ACID, Transaction Management and Conccurrency Control

Hogeschool Rotterdam
Rotterdam, Netherlands

Introduction

Lecture topics

- ACID.
- Transaction Management.
- Concurrency Control.

Transactions and Concurrency

Reasons

- *Concurrent* execution of user programs is essential for good DBMS performance. Because disk accesses are frequent, and relatively slow, it is important to keep the cpu busy by working on several user programs concurrently.
- A user's program may carry out many operations on the data retrieved from the database, but the DBMS is only concerned about what data is read/written from/to the database.
- A *transaction* is the DBMS's abstract view of a user program: a sequence of reads and writes .

Transactions and Concurrency

- Users submit transactions, and can think of each transaction as executing by itself.
  - *Concurrency* is achieved by the DBMS, which interleaves actions (reads/writes of DB objects) of various transactions.
  - transaction must leave the database in a consistent state if the DB is consistent when the transaction begins.
  - Beyond this, the DBMS does not really understand the semantics of the data. (e.g., it does not understand how the interest on a bank account is computed).
- Issues: Effect of interleaving transactions, and crashes.

The ACID properties

- A RDBS ensures this four properties of a transaction:
  - Atomicity: states that database modifications must follow an all or nothing rule.

The ACID properties

- A RDBS ensures this four properties of a transaction:
  - Atomicity: states that database modifications must follow an all or nothing rule.
  - Consistency: states that only valid data will be written to the database.

The ACID properties

- A RDBS ensures this four properties of a transaction:
  - Atomicity: states that database modifications must follow an all or nothing rule.
  - Consistency: states that only valid data will be written to the database.
  - Isolation: requires that multiple transactions occurring at the same time not impact each others execution.

Transactions and Concurrency

The ACID properties

- A RDBS ensures this four properties of a transaction:
  - Atomicity: states that database modifications must follow an all or nothing rule.
  - Consistency: states that only valid data will be written to the database.
  - Isolation: requires that multiple transactions occurring at the same time not impact each others execution.
  - Durability: ensures that any transaction committed to the database will not be lost.

Atomicity of Transactions

- A transaction might commit after completing all its actions, or it could abort (or be aborted by the DBMS) after executing some actions.
- A very important property guaranteed by the DBMS for all transactions is that they are atomic. That is, a user can think of a transaction as always executing all its actions in one step, or not executing any actions at all.
- DBMS logs all actions so that it can undo the actions of aborted transactions.

Transaction

```
BEGIN;
UPDATE ships SET name = 'Alpha'
        WHERE name = 'Oleg';
SAVEPOINT my_savepoint;
UPDATE ships SET integrity = 30
        WHERE name = 'Alpha';
-- oops ... forget to update also Beta
ROLLBACK TO my_savepoint;
UPDATE ships SET integrity = integrity + 10
        WHERE name = 'Beta';
COMMIT;
```

When are those updates valid states for other actions?

Example of a Transaction

Transactions

- We simplify transaction queries for readability in our next example

Example of a Transaction

Transactions

- We simplify transaction queries for readability in our next example
- Consider those two transactions:
  - T1: BEGIN A=A+100, B=B-100 END
  - T2: BEGIN A=1.06*A, B=1.06*B END
- Intuitively, the first transaction is transferring $100 from B's account to A's account. The second is crediting both accounts with a 6% interest payment.
- How are those transaction scheduled?

Example of a Transaction

Transactions

- possibility one
  - T1:A=A+100, B=B-100
  - T2: A=1.06*A, B=1.06*B
- possibility two
  - T1:A=A+100, B=B-100
  - T2: A=1.06*A, B=1.06*B
- The DBMS's view of the second schedule:
  - T1:R(A),W(A), R(B), W(B)
  - T2: R(A), W(A), R(B), W(B)

R(A): read A, W(A): write A