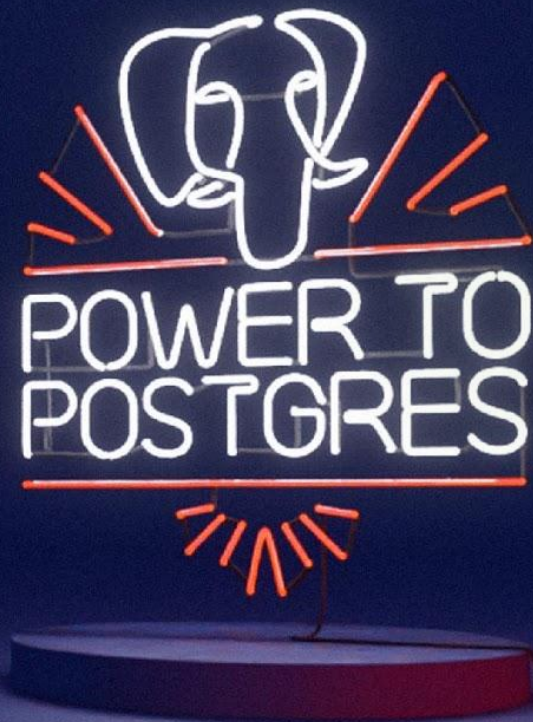


PostgreSQLのロックマネージャをアンロックする

Koichi Suzuki

Mar. 30, 2022

Apr. 13, 2022



謝辞

この Webinar は、PostgreSQL Evangelist である Bruce Momjian 氏の全面的な協力で、氏の持つプレゼンテーション資料を利用して行うものです

オリジナルの資料は

<https://momjian.us/main/writings/pgsql/locking.pdf> で公開されています

オリジナルのSQL文は

<https://momjian.us/main/writings/pgsql/locking.sql> で公開されています



内容

PostgreSQL 内部のロックの使い方、調べ方及び実装の一部を説明します

- PostgreSQL のロックの概要
- ロックの例
- デッドロックその他のロック廻りの話題



自己紹介

- NTT 研究所, NTT DATA, NTT DATA 先端技術にて
 - Unix の日本語化
 - Oracle 移植と日本語化
 - UniSQL オブジェクトリレーショナルデータベース開発
 - PostgreSQL (NTT OSS センター)
 - Postgres-XC, Postgres-XL 水平分散型データベースリーダー
- 2ndQuadrant
- 現在 EDB メンバ
 - Senior Principal Support Engineer



ロックが好きな人はいないでしょうが、正しくデータベースを動かすには必要なのです





PostgreSQL のロックの型

ACCESS SHARE

ROW SHARE

ROW EXCLUSIVE

SHARE UPDATE EXCLUSIVE

SHARE

SHARE ROW EXCLUSIVE

EXCLUSIVE

ACCESS EXCLUSIVE

Share/Exclusive の型

ACCESS SHARE

ROW SHARE

ROW EXCLUSIVE

SHARE UPDATE EXCLUSIVE

SHARE

SHARE ROW EXCLUSIVE

EXCLUSIVE

ACCESS EXCLUSIVE

Row/Access 型

ACCESS SHARE

ROW SHARE

ROW EXCLUSIVE

SHARE UPDATE EXCLUSIVE

SHARE

SHARE ROW EXCLUSIVE

EXCLUSIVE

ACCESS EXCLUSIVE

ロックモードの概要

ロックモード	主な取得契機	概要
ACCESS SHARE	SELECT文	テーブルの読み取りのみで変更を行わない場合
ROW SHARE	SELECT FOR UPDATE (FETCHされていない行) SELECT FOR SHARE	他のトランザクションによる行変更を保護する
ROW EXCLUSIVE	UPDATE/DELETE/INSERT	他のトランザクションによる行の読み込みを保護する
SHARE UPDATE EXCLUSIVE	VACUUM/ANALYZE/CREATE INDEX CONCURRENTLY, REINDEX CONCURRENTLY, CREATE STATISTICS など	スキーマ変更及び VACUUM コマンドの実行から保護する

ロックモードの概要

ロックモード	主な取得契機	概要
SHARE	CREATE INDEX (CONCURRENTLYなし)	同時実行されるデータ変更から保護する
SHARE ROW EXCLUSIVE	CREATE TRIGGER, ALTER TABLE の一部	一つのセッションだけが保持できる。同時実行されるデータ変更から保護。
EXCLUSIVE	REFRESH MATERIALIZED VIEW CONCURRENTLY	他のトランザクションがデータの読み取りのみができるように保護
ACCESS EXCLUSIVE	DROP TABLE /TRUNCATE /REINDEX /CLUSTER /VACUUM FULL /REFRESH MATERIALIZED VIEW/ LOCK TABLE	ロック保持者以外にテーブルにアクセスするトランザクションがないことを補償

ロックの競合

要求するロックモード	既存のロックモード							
	ACCESS SHARE	ROW SHARE	ROW EXCL.	SHARE UPDATE EXCL.	SHARE	SHARE ROW EXCL.	EXCL.	ACCESS EXCL.
ACCESS SHARE								X
ROW SHARE							X	X
ROW EXCL.					X	X	X	X
SHARE UPDATE EXCL.				X	X	X	X	X
SHARE			X	X		X	X	X
SHARE ROW EXCL.			X	X	X	X	X	X
EXCL.		X	X	X	X	X	X	X
ACCESS EXCL.	X	X	X	X	X	X	X	X

<https://www.postgresql.jp/document/13/html/explicit-locking.html>

二相ロックプロトコル

- トランザクションで確保したロックはそのトランザクションが完了する (Commit/Abort) するまで、開放することはありません
- Commit/Abort の契機で確保したロックは一気に解放します
- トランザクション内でロックを解放することで、未確定の更新が他のトランザクションから見えて、データの一貫性が失われることを防ぐためです

PostgreSQL 内部には、別なレイヤの内部ロックがあります (LWLock, Spinlock)。これらは、PostgreSQL の内部リソースの保護に使われており、二相ロックプロトコルには従っていません。

増永: リレーショナルデータベース入門 (2003), p.290
Hector Garcia-Molina: Database Systems, the complete book (2002), p.936

トランザクションID



<https://fyparts.jp/archives/2150>

赤にご注目下さい



<https://www.flickr.com/photos/alltheaces/>

使われているプロセス番号 (PID, process identifier)

```
SELECT pg_backend_pid();  
pg_backend_pid  
-----  
1525593
```

仮想トランザクションID

```
SELECT virtualtransaction AS vxid, transactionid::text
FROM
pg_locks
WHERE pid = pg_backend_pid()
ORDER BY 1, 2
LIMIT 1;
 vxid  | transactionid
-----+-----
 3/174 |
```

