Playing around a bit with SAP HANA…

Lars Breddemann, SAP Customer Solution Adoption (CSA)
June 2013, SAP HANA SPS 5

Customer

Playground setup

Yes, go and do try this at home!

All SQL commands used in this presentation can be downloaded at
http://www.sdn.sap.com/irj/scn/index?rid=/library/uuid/d0b547e1-ddf6-3010-6290-abd2184d4d4a

Setting up the playground

We will have two tables (look like DSO tables, doesn't matter though ☺

One table will get around 50 Mio records, the other 1 Mio. records

```
set schema indexperf;
drop sequence recseq;

create sequence recseq;

CREATE COLUMN TABLE PSEUDODSO
   ("REQUEST" NVARCHAR(30) DEFAULT '' NOT NULL ,
   "DATAPAKID" NVARCHAR(6) DEFAULT '000000' NOT NULL ,
   "RECORD" INTEGER  DEFAULT 0 NOT NULL ,
   "RECORDMODE" NVARCHAR(1) DEFAULT '' NOT NULL ,
     "/BIC/Z1" NVARCHAR(2) DEFAULT '0' NOT NULL ,
     "/BIC/Z10" NVARCHAR(8) DEFAULT '00000000' NOT NULL ,
     "/BIC/Z100" NVARCHAR(8) DEFAULT '00000000' NOT NULL ,
     "/BIC/Z1000" NVARCHAR(8) DEFAULT '00000000' NOT NULL ,
     "/BIC/ZDATE" NVARCHAR(8) DEFAULT '00000000' NOT NULL ,
     "/BIC/ZAMOUNT1" DECIMAL(17,3) DEFAULT 0 NOT NULL,
     "/BIC/ZAMOUNT2" DECIMAL(17,3) DEFAULT 0 NOT NULL,
     "/BIC/ZAMOUNT3" DECIMAL(17,3) DEFAULT 0 NOT NULL)
partition  BY HASH  ("/BIC/Z1","/BIC/Z1000") PARTITIONS 1;
```

Same for PSEUDODSO_S!

```
insert into pseudoDSO  (
   select top 50000000
   'xyz10', 'DP1', recseq.nextval, 'X',
    to_nvarchar(to_integer(rand() *10)),
    to_nvarchar(to_integer(mod(rand() * 100000, 10))),
    to_nvarchar(to_integer(mod(rand() * 100000, 100))),
    to_nvarchar(to_integer(mod(rand() * 100000, 1000))),
    '20121001',
    to_decimal (rand()*10, 17,3),
    to_decimal (rand()*10, 17,3),
    to_decimal (rand()*10, 17,3)
   from
       objects cross join objects
    )
;

insert into pseudodso_s (select top 1000000 * from
pseudodso);

merge delta of pseudodso   WITH PARAMETERS
('FORCED_MERGE' ='ON');
merge delta of pseudodso_s WITH PARAMETERS
('FORCED_MERGE' ='ON');
```

Start situation

WR1 (I028297) ld9506.wdf.sap.corp 00

Table Name: PSEUDODSO

Schema: INDEXPERF

Columns | Indexes | Further Properties | Runtime Information

	Name	SQL Data Type	Dim	Column Store Data Type	Key	Not Null	Default
1	REQUEST	NVARCHAR	30	STRING		X	
2	DATAPAKID	NVARCHAR	6	STRING		X	000000
3	RECORD	INTEGER		INT	X(1)	X	0
4	RECORDMODE	NVARCHAR	1	STRING		X	
5	/BIC/Z1	NVARCHAR	2	STRING		X	0
6	/BIC/Z10	NVARCHAR	8	STRING		X	00000000
7	/BIC/Z100	NVARCHAR	8	STRING		X	00000000
8	/BIC/Z1000	NVARCHAR	8	STRING		X	00000000
9	/BIC/ZDATE	NVARCHAR	8	STRING		X	00000000
10	/BIC/ZAMOUNT1	DECIMAL	17,3	FIXED		X	0
11	/BIC/ZAMOUNT2	DECIMAL	17,3	FIXED		X	0
12	/BIC/ZAMOUNT3	DECIMAL	17,3	FIXED		X	0

Start situation – big table

Table Name: PSEUDODSO | Schema: INDEXPERF | Type: Column Store

Columns | Indexes | Further Properties | Runtime Information

General

Total Memory Consumption (KB): 1.122.612
Number of Entries: 59.638.944
Size on Disk (KB): 1.348.624
Partition Specification: HASH 1 /BIC/Z1,/BIC/Z1000

Memory Consumption in Main Storage (KB): 1.122.584
Memory Consumption in Delta Storage (KB): 27
Estimated Maximum Memory Consumption (KB): 1.122.609

Details for Table

Parts | Columns

Host:Port/Partition/...	Part ID	Range	Total Size (KB)	Main Size (KB)	Delta Size (KB)	Estimated Maximum Size...	Number of Entries	Created
Id9506:30003			1.122.612	1.122.584	28	1.122.609	59.638.944	28.06.2013 08:46:17

Details for Table

Parts | Columns

Column Name	Index Type	Loaded	Part ID	Host	Port	Total Size (KB)	Main Size (KB)	Delta Size (KB)	Number of Entries	Distinct
/BIC/Z1		TRUE	0	Id9506	30003	18.888	18.886	2	59.638.944	
/BIC/Z10		TRUE	0	Id9506	30003	28.774	28.772	2	59.638.944	
/BIC/Z100		TRUE	0	Id9506	30003	50.965	50.963	2	59.638.944	
/BIC/Z1000		TRUE	0	Id9506	30003	72.808	72.806	2	59.638.944	
/BIC/ZAMOUNT1		TRUE	0	Id9506	30003	12.018	12.016	2	59.638.944	
/BIC/ZAMOUNT2		TRUE	0	Id9506	30003	102.003	102.001	2	59.638.944	
/BIC/ZAMOUNT3		TRUE	0	Id9506	30003	102.003	102.001	2	59.638.944	
/BIC/ZDATE		TRUE	0	Id9506	30003	3	1	2	59.638.944	
DATAPAKID		TRUE	0	Id9506	30003	28	26	2	59.638.944	
RECORD		TRUE	0	Id9506	30003	422.251	422.249	2	59.638.944	5
RECORDMODE		TRUE	0	Id9506	30003	3	1	2	59.638.944	
REQUEST		TRUE	0	Id9506	30003	3	1	2	59.638.944	

