SAP.C_ABAPD_2309.v2026-04-18.q39

Exam Code:	C_ABAPD_2309
Exam Name:	SAP Certified Associate - Back-End Developer - ABAP Cloud
Certification Provider:	SAP
Free Question Number:	39
Version:	v2026-04-18
# of views:	137
# of Questions views:	390
https://www.freepdfdumps.com/SAP.C_ABAPD_2309.v2026-04-18.q39.html	

NEW QUESTION: 1
In ABAP SQL, which of the following can be assigned an alias? Note: There are 2 correct answers to this question.
A. order criterion (from order by clause)
B. field (from field list)
C. database table
D. group criterion (from group by clause)
Answer: (SHOW ANSWER)
In ABAP SQL, an alias is a temporary name that can be assigned to a field or a database table in a query. An alias can be used to make the query more readable, to avoid name conflicts, or to access fields or tables with long names. An alias is created with the AS keyword and is only valid for the duration of the query1.
The following are examples of how to assign an alias to a field or a database table in ABAP SQL:
* B. field (from field list): A field is a column of a table or a view that contains data of a certain type. A field can be assigned an alias in the field list of a SELECT statement, which specifies the fields that are selected from the data source. For example, the following query assigns the alias name to the field carrname of the table scarr:
SELECT carrid, carrname AS name FROM scarr.
The alias name can be used instead of carrname in other clauses of the query, such as WHERE, GROUP BY, ORDER BY, and so on2.
* C. database table: A database table is a collection of data that is organized in rows and columns. A database table can be assigned an alias in the FROM clause of a SELECT statement, which specifies the data source that is selected from. For example, the following query assigns the alias c to the table scarr:
SELECT c.carrid, c.carrname FROM scarr AS c.
The alias c can be used instead of scarr in other clauses of the query, such as WHERE, JOIN, GROUP BY, ORDER BY, and so on3.

The following are not valid for assigning an alias in ABAP SQL:
* A. order criterion (from order by clause): An order criterion is a field or an expression that is used to sort the result set of a query in ascending or descending order. An order criterion cannot be assigned an alias in the ORDER BY clause of a SELECT statement, because the alias is not visible in this clause. The alias can only be used in the clauses that follow the clause where it is defined1.
* D. group criterion (from group by clause): A group criterion is a field or an expression that is used to group the result set of a query into subsets that share the same values. A group criterion cannot be assigned an alias in the GROUP BY clause of a SELECT statement, because the alias is not visible in this clause. The alias can only be used in the clauses that follow the clause where it is defined1.
References: 1: ALIASES - ABAP Keyword Documentation 2: SELECT List - ABAP Keyword Documentation 3: FROM Clause - ABAP Keyword Documentation

NEW QUESTION: 2
You want to provide a short description of the data definition for developers that will be attached to the database view Which of the following annotations would do this if you inserted it on line #27
A. @UI headerinto description label
B. @UI.badge.title.label
C. @EndUserText.quickInfo
D. @EndUserText label
Answer: (SHOW ANSWER)
The annotation that can be used to provide a short description of the data definition for developers that will be attached to the database view is the @EndUserText.label annotation. This annotation is used to specify a text label for the data definition that can be displayed in the development tools or in the documentation. The annotation can be inserted on line #27 in the code snippet provided in the question12. For example:
* The following code snippet uses the @EndUserText.label annotation to provide a short description of the data definition for the CDS view ZCDS_VIEW:
@AbapCatalog.sqlViewName: 'ZCDS_VIEW' @AbapCatalog.compiler.compareFilter: true @AbapCatalog.preserveKey: true @AccessControl.authorizationCheck: #CHECK @EndUserText.label:
'CDS view for flight data' "short description for developers define view ZCDS_VIEW as select from sflight { key carrid, key connid, key fldate, seatsmax, seatsocc } You cannot do any of the following:
* @UI.headerInfo.description.label: This annotation is used to specify a text label for the description field of the header information of a UI element. This annotation is not relevant for the data definition of a database view12.
* @UI.badge.title.label: This annotation is used to specify a text label for the title field of a badge UI element. This annotation is not relevant for the data definition of a database view12.

* @EndUserText.quickInfo: This annotation is used to specify a quick information text for the data definition that can be displayed as a tooltip in the development tools or in the documentation. This annotation is not the same as a short description or a label for the data definition12.
References: 1: ABAP CDS - SAP Annotations - ABAP Keyword Documentation - SAP Online Help 2: ABAP CDS - Data Definitions - ABAP Keyword Documentation - SAP Online Help

NEW QUESTION: 3
What are valid statements? Note: There are 3 correct answers to this question
A. In class CL1, the interface method is named if-ml.
B. Class CL2 uses the interface.
C. Class CL1 uses the interface.
D. In class CL2, the interface method is named ifl-ml.
E. Class CL1 implements the interface.
Answer: B,D,E (LEAVE A REPLY)
The following are the explanations for each statement:
* C: This statement is valid. Class CL1 uses the interface. This is because class CL1 implements the interface ifl using the INTERFACES statement in the public section of the class definition. The INTERFACES statement makes the class compatible with the interface and inherits all the components of the interface. The class can then use the interface components, such as the method ml, by using the interface component selector ~, such as ifl~ml12
* E: This statement is valid. Class CL1 implements the interface. This is because class CL1 implements the interface ifl using the INTERFACES statement in the public section of the class definition. The INTERFACES statement makes the class compatible with the interface and inherits all the components of the interface. The class must then provide an implementation for the interface method ml in the implementation part of the class, unless the method is declared as optional or abstract12
* D: This statement is valid. In class CL2, the interface method is named ifl~ml. This is because class CL2 has a data member named m0_ifl of type REF TO ifl, which is a reference to the interface ifl. The interface ifl defines a method ml, which can be called using the reference variable m0_ifl. The interface method ml has the name ifl~ml in the class, where ifl is the name of the interface and the character ~ is the interface component selector12 The other statements are not valid, as they have syntax errors or logical errors. These statements are:
* A: This statement is not valid. In class CL1, the interface method is named ifl~ml, not if-ml. This is
* because class CL1 implements the interface ifl using the INTERFACES statement in the public section of the class definition. The interface ifl defines a method ml, which can be called using the class name or a reference to the class. The interface method ml has the name ifl~ml in the class, where ifl is the name of the interface and the character ~ is the interface component selector. Using the character - instead of the character ~ will cause a syntax error12
* B: This statement is not valid. Class CL2 does not use the interface, but only has a reference to the interface. This is because class CL2 has a data member named m0_ifl of type REF TO ifl,

which is a reference to the interface ifl. The interface ifl defines a method ml, which can be called using the reference variable m0_ifl. However, class CL2 does not implement the interface ifl, nor does it inherit the interface components. Therefore, class CL2 does not use the interface, but only references the interface12 References: INTERFACES - ABAP Keyword Documentation, CLASS - ABAP Keyword Documentation

NEW QUESTION: 4
In a test method you call method cl_abap_unit_assert=>assert_equals(..) in the following way:
CLASS ltcl1 DEFINITION FOR TESTING RISK LEVEL HARMLESS DURATION SHORT.
PRIVATE SECTION.
METHODS m1 FOR TESTING.
ENDCLASS.
CLASS ltcl1 IMPLEMENTATION.
METHOD m1.
DATA: go_test_object TYPE REF TO zcl_to_be_tested.
CONSTANTS: lco_exp TYPE string VALUE 'test2'.
CREATE OBJECT go_test_object.
cl_abap_unit_assert=>assert_equals(
EXPORTING
act = go_class->mv_attribute
exp = lco_exp
msg = 'assert equals failed ' && go_test_object->mv_attribute && ' ' && lco_exp ENDMETHOD.
ENDCLASS.
What will happen if method parameters act and exp are not equal?
A. The test will be aborted.
B. The tested unit cannot be transported.
C. There will be a message in the test log.
D. The tested unit will automatically be appended to a default ABAP Test Cockpit Variant.
Answer: C (LEAVE A REPLY)

NEW QUESTION: 5
In a RESTful Application Programming application, in which objects do you bind a CDS view to create a value help? Note: There are 3 correct answers to this question.
A. Projection View
B. Service Definition
C. Metadata Extension
D. Data model view
E. Behavior definition
Answer: (SHOW ANSWER)

NEW QUESTION: 6

Which internal table type allows unique and non-unique keys?
A. Sorted
B. Hashed
C. Standard
Answer: (SHOW ANSWER)
The internal table type that allows both unique and non-unique keys is the standard table. A standard table has an internal linear index that can be used to access the table entries. The key of a standard table is always non- unique, which means that the table can contain duplicate entries. However, the system does not check the uniqueness of the key when inserting new entries, so the programmer can ensure that the key is unique by using appropriate logic. A standard table can be accessed either by using the table index or the key, but the response time for key access is proportional to the table size.
The other two internal table types, sorted and hashed, do not allow non-unique keys. A sorted table is filled in sorted order according to the defined table key, which must be unique. A sorted table can be accessed either by using the table index or the key, but the response time for key access is logarithmically proportional to the table size. A hashed table can only be accessed by using a unique key, which must be specified when declaring the table. A hashed table has no index, and the response time for key access is constant, regardless of the table size.
References: Internal Tables - ABAP Keyword Documentation, SAP ABAP: Types Of Internal Table Declaration - dan852.com

NEW QUESTION: 7
Which ABAP SQL clause allows the use of inline declarations?
A. FROM
B. INTO CORRESPONDING FIELDS OF
C. INTO
D. FIELDS
Answer: (SHOW ANSWER)
The ABAP SQL clause that allows the use of inline declarations is the INTO clause. The INTO clause is used to specify the target variable or field symbol where the result of the SQL query is stored. The INTO clause can use inline declarations to declare the target variable or field symbol at the same position where it is used, without using a separate DATA or FIELD-SYMBOLS statement. The inline declaration is performed using the DATA or @DATA operators in the declaration expression12. For example:
* The following code snippet uses the INTO clause with an inline declaration to declare a local variable itab and store the result of the SELECT query into it:
SELECT * FROM scarr INTO TABLE @DATA (itab).
* The following code snippet uses the INTO clause with an inline declaration to declare a field symbol
<fs> and store the result of the SELECT query into it:
SELECT SINGLE * FROM scarr INTO @<fs>.

