Database Systems

Transactions, Concurrency Control and Recovery

Jing Sun and Miao Qiao
The University of Auckland

Transactions

- Transaction : a **logical unit** of database processing that must be completed in its entirety to ensure correctness

Examples: Relation *Accounts*(*acctNo*, *balance*)

- UPDATE Accounts SET balance = balance + 100 WHERE accNo = 456;
- UPDATE Accounts SET balance = balance - 100 WHERE accNo = 123;

Other applications:

- Airline reservations
- Banking (credit card transaction)
- Online retail system
- ...

Reading material: Chapter 20 of the textbook.

Transactions

- Transaction : a **logical unit** of database processing that must be completed in its entirety to ensure correctness

Write transactions in host languages:

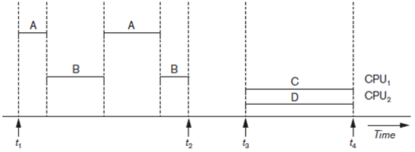
- BEGIN TRANSACTION
- READ / WRITE OPERATIONS
  - READ: data retrievals
  - WRITE: updates, insertions and deletions
- COMMIT: the transaction completes successfully, all the updates are permanently made to the database
- ROLLBACK: the transaction ends unsuccessfully, undo the operations in the transaction
- END TRANSACTION

Why we have commit and rollback operations?

Transaction: Concurrency Control

THE UNIVERSITY OF AUCKLAND
Te Whare Wananga o Tamaki Makaurau
NEW ZEALAND
SCIENCE
DEPARTMENT OF COMPUTER SCIENCE

- Transaction processing systems : systems with large databases and hundreds of concurrent users
- Consider two types of operations:
  - computation
  - input and output (I/O)
- One cpu, how to serve multiple users concurrently to minimize the average delay?
  - Interleaved concurrency, e.g., $[t_1, t_2]$, when a process requires I/O, one keeps the cpu busy by switching to execute another process instead of waiting.
- Multiple cpus, how to serve multiple users concurrently to minimize the average delay? E.g., $[t_3, t_4]$. parallel processing + interleaved concurrency



Transaction: Recovery

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND
SCIENCE
DEPARTMENT OF COMPUTER SCIENCE

Types of failures:

- Computer failure (system crash)
- Concurrency control enforcement
- Disk failure
- Physical problems and catastrophes

Transaction: ACID Properties

■ Atomicity	Transaction performed in its entirety or not at all
■ Consistency preservation	Takes database from one consistent state to another
■ Isolation	Not interfered by other transactions
■ Durability or permanency	Changes must be persist in the database

Transactions

How to achieve ACID?

- BEGIN TRANSACTION
- READ / WRITE OPERATIONS
  - READ: data retrievals
  - WRITE: updates, insertions and deletions
- COMMIT: the transaction completes successfully, all the updates are permanently made to the database
- ROLLBACK: the transaction ends unsuccessfully, undo the operations in the transaction
- END TRANSACTION

Transaction

Recall that a **database** is a collection of **named data items**.

- A file
- A subtree on the B^+-tree
- A disk block
- A database record
- A field

The size of a data item is called its **granularity**. We assume that each data item has a unique name.

Transaction

For a data item (named) X, a transaction T may carry out two database access operations:

- **read(X)**: Reads a database item named X into a program variable named X_T
- **write(X)**: Write a program variable named X_T to a database item named X

When the context is clear, we omit the subscription T of a program variable.

Discussion: consider the **buffer manager**, what are the steps of read(X) and write(X), respectively?

Transaction

Read(X)

- Find the address of the disk block of X
- Copy the disk block of X to the memory buffer
- Copy X from the buffer to the program variable X_T

Write(X)

- Find the address of the disk block of X
- Copy the disk block of X to the memory buffer
- Copy X from the program variable X_T to the buffer
- Write the updated block from the buffer back to the disk

controlled by buffer manager

Transaction: Concurrency Issues

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND
SCIENCE
DEPARTMENT OF COMPUTER SCIENCE

Given two transactions T_1 and T_2,

(a)

T_1
read_item(X);
$X := X - N$;
write_item(X);
read_item(Y);
$Y := Y + N$;
write_item(Y);

(b)

T_2
read_item(X);
$X := X + M$;
write_item(X);

The interleaved processing of T_1 and T_2 may lead to two problems:

- **Lost update**
- **Dirty read**

Transaction: Concurrency Issues

Lost update.

(a)

Time	T_1	T_2
	read_item(X); $X := X - N$;	
		read_item(X); $X := X + M$;
	write_item(X); read_item(Y);	
		write_item(X);
	$Y := Y + N$; write_item(Y);	

Plug $X = 10$, $M = 2$ and $N = 3$ in the execution, what will X be after T_1 and T_2?

