Mapping natural identifier with @NaturalId

Define an attribute as a natural id

The only thing you have to do to model an attribute is a natural id, is to add the *@NaturalId* annotation. You can see an example in the following code snippet.

Natural IDs are immutable by default and you should not provide setter methods for them. If you need mutable, natural identifier, you have to set the mutable attribute of the @NaturalId annotation to true.

```
@Entity
public class Book {


  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  @Column(name = “id”, updatable = false, nullable = false)
  private Long id;


  @NaturalId
  private String isbn;


  …
}
```

Get an entity by its natural id

Hibernate's Session interface provides the methods *byNaturalId* and *bySimpleNaturalId* to read an entity by its natural identifier from the database.

The call of the *using* method provides the name of the natural ID attribute and its value. If the natural ID consists of multiple attributes, you have to call this method multiple times to define each part of the ID.

After you've provided the value of the natural id, you can call the load method to get the entity identified by it.

```
EntityManager em = emf.createEntityManager();
em.getTransaction().begin();
Session session = em.unwrap(Session.class);

Book b = session.byNaturalId(Book.class)
                .using(Book_.isbn.getName(), "978-0321356680")
                .load();
```

The *bySimpleNaturalId* method provides a convenient option to select entities with simple natural IDs that consist of only one attribute. As you can see in the following code snippet, you can provide the natural ID value directly to the load method and don't need to call the using method.

```
EntityManager em = emf.createEntityManager();
em.getTransaction().begin();
Session session = em.unwrap(Session.class);

Book b = session.bySimpleNaturalId(Book.class)
                        .load("978-0321356680");
```

3 Options to retrieve the entity

Method	Description
load()	Gets a reference to the initialized entity.
loadOptional()	Gets a reference to the initialized entity or null and wraps it into an Optional. I explained Hibernate's Optional support in more detail in How to use Java 8's Optional with Hibernate.
getReference	Gets a reference to the entity or an uninitialized proxy.

Locking

The interfaces *NaturalIdLoadAccess* and *SimpleNaturalIdLoadAccess* provide the *with(LockOptions lock)* method. You probably know it from the *IdentifierLoadAccess* interface which gets returned by the *Session.byId(Class entity)* method. You can use this method to define which lock mode Hibernate shall use for the query.

```
EntityManager em = emf.createEntityManager();
em.getTransaction().begin();
Session session = em.unwrap(Session.class);

Book b = session.bySimpleNaturalId(Book.class)
                        .with(LockOptions.UPGRADE)
                        .load("978-0321356680");
```