3はバックエンドidです。174はこのバックエンドの仮想トランザクション ID です。つまり、バックエンド id/バックエンド-ローカルなID で仮想トランザクション IDを表現しています

バックエンドID を調べる

```
SELECT *  
FROM    pg_stat_get_backend_idset() AS t(id)  
WHERE   pg_stat_get_backend_pid(id) = pg_backend_pid();  
      id  
-----  
      3
```

バックエンドID の最大所は *max_connections* で設定します

VXID は増加していきます

```
SELECT virtualtransaction AS vxid, transactionid::text
FROM pg_locks
WHERE pid = pg_backend_pid()
ORDER BY 1, 2
LIMIT 1;
 vxid | transactionid
-----+-----
 3/176 |
(1 row)
```

```
SELECT virtualtransaction AS vxid, transactionid::text
FROM pg_locks
WHERE pid = pg_backend_pid()
ORDER BY 1, 2
LIMIT 1;
 vxid | transactionid
-----+-----
 3/177 |
```

VXID は増加していきます

```
SELECT virtualtransaction AS vxid, transactionid::text
FROM pg_locks
WHERE pid = pg_backend_pid()
ORDER BY 1, 2
LIMIT 1;
 vxid | transactionid
-----+-----
 3/176 |
(1 row)
```

```
SELECT virtualtransaction AS vxid, transactionid::text
FROM pg_locks
WHERE pid = pg_backend_pid()
ORDER BY 1, 2
LIMIT 1;
 vxid | transactionid
-----+-----
 3/177 |
```

実際の/外部の/仮想でない XID を割り当てる

```
BEGIN WORK;
```

```
SELECT virtualtransaction AS vxid, transactionid::text
FROM   pg_locks
WHERE  pid = pg_backend_pid()
ORDER BY 1, 2
LIMIT 1;
```

vxid	transactionid
3/178	

```
ANALYZE pg_language;
```

実際の/外部の/仮想でない XID を割り当てる

```
SELECT virtualtransaction AS vxid, transactionid::text
FROM pg_locks
WHERE pid = pg_backend_pid()
ORDER BY 1, 2
LIMIT 1;
```

```
vxid | transactionid
-----+-----
3/178 | 27052726
```

```
SELECT txid_current();
txid_current
-----
27052726
```

```
Time: 0.597 ms
koichi=*# COMMIT;
```

トランザクションidの範囲は3から40億 (2^{32}) です。ゼロは不正トランザクションIDであり、1と2はフリーズしたxidに使います (それぞれコミット及びアボートしたもの)。

XID をリクエストして割り当てる

```
SELECT virtualtransaction AS vxid, transactionid::text
FROM      pg_locks
WHERE     pid = pg_backend_pid()
ORDER BY 1, 2
LIMIT 1;
```

```
vxid | transactionid
-----+-----
3/180 |
```

-- これで仮想でない xid を（もし割り当てられていないなら）割り当てる

```
SELECT txid_current();
txid_current
```

```
-----
27052728
```

XID をリクエストして割り当てる

```
SELECT virtualtransaction AS vxid, transactionid::text
FROM   pg_locks
WHERE  pid = pg_backend_pid()
ORDER BY 1, 2
LIMIT 1;
 vxid  | transactionid
-----+-----
 3/180 | 27052728

COMMIT;
```

ロックの型



設定: lockview ビューを作る

```
-- 他のセッションでも使うので、一時ビューではまずい
CREATE VIEW lockview AS
SELECT  pid, virtualtransaction AS vxid, locktype AS lock_type,
        mode AS lock_mode, granted,
        CASE
            WHEN virtualxid IS NOT NULL AND transactionid IS NOT NULL
            THEN      virtualxid || ' ' || transactionid
            WHEN virtualxid::text IS NOT NULL
            THEN      virtualxid
            ELSE      transactionid::text
        END AS xid_lock,
        relname, page, tuple, classid, objid, objsubid
FROM    pg_locks LEFT OUTER JOIN pg_class ON (pg_locks.relation = pg_class.oid)
WHERE   -- このビューのロックは表示しないようにする
        pid != pg_backend_pid() AND
        -- 自分の vxid へのロックは表示不要
        virtualtransaction IS DISTINCT FROM virtualxid
-- granted での並び順を優先するようにする
ORDER BY 1, 2, 5 DESC, 6, 3, 4, 7;
```

lockview1 ビューを作る

```
CREATE VIEW lockview1 AS
SELECT  pid, vxid, lock_type, lock_mode,
        granted, xid_lock, relname
FROM    lockview
-- granted でまず並べ替えるようにする
ORDER BY 1, 2, 5 DESC, 6, 3, 4, 7;
```

```
CREATE VIEW lockview2 AS
SELECT  pid, vxid, lock_type, page,
        tuple, classid, objid, objsubid
FROM    lockview
-- granted が最初
-- lockview の順番にマッチする非表示のカラムを追加する
ORDER BY 1, 2, granted DESC, vxid, xid_lock::text, 3, 4, 5, 6, 7, 8;
```

```
CREATE TABLE lockdemo (col int);
```

```
INSERT INTO lockdemo VALUES (1);
```

```
BEGIN WORK;
```

```
LOCK TABLE lockdemo IN ACCESS SHARE MODE;
```

```
-- force future psql commands to use the current database  
\setenv PGDATABASE :DBNAME
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/189	relation	AccessShareLock	t		lockdemo

```
\! psql -e -c 'SELECT * FROM lockview2;' | sed 's/^/\t/g'
SELECT * FROM lockview2;
    pid    | vxid  | lock_type | page | tuple | classid | objid | objsubid
-----+-----+-----+-----+-----+-----+-----+-----
  1553323 | 3/189 | relation  |      |       |         |      |
COMMIT;
```

以降、意味のある情報がある場合は *lockview2* のみを示すことにする

```
BEGIN WORK;
```

```
SELECT * FROM lockdemo;
```

```
col
-----
1
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^\t/g'
```

```
\timing on
```

```
Timing is on.
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/190	relation	AccessShareLock	t		lockdemo

```
COMMIT;
```

複数テーブルの Access Share ロッキング

```
BEGIN WORK;
```

```
SELECT  pg_class.oid  
FROM    pg_class JOIN pg_namespace ON (relnamespace = pg_namespace.oid)  
        JOIN pg_attribute ON (pg_class.oid = pg_attribute.attrelid)  
LIMIT 1;  
      oid  
-----  
    16591
```

