

トランザクション、ロック

1日目

2日目

3日目

4日目

5日目

6日目

7日目

8日目

9日目

10日目

11日目

12日目

13日目

14日目

15日目

16日目

17日目

18日目

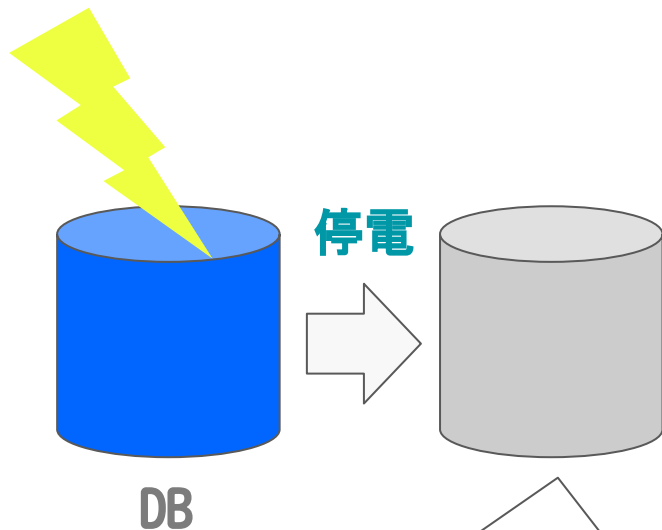
19日目

20日目

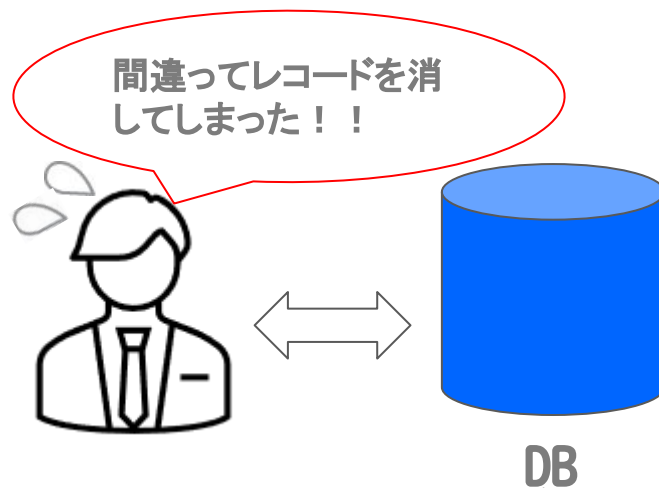
21日目

正確にデータを管理するには

DBは、データを正確に管理する必要がある



停電して、DBが停止しても正確な値でなければならない
(一部のテーブルが更新されていて、一部がされてい
はいけない)



作業中、誤ってレコードを削除しても、元に戻せるように
しなければならない

トランザクションとは

トランザクションとは

- 複数のSQLをひとかたまりとして、扱うようにDBに指示する
- 実行に成功したら、結果を反映し(コミット)、失敗したら元に戻す(ロールバック)



DBの原子性を担保する

原子性について

トランザクションに含まれる複数のSQLが不可分のものであり、必ず「**すべてが実行されている**」か、「**1つも実行されていない**」状態に制御しなければいけないとするDBの性質



A子から**B子**に**1万円**を振込んだ

この2つの処理は、「**すべてが実行されている**」か、「**1つも実行されていない**」状態にしなければいけない= **原子性**

トランザクション処理の書き方

START TRANSACTION

トランザクションを開始するSQLコマンド(他のDBだとSTARTでなくBEGINの場合もある)

トランザクションの開始

START TRANSACTION;

トランザクション内の処理(複数のSQLはひとまとめにして実行させる)

INSERT ...

DELETE ...

COMMIT

トランザクションを完了して、実行結果をDBに反映させる(コミットする)SQL
コマンド

トランザクションの開始

START TRANSACTION;

トランザクション内の処理(複数のSQLはひとまとめにして実行させる)

INSERT ...

DELETE ...

