**Scenarios: Fetchtype and switch FestMode**

**Setup of database:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **company\_users** | table to hold the references between the tables of table\_companies and table\_employees | | | |
| **Column** | **type** | **null** | **link to** |  |
| cpy\_ID | bigint(20) | **no** | table\_companies -> cpy\_ID | PRIMARY. BTREE, UNIQUE, NOT NULL |
| emp\_ID | bigint(20) | **no** | table\_employees -> emp\_ID | PRIMARY. BTREE, UNIQUE, NOT NULL |

|  |  |  |  |
| --- | --- | --- | --- |
| **id\_gen** |  |  |  |
| **Column** | **type** | **null** | **Properties** |
| GEN\_KEY | varchar(255) | **no** | PRIMARY, BTREE, UNIQUE,  NOT NULL |
| GEN\_VALUE | bigint(20) | **yes** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **table\_companies** |  |  |  |
| **Column** | **type** | **null** | **Properties** |
| cpy\_ID | bigint(20) | **no** | PRIMARY,BTREE, UNIQUE,NOT NULL |
| cpy\_NAME | varchar(255) | **yes** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **table\_employees** |  |  |  |
| **Column** | **type** | **null** | **Properties** |
| empy\_ID | bigint(20) | **no** | PRIMARY,BTREE, UNIQUE,NOT NULL |
| emp\_LASTNAME | varchar(255) | **yes** |  |

