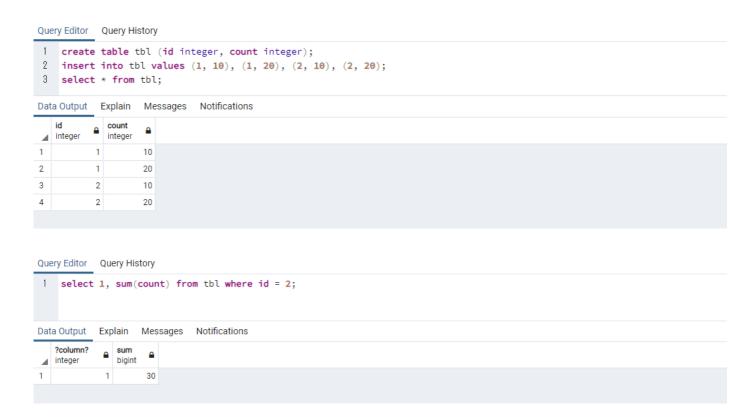
# 10-3 トランザクションの分離性

#### 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



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。



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

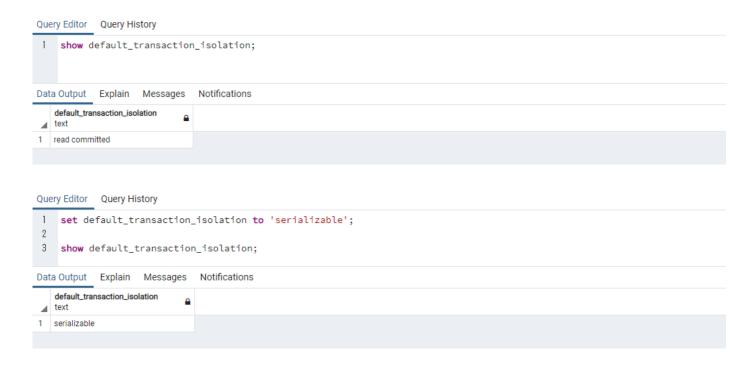
Repeatable read:ファントムリード、直列化異常が起こる

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。



### 現在のトランザクションの分離レベルのみを変更する。

Que	y Editor Query History	
1	begin isolation level read committed;	
2		
3	<pre>show transaction_isolation;</pre>	
Data Output Explain Messages Notifications		
4	ransaction_isolation ext	
1	ead committed	

## 10-3-3 分離性による振る舞いの違い



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。



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。