METODY PROGRAMOWANIA

Wzorzec Unit of Work

4 listopada 2017

Krzysztof Pawłowski
kpawlowski@pjwstk.edu.pl

PO CO NAM UNIT OF WORK?

- Dużo małych zapytań
- Transakcje
- Śledzenie zmian

TRANSAKCJE BAZODANOWE

- Zbiór operacji na bazie, które stanowią całość i powinny być wykonane wszystkie albo żadna z nich

- Transakcje powinny spełniać warunki ACID

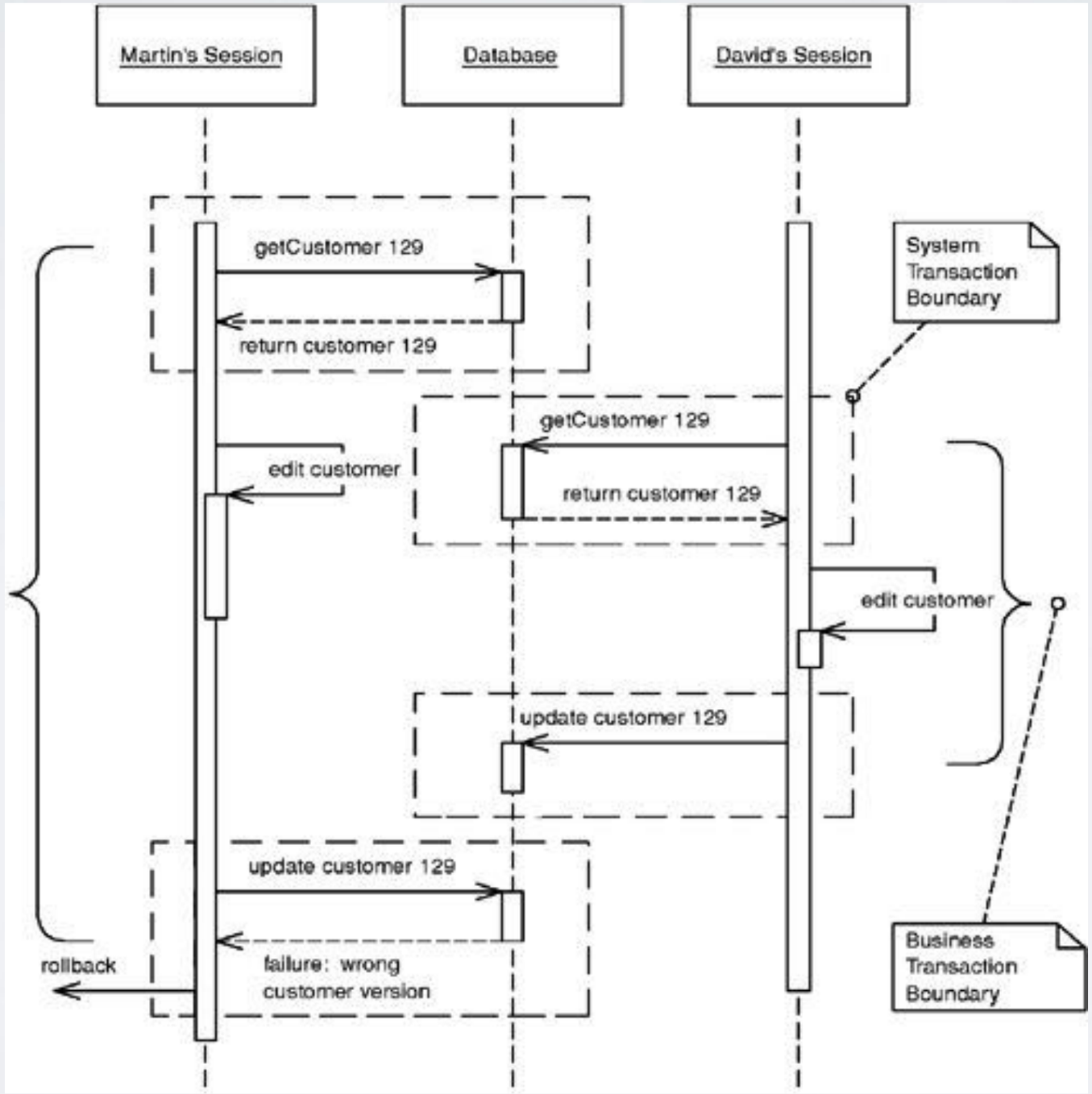
ZASADY ACID

- Atomicity
- Consistency
- Isolation
- Durability

BLOKADA

- Blokada (ang. lock) - mechanizm służący do zapobiegania konfliktom w dostępie do zasobów w środowiskach wielozadaniowych.