Start situation – small table

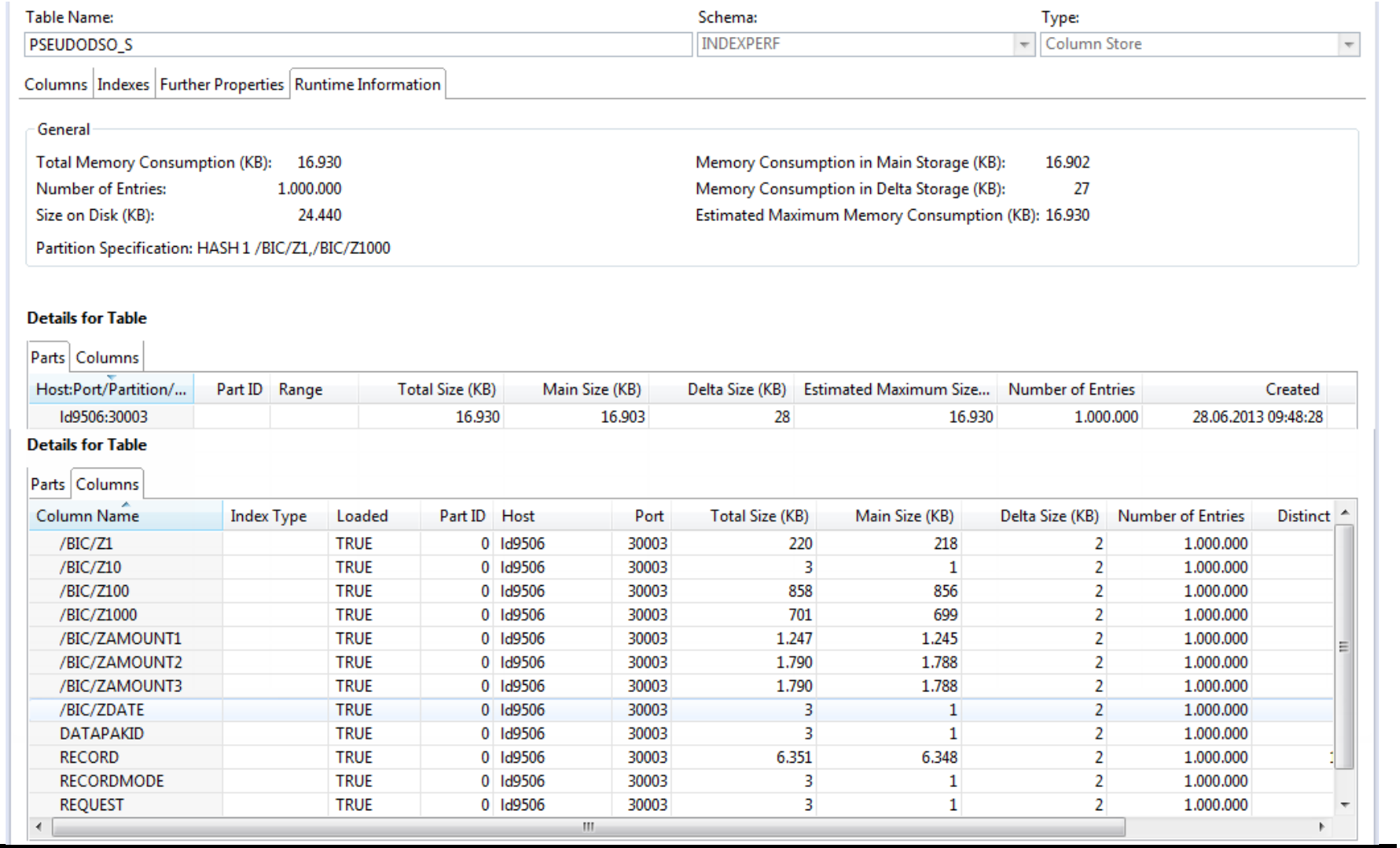


Table Name: PSEUDODSO_S | Schema: INDEXPERF | Type: Column Store

Columns | Indexes | Further Properties | Runtime Information

General

Total Memory Consumption (KB): 16.930
Number of Entries: 1.000.000
Size on Disk (KB): 24.440
Partition Specification: HASH 1 /BIC/Z1,/BIC/Z1000

Memory Consumption in Main Storage (KB): 16.902
Memory Consumption in Delta Storage (KB): 27
Estimated Maximum Memory Consumption (KB): 16.930

Details for Table

Parts | Columns

Host:Port/Partition/...	Part ID	Range	Total Size (KB)	Main Size (KB)	Delta Size (KB)	Estimated Maximum Size...	Number of Entries	Created
Id9506:30003			16.930	16.903	28	16.930	1.000.000	28.06.2013 09:48:28

Details for Table

Parts | Columns

Column Name	Index Type	Loaded	Part ID	Host	Port	Total Size (KB)	Main Size (KB)	Delta Size (KB)	Number of Entries	Distinct
/BIC/Z1		TRUE	0	Id9506	30003	220	218	2	1.000.000	
/BIC/Z10		TRUE	0	Id9506	30003	3	1	2	1.000.000	
/BIC/Z100		TRUE	0	Id9506	30003	858	856	2	1.000.000	
/BIC/Z1000		TRUE	0	Id9506	30003	701	699	2	1.000.000	
/BIC/ZAMOUNT1		TRUE	0	Id9506	30003	1.247	1.245	2	1.000.000	
/BIC/ZAMOUNT2		TRUE	0	Id9506	30003	1.790	1.788	2	1.000.000	
/BIC/ZAMOUNT3		TRUE	0	Id9506	30003	1.790	1.788	2	1.000.000	
/BIC/ZDATE		TRUE	0	Id9506	30003	3	1	2	1.000.000	
DATAPAKID		TRUE	0	Id9506	30003	3	1	2	1.000.000	
RECORD		TRUE	0	Id9506	30003	6.351	6.348	2	1.000.000	
RECORDMODE		TRUE	0	Id9506	30003	3	1	2	1.000.000	
REQUEST		TRUE	0	Id9506	30003	3	1	2	1.000.000	

Start situation – real DBA style… ☺

```
select table_name,loaded, memory_size_in_total, memory_size_in_main, memory_size_in_delta
from m_cs_tables
where schema_name ='INDEXPERF' and table_name in ('PSEUDODSO', 'PSEUDODSO_S')
order by table_name;
```

http://help.sap.com/hana/html/_m__c_s__t_a_b_l_e_s.html

Runtime data of column tables

```
TABLE_NAME  LOADED  MEMORY_SIZE_IN_TOTAL    MEMORY_SIZE_IN_MAIN MEMORY_SIZE_IN_DELTA
PSEUDODSO   FULL    1.149.554.795           1.149.526.363       28.432
PSEUDODSO_S FULL    17.336.767              17.308.335          28.432
```

```
select table_name, loaded, sum(memory_size_in_total) col_total, sum(memory_size_in_main) col_main,
sum(memory_size_in_delta) col_delta
from m_cs_columns
where schema_name ='INDEXPERF' and table_name in ('PSEUDODSO', 'PSEUDODSO_S')
group by table_name, loaded
order by table_name;
```

