OOPSLA 2019

OCT 25, 2019

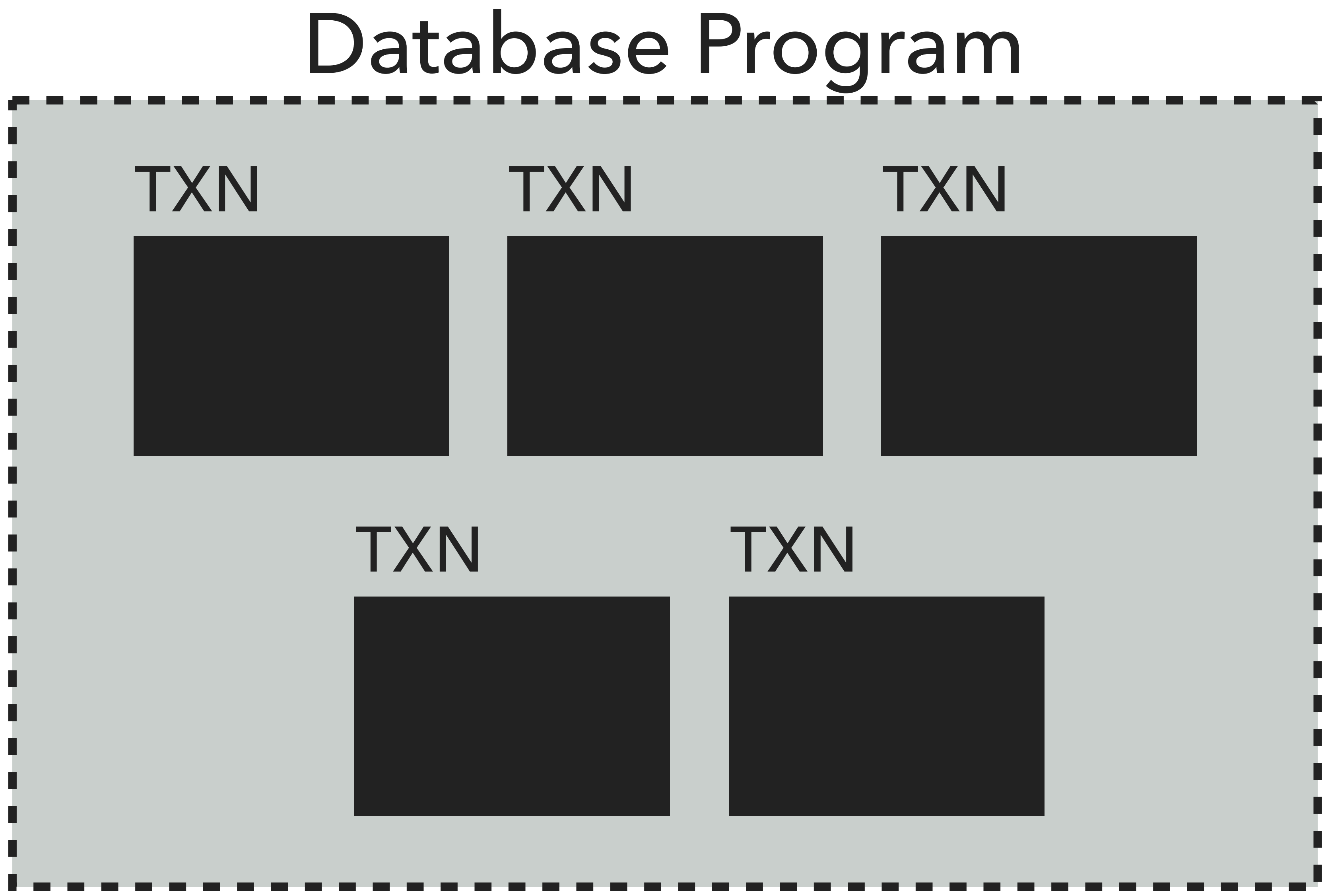
# CLOTHO: DIRECTED TEST GENERATION FOR WEAKLY CONSISTENT DATABASE SYSTEMS

Kia Rahmani
Kartik Nagar
Benjamin Delaware
**Suresh Jagannathan**

Purdue University
PURPL

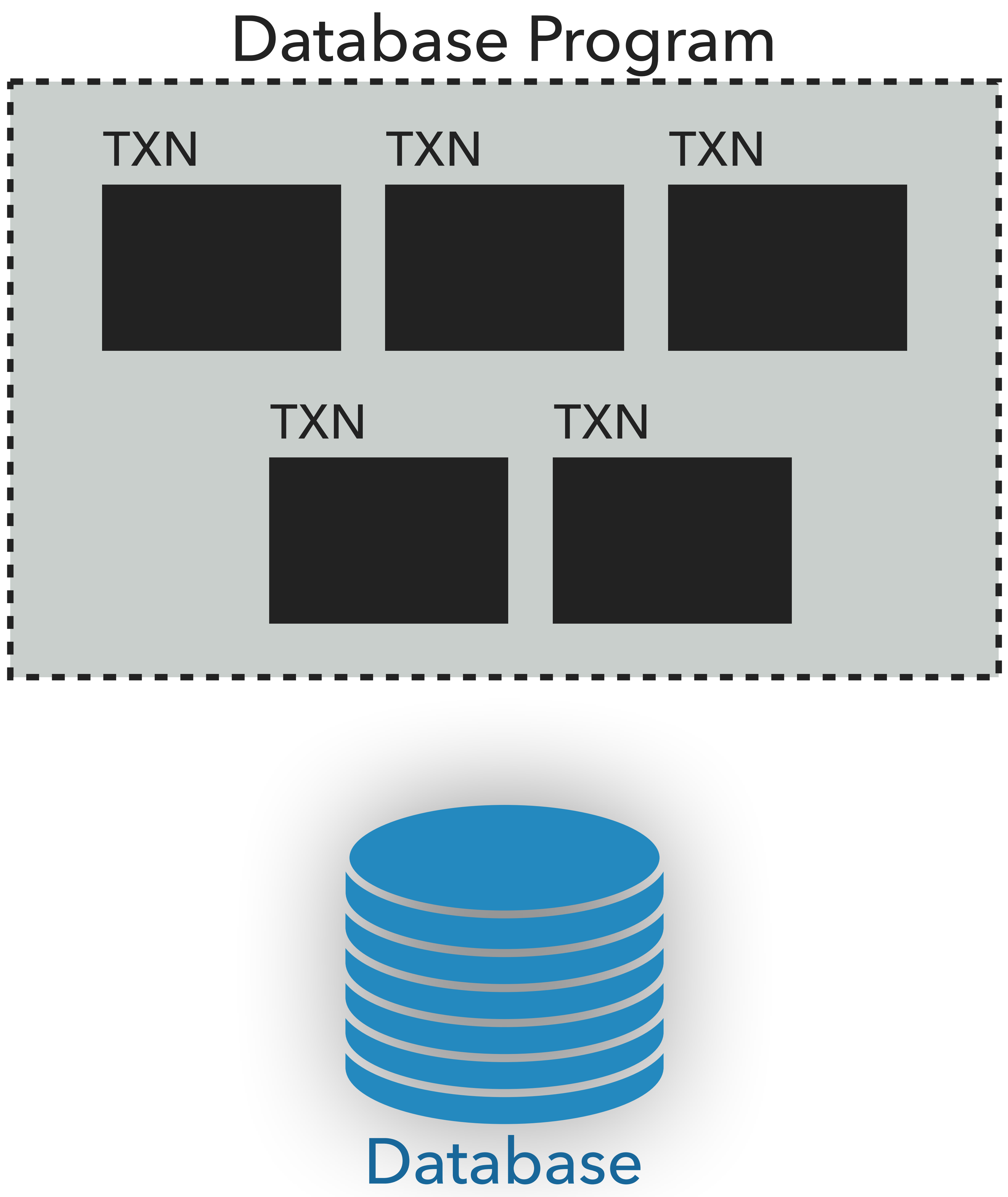
# TRADITIONAL DATABASE PROGRAMMING

- Transactional support



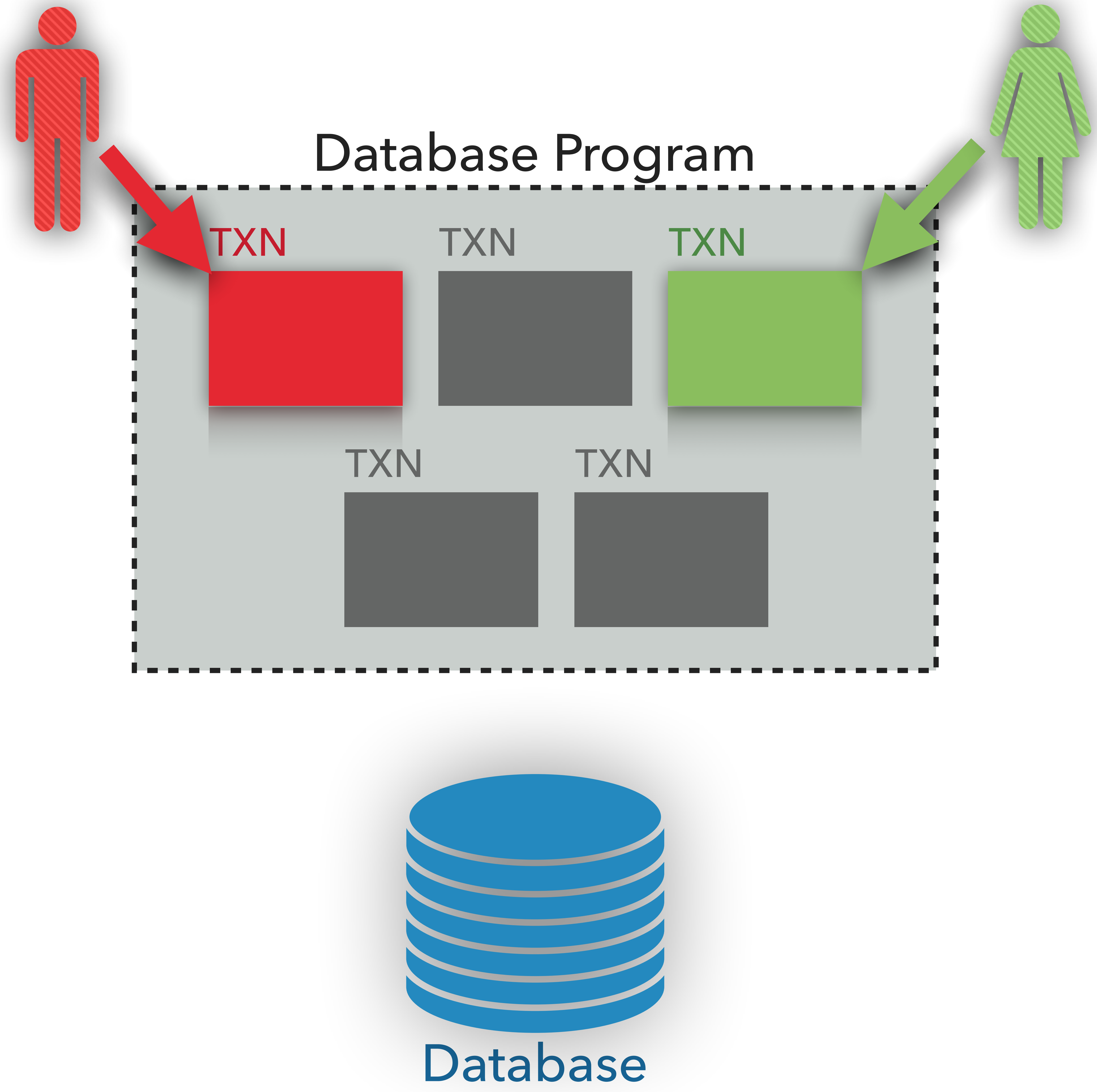
# TRADITIONAL DATABASE PROGRAMMING

- Transactional support
- Highly structured relational data



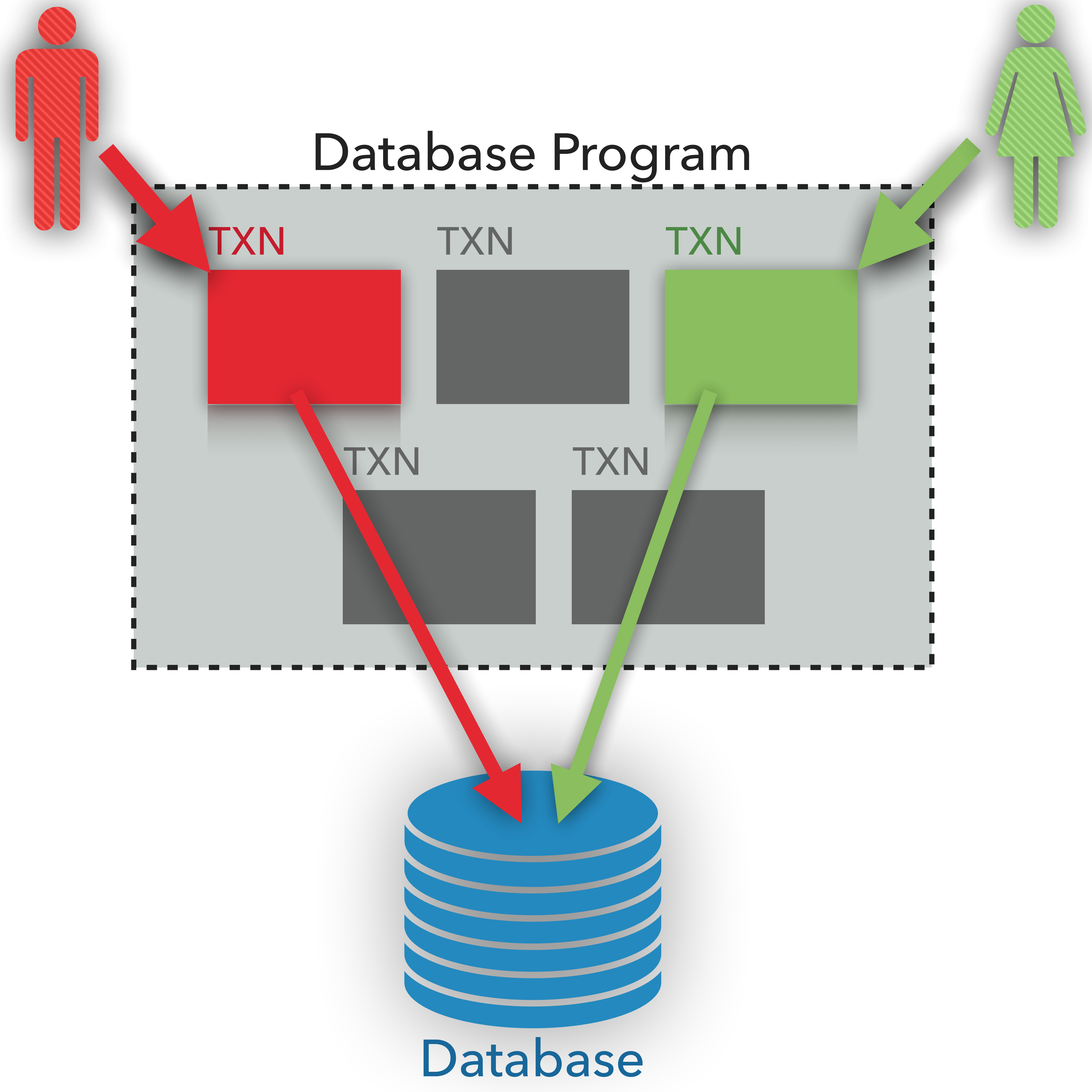
# TRADITIONAL DATABASE PROGRAMMING

- Transactional support
- Highly structured relational data
- Clients invoke transactions



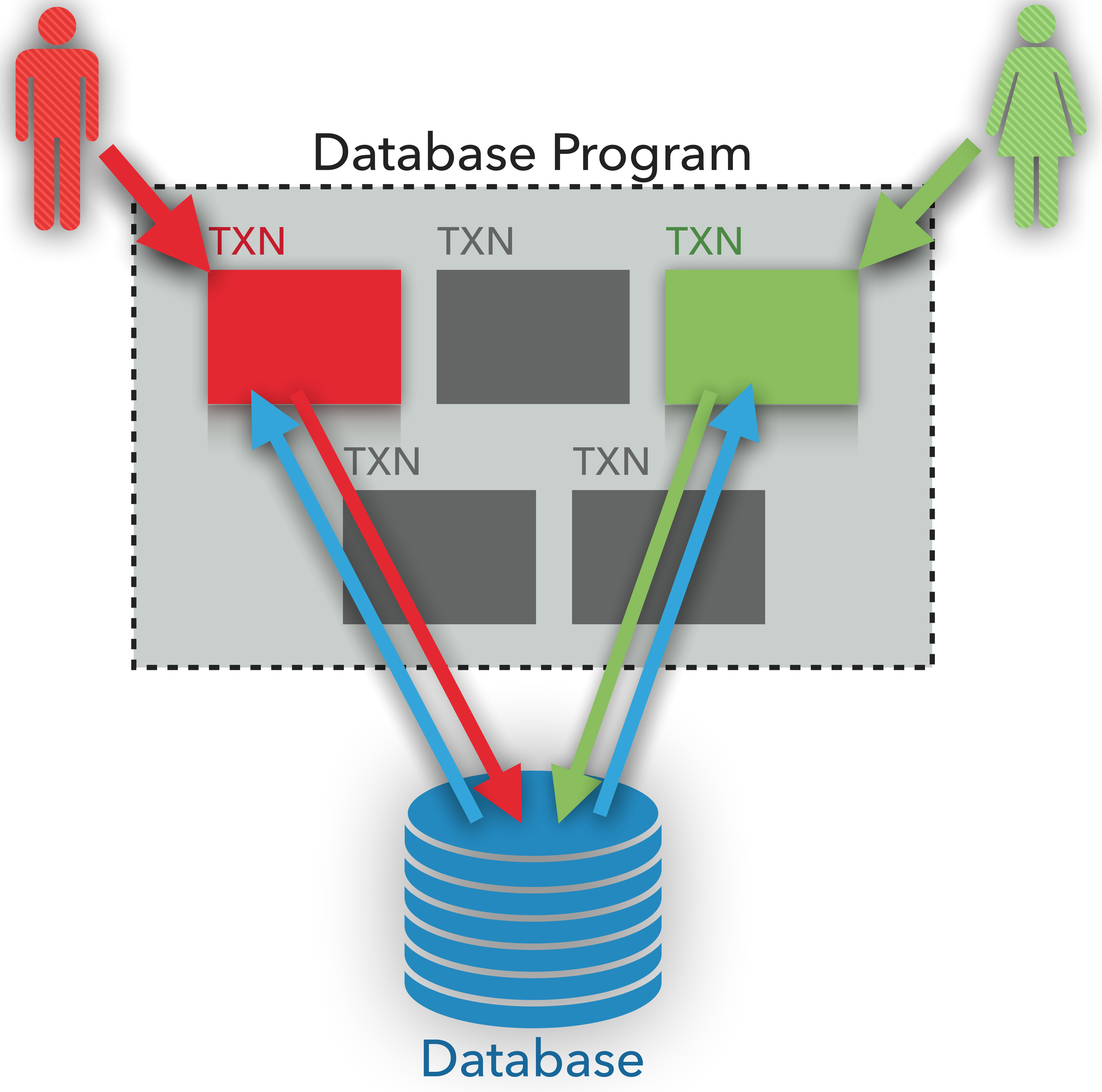
# TRADITIONAL DATABASE PROGRAMMING

- Transactional support
- Highly structured relational data
- Clients invoke transactions
- Structured query language for data retrieval/modification



# TRADITIONAL DATABASE PROGRAMMING

- Transactional support
- Highly structured relational data
- Clients invoke transactions
- Structured query language for data retrieval/modification
- Queries processed and responded by the DBMS

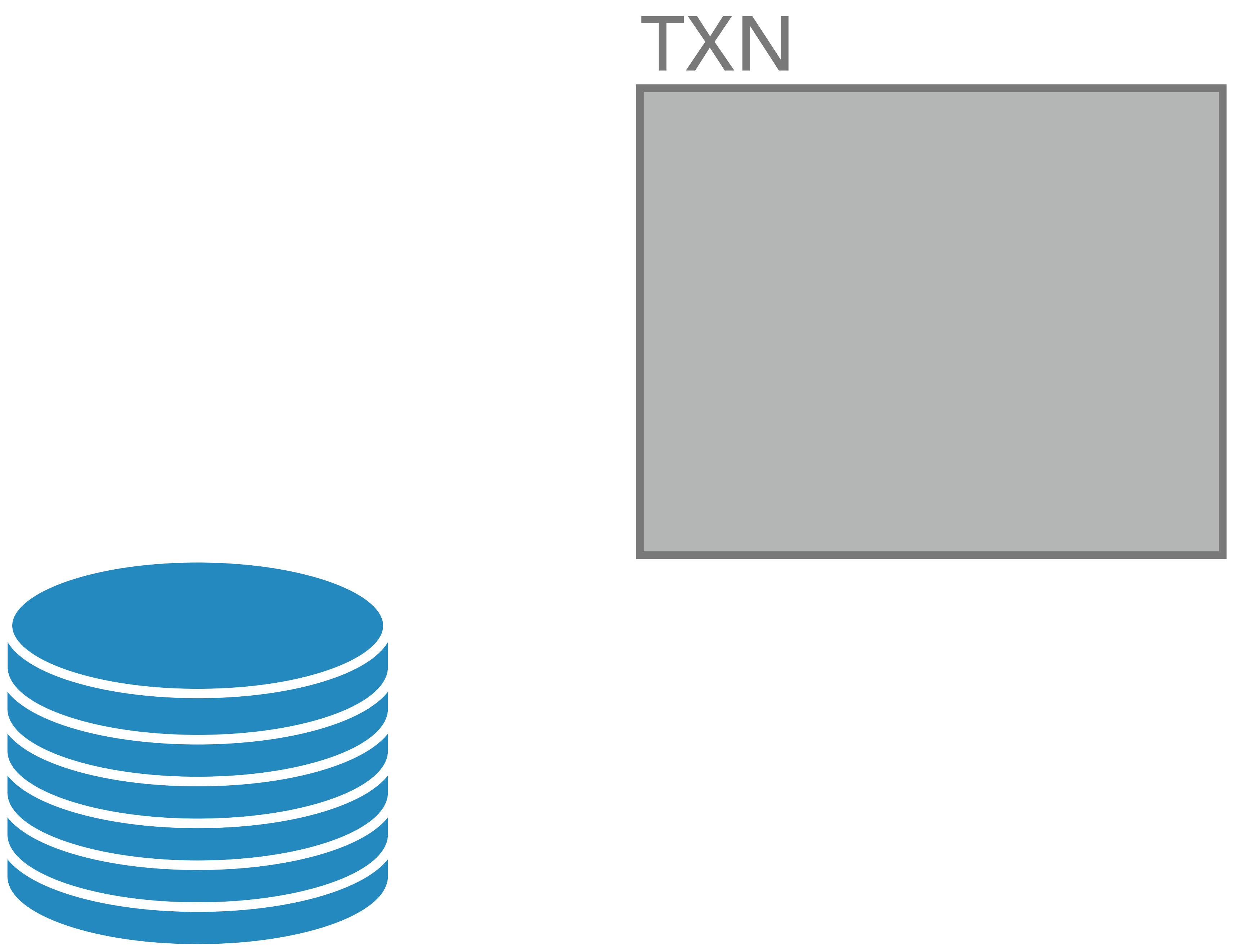


## TRADITIONAL DATABASE PROGRAMMING

- ACID guarantees

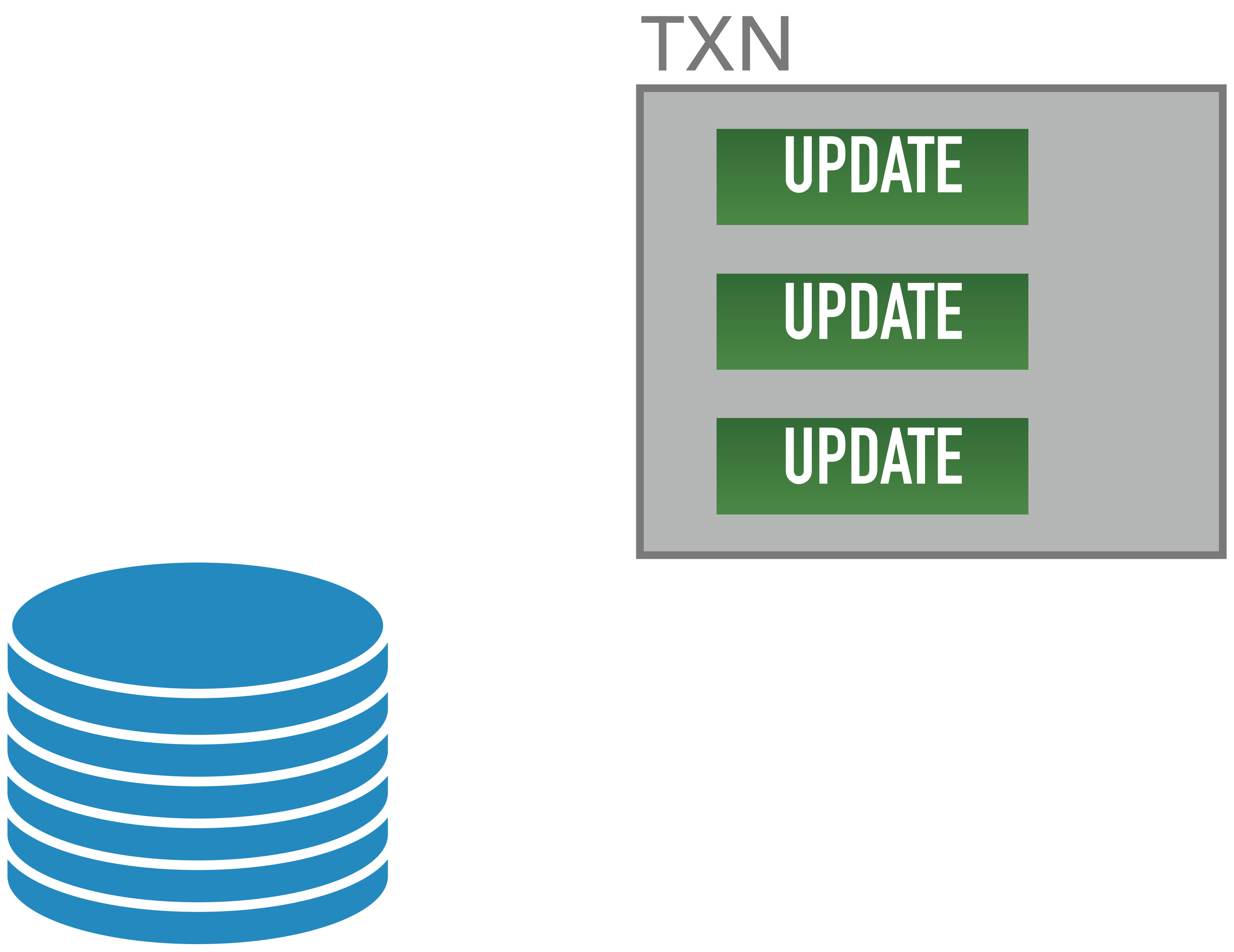
# TRADITIONAL DATABASE PROGRAMMING

- ACID guarantees
  - **A**tomicity



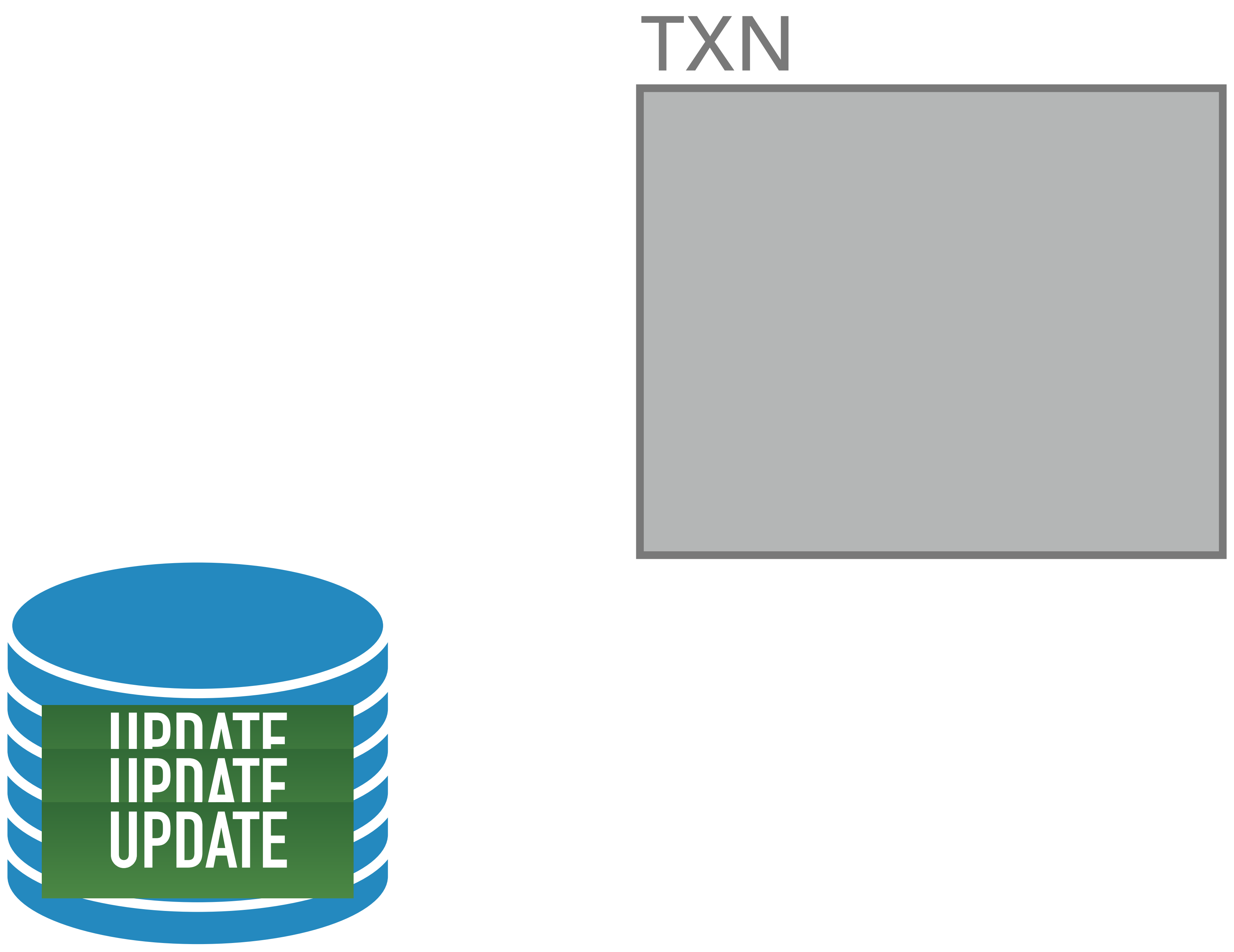
# TRADITIONAL DATABASE PROGRAMMING

- ACID guarantees
  - **A**tomicity



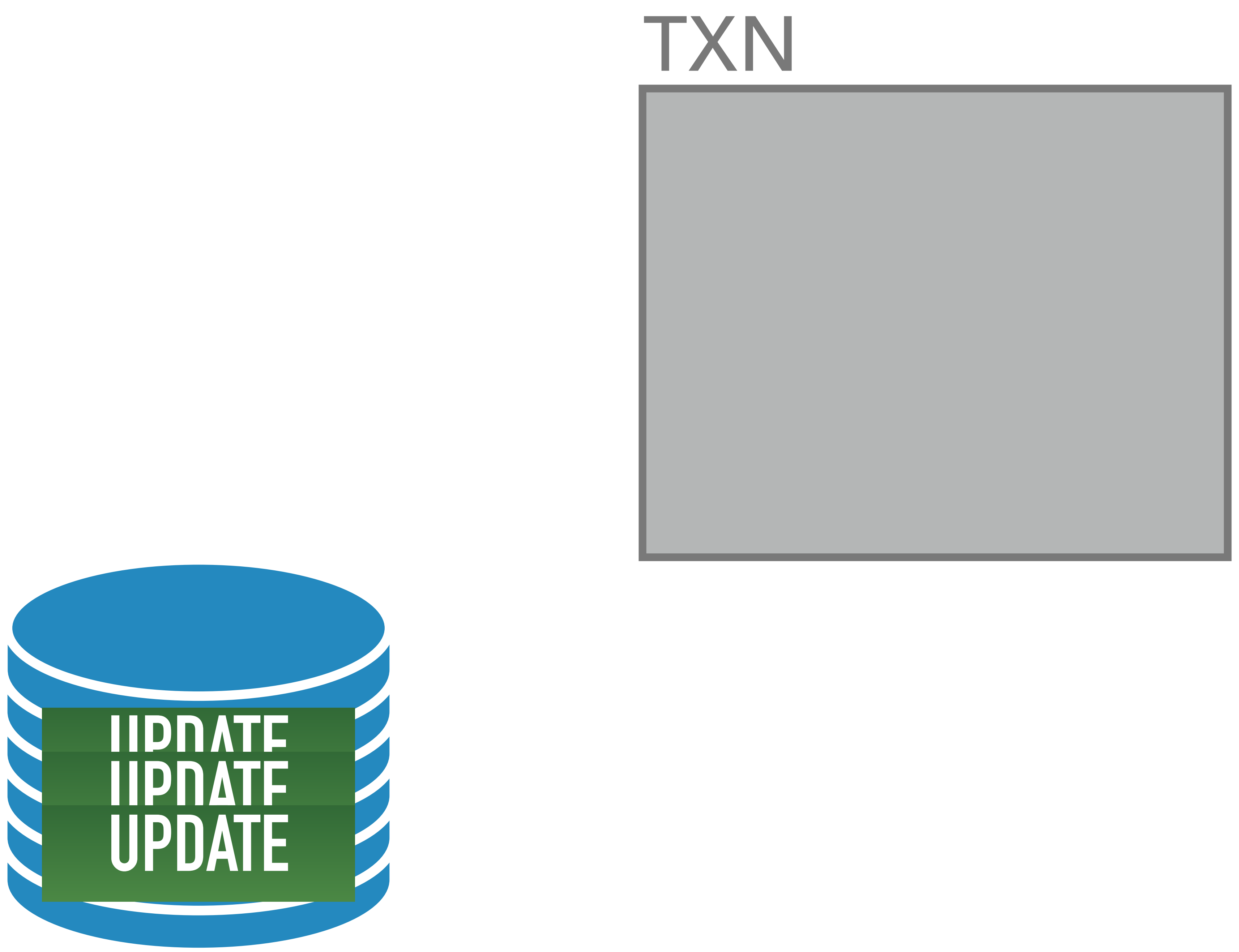
# TRADITIONAL DATABASE PROGRAMMING

- ACID guarantees
  - **A**tomicity



## TRADITIONAL DATABASE PROGRAMMING

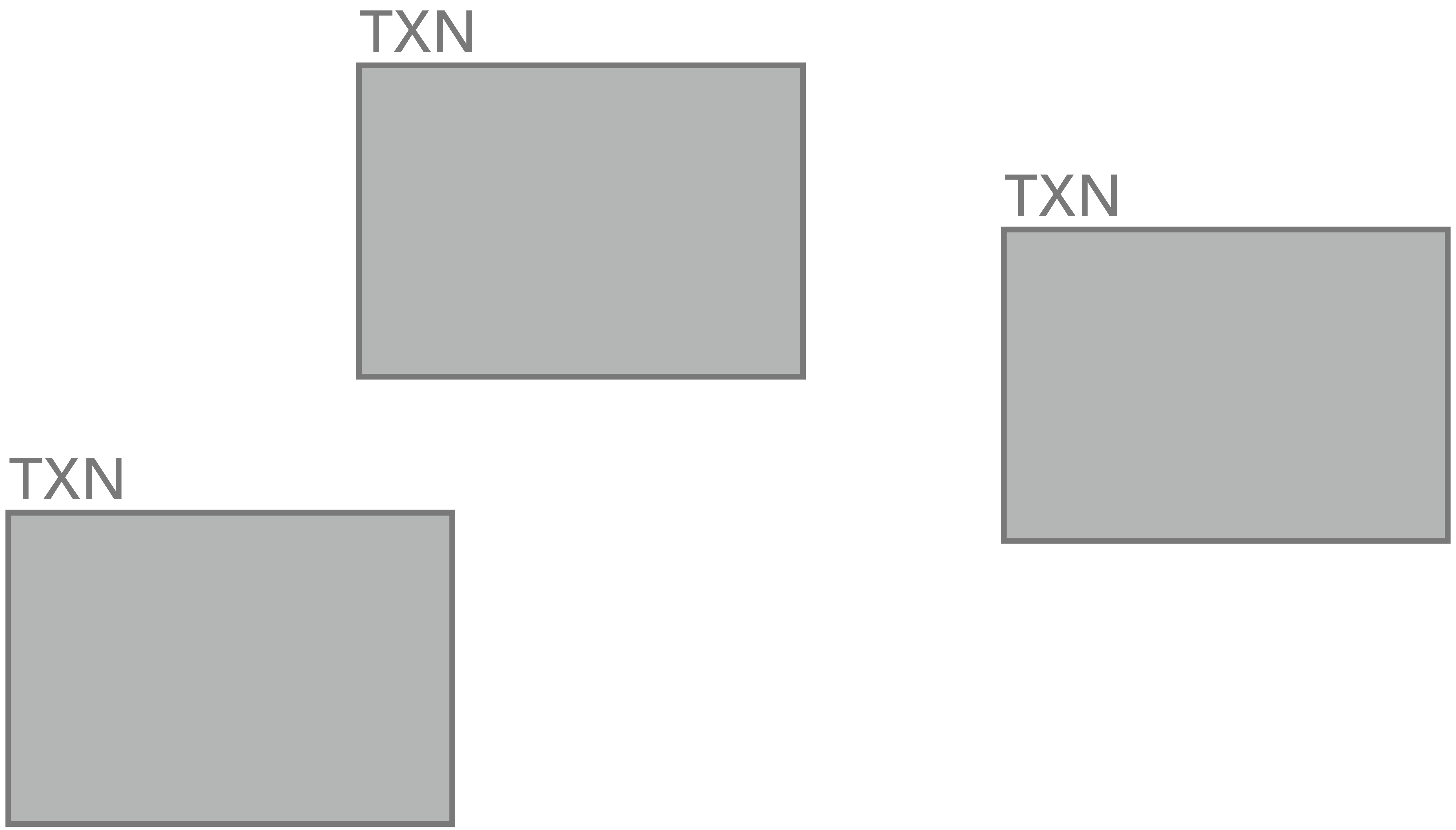
- ACID guarantees
  - **A**tomicity



*“All or None”*

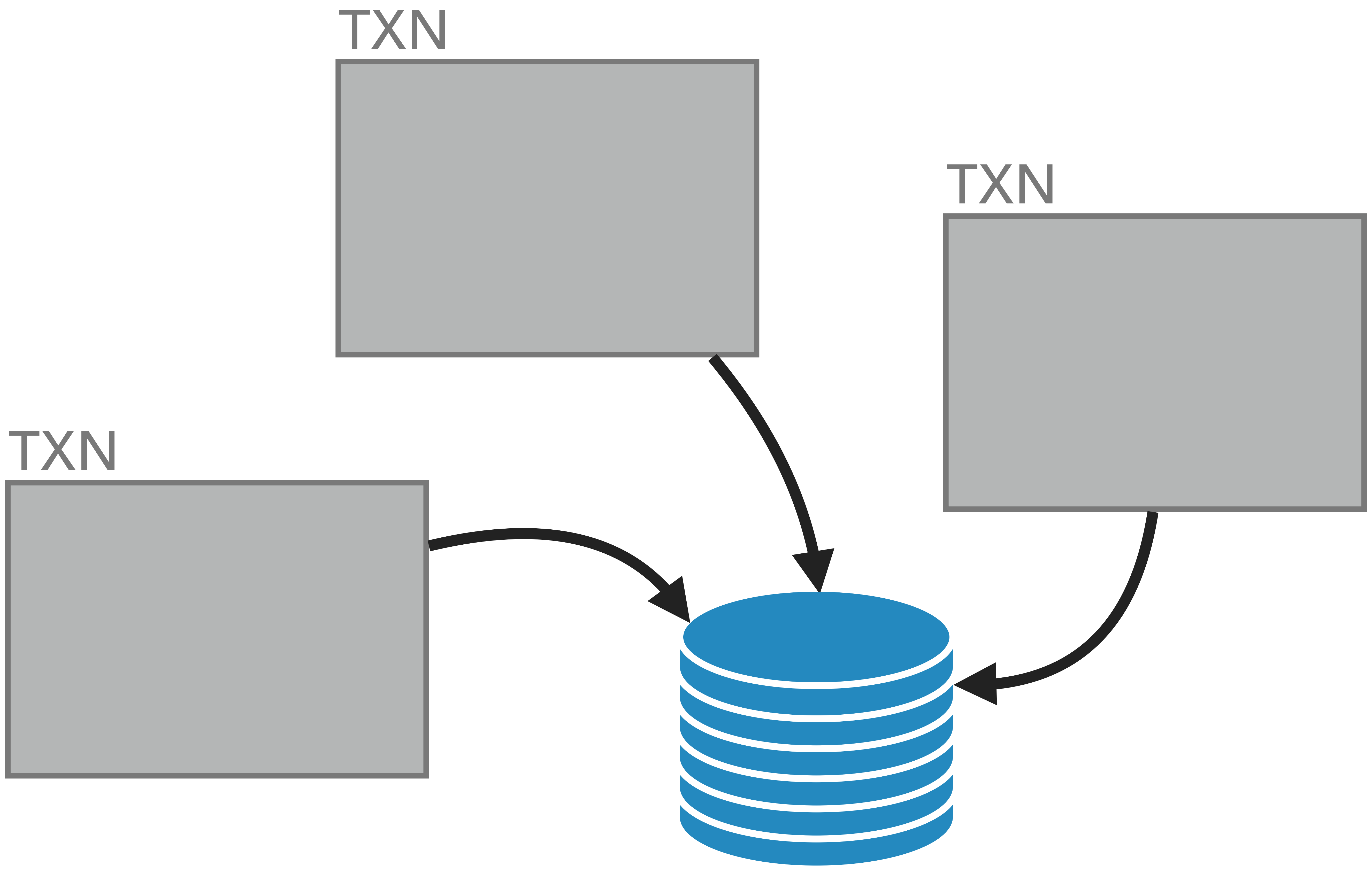
# TRADITIONAL DATABASE PROGRAMMING

- ACID guarantees
  - **A**tomicity
  - **C**onsistency



# TRADITIONAL DATABASE PROGRAMMING

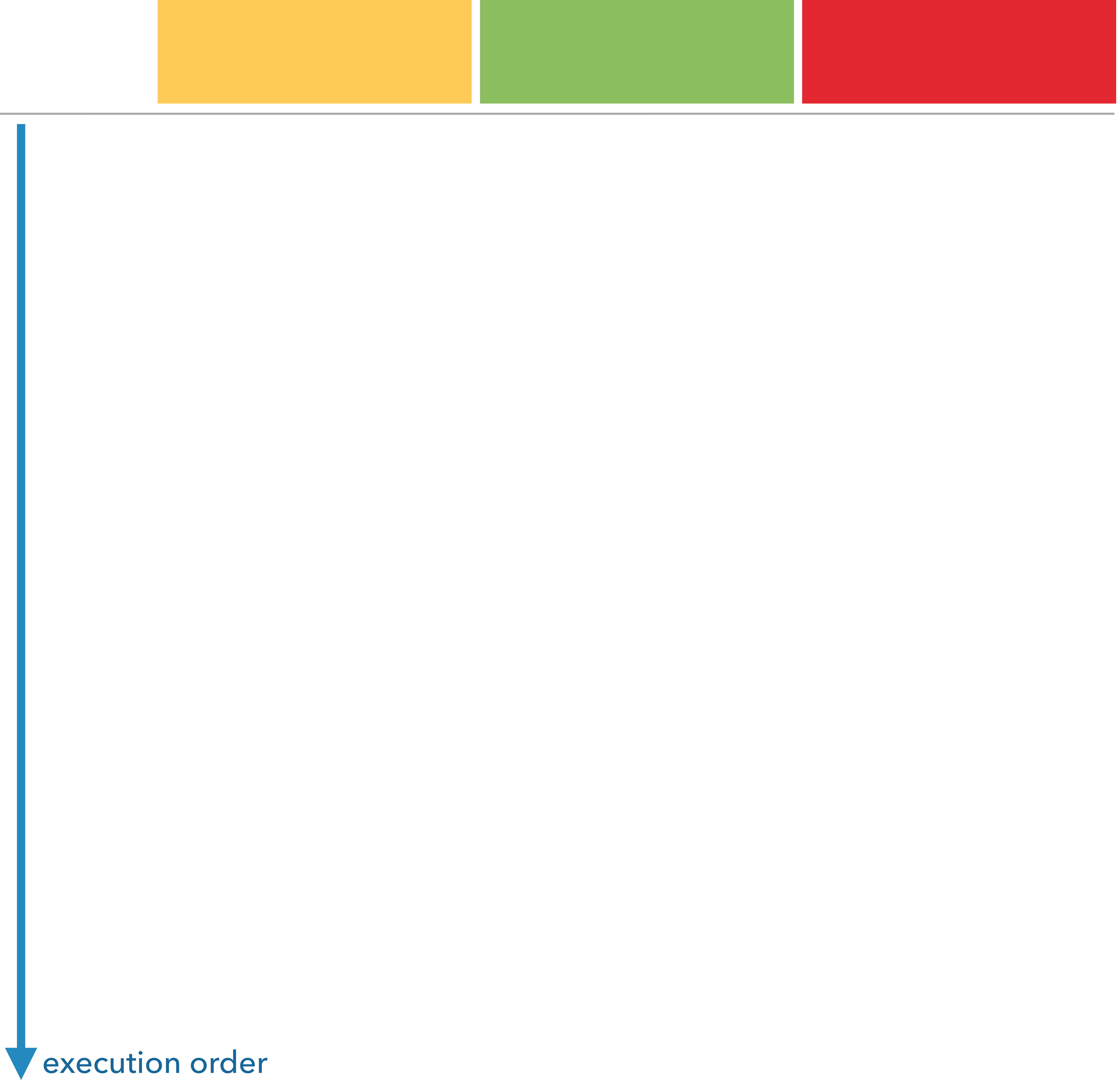
- ACID guarantees
  - **A**tomicity
  - **C**onsistency



*"Single Copy of Data"*

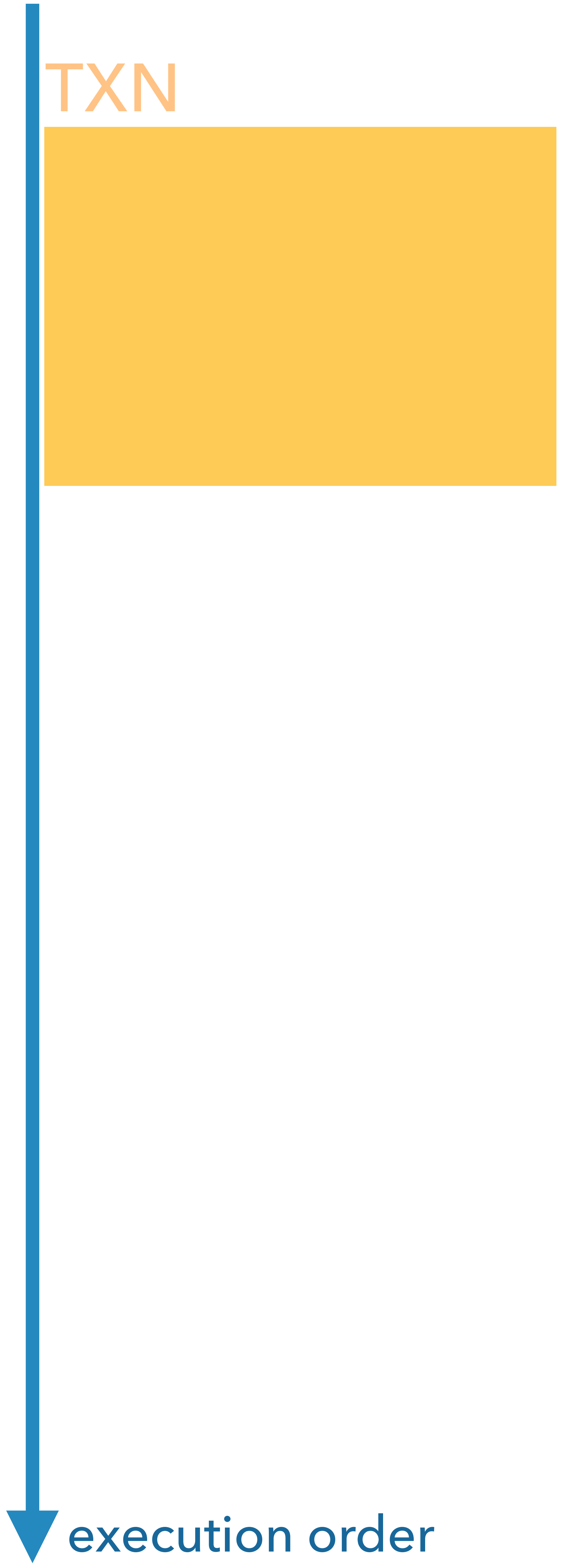
# TRADITIONAL DATABASE PROGRAMMING

- ACID guarantees
  - **A**tomicity
  - **C**onsistency
  - **I**solation



# TRADITIONAL DATABASE PROGRAMMING

- ACID guarantees
  - **A**tomicity
  - **C**onsistency
  - **I**solation

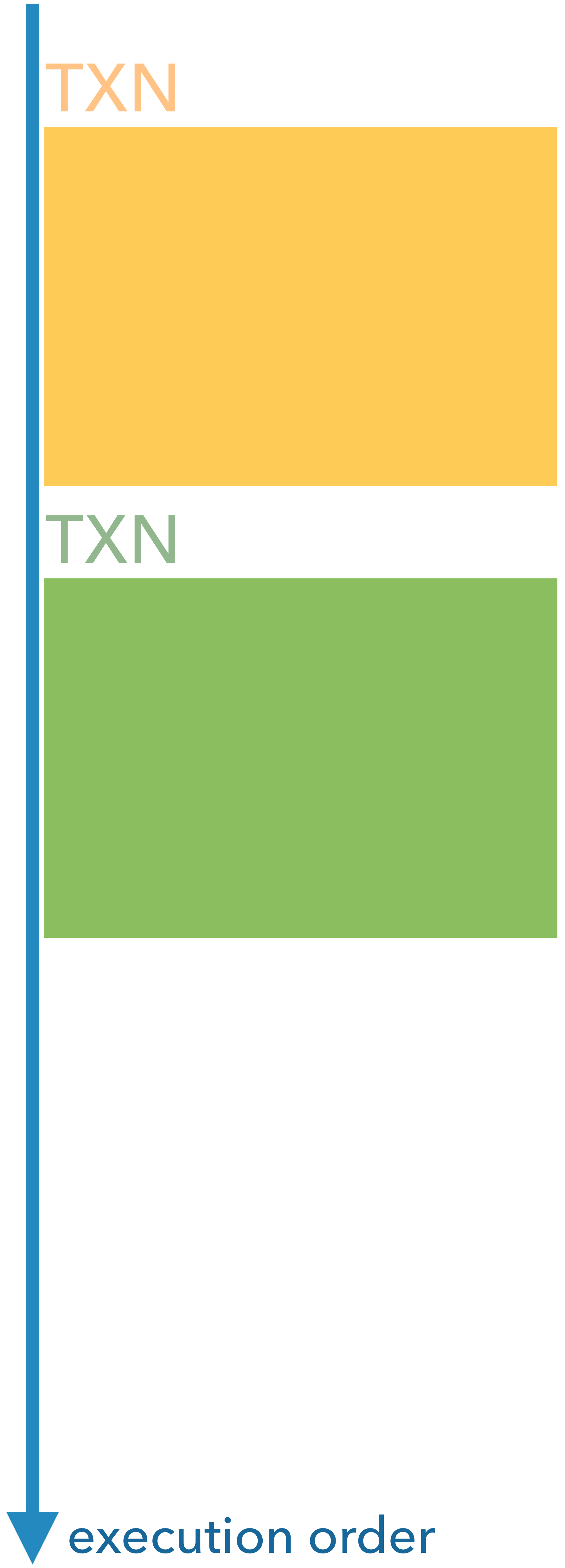


TXN

execution order

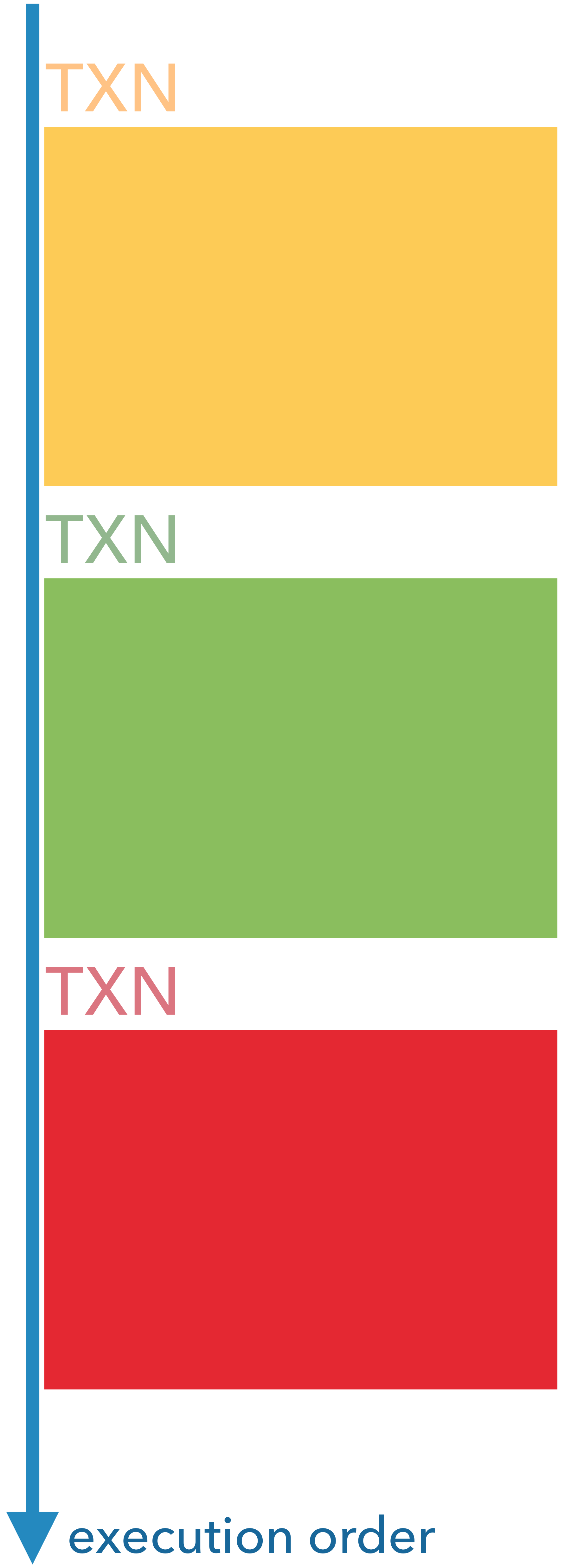
# TRADITIONAL DATABASE PROGRAMMING

- ACID guarantees
  - **A**tomicity
  - **C**onsistency
  - **I**solation



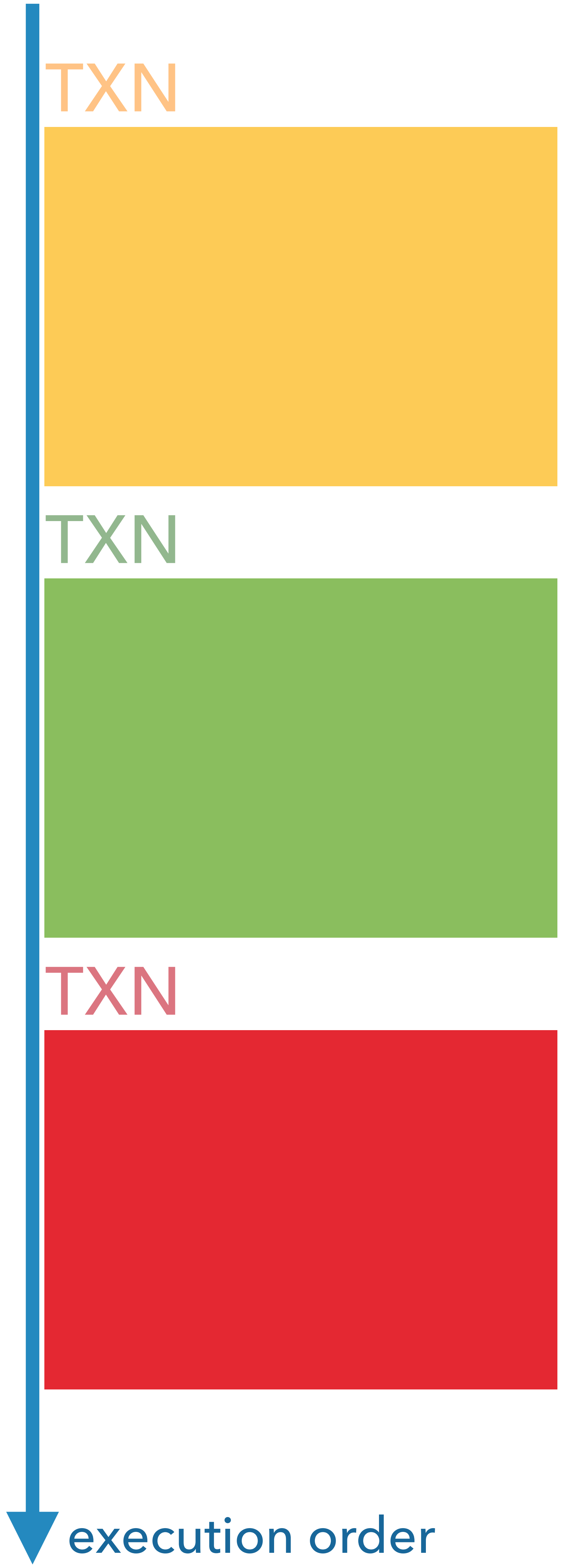
# TRADITIONAL DATABASE PROGRAMMING

- ACID guarantees
  - **A**tomicity
  - **C**onsistency
  - **I**solation



# TRADITIONAL DATABASE PROGRAMMING

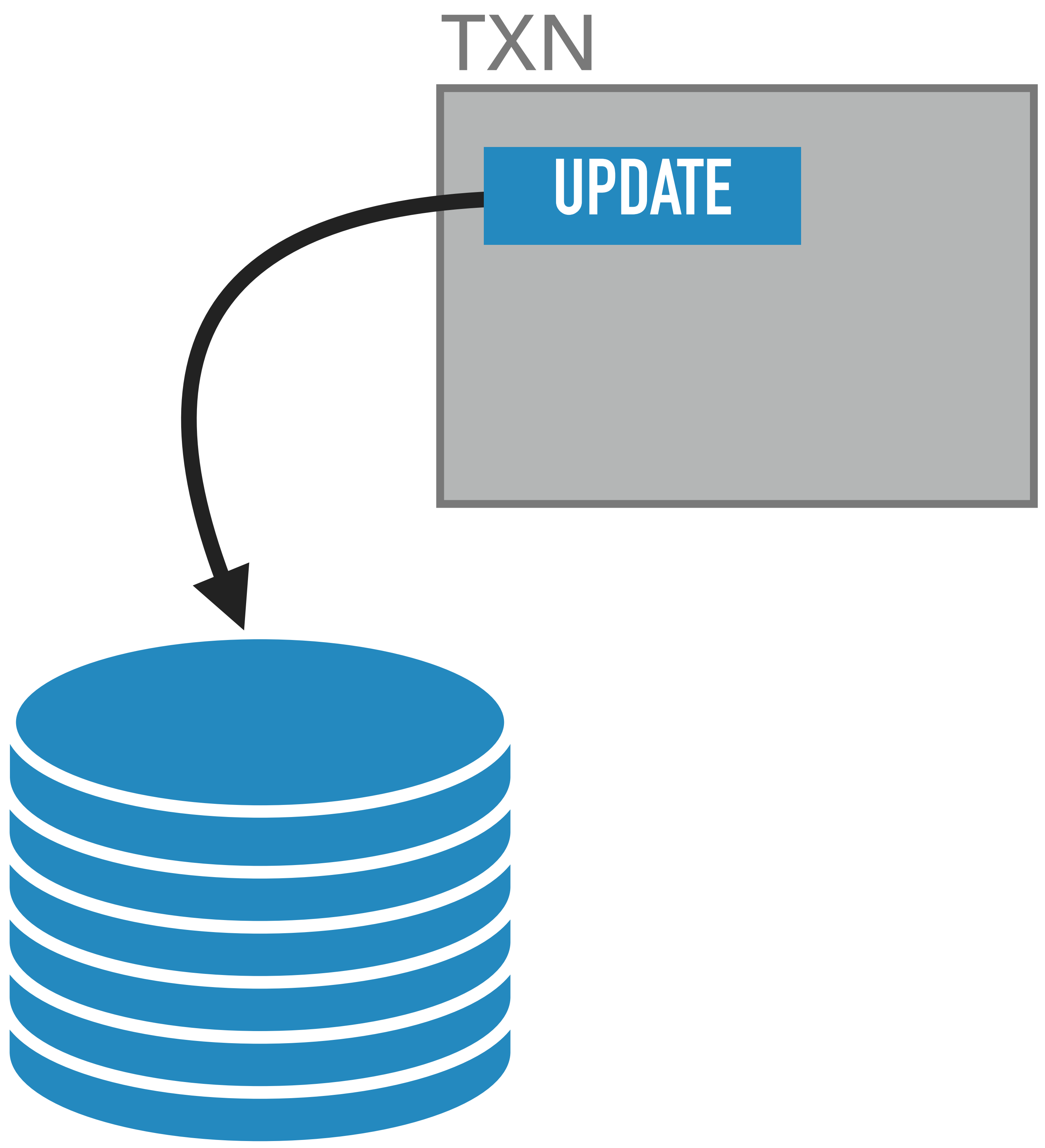
- ACID guarantees
  - **A**tomicity
  - **C**onsistency
  - **I**solation



