Optimize Your Custom ABAP Code for SAP HANA

CAA104

PUBLIC

THE BEST RUN SAP

Speakers

Las Vegas

September 24–27, 2019

Christian Stork

Diego Will

Barcelona

October 8-10, 2019

Carlos Machado

Felix Fabis

Bangalore

November 13-15, 2019

B Sachin

Take the session survey.

We want to hear from you!

Complete the session evaluation for this session **CAA104** on the SAP TechEd mobile app.

Download the app from
iPhone App Store or Google Play.

WELCOME

SAP TECHED

SAP TechEd
Session Survey

Disclaimer

The information in this presentation is confidential and proprietary to SAP and may not be disclosed without the permission of SAP. Except for your obligation to protect confidential information, this presentation is not subject to your license agreement or any other service or subscription agreement with SAP. SAP has no obligation to pursue any course of business outlined in this presentation or any related document, or to develop or release any functionality mentioned therein.

This presentation, or any related document and SAP's strategy and possible future developments, products and or platforms directions and functionality are all subject to change and may be changed by SAP at any time for any reason without notice. The information in this presentation is not a commitment, promise or legal obligation to deliver any material, code or functionality. This presentation is provided without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement. This presentation is for informational purposes and may not be incorporated into a contract. SAP assumes no responsibility for errors or omissions in this presentation, except if such damages were caused by SAP’s intentional or gross negligence.

All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, which speak only as of their dates, and they should not be relied upon in making purchasing decisions.

Agenda

Overview

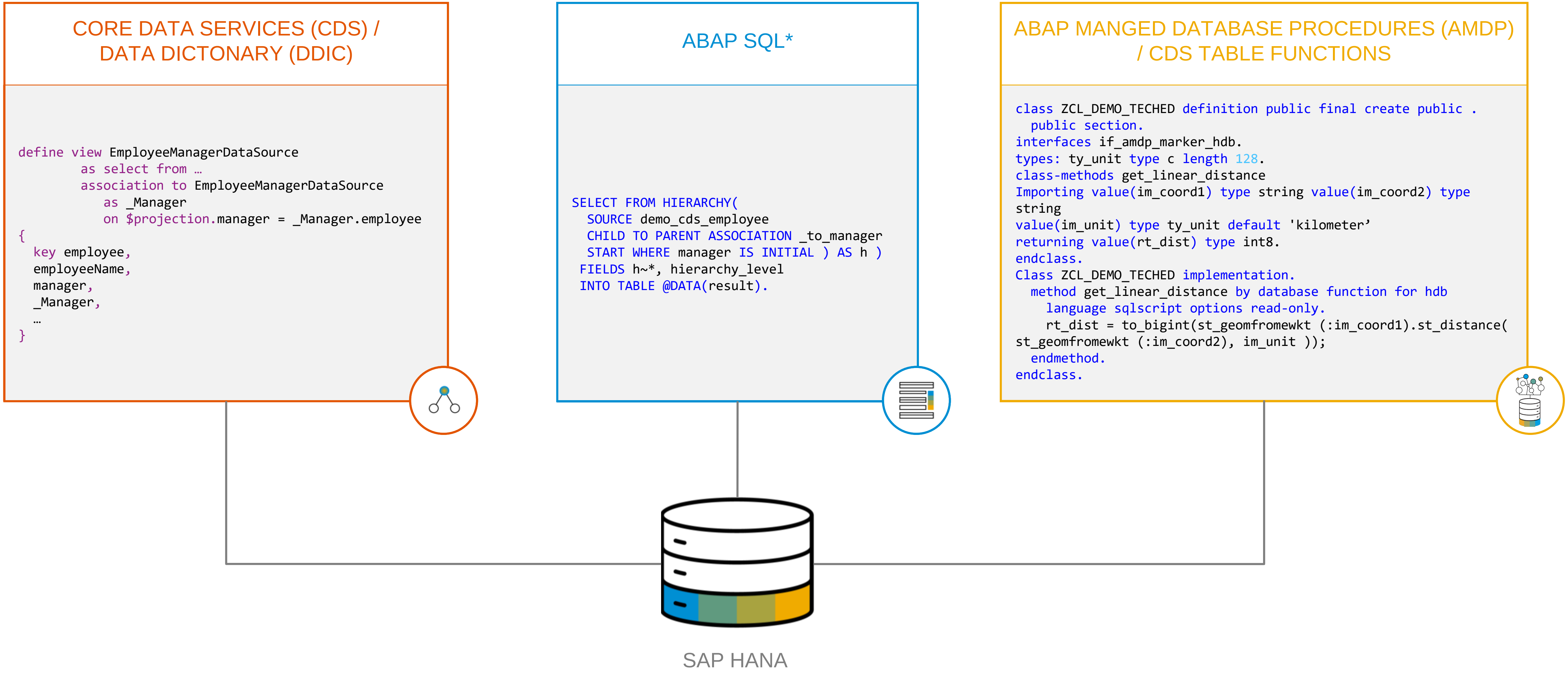
ABAP SQL

ABAP SQL Windowing

Hierarchies

New Datatypes in ABAP

Overview



CORE DATA SERVICES (CDS) / DATA DICTONARY (DDIC)

```
define view EmployeeManagerDataSource
        as select from …
        association to EmployeeManagerDataSource
           as _Manager
           on $projection.manager = _Manager.employee
{
  key employee,
  employeeName,
  manager,
  _Manager,
  …
}
```

ABAP SQL*

```
SELECT FROM HIERARCHY(
  SOURCE demo_cds_employee
  CHILD TO PARENT ASSOCIATION _to_manager
  START WHERE manager IS INITIAL ) AS h )
 FIELDS h~*, hierarchy_level
 INTO TABLE @DATA(result).
```

ABAP MANGED DATABASE PROCEDURES (AMDP) / CDS TABLE FUNCTIONS

```
class ZCL_DEMO_TECHED definition public final create public .
  public section.
interfaces if_amdp_marker_hdb.
types: ty_unit type c length 128.
class-methods get_linear_distance
Importing value(im_coord1) type string value(im_coord2) type
string
value(im_unit) type ty_unit default 'kilometer’
returning value(rt_dist) type int8.
endclass.
Class ZCL_DEMO_TECHED implementation.
  method get_linear_distance by database function for hdb
    language sqlscript options read-only.
    rt_dist = to_bigint(st_geomfromewkt (:im_coord1).st_distance(
st_geomfromewkt (:im_coord2), im_unit ));
  endmethod.
endclass.
```

SAP HANA

* former OPEN SQL

ABAP SQL

Inline declarations / host expressions - Definition

Inline declarations are a way of declaring variables and field symbols at operand positions

Let ABAP derive the correct type of variables from context and get rid of boilerplate coding

Inline declarations / host expressions

```
select uzeit, uname from snap where datum = @sy-datum and seqno = '000'
  into table @data(dumps_today).
```

INLINE DECLARATION

```
select uzeit, uname from (lv_nap) where datum = @sy-datum and seqno = '000'
  into table new @data(dumps_today).
```

INLINE DECLARATION

```
modify t100 from table @( value #( ( sprsl = 'E' arbgb = 'ABAP_SQL' msgnr = 'TOP' text = 'Improvements' )
                                   ( sprsl = 'E' arbgb = 'ABAP_SQL' msgnr = 'NEW' text = 'Features' ) ) ).
```

HOST EXPRESSION

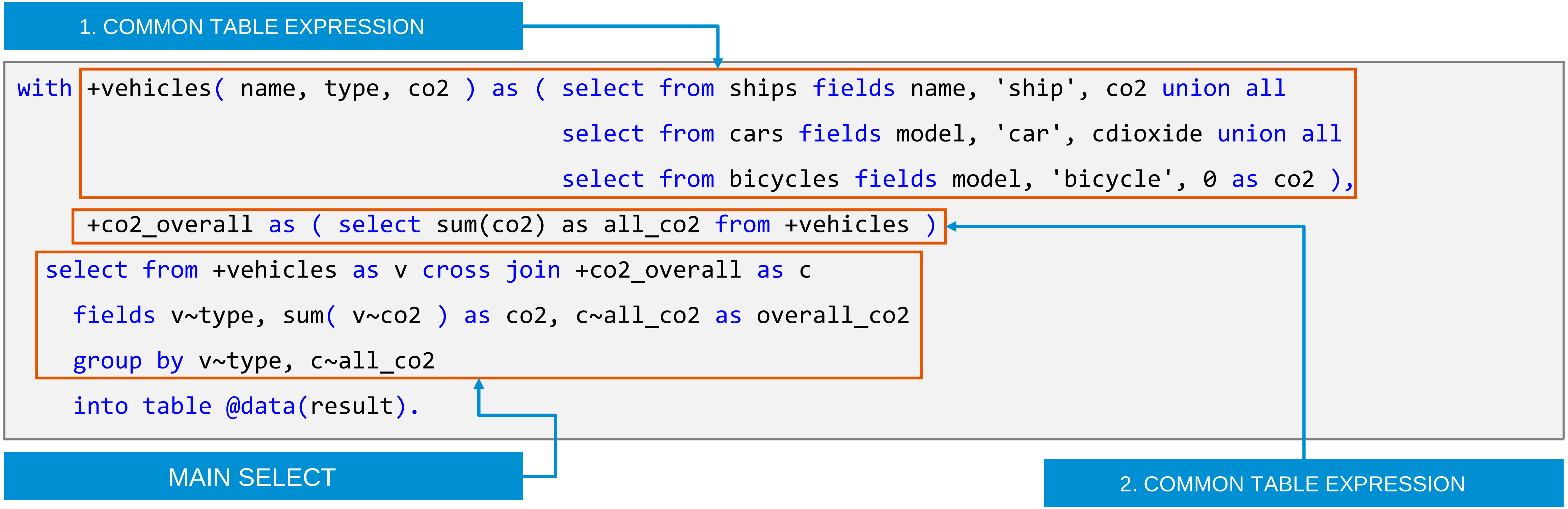
Common Table Expression (CTE) - Definition

Common Table Expressions (CTE) define temporary results sets in SQL queries that can be accessed in other SQL-statements

CTEs simplify complex joins and subqueries and provide straight-forward access to hierarchical data

Common Table Expression (CTE)

Example: Aggregate the carbon dioxide per vehicle type and select the overall consumption



1. COMMON TABLE EXPRESSION

```
with +vehicles( name, type, co2 ) as ( select from ships fields name, 'ship', co2 union all
                                       select from cars fields model, 'car', cdioxide union all
                                       select from bicycles fields model, 'bicycle', 0 as co2 ),
     +co2_overall as ( select sum(co2) as all_co2 from +vehicles )
  select from +vehicles as v cross join +co2_overall as c
    fields v~type, sum( v~co2 ) as co2, c~all_co2 as overall_co2
    group by v~type, c~all_co2
    into table @data(result).
```