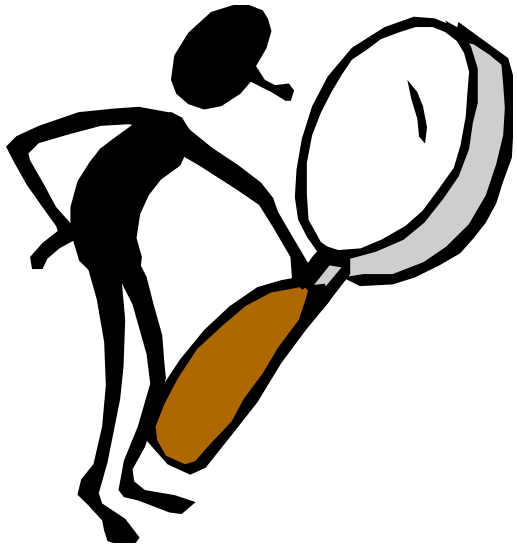
http://help.sap.com/hana/html/_m__c_s__c_o_l_u_m_n_s.html

Runtime information of columns of column tables

```
TABLE_NAME  LOADED  COL_TOTAL   COL_MAIN    COL_DELTA
PSEUDODSO   TRUE    829.183.035 829.156.891 26.144
PSEUDODSO_S TRUE    13.285.799  13.259.655  26.144
```

Some observations…

Stuff you start to notice after a while…



Observation 1 - primary key space requirements (single column)

```
alter table pseudodso add primary key ("RECORD");
successfully executed in 8.488 seconds  (server processing time: 8.428 seconds) - Rows Affected: 0

TABLE SIZE BEFORE:
TABLE_NAME  LOADED  MEMORY_SIZE_IN_TOTAL    MEMORY_SIZE_IN_MAIN MEMORY_SIZE_IN_DELTA
PSEUDODSO   FULL    1.149.554.795           1.149.526.363       28.432

AFTER:
TABLE_NAME  LOADED  MEMORY_SIZE_IN_TOTAL    MEMORY_SIZE_IN_MAIN MEMORY_SIZE_IN_DELTA
PSEUDODSO   FULL    1.343.381.379           1.343.352.947       28.432

DIFFERENCE in MAIN:  193.826.584

SUM OF COLUMN SIZES BEFORE:
TABLE_NAME  LOADED  COL_TOTAL   COL_MAIN    COL_DELTA
PSEUDODSO   TRUE    829.183.035 829.156.891 26.144

AFTER:
TABLE_NAME  LOADED  COL_TOTAL       COL_MAIN        COL_DELTA
PSEUDODSO   TRUE    1.023.009.619   1.022.983.475   26.144

DIFFERENCE in MAIN: 193.826.584

alter table pseudodso drop primary key;
```

Observation 1 - primary key space requirements (multiple columns)

```
LOAD pseudodso ALL;
alter table pseudodso add primary key ("REQUEST", "RECORD");
successfully executed in 1:09.900 minutes  (server processing time: 1:09.862 minutes) - Rows
Affected: 0

TABLE_NAME  LOADED  MEMORY_SIZE_IN_TOTAL    MEMORY_SIZE_IN_MAIN MEMORY_SIZE_IN_DELTA
PSEUDODSO   FULL    2.013.631.307           2.013.600.675       30.632

BEFORE
TABLE_NAME  LOADED  COL_TOTAL   COL_MAIN    COL_DELTA
PSEUDODSO   TRUE    829.183.035 829.156.891 26.144

AFTER
TABLE_NAME  LOADED  COL_TOTAL       COL_MAIN        COL_DELTA
PSEUDODSO   TRUE    1.045.374.439   1.045.348.295   26.144
```

Observation 1 – Comparison single vs multi column PK

M_CS_TABLES
BEFORE

TABLE_NAME	LOADED	MEMORY_SIZE_IN_TOTAL	MEMORY_SIZE_IN_MAIN	MEMORY_SIZE_IN_DELTA
PSEUDODSO	FULL	1.149.554.795	1.149.526.363	28.432

AFTER Single Col PK:

TABLE_NAME	LOADED	MEMORY_SIZE_IN_TOTAL	MEMORY_SIZE_IN_MAIN	MEMORY_SIZE_IN_DELTA
PSEUDODSO	FULL	1.343.381.379	1.343.352.947	28.432

DIFFERENCE in MAIN: 193.826.584

AFTER Multi Col PK:

TABLE_NAME	LOADED	MEMORY_SIZE_IN_TOTAL	MEMORY_SIZE_IN_MAIN	MEMORY_SIZE_IN_DELTA
PSEUDODSO	FULL	2.013.631.307	2.013.600.675	30.632

DIFFERENCE in MAIN 864.076.512

Pretty large difference…

Observation 1 – Comparison single vs multi column PK

M_CS_COLUMNS
BEFORE

TABLE_NAME	LOADED	COL_TOTAL	COL_MAIN	COL_DELTA
PSEUDODSO	TRUE	829.183.035	829.156.891	26.144

AFTER Single Col PK:

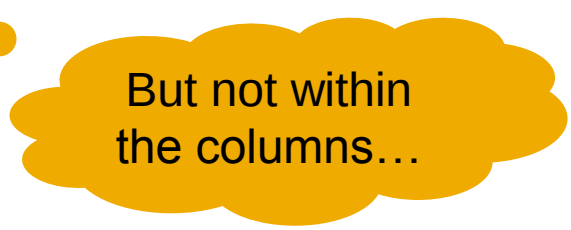
TABLE_NAME	LOADED	COL_TOTAL	COL_MAIN	COL_DELTA
PSEUDODSO	TRUE	1.023.009.619	1.022.983.475	26.144

DIFFERENCE in MAIN: 193.826.584

AFTER Multi Col PK:

TABLE_NAME	LOADED	COL_TOTAL	COL_MAIN	COL_DELTA
PSEUDODSO	TRUE	1.045.374.439	1.045.348.295	26.144

DIFFERENCE in MAIN: 216.191.404



Observation 1 – summary

- Space requirements for single and multicolumn primary keys are huge
- Multicolumn primary keys need way more memory

Observation 2 – Size information doesn't add up

```
select table_name,loaded, memory_size_in_total, memory_size_in_main, memory_size_in_delta
from m_cs_tables
where schema_name ='INDEXPERF' and table_name in ('PSEUDODSO' )
order by table_name;
```

AFTER Multi Col PK

TABLE_NAME	LOADED	MEMORY_SIZE_IN_TOTAL	MEMORY_SIZE_IN_MAIN	MEMORY_SIZE_IN_DELTA
PSEUDODSO	FULL	2.013.631.307	2.013.600.675	30.632

```
select table_name, loaded, sum(memory_size_in_total) col_total, sum(memory_size_in_main) col_main,
sum(memory_size_in_delta) col_delta
from m_cs_columns
where schema_name ='INDEXPERF' and table_name in ('PSEUDODSO' )
group by table_name, loaded
order by table_name;
```

