



Week 1: What's New in SAP HANA 2.0 SPS 03

Unit 3: SQLScript Language Enhancements

SQLScript Language Enhancements

Overview: What? Why?

SQLScript is

- Extension of ANSI standard SQL
- Language for creating stored procedures and user-defined functions in SAP HANA
- Set-based declarative SQL and imperative control flow constructs
- Suitable for mass data processing and OLTP scenarios as well

Advantages compared to plain SQL:

- Smaller, parameterized reusable chunks of code (procedures, functions)
- Well structured query logic (using table variables)
 - Improving maintainability
 - Gaining performance by parallel execution
 - Debuggable intermediate steps
- Supports invoker security mode

SQLScript Language Enhancements

Libraries: SQLScript user-defined libraries

Library is a set of related variables, procedures, and functions. A user-defined library is a library written by the user in SQLScript. Users can create their own library and utilize it in other procedures or functions.

- .hdblibrary artifact
- Public/Private access modes
- Library members
 - Scalar variables(handled via session variables)
 - Procedures
 - Functions

```
MasterData.hdblibrary x
1  LIBRARY "MasterData"
2  LANGUAGE SQLSCRIPT AS
3  BEGIN
4
5      public variable rowcount int;
6
7      public function "employee_exists"(
8          | in im_employee_id nvarchar(10) |
9          | returns res boolean as
10
11      begin
12          | declare lv_count int;
13          | select count(*) into lv_count from "MD.Employees"
14          | | | | | where employeeid = im_employee_id;
15          | if lv_count > 0 then
16          |     res = true;
17          | else
18          |     res = false;
19          | end if;
20      end;
21
22      public procedure "get_employee_data"(
23          | | | | | out ex_employees "MD.Employees" )
24      LANGUAGE SQLSCRIPT
25      SQL SECURITY INVOKER
26      AS
27      begin
28          | ex_employees = select * from "MD.Employees";
29          | rowcount = record_count(:ex_employees);
30      end;
```

SQLScript Language Enhancements

Libraries: SQLScript user-defined libraries – Continued

The library is designed to be used only in a SQLScript procedure or function and is not available in other SQL statements.

The user must declare the library via USING.

Variables, procedures, and functions can then be referenced or called via the assigned aliases.

Current Limitations

- Not possible to use library variables in SELECT INTO statement and INTO-clause of dynamic SQL
- Variables cannot be declared using LIKE
- Non-constant variables cannot have default value
- Table type library variable is not supported
- Library member function cannot be used in queries

```
get_master_data.hdbprocedure ×
1  PROCEDURE "get_master_data" (
2      out ex_emp_exists boolean,
3      out ex_employees "MD.Employees",
4      out ex_emp_count int,
5      out ex_products "MD.Products",
6      out ex_prod_count int,
7      out ex_business_partners "MD.BusinessPartner",
8      out ex_bp_count int)
9      LANGUAGE SQLSCRIPT
10     SQL SECURITY INVOKER
11     READS SQL DATA AS
12 BEGIN USING "MasterData" as MData;
13
14 ex_emp_exists = MData:"employee_exists"('199999');
15
16 call MData:"get_employee_data"( ex_employees );
17 ex_emp_count = MData:rowcount;
18
19 call MData:"get_product_data"( ex_products );
20 ex_prod_count = MData:rowcount;
21
22 call MData:"get_business_partner_data"( ex_business_partners );
23 ex_bp_count = MData:rowcount;
24
25 END;
```

SQLScript Language Enhancements

Libraries: SQLScript Sync built-in library

The SQLSCRIPT_SYNC library contains procedures for putting processes to sleep and waking them up:

- SLEEP_SECONDS – makes the current process wait for the given number of seconds
- WAKEUP_CONNECTION – resumes a process waiting due to SLEEP_SECONDS

```
1 DO
2   BEGIN USING SQLSCRIPT_SYNC as SYNCLIB;
3
4   CALL SYNCLIB:SLEEP_SECONDS(10);
5
6   END;
```

```
1 DO
2   BEGIN USING SQLSCRIPT_SYNC AS SYNCLIB;
3
4   DECLARE i integer;
5   lt_connections = SELECT CONNECTION_ID FROM M_SERVICE_THREADS
6                     WHERE LOCK_WAIT_NAME LIKE 'SQLSCRIPT_WAIT%';
7
8   FOR i in 1..record_count(:lt_connections) DO
9     CALL SYNCLIB:WAKEUP_CONNECTION(:lt_connections.CONNECTION_ID[:i]);
10  END FOR;
11
12  END
```

SQLScript Language Enhancements

Libraries: SQLScript String built-in library

The SQLSCRIPT_STRING library contains several functions which are useful for string manipulation.

