トランザクション管理

授業計画

第1回　ガイダンス・データベースの基本概念
第2回　データモデル
第3回　関係代数
第4回　データベース設計
第5回　リレーションの正規化
第6回　中間まとめ
第7回　関係データベース言語（SQL1）
第8回　関係データベース言語（SQL2）
第9回　計算機実習
第10回　データの検索機構　MySQL実習
第11回　トランザクション管理　MySQL実習
第12回　障害回復　MySQL実習
第13回　分散データベース　MySQL実習
第14回　期末まとめ
第15回　応用技術と将来動向　MySQL実習

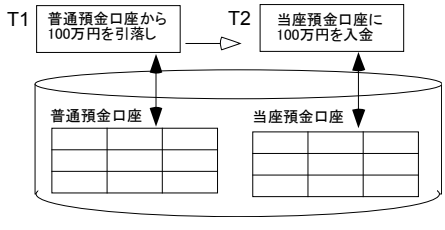
講義内容

- トランザクション管理の概要
- ACID特性（原子性，整合性，隔離性，耐久性）
- 同時実行制御
  - 並列トランザクションに起因する問題
  - 同時実行制御
  - （ロック／アンロック，時刻印，楽観的同時実行制御）
- 直列可能性
  - 直列可能性　2相ロッキングプロトコル
- 隔離性水準

１．トランザクション管理の概要

■トランザクション（transaction）：　論理的にそれ以上分割することができない一連の操作であり，障害回復に関する基本単位としてSQL文実行の並びとして定義されている。

分割出来ない一連の操作の例
（顧客の普通預金口座から当座預金口座に100万円を移動させる銀行のトランザクション）



コンピュータから見れば2つの操作（普通預金口座から100万円を引落す出金処理，当座預金口座に100万円を入金する入金処理から構成される。

９．トランザクション管理

１．トランザクション管理の概要

■トランザクションを終了するためのSQL

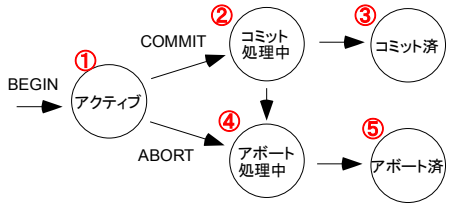
- トランザクション処理が成功した場合に，その結果を確定させることを“トランザクションをコミットする”という。
- **COMMIT文**は，トランザクションをコミットし正常に終了させる。
- トランザクションをコミットすると，そのトランザクションで実行した全ての**更新がデータベースに確実に反映される**ことになる。

- トランザクション処理が失敗した場合に，その処理結を反映させないことを“**ロールバック**する”という。
- **ROLLBACK文**は，トランザクションをロールバックし終了させる。
- トランザクションをロールバックすると，そのトランザクションで実行した**全ての更新が取り消される**ことになる。

９．トランザクション管理

１．トランザクション管理の概要

■トランザクションの状態遷移



①アクティブ： トランザクションを実行中の状である。

②コミット処理中： COMMIT命令が呼び出され，コミットのための処理を実行中の状態である。

③コミット済： コミットのための理が終了した状態である。

④アボート済： アボートのための処理が終了した態である。

⑤アボート処理中： アボートのための処理を実行中の状態である。明示にABORT命令が呼び出された場合の他，何らかの理由でコミット処理を正常に終了できない場合などもこの状態に到達する。

９．トランザクション管理

２．ACID特性

■ **ＡＣＩＤ特性**（ACID properties）：**トランザクションが有効であるための特性**

①原子性（atomicity）：**トランザクションの処理がすべて実行されるか，まったく実行されないかのいずれかで終了するという特性である。**

②整合性（consistency）：**データベースの内容が矛盾がない状態であるという特性である。**

③隔離性/独立性（isolation）：**複数のトランザクションを同時に実行した場合と，順番に実行した場合の結果が一致するという特性である。**

④耐久性（durability）:**トランザクションが正常終了すると，障害が発生しても更新結果がデータベースから消失しないという特性である。**

９．トランザクション管理

３．同時実行制御

■ **並列トランザクションに起因する問題**

① **ロストアップデート**： あるトランザクションT1の更新が，他のトランザクションT2によって上書きされ，T1の更新した内容が失われてしまう現象をロストアップデートという。

② **コミットされていない依存性の問題**： あるトランザクションが，他のトランザクションのアクセスしているデータを更新してしまうことにより不整合が発生する問題を“コミットされていない依存性の問題”という。

ダーティリード（dirty read），
ノンリピータブルリード（non-repeatable read），
ファントムリード（phantom read）

３．同時実行制御

■ **同時実行制御**

- 並列トランザクションに起因する問題を解決するために，DBMSは同時実行制御の機能を有している。
- 同時実行制御の方式として，ロック／アンロック，時刻印，および楽観的同時実行制御がある。

