

Analyze AMDP Performance



Details

Code Snippets

[// Explore More Tutorials](#)

Analyze AMDP Performance

[Julie Plummer](#) 03/11/2019

Beginner ⌚ 45 min.

 [SAP Cloud Platform, ABAP environment, Tutorial, SAP Cloud Platform, Beginner, ABAP Development](#)

Analyze the runtime performance of AMDPs and the executed SQL statements using the AMDP Profiler in ABAP Development Tools (ADT).

You will learn

- ✓ How to create an ABAP class containing an ABAP Managed Database Procedure (AMDP)
- ✓ How to run the ABAP Profiler on this class

Throughout this tutorial, objects name include the suffix **XXX**. Always replace this with your group number or initials.

You should be familiar with ABAP Managed Database Procedures (AMDP). Briefly, AMDP allows you to optimize your ABAP code (for ABAP on SAP HANA) by calling HANA database procedures from a global ABAP class. For more details, see:

- [SAP Help Portal: ABAP Managed Database Procedures \(AMDP\)](#)
- [ABAP Keyword Documentation: AMDP - ABAP Managed Database Procedures](#)

1 Step 1: Install the Flight Reference Scenario using abapGit

1. Create the package `/DMO/FLIGHT` as a sub-package under the package `/DMO/SAP` (keep the default values). **IMPORTANT:** Make sure that the software component is also `/DMO/SAP`.
2. Open the **abapGit** view by choosing **Window > Show View > Other... > abapGit Repositories**. Make sure you have the correct ABAP Cloud Project marked (See the little headline in the **abapGit** view for the current project.)

3. Clone a repository by choosing **+** and enter the repository URL:
<https://github.com/SAP/abap-platform-refscen-flight.git>.
4. On the next page choose the master branch and provide the package **/DMO/FLIGHT**.
5. Create or assign a transport request and choose **Finish**. This starts the cloning of the repository, which might take a few minutes
6. Once the cloning has finished, refresh your project tree.
7. Finally, generate some sample data by running the ABAP class **/DMO/CL_FLIGHT_DATA_GENERATOR** by choosing **Run as Console Application (F9)**.

 Done

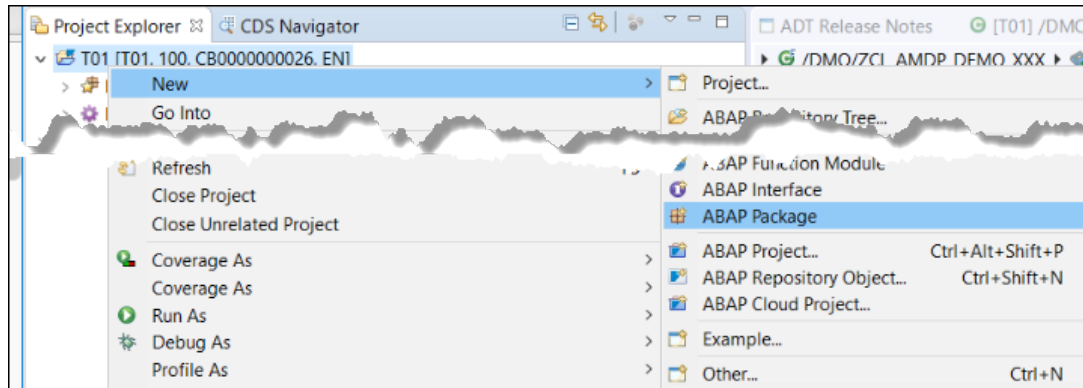
Log on to answer question

2

Step 2: Create an ABAP package in **/DMO/SAP**

One of the restrictions of the ABAP Environment on SAP Cloud Platform is that you can only use other objects if they are released objects (whitelisted) or in the same software component. Since you will be using objects in the **/DMO/SAP** software component, you need to create your package in **/DMO/SAP**.

