Database Systems

Lecture

Transaction

Transaction

- Imagine going to ATM and transferring some amount to another account, e.g.
  - Paying utility bill
  - Making payment against an online purchase
- What behind-the-scenes steps are involved in the above process?
  1. Reading balance of your account
  2. Computing the new balance after deduction
  3. Writing new balance to your account
  4. Reading balance of the recipient
  5. Computing new balance for recipient
  6. Writing new balance to recipient’s account

Transaction

- You noticed three different types of operations being performed
  - Reading from database
  - Computing something
  - Writing to database
- While computing does not access database, reading and writing does!
  - Reading requires retrieval queries
  - Writing may consist of insert, update, or delete queries
    - Write operations always change the database state

Transaction

- The set of operations to perform a specific task on the database is called a transaction
- Transaction is a logical unit of database processing, requiring one or more access operations (reading, writing)
- The task of transferring some amount from one account to another account is one transaction

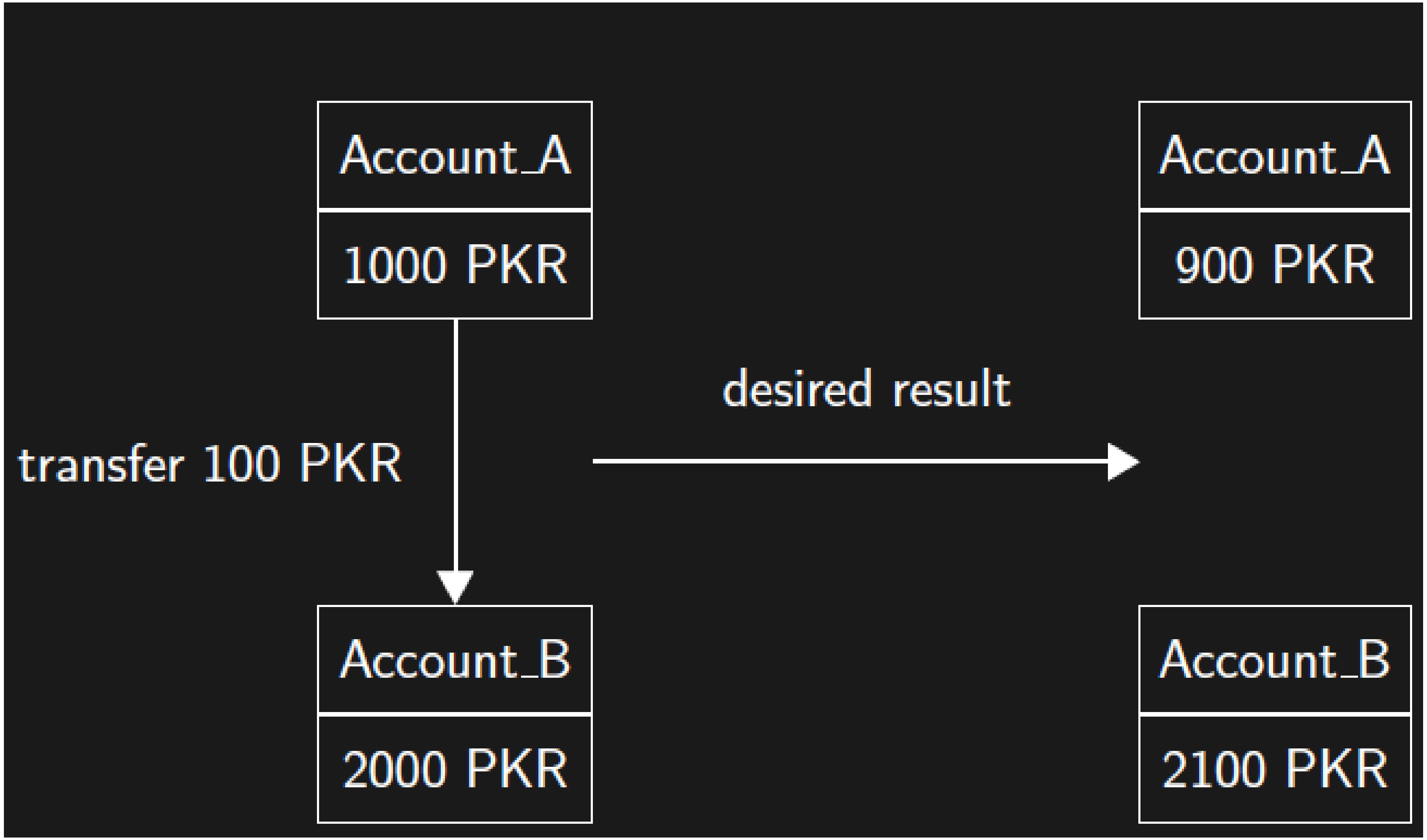
Transaction Properties

- Transactions must have certain properties:
- A C I D
  - **A**tomicity
  - **C**onsistency
  - **I**solation
  - **D**urability

