Prüfungsnummer:70-762-deutsch

Prüfungsname: Developing SQL Databases

Version:demo

http://www.it-pruefungen.ch/

Achtung: Aktuelle englische Version zu 70-762-deutsch bei uns ist gratis!!

1. Hinweis: Diese Aufgabe gehört zu einer Reihe von Fragestellungen, für die dieselben Antwortmöglichkeiten zur Auswahl stehen. Eine Antwort kann für mehr als eine Frage der Serie richtig sein. Die Fragen sind voneinander unabhängig. Die bereitgestellten Informationen und Details beziehen sich jeweils nur auf die Aufgabe, die diese Informationen enthält.
Sie haben eine SQL Server 2016 Datenbank mit dem Namen DB1. Die Datenbank enthält keine speicheroptimierten Dateigruppen.
Sie verwenden die folgende Transact SQL-Anweisung für das Erstellen einer neuen Tabelle:

```
CREATE TABLE tblTransaction (
    TransactionID int NOT NULL PRIMARY KEY,
    TransactionDate date NOT NULL,
    AccountID int NOT NULL,
    ValueType char(3) NOT NULL,
    Amount decimal(20,2) NULL
);
```

Die Tabelle wird bislang in Verbindung mit OLTP-Arbeitslasten genutzt.
Die Analytiker des Unternehmens müssen operative Echtzeitanalysen durchführen, die einen Großteil der Zeilen der Tabelle scannen und Aggregate für mehrere Spalten bilden.
Sie müssen einen möglichst effizienten Index für die Unterstützung der analytischen Arbeitslasten erstellen. Sie dürfen die Leistung für die bestehenden OLTP-Arbeitslasten nicht beeinträchtigen.
Wie gehen Sie vor?

A.Erstellen Sie einen gruppierten Index für die Tabelle.
B.Erstellen Sie einen nicht gruppierten Index für die Tabelle.
C.Erstellen Sie einen nicht gruppierten, gefilterten Index für die Tabelle.
D.Erstellen Sie einen gruppierten Columnstore Index für die Tabelle.
E.Erstellen Sie einen nicht gruppierten Columnstore Index für die Tabelle.
F.Erstellen Sie einen Hashindex für die Tabelle.

Korrekte Antwort: E

Erläuterungen:
Ein nicht gruppierter Columnstore-Index und ein gruppierter Columnstore-Index sind funktional gleich. Der Unterschied besteht darin, dass ein nicht gruppierter Index ein sekundärer Index ist, der für eine Rowstore-Tabelle erstellt wird, während einer gruppierter Columnstore-Index den primären Speicher für die gesamte Tabelle darstellt.
Der nicht gruppierte Index enthält eine Kopie eines Teils oder aller Zeilen und Spalten der

zugrundeliegenden Tabelle. Der Index ist als eine oder mehrere Spalte(n) der Tabelle definiert und weist eine optionale Bedingung auf, die zum Filtern der Zeilen dient.
Ein nicht gruppierter Columnstore-Index ermöglicht operative Echtzeitanalyse, bei der die OLTP-Arbeitsauslastung den zugrundeliegenden gruppierten Index verwendet, während die Analyse parallel auf dem Columnstore-Index ausgeführt wird.

2. Hinweis: Diese Aufgabe gehört zu einer Reihe von Fragestellungen, für die dieselben Antwortmöglichkeiten zur Auswahl stehen. Eine Antwort kann für mehr als eine Frage der Serie richtig sein. Die Fragen sind voneinander unabhängig. Die bereitgestellten Informationen und Details beziehen sich jeweils nur auf die Aufgabe, die diese Informationen enthält.
Sie haben eine SQL Server 2016 Datenbank mit dem Namen DB1. Die Datenbank enthält keine speicheroptimierten Dateigruppen.
Die Datenbank enthält eine Tabelle und eine gespeicherte Prozedur. Die beiden Objekte wurden mit dem folgenden Transact SQL-Skript erstellt:

```
CREATE TABLE Employee
(
    EmployeeID int NOT NULL PRIMARY KEY,
    FirstName varchar(20),
    LastName varchar(20),
    Status char(1),
    Address varchar(100),
    Department int NOT NULL
);

CREATE PROCEDURE uspSelectEmployeeDetails
(
    @LastName varchar(20)
)
AS
BEGIN
SELECT e.FirstName, e.LastName, d.DepartmentName
FROM Employee e
JOIN Department d ON e.DepartmentID = d.DepartmentID
WHERE e.Status = 'T' AND e.LastName = @LastName;
END;
```

Die Tabelle Employee ist datenträgerbasiert.
Sie fügen der Tabelle Employee 2000 Datensätze hinzu.
Sie müssen einen Index erstellen, der folgenden Anforderungen entspricht:

- Die Leistung der gespeicherten Prozedur muss optimiert werden.
- Alle relevanten Spalten der Tabelle Employee müssen eingeschlossen sein.
- FirstName und LastName sollen eingeschlossene Spalten sein.
- Die Speichergröße und die Schlüsselgröße des Index sollen möglichst gering sein.