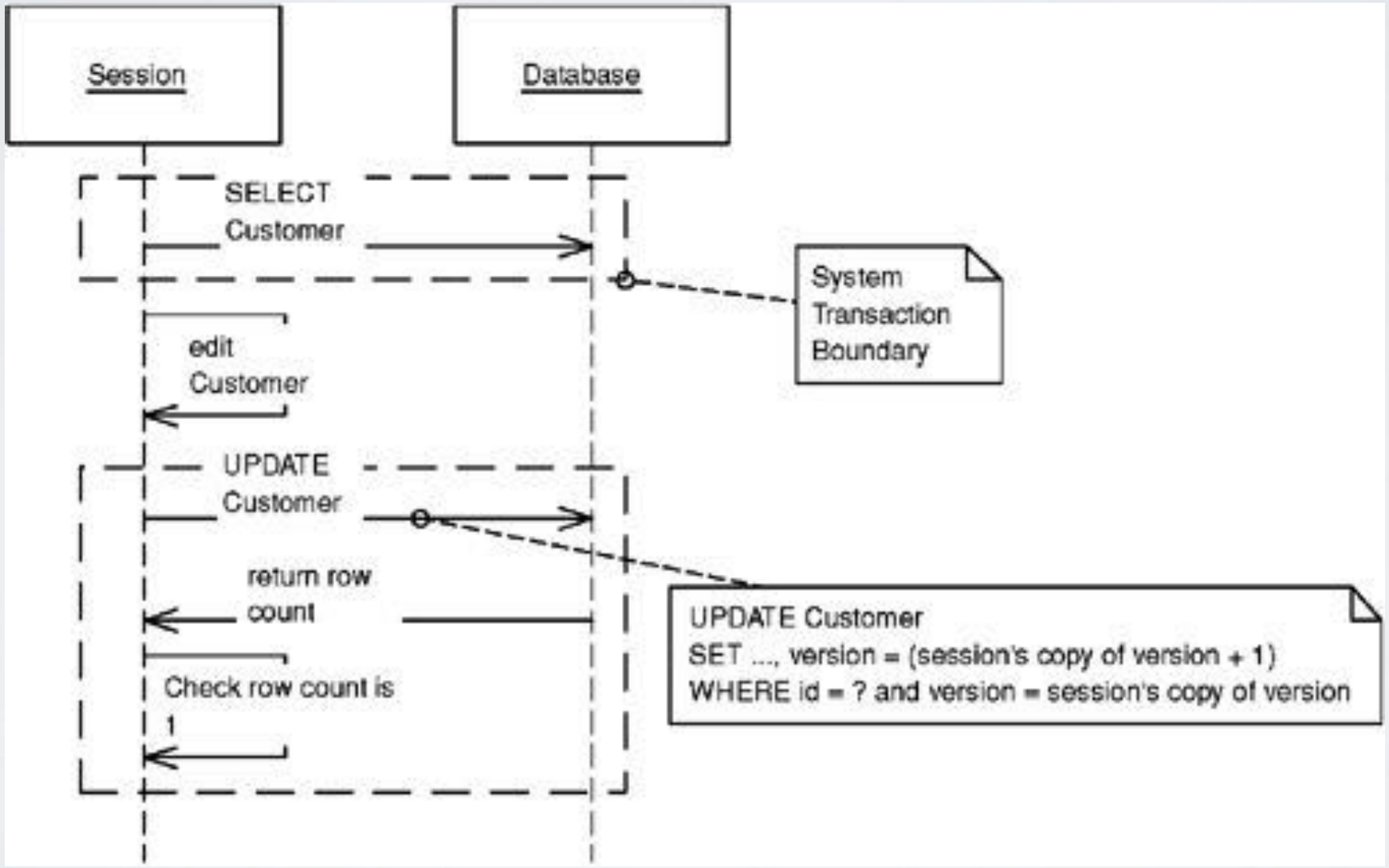
KIEDY POTRZEBA JEST BLOKADA?



