

Hibernate Envers

Easy Entity Auditing

Auditando suas classes de persistência com Hibernate Envers

Castro (@CastroAlexandre)

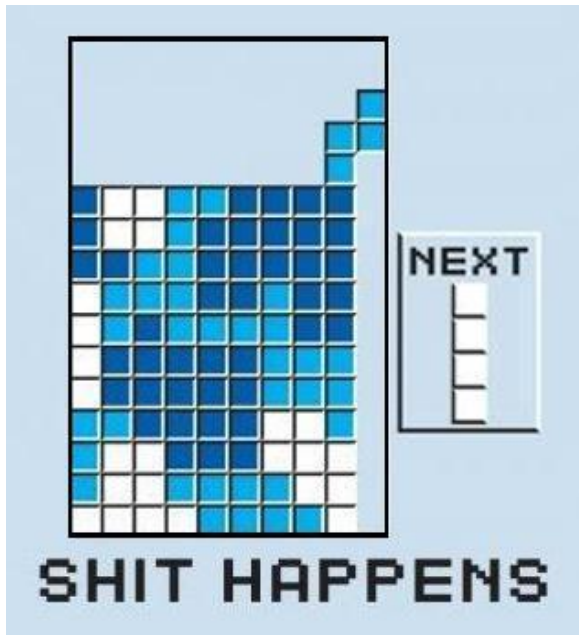
Consultor (Summa) e Instrutor (Globalcode)

SCJP, SCWCD, SCBCD, SCEA-I, SCSNI



Motivação

- ✓ Rastrear o histórico de mudanças de entidades para saber quem, quando e o que foi alterado.



Hibernate Envers

- ✓ API para auditar classes de persistência
- ✓ Baseado no conceito de revisões similar ao Subversion
- ✓ Cada transação gera uma revisão

✿ ANNOUNCEMENT: Envers is now part of Hibernate. [Hide Details](#)

From Hibernate 3.5, Envers is included as a Hibernate core module.

For downloads, please go to the [Hibernate downloads page](#). Envers is also available in JBoss Maven repositories.

Documentation is hosted on the [Hibernate docs site](#).

This page won't be updated in the future. All new releases, documentation updates etc. will be done through the [Hibernate site](#).

You can track Envers updates via the [Envers blog](#).

by envers at Tue Oct 12 05:53:39 EDT 2010

JBoss
Community



Configuração

- ✓ Configure os event listeners do Hibernate Envers no `persistence.xml`
- ✓ Anote as classes ou os atributos da classe com `@Audited`



Configuração no persistence.xml

```
<property
  name="hibernate.ejb.event.post-insert"
  value="org.hibernate.ejb.event.EJB3PostInsertEventListener,
        org.hibernate.envers.event.AuditEventListener" />
<property
  name="hibernate.ejb.event.post-update"
  value="org.hibernate.ejb.event.EJB3PostUpdateEventListener,
        org.hibernate.envers.event.AuditEventListener" />
<property
  name="hibernate.ejb.event.post-delete"
  value="org.hibernate.ejb.event.EJB3PostDeleteEventListener,
        org.hibernate.envers.event.AuditEventListener" />
<property
  name="hibernate.ejb.event.pre-collection-update"
  value="org.hibernate.envers.event.AuditEventListener" />
<property
  name="hibernate.ejb.event.pre-collection-remove"
  value="org.hibernate.envers.event.AuditEventListener" />
<property
  name="hibernate.ejb.event.post-collection-recreate"
  value="org.hibernate.envers.event.AuditEventListener" />
```

Configuração na classe

```
@Entity
@Audited
public class City
{
    @Id
    @GeneratedValue
    private int id;
    private String name;
    @ManyToOne
    private State state;

    // CONSTRUCTORS
    // GETTERS AND SETTERS
    // ETC...
}
```

```
@Entity
public class City
{
    @Id
    @GeneratedValue
    private int id;
    @Audited
    private String name;
    @Audited
    @ManyToOne
    private State state;

    // CONSTRUCTORS
    // GETTERS AND SETTERS
    // ETC...
}
```

As tabelas `City_AUD` e `State_AUD` serão criadas automaticamente para armazenar o histórico de mudanças das entidades `City` e `State` respectivamente.

