



SQL Transactions

- Транзакции
- Уровни изоляции транзакций
- Locking
- Savepoints
- Spring Transactional
- Distributed DBs

.. T .. Systems ..

Что такое транзакция?

- Транзакция – ...

... T ... Systems ...

Что такое транзакция?

- Транзакция – группа логически объединённых последовательных операций по работе с данными, обрабатываемая или отменяемая целиком.
- Типичный пример транзакции – банковский перевод

... T ... Systems ...

- ACID – требования к транзакционной системе

- Atomicity
- Consistency
- Isolation
- Durability

... T ... Systems ...

- Транзакции
- Уровни изоляции транзакций
- Locking
- Savepoints
- Spring Transactional
- Distributed DBs

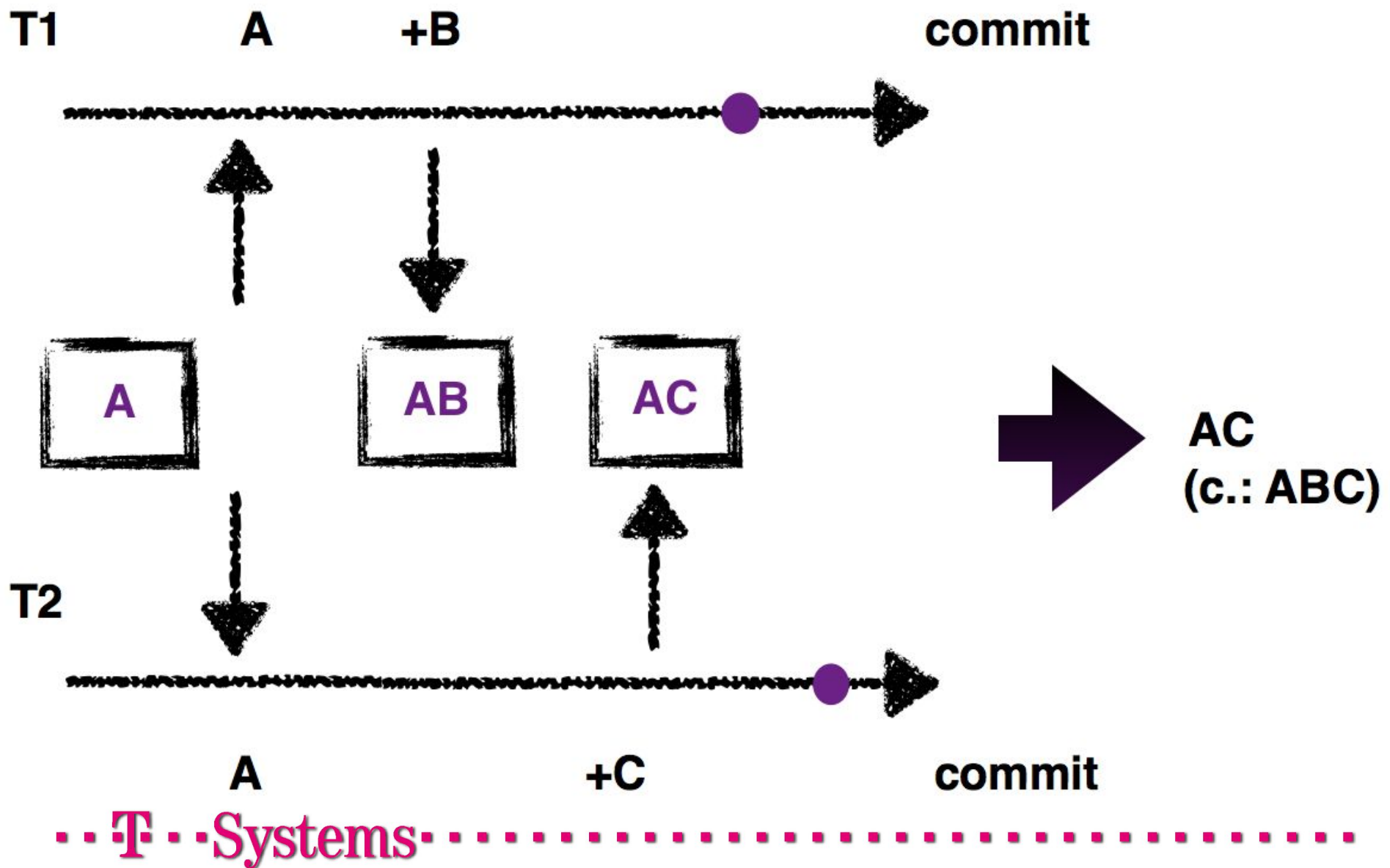
.. T .. Systems ..

