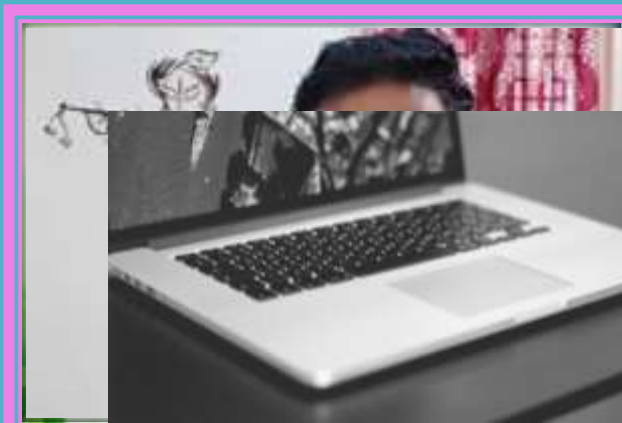
ABAP ON HANA

Introduction

Ram Niwas

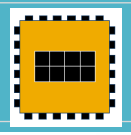


SAP TECHNOMANIAC

HANA

Magic Words

01 IN-MEMORY DATABASE

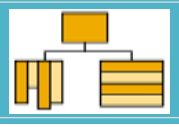


02 COLUMN STORE

03 DATA COMPRESSION

04 OLTP & OLAP

05 INSET ONLY APPROACH



06 TABLE PARTITIONING

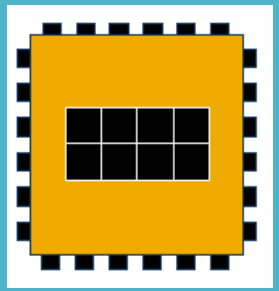
07 CODE PUSH DOWN

Hardware Innovation

- In-Memory Database

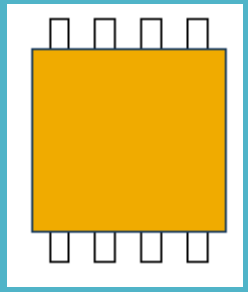
MUTICORE ARCHETECTURE

8 CPUs * (8-16) cores per blade



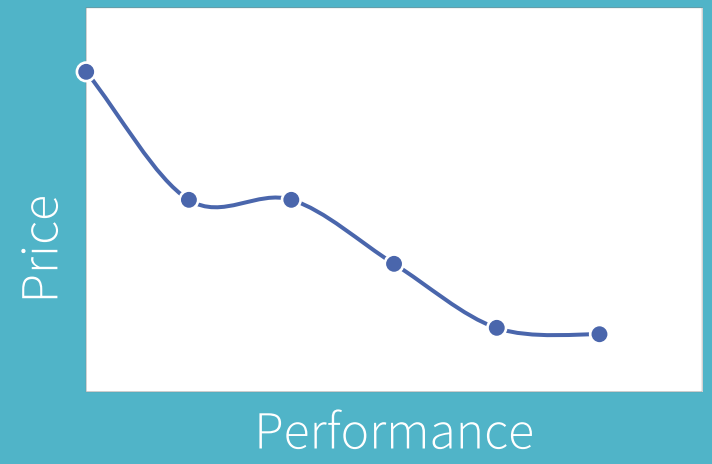
ADDRESS SPACE

64 bit address space (4 TB in current server boards



PERFORMANCE VS PRICE

In-Memory Database



MEMORY DEVICE	ACCESS TIME (NS)	ACCESS TIME
Hard Disk	6,000,000 – 8,000,000	6 – 8 milliseconds
Flash Memory, SSD	200,000	200 microseconds
Main Memory (DRAM)	60 - 100	60 – 100 nanoseconds
L3 Cache (CPU)	16 (about 40 cycles)	16 nanoseconds
L2 Cache (CPU)	4 (about 10 cycles)	4 nanoseconds
L1 Cache (CPU)	1.5 (abut 3 to 4 cycles)	1.5 nanoseconds
CPU Register	< 1 (1 cycle)	< 1 nanosecond

	Hard Disk	RAM
PRICE	0.05 USD/GB	7 USD/GB
PRICE 1 TB	100 USD	7000 USD

IN-MEMORY STORAGE TYPE

02

ROW STORE

ROW vs COLUMN STORE

Low compression rate

The Content of the row placed next to each other in main memory

We need secondary indexes for fast data reading

Order	Customer	Currency	Amount
456	JaTeCo	EUR	1300
457	SAP	EUR	750
458	Sorali	EUR	115
459	SAP	EUR	30.000

```
SELECT * ….
 WHERE ORDER = 457
```

Good performance

```
SELECT SUM(Amount)…
```

Low performance

456 JaTeCO EUR 1300 457 SAP EUR 750 458 Sorali EUR

COLUMN STORE

ROW vs COLUMN STORE

High compression rate

The Content of the column placed next to each other in main memory

Do not need secondary insexes in most of the cases

Order	Customer	Currency	Amount
456	JaTeCo	EUR	1300
457	SAP	EUR	750
458	Sorali	EUR	115
459	SAP	EUR	30.000

```
SELECT * ….
 WHERE ORDER = 457
```

Low performance

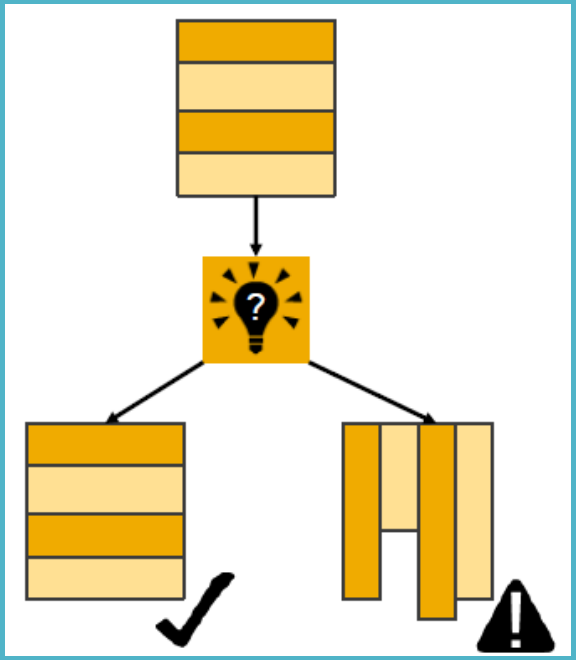
```
SELECT SUM(Amount)…
```

Good performance

456 456 457 458 458 JaTego SAP Sorali SAP EUR EUR

COLUMN STORE

ROW vs COLUMN STORE



Columnar storage is best for tables

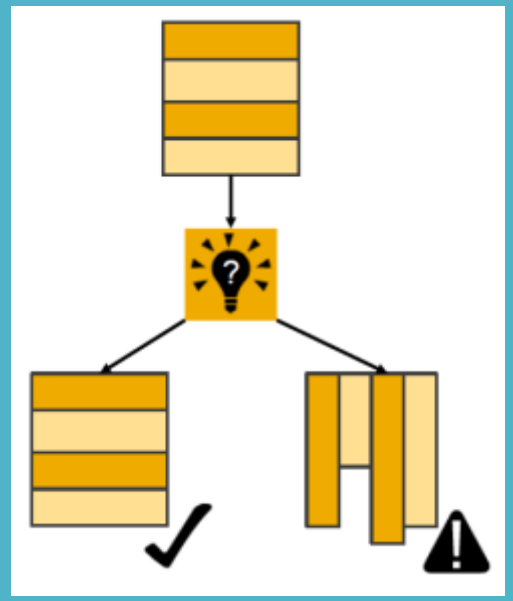
- That are subject to column operations on a large number of rows
- That have a large number of columns, more unused
- That are subject to aggregations and intensive search operations

ROW STORE

ROW vs COLUMN STORE

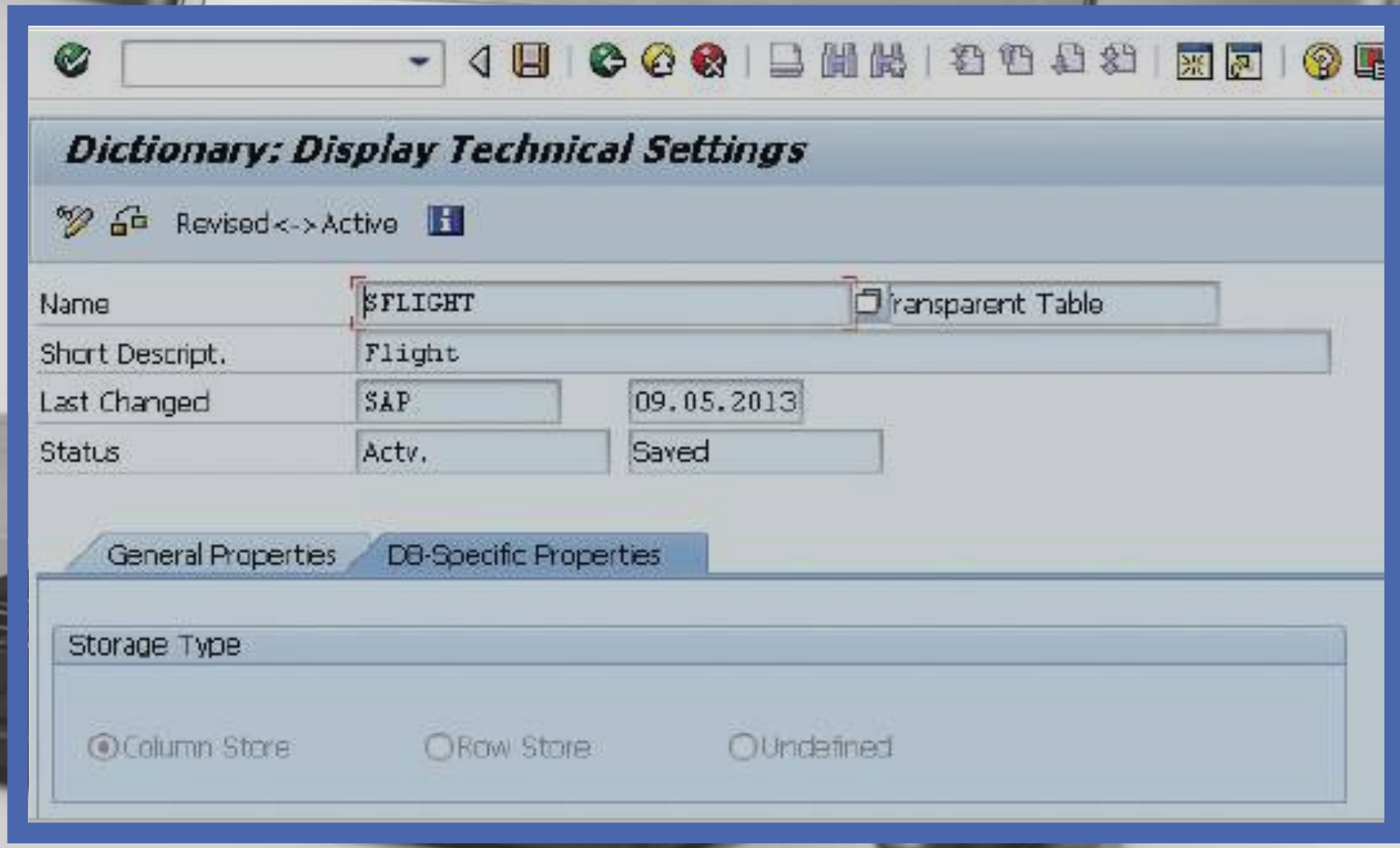
Row storage is more suitable for tables

- That contain mainly distinct values leading to low compression rate
- In which most/all columns are relevant
- That are not subject to aggregation or search operations on non-indexed columns
- That are fully buffered
- That have a small number of records



SAP TECHNICAL SETTINGS

ROW vs COLUMN STORE



THANK YOU

CONTACT

Ram Niwas

LinkedIn :- https://www.linkedin.com/in/ram-niwas-04/

FB group :-
https://www.facebook.com/groups/586730659057346/

SUBSRIBE

LIKE

COMMENT

SHARE

THANKS

DATA COMPRESSION

VECTOR REPRESENTATION

DATA COMPRESSION

RECORD	LAST NAME	LOCATION	GENDER
...	...	...	...
3	BROWN	CHICAGO	M
4	BROWN	SAN FRANCISCO	F
5	DOE	DALLAS	M
6	DOE	SAN FRANCISCO	F
7	SMITH	DALLAS	M
...	...	...	...

DICTIONARY VECTOR

GENDER	POSITION
F	1
M	2

LAST NAME	POSITION
...	...
BROWN	7
DOE	8
SMITH	9
...	...

LAST NAME	POSITION
...	...
CHICAGO	5
DALLS	6
SAN FRANCISCO	7
...	...

ATTRIBUTE VECTOR

7	7	8	8	9	5	7	6	7	6	2	1	2	1	2
3	4	5	6	7	3	4	5	6	7	3	4	5	6	7
LAST NAME ATTRIBUTE VECTOR					LOCATION ATTRIBUTE VECTOR					GENDER ATTRIBUTE VECTOR				

ADVANTAGES

DATA COMPRESSION

- Lower storage requirement
- Accelerate data transfer from the main memory to CPU
- Faster processing of integer values

OLTP & OLAP

04

OLTP & OLAP Difference

OLTP (Online Transactional Processing)	OLAP (Online Analytic Processing)
Current Data (6-18 Months)	Historic data (2-7 Years)
Day to day operation	Decision making planning problem solving
Less data compared to OLAP hence fast processing	Huge data hence slow processing
Read /Update/Add/Delete/Modify	Read
Simple Queries	Very complex Queries
Row store	Column store

OLTP & OLAP

Disadvantage of Current Process

- Replication of same data
- Extraction transporting and loading process (ETL) is used for transforming the different format data (date and currency) from different geography for reporting purpose
- BW is not real time system

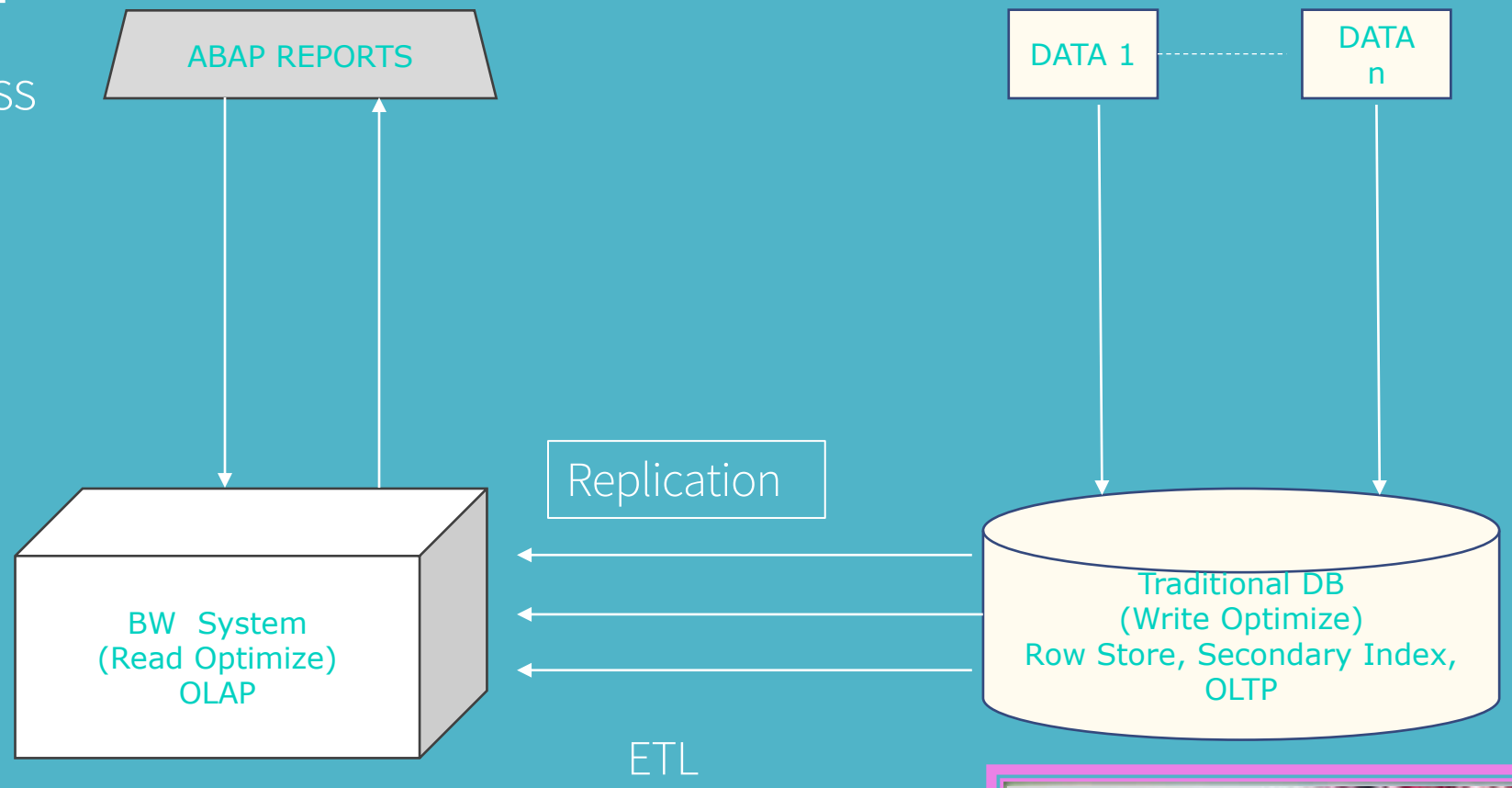


TABLE PARTITIONING

ROUND ROBIN PARTITION

Table Partitioning

Non-Partitioned Table

Column: Name

AV	DV
1	Alex
2	Anna
3	Christopher
4	Dan
5	David
6	Eric
7	Erica
8	Henry
9	Martina
10	Thomas
11	Tina
12	Yvonne

