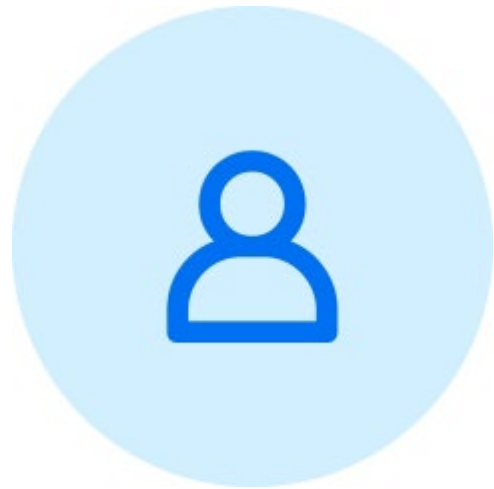


[Step-by-Step procedure for creation, execution and storing of ABAP Managed Database Procedures in HANA](#)



[former_member260021](#)

Discoverer

[Options](#)

2016 Nov 23 12:20 PM

[26 Kudos](#)

147,499

SAP Managed Tags: •

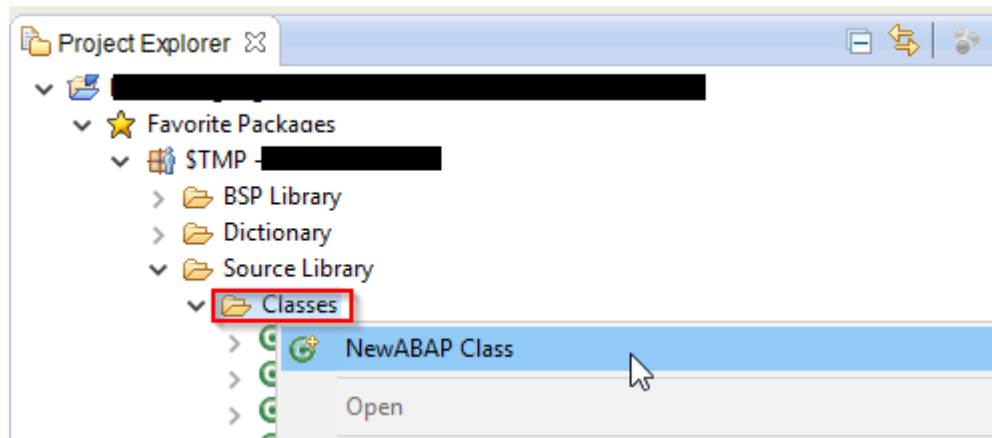
[ABAP Development](#) •

ABAP Managed Database Procedure

Database Procedures are stored and executed in the Database. We can create and execute database procedures in HANA database through ABAP using AMDP Class and AMDP Method called ABAP Managed Database Procedures. SQL SCRIPT is the language for creating stored procedures in HANA. Main benefit of using SQL Script is to allow the execution of complex calculations inside HANA database. The language is varies from one database system to another. The ABAP Managed Database procedures should be created using ABAP Development Tools (Eclipse or HANA Studio).

Creation of ABAP Managed Database Procedure in ABAP

1. Open ABAP Development Tool (Eclipse or HANA studio) and Go to ABAP Perspective. Create new ABAP Class.



2. Provide Name and Description. Click on NEXT Button.

New ABAP Class

ABAP Class
Create an ABAP class

Project * [Redacted] Browse...

Package: * STMP Browse...

☐ Add to favorite packages

Name: * ZCL_SALESORDER_DETAILS

Description: * To get Sales Order details using AMDP in HANA

Original Language: EN

Superclass

< Back Next > Finish Cancel

3. Click on Finish button.

The screenshot shows the "Selection of Transport Request" step in the SAP ABAP Class Wizard. The window title is "New ABAP Class". Below the title bar, there's a section titled "Selection of Transport Request" with an information icon and the message "No change recording enabled for package \$TMP".

There are two main options:

- ☐ Choose from requests in which I am involved
- ☐ Create a new request

Under the first option, there is a table with four columns: "Transport Request", "User", "Target", and "Text". The table is currently empty.

Under the second option, there are two input fields:

- "Request description:" followed by a text box.
- "Enter a request number" (selected) with a "Request number:" label, a text box, and a "Browse..." button.

At the bottom, there are navigation buttons: "< Back", "Next >", "Finish" (highlighted with a red rectangle), and "Cancel". A question mark icon is also present in the bottom left corner.