Проблемы параллельного выполнения транзакций

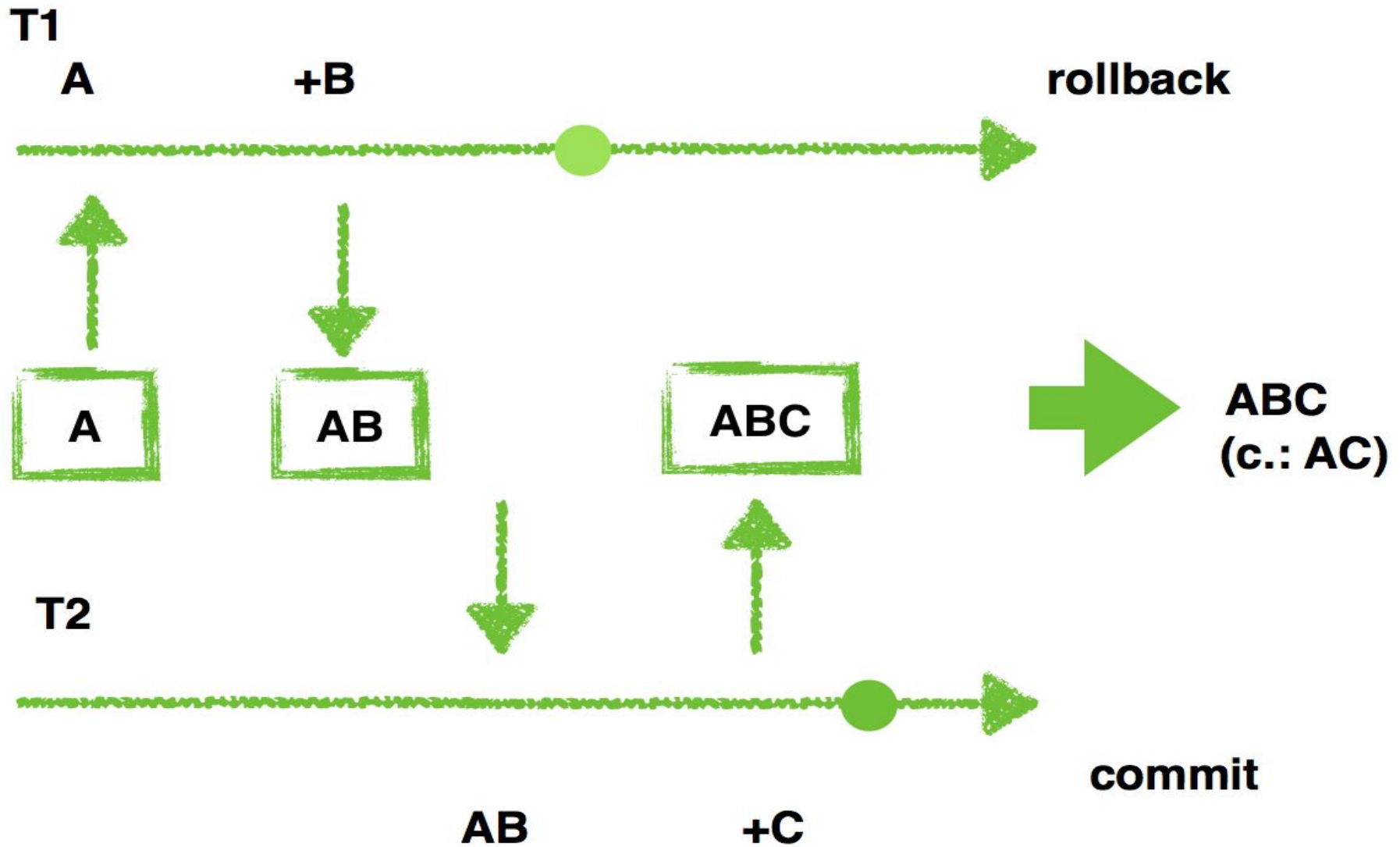
- Lost update
- Dirty read
- Non-repeatable read
- Phantom read

.. T .. Systems ..