複数テーブルの Access Share ロッキング

```
psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/191	relation	AccessShareLock	t		pg_attribute
1553323	3/191	relation	AccessShareLock	t		pg_attribute_relid_attnam_index
1553323	3/191	relation	AccessShareLock	t		pg_attribute_relid_attnum_index
1553323	3/191	relation	AccessShareLock	t		pg_class
1553323	3/191	relation	AccessShareLock	t		pg_class_oid_index
1553323	3/191	relation	AccessShareLock	t		pg_class_relname_nsp_index
1553323	3/191	relation	AccessShareLock	t		pg_class_tblspc_relfilenode_index
1553323	3/191	relation	AccessShareLock	t		pg_namespace
1553323	3/191	relation	AccessShareLock	t		pg_namespace_nspname_index
1553323	3/191	relation	AccessShareLock	t		pg_namespace_oid_index

```
COMMIT;
```

明示的な Row Share Share ロッキング

```
BEGIN WORK;
```

```
LOCK TABLE lockdemo IN ROW SHARE MODE;
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/192	relation	RowShareLock	t		lockdemo

```
COMMIT;
```

```
BEGIN WORK;
```

```
SELECT * FROM lockdemo FOR SHARE;
```

```
col
```

```
-----
```

```
1
```

```
SELECT txid_current();
```

```
txid_current
```

```
-----
```

```
27052734
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/193	transactionid	ExclusiveLock	t	27052734	
1553323	3/193	relation	RowShareLock	t		lockdemo

```
COMMIT;
```

```
BEGIN WORK;
```

```
LOCK TABLE lockdemo IN ROW EXCLUSIVE MODE;
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/194	relation	RowExclusiveLock	t		lockdemo

```
COMMIT;
```

```
BEGIN WORK;
```

```
DELETE FROM lockdemo;
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/195	transactionid	ExclusiveLock	t	27052735	
1553323	3/195	relation	RowExclusiveLock	t		lockdemo

```
ROLLBACK WORK;
```

明示的な Share Update Exclusive ロッキング

```
BEGIN WORK;
```

```
LOCK TABLE lockdemo IN SHARE UPDATE EXCLUSIVE MODE;
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/196	relation	ShareUpdateExclusiveLock	t		lockdemo

```
COMMIT;
```

```
BEGIN WORK;
```

```
ANALYZE lockdemo;
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/197	transactionid	ExclusiveLock	t	27052736	
1553323	3/197	relation	ShareUpdateExclusiveLock	t		lockdemo

```
ROLLBACK WORK;
```

```
BEGIN WORK;
```

```
LOCK TABLE lockdemo IN SHARE MODE;
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
\timing on
```

```
Timing is on.
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/198	relation	ShareLock	t		lockdemo

```
COMMIT;
```

```
BEGIN WORK;
```

```
CREATE UNIQUE INDEX i_lockdemo on lockdemo(col);
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
\timing on
```

```
Timing is on.
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/199	transactionid	ExclusiveLock	t	27052737	
1553323	3/199	relation	AccessExclusiveLock	t		
1553323	3/199	relation	ShareLock	t		lockdemo

```
COMMIT;
```

```
BEGIN WORK;
```

```
LOCK TABLE lockdemo IN SHARE ROW EXCLUSIVE MODE;
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/200	relation	ShareRowExclusiveLock	t		lockdemo

```
COMMIT;
```

```
BEGIN WORK;
```

```
CREATE TRIGGER lockdemo_trigger  
BEFORE UPDATE ON lockdemo  
FOR EACH ROW EXECUTE FUNCTION suppress_redundant_updates_trigger();
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/201	transactionid	ExclusiveLock	t	27052738	
1553323	3/201	relation	ShareRowExclusiveLock	t		lockdemo

```
ROLLBACK WORK;
```

明示的な Exclusive ロッキング

```
BEGIN WORK;
```

```
LOCK TABLE lockdemo IN EXCLUSIVE MODE;
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/202	relation	ExclusiveLock	t		lockdemo

```
COMMIT;
```

このロックモードは PostgreSQL の SQL コマンドで自動的に使われることはありません

```
BEGIN WORK;
```

```
LOCK TABLE lockdemo IN ACCESS EXCLUSIVE MODE;
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
\timing on
```

```
Timing is on.
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/203	transactionid	ExclusiveLock	t	27052739	
1553323	3/203	relation	AccessExclusiveLock	t		lockdemo

```
COMMIT;
```

Access Exclusive は LOCK コマンドのデフォルトのモードです

```
BEGIN WORK;
```

```
CLUSTER lockdemo USING i_lockdemo;
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/204	transactionid	ExclusiveLock	t	27052740	
1553323	3/204	object	AccessExclusiveLock	t		
1553323	3/204	object	AccessExclusiveLock	t		
1553323	3/204	relation	AccessExclusiveLock	t		i_lockdemo
1553323	3/204	relation	AccessExclusiveLock	t		lockdemo
1553323	3/204	relation	AccessExclusiveLock	t		
1553323	3/204	relation	AccessShareLock	t		i_lockdemo
1553323	3/204	relation	ShareLock	t		lockdemo

```
\! psql -e -c 'SELECT * FROM lockview2;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview2;
```

pid	vxid	lock_type	page	tuple	classid	objid	objsubid
1553323	3/204	transactionid					
1553323	3/204	object			1247	24805	0
1553323	3/204	object			1247	24806	0
1553323	3/204	relation					
1553323	3/204	relation					
1553323	3/204	relation					
1553323	3/204	relation					
1553323	3/204	relation					

```
COMMIT;
```

1247はpg_typeでのpg_classのエントリです。24805と24806は一時ファイル名として用いられています。

ロックの例



行ロックは pg_locks では見えません

```
DELETE FROM lockdemo;
```

```
BEGIN WORK;
```

```
INSERT INTO lockdemo VALUES (1);
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/206	transactionid	ExclusiveLock	t	27052742	
1553323	3/206	relation	RowExclusiveLock	t		lockdemo

追加した2つの行も見えません

```
INSERT INTO lockdemo VALUES (2), (3);
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/206	transactionid	ExclusiveLock	t	27052742	
1553323	3/206	relation	RowExclusiveLock	t		lockdemo

```
COMMIT;
```

UPDATE ではインデックスもロックされます

```
BEGIN WORK;
```

```
UPDATE lockdemo SET col = 1 WHERE col = 1;
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/207	transactionid	ExclusiveLock	t	27052743	
1553323	3/207	relation	RowExclusiveLock	t		i_lockdemo
1553323	3/207	relation	RowExclusiveLock	t		lockdemo

行を2つ更新しても同様です

```
UPDATE lockdemo SET col = 2 WHERE col = 2;
```

```
UPDATE lockdemo SET col = 3 WHERE col = 3;
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/207	transactionid	ExclusiveLock	t	27052743	
1553323	3/207	relation	RowExclusiveLock	t		i_lockdemo
1553323	3/207	relation	RowExclusiveLock	t		lockdemo

```
COMMIT;
```

