Your remote PostgreSQL DBA Team

data egret

Practical transactions theory for PostgreSQL users

Ilya Kosmodemiansky

ik@dataegret.com

Plan

- What is a transaction and why do we need them
- How transactions are implemented
- Transactions in PostgreSQL

Why do we need transactions?

How much?

```
Sum: EUR 1000
-------------------------------------
                to account B     -100€
                to account C     -200€



Sum: EUR ?
-------------------------------------
```

How?

```
send_money(src_acc, dst_acc, amount):
    balance := src_acc.balance();
    if( balance - amount > 0 ):
        dst_acc.balance +=  amount;
        src_acc.balance = balance - amount;
        return 0;
    else:
        return 1;
```


Why do wee need transactions?

Action 1

```
send_money(src_acc, dst_acc, amount):
    balance := src_acc.balance();
    if( balance - amount > 0 ):
        dst_acc.balance +=  amount;


        src_acc.balance = balance - amount;
        return 0;
    else:
        return 1;
```

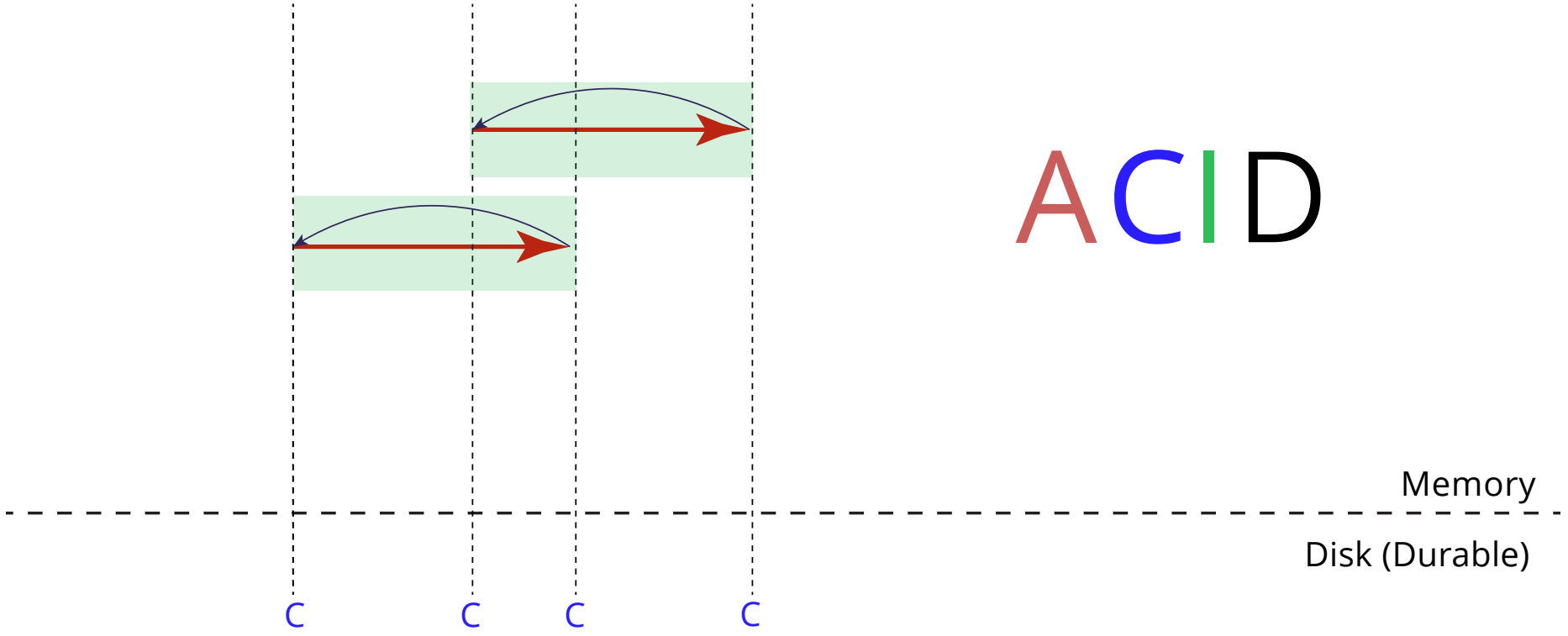
Action 2

```
send_money(src_acc, dst_acc, amount):
    balance := src_acc.balance();
    if( balance - amount > 0 ):
        dst_acc.balance +=  amount;
        src_acc.balance = balance - amount;
        return 0;
    else:
        return 1;
```


What do we already learned?