OPTIMISTIC OFFLINE LOCK

- Zapobiega konfliktom poprzez sprawdzenie czy rekord, który chcemy zmienić nie został zmieniony przez kogoś innego.
- Po wykryciu konfliktu zmiany są wycofywane.
- Operacja walidacji i modyfikacji musi być atomowa

OPTIMISTIC OFFLINE LOCK



PRZYKŁADY OPTIMISTIC OFFLINE LOCK

- System wyliczający podatek na podstawie adresu zamieszkania - co jeśli adres się zmieni w trakcie wykonywania operacji?

- System kontroli wersji - dlaczego optimistic lock się tutaj sprawdza?

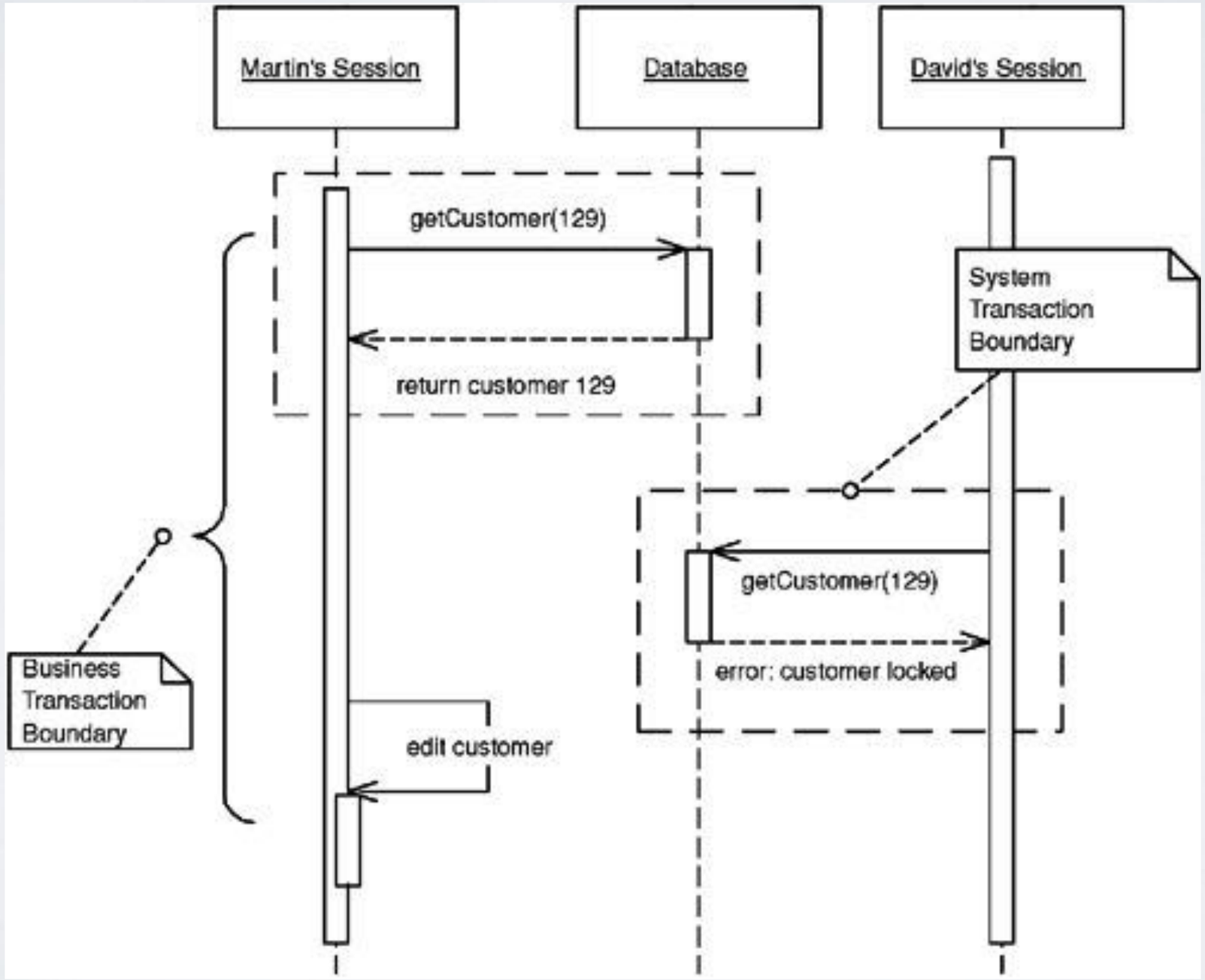
OPTIMISTIC OFFLINE LOCK - KIEDY UŻYWAĆ?