SPLIT functions allow for splitting strings into variables, arrays, and intermediate tables

- SPLIT / SPLIT_REGEXPR
- SPLIT_TO_ARRAY / SPLIT_REGEXPR_TO_ARRAY
- SPLIT_TO_TABLE / SPLIT_REGEXPR_TO_TABLE

FORMAT functions support new Python-style formatting for formatting strings into variables, arrays, and intermediate tables

- FORMAT
- FORMAT_TO_ARRAY
- FORMAT_TO_TABLE
- TABLE_SUMMARY

```
1 PROCEDURE "test_string_library" (  
2     out ex_strout table(RESULT nvarchar(5000)),  
3     out ex_tabout table(PRODUCTID nvarchar(10), PRICE decimal(15,2)) )  
4 LANGUAGE SQLSCRIPT  
5 SQL SECURITY INVOKER  
6 READS SQL DATA AS  
7 BEGIN USING SQLSCRIPT_STRING AS LIB;  
8  
9     declare lv_index int;  
10    declare lv_productid string;  
11    declare lv_price string;  
12    DECLARE lt_products TABLE (PRODUCTID NVARCHAR(10), PRICE decimal(15,2) );  
13  
14    lt_products.PRODUCTID[1] = 'HT-1000';  
15    lt_products.PRICE[1] = '19.99';  
16    lt_products.PRODUCTID[2] = 'HT-1001';  
17    lt_products.PRICE[2] = '29.99';  
18    lt_products.PRODUCTID[3] = 'HT-1002';  
19    lt_products.PRICE[3] = '39.99';  
20  
21    ex_strout = LIB:FORMAT_TO_TABLE('{PRODUCTID},{PRICE}', :lt_products);  
22  
23    FOR lv_index IN 1..RECORD_COUNT(:ex_strout) DO  
24        (lv_productid, lv_price) = LIB:SPLIT(:ex_strout.result[lv_index], ',');  
25        ex_tabout.PRODUCTID[lv_index] = lv_productid;  
26        ex_tabout.PRICE[lv_index] = lv_price;  
27    END FOR;  
28  
29 END
```

Result1 × Result2 × Messages ×			
Rows (3)			
☐		AB RESULT	
☐	1	HT-1000,19.99	
☐	2	HT-1001,29.99	
☐	3	HT-1002,39.99	

Result1 × Result2 × Messages ×			
Rows (3)			
☐		AB PRODUCTID	12 PRICE
☐	1	HT-1000	19.99
☐	2	HT-1001	29.99
☐	3	HT-1002	39.99

SQLScript Language Enhancements

Libraries: SQLScript Print built-in library

The SQLSCRIPT_PRINT library functions create the server-side results from the parameters and push them into an internal buffer.