Column: Gender

AV	DV
1	f
2	m
1	
1	
1	
1	
2	
1	
2	
1	
2	
2	

Partitioned Table

P1 Column: Name

AV	DV
1	Alex
2	Christopher
3	David
4	Erica
5	Martina
6	Tina

P1 Column: Gender

AV	DV
1	f
1	m
1	
2	
2	
2	

P2 Column: Name

AV	DV
1	Anna
2	Dan
3	Eric
4	Henry
5	Thomas
6	Yvonne

P2 Column: Gender

AV	DV
2	f
1	m
1	
1	
1	
2	

ADVANTAGES

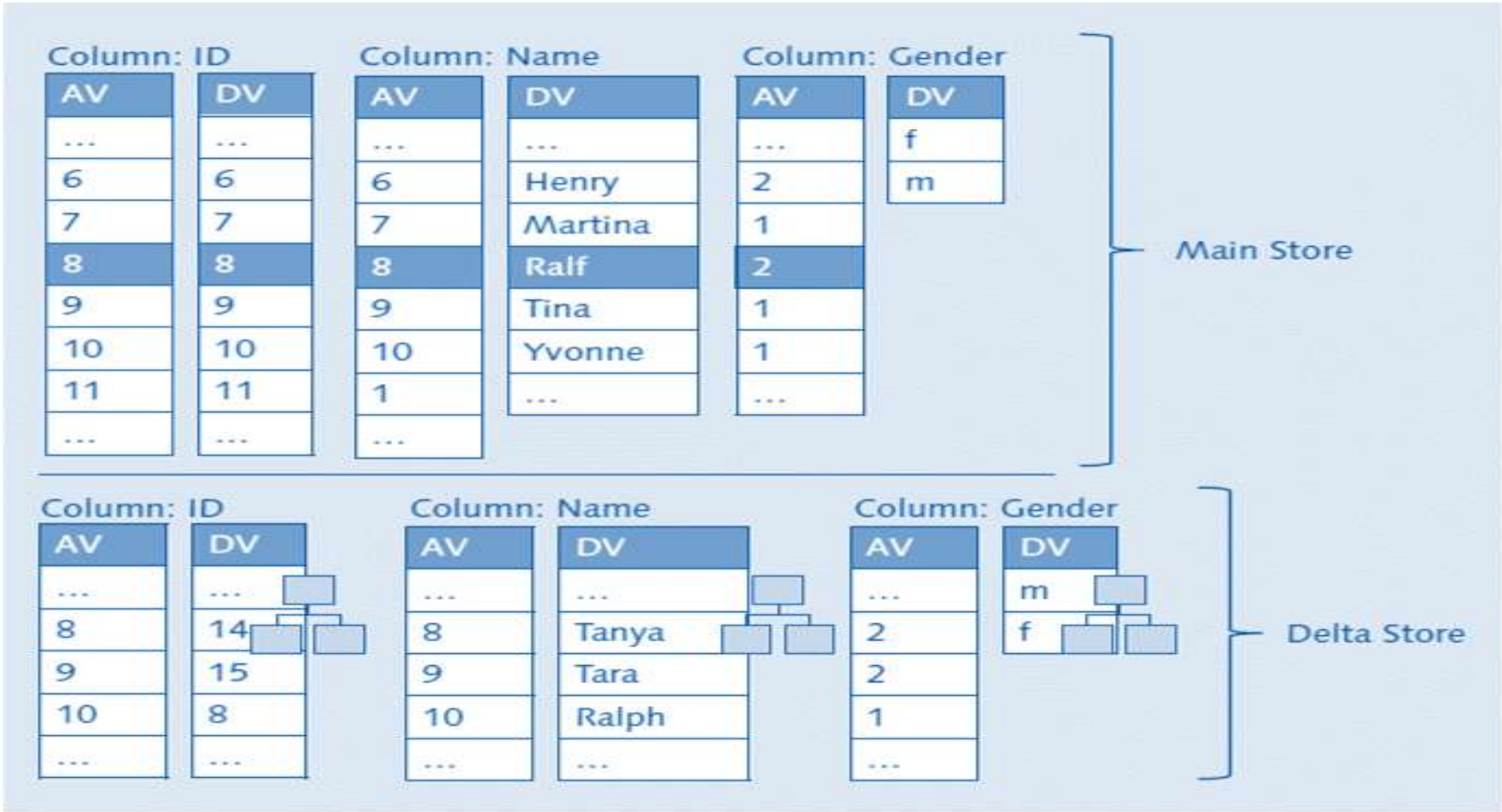
TABLE PARTITIONING

- Load distribution
- Partition pruning
- Deletion of data
- Parallelization

INSET ONLY APPROACH

MAIN & DELTA STORE

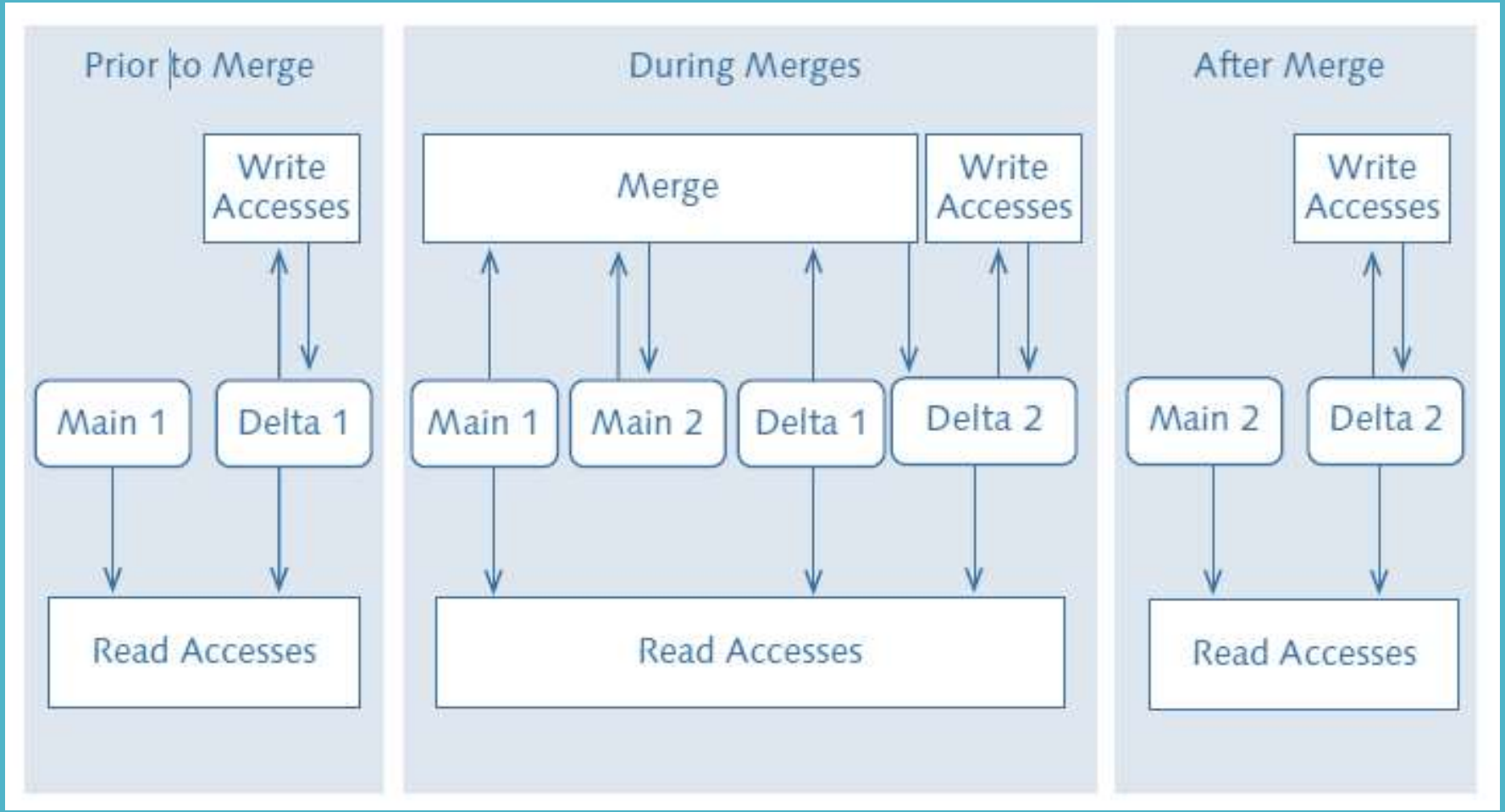
Insert Only Approach



We can have multiple entries in database with same primary key

MERGING

Insert Only Approach

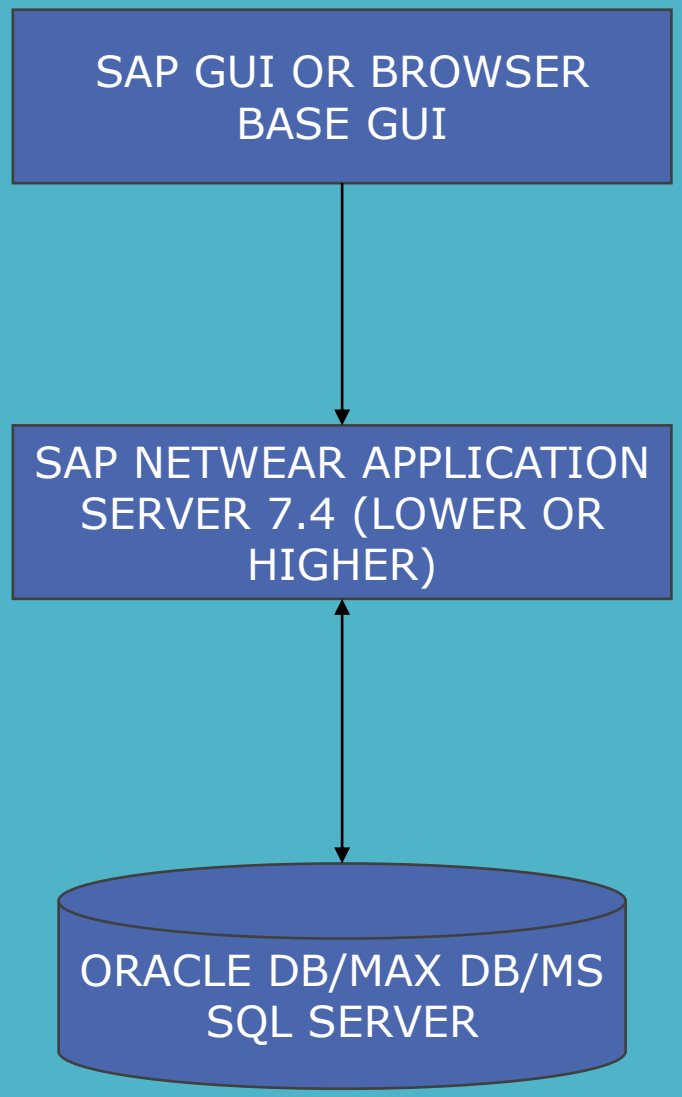


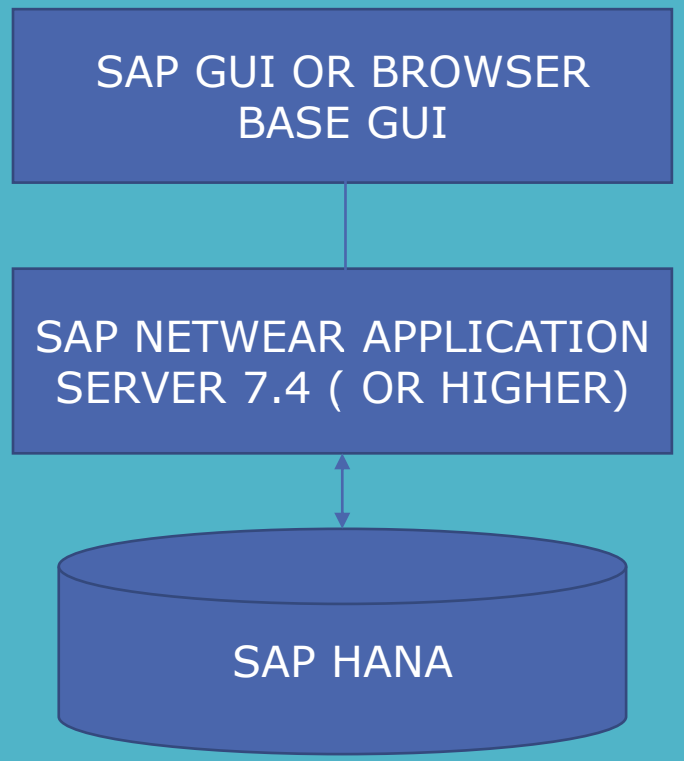
CODE PUSH DOWN

SAP NETWEAR APPLICATION SERVER

Code Push Down

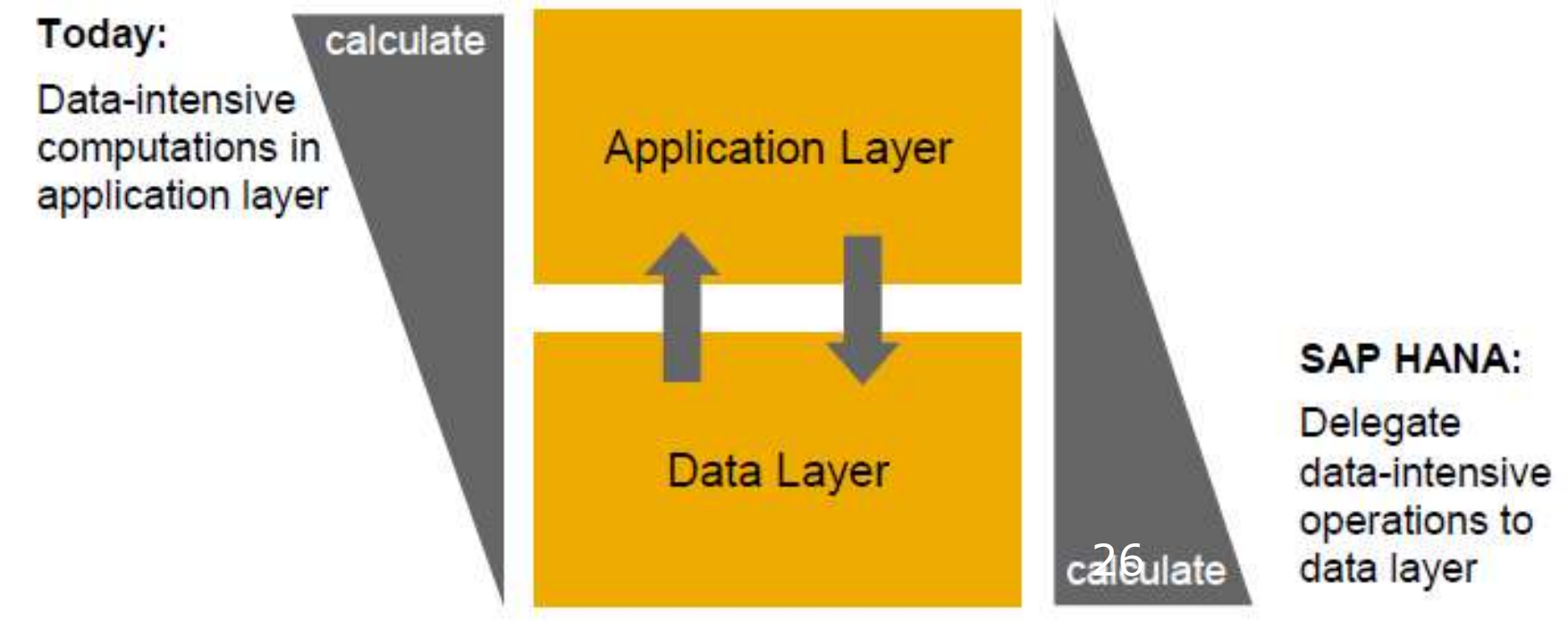
Architecture





IMPLICATIONS OF AN IN-MEMORY DATABASE

Code Push Down



In-Memory computing imperative:

- Avoid (unnecessary) movement of large volume of data
- Perform data-Intensive calculation in database

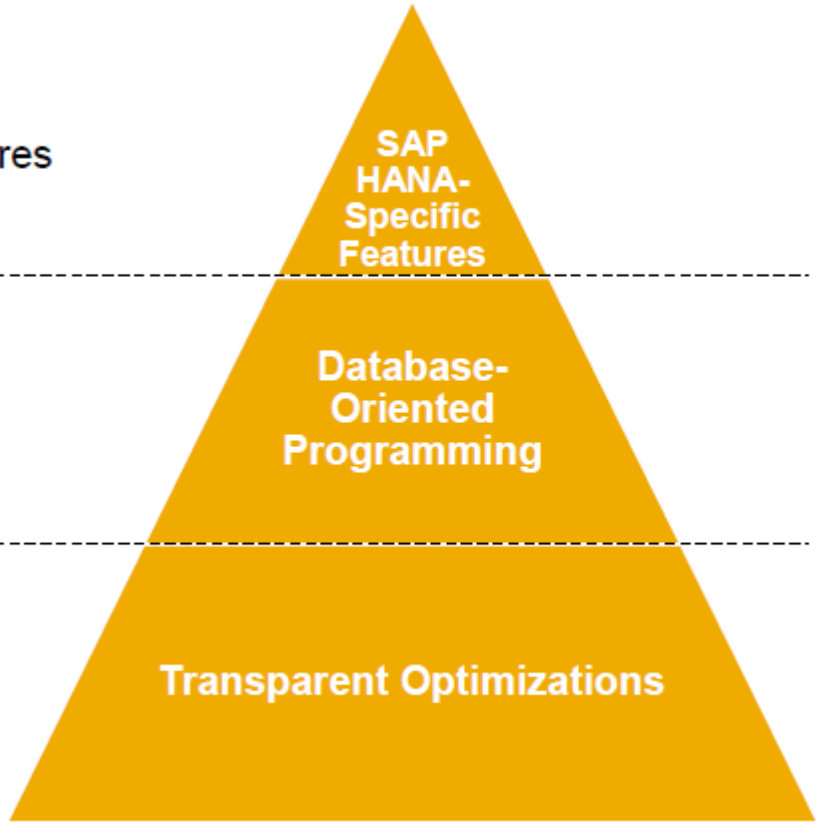
OVERVIEW

Code Push Down

- ABAP managed database procedures
- Native SQL

- Open SQL enhancements
- Advanced view definition

- Optimized protocols, such as Fast Data Access
- Table buffer enhancements
- …



OVERVIEW

Code Push Down

Performance Improvement vs. Code Adjustments

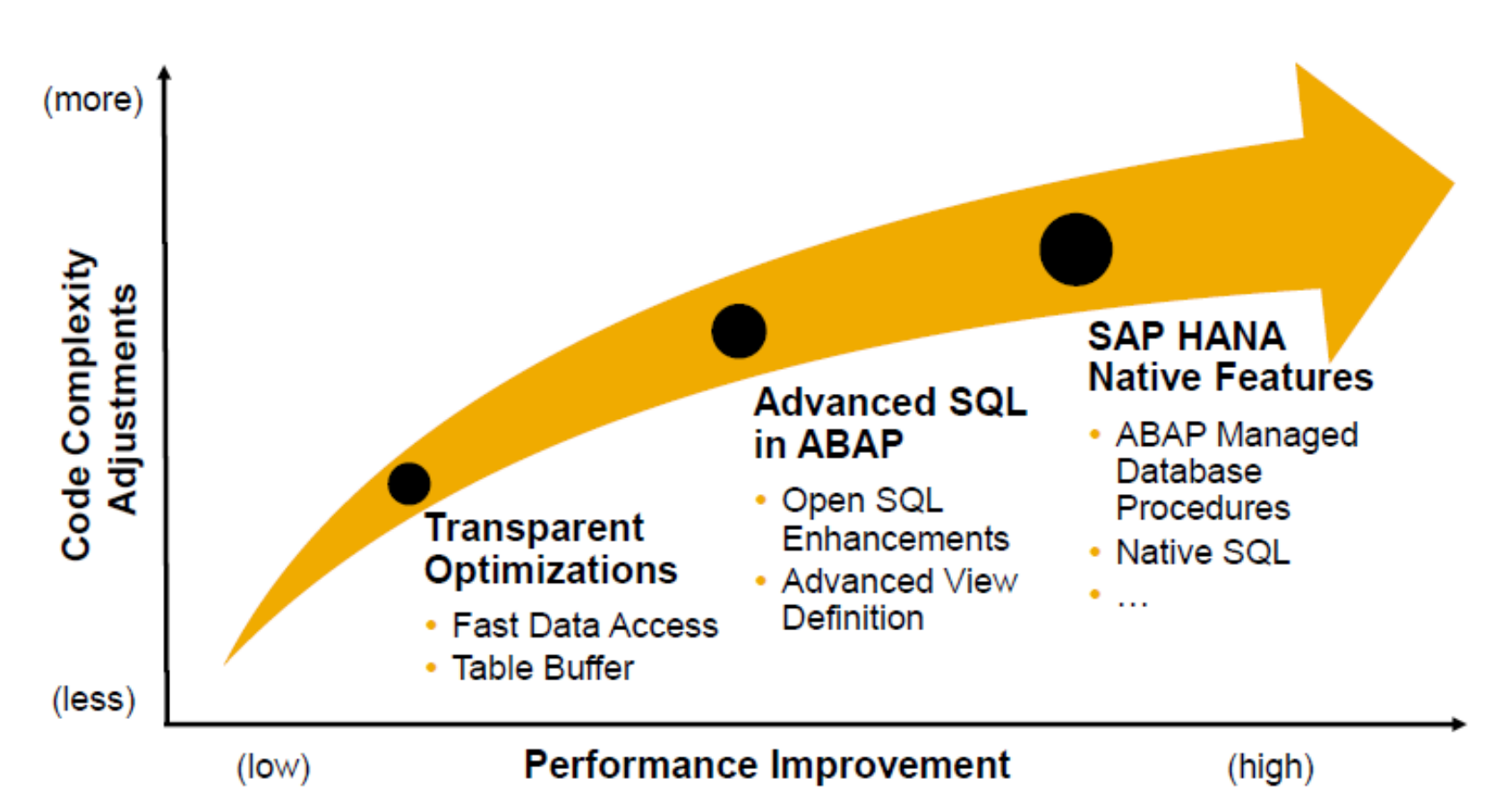
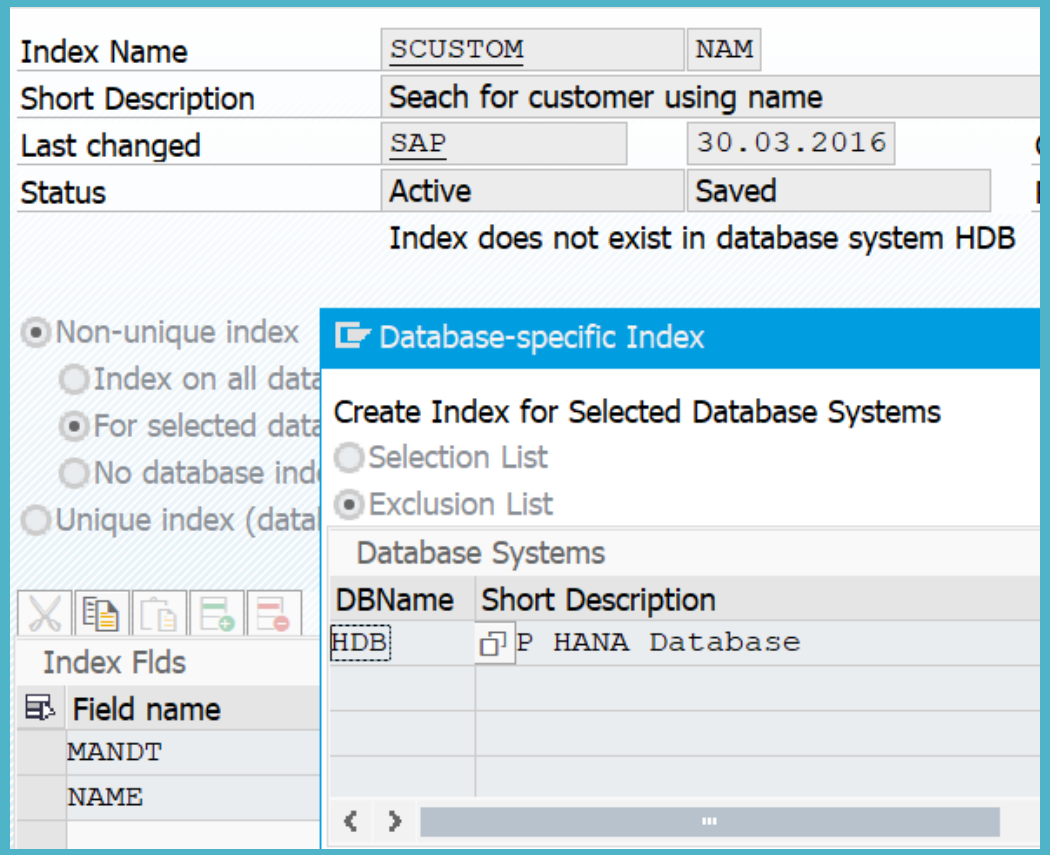


Table enhancement on ABAP on HANA

Deactivation of Secondary Indexes:-

ABAP on HANA DB



Other Database :-

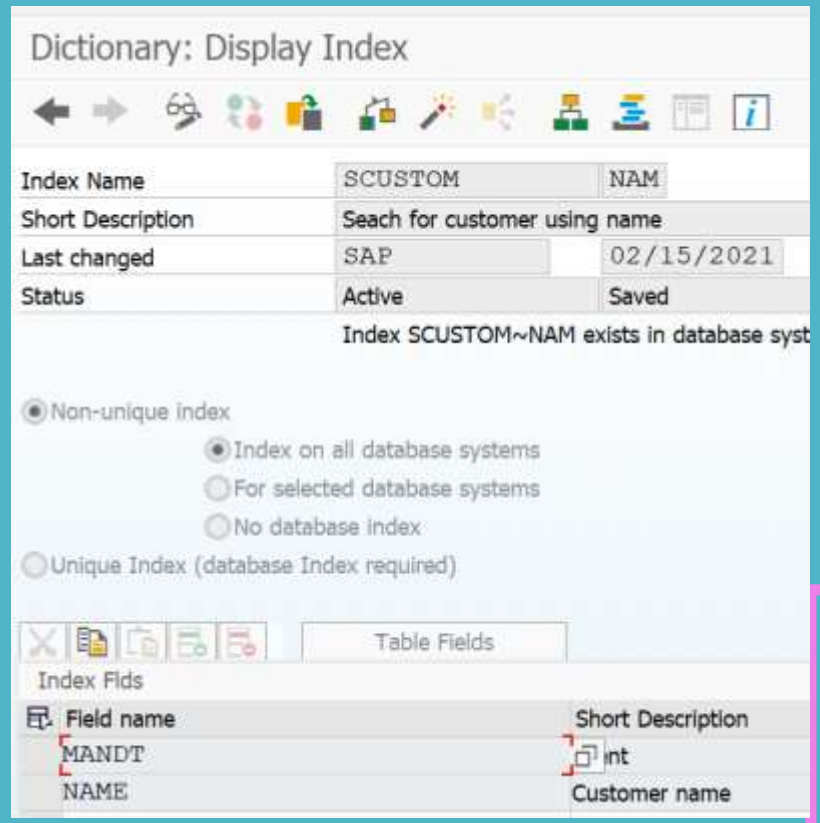
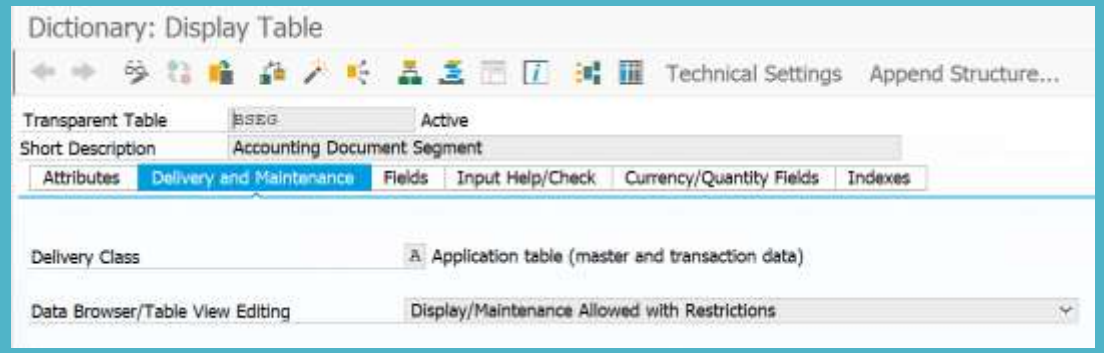