Lost update

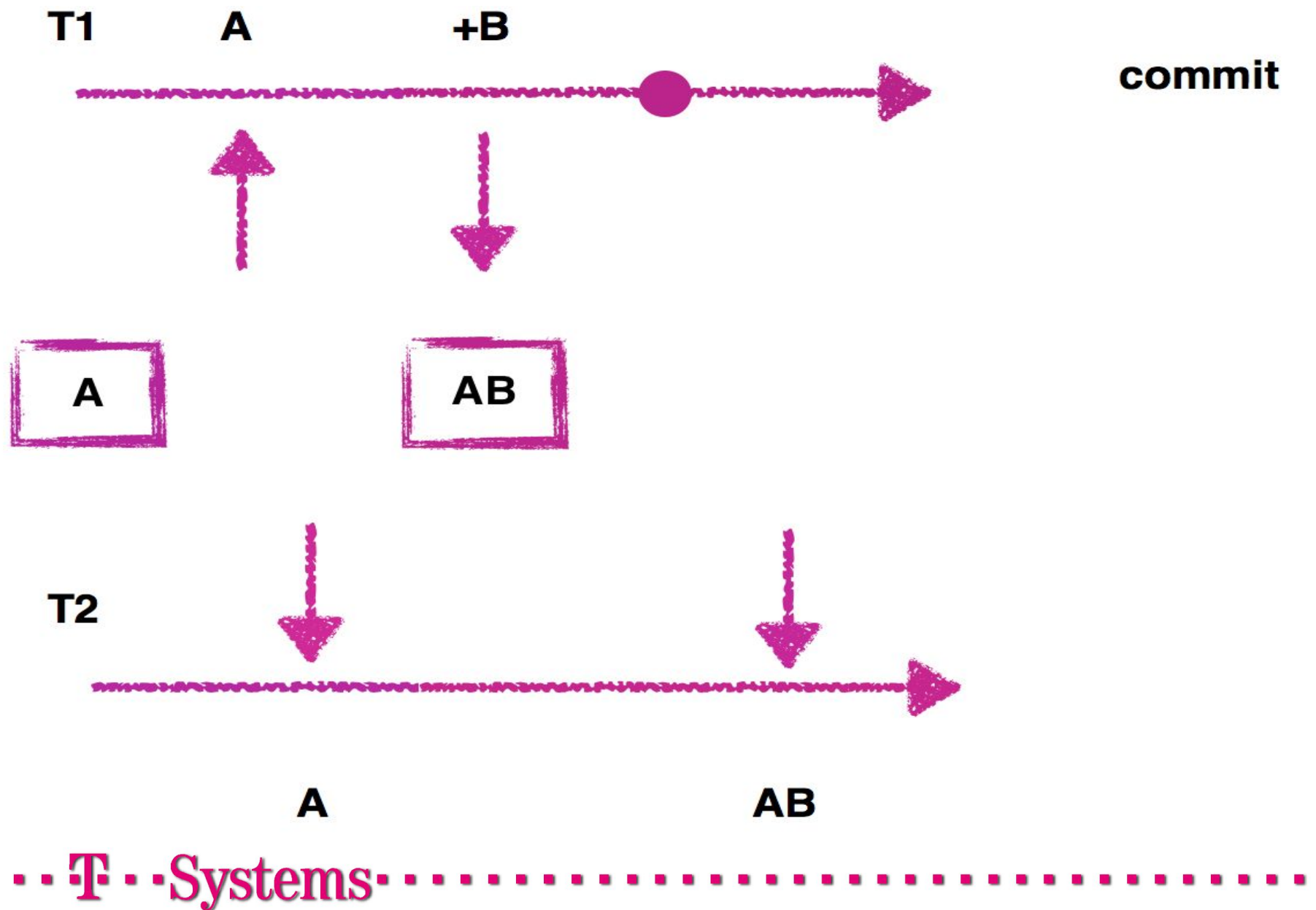


Dirty read

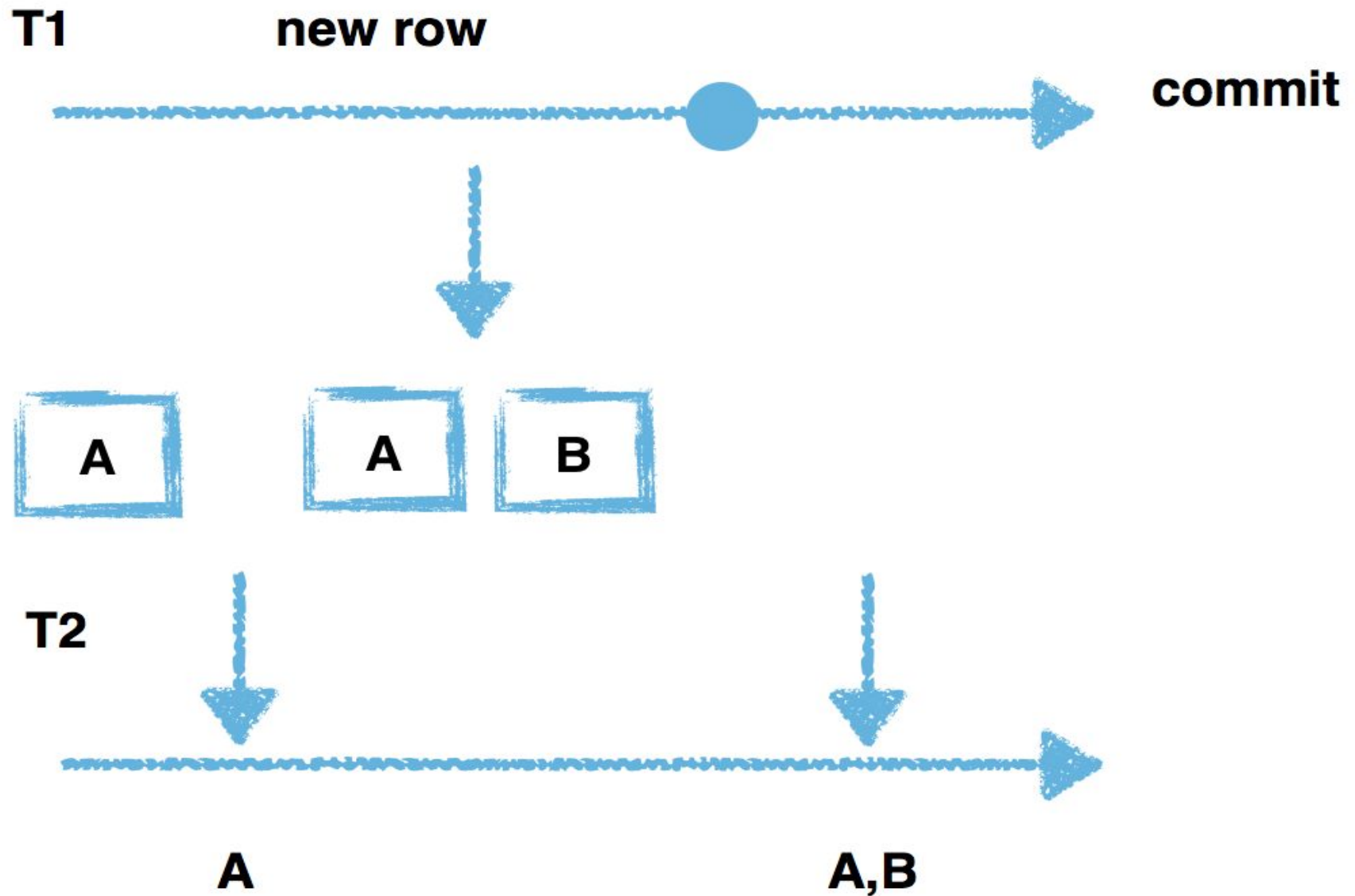


... **T** ... Systems ...

Non-repeatable read



Phantom read



... T ... Systems ...

Уровни изолированности транзакций

Уровень изоляции	Phantom read	Non-repeatable read	Dirty read	Lost update
