

Re: Printing the KEYS of all tables

Source:

<http://www.tech-archive.net/Archive/SQL-Server/microsoft.public.sqlserver.programming/2004-03/3473.html>

From: Grant Case (*hangtime79_at_DONThSoPtAmMail.com*)

Date: 03/14/04

Date: Sun, 14 Mar 2004 12:03:52 -0600

Eh? The system tables are documented and supported. I . capabilities that the old features have.

I understood the system tables as being unsupported totally by Microsoft not just a few columns from a former mentor of mine and never looked any further than that. I am mistaken as put in the Books Online:

Reference of documented columns in system tables is permissible. However, many of the columns in system tables are not documented. Applications should not be written to query undocumented columns directly. Applications should instead use any of these components to retrieve information stored in the system tables:

- Information schema views
- System stored procedures
- Transact-SQL statements and functions
- SQL-DMO
- Database application programming interfaces (API) catalog functions

These components constitute a published API for obtaining system information from SQL Server. Microsoft maintains the compatibility of these components from release to release. The format of the system tables is dependent upon the internal architecture of SQL Server and may change from release to release. Therefore, applications that directly access the undocumented columns of system tables may have to be changed before they can access a later version of SQL Server.

If you feel more comfortable with INFORMATION_SCHEMA, fine. But I don't think you should not advice other people to use them in favour of the system tables, and certainly not give incorrect arguments like use of the tables is unsupported, or that code that access system tables will not work in Yukon.

First, never in my posts did I say that code to access system tables in Yukon will not work; I said in my original post that the system tables will be seriously altered. I should have prefaced that by saying that this altering will not break pre-existing code but may not return the expected results as returned today by SQL Server 2000 either (see bottom of the message). I was mistaken in my understanding in terms of Microsoft's support for the system tables. However, I will still encourage use of the Information_Schema views over the system tables for individuals who are trying to find the basic metadata information (column names, datatypes, etc.). It's easier to point a web developer who's concept of SQL is "SELECT * FROM Customer" to the Information_Schema views then try to guide them through the system tables.

I for my part prefer the system tables for a couple of reasons:

1) I have worked with them for many years, so I don't see the point of learning a new schema.

We all will/are going to learn a new schema in Yukon in the form of the catalog views regardless. Never stop learning J.

2) I prefer one-stop shopping. The INFORMATION_SCHEMA are hopelessly incomplete.

I never referred to the Information_Schema as being the be all, end all in fact I referred to the system procedures Microsoft provided as helping in the understanding of metadata. I used the arguments of ease of use, portability, and upgradability over completeness.

3) All that uppercase is ugly.

Yea, I'm not a real big fan of uppercase in my object and column naming convention either. Different strokes, gotta throw a bone to all those DB2 guys. However, this is not much of a reason to avoid an excellent tool altogether unless you think its screaming at you all the time.

4) SQL Server MVP Tony Rogerson is quick to point out that they don't scale well.

I do not know how scale came into the question, but sure why not. However, if you're building an application that needs performance out of querying metadata, you're probably building a metadata application and already looking at the system tables and are making plans to deal with catalog views in Yukon.

5) In Yukon, the catalog views will be the right place to query metadata, so it's not much of a point to learn a concept that you will only use for a year anyway.

Agreed, catalog views will be the most complete place to get metadata information in Yukon, but as stated on Microsoft's site at

<http://www.microsoft.com/technet/prodtechnol/sql/yukon/deploy/sqlsysec.msp>

"Introduction to SQL Server 'Yukon' Relational Engine Security Features"

Visibility of Metadata

System objects, including the new catalog views, expose metadata to the user. One of the goals of SQL Server "Yukon" is to control metadata visibility so that users only see the metadata of objects on which they have permissions. The following actions will selectively disclose metadata:

- .
SELECTing from a catalog view in the sys schema

- .
Accessing an object's extended properties

- .
SELECTing from an INFORMATION_SCHEMA view

- .
(These views are provided to meet ANSI compliance, and in SQL Server Yukon they are defined by referencing the catalog views.)

- .
Backward-compatible views

SQL Server Yukon maintains a set of views, corresponding to the SQL Server 2000 "sys" tables. Some of these tables were found in the master database only, and some were found in all databases. These are the so-called 'Backwards-compatible Views' that correspond to each of the SQL Server 2000 system tables like sysobjects, syscomments, and sysindexes. These backwards-compatible views can be accessed just as they were in SQL Server 2000, so that when you execute this code in SQL Server Yukon:

```
SELECT * FROM sysobjects
```

SQL Server will actually select from the sys.objects catalog view:

```
SELECT * FROM sys.objects
```

Note that the backward-compatible views such as sys.objects are subject to metadata disclosure restrictions because they are actually referencing the new SQL Server Yukon catalog. Also note that although the backward-compatible views are based on the new Yukon views, the two statements above will not give identical results, or even contain the same number of rows in their result set. There are new SQL Server Yukon objects like Service Broker Queues, XML indexes, partitioned indexes, etc., that cannot be mapped into the SQL Server 2000 sysobjects format. In addition, you will gain a performance advantage by using the new SQL Server Yukon catalog views instead of the backward-compatible views. It is recommended

that you use the new catalog views, such as sys.objects whenever possible.

In essence to get at the most complete information, you will need to hit the catalog views and anything written off the system tables will not necessarily return all the information you were expecting thus being incomplete. The Information_Schema views will still perform as they do today. In fact, I expect a great deal more of the columns to be used in the Information_Schema views once Yukon rolls out. Hopefully, as you pointed out in your post the Information_Schema.Routines.Last_Altered field will actually have the correct altered date. However, someone had to make the call back in development that the creation date was also the last altered date. If it had been me, I would have left it as NULL.

Back to your original statement: so it's not much of a point to learn a concept that you will only use for a year anyway. When information is equal between the Information_Schema views and the system tables I submit that it is better to select off the Information_Schema views to get your information and therefore teach others to do the same. In two years, if you are using Yukon and the information you get today comes from the system tables, you will not be using the backward-compatible views, you will be using the system catalogs because the information contained within the backward-compatibility views as stated previously by the Microsoft article is an incomplete view of what's happening in the system. However, if the information you use today comes from the Information_Schema views, you will still have that same information returned to you in Yukon and therefore have no need to modify your queries to arrive at the same result. Thus while selecting from the system tables may be supported by Microsoft today and into Yukon, it is still better to query the Information_Schema views when the same information exists in both places because the fashion upon which system table queries are made will change in Yukon while those from the Information_Schema views will not unless as you stated previously there exists a need to have high performance queries running off metadata tables and even then this will not help you once you get to Yukon because you will need to rewrite your queries anyway. In conclusion, if we were to accept your argument that we not learn a concept that will not be used in a year, no one should learn the system tables today because as stated they will be incomplete and superseded by the system catalog in Yukon whereas the Information_Schema views will not, however, we know we should learn the system tables because they guide how SQL Server 2000 works.

HTH,

Grant