

SQL Server Interview Questions

By

Rafat Sarosh, Mechele Gruhn

SQL Interview Questions © by Rafat Sarosh.

This document, including sample applications herein, is provided for informational purposes only and neither Azan International nor the Authors makes any warranties, either express or implied, in this document. Information in this document, including samples, URL and other Internet Web site references, is subject to change without notice. The entire risk of the use or the results of the use of this document remains with the user.

The primary purpose of a sample is to illustrate a concept, or a reasonable use of a particular statement or clause. Most samples do not include all of the code that would normally be found in a full production system, as a lot of the usual data validation and error handling is removed to focus the sample on a particular concept or statement. Technical support is not available for these samples or for the provided source code.

Unless otherwise noted, the example companies, organizations, products, people, and events depicted herein are fictitious and no association with any real company, organization, product, person, or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Azan International.

©2002-2004 Rafat Sarosh. All rights reserved.

Active Directory, ActiveX, BackOffice, CodeView, Developer Studio, FoxPro, JScript, Microsoft, Microsoft Press, Microsoft SQL Server, MSDN, MS-DOS, Outlook, PivotChart, PivotTable, PowerPoint, Visual Basic, Visual C++, Visual Studio, Win32, Windows 2000, Windows, and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Contents

A Note from the Author	12
SQL SERVER.....	14
General Relational DBMS knowledge	14
Q. What is a candidate key?.....	14
Q. What are alternate keys?.....	15
Q. What is a primary key?.....	16
Q. What is the difference between a primary key and a unique key? Are they the same?.....	17
Q. What is referential integrity?.....	18
Q. What is an Entity-Relationship Diagram (ERD)?.....	19
Q. What is a subquery?.....	21
Q. What is a correlated subquery?.....	21
Q. How do you use a subquery to find records that exist in one table and do not exist in another? .	22
Q. What does it mean to normalize a database and why would you do it?.....	23
Q. What is denormalization?.....	27
Design and Programming	28
Q. How can I detect whether a given connection is blocked?.....	28
Q. How would you design a database for an online site, which would average a million hits a day?28	
Q. You are testing the performance of a query. The first time you run the query, the performance is slow. The second time you run the query, the performance is fast. Why is this?.....	29
Q. What you can do to remove data from the cache and query plans from memory for testing the performance of a query repeatedly?.....	29
Q. What is wrong in the following SQL statement?.....	29
Q. Write the fastest query to find out how many rows exist in a table.	30
Q. The COUNT() function always returns a <i>int</i> value type. What should you do if you need to count rows from a query which you know will return a value that is too large for an <i>int</i> value type?	31
Q. What is a derived table?.....	31
Q. You have a table with three columns: Amount1, Amount2, and Amount3. In a single row only one of these three columns will have a value and the other two will be null. Write a SQL query to retrieve the values from these columns.	31
Q. How can you fix a poorly performing query?.....	32
Q. What is the physical representation for a many-to-many relationship?.....	33
Q. What is the maximum length of an extended property?.....	33
Q. In which database can extended stored procedures be added?.....	33
Q. Why does this query return 0?.....	34
Q. If a column is an image value type, how you can compare column values? How you can use this column in join clause?	34
Q. Which data type cannot be used as a parameter in a UDF?.....	34
Q. Which four data types cannot be used as a return type from a user-defined function?.....	34
Q. Can an extended stored procedure be called from inside a user-defined function?.....	34
Q. How you can make a parameterized view?.....	35
Q. How long can an Error message be in RAISEERROR function?.....	35
Q. What are the constraints on severity level in RAISEERROR?.....	35
Q. How can you log an error to the server event log from a stored procedure?.....	36
Q. Explain the Rollup operator.	36
Q. Explain the Cube operator.....	37
Q. What is an application role and explain a scenario when you would use one?.....	39
Q. On Friday, you issued several INSERT statements using Query Analyzer. You then verified the data had been correctly entered with a SELECT statement. On Monday, your users report that the data is not there. What happened?.....	40
Q. You have two tables with a one to many join based on a field named ID. You need to find records in the parent table that are not represented in the child table. How would you accomplish this? ...	40

Q. Give an example of why you would want to denormalize a database.....	40
Q. Explain extended properties.....	41
Q. You have couple of stored procedures that depend on a table. You dropped the table and recreated it. What do you have to do to reestablish those stored procedure dependencies?.....	41
Q. How can you append an identity column to a temporary table?.....	41
Q. You schedule a job to run every minute. What will happen if the first job runs more than 1 min? Will the second instance of the job start?.....	41
Q. Why should you use or avoid SELECT * statements?.....	42
Q. What is wrong with this statement?.....	42
Q. You have several tables, and they are joined together for querying. The tables contain both clustered indexes and non clustered indexes. To optimize performance, how should you distribute the tables and their indexes onto different file groups?.....	42
Q. Which event (Check constraints, Foreign Key, Rule, Trigger, Primary key check) will be performed last for an integrity check?.....	42
Q. What is the fastest way to permanently delete a 1 million row table named customers?.....	42
Q. After removing a table from database, what other related objects have to be dropped explicitly?.....	44
Q. You want to check the syntax of a complicated Update SQL statement without executing it. What command should you use?.....	44
Q. You are the database developer for a leasing company. Your leasing database includes a lease table that has a column which keeps Social security numbers. Each SSN must be unique. You want the data to be physically stored in order by SSN. What constraint should you add to the SSN column on the lease table?.....	45
Q. You are designing a database for your company. Your Company has 10 departments and each department has between 25 to 100 employees. Each employee may work for one or more departments. How can you represent this relationship in your ERD (entity relationship diagram)?.....	45
Q. Mary's company is a research company that does experiments. Mary's company database includes an Experiment table. The Experiments table stores all the Experiments and their ID. There is also a Project table, which keeps all the Projects with their ID. How should the database be designed, so that an experiment cannot be repeated in a Project, but a single experiment can belong to more than one Project?.....	46
Q. John exports information periodically from a Microsoft SQL Server database to an Oracle Database. What is the best way to do this?.....	46
Q. You are designing a database for your human resources department. In the employee table, there is a field for Social Security Number, which cannot contain NULL values. If no value is given, you want a value of UNKNOWN to be inserted in this field. What is the best approach?.....	46
Q. Is it true that rules do not apply to data already existing in a database at the time the rule is created?.....	46
Q. Is it true, that there is no difference between a rule and a check constraint?.....	46
Q. Can a rule be bound to any column of any data type?.....	46
Q. You are responsible for designing the physical architecture of an OLTP system. You have two tables TableA and TableB that will be joined together by several queries. You need good reliability and high performance for these queries. What should you do?.....	47
Q. What is RAID, and how it can influence database performance?.....	47
Q. You work at the corporate office of a Fortune 500 company that has regional offices in 100 countries. Each regional office maintains its own customer table. This information needs to be brought to the corporate office and merged, to do some analysis for the marketing department. The marketing department also needs to uniquely identify each customer. How should you design the customer table?.....	48
Q. Explain the DBCC PINTABLE command. When would you use it?.....	49
Q. You have to design a database for a manufacturing plant. Among other things, this database maintains the attendance of the workshop employees who work in 3 shifts. The Attendance table should have a field named 'Shift', which stores the shift that the employee worked. The Workshop supervisor should be able to input only 1, 2 or 3 in this field. What is the best design, to implement this requirement?.....	49

Q. What is a transaction and why is it important?	50
Q. What is deadlock?	50
Q. What is a livelock?	50
Q. How you can minimize the deadlock situation?	51
Q. What is the importance of concurrency control?	51
Q. Define Joins.	52
Q. What is an outer join?	53
Q. Define a cross join.....	54
Q. How you can change a cross join into an inner join?.....	54
Q. Define constraints and give an example of their use?.....	55
Q. Write a SQL Query to sort on different column name according to the parameters passed in the function.	56
Q. What is the downside of using UDF?.....	57
Q. You have a stored procedure, which execute a lengthy batch job. This stored procedure is called from a trigger. You do not want to slow the data entry process. You do not want trigger to wait for this batch job to finish before it completes itself. What you can do to speed up the process?	58
Q. Write a SQL Query to delete duplicate records from a table called TABLE1	59
Q. What is an index?	60
Q. What is the preferred way to create a clustered and non-clustered index? Which index should you create first the clustered or non-clustered?.....	60
Q. Can a unique index be created on a column, which contains NULL?	60
Q. How would you choose between a clustered and a non-clustered index?.....	61
Q. Your table has a large character field. There are queries that use this field in their search clause. What should you do?.....	63
Q. What is a fill factor?.....	63
Q. When you should use a low fill factor?	64
Q. What are statistics?	64
Q. What is clustering?	64
Q. What is the difference between DBCC INDEXDEFRAG and DBCC REINDEX?	64
Q. How you can find out if an index is useful to the optimizer?.....	65
Q. Where are full-text indexes stored?	65
Q. How many full-text indexes can a table have?	65
Q. Indexes are updated automatically. Is the full-text index also updated automatically?	65
Q. How is a full-text index updated?	65
Q. You have a table called 'Item', which has columns as follows:	66
Q. Can you force a query to use a specific Index?.....	66
Q. Which data type columns are the best candidates for full-text indexing?.....	66
Q. When would you prefer to have a minimum number of indexes?	66
Q. You created a table as follows	67
How you would rewrite the SQL Query to return the CUSTOMERID sorted numerically?	67
Q. Why is there a performance difference between two similar queries where one uses UNION and the other uses UNION ALL?.....	68
Q. You have a table 'test' which is a copy of northwind employee table. You have written a trigger to update the field 'HireDate' with the current date.....	69
Q. What is the difference between OPENROWSET and OPENQUERY?	69
Q. How you can add messages to the NT event log from within a stored procedure?.....	69
Q. What are three ways you can use an identity value inside a trigger? Why would you prefer one way over another?.....	70
View	72
Q. List some of the rules that apply to creating and using a 'view'.	72
Q. You added a row to a view, but the row is not shown on the view. Explain how this can happen, and how you can remedy the situation.....	73
Q. Can an ORDER BY clause be used in a creation of a view?.....	73
Q. 'Order by' is not allowed in a view. How can you sort information from a view?	74

Q. What is a derived Table?	75
Q. What are Information Schema Views?	76
Q. What is a partitioned view?	77
Q. What is an Indexed View?	78
SQL Server Administration	79
Q. How you can get a list of all the table constraints in a database?	79
Q. How you can get the list of largest tables in a database?	79
Q. How you can move data or databases between servers and databases in SQL Server?	79
Q. If no size is defined while creating the database, what size will the database have?	80
Q. Can a database be shrunk with users active?	80
Q. As a general practice, it is recommended to have dbo be the owner of all database objects. However, in your database you find number of tables owned by a user other than dbo, how could you fix this?	81
Q. How you can list all the indexes for a table in a database?	82
Q. Why does a SQL statement work correctly outside of a user-defined function, but incorrectly inside it?	82
Q. Can a table be moved to different Filegroup?	82
Q. Can a database be shrunk to 0 Bytes, if not, why?	82
Q. What does the Automatic recovery do?	83
Q. Can an automatic recovery be initiated by a user?	83
Q. What is the primary use of the model database?	83
Q. What information is maintained within the msdb database?	83
Q. What stored procedure can you use to display the current processes?	83
Q. What stored procedure would you use to view lock information?	83
Q. How can a database be repaired?	83
Q. How can you find out if the current user is a member of the specified Microsoft® Windows NT® group or Microsoft SQL Server™ role?	84
Q. Your SQL Server is running out of disk space. You notice that there are several large files with LDF extensions. What are these files?	84
Q. You notice that the transaction log on one of your databases is over 4GB. The size of the data file is 2MB. What could cause this situation, and how can you fix it?	84
Q. You accidentally delete the MSDB database. What effect does this have on your existing SQL databases, and how do you recover?	84
Q. Where can you add custom error messages to SQL Server?	84
Q. Is it important for a Database Administrator to understand the operating system and file access?	84
Q. What is the difference between writing data to mirrored drives versus RAID5 drives.	84
Q. In the physical file layout, where should the transaction log be stored in relation to the data file?	84
Q. You have separate development and production systems. You want to move a copy of a development database into production. To do this, you do a backup on the development system and restore to the production system. After a few minutes, you begin getting calls from several customers saying that they are denied access to the system. Why?	85
Q. What is a mixed extent?	85
Q. You have a table with close to 100 million records. Recently, a huge amount of this data was updated. Now, various queries against this table have slowed down considerably. What is the quickest option to remedy the situation?	85
Q. How can you check the level of fragmentation on a table?	85
Q. You have developed an application which uses many stored procedures and triggers to update various tables. Users occasionally get locking problems. Which tool is best suited to help you diagnose the problem?	85
Q. Which table keeps the locking information?	86
Q. You want to be sure that queries in a database always execute at the maximum possible speed. To achieve this goal you have created various indexes on tables. Which other statement will keep the database in good condition?	86

Q. During a recent migration project, John inserted 10,000 records in a table that has an Identity Column called ticketID, which automatically increases by 1 each time a record is inserted in the table. A month after the database went live; John noticed that record with ticketID 5123 has some incorrect information. So John deletes this record and decides to re-insert this record in the table. He wants to re-use the ticketID 5123. He needs to achieve this while the database is in production. What should he do?.....	86
Q. Jenny wants to export data to Pivot table in Excel spreadsheet from a table in SQL Server. This data changes frequently. She wants to automate the process of updating the Excel spreadsheet using the SQL Job Scheduler. What tool is the best choice for the task?.....	87
Q. You have written an application in VB which uses SQL 7 for its database. You have used many stored procedure and triggers to make your application fast. However, users complain that saving records take too much time. To rectify the problem, you start the profiler and create a trace using the trace wizard. How would you go about identifying the cause of the problem?	87
Q. You have a table with employee information that rarely changes. However this table is used from many applications in the organization to validate the data and to produce reports. What would be the optimal fill factor to choose for indexes created on this table?	87
Q. What is the difference between a fill factor of 100 and 0?.....	87
Q. How will you know when statistics on a table are obsolete?	89
Q. Explain different backup plans.	90
Q. What is a full backup?.....	90
Q. Explain a differential backup.	90
Q. Explain an incremental backup.	90
Q. What is Log shipping?	91
Q. Every night you run a full backup. After every 3 three hours you make a differential backup. Every hour you make an incremental backup. In a worst-case scenario, how much work you can lose?..	91
Q. Explain a Checkpoint?.....	91
Q. Explain an Automatic Checkpoint.....	91
Q. How you can list all the tables in a database?.....	92
Q. How can you list all the columns in a database?	92
Q. How can you list all the table constraints in a database?	92
Q. What are the advantages DTS has over bcp?	92
Q. How do you rebuild an identity column?.....	92
Q. Create a DTS package to produce a text file using the 'UPDATE STATISTICS' command for the tables in a database with obsolete statistics. For example, the file content should look like:.....	93
Q. How can you transfer data from a text file to a database table? Or how can you export data from a table to a comma delimited (CSV) file? Or how can you import data from MS Access to a table in a database? Or how can you export data from a table to an Excel file?	102
Q. When does the auto update index statistics feature in SQL server turn itself on?.....	102
Q. What specific conditions database should meet, before you can Bulk copy data into it using BCP.	102
Replication	103
Q. What is database replication? What are the different types of replication you can set up in SQL Server?	103
Q. Which database stores information about replication?	103
Q. Your company has 50 branches all over the country. All the branches, including the Head Office have SQL Server as the database. Every night all 50 branches upload certain information to the Head Office. Which Replication topology is best suited for the above scenario?.....	103
Q. Which of the following option(s) is(are) an inappropriate usage of merge replication: a company time sheet database, a static look up table, a high transaction based application that requires real time replication to subscribers or a company information price list that is updated at remote sites and replicated to a central database.....	103
Q. What are the restraints imposed on the table design by a Merge Replication?.....	104
Security	105

Q. A user is a member of the Public role and the Sales role. The Public role has select permission on all the tables. The Sales role does not have select permission on some of the tables. Will the user be able to select from all tables?	105
Q. If a user does not have permission to a table, but has permission to a view created on it, will he be able to view the data in table?	105
Q. Tony works in the Sales Department and has created a table named OrderDetail for the Sales department. You write a stored procedure which will help Mark, a sales representative, update the OrderDetail table. However, when Mark uses the stored procedure he gets an error. You realize that this is a security issue. What is required to enable Mark to use your stored procedure?.....	106
Transactions.....	107
Q. Define Concurrency Control.....	107
Q. What is Pessimistic Concurrency Control?	107
Q. What is Optimistic Concurrency Control?	107
Q. Explain Isolation levels.	107
Q. What is the difference between the REPEATABLE READ and SERIALIZABLE isolation levels?	108
Q. What is the default isolation level in SQL server?	108
Q. What is the most restrictive isolation level?.....	108
Q. What is the least restrictive isolation level?	108
Q. How do you determine the current isolation level?.....	108
Q. What are the conditions an underlying table must satisfy before a cursor can be used by a positioned UPDATE or DELETE statement?	108
Q. Explain Locks.	109
Q. Explain Lock escalation.	109
Q. Under what condition it is possible to have a page level lock and row lock at the same time for a query?.....	109
Q. Explain how to use transactions efficiently.	109
Triggers.....	110
Q. What you can do to delete a table without the delete trigger firing?	110
Q. Other than <i>Truncate</i> statement, which other command can by-pass the trigger on the tables?..	110
Q. What are the differences between Triggers and Stored Procedures?.....	110
Q. Is this statement true: A trigger can reference objects outside the current database?.....	110
Q. Can a trigger be created on a view?.....	110
Q. Will the WRITETEXT statement activate a trigger?.....	110
Q. Can a table be created inside a Trigger?	110
Q. When should you use an INSTEAD OF trigger?.....	111
Q. Can the "If Update (colName)" statement be used in a delete trigger?.....	112
Stored Procedures.....	113
Q. Why should one not prefix user stored procedures with 'sp_'?.....	113
Q. What are the results of running this script?	113
Q. Which table keeps information about Stored Procedures?.....	113
Q. What will be the value of @@FETCH_STATUS if a row that was a part of the cursor resultset has been deleted from the database after the time the stored procedure that opened the cursor was executed?.....	114
Q. Why and when do stored procedure recompile?.....	115
Q. How can you find out which stored procedures are recompiling?	115
Q. How can you stop stored procedures from recompiling?.....	117
Cursors.....	118
Q. How many types of cursor type are there?.....	118
Q. What is the difference between insensitive and scroll cursor?	118
Q. If a table does not have a unique index, can a cursor be opened on it?	118
Q. Can a cursor be updated? If yes, how you can protect which columns are updated?.....	118
Q. While using a cursor, how can you differentiate between a deleted row and a row that has been inserted with NULL data values?	119

Q. How can you know if the row fetched from cursor is still valid in underlying table?	119
Q. How can you find out how many rows returned in a cursor?	119
Q. What does it mean if @@CURSOR_ROW returns a negative number?.....	119
Q. How can you set the threshold at which SQL Server will generate keysets asynchronously?.....	119
Q. What is the difference between Deallocate cursor and Close cursor?.....	119
Q. Define Cursor Locking.....	120
Datatypes	121
Q. Between Cast and Convert which function would you prefer and why?.....	121
Q. What are the new data types are introduced in SQL 2000?	121
Q. Why it is recommended to avoid referencing a floating point column in the WHERE clause?..	121
Q. You have to store user responses of 'Yes' and 'No'. What kind of data type is best suited for this task?.....	121
Q. What is the Exact Numeric data type in SQL?.....	121
Q. Explain timestamp datatype?.....	122
Q. How can a user-defined datatype be created?.....	122
Q. What are the approximate numeric data types?.....	123
Q. You are creating an application where Users are asked their gender. In the gender combo box you have three options: 'Male' , 'Female' and 'I choose not to disclose'. These options are stored in the table as 1, 0 or NULL. Which datatype should you use?	123
Q. Which data types generate inaccurate results if used with an = or <> comparison in a WHERE clause of a SQL statement?	123
Q. Beginning with SQL Server Version 7.0, a new enhanced data type nchar was added. What type of data is supported with this data type?.....	123
Q. What will happen if a column containing char type data is changed to the nchar data type?	123
Q. To automatically record the time on which the data was modified in a table, which data type should you choose for the column?.....	124
XML in SQL.....	125
Q. What is XDR?.....	125
Q. What is the difference between FOR AUTO and FOR NESTED?.....	126
Q. What is the difference between FOR XML RAW and FOR XML AUTO?.....	128
Q.Explain FOR XML EXPLICIT Mode?	131
Q. Using the Customer, and Order table in Northwind database, Please write a query to produce following XML?	131
Q. What is XML Schema?	132
Q. What is mapping schema?.....	132
Q. You have a table 'customers' , which has three fields, Address, CustomerID and ContactName. You would like to retrieve rows as follows:	133
Q. What is XPath?.....	134
Q. What keyword you will use to get schema appended to the result set of a 'for XML' query?	135
Q. If I execute this query.....	136
How can I rewrite this query to get all attributes as an element?.....	136
Clustering.....	137
Q. Explain Active/Passive and Active/Active cluster configurations?.....	137
SQL Server TID BITS.....	138

A Note from the Author

This book is based on a collective effort and personal experience of many individuals. These Interview questions are collected after personally going through many interviews, reading books, reviewing magazine, internet articles and lurking in news groups.

This book is a collection of all those questions and answers in one place, for easy reference and revision. *By no means does it replace your online manual or a good SQL book.* Our goal is to provide you with frequently asked questions in interviews, and some explanations to get you started on your quest for deeper understanding.

Many of the examples in this book use the northwind and pubs databases that are installed with SQL server.

The latest version of this book can be downloaded from <http://www.manyquestions.com>. The online version is continuously updated and kept current. For a year, for no extra cost the new updated versions will be e-mailed to you.

I highly encourage you to please send me your feedback. Send your questions, your comments, your criticism, and your suggestions to rsarosh@hotmail.com.