Wie gehen Sie vor?

Erläuterungen:
Sie können der Blattebene eines nicht gruppierten Indexes Nichtschlüsselspalten hinzufügen, um vorhandene Beschränkungen des Indexschlüssels zu umgehen und vollständig abgedeckte, indizierte Abfragen auszuführen
Indem Sie Nichtschlüsselspalten einschließen, erstellen Sie nicht gruppierte Indizes, die eine größere Anzahl von Abfragen abdecken. Ein nicht gruppierter Index, der zusätzliche Spalten enthält wird daher auch als abdeckender Index bezeichnet. Dies ist der Fall, weil Nichtschlüsselspalten die folgenden Vorteile aufweisen:

Es kann sich um Datentypen handeln, die als Indexschlüsselspalten nicht zulässig sind.

Sie werden von Datenbankmodul beim Berechnen der Indexschlüsselspalten oder Indexschlüsselgröße nicht berücksichtigt.

Ein Index mit Nichtschlüsselspalten kann die Abfrageleistung erheblich steigern, wenn alle Spalten in der Abfrage in den Index als Schlüssel- oder Nichtschlüsselspalten eingeschlossen werden. Leistungsvorteile werden erzielt, weil der Abfrageoptimierer alle Spaltenwerte im Index finden kann; auf Daten der Tabelle oder des gruppierten Indexes wird nicht zugegriffen, sodass als Ergebnis weniger Datenträger-E/A-Vorgänge auftreten.
Beipiel:

```
USE AdventureWorks2012;
GO
CREATE NONCLUSTERED INDEX IX_Address_PostalCode
ON Person.Address (PostalCode)
INCLUDE (AddressLine1, AddressLine2, City, StateProvinceID);
GO
```

A.Erstellen Sie einen gruppierten Index für die Tabelle.
B.Erstellen Sie einen nicht gruppierten Index für die Tabelle.
C.Erstellen Sie einen nicht gruppierten, gefilterten Index für die Tabelle.
D.Erstellen Sie einen gruppierten Columnstore Index für die Tabelle.
E.Erstellen Sie einen nicht gruppierten Columnstore Index für die Tabelle.
F.Erstellen Sie einen Hashindex für die Tabelle.

Korrekte Antwort: B

3. Sie sind als Datenbankentwickler für das Unternehmen it-pruefungen tätig. Sie haben eine SQL Server 2016 Datenbank mit dem Namen DB1.
Sie stellen fest, dass es bei der Ausführung von Abfragen häufiger zu Deadlocks kommt.
Sie müssen sicherstellen, dass alle Deadlocks in einem XML-Format protokolliert werden.
Wie gehen Sie vor?

A.Erstellen Sie ein Microsoft SQL Server Integrationsdienste-Paket, das die dynamische Verwaltungsicht sys.dm_tran_locks verwendet.
B.Verwenden Sie Database Consistency Checker (DBCC) und aktivieren Sie das Ablaufverfolgungsflag 1224.
C.Konfigurieren Sie die Microsoft SQL Server Startoptionen und aktivieren Sie das Ablaufverfolgungsflag 1222.
D.Verwenden Sie die Microsoft SQL Server Profiler-Ereignisklasse Lock:Deadlock.