行を1つ削除しても同様です

```
BEGIN WORK;
```

```
DELETE FROM lockdemo WHERE col = 1;
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/208	transactionid	ExclusiveLock	t	27052744	
1553323	3/208	relation	RowExclusiveLock	t		i_lockdemo
1553323	3/208	relation	RowExclusiveLock	t		lockdemo

行を2つ削除しても同様です

```
DELETE FROM lockdemo;
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/208	transactionid	ExclusiveLock	t	27052744	
1553323	3/208	relation	RowExclusiveLock	t		i_lockdemo
1553323	3/208	relation	RowExclusiveLock	t		lockdemo

```
ROLLBACK WORK;
```

明示的な行ロックも同様です

```
BEGIN WORK;
```

```
SELECT * FROM lockdemo WHERE col = 1 FOR UPDATE;
```

```
col
```

```
-----  
1
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/209	transactionid	ExclusiveLock	t	27052745	
1553323	3/209	relation	RowShareLock	t		i_lockdemo
1553323	3/209	relation	RowShareLock	t		lockdemo

もとのスライドとロックモードが異なりますが、これはPGのバージョンの違いと思われます

3行明示的にロックしても同様です

```
SELECT * FROM lockdemo FOR UPDATE;
```

```
col
```

```
-----
```

```
1
```

```
2
```

```
3
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/209	transactionid	ExclusiveLock	t	27052745	
1553323	3/209	relation	RowShareLock	t		i_lockdemo
1553323	3/209	relation	RowShareLock	t		lockdemo

```
COMMIT;
```

もとのスライドとロックモードが異なりますが、これはPGのバージョンの違いと思われます

明示的な Share Row Lock も同様です

```
BEGIN WORK;
```

```
SELECT * FROM lockdemo WHERE col = 1 FOR SHARE;
```

```
col
-----
1
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/210	transactionid	ExclusiveLock	t	27052746	
1553323	3/210	relation	RowShareLock	t		i_lockdemo
1553323	3/210	relation	RowShareLock	t		lockdemo

もとのスライドとロックモードが異なりますが、これはPGのバージョンの違いと思われます

3行明示的に Share Row Lock しても同様です

```
SELECT * FROM lockdemo FOR SHARE;
```

```
col
```

```
-----
```

```
1
```

```
2
```

```
3
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/210	transactionid	ExclusiveLock	t	27052746	
1553323	3/210	relation	RowShareLock	t		i_lockdemo
1553323	3/210	relation	RowShareLock	t		lockdemo

```
COMMIT;
```

もとのスライドとロックモードが異なりますが、これはPGのバージョンの違いと思われる

ここで lockdemo を元に戻します

```
DELETE FROM lockdemo;
```

```
INSERT INTO lockdemo VALUES (1);
```

UPDATE が SELECT でブロックされることはありません

```
BEGIN WORK;
```

```
SELECT ctid, xmin, * FROM lockdemo;
```

ctid	xmin	col
(0,8)	27052748	1

UPDATE が SELECT でブロックされることはありません

```
SELECT pg_backend_pid();  
pg_backend_pid  
-----  
1553323
```

```
SELECT txid_current();  
txid_current  
-----  
27052749
```

UPDATE が SELECT でブロックされることはありません

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
SELECT * FROM lockview1;
   pid   | vxid | lock_type | lock_mode | granted | xid_lock | relname
-----+-----+-----+-----+-----+-----+-----
 1553323 | 3/213 | transactionid | ExclusiveLock | t       | 27052749 |
 1553323 | 3/213 | relation    | AccessShareLock | t       |          | i_lockdemo
 1553323 | 3/213 | relation    | AccessShareLock | t       |          | lockdemo

\! psql -e -c 'UPDATE lockdemo SET col = 2; \
SELECT pg_sleep(0.200); \
SELECT ctid, xmin, * FROM lockdemo;' | \
sed 's/^/\t/g' &
```

バックスラッシュを使ってコマンド行の継続を示していますが、psql ではこのようなバックスラッシュの使い方はできません。コマンドは実際には1行で入力しなければなりません。

UPDATE が SELECT でブロックされることはありません

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

SELECT * FROM lockview1;						
pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/213	transactionid	ExclusiveLock	t	27052749	
1553323	3/213	relation	AccessShareLock	t		i_lockdemo
1553323	3/213	relation	AccessShareLock	t		lockdemo
1794256	4/7185	transactionid	ExclusiveLock	t	27052750	
1794256	4/7185	relation	AccessShareLock	t		pg_proc
1794256	4/7185	relation	AccessShareLock	t		pg_proc_proname_args_nsp_index
1794256	4/7185	relation	RowExclusiveLock	t		i_lockdemo
1794256	4/7185	relation	RowExclusiveLock	t		lockdemo

ctid	xmin	col
(0,9)	27052750	2

```
COMMIT WORK;
```

ここで lockdemo を元に戻します

```
DELETE FROM lockdemo;
```

```
INSERT INTO lockdemo VALUES (1);
```

並列して行う Update でのロッキング

```
BEGIN WORK;
```

```
SELECT ctid, xmin, * FROM lockdemo;
```

ctid	xmin	col
(0,10)	27052752	1

```
UPDATE lockdemo SET col = 2;
```

```
SELECT ctid, xmin, * FROM lockdemo;
```

ctid	xmin	col
(0,11)	27052753	2

```
SELECT pg_backend_pid();
```

pg_backend_pid
1553323

```
SELECT txid_current();
```

txid_current
27052753

並列して行う Update でのロッキング

```
\! psql -e -c 'BEGIN WORK; UPDATE lockdemo SET col = 3; \
SELECT pg_sleep(0.200); \
COMMIT;' | \
sed 's/^\t/g' &
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^\t/g'
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/216	transactionid	ExclusiveLock	t	27052753	
1553323	3/216	relation	AccessShareLock	t		i_lockdemo
1553323	3/216	relation	AccessShareLock	t		lockdemo
1553323	3/216	relation	RowExclusiveLock	t		i_lockdemo
1553323	3/216	relation	RowExclusiveLock	t		lockdemo
1801662	4/7389	transactionid	ExclusiveLock	t	27052754	
1801662	4/7389	relation	RowExclusiveLock	t		i_lockdemo
1801662	4/7389	relation	RowExclusiveLock	t		lockdemo
1801662	4/7389	tuple	AccessExclusiveLock	t		lockdemo
1801662	4/7389	transactionid	ShareLock	f	27052753	