MAIN SELECT

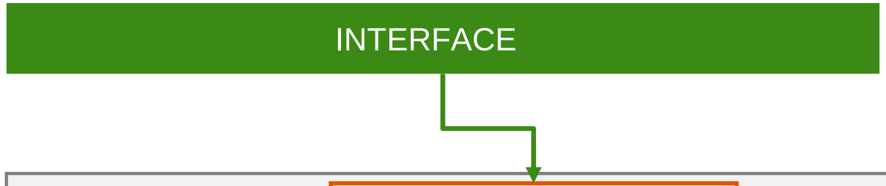
2. COMMON TABLE EXPRESSION

Common Table Expression (CTE)

NAME OF A CTE, FIRST CHARACTER HAS TO BE ‚+‘

```
with +vehicles ( name, type, co2 ) as ( select from ships fields name, 'ship', co2 union all
                                       select from cars fields model, 'car', cdioxide union all
                                       select from bicycles fields model, 'bicycle', 0 as co2 ),
     +co2_overall as ( select sum(co2) as all_co2 from +vehicles )
  select from +vehicles as v cross join +co2_overall as c
    fields v~type, sum( v~co2 ) as co2, c~all_co2 as overall_co2
    group by v~type, c~all_co2
    into table @data(result).
```

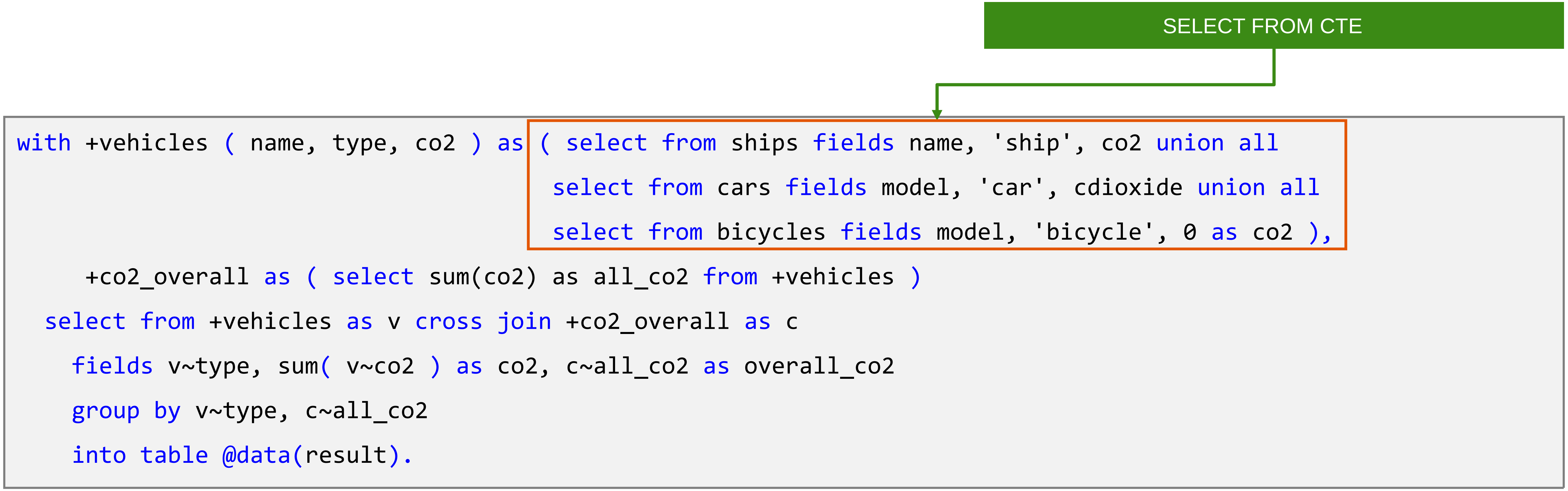
Common Table Expression (CTE)



```
with +vehicles ( name, type, co2 ) as ( select from ships fields name, 'ship', co2 union all
                                       select from cars fields model, 'car', cdioxide union all
                                       select from bicycles fields model, 'bicycle', 0 as co2 ),
     +co2_overall as ( select sum(co2) as all_co2 from +vehicles )
  select from +vehicles as v cross join +co2_overall as c
    fields v~type, sum( v~co2 ) as co2, c~all_co2 as overall_co2
    group by v~type, c~all_co2
    into table @data(result).
```

Common Table Expression (CTE)

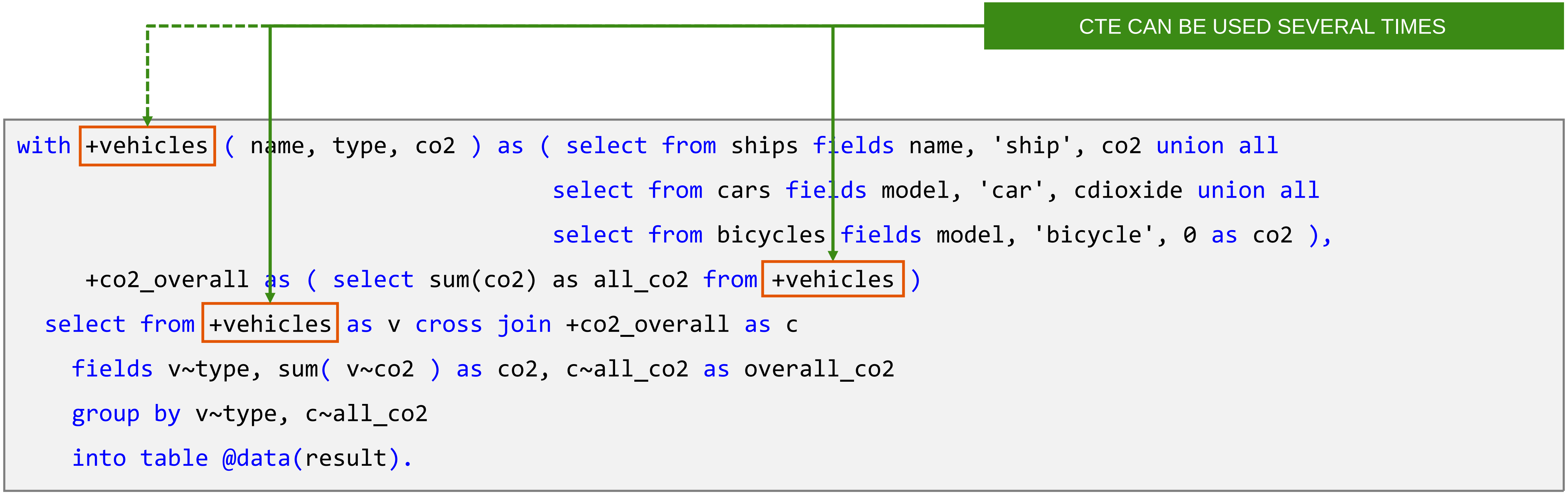
ABAP SQL



SELECT FROM CTE

```
with +vehicles ( name, type, co2 ) as ( select from ships fields name, 'ship', co2 union all
                                       select from cars fields model, 'car', cdioxide union all
                                       select from bicycles fields model, 'bicycle', 0 as co2 ),
     +co2_overall as ( select sum(co2) as all_co2 from +vehicles )
  select from +vehicles as v cross join +co2_overall as c
    fields v~type, sum( v~co2 ) as co2, c~all_co2 as overall_co2
    group by v~type, c~all_co2
    into table @data(result).
```

Common Table Expression (CTE)



CTE CAN BE USED SEVERAL TIMES

```
with +vehicles ( name, type, co2 ) as ( select from ships fields name, 'ship', co2 union all
                                       select from cars fields model, 'car', cdioxide union all
                                       select from bicycles fields model, 'bicycle', 0 as co2 ),
     +co2_overall as ( select sum(co2) as all_co2 from +vehicles )
  select from +vehicles as v cross join +co2_overall as c
    fields v~type, sum( v~co2 ) as co2, c~all_co2 as overall_co2
    group by v~type, c~all_co2
    into table @data(result).
```

COMMON TABLE EXPRESSIONS ARE TRANSIENT VIEWS IN AN SQL QUERY

UNION

ABAP SQL

CDS

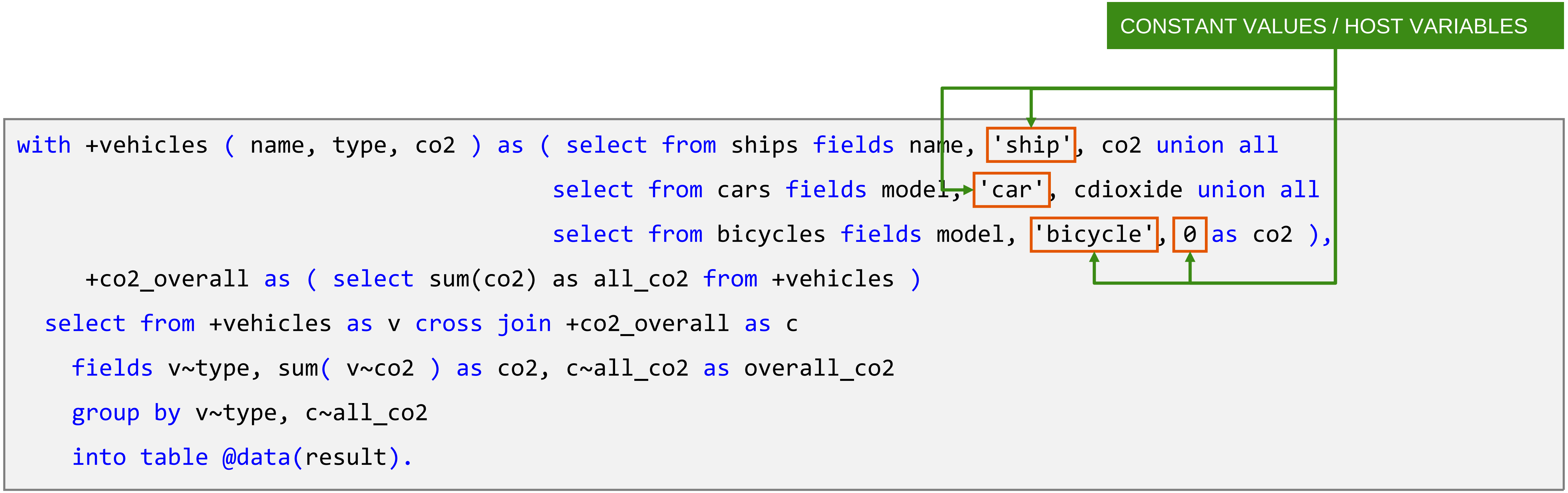
UNION ALL / UNION [DISTINCT]

```
with +vehicles ( name, type, co2 ) as ( select from ships fields name, 'ship', co2 union all
                                       select from cars fields model, 'car', cdioxide union all
                                       select from bicycles fields model, 'bicycle', 0 as co2 ),
     +co2_overall as ( select sum(co2) as all_co2 from +vehicles )
  select from +vehicles as v cross join +co2_overall as c
    fields v~type, sum( v~co2 ) as co2, c~all_co2 as overall_co2
    group by v~type, c~all_co2
    into table @data(result).
```

