DUMPSARENA

DB2 9.7 Application Development

IBM C2090-543

Version Demo

Total Demo Questions: 12

Total Premium Questions: 120

Buy Premium PDF

https://dumpsarena.co

sales@dumpsarena.co

sales@dumpsarena.co
dumpsarena.co

Topic Break Down

Topic	No. of Questions
Topic 1, SQL	55
Topic 2, XQuery	12
Topic 3, Application Development	53
Total	120

QUESTION NO: 1

Which two commands can be used to make use of the latest statistics? (Choose two.)

A. INSPECT

B. FLUSH PACKAGE CACHE

C. REBIND

D. RUNSTATS

ANSWER: B C

Explanation:

`FLUSH PACKAGE CACHE` and `REBIND` can cause Db2 to use newly available optimizer statistics. Db2 uses statistics when it compiles or recompiles an access plan. Dynamic statements can have their compiled sections retained in the package cache; issuing `FLUSH PACKAGE CACHE` removes cached sections, so subsequent executions must compile again and can select plans based on the current catalog statistics. This is useful when a workload uses dynamic SQL and a plan needs to be regenerated after statistics have been refreshed.

`REBIND` recompiles the SQL statements in an existing package. Since static SQL access plans are saved with the package, rebinding is the action that makes those statements eligible for optimization using the current statistics. It is therefore the appropriate package-management command after statistics changes when an application depends on static SQL. The Db2 command reference describes the package-cache flush operation and its effect on cached sections, while the rebinding documentation explains that package SQL statements are rebound and their access plans regenerated. See IBM's FLUSH PACKAGE CACHE command reference and REBIND command reference.

QUESTION NO: 2

.

While developing a CLI application, you use the code shown below:

SQLCHAR *stmt = (SQLCHAR *)"DELETE FROM org WHERE deptnumb = ? ";

cliRC = SQLSetStmtAttr(hstmt, SQL_ATTR_DEFERRED_PREPARE, SQL_DEFERRED_PREPARE_ON)

cliRC = SQLPrepare(hstmt, stmt, SQL_NTS);

Now, the ORG table does not exist in the database.

What will be the value of "cliRC" after executing the SQLPrepare command?

A. SQL_SUCCESS

B. SQL_SUCCESS_WITH_INFO

C. SQL_INVALID_HANDLE

D. SQL_ERROR

ANSWER: A

Explanation:

SQL_SUCCESS is correct because the statement attribute `SQL_ATTR_DEFERRED_PREPARE` is set to `SQL_DEFERRED_PREPARE_ON` before `SQLPrepare` is called. With deferred prepare enabled, Db2 CLI does not immediately send the prepare request to the database server. Instead, it retains the SQL text locally and defers the server-side prepare until the application executes the statement, typically through `SQLExecute` or `SQLExecDirect`. Consequently, at the point where `SQLPrepare` returns, Db2 has not yet validated the referenced ORG table at the server. The missing table therefore does not affect the return code from this `SQLPrepare` call, and `cliRC` receives `SQL_SUCCESS`. When execution later

causes the deferred prepare to occur, Db2 validates the statement and resolves the table reference; that later operation reports the error associated with the nonexistent table. Deferred preparation can reduce network traffic and avoid unnecessary prepare activity for statements that might not ultimately be executed. IBM documents this behavior for the deferred-prepare statement attribute and for the CLI `SQLPrepare` function: SQL_ATTR_DEFERRED_PREPARE and SQLPrepare.

QUESTION NO: 3

An SQL procedure must return a user-defined error condition to its invoking application. Which two SQL PL statements can be used to generate or propagate an SQLSTATE condition? (Choose two.)

A. SIGNAL

B. RESIGNAL

C. DECLARE CONDITION

D. DECLARE CONTINUE HANDLER

ANSWER: A B

Explanation:

`SIGNAL` raises a condition explicitly. An SQL procedure can use `SIGNAL SQLSTATE` to return a user-defined SQLSTATE and, optionally, diagnostic information such as a message text to the invoking application. This is useful when a business rule detects an invalid condition that must stop normal processing.