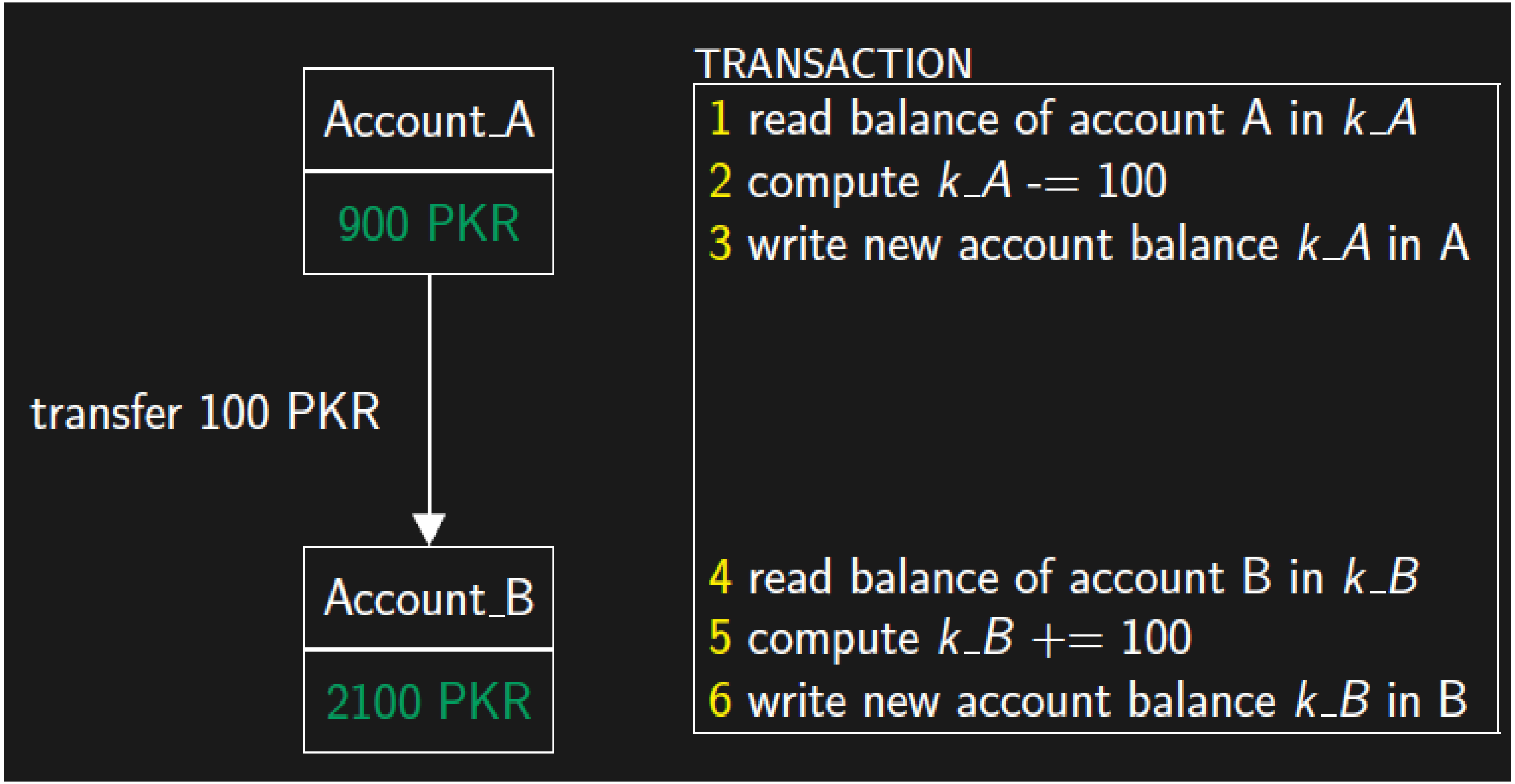
Transaction Properties - Atomicity

- Each transaction shall be ‘all or nothing’
- If one operation within a transaction fails, the entire transaction fails
- There must be only two possible results of executing a transaction
  - All actions are successfully performed and the transaction commits
  - One or more actions are unsuccessful – the transaction aborts
- For the end-user, a transaction is just one operation!
  - It is either successful or unsuccessful
- What will happen if a transaction does not have this property?
  - The database state will have undesired results!

