Examcollection

70-029

Microsoft

Designing and Implementing Databases with Microsoft
SQL Server 7.0

http://www.ipass4sure.com/exams.asp?examcode=70-029

Thousands of IT Professionals before you have already passed their 70-029 certification exams using the Microsoft 70-029 Practice Exam from ipass4sure.com. Once you start using our 70-029 exam questions you simply can't stop! You are guaranteed to pass your Microsoft 70-029 test with ease and in your first attempt.

Here's what you can expect from the ipass4sure Microsoft 70-029 course:

* Up-to-Date Microsoft 70-029 questions designed to familiarize you with the real exam.
* 100% correct Microsoft 70-029 answers you simply can't find in other 70-029 courses.
* All of our tests are easy to download. Your file will be saved as a 70-029 PDF.
* Microsoft 70-029 brain dump free content featuring the real 70-029 test questions.

Microsoft 70-029 Certification Exam is of core importance both in your Professional life and Microsoft Certification Path. With Microsoft Certification you can get a good job easily in the market and get on your path for success. Professionals who passed Microsoft 70-029 Certification Exam are an absolute favorite in the industry. If you pass Microsoft 70-029 Certification Exam then career opportunities are open for you.

Our 70-029 Questions & Answers provide you an easy solution to your Microsoft 70-029 Exam Preparation. Our 70-029 Q&As contains the most updated Microsoft 70-029 real tests. You can use our 70-029 Q&As on any PC with most versions of Acrobat Reader and prepare the exam easily.

QUESTION 1

You are implementing a logical data model for an Online Transaction Processing (OLTP) application. One entity from the logical model is in third normal form and currently has 10 attributes. One attribute is the primary key. Six of the attributes are foreign key referencing into six other entities. The last three attributes represent columns that hold Numeric values.
How should this entity from the logical model be implemented?

A. Create one table by denormalizing the entity. Add the information from the six foreign key references as additional column of the table.
B. Create two tables by denormalizing the entity. Add the three primary keys and the three numeric values as column of the one table. Add the primary key and the information from the six foreign key references as columns of the other table.
C. Create the table as described in the logical data model.
D. Create a view that joins the six foreign keys references.

Answer: C.

Explanation: Usually the logical database design is normalized. Often the normalized design is the most efficient, or the most part of the normalized design. Sometimes, for performance reasons, the implementer chooses to denormalize parts of the implementation. One common way to denormalize is when there are queries that run frequently which have four or more joins. Then tables could be denormalized, by duplicating columns or by combining tables, to get fewer joins. Another way to denormalize is to store computed values instead of calculating them every time they are needed.
In this scenario we know that this is an OLTP system. And it is important for an OLTP system to be able to do fast updates and searches. We also know that there is a table containing one primary key column, six foreign key columns, and three numeric columns. We don't have any information on other tables or how queries will use this table. Therefore we can't find any reason for denormalizing this table and we keep the physical implementation exactly as the logical design, since the normalized implementation usually gives the best performance.
Incorrect answers:
A: There is nothing to gain by adding the columns that the foreign key columns reference to the table. Performance would suffer because of the redundant data, which would make the table larger.
B: There is no reason for splitting the table into two in this scenario. It would make queries that use the table in joins to make an extra join. Performance would most likely suffer.
Splitting a table vertically like this could be a good idea if there are only a few columns that change or are of particular interest. Then the other columns could be moved to another table. The two tables would be linked by a one-to-one relationship.
D: The six foreign key references are most likely not used all at once.
Making a view would not be provide any performance advantages.

QUESTION 2

You are designing a database that will be used to store information about tasks assigned to various employees. Each task is assigned to only one employee. The database contains a table named Task that is modeled as shown in the exhibit. .

Exhibit:
Task
• TaskNo
• EmployeeNo
• Title
• Description
• Status
• DateCompleted
You want to use a PRIMARY KEY constraint to uniquely identify each row in the Task table. On which
column or columns should you define the PRIMARY KEY constraint? (Choose all that apply)

A. TaskNo
B. EmployeeNo
C. Status

Answer: A.

