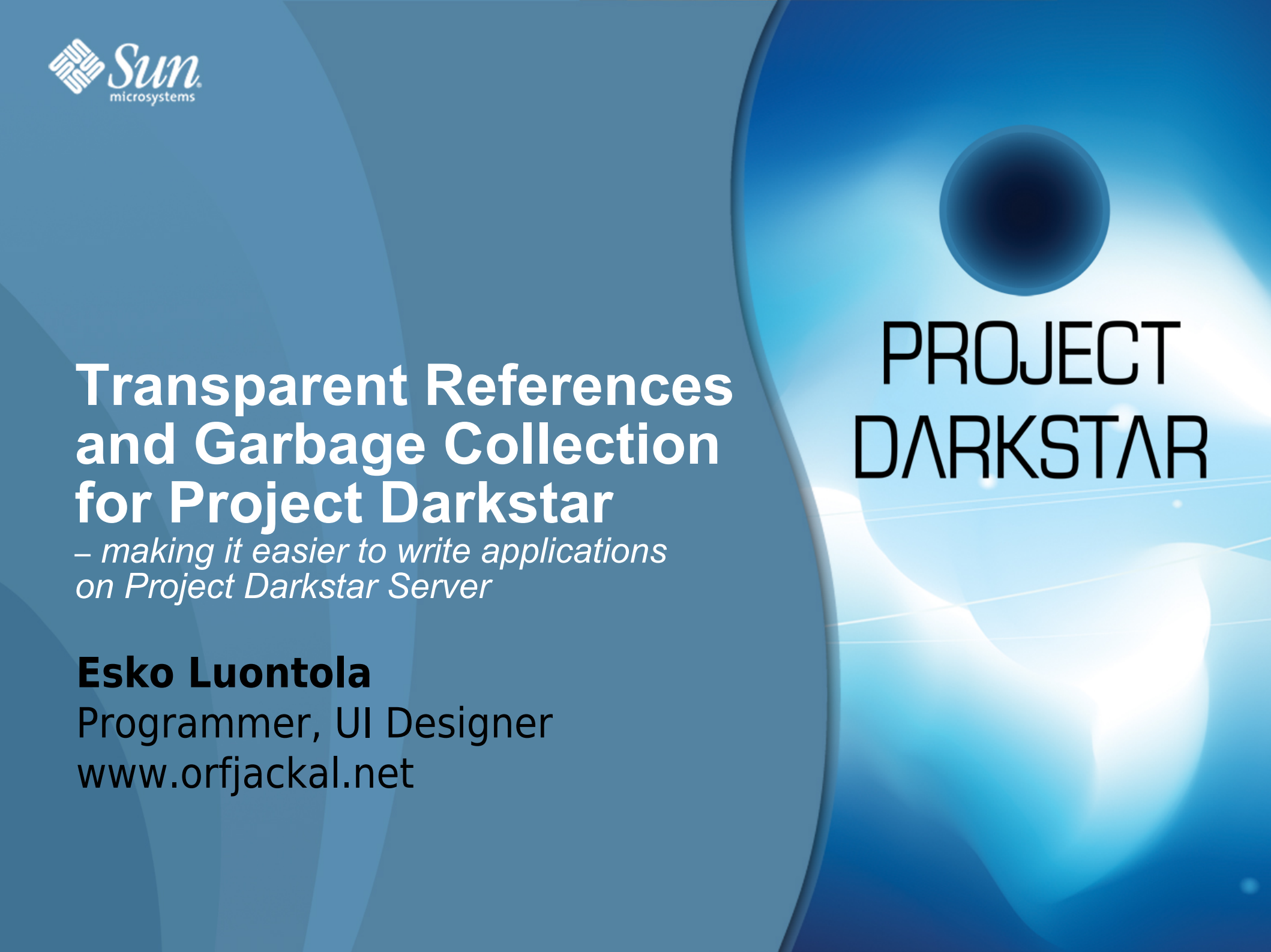


Transparent References and Garbage Collection for Project Darkstar

– making it easier to write applications on Project Darkstar Server

Esko Luontola
Programmer, UI Designer
www.orfjackal.net

Agenda

- How Darkstar's persistence model is currently?
- How transparent references and garbage collection change the persistence model?
- Future directions

Agenda

- **How Darkstar's persistence model is currently?**
- How transparent references and garbage collection change the persistence model?
- Future directions

```java
public class Person implements ManagedObject, Serializable {
    private String name;
    private Address address;
    private ManagedReference<Person> roommate;

    public Person(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public Address getAddress() {
        return address;
    }

    public void setAddress(Address address) {
        this.address = address;
    }

    public Person getRoommate() {
        if (roommate == null) {
            return null;
        } else {
            return roommate.get();
        }
    }

    public void setRoommate(Person roommate) {
        if (roommate == null) {
            this.roommate = null;
        } else {
            this.roommate = AppContext.getDataManager().createReference(roommate);
        }
    }
}
```

```java
public class Address implements Serializable {
    private String city;
    private String street;

    public Address(String city, String street) {
        this.city = city;
        this.street = street;
    }

    public String getCity() {
        return city;
    }

    public String getStreet() {
        return street;
    }
}
```

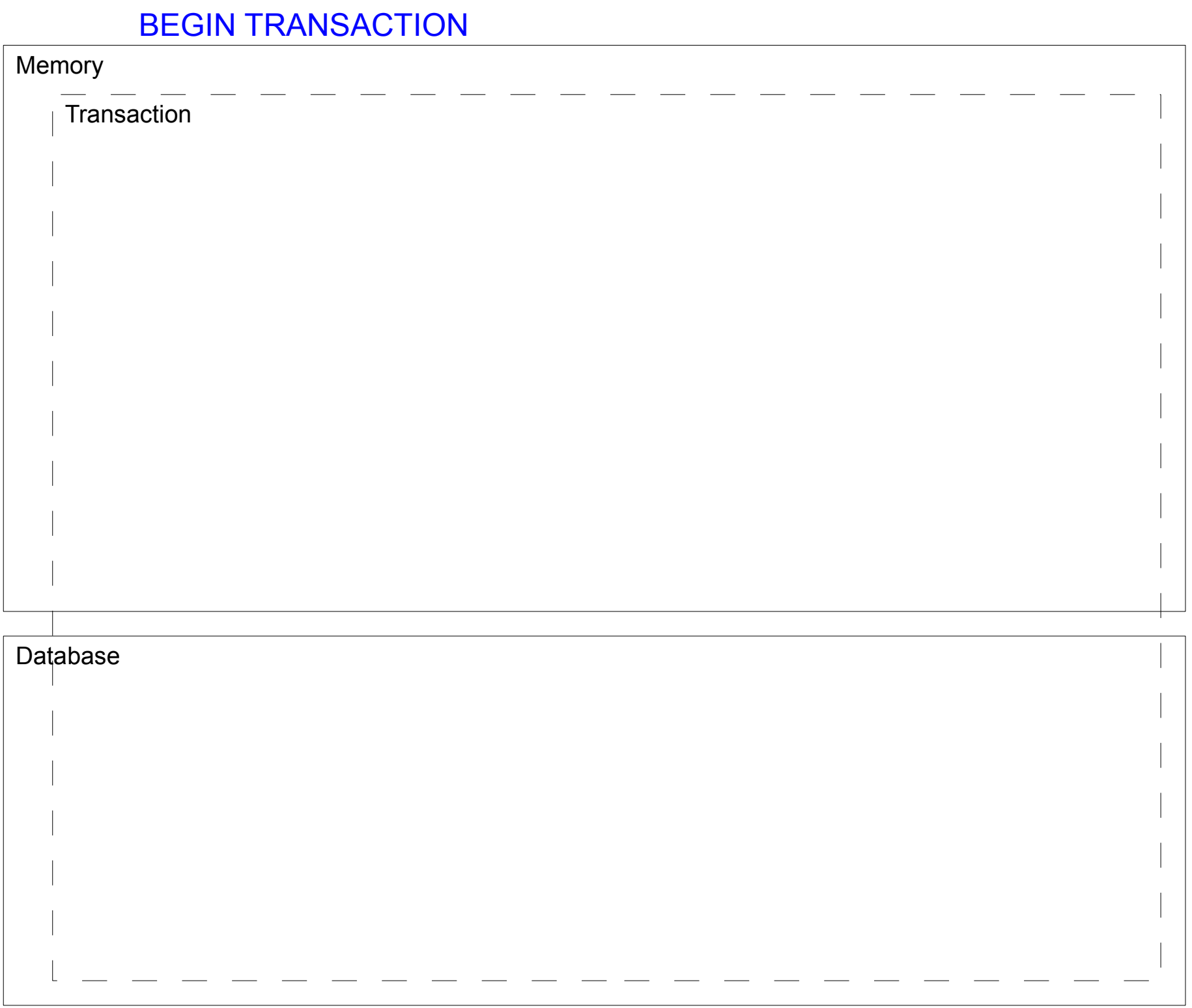
Darkstar requires special handling for ManagedObjects

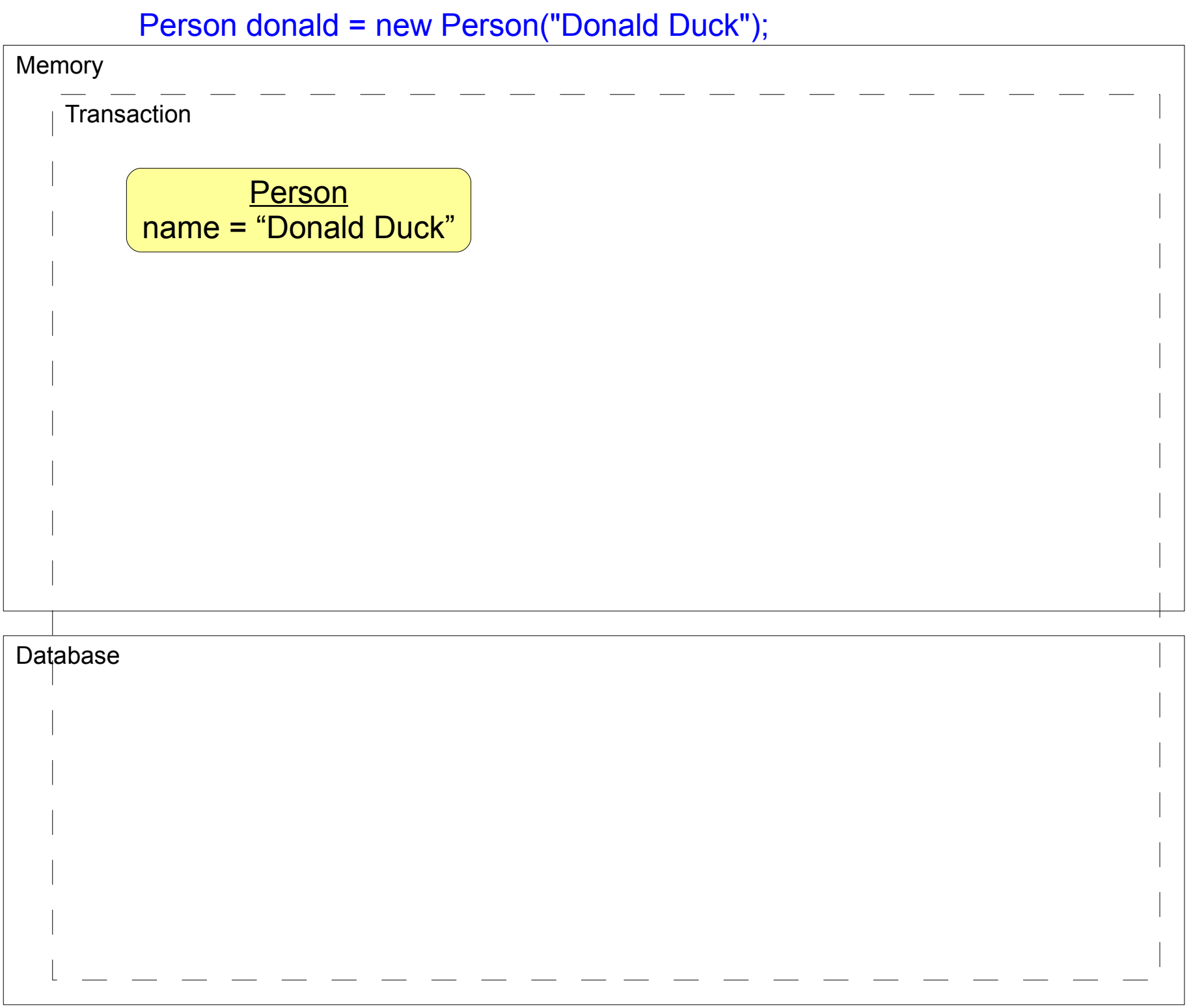
What happens under the hood when the following is executed?

```
Person donald = new Person("Donald Duck");
donald.setAddress(new Address("Duckburg", "Duck Street 313"));

Person mickey = new Person("Mickey Mouse");
mickey.setAddress(donald.getAddress());
mickey.setRoommate(donald);
donald.setRoommate(mickey);

AppContext.getDataManager().setBinding("donald", donald);
```

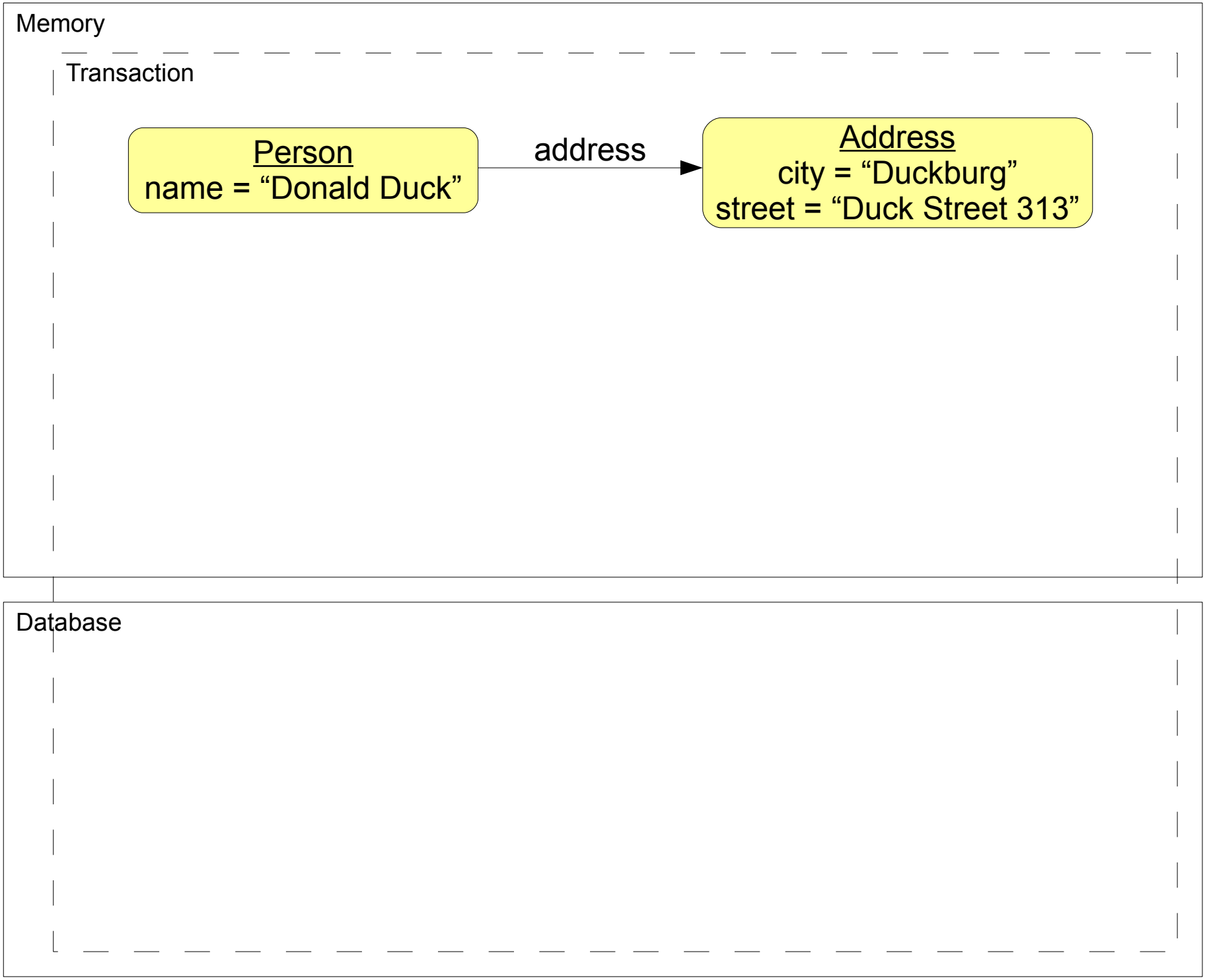
Memory

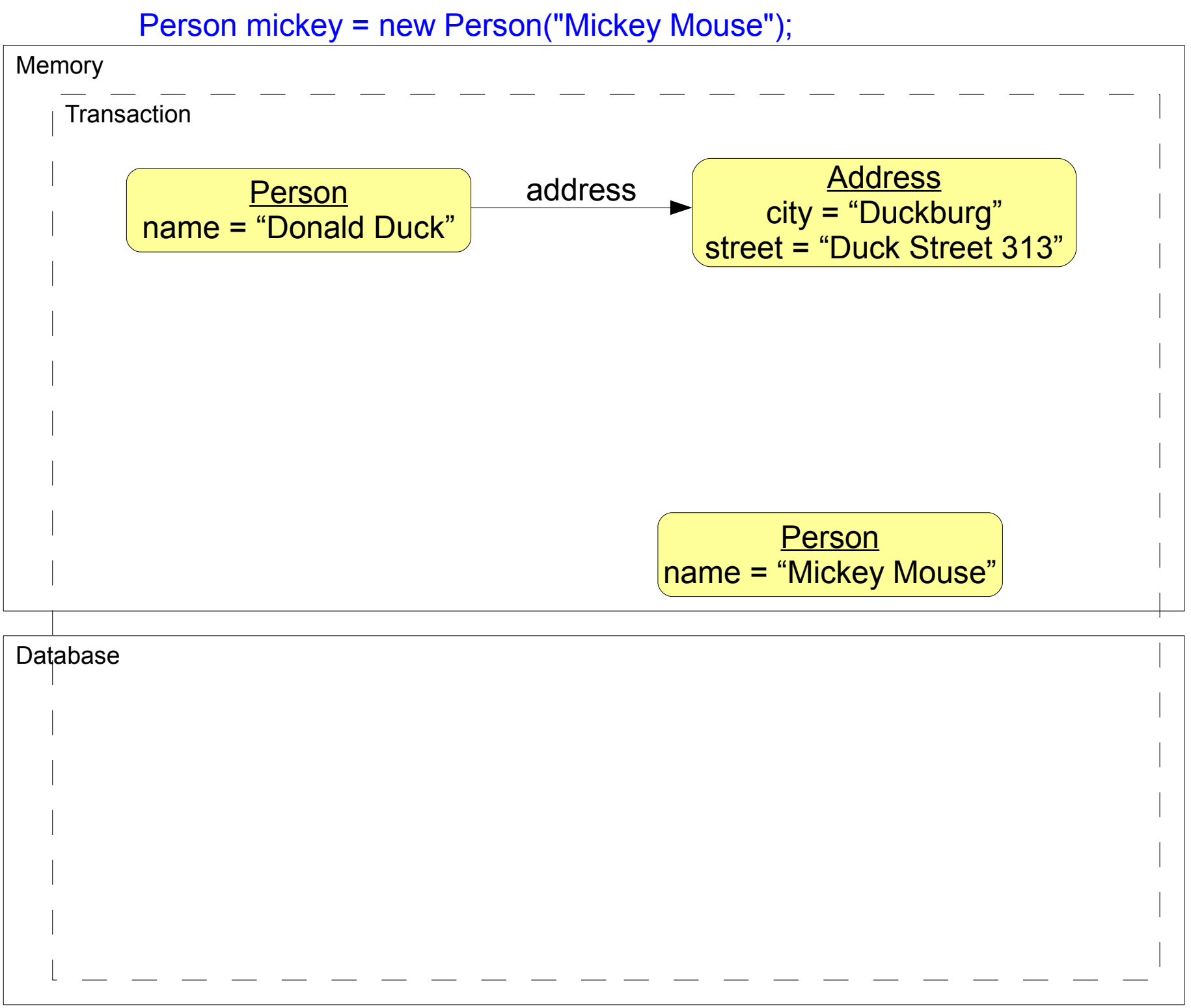
Database


BEGIN TRANSACTION
Memory
Transaction
Database

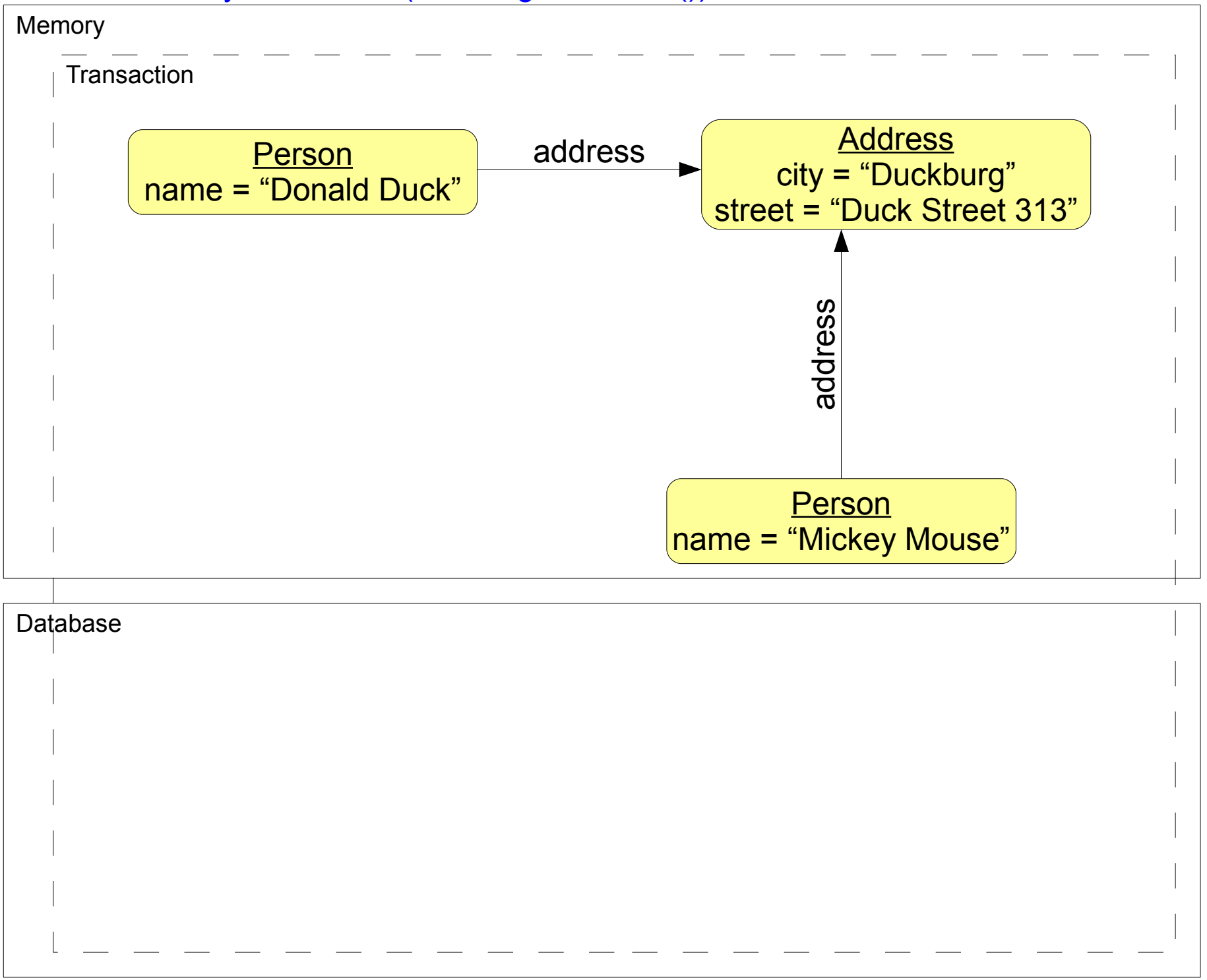

Person donald = new Person("Donald Duck");
Memory
Transaction
Person
name = “Donald Duck”
Database

donald.setAddress(new Address("Duckburg", "Duck Street 313"));