Transaction Properties - Atomicity

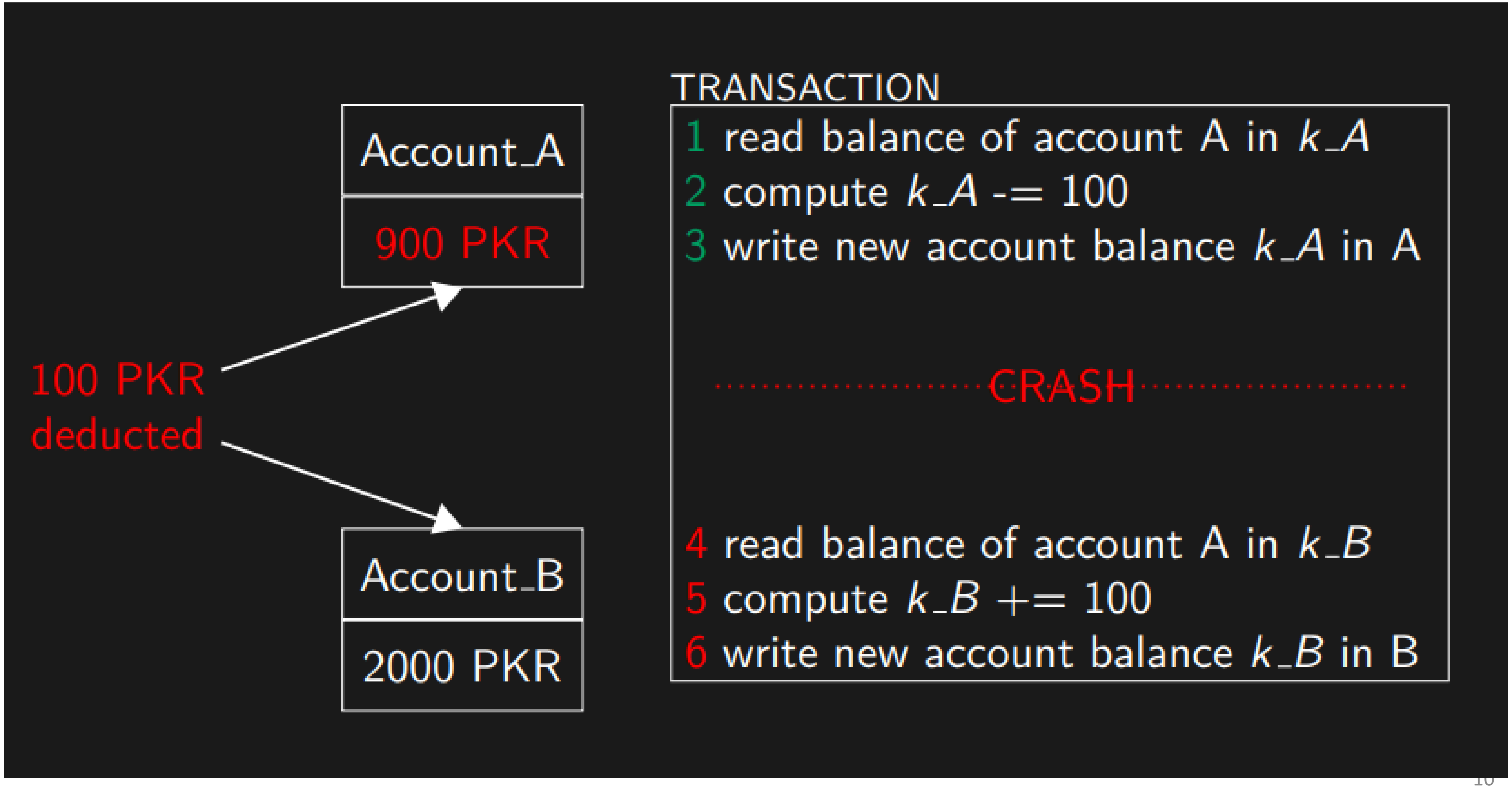


Transaction Properties - Atomicity



1 read balance of account A in k_A
2 compute k_A -= 100
3 write new account balance k_A in A
4 read balance of account B in k_B
5 compute k_B += 100
6 write new account balance k_B in B