トランザクションが終了(実行結果が反映される)

COMMIT;

ROLLBACK

トランザクションを破棄して、DBをトランザクション前の状態に戻す(ロールバックする)SQLコマンド

トランザクションの開始

START TRANSACTION;

トランザクション内の処理(複数のSQLはひとまとめにして実行させる)

INSERT ...

DELETE ...

トランザクションを破棄(明示的にROLLBACKを実行せずに、DBとの接続が切断された場合もロールバックされる)

ROLLBACK;

自動コミットのON/OFF

自動コミットモードについて

多くのDBMSでは、明示的にトランザクションを開始しない限り、SQLを実行すると自動的にコミットされる。これを自動コミットモードという

自動コミットモードかどうか確認する

```
SHOW VARIABLES WHERE variable_name="autocommit";
```

自動コミットモードを解除する

```
SET AUTOCOMMIT=0
```

トランザクションとロック

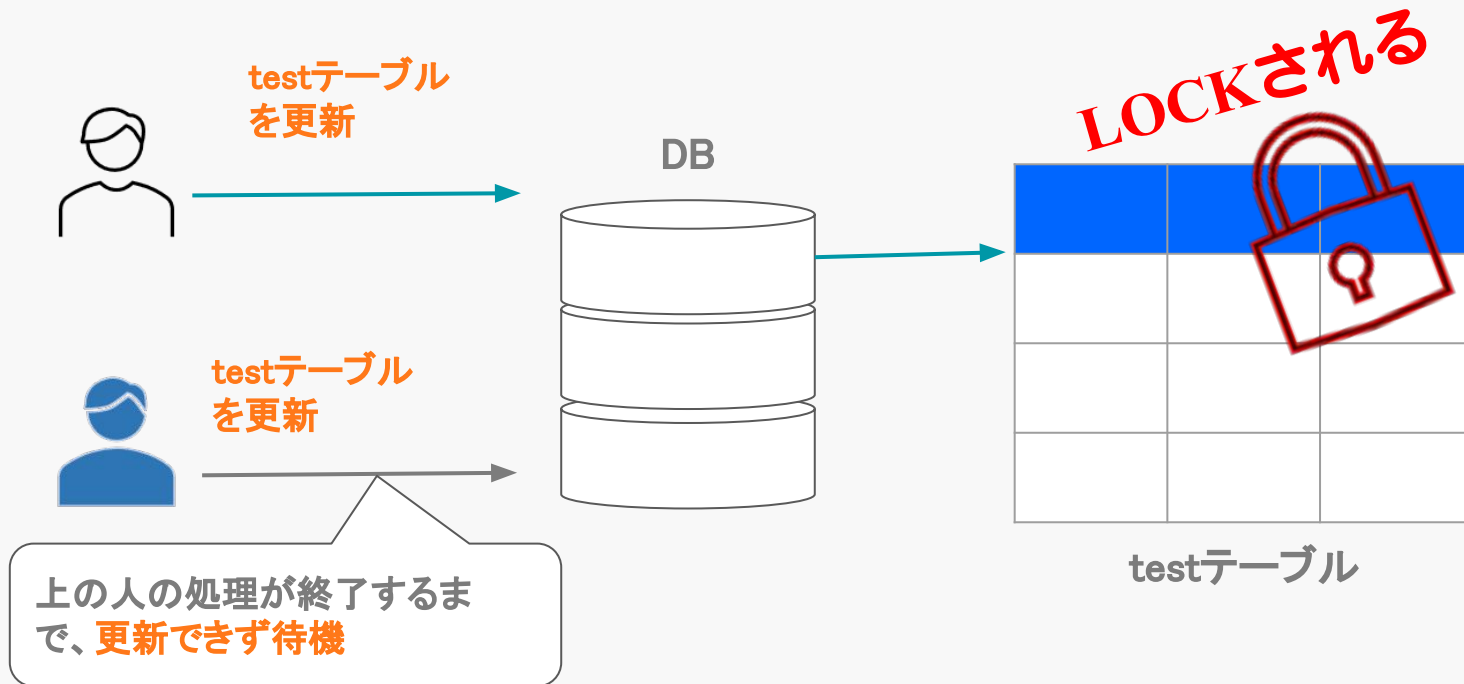


作成したサービスを通してDBを利用する
人は、**世界中にたくさんいる**

- ・テーブルが、**全く同じタイミング**で複数の場所から更新されたら？
- ・テーブルからレコードを取得している**最中に**、レコードが削除されたら？

ロックとは

