

# SQLNumResultCols returns 0 when MSSQL procedure has IF clause

---

*Source:*

<http://www.tech-archive.net/Archive/SQL-Server/microsoft.public.sqlserver.odbc/2006-12/msg00021.html>

---

- *From:* KM <[KM@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:KM@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Sun, 17 Dec 2006 23:09:00 -0800
- 

SQLNumResultCols returns 0 for following stored procedure using both SQL Native client and SQL Server client. It seems to be related to STATIC cursor because if I do not set cursor to STATIC, then SQLNumResultCols returns "1", which is expected.

Datasource  
Microsoft SQL Server 2000 – 8.00.2039 (Intel X86)

ODBC Driver  
SQL Native Client 2005.90.1399.00  
SQL Server 2000.85.1117.00

If we do not have IF clause, SQLNumResultCols returns column number as expected.

Here is the SQL Server procedure DDL:

```
CREATE PROCEDURE J15USER1.SPGVTFED_RS3 @V1 INTEGER AS
IF (@V1 = 0)
SELECT 1 AS "1"
FROM "j15user1".SPGVTFED_TAB
WHERE COLSML=1
ELSE IF (@V1 = 1)
SELECT COLSML,COLDEC,COLVCH,COLTSP
FROM SPGVTFED_TAB
WHERE COLSML <=10 ORDER BY COLSML
```

Table used in the procedure is as follows:

```
CREATE TABLE [j15user1].[SPGVTFED_TAB](
[PARAM] [char](1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
[COLSML] [smallint] NULL,
[COLINT] [int] NULL,
[COLBIT] [bigint] NULL,
[COLFLT] [float] NULL,
[COLREL] [real] NULL,
[COLDBL] [float] NULL,
```

## SQLNumResultCols returns 0 when MSSQL procedure has IF clause

```
[COLDEC] [decimal](10, 5) NULL,  
[COLNUM] [decimal](8, 2) NULL,  
[COLCHR] [char](8) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,  
[COLCHB] [binary](8) NULL,  
[COLVCH] [varchar](12) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,  
[COLVCB] [varbinary](12) NULL,  
[COLDAT] [datetime] NULL,  
[COLTIM] [datetime] NULL,  
[COLTSP] [datetime] NULL  
) ON [PRIMARY]
```

Does anyone know if this is a bug or not?

Thank you,  
Kaoru

Here is the sample code which I have tested:

```
#include <stdio.h>  
#include <string.h>  
#include <windows.h>  
#include <sql.h>  
#include <sqlext.h>  
#include <odbcss.h>  
  
#define MAXBUFLLEN 255  
#define MAXNAME 255  
  
SQLHENV henv = SQL_NULL_HENV;  
SQLHDBC hdbc1 = SQL_NULL_HDBC;  
SQLHSTMT hstmt1 = SQL_NULL_HSTMT;  
SQLHSTMT hstmt2 = SQL_NULL_HSTMT;  
  
SQLCHAR server[] = "DSN";  
SQLCHAR user[] = "user";  
SQLCHAR auth[] = "password";  
  
SQLSMALLINT i;  
SQLLEN n;  
SQLCHAR name;  
SQLSMALLINT namelen;  
SQLSMALLINT datatype;  
unsigned long colsize;  
SQLSMALLINT dec;  
SQLSMALLINT null;  
struct timeval start;  
struct timeval end;  
char buf[32];  
int data=1;  
unsigned char stmt_text1[160000] = "";
```

## SQLNumResultCols returns 0 when MSSQL procedure has IF clause

```
void ProcessLogMessages(SQLSMALLINT plm_handle_type,
SQLHANDLE plm_handle, char *logstring,
int ConnInd);
void e(SQLRETURN, char*);

void Cleanup();

int main() {
RETCODE retcode;
/* SQLBindParameter variables. */
SWORD sParm1=0, sParm2=1;
long cbParm1=SQL_NTS, cbParm2=SQL_NTS;

/* Allocate the ODBC environment and save handle. */
retcode = SQLAllocHandle (SQL_HANDLE_ENV, 0, &henv);
if( (retcode != SQL_SUCCESS_WITH_INFO) &&
(retcode != SQL_SUCCESS)) {
printf("SQLAllocHandle(Env) Failed\n\n");
Cleanup();
return(9);
}

/* Notify ODBC that this is an ODBC 3.0 app. */
retcode = SQLSetEnvAttr(henv, SQL_ATTR_ODBC_VERSION,
(SQLPOINTER) SQL_OV_ODBC3, SQL_IS_INTEGER);
if( (retcode != SQL_SUCCESS_WITH_INFO) &&
(retcode != SQL_SUCCESS)) {
printf("SQLSetEnvAttr(ODBC version) Failed\n\n");
Cleanup();
return(9);
}

/* Allocate ODBC connection handle and connect. */
retcode = SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc1);
if( (retcode != SQL_SUCCESS_WITH_INFO) &&
(retcode != SQL_SUCCESS)) {
printf("SQLAllocHandle(hdbc1) Failed\n\n");
Cleanup();
return(9);
}

retcode = SQLConnect(hdbc1,
server, SQL_NTS,
user, SQL_NTS,
auth, SQL_NTS);
if( (retcode != SQL_SUCCESS) &&
(retcode != SQL_SUCCESS_WITH_INFO) ) {
printf("SQLConnect() Failed\n\n");
Cleanup();
return(9);
}
```

