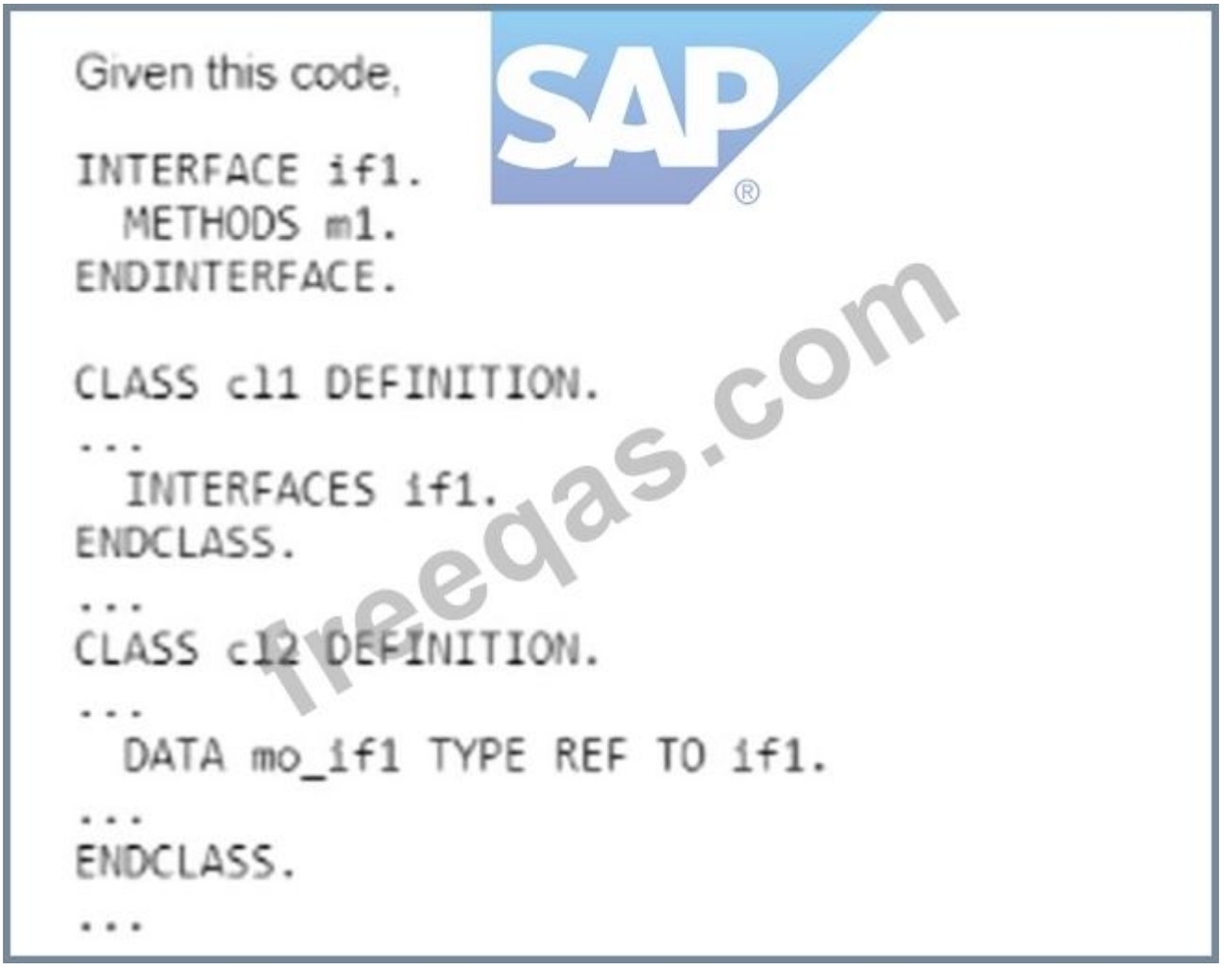
SAP.C_ABAPD_2309.v2024-07-26.q65

Exam Code:	C_ABAPD_2309
Exam Name:	SAP Certified Associate - Back-End Developer - ABAP Cloud
Certification Provider:	SAP
Free Question Number:	65
Version:	v2024-07-26
# of views:	354
# of Questions views:	650
https://www.freeqas.com/qa/SAP/C_ABAPD_2309/SAP.C_ABAPD_2309.v2024-07-26.q65.html	

NEW QUESTION: 1



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: (SHOW ANSWER)
Explanation
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 interfacemethod 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: 2
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.

NEW QUESTION: 3

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)

Explanation

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: 4
Which of the following are ABAP Cloud Development Model rules?
Note: There are 2 correct answers to this question.
A. Use public SAP APIs and SAP extension points.
B. Build ABAP RESTful application programming model-based services.
C. Reverse modifications when a suitable public SAP API becomes available.
D. Build ABAP reports with either ABAP List Viewer (ALV) or SAP Fiori.
Answer: A (LEAVE A REPLY)
Use public SAP APIs and SAP extension points. This rule ensures that the ABAP Cloud code is stable, reliable, and compatible with the SAP solutions and the cloud operations. Public SAP APIs and SAP extension points are the only allowed interfaces and objects to access the SAP platform and the SAP applications. They are documented, tested, and supported by SAP. They also guarantee the lifecycle stability and the upgradeability of the ABAP Cloud code1.
Build ABAP RESTful application programming model-based services. This rule ensures that the ABAP Cloud code follows the state-of-the-art development paradigm for building cloud-ready

business services. The ABAP RESTful application programming model (RAP) is a framework that provides a consistent end-to-end programming model for creating, reading, updating, and deleting (CRUD) business data. RAP also supports draft handling, authorization checks, side effects, validations, and custom actions. RAP exposes the business services as OData services that can be consumed by SAP Fiori apps or other clients2.

NEW QUESTION: 5

Image:

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

A.

```
SELECT FROM dbtab1 FIELDS F1,
CASE
WHEN F2 = '1' THEN 'Value 1
WHEN f2'2' THEN 'Value 2' ELSE "Value for the rest' END AS f case
INTO TABLE @et t1.
```


B.

```
SELECT FROM dbta  LDS F1,
CASE f2,
WHEN '1' THEN 'Value 1',
WHEN '2' THEN 'Value 2',
WHEN OTHERS "Value for the rest", ENDCASE AS f_case
INTO TABLE @gt t1.
```

C.

```
SELECT FROM dbtab1 FIELDS f1,
CASE f2
WHEN '1' THEN 'Value 1'
WHEN '2' THEN 'Value 2'
ELSE "Value for the rest' END AS f_case
INTO TABLE @gt_t1.
```

D.

```
SELECT FROM dbtab1 FIELDS F1,
CASE
WHEN F2 = '1' THEN "Value 1' WHEN f2 < f3 AND f2 = '2' THEN "Value 2'
WHEN OTHERS 'Value for the rest' ENDCASE AS f_case
INTO TABLE @ut t1.
```

Answer: (SHOW ANSWER)

NEW QUESTION: 6

Refer to the Exhibit.

Given this code,

```
INTERFACE if1.
  METHODS m1.
ENDINTERFACE.

CLASS cl1 DEFINITION.
...
  INTERFACES if1.
ENDCLASS.
...
CLASS cl2 DEFINITION.
...
  DATA mo_if1 TYPE REF TO if1.
...
ENDCLASS.
...
```

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: C,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

NEW QUESTION: 7
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: C (LEAVE A REPLY)
Explanation
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: 8
What would be the correct expression to change a given string value 'mr joe doe' into 'JOE' in an ABAP SQL field list?
A. SELECT FROM TABLE dbtabl FIELDS
Of1,
upper(left('mr joe doe', 6)) AS f2_up_left, f3,
B. SELECT FROM TABLE dbtabl FIELDS
Of1,
left(lower(substring('mr joe doe', 4, 3)), 3) AS f2_left_lo_sub, f3,
C. SELECT FROM TABLE dbtabl FIELDS
Of1,
substring(upper('mr joe doe'), 4, 3) AS f2_sub_up, f3,...
D. SELECT FROM TABLE dbtabl FIELDS
Of1,
substring(lower(upper('mr joe doe')), 4, 3) AS f2_sub_lo_up, f3,

Answer: C (LEAVE A REPLY)

The correct expression to change a given string value 'mr joe doe' into 'JOE' in an ABAP SQL field list is C. SELECT FROM TABLE dbtabl FIELDS Of1, substring(upper('mr joe doe'), 4, 3) AS f2_sub_up, f3,... This expression uses the following SQL functions for strings12:

upper: This function converts all lowercase characters in a string to uppercase. For example, upper('mr joe doe') returns 'MR JOE DOE'.

substring: This function returns a substring of a given string starting from a specified position and with a specified length. For example, substring('MR JOE DOE', 4, 3) returns 'JOE'.

AS: This keyword assigns an alias or a temporary name to a field or an expression in the field list. For example, AS f2_sub_up assigns the name f2_sub_up to the expression substring(upper('mr joe doe'), 4, 3).

You cannot do any of the following:

A) SELECT FROM TABLE dbtabl FIELDS Of1, upper(left('mr joe doe', 6)) AS f2_up_left, f3,...: This expression uses the wrong SQL function for strings to get the desired result. The left function returns the leftmost characters of a string with a specified length, ignoring the trailing blanks. For example, left('mr joe doe', 6) returns 'mr joe'. Applying the upper function to this result returns 'MR JOE', which is not the same as 'JOE'.

B) SELECT FROM TABLE dbtabl FIELDS Of1, left(lower(substring('mr joe doe', 4, 3)), 3) AS f2_left_lo_sub, f3,...: This expression uses unnecessary and incorrect SQL functions for strings to get the desired result. The lower function converts all uppercase characters in a string to lowercase. For example, lower(substring('mr joe doe', 4, 3)) returns 'joe'. Applying the left function to this result with the same length returns 'joe' again, which is not the same as 'JOE'.

D) SELECT FROM TABLE dbtabl FIELDS Of1, substring(lower(upper('mr joe doe')), 4, 3) AS f2_sub_lo_up, f3,...: This expression uses unnecessary and incorrect SQL functions for strings to get the desired result. The lower function converts all uppercase characters in a string to lowercase, and the upper function converts all lowercase characters in a string to uppercase. Applying both functions to the same string cancels out the effect of each other and returns the original string. For example, lower(upper('mr joe doe')) returns 'mr joe doe'. Applying the substring function to this result returns 'joe', which is not the same as 'JOE'.

NEW QUESTION: 9

In this nested join below in which way is the join evaluated?