Many thanks to the following contributors and reviewers: Pete Levenson, Bott Bruce Ted Nethery and Rawish Khan.

Disclaimers

SQL Server Books On-Line is referenced many times throughout this book. The authors believe that SQL Server Books On-Line is an excellent reference and there is no need to repeat what has already been wonderfully explained. This book should be considered a supplement to SQL Server Books On-Line. In the unlikely event that this book provides conflicting information with the SQL Server Books On-Line, SQL Server Books On-Line should be considered the ultimate reference.

What this book is:

This book is not intended to be an introduction to SQL Server or Transact SQL programming. There are several excellent books already created on those subjects. Rather, this book is meant to be a tool to help those who currently work with SQL Server, or those who have SQL Server education. Use this as a guide to prepare for some of the technical questions that may be asked in an interview situation, for test preparation, memory refreshing, and as a general reference.

Intended Audience:

The target audience for this book will have basic SQL Server knowledge and have at least a rudimentary understanding of the Transact SQL language. In addition, the reader should have a grasp of the fundamentals of computing.

Target database:

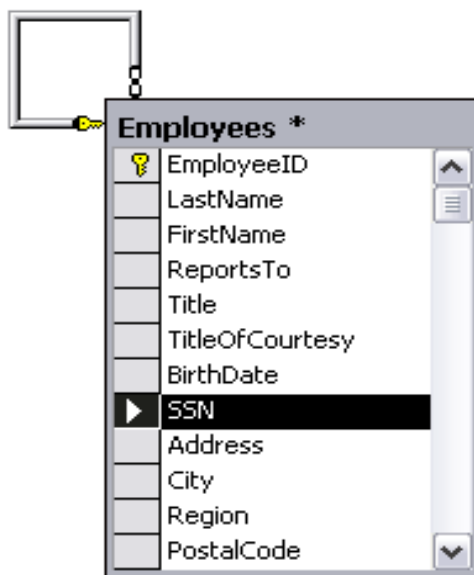
SQL 2000 was used to develop and test the examples in this book. Most of the code examples are written in ANSI-92 compliant T-SQL and will work equally well on SQL 6.5 or SQL 7.0 as they do in SQL 2000. However, please note that SQL 2000 was the target development environment.

SQL SERVER

General Relational DBMS knowledge

Q. What is a candidate key?

A. A candidate key is one that can uniquely identify each row of a table. Generally, a candidate key becomes the primary key of the table. If the table has more than one candidate key, one of them will become the primary key and the rest are called alternate keys.

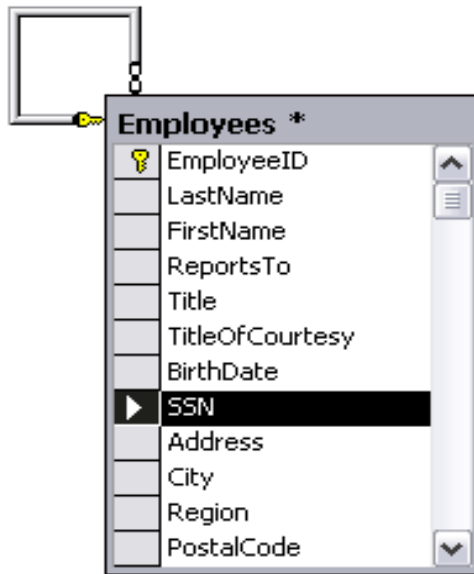


For example, the EmployeeID may uniquely identify a particular employee. If so, then the EmployeeID is a candidate key. The Social Security Number (SSN) can also be another candidate key, as it can also be used to define a unique row in the table. One of these candidate keys may be chosen as a primary key. If EmployeeID is chosen as a primary key, SSN will become an alternate key.

Q. What are alternate keys?

A. Any candidate keys not chosen as the primary key. See previous question for definition of candidate keys.

In this example, both EmployeeID and SSN could be defined as the primary key, as both are unique. EmployeeID has been chosen as the primary key to address privacy concerns and therefore SSN becomes an alternate key.



Q. What is a primary key?






A. A primary key is a special candidate key. It is an attribute or minimal set of attributes that not only uniquely identifies a specific occurrence of the entity, but also exists for every occurrence of an entity. The difference between a primary key and any other unique key is that SQL server automatically uses the primary key definition in various Referential integrity constructs.

In other words, the primary key is a field (or set of fields) that uniquely identifies each record of a table. A table can have only one key declared as primary. If a primary key consists of more than one column, duplicate values are allowed in one of the columns, but the combination of values from all the columns in the primary key must be unique.

For example, if the combination OrderID and ProductID are used to define the primary key for the Order Details table, each OrderID can be represented in the table multiple times. In addition, each product can be represented in the table multiple times. However, each OrderID can only contain unique ProductID fields with in the order.

This will allow customers to order multiple products per order, as well as allow multiple orders to have the same product.

When you work with joins, a PRIMARY KEY constraint can be used to relate one table to another.

Order Details	
	OrderID
	ProductID
	UnitPrice
	Quantity
	Discount

**Q. What is the difference between a primary key and a unique key?
Are they the same?**

A. No, these two are two different keys. By default, a primary key creates a clustered index on the column and unique keys create a nonclustered index. Primary keys do not allow NULL values, but unique keys allow one NULL value. However, both primary and unique keys enforce uniqueness of the column on which they are defined. You can use primary key constraints to enforce uniqueness as well as referential integrity. Defining a primary key on a table helps to relate that table to other tables. These relationships help to normalize a database. A table can have only one primary key.

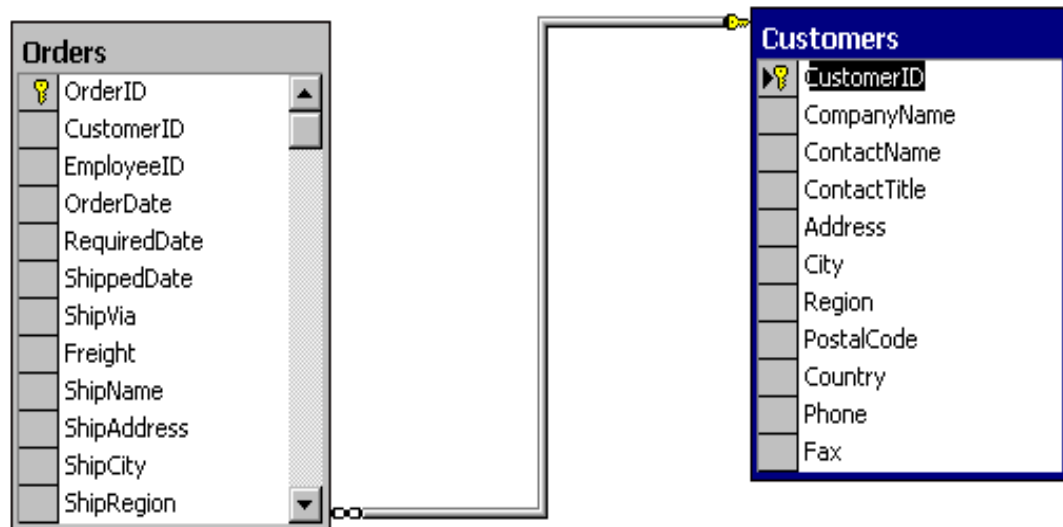
Q. What is referential integrity?

A. Referential integrity is a relationship that requires that all foreign key column values must match to a primary key column value.

In a relational database, foreign key relationships primary purpose is not really to prevent redundancy, but to prevent ORPHANED data. UNIQUE keys in conjunction with foreign keys prevent redundant data. For example, if you are designing a database that will track information about customers and company sales, you might have a table called Orders that stores information about each order, such as the Order Number, Item Number, and customer. There is also information you might want to store about the customer, such as the customer phone number, and address. If you were to store all of this information in the Orders table, the customer phone number would be duplicated for each order that the customer placed.

A better solution is to store the customer information only once in a separate table, Customer. You would then put a pointer in the Orders table that references an entry in the customer table.

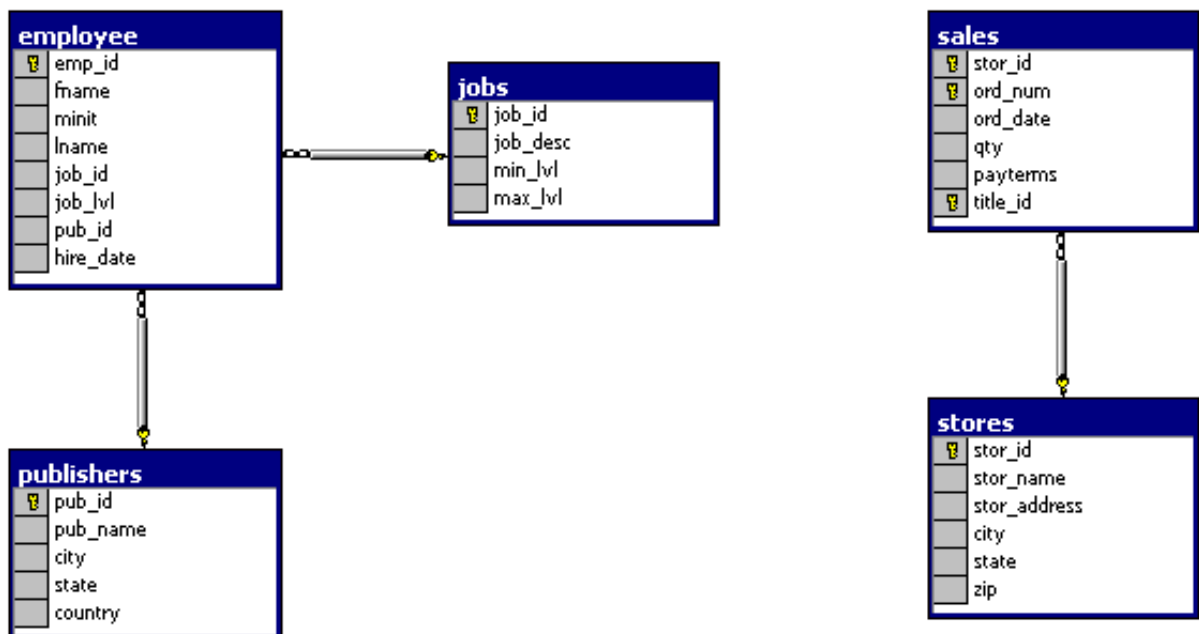
To make sure your data is consistent; you can enforce referential integrity between the Orders and Customer tables. Referential integrity relationships help ensures information in one table matches information in another. For example, each sale in the Orders table must be associated with a specific customer in the Customer table. A customer who does not exist in the Customer table cannot be added to the Orders table.



[TOC](#)

Q. What is an Entity-Relationship Diagram (ERD)?

A. An Entity Relationship Diagram (ERD) is a way to graphically describe the way tables and columns are connected in your database. Entities equate to tables, and relationships describe the connection between two entities, i.e. one-to-one, one-to-many, and many-to-many. A relationship works by matching data in key columns. A good practice is to use the same name for connected columns in a database. For example, the JobID field in an Employee table should represent the same data named JobID in a Jobs table. In most cases, the relationship matches the primary key from one table, which provides a unique identifier for each row, with an entry in the foreign key in the other table.



There are three possible types of relationships between tables. The type of relationship depends on how the related columns are defined. The three types of relationships are:

- One-to-Many Relationships
- Many-to-Many Relationships
- One-to-One Relationships

One-to-Many Relationships

A one-to-many relationship is the most common type of relationship. In this type of relationship, a row in Table A can have many matching rows in Table B, but a row in Table B can have only one matching row in Table A. For example, the Sales and Stores tables have a one-to-many relationship with each other: each Store can have many sales, but each sale can have only one store. This relationship uses the Stor_ID field for its key field.

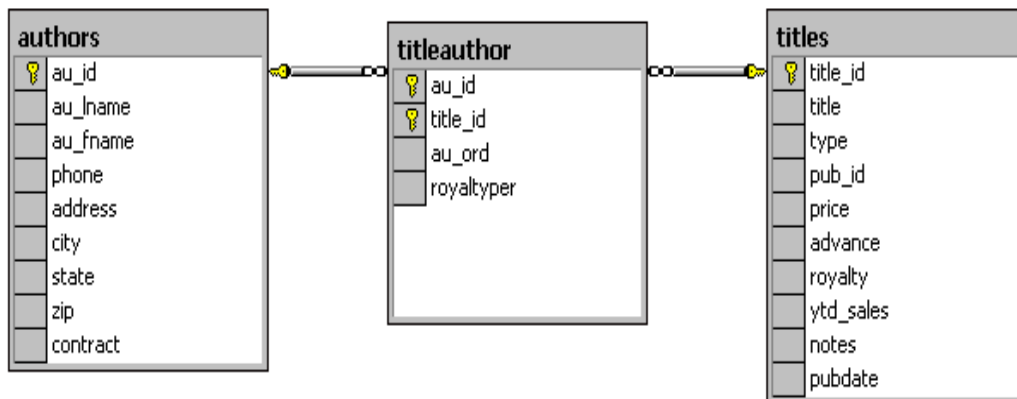
The primary key side of a one-to-many relationship is denoted by a key symbol. The foreign key side of a one-to-many relationship is denoted by an infinity symbol.

A many-to-one relationship between Table A and Table B is the same as a one-to-many relationship between Table B and Table A.

Many-to-Many Relationships

In a many-to-many relationship, a row in Table A can have many matching rows in Table B, and vice versa. In SQL Server, you create such a relationship by defining a third table, called a “junction table”, whose primary key consists of the foreign keys from both Table A and Table B. For example, the Authors table and the Titles table have a many-to-many relationship that is defined by a one-to-many relationship from each of these tables to the Titleauthors table. The primary key of the Titleauthors table is the combination of the au_id column (the Authors table's primary key) and the title_id column (the Titles table's primary key).

A many-to-many relationship is the only relationship of the three types of relationships that cannot be directly defined in SQL Server by using foreign key relationships and constraints.



One-to-One Relationships

In a one-to-one relationship, a row in Table A can have no more than one matching row in Table B, and vice versa. A one-to-one relationship is created if both of the related columns are primary keys or have unique constraints.

This type of relationship is not common. You might use a one-to-one relationship to divide a table with many columns for security purposes or to increase data access performance.

Q. What is a subquery?

A. A subquery is a nested select statement where the inner select statement is evaluated first. You can use the results of the inner select statement as the input for another select statement. For example, you can use the results of a subquery as a search condition that uses the IN () function or EXISTS operator:

```
USE northwind
```

```
SELECT * FROM Orders WHERE orderId IN
  (SELECT orderId FROM [Order Details]
   WHERE OrderId = 10248)
```

```
USE pubs
```

```
SELECT * FROM authors WHERE au_id
  NOT IN (SELECT au_id FROM titleauthor)
```

Q. What is a correlated subquery?

A. A correlated subquery is a subquery where the inner query is evaluated once for every value returned by the outer query. Generally a query which has a subquery will execute the subquery once and substitute the resulting value or values into the WHERE clause of the outer query. In queries that include a *correlated subquery* (also known as a repeating subquery), the subquery depends on the outer query for its values.

For example, using the northwind database, if you want to find out the price of all the books ordered, you can use a correlated subquery. Price information about each book is in the [Order Details] table, and the Orders table has all the books ordered.

```
SELECT O.OrderID, O.OrderDate,
  (SELECT MAX (Od.UnitPrice) FROM [Order Details] AS Od
   WHERE Od.OrderID = O.orderid) AS MaxUnitPrice
  FROM Orders AS O
```

The subquery runs once for each row that the main query returns. This repeated access to the [Order Details] table could be inefficient. Generally, correlated subqueries can be re-written as a query using a join.

[TOC](#)

Q. How do you use a subquery to find records that exist in one table and do not exist in another?

A. Use a subquery with the NOT IN clause or NOT EXISTS clause.

For example, Find out authors who do not have any records in the titleauthor table?

```
USE pubs
SELECT * FROM authors WHERE au_id
    NOT IN (SELECT au_id FROM titleauthor)
```

Preferred method is now an ANSI outer join where outside join table value is NULL.

```
SELECT a.*
FROM a
LEFT OUTER JOIN b
ON a.joinColumn = b.joinColumn
WHERE b.joinColumn IS NULL
```

Q. What does it mean to normalize a database and why would you do it?

A. Normalization is the process of organizing data in the most optimized way. This includes creating tables and establishing relationships between those tables to eliminate redundancy and inconsistent data.

There are rules for database normalization. Each rule is called a "normal form". If the first rule is observed, the database is said to be in "first normal form". If the first three rules are observed, the database is considered to be in "third normal form". Although additional levels of normalization are possible, third normal form is considered the highest level necessary for most applications. If a database adheres to the first rule and the third rule, but is not in accordance with the second rule, it is still considered in the first normal form.

Normalization provides following benefits:

- Minimizes amount of space required to store the data by eliminating redundant data.
- Minimizes the risk of data inconsistencies within a database.
- Minimizes the introduction of possible update and delete anomalies.
- Maximizes the stability of the data structure.

First Normal form (1NF):

First normal form eliminates repeating groups of data through the creation of separate tables of related data. You can reduce entities to first normal form by removing repeating or multivalued attributes to another child entity.

For example if you have an employee table with the following structure:

Employee Table

Employee	
▶	EmpNo
	DOB
	BirthYear
	FirstName
	LastName
	JanSal
	FebSal
	MarSal
	AprSal
	MaySal

To achieve first normal form, we will eliminate the repeating column, and create a salary table.

Salary Table

Salary *	
🔑	EmpNo
	[Date]
	Salary

Employee Table

Employee *	
	EmpNo
	DOB
	BirthYear
	FirstName
	LastName

In this manner, the salary information is not repeated in the Employee table, and/or the employee information is not repeated in the Salary table.

Second Normal Form (2NF):

Second normal form requires that each non-key attribute be fully dependent upon the *entire* primary key. For example, lets take a table called LineItem.

LineItem

LineItem	
▶	OrderNo
□	ItemNo
□	ItemDescription
□	Qty
□	Price

In the above table, the primary key is OrderNo and ItemNo. What we see here is a violation of the second normal form, as the Item Description and Price fields are *not* dependent on the *whole* primary key, but are only dependent on part of the primary key. So to achieve 2NF (Second Normal Form), we create another table called Item.

Item

item *	
🔑	ItemNo
□	ItemDescription
□	Price

LineItem

LineItem *	
□	OrderNo
□	ItemNo
□	Qty

Third Normal Form (3NF):

Third normal form requires that each non-key attribute depend on the entire primary key (exactly like 2NF) and nothing but the primary key .

To achieve 3NF eliminate fields that do not depend on the key.

Order

OrderNo
VendorNo
VendorCity

In the above example, the primary key OrderNo *can* uniquely determine VendorCity, but VendorCity *also depend* upon VendorNo. To normalize this table to 3NF, create another table called Vendor.

Vendor

Vendor *	
VendorNo	
VendorCity	

Order

Order *	
OderNo	
VendorNo	

Boyce/Codd normal form (BCNF):

An entity is in BCNF if and only if all attributes are determined only by each full candidate (primary or alternate) key, and not by any other subset of candidate keys.

For example, if a table already included a birth date field, it could not also include a birth year field, since this information would be redundant.

Employee Table

Employee	
<input checked="" type="checkbox"/>	EmpNo
<input type="checkbox"/>	DOB
<input type="checkbox"/>	BirthYear
<input type="checkbox"/>	FirstName
<input type="checkbox"/>	LastName

We can realize the Boyce/Codd normal form by removing the column BirthYear, as this column is dependent upon the DOB column not on the primary key Empno.

Employee Table

Employee *	
<input type="checkbox"/>	EmpNo
<input type="checkbox"/>	DOB
<input type="checkbox"/>	FirstName
<input checked="" type="checkbox"/>	LastName

Fourth Normal Form (4NF):

Fourth normal form is a form of normalization that is not commonly used which further separates any multivalued attributes that may exist in tables by separating them into child tables. For additional information, refer to SQL Server Books On-Line.

[TOC](#)

Q. What is denormalization?

A. Denormalization is the opposite of normalization process. It is deliberately introducing redundancy into a table design. In some situation, redundancy helps query performance as the number of joins can be reduced, and some calculated values can be kept in a column reducing the overhead of doing calculations with every query. An example of denormalization is a database that stores the quarterly sales figures for each item as a separate piece of data. This approach is often used in data warehouse or in on-line analytical processing applications.

Design and Programming

Q. How can I detect whether a given connection is blocked?

A. A connection is blocked when it requires an object that another connection is using and has a lock on. One way is to use SQL Enterprise; Click on Node Management to expand it, then expand Current Activity by clicking on it, then click on Locks. Once you will click on Lock/Process ID, it will show all the Locks on the right hand side window.

You can use the system stored procedure sp_lock to retrieve information about the current locks in SQL Server, to find out programmatically if a connection is blocked see

<http://www.sqlmag.com/Articles/Index.cfm?ArticleID=21882>

Check sp_who, the output from sp_who has a column named 'blk' which shows system process ID of the blocking process.

Q. How would you design a database for an online site, which would average a million hits a day?

A. This is an open ended question to check your understanding about the subject. The interviewer wants to see how inquisitive you are and how well you prepare before you jump on the design board. The interviewer certainly expects some more questions in response from you to see your analytical ability.

Before you answer any thing you must ask, about the application. What kind of application it is supporting? Is it an On-line Transaction Processing (OLTP) or Decision Support System (DSS) application? What time of day is the traffic is highest? Is the traffic evenly distributed, or does it come and go in peaks? What is the spike threshold the site is expected to endure? What is the size of the database? How many tables will be there? What is the hardware configuration? Is it preexisting, or is the interviewer expecting you to design that as well? How many hard disks you will have? What is the performance expectation? What hardware is available? Are the servers running clustering or SAN technology?

To approach this question, first get all the facts about the application and the system, and then begin reciting the best practices of database design.