Serializable	+	+	+	+
Repeatable read	-	+	+	+
Read committed	-	-	+	+
Read uncommitted	-	-	-	+

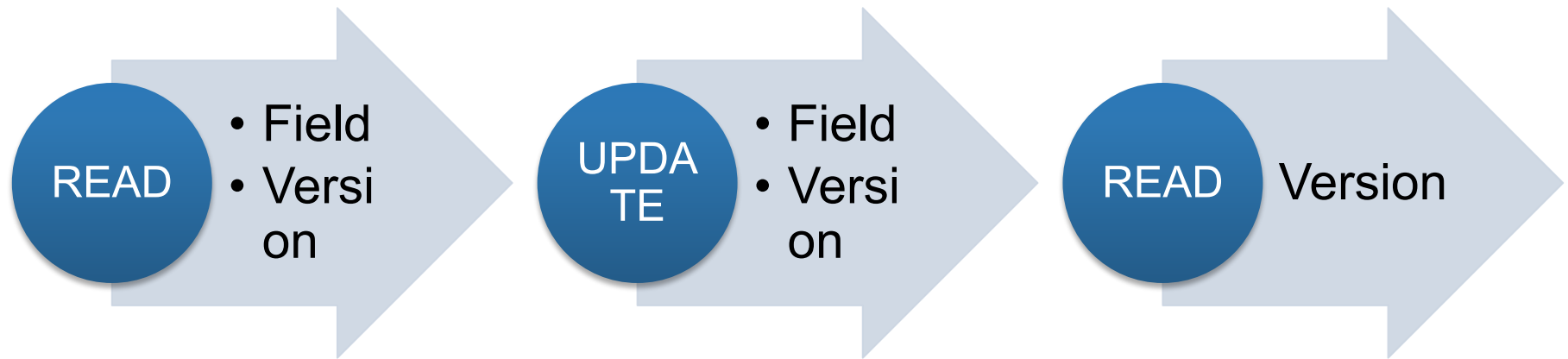
...T...Systems.....

- Транзакции
- Уровни изоляции транзакций
- Locking
- Savepoints
- Spring Transactional
- Distributed DBs

.. T .. Systems ..