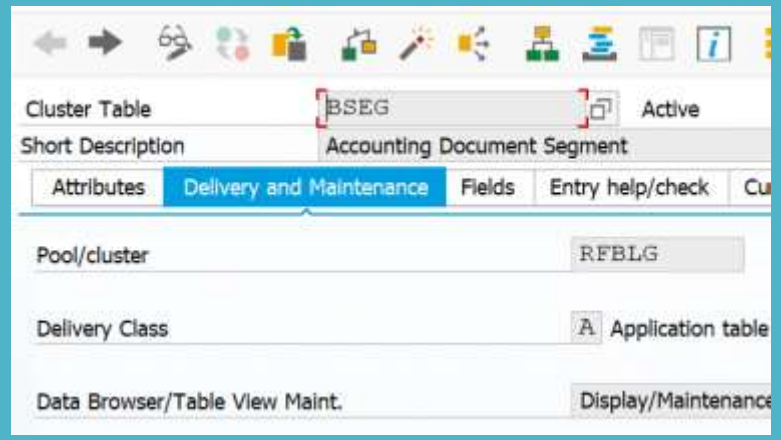
Table enhancement ABAP on HANA

De-Pooling and De-Clustering :-

ABAP on HANA DB



Other Database



SAP TECHNOMANIAC

Pooled table TABA

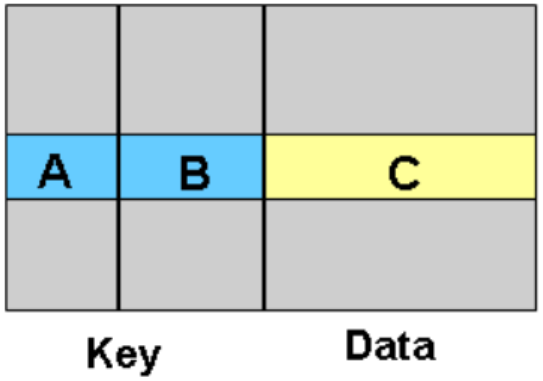
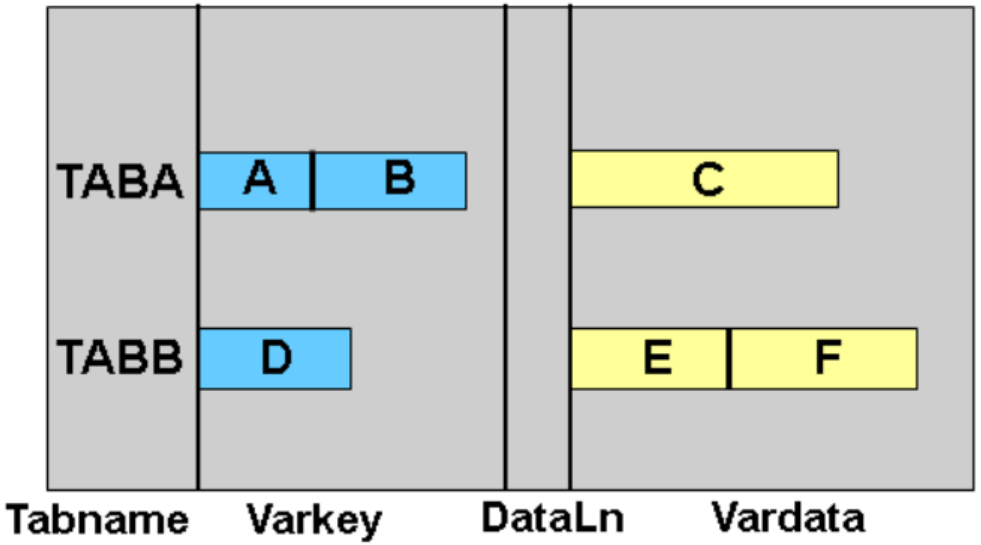
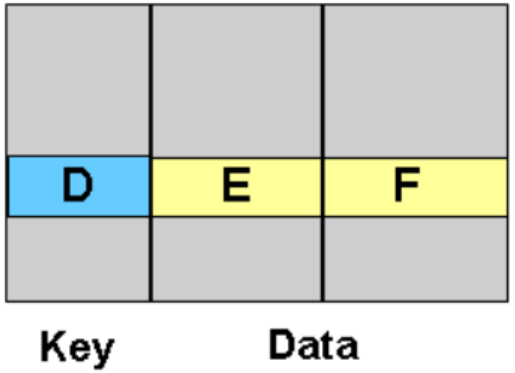


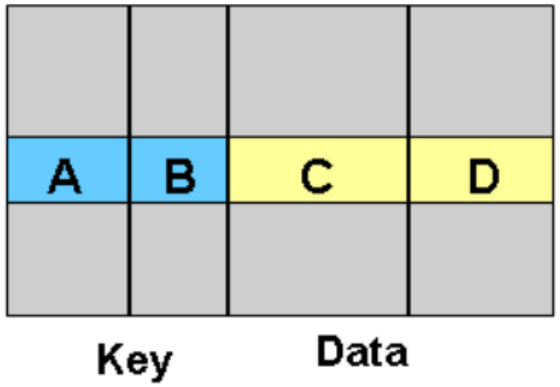
Table pool in the database



Pooled table TABB



Cluster table TABA



Cluster table TABB

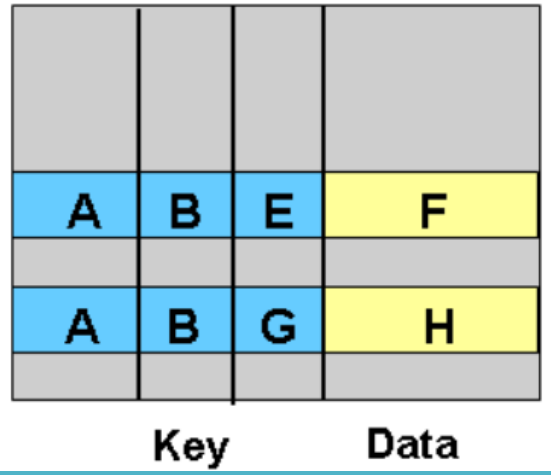


Table cluster in the database

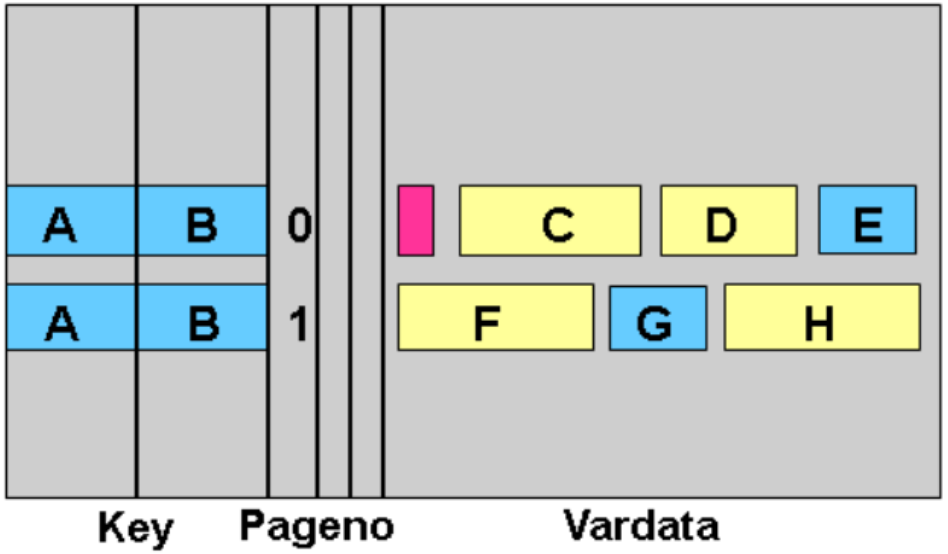
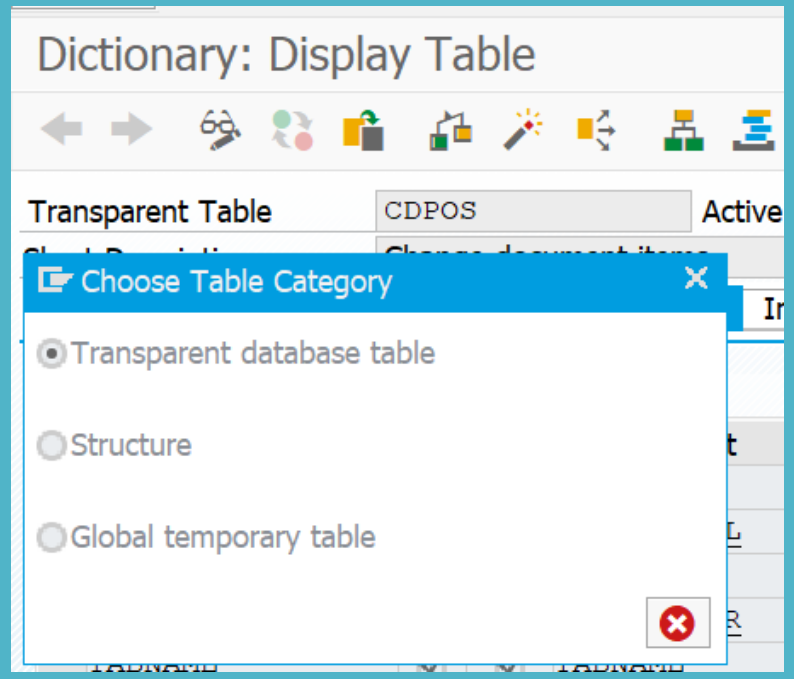


Table enhancement on ABAP on HANA

De-Pooling and De-Clustering :-

ABAP on HANA DB



Other Database :-

Extras → Change/Display table category

Table enhancement on ABAP on HANA

Empty Aggregate Tables:-

MKPF for document header information and
MSEG for document item data
MARC, MARD and MCHB ***Material master date***
MSSA containing only aggregated actual stock quantities

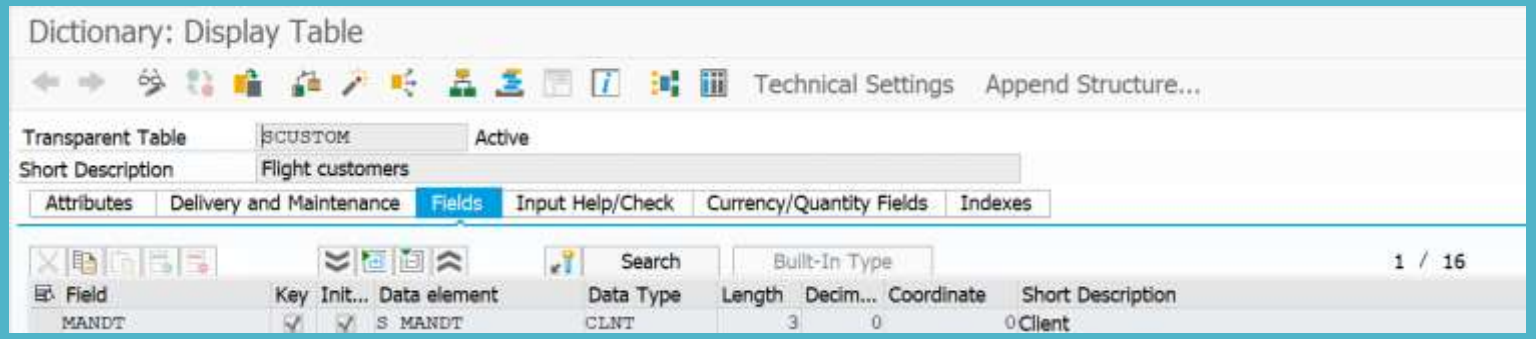
MATDOC :- MSEG + MKPF
MSSA not available you have to calculate on fly

Table	Table description	DDL Source of CDS View for redirect	View to read the content of the database table (w/o redirect to compatibility view)	View to read the master data attributes only
MKPF	Material document header	NSDM_DDL_MKPF	NSDM_V_MKPF	-
MSEG	Material document item	NSDM_DDL_MSEG	NSDM_V_MSEG	-
MARC	Plant Data for Material	NSDM_DDL_MARC	NSDM_V_MARC	V_MARC_MD

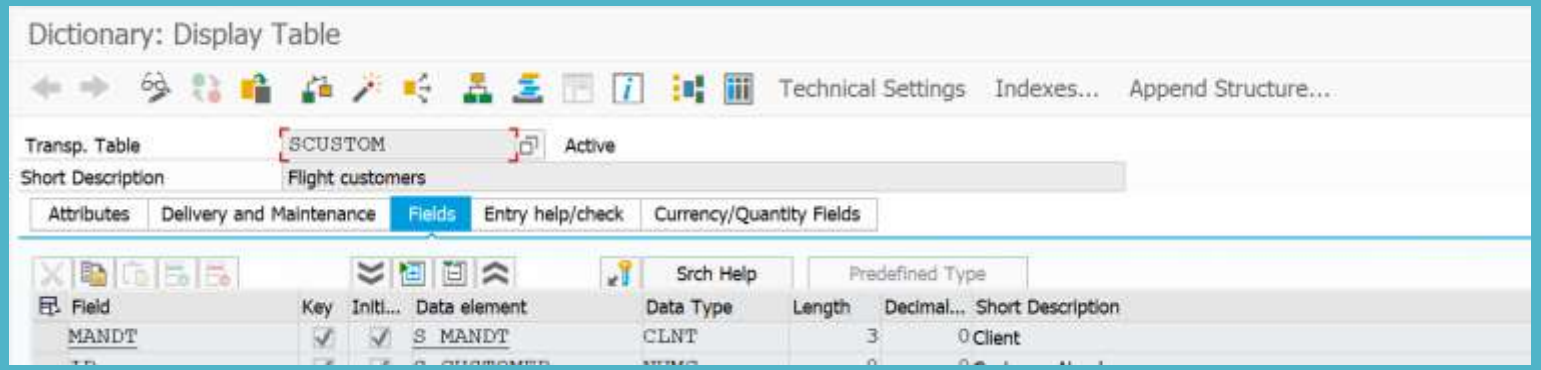
Table enhancement on ABAP on HANA

Deactivation of Secondary Indexes:-

ABAP on HANA DB



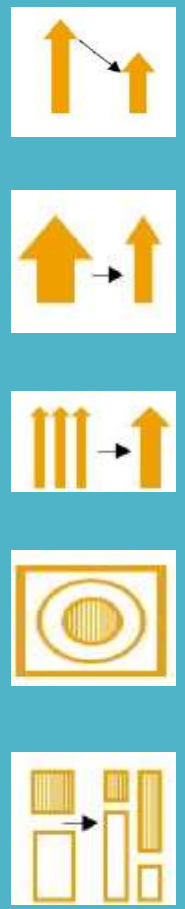
Other Database :-



SAP TECHNOMANIAC

SQL Performance Rules for SAP HANA

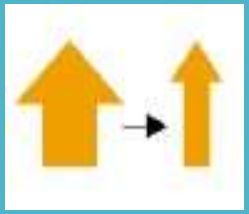
- Keep Result set small
- Minimize amount of transfer data
- Minimize number of DB access
- Minimize search overhead
- Keep unnecessary load away from DB



Five Golden rule in older ABAP

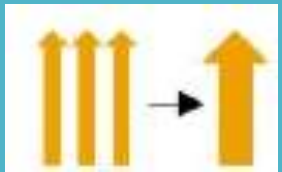
These guideline Become More Important on SAP HANA

- Minimize amount of transfer data



  Avoid unpacking columns unnecessary

- Minimize number of DB access



  Avoid unpacking same columns/tables unnecessary

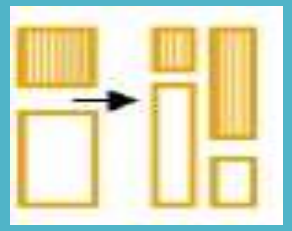
These guideline Become less Important on SAP HANA

- Minimize search overhead

  Where clauses using non indexed fields are not so bad anymore

- Keep unnecessary load away from DB

  Push data intensive calculation to SAP HANA



Structured Query Language

- What is Structured Query Language (SQL)

- Categories of SQL
  - Data Definition Language (DDL)
  - Data Manipulation Language (DML)
  - Data Control Language (DCL)

Limitation of Classical Open SQL

- Database independent syntax
- Database independent semantics
- Covers only a small part of standard DML
- No DDL or DCL statements

Limitation of ABAP Dictionary(Alternative DDL)

- Database independent tool (mostly graphical)
- Create and maintain definitions of database objects (tables, views)
- Covers only a small part of standard DDL features

What about DCL?

- No access control on DB level. In ABAP systems, the DB only knows one user.

- No transaction control on DB level. DB commit after each dialogue step.

Open SQL, Native SQL and the Database Interface

Database-independent interface

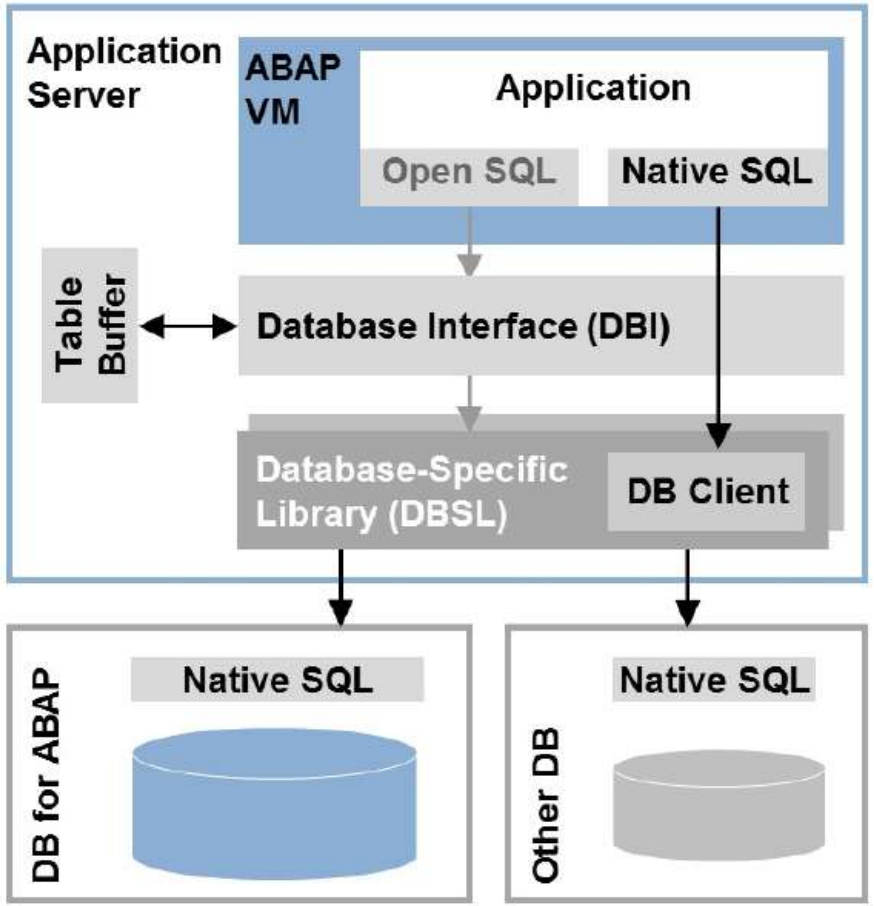
- Automatic client handling, ABAP table buffer …

Database-specific library

- Translates Open SQL → Native SQL
- Connects to database
- Different remote databases possible with multiple DB clients

ABAP database users and schemas

- One database account used, by deafult SAP<SID> or SAPR3
- Data stored in account's schema



ST05 Trace Main Records

Start Time	Duration	Records	Program Name	Object Name	Statement
13:47:49.142	371	1	SDYNPRAL	TFDIR	SELECT WHERE "FUNCNAME" = 'HOOK_DYNP_RAL_OFL' WITH RANGE_RESTRICTION('CURRENT')
13:47:49.153	11,011	1,029	Z_OPEN_VS_NATIVE_SQL	VBAK	SELECT <FDA READ> WHERE "MANDT" = '200' WITH RANGE_RESTRICTION('CURRENT')
13:47:49.174	41,387	1,388	Z_OPEN_VS_NATIVE_SQL	VBAP	SELECT <FDA WRITE> DISTINCT WHERE "VBAP" . "MANDT" = '200' AND "VBAP" . "VBELN" = "t_00" . "C

```sql
REPORT z_open_vs_native_sql.

SELECT
 FROM vbak
 FIELDS vbeln,vkorg
 INTO TABLE @DATA(lt_vbak).
IF sy-subrc IS INITIAL.
  SELECT
   FROM vbap
   FIELDS vbeln,posnr,netpr,matnr
   FOR ALL ENTRIES IN @lt_vbak
   WHERE vbeln = @lt_vbak-vbeln
   INTO TABLE @DATA(lt_vbap).
  IF sy-subrc IS INITIAL.
  ENDIF.
ENDIF.
```

SAP TECHNOMANIAC

Details for Selected SQL Trace Record

```
SELECT
  /* FDA READ */
  "VBELN" , "VKORG"
FROM
  "VBAK"
WHERE
  "MANDT" = ?
LIMIT  2
  WITH RANGE_RESTRICTION('CURRENT')
```

Variables

```
A0(CH,3)          = '200'
```

```
REPORT z_open_vs_native_sql.

SELECT
 FROM vbak
 FIELDS vbeln,vkorg
 INTO TABLE @DATA(lt_vbak).
IF sy-subrc IS INITIAL.
  SELECT
   FROM vbap
   FIELDS vbeln,posnr,netpr,matnr
   FOR ALL ENTRIES IN @lt_vbak
   WHERE vbeln = @lt_vbak-vbeln
   INTO TABLE @DATA(lt_vbap).
  IF sy-subrc IS INITIAL.
  ENDIF.
ENDIF.
```

Details for Selected SQL Trace Record

```
SELECT
  DISTINCT "VBELN" , "POSNR" , "NETPR" , "MATNR"
FROM
  "VBAP"
WHERE
  "MANDT" = ? AND "VBELN" IN ( ? , ? ) WITH RANGE_RESTRICTION('CURRENT')
```

Variables

```
A0(CH,3)          = '200'
A1(CH,10)         = '0060000035'
A2(CH,10)         = '0000000002'
```

SAP TECHNOMANIAC

New Syntax of Open SQL

```
**"using the "old" Open SQL syntax
SELECT so_id
       currency_code
       gross_amount
       delivery_status
  FROM snwd_so
  INTO TABLE lt_so_amount.
```

```
*"using the "new" Open SQL syntax
SELECT so_id,
       currency_code,
       gross_amount,
       delivery_status
  FROM snwd_so
  INTO TABLE @DATA(lt_result).
```

New Syntax of Open SQL

Old

- Blank separated
- No Marking ABAP Var

New

- **Comma separated**
- **Marking With @**

```
**"using the "old" Open SQL syntax
SELECT so_id
       currency_code
       gross_amount
       delivery_status
  FROM snwd_so
  INTO TABLE lt_so_amount.
```

```
*"using the "new" Open SQL syntax
SELECT so_id,
       currency_code,
       gross_amount,
       delivery_status
  FROM snwd_so
  INTO TABLE @DATA(lt_result).
```

S4 HANA ABAP vs TRADITIONAL ABAP

HANA DB	ORACLE , DB2 ,Microsoft SQL
IT is more powerful then traditional database	
Hardware innovation :- we are getting more thing in less price	
Software Innovation :- Columnar store +data compression + insert only approach + ..	These all innovation never came together for these DBSS
We have to process data at DB layer using technologies like CDS ,AMDP and Enhanced SQL	We were processing data at application layer after getting from DB

THANK YOU

CONTACT

Ram Niwas

LinkedIn :- https://www.linkedin.com/in/ram-niwas-04/

FB group :-
https://www.facebook.com/groups/586730659057346/

SUBSRIBE

LIKE

COMMENT

SHARE

THANKS

CORE DATA SERVICES (CDS)

- Why CDS?