4. AMDP Class Definition

An AMDP is implemented in an AMDP class with a regular static method or instance method in any visibility section. The editing environment for AMDP is the ABAP class editor.

The AMDP class must contain the appropriate tag interface. **IF_AMDP_MARKER_HDB** is Marker Interface for DB Procedures.

Example:

- a. In Class Definition provide interface **IF_AMDP_MARKER_HDB**.
- b. Define the table type **TT_ORDER** and structure type **TY_ORDER**.
- c. Define the method **GET_SALESORDER_DETAILS** (Method parameters should be Passed by value).

```

▶ ZCL_SALESORDER_DETAILS ▶
CLASS zcl_salesorder_details DEFINITION
    PUBLIC
    FINAL
    CREATE PUBLIC .
    PUBLIC SECTION.

    * Marker interface for Database Procedures
    INTERFACES: IF_AMDP_MARKER_HDB.
    * Structure
    TYPES:
        BEGIN OF ty_order,
            vbeln      TYPE vbeln,      "Sales Order Number
            posnr       TYPE posnr_va,   "Item Number
            vkorg       TYPE vkorg,      "Sales Organization
            item_price  TYPE netwr_ap,    "Item Price
            status      TYPE char30,     "Delivery Status
        END OF ty_order.
    * Table type
    TYPES:
        tt_order TYPE STANDARD TABLE OF ty_order WITH EMPTY KEY.

    * Method Definition
    CLASS-METHODS get_salesorder_details
        IMPORTING
            VALUE(iv_vbeln) TYPE vbeln
        EXPORTING
            VALUE(et_order)  TYPE tt_order.

    PROTECTED SECTION.
    PRIVATE SECTION.
ENDCLASS.

```

Logic:

```

CLASS zcl_salesorder_details DEFINITION
    PUBLIC
    FINAL
    CREATE PUBLIC.
    PUBLIC SECTION.

    *Marker interface for Database Procedures
    INTERFACES: if_amdp_marker_hdb.
    *Structure
    TYPES:
        BEGIN OF ty_order,
            vbeln      TYPE vbeln, "Sales Order Number
            posnr       TYPE posnr_va, "Item Number
            vkorg       TYPE vkorg, "Sales Organization
            item_price  TYPE netwr_ap, "Item Price
            status      TYPE char30, "Delivery Status
        END OF ty_order.
    * Table type
    TYPES:
        tt_order TYPE STANDARD TABLE OF ty_order WITH EMPTY KEY.

```

* Method Definition

```
CLASS-METHODS get_salesorder_details
IMPORTING
VALUE(iv_vbeln) TYPE vbeln
EXPORTING
VALUE(et_order) TYPE tt_order.
PROTECTED SECTION.
PRIVATE SECTION.
ENDCLASS.
```

5. Implementaion of AMDP Method in AMDP Class

The screenshot shows the SAP IDE editor for the class `zcl_salesorder_details`. The code is as follows:

```
CLASS zcl_salesorder_details IMPLEMENTATION.
METHOD get_salesorder_details BY DATABASE PROCEDURE
    FOR HDB
    LANGUAGE SQLSCRIPT
    OPTIONS READ-ONLY
    USING vbak vbap vbup.

    *To get Sales Order details

    et_order = SELECT vbak.vbeln,
                      vbap.POSNR,
                      vbak.vkorg,
                      vbap.netwr as item_price,
                      CASE LFSTA
                        WHEN ' ' then 'Not Relevant'
                        WHEN 'A' then 'Not yet processed'
                        WHEN 'B' then 'Partially processed'
                        WHEN 'C' then 'Completely processed'
                      END AS status
    FROM vbak AS vbak
    INNER JOIN vbap AS vbap
      ON vbak.vbeln = vbap.vbeln
    INNER JOIN vbup AS vbup
      ON vbup.vbeln = vbap.vbeln AND vbup.posnr = vbap.posnr
    WHERE vbak.vbeln = iv_vbeln;

ENDMETHOD.
ENDCLASS.
```

Annotations in the image:

- Defines an AMDP procedure implementation for implementing a Database Procedure** (points to `BY DATABASE PROCEDURE`)
- HANA Database** (points to `FOR HDB`)
- Defines the database-specific language in which the AMDP is implemented.** (points to `LANGUAGE SQLSCRIPT`)
- Only reads are permitted on the database tables in the database procedure.** (points to `OPTIONS READ-ONLY`)

