Certshared now are offering 100% pass ensure 70-762 dumps!
https://www.certshared.com/exam/70-762/ (160 Q&As)

Microsoft

Exam Questions 70-762
Developing SQL Databases (beta)
NEW QUESTION 1
You are designing a stored procedure for a database named obi.
The following requirements must be met during the entire execution of the stored procedure:
*The stored procedure must only read changes that are persisted to the database.
*select statements within the stored procedure should only show changes to the data that are made by the stored procedure
You need to configure the transaction isolation level for the stored procedure. Which Transact-SQL statement or statements should you run?
A)  
B)  
C)  
D)  

A. Option A  
B. Option B  
C. Option C  
D. Option D  

Answer: A

NEW QUESTION 2
You are analyzing the performance of a database environment.
Applications that access the database are experiencing locks that are held for a large amount of time. You are experiencing isolation phenomena such as dirty, nonrepeatable and phantom reads.
You need to identify the impact of specific transaction isolation levels on the concurrency and consistency of data.
What are the consistency and concurrency implications of each transaction isolation level? To answer, drag the appropriate isolation levels to the correct locations.
Each isolation level may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.

Answer:

Explanation: Read Uncommitted (aka dirty read): A transaction T1 executing under this isolation level can access data changed by concurrent transaction(s).
Pros: No read locks needed to read data (i.e. no reader/writer blocking). Note, T1 still takes transaction duration locks for any data modified.
Cons: Data is not guaranteed to be transactionally consistent.

Read Committed: A transaction T1 executing under this isolation level can only access committed data.
Pros: Good compromise between concurrency and consistency.
Cons: Locking and blocking. The data can change when accessed multiple times within the same transaction.

Repeatable Read: A transaction T1 executing under this isolation level can only access committed data with an additional guarantee that any data read cannot change (i.e. it is repeatable) for the duration of the transaction.
Pros: Higher data consistency.
Cons: Locking and blocking. The S locks are held for the duration of the transaction that can lower the concurrency. It does not protect against phantom rows.

Serializable: A transaction T1 executing under this isolation level provides the highest data consistency including elimination of phantoms but at the cost of reduced concurrency. It prevents phantoms by taking a range lock or table level lock if range lock can't be acquired (i.e. no index on the predicate column) for the
duration of the transaction.
Pros: Full data consistency including phantom protection.
Cons: Locking and blocking. The S locks are held for the duration of the transaction that can lower the concurrency.
References:

NEW QUESTION 3
You have multiple stored procedures inside a transaction.
You need to ensure that all the data modified by the transaction is rolled back if a stored procedure causes a deadlock or times out.
What should you do?
A. Use the NOLOCK option.
B. Execute the DBCC UPDATEUSAGE statement.
C. Use the max worker threads Option.
D. Use a table-valued parameter.
E. Set SET ALLOW SNAPSHOT ISOLATION to ON,
F. Set SET XACT ABORT to ON.
G. Execute the alter table T1 set (lock escalation = auto); statement.
H. Use the output parameters.
Answer: B

NEW QUESTION 4
You use Query Store to optimize a query in a database. The query has two execution plans:
Plan 10 is shown in the Plan 10 Execution Plan exhibit.
You create an index at 22:24 based on the missing index suggestion in Plan 2.
The average duration statistics for the query is shown in the Tracked Queries exhibit. (Click the Exhibit button.)
You need to analyze the operators in the two execution plans.
For each of the following statements, select Yes if the statement is true. Otherwise, select No. Plan 2 Execution Plan

Plan 10 Execution Plan

Tracked Queries
NEW QUESTION 5

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.

After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.

You have a database that is 130 GB and contains 500 million rows of data. Granular transactions and mass batch data imports change the database frequently throughout the day. Microsoft SQL Server Reporting Services (SSRS) uses the database to generate various reports by using several filters.

You discover that some reports time out before they complete. You need to reduce the likelihood that the reports will time out.

Solution: You increase the number of log files for the database. You store the log files across multiple disks. Does this meet the goal?

A. Yes
B. No

Answer: A

NEW QUESTION 6

You have the following stored procedure that is called by other stored procedures and applications:
CREATE PROCEDURE UpdateCustomer @CustomerId INT
AS
BEGIN
EXEC ProcessCustomer_Internal @CustomerId
DECLARE @Status INT
SELECT @Status = Status FROM Customer WHERE CustomerId = @CustomerId
END
You need to modify the stored procedure to meet the following requirements:
Always return a value to the caller.
Return 0 if @Status is NULL.
Callers must be able to use @Status as a variable.
Which two actions should you perform? Each correct answer presents part of the solution. NOTE: Each correct selection is worth one point.