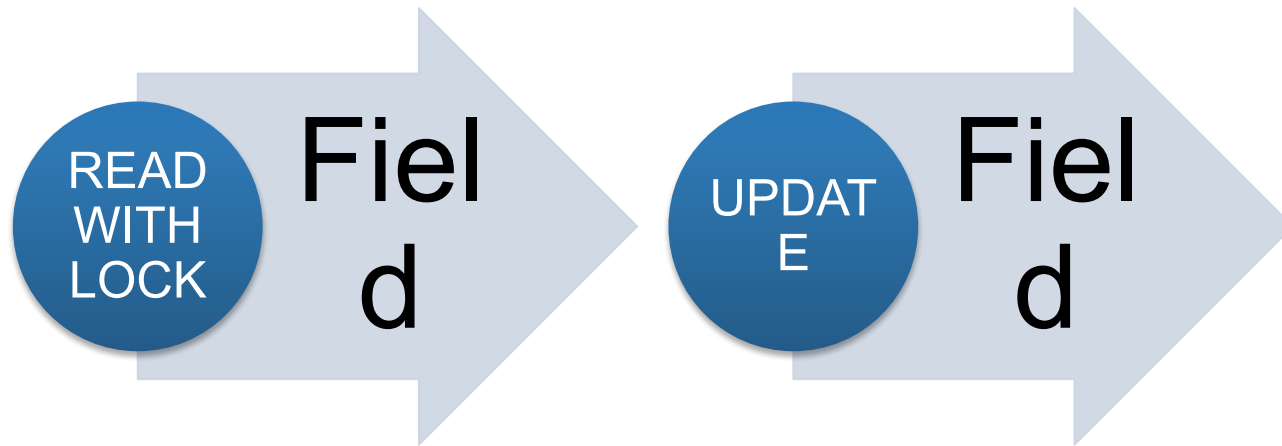
- Optimistic locking
- Pessimistic locking
 - Pessimistic write (exclusive lock)
 - Pessimistic read (shared lock)

Optimistic locking



... T ... Systems ...

Pessimistic locking



- Pessimistic write (SELECT ... FOR UPDATE)
- Pessimistic read (SELECT ... LOCK IN SHARE MODE)

... T ... Systems ...

- Транзакции
- Уровни изоляции транзакций
- Locking
- Savepoints
- Spring Transactional
- Distributed DBs

.. T .. Systems ..

Savepoints



...T...Systems.....

- Транзакции
- Уровни изоляции транзакций
- Locking
- Savepoints
- Spring Transactional
- Distributed DBs

.. T .. Systems ..

The @Transactional Annotation

```
@Service
@Transactional
public class FooService {
    . . . .
}

@Override
@Transactional(propagation = Propagation.REQUIRES_NEW,
               rollbackFor = Exception.class,
               timeout = 60)
public void cancelManualLegitimationCheck(
    PrepaidLegitimationCheckOrderVO orderVO) {
    . . . .
}
```

The annotation supports further configuration as well:

- the **Propagation Type** of the transaction
- the **Isolation Level** of the transaction
- a **Timeout** for the operation wrapped by the transaction
- a **readOnly flag** – a hint for the persistence provider that the transaction should be read only
- the **Rollback** rules for the transaction

.. T .. Systems ..

Potential Pitfalls

- Transactions and Proxies
- Default rollbackFor is java.lang.RuntimeException
- Read-only Transactions
- Changing the Isolation level
- Transaction logging

```
@Override  
public void foo() {  
    bar();  
}
```

```
@Transactional  
public void bar() {  
    // ...  
}
```

.. **T** .. Systems ..

- Транзакции
- Уровни изоляции транзакций
- Locking
- Savepoints
- Spring Transactional
- Distributed DBs

.. T .. Systems ..