You cannot do any of the following:
* FROM: The FROM clause is used to specify the data source of the SQL query, such as a table, a view, or a join expression. The FROM clause does not allow the use of inline declarations12.
* INTO CORRESPONDING FIELDS OF: The INTO CORRESPONDING FIELDS OF clause is used to specify the target structure or table where the result of the SQL query is stored. The INTO CORRESPONDING FIELDS OF clause does not allow the use of inline declarations. The target structure or table must be declared beforehand using a DATA or FIELD-SYMBOLS statement12.
* FIELDS: The FIELDS clause is used to specify the columns or expressions that are selected from the data source of the SQL query. The FIELDS clause does not allow the use of inline declarations. The FIELDS clause must be followed by an INTO clause that specifies the target variable or field symbol where the result is stored12.
References: 1: SELECT - ABAP Keyword Documentation - SAP Online Help 2: Inline Declarations - ABAP Keyword Documentation - SAP Online Help

NEW QUESTION: 8
Which of the following are parts of answers to this question.
A. Partitioning attributes
B. Extension
C. Field list
D. Semantic table attributes
Answer: (SHOW ANSWER)
A CDS view is a data definition that defines a data structure and a data selection from one or more data sources. A CDS view consists of several parts, but two of them are:
* Extension: An extension is an optional clause that allows a CDS view to extend another CDS view by adding new elements, annotations, or associations. The extension clause has the syntax EXTEND VIEW view_name WITH view_name. The first view_name is the name of the CDS view that is being extended, and the second view_name is the name of the CDS view that is doing the extension1.
* Field list: A field list is a mandatory clause that specifies the elements of the CDS view. The field list has the syntax SELECT FROM data_source { element_list }. The data_source is the name of the data
* source that the CDS view selects data from, and the element_list is a comma-separated list of elements that the CDS view exposes. The elements can be fields of the data source, expressions, associations, or annotations2.
The following example shows a CDS view that extends another CDS view and defines a field list: @AbapCatalog.sqlViewName: 'ZCDS_EXT' define view Z_CDS_Extension extend view Z_CDS_Base with Z_CDS_Extension as select from ztable { // field list key ztable.id as ID, ztable.name as Name, ztable.age as Age, // extension @Semantics.currencyCode: true ztable.currency as Currency } The other options are not parts of a CDS view, but rather related concepts:

* Partitioning attributes: Partitioning attributes are attributes that are used to partition a table into smaller subsets of data. Partitioning attributes are defined in the ABAP Dictionary for transparent tables and can improve the performance and scalability of data access. Partitioning attributes are not part of the CDS view definition, but rather the underlying table definition3.
* Semantic table attributes: Semantic table attributes are attributes that provide additional information about the meaning and usage of a table. Semantic table attributes are defined in the ABAP Dictionary for transparent tables and can be used to enhance the data modeling and consumption of the table. Semantic table attributes are not part of the CDS view definition, but rather the underlying table definition4.
References: 1: Extending CDS Views | SAP Help Portal 2: SELECT List - ABAP Keyword Documentation 3:
Partitioning Attributes - ABAP Keyword Documentation 4: Semantic Table Attributes - ABAP Keyword Documentation

NEW QUESTION: 9
When does SAP recommend to use a sorted or a hashed table respectively? Note: There are 2 correct answers to this question.
A. A hashed table, when you read a subset in a loop and specify a part of the key from the left without gaps.
B. A sorted table, when you read a single record and specify non key fields.
C. A sorted table, when you read a subset in a loop and specify a part of the key from the left ^ without gaps.
D. A hashed table, when you read a single record and specify the complete key.
Answer: (SHOW ANSWER)

NEW QUESTION: 10
Which of the following ABAP SQL statements are valid? Note: There are 2 correct answers to this question.
A. SELECT FROM /dmo/connection FIELDS carrid O airpfrom,
MAX(distance) AS dist_max, MIN(distance) AS dist_min GROUP BY carrid, airpfrom INTO TABLE
@DATA(lt_hits)
B. SELECT FROM /dmo/connection FIELDS \/ O carrid, airpfrom,
MAX(distance) AS dist_max, MIN(distance) AS dist_min INTO TABLE @DATA(lt_hits)
C. SELECT FROM /dmo/connection FIELDS V D MAX(distance) AS dist_max
MIN(distance) AS dist_min INTO TABLE @DATA(lt_hits).
D. SELECT FROM /dmo/connection FIELDS r-i carrid, airpfrom u GROUP BY carrid, connid INTO TABLE @DATA(lt_hits).
Answer: A,B (LEAVE A REPLY)
The following are the explanations for each ABAP SQL statement:

* A: This statement is valid. It selects the fields carrid, airpfrom, and the aggregate functions MAX(distance) and MIN(distance) from the table /dmo/connection, and groups the results by carrid and airpfrom. The aggregate functions are aliased as dist_max and dist_min. The results are stored in an internal table named It_hits, which is created using the inline declaration operator @DATA.
* B: This statement is valid. It is similar to statement A, except that it does not specify the GROUP BY clause. This means that the aggregate functions are applied to the entire table, and the results are stored in an internal table named It_hits, which is created using the inline declaration operator @DATA.
* C: This statement is invalid. It selects the aggregate functions MAX(distance) and MIN(distance) from the table /dmo/connection, but it does not specify any grouping or non-aggregate fields. This is not allowed in ABAP SQL, as the SELECT list must contain at least one non-aggregate field or a GROUP BY clause. The statement will cause a syntax error.
* D: This statement is invalid. It selects the fields carrid and airpfrom from the table /dmo/connection, and groups the results by carrid and connid. However, the field connid is not included in the SELECT list, which is not allowed in ABAP SQL, as the GROUP BY clause must contain only fields that are also in the SELECT list. The statement will cause a syntax error.
References: SELECT - ABAP Keyword Documentation, GROUP BY - ABAP Keyword Documentation

NEW QUESTION: 11
Which of the following are valid sort operations for internal tables? Note: There are 3 correct answers to this question.
A. SORT itab BY field1 field2.
Sort a standard table using
B. Sort a standard table using
SORT itab ASCENDING.
Sort a sorted table using
C. SORT itab BY fieldl ASCENDING field2 DESCENDING.
Sort a standard table using
D. SORT itab DESCENDING.
E. SORT itab.
Sort a sorted table using
Answer: A,B,E (LEAVE A REPLY)

NEW QUESTION: 12
Exhibit
Which of the following ABAP SQL snippets are syntactically correct ways to provide a value for the parameter on line #4? Note: There are 2 correct answers to this question
A. ...SELECT * FROM deno_cds_param_view_entity (p_date - '20230101')...)
B. ...SELECT * FROM demo_cds_param_view_entity (p_date: 20238181')...)

C. ...SELECT * FROM deno_cds_param_view_entity (p_date = @(cl_abap_context_info->get_system_date ())...

D. ...SELECT * FROM demo_cds_param_view entity (p_date: $session.system_date)...

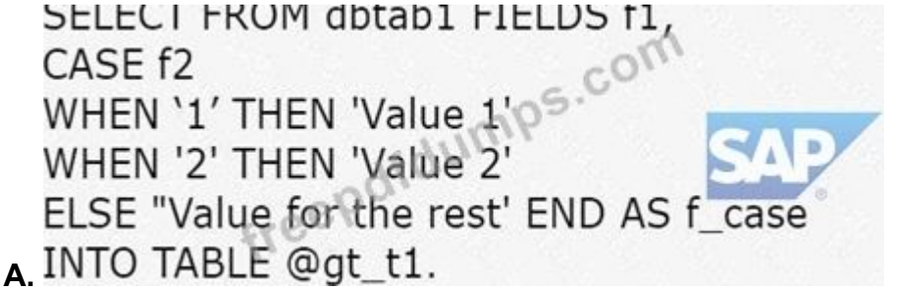
Answer: A,C (LEAVE A REPLY)

NEW QUESTION: 13

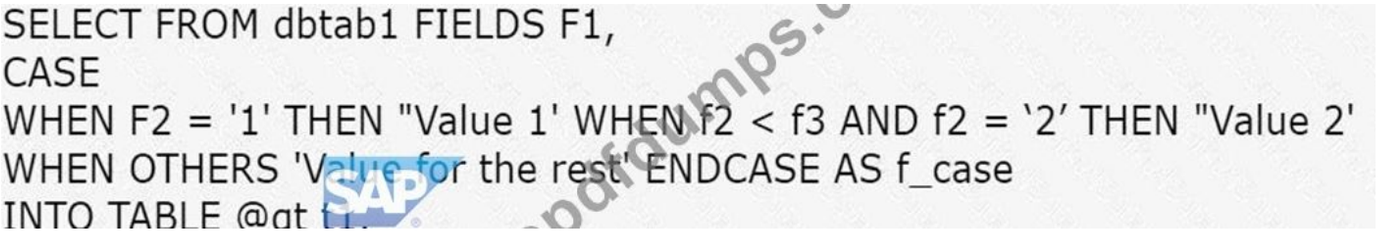
Image:

In the following ABAP SQL code, what are valid case distinctions? Note: There are 2 correct answers to this question.

