Exam Ref 70-762 Developing SQL Databases

Authors: Louis Davidson and Stacia Varga

|  |  |  |  |
| --- | --- | --- | --- |
| Status | Page | Print | Correction |
|  | 20, last paragraph | 1 | **Change:**Now the expression be evaluated during access in a statement, but is ...**To:**Now the expression **is not** evaluated during access in a statement, but is ... |
|  | 34, paragraph below Figure 1-4, third sentence | 1 | **Change**: SQL Server now uses the index-seek operation to find the six matching rows, but all it has are the CustomerID and the OrderID from the index keys.**To:**SQL Server now uses the index-seek operation to find the six matching rows, but all it has are the **CustomerPurchaseOrderNumber** and the OrderID from the index keys. |
|  | 35, second sentence | 1 | **Change:** You can tell if a column can be indexed, even if it is computed by using the **COLUMNPROPERTYEX()** function:**To:** You can tell if a column can be indexed, even if it is computed by using the **COLUMNPROPERTYEX** function: |
|  | 49, Listing 1-1, first four lines | 1 | **Change:** --2074 RowsSELECT \*INTO Examples.PurchaseOrdersFROM WideWorldImporters.Purchasing.PurchaseOrders;**To:** **CREATE SCHEMA Examples AUTHORIZATION dbo;****GO**--2074 RowsSELECT \*INTO Examples.PurchaseOrdersFROM WideWorldImporters.Purchasing.PurchaseOrders; |
|  | 52, third bullet | 1 | **Change:** Encrypts the entry in sys.syscomments that contains the text of the VIEW create statement.**To:** Encrypts the text of the VIEW object. |
|  | 52, first bullet (Schema binding), last sentence | 1 | **Change:**Columns, not references can be removed, or new columns added.**To:** Columns **not referenced** can be removed, or new columns added. |
|  | 58, Listing 1-6, fourth line | 1 | **Change:** UPPER(GadgetType) AS UpperGadgedType**To:** UPPER(GadgetType) AS **UpperGadgetType** |
|  | 59, code after 3rd paragraph, 2nd and 3rd line of code | 1 | **Change:** replace curly quotes in code with straight quotesVALUES (4,’00000004’,’Electronic’,’XXXXXXXXXX’), --row we can see in view(5,’00000005’,’Manual’,’YYYYYYYYYY’); --row we cannot see in view**To:**  Replace curly quotes in above code with straight quotes |
|  | 89, bottom, Listing 1-18 | 1 | On the first line of Listing 1-18 there's a dot at the first character:**Change:** .CREATE TABLE [Fact].[SaleLimited](**To:** CREATE TABLE [Fact].[SaleLimited]( |
|  | 99, last sentence before bullets | 1 | **Change:**- The PRIMARY KEY constraint is assumed to have caused the creation of a clustered index as it is included on the CREATE TABLE- The CREATE CLUSTERED COLUMNSTORE INDEX will expectadly fail with the following message: Msg 35372, Level 16, State 3, Line 10**To:**expectedlyAnd: Add note below to the CREATE CLUSTERED COLUMNSTORE INDEX bullet on page 100: “Note that you would need to change the PRIMARY KEY constraint to nonclustered for this one to work.” |
|  | 100, last sentence | 1 | **Change:** It works nicely with the PRIMARY KEY constraint index to allow singleton updates/seeks as needed for ETL and simple queries also.Yet:- The PRIMARY KEY constraint is assumed to have caused the creation of a clustered index as it is included on the CREATE TABLE- The CREATE CLUSTERED COLUMNSTORE INDEX will fail: Msg 35372, Level 16, State 3, Line 10 You cannot create more than one clustered index on table 'Sales.InvoiceItemFact'. Consider creating a new clustered index using 'with (drop\_existing = on)' option. |
|  | 106, after middle of page | 1 | **Change:** Now, an attempt to insert a row with the duplicated tag value of G001:INSERT INTO Equipment.Tag(Tag, TagCompanyId)VALUES ('G001',1);**To:**Now, an attempt to insert a row with the duplicated GadgetCode value of Gadget:INSERT INTO Examples.Gadget(GadgetCode)VALUES ('Gadget');  |
|  | 106, middle – second bullet | 1 | **Change:**The columns of the key allow NULL values (NULL values are treated as distinct values, ...**To:**The columns of the key allow NULL values (yet, quite unusually, NULL values are treated as equal on this case) |
|  | 113, first line | 1 | **Change:** ALTER TABLE Alt.TwoPartKeyReference**To:** ALTER TABLE Examples.TwoPartKeyReference |
|  | 118, middle | 1 | **Change:**There's an extra space between the database schema name and the table name: ALTER TABLE Examples. Attendee**To:** ALTER TABLE Examples.Attendee |
|  | 124, 9th line | 1 | **Change:** There is a missing parentheses: CHECK ScenarioTestType IN ('Type1','Type2'))**To:** CHECK (ScenarioTestType IN ('Type1','Type2')) |
|  | 222, “Repeatable Read section,” first sentence | 1 | **Change:**The behavior of the REPEATABLE READ isolation level is much like that of READ COMMITTED, except that it ensures that multiple reads of the same data within a transaction **is** consistent.**To:** The behavior of the REPEATABLE READ isolation level is much like that of READ COMMITTED, except that it ensures that multiple reads of the same data within a transaction **are** consistent.  |
|  | 222, “Repeatable Read section,” bottom, last paragraph before code  | 1 | **Change:**In this case, the first read operations blocks the update operation, which executes when the **first read’s** locks are released**, the update commits the data change, but the second query returns the same rows as the first query due to the isolation level of the transaction**:**To:**In this case, the first read operation blocks the update operation, which executes when the **transaction's** locks are released. **The update in the second session commits the data change after the transaction. Both queries in the first session return the same results because the update occurs after the transaction ends:** |
|  | 223, top, above the “Serializable” heading | 1 | **Change:** Add new paragraph:**If you were to execute an INSERT statement in the second session instead, the first and second queries in the first session return different results by including the new rows in the second query. The isolation level prevents changes to existing data, but allows the insertion of new data.** |