並列して行う Update でのロッキング

```
\! psql -e -c 'SELECT * FROM lockview2;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview2;
```

pid	vxid	lock_type	page	tuple	classid	objid	objsubid
1553323	3/216	transactionid					
1553323	3/216	relation					
1553323	3/216	relation					
1553323	3/216	relation					
1553323	3/216	relation					
1801662	4/7389	transactionid					
1801662	4/7389	relation					
1801662	4/7389	relation					
1801662	4/7389	tuple	0	10			
1801662	4/7389	transactionid					

```
COMMIT;
```

3つの並列 Update でのロッキング

```
CREATE VIEW lockinfo_hierarchy AS
  WITH RECURSIVE lockinfo1 AS (
    SELECT pid, vxid, granted, xid_lock, lock_type, relname, page, tuple
    FROM lockview
    WHERE xid_lock IS NOT NULL AND
          relname IS NULL AND
          granted
    UNION ALL
    SELECT lockview.pid, lockview.vxid, lockview.granted, lockview.xid_lock,
           lockview.lock_type, lockview.relname, lockview.page,
lockview.tuple
    FROM lockinfo1 JOIN lockview ON (lockinfo1.xid_lock = lockview.xid_lock)
    WHERE lockview.xid_lock IS NOT NULL AND
          lockview.relname IS NULL AND
          NOT lockview.granted AND
          lockinfo1.granted),
```

3つの並列 Update でのロッキング

```
lockinfo2 AS (  
    SELECT pid, vxid, granted, xid_lock, lock_type, relname, page, tuple  
    FROM lockview  
    WHERE lock_type = 'tuple' AND  
           granted  
    UNION ALL  
    SELECT lockview.pid, lockview.vxid, lockview.granted, lockview.xid_lock,  
           lockview.lock_type, lockview.relname, lockview.page, lockview.tuple  
    FROM lockinfo2 JOIN lockview ON (  
        lockinfo2.lock_type = lockview.lock_type AND  
        lockinfo2.relname = lockview.relname AND  
        lockinfo2.page = lockview.page AND  
        lockinfo2.tuple = lockview.tuple)  
    WHERE lockview.lock_type = 'tuple' AND  
           NOT lockview.granted AND  
           lockinfo2.granted  
)  
SELECT * FROM lockinfo1  
UNION ALL  
SELECT * FROM lockinfo2;
```

3つの並列 Update でのロッキング

```
BEGIN WORK;
```

```
SELECT ctid, xmin, * FROM lockdemo;
```

ctid	xmin	col
(0,12)	27052754	3

```
UPDATE lockdemo SET col = 4;
```

```
SELECT ctid, xmin, * FROM lockdemo;
```

ctid	xmin	col
(0,13)	27052756	4

```
SELECT pg_backend_pid();
```

pg_backend_pid
1553323

```
SELECT txid_current();
```

txid_current
27052756

3つの並列 Update でのロッキング

```
\! psql -e -c 'BEGIN WORK; UPDATE lockdemo SET col = 5; \  
SELECT pg_sleep(0.200); \  
COMMIT;' | \  
sed 's/^/\t/g' &
```

```
\! psql -e -c 'BEGIN WORK; UPDATE lockdemo SET col = 6; \  
SELECT pg_sleep(0.200); \  
COMMIT;' | \  
sed 's/^/\t/g' &
```

```
\! psql -e -c 'BEGIN WORK; UPDATE lockdemo SET col = 7; \  
SELECT pg_sleep(0.200); \  
COMMIT;' | \  
sed 's/^/\t/g' &
```

3つの並列 Update でのロッキング

```
koichi=# \! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/218	transactionid	ExclusiveLock	t	27052756	
1553323	3/218	relation	AccessShareLock	t		i_lockdemo
1553323	3/218	relation	AccessShareLock	t		lockdemo
1553323	3/218	relation	RowExclusiveLock	t		i_lockdemo
1553323	3/218	relation	RowExclusiveLock	t		lockdemo
1807114	4/7523	transactionid	ExclusiveLock	t	27052757	
1807114	4/7523	relation	RowExclusiveLock	t		i_lockdemo
1807114	4/7523	relation	RowExclusiveLock	t		lockdemo
1807114	4/7523	tuple	AccessExclusiveLock	t		lockdemo
1807114	4/7523	transactionid	ShareLock	f	27052756	
1807164	5/10	transactionid	ExclusiveLock	t	27052758	
1807164	5/10	relation	RowExclusiveLock	t		i_lockdemo
1807164	5/10	relation	RowExclusiveLock	t		lockdemo
1807164	5/10	tuple	AccessExclusiveLock	f		lockdemo
1807209	6/4	transactionid	ExclusiveLock	t	27052759	
1807209	6/4	relation	RowExclusiveLock	t		i_lockdemo
1807209	6/4	relation	RowExclusiveLock	t		lockdemo
1807209	6/4	tuple	AccessExclusiveLock	f		lockdemo

3つの並列 Update でのロッキング

```
\! psql -e -c 'SELECT * FROM lockview2;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview2;
```

pid	vxid	lock_type	page	tuple	classid	objid	objsubid
1553323	3/218	transactionid					
1553323	3/218	relation					
1553323	3/218	relation					
1553323	3/218	relation					
1553323	3/218	relation					
1807114	4/7523	transactionid					
1807114	4/7523	relation					
1807114	4/7523	relation					
1807114	4/7523	tuple	0	12			
1807114	4/7523	transactionid					
1807164	5/10	transactionid					
1807164	5/10	relation					
1807164	5/10	relation					
1807164	5/10	tuple	0	12			
1807209	6/4	transactionid					
1807209	6/4	relation					
1807209	6/4	relation					
1807209	6/4	tuple	0	12			

3つの並列 Update でのロッキング

```
\! psql -e -c 'SELECT * FROM lockinfo_hierarchy;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockinfo_hierarchy;
```

pid	vxid	granted	xid_lock	lock_type	relname	page	tuple
1553323	3/218	t	27052756	transactionid			
1807114	4/7523	t	27052757	transactionid			
1807164	5/10	t	27052758	transactionid			
1807209	6/4	t	27052759	transactionid			
1807114	4/7523	f	27052756	transactionid			
1807114	4/7523	t		tuple	lockdemo	0	12
1807164	5/10	f		tuple	lockdemo	0	12
1807209	6/4	f		tuple	lockdemo	0	12

3つの並列 Update でのロッキング

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/218	transactionid	ExclusiveLock	t	27052756	
1553323	3/218	relation	AccessShareLock	t		i_lockdemo
1553323	3/218	relation	AccessShareLock	t		lockdemo
1553323	3/218	relation	RowExclusiveLock	t		i_lockdemo
1553323	3/218	relation	RowExclusiveLock	t		lockdemo
1807114	4/7523	transactionid	ExclusiveLock	t	27052757	
1807114	4/7523	relation	RowExclusiveLock	t		i_lockdemo
1807114	4/7523	relation	RowExclusiveLock	t		lockdemo
1807114	4/7523	tuple	AccessExclusiveLock	t		lockdemo
1807114	4/7523	transactionid	ShareLock	f	27052756	
1807164	5/10	transactionid	ExclusiveLock	t	27052758	
1807164	5/10	relation	RowExclusiveLock	t		i_lockdemo
1807164	5/10	relation	RowExclusiveLock	t		lockdemo
1807164	5/10	tuple	AccessExclusiveLock	f		lockdemo
1807209	6/4	transactionid	ExclusiveLock	t	27052759	
1807209	6/4	relation	RowExclusiveLock	t		i_lockdemo
1807209	6/4	relation	RowExclusiveLock	t		lockdemo
1807209	6/4	tuple	AccessExclusiveLock	f		lockdemo