AFTER Multi Col PK

TABLE_NAME	LOADED	COL_TOTAL	COL_MAIN	COL_DELTA
PSEUDODSO	TRUE	1.045.374.439	1.045.348.295	26.144

DIFFERENCE to table = **968.256.868**

→ **close to 50% of the whole table!!!**

Observation 3 – LOADED status of tables and columns inconsistent?

```
UNLOAD pseudodso;

M_CS_TABLES
TABLE_NAME  LOADED  MEMORY_SIZE_IN_TOTAL    MEMORY_SIZE_IN_MAIN MEMORY_SIZE_IN_DELTA
PSEUDODSO   NO      0                       0                   0

M_CS_COLUMNS
TABLE_NAME  LOADED  COL_TOTAL   COL_MAIN    COL_DELTA
PSEUDODSO   FALSE   0           0           0

→ let's load some data…

select top 10 * from pseudodso;

M_CS_TABLES
TABLE_NAME  LOADED      MEMORY_SIZE_IN_TOTAL    MEMORY_SIZE_IN_MAIN MEMORY_SIZE_IN_DELTA
PSEUDODSO   PARTIALLY   1.819.804.491           1.819.773.859       30.632

M_CS_COLUMNS
TABLE_NAME  LOADED  COL_TOTAL       COL_MAIN        COL_DELTA
PSEUDODSO   TRUE    1.045.374.439   1.045.348.295   26.144
```

Observation 4 - speed of joins, single column join

Single column join

```
select
    t1.*, t2.*
from
 pseudodso t1 inner join pseudodso_s t2
 on t1.record = t2.record;
```

Execution #1
successfully executed in 14.667 seconds (server processing time: 14.576 seconds)

Subsequent executions
successfully executed in 685 ms 772 µs (server processing time: 593 ms 189

Observation 4 - speed of joins, single column join

What about the table sizes now?

M_CS_TABLES
ORIGINAL (before PK on PSEUDODSO)

TABLE_NAME	LOADED	MEMORY_SIZE_IN_TOTAL	MEMORY_SIZE_IN_MAIN	MEMORY_SIZE_IN_DELTA
PSEUDODSO	FULL	**1.149.554.795**	1.149.526.363	28.432
PSEUDODSO_S	FULL	17.336.767	17.308.335	28.432

AFTER SINGLE COLUMN JOIN

TABLE_NAME	LOADED	MEMORY_SIZE_IN_TOTAL	MEMORY_SIZE_IN_MAIN	MEMORY_SIZE_IN_DELTA
PSEUDODSO	PARTIALLY	1.819.804.491	1.819.773.859	30.632
PSEUDODSO_S	FULL	17.336.767	17.308.335	28.432

M_CS_COLUMNS
ORIGINAL (before PK on PSEUDODSO)

TABLE_NAME	LOADED	COL_TOTAL	COL_MAIN	COL_DELTA
PSEUDODSO	TRUE	829.183.035	829.156.891	26.144
PSEUDODSO_S	TRUE	13.285.799	13.259.655	26.144

AFTER SINGLE COLUMN JOIN

TABLE_NAME	LOADED	COL_TOTAL	COL_MAIN	COL_DELTA
PSEUDODSO	TRUE	1.045.374.439	1.045.348.295	26.144
PSEUDODSO_S	TRUE	13.285.799	13.259.655	26.144

Observation 4 - speed of joins, multi column join

```
select
    t1.*, t2.*
from
 pseudodso t1 inner join pseudodso_s t2
 on  t1.record = t2.record
 and t1.request = t2.request;
```

Execution #1
successfully executed in **21.972 seconds** (server processing time: 21.903 seconds)

Subsequent executions
successfully executed in **703 ms 152 µs** (server processing time: 624 ms 686 µs)

First join execution MUCH slower than with the single column join

single column join **14.667 seconds** vs. multi column join **21.972 seconds**

subsequent join executions are nearly equally quick...

single column join **685 ms 772 µs** vs. multi column join **629 ms 956 µs**

Observation 4 - speed of joins, multi column join

What about the table sizes now?

M_CS_TABLES
ORIGINAL (before PK on PSEUDODSO)

TABLE_NAME	LOADED	MEMORY_SIZE_IN_TOTAL	MEMORY_SIZE_IN_MAIN	MEMORY_SIZE_IN_DELTA
PSEUDODSO	FULL	1.149.554.795	1.149.526.363	28.432
PSEUDODSO_S	FULL	17.336.767	17.308.335	28.432

AFTER SINGLE COLUMN JOIN

TABLE_NAME	LOADED	MEMORY_SIZE_IN_TOTAL	MEMORY_SIZE_IN_MAIN	MEMORY_SIZE_IN_DELTA
PSEUDODSO	PARTIALLY	1.819.804.491	1.819.773.859	30.632
PSEUDODSO_S	FULL	17.336.767	17.308.335	28.432

AFTER MULTI COLUMN JOIN

TABLE_NAME	LOADED	MEMORY_SIZE_IN_TOTAL	MEMORY_SIZE_IN_MAIN	MEMORY_SIZE_IN_DELTA
PSEUDODSO	PARTIALLY	2.013.631.155	2.013.600.523	30.632
PSEUDODSO_S	FULL	27.945.347	27.914.715	30.632

DIFFERENCES	
PSEUDODSO	193.826.664
PSEUDODSO_S	10.608.580
TOTAL	**204.435.244 (bytes, 194 MB!!)**

Observation 4 - speed of joins, multi column join

M_CS_COLUMNS
ORIGINAL (before PK on PSEUDODSO)

TABLE_NAME	LOADED	COL_TOTAL	COL_MAIN	COL_DELTA
PSEUDODSO	TRUE	829.183.035	829.156.891	26.144
PSEUDODSO_S	TRUE	13.285.799	13.259.655	26.144

AFTER SINGLE COLUMN JOIN

TABLE_NAME	LOADED	COL_TOTAL	COL_MAIN	COL_DELTA
PSEUDODSO	TRUE	**1.045.374.439**	1.045.348.295	26.144
PSEUDODSO_S	TRUE	13.285.799	13.259.655	26.144

AFTER MULTI COLUMN JOIN

TABLE_NAME	LOADED	COL_TOTAL	COL_MAIN	COL_DELTA
PSEUDODSO	TRUE	**1.045.374.439**	1.045.348.295	26.144
PSEUDODSO_S	TRUE	13.285.799	13.259.655	26.144

NO differences in the size of the columns, though... WHY?

Strange…

- **All this is pretty odd, isn't it?**
- **We must be missing something here!**

… and we do!

Let's check `M_CS_ALL_COLUMNS` instead!

Runtime information of all columns of column tables, **inclusive internal ones**

http://help.sap.com/hana/html/_m__c_s__a_l_l__c_o_l_u_m_n_s.html

TABLE_NAME	LOADED	COL_TOTAL	COL_MAIN	COL_DELTA
PSEUDODSO	TRUE	2.006.176.131	2.006.145.651	30.480
PSEUDODSO	FALSE	7.455.024	7.454.872	152
PSEUDODSO_S	TRUE	27.945.347	27.914.715	30.632

Now the numbers add up!

And now we see that some columns of `PSEUDODSO` are loaded into memory and others are not. So the table load status `PARTIALLY` is actually correct!

