Database Systems

In This Video

- Why do we use database systems?
- ACID

Why Use A Database System?

Why Use A Database System?

- Replicating DBMS functionality involves

Why Use A Database System?

- Replicating DBMS functionality involves
  - Algorithms for storing data to minimize space and facilitate searching

Why Use A Database System?

- Replicating DBMS functionality involves
  - Algorithms for storing data to minimize space and facilitate searching
  - Algorithms to efficiently search, scan, filter, and/or aggregate data

Why Use A Database System?

- Replicating DBMS functionality involves
  - Algorithms for storing data to minimize space and facilitate searching
  - Algorithms to efficiently search, scan, filter, and/or aggregate data
  - Algorithms to automatically validate new data

Why Use A Database System?

- Replicating DBMS functionality involves
  - Algorithms for storing data to minimize space and facilitate searching
  - Algorithms to efficiently search, scan, filter, and/or aggregate data
  - Algorithms to automatically validate new data
  - Algorithms to change the shape of data

Why Use A Database System?

- Replicating DBMS functionality involves
  - Algorithms for storing data to minimize space and facilitate searching
  - Algorithms to efficiently search, scan, filter, and/or aggregate data
  - Algorithms to automatically validate new data
  - Algorithms to change the shape of data
  - Algorithms to scale data from GBs to TBs to PBs across dozens of machines

Why Use A Database System?

- Replicating DBMS functionality involves
  - Algorithms for storing data to minimize space and facilitate searching
  - Algorithms to efficiently search, scan, filter, and/or aggregate data
  - Algorithms to automatically validate new data
  - Algorithms to change the shape of data
  - Algorithms to scale data from GBs to TBs to PBs across dozens of machines

- And this is before we even talk about the real hard part!

ACID

- **A**tomicity
- **C**onsistency
- **I**solation
- **D**urability

Atomicity

```
a = BankAccount("Alice", balance=100)
b = BankAccount("Bob", balance=0)

# transfer money from Alice to Bob
a.withdraw(100)
b.deposit(100)
```

Atomicity

```
a = BankAccount("Alice", balance=100)
b = BankAccount("Bob", balance=0)

# transfer money from Alice to Bob
a.withdraw(100)
b.deposit(100)
```

Alice	Bob
100	0

Atomicity

```
a = BankAccount("Alice", balance=100)
b = BankAccount("Bob", balance=0)

# transfer money from Alice to Bob
a.withdraw(100)
b.deposit(100)
```

Alice	Bob
100	0

Atomicity

```
a = BankAccount("Alice", balance=100)
b = BankAccount("Bob", balance=0)

# transfer money from Alice to Bob
a.withdraw(100)
b.deposit(100)
```

Alice	Bob
0	0

Atomicity

```
a = BankAccount("Alice", balance=100)
b = BankAccount("Bob", balance=0)

# transfer money from Alice to Bob
a.withdraw(100)
b.deposit(100)
```

Alice	Bob
0	0

Atomicity

```
a = BankAccount("Alice", balance=100)
b = BankAccount("Bob", balance=0)

# transfer money from Alice to Bob
a.withdraw(100)
b.deposit(100)
```

Alice	Bob
0	0

Atomicity

```
a = BankAccount("Alice", balance=100)
b = BankAccount("Bob", balance=0)

# transfer money from Alice to Bob
a.withdraw(100)
b.deposit(100)
```

Alice	Bob
0	0

Atomicity

```
a = BankAccount("Alice", balance=100)
b = BankAccount("Bob", balance=0)

# transfer money from Alice to Bob
a.withdraw(100)
b.deposit(100)
```

wrap in a transaction

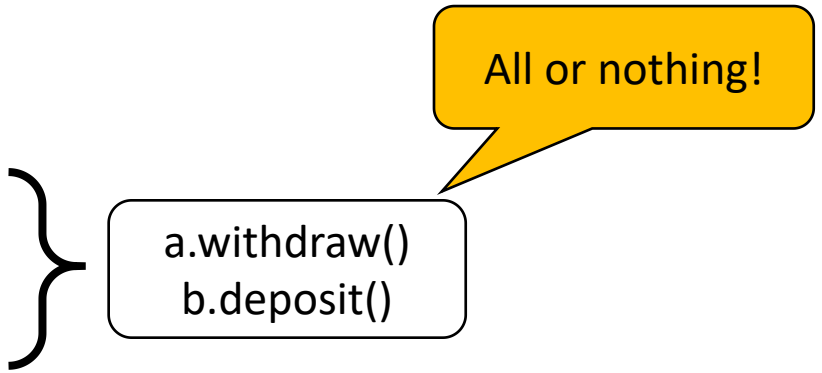
Atomicity

```
a = BankAccount("Alice", balance=100)
b = BankAccount("Bob", balance=0)

# transfer money from Alice to Bob
a.withdraw(100)
b.deposit(100)
```

a.withdraw()
b.deposit()

