Task 1: In the figures below, please check whether the implementation of the Conversation (User1 raising the salary of the Guide[id=2] to 3000) is described truly.

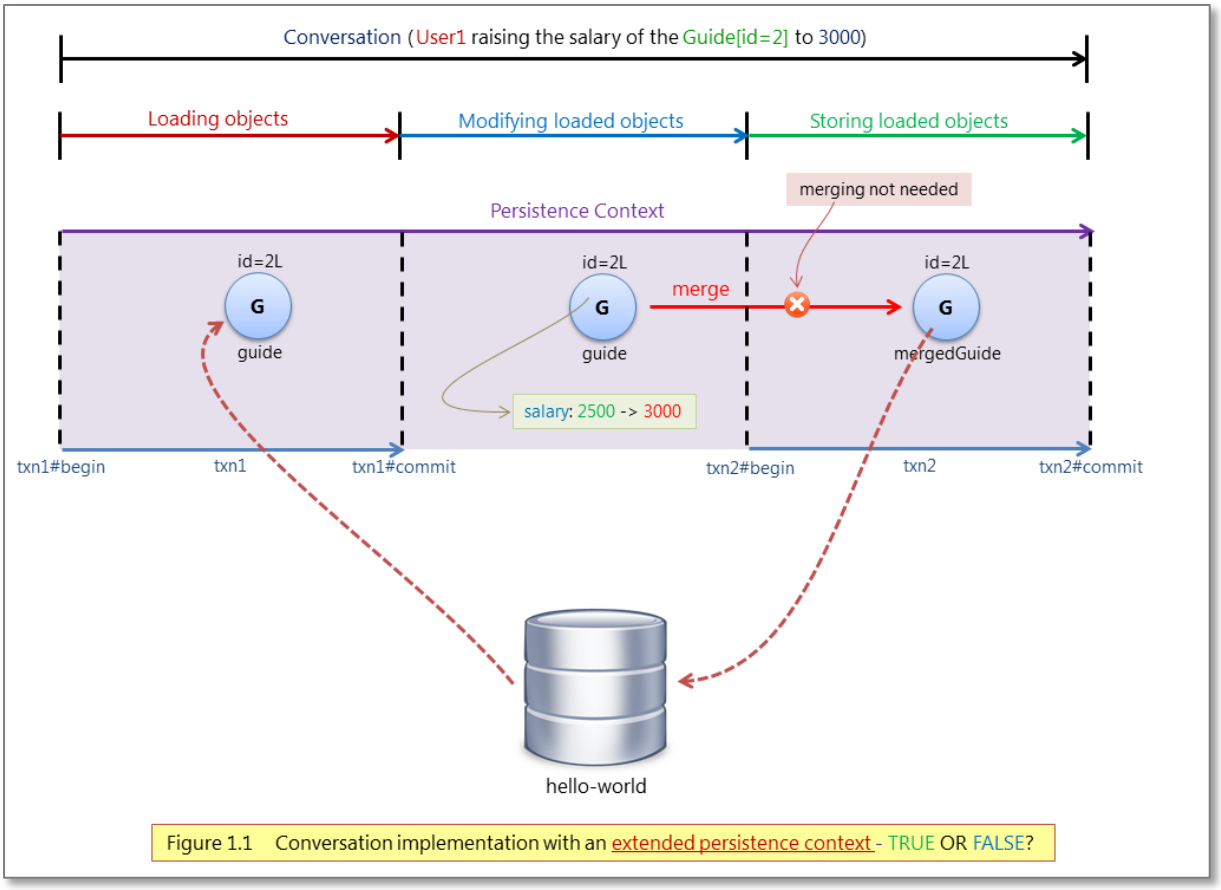

Figure 1.1 Conversation implementation with an extended persistence context - TRUE OR FALSE?

