロックでは、他者からの定義内容の参照にはロック解除待ちは発生しません 例）オブジェクトのコピー操作中の定義変更防止
ブレーク可能解析ロック	DML 実行時に、その定義を変更されることを防ぎます このロックは、後からの排他 DDL ロックにはロック解除待ちが発生します 例）実行計画作成中での列定義変更防止

DML ロックのモード一覧

【排他制御ロック】　ロックの種類と内容、およびロック実施 SQL ステートメント

ロック名称	Type	略称	コード	説明
排他行ロック	TX		6	INSERT、UPDATE、DELETE、SELECT　FOR　UPDATE 行に対する排他的なロック
行共有テーブルロック	TM	RS	2	LOCK　TABLE　<テーブル名>　IN　ROW　SHARE　MODE テーブルに対する共有ロック（他者からの SELECT は許可） ロック単位は、行になります
行排他テーブルロック	TM	RX	3	INSERT、UPDATE、DELETE、SELECT FOR UPDATE、 LOCK　TABLE　<テーブル名>　IN　ROW　EXCLUSIVE　MODE COMMIT 前のデータであり、他者への SELECT　FOR　UPDATE は許可しない
共有テーブルロック	TM	S	4	LOCK　TABLE　<テーブル名>　IN　SHARE　MODE テーブルに対する共有可能ロックです（他者からの SELECT は許可） 他のトランザクションによるデータの更新を防止します ロック単位は、テーブル全体になります ※ 自身のトランザクションでもデータ更新は出来ず、一貫性のあるデータ読取りを行う目的で使用されます
共有行排他テーブルロック	TM	SRX	5	LOCK TABLE　<テーブル名> IN　SHARE　ROW　EXCLUSIVE MODE テーブルに対する共有可能ロックです（他者からの SELECT は許可） 他のトランザクションによるデータの更新を防止します ロック単位は、テーブル全体になります ※ 自身のトランザクションだけで排他的なデータ更新を行うための目的で使用されます
排他テーブルロック	TM	X	6	LOCK　TABLE　<テーブル名>　IN　EXCLUSIVE　MODE すべてのレコードに対する排他的なロックです 他者からの SELECT は、ロック解除まで待たせます ロック単位は、テーブル全体になります

SELECT　type,　lmode,　request　FROM　V$LOCK　;

V$LOCK の type 列の値が示すロックの種類

ロック名称	Type	略称	コード	説明
	TX			トランザクション同士のロック競合 (レコード・ロック)
	TM			SQL がロックをかけてしまい以降の処理が止まっている (テーブルロック)
	UL			アプリケーションにて、明示的に DBMS_LOCK パッケージを使ってのロッ 制御呼出し (並行処理防止のためのロック機構)
	CF			制御ファイルトランザクションエンキュー (D/B 内部システムによるロック)
	JQ			ジョブキュー関連のエンキュー (D/B 内部システムによるロック)
	ST			領域管理トランザクションエンキュー (D/B 内部システムによるロック)
	SQ			シーケンス関連のエンキュー (D/B 内部システムによるロック)

SELECT　type,　lmode,　request　FROM　V$LOCK　;

排他制御ロック待ちの調査

※ 過去のロックについての状況を調査する場合には、
DBA_HIST_ACTIVE_SESS_HISTORY と DBA_HIST_SQLTEXT~~、DBA_LOCK~~
を使用する

```
SELECT   ses.sid ,   ses.serial# ,   lk.type ,   lk.lmode ,   lk.request ,   lk.ctime ,
         lk.block ,   ses.blocking_session ,
         substr(dbo.object_name, 0, 10)   , substr(sql.sql_text, 0, 50) ,   ses.program
  FROM   v$lock   lk ,   v$session   ses ,   dba_objects   dbo ,   v$sql   sql
 WHERE      lk.type   like   'T%'
         AND   ses.type   =   'USER'
         AND   lk.id1   =   dbo.object_id
         AND   lk.sid   =   ses.sid
         AND   ses.sql_address   =   sql.address (+)
         AND   ses.sql_hash_value   =   sql.hash_value (+)   ;
```

1. ses.sid 「セッション ID」
2. ses.serial# 「シリアル番号」 Oracle のトランザクション管理用シーケンス番号
3. lk.type 「ロックの略称」
4. lk.lmode 「ロックのコード（取得済）」
5. lk.request 「ロックのコード（解除待ち）」
   ※ ロック待ちの時も値が入ってこなかった
6. lk.ctime 「ロック取得継続時間」
   or 「ロック要求継続時間（ロック待ち時間）」
7. lk.block 「ブロックしている要求の有無」
8. dbo.object_name 「ロックしているオブジェクト名」
   or 「ロックを要求しているオブジェクト名」
9. blocking_session 「ロックを保持していて、解除を待たせているセッション ID」
10. sql.sql_text 「ロックを要求している SQL 文」
    ※ ロックを取得している側は、先の処理に進んでいるので、この SQL 文の内容は当てにならない

