

Re: TSQL INNER JOINS and null fields

Source:

<http://www.tech-archive.net/Archive/SQL-Server/microsoft.public.sqlserver.programming/2004-05/2321.html>

From: Hugo Kornelis (*hugo_at_pe_NO_rFact.in_SPAM_fo*)

Date: 05/11/04

Date: Tue, 11 May 2004 11:07:41 +0200

On Tue, 11 May 2004 01:36:02 -0700, PMcG wrote:

> i'm not intrested in the statements thenseleves,i'm just wondering why the first statement will not return both records as the seondry identifier field contains null and and i would expect that to match.

Hi PMcG,

It has to do with the meaning of NULL and with something called "three-valued logic".

In the relational model, NULL can be best thought of as "Unknown". In the following table, the age of both Paul and Rick is unknown. Clara's job is unknown as well (we know that a NULL for a minor's job means "n/a", but the database doesn't – one of the many known probles with NULLS)

Name Age Job

Sandra 37 Dentist
Paul (null) Programmer
Rick (null) Barkeeper
Clara 5 (null)
Pete 37 Programmer

If we are looking for people with the same age, Sandra and Pete qualify. But do Sandra and Paul have the same age? Or Paul and Rick? Since you don't know Paul's and Rick's age, you can't tell, so the only verifiable correct answer is "Maybe". That's the third value in 3-valued logic (the other two being true anf false).

In databases, a comparison of anything to NULL is always unknown. Is Pete's age equal to Rick's age? maybe. But the answer to "Is Pete's age UNEqual to Rick's age?" is "maybe" as well! The same goes for a comparison between Paul's and Rick's age, therefore any comparison of NULL to NULL will evaluate to "maybe".

The select statement will only return rows where the WHERE condition evaluates to "True". No benefit of the doubt – if the WHERE condition evaluates to "Maybe", the row won't be returned.

Q: But how do I find rows with a NULL value in a column?

A: Use IS NULL. To find all persons without job, you use
SELECT Name
FROM People
WHERE Job IS NULL

Q: How can I make SQL Server treat two NULLs as being equal?

A: There is a SET option, provided for backward compatibility. I don't know if this option will still exist in future releases. I recommend you don't use it, as you'll get in the habit of writing non-standard, non-portable SQL. If your next job involves writing SQL for another DBMS, you'll be very sorry.

A better solution is to include the possibility of NULLs in the query. To compare ages as if all NULLs were the same, use either:

```
WHERE ( a.Age = b.Age  
       OR (a.Age IS NULL and b.Age IS NULL))
```

or:

```
WHERE COALESCE(a.Age, -1) = COALESCE(b.Age, -1)
```

(This works only if you have a value that fits the columns' domain but will never be actually stored in the column)

Best, Hugo

--

(Remove _NO_ and _SPAM_ to get my e-mail address)