M_CS_ALL_COLUMNS – more information

```
TABLE_NAME   COLUMN_NAME          LOADED  MEMORY_SIZE_IN_TOTAL    INTERNAL_ATTRIBUTE_TYPE
PSEUDODSO    $rowid$              TRUE    312.916.584             ROWID
PSEUDODSO    $trex_udiv$          FALSE   7.455.024               TREX_UDIV
PSEUDODSO    $trexexternalkey$    TRUE    647.885.108             TREX_EXTERNAL_KEY
PSEUDODSO    /BIC/Z1              TRUE    19.341.179              NULL
[…]

PSEUDODSO_S  $REQUEST$RECORD$     TRUE    10.608.580              CONCAT_ATTRIBUTE
PSEUDODSO_S  $rowid$              TRUE    3.925.656               ROWID
PSEUDODSO_S  $trex_udiv$          TRUE    125.312                 TREX_UDIV
PSEUDODSO_S  /BIC/Z1              TRUE    225.643                 NULL
[…]
```

Ok, there are internal columns, named $...$

Can those be queried?

M_CS_ALL_COLUMNS – more information

Yep!

```
select top 10 "$rowid$","$trexexternalkey$", "$trex_udiv$" from pseudodso;
```

```
$rowid$ $trexexternalkey$   $trex_udiv$
30.099  5,xyz10;30099       1
66.466  5,xyz10;66466       2
90.445  5,xyz10;90445       3
92.970  5,xyz10;92970       4
115.225 5,xyz10;115225      5
143.412 5,xyz10;143412      6
307.968 5,xyz10;307968      7
328.871 5,xyz10;328871      8
653.581 5,xyz10;653581      9
691.033 5,xyz10;691033      10
```

Internal columns ($...$)

- **`$rowid$/ROWID`**
  - internal logical ID for a row, used only for internal stuff (e.g. during merge etcetc.)
- **$trex_udiv$/TREX_UDIV**
  - internal information used for row visibility stuff
- **$trexexternalkey$/TREX_EXTERNAL_KEY**
  - created to be used for the primary key (remember? it's `[REQUEST, RECORD]` for `PSEUDODSO`
  - the information is concatenated into a string separated by a colon **`;`**
  - if the source column is a variable size column e.g. `VARCHAR`, the length information is put in front of the actual value, separated by comma (size of REQUEST: 5 -> 5,xyz10)
  - An integer number is added as is (e.g 30099) leading to a primary key entry "`5,xyz10;30099`"

Internal column - $trexexternalkey$

Not so nice… space consumption:

3.7 seconds for the query? → due to compression size info! (SAP NOTE #1855728)

```
select  table_name, column_name, loaded, memory_size_in_total,
        COMPRESSION_RATIO_IN_PERCENTAGE "COMPR_%"
        , internal_attribute_type
from m_cs_all_columns
where schema_name ='INDEXPERF' and table_name in ('PSEUDODSO', 'PSEUDODSO_S')
order by table_name, memory_size_in_total desc;

Successfully executed in 3.721 seconds  (server processing time: 3.700 seconds)
```

TABLE_NAME	COLUMN_NAME	LOADED	MEMORY_SIZE_IN_TOTAL	COMPR_%	INTERNAL_ATTRIBUTE_TYPE
PSEUDODSO	$trexexternalkey$	TRUE	647.885.108	64,61	TREX_EXTERNAL_KEY
PSEUDODSO	RECORD	TRUE	626.211.968	262,5	NULL
PSEUDODSO	$rowid$	TRUE	312.916.584	65,58	ROWID
PSEUDODSO	/BIC/ZAMOUNT3	TRUE	104.451.192	21,89	NULL
[…]					
PSEUDODSO_S	$REQUEST$RECORD$	TRUE	10.608.580	63,07	CONCAT_ATTRIBUTE
PSEUDODSO_S	RECORD	TRUE	6.502.960	162,37	NULL
PSEUDODSO_S	$rowid$	TRUE	3.925.656	49,03	ROWID
PSEUDODSO_S	/BIC/ZAMOUNT2	TRUE	1.832.960	22,89	NULL
PSEUDODSO_S	/BIC/ZAMOUNT3	TRUE	1.832.960	22,89	NULL
[…]					

Internal column - $trexexternalkey$

- **Unique columns cannot be compressed by dictionary compression**
- **`$trexexternalkey$` by definition is unique and contains data already stored in the source columns**
- **Low compression ratio (➔ high compression) information in the views due to the way the source size of data in columns is calculated (based on data type, no. of rows and built-in defaults)**

Internal column - CONCAT_ATTRIBUTE

$REQUEST$RECORD$/CONCAT_ATTRIBUTE

on PSEUDODSO_S where we didn't create a PK before we now find a CONCAT_ATTRIBUTE that technically looks the same as the PK of PSEUDODSO

```
select top 10 "$REQUEST$RECORD$" from pseudodso_s;
```

```
$REQUEST$RECORD$
5,xyz10;796472
5,xyz10;2946784
5,xyz10;21654930
5,xyz10;30054267
5,xyz10;34425827
5,xyz10;34580861
5,xyz10;35138655
5,xyz10;39080126
5,xyz10;39838629
5,xyz10;49594515
```

Multicolumn join

- **Up to SPS 5 SAP HANA can only join single columns**
- **to perform multi column joins, a concatenated column of all join columns is created on the fly AND persisted!**
- **If there already is a concatenated column (either `TREX_EXTERNAL_KEY` or `CONCAT_ATTRIBUTE` on one table) then it will be used.**
- **If the other table doesn't have a fitting concatenated column yet, it will be created in the same order of columns (see here `[REQUEST, RECORDS]`)**
- **If there is no matching concatenated column on either join table, the column order will be sorted alphabetically.**
- **In this case that would have been `[RECORDS, REQUEST]`**

Multicolumn join – explorations …

Can subsets of the concatenated attributes be used?

Let's start and perform a join via `REQUEST,RECORDS,RECORDMODE,DATAPAKID`

```
select
    t1.*, t2.*
from
 pseudodso t1 inner join pseudodso_s t2
 on  t1.record = t2.record
 and t1.request = t2.request
 and t1.recordmode = t2.recordmode
 and t1.datapakid = t2.datapakid;

successfully executed in 1:32.012 minutes  (server processing time: 1:31.936 minutes)
```

Multicolumn join – explorations …

TABLE_NAME	COLUMN_NAME	LOADED	MEMORY_SIZE_IN_TOTAL	COMPR_%	INT_ATTRIBUTE_TYPE
PSEUDODSO	$DATAPAKID$RECORD$RECORDMODE$REQUEST$	TRUE	1.356.116.735	84,8	CONCAT_ATTRIBUTE
PSEUDODSO	$trexexternalkey$	TRUE	647.885.108	64,61	TREX_EXTERNAL_KEY
PSEUDODSO	RECORD	TRUE	626.211.968	262,5	NULL
PSEUDODSO	$rowid$	TRUE	312.916.584	65,58	ROWID
PSEUDODSO	/BIC/ZAMOUNT2	TRUE	104.451.192	21,89	NULL
[…]					
PSEUDODSO_S	$DATAPAKID$RECORD$RECORDMODE$REQUEST$	TRUE	22.483.617	83,83	CONCAT_ATTRIBUTE
PSEUDODSO_S	$REQUEST$RECORD$	TRUE	10.608.580	63,07	CONCAT_ATTRIBUTE
PSEUDODSO_S	RECORD	TRUE	6.502.960	162,37	NULL
PSEUDODSO_S	$rowid$	TRUE	3.925.656	49,03	ROWID
PSEUDODSO_S	/BIC/ZAMOUNT2	TRUE	1.832.960	22,89	NULL
[…]					