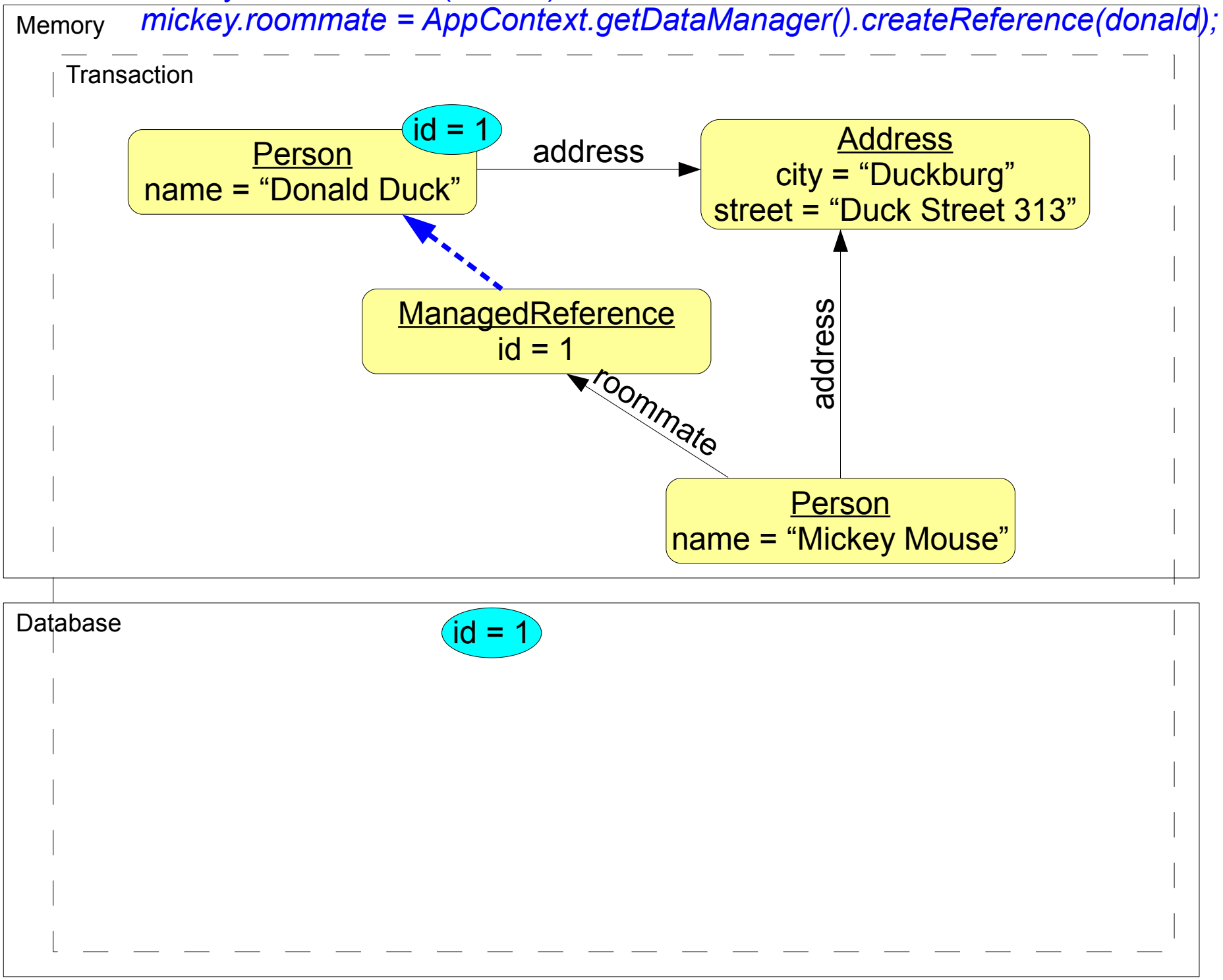

Person mickey = new Person("Mickey Mouse");
Memory
Transaction
Person
name = “Donald Duck”
address
Address
city = “Duckburg”
street = “Duck Street 313”
Person
name = “Mickey Mouse”
Database

```
mickey.setAddress(donald.getAddress());
```



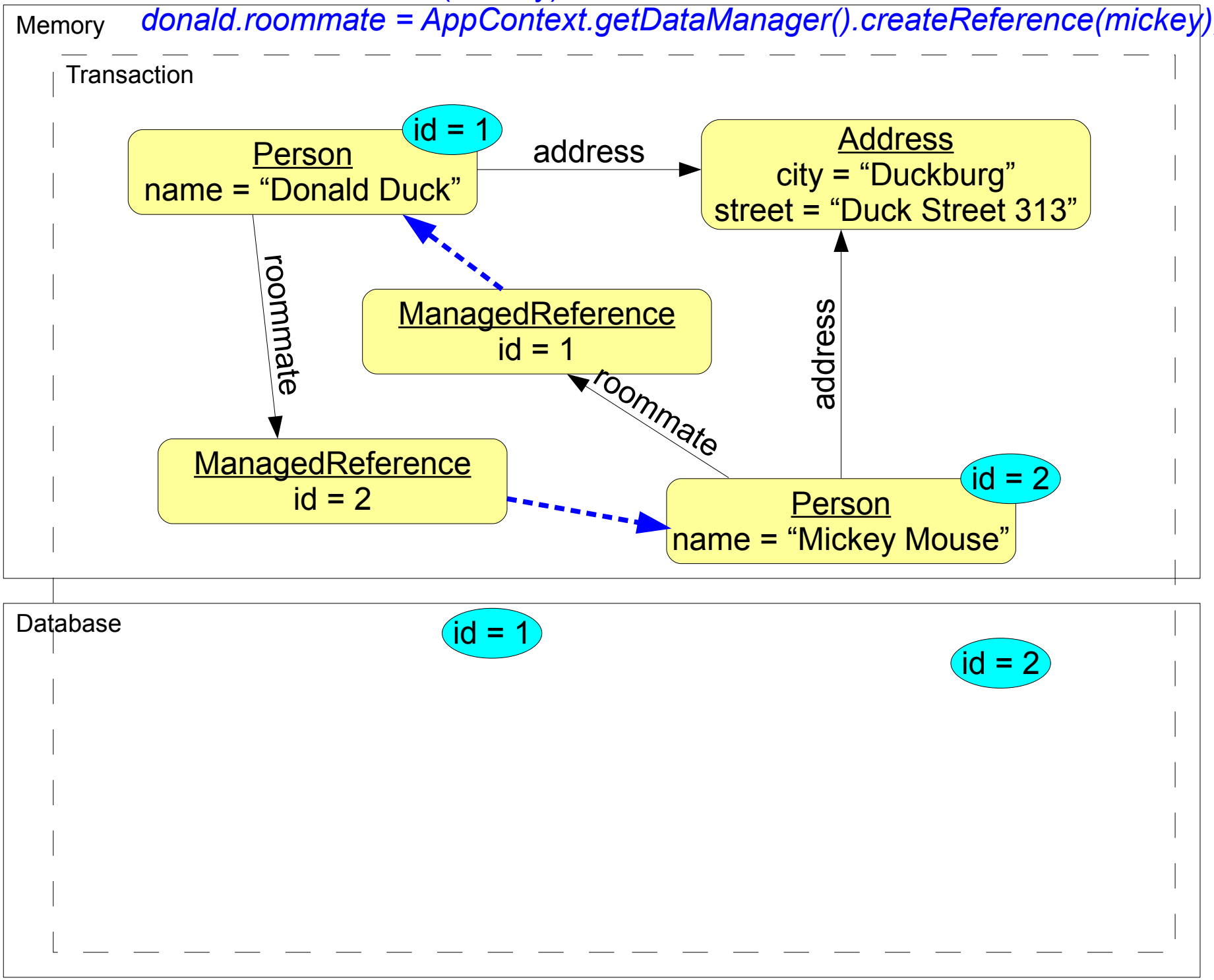
mickey.setRoommate(donald);

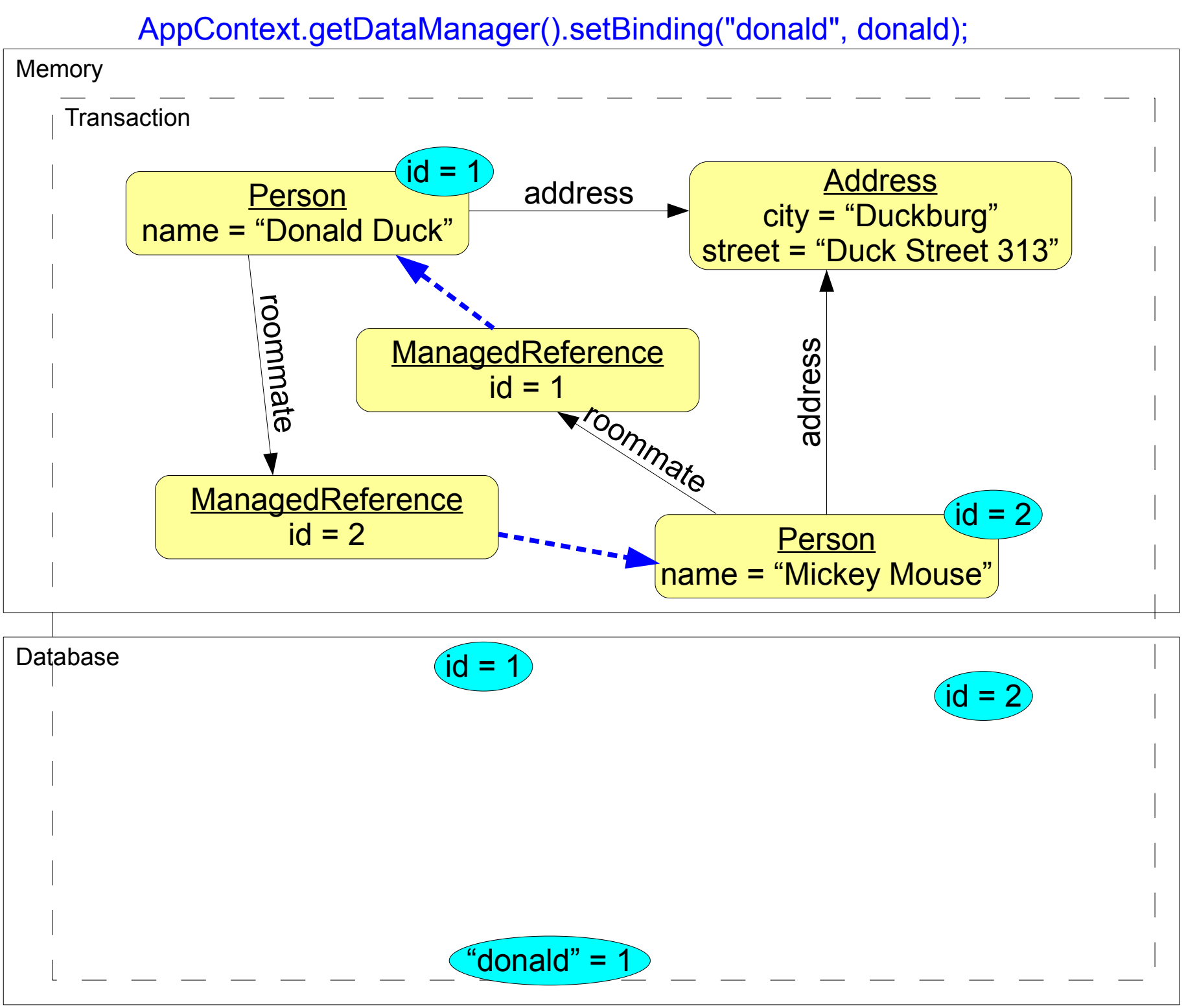
mickey.roommate = AppContext.getDataManager().createReference(donald);

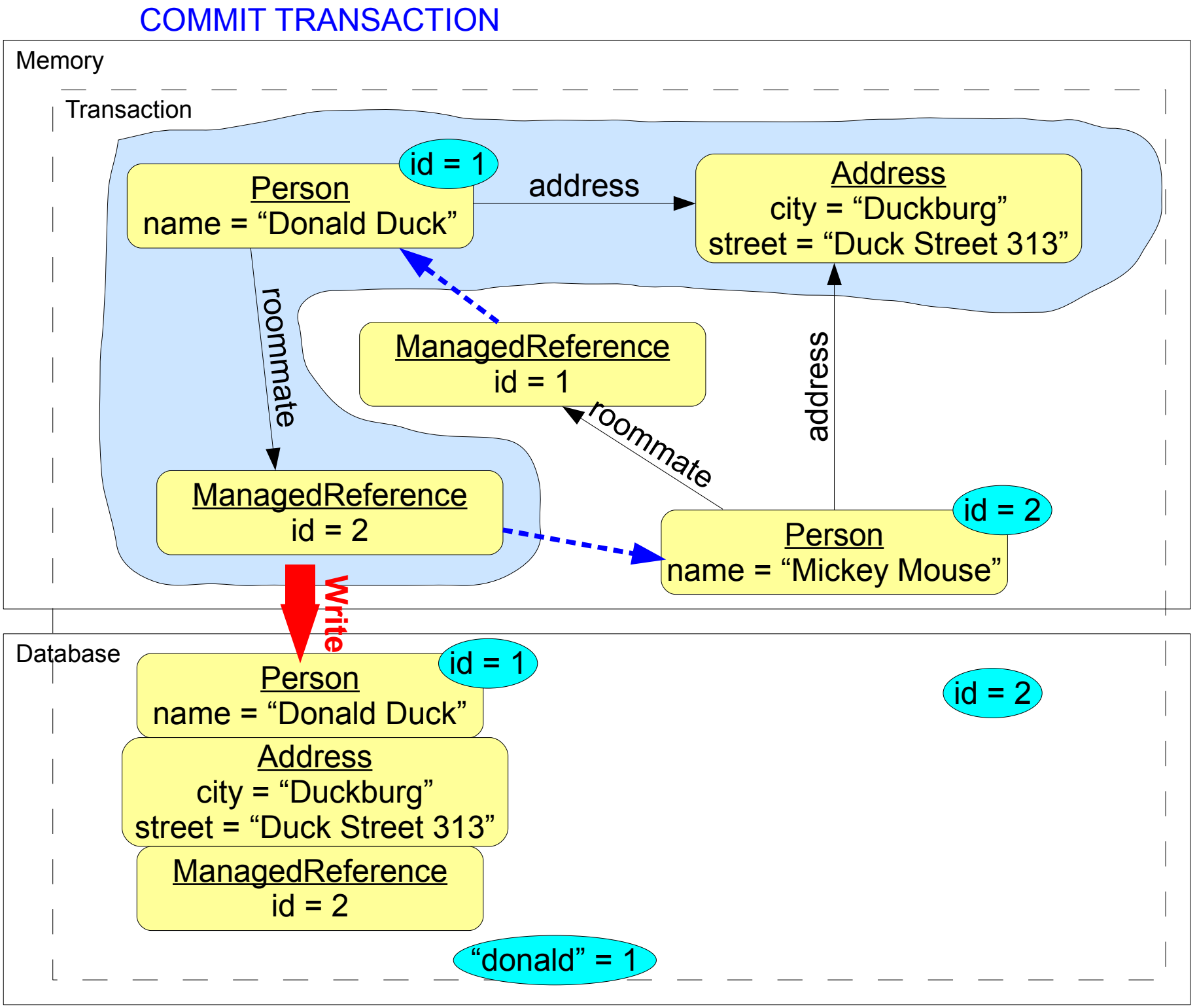


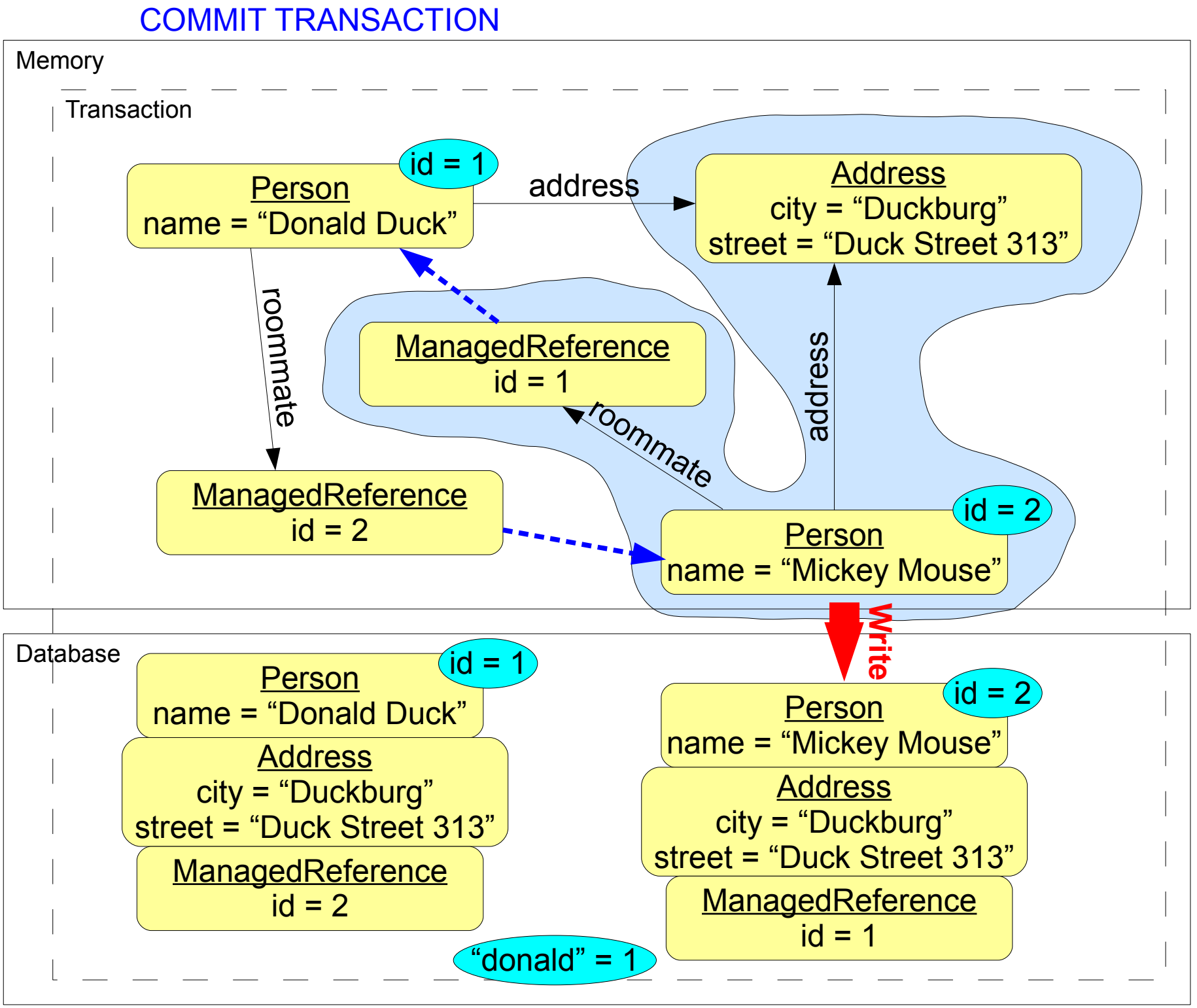
donald.setRoommate(mickey);

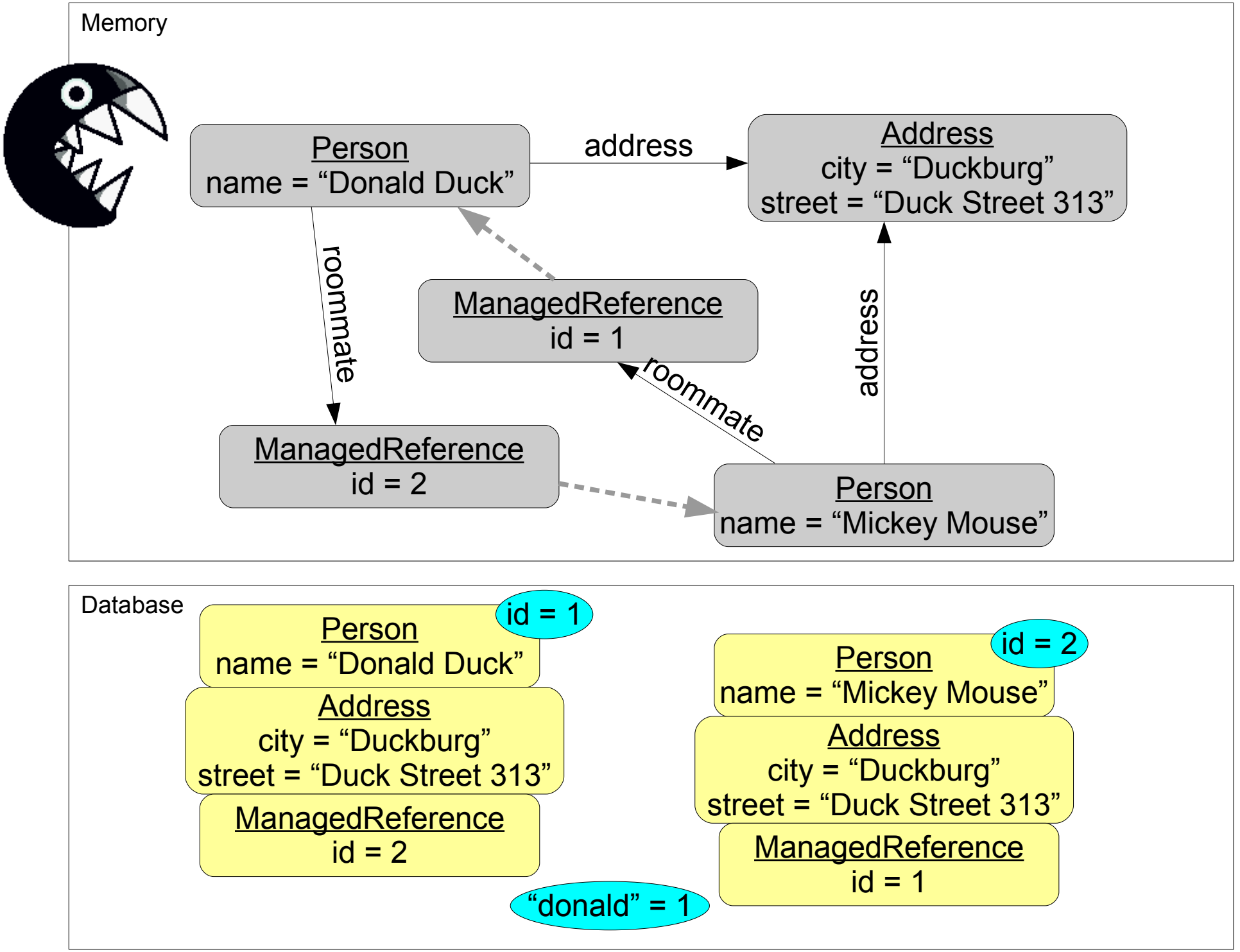
donald.roommate = AppContext.getDataManager().createReference(mickey);



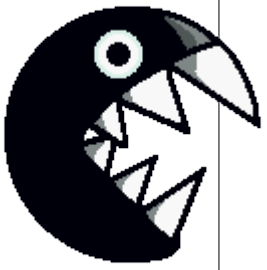

AppContext.getDataManager().setBinding("donald", donald);
Memory
Transaction
id = 1
Person
name = “Donald Duck”
address
Address
city = “Duckburg”
street = “Duck Street 313”
roommate
ManagedReference
id = 1
address
roommate
ManagedReference
id = 2
id = 2
Person
name = “Mickey Mouse”
Database
id = 1
id = 2
“donald” = 1


COMMIT TRANSACTION
Memory
Transaction
id = 1
Person
name = “Donald Duck”
address
Address
city = “Duckburg”
street = “Duck Street 313”
roommate
ManagedReference
id = 1
address
roommate
ManagedReference
id = 2
id = 2
Person
name = “Mickey Mouse”
Write
Database
id = 1
Person
name = “Donald Duck”
id = 2
Address
city = “Duckburg”
street = “Duck Street 313”
ManagedReference
id = 2
“donald” = 1


COMMIT TRANSACTION
Memory
Transaction
Person
name = “Donald Duck”
id = 1
address
Address
city = “Duckburg”
street = “Duck Street 313”
roommate
ManagedReference
id = 1
roommate
address
ManagedReference
id = 2
Person
name = “Mickey Mouse”
id = 2
Write
Database
Person
name = “Donald Duck”
id = 1
Address
city = “Duckburg”
street = “Duck Street 313”
ManagedReference
id = 2
Person
name = “Mickey Mouse”
id = 2
Address
city = “Duckburg”
street = “Duck Street 313”
ManagedReference
id = 1
“donald” = 1


Memory
Person
name = “Donald Duck”
address
Address
city = “Duckburg”
street = “Duck Street 313”
roommate
ManagedReference
id = 1
address
roommate
ManagedReference
id = 2
Person
name = “Mickey Mouse”
Database
id = 1
Person
name = “Donald Duck”
Address
city = “Duckburg”
street = “Duck Street 313”
ManagedReference
id = 2
id = 2
Person
name = “Mickey Mouse”
Address
city = “Duckburg”
street = “Duck Street 313”
ManagedReference
id = 1
“donald” = 1

Memory



Database

Person
name = “Donald Duck”

id = 1

Address
city = “Duckburg”
street = “Duck Street 313”