*“No Interference”*

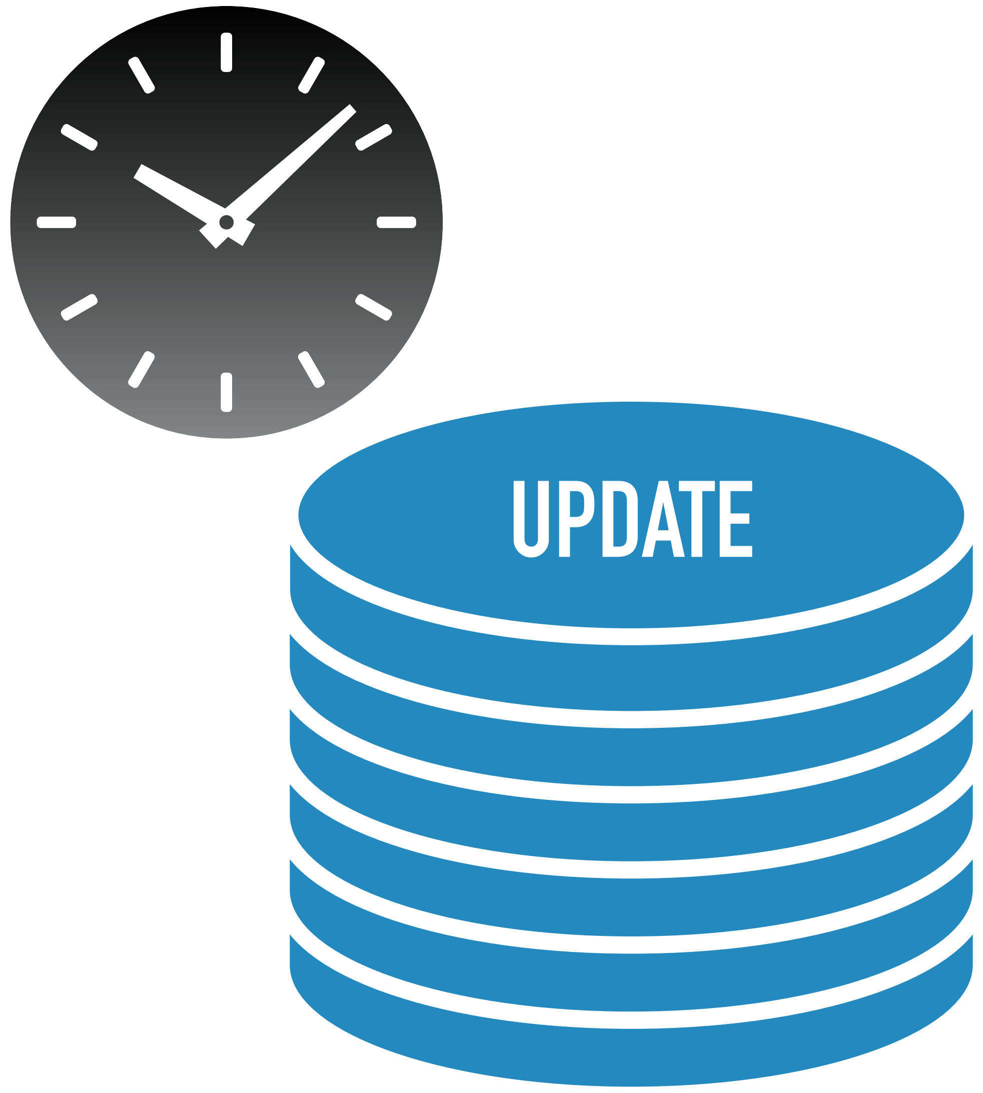
# TRADITIONAL DATABASE PROGRAMMING

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



# TRADITIONAL DATABASE PROGRAMMING

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



*"Permanent Commits"*

# SERIALIZABILITY GUARANTEES

- ACID guarantees
  - **A**tomicity
  - **C**onsistency
  - **I**solation
  - **D**urability
- **Serializability**

# SERIALIZABILITY GUARANTEES

- ACID guarantees
  - **A**tomicity
  - **C**onsistency
  - **I**solation
  - **D**urability
- **Serializability** facilitates program design and reasoning

# SERIALIZABILITY GUARANTEES

- ACID guarantees
  - **A**tomicity
  - **C**onsistency
  - **I**solation
  - **D**urability
- **Serializability** facilitates program design and reasoning

## SERIALIZABILITY GUARANTEES

- ACID guarantees
  - **A**tomicity
  - **C**onsistency
  - **I**solation
  - **D**urability
- **Serializability** facilitates program design and reasoning

TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg
UPDATE pay_cnt=v+1
  WHERE id=arg
```

# SERIALIZABILITY GUARANTEES

- ACID guarantees
  - **A**tomicity
  - **C**onsistency
  - **I**solation
  - **D**urability
- **Serializability** facilitates program design and reasoning

TXN (arg)

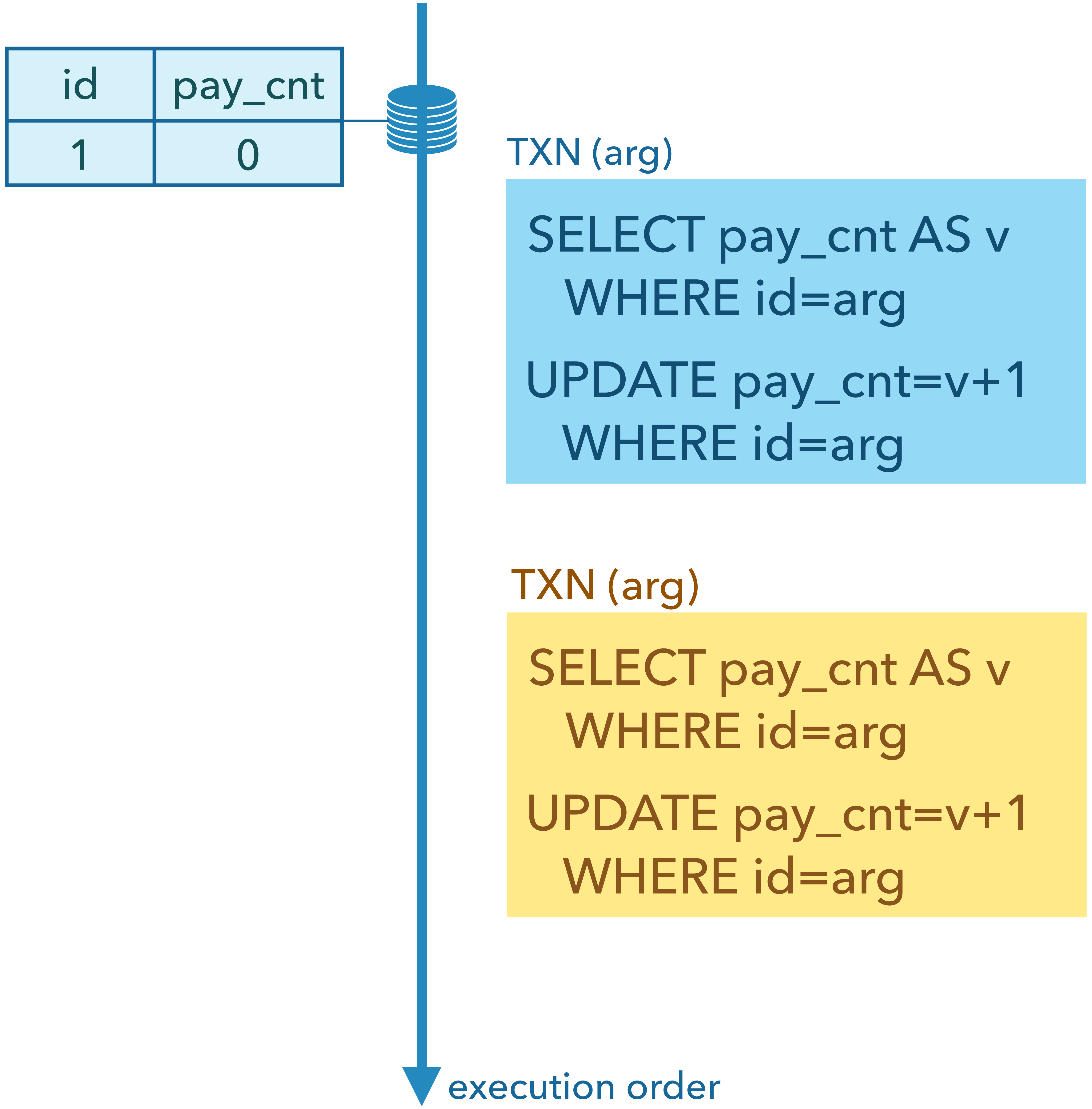
```
SELECT pay_cnt AS v
  WHERE id=arg
UPDATE pay_cnt=v+1
  WHERE id=arg
```

TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg
UPDATE pay_cnt=v+1
  WHERE id=arg
```

# SERIALIZABILITY GUARANTEES

- ACID guarantees
  - **Atomicity**
  - **Consistency**
  - **Isolation**
  - **Durability**
- **Serializability** facilitates program design and reasoning



| id | pay_cnt |
|---|---|
| 1 | 0 |

TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg
UPDATE pay_cnt=v+1
  WHERE id=arg
```

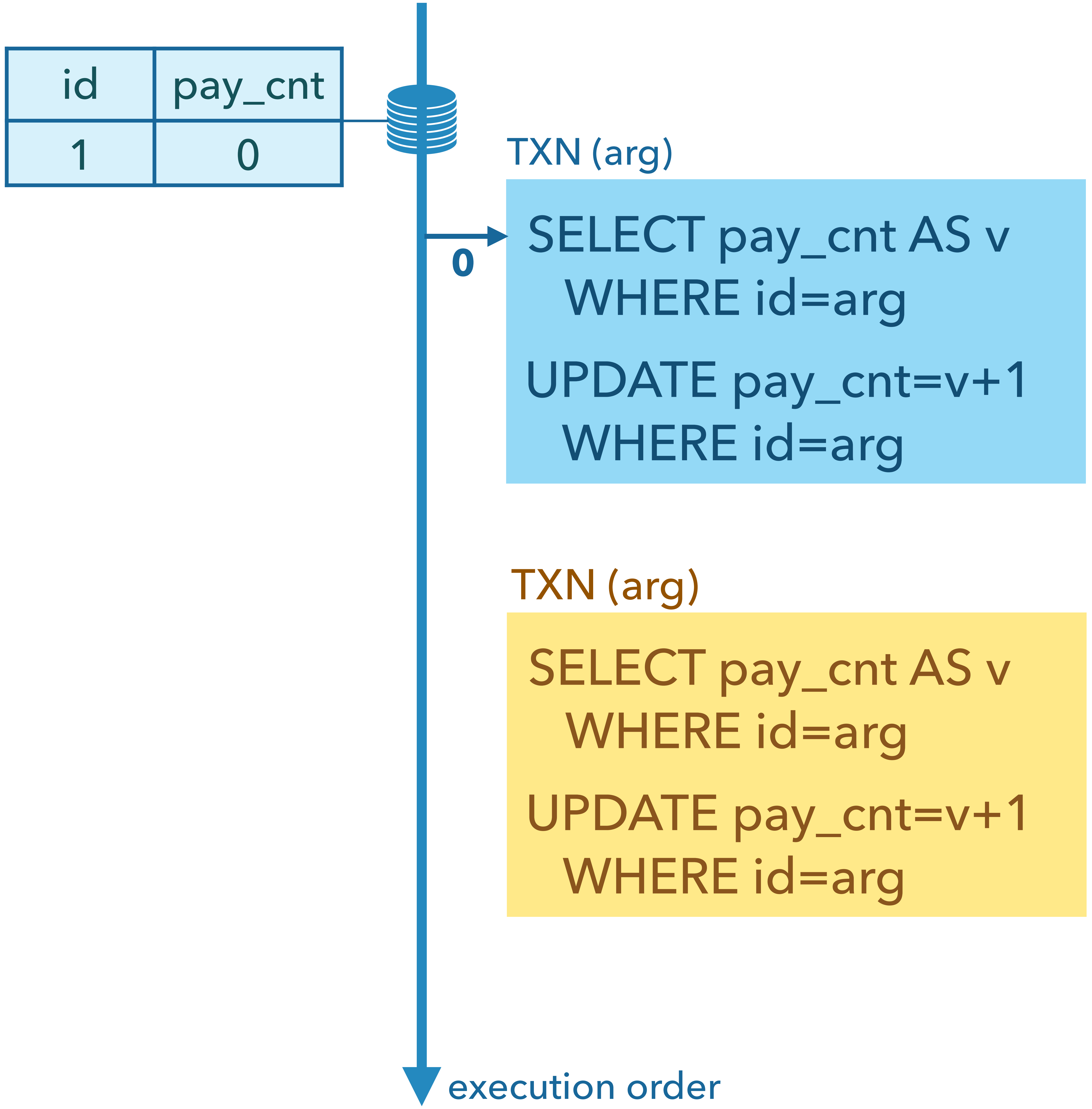
TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg
UPDATE pay_cnt=v+1
  WHERE id=arg
```

# SERIALIZABILITY GUARANTEES

- ACID guarantees
  - **Atomicity**
  - **Consistency**
  - **Isolation**
  - **Durability**

- **Serializability** facilitates program design and reasoning



| id | pay_cnt |
|---|---|
| 1 | 0 |

TXN (arg)

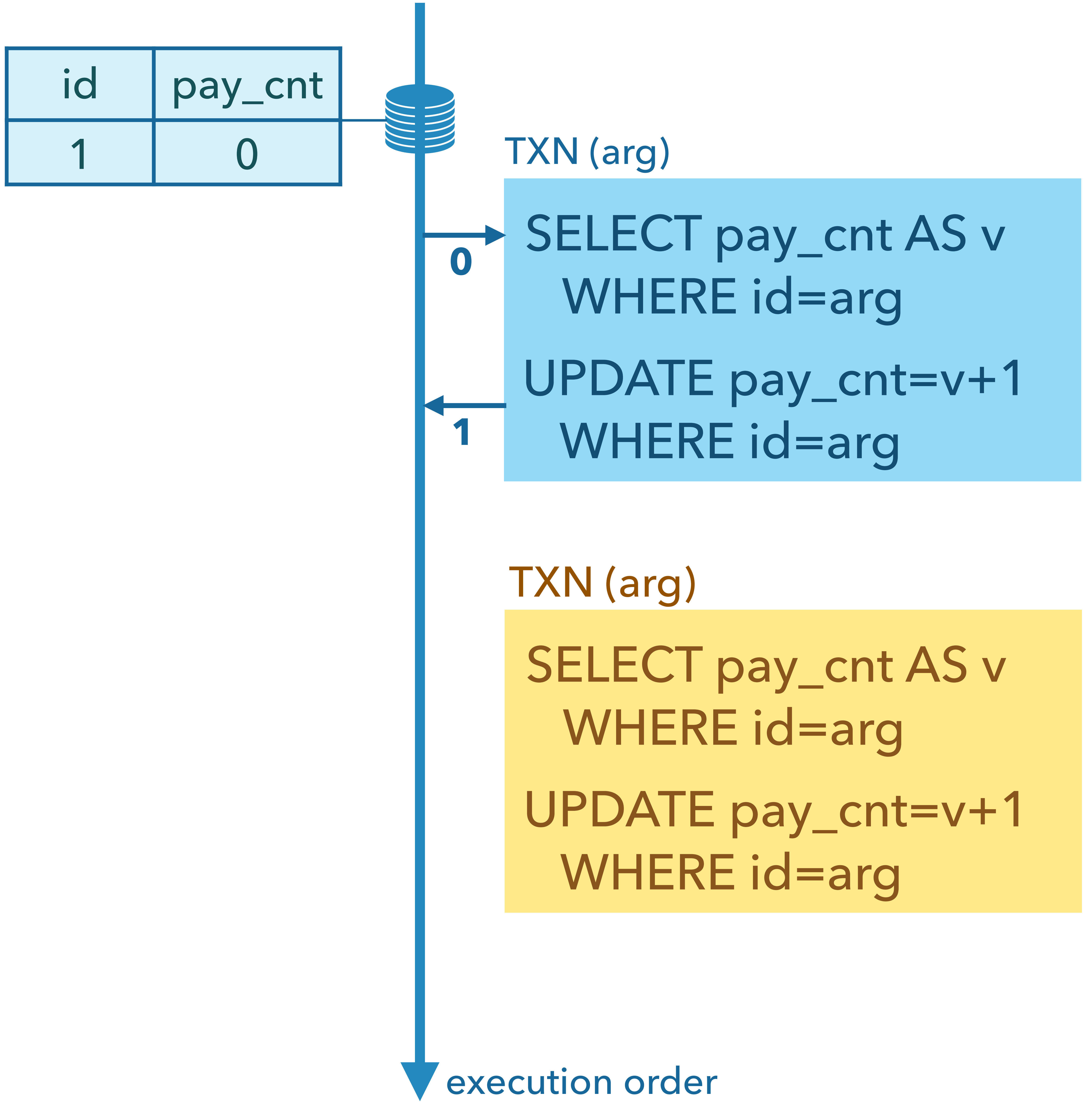
```
SELECT pay_cnt AS v
  WHERE id=arg
UPDATE pay_cnt=v+1
  WHERE id=arg
```

TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg
UPDATE pay_cnt=v+1
  WHERE id=arg
```

## SERIALIZABILITY GUARANTEES

- ACID guarantees
  - **Atomicity**
  - **Consistency**
  - **Isolation**
  - **Durability**
- **Serializability** facilitates program design and reasoning



| id | pay_cnt |
|---|---|
| 1 | 0 |

TXN (arg)

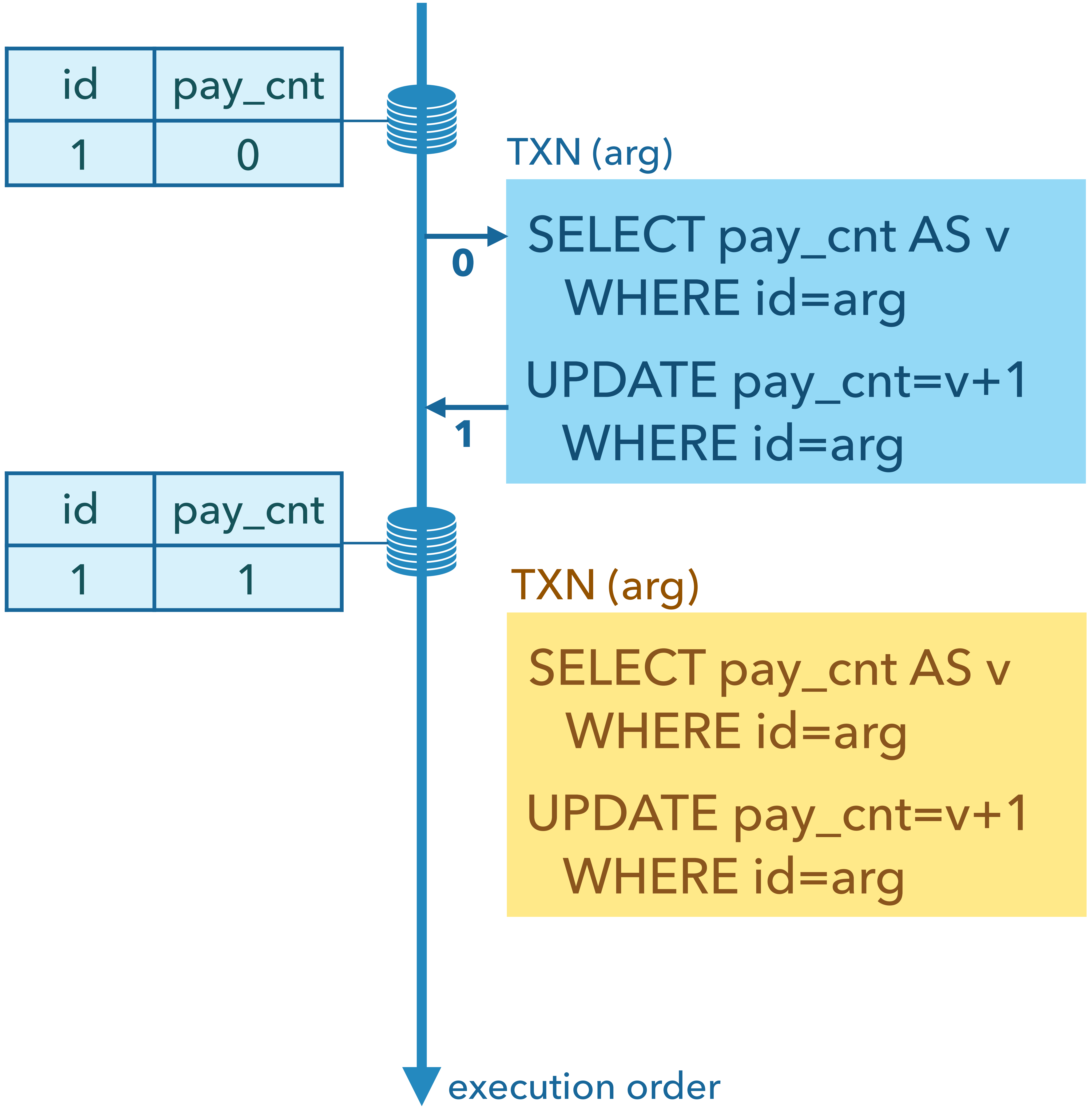
```
SELECT pay_cnt AS v
  WHERE id=arg
UPDATE pay_cnt=v+1
  WHERE id=arg
```

TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg
UPDATE pay_cnt=v+1
  WHERE id=arg
```

# SERIALIZABILITY GUARANTEES

- ACID guarantees
  - **Atomicity**
  - **Consistency**
  - **Isolation**
  - **Durability**
- **Serializability** facilitates program design and reasoning



| id | pay_cnt |
|---|---|
| 1 | 0 |

TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg
UPDATE pay_cnt=v+1
  WHERE id=arg
```

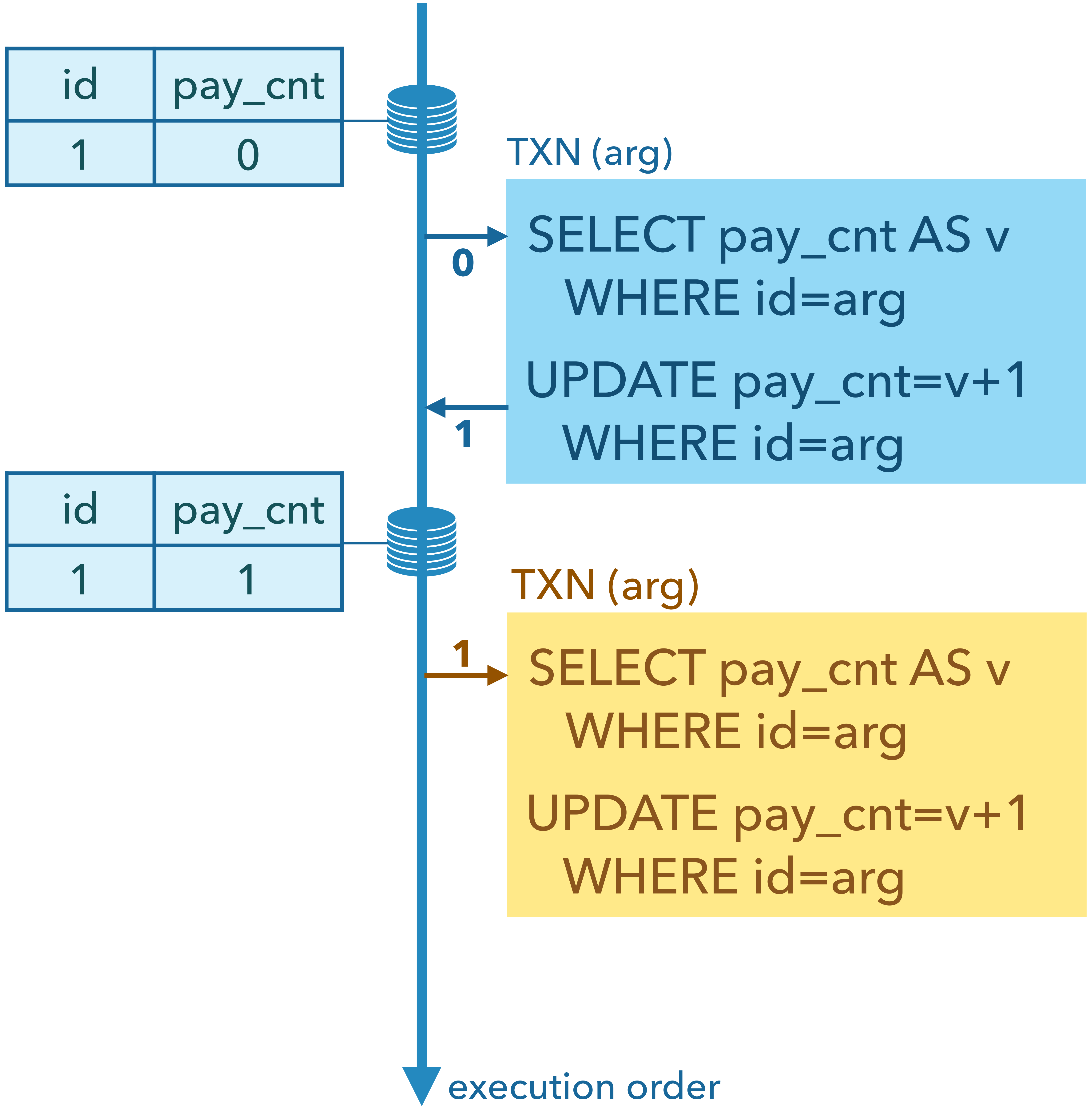
| id | pay_cnt |
|---|---|
| 1 | 1 |

TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg
UPDATE pay_cnt=v+1
  WHERE id=arg
```

# SERIALIZABILITY GUARANTEES

- ACID guarantees
  - **Atomicity**
  - **Consistency**
  - **Isolation**
  - **Durability**
- **Serializability** facilitates program design and reasoning



| id | pay_cnt |
|---|---|
| 1 | 0 |

TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg
UPDATE pay_cnt=v+1
  WHERE id=arg
```

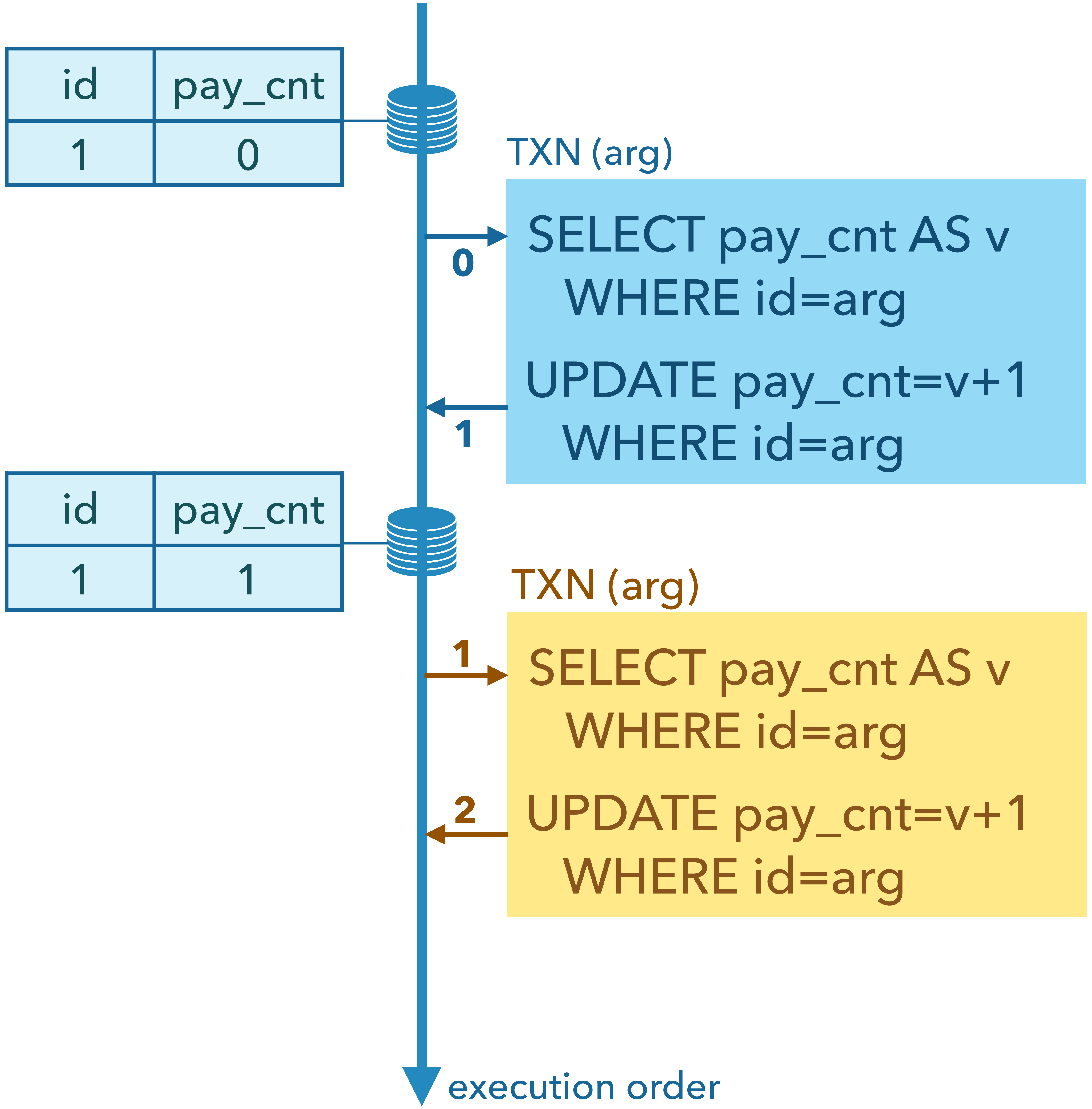
| id | pay_cnt |
|---|---|
| 1 | 1 |

TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg
UPDATE pay_cnt=v+1
  WHERE id=arg
```

# SERIALIZABILITY GUARANTEES

- ACID guarantees
  - **Atomicity**
  - **Consistency**
  - **Isolation**
  - **Durability**
- **Serializability** facilitates program design and reasoning



| id | pay_cnt |
|---|---|
| 1 | 0 |

TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg
UPDATE pay_cnt=v+1
  WHERE id=arg
```

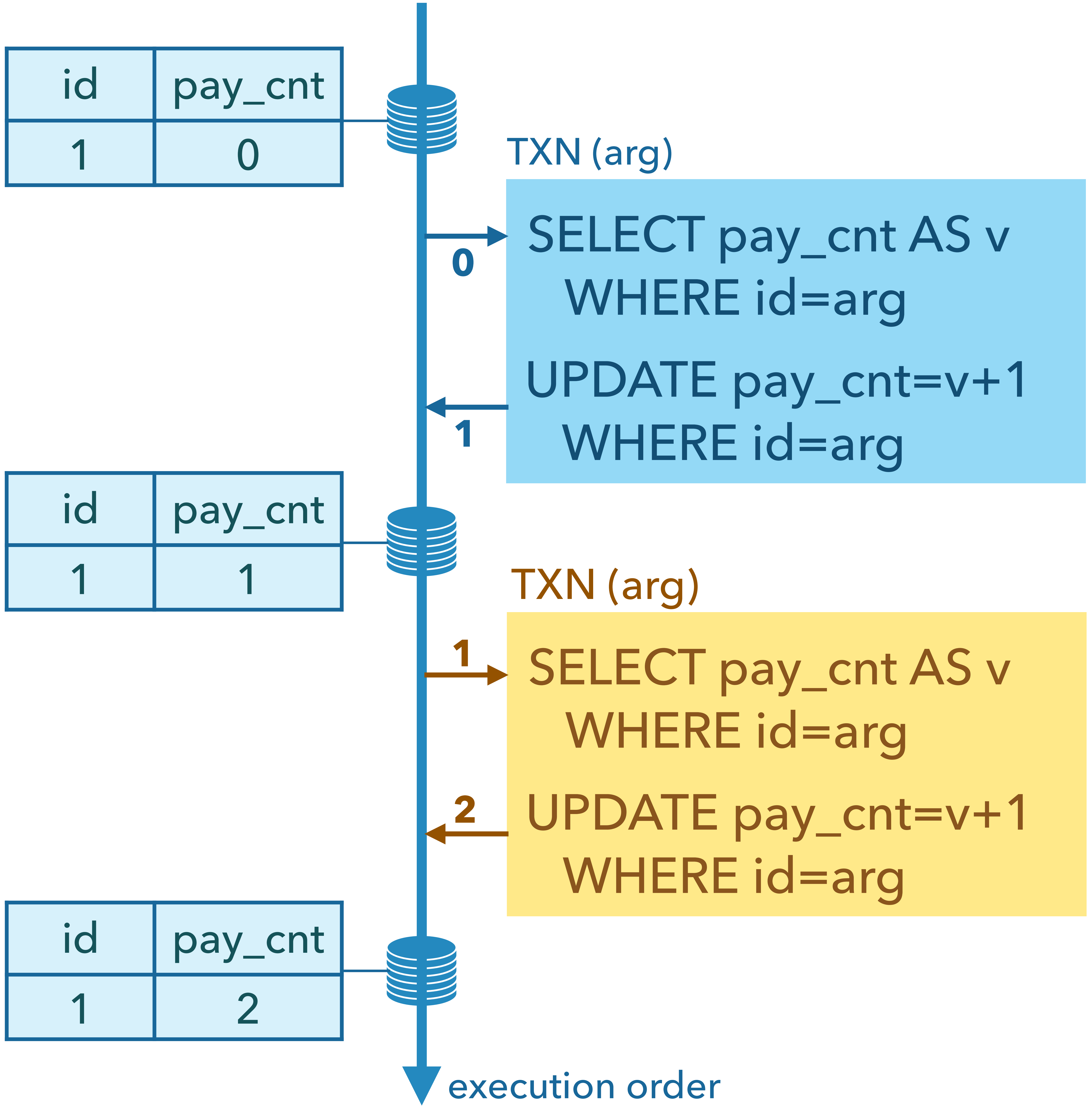
| id | pay_cnt |
|---|---|
| 1 | 1 |

TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg
UPDATE pay_cnt=v+1
  WHERE id=arg
```

# SERIALIZABILITY GUARANTEES

- ACID guarantees
  - **Atomicity**
  - **Consistency**
  - **Isolation**
  - **Durability**
- **Serializability** facilitates program design and reasoning



| id | pay_cnt |
|---|---|
| 1 | 0 |

TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg
UPDATE pay_cnt=v+1
  WHERE id=arg
```

| id | pay_cnt |
|---|---|
| 1 | 1 |

TXN (arg)

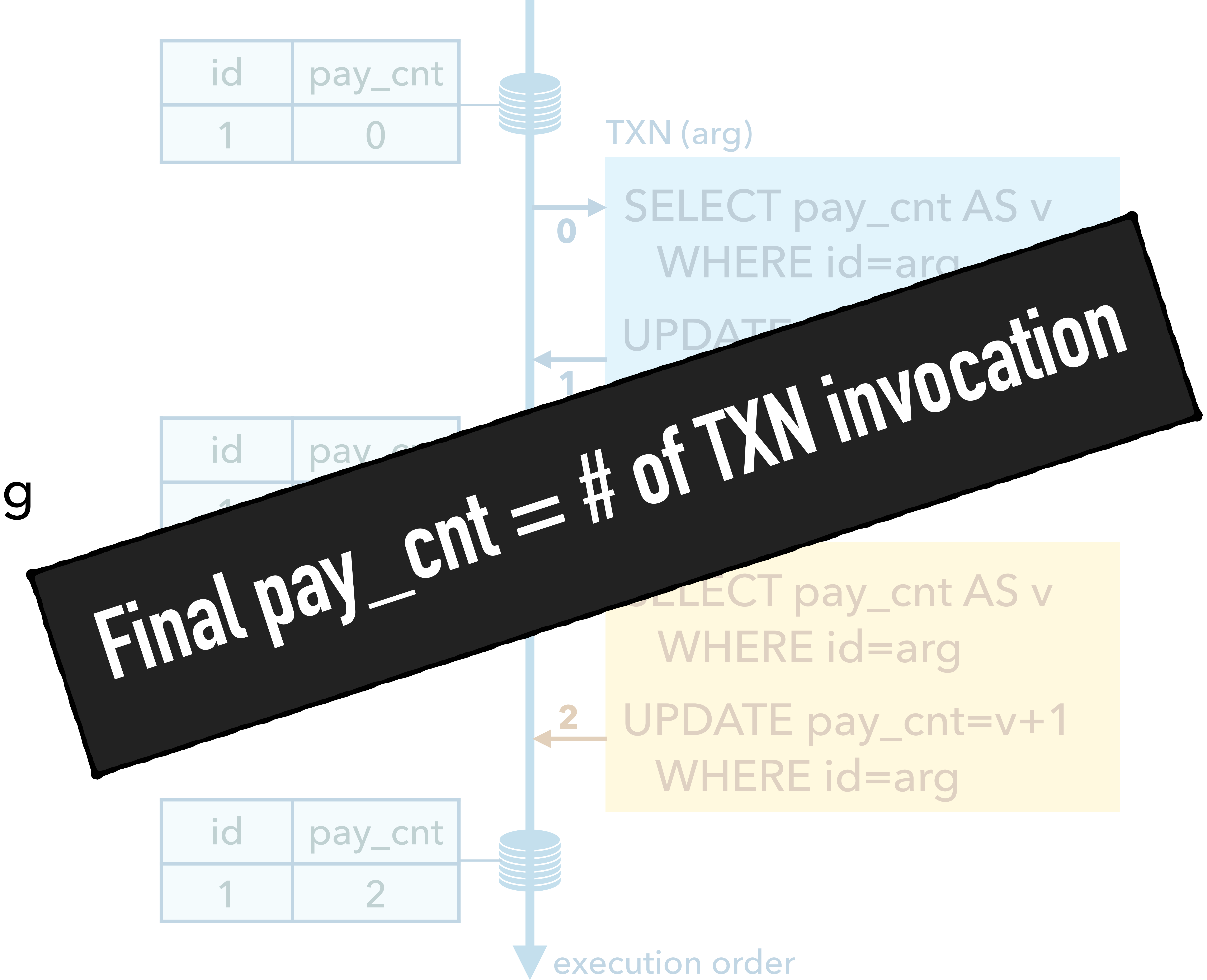
```
SELECT pay_cnt AS v
  WHERE id=arg
UPDATE pay_cnt=v+1
  WHERE id=arg
```

| id | pay_cnt |
|---|---|
| 1 | 2 |

# SERIALIZABILITY GUARANTEES

- ACID guarantees
  - **Atomicity**
  - **Consistency**
  - **Isolation**
  - **Durability**

- **Serializability** facilitates program design and reasoning



| id | pay_cnt |
|---|---|
| 1 | 0 |

| id | pay_cnt |
|---|---|
| 1 | 2 |

Final pay_cnt = # of TXN invocation

# SERIALIZABILITY GUARANTEES

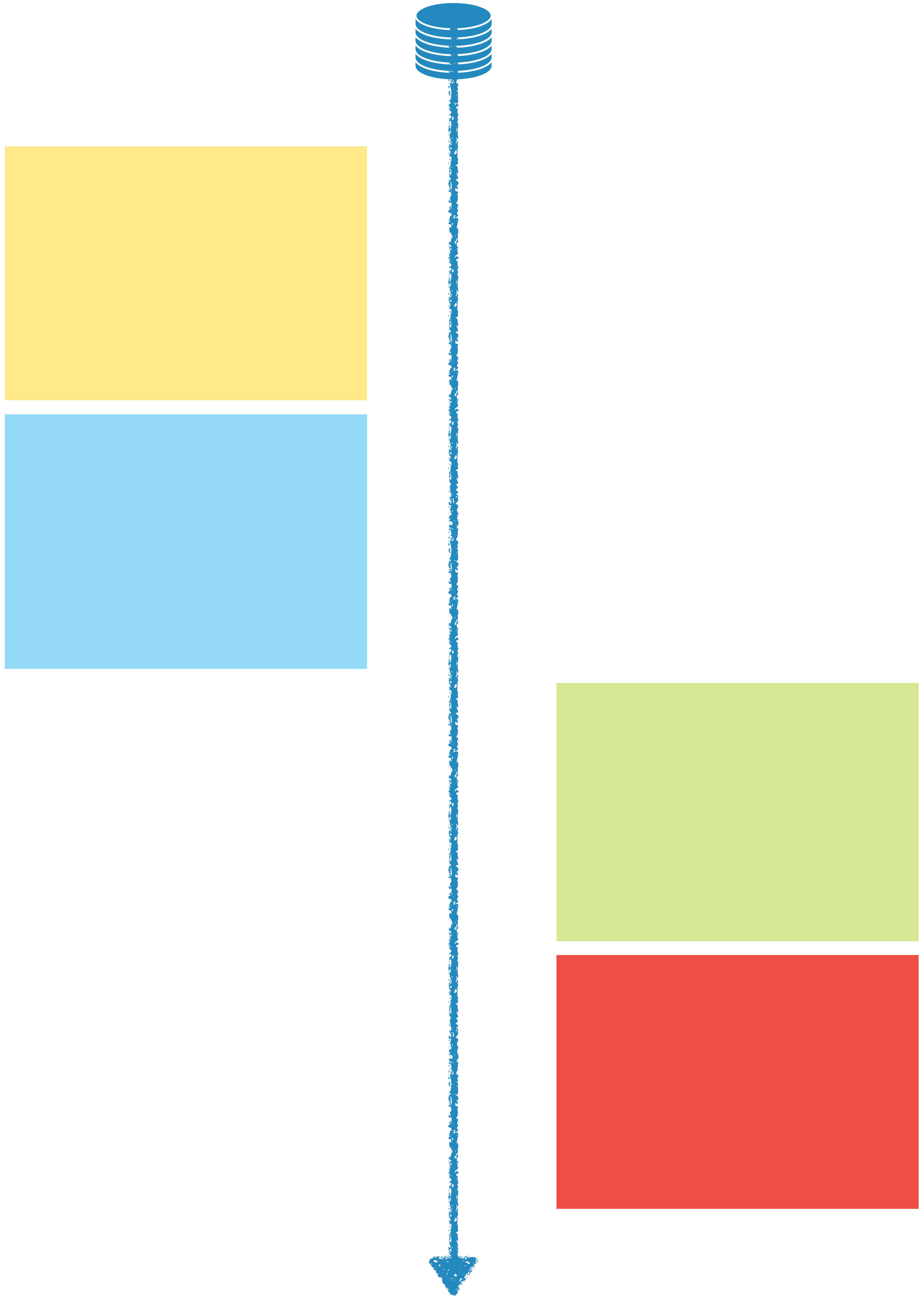
- ACID guarantees
  - **A**tomicity
  - **C**onsistency
  - **I**solation
  - **D**urability
- **Serializability** facilitates program design and reasoning
- Requires heavy synchronization

## SERIALIZABILITY GUARANTEES

- ACID guarantees
  - **Atomicity**
  - **Consistency**
  - **Isolation**
  - **Durability**

- **Serializability** facilitates program design and reasoning

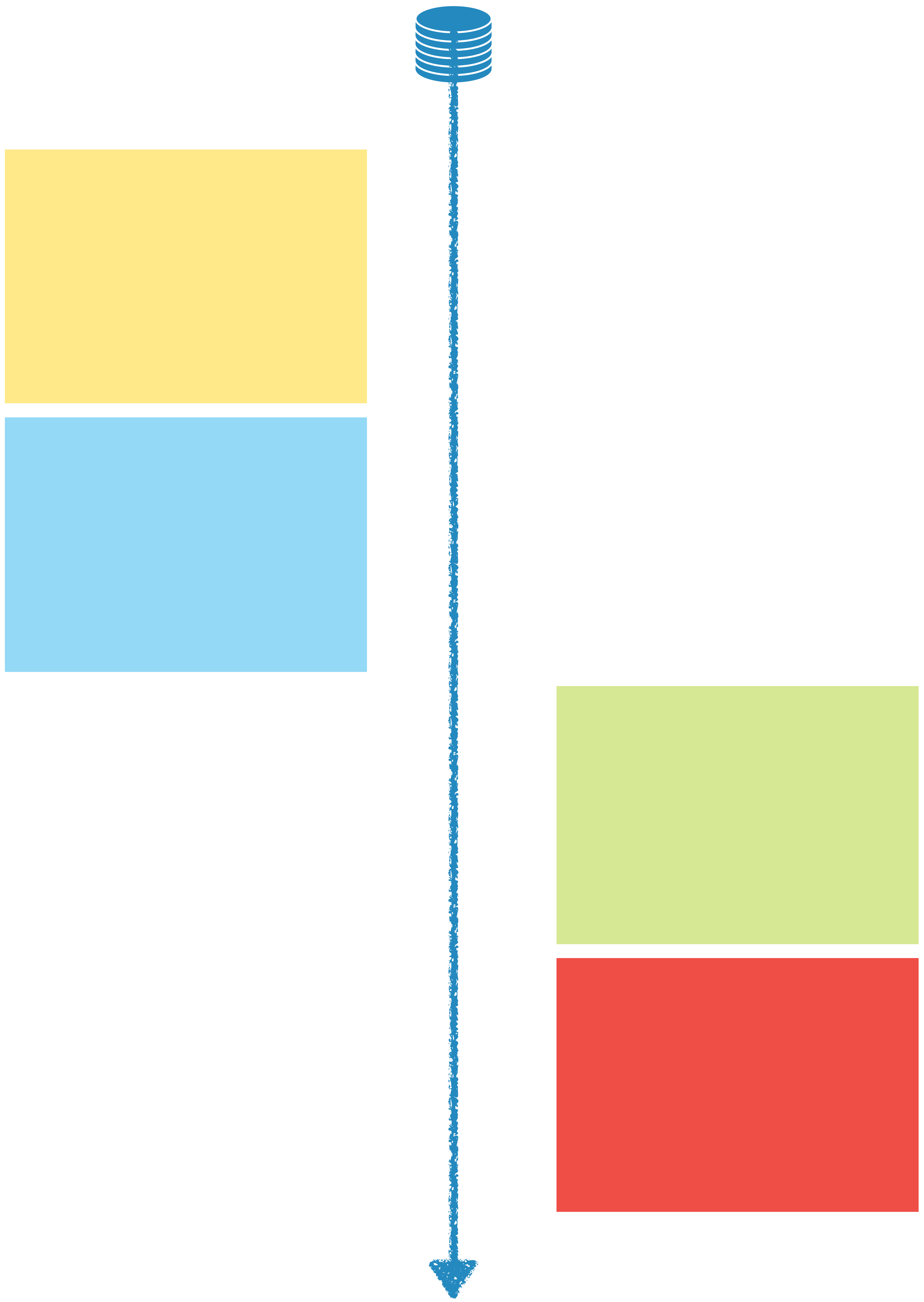
- Requires heavy synchronization



# SERIALIZABILITY GUARANTEES

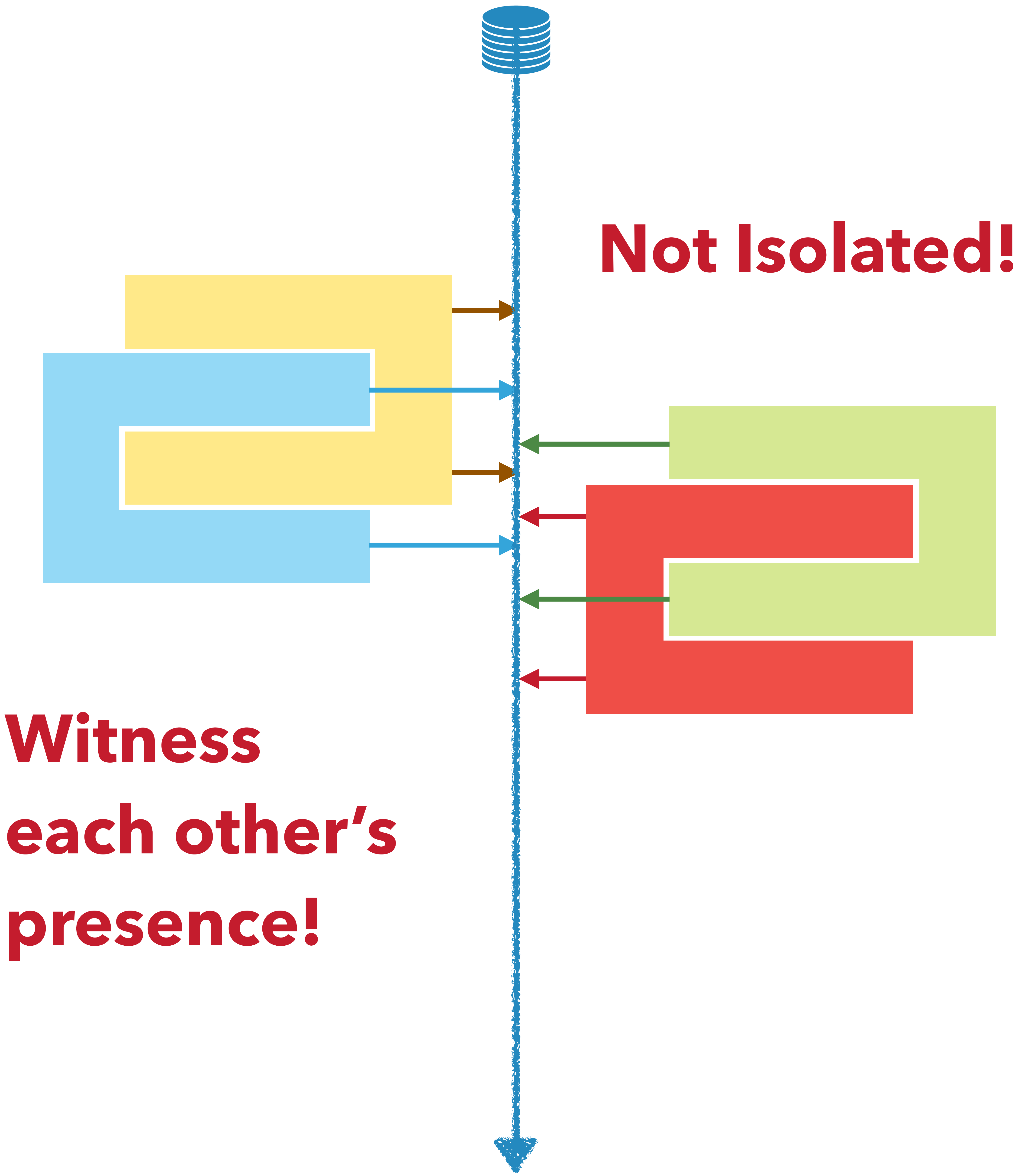
- ACID guarantees
  - **Atomicity**
  - **Consistency**
  - **Isolation**
  - **Durability**
- **Serializability** facilitates program design and reasoning
- Requires heavy synchronization

Unacceptable cost for web-scale applications



# SERIALIZABILITY GUARANTEES

- ACID guarantees
  - **Atomicity**
  - **Consistency**
  - **Isolation**
  - **Durability**
- **Serializability** facilitates program design and reasoning
- Requires heavy synchronization
- Weaker guarantees are offered in favor of higher performance



**Not Isolated!**

**Witness each other's presence!**

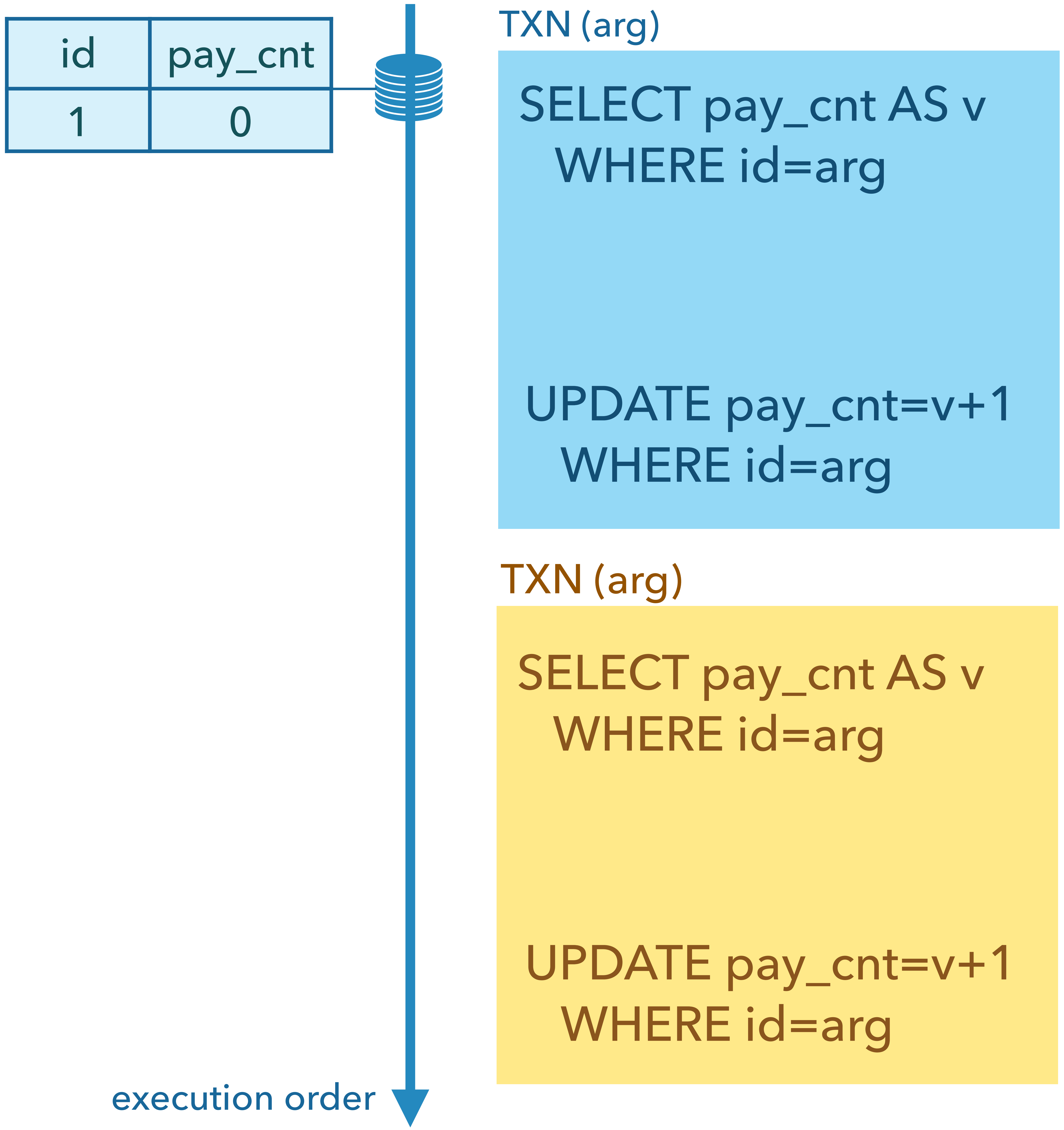
# EXAMPLE: A SERIALIZABILITY ANOMALY

# EXAMPLE: A SERIALIZABILITY ANOMALY

- Unexpected behaviors can occur under weak guarantees

# EXAMPLE: A SERIALIZABILITY ANOMALY

- Unexpected behaviors can occur under weak guarantees



| id | pay_cnt |
|---|---|
| 1 | 0 |

TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg

UPDATE pay_cnt=v+1
  WHERE id=arg
```