- **By now the internal columns use up more space than the user defined columns**

Multicolumn join – explorations …

```
select  table_name
      , MAP(internal_attribute_type, NULL, 'USER DEF COL', 'INTERNAL') type_of_col
      , sum(memory_size_in_total)
from m_cs_all_columns
where schema_name ='INDEXPERF' and table_name in ('PSEUDODSO', 'PSEUDODSO_S')
group by table_name, MAP(internal_attribute_type, NULL, 'USER DEF COL', 'INTERNAL')
order by table_name, sum(memory_size_in_total) desc;
```

TABLE_NAME	TYPE_OF_COL	SUM(MEMORY_SIZE_IN_TOTAL)
PSEUDODSO	INTERNAL	2.324.373.451
PSEUDODSO	USER DEF COL	1.045.374.439
PSEUDODSO_S	INTERNAL	37.143.165
PSEUDODSO_S	USER DEF COL	13.285.799

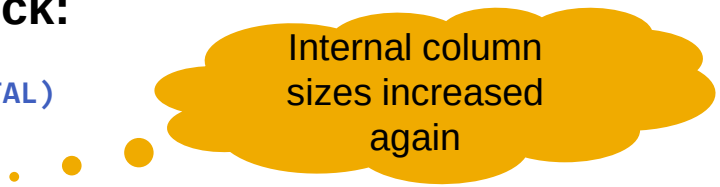
Multicolumn join – explorations …

Let's see if I can reuse the new large concatenated column to join via a smaller set of columns:

```
select
    t1.*, t2.*
from
 pseudodso t1 inner join pseudodso_s t2
 on  t1.record = t2.record
 and t1.request = t2.request
 and t1.datapakid = t2.datapakid;
successfully executed in 1:31.356 minutes  (server processing time: 1:31.281 minutes)
```

doesn't seem to be the case... let's check:

TABLE_NAME	TYPE_OF_COL	SUM(MEMORY_SIZE_IN_TOTAL)
PSEUDODSO	INTERNAL	3.441.931.674
PSEUDODSO	USER DEF COL	1.045.374.439
PSEUDODSO_S	INTERNAL	55.626.734
PSEUDODSO_S	USER DEF COL	13.285.799



Multicolumn join – explorations …

TABLE_NAME	COLUMN_NAME	LOADED	MEMORY_SIZE_IN_TOTAL	COMPR_%	INT_ATTRIBUTE_TYPE
PSEUDODSO	$DATAPAKID$RECORD$RECORDMODE$REQUEST$	TRUE	1.356.116.735	84,8	CONCAT_ATTRIBUTE
PSEUDODSO	**$DATAPAKID$RECORD$REQUEST$**	TRUE	1.117.558.223	82,14	CONCAT_ATTRIBUTE
PSEUDODSO	$trexexternalkey$	TRUE	647.885.108	64,61	TREX_EXTERNAL_KEY
PSEUDODSO	RECORD	TRUE	626.211.968	262,5	NULL
PSEUDODSO	$rowid$	TRUE	312.916.584	65,58	ROWID
[…]					
PSEUDODSO_S	$DATAPAKID$RECORD$RECORDMODE$REQUEST$	TRUE	22.483.617	83,83	CONCAT_ATTRIBUTE
PSEUDODSO_S	**$DATAPAKID$RECORD$REQUEST$**	TRUE	18.483.569	81	CONCAT_ATTRIBUTE
PSEUDODSO_S	$REQUEST$RECORD$	TRUE	10.608.580	63,07	CONCAT_ATTRIBUTE
PSEUDODSO_S	RECORD	TRUE	6.502.960	162,37	NULL
PSEUDODSO_S	$rowid$	TRUE	3.925.656	49,03	ROWID
[…]					

- **In fact for every new combination of columns a new concatenated column gets created**
- **Happens automagically, by running SELECTs (no special permission required)!**
- **The order of the columns is actually not relevant - permutations of the same columns will reuse one concatenated column**

Concatenated Columns

The concat columns are now actual part of the data model for this table:

“Export SQL” in HANA Studio or (DBA style):

```
call get_object_definition ('INDEXPERF','PSEUDODSO');
SCHEMA_NAME OBJECT_NAME OBJECT_TYPE OBJECT_OID
INDEXPERF   PSEUDODSO   TABLE       29478747

OBJECT_CREATION_STATEMENT
CREATE COLUMN TABLE "INDEXPERF"."PSEUDODSO"
("REQUEST" NVARCHAR(30) DEFAULT '' NOT NULL ,
 "DATAPAKID" NVARCHAR(6) DEFAULT '000000' NOT NULL ,
 "RECORD" INTEGER CS_INT DEFAULT 0 NOT NULL ,
 "RECORDMODE" NVARCHAR(1) DEFAULT '' NOT NULL ,
 "/BIC/Z1" NVARCHAR(2) DEFAULT '0' NOT NULL ,
 "/BIC/Z10" NVARCHAR(8) DEFAULT '00000000' NOT NULL ,
 "/BIC/Z100" NVARCHAR(8) DEFAULT '00000000' NOT NULL ,
 "/BIC/Z1000" NVARCHAR(8) DEFAULT '00000000' NOT NULL ,
 "/BIC/ZDATE" NVARCHAR(8) DEFAULT '00000000' NOT NULL ,
 "/BIC/ZAMOUNT1" DECIMAL(17,3) CS_FIXED DEFAULT 0 NOT NULL ,
 "/BIC/ZAMOUNT2" DECIMAL(17,3) CS_FIXED DEFAULT 0 NOT NULL ,
 "/BIC/ZAMOUNT3" DECIMAL(17,3) CS_FIXED DEFAULT 0 NOT NULL ,
 PRIMARY KEY ("REQUEST", "RECORD"))
WITH PARAMETERS ('PARTITION_SPEC' = 'HASH 1 /BIC/Z1,/BIC/Z1000');

ALTER TABLE "INDEXPERF"."PSEUDODSO" WITH PARAMETERS
('CONCAT_ATTRIBUTE'=('$DATAPAKID$RECORD$RECORDMODE$REQUEST$'
,'DATAPAKID','RECORD','RECORDMODE','REQUEST'));

ALTER TABLE "INDEXPERF"."PSEUDODSO" WITH PARAMETERS
('CONCAT_ATTRIBUTE'=('$DATAPAKID$RECORD$REQUEST$','DATAPAKID
','RECORD','REQUEST'));
```