ManagedReference
id = 2

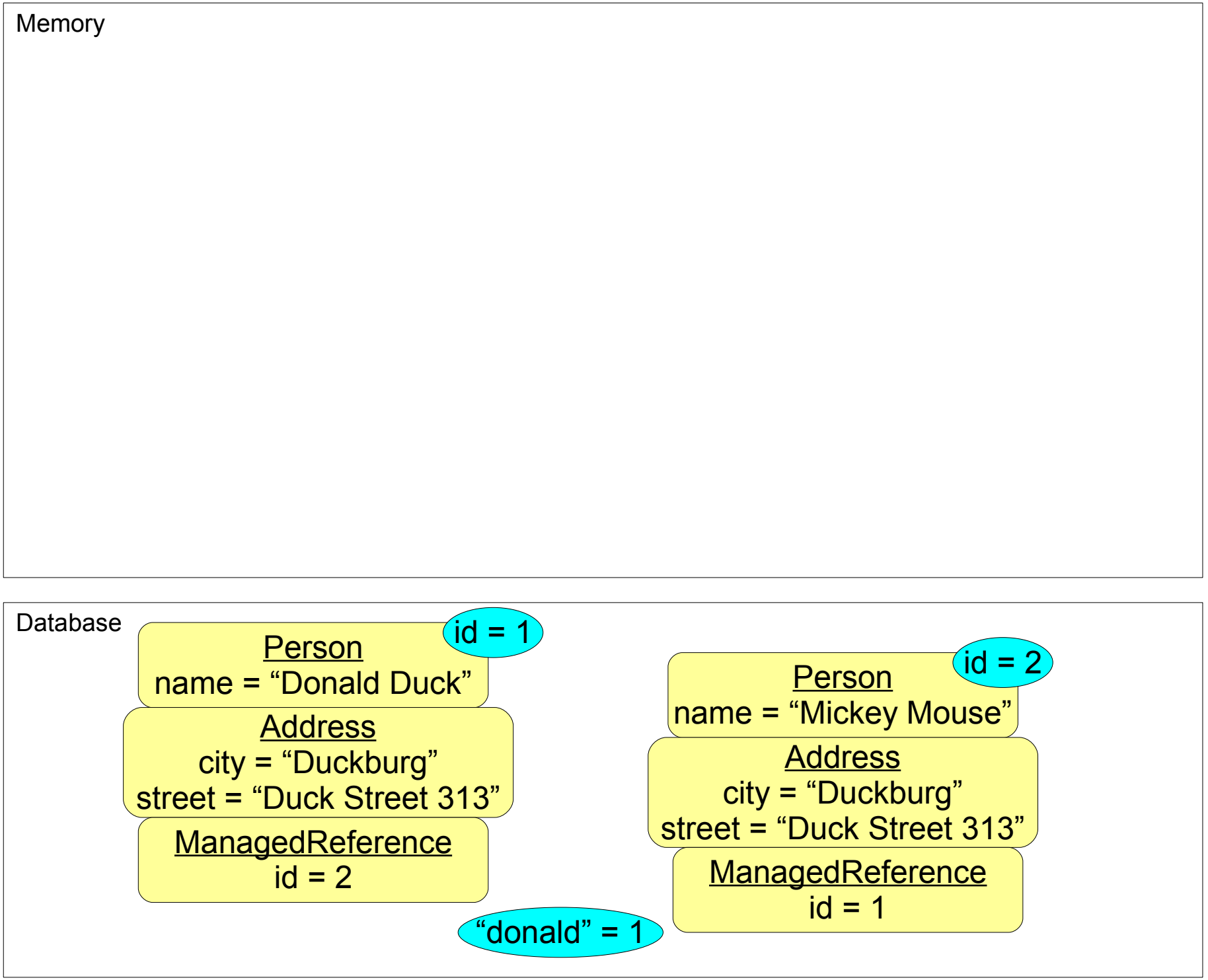
“donald” = 1

Person
name = “Mickey Mouse”

id = 2

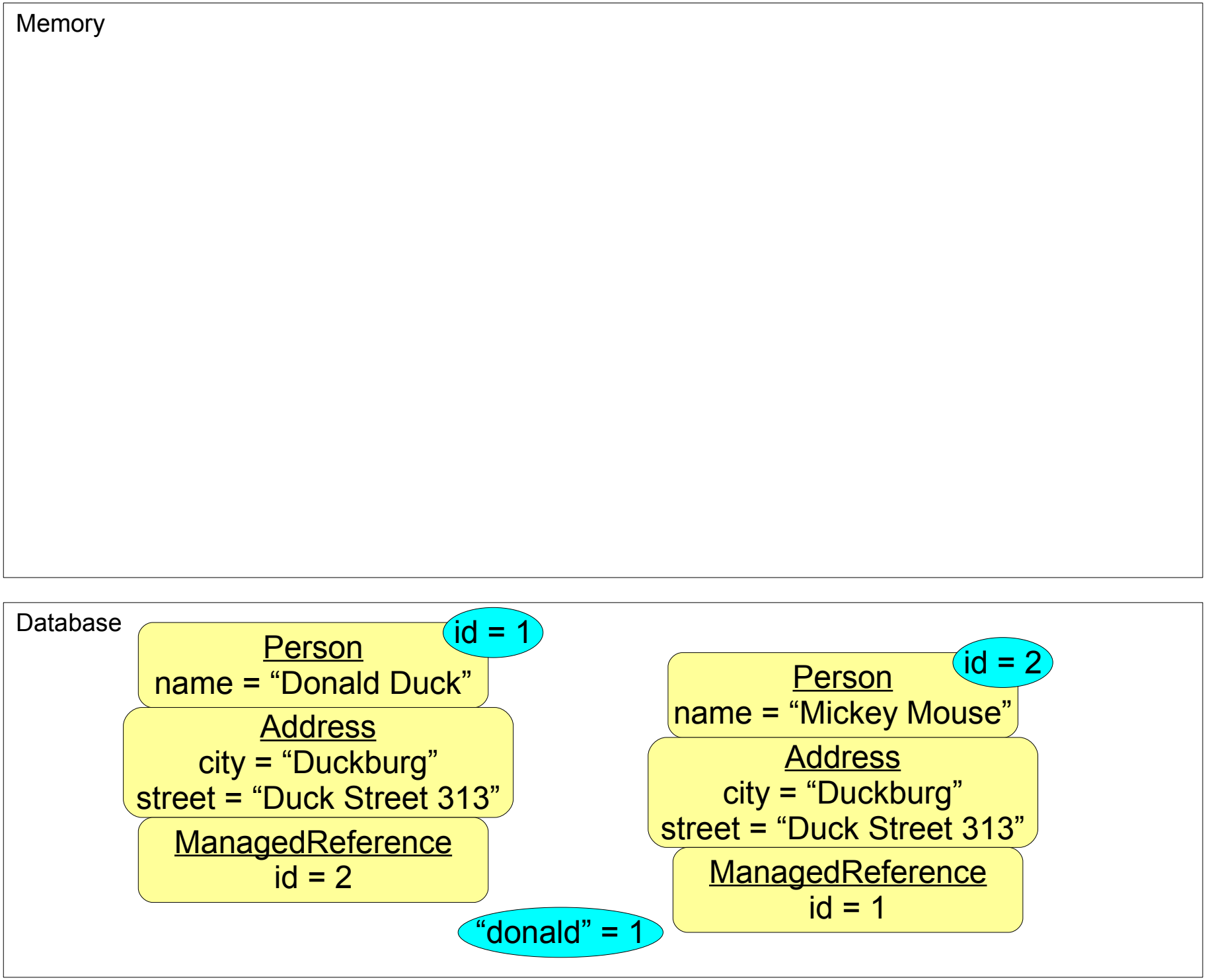
Address
city = “Duckburg”
street = “Duck Street 313”

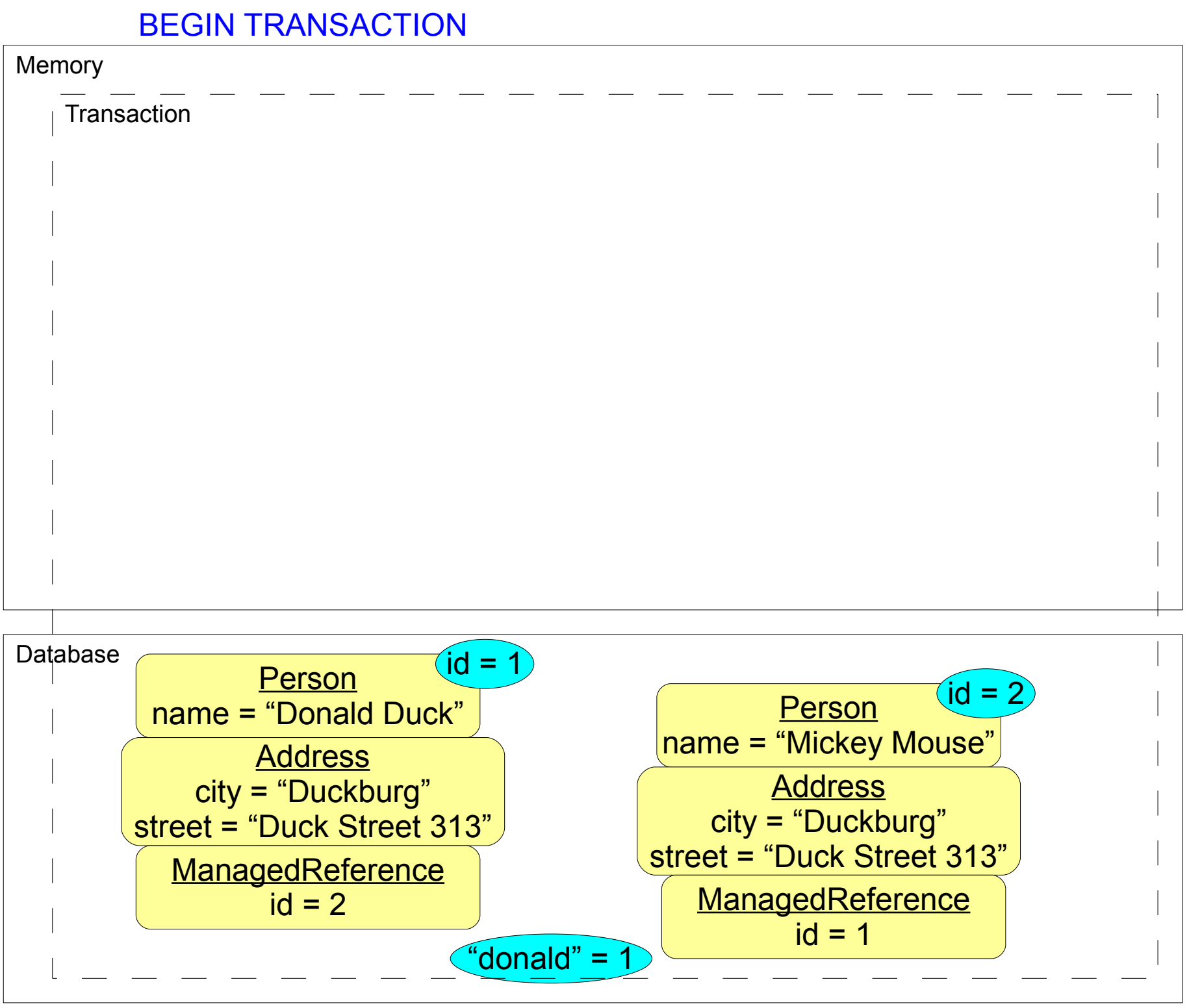
ManagedReference
id = 1

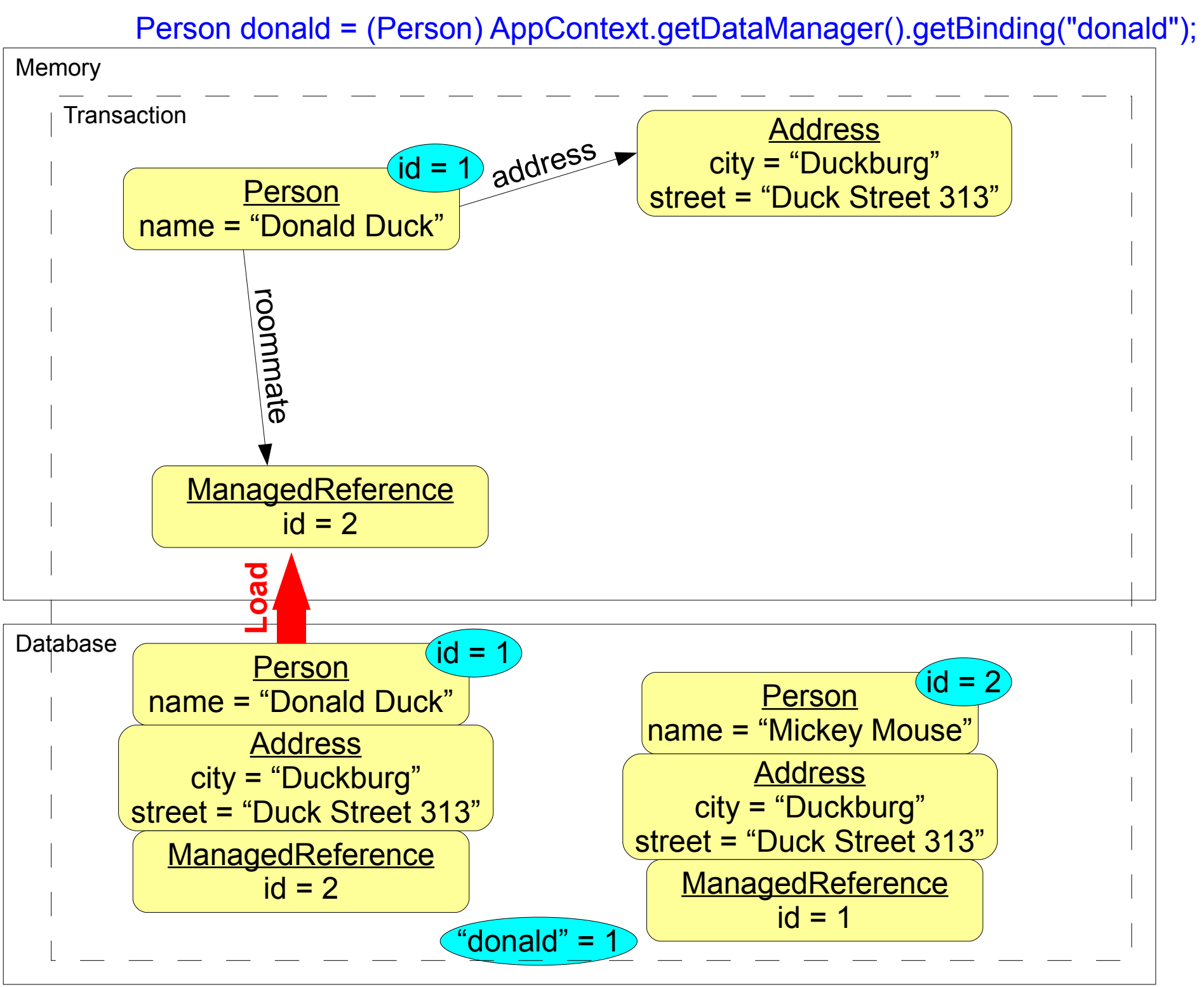

Memory
Database
Person
name = “Donald Duck”
id = 1
Address
city = “Duckburg”
street = “Duck Street 313”
ManagedReference
id = 2
“donald” = 1
Person
name = “Mickey Mouse”
id = 2
Address
city = “Duckburg”
street = “Duck Street 313”
ManagedReference
id = 1

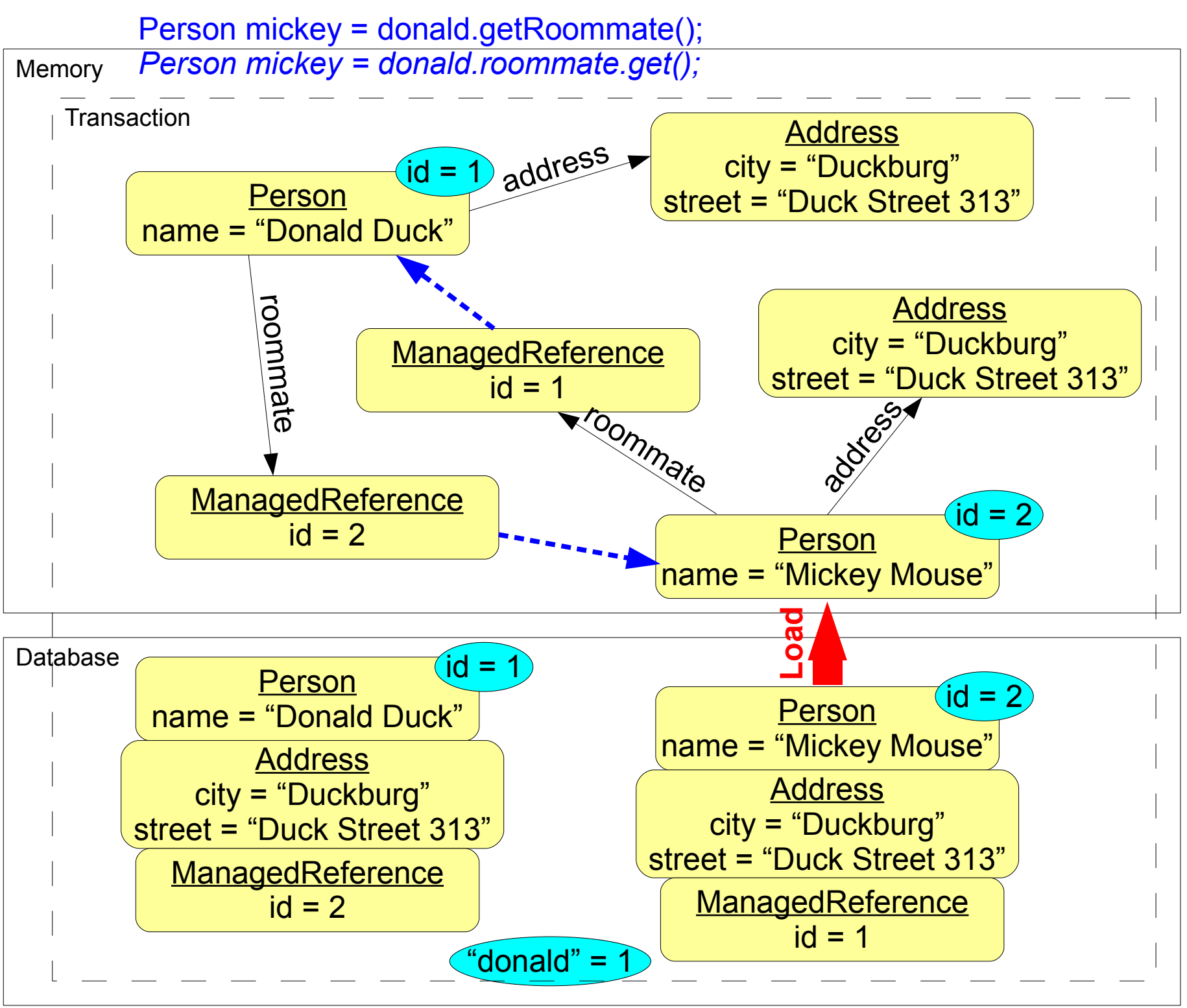
What happens under the hood when the following is executed?

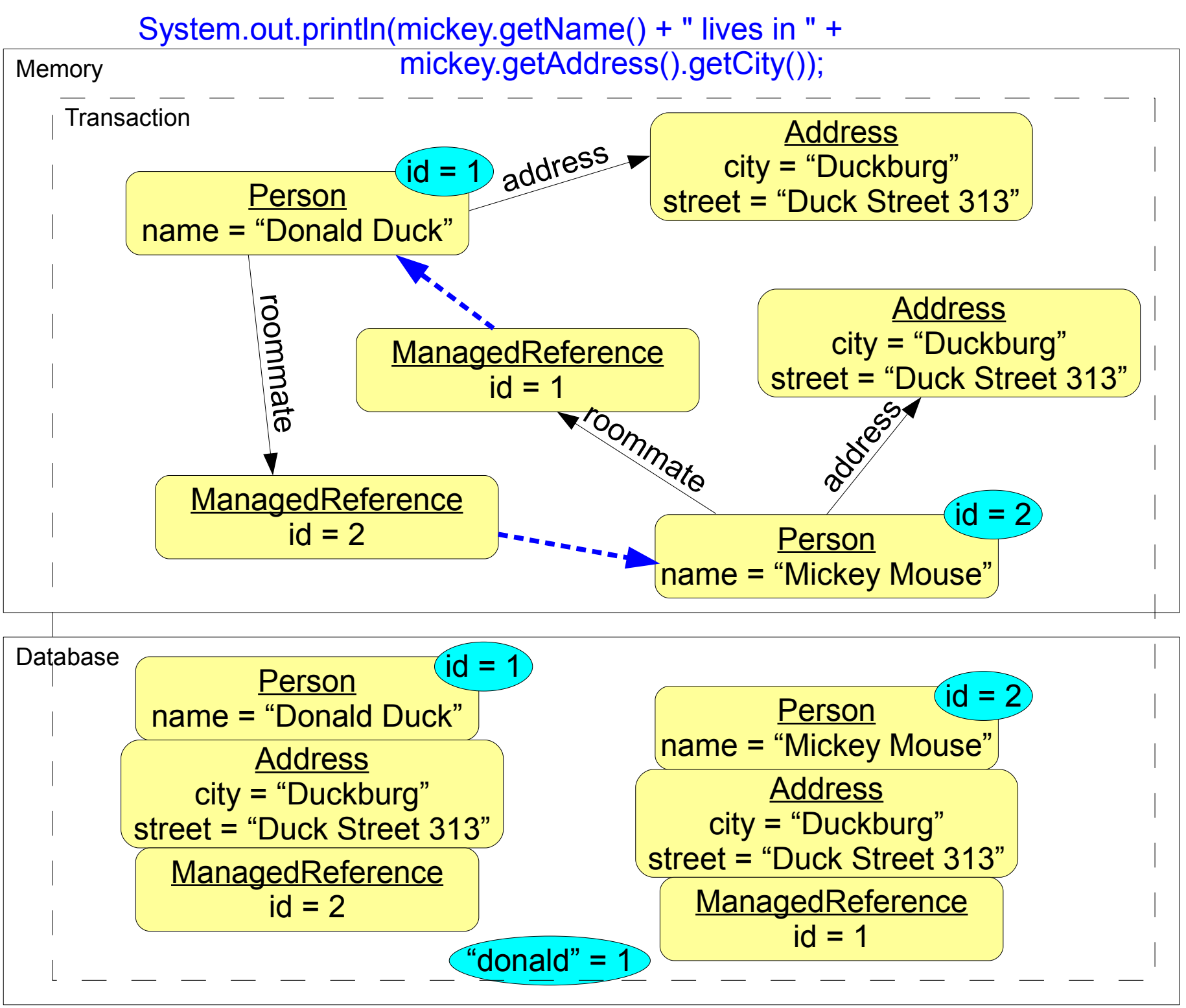
```
Person donald = (Person) AppContext.getDataManager().getBinding("donald");

Person mickey = donald.getRoommate();
System.out.println(mickey.getName() + " lives in " +
                   mickey.getAddress().getCity());

donald.setRoommate(null);
AppContext.getDataManager().removeObject(mickey);
```


Memory
Database
Person
name = “Donald Duck”
id = 1
Address
city = “Duckburg”
street = “Duck Street 313”
ManagedReference
id = 2
“donald” = 1
Person
name = “Mickey Mouse”
id = 2
Address
city = “Duckburg”
street = “Duck Street 313”
ManagedReference
id = 1