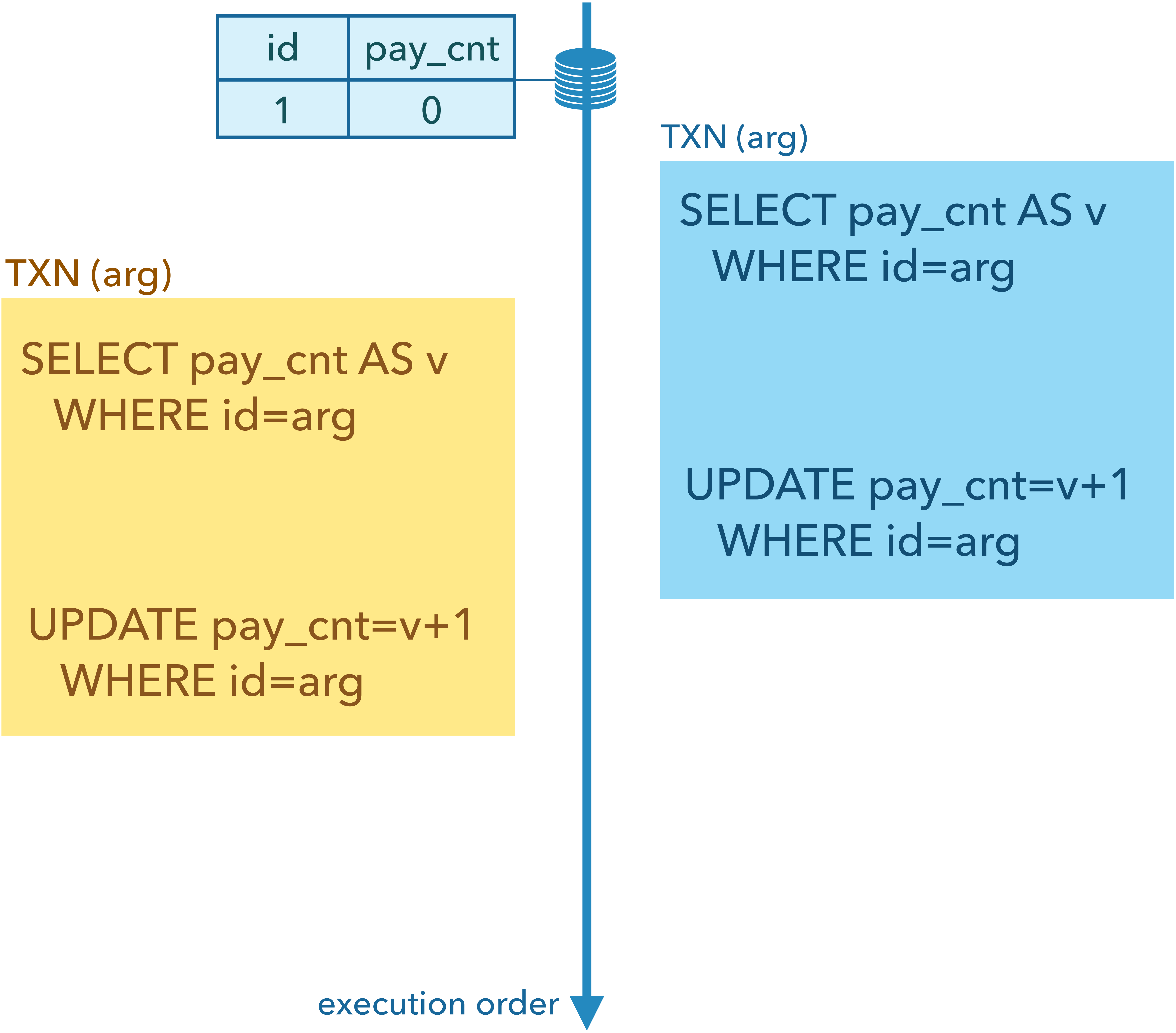
TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg

UPDATE pay_cnt=v+1
  WHERE id=arg
```

# EXAMPLE: A SERIALIZABILITY ANOMALY

- Unexpected behaviors can occur under weak guarantees



| id | pay_cnt |
|---|---|
| 1 | 0 |

TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg

UPDATE pay_cnt=v+1
  WHERE id=arg
```

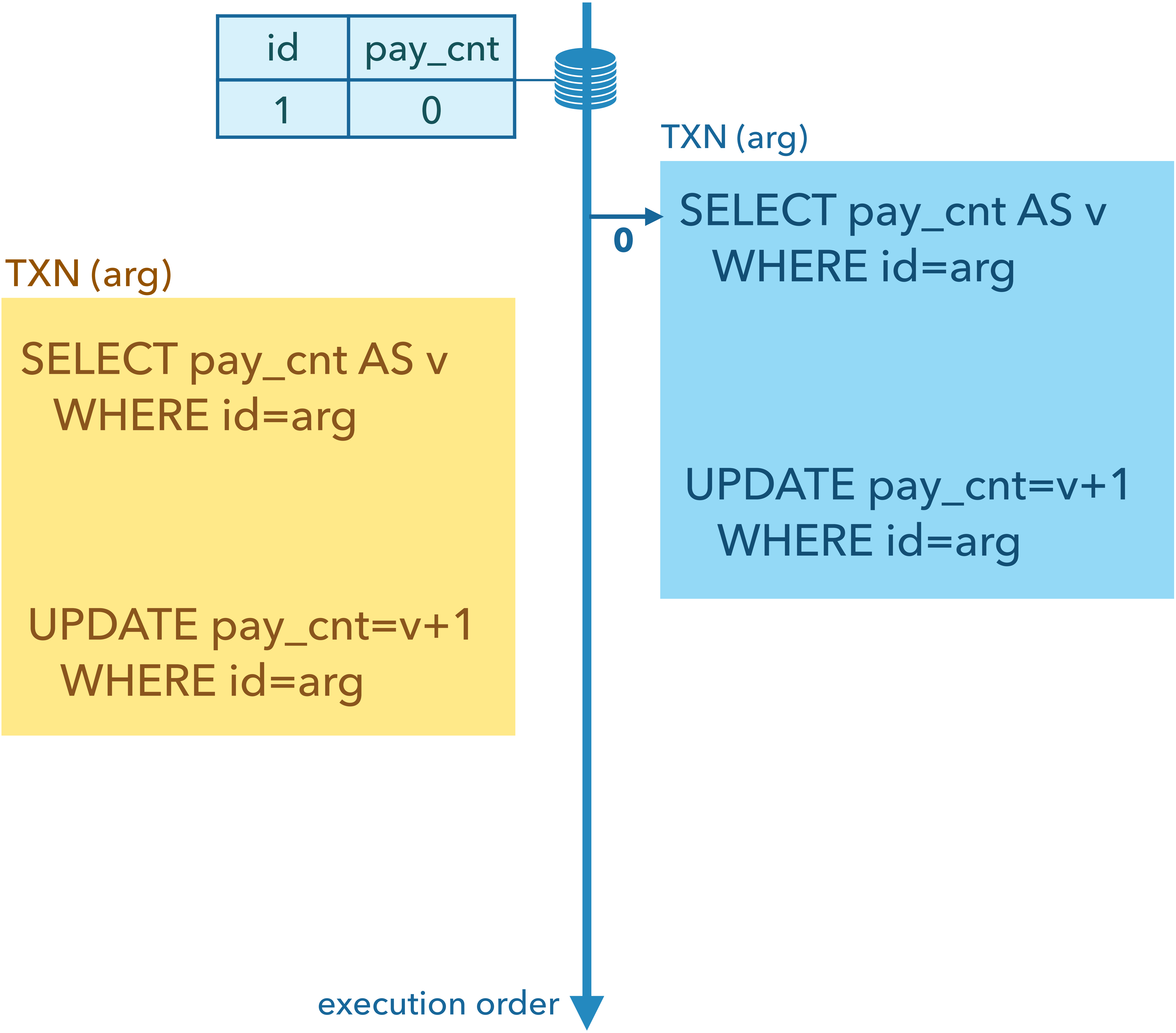
TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg

UPDATE pay_cnt=v+1
  WHERE id=arg
```

# EXAMPLE: A SERIALIZABILITY ANOMALY

- Unexpected behaviors can occur under weak guarantees



| id | pay_cnt |
|---|---|
| 1 | 0 |

TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg

UPDATE pay_cnt=v+1
  WHERE id=arg
```

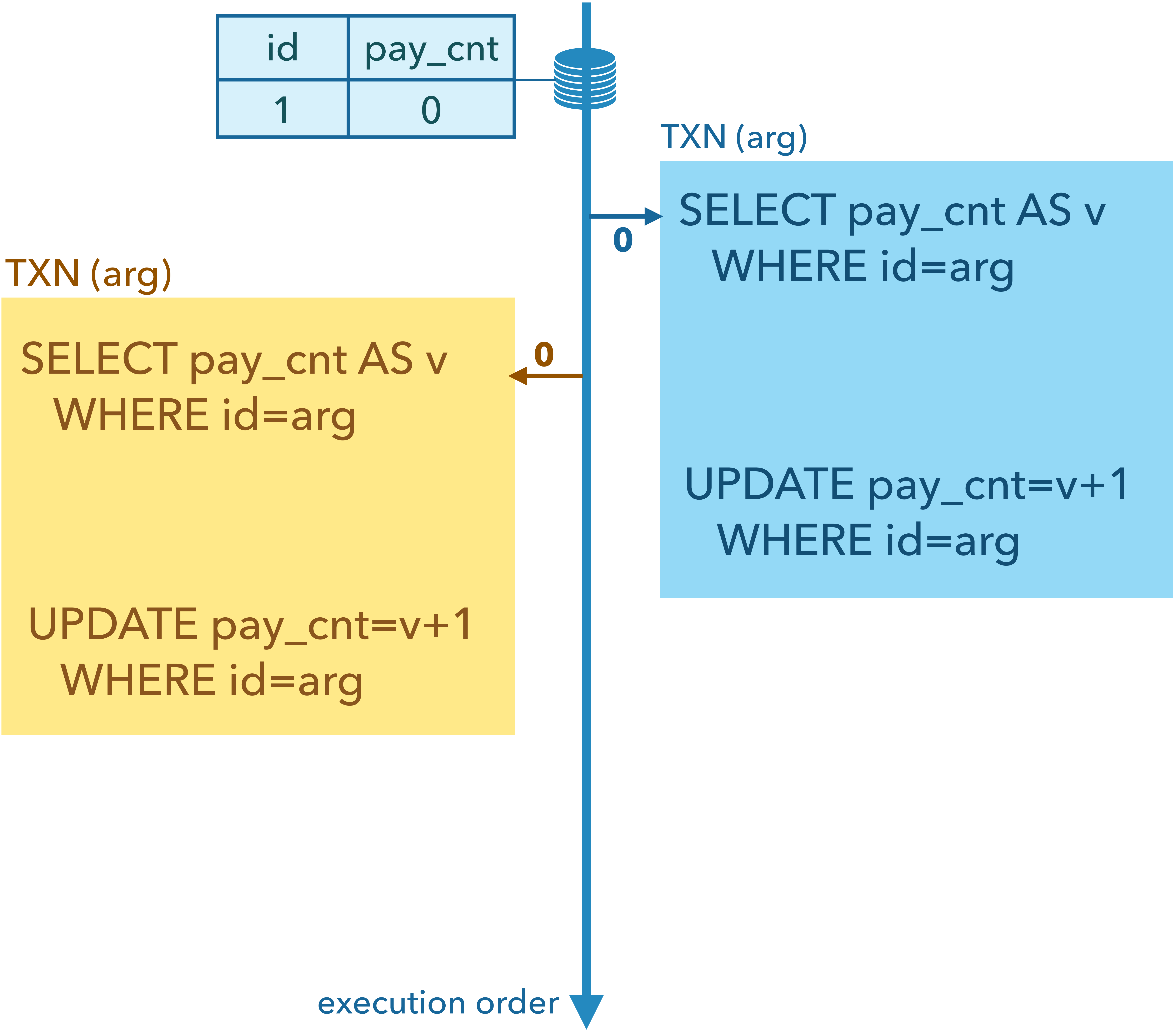
TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg

UPDATE pay_cnt=v+1
  WHERE id=arg
```

# EXAMPLE: A SERIALIZABILITY ANOMALY

- Unexpected behaviors can occur under weak guarantees



| id | pay_cnt |
|---|---|
| 1 | 0 |

TXN (arg)

SELECT pay_cnt AS v
WHERE id=arg

UPDATE pay_cnt=v+1
WHERE id=arg

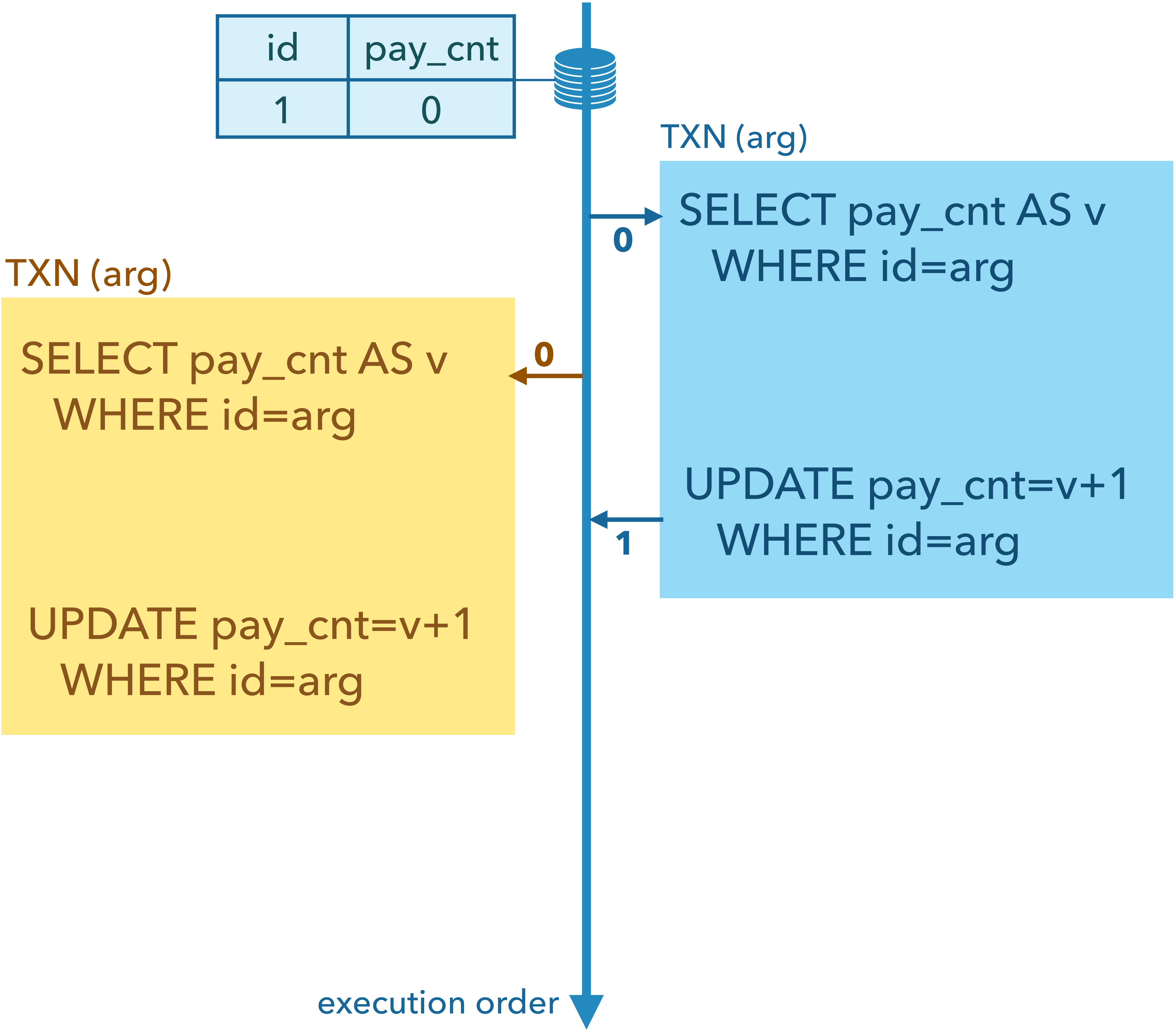
TXN (arg)

SELECT pay_cnt AS v
WHERE id=arg

UPDATE pay_cnt=v+1
WHERE id=arg

# EXAMPLE: A SERIALIZABILITY ANOMALY

- Unexpected behaviors can occur under weak guarantees



| id | pay_cnt |
|---|---|
| 1 | 0 |

TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg

UPDATE pay_cnt=v+1
  WHERE id=arg
```

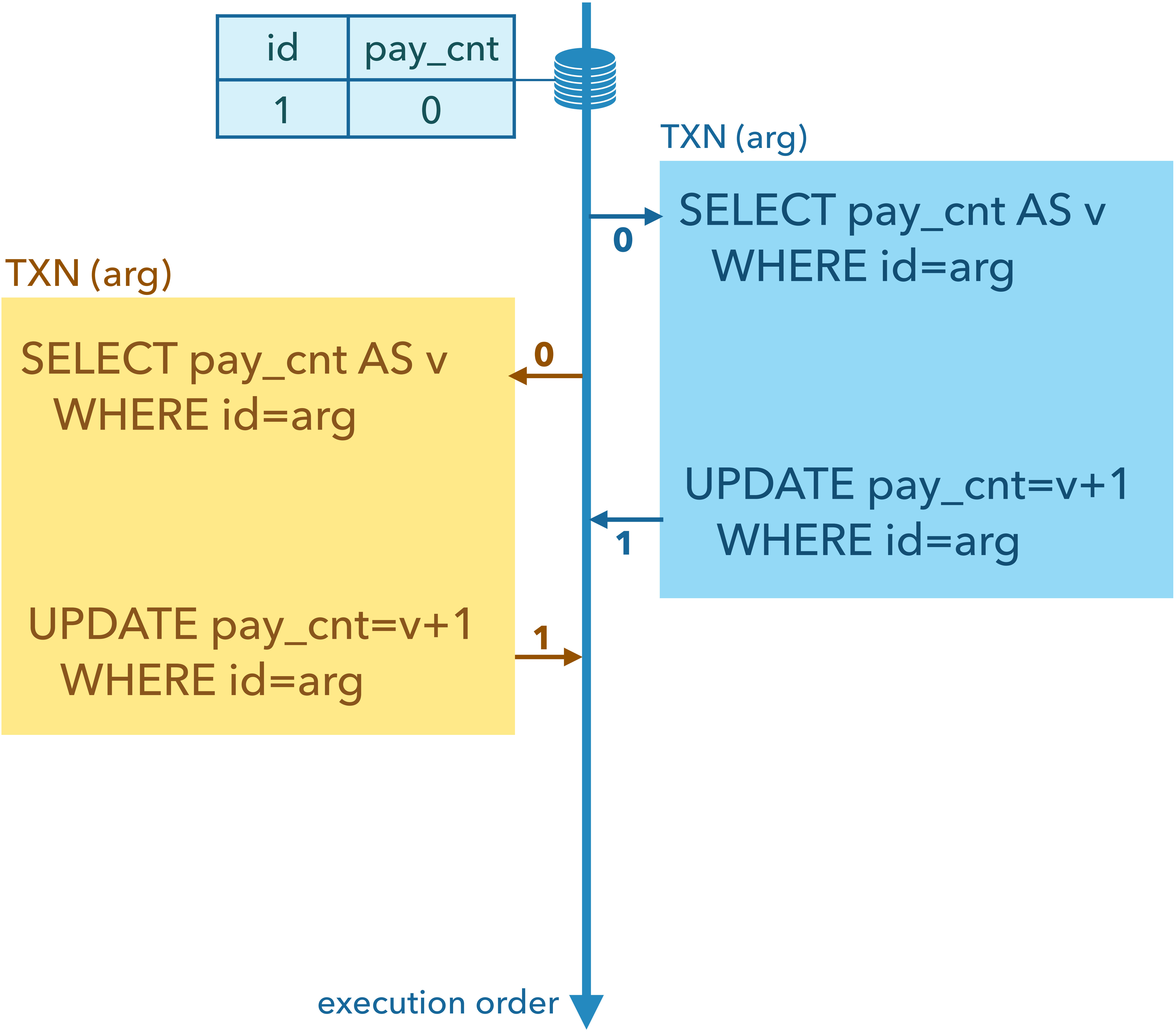
TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg

UPDATE pay_cnt=v+1
  WHERE id=arg
```

## EXAMPLE: A SERIALIZABILITY ANOMALY

- Unexpected behaviors can occur under weak guarantees



| id | pay_cnt |
|---|---|
| 1 | 0 |

TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg

UPDATE pay_cnt=v+1
  WHERE id=arg
```

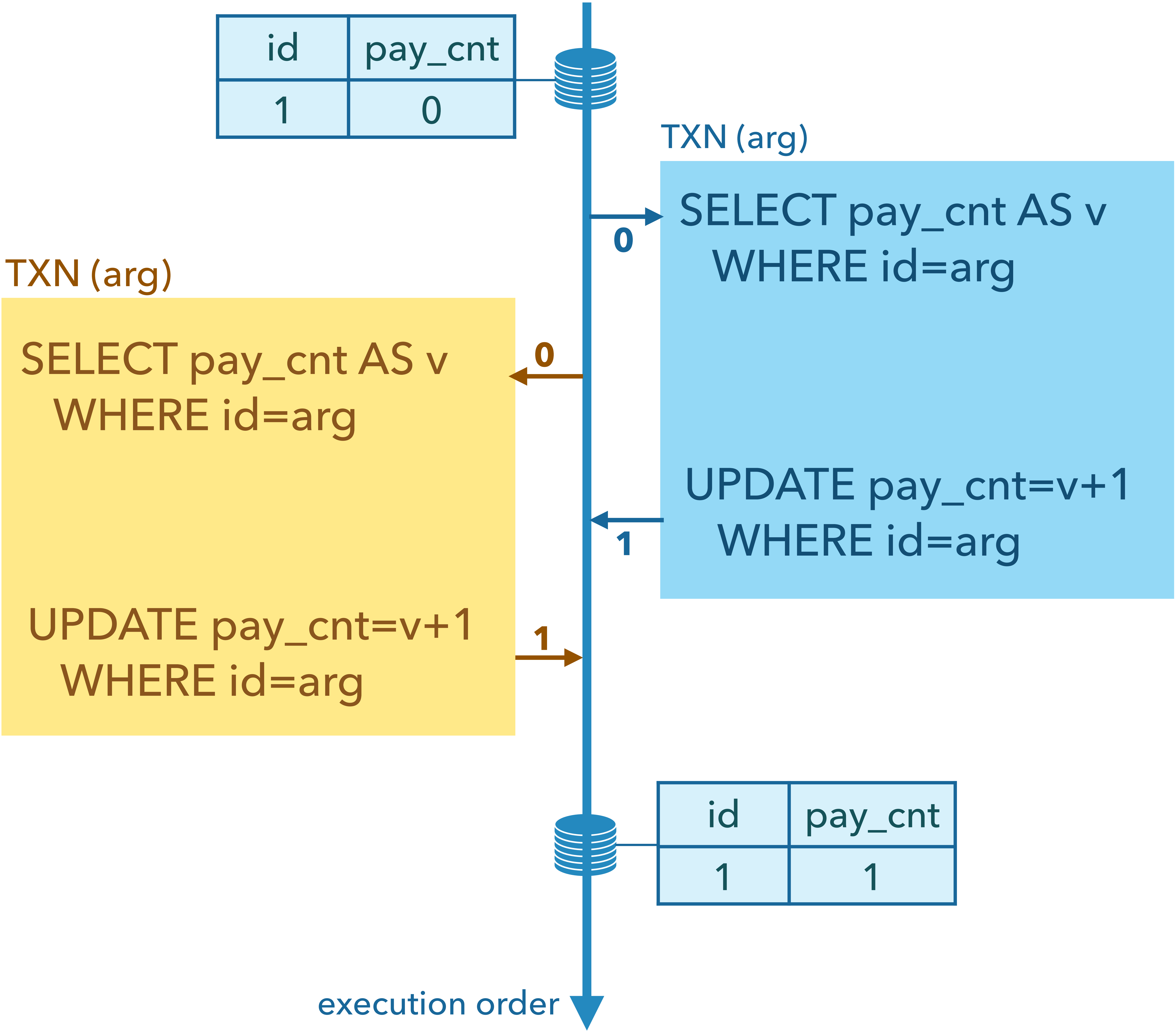
TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg

UPDATE pay_cnt=v+1
  WHERE id=arg
```

# EXAMPLE: A SERIALIZABILITY ANOMALY

- Unexpected behaviors can occur under weak guarantees



| id | pay_cnt |
|---|---|
| 1 | 0 |

TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg
UPDATE pay_cnt=v+1
  WHERE id=arg
```

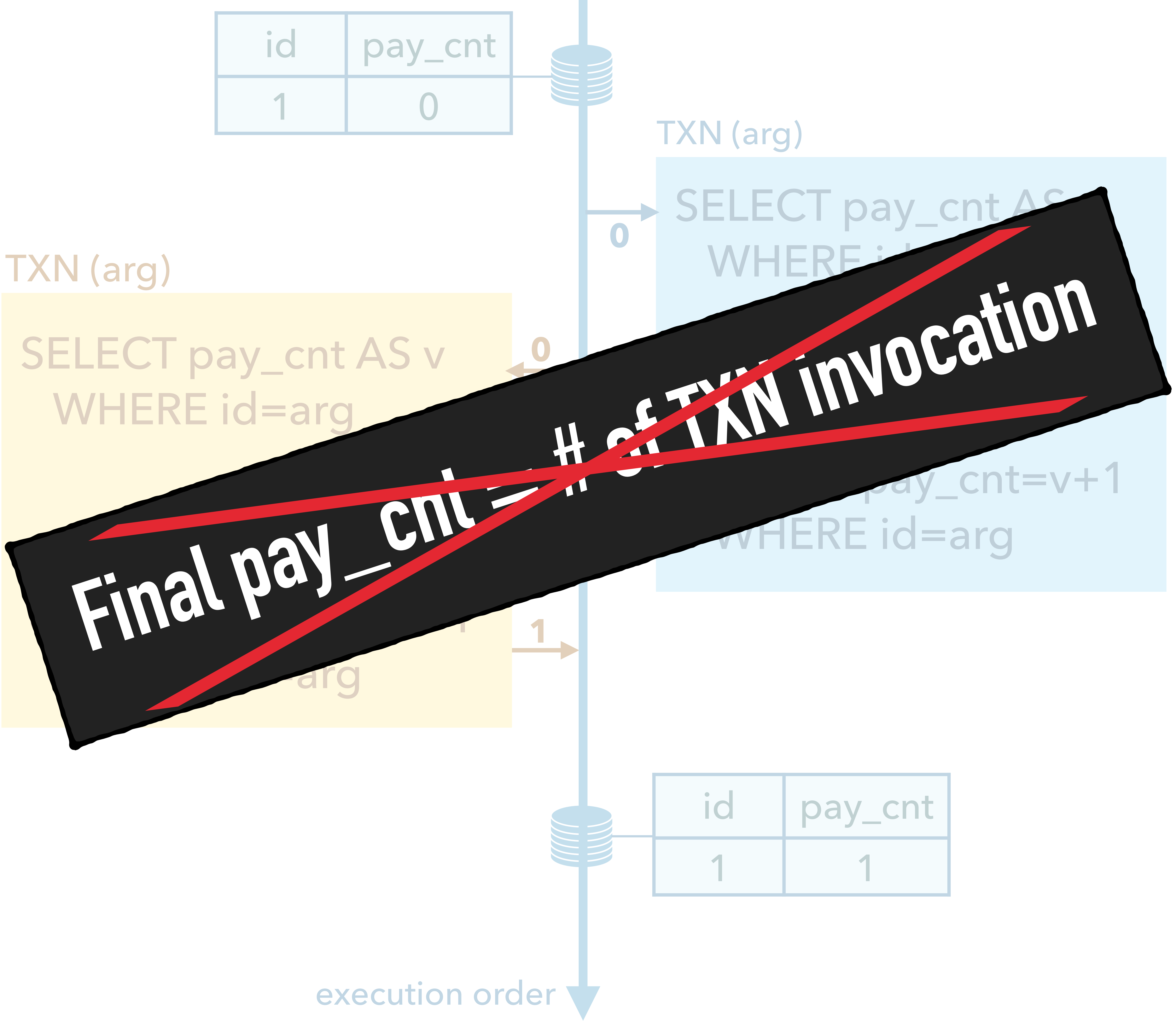
TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg
UPDATE pay_cnt=v+1
  WHERE id=arg
```

| id | pay_cnt |
|---|---|
| 1 | 1 |

# EXAMPLE: A SERIALIZABILITY ANOMALY

- Unexpected behaviors can occur under weak guarantees
- Assumed program invariants can be violated



| id | pay_cnt |
|---|---|
| 1 | 0 |

TXN (arg)

SELECT pay_cnt AS v
WHERE id=arg

TXN (arg)

SELECT pay_cnt AS
WHERE id=

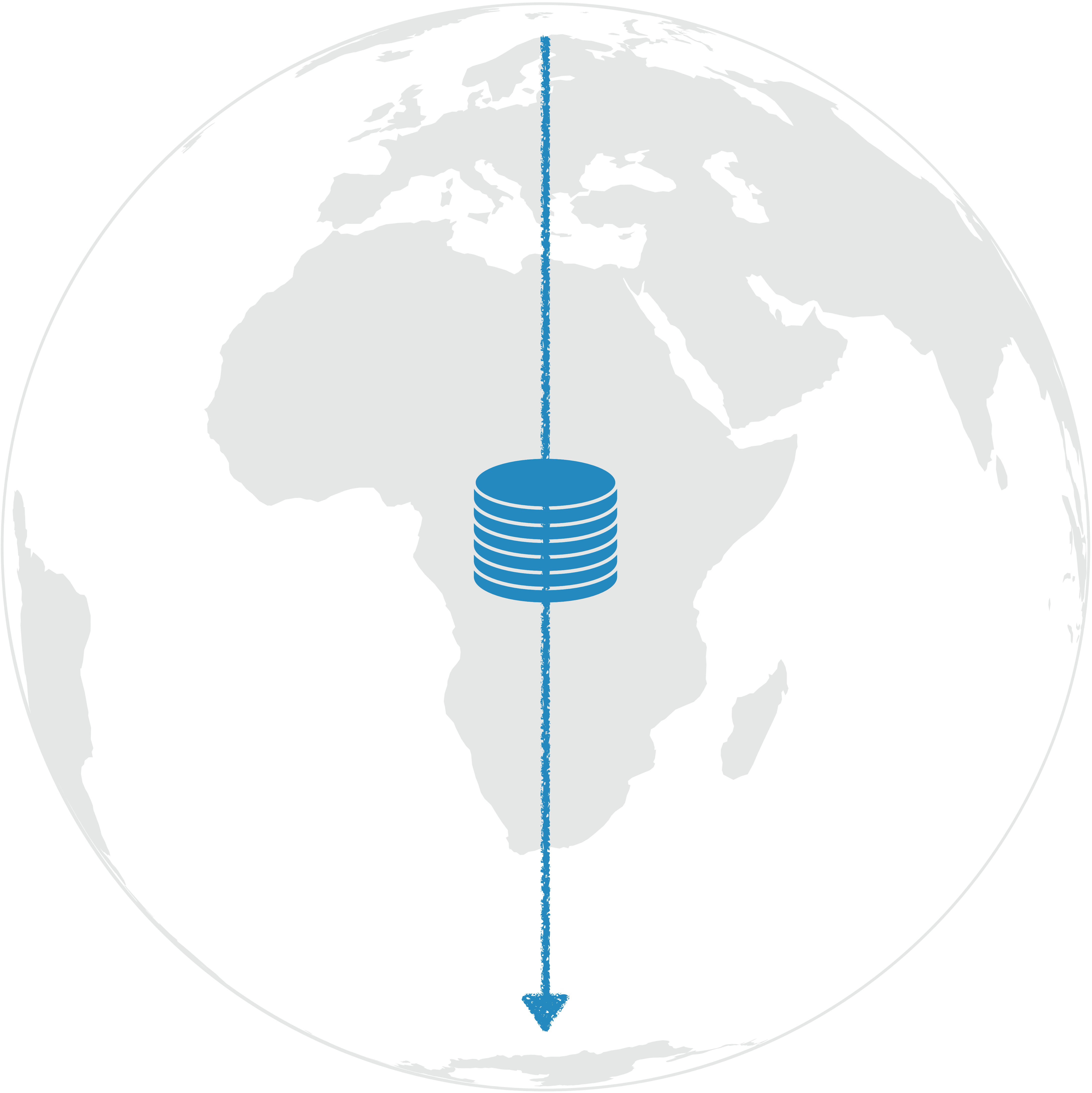
pay_cnt=v+1
WHERE id=arg

arg

| id | pay_cnt |
|---|---|
| 1 | 1 |

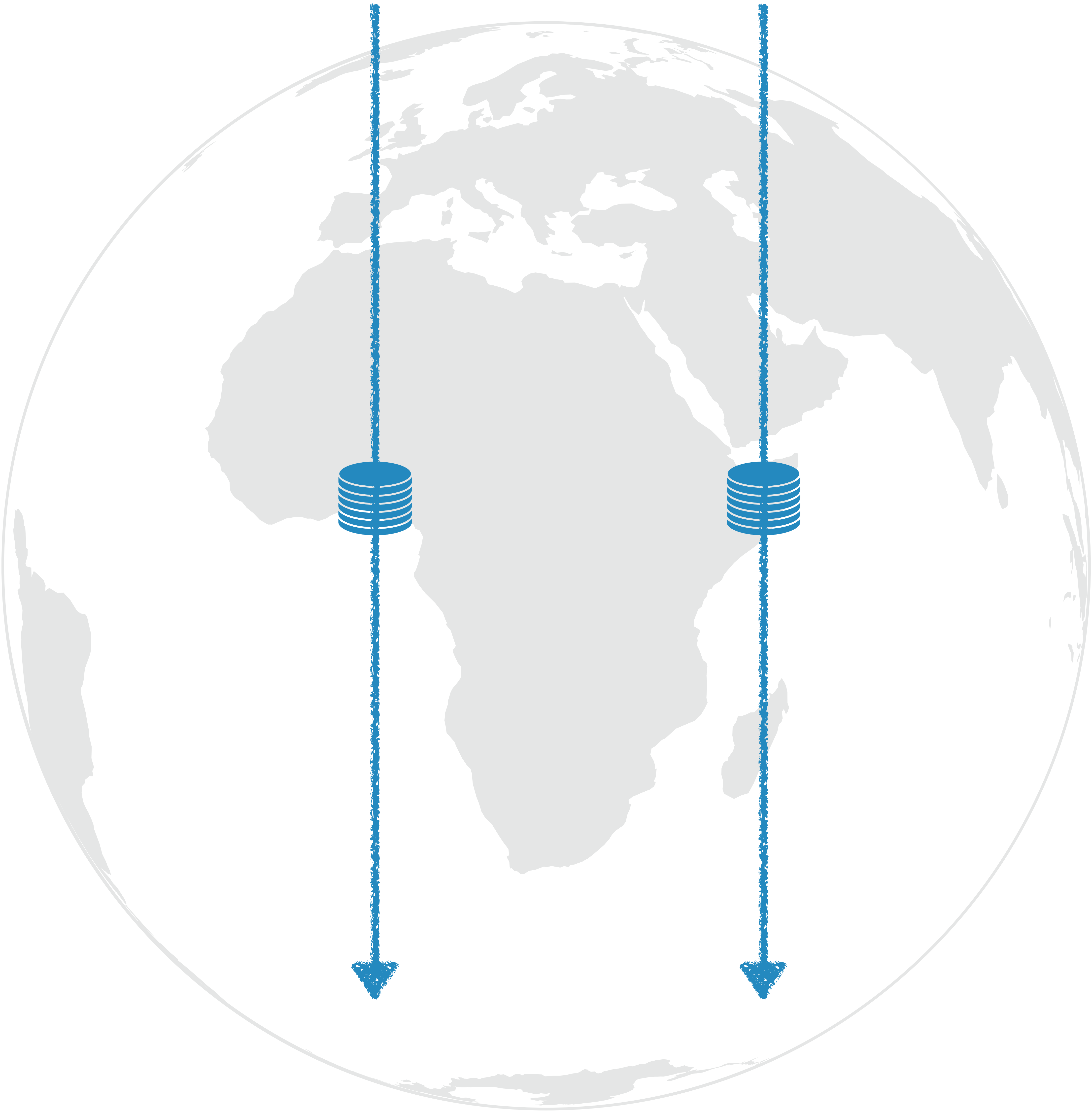
~~Final pay_cnt = # of TXN invocation~~

# WEAKLY CONSISTENT REPLICATED DATABASE SYSTEMS



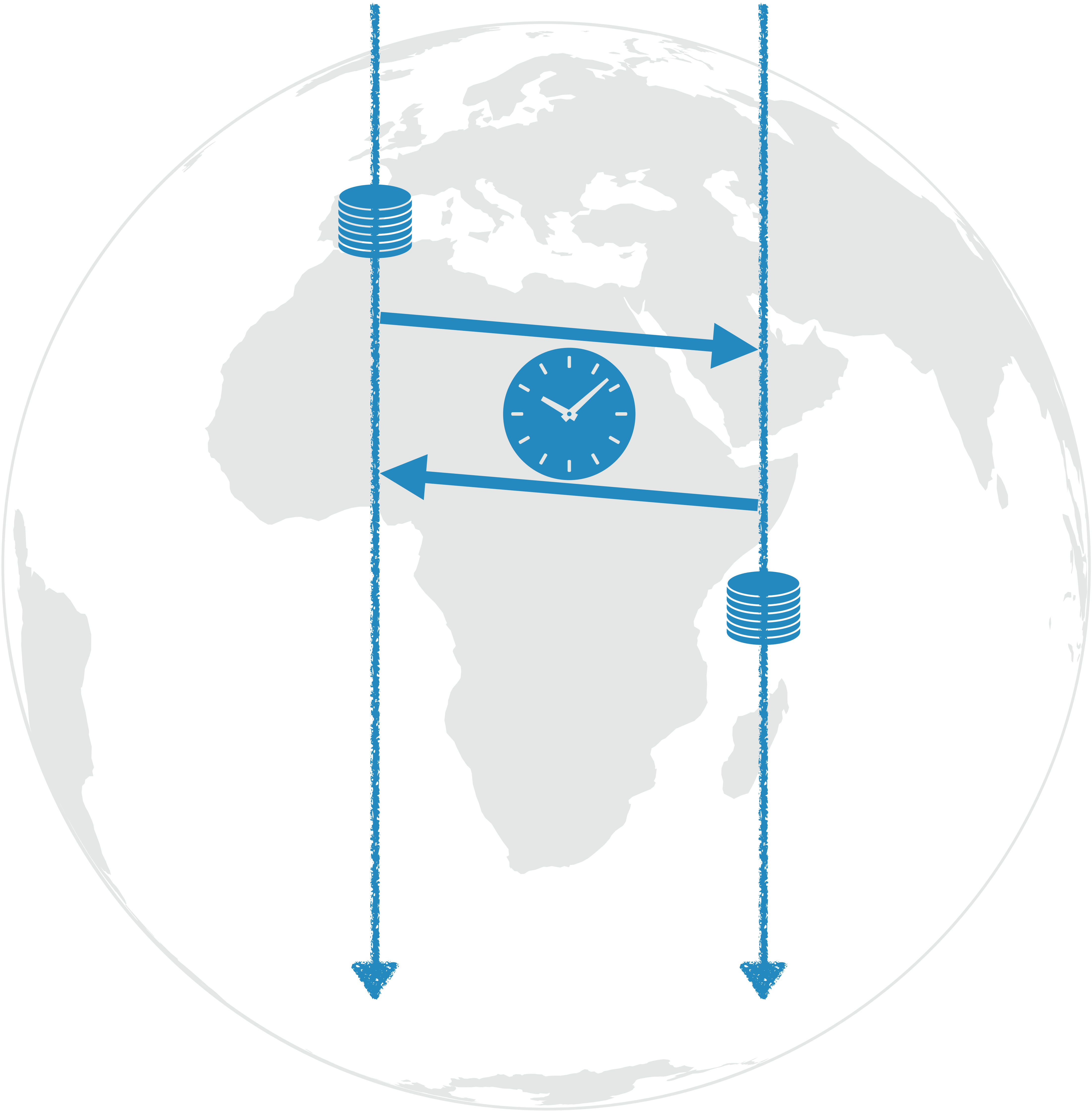
# WEAKLY CONSISTENT REPLICATED DATABASE SYSTEMS

- Data is geo-replicated in highly-available DBMSs



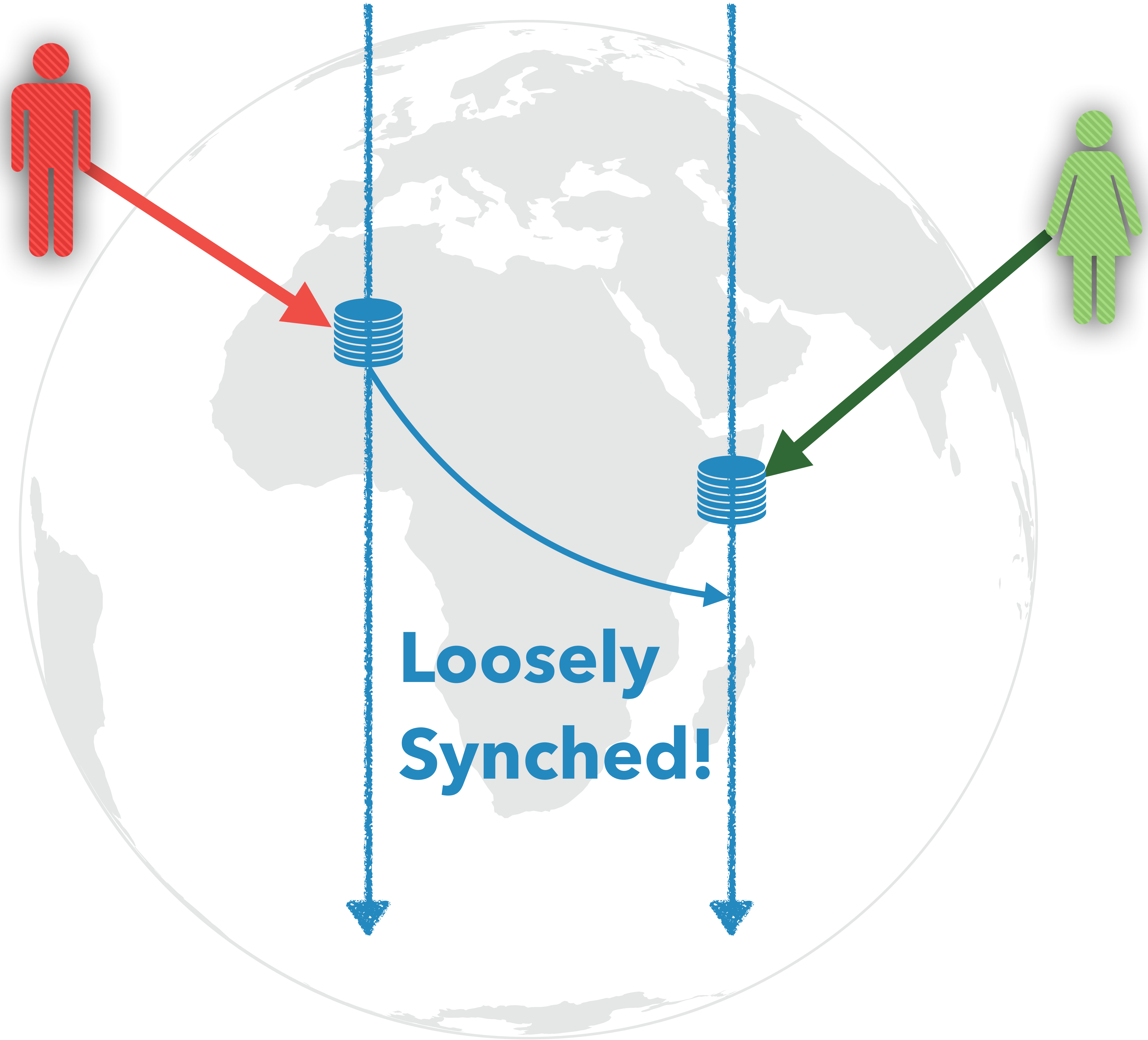
# WEAKLY CONSISTENT REPLICATED DATABASE SYSTEMS

- Data is geo-replicated in highly-available DBMSs
- Worldwide synchronization is **extremely** costly



# WEAKLY CONSISTENT REPLICATED DATABASE SYSTEMS

- Data is geo-replicated in highly-available DBMSs
- Worldwide synchronization is **extremely** costly
- Strongly synchronized data cannot be available



# WEAKLY CONSISTENT REPLICATED DATABASE SYSTEMS

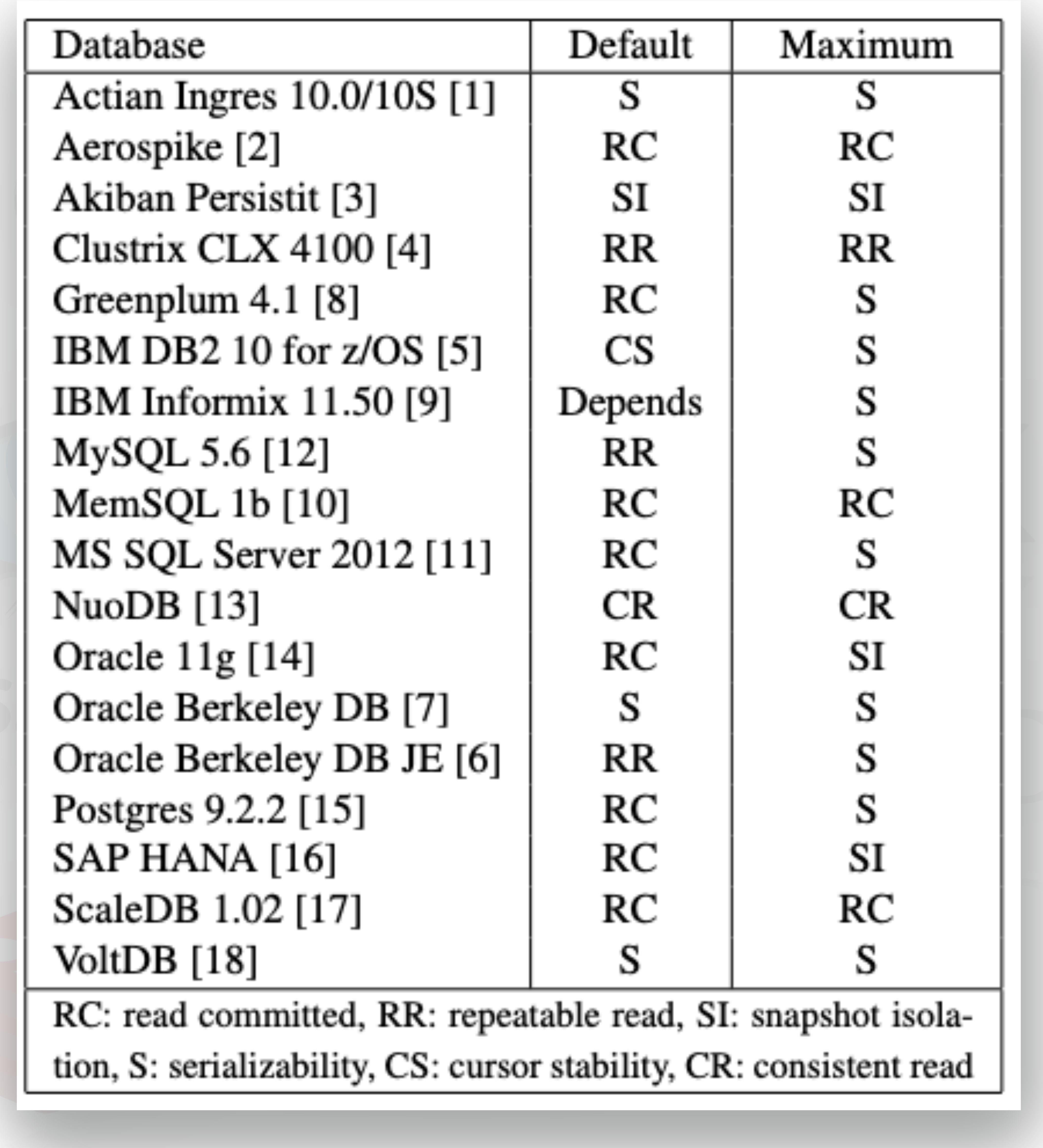
- Data is geo-replicated in highly-available DBMSs
- Worldwide synchronization is **extremely** costly
- Strongly synchronized data cannot be available

# WEAKLY CONSISTENT REPLICATED DATABASE SYSTEMS

- Data is geo-replicated in highly-available DBMSs
- Worldwide synchronization is **extremely** costly
- Strongly synchronized data cannot be available
- Weak consistency semantics are very popular

# WEAKLY CONSISTENT REPLICATED DATABASE SYSTEMS

- Data is geo-replicated in highly-available DBMSs
- Worldwide synchronization is **extremely** costly
- Strongly synchronized data cannot be available

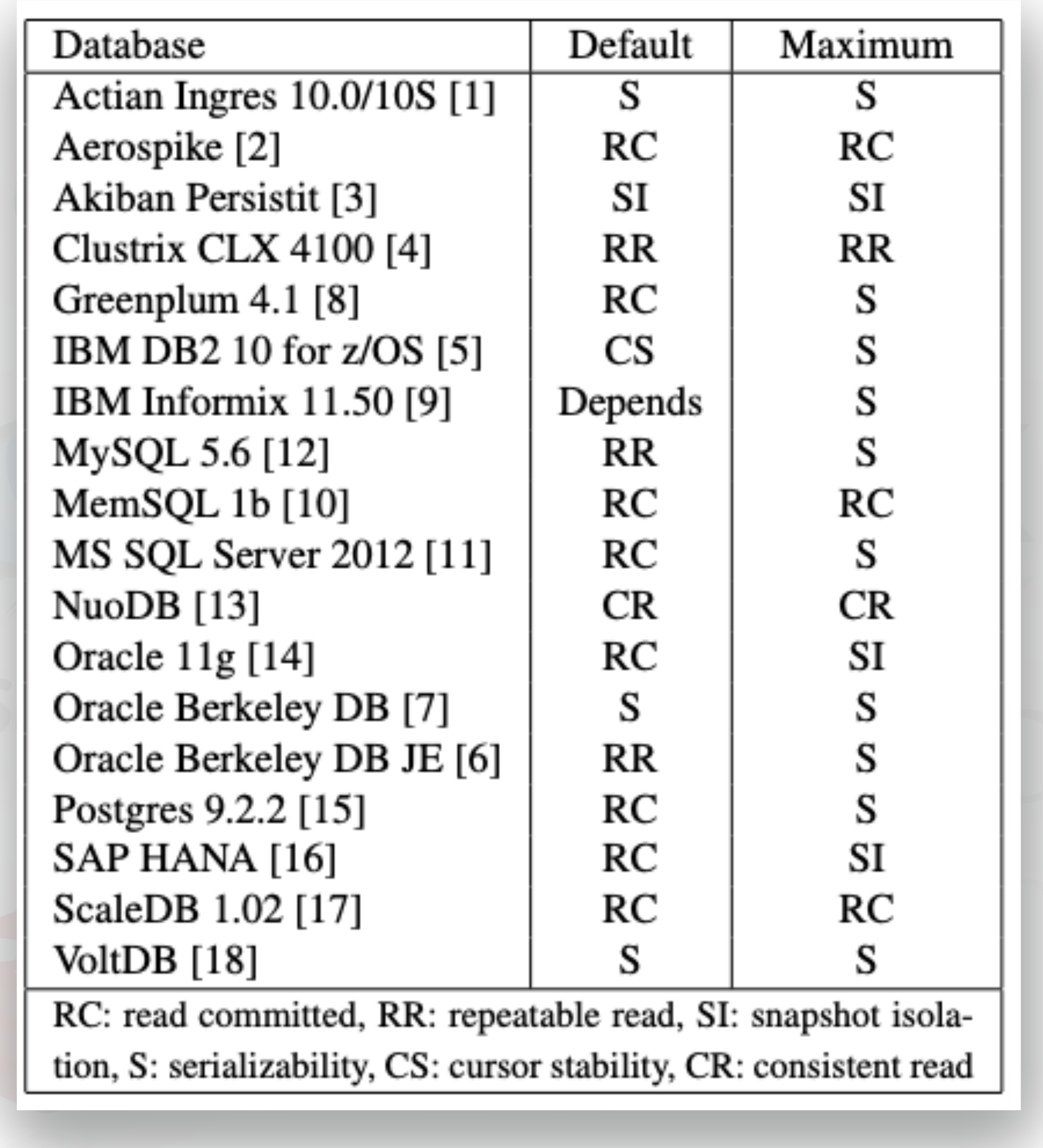


| Database | Default | Maximum |
|---|---|---|
| Actian Ingres 10.0/10S [1] | S | S |
| Aerospike [2] | RC | RC |
| Akiban Persistit [3] | SI | SI |
| Clustrix CLX 4100 [4] | RR | RR |
| Greenplum 4.1 [8] | RC | S |
| IBM DB2 10 for z/OS [5] | CS | S |
| IBM Informix 11.50 [9] | Depends | S |
| MySQL 5.6 [12] | RR | S |
| MemSQL 1b [10] | RC | RC |
| MS SQL Server 2012 [11] | RC | S |
| NuoDB [13] | CR | CR |
| Oracle 11g [14] | RC | SI |
| Oracle Berkeley DB [7] | S | S |
| Oracle Berkeley DB JE [6] | RR | S |
| Postgres 9.2.2 [15] | RC | S |
| SAP HANA [16] | RC | SI |
| ScaleDB 1.02 [17] | RC | RC |
| VoltDB [18] | S | S |

RC: read committed, RR: repeatable read, SI: snapshot isolation, S: serializability, CS: cursor stability, CR: consistent read

## WEAKLY CONSISTENT REPLICATED DATABASE SYSTEMS

- Data is geo-replicated in highly-available DBMSs
- Worldwide synchronization is **extremely** costly
- Strongly synchronized data cannot be available
- Weak consistency semantics are very popular
- Serializabiliabity is rarely assumed by default [Bailis et.al]



| Database | Default | Maximum |
|---|---|---|
| Actian Ingres 10.0/10S [1] | S | S |
| Aerospike [2] | RC | RC |
| Akiban Persistit [3] | SI | SI |
| Clustrix CLX 4100 [4] | RR | RR |
| Greenplum 4.1 [8] | RC | S |
| IBM DB2 10 for z/OS [5] | CS | S |
| IBM Informix 11.50 [9] | Depends | S |
| MySQL 5.6 [12] | RR | S |
| MemSQL 1b [10] | RC | RC |
| MS SQL Server 2012 [11] | RC | S |
| NuoDB [13] | CR | CR |
| Oracle 11g [14] | RC | SI |
| Oracle Berkeley DB [7] | S | S |
| Oracle Berkeley DB JE [6] | RR | S |
| Postgres 9.2.2 [15] | RC | S |
| SAP HANA [16] | RC | SI |
| ScaleDB 1.02 [17] | RC | RC |
| VoltDB [18] | S | S |

RC: read committed, RR: repeatable read, SI: snapshot isolation, S: serializability, CS: cursor stability, CR: consistent read

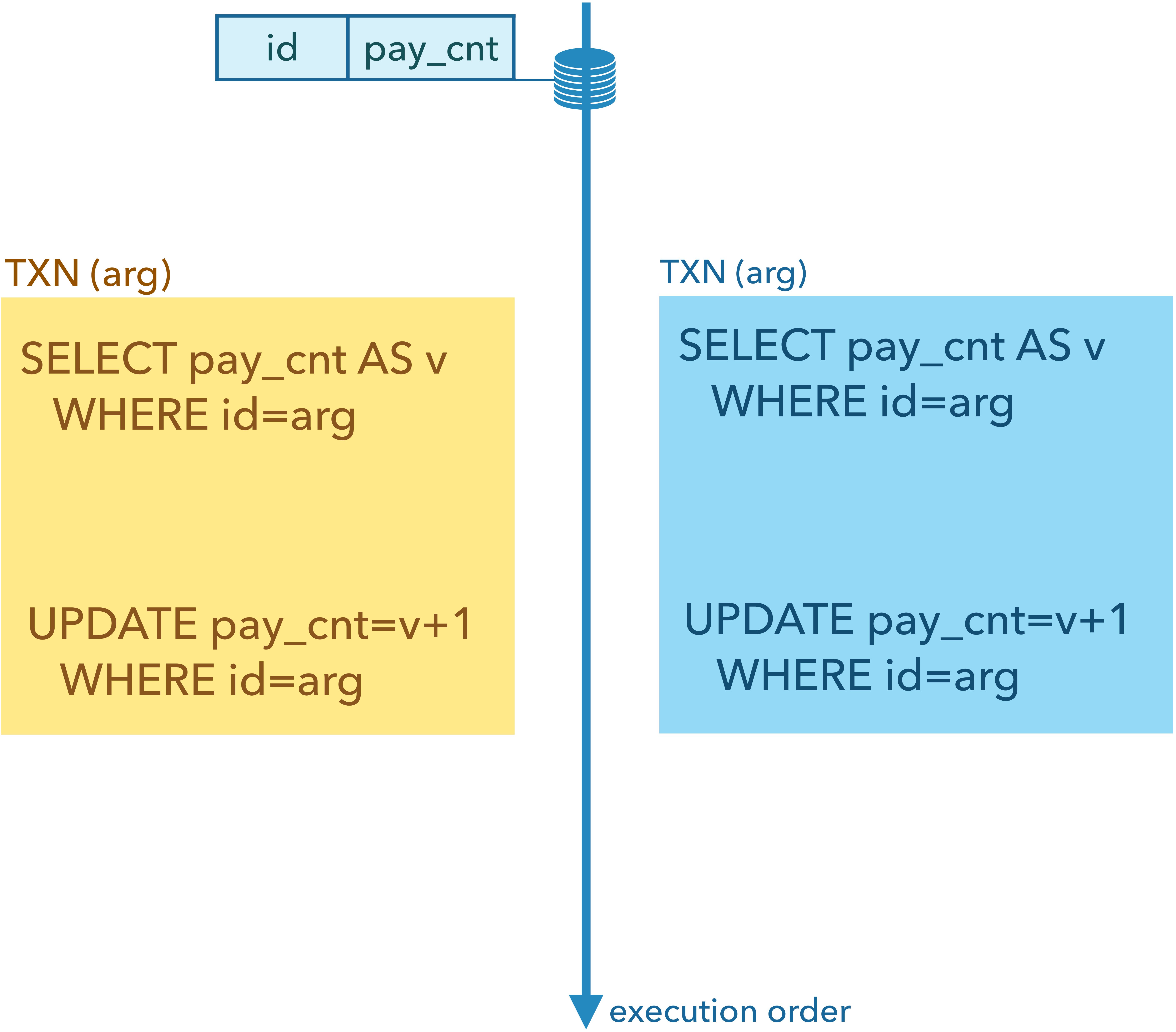
# TESTING: FUNDAMENTAL CHALLENGES

# TESTING: FUNDAMENTAL CHALLENGES

- Triggering anomalies requires determining many parameters

# TESTING: FUNDAMENTAL CHALLENGES

- Triggering anomalies requires determining many parameters



| id | pay_cnt |
|---|---|

TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg

UPDATE pay_cnt=v+1
  WHERE id=arg
```