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)

Explanation

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: 10
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)
Explanation
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

NEW QUESTION: 11
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(It_hits)
B. SELECT FROM /dmo/connection FIELDS \/ O carrid, airpfrom,
MAX(distance) AS dist_max, MIN(distance) AS dist_min INTO TABLE @DATA(It_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)
Explanation
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 lt_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 lt_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: 12
Refer to the Exhibit.
Image:


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

A.

```
SELECT FROM dbtab1 FIELDS F1,
CASE
WHEN F2 = '1' THEN 'Value 1
WHEN f2'2' THEN 'Value 2' ELSE "Value for the rest' END AS f case
INTO TABLE @et t1.
```

B.

```
SELECT FROM dbtab1 FIELDS f1,
CASE f2
WHEN '1' THEN 'Value 1'
WHEN '2' THEN 'Value 2'
ELSE "Value for the rest' END AS f_case
INTO TABLE @gt_t1.
```

C.

```
SELECT FROM dbtab1 FIELDS F1,
CASE f2,
WHEN '1' THEN 'Value 1',
WHEN '2' THEN 'Value 2',
WHEN OTHERS "Value for the rest", ENDCASE AS f_case
INTO TABLE @gt t1.
```

D.

```
SELECT FROM dbtab1 FIELDS F1,
CASE
WHEN F2 = '1' THEN "Value 1' WHEN f2 < f3 AND f2 = '2' THEN "Value 2'
WHEN OTHERS 'Value for the rest' ENDCASE AS f_case
INTO TABLE @gt t1.
```

Answer: A,B (LEAVE A REPLY)

NEW QUESTION: 13

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)
Explanation
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: 14
Given the following Core Data Service View Entity Data Definition:
1 @AccessControl.authorizationCheck: #NOT_REQUIRED
2 DEFINE VIEW ENTITY demo_flight_info_join
3 AS SELECT
4 FROM scarr AS a
5 LEFT OUTER JOIN scounter AS c
6 LEFT OUTER JOIN sairport AS p
7 ON p.id = c.airport
8 ON a.carrid = c.carrid
9 {

10 a.carrid AS carrier_id,
11 p.id AS airport_id,
12 c.countnum AS counter_number
13 }
In what order will the join statements be executed?
A. scarr will be joined with scounter first and the result will be joined with sairport.
B. sairport will be joined to scounter first and the result will be joined with scarr.
C. scarr will be joined with sairport first and the result will be joined with scounter.
D. scounter will be joined to sairport first and the result will be joined with scarr.
Answer: A (LEAVE A REPLY)
The order in which the join statements will be executed is:
scarr will be joined with scounter first and the result will be joined with sairport.
This is because the join statements are nested from left to right, meaning that the leftmost data source is joined with the next data source, and the result is joined with the next data source, and so on. The join condition for each pair of data sources is specified by the ON clause that follows the data source name. The join type for each pair of data sources is specified by the join operator that precedes the data source name. In this case, the join operator is LEFT OUTER JOIN, which means that all the rows from the left data source are included in the result, and only the matching rows from the right data source are included. If there is no matching row from the right data source, the corresponding fields are filled with initial values1.
Therefore, the join statements will be executed as follows:
First, scarr AS a will be joined with scounter AS c using the join condition a.carrid = c.carrid. This means that all the rows from scarr will be included in the result, and only the rows from scounter that have the same value for the carrid field will be included. If there is no matching row from scounter, the countnum field will be filled with an initial value.
Second, the result of the first join will be joined with sairport AS p using the join condition p.id = c.airport. This means that all the rows from the first join will be included in the result, and only the rows from sairport that have the same value for the id field as the airport field from the first join will be included. If there is no matching row from sairport, the id field will be filled with an initial value.

NEW QUESTION: 15
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: A,D (LEAVE A REPLY)
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

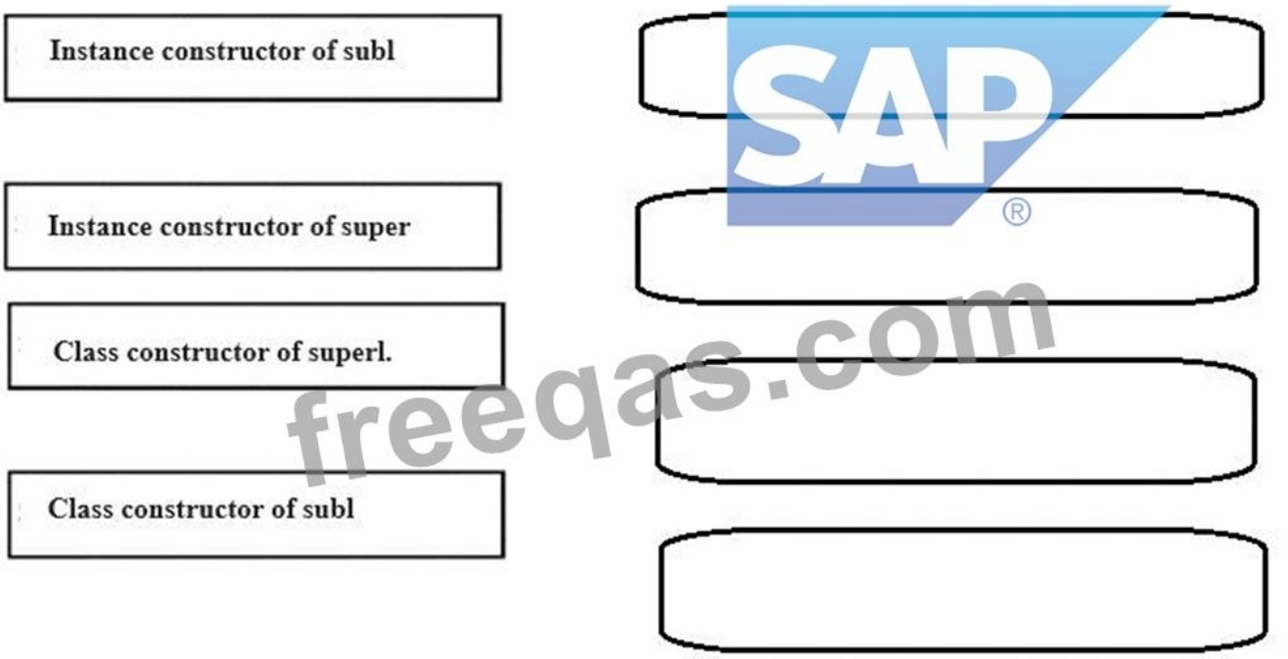
subset of rows. Key fields can be used to enforce data integrity, perform efficient searches, and establish relationships to other tables. Database tables can have relationships to other tables, which are associations or links between the key fields of two or more tables. Relationships can be used to model the logical connections between different entities, join data from multiple tables, and enforce referential integrity12.

NEW QUESTION: 16
Refer to the Exhibit.

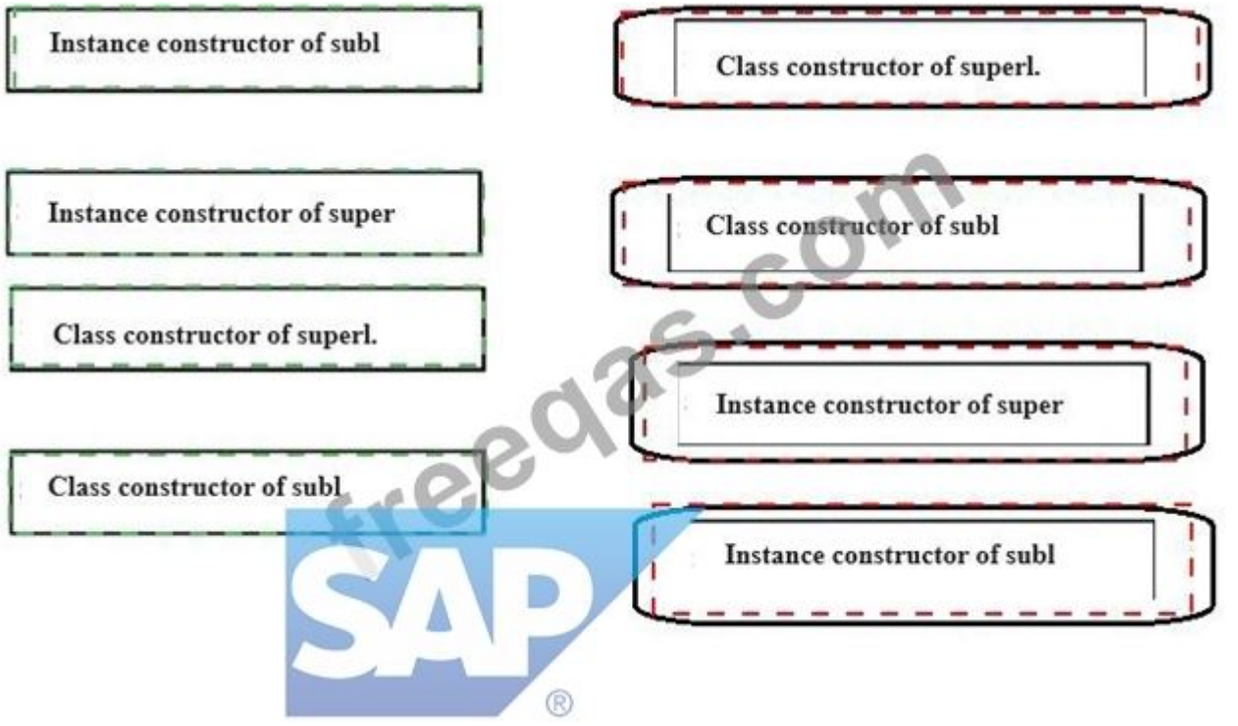
The class zcl_demo_class is in a software component with the language version set to "Standard ABAP". The function module "ZF11 is in a software component with the language version set to "ABAP Cloud". Both the class and function module are customer created. Regarding line #6, which of the following is a valid statement?
A. 'ZF1' can be called whether it has been released or not for cloud development.
B. 'ZF1' can be called via a wrapper that itself has been released for cloud development.
C. 'ZF1' can be called via a wrapper that itself has not been released for cloud development.
D. 'ZF1' must be released for cloud development to be called.
Answer: (SHOW ANSWER)
The function module ZF1 is in a software component with the language version set to "ABAP Cloud". This means that it follows the ABAP Cloud Development Model, which requires the usage of public SAP APIs and extension points 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 a class with the language version set to "Standard ABAP" is not allowed and will result in a syntax error. However, there is a possible way to call a function module indirectly from a class with the language version set to "Standard ABAP":
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 "ABAP Cloud" and must be marked as released for cloud development using the annotation @EndUserText.label. The wrapper can then be called from a class with the language version set to "Standard ABAP" using the public methods or attributes2.

For example, the following code snippet shows how to create a wrapper class for the function module ZF1 and call it from the class zcl_demo_class:
@EndUserText.label: 'Wrapper for ZF1' CLASS zcl_wrapper_zf1 DEFINITION PUBLIC FINAL CREATE PUBLIC. PUBLIC SECTION. CLASS-METHODS: call_zf1 IMPORTING iv_a TYPE i iv_b TYPE i EXPORTING ev_result TYPE i. ENDCLASS.
CLASS zcl_wrapper_zf1 IMPLEMENTATION. METHOD call_zf1. CALL FUNCTION 'ZF1' EXPORTING a = iv_a b = iv_b IMPORTING result = ev_result. ENDMETHOD. ENDCLASS.
CLASS zcl_demo_class DEFINITION. METHODS: m1. ENDCLASS.
CLASS zcl_demo_class IMPLEMENTATION. METHOD m1. DATA(lv_result) = zcl_wrapper_zf1=>call_zf1(iv_a = 2 iv_b = 3). WRITE: / lv_result. ENDMETHOD. ENDCLASS.
The output of this code is:
5

NEW QUESTION: 17
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: 18
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: D (LEAVE A REPLY)
Explanation
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: 19
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)
Explanation
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 Help2:ABAP Data Types - ABAP Keyword Documentation - SAP Online Help

NEW QUESTION: 20
Which patterns raise an exception? Note: There are 3 correct answers to this question.
A. DATA: gv_target TYPE p DECIMALS 2. CONSTANTS: go intl TYPE i VALUE 3. gv_target -U EXACT (2 gcojntl).
B. DATA: gv_target TYPE string. □ CONSTANTS: gco_string TYPE LENGTH 16 VALUE 0123456789ABCDEF*. gv_target = EXACT # gco_string+5 (5)).
C. DATA: gv_target TYPE c LENGTH 5. V □ CONSTANTS: ECO string TYPE string VALUE 0123456789ABCDEF". gv_target - EXACT (gco_string + 5 (6)).
D. DATA: Ev target TYPE p DECIMALS 3. CONSTANTS: gcojntl TYPE i VALUE 2. Ev_target -U EXACT #2 / gcojntl).
E. DATA: gv_target TYPE d. s/ □ CONSTANTS: gco_date TYPE d VALUE '20331233*. gv_target EXACT (geo_date).
Answer: A,C,E (LEAVE A REPLY)
The patterns that raise an exception are those that use the constructor operator EXACT to perform a lossless assignment or calculation, but the result cannot be converted to the target data type without data loss. The following are the explanations for each pattern:
A: This pattern raises the exception CX_SY_CONVERSION_LOST because the result of the calculation 2 * 3 is 6, which cannot be assigned to a packed number with two decimal places without losing the integer part. The operator -U is used to perform a lossless calculation with the calculation type decfloat34.
B: This pattern does not raise an exception because the result of the substring expression gco_string+5(5) is '6789A', which can be assigned to a string without data loss. The operator EXACT # is used to perform a lossless assignment with the data type of the argument.