- A set of domain-specific languages and services, called CDS, for defining and consuming semantically rich data models.

  - Domain-specific languages and services DDL,DQL,DCL

  - Semantically rich data models :- Annotation

CDS

- SQL function not possible
- Can’t be created and edited in SAP GUI .
- Outer join possible
- Union is possible
- Input Parameter to filter data
- Nested View
- Code Push Down follow
- Support Annotation
  - OData with annotation
  - Easy build Fiori app
- Support System variable

Classical DB view

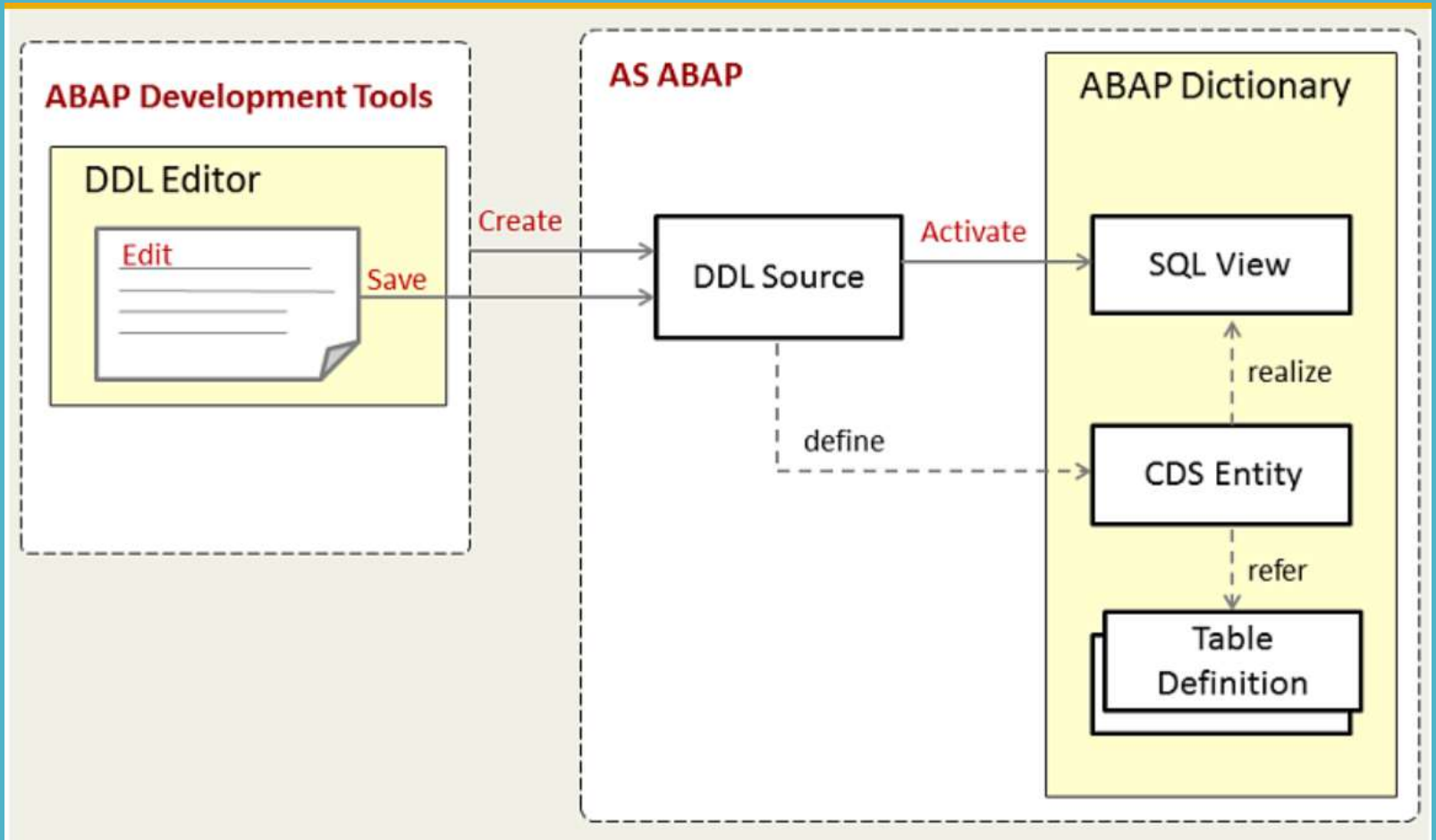
- SQL Function possible
- We can create in SAP GUI and edit it
- No Outer Join
- No Union
- No Input parameter
- Not supported
- Don’t folloe
- No Annotation
  - We have to use SEGW
  - We have to use JS

CDS Creation

- DDLS activated then -> CDS SQL VIEW + CDS Entity
- DDLS will be transported (DATA Defination)

CDS Creation

- DDLS activated then -> SQL VIEW + CDS Entity (HANA View in DB)

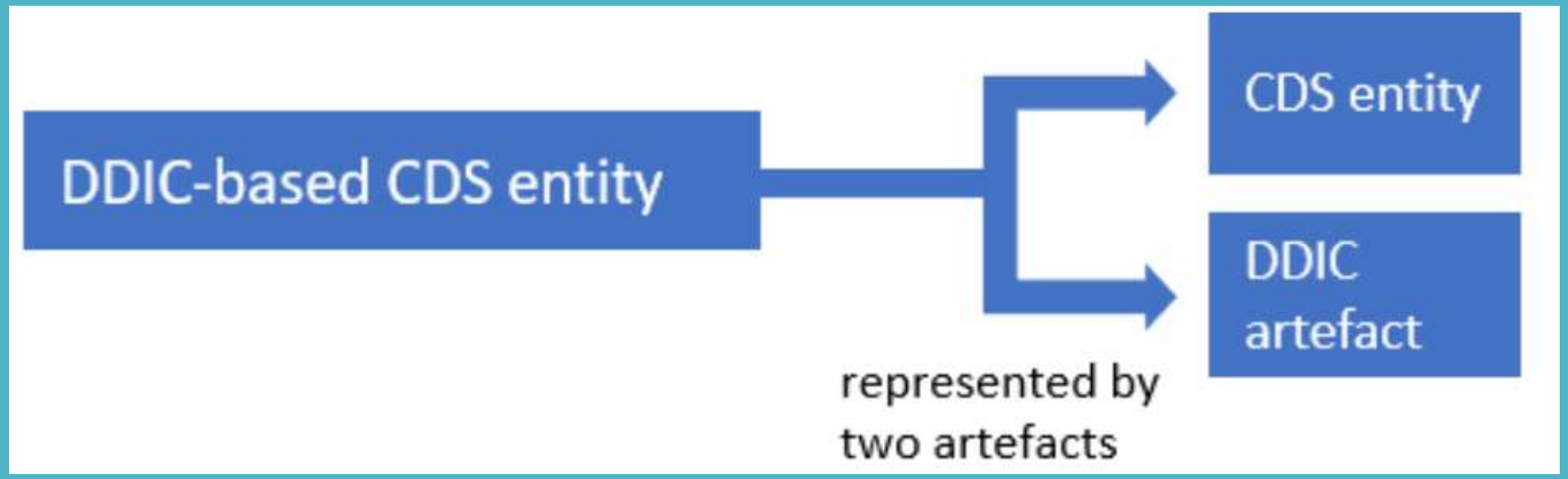


ABAP CDS Entities

- These are the data model based on DDL/DCL specification
- Managed by the ABAP Dictionary

- The following types of ABAP CDS entities are supported
  - ABAP CDS Views
  - ABAP CDS Table Functions

ABAP CDS - DDIC-Based Entities



Client handling

- The client dependency of a view is determined by the data sources used:
  - If one of the data sources used in the view is client-dependent, the view is client-dependent.
  - If none of the data sources used in the view is client-dependent, the view is client-independent.

Determining Client Handling

Left Side	Right Side	INNER JOIN	LEFT OUTER JOIN	RIGHT OUTER JOIN	CROSS JOIN
Client-dependent	Client-dependent	Compares the client columns in the ON condition	Compares the client columns in the ON condition	Compares the client columns in the ON condition	Transforms the cross join to an inner join using an ON condition for the client columns
Client-independent	Client-dependent	-	The left side is replaced by a cross join of the client-independent data source with the DDIC database table T000 and a comparison of the client columns in the ON condition.	-	-
Client-dependent	Client-independent	-	-	The right side is replaced by a cross join of the client-independent data source with the DDIC database table T000 and the client columns are compared in the ON condition.	-
Client-	Client-	-	-	-	-

Determining Client Handling

Left Side	Right Side	INNER JOIN	LEFT OUTER JOIN	RIGHT OUTER JOIN	CROSS JOIN
Client-dependent	Client-dependent	Compares the client columns in the ON condition	Compares the client columns in the ON condition	Compares the client columns in the ON condition	Transforms the cross join to an inner join using an ON condition for the client columns
Client-independent	Client-dependent	-	Compares the client column with the value of the session variable $session.client in the ON condition	-	-
Client-dependent	Client-independent	-	-	Compares the client column with the value of the session variable $session.client in the ON condition	-
Client-independent	Client-independent	-	-	-	-

Why CDS View Entities?

- No Use of SQL View which generated during activation of DDIC based CDS view.
  - We always use CDS entity name in select
  - We are creating unnecessary multiple object
- There will be always now one name instead of three name

CDS view entities

- 7.55 (In old version you don't get option to create)
- DEFINE VIEW ENTITY
- No additional DDIC based view Created
- Improved performance during view activation
- Optimization and simplification of syntax
- @AbapCatalog.sqlViewName annotation not available
- Name list are not supported
- Below annotation not required :-
  - @AbapCatalog.compiler.compareFilter: true
  - @AbapCatalog.preserveKey: true,
- Client handling takes place implicitly and doesn't require any development effort.
- Buffering annotation not suported

CDS - DDIC-Based View

- 7.40, SP05
- DEFINE VIEW.
- CDS-managed DDIC view created
- These are still supported to ensure downward compatibility.

ABAP CDS - View Entities

- Client handling annotation not required :-

```
@ClientHandling: {
    type: #INHERITED,
    algorithm:#SESSION_VARIABLE
}
```

- The client dependency of a view is determined by the data sources used.
- It is not possible to access the data of different clients in a single read. Due to algorithm:#SESSION_VARIABLE

ABAP CDS - View Entities

- The name of the DDL source and of the CDS entity must be identical
- In ABAP CDS, the CDS entity can be used as a data source of other CDS entities.
- In ABAP programs, the CDS entity can be used as a data type and in ABAP SQL read statements.
- The CDS entity cannot be used as a data type for definitions of dictionary objects.

Upcoming TOPIC

- CDS Use in ABAP Report
- CDS With Parameter
- CDS View on View
- CDS with literals and aggregate group by having clause
- Arithmetic operation and cast
- Case statement and Coalesce function
- View Extension
- CDS with Union and Union all
- CDS Join
- CDS Association
- ZRAM_TRAINING

ABAP CDS - Table Functions

- DDL source code

```
@EndUserText.label: 'CDS Table Function'
define table function Yddls_Table_Function_01
with parameters parameter_name : parameter_type
returns {
  client_element_name : abap.clnt;
  element_name : element_type;

}
implemented by method class_name=>method_name;
```

- DEFINE TABLE FUNCTION

- CDS table function includes the following:
  - The CDS entity
  - An AMDP function implementation

ABAP CDS - Basics

- CDS table functions can only be used in a database system that supports AMDP.
- CDS entity activated first-> AMDP function implementation is created.
- CDS entity is first transported then the AMDP function implementation

```
@EndUserText.label: 'CDS Table Function'
@ClientHandling.type: #CLIENT_INDEPENDENT
define table function Yddls_Table_Function_01
  with parameters
    clnt : abap.clnt
returns
{
  client_element_name : abap.clnt;
  vbeln               : vbeln;

}
implemented by method
  YCl_cds_table_function=>get_so_data;
```

The CDS entity uses

- It can be used as a
  - Data source of other CDS entities.
  - Data type and in ABAP SQL read statements (Select).
  - Data type for definitions of dictionary objects.

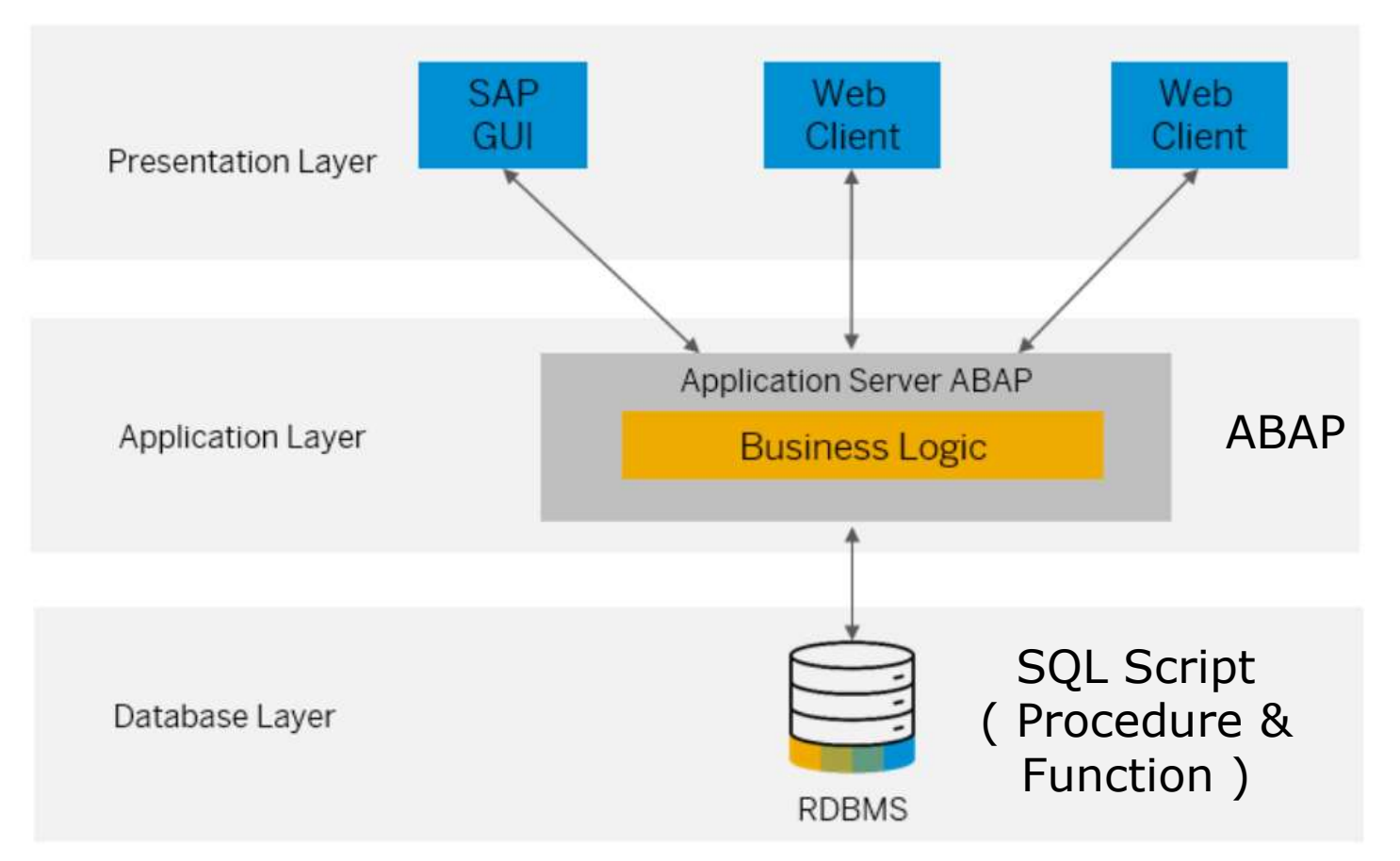
```
@EndUserText.label: 'CDS Table Function'
@ClientHandling.type: #CLIENT_INDEPENDENT
define table function Yddls_Table_Function_01
  with parameters
    clnt : abap.clnt
returns
{
  client_element_name : abap.clnt;
  vbeln               : vbeln;

}
implemented by method
  YCl_cds_table_function=>get_so_data;
```

AMDP

Framework used to manage and call database procedures and database functions (User define function UDF) in ABAP.

SAP TECHNOMANIAC

AMDP



SAP three Tier Architecture

SQL Script Logic Containers

- SQL Script there are two different logic containers
  - Procedure
  - User-Defined Function
    - Scalar User-Defined Function
    - Table User-Defined Function.

- These allows you to group the SQL statement and other logic into a single block.

SQL Script Logic Containers

Database procedure

- No Mandatory return value
- We call this using CALL statement

```
CALL proc (1000, 'EUR', ?, ?);
```

- Read + Write (INSERT UPDATE SELECT)

User Define Function (UDF)

- UDFs accept multiple input parameters and return exactly one result which is mandatory
- It can be directly use in select read position
- Read only operation (SELECT)

- AMDP Manages
  1. **AMDP procedures**
  2. AMDP functions
     - AMDP table functions
       - AMDP table functions for AMDP methods
       - AMDP Table Functions for CDS Table Functions
     - AMDP scalar function

- AMDP Manages
  - AMDP procedures and AMDP functions

- SQL Script, L … in an

  - AMDP procedure implementation
    - AMDP method without return value and this indicated by “BY DATABASE PROCEDURE”
    - It called just like ABAP method

```
METHOD get_so_data BY DATABASE PROCEDURE
                   FOR HDB
                   LANGUAGE SQLSCRIPT
                   OPTIONS READ-ONLY
                   USING vbak.
```

- AMDP Manages
  - AMDP procedures and AMDP functions

- SQL Script, L … in an

```
METHOD get_so_data BY DATABASE FUNCTION
                   FOR HDB
                   LANGUAGE SQLSCRIPT
                   OPTIONS READ-ONLY
                   USING vbak.
```

  - AMDP function implementation
    - AMDP method **with return value** that is indicated by "BY DATABASE FUNCTION"
    - AMDP scalar function :- can be called in ABAP programs like a regular functional method
    - AMDP table function :- It can be used as a data source of ABAP SQL read statements using the CDS entity

General Points AMDP

- AMDP only supports DB procedures & functions from the HANA
- AMDP is designed so that stored procedures and functions from other database systems can also be supported.
- CL_ABAP_DBFEATURES ->CALL_AMDP_METHOD
- ADT only
- AMDP framework uses the Native SQL interface to access the database.

When to use AMDP?

- The use of AMDP is **not recommended** if the same task can be achieved using ABAP SQL (or ABAP CDS).

- AMDP should be used only if it enables **database-specific functions** to be accessed that do not exist in ABAP SQL

- If large process flows or analyses that incur repeated transports of large amounts of data between the database and the AS instance can be swapped out

AMDP - Classes

- This is a global class with Interface IF_AMDP_MARKER_* Interface for HANA DB IF_AMDP_MARKER_HDB

- An AMDP class can contain both regular methods and AMDP methods.

```
CLASS ycl_cds_table_function DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .
  PUBLIC SECTION.
    INTERFACES if_amdp_marker_hdb.
    CLASS-METHODS get_so_data
        FOR TABLE FUNCTION yddls_table_function_01 .
  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.
```

AMDP - Methods

- There are two types of AMDP methods
  - AMDP procedures :- methods without a return value
    addition BY DATABASE PROCEDURE
  - AMDP functions. :- methods with a return value
    addition BY DATABASE FUNCTION

- The implementation of an AMDP method is saved to the ABAP database schema by the ABAP runtime environment.

- FM DB_DBSCHEMA_CURRENT will return ABAP DB SCHEMA

AMDP - Function Implementations

- AMDP Table Functions
  - AMDP table functions for AMDP methods
  - AMDP Table Functions for CDS Table Functions

- AMDP Scalar Functions

AMDP Table Functions for CDS Table Functions syntax

- It can only be declared in the public visibility section of a static AMDP class. It is not possible in interfaces.

```
  PUBLIC SECTION.
    INTERFACES if_amdp_marker_hdb.

    CLASS-METHODS get_so_data
        FOR TABLE FUNCTION yddls_table_function_01 .
```

```
define table function Yddls_Table_Function_01
  with parameters
    @Environment.systemField: #CLIENT
    clnt : abap.clnt
returns
{
  client_element_name : abap.clnt;
  vbeln               : vbeln;

}
implemented by method
  YCl_cds_table_function=>get_so_data;
```

```
METHOD get_so_data BY DATABASE FUNCTION
                   FOR HDB
                   LANGUAGE SQLSCRIPT
                   OPTIONS READ-ONLY
                   USING vbak.
  it_vbeln =  select so.mandt as client_element_name,
              so.vbeln as vbeln
        from vbak as so;
  return :it_vbeln;
ENDMETHOD.
```

AMDP Table Functions for CDS Table Functions syntax

- Parameter is generated in accordance with the associated CDS table function using the statement DEFINE TABLE FUNCTION in the ABAP CDS CDS DDL:

```
define table function Yddls_Table_Function_01
  with parameters
    @Environment.systemField: #CLIENT
    clnt : abap.clnt
returns
{
  client_element_name : abap.clnt;
  vbeln               : vbeln;

}
implemented by method
  YCl_cds_table_function=>get_so_data;
```

```
  PUBLIC SECTION.
    INTERFACES if_amdp_marker_hdb.

    CLASS-METHODS get_so_data
        FOR TABLE FUNCTION yddls_table_function_01 .