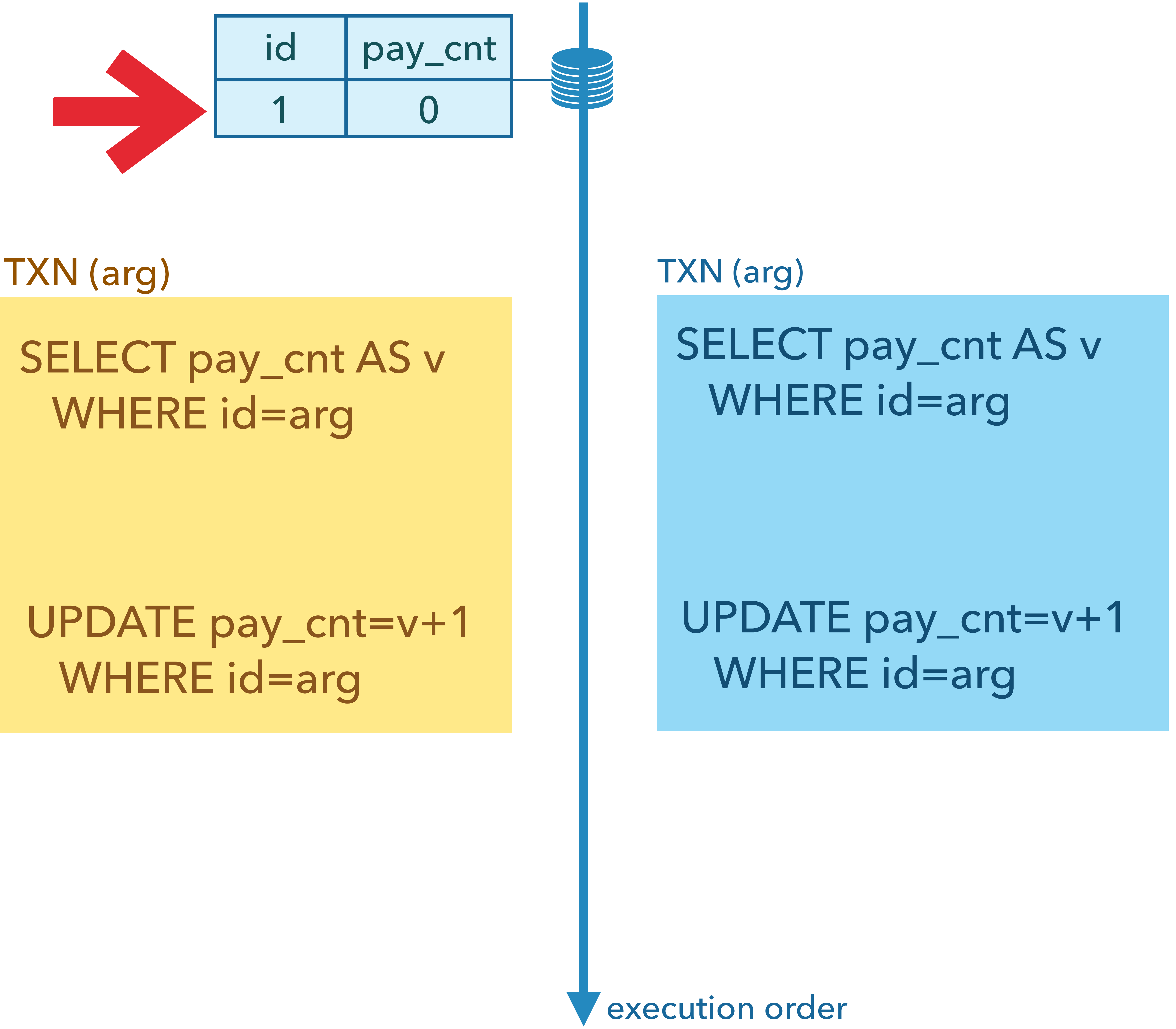
TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg

UPDATE pay_cnt=v+1
  WHERE id=arg
```

# TESTING: FUNDAMENTAL CHALLENGES

- Triggering anomalies requires determining many parameters
  - Initial database state



| id | pay_cnt |
|---|---|
| 1 | 0 |

TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg

UPDATE pay_cnt=v+1
  WHERE id=arg
```

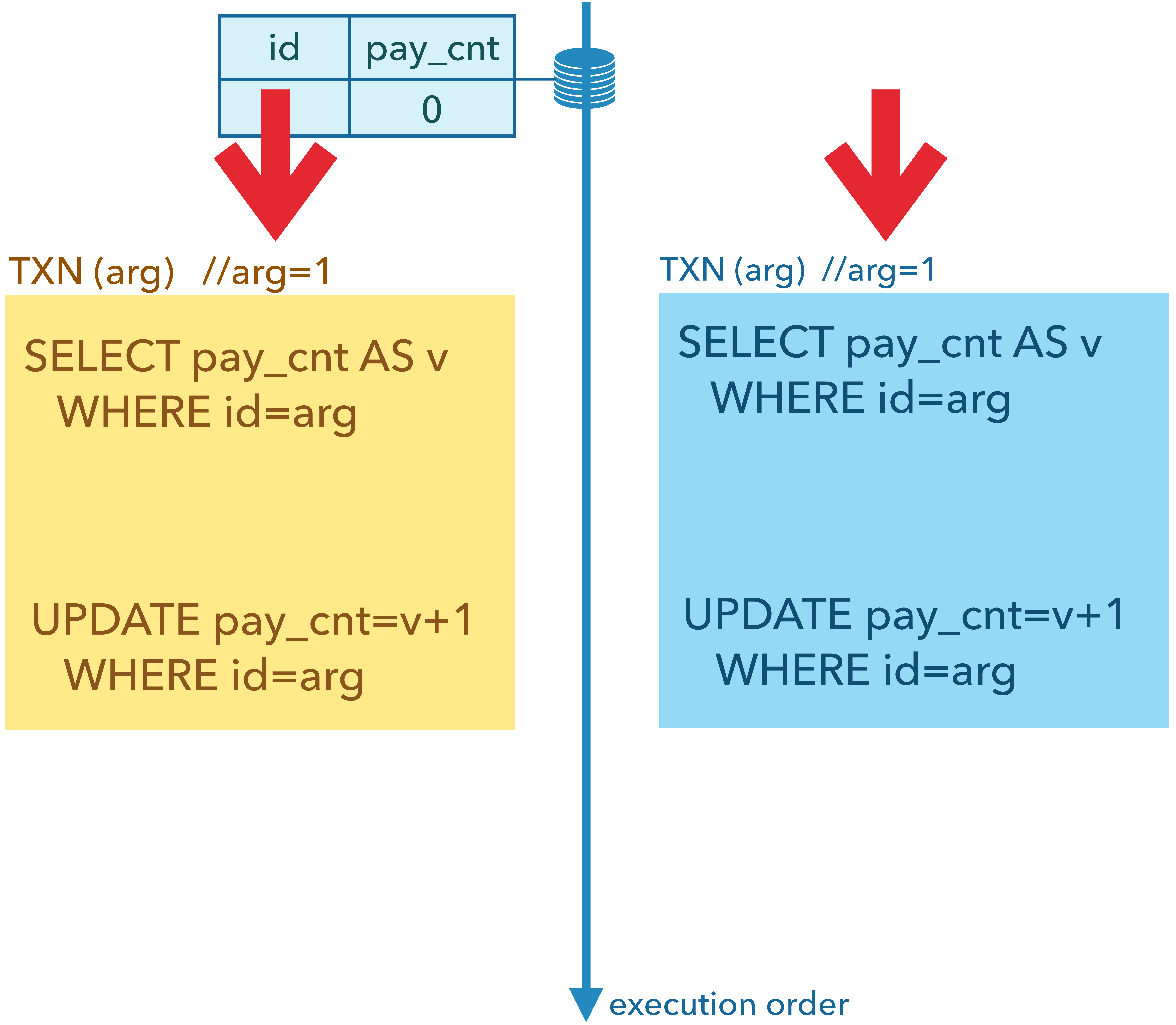
TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg

UPDATE pay_cnt=v+1
  WHERE id=arg
```

# TESTING: FUNDAMENTAL CHALLENGES

- Triggering anomalies requires determining many parameters
  - Initial database state
  - Input arguments



| id | pay_cnt |
|---|---|
| | 0 |

TXN (arg) //arg=1

```
SELECT pay_cnt AS v
  WHERE id=arg

UPDATE pay_cnt=v+1
  WHERE id=arg
```

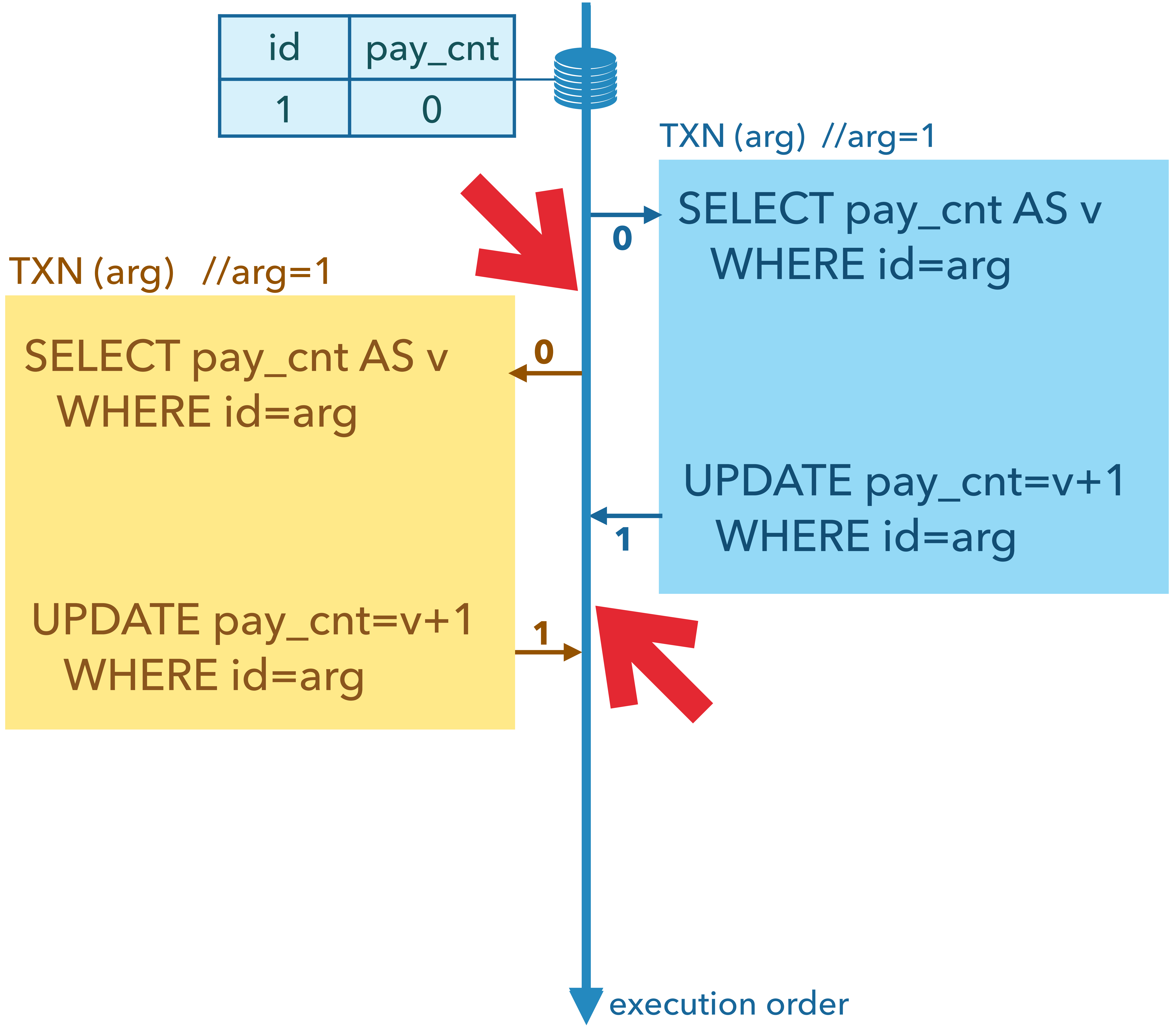
TXN (arg) //arg=1

```
SELECT pay_cnt AS v
  WHERE id=arg

UPDATE pay_cnt=v+1
  WHERE id=arg
```

# TESTING: FUNDAMENTAL CHALLENGES

- Triggering anomalies requires determining many parameters
  - Initial database state
  - Input arguments
  - Execution order



| id | pay_cnt |
|---|---|
| 1 | 0 |

TXN (arg) //arg=1

```
SELECT pay_cnt AS v
  WHERE id=arg

UPDATE pay_cnt=v+1
  WHERE id=arg
```

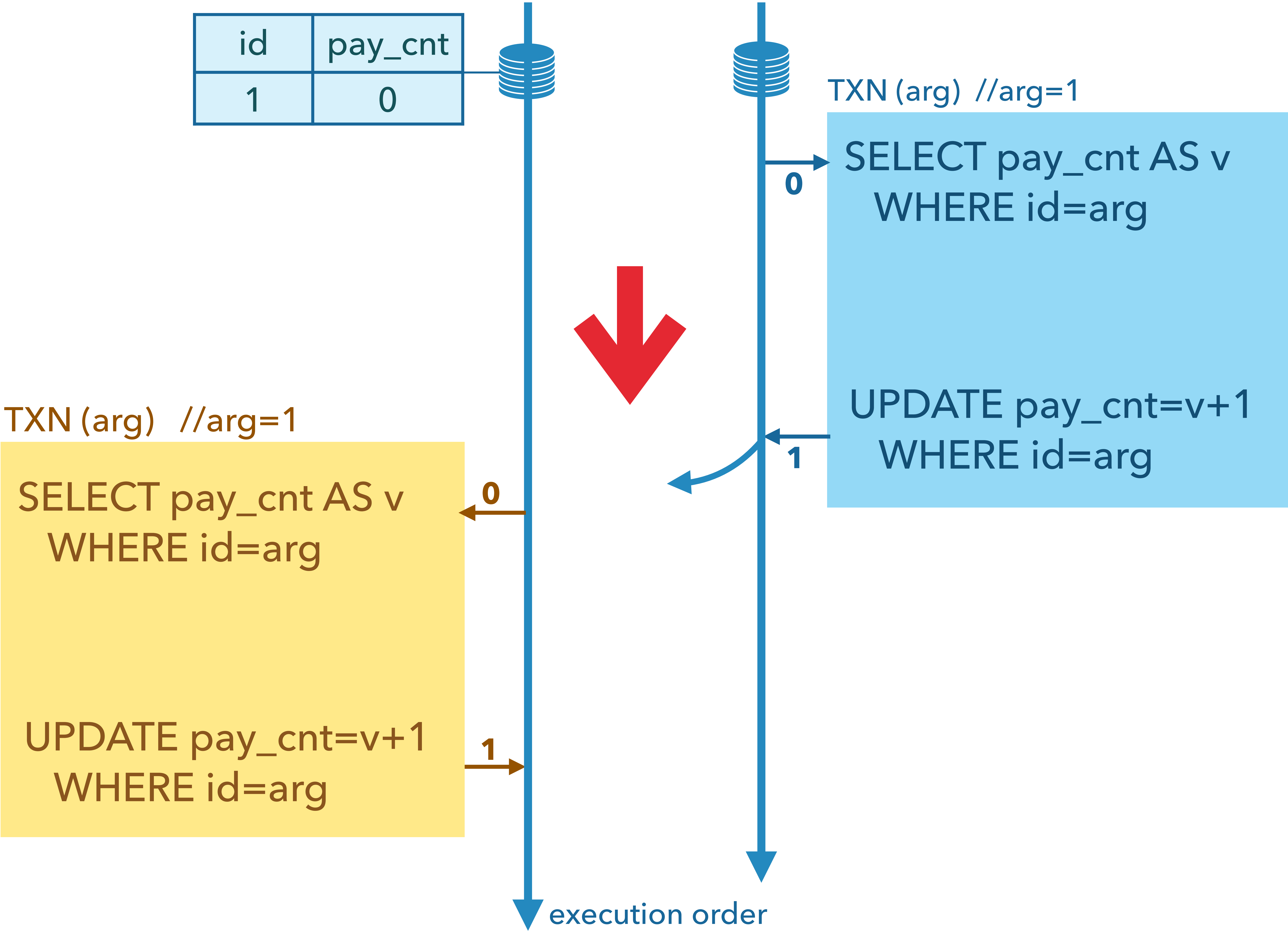
TXN (arg) //arg=1

```
SELECT pay_cnt AS v
  WHERE id=arg

UPDATE pay_cnt=v+1
  WHERE id=arg
```

# TESTING: FUNDAMENTAL CHALLENGES

- Triggering anomalies requires determining many parameters
  - Initial database state
  - Input arguments
  - Execution order
  - Network delays



| id | pay_cnt |
|---|---|
| 1 | 0 |

TXN (arg) //arg=1

```
SELECT pay_cnt AS v
  WHERE id=arg

UPDATE pay_cnt=v+1
  WHERE id=arg
```

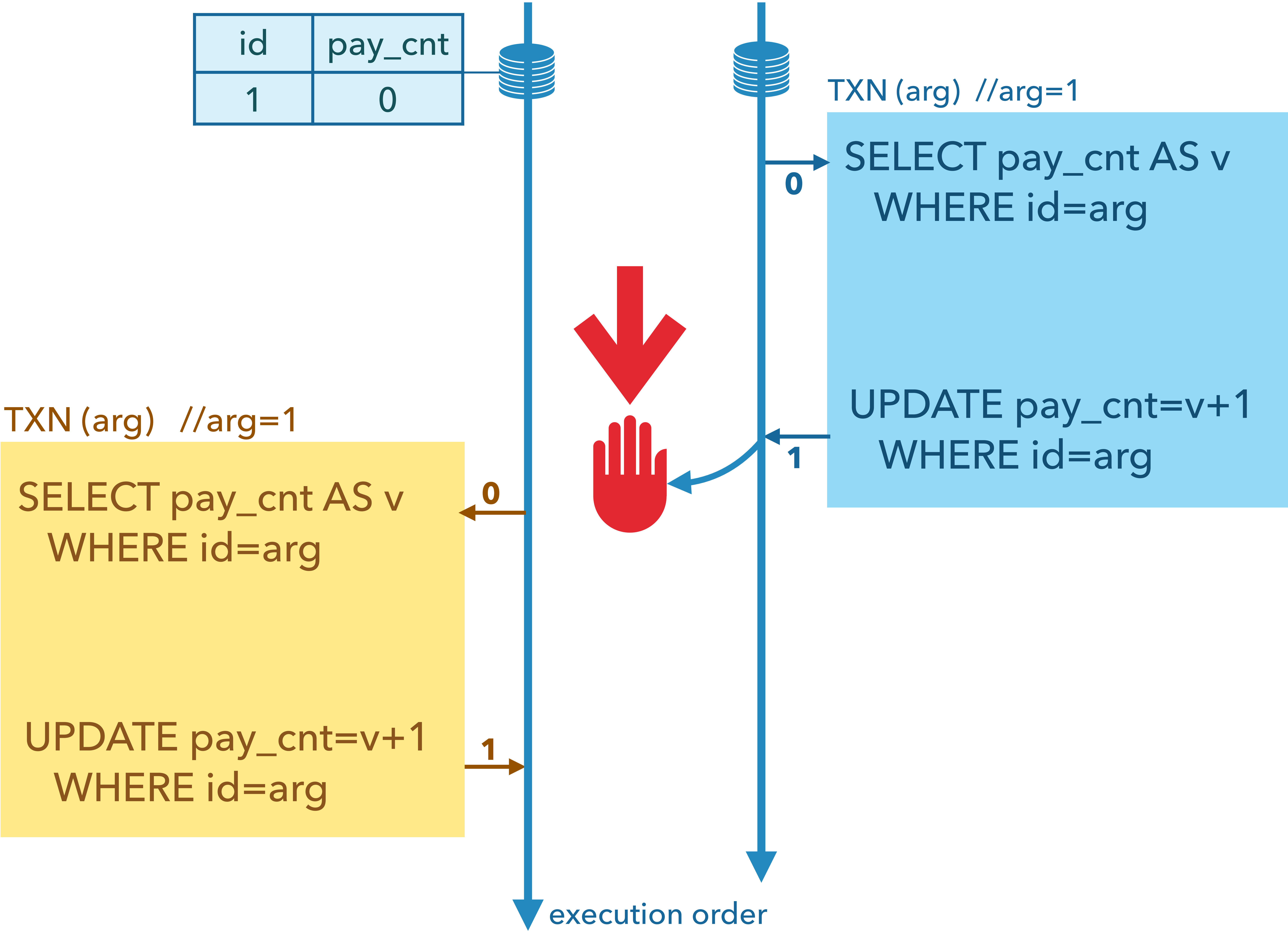
TXN (arg) //arg=1

```
SELECT pay_cnt AS v
  WHERE id=arg

UPDATE pay_cnt=v+1
  WHERE id=arg
```

# TESTING: FUNDAMENTAL CHALLENGES

- Triggering anomalies requires determining many parameters
  - Initial database state
  - Input arguments
  - Execution order
  - Network delays



| id | pay_cnt |
|---|---|
| 1 | 0 |

TXN (arg) //arg=1

```
SELECT pay_cnt AS v
  WHERE id=arg

UPDATE pay_cnt=v+1
  WHERE id=arg
```

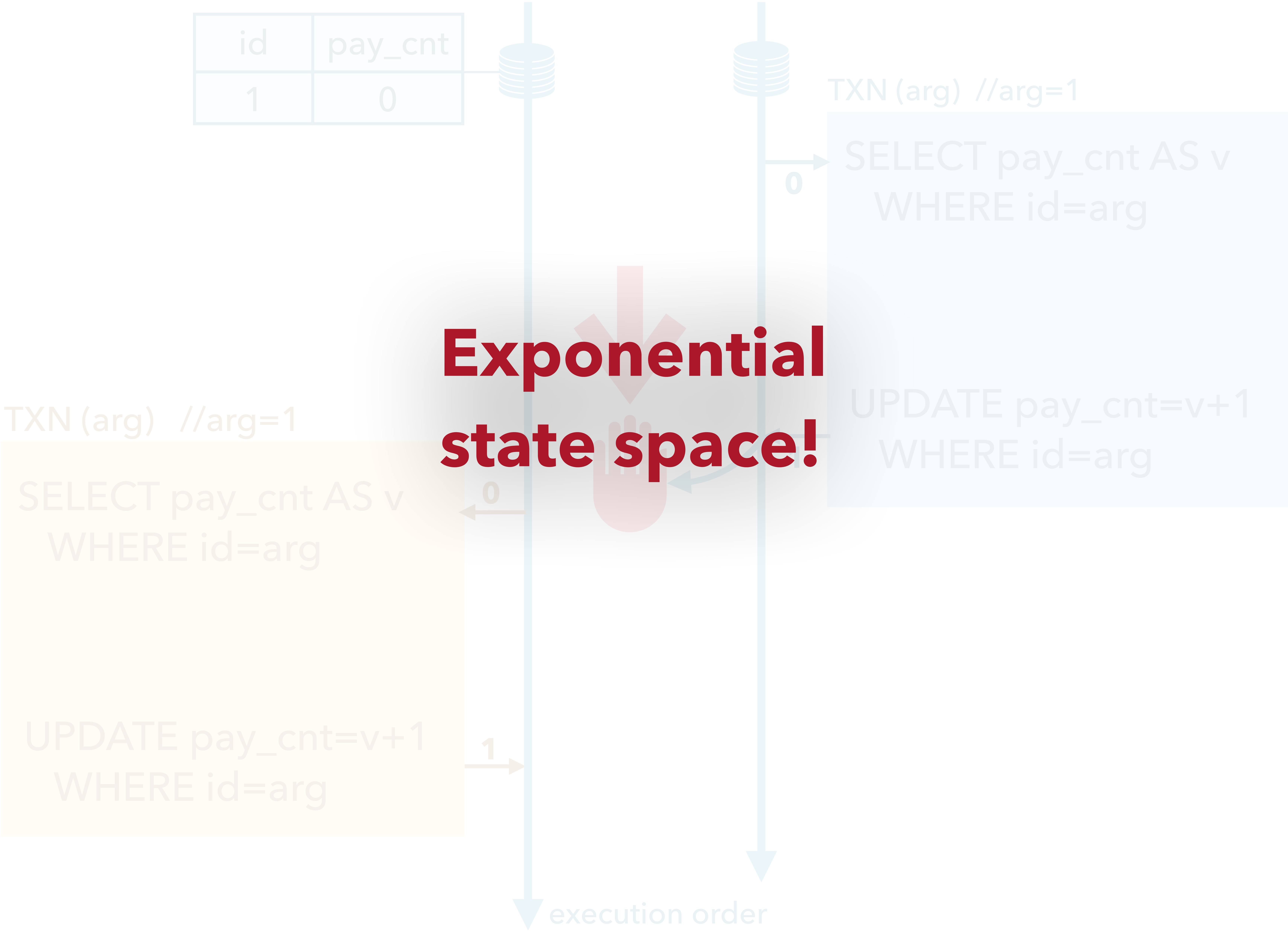
TXN (arg) //arg=1

```
SELECT pay_cnt AS v
  WHERE id=arg

UPDATE pay_cnt=v+1
  WHERE id=arg
```

## TESTING: FUNDAMENTAL CHALLENGES

- Triggering anomalies requires determining many parameters
  - Initial database state
  - Input arguments
  - Execution order
  - Network delays



**Exponential state space!**

# BLACKBOX TESTING

# BLACKBOX TESTING

- Independent of application semantics

HOTEL RESERVATION

BANKING

ONLINE SHOP

# BLACKBOX TESTING

- Independent of application semantics
- Independent of database specific guarantees

HOTEL RESERVATION

BANKING

ONLINE SHOP

POSTGRESQL

CALVIN

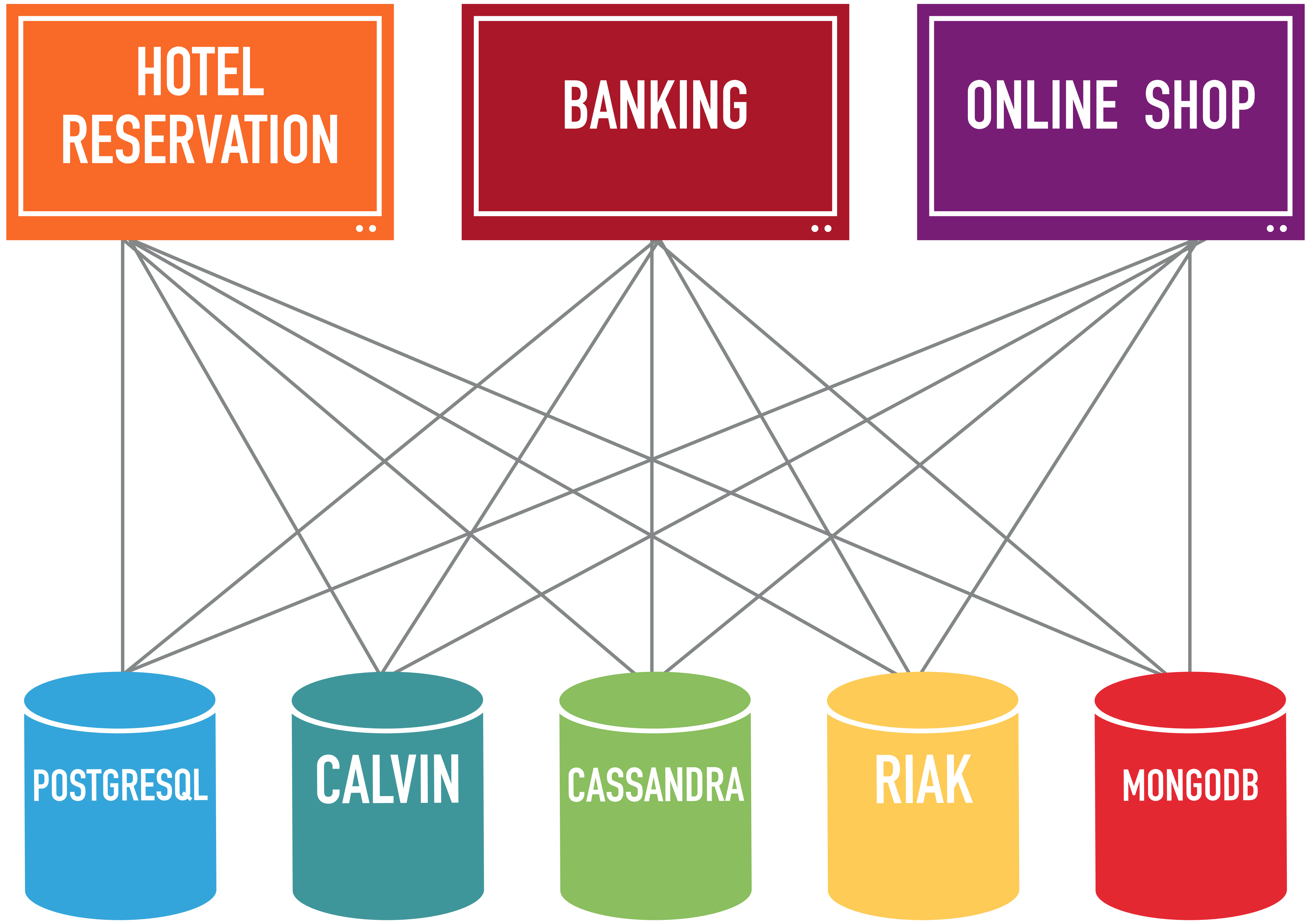
CASSANDRA

RIAK

MONGODB

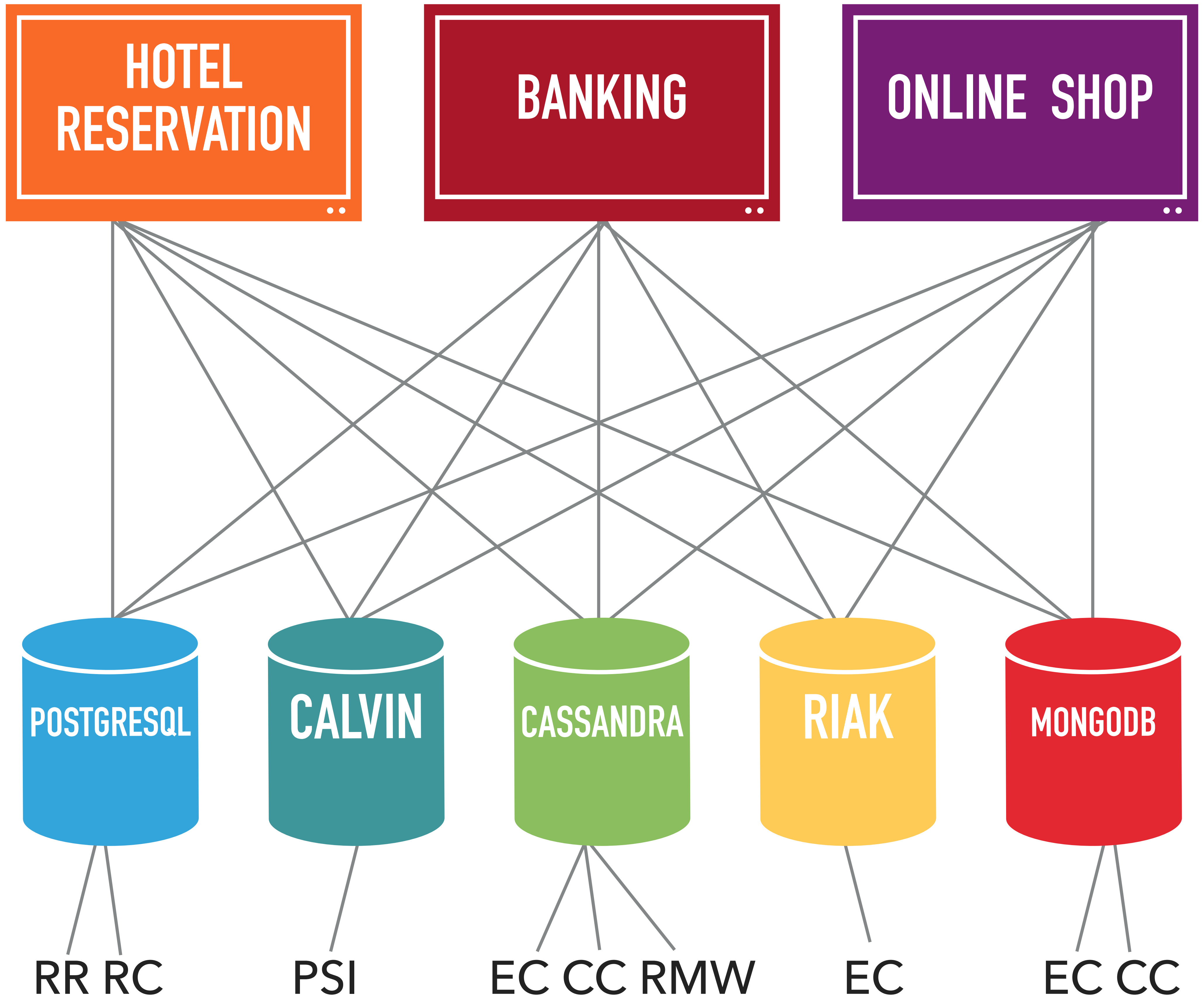
# BLACKBOX TESTING

- Independent of application semantics
- Independent of database specific guarantees
- Not reproducible



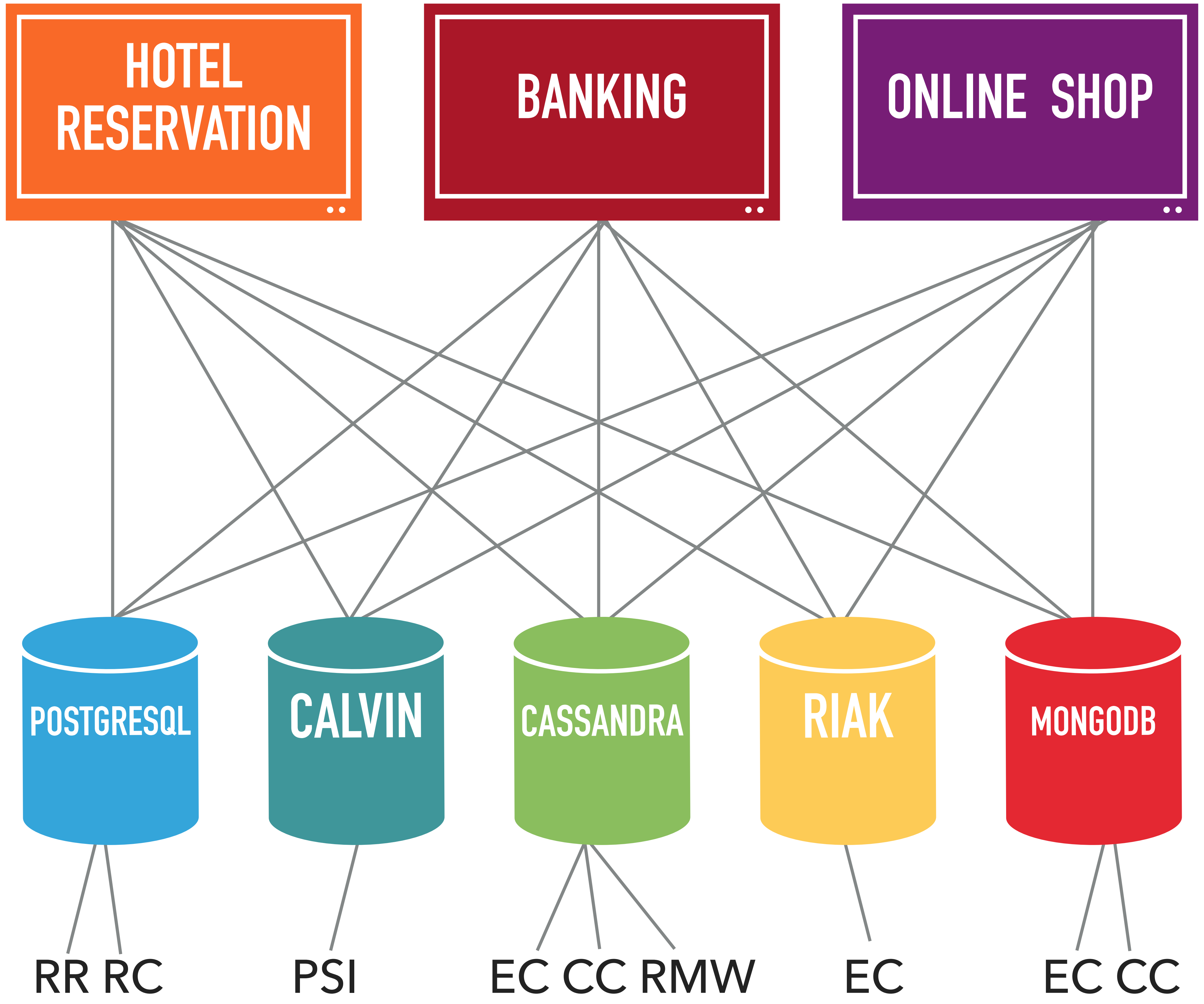
# BLACKBOX TESTING

- Independent of application semantics
- Independent of database specific guarantees
- Not reproducible
- Each database may offer multiple guarantees



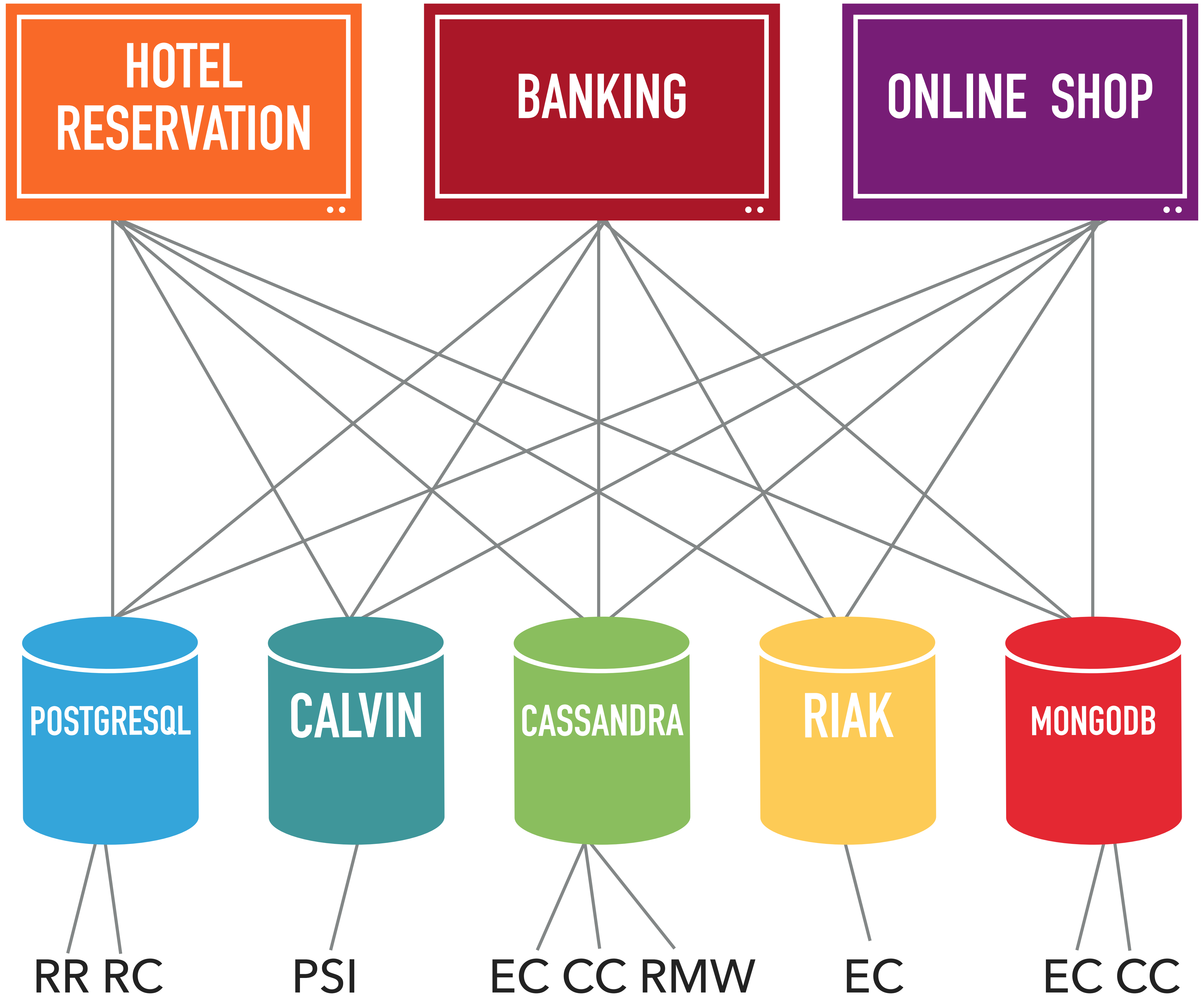
# BLACKBOX TESTING

- Independent of application semantics
- Independent of database specific guarantees
- Not reproducible
- Each database may offer multiple guarantees
- Time and resource consuming!



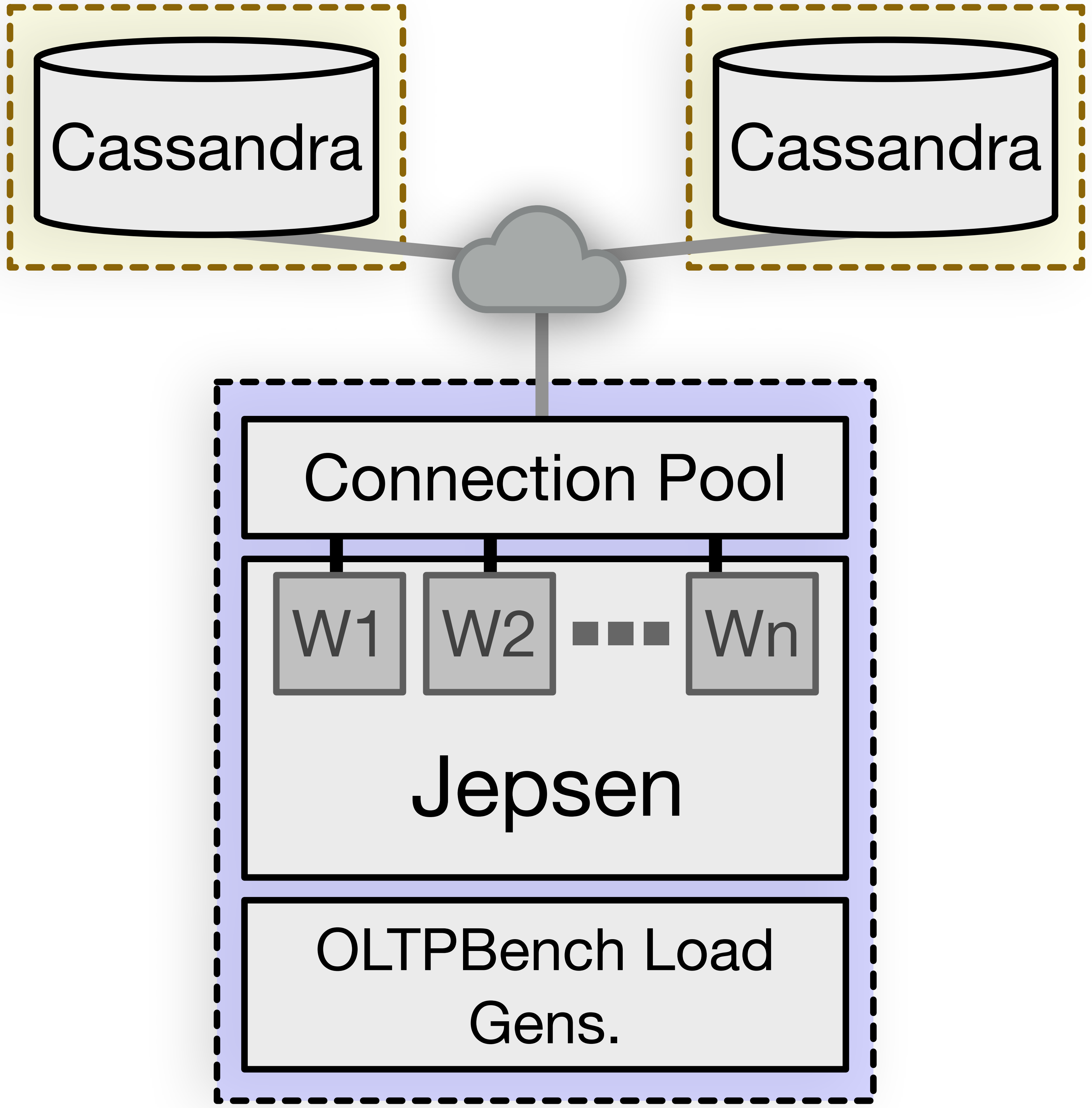
# BLACKBOX TESTING

- Independent of application semantics
- Independent of database specific guarantees
- Not reproducible
- Each database may offer multiple guarantees
- Time and resource consuming!
- No guarantees



# BLACKBOX TESTING IN ACTION

- State of the art cloud-based testing framework using ***Jepsen*** and ***OLTPBench***

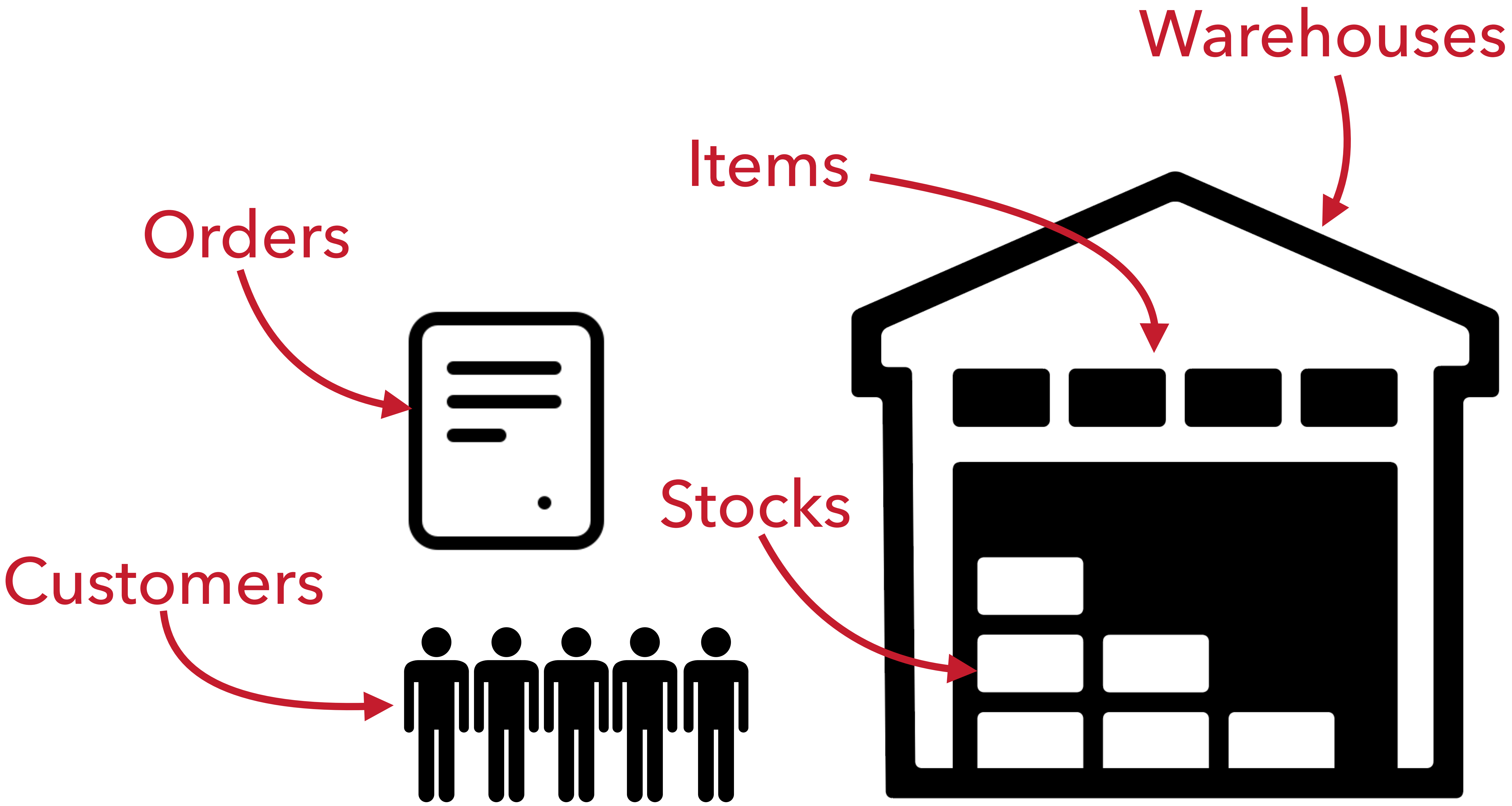


# BLACKBOX TESTING IN ACTION

- State of the art cloud-based testing framework using *Jepsen* and *OLTPBench*
- TPC-C benchmark

## BLACKBOX TESTING IN ACTION

- State of the art cloud-based testing framework using *Jepsen* and *OLTPBench*
- TPC-C benchmark

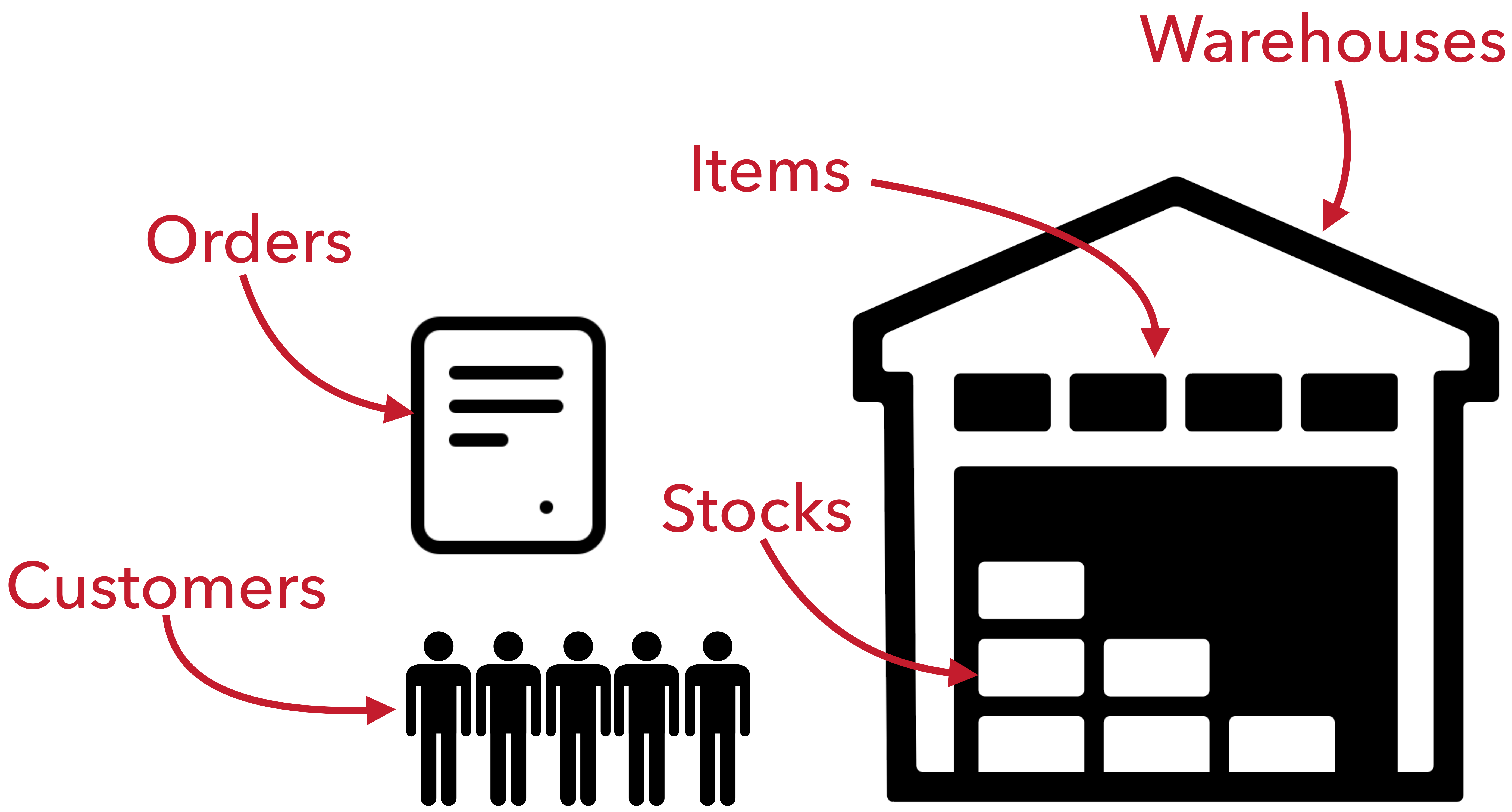


# BLACKBOX TESTING IN ACTION

- State of the art cloud-based testing framework using *Jepsen* and *OLTPBench*
- TPC-C benchmark

NEW_ORDER STOCK_LEVEL PAYMENT

ORDER_STATUS DELIVERY



## BLACKBOX TESTING IN ACTION

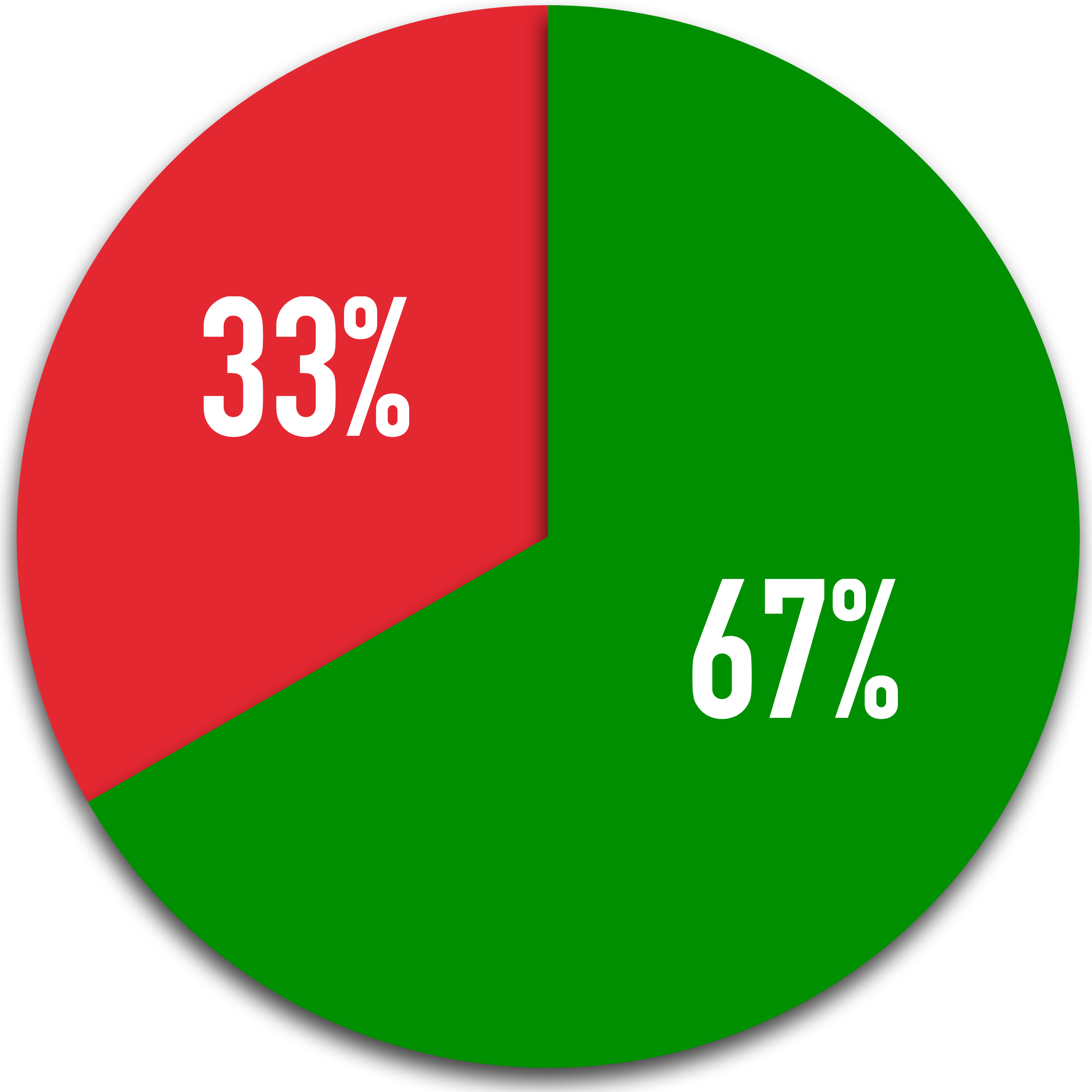
- State of the art cloud-based testing framework using *Jepsen* and *OLTPBench*
- TPC-C benchmark
- 21 application-level invariants were analyzed

| Invariant |
|---|
| CR1 |
| CR2 |
| CR3 |
| CR4 |
| CR5A |
| CR5B |
| CR6 |
| CR7A |
| CR7B |
| CR8 |
| CR9 |
| CR10 |
| CR11 |
| CR12 |
| NCR1 |
| NCR2 |
| NCR3 |
| NCR4 |
| NCR5 |
| NCR6 |
| NCR7 |

# BLACKBOX TESTING IN ACTION

- State of the art cloud-based testing framework using *Jepsen* and *OLTPBench*
- TPC-C benchmark
- 21 application-level invariants were analyzed
- Only **14 out of 21** invariants were broken at best

| Invariant | Broken? |
|---|---|
| CR1 | Y |
| CR2 | Y |
| CR3 | Y |
| CR4 | Y |
| CR5A | N |
| CR5B | N |
| CR6 | Y |
| CR7A | N |
| CR7B | N |
| CR8 | Y |
| CR9 | Y |
| CR10 | Y |
| CR11 | Y |
| CR12 | Y |
| NCR1 | Y |
| NCR2 | Y |
| NCR3 | N |
| NCR4 | N |
| NCR5 | Y |
| NCR6 | Y |
| NCR7 | N |



**1/3 of invariants are assumed to be preserved**

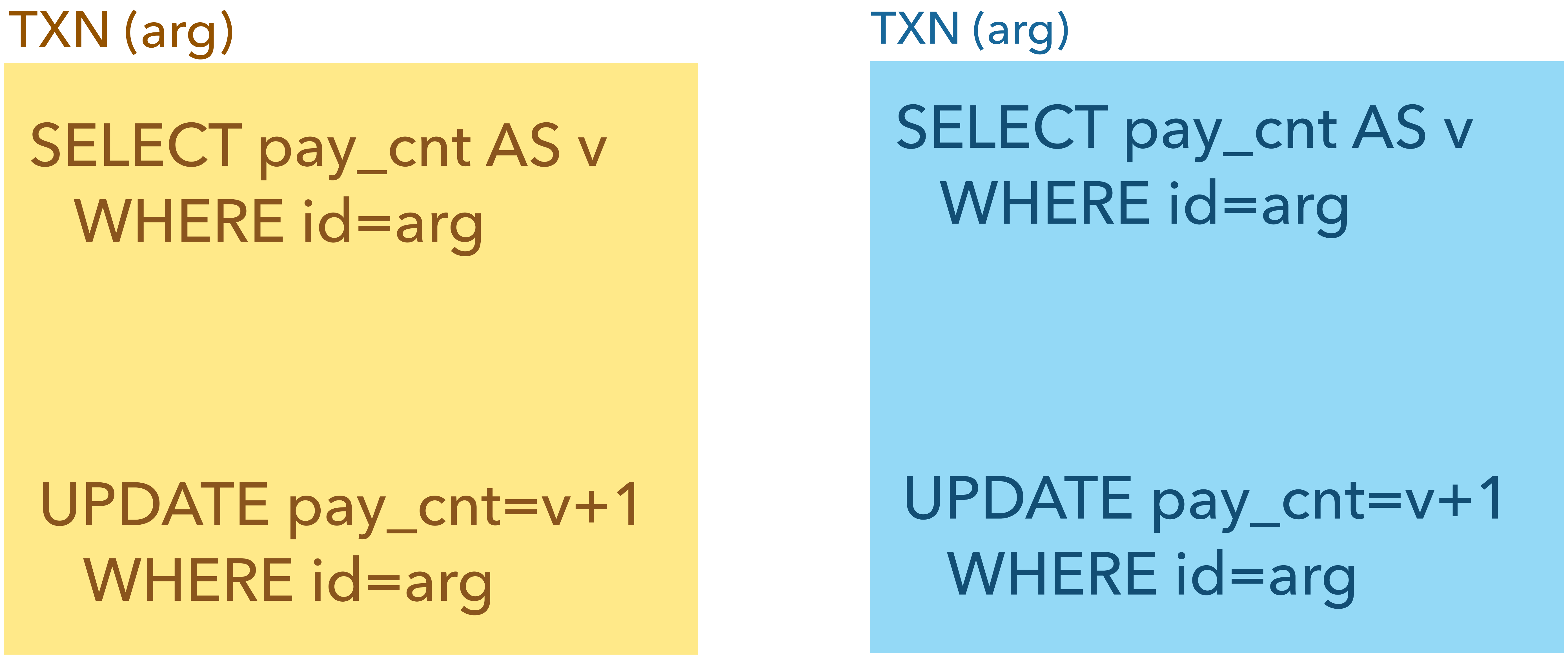
# WHITE-BOX ANALYSIS

# WHITE-BOX ANALYSIS

- Systematic assessment of anomalous executions **within a given program**

# WHITE-BOX ANALYSIS

- Systematic assessment of anomalous executions **within a given program**



TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg


UPDATE pay_cnt=v+1
  WHERE id=arg
```

