Delete operation in Association mapping

==========

==========

(From OneToMany Prospective)

In One to many Assoication if we delete parent object... then it not only deletes parent table record.. but it also deletes the associated reccords in child table..becoz of cascadeType.ALL

In service Impl class

```
@Override
public String deleteByPerson(int personId) {
//Load Parent obj ...
Optional<Person> opt=perRepo.findById(personId); if(opt.isPresent()) {
perRepo.delete(opt.get());
The db tables of the association can not be there
in different DB s/ws i.e they must be there in same DB s/w
return "Person and his PhoneNumbers are deleted";
return "Person not found";
}
}
a
```

Deleling only childs of parent In OneToMany Association

=======

=========

```
a
```

=> Here Load parent obj, get assoicated child objs of parent object, nullify Parent object from child objects .. then perform delete operation on child objects.by nullifying parent obj from child objs. Code in Service Impl class

```
@Override
public String deleteAllPhoneNumbersOfAPerson(int personId) {
//Load Parent obj
Optional<Person> opt=perRepo.findById(personId);
if(opt.isPresent()) {
//get all childs of a parent
Set<PhoneNumber> childs=opt.get().getContactDetails();
//delete all childs
childs.forEach(ph->{
});
ph.setPerson(null);
phoneRepo.deleteAll(childs); (or) phoneRepo.deleteAllinBatch(childs);
```

```
return childs.size()+" Phonenumbers of "+personId +"Person are deleted";
return personId+" Person not found";
}
```

Adding new Child to existing childs of a Parent

=========

==========

In service Impl class

```
@Override
public void addNewChildToAParentById(int id) {
//Load parent object
Optional<Person> opt=personRepo.findById(id);
if(opt.isPresent()) {
Person per-opt.get();
//get childs of a Parent
Set<PhoneNumber> childs=per.getContactDetails();
//create the new child object
```

cascade operations are relted

to non-select persistence operations where as fetch operations are

related to select persistence operations

```
PhoneNumber ph=new PhoneNumber(7777777L, "vodafone","personal"); //link child to parent
ph.setPersonInfo(per);
childs.add(ph);
personRepo.save(per);
System.out.println("New Child is added to the existing childs of a Parent");
```

In Paarent Entity class we must

take FetchType.EAGER while executing this code.

=>Assume the requirement at tables level but implement requirement through the objects of the AssociationMapping

```
}
else {
}
 }
System.out.println(id+" person not found for operation");
```

Deleting childs and its parent

======

```
@Override
```

========

(From Many To One Prospective)

public String remove PhoneNumbersAndTheirCustomer(Iterable<Integer> regNos) {

//Load childs by ids

Iterable<PhoneNumber> list-phoneRepo.findAllById(regNos);

//detele childs objs

list.forEach(ph->{

});

phoneRepo.delete(ph);

return StreamSupport.stream(list.spliterator(), false).count()+" no.of childs and their parents are deleted";

}

Joins in Spring data JPA Using HQL/JPQL

==========

=> Joins are given to get data from two db tables of association having some implicit conditions..

=> we can also add new conditions on the top of implicit conditions..

=> in SQL to work with joins the two db tables need not be in relationship.. where as in HQL/JPQL

the two db tables needs to be in association to apply joins.. (tables must be there relation/association using FK column)

=> HQL/JPQL supports 4 types of joins

a) inner join b) rigth join /right outer join c) left join /left outer join d)full join or full outer join

OTM_PERSON (parent table

Columns Data Model Constraints Grai Xa Sort..