1. Select your project. From the context menu, choose **New > ABAP Package**.



2. Enter a name and description for your package, then choose **Next**. **IMPORTANT:** Make sure that you prefix the name with **/DMO/**, for example: Name = **/DMO/AMDP_XXX** (replacing **XXX** with your group number or initials)

Project: *	<input type="text" value="T01"/>	<input type="button" value="Browse..."/>
Name: *	<input type="text" value="/DMO/AMDP_XXX"/>	
Description: *	<input type="text" value="Test AMDP in ADT 3.0"/>	
Original Language:	<input type="text" value="EN"/>	
Superpackage:	<input type="text"/>	
Package Type: *	<input type="text" value="Development"/>	

3. Enter the software component **/DMO/SAP**. You can leave the other fields blank.

New ABAP Package

ABAP Package

Select basic package properties

Software Component:

Application Component:

Transport Layer:

4. Create or assign a transport request and choose **Finish**.

Done

Log on to answer question

3 Step 3: Create an ABAP class

1. Again, from the context menu of your package, choose **New > ABAP Class**.
2. Enter a name and description, then choose **Next**. Make sure that you prefix the class name with `/DMO/`, for example:
 - Name: `/DMO/ZCL_AMDP_DEMO_XXX`
 - Description: AMDP Demo w Flight ref

3. Assign the transport request and choose **Finish**.

 Done

Log on to answer question

4

Step 4: Add two interfaces

Add two interfaces by adding this code to the public section.

- **if_amdp_marker_hdb** defines the class as an AMDP class, allowing you to implement AMDP methods - that is, ABAP methods that call a SAP HANA database procedure from within a global ABAP class.
- **if_oo_adt_classrun** allows you to output the results to the ABAP Console.

ABAP

Copy

```
1 | INTERFACES: if_amdp_marker_hdb,  
2 |   if_oo_adt_classrun.  
3 |
```

 Done

Log on to answer question

5

Step 5: Create structures and table types

Add these structures and types to the public section, just after the interface definitions. Note the data elements that you imported earlier.

ABAP

[Copy](#)

```
1  TYPES:
2    BEGIN OF ty_result_line,
3      airline          TYPE /dmo/carrier_name,
4      flight_connection TYPE /dmo/connection_id,
5      old_price         TYPE /dmo/flight_price,
6      old_currency      TYPE /dmo/currency_code,
7      new_price         TYPE /dmo/flight_price,
8      new_currency      TYPE /dmo/currency_code,
9    END OF ty_result_line,
10
11   BEGIN OF ty_flights_line,
12     airline          TYPE /dmo/carrier_name,
13     flight_connection TYPE /dmo/connection_id,
14     price            TYPE /dmo/flight_price,
15     currency         TYPE /dmo/currency_code,
16   END OF ty_flights_line,
17
18   ty_result_table  TYPE STANDARD TABLE OF ty_result_line WITH EMPTY
19   ty_flights_table TYPE STANDARD TABLE OF ty_flights_line WITH EMPTY
20   ty_flights       TYPE STANDARD TABLE OF /dmo/flight.
21
```

Done

Log on to answer question

6

Step 6: Add method definitions

Add these two method definitions to your code. Ignore the errors for now.

ABAP

[Copy](#)

```
1  METHODS:
2      get_flights
3          EXPORTING
4              VALUE(result) TYPE ty_result_table
5          RAISING   cx_amdp_execution_error,
6
7      convert_currency
8          IMPORTING
9              VALUE(flights) TYPE ty_flights_table
10         EXPORTING
11             VALUE(result) TYPE ty_result_table
12         RAISING   cx_amdp_execution_error.
13
14
```