- Transfer actions consist of smaller simple actions
- Results might be affected by order
- If there is only one action, we do not have any problems
- There is no problem to read data
- Problems come when we write data
- More concurency lead to more problems

What can we do?



ACID

Action with ACID properties

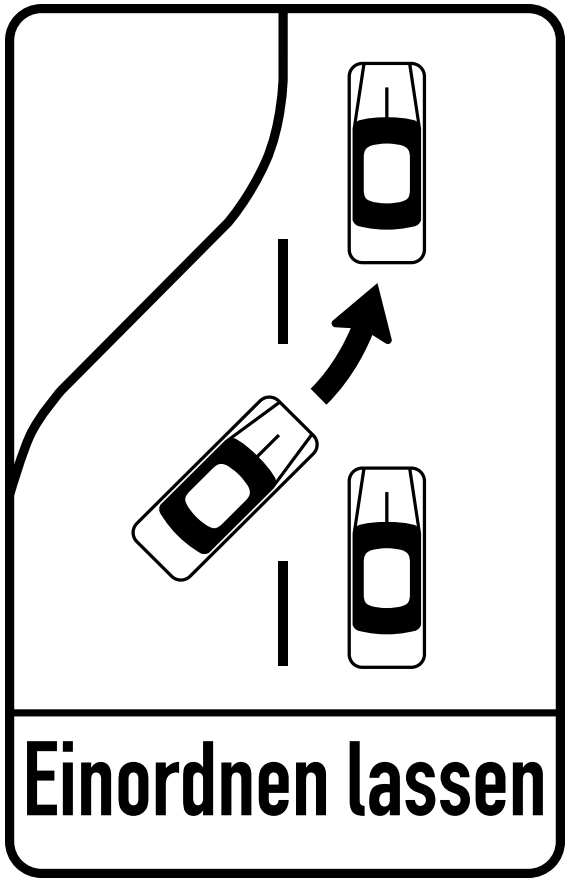
- **Atomicity** - happens completely or fails completely
- **Consistency** - brings data from one cosistent state to another
- **Isolation** - “thinks" it happens alone
- **Durability** - what is saved to disk is safe

Action with ACID properties

- **Atomicity** - happens completely or fails completely
- **Consistency** - brings data from one cosistent state to another
- **Isolation** - “thinks" it happens alone
- **Durability** - what is saved to disk is safe

We call such an action a Transaction

How transactions are implemented?

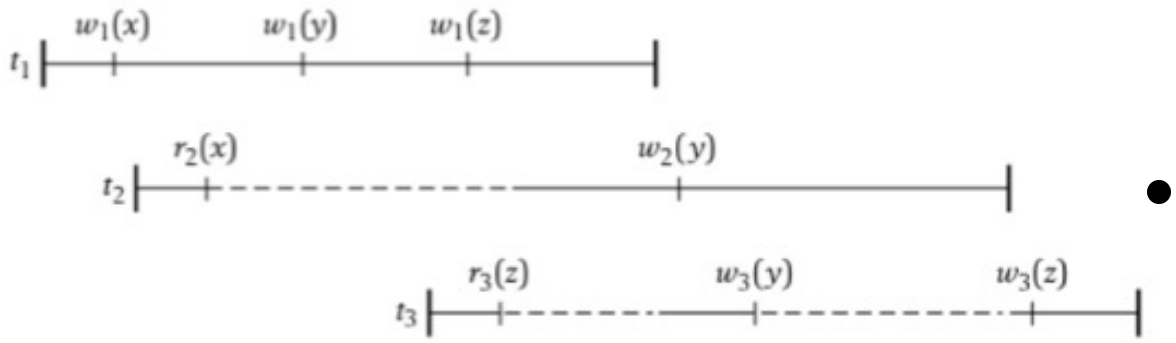


3247, Public domain, via Wikimedia Commons

Problem: Serializability

- In theory everything is simple
- … as **Reißverschlussverfahren**
- But in practice it is either slow or cause failures
- …or we need an algorythm