- **PRINT_LINE** – introduces a string as a PRINT result. Only supports NVARCHAR as input type. Users can freely introduce string manipulation with either concatenation operator (||), **TO_NVARCHAR()** value formatting, or newly introduced **SQLSCRIPT_STRING** built-in library.
- **PRINT_TABLE** – introduces a table variable as a PRINT result. **PRINT_TABLE()** parses a table variable into a single string and sends the string to the client.
- Currently only supported in hdbSQL interface

```
1 DO
2 BEGIN
3     USING SQLSCRIPT_PRINT as PRTLIB;
4     USING SQLSCRIPT_STRING as STRLIB;
5
6     PRTLIB:PRINT_LINE('Start of Print');
7
8     PRTLIB:PRINT_LINE('Start of Example 1');
9     lt_users = SELECT * FROM SYS.USERS LIMIT 5;
10    PRTLIB:PRINT_LINE( STRLIB:TABLE_SUMMARY(:lt_users) );
11
12    PRTLIB:PRINT_LINE('Start of Example 2');
13    lt_roles = SELECT * FROM SYS.ROLES LIMIT 5;
14    PRTLIB:PRINT_TABLE(:lt_roles);
15
16    PRTLIB:PRINT_LINE('End of Print');
17
18 END;
```

```
hdbsql HXE-> DO BEGIN USING SQLSCRIPT_PRINT as PRTLIB; USING SQLSCRIPT_STRING as STRLIB; PRTLIB:PRINT_LINE('Start of Print'); PRTL
IB:PRINT_LINE('Start of Example 1'); lt_users = SELECT * FROM SYS.USERS LIMIT 5; PRTLIB:PRINT_LINE( STRLIB:TABLE_SUMMARY(:lt_users
) ); PRTLIB:PRINT_LINE('Start of Example 2'); lt_roles = SELECT * FROM SYS.ROLES LIMIT 5; PRTLIB:PRINT_TABLE(:lt_roles); PRTLIB:PR
INT_LINE('End of Print'); END;
0 rows affected (overall time 127.882 msec; server time 17.064 msec)

Start of Print
Start of Example 1
USER_NAME,USER_ID,USERGROUP_NAME,USER_MODE,EXTERNAL_IDENTITY,CREATOR,CREATE_TIME,VALID_FROM,VALID_UNTIL,LAST_SUCCESSFUL_CONNECT,LA
ST_INVALID_CONNECT_ATTEMPT,INVALID_CONNECT_ATTEMPTS,ADMIN_GIVEN_PASSWORD,LAST_PASSWORD_CHANGE_TIME,PASSWORD_CHANGE_NEEDED,IS_PASSW
ORD_LIFETIME_CHECK_ENABLED,USER_DEACTIVATED,DEACTIVATION_TIME,IS_PASSWORD_ENABLED,IS_KERBEROS_ENABLED,IS_SAML_ENABLED,IS_JWT_ENABL
ED,IS_LDAP_ENABLED,IS_X509_ENABLED,IS_SAP_LOGIN_TICKET_ENABLED,IS_SAP_ASSERTION_TICKET_ENABLED,IS_RESTRICTED,IS_RESTRICTED_DETAILS
,IS_CLIENT_CONNECT_ENABLED,HAS_REMOTE_USERS,PASSWORD_CHANGE_TIME,AUTHORIZATION_MODE,COMMENTS,CREATE_PROVIDER_TYPE,CREATE_PROVIDER_
NAME
"CUPS_SFLIGHT",155053,"null","LOCAL","null","SYSTEM",2018-02-27 07:13:00.7810000,2018-02-27 07:13:00.7810000,null,2018-02-27 09:21