A. Replace NULL values with 0. Add a PRINT statement to return @Status.
B. Add a RETURN statement.
C. Replace NULL values with 0. Add an output parameter to return @Status.
D. Replace NULL values with 0. Add a SELECT statement to return @Status.
E. Add a PRINT statement.
F. Add a SELECT statement to return @Status.
G. Add an output parameter to return @Status.

Answer: BC

Explanation: There are three ways of returning data from a procedure to a calling program: result sets, output parameters, and return codes.
References:

NEW QUESTION 7
Note: This question is part of a series of questions that present the same scenario. Each question in this series contains a unique solution. Determine whether the solution meets the stated goals.
The Account table was created by using the following Transact-SQL statement:

CREATE TABLE Account
(
    AccountNumber int NOT NULL,
    ProductCode char(2) NOT NULL,
    Status tinyint NOT NULL,
    OpenDate date NOT NULL,
    CloseDate date,
    Balance decimal(15,2),
    AvailableBalance decimal(15,2)
);

There are more than 1 billion records in the Account table. The Account Number column uniquely identifies each account. The ProductCode column has 100 different values. The values are evenly distributed in the table. Table statistics are refreshed and up to date.
You frequently run the following Transact-SQL SELECT statements:

SELECT ProductCode, SUM(Balance) AS TotalSum FROM Account WHERE ProductCode <> 'CD' GROUP BY ProductCode;
SELECT AccountNumber, Balance FROM Account WHERE ProductCode = 'CD'

You must avoid table scans when you run the queries. You need to create one or more indexes for the table. Solution: You run the following Transact-SQL statement:

CREATE NONCLUSTERED INDEX PK Account ON Account(AccountNumber);
CREATE NONCLUSTERED INDEX IX_Account_ProductCode ON Account(ProductCode) INCLUDE (Balance);

Does the solution meet the goal?

A. Yes
B. No

Answer: B

Explanation: Create a clustered index on the AccountNumber column as it is unique, not a non nonclustered one. References: https://msdn.microsoft.com/en-us/library/ms190457.aspx
NEW QUESTION 8
Note: This question is part of a series of questions that use the same answer choices. An answer choice may be correct for more than one question on the series. Each question is independent of the other questions in this series. Information and details provided in a question apply only to that question.
You work on an OLTP database that has no memory-optimized file group defined.
You have a table named tblTransaction that is persisted on disk and contains the information described in the following table:

<table>
<thead>
<tr>
<th>Column</th>
<th>Name</th>
<th>Data Type</th>
<th>Nullable</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column</td>
<td>TransactionDate</td>
<td>Date</td>
<td>No</td>
<td>For each transaction date, there are only about 100,000 records. The table contains over one billion records in total.</td>
</tr>
<tr>
<td>Column</td>
<td>SequenceNo</td>
<td>bigint</td>
<td>No</td>
<td>Uniquely identifies a transaction record within a date</td>
</tr>
<tr>
<td>Column</td>
<td>AccountId</td>
<td>int</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Column</td>
<td>ValueType</td>
<td>varchar</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Column</td>
<td>Amount</td>
<td>decimal</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>IX_ValueType</td>
<td></td>
<td></td>
<td></td>
<td>Nonclustered columnstore index on the ValueType column.</td>
</tr>
</tbody>
</table>

Users report that the following query takes a long time to complete.

SELECT TransactionDate, COUNT(*) AS TotalCount FROM tblTransaction
WHERE TransactionDate - DATETRUNC(D, -1, CONVERT(DATE, CONVERT(VARCHAR(8), GETDATE(), 112)))
GROUP BY TransactionDate;

You need to create an index that:
- improves the query performance
- does not impact the existing index
- minimizes storage size of the table (inclusive of index pages). What should you do?

A. Create a clustered index on the table.
B. Create a nonclustered index on the table.
C. Create a nonclustered filtered index on the table.
D. Create a clustered columnstore index on the table.
E. Create a nonclustered columnstore index on the table.
F. Create a hash index on the table.