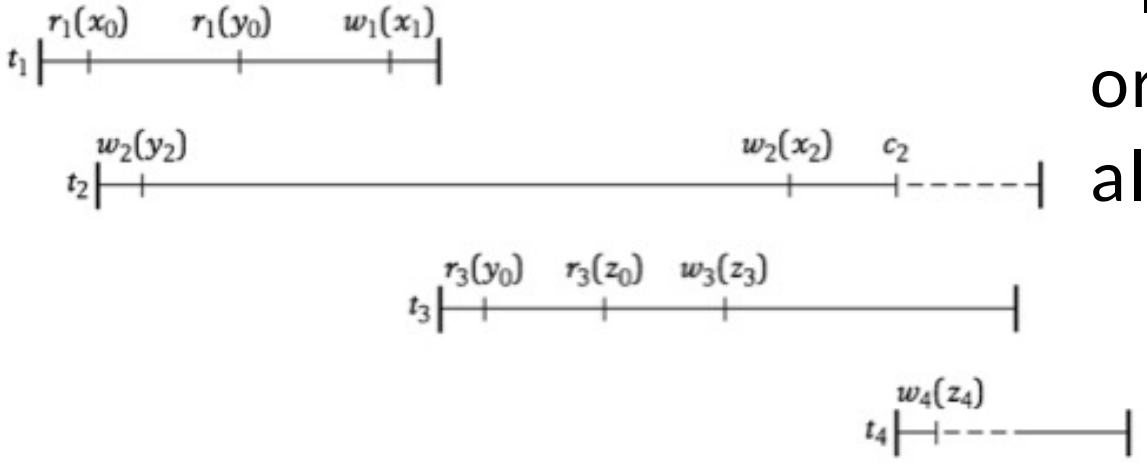
2 Phase Locking



Gerhard Weikum, Gottfried Vossen, Transactional Information Systems

- **1. Phase:** We take all locks this scope of transactions needs
- **2. Phase** We release all those locks and do not take new locks
- It works, but:
  - it is slow
  - Deadlocks

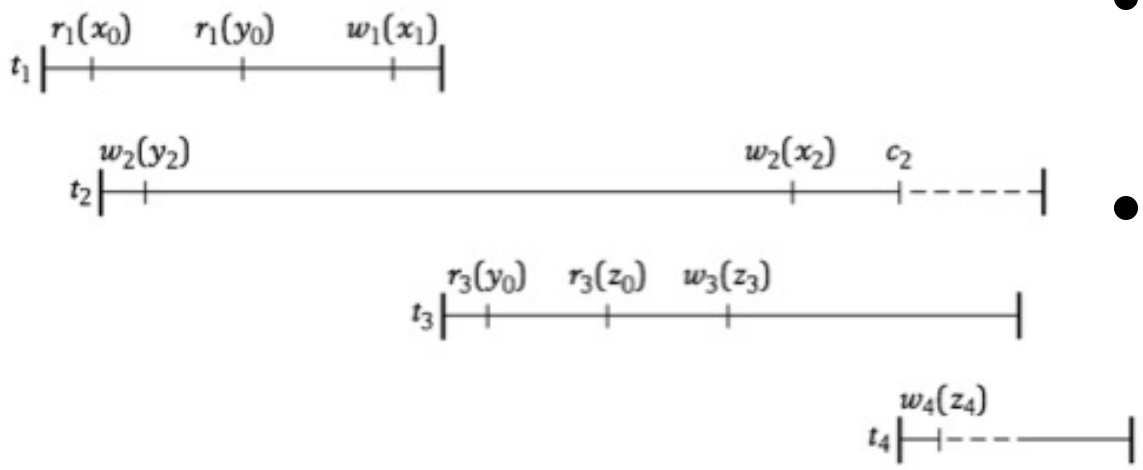
Multi Version 2 Phase Locking



* Instead of waiting, we can read or write an old version * Which is also better for serialization