(pk) PID PADDRS PNAME

OTM_PHONENUMBER (child table)

Columns Data Model Constraints Grants Statistics Triggers Flashback | Depe

EXPL Sort.. Filter:

REGNO PHONE_NO PROVIDER TYPE

PERSON_ID (FK)

29 hyd

rajesh

1

1014 66576155 vodafone

office

2

34 vizasg

suresh

2

1012

9999999 airtel

office

30 hya

karan

3

1020 432432442 airtel

uncommon data

4

1013

5

1015

8888888 vodafone 76576757 jio

office residence office

30 29 (null) 29

uncommon data

30

Venn Diagram

OTM_PERSON (parent)

Tteft)

34 vizag suresh

OTM_PHONENUMBER

child) (right)

29-1013 1020 4324$2442 airtel office 29-1012

---- 1 ----

30-1014 30-1015 --3--

---2---

a) inner join (gives 3 area content)

=>gives common of data both db tables (left side and right side db tables)

b) right join /right outer join (gives 3 + 2 area content)

=>gives common of data both db tables (left side and right side db tables) and also gives uncommon data of right side db table

b) left join /left outer join (gives 1+3 area content)

=>gives common of data both db tables (left side and right side db tables) and also gives uncommon data of left side db table

c) full join /full outer join (gives 1,2,3 content)

=>gives common and uncommon data of both db tables.

to

=>To work with HQL/JPQL Joins we need keep db tables and their entity classes in relationship using association mapping concepts..

Join

HQL/JPQL Query syntax parent to child

==========

parent table/left side table

select <parent class properties>,<child class properties> from <Parent class> <alias name>

<join type> <parent class HAS-A property> <alias name> <addtional conditions> ↓ child table/right side table inner join right join

left join full join

Example

Code IPersonRepo Repository Interface

```
public interface IPersonRepo extends JpaRepository<Person,Integer> {
//@Query("SELECT p.pid,p.pname,p.paddrs,ph.regno,ph.phoneNo,ph.provider,ph.type from Person p inner join p.contactDetails ph") //@Query("SELECT p.pid,p.pname,p.paddrs,ph.regno,ph.phoneNo, ph.provider,ph.type from Person p right join p.contactDetails ph") //@Query("SELECT p.pid,p.pname,p.paddrs, ph.regno,ph.phoneNo,ph.provider,ph.type from Person p left join p.contactDetails ph") @Query("SELECT p.pid,p.pname,p.paddrs,ph.regno,ph.phoneNo,ph.provider,ph.type from Person p full join p.contactDetails ph") public List<Object[]> fetch Data UsingJoinsByParent();
```

Code in Serivce Interface

```
public interface IPersonMgmtService {
public List<Object[]>fetchDataByJoins UsingParent();
}
```

Code in service Impl class

```
@Service("personService")
public class Person MgmtServiceImpl implements IPerson MgmtService {
@Autowired
private IPersonRepo perRepo;
@Autowired
private IPhoneNumberRepo phoneRepo;
@Override
public List<Object[]>fetchDataByJoinsUsingParent() {
return perRepo.fetchData UsingJoinsByParent();
}
```

Code in Runner class

==========

```
@Component
public class Association TestRunner implements CommandLineRunner {
@Autowired
```

```
private IPerson MgmtService service;
@Override
public void run(String... args) throws Exception {
service.fetchDataByJoins UsingParent().forEach(row->{
for(Object val:row) {
System.out.print(val+" ");
}
System.out.println();
});
}//run
}//class
```

Assignment :: write the similar HQL Joins from child to parent

In Child Repository Interface

```
//@Query("select ph.regId,ph.type, ph.provider,ph.phone,p.pid,p.pname,p.paddrs from PhoneNumber ph inner join ph.person p") @Query("select ph.regId,ph.type, ph.provider,ph.phone,p.pid,p.pname,p.paddrs from Phone Number ph full join ph.person p")
public List<Object[]> showJoinsDataChildToParent();
```

In service Interface

==========

```
public List<Object[]> fetchChildToParentJoins Data();
```

In Service Impl class

=======

```
@Override
public List<Object[]>fetchChildToParentJoins Data() {
return phoneRepo.showJoins DataChildToParent();
}
```

In Runner class

==========

```
personService.fetch ChildToParentJoins Data().forEach(row->{
System.out.println(Arrays.toString(row));
});
```