A.



B.



C.

Answer: A,C (LEAVE A REPLY)

NEW QUESTION: 14
Exhibit:
What are valid statements? Note: There are 3 correct answers to this question.
A. go_if 1 may call method ml with go_ift->ml().
B. Instead of go ell = NEW #(...) you could use go ifl = NEW cll(. ...).
C. go_cll may call method ml with go_dl->ifl-ml().
D. Instead of go_cll = NEW #() you could use go_iff - NEW #(...).
E. go_ifl may call method m2 with go if->m2(...).
Answer: (SHOW ANSWER)
The following are the explanations for each statement:
* A: This statement is valid. go_ifl may call method ml with go_ifl->ml(). This is because go_ifl is a data object of type REF TO ifl, which is a reference to the interface ifl. The interface ifl defines a method ml, which can be called using the reference variable go_ifl. The class cll implements the interface ifl, which means that it provides an implementation of the method ml. The data object go_ifl is assigned to a new instance of the class cll using the NEW operator and the inline declaration operator @DATA. Therefore, when go_ifl->ml() is called, the implementation of the method ml in the class cll is executed123
* B: This statement is valid. Instead of go_cll = NEW #(...) you could use go_ifl = NEW cll(...). This is because go_ifl is a data object of type REF TO ifl, which is a reference to the interface ifl. The class cll implements the interface ifl, which means that it is compatible with the interface ifl. Therefore, go_ifl can be assigned to a new instance of the class cll using the NEW operator and the class name cll. The inline declaration operator @DATA is optional in this case, as go_ifl is already declared. The parentheses after the class name cll can be used to pass parameters to the constructor of the class cll, if any123
* E: This statement is valid. go_ifl may call method m2 with go_ifl->m2(...). This is because go_ifl is a data object of type REF TO ifl, which is a reference to the interface ifl. The class cll implements the interface ifl, which means that it inherits all the components of the interface ifl. The class cll also defines a method m2, which is a public method of the class cll. Therefore, go_ifl can call the method m2 using the reference variable go_ifl. The method m2 is not defined in the interface ifl, but it is accessible
* through the interface ifl, as the interface ifl is implemented by the class cll. The parentheses after the method name m2 can be used to pass parameters to the method m2, if any123 The other statements are not valid, as they have syntax errors or logical errors. These statements are:
* C: This statement is not valid. go_cll may call method ml with go_cll->ifl~ml(). This is because go_cll is a data object of type REF TO cll, which is a reference to the class cll. The class cll implements the interface ifl, which means that it inherits all the components of the interface ifl. The interface ifl defines a method ml, which can be called using the reference variable go_cll. However, the syntax for calling an interface method using a class reference is go_cll->ml(), not go_cll->ifl~ml(). The interface component selector ~ is only used when calling an interface

method using an interface reference, such as go_ifl->ifl~ml(). Using the interface component selector ~ with a class reference will cause a syntax error123
* D: This statement is not valid. Instead of go_cll = NEW #() you could use go_ifl = NEW #(...). This is because go_ifl is a data object of type REF TO ifl, which is a reference to the interface ifl. The interface ifl cannot be instantiated, as it does not have an implementation. Therefore, go_ifl cannot be assigned to a new instance of the interface ifl using the NEW operator and the inline declaration operator @DATA.
This will cause a syntax error or a runtime error. To instantiate an interface, you need to use a class that implements the interface, such as the class cll123 References: INTERFACES - ABAP Keyword Documentation, CLASS - ABAP Keyword Documentation, NEW - ABAP Keyword Documentation

NEW QUESTION: 15
What are some features of a unique secondary key? Note: There are 2 correct answers to this question.
A. It is created when a table is filled.
B. It is updated when the modified table is read again.
C. It is created with the first read access of a table.
D. It is updated when the table is modified.
Answer: C,D (LEAVE A REPLY)
A unique secondary key is a type of secondary key that ensures that the key combination of all the rows in a table is unique. A unique secondary key has two purposes: firstly, to speed up access to the table, and secondly, to enforce data integrity1.
* It is created with the first read access of a table: This is true. A unique secondary key is created when an internal table is filled for the first time using the statement READ TABLE or a similar statement. The system assigns a name and an index to each row of the table based on the key fields23.
* It is updated when the modified table is read again: This is false. A unique secondary key does not need to be updated when the internal table content changes, because it already ensures data uniqueness. The system uses a lazy update strategy for non-unique secondary keys, which means that it delays updating them until they are actually accessed23.
You cannot do any of the following:
* It is created when a table is filled: This is false. As explained above, a unique secondary key is created only with the first read access of a table23.
* It is updated when the modified table is read again: This is false. As explained above, a unique secondary key does not need to be updated when the internal table content changes23.
References: 1: Improving Internal Table Performance Using Secondary Keys - SAP Learning 2: [Secondary Key - ABAP Keyword Documentation - SAP Online Help] 3: [Secondary Table Key - ABAP Keyword Documentation - SAP Online Help]

NEW QUESTION: 16

In which products must you use the ABAP Cloud Development Model? Note: There are 2 correct answers to this question.
A. SAP S/4HANA Cloud, private edition
B. SAP BTP, ABAP environment
C. SAP S/4HANA on premise
D. SAP S/4HANA Cloud, public edition
Answer: A,B (LEAVE A REPLY)
The ABAP Cloud Development Model is the ABAP development model to build cloud-ready business apps, services, and extensions. It comes with SAP BTP and SAP S/4HANA. It works with public or private cloud, and even on-premise1. However, the complete ABAP Cloud Development Model, including the cloud-optimized ABAP language and public local SAP APIs and extension points, is available only in SAP BTP ABAP Environment and in the 2208/2022 versions of the SAP S/4HANA editions1. Therefore, you must use the ABAP Cloud Development Model in SAP BTP, ABAP environment and SAP S/4HANA Cloud, private edition. You can also use it in SAP S/4HANA on premise, but it is not mandatory. You cannot use it in SAP S/4HANA Cloud, public edition, because it does not allow custom ABAP code2. References: 1: ABAP Cloud | SAP Blogs 2: SAP S/4HANA Cloud Extensibility - Overview and Comparison | SAP Blogs

NEW QUESTION: 17
In a program you find this source code
AUTHORITY-CHECK OBJECT '/DWO/TRVL (ID 'CNTRY' FIELD 'DE*
ID ACTVT FIELD '03".
Which of the following apply? Note: There are 2 correct answers to this question.
A. AUTHORITY CHECK verifies whether a user is authorized for/DMO/TRVL" with the listed field values.
B. If the user is authorized for 'CNTRY = 'DE' AND for 'ACTVT = '03 then the return code is 0.
C. If the user is NOT authorized for 'CNTRY' = 'DE' OR for 'ACTVT' = '03 then the program will terminate.
D. If the user is authorized for 'CNTRY = 'DE' then the return code is always 0.
Answer: A,B (LEAVE A REPLY)

NEW QUESTION: 18

Why would you use Access Controls with CDS Views? Note: There are 2 correct answers to this question.

A. Only the data corresponding to the user's authorization is transferred from the database to the application layer.

B. The system field sy-subrc is set, giving you the result of the authorization check

C. You do not have to remember to implement AUTHORITY CHECK statements.

D. All of the data from the data sources is loaded into your application automatically and filtered there according to the user's authorization.

Answer: A,C (LEAVE A REPLY)

You would use Access Controls with CDS Views for the following reasons:

* A. Only the data corresponding to the user's authorization is transferred from the database to the application layer. This is true because Access Controls allow you to define CDS roles that specify the authorization conditions for accessing a CDS view. The CDS roles are evaluated for every user at runtime and the system automatically adds the restrictions to the selection conditions of the CDS view.

This ensures that only the data that the user is authorized to see is read from the database and transferred to the application layer. This improves the security and the performance of the data access1.

