William Andrade Batista
July 11, 2018 3 minute read

Looping in SQLScript Like ABAP

Hi All,

Today I am here to quickly show you a useful technique for making loops (index-based cell access) in SQLScript the same way you can in ABAP. For those of you who are "HANA Beginners" and have a hard time with from/to, I would like to say that in most scenarios with small amount of data it will be fine and mainly for those you need to create registers amongst and/or other few scenarios…

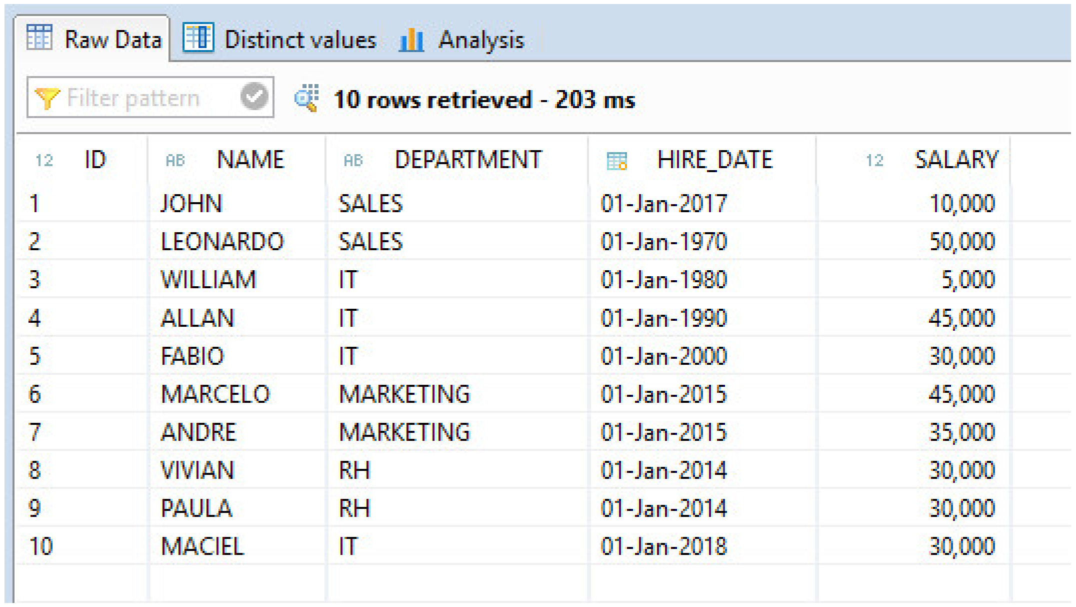
Bear in mind that this technique has bad impact on performance for large amount of data and should be used as an exception and in specific cases where a DML/DQL is not enough…

Whenever possible try to put your logic on DQL/DML statements, cause HANA has engines which processes data in optimal way and considerably faster…

Initially, it was hard for me to find an alternative in a straightforward text for the most common ways to do loops: cursors and select with offset.

Let´s get to it:

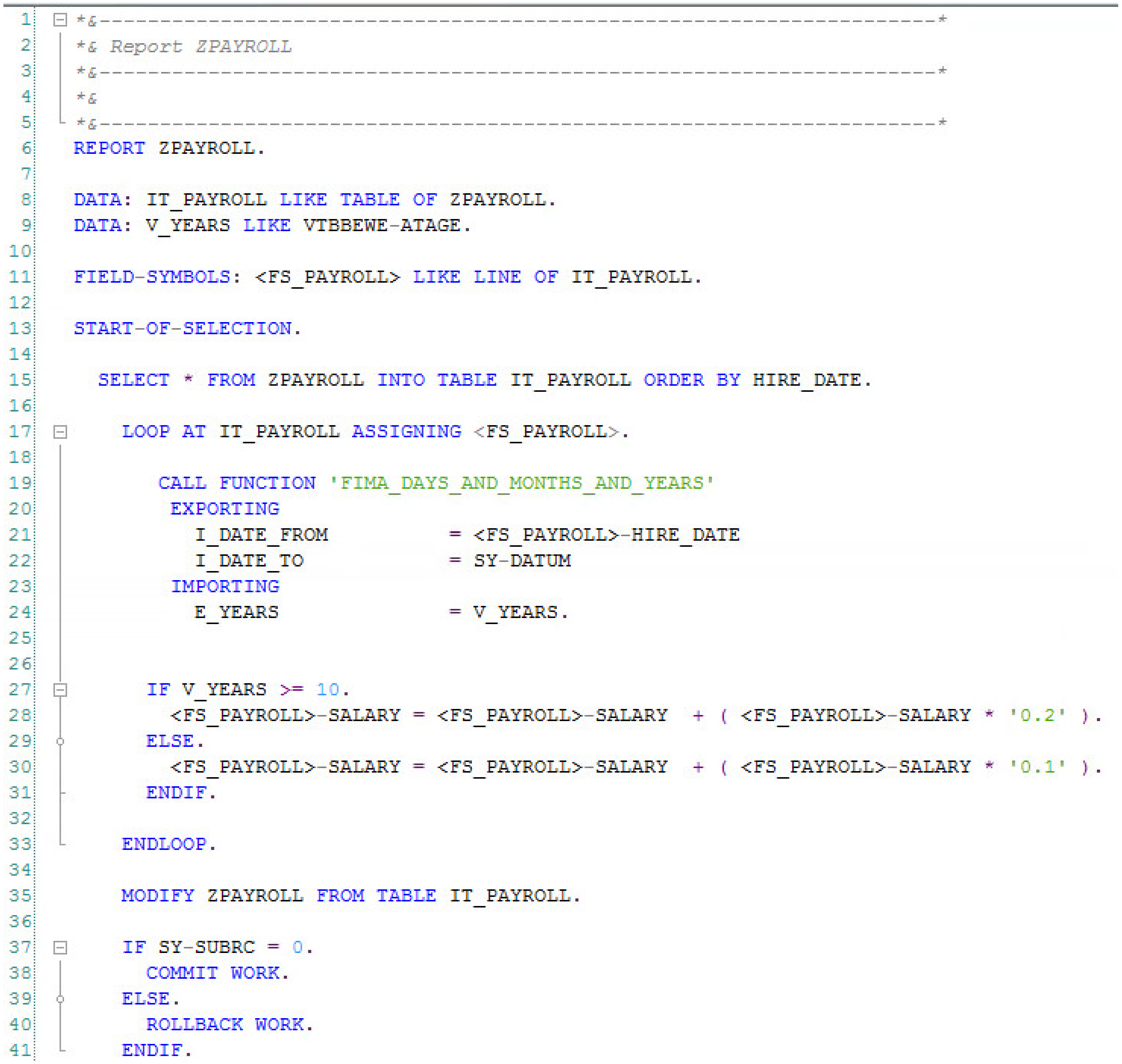
Given a scenario where we have a payroll table in which we want to give a salary raise of 20% to employees which were hired 10 or more years ago, and 10% to employees hired less than 10 years ago ... the table is shown below:



10 rows retrieved - 203 ms

ID	NAME	DEPARTMENT	HIRE_DATE	SALARY
1	JOHN	SALES	01-Jan-2017	10,000
2	LEONARDO	SALES	01-Jan-1970	50,000
3	WILLIAM	IT	01-Jan-1980	5,000
4	ALLAN	IT	01-Jan-1990	45,000
5	FABIO	IT	01-Jan-2000	30,000
6	MARCELO	MARKETING	01-Jan-2015	45,000
7	ANDRE	MARKETING	01-Jan-2015	35,000
8	VIVIAN	RH	01-Jan-2014	30,000
9	PAULA	RH	01-Jan-2014	30,000
10	MACIEL	IT	01-Jan-2018	30,000

If we were developing an ABAP, I would do this (I am assuming that the table is already created and populated):