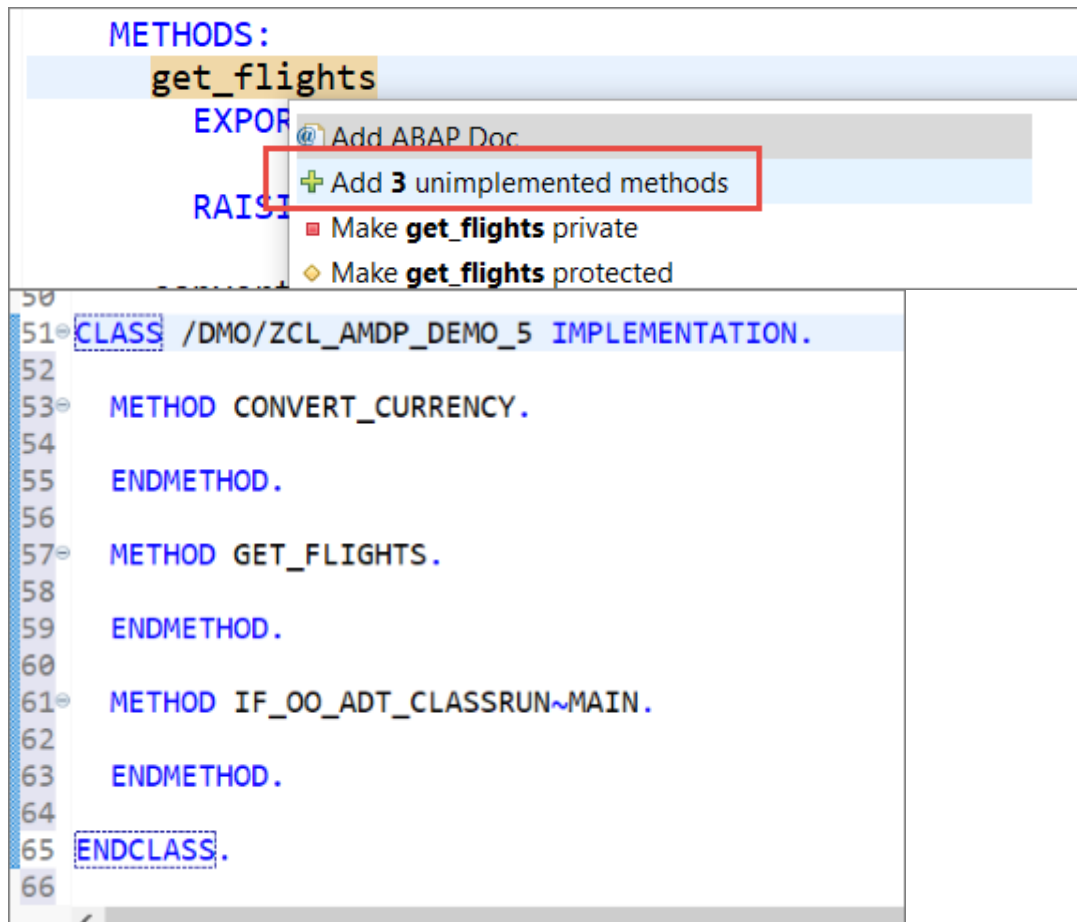

Both of these are AMDP methods.

 Done

Log on to answer question

7 Step 7: Implement get_flights

1. In the class definition, select any one of the methods and choose **Add 3 unimplemented methods**. All three (empty) implementations will appear in the class implementation.



2. Add the following to the method `get_flights`. Both this and `convert_currency` are SQLScript.

For more information on SAP HANA SQLScript, see [SAP HANA SQLScript Reference](#)

ABAP

Copy

```
1 METHOD GET_FLIGHTS by database procedure  
2 for hdb
```

```
3 | language sqlscript
4 | options read-only
5 | using
6 |     /dmo/flight
7 |     /dmo/carrier
8 |     /dmo/zcl_ampd_demo_xxx=>convert_currency.
9 |
```

You must specify all ABAP tables, views, and AMDP procedures in the USING clause. For more details on these clauses, see [ABAP Keyword Documentation: Method - By Database Procedure, Function](#)

3. Add the SELECT statement. Ignore the warning for now.

ABAP

Copy

```
1 | flights = select distinct
2 |
3 |     from "/DMO/FLIGHT" as f
4 |     inner join "/DMO/CARRIER" as c
5 |         on f.carrier_id = c.carrier_id;
6 |
```

4. Add the fields. You can do this using Auto-complete (Ctrl+1), to make sure you are using the correct field names.

ABAP

Copy

```
1 | flights = select distinct
2 |     c.name as airline,
3 |     f.connection_id as flight_connection,
```

```

4   f.price    as price,
5   f.currency_code as currency
6   from "/DMO/FLIGHT" as f
7   inner join "/DMO/CARRIER" as c
8       on f.carrier_id = c.carrier_id;
9

```

5. Finally, call the other AMDP method, `convert_currency`. Your method should look like this:

ABAP

[Copy](#)

```

1  METHOD get_flights BY DATABASE PROCEDURE
2      FOR HDB
3      LANGUAGE SQLSCRIPT
4      OPTIONS READ-ONLY
5      USING
6          /dmo/flight
7          /dmo/carrier
8          /dmo/zcl_amdp_demo_xxx=>convert_currency.
9
10 flights = select distinct
11     c.name as airline,
12     f.connection_id as flight_connection,
13     f.price    as price,
14     f.currency_code as currency
15     from "/DMO/FLIGHT" as f
16     inner join "/DMO/CARRIER" as c
17         on f.carrier_id = c.carrier_id;
18
19 call "/DMO/ZCL_AMDP_DEMO_XXX=>CONVERT_CURRENCY"( :flights, re
20

```

21

ENDMETHOD.