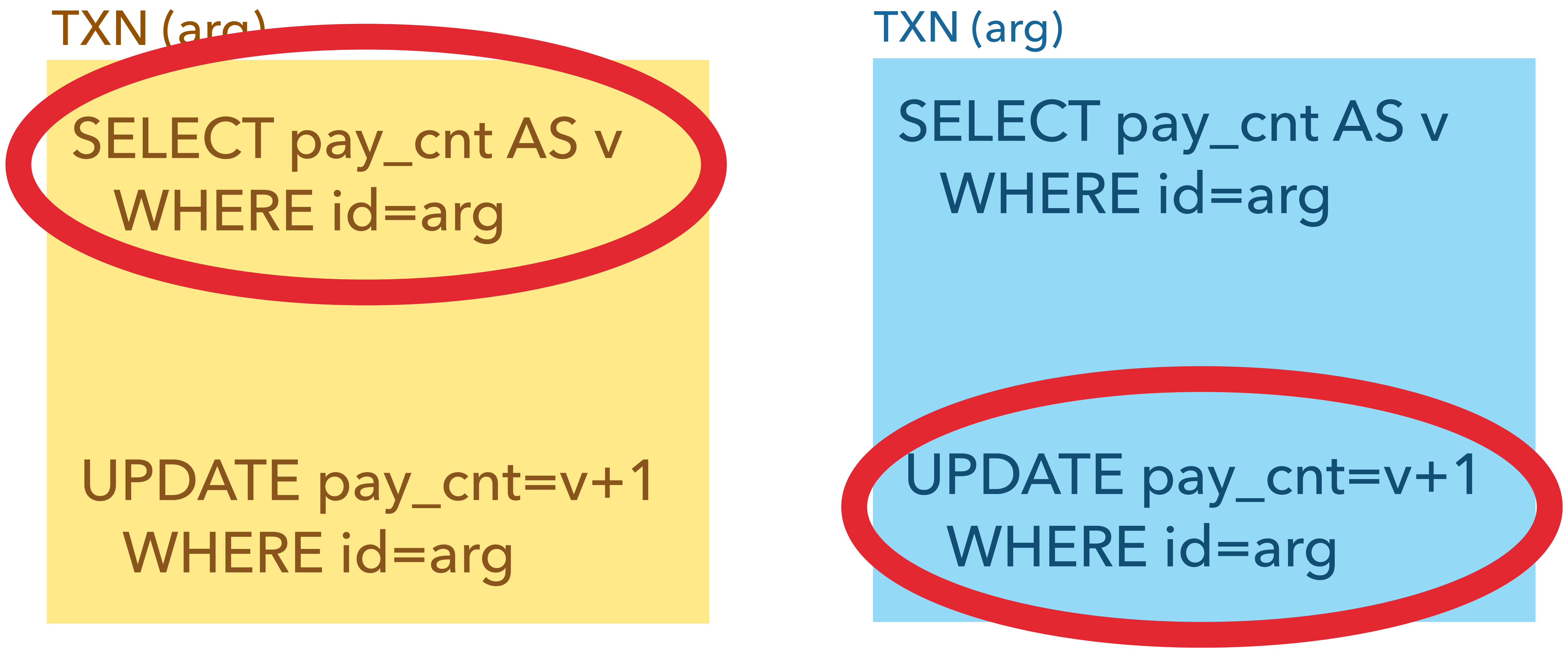
TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg


UPDATE pay_cnt=v+1
  WHERE id=arg
```

# WHITE-BOX ANALYSIS

- Systematic assessment of anomalous executions within a given program
- Data dependencies among database operations



TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg

UPDATE pay_cnt=v+1
  WHERE id=arg
```

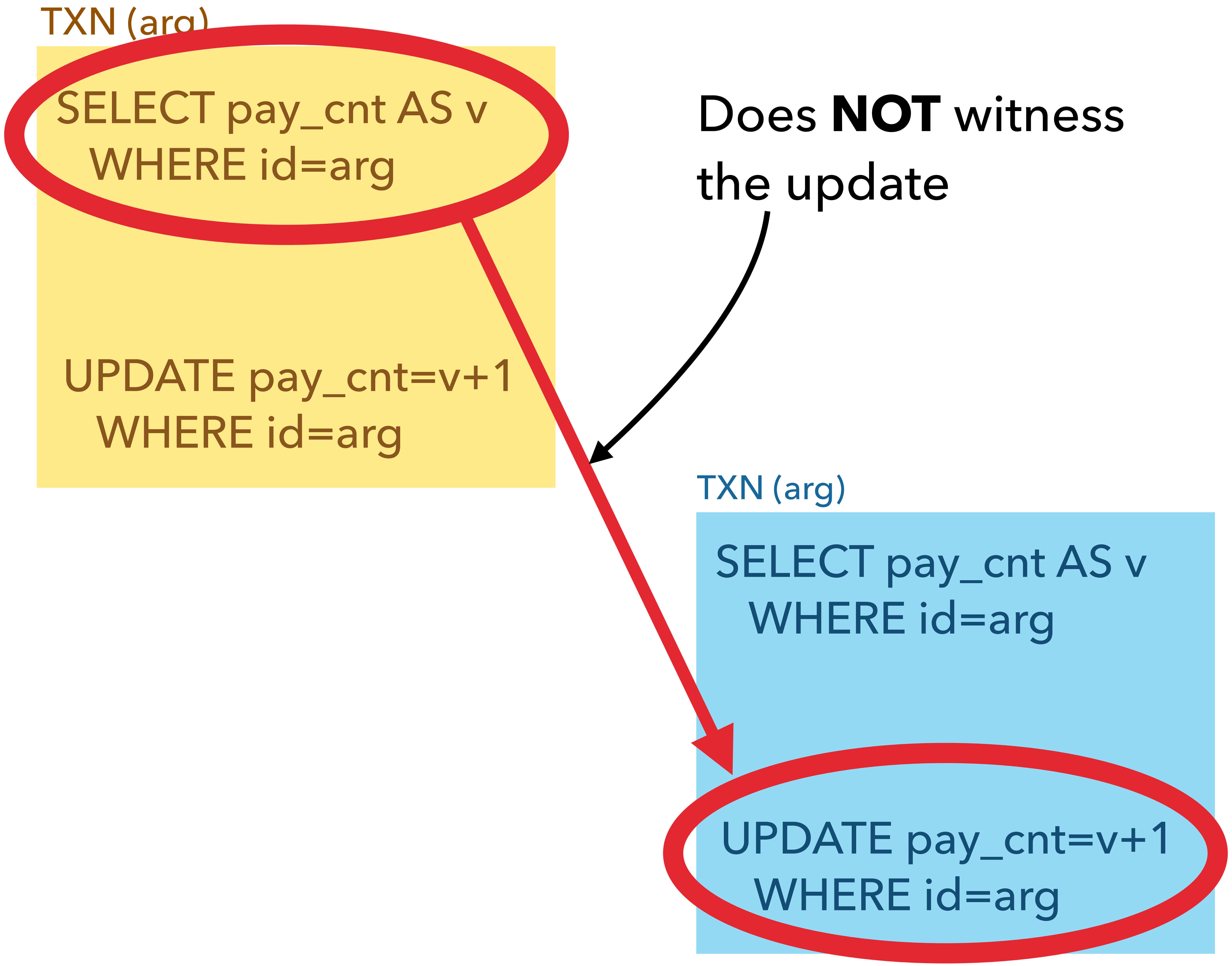
TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg

UPDATE pay_cnt=v+1
  WHERE id=arg
```

# WHITE-BOX ANALYSIS

- Systematic assessment of anomalous executions within a given program
- Data dependencies among database operations
- Execution properties (e.g. order) affect dependent operations



TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg

UPDATE pay_cnt=v+1
  WHERE id=arg
```

Does **NOT** witness the update

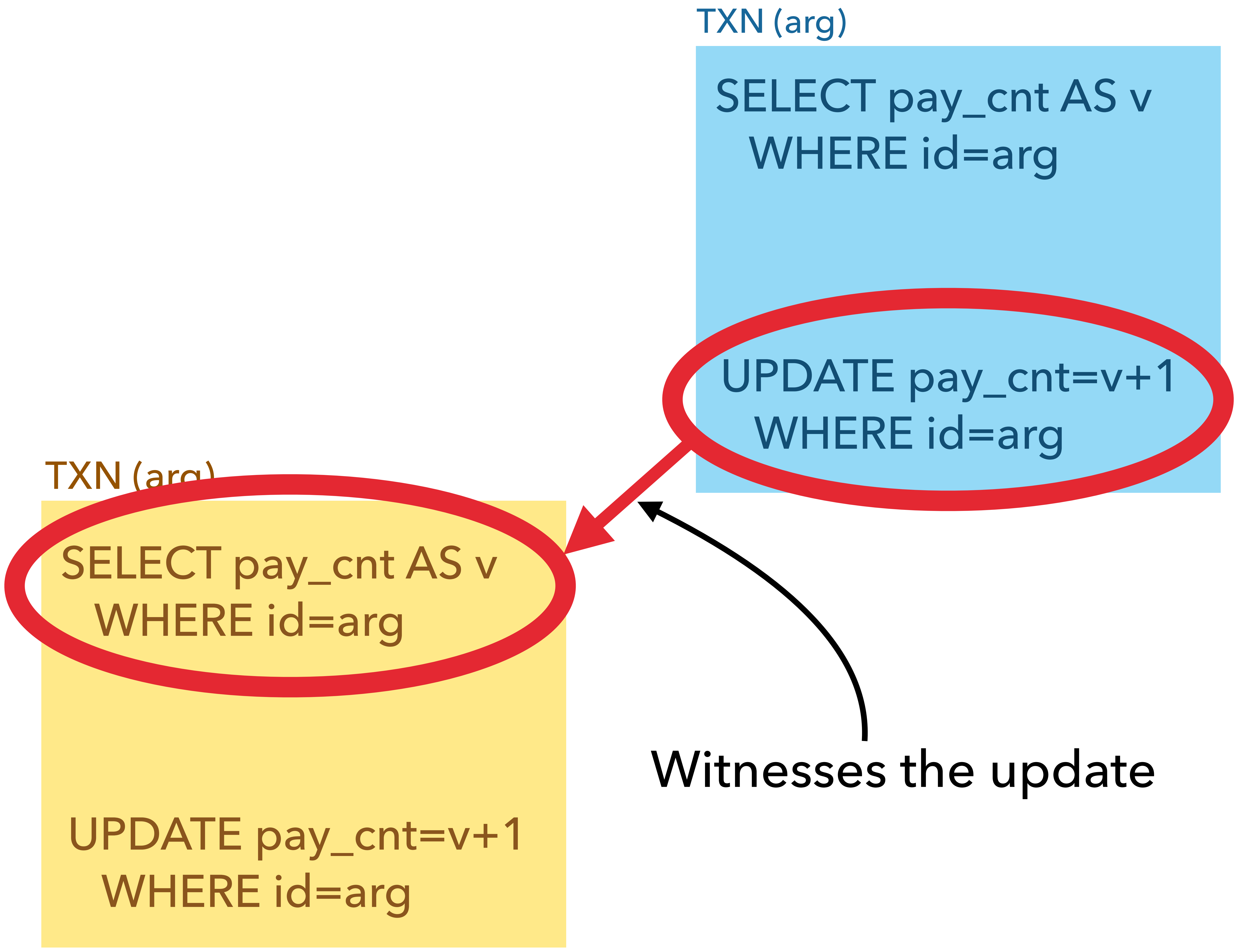
TXN (arg)

```
SELECT pay_cnt AS v
  WHERE id=arg

UPDATE pay_cnt=v+1
  WHERE id=arg
```

# WHITE-BOX ANALYSIS

- Systematic assessment of anomalous executions within a given program
- Data dependencies among database operations
- Execution properties (e.g. order) affect dependent operations



TXN (arg)

SELECT pay_cnt AS v
WHERE id=arg

UPDATE pay_cnt=v+1
WHERE id=arg

TXN (arg)

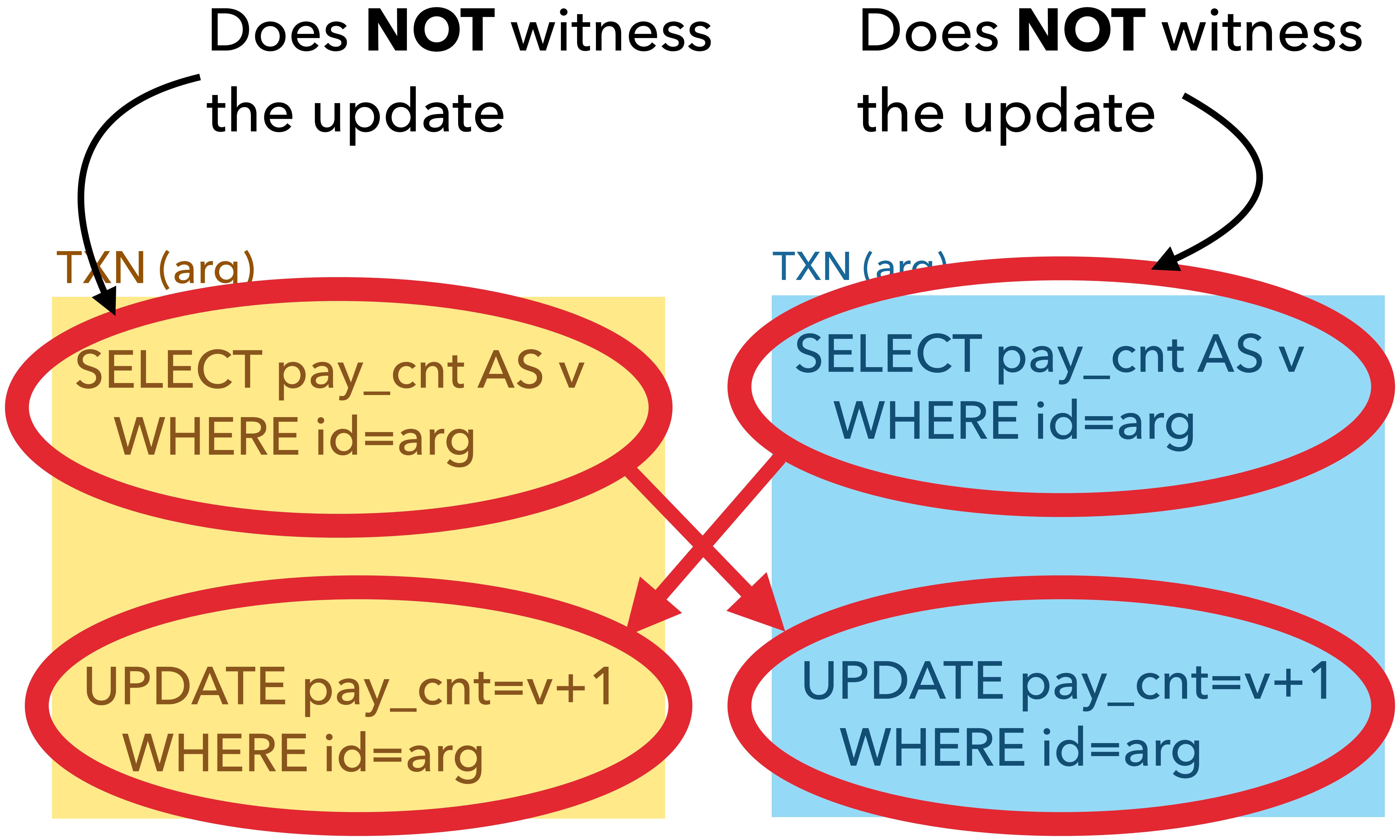
SELECT pay_cnt AS v
WHERE id=arg

UPDATE pay_cnt=v+1
WHERE id=arg

Witnesses the update

# WHITE-BOX ANALYSIS

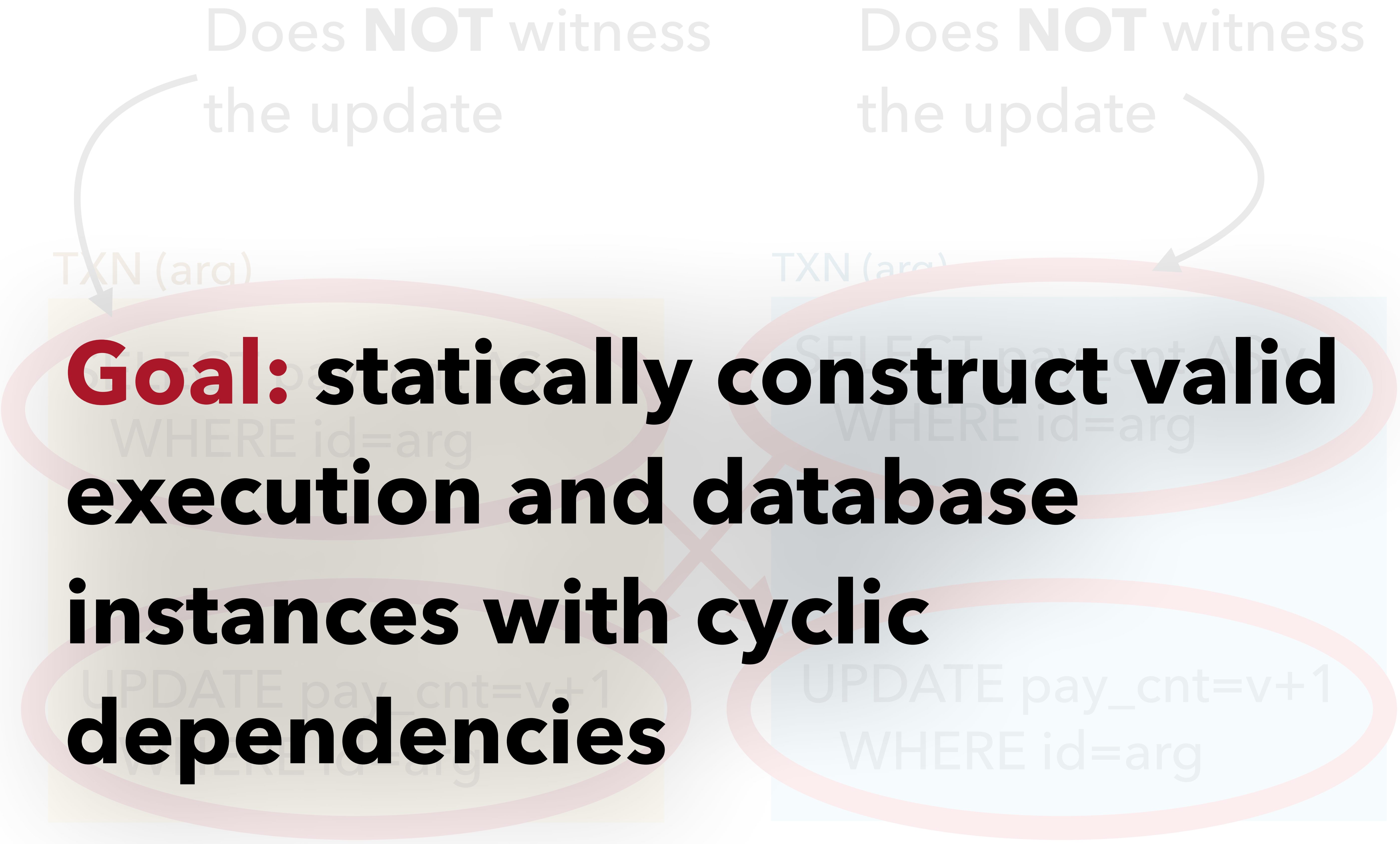
- Systematic assessment of anomalous executions within a given program
- Data dependencies among database operations
- Execution properties (e.g. order) affect dependent operations
- Cyclic dependencies between transactions correspond to anomalous executions



Does **NOT** witness the update
Does **NOT** witness the update
TXN (arg)
SELECT pay_cnt AS v WHERE id=arg
UPDATE pay_cnt=v+1 WHERE id=arg
TXN (arg)
SELECT pay_cnt AS v WHERE id=arg
UPDATE pay_cnt=v+1 WHERE id=arg

## WHITE-BOX ANALYSIS

- Systematic assessment of anomalous executions within a given program
- Data dependencies among database operations
- Execution properties (e.g. order) affect dependent operations
- Cyclic dependencies between transactions correspond to anomalous executions



**Goal: statically construct valid execution and database instances with cyclic dependencies**

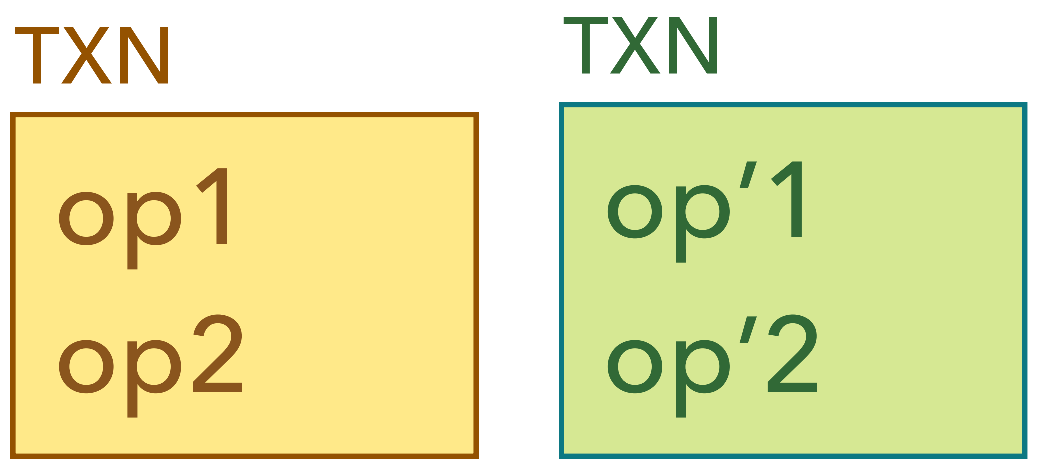
# FORMAL EXECUTION MODEL

## FORMAL EXECUTION MODEL

- Transactions are arbitrarily invoked

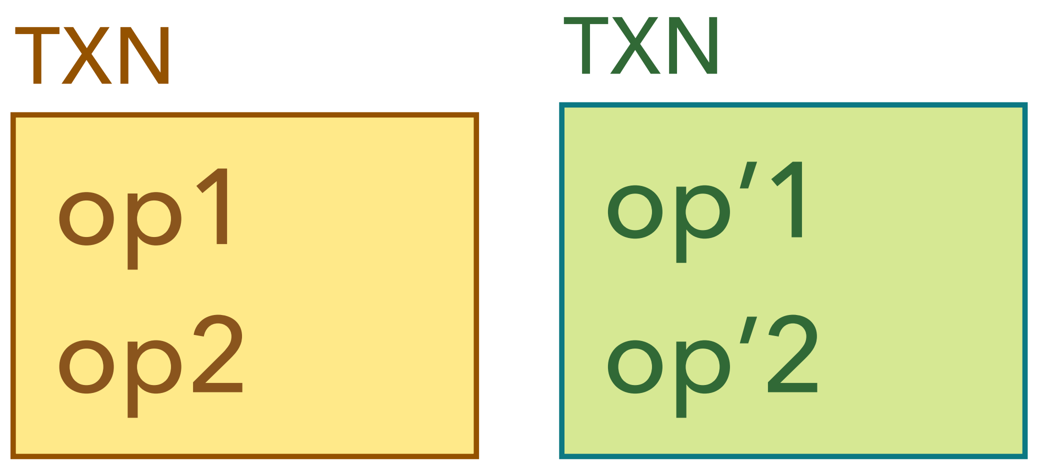
## FORMAL EXECUTION MODEL

- Transactions are arbitrarily invoked



# FORMAL EXECUTION MODEL

- Transactions are arbitrarily invoked
- An **Operation** from an arbitrary transaction is executed at a random **partition**



TXN

op1

op2

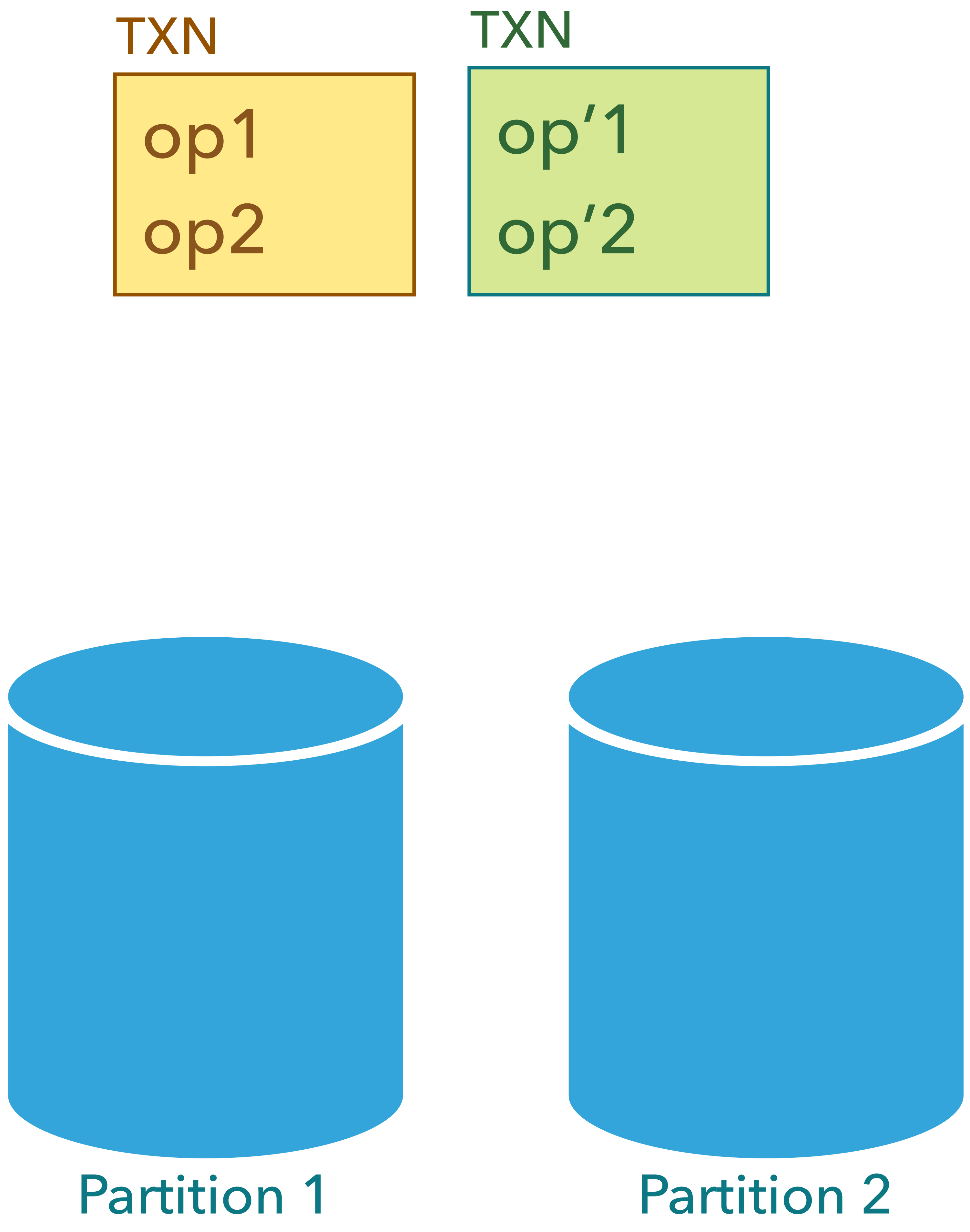
TXN

op’1

op’2

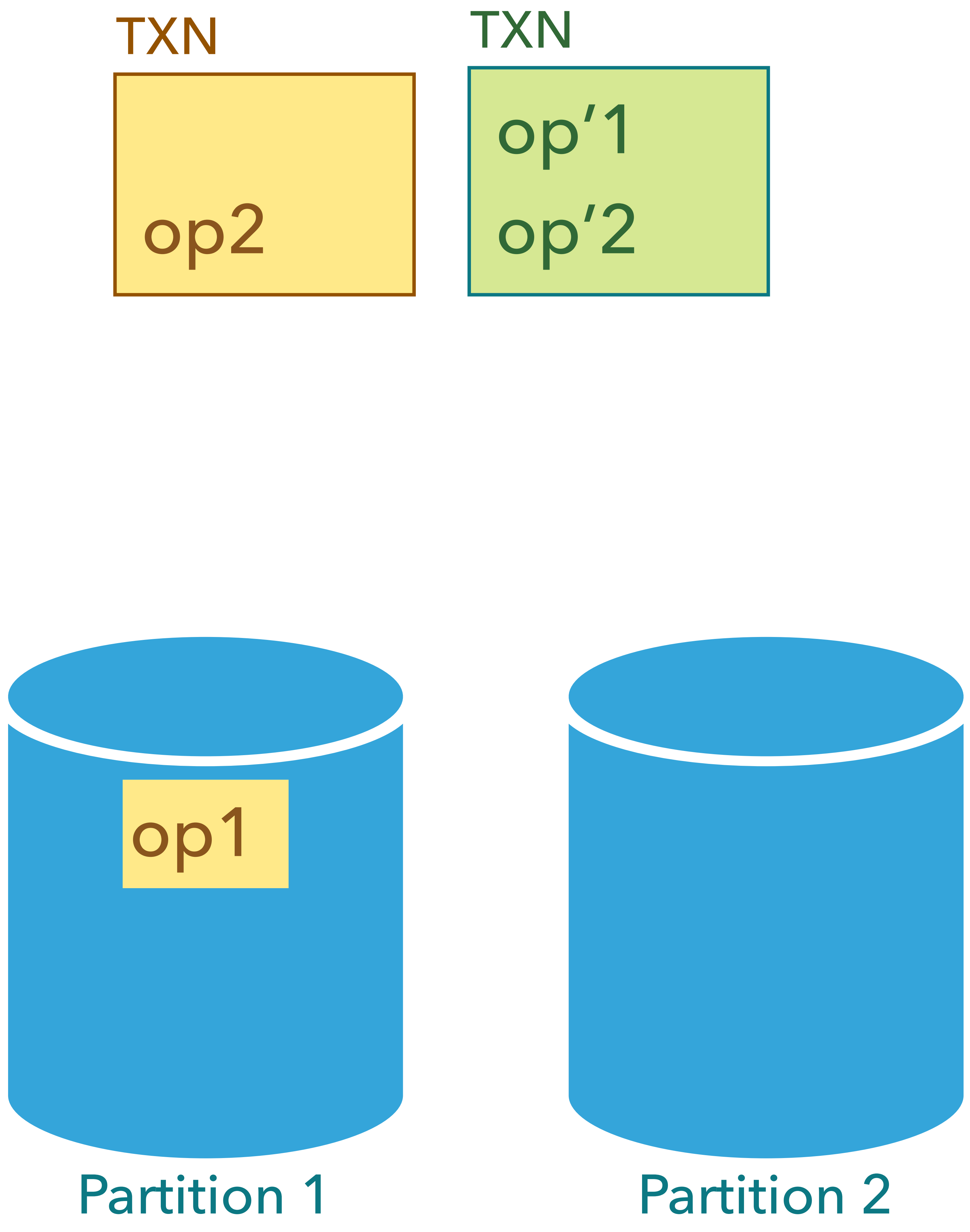
# FORMAL EXECUTION MODEL

- Transactions are arbitrarily invoked
- An **Operation** from an arbitrary transaction is executed at a random **partition**



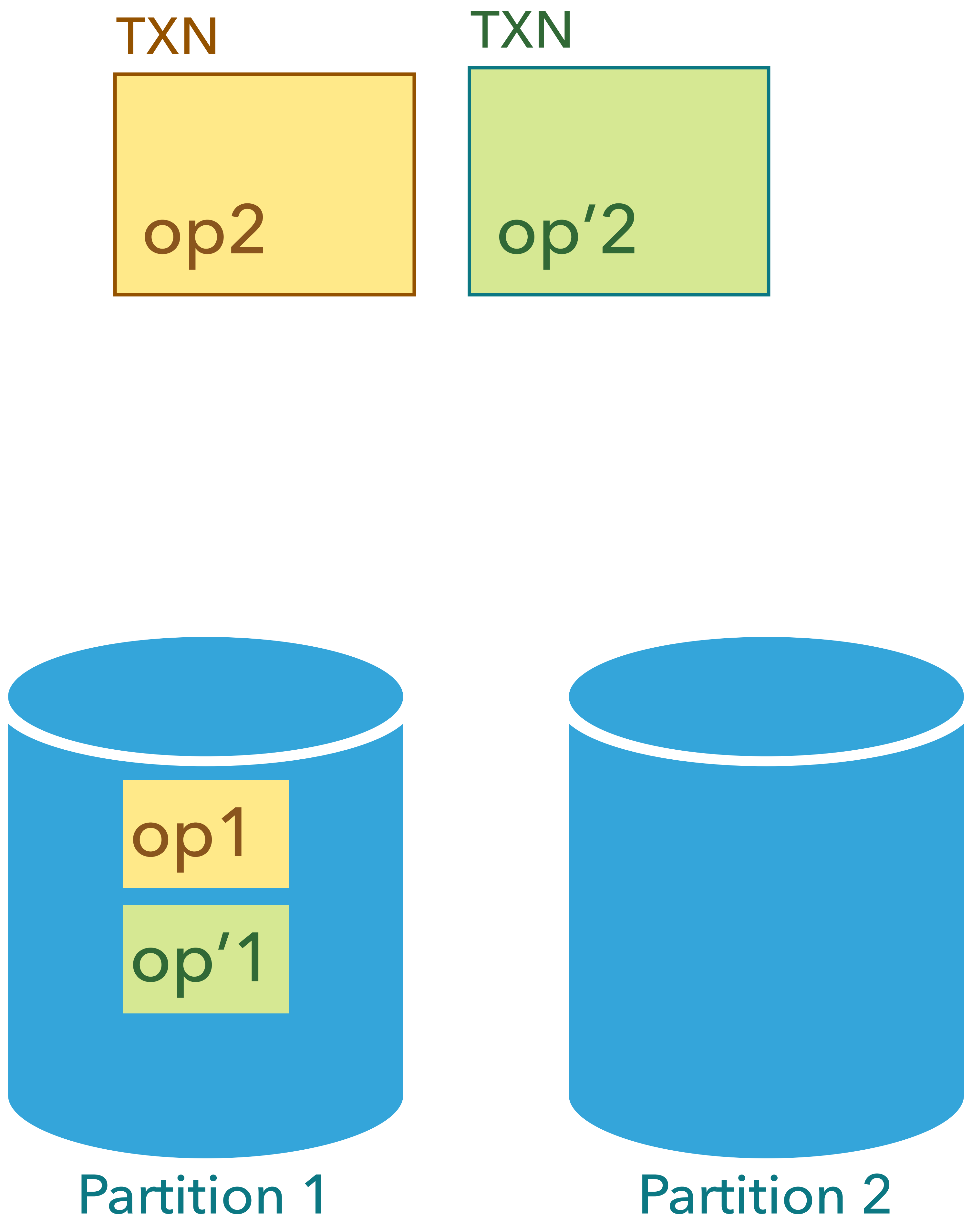
# FORMAL EXECUTION MODEL

- Transactions are arbitrarily invoked
- An **Operation** from an arbitrary transaction is executed at a random **partition**



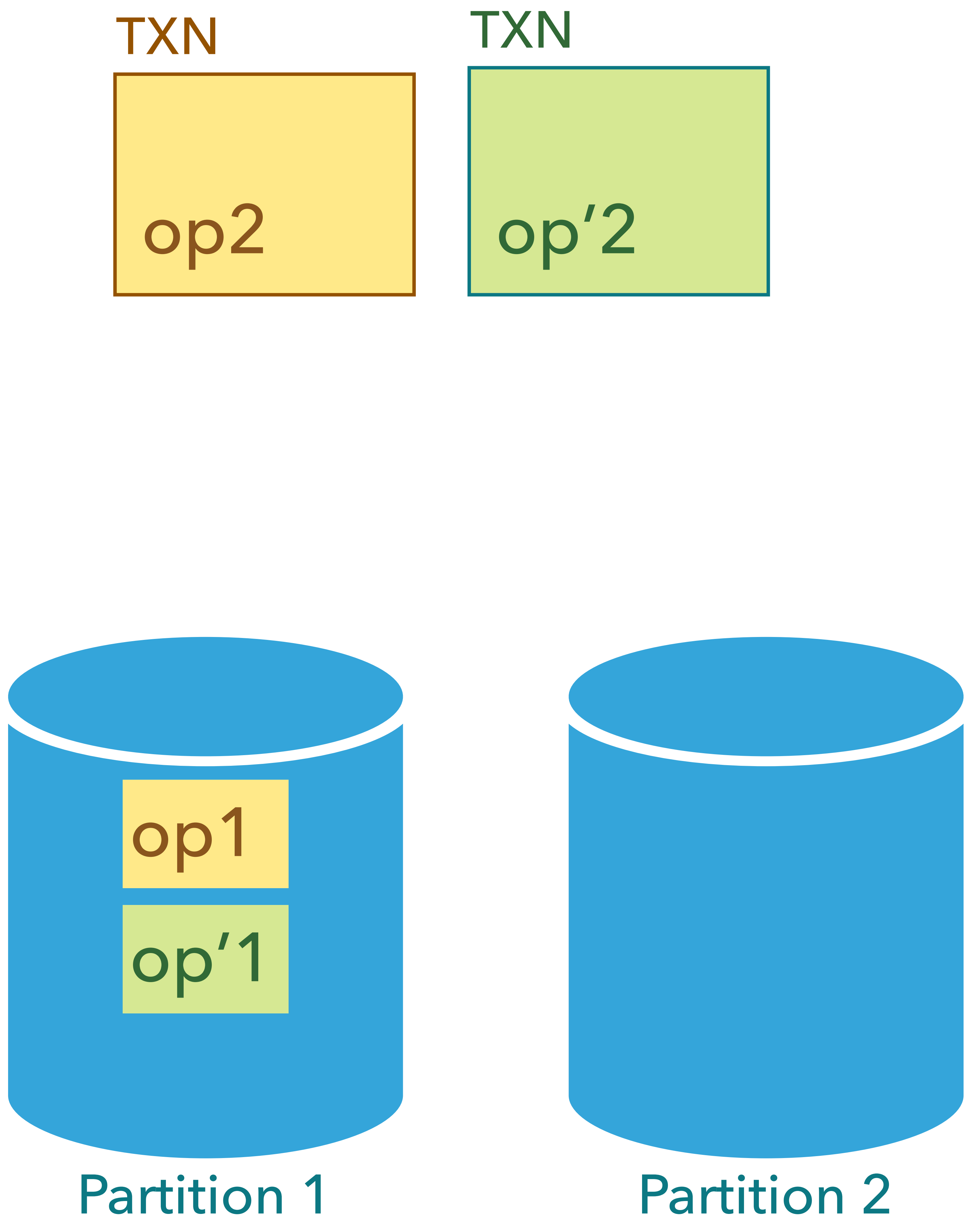
# FORMAL EXECUTION MODEL

- Transactions are arbitrarily invoked
- An **Operation** from an arbitrary transaction is executed at a random **partition**

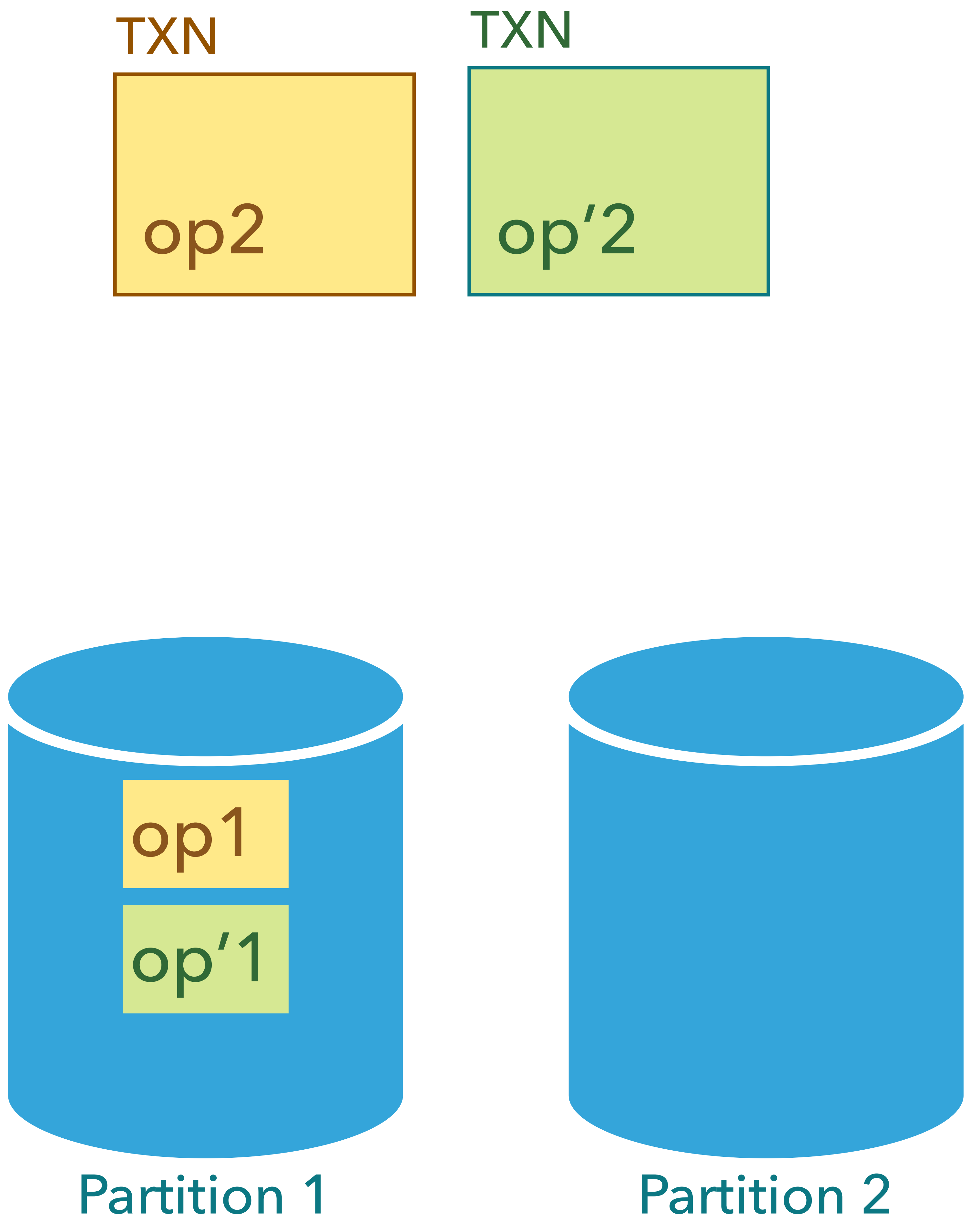


# FORMAL EXECUTION MODEL

- Transactions are arbitrarily invoked
- An **Operation** from an arbitrary transaction is executed at a random **partition**
- Operations create a set of read and write **effects** upon execution in the partition

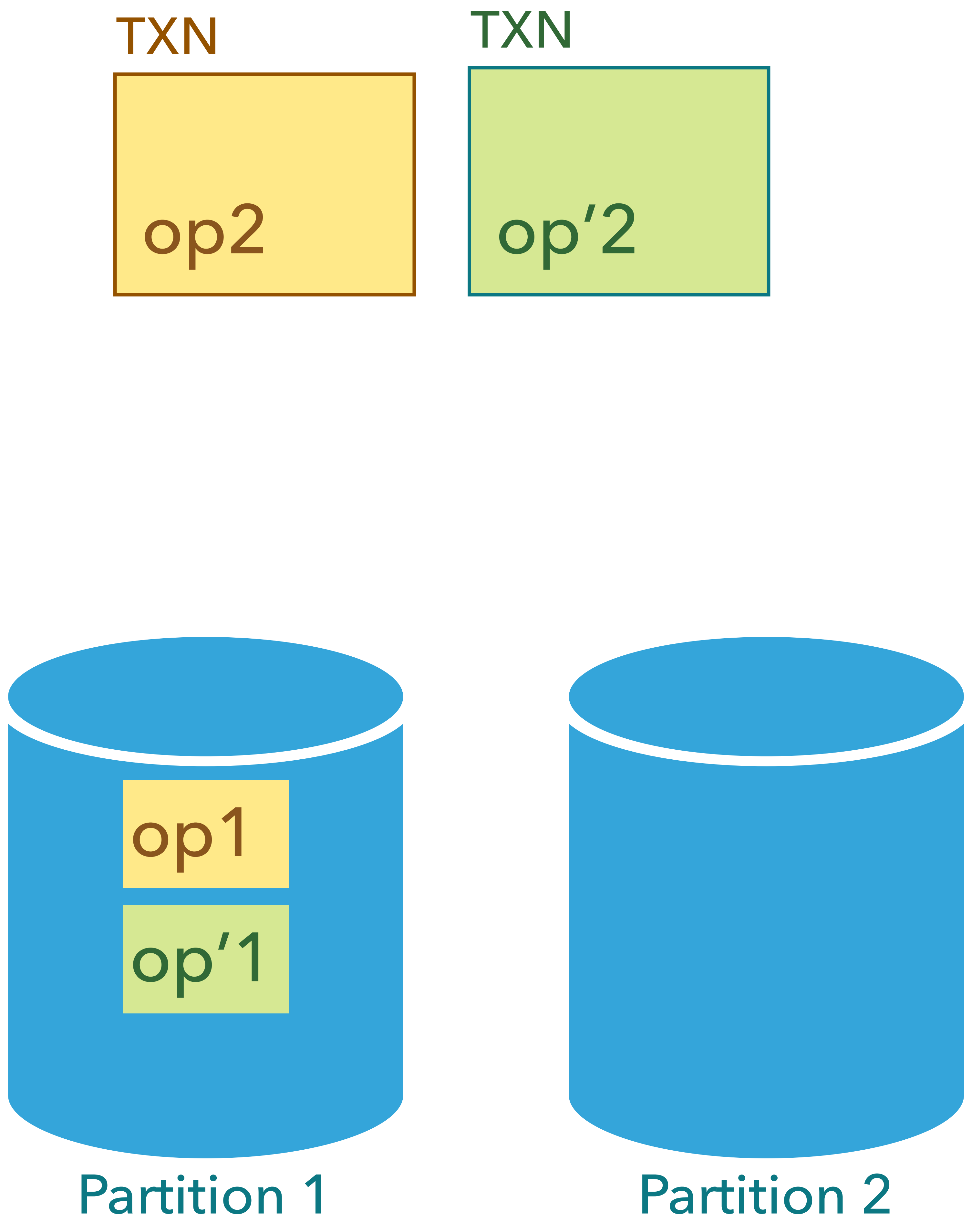


- Transactions are arbitrarily invoked
- An **Operation** from an arbitrary transaction is executed at a random **partition**
- Operations create a set of read and write **effects** upon execution in the partition
- A relations on the set of effects



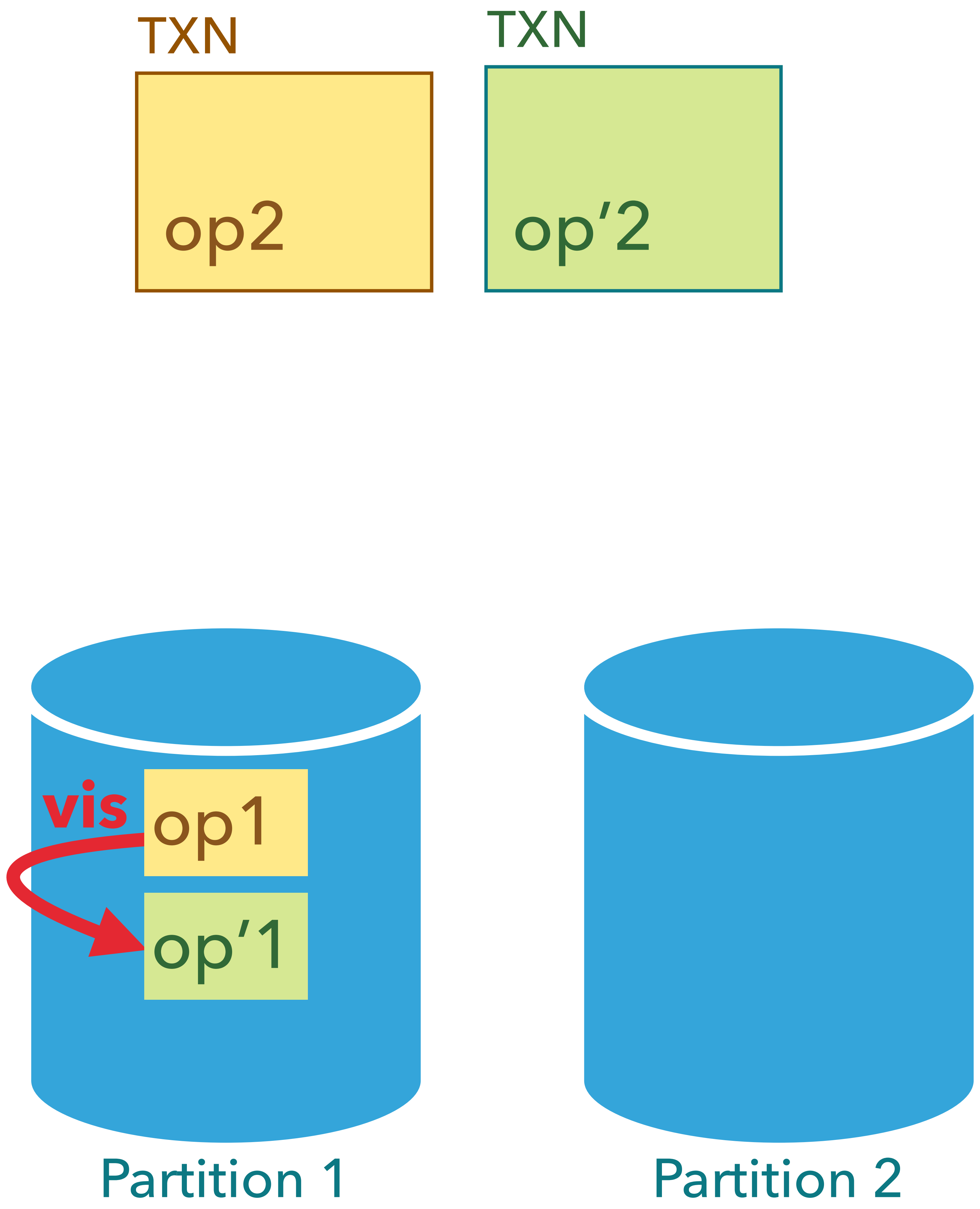
# FORMAL EXECUTION MODEL

- Transactions are arbitrarily invoked
- An **Operation** from an arbitrary transaction is executed at a random **partition**
- Operations create a set of read and write **effects** upon execution in the partition
- A relations on the set of effects
  - **visibility:** *causal* precedence between effects