For a good database design there are many things to be considered, but few of them are of paramount importance, namely :

- Normalization and appropriate denormalization of the database
- Proper indexing of the tables
- Optimum use of the stored procedures, functions etc.
- Distribution of data tables, their indices and transaction logs on different hard disks
- Proper choice of data types

After basic design with the help of a performance monitor, use the Index Tuning Wizard and Profiler to further refine the design to a more detailed level.

Above all, make sure you are designing the right system. It would be unfortunate to ramble on for 15 minutes about all the correct things for a SQL Server implementation only to find out that the database will reside on a legacy server running Linux.

Q. You are testing the performance of a query. The first time you run the query, the performance is slow. The second time you run the query, the performance is fast. Why is this?

A. The second query is faster because SQL server caches the data, and keeps the query plans in memory.

Q. What you can do to remove data from the cache and query plans from memory for testing the performance of a query repeatedly?

A. If you want to run your query and test its performance under the same circumstances each time, use command DBCC DROPCLEANBUFFERS after each run to remove the data from the memory and DBCC FREEPROCCACHE to remove the query plans from the memory.

Q. What is wrong in the following SQL statement?

```
UPDATE customer
SET c.name = u.name
FROM customer c, customerdetail cd
WHERE c.custid = cd.custid
```

A. In update clause, once an alias is defined, you must use that alias in the update clause or an error will be generated. The correct statement is below.

```
UPDATE c
SET c.name = u.name
FROM customer c, customerdetail cd
WHERE c.custid = cd.custid
```

Q. Write the fastest query to find out how many rows exist in a table.

A. The fastest query for this purpose is:

```
SELECT rows
FROM sysindexes
WHERE id = OBJECT_ID (tablename)
      AND indid < 2
```

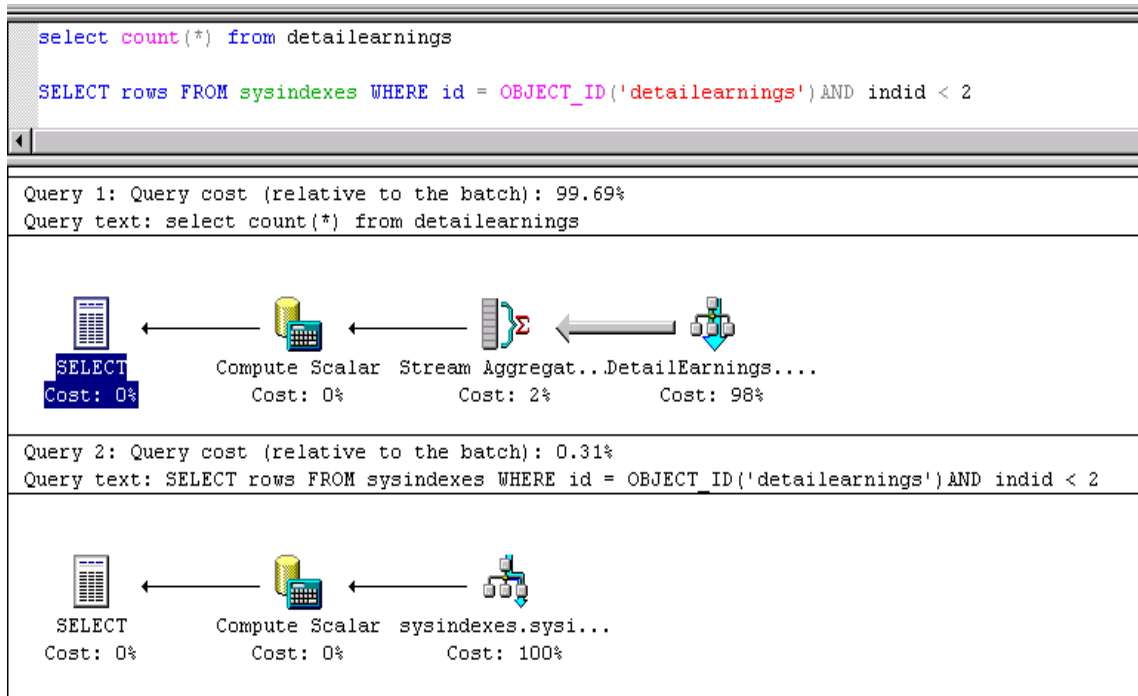
With this query, SQL does not have to do a full table scan or an index scan to find out the number of rows in a table. The *rows* column in the *sysindexes* table keeps the current rowcount dynamically for the clustered index. If table has a clustered index then *indid* = 1. If no clustered index exists then *indid* = 0 for the table.

The statement

```
SELECT COUNT(*) FROM TableName
```

requires a full table scan or full clustered index scan to determine the total number of rows. Recall that the interviewer asked for the *fastest* query that would return the results. `SELECT COUNT(*)...` will return the correct answer, but it will not be the fastest.

Pay attention to Query cost in the diagram



Q. The COUNT() function always returns a *int* value type. What should you do if you need to count rows from a query which you know will return a value that is too large for an *int* value type?

A. If you think that the count function will be returning rows more than $2^{31} - 1$, then you can use the new function COUNT_BIG, which always returns a big integer value type.

Q. What is a derived table?

A. A *derived table* is just another name for the result of using another SELECT statement in the FROM clause of a SELECT statement. The result of an inner SELECT statement is a table, which is exactly what the FROM clause requires.

```
SELECT *
FROM (SELECT * FROM authors) as T
WHERE T.city LIKE 'san%'
```

Q. You have a table with three columns: Amount1, Amount2, and Amount3. In a single row only one of these three columns will have a value and the other two will be null. Write a SQL query to retrieve the values from these columns.

A. Use the COALESCE function.

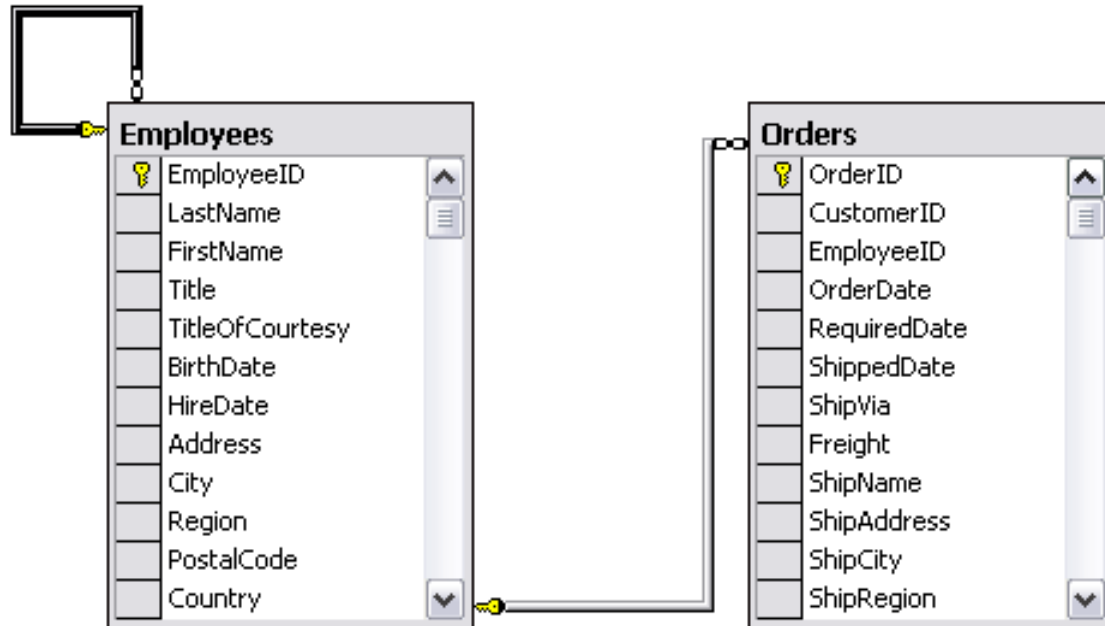
```
SELECT `Amount`= COALESCE (Amount1, Amount2, Amount3)
FROM TableName
```

COALESCE is equivalent to a searched CASE expression where a NOT NULL expression1 returns expression1 and a NULL expression1 returns expression2. In searched CASE expression form, it would look like this:

```
CASE
    WHEN expression1 IS NOT NULL THEN expression1
    ELSE expression2
END
```

COALESCE returns the first non-null value from the list.

Q. Write a query to get all of the employees and all of the orders that were processed this year. Employees who have no orders should also be displayed.



A. Use this query:

```
SELECT e.EmpID, e.FirstName, e.LastName, o.Orders
FROM Orders o
RIGHT OUTER JOIN Employee e ON
    o.EmpID = e.EmpID
    AND DATEPART(YYYY, OrderDate) = DATEPART(YYYY, GetDate())
ORDER BY EmpID
```

See outer joins for additional information.

Q. How can you fix a poorly performing query?

A. Some general issues that would cause a query to perform poorly that you could talk about would be: no indexes, table scans, missing or out of date statistics, blocking, excess recompilations of stored procedures, having procedures and triggers without the SET NOCOUNT ON directive, unnecessarily complicated joins, excessive normalization of the database, or inappropriate or unnecessary usage of cursors and temporary tables.

Some of the tools and techniques that help you troubleshoot performance problems are: SET SHOWPLAN_ALL ON, SET SHOWPLAN_TEXT ON, SET STATISTICS IO ON, SQL Server Profiler, Windows NT /2000 Performance monitor, and the graphical execution plan in Query Analyzer.

Q. What is the physical representation for a many-to-many relationship?

A. A many-to-many relationship is usually represented by a third table that resolves all valid combinations of the two entities.

For example, consider a hospital, where a patient can be treated by different doctors and a doctor can have many patients. If we have two tables, defining Patients and Doctors each, we need a third table to define the relationship between Patients and Doctors.

Another example can be found in the northwind database. There are orders, which can have many products, and each product can belong to many orders. To define the relationship between the order and product we use a third table, the [Order Details] table.



[TOC](#)

Q. What is the maximum length of an extended property?

A. The value of an extended property is a **sql_variant**. It can contain up to 7,500 bytes of data. (MRG: Add information about extended properties)

[TOC](#)

Q. In which database can extended stored procedures be added?

A. Extended stored procedures can only be added to the master database. In SQL Enterprise Manager, the master database must be selected to activate the extended stored procedures option. Only a person in the sysadmin role can add extended stored procedures.

Extended stored procedures may also be installed via *sp_addextendedproc*, which will register the procedure in the master database.

Q. Why does this query return 0?

```
SELECT (3/4) * 100
```

A. Because 3/4 (.75) is converted into integer, but the result is less than one, therefore it becomes 0. To make this query work use

```
SELECT (3.0/4) * 100
```

or

```
SELECT (CAST (3 AS DOUBLE) / (4 AS DOUBLE) ) * 100
```

[TOC](#)**Q. If a column is an image value type, how you can compare column values? How you can use this column in join clause?**

A. You cannot use the equal (=) sign to compare a column of type **image**. You have to use substring function.

E.g.

```
SELECT      d.emp, e.empname
FROM        dept d
INNER JOIN  Employee e ON
            SUBSTRING(d.empNo,1,30) = SUBSTRING(e.empno,1,30)
```

[TOC](#)**Q. Which data type cannot be used as a parameter in a UDF?**

A. The **timestamp** data type and user-defined data types are not supported.

Q. Which four data types cannot be used as a return type from a user-defined function?

A. text, ntext, image, and timestamp.

[TOC](#)**Q. Can an extended stored procedure be called from inside a user-defined function?**

A. Yes, however, the extended stored procedure, when called from inside a function, cannot return result sets to the client. Any ODS (open data services) APIs that return result sets to the client will return FAIL.

[TOC](#)

Q. How you can make a parameterized view?

A. Inline table-valued functions can be used to achieve the functionality of parameterized views.

```
CREATE FUNCTION CustomerNamesInRegion ( @RegionParameter nvarchar(30) )
RETURNS table
AS
RETURN ( SELECT CustomerID, CompanyName
        FROM Northwind.dbo.Customers
        WHERE Region = @RegionParameter
        )
GO

SELECT *
FROM CustomerNamesInRegion(N'WA')
GO
```

The RETURNS clause contains only the keyword table. You do not have to define the format of a return variable because it is set by the format of the result set of the SELECT statement in the RETURN clause.

There is no function_body delimited by BEGIN and END.

The RETURN clause contains a single SELECT statement in parentheses. The result set of the SELECT statement forms the table returned by the function. The SELECT statement used in an inline function is subject to the same restrictions as SELECT statements used in views.

[TOC](#)**Q. How long can an Error message be in RAISEERROR function?**

A. The error message can have as many as 255 characters.

Q. What are the constraints on severity level in RAISEERROR?

A. Severity levels 0 through 18 can be used by any user. For severity levels 19 through 25, the WITH LOG option is required. Only the system administrator can issue RAISERROR with a severity level of 19 through 25.

Q. How can you log an error to the server event log from a stored procedure?

A. Use the WITH LOG option in the RAISERROR function.

```
RAISEERROR ("This is an error", 19, NULL) WITH LOG
```

The WITH LOG option logs the error in the server error log and the event log. This option is required for messages with a severity level of 19 through 25, and it can be issued only by the system administrator.

Other options can be:

NOWAIT Sends messages immediately to the client server.

SETERROR Sets @@ERROR value to msg_id or 50000 regardless of the severity level.

This example returns an @@ERROR value of 50000:

```
RAISERROR('Test Only',1,2) WITH SETERROR
```

This example returns an @@ERROR value of 101:

```
RAISERROR (101, 1, 2) WITH SETERROR
```

[TOC](#)**Q. Explain the Rollup operator.**

A. The ROLLUP operator is an aggregate operator that delivers aggregates and super-aggregates for elements within a GROUP BY statement. For additional information see the example in the following question.

Q. Explain the Cube operator.

A. The ROLLUP operator is an aggregate operator that delivers aggregates and super-aggregates for elements within a GROUP BY statement. It differs from the CUBE operator only in that it is sensitive to column position in the GROUP BY clause.

Example

This example uses a SELECT query that contains an aggregate function and a GROUP BY clause, which lists pub_name, au_lname, and title, in that order.

```
SELECT pub_name, au_lname, title, "SUM" = SUM (qty)
FROM publishers, authors, titles, titleauthor, sales
WHERE publishers.pub_id = titles.pub_id
AND authors.au_id = titleauthor.au_id
AND titleauthor.title_id = titles.title_id
AND titles.title_id = sales.title_id
GROUP BY pub_name, au_lname, title
WITH ROLLUP
```

By using the ROLLUP operator, these groupings are created by moving right to left along the list of columns:

pub_name	au_lname	title	SUM (qty)
pub_name	au_lname	(null)	SUM (qty)
pub_name	(null)	(null)	SUM (qty)
(null)	(null)	(null)	SUM (qty)

The (null) value represents all values for that column.

If you use the SELECT statement without the ROLLUP operator, the statement creates a single grouping. The query returns a sum value for each unique combination of pub_name, au_lname, and title.

pub_name	au_lname	title	SUM(qty)
----------	----------	-------	----------

Compare these examples with the groupings created by using the CUBE operator on the same query:

pub_name	au_lname	title	SUM(qty)
pub_name	au_lname	(null)	SUM(qty)
pub_name	(null)	(null)	SUM(qty)
(null)	(null)	(null)	SUM(qty)
(null)	au_lname	title	SUM(qty)
(null)	au_lname	(null)	SUM(qty)
pub_name	(null)	title	SUM(qty)
(null)	(null)	title	SUM(qty)

For cube, Columns included in the GROUP BY clause are cross-referenced to produce a superset of groups. The aggregate function specified in the select_list is applied to these groups to produce summary values for the additional super-aggregate rows. The number of extra groups in the results set is determined by the number of columns included in the GROUP BY clause.

Every possible combination of the columns or expressions in the GROUP BY clause is used to produce super-aggregates. If you have n columns or expressions, there are $2^n - 1$ possible super-aggregate combinations. Mathematically, these combinations form an n-dimensional cube, which is how the operator got its name.

This example demonstrates the results set from a SELECT statement that uses the CUBE operator. The SELECT statement covers a one-to-many relationship between book titles and the quantity sold of each book. By using the CUBE operator, the statement returns an extra row.

```
SELECT title, "Qty" = SUM(qty)
FROM sales, titles
WHERE sales.title_id = titles.title_id
```

```
GROUP BY title
WITH CUBE
```

This is the results set:

Title	Qty
-----	-----
But Is It User Friendly?	30
Computer Phobic AND Non-Phobic Individuals: Behavi	20
You Can Combat Computer Stress!	35
(null)	85

Q. What is an application role and explain a scenario when you would use one?

A. Application roles allow the application, rather than SQL Server, to take over the responsibility of user authentication. A user's association with an application role is due to his ability to run an application that activates the role, rather than his being a member of the role.

Application roles are different from standard roles. Application roles contain no members. Microsoft Windows NT® 4.0 or Windows® 2000 groups, users, and roles cannot be added to application roles; the permissions of the application role are gained when the application role is activated for the user connection through a specific application or applications. Application roles are inactive by default and require a password to be activated. Approle is only active for the duration of the connection context in which it is established.

As an example of application role usage, a user, Alexis, runs a sales application that requires SELECT, UPDATE, and INSERT permissions on the Products and Orders tables in database Sales to work, but she should not have any SELECT, INSERT, or UPDATE permissions when accessing the Products or Orders tables using SQL Query Analyzer or any other tool. To ensure this security structure, create one user-database role that denies SELECT, INSERT, or UPDATE permissions on the Products and Orders tables, and adds Alexis as a member of that database role. Then create an application role in the Sales database with SELECT, INSERT, and UPDATE permissions on the Products and Orders tables. When the application runs, it provides the password to activate the application role by using `sp_setapprole`, and gains the permissions to access the Products and Orders tables. If Alexistries to log in to an instance of SQL Server using any tool except the application, she will not be able to access the Products or Orders tables.

Q. On Friday, you issued several INSERT statements using Query Analyzer. You then verified the data had been correctly entered with a SELECT statement. On Monday, your users report that the data is not there. What happened?

A. IMPLICIT_TRANSACTION mode was enabled (on) for the connection.

In this mode, after inserting or manipulating data, you must issue the COMMIT TRANSACTION statement to save the data permanently. Otherwise, once the connection is broken all your inserts are rolled back. To turn on this option use the SQL statement:

```
SET IMPLICIT_TRANSACTIONS ON
```

To turn this options off use the SQL statement:

```
SET IMPLICIT_TRANSACTIONS OFF
```

Q. You have two tables with a one to many join based on a field named ID. You need to find records in the parent table that are not represented in the child table. How would you accomplish this?

A. You can use a LEFT JOIN to show records that exist in one table, but not another. For example:

```
SELECT parent.ID, child.ID  
FROM parent  
     LEFT JOIN child  
         ON parent.ID = child.ID  
WHERE child.ID is NULL
```

Q. Give an example of why you would want to denormalize a database.

A. The best reason to denormalize a database is to increase performance.

For example, in an OLAP (On-Line Analytical Processing) application, you could denormalize the database to include summarized numbers for quarterly profit that are often reports. Instead of calculating these values each time you need them, the exist to be queried agains without calculation. This type of denormalization works best when the data changes infrequently, such as is often the case with historical data.

Q. Explain extended properties.

A. Extended properties are the properties that a user can assign to any object in a database. These extended properties can be used to store application-specific or site-specific information about the database objects. Because the property is stored in the database, all applications reading the property can evaluate the object in the same way. This helps enforce consistency in the way data is treated by all of the programs in the system.

For example, the caption of a column can be defined as an extended property. All applications can use the same caption in a user interface that displays information from that column. Another example can store help description information as an extended property.

To create an extended property use the following steps:

- 1) Open the Query Analyzer and launch the object browser.
- 2) Navigate to the column of the desired table.
- 3) Right click the column and select Extended Property.

A new dialog box will let you create the extended properties or edit existing one.

To access them from your program use the `fn_listextendedproperty` function.

Q. You have couple of stored procedures that depend on a table. You dropped the table and recreated it. What do you have to do to reestablish those stored procedure dependencies?

A. You have to recreate the stored procedure, at the time of this writing SQL2000 doesn't have very reliable automatic dependency tracking feature. This limitation may be removed in future version of SQL Server.

Q. How can you append an identity column to a temporary table?

A.

```
SELECT CommonName INTO #TempTable FROM tblApplication
ALTER TABLE #TempTable ADD NameID INT identity
```

The above query is not very optimized. Alter table command logs updates on a row-by-row basis, and if the appropriate fill factor is not specified it will split pages too. The better option can be select into, with identity function, because SQL server has to pass through the data only once and SELECT INTO uses FAST BULK LOAD API to copy the data.

```
SELECT IDENTITY (int, 1,1) AS NAMEID
CommonName INTO #TempTable FROM tblApplication
```

Write both the queries in SQL Query analyzer and view the query execution plan for each.

Q. You schedule a job to run every minute. What will happen if the first job runs more than 1 min? Will the second instance of the job start?

A. No, Only one instance of a job can run at a time. If you execute a job manually while it is running as scheduled, SQL Server Agent refuses the request. The timer for the second job will start only after first job ends.

Q. Why should you use or avoid SELECT * statements?

A. You can use a SELECT * statement when you are familiar with the data structure and are aware of which columns you will be returning.

SELECT * statements should be avoided in several cases. For example, in the case of a stored procedure, if an underlying table structure is modified that the SELECT statement relies upon, aspects of the stored procedure will change. You may also notice a performance decrease when using a SELECT * when many columns are added to the table specified in the FROM clause.

Q. What is wrong with this statement?

```
SELECT TOP 5  
FROM tblTable
```

A. The compiler will return an error because no output columns were specified. A correct statement would be

```
SELECT TOP 5 *  
FROM tblTable
```

Q. You have several tables, and they are joined together for querying. The tables contain both clustered indexes and non clustered indexes. To optimize performance, how should you distribute the tables and their indexes onto different file groups?

A. Try to keep the non-clustered indexes on different file groups from their tables and on different disks so that they can be scanned in parallel for data. You cannot separate clustered indexes from the underlying table.