```

```
METHOD get_so_data BY DATABASE FUNCTION
                   FOR HDB
                   LANGUAGE SQLSCRIPT
                   OPTIONS READ-ONLY
                   USING vbak.
  it_vbeln =  select so.mandt as client_element_name,
              so.vbeln as vbeln
         from vbak as so;
  return :it_vbeln;
ENDMETHOD.
```

AMDP Table Functions for CDS Table Functions

- CDS Entity can be used as a data source of:-
  - ABAP SQL read statements in ABAP
  - CDS view entities in the CDS DDL of ABAP CDS.
  - CDS DDIC-based views in the CDS DDL of ABAP CDS.
- Calls from other AMDP methods are possible.
- Calls as regular functional methods are not possible in an ABAP program.
- Calls from non-AMDP-managed database procedures or database functions are, like any database table function, possible but not recommended.

```
define table function Yddls_Table_Function_01
  with parameters
    @Environment.systemField: #CLIENT
    clnt : abap.clnt
returns
{
  client_element_name : abap.clnt;
  vbeln               : vbeln;

}
implemented by method
  YCl_cds_table_function=>get_so_data;
```

```
METHOD get_so_data BY DATABASE FUNCTION
                   FOR HDB
                   LANGUAGE SQLSCRIPT
                   OPTIONS READ-ONLY
                   USING vbak.
  it_vbeln =  select so.mandt as client_element_name,
              so.vbeln as vbeln
          from vbak as so;
  return :it_vbeln;
ENDMETHOD.
```

SAP TECHNOMANIAC

AMDP Table Functions for CDS Table Functions

- When an AMDP function implementation is created for a CDS table function, it must already exist as an active function.
- When a new CDS table function is activated, an empty AMDP table function is created in the database. This function raises an exception if a non-AMDP access is performed.
- When the CDS table function is accessed by the ABAP runtime environment for the first time (for example, using ABAP SQL), the AMDP function implementation implements the empty AMDP table function.
- Other frameworks that evaluate CDS entities using their annotations, however, can also cause the AMDP table function to be implemented. It is possible to access the table function in a native way only after the implementation.

ABAP CDS - Client Handling in CDS Table Functions

- The CDS annotation @ClientHandling.type
  - #CLIENT_DEPENDENT enables client dependency.
  - #CLIENT_INDEPENDENT disables client dependency.

- Client dependency is enabled by default.

```
@ClientHandling.type: #CLIENT_DEPENDENT
@ClientHandling.type: #CLIENT_INDEPENDENT
define table function Yddls_Table_Function_01
  with parameters
    @Environment.systemField: #CLIENT
    clnt : abap.clnt
returns
{
  client_element_name : abap.clnt;
  vbeln               : vbeln;

}
implemented by method
  YCl_cds_table_function=>get_so_data;
```

AMDP Table Functions for CDS Table Functions

- The line type of the return value result of an AMDP function implementation for a client-dependent CDS table function does not contain a client field, even though this field must be declared in the element list.

```
@EndUserText.label: 'CDS Table Function'
@ClientHandling.type: #CLIENT_DEPENDENT
define table function Yddls_Table_Function_01
  with parameters
    @Environment.systemField: #CLIENT
    clnt : abap.clnt
returns
{
  client_element_name : abap.clnt;
  vbeln               : vbeln;

}
implemented by method
  YCl_cds_table_function=>get_so_data;
```

- An AMDP function implementation can only be associated with a single CDS table function

```
    RETURN
    SELECT so.mandt as client_element_name,
           so.vbeln as vbeln
        from vbak as so
            where so.mandt = :clnt ;
```

Client-dependent CDS Table Function

- The element list of a client-dependent CDS table function must have an explicit client field (Get an error)

- Highlighted Client field is not a component of the structured data type represented by the CDS entity.

```
define table function Yddls_Table_Function_01
  with parameters
    @Environment.systemField: #CLIENT
    clnt : abap.clnt
returns
{
  client_element_name : abap.clnt;
  vbeln               : vbeln;

}
implemented by method
  YCl_cds_table_function=>get_so_data;
```

Client-dependent CDS Table Function

- Always data comes from current client irrespective we didn't specify client in native SQL
- For performance reasons always pass client as parameter and default it using environment annotation
- SQL specific client handling discuss in future video with CLIENT SPECIFIED and USING CLIENT

```
define table function Yddls_Table_Function_01
  with parameters
    @Environment.systemField: #CLIENT
    clnt : abap.clnt
returns
{
  client_element_name : abap.clnt;
  vbeln               : vbeln;

}
implemented by method
  YCl_cds_table_function=>get_so_data;
```

Client-independent CDS Table Function

- The element list of a client-dependent CDS table function does not need to have an explicit client field
- If the first element has the type CLNT, it does not function as a client field. Instead, it is a column of the tabular return value .
- The annotation @Environment.systemField: #CLIENT cannot be used in the parameter list of a client-independent CDS table function.

```
@EndUserText.label: 'CDS Table Function'
@ClientHandling.type: #CLIENT_INDEPENDENT
define table function Yddls_Table_Function_01
  with parameters
    clnt : abap.clnt
returns
{
  client_element_name : abap.clnt;
  vbeln               : vbeln;

}
implemented by method
  YCl_cds_table_function=>get_so_data;
```

- The AMDP function implementation must not exist when the CDS table function is created and activated.

- An AMDP function implementation can only be associated with a single CDS table function function (1: 1 relation).

AMDP Table Functions for CDS Table Functions

- The input parameters in the AMDP function implementation are determined by the input parameters of the CDS table function.
- A return value with the type of a standard table is created with an empty table key named result with a structured line type.
- The components of the line type are determined by the elements of the CDS table function.

```
define table function Yddls_Table_Function_01
  with parameters
    @Environment.systemField: #CLIENT
    clnt : abap.clnt
returns
{
  client_element_name : abap.clnt;
  vbeln               : vbeln;

}
implemented by method
  YCl_cds_table_function=>get_so_data;
```

AMDP Table Functions

- RETURNING Must
- Allow tabular input parameters
- No input/output parameters or output parameters.
- No class-based exceptions can be declared using RAISING
- AMDP function implementation must be restricted to reads. Using OPTIONS READ-ONLY

When to use CDS table Function?

When to use CDS table Function?

- To access **cross schema** tables in CDS views, so we are using table functions to do that.

- CDS Table functions are used for instance in case you want to integrate some **HANA features**, not available directly via the ABAP SQL or CDS layer, into your VDM.

- Table functions that are implemented natively on the database we can call these directly in CDS table function

- If we have to encapsulate complex logic that would require > 2 CDS views to achieve. Ex :- ABAP CDS View: join tables on columns of different type

Video topics :-

1. Intro, Syntax, and Imp Points
2. AMDP Framework
3. Database Procedure and Function
4. AMDP Procedure and Function
5. AMDP General Point
6. When to use AMDP?
7. AMDP-Classes
8. AMDP-Methods
9. AMDP for CDS table Function

SAP TECHNOMANIAC

ABAP CDS - Client Handling in CDS Table Functions

- The CDS annotation @ClientHandling.type
  - #CLIENT_DEPENDENT enables client dependency.
  - #CLIENT_INDEPENDENT disables client dependency.

- Client dependency is enabled by default.

```
@ClientHandling.type: #CLIENT_DEPENDENT
@ClientHandling.type: #CLIENT_INDEPENDENT
define table function Yddls_Table_Function_01
  with parameters
    @Environment.systemField: #CLIENT
    clnt : abap.clnt
returns
{
  client_element_name : abap.clnt;
  vbeln               : vbeln;

}
implemented by method
  YCl_cds_table_function=>get_so_data;
```

Client-dependent CDS Table Function

- The element list of a client-dependent CDS table function must have an explicit client field (Get an error)

- Highlighted Client field is not a component of the structured data type represented by the CDS entity.

```
define table function Yddls_Table_Function_01
  with parameters
    @Environment.systemField: #CLIENT
    clnt : abap.clnt
returns
{
  client_element_name : abap.clnt;
  vbeln               : vbeln;

}
implemented by method
  YCl_cds_table_function=>get_so_data;
```

Client-dependent CDS Table Function

- The line type of the return value result of an AMDP function implementation for a client-dependent CDS table function does not contain a client field, even though this field must be declared in the element list.

```
@EndUserText.label: 'CDS Table Function'
@ClientHandling.type: #CLIENT_DEPENDENT
define table function Yddls_Table_Function_01
  with parameters
    @Environment.systemField: #CLIENT
    clnt : abap.clnt
returns
{
  client_element_name : abap.clnt;
  vbeln               : vbeln;

}
implemented by method
  YCl_cds_table_function=>get_so_data;
```

```
RETURN
SELECT so.mandt as client_element_name,
       so.vbeln as vbeln
    from vbak as so
        where so.mandt = :clnt ;
```

We will continue this topic in next video link is available on-screen, cards and in description

In next video we will call this CDS table function in ABAP program and try to understand further client handling

This Video will be continuation of last video

SAP TECHNOMANIAC

Client-dependent CDS Table Function

- Always data comes from current client irrespective we didn't specify client in native SQL

- For performance reasons always pass client as parameter and default it using environment annotation

```
define table function Yddls_Table_Function_01
  with parameters
    @Environment.systemField: #CLIENT
    clnt : abap.clnt
returns
{
  client_element_name : abap.clnt;
  vbeln               : vbeln;

}
implemented by method
  YCl_cds_table_function=>get_so_data;
```

Client-independent CDS Table Function

- Client field is not mandatory
- If we specify it acts as a column of the tabular return value .
- The annotation @Environment.systemField: #CLIENT cannot be used in the parameter list of a client-independent CDS table function.

```
@EndUserText.label: 'CDS Table Function'
@ClientHandling.type: #CLIENT_INDEPENDENT
define table function Yddls_Table_Function_01
  with parameters
    clnt : abap.clnt
returns
{
  client_element_name : abap.clnt;
  vbeln               : vbeln;

}
implemented by method
  YCl_cds_table_function=>get_so_data;
```

dependent

- When a client-dependent CDS table function is accessed using SELECT without the obsolete addition CLIENT SPECIFIED, only those rows are selected implicitly from the result set of the function that contain the ID of the current client or the client specified in the addition USING CLIENT in the client field.
- Note that if the ABAP-specific session variables CLIENT and CDS_CLIENT are accessed in the implementation of a CDS table function, the addition USING CLIENT of the ABAP SQL statement SELECT only acts on the session variableCDS_CLIENT. If the AMDP function is used in an AMDP method called from ABAP, there is no equivalent for USING CLIENT.
- If the obsolete addition CLIENT SPECIFIED is specified, the column is added to the result set and is filled with the associated client ID for each row. Before this column can be used in the SELECT statement, a name must be assigned to it after the addition CLIENT SPECIFIED. If the name is not defined, no addressing is possible in a clause and no inline declarations can be made using @DATA(...) after INTO. The defined name is also used in the case of INTO CORRESPONDING. If no name is defined, the client column is not transported.

AMDP Table Functions for CDS Table Functions

- The input parameters in the AMDP function implementation are determined by the input parameters of the CDS table function.
- A return value with the type of a standard table is created with an empty table key named result with a structured line type.
- The components of the line type are determined by the elements of the CDS table function.

```
define table function Yddls_Table_Function_01
  with parameters
    @Environment.systemField: #CLIENT
    clnt : abap.clnt
returns
{
  client_element_name : abap.clnt;
  vbeln               : vbeln;

}
implemented by method
  YCl_cds_table_function=>get_so_data;
```

AMDP Table Functions

- RETURNING Must
- Allow tabular input parameters
- No input/output parameters or output parameters.
- No class-based exceptions can be declared using RAISING
- AMDP function implementation must be restricted to reads. Using OPTIONS READ-ONLY

IT is the one of the Industry which does not have nepotism.
Forget about kids' fathers are still ***upgrading*** their ***skills*** to survive!! ☺ ☺

SAP TECHNOMANIAC

AMDP procedure implementation

- AMDP method **without return value** that is indicated by "**BY DATABASE PROCEDURE**"
- It always implemented in DB language as "ABAP Managed Database Procedures"
- It is declared in an AMDP class like a regular static method or instance method in any visibility section.
- In declaration part we cant differentiate b/w regular and AMDP method

AMDP procedure implementation

- The database objects of the current database schema accessed in the AMDP method must be declared using an addition USING.

Imp point for Parameter Interface

- The typing of the parameters cannot be generic
- Only elementary data types and table types with a structured line type can be used
- The parameters must be declared using VALUE for pass by value. Pass by reference is not allowed.
- Return values cannot be declared using RETURNING.
- Only input parameters can be flagged as optional parameters.

Parameter names:

- Parameter names cannot start with the characters "%_".
- The parameter name connection can only be used for an input parameter of type DBCON_NAME, if the name of the database connection can be passed to the input parameter.
- The parameter name client is reserved for future enhancements.
- The parameter name endmethod is not allowed.

The following restrictions apply to method implementation:

- An AMDP method must not be empty.
- DDL statements are not allowed for creating, changing or deleting database objects.

AMDP OPTIONS in Declaration

- If Implementation is normal
- Tag Interface mandatory
- Read Only
- CDS SESSION CLIENT clnt|CURRENT
- ABAP specific Session variable on HANA DB
- $session.client = sy-mandt

```
CLASS zcl_amdp_class DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.
    INTERFACES if_amdp_marker_hdb.
    TYPES: BEGIN OF ty_cust,
             cust_name TYPE kna1-name1,
             netwr     TYPE vbak-netwr,
           END OF ty_cust,
           tt_cust TYPE TABLE OF ty_cust.

    METHODS get_cust_detail
      AMDP OPTIONS READ-ONLY
                   CDS SESSION CLIENT mandt
      IMPORTING
        VALUE(et_num)    TYPE i
        VALUE(mandt)     TYPE mandt
      EXPORTING
        VALUE(top_cust)  TYPE tt_cust
        VALUE(flop_cust) TYPE tt_cust.


  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.
```

SAP TECHNOMANIAC

ABAP-Specific Session Variables in SAP HANA

- Session variables are global variables in the SAP HANA database
- They can be read there with the built-in function SESSION_CONTEXT ('SYSTVAR')
- CLIENT = SY-MANDT
- CDS_CLIENT = SY-MANDT
- APPLICATIONUSER = SY-UNAME
- LOCALE_SAP = SY-LANGU
- SAP_SYSTEM_DATE = SY-DATUM

AMDP OPTIONS in Declaration

```
CLASS zcl_amdp_class DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.
    INTERFACES if_amdp_marker_hdb.
    TYPES: BEGIN OF ty_cust,
             cust_name TYPE kna1-name1,
             netwr     TYPE vbak-netwr,
           END OF ty_cust,
           tt_cust TYPE TABLE OF ty_cust.

    METHODS get_cust_detail
      AMDP OPTIONS READ-ONLY
                   CDS SESSION CLIENT mandt
      IMPORTING
        VALUE(et_num)    TYPE i
        VALUE(mandt)     TYPE mandt
      EXPORTING
        VALUE(top_cust)  TYPE tt_cust
        VALUE(flop_cust) TYPE tt_cust.


  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.
```

- READ-ONLY
- CDS SESSION CLIENT clnt|CURRENT

ABAP-Specific Session Variables in SAP HANA

- CDS_CLIENT like CLIENT but with the following differences:
  - CDS_CLIENT contains the same value as CLIENT by default, but can be modified
    - During the execution of an ABAP SQL statement by the addition USING CLIENT
    - In an AMDP method call from ABAP by the addition AMDP OPTIONS CDS SESSION CLIENT.

AMDP - Client Handling

- AMDP does not support implicit client handling.
- We should pass client in parameter interface
- Using an input parameter for the client ID is particularly advisable for AMDP function implementations of client-dependent CDS table functions.
- Exception :- When we are calling another CDS view which is using @ClientHandling.algorithm:#SESSION_VARIABLE.
- In this case CDS_CLIENT is used in where condition which came from AMDP OPTIONS CDS SESSION CLIENT .
- If it is not specified or passed from importing
- parameter then and it is different from CDS_CLIENT
- we will get empty result

Schema Name for ABAP server

TRL_EN
- Favorite Packages
- Favorite Objects
- Released Objects

YTS_200_gabap3_en

ZCL_AMDP_CLASS

```
*methods without a return value
*•AMDP procedure implementations with the addition BY DATABASE PROCEDURE
*▪An AMDP method must not be empty.
*▪Writes cannot be performed on database tables where table buffering is switched on, since SC
*▪AMDP methods do not have any implicit enhancement options
*▪DDL statements are not allowed for creating, changing or deleting database objects
*▪no database commits and  database rollbacks using COMMIT and ROLLBACK statements are allowec

*********************************************************************
```

*Global Class | Class-relevant Local Types | Local Types | Test Classes | Macros

Search term

define AMD
and method
value that c
functions.

- AMD
imple
- AMD
imple

Properties × | Problems | Feed Reader | Variables | Debug | Progress | History | Transport Organizer | Bookmarks

TRL_EN

System	System information for application server 0xa0b2fdc_TRL_00:
Resource	
Database:	HDB, Release: 4.00.000.0, Name: H00/00, Server: [redacted], Schema: SAPABAP, Library: [redacted]
Operating System:	Linux, Version: 5.4.0-121-generic
Server:	Machine Type: x86_64, SAP System ID: [redacted], IP Address: [redacted], Node Name: [redacted], Unicode System: True
Kernel:	Release: 789, Patch Level: 12, Compilation Date: Linux GNU SLES-15 x86_64 cc10.3.0 use-pr220812 Aug 12 2022 22:36:23, Kind: opt

Database data

Database System	HDB	Release	[redacted]
Name	[redacted]	Host	[redacted]
Schema	SAPABAP1	User	[redacted]

FM DB_DBSCHEMA_CURRENT will return ABAP DB SCHEMA

SAP TECHNOMANIAC

- The option CDS SESSION CLIENT is mainly required if an AMDP method accesses
- CDS-managed DDIC view of a CDS DDIC-based view, whose client handling is determined by the annotation @ClientHandling.algorithm: #SESSION_VARIABLE

- 1.We can point schema tables like “SCHEMANAME.VBRP” or SCHEMANAME.VBRP.

- 2.If you want to access current schema tables then you can add tables in ‘USING’ clause.

User Defined Function(UDF)

Scalar UDF

- A scalar UDF can be called in SQL statements in the same parameter positions as table column names.
  - These occur in the SELECT and WHERE clauses of SQL statements.
  - For example, SELECT myScalarUDF(1) AS myColumn FROM DUMMY
- Must return a table
- Input Primitive SQL type and Table types

Table UDF

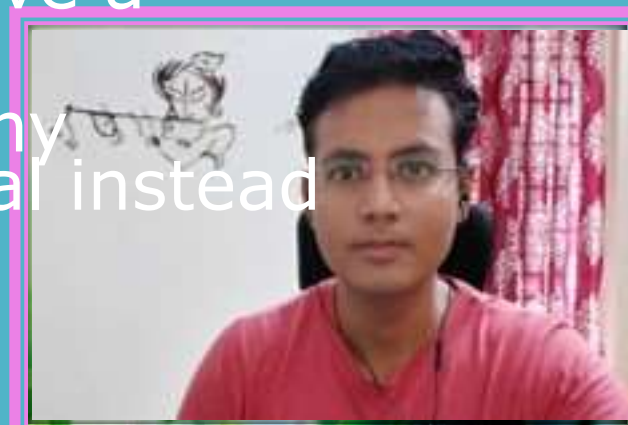
- A table UDF can only be called in the FROM –clause of an SQL statement in the same parameter positions as table names. No Mandatory return value
  - SELECT * FROM myTableUDF(1)
- Must return scalar values
- Input Primitive SQLs

AMDP - Procedure Implementations

- An AMDP Method => database procedure.
- It can be static or instance method
- It can be declare in any visibility section
- In declaration part we can’t say it is AMDP method or not
- The database objects of the current database schema accessed in the AMDP method must be declared using an addition USING.

AMDP - Procedure Implementations

- parameter interface of an AMDP procedure implementation:
- The typing of the parameters must not be generic.
  - Only elementary data types and table types with a structured row type can be used.
  - The row type of a tabular type can only contain elementary data types as components
- The parameters must be declared using VALUE for pass by value
- Return values cannot be declared using RETURNING.
- Only input parameters can be flagged as optional parameters.
- Each elementary optional input parameter must have a replacement parameter declared using DEFAULT.
- An optional tabular input parameter cannot have any replacement parameters and must be made optional instead using OPTIONAL.



AMDP procedure and Function Access

Parameter interface exchange with SQL SCRIPT

SAP TECHNOMANIAC

AMDP procedure and Function Access

- Calls from ABAP
  - AMDP procedure or AMDP scalar function .
    - call meth(...)
  - AMDP table function by specifying the assigned CDS table function
    - ABAP SQL read statement

AMDP procedure and Function Access

- **Calls from other AMDP procedures or functions**
  - Called AMDP procedure implementation
    - CALL "CLASS=>METH"(f1 => a1, f2 => a2, ...);

- Call AMDP function implemented
  - SELECT ...
    FROM "CLASS=>METH"(f1 => a1, f2 => a2, ...);

AMDP procedure and Function Access

- Calls from regular database procedures:-
  - An SQLScript procedure or function created on database can call AMDP but is is **no recommended** by sap since it is manged by ABAP

- Access In SAP Web IDE for SAP HANA :-
  - AMDP's are visible in SAP Web IDE for SAP HANA and can even be edited. This is **not recommended** since this kind of change has no effect on the implementation in the AMDP method and can be overwritten by the ABAP runtime environment at any time.

Syntax of AMDP

- Syntax of AMDP's = The syntax of a SQL Script procedure or function written in SQL Script

- The character * at the start of a line indicates a comment line, as in ABAP. When the procedure or function is saved in the database system, the asterisk, *, is transformed to the usual double hyphens, --

The parameter interface

- The parameter interface of an SQLScript procedure supports
  - IN,
  - OUT,
  - INOUT(It can be only scalar)- SQL script not support INOUT tabular parameter

- AMDP -> SQL script Conversion parameter interface
  - IMPORTING => IN
  - EXPORTING => OUT
  - CHANGING (SCALAR) => INOUT
  - CHANGING (Tabular) => IN and OUT two parameter
    - OUT = Changing parameter name
    - IN = Changing parameter + _IN_ name

Handle SELECT-OPTIONS in AMDP

SAP TECHNOMANIAC

Handle SELECT-OPTIONS in AMDP

```
SELECT  a~vbeln AS vbeln,
        b~posnr AS posnr,
        b~matnr AS matnr,
        d~maktx AS maktx,
        a~kunnr AS kunnr,
        c~name1 AS name1
  FROM vbak AS a
  INNER JOIN vbap AS b
    ON a~vbeln = b~vbeln
  LEFT OUTER JOIN kna1 AS c
    ON a~kunnr = c~kunnr
  LEFT OUTER JOIN makt AS d
    ON  b~matnr = d~matnr
    AND d~spras  = 'E'
 WHERE a~vbeln IN @s_vbeln
 AND b~posnr IN @s_posnr
 AND a~kunnr IN @s_kunnr
 AND b~matnr IN @s_matnr
 INTO @DATA(it_out).
 IF sy-subrc IS INITIAL.

 ENDIF.
```

```
METHOD get_data_ddic_cds BY DATABASE PROCEDURE
                            FOR HDB
                            LANGUAGE SQLSCRIPT
                            USING zddls_sample_02
                            .
  et_vbak = SELECT so,
                   sales__org,
                   vkgrp,
                   num_lit,
                   char_lit,
                   syst_date
              FROM ZDDLS_SAMPLE_02( session_context('SAP_SYSTEM_DATE') )
                WHERE mandt = :iv_mandt
                and  so in ( select low from :it_vbeln ) ;
ENDMETHOD.
```

SAP TECHNOMANIAC

Handle SELECT-OPTIONS in AMDP

- Conversion of the selection tables into an SQL WHERE clause using method CL_SHDB_SELTAB=>COMBINE_SELTABS( )

- Handling of dynamic WHERE clauses within the AMDP method using the function APPLY_FILTER

- The class CL_LIB_SELTAB and its methods are obsolete

- AMDP Manages
  1. **AMDP procedures**
  2. AMDP functions
     - AMDP table functions
       - AMDP table functions for AMDP methods
       - AMDP Table Functions for CDS Table Functions
     - AMDP scalar function

AMDP table functions for AMDP methods

- Functions that can only be accessed in other AMDP methods
- Must have return value as table using RETURNING
- No changing or output parameters.
- Read Only.