Concatenated Columns - Indexes

- **Will be exported/imported**
- **Alternative ways to create concatenated columns:**
  - Define joins in modelled views and activate them. The concat attributes will then be created during activation.
  - `ALTER TABLE ... WITH PARAMETERS ('CONCAT_ATTRIBUTE'= ('$...$' , 'COL1', … ))`
  - `CREATE INDEX` command:

```
create index PDSO_I1 on PSEUDODSO_S (REQUEST, DATAPAKID);
```

TABLE_NAME	COLUMN_NAME	LOADED	MEMORY_SIZE_IN_TOTAL	COMPR_%	INT_ATTRIBUTE_TYPE
PSEUDODSO_S	$DATAPAKID$RECORD$RECORDMODE$REQUEST$	TRUE	22.483.617	83,83	CONCAT_ATTRIBUTE
PSEUDODSO_S	$DATAPAKID$RECORD$REQUEST$	TRUE	18.483.569	81	CONCAT_ATTRIBUTE
PSEUDODSO_S	$REQUEST$RECORD$	TRUE	10.608.580	63,07	CONCAT_ATTRIBUTE
PSEUDODSO_S	RECORD	TRUE	6.502.960	162,37	NULL
PSEUDODSO_S	$rowid$	TRUE	3.925.656	49,03	ROWID
PSEUDODSO_S	/BIC/ZAMOUNT2	TRUE	1.832.960	22,89	NULL
PSEUDODSO_S	/BIC/ZAMOUNT3	TRUE	1.832.960	22,89	NULL
PSEUDODSO_S	$REQUEST$DATAPAKID$	FALSE	2.200	0	CONCAT_ATTRIBUTE
[…]					

Concat col. not small – just not loaded to RAM yet…

Concatenated Columns - Indexes

What happens if I drop the index?

```
drop index PDSO_I1;
```

TABLE_NAME	COLUMN_NAME	LOADED	MEMORY_SIZE_IN_TOTAL	COMPR_%	INT_ATTRIBUTE_TYPE
PSEUDODSO_S	$DATAPAKID$RECORD$RECORDMODE$REQUEST$	TRUE	22.483.617	83,83	CONCAT_ATTRIBUTE
PSEUDODSO_S	$DATAPAKID$RECORD$REQUEST$	TRUE	18.483.569	81	CONCAT_ATTRIBUTE
PSEUDODSO_S	$REQUEST$RECORD$	TRUE	10.608.580	63,07	CONCAT_ATTRIBUTE
PSEUDODSO_S	RECORD	TRUE	6.502.960	162,37	NULL
PSEUDODSO_S	$rowid$	TRUE	3.925.656	49,03	ROWID
PSEUDODSO_S	/BIC/ZAMOUNT2	TRUE	1.832.960	22,89	NULL
PSEUDODSO_S	/BIC/ZAMOUNT3	TRUE	1.832.960	22,89	NULL
PSEUDODSO_S	/BIC/ZAMOUNT1	TRUE	1.276.864	15,95	NULL
PSEUDODSO_S	/BIC/Z100	TRUE	878.767	30,24	NULL
PSEUDODSO_S	/BIC/Z1000	TRUE	718.010	18,43	NULL
PSEUDODSO_S	/BIC/Z1	TRUE	225.643	11,25	NULL
PSEUDODSO_S	$trex_udiv$	TRUE	125.312	3,13	TREX_UDIV
PSEUDODSO_S	/BIC/ZDATE	TRUE	3.539	0,04	NULL
PSEUDODSO_S	REQUEST	TRUE	3.539	0,06	NULL
PSEUDODSO_S	/BIC/Z10	TRUE	3.519	0,18	NULL
PSEUDODSO_S	RECORDMODE	TRUE	3.519	0,18	NULL
PSEUDODSO_S	DATAPAKID	TRUE	3.519	0,09	NULL

Concatenated attribute is gone as well!

Concatenated Columns - Indexes

- **This means we got an easy way to get rid of those concats again, but only if they had been created for SQL indexes before**
- **Any other way to get rid of the concatenated columns?**
- **YEP: undocumented!**
- ```
  ALTER TABLE <table_name>
  WITH PARAMETERS ('DELETE_CONCAT_ATTRIBUTE' = '<concat attribute name>');
  ```
  ```

Concatenated Columns – drop concat attributes

```
ALTER TABLE PSEUDODSO_S
WITH PARAMETERS ('DELETE_CONCAT_ATTRIBUTE' = '$DATAPAKID$RECORD$RECORDMODE$REQUEST$' );
```

TABLE_NAME	COLUMN_NAME	LOADED	MEMORY_SIZE_IN_TOTAL	COMPR_%	INT_ATTRIBUTE_TYPE
PSEUDODSO_S	$DATAPAKID$RECORD$REQUEST$	TRUE	18.483.569	81	CONCAT_ATTRIBUTE
PSEUDODSO_S	$REQUEST$RECORD$	TRUE	10.608.580	63,07	CONCAT_ATTRIBUTE
PSEUDODSO_S	RECORD	TRUE	6.502.960	162,37	NULL
PSEUDODSO_S	$rowid$	TRUE	3.925.656	49,03	ROWID
PSEUDODSO_S	/BIC/ZAMOUNT2	TRUE	1.832.960	22,89	NULL
PSEUDODSO_S	/BIC/ZAMOUNT3	TRUE	1.832.960	22,89	NULL
PSEUDODSO_S	/BIC/ZAMOUNT1	TRUE	1.276.864	15,95	NULL
PSEUDODSO_S	/BIC/Z100	TRUE	878.767	30,24	NULL
PSEUDODSO_S	/BIC/Z1000	TRUE	718.010	18,43	NULL
PSEUDODSO_S	/BIC/Z1	TRUE	225.643	11,25	NULL
PSEUDODSO_S	$trex_udiv$	TRUE	125.312	3,13	TREX_UDIV
PSEUDODSO_S	REQUEST	TRUE	3.539	0,06	NULL
PSEUDODSO_S	/BIC/ZDATE	TRUE	3.539	0,04	NULL
PSEUDODSO_S	DATAPAKID	TRUE	3.519	0,09	NULL
PSEUDODSO_S	RECORDMODE	TRUE	3.519	0,18	NULL
PSEUDODSO_S	/BIC/Z10	TRUE	3.519	0,18	NULL

The concat attribute will now be recreated the next time the multi column join is executed.

Concatenated Columns – drop concat attributes

What if I drop the concat attribute that belongs to an index?