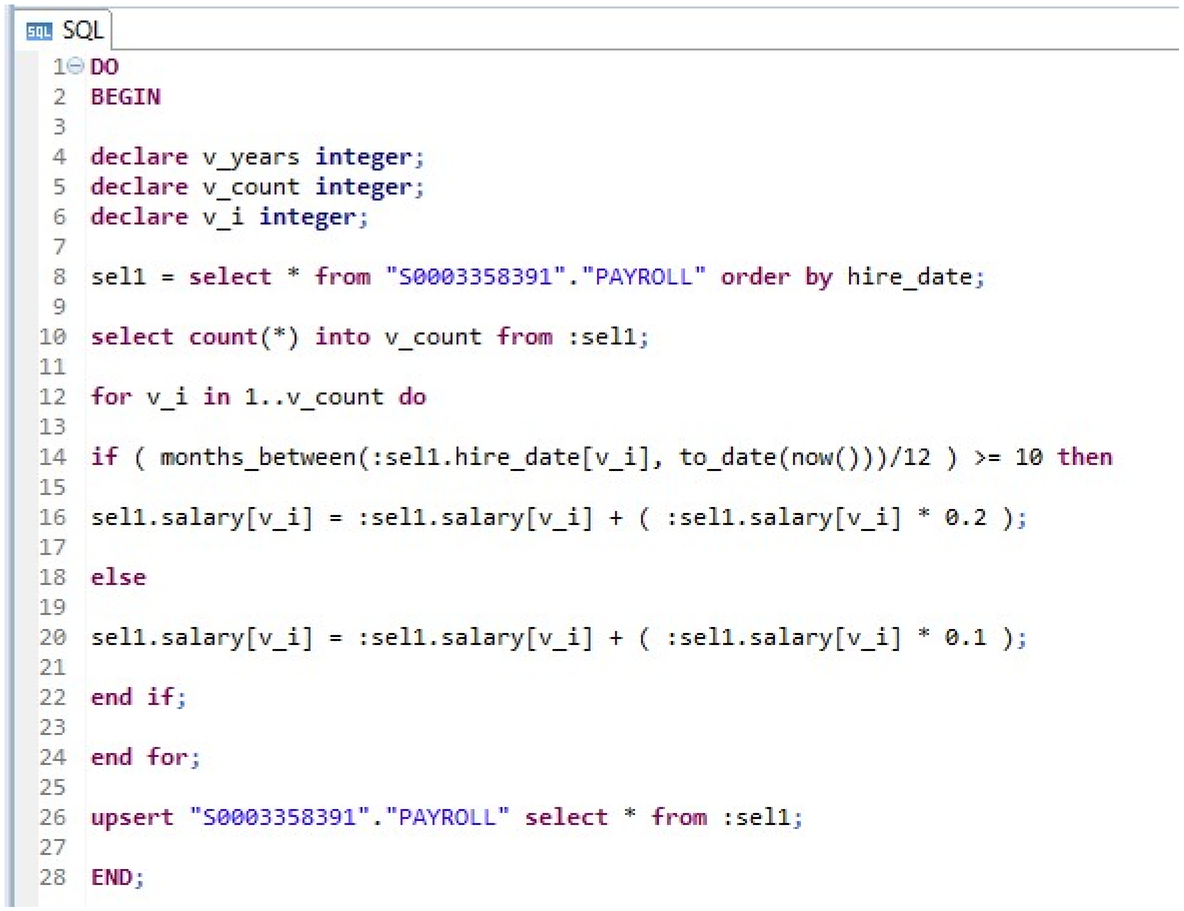


```
*&---------------------------------------------------------------------*
*& Report ZPAYROLL
*&---------------------------------------------------------------------*
*&
*&---------------------------------------------------------------------*
REPORT ZPAYROLL.

DATA: IT_PAYROLL LIKE TABLE OF ZPAYROLL.
DATA: V_YEARS LIKE VTBBEWE-ATAGE.

FIELD-SYMBOLS: <FS_PAYROLL> LIKE LINE OF IT_PAYROLL.

START-OF-SELECTION.

  SELECT * FROM ZPAYROLL INTO TABLE IT_PAYROLL ORDER BY HIRE_DATE.

    LOOP AT IT_PAYROLL ASSIGNING <FS_PAYROLL>.

       CALL FUNCTION 'FIMA_DAYS_AND_MONTHS_AND_YEARS'
        EXPORTING
          I_DATE_FROM          = <FS_PAYROLL>-HIRE_DATE
          I_DATE_TO            = SY-DATUM
        IMPORTING
          E_YEARS              = V_YEARS.


      IF V_YEARS >= 10.
        <FS_PAYROLL>-SALARY = <FS_PAYROLL>-SALARY  + ( <FS_PAYROLL>-SALARY * '0.2' ).
      ELSE.
        <FS_PAYROLL>-SALARY = <FS_PAYROLL>-SALARY  + ( <FS_PAYROLL>-SALARY * '0.1' ).
      ENDIF.

    ENDLOOP.

    MODIFY ZPAYROLL FROM TABLE IT_PAYROLL.

    IF SY-SUBRC = 0.
      COMMIT WORK.
    ELSE.
      ROLLBACK WORK.
    ENDIF.
```

How we would translate the above code to a HANA SQLScript procedure?

Using anonymous block we can code an SQLSCript into HANA's SQL Console directly...