Logic:

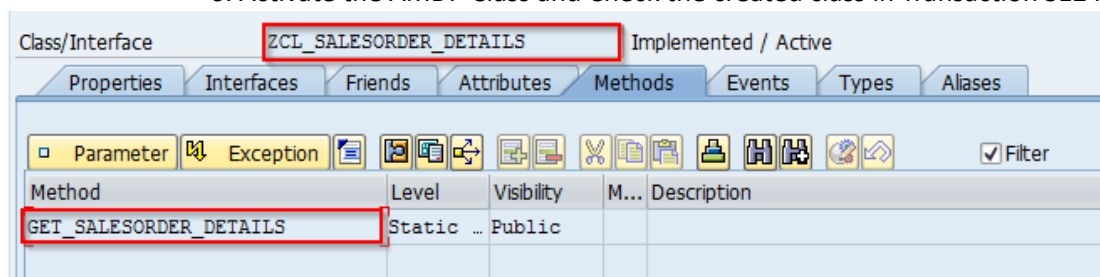
```
CLASS zcl_salesorder_details IMPLEMENTATION.
METHOD get_salesorder_details BY DATABASE PROCEDURE
    FOR HDB
    LANGUAGE SQLSCRIPT
    OPTIONS READ-ONLY
    USING vbak vbap vbup.
    *To get Sales Order details
    et_order = SELECT vbak.vbeln,
                      vbap.posnr,
                      vbak.vkorg,
                      vbap.netwr as item_price,
                      CASE LFSTA
```

```

        WHEN ' ' then 'Not Relevant'
        WHEN 'A' then 'Not yet processed'
        WHEN 'B' then 'Partially processed'
        WHEN 'C' then 'Completely processed'
        END AS status
FROM vbak AS vbak INNER JOIN vbap AS vbap
    ON vbak.vbeln = vbap.vbeln
    INNER JOIN vbup AS vbup
    ON vbup.vbeln = vbap.vbeln AND vbup.posnr = vbap.posnr
WHERE vbak.vbeln = iv_vbeln;ENDMETHOD.
ENDCLASS.

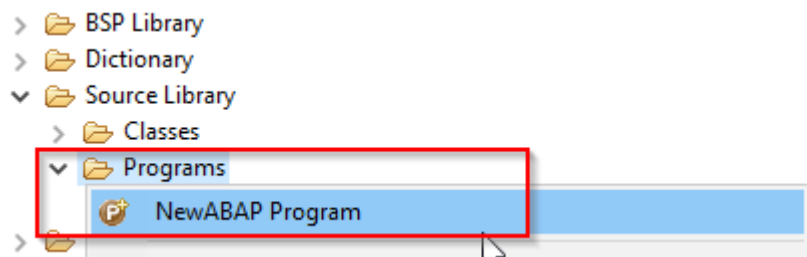
```

6. Activate the AMDP Class and Check the created class in Transaction SE24.



Execute the ABAP Managed Database Procedure through Report

1. Create a New ABAP Program.



2. Provide Name and Description. Click on NEXT button.

New ABAP Program

ABAP Program
Create an ABAP program

Project * Browse...

Package: * Browse...

☐ Add to favorite packages

Name: *

Description: *

Original Language:

☐ System Program

< Back **Next >** Finish Cancel

3. Click on Finish button.

New ABAP Program

Selection of Transport Request
 No change recording enabled for package STMP

☐ Choose from requests in which I am involved

Transport Request	User	Target	Text
-------------------	------	--------	------

☐ Create a new request

Request description:

☐ Enter a request number

Request number: Browse...

< Back Next > **Finish** Cancel

4. Call the AMDP Method in ABAP editor.

```
[ME1] ZR_CALL_AMDP ⌕
▶ P ZR_CALL_AMDP ▶
REPORT zr_call_amdp.

PARAMETER p_vbeln TYPE vbeln.

* To Call AMDP Method
zcl_salesorder_details=>get_salesorder_details(
    EXPORTING iv_vbeln = p_vbeln
    IMPORTING et_order = data(lt_order) ).


* To display Sales Order Details
cl_demo_output=>display_data( name = 'Sales Order Details'
    value = lt_order ).
```

Logic:

```
REPORT zr_call_amdp.
PARAMETER p_vbeln TYPE vbeln.
* To Call AMDP Method
zcl_salesorder_details=>get_salesorder_details(
    EXPORTING iv_vbeln = p_vbeln
    IMPORTING et_order = data(lt_order) ).
* To display Sales Order Details
cl_demo_output=>display_data( name = 'Sales Order Details'
    value = lt_order ).
```