Configuração na classe

- ✓ O nome da tabela que guarda o histórico de mudanças pode ser alterado com o uso da anotação `@AuditTable`
- ✓ Ou através de propriedades no `persistenc.xml` que definem o prefixo e o sufixo do nome da tabela.

```
@Entity
@Audited
@AuditTable( "TB_City_AUDIT" )
public class City
{
    @Id
    @GeneratedValue
    private int id;
    private String name;
    @ManyToOne
    private State state;

    // CONSTRUCTORS
    // GETTERS AND SETTERS
    // ETC...
}
```

```
<property
    name="org.hibernate.envers.audit_table_prefix"
    value="TB_" />
<property
    name="org.hibernate.envers.audit_table_suffix"
    value="_AUDIT" />
```

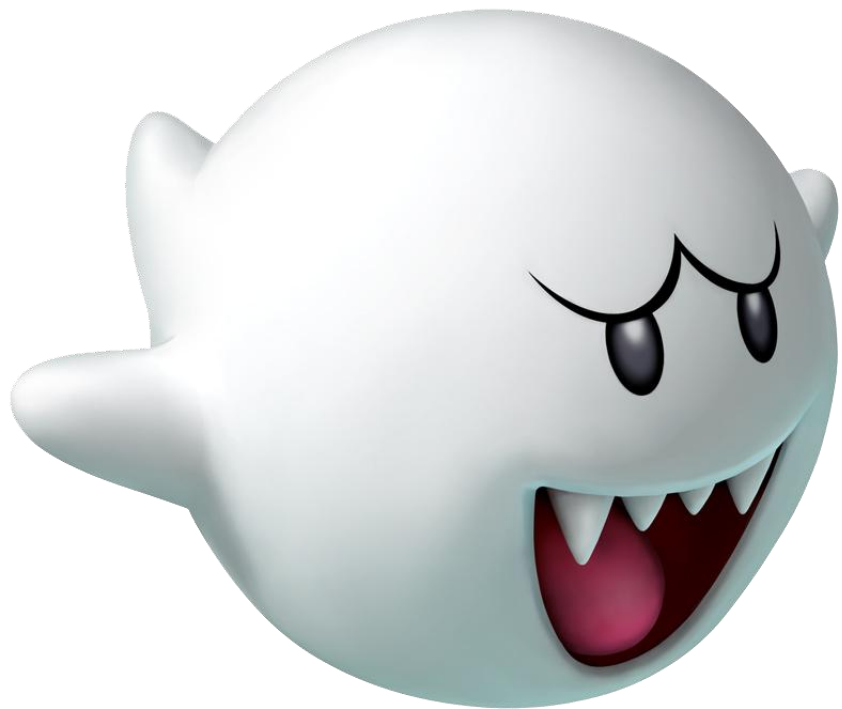
O que pode ser anotado com `@Audited`

- ✓ Tipos simples (Integer, String, Date, etc)
- ✓ Coleções
- ✓ Mapas de tipos simples
- ✓ Relacionamentos
- ✓ Componentes
- ✓ Custom Types



@Audited

DEMO



Adicionando metadados nas revisões

✓ Para logar dados adicionais em uma revisão crie uma classe persistente e anote com `@RevisionEntity`, depois:

Opção 1 : Defina pelo menos dois atributos nesta classe:
+ `id` : do tipo `int` ou `long` anotado com `@RevisionNumber`
+ `date` : do tipo `long` ou `java.util.Date` anotado com `@RevisionTimestamp`

Opção 2 : Simplesmente faça a classe estender `org.hibernate.envers.DefaultRevisionEntity`

✓ Crie um `org.hibernate.envers.RevisionListener` e passe a classe como parâmetro da anotação `@RevisionEntity`

Configuração da RevisionEntity

```
@Entity
@RevisionEntity( MyRevisionListener.class )
public class MyRevisionEntity extends DefaultRevisionEntity
{
    private String user;

    public String getUser() {
        return user;
    }
    public void setUser( final String user ) {
        this.user = user;
    }
}
```

```
public class MyRevisionListener implements RevisionListener
{
    public void newRevision( final Object revisionEntity ) {
        ( (MyRevisionEntity) revisionEntity ).setUser( "Castro" );
    }
}
```