:09.2460000,null,0,"TRUE",2018-02-27 07:13:00.7810000,"FALSE","FALSE","FALSE",null,"TRUE","FALSE","FALSE","FALSE","FALSE","FALSE",
"FALSE","FALSE","FALSE","null","TRUE","FALSE",null,"LOCAL","null","null","null"
"DEMO_1",160631,"null","LOCAL","null","SYS_DI_SU",2018-03-05 14:08:20.9480000,2018-03-05 14:08:20.9480000,null,null,null,0,"TRUE"
,2018-03-05 14:08:20.9600000,"FALSE","TRUE","TRUE",2018-03-05 14:08:20.9550000,"TRUE","FALSE","FALSE","FALSE","FALSE","FALSE","FA
SE","FALSE","TRUE","ROLE PUBLIC","FALSE","FALSE",2018-03-05 14:08:20.9600000,"LOCAL","null","null","null"
"DEMO_1#DI",160345,"null","LOCAL","null","SYS_DI_SU",2018-03-05 14:08:19.2190000,2018-03-05 14:08:19.2190000,null,null,0,"TR
UE",2018-03-05 14:08:19.2390000,"FALSE","TRUE","TRUE",2018-03-05 14:08:19.2780000,"TRUE","FALSE","FALSE","FALSE","FALSE","FALSE","F
ALSE","FALSE","TRUE","ROLE PUBLIC","FALSE","FALSE",2018-03-05 14:08:19.2390000,"LOCAL","null","null","null"
"DEMO_1#OD",160633,"null","LOCAL","null","SYS_DI_SU",2018-03-05 14:08:21.0990000,2018-03-05 14:08:21.0990000,null,null,0,"TR
UE",2018-03-05 14:08:21.1100000,"FALSE","TRUE","TRUE",2018-03-05 14:08:21.1360000,"TRUE","FALSE","FALSE","FALSE","FALSE","FALSE","F
ALSE","FALSE","TRUE","null","FALSE","FALSE",2018-03-05 14:08:21.1100000,"LOCAL","null","null","null"
"DEMO_1_253BRAN4KEBNG69L0INS160W_RT",160718,"null","LOCAL","null","SYS_XS_HANA_BROKER_INTERNAL",2018-03-05 14:08:29.0200000,2018-
03-05 14:08:29.0200000,null,null,null,0,"TRUE",2018-03-05 14:08:29.0200000,"FALSE","FALSE","FALSE",null,"TRUE","FALSE","FALSE","F
ALSE","FALSE","FALSE","FALSE","FALSE","FALSE","null","TRUE","FALSE",null,"LOCAL","null","null","null"
Start of Example 2
ROLE_SCHEMA_NAME,ROLE_NAME,ROLE_ID,ROLE_MODE,GLOBAL_IDENTITY,CREATOR,CREATE_TIME,COMMENTS,CONTEXT
"null","PUBLIC",131076,"LOCAL","null","SYS",2018-02-27 07:10:13.4290000,"null","null"
"null","MONITORING",131077,"LOCAL","null","SYS",2018-02-27 07:10:13.4290000,"null","null"
"null","RESTRICTED_USER_JDBC_ACCESS",131078,"LOCAL","null","SYS",2018-02-27 07:10:13.4290000,"null","null"
"null","RESTRICTED_USER_ODBC_ACCESS",131079,"LOCAL","null","SYS",2018-02-27 07:10:13.4290000,"null","null"
"null","SAP_INTERNAL_HANA_SUPPORT",144555,"LOCAL","null","SYS",2018-02-27 07:10:38.3080000,"null","null"
End of Print
hdbsql HXE->
```