Distributed DBs

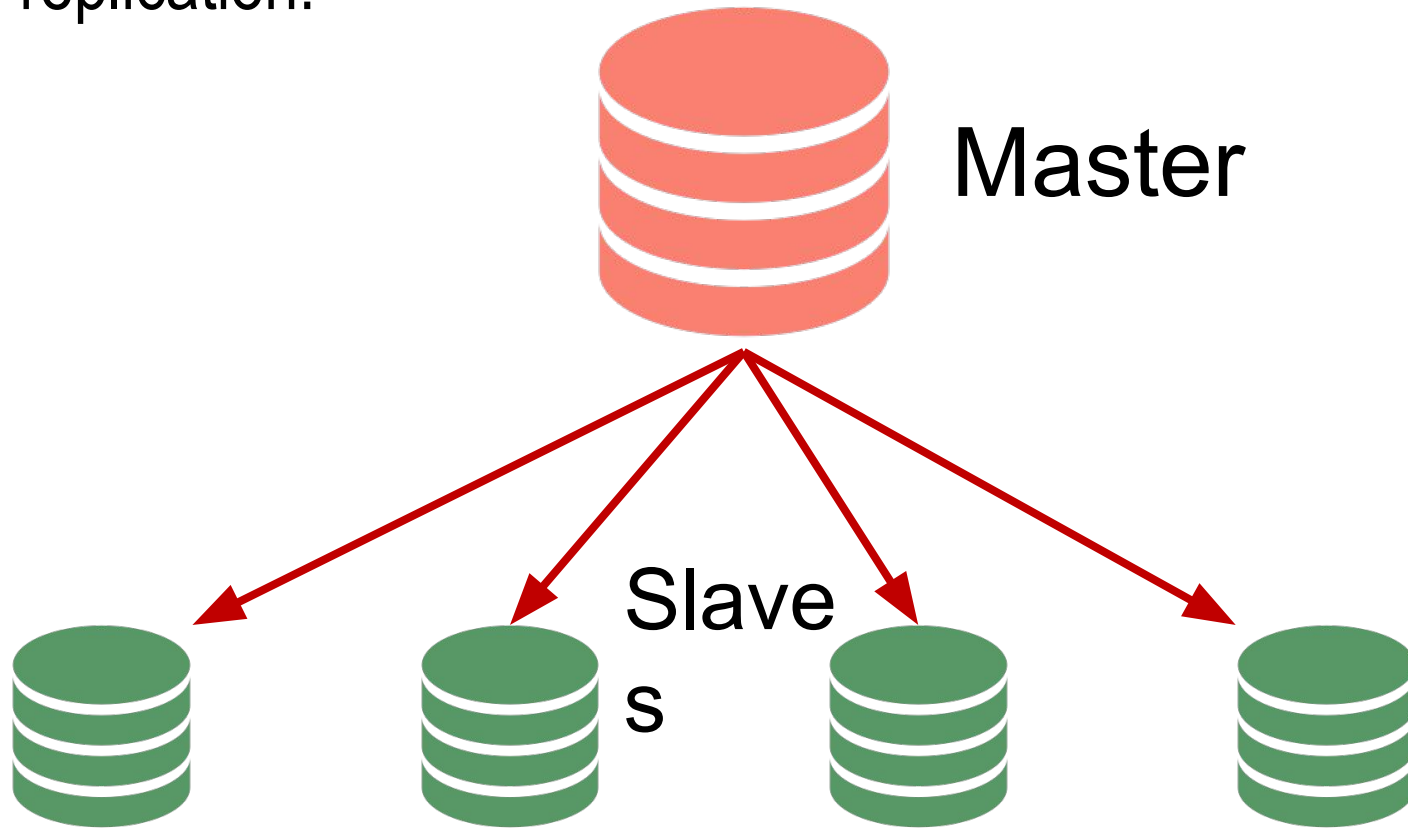


- Stateless applications are easy to scale
- Stateful apps have to share & sync their state somehow
- Many app instances are working with one DB
- Two ways to scale DB:
 - Replications
 - Sharding

... **T** ... **Systems** ...

Distributed DBs

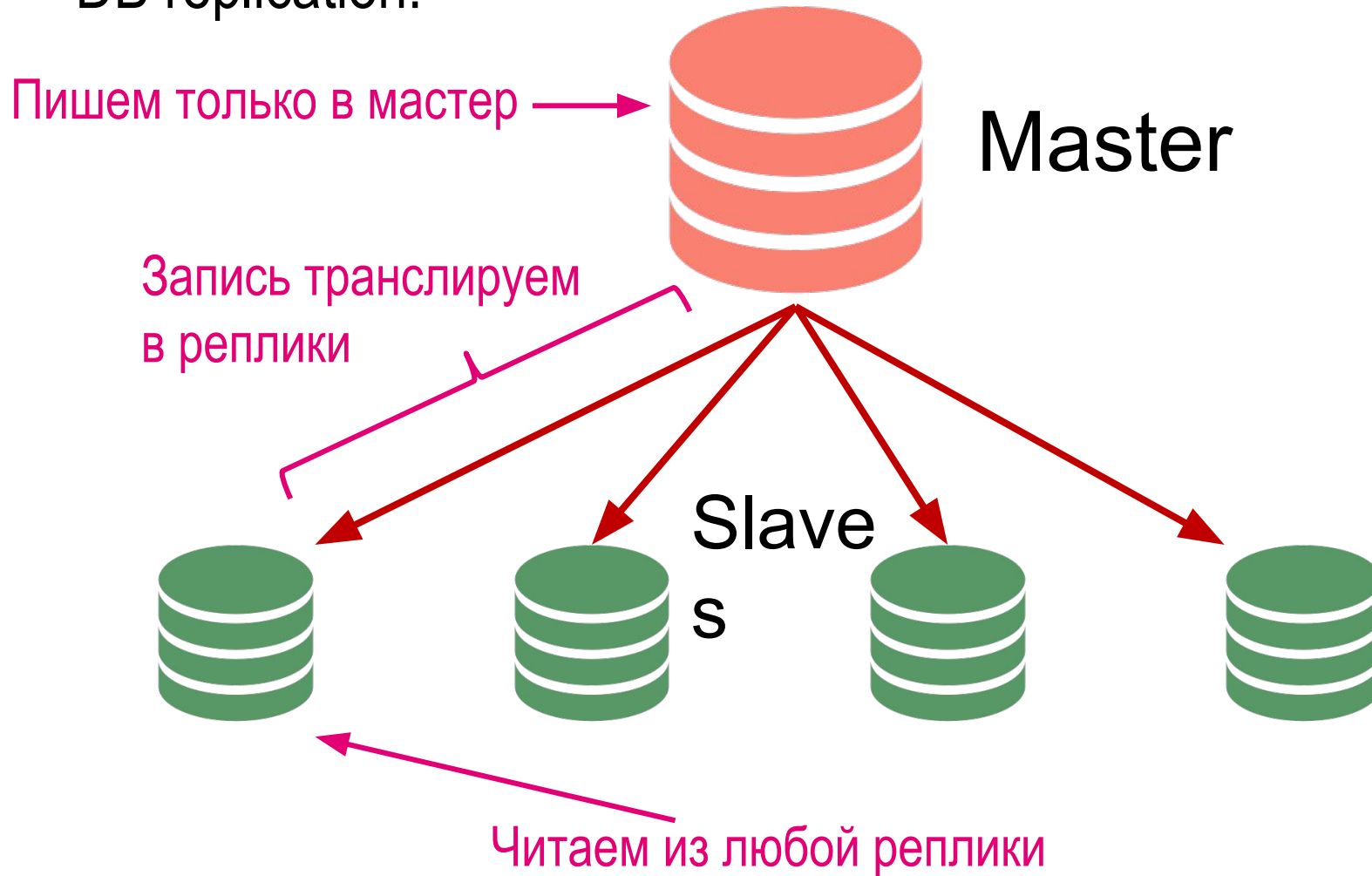
- DB replication:



... T ... Systems ...

Distributed DBs

- DB replication:



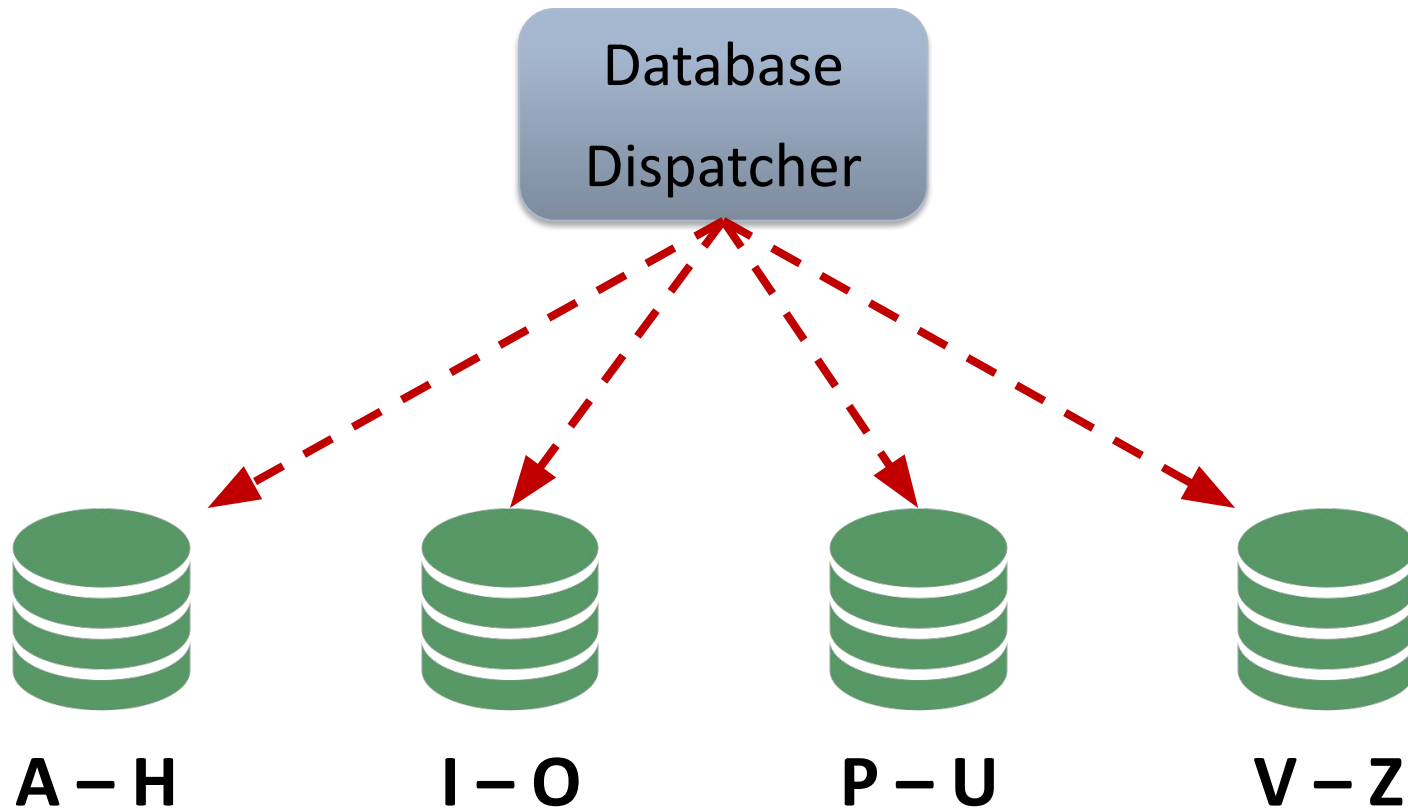
... T ... Systems ...