Host variables / constant values

ABAP SQL

CDS



CONSTANT VALUES / HOST VARIABLES

```
with +vehicles ( name, type, co2 ) as ( select from ships fields name, 'ship', co2 union all
                                       select from cars fields model, 'car', cdioxide union all
                                       select from bicycles fields model, 'bicycle', 0 as co2 ),
     +co2_overall as ( select sum(co2) as all_co2 from +vehicles )
  select from +vehicles as v cross join +co2_overall as c
    fields v~type, sum( v~co2 ) as co2, c~all_co2 as overall_co2
    group by v~type, c~all_co2
    into table @data(result).
```

JOIN

ABAP SQL

CDS

CROSS JOIN / RIGHT OUTER JOIN

```
with +vehicles ( name, type, co2 ) as ( select from ships fields name, 'ship', co2 union all
                                       select from cars fields model, 'car', cdioxide union all
                                       select from bicycles fields model, 'bicycle', 0 as co2 ),
     +co2_overall as ( select sum(co2) as all_co2 from +vehicles )
  select from +vehicles as v cross join +co2_overall as c
    fields v~type, sum( v~co2 ) as co2, c~all_co2 as overall_co2
    group by v~type, c~all_co2
    into table @data(result).
```

ABAP SQL
Windowing

Windowing - Definition

Windowing allows dividing data sets into subsets

Introduce ordering on your subsets and use it for access navigation, aggregate information over a subset without the need for grouping

Windowing

Example: Simple Data Model

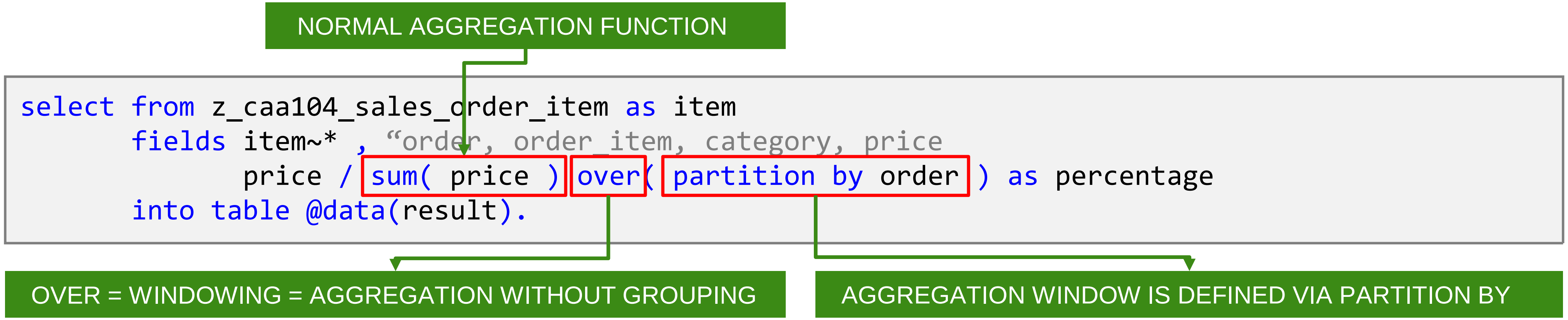
Order	Order Item	Category	Price	
1	11X127	Laptop	700€	
1	11X128	Laptop	300€	
1	11Y001	Desktop	2500€	
2	09A012	Drill hammer	500€	

Windowing

Order	Order Item	Category	Price	Price / Total
1	11X127	Laptop	700€	20%
1	11X128	Laptop	300€	9%
1	11Y001	Desktop	2500€	71%
2	09A012	Drill hammer	500€	100%

WINDOW

Windowing



NORMAL AGGREGATION FUNCTION

```
select from z_caa104_sales_order_item as item
       fields item~* , “order, order_item, category, price
              price / sum( price ) over( partition by order ) as percentage
       into table @data(result).
```

OVER = WINDOWING = AGGREGATION WITHOUT GROUPING

AGGREGATION WINDOW IS DEFINED VIA PARTITION BY

Order	Order Item	Category	Price	Price / Total
1	11X127	Laptop	700€	❯ 20%
1	11X128	Laptop	300€	❯ 9%
1	11Y001	Desktop	2500€	❯ 71%
2	09A012	Drill hammer	500€	❯ 100%

Combine grouping and windowing

Order	Order Item	Category	Price
1	11X127	Laptop	700€
1	11X128	Laptop	300€
1	11Y001	Desktop	2500€
2	09A012	Drill hammer	500€

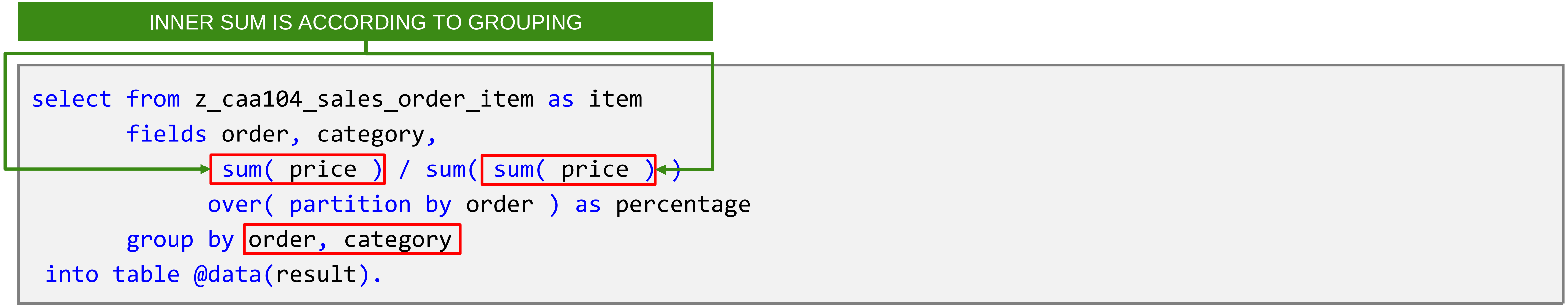
Grouping

Order	Category	Price
1	Laptop	1000€
1	Desktop	2500€
2	Drill hammer	500€

Windowing

Order	Category	Price / Total
1	Laptop	29%
1	Desktop	71%
2	Drill hammer	100%

Combine grouping and windowing



```
select from z_caa104_sales_order_item as item
       fields order, category,
             sum( price ) / sum( sum( price ) )
             over( partition by order ) as percentage
       group by order, category
 into table @data(result).
```

Order	Order Item	Category	Price
1	11X127	Laptop	700€
1	11X128	Laptop	300€
1	11Y001	Desktop	2500€
2	09A012	Drill hammer	500€

Grouping

Order	Category	Price
1	Laptop	1000€
1	Desktop	2500€
2	Drill hammer	500€

Combine grouping and windowing

ABAP SQL

OUTER SUM IS WINDOW FUNCTION

```
select from z_caa104_sales_order_item as item
       fields order, category,
              sum( price ) / sum( sum( price ) )
             over( partition by order ) as percentage
       group by order, category
 into table @data(result).
```

Order	Order Item	Category	Price
1	11X127	Laptop	700€
1	11X128	Laptop	300€
1	11Y001	Desktop	2500€
2	09A012	Drill hammer	500€

Grouping

Order	Category	Price
1	Laptop	1000€
1	Desktop	2500€
2	Drill hammer	500€

Windowing

Order	Category	Price / Total
1	Laptop	29%
1	Desktop	71%
2	Drill hammer	100%

Further Window Functions

FIRST GROUP BY CATEGORY IN CTE

```
with +catgry as ( select from z_caa104_sales_order_item as item
                         fields sales_order_nr, item_category, sum( price ) as sum
                         group by sales_order_nr, item_category )

 select from +catgry
        fields sales_order_nr, item_category,
               rank( ) over( partition by sales_order_nr order by sum descending ) as rank,
               sum / sum( sum ) over( partition by sales_order_nr ) as percentage,
into table @data(result).
```

BUILD RANK IN WINDOW ACCORDING TO SUM

Order	Order Item	Category	Price
1	11X127	Laptop	700€
1	11X128	Laptop	300€
1	11Y001	Desktop	2500€
2	09A012	Drill hammer	500€

Grouping

Order	Category	Price
1	Laptop	1000€
1	Desktop	2500€
2	Drill hammer	500€

Windowing

Order	Category	Price / Total	Rank
1	Laptop	29%	1
1	Desktop	71%	2
2	Drill hammer	100%	1

Further Window Functions

```
with +catgry as ( select from z_caa104_sales_order_item as item
                         fields sales_order_nr, item_category, sum( price ) as sum
                         group by sales_order_nr, item_category )

 select from +catgry
        fields sales_order_nr, item_category,
               rank( ) over( partition by sales_order_nr order by sum descending ) as rank,
               sum / sum( sum ) over( partition by sales_order_nr ) as percentage,
into table @data(result).
```

- ALL *CLASSICAL* AGGREGATE FUNCTIONS
- RANK in a window, may contain gaps, if some rows are equal
- DENSE_RANK as RANK but, without gaps
- ROW_NUMBER - Numbering of each row
- LEAD - Access to a subsequent line in a window
- LAG - Access to a prior line in a window

Hierarchies

Hierarchies - Definition

ABAP SQL

CDS

Hierarchies arrange data sets with self-associations into a tree model

Easily follow relations in your data set over arbitrary association steps, aggregate information along hierarchical relations and work on subtrees