Gerhard Weikum, Gottfried Vossen, Transactional Information Systems

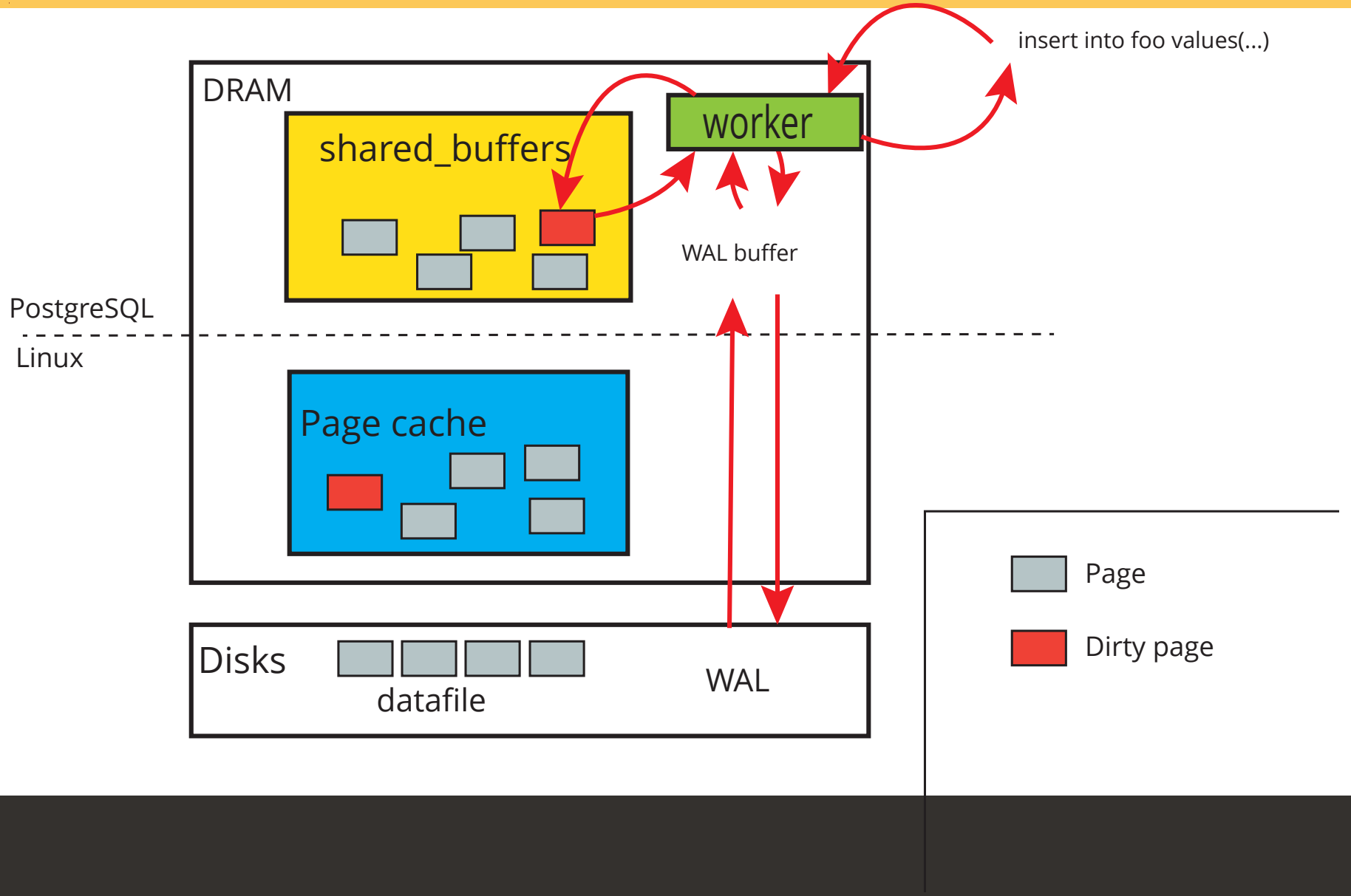
Multi Version 2 Phase Locking



- Instead of waiting, we can read or write an old version
- Which is also better for serialization

Gerhard Weikum, Gottfried Vossen, Transactional Information Systems

Postgres: inserting data



Postgres: inserting data

Internal

```
StartTransactionCommand;
    StartTransaction;
ProcessUtility;
    BeginTransactionBlock;
CommitTransactionCommand;

StartTransactionCommand;
ProcessQuery;
CommitTransactionCommand;
    CommandCounterIncrement;

StartTransactionCommand;
ProcessUtility;
    EndTransactionBlock;
CommitTransactionCommand;
    CommitTransaction;
```

SQL

```
BEGIN;



INSERT...



COMMIT
```


in PostgreSQL everything is transactional

- a writing transaction get a **XID**
- each tuple has **xmin** (**XID** of the "yangest" transaction which has updated this tuple
- and **xmax** (**XID** of the oldest transaction which can see this tuple
- each backend has its **xmin** and syncronizes it through **MyProc->xmin** - it is a way how Snapshot works

PostgreSQL is a MVCC Database

- that dooesn't mean, that there is no 2PL
- it is possible to go through many old versions, but through all of them
- At some point we were considering such an architeckture as a fail, but now it is rather future

Questions?

ik@dataegret.com