Output:

Provide the sales order number as the input

<i>To call AMDP Method in Report</i>	
	
Sales Order Number	0000009531

Output				
Sales Order Details				
VBELN	POSNR	VKORG	ITEM_PRICE	STATUS
0000009531	000010	3020	14763.0	Partially processed
0000009531	000030	3020	34684.0	Completely processed
0000009531	000040	3020	16390.7	Completely processed
0000009531	000020	3020	18815.1	Partially processed

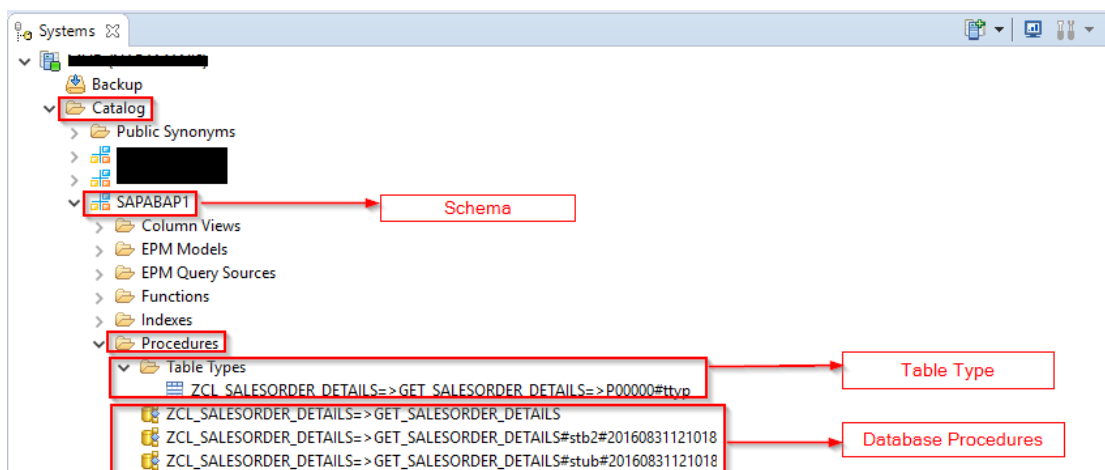
Stored in HANA Database

1. Check the DB Connection in tcode DBACOCKPIT.

State	System	DB System	DB Release	DB Host	Release	Connection Name	DB User	RFC Destination	Default
✓		SAP HANA database					SAPABAP1		

2. Database procedure will create in HANA DB at the first call of AMDP Method .
3. Go to SAP HANA Development perspective --> HANA DB System --> Catalog --> Schema --> Procedures.

The AMDP Method Implementation will be stored as Database procedure and Table Types of AMDP Class also stored under Schema 'SAPABAP1'.



4. The Table Type 'TT_ORDER ' of AMDP Class will be stored as "ZCL_SALESORDER_DETAILS=>GET_SALESORDER_DETAILS=>P00000#ttyp"

Table Name:			Schema:		
ZCL_SALESORDER_DETAILS=>GET_SALESORDER_DETAILS=>P00000#tty			SAPABAP1		
Columns	Indexes	Further Properties	Runtime Information		
	Name	SQL Data Type	Di...	Column Store Data Type	Key
1	VBELN	NVARCHAR	10	STRING	
2	POSNR	NVARCHAR	6	STRING	
3	VKORG	NVARCHAR	4	STRING	
4	ITEM_PRICE	DECIMAL	15,2	FIXED	
5	STATUS	NVARCHAR	30	STRING	