Korrekte Antwort: C

Erläuterungen:
Ein Deadlock tritt auf, wenn sich zwei Tasks dauerhaft gegenseitig blockieren, weil jeder der Tasks eine Sperre für eine Ressource aufrecht erhält, die die anderen Tasks zu sperren versuchen.
SQL Server Database Engine (Datenbankmodul) erkennt Deadlockzyklen in SQL Server automatisch. Database Engine (Datenbankmodul) wählt eine der Sitzungen als Deadlockopfer aus, und die aktuelle Transaktion wird mit einem Fehler beendet, um den Deadlock zu durchbrechen.
Wenn Deadlocks auftreten, geben die Ablaufverfolgungsflags 1204 und 1222 Informationen zurück, die im SQL Server 2005-Fehlerprotokoll erfasst werden. Ablaufverfolgungsflag 1204 meldet von jedem im Deadlock beteiligten Knoten formatierte Deadlockinformationen. Ablaufverfolgungsflag 1222 formatiert Deadlockinformationen, zunächst prozessweise, anschließend Ressource für Ressource. Es ist möglich, beide Ablaufverfolgungsflags zu aktivieren, um zwei Darstellungen desselben Deadlockereignisses zu erhalten.
Das Ablaufverfolgungsflags 1222 gibt Informationen in einem XML-ähnlichen Format zurück.
Der folgende Technet-Artikel enthält weitere Informationen zum Thema:
Erkennen und Beenden von Deadlocks

4. Hinweis: Diese Aufgabe gehört zu einer Reihe von Fragestellungen, die dasselbe Szenario verwenden. Jede Aufgabe dieser Reihe bietet einen anderen Lösungsweg. Sie müssen entscheiden, ob die Lösung geeignet ist, das Ziel zu erreichen.
Sie haben eine SQL Server 2016 Datenbank mit dem Namen DB1. Die Datenbank enthält eine Tabelle mit dem Namen Account. Die Tabelle wurde mithilfe der folgenden Transact-SQL Anweisung erstellt:

```
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)
```

);
Die Tabelle Account enthält 1 Milliarde Datensätze. Die Werte der Spalte AccountNumber identifizieren jedes Konto eindeutig. Die Spalte ProductCode enthält 100 verschiedene Werte. Die Werte sind gleichmäßig in der Tabelle verteilt. Die Tabellenstatistiken sind auf dem aktuellen Stand.
Sie führen häufig die folgenden SELECT-Anweisungen aus:

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

Sie müssen verhindern, dass für die Abfragen Tabellenscans ausgeführt werden. Sie müssen einen oder mehrere Indizes für die Tabelle erstellen.
Lösung: Sie führen die folgende Transact-SQL Anweisung aus:

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

Erfüllt das Vorgehen Ihr Ziel?

A.Ja
B.Nein

Korrekte Antwort: B

Erläuterungen:
Nach Erstellung der Indizes wird für die erste Abfrage weiterhin ein Tabellenscan ausgeführt. Für die zweite Abfrage wird ein Index Seek ausgeführt.
Um zu verhindern, dass Tabellenscans ausgeführt werden, muss ein gruppierter Index erstellt werden. In der Aufgabenstellung heißt es, dass die Werte der Spalte AccountNumber eindeutig sind. Der Primärschlüssel bzw. der gruppierte Index sollte daher für diese Spalte erstellt werden.
Der folgende Technet-Artikel enthält weitere Informationen zum Thema:
Beschreibung von gruppierten und nicht gruppierten Indizes

5. Hinweis: Diese Aufgabe gehört zu einer Reihe von Fragestellungen, die dasselbe Szenario verwenden. Jede Aufgabe dieser Reihe bietet einen anderen Lösungsweg. Sie müssen entscheiden, ob die Lösung geeignet ist, das Ziel zu erreichen.
Sie haben eine SQL Server 2016 Datenbank mit dem Namen DB1. Die Datenbank enthält eine Tabelle mit dem Namen Account. Die Tabelle wurde mithilfe der folgenden

Transact-SQL Anweisung erstellt:

```
    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)
);
```

Die Tabelle Account enthält 1 Milliarde Datensätze. Die Werte der Spalte AccountNumber identifizieren jedes Konto eindeutig. Die Spalte ProductCode enthält 100 verschiedene Werte. Die Werte sind gleichmäßig in der Tabelle verteilt. Die Tabellenstatistiken sind auf dem aktuellen Stand.
Sie führen häufig die folgenden SELECT-Anweisungen aus:

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

SELECT AccountNumber,
    Balance
FROM Account
WHERE ProductCode = 'CD'
```

Sie müssen verhindern, dass für die Abfragen Tabellenscans ausgeführt werden. Sie müssen einen oder mehrere Indizes für die Tabelle erstellen.
Lösung: Sie führen die folgende Transact-SQL Anweisung aus:

```
CREATE CLUSTERED INDEX PK_Account ON Account(ProductCode);
```

Erfüllt das Vorgehen Ihr Ziel?

A.Ja
B.Nein

Korrekte Antwort: B

Erläuterungen:
Die Werte der Spalte ProductCode sind nicht eindeutig. Für diese Spalte kann kein gruppierter Index erstellt werden.
Um zu verhindern, dass Tabellenscans ausgeführt werden, muss ein gruppierter Index erstellt werden. In der Aufgabenstellung heißt es, dass die Werte der Spalte AccountNumber eindeutig sind. Der Primärschlüssel bzw. der gruppierte Index sollte daher für diese Spalte erstellt werden.
Der folgende Technet-Artikel enthält weitere Informationen zum Thema:
Beschreibung von gruppierten und nicht gruppierten Indizes

6. Hinweis: Diese Aufgabe gehört zu einer Reihe von Fragestellungen, die dasselbe Szenario verwenden. Jede Aufgabe dieser Reihe bietet einen anderen Lösungsweg. Sie müssen entscheiden, ob die Lösung geeignet ist, das Ziel zu erreichen.
Sie haben eine SQL Server 2016 Datenbank mit dem Namen DB1. Die Datenbank enthält eine Tabelle mit dem Namen Account. Die Tabelle wurde mithilfe der folgenden Transact-SQL Anweisung erstellt:

```
    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)
);
```

Die Tabelle Account enthält 1 Milliarde Datensätze. Die Werte der Spalte AccountNumber identifizieren jedes Konto eindeutig. Die Spalte ProductCode enthält 100 verschiedene Werte. Die Werte sind gleichmäßig in der Tabelle verteilt. Die Tabellenstatistiken sind auf dem aktuellen Stand.
Sie führen häufig die folgenden SELECT-Anweisungen aus:

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

SELECT AccountNumber,
    Balance
FROM Account
WHERE ProductCode = 'CD'
```

Sie müssen verhindern, dass für die Abfragen Tabellenscans ausgeführt werden. Sie müssen einen oder mehrere Indizes für die Tabelle erstellen.
Lösung: Sie führen die folgende Transact-SQL Anweisung aus:

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

Erfüllt das Vorgehen Ihr Ziel?

A.Ja
B.Nein

Korrekte Antwort: A

Erläuterungen:
Um zu verhindern, dass Tabellenscans ausgeführt werden, muss ein gruppierter Index erstellt werden. In der Aufgabenstellung heißt es, dass die Werte der Spalte AccountNumber eindeutig sind. Der Primärschlüssel bzw. der gruppierte Index sollte daher für diese Spalte erstellt werden.
Der folgende Technet-Artikel enthält weitere Informationen zum Thema:
Beschreibung von gruppierten und nicht gruppierten Indizes

7. Hinweis: Diese Aufgabe gehört zu einer Reihe von Fragestellungen, die dasselbe Szenario verwenden. Jede Aufgabe dieser Reihe bietet einen anderen Lösungsweg. Sie müssen entscheiden, ob die Lösung geeignet ist, das Ziel zu erreichen.
Sie haben eine SQL Server 2016 Datenbank mit dem Namen DB1. Die Datenbank enthält eine Tabelle mit dem Namen Account. Die Tabelle wurde mithilfe der folgenden Transact-SQL Anweisung erstellt:

```
    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)
);
```

Die Tabelle Account enthält 1 Milliarde Datensätze. Die Werte der Spalte AccountNumber identifizieren jedes Konto eindeutig. Die Spalte ProductCode enthält 100 verschiedene Werte. Die Werte sind gleichmäßig in der Tabelle verteilt. Die Tabellenstatistiken sind auf dem aktuellen Stand.
Sie führen häufig die folgenden SELECT-Anweisungen aus:

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

SELECT AccountNumber,
    Balance
FROM Account
WHERE ProductCode = 'CD'
```

Sie müssen verhindern, dass für die Abfragen Tabellenscans ausgeführt werden. Sie müssen einen oder mehrere Indizes für die Tabelle erstellen.
Lösung: Sie führen die folgende Transact-SQL Anweisung aus:

```
CREATE NONCLUSTERED INDEX IX_Account_ProductCode On Account(ProductCode);
```

Erfüllt das Vorgehen Ihr Ziel?

A.Ja
B.Nein

Korrekte Antwort: B

Erläuterungen:
Um zu verhindern, dass Tabellenscans ausgeführt werden, muss ein gruppierter Index erstellt werden. In der Aufgabenstellung heißt es, dass die Werte der Spalte AccountNumber eindeutig sind. Der Primärschlüssel bzw. der gruppierte Index sollte daher für diese Spalte erstellt werden.
Der folgende Technet-Artikel enthält weitere Informationen zum Thema:
Beschreibung von gruppierten und nicht gruppierten Indizes