Hierarchies

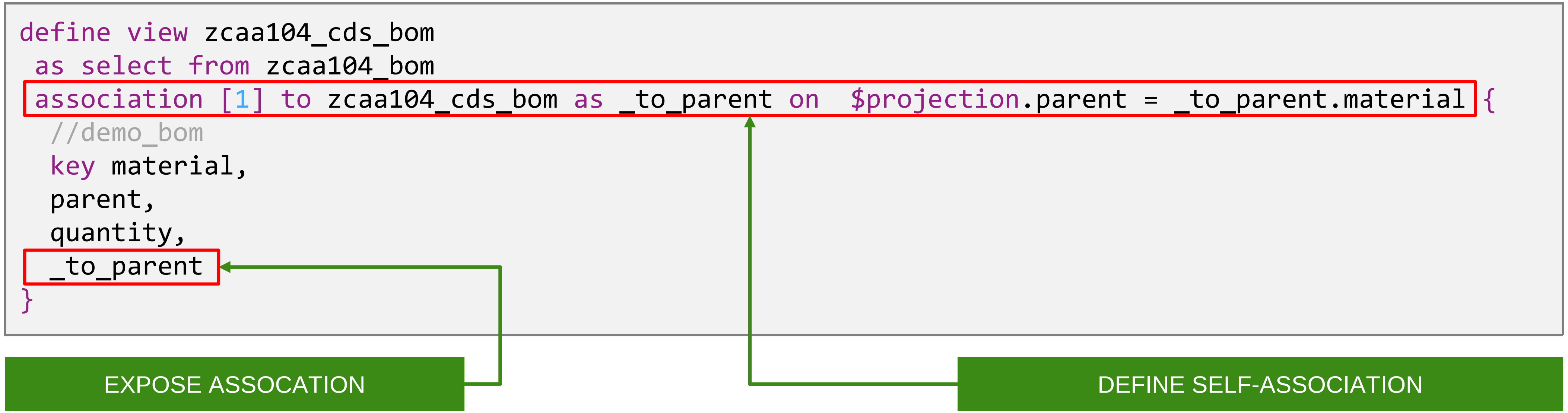
ABAP SQL

CDS

Example: Bill of materials (BOM)

Material	Parent	Quantity
Car		1
Tire	Car	4
Screw	Tire	5

Hierarchies - Hierarchy Source



```
define view zcaa104_cds_bom
 as select from zcaa104_bom
 association [1] to zcaa104_cds_bom as _to_parent on  $projection.parent = _to_parent.material {
  //demo_bom
  key material,
  parent,
  quantity,
  _to_parent
}
```

EXPOSE ASSOCATION

DEFINE SELF-ASSOCIATION

Material	Parent	Quantity
Car		1
Tire	Car	4
Screw	Tire	5

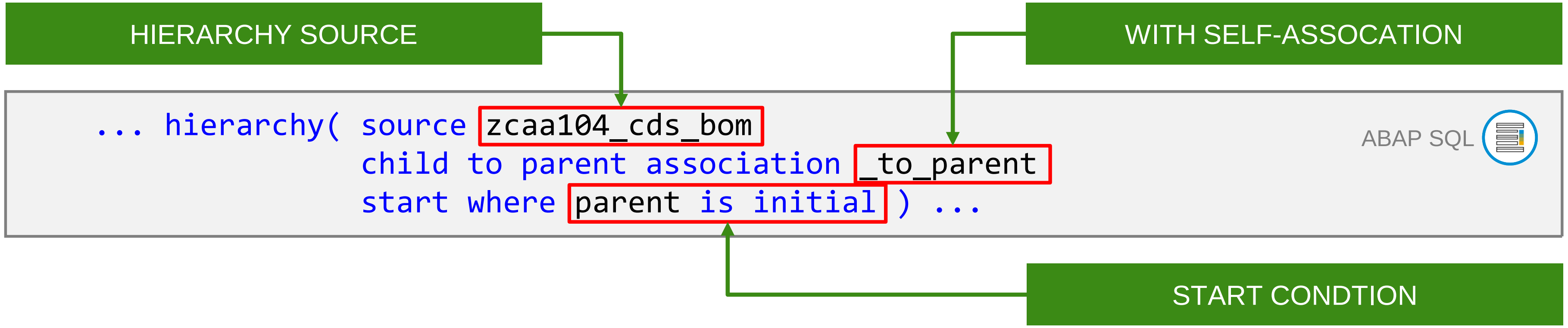
Hierarchies - Hierarchy Source

```
with +demo_asql_bom as ( select from zcaa104_bom fields material, parent, quantity )
       with associations ( join to one +demo_asql_bom as _to_parent
                                on +demo_asql_bom~parent = _to_parent~material ), ...
```

DEFINE SELF-ASSOCIATION, AUTOMATICALLY EXPOSED

Material	Parent	Quantity
Car		1
Tire	Car	4
Screw	Tire	5

Hierarchies - Definition



```
... hierarchy( source zcaa104_cds_bom
               child to parent association _to_parent
               start where parent is initial ) ...
```

Material	Parent	Quantity
Car		1
Tire	Car	4
Screw	Tire	5

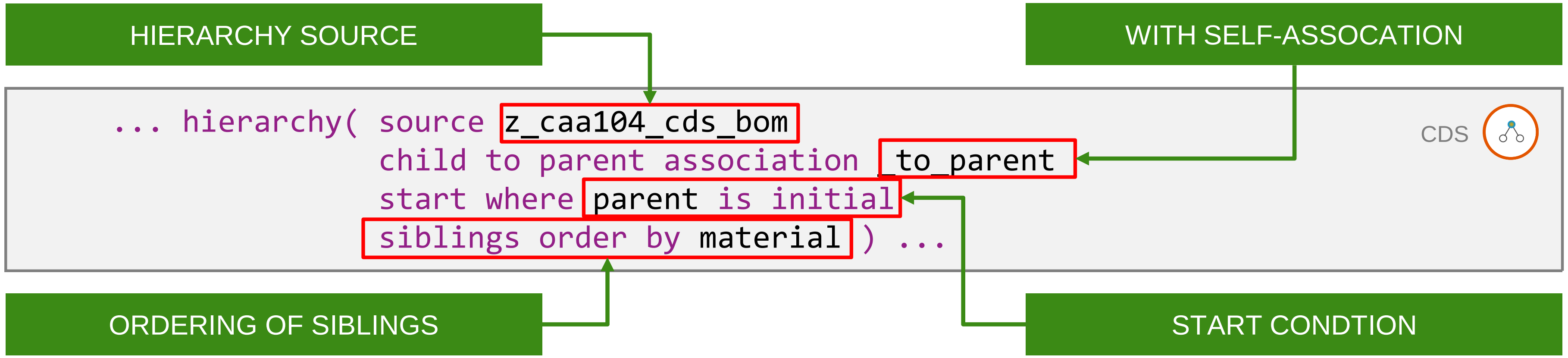
Hierarchies - Definition

ABAP SQL

CDS

ABAP SQL

```
... hierarchy( source zcaa104_cds_bom
               child to parent association _to_parent
               start where parent is initial ) ...
```



HIERARCHY SOURCE

WITH SELF-ASSOCATION

CDS

```
... hierarchy( source z_caa104_cds_bom
               child to parent association _to_parent
               start where parent is initial
               siblings order by material ) ...
```

ORDERING OF SIBLINGS

START CONDTION

Material	Parent	Quantity
Car		1
Tire	Car	4
Screw	Tire	5

ABAP SQL

CDS

Hierarchies
Exposing

EXPOSE HIERARCHY AND ASSOCIATION

ABAP SQL

```
with +hierarchy as ( select * from hierarchy( source z_caa104_cds_bom
                                               child to parent association _to_parent
                                               start where parent is initial ) as h )
                     with hierarchy h
                     with associations ( \_to_parent as to_parent ) ...
```

CDS

```
define hierarchy zcaa104_bom_hierarchy as parent child hierarchy( source zcaa104_cds_bom
                                        child to parent association _to_parent
                                        start where parent is initial
                                        siblings order by material ) {
material, parent, quantity, _to_parent }
```

Material	Parent	Quantity
Car		1
Tire	Car	4
Screw	Tire	5

EXPOSE HIERARCHY AND ASSOCIATION

Hierarchies - SELECT

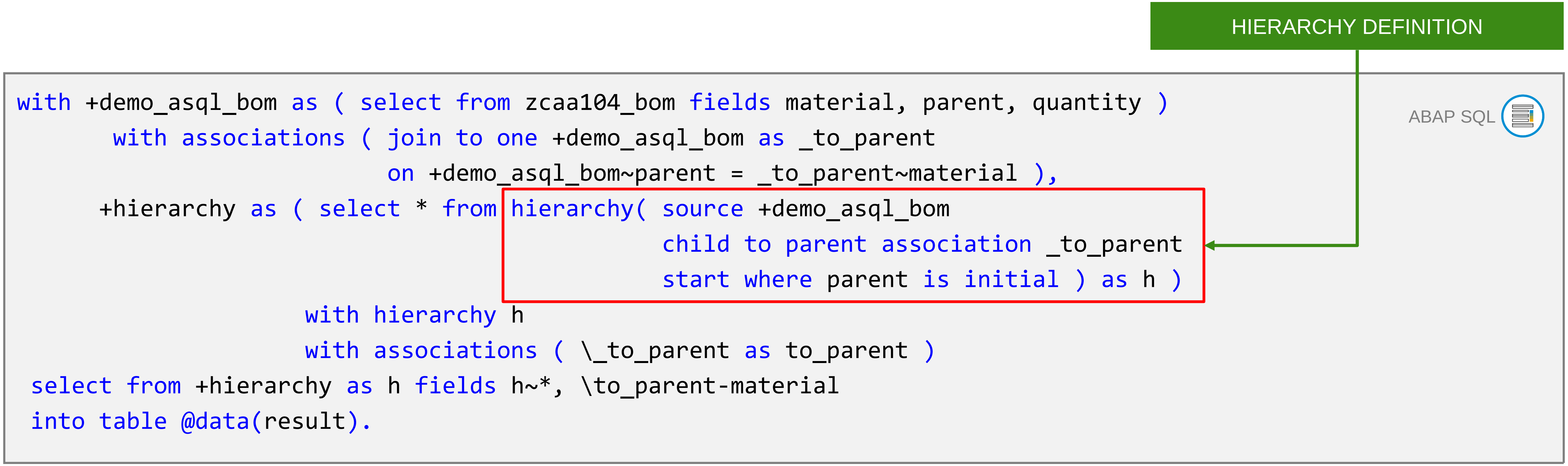
HIERARCHY SOURCE

ABAP SQL

```
with +demo_asql_bom as ( select from zcaa104_bom fields material, parent, quantity )
       with associations ( join to one +demo_asql_bom as _to_parent
                           on +demo_asql_bom~parent = _to_parent~material ),
      +hierarchy as ( select * from hierarchy( source +demo_asql_bom
                                               child to parent association _to_parent
                                               start where parent is initial ) as h )
                     with hierarchy h
                     with associations ( \_to_parent as to_parent )
 select from +hierarchy as h fields h~*, \to_parent-material
 into table @data(result).
```

Material	Parent	Quantity
Car		1
Tire	Car	4
Screw	Tire	5

Hierarchies - SELECT



HIERARCHY DEFINITION

```
with +demo_asql_bom as ( select from zcaa104_bom fields material, parent, quantity )
       with associations ( join to one +demo_asql_bom as _to_parent
                           on +demo_asql_bom~parent = _to_parent~material ),
      +hierarchy as ( select * from hierarchy( source +demo_asql_bom
                                               child to parent association _to_parent
                                               start where parent is initial ) as h )
                     with hierarchy h
                     with associations ( \_to_parent as to_parent )
 select from +hierarchy as h fields h~*, \to_parent-material
 into table @data(result).
```

