Transaction Management

Transaction Management

Transaction Management

- Transaction management comprises the following two areas
  - **Recovery**, i.e., handling of failure situations that have occurred and are often unavoidable.
  - **Concurrency control** of multiple transactions running simultaneously on the database.

Example of a Transaction

- Transfer money from account A to account B:
  - Read the balance of A into the variable a: **read**(A,a);
  - Reduce the balance by 50 euros: $a/ := a/ - 50$;
  - Write the new balance into the database: **write**(A,a);
  - Read the balance of B into the variable b: **read**(B,b);
  - Increase the balance by 50 euros: $b/ := b/ + 50$;
  - Write the new balance into the database: **write**(B,b);

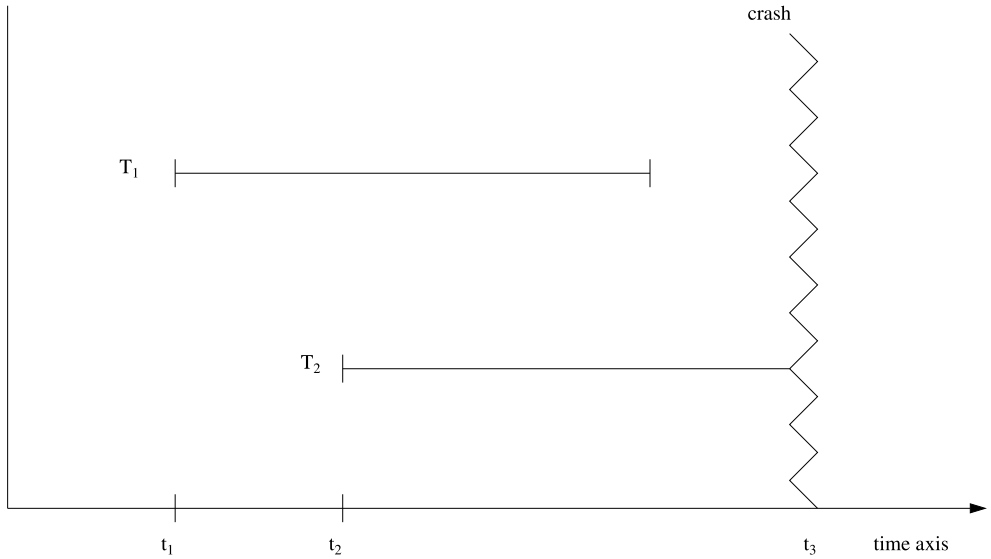
Operations

- **begin of transaction** (**BOT**):
  - Marks the beginning of a transaction
- **commit**:
  - Successful termination of a transaction
  - Durable propagation of all changes into the database
- **abort**:
  - Self-abort of the transaction, unsuccessful termination
  - Resetting the database to the state before the beginning of the transaction

Operations(2)

- **define savepoint**:
  - Define a savepoint to which the (still active) transaction can be rolled back.
- **backup transaction**:
  - The still active transaction is rolled back to the most recent (last created) savepoint
  - Possibly also rolling back to earlier savepoints

System Crash



- Changes of transaction T_1, completed at time t_3, must be present in the database
- Changes of transaction T_2, not yet completed at t_3, must be completely removed from the database (execution of T_2 by restarting it)

- **commit (work)**:
  - End the transaction and make the changes persistent
  - only works if no other errors occurred (e.g., consistency violation by the transaction)
- **rollback (work)**:
  - End the transaction and undo all changes
  - Unlike **commit**, the DBMS must always be able to guarantee the "successful" execution of **rollback**

- Example:

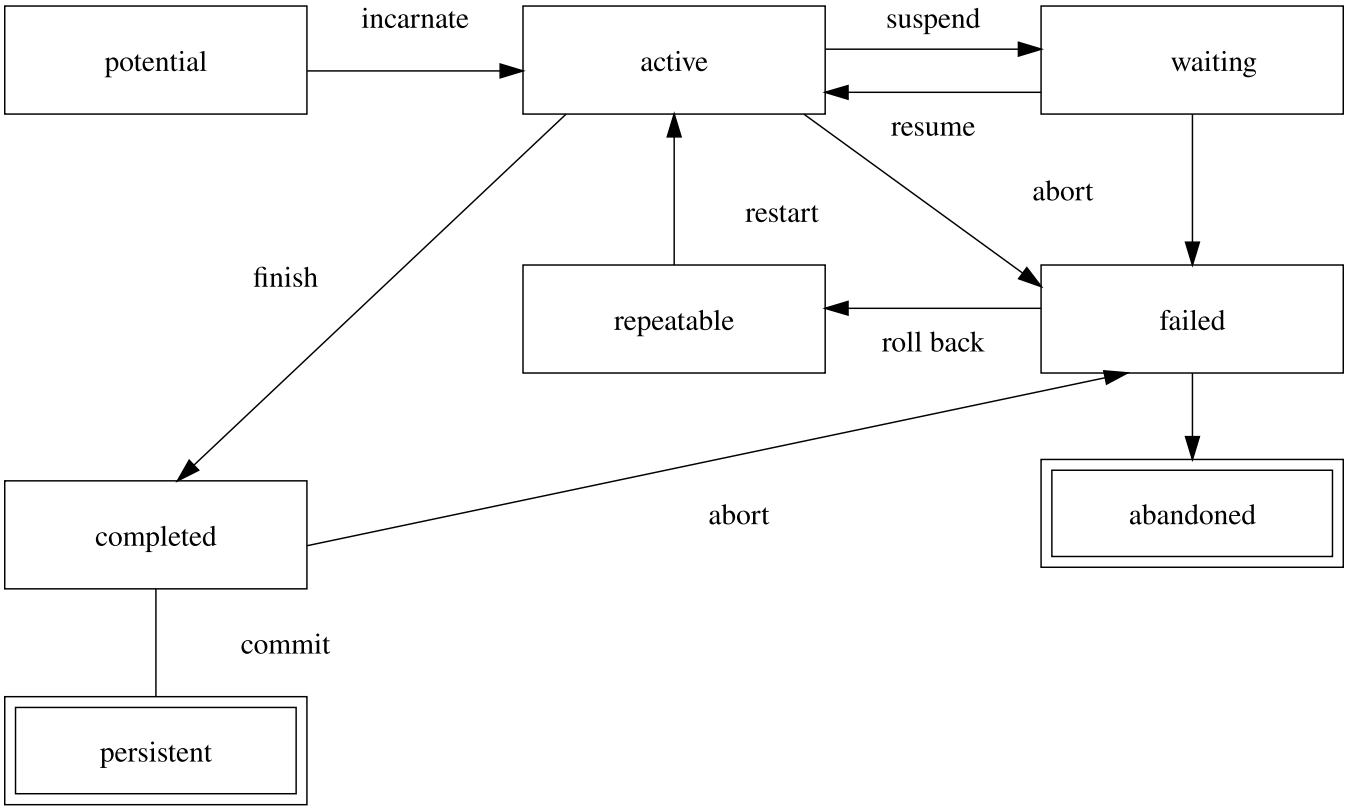
```
insert into Lectures
    values (5275, 'Nuclear Physics', 3, 2141);

insert into Professors
    values (2141, 'Meitner', 'C4', 205);
```

commit work

- **commit** after the first **insert** could not be executed successfully, since referential integrity is violated at this point

State Transitions



Summary

- Transactions should have the ACID properties
- ACID:
  - Atomicity
  - Consistency
  - Isolation
  - Durability

Summary(2)

- Atomicity: “all or nothing”, i.e., either all operations of a transaction are executed or none
- Consistency: if a transaction starts from a consistent database state, the state is still consistent after the transaction terminates
- Isolation: concurrently executed transactions must not have side effects on each other
- Durability: all changes made persistent with commit must survive (even in case of a system crash)