SQLScript Language Enhancements

SQLScript language features: dynamic SQL with in/out scalar parameters

- Allows to pass scalar parameters to the SQL query as well as return scalar output values
 - Use ? or \$1, \$2, \$3 to parameterize the query. Can only use in SQL statements where query parameters are supported.
 - Query will fail if the result set does not have a single row

```
1  PROCEDURE "dynamic_sql" ( out ex_value decimal(15,2) )
2      LANGUAGE SQLSCRIPT
3      SQL SECURITY INVOKER
4      AS
5      BEGIN
6
7          DECLARE lv_multiplier decimal(15,2) = 1.25;
8          DECLARE lv_tabName nvarchar(32) = '"MD.Products"';
9          DECLARE lv_colName nvarchar(32) = 'PRICE';
10
11          EXECUTE IMMEDIATE 'select max(' || :lv_colName ||
12              ' ) * ? as a from ' || :lv_tabName
13              INTO ex_value
14              USING :lv_multiplier;
15
16  END
```



SQLScript Language Enhancements

SQLScript Language Features: Dynamic SQL with Table Variables

- Enables table variables to be used in USING and INTO clause and supports CALL statement parameters with USING and INTO clauses as well.

SQL Console 9.sql x

Connected to: WORKSHOP_00-sezicxb1n7bb3q32-com.sap.op...

```
1 CALL "dynamic_sql_table_var"(  
2     IM_TABLE_NAME => 'MD.Products',  
3     IM_MULTIPLIER => .50 );
```

Result x Messages x Hold SHIFT and press ARROW keys

Rows (106)

	PRODUCTID	PRICE	NEWPRICE
1	HT-1000	956	1434
2	HT-1001	1249	1873.5
3	HT-1002	1570	2355
4	HT-1003	1650	2475

dynamic_sql_table_var.hdbpro... x

```
1 PROCEDURE "dynamic_sql_table_var" (  
2     in im_table_name nvarchar(256),  
3     in im_multiplier decimal(15,2))  
4     LANGUAGE SQLSCRIPT  
5     SQL SECURITY INVOKER  
6     AS  
7     BEGIN  
8  
9     declare lt_results table( productid nvarchar(10),  
10        price decimal(15,2),  
11        newprice decimal(15,2) );  
12  
13 EXEC 'Select PRODUCTID, PRICE, (PRICE * ?) + PRICE as newprice from ' ||  
14     im_table_name into lt_results using im_multiplier;  
15  
16 select * from :lt_results;  
17  
18 END  
19
```

SQLScript Language Enhancements

SQLScript language features: INSERT/UPDATE/DELETE operators for table variables

- Allows the developer to execute DML-like statements on intermediate table variables without involving the SQL layer
- Leads to increased performance
- Operations supported
 - INSERT
 - UPDATE
 - DELETE

```
1  READ SQL ...
2  BEGIN
3
4  declare lt_products table like :ex_products;
5  declare lv_index int = 0;
6  declare lv_del_index int array;
7  declare lv_array_index int = 0;
8
9  lt_products = select PRODUCTID, CATEGORY, PRICE from "MD.Products";
10 :ex_products.INSERT(:lt_products);
11 :ex_products.INSERT(('ProductA', 'Software', '1999.99'), 1);
12 :ex_products.INSERT(('ProductB', 'Software', '2999.99'), 2);
13 :ex_products.INSERT(('ProductC', 'Software', '3999.99'), 3);
14
15 FOR lv_index IN 1..record_count(:ex_products) DO
16   :ex_products.(PRICE).UPDATE((:ex_products.PRICE[lv_index] * 1.25), lv_index);
17 END FOR;
18
19 FOR lv_index IN 1..record_count(:ex_products) DO
20   IF :ex_products.PRICE[lv_index] <= 2500.00 THEN
21     lv_array_index = lv_array_index + 1;
22     lv_del_index[lv_array_index] = lv_index;
23   END IF;
24 END FOR;
25
26 :ex_products.DELETE(:lv_del_index);
27
28 lv_index = :ex_products.SEARCH("CATEGORY", 'PC', 1);
29 ex_pc_productid = :ex_products.PRODUCTID[lv_index];
30
31 END
```