5.The AMDP Method 'GET_SALESORDER_DETAILS' of AMDP Class 'ZCL_SALESORDER_DETAILS' will be stored as Database procedure 'ZCL_SALESORDER_DETAILS=>GET_SALESORDER_DETAILS' as shown below.

```

SQLScript
create procedure
  "ZCL_SALESORDER_DETAILS=>GET_SALESORDER_DETAILS"
(
  in   "IV_VBELN" NVARCHAR (000010),
  out  "ET_ORDER" "ZCL_SALESORDER_DETAILS=>GET_SALESORDER_DETAILS=>P00000#tty"
)
language sqlscript sql security invoker reads sql data as begin

--To get Sales Order details

et_order = SELECT vbak.vbeln,
                  vbap.POSNR,
                  vbak.vkorg,
                  vbap.netwr as item_price,
                  CASE LFSTA
                    WHEN ' ' then 'Not Relevant'
                    WHEN 'A' then 'Not yet processed'
                    WHEN 'B' then 'Partially processed'
                    WHEN 'C' then 'Completely processed'
                  END AS status
FROM "ZCL_SALESORDER_DETAILS=>VBAK#covw" AS vbak
INNER JOIN "ZCL_SALESORDER_DETAILS=>VBAP#covw" AS vbap
  ON vbak.vbeln = vbap.vbeln
INNER JOIN "ZCL_SALESORDER_DETAILS=>VBUP#covw" AS vbup
  ON vbup.vbeln = vbap.vbeln AND vbup.posnr = vbap.posnr
WHERE vbak.vbeln = iv_vbeln;

end;

```

Logic:

```

create procedure
  "ZCL_SALESORDER_DETAILS=>GET_SALESORDER_DETAILS"
(
  in "IV_VBELN" NVARCHAR (000010),
  out "ET_ORDER" "ZCL_SALESORDER_DETAILS=>GET_SALESORDER_DETAILS=>P00000#tty"
)
language sqlscript sql security invoker reads sql data as begin
  --To get Sales Order details

```

```

et_order = SELECT vbak.vbeln,
                vbap.posnr,
                vbak.vkorg,
                vbap.netwr as item_price,
                CASE LFSTA
                WHEN ' ' then 'Not Relevant'
                WHEN 'A' then 'Not yet processed'
                WHEN 'B' then 'Partially processed'
                WHEN 'C' then 'Completely processed'
                END AS status
FROM "ZCL_SALESORDER_DETAILS=>VBAK#covw" AS vbak
INNER JOIN "ZCL_SALESORDER_DETAILS=>VBAP#covw" AS vbap
        ON vbak.vbeln = vbap.vbeln
INNER JOIN "ZCL_SALESORDER_DETAILS=>VBUP#covw" AS vbup
        ON vbup.vbeln = vbap.vbeln AND vbup.posnr = vbap.posnr
WHERE vbak.vbeln = iv_vbeln;
end;

```

6. ZCL_SALESORDER_DETAILS=>GET_SALESORDER_DETAILS#stb2#20160831121018

and
ZCL_SALESORDER_DETAILS=>GET_SALESORDER_DETAILS#stub#20160831121018 ar
e for calling Database procedure
"ZCL_SALESORDER_DETAILS=>GET_SALESORDER_DETAILS"

SQLScript

```

create procedure
    "ZCL_SALESORDER_DETAILS=>GET_SALESORDER_DETAILS#stb2#20160831121018"
(
    in    "IV_VBELN" NVARCHAR (000010)
)
language sqlscript sql security invoker reads sql data as begin

    call "ZCL_SALESORDER_DETAILS=>GET_SALESORDER_DETAILS" (
        "IV_VBELN" => :IV_VBELN ,
        "ET_ORDER" => :ET_ORDER
    );
    select * from :ET_ORDER;

end;

```

Logic:

```

create procedure
    "ZCL_SALESORDER_DETAILS=>GET_SALESORDER_DETAILS#stb2#20160831121018"
(
    in "IV_VBELN" NVARCHAR (000010)
)
language sqlscript sql security invoker reads sql data as begin
call "ZCL_SALESORDER_DETAILS=>GET_SALESORDER_DETAILS" (
    "IV_VBELN" => :IV_VBELN ,

```

```

"ET_ORDER" => :ET_ORDER
);
select * from :ET_ORDER;
end;

```

```

SQLScript
create procedure
  "ZCL_SALESORDER_DETAILS=>GET_SALESORDER_DETAILS#stub#20160831121018"
(
  in   "IV_VBELN" NVARCHAR (000010)
)
language sqlscript sql security invoker reads sql data as begin

  call "ZCL_SALESORDER_DETAILS=>GET_SALESORDER_DETAILS" (
    "IV_VBELN" => :IV_VBELN ,
    "ET_ORDER" => :ET_ORDER
  );
  select * from :ET_ORDER;
end;

```

Logic:

```

create procedure
  "ZCL_SALESORDER_DETAILS=>GET_SALESORDER_DETAILS#stub#20160831121018"
(
  in "IV_VBELN" NVARCHAR (000010)
)
language sqlscript sql security invoker reads sql data as begin
call "ZCL_SALESORDER_DETAILS=>GET_SALESORDER_DETAILS" (
  "IV_VBELN" => :IV_VBELN ,
  "ET_ORDER" => :ET_ORDER
);
select * from :ET_ORDER;
end;