（1）ロック／アンロック：

- データベースにアクセスする際，アクセス対象となるデータを他のトランザクションからアクセスできないロック状態にし，排他制御する方法である。
- ロックの対象には，レコード（行），テーブル（表），データベースがある。
- ロックの対象が細かくなれば，同時に実行できるトランザクションが多くなり，トランザクション同士の待ち時間も少なくなるが，ロックを管理するオーバヘッドが大きくなる。

３．同時実行制御

（1）ロック／アンロック：

ロックのタイプ

共有ロック	トランザクションが SELECT 文によりデータを参照するものであるときのロックのモードである。共有ロックによりデータがロックされている間は，他のトランザクションからデータを参照することはできるが，データの変更はできない。
占有ロック（排他ロック）	トランザクションが INSERT 文，UPDATE 文および DELETE 文であるときのロックのモードである。占有ロックされている間は，他のトランザクションからデータの参照も変更もできない。

９．トランザクション管理

３．同時実行制御

（1）ロック／アンロック：

共有ロックと占有ロックの両立性行列

	共有ロック（S loack）	占有ロック（X loack）
共有ロック（S loack）	○	×
占有ロック（X loack）	×	×

両立性行列は，**占有ロックは排他的**であるが**共有ロック同士は両立**することを示している。

９．トランザクション管理

３．同時実行制御

（1）ロック／アンロック：

デッドロック

ロック／アンロックを用いる場合，双方のトランザクションが他方に必要なデータをロックしてしまい，どちらもアンロック待ちに陥って処理が進まなくなる状態になることがある。

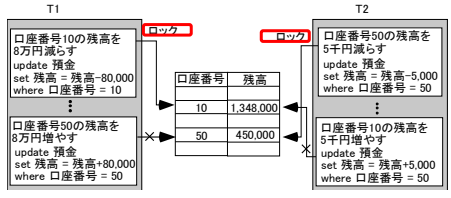
このことをデッドロック（dead lock）が発生したという。

９．トランザクション管理

３．同時実行制御

（1）ロック／アンロック：

銀行口座間の振込み処理におけるデッドロックの発生例



- T1は口座番号10の口座から口座番号50の口座に8万円の振込み処理，
- T2はこれとは逆に口座番号50の口座から口座番号10の口座に5千円の振込み処理と仮定

- T1が口座番号10のレコードを変更後，T2が口座番号50のレコードを変更したとする。この時点で，T1は口座番号10のレコードを，T2は口座番号50のレコードを占有ロックする。

- その後，T1は口座番号50のレコードを変更しようとするが，そのレコードにはT2が占有ロックを掛けているため待機状態になる。
- 一方，T2も口座番号10のレコードを変更しようとするが，このレコードはT1が占有ロックを掛けているためやはり待機状態になる。

互いに相手のロックが解除されるのを待機して処理が進まなくなる状態がデッドロックである。

９．トランザクション管理

３．同時実行制御

（1）ロック／アンロック：

デッドロックを回避する方法

① 資源をアクセスする順序を統一する（資源割当の固定化という）。
② 検索後に更新する場合は，検索時点から同時実行が来ない占有ロックを掛ける。

デッドロックの検知

- DBMSのロック管理機能がロック解除待ちの有効グラフを作成することによって行われる。
- ロックを掛けた場合に，この有効グラフがループする場合にはデッドロックが発生する。
- この有効グラフのことを待ちグラフ（WFG : wait-for graph）という。

９．トランザクション管理

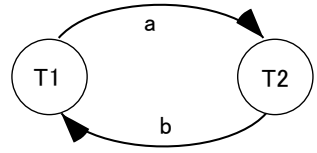
３．同時実行制御

（1）ロック／アンロック：

デッドロックの発生例（トランザクションと待ちグラフ）

Time	T1	T2
t1	L1(a)	
t2	R1(a)	L2(b)
t3	a=a+10	R2(b)
t4	W1(a)	b=b−20
t5	L1(b)	W2(b)
t6		L2(a)
t7		

ループが存在するので，デッドロックの存在を検知することができる。



L1(a)：T1による資源aのロック
R1(a)：T1による資源aのread
W1(a)：T1による資源aのwrite
L1(b)：T1による資源bのロック
L2(b)：T2による資源bのロック
R2(b)：T2による資源bのread
W2(b)：T2による資源bのwrite
L2(a)：T2による資源aのロック