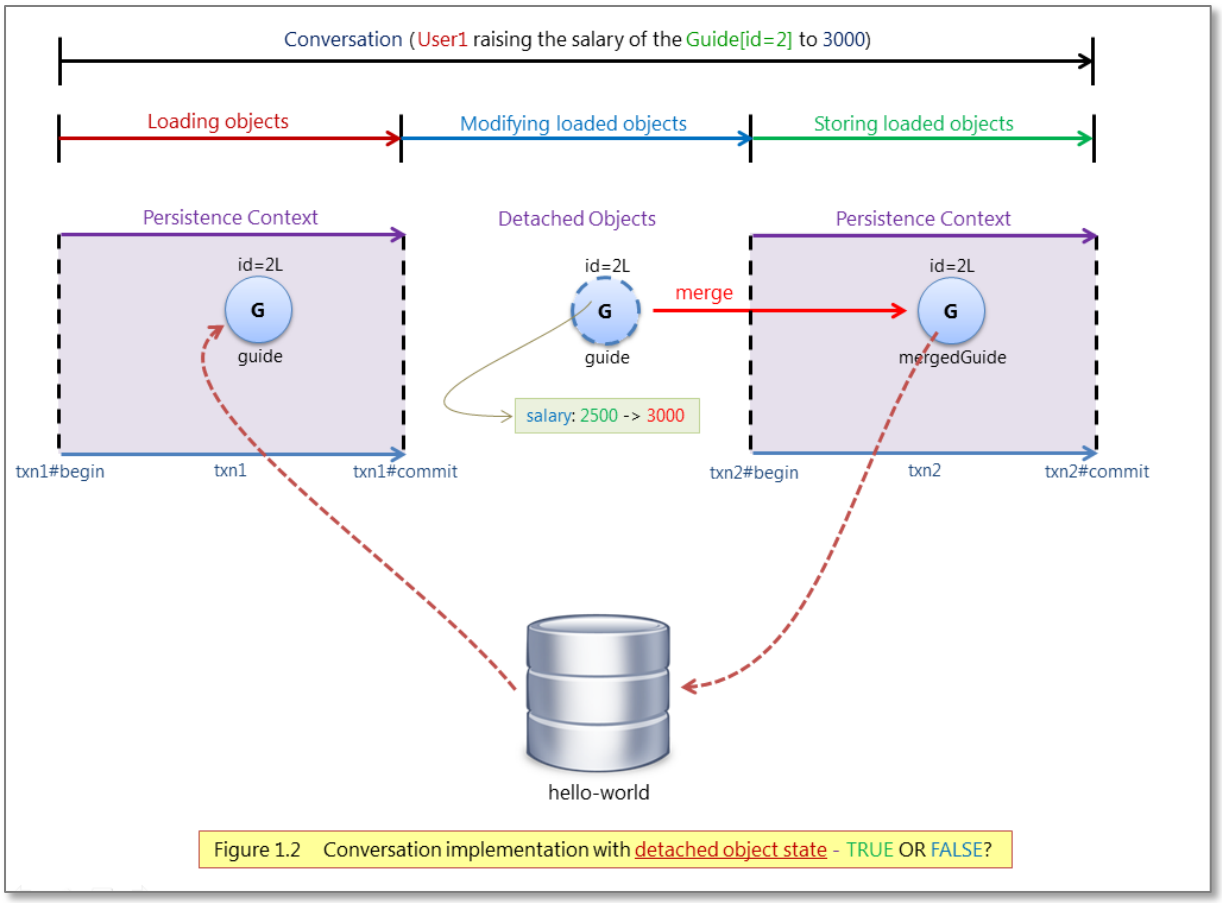

Figure 1.2 Conversation implementation with detached object state - TRUE OR FALSE?

Task 2: As explain in the lecture on Optimistic Locking and Versioning, simulate a Lost Update situation while implementing the concurrent conversations of two users, User1 and User2, where User1 wants to modify the salary of the Guide[id=2] to 3000, whereas User2 wants to modify the salary of the same Guide to 4000.

Consider the following to be the starting state of the guide table:

guide

hello-world.guide: 3 rows total (approximately)

id	name	salary	staff_id
1	Mike Lawson	1000	2000MO10789
2	Ian Lamb	2500	2000IM10901
3	David Crow	3000	2000DO10777

Task 3: While simulating the Lost Update situation in Task 2, please check whether the last transaction commit won (usually referred as Last Commit Wins) or not, as it was explained in the lecture?

Task 4: Solve the Lost Update situation while performing the Task 2 using the Versioning strategy (Optimistic Locking).

Hint: Add the version column in the guide table using the following SQL:

```
ALTER TABLE `guide`
            ADD `version` INT(11) NOT NULL DEFAULT '0';
```

The source code files for the lecture on "Optimistic Locking and Versioning" are available to be downloaded with this lab exercise. You could use them to complete the given tasks successfully.

You could also look at the source code inside all the downloadable files below.

Source Code: Optimistic Locking and Versioning

Schema for hello-world database

```
CREATE DATABASE `hello-world`;
```

persistence.xml

(to be placed inside META-INF folder; the META-INF folder should be in the root of the classpath of your application)

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
   version="2.0" xmlns="http://java.sun.com/xml/ns/persistence">

   <persistence-unit name="hello-world" transaction-type="RESOURCE_LOCAL">
      <properties>

         <!-- Database connection settings -->
         <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver" />
         <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/hello-
world" />
         <property name="javax.persistence.jdbc.user" value="root" />
         <property name="javax.persistence.jdbc.password" value="password" />

         <!-- SQL dialect -->
         <property name="hibernate.dialect" value="org.hibernate.dialect.MySQLDialect" />

         <!-- Create/update tables automatically using mapping metadata -->
         <property name="hibernate.hbm2ddl.auto" value="update" />

         <!-- Pretty print the SQL in the log file and console -->
         <property name="hibernate.format_sql" value="true" />
      </properties>