C: This pattern raises the exception CX_SY_CONVERSION_LOST because the result of the substring expression gco_string+5(6) is '6789AB', which cannot be assigned to a character field with length 5 without losing the last character. The operator EXACT is used to perform a lossless assignment with the data type of the target field.
D: This pattern does not raise an exception because the result of the calculation 2 / 2 is 1, which can be assigned to a packed number with three decimal places without data loss. The operator -U is used to perform a lossless calculation with the calculation type decfloat34.
E: This pattern raises the exception CX_SY_CONVERSION_ERROR because the constant gco_date contains an invalid value '20331233' for a date data type, which cannot be converted to a valid date. The operator EXACT is used to perform a lossless assignment with the data type of the target field.

NEW QUESTION: 21
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: $session.system_date)...
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: 20238181')...)
Answer: A,C (LEAVE A REPLY)

NEW QUESTION: 22
When processing an internal table with the statement LOOP AT itab... ENDLOOP, what system variable contains the current row number?

A. sy-index
B. sy-subrc
C. sy-linno
D. sy-tabix
Answer: (SHOW ANSWER)
When processing an internal table with the statement LOOP AT itab... ENDLOOP, the system variable that contains the current row number is sy-tabix. The sy-tabix variable is a predefined field of the system structure sy that holds the index or the row number of the current line in an internal table loop. The sy-tabix variable is initialized with the value 1 for the first loop pass and is incremented by 1 for each subsequent loop pass. The sy-tabix variable can be used to access or modify the current line of the internal table using the index access12.

NEW QUESTION: 23
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 = @ (cl_abap_context_info->get_system_date ())...
B. ...SELECT * FROM demo_cds_param_view entity (p_date: $session.system_date)...
C. ...SELECT * FROM deno_cds_param_view_entity (p_date - '20230101')...)
D. ...SELECT * FROM demo_cds_param_view_entity (p_date: 20238181')...)
Answer: (SHOW ANSWER)

NEW QUESTION: 24
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)

Explanation

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:

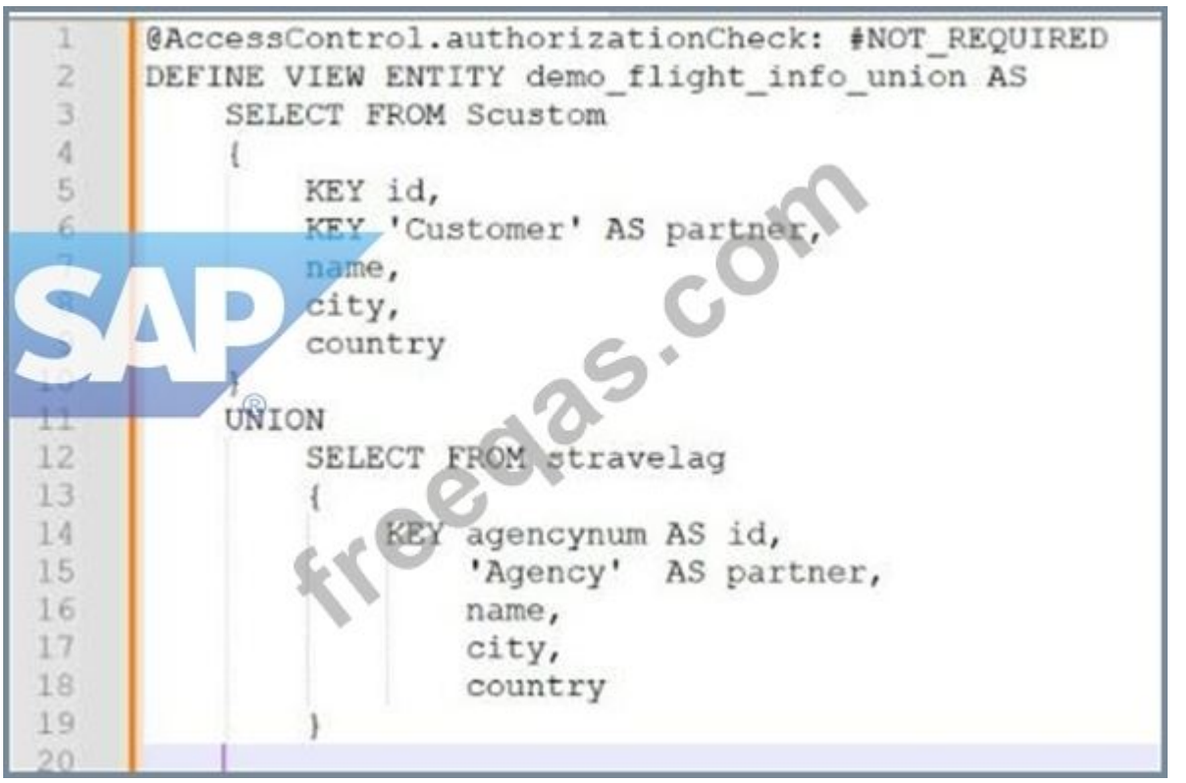
B). The system field sy-subrc is set, giving you the result of the authorization check. This is false because the system field sy-subrc is not used by Access Controls. The sy-subrc field is used by the AUTHORITY CHECK statement to indicate the result of the authorization check, but Access Controls do 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: 25

Refer to the Exhibit.



```
@AccessControl.authorizationCheck: #NOT_REQUIRED
DEFINE VIEW ENTITY demo_flight_info_union AS
    SELECT FROM Scustom
    {
        KEY id,
        KEY 'Customer' AS partner,
        name,
        city,
        country
    }
    UNION
        SELECT FROM stravelag
        {
            KEY agencynum AS id,
                'Agency'  AS partner,
                name,
                city,
                country
        }
```


when you attempt to activate the definition, what will be the response?

A. Activation error because the field names of the union do not match

B. Activation error because the field types of the union do not match

C. Activation error because the key fields of the union do not match

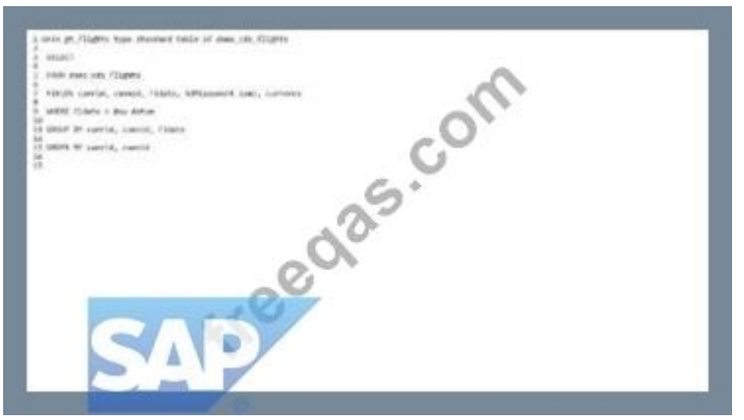
D. Activation successful

Answer: A (LEAVE A REPLY)

The response will be an activation error because the field names of the union do not match. This is because the field names of the union must match in order for the definition to be activated. The union operator combines the result sets of two or more queries into a single result set. The queries that are joined by the union operator must have the same number and type of fields, and the fields must have the same names1. In the given code, the field names of the union do not match, because the first query has the fields carrname, connid, cityfrom, and cityto, while the second query has the fields carrname, carrier_id, cityfrom, and cityto. The field connid in the first query does not match the field carrier_id in the second query. Therefore, the definition cannot be activated.

NEW QUESTION: 26

Refer to the Exhibit.



To adhere to the most recent ABAP SQL syntax conventions from SAP, on which line must you insert the "INTO TABLE @gt flights" clause to complete the SQL statement?

A. #15

B. #4

C. #6

D. #8

Answer: B (LEAVE A REPLY)

To adhere to the most recent ABAP SQL syntax conventions from SAP, you must insert the "INTO TABLE @gt flights" clause on line #4 to complete the SQL statement. This is because the INTO or APPENDING clause should be specified immediately after the SELECT clause, according to the ABAP SQL syntax conventions1. The INTO or APPENDING clause defines the data object to which the results set of the SELECT statement is assigned. The data object can be an internal table, a work area, or an inline declaration. In this case, the data object is an internal table named gt_flights, which is created using the inline declaration operator @DATA. The inline declaration operator allows you to declare and create a data object in the same statement where it is used, without the need for a separate DATA statement2.

The other lines are not suitable for inserting the "INTO TABLE @gt flights" clause, as they would violate the ABAP SQL syntax conventions or cause syntax errors. These lines are:

#6: This line is not suitable for inserting the "INTO TABLE @gt flights" clause, as it would cause a syntax error. This is because the FROM clause must be specified before the INTO or APPENDING clause, according to the ABAP SQL syntax conventions1. The FROM clause defines the data sources from which the data is read, such as database tables, CDS view entities, or CDS DDIC-based views. In this case, the data source is the database table flights.

#8: This line is not suitable for inserting the "INTO TABLE @gt flights" clause, as it would cause a syntax error. This is because the ORDER BY clause must be specified after the INTO or APPENDING clause, according to the ABAP SQL syntax conventions1. The ORDER BY clause defines the sort order of the results set of the SELECT statement. In this case, the results set is sorted by the fields carrid, connid, and fltime.

#15: This line is not suitable for inserting the "INTO TABLE @gt flights" clause, as it would violate the ABAP SQL syntax conventions. This is because the INTO or APPENDING clause should be specified as close as possible to the SELECT clause, according to the ABAP SQL syntax conventions1. The INTO or APPENDING clause should not be separated from the SELECT

clause by other clauses, such as the WHERE clause, the GROUP BY clause, the HAVING clause, the UNION clause, or the ORDER BY clause. This is to improve the readability and maintainability of the ABAP SQL statement.

NEW QUESTION: 27

To adhere to the most recent ABAP SQL syntax conventions from SAP, on which line must you insert the
"INTO TABLE @gt flights" clause to complete the SQL statement?