Atomicity

```
a = BankAccount("Alice", balance=100)
b = BankAccount("Bob", balance=0)

# transfer money from Alice to Bob
a.withdraw(100)
b.deposit(100)
```



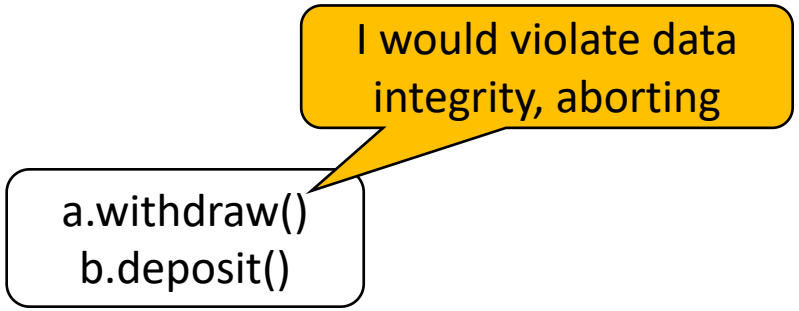
Consistency

```
a = BankAccount("Alice", balance=0)
b = BankAccount("Bob", balance=0)

# transfer money from Alice to Bob
a.withdraw(100)
b.deposit(100)
```

a.withdraw()
b.deposit()

Consistency

```
a = BankAccount("Alice", balance=0)
b = BankAccount("Bob", balance=0)

# transfer money from Alice to Bob
a.withdraw(100)
b.deposit(100)
```

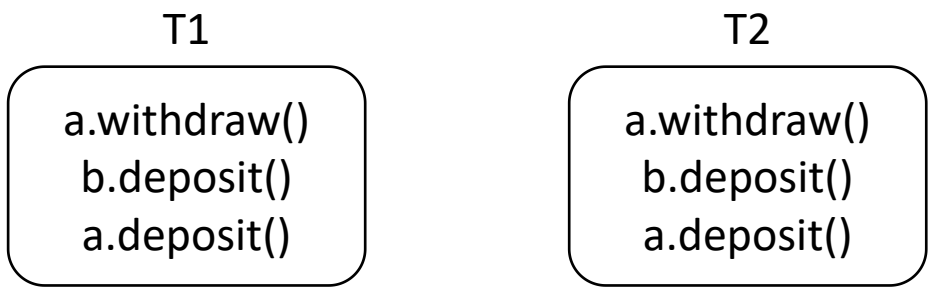


Isolation

```
a = BankAccount("Alice", balance=100)
b = BankAccount("Bob", balance=0)

# transfer money from Alice to Bob
a.withdraw(100)
b.deposit(100)
a.deposit(100) # Alice wins a prize
```

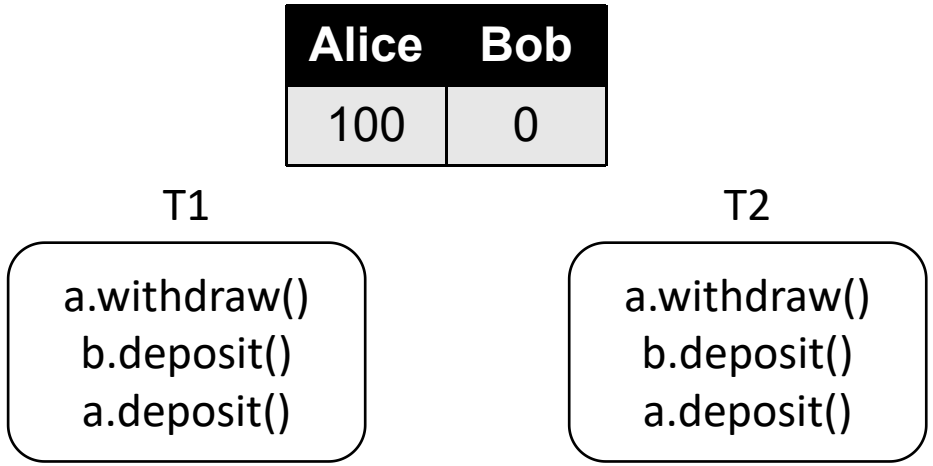
Isolation

```
a = BankAccount("Alice", balance=100)
b = BankAccount("Bob", balance=0)

# transfer money from Alice to Bob
a.withdraw(100)
b.deposit(100)
a.deposit(100) # Alice wins a prize
```