- DB replication:
 - Горизонтальное, но не вертикальное масштабирование
 - Многие СУБД предоставляют функционал реплицирования «из коробки»
 - Гибкость
 - Отказоустойчивость
 - Есть нюансы 
 - Стоимость Long Query остаётся той же
 - Ограниченная масштабируемость

.. T .. Systems ..

Distributed DBs

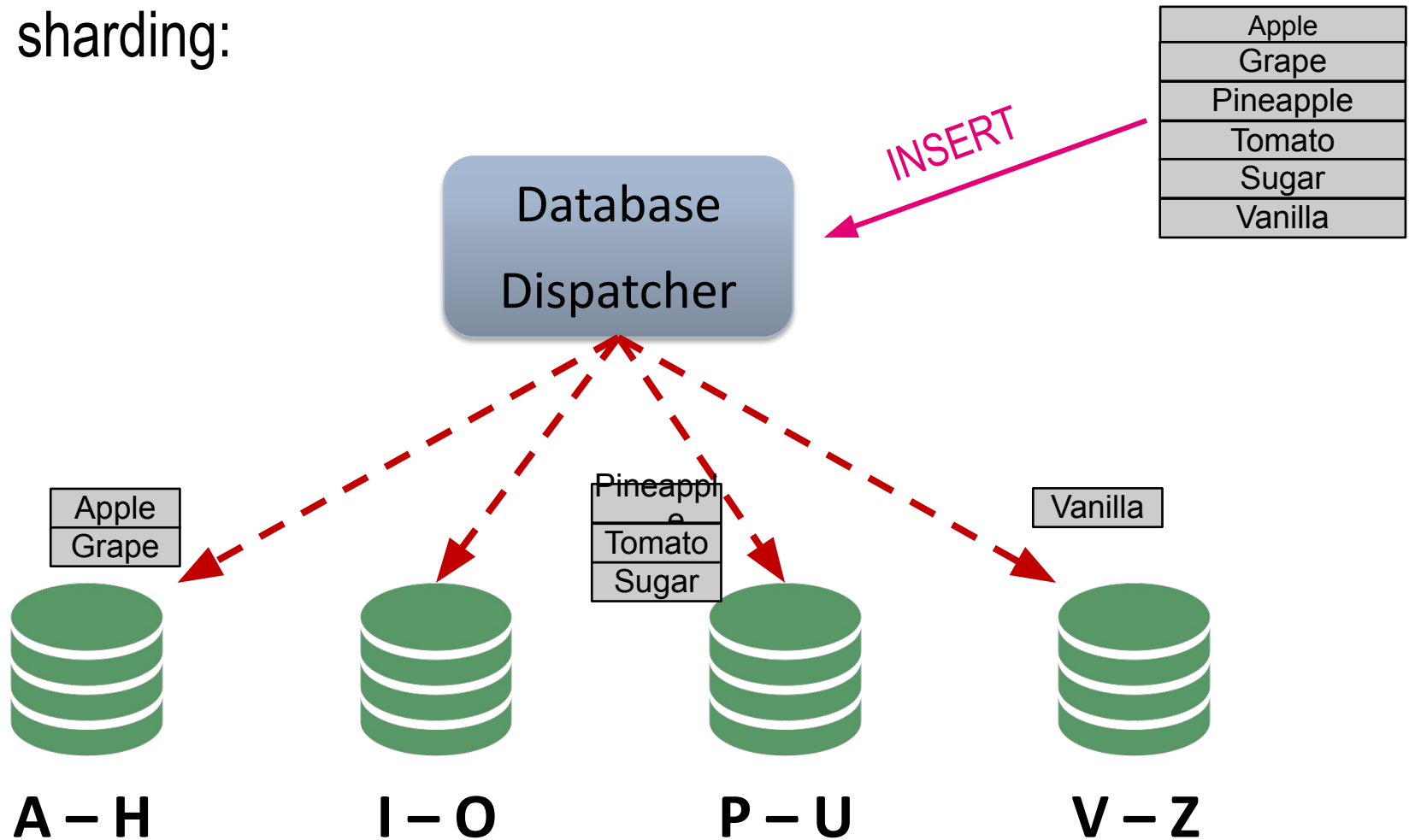
- DB sharding:



... T ... Systems ...

Distributed DBs

- DB sharding:



... T ... Systems ...

- DB sharding:
 - Может ускорить Long Query
 - Масштабируемость ограничена только логикой диспатчера
 - Риск потери данных растёт с количеством шардов
 - Можно компенсировать комбинацией с репликами
 - Сложен в реализации – нет готовых решений
 - Часто требует вмешательство в логику работы приложения
 - Постоянный «налог на шардирование»
 - Чувствителен к выбранным эвристикам



... T ... Systems ...



.. T .. Systems