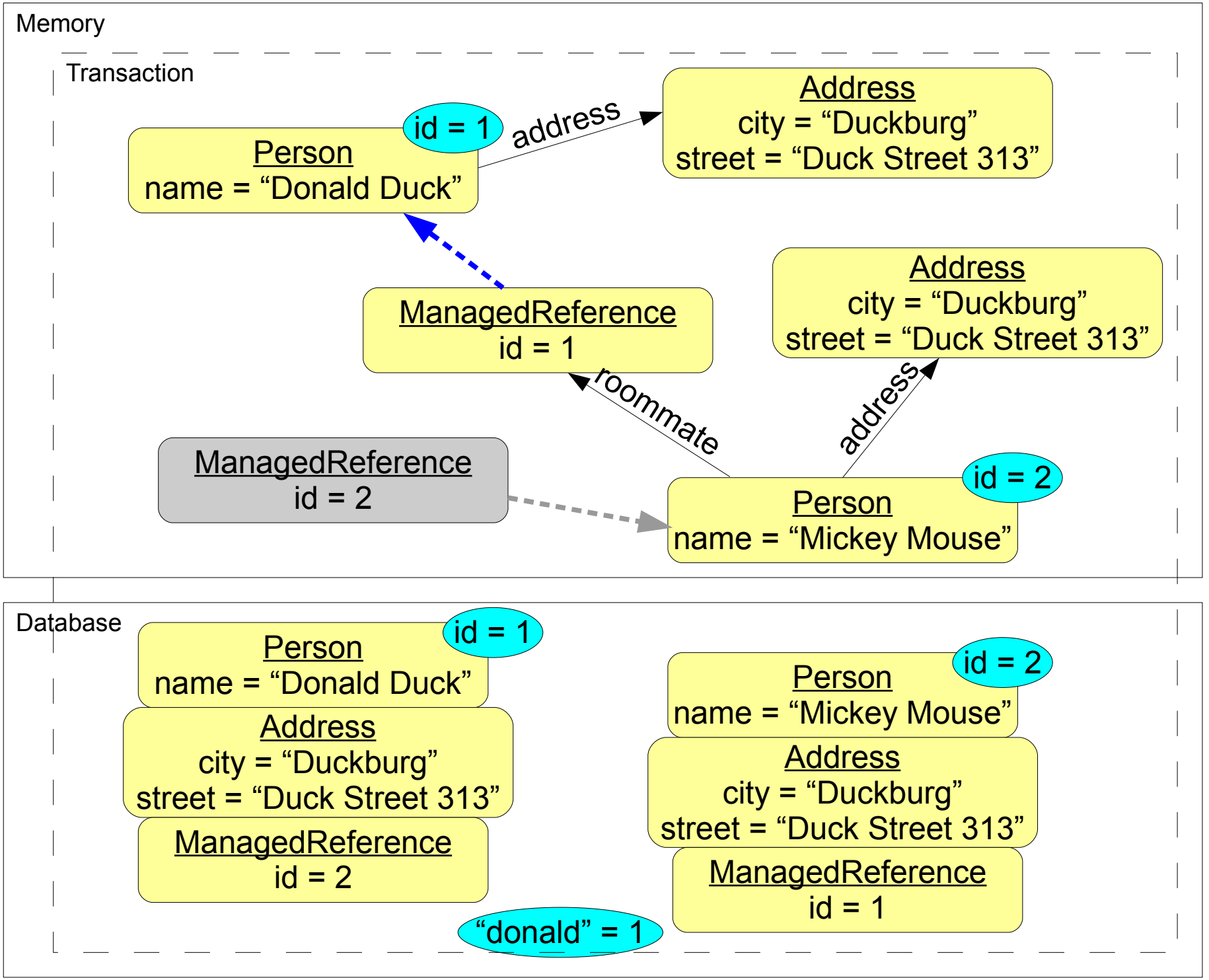

BEGIN TRANSACTION
Memory
Transaction
Database
Person
name = “Donald Duck”
id = 1
Address
city = “Duckburg”
street = “Duck Street 313”
ManagedReference
id = 2
Person
name = “Mickey Mouse”
id = 2
Address
city = “Duckburg”
street = “Duck Street 313”
ManagedReference
id = 1
“donald” = 1

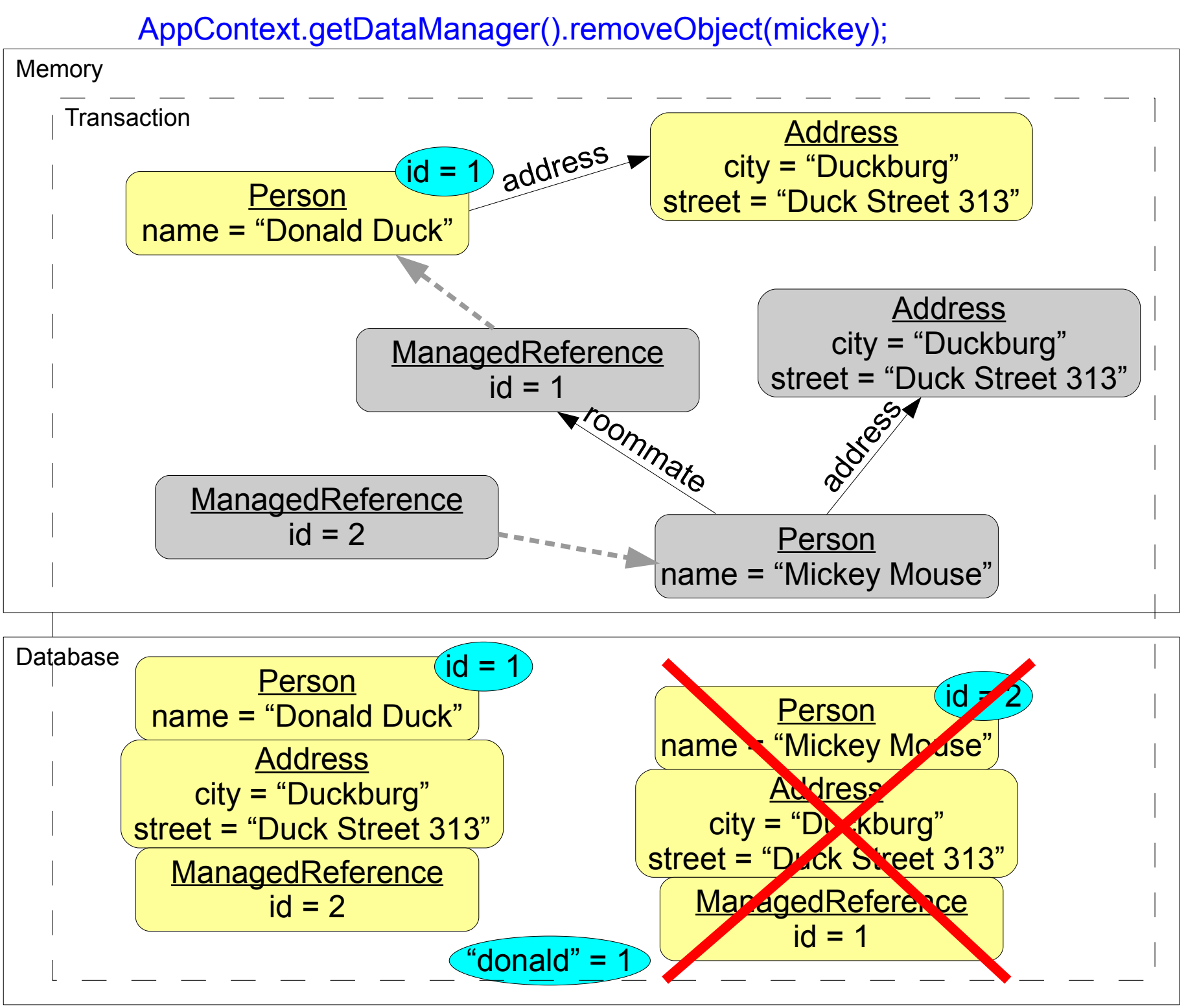

Person donald = (Person) AppContext.getDataManager().getBinding("donald");
Memory
Transaction
id = 1
Person
name = “Donald Duck”
address
Address
city = “Duckburg”
street = “Duck Street 313”
roommate
ManagedReference
id = 2
Load
Database
id = 1
Person
name = “Donald Duck”
Address
city = “Duckburg”
street = “Duck Street 313”
ManagedReference
id = 2
id = 2
Person
name = “Mickey Mouse”
Address
city = “Duckburg”
street = “Duck Street 313”
ManagedReference
id = 1
“donald” = 1

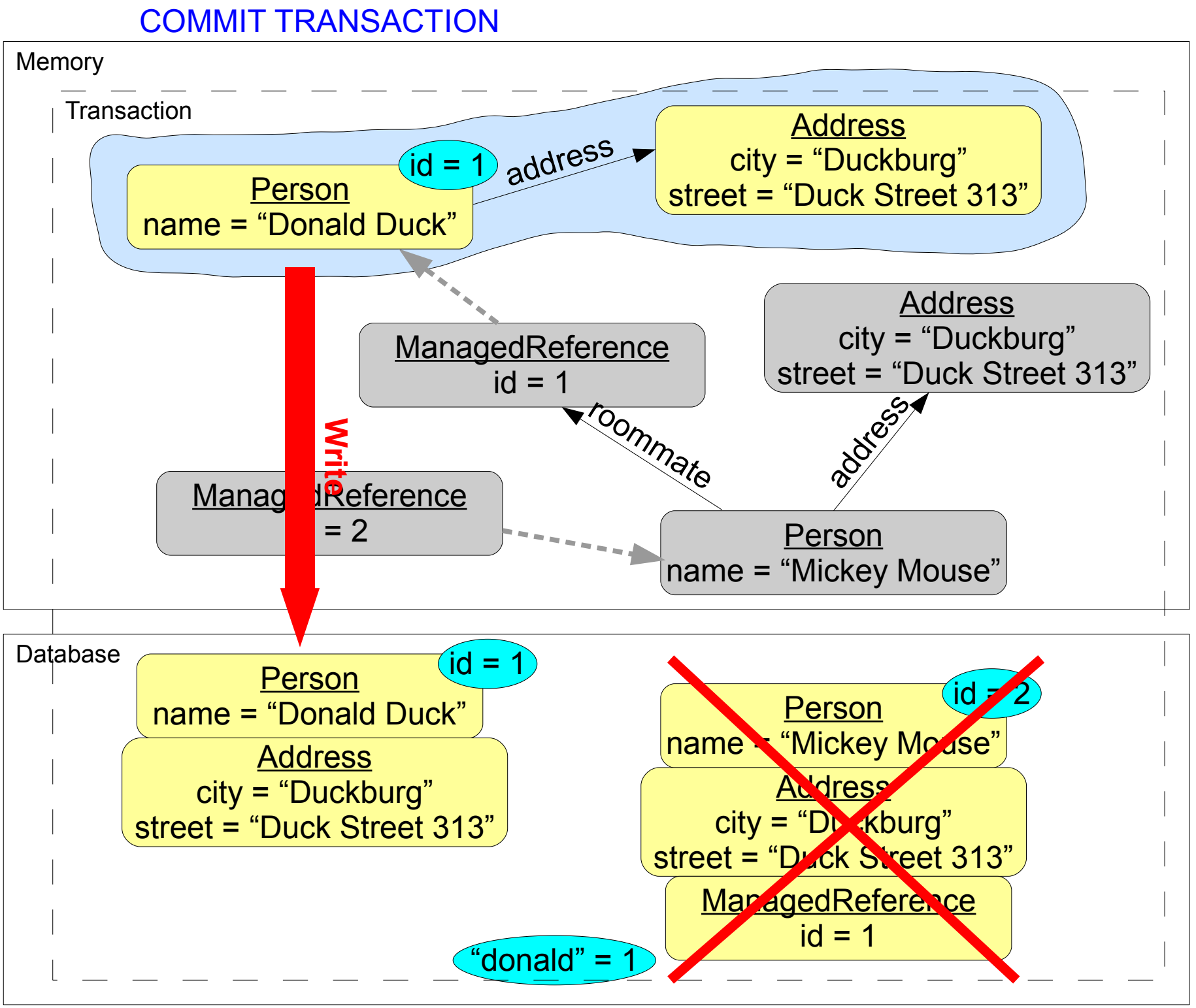

Person mickey = donald.getRoommate();
Person mickey = donald.roommate.get();
Memory
Transaction
Person
name = “Donald Duck”
id = 1
address
Address
city = “Duckburg”
street = “Duck Street 313”
roommate
ManagedReference
id = 1
Address
city = “Duckburg”
street = “Duck Street 313”
roommate
address
ManagedReference
id = 2
Person
name = “Mickey Mouse”
id = 2
Load
Database
Person
name = “Donald Duck”
id = 1
Address
city = “Duckburg”
street = “Duck Street 313”
ManagedReference
id = 2
Person
name = “Mickey Mouse”
id = 2
Address
city = “Duckburg”
street = “Duck Street 313”
ManagedReference
id = 1
“donald” = 1

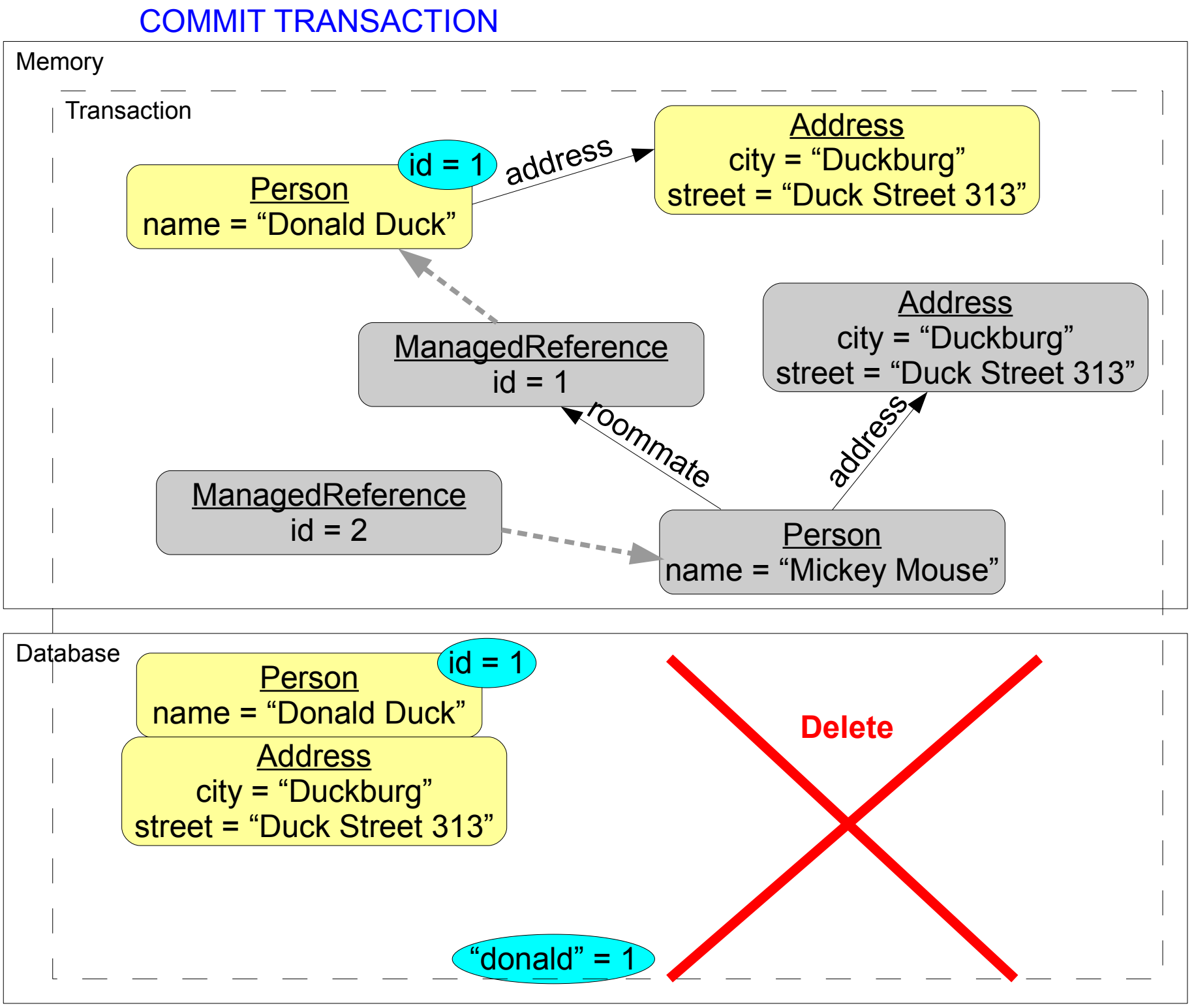

System.out.println(mickey.getName() + " lives in " +
mickey.getAddress().getCity());
Memory
Transaction
Person
name = “Donald Duck”
id = 1
address
Address
city = “Duckburg”
street = “Duck Street 313”
roommate
ManagedReference
id = 1
Address
city = “Duckburg”
street = “Duck Street 313”
roommate
address
ManagedReference
id = 2
Person
name = “Mickey Mouse”
id = 2
Database
Person
name = “Donald Duck”
id = 1
Address
city = “Duckburg”
street = “Duck Street 313”
ManagedReference
id = 2
Person
name = “Mickey Mouse”
id = 2
Address
city = “Duckburg”
street = “Duck Street 313”
ManagedReference
id = 1
“donald” = 1

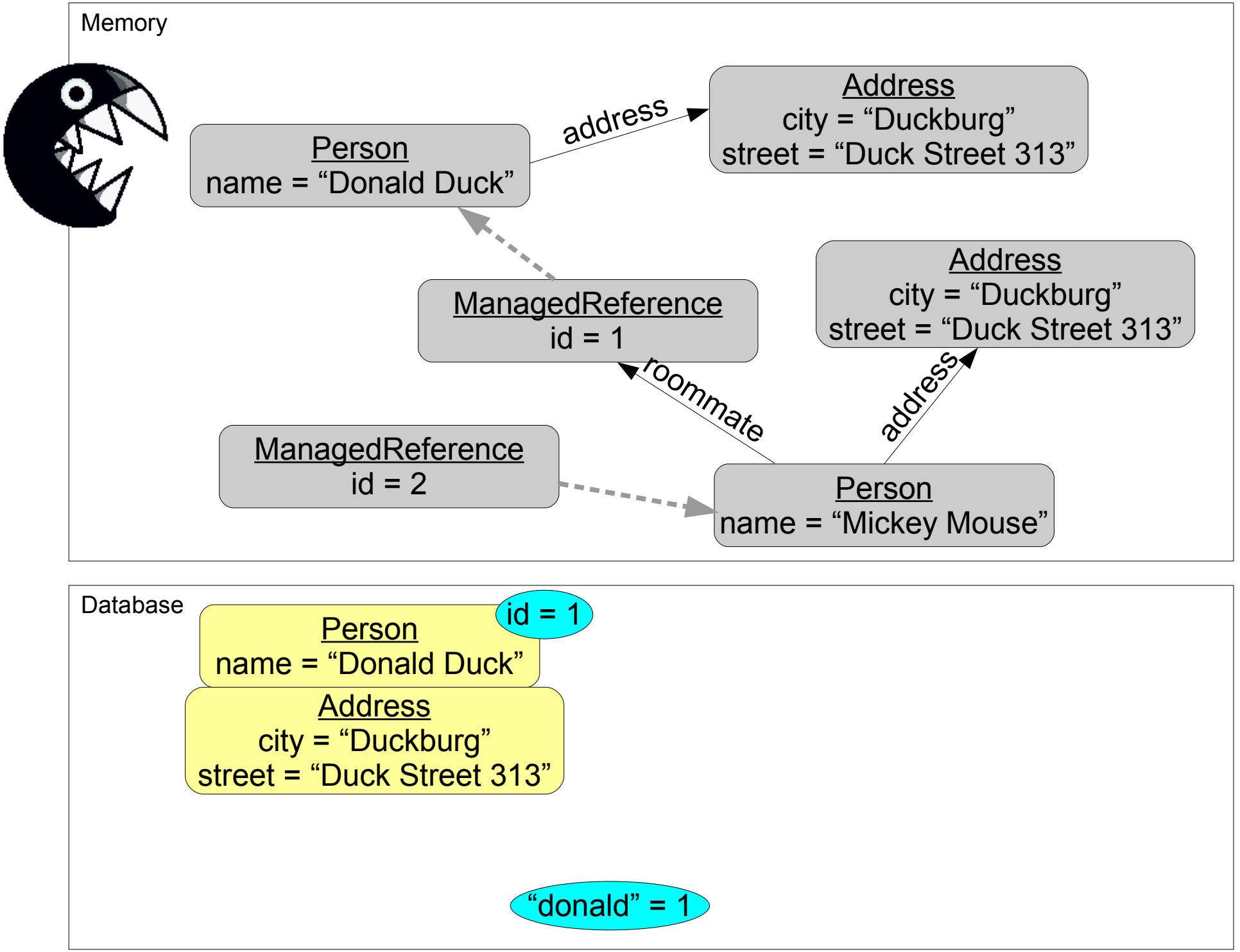
donald.setRoommate(null);




AppContext.getDataManager().removeObject(mickey);
Memory
Transaction
Person
name = “Donald Duck”
id = 1
address
Address
city = “Duckburg”
street = “Duck Street 313”
ManagedReference
id = 1
roommate
Address
city = “Duckburg”
street = “Duck Street 313”
address
ManagedReference
id = 2
Person
name = “Mickey Mouse”
Database
Person
name = “Donald Duck”
id = 1
Address
city = “Duckburg”
street = “Duck Street 313”
ManagedReference
id = 2
“donald” = 1
Person
name = “Mickey Mouse”
id = 2
Address
city = “Duckburg”
street = “Duck Street 313”
ManagedReference
id = 1


COMMIT TRANSACTION
Memory
Transaction
Person
name = “Donald Duck”
id = 1
address
Address
city = “Duckburg”
street = “Duck Street 313”
ManagedReference
id = 1
roommate
Address
city = “Duckburg”
street = “Duck Street 313”
address
Write
ManagedReference
= 2
Person
name = “Mickey Mouse”
Database
Person
name = “Donald Duck”
id = 1
Address
city = “Duckburg”
street = “Duck Street 313”
Person
name = “Mickey Mouse”
id = 2
Address
city = “Duckburg”
street = “Duck Street 313”
ManagedReference
id = 1
“donald” = 1


COMMIT TRANSACTION
Memory
Transaction
Person
name = “Donald Duck”
id = 1
address
Address
city = “Duckburg”
street = “Duck Street 313”
ManagedReference
id = 1
roommate
Address
city = “Duckburg”
street = “Duck Street 313”
address
ManagedReference
id = 2
Person
name = “Mickey Mouse”
Database
Person
name = “Donald Duck”
id = 1
Address
city = “Duckburg”
street = “Duck Street 313”
Delete
“donald” = 1


Memory
Person
name = “Donald Duck”
address
Address
city = “Duckburg”
street = “Duck Street 313”
ManagedReference
id = 1
Address
city = “Duckburg”
street = “Duck Street 313”
roommate
address
ManagedReference
id = 2
Person
name = “Mickey Mouse”
Database
Person
name = “Donald Duck”
id = 1
Address
city = “Duckburg”
street = “Duck Street 313”
“donald” = 1

Memory

Database

Person
name = “Donald Duck”

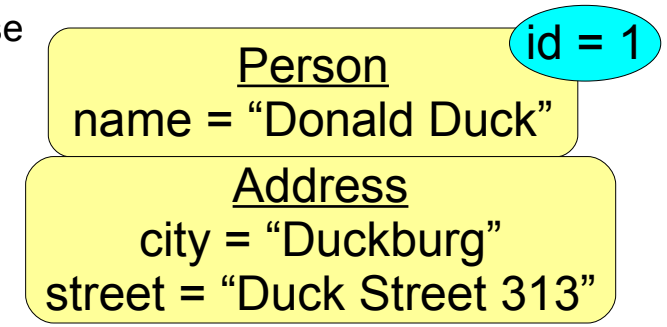
id = 1

Address
city = “Duckburg”
street = “Duck Street 313”

“donald” = 1

Memory

Database



“donald” = 1

What is the problem?

Too much work for the programmer!

Mixed responsibilities! Leaking abstractions! Tight coupling!

```
public class Person implements ManagedObject, Serializable {
    private String name;
    private Address address;
    private ManagedReference<Person> roommate;
      ...
    public Person getRoommate() {
        if (roommate == null) {
            return null;
        } else {
            return roommate.get();
        }
    }

    public void setRoommate(Person roommate) {
        if (roommate == null) {
            this.roommate = null;
        } else {
            this.roommate = AppContext.getDataManager().createReference(roommate);
        }
    }
}
```

Solution

- Make Darkstar itself responsible for taking care of the technical details of ManagedObjects referring to each other
- Code using an object should not know the implementation details of that object – whether it is a ManagedObject or not
- Refer to ManagedObjects through proxies
  - Allows lazy loading
  - Separates object graphs during serialization
  - And the code using the object will never know it!