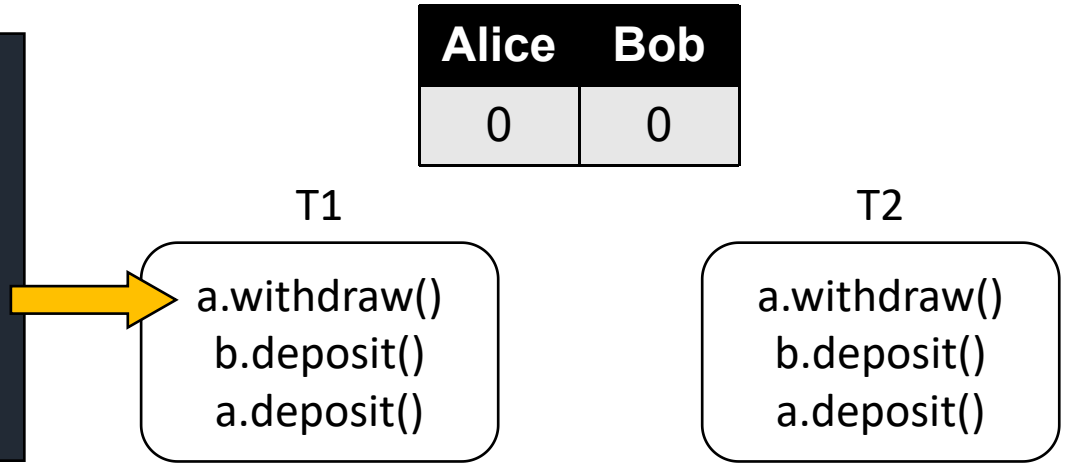
Isolation

```
a = BankAccount("Alice", balance=100)
b = BankAccount("Bob", balance=0)

# transfer money from Alice to Bob
a.withdraw(100)
b.deposit(100)
a.deposit(100) # Alice wins a prize
```



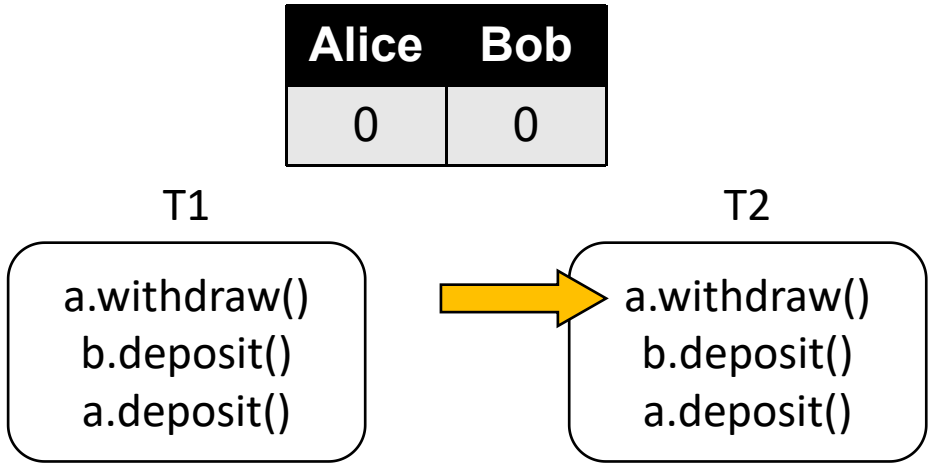
Isolation

```
a = BankAccount("Alice", balance=100)
b = BankAccount("Bob", balance=0)

# transfer money from Alice to Bob
a.withdraw(100)
b.deposit(100)
a.deposit(100) # Alice wins a prize
```