A. #15

B. #4

C. #6

D. #8

Answer: (SHOW ANSWER)

Explanation

To adhere to the most recent ABAP SQL syntax conventions from SAP, you must insert the "INTO TABLE
@gt flights" clause on line #4 to complete the SQL statement. This is because the INTO or APPENDING clause should be specified immediately after the SELECT clause, according to the ABAP SQL syntax conventions1. The INTO or APPENDING clause defines the data object to which the results set of the SELECT statement is assigned. The data object can be an internal table, a work area, or an inline declaration.

In this case, the data object is an internal table named gt_flights, which is created using the inline declaration operator @DATA. The inline declaration operator allows you to declare and create a data object in the same statement where it is used, without the need for a separate DATA statement2.

The other lines are not suitable for inserting the "INTO TABLE @gt flights" clause, as they would violate the ABAP SQL syntax conventions or cause syntax errors. These lines are:

#6: This line is not suitable for inserting the "INTO TABLE @gt flights" clause, as it would cause a syntax error. This is because the FROM clause must be specified before the INTO or APPENDING clause, according to the ABAP SQL syntax conventions1. The FROM clause

defines the data sources from which the data is read, such as database tables, CDS view entities, or CDS DDIC-based views. In this case, the data source is the database table flights.
#8: This line is not suitable for inserting the "INTO TABLE @gt flights" clause, as it would cause a syntax error. This is because the ORDER BY clause must be specified after the INTO or APPENDING clause, according to the ABAP SQL syntax conventions1. The ORDER BY clause defines the sort order of the results set of the SELECT statement. In this case, the results set is sorted by the fields carrid, connid, and fltime.
#15: This line is not suitable for inserting the "INTO TABLE @gt flights" clause, as it would violate the ABAP SQL syntax conventions. This is because the INTO or APPENDING clause should be specified as close as possible to the SELECT clause, according to the ABAP SQL syntax conventions1. The INTO or APPENDING clause should not be separated from the SELECT clause by other clauses, such as the WHERE clause, the GROUP BY clause, the HAVING clause, the UNION clause, or the ORDER BY clause. This is to improve the readability and maintainability of the ABAP SQL statement.
References: SELECT - ABAP Keyword Documentation, Inline Declarations - ABAP Keyword Documentation

NEW QUESTION: 28
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)

Explanation
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: 29
Which function call returns 0?
A. Count_any_of (val - 'ABAP ABAP abap' sub "AB")
B. Count (val - 'ABAP ABAP abap' sub - 'AB')
C. find_any_of (val = "ABAP ABAP abap' sub = "AB")
D. find_any_not_of(val 'ABAP ABAP abap' sub = 'AB')
Answer: (SHOW ANSWER)
The function find_any_not_of returns the position of the first character in the string val that is not contained in the string sub. If no such character is found, the function returns 0. In this case, the string val contains only the characters A, B, and a, which are all contained in the string sub, so the function returns 0. The other functions return positive values, as follows:
Count_any_of returns the number of occurrences of any character in the string sub within the string val. In this case, it returns 8, since there are 8 A's and B's in val.
Count returns the number of occurrences of the string sub within the string val. In this case, it returns 2, since there are 2 AB's in val.
find_any_of returns the position of the first character in the string val that is contained in the string sub. In this case, it returns 1, since the first character A is in sub. Reference: String Functions - ABAP Keyword Documentation, Examples of String Functions - ABAP Keyword Documentation

NEW QUESTION: 30
Which of the following is a generic internal table type?

A. SORTED TABLE
B. INDEX TABLE
C. STANDARD TABLE
D. HASHED TABLE
Answer: B (LEAVE A REPLY)
A generic internal table type is a table type that does not define all the attributes of an internal table in the ABAP Dictionary; it leaves some of these attributes undefined. A table type is generic in the following cases1:
You have selected Index Table or Not Specified as the access type.
You have not specified a table key or specified an incomplete table key.
You have specified a generic secondary table key.
A generic table type can be used only for typing formal parameters or field symbols. A generic table type cannot be used for defining data objects or constants2.
Therefore, the correct answer is B. INDEX TABLE, which is a generic table type that does not specify the access type or the table key. The other options are not generic table types, because:
A) SORTED TABLE is a table type that specifies the access type as sorted and the table key as a unique or non-unique primary key3.
C) STANDARD TABLE is a table type that specifies the access type as standard and the table key as a non-unique standard key that consists of all the fields of the table row in the order in which they are defined4.
D) HASHED TABLE is a table type that specifies the access type as hashed and the table key as a unique primary key5.

NEW QUESTION: 31

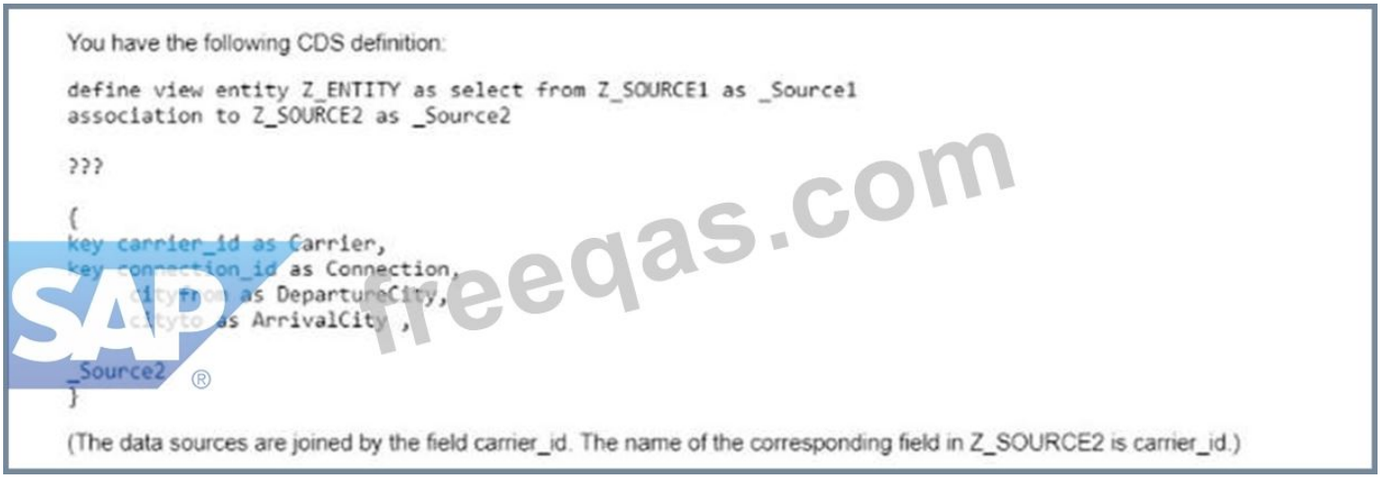
Using ABAP SQL, which select statement selects the mat field on line #17?
A. SELECT mat FROM Material...
B. SELECT mat FROM demo_sales_cds_so_i_ve...
C. SELECT mat FROM demo_sales_so_i...
D. SELECT mat FROM demo sales cds material ve...
Answer: B (LEAVE A REPLY)
Explanation
Using ABAP SQL, the select statement that selects the mat field on line #17 is:

SELECT mat FROM demo_sales_cds_so_i_ve...
This statement selects the mat field from the CDS view demo_sales_cds_so_i_ve, which is defined on line #1.
The CDS view demo_sales_cds_so_i_ve is a projection view that projects the fields of the CDS view demo_sales_cds_so_i, which is defined on line #2. The CDS view demo_sales_cds_so_i is a join view that joins the fields of the database table demo_sales_so_i, which is defined on line #3, and the CDS view demo_sales_cds_material_ve, which is defined on line #4. The CDS view demo_sales_cds_material_ve is a value help view that provides value help for the material field of the database table demo_sales_so_i. The mat field is an alias for the material field of the database table demo_sales_so_i, which is defined on line #91.
The other options are not valid because:
A). SELECT mat FROM Material... is not valid because Material is not a valid data source in the given code. There is no CDS view or database table named Material.
C). SELECT mat FROM demo_sales_so_i... is not valid because demo_sales_so_i is not a valid data source in the given code. There is no CDS view named demo_sales_so_i, only a database table. To access a database table, the keyword TABLE must be used, such as SELECT mat FROM TABLE demo_sales_so_i...
D). SELECT mat FROM demo sales cds material ve... is not valid because demo sales cds material ve is not a valid data source in the given code. There is no CDS view or database table named demo sales cds material ve. The correct name of the CDS view is demo_sales_cds_material_ve, with underscores instead of spaces.
References: 1: Projection Views - ABAP Keyword Documentation

NEW QUESTION: 32
Which statement can you use to change the contents of a row of data in an internal table?
A. Append table
B. Modify table
C. Insert table
D. Update table
Answer: B (LEAVE A REPLY)
Explanation

The statement that can be used to change the contents of a row of data in an internal table is MODIFY table.
The MODIFY table statement can be used to change the contents of one or more rows of an internal table, either by specifying the table index, the table key, or a condition. The MODIFY table statement can also be used to change the contents of a database table, by specifying the table name and a work area or an internal table. The MODIFY table statement can use the TRANSPORTING addition to specify which fields should be changed, and the WHERE addition to specify which rows should be changed.
The other statements are not suitable for changing the contents of a row of data in an internal table, as they have different purposes and effects. These statements are:
APPEND table: This statement can be used to add a new row of data to the end of an internal table, either by specifying a work area or an inline declaration. The APPEND table statement does not change the existing rows of the internal table, but only increases the number of rows by one.
INSERT table: This statement can be used to insert a new row of data into an internal table, either by specifying the table index, the table key, or a sorted position. The INSERT table statement does not change the existing rows of the internal table, but only shifts them to make room for the new row. The INSERT table statement can also be used to insert a new row of data into a database table, by specifying the table name and a work area or an inline declaration.
UPDATE table: This statement can be used to update the contents of a database table, by specifying the table name and a work area or an internal table. The UPDATE table statement can use the SET addition to specify which fields should be updated, and the WHERE addition to specify which rows should be updated. The UPDATE table statement does not affect the internal table, but only the corresponding database table.
References: MODIFY table - ABAP Keyword Documentation, APPEND table - ABAP Keyword Documentation, INSERT table - ABAP Keyword Documentation, UPDATE table - ABAP Keyword Documentation

NEW QUESTION: 33
Refer to the Exhibit.



Which of the following ON conditions must you insert in place of "???"?

A. ON Z_Sourcel.camer_id = 7_Source2 carrier_id
B. ON Sprojection Camer=Source2 carrier_id
C. ON Sprojection. Carrier Source2.carrier
D. ON Sprojection.carrier_id=Z_Source2.carrier_id
Answer: D (LEAVE A REPLY)
The correct ON condition that must be inserted in place of "???" is:
ON Sprojection.carrier_id=Z_Source2.carrier_id
This ON condition specifies the join condition between the CDS view Sprojection and the database table Z_Source2. The join condition is based on the field carrier_id, which is the primary key of both the CDS view and the database table. The ON condition ensures that only the records that have the same value for the carrier_id field are joined together1.
The other options are not valid ON conditions, because:
A) ON Z_Sourcel.camer_id = 7_Source2 carrier_id is not valid because Z_Sourcel and 7_Source2 are not valid data sources in the given code. There is no CDS view or database table named Z_Sourcel or 7_Source2. The correct names are Z_Source1 and Z_Source2. Moreover, the field camer_id is not a valid field in the given code. There is no field named camer_id in any of the data sources. The correct name is carrier_id.
B) ON Sprojection Camer=Source2 carrier_id is not valid because Sprojection and Source2 are not valid data sources in the given code. There is no CDS view or database table named Sprojection or Source2. The correct names are Sprojection and Z_Source2. Moreover, the field Camer is not a valid field in the given code. There is no field named Camer in any of the data sources. The correct name is carrier_id. Furthermore, the ON condition is missing the dot (.) operator between the data source name and the field name, which is required to access the fields of the data source1.
C) ON Sprojection. Carrier Source2.carrier is not valid because Carrier and carrier are not valid fields in the given code. There is no field named Carrier or carrier in any of the data sources. The correct name is carrier_id. Moreover, the ON condition is missing the dot (.) operator between the data source name and the field name, which is required to access the fields of the data source1.