Agenda

- How Darkstar's persistence model is currently?
- **How transparent references and garbage collection change the persistence model?**
- Future directions

```java
@Entity(ProxyType.CLASS)
public class Person implements Serializable {
    private String name;
    private Address address;
    private Person roommate;

    public Person(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public Address getAddress() {
        return address;
    }

    public void setAddress(Address address) {
        this.address = address;
    }

    public Person getRoommate() {
        return roommate;
    }

    public void setRoommate(Person roommate) {
        this.roommate = roommate;
    }
}
```

```java
public class Address implements Serializable {
    private String city;
    private String street;

    public Address(String city, String street) {
        this.city = city;
        this.street = street;
    }

    public String getCity() {
        return city;
    }

    public String getStreet() {
        return street;
    }
}
```

ProxyType.CLASS
- generated proxy extends target class
- all fields must be private
- all methods must be non-final
- the class must be non-final

(- an accessible default constructor is not needed, as the constructor will not be called)

ProxyType.INTERFACE
- generated proxy implements the same interfaces as the target class
- instances of the class must always be used through their interfaces

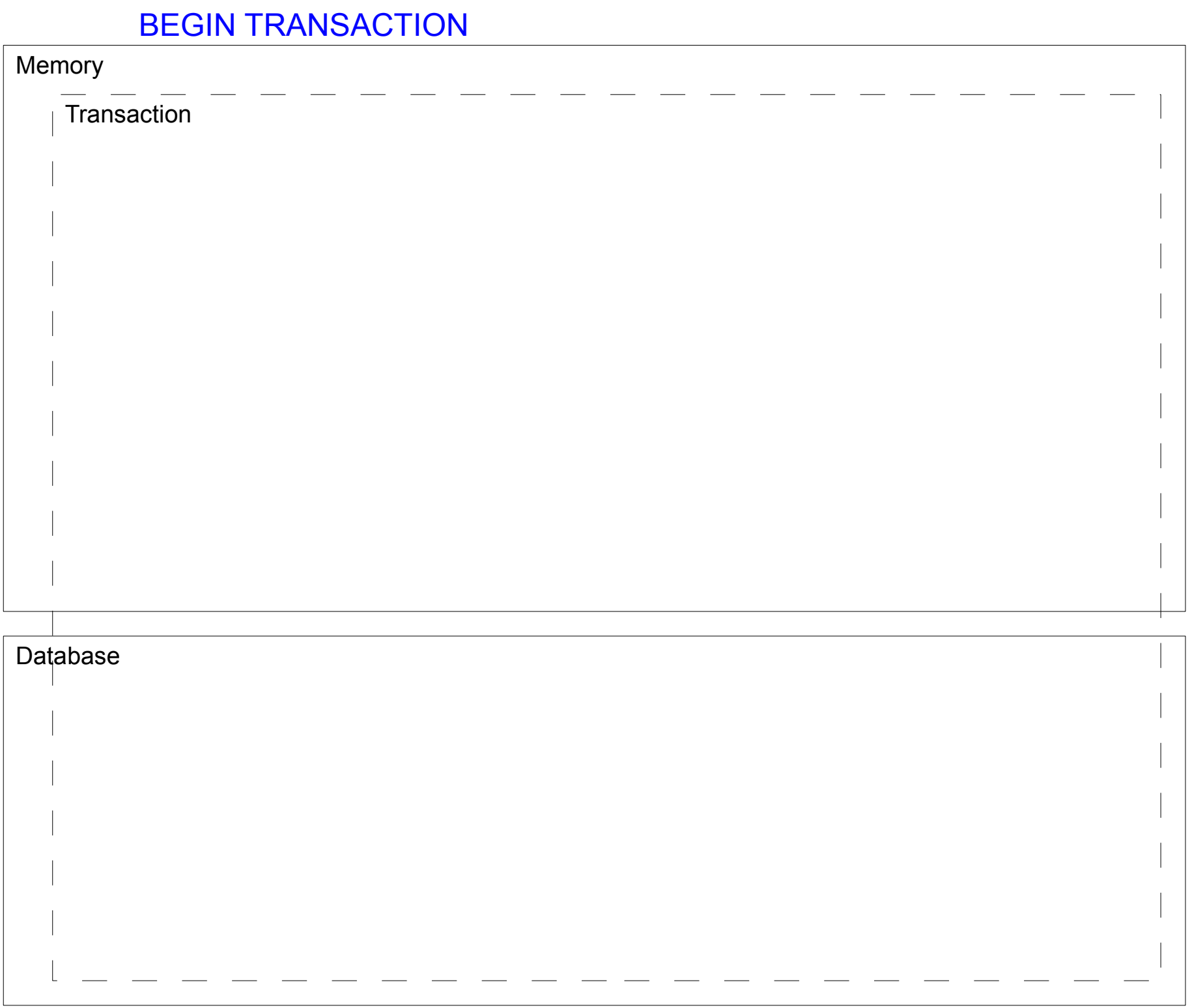
What happens under the hood when the following is executed?

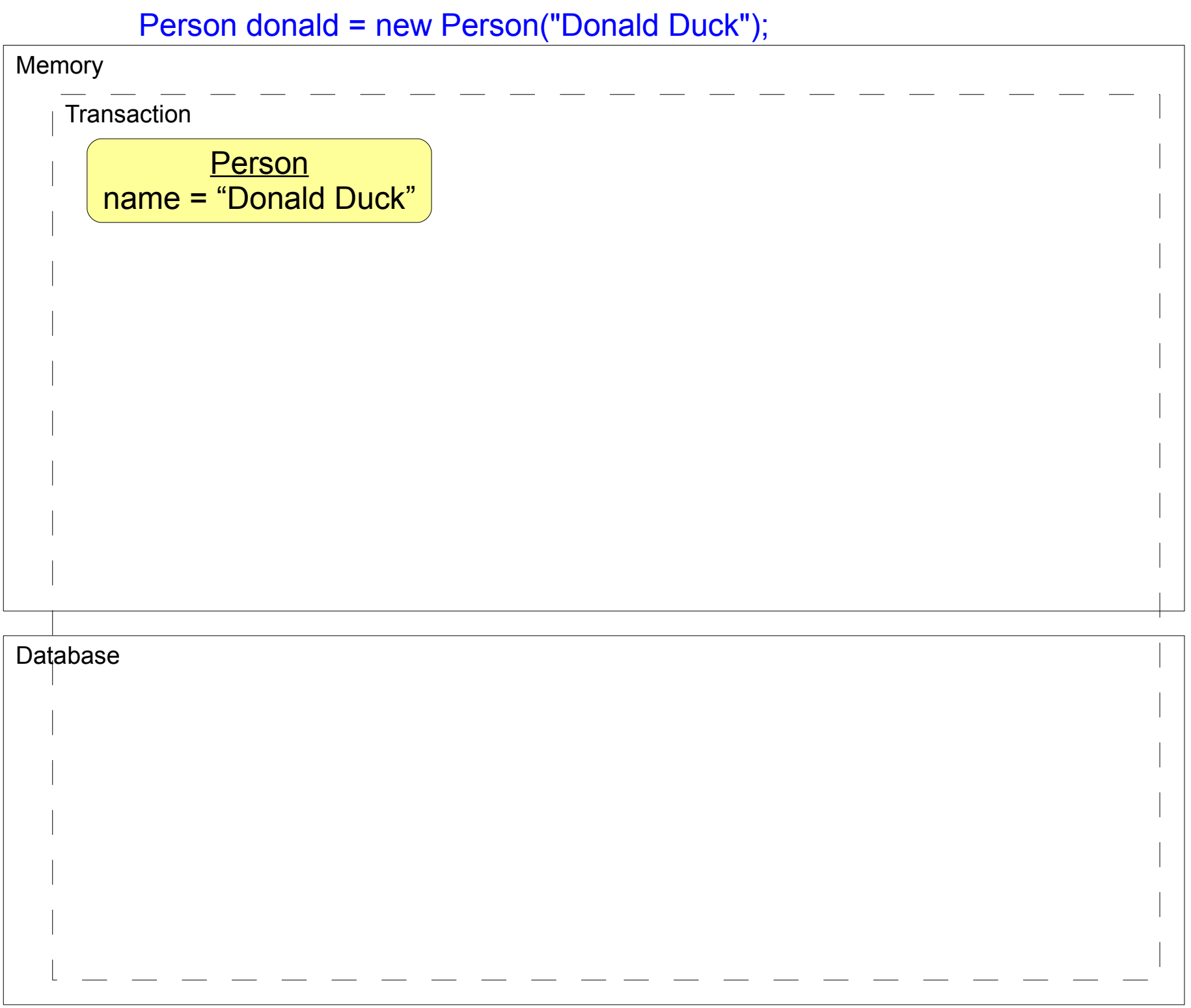
– with transparent references and garbage collection

```
Person donald = new Person("Donald Duck");
donald.setAddress(new Address("Duckburg", "Duck Street 313"));

Person mickey = new Person("Mickey Mouse");
mickey.setAddress(donald.getAddress());
mickey.setRoommate(donald);
donald.setRoommate(mickey);

AppContext.getDataManager().setBinding("donald", donald);
```

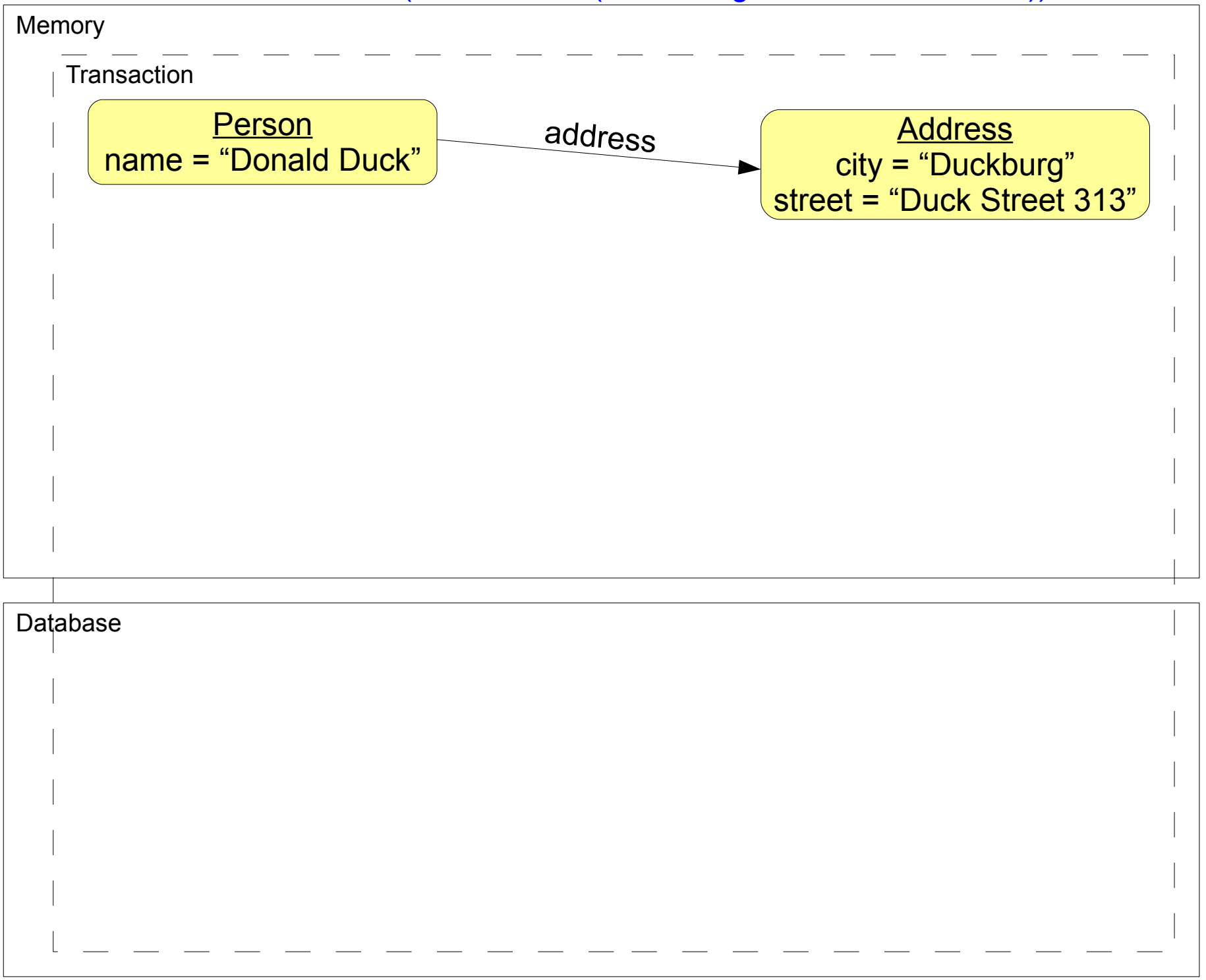
Memory

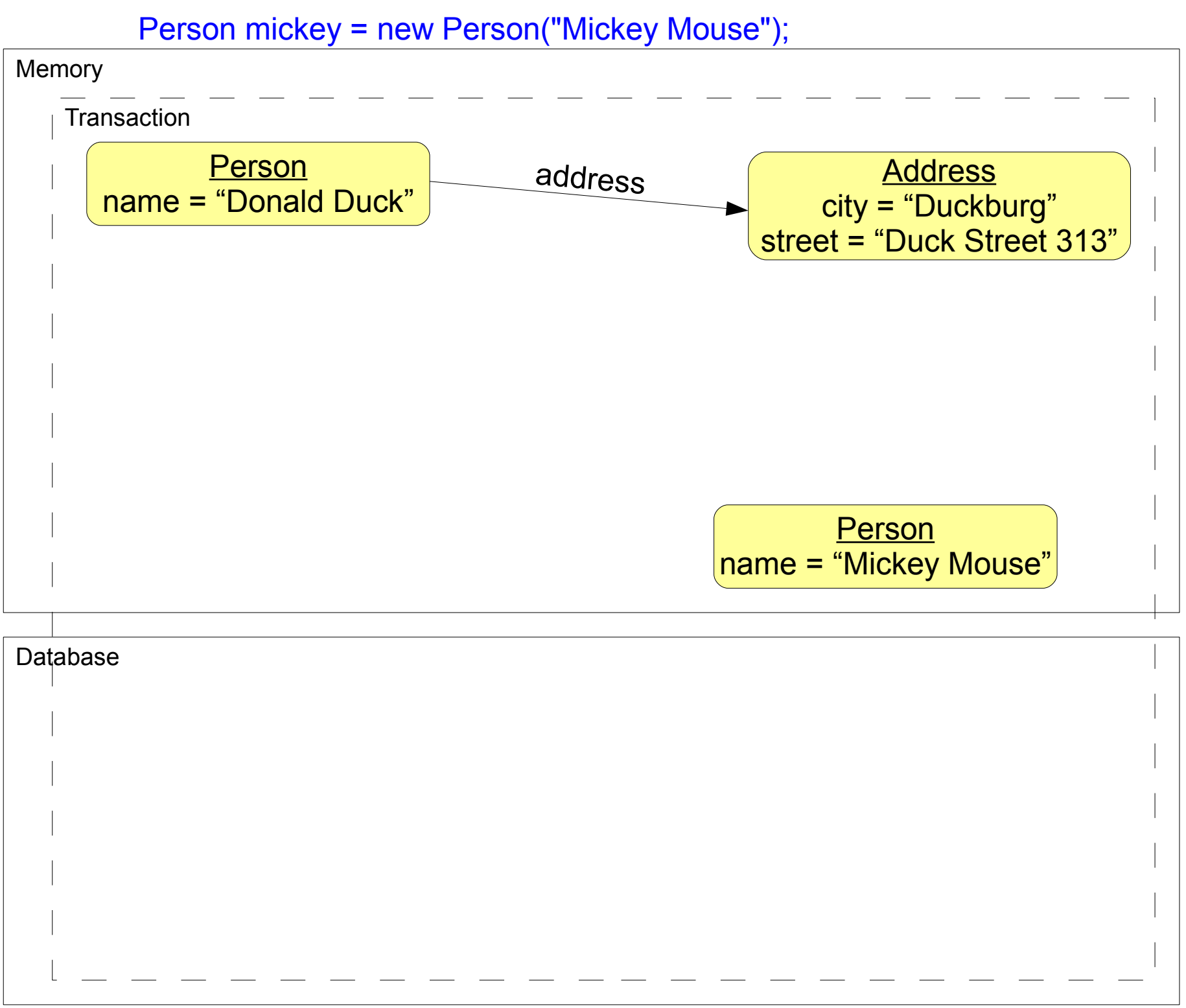
Database


BEGIN TRANSACTION
Memory
Transaction
Database

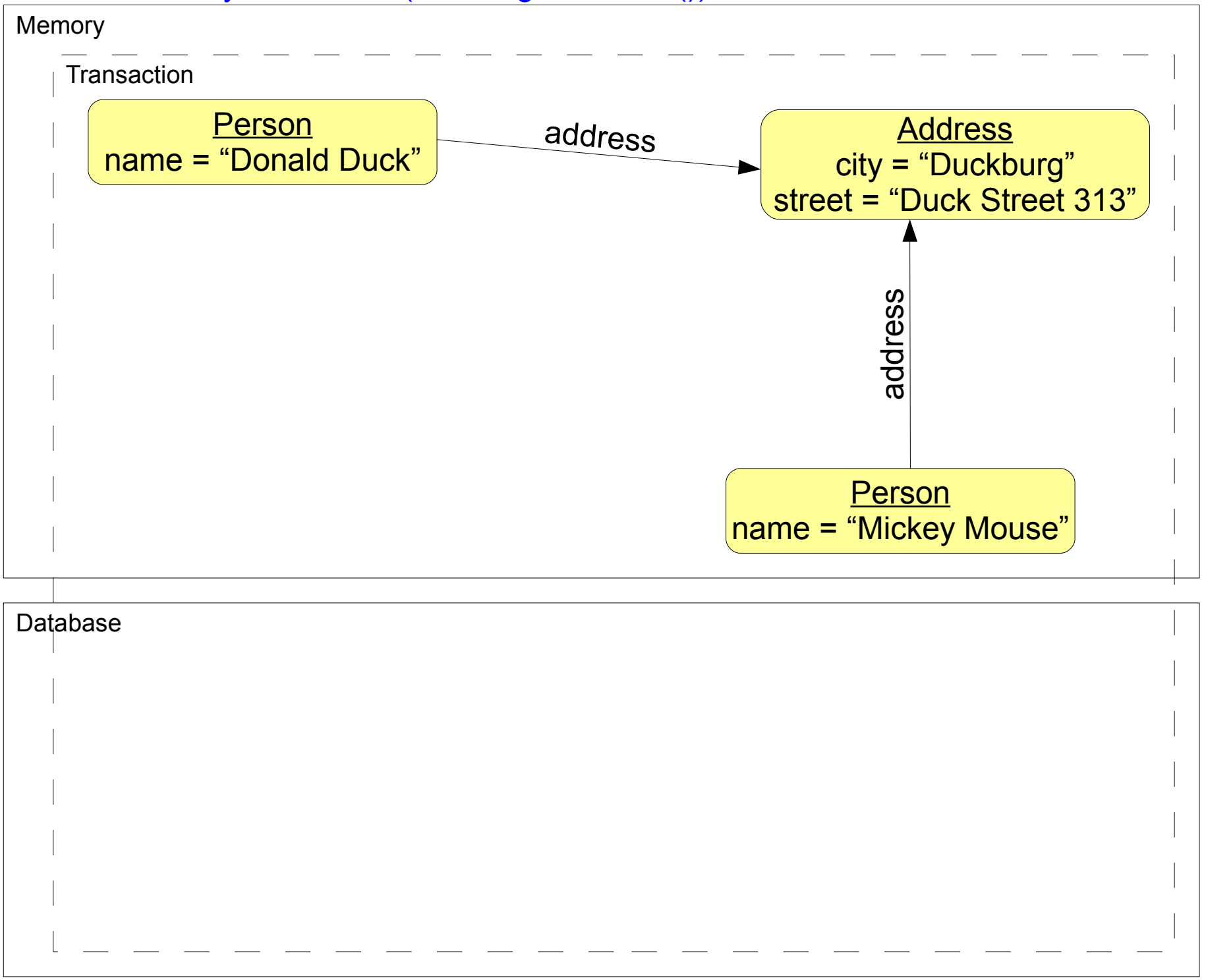

Person donald = new Person("Donald Duck");
Memory
Transaction
Person
name = “Donald Duck”
Database

donald.setAddress(new Address("Duckburg", "Duck Street 313"));

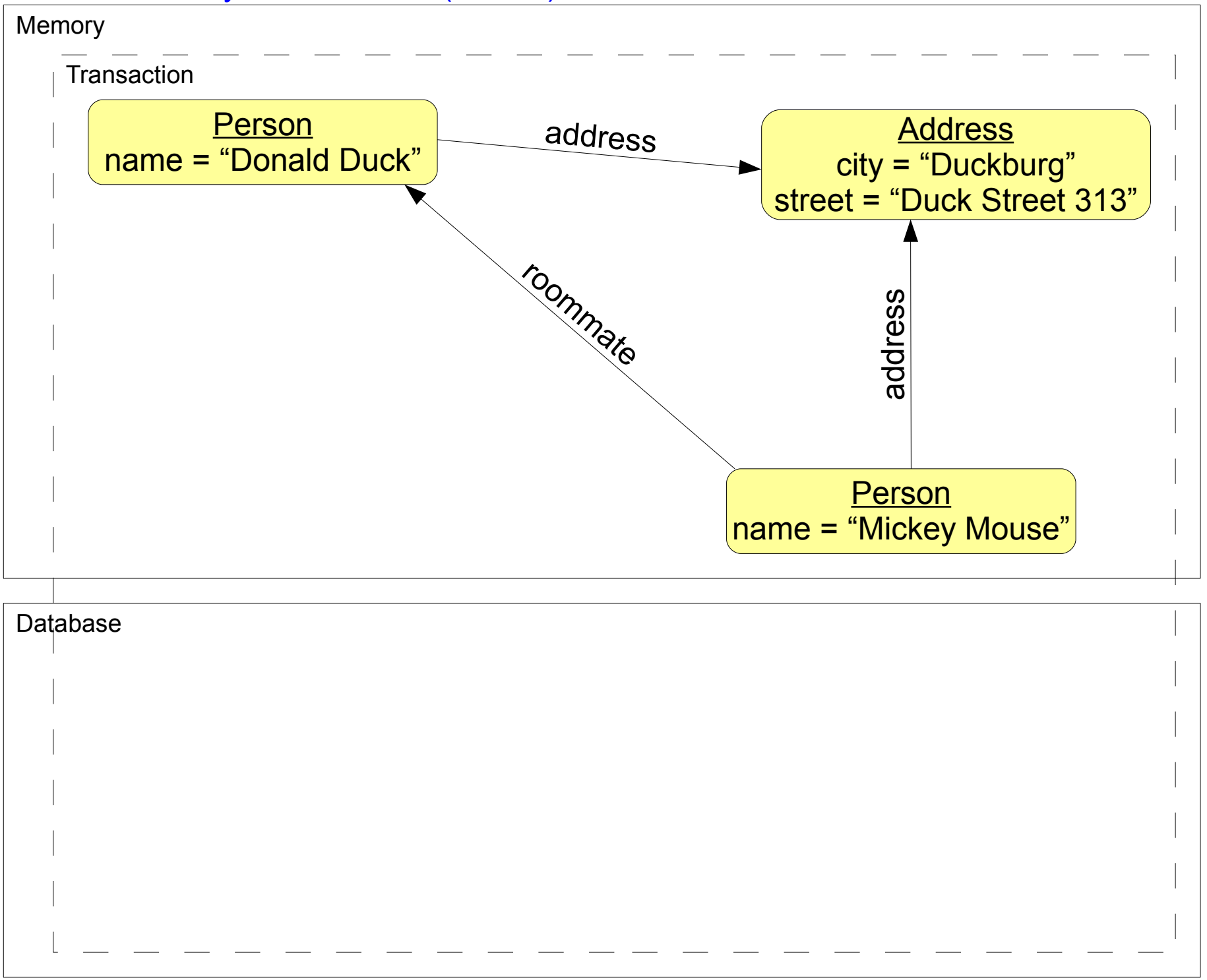



Person mickey = new Person("Mickey Mouse");
Memory
Transaction
Person
name = “Donald Duck”
address
Address
city = “Duckburg”
street = “Duck Street 313”
Person
name = “Mickey Mouse”
Database