Transaction Properties - Atomicity



TRANSACTION

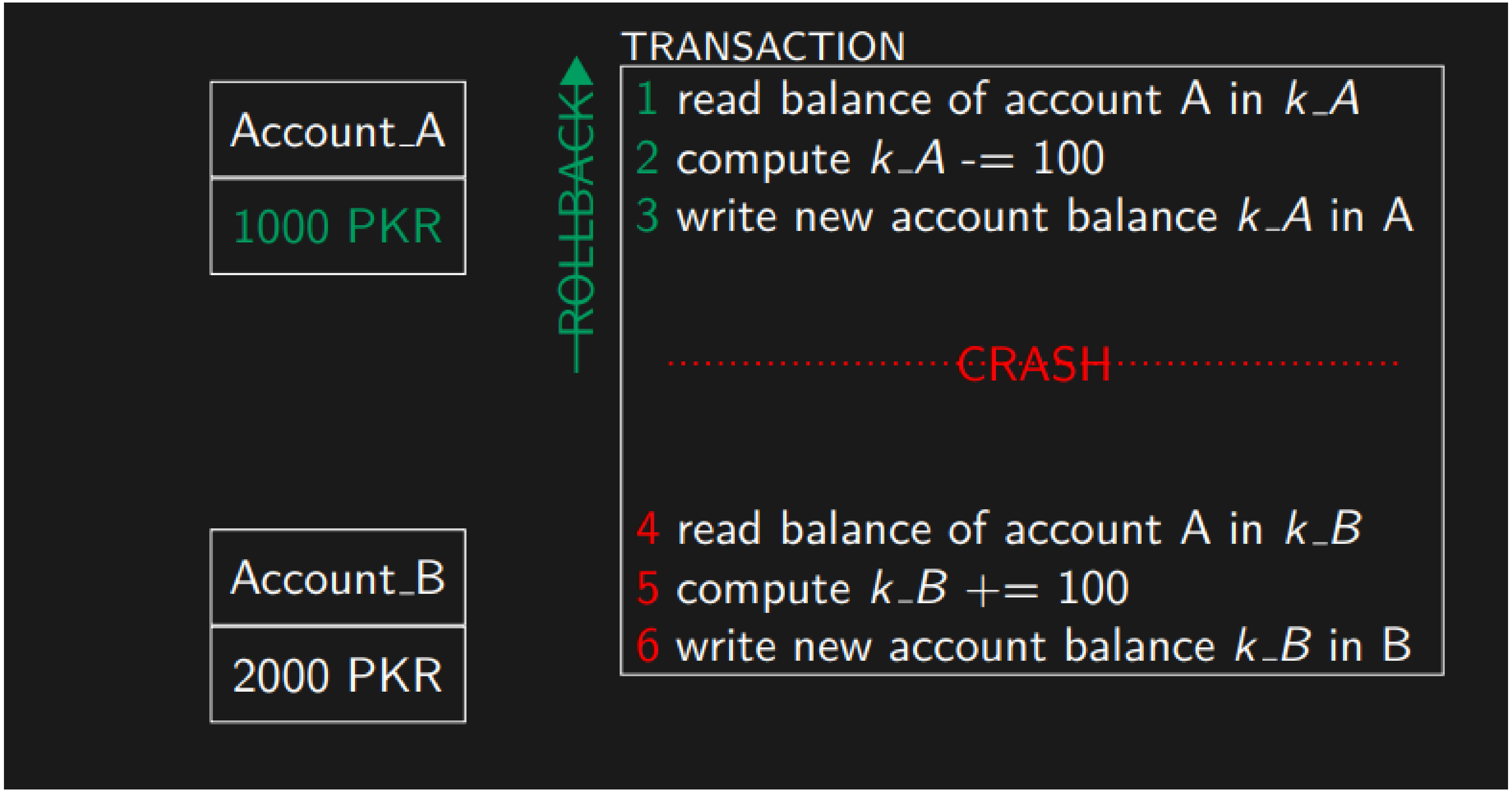
1 read balance of account A in k_A
2 compute k_A -= 100
3 write new account balance k_A in A

4 read balance of account A in k_B
5 compute k_B += 100
6 write new account balance k_B in B