AMDP Scalar Functions

- Return value is elementary.
- Input parameters must also be elementary.
- An AMDP scalar function can be called in ABAP like a regular method and can be used as a functional method in a functional method call.
- Elementary data type :- Data type of fixed or variable length that is neither structured, nor a table type or a reference type.

  In particular, the built-in ABAP types are elementary.

The built-in ABAP types are:
b, c, d, decfloat16, decfloat34, f, i, int8, n, p, s, string,
t, utclong, x, and xstring.

ABAP CDS - Access Control

- CDS Data control language (CDS DCL)
- It will further restrict the data from CDS entity
- A CDS role is not assigned to individual users and is evaluated for every user instead.

```
define role YDCL_SAMPLE_02 {
    grant
        select
            on
                YDDLS_SAMPLE_02
                    where
                        sales_org = '1000';
}
```

ABAP CDS - Access Control

➢Access rules :-

Access rules can define access conditions, but also provide full access.

```
define role YDCL_SAMPLE_02 {
    grant
        select
            on
                YDDLS_SAMPLE_02
                    where
                        sales_org = '1000';
}
```

```
define role demo_cds_role_fullaccess {
  grant select on demo_cds_auth_fullaccess; }
```

ABAP CDS - Access Control

➢Access conditions:-

Access conditions are based primarily on

- Literal values
- On classic authorizations of the current users
- On data from other CDS entities defined by a selection with the current user (self-defined aspects).

```
define role YDCL_SAMPLE_02 {
    grant
        select
            on
                YDDLS_SAMPLE_02
                    where
                        sales_org = '1000';
}
```

```
define role C_OPENSALESORDERS_F2200 {
  grant select on C_OPENSALESORDERS_F2200
  where (SalesOrderType) =
  aspect pfcg_auth (V_VBAK_AAT,
            auart,
            actvt = '03')
```

Access control Annotation

- The access conditions are evaluated implicitly in each ABAP SQL read

- Based on @AccessControl.authorizationCheck
  - CHECK (Default value)
  - NOT_REQUIRED
  - NOT_ALLOWED

- If access control is enabled, only that data is read that meets the access conditions.

Access control can be disabled in the following ways:

- #NOT_ALLOWED
- WITH PRIVILEGED ACCESS in ABAP SQL query.
- By creating a full access rule for the entity in a customer CDS role.
- A CDS entity can also be used as a data source in another CDS entity for which access control is disabled.

Important Points

- Access control will not apply When a CDS entity is used as a data source in another CDS entity

- When CDS entities are accessed using ABAP SQL, ABAP programs cannot distinguish whether data is not read because it does not exist or because they are not allowed by CDS access control.

ABAP CDS - Access Control

- What is access rule in CDS access control?
- Different types of **Access Rules** in CDS access control?
- How to create authorization object in **SU21?**
- Creation of role in **PFCG** T-code.
- Assign Role to a user.
- Use classical authorization object in CDS Access control

SAP TECHNOMANIAC

ABAP CDS - Access Control

➢Access rules :-

Access rules can define access conditions, but also provide full access.

```
define role YDCL_SAMPLE_02 {
    grant
        select
            on
                YDDLS_SAMPLE_02
                    where
                        sales_org = '1000';
}
```

CDS DCL - Access Rules

- CONDITIONAL_RULE :- which control access using access conditions
- GRANT_RULE :-Grant unrestricted access
- INHERITED_RULE :- Applied from existing CDS roles (It can occur only once in CDS ROLE)

.

CDS DCL - Access Rules

- The access rules defined by different CDS roles for a CDS entity are joined by a logical "or".

- It is advisable to use only one access rule in a CDS role.

```
define role YDCL_SAMPLE_02 {
    grant
        select
            on
                YDDLS_SAMPLE_02
                where
                    (sales_org) = aspect pfcg_auth( YVKORG,YSALES_ORG,ACTVT='03' );
   grant
        select
            on
                YDDLS_SAMPLE_02
                where
                    sales_org = 'NA01';
}
```

CDS DCL - Access Rules (GRANT_RULE)

- GRANT SELECT ON without the addition WHERE
- Partners and customers can use full access rules to override roles supplied by SAP.

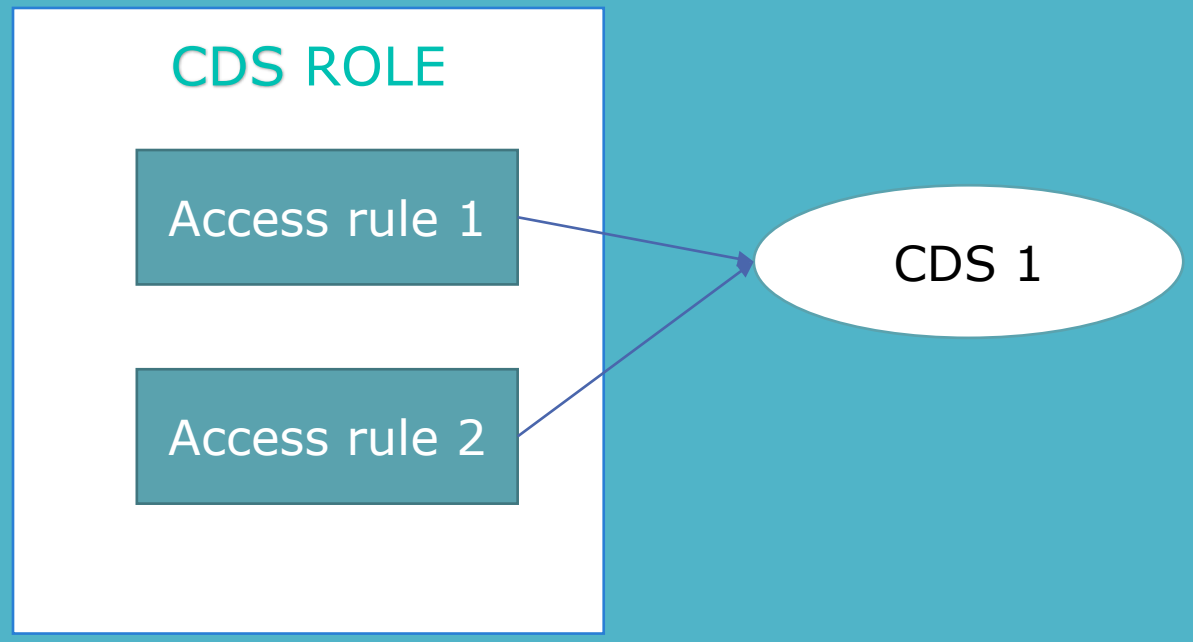
```
@EndUserText.label: 'Access control sample 01'
@MappingRole: true
define role ZDCL_SAMPLE_01 {
    grant
        select
            on
                zcds_sample_01;
}
```

CDS DCL - Access Rules (CONDITIONAL_RULE)

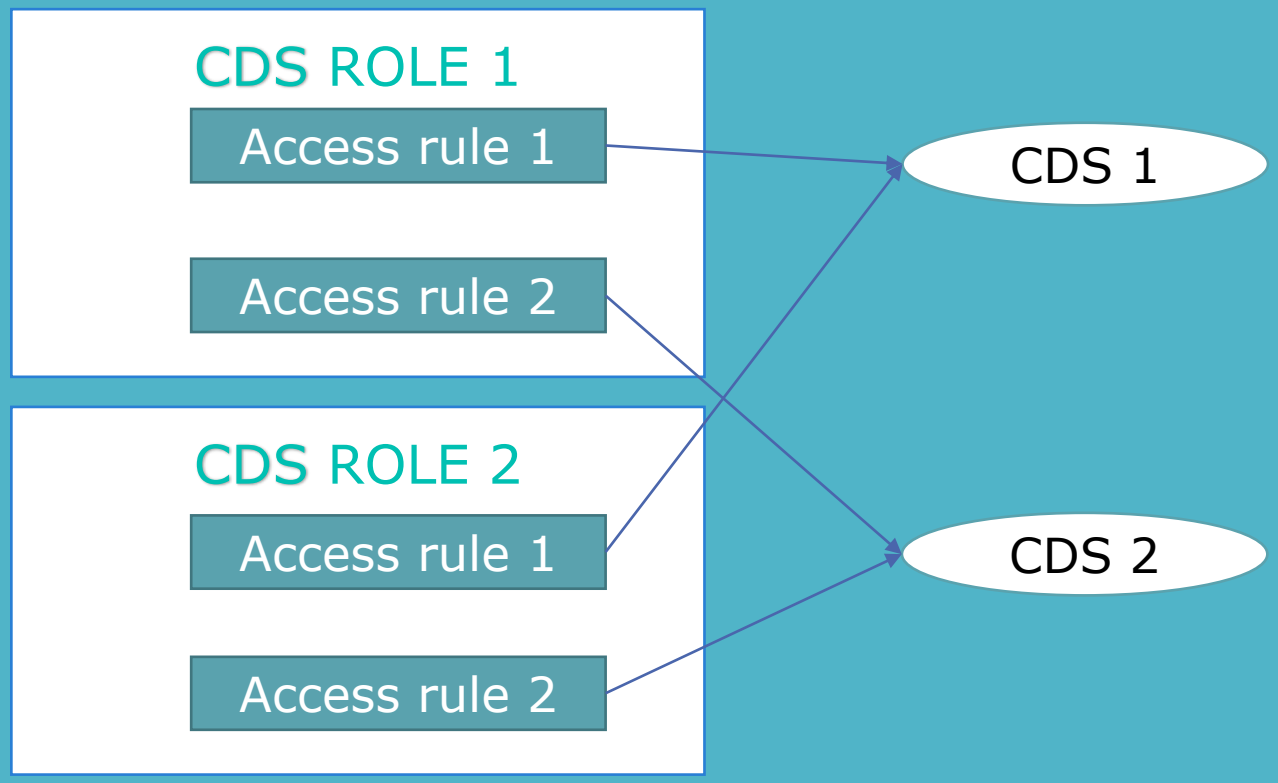
- With the addition WHERE restricts access to a CDS entity using access conditions.

```
@EndUserText.label: 'Access control sample 01
@MappingRole: true
define role ZDCL_SAMPLE_01 {
    grant
        select
            on
                zcds_sample_01
                  where vkorg = 'VVSO';
}
```

A single CDS entity can be specified in multiple access rules of a CDS role




```
define role YDCL_SAMPLE_02 {
    grant
        select
            on
                YDDLS_SAMPLE_02
                where
                    (sales_org) = aspect pfcg_auth( YVKORG,YSALES_ORG,ACTVT='03' );
   grant
        select
            on
                YDDLS_SAMPLE_02
                where
                    sales_org = 'NA01';
}
```



Multiple CDS roles can contain access rules for a single CDS entity.

```
define role YDCL_SAMPLE_02 {
    grant
        select
            on
                YDDLS_SAMPLE_02
                where
                    (sales_org) = aspect pfcg_auth( YVKORG,YSALES_ORG,ACTVT='03' );
   grant
        select
            on
                YDDLS_SAMPLE_02
                where
                    sales_org = 'NA01';
}
```

```
@EndUserText.label: 'YDCL_sample_02_1'
@MappingRole: true
define role YDCL_SAMPLE_02_1 {
    grant
        select
            on
                YDDLS_SAMPLE_02
                where
                    vbtyp = 'C';
}
```

CDS DCL - Access Rules (CONDITIONAL_RULE)

- COMBINATION MODE AND|OR (Optional) used for multiple access rules for same CDS entity.

```
define role YDCL_SAMPLE_02 {
    grant
        select
            on
                YDDLS_SAMPLE_02
                where
                    (sales_org) = aspect pfcg_auth( YVKORG,YSALES_ORG,ACTVT='03' );
   grant
        select
            on
                YDDLS_SAMPLE_02
                combination mode and  // OR
                where
                    sales_org = 'NA01';
}
```

CDS DCL - Access Rules (CONDITIONAL_RULE)

```
@EndUserText.label: 'DCL for SAMPLE_02'
@MappingRole: true
define role YDCL_SAMPLE_02 {
    grant
        select
            on
                YDDLS_SAMPLE_02
                where
                    (sales_org) =
      aspect pfcg_auth( YVKORG,YSALES_ORG,ACTVT='03' );

   grant
        select
            on
                YDDLS_SAMPLE_02
                combination mode and  // OR
                where
                    sales_org = 'NA01';
}
```

REDEFINITION

```
@EndUserText.label: 'YDCL_sample_02_1'
@MappingRole: true
define role YDCL_SAMPLE_02_1 {
    grant
        select
            on
                YDDLS_SAMPLE_02
                redefinition
                where
                    vbtyp = 'C';
}
```

- The addition can be used for a maximum of one access rule for a CDS entity.
- This addition also disables existing full access rules for a CDS entity.

CDS DCL - Access Rules (INHERITED_RULE)

- A CDS role can only contain one inherited access rule
- The existing CDS role parent_role can only contain a single access rule for exactly one CDS entity, which itself can be an inherited access rule.

```
@MappingRole: true
define role demo_cds_role_lit_pfcg {
  grant select on demo_cds_auth_lit_pfcg
  where (carrid) =
  aspect pfcg_auth (s_carrid, carrid, actvt='03') and
         currcode = 'EUR'; }
```

```
@MappingRole: true
define role demo_cds_role_inh_obs {
  grant select on demo_cds_auth_inh_obs
                inherit demo_cds_role_lit_pfcg or currcode = 'USD'; }
```

ABAP CDS - Access Control

- What is access rule in CDS access control?
- Different types of **Access Rules** in CDS access control?
- How to create authorization object in **SU21?**
- Creation of role in **PFCG** T-code.
- Assign Role to a user.
- Use classical authorization object in CDS Access control

SAP TECHNOMANIAC

ABAP CDS - Access Control

- Different kind of access condition in access control DCL
- Creation of auth object with multiple field multiple values
- Different ways to apply PFCG_CONDITION

CDS DCL - DEFINE ROLE, condition

```
@EndUserText.label: 'Access control sample 01'
@MappingRole: true
define role ZDCL_SAMPLE_01 {
    grant
        select
            on
                zcds_sample_01
                  where (vkorg) = aspect pfcg_auth( zvkorg,ZVKORG,ACTVT='03');
   grant
        select
            on
                zcds_sample_01
                combination mode or
                  where vkorg = '1710';
}
```

SAP TECHNOMANIAC

CDS DCL - DEFINE ROLE, condition

- PFCG_CONDITION
- LITERAL_CONDITION
- ASPECT_CONDITION
- USER_CONDITION
- INHERIT_CONDITION
- TRUE
- FALSE
- VOID

CDS DCL - PFCG_CONDITION

- PFCG_AUTH is a predefined aspect
- Aspect :-Object used for the definition of user-specific values and their use in the conditions of a CDS role.

```
@EndUserText.label: 'Access control sample 01'
@MappingRole: true
define role ZDCL_SAMPLE_01 {
    grant
        select
            on
                zcds_sample_01
                  where (vkorg) = aspect pfcg_auth( zvkorg,ZVKORG,ACTVT='03');
}
```

CDS DCL - PFCG_CONDITION

- The left side consists of a parenthesized comma-separated list consisting of zero, one, or multiple CDS elements

```
WHERE
   ( SALES_ORG,VBTYP )                                  ?= ASPECT PFCG_AUTH( YTWOFIELD,YSALES_ORG,VBTYP,ACTVT='03')
OR ( SALES_ORG  )                                        = ASPECT PFCG_AUTH( YTWOFIELD,YSALES_ORG,VBTYP='C',ACTVT='03')
OR ( )                                                   = ASPECT PFCG_AUTH( YTWOFIELD,YSALES_ORG='1070',VBTYP='C',ACTVT='03')
OR NOT ( )                                               = ASPECT PFCG_AUTH( YTWOFIELD,YSALES_ORG='1710',VBTYP='C',ACTVT='03')
OR ( SALES_ORG BYPASS WHEN IS INITIAL OR NULL )          = ASPECT PFCG_AUTH( YTWOFIELD,YSALES_ORG,VBTYP='C',ACTVT='03');
```

- ELEMENT, MAPPING_FIELD , AUTH_FILEDS
- The operator **NOT** can only be specified in front of PFCG conditions with **empty parentheses**

CDS DCL - PFCG_CONDITION

```
WHERE
   ( SALES_ORG,VBTYP )                                  ?= ASPECT PFCG_AUTH( YTWOFIELD,YSALES_ORG,VBTYP,ACTVT='03')
OR ( SALES_ORG  )                                        = ASPECT PFCG_AUTH( YTWOFIELD,YSALES_ORG,VBTYP='C',ACTVT='03')
OR ( )                                                   = ASPECT PFCG_AUTH( YTWOFIELD,YSALES_ORG='1070',VBTYP='C',ACTVT='03'
OR NOT ( )                                               = ASPECT PFCG_AUTH( YTWOFIELD,YSALES_ORG='1710',VBTYP='C',ACTVT='03'
OR ( SALES_ORG BYPASS WHEN IS INITIAL OR NULL )          = ASPECT PFCG_AUTH( YTWOFIELD,YSALES_ORG,VBTYP='C',ACTVT='03');
```

- It is advisable to specify an element of the CDS entity directly and to only use **path expressions** in exceptional cases
- PFCG conditions can be **combined** within an access rule using literal conditions, user conditions, and inheritance conditions.
- The operator ?= is applied to all CDS elements

in the left parentheses. It cannot be restricted to individual elements. With BYPASS WHEN, a better alternative exists

CDS DCL - PFCG_CONDITION

➢BYPASS WHEN :- IS NULL , IS INITIAL, IS INITIAL OR NULL

```
WHERE
   ( SALES_ORG bypass when is null,
     VBTYP     bypass when is initial )
    = ASPECT PFCG_AUTH( YTWOFIELD,YSALES_ORG,VBTYP,ACTVT='03')
```

SALES_ORG	VBTYP value	Filtering Result
1710	C	OK
NULL	C	OK (by bypassing for field1)
1710	INITIAL	OK (by bypassing for field2)
NULL	INITIAL	OK (by bypassing for field1 and field2)
Na01	INITIAL	Blocked
NULL	B	Blocked
Na01	B	Blocked
INITIAL	NULL	Blocked (NULL and INITIAL are distinguished

CDS DCL - DEFINE ROLE, condition

- PFCG_CONDITION
- LITERAL_CONDITION
- ASPECT_CONDITION
- USER_CONDITION
- INHERIT_CONDITION
- TRUE
- FALSE
- VOID

CDS DCL - INHERIT_CONDITION

- INHERIT FOR GRANT
- INHERITING CONDITIONS FROM ENTITY
- INHERITING CONDITIONS FROM SUPER

CDS DCL - INHERIT_CONDITION

- INHERIT FOR GRANT :-

  Applies the access conditions from a different CDS role.

```
define view entity YDDLS_SAMPLE_05 as select from YDDLS_SAMPLE_04 {
    key sales_order,
    kunnr,
    name1,
    posnr,
    matnr,
    /* Associations */
    _doc_flow[1:vbtyp_n = 'M'].vbeln as invoice,
    _matdesc as material_desc
}
```

```
define role YDCL_04 {
    grant
        select
            on
                YDDLS_SAMPLE_04
                where
                    kunnr = '0017100001';
}
```

```
define role YDCL_05 {
    grant
        select
            on
                YDDLS_SAMPLE_05
                where
                    inherit YDCL_04
                    for grant select on YDDLS_SAMPLE_04
                    and posnr = '000020';
}
```

Parent Role

CDS DCL - INHERIT_CONDITION

- INHERIT FOR GRANT :-
  - parent_role can have multiple access rules for the same CDS entity cds_entity, they joined by a logical "or".
  - Full access rules cannot be inherited.

```
define role YDCL_05 {
    grant
        select
            on
                YDDLS_SAMPLE_05
                where
                    inherit YDCL_04
                    for grant select on YDDLS_SAMPLE_04
                    and posnr = '000020';
}
```

Parent Role

CDS DCL - INHERIT_CONDITION

- INHERITING CONDITIONS FROM ENTITY:-
  Applies the access conditions from a CDS entity

```
define view entity YDDLS_SMAPLE_01
   as select from vbak {
    key vbeln,
        kunnr
}
```

```
define role YDCL_04 {
    grant
        select
            on
                YDDLS_SAMPLE_04
                    where
                        kunnr = '0017100001';
}
```

```
define role YDCL_01 {
    grant
        select
            on
                YDDLS_SMAPLE_01
                    where
                        inheriting conditions from entity YDDLS_SAMPLE_04;
}
```

CDS from access condition derived

CDS DCL - INHERIT_CONDITION

- If the CDS entity does not have any access conditions yet, the addition DEFAULT must be specified.
- If wrong field or path is specified, all access rules of the parent CDS role are ignored.
  - No full access rule CDS role cannot be activated.
  - Full access rule CDS role can be activated with syntax warning .
- This variant also allows the inheritance of a full access rule

```
define role YDCL_01 {
    grant
 select
     on
         YDDLS_SMAPLE_01
             where
                 inheriting conditions from entity YDDLS_SAMPLE_04 default true;
}
```

CDS DCL - INHERIT_CONDITION

➢ INHERITING CONDITIONS FROM SUPER :-

Applies the access conditions from roles that are redefined by the current role.

- This variant is possible only if the access rule has the addition REDEFINITION
- only if the inherited CDS entity has access controls.

```
define role YDCL_SAMPLE_02_1 {
grant
select
    on
     YDDLS_SAMPLE_02
     redefinition
    WHERE
     inheriting conditions from super;
}
```

```
define role YDCL_SAMPLE_02 {
    grant
select
    on
        YDDLS_SAMPLE_02
        where
            (sales_org) = aspect pfcg_auth( YVKORG,YSALES_ORG,ACTVT='03' );

}
```

CDS DCL - INHERIT_CONDITION

- Multiple inheritance conditions can be specified within a single access condition, and these can be combined with literal conditions, PFCG conditions, and user conditions.

- The inherited access conditions are parenthesized implicitly. It is not necessary to set parentheses explicitly.

- An inheritance conditions cannot be negated using NOT.

CDS DCL - DEFINE ROLE, condition

- PFCG_CONDITION
- LITERAL_CONDITION
- ASPECT_CONDITION
- USER_CONDITION
- INHERIT_CONDITION
- TRUE
- FALSE
- VOID

CDS DCL - LITERAL_CONDITION

- A and B are literal values here:-
- CDS Element =,<>,>,<,<=,>=,?= Literal value
- CDS Element BETWEEN A AND B
- CDS Element NOT BETWEEN A AND B
- CDS Element LIKE A%b_C
- CDS Element NOT LIKE A%b_C
- CDS Element NOT NULL
- CDS Element NOT INITIAL

CDS DCL - LITERAL_CONDITION

```
define role YDCL_01 {
    grant
 select
     on
         YDDLS_SMAPLE_01
           where
            vbeln between '0000000001' and '0000000011'
            or
            vbeln like '000000053_'
            or
            vbeln like '000000053%'
```

CDS DCL - USER_CONDITION

- element =|<>|?= ASPECT user

```
where
  ernam = aspect user
```

- User conditions can be combined within an access rule using **literal conditions** and **PFCG conditions**, and **inheritance conditions**.
- Acts like a comparison with the session variable **$session.user** in the CDS DDL.
- It is **not** currently possible to use **session variables** on the **right side** of conditions in DCL.

