MappedBy in bi-directional @ManyToMany : what is the reason

Question:

1. What is the reason for setting MappedBy in bidirectional many-to-many relationships?

2. When one table has significant amount of records, while other has a few, which side is better to put **mappedBy**

Answer:

It's actually a good question, and it helps to understand the concept of an "owning" entity. If you want to prevent both sides (in a bidirectional relationship) from having join tables, a good idea, then you need to have a **mappedBy=** element on one side.

Whether or not there is a join table is controlled by the **mappedBy="name"** element of the `@ManyToMany` annotation. The Javadoc for **mappedBy** for the **ManyToMany** annotation says:

> The field that owns the relationship. Required unless the relationship is unidirectional.

For your (bidirectional) example, if there were only two `@ManyToMany` annotations and no **mappedBy=** element, the default will have two Entity tables and two Join Tables:

```
Hibernate: create table SideA (id bigint not null, primary key (id)) Hibernate: create table
SideA_SideB (sidea_id bigint not null, sidebs_id bigint not null, primary key (sidea_id, sidebs_id))
Hibernate: create table SideB (id bigint not null, primary key (id)) Hibernate: create table
SideB_SideA (sideb_id bigint not null, sideas_id bigint not null, primary key (sideb_id, sideas_id))
```

While this is saying that each Entity "owns" its **ManyToMany** relationship, the extra join table is redundant in the typical use case, and the Javadoc says you need a **mappedBy** annotation. If I decide to have SideA "own" the relationship, then I add the **mappedBy**= element to the SideB entity to specify that it doesn't own the relationship:

```
@Entity
public class SideA {
     @ManyToMany
     Set<SideB> sidebs;
}
@Entity
public class SideB {
     @ManyToMany(mappedBy="sidebs")
     Set<SideA> sideas;
}
```

Since the SideB entity no longer owns its **ManyToMany** relationship, the extra JoinTable will not be created:

```
Hibernate: create table SideA (id bigint not null, primary key (id)) Hibernate: create table SideB
(id bigint not null, primary key (id)) Hibernate: create table SideA_SideB (sideas_id bigint not
null, sidebs_id bigint not null, primary key (sideas_id, sidebs_id))
```

This is important to the developer because he or she must understand that no relationship is persisted unless it's added to the owning entity, in this case the SideA entity. However, since this a bidirectional relationship, the developer should be adding both a SideA to SideB.sideas and a SideB to SideA.sidebs anyway.

So, if you have a bidirectional **ManyToMany** relationship, which means you have **ManyToMany** on both entities involved, then you should add a **mappedBy**="name" on one of them as per the Javadoc and to avoid having a redundant join table. Since it's bidirectional, I don't think it matters which side you make the owning entity - it depends more your application preferences. As always, it's always a good idea to enable the sql logs and see what's going on in the database.