V$LOCK の構成列

列（カラム）	内　　容
sid	セッション ID（識別子）
type	ロックの略称（**ロックの種類**）　　次ページ参照 数値
lmode	保持（取得済）しているロックのコード ロック解除待ちのセッションは、0
request	ロック要求していて、ロック待ちが発生しているロックのコード ロックを取得済の場合は、0 ―2019/10/20 テスト結果― 　**ロック待ちの時でも、0 であった**
ctime	経過時間
block	他者をブロックしている場合は、 1 他者をブロックしていない場合は、0 ―2019/10/20 テスト結果― 　**ブロックしている場合でも、0 であった**

2019/10/20 のテスト結果

端末 A)

```
SELECT * FROM EMP_TABLE WHERE EMPNO = 1 FOR UPDATE ;
```

端末 B)

```
SELECT * FROM EMP_TABLE WHERE EMPNO = 2 FOR UPDATE ;
SELECT * FROM EMP_TABLE WHERE EMPNO = 1 FOR UPDATE ;
```

端末 A）での COMMIT 待ちで、ロックが発生

端末 C)

```
SELECT * FROM EMP_TABLE WHERE EMPNO = 2 FOR UPDATE ;
```

端末 B）での COMMIT 待ちで、ロックが発生

この時の前ページの SQL コマンド実行

SID	SERIAL#	TY	LMODE	REQUEST	CTIME	BLOCK	OBJECT_NAM
100	63	TM	3	0	749	0	EMP_TABLE
111	2507	TM	3	0	178	0	EMP_TABLE
128	3828	TM	3	0	17	0	EMP_TABLE

BLOCKING_SESSION	SQL_TEXT
(NULL)	(NULL)
100	select * from emp where empno=1 for update
111	select * from emp where empno=2 for update

ブロック元となっているセッション番号

自身のセッションの（ブロック解放待ち状態の）実行 SQL 文

BLOCKING_SESSION が NULL のセッションが、原因元（ロック元）のセッションとなる

なお、原因元（ロック元）の SQL_TEXT は、今処理中の SQL 文であるが、ロックの原因の SQL 文とはならないことがある⟶（次の処理に進んでしまっている場合）

【考察】

端末 B、C）について　　**【ロック待ちタスク側】**

TYPE 列にロックタイプの値が入ってくる

ロック待ちが発生すると、BLOCKING_SESSION 列にロックを保持している原因元のセッション ID が入ってくる

この２つから**ロック待ちが発生**していると判断できる

また、この時の SQL_TEXT 列の値には、実行しようとしていた SELECT 文が入ってくる

ロック待ちが発生した時の REQUEST 列は、「０」であったので、使用できない

端末 B）は、端末 A）の COMMIT を待っているが、端末 C）に対しては待たせている立場になっている

端末 A）について　　**【ロック保持タスク側】**

TYPE 列にロックタイプの値が入ってくる

ロック保持だけである場合ならば、BLOCKING_SESSION 列は NULL である

この２つから**ロックを保持している**と判断できる

端末 B）をロック待ちにさせていても、BLOCK 列は、「０」であったので、使用できない

ロックを保持している方の SQL_TEXT 列の情報は、処理が先に進んでいるので、ここに表示される SQL 文がロック元ということにはならない。また入力待ちなどの状態である場合には NULL のことがある

【ロックが発生している元凶となるプロセスの調査方法】

特定端末のロックの原因元調査（ロック発生の端末が分かっている場合）

select distinct terminal, osuser, username　from v$session order by terminal;