Q. Which event (Check constraints, Foreign Key, Rule, Trigger, Primary key check) will be performed last for an integrity check?

A. Triggers are performed last for integrity checks.

Q. What is the fastest way to permanently delete a 1 million row table named customers?

A. Truncate the customers table and then drop table customers.

The TRUNCATE TABLE statement is a fast, nonlogged method of deleting all rows in a table. It is almost always faster than a DELETE statement with no conditions because DELETE logs each row deletion and TRUNCATE TABLE logs only the deallocation of whole data pages. TRUNCATE TABLE immediately frees all the space occupied by data and indexes. The distribution pages for all indexes are also freed.

When a TRUNCATE TABLE is performed, the definition of the table remains in the database, along with its indexes and other associated objects. The DROP TABLE statement must be used to drop the definition of the table.

Q. In a stored procedure, some errors terminate the stored procedure, how you can bypass these errors and continue execution? For example

```
USE Northwind
GO
SELECT * FROM aTable
IF @@ERROR<>0 BEGIN
    PRINT ' Table does not exist'
    -- create a new table here
END
```

A. One workaround is to use the sp_executesql command.

```
DECLARE @err int
EXEC @err = sp_executesql N' SELECT * FROM aTable'
IF @err <> 0 BEGIN
    PRINT ' Table does not exist'
    -- create a new table
END
```

Other alternative is write a separate procedure, and check @@ERROR after the execution.

```
CREATE PROCEDURE CheckTable
AS
SELECT * FROM aTable

USE Northwind
GO
EXEC CheckTable
IF @@ERROR <> 0 BEGIN
    PRINT 'Table does not exist'
    -- create a new table
END
```

Q. After removing a table from database, what other related objects have to be dropped explicitly?

A. Referring Stored Procedures and views that refer to the table need to be dropped explicitly.

DROP TABLE removes a table definition and all data, indexes, triggers, constraints, and permission specifications for that table, however, any view or stored procedure that references the dropped table must be explicitly dropped by using the DROP VIEW or DROP PROCEDURE statement.

DROP TABLE cannot be used to drop a table being referenced by a FOREIGN KEY constraint. The referencing FOREIGN KEY constraint or the referencing table must first be dropped.

When a table is dropped, “rules” or “defaults” on it lose their binding, and any constraints or triggers associated with it are automatically dropped. If you re-create a table, you must rebind the appropriate rules and defaults, re-create any triggers, and add all necessary constraints.

You cannot use the DROP TABLE statement on system tables.

If you delete all rows in a table (DELETE tablename) or use the TRUNCATE TABLE statement, the table exists until it is dropped.

Q. You want to check the syntax of a complicated Update SQL statement without executing it. What command should you use?

A. SET NOEXEC ON

When SET NOEXEC is ON, Microsoft SQL Server compiles each batch of Transact-SQL statements but does not execute them. When SET NOEXEC is OFF, all batches are executed after compilation.

The execution of statements in SQL Server consists of two phases: compilation and execution. This setting is useful for having SQL Server validate the syntax and object names in Transact-SQL code when executing. It is also useful for debugging statements that would usually be part of a larger batch of statements.

SET NOEXEC is set at execute or run time and not at parse time.

Q. You are the database developer for a leasing company. Your leasing database includes a lease table that has a column which keeps Social security numbers. Each SSN must be unique. You want the data to be physically stored in order by SSN. What constraint should you add to the SSN column on the lease table?

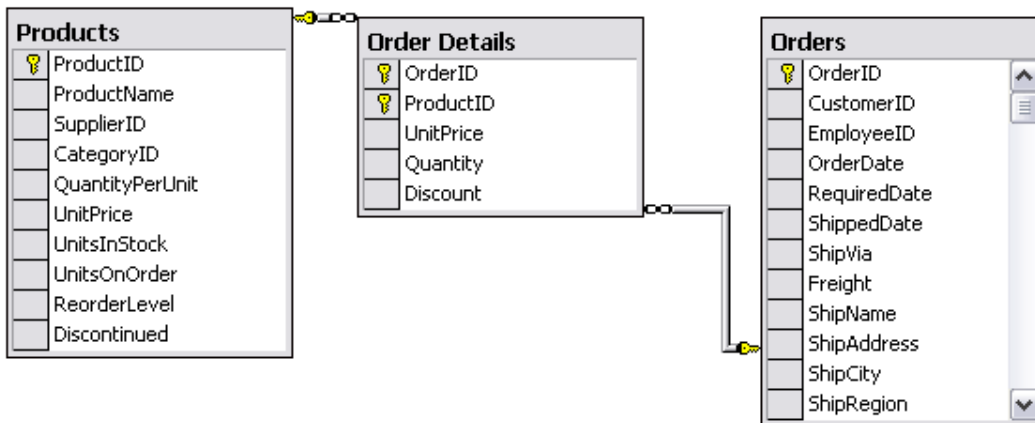
A. A UNIQUE CLUSTERED constraint should be used to achieve the requirement that the data is physically stored in the order of SSN.

Q. You are designing a database for your company. Your Company has 10 departments and each department has between 25 to 100 employees. Each employee may work for one or more departments. How can you represent this relationship in your ERD (entity relationship diagram)?

A. This is a classic example of a many-to-many relationship. Create a new table entity. Create a one-to-many relation from the employee to the new entity. Create a many-to-one relationship from the entity to the department entity.

You need to create an additional table to act as a junction table to let the many-to-many relationship be translated into two one-to-many relationships.

In an example from the northwind database, each order can have many products, and a product can belong to many orders. To represent this many to many relationship, a new table called [Order details] is created, which has relationships with the orders and products table as shown in the diagram.



Q. Mary's company is a research company that does experiments. Mary's company database includes an Experiment table. The Experiments table stores all the Experiments and their ID. There is also a Project table, which keeps all the Projects with their ID. How should the database be designed, so that an experiment cannot be repeated in a Project, but a single experiment can belong to more than one Project?

A. Add a column in Experiment table, and name that column ProjectID. For Experiment.ProjectID define a foreign key constraint from Experiment table to Projects Table.

Q. John exports information periodically from a Microsoft SQL Server database to an Oracle Database. What is the best way to do this?

A. Data Transformation Services (DTS) is the best way. Create a DTS package to connect to the Oracle Database and export the information. DTS is discussed in more depth in a later section of the book.

Q. You are designing a database for your human resources department. In the employee table, there is a field for Social Security Number, which cannot contain NULL values. If no value is given, you want a value of UNKNOWN to be inserted in this field. What is the best approach?

A. Create a DEFAULT constraint on the Social Security Number field that will populate the field with a value of UNKNOWN if the Social Security Number is left blank. Remember, this will work only if Social security field does not have unique constraint.

Alternately, in the design of the table, you can define a default property for the Social Security Number.

Q. Is it true that rules do not apply to data already existing in a database at the time the rule is created?

A. Yes. Rules do not apply to data already existing in the database at the time the rule is created.

Q. Is it true, that there is no difference between a rule and a check constraint?

A. No, it is false. Rules are a backward compatibility feature that performs some of the same functions as CHECK constraints. CHECK constraints are the preferred, standard way to restrict the values in a column.

Rules do not apply to data already existing in the database at the time the rules are created, and rules cannot be bound to system data types.

Q. Can a rule be bound to any column of any data type?

A. No. Rules cannot be bound to system data types.

Q. You are responsible for designing the physical architecture of an OLTP system. You have two tables TableA and TableB that will be joined together by several queries. You need good reliability and high performance for these queries. What should you do?

A. Create three filegroups FGA, FGB, FGC on three separate disks. Place TableA on FGA, TableB on FGB, and an index of the two tables on FGC.

To maximize performance, create files or filegroups on as many different available local physical disks as possible, and place objects that compete heavily for space in different filegroups.

Use filegroups to allow placement of objects on specific physical disks so that they can be scanned in parallel.

Place different tables frequently joined in the same query in different filegroups. This will improve performance due to parallel disk I/O searching for joined data.

Place heavily accessed tables and the nonclustered indexes belonging to those tables on different filegroups. This will improve performance, due to parallel I/O if the files are located on different physical disks. Do not place the transaction log file(s) on the same physical disk with the other files and filegroups.

Q. What is RAID, and how it can influence database performance?

A. RAID is a redundant array of inexpensive (or independent) disks.

RAID

Configure the database on a RAID 0 drive and then place the transaction log on a mirrored drive (RAID 1).

A hardware disk array improves I/O performance because I/O functions, such as striping and mirroring, are handled efficiently in firmware. Conversely, an operating system-based RAID offers lower cost but consumes processor cycles. When cost is a consideration and redundancy and high performance are required, Microsoft Windows NT(r) stripe sets with parity are a good solution.

Data striping (RAID 0) is the RAID configuration with the highest performance, but if one disk fails, all the data on the stripe set becomes inaccessible. A common installation technique for relational database management systems is to configure the database on a RAID 0 drive and then place the transaction log on a mirrored drive (RAID 1). You can get the best disk I/O performance for the database and maintain data recoverability (assuming you perform regular database backups) through a mirrored transaction log.

If data must be quickly recoverable, consider mirroring the transaction log and placing the database on a RAID 5 disk. RAID 5 provides redundancy of all data on the array, allowing a single disk to fail and be replaced in most cases without system downtime. RAID 5 offers lower performance than RAID 0 or RAID 1 but higher reliability and faster recovery.

Q. You work at the corporate office of a Fortune 500 company that has regional offices in 100 countries. Each regional office maintains its own customer table. This information needs to be brought to the corporate office and merged, to do some analysis for the marketing department. The marketing department also needs to uniquely identify each customer. How should you design the customer table?

A. Use a uniqueidentifier datatype to contain the customer ID.

```
CREATE TABLE customer(  
cust_id uniqueidentifier NOT NULL DEFAULT newid(),  
company varchar(30) NOT NULL, ...
```

A single globally unique identifier column can be created within each table that contains values unique across all networked computers in the world. A column that is guaranteed to contain globally unique values is often useful when similar data from multiple database systems must be merged (for example, in a customer billing system with data located in various company subsidiaries around the world). When the data is merged into the central site for consolidation and reporting, using globally unique values prevents customers in different countries from having the same billing number or customer ID.

Microsoft SQL Server also uses globally unique identifier columns for merge replication to ensure that rows are uniquely identified across multiple copies of the table.

SQL Server uses the IDENTITY and ROWGUIDCOL properties to implement identifier columns.

The uniqueidentifier data type stores 16-byte binary values that operate as globally unique identification numbers (GUID). A GUID is a binary number that is guaranteed to be unique; no other computer in the world will generate a duplicate of that GUID value. The main use for a GUID is for assigning an identifier that must be unique in a network that has many computers at many sites.

The uniqueidentifier data type does not automatically generate new IDs for inserted rows the way the IDENTITY property does. To get new uniqueidentifier values, a table must have a DEFAULT clause specifying the NEWID function, or INSERT statements must use the NEWID function

A table can have multiple uniqueidentifier columns. One uniqueidentifier column per table may be specified with the ROWGUIDCOL property. The ROWGUIDCOL property indicates that the uniqueidentifier values in the column uniquely identify rows in the table. The property does not do anything to enforce this, however. The uniqueness must be enforced through other mechanisms, such as specifying the PRIMARY KEY constraint for the column. The ROWGUIDCOL property is primarily used by SQL Server replication.

If you subscribe to a merge publication without initializing the subscription, you are responsible for ensuring that the table schema and data are identical for the published article and the destination table. Both the Publisher and Subscriber tables must already have a ROWGUIDCOL column and the values for the ROWGUIDCOL column at the Publisher and Subscriber must be identical. Do not simply run at both the Publisher and Subscriber a script to CREATE TABLE, add a ROWGUIDCOL column, and assign a default of newid() to the column. The script will not assign identical values because the information used to calculate the ROWGUIDCOL values is different at the Publisher and Subscriber. Instead, use DTS or bulk copy to move the ROWGUIDCOL values (and other data) from the Publisher to the Subscriber.

An alternate method that is used frequently due to the size of the uniqueidentifier data type is to give each of the branch offices a specific data range to use. For example, the first office could use 1000000000 to 1099999999 and the second could use 1100000000 to 1199999999, etc. This method may be faster but it also has the possibility for duplicates.

Choose the best method for your specific needs.

Q. Explain the DBCC PINTABLE command. When would you use it?

A. DBCC PINTABLE is best used to optimize performance from small, frequently referenced tables. The pages for the small table are read into memory the first time they are used, and subsequent references to the data does not require a disk read.

SQL Server does not flush pinned pages when it needs space to read in a new page. DBCC PINTABLE does not cause the table to be read into memory. As the pages from the table are read into the buffer cache by normal Transact-SQL statements, they are marked as pinned pages. SQL Server still logs updates to the page and, if necessary, writes the updated page back to disk. However, SQL Server does keep a copy of the page available in the buffer cache until the table is unpinned with the DBCC UNPINTABLE statement.

Q. There is a table with 100 rows of data. You want to add a new column to the table using the ALTER TABLE command. Which of the following is true?

- A. This column can allow Null values
- B. This column can be restricted to Not Null
- C. This column can be an identity column

A. The answer is both A and C. You can add an identity column to a table that already has existing data.

You can add a column, which allows null. If you want to add a column which does not allow null then you must define a default value for the column.

Q. You have to design a database for a manufacturing plant. Among other things, this database maintains the attendance of the workshop employees who work in 3 shifts. The Attendance table should have a field named 'Shift', which stores the shift that the employee worked. The Workshop supervisor should be able to input only 1, 2 or 3 in this field. What is the best design, to implement this requirement?

A. Use a check constraint. Constraints offer a way to have Microsoft SQL Server enforce the integrity of a database automatically. Constraints define rules regarding the values allowed in columns and are the standard mechanism for enforcing integrity, preferred over triggers, rules, and defaults. They are also used by the query optimizer to improve performance in selectivity estimation, cost calculations, and query rewriting. Rules, a backward compatibility feature, also perform some of the same functions as check constraints.

Q. What is a transaction and why is it important?

A. A transaction is a sequence of one or more actions, which together form a logical unit of work. Generally, the actions are interdependent and all must be successfully completed or undone for the database to remain in a consistent state. The popular example of a transaction is a bank account transfer from checking to savings. The money is withdrawn from checking and deposited to savings as one complete unit. It would be an incomplete transaction if the data were removed from checking and the savings information was not updated.

[TOC](#)**Q. What is deadlock?**

A. A deadlock occurs between two processes, when one process is waiting for an object the other process has locked. A thread in a multi-threaded system may acquire one or more resources (locks). If another thread (Thread B) currently owns the resource being acquired, the first thread (Thread A) may have to wait for the owning thread to release the target resource. The waiting thread (Thread B) is said to have a dependency on the owning thread (Thread A) for that particular resource.

If the owning thread (Thread A) wants to acquire another resource that is currently owned by the waiting thread, the situation becomes a deadlock: both threads cannot release the resources they own until their transactions are committed or rolled back, and their transactions cannot be committed or rolled back because they are waiting on resources the other owns.

Deadlocking is often confused with normal blocking. When one transaction has a lock on a resource that another transaction wants, the second transaction waits for the lock to be released. The second transaction is blocked, not deadlocked.

Q. What is a livelock?

A. A livelock occurs when a request for an exclusive lock is repeatedly denied because a series of overlapping shared locks keeps interfering. SQL Server detects the situation after four denials and refuses further shared locks. A livelock also occurs when read transactions monopolize a table or page, forcing a write transaction to wait indefinitely.

Q. How you can minimize the deadlock situation?

A. To minimize deadlocks:

Access objects in the same order.

If all concurrent transactions access objects in the same order, deadlocks are less likely to occur. Use stored procedures for all data modifications, to standardize the order of accessing objects.

Avoid user interaction in transactions.

Do not write an application where a user needs to reply a prompt requested by an application during a transaction.

Keep transactions short and in one batch.

When several long-running transactions execute concurrently in the same database, chances of having a deadlock increase. The longer the transaction, the longer the exclusive or update locks are held, blocking other activity and leading to possible deadlock situations.

Use a low isolation level.

Read committed holds shared locks for a shorter duration than a higher isolation level such as serializable.

Use bound connections.

Two or more connections opened by the same application can cooperate. Any locks acquired by the secondary connections are held as if they were acquired by the primary connection, and vice versa, and therefore do not block each other.

Q. What is the importance of concurrency control?

A. Managing records so that no more than one person can update a record at any time.

Q. Define Joins.

A. A join is a way of taking the results of one table, view, or query and correlating them with the results of another table, view, or query. Join conditions can be specified in either the FROM or WHERE clauses; specifying them in the FROM clause is recommended. WHERE and HAVING clauses can contain search conditions to further filter the rows selected by the join conditions. Joins can be categorized as inner, outer and cross joins.

Inner joins (the most common join operation, which uses some comparison operator like = or <>). These include equi-joins and natural joins.

Inner joins use a comparison operator to match rows from two tables based on the values in columns from each table.

When executing an INNER JOIN the keyword INNER is optional.

For example, here is an inner join retrieving the authors who live in the same city and state as a publisher:

```
USE          pubs
SELECT      a.au_fname, a.au_lname, p.pub_name
FROM        authors a
           INNER JOIN publishers p
           ON a.city = p.city
           AND a.state = p.state
ORDER BY    a.au_lname ASC, a.au_fname ASC
```

The tables or views in the FROM clause can be specified in any order with an inner join or full outer join; however, the order of tables or views specified when using either a left or right outer join is important.

Other kind of joins are outer join and cross join, they are discussed in detail in next question.

[TOC](#)

Q. What is an outer join?

A. A join that includes all rows from the side specified as 'outer' regardless of whether there is a matching row or not.

```
SELECT      *
FROM        authors a
           FULL OUTER JOIN titleauthor ta
           ON a.au_id = ta.au_id
```

Because an outer join includes unmatched rows, you can use it to find rows that violate foreign key constraints.

When you create an outer join, the order in which tables appear in the SQL statement is significant. The first table you add becomes the "left" table and the second table becomes the "right" table. When you specify a left or right outer join, you are referring to the order in which the tables were added to the query and to the order in which they appear in the SQL statement in the SQL pane.

There are three variations of outer join.

Left outer join All rows from the first-named table (the "left" table, which appears leftmost in the JOIN clause) are included. Unmatched rows in the right table do not appear. In this example the "left" table is titles and the "right" table is "publishers".

```
SELECT      titles.title_id,
           titles.title,
           publishers.pub_name
FROM        titles
LEFT OUTER JOIN publishers
           ON titles.pub_id = publishers.pub_id
```

The above query include all titles, even those who do not have publisher id.

Right outer join All rows in the second-named table (the "right" table, which appears rightmost in the JOIN clause) are included. A right outer join between the titles and publishers tables will include all publishers, even those who have no titles in the titles table. The resulting SQL might look like this:

```
SELECT      titles.title_id,
           titles.title,
           publishers.pub_name
FROM        titles
RIGHT OUTER JOIN publishers
           ON titles.pub_id = publishers.pub_id
```

[TOC](#)

Q. Define a cross join.

A. This is an example of a Transact-SQL cross join:

```
USE pubs
SELECT      au_fname, au_lname, pub_name
FROM        authors
           CROSS JOIN publishers
ORDER BY   au_lname DESC
```

The result set contains the number of author rows multiplied by the number of publisher rows. (For example, if authors has 23 rows and publishers has 8; 23 multiplied by 8 equals 184 total rows).

A cross join that does not have a WHERE clause produces the *Cartesian* product of the tables involved in the join. The size of a Cartesian product result set is the number of rows in the first table multiplied by the number of rows in the second table.

However, if a WHERE clause is added, the cross join behaves as an inner join.

Q. How you can change a cross join into an inner join?

A. If a where clause is added to the cross join query it will behave as an inner join.

Q. Define constraints and give an example of their use?

A. Constraints are used to maintain domain and entity integrity in a database.

Types of constraints: NOT NULL, CHECK, UNIQUE, PRIMARY KEY, FOREIGN KEY.

A NOT NULL constraint would prevent a particular column from having null entries. For example, the Birth Date field in a medical record.

The CHECK constraint will limit a column to values that are defined by the check constraint. For example, a date field could be limited to dates that are greater than or equal to the current date.

A UNIQUE constraint will return an error if a value for a column has been used in another row in the table. This is helpful when you want to enforce that all values are unique, but the column is not the Primary Key, for example, with Drivers License or Social Security Numbers.

A PRIMARY KEY constraint works similarly to the UNIQUE constraint; however, the PRIMARY KEY also impacts the physical storage of the data. SQL Server enforces a PRIMARY KEY constraint with a UNIQUE constraint.

A FOREIGN KEY constraint works similarly to a CHECK constraint, except that for its domain of valid values, it will use the PRIMARY KEY of a table. For example, a database has a BOOKS table and an AUTHORS table. The BOOKS table has an AUTHORID. The AUTHORS table has a PRIMARY KEY called AUTHORID. The BOOKS table could create a FOREIGN KEY constraint that would validate that any AUTHORID entered was already in the AUTHORS table.

Q. Write a SQL Query to sort on different column name according to the parameters passed in the function.

A.

```
CREATE FUNCTION [ListProducts] (@OrderBy int)
RETURNS TABLE
AS
RETURN
(
    SELECT productID, ProductName FROM Products
        ORDER BY
            CASE WHEN @OrderBy = 1 THEN ProductID END,
            CASE WHEN @OrderBy = 2 THEN ProductName END
)

select * from NorthWind.dbo.ListProducts (2)
```


Q. What is the downside of using UDF?

A. User defined functions are powerful and flexible; however, UDF's use row-by-row processing, which can be very expensive. Let us see it in detail with an example:

If I have a function, which gives me the day of the week

```
CREATE FUNCTION GetDayOfWeek (@Date Datetime, @Day int)
RETURNS CHAR(12)
BEGIN
    RETURN ( SELECT CONVERT (CHAR, DATEADD(dd,-(DATEPART(dw, @Date)
) - @Day ), @Date), 101))
END
```

if I execute these two statements, one with UDF, one with out it.

```
select top 100 Empid, dbo.GetDayOfWeek (JobDate, 1 ) from Timesheet

select top 100 Empid,
    CONVERT (CHAR, DATEADD(dd,-(DATEPART(dw, JobDate ) - 1 ), JobDate),
101)
from Timesheet
```

After profiling these queries, I see the Query with UDF did many more reads, took excessively much time, and consumed more CPU too.

