Performance considerations for Triggers vs Constraints

Asked 16 years, 2 months ago Modified 10 years, 11 months ago

Viewed 14k times

19

I'm trying to find out whether I should be using business critical logic in a trigger or constraint inside of my database.
So far I've added logic in triggers as it gives me the control over what happens next and means I can provide custom user messages instead of an error that will probably confuse the users.

Is there any noticable performance gain in using constraints over triggers and what are the best practices for determining which to use.

database-design triggers constraints

Share

Improve this question

Follow

edited Dec 6, 2012 at 16:06

Mitch Wheat

300k ● 44 ● 477 ● 550

asked Sep 29, 2008 at 9:47

HAdes

17k ● 23 ● 62 ● 75

1 @hades, consider taking away all sql server references. I think the question and the answers are database-agnostic.
– Sklivvz Sep 29, 2008 at 13:17

12 Answers

Sorted by: Highest score (default)

16

Constraints hands down!

- With constraints you specify relational principles, i.e. facts about your data. You will never need to change your constraints, unless some fact changes (i.e. new requirements).
- With triggers you specify how to handle data (in inserts, updates etc.). This is a "non-relational" way of doing things.

To explain myself better with an analogy: the proper way to write a SQL query is to specify "what you want" instead of "how to get it" – let the RDBMS figure out the best way to do it for you. The same applies here: if you use triggers you have to keep in mind various things like the order of execution, cascading, etc... Let SQL do that for you with constraints if possible.

That's not to say that triggers don't have uses. They do: sometimes you can't use a constraint to specify some fact about your data. It is extremely rare though. If it happens to you *a lot*, then there's probably some issue with the schema.

Share Improve this answer

Follow

answered Sep 29, 2008 at 10:25

Sklivvz

31.1k ● 24 ● 118 ● 174

That's last statement is not entirely true. For example, what if inserting into one table requires a row to be inserted in another table? For that you need a trigger, and it does not imply there is anything wrong with the schema.
– Mitch Wheat Sep 29, 2008 at 10:31

2 Ok, but in that case your database is not normalized correctly!? – Sklivvz Sep 29, 2008 at 10:48

2 @Skliwz, no that is often used to insert to audit tables for instance which is not a schema problem. – HLGEM Sep 27, 2011 at 17:17

14

Best Practice: if you can do it with a constraint, use a constraint.

Triggers are not quite as bad as they get discredit for (if used correctly), although I would always use a constraint where ever possible. **In a modern RDMS, the performance overhead of triggers is comparable to constraints** (of course, that doesn't mean someone can't place horrendous code in a trigger!).

Occasionally it's necessary to use a trigger to enforce a 'complex' constraint such as the situation of wanting to enforce that one and only one of a Table's two Foreign key fields are populated (I've seen this situation in a few domain models).

The debate of whether the business logic should reside in the application rather than the DB, depends to some extent on the environment; if you have many applications accessing the DB, both constraints and triggers can serve as final guard that data is correct.

Share Improve this answer Follow

edited Jul 8, 2013 at 1:45

answered Sep 29, 2008 at 10:10

Mitch Wheat
300k ● 44 ● 477 ● 550

First, I completely agree with your answer; but, I think you messed up in the example you provided for a 'complex' constraint where a trigger is needed. If a table has two foreign key columns and you want to ensure only one is populated, a check constraint is the perfect fit: `CHECK (ColA is null and ColB is not null or ColA is not null and ColB is null)` – tomosius Jan 16, 2017 at 6:04

For that example, maybe you meant a case where two different tables can reference a third table but a record in the third table can only be referenced by either first or second table but not both. In this case, a check constraint will not work but a trigger can. – tomosius Jan 16, 2017 at 6:08

7

Triggers can blossom into a performance problem. About the same time that happens they've also become a maintenance nightmare. You can't figure out what's happening **and** (bonus!) the application behaves erratically with "spurious" data problems. [Really, they're trigger issues.]

No end-user touches SQL directly. They use application programs. Application programs contain business logic in a much smarter and more maintainable way than triggers. Put the application logic in application programs. Put data in the database.