Material	Parent	Quantity
Car		1
Tire	Car	4
Screw	Tire	5

Hierarchies - SELECT

HIERARCHY EXPOSE

ABAP SQL

```
with +demo_asql_bom as ( select from zcaa104_bom fields material, parent, quantity )
       with associations ( join to one +demo_asql_bom as _to_parent
                           on +demo_asql_bom~parent = _to_parent~material ),
      +hierarchy as ( select * from hierarchy( source +demo_asql_bom
                                               child to parent association _to_parent
                                               start where parent is initial ) as h )
                     with hierarchy h
                     with associations ( \_to_parent as to_parent )
 select from +hierarchy as h fields h~*, \to_parent-material
 into table @data(result).
```

Material	Parent	Quantity
Car		1
Tire	Car	4
Screw	Tire	5

Hierarchies - SELECT

ABAP SQL

```
with +demo_asql_bom as ( select from zcaa104_bom fields material, parent, quantity )
       with associations ( join to one +demo_asql_bom as _to_parent
                           on +demo_asql_bom~parent = _to_parent~material ),
      +hierarchy as ( select * from hierarchy( source +demo_asql_bom
                                               child to parent association _to_parent
                                               start where parent is initial ) as h )
                     with hierarchy h
                     with associations ( \_to_parent as to_parent )
 select from +hierarchy as h fields h~*, \to_parent-material
 into table @data(result).
```

HIERARCHY SELECT

Material	Parent	Quantity
Car		1
Tire	Car	4
Screw	Tire	5

Hierarchies - Accessor Functions

ABAP SQL

```
with +demo_asql_bom as ( select from zcaa104_bom fields material, parent, quantity )
       with associations ( join to one +demo_asql_bom as _to_parent
                           on +demo_asql_bom~parent = _to_parent~material ),
      +hierarchy as ( select * from hierarchy( source +demo_asql_bom
                                              child to parent association _to_parent
                                              start where parent is initial
                                              siblings order by material ) as h )
                     with hierarchy h
                     with associations ( \_to_parent as to_parent )
 select from hierarchy_ancestors_aggregate( source +hierarchy
                                            start where parent = 'Car'
                                            measures product( quantity ) as total_quantity
                                            where material = 'Screw' )
 fields *
 into table @data(result).
```

HIERARCHY DEFINITION

Material	Parent	Quantity
Car		1
Tire	Car	4
Screw	Tire	5

Hierarchies - Accessor Functions

- HIERARCHY_DESCENDANTS
  Navigate through descendants
- HIERARCHY_ANCESTORS
  Navigate through ancestors
- HIERARCHY_SIBLINGS
  Navigate through siblings
- HIERARCHY_DESCENDANTS_AGGREGATE
  Navigate through descendants with aggregation
- HIERARCHY_ANCESTORS_AGGREGATE
  Navigate through ancestors with aggregation

New Data Types
in ABAP

New Built-in data types

ABAP SQL

DDIC

DDIC Type	**Decfloat16**
ABAP Type	DECFLOAT16
Name	Decimal floating point number with 16 places

DDIC Type	**Decfloat34**
ABAP Type	DECFLOAT34
Name	Decimal floating point number with 34 places

DDIC Type	**Utclong**
ABAP Type	UTCLONG NEW ABAP TYPE !
Name	UTC Time stamp field

DDIC Type	**Datn**
ABAP Type	Date
Name	Date

DDIC Type	**Timn**
ABAP Type	TIME
Name	Time

DDIC Type	**Geom_ewbk**
ABAP Type	XSTRING
Name	Geometric data in EWKB representation

New Built-in data types - Query

ABAP SQL

```
select code, location , tracka_size, trackb_size, contruction_date, changelog
  from zcaa104_airports
  into table @data(lt_airports).
```

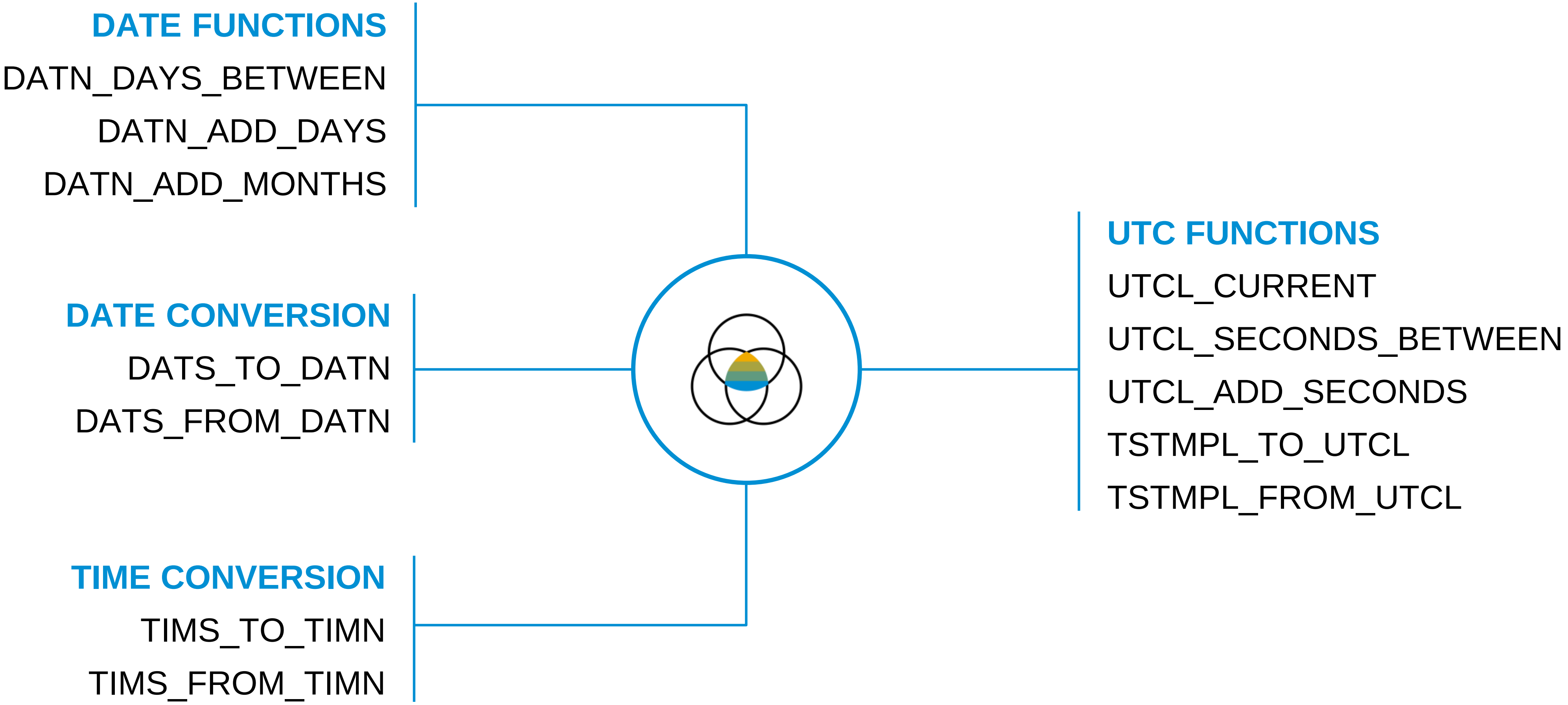
DDIC

TYPE	DECFLOAT34	DECFLOAT16	DATN	UTCLONG
Code	Tracka_size	Trackb_size	Construction_date	Changelog
FRA	13123	9186	1936-05-08	2019-07-06T18:30:32Z
JFK	11352	14573	1948-06-01	2019-07-03T08:32:00Z

