Optimistic and pessimistic offline lock

Optimistic and pessimistic offline lock

- **Naivní přístup:**
  - Uživatel A načte data
  - Uživatel B načte (stejná) data
  - Uživatel A upraví data a uloží je (např do DB)
  - Uživatel B data také upraví a uloží
  - ..... kde je problém?

- **Řešení:**
  - Zámky

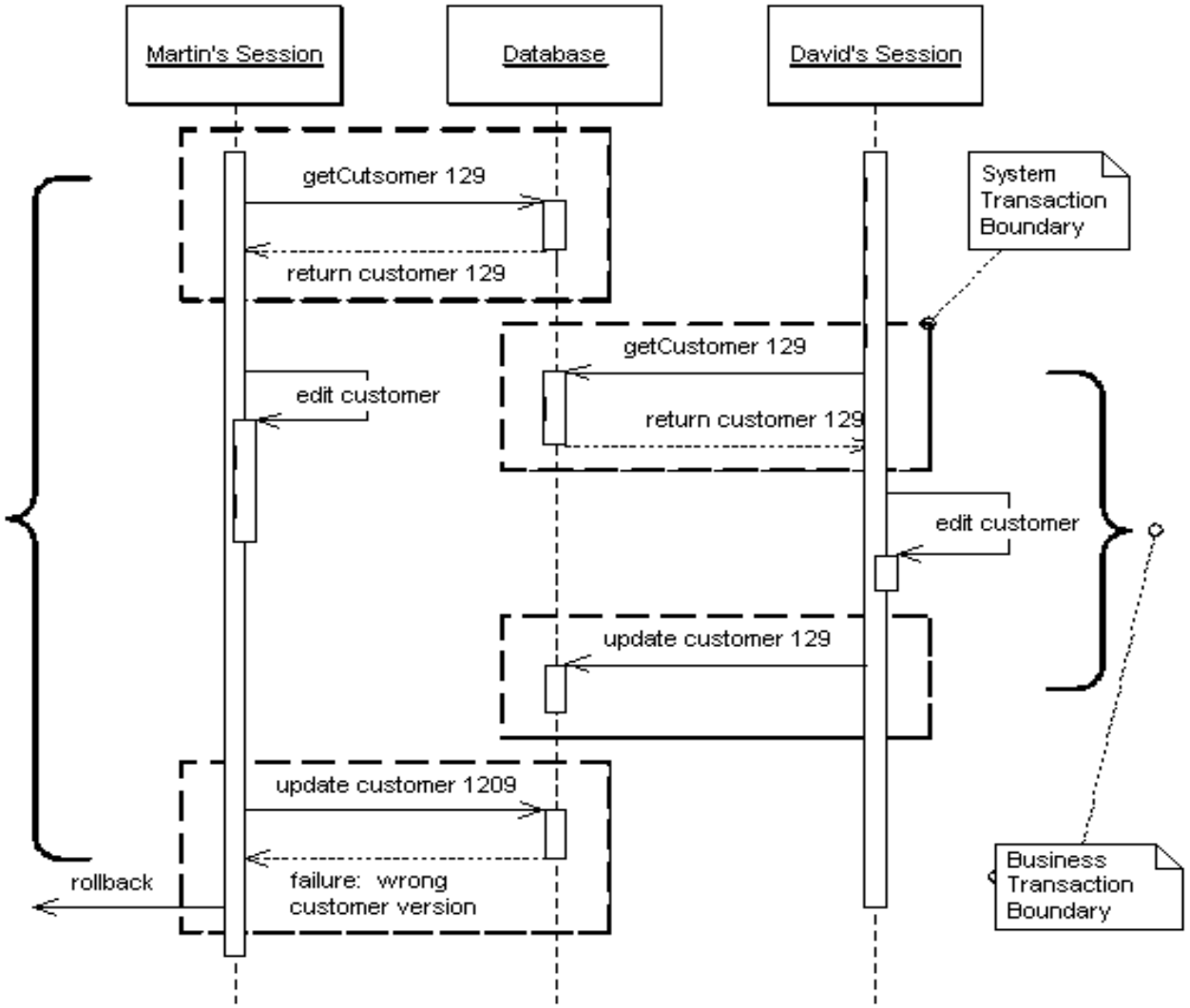
Optimistic offline lock

- **Základní prvky:**
  - Založeno na myšlence, že nám nikdo nezmění data
  - Neprovádí se zamykání položek
  - Každý prvek má počítadlo