IMPORTANT REMARK: All code below was written in version 1.0 SPS12(1.00.122.15.1516128271)



```
DO
BEGIN

declare v_years integer;
declare v_count integer;
declare v_i integer;

sel1 = select * from "S0003358391"."PAYROLL" order by hire_date;

select count(*) into v_count from :sel1;

for v_i in 1..v_count do

if ( months_between(:sel1.hire_date[v_i], to_date(now()))/12 ) >= 10 then

sel1.salary[v_i] = :sel1.salary[v_i] + ( :sel1.salary[v_i] * 0.2 );

else

sel1.salary[v_i] = :sel1.salary[v_i] + ( :sel1.salary[v_i] * 0.1 );

end if;

end for;

upsert "S0003358391"."PAYROLL" select * from :sel1;

END;
```

From HANA 2.0 SPS00 on, instead of this:

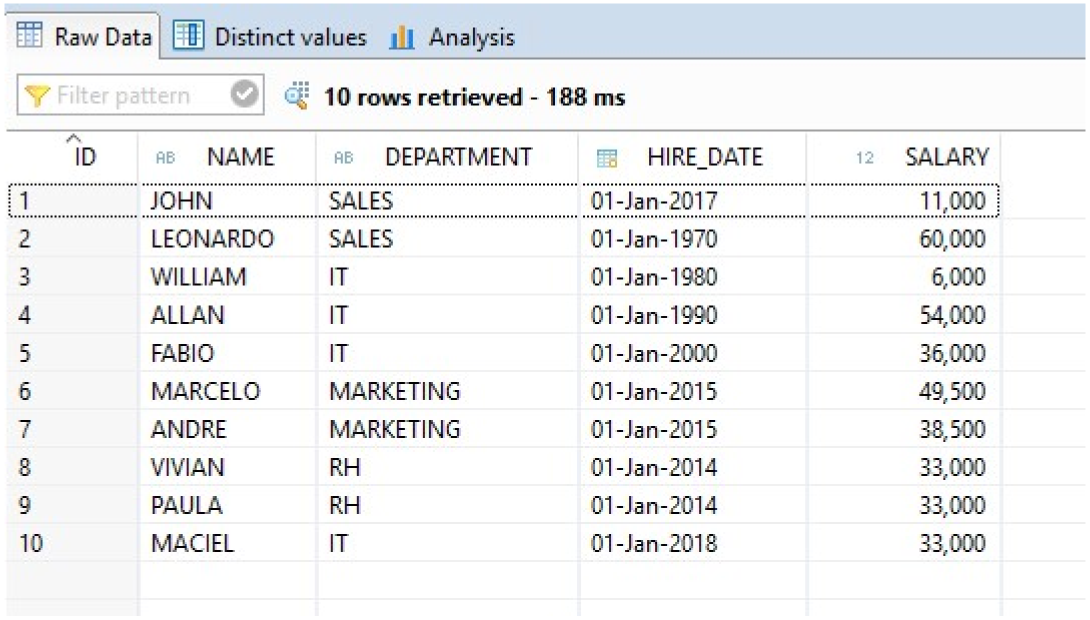
```
select count(*) into v_count from :sel1;

for v_i in 1..v_count do
```

you can use operator RECORD_COUNT, to accomplish the same “old school” task in an updated and modern style:

```
FOR i IN 1 ..RECORD_COUNT(:sel1)
```

Another important remark is that there are some places in the code where you will be unable to use index-based cell access directly, such as WHERE clauses etc. For these scenarios you can assign the table cell to a variable and then use this variable wherever it is needed.

After executing the above code, here are the results:



10 rows retrieved - 188 ms

ID	NAME	DEPARTMENT	HIRE_DATE	SALARY
1	JOHN	SALES	01-Jan-2017	11,000
2	LEONARDO	SALES	01-Jan-1970	60,000
3	WILLIAM	IT	01-Jan-1980	6,000
4	ALLAN	IT	01-Jan-1990	54,000
5	FABIO	IT	01-Jan-2000	36,000
6	MARCELO	MARKETING	01-Jan-2015	49,500
7	ANDRE	MARKETING	01-Jan-2015	38,500
8	VIVIAN	RH	01-Jan-2014	33,000
9	PAULA	RH	01-Jan-2014	33,000
10	MACIEL	IT	01-Jan-2018	33,000

Thanks and see you in next post!