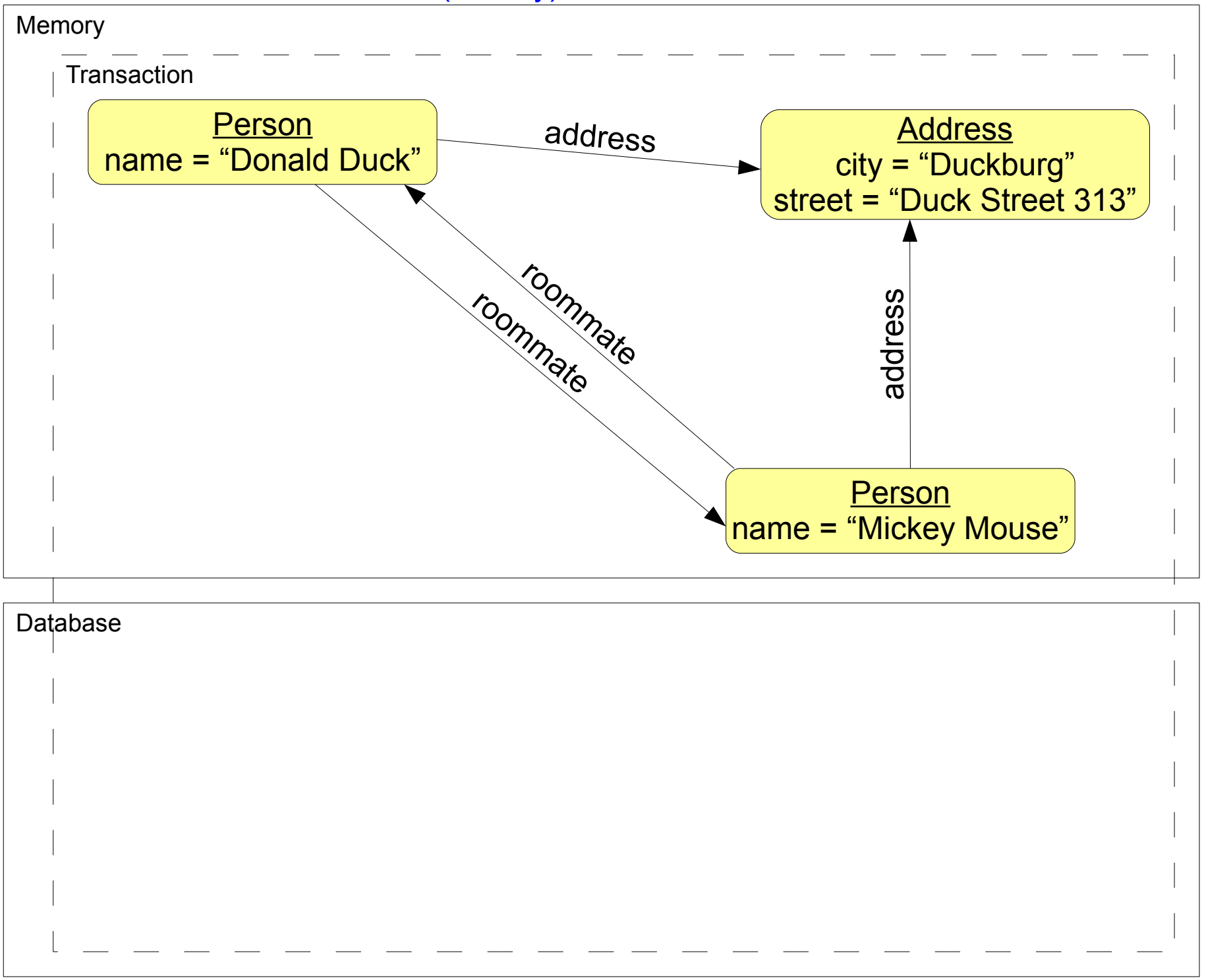
```
mickey.setAddress(donald.getAddress());
```

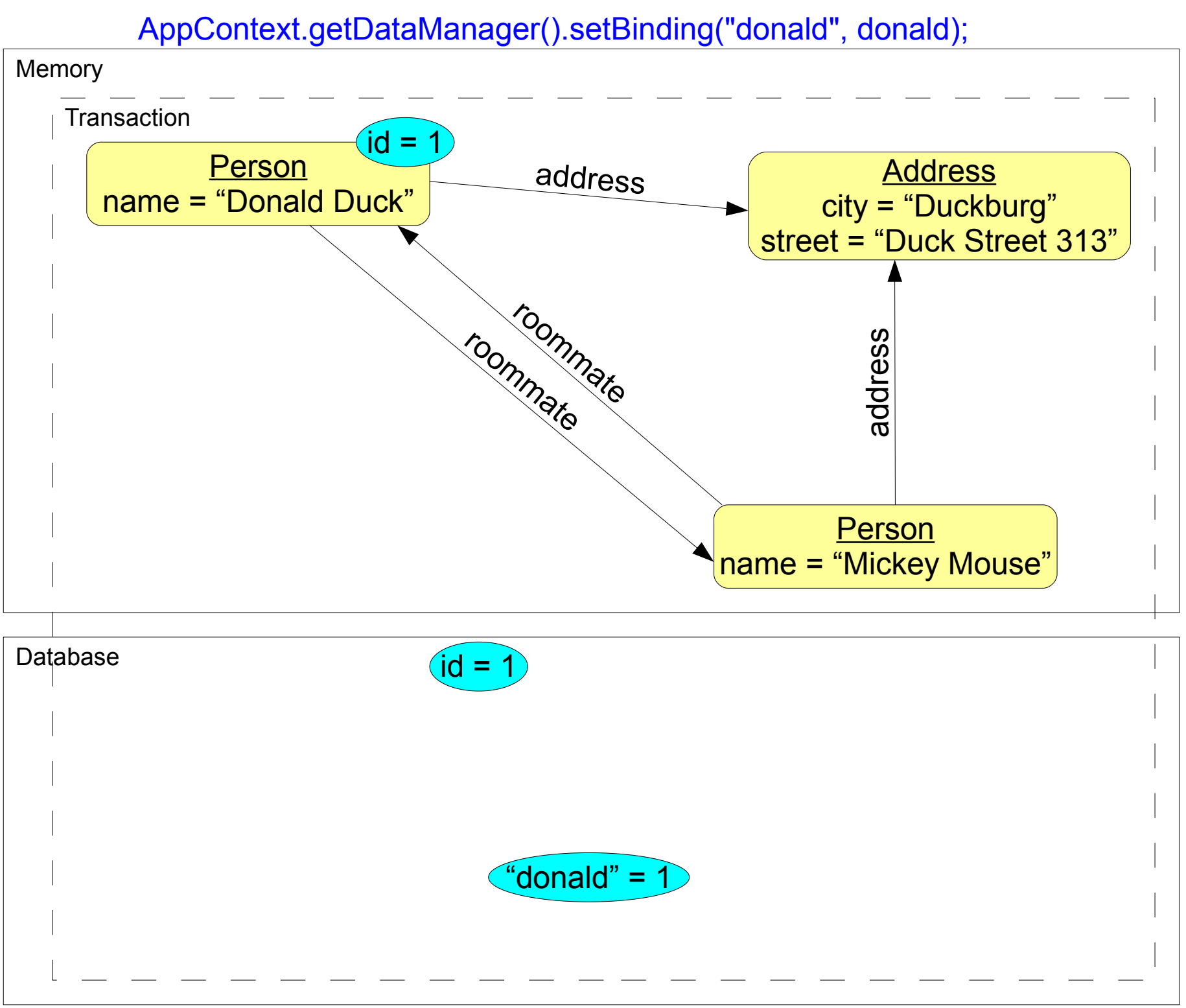


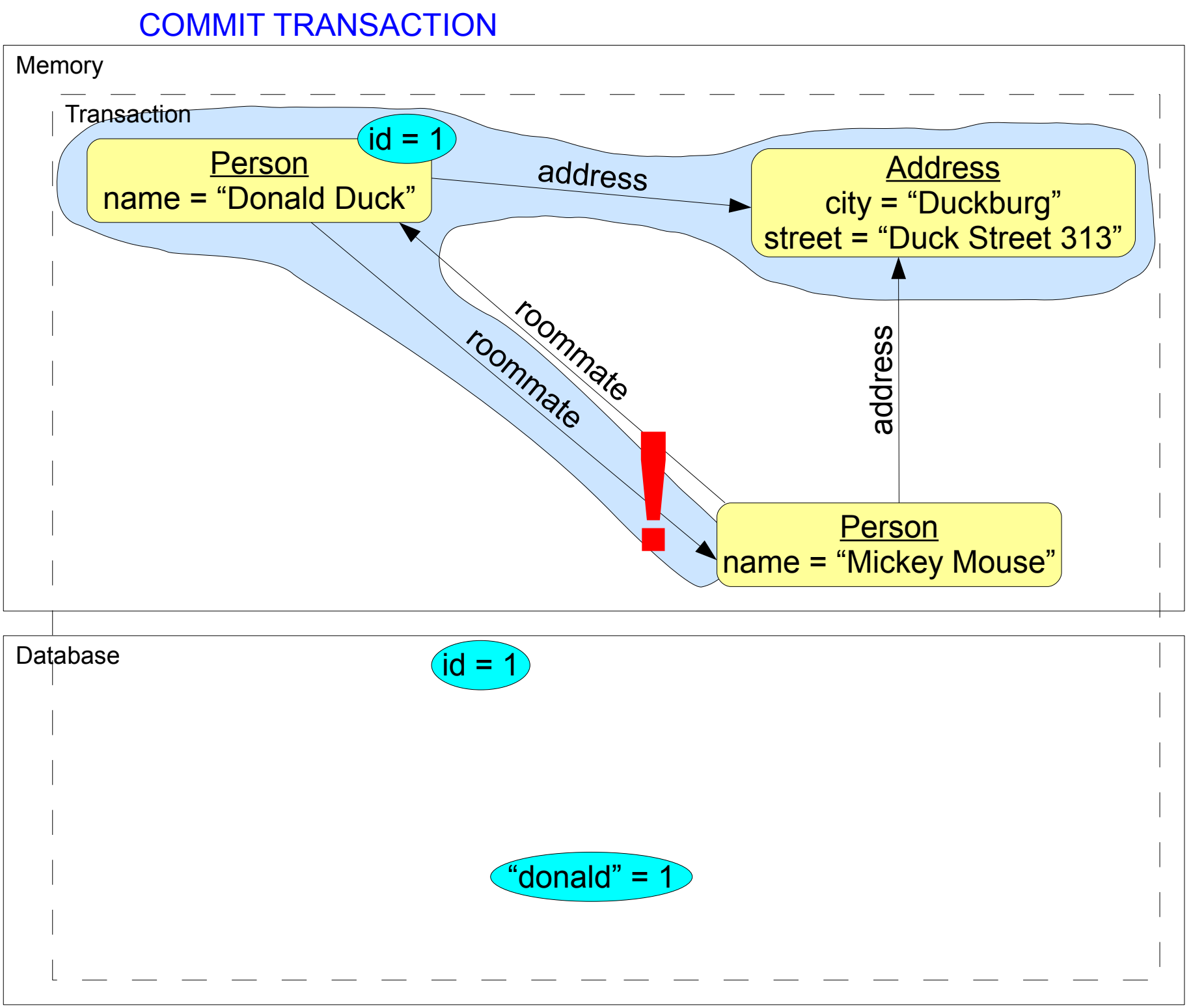
mickey.setRoommate(donald);

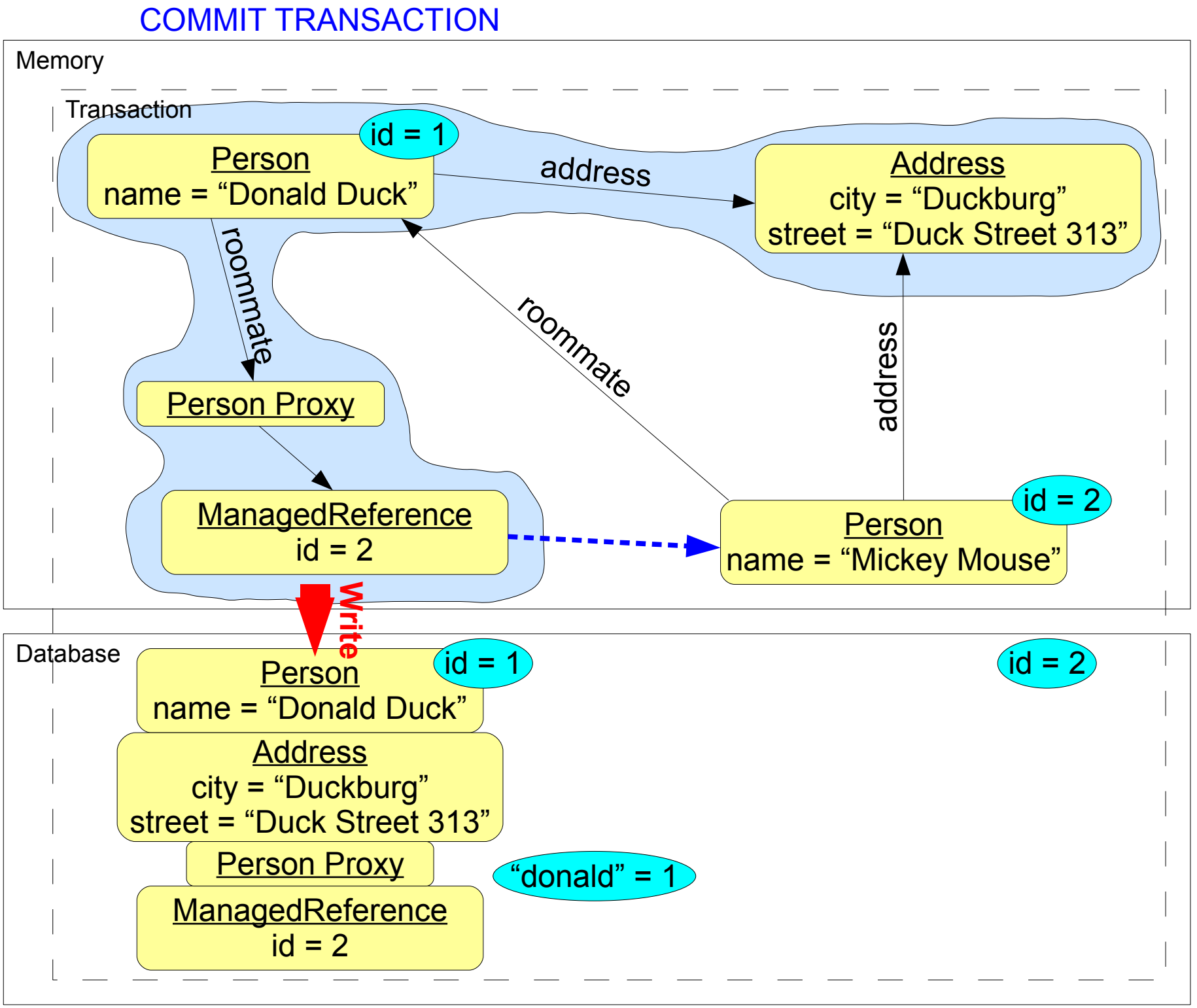


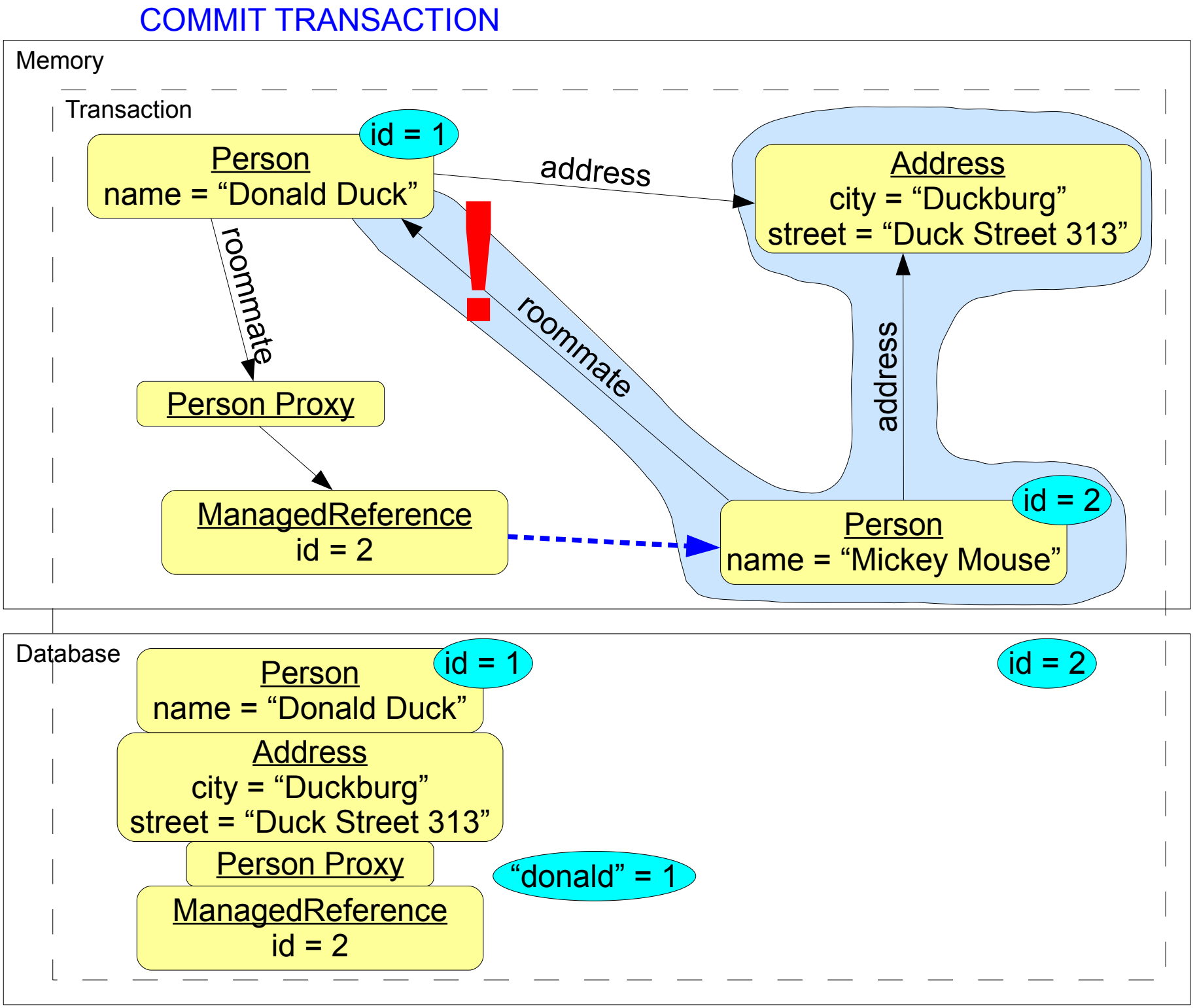
donald.setRoommate(mickey);

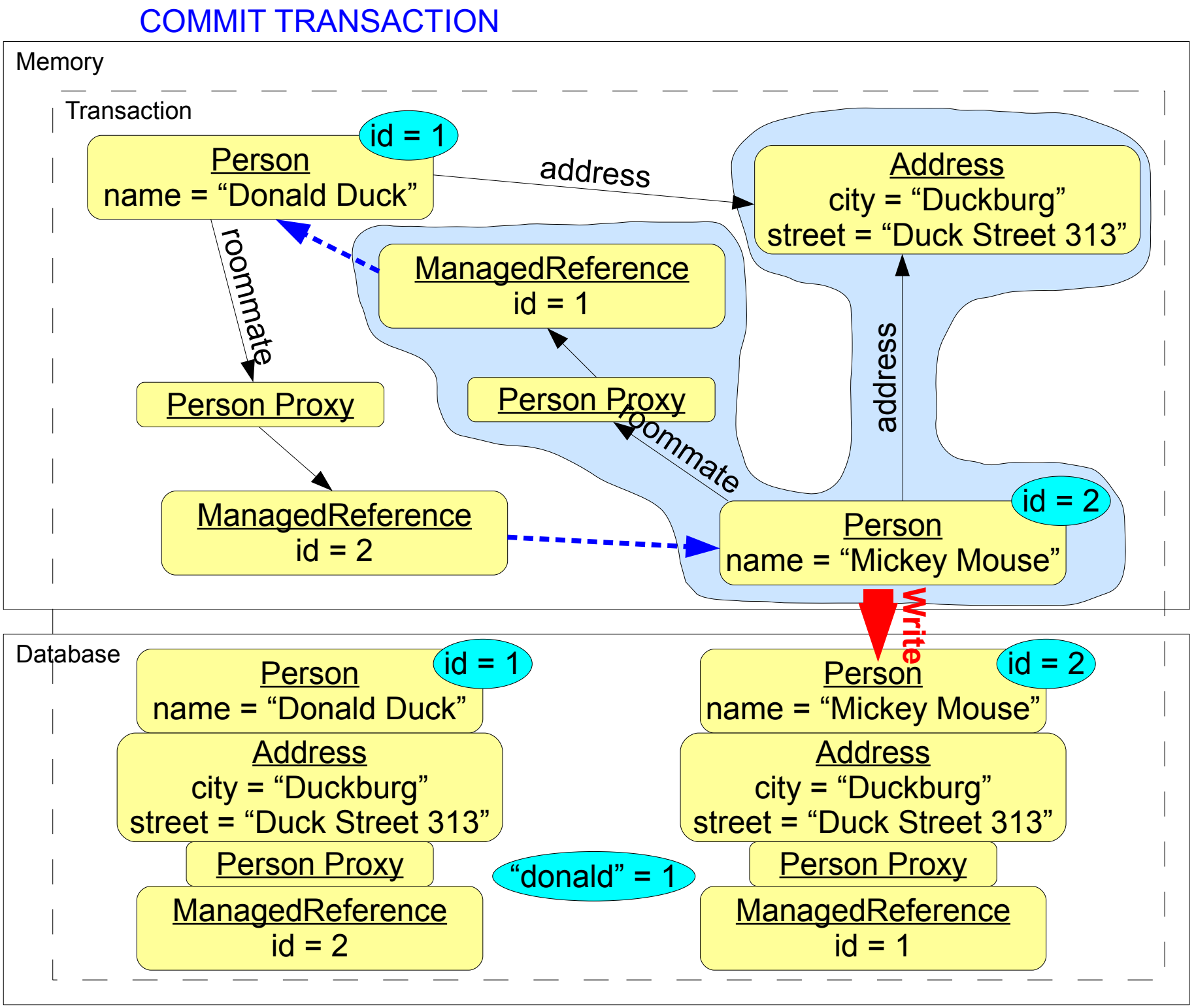


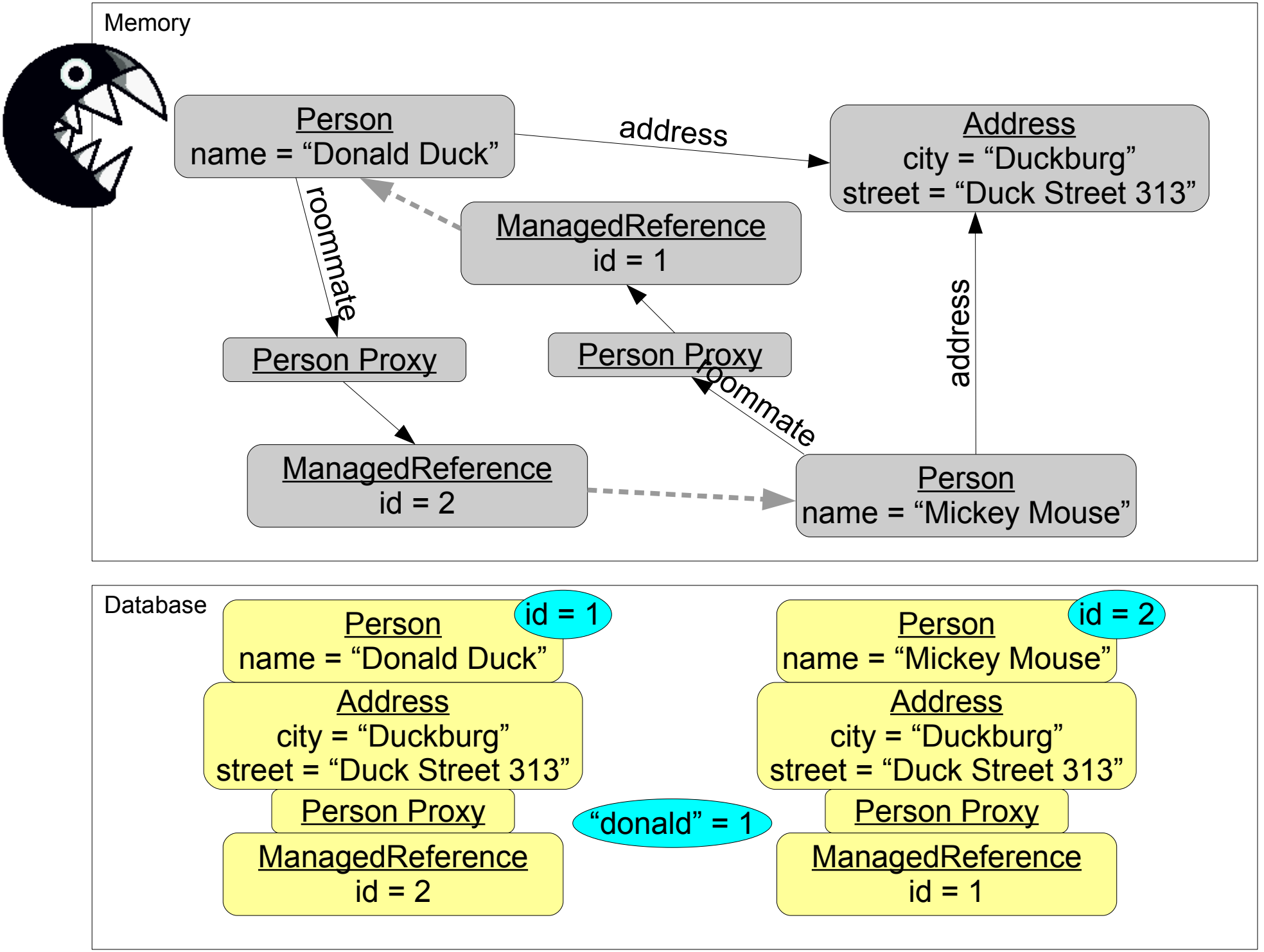

AppContext.getDataManager().setBinding("donald", donald);
Memory
Transaction
id = 1
Person
name = “Donald Duck”
address
Address
city = “Duckburg”
street = “Duck Street 313”
roommate
roommate
address
Person
name = “Mickey Mouse”
Database
id = 1
“donald” = 1