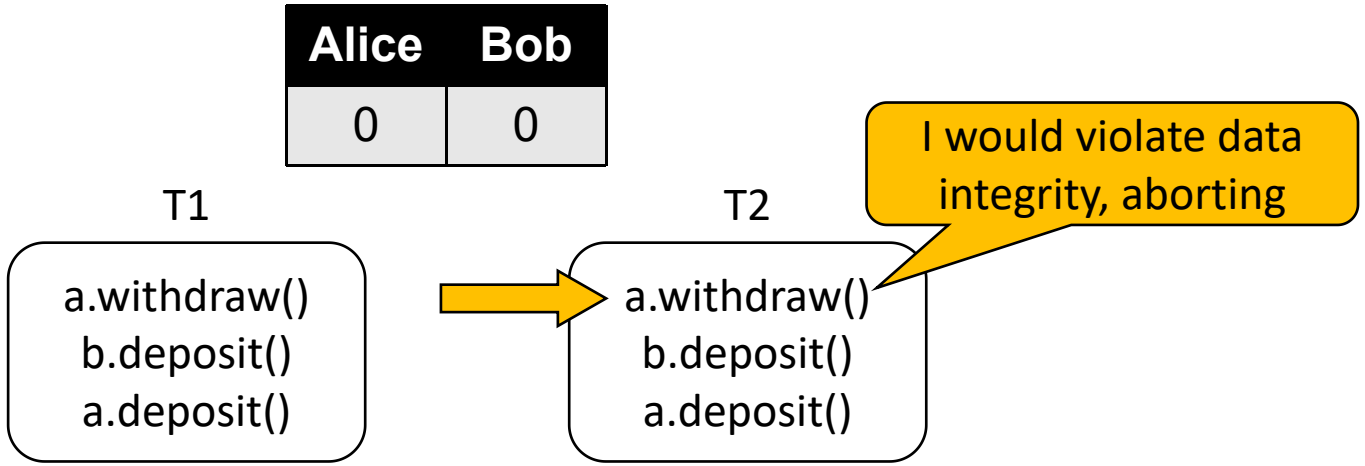
Isolation

```
a = BankAccount("Alice", balance=100)
b = BankAccount("Bob", balance=0)

# transfer money from Alice to Bob
a.withdraw(100)
b.deposit(100)
a.deposit(100) # Alice wins a prize
```



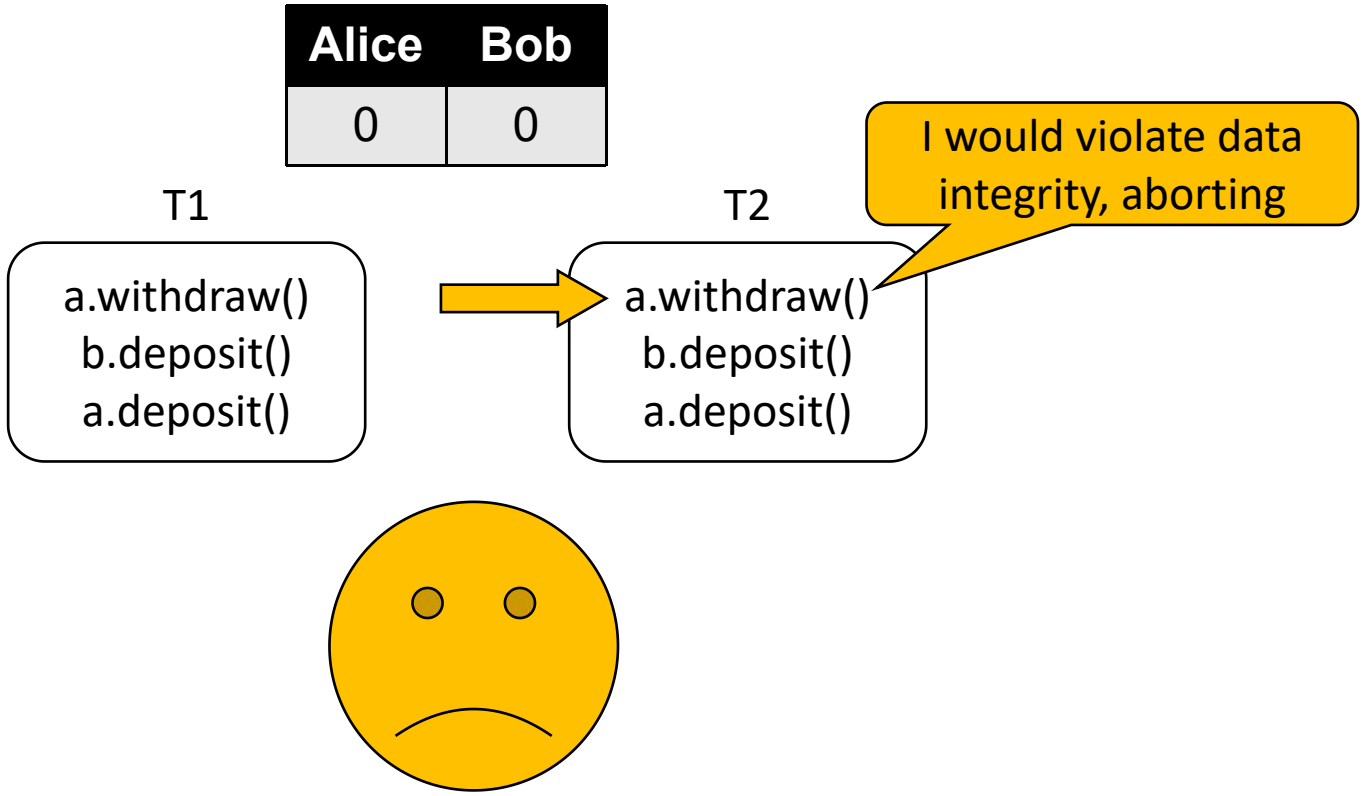
Isolation

```
a = BankAccount("Alice", balance=100)
b = BankAccount("Bob", balance=0)

# transfer money from Alice to Bob
a.withdraw(100)
b.deposit(100)
a.deposit(100) # Alice wins a prize
```