## SQLNumResultCols returns 0 when MSSQL procedure has IF clause

```
}

/* Allocate statement handle. */
retcode = SQLAllocHandle(SQL_HANDLE_STMT, hdbc1, &hstmt1);
if( (retcode != SQL_SUCCESS) &&
    (retcode != SQL_SUCCESS_WITH_INFO) ) {
    printf("SQLAllocHandle(hstmt1) Failed\n\n");
    Cleanup();
    return(9);
}

/* Allocate statement handle. */
retcode = SQLAllocHandle(SQL_HANDLE_STMT, hdbc1, &hstmt2);
if( (retcode != SQL_SUCCESS) &&
    (retcode != SQL_SUCCESS_WITH_INFO) ) {
    printf("SQLAllocHandle(hstmt2) Failed\n\n");
    Cleanup();
    return(9);
}

/* Set cursor type static */

retcode = SQLSetStmtAttr( hstmt1, SQL_ATTR_CURSOR_TYPE,
    (SQLPOINTER)SQL_CURSOR_STATIC, 0);
e(retcode, "hstmt1: cursor static");
if( (retcode != SQL_SUCCESS_WITH_INFO) &&
    (retcode != SQL_SUCCESS) ) {
    printf("SQLSetStmtAttr(hstmt1) Failed\n\n");
    Cleanup();
    return(9);
}

/* Bind the input parameter to variable sParm1. */

retcode = SQLBindParameter(hstmt1,1,SQL_PARAM_INPUT,SQL_C_SSHORT,
    SQL_INTEGER,0,0,&sParm1,0,&cbParm1);
e(retcode, "hstmt1: bind parameter");
if( (retcode != SQL_SUCCESS) &&
    (retcode != SQL_SUCCESS_WITH_INFO) ) {
    printf("SQLBindParameter(sParm1) Failed\n\n");
    Cleanup();
    return(9);
}

/* Execute the command. */

retcode = SQLExecDirect(hstmt1, (UCHAR*)" {call J15USER1.SPGVTFED_RS3(?)}",
    SQL_NTS);
e(retcode, "hstmt1 : execute");

if( (retcode != SQL_SUCCESS) &&
```

## SQLNumResultCols returns 0 when MSSQL procedure has IF clause

```
(retcode != SQL_SUCCESS_WITH_INFO) ) {
printf("SQLExecDirect Failed\n\n");
ProcessLogMessages(SQL_HANDLE_STMT, hstmt1,
"SQLExecute() Failed\n\n", 1);
Cleanup();
return(9);
}

retcode = SQLNumResultCols (hstmt1, &i);
e(retcode, "hstmt1 : num res cols");
if( (retcode != SQL_SUCCESS) &&
(retcode != SQL_SUCCESS_WITH_INFO) ) {
printf("SQLExecDirect Failed\n\n");
ProcessLogMessages(SQL_HANDLE_STMT, hstmt1,
"SQLNumResultCols() Failed\n\n", 1);
Cleanup();
return(9);
}

printf("SQLNumResultCols: Output = %d\n",i);

/* Clear any result sets generated. */
while ( ( retcode = SQLMoreResults(hstmt1) ) != SQL_NO_DATA );
e(retcode, "hstmt1: more results");

/* Clean up. */
SQLFreeHandle(SQL_HANDLE_STMT, hstmt1);
SQLDisconnect(hdbc1);
SQLFreeHandle(SQL_HANDLE_DBC, hdbc1);
SQLFreeHandle(SQL_HANDLE_ENV, henv);
return(0);
}

void Cleanup()
{
if (hstmt1 != SQL_NULL_HSTMT)
SQLFreeHandle(SQL_HANDLE_STMT, hstmt1);
if (hdbc1 != SQL_NULL_HDBC)
{
SQLDisconnect(hdbc1);
SQLFreeHandle(SQL_HANDLE_DBC, hdbc1);
}
if (henv != SQL_NULL_HENV)
SQLFreeHandle(SQL_HANDLE_ENV, henv);
}

void ProcessLogMessages(SQLSMALLINT plm_handle_type,
SQLHANDLE plm_handle,
char *logstring, int ConnInd)
{
RETCODE plm_retcode = SQL_SUCCESS;
```

## SQLNumResultCols returns 0 when MSSQL procedure has IF clause

```
UCHAR plm_szSqlState[MAXBUFLEN] = "",
plm_szErrorMsg[MAXBUFLEN] = "";
SDWORD plm_pfNativeError = 0L;
SWORD plm_pcbErrorMsg = 0;
SQLSMALLINT plm_cRecNbr = 1;
SDWORD plm_SS_MsgState = 0, plm_SS_Severity = 0;
SQLINTEGER plm_Rownumber = 0;
USHORT plm_SS_Line;
SQLSMALLINT plm_cbSS_Procname, plm_cbSS_Srvname;
SQLCHAR plm_SS_Procname[MAXNAME], plm_SS_Srvname[MAXNAME];
```

```
if (logstring)
printf(logstring);
```

```
plm_retcode = SQLGetDiagRec(plm_handle_type, plm_handle,
plm_cRecNbr, plm_szSqlState, &plm_pfNativeError,
plm_szErrorMsg, MAXBUFLEN - 1, &plm_pcbErrorMsg);
```

```
printf("szSqlState = %s\n", plm_szSqlState);
printf("pfNativeError = %d\n", plm_pfNativeError);
printf("szErrorMsg = %s\n", plm_szErrorMsg);
printf("pcbErrorMsg = %d\n", plm_pcbErrorMsg);
}
```

```
void e (SQLRETURN rc, char *text)
{
switch (rc) {
case SQL_SUCCESS:
printf("%s: rc = SQL_SUCCESS\n", text);
return;

case SQL_SUCCESS_WITH_INFO:
printf("%s: rc = SQL_SUCCESS_WITH_INFO\n", text);
break;

case SQL_ERROR:
printf("%s: rc = SQL_ERROR\n", text);
break;

case SQL_INVALID_HANDLE:
printf("%s: rc = SQL_INVALID_HANDLE\n", text);
break;
}
}
.
```