９．トランザクション管理

３．同時実行制御

（2）時刻印：

- トランザクション開始時の時刻印をトランザクション毎に保持しておき，データのアクセスが競合したときには，先に開始したトランザクションを優先して，他のトランザクションを取り消す方法である。
- 時刻印を用いた方法は，ロックを用いないためデッドロックの問題が発生しない。
- 処理時間の長いトランザクションほどロールバックされる可能性が高くなる問題点や，更新中のデータを他のトランザクションから参照されてしまうという問題点がある。

３．同時実行制御

（3）楽観的同時実行制御：

- 更新予定のデータに対する更新前の値を保持しておき，更新時にその値を用いて，更新するデータが他のトランザクションによって更新されていないことを確認後に実際に更新する方法である。
- 更新時に他のトランザクションによって更新されていた場合は，トランザクションをロールバックして取り消すことになる。
- 時刻印を用いた排他制御と同様に，ロックを用いないためデッドロックの問題が発生しない方法であるが，トランザクション間でアクセスの競合頻度が少ない場合を想定した方式である。

４．直列可能性

実行するトランザクションを全て直列に実行すれば，トランザクション間の干渉がなくなりデータの一貫性を保障することができる。

（1）直列可能性：

- 並列実行に起因する問題発生を防止するために考えられた概念である。
- すなわち，2つのトランザクションT1とT2において，T1とT2を並列に実行した結果が，T1→T2またはT2 →T1の順に実行した結果と等しい時，このスケジュールは直列可能であるという。
- 直列可能でないスケジュールでは，トランザクションの干渉によるデータの不整合が発生する。

４．直列可能性

■ 直列可能性が保障されるトランザクションの例

T1とT2ともに，アクセス競合資源を単純にロックし，単純にアンロックしているので，重複実行されることがなく直列可能性が保障されている。

T1	T2
LOCK a	LOCK a
READ a	READ a
a = a + x	LOCK b
STORE a	READ b
LOCK b	UNLOCK b
READ b	UNLOCK a
b = b + x	
STORE b	
UNLOCK b	
UNLOCK a	

LOCK：ロック
READ：読込み
STORE：書込み
UNLOCK：アンロック

４．直列可能性

（2）2相ロッキングプロトコル

- トランザクションにおいて，データ操作の前に排他資源に対し，一斉にロックを掛け，操作終了後に一斉にロックを解く方式を2相ロッキングプロトコルという。

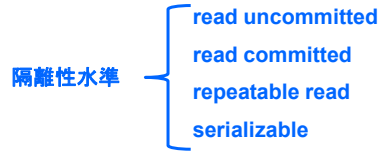
T1	
LOCK a	成長フェーズ（第1相）
LOCK b	
READ a	
READ b	
b = a+ b	
STORE b	
UNLOCK a	縮退フェーズ（第2相）
UNLOCK b	

LOCK：ロック
READ：読込み
STORE：書込み
UNLOCK：アンロック

９．トランザクション管理

５．隔離性水準

- トランザクションの**隔離性水準**は，同時に実行される**他のトランザクションからの影響を受ける度合い**を意味する。
- 隔離性水準が高いほど他のトランザクションの影響を受けない。
- 逆に，隔離性水準が低いほど他のトランザクションの影響を受けることになるが，同時実行性能を向上させることができる。
- 隔離性水準には，**4つのレベル**があり，各レベルでトランザクションの並列トランザクションに起因する問題（ダーティリード，ノンリピータブルリード，ファントムリード）が異なる。



９．トランザクション管理

５．隔離性水準

① **未コミットデータの読込み**（read uncommitted）： コミットされていない更新データを他のトランザクションが読み込むことを許可する水準であり，全ての問題が発生する。

② **コミット済みデータの読込み**（read committed）： コミットされていない更新データを他のトランザクションが読み込むことを許可しない水準であり，ダーティリードを防止することができる。

③ **繰り返し可能読込み**（repeatable read）： 同じデータを読込む場合に，当該トランザクションが終了するまで他のトランザクションによりデータが更新されないことを保障する。ダーティリードとノンリピータブルリードを防止することができる。

④ **直列化可能**（serializable）： 直列可能性を保障するレベルである。全ての問題を防止することができる。

５．隔離性水準

隔離性水準と発生する問題

隔離性水準	ダーティリード（不正読込み）	ノンリピータブルリード	ファントムリード
read uncommitted	発生する	発生する	発生する
read committed	発生しない	発生する	発生する
repeatable read	発生しない	発生しない	発生する
serializable	発生しない	発生しない	発生しない

まとめ

- トランザクション管理の概要
- ACID特性（原子性，整合性，隔離性，耐久性）
- 同時実行制御
  - 並列トランザクションに起因する問題
  - 同時実行制御
  - （ロック／アンロック，時刻印，楽観的同時実行制御）
- 直列可能性
  - 直列可能性　2相ロッキングプロトコル
- 隔離性水準