Explanation: The TaskNo column must be a part of the primary key. The question is whether the primary key
should be composite or not. It is clear that Status should not be a part of the key, but it might be tempting to add
the EmployeeNo column to the primary key. If we create a composite primary key with the columns TaskNo
and EmployeeNo only the combination of TaskNo and EmployeeNo would have to be unique. The same
TaskNo could be used on several rows. In particular we could have the following two rows:

TaskNo	EmployeeNo	Title	Description	Status	DateCompleted
1	100	RW1	Repair car	Started	2001-04-03
1	101	RW1	Repair-car	Started	2001-04-03

And we would have two employees assigned to the same project.
We should not use a composite key.
Incorrect answers:
B: These would allow several employees on the same task.
C: If status was part of the Primary key the table could contain duplicate tasks.

QUESTION 3
You have a table with a clustered primary key. New records are added by a batch job that runs at night.
The table grows by 20 percent per year.
During the day, the table is used frequently for queries. Queries return ranges of rows based on the
primary key. Response times for queries have become worse over time.
You run the DBCC SHOWCONTIG statement. The statement provides the following output:
Pages Scanned 354
Extents Scanned 49
Extent Switches 253
Avg. pages per extent 7.2
Scan Density [Best Count: Actual Count] ...... 17.79% [45:94]
Extent Scan Fragmentation 82.21%
Avg. Bytes Free per Page 485.2
Avg. Page Density (full) 94.01%

How can you improve the query performance?

A. Update statistics on the clustered index
B. Change the row size to fit efficiently on a page
C. Rebuild the clustered index with a fillfactor value set to 100
D. Rebuild the clustered index with a fillfactor value set to 25
E. Rebuild the clustered index with a fillfactor value set to 75

Answer: E

Explanation: In general, you need to do very little in terms of index maintenance. However, indexes can become fragmented. You can use the DBCC SHOWCONTIG command to report on the fragmentation of an index. You can use the Extent Scan Fragmentation value to determine the general level of fragmentation in a table. This value should be as close to zero as possible (although a value from 0% through 10% may be acceptable).
In this scenario we see that extent fragmentation is well above 10%, in fact it is 82%. To resolve the fragmentation problem we must rebuild the index. And we must decide which fillfactor to use.
The fillfactor specifies a percentage that indicates how full SQL Server should make the leaf level of each index page during index creation. For example:
• a fillfactor of 25% fills 25% of is page and leaves 75% for growth.
• a fillfactor of 75% fills 75% of is page and leaves 25% for growth.
When an index page fills up, SQL Server must take time to split the index page to make room for new rows, which is quite expensive. For update-intensive tables, a properly chosen FILLFACTOR value yields better update performance than an improper FILLFACTOR value.
The table increases 20% every year, updates are daily and queries it used frequently. By choosing a fillfactor of 75% when the index is rebuilt, we would allow a 25% growth before the data pages fill up. And very few page splits would occur for a year.
Most likely the index would have to rebuild in the same way after one more year.
Incorrect answers:
A: The index is fragmented and has to be rebuilt.
Statistics is used by SQL Server to optimize queries at runtime. Updating the statistics might improve performance, but the fragmentation of the index is the main problem.
B: There should be no need to adjust the size of a row to fit a data page. Rebuilding the index with an appropriate fillfactor would improve performance.
C: A fillfactor of 100% would set the data pages to fill completely (no free space). This would result in many page splits and decreased performance, since the database grows daily.
D: A fillfactor of 25% would leave 75% of the data pages empty which is a bit wasteful, since the database only grows by 20% yearly.
The query performance would be better with a fillfactor of 75% since less data pages would have to be searched.

QUESTION 4
You have a table with a clustered primary key. The table is used frequently for both queries and data modification. As part of a review of data storage and disk utilization, you run the DBCC SHOWCONTIG statement. The statement provides the following output:

Pages Scanned 158
Extents Scanned 21
Extent Switches 20
Avg. pages per extent 7.5
Scan Density [Best Count: Actual count]. . . . 95.24% [20:21]
Extent Scan Fragmentation 4.76%
Avg. Bytes Free per Page 408.4
Avg. Page Density (full) 94.95%
What does this output tell you about how the data is stored? (Choose all that apply)

A. The table is not externally fragmented.
B. The table is not internally fragmented.
C. The number of Extent Switches is excessive.
D. The row size does not efficiently fit on a page.
E. The IAM page does not reflect the actual extent usage.