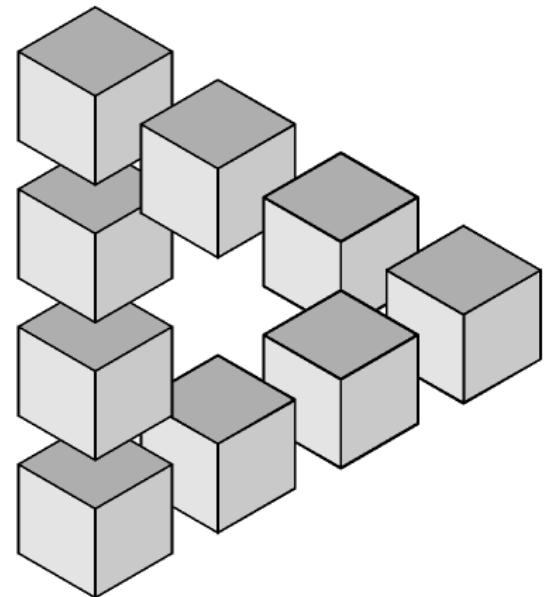
Recuperando revisões

O histórico de mudanças tem **duas dimensões**:

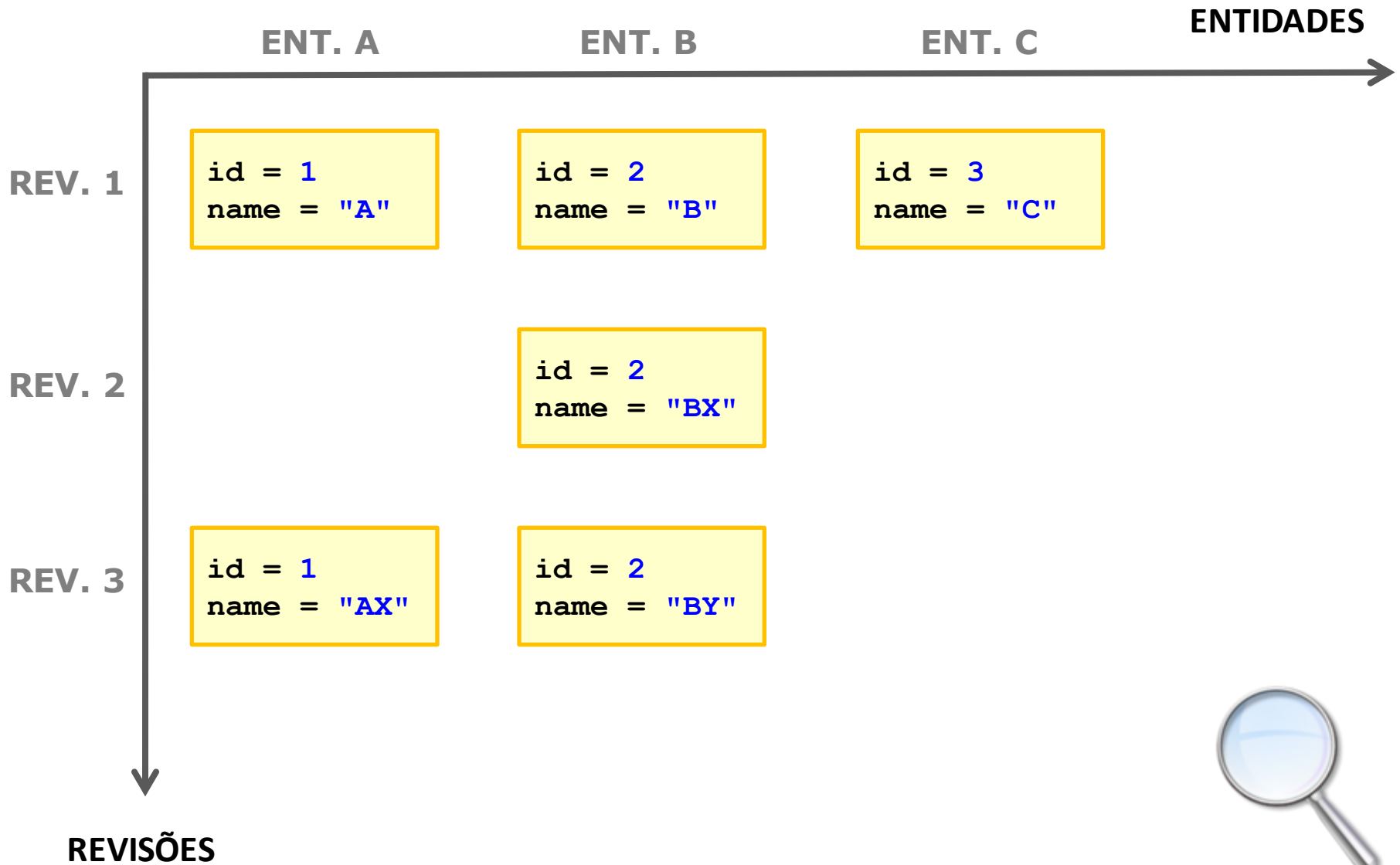
- A primeira (Horizontal) é o estado do banco de dados em uma dada revisão.
- A segunda (Vertical) são revisões de cada entidade alterada.