```
koichi=*# COMMIT;
```

```
DELETE FROM lockdemo;
```

```
INSERT INTO lockdemo VALUES (50), (80);
```

```
BEGIN WORK;
```

```
UPDATE lockdemo SET col = 50 WHERE col = 50;
```

```
SELECT pg_backend_pid();
pg_backend_pid
-----
1553323
```

```
SELECT txid_current();
txid_current
-----
27052762
```

```
\! psql -e -c 'BEGIN WORK; \  
UPDATE lockdemo SET col = 81 WHERE col = 80; \  
UPDATE lockdemo SET col = 51 WHERE col = 50; \  
COMMIT;' | \  
sed 's/^/\t/g' &
```

```
koichi=*# \! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/221	transactionid	ExclusiveLock	t	27052762	
1553323	3/221	relation	RowExclusiveLock	t		i_lockdemo
1553323	3/221	relation	RowExclusiveLock	t		lockdemo
1829314	4/8071	transactionid	ExclusiveLock	t	27052763	
1829314	4/8071	relation	RowExclusiveLock	t		i_lockdemo
1829314	4/8071	relation	RowExclusiveLock	t		lockdemo
1829314	4/8071	tuple	AccessExclusiveLock	t		lockdemo
1829314	4/8071	transactionid	ShareLock	f	27052762	

```
\! psql -e -c 'SELECT * FROM lockview2;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview2;
```

pid	vxid	lock_type	page	tuple	classid	objid	objsubid
1553323	3/221	transactionid					
1553323	3/221	relation					
1553323	3/221	relation					
1829314	4/8071	transactionid					
1829314	4/8071	relation					
1829314	4/8071	relation					
1829314	4/8071	tuple	0	17			
1829314	4/8071	transactionid					

-- deadlock_timeout の待ち合わせの間の lockview を表示する

```
\! psql -e -c 'SELECT pg_sleep(0.200); \  
SELECT * FROM lockview1;' | \  
sed 's/^\t/g' &  
\! psql -e -c 'SELECT pg_sleep(0.400); \  
SELECT * FROM lockview2;' | \  
sed 's/^\t/g' &
```

-- 次の行は deadlock timeout の待ち合わせでハングする

```
UPDATE lockdemo SET col = 80 WHERE col = 80;
```

```
SELECT pg_sleep(0.200); SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/221	transactionid	ExclusiveLock	t	27052762	
1553323	3/221	relation	RowExclusiveLock	t		i_lockdemo
1553323	3/221	relation	RowExclusiveLock	t		lockdemo
1553323	3/221	tuple	ExclusiveLock	t		lockdemo
1553323	3/221	transactionid	ShareLock	f	27052763	
1829314	4/8071	transactionid	ExclusiveLock	t	27052763	
1829314	4/8071	relation	RowExclusiveLock	t		i_lockdemo
1829314	4/8071	relation	RowExclusiveLock	t		lockdemo
1829314	4/8071	tuple	AccessExclusiveLock	t		lockdemo
1829314	4/8071	transactionid	ShareLock	f	27052762	

```
SELECT pg_sleep(0.400); SELECT * FROM lockview2;
```

pid	vxid	lock_type	page	tuple	classid	objid	objsubid
1553323	3/221	transactionid					
1553323	3/221	relation					
1553323	3/221	relation					
1553323	3/221	tuple	0	18			
1553323	3/221	transactionid					
1829314	4/8071	transactionid					
1829314	4/8071	relation					
1829314	4/8071	relation					
1829314	4/8071	tuple	0	17			
1829314	4/8071	transactionid					

ERROR: deadlock detected

DETAIL: Process 1829314 waits for ShareLock on transaction 27052762; blocked by process 1553323.

Process 1553323 waits for ShareLock on transaction 27052763; blocked by process 1829314.

HINT: See server log for query details.

commit;

```
DELETE FROM lockdemo;
```

```
INSERT INTO lockdemo VALUES (40), (60), (80);
```

```
BEGIN WORK;
```

```
UPDATE lockdemo SET col = 40 WHERE col = 40;
```

```
SELECT pg_backend_pid();  
pg_backend_pid
```

```
-----  
1553323
```

```
SELECT txid_current();  
txid_current
```

```
-----  
27052771
```

```
\! psql -e -c 'BEGIN WORK; \  
    UPDATE lockdemo SET col = 61 WHERE col = 60; \  
    UPDATE lockdemo SET col = 41 WHERE col = 40; \  
    COMMIT;' | \  
    sed 's/^/\t/g' &  
\! psql -e -c 'BEGIN WORK; \  
    UPDATE lockdemo SET col = 81 WHERE col = 80; \  
    UPDATE lockdemo SET col = 61 WHERE col = 60; \  
    COMMIT;' | \  
    sed 's/^/\t/g' &
```

```
\! psql -e -c 'SELECT pg_sleep(0.200); \
SELECT * FROM lockview1;' | \
sed 's/^/\t/g' &
\! psql -e -c 'SELECT pg_sleep(0.400); \
SELECT * FROM lockview2;' | \
sed 's/^/\t/g' &
```

-- 次の行では deadlock timeout を待ち合わせてハングする
UPDATE lockdemo SET col = 80 WHERE col = 80;

3つのトランザクションによるデッドロック