CDS DCL - ASPECT_CONDITION

- Left-side host expressions are not supported in this language element.
- The only comparison operator allowed is the equality operator =.

```
define accesspolicy YDCL_01_ACCESSPOLICY_ASPECT {
    @EndUserText.label: 'YDCL_01_accesspolicy_aspect'
    define aspect YDCL_01_ACCESSPOLICY_ASPECT as
        select from YDDLS_SMAPLE_01
             with user element ernam
            {

              sales_org
            }

}
```

```
define role YDCL_SAMPLE_02_1 {
grant
select
    on
     YDDLS_SAMPLE_02
    WHERE
     (sales_org) = aspect YDCL_01_ACCESSPOLICY_ASPECT;
```

CDS DCL – TRUE,FALSE and Void

- TRUE and FALSE :-
  - These conditions are either always met or not met.
  - They are usually not needed in a role definition but can be created implicitly in the inheritance of conditions.
- A condition with the value VOID is handled as nonexistent.
  - VOID conditions are not required in the definition of a role and can be created implicitly in inheritances.
  - The following rules apply in combination with other conditions:
    - X AND VOID = VOID AND X = X
    - X OR VOID = VOID OR X = X
    - NOT VOID = VOID
  - An access rule cannot consist solely of VOID conditions.

```
define role ZDCL_SAMPLE_02 {
    grant
        select
            on
 zcds_sample_01
   where
     true
```

CDS DCL - Left Side Host Expressions

- CDS entity can be replaced by left side :-
  - The actual value of a parameter by $parameters.pname
  - The value of a session variable replaced by $session.vname
  - A literal value
- Left side host expressions are evaluated before the expression is passed to the database

```
define role YDCL_01 {
    grant
 select
     on
         YDDLS_SMAPLE_01
             where
              $parameters.p_date = '20220511'
              and
              $session.user = aspect user
              and
              'user_id'  = aspect user ;
}
```

Fuzzy Search in SAP ABAP

- ABAP specific search options
  - What is Proposal search ?
  - What is full text fuzzy search?

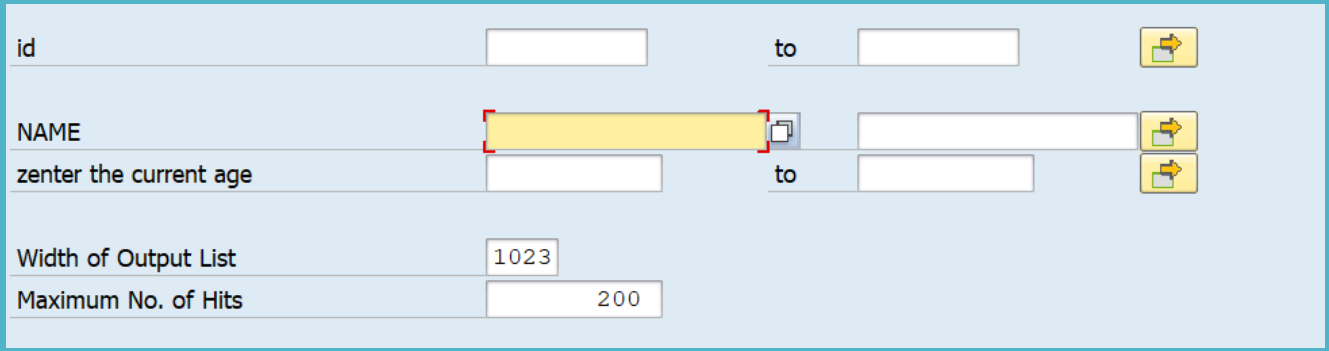
- Full-Text Search with SQL Script when working with AMDP :-
  - Exact search
  - Fuzzy search
  - Linguistic search

- How to use the contains() and score() functions in SQL script?

SAP TECHNOMANIAC

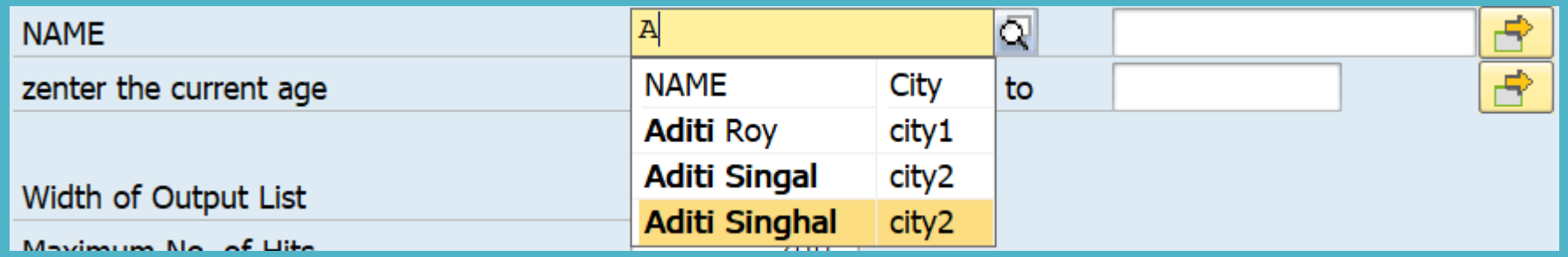
Classical search help

- You must click on F4 to get data or click on search help button icon
- You can use string wild card *



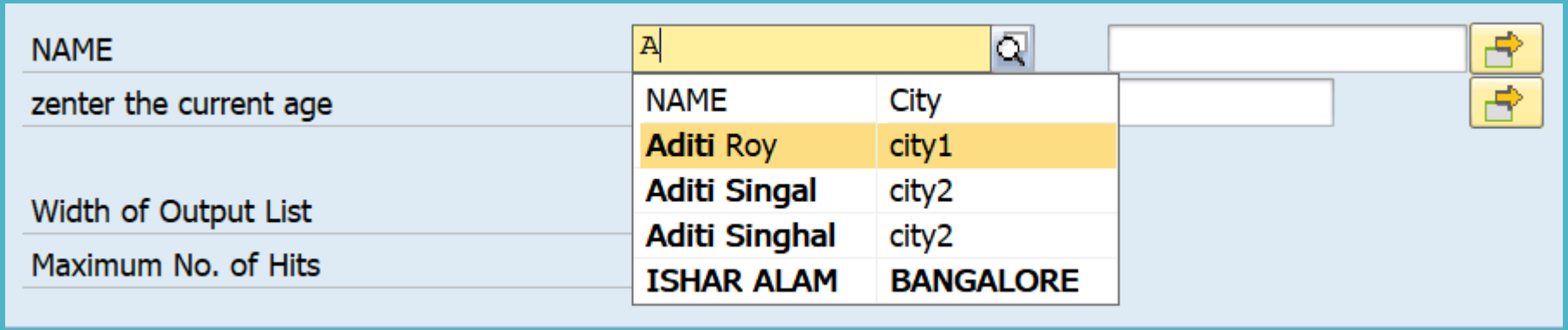
Proposal search(Type ahead)

- ABAP 7.4 SP05 and SAP GUI 7.30 Patch Level 05 or higher
- It is not DB specific
- ABAP systems before 7.4 SP06 Call cl_dsh_dynpro_properties=>enable_type_ahead(...) to enable proposal search explicitly in the screen



Multi-column Full Text Fuzzy search

- This is DB specific
- We can provide accuracy value for error tolerant search
- Multicolumn
- Elementary search help must be based on a column-store table or a CDS view.



User-specific deactivation of the enhanced search

Search:

- Visual Design
  - Theme Settings
  - Font Settings
  - Branding
  - Define Custom Colors
  - Color Settings
    - Colors in System
- Interaction Design
  - Keyboard Settings
  - Visualization 1
  - Visualization 2
  - Notifications
  - Control Settings
  - Sound Settings
- Accessibility & Scripting
- Multilingual Settings
- Local Data
- Traces
- Security
- Front End Print
- Restore & Cleanup

Window Title

☐ Show system name in taskbar button

Ellipsis

☑ Show ellipsis (...) if text cannot be displayed entirely

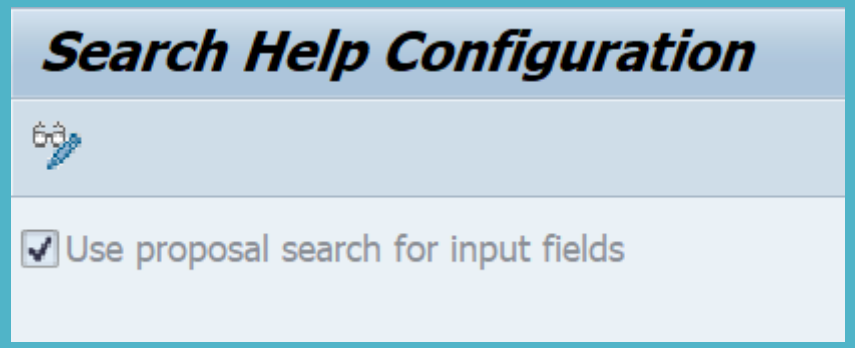
Enhanced Search

◉ Show Enhanced Search automatically
○ Show Enhanced Search on demand using shortcut (Ctrl+Shift+Q)
○ Do not show Enhanced Search

OK | Cancel | Apply | Help

Central deactivation of the enhanced search

- Go to transaction SDSH_CONFIG
- Switch to the Change mode
- Deselect the Use proposal search for input fields option



Practice in the system

Full-Text Search with SQL Script

- The tables you want to search are column-oriented

```
SELECT * FROM <table> WHERE CONTAINS (<co1>, <search_string>,<Search_type> )
```

- Search type
  - EXACT by Default
  - LINGUISTIC
  - FUZZY
- Full-text indexes have been created for the search-relevant columns.
- For column of type TEXT and SHORTTEXT, this is done automatically

Fuzzy search-Full text Index

- We require full text index for :-
  - To get all features of text search in sap hana
  - For Good performance
- To create full text index table must be column store
- Data type should be NVATCHAR(CHAR, STRING or SSTRING in ABAP).
- ABAP release 7.4 SP03 or higher to create full text index
- Each full-text index that you create increases the footprint of the table in memory.
- Index update mode Synchronous, Asynchronous

Full-Text Search with SQL Script

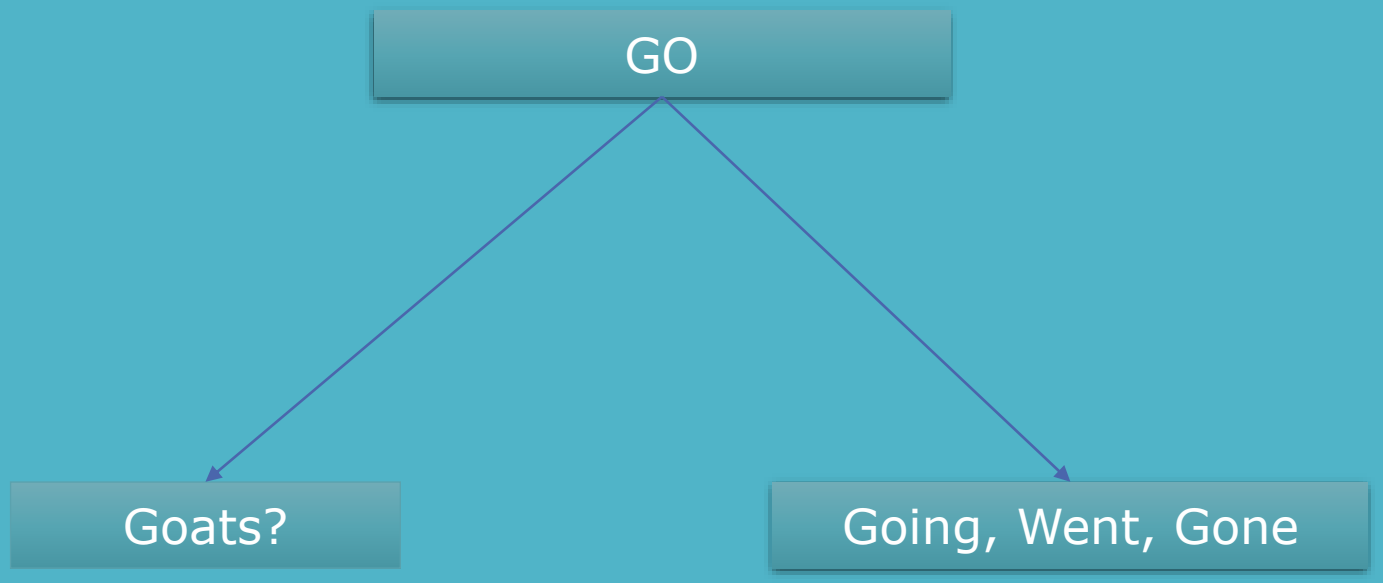
- Wild Card :- * % in search string

```
SELECT * FROM ZEMPLOYEE
      WHERE
      CONTAINS (NAME, 'A' )
      CONTAINS (NAME, 'A' ,EXACT)
      CONTAINS (NAME, 'GO' , LINGUISTIC )
```

Fuzzy Search Usage

- CONTAINS() function in the WHERE clause
- SCORE() is the ranking of each result in the result set
- The fuzzy search (and later the linguistic search) is only available in native SQL.

Linguistic Search



Fuzzy search

Linguistic search

Linguistic Search

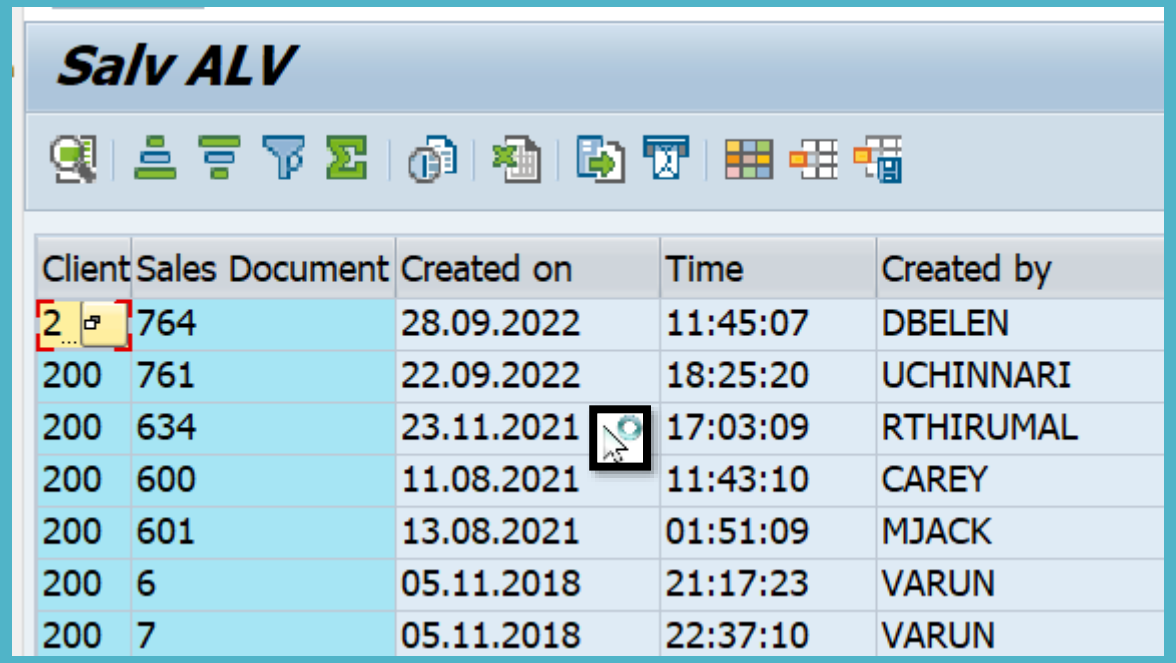
- Level of linguistic analysis
- LINANALYSYS Basic :- String to words
- LINANALYSYS Stems :- String to words and find root word went- > go
- LINANALYSYS Full :- String to words and find root word went- > go and tag theparts of speech as noun, verb and so on

ABAP List Viewer with Integrated Database Access (ALV IDA)

- Why **ALV IDA** and limitation of **classical ALV**?
- Differences between **ALV with IDA** and **classical ALV**?
- Display data with **ALV with IDA & End user Perspective**
- Use **select options** in the data retrieval of ALV with IDA
- Supply values for input parameters of **CDS views**
- Complex Condition & **Checking Authorization** within database

SAP TECHNOMANIAC

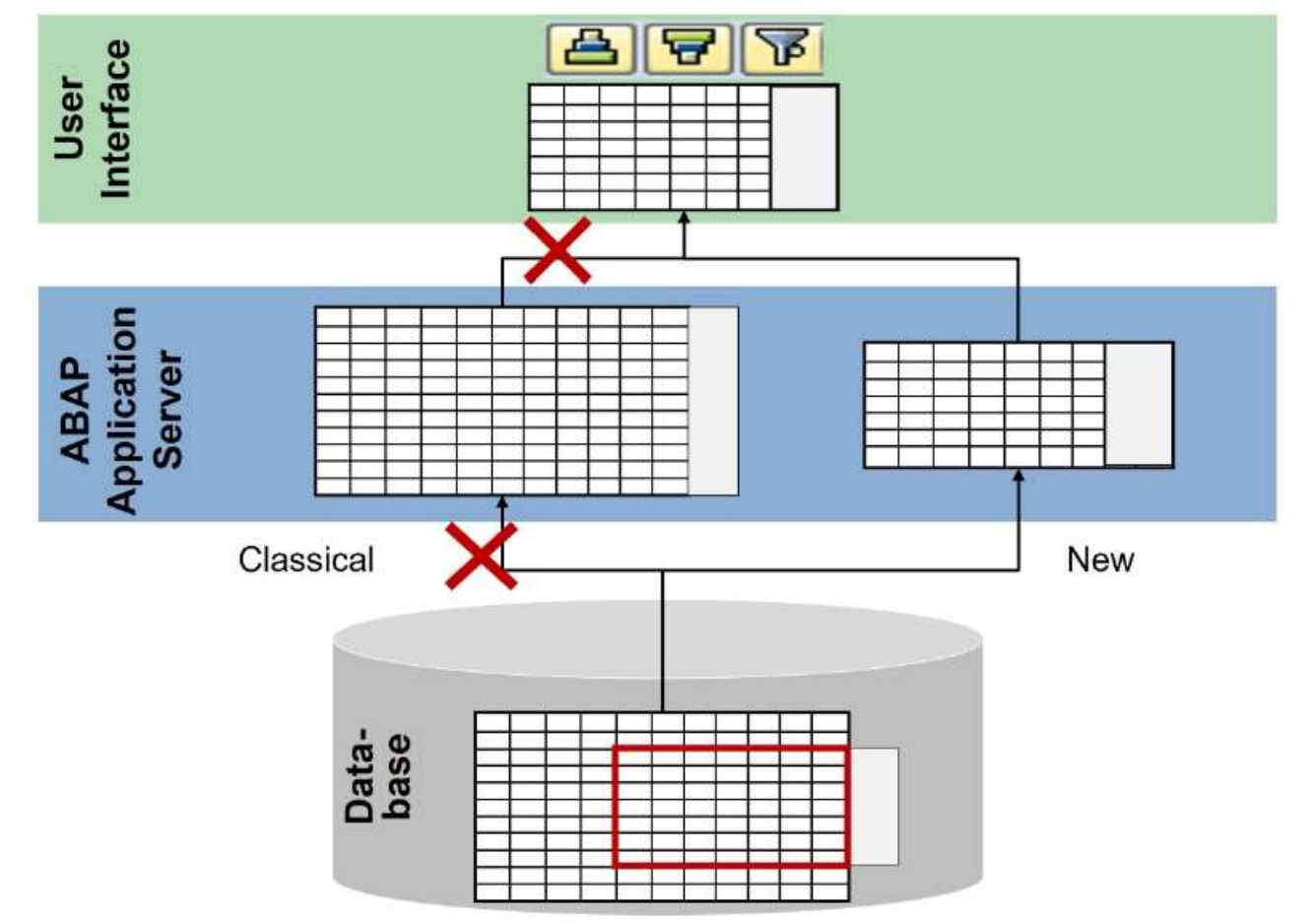
Why ALV IDA and limitation of classical ALV?



Client	Sales Document	Created on	Time	Created by
2	764	28.09.2022	11:45:07	DBELEN
200	761	22.09.2022	18:25:20	UCHINNARI
200	634	23.11.2021	17:03:09	RTHIRUMAL
200	600	11.08.2021	11:43:10	CAREY
200	601	13.08.2021	01:51:09	MJACK
200	6	05.11.2018	21:17:23	VARUN
200	7	05.11.2018	22:37:10	VARUN

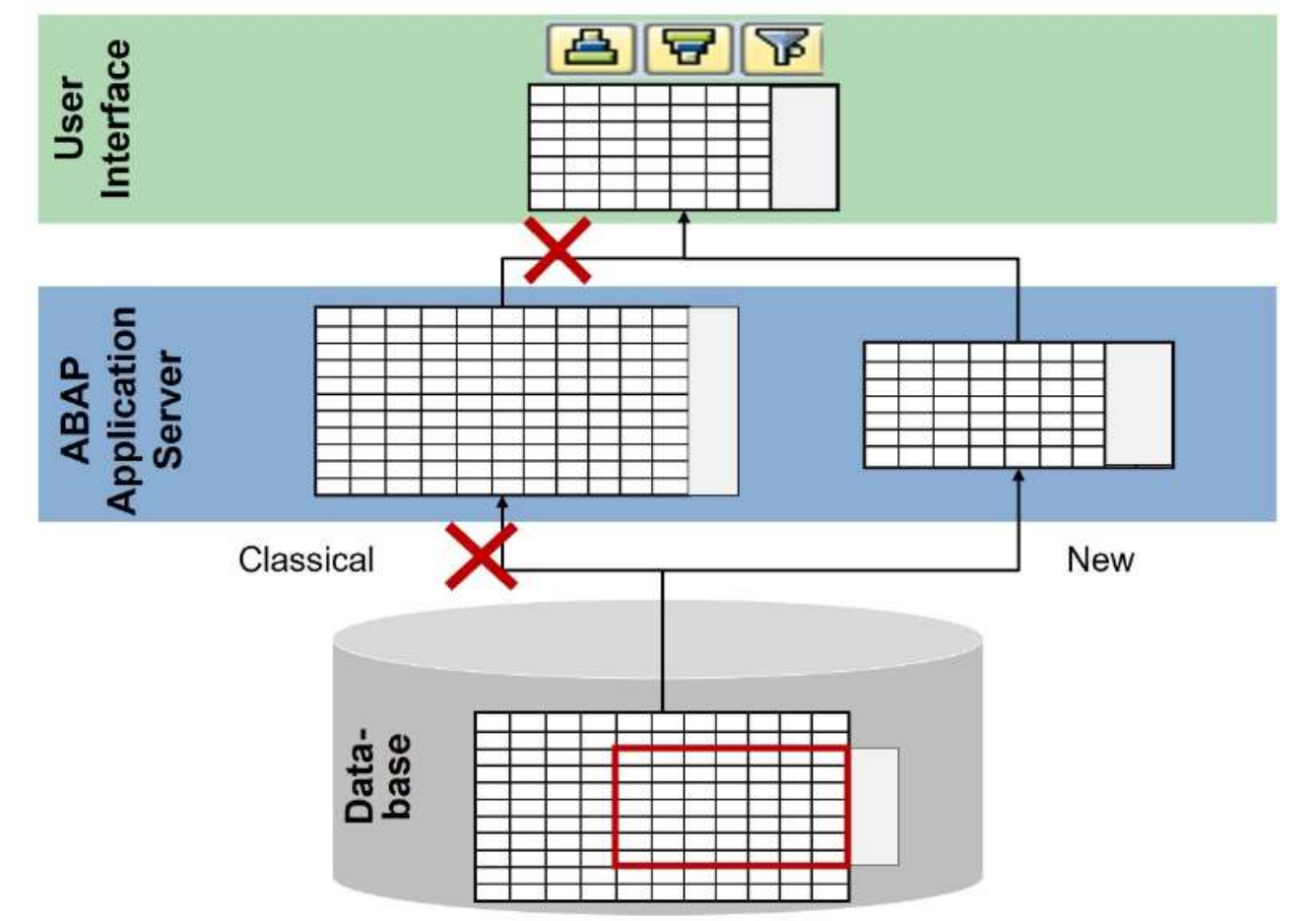
- Larger lists
- Calculated column
- Grouping or sort entries or include aggregates

What is ALV with IDA?