	Read	CPU	Duration	
Query With UDF	220	18	15	
With out UDF	16	0	0	

Read more on SQL Server magazine InstantDoc ID: 25630

Q. You have a stored procedure, which execute a lengthy batch job. This stored procedure is called from a trigger. You do not want to slow the data entry process. You do not want trigger to wait for this batch job to finish before it completes itself. What you can do to speed up the process?

A. One solution can be to execute the stored procedure from a job, and that job in turn can be called from Trigger.

Using SQL Server Agent, define a job 'MyBatchJob'. This job should call the stored procedure to execute the batch job.

From trigger, this job can be called as follows:

```
EXEC msdb..sp_start_job 'MyBatchJob'
```

This job starts and run under SQL server Agent control. The *sp_start_job* procedure returns control to the calling T-SQL code as soon as the job starts. Trigger does not wait for this job to complete, rest of the trigger code executed as soon as job starts, which is a great improvement over waiting for the whole batch processing to finish. SQL server Agent initiates a new connection and run the job in background. You have to be careful about the concurrency though, now two different connections accessing the same data.

Read More on: [SQL server magazine](#). InstantDoc ID 21975

Q. Write a SQL Query to delete duplicate records from a table called TABLE1.

Select * from TABLE1

Above query produces following output

EmpID	FirstName	LastName
1	aaa	aaa
2	bbb	bbb
3	aaa	aaa
4	aaa	aaa
5	bbb	bbb
6	aaa	aaa

A. First find out the duplicate records

```
SELECT distinct A.EmpID from TABLE1 A, TABLE1 B WHERE
  ( A.FirstName = B.FirstName and A.LastName = B.LastName )
  AND A.EmpID < B.EmpID )
```

The above query will give you all the duplicate records, which can be deleted. The result of above query will be as follows:

EmpID
1
2
3
4

Now write the complete delete query:

```
DELETE FROM TABLE1 WHERE EmpID
  IN ( SELECT distinct A.EmpID from TABLE1 A, TABLE1 B WHERE
    ( A.FirstName = B.FirstName and A.LastName = B.LastName )
    AND A.EmpID < B.EmpID
  )
```

Check the results:

SELECT * FROM TABLE1

EmpID	FirstName	LastName
5	bbb	bbb
6	aaa	aaa

Indexes

Q. What is an index?

A. An index is a structure, which provides an efficient method for locating a particular row or set of rows. It is commonly used to enforce uniqueness constraints.

[TOC](#)**Q. What is the preferred way to create a clustered and non-clustered index? Which index should you create first the clustered or non-clustered?**

A. The preferred way to build indexes on tables is to start with the clustered index and then build the nonclustered indexes. When dropping all indexes, drop the nonclustered indexes first and the clustered index last. That way, no indexes need to be rebuilt.

If a clustered index is created on a table with several secondary indexes, all of the secondary indexes must be rebuilt so that they contain the clustering key value instead of the row identifier (RID). Likewise, if a clustered index is deleted on a table that has several nonclustered indexes, the nonclustered indexes are all rebuilt as part of the DROP operation.

Q. Can a unique index be created on a column, which contains NULL?

A. If a column contains NULL in more than one row, you cannot create a unique index on that column. NULL values are treated as duplicate values for indexing purposes. Therefore, if there is only one row with a null value, a unique index can be created on it, however if more than one row has a NULL value, then a unique index can not be created.

Read more: SQL Server Magazine July 2002, SQL Server Savvy column

Q. How would you choose between a clustered and a non-clustered index?

A. A clustered index determines the storage order of data in a table. A table can contain only one clustered index.

Before creating clustered indexes, understand how your data will be accessed.

Consider using a clustered index for:

- Columns that contain a limited number of distinct values, such as a state column that contains only 50 distinct state codes. However, if there are very few distinct values, such as only 1 and 0, no index should be created.
- Queries that return a range of values using operators such as BETWEEN, >, >=, <, and <=.
- Columns that are accessed sequentially.
- Queries that return large result sets.
- Columns that are frequently accessed by queries involving join or GROUP BY clauses; typically these are foreign key columns.
- Column(s) specified in the ORDER BY or GROUP BY clause. This eliminates the need for SQL Server to sort the data because the rows are already sorted.
- OLTP-type applications where very fast single row lookup is required, typically by means of the primary key. Create a clustered index on the primary key.

Clustered indexes *are not a good* choice for:

- Columns that undergo frequent changes. When columns change that are part of a clustered index, the entire row moves because SQL Server must keep the rows data values in physical order. This is an important consideration in high-volume transaction processing systems where data tends to be volatile.
- Covered queries. The more columns within the search key, the greater the chance for the data in the indexed column to change, resulting in additional I/O.

Using Nonclustered Indexes

A nonclustered index is analogous to an index in a textbook. The data is stored in one place, the index in another, with pointers to the storage location of the data. The items in the index are stored in the order of the index key values, but the information in the table is stored in a different order (which can be dictated by a clustered index). If no clustered index is created on the table, the rows are not guaranteed to be in any particular order.

Multiple Nonclustered Indexes

Some books contain multiple indexes. For example, a gardening book can contain one index for the common names of plants and another index for the scientific names because these are the two most common ways in which the readers find information. The same is true for nonclustered indexes. You can define a nonclustered index for each of the columns commonly used to find the data in the table.

Considerations

Before creating nonclustered indexes, understand how your data will be accessed. Consider using nonclustered indexes for:

- Columns that contain a high number of distinct values, such as a combination of last name and first name (if a clustered index is used for other columns). If there are very few distinct values, such as only 1 and 0, no index should be created.
- Queries that do not return large result sets.

- Columns frequently involved in search conditions of a query (WHERE clause) that return exact matches.
- Decision Support System applications for which joins and grouping are frequently required. Create multiple nonclustered indexes on columns involved in join and grouping operations, and a clustered index on any foreign key columns.

[TOC](#)

Q. Your table has a large character field. There are queries that use this field in their search clause. What should you do?

A. Creating indexes on very large fields is not advisable. SQL Server 2000 provides a new function CHECKSUM, which computes a checksum on a row or a column, and its value is always a 4 byte integer. You can create a computed column to be the checksum of your large character field and then build an index on that computed column. The values returned by CHECKSUM are not guaranteed to be absolutely unique, but there will be few duplicates. Since there is the possibility of two character string values having the same value for the checksum, your queries will need to include the full string that you're looking for.

```
DROP INDEX titles.titleind
GO

ALTER TABLE titles
ADD check_title AS CHECKSUM(title)
GO

CREATE INDEX check_title_index ON titles(check_title)
GO

SELECT      *
FROM        titles
WHERE       title = 'Straight Talk About Computers'
AND check_title =
           CHECKSUM(' Straight Talk About Computers')
```

Q. What is a fill factor?

A. When creating an index, you can specify a fill factor to leave extra gaps and reserve a percentage of free space on each leaf level page of the index to accommodate future expansion in the storage of the table data and reduce the potential for page splits. The fill factor value is a percentage from 0 to 100 that specifies how much to fill the data pages after the index are created.

Q. When you should use a low fill factor?

A. A large number of unique values and data is updated often dictate that the fill factor should be low and the index should be non-clustered.

If data is changing very frequently, keep the fill factor low (75%) so less page splits will take place, which will increase the performance.

[TOC](#)

Q. What are statistics?

A. Statistics determine the selectivity of the indexes. If an indexed column has unique values then the selectivity of that index is more, as opposed to an index with non-unique values. Query optimizer uses these statistics in determining whether to choose an index or not while executing a query.

Some situations under which you should update statistics:

- 1) If there have been significant changes in the key values in the index
- 2) If a large amount of data in an indexed column has been added, changed, or removed (that is, if the distribution of key values has changed), or the table has been truncated using the TRUNCATE TABLE statement and then repopulated
- 3) The database has been upgraded from a previous version

SQL 2000 provides auto updating of statistics.

Q. What is clustering?

A. Clustering is physically ordering the rows of table to match the order of the index.

[TOC](#)

Q. What is the difference between DBCC INDEXDEFRAG and DBCC REINDEX?

A. DBCC REINDEX drops the index and creates the index again. DBCC INDEXDEFRAG is an online operation, it does not hold long-term locks that can block running queries or updates. With INDEXDEFRAG the index is always available, unlike DBREINDEX.

DBCC INDEXDEFRAG can be considerably faster than running DBCC DBREINDEX on a relatively unfragmented index, but a large amount of fragmentation can cause INDEXDEFRAG to run considerably longer than DBREINDEX, which may or may not outweigh the benefit of its online capabilities.

To improve the clustering of pages, rebuild the index.

DBCC DBREINDEX rebuilds an index for a table or all indexes defined for a table. It can rebuild all of the indexes for a table in one statement.

DBCC DBREINDEX is not supported for use on system tables.

[TOC](#)

Q. How you can find out if an index is useful to the optimizer?

A. DBCC SHOW_STATISTICS (*table*, *target*)

The results returned indicate the selectivity of an index and provide the basis for determining whether an index is useful to the optimizer. The lower the density returned, the higher the selectivity.

Q. Where are full-text indexes stored?

A. Full-text indexes are stored in the server's file system. Regular SQL indexes are stored in the database in which they are defined.

Q. How many full-text indexes can a table have?

A. One. Each table can have only one full-text index. Each table can have several regular SQL indexes.

Q. Indexes are updated automatically. Is the full-text index also updated automatically?

A. No. Full-text indexes are not updated automatically.

Q. How is a full-text index updated?

A. Full-text indexes are not updated automatically. The stored procedure *sp_fulltext_catalog* must be run to update the full-text indexes. Regular SQL indexes are updated automatically as data is modified.

Full-text indexes are grouped into a catalog of full-text indexes for the database in which the index is defined. Regular SQL indexes are not grouped.

Full-text indexes are created, managed, and dropped via stored procedures.

Q. You have a table called 'Item', which has columns as follows:

```
[ItemID]
[Item Desc]
[Year Manufactured]
[Company Name]
```

There is a clustered index on ItemID. Most of the time you query this table on [Item Desc] and [Year Manufactured], what other indexes should you create to make your queries run faster?

A. Create a composite non-clustered index on [Item Desc] and [Year Manufactured]. There is no need to create two non-clustered indexes, as the [Year Manufactured] is not going to change much. [Year Manufactured] will not have many distinct values; therefore, a separate index on this column is not justified.

Most selective columns should be put leftmost in the key of nonclustered indexes. The index is built in the order that the columns are defined for the constraint. You should adjust the order of the columns in the constraint to make the index most useful to queries.

For this example, create an index as follows:

```
CREATE INDEX Item
ON ([Item Desc] , [Year Manufactured])
```

Note: If you query the table based **only** on [Year Manufactured], the composite index will not be used since, it is not the first column in the index.

Q. Can you force a query to use a specific Index?

A. Yes, by using index hints.

```
SELECT      fname, lname
FROM        emps (INDEX (idx_fname, idx_lname))
```

Q. Which data type columns are the best candidates for full-text indexing?

A. Datatypes text and varchar are the best candidates for full-text indexing.

Q. When would you prefer to have a minimum number of indexes?

A. In an OLTP application, you want a minimum number of indexes. An online transaction processing (OLTP) database is typically characterized by having a large number of concurrent users' actively adding and modifying data.

Indexes must be updated each time a row is added or modified. To avoid over indexing heavily updated tables, keep indexes narrow.

Q. You created a table as follows

```
CREATE TABLE TEST (  
    CUSTOMERID CHAR(5),  
    CUSTOMERNAME CHAR(30))
```

You inserted values as follows:

```
INSERT INTO TEST VALUES ('1', 'AAA')  
INSERT INTO TEST VALUES ('11', 'BBB')  
INSERT INTO TEST VALUES ('2', 'CCC')  
INSERT INTO TEST VALUES ('12', 'DDD')  
INSERT INTO TEST VALUES ('3', 'EEE')
```

After executing the following statement

```
SELECT * FROM TEST ORDER BY CUSTOMERID;
```

You see the following results:

CUSTOMERID	CUSTOMERNAME
1	AAA
11	BBB
12	DDD
2	CCC
3	EEE

How you would rewrite the SQL Query to return the CUSTOMERID sorted numerically?

A. Because the CUSTOMERID column is a character data type, the values in the column are being sorted alphabetically. To sort them numerically, in the ORDER BY statement, force a cast operation on the character value of CUSTOMERID to convert it to an integer.

```
SELECT * FROM TEST ORDER BY CAST(CUSTOMERID AS INT )
```

CUSTOMERID	CUSTOMERNAME
1	AAA
2	CCC
3	EEE
11	BBB
12	DDD

Q. Why is there a performance difference between two similar queries where one uses UNION and the other uses UNION ALL?

A. The UNION query has to sort all of the results and remove any duplicate rows, this step is performed even if there are no duplicated rows in the results. The UNION ALL query bypasses the sort step and will use the indexes you have on the individual tables.

The UNION operator allows you to combine the results of two or more SELECT statements into a single result set. Result sets combined using UNION must have the same structure. They must have the same number of columns, in the same order, and the corresponding result set columns must have compatible data types.

Q. You have a table 'test' which is a copy of northwind employee table. You have written a trigger to update the field 'HireDate' with the current date.

```
CREATE TRIGGER inserttrg ON dbo.test FOR INSERT
AS
    Declare
    @eid int
    select @eid = employeeID from inserted
    update test set HireDate = getdate() where employeeID = @eid
```

The table later on filled with the following statement:

```
INSERT INTO test SELECT * FROM Employees.
```

What happens after the above statement is executed? Will the trigger will fire and all the hire date will be filled with the current date?

A. No, SQL server triggers fire only once per statement, not for every row affected. Only the last row is updated with the current date. But rest of all the rows will have the original data.

Q. What is the difference between OPENROWSET and OPENQUERY?

A. Both of these functions execute a pass-through Query on the given linked server. OPENROWSET, Includes all connection information necessary to access remote data from an OLE DB data source. This method is an alternative to accessing tables in a linked server and is a one-time, ad hoc method of connecting and accessing remote data using OLE DB. Both the functions can be referenced in the FROM clause of a query as though it is a table name. Although the query may return multiple result sets, but these functions returns only the first one.

OPENQUERY is the preferred way, OPENROWSET has to have connection information, password, and user name hard coded into T-SQL, which makes it vulnerable to breaking, if ever the connection information changes. Writing passwords in connection string text also increase the chance of security breaches.

Q. How you can add messages to the NT event log from within a stored procedure?

A. XP_logevent will log a user-defined message into the SQL Server™ log file and the Microsoft Windows NT Event Viewer. xp_logevent can also be used to send an alert without sending a message to the client.

Q. What are three ways you can use an identity value inside a trigger? Why would you prefer one way over another?

A. There are three ways to get an Identity value. IDENT_CURRENT(), SCOPE_IDENTITY() and @@IDENTITY. All three return last-generated identity values. However, the scope and session make the result different in each of these functions.

IDENT_CURRENT is not limited by scope and session; it is limited to a specified table.

@@IDENTITY returns the last identity value generated for any table in the current session, across all scopes.

SCOPE_IDENTITY returns the last identity value generated for any table in the current session and the current scope.

For example, you have two tables, T1 and T2, and an INSERT trigger defined on T1. When a row is inserted to T1, it inserts 1111 in the identity column, the trigger fires and inserts a row in T2. For example, let say 2222 is inserted in the T2 identity column. This scenario illustrates two scopes: the insert on T1, and the insert on T2 as a result of the trigger.

IDENTITY and SCOPE_IDENTITY will return different values at the end of an INSERT statement on T1.

@@IDENTITY will return the last IDENTITY column value inserted across any scope in the current session, which is the value inserted in T2, which in this case is 2222.

SCOPE_IDENTITY() will return the IDENTITY value inserted in T1, which was the last INSERT that occurred in the same scope. It will return 1111.

IDENT_CURRENT (T1) would have returned 1111 and 2222 for IDENT_CURRENT(T2).

Choose the method that is appropriate for your needs.

Q. How can I get the name of the first day of the month using SQL Query?

A.

```
SELECT DATENAME (dw, CAST(DATEPART ( mm,GETDATE()) AS CHAR(2)) + '/1' +
 '/' + CAST(DATEPART (YYYY,GETDATE()) AS CHAR(4)))
```

Q. How can I return the name of last day of the current month using SQL Query?

A. This is a little more interesting. The easiest way to do this is with a small trick. Divide and conquer. If the current month is December, return the DATENAME value for 12/31 concatenated with the current year. If the month is not December, return the value for the first of the next month, then subtract one to return the last day of the previous month. This will return the last day of the current month and will automatically take into account leap years and leap centuries as well as months that just do not have 31 days.

```
SELECT
    CASE
        WHEN DATEPART ( mm,GETDATE()) = 12 THEN
            DATENAME(dw, CAST( '12/31/' + CAST((DATEPART (YYYY,GETDATE())) AS
            VARCHAR(4)) AS DATETIME) - 1)
        ELSE
            DATENAME(dw, CAST(CAST((DATEPART ( mm,GETDATE())+1) AS VARCHAR(2)) +
            '/1' + '/' +
            CAST((DATEPART (YYYY,GETDATE())) AS VARCHAR(4)) AS DATETIME) - 1)
```

END

View

Q. List some of the rules that apply to creating and using a 'view'.

A. When creating and using a view, these rules apply:

CREATE VIEW statements cannot be combined with other SQL statements in a single batch.

You cannot create a trigger on a view.

Data modification statements (INSERT or UPDATE) are allowed on multi-table views if the data modification statement affects only one base table. Data modification statements cannot be used on more than one table in a single statement.

INSERT statements are not allowed if a computed column exists within the view.

All column(s) being modified must adhere to all restrictions of the base table. This applies to column nullability, constraints, identity columns, and columns with rules and/or defaults and base table triggers.

UPDATE statements cannot change any column in a view that is a computation, nor can they change a view that includes aggregate functions, built-in functions, a GROUP BY clause, or DISTINCT.

You cannot use READTEXT or WRITETEXT on text or image columns in views.

Q. You added a row to a view, but the row is not shown on the view. Explain how this can happen, and how you can remedy the situation.

A. By default, data modification statements on views are not checked to determine whether the rows affected will be within the scope of the view. You can issue an INSERT statement on a view to add a row to the underlying base table, but not to add it to the view. Similarly, you can issue an UPDATE statement that changes a row so that the row no longer meets the criteria for the view. If all modifications should be checked, use the WITH CHECK option.

For example, if you insert a value of 'AAA' into a view that only returns values that are LIKE 'B%'. This would not be allowed if the WITH CHECK option was in place.

[TOC](#)

Q. Can an ORDER BY clause be used in a creation of a view?

A. Yes, but only if the TOP clause is also used.

For example:

```
CREATE VIEW [ORDERBYVIEW] AS
SELECT TOP 10 customerID, companyname
FROM      customers
ORDER BY  companyname
```

The ORDER BY clause is invalid in views, inline functions, derived tables, and subqueries, unless TOP is also specified.

Q. 'Order by' is not allowed in a view. How can you sort information from a view?

A. Using 'SELECT TOP' you can have a view sorted in the desired order, for e.g.

```
CREATE VIEW vAuthors AS
SELECT TOP 100 PERCENT au_lname, au_fname
      FROM Authors ORDER BY au_lname
GO
```

```
SELECT * FROM vAuthors
GO
```

au_lname	au_fname
Bennet	Abraham
Blotchet-Halls	Reginald
Carson	Cheryl
DeFrance	Michel
del Castillo	Innes
Dull	Ann
Green	Marjorie
Greene	Morningstar
Gringlesby	Burt
Hunter	Sheryl
Karsen	Livia
Locksley	Charlene
MacFeather	Stearns
McBadden	Heather
O'Leary	Michael

Q. What is a derived Table?

A. Derived tables are a SELECT statement that you use in a FROM clause in place of tables references.

Transact-SQL has extensions that support the specification of objects other than tables or views in the FROM clause. These other objects return a result set, or rowset that form a virtual table. The SELECT statement then operates as if the result set were a table.

These virtual tables can help you in executing certain queries that are not possible without having a view or a temp table.

```
SELECT au_lname, au_fname , royaltyper
FROM (SELECT A.au_lname, A.au_fname, TA.royaltyper
      FROM Authors A
           JOIN TitleAuthor TA on
              A.au_id = TA.au_id
      WHERE state = 'CA')
ORDER BY au_lname
```

au_lname	au_fname	royaltyper
Dull	Ann	50
Green	Marjorie	40
Green	Marjorie	100
Gringlesby	Burt	30
Hunter	Sheryl	50
Karsen	Livia	75
Locksley	Charlene	100
Locksley	Charlene	100
MacFeather	Stearns	60
MacFeather	Stearns	25
O'Leary	Michael	40
O'Leary	Michael	30
Straight	Dean	100
White	Johnson	100
Yokomoto	Akiko	40

Q. What are Information Schema Views?

A. Information Schema Views are objects that provide indirect access to the data in the system catalogs.

SQL server provides a number of views for accessing the system catalogs. These objects provide meta data and system level information from the server. You should always use these views instead of querying system catalog tables directly.

These views are ANSI SQL-92 compliant, so you can depend up on them, even if SQL server internal tables or their structure will change; these views should remain consistent.

You must refer to these views using INFORMATION_SCHEMA database schema. These views reside in MASTER database, but they run in the context of the current database.

```
USE pubs
SELECT *
FROM INFORMATION_SCHEMA.TABLES
```

There are number of ANSI SQL Schema Views are defined. Check SQL Server Books On Line for a list of views.

	TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	TABLE_TYPE
1	pubs	dbo	authors	BASE TABLE
2	pubs	dbo	discounts	BASE TABLE
3	pubs	dbo	dtproperties	BASE TABLE
4	pubs	dbo	employee	BASE TABLE
5	pubs	dbo	jobs	BASE TABLE
6	pubs	dbo	pub_info	BASE TABLE
7	pubs	dbo	publishers	BASE TABLE
8	pubs	dbo	roysched	BASE TABLE
9	pubs	dbo	sales	BASE TABLE
10	pubs	dbo	stores	BASE TABLE
11	pubs	dbo	sysconstraints	VIEW
12	pubs	dbo	syssegments	VIEW

Q. What is a partitioned view?

A. A partitioned view is a view that unites tables together making the data appear as it is from one table. For example for a large internet site may use a partitioned view to unite twelve tables for the year. Every month would have a separate table. Using partitioned views would allow these tables to be treated as a single table.

There are two types of partitioned views: Local Partitioned View (LPV) and Distributed Partitioned View (DPV). In a local partitioned view, all participating tables and the view reside on the same instance of SQL Server. In a distributed partitioned view, at least one of the participating tables resides on a different (remote) server.

To use a partitioned view, you should have partitioned tables. To create a partitioned table, you must first partition a table horizontally. Data belongs to each member table. The original table is replaced with several smaller member tables. Each member table has the same number and type of columns as the original table. In a distributed partitioned view, each member table is on a separate member server.

You design the member tables so that each table stores a horizontal slice of the original table based on a range of key values. The ranges are based on the data values in a partitioning column. The range of values in each member table is enforced by a CHECK constraint on the partitioning column, and ranges cannot overlap.

For example, you are partitioning a Sales table into three tables. The CHECK constraint for these tables is:

-- On Server1:

```
CREATE TABLE Sales1
  ([Month]    INTEGER PRIMARY KEY
   CHECK (Month = 1),
  Sales float)
```

-- On Server2:

```
CREATE TABLE Sales2
  ([Month]    INTEGER PRIMARY KEY
   CHECK (Month = 2),
  Sales float)
```

-- On Server3:

```
CREATE TABLE Sales3
  ([Month]    INTEGER PRIMARY KEY
   CHECK (Month = 3),
  Sales float)
```

After creating the member tables, you define same partitioned view on each server all having the same name. This allows queries referencing the distributed partitioned view name to run on any of the member servers. The system works as if a copy of the original table is on each member server, but in fact each server has only a member table and a distributed partitioned view. The location of the data is transparent to the application.

```
CREATE VIEW Sales AS
  SELECT * FROM DB1. Owner.Sales1
```

```
UNION ALL
  SELECT * FROM Server2.DB1. Owner.Sales2
UNION ALL
  SELECT * FROM Server3.DB1. Owner.Sales2
```

The most important rule is that a partitioning column exists on each member table and, through CHECK constraints, identifies the data available in that specific table. Partitioning column key values are enforced by CHECK constraints. The key ranges of the CHECK constraints in each table do not overlap with the ranges of any other table. Any given value of the partitioning column must map to only one table.

Q. What is an Indexed View?

A. A View is a conceptual table that is sometimes called a virtual table. Views do not store any data. The result set of a non-indexed view is not stored permanently in the database. Each time a query references the view, SQL Server dynamically merges the logic needed to build the view result set with the logic needed to build the complete query result set from the data in the base tables.

The overhead of dynamically executing query and building the result set can be substantial for views that involve complex processing of large numbers of rows.

Indexed views have the answer to this. If complex views are frequently referenced in queries, you can improve performance by creating a unique clustered index on the view. When a unique clustered index is created on a view, the view is executed and the result set is stored in the database in the same way a table with a clustered index is stored.

SQL optimizer starts using the view indexes in queries that do not explicitly refer the view in their FROM clause, because of this existing queries also benefit from the improved efficiency of the indexed view. You must have set specific SET options before you can create an index on a view. The query optimizer will not consider the index for any subsequent SQL statements unless the connection executing the statement has the same option settings. See additional detail in SQL Server Books On Line.

All indexes on a view are dropped if the view is dropped. All nonclustered indexes on the view are dropped if the clustered index is dropped. Nonclustered indexes can be dropped individually. Dropping the clustered index on the view removes the stored result set, and the optimizer returns to processing the view like a standard view.

Indexed views work best when the underlying data doesn't change often. The overhead associated with keeping the indexes updated for frequently changing data may be more than their benefit.

SQL Server Administration

Q. How you can get a list of all the table constraints in a database?

A.

```
SELECT * FROM INFORMATION_SCHEMA.CONSTRAINT_TABLE_USAGE.
```

See the question on ANSI SQL Schema view for detail.

Q. How you can get the list of largest tables in a database?

A. System table 'sysindexes' has all the information, you need.

```
SELECT object_Name (id) , dpages * 8  
FROM sysindexes  
WHERE indid in (1,0) AND objectproperty (id, 'IsUserTable') = 1  
ORDER BY rowcnt desc
```

Q. How you can move data or databases between servers and databases in SQL Server?

A. Some of the options you have are: BACKUP/RESTORE, detaching and attaching databases, replication, DTS, BCP, log shipping, INSERT...SELECT, SELECT...INTO, creating INSERT scripts to generate data.

Q. If no size is defined while creating the database, what size will the database have?

A. The size of the model database determines the initial size of a database if no size is indicated in the CREATE DATABASE statement.

Q. Can a database be shrunk with users active?

A. Yes. Users can be working in the database when it is shrunk. This includes system databases.

Q. How can you set the database to single user mode and restrict the access to dbo use only?

A. In SQL Server 2000, a database cannot be in single-user mode with **dbo** use only. Instead, the following alternative options are available by using the ALTER DATABASE command:

```
ALTER DATABASE database SET SINGLE_USER.
```

This command restricts access to the database to only one user at a time.

```
ALTER DATABASE database SET RESTRICTED_USER.
```

This command restricts access to the database to only members of the **db_owner**, **dbcreator**, or **sysadmin** roles.

```
ALTER DATABASE database SET MULTI_USER.
```

This command returns access to the database to its normal operating state.

[TOC](#)

Q. As a general practice, it is recommended to have dbo be the owner of all database objects. However, in your database you find number of tables owned by a user other than dbo, how could you fix this?

A. One alternative is

```
DECLARE
@TableName varchar (50)

DECLARE TableCur CURSOR FOR
    SELECT NAME FROM sysobjects WHERE xtype = 'U'

OPEN TableCur

FETCH NEXT FROM TableCur into @TableName
    WHILE @@FETCH_STATUS = 0
    BEGIN
        SELECT @TableName = 'OLDUSERNAME.' + @TableName
        EXEC sp_changeobjectowner @TableName , 'dbo'
        FETCH NEXT FROM TableCur into @TableName
    END

CLOSE TableCur
DEALLOCATE TableCur
```

Q.How you can list all the indexes for a table in a database?

A.

```

DECLARE
@TABLENAME varchar(255)
select @TABLENAME ='Application'

select sysindexes.name, 'TableName' = sysobjects.name,
       'INDEX-TYPE' =
       CASE
           WHEN indid = 1 THEN 'clusterd'
           WHEN indid > 1 THEN 'non-clustered'

       END
from sysindexes JOIN sysobjects on
       sysindexes.id = sysobjects.id
       where sysobjects.name = @TABLENAME
           and sysobjects.xtype = 'U'
           and indid < 255

```

Other Alternative is an undocumented function

```

Use
sp_indexes_rowset 'tableName'

```

All indexes for the database

```

Use pubs
sp_indexes_rowset; 2

```

An easier method to listing the indexes for a table is to use the SQL Enterprise Manager and right click on the table name and then choose indexes.

Q. Why does a SQL statement work correctly outside of a user-defined function, but incorrectly inside it?

A. You may have included a statement in the BEGIN-END block that has side effects, which are not allowed in user-defined functions. Function side effects are any permanent changes to the state of a resource that has a scope outside the function. Changes can be made only to local objects such as local cursors or variables. Examples of actions that cannot be performed in a function include modifications to database tables, operations on cursors that are not local to the function, sending e-mail, attempting a catalog modification.

[TOC](#)

Q. Can a table be moved to different Filegroup?

A. Once a table is created it cannot be moved to a different file group.

Q. Can a database be shrunk to 0 Bytes, if not, why?

A. A database cannot be shrunk smaller than the Model system database, which starts at 1.5 MB.

SQL Server does not shrink a database to a size any smaller than the size that was specified in the CREATE DATABASE statement.

Q. What does the Automatic recovery do?

A. The automatic recovery process is initiated when the SQL Server service starts to insure that all databases are consistent. It rolls forward any committed transactions that have not been written to the database and rolls back any uncommitted transactions.

Q. Can an automatic recovery be initiated by a user?

A. Automatic recovery is performed by the SQL Server Service when the SQL server restarts, It cannot be initiated by a user.

Q. What is the primary use of the model database?

A. The model database is used as a template for all new user databases. Users do not use the model database. Template can include specific settings, security constructs and all sorts of useful stuff.

Q. What information is maintained within the msdb database?

A. The msdb database contains information about tasks, alerts, and operators for both user-defined tasks and tasks related to replication.

Q. What stored procedure can you use to display the current processes?

A. The stored procedures sp_who and sp_who2 can be used to display the current processes as well as which process is blocking another process.

Q. What stored procedure would you use to view lock information?

A. The stored procedure sp_lock can be used to obtain information about all locks or about locks affiliated with a specific process ID.

Q. How can a database be repaired?

A. DBCC CHECKDB is the safest repair statement because it catches and repairs the widest possible range of errors. If only allocation errors are reported for a database, execute DBCC CHECKALLOC with a repair option to correct them.

DBCC CHECKDB is a superset of DBCC CHECKALLOC and includes allocation checks in addition to checks of index structure and data integrity.

DBCC CHECKTABLE checks the linkages and sizes of **text**, **ntext** and **image** pages for the specified table. However, DBCC CHECKTABLE does not check the allocations of pages in the specified table. Use DBCC CHECKALLOC to check page allocations.

DBCC CHECKTABLE requires a shared lock on all tables and indexes in the database for the duration of the operation.

To perform DBCC CHECKTABLE on every table in the database, use DBCC CHECKDB.

Q. How can you find out if the current user is a member of the specified Microsoft® Windows NT® group or Microsoft SQL Server™ role?

A. The Is_Member function.

```
IS_MEMBER ({group | role})
```

Q. Your SQL Server is running out of disk space. You notice that there are several large files with LDF extensions. What are these files?

A. LDF files are the database log files.

Q. You notice that the transaction log on one of your databases is over 4GB. The size of the data file is 2MB. What could cause this situation, and how can you fix it?

A. A large transaction log can mean several things. It is helpful to know if the log grew slowly over time or all at once. It is possible that an exceptional number of transactions occurred quickly, thereby increasing the size of the log. It is also possible that the log was set up to grow automatically and has never been truncated or shrunk. Should check for a stuck transaction too.

Q. You accidentally delete the MSDB database. What effect does this have on your existing SQL databases, and how do you recover?

A. Your ability to use SQL Server Agent to schedule alerts, jobs, and recording operators has been eliminated. To recover from this situation, restore the MSDB database from a backup.

Q. Where can you add custom error messages to SQL Server?

A. Custom messages can be added to the sysmessages table in the master database.

Q. Is it important for a Database Administrator to understand the operating system and file access?

A. You should answer this one honestly for you. Different companies have different philosophies on this, and no answer is correct for all situations.

Q. What is the difference between writing data to mirrored drives versus RAID5 drives.

A. The difference is performance. RAID5 drives can be much slower than writing to a mirrored drive set. This is due to the overhead of computing and writing the check data that is needed for recovery.

Q. In the physical file layout, where should the transaction log be stored in relation to the data file?

A. For up to the minute restorations, the transaction log should be stored on a different physical drive as the data file.

Q. You have separate development and production systems. You want to move a copy of a development database into production. To do this, you do a backup on the development system and restore to the production system. After a few minutes, you begin getting calls from several customers saying that they are denied access to the system. Why?

A. When the restore happened, you also restored the security from the development system onto the production system. The customers do not have access to the development system. One way to avoid this is to script out all the security in the production database prior to doing the restore and then to reapply the security after the restore.

Q. What is a mixed extent?

A. A mixed extent is a single extent that contains multiple tables. Extents are the basic unit in which space is allocated to tables and indexes.

To make its space allocation efficient, SQL Server does not allocate entire extents to tables with small amounts of data. SQL Server has two types of extents:

Uniform extents are owned by a single object; all eight pages in the extent can only be used by the owner object.

Mixed extents are shared by up to eight objects.

Q. You have a table with close to 100 million records. Recently, a huge amount of this data was updated. Now, various queries against this table have slowed down considerably. What is the quickest option to remedy the situation?

A. Run the UPDATE STATISTICS command.

UPDATE STATISTICS will update the statistics for the table. The sp_updatestats stored procedure will update all the tables in the database, which is not required in the situation. There is no need to recreate the index, as the updating statistics should be first step to see the query reached to the optimized level of speed.

Once the statistics have been updated, the query analyzer will be able to make better decisions about which index to use.

If Auto update Statistics is on, then you don't have to do any of this.

Q. How can you check the level of fragmentation on a table?

A. DBCC SHOWCONTIG. To display fragmentation information for the data and indexes of the specified table, use DBCC SHOWCONTIG. See SQL Server Books On-Line for additional information.

Q. You have developed an application which uses many stored procedures and triggers to update various tables. Users occasionally

get locking problems. Which tool is best suited to help you diagnose the problem?

A. SQL Server Profiler. Use SQL Server Profiler with a trace to observe Lock conflicts.

Q. Which table keeps the locking information?

A. SysLockInfo contains information on all granted, converting, and waiting lock requests. This table is a denormalized tabular view of internal data structures of the lock manager and is stored only in the master database.

Q. You want to be sure that queries in a database always execute at the maximum possible speed. To achieve this goal you have created various indexes on tables. Which other statement will keep the database in good condition?

A. Execute the Update Statistics command for tables that are accessed in the query. Update Statistics updates information about the distribution of key values for one or more statistics groups in the specified table. UPDATE STATISTICS is run automatically when an index is created on a table that already contains data.

The stored procedure Sp_Updatestats will update statistics for all the tables in the current database.

Q. During a recent migration project, John inserted 10,000 records in a table that has an Identity Column called ticketID, which automatically increases by 1 each time a record is inserted in the table. A month after the database went live; John noticed that record with ticketID 5123 has some incorrect information. So John deletes this record and decides to re-insert this record in the table. He wants to re-use the ticketID 5123. He needs to achieve this while the database is in production. What should he do?

A. John can do this with the following code:

```
SET IDENTITY_INSERT Tickets ON
GO

INSERT INTO Tickets ( ticketID, FirstName, LastName)
Values (5123, 'John,Doe)
GO
```

Set IDENTITY_INSERT Tickets ON allows explicit values to be inserted into the identity column of a table.

At any time, only one table in a session can have the IDENTITY_INSERT property set to ON. If a table already has this property set to ON, and a SET IDENTITY_INSERT ON statement is issued for another table, Microsoft® SQL Server™ returns an error message that states SET IDENTITY_INSERT is already ON and reports which table it is set ON for.

If the value inserted is larger than the current identity value for the table, SQL Server automatically uses the new inserted value as the current identity value.

The setting of SET IDENTITY_INSERT is set at execute or run time and not at parse time.

Q. Jenny wants to export data to Pivot table in Excel spreadsheet from a table in SQL Server. This data changes frequently. She wants to automate the process of updating the Excel spreadsheet using the SQL Job Scheduler. What tool is the best choice for the task?

A. Use Data Transformation Services (DTS) to populate the spreadsheet. Using DTS you can import and export data between heterogeneous sources. DTS is covered in depth later in the book.

Q. You have written an application in VB which uses SQL 7 for its database. You have used many stored procedure and triggers to make your application fast. However, users complain that saving records take too much time. To rectify the problem, you start the profiler and create a trace using the trace wizard. How would you go about identifying the cause of the problem?

A. Find the worst performing query. As you know from the users that the problem is in while saving the record, find the worst performing query. You can create a trace that captures events relating to T-SQL event classes, specifically RPC:Completed and SQL:BatchCompleted. Include all data columns in the trace, group by duration, and specify event criteria, for example, that the duration of the event must be at least 1,000 milliseconds. This event criterion eliminates short-running events from the trace. The duration minimum value can be increased as required. If you want to monitor only one database at a time, specify a value for the database ID event criteria. Use the find the worst performing queries option in the Create Trace Wizard to automatically create this trace definition.

You may be asked to do this exercise during an interview. We suggest that you should practice this before the interview.

Q. You have a table with employee information that rarely changes. However this table is used from many applications in the organization to validate the data and to produce reports. What would be the optimal fill factor to choose for indexes created on this table?

A. 100. Tables that rarely change can use fill factor of 100%. More frequent change requires low fill factor. If you set fill factor to 100, SQL Server creates both clustered and nonclustered indexes with each page 100 percent full. Setting fill factor to 100 is suitable only for read-only tables, to which additional data is never added.

The default for fill factor is 0. Smaller fill factor values cause SQL Server to create new indexes with pages that are not full. For example, a fill factor value of 10 is a reasonable choice if you are creating an index on a table that you know contains only a small portion of the data that it will eventually hold. Smaller fill factor values cause each index to take more storage space, allowing room for subsequent insertions without requiring page splits.

Since the table in question keeps the employee information and will be changed rarely, only when an employee leaves the company or a new employee joins the company, which is a rare occurrence not a daily or weekly occurrence. It is safe to have this table with 100% fill factor.

Q. What is the difference between a fill factor of 100 and 0?

A. The fill factor value is a percentage from 0 to 100 that specifies how much to fill the data pages after the index is created. A value of 100 means the pages will be full and will take the least amount of storage space. This setting should be used only when there will be no changes to the data, for example, on a read-

only table. A lower value leaves more empty space on the data pages, which reduces the need to split data pages as indexes grow but requires more storage space. This setting is more appropriate when there will be changes to the data in the table. A value of 0 behaves like 100 but leaves some extra space in the upper level of the data tree.

Q. How will you know when statistics on a table are obsolete?

A. Looking at the query plan and look at the numbers for 'Estimated Row count' and 'Row Count', if the difference is very big, say more than double or three times, it means the statistics are obsolete on the table. You should run update statistics on the table to make the statistics current.

Query 1: Query cost (relative to the batch): 100.00%

Query text: SELECT * FROM [meal]

SELECT
Cost: 1%

Table Scan
Cost: 99%

Table Scan
Scan rows from a table.

Physical operation:	Table Scan
Logical operation:	Table Scan
Row count:	16,245
Estimated row size:	241
I/O cost:	0.195
CPU cost:	0.0179
Number of executes:	1
Cost:	0.213304(99%)
Subtree cost:	0.213
Estimated row count:	16,245

Argument:
OBJECT:([dbTimeColl],[dbo].[Meal])

Grids Execution Plan Message

Query batch completed.

Out-of-date or missing statistics are indicated as warnings (table name in red text) when the execution plan of a query is graphically displayed using SQL Query Analyzer.

Q. Explain different backup plans.

A. *Full database backup*, which backs up the entire database including the transaction log.

Differential database backup performed between full database backups. You cannot perform a differential backup, if you have not already done the full database backup. Differential backup backs up all the data since the last Full backup.

Transaction log backup or incremental backup.

File(s) and Filegroup(s) backup.

Use BACKUP to back up database files and filegroups instead of the full database when time constraints make a full database backup impractical. Separate transaction log backups must be performed. After restoring a file backup, apply the transaction log to roll the file contents forward to make it consistent with the rest of the database.

Q. What is a full backup?

A. A full database backup copies all the pages from a database to a backup device; it can be a local disk file or a network disk file, a local tape drive. SQL Server also copies the portion of the transaction log that was active while the backup was in process.

Q. Explain a differential backup.

A. A differential backup copies only the extents that have changed since the last full backup. Generally, you make several differential backups between full backups, and each differential backup contains all the changes since the last full backup.

SQL Server 2000 tells which extents need to be backed up by examining a special page called the Differential Changed Map (DCM) in each file of the database. A file's DCM contains a bit for each extent in the file. Each time you make a full backup, all the bit values revert to 0. When any page in an extent is changed, the page's corresponding bit in the DCM page changes to 1. SQL Server copies the portion of the transaction log that was active during the backup.

Q. Explain an incremental backup.

A. A database has a transaction log that records data modifications made in the database. The log records every transaction. A transaction log backup copies all the log records that SQL Server has written since the last log backup. Even if you've made full database backups, a log backup always contains all the records since the last log backup.

Exact behavior of the BACKUP LOG command depends on your database's recovery-model setting. If the database is using the full recovery model, the BACKUP LOG command copies the entire contents of the transaction log. If the database is using the simple recovery model, you cannot perform a log backup because the log is truncated regularly.

In the bulk_logged recovery model, a transaction log backup copies the contents of the log and all the extents containing data pages that bulk operations have modified since the last log backup.

In a typical scenario, an administrator would make a series of log backups between full database backups, with each log backup containing only the log records recorded since the last log backup.

Q. What is Log shipping?

A. Log shipping increases a SQL Server database's availability by automatically copying and restoring the database's transaction logs to another database on a standby server. Because the standby database receives all changes to the original database, it's an exact duplicate of the original database—out of date only by the delay in the copy-and-load process.

Log shipping is better than replication in some scenarios. For example, after a switchover, you need to know exactly which transactions the standby server holds. When you use replication that information is not available, and you have to be willing to lose a certain number of transactions. However, when you use log shipping, you can find out which transactions the standby server holds because you know the last point of consistency.

Q. Every night you run a full backup. After every 3 three hours you make a differential backup. Every hour you make an incremental backup. In a worst-case scenario, how much work you can lose?