`RESIGNAL` is used within a condition handler to propagate the condition being handled, optionally changing its SQLSTATE or diagnostic information. It allows a handler to perform local work and then return the condition to an outer handler or to the calling application. `DECLARE CONDITION` only associates a name with an SQLSTATE, while `DECLARE CONTINUE HANDLER` establishes handling behavior; neither statement itself raises a condition. See the IBM documentation for the SIGNAL statement and the RESIGNAL statement.

QUESTION NO: 4

Which two techniques guarantee that XML document validation will be performed? (Choose two.)

A. Implicit use of the XMLCHECK() function in the SQL statements.

B. Use triggers for INSERT and UPDATE operations with the XMLVALIDATE() function.

C. Implicit use of the XMLVALIDATE() function in SQL statements.

D. Use triggers for INSERT and UPDATE operations with the XMLSERIALIZE() function.

ANSWER: B C

Explanation:

Use triggers for INSERT and UPDATE operations with the XMLVALIDATE() function guarantees validation because the trigger executes as part of each applicable data-change operation and invokes the Db2 XML validation operation before the change can complete. This is an effective way to enforce a validation rule consistently when applications might otherwise submit XML without performing validation themselves. The trigger must cover both INSERT and UPDATE, since either operation can introduce a new or changed XML value.

Implicit use of the XMLVALIDATE() function in SQL statements also guarantees validation when Db2 requires the value to conform to the applicable registered XML schema. `XMLVALIDATE` validates an XML document against an XML schema and annotates the resulting XML value with the schema information used by Db2. Consequently, validation is performed as part of the SQL expression evaluation rather than being left to an external application convention. Db2 documents the function's

schema-based validation behavior in its XMLVALIDATE function reference and describes schema validation and typed XML processing in the XML schema validation documentation.

QUESTION NO: 5

The table shown below contains a large number of financial transactions:

CREATE TABLE webstore.transactions (transaction_id INTEGER NOT NULL PRIMARY KEY, order_date TIMESTAMP NOT NULL, shipped_date TIMESTAMP, customer_id INTEGER NOT NULL, shipping_info XML NOT NULL, billing_info XML NOT NULL, invoice XML NOT NULL)

Only members of the AUDIT_TEAM group have SELECT privilege on the WEBSTORE.TRANSACTIONS table. For appropriate supply-chain management, members of the INVENTORY_CONTROL group need to see the INVOICE document for each transaction that has a NULL SHIPPED_DATE, but are restricted from seeing any shipping or billing information.

Which database object can a member of the AUDIT_TEAM group create to enable the INVENTORY_CONTROL group to access the information needed from WEBSTORE.TRANSACTIONS?

A. alias

B. sequence

C. trigger

D. view

ANSWER: D

Explanation:

The correct object is a view. A view can define a restricted, named result set over `WEBSTORE.TRANSACTIONS` that exposes only the `INVOICE` column and includes only rows whose `SHIPPED_DATE` is NULL. For example, an authorized user can create a view using a query conceptually equivalent to `CREATE VIEW webstore.unshipped_invoices AS SELECT invoice FROM webstore.transactions WHERE shipped_date IS NULL`, then grant `SELECT` on that view to `INVENTORY_CONTROL`. This gives the group access to precisely the documents required for supply-chain work without granting direct `SELECT` access on the base table or exposing the `SHIPPING_INFO` and `BILLING_INFO` XML columns. Db2 views are specifically intended to provide controlled access to selected columns and rows from one or more underlying tables. The view owner must have the necessary privilege on the underlying table, and the view can subsequently be used as the grantable interface for other users or groups. The row predicate remains part of the view definition, ensuring that queries through the view are limited to unshipped transactions. See IBM's documentation on CREATE VIEW and GRANT (table, view, or nickname privileges).

QUESTION NO: 6

Which three column definitions can be used in an XMLTABLE COLUMNS clause? (Choose three.)

A. line_no FOR ORDINALITY

B. product_name VARCHAR(30) PATH 'name'

C. status VARCHAR(10) DEFAULT 'NEW'

D. price DECIMAL(9,2) FROM './price'

E. item_id INTEGER USING '@id'

ANSWER: A B C

Explanation:

An XMLTABLE column can be defined with `FOR ORDINALITY` to generate an integer value representing the position of each row generated by XMLTABLE. This is useful when the application must preserve the sequence of repeated XML elements.