Improved Example App OneToMany BiDirectional Association

==========

BootDataJpaProj13-Association MappingOneToManyBi [boot]

src/main/java

#com.nt

> BootDataJpaProj13AssociationOTOMBIApp.java

com.nt.entity

Department.java

==========

Entity Class (parent)

==========

package com.nt.entity;

Employee.java

#com.nt.repository

>

IDepartmentRepository.java

import java.util.Set;

> IEmployeeRepository.java

com.nt.runners

> OTMBiDiAssociation TestRunner.java

>

com.nt.service

CompanyMgmtServiceImpl.java

> ICompanyMgmtService.java

src/main/resources

src/test/java

>

JRE System Library [JavaSE-17]

> Maven Dependencies

target/generated-sources/annotations

target/generated-test-sources/test-annotations

>

>

src

>

target

HELP.md

mvnw

import jakarta.persistence.CascadeType; import jakarta.persistence.Column;

import jakarta.persistence.Entity; import jakarta.persistence.FetchType; import jakarta.persistence.GeneratedValue; import jakarta.persistence.GenerationType; import jakarta.persistence.Id; import jakarta.persistence.JoinColumn; import jakarta.persistence.OneToMany;

import jakarta.persistence.SequenceGenerator;

import jakarta.persistence.Table;

import lombok.Getter;

```
import lombok.Setter;
@Table(name="JPA_OTM_BI_DEPT")
@Setter
@Getter
@Entity
Entity class child
==========
package com.nt.entity;
import jakarta.persistence.CascadeType;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.FetchType; import jakarta.persistence.GeneratedValue; import
jakarta.persistence.GenerationType; import jakarta.persistence.Id; import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.Table; import lombok.Getter;
import lombok.Setter;
@Entity
@Table(name="JPA_OTM_BI_EMPLOYEE")
@Setter
public class Department {
@SequenceGenerator(name="gen1",sequenceName = "dno_seq",initialValue = 1,allocationSize = 1)
@GeneratedValue(strategy = GenerationType.SEQUENCE,generator = "gen1")
@Id
private Integer dno;
@Column(length = 30)
private String dname;
@Column(length = 30)
private String location;
//for One To Many Association (Collection of HAS-A (child) properties)
@OneToMany (targetEntity = Employee.class, cascade = CascadeType.ALL, orphanRemoval = true, fetch =
FetchType.EAGER, mappedBy = "dept") //@JoinColumn(name = "DEPT_NO", referencedColumnName =
"DNO")
private Set<Employee> emps;
public Department() {
System.out.println("Department:: 0-param constructor::"+this.getClass());
}
//alt+Shift+s,s
@Override
```

```
public String toString() {
return "Department [dno=" + dno + ", dname=" + dname + ", location=" + location + "]";
}
}
@Getter
public class Employee {
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private Integer empno;
@Column(length = 30)
private String ename;
@Column(length = 30)
private String eaddrs;
// build Many One Association (HAS -A property)
@ManyToOne(targetEntity = Department.class,cascade = CascadeType.ALL, fetch = FetchType.EAGER)
@JoinColumn(name="DEPT_NO", referencedColumnName = "DNO")
private Department dept;
public Employee() {
System.out.println("Employee:: 0-param cosntructor::"+this.getClass());
}
}
//alt+shift+s,s (toString())
@Override
public String toString() {
return "Employee [empno=" + empno + ", ename=" + ename + ", eaddrs=" + eaddrs + "]";
}
```

Service Interface

==========

```
package com.nt.service;
import java.util.List;
import java.util.Set;
import com.nt.entity.Department;
import com.nt.entity.Employee;
public interface ICompanyMgmtService {
public String registerDepartment(Department dept); //parent to child saving
public String registerEmployees (Set<Employee> emps); //child to parent saving public List<Department>
showAll DepartmentsAnd Its Employees(); // parent to child Loading public List<Employee>
showAllEmployeesAnd Its Departments(); // child to parent loading public String remove
```