* C. You do not have to remember to implement AUTHORITY CHECK statements. This is true because Access Controls provide a declarative and centralized way of defining the authorization logic for a CDS view. You do not have to write any procedural code or use the AUTHORITY CHECK statement to check the user's authorization for each data source or field. The system handles the authorization check automatically and transparently for you2.

The following reasons are not valid for using Access Controls with CDS Views:

* B. The system field sy-subrc is set, giving you the result of the authorization check. This is false because the system field sy-subrc is not used by Access Controls. The sy-subrc field is used by the AUTHORITY CHECK statement to indicate the result of the authorization check, but Access Controls do not use this statement. Instead, Access Controls use CDS roles to filter the data according to the user's authorization2.

* D. All of the data from the data sources is loaded into your application automatically and filtered there according to the user's authorization. This is false because Access Controls do not load all the data from the data sources into the application layer. Access Controls filter the data at the database layer, where the data resides, and only transfer the data that the user is authorized to see to the application layer. This reduces the data transfer and the memory consumption of the application layer1.

References: 1: Access Controls | SAP Help Portal 2: ABAP CDS - Access Control - ABAP Keyword Documentation

NEW QUESTION: 19

For the assignment, gv_target = gv_source.

which of the following data declarations will always work without truncation or rounding? Note: There are 2 correct answers to this question.

A. DATA gv_source TYPE string, to DATA gv_target TYPE c.

B. DATA gv_source TYPE c. to DATA gv_target TYPE string.

C. DATA gv_source TYPE d. to DATA gv_target TYPE string.

D. DATA gv_source TYPE p LENGTH 8 DECIMALS 3. to DATA gv_target TYPE p LENGTH 16 DECIMALS 2.

Answer: (SHOW ANSWER)

The data declarations that will always work without truncation or rounding for the assignment gv_target = gv_source are B and C. This is because the target data type string is a variable-length character type that can hold any character string, including those of data types c (fixed-length character) and d (date). The assignment of a character or date value to a string variable will not cause any loss of information or precision, as the string variable will adjust its length to match the source value12.

You cannot do any of the following:

* A. DATA gv_source TYPE string, to DATA gv_target TYPE c.: This data declaration may cause truncation for the assignment gv_target = gv_source. This is because the target data type c is a fixed-length character type that has a predefined length. If the source value of type string is longer than the target length of type c, the source value will be truncated on the right to fit the target length12.

* D. DATA gv_source TYPE p LENGTH 8 DECIMALS 3. to DATA gv_target TYPE p LENGTH 16 DECIMALS 2.: This data declaration may cause rounding for the assignment gv_target = gv_source.

This is because the target data type p is a packed decimal type that has a predefined length and number of decimal places. If the source value of type p has more decimal places than the target type p, the source value will be rounded to the target number of decimal places12.

References: 1: ABAP Data Types - ABAP Keyword Documentation - SAP Online Help 2: ABAP Assignment Rules - ABAP Keyword Documentation - SAP Online Help

NEW QUESTION: 20

What are advantages of using a field symbol for internal table row access? Note: There are answers to this question.

A. The field symbol can be reused for other programs.

B. A MODIFY statement to write changed contents back to the table is not required.

C. The row content is copied to the field symbol instead to a work area

D. Using a field symbol is faster than using a work area.

Answer: B,D (LEAVE A REPLY)

A field symbol is a pointer that allows direct access to a row of an internal table without copying it to a work area. Using a field symbol for internal table row access has some advantages over using a work area, such as12:

* A MODIFY statement to write changed contents back to the table is not required: This is true. When you use a work area, you have to copy the row content from the internal table to the work area, modify it, and then copy it back to the internal table using the MODIFY statement. This can be costly in terms of performance and memory consumption. When you use a field symbol, you can modify the row content directly in the internal table without any copying. Therefore, you do not need the MODIFY statement12.
* Using a field symbol is faster than using a work area: This is true. As explained above, using a field
* symbol avoids the overhead of copying data between the internal table and the work area. This can improve the performance of the loop considerably, especially for large internal tables. According to some benchmarks, using a field symbol can save 25-40% of the runtime compared to using a work area12.
You cannot do any of the following:
* The field symbol can be reused for other programs: This is false. A field symbol is a local variable that is only visible within the scope of its declaration. It cannot be reused for other programs unless it is declared globally or passed as a parameter. Moreover, a field symbol must have the same type as the line type of the internal table that it accesses. Therefore, it cannot be used for any internal table with a different line type12.
* The row content is copied to the field symbol instead to a work area: This is false. As explained above, using a field symbol does not copy the row content to the field symbol. Instead, the field symbol points to the memory address of the row in the internal table and allows direct access to it. Therefore, there is no copying involved when using a field symbol12.
References: 1: Using Field Symbols to Process Internal Tables - SAP Learning 2: Access to Internal Tables - ABAP Keyword Documentation - SAP Online Help

NEW QUESTION: 21
You have two internal tables itab1 and itab2.What is true for using the expression itab1 = corresponding #(itab2)? Note: There are 2 correct answers to this question.
A. Fields with the same name but with different types may be copied from itab2 to itab1.
B. itab1 and itab2 must have at least one field name in common.
C. Fields with the same name and the same type will be copied from itab2 to itab1.
D. itab1 and itab2 must have the same data type.
Answer: B,C (LEAVE A REPLY)
The expression itab1 = corresponding #(itab2) is a constructor expression with the component operator CORRESPONDING that assigns the contents of the internal table itab2 to the internal table itab1. The following statements are true for using this expression:
* B: itab1 and itab2 must have at least one field name in common. This is because the component operator CORRESPONDING assigns the identically named columns of itab2 to the identically named columns of itab1 by default, according to the rules of MOVE-CORRESPONDING for internal tables. If itab1 and itab2 do not have any field name in common, the expression will not assign any value to itab1 and it will remain initial or unchanged1

* C: Fields with the same name and the same type will be copied from itab2 to itab1. This is because the component operator CORRESPONDING assigns the identically named columns of itab2 to the identically named columns of itab1 by default, according to the rules of MOVE-CORRESPONDING for
* internal tables. If the columns have the same name but different types, the assignment will try to perform a conversion between the types, which may result in a loss of precision, a truncation, or a runtime error, depending on the types involved1 The following statements are false for using this expression:
* A: Fields with the same name but with different types may be copied from itab2 to itab1. This is not true, as explained in statement C. The assignment will try to perform a conversion between the types, which may result in a loss of precision, a truncation, or a runtime error, depending on the types involved1
* D: itab1 and itab2 must have the same data type. This is not true, as the component operator CORRESPONDING can assign the contents of an internal table of one type to another internal table of a different type, as long as they have at least one field name in common. The target type of the expression is determined by the left-hand side of the assignment, which is itab1 in this case. The expression will create an internal table of the same type as itab1 and assign it to itab11 References: CORRESPONDING - Component Operator - ABAP Keyword Documentation

NEW QUESTION: 22
Exhibit:
With Icl_super being superclass for Icl_subl and Icl_sub2 and with methods subl_methl and sub2_methl being subclass-specific methods of Id_subl or Icl_sub2, respectivel. What will happen when executing these casts?
Note:
There are 2 correct answers to this question
A. go subl = CAST # go super), will not work
B. go_sub2 = CAST # go super), will work. go_subl CAST #go_super), will work
C. go_sub2 = CAST #(go_super). will not work.] go sub2->sub2 meth 1(...). will work
D. go_subl->subl_meth !(...)* w'll work.
Answer: (SHOW ANSWER)
The following are the explanations for each statement:
* A: This statement is correct. go_subl = CAST #(go_super) will not work. This is because go_subl is a data object of type REF TO cl_subl, which is a reference to the subclass cl_subl. go_super is a data object of type REF TO cl_super, which is a reference to the superclass cl_super. The CAST operator is used to perform a downcast or an upcast of a reference variable to another reference variable of a compatible type. A downcast is a conversion from a more general type to a more specific type, while an upcast is a conversion from a more specific type to a more general type. In this case, the CAST operator is trying to perform a downcast from go_super to go_subl, but this is not possible, as go_super is not pointing to an instance of cl_subl,