|  |  |  |
| --- | --- | --- |
| **Database records created for the tests:** | | |
| |  |  | | --- | --- | | **Table: table\_companies** | | | **cpy\_ID** | **cpy\_NAME** | | 1 | SOBIS Software GmbH | | 2 | SOBIS Software Pvt. Ltd. | | 3 | SOBIS K.S.A. | | |  |  | | --- | --- | | **Table: company\_users** | | | **cpy\_ID** | **emp\_ID** | | **1 ->** | **1 ->** | | 2 -> | 2 -> | | **1 ->** | **3 ->** | | 3 -> | 4 -> | | **1 ->** | **5 ->** | | 2 -> | 6 -> | | **1 ->** | **7 ->** | | 3 -> | 8 -> | | **1 ->** | **9 ->** | | 2 -> | 10 -> | | |  |  | | --- | --- | | **Table: table\_employees** | | | **emp\_ID** | **emp\_LASTNAME** | | 1 | Employee Number 101 | | 2 | Employee Number 102 | | 3 | Employee Number 103 | | .... | ... | | 10 | Employee Number 110 | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Annotations**  **Company.java and Employee.java** | **Set criteria to get 1 company with different FetchTypes** | **Commit and access list**  the printList functions prints first the information in the company class only !  Then it tries to print information from the employees table where a join is required, which should only be available - when the commit statement has not been executed - when the Fetchtype is set to EAGER.  **//this print statement should always work**  printMessage("Info:", "# Get only the company name:" + "# Company Name: " + company.getCompanyName());  **//this print statement will sometimes cause an error because of the company.getEmployees()!**  printMessage("Info","# Get the company and the accociated employees: " + company.getCompanyName() + " " + **company.getEmployees()**); | **results in** | **Comment** |
| **Company.java**  **@OneToMany(fetch=FetchType.*LAZY*)**  **Employee.java**  **@ManyToOne(cascade=CascadeType.*ALL*, fetch=FetchType.*LAZY*)** | Criteria crit = *session*  .createCriteria(Company.**class**)  .setFetchMode(Company.***FIELD\_COMPANY\_NAME***, FetchMode.***SELECT***) //**LAZY**  .add(Restrictions.*eq*(Company.***FIELD\_ID***,  (**long**) 1)); | // print list before commit  dbFunction.printList(list);  tx.commit();  // print list after commit  **Note: Session is still open when accessing information which needs a join. Joining data is reloaded automatically then.** | Records found: **1**  Records found: **[annotations.Company@578524c3]**  ##################################  **# Get only the company name:# Company Name: SOBIS Software GmbH**  Hibernate:  select  employees0\_.cpy\_ID as cpy\_ID1\_0\_0\_,  employees0\_.emp\_ID as emp\_ID2\_0\_0\_,  employee1\_.emp\_ID as emp\_ID1\_2\_1\_,  employee1\_.emp\_LASTNAME as emp\_LAST2\_2\_1\_,  employee1\_1\_.cpy\_ID as cpy\_ID1\_0\_1\_  from  company\_users employees0\_  inner join  table\_Employees employee1\_  on employees0\_.emp\_ID=employee1\_.emp\_ID  left outer join  company\_users employee1\_1\_  on employee1\_.emp\_ID=employee1\_1\_.emp\_ID  where  employees0\_.cpy\_ID=?  **# Get the company and the accociated employees: SOBIS Software GmbH [annotations.Employee@61a002b1, annotations.Employee@3aa3193a, annotations.Employee@3dd69f5a, annotations.Employee@59a67c3a, annotations.Employee@1ee4730]**  ----------------------------------  ----------------------------------  # Employee sql id: 1  # Employee last name: Employee Number 101  ----------------------------------  # Employee sql id: 5  # Employee last name: Employee Number 105  ----------------------------------  # Employee sql id: 3  # Employee last name: Employee Number 103  ----------------------------------  # Employee sql id: 9  # Employee last name: Employee Number 109  ----------------------------------  # Employee sql id: 7  # Employee last name: Employee Number 107 | **this is working as designed !**  because the list is printed before the commit has been executed it will execute the  **company.getEmployees()** |
| **Company.java**  **@OneToMany(fetch=FetchType.*LAZY*)**  **Employee.java**  **@ManyToOne(cascade=CascadeType.*ALL*, fetch=FetchType.*LAZY*)** | Criteria crit = *session*  .createCriteria(Company.**class**)  .setFetchMode(Company.***FIELD\_COMPANY\_NAME***, FetchMode.***SELECT***) //**LAZY**  .add(Restrictions.*eq*(Company.***FIELD\_ID***,  (**long**) 1)); | tx.commit();  // print list after commit  dbFunction.printList(list);  **Note: Session is closed before accessing information which needs a join. Lazy fetchtype is selected and fetchmode is lazy as well.** | Records found: 1  Records found: [annotations.Company@578524c3]  ##################################  # Get only the company name:# Company Name: SOBIS Software GmbH  **Hibernate exception !failed to lazily initialize a collection of role: annotations.Company.employees, could not initialize proxy - no Session** | **this is working as designed !**  **Hibernate exception !failed to lazily initialize a collection of role: annotations.Company.employees, could not initialize proxy - no Session** |
| **Company.java**  **@OneToMany(fetch=FetchType.*EAGER*)**  **Employee.java**  **@ManyToOne(cascade=CascadeType.*ALL*, fetch=FetchType.*EAGER*)** | Criteria crit = *session*  .createCriteria(Company.**class**)  .setFetchMode(Company.***FIELD\_COMPANY\_NAME***, FetchMode.***SELECT***) //**LAZY**  .add(Restrictions.*eq*(Company.***FIELD\_ID***,  (**long**) 1)); | **// print list before commit**  dbFunction.printList(list);  tx.commit();  // print list after commit  **Note: Session is still open when accessing information which needs a join. Joining data is reloaded automatically then, even if the FetchMode is set to LAZY.** | Hibernate:  /\* criteria query \*/ select  this\_.cpy\_ID as cpy\_ID1\_1\_1\_,  this\_.cpy\_NAME as cpy\_NAME2\_1\_1\_,  employees2\_.cpy\_ID as cpy\_ID1\_0\_3\_,  employee3\_.emp\_ID as emp\_ID2\_0\_3\_,  employee3\_.emp\_ID as emp\_ID1\_2\_0\_,  employee3\_.emp\_LASTNAME as emp\_LAST2\_2\_0\_,  employee3\_1\_.cpy\_ID as cpy\_ID1\_0\_0\_  from  table\_Companies this\_  left outer join  company\_users employees2\_  on this\_.cpy\_ID=employees2\_.cpy\_ID  left outer join  table\_Employees employee3\_  on employees2\_.emp\_ID=employee3\_.emp\_ID  left outer join  company\_users employee3\_1\_  on employee3\_.emp\_ID=employee3\_1\_.emp\_ID  where  this\_.cpy\_ID=?  ##################################  Records found: **5**  Records found: [annotations.Company@3dd69f5a, annotations.Company@3dd69f5a, annotations.Company@3dd69f5a, annotations.Company@3dd69f5a, annotations.Company@3dd69f5a]  ##################################  **Below output appears 5 \* in the loop because of the sql statement created, which is not required but currently so coded.**  # Get only the company name:# Company Name: SOBIS Software GmbH  # Get the company and the accociated employees: SOBIS Software GmbH [annotations.Employee@ca27722, annotations.Employee@1ac85b0c, annotations.Employee@70ab80e3, annotations.Employee@23a9ba52, annotations.Employee@6bc248ed]  ----------------------------------  ----------------------------------  # Employee sql id: 7  # Employee last name: Employee Number 107  ----------------------------------  # Employee sql id: 1  # Employee last name: Employee Number 101  ----------------------------------  # Employee sql id: 9  # Employee last name: Employee Number 109  ----------------------------------  # Employee sql id: 5  # Employee last name: Employee Number 105  ----------------------------------  # Employee sql id: 3  # Employee last name: Employee Number 103  # Get only the company name:# Company Name: | **this is working as designed !**  **because the employees information is reloaded because the session is still open.** |
| **Company.java**  **@OneToMany(fetch=FetchType.*EAGER*)**  **Employee.java**  **@ManyToOne(cascade=CascadeType.*ALL*, fetch=FetchType.*EAGER*)** | Criteria crit = *session*  .createCriteria(Company.**class**)  .setFetchMode(Company.***FIELD\_COMPANY\_NAME***, FetchMode.***SELECT***) //**LAZY**  .add(Restrictions.*eq*(Company.***FIELD\_ID***,  (**long**) 1)); | tx.commit();  // print list after commit  dbFunction.printList(list);  **Note: Session is closed before accessing information which needs a join. Eager fetchtype is selected and fetchmode will be reduced to lazy which is NOT the result !!** | Hibernate:  /\* criteria query \*/ select  this\_.cpy\_ID as cpy\_ID1\_1\_1\_,  this\_.cpy\_NAME as cpy\_NAME2\_1\_1\_,  employees2\_.cpy\_ID as cpy\_ID1\_0\_3\_,  employee3\_.emp\_ID as emp\_ID2\_0\_3\_,  employee3\_.emp\_ID as emp\_ID1\_2\_0\_,  employee3\_.emp\_LASTNAME as emp\_LAST2\_2\_0\_,  employee3\_1\_.cpy\_ID as cpy\_ID1\_0\_0\_  from  table\_Companies this\_  left outer join  company\_users employees2\_  on this\_.cpy\_ID=employees2\_.cpy\_ID  left outer join  table\_Employees employee3\_  on employees2\_.emp\_ID=employee3\_.emp\_ID  left outer join  company\_users employee3\_1\_  on employee3\_.emp\_ID=employee3\_1\_.emp\_ID  where  this\_.cpy\_ID=?  Records found: 5  Records found: [annotations.Company@3dd69f5a, annotations.Company@3dd69f5a, annotations.Company@3dd69f5a, annotations.Company@3dd69f5a, annotations.Company@3dd69f5a]  ##################################  **Below output appears 5 \* in the loop because of the sql statement created, which is not required but currently so coded.**  # Get only the company name:# Company Name: SOBIS Software GmbH  # Get the company and the accociated employees: SOBIS Software GmbH [annotations.Employee@ca27722, annotations.Employee@1ac85b0c, annotations.Employee@70ab80e3, annotations.Employee@23a9ba52, annotations.Employee@6bc248ed]  ----------------------------------  ----------------------------------  # Employee sql id: 7  # Employee last name: Employee Number 107  ----------------------------------  # Employee sql id: 1  # Employee last name: Employee Number 101  ----------------------------------  # Employee sql id: 9  # Employee last name: Employee Number 109  ----------------------------------  # Employee sql id: 5  # Employee last name: Employee Number 105  ----------------------------------  # Employee sql id: 3  # Employee last name: Employee Number 103 | **This is a problem !!! Reset from FetchType.EAGER to use a criteria with FetchMode.SELECT (LAZY) is not working !!**  The default mode in the classes are set to FetchType.EAGER but in the criteria it is reset to FetchMode.SELECT (LAZY) which should read only the company class in my example without joining the employees of the company.  **Reducing to FetchMode.Lazy should return only the company not also the employees assigned to the company.** |
| **Company.java**  **@OneToMany(fetch=FetchType.*LAZY*)**  **Employee.java**  **@ManyToOne(cascade=CascadeType.*ALL*, fetch=FetchType.*LAZY*)** | Criteria crit = *session*  .createCriteria(Company.**class**)  .setFetchMode(Company.***FIELD\_COMPANY\_NAME***, FetchMode.***JOIN***) //**EAGER**  .add(Restrictions.*eq*(Company.***FIELD\_ID***,  (**long**) 1)); | tx.commit();  // print list after commit  dbFunction.printList(list); | Records found: 1  Records found: [annotations.Company@578524c3]  ##################################  # Get only the company name:# Company Name: SOBIS Software GmbH  **Hibernate exception !failed to lazily initialize a collection of role: annotations.Company.employees, could not initialize proxy - no Session** | **This is a problem !!! Reset from FetchType.LAZY to use a criteria with FetchMode.JOIN (EAGER) is also not working !!**  **Switch from Lazy to Eager is also not working outside of the session commit !!!**  **Hibernate exception !failed to lazily initialize a collection of role: annotations.Company.employees, could not initialize proxy - no Session**  **Extending to FetchMode.EAGER should return not only the company but also the employees assigned to the company.** |

**printlist executes**

printMessage("Info:", "# Get only the company name:" + "# Company Name: " + company.getCompanyName());

printMessage("Info","# Get the company and the accociated employees: " + company.getCompanyName() + " " + **company.getEmployees()**);