- Gdy ryzyko konfliktu jest małe
- Gdy zależy nam na wydajności

PESSIMISTIC OFFLINE LOCK

- Zapobiega konfliktom poprzez umożliwienie dostępu do danych tylko jednemu użytkownikowi w tym samym czasie

PESSIMISTIC OFFLINE LOCK



PESSIMISTIC OFFLINE LOCK

- Rodzaje blokad w Pessimistic Offline Lock:
  - Read lock
  - Write lock
  - Read/Write lock

PESSIMISTIC OFFLINE LOCK

- Menedżer blokad (ang. lock manager)
- Procedura zakładania blokady:
  - kiedy założyć blokadę?
  - na jakich obiektach założyć blokadę?
  - kiedy zwolnić blokadę?
  - co zrobić jeśli nie da się uzyskać blokady?

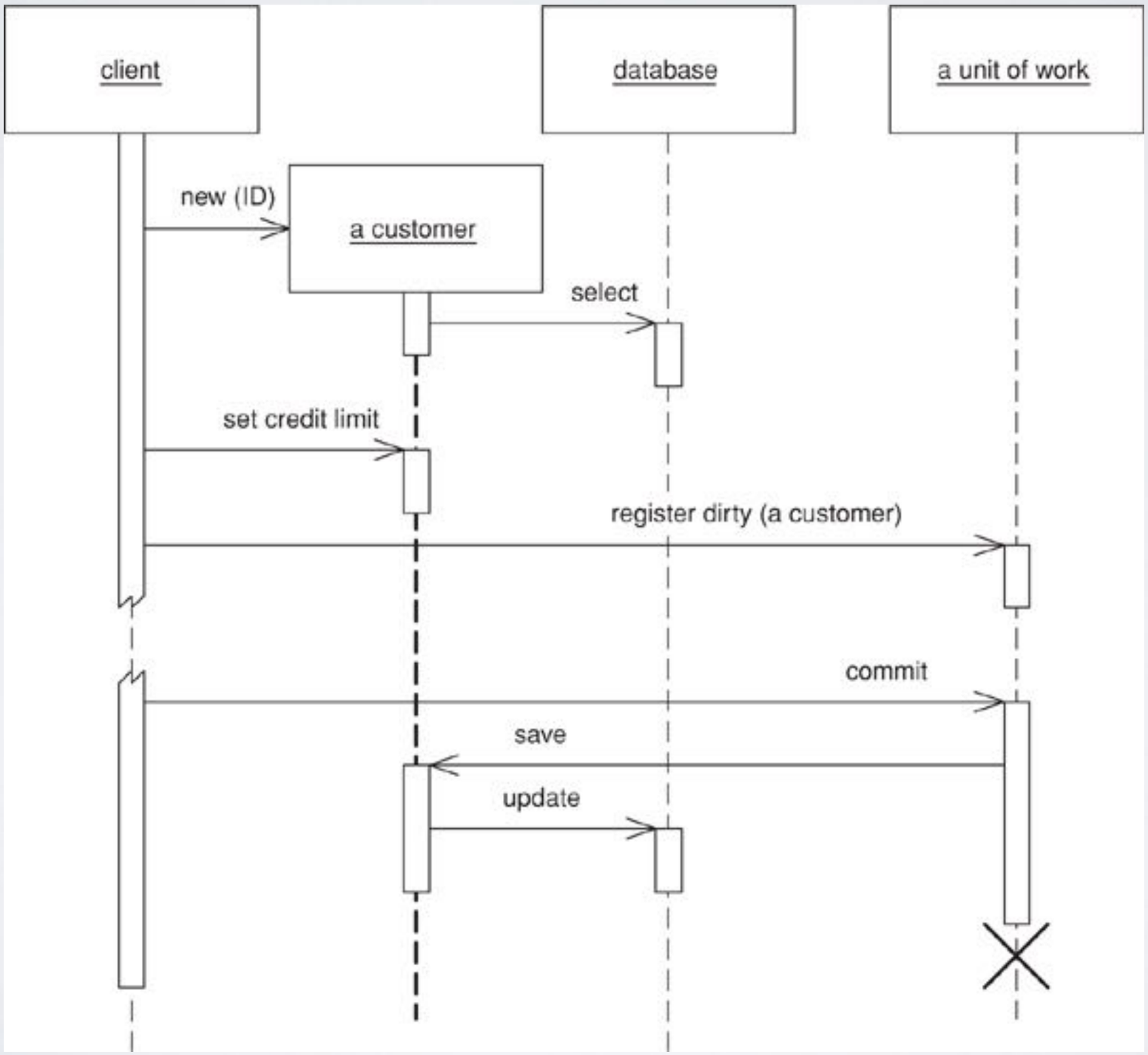
PESSIMISTIC OFFLINE LOCK - KIEDY UŻYWAĆ?