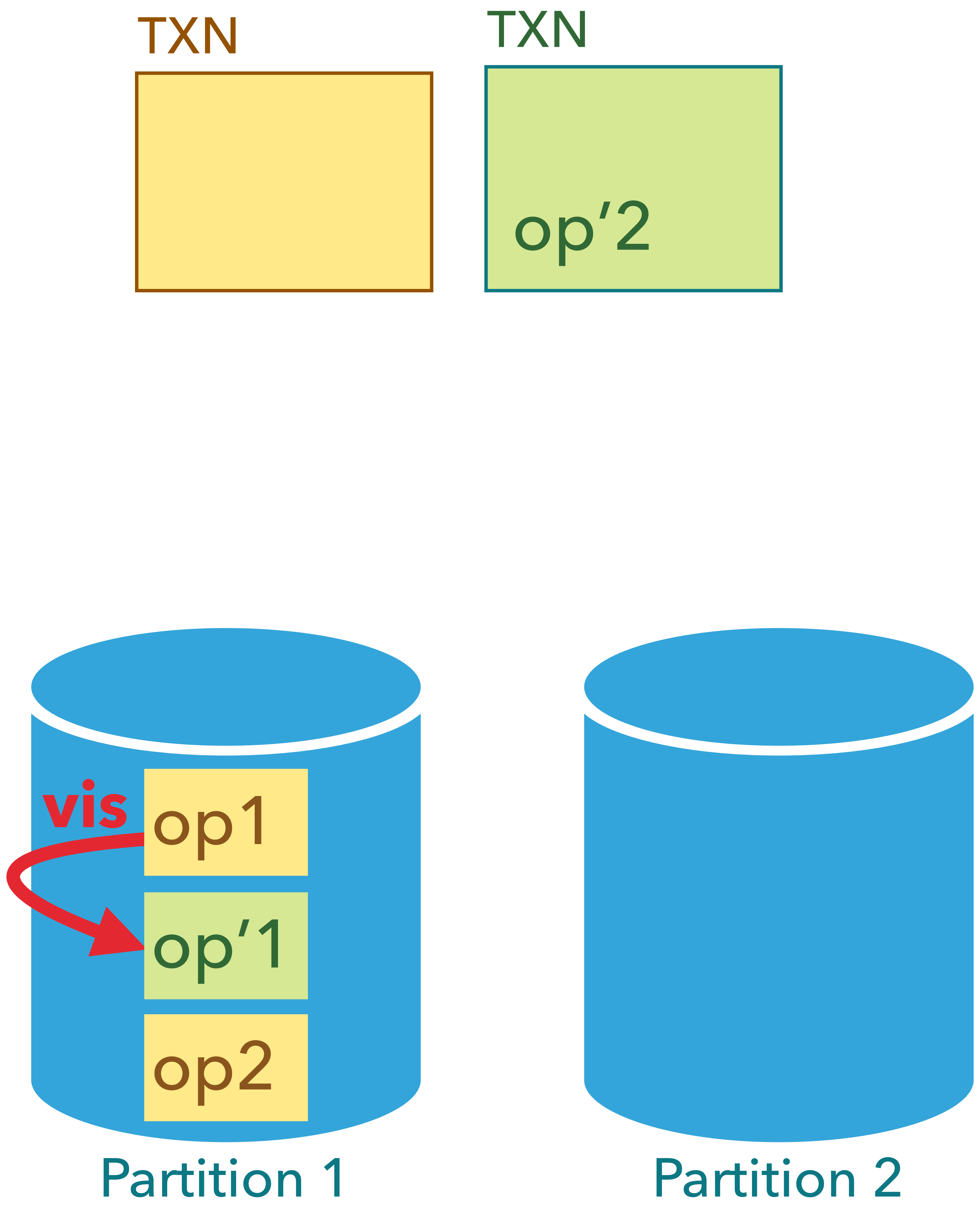
# FORMAL EXECUTION MODEL

- Transactions are arbitrarily invoked
- An **Operation** from an arbitrary transaction is executed at a random **partition**
- Operations create a set of read and write **effects** upon execution in the partition
- A relations on the set of effects
  - **visibility:** *causal* precedence between effects



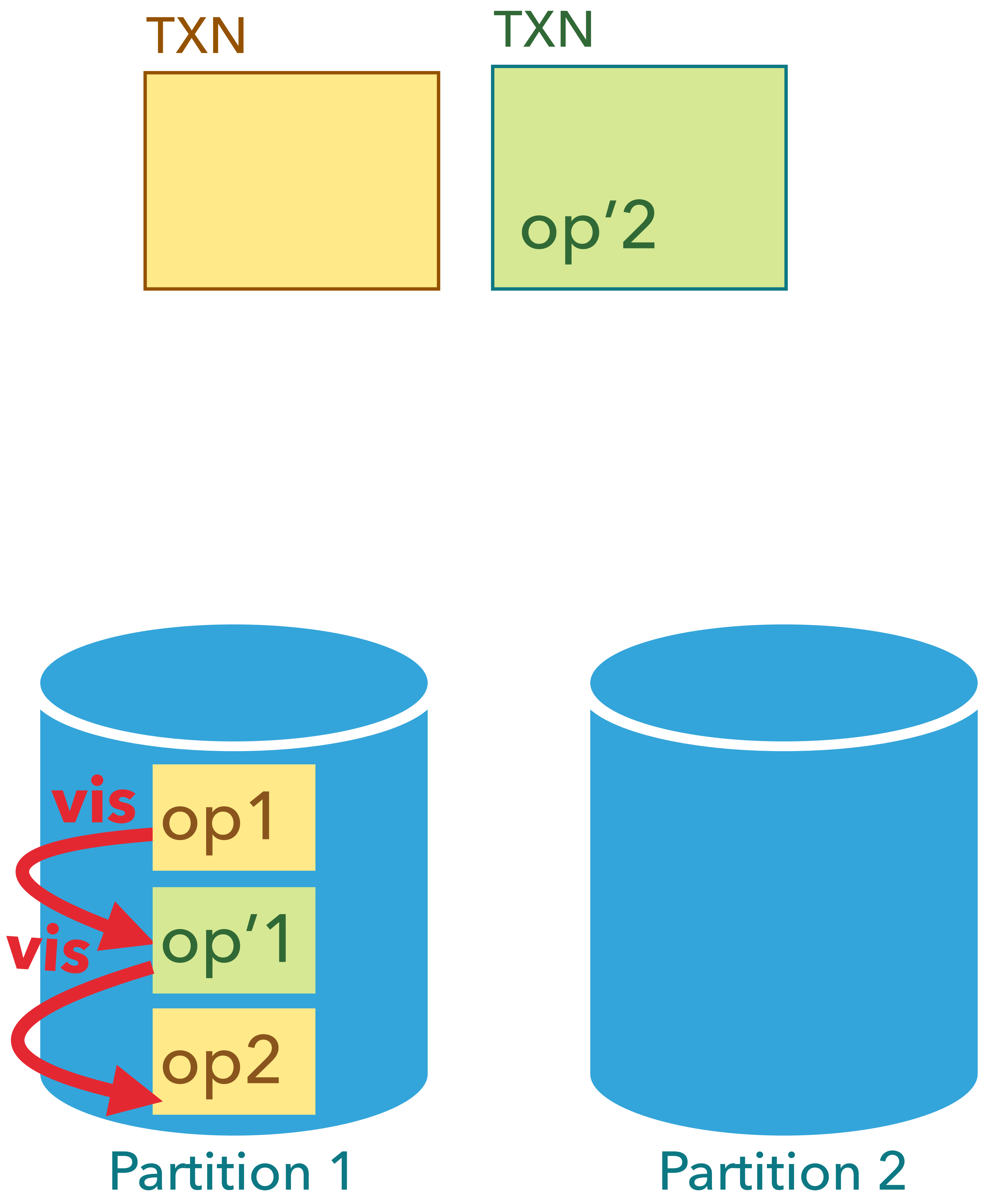
# FORMAL EXECUTION MODEL

- Transactions are arbitrarily invoked
- An **Operation** from an arbitrary transaction is executed at a random **partition**
- Operations create a set of read and write **effects** upon execution in the partition
- A relations on the set of effects
  - **visibility:** *causal* precedence between effects



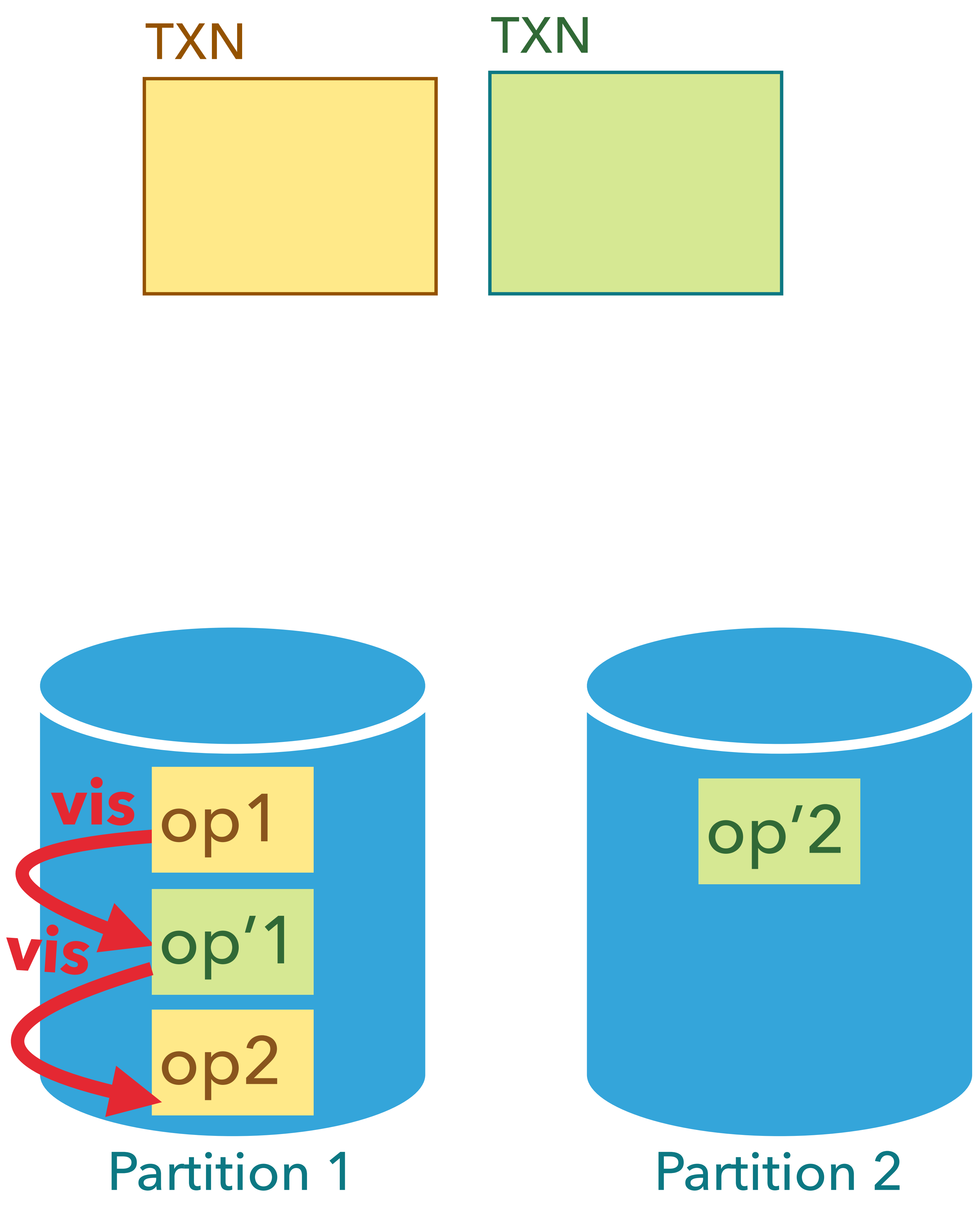
# FORMAL EXECUTION MODEL

- Transactions are arbitrarily invoked
- An **Operation** from an arbitrary transaction is executed at a random **partition**
- Operations create a set of read and write **effects** upon execution in the partition
- A relations on the set of effects
  - **visibility:** *causal* precedence between effects



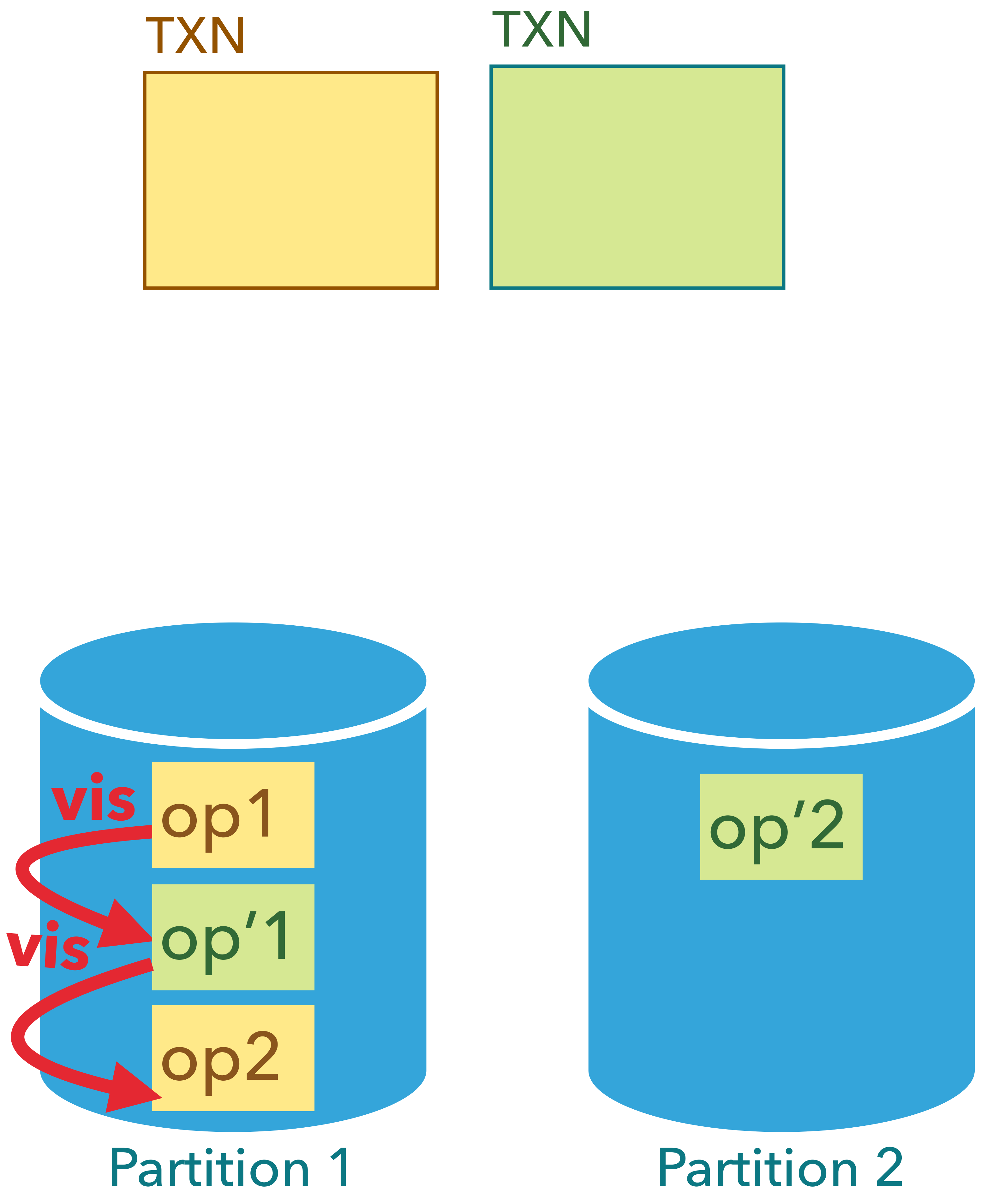
# FORMAL EXECUTION MODEL

- Transactions are arbitrarily invoked
- An **Operation** from an arbitrary transaction is executed at a random **partition**
- Operations create a set of read and write **effects** upon execution in the partition
- A relations on the set of effects
  - **visibility:** *causal* precedence between effects



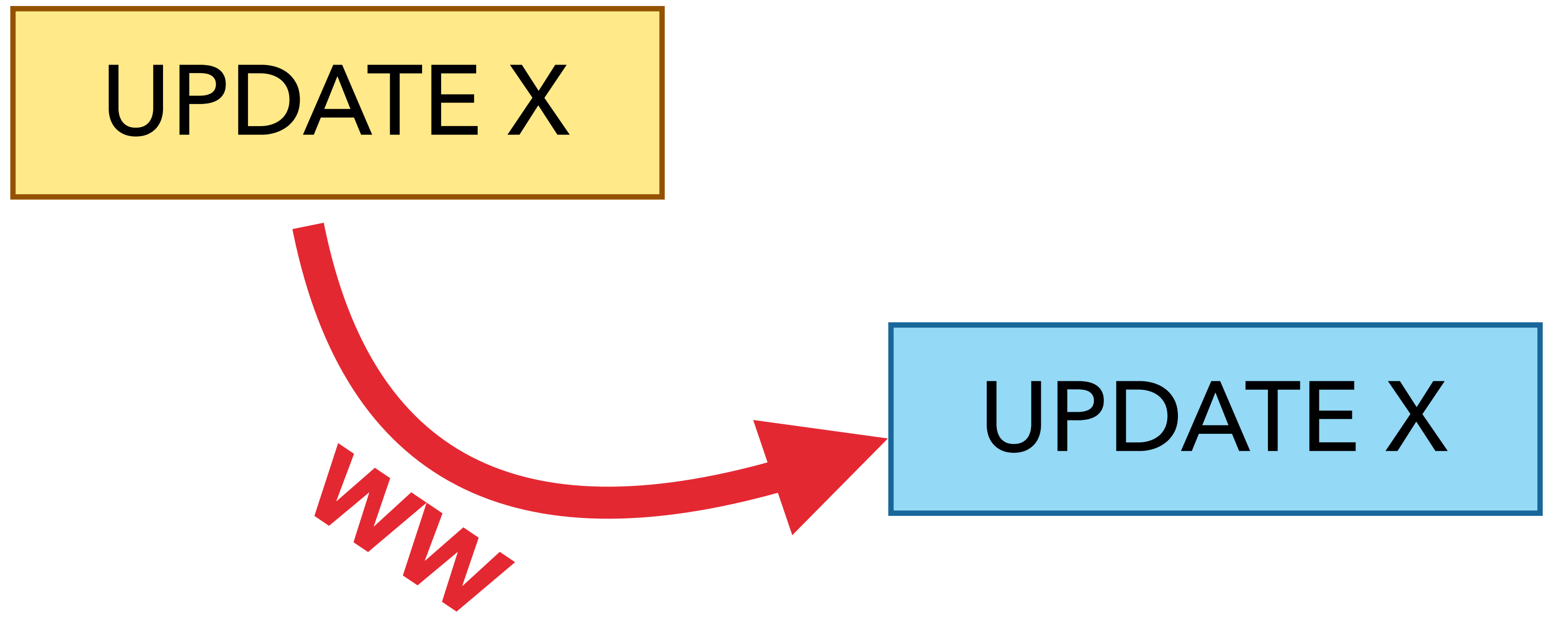
# FORMAL EXECUTION MODEL

- Transactions are arbitrarily invoked
- An **Operation** from an arbitrary transaction is executed at a random **partition**
- Operations create a set of read and write **effects** upon execution in the partition
- A relations on the set of effects
  - **visibility:** *causal* precedence between effects
  - Only within a partition!



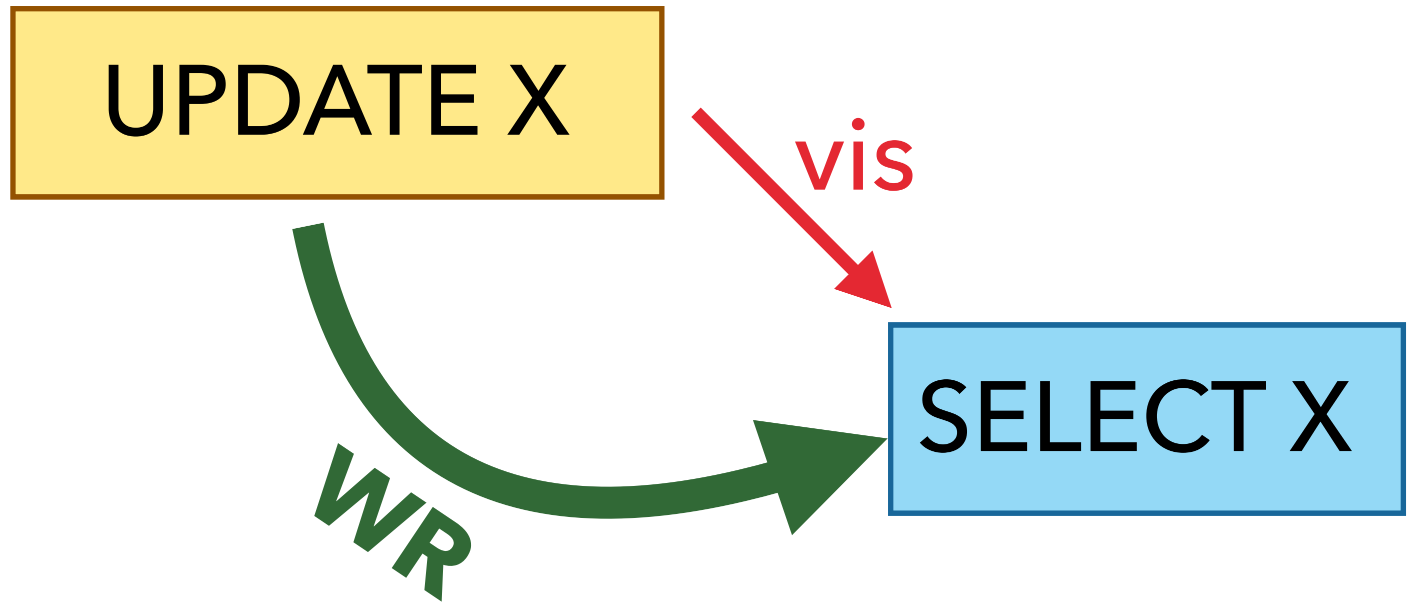
# DEPENDENCY RELATIONS

- Operation-level dependencies
  - write dependency (**WW**)



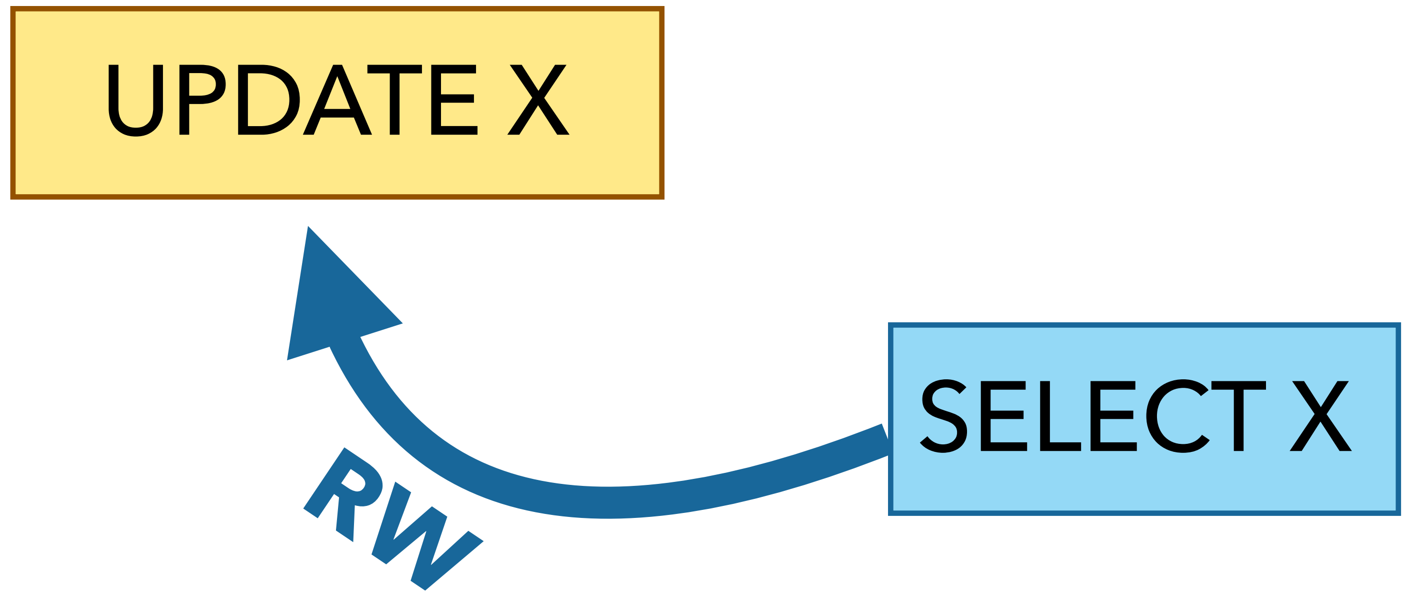
# DEPENDENCY RELATIONS

- Operation-level dependencies
  - write dependency (**WW**)
  - read dependency (**WR**)



# DEPENDENCY RELATIONS

- Operation-level dependencies
  - write dependency (**WW**)
  - read dependency (**WR**)
  - read anti-dependency (**RW**)



# FOL ENCODING

# FOL ENCODING

- A language of axiomatic relations encoded as a decidable fragment of first order logic (FOL)

# FOL ENCODING

- A language of axiomatic relations encoded as a decidable fragment of first order logic (FOL)
- Finding bounded anomalies against a database abstraction is reduced to finding satisfying assignments to a formula $\varphi$

- A language of axiomatic relations encoded in a decidable fragment of first order logic (FOL)
- Finding bounded anomalies against a database abstraction is reduced to finding satisfying assignments to a formula
- Valid assignments are constrained by **five conjuncts**

$$\varphi \equiv \varphi_{\mathrm{CONTEXT}} \wedge \varphi_{\mathrm{DB}} \wedge \varphi_{\mathrm{DEP}\rightarrow} \wedge \varphi_{\rightarrow\mathrm{DEP}} \wedge \varphi_{\mathrm{ANOMALY}}$$

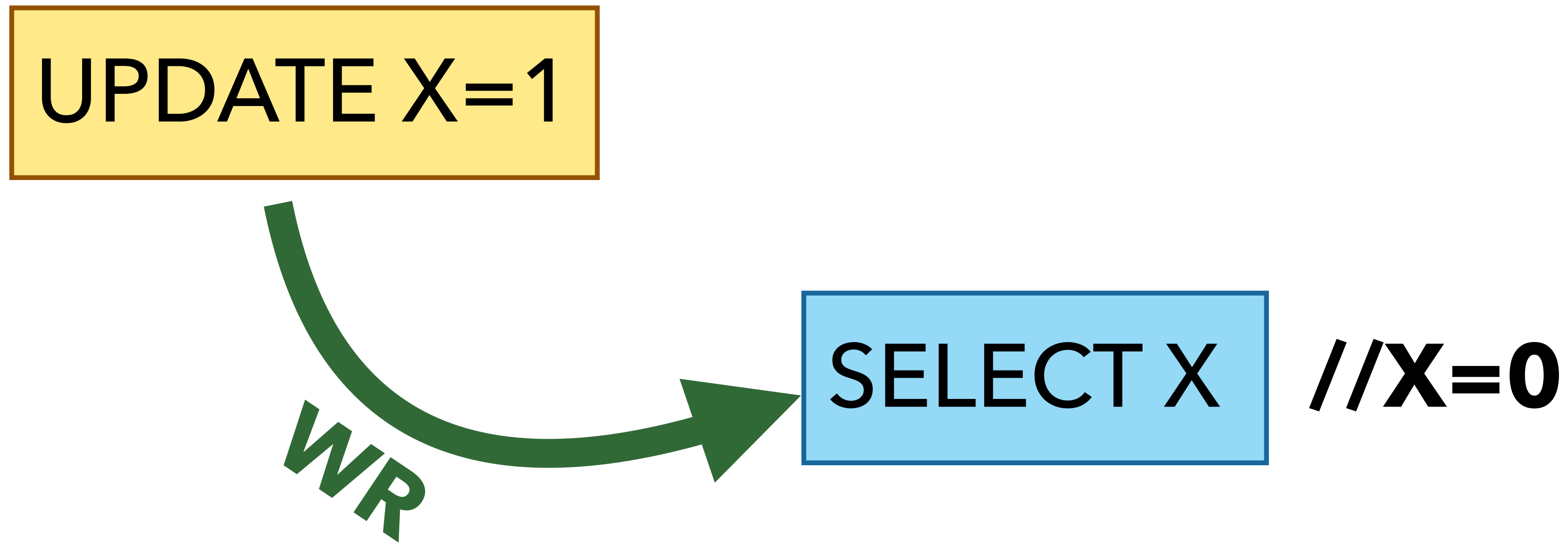
$$\varphi \equiv \boxed{\varphi_{\textsc{context}}} \wedge \varphi_{\textsc{db}} \wedge \varphi_{\textsc{dep}\rightarrow} \wedge \varphi_{\rightarrow\textsc{dep}} \wedge \varphi_{\textsc{anomaly}}$$

$$\varphi \equiv \boxed{\varphi_{\text{CONTEXT}}} \wedge \varphi_{\text{DB}} \wedge \varphi_{\text{DEP}\rightarrow} \wedge \varphi_{\rightarrow\text{DEP}} \wedge \varphi_{\text{ANOMALY}}$$

- A set of constraints which must be satisfied by **any execution** of **any program**

$$\varphi \equiv \boxed{\varphi_{\text{CONTEXT}}} \wedge \varphi_{\text{DB}} \wedge \varphi_{\text{DEP}\rightarrow} \wedge \varphi_{\rightarrow\text{DEP}} \wedge \varphi_{\text{ANOMALY}}$$

- A set of constraints which must be satisfied by **any execution** of **any program**

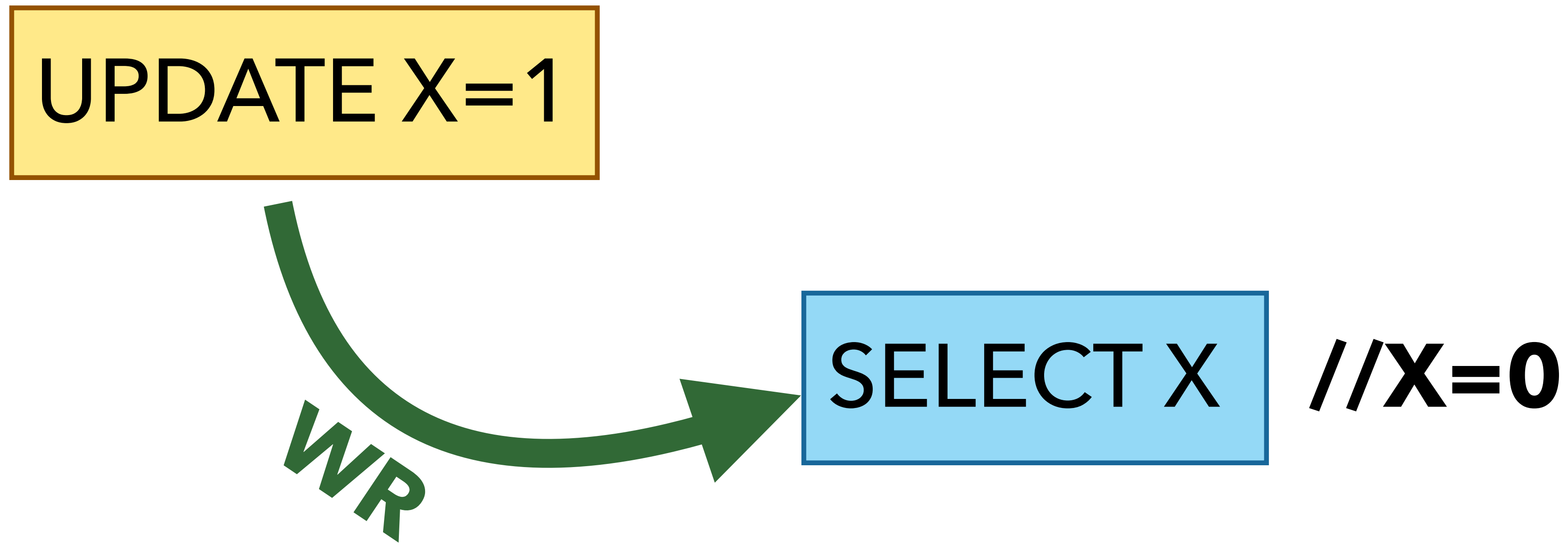

UPDATE X=1

SELECT X

//X=0

$$\varphi \equiv \boxed{\varphi_{\textsc{context}}} \wedge \varphi_{\textsc{db}} \wedge \varphi_{\textsc{dep}\rightarrow} \wedge \varphi_{\rightarrow\textsc{dep}} \wedge \varphi_{\textsc{anomaly}}$$

- A set of constraints which must be satisfied by **any execution** of **any program**



UPDATE X=1

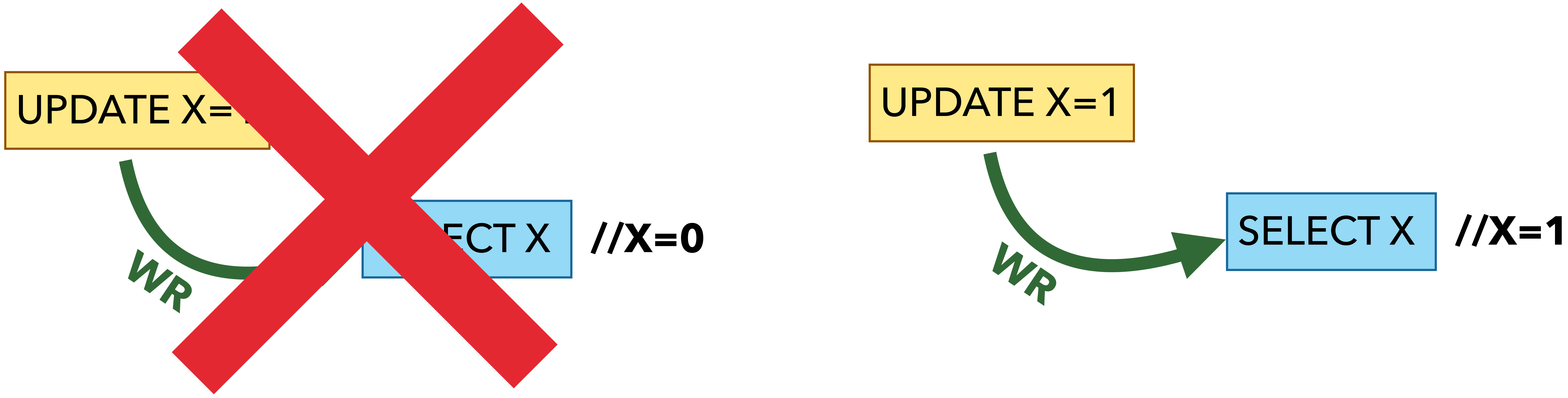
SELECT X

**//X=0**

**WR** induces the **same** read/written values

$$\varphi \equiv \boxed{\varphi_{\text{CONTEXT}}} \land \varphi_{\text{DB}} \land \varphi_{\text{DEP}\rightarrow} \land \varphi_{\rightarrow\text{DEP}} \land \varphi_{\text{ANOMALY}}$$

- A set of constraints which must be satisfied by **any execution** of **any program**



**WR** induces the **same** read/written values

$$\varphi \equiv \varphi_{\text{CONTEXT}} \land \boxed{\varphi_{\text{DB}}} \land \varphi_{\text{DEP}\rightarrow} \land \varphi_{\rightarrow\text{DEP}} \land \varphi_{\text{ANOMALY}}$$

$$\varphi \equiv \varphi_{\textsc{context}} \wedge \boxed{\varphi_{\textsc{db}}} \wedge \varphi_{\textsc{dep}\rightarrow} \wedge \varphi_{\rightarrow\textsc{dep}} \wedge \varphi_{\textsc{anomaly}}$$

- Includes a set of user-defined constraints on records

## $\varphi \equiv \varphi_{\text{CONTEXT}} \wedge \boxed{\varphi_{\text{DB}}} \wedge \varphi_{\text{DEP}\rightarrow} \wedge \varphi_{\rightarrow\text{DEP}} \wedge \varphi_{\text{ANOMALY}}$

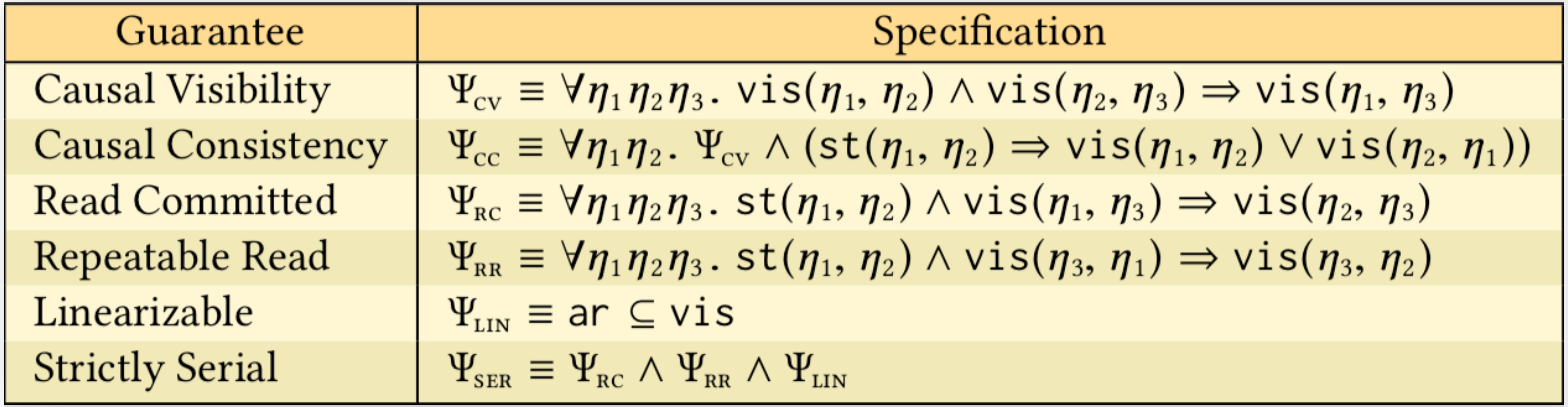
- Includes a set of user-defined constraints on records
  - e.g. *"all customer records must be older than 21"*

## $\varphi \equiv \varphi_{\text{CONTEXT}} \wedge \boxed{\varphi_{\text{DB}}} \wedge \varphi_{\text{DEP}\rightarrow} \wedge \varphi_{\rightarrow\text{DEP}} \wedge \varphi_{\text{ANOMALY}}$

- Includes a set of user-defined constraints on records
  - e.g. *“all customer records must be older than 21”*
- Includes database-specific consistency and isolation constraints

$$\varphi \equiv \varphi_{\text{CONTEXT}} \wedge \boxed{\varphi_{\text{DB}}} \wedge \varphi_{\text{DEP}\rightarrow} \wedge \varphi_{\rightarrow\text{DEP}} \wedge \varphi_{\text{ANOMALY}}$$

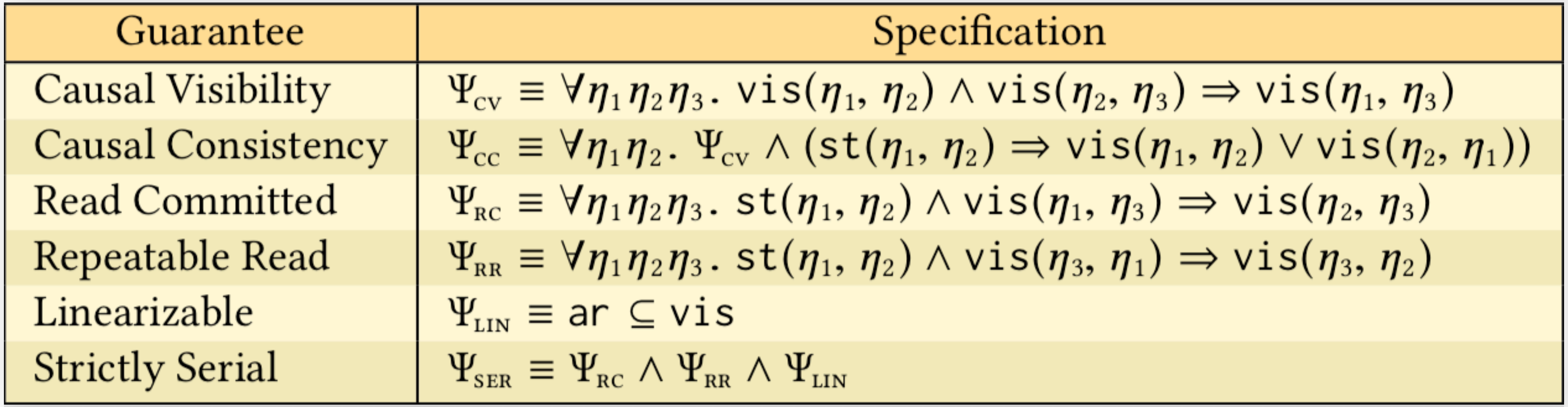
- Includes a set of user-defined constraints on records
  - e.g. *"all customer records must be older than 21"*
- Includes database-specific consistency and isolation constraints



| Guarantee | Specification |
|---|---|
| Causal Visibility | $\Psi_{\text{CV}} \equiv \forall \eta_1 \eta_2 \eta_3.\ \mathtt{vis}(\eta_1, \eta_2) \wedge \mathtt{vis}(\eta_2, \eta_3) \Rightarrow \mathtt{vis}(\eta_1, \eta_3)$ |
| Causal Consistency | $\Psi_{\text{CC}} \equiv \forall \eta_1 \eta_2.\ \Psi_{\text{CV}} \wedge (\mathtt{st}(\eta_1, \eta_2) \Rightarrow \mathtt{vis}(\eta_1, \eta_2) \vee \mathtt{vis}(\eta_2, \eta_1))$ |
| Read Committed | $\Psi_{\text{RC}} \equiv \forall \eta_1 \eta_2 \eta_3.\ \mathtt{st}(\eta_1, \eta_2) \wedge \mathtt{vis}(\eta_1, \eta_3) \Rightarrow \mathtt{vis}(\eta_2, \eta_3)$ |
| Repeatable Read | $\Psi_{\text{RR}} \equiv \forall \eta_1 \eta_2 \eta_3.\ \mathtt{st}(\eta_1, \eta_2) \wedge \mathtt{vis}(\eta_3, \eta_1) \Rightarrow \mathtt{vis}(\eta_3, \eta_2)$ |
| Linearizable | $\Psi_{\text{LIN}} \equiv \mathtt{ar} \subseteq \mathtt{vis}$ |
| Strictly Serial | $\Psi_{\text{SER}} \equiv \Psi_{\text{RC}} \wedge \Psi_{\text{RR}} \wedge \Psi_{\text{LIN}}$ |

$$\varphi \equiv \varphi_{\text{CONTEXT}} \wedge \boxed{\varphi_{\text{DB}}} \wedge \varphi_{\text{DEP}\rightarrow} \wedge \varphi_{\rightarrow\text{DEP}} \wedge \varphi_{\text{ANOMALY}}$$

- Includes a set of user-defined constraints on records
  - e.g. *"all customer records must be older than 21"*
- Includes database-specific consistency and isolation constraints



| Guarantee | Specification |
|---|---|
| Causal Visibility | $\Psi_{\text{CV}} \equiv \forall \eta_1\eta_2\eta_3.\ \texttt{vis}(\eta_1,\eta_2) \wedge \texttt{vis}(\eta_2,\eta_3) \Rightarrow \texttt{vis}(\eta_1,\eta_3)$ |
| Causal Consistency | $\Psi_{\text{CC}} \equiv \forall \eta_1\eta_2.\ \Psi_{\text{CV}} \wedge (\texttt{st}(\eta_1,\eta_2) \Rightarrow \texttt{vis}(\eta_1,\eta_2) \vee \texttt{vis}(\eta_2,\eta_1))$ |
| Read Committed | $\Psi_{\text{RC}} \equiv \forall \eta_1\eta_2\eta_3.\ \texttt{st}(\eta_1,\eta_2) \wedge \texttt{vis}(\eta_1,\eta_3) \Rightarrow \texttt{vis}(\eta_2,\eta_3)$ |
| Repeatable Read | $\Psi_{\text{RR}} \equiv \forall \eta_1\eta_2\eta_3.\ \texttt{st}(\eta_1,\eta_2) \wedge \texttt{vis}(\eta_3,\eta_1) \Rightarrow \texttt{vis}(\eta_3,\eta_2)$ |
| Linearizable | $\Psi_{\text{LIN}} \equiv \texttt{ar} \subseteq \texttt{vis}$ |
| Strictly Serial | $\Psi_{\text{SER}} \equiv \Psi_{\text{RC}} \wedge \Psi_{\text{RR}} \wedge \Psi_{\text{LIN}}$ |

**Only executions valid for the database abstraction are constructed**

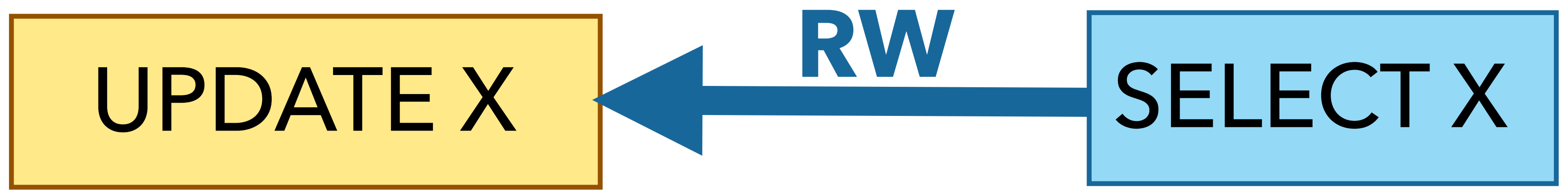
$$\varphi \equiv \varphi_{\textsc{context}} \wedge \varphi_{\textsc{db}} \wedge \boxed{\varphi_{\textsc{dep}\rightarrow}} \wedge \varphi_{\rightarrow\textsc{dep}} \wedge \varphi_{\textsc{anomaly}}$$

$$\varphi \equiv \varphi_{\text{CONTEXT}} \wedge \varphi_{\text{DB}} \wedge \boxed{\varphi_{\text{DEP}\rightarrow}} \wedge \varphi_{\rightarrow\text{DEP}} \wedge \varphi_{\text{ANOMALY}}$$

- **Necessary** conditions to establish a dependency relation between two operation instances

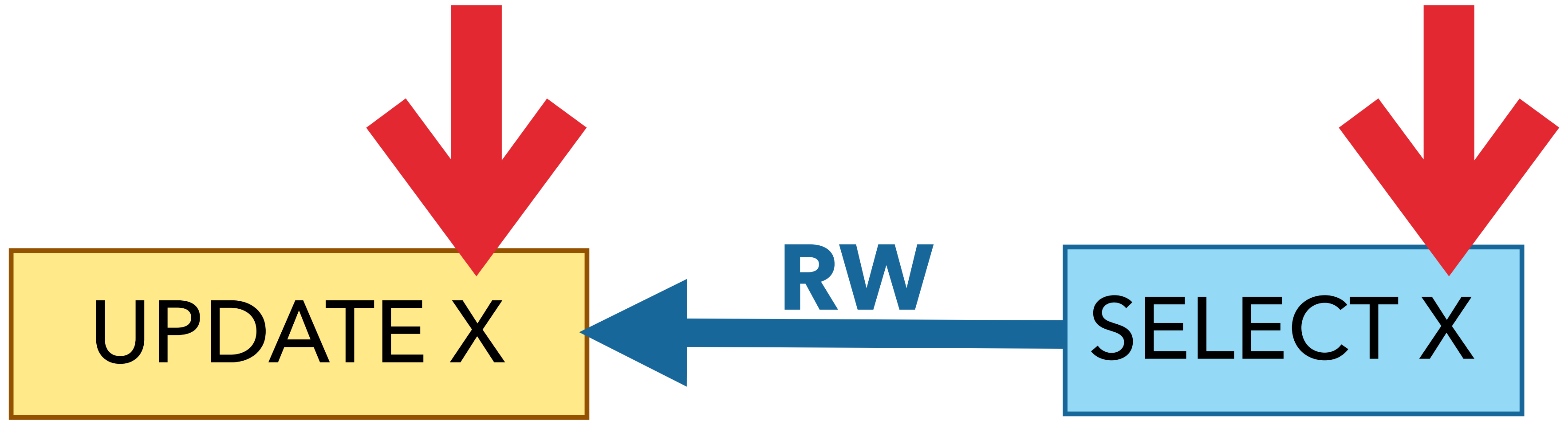
$$\varphi \equiv \varphi_{\text{CONTEXT}} \wedge \varphi_{\text{DB}} \wedge \boxed{\varphi_{\text{DEP}\rightarrow}} \wedge \varphi_{\rightarrow\text{DEP}} \wedge \varphi_{\text{ANOMALY}}$$

- **Necessary** conditions to establish a dependency relation between two operation instances



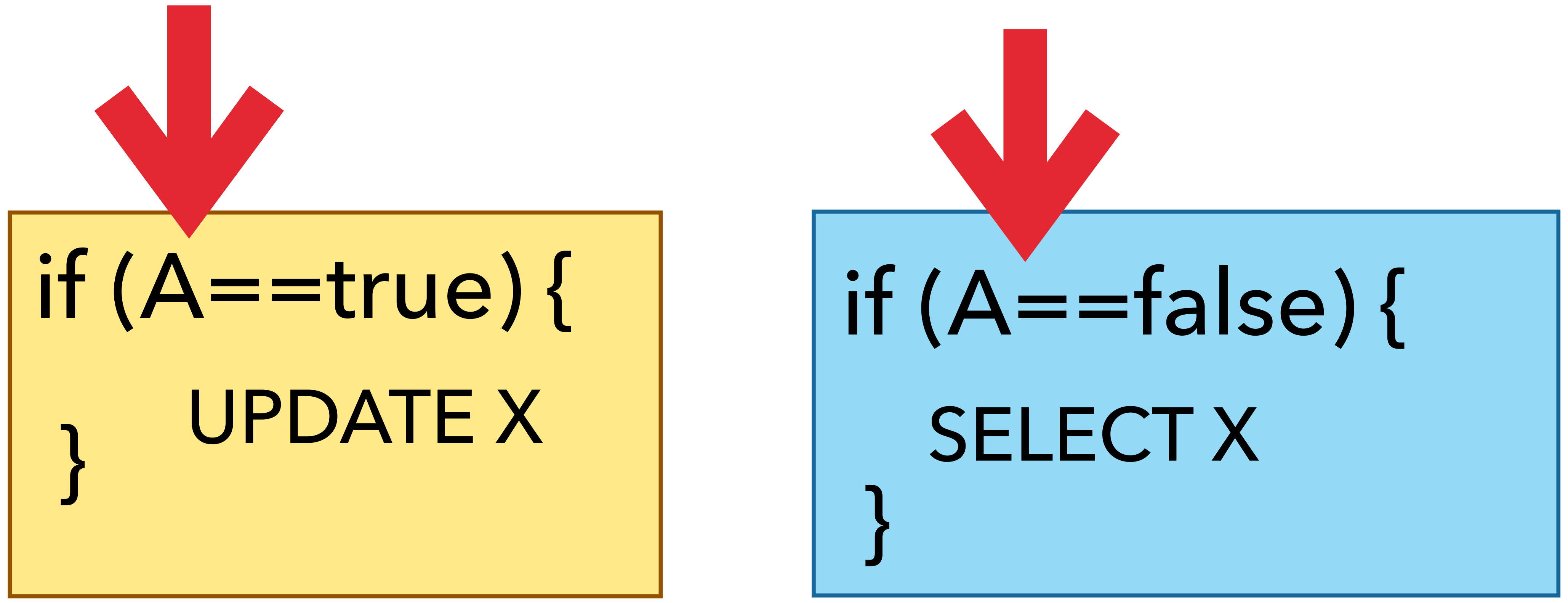
$$\varphi \equiv \varphi_{\text{CONTEXT}} \wedge \varphi_{\text{DB}} \wedge \boxed{\varphi_{\text{DEP}\rightarrow}} \wedge \varphi_{\rightarrow\text{DEP}} \wedge \varphi_{\text{ANOMALY}}$$

- **Necessary** conditions to establish a dependency relation between two operation instances
  - There is a mutually accessed record



$$\varphi \equiv \varphi_{\text{CONTEXT}} \wedge \varphi_{\text{DB}} \wedge \boxed{\varphi_{\text{DEP}\rightarrow}} \wedge \varphi_{\rightarrow\text{DEP}} \wedge \varphi_{\text{ANOMALY}}$$

- **Necessary** conditions to establish a valid dependency relation between two operation instances
  - There is a mutually accessed record
  - Both operations are simultaneously reached by the control flow

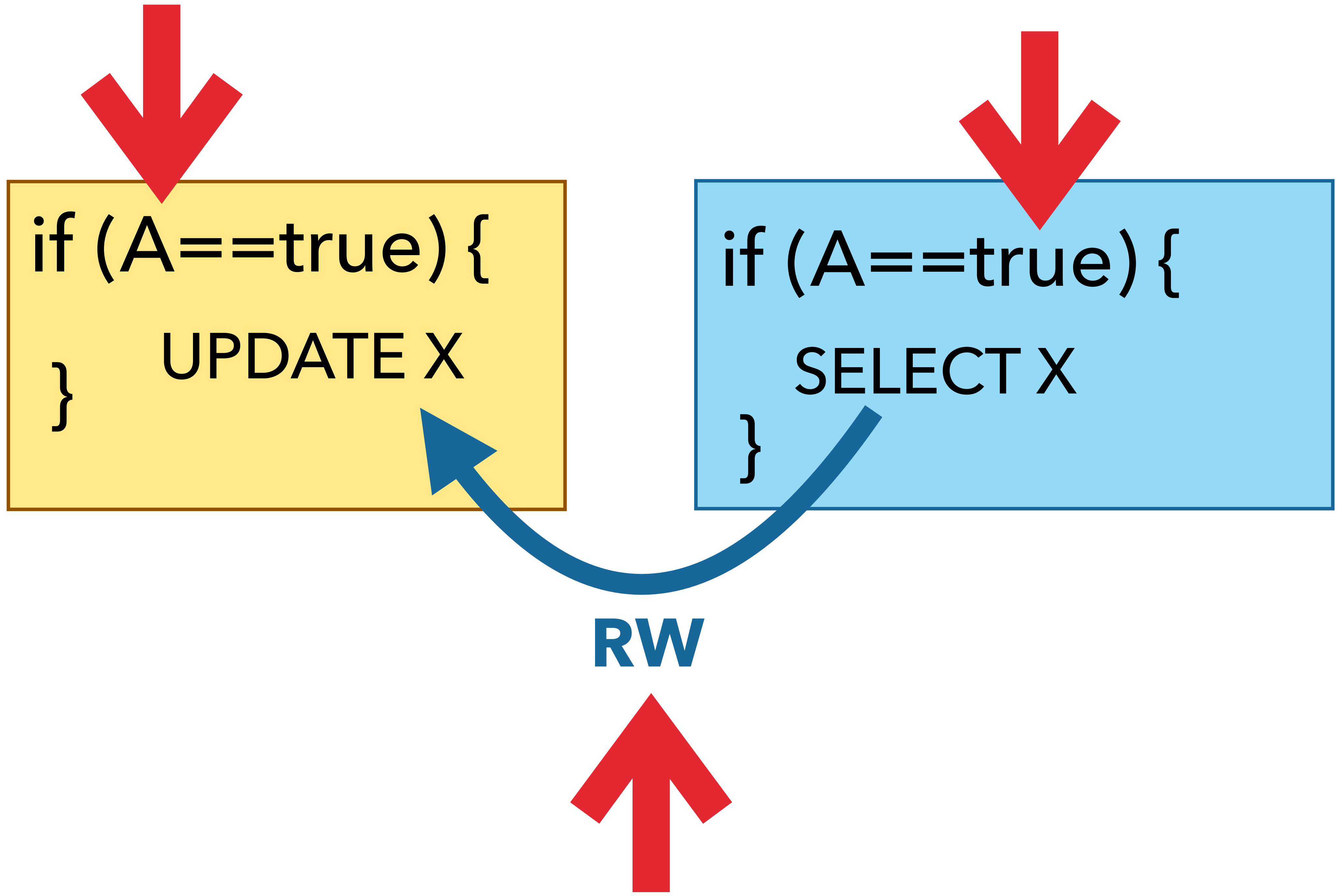


```
if (A==true) {
      UPDATE X
}
```

```
if (A==false) {
    SELECT X
}
```

$$\varphi \equiv \varphi_{\text{CONTEXT}} \wedge \varphi_{\text{DB}} \wedge \boxed{\varphi_{\text{DEP}\rightarrow}} \wedge \varphi_{\rightarrow\text{DEP}} \wedge \varphi_{\text{ANOMALY}}$$

- **Necessary** conditions to establish a valid dependency relation between two operation instances
  - There is a mutually accessed record
  - Both operations are simultaneously reached by the control flow



```
if (A==true) {
    UPDATE X
}
```

```
if (A==true) {
    SELECT X
}
```

$$\varphi \equiv \varphi_{\textsc{context}} \wedge \varphi_{\textsc{db}} \wedge \varphi_{\textsc{dep}\rightarrow} \wedge \boxed{\varphi_{\rightarrow\textsc{dep}}} \wedge \varphi_{\textsc{anomaly}}$$

$$\varphi \equiv \varphi_{\textsc{context}} \wedge \varphi_{\textsc{db}} \wedge \varphi_{\textsc{dep}\rightarrow} \wedge \boxed{\varphi_{\rightarrow\textsc{dep}}} \wedge \varphi_{\textsc{anomaly}}$$

- **Sufficient** conditions to establish a dependency relation between two operation instances

$$\varphi \equiv \varphi_{\text{CONTEXT}} \land \varphi_{\text{DB}} \land \varphi_{\text{DEP}\rightarrow} \land \boxed{\varphi_{\rightarrow\text{DEP}}} \land \varphi_{\text{ANOMALY}}$$

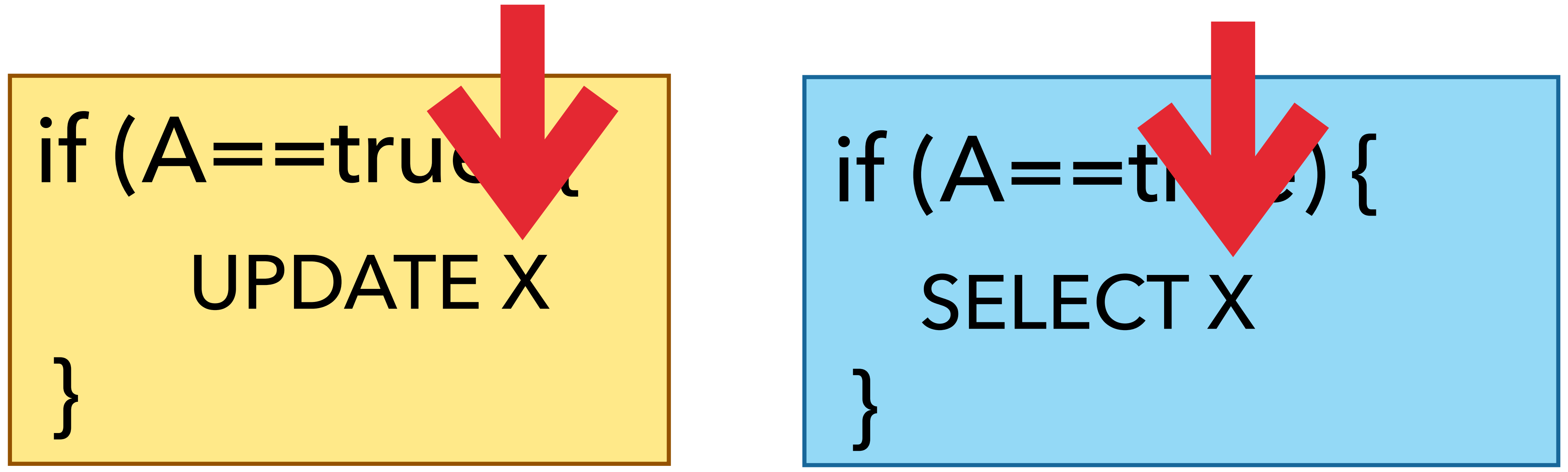
- **Sufficient** conditions to establish a dependency relation between two operation instances

```
if (A==true) {
        UPDATE X
}
```

```
if (A==true) {
    SELECT X
}
```

$$\varphi \equiv \varphi_{\text{CONTEXT}} \wedge \varphi_{\text{DB}} \wedge \varphi_{\text{DEP}\rightarrow} \wedge \boxed{\varphi_{\rightarrow\text{DEP}}} \wedge \varphi_{\text{ANOMALY}}$$