Transaction Properties - Atomicity

- How do databases prevent such situation?
  - The rollback mechanism

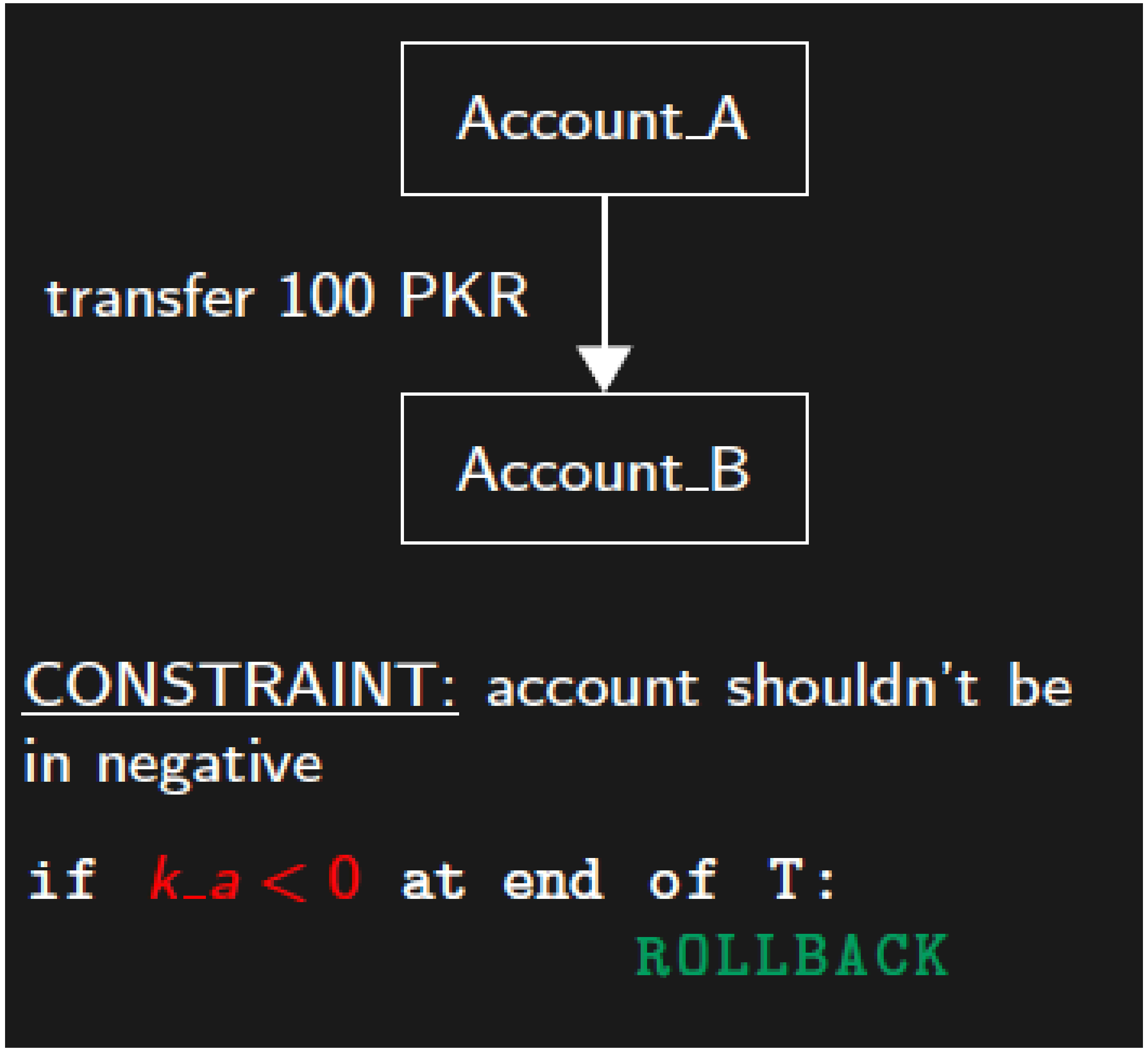
Transaction Properties - Atomicity


Account_A
1000 PKR
Account_B
2000 PKR
ROLLBACK
TRANSACTION
1 read balance of account A in k_A
2 compute k_A -= 100
3 write new account balance k_A in A
CRASH
4 read balance of account A in k_B
5 compute k_B += 100
6 write new account balance k_B in B

Transaction Properties - Consistency

- A transaction should bring the database from one valid state to another valid state
- What does valid state mean?
  - The database should not violate any of the constraints before and after the transaction is performed
- For example, a constraint on the example scenario could be: ‘account balance cannot be less than zero’
- In such a case, the transaction should rollback if the account of the sender has less than zero Rupees after the deduction has been made
  - Or, the application should check the balance before initiating the transaction

Transaction Properties - Consistency



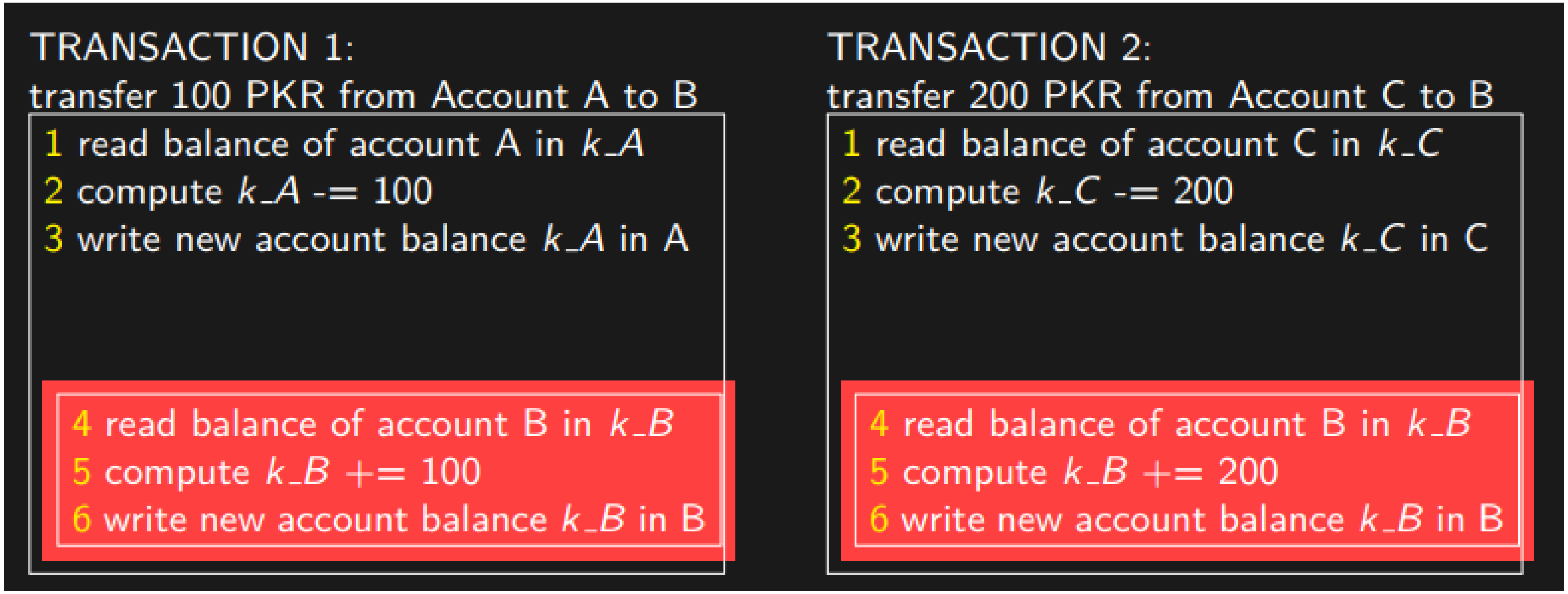
CONSTRAINT: account shouldn't be in negative

```
if $k\_a < 0$ at end of T:
        ROLLBACK
```