- Gdy ryzyko konfliktu jest wysokie
- Gdy koszt konfliktu jest zbyt wysoki

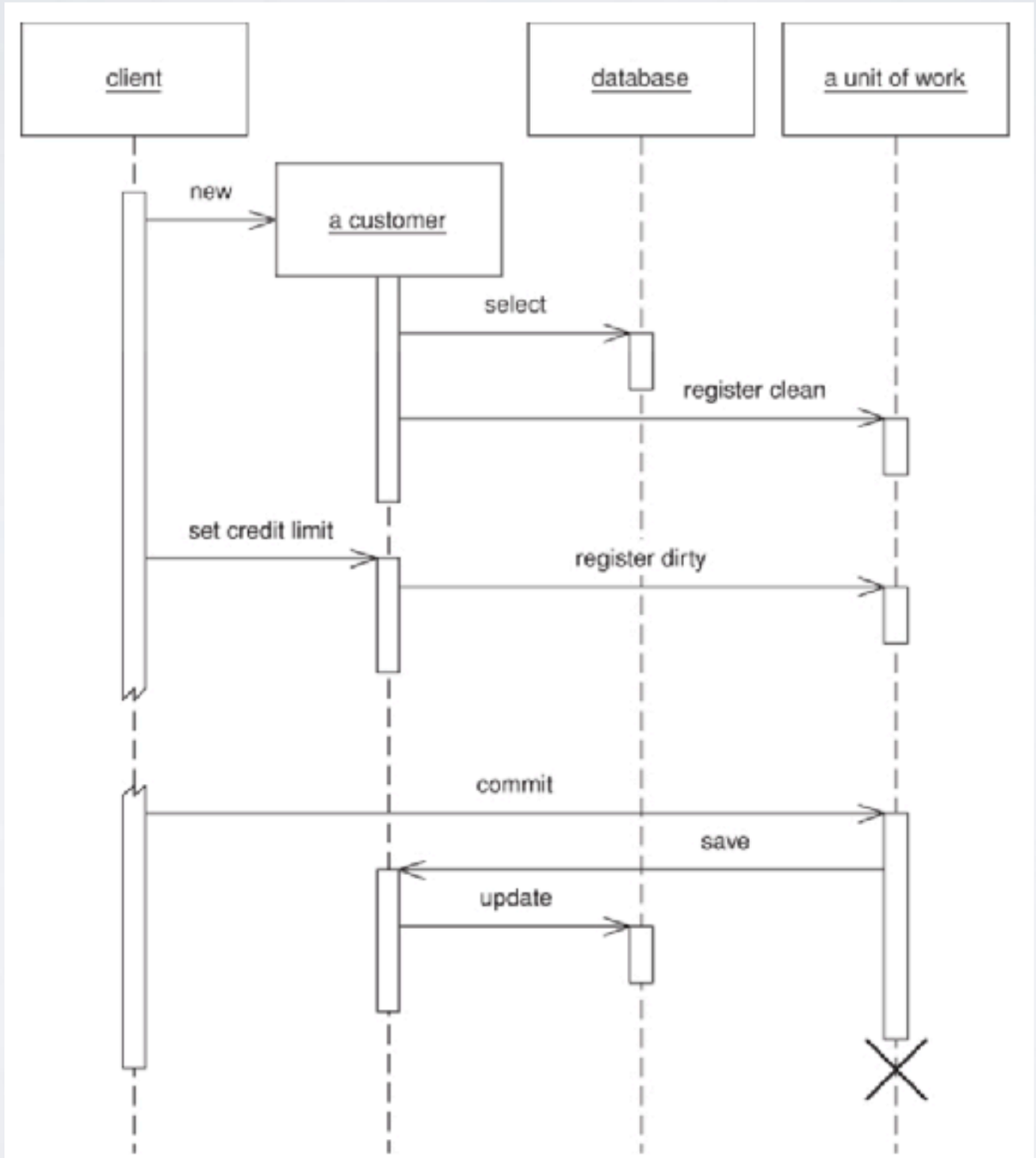
ŚLEDZENIE ZMIAN

- Śledzone są tylko zarejestrowane obiekty
  - registerNew(object)
  - registerDirty(object)
  - registerClean(object)
  - registerDeleted(object)
  - commit()