A. In the worst case you will loose, one hour of work, as every hour you take incremental backup.

Q. Explain a Checkpoint?

A. A Checkpoint is essentially a bookmark for activity in the database transaction log. A database has a transaction log that records data modifications made in the database. The log records every transaction. SQL Server stores enough information in the log to either redo (roll forward) or undo (roll back) the data modifications that make up a transaction

As writing data on disk is a slow process, SQL Server caches modifications in buffers for a period of time to optimize disk writes. A page in this cached buffer is known as a dirty page. Writing this dirty buffer page to disk is called flushing the page. In SQL Server, first log images are written to disk before the corresponding data modification.

A commit operation writes all log records for a transaction to the log file. A commit operation does not have to force all the modified data pages to disk as long as all the log records are flushed to disk. A system recovery can roll the transaction forward or backward using only the log records.

Periodically, SQL Server ensures that all dirty log and data pages are flushed. This is called a checkpoint.

Q. Explain an Automatic Checkpoint.

A. By default SQL Server always generates automatic checkpoints. The interval between automatic checkpoints does not depend on time but on the number of records in the log.

The interval between automatic checkpoints is calculated from the 'recovery interval' server configuration option. This option specifies the maximum time SQL Server should use to recover a database during a system restart. SQL Server estimates how many log records it can process in the recovery interval during a recovery operation. The interval between automatic checkpoints also depends on whether or not the database is using the simple recovery model.

If the database is using the simple recovery model, an automatic checkpoint is generated whenever the log becomes 70 percent full or the number of log records reaches the number SQL Server estimates it can process during the time specified in the **recovery interval** option.

If the database is using either the full or bulk-logged recovery model, an automatic checkpoint is generated whenever the number of log records reaches the number SQL Server estimates it can process during the time specified in the **recovery interval** option.

Automatic checkpoints truncate the unused portion of the transaction log if the database is using the simple recovery model. The log is not truncated by automatic checkpoints if the database is using the full or bulk-logged recovery models.

Q. How you can list all the tables in a database?

A. `SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES`

Or

```
SELECT name FROM sysobjects WHERE type = 'U'
```

The first query will tell you all table name plus system tables and views too. For additional information see Information Schema Views in SQL Server Books On-Line.

Q. How can you list all the columns in a database?

A. `SELECT * FROM INFORMATION_SCHEMA.COLUMNS`

For additional information see Information Schema Views in SQL Server Books On-Line.

Q. How can you list all the table constraints in a database?

A. `SELECT * FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS.`

For additional information see Information Schema Views in SQL Server Books On-Line.

Q. What are the advantages DTS has over bcp?

A. DTS has many functions that bcp does not provide, Moving data between:

SQL Server and non-SQL Server databases.

Two different kind of ODBC data sources.

DTS contains functions for modifying and transforming data fields during the bulk copy operation.

DTS can copy other objects (views, stored procedures, constraints, rules, user defined datatypes and user defined function etc) between two SQL database . It can copy even indexes and triggers between two sql databases.

Q. How do you rebuild an identity column?

A. `DBCC CHECKIDENT`

Q. Create a DTS package to produce a text file using the 'UPDATE STATISTICS' command for the tables in a database with obsolete statistics. For example, the file content should look like:

```
UPDATE STATISTICS Table1
UPDATE STATISTICS Table2
UPDATE STATISTICS table3
```

Add a task in the package to run this script file, and result should be sent to you in mail. This question is more an exercise in how to create and manage DTS package, rather than mirroring a real life scenario.

A. First write SQL to find out the table names with obsolete statistics. 'SysIndexes' tables has two fields 'rows' and 'rowmodctr', which keeps the number of total rows in a table and the number of modified rows. If more than 20% rows are modified you should update statistics of the tables. To find out the name of tables you can run the following query:

```
SELECT o.name TABLENAME, i.name INDEXNAME
FROM sysindexes i join Sysobjects o on o.id = i.id
WHERE i.id in (select id from sysobjects where Type = 'U' )
AND i.rowmodctr > (i.rows /5)
AND ( i.indid > 0 and i.indid < 255 )
```

This statement will give us the list of all the tables and their index names, which has more than 20% modified rows.

TABLENAME	INDEXNAME
Dtproperties	pk_dtproperties
timeSheetAuditing	_WA_Sys_posted_7231DAC4
timeSheetAuditing	_WA_Sys_DeptNo_7231DAC4
timeSheetAuditing	_WA_Sys_JobNo_7231DAC4
timeSheetAuditing	_WA_Sys_EmpID_7231DAC4
timeSheetAuditing	_WA_Sys_UserID_7231DAC4

To produce a UPDATE STATISTICS command, modify the above defined query

```
SELECT 'UPDATE STATISTICS ' + o.name + ' ' + i.name + ' ' AS comman
FROM sysindexes i JOIN Sysobjects o ON o.id = i.id
WHERE i.id IN (SELECT id FROM sysobjects WHERE Type = 'U' )
```

```
AND i.rowmodctr > (i.rows /5)
AND( i.indid > 0 AND i.indid < 255 )
```

This query will produce the following result:

```
UPDATE STATISTICS dtproperties pk_dtproperties
UPDATE STATISTICS timeSheetAuditing _WA_Sys_posted_7231DAC4
UPDATE STATISTICS timeSheetAuditing _WA_Sys_DeptNo_7231DAC4
UPDATE STATISTICS timeSheetAuditing _WA_Sys_JobNo_7231DAC4
```

Now once we have the proper query, we will make a DTS package:

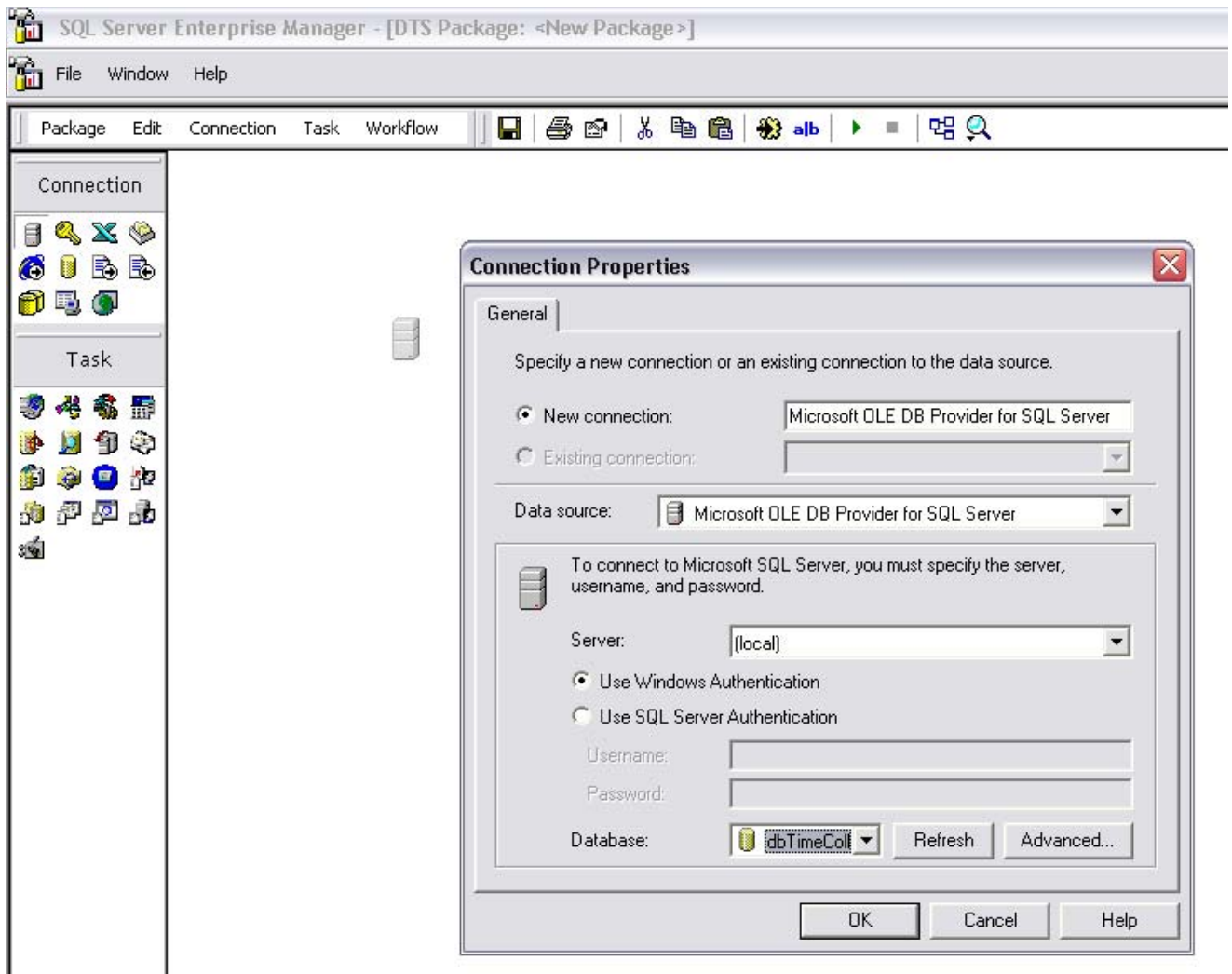
Open SQL Enterprise manager

Click on Data Transformation Services

Click on Local Packages

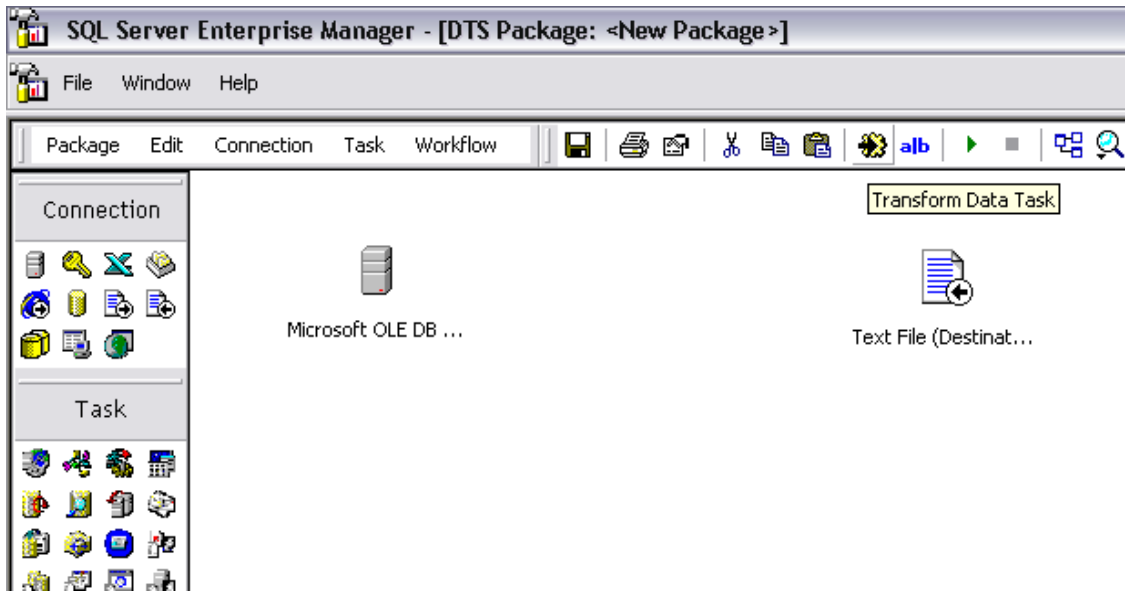
Right click and choose new package

Drag and drop 'Microsoft OLE DB Provider for SQL Server' connection, and choose the appropriate database in the connection property.

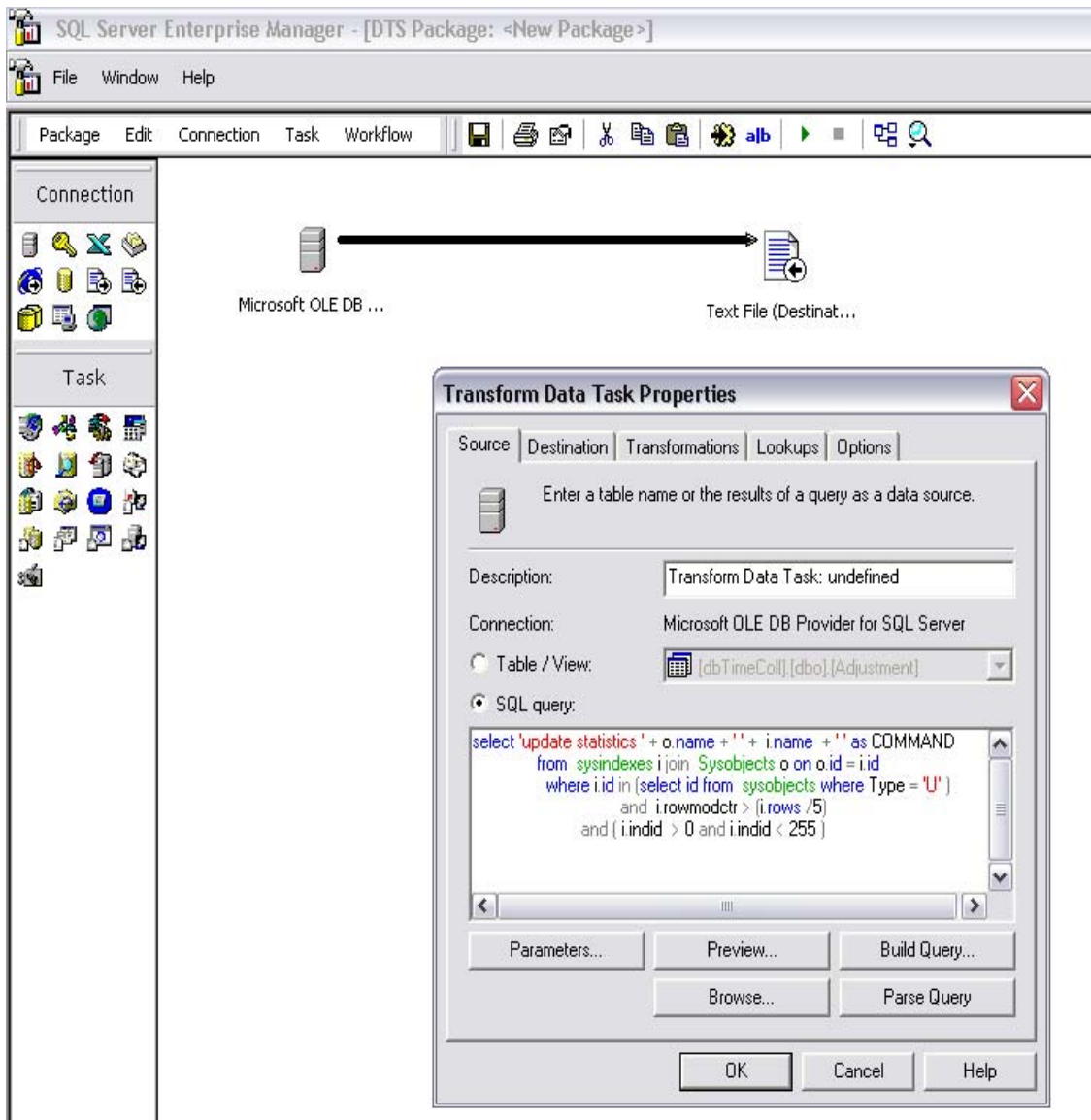


After this drag and drop the destination File. Destination file you will find on Top left window (it has a file icon, with an arrow going towards left). After dropping it on the right pane, Right click on the icon to get to property window of destination file name, type an existing file name.

After this click on Transform data task, and then click on 'Microsoft OLE DB' icon and then click on destination connection.



Once you will click on the destination connection, you will find an Arrow is formed between Source to Destination. Now click on the arrow and choose property.

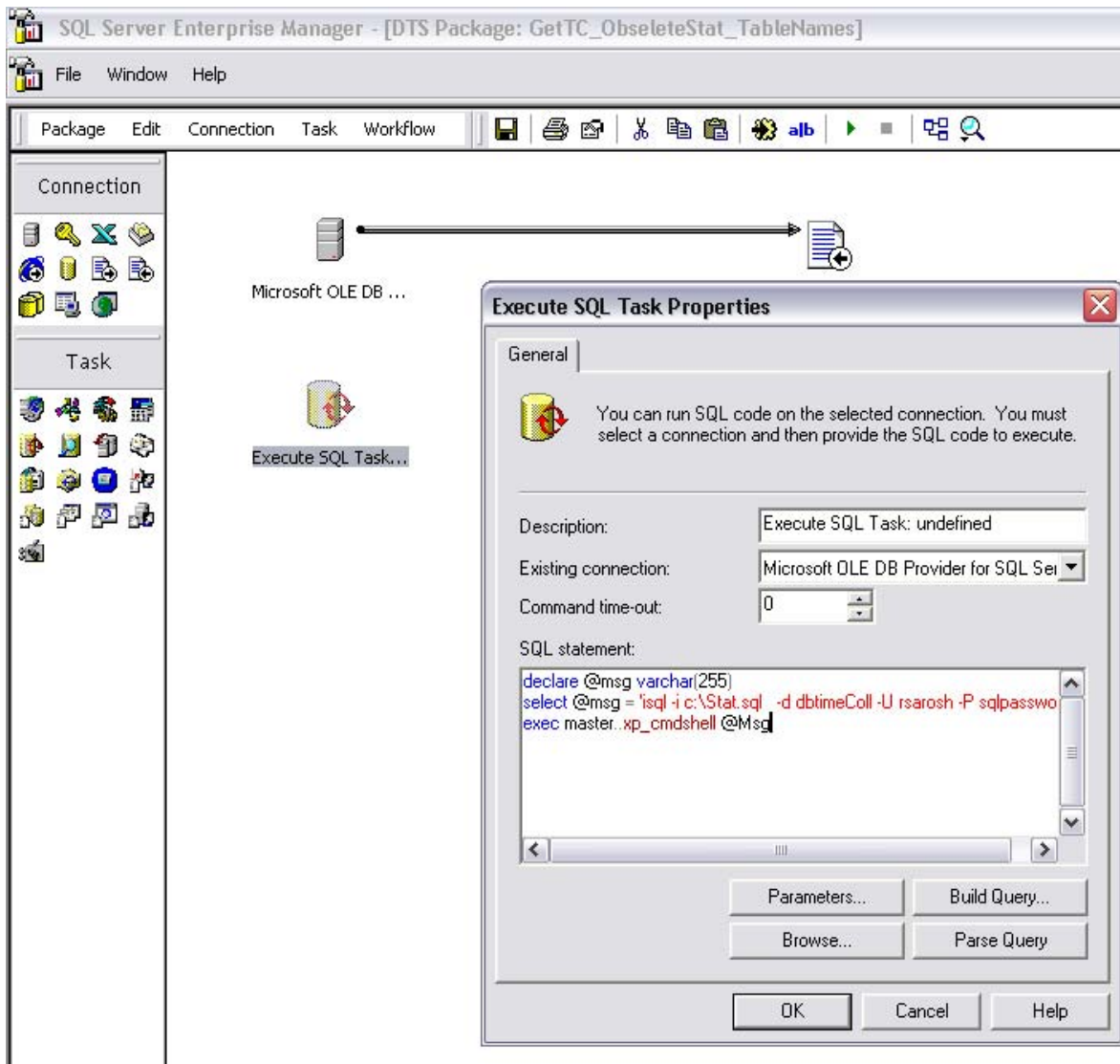


In the source tab, choose SQL Query and type the query as described above.

Click on the Destination Tab, and choose the txt file name if it is not already selected.

Now create another SQL task to run the scripts from the text file.

Drag and drop the SQL task from the left bar to design screen.