Answer: A, B

Explanation: The DBCC SHOWCONTIG command is used to display fragmentation information for the data and indexes of the specified table. A database administrator must be able to use the output to decide the level of fragmentation of the table.
External fragmentation:
External fragmentation occurs when the logical order of pages does not match the physical order, or when the extents belonging to a table are not contiguous.
Scan Density [Best Count: Actual count] should be close to 100% to have acceptable external fragmentation. In this scenario it is 95%, which is ok.
Internal fragmentation:
Internal fragmentation means that the index is taking up more space than it needs to. Internal fragmentation should be below 10%. In this scenario it is 4.76%, which is acceptable.
Extent switches:
One extent consists of eight data pages. Extent switches are the number of times the DBCC statement moved from one extent to another while it traversed the pages of the table or index. Here we have 21 extents and 20 extent switches. We have an optimal number of extent switches.
Row size fitting page
Avg. Bytes Free per Page shows how many bytes are free per page. Only 408.4 bytes are free so the row size fits the page quite well.
IAM page reflecting actual extent usage
Index Allocation Map (IAM) pages map the extents in a database file used by a heap or index. IAM pages list the extents in disk order.
The table is not fragmented, the extent switches are optimal and the rows size fits the data pages, so the IAM page would reflect actual extent usage very well.
Incorrect answers:
C: The number of Extent Switches is minimal, not excessive.
D: The row size does efficiently fit on a page.
E: The IAM page does reflect the actual extent usage.

QUESTION 5

You are the database administrator for Certkiller . You receive reports that your Sales application has very poor response time.
The database includes a table that is defined as follows:
CREATE TABLE dbo.Orders (

OrderID	Int IDENTITY(1,1)	NOT NULL,
SalesPersonID	Int	NOT NULL,
RegionID	Int	NOT NULL,
OrderDate	Datetime	NOT NULL,
OrderAmount	Int	NOT NULL,
CustomerID	Int	NULL

The OrderID column is the primary key of the table. There are also indexes on the RegionID and OrderAmount columns.
You decide to run a showplan on all queries in the application. The following query, which access this table, is used to list total average sales by region:
SELECT t1.RegionID, AVG(t1.SalesTotal) AS RegionAverage
FROM (SELECT RegionID, SalesPersonID, SUM(OrderAmount) AS SalesTotal
SalesTotal
FROM Orders
GROUP BY RegionID, SalesPersonID) AS t1
GROUP BY t1.RegionID
You set the SHOWPLAN_TEXT option to ON and execute the query. The showplan output is as follows:
I--Compute Scalar(DEFINE:([Expr1003]=If[Expr1006]=0) then
NULL else ([Expr1007]/[Expr1006])))
I--Stream Aggregate(GROUP BY:([Orders].[RegionID])
DEFINE:([Expr1006]=COUNT([Expr1002]),
[Expr1007]=SUM([Expr1002])))
I--Compute Scalar(DEFINE:([Expr1002]=If
[Expr1004]=0) then NULL else ([Exprl005])))
I--Stream Aggregate(GROUP
BY:([Orders].[RegionID],[Orders].[SalesPersonID])
DEFINE:([Expr1004]=COUNT(*), [Exprl005]=SUM([Orders].[OrderAmount])))
I--Sort(ORDER BY:([Orders].[RegionID]
ASC, [Orders].[SalesPersonID] ASC))
I--Table
Scan(OBJECT:([ServerA].[dbo].[Orders])
You suspect that this query is part of the problem because the showplan indicates that the query is performing a table scan operation. What is the most likely reason that this query is performing a table scan?

A. There is no composite index on OrderID, RegionID, and OrderAmount.
B. There is no WHERE clause in the query.
C. The query contains a subquery.
D. The query is performing aggregate operations.