Transaction Properties - Isolation

- For understanding isolation, we must first understand the concept of concurrency
- The real databases are generally multi-user
  - Many users are accessing the same databases
  - Quite frequently, different users are accessing and modifying the same data…
  - Can you think of an example?
    - Booking online tickets for train just before Eid…
- In such a scenario, when multiple transactions are being executed concurrently, some issues can arise

Transaction Properties - Isolation



TRANSACTION 1:
transfer 100 PKR from Account A to B

```
1 read balance of account A in k_A
2 compute k_A -= 100
3 write new account balance k_A in A

4 read balance of account B in k_B
5 compute k_B += 100
6 write new account balance k_B in B
```

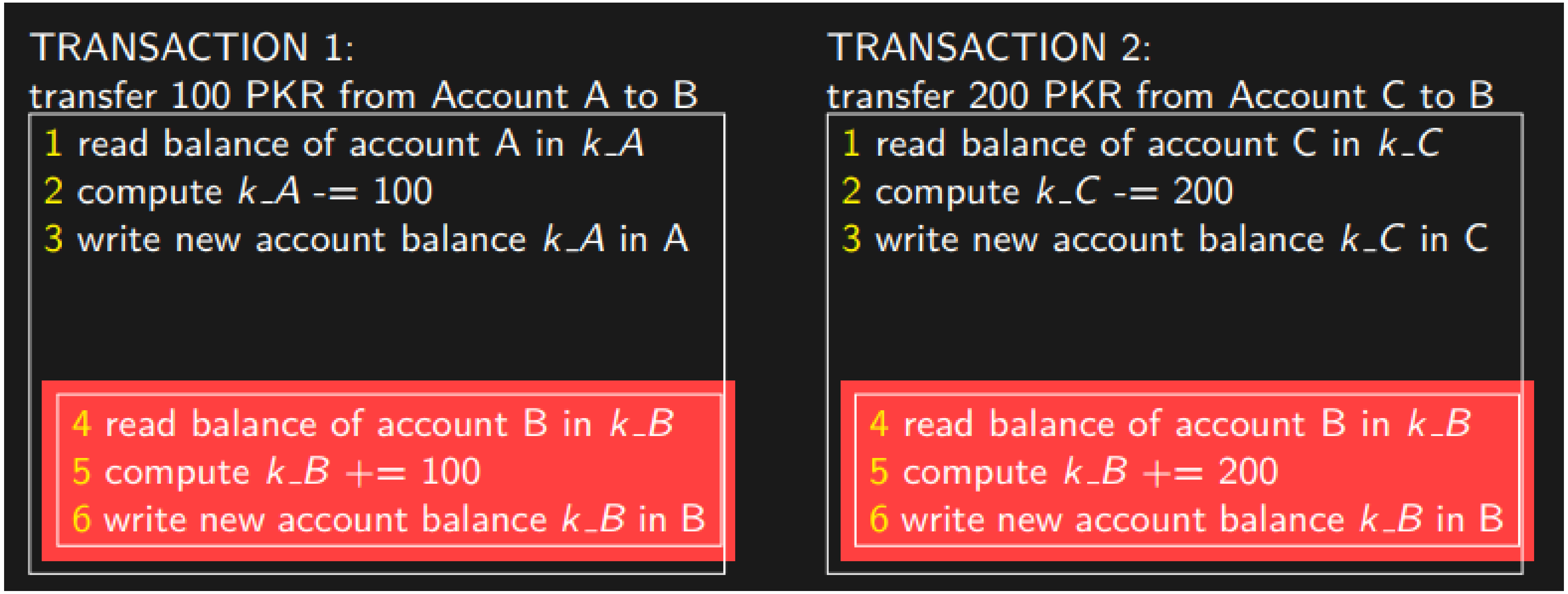
TRANSACTION 2:
transfer 200 PKR from Account C to B

```
1 read balance of account C in k_C
2 compute k_C -= 200
3 write new account balance k_C in C

4 read balance of account B in k_B
5 compute k_B += 200
6 write new account balance k_B in B
```

