Check if record exists with Dapper ORM

Asked 3 years ago Active 2 months ago Viewed 9k times

15

What is the simplest way to check if record exists using the Dapper ORM?

Do I really need to define POCO objects for a query where I only want to check if a record exists?

dapper

4

asked Aug 17 '16 at 21:23

webworm

4,605 21 85 157

7 Answers

38

✓

```
int id = ...
var exists = conn.ExecuteScalar<bool>("select count(1) from Table where Id=@id", new
{id});
```

should work...

answered Aug 18 '16 at 16:39

Marc Gravell ♦

821k 212 2223 2621

2 @webworm note it gets tricky if you have more than 1 record with the key, but ... you shouldn't have that :) – Marc Gravell ♦ Aug 18 '16 at 16:41

1 Just use `COUNT(DISTINCT 1)` to ensure you get an answer of 1 (true) even if you have multiple records with the same id/key. Admittedly that should be an edge case. – Caltor Apr 19 '17 at 12:21

problem. I'd prefer `<int>` personally – Marc Gravell ♦ Sep 5 '17 at 14:41

@MarcGravell what if I needed to do a check on two fields to ensure the record is not a duplicate (e.g. where I have two primary keys on a table)? How would I do that? – Euridice01 Oct 11 '17 at 16:51

4

I think this may have a tad less overhead as there's no function call or data type conversions:

```
int id = ...
var exists = connection.Query<object>(
    "SELECT 1 WHERE EXISTS (SELECT 1 FROM MyTable WHERE ID = @id)", new { id })
    .Any();
```

edited Apr 13 '17 at 23:15

answered Apr 12 '17 at 22:32

Kevin Finck
124 1 5

2

You can have your query to return a bool:

```
    [Test]
    public void TestExists()
    {
        var sql = @"with data as
                    (
                        select 1 as 'Id'
                    )
                    select CASE WHEN EXISTS (SELECT Id FROM data WHERE Id = 1)
                           THEN 1
                           ELSE 0
                      END AS result
                    from data ";

        var result = _connection.Query<bool>(sql).FirstOrDefault();

        Assert.That(result, Is.True);
    }
```

7,106 4 23 49

1 that should work, but I think you're over-complicating it a bit... – Marc Gravell ♦ Aug 18 '16 at 16:40

Agreed. "Count" is a neat trick! – Void Ray Aug 18 '16 at 17:53

2 Nope, disagree. 'Count' processes the entire table since the query optimizer can't rule out multiple matches for your where clause.. 'Exists' returns as soon as the first one is found. The difference can be significant when you have a lot of rows in your table, especially if this query is repeated often. For an example see this blogpost I found - sqlblog.com/blogs/andrew_kelly/archive/2007/12/15/… – Volkirith Aug 24 '17 at 15:15

2 @Volkirith In general, that's true, but there is an exception. If the `WHERE` clause filters by a column with a unique index, the optimizer **can** conclude that there will be 0 or 1 rows and perform a seek instead of a table scan. I guess this is the case here as the column is named "id". – Alejandro Sep 5 '17 at 13:25

2

```
const string sql = "SELECT CAST(CASE WHEN EXISTS (SELECT 1 FROM MyTable WHERE Id = @Id)
THEN 1 ELSE 0 END as BIT)";
bool exists = db.ExecuteScalar<bool>(sql, new { Id = 123 });
```

edited Jan 19 '18 at 13:59

answered Jan 19 '18 at 12:17

janeks malinovskis

263 2 8

I like this answer as the SQL expresses intent and you don't rely on type conversion. – Justin J Stark Jun 8 '18 at 18:59

0

Another option that will run with duplicate records, i.e. not querying the id of the table

```
bool exists = connection.ExecuteScalar<int>(
    "select count(1) from Table where notanId=@value", new { value = val})
     > 0;
```

answered Sep 5 '17 at 13:10

Liam

17.3k 16 80 134

0

If you need to do this sort of query against a non-unique field you can use `HAVING` to handle counts greater than 1.

```
SELECT 1
FROM Table
WHERE Col=@val
HAVING COUNT(1) > 0
```

answered May 3 at 23:53

mikesigs
4,780 3 25 36

0

imho `SELECT TOP(1)` is better than `SELECT COUNT(1)`

```
    bool exists = connection.Query<ValueTuple<long>>(
        "SELECT top(1) Id FROM MYTABLE WHERE MYTABLE.Id=@Id",
        new {Id}).Any());
```

The `ValueTuple<long>` is value type . `Query<object>` map to reference type and causes boxing .

edited Jul 2 at 7:39

marc_s
604k 137 1156
1290

answered Jul 1 at 10:36

Александр Шмыков
6 4