DBのテーブルや行を固定して、別のトランザクションからテーブルや行の参照・更新ができないようにすること



共有ロック、排他(占有)ロックとは？

ロックには、大きく2つの方法がある

共有ロック・・・他のトランザクション(ユーザー)から、参照はできるが更新はできなくなるロック

排他ロック・・・他のトランザクション(ユーザー)から、参照も更新もできなくなるロック

共有ロック



排他ロック



複数のトランザクションがロックをかけた場合

複数のトランザクションが共有ロック、排他ロックをかけようとした場合の処理は以下の表のようになる

	共有ロック	排他ロック
共有ロック	○	×
排他ロック	×	×

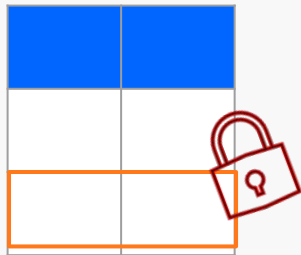
○: 共存可能
×: 共存不可能

- 共有ロックは、**複数のトランザクション**からかけられる
- 排他ロックをかけると、**別のトランザクション**から**共有ロック、排他ロック**をかけられない(排他ロックは1つのトランザクションからしかかけることができない)

行ロックとテーブルロック

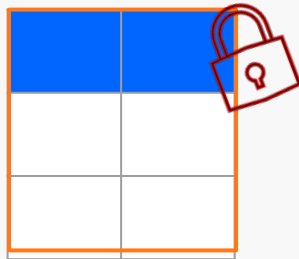
ロックする対象には、主に**行とテーブル**がある(DB自体にロックすることもできるが、ここでは取り上げない)

行ロック



この行だけロック
(他から更新できない)

テーブルロック



テーブル全体をロック
(他から更新できない)

