Database Management Systems

Transaction Management

Anomalies

Transactions

- A transaction consists one or multiple queries and/or updates that can be translated to a sequence of requests to read/write operations on physical objects in the database.
- Each transaction terminates in one of the two ways:
  - commit (keep all the change(s) made by this transaction)
  - abort/rollback (erase any change made earlier by this transaction, as if it never happened in the first place)
- Goals:
  - concurrent execution of transactions
  - guarantee data integrity (consistent data only in the database)

Multi-user System

- Very few database systems assume to be single user systems
- In a multi-user system, multiple users/applications need and should be allowed to use the system concurrently.
- How to execute concurrent processes
  - serially
  - interleaved concurrency
  - parallel processing (with multiple CPU) and access data concurrently

Assumptions

- Assume that each transaction is guaranteed to leave a database in a consistent state **after** it completes.
- A transaction may not keep the database consistent while it is still in progress.
- Assume that all transactions access a single copy of data, i.e., there is only one database.
- Assume that the database contains a fixed set of objects (for now)
    - individual attributes
    - records
    - physical pages
    - relations/tables
    - files

ACID Properties

- Traditional database applications usually require ACID properties in transaction management:
- **A**tomicity: all or nothing
- **C**onsistency: only consistent data in database
- **I**solation: as if only one transaction at a time
- **D**urability: committed transactions are forever

BASE Properties

- Newer applications and database models (such as NoSQL) usually are developed for less rigid situations, and sometimes favour more flexible transaction management properties:

- **B**asic **A**vailability.

- **S**oft State.

- **E**ventual Consistency.

Anomalies Due to Interleaved Execution (I)

- Reading Dirty Data (WR Conflicts)

- Suppose X is a joint bank account.

- T1 wants to withdraw
  $N from account X,
  but decided to abort
  the transaction in the end;

- T2 wants to deposit
  a cheque of $M to account X.

T_1	T_2

read-item(X, v);
v' = v - N;
write-item(X, v');	

	read-item(X, v);
	v' = v + M;
	write-item(X, v');
	commit;

abort;	

Anomalies Due to Interleaved Execution (II)

- Unrepeatable Reads (RW Conflicts)

- T1 wants to transfer $N from account X to account Y;
- T2 wants to find out your total asset worth (for the purpose of process your mortgage application).

T_1	T_2
	sum = 0;

read-item(X, v);
v' = v - N;
write-item(X, v');	

	read-item(X, v);
	sum = sum + v;
	read-item(Y, v);
	sum = sum + v;

read-item(Y, v);
v' = v + N;
write-item(Y, v');	

Anomalies Due to Interleaved Execution (III)

- Lost Updates (WW Conflicts)

- X is a joint account.
- T1 wants to transfer $N from account Y to account X;
- Meanwhile, T2 wants to deposit a cheque of $M to account X

T_1	T_2
read-item(Y, v); v’ = v - N;	
	read-item(X, v); v’ = v + M;
write-item(Y, v’); read-item(X, v);	
	write-item(X, v’);
v’ = v + N; write-item(X, v’);	