Transaction: Concurrency Issues

Lost update.

(a)

Time ↓

T_1	T_2
read_item(X); $X := X - N$;	
	read_item(X); $X := X + M$;
write_item(X); read_item(Y);	
	write_item(X); ← Item X has an incorrect value because its update by T_1 is *lost* (overwritten).
$Y := Y + N$; write_item(Y);	

Transaction: Concurrency Issues

Dirty read.

(b)

Time ↓

T_1	T_2

read_item(X);
$X := X - N$;
write_item(X);	

	read_item(X);
	$X := X + M$;
	write_item(X);

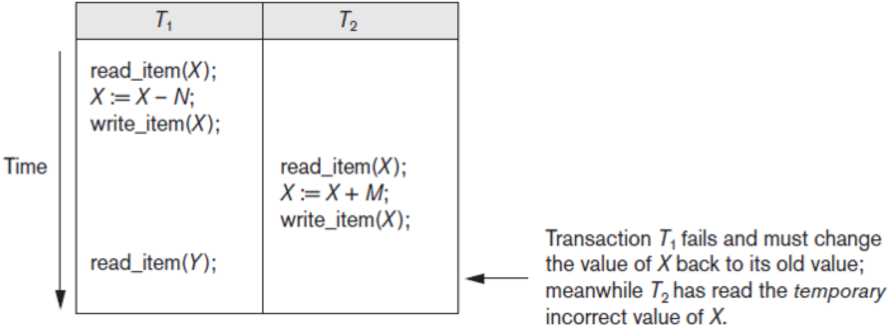
read_item(Y);	

Plug $X = 10$, $M = 2$ and $N = 3$ in the execution, what will happen in T_2 if T_1 rolls back after read_item(Y)?

Transaction: Concurrency Issues

Dirty read. Accessing an updated value that has not been committed is considered a dirty read because it is possible for that value to be rolled back to its previous value. If you read a value that is later rolled back, you will have read an invalid value.

(b)



Transaction: Concurrency Issues

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND
SCIENCE
DEPARTMENT OF COMPUTER SCIENCE

Unrepeatable read. Always read committed data items but get two different values in reading the same data item.

T_1	T_2
Read(X)	
	Read(X)
...	
	...
	Write(X)
	Commit
Read(X)	
...	

Transaction: Concurrency Issues

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND
SCIENCE
DEPARTMENT OF COMPUTER SCIENCE

Phantom record. Always read committed values and there is no "unrepeatable read", the query result may have a phantom record t.

- select * from R where c_1: read all database records that satisfy c_1
- insert a record t that satisfies c_1 to R

Time	T_1	T_2
Order	select * from R where c_1	
1		insert a record t that satisfies c_1

Time	T_1	T_2
Order		insert a record t that satisfies c_1
2	Select * from R where c_1	

Transaction: System Log

System log: a sequential, append-only file that tracks the transaction operations.

Associate each transaction with an ID, e.g., T
System log includes the following items:

- [start_transaction, T]
- [write, T, X, old_value, new_value]
- [read, T, X]
- [commit, T]
- [abort,T] (rollback)

Transaction: System Log

System log: to ensure that the database is not affected by failure

- Log buffer
- Log file is backed up periodically
- Commit point, undo and redo operations

Transaction: Commit Point

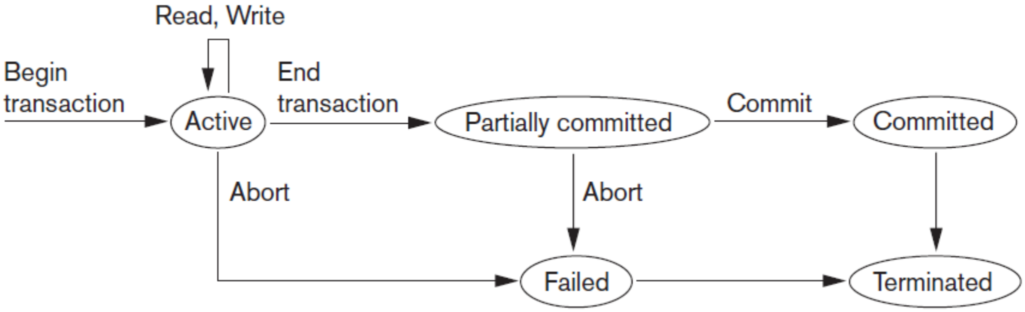
A transaction reaches its **commit point** if:

- All of its database access operations have been executed successfully,
- The effect of the transaction operations to the database has been recorded (**force-write** log buffer before commit point) in the log.

Beyond the commit point, the transaction is called **committed**: its effect must be permanently recorded in the database.

Transaction: States

State transitions via operations:



THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

SCIENCE

Thank you for your attention!

Any questions?