   </persistence-unit>
</persistence>
```

log4j.properties

```
# Direct to file
log4j.appender.file=org.apache.log4j.RollingFileAppender
log4j.appender.file.File=hello-world.log
log4j.appender.file.MaxFileSize=2MB
log4j.appender.file.MaxBackupIndex=1
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L - %m%n

# Direct to stdout
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L - %m%n

# Root logger option
log4j.rootLogger=OFF, stdout, file

#Log everything (this will also include the logging information configured by
"log4j.logger.org.hibernate.SQL=ALL" and
"log4j.logger.org.hibernate.type.descriptor.sql.BasicBinder=TRACE")
#log4j.logger.org.hibernate=ALL

# Show SQL statements
log4j.logger.org.hibernate.SQL=ALL

# Show the bind parameter values
log4j.logger.org.hibernate.type.descriptor.sql.BasicBinder=TRACE
```

User1Client.java [Optimistic Locking and Versioning]

```
package client;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.OptimisticLockException;
import javax.persistence.Persistence;

import entity.Guide;

public class User1Client {
   public static void main(String[] args) {

      EntityManagerFactory emf = Persistence.createEntityManagerFactory("hello-world");
      EntityManager em1 = emf.createEntityManager();
      em1.getTransaction().begin();

      Guide guide = em1.find(Guide.class, 2L);

      em1.getTransaction().commit();
      em1.close();

      guide.setSalary(3000);

      EntityManager em2 = emf.createEntityManager();
      EntityTransaction txn2 = em2.getTransaction();
      try {
         txn2.begin();

         Guide mergedGuide = em2.merge(guide);

         txn2.commit();
      } catch (OptimisticLockException ole) {
         if(txn2 != null) {
            txn2.rollback();
            System.err.println("The guide was updated by some other user while you were
doing interesting things.");
         }
         ole.printStackTrace();
      } finally {
         if(em2 != null) { em2.close(); }
      }
   }
}
```

User2Client.java [Optimistic Locking and Versioning]

```
package client;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.OptimisticLockException;
import javax.persistence.Persistence;

import entity.Guide;

public class User2Client {
   public static void main(String[] args) {

      EntityManagerFactory emf = Persistence.createEntityManagerFactory("hello-world");
      EntityManager em1 = emf.createEntityManager();
      em1.getTransaction().begin();

      Guide guide = em1.find(Guide.class, 2L);

      em1.getTransaction().commit();
      em1.close();

      guide.setSalary(4000);

      EntityManager em2 = emf.createEntityManager();
      em2.getTransaction().begin();

      Guide mergedGuide = em2.merge(guide);

      em2.getTransaction().commit();
      em2.close();
   }
}
```

Guide.java

```
package entity;

import java.util.HashSet;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.Version;

@Entity
public class Guide {

   @Id
   @GeneratedValue(strategy = GenerationType.AUTO)
   private Long id;

   @Column(name = "staff_id", nullable = false)
   private String staffId;

   private String name;
   private Integer salary;

   @Version
   private Integer version;

   @OneToMany(mappedBy = "guide", cascade={CascadeType.PERSIST,
CascadeType.MERGE})
   private Set<Student> students = new HashSet<Student>();

   public Guide() { }
   public Guide(String staffId, String name, Integer salary) {
      this.staffId = staffId;
      this.name = name;
      this.salary = salary;
   }
   public void addStudent(Student student) {
      students.add(student);
      student.setGuide(this);
   }
```

```
    public void setSalary(Integer salary) {
        this.salary = salary;
    }
    public Set<Student> getStudents() {
        return students;
    }

    public String toString() {
        return "Guide [id=" + id + ", staffId=" + staffId + ", name=" + name
                + ", salary=" + salary + ", students=" + students + "]";
    }

}
```

Student.java

```
package entity;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;

import org.apache.commons.lang3.builder.EqualsBuilder;
import org.apache.commons.lang3.builder.HashCodeBuilder;

@Entity
public class Student {

   @Id @GeneratedValue(strategy=GenerationType.AUTO)
   private Long id;

   @Column(name="enrollment_id", nullable=false)
   private String enrollmentId;

   private String name;

   @ManyToOne(fetch=FetchType.LAZY)
   @JoinColumn(name="guide_id")
   private Guide guide;

   public Student() {}
   public Student(String enrollmentId, String name) {
      this.enrollmentId = enrollmentId;    this.name = name;
   }

   public void setGuide(Guide guide) {
      this.guide = guide;
   }
   public void setName(String name) {
      this.name = name;
   }
   public Long getId() {
      return id;
   }
```

```
    @Override
    public int hashCode() {
        return new HashCodeBuilder().append(enrollmentId).toHashCode();
    }

    @Override
    public boolean equals(Object obj) {
        if(!(obj instanceof Student)) return false;
        Student other = (Student) obj;
        return new EqualsBuilder().append(enrollmentId, other.enrollmentId).isEquals();
    }

    @Override
    public String toString() {
        return "Student [id=" + id + ", enrollmentId=" + enrollmentId
                + ", name=" + name + ", guide=" + guide + "]";
    }

}
```