but to an instance of cl_super. Therefore, the CAST operator will raise an exception CX_SY_MOVE_CAST_ERROR at runtime12
* B: This statement is incorrect. go_sub2 = CAST #(go_super) will work. go_subl = CAST #(go_super) will not work. This is because go_sub2 is a data object of type REF TO cl_sub2, which is a reference to the subclass cl_sub2. go_super is a data object of type REF TO cl_super, which is a reference to the superclass cl_super. The CAST operator is used to perform a downcast or an upcast of a reference variable to another reference variable of a compatible type. A downcast is a conversion from a more general type to a more specific type, while an upcast is a conversion from a more specific type to a more general type. In this case, the CAST operator is trying to perform a downcast from go_super to go_sub2, and this is possible, as go_super is pointing to an instance of cl_sub2, which is a subclass of cl_super.
* Therefore, the CAST operator will assign the reference of go_super to go_sub2 without raising an exception. However, the CAST operator will not work for go_subl, as explained in statement A12
* C: This statement is incorrect. go_sub2 = CAST #(go_super) will work. go_sub2->sub2_meth1(...) will not work. This is because go_sub2 is a data object of type REF TO cl_sub2, which is a reference to the subclass cl_sub2. go_super is a data object of type REF TO cl_super, which is a reference to the superclass cl_super. The CAST operator is used to perform a downcast or an upcast of a reference variable to another reference variable of a compatible type. A downcast is a conversion from a more general type to a more specific type, while an upcast is a conversion from a more specific type to a more general type. In this case, the CAST operator is trying to perform a downcast from go_super to go_sub2, and this is possible, as go_super is pointing to an instance of cl_sub2, which is a subclass of cl_super.
Therefore, the CAST operator will assign the reference of go_super to go_sub2 without raising an exception. However, the method call go_sub2->sub2_meth1(...) will not work, as sub2_meth1 is a subclass-specific method of cl_sub2, which is not inherited by cl_super. Therefore, the method call will raise an exception CX_SY_DYN_CALL_ILLEGAL_METHOD at runtime123
* D: This statement is correct. go_subl->subl_meth1(...) will work. This is because go_subl is a data object of type REF TO cl_subl, which is a reference to the subclass cl_subl. subl_meth1 is a subclass-specific method of cl_subl, which is not inherited by cl_super. Therefore, the method call go_subl->subl_meth1(...) will work, as go_subl is pointing to an instance of cl_subl, which has the method subl_meth1123 References: NEW - ABAP Keyword Documentation, CAST - ABAP Keyword Documentation, Method Call - ABAP Keyword Documentation

NEW QUESTION: 23
In what order are objects created to generate a RESTful Application Programming application?
A. Database table 1
B. Service binding Projection view 4
C. Service definition 3
D. Data model view 2
E. D A B C

F. B D C A
G. A D C B
H. C B A B
Answer: (SHOW ANSWER)
The order in which objects are created to generate a RESTful Application Programming application is A, D, C, B. This means that the following steps are followed:
* First, a database table is created to store the data for the application. A database table is a CDS DDIC-based view that defines a join or union of database tables. A database table has an SQL view attached and can be accessed by Open SQL or native SQL.
* Second, a data model view is created to define a data model based on the database table or other CDS view entities. A data model view is a CDS view entity that can have associations, aggregations, filters, parameters, and annotations. A data model view can also define the behavior definition and implementation for the business object.
* Third, a service definition is created to define the service interface for the application. A service definition is a CDS view entity that defines a projection on a data model view or another service definition. A service definition can also define service metadata, such as service name, version, description, and annotations.
* Fourth, a service binding is created to define the service binding for the application. A service binding is a CDS view entity that defines a projection on a service definition. A service binding can also define the service protocol, such as OData V2, OData V4, or REST, and the service URL.
References: CDS Data Model Views - ABAP Keyword Documentation, CDS Service Definitions - ABAP Keyword Documentation, CDS Service Bindings - ABAP Keyword Documentation, CDS Projection Views - ABAP Keyword Documentation

NEW QUESTION: 24
Setting a field to read-only in which object would make the field read-only in all applications of the RESTful Application Programming model?
A. Service definition
B. Behaviour definition
C. Projection view
D. Metadata extension
Answer: (SHOW ANSWER)
The object that can be used to set a field to read-only in all applications of the RESTful Application Programming model (RAP) is the behaviour definition. The behaviour definition is a CDS artefact that defines the business logic and the UI behaviour of a business object. A business object is a CDS entity that represents a business entity or concept, such as a customer, an order, or a product. The behaviour definition can specify the properties of the fields of a business object, such as whether they are mandatory, read-only, or transient. These properties are valid for all applications that use the business object, such as transactional, analytical, or draft-enabled apps12. For example:

* The following code snippet defines a behaviour definition for a business object ZI_PB_APPLICATION.
It sets the field APPLICATION to read-only for all applications that use this business object: define behavior for ZI_PB_APPLICATION { field (read only) APPLICATION; ... } You cannot do any of the following:
* A. Service definition: A service definition is a CDS artefact that defines the interface and the binding of a service. A service is a CDS entity that exposes the data and the functionality of one or more business objects as OData, InA, or SQL services. A service definition can specify the properties of the fields of a service, such as whether they are filterable, sortable, or aggregatable. However, these properties are only valid for the specific service that uses the business object, not for all applications that use the business object12.
* C. Projection view: A projection view is a CDS artefact that defines a view on one or more data sources, such as tables, views, or associations. A projection view can select, rename, or aggregate the fields of the data sources, but it cannot change the properties of the fields, such as whether they are read-only or not. The properties of the fields are inherited from the data sources or the behaviour definitions of the business objects12.
* D. Metadata extension: A metadata extension is a CDS artefact that defines additional annotations for a CDS entity, such as a business object, a service, or a projection view. A metadata extension can specify the properties of the fields of a CDS entity for UI or analytical purposes, such as whether they are visible, editable, or hidden. However, these properties are only valid for the specific UI or analytical application that uses the metadata extension, not for all applications that use the CDS entity12.
References: 1: ABAP CDS - Data Definitions - ABAP Keyword Documentation - SAP Online Help 2: ABAP CDS - Behavior Definitions - ABAP Keyword Documentation - SAP Online Help

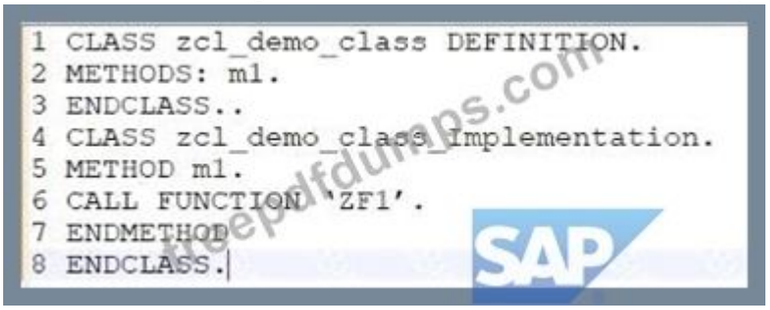
NEW QUESTION: 25
In this nested join below in which way is the join evaluated?

```
1  SELECT FROM t_a AS a
2      LEFT OUTER JOIN t_b AS b
3      LEFT OUTER JOIN t_c AS c
4      ON c~f1 = b~f1 AND c~f2 = b~f2
5      ON b~f1 = a~f1
6      WHERE ....
```

A. From the left to the right in the order of the tables:
1.
a is joined with b
2.
b is joined with c
B. From the right to the left in the order of the tables:
1.
b is joined with c.

2.
b is joined with a.
C. From the top to the bottom in the order of the on conditions
1.
b is joined with c
2.
a is joined with b
D. From the bottom to the top in the order of the on conditions:
1.
a is joined with b
2.
b is joined with c
Answer: C (LEAVE A REPLY)
The nested join is evaluated from the top to the bottom in the order of the ON conditions. This means that the join expression is formed by assigning each ON condition to the directly preceding JOIN from left to right.
The join expression can be parenthesized implicitly or explicitly to show the order of evaluation. In this case, the implicit parentheses are as follows:
SELECT * FROM (a INNER JOIN (b INNER JOIN c ON b~c = c~c) ON a~b = b~b) This means that the first join expression is b INNER JOIN c ON b~c = c~c, which joins the columns of tables b and c based on the condition that b~c equals c~c. The second join expression is a INNER JOIN (b INNER JOIN c ON b~c = c~c) ON a~b = b~b, which joins the columns of table a and the result of the first join expression based on the condition that a~b equals b~b. The final result set contains all combinations of rows from tables a, b, and c that satisfy both join conditions.
References: 1: SELECT, FROM JOIN - ABAP Keyword Documentation - SAP Online Help

NEW QUESTION: 26
Given the following code in an SAP S/4HANA Cloud private edition tenant:



The class zcl_demo_class is in a software component with the language version set to "ABAP Cloud". The function module ZF1' is in a different software component with the language version set to "Standard ABAP".
Both the class and function module are customer created.
Regarding line #6, which of the following are valid statements? Note: There are 2 correct answers to this question.