SQLScript Language Enhancements

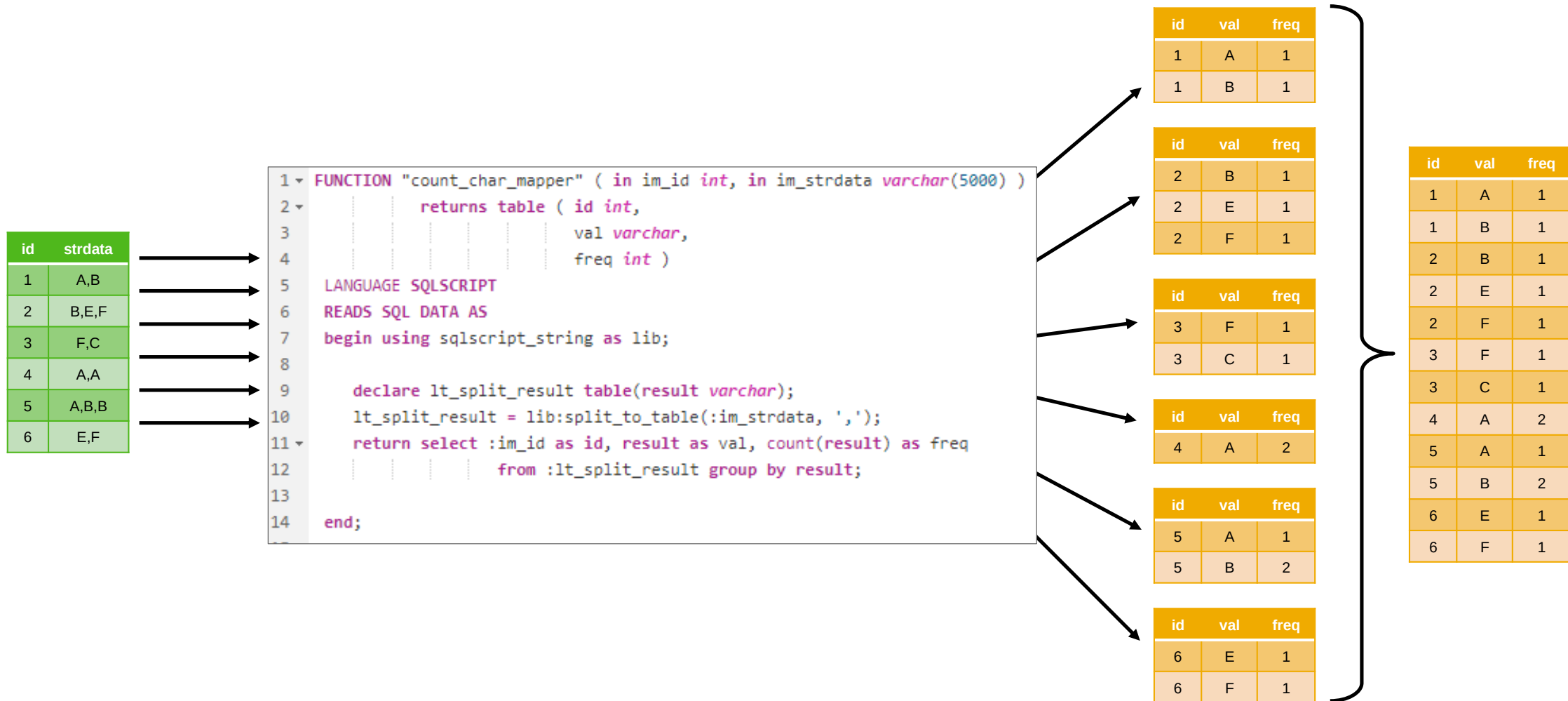
SQLScript language features: SEARCH operator for table variables