Answer: C

Explanation: A filtered index is an optimized nonclustered index, especially suited to cover queries that select from a well-defined subset of data. It uses a filter predicate to index a portion of rows in the table. A well-designed filtered index can improve query performance, reduce index maintenance costs, and reduce index storage costs compared with full-table indexes.

NEW QUESTION 9
Note: This question is part of a series of questions that present the same scenario. Each question in this series contains a unique solution. Determine whether the solution meets the stated goals.
Your company has employees in different regions around the world.
You need to create a database table that stores the following employee attendance information:
- Employee ID
- date and time employee checked in to work
- date and time employee checked out of work
Date and time information must be time zone aware and must not store fractional seconds. Solution: You run the following Transact-SQL statement:

```
CREATE TABLE [dbo].[EmployeeAttendance] (EmployeeID int NOT NULL,
DateChekedIn datetimetz NOT NULL,
DateChechedOut datetimetz NOT NULL)
```

Does the solution meet the goal?

A. Yes
B. No

Answer: B

Explanation: Datetimetz, not datetimetz, defines a date that is combined with a time of a day that has time zone awareness and is based on a 24-hour clock.
Syntax: datetimetz [(fractional seconds precision)]
NEW QUESTION 10
Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution. Determine whether the solution meets the stated goals.

The Account table was created by using the following Transact-SQL statement:

```
CREATE TABLE Account
(AccountNumber Int NOT NULL,
ProductCode char(2) NOT NULL,
Status tinyint NOT NULL,
OpenDate date NOT NULL,
CloseDate date NOT NULL,
Balance decimal(15,2));
```

There are more than 1 billion records in the Account table. The AccountNumber column uniquely identifies each account. The ProductCode column has 100 different values. The values are evenly distributed in the table. Table statistics are refreshed and up to date.

You frequently run the following Transact-SQL select statements:

- `SELECT ProductCode, SUM(Balance) AS TotalAmount FROM Account WHERE ProductCode <> 'CD' GROUP BY ProductCode;` 
- `SELECT AccountNumber, Balance FROM Account WHERE ProductCode = 'CD';`

You must avoid table scans when you run the queries. You need to create one or more indexes for the table. Solution: You run the following Transact-SQL statements:

```
CREATE CLUSTERED INDEX PK_Account ON Account(AccountNumber);
CREATE NONCLUSTERED INDEX IX_Account_ProductCode ON Account(ProductCode) INCLUDE (Balance);
```

Does the solution meet the goal?

A. Yes
B. No

Answer: A

NEW QUESTION 11

You have a Microsoft SQL Server database that has a table named Sales. The table is used for retrieving data and is updated during non-business hours.

You run the following Transact-SQL statement:

```
SELECT ProductStandardCost, ((ProductStandardCost/TotalProductCost)*100) StandardCostRatio, ProductKey FROM Sales
ORDER BY ProductStandardCost ASC, ProductKey
```

You analyze the execution plan for the statement. (Click the Exhibit button).

You need to add an index that optimizes performance. How should you complete the Transact-SQL statement?

```
CREATE INDEX IX_Sales_1 ON Sales
(ProductStandardCost ASC, ProductKey ASC);
```

Answer:

Explanation:
NEW QUESTION 12
You run the following Transact-SQL statement:

```
CREATE TABLE OrderLines
(OrderLineID INT NOT NULL IDENTITY PRIMARY KEY CLUSTERED,
OrderID INT NOT NULL,
StockItemID INT NOT NULL,
Description NVARCHAR(100) NOT NULL,
Quantity INT NOT NULL,
UnitPrice DECIMAL(18, 2) NULL
)
```

There are multiple unique OrderID values. Most of the UnitPrice values for the same OrderID are different. You need to create a single index seek query that does not use the following operators:
- Nested loop
- Sort
- Key lookup

A. CREATE INDEX IX_OrderLines_l ON OrderLines (OrderID, UnitPrice) INCLUDE(Description, Quantity)
B. CREATE INDEX IX_OrderLines_l ON OrderLines (OrderID, UnitPrice) INCLUDE(Quantity)
C. CREATE INDEX IX_OrderLines_l ON OrderLines (OrderID, UnitPrice, Quantity)
D. CREATE INDEX IX_OrderLines_l ON OrderLines (UnitPrice, OrderID) INCLUDE(Description, Quantity)

**Answer:** A

NEW QUESTION 13
You are developing a database reporting solution for a table that contains 900 million rows and is 103 GB. The table is updated thousands of times a day, but data is not deleted. The SELECT statements vary in the number of columns used and the amount of rows retrieved. You need to reduce the amount of time it takes to retrieve data from the table. The must prevent data duplication.