 Done

Log on to answer question

8

Step 8: Implement the method convert_currency

Similarly, implement the convert_currency method.

ABAP

[Copy](#)

```
1  METHOD convert_currency BY DATABASE PROCEDURE
2    FOR HDB
3    LANGUAGE SQLSCRIPT
4    OPTIONS READ-ONLY.
5
6    declare today date;
7    declare new_currency nvarchar(3);
8
9    select current_date into today from dummy;
10   new_currency := 'EUR';
11
12   result = select distinct
```

```
13  airline,  
14  flight_connection,  
15  price    as old_price,  
16  currency as old_currency,  
17  convert_currency(  
18    "AMOUNT"      => price,  
19    "SOURCE_UNIT"  => currency,  
20    "TARGET_UNIT"  => :new_currency,  
21    "REFERENCE_DATE" => :today,  
22    "CLIENT"       => '100',  
23    "ERROR_HANDLING" => 'set to null',  
24    "SCHEMA"       => current_schema  
25  ) as new_price,  
26  :new_currency as new_currency  
27  from :flights;  
28  
29  ENDMETHOD.  
30
```

 Done

Log on to answer question

9

Step 9: Implement the method `main` of the interface `if_oo_adt_classrun`

Finally, implement the `main` method of the interface `if_oo_adt_classrun`. This will allow you to output your results to the ABAP Console.

1. Call the method `get_flights` from the current instance of the class:

ABAP

[Copy](#)

```
1 | me->get_flights(  
2 |     IMPORTING  
3 |     result = DATA(lt_result) ).
```

2. Output the result to the console:

```
out->write( lt_result ).
```

3. Wrap this in an exception:

ABAP

[Copy](#)

```
1 | TRY.  
2 |  
3 | CATCH cx_amdp_execution_error INTO DATA(lx_amdp).  
4 |     out->write( lx_amdp->get_longtext( ) ).  
5 |
```

Your method should now look like this:

ABAP

[Copy](#)

```
1 METHOD if_oo_adt_classrun~main.
2
3 TRY.
4     me->get_flights(
5         IMPORTING
6         result = DATA(lt_result) ).
7
8     CATCH cx_amdp_execution_error INTO DATA(lx_amdp).
9         out->write( lx_amdp->get_longtext( ) ).
10 ENDTRY.
11
12 out->write( lt_result ).
13
14 ENDMETHOD.
15
```

 Done

Log on to answer question

10

Step 10: Check your code

Your code should look like this.

ABAP

[Copy](#)

```
1 CLASS /dmo/zcl_amdp_demo_xxx DEFINITION
2     PUBLIC
```



```

3  FINAL
4  CREATE PUBLIC .
5
6  PUBLIC SECTION.
7      INTERFACES: if_amdp_marker_hdb,
8                  if_oo_adt_classrun.
9
10     TYPES:
11         BEGIN OF ty_result_line,
12             airline          TYPE /dmo/carrier_name,
13             flight_connection TYPE /dmo/connection_id,
14             old_price        TYPE /dmo/flight_price,
15             old_currency     TYPE /dmo/currency_code,
16             new_price        TYPE /dmo/flight_price,
17             new_currency     TYPE /dmo/currency_code,
18         END OF ty_result_line,
19
20         BEGIN OF ty_flights_line,
21             airline          TYPE /dmo/carrier_name,
22             flight_connection TYPE /dmo/connection_id,
23             price            TYPE /dmo/flight_price,
24             currency         TYPE /dmo/currency_code,
25         END OF ty_flights_line,
26
27         ty_result_table TYPE STANDARD TABLE OF ty_result_line WITH EM
28         ty_flights_table TYPE STANDARD TABLE OF ty_flights_line WITH E
29         ty_flights      TYPE STANDARD TABLE OF /dmo/flight.
30
31     METHODS:
32         get_flights
33             EXPORTING
34                 VALUE(result) TYPE ty_result_table
35             RAISING    cx_amdp_execution_error,
36