- **Počítadlo (counter):**
  - Při změně nebo smazání položky se provede inkrementace počítadla
  - Pamatujeme si hodnotu počítadla
  - Pokud nám někdo změnil data, hodnota počítadla nesouhlasí se zapamatovanou hodnotou

- **Důsledky:**
  - Konflikt se odhalí až při zápisu
  - Nutnost navrácení se do původního stavu

Optimistic offline lock



Optimistic offline lock

- **Vylepšení:**
  - Rychlejší detekce uzamčení – časovač s vysokým rozlišením
- **Ale:**
  - nedoporučuje se ho používat – více serverů
- **Kdy se používá:**
  - Když se pravděpodobnost na konflikt mezi dvěma transakcemi nízká

Optimistic offline lock

- **Jak by mohla vypadat implementace:**

```
function saveData($myData){
    global $db;
    try{
        $db->startTransaction();

        $db->query("SELECT `revision` FROM `data_storage` WHERE `id` = '".$myData->getId()."';");
        if ($db->getResultInt() != $myData->getRevision())
            throw new LockException("Během upravování dat je upravil někdo jiný.");

        $myData->incRevision();  // increase object revision
        $db->query("UPDATE `data_storage` SET `foo` = '".$myData->getFoo()."', "
            ." ... "
            ."WHERE `id` = '".$myData->getId()."';");

        $db->commitTransaction();
    } catch (Exception $e) {
        $db->endTransaction();
        /* rethrow it */
        throw $e;
    }
}
```