- Enables efficient search by key-value pairs in table variables.
 - The size of the column and value lists must be the same; the columns and values are matched by their position in the list
 - The <start_position> is optional, default is 1 (1st position). i.e. scan of all data
 - Any expression can be used as values and start position
 - The position of the first matching record is returned, or NULL if no record matches
 - This result can be used in conjunction with other table variable operators (DELETE, UPDATE)

```
search_by_product_id.hdbproc... ×
1  PROCEDURE "search_by_product_id" (
2      in im_productid nvarchar(10),
3      out ex_found boolean )
4      LANGUAGE SQLSCRIPT
5      SQL SECURITY INVOKER
6      READS SQL DATA AS
7  BEGIN
8
9      lt_products = select * from "MD.Products";
10
11  if ( :lt_products.SEARCH("PRODUCTID", :im_productid, 1) ) > 0 then
12      ex_found = true;
13  else
14      ex_found = false;
15  end if;
16
17  END
18
```

SQLScript Language Enhancements

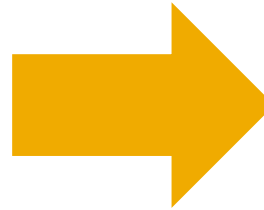
SQLScript language features: MAP_MERGE operator



SQLScript Language Enhancements

SQLScript language features: MAP_MERGE operator – Continued

```
1 do begin
2
3 declare lt_input table( id int, strdata varchar(5000));
4 declare lt_mapper_result table( id int, val varchar, freq int);
5 declare cursor c_input for select * from :lt_input;
6
7 lt_input.id[1] = '1';   lt_input.strdata[1] = 'A,B';
8 lt_input.id[2] = '2';   lt_input.strdata[2] = 'B,E,F';
9 lt_input.id[3] = '3';   lt_input.strdata[3] = 'F,C';
10 lt_input.id[4] = '4';   lt_input.strdata[4] = 'A,A';
11 lt_input.id[5] = '5';   lt_input.strdata[5] = 'A,B,B';
12 lt_input.id[6] = '6';   lt_input.strdata[6] = 'E,F';
13
14 for ls_input as c_input do
15   lt_mapper_result = select * from :lt_mapper_result
16                       union all
17                       select * from "count_char_mapper"(:ls_input.id,
18                                                         :ls_input.strdata);
19 end for;
20
21 select * from :lt_mapper_result order by id, val;
22
23 end;
```

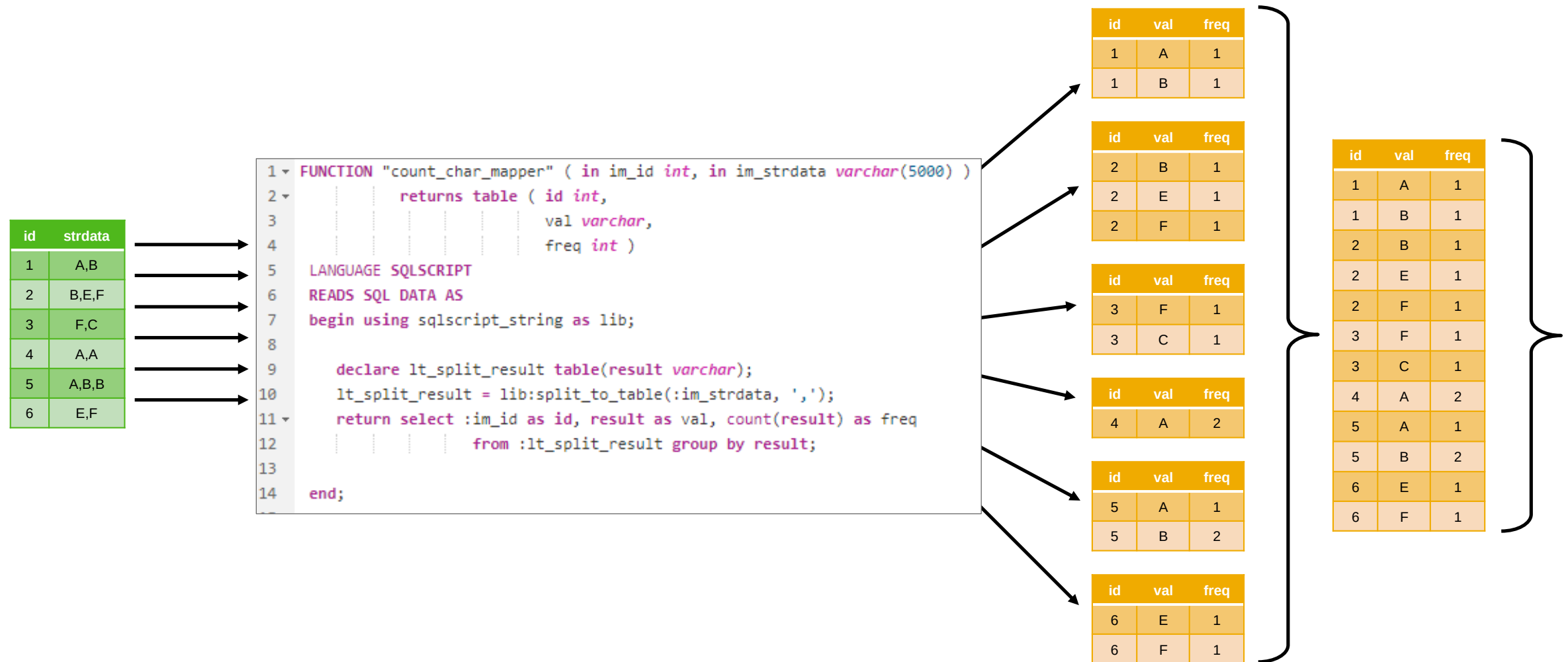


```
1 do begin
2
3 declare lt_input table( id int, strdata varchar(5000));
4 declare lt_mapper_result table( id int, val varchar, freq int);
5
6 lt_input.id[1] = '1';   lt_input.strdata[1] = 'A,B';
7 lt_input.id[2] = '2';   lt_input.strdata[2] = 'B,E,F';
8 lt_input.id[3] = '3';   lt_input.strdata[3] = 'F,C';
9 lt_input.id[4] = '4';   lt_input.strdata[4] = 'A,A';
10 lt_input.id[5] = '5';   lt_input.strdata[5] = 'A,B,B';
11 lt_input.id[6] = '6';   lt_input.strdata[6] = 'E,F';
12
13 lt_mapper_result = MAP_MERGE( :lt_input,
14                               "count_char_mapper"(:lt_input.id,
15                                                     :lt_input.strdata) );
16
17 select * from :lt_mapper_result order by id, val;
18
19 end;
```