New Built-in data types - SQL Functions

ABAP SQL

CDS



DATE FUNCTIONS

DATN_DAYS_BETWEEN

DATN_ADD_DAYS

DATN_ADD_MONTHS

DATE CONVERSION

DATS_TO_DATN

DATS_FROM_DATN

TIME CONVERSION

TIMS_TO_TIMN

TIMS_FROM_TIMN

UTC FUNCTIONS

UTCL_CURRENT

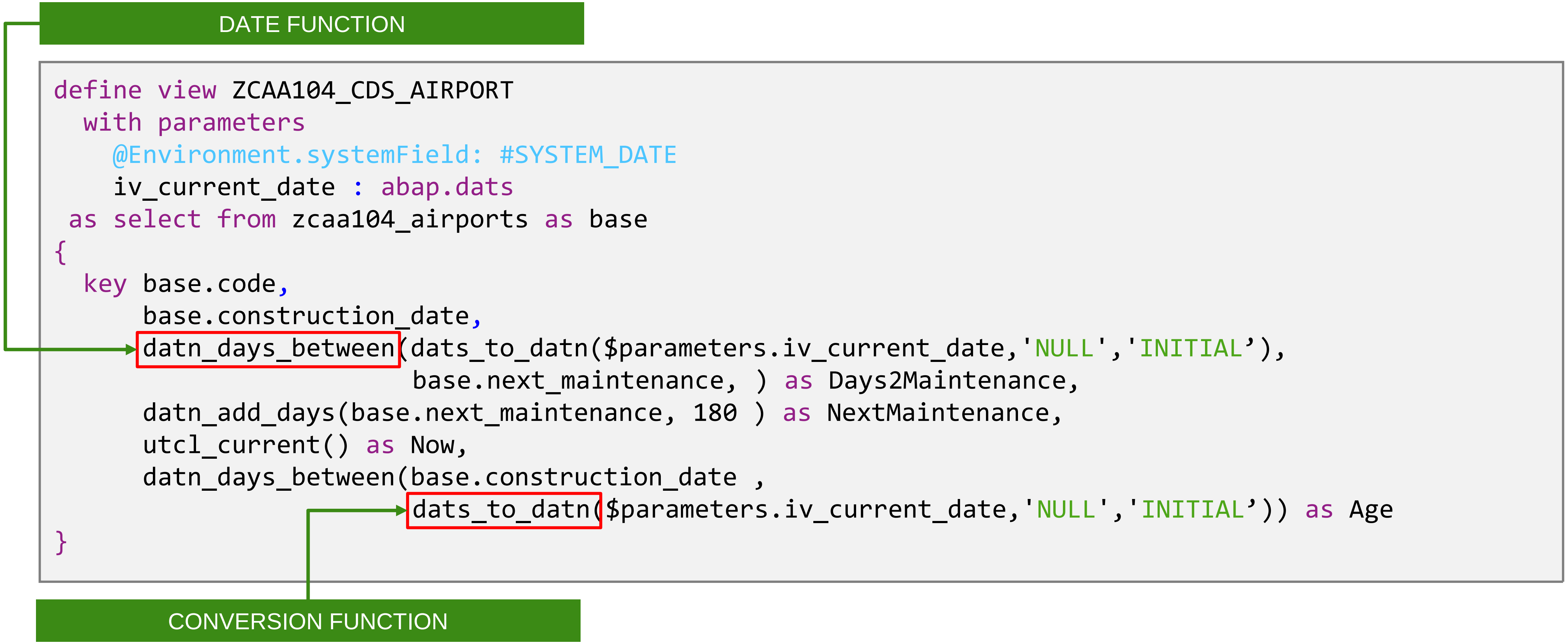
UTCL_SECONDS_BETWEEN

UTCL_ADD_SECONDS

TSTMPL_TO_UTCL

TSTMPL_FROM_UTCL

New Built-in data types - CDS SQL Functions

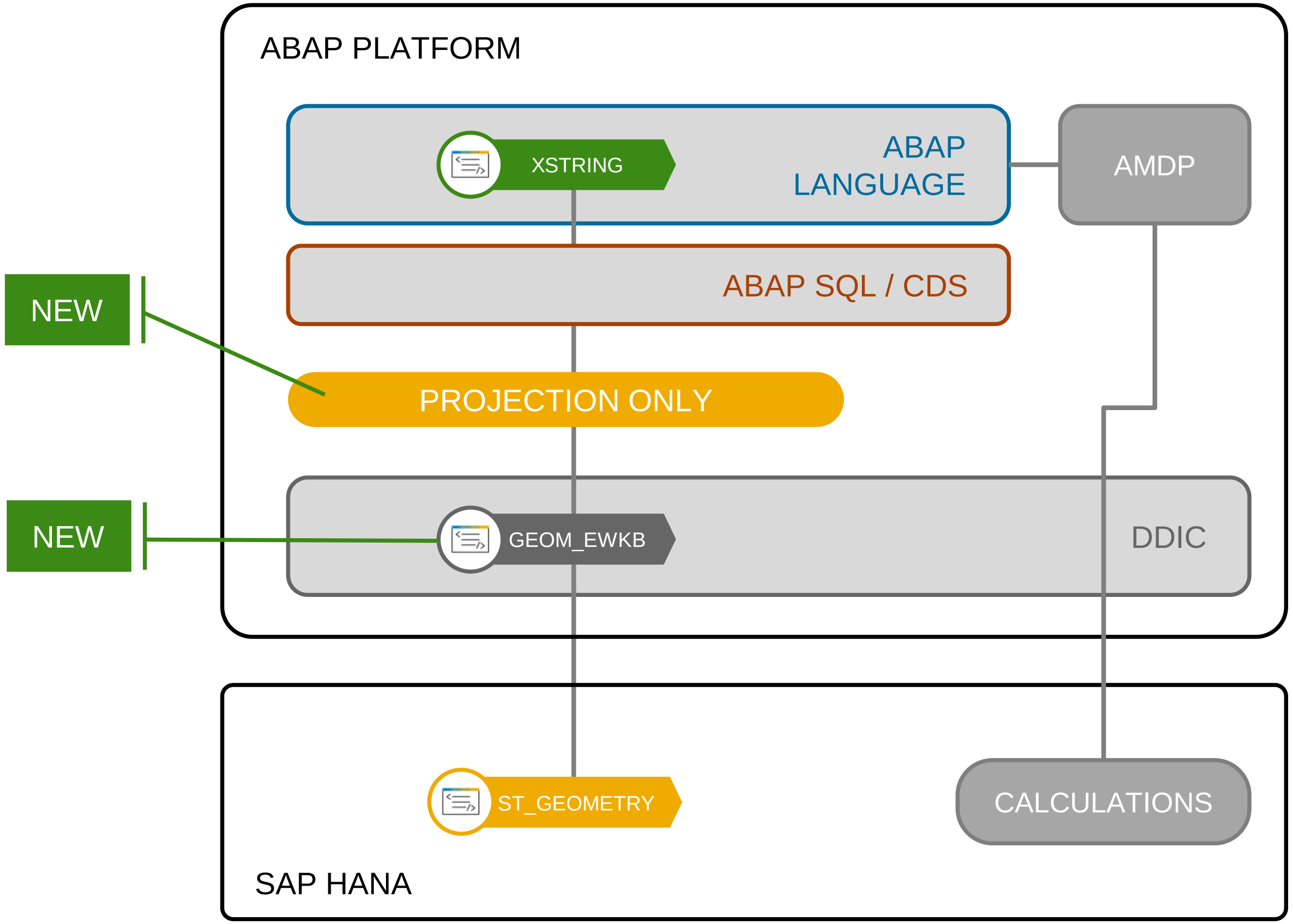


DATE FUNCTION

```
define view ZCAA104_CDS_AIRPORT
  with parameters
    @Environment.systemField: #SYSTEM_DATE
    iv_current_date : abap.dats
 as select from zcaa104_airports as base
{
  key base.code,
      base.construction_date,
      datn_days_between(dats_to_datn($parameters.iv_current_date,'NULL','INITIAL’),
                        base.next_maintenance, ) as Days2Maintenance,
      datn_add_days(base.next_maintenance, 180 ) as NextMaintenance,
      utcl_current() as Now,
      datn_days_between(base.construction_date ,
                        dats_to_datn($parameters.iv_current_date,'NULL','INITIAL’)) as Age
}
```

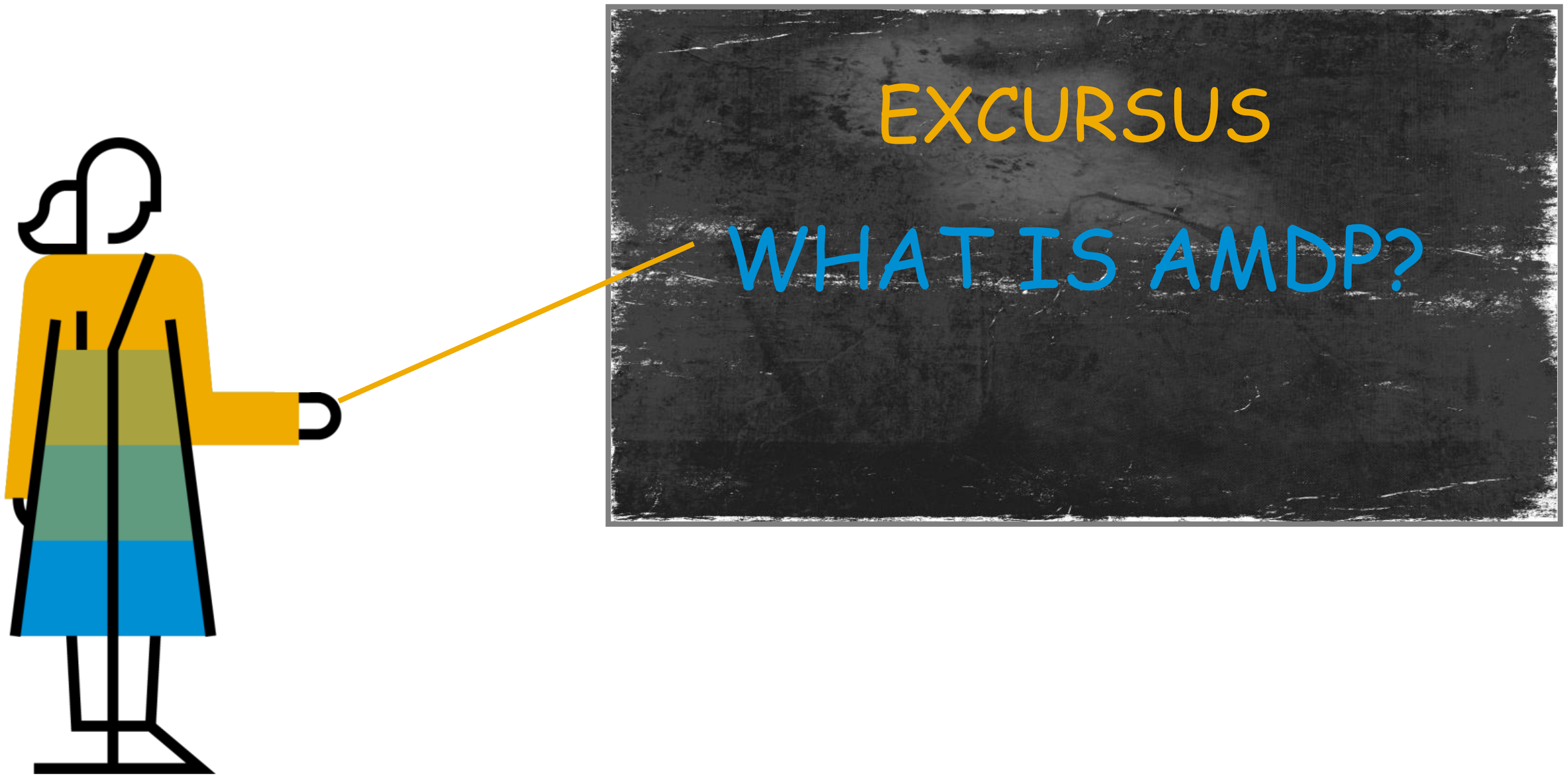
CONVERSION FUNCTION

GEO Spatial Data



OVERALL SCOPE

- New GEO Type in DDIC to allow usage in SAP HANA artefacts
  (DB Table, CDS View, SQL Query)
- **NO** GEO-specific semantics on ABAP-level. ABAP just allows dispatching of GEO data between consumers and SAP HANA
- GEO-specific functionality in SAP HANA can be used via AMDP



EXCURSUS

WHAT IS AMDP?

Why code ABAP Managed Database Procedures make sense …

Unleash the full power of your underlying SAP HANA database

Some scenarios require selective measures

Highest performance requirements e.g. with complex calculations

Use of database / analytical engine, specialized functions required

ABAP SQL and CDS views are not sufficient to solve problem efficiently

Restrictions

Database-specific

SAP HANA only

No automatic fallback for anyDB (!)