Unless you and your "users" don't share a common language, you can explain the constraint violations to them. The alternative -- not explaining -- turns a simple database into a problem because it conflates the data and the application code into an unmaintainable quagmire.

"How do I get absolute assurance that everyone's using the data model correctly?"

Two (and a half) techniques.

1. Make sure the model is **right**: it matches the real-world problem domain. No hacks or workaround or shortcuts that can only be sorted out through complex hand-waving explanations, stored procedures and triggers.
2. Help define the business model layer of the applications. The layer of application code that everyone shares and reuses.

   a. Also, be sure that the model layer meets people's needs. If the model layer has the right methods and

> collections, there's less incentive to bypass it to get direct access to the underlying data. Generally, if the model is right, this isn't a profound concern.

Triggers are an train-wreck waiting to happen. Constraints aren't.

Share Improve this answer Follow

edited Sep 29, 2008 at 13:43

answered Sep 29, 2008 at 10:02

S.Lott
391k ● 82 ● 517 ● 788

---

6 Surely by relying on applications to enforce the business critical logic is a problem waiting to happen. What about new applications, they will also have to include this business logic, and therefore relying on the app developer to not forget this. Controlling at DB end is surely the best way. – HAdes Sep 29, 2008 at 12:57

I think that's why we have reusable code libraries and a separate model layer -- to assure that all applications are using the common business model. Anyway, that's what I do -- build a business model layer. – S.Lott Sep 29, 2008 at 13:14

---

4 Logic should never be only in the application. It must be at the database level or data integrity is threatened when other applications or direct queries or imported data are put into the database. – HLGEM Sep 29, 2008 at 13:15

---

4 People will still often bypass the business logic layer. Even if the business layer is easy to use, someone will think they've got an easier way. My own philosophy is to include all business flow logic in the business layer, but data constraints

at the DB. A DBMS is more than just a holding place – Tom H Sep 29, 2008 at 14:07

1 Remember, the code that is out of your hands, is in the hands of the enemy. In this case, the enemy is the jerk that thinks he knows better and tinkers with your data directly instead of using the API you gave him. Enforce integrity at the DB and he won't have a chance to hose it. – Coderer Oct 27, 2008 at 20:16

7

In addition to the other reasons to use constraints, the Oracle optimizer can use constraints to it's advantage.

For example, if you have a constraint saying `(Amount >= 0)` and then you query with `WHERE (Amount = -5)` Oracle knows immediately that there are no matching rows.

Share Improve this answer Follow

answered Nov 12, 2008 at 23:01

WW.
24.3k ● 15 ● 97 ● 124

5

Constraints and triggers are for 2 different things. Constraints are used to *constrain* the domain (valid inputs) of your data. For instance, a SSN would be stored as char(9), but with a constraint of [0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9] (all numeric).

Triggers are a way of enforcing business *logic* in your database. Taking SSN again, perhaps an audit trail needs to be maintained whenever an SSN is changed - that would be done with a trigger, In general, data integrity issues in a modern RDBMS can be handled with some variation of a constraint. However, you'll sometimes get into a situation where improper normalization (or changed requirements, resulting in now improper normalization) prevents a constraint. In that case, a trigger may be able to enforce your constraint - but it is opaque to the RDBMS, meaning it can't be used for optimization. It's also "hidden" logic, and can be a maintenance issue. Deciding whether to refactor the schema or use a trigger is a judgment call at that point.

Share Improve this answer Follow

answered Sep 29, 2008 at 11:04

Mark Brackett

85.6k ● 17 ● 111 ● 155

4

Generally speaking I would prefer constraints and my code would catch sql server errors and present something more friendly to the user.