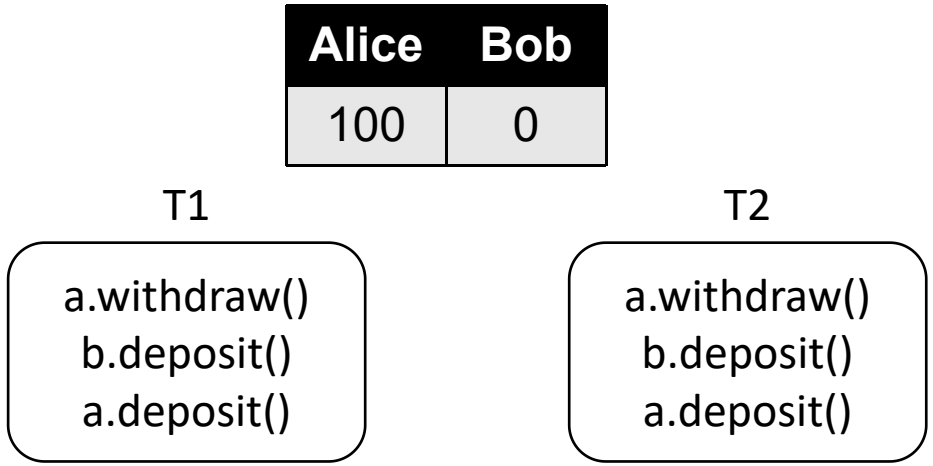
Isolation

```
a = BankAccount("Alice", balance=100)
b = BankAccount("Bob", balance=0)

# transfer money from Alice to Bob
a.withdraw(100)
b.deposit(100)
a.deposit(100) # Alice wins a prize
```



Isolation

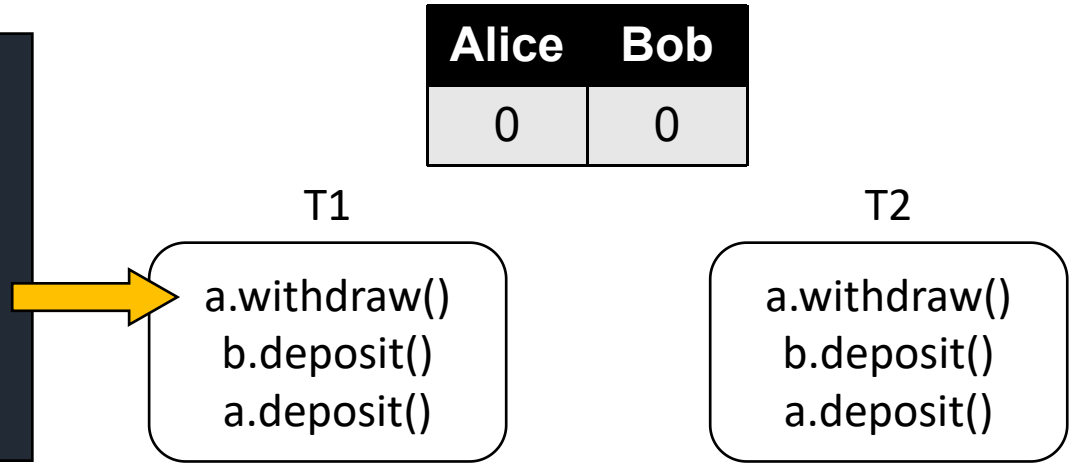
```
a = BankAccount("Alice", balance=100)
b = BankAccount("Bob", balance=0)

# transfer money from Alice to Bob
a.withdraw(100)
b.deposit(100)
a.deposit(100) # Alice wins a prize
```



