SQL Server: how to constrain a table to contain a single row?

Ask Question

66

I want to store a single row in a configuration table for my application. I would like to enforce that this table can contain only one row.

What is the simplest way to enforce the single row constraint ?

sql-server database-design singleton

19

edited Oct 19 '10 at 14:49

A-K

14k 5 46 66

asked Oct 19 '10 at 10:25

Martin

6,754 5 31 30

Why not use a table with columns `(Name, Value)` with a primary key on Name. Then you can `select Value from Table where Name = ?` with certainty that either no rows or one row will be returned. – a'r Oct 19 '10 at 10:28

2 I'm not sure sql is the best solution here. Maybe a simple xml file is more appropriate for configuration. I use to think that configuration != data and

sql was made for data. – remi bourgarel Oct 19 '10 at 10:29

2 @ar - I've seen that go badly wrong when you're expecting to read, say, an integer, and you get some badly formatted value in the value column. – Damien_The_Unbeliever Oct 19 '10 at 10:40

@Damien_The_Unbeliever Why would that happen? Because you specified a nonexistent value for `Name` ? – Noumenon May 9 '18 at 14:38

1 @Noumenon - note that my comment was a response to `ar` s comment. The issue is, if you're just storing name/value pairs, the value pretty well has to be string, and you've got no means of *enforcing* validation in the database. When you use a single-row table with separate *columns* for each setting (as the OP wanted) then you can *easily* enforce validation for each configuration setting via check constraints. – Damien_The_Unbeliever May 9 '18 at 14:50

10 Answers

81

You make sure one of the columns can only contain one value, and then make that the primary key (or apply a uniqueness constraint).

```
CREATE TABLE T1(
    Lock char(1) not null,
    /* Other columns */,
    constraint PK_T1 PRIMARY KEY (Lock),
    constraint CK_T1_Locked CHECK (Lock='X')
)
```

I have a number of these tables in various databases, mostly for storing config. It's a lot nicer knowing that, if the config item should be an int, you'll only ever read an int from the DB.

edited Apr 1 '16 at 8:55

Luke Girvin
10.9k 8 52 74

answered Oct 19 '10 at 10:36

Damien_The_Unbeliever
197k 17 255 344

1 +1. This is the approach Celko uses for an auxiliary constants table in "SQL for Smarties" – Martin Smith Oct 19 '10 at 10:42

This is about as simple as it gets. Thanks. – Martin Oct 19 '10 at 11:06

+1: Interesting solution. I had not seen this before so thanks for sharing. Always the way, easy when you know how.... – John Sansom Oct 19 '10 at 11:50

Here's a follow up question to think over. What is the primary key of this table? :) – nvogel Oct 20 '10 at 15:30

1 @BZ - c.d.t is shorthand for `comp.databases.theory` , a usenet group (visible through Google groups) that I admit I haven't read much of recently. It was more oriented around relational theory than SQL - but I happened to know that dportas/sqlvogel also frequented the same group. TTM was a reference to The Third Manifesto, which is a good book talking (again) about relational theory rather than SQL. – Damien_The_Unbeliever Mar 28 '13 at 7:03

45

I usually use Damien's approach, which has always worked great for me, but I also add one thing:

```
CREATE TABLE T1(
    Lock char(1) not null DEFAULT 'X',
    /* Other columns */,
    constraint PK_T1 PRIMARY KEY (Lock),
    constraint CK_T1_Locked CHECK (Lock='X')
)
```

Adding the "DEFAULT 'X'", you will never have to deal with the Lock column, and won't have to remember which was the lock value when loading the table for the first time.

answered Oct 19 '10 at 18:54

ACB
705 5 8

3 +1 - nice tip, thanks. – Martin Oct 20 '10 at 6:00

1 The default constraint should also be named or else it will get a autogenerated confusing name. `Lock char(1) not null CONSTRAINT DF_T1_Lock DEFAULT 'X'` – David S. May 18 '17 at 12:29

14

You may want to rethink this strategy. In similar situations, I've often found it invaluable to leave the old configuration rows lying around for historical information.

To do that, you actually have an extra column `creation_date_time` (date/time of insertion or update) and an insert or insert/update trigger which will populate it correctly with the current date/time.

Then, in order to get your current configuration, you use something like:

```
select * from config_table order by creation_date_time desc fetch fi
```

(depending on your DBMS flavour).