COMMIT TRANSACTION
Memory
Transaction
id = 1
Person
name = “Donald Duck”
address
Address
city = “Duckburg”
street = “Duck Street 313”
roommate
roommate
address
!
Person
name = “Mickey Mouse”
Database
id = 1
“donald” = 1


COMMIT TRANSACTION
Memory
Transaction
id = 1
Person
name = “Donald Duck”
address
Address
city = “Duckburg”
street = “Duck Street 313”
roommate
roommate
address
Person Proxy
ManagedReference
id = 2
id = 2
Person
name = “Mickey Mouse”
Write
Database
Person
name = “Donald Duck”
id = 1
id = 2
Address
city = “Duckburg”
street = “Duck Street 313”
Person Proxy
“donald” = 1
ManagedReference
id = 2


COMMIT TRANSACTION
Memory
Transaction
Person
name = “Donald Duck”
id = 1
address
Address
city = “Duckburg”
street = “Duck Street 313”
roommate
roommate
address
Person Proxy
ManagedReference
id = 2
Person
name = “Mickey Mouse”
id = 2
Database
Person
name = “Donald Duck”
id = 1
id = 2
Address
city = “Duckburg”
street = “Duck Street 313”
Person Proxy
“donald” = 1
ManagedReference
id = 2


COMMIT TRANSACTION
Memory
Transaction
Person
name = “Donald Duck”
id = 1
address
Address
city = “Duckburg”
street = “Duck Street 313”
roommate
ManagedReference
id = 1
Person Proxy
Person Proxy
roommate
address
ManagedReference
id = 2
Person
name = “Mickey Mouse”
id = 2
Write
Database
Person
name = “Donald Duck”
id = 1
Address
city = “Duckburg”
street = “Duck Street 313”
Person Proxy
ManagedReference
id = 2
“donald” = 1
Person
name = “Mickey Mouse”
id = 2
Address
city = “Duckburg”
street = “Duck Street 313”
Person Proxy
ManagedReference
id = 1


Memory
Person
name = “Donald Duck”
address
Address
city = “Duckburg”
street = “Duck Street 313”
roommate
ManagedReference
id = 1
Person Proxy
Person Proxy
address
roommate
ManagedReference
id = 2
Person
name = “Mickey Mouse”
Database
Person
name = “Donald Duck”
id = 1
Address
city = “Duckburg”
street = “Duck Street 313”
Person Proxy
ManagedReference
id = 2
“donald” = 1
Person
name = “Mickey Mouse”
id = 2
Address
city = “Duckburg”
street = “Duck Street 313”
Person Proxy
ManagedReference
id = 1

Memory

Database

Person
name = “Donald Duck”
id = 1

Address
city = “Duckburg”
street = “Duck Street 313”

Person Proxy

ManagedReference
id = 2

“donald” = 1

Person
name = “Mickey Mouse”
id = 2

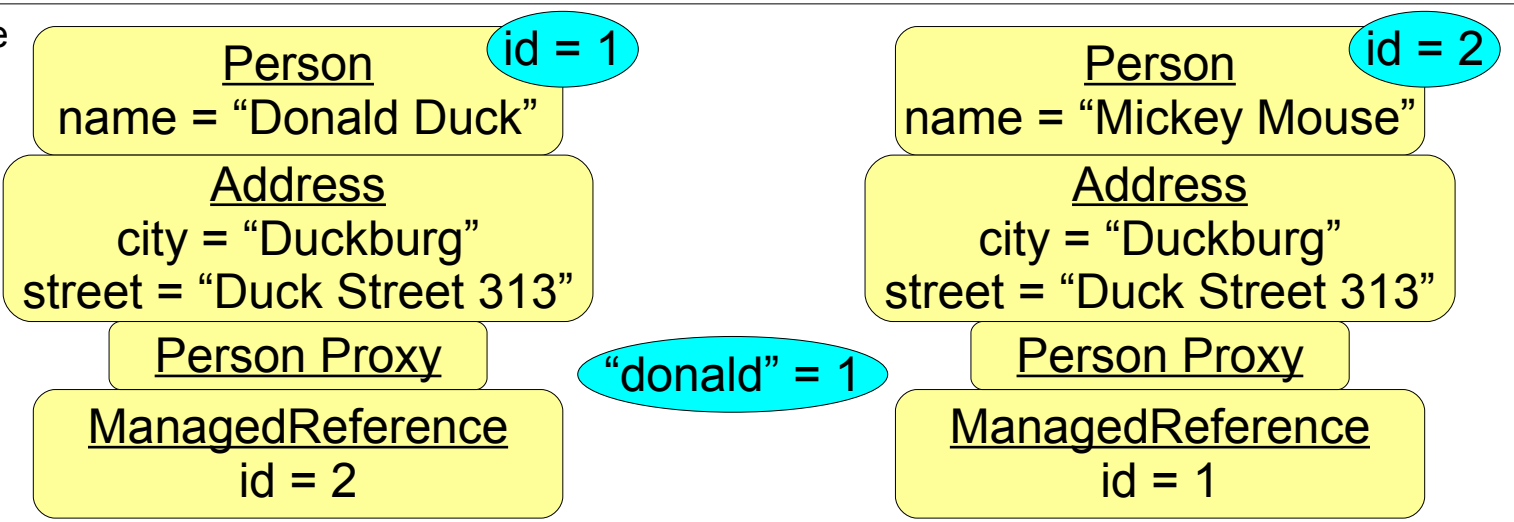
Address
city = “Duckburg”
street = “Duck Street 313”

Person Proxy

ManagedReference
id = 1

Memory

Database



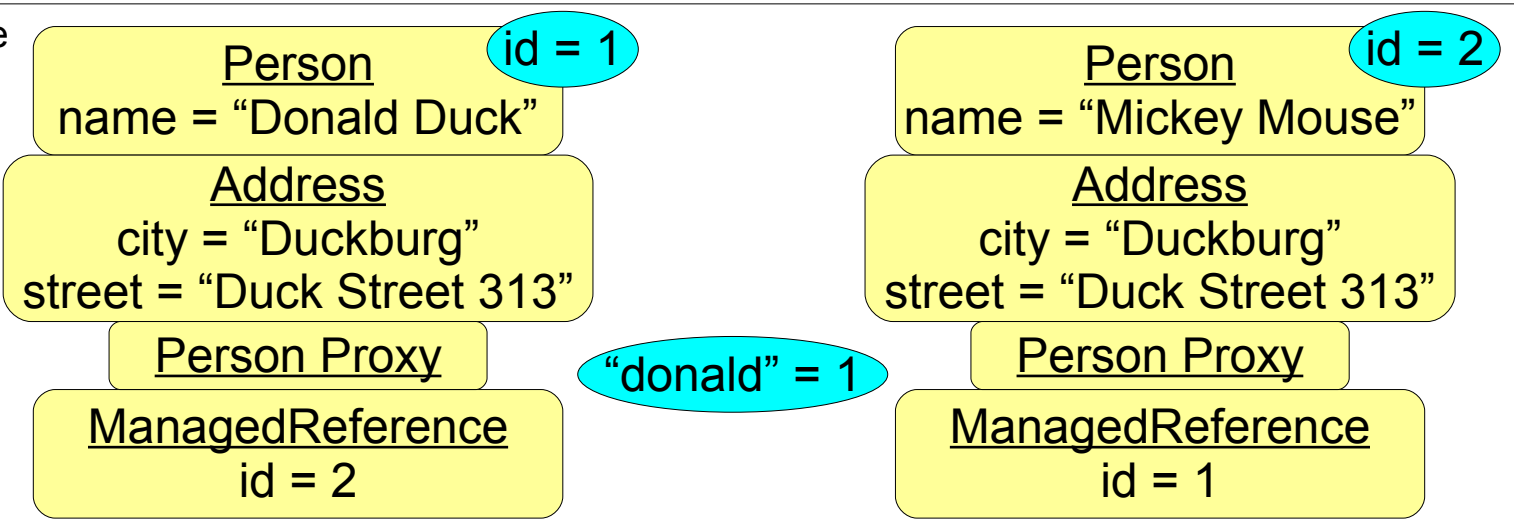
What happens under the hood when the following is executed?

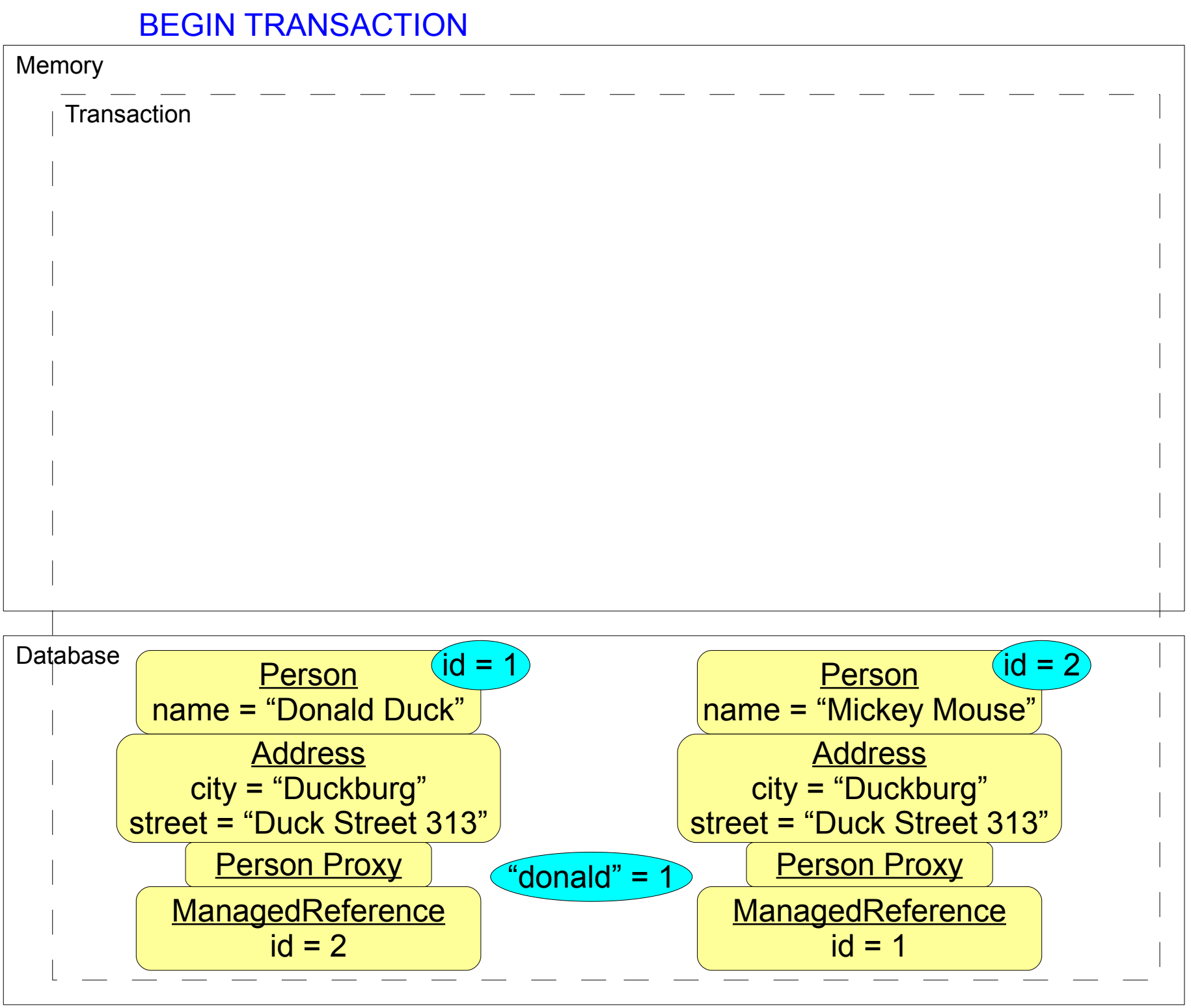
– with transparent references and garbage collection

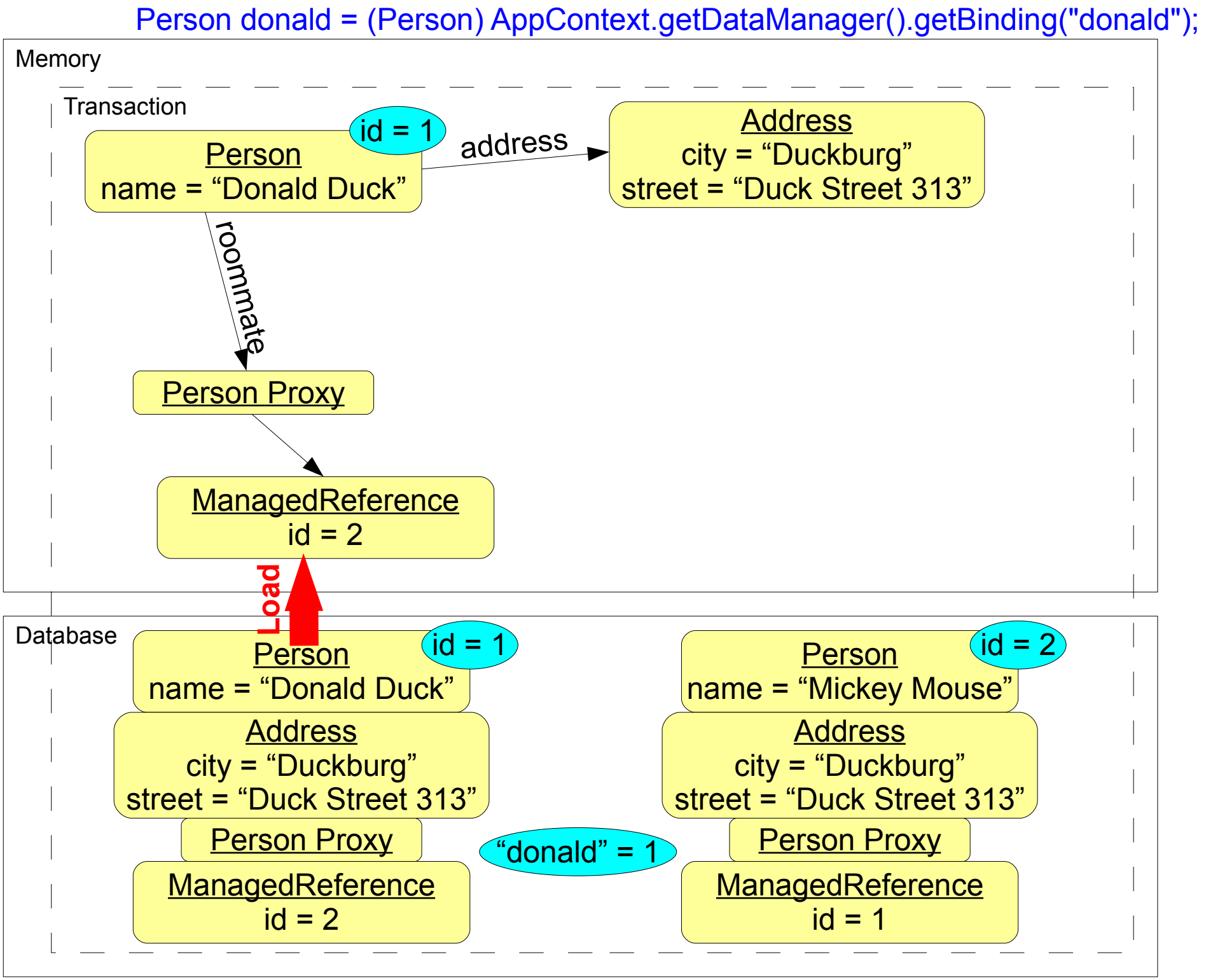
```
Person donald = (Person) AppContext.getDataManager().getBinding("donald");

Person mickey = donald.getRoommate();
System.out.println(mickey.getName() + " lives in " +
                   mickey.getAddress().getCity());

donald.setRoommate(null);
```

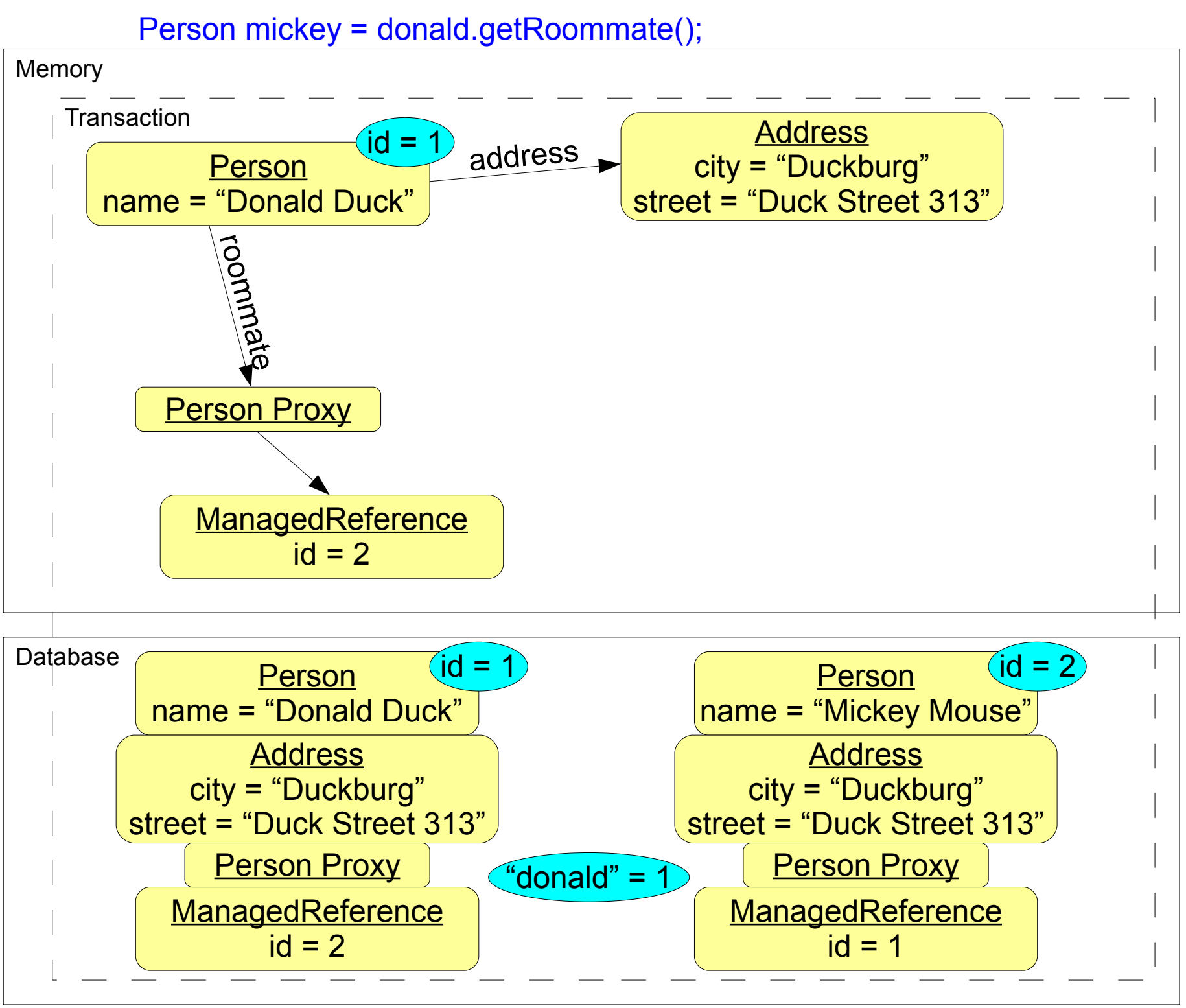
Memory

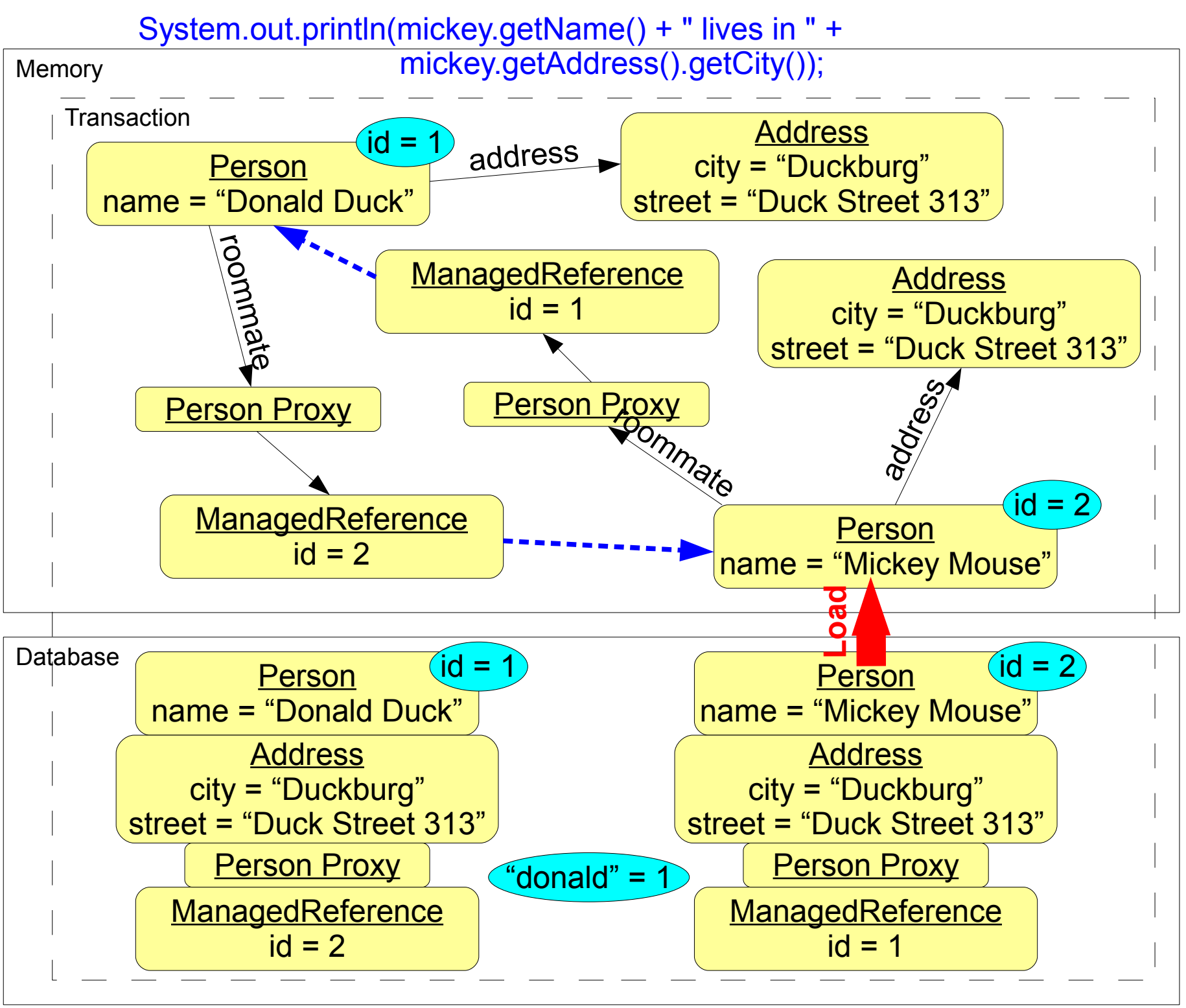
Database



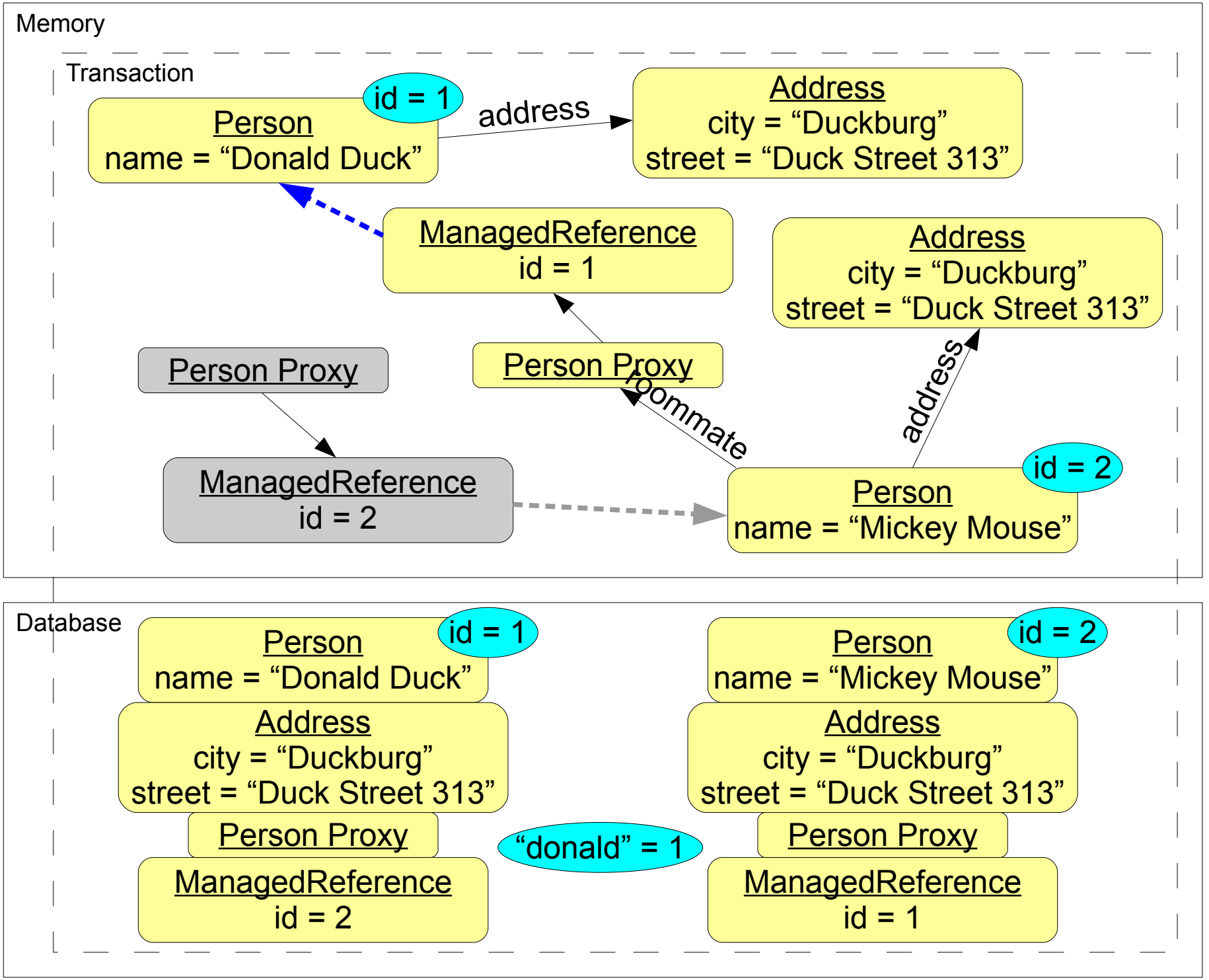

BEGIN TRANSACTION
Memory
Transaction
Database
Person
name = “Donald Duck”
id = 1
Address
city = “Duckburg”
street = “Duck Street 313”
Person Proxy
ManagedReference
id = 2
“donald” = 1
Person
name = “Mickey Mouse”
id = 2
Address
city = “Duckburg”
street = “Duck Street 313”
Person Proxy
ManagedReference
id = 1