Which indexing strategy should you use?

A. a nonclustered index for each column in the table
B. a clustered columnstore index for the table
C. a hash index for the table
D. a clustered index for the table and nonclustered indexes for nonkey columns

**Answer:** B

**Explanation:** Columnstore indexes are the standard for storing and querying large data warehousing fact tables. It uses column-based data storage and query processing to achieve up to 10x query performance gains in your data warehouse over traditional row-oriented storage. A clustered columnstore index is the physical storage for the entire table. Generally, you should define the clustered index key with as few columns as possible. A nonclustered index contains the index key values and row locators that point to the storage location of the table data. You can create multiple nonclustered indexes on a table or indexed view. Generally, nonclustered indexes should be designed to improve the performance of frequently used queries that are not covered by the clustered index.

References:

NEW QUESTION 14
You must create two staging database tables. The tables have the following requirements:
You need to select the correct storage mechanism for each table. Which storage mechanism should you use? To answer, drag the appropriate table types to the correct tables. Each table type may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.

**Table types**
- disk-based table
- local temporary table
- global temporary table
- memory-optimized non-durable table
- table variable
- memory-optimized table variable
- user-defined table

**Answer area**

<table>
<thead>
<tr>
<th>Table</th>
<th>Table type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table1</td>
<td></td>
</tr>
<tr>
<td>Table2</td>
<td></td>
</tr>
</tbody>
</table>

**Answer:**

**Explanation:**

NEW QUESTION 15

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution. Determine whether the
solution meets the stated goals.
You have a table that has a clustered index and a nonclustered index. The indexes use different columns from the table. You have a query named Query1 that uses the nonclustered index.
Users report that Query1 takes a long time to report results. You run Query1 and review the following statistics for an index seek operation:

<table>
<thead>
<tr>
<th>Physical Operation</th>
<th>Index Seek</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical Operation</td>
<td>Index Seek</td>
</tr>
<tr>
<td>Actual Execution Mode</td>
<td>Row</td>
</tr>
<tr>
<td>Actual Number of Rows</td>
<td>3571454</td>
</tr>
<tr>
<td>Actual Number of Batches</td>
<td>0</td>
</tr>
<tr>
<td>Estimated I/O Cost</td>
<td>0.0093577</td>
</tr>
<tr>
<td>Estimated Operator Cost</td>
<td>0.0107304 (0%)</td>
</tr>
<tr>
<td>Estimated CPU Cost</td>
<td>0.0013727</td>
</tr>
<tr>
<td>Estimated Subtree Cost</td>
<td>0.0107304</td>
</tr>
<tr>
<td>Estimated Number of Executions</td>
<td>1</td>
</tr>
<tr>
<td>Number of Executions</td>
<td>8</td>
</tr>
<tr>
<td>Estimated Number of Rows</td>
<td>0</td>
</tr>
<tr>
<td>Estimated Row Size</td>
<td>19 B</td>
</tr>
<tr>
<td>Actual Rebinds</td>
<td>0</td>
</tr>
<tr>
<td>Actual Rewinds</td>
<td>0</td>
</tr>
<tr>
<td>Ordered</td>
<td>True</td>
</tr>
<tr>
<td>Node ID</td>
<td>100</td>
</tr>
</tbody>
</table>

You need to resolve the performance issue.
Solution: You update statistics for the nonclustered index. Does the solution meet the goal?

A. Yes
B. No

Answer: A

Explanation: We see Actual Number of Row is 3571454, while Estimated Number of Rows is 0. This indicates that the statistics are old, and need to be updated.

NEW QUESTION 16

......
Thank You for Trying Our Product

We offer two products:

1st - We have Practice Tests Software with Actual Exam Questions

2nd - Questions and Answers in PDF Format

70-762 Practice Exam Features:

* 70-762 Questions and Answers Updated Frequently
* 70-762 Practice Questions Verified by Expert Senior Certified Staff
* 70-762 Most Realistic Questions that Guarantee you a Pass on Your First Try
* 70-762 Practice Test Questions in Multiple Choice Formats and Updates for 1 Year

100% Actual & Verified — Instant Download, Please Click
Order The 70-762 Practice Test Here