NEW QUESTION: 34
In RESTful Application Programming, which EML statement retrieves an object?
A. Find entity
B. Select entity
C. Get entity
D. Read entity
Answer: C (LEAVE A REPLY)
In RESTful Application Programming, the EML statement that retrieves an object is GET entity. The GET entity statement is used to read data of an entity instance from the database or the transaction buffer. The GET entity statement can specify the entity name, the entity key, and the entity elements to be retrieved. The GET entity statement can also use the IN LOCAL MODE addition to bypass the access control, authorization control, and feature control checks. The GET

entity statement returns a single entity instance or raises an exception if no instance is found or multiple instances match the key.
The other EML statements are not used to retrieve an object, but have different purposes and effects. These statements are:
FIND entity: This statement is used to search for entity instances that match a given condition. The FIND entity statement can specify the entity name, the entity elements to be returned, and the condition to be applied. The FIND entity statement can also use the IN LOCAL MODE addition to bypass the access control, authorization control, and feature control checks. The FIND entity statement returns a table of entity instances or an empty table if no instances match the condition.
SELECT entity: This statement is used to query data of entity instances from the database or the transaction buffer. The SELECT entity statement can specify the entity name, the entity elements to be returned, and the filter, order, and aggregation options to be applied. The SELECT entity statement can also use the IN LOCAL MODE addition to bypass the access control, authorization control, and feature control checks. The SELECT entity statement returns a table of entity instances or an empty table if no instances match the query.
READ entity: This statement is not a valid EML statement, but an ABAP statement. The READ statement is used to access a single row of an internal table using the table index or the table key. The READ statement can also use the TRANSPORTING addition to specify which fields should be returned, and the INTO addition to specify the target variable. The READ statement returns a single row of the internal table or raises an exception if no row is found or multiple rows match the key.

NEW QUESTION: 35
Which extensibility type does SAP recommend you use to enhance the existing UI for an SAP Fiori app?
A. Key user
B. Classic
C. Side-by-side
D. Developer
Answer: (SHOW ANSWER)
According to the SAP clean core extensibility and ABAP cloud topic, SAP recommends using developer extensibility to enhance the existing UI for an SAP Fiori app. Developer extensibility allows you to use the UI adaptation editor in SAP Web IDE to modify the UI layout, add or remove fields, and bind them to the data model. You can also use the SAPUI5 framework to create custom controls, views, and controllers. Developer extensibility is based on the in-app extensibility concept, which means that the extensions are part of the same application and are deployed together with the app. Developer extensibility requires developer skills and access to the source code of the app. Reference: SAP Learning Hub, SAP S/4HANA Cloud Extensibility - In-App Extensibility, SAP Fiori: Extensibility

NEW QUESTION: 36
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)
Explanation
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: 37
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: A,C (LEAVE A REPLY)
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.

NEW QUESTION: 38
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: D (LEAVE A REPLY)

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.

NEW QUESTION: 39

Given this code:

```
INTERFACE if1.
  METHODS m1.
ENDINTERFACE.

CLASS cl1 DEFINITION.
...
  INTERFACES if1.
ENDCLASS.
...
CLASS cl2 DEFINITION.
...
  DATA mo_if1 TYPE REF TO if1.
...
ENDCLASS.
...
```

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: C,D,E (LEAVE A REPLY)

Explanation

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: 40

To adhere to the most recent ABAP SQL syntax conventions from SAP, on which line must you insert the
"INTO TABLE @gt flights" clause to complete the SQL statement?
A. #15
B. #4
C. #6
D. #8
Answer: B (LEAVE A REPLY)
Explanation
To adhere to the most recent ABAP SQL syntax conventions from SAP, you must insert the "INTO TABLE
@gt flights" clause on line #4 to complete the SQL statement. This is because the INTO or APPENDING clause should be specified immediately after the SELECT clause, according to the ABAP SQL syntax conventions1. The INTO or APPENDING clause defines the data object to which the results set of the SELECT statement is assigned. The data object can be an internal table, a work area, or an inline declaration.

In this case, the data object is an internal table named gt_flights, which is created using the inline declaration operator @DATA. The inline declaration operator allows you to declare and create a data object in the same statement where it is used, without the need for a separate DATA statement2.
The other lines are not suitable for inserting the "INTO TABLE @gt flights" clause, as they would violate the ABAP SQL syntax conventions or cause syntax errors. These lines are:
#6: This line is not suitable for inserting the "INTO TABLE @gt flights" clause, as it would cause a syntax error. This is because the FROM clause must be specified before the INTO or APPENDING clause, according to the ABAP SQL syntax conventions1. The FROM clause defines the data sources from which the data is read, such as database tables, CDS view entities, or CDS DDIC-based views. In this case, the data source is the database table flights.
#8: This line is not suitable for inserting the "INTO TABLE @gt flights" clause, as it would cause a syntax error. This is because the ORDER BY clause must be specified after the INTO or APPENDING clause, according to the ABAP SQL syntax conventions1. The ORDER BY clause defines the sort order of the results set of the SELECT statement. In this case, the results set is sorted by the fields carrid, connid, and fltime.
#15: This line is not suitable for inserting the "INTO TABLE @gt flights" clause, as it would violate the ABAP SQL syntax conventions. This is because the INTO or APPENDING clause should be specified as close as possible to the SELECT clause, according to the ABAP SQL syntax conventions1. The INTO or APPENDING clause should not be separated from the SELECT clause by other clauses, such as the WHERE clause, the GROUP BY clause, the HAVING clause, the UNION clause, or the ORDER BY clause. This is to improve the readability and maintainability of the ABAP SQL statement.
References: SELECT - ABAP Keyword Documentation, Inline Declarations - ABAP Keyword Documentation

NEW QUESTION: 41
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.

NEW QUESTION: 42
After you created a database table in the RESTful Application Programming model, what do you create next?
A. A metadata extension
B. A projection view
C. A data model view
D. A service definition
Answer: B (LEAVE A REPLY)
After you created a database table in the RESTful Application Programming model (RAP), the next step is to create a projection view on the database table. 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. For example:
The following code snippet defines a projection view ZI_AGENCY on the database table /DMO/AGENCY:
define view ZI_AGENCY as select from /dmo/agency { key agency_id, agency_name, street, city, region, postal_code, country, phone_number, url } The projection view is used to expose the data of the database table to the service definition, which is the next step in the RAP. The 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 aggregatable12. For example:
The following code snippet defines a service definition ZI_AGENCY_SRV that exposes the projection view ZI_AGENCY as an OData service:
define service ZI_AGENCY_SRV { expose ZI_AGENCY as Agency; }
You cannot do any of the following:
A) A 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, a metadata extension

is not the next step after creating a database table in the RAP, as it is not required to expose the data of the database table to the service definition. A metadata extension can be created later to customize the UI or analytical application that uses the service12.
C) A data model view: A data model view is a CDS artefact that defines a view on one or more data sources, such as tables, views, or associations. A data model view can select, rename, or aggregate the fields of the data sources, and it can also change the properties of the fields, such as whether they are read-only or not. The properties of the fields are defined by the annotations or the behaviour definitions of the data model view. A data model view is used to define the data model of a business object, which is a CDS entity that represents a business entity or concept, such as a customer, an order, or a product. However, a data model view is not the next step after creating a database table in the RAP, as it is not required to expose the data of the database table to the service definition. A data model view can be created later to define a business object that uses the database table as a data source12.
D) 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, a service definition is not the next step after creating a database table in the RAP, as it requires a projection view or a data model view to expose the data of the database table. A service definition can be created after creating a projection view or a data model view on the database table12.

NEW QUESTION: 43
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.

NEW QUESTION: 44
What are valid statements? Note: There are 2 correct answers to this question.
A. ##NEEDED is checked by the syntax checker.
B. The pragma is not checked by the syntax checker.
C. #EC_NEEDED is not checked by the syntax checker.
D. The pseudo-comment is checked by the syntax checker
Answer: (SHOW ANSWER)
Explanation
Both statements are valid in ABAP, but they have different effects on the program.
##NEEDED is a pragma that can be used to hide warnings from the ABAP compiler syntax check. It tells the check tools that a variable or a parameter is needed for further processing, even if it is not used in the current statement. For example, if you declare a variable without assigning any value to it, you can use ##NEEDED to suppress the warning about unused variables12.
The pragma is not checked by the syntax checker means that you can use any pragma to hide any warning from the ABAP compiler syntax check, regardless of its effect on the program logic or performance. For example, if you use ##SHADOW to hide a warning about an obscured function, you can also use it to hide a warning about an invalid character in a string12.
You cannot do any of the following:
#EC_NEEDED is not checked by the syntax checker: This is not a valid statement in ABAP. There is no pseudo-comment with #EC_NEEDED in ABAP3.
The pseudo-comment is checked by the syntax checker: This is false. Pseudo-comments are obsolete and should no longer be used in ABAP. They were replaced by pragmas since SAP NW 7.0 EhP2 (Enhancement Package)4.
References: 1: Pragmas - ABAP Keyword Documentation - SAP Online Help 2: [What are pragmas and pseudo comments in ABAP? | SAP Blogs - SAP Community] 3: ABAP Keyword Documentation - SAP Online Help 4: What are PRAGMAS and Pseudo comments in SAP ABAP

NEW QUESTION: 45
Which of the following results in faster access to internal tables? Note: There are 3 correct answers to this question.
A. In a sorted internal table, specifying the primary key partially from the left without gaps.
B. In a sorted internal table, specifying the primary key completely.

C. In a standard internal table, specifying the primary key partially from the left without gaps.
D. In a hashed internal table, specifying the primary key partially from the left without gaps.
E. In a hashed internal table, specifying the primary key completely.
Answer: B,D,E (LEAVE A REPLY)
The access to internal tables can be optimized by using the appropriate table type and specifying the table key. The table key is a set of fields that uniquely identifies a row in the table and determines the sorting order of the table. The table key can be either the primary key or a secondary key. The primary key is defined by the table type and the table definition, while the secondary key is defined by the user using the KEY statement1.
The following results in faster access to internal tables:
B) In a sorted internal table, specifying the primary key completely. A sorted internal table is a table type that maintains a predefined sorting order, which is defined by the primary key in the table definition. The primary key can be either unique or non-unique. A sorted internal table can be accessed using the primary key or the table index. The access using the primary key is faster than the access using the table index, because the system can use a binary search algorithm to find the row. However, the primary key must be specified completely, meaning that all the fields of the primary key must be given in the correct order and without gaps2.
D) In a hashed internal table, specifying the primary key partially from the left without gaps. A hashed internal table is a table type that does not have a predefined sorting order, but uses a hash algorithm to store and access the rows. The primary key of a hashed internal table must be unique and cannot be changed. A hashed internal table can only be accessed using the primary key, not the table index. The access using the primary key is very fast, because the system can directly calculate the position of the row using the hash algorithm. The primary key can be specified partially from the left without gaps, meaning that some of the fields of the primary key can be omitted, as long as they are the rightmost fields and there are no gaps between the specified fields.
E) In a hashed internal table, specifying the primary key completely. A hashed internal table is a table type that does not have a predefined sorting order, but uses a hash algorithm to store and access the rows. The primary key of a hashed internal table must be unique and cannot be changed. A hashed internal table can only be accessed using the primary key, not the table index. The access using the primary key is very fast, because the system can directly calculate the position of the row using the hash algorithm. The primary key can be specified completely, meaning that all the fields of the primary key must be given in the correct order.
The following do not result in faster access to internal tables, because:
A) In a sorted internal table, specifying the primary key partially from the left without gaps. A sorted internal table is a table type that maintains a predefined sorting order, which is defined by the primary key in the table definition. The primary key can be either unique or non-unique. A sorted internal table can be accessed using the primary key or the table index. The access using the primary key is faster than the access using the table index, because the system can use a binary search algorithm to find the row. However, the primary key must be specified completely, meaning that all the fields of the primary key must be given in the correct order and without gaps.