A. ZF1' can be called only if it is released for cloud development.
B. 'ZF1' can be called if a wrapper is created for it and the wrapper itself is released for cloud development.
C. "ZF1" can be called whether it is released or not for cloud development
D. ZF1" can be called if a wrapper is created for it but the wrapper itself is not released for cloud development.
Answer: (SHOW ANSWER)
The ABAP Cloud Development Model requires that only public SAP APIs and extension points are used to access SAP functionality and data. These APIs and extension points are released by SAP and documented in the SAP API Business Hub1. Customer-created function modules are not part of the public SAP APIs and are not released for cloud development. Therefore, calling a function module directly from an ABAP Cloud class is not allowed and will result in a syntax error. However, there are two possible ways to call a function module indirectly from an ABAP Cloud class:
* Create a wrapper class or interface for the function module and release it for cloud development. A wrapper is a class or interface that encapsulates the function module and exposes its functionality through public methods or attributes. The wrapper must be created in a software component with the language version set to "Standard ABAP" and must be marked as released for cloud development using the annotation @EndUserText.label. The wrapper can then be called from an ABAP Cloud class using the public methods or attributes2.
* Use the ABAP Cloud Connector to call the function module as a remote function call (RFC) from an ABAP Cloud class. The ABAP Cloud Connector is a service that enables the secure and reliable communication between SAP BTP, ABAP environment and on-premise systems. The function module must be exposed as an RFC-enabled function module in the on-premise system and must be registered in the ABAP Cloud Connector. The ABAP Cloud class can then use the class cl_rfc_destination_service to get the destination name and the class cl_abap_system to create a proxy object for the function module. The proxy object can then be used to call the function module3.
References: 1: SAP API Business Hub 2: Creating an ABAP Cloud Project | SAP Help Portal 3: Calling Remote Function Modules | SAP Help Portal

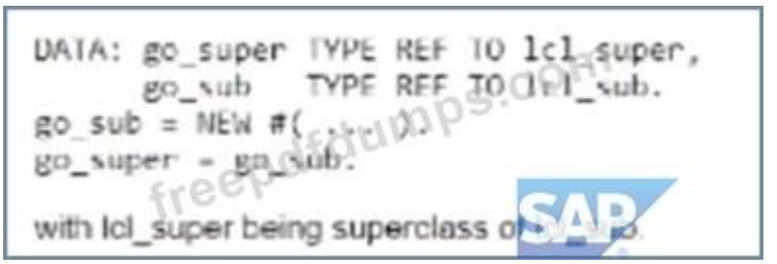
NEW QUESTION: 27
In the assignment, data (gv_result) = 1/8. what will be the data type of gv_result?
A. OTYPE I
B. TYPE DEFLOAT 16
C. TYPE P DECIMALS 3
D. TYPE P DECIMALS 2
Answer: B (LEAVE A REPLY)
The data type of gv_result in the assignment data (gv_result) = 1/8 will be TYPE DECFLOAT 16. This is because the assignment operator (=) in ABAP performs an implicit type conversion from the source type to the target type, according to the following rules12:

* If the target type is specified explicitly, the source value is converted to the target type.
* If the target type is not specified explicitly, the source type is used as the target type, unless the source type is a literal or an expression, in which case the target type is determined by the following priority order: DECFLOAT34, DECFLOAT16, P, F, I, C, N, X, STRING, XSTRING.
In this case, the target type is not specified explicitly, and the source type is an expression (1/8). Therefore, the target type is determined by the priority order, and the first matching type is DECFLOAT16, which is a decimal floating point type with 16 digits of precision12.
References: 1: ABAP Assignment Rules - ABAP Keyword Documentation - SAP Online Help 2: ABAP Data Types - ABAP Keyword Documentation - SAP Online Help

NEW QUESTION: 28
For what kind of applications would you consider using on-stack developer extensions? Note: There are 2 correct answers to this question.
A. Applications that provide APIs for side by side SAP BTP apps
B. Applications that access SAP S/4HANA data using complex SQL
C. Applications that integrate data from several different systems
D. Applications that run separate from SAP S/4HANA
Answer: A,B (LEAVE A REPLY)
On-stack developer extensibility is a type of extensibility that allows you to create development projects directly on the SAP S/4HANA Cloud technology stack. It gives you the opportunity to develop cloud-ready and upgrade-stable custom ABAP applications and services inside the SAP S/4HANA Cloud, public edition system. You can use the ABAP Development Tools in Eclipse to create and deploy your on-stack extensions.
On-stack developer extensibility is suitable for the following kinds of applications:
* Applications that provide APIs for side by side SAP BTP apps. On-stack developer extensibility allows you to create OData services or RESTful APIs based on CDS view entities or projection views. These services or APIs can expose SAP S/4HANA data and logic to other applications that run on the SAP Business Technology Platform (SAP BTP) or other platforms. This way, you can create a loosely coupled integration between your SAP S/4HANA system and your side by side SAP BTP apps.
* Applications that access SAP S/4HANA data using complex SQL. On-stack developer extensibility allows you to use ABAP SQL to access SAP S/4HANA data using complex queries, such as joins, aggregations, filters, parameters, and code pushdown techniques. You can also use ABAP SQL to perform data manipulation operations, such as insert, update, delete, and upsert. This way, you can create applications that require advanced data processing and analysis on SAP S/4HANA data.
The other kinds of applications are not suitable for on-stack developer extensibility, as they have different requirements and challenges. These kinds of applications are:
* Applications that integrate data from several different systems. On-stack developer extensibility is not meant for creating applications that integrate data from multiple sources, such as other SAP systems, third-party systems, or cloud services. This is because on-stack developer extensibility

does not support remote access or data replication, and it may cause performance or security issues. For this kind of applications, you should use side by side extensibility, which allows you to create applications that run on the SAP BTP and communicate with the SAP S/4HANA system via public APIs or events.
* Applications that run separate from SAP S/4HANA. On-stack developer extensibility is not meant for creating applications that run independently from the SAP S/4HANA system, such as standalone apps, microservices, or web apps. This is because on-stack developer extensibility requires a tight coupling with the SAP S/4HANA system, and it may limit the scalability, flexibility, and portability of the applications. For this kind of applications, you should use side by side extensibility, which allows you to create applications that run on the SAP BTP and leverage the cloud-native features and services of the platform.
References: Developer Extensibility in SAP S/4HANA Cloud ABAP Environment, SAP S/4HANA Extensibility - Simplified Guide for Beginners

NEW QUESTION: 29
What are some properties of database tables? Note: There are 2 correct answers to this question.
A. They store information in two dimensions.
B. They may have key fields.
C. They can have any number of key fields.
D. They can have relationships to other tables.
Answer: (SHOW ANSWER)
Database tables are data structures that store information in two dimensions, using rows and columns. Each row represents a record or an entity, and each column represents an attribute or a field. Database tables may have key fields, which are columns that uniquely identify each row or a subset of rows. Key fields can be used to enforce data integrity, perform efficient searches, and establish relationships to other tables. Database tables can have relationships to other tables, which are associations or links between the key fields of two or more tables. Relationships can be used to model the logical connections between different entities, join data from multiple tables, and enforce referential integrity12.
References: 1: Table (database) - Wikipedia 2: Database design basics - Microsoft Support

NEW QUESTION: 30


When accessing the subclass instance through go_super, what can you do? Note: There are 2 correct answers to this question.
A. Access the inherited private components.

B. Access the inherited public components.
C. Call a subclass specific public method
D. Call inherited public redefined methods.
Answer: (SHOW ANSWER)
When accessing the subclass instance through go_super, you can do both of the following:
* Access the inherited private components: A subclass inherits all the private attributes and methods of its superclass, unless they are explicitly overridden by the subclass. Therefore, you can access the inherited private components of the superclass through go_super, as long as they are not hidden by other attributes or methods in the subclass12.
* Access the inherited public components: A subclass inherits all the public attributes and methods of its superclass, unless they are explicitly overridden by the subclass. Therefore, you can access the inherited public components of the superclass through go_super, as long as they are not hidden by other attributes or methods in the subclass12.
You cannot do any of the following:
* Call a subclass specific public method: A subclass does not have any public methods that are not inherited from its superclass. Therefore, you cannot call a subclass specific public method through go_super12.
* Call inherited public redefined methods: A subclass does not have any public methods that are redefined from its superclass. Therefore, you cannot call inherited public redefined methods through go_super12.
References: 1: Object Oriented - ABAP Development - Support Wiki 2: Inheritance and Instantiation - ABAP Keyword Documentation

NEW QUESTION: 31
In ABAP SQL, which of the following retrieves the association field_Airline-Name of a CDS view?
A. _Airline-Name
B. /_Airline Name
C. @_Airline-Name
D. "_Airline Name
Answer: (SHOW ANSWER)
In ABAP SQL, the syntax to retrieve the association field of a CDS view is to use the @ sign followed by the association name and the field name, separated by a period sign (.). For example, to retrieve the association field _Airline-Name of a CDS view, the syntax is @_Airline.Name. This syntax allows the access to the fields of the target data source of the association without explicitly joining the data sources1. The other options are incorrect because they use the wrong symbols or formats to access the association field.
References: 1: Path Expressions - ABAP Keyword Documentation

NEW QUESTION: 32

You are given the following information:

```
1  SELECT SINGLE *
2  FROM SPFLI
3  WHERE CARRID = 'LH' AND CONNID = '1234'
4  INTO @data(ls)
```

1.
The data source "spfli" on line #2 is an SAP HANA database table
2.
"spfli" will be a large table with over one million rows.
3.
This program is the only one in the system that accesses the table.
4.
This program will run rarely.
Based on this information, which of the following general settings should you set for the spfli database table? Note: There are 2 correct answers to this question.

A. "Storage Type" to "Column Store"
B. "Load Unit to "Column Loadable"
C. "Storage Type" to "Row Store"
D. "Load Unit' to 'Page Loadable"

Answer: C,D (LEAVE A REPLY)

Based on the given information, the spfli database table should have the following general settings:

* "Storage Type" to "Row Store": This setting determines how the data is stored in the SAP HANA database. Row store is suitable for tables that are accessed by primary key or by a small number of columns. Column store is suitable for tables that are accessed by a large number of columns or by complex analytical queries. Since the spfli table is a large table with over one million rows, and this program is the only one in the system that accesses the table, it is likely that the program will use primary key access or simple queries to access the table. Therefore, row store is a better choice than column store for this table12.