O que pode ser recuperado:

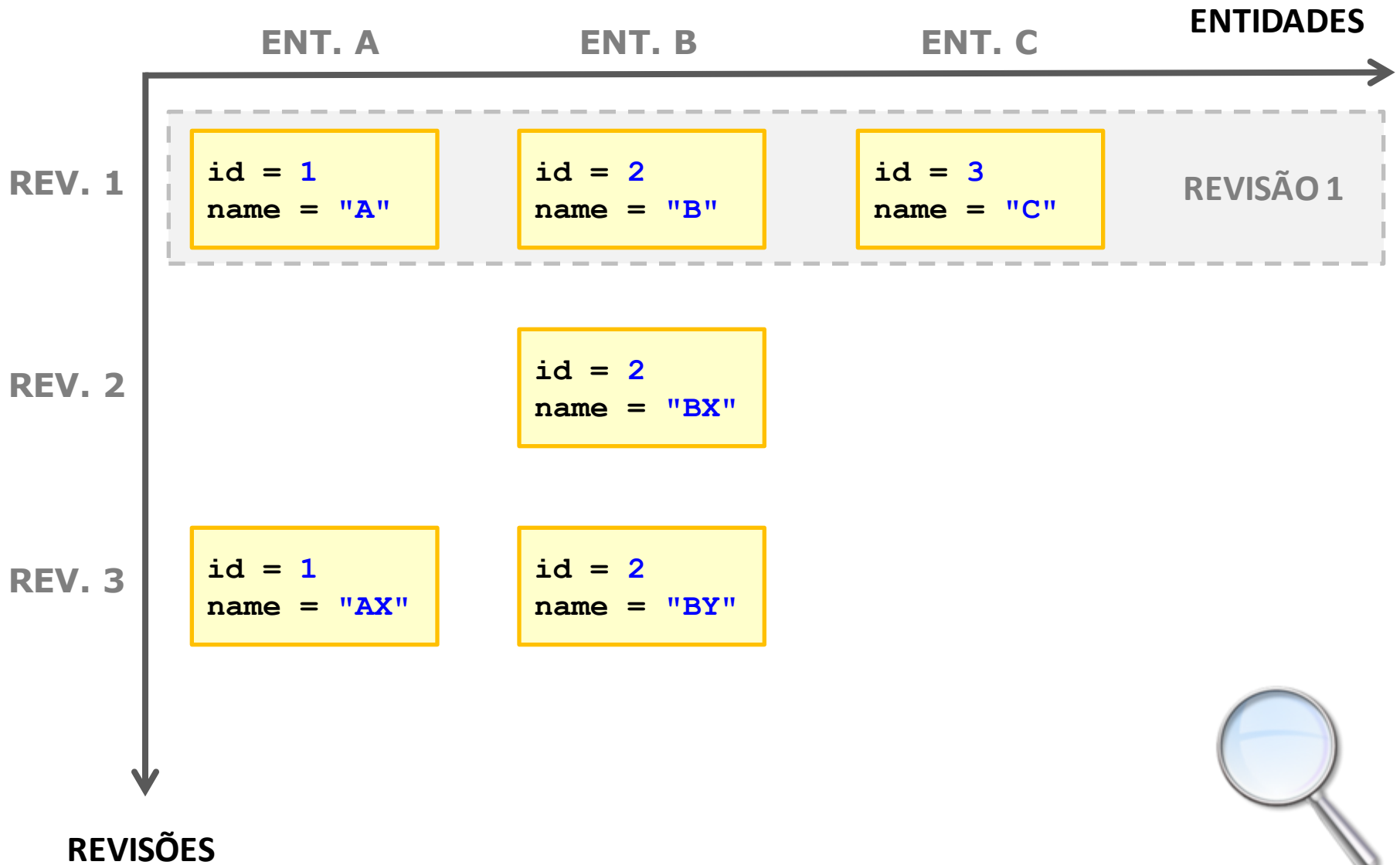
- Revisões da Entidade
- Entidades da Revisão



