Logic: Database or Application/2 (constraints check)

Asked 16 years, 2 months ago Modified 15 years, 5 months ago

Viewed 1k times

9

This is a specific version of this question.
I want to check if I am inserting a duplicate row. Should I check it programmatically in my application layer:

```
if (exists(obj))
{
    throw new DuplicateObjectException();
}
HibernateSessionFactory.getSession().save(obj);
```

or should I catch the exception thrown by the database layer and triggered when I violate the contraint?

```
try
{

HibernateSessionFactory.getSession().save(obj);
}
catch(ConstraintViolationException e)
{
    throw new DuplicateObjectException();
}
```

EDIT: In other words: though the constraint is there to remain (it's good database design anyway, and I can't be sure my app will be the only one accessing the table) shall I rely on the constraint and handle the exception its violation will raise, or I'd better check anyway?

EDIT2: Of course I do check+insert within a transaction, locking the table to ensure no other process is writing another record in the meantime

database business-logic-layer

Share

Improve this question

Follow

edited May 23, 2017 at 12:16

Community Bot

1 ● 1

asked Oct 3, 2008 at 14:31

Manrico Corazzi

11.4k ● 10 ● 50 ● 62

7 Answers

Sorted by: Highest score (default)

8

First, you **must** have a primary key or unique constraint on the database to enforce this uniqueness properly - no question.

Given that the constraint exists, which way should you code in the application? My preference would be to try the insert and catch the exceptions. Because presumably most inserts will succeed, only a few will fails as duplicates (that's what "exception" implies!): it is inefficient to perform an exists check before every insert, when the database is going to be performing its own constraint checking anyway.

Also, it is theoretically possible for the exists check to be wrong anyway - if someone else manages to commit a record with the same key value in the small interval between your exists check and your insert. Then, if you don't trap the database exception, you will believe the insert succeeded when in fact it didn't.

Share Improve this answer Follow

edited Oct 3, 2008 at 14:53

answered Oct 3, 2008 at 14:44

Tony Andrews
132k ● 23 ● 227 ● 264

2

You check that the object exists solely in application code, and then once satisfied that it does not, blithely save the object. But another concurrent client might insert their own object in the moment between your two lines of code. So you'd get a Duplicate exception anyway, only this time you don't catch it.

You must do the save() and catch the exception. Otherwise you have a race condition with other concurrent clients working on the same database.

Share Improve this answer Follow

answered Oct 3, 2008 at 16:13

Bill Karwin
561k ● 87 ● 698 ● 854

2

You need to catch the database exception unless you can guarantee that your application is the only one that ever inserts rows (and ever will insert rows) into your database.

EDIT: I may have misunderstand the question, but I would still argue that option B (HibernateSessionFactory throws the ConstraintException from the database) is the better option. There's always a small chance that another application could insert something in the sliver of time between your check and the actual function call. In addition, the only way to check for a dupe is to perform an additional query which is just a needless drain on performance.

My original understanding of the question was that in option A the dupe check would be performed internally (i.e. by using only the data structures that the program had already created, and with no query until the INSERT). My original answer was in response to this method.

Share Improve this answer Follow

edited Oct 3, 2008 at 17:02

answered Oct 3, 2008 at 14:37

JPLemme
4,484 ● 6 ● 32 ● 35

1

In general, I try to avoid coding that relies on errors being thrown because I did something wrong. Sometimes, though, that's all you can do. In your situation, I think you should check first.

Share Improve this answer Follow

answered Oct 3, 2008 at 14:36

wcm
9,241 ● 7 ● 41 ● 65

1

This will break (allowing duplicate entries) if the constraint gets dropped for some reason (typically maintenance work where the DBA neglects to re-enable it). You should check for this situation within the application.

However, it is good database design to have the database enforce the constraint (as you have quite rightly pointed out) as others may also be using the database. As a generalisation it is best to assume that applications and databases live in a M:M relationship - this will be the case almost all of the time.

Share Improve this answer Follow

answered Oct 3, 2008 at 14:56

ConcernedOfTunbridge Wells
66.5k ● 15 ● 148 ● 198

1

The exceptions thrown by Hibernate (or any ORM component) tend to be hard to interpret.

If the exception has enough information that you can produce an error message that actually helps the user, then just catch the exception, analyze it, and move on.

If the exception doesn't have enough information, then you have to check for the error condition, and produce a helpful error message to the user that they're doing something wrong.

The question is one of "how opaque is the exception"? Some are pretty opaque. Others have enough that you can parse the message string and figure out what to say to the user.

Share Improve this answer Follow

answered Oct 3, 2008 at 15:09

S.Lott
391k ● 82 ● 517 ● 788

0

Once hibernate throws an exception from the session you must discard the session (see section 11.2.3). So, if you need to check for dups and continue using the same session then you have no choice but to check first in the application.

Also there is a possibility with the code in the 1st snippet that another process could insert a record that would cause the duplicate exception to be thrown between the time you check for the duplicate record and the time it actually gets inserted.

Share Improve this answer Follow

answered Oct 3, 2008 at 15:11

JMM

556 ● 2 ● 8