```

7. The database tables VBAK VBAP and VBUP are used in AMDP Method will be created as VIEWS in HANA Database system.
- i) **ZCL_SALESORDER_DETAILS=>VBAK#covw**

View Name:

Schema:

ZCL_SALESORDER_DETAILS=>VBAK#covw

SAPABAP1

Columns Create Statement

	View Column	Table Column	SQL Data Type	Dimension	Not Null	Default
1	MANDT	MANDT	NVARCHAR	3	X	000
2	VBELN	VBELN	NVARCHAR	10	X	
3	ERDAT	ERDAT	NVARCHAR	8	X	00000000
4	ERZET	ERZET	NVARCHAR	6	X	000000
5	ERNAM	ERNAM	NVARCHAR	12	X	
6	ANGDT	ANGDT	NVARCHAR	8	X	00000000
7	BNDDT	BNDDT	NVARCHAR	8	X	00000000
8	AUDAT	AUDAT	NVARCHAR	8	X	00000000
9	VBTP	VBTP	NVARCHAR	1	X	
10	TRVOG	TRVOG	NVARCHAR	1	X	
11	AUART	AUART	NVARCHAR	4	X	
12	AUGRU	AUGRU	NVARCHAR	3	X	
13	GWLDT	GWLDT	NVARCHAR	8	X	00000000
14	SUBMI	SUBMI	NVARCHAR	10	X	

ii) ZCL_SALESORDER_DETAILS=>VBAP#covw

View Name:

Schema:

ZCL_SALESORDER_DETAILS=>VBAP#covw

SAPABAP1

Columns Create Statement

	View Column	Table Column	SQL Data Type	Dimension	Not Null
1	MANDT	MANDT	NVARCHAR	3	X
2	VBELN	VBELN	NVARCHAR	10	X
3	POSNR	POSNR	NVARCHAR	6	X
4	MATNR	MATNR	NVARCHAR	18	X
5	MATWA	MATWA	NVARCHAR	18	X
6	PMATN	PMATN	NVARCHAR	18	X
7	CHARG	CHARG	NVARCHAR	10	X
8	MATKL	MATKL	NVARCHAR	9	X
9	ARKTX	ARKTX	NVARCHAR	40	X
10	PSTYV	PSTYV	NVARCHAR	4	X
11	POSAR	POSAR	NVARCHAR	1	X
12	LFREL	LFREL	NVARCHAR	1	X
13	FKREL	FKREL	NVARCHAR	1	X
14	UEPOS	UEPOS	NVARCHAR	6	X
15	GRPOS	GRPOS	NVARCHAR	6	X
16	ABGRU	ABGRU	NVARCHAR	2	X
17	PRODH	PRODH	NVARCHAR	18	X
18	ZWERT	ZWERT	DECIMAL	13,2	X

iii) ZCL_SALESORDER_DETAILS=>VBUP#covw

View Name:



Schema:

ZCL_SALESORDER_DETAILS=>VBUP#covw

SAPABAP



Columns

Create Statement

	View Column	Table Column	SQL Data Type	Dimension	Not N
1	MANDT	MANDT	NVARCHAR	3	X
2	VBELN	VBELN	NVARCHAR	10	X
3	POSNR	POSNR	NVARCHAR	6	X
4	RFSTA	RFSTA	NVARCHAR	1	X
5	RFGSA	RFGSA	NVARCHAR	1	X
6	BESTA	BESTA	NVARCHAR	1	X
7	LFSTA	LFSTA	NVARCHAR	1	X
8	LFGSA	LFGSA	NVARCHAR	1	X
9	WBSTA	WBSTA	NVARCHAR	1	X
10	FKSTA	FKSTA	NVARCHAR	1	X
11	FKSAA	FKSAA	NVARCHAR	1	X
12	ABSTA	ABSTA	NVARCHAR	1	X
13	GBSTA	GBSTA	NVARCHAR	1	X
14	KOSTA	KOSTA	NVARCHAR	1	X
15	LVSTA	LVSTA	NVARCHAR	1	X
16	UVALL	UVALL	NVARCHAR	1	X
17	UVVLK	UVVLK	NVARCHAR	1	X
18	UVFAK	UVFAK	NVARCHAR	1	X