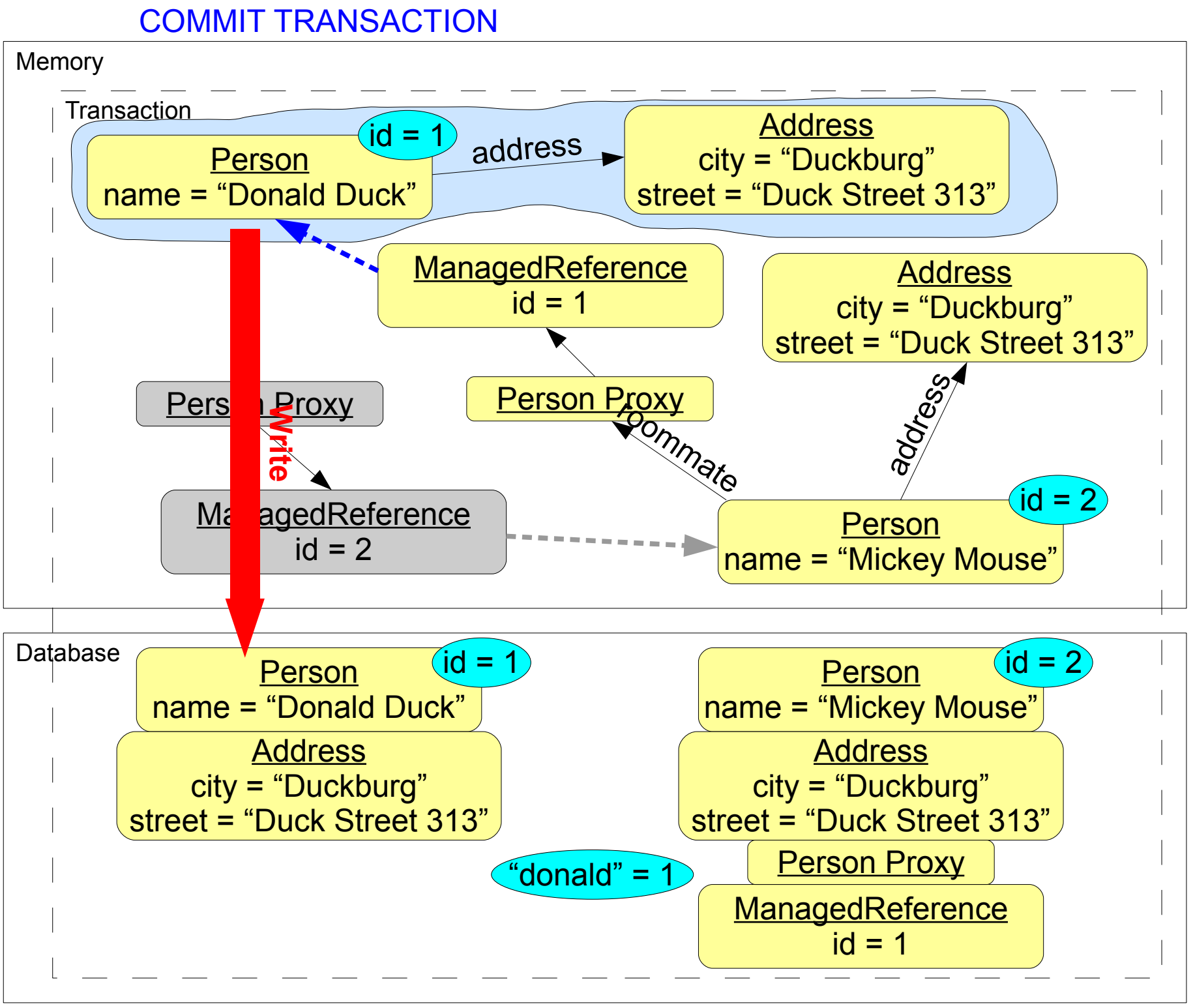

Person donald = (Person) AppContext.getDataManager().getBinding("donald");
Memory
Transaction
id = 1
Person
name = “Donald Duck”
address
Address
city = “Duckburg”
street = “Duck Street 313”
roommate
Person Proxy
ManagedReference
id = 2
Load
Database
Person
name = “Donald Duck”
id = 1
Address
city = “Duckburg”
street = “Duck Street 313”
Person Proxy
ManagedReference
id = 2
“donald” = 1
Person
name = “Mickey Mouse”
id = 2
Address
city = “Duckburg”
street = “Duck Street 313”
Person Proxy
ManagedReference
id = 1

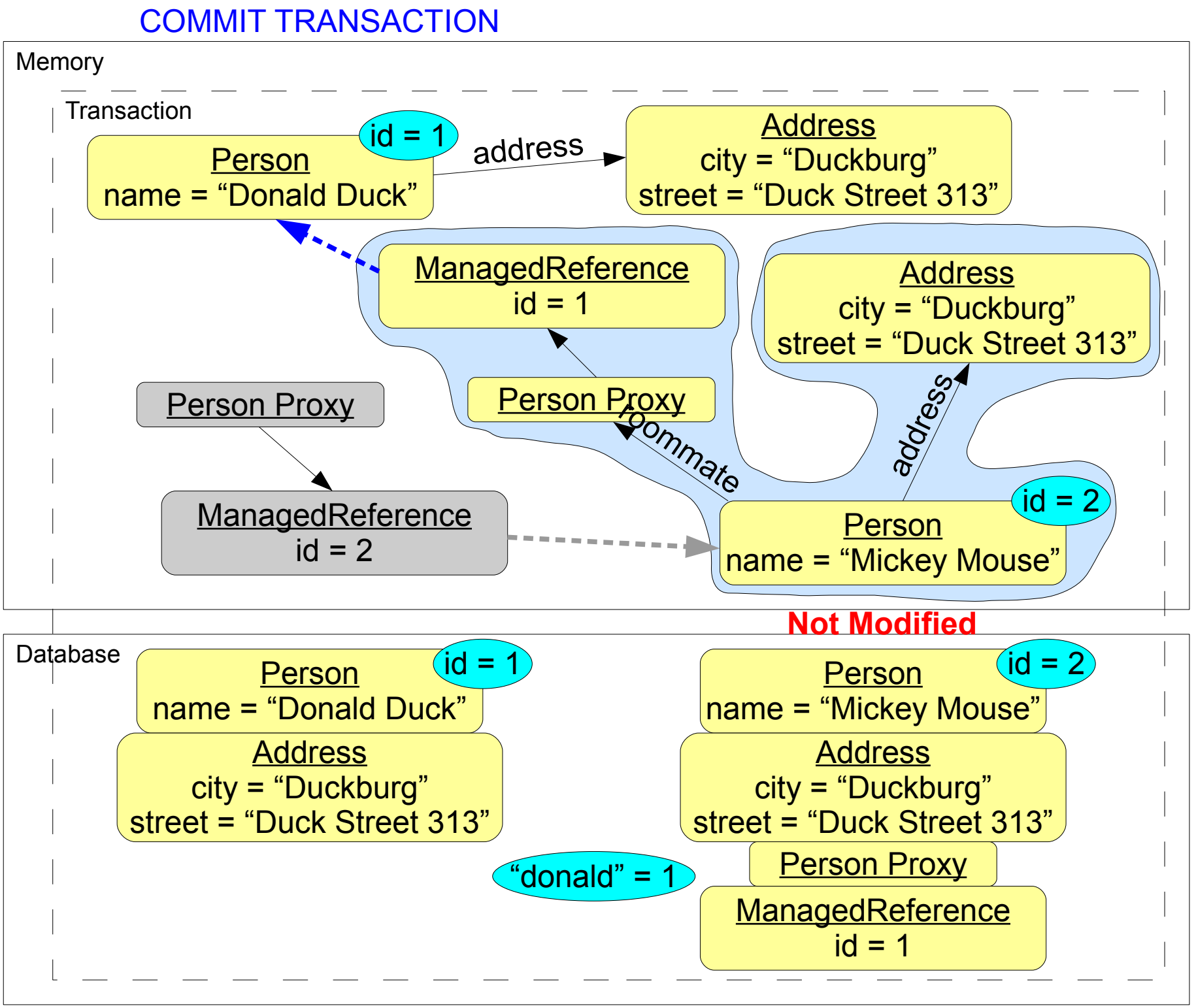

Person mickey = donald.getRoommate();
Memory
Transaction
id = 1
Person
name = “Donald Duck”
address
Address
city = “Duckburg”
street = “Duck Street 313”
roommate
Person Proxy
ManagedReference
id = 2
Database
Person
name = “Donald Duck”
id = 1
Address
city = “Duckburg”
street = “Duck Street 313”
Person Proxy
ManagedReference
id = 2
“donald” = 1
Person
name = “Mickey Mouse”
id = 2
Address
city = “Duckburg”
street = “Duck Street 313”
Person Proxy
ManagedReference
id = 1

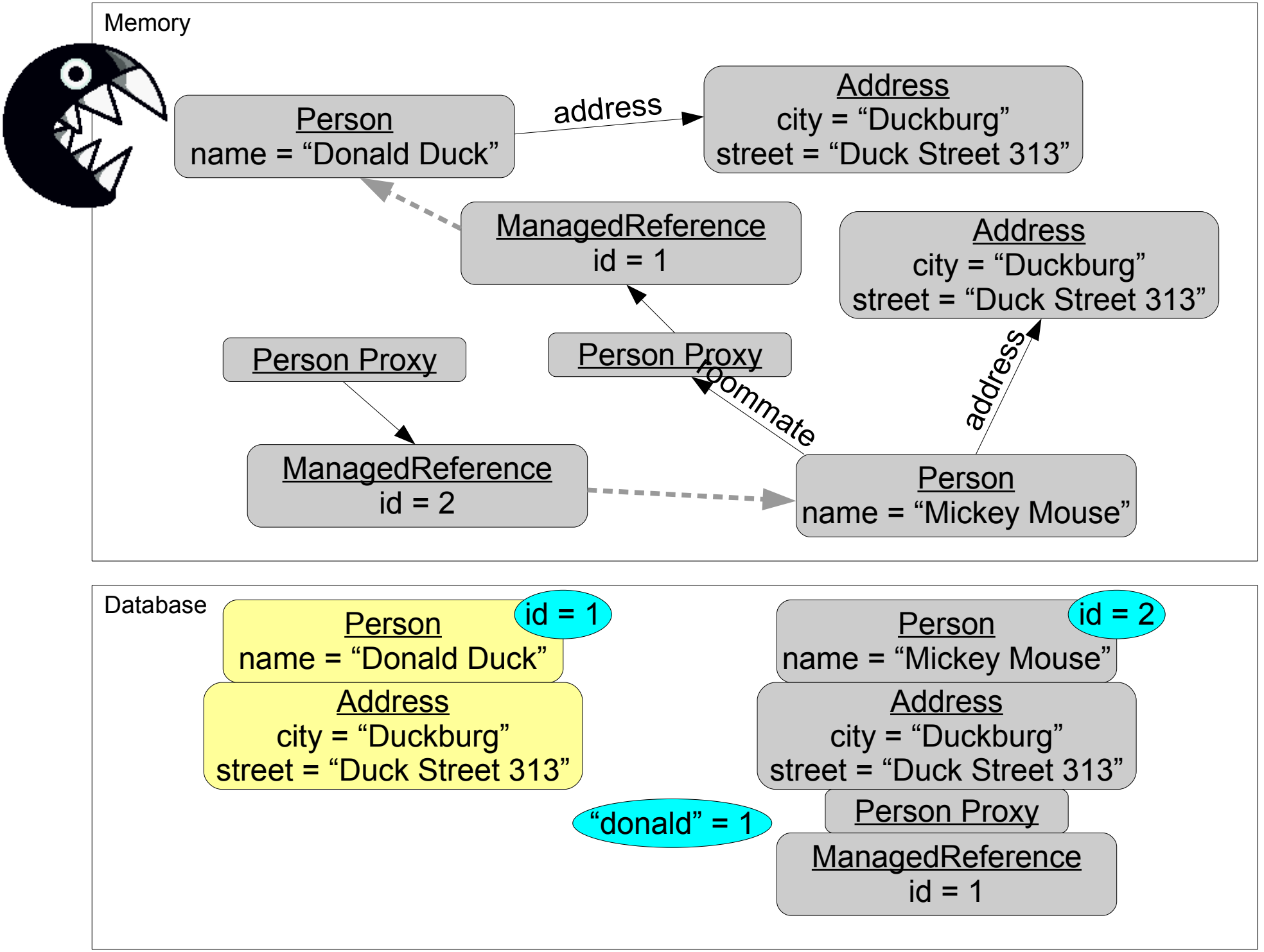

System.out.println(mickey.getName() + " lives in " +
mickey.getAddress().getCity());
Memory
Transaction
id = 1
Person
name = “Donald Duck”
address
Address
city = “Duckburg”
street = “Duck Street 313”
roommate
ManagedReference
id = 1
Address
city = “Duckburg”
street = “Duck Street 313”
Person Proxy
Person Proxy
roommate
address
ManagedReference
id = 2
id = 2
Person
name = “Mickey Mouse”
Load
Database
id = 1
Person
name = “Donald Duck”
id = 2
Person
name = “Mickey Mouse”
Address
city = “Duckburg”
street = “Duck Street 313”
Address
city = “Duckburg”
street = “Duck Street 313”
Person Proxy
“donald” = 1
Person Proxy
ManagedReference
id = 2
ManagedReference
id = 1

```
donald.setRoommate(null);
```




COMMIT TRANSACTION
Memory
Transaction
Person
name = “Donald Duck”
id = 1
address
Address
city = “Duckburg”
street = “Duck Street 313”
ManagedReference
id = 1
Address
city = “Duckburg”
street = “Duck Street 313”
Person Proxy
Person Proxy
Write
roommate
address
ManagedReference
id = 2
Person
name = “Mickey Mouse”
id = 2
Database
Person
name = “Donald Duck”
id = 1
Address
city = “Duckburg”
street = “Duck Street 313”
Person
name = “Mickey Mouse”
id = 2
Address
city = “Duckburg”
street = “Duck Street 313”
“donald” = 1
Person Proxy
ManagedReference
id = 1


COMMIT TRANSACTION
Memory
Transaction
Person
name = “Donald Duck”
id = 1
address
Address
city = “Duckburg”
street = “Duck Street 313”
ManagedReference
id = 1
Address
city = “Duckburg”
street = “Duck Street 313”
Person Proxy
Person Proxy
roommate
address
ManagedReference
id = 2
Person
name = “Mickey Mouse”
id = 2
Not Modified
Database
Person
name = “Donald Duck”
id = 1
Address
city = “Duckburg”
street = “Duck Street 313”
Person
name = “Mickey Mouse”
id = 2
Address
city = “Duckburg”
street = “Duck Street 313”
“donald” = 1
Person Proxy
ManagedReference
id = 1


Memory
Person
name = “Donald Duck”
address
Address
city = “Duckburg”
street = “Duck Street 313”
ManagedReference
id = 1
Address
city = “Duckburg”
street = “Duck Street 313”
Person Proxy
Person Proxy
roommate
address
ManagedReference
id = 2
Person
name = “Mickey Mouse”
Database
Person
name = “Donald Duck”
id = 1
Address
city = “Duckburg”
street = “Duck Street 313”
Person
name = “Mickey Mouse”
id = 2
Address
city = “Duckburg”
street = “Duck Street 313”
“donald” = 1
Person Proxy
ManagedReference
id = 1

Memory

Database

Person
name = “Donald Duck”
id = 1

Address
city = “Duckburg”
street = “Duck Street 313”

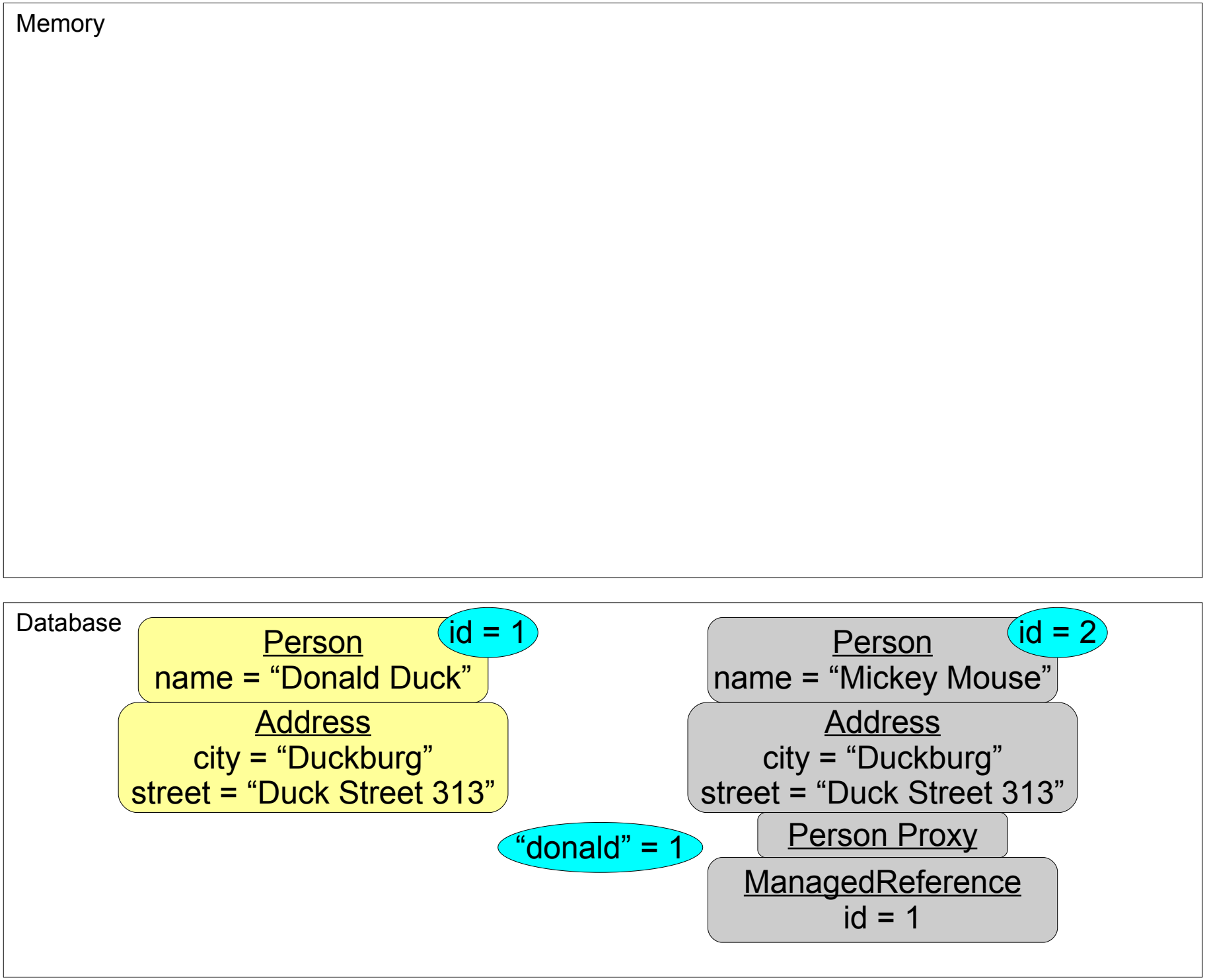
“donald” = 1

Person
name = “Mickey Mouse”
id = 2

Address
city = “Duckburg”
street = “Duck Street 313”

Person Proxy

ManagedReference
id = 1


Memory
Database
Person
name = “Donald Duck”
id = 1
Address
city = “Duckburg”
street = “Duck Street 313”
“donald” = 1
Person
name = “Mickey Mouse”
id = 2
Address
city = “Duckburg”
street = “Duck Street 313”
Person Proxy
ManagedReference
id = 1

Memory

Database

Person
name = “Donald Duck”

id = 1

Address
city = “Duckburg”
street = “Duck Street 313”

“donald” = 1

Person
name = “Mickey Mouse”

id = 2

Address
city = “Duckburg”
street = “Duck Street 313”

Person Proxy

ManagedReference
id = 1

Memory

Database

Person
name = “Donald Duck”

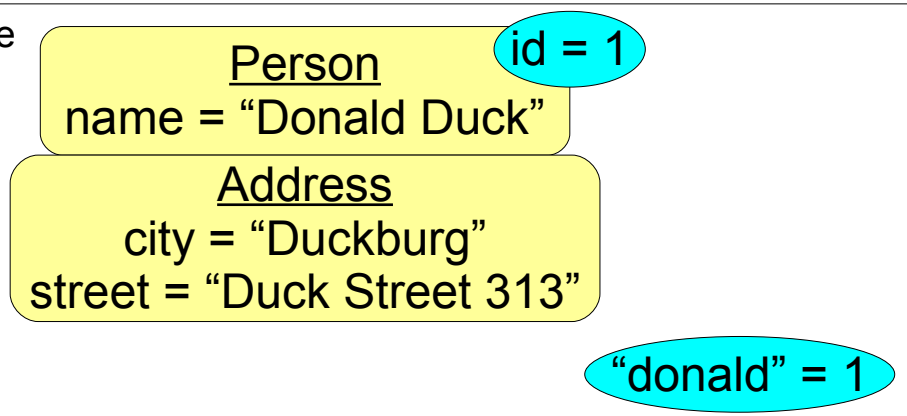
id = 1

Address
city = “Duckburg”
street = “Duck Street 313”

“donald” = 1

Memory

Database



Agenda

- How Darkstar's persistence model is currently?
- How transparent references and garbage collection change the persistence model?
- **Future directions**

Future directions

- Include transparent references in the official Project Darkstar Server distribution
- Areas which need more work
  - Scalability of the multi-node version
  - Improved garbage collector, multi-node support, alternative algorithms, reference counting vs. tracing collectors
  - Refactoring serialized data, rolling upgrades
  - Indexing and querying the database
  - ...
- It's all open source!

Questions?

Project Darkstar Community

– official web site, discussion forums

http://www.projectdarkstar.com/

Darkstar EXP

– unofficial distribution of Darkstar Server with experimental features, includes transparent references and garbage collector

http://code.google.com/p/darkstar-exp/

Dimdwarf Application Server

– same transparent reference implementation, alternative server implementation, primarily designed for embedded standalone mode

http://dimdwarf.sourceforge.net/