- **Sufficient** conditions to establish a dependency relation between two operation instances
  - **If** there is a mutually accessed record

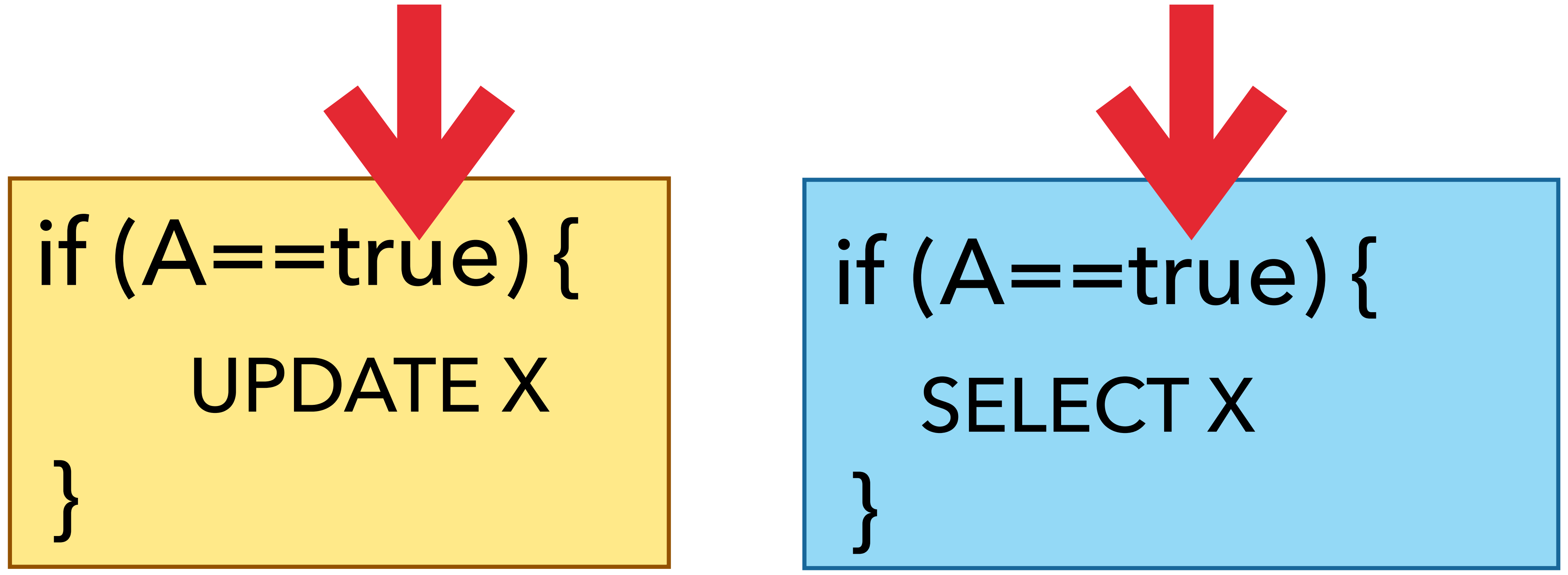


```
if (A==true) {
      UPDATE X
}
```

```
if (A==true) {
    SELECT X
}
```

$$\varphi \equiv \varphi_{\text{CONTEXT}} \wedge \varphi_{\text{DB}} \wedge \varphi_{\text{DEP}\rightarrow} \wedge \boxed{\varphi_{\rightarrow\text{DEP}}} \wedge \varphi_{\text{ANOMALY}}$$

- **Sufficient** conditions to establish a dependency relation between two operation instances
  - **If** there is a mutually accessed record
  - **and** both operations are reached

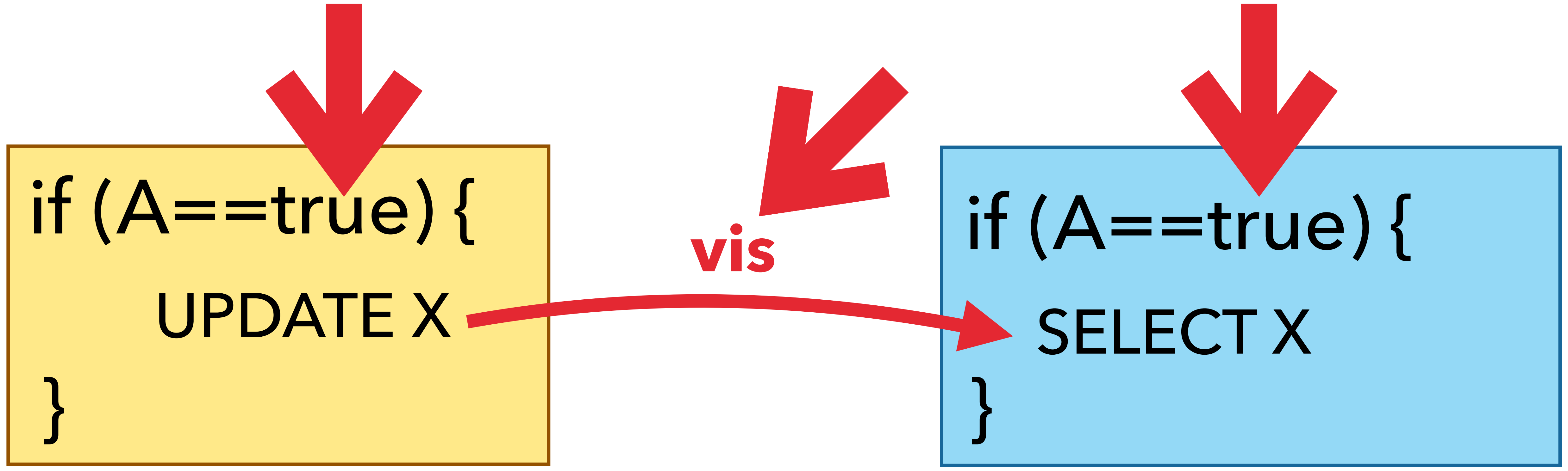


```
if (A==true) {
        UPDATE X
}
```

```
if (A==true) {
     SELECT X
}
```

$$\varphi \equiv \varphi_{\text{CONTEXT}} \wedge \varphi_{\text{DB}} \wedge \varphi_{\text{DEP}\rightarrow} \wedge \boxed{\varphi_{\rightarrow\text{DEP}}} \wedge \varphi_{\text{ANOMALY}}$$

- **Sufficient** conditions to establish a dependency relation between two operation instances
  - **If** there is a mutually accessed record
  - **and** both operations are reached
  - **and** the update is visible to the select



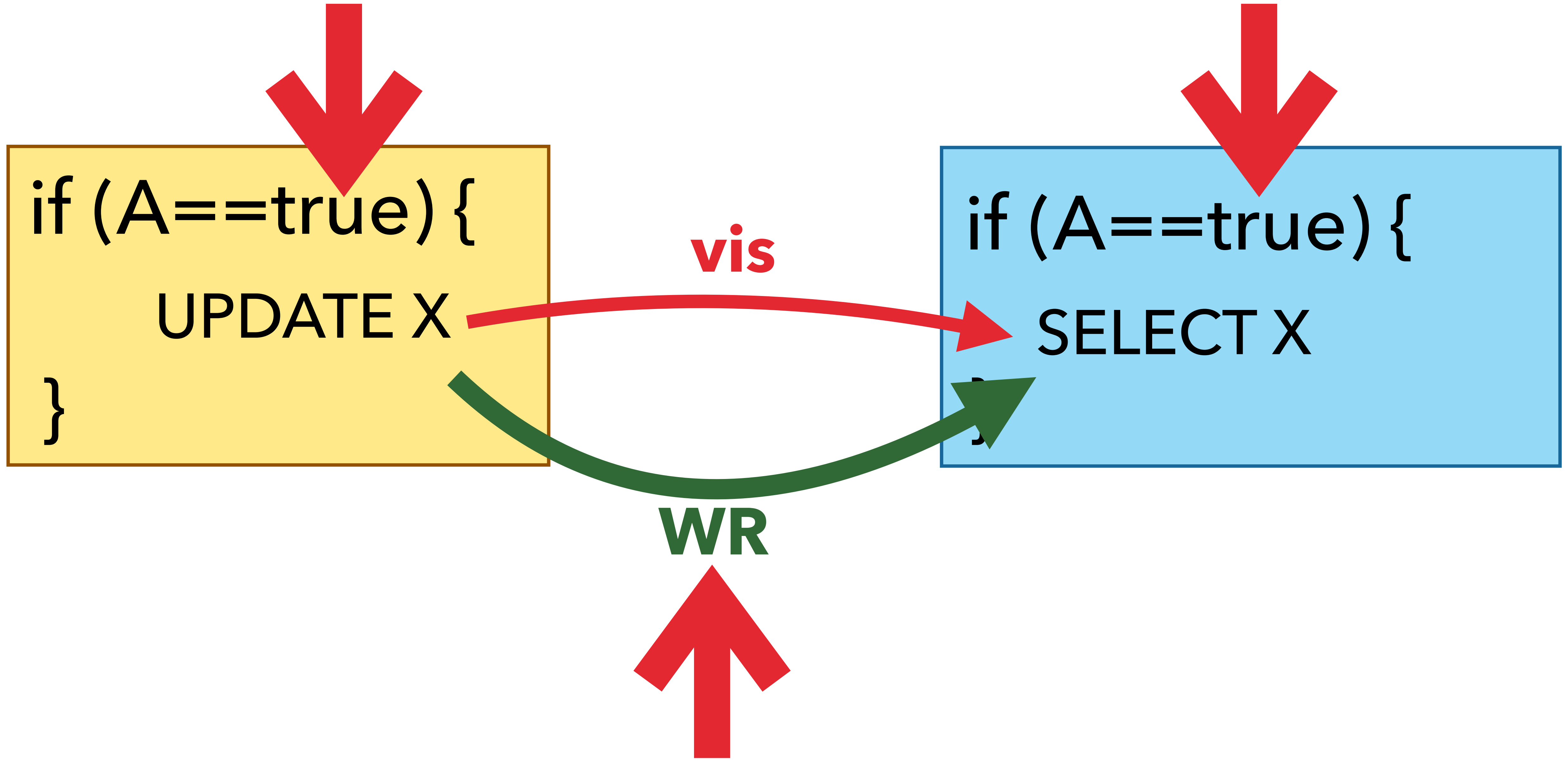
```
if (A==true) {
      UPDATE X
}
```

```
if (A==true) {
    SELECT X
}
```

$$\varphi \equiv \varphi_{\text{CONTEXT}} \wedge \varphi_{\text{DB}} \wedge \varphi_{\text{DEP}\rightarrow} \wedge \boxed{\varphi_{\rightarrow\text{DEP}}} \wedge \varphi_{\text{ANOMALY}}$$

- **Sufficient** conditions to establish a dependency relation between two operation instances
  - **If** there is a mutually accessed record
  - **and** both operations are reached
  - **and** the update is visible to the select

Operations **must** be dependent by **WR**



```
if (A==true) {
      UPDATE X
}
```

```
if (A==true) {
   SELECT X
}
```

$$\varphi \equiv \varphi_{\textsc{context}} \wedge \varphi_{\textsc{db}} \wedge \varphi_{\textsc{dep}\rightarrow} \wedge \varphi_{\rightarrow\textsc{dep}} \wedge \boxed{\varphi_{\textsc{anomaly}}}$$

$$\varphi \equiv \varphi_{\textsc{context}} \wedge \varphi_{\textsc{db}} \wedge \varphi_{\textsc{dep}\rightarrow} \wedge \varphi_{\rightarrow\textsc{dep}} \wedge \boxed{\varphi_{\textsc{anomaly}}}$$

- Enforces the existence of an anomaly

$$\varphi \equiv \varphi_{\textsc{context}} \wedge \varphi_{\textsc{db}} \wedge \varphi_{\textsc{dep}\rightarrow} \wedge \varphi_{\rightarrow\textsc{dep}} \wedge \boxed{\varphi_{\textsc{anomaly}}}$$

- Enforces the existence of an anomaly
- Parametrized over three variables: **i**, **j** and **k**

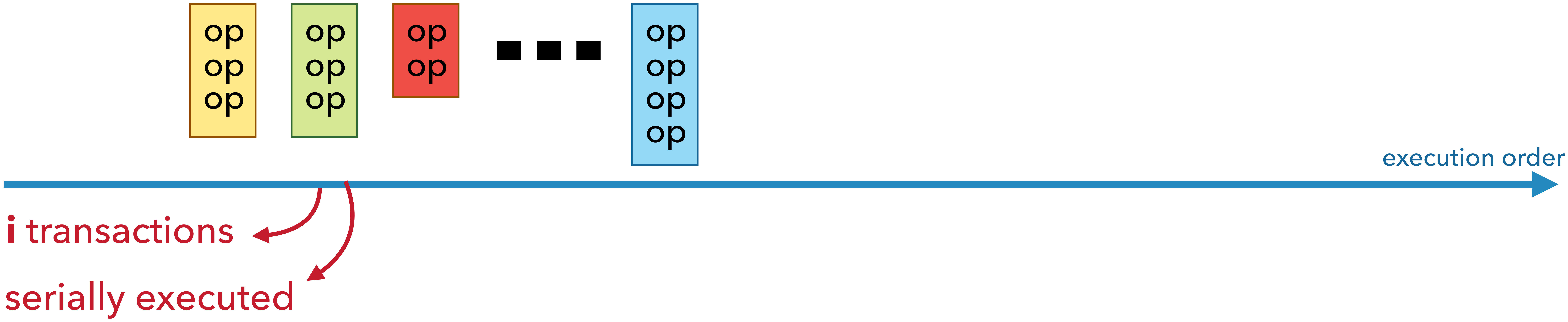
$$\varphi \equiv \varphi_{\text{CONTEXT}} \wedge \varphi_{\text{DB}} \wedge \varphi_{\text{DEP}\rightarrow} \wedge \varphi_{\rightarrow\text{DEP}} \wedge \boxed{\varphi_{\text{ANOMALY}}}$$

- Enforces the existence of an anomaly
- Parametrized over three variables: **i**, **j** and **k**

Bounds on the state space

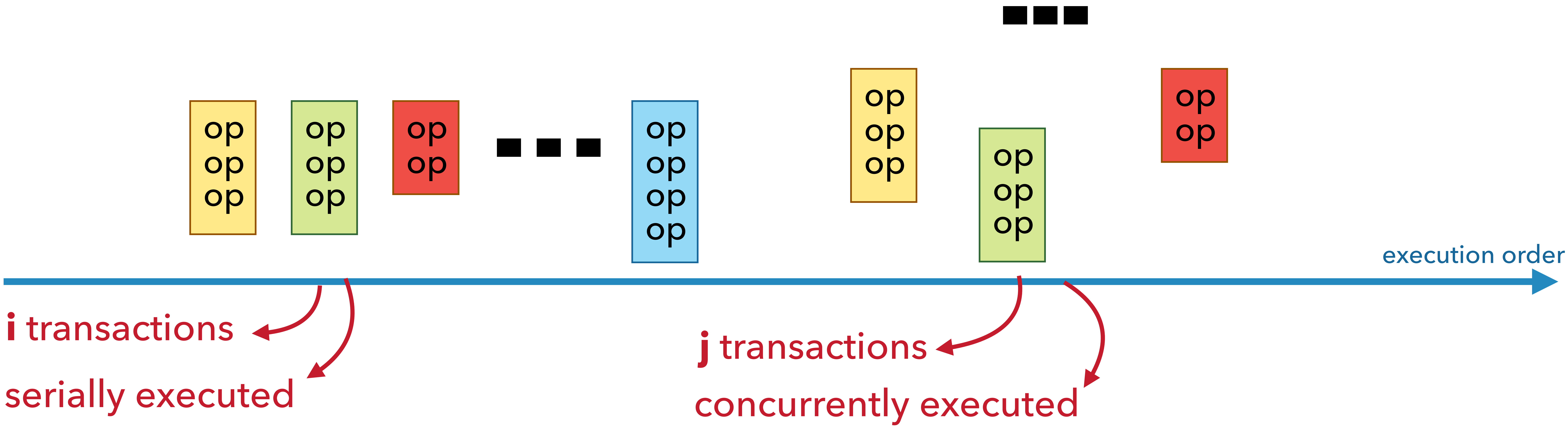
$$\varphi \equiv \varphi_{\text{CONTEXT}} \wedge \varphi_{\text{DB}} \wedge \varphi_{\text{DEP}\rightarrow} \wedge \varphi_{\rightarrow\text{DEP}} \wedge \boxed{\varphi_{\text{ANOMALY}}}$$

- Enforces the existence of an anomaly
- Parametrized over three variables: **i**, **j** and **k**
- Instantiates **i** serially executed transactions,



$$\varphi \equiv \varphi_{\text{CONTEXT}} \wedge \varphi_{\text{DB}} \wedge \varphi_{\text{DEP}\to} \wedge \varphi_{\to\text{DEP}} \wedge \boxed{\varphi_{\text{ANOMALY}}}$$

- Enforces the existence of an anomaly
- Parametrized over three variables: **i**, **j** and **k**
- Instantiates **i** serially executed transactions,
- leading to **j** concurrent transactions



execution order

**i** transactions

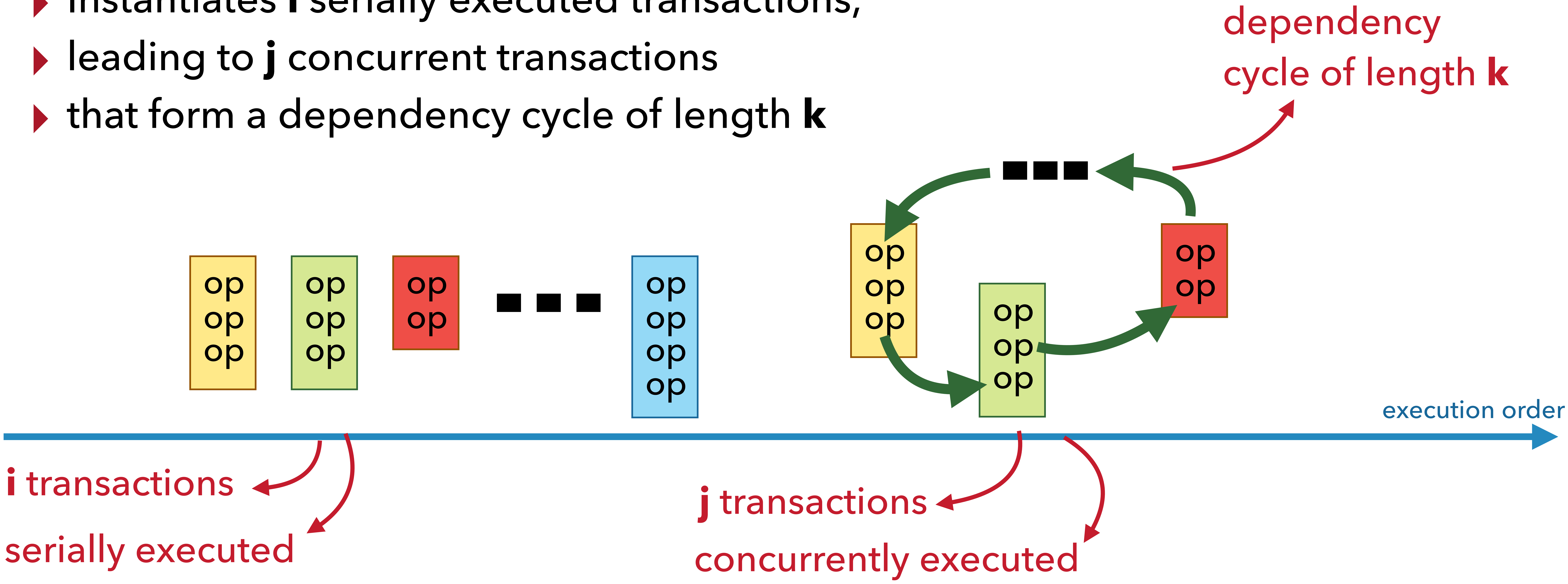
serially executed

**j** transactions

concurrently executed

$$\varphi \equiv \varphi_{\text{CONTEXT}} \wedge \varphi_{\text{DB}} \wedge \varphi_{\text{DEP}\rightarrow} \wedge \varphi_{\rightarrow\text{DEP}} \wedge \boxed{\varphi_{\text{ANOMALY}}}$$

- Enforces the existence of an anomaly
- Parametrized over three variables: **i**, **j** and **k**
- Instantiates **i** serially executed transactions,
- leading to **j** concurrent transactions
- that form a dependency cycle of length **k**



dependency cycle of length **k**

execution order

**i** transactions

serially executed

**j** transactions

concurrently executed

# TESTING; FUNDAMENTAL CHALLENGES (REVISITED)

## TESTING: FUNDAMENTAL CHALLENGES (REVISITED)

- Rich and precise encoding

## TESTING: FUNDAMENTAL CHALLENGES (REVISITED)

- Rich and precise encoding

$$\varphi$$

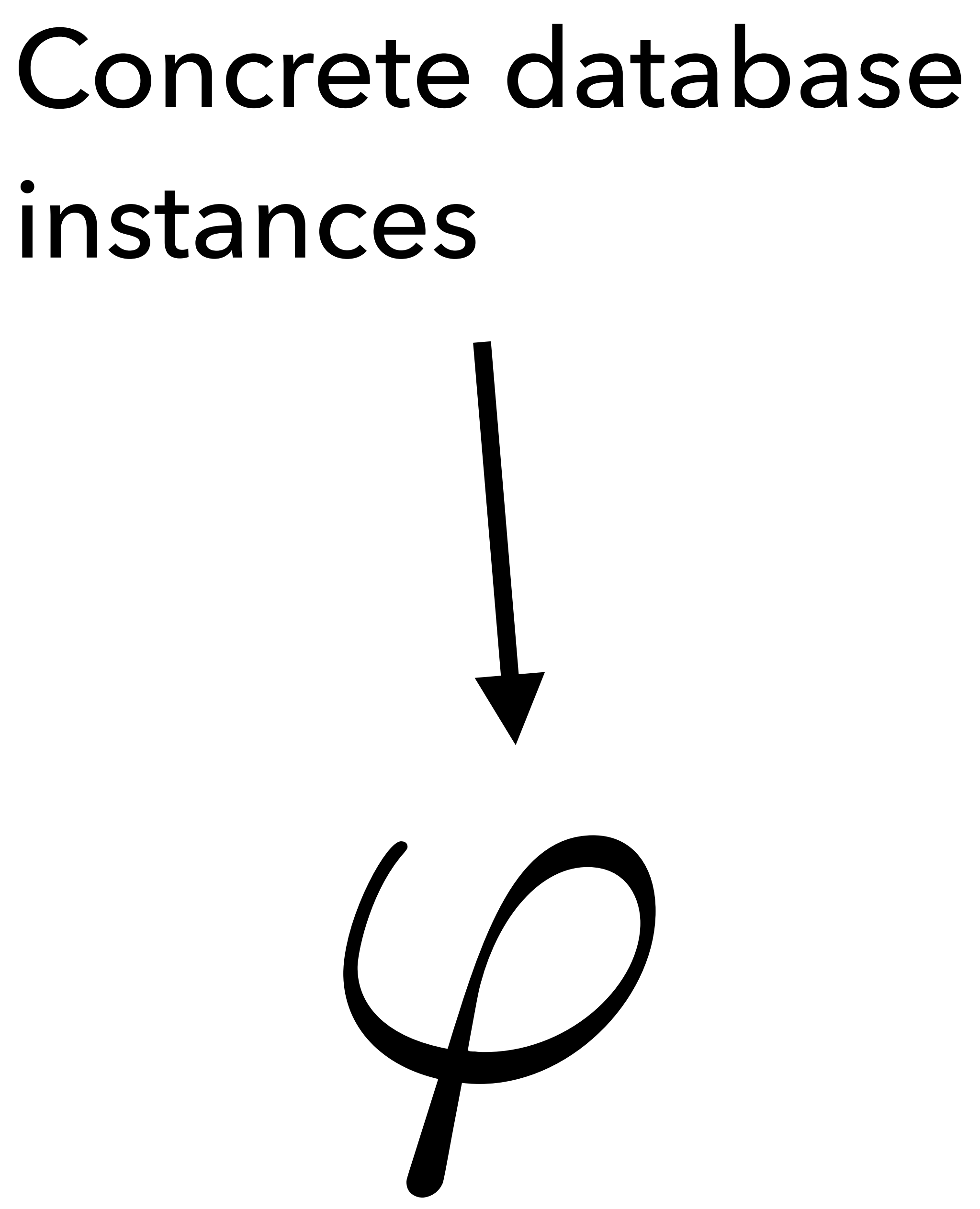
## TESTING: FUNDAMENTAL CHALLENGES (REVISITED)

- Rich and precise encoding
- Triggering anomalies requires determining:
  - Initial database state
  - Input arguments
  - Execution order
  - Network delays

$$\varphi$$

# TESTING: FUNDAMENTAL CHALLENGES (REVISITED)

- Rich and precise encoding
- Triggering anomalies requires determining:
  - Initial database state ✅
  - Input arguments
  - Execution order
  - Network delays

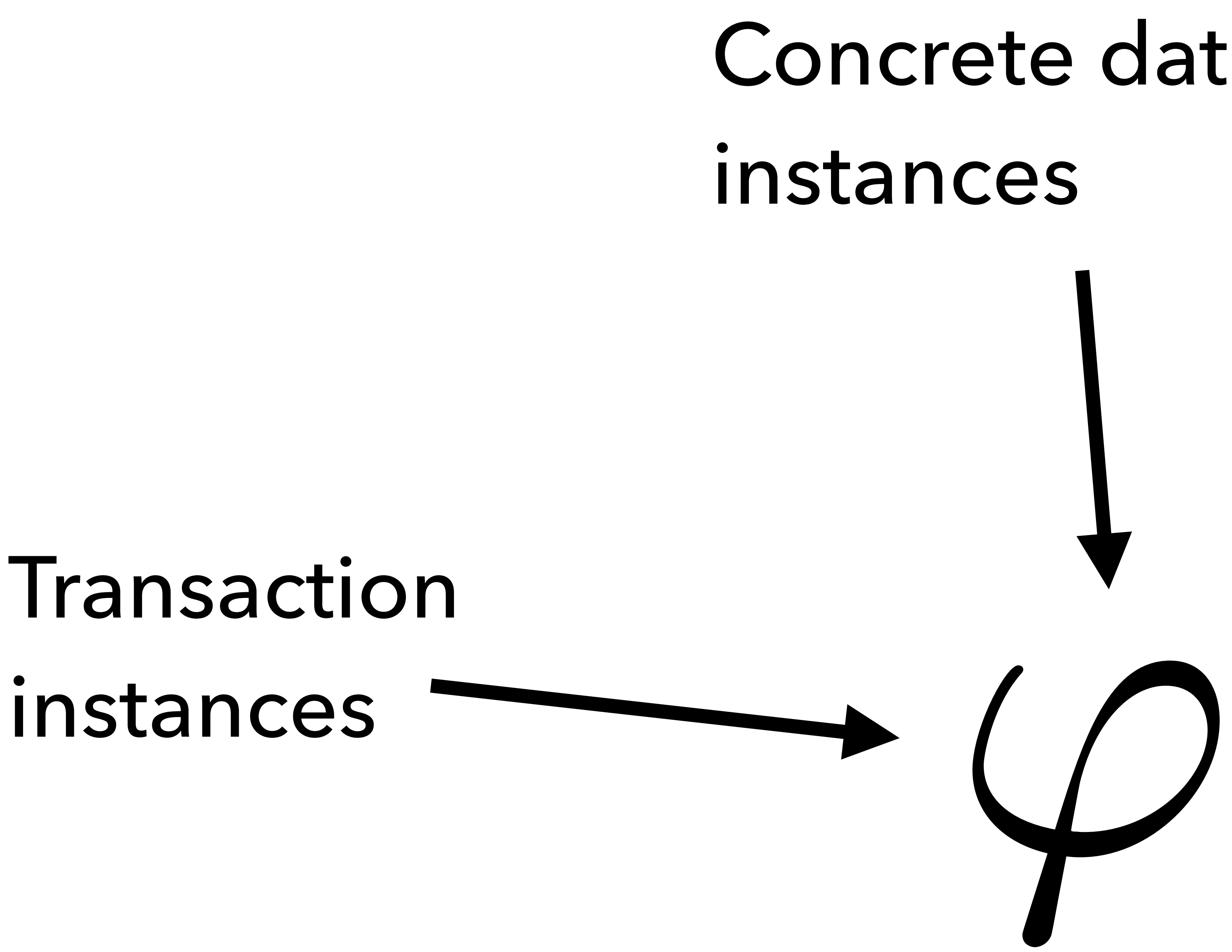


Concrete database instances

$\varphi$

# TESTING: FUNDAMENTAL CHALLENGES (REVISITED)

- Rich and precise encoding
- Triggering anomalies requires determining:
  - Initial database state ✅
  - Input arguments ✅
  - Execution order
  - Network delays



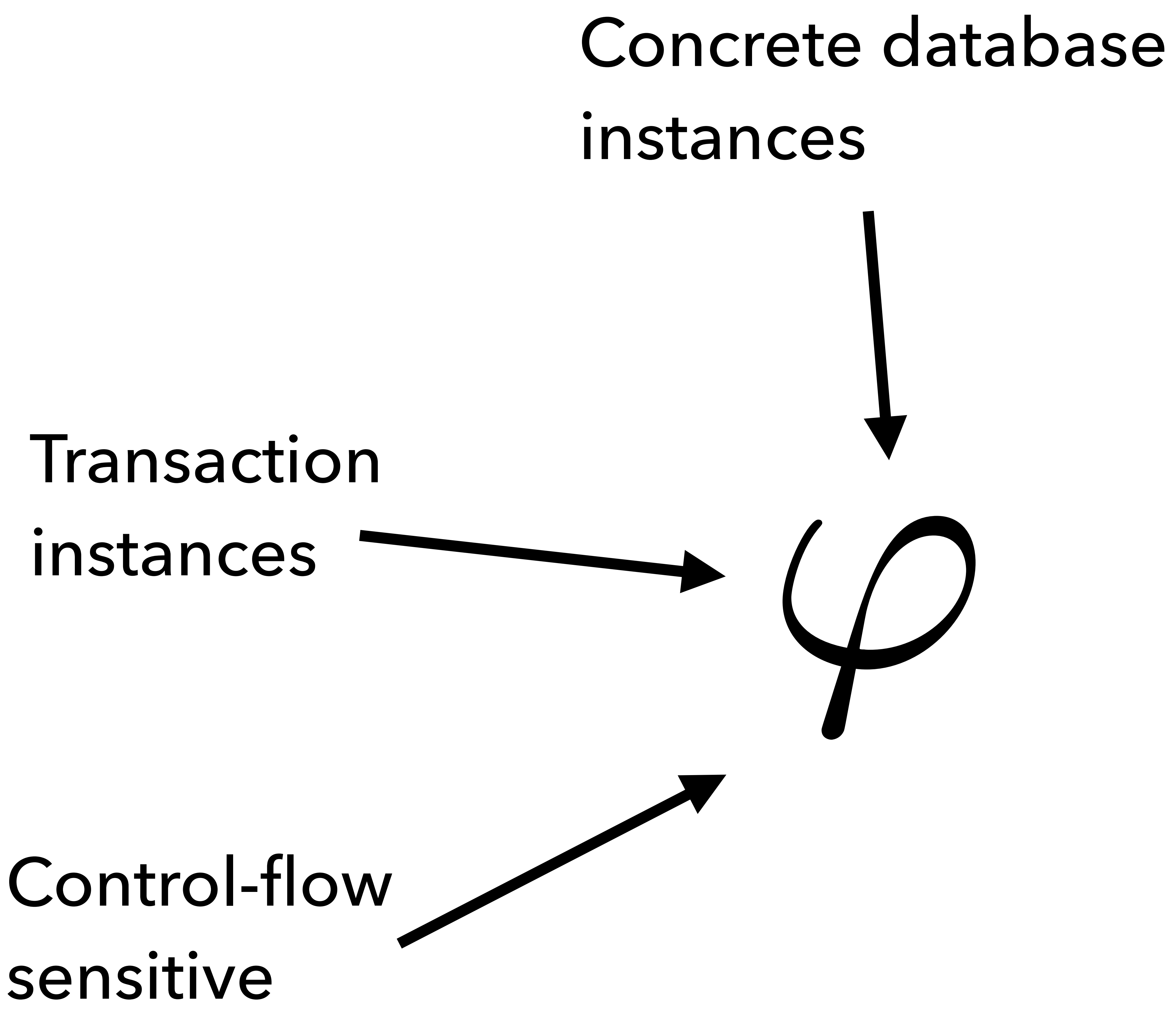
Concrete database instances

Transaction instances

$\varphi$

# TESTING: FUNDAMENTAL CHALLENGES (REVISITED)

- Rich and precise encoding
- Triggering anomalies requires determining:
  - Initial database state ✅
  - Input arguments ✅
  - Execution order
  - Network delays



Concrete database instances

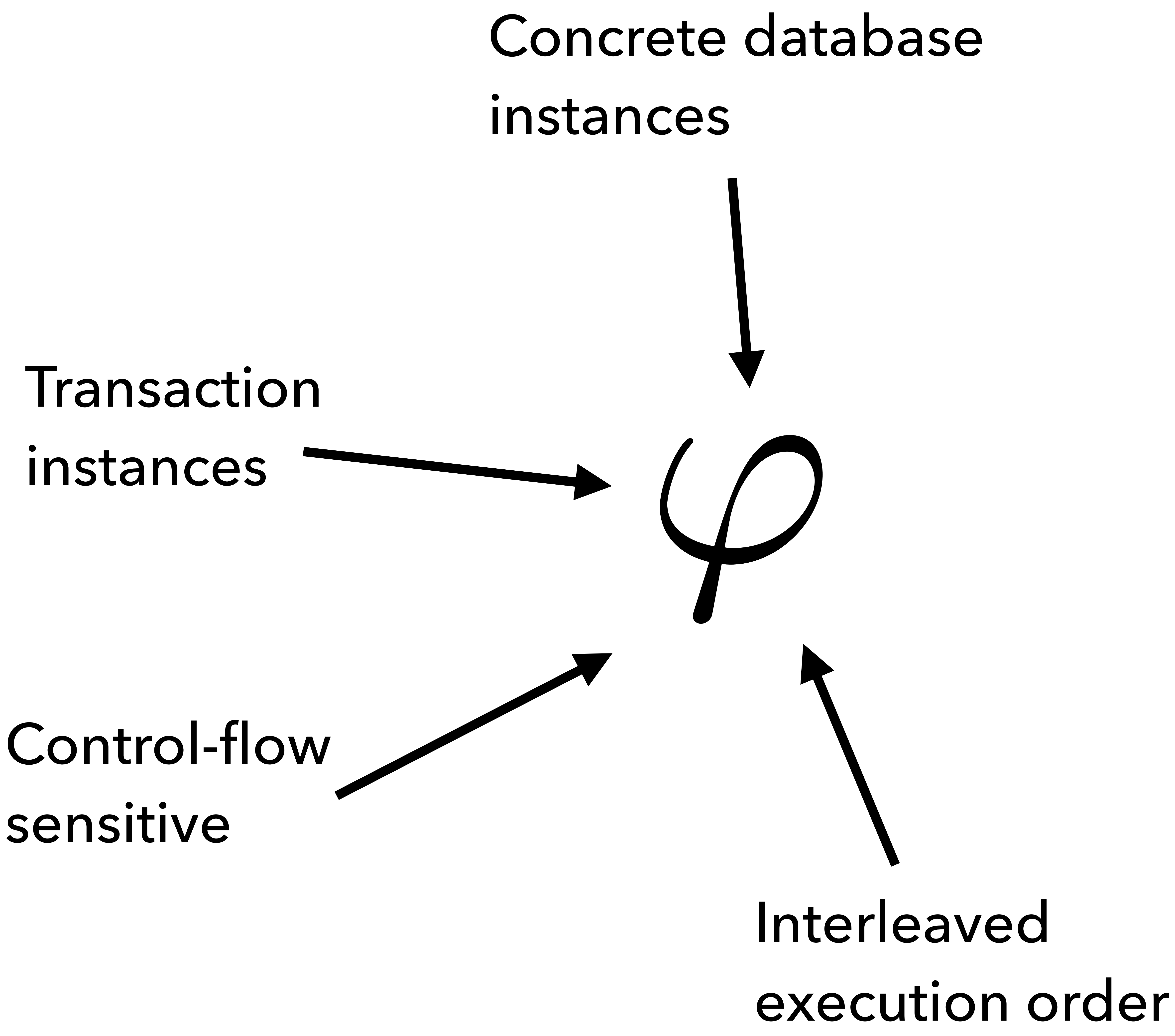
Transaction instances

Control-flow sensitive

$\varphi$

# TESTING: FUNDAMENTAL CHALLENGES (REVISITED)

- Rich and precise encoding
- Triggering anomalies requires determining:
  - Initial database state ✅
  - Input arguments ✅
  - Execution order ✅
  - Network delays



Concrete database instances

Transaction instances

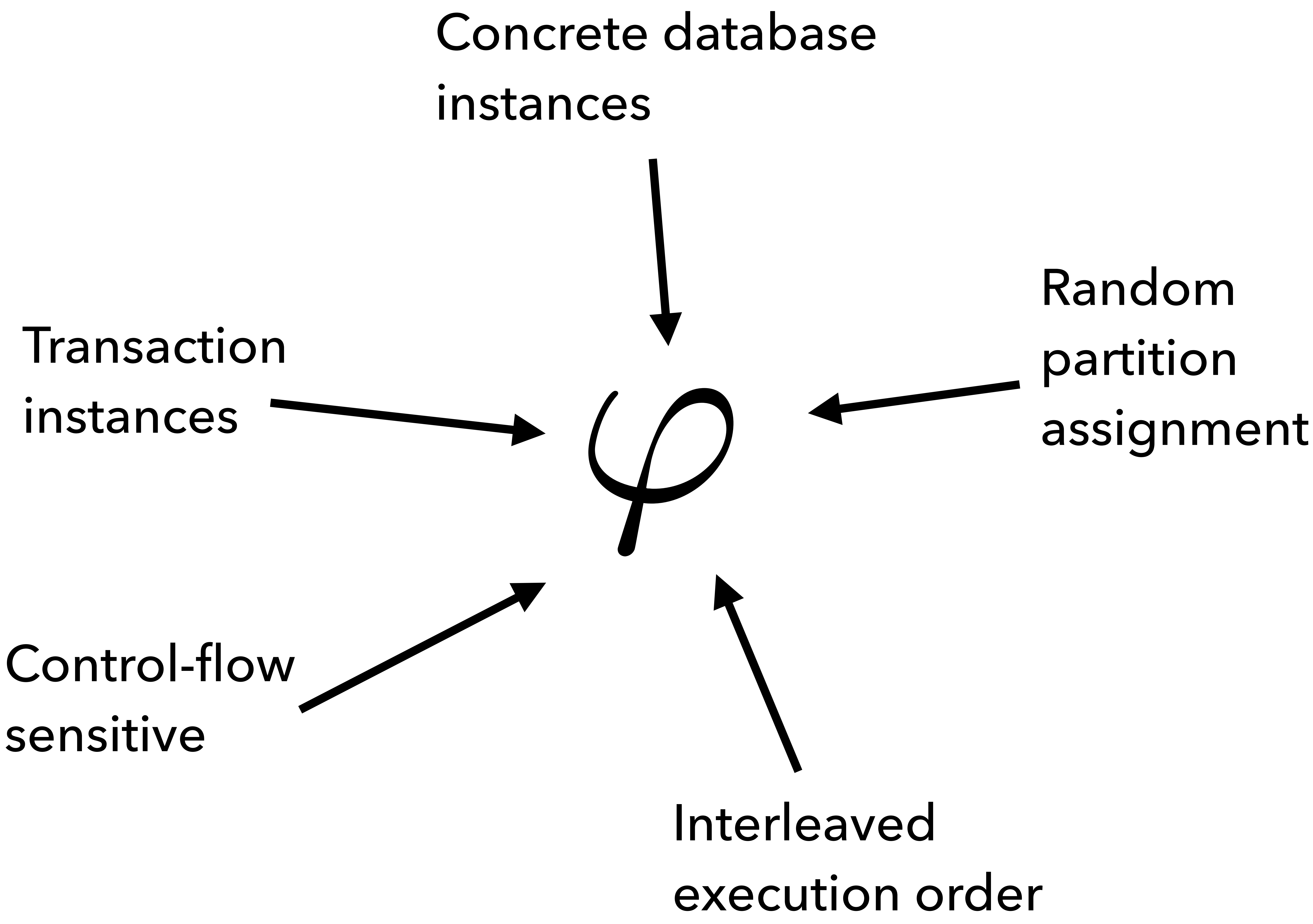
$\varphi$

Control-flow sensitive

Interleaved execution order

# TESTING: FUNDAMENTAL CHALLENGES (REVISITED)

- Rich and precise encoding
- Triggering anomalies requires determining:
  - Initial database state ✅
  - Input arguments ✅
  - Execution order ✅
  - Network delays ✅



Concrete database instances

Transaction instances

$\varphi$

Random partition assignment

Control-flow sensitive

Interleaved execution order

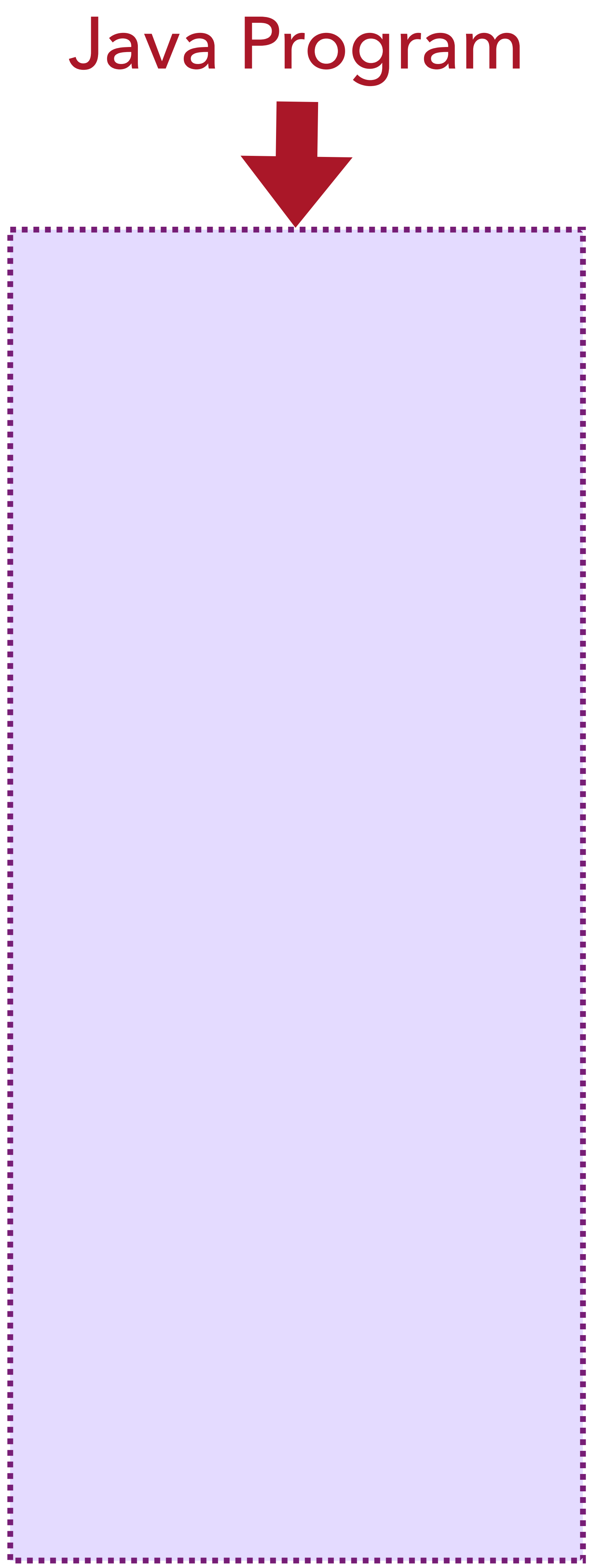
# CLOTHO: BUG DETECTION MECHANISM

# CLOTHO: BUG DETECTION MECHANISM

- Static analysis engine for java programs

# CLOTHO: BUG DETECTION MECHANISM

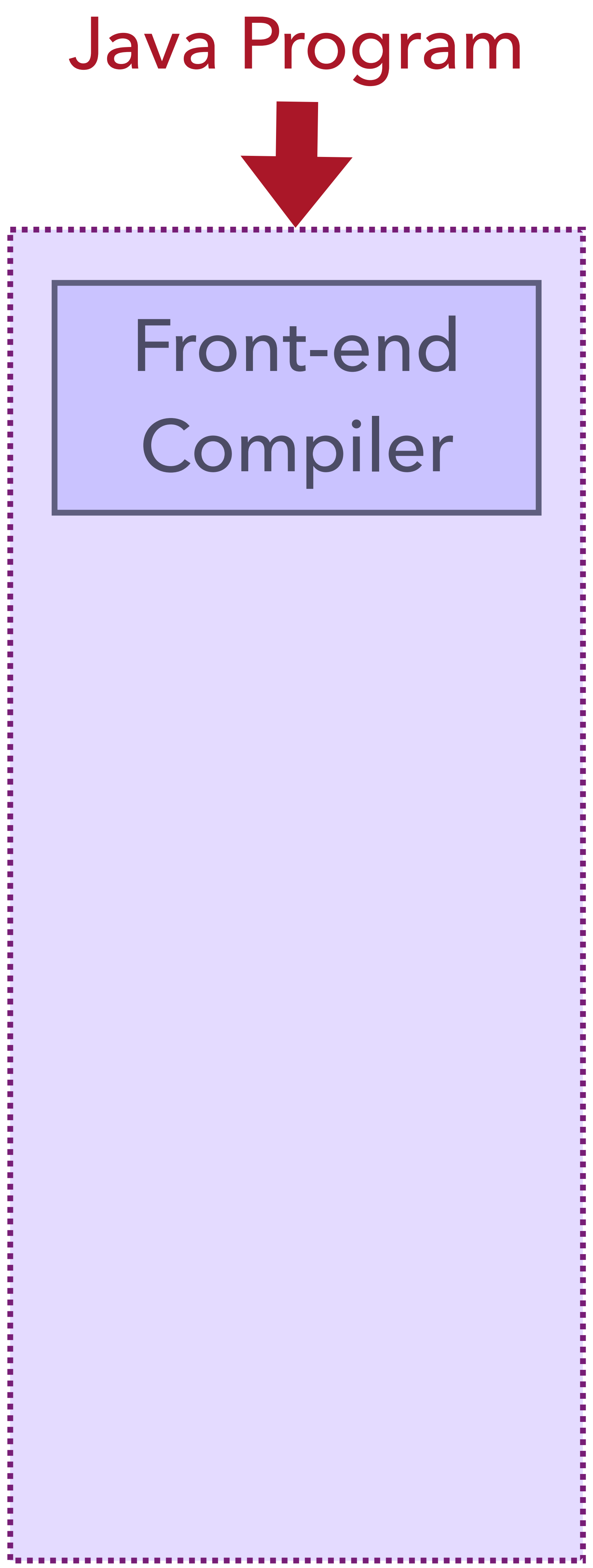
- Static analysis engine for java programs



Java Program

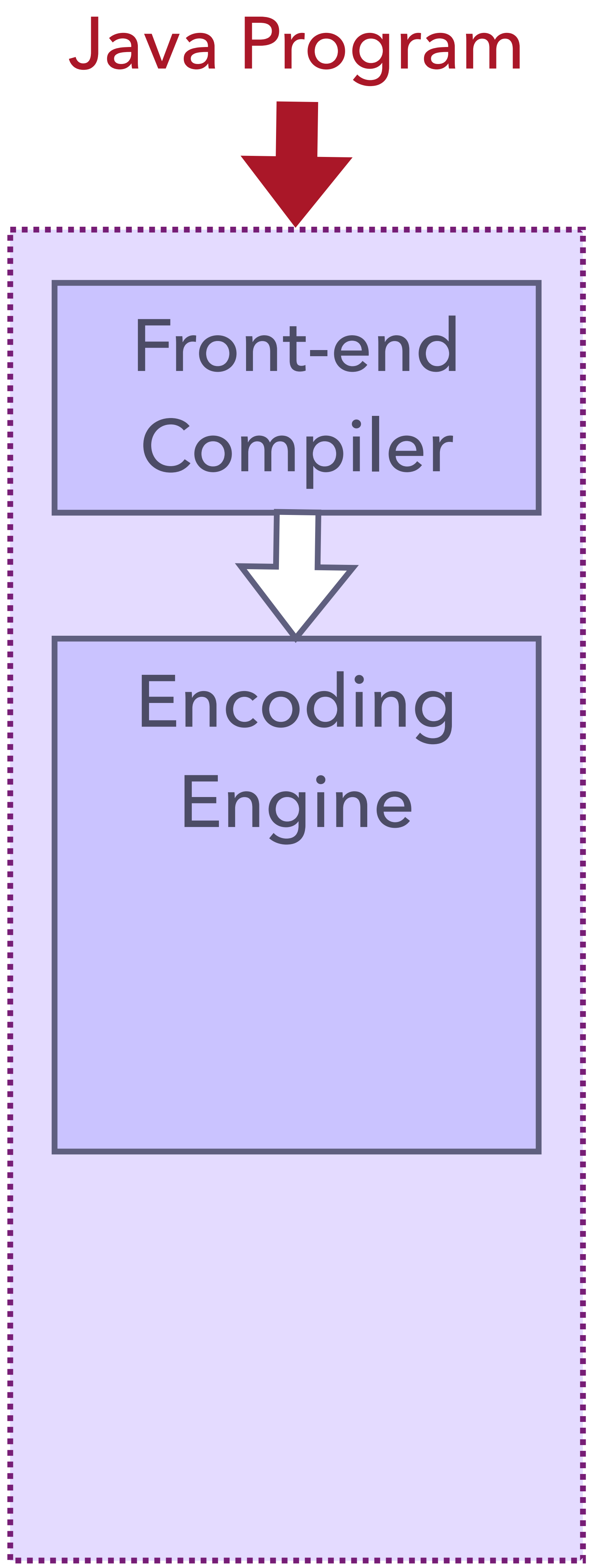
# CLOTHO: BUG DETECTION MECHANISM

- Static analysis engine for java programs
- Compiles programs down to an abstract representation



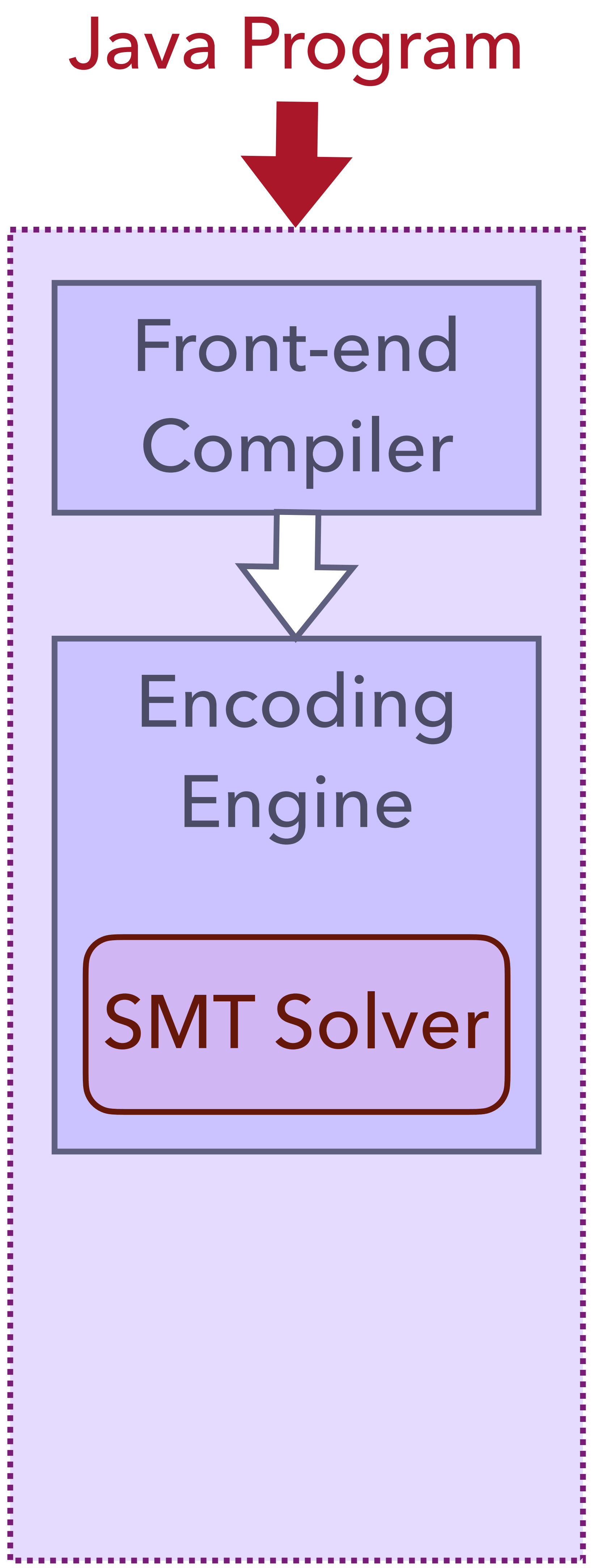
# CLOTHO: BUG DETECTION MECHANISM

- Static analysis engine for java programs
- Compiles programs down to an abstract representation
- FOL encoding engine, backed by Z3 SMT solver



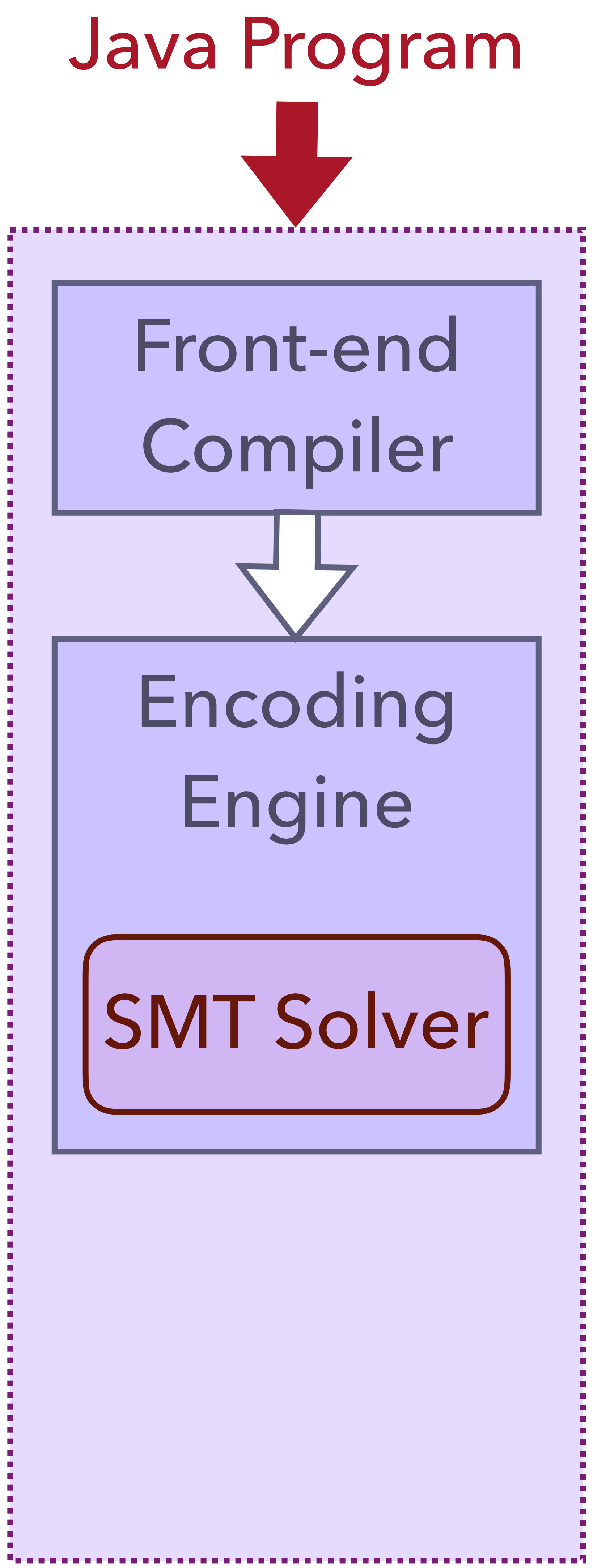
# CLOTHO: BUG DETECTION MECHANISM

- Static analysis engine for java programs
- Compiles programs down to an abstract representation
- FOL encoding engine, backed by Z3 SMT solver