If the primary key is specified partially from the left without gaps, the system cannot use the binary search algorithm and has to perform a linear search, which is slower2.
C) In a standard internal table, specifying the primary key partially from the left without gaps. A standard internal table is a table type that does not have a predefined sorting order, but uses a sequential storage and access of the rows. The primary key of a standard internal table is the standard key, which consists of all the fields of the table row in the order in which they are defined. A standard internal table can be accessed using the primary key or the table index. The access using the primary key is slower than the access using the table index, because the system has to perform a linear search to find the row. The primary key can be specified partially from the left without gaps, but this does not improve the access speed, because the system still has to perform a linear search.

NEW QUESTION: 46
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.

NEW QUESTION: 47

Exhibit:

Which of the following statements are correct? Note: There are 2 correct answers to this question.

A. FOR defines a loop that runs over the content of source_itab

B. source_itab is only visible within the loop.

C. row is a predefined name and cannot be chosen arbitrarily.

D. row is only visible within the loop.

Answer: A,D (LEAVE A REPLY)

Explanation

The code snippet in the image is an example of using the FOR statement to create an internal table with a constructor expression. The FOR statement introduces an iteration expression that runs over the content of source_itab and assigns each row to the variable row. The variable row is then used to populate the fields of target_itab12. Some of the correct statements about the code snippet are:

FOR defines a loop that runs over the content of source_itab: This is true. The FOR statement iterates over the rows of source_itab and assigns each row to the variable row. The iteration expression can also specify a range or a condition for the loop12.

row is only visible within the loop: This is true. The variable row is a local variable that is only visible within the scope of the iteration expression. It cannot be accessed outside the loop12.
You cannot do any of the following:
source_itab is only visible within the loop: This is false. The variable source_itab is not a local variable that is defined by the FOR statement. It is an existing internal table that is used as the data source for the iteration expression. It can be accessed outside the loop12.
row is a predefined name and cannot be chosen arbitrarily: This is false. The variable row is not a predefined name that is reserved by the FOR statement. It is a user-defined name that can be chosen arbitrarily. However, it must not conflict with any existing names in the program12.
References: 1: FOR - Iteration Expressions - ABAP Keyword Documentation - SAP Online Help 2: ABAP 7.4 Syntax - FOR Loop iteration | SAP Community

NEW QUESTION: 48
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: (SHOW ANSWER)
Explanation
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: 49
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.

NEW QUESTION: 50
In class ZCL_CLASS_A, you use the statement DATA var TYPE ***
What may stand in place of ***? Note: There are 2 correct answers to this question.
A. The name of a type defined privately in class ZCL_CLASS_A
B. The name of a data element from the ABAP Dictionary
C. The name of a type defined privately in another class
D. The name of a domain from the ABAP Dictionary
Answer: (SHOW ANSWER)
In class ZCL_CLASS_A, you use the statement DATA var TYPE *** to declare a data object named var with a data type specified by ***. The data type can be any of the following1:
A predefined ABAP type, such as i, f, c, string, xstring, and so on.
A data element from the ABAP Dictionary, such as matnr, carrid, bukrs, and so on. A data element defines the semantic and technical attributes of a data field, such as the domain, the length, the data type, the description, and the value range2.

A domain from the ABAP Dictionary, such as matnr_d, carrid_d, bukrs_d, and so on. A domain defines the technical attributes of a data field, such as the data type, the length, the output length, the number of decimal places, and the value range3.
A type defined globally in a class, an interface, or a type pool, such as zcl_class_b=>type_a, zif_interface_c=>type_b, ztype_pool_d=>type_c, and so on. A global type is a type that is defined in a global repository object and can be used in any program or class4.
A type defined locally in the current class, such as type_a, type_b, type_c, and so on. A local type is a type that is defined in the declaration part of a class and can only be used within the class5.
Therefore, the possible values for *** are B. the name of a data element from the ABAP Dictionary and D. the name of a domain from the ABAP Dictionary. The other options are not valid because:
A) The name of a type defined privately in class ZCL_CLASS_A is a local type and cannot be used with the DATA statement. A local type can only be used with the TYPES statement5.
C) The name of a type defined privately in another class is a private type and cannot be accessed from outside the class. A private type can only be used within the class that defines it.

NEW QUESTION: 51
Which patterns raise an exception? Note: There are 3 correct answers to this question.
A. DATA: gv_target TYPE p DECIMALS 2. CONSTANTS: go intl TYPE i VALUE 3. gv_target -U EXACT (2 gcojntl).
B. DATA: gv_target TYPE string. CONSTANTS: gco_string TYPE LENGTH 16 VALUE 0123456789ABCDEF*. gv_target = EXACT # gco_string+5 (5)).
C. DATA: gv_target TYPE c LENGTH 5. V CONSTANTS: ECO string TYPE string VALUE 0123456789ABCDEF". gv_target - EXACT (gco_string + 5 (6)).
D. DATA: Ev target TYPE p DECIMALS 3. CONSTANTS: gcojntl TYPE i VALUE 2. Ev_target -U EXACT #2 / gcojntl).
E. DATA: gv_target TYPE d. s/ CONSTANTS: gco_date TYPE d VALUE '20331233*. gv_target EXACT (geo_date).
Answer: A,C,E (LEAVE A REPLY)
Explanation
The patterns that raise an exception are those that use the constructor operator EXACT to perform a lossless assignment or calculation, but the result cannot be converted to the target data type without data loss. The following are the explanations for each pattern:
A: This pattern raises the exception CX_SY_CONVERSION_LOST because the result of the calculation 2 * 3 is 6, which cannot be assigned to a packed number with two decimal places without losing the integer part. The operator -U is used to perform a lossless calculation with the calculation type decfloat34.
B: This pattern does not raise an exception because the result of the substring expression gco_string+5(5) is '6789A', which can be assigned to a string without data loss. The operator EXACT # is used to perform a lossless assignment with the data type of the argument.

C: This pattern raises the exception CX_SY_CONVERSION_LOST because the result of the substring expression gco_string+5(6) is '6789AB', which cannot be assigned to a character field with length 5 without losing the last character. The operator EXACT is used to perform a lossless assignment with the data type of the target field.
D: This pattern does not raise an exception because the result of the calculation 2 / 2 is 1, which can be assigned to a packed number with three decimal places without data loss. The operator -U is used to perform a lossless calculation with the calculation type decfloat34.
E: This pattern raises the exception CX_SY_CONVERSION_ERROR because the constant gco_date contains an invalid value '20331233' for a date data type, which cannot be converted to a valid date.
The operator EXACT is used to perform a lossless assignment with the data type of the target field.
References: EXACT - Lossless Operator - ABAP Keyword Documentation, Lossless Assignments - ABAP Keyword Documentation

NEW QUESTION: 52
Class super has subclass sub. Which rules are valid for the sub constructor? Note: There are 2 correct answers to this question.
A. The method signature can be changed.
B. Import parameters can only be evaluated after calling the constructor of super.
C. The constructor of super must be called before using any components of your own instance.
D. Events of your own instance cannot be raised before the registration of a handler in super.
Answer: (SHOW ANSWER)
Explanation
The sub constructor is the instance constructor of the subclass sub that inherits from the superclass super. The sub constructor has some rules that it must follow when it is defined and implemented12. Some of the valid rules are:
The method signature can be changed: This is true. The sub constructor can have a different method signature than the super constructor, which means that it can have different input parameters, output parameters, or exceptions. However, the sub constructor must still call the super constructor with appropriate actual parameters that match its interface12.
The constructor of super must be called before using any components of your own instance: This is true.
The sub constructor must ensure that the super constructor is called explicitly using super->constructor before accessing any instance components of its own class, such as attributes or methods. This is because the super constructor initializes the inherited components of the subclass and sets the self-reference me-> to the current instance12.
You cannot do any of the following:
Import parameters can only be evaluated after calling the constructor of super: This is false. The sub constructor can evaluate its own import parameters before calling the constructor of super, as long as it does not access any instance components of its own class. For example, the sub

constructor can use its import parameters to calculate some values or check some conditions that are needed for calling the super constructor12.
Events of your own instance cannot be raised before the registration of a handler in super: This is false.
The sub constructor can raise events of its own instance before calling the constructor of super, as long as it does not access any instance components of its own class. For example, the sub constructor can raise an event to notify the consumers of the subclass about some status or error that occurred during the initialization of the subclass12.
References: 1: Inheritance and Constructors - ABAP Keyword Documentation - SAP Online Help 2: Using Static and Instance constructor methods | SAP Blogs

NEW QUESTION: 53
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: (SHOW ANSWER)
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. Reference: 1: ABAP Cloud | SAP Blogs 2: SAP S/4HANA Cloud Extensibility - Overview and Comparison | SAP Blogs

NEW QUESTION: 54
What are some characteristics of secondary keys for internal tables? Note: There are 3 correct answers to this question.
A. Secondary keys must be chosen explicitly when you actually read from an internal table.
B. Multiple secondary keys are allowed for any kind of internal table.
C. Hashed secondary keys do NOT have to be unique.
D. Sorted secondary keys do NOT have to be unique.
E. Secondary keys can only be created for standard tables.
Answer: (SHOW ANSWER)
Secondary keys are additional keys that can be defined for internal tables to optimize the access to the table using fields that are not part of the primary key. Secondary keys can be either sorted

or hashed, depending on the table type and the uniqueness of the key. Secondary keys have the following characteristics1:

A) Secondary keys must be chosen explicitly when you actually read from an internal table. This means that when you use a READ TABLE or a LOOP AT statement to access an internal table, you have to specify the secondary key that you want to use with the USING KEY addition. For example, the following statement reads an internal table itab using a secondary key sec_key: READ TABLE itab USING KEY sec_key INTO DATA(wa).

If you do not specify the secondary key, the system will use the primary key by default2.

B) Multiple secondary keys are allowed for any kind of internal table. This means that you can define more than one secondary key for an internal table, regardless of the table type. For example, the following statement defines an internal table itab with two secondary keys sec_key_1 and sec_key_2:

DATA itab TYPE SORTED TABLE OF ty_itab WITH NON-UNIQUE KEY sec_key_1 COMPONENTS field1 field2 sec_key_2 COMPONENTS field3 field4.

You can then choose which secondary key to use when you access the internal table1.