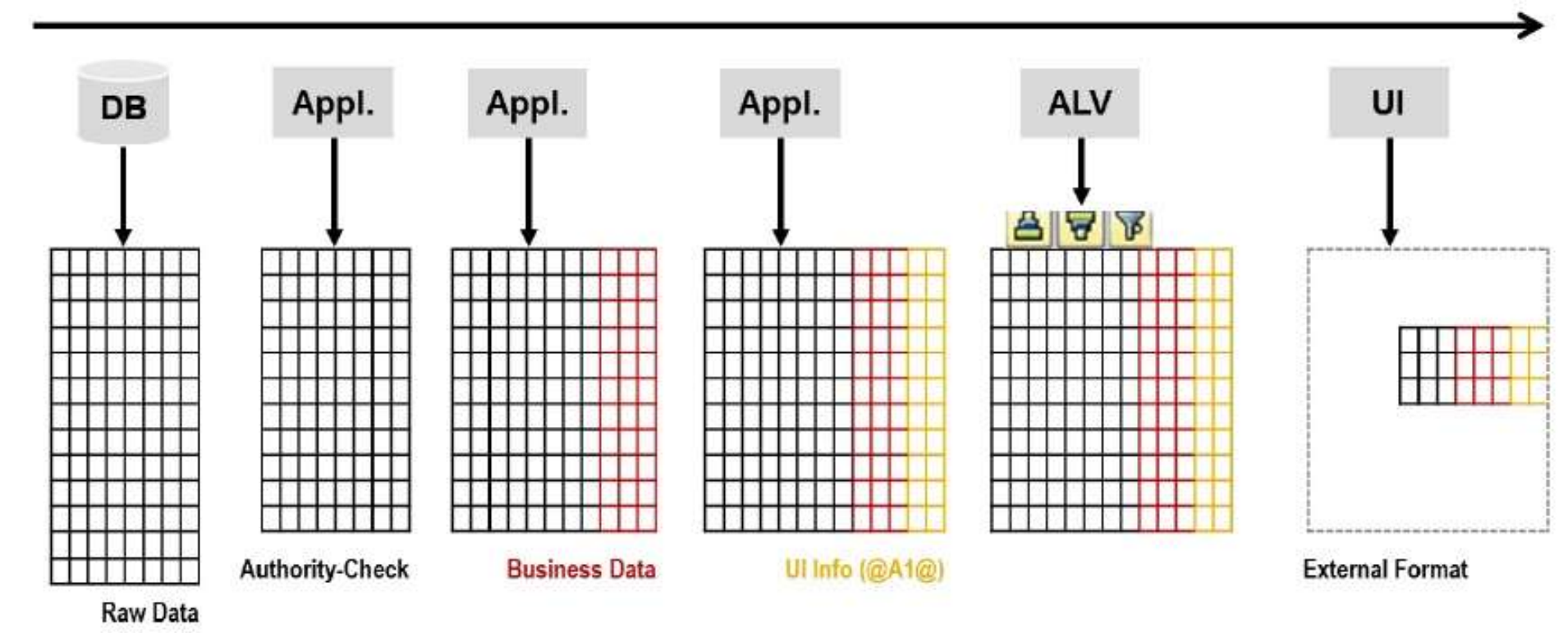
- Basic principles:-
  - Only retrieve, from the database, data which is to be displayed on the screen
  - Use database services where possible – ALV features pushed down to the database
  - Data described declaratively instead of passing big internal tables

What is ALV with IDA?



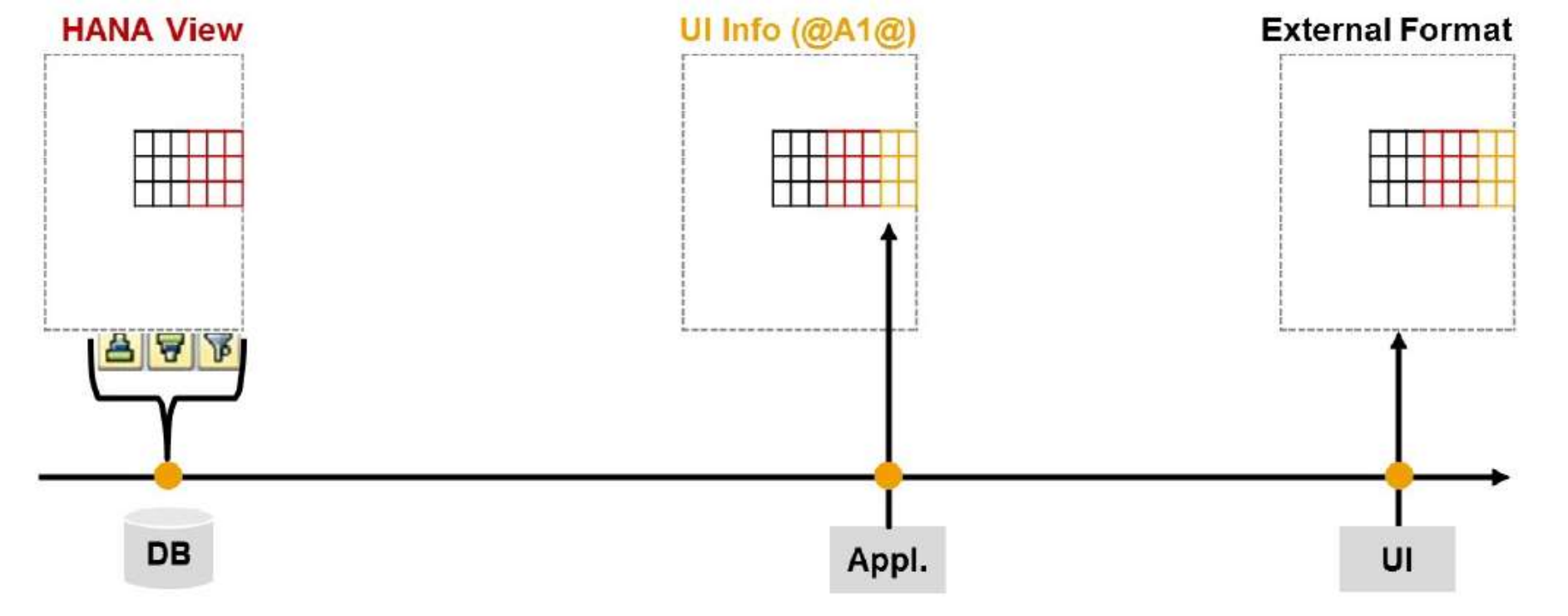
- Advantages:-
  - Retrieval of results is much faster
  - Better performance and reduced memory consumption
  - Improved user experience

Classical ALV works:-

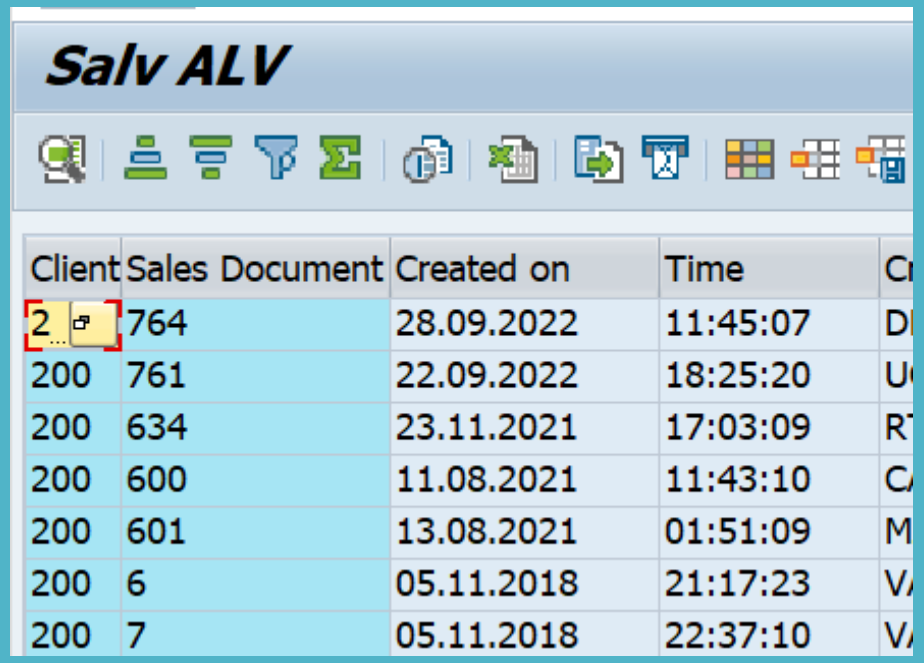


ALV IDA

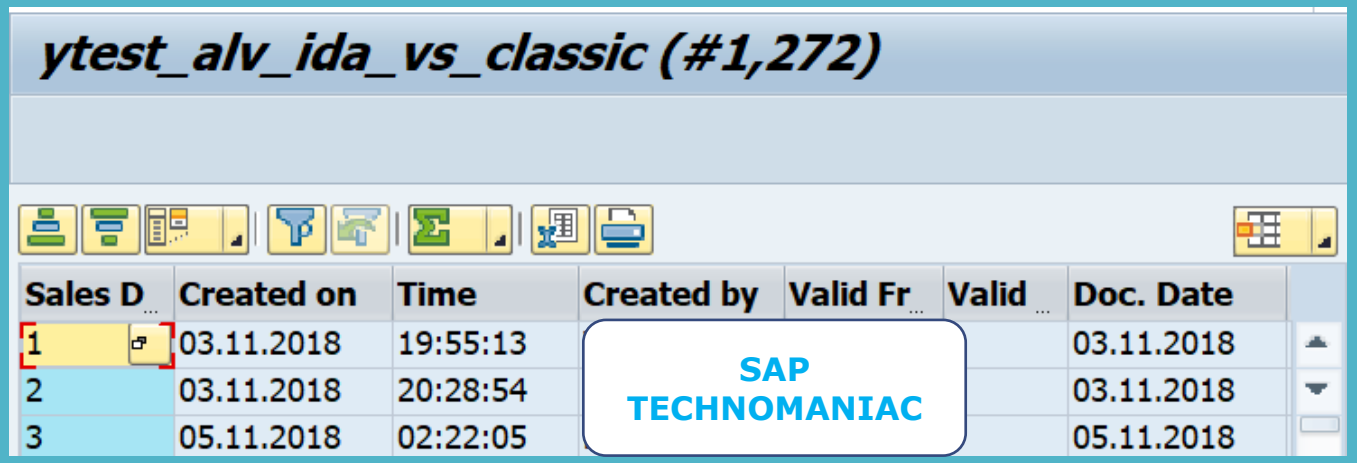
- Auth check and Business data calculation done at DB



End User Perspective



Client	Sales Document	Created on	Time	Cr
2	764	28.09.2022	11:45:07	DI
200	761	22.09.2022	18:25:20	U
200	634	23.11.2021	17:03:09	R
200	600	11.08.2021	11:43:10	C
200	601	13.08.2021	01:51:09	M
200	6	05.11.2018	21:17:23	V
200	7	05.11.2018	22:37:10	V



Sales D	Created on	Time	Created by	Valid Fr	Valid	Doc. Date
1	03.11.2018	19:55:13				03.11.2018
2	03.11.2018	20:28:54				03.11.2018
3	05.11.2018	02:22:05				05.11.2018

SAP TECHNOMANIAC

The ALV with IDA will look and feel familiar to end users who have worked with the classical ALV, but there are some small differences

Instantiation of ALV with IDA

➢Two different factory methods exist

```
cl_salv_gui_table_ida=>create(
  EXPORTING
    iv_table_name          = 'VBAK'
*    io_gui_container       =
*    io_calc_field_handler =
*  RECEIVING
*    ro_alv_gui_table_ida   =
)->fullscreen( )->display( ).
```

```
cl_salv_gui_table_ida=>create_for_cds_view(
  EXPORTING
    iv_cds_view_name       = 'ZDDLS_SAMPL_01'
*    io_gui_container       =
*    io_calc_field_handler =
*  RECEIVING
*    ro_alv_gui_table_ida   =
).
```

- Database tables
- Database views
- External views

- CDS View

SAP TECHNOMANIAC

Display of ALV with IDA

With Container

```
lo_container_d0555 = NEW #( 'D0555_CONTAINER' ).

" Instantiate and display ALV
lo_alv_display = cl_salv_gui_table_ida=>create( iv_table_name     = 'SFLIGHT'
                                                io_gui_container = lo_container_d0555 ).
```

Without Container

```
cl_salv_gui_table_ida=>create_for_cds_view( 'ZDDLS_SAMPL_01')->fullscreen( )->display( ).
```

Performance Considerations

- Setting Selection Options
- Complex Conditions
- Checking Authorizations
- Text Search Across the Whole Table

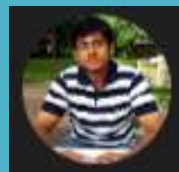
	SAP List Viewer	ALV with IDA
Data Retrieval	Responsibility of application, data is collected in an internal ABAP table (ITAB)	Responsibility of ALV, table name has to be transferred
Data Contents	All data from the ITAB	Visible area only
Roundtrip (e.g. scrolling)	From ALV perspective only operations on the ABAP Server, new area from the ITAB is displayed	Paging on the database, that is, new SQL statement is executed
Application ALV Services (sorting, filtering…)	On the ITAB (snapshot behavior)	New call to the database
Memory Consumption	Depends on the size of the ITAB	Visible area only, relating to columns and rows
Speed	Time required for all data to be transferred to the ITAB on first display	Time required for visible area to be transferred

- Handling of data that is changed during display is not supported. Data records that are removed during the display are automatically filled by empty lines.
- The Data Display is restricted to a maximum of 2 billion cells. All available operations are then executed on the entire data set even if this exceeds the maximum display size.
- Aggregation of amount fields with currencies or quantities: Values for the aggregation can only be displayed if all fields values refer to the same currency or quantity type (e.g. currency euro).
- Table fields of type STRING should not be used.
- The ready for input status for cells is not supported in IDA ALV.
- IDA-ALV programs cannot be scheduled as background jobs or be included in background management.

Open SQL vs ABAP CDS

SAP TECHNOMANIAC

Open SQL vs CDS



@calltobharath • 2 hours ago

Bhai - I have your CDS playlist couple of times now. I had to revisit it a few times to absorb so many nuggets of knowledge that you have generously shared with us.

One question remains - When we can do almost everything in CDS as well as regular open SQL, why do we go for CDS in real world? What do developers like you do on a daily basis - Do you go for CDS or you simply write open SQL query? Do you use OO ALV or do you use HANA ALV? What will be your reason to choose CDS over open SQL?

Open SQL vs ABAP CDS

ABAP CDS and Open SQL are not competitors.
On the contrary, they complete each other.

Open SQL over CDS

- Which all are the things can be achieved using Open SQL in ABAP programs no need to go for CDS

- Some of the things we can’t do using CDS:-
  - If we are doing dynamic programming
  - If we need to use for all entries (to use buffering for example)
  - We have to manipulate the data in database.(update, modify, delete)

CDS over Open SQL

- When we have reusability
- When we need access control on select
- ODATA service creation through annotation
- We are using domain-specific metadata
- Modification-free extensibility
- SQL function is available only in CDS(not in open SQL) and can push down logic to the database.
- When we need to create data model which can be used across technologies (ODATA, RAP, BOPF)

Open SQL vs CDS

- SQL function is available only in CDS(not in open SQL) and can push down logic to the database.
- When we have reusability
- When we need to create data model which can be used across technologies
- When we need access control on select
- We need to create ODATA service
- We are using domain-specific metadata

When to use CDS table Function?

- To access **cross schema** tables in CDS views, so we are using table functions to do that.

- CDS Table functions are used for instance in case you want to integrate some **HANA features**, not available directly via the ABAP SQL or CDS layer, into your VDM.

- Table functions that are implemented natively on the database we can call these directly in CDS table function

- If we have to encapsulate complex logic that would require > 2 CDS views to achieve. Ex :- ABAP CDS View: join tables on columns of different type

THANK YOU

CONTACT

Ram Niwas

LinkedIn :- https://www.linkedin.com/in/ram-niwas-04/

FB group :-
https://www.facebook.com/groups/586730659057346/

Select-Option ALV IDA

```
DATA(lo_collector) = NEW cl_salv_range_tab_collector( ).
lo_collector->add_ranges_for_name( iv_name = 'VBELN'      it_ranges = s_vbeln[] ).
lo_collector->add_ranges_for_name( iv_name = 'VKORG'      it_ranges = s_vkorg[] ).

lo_collector->get_collected_ranges( IMPORTING et_named_ranges = DATA(lt_name_range_pairs) ).

DATA(lr_alv) = cl_salv_gui_table_ida=>create_for_cds_view( 'ZCDS_SAMPLE_01' ).

lr_alv->set_select_options( it_ranges = lt_name_range_pairs ).
lr_alv->fullscreen( )->display( ).
```

SAP TECHNOMANIAC

Complex Conditions

```
DATA(lr_alv) = cl_salv_gui_table_ida=>create_for_cds_view( 'ZDDLS_SAMPLE_02' ).
lr_alv->set_view_parameters( it_parameters = VALUE #( ( name = 'P_DATE' value = sy-datum ) ) )

data(lr_cond) = lr_alv->condition_factory( ).
data(lr_add_cond) = lr_cond->equals( name = 'SALES__ORG' value = 'NA01' )->or(
                    lr_cond->equals( name = 'SALES__ORG' value = '1710' )  ).
lr_alv->fullscreen( )->display( ).
```

Views with Input Parameters

```
DATA(lr_alv) = cl_salv_gui_table_ida=>create_for_cds_view( 'ZDDLS_SAMPLE_02' ).
lr_alv->set_view_parameters( it_parameters = VALUE #( ( name = 'P_DATE' value = sy-datum ) ) ).
lr_alv->fullscreen( )->display( ).
```

Checking Authorizations

```
data(lo_authorization_provider) = CL_SADL_COND_PROV_FACTORY_PUB=>create_for_authorization( ).
lo_authorization_provider->add_authorization_for_object( iv_authorization_object = 'S_CARRID'
                                                          it_activities           = VALUE #( ( auth_field = 'ACTVT' value = '03' ) )
                                                          it_field_mapping        = VALUE #( ( auth_field = 'CARRID' view_field = 'CARRID' ) ) ).
"Runtime error until Downport has been done
call method lo_alv_display->set_authorization_provider( lo_authorization_provider ).
lo_alv_display->fullscreen( )->display( ).
```

Transferring the authorization object has the consequence that only that data is read from the database to which the current user actually has access. This can potentially drastically reduce the data transfer to the user server.

Text Search Across the Whole Table

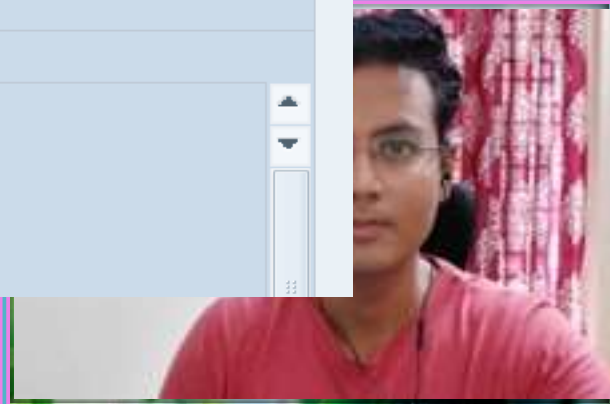
```
lo_container_d0555 = NEW #( 'D0555_CONTAINER' ).

" Instantiate and display ALV
lo_alv_display = cl_salv_gui_table_ida=>create( iv_table_name    = 'SFLIGHT'
                                                io_gui_container = lo_container_d0555 ).
"Allow Text Search in general
lo_alv_display->standard_functions( )->set_text_search_active( abap_true ).
"Release Text Search on textual columns
lo_alv_display->field_catalog( )->enable_text_search( 'CARRID' ).
lo_alv_display->field_catalog( )->enable_text_search( 'CONNID' ).
lo_alv_display->field_catalog( )->enable_text_search( 'PLANETYPE' ).
```

Search in all text columns: AZ

Search Options:

IDA ALV Sample: 1. Example to display Data (#52)

ID	No.	Flight Date	Airfare	Curr.	Plane Type	Capac	Occupi	Total	Capac	Occupi	Capac	Occupi
A	555	25.07.2019	185.00	EUR	A319-100	120	115	26,048.00	8	7	8	8
AZ	555	26.08.2019	185.00	EUR	A319-100	120	118	26,623.35	8	7	8	8
AZ	555	27.09.2019	185.00	EUR	A319-100	120	116	26,242.25	8	8	8	7
AZ	555	29.10.2019	185.00	EUR	A319-100	120	112	25,306.15	8	7	8	7
AZ	555	30.11.2019	185.00	EUR	A319-100	120	112	25,548.50	8	7	8	8



- ALV with IDA can be used with conventional databases as well as with in-memory databases.

Unlocking Performance: A Deep Dive into AMDP

- AMDP from basic to advance.
- This free playlist helps you to understand AMDP in detail.
- >AMDP basics and SQL Script session variable
- >Call CDS inside AMDP and how to handle clients inside AMDP
- >Call AMDP another AMDP and Parameter Interface mapping
- >Select option handling in AMDP
- >AMDP Scalar and AMDP tabular function
- >CDS table function vs AMDP table Function

SAP TECHNOMANIAC

THANK YOU

CONTACT

Ram Niwas

LinkedIn :- https://www.linkedin.com/in/ram-niwas-04/

FB group :-
https://www.facebook.com/groups/586730659057346/

SUBSRIBE

LIKE

COMMENT

SHARE

THANKS