A column can also use a `PATH` expression to extract a value from the XML node selected for the XMLTABLE row. A `DEFAULT` clause is valid for supplying a value when the path expression returns an empty sequence. These clauses enable XMLTABLE to convert selected XML nodes into relational columns. See the IBM documentation for XMLTABLE.

QUESTION NO: 7

Which two Java objects can be used to produce a result set using a SELECT statement? (Choose two.)

A. Statement

B. PreparedStatement

C. ResultSet

D. CallableStatement

ANSWER: A B

Explanation:

`Statement` and `PreparedStatement` are the JDBC objects intended to execute a SQL `SELECT` statement and return its rows as a `ResultSet`. A `Statement` is appropriate when the SQL text can be issued directly, such as `statement.executeQuery("SELECT ...")`. Its `executeQuery` method executes a SQL query and returns the resulting `ResultSet`, which the application then reads using cursor methods such as `next()` and typed getter methods.

A `PreparedStatement` is a specialized statement that represents SQL prepared in advance. It is especially useful for parameterized SELECT operations: the application defines SQL containing parameter markers, supplies values through setter methods, and calls `executeQuery()` to obtain the `ResultSet`. This model supports efficient repeated execution and safely separates parameter values from the SQL command text. IBM Db2 JDBC applications use these standard JDBC interfaces to submit queries and process returned rows. The Java API documentation describes `Statement.executeQuery` and `PreparedStatement.executeQuery` as returning a single `ResultSet` for a query. See the Java Statement API and IBM's Db2 Java application programming documentation.

QUESTION NO: 8

The table shown below exists within a database:

CREATE TABLE s1.t1 (c1 INTEGER NOT NULL PRIMARY KEY, c2 CLOB(2G))

A CLI/ODBC application uses the SQLGetSubString() API to retrieve the first 1024 and last 1024 bytes of data from column C2.

Which mechanism is used to minimize the amount of data sent to the client for this operation?

A. module

B. temp file

C. locator

D. log file

ANSWER: C

Explanation:

locator is correct. A CLOB can contain very large values, and transferring an entire 2 GB value merely to obtain two 1024-byte portions would be unnecessarily expensive. Db2 uses a LOB locator as a handle that represents the large object at the server rather than materializing the complete LOB value in the application's client memory. The application can then request specified portions of the LOB through `SQLGetSubString()`, supplying the relevant start position and length. Db2 returns only the requested substring data, so the first and last portions can be retrieved without sending the intervening contents across the client-server connection.

This locator-based access is particularly appropriate for random or partial access to large character and binary objects. The locator identifies the CLOB during the applicable unit of work, while the substring API lets the client retrieve the needed segments incrementally. Thus, the amount of network data is driven by the requested 1024-byte sections rather than by the CLOB's total size. IBM documents `SQLGetSubString()` as the CLI function for obtaining part of a LOB value, and its Db2 LOB documentation describes locators as values used to refer to LOB data. See the IBM SQLGetSubString documentation and the IBM large objects documentation.

QUESTION NO: 9

.

Click the Exhibit button.

John Wayne

New York

US

850-734-6672

796-858-1272

646-252-1053

Barbara Wayne

New York

US

850-734-6672

796-858-1231

646-252-1153

The table PERSON is declared as shown below:

CREATE TABLE person (id BIGINT, info XML)

The documents shown in the exhibit are successfully inserted into the table.

How many phone numbers will be affected by the statement shown below?

UPDATE xmlapp.person

SET info = xmlquery('copy $new := $INFO

modify for $j in $new/personinfo//phone return

do replace value of $j with "444-444-4444"

return $new')

WHERE XMLEXISTS('$INFO/personinfo[name="John Wayne"]')

A. 0

B. 3

C. 4

D. 6

ANSWER: B

Explanation:

3 is correct. The `XMLEXISTS` predicate limits the SQL UPDATE to the XML document whose `personinfo` element has a `name` child with the value `John Wayne`. For that qualifying document, the XQuery copy-modify-return expression first makes a copy of the XML value in `$INFO` and binds it to `$new`. This is the appropriate pattern for applying XQuery Update Facility operations while returning a revised XML value for assignment back to the XML column.