トランザクションの開始

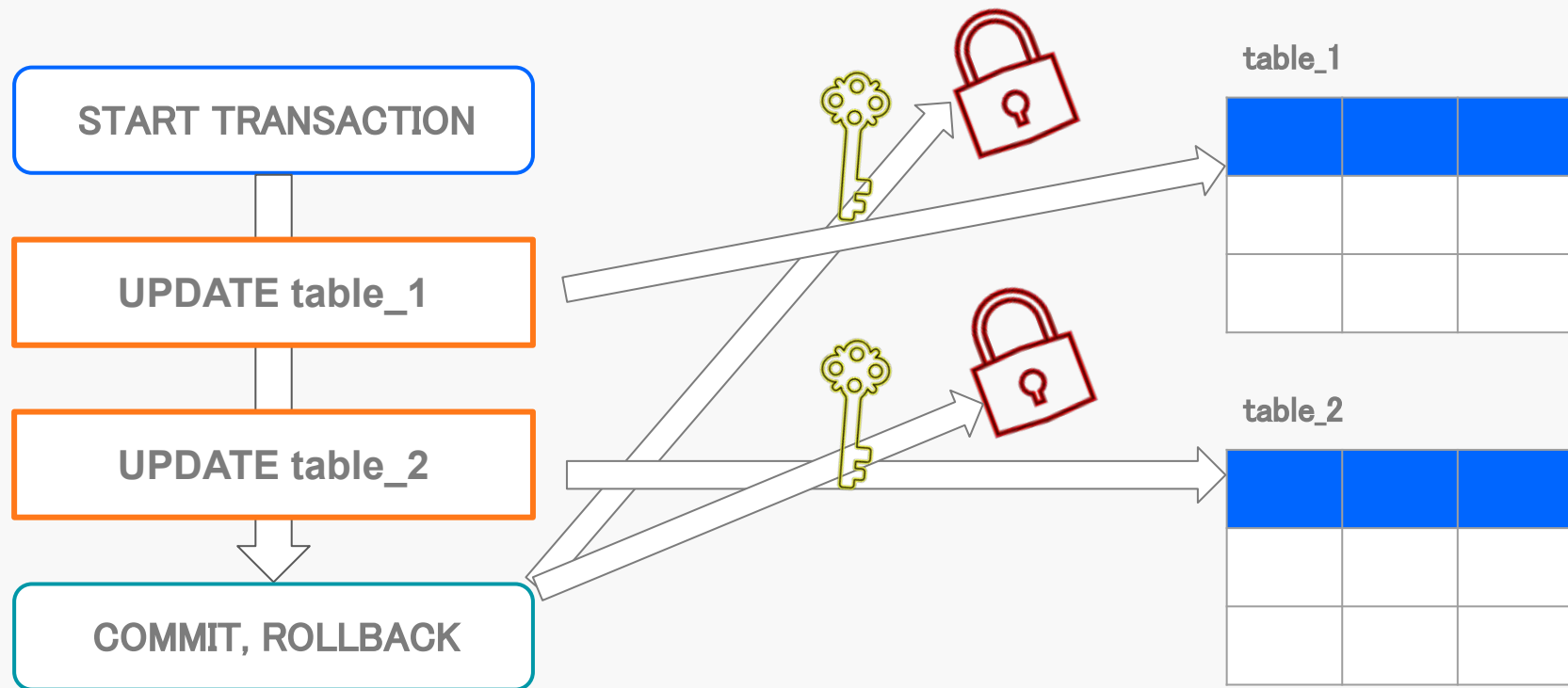
START TRANSACTION;

UPDATE

UPDATE table SET name="〇〇" WHERE id=12; — idが12の行だけロックがかかる(行ロック)

トランザクションとロック

トランザクションを開始して、テーブルの更新・削除等、ロックに関連する処理を実行するとトランザクションが終了するまでテーブル・レコードがロックされる



SQLとロック①～SELECT～

SELECT処理を実行した場合、以下のように実行した場合には、テーブルがロックされる*)

*) MySQL(ver 8)の場合のロック方法を記載します

テーブルロック

共有ロック(参照できるが、更新できない)

```
SELECT * FROM table_name FOR SHARE
```

排他ロック(参照も更新もできない)

```
SELECT * FROM table_name FOR UPDATE (NOWAIT)
```

行ロック(主キーまたはユニークカラムで絞り込んだ場合)

共有ロック(参照できるが、更新できない)

```
SELECT * FROM table_name WHERE id=1 FOR SHARE
```

排他ロック(参照も更新もできない)

```
SELECT * FROM table_name WHERE id=1 FOR UPDATE
```

SQLとロック②～UPDATE, DELETE, INSERT～

UPDATE, DELETE, INSERT処理を実行した場合、自動的にテーブル・レコードが
排他ロックされる

テーブルロック

排他ロック

```
UPDATE table_name SET name="AA"
```

排他ロック

```
DELETE FROM table_name WHERE name="AA"
```

行ロック(主キーまたはユニークカラムで絞り込んだ場合)

排他ロック

```
UPDATE table_name SET name="AA"
```

```
DELETE FROM table_name WHERE id=1
```

```
INSERT INTO table_name VALUES(○, ○, ○)
```

明示的なテーブルのロック、デッドロック

明示的にテーブルをロックする

DBにはトランザクション以外にもテーブルをロックするコマンドが存在する

LOCK TABLE テーブル名 READ

- 実行したセッションは、テーブルを **読み込むことができるが、書き込むことはできない**
- 他のセッションも、テーブルを **読み込むことはできるが、書き込むことはできない**

```
LOCK TABLE table READ
```