DepartmentAndItsEmployess(int dno); public String removeEmployeesAndTheir Department(List<Integer> empIds); public String add NewEmployee To Department(int dno, Employee emp);

public String

public String

}

removeAllEmployeesOfa Department(int dno);

removeOneEmployeeOfADeparment(int dno,int empno);

package com.nt.service;

Service Impl class

==========

```
import java.util.List;
import java.util.Optional;
import java.util.Set;
import java.util.stream.Collectors;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.nt.entity.Department;
import com.nt.entity.Employee;
import com.nt.repository.IDepartmentRepository;
import com.nt.repository.IEmployee Repository;
@Service("companyService")
public class CompanyMgmtServiceImpl implements ICompanyMgmtService {
@Autowired
private IDepartmentRepository deptRepo;
@Autowired
private IEmployee Repository empRepo;
@Override
public String registerDepartment (Department dept) {
//save the parent and its child objs
int idVal=deptRepo.save(dept).getDno();
return "Dept and the associated employees are saved with the id Value"+idVal;
}
@Override
public String register Employees (Set<Employee> emps) {
//save the childs along with its parent
List<Employee> savedList=empRepo.saveAll(emps);
//capture only ids from the savedList
```

```
}
List<Integer> ids=saved List.stream().map(Employee::getEmpno).collect(Collectors.toList());
return "Employees and the associated Department are saved with the id Values:"+ids;
@Override
//Load parent objects
public List<Department> showAll DepartmentsAndItsEmployees() {
List<Department> deptList=deptRepo.findAll();
return deptList;
}
@Override
public List<Employee> showAllEmployeesAnd Its Departments() {
//Load childs objects
List<Employee> listEmps=empRepo.findAll();
return listEmps;
}
@Override
public String removeDepartmentAndItsEmployess(int dno) {
//Load Parent and its Assoicated child objects
Optional <Department> opt=deptRepo.findById(dno);
if(opt.isPresent()) {
Department dept=opt.get();
deptRepo.delete(dept);
return " Department and its employees are deleted";
}
return "Department is not found";
}
@Override
public String removeEmployeesAndTheirDepartment (List<Integer> empIds) {
//Load the childs by Ids
List<Employee> listChilds=empRepo.findAllById(empIds);
if(listChilds!=null && listChilds.size()!=0) {
empRepo.deleteAll(listChilds);
return " Given Employees and their Dept is deleted";
}
return " Given Employees are not found";
}//method
@Override
```

```
public String add NewEmployeeToDepartment(int dno, Employee emp) {
// Load Department
Optional<Department> opt=deptRepo.findById(dno);
if(opt.isPresent()) {
//get Department (parent)
Department dept=opt.get();
//get collection of childs
Set<Employee> empsSet-dept.getEmps();
//set parent to new child
emp.setDept(dept);
// add new child to parent
empsSet.add(emp);
//save the parent
deptRepo.save(dept);
return "new Employee is added to existing Department";
}
return " Department is not found";
}
@Override
public String removeAllEmployeesOfa Department(int dno) {
// Load Department
Optional <Department> opt=deptRepo.findById (dno);
if(opt.isPresent()) {
//get Department (parent)
Department dept=opt.get();
// get all childs (Employees) of parent
Set<Employee> empsSet-dept.getEmps();
//nully dept from from childs (Employees)
emp.setDept(null);
for(Employee emp:empsSet) {
}
// delete only childs
empRepo.deleteAllInBatch(empsSet);
return " All employees of the Given Department are deleted";
}
return "Department is not found";
}
```

```
@Override
public String removeOneEmployeeOfADeparment (int dno,int eno) {
//
Load
parent
Optional <Department> opt-deptRepo.findById(dno);
//Load chid
Optional <Employee> opt1=empRepo.findById(eno);
if(opt.isPresent() && opt1.isPresent()) {
// get child (Employee)
Employee emp=opt1.get();
//get Department (parent)
Department dept=opt.get();
// remove the link with parent from child
emp.setDept(null);
//remove the child return "One Employee from Department is deleted";
empRepo.deleteAllInBatch (List.of(emp));
return " Given Department or Given Empoyee is not found";
}
}
```