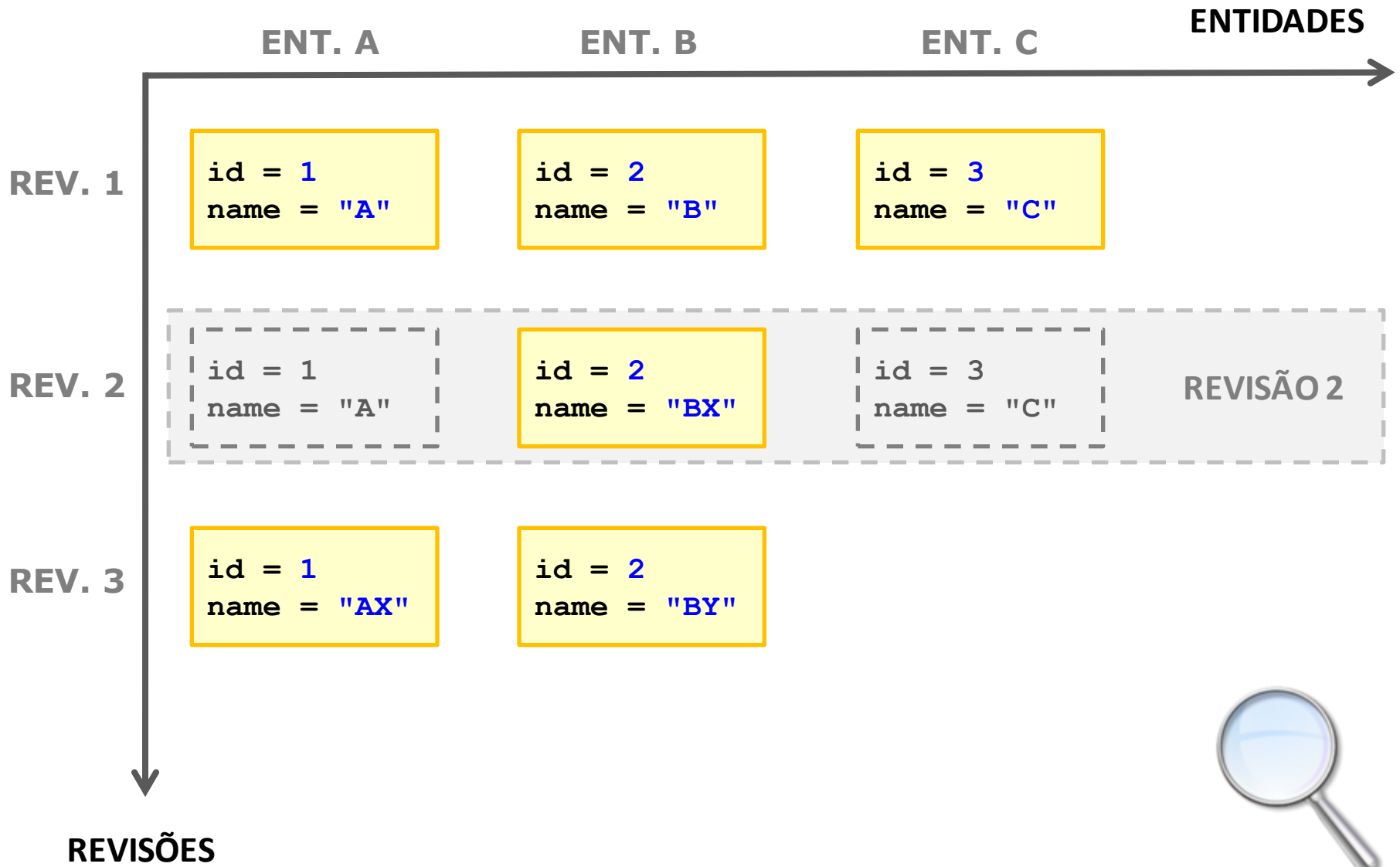
Entidades e Revisões



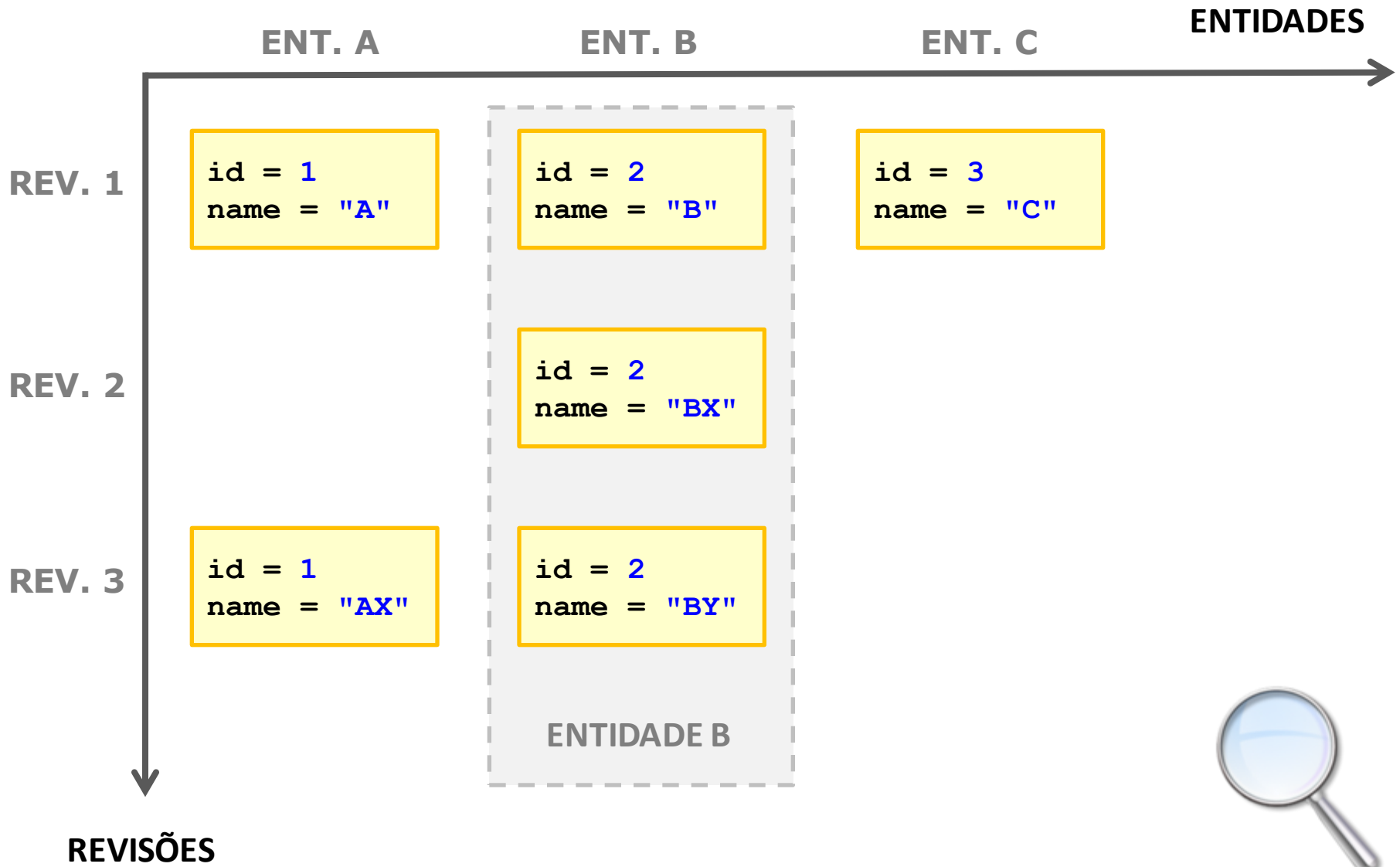
Entidades da Revisão



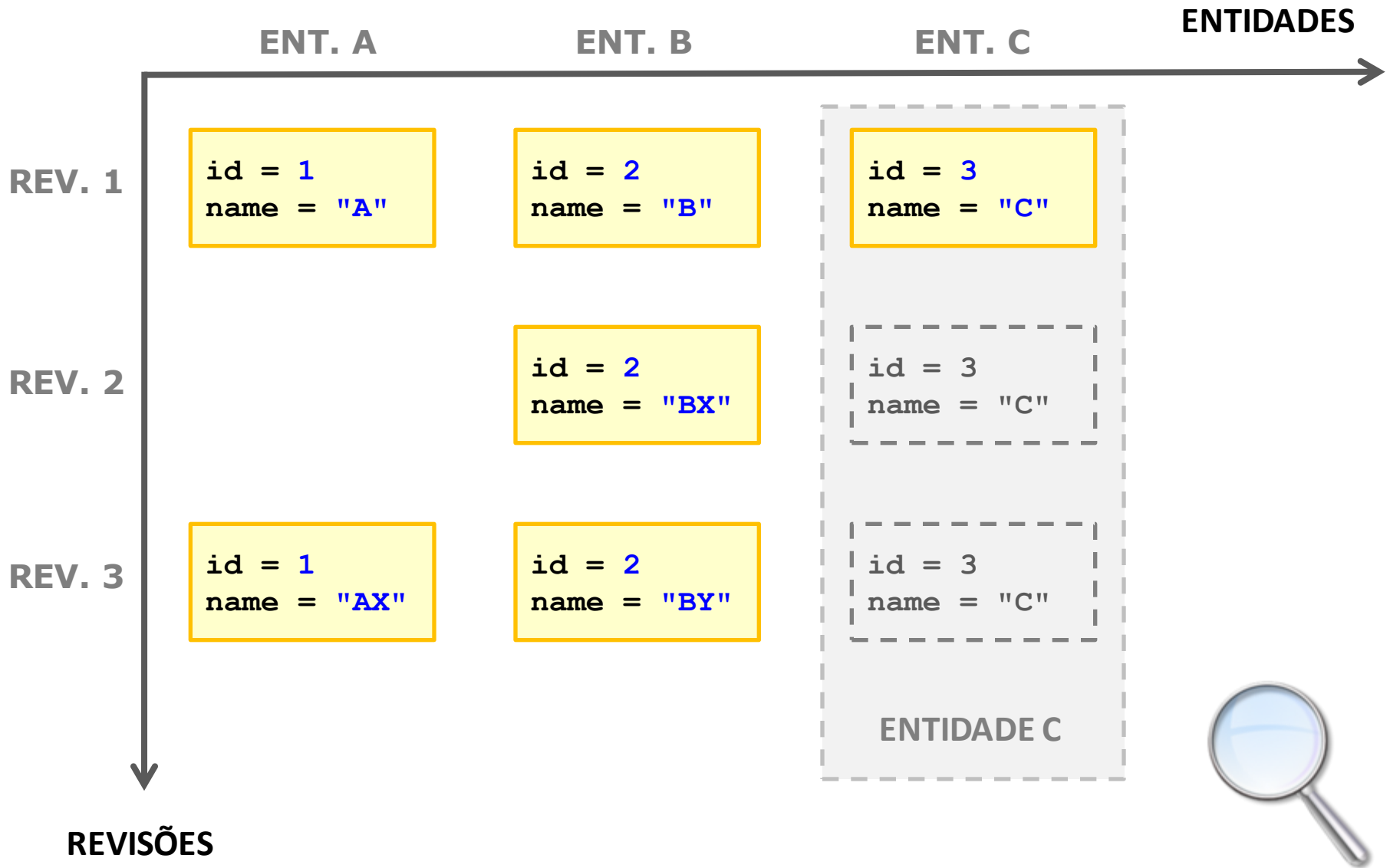
Entidades da Revisão



Revisões da Entidade



Revisões da Entidade



Recuperando Revisões

- ✓ O histórico de mudanças pode ser recuperado através de **queries** semelhantes ao **Hibernate Criteria**
- ✓ A criação das queries é realizada através de um `org.hibernate.envers.AuditReader`

```
AuditReader reader = AuditReaderFactory.get( entityManager );
```