The path `$new/personinfo//phone` selects every descendant `phone` element under `personinfo`, regardless of whether the element has a `type` attribute. In John Wayne's document, there are three such elements: the untyped phone number, the mobile phone number, and the security phone number. The `for` expression iterates over all three selected nodes, and `replace value of` changes the text value of each one to `444-444-4444`. Thus, three phone-number values are affected, while the other XML document is not selected by the WHERE condition.

See IBM's documentation for XMLEXISTS and the XQuery transform expression.

QUESTION NO: 10

Which statement(s) will create and bind the package for the program myprogram.sqc?

A. BIND myprogram.bnd

B. PRECOMPILE myprogram.sqc

C. gcc myprogram.sqc -o myprogram
BIND myprogram.bnd

D. gcc myprogram.sqc -o myprogram
REBIND myprogram.sqc

ANSWER: B

Explanation:

`PRECOMPILE myprogram.sqc` is correct because the Db2 precompile operation processes an embedded-SQL source file and, by default, performs the associated package bind processing. Precompilation recognizes and validates the SQL statements in `myprogram.sqc`, replaces embedded SQL with the appropriate Db2 runtime calls in the generated host-language source, and produces the database access information needed for the application package. The package contains the executable form of the application's static SQL and is bound to the target database as part of normal precompile processing unless an option is supplied to defer binding and produce a bind file for separate execution.

This behavior allows an embedded SQL application to be prepared for compilation while its SQL is made available to Db2 through the package. After precompilation, the generated C source must still be compiled and linked with the appropriate Db2 application libraries before the executable can run, but those build steps are separate from creating and binding the package. IBM documents the PRECOMPILE command and its package/bind-file behavior in the Db2 PRECOMPILE command reference. The broader embedded SQL build sequence is also described in IBM's embedded SQL application build documentation.

QUESTION NO: 11

Given the application shown below:

DECLARE csr1 CURSOR WITH HOLD FOR SELECT * FROM employee;

OPEN csr1;

FETCH csr1;

COMMIT;

CLOSE csr1;

How long does cursor CSR1 remain open?

A. CSR1 remains open until a COMMIT is executed.

B. CSR1 remains open until the last row is read from the result set.

C. CSR1 remains open until CLOSE csr1 is executed.

D. CSR1 remains open until a cursor with a different name is opened.

ANSWER: C

Explanation:

CSR1 remains open until CLOSE csr1 is executed. The cursor is declared using `WITH HOLD`, which gives it holdable-cursor behavior. In Db2, a holdable cursor is not closed when a `COMMIT` operation completes. Thus, after the application opens CSR1, fetches a row, and executes `COMMIT`, the cursor remains open and can continue to be used for further fetches. Its current position is retained through the commit, although Db2 may release locks associated with rows that have already been fetched, subject to the applicable isolation level and cursor characteristics. The subsequent explicit `CLOSE csr1` statement is therefore the event in this program that closes the cursor and releases cursor resources. This behavior is useful when an application needs to commit a unit of work while continuing to process a result set. Db2 documents the `WITH HOLD` clause as specifying that the cursor remains open across a commit operation. See the IBM documentation for DECLARE CURSOR and the Db2 overview of cursor characteristics.

QUESTION NO: 12

A CLI/ODBC application must obtain catalog information about tables and columns available through a database connection.

Which two APIs can be used? (Choose two.)

A. SQLTables()

B. SQLColumns()

C. SQLGetData()

D. SQLFetch()

ANSWER: A B

Explanation:

`SQLTables()` returns information about tables, views, aliases, and other eligible table-type objects. The application can supply catalog, schema, table-name, and table-type search patterns to limit the metadata returned. The result is made available through a result set associated with the statement handle.

`SQLColumns()` returns metadata about columns of tables and views that match the supplied search criteria. Its result set includes information such as column names, SQL data types, sizes, nullability, and ordinal positions. `SQLGetData()` retrieves data values from an already fetched result row, while `SQLFetch()` advances a cursor and does not itself request catalog metadata. See the IBM documentation for SQLTables and SQLColumns.