D) Sorted secondary keys do NOT have to be unique. This means that you can define a sorted secondary key for an internal table that allows duplicate values for the key fields. A sorted secondary key maintains a predefined sorting order for the internal table, which is defined by the key fields in the order in which they are specified. For example, the following statement defines a sorted secondary key sec_key for an internal table itab that sorts the table by field1 in ascending order and field2 in descending order:

DATA itab TYPE STANDARD TABLE OF ty_itab WITH NON-UNIQUE SORTED KEY sec_key COMPONENTS field1 ASCENDING field2 DESCENDING.

You can then access the internal table using the sorted secondary key with a binary search algorithm, which is faster than a linear search3.

The following are not characteristics of secondary keys for internal tables, because:

C) Hashed secondary keys do NOT have to be unique. This is false because hashed secondary keys must be unique. This means that you can only define a hashed secondary key for an internal table that does not allow duplicate values for the key fields. A hashed secondary key does not have a predefined sorting order for the internal table, but uses a hash algorithm to store and access the table rows. For example, the following statement defines a hashed secondary key sec_key for an internal table itab that hashes the table by field1 and field2:

DATA itab TYPE STANDARD TABLE OF ty_itab WITH UNIQUE HASHED KEY sec_key COMPONENTS field1 field2.

You can then access the internal table using the hashed secondary key with a direct access algorithm, which is very fast.

E) Secondary keys can only be created for standard tables. This is false because secondary keys can be created for any kind of internal table, such as standard tables, sorted tables, and hashed tables. However, the type of the secondary key depends on the type of the internal table. For example, a standard table can have sorted or hashed secondary keys, a sorted table can have sorted secondary keys, and a hashed table can have hashed secondary keys1.

NEW QUESTION: 55

Given the following Core Data Services View Entity Data Definition:

```
1 @AccessControl.authorizationCheck: #NOT_REQUIRED
2 DEFINE VIEW ENTITY demo_cds_assoc_element
3   AS SELECT FROM scarr
4   ASSOCIATION OF ONE TO MANY demo_cds_assoc_spfli AS _spfli
5   ON scarr.carrid = _spfli.carrid
6   {
7     KEY carrid,
8 ?
9     carrname
10  }
```


The "demo_ods_assoc_spfi data source referenced in line #4 contains a field "connid" which you would like to expose in the element list.

Which of the following statements would do this if inserted on line #8?

A. demo_ods_assoc_spfli.connid,

B. demo_ods_assoc_spfli-connid/

C. spfli-connid,

D. _spfli.connid/

Answer: A (LEAVE A REPLY)

Explanation

The statement that can be used to expose the field "connid" of the data source "demo_ods_assoc_spfli" in the element list is A. demo_ods_assoc_spfli.connid,. This statement uses the dot notation to access the field

"connid" of the data source "demo_ods_assoc_spfli", which is an association defined on line #4. The association "demo_ods_assoc_spfli" links the data source "demo_ods" with the table "spfli" using the field

"carrid". The statement also ends with a comma to separate it from the next element in the list12.

You cannot do any of the following:

B: demo_ods_assoc_spfli-connid/: This statement uses the wrong syntax to access the field "connid" of the data source "demo_ods_assoc_spfli". The dash notation is used to access the components of a structure or a table, not the fields of a data source. The statement also ends with a slash, which is not a valid separator for the element list12.

C: spfli-connid,: This statement uses the wrong data source name to access the field "connid". The data source name should be "demo_ods_assoc_spfli",not "spfli". The statement also uses the wrong syntax to access the field "connid", as explained above12.

D:_spfli.connid/: This statement uses the wrong data source name and the wrong separator to access the field "connid". The data source name should be "demo_ods_assoc_spfli", not "_spfli". The statement also ends with a slash, which is not a valid separator for the element list12.

References: 1: ABAP CDS - SELECT, select_list - ABAP Keyword Documentation - SAP Online Help 2: ABAP CDS - SELECT, from - ABAP Keyword Documentation - SAP Online Help

NEW QUESTION: 56

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

```
1 CLASS zcl_demo_class DEFINITION.
2 METHODS: m1.
3 ENDCLASS..
4 CLASS zcl_demo_class Implementation.
5 METHOD m1.
6 CALL FUNCTION 'ZF1'.
7 ENDMETHOD
8 ENDCLASS.
```


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: A,B (LEAVE A REPLY)

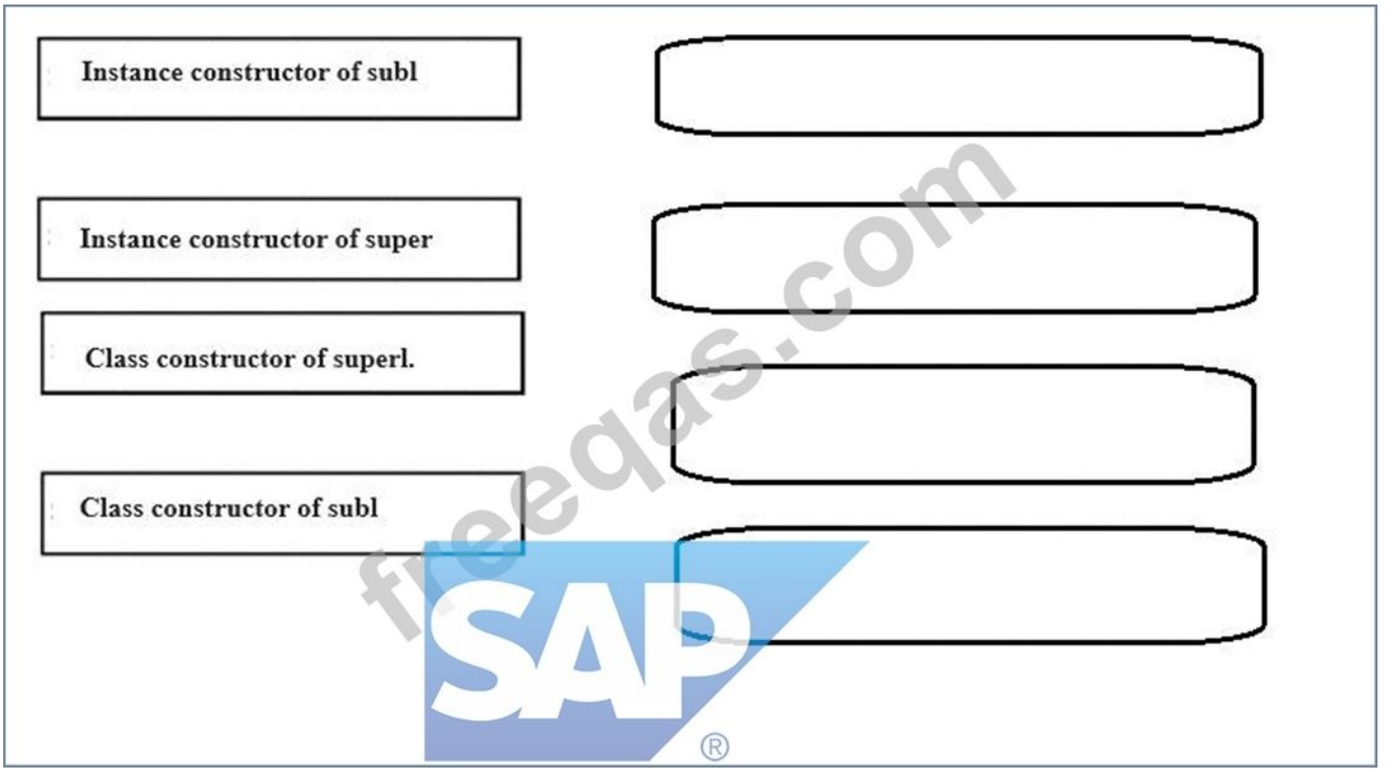
The ABAP Cloud Development Model requires that only public SAP APIs and extension points are used to access SAP functionality and data. These APIs and extension points are released by SAP and documented in the SAP API Business Hub1. Customer-created function modules are not part of the public SAP APIs 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.

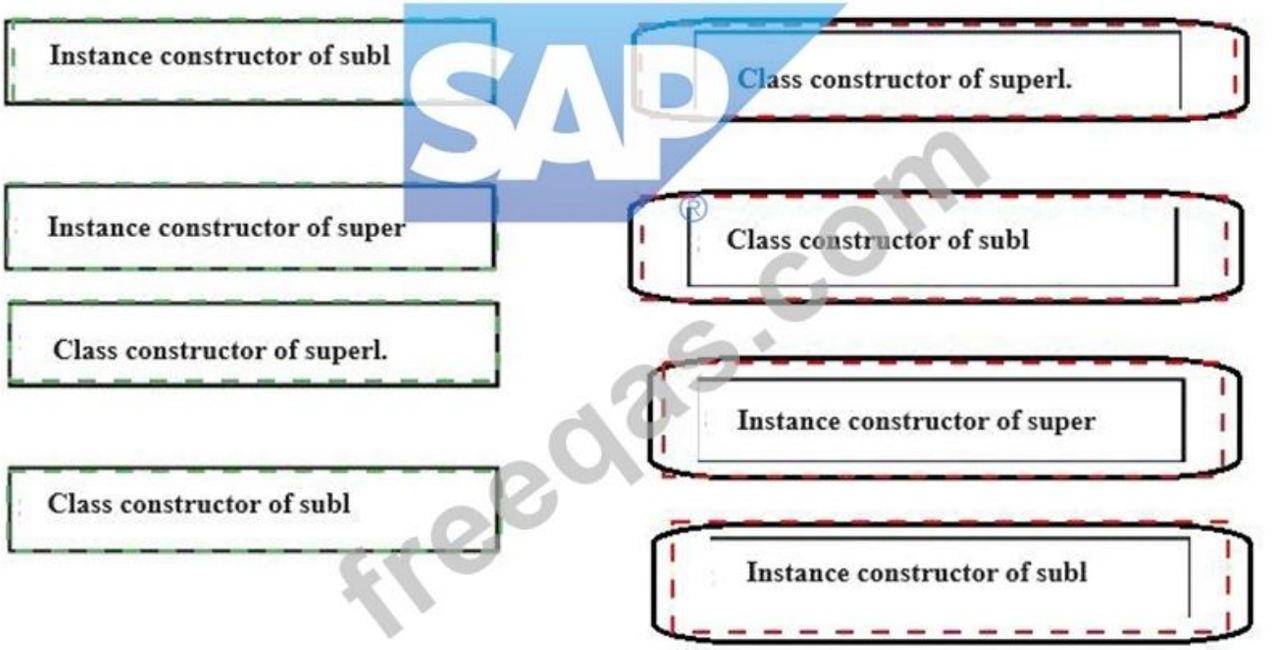
Use the ABAP Cloud Connector to call the function module as a remote function call (RFC) from an ABAP Cloud class. The ABAP Cloud Connector is a service that enables the secure and reliable communication between SAP BTP, ABAP environment and on-premise systems. The function module must be exposed as 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.