Recuperando Revisões

```
City oldCity =  
    getAuditReader.find( City.class, cityId, revision );
```

- ✓ Recupera a entidade `City` com o `id` especificado na revisão informada, se a entidade não existir na revisão informada retorna `null`, se apenas algumas propriedades foram anotadas com `@Audited` apenas os dados dessas propriedades serão populados as demais propriedades não auditadas recebem `null`.



Recuperando Revisões

```
Number revisionCount =  
    (Number) getAuditReader()  
        .createQuery()  
        .forRevisionsOfEntity( City.class, false, true )  
        .setProjection( AuditEntity.revisionNumber().count() )  
        .getSingleResult();
```

- ✓ Recupera a quantidade de revisões da entidade `City`.



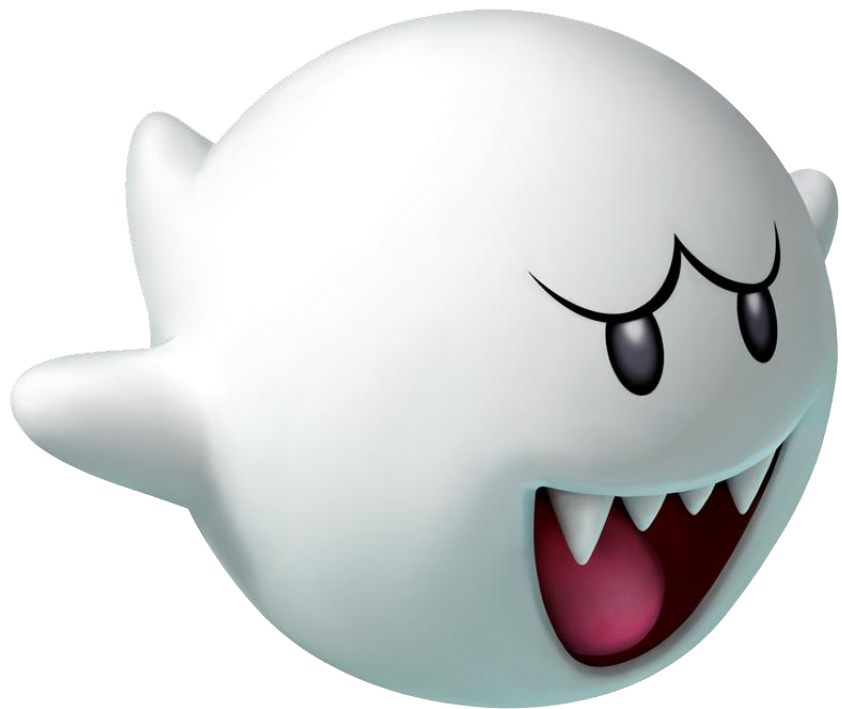
Recuperando Revisões

```
List<Object> result =  
    getAuditReader()  
        .createQuery()  
        .forRevisionsOfEntity( City.class, false, true )  
        .add( AuditEntity.property( "name" ).eq( "X" ) )  
        .getResultList();
```

✓ Recupera as revisões da entidade `City` cujo nome da cidade é "X".



DEMO



Resumindo

Hibernate Envers...

- ... não é intrusivo
- ... é transparente
- ... é baseado em revisões
- ... é fácil de usar
- ... permite adicionar metadados para cada revisão
- ... permite pesquisar o histórico de mudanças através de critérios

Q&A



Referências

<http://www.jboss.org/envers>

<http://download.jboss.org/envers/envers-1.2.2.ga-hibernate-3.3.pdf>

<http://docs.jboss.org/envers/api-new/index.html>

<http://jazon.com/portals/0/Content/ArchivWebsite/jazon.com/jazon09/download/presentations/6220.pdf>

THANKS!



**Castro (@CastroAlexandre)
alexandre.s.castro@gmail.com**