P.S. If you wish to reproduce the above test in SAP HANA, here are the codes to make things easier, just in case 😉

****************************** SQL DDL/DML ******************************

CREATE TABLE “S0003358391”.”PAYROLL” (
“ID” INTEGER CS_INT,
“NAME” NVARCHAR(30),

```
“DEPARTMENT” NVARCHAR(30),
“HIRE_DATE” date,
“SALARY” decimal(15,2),
PRIMARY KEY (“ID”)) UNLOAD PRIORITY 5 AUTO MERGE

insert into “S0003358391.”PAYROLL” values(1,’JOHN’,’SALES’,’20170101′,10000);
insert into “S0003358391″.”PAYROLL” values(2,’LEONARDO’,’SALES’,’19700101′,50000);
insert into “S0003358391″.”PAYROLL” values(3,’WILLIAM’,’IT’,’19800101′,5000);
insert into “S0003358391″.”PAYROLL” values(4,’ALLAN’,’IT’,’19900101′,45000);
insert into “S0003358391″.”PAYROLL” values(5,’FABIO’,’IT’,’20000101′,30000);
insert into “S0003358391″.”PAYROLL” values(6,’MARCELO’,’MARKETING’,’20150101′,45000);
insert into “S0003358391″.”PAYROLL” values(7,’ANDRE’,’MARKETING’,’20150101′,35000);
insert into “S0003358391″.”PAYROLL” values(8,’VIVIAN’,’RH’,’20140101′,30000);
insert into “S0003358391″.”PAYROLL” values(9,’PAULA’,’RH’,’20140101′,30000);
insert into “S0003358391″.”PAYROLL” values(10,’MACIEL’,’IT’,’20180101′,30000);
```

You can replace schema “S0003358391” with the one corresponding to your user or any of your preference

```
***************************** SQLSCRIPT CODE ******************************

DO

BEGIN

declare v_years integer;

declare v_count integer;

declare v_i integer;

sel1 = select * from “S0003358391″.”PAYROLL” order by hire_date;

select count(*) into v_count from :sel1;
```

```
for v_i in 1..v_count do

if ( months_between(:sel1.hire_date[v_i], to_date(now()))/12 ) >= 10 then

sel1.salary[v_i] = :sel1.salary[v_i] + ( :sel1.salary[v_i] * 0.2 );

else

sel1.salary[v_i] = :sel1.salary[v_i] + ( :sel1.salary[v_i] * 0.1 );

end if;

end for;

upsert “S0003358391”.”PAYROLL” select * from :sel1;

END;
```

Alert Moderator

Assigned tags

SAP HANA | abap | Anonymous Block | hana development | loop |

View more...

Related Blog Posts

SQLQuery Thoughts

By **Former Member** , Apr 09, 2012

What HANA needs now is some ...

By **Lars Breddemann** , Aug 16, 2017

SQLScript Debugging from External Session

By **Rich Heilman** , Jul 02, 2013

Related Questions

AMDP_OBJECT_NOT_FOUND in HANA

By **madhavi kundarapu** , Jan 23, 2019

Looping in calculation view

By **prem kumar** , Mar 31, 2019

Concat BLOB/CLOB SAP HANA

By **Former Member** , Aug 14, 2015

3 Comments

Walter Müllner

July 12, 2018 at 1:07 pm

Why would you like to do a "select * from <table>" in an sql-script procedure? This is what exactly should be avoided not to kill HANA performance.

Why not just use two update-statements: (in the correct SQL terminology, of course):

*"update payroll set salary=salary*1.02 where months_between(hiredat, now) >=12"*

and another one with

*"update payroll set salary=salary*1.01 where months_between(hiredat, now) < 12"*

??

br, WM

Like (1)

William Andrade Batista | Post author

July 13, 2018 at 2:00 am

Hi Walter,

Thanks for your comments!

Yeah, I agree with you about performance issues using loops, that's why I put a proper remark in the beginning 😉 95% of cases this is not recommended, but in 5% there are scenarios that only using loops you can solve some problems for example: In my first experience, I had to convert a interval based table to a CALMONTH(YYYYMM) table and scenarios like this I don't know other way without use of loops. At my start, was hard to find a straightforward information as I set out to do above…..

Regarding "update", apparently, you have not understood this post objective: For abapers that do not know anything about SQLScript(my case one year ago) and any other beginner used to traditional databases. For those people is usual this approach of loops since they do not have neither tables in-memory nor in most cases SQL/Join engines… they need to pull data to memory and then process it through loops 🙂 Second, in the way I put I am sure it will help them to get familiarized faster than try change the mindset to a DQL/DML approach and last but not least I wanted to introduce 3 concepts: Anonymous blocks, Index-based cell access and loops using for..endfor and I choose to be as simple as possible with salary raise scenario… If I followed your suggestion, my post would be pointless…

So this post is not intended to teach Best Practices on use of DML/DQL for advanced users in SQLScript… sorry if I disappointed you…

All the best,

William

Like (0)

Walter Müllner

July 26, 2018 at 10:13 am

It's ok, I'm not disappointed 🙂

I obviously did not understand your objective…

br, Walter

Like (0)