Isolation can guarantee T1 and T2 are NOT concurrent

Isolation

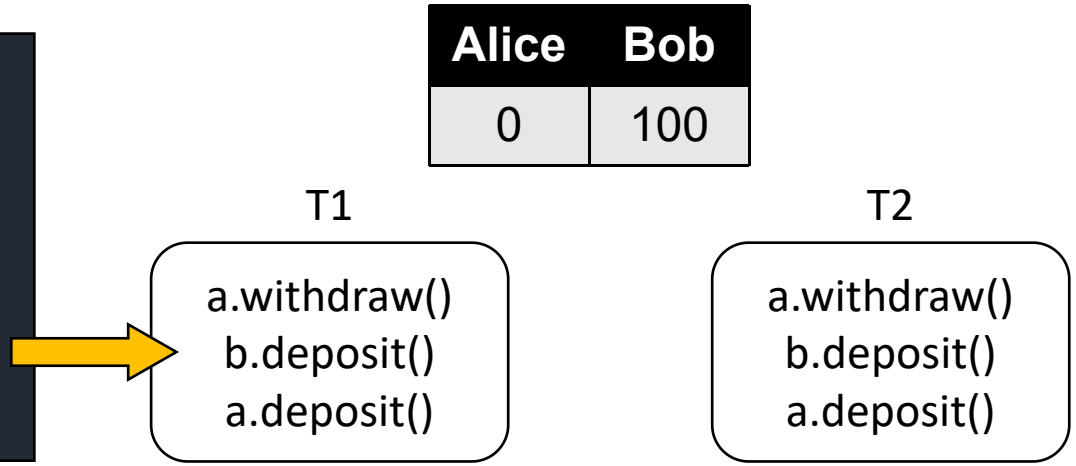
```
a = BankAccount("Alice", balance=100)
b = BankAccount("Bob", balance=0)

# transfer money from Alice to Bob
a.withdraw(100)
b.deposit(100)
a.deposit(100) # Alice wins a prize
```



Isolation can guarantee T1 and T2 are NOT concurrent

Isolation

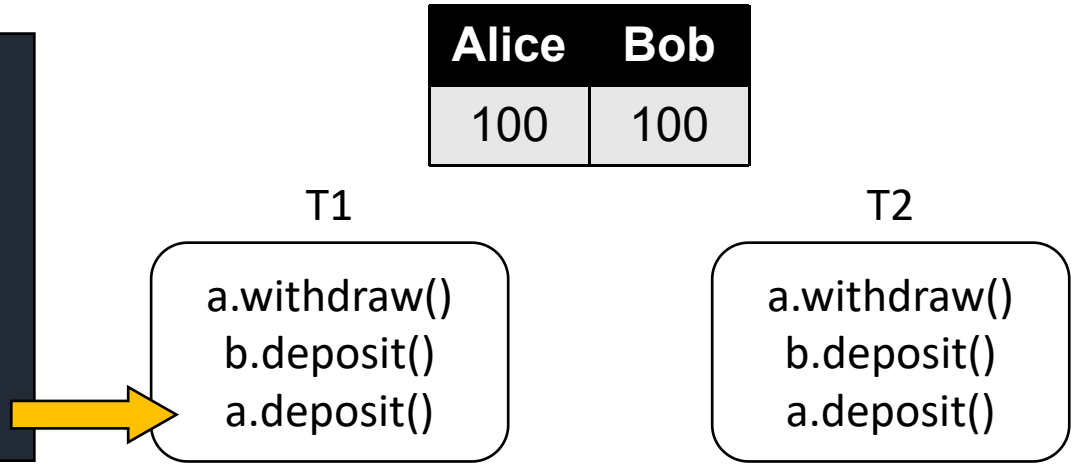
```
a = BankAccount("Alice", balance=100)
b = BankAccount("Bob", balance=0)

# transfer money from Alice to Bob
a.withdraw(100)
b.deposit(100)
a.deposit(100) # Alice wins a prize
```



Isolation can guarantee T1 and T2 are NOT concurrent

Isolation

```
a = BankAccount("Alice", balance=100)
b = BankAccount("Bob", balance=0)

# transfer money from Alice to Bob
a.withdraw(100)
b.deposit(100)
a.deposit(100) # Alice wins a prize
```