ABAP managed database procedures (AMDP) for SAP HANA

Utilize native SAP HANA entities

Stored procedures and database functions supported

Complex logic with if / else…

Parameterized requests and multiple result sets

Fully integrated in the ABAP infrastructure

Development, runtime error analysis, enhancement, transport

SQLScript coding embedded in ABAP classes

Seamless integration with CDS

Easy access to SAP HANA advanced engines / libraries

Like predictive analysis, financials, text mining, calculation engine

Seamless AMDP integration into CDS using CDS Table Functions

CDS table function definition

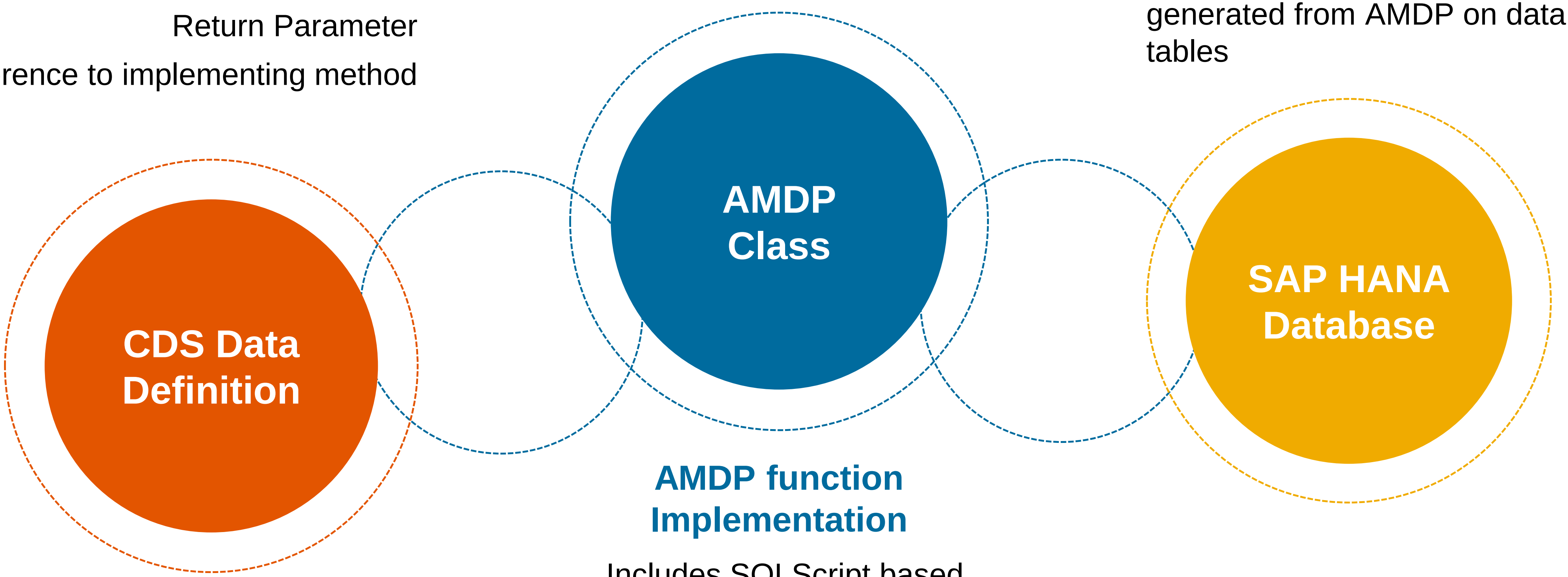
Parameter list

Return Parameter

Reference to implementing method

Runtime for table function

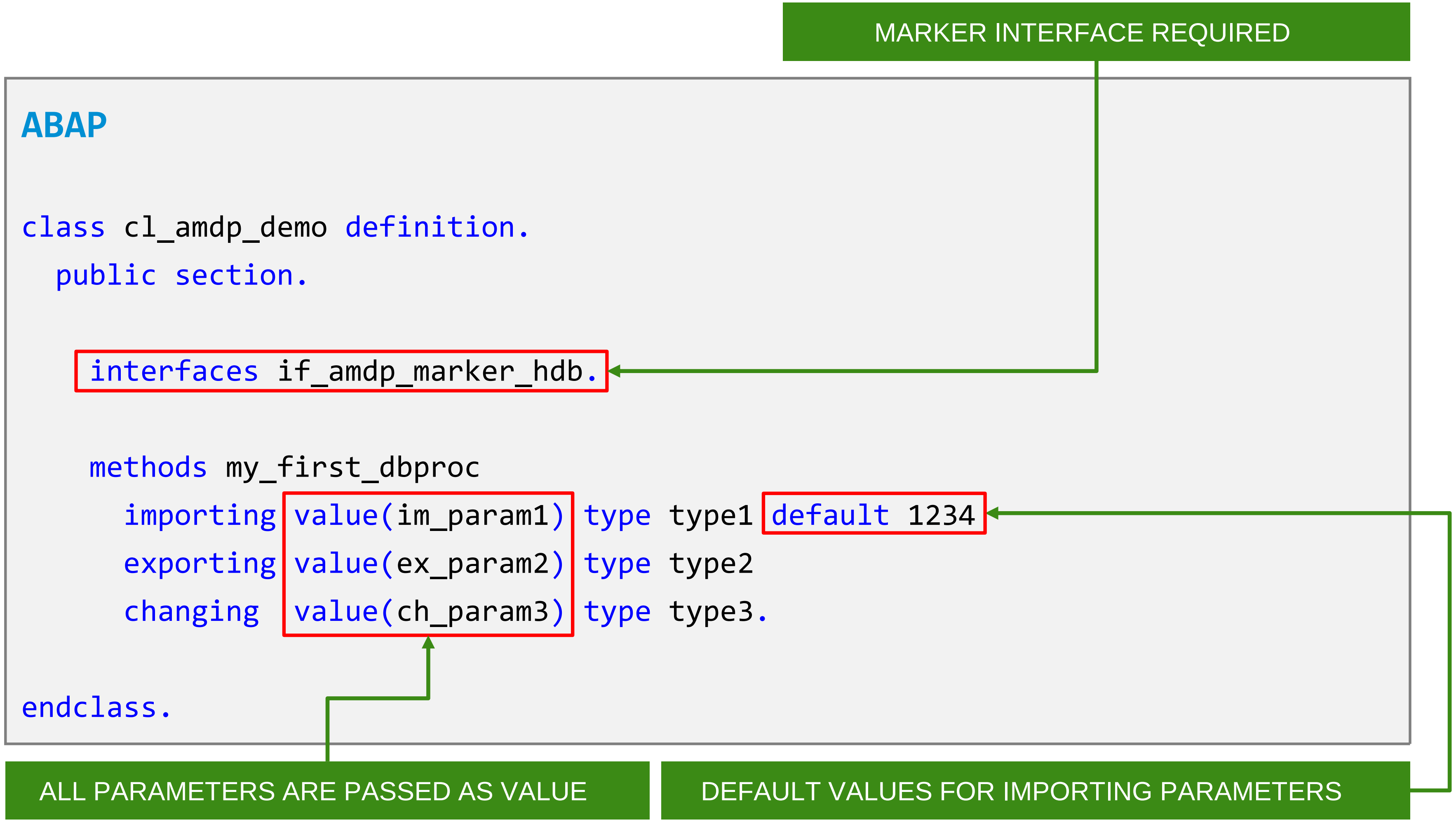
Runs stored SQLScript procedure generated from AMDP on database tables



AMDP function Implementation

Includes SQLScript based database function body

How to write AMDPs - Definition



MARKER INTERFACE REQUIRED

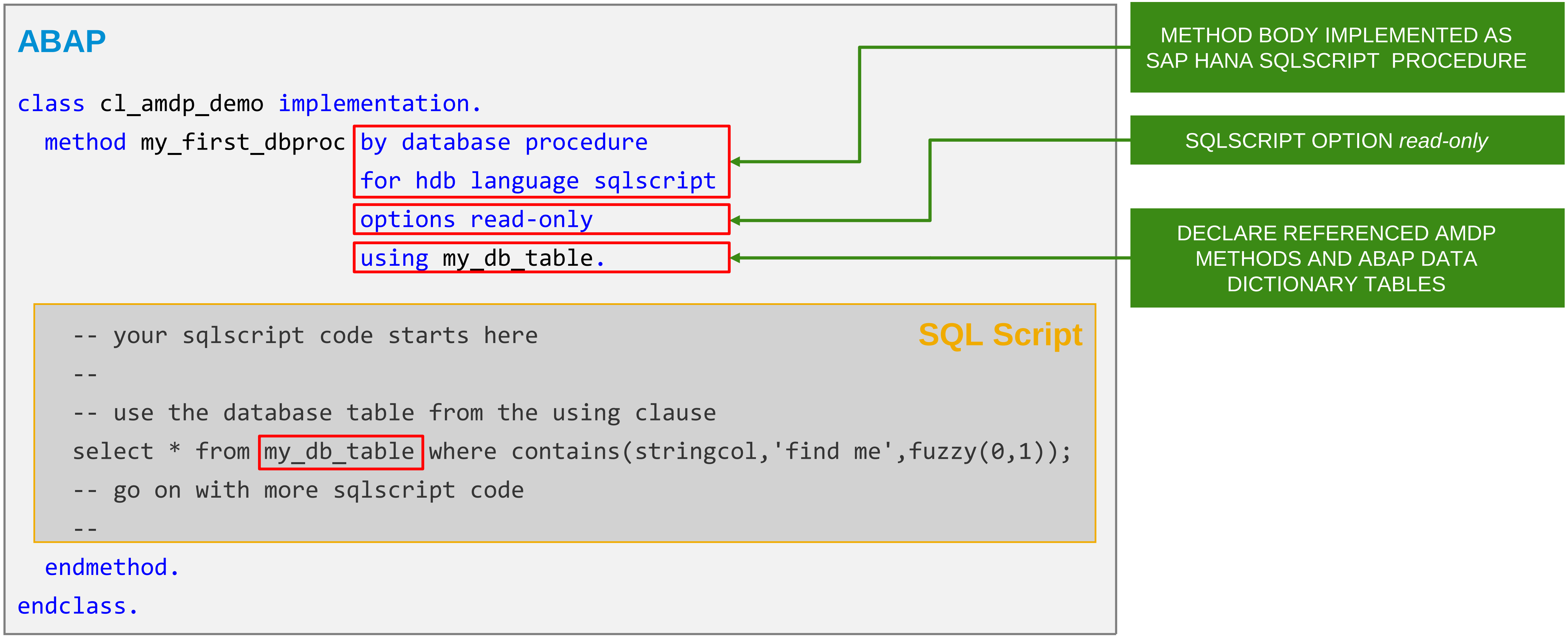
ABAP

```
class cl_amdp_demo definition.
  public section.

    interfaces if_amdp_marker_hdb.

    methods my_first_dbproc
      importing value(im_param1) type type1 default 1234
      exporting value(ex_param2) type type2
      changing  value(ch_param3) type type3.

endclass.
```

ALL PARAMETERS ARE PASSED AS VALUE

DEFAULT VALUES FOR IMPORTING PARAMETERS

AMDPs - Implementation



ABAP

```
class cl_amdp_demo implementation.
  method my_first_dbproc by database procedure
                         for hdb language sqlscript
                         options read-only
                         using my_db_table.

    -- your sqlscript code starts here
    --
    -- use the database table from the using clause
    select * from my_db_table where contains(stringcol,'find me',fuzzy(0,1));
    -- go on with more sqlscript code
    --
  endmethod.
endclass.
```