- At beginning of T1 and T2, balances are as follows:
  - A: 1000, B: 2000, C: 3000

Transaction Properties - Isolation



TRANSACTION 1:
transfer 100 PKR from Account A to B

1 read balance of account A in k_A
2 compute k_A -= 100
3 write new account balance k_A in A

4 read balance of account B in k_B
5 compute k_B += 100
6 write new account balance k_B in B

TRANSACTION 2:
transfer 200 PKR from Account C to B

1 read balance of account C in k_C
2 compute k_C -= 200
3 write new account balance k_C in C

4 read balance of account B in k_B
5 compute k_B += 200
6 write new account balance k_B in B

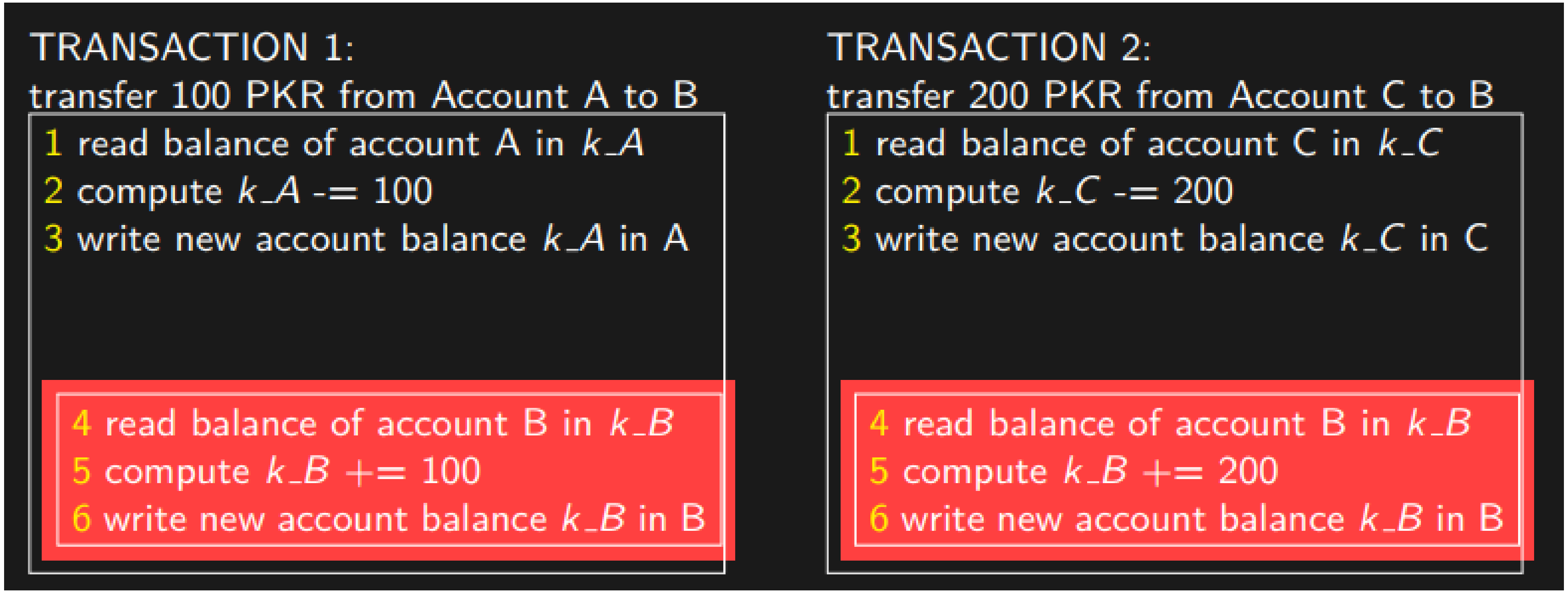
- Let’s assume that the transactions are executing concurrently. After both transactions complete, there are three possibilities of balances:

A: 900, B: 2100, C: 2800

A: 900, B: 2200, C: 2800

A: 900, B: 2300, C: 2800

Transaction Properties - Isolation



TRANSACTION 1:
transfer 100 PKR from Account A to B

```
1 read balance of account A in k_A
2 compute k_A -= 100
3 write new account balance k_A in A

4 read balance of account B in k_B
5 compute k_B += 100
6 write new account balance k_B in B
```

TRANSACTION 2:
transfer 200 PKR from Account C to B

```
1 read balance of account C in k_C
2 compute k_C -= 200
3 write new account balance k_C in C

4 read balance of account B in k_B
5 compute k_B += 200
6 write new account balance k_B in B
```

- Let's assume that the transactions are executing concurrently. After both transactions complete, there are three possibilities of balances:
- Which of these is correct?

