#

**Article:** Avoid using SELECT \* in a View

**Date**: 14/12/2011

**Posted by**: HeelpBook Staff

**Source**: [Link](http://blog.devstone.com/aaron/CommentView%2Cguid%2C49757941-ed25-4831-8d2f-84f13abf6822.aspx)

**Permalink**: [Link](http://heelpbook.altervista.org/2011/sql-server-caution-avoid-using-select-in-a-sql-view/)

Suppose you have a *table* (**tblDemo**) and a view (**vwDemo**) over that table that joins **tblDemo** to another table to retrieve their results.

The view might resemble the following:

CREATE VIEW vwDemo

AS

SELECT d.\*, o.OtherField01, o.OtherField02

FROM tblDemo d

INNER JOIN tblOther o ON d.ID = o.ID

While I’ve never advocated selecting \* from anything, it does have its uses; but it turns out there’s potentially a nasty gotcha.  Personally, I like to err on the side of higher specificity, though that usually requires a bit more manual tweaking as time goes on.

What happens if the schema of **tblDemo** changes?  Well, if you don’t update or recreate your view, you may very well get some unexpected side effects.

For instance, if a new field were added to **tblDemo** (e.g.,*ALTER TABLE tblDemo ADD NewField int*), the view, when inspected in **SQL** shows the following:

SELECT d.ID, d.Field01, d.Field02, d.Field03, d.NewField AS OtherField01, o.OtherField01 AS OtherField02, o.OtherField02

 FROM tblDemo d

INNER JOIN tblOther o ON d.ID = o.ID

Notice that the *‘NewField’* got aliased with the field name at the new fields position! That field, in turn, received the next field’s alias, and so on. Understandably, this can have some very negative side effects in your applications.

For this reason I would strongly recommend AGAINST using **SELECT \*** in a view in **SQL Server**.  Of course, if your view does nothing but **SELECT \*** on a single table or the ‘\*’ is the last part of the **SELECT** clause (*such as SELECT o.OtherField01, o.OtherField02, d.\* FROM …*) then you circumvent the issue.

When used in conjunction with other columns, however, there are indeed repercussions.

**Use sp\_refreshview to fix this “issue”**

To refresh the view’s metadata information, run the **sp\_refreshview** stored procedure against a typical view called, for example **V1:**

EXEC sp\_refreshview ‘dbo.V1’;

A typical example is where a schema change in the underlying objects is not reflected in the views’ metadata information. You might find it to be a good practice to refresh all views’ metadata information after applying schema changes to objects in the database.

To avoid the tedious process of writing the **sp\_refreshview**statements you can use the following query:

SELECT N’EXEC sp\_refreshview ‘

+ QUOTENAME(VIEW\_NAME, ‘’’’) + ‘;’ AS cmd

FROM (SELECT QUOTENAME(TABLE\_SCHEMA)

+ N’.’ + QUOTENAME(TABLE\_NAME) AS VIEW\_NAME

FROM INFORMATION\_SCHEMA.VIEWS) AS V

WHERE OBJECTPROPERTY(OBJECT\_ID(VIEW\_NAME) , ‘IsSchemaBound’) = 0;

The query will generate as its output the lines of code with the **sp\_refreshview**statements against all views in the database that are not schema-bound.