SQL Script

METHOD BODY IMPLEMENTED AS SAP HANA SQLSCRIPT PROCEDURE

SQLSCRIPT OPTION *read-only*

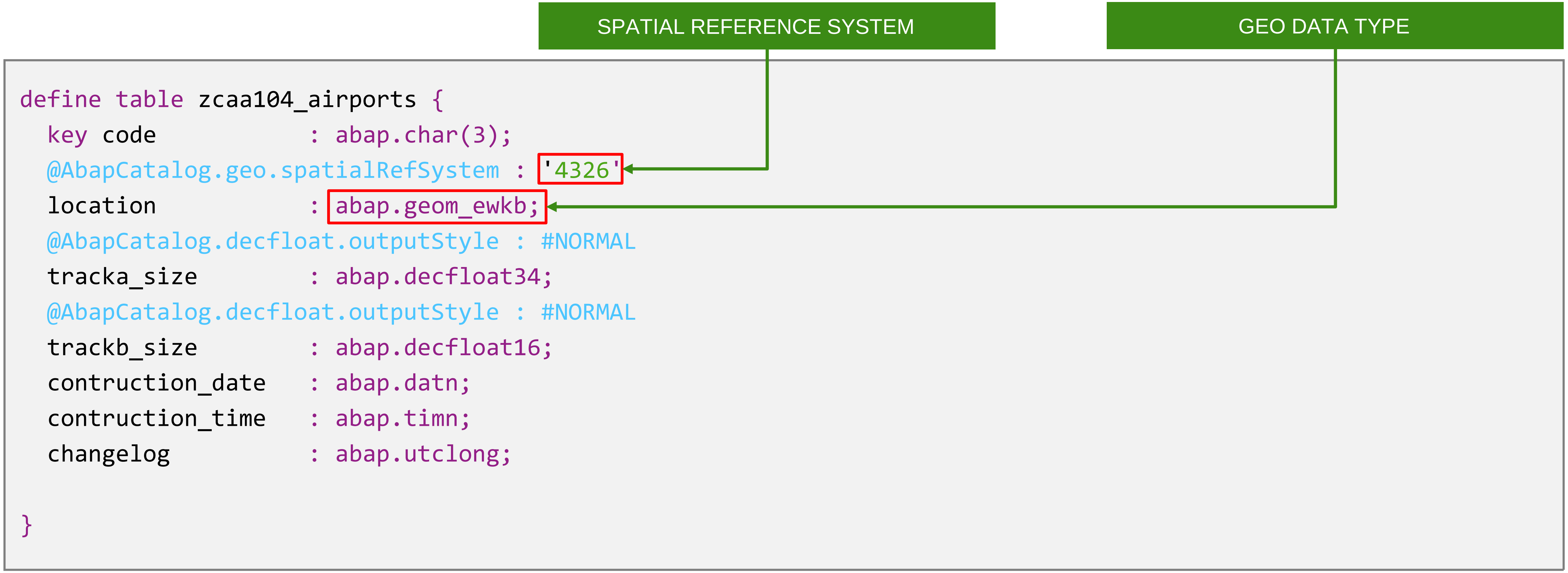
DECLARE REFERENCED AMDP METHODS AND ABAP DATA DICTIONARY TABLES

~~EXCURSUS~~

~~WHAT IS AMDP?~~

BACK TO GEO TYPE

GEO Spatial Data - Defining database table



SPATIAL REFERENCE SYSTEM

GEO DATA TYPE

```
define table zcaa104_airports {
  key code                  : abap.char(3);
  @AbapCatalog.geo.spatialRefSystem : '4326'
  location                  : abap.geom_ewkb;
  @AbapCatalog.decfloat.outputStyle : #NORMAL
  tracka_size               : abap.decfloat34;
  @AbapCatalog.decfloat.outputStyle : #NORMAL
  trackb_size               : abap.decfloat16;
  contruction_date          : abap.datn;
  contruction_time          : abap.timn;
  changelog                 : abap.utclong;

}
```

GEO Spatial Data - Query

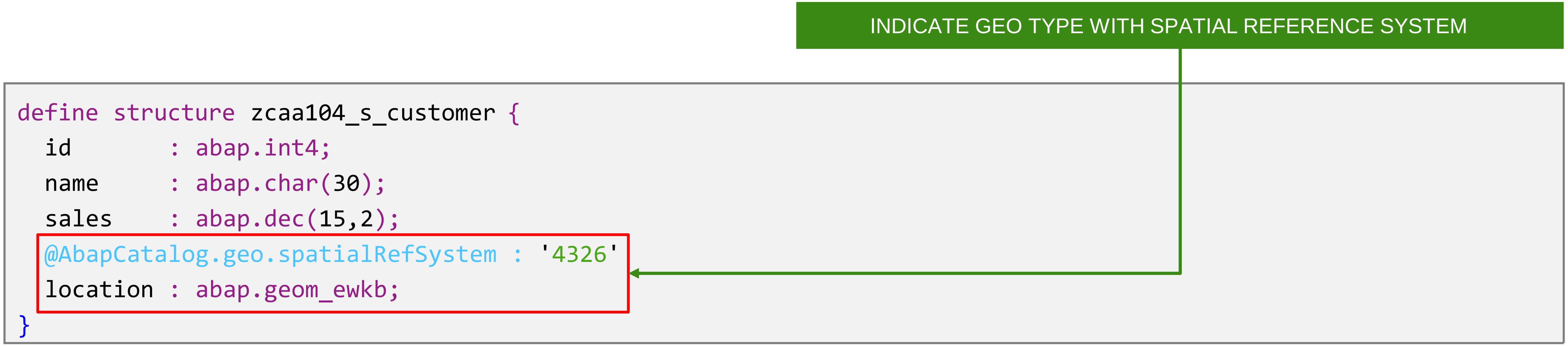
ABAP SQL

GEO TYPE FIELD

```
select id, name, location
  from zcaa014_customer
  into table @data(lt_customer).
```

id	name	location
1	SAP	0101000020E61000000000000000000014400000000000000014C0
2	SAP	0101000020E61000000000000000000014400000000000000014C0
3	SAP	0101000020E61000000000000000000014400000000000000014C0

GEO Spatial Data - Query



INDICATE GEO TYPE WITH SPATIAL REFERENCE SYSTEM

```
define structure zcaa104_s_customer {
  id        : abap.int4;
  name      : abap.char(30);
  sales     : abap.dec(15,2);
  @AbapCatalog.geo.spatialRefSystem : '4326'
  location : abap.geom_ewkb;
}
```

GEO Spatial Data - Query

AMDP

DDIC

```
define structure zcaa104_s_customer {
  id       : abap.int4;
  name     : abap.char(30);
  sales    : abap.dec(15,2);
  @AbapCatalog.geo.spatialRefSystem : '4326'
  location : abap.geom_ewkb;
}
```

AMDP

```
class zcl_caa104_spacial_data_amdp definition public final create public .
  public section.
    interfaces if_amdp_marker_hdb.
    types tt_customer type standard table of zcaa104_s_customer with empty key.
    class-methods coverage
      importing value(iv_id) type int4
      returning value(et_customer) type tt_customer.
endclass.

class zcl_caa104_spacial_data_amdp implementation.
 method coverage by database function for hdb language sqlscript options read-only
                     using zcaa104_customer zcaa104_state.

    return select id,  name, sales, location from zcaa104_customer as c
            where ( select boundary.st_covers(c.location) from zcaa104_state where id = :iv_state_id ) = 1;
  endmethod.
endclass.
```

SAP HANA FUNCTIONS

GEO Spatial Data - Query

```
class zcl_caa104_spacial_data_amdp definition public final create public.
  public section.
    interfaces if_amdp_marker_hdb.
    class-methods coverage
      importing value(iv_id) type int4
      returning value(et_customer) type tt_customer.
endclass.

class zcl_caa104_spacial_data_amdp implementation.


 method run.

    data(lt_customers) = zcl_caa104_spacial_data_amdp=>COVERAGE( iv_state_id = '1001'  ).

    "display result

  endmethod.
endclass.
```

AMDP CALL

Continue your SAP TechEd 2019 Learning Experience

Join the digital SAP TechEd Learning Room 2019 in SAP Learning Hub

- Access SAP TechEd **Learning Journeys**
- Discover **related** learning content
- Watch **webinars** of SAP TechEd lectures
- Learn about SAP’s latest innovations with **openSAP**
- Collaborate with **SAP experts**
- **Self-test** your knowledge
- Earn a SAP TechEd **knowledge badge**

Engage with the SAP TechEd Community

Access replays and continue your SAP TechEd discussion after the event within the **SAP Community**

Access replays

- Keynotes
- Live interviews
- Select lecture sessions

http://sapteched.com/online

Continue the conversation

- Read and reply to blog posts
- Ask questions
- Join discussions

sap.com/community

Check out the latest blogs

- See all SAP TechEd blog posts
- Learn from peers and experts

SAP TechEd blog posts

More information

Related SAP TechEd Learning Journeys

- CAA9 - Take your ABAP skills to SAP HANA and the Cloud

Related SAP TechEd sessions

- CAA100 - ABAP STRATEGY
- CAA101 - OVERVIEW SAP CLOUD PLATFORM, ABAP ENVIRONMENT
- CAA800 - ROAD MAP: ABAP PLATFORM
- CAA102 - CUSTOMER SUCCESS WITH SAP CLOUD PLATFORM ABAP ENVIRONMENT
- CAA103 - GET THE BIG PICTURE OF THE ABAP RESTFUL PROGRAMMING MODEL
- CAA361 - BUILD A TRANSACTIONAL SAP FIORI APP WITH THE ABAP RESTFUL PROGRAMMING MODEL
- CAA260 - MOVE YOUR ABAP CODE TO THE CLOUD
- CAA104 - OPTIMIZE YOUR CUSTOM ABAP CODE FOR SAP HANA
- CAA388 - SAP CLOUD PLATFORM ABAP ENVIRONMENT: EXTENSION AND INTEGRATION SCENARIO
- CAA300 - ABAP GIT INTEGRATION

Public SAP Web sites

- SAP Community: www.sap.com/community
- SAP products: www.sap.com/products

Thanks for attending this session.

Telekom.de 10:52

WELCOME

SAP TECHED

SAP TechEd
Session Survey

Feedback

Please complete your session evaluation
for CAA104.

Contact for further topic inquiries

Christian Stork
Development Architect
christian.stork@sap.com

Diego Will
Developer
diego.sebastian.will@sap.com

THE BEST RUN SAP®

Follow us

www.sap.com/contactsap

THE BEST RUN SAP®