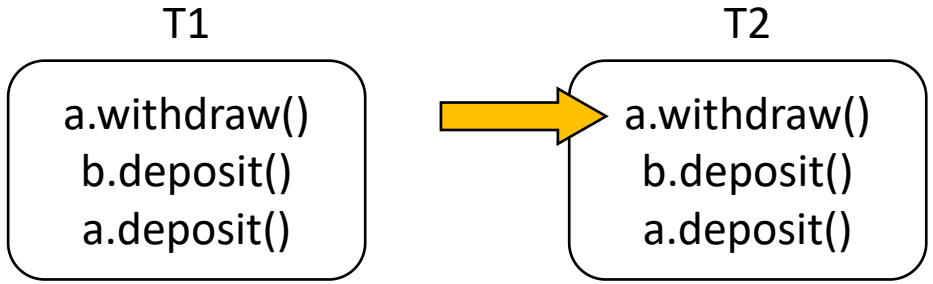


Isolation can guarantee T1 and T2 are NOT concurrent

Isolation

```
a = BankAccount("Alice", balance=100)
b = BankAccount("Bob", balance=0)

# transfer money from Alice to Bob
a.withdraw(100)
b.deposit(100)
a.deposit(100) # Alice wins a prize
```

Alice	Bob
0	100

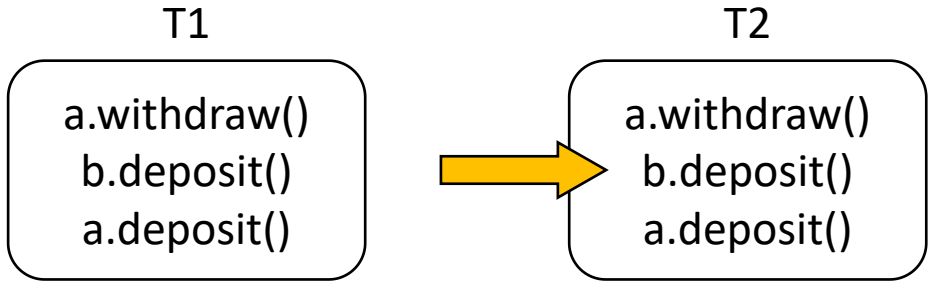


Isolation can guarantee T1 and T2 are NOT concurrent

Isolation

```
a = BankAccount("Alice", balance=100)
b = BankAccount("Bob", balance=0)

# transfer money from Alice to Bob
a.withdraw(100)
b.deposit(100)
a.deposit(100) # Alice wins a prize
```

Alice	Bob
0	200



Isolation can guarantee T1 and T2 are NOT concurrent

Isolation

```
a = BankAccount("Alice", balance=100)
b = BankAccount("Bob", balance=0)

# transfer money from Alice to Bob
a.withdraw(100)
b.deposit(100)
a.deposit(100) # Alice wins a prize
```

Alice	Bob
100	200

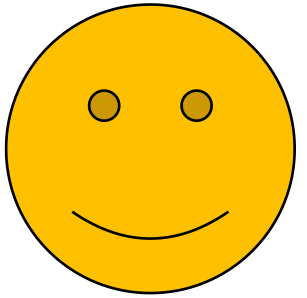
T1

```
a.withdraw()
b.deposit()
a.deposit()
```

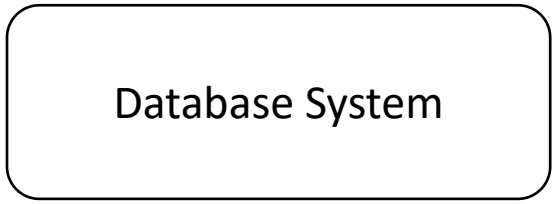
T2

```
a.withdraw()
b.deposit()
a.deposit()
```

Isolation can guarantee T1 and T2 are NOT concurrent



Durability



Durability

Database System

```
a = BankAccount("Alice", balance=100)
b = BankAccount("Bob", balance=0)
```

Durability

Database System

Alice	Bob
100	0

```
a = BankAccount("Alice", balance=100)
b = BankAccount("Bob", balance=0)
```

Durability

Database System

Alice	Bob
100	0

T1

a.withdraw()
b.deposit()
a.deposit()

Durability

Database System

Alice	Bob
100	100

T1

a.withdraw()
b.deposit()
a.deposit()

Durability

Database System

Alice	Bob
100	100

T2

a.withdraw()
b.deposit()
a.deposit()