Runner class

```
==========
package com.nt.runners;
import java.util.List;
import java.util.Set;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;
import com.nt.entity.Department;
import com.nt.entity.Employee;
import com.nt.service.ICompanyMgmtService;
@Component
public class OTMBiDiAssociation TestRunner implements CommandLineRunner {
@Autowired
private ICompanyMgmtService compService;
@Override
public void run(String... args) throws Exception {
```

```
/*try
{
//parent object
Department dept=new Department();
dept.setDname("SALES"); dept.setLocation("hyd");
//child objs
Employee emp1=new Employee();
emp1.setEname("raja1");emp1.setEaddrs("hyd");
Employee emp2=new Employee();
emp2.setEname("rajesh1");emp2.setEaddrs("vizag");
//assign parent to childs emp1.setDept(dept); emp2.setDept(dept);
//assign childs to parent
dept.setEmps(Set.of(emp1,emp2));
//invoke service method String msg=compService.registerDepartment(dept);
System.out.println(msg);
}//try
catch(Exception e) {
e.printStackTrace();
}
*/
/*try {
//parent
object
Department dept=new Department();
}
//child objs
}
}
dept.setDname("IT"); dept.setLocation("hyd");
Employee emp1=new Employee();
emp1.setEname("karan");emp1.setEaddrs("hyd");
Employee emp2=new Employee();
emp2.setEname("chinna");emp2.setEaddrs("vizag");
//assign parent to childs
emp1.setDept(dept); emp2.setDept(dept);
//assign childs to parent
Set<Employee> empsSet=new HashSet<Employee>();
```

```
empsSet.add(emp1); empsSet.add(emp2);
dept.setEmps(empsSet);
//invoke Service method
String msg=compService.registerEmployees (empsSet);
System.out.println(msg);
}
catch(Exception e) {
e.printStackTrace();
*/
/*
try {
//invoke the method
List<Department> deptList-compService.showAll DepartmentsAnditsEmployees();
deptList.forEach(dept->{
System.out.println("Depart (parent) ::" +dept);
Set<Employee> childs=dept.getEmps();
if(childs!=null) {
childs.forEach(emp->{
System.out.println("Employee (child)::"+emp);
});
System.out.println("-
}
});
}//try
catch(Exception e) {
e.printStackTrace();
}*/
1*
try {
");
//invoke service method
List<Employee> listEmps=compService.showAllEmployeesAnd Its Departments();
listEmps.forEach(emp->{
System.out.println("Employee (Child) ::"+emp);
//get depart of a employee
Department dept=emp.getDept();
System.out.println("Department (parent::" +dept);
```

```
System.out.println("-
-");
});
}//try
catch(Exception e) {
e.printStackTrace();
}*/
/*
try {
String msg=compService.removeDepartmentAndItsEmployess(1);
System.out.println(msg);
}
catch(Exception e) {
e.printStackTrace();
}
*/
/*try {
String msg=compService.removeEmployeesAndTheirDepartment(List.of(102,103));
System.out.println(msg);
}
catch(Exception e) {
e.printStackTrace();
}
*/
/*try {
Employee emp1=new Employee();
emp1.setEname("mahesh"); emp1.setEaddrs("hyd");
String msg=compService.addNewEmployeeToDepartment(2, emp1);
System.out.println(msg);
}
catch(Exception e) {
e.printStackTrace();
/*
try {
String msg=compService.removeAllEmployeesOfa Department(2);
System.out.println(msg);
}
```

```
catch(Exception e) {
e.printStackTrace();
}*/
try {
}
String msg=compService.removeOneEmployeeOfADeparment(2, 102);
System.out.println(msg);
catch(Exception e) {
e.printStackTrace();
```