* "Load Unit" to "Page Loadable": This setting determines how the data is loaded into the memory when the table is accessed. Page loadable means that the data is loaded in pages of 16 KB each, and only the pages that are needed are loaded. Column loadable means that the data is loaded in columns, and only the columns that are needed are loaded. Since the spfli table is a row store table, and this program will run rarely, it is more efficient to use page loadable than column loadable for this table. Page loadable will reduce the memory consumption and the loading time of the table13.
References: 1: Table Types in SAP HANA | SAP Help Portal 2: [Row Store vs Column Store in SAP HANA | SAP Blogs] 3: [Load Unit | SAP Help Portal]

NEW QUESTION: 33
In a RESTful Application Programming application, in which objects do you bind a CDS view to create a value help? Note: There are 3 correct answers to this question.
A. Data model view
B. Behavior definition
C. Metadata Extension
D. Service Definition
E. Projection View
Answer: A,C,E (LEAVE A REPLY)
In a RESTful Application Programming (RAP) application, you can bind a CDS view to create a value help in the following objects:
* Data model view: A data model view is a CDS view that defines the data structure and the associations of an entity in the RAP application. You can use the annotation @Consumption.valueHelpDefinition to bind a value help provider CDS view to an element of the data model view. The value help provider CDS view must contain the key fields of the value help entity and the fields that are displayed in the value help dialog. The value help annotation specifies the entity name, the element name, and optionally the additional binding conditions for the value help provider1.
* Metadata Extension: A metadata extension is a CDS view that extends the metadata of another CDS view without changing its data structure. You can use the annotation @MetadataExtension.extendView to specify the target CDS view that you want to extend. You can then use the same annotation @Consumption.valueHelpDefinition to bind a value help provider CDS view to an element of the target CDS view. The metadata extension allows you to add value help definitions to existing CDS views without modifying them2.
* Projection View: A projection view is a CDS view that defines the projection of another CDS view. You can use the annotation @AbapCatalog.sqlViewType: #PROJECTION to specify that the CDS view is a projection view. You can then use the same annotation @Consumption.valueHelpDefinition to bind a value help provider CDS view to an element of the projection view. The projection view allows you to add value help definitions to projected elements of another CDS view3.

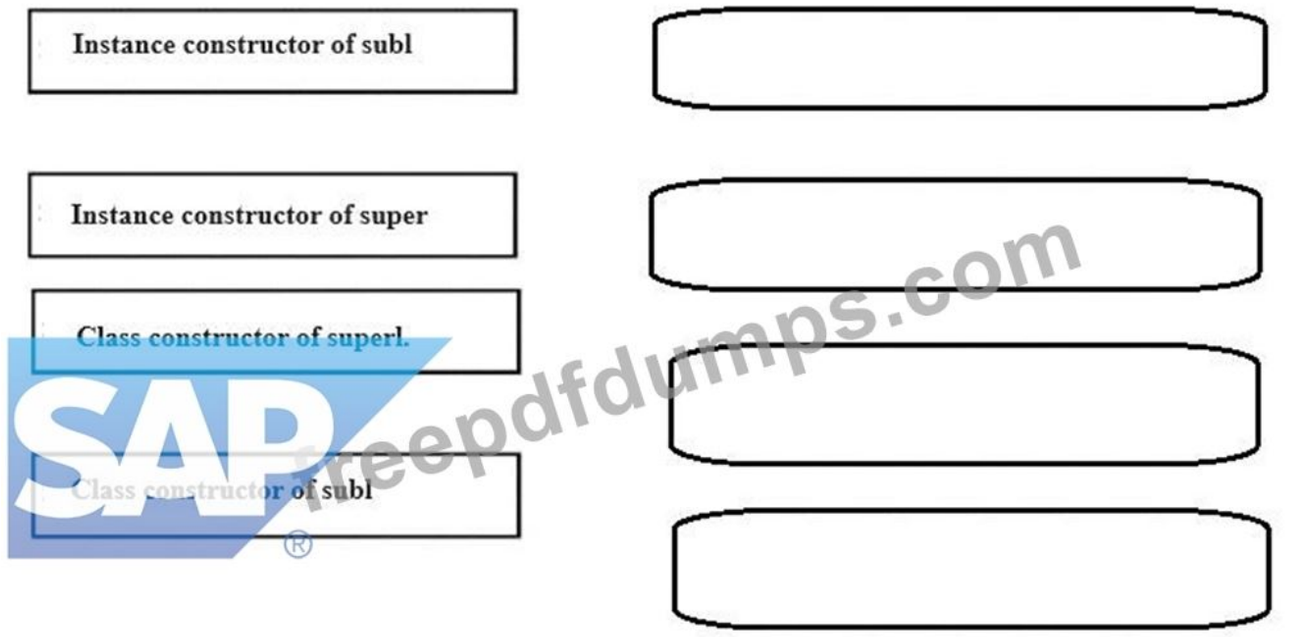
You cannot bind a value help provider CDS view to a behavior definition or a service definition, because these objects do not define the data structure or the metadata of an entity in the RAP application. A behavior definition defines the behavior and the validation rules of an entity, such as the create, read, update, and delete (CRUD) operations, the draft handling, the authorization checks, and the side effects4. A service definition defines the service exposure and the service binding of an entity, such as the protocol, the version, the namespace, and the service name5. References: 1: Value Help with Additional Binding | SAP Help Portal 2: Metadata Extensions - ABAP Keyword Documentation 3: Projection Views - ABAP Keyword Documentation 4: Behavior Definition - ABAP Keyword Documentation 5: Service Definition - ABAP Keyword Documentation

NEW QUESTION: 34
You want to define the following CDS view entity with an input parameter:
Define view entity Z_CONVERT With parameters currency : ???
Which of the following can you use to replace "???? Note: There are 2 correct answers to this question.
A. built-in ABAP type
B. A built-in ABAP Dictionary type
C. A data element
D. A component of an ABAP Dictionary structure
Answer: (SHOW ANSWER)
The possible replacements for "???" in the CDS view entity definition with an input parameter are A. built-in ABAP type and C. A data element. These are the valid types that can be used to specify the data type of an input parameter in a CDS view entity. A built-in ABAP type is a predefined elementary type in the ABAP language, such as abap.char, abap.numc, abap.dec, etc. A data element is a reusable semantic element in the ABAP Dictionary that defines the technical attributes and the meaning of a field12. For example:
* The following code snippet defines a CDS view entity with an input parameter currency of type abap.cuky, which is a built-in ABAP type for currency key:
Define view entity Z_CONVERT With parameters currency : abap.cuky as select from ... { ... }
* The following code snippet defines a CDS view entity with an input parameter currency of type waers, which is a data element for currency key:
Define view entity Z_CONVERT With parameters currency : waers as select from ... { ... } You cannot do any of the following:
* B. A built-in ABAP Dictionary type: This is not a valid type for an input parameter in a CDS view entity. A built-in ABAP Dictionary type is a predefined elementary type in the ABAP Dictionary, such as CHAR, NUMC, DEC, etc. However, these types cannot be used directly in a CDS view entity definition. Instead, they have to be prefixed with abap. to form a built-in ABAP type, as explained above12.
* D. A component of an ABAP Dictionary structure: This is not a valid type for an input parameter in a CDS view entity. A component of an ABAP Dictionary structure is a field that belongs to a structure type, which is a complex type that consists of multiple fields. However, an input