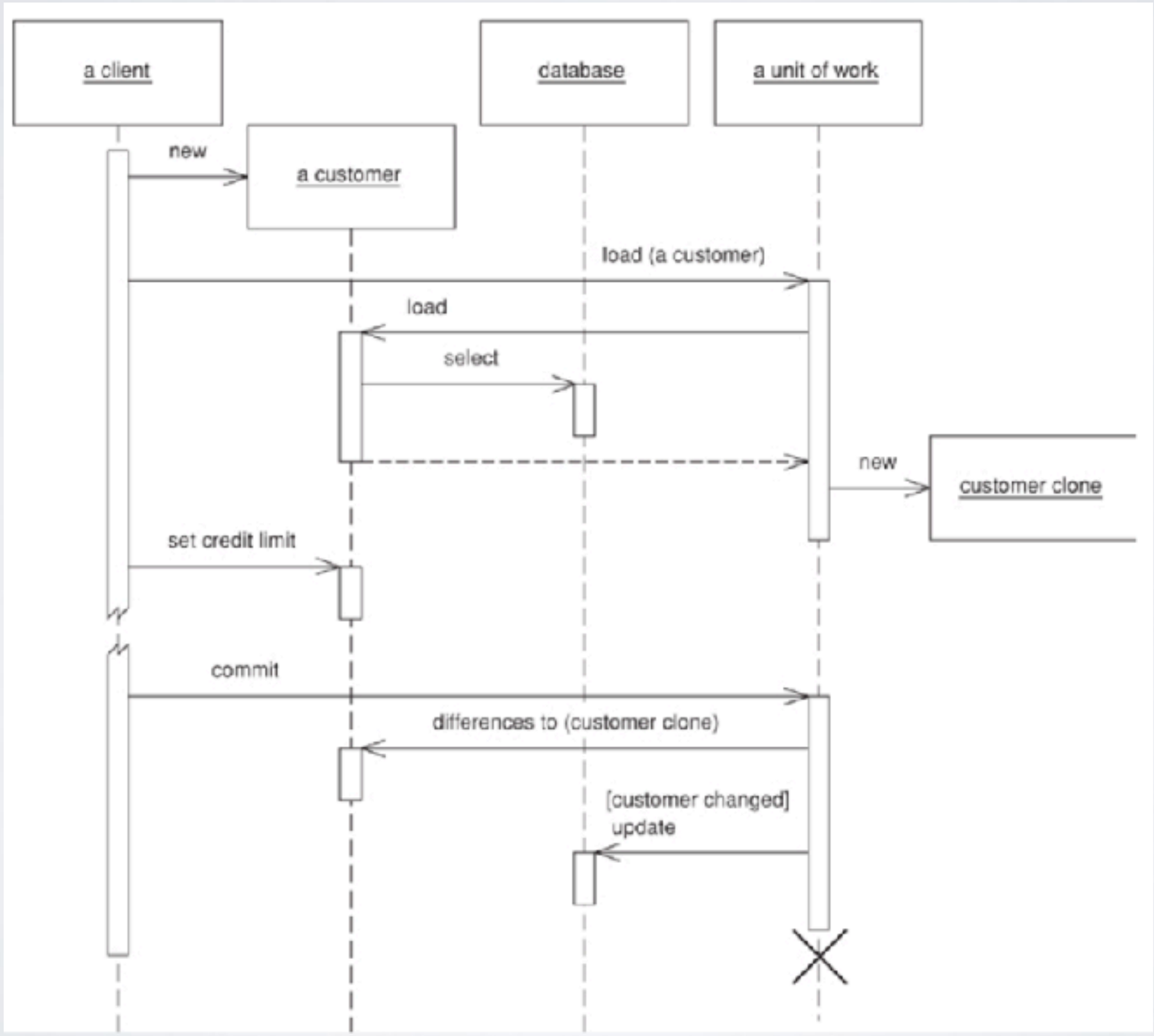
UNIT OF WORK - WARIANT I



UNIT OF WORK - WARIANT II



UNIT OF WORK - WARIANT III



PRZYKŁADOWA IMPLEMENTACJA

```
class UnitOfWork...

private List newObjects = new ArrayList<Entity>();
private List dirtyObjects = new ArrayList<Entity>()();
private List removedObjects = new ArrayList<Entity>()();
```

PRZYKŁADOWA IMPLEMENTACJA

```
class UnitOfWork...

public void registerNew(Entity e) {
  Assert.notNull("id not null", e.getId());
  Assert.isTrue("not dirty", !dirtyObjects.contains(e));
  Assert.isTrue("not removed", !removedObjects.contains(e));
  Assert.isTrue("not new already", !newObjects.contains(e));
  newObjects.add(e);
}
```

PRZYKŁADOWA IMPLEMENTACJA

```
class UnitOfWork...

public void registerDirty(Entity e) {
  Assert.notNull("id not null", e.getId());
  Assert.isTrue("object not removed",
    !removedObjects.contains(e));
  if (!dirtyObjects.contains(e) && !
    newObjects.contains(e)) {
    dirtyObjects.add(e);
  }
}
```

PRZYKŁADOWA IMPLEMENTACJA

```
class UnitOfWork...

public void registerRemoved(Entity e) {
  Assert.notNull("id not null", e.getId());
  if (newObjects.remove(e)) return;
  dirtyObjects.remove(e);
  if (!removedObjects.contains(e)) {
    removedObjects.add(e);
  }
}
```

PRZYKŁADOWA IMPLEMENTACJA

```
class UnitOfWork...

public void registerClean(Entity e) {