A: 900, B: 2100, C: 2800 ✗

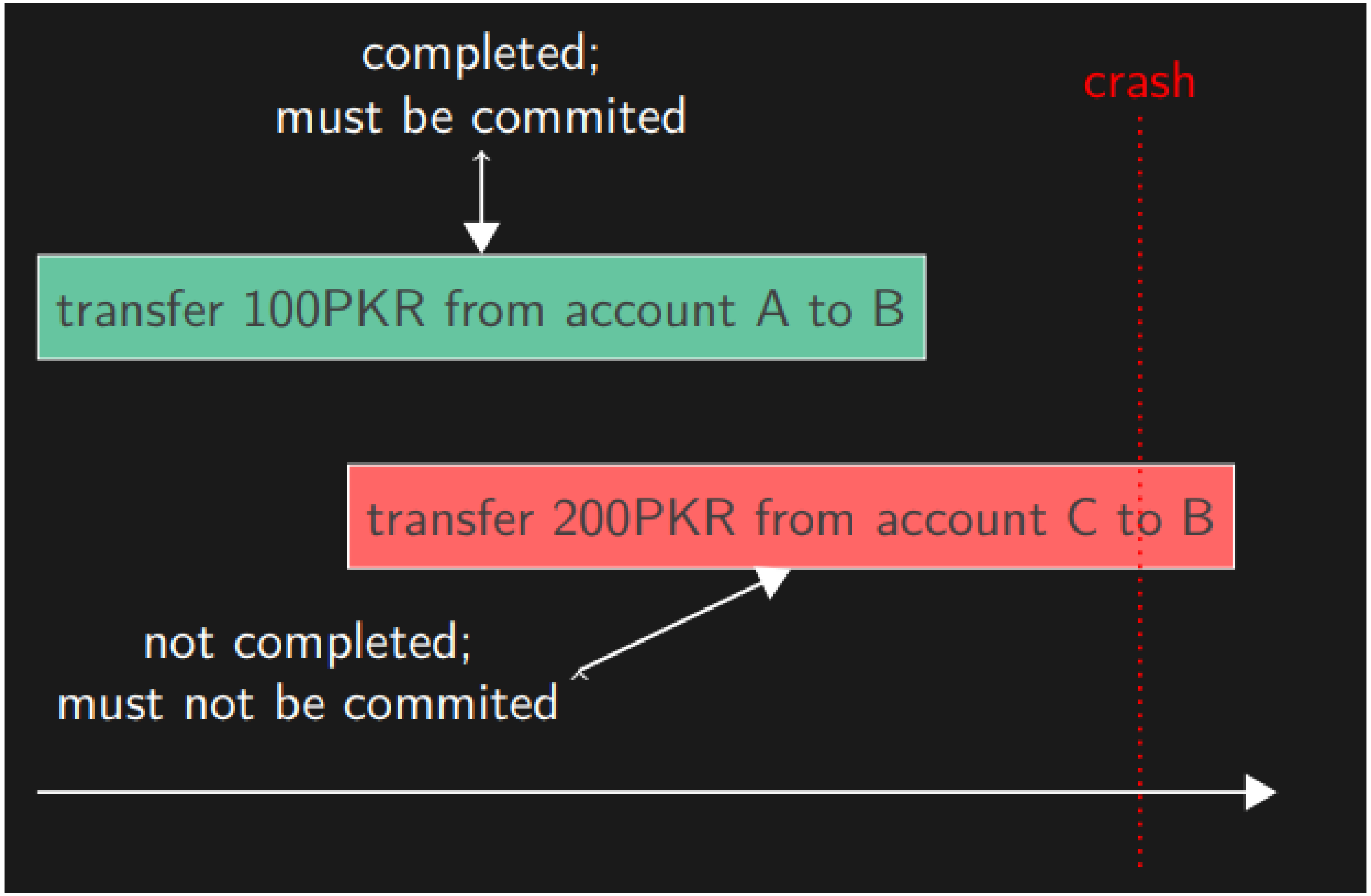
A: 900, B: 2200, C: 2800 ✗

A: 900, B: 2300, C: 2800 ✓

Transaction Properties - Durability

- A committed transaction should remain committed!
  - The changes applied to the database by a committed transaction must persist in the database
  - These changes must not be lost because of any failure such as power loss, system crash, network unavailability, or any such errors

Transaction Properties - Durability


completed;
must be commited
crash
transfer 100PKR from account A to B
transfer 200PKR from account C to B
not completed;
must not be commited

Stored Procedures

Stored Procedures

- Just like we did in programming languages, we can create a procedure for carrying out routine SQL commands
  - These are called stored procedures in SQL
- The stored procedures are database objects
- These can take input parameters, and return output parameters

Stored Procedures

```
Delimiter //

CREATE PROCEDURE GetStdsByDept( IN dept_name VARCHAR(50), OUT std_count INT )

BEGIN -- Calculate the number of students in the specified department

SELECT COUNT(*) INTO std_count  FROM students

WHERE department = dept_name;

END //

Delimiter ;

CALL GetStdsByDept(‘Cyber Security', @count);

SELECT @count;
```

Thanks a lot