```
SELECT pg_sleep(0.200); SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1553323	3/228	transactionid	ExclusiveLock	t	27052771	
1553323	3/228	relation	RowExclusiveLock	t		i_lockdemo
1553323	3/228	relation	RowExclusiveLock	t		lockdemo
1835865	4/8239	transactionid	ExclusiveLock	t	27052772	
1835865	4/8239	relation	RowExclusiveLock	t		i_lockdemo
1835865	4/8239	relation	RowExclusiveLock	t		lockdemo
1835865	4/8239	tuple	AccessExclusiveLock	t		lockdemo
1835865	4/8239	transactionid	ShareLock	f	27052771	
1835870	5/20	transactionid	ExclusiveLock	t	27052773	
1835870	5/20	relation	RowExclusiveLock	t		i_lockdemo
1835870	5/20	relation	RowExclusiveLock	t		lockdemo
1835870	5/20	tuple	AccessExclusiveLock	t		lockdemo
1835870	5/20	transactionid	ShareLock	f	27052772	

3つのトランザクションによるデッドロック

```
SELECT pg_sleep(0.400); SELECT * FROM lockview2;
```

pid	vxid	lock_type	page	tuple	classid	objid	objsubid
1553323	3/228	transactionid					
1553323	3/228	relation					
1553323	3/228	relation					
1835865	4/8239	transactionid					
1835865	4/8239	relation					
1835865	4/8239	relation					
1835865	4/8239	tuple	0	30			
1835865	4/8239	transactionid					
1835870	5/20	transactionid					
1835870	5/20	relation					
1835870	5/20	relation					
1835870	5/20	tuple	0	31			
1835870	5/20	transactionid					

3つのトランザクションによるデッドロック

ERROR: deadlock detected

DETAIL: Process 1553323 waits for ShareLock on transaction 27052773; blocked by process 1835870.

Process 1835870 waits for ShareLock on transaction 27052772; blocked by process 1835865.

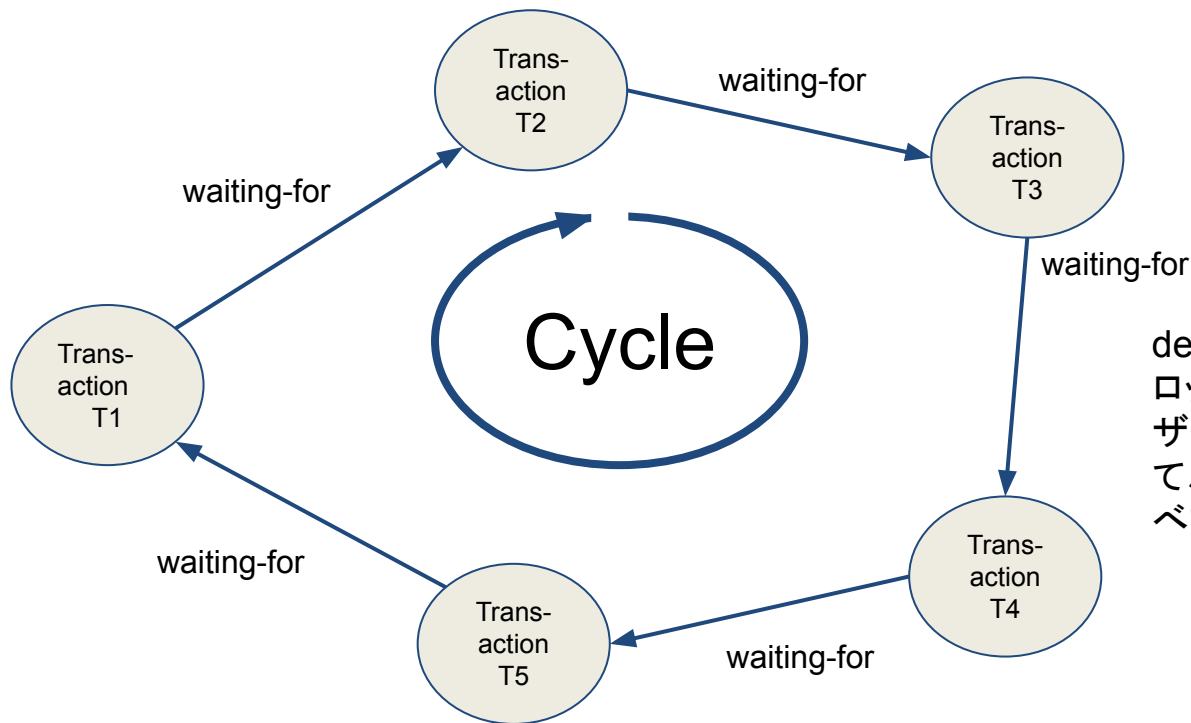
Process 1835865 waits for ShareLock on transaction 27052771; blocked by process 1553323.

HINT: See server log for query details.

commit;

実際にはロック待ちのサイクルを探します

デッドロックがある場合は、トランザクション待ちがサイクルになります*



deadlock_timeout の時間以内に
ロックが取得できない場合、トラン
ザクション間のロック待ちを解析し
て、サイクルになるかどうかを調
べます

* Jim Gray et al., "Transaction Processing", Morgan Kaufmann, 1993, pp.425,
Hector Garcia-Molina et al., "Database Systems, The Complete Book", Prentice Hall, 2002, pp.1010

ここで lockdemo を元に戻します

```
DELETE FROM lockdemo;
```

```
INSERT INTO lockdemo VALUES (1);
```

```
BEGIN WORK;
```

```
SELECT * FROM lockdemo;
```

```
col
```

```
-----
```

```
1
```

```
SELECT pg_backend_pid();  
pg_backend_pid
```

```
-----  
1884726
```

```
SELECT txid_current();  
txid_current
```

```
-----  
27052777
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1884726	3/865	transactionid	ExclusiveLock	t	27052777	
1884726	3/865	relation	AccessShareLock	t		i_lockdemo
1884726	3/865	relation	AccessShareLock	t		lockdemo

```
COMMIT;
```

```
BEGIN WORK;
```

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

```
SELECT * FROM lockdemo;
```

```
col
```

```
-----
```

```
1
```

```
SELECT pg_backend_pid();  
pg_backend_pid
```

1884726

```
SELECT txid_current();  
txid_current
```

27052778

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1884726	3/866	transactionid	ExclusiveLock	t	27052778	
1884726	3/866	relation	AccessShareLock	t		i_lockdemo
1884726	3/866	relation	AccessShareLock	t		lockdemo
1884726	3/866	relation	SIReadLock	t		lockdemo

```
COMMIT;
```

Predicate Lock とよばれる手法:

- どのトランザクションがどのタプルを読んだかを記録する
- 書き込みを行う時は、この書き込みに矛盾する読み込みを行っているかどうかをチェックする
- 読み込みを行うときは、この読み込みと矛盾する書き込みを行っているかどうかをチェックする

これは、MVCC とは独立した実装になっている。(src/backend/storage/lmgr/predicate.c)

- ロックの単位は、行・ページ・テーブル
- ロック数が多くなると、行 → ページ → テーブルとロック単位をエスカレーションする

開発者によれば、上記のオーバーヘッドは 5% 位とのこと (PGCon での話)。