```

```

37         convert_currency
38         IMPORTING
39             VALUE(flights) TYPE ty_flights_table
40         EXPORTING
41             VALUE(result) TYPE ty_result_table
42         RAISING    cx_amdp_execution_error.
43
44     PROTECTED SECTION.
45     PRIVATE SECTION.
46 ENDCLASS.
47
48
49 CLASS /dmo/zcl_amdp_demo_xxx IMPLEMENTATION.
50
51     METHOD get_flights BY DATABASE PROCEDURE
52     FOR HDB
53     LANGUAGE SQLSCRIPT
54     OPTIONS READ-ONLY
55     USING
56         /dmo/flight
57         /dmo/carrier
58         /dmo/zcl_amdp_demo_xxx=>convert_currency.
59
60     flights = select distinct
61         c.name as airline,
62         f.connection_id as flight_connection,
63         f.price      as price,
64         f.currency_code as currency
65     from "/DMO/FLIGHT" as f
66     inner join "/DMO/CARRIER" as c
67         on f.carrier_id = c.carrier_id;
68
69     call "/DMO/ZCL_AMDP_DEMO_XXX=>CONVERT_CURRENCY"( :flights, resul
70

```

```
71 ENDMETHOD.  
72  
73  
74 METHOD convert_currency BY DATABASE PROCEDURE  
75     FOR HDB  
76     LANGUAGE SQLSCRIPT  
77     OPTIONS READ-ONLY.  
78  
79     declare today date;  
80     declare new_currency nvarchar(3);  
81  
82     select current_date into today from dummy;  
83     new_currency := 'EUR';  
84  
85     result = select distinct  
86         airline,  
87         flight_connection,  
88         price    as old_price,  
89         currency as old_currency,  
90         convert_currency(  
91             "AMOUNT"      => price,  
92             "SOURCE_UNIT" => currency,  
93             "TARGET_UNIT" => :new_currency,  
94             "REFERENCE_DATE" => :today,  
95             "CLIENT"      => '100',  
96             "ERROR_HANDLING" => 'set to null',  
97             "SCHEMA"      => current_schema  
98         ) as new_price,  
99         :new_currency as new_currency  
100     from :flights;  
101  
102 ENDMETHOD.  
103  
104
```

```
105 METHOD if_oo_adt_classrun~main.
106
107 TRY.
108     me->get_flights(
109         IMPORTING
110         result = DATA(lt_result) ).
111
112     CATCH cx_amdp_execution_error INTO DATA(lx_amdp).
113         out->write( lx_amdp->get_longtext( ) ).
114     ENDTRY.
115
116     out->write( lt_result ).
117
118 ENDMETHOD.
119 ENDCLASS.
120
```

 Done


Log on to answer question

11 Step 11: Save, activate, and test your code

1. Save and activate your code by choosing **Ctrl+S**, **Ctrl+3**.
2. Optional: Test your class by running it the ABAP Console (**F9**).

Your output should look like this:

Global Class Class-relevant Local Types Local Types Test Classes Macros					
Problems Properties Templates Bookmarks Task Repositories Task List Feed Reader Transport Organizer Console					
ABAP Console					
Table					
AIRLINE	FLIGHT_CONNECTION	OLD_PRICE	OLD_CURRENCY	NEW_PRICE	NEW_CURRENCY
Singapore Airlines Limited	0001	10818.0	SGD	6596.34	EUR
Singapore Airlines Limited	0001	5950.0	SGD	3628.04	EUR
Singapore Airlines Limited	0002	11765.0	SGD	7173.78	EUR
Singapore Airlines Limited	0002	10953.0	SGD	6678.65	EUR
Singapore Airlines Limited	0011	2359.0	SGD	1438.41	EUR
Singapore Airlines Limited	0011	4880.0	SGD	2975.6	EUR
Singapore Airlines Limited	0012	4665.0	SGD	2844.51	EUR
Singapore Airlines Limited	0012	2574.0	SGD	1569.51	EUR
United Airlines, Inc.	0058	6629.0	USD	7052.12	EUR
United Airlines, Inc.	0058	4996.0	USD	5314.89	EUR
United Airlines, Inc.	0059	4131.0	USD	4394.68	EUR
United Airlines, Inc.	0059	6053.0	USD	6439.36	EUR