Durability

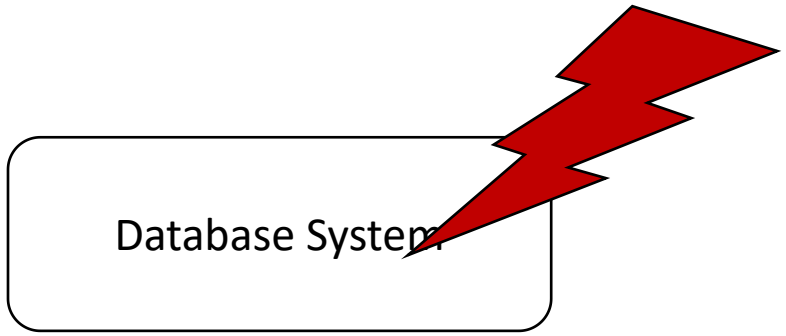
Database System

Alice	Bob
100	200

T2

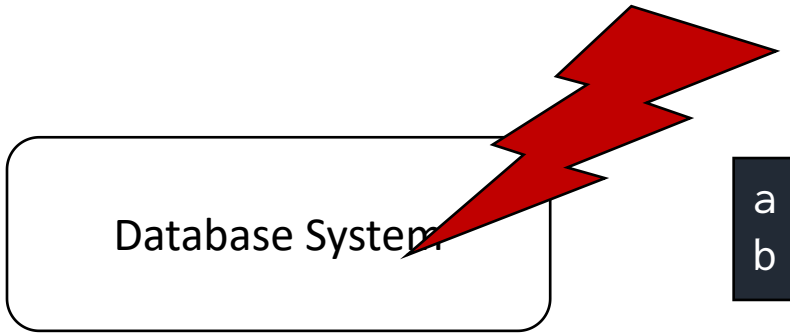
a.withdraw()
b.deposit()
a.deposit()

Durability



Alice	Bob
100	200

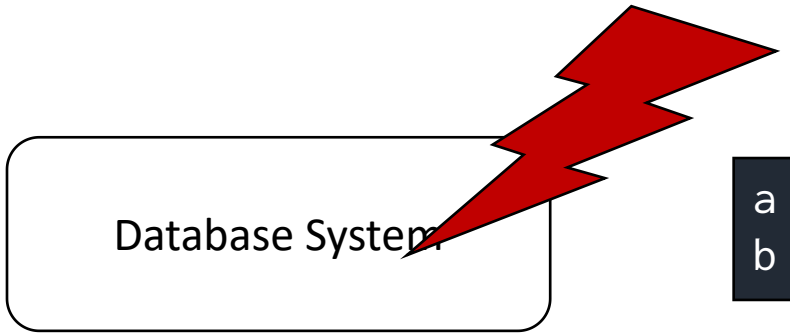
Durability




```
a = BankAccount("Alice", balance=100)
b = BankAccount("Bob", balance=0)
```

Alice	Bob
100	0

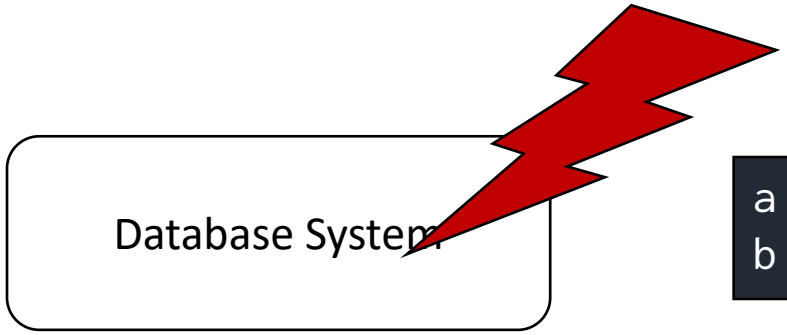
Durability




```
a = BankAccount("Alice", balance=100)
b = BankAccount("Bob", balance=0)
```

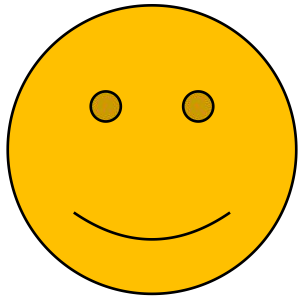
Alice	Bob
100	0

Durability




```
a = restore("Alice")
b = restore("Bob")
```

Alice	Bob
100	200



Why Use A Database System?

- Using an existing database system saves monumental effort
- Reasonably efficient
- Reasonably safe (ACID)
- Reasonably easy to use (SQL)