※ 過去のロックについての状況を調査する場合には、
DBA_HIST_ACTIVE_SESS_HISTORY と DBA_HIST_SQLTEXT、DBA_LOCK を使用する

```
DECLARE
  terminal_data            VARCHAR2(20);
  sid_data                 NUMBER;
  blocking_session_data    NUMBER;
  sql_address_data         VARCHAR2(8);
  sql_hash_value_data      NUMBER;

  object_name_data         dba_objects.object_name%TYPE  ;
  sql_text_data            v$sql.sql_text%TYPE  ;
  CURSOR  c_lock  IS
  SELECT  dbo.object_name  , sql.sql_text
    FROM  v$lock  lk ,  v$session  ses ,  dba_objects  dbo ,  v$sql  sql
    WHERE  ses.sid  =  sid_data
      AND  lk.id1  =  dbo.object_id
      AND  lk.sid  =  ses.sid
      AND  ses.sql_address  =  sql.address ( + )
      AND  ses.sql_hash_value = sql.hash_value ( + )  ;

BEGIN
-- ■端末指定
  SELECT  ses.blocking_session ,  sid  INTO  blocking_session_data , sid_data
    FROM  v$session  ses
    WHERE     ses.terminal  =  '&INPUT_terminal_name'
      AND     ses.blocking_session  is  not  null ;
  DBMS_OUTPUT.PUT_LINE('ロック元セッション : ' || blocking_session_data || ' ← SID :
                                    ' || sid_data );

-- ■ロック対象情報の出力
        --  カーソル・オープン
  OPEN  c_lock  ;
  LOOP
        --  フェッチ
```

```
    FETCH   c_lock   INTO   object_name_data ,   sql_text_data ;
        --   フェッチの終了判断
    EXIT   WHEN   c_lock%NOTFOUND   ;
    /*    データが存在した時の処理    */
    /*    読込んだデータは、「emp_record. 列名」の中にセットされている    */
    DBMS_OUTPUT.PUT_LINE(' オブジェクト : ' || object_name_data || '   ＳＱＬ文 : ' ||
                                sql_text_data );
  END   LOOP   ;
        --   カーソル・クローズ
  CLOSE   c_lock   ;
  DBMS_OUTPUT.PUT_LINE('.    ');

-- ■ロック元セッションへの追究
  LOOP
    EXIT   WHEN   blocking_session_data is NULL   ;   --   フェッチの終了判断
    SELECT   ses.blocking_session ,   ses.sid   INTO   blocking_session_data , sid_data
      FROM   v$session   ses
      WHERE     ses.sid   =   blocking_session_data ;
    DBMS_OUTPUT.PUT_LINE('ロック元セッション : ' || blocking_session_data || ' ←
                                SID : ' || sid_data );
    DBMS_OUTPUT.PUT_LINE('.    ');
  END   LOOP   ;

-- ■ロック元の SQL 文調査
  OPEN   c_lock   ;                                          --   カーソル・オープン
  LOOP
    FETCH   c_lock   INTO   object_name_data ,   sql_text_data ;        --   フェッチ
    EXIT   WHEN   c_lock%NOTFOUND   ;     --   フェッチの終了判断
    /*    データが存在した時の処理    */
    /*    読込んだデータは、「emp_record. 列名」の中にセットされている    */
    DBMS_OUTPUT.PUT_LINE(' オブジェクト : ' || object_name_data || '   ＳＱＬ文 : ' ||
                                sql_text_data );
  END   LOOP   ;
  CLOSE   c_lock   ;                                         --   カーソル・クローズ
  DBMS_OUTPUT.PUT_LINE('.    ');
END;
/
```

ロックが発生している
端末名を入力する

input_terminal_name に値を入力してください: BUSINESS2-PC

注 意） 全文字大文字で

旧 26: WHERE ses.terminal = '&INPUT_terminal_name'
新 26: WHERE ses.terminal = 'BUSINESS2-PC'

＊＊＊ 実行結果 ＊＊＊

１行目は、指定した端末のセッションの直接のロック情報

ロック元セッション：170 ← SID：113
オブジェクト：ORA$BASE ＳＱＬ文：select * from emp where empno=2 for update
オブジェクト：EMP ＳＱＬ文：select * from emp where empno=2 for update
.
ロック元セッション：120 ← SID：170
・
・
・
・
・

途中の行は、ロックを行っているセッションを上位のロックへ遡った情報のみ
ロックとなった SQL 文は、**表示されない**

最後（ロック元セッション＝NULL）の行は、大元のロック

.
ロック元セッション： ← SID：112
オブジェクト：ORA$BASE ＳＱＬ文：SELECT DBO.OBJECT_NAME ,
SQL.SQL_TEXT FROM V$LOCK LK ,
オブジェクト：EMP ＳＱＬ文：SELECT DBO.OBJECT_NAME , SQL.SQL_TEXT
FROM V$LOCK LK , V$SES

※ ロック元のセッションの SQL 文は、最新の SQL 文が表示されている。ロック元となった SQL 文は、これ以前に実行された過去の SQL 文の可能性がある
よって、ロック元の SQL 文を決定するには、ロック元の対象セッションが実行してきた SQL 文を、時間を遡って洗い出し、各 SQL 文が原因になっていないか個々に検討する必要がある

ロック元セッションが確保（ロック）しているオブジェクトの名前調査

~~※ 過去のロックについての状況を調査する場合には、~~
~~DBA_HIST_ACTIVE_SESS_HISTORY と DBA_HIST_SQLTEXT、DBA_LOCK を使用する~~

```
SELECT   ses.sid ,   ses.serial# ,   substr(dbo.object_name, 0, 10) ,   lk.ADDR ,
         lk.KADDR ,   ses.program
    FROM   v$lock   lk ,   v$session   ses ,   dba_objects   dbo ,   v$sql   sql
   WHERE      ses.sid   =   '&ロック元セッション番号'
         AND   ses.type   =   'USER'
         AND   lk.id1   =   dbo.object_id
         AND   lk.sid   =   ses.sid
         AND   ses.sql_address   =   sql.address (+)
         AND   ses.sql_hash_value   =   sql.hash_value (+)   ;
```

SID	SUBSTR(DBO.OBJECT_NA	ADDR	KADDR
112	EMP	141FA25C	141FA28C
112	DEPT	141FA25C	141FA28C

ロック確保しているオブジェクトのロック情報が表示される

ロックされている側が、どちらのオブジェクトに影響しているかは、ロックされた側の待機中 SQL 文から判断を行う