```
create index PDSO_I1 on PSEUDODSO_S (REQUEST, DATAPAKID);
Statement 'create index PDSO_I1 on PSEUDODSO_S (REQUEST, DATAPAKID)'
successfully executed in 591 ms 555 µs
```

That was quick!

TABLE_NAME	COLUMN_NAME	LOADED	MEMORY_SIZE_IN_TOTAL	COMPR_%	INTE_ATTRIBUTE_TYPE
PSEUDODSO_S	$DATAPAKID$RECORD$REQUEST$	TRUE	18.483.569	81	CONCAT_ATTRIBUTE
PSEUDODSO_S	$REQUEST$RECORD$	TRUE	10.608.580	63,07	CONCAT_ATTRIBUTE
PSEUDODSO_S	RECORD	TRUE	6.502.960	162,37	NULL
PSEUDODSO_S	$rowid$	TRUE	3.925.656	49,03	ROWID
PSEUDODSO_S	/BIC/ZAMOUNT3	TRUE	1.832.960	22,89	NULL
PSEUDODSO_S	/BIC/ZAMOUNT2	TRUE	1.832.960	22,89	NULL
PSEUDODSO_S	/BIC/ZAMOUNT1	TRUE	1.276.864	15,95	NULL
PSEUDODSO_S	/BIC/Z100	TRUE	878.767	30,24	NULL
PSEUDODSO_S	/BIC/Z1000	TRUE	718.010	18,43	NULL
PSEUDODSO_S	/BIC/Z1	TRUE	225.643	11,25	NULL
PSEUDODSO_S	$trex_udiv$	TRUE	125.312	3,13	TREX_UDIV
PSEUDODSO_S	/BIC/ZDATE	TRUE	3.539	0,04	NULL
PSEUDODSO_S	REQUEST	TRUE	3.539	0,06	NULL
PSEUDODSO_S	/BIC/Z10	TRUE	3.519	0,18	NULL
PSEUDODSO_S	RECORDMODE	TRUE	3.519	0,18	NULL
PSEUDODSO_S	DATAPAKID	TRUE	3.519	0,09	NULL
PSEUDODSO_S	$REQUEST$DATAPAKID$	FALSE	2.200	0	CONCAT_ATTRIBUTE

Concat column currently NOT in memory!

Concatenated Columns – drop concat attributes

Let's load it to memory:

```
LOAD PSEUDODSO_S ("$REQUEST$DATAPAKID$");
```

```
Could not execute 'LOAD PSEUDODSO_S ("$REQUEST$DATAPAKID$")' in 20 ms 777 µs .
SAP DBTech JDBC: [2052] (at 5): not supported dml type for column table: cannot update internal
field: line 1 col 6 (at pos 5)
```

Either `LOAD PSEUDODSO_S ALL` or

```
select "$REQUEST$DATAPAKID$" from pseudodso_s;
```

Now let's drop it

```
ALTER TABLE PSEUDODSO_S WITH PARAMETERS ('DELETE_CONCAT_ATTRIBUTE' = '$REQUEST$DATAPAKID$' );
```

Concatenated Columns – drop concat attributes

TABLE_NAME	COLUMN_NAME	LOADED	MEMORY_SIZE_IN_TOTAL	COMPR_%	INT_ATTRIBUTE_TYPE
PSEUDODSO_S	$DATAPAKID$RECORD$REQUEST$	TRUE	18.483.569	81	CONCAT_ATTRIBUTE
PSEUDODSO_S	$REQUEST$RECORD$	TRUE	10.608.580	63,07	CONCAT_ATTRIBUTE
PSEUDODSO_S	RECORD	TRUE	6.502.960	162,37	NULL
PSEUDODSO_S	$rowid$	TRUE	3.925.656	49,03	ROWID
PSEUDODSO_S	/BIC/ZAMOUNT2	TRUE	1.832.960	22,89	NULL
PSEUDODSO_S	/BIC/ZAMOUNT3	TRUE	1.832.960	22,89	NULL
PSEUDODSO_S	/BIC/ZAMOUNT1	TRUE	1.276.864	15,95	NULL
PSEUDODSO_S	/BIC/Z100	TRUE	878.767	30,24	NULL
PSEUDODSO_S	/BIC/Z1000	TRUE	718.010	18,43	NULL
PSEUDODSO_S	/BIC/Z1	TRUE	225.643	11,25	NULL
PSEUDODSO_S	$trex_udiv$	TRUE	125.312	3,13	TREX_UDIV
PSEUDODSO_S	REQUEST	TRUE	3.539	0,06	NULL
PSEUDODSO_S	/BIC/ZDATE	TRUE	3.539	0,04	NULL
PSEUDODSO_S	DATAPAKID	TRUE	3.519	0,09	NULL
PSEUDODSO_S	RECORDMODE	TRUE	3.519	0,18	NULL
PSEUDODSO_S	/BIC/Z10	TRUE	3.519	0,18	NULL

Works, the concat column is gone!

Concatenated Columns – drop concat attributes

What about the index?

```
select table_name, index_name from indexes
 where schema_name ='INDEXPERF' and table_name ='PSEUDODSO_S';
```

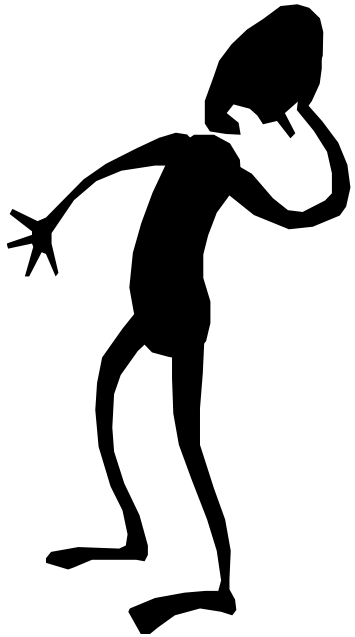
```
TABLE_NAME  INDEX_NAME
PSEUDODSO_S PDSO_I1
```

Still there…let's try to drop it:

```
drop index PDSO_I1;
```

```
Could not execute 'drop index PDSO_I1' in 23 ms 352 µs .
SAP DBTech JDBC: [2048]: column store error:
delete concat attribute error:  [2957] undefined index attribute
```

Too bad, now your DB catalog is inconsistent!



Concatenated Columns – drop concat attributes

Only workaround I now: recreate and ...

Undocumented and not supported for manual use

```
ALTER TABLE "INDEXPERF"."PSEUDODSO_S" WITH PARAMETERS
    ('CONCAT_ATTRIBUTE'=('$REQUEST$DATAPAKID$','REQUEST','DATAPAKID'));
```

… drop the index then:

```
drop index PDSO_I1;

Statement 'drop index PDSO_I1'
successfully executed in 27 ms 629 µs  (server processing time: 8 ms 90 µs) - Rows Affected: 0
```

Summary

- **Concat attributes tend to take a lot of space as they are very often created over high cardinality columns (many different values)**
- **They are created and handled in the background w/o user interaction or consent**
- **They are hidden from usual DBA tools and size reporting**
- **They can have a tremendous impact on the memory utilization and performance**
- **For multi column joins there are up to SPS 5 no alternatives, except remodelling of the table structures**
- **As of SPS 6 multi column join capabilities w/o concat attributes will (likely) be available (currently in testing).**

Thank you

We value your opinion. Let us know how you like this CSA presentation and how we can make adopting SAP innovations a more beautiful experience. Please send questions and comments to csa_feedback@sap.com.

Contact information:

Lars Breddemann

RIG Expert, Customer Solution Adoption (CSA)
mailto:lars.breddemann@sap.com