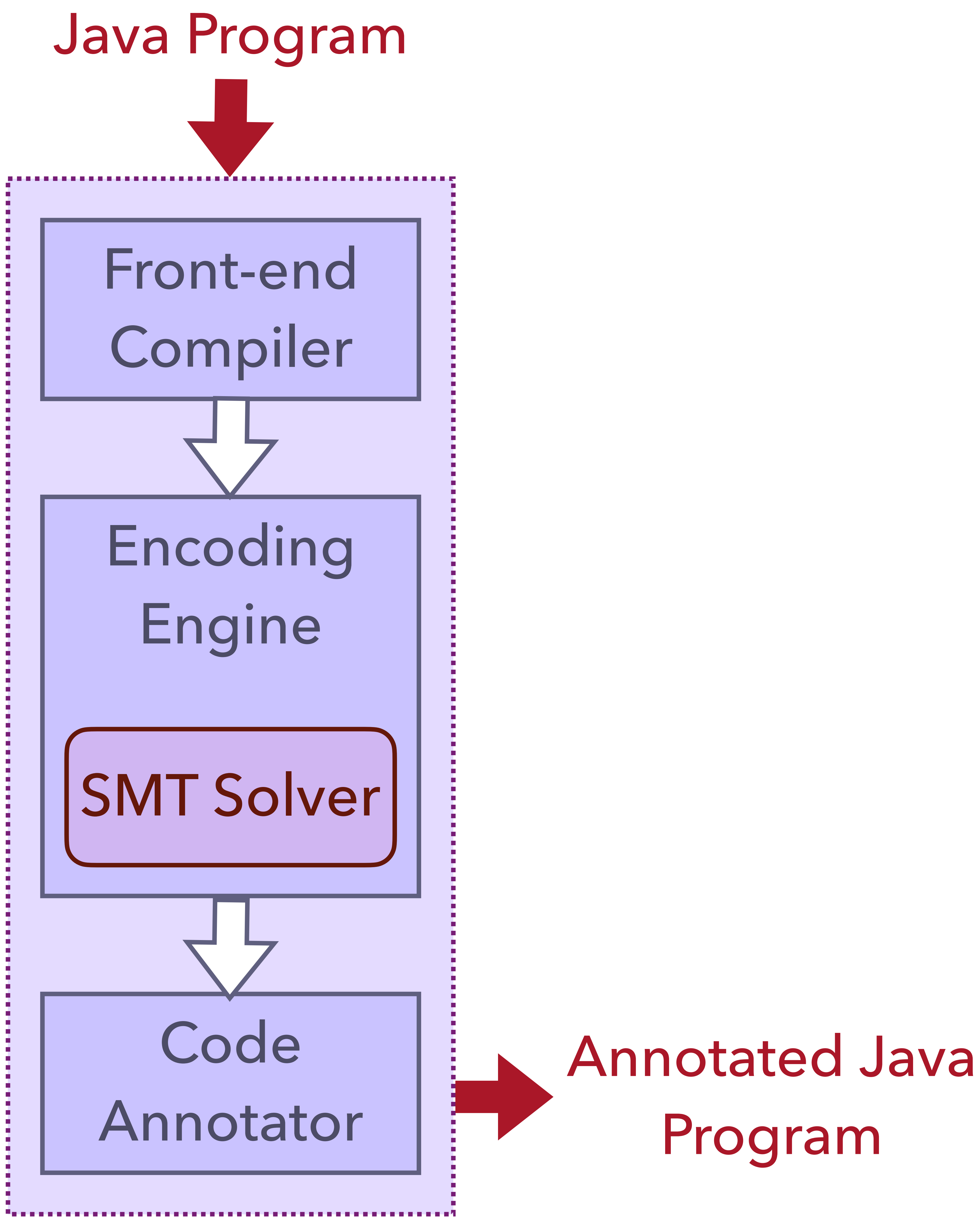
# CLOTHO: BUG DETECTION MECHANISM

- Static analysis engine for java programs
- Compiles programs down to an abstract representation
- FOL encoding engine, backed by Z3 SMT solver
- Efficient search algorithm



# CLOTHO: BUG DETECTION MECHANISM

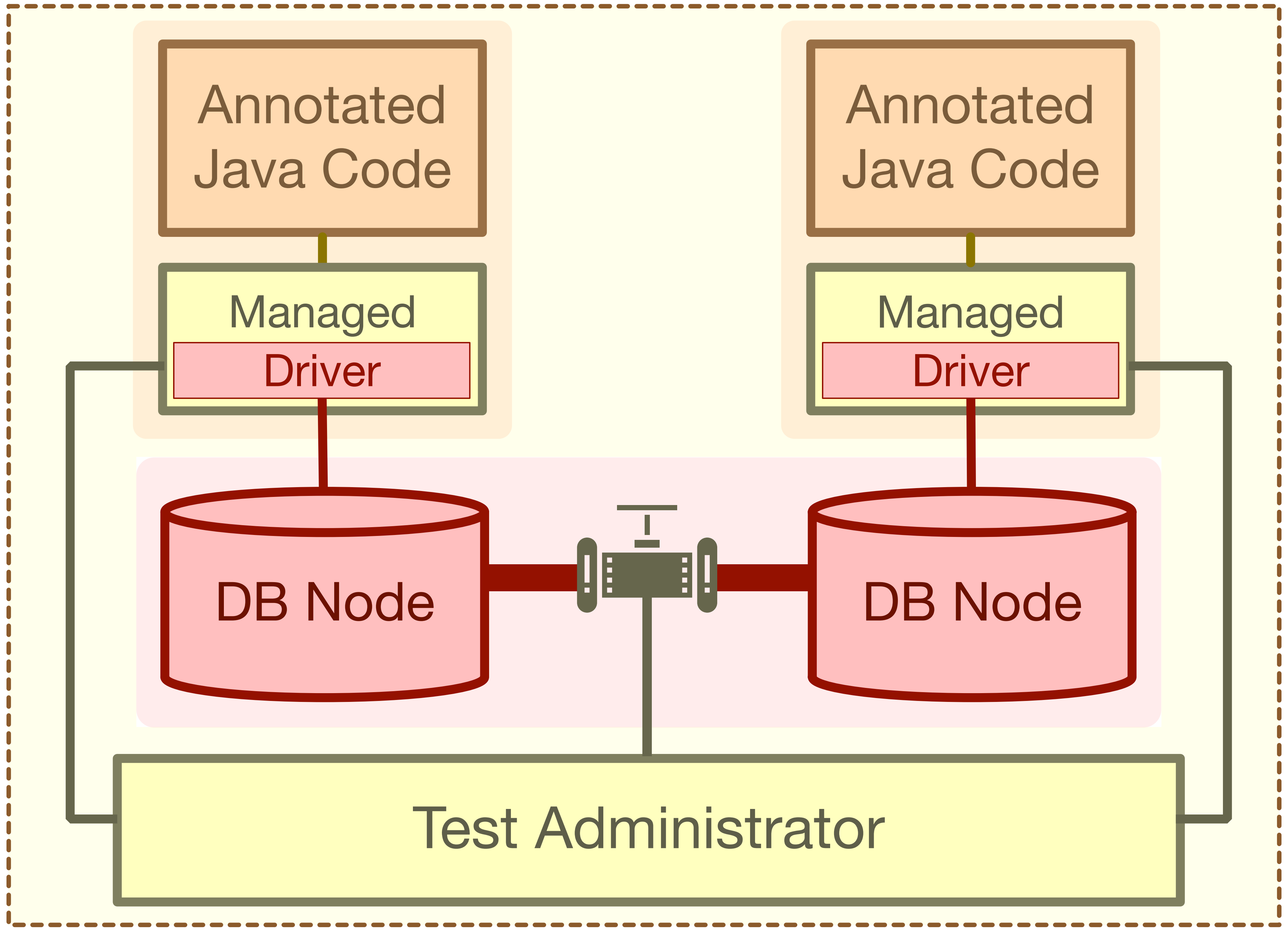
- Static analysis engine for java programs
- Compiles programs down to an abstract representation
- FOL encoding engine, backed by Z3 SMT solver
- Efficient search algorithm
- Returns annotated code containing concrete anomalies



# CLOTHO: REPLAYING MECHANISM

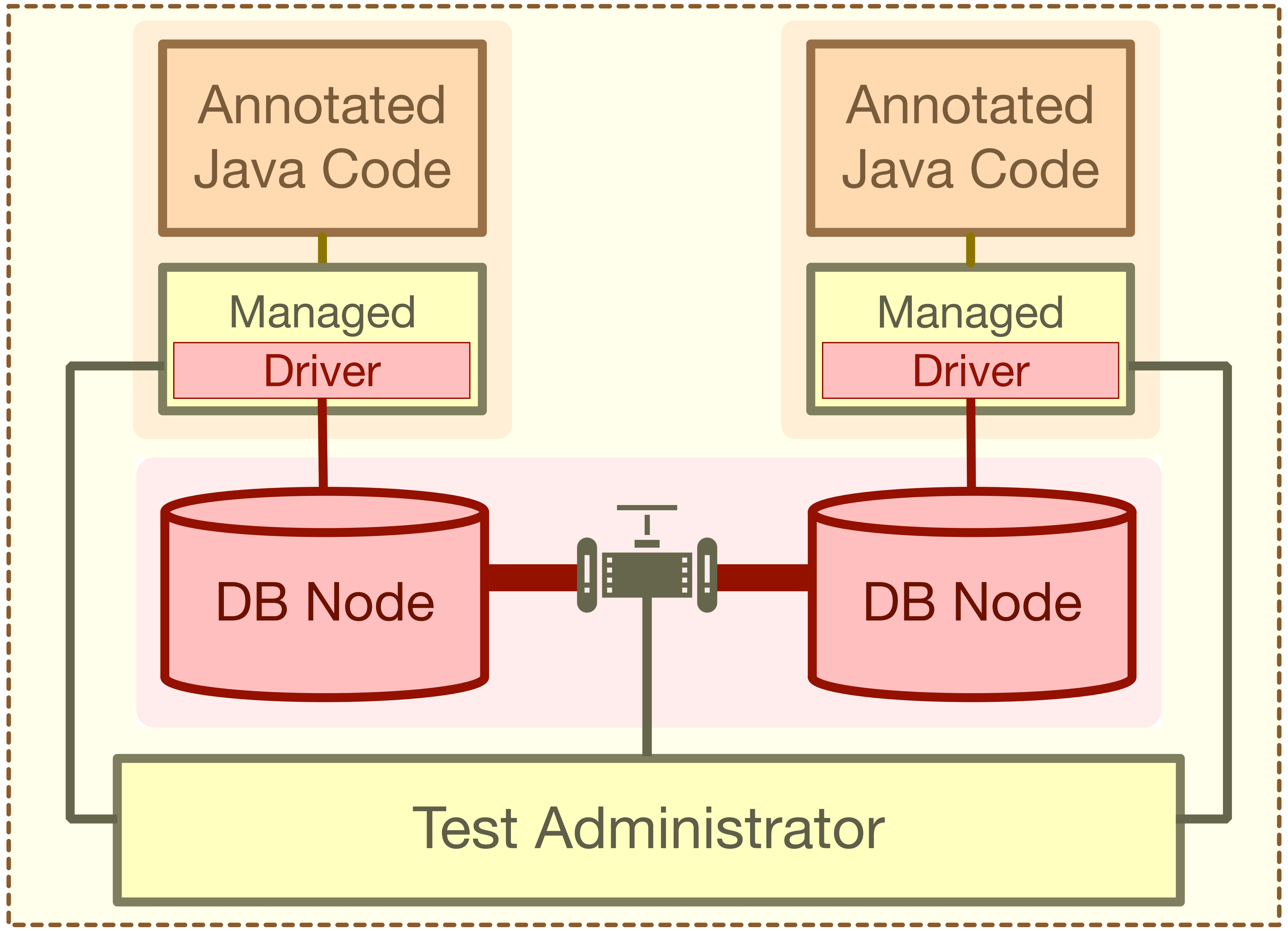
# CLOTHO: REPLAYING MECHANISM

- Directed test framework



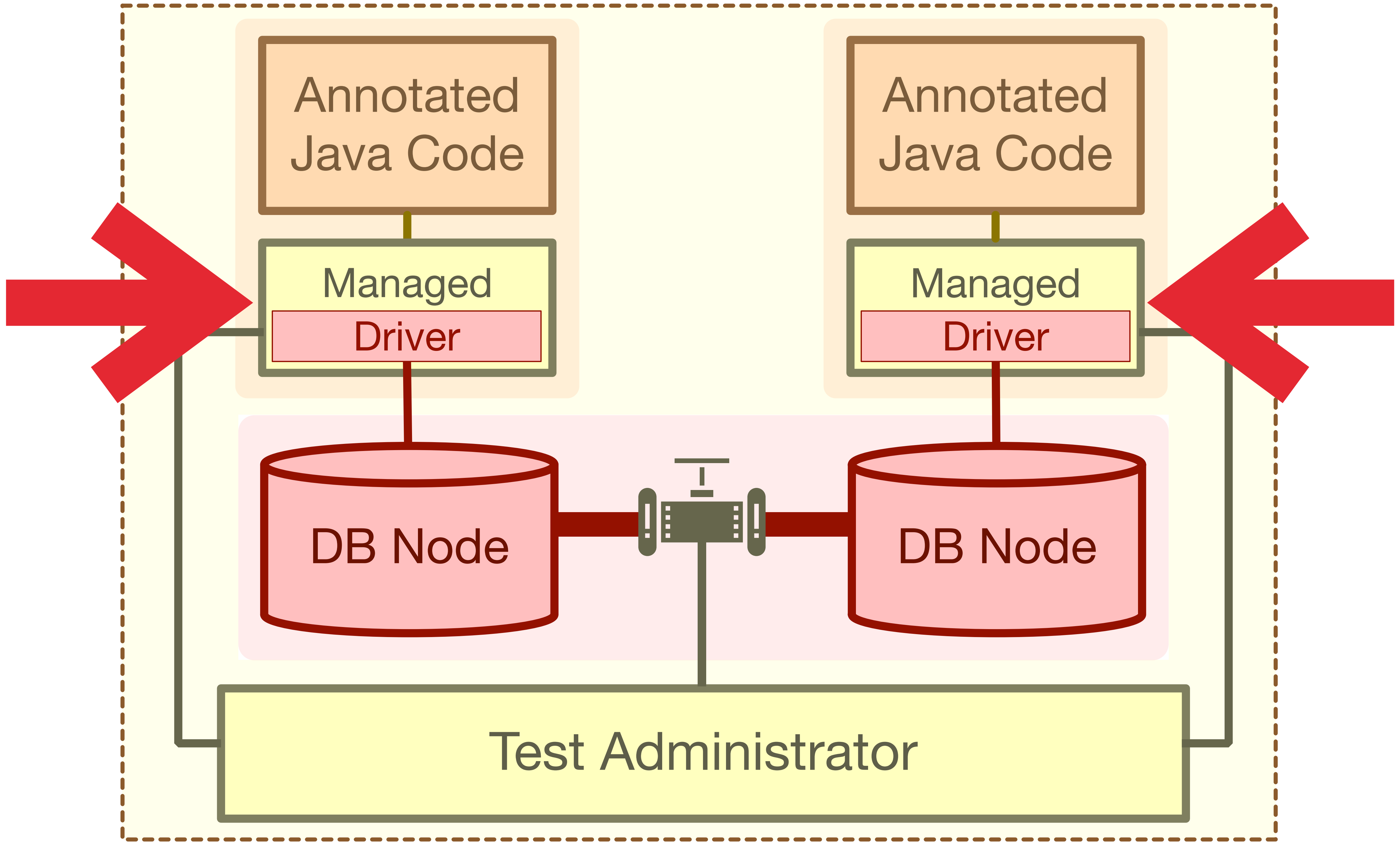
# CLOTHO: REPLAYING MECHANISM

- Directed test framework
  - automated step-by-step replaying of annotated buggy programs



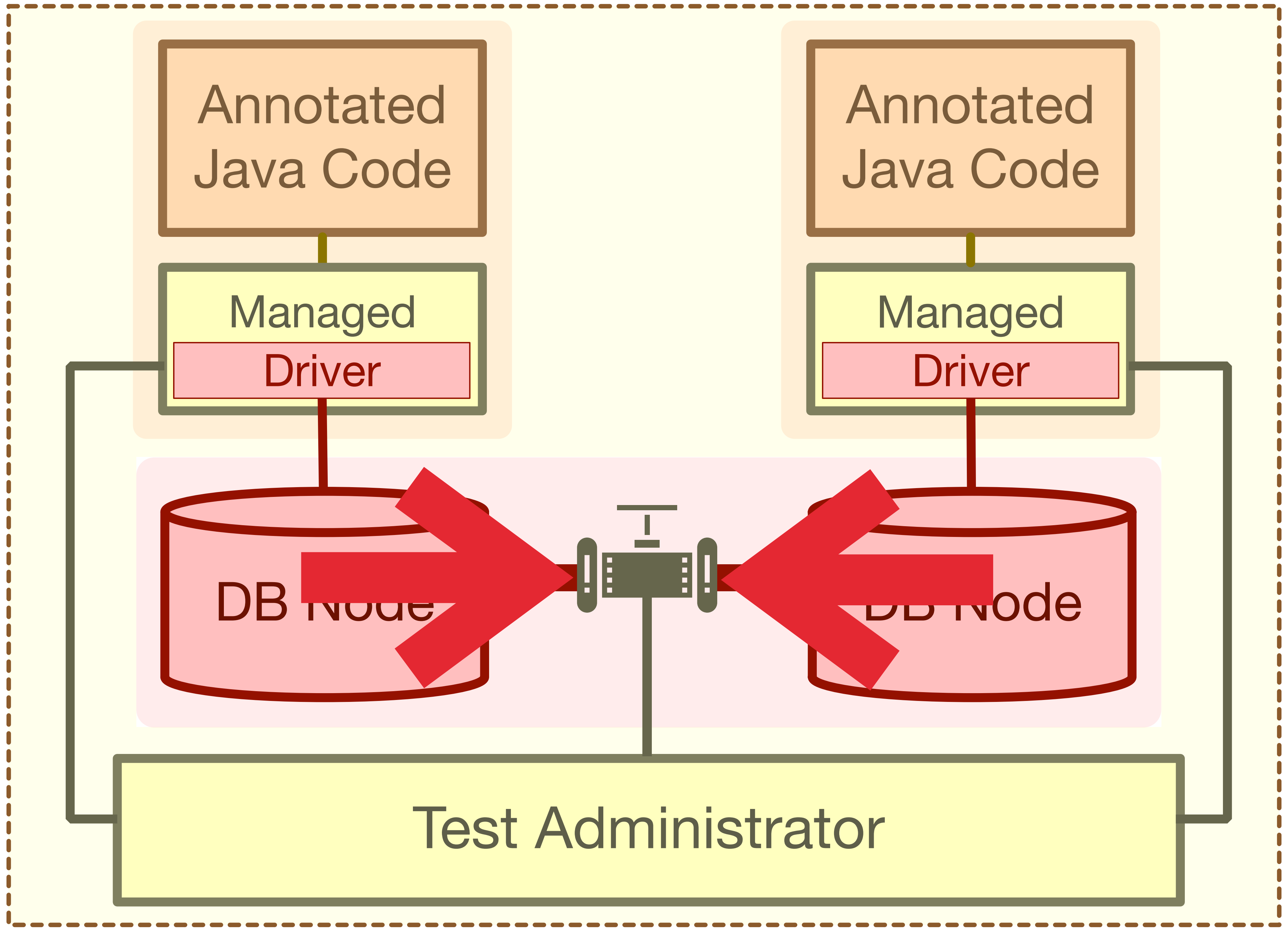
# CLOTHO: REPLAYING MECHANISM

- Directed test framework
  - automated step-by-step replaying of annotated buggy programs
  - synchronized drivers



# CLOTHO: REPLAYING MECHANISM

- Directed test framework
  - automated step-by-step replaying of annotated buggy programs
  - synchronized drivers
  - managed connection throttler in a cluster of database nodes



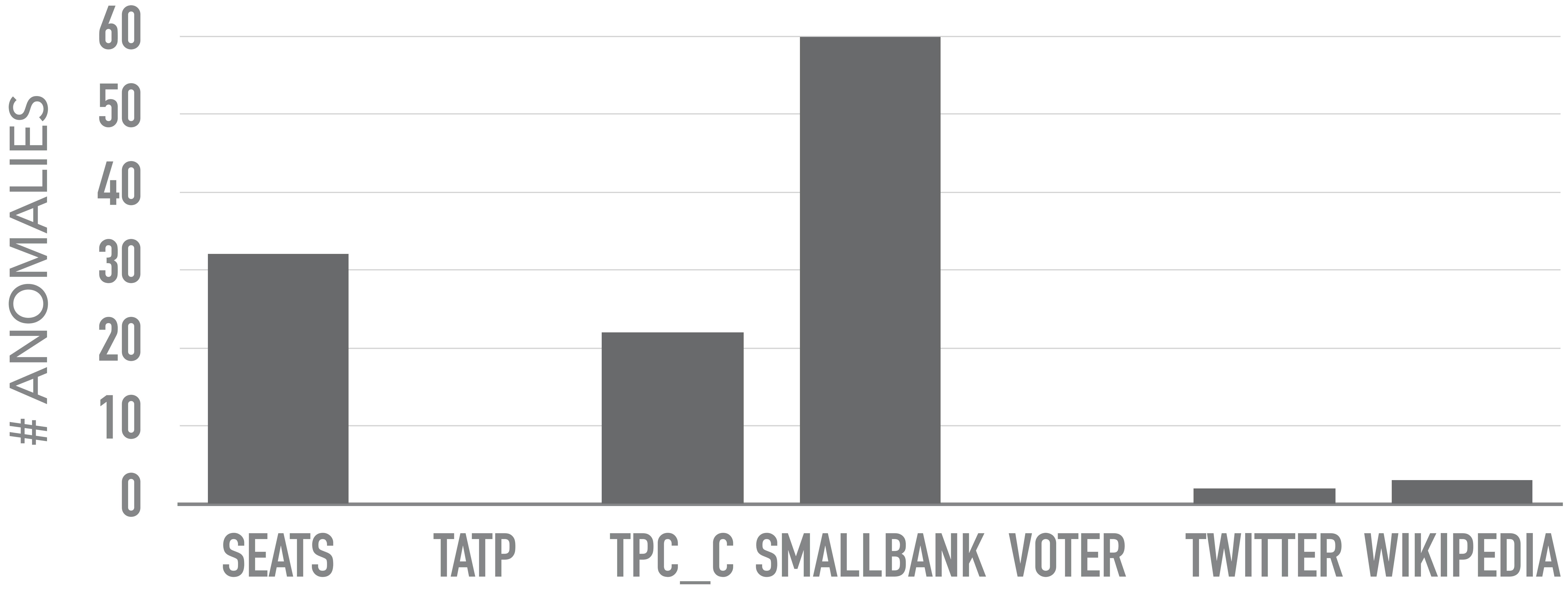
# EMPIRICAL RESULTS: APPLICABILITY

# EMPIRICAL RESULTS: APPLICABILITY

- **7 benchmarks** of various complexity and different properties were analyzed

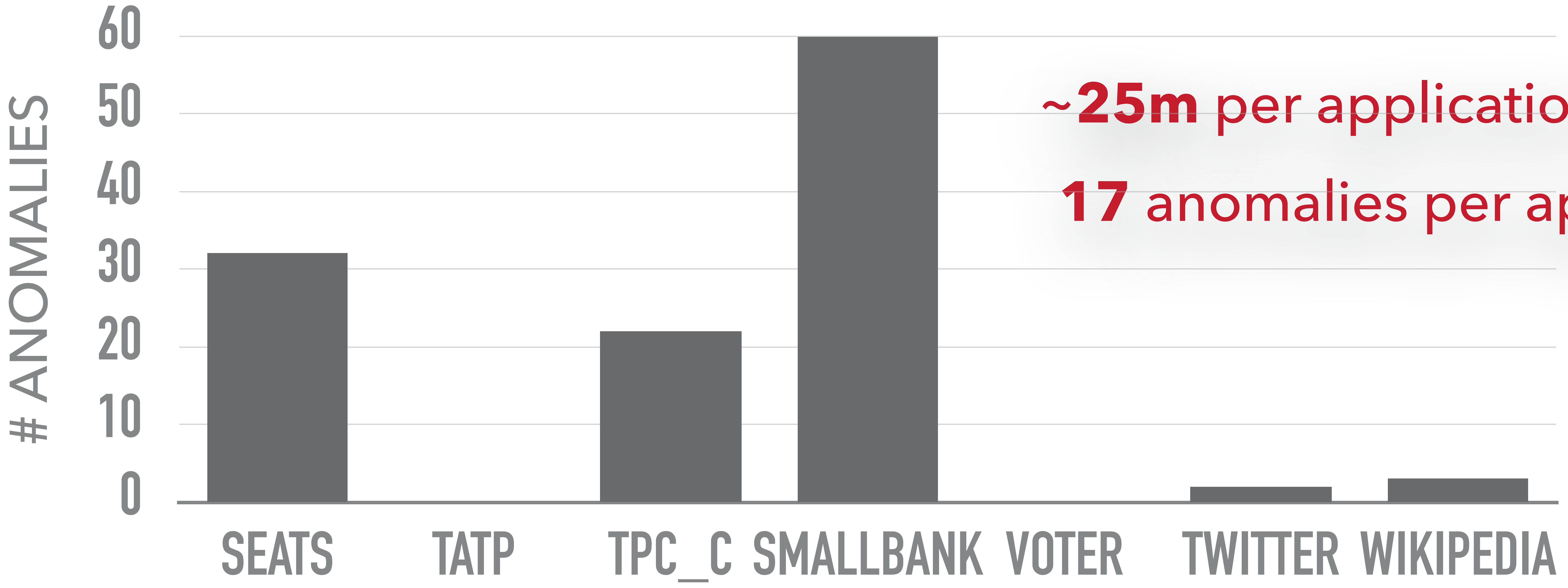
# EMPIRICAL RESULTS: APPLICABILITY

- **7 benchmarks** of various complexity and different properties were analyzed
- Serializability anomalies were found and successfully replayed in 5 application



# EMPIRICAL RESULTS: APPLICABILITY

- **7 benchmarks** of various complexity and different properties were analyzed
- Serializability anomalies were found and successfully replayed in 5 application



~**25m** per application (avg)

**17** anomalies per application (avg)

# EMPIRICAL RESULTS: COMPARISON TO BLACKBOX TESTING

| Invariant | Blackbox |
|---|---|
| CR1 | Y |
| CR2 | Y |
| CR3 | Y |
| CR4 | Y |
| CR5A | N |
| CR5B | N |
| CR6 | Y |
| CR7A | N |
| CR7B | N |
| CR8 | Y |
| CR9 | Y |
| CR10 | Y |
| CR11 | Y |
| CR12 | Y |
| NCR1 | Y |
| NCR2 | Y |
| NCR3 | N |
| NCR4 | N |
| NCR5 | Y |
| NCR6 | Y |
| NCR7 | N |

# EMPIRICAL RESULTS: COMPARISON TO BLACKBOX TESTING

- Case study: TPC-C

| Invariant | Blackbox |
|---|---|
| CR1 | Y |
| CR2 | Y |
| CR3 | Y |
| CR4 | Y |
| CR5A | N |
| CR5B | N |
| CR6 | Y |
| CR7A | N |
| CR7B | N |
| CR8 | Y |
| CR9 | Y |
| CR10 | Y |
| CR11 | Y |
| CR12 | Y |
| NCR1 | Y |
| NCR2 | Y |
| NCR3 | N |
| NCR4 | N |
| NCR5 | Y |
| NCR6 | Y |
| NCR7 | N |

# EMPIRICAL RESULTS: COMPARISON TO BLACKBOX TESTING

- Case study: TPC-C
- Anomalies were studied and mapped to invariant violations

| Invariant | Blackbox |
|---|---|
| CR1 | Y |
| CR2 | Y |
| CR3 | Y |
| CR4 | Y |
| CR5A | N |
| CR5B | N |
| CR6 | Y |
| CR7A | N |
| CR7B | N |
| CR8 | Y |
| CR9 | Y |
| CR10 | Y |
| CR11 | Y |
| CR12 | Y |
| NCR1 | Y |
| NCR2 | Y |
| NCR3 | N |
| NCR4 | N |
| NCR5 | Y |
| NCR6 | Y |
| NCR7 | N |

# EMPIRICAL RESULTS: COMPARISON TO BLACKBOX TESTING

- Case study: TPC-C
- Anomalies were studied and mapped to invariant violations
- **All invariants were broken** as a result of at least one serializability anomaly

| Invariant | Blackbox | CLOTHO |
|---|---|---|
| CR1 | Y | Y |
| CR2 | Y | Y |
| CR3 | Y | Y |
| CR4 | Y | Y |
| CR5A | N | Y |
| CR5B | N | Y |
| CR6 | Y | Y |
| CR7A | N | Y |
| CR7B | N | Y |
| CR8 | Y | Y |
| CR9 | Y | Y |
| CR10 | Y | Y |
| CR11 | Y | Y |
| CR12 | Y | Y |
| NCR1 | Y | Y |
| NCR2 | Y | Y |
| NCR3 | N | Y |
| NCR4 | N | Y |
| NCR5 | Y | Y |
| NCR6 | Y | Y |
| NCR7 | N | Y |

# EMPIRICAL RESULTS: COMPARISON TO BLACKBOX TESTING

- Case study: TPC-C
- Anomalies were studied and mapped to invariant violations
- **All invariants were broken** as a result of at least one serializability anomaly
- Only 3 serializability anomalies did not result in any invariant violation

| Invariant | Blackbox | CLOTHO |
|---|---|---|
| CR1 | Y | Y |
| CR2 | Y | Y |
| CR3 | Y | Y |
| CR4 | Y | Y |
| CR5A | N | Y |
| CR5B | N | Y |
| CR6 | Y | Y |
| CR7A | N | Y |
| CR7B | N | Y |
| CR8 | Y | Y |
| CR9 | Y | Y |
| CR10 | Y | Y |
| CR11 | Y | Y |
| CR12 | Y | Y |
| NCR1 | Y | Y |
| NCR2 | Y | Y |
| NCR3 | N | Y |
| NCR4 | N | Y |
| NCR5 | Y | Y |
| NCR6 | Y | Y |
| NCR7 | N | Y |

# SUMMARY

## SUMMARY

- CLOTHO: an end-to-end directed testing framework for weakly consistent database programs

## SUMMARY

- CLOTHO: an end-to-end directed testing framework for weakly consistent database programs
- The problem of finding serializability anomalies is reduced to finding satisfying assignments to a formula

## SUMMARY

- CLOTHO: an end-to-end directed testing framework for weakly consistent database programs
- The problem of finding serializability anomalies is reduced to finding satisfying assignments to a formula
- Applicable on many benchmark applications

# SUMMARY

- CLOTHO: an end-to-end directed testing framework for weakly consistent database programs
- The problem of finding serializability anomalies is reduced to finding satisfying assignments to a formula
- Applicable on many benchmark applications
- Outperforms state of the art blackbox testing techniques

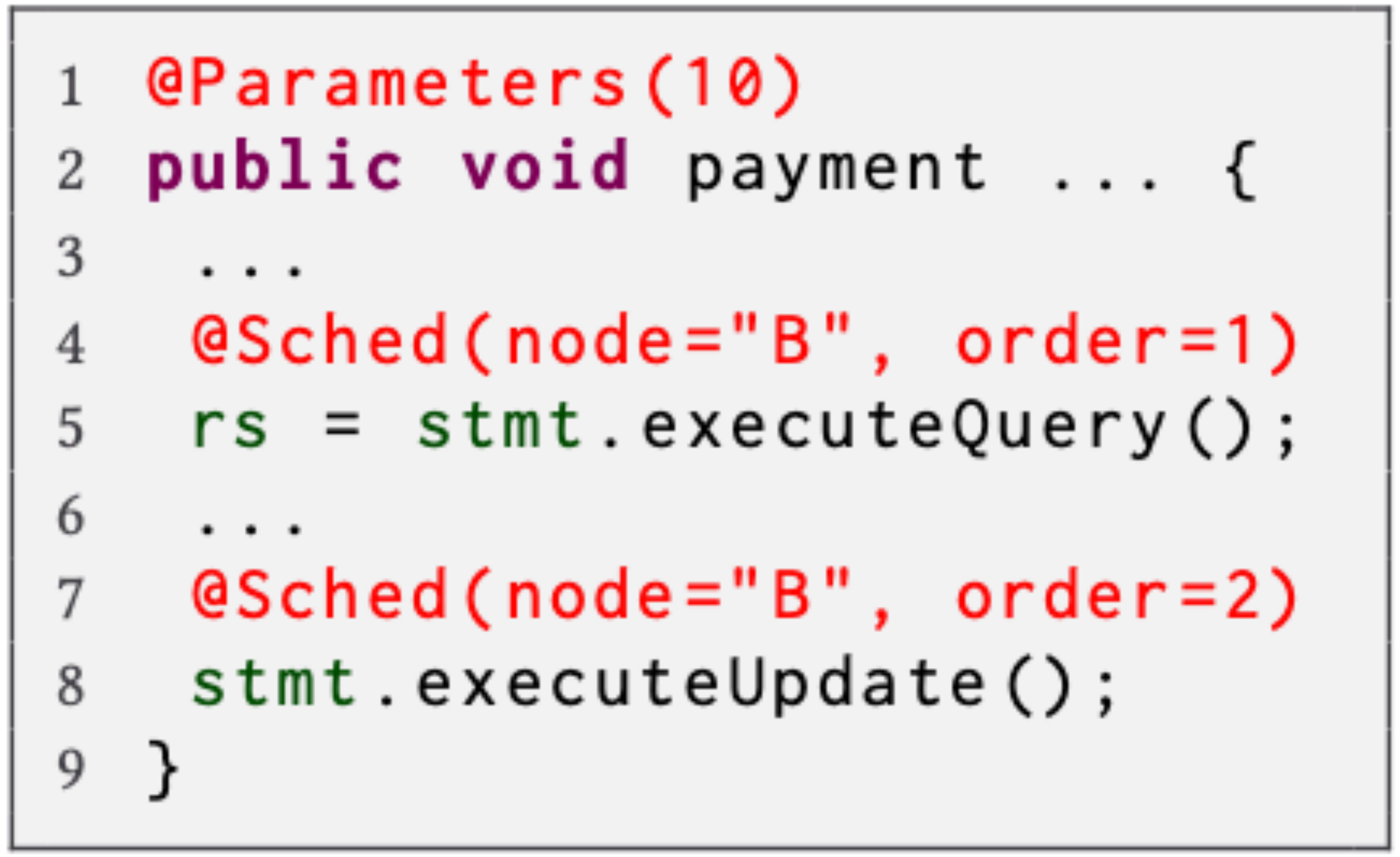
THANK YOU!

# QUESTIONS?

TOOL AVAILABLE

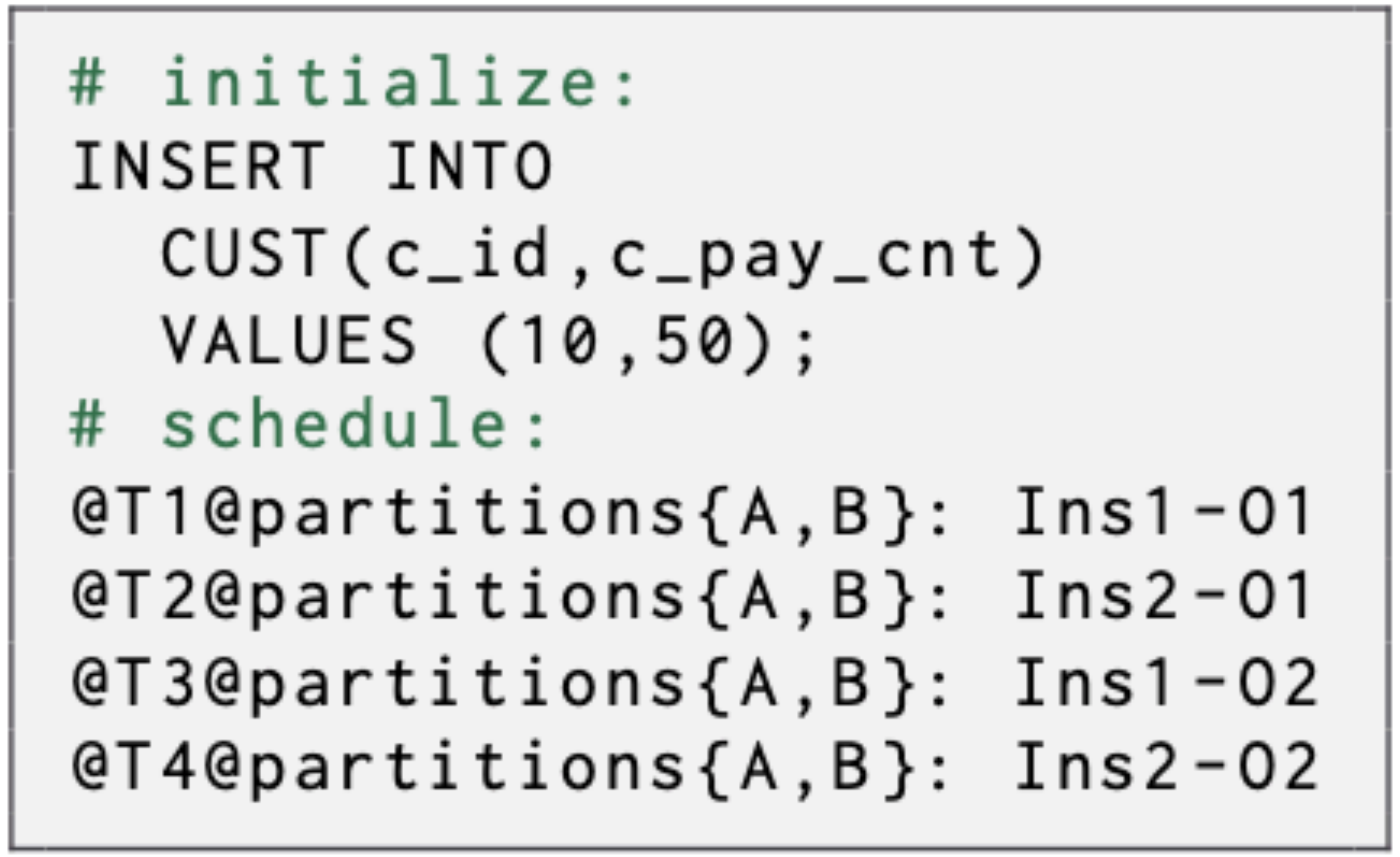
# ANNOTATED CODE EXAMPLE

- Includes transaction instances, arguments
- Accompanied by a test configuration file specifying execution order and networking details



```java
@Parameters(10)
public void payment ... {
 ...
 @Sched(node="B", order=1)
 rs = stmt.executeQuery();
 ...
 @Sched(node="B", order=2)
 stmt.executeUpdate();
}
```

A1_Ins2.java

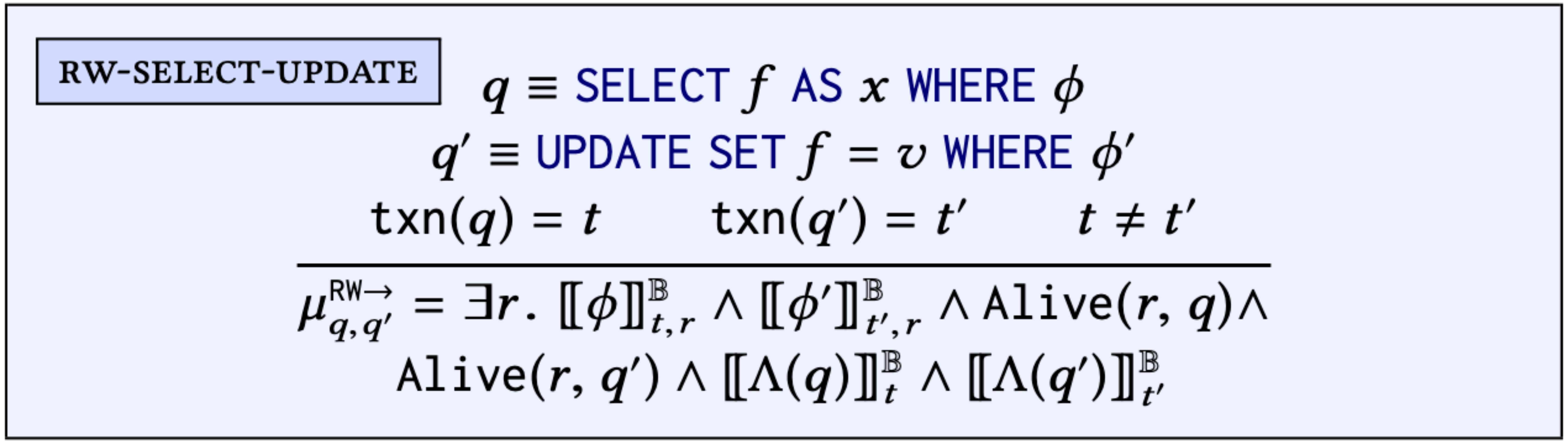


```
# initialize:
INSERT INTO
  CUST(c_id,c_pay_cnt)
  VALUES (10,50);
# schedule:
@T1@partitions{A,B}: Ins1-01
@T2@partitions{A,B}: Ins2-01
@T3@partitions{A,B}: Ins1-02
@T4@partitions{A,B}: Ins2-02
```

A1.conf

# NECESSARY RULE EXAMPLE

- Rules specify the necessary conditions for establishing a dependency relation between two database operation instances

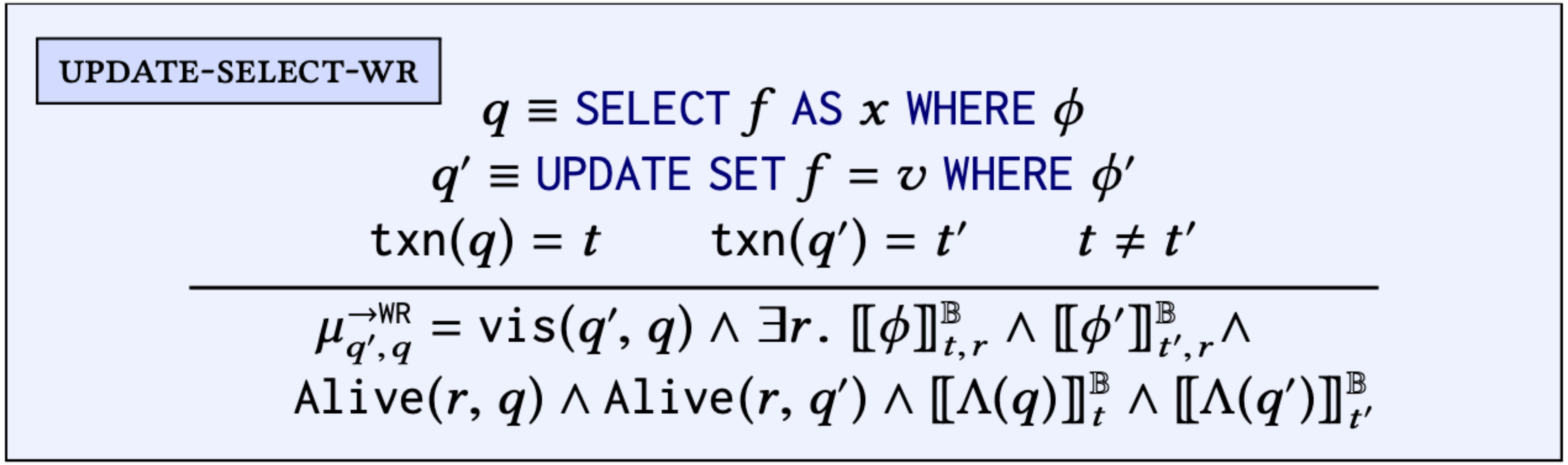


RW-SELECT-UPDATE

$$\frac{\begin{array}{c} q \equiv \mathsf{SELECT}\ f\ \mathsf{AS}\ x\ \mathsf{WHERE}\ \phi \\ q' \equiv \mathsf{UPDATE\ SET}\ f = v\ \mathsf{WHERE}\ \phi' \\ \mathsf{txn}(q) = t \qquad \mathsf{txn}(q') = t' \qquad t \neq t' \end{array}}{\begin{array}{c} \mu^{\mathsf{RW}\rightarrow}_{q,q'} = \exists r.\ [\![\phi]\!]^{\mathbb{B}}_{t,r} \wedge [\![\phi']\!]^{\mathbb{B}}_{t',r} \wedge \mathtt{Alive}(r, q) \wedge \\ \mathtt{Alive}(r, q') \wedge [\![\Lambda(q)]\!]^{\mathbb{B}}_{t} \wedge [\![\Lambda(q')]\!]^{\mathbb{B}}_{t'} \end{array}}$$

# SUFFICIENT RULE EXAMPLE

- Rules specify the sufficient conditions for establishing a dependency relation between two database operation instances

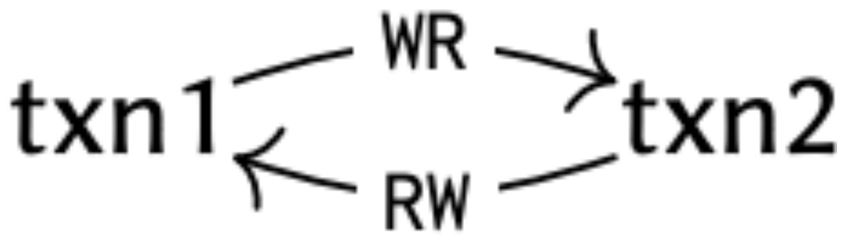


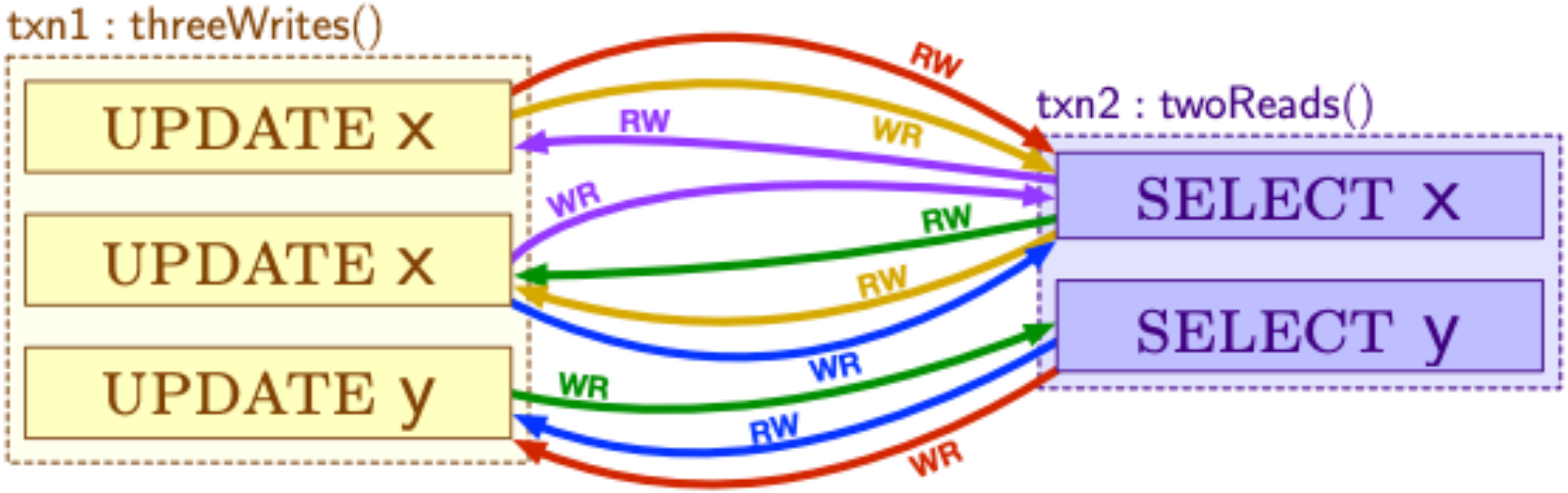
UPDATE-SELECT-WR

$$\frac{\begin{array}{c} q \equiv \mathsf{SELECT}\ f\ \mathsf{AS}\ x\ \mathsf{WHERE}\ \phi \\ q' \equiv \mathsf{UPDATE\ SET}\ f = \upsilon\ \mathsf{WHERE}\ \phi' \\ \mathtt{txn}(q) = t \qquad \mathtt{txn}(q') = t' \qquad t \neq t' \end{array}}{\begin{array}{c} \mu^{\to\mathsf{WR}}_{q',q} = \mathtt{vis}(q', q) \wedge \exists r.\ [\![\phi]\!]^{\mathbb{B}}_{t,r} \wedge [\![\phi']\!]^{\mathbb{B}}_{t',r} \wedge \\ \mathtt{Alive}(r, q) \wedge \mathtt{Alive}(r, q') \wedge [\![\Lambda(q)]\!]^{\mathbb{B}}_{t} \wedge [\![\Lambda(q')]\!]^{\mathbb{B}}_{t'} \end{array}}$$

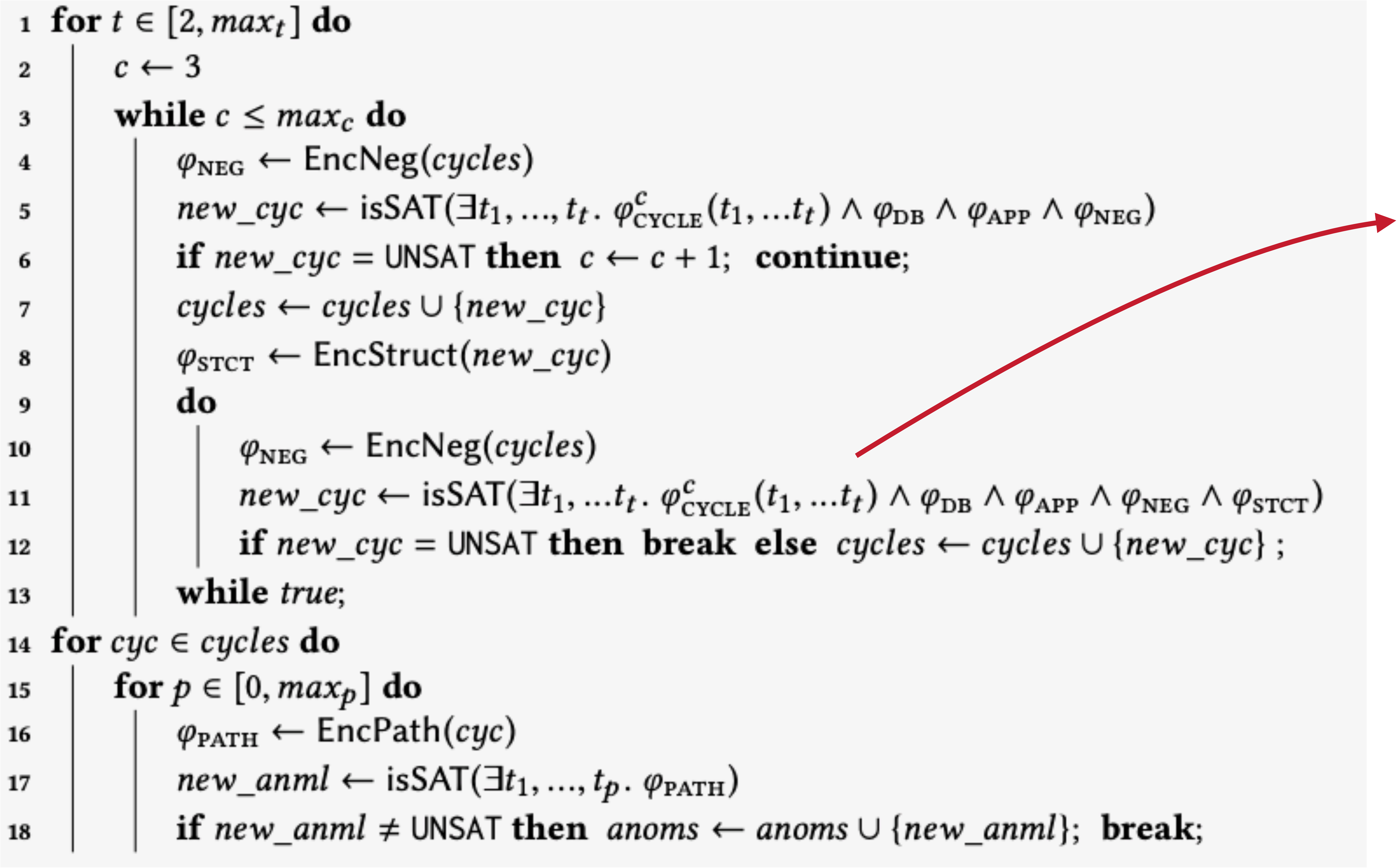
# STRUCTURALLY SIMILAR ANOMALIES

- All share the same transaction instances and the same edges between them:





# SEARCH ALGORITHM

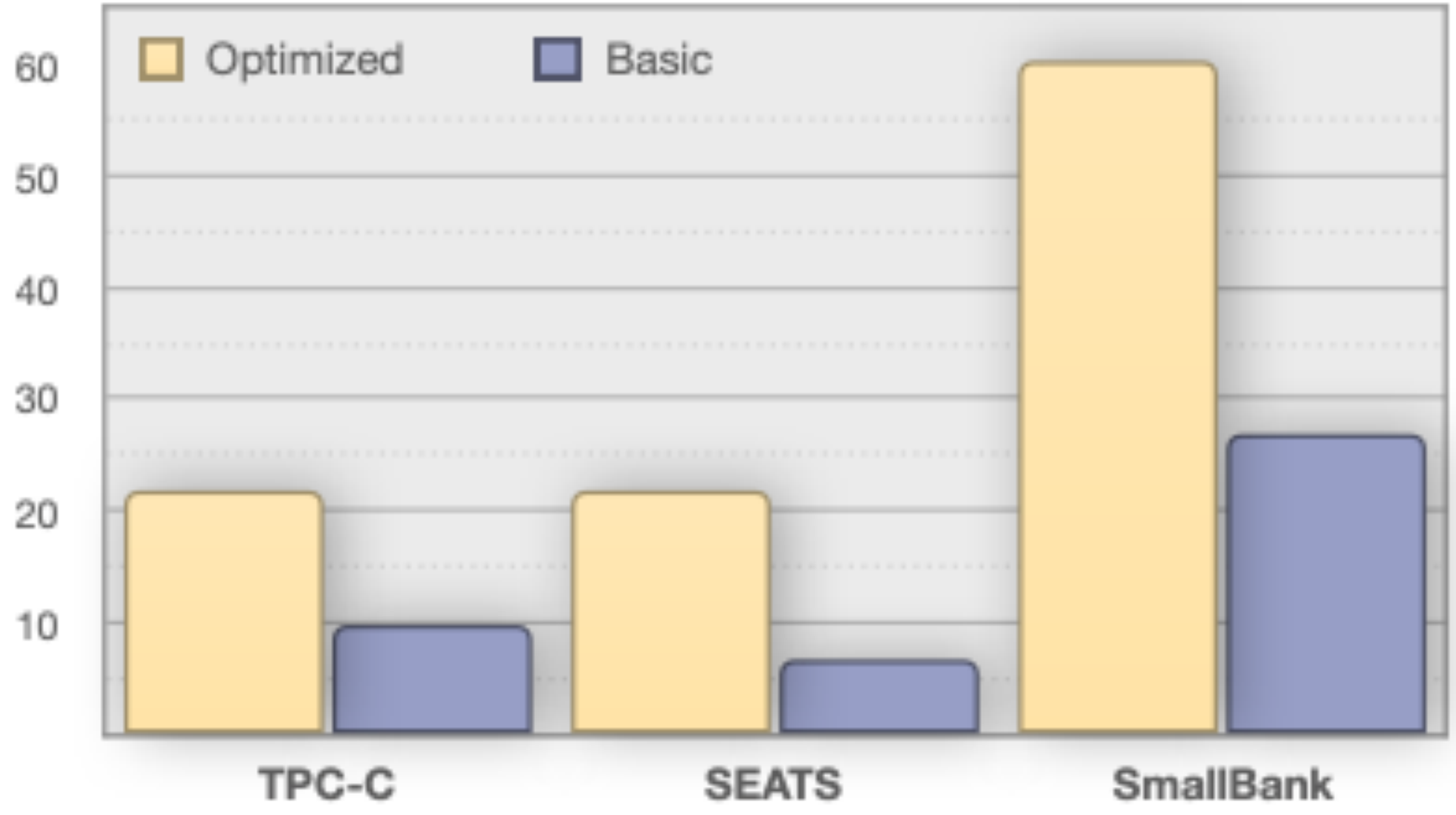


```
for $t \in [2, max_t]$ do
    $c \leftarrow 3$
    while $c \leq max_c$ do
        $\varphi_{\text{NEG}} \leftarrow \text{EncNeg}(cycles)$
        $new\_cyc \leftarrow \text{isSAT}(\exists t_1, \ldots, t_t .\ \varphi^c_{\text{CYCLE}}(t_1, \ldots t_t) \wedge \varphi_{\text{DB}} \wedge \varphi_{\text{APP}} \wedge \varphi_{\text{NEG}})$
        if $new\_cyc = \text{UNSAT}$ then $c \leftarrow c + 1$; continue;
        $cycles \leftarrow cycles \cup \{new\_cyc\}$
        $\varphi_{\text{STCT}} \leftarrow \text{EncStruct}(new\_cyc)$
        do
            $\varphi_{\text{NEG}} \leftarrow \text{EncNeg}(cycles)$
            $new\_cyc \leftarrow \text{isSAT}(\exists t_1, \ldots t_t .\ \varphi^c_{\text{CYCLE}}(t_1, \ldots t_t) \wedge \varphi_{\text{DB}} \wedge \varphi_{\text{APP}} \wedge \varphi_{\text{NEG}} \wedge \varphi_{\text{STCT}})$
            if $new\_cyc = \text{UNSAT}$ then break else $cycles \leftarrow cycles \cup \{new\_cyc\}$ ;
        while $true$;
for $cyc \in cycles$ do
    for $p \in [0, max_p]$ do
        $\varphi_{\text{PATH}} \leftarrow \text{EncPath}(cyc)$
        $new\_anml \leftarrow \text{isSAT}(\exists t_1, \ldots, t_p .\ \varphi_{\text{PATH}})$
        if $new\_anml \neq \text{UNSAT}$ then $anoms \leftarrow anoms \cup \{new\_anml\}$; break;
```

optimization: inner loop for finding structurally similar anomalies

## EFFECT OF OPTIMIZATIONS IN SEARCH ALGORITHM

Number of anomalies found within the same given time period



# RELATED WORKS

- [Kaki et al. 2018], [Nagar et al. 2018]
  - Do not incorporate their techniques into a full test-and-reply environment
- [Brutschy et al. 2018]
  - Does not suit query-based models where dependences between two operations cannot be decided locally, but are reliant on other operations
- [Warszawski and Bailis 2017]
  - Does not consider how to help determining if applications executing on storage systems that expose guarantees weaker than serializability are actually **correct**