Rows (12)		SQL		
	ID	VAL	FREQ	
1	1	A	1	
2	1	B	1	
3	2	B	1	
4	2	E	1	
5	2	F	1	
6	3	C	1	
7	3	F	1	
8	4	A	2	
9	5	A	1	
10	5	B	2	
11	6	E	1	
12	6	F	1	

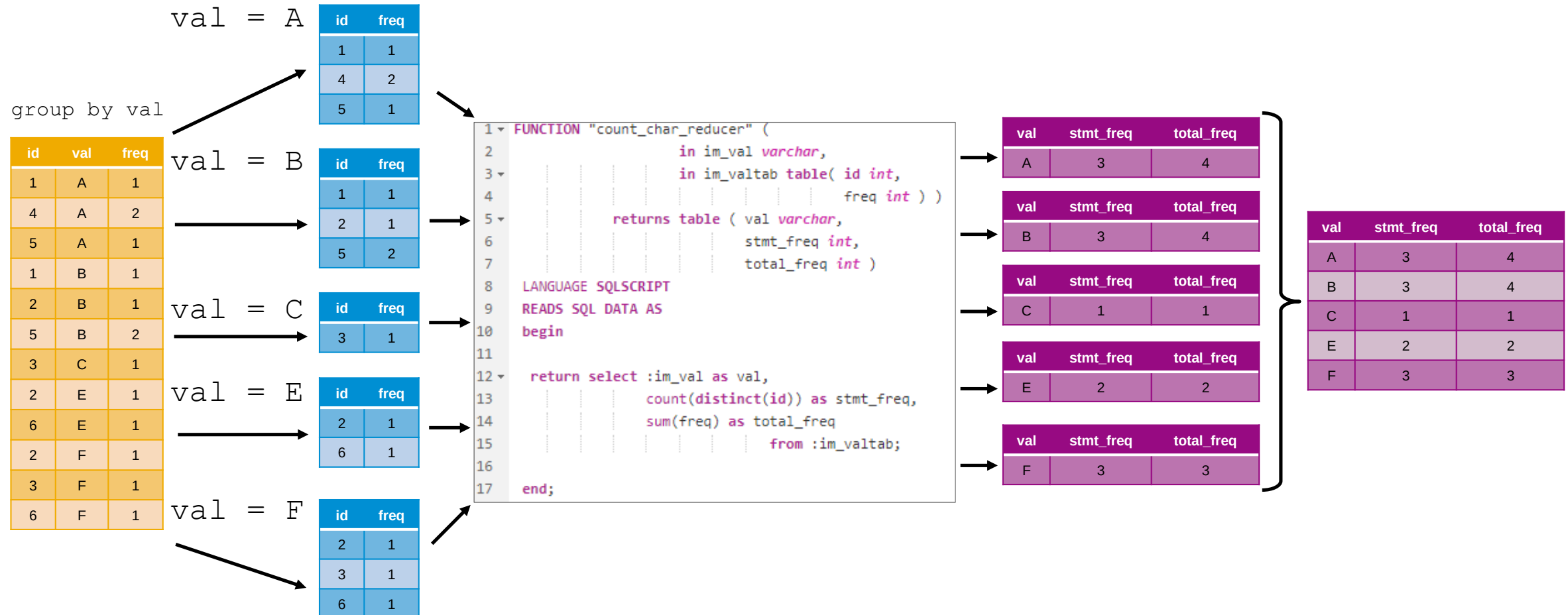
SQLScript Language Enhancements

SQLScript language features: MAP_REDUCE operator



SQLScript Language Enhancements

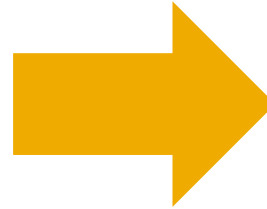
SQLScript language features: MAP_REDUCE operator – Continued



SQLScript Language Enhancements

SQLScript language features: MAP_REDUCE operator – Continued

```
1 do begin
2
3   declare lt_input table( id int, strdata varchar(5000));
4   declare lt_mapper_result table( id int, val varchar, freq int);
5   declare lt_result table(val varchar, stmt_freq int, total_freq int);
6
7   lt_input.id[1] = '1';   lt_input.strdata[1] = 'A,B';
8   lt_input.id[2] = '2';   lt_input.strdata[2] = 'B,E,F';
9   lt_input.id[3] = '3';   lt_input.strdata[3] = 'F,C';
10  lt_input.id[4] = '4';   lt_input.strdata[4] = 'A,A';
11  lt_input.id[5] = '5';   lt_input.strdata[5] = 'A,B,B';
12  lt_input.id[6] = '6';   lt_input.strdata[6] = 'E,F';
13
14  begin
15   declare cursor c_input for select * from :lt_input;
16   for ls_input as c_input do
17     lt_mapper_result = select * from :lt_mapper_result
18                       union all
19                       select * from "count_char_mapper"(:ls_input.id,
20                                                         :ls_input.strdata);
21   end for;
22 end;
23
24 begin
25   declare cursor c_mapper_result for
26     select distinct val from :lt_mapper_result;
27   for ls_mapper_result as c_mapper_result do
28     lt_valtab = select id, freq from :lt_mapper_result
29                where val = :ls_mapper_result.val;
30     lt_result = select * from :lt_result
31                union all
32                select * from "count_char_reducer"(:ls_mapper_result.val,
33                                                    :lt_valtab);
34   end for;
35 end;
36
37 select * from :lt_result order by val;
38
39 end;
```



```
1 do begin
2
3   declare lt_input table( id int, strdata varchar(5000));
4   declare lt_result table(val varchar, stmt_freq int, total_freq int);
5
6   lt_input.id[1] = '1';   lt_input.strdata[1] = 'A,B';
7   lt_input.id[2] = '2';   lt_input.strdata[2] = 'B,E,F';
8   lt_input.id[3] = '3';   lt_input.strdata[3] = 'F,C';
9   lt_input.id[4] = '4';   lt_input.strdata[4] = 'A,A';
10  lt_input.id[5] = '5';   lt_input.strdata[5] = 'A,B,B';
11  lt_input.id[6] = '6';   lt_input.strdata[6] = 'E,F';
12
13  lt_result = MAP_REDUCE( :lt_input,
14                          "count_char_mapper"(:lt_input.id, :lt_input.strdata)
15                          group by val as map_result,
16                          "count_char_reducer"(map_result.val, map_result));
17
18  select * from :lt_result order by val;
19
20 end;
```

Rows (5)				SQL	↓	↺	⌕
	VAL	STMT_FREQ	TOTAL_FREQ				
1	A	3	4				
2	B	3	4				
3	C	1	1				
4	E	2	2				
5	F	3	3				

Thank you.

Contact information:

open@sap.com

Follow all of SAP



www.sap.com/contactsap

© 2018 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platforms, directions, and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

See www.sap.com/corporate-en/legal/copyright/index.epx for additional trademark information and notices.