Concurrency Control

Concurrency Control

CITY UNIVERSITY LONDON
EST 1894

- Concurrency control: ensure that one user's work does not inappropriately influence another user's work
- No single concurrency control technique is ideal for all circumstances
- Trade-offs need to be made between level of protection and throughput

Atomic Transactions

CITY UNIVERSITY LONDON
EST 1894

- A transaction, or logical unit of work (LUW), is a series of actions taken against the database that occurs as an atomic unit
- Either all actions in a transaction occur - **`COMMIT`**
- Or none of them do - **`ABORT`**

Example

The common example is the transfer of money

```
read(ACC1);
ACC1 := ACC1 - 50;
write(ACC1);  read(ACC2);
ACC2 := ACC2 + 50;
write(ACC2);
```

Transaction Outcomes & Boundaries

CITY UNIVERSITY LONDON
EST 1894

- Database transaction - a set of operations of these kinds:
  - Transaction boundary/edge: begin, commit, abort (rollback)
  - Reads (SELECT), writes (DELETE, INSERT, UPDATE)
- Committed: when it completes successfully;
- Aborted (Rolled-back): when it is not executed successfully;

ACID

Transaction properties:

- Atomicity: either all the operations in a transaction are executed against the database, or none is.
- Consistency: the execution of a transaction must preserve the consistency of the database
- Isolation: the execution of a transaction must preserve the consistency of the database.
- Durability: the effects of a successfully completed transaction are stored in the database permanently (not lost due to HW/SW failure)

Concurrent Transaction

CITY UNIVERSITY LONDON
EST 1894

- Concurrent transactions: transactions that are being processed at the same time
- Can be also run on one CPU
- Transactions are interleaved
- Concurrency problems
  - Lost updates
  - Inconsistent reads

Concurrency Issues

Transfer(A, B, 100)	Transfer(B, A, 100)
1. mA = read(moneyA) // 1000	
2. mB = read(moneyB) // 1000	
	3. mB = read(moneyB) // 1000
	4. mA = read(moneyA) // 1000
	5. write(moneyB, mB - 100)
6. write(moneyA, mA - 100)	
7. write(moneyB, mB + 100) // 1100	
	8. write(moneyA, mA + 100) // 1100

Concurrency Control

CITY UNIVERSITY LONDON
EST 1894

- A definition: The mechanism of managing simultaneous operations (executed by multiple transactions) on the database preventing them to interfere with one another.
- How can the system control the interaction among, and effects of, concurrent transactions?
- Different CC mechanisms exist (in different DBMSs)
- If control of execution of concurrent operations is left to the underlying OS, many possible schedules, i.e. orders of operation executions, are possible
- Including the ones that would leave DB in an inconsistent state, and/or the ones that would send wrong results to DB users.

Transaction isolation levels

CITY UNIVERSITY LONDON
EST 1894

- The Isolation levels control the degree to which the execution of one transaction is isolated from the execution of other concurrent transactions (accessing the same data).
- A lower isolation level increases concurrency (improves performance), but can impair data correctness. Conversely, a higher isolation level ensures that data remains correct, but can negatively affect concurrency.
- ISO SQL-92 defines the following isolation levels (from the strictest to the most relaxed):
    - Serializable, Repeatable read, Read committed and Read uncommitted
    - Based on the defined anomalies: Dirty Read, Non-Repeatable and Phantom Reads.

Isolation Levels

CITY UNIVERSITY LONDON
EST 1894

Isolation Level	Transactions	Dirty Reads	Non-Repeatable Reads	Phantom Reads
TRANSACTION_NONE	Not supported	Not applicable	Not applicable	Not applicable
READ_UNCOMMITTED	Supported	Allowed	Allowed	Allowed
READ_COMMITTED	Supported	Prevented	Allowed	Allowed
REPEATABLE_READ	Supported	Prevented	Prevented	Allowed
SERIALIZABLE	Supported	Prevented	Prevented	Prevented

Serializability

CITY UNIVERSITY LONDON
EST 1894

- Transaction Schedule: Sequence of operations by concurrent transactions that preserves the order of the operations in each individual transaction
- Serial Schedule: The transactions are performed in serial order (operations are not interleaved) sequentially with no overlap in time
- A transaction schedule is serializable if there exists a serial schedule leading to the same state of the database
- A database transaction manager is serializable if this happens for all schedules

Locks

CITY UNIVERSITY LONDON
EST 1894

- How can we guarantee serializability?
- We can use locks on the data
- It controls the access to the data
- Two lock types:
  - Exclusive (write) lock
  - Shared (read) lock

Lock Granularity trade-offs

CITY UNIVERSITY LONDON
EST 1894

		Concurrency/Performance	Data Correctness
Coarse		low	high
	Entire DB		
	File		
	Page		
	Record		
	Field value		
Fine		high	low

Two-Phase Locking Protocol (2PL)

CITY UNIVERSITY LONDON

EST 1894

- Phase 1 (Growing/expanding phase)
  - A transaction T always acquires a lock on an object before reading (S-lock), or writing (X-lock), it.
- Phase 2 (Shrinking phase)
  - After releasing a lock, transaction T cannot acquire a new lock.

There is impossible to get lock after the first unlock.

Several variations: Conservative 2PL, Strict 2PL, Strong Strict 2PL

Original 2PL takes a lock when necessary

Conservative 2PL takes all at the beginning

Deadlock

When transactions needs some resource that is already locked and it is impossible to release anything.

```
Transfer(A, B, 100)                Transfer(B, A, 100)

1. lock(accountA)
2. accountA += 100
                                   3. lock(accountB)
                                   4. accountB += 100
5. lock(accountB) // cannot
                                   6. lock(accountA) // cannot
```

Deadlock. Consequences.

CITY UNIVERSITY LONDON

EST 1894

- It cannot happen in Conservative PL. It just takes the locks in some proper order.
- To find it, the database uses dynamic ownership graph.
- To solve it, one of the transactions is rolled back and timeouted.
- There is some logic which decides which one to roll back.

Sources

C.J. Date, An Introduction to Database Systems (Sections 15 and 16)

J. Ullman and J. Widom, A First Course in Database Systems (Section 6.6)

Bernstein P.A., Hadzilacos V., Goodman N. Concurrency Control and Recovery in Database Systems