NEW QUESTION: 57

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

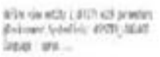
The sequence in which the constructors will be executed is as follows:

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: 58
What RESTful Application Programming feature is used to ensure the uniqueness of a semantic key?
A. Validation
B. Action
C. Determination
Answer: C (LEAVE A REPLY)
The RESTful Application Programming feature that is used to ensure the uniqueness of a semantic key is determination. A determination is a type of behavior implementation that defines a logic that is executed automatically when certain events occur, such as create, update, delete, or activate. A determination can be used to calculate or derive values for certain fields, such as semantic keys, based on other fields or external sources. A determination can also be used to check the uniqueness of a semantic key by comparing it with the existing values in the database or the transaction buffer. A determination can use the ABAP SQL or the EML syntax to access

and manipulate data. A determination can be defined using the DETERMINE action clause in the behavior definition of a CDS view entity or a projection view. A determination can also be annotated with the @ObjectModel.determination annotation to specify the event, the timing, and the scope of the determination12 The other RESTful Application Programming features are not used to ensure the uniqueness of a semantic key, but have different purposes and effects. These features are:
Validation: A validation is a type of behavior implementation that defines a logic that is executed automatically when certain events occur, such as create, update, delete, or activate. A validation can be used to check the consistency and correctness of the data, such as mandatory fields, data types, value ranges, or business rules. A validation can use the ABAP SQL or the EML syntax to access and manipulate data. A validation can be defined using the VALIDATE action clause in the behavior definition of a CDS view entity or a projection view. A validation can also be annotated with the @ObjectModel.validation annotation to specify the event, the timing, and the scope of the validation12 Action: An action is a type of behavior implementation that defines a logic that is executed explicitly by the user or the application. An action can be used to perform a specific business operation, such as creating, updating, deleting, or activating an entity instance, or triggering a workflow or a notification. An action can use the ABAP SQL or the EML syntax to access and manipulate data. An action can be defined using the ACTION clause in the behavior definition of a CDS view entity or a projection view. An action can also be annotated with the @ObjectModel.action annotation to specify the name, the description, the parameters, and the visibility of the action12

NEW QUESTION: 59
What are the effects of this annotation? Note: There are 2 correct answers to this question.

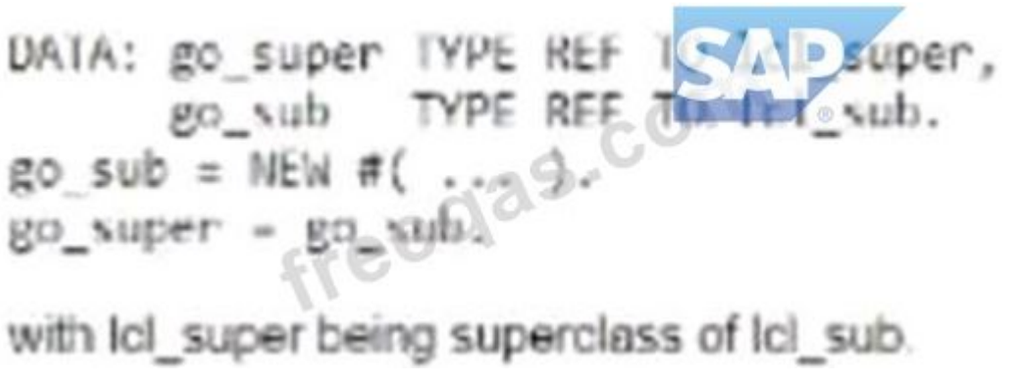


A. The value of sy-langu will be passed to the CDS view automatically both when you use the -1 CDS view in ABAP and in another CDS view entity (view on view).
B. You can still override the default value with a value of your own.
C. The value of sy-langu will be passed to the CDS view automatically when you use the CDS view in ABAP but not when you use it in another view entity
D. It is no longer possible to pass your own value to the parameter.
Answer: A,B (LEAVE A REPLY)
Explanation

The annotation @Environment.systemField: #LANGUAGE is used to assign the ABAP system field sy-langu to an input parameter of a CDS view or a CDS table function. This enables the implicit parameter passing in Open SQL, which means that the value of sy-langu will be automatically passed to the CDS view without explicitly specifying it in the WHERE clause. This also applies to the CDS views that use the annotated CDS view as a data source, which means that the value of sy-langu will be propagated to the nested CDS views (view on view)12. For example:
The following code snippet defines a CDS view ZI_FLIGHT_TEXTS with an input parameter p_langu that is annotated with @Environment.systemField: #LANGUAGE:
define view ZI_FLIGHT_TEXTS with parameters p_langu : syst_langu
@<Environment.systemField:
#LANGUAGE as select from sflight left outer join scarr on sflight.carrid = scarr.carrid left outer join stext on scarr.carrid = stext.carrid { sflight.carrid, sflight.connid, sflight.fldate, scarr.carrname, stext.text as carrtext } where stext.langu = :p_langu The following code snippet shows how to use the CDS view ZI_FLIGHT_TEXTS in ABAP without specifying the value of p_langu in the WHERE clause. The value of sy-langu will be automatically passed to the CDS view:
SELECT carrid, connid, fldate, carrname, carrtext FROM zi_flight_texts INTO TABLE @DATA(lt_flights).
The following code snippet shows how to use the CDS view ZI_FLIGHT_TEXTS in another CDS view ZI_FLIGHT_REPORT. The value of sy-langu will be automatically passed to the nested CDS view ZI_FLIGHT_TEXTS:
define view ZI_FLIGHT_REPORT with parameters p_langu : syst_langu
@<Environment.systemField:
#LANGUAGE as select from zi_flight_texts(p_langu) { carrid, connid, fldate, carrname, carrtext, count(*) as flight_count } group by carrid, connid, fldate, carrname, carrtext The annotation @Environment.systemField: #LANGUAGE does not prevent the possibility of overriding the default value with a value of your own. You can still specify a different value for the input parameter p_langu in the WHERE clause, either in ABAP or in another CDS view. This will override the value of sy-langu and pass the specified value to the CDS view12. For example:
The following code snippet shows how to use the CDS view ZI_FLIGHT_TEXTS in ABAP with a specified value of p_langu in the WHERE clause. The value 'E' will be passed to the CDS view instead of the value of sy-langu:
SELECT carrid, connid, fldate, carrname, carrtext FROM zi_flight_texts WHERE p_langu = 'E' INTO TABLE @DATA(lt_flights).
The following code snippet shows how to use the CDS view ZI_FLIGHT_TEXTS in another CDS view ZI_FLIGHT_REPORT with a specified value of p_langu in the WHERE clause. The value 'E' will be passed to the nested CDS view ZI_FLIGHT_TEXTS instead of the value of sy-langu:
define view ZI_FLIGHT_REPORT with parameters p_langu : syst_langu
@<Environment.systemField:
#LANGUAGE as select from zi_flight_texts(p_langu) { carrid, connid, fldate, carrname, carrtext, count(*) as flight_count } where p_langu = 'E' group by carrid, connid, fldate, carrname, carrtext

References: 1: ABAP CDS - parameter_annot - ABAP Keyword Documentation - SAP Online Help 2: ABAP CDS - session_variable - ABAP Keyword Documentation - SAP Online Help

NEW QUESTION: 60



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)

Explanation

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 componentsof 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: 61

What is the purpose of a foreign key relationship between two tables in the ABAP Dictionary?

A. To document the relationship between the two tables
B. To ensure the integrity of data in the corresponding database tables
C. To create a corresponding foreign key relationship in the database
Answer: B (LEAVE A REPLY)
Explanation
The purpose of a foreign key relationship between two tables in the ABAP Dictionary is to ensure the integrity of data in the corresponding database tables. A foreign key relationship defines a logical link between a foreign key table and a check table, where the foreign key fields of the former are assigned to the primary key fields of the latter. This means that the values entered in the foreign key fields must exist in the check table, otherwise the system will reject the entry. This way, the foreign key relationship prevents the insertion of invalid or inconsistent data in the database tables.
A foreign key relationship also serves to document the relationship between the two tables in the ABAP Dictionary, but this is not its primary purpose. A foreign key relationship does not necessarily create a corresponding foreign key relationship in the database, as this depends on the database system and the settings of the ABAP Dictionary. Some database systems do not support foreign keys at all, while others require additional steps to activate them. Therefore, the foreign key relationship in the ABAP Dictionary is mainly a logical concept that is enforced by the ABAP runtime environment.
References: Foreign Keys (SAP Library - ABAP Dictionary), Foreign Keys (SAP Library - BC - ABAP Dictionary)
https://help.sap.com/doc/saphelp_snc70/7.0/en-US/cf/21ea77446011d189700000e8322d00/content.htm

NEW QUESTION: 62
Which internal table type allows unique and non-unique keys?
A. Sorted
B. Hashed
C. Standard
Answer: C (LEAVE A REPLY)
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.

NEW QUESTION: 63
Which of the following actions cause an indirect change to a database table requiring a table conversion? Note: There are 2 correct answers to this question.
A. Renaming a field in a structure that is included in the table definition
B. Changing the field labels of a data element that is used in the table definition.
C. Deleting a field from a structure that is included in the table definition.
D. Shortening the length of a domain used in a data element that is used in the table definition.
Answer: A,C (LEAVE A REPLY)
The following are the explanations for each action:
A: Renaming a field in a structure that is included in the table definition causes an indirect change to the database table, as the field name in the table is derived from the structure. This change requires a table conversion, as the existing data in the table must be copied to a new table with the new field name, and the old table must be deleted.
B: Changing the field labels of a data element that is used in the table definition does not cause an indirect change to the database table, as the field labels are only used for documentation and display purposes. This change does not require a table conversion, as the existing data in the table is not affected by the change.
C: Deleting a field from a structure that is included in the table definition causes an indirect change to the database table, as the field is removed from the table as well. This change requires a table conversion, as the existing data in the table must be copied to a new table without the deleted field, and the old table must be deleted.
D: Shortening the length of a domain used in a data element that is used in the table definition causes an indirect change to the database table, as the field length in the table is derived from the domain. This change requires a table conversion, as the existing data in the table must be checked for compatibility with the new field length, and any data that exceeds the new length must be truncated or rejected.

NEW QUESTION: 64

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)

Explanation

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: 65
Which of the following string functions are predicate functions? Note: There are 2 correct answers to this question.
A. find_any_not_of()
B. contains_any_of()
C. count_any_of()
D. matchesQ
Answer: B,D (LEAVE A REPLY)
String functions are expressions that can be used to manipulate character-like data in ABAP. String functions can be either predicate functions or non-predicate functions. Predicate functions are string functions that return a truth value (true or false) for a condition of the argument text. Non-predicate functions are string functions that return a character-like result for an operation on the argument text1.
The following string functions are predicate functions:
B) contains_any_of(): This function returns true if the argument text contains at least one of the characters specified in the character set. For example, the following expression returns true, because the text 'ABAP' contains at least one of the characters 'A', 'B', or 'C':
contains_any_of(val = 'ABAP' set = 'ABC').
D) matches(): This function returns true if the argument text matches the pattern specified in the regular expression. For example, the following expression returns true, because the text 'ABAP' matches the pattern that consists of four uppercase letters:
matches(val = 'ABAP' regex = '[A-Z]{4}').
The following string functions are not predicate functions, because they return a character-like result, not a truth value:
A) find_any_not_of(): This function returns the position of the first character in the argument text that is not contained in the character set. If no such character is found, the function returns 0. For example, the following expression returns 3, because the third character of the text 'ABAP' is not contained in the character set 'ABC':
find_any_not_of(val = 'ABAP' set = 'ABC').
C) count_any_of(): This function returns the number of characters in the argument text that are contained in the character set. For example, the following expression returns 2, because there are two characters in the text 'ABAP' that are contained in the character set 'ABC':
count_any_of(val = 'ABAP' set = 'ABC').