  Assert.notNull("id not null", e.getId());
}
```

PRZYKŁADOWA IMPLEMENTACJA

```
class UnitOfWork...


public void commit() {
  insertNew();
  updateDirty();
  deleteRemoved();
}


private void insertNew() {
  for (Iterator objects = newObjects.iterator(); objects.hasNext();) {
    Entity e = (Entity) objects.next();
    MapperRegistry.getMapper(e.getClass()).insert(obj);
  }
}
```

PRZYKŁADOWA IMPLEMENTACJA

```
class UnitOfWork extends ThreadLocal...


private static ThreadLocal current = new ThreadLocal();

public static void newCurrent() {
  setCurrent(new UnitOfWork());
}

public static void setCurrent(UnitOfWork uow) {
  current.set(uow);
}

public static UnitOfWork getCurrent() {
  return (UnitOfWork) current.get();
}
```

PRZYKŁADOWA IMPLEMENTACJA

```
class Entity...


protected void markNew() {
  UnitOfWork.getCurrent().registerNew(this);
}

protected void markClean() {
  UnitOfWork.getCurrent().registerClean(this);
}

protected void markDirty() {
  UnitOfWork.getCurrent().registerDirty(this);
}

protected void markRemoved() {
  UnitOfWork.getCurrent().registerRemoved(this);
}
```

PRZYKŁADOWA IMPLEMENTACJA

```
class Album extends Entity...

public static Album create(String name) {
  Album obj = new Album(IdGenerator.nextId(), name);
  obj.markNew();
  return obj;
}

public void setTitle(String title) {
  this.title = title;
  markDirty();
}
```