That way, you still get to maintain the history for recovery purposes (you can institute cleanup procedures if the table gets too big but this is unlikely) and you still get to work with the latest configuration.

answered Oct 19 '10 at 10:35

paxdiablo
641k 174 1262 1681

Home

PUBLIC

Stack Overflow

Tags

Users

Jobs

Teams
Q&A for work

Learn More

+1: I hear what you are saying, but I prefer to record the history of changes in separate audit tables. – Martin Oct 19 '10 at 11:03

+1: For sharing the interesting idea of implementing an audit trail. – John Sansom Oct 19 '10 at 11:49

Nice. Just `SELECT TOP 1 ... ORDER BY creation_date_time DESC` – Baodad Feb 8 '16 at 19:54

4

You can implement an INSTEAD OF Trigger to enforce this type of business logic within the database.

The trigger can contain logic to check if a record already exists in the table and if so, ROLLBACK the Insert.

Now, taking a step back to look at the bigger picture, I wonder if perhaps there is an alternative and more suitable way for you to store this information, perhaps in a configuration file or environment variable for example?

edited Oct 19 '10 at 11:47

answered Oct 19 '10 at 10:27

John Sansom
35.7k 8 62 78

+1 - I can see how the trigger would work, and maybe I will fall back on that approach, but I like Damien's simple constraint. There is an interesting discussion to be had on what type of configuration data belongs in the config file and what belongs in the DB. In this case I think the DB is the correct place. No doubt I will live to regret this... :-) – Martin Oct 19 '10 at 11:13

I use a bit field for primary key with name IsActive. So there can be 2 rows at most and and the sql to get the valid row is: select * from Settings where IsActive = 1 if the table is named Settings.

2

answered Mar 5 '14 at 9:52

user3382925

21 1

Here's a solution I came up with for a lock-type table which can contain only one row, holding a Y or N (an application lock state, for example).

2

Create the table with one column. I put a check constraint on the one column so that only a Y or N can be put in it. (Or 1 or 0, or whatever)

Insert one row in the table, with the "normal" state (e.g. N means not locked)

Then create an INSERT trigger on the table that only has a SIGNAL (DB2) or RAISERROR (SQL Server) or RAISE_APPLICATION_ERROR (Oracle). This makes it so application code can update the table, but any INSERT fails.

DB2 example:

```
create table PRICE_LIST_LOCK
(
    LOCKED_YN       char(1)   not null
        constraint PRICE_LIST_LOCK_YN_CK  check (LOCKED_YN in ('Y',
);
--- do this insert when creating the table
insert into PRICE_LIST_LOCK
values ('N');

--- once there is one row in the table, create this trigger
CREATE TRIGGER ONLY_ONE_ROW_IN_PRICE_LIST_LOCK
   NO CASCADE
   BEFORE INSERT ON PRICE_LIST_LOCK
   FOR EACH ROW
```

```
SIGNAL SQLSTATE '81000'  -- arbitrary user-defined value
  SET MESSAGE_TEXT='Only one row is allowed in this table';
```

Works for me.

answered Mar 11 '15 at 14:35

David A Beamer
45 1 8

1

You can write a trigger on the insert action on the table. Whenever someone tries to insert a new row in the table, fire away the logic of removing the latest row in the insert trigger code.

answered Oct 19 '10 at 10:28

Karan
1,249 4 31 58

1

Old question but how about using IDENTITY(MAX,1) of a small column type?

```
CREATE TABLE [dbo].[Config](
[ID] [tinyint] IDENTITY(255,1) NOT NULL,
[Config1] [nvarchar](max) NOT NULL,
[Config2] [nvarchar](max) NOT NULL
```

answered Apr 13 '15 at 13:55

NeutronCode
199 2 13

You could add another row by using SET IDENTITY_INSERT. – Razvan Socol Oct 5 '18 at 4:44

0

```
IF NOT EXISTS ( select * from table )
BEGIN
    ///Your insert statement
END
```

answered Oct 19 '10 at 10:27

Sachin Shanbhag
44.1k 9 74 96

0

Here we can also make an invisible value which will be the same after first entry in the database.Example: Student Table: Id:int firstname:char Here in the entry box,we have to specify the same value for id column which will restrict as after first entry other than writing lock bla bla due to primary key constraint thus having only one row forever. Hope this helps!

answered Feb 26 '13 at 16:15

Yishagerew
1