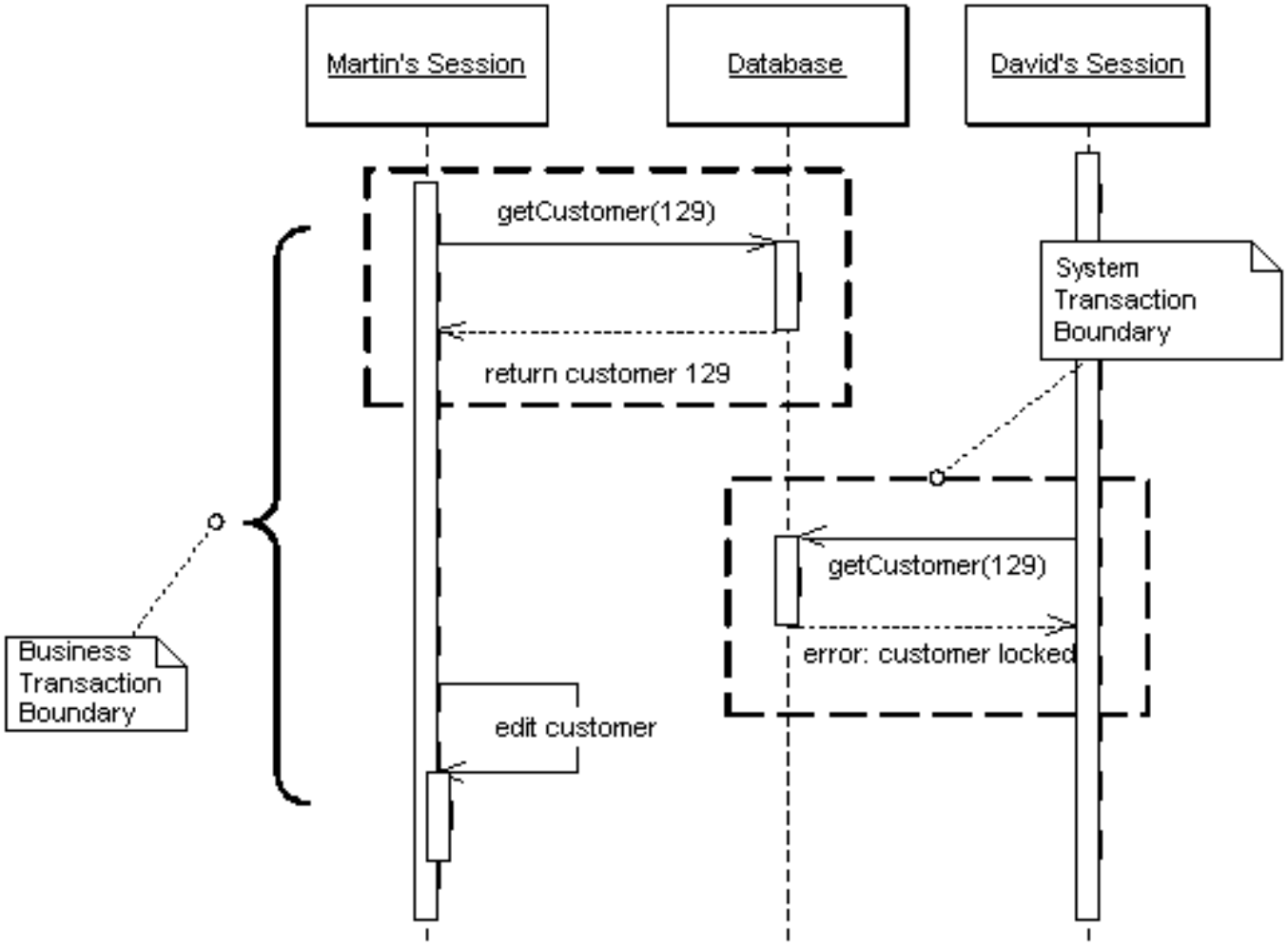
Pessimistic offline lock

- **Základní prvky:**
  - Založeno na myšlence, že ostatní budou měnit data
  - Provádí se zamykání položek
  - Pouze jeden přístup

- **Druhy zámků:**
  - Exclusive write lock
  - Exclusive read lock
  - Read / write lock

- **Důsledky:**
  - Nevznikají konflikty – není potřeba vracet se do původního stavu
  - V nejhorším případě jen jeden může pracovat s uzamčenými daty

Pessimistic offline lock



Pessimistic offline lock

- **Nevýhody**
  - Možnost vzniku deadloků – odstranění pomocí timeoutů

Pessimistic offline lock

- **Ukázka použití**

```
public Row loadAndLockRow(int rowId) throws LockException {
  Row lockedRow;
  try {
    DbConnector.startTransaction();
    lockedRow = dataManager.getRowById(rowId);
    if (lockedRow == null) {
      Logger.getLogger(this.getClass()).debug("No appropriate row");
      throw new RuntimeException(String.format("Řádek id %d neexistuje", rowId));
    }
    if (!dataManager.lockRow(lockedRow, user, true)) {
      UserImpl other = dataManager.getLockUser(rowId);
      throw new LockException(String.format("Záznam id %d je již zamčen uživatelem %s",
                                            rowId, other.toString()));
    }
    DbConnector.commitTransaction();
    return lockedRow;
  } finally {
    DbConnector.endTransaction();
  }
}
```

Optimistic and pessimistic offline lock

- **Optimistic offline lock:**
  - Neuzamyká data
  - Malé souběžné přístupy k datům
  - Detekce konfliktu na konci transakce

- **Pessimistic offline lock:**
  - Uzamyká data
  - Velké přístupy k datům
  - Detekce konfliktu na počátku transakce
  - Existence více druhů zámků

Vzor – shrnutí

- **Jakou implementaci?**
  - Jako u jiných návrhových vzorů závisí na prostředí nasazení...
  - Je potřeba zvážit:
    - Jaký je poměr editací a prohlížení (read-only)?
    - Jaká je pravděpodobnost souběhu přístupů?
    - Pracuje systém se systémy třetích stran? Podporují rollback? (např. Banka...)
    - Jaká je povaha zadávaných dat? (uživatelské údaje, smlouva o stovkách položek...)

- **Související vzory**
  - Všechno souvisí se vším a vzory lze různě kombinovat :)
  - **Identity map** – pokud k databázi přistupuje jedna instance serveru, nemusíme propagovat zámky do databáze, stačí je mít v Identity mapě
  - **Table/Row Data Gateway –** řešení zámků může být na této úrovni (?)