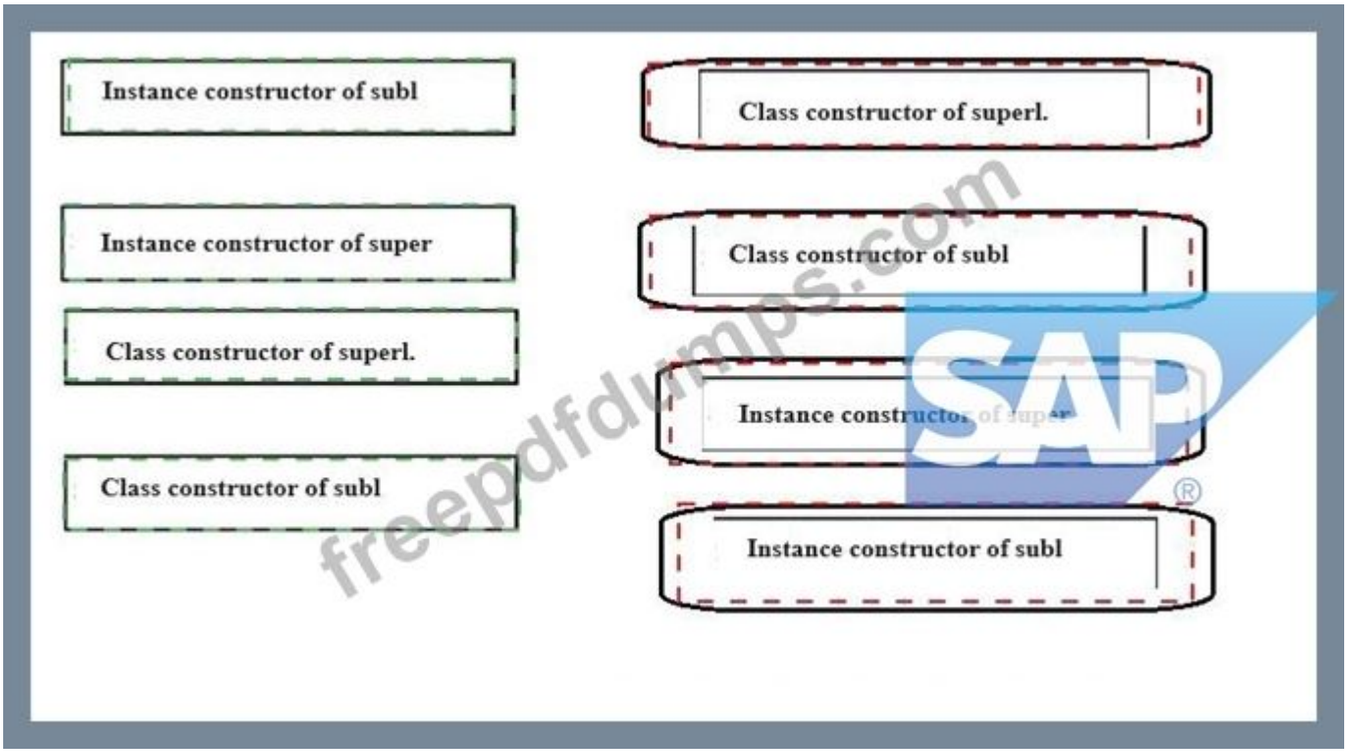
parameter in a CDS view entity can only be typed with an elementary type, which is a simple type that has no internal structure12.
References: 1: ABAP CDS - SELECT, parameter_list - ABAP Keyword Documentation - SAP Online Help 2:
ABAP Data Types - ABAP Keyword Documentation - SAP Online Help

NEW QUESTION: 35
Which restrictions exist for ABAP SQL arithmetic expressions? Note: There are 2 correct answers to this question.
A. Floating point types and integer types can NOT be used in the same expression.
B. The operator/is allowed only in floating point expressions.
C. Decimal types and integer types can NOT be used in the same expression.
D. The operator is allowed only in floating point expressions.
Answer: B,D (LEAVE A REPLY)
ABAP SQL arithmetic expressions have different restrictions depending on the data type of the operands. The following are some of the restrictions:
* Floating point types and integer types can be used in the same expression, as long as the integer types are cast to floating point types using the cast function. For example, CAST (num1 AS FLTP) / CAST (num2 AS FLTP) is a valid expression, where num1 and num2 are integer types.
* The operator / is allowed only in floating point expressions, where both operands have the type FLTP or f. For example, num1 / num2 is a valid expression, where num1 and num2 are floating point types. If the operator / is used in an integer expression or a decimal expression, a syntax error occurs.
* Decimal types and integer types can be used in the same expression, as long as the expression is a decimal expression. A decimal expression has at least one operand with the type DEC, CURR, or QUAN or p with decimal places. For example, num1 + num2 is a valid expression, where num1 is a decimal type and num2 is an integer type.
* The operator ** is allowed only in floating point expressions, where both operands have the type FLTP or f. For example, num1 ** num2 is a valid expression, where num1 and num2 are floating point types.
If the operator ** is used in an integer expression or a decimal expression, a syntax error occurs.
References: sql_exp - sql_arith - ABAP Keyword Documentation, SQL Expressions, Arithmetic Calculations - ABAP Keyword Documentation

NEW QUESTION: 36
You have a superclass superl and a subclass subl of superl. Each class has an instance constructor and a static constructor. The first statement of your program creates an instance of subl. In which sequence will the constructors be executed?



Answer:



Explanation:

The sequence in which the constructors will be executed is as follows:

* Class constructor of superl. This is because the class constructor is a static method that is executed automatically before the class is accessed for the first time. The class constructor is used to initialize the static attributes and components of the class. The class constructor of the superclass is executed before the class constructor of the subclass, as the subclass inherits the static components of the superclass12

* Class constructor of subl. This is because the class constructor is a static method that is executed automatically before the class is accessed for the first time. The class constructor is used to initialize the static attributes and components of the class. The class constructor of the subclass is executed after the class constructor of the superclass, as the subclass inherits the static components of the superclass12
* Instance constructor of superl. This is because the instance constructor is an instance method that is executed automatically when an instance of the class is created using the statement CREATE OBJECT.
The instance constructor is used to initialize the instance attributes and components of the class. The instance constructor of the superclass is executed before the instance constructor of the subclass, as the subclass inherits the instance components of the superclass. The instance constructor of the subclass must call the instance constructor of the superclass explicitly using super->constructor, unless the superclass is the root node object12
* Instance constructor of subl. This is because the instance constructor is an instance method that is executed automatically when an instance of the class is created using the statement CREATE OBJECT.
The instance constructor is used to initialize the instance attributes and components of the class. The instance constructor of the subclass is executed after the instance constructor of the superclass, as the subclass inherits the instance components of the superclass. The instance constructor of the subclass must call the instance constructor of the superclass explicitly using super->constructor, unless the superclass is the root node object12 References: Constructors of Classes - ABAP Keyword Documentation, METHODS - constructor - ABAP Keyword Documentation

NEW QUESTION: 37
When processing a loop with the statement DO... ENDDO, what system variable contains the implicit loop counter?
A. sy-linno
B. sy-labix
C. sy-subrc
D. sy-index
Answer: (SHOW ANSWER)
When processing a loop with the statement DO... ENDDO, the system variable that contains the implicit loop counter is sy-index. The loop counter is a numeric value that indicates how many times the loop has been executed. The loop counter is initialized to 1 before the first execution of the loop and is incremented by 1 after each execution. The loop counter can be used to control the number of loop iterations or to access the loop elements by index. The loop counter can also be accessed or modified within the loop body, but this is not recommended as it may cause unexpected results or errors1.
For example, the following code snippet uses the loop counter sy-index to display the numbers from 1 to 10:

DO 10 TIMES. WRITE: / sy-index. ENDDO.
The output of this code is:
1 2 3 4 5 6 7 8 9 10
References: 1: DO - ABAP Keyword Documentation

NEW QUESTION: 38
What RESTful Application Programming object contains only the fields required for a particular app?
A. Database view
B. Metadata extension
C. Projection View
D. Data model view
Answer: C (LEAVE A REPLY)
A projection view is a RESTful Application Programming object that contains only the fields required for a particular app. A projection view is a CDS view entity that defines a projection on an existing CDS view entity or CDS DDIC-based view. A projection view exposes a subset of the elements of the projected entity, which are relevant for a specific business service. A projection view can also define aliases, virtual elements, and annotations for the projected elements. A projection view is the top-most layer of a CDS data model and prepares data for a particular use case. A projection view can have different provider contracts depending on the type of service it supports, such as transactional query, analytical query, or transactional interface.
A database view is a CDS DDIC-based view that defines a join or union of database tables. A database view has an SQL view attached and can be accessed by Open SQL or native SQL. A database view can be used as a projected entity for a projection view, but it does not contain only the fields required for a particular app.
A metadata extension is a RESTful Application Programming object that defines additional annotations for a CDS view entity or a projection view. A metadata extension can be used to enhance the metadata of a CDS data model without changing the original definition. A metadata extension does not contain any fields, but only annotations.
A data model view is a CDS view entity that defines a data model based on database tables or other CDS view entities. A data model view can have associations, aggregations, filters, parameters, and annotations. A data model view can be used as a projected entity for a projection view, but it does not contain only the fields required for a particular app.
References: CDS Projection Views - ABAP Keyword Documentation, CDS Projection Views in ABAP CDS:
What's Your Flavor, Business Object Projection - ABAP Keyword Documentation

NEW QUESTION: 39
What are some of the reasons that Core Data Services are preferable to the classical approach to data modeling? Note: There are 2 correct answers to this question.
A. They implement code pushdown.

B. They avoid data transfer completely.
C. They transfer computational results to the application server.
D. They compute results on the application server.
Answer: A,C (LEAVE A REPLY)
Core Data Services (CDS) are preferable to the classical approach to data modeling for several reasons, but two of them are:
* They implement code pushdown. Code pushdown is the principle of moving data-intensive logic from the application server to the database server, where the data resides. This reduces the data transfer between the application server and the database server, which improves the performance and scalability of the application. CDS enable code pushdown by allowing the definition of semantic data models and business logic in the database layer, using SQL and SQL-based expressions1.
* They transfer computational results to the application server. CDS allow the application server to access the data and the logic defined in the database layer by using Open SQL statements. Open SQL is a standardized and simplified subset of SQL that can be used across different database platforms. Open SQL statements are translated into native SQL statements by the ABAP runtime environment and executed on the database server. The results of the computation are then transferred to the application server, where they can be further processed or displayed2.
References: 1: ABAP - Core Data Services (ABAP CDS) - ABAP Keyword Documentation 2: Open SQL - ABAP Keyword Documentation