10-3　トランザクションの分離性

10-3-1　トランザクションの分離レベル

トランザクションの分離性：トランザクションは、別のトランザクションによる処理の影響を受けない

しかし、実際には他のトランザクションの影響を受けてしまうことがある。

・ダーティリード：別トランザクションの**まだコミットされていない更新／挿入／削除結果**が見えてしまうこと
・ファジーリード：別のトランザクションの**コミットされた更新／削除結果**が見えてしまうこと
・ファントムリード：別のトランザクションの**コミットされた挿入結果**が見えてしまうこと
・直列化異常：複数のトランザクションのコミット結果が、どのような処理順序を仮定しても一貫性のない状態になってしまうこと

Query Editor　Query History

```
1  create table tbl (id integer, count integer);
2  insert into tbl values (1, 10), (1, 20), (2, 10), (2, 20);
3  select * from tbl;
```

Data Output　Explain　Messages　Notifications

	id integer	count integer
1	1	10
2	1	20
3	2	10
4	2	20

Query Editor　Query History

```
1  select 1, sum(count) from tbl where id = 2;
```

Data Output　Explain　Messages　Notifications

	?column? integer	sum bigint
1	1	30

selectでは、表の数字ではなく、単に「1」という数字を取り出している。
そのため、数字は何でも良い。試しに「50」でも実行してみる。

Query Editor　Query History

```
1  select 50, sum(count) from tbl where id = 2;
```

Data Output　Explain　Messages　Notifications

	?column? integer	sum bigint
1	50	30

Read uncommitted：いずれも起こりうる

Read committed：ファジーリード、ファントムリード、直列化異常が起こる
Repeatable read：ファントムリード、直列化異常が起こる
Serializable：いずれも起こらない

PostgreSQLでは、デフォルトでRead committedとなっており、ファジーリードとファントムリードと直列化異常が起こる。

10-3-2　分離レベルの設定

分離レベルの設定は、「default_transaction_isolation」や「BEGIN」、「SET文」などで可能である。

Query Editor　Query History

```
1  show default_transaction_isolation;
```

Data Output　Explain　Messages　Notifications

	default_transaction_isolation text
1	read committed

Query Editor　Query History

```
1  set default_transaction_isolation to 'serializable';
2
3  show default_transaction_isolation;
```

Data Output　Explain　Messages　Notifications

	default_transaction_isolation text
1	serializable

現在のトランザクションの分離レベルのみを変更する。

Query Editor　Query History

```
1  begin isolation level read committed;
2
3  show transaction_isolation;
```

Data Output　Explain　Messages　Notifications

	transaction_isolation text
1	read committed

10-3-3　分離性による振る舞いの違い

Query Editor　Query History

```
1  drop table tbl;
2  create table tbl (id integer, value integer);
3  insert into tbl values (1, 100), (2, 150), (3, 200);
4  select * from tbl;
```

Data Output　Explain　Messages　Notifications

	id integer	value integer
1	1	100
2	2	150
3	3	200

トランザクション（tx1）とトランザクション（tx2）を並行して実行していく。

Query Editor　Query History

```
1  begin;
2
3   begin;
4
5  update tbl set value=150 where id = 1;
6  select * from tbl order by id;
```

Data Output　Explain　Messages　Notifications

	id integer	value integer
1	1	150
2	2	150
3	3	200

何度か実行してみたが、pgAdminでは教科書通りの方法で
トランザクションの分離性による振る舞いの違いを見ることはできないため
この節のコードはスキップとする。

Read committedでは、tx1が更新してコミットしたら、tx2がまだコミットしていなくても更新結果が見えてしまうが
Serializableにおいては、tx2もコミットし終わってからでないと更新結果が見えないようになっている。
このようにトランザクションの分離性では、トランザクション実行中での他トランザクションへの影響はないことが理想的である。