```
koichi=# \d lockdemo
```

```
        Table "public.lockdemo"
```

Column	Type	Collation	Nullable	Default
col	integer			

```
Indexes:
```

```
    "i_lockdemo" UNIQUE, btree (col) CLUSTER
```

Unique 値の追加でのロッキング

```
BEGIN WORK;
```

```
INSERT INTO lockdemo VALUES (2);
```

```
SELECT pg_backend_pid();  
pg_backend_pid
```

```
-----
```

```
1884726
```

```
SELECT txid_current();  
txid_current
```

```
-----
```

```
27052779
```

Unique 値の追加でのロッキング

```
\! PGOPTIONS='-c statement_timeout=200' \
psql -e -c 'INSERT INTO lockdemo VALUES (2);' | \
sed 's/^\t/g' &
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^\t/g'
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1884726	3/876	transactionid	ExclusiveLock	t	27052779	lockdemo
1884726	3/876	relation	RowExclusiveLock	t		
1888807	5/22	transactionid	ExclusiveLock	t	27052780	i_lockdemo
1888807	5/22	relation	RowExclusiveLock	t		
1888807	5/22	relation	RowExclusiveLock	t		lockdemo
1888807	5/22	transactionid	ShareLock	f	27052779	

```
SELECT pg_sleep(0.400);
ERROR: canceling statement due to statement timeout
```

```
ROLLBACK WORK;
```

```
BEGIN WORK;
```

```
UPDATE lockdemo SET col = 1;
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1884726	3/877	transactionid	ExclusiveLock	t	27052781	
1884726	3/877	relation	RowExclusiveLock	t		i_lockdemo
1884726	3/877	relation	RowExclusiveLock	t		lockdemo

```
SAVEPOINT lockdemo1;
```

```
UPDATE lockdemo SET col = 2;
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1884726	3/877	transactionid	ExclusiveLock	t	27052781	
1884726	3/877	transactionid	ExclusiveLock	t	27052782	
1884726	3/877	relation	RowExclusiveLock	t		i_lockdemo
1884726	3/877	relation	RowExclusiveLock	t		lockdemo

```
ROLLBACK WORK TO SAVEPOINT lockdemo1;
```

```
UPDATE lockdemo SET col = 3;
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
\timing on
```

```
Timing is on.
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
1884726	3/877	transactionid	ExclusiveLock	t	27052781	
1884726	3/877	transactionid	ExclusiveLock	t	27052783	
1884726	3/877	relation	RowExclusiveLock	t		i_lockdemo
1884726	3/877	relation	RowExclusiveLock	t		lockdemo

```
COMMIT;
```

```
koichi=# BEGIN
```

```
SELECT pg_advisory_lock(col) FROM lockdemo;
pg_advisory_lock
```

```
-----
```

```
\! psql -e -c 'SELECT * FROM lockview1;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview1;
```

pid	vxid	lock_type	lock_mode	granted	xid_lock	relname
-----+	-----+	-----+	-----+	-----+	-----+	-----+
1884726	3/878	advisory	ExclusiveLock	t		
1884726	3/878	relation	AccessShareLock	t		i_lockdemo
1884726	3/878	relation	AccessShareLock	t		lockdemo

```
\! psql -e -c 'SELECT * FROM lockview2;' | sed 's/^/\t/g'
```

```
SELECT * FROM lockview2;
```

pid	vxid	lock_type	page	tuple	classid	objid	objsubid
-----	-----	-----	-----	-----	-----	-----	-----
1884726	3/878	advisory			0	3	1
1884726	3/878	relation					
1884726	3/878	relation					

```
SELECT pg_advisory_unlock(col) FROM lockdemo;
```

```
pg_advisory_unlock
```

```
-----  
t
```

```
COMMIT;
```

pg_locks とpg_stat_activity をjoinする

-- 他のセッションでも使うので、一時ビューにはできない

```
CREATE VIEW lock_stat_view AS
SELECT  pg_stat_activity.pid AS pid,
        query, wait_event, vxid, lock_type,
        lock_mode, granted, xid_lock
FROM    lockview JOIN pg_stat_activity ON (lockview.pid = pg_stat_activity.pid);
```

pg_locks と pg_stat_activity を join する

```
BEGIN WORK;
```

```
UPDATE lockdemo SET col = 1;
```

```
SELECT pg_backend_pid();  
pg_backend_pid  
-----  
1884726
```

```
SELECT txid_current();  
txid_current  
-----  
27052789
```

pg_locks と pg_stat_activity を join する

```
\! psql -e -c 'UPDATE lockdemo SET col = 2;' | sed 's/^/\t/g' &
\! psql -e -c 'UPDATE lockdemo SET col = 3;' | sed 's/^/\t/g' &
\! psql -e -c 'SELECT * FROM lock_stat_view;' | sed 's/^/\t/g'
```

pid	query	wait_event	vxid	lock_type	lock_mode	granted	xid_lock
1884726	SELECT txid_current();	ClientRead	3/881	transactionid	ExclusiveLock	t	27052789
1884726	SELECT txid_current();	ClientRead	3/881	relation	RowExclusiveLock	t	
1884726	SELECT txid_current();	ClientRead	3/881	relation	RowExclusiveLock	t	
1893933	UPDATE lockdemo SET col = 2;	tuple	4/9211	transactionid	ExclusiveLock	t	27052791
1893933	UPDATE lockdemo SET col = 2;	tuple	4/9211	relation	RowExclusiveLock	t	
1893933	UPDATE lockdemo SET col = 2;	tuple	4/9211	relation	RowExclusiveLock	t	
1893933	UPDATE lockdemo SET col = 2;	tuple	4/9211	tuple	AccessExclusiveLock	f	
1893934	UPDATE lockdemo SET col = 3;	transactionid	5/26	transactionid	ExclusiveLock	t	27052790
1893934	UPDATE lockdemo SET col = 3;	transactionid	5/26	relation	RowExclusiveLock	t	
1893934	UPDATE lockdemo SET col = 3;	transactionid	5/26	relation	RowExclusiveLock	t	
1893934	UPDATE lockdemo SET col = 3;	transactionid	5/26	tuple	ExclusiveLock	t	
1893934	UPDATE lockdemo SET col = 3;	transactionid	5/26	transactionid	ShareLock	f	27052789

```
SELECT pg_blocking_pids(1893933);
pg_blocking_pids
```

```
{1893934}
```

```
SELECT pg_blocking_pids(1893934);
pg_blocking_pids
```

```
{1884726}
```

```
COMMIT;
```

ご清聴ありがとうございます

Koichi Suzuki

koichi.suzuki@enterprisedb.com