LOCK TABLE テーブル名 WRITE

- 実行したセッションは、テーブルを **読み込むことも書き込むこともできる**
- 他のセッションも、テーブルを **読み込むことも書き込むことはできない**

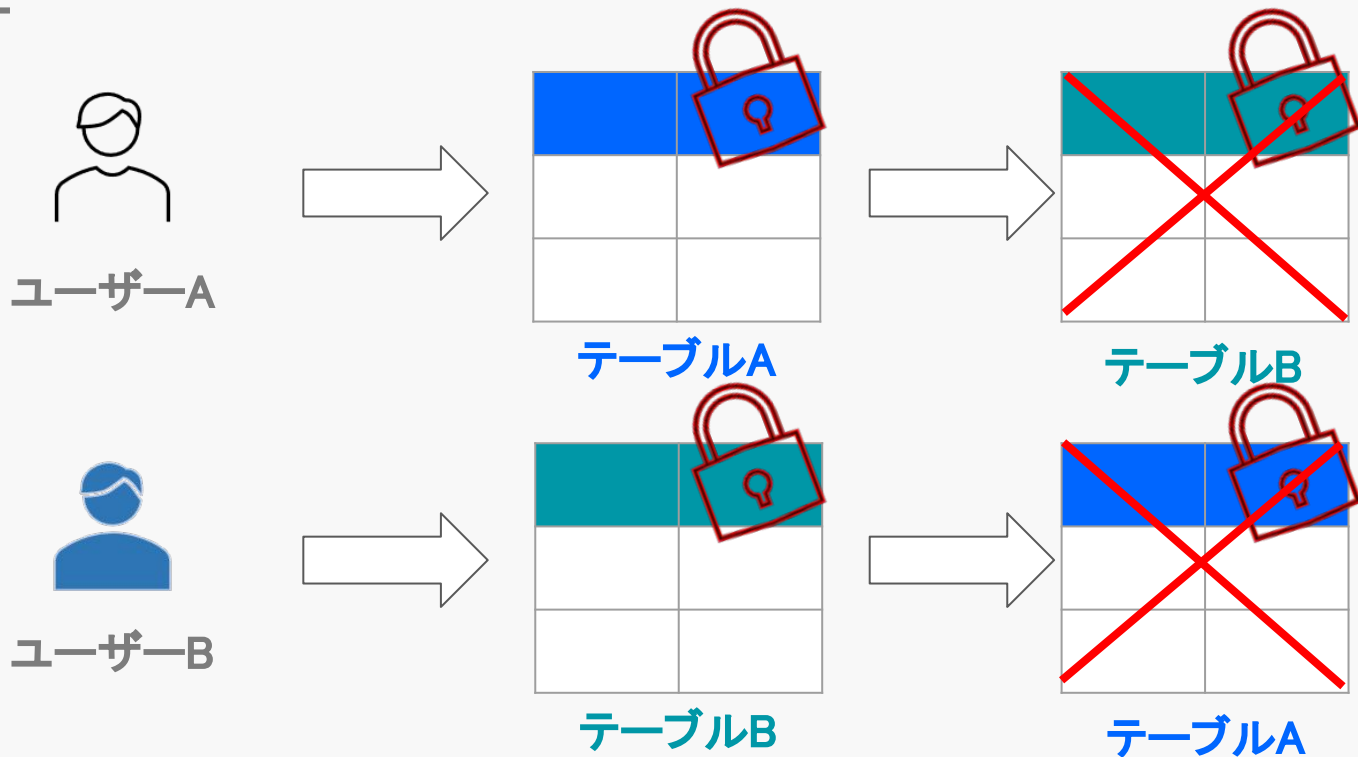
```
LOCK TABLE table READ
```

UNLOCK TABLES

現セッションの保有するテーブルロックを全て解除する

デッドロック

2つのセッションが**互いの更新対象**のテーブルをロックしていて、処理が進まない状態のこと



アプリケーションとロックとトランザクション

アプリケーションでのトランザクション処理

アプリケーションで、トランザクションのコードを記述することで、トランザクションを実行することができる