Share Improve this answer Follow

answered Sep 29, 2008 at 9:59

Sam Meldrum

14k ● 6 ● 35 ● 40

3

@onedaywhen

You can have a query as a constraint in SQL Server, you just have to be able to fit it in a scalar function:http://www.eggheadcafe.com/software/aspnet/30056435/check-contraints-and-tsql.aspx

Share Improve this answer Follow

answered Oct 1, 2008 at 21:43

Meff

5,989 ● 29 ● 36

1 There are problems associated with UDFs in `CHECK` constraints (see here). – onedaywhen Feb 22, 2012 at 15:44

2

@Mark Brackett: "Constraints are used to constrain the domain... Triggers are a way of enforcing business logic": It's not that simple in SQL Server because its constraints' functionality is limited e.g. not yet full SQL-92. Take the classic example of a sequenced 'primary key' in a temporal database table: ideally I'd use a CHECK constraint with a subquery to prevent overlapping periods for the same entity but SQL Server can't do that so I have to use a trigger. Also missing from SQL Server is the SQL-92 ability to defer the checking of constraints but instead they are (in effect) checked after every SQL statement, so again a trigger may be necessary to work around SQL Server's limitations.

Share Improve this answer

edited Sep 30, 2008 at 7:40

Follow

answered Sep 29, 2008 at 13:58

onedaywhen
56.9k ● 12 ● 102 ● 142

2

If at all possible use constraints. They tend to be slighlty faster. Triggers should be used for complex logic that a constraint can't handle. Trigger writing is tricky as well and if you find you must write a trigger, make sure to use set-based statements becasue triigers operate against the whole insert, update or delete (Yes there will be times when more than one record is affected, plan on that!), not just one record at a time. Do not use a cursor in a trigger if it can be avoided.

As far whether to put the logic in the application instead of a trigger or constraint. DO NOT DO THAT!!! Yes, the applications should have checks before they send the data, but data integrity and business logic must be at the database level or your data will get messed up when multiple applications hook into it, when global inserts are done outsiide the application etc. Data integrity is key to databases and must be enforced at the database level.

Share Improve this answer Follow

edited Sep 27, 2011 at 17:14

answered Sep 29, 2008 at 13:21

HLGEM
96.4k ● 15 ● 119 ● 189

1

@Meff: there are potential problems with the approach of using a function because, simply put, SQL Server CHECK constraints were designed with a single row as the unit of work, and has flaws when working on a resultset. For some more details on this, see: [http://blogs.conchango.com/davidportas/archive/2007/02/19/Trouble-with-CHECK-Constraints.aspx][1].

[1]: David Portas' Blog: Trouble with CHECK constraints.

Share Improve this answer Follow

answered Oct 2, 2008 at 10:11

onedaywhen
56.9k ● 12 ● 102 ● 142

Very useful, thank you. I've only used a scalar UDF as a constraint once in a production db, and from that link it'll be OK, as the table only had 6 rows! – Meff Oct 9, 2008 at 10:58

1

Same as Skliwz. Just to let you know a **canonical use of trigger** is audit table. If many procedures update/insert/delete a table you want to audit (who modified what and when), trigger is the simplest way to do it. one way is to simply add a flag in your table (active/inactive with some unicity constraint) and insert something in the audit table.

Another way if you want the table not to hold the historical data is to copy the former row in your audit table...

Many people have many ways of doing it. But one thing is for sure, you'll have to perform an insert for each update/insert/delete in this table

To avoid writing the insert in dozen of different places, you can here use a trigger.

Share Improve this answer Follow

answered Jan 9, 2014 at 8:22

user2346536

1,474 ● 2 ● 21 ● 43

0

I agree with everyone here about constraints. Use them as much as possible.

There is a tendency to overuse triggers, especially with new developers. I have seen situations where a trigger fires another trigger which fires another trigger that repeats the first trigger, creating a cascading trigger that ties up your server. This is a non-optimal user of triggers ;o)

That being said, triggers have their place and should be used when appropriate. They are especially good for tracking changes in data (as Mark Brackett mentioned).

You need to answer the question "Where does it make the most sense to put my business logic"? Most of the time I think it belongs in the code but you have to keep an open mind.

Share Improve this answer Follow

answered Sep 29, 2008 at 13:00

wcm
9,241 ● 7 ● 41 ● 65