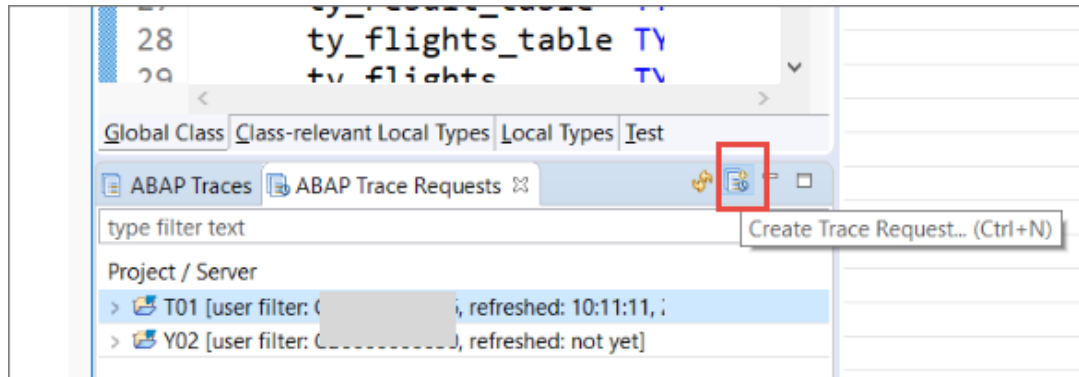
 Done

Log on to answer question

12

Step 12: Create an ABAP trace request

1. Switch to the ABAP Profiling perspective.
2. Open the ABAP Trace Requests view.
3. Choose your project and choose **Create Trace Request...** from the toolbar.



4. Choose the following options, then choose **Next**:

- Which requests: Any
- Object type: Any
- Server: All servers
- Maximum trace files... : 3
- Title: Any for AMDP Demo

▼ Which requests do you want to trace?

Any

Limit to

Object type: Any

Name: Browse...

User:

Client: 100

Server: All servers

▼ Which restrictions should apply?

Maximum number of trace files per server: 3

Schedule expires at: 21.02.2019 12:10:56

▼ Title of the trace file

AMDP test 1

5. On the next screen, leave the other defaults, choose **Enable AMDP trace**, then choose **Finish**.

▼ Perform aggregated measurement?

☒ No, I need the Call Sequence (large file size)

☐ Yes, I need the Aggregated Call Tree (medium file size)

☐ Yes, Hit List is sufficient (small file size)

▼ Which ABAP statements should be traced?

☒ Procedural units, SQL

☐ Procedural units, SQL, internal tables

☐ Only procedural units

☐ Custom statements:

▼ Details

☒ Procedural units

☒ SQL database access

☐ Access to internal tables

☐ Dynpro events

☐ Other ABAP events

☐ System and kernel events

▼ When should the trace start?

☒ Immediately

☐ Explicitly switch on and off (e.g. within Debugger)

▼ Advanced parameters

Maximum execution time: 30 minutes

Maximum file size: 30 MB

☐ Trace RFC and update requests

▼ AMDP trace options

☒ Enable AMDP trace

Procedure filter:

Log on to answer question

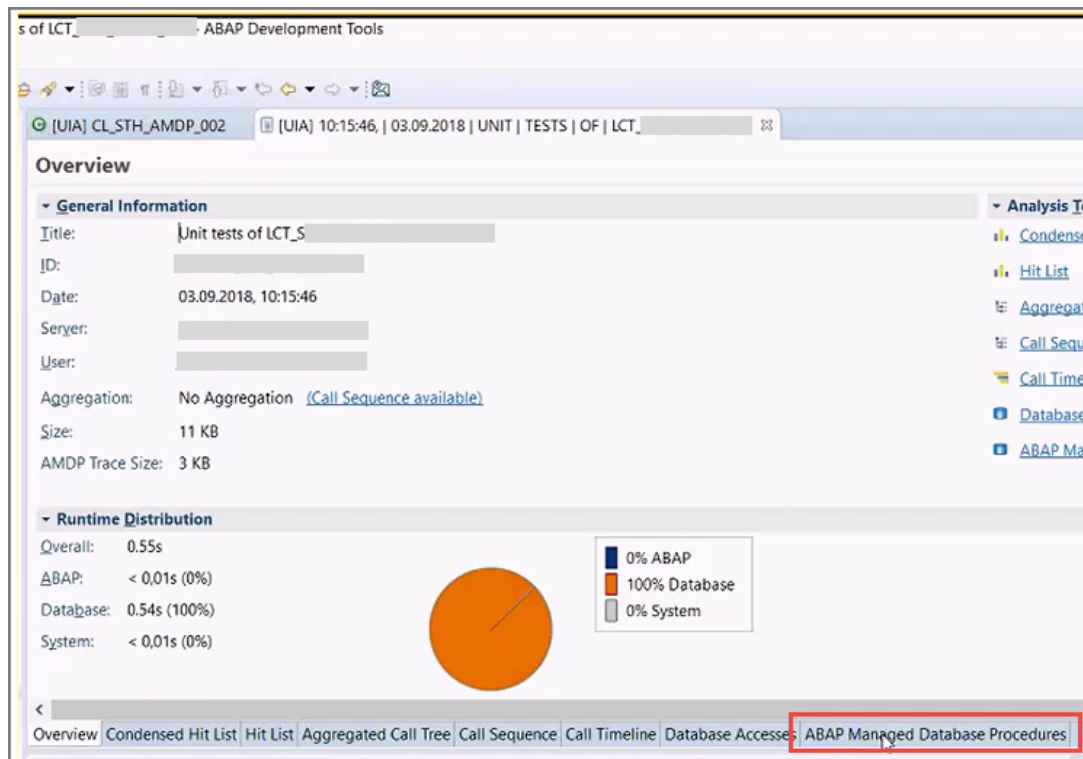
13

Step 13: Examine the AMDP Profiling Result

1. Refresh your trace request. Once it is finished, return to the ABAP Traces View.
2. Open the Profiling Overview by choosing (double-clicking) your trace.

Overview		Condensed Hit List	Hit List	Aggregated Call Tree	Aggregated Timeline	Call Se
ABAP Traces		ABAP Trace Requests				
type filter text						
Project / Trace Title						Time
Y02 [user filter: CB0000000050, refreshed: 10:03:02, 21.02.2019]						
✓ T01 [user filter: CB0000000026, refreshed: 11:23:45, 21.02.2019]						
Any for AMDP Demo						10:10:
Any for AMDP Demo						10:11:
Any for AMDP Demo						10:11:

3. Choose the ABAP Database Managed Procedures tab from the bottom.



4. The ABAP Managed Database Procedures overview displays the result of the AMDP profiling analysis in a table.

To get more details about the statement which has been executed at ABAP runtime, view the table and expand the relevant procedure nodes.

5. Optional: To navigate to the relevant position within your ABAP source code, double-click the corresponding statement. The development object is then opened and the cursor is positioned at the relevant position.

You can now analyze the results of your AMDP profiling. For more details, see:- [AMDP Profiling- Understanding AMDP Profiling Results](#)

Log on to answer question

14

Step 14: Test yourself

Define an AMDP method `get_sales_orders` (based on the AMDP method `get_flights`). Specify the options:

- Database = SAP HANA
- Language = SQLScript
- Options = read-only
- Objects used = `snwd_so`, `snwd_so_i`, and the method `read_sales_orders` of the class `zc1_demo`.

Do not indent your code.

Enter your code in the box below and choose **Submit Answer**.

 Submit Answer

Log on to answer question ✕

Developer Products

[ABAP Platform](#)

[SAP Cloud Platform](#)

[SAP Data Hub](#)

[SAP HANA](#)

[SAP Web IDE](#)

[All Products](#)

Trials & Downloads

[ABAP Development Tools](#)

[SAP NetWeaver AS ABAP 7.51 SP02
Developer Edition on ASE](#)

[SAP Cloud Platform](#)

[SAP Data Hub, developer edition](#)

[SAP HANA](#)

[SAP Web IDE Full-Stack](#)

[All Trials & Downloads](#)

Site Information

[Privacy](#)

[Terms of Use](#)

[Legal Disclosure](#)

[Copyright](#)

[Trademark](#)

[Cookie Preference](#)

[Sitemap](#)



Text View

Newsletter

Share & Follow