Type the query as follows to run the script file;

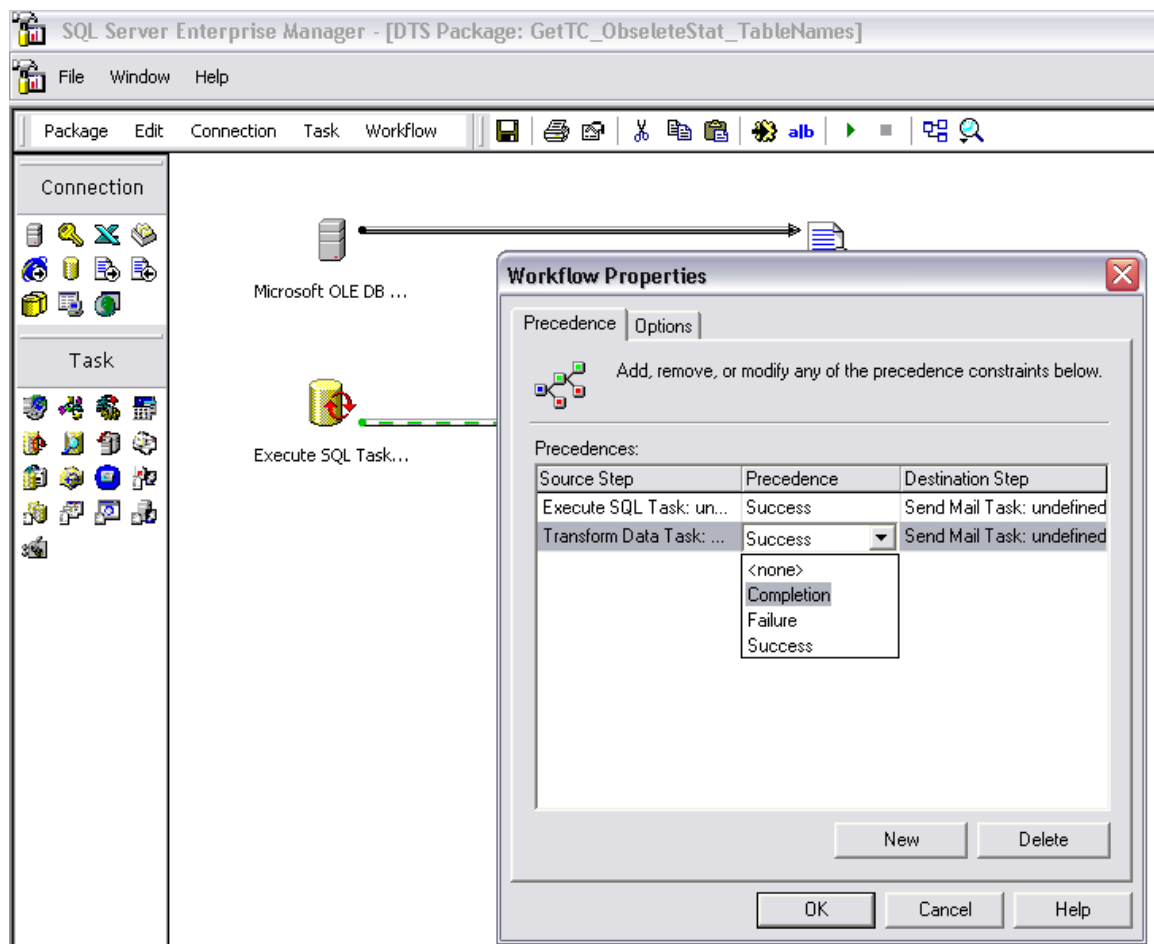
```

DECLARE @msg varchar(255)
SELECT @msg = 'isql -i c:\Stat.sql -d dbtimeColl -U rsarosh -P
sqlpassword'
EXEC master..xp_cmdshell @Msg

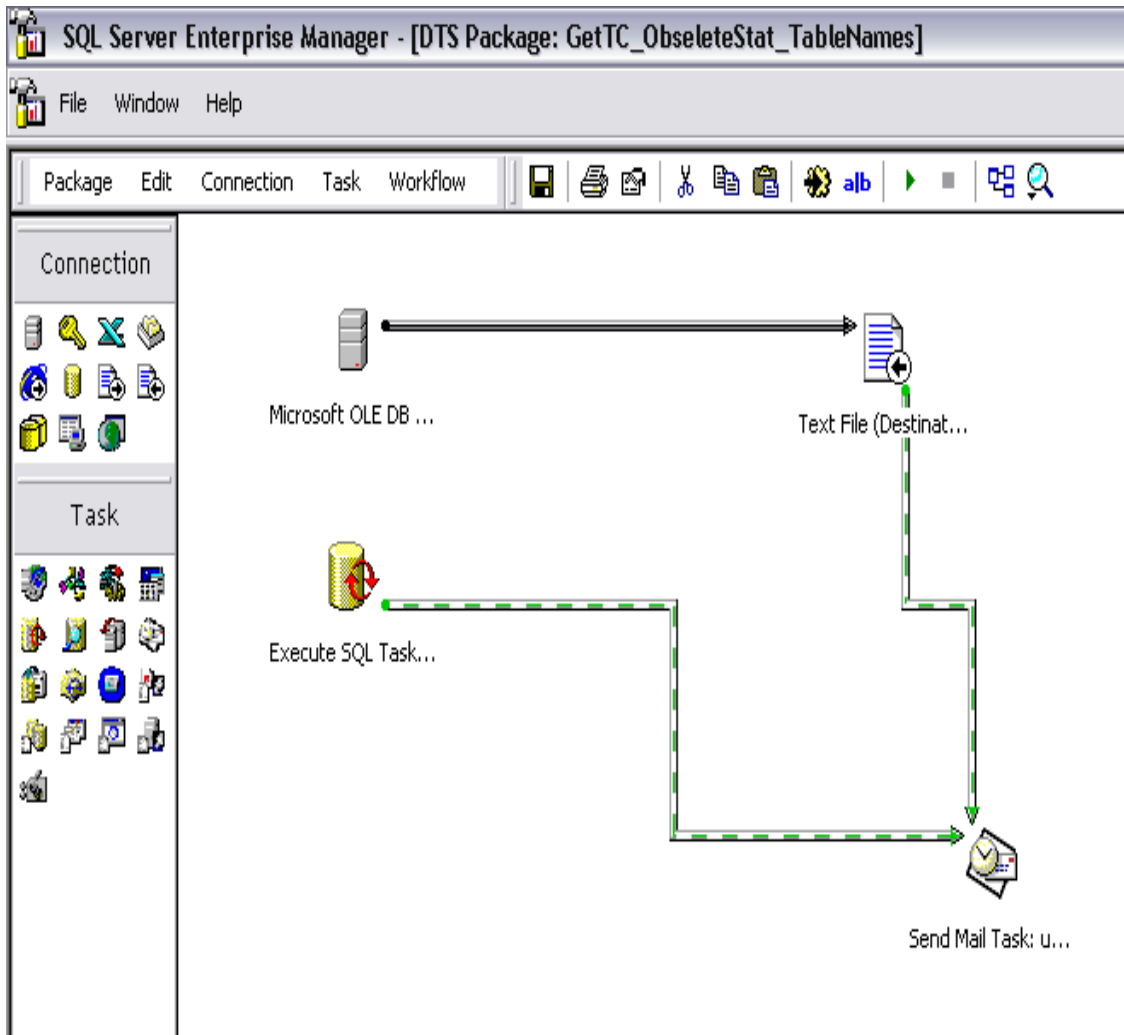
```

To get the automatic message from the package, drag and drop the Mail icon from the task pane to design. Choose the property and fill up all the required email address and subject line etc.

Right click on mail icon, and choose workflow.



In the procedure Tab, click on the New button and choose the source step and then choose precedence. Click ok and you will see the final package as follows :



Save the package by clicking Save Menu.

After saving and closing the window, go back to Local Packages and right click on your saved package, choose 'schedule package' to schedule it to automatically run.

Q. How can you transfer data from a text file to a database table? Or how can you export data from a table to a comma delimited (CSV) file? Or how can you import data from MS Access to a table in a database? Or how can you export data from a table to an Excel file?

A. To transfer data from a text file to a database table or vice versa use DTS.

Open SQL server enterprise manager, right click on a database, and choose 'All Tasks' and then 'Import task'. Follow the instruction on the DTS wizard.

You should practice couple of scenarios before you go for an interview as it often happens that you may asked to perform these tasks.

Q. When does the auto update index statistics feature in SQL server turn itself on?

A. If the number of rows in a table are greater than 6, but less than or equal to 500, then statistics are automatically updated when there have been 500 modifications made.

If the number of rows in the table is greater than 500, then updates are automatically made when (500 plus 20 percent of the number of rows in the table) has been modified.

For addtioinal information refer to <http://support.microsoft.com/directory/article.asp?ID=KB;EN-US;Q195565&>.

Q. What specific conditions database should meet, before you can Bulk copy data into it using BCP.

A.First is *Select into/bulk copy* option should be activated on the database. You can reach to this option by right clicking on the database and choosing property option.

Second, the destination table should be free of indexes.

Replication

Q. What is database replication? What are the different types of replication you can set up in SQL Server?

A. Replication is the process of copying/moving data between databases on the same or different servers. SQL Server supports the following types of replication scenarios:

Snapshot replication

Transactional replication (with immediate updating subscribers, with queued updating subscribers)

Merge replication

All these different kind of replications are very well defined in 'Books on line', please Refer to BOL for further detail.

Q. Which database stores information about replication?

A. The master database stores information about replication. The master and distribution databases keep all information about tables used in replication. See System Tables in books online. MSDB has information about the replication jobs.

Q. Your company has 50 branches all over the country. All the branches, including the Head Office have SQL Server as the database. Every night all 50 branches upload certain information to the Head Office. Which Replication topology is best suited for the above scenario?

A. Central Subscriber is the best topology for this scenario. In a Central Subscriber scenario, a number of Publishers replicate information into a common destination table at a Subscriber. The destination table is partitioned horizontally and contains a location-specific column as part of the primary key. Each Publisher replicates rows containing location-specific data.

This replication configuration may be useful, for example, for rolling up inventory data from a number of servers at local warehouses into a central Subscriber at corporate headquarters. It could also be used to roll up information from autonomous business divisions within a company, or to consolidate order processing from dispersed locations.

Q. Which of the following option(s) is(are) an inappropriate usage of merge replication: a company time sheet database, a static look up table, a high transaction based application that requires real time replication to subscribers or a company information price list that is updated at remote sites and replicated to a central database.

A. The static look up table and high transaction based applications that require real time replication to subscribers are not good candidates for merge replication.

Static look up tables are most effective with snap shot replication. High transaction based applications that require real time replication are best implemented with transactional replication.

Q. What are the restraints imposed on the table design by a Merge Replication?

A. You should not use a timestamp column in the table. Merge replication does not support timestamp columns. Timestamp values are generated automatically by the local server and guaranteed unique within a specific database only. Therefore, it is impossible for a change to the timestamp value created at one server to be applied to the timestamp column at another server. You must remove the timestamp column from any table you want to publish using merge replication.

Security

Q. A user is a member of the Public role and the Sales role. The Public role has select permission on all the tables. The Sales role does not have select permission on some of the tables. Will the user be able to select from all tables?

A. No, the user will not be able to select information from tables. Permissions assigned to a role supercede those assigned to the role Public. This is true whether permissions assigned to the role are more restrictive or less restrictive than those assigned to the role Public.

[TOC](#)

Q. If a user does not have permission to a table, but has permission to a view created on it, will he be able to view the data in table?

A. Yes. The user permission for the object referenced by a view is ignored. If a user does not have select permission on the table, but if he has permission on the view based on same table, he can select from the view.

In this case, view and table *must* belong to the same owner, if the table owner is different then the view owner, then user will not be able to select on view. The owner of the view must own the table also for the above scenario.

Q. A user has not been given a user account for any database, but has a valid login in the SQL server. Can he use the database?

A. Any user who has not been given a User Account for a database will be allowed to use permissions assigned to the User Account guest. Thus, if a user has a valid Login ID for an SQL Server but has not been assigned to a database role or given a User Account for the database, the user has access via the account guest.

A user who has been given a User Account for a database will automatically be a member of the database role public. The user may also be assigned to other database roles.

Q. Tony works in the Sales Department and has created a table named OrderDetail for the Sales department. You write a stored procedure which will help Mark, a sales representative, update the OrderDetail table. However, when Mark uses the stored procedure he gets an error. You realize that this is a security issue. What is required to enable Mark to use your stored procedure?

A. You must assign EXECUTE permission to Mark on your stored procedure, and Tony must assign UPDATE permission on the Table. For tables and views, the owner can grant INSERT, UPDATE, DELETE, SELECT, and REFERENCES permissions, or ALL permissions. A user must have INSERT, UPDATE, DELETE, or SELECT permissions on a table before they can use it in any INSERT, UPDATE, DELETE, or SELECT statements.

The owner of a stored procedure can grant EXECUTE permissions for the stored procedure. If the owner of a base table wants to prevent users from accessing the table directly, they can grant permissions on views or stored procedures referencing the table, but not grant any permissions on the table itself.

The REFERENCES permission lets the owner of another table use columns in your table as the target of a REFERENCES FOREIGN KEY constraint from their table.

You can eliminate this problem by having dbo own all objects in a database. In that case, the only thing you would need to have done was to grant Mark execute permissions to the stored procedure.

Transactions

Q. Define Concurrency Control.

A. Concurrency Control is the practice of defining controls so that multiple users can use and update data in a database simultaneously. In a multi-user system, we require some kind of mechanism to protect one-person change adversely affecting the others change. There are two main types of concurrency control, Optimistic Concurrency Control, and Pessimistic Concurrency Control.

Q. What is Pessimistic Concurrency Control?

A. In Pessimistic Concurrency Control a lock is placed on data while it is in use and no other process or user can access that data until that lock is released. Use Pessimistic Concurrency Control when the need for data integrity is paramount. Pessimistic Concurrency Control is not appropriate in a multi-user read-only situation, such as a data warehouse or on-line analytical processing application.

Q. What is Optimistic Concurrency Control?

A. In Optimistic concurrency control, works on the assumption that no other user will be changing the data. Users do not lock data when they read it. When an update is performed, the system checks to see if another user changed the data after it was read. If another user updated the data, an error is raised. Typically, the user receiving the error rolls back the transaction and starts over.

Users specify the type of concurrency control in a transactions and in cursor by specifying there isolation level.

Q. Explain Isolation levels.

A. An isolation level determines the degree of isolation of data between concurrent transactions. SQL-92 defines the following isolation levels, all of which are supported by SQL Server:

READ COMMITTED

Specifies that shared locks are held *only* while the data is being read to avoid dirty reads, but the data can be changed before the end of the transaction, resulting in nonrepeatable reads or phantom data. This option is the SQL Server default.

READ UNCOMMITTED

Implements dirty read, or isolation level 0 locking, which means that no shared locks are issued and no exclusive locks are honored. When this option is set, it is possible to read uncommitted or dirty data. Values in the data can be changed and rows can appear or disappear in the data set before the end of the transaction. This is the least restrictive of the four isolation levels.

REPEATABLE READ

Locks are placed on all data that is used in a query, preventing other users from updating the data, but new phantom rows can be inserted into the data set by another user and are included in later reads in the current transaction.

SERIALIZABLE

Places a range lock on the data set, preventing other users from updating or inserting rows into the data set until the transaction is complete. This is the most restrictive of the four isolation levels.

The default isolation level is READ COMMITTED. When the isolation level is specified, the locking behavior for all SELECT statements in the SQL Server session operates at that isolation level and remains in effect until the session terminates, or until the isolation level is set to another level.

```
USE pubs
GO
```

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
GO
```

```
BEGIN TRANSACTION
SELECT au_lname FROM authors
```

Q. What is the difference between the REPEATABLE READ and SERIALIZABLE isolation levels?

A. In a Repeatable Read isolation level, new rows can be inserted into the dataset. In a Serializable isolation level all the rows are locked for the duration of the transaction, no insert, update or delete is allowed.

Q. What is the default isolation level in SQL server?

A. READ COMMITTED is the default isolation level for SQL server. READ COMMITTED specifies that shared locks are held while the data is being read to avoid dirty reads, but the data can be changed before the end of the transaction, resulting in nonrepeatable reads or phantom data. This option is the SQL Server default.

Q. What is the most restrictive isolation level?

A. SERIALIZABLE is the most restrictive isolation level. SERIALIZABLE places a range lock on the data set, preventing other users from updating or inserting rows into the data set until the transaction is complete. This is the most restrictive of the four isolation levels.

Q. What is the least restrictive isolation level?

A. READ UNCOMMITTED is the least restrictive isolation level. READ UNCOMMITTED implements dirty read, or isolation level 0 locking, which means that no shared locks are issued and no exclusive locks are honored. When this option is set, it is possible to read uncommitted or dirty data; values in the data can be changed and rows can appear or disappear in the data set before the end of the transaction. This is the least restrictive of the four isolation levels.

Q. How do you determine the current isolation level?

A. To determine the transaction isolation level currently set, use the DBCC USEROPTIONS statement.

Q. What are the conditions an underlying table must satisfy before a cursor can be used by a positioned UPDATE or DELETE statement?

A. The SELECT statement in the cursor declaration must contain the FOR BROWSE option. To use the FOR BROWSE option, the table must have both a unique index and a timestamp column.

Q. Explain Locks.

A. A lock is an object created by SQL Server to indicate that a user is currently using a resource. The lock does not allow other users to perform operations on the resource that would adversely affect the first user who owns the lock. Locks are managed internally by system software, are acquired, and released based on actions taken by the user.

Microsoft® SQL Server™ 2000 uses locks to implement concurrency control among multiple users in a database at the same time. Transactions and locks are managed on a per connection basis.

SQL Server locks are applied at various levels of granularity in the database. Locks can be acquired on rows, pages, keys, ranges of keys, indexes, tables, or databases. SQL Server dynamically determines the appropriate level at which to place locks for each Transact-SQL statement.

The level at which locks are applied does not have to be specified by users and needs no configuration by administrators.

Q. Explain Lock escalation.

A. Lock escalation is the process of converting many fine-grain locks into fewer coarse-grain locks, reducing system overhead. Microsoft® SQL Server™ 2000 automatically escalates row locks and page locks into table locks when a transaction exceeds its escalation threshold.

SQL Server may dynamically escalate or deescalate the granularity or type of locks. For example, if an update acquires a large number of row locks and has locked a significant percentage of a table, the row locks are escalated to a table lock. If a table lock is acquired, the rowlocks are released. SQL Server 2000 rarely needs to escalate locks; the query optimizer usually chooses the correct lock granularity at the time the execution plan is compiled.

Q. Under what condition it is possible to have a page level lock and row lock at the same time for a query?

A. If enough contiguous keys in a nonclustered index node are selected to satisfy the query, SQL Server may choose to do both row and page locking for the same query, placing page locks on the index and row locks on the data. This reduces the likelihood that lock escalation will be necessary.

Q. Explain how to use transactions efficiently.

A. Get all required input from users before a transaction is started. Do not open a transaction while browsing through data. Keep the transaction as short as possible. After you get all the data, start a transaction, execute the modification statements, then immediately commit or roll back. Do not open the transaction before it is required. Make intelligent use of lower transaction isolation levels. Not all transactions require the serializable transaction isolation level. Make intelligent use of lower cursor concurrency options, such as optimistic concurrency options. Access the least amount of data possible while in a transaction. This lessens the number of locked rows, thereby reducing contention between transactions.

Triggers

Q. What you can do to delete a table without the delete trigger firing?

A. TRUNCATE TABLE statement is not caught by a DELETE trigger. Although a TRUNCATE TABLE statement is, in effect, like a DELETE without a WHERE clause (it removes all rows), it is not logged and thus cannot execute a trigger. Since permission for the TRUNCATE TABLE statement defaults to the table owner and is not transferable, only the table owner should be concerned about inadvertently circumventing a DELETE trigger with a TRUNCATE TABLE statement.

Q. Other than *Truncate* statement, which other command can by-pass the trigger on the tables?

A. The WRITETEXT statement, whether logged or unlogged, does not activate a trigger.

Disabling the **recursive triggers** setting only prevents direct recursions. To disable indirect recursion as well, set the **nested triggers** server option to 0 using **sp_configure**.

If any of the triggers do a ROLLBACK TRANSACTION, regardless of the nesting level, no further triggers are executed.

Q. What are the differences between Triggers and Stored Procedures?

A. Triggers are called automatically by SQL server in response to some action on the table, e.g. insert, delete and update. Stored Procedures can be called from triggers.

Triggers are a special class of stored procedure defined to execute automatically when an UPDATE, INSERT, or DELETE statement is issued against a table.

Triggers contain Transact-SQL statements, much the same as stored procedures.

Q. Is this statement true: A trigger can reference objects outside the current database?

A. Yes, a trigger is created only in the current database; however, a trigger can reference objects outside the current database.

Q. Can a trigger be created on a view?

A. No, a trigger cannot be created on a view.

Q. Will the WRITETEXT statement activate a trigger?

A. No, the WRITETEXT statement, whether logged or unlogged, does not activate a trigger.

Q. Can a table be created inside a Trigger?

A. CREATE TABLE statements are not allowed in a trigger.

Q. When should you use an INSTEAD OF trigger?

A. INSTEAD OF triggers are executed instead of the triggering action. INSTEAD OF triggers can also be defined on views. Each table or view can have one INSTEAD OF trigger for each triggering action (UPDATE, DELETE, and INSERT). You can work around this limitation by creating multiple views on top of the table or view in question, each with their own INSTEAD OF Triggers.

INSTEAD OF triggers allow views that would normally not support updates to allow updates. A view made up of multiple base tables must use an INSTEAD OF trigger to support inserts, updates and deletes that reference data in the tables. Another advantage of INSTEAD OF triggers is that they allow you to process data before updating the table. Let's explore this with an example

```
SET NOCOUNT ON
GO

USE tempdb
GO

CREATE TABLE Customer
(ID int Identity,
 LastName varchar(30),
 FirstName varchar (30),
 Status varchar (30)
)
GO

CREATE TABLE CreditLimit
(CustomerID int,
 CreditLimit float)
GO

INSERT Customer VALUES ('Rafat', 'Sarosh', 'Normal')
INSERT Customer VALUES ('Tony', 'Nitzke', 'Normal')
INSERT Customer VALUES ('Paul', 'Sponenburg', 'Premium')

INSERT CreditLimit VALUES (1,10000)
INSERT CreditLimit VALUES (2,15000)
INSERT CreditLimit VALUES (3,20000)

GO

CREATE VIEW ViewCustomer AS
SELECT FirstName , LastName , CreditLimit FROM Customer join CreditLimit
ON ID = CustomerID

GO

CREATE TRIGGER ViewCustomer_INSERT ON ViewCustomer INSTEAD OF INSERT
AS

INSERT Customer (FirstName, LastName,status)
SELECT  FirstName, LastName, CASE
        WHEN CreditLimit < 20000 THEN 'Normal'
        ELSE 'Premium'
      END
```

```
FROM inserted
INSERT CreditLimit SELECT SCOPE_IDENTITY() , creditlimit FROM
inserted
GO

INSERT ViewCustomer (FirstName, LastName,CreditLimit) VALUES
('Aseem','Shukla', 25000)
GO

SELECT * FROM Customer
SELECT * FROM CreditLimit
GO

DROP TABLE Customer
DROP TABLE CreditLimit
DROP VIEW ViewCustomer
GO
```

Use an INSTEAD OF trigger when you would like to:

- Ignore parts of a batch.
- Log the problem rows.
- Take an alternative action if an error condition is encountered.

Q. Can the "If Update (colName)" statement be used in a delete trigger?

A. No, IF UPDATE (column) Tests for an INSERT or UPDATE action to a specified column and is not used with DELETE operations, as you cannot delete specific columns.

[TOC](#)

Stored Procedures

Q. Why should one not prefix user stored procedures with 'sp_'?

A. If the stored procedure does not exist in the master database, do not prefix your stored procedure with 'sp_', as it causes a performance penalty. SQL server gives name resolution preference to the master database for procedure that have 'sp_' suffix. Once the SQL fails to find the stored procedure in the master database, it assumes that this procedure requires a recompilation as it is not in the cache, and acquires an exclusive lock on the stored procedure for a short time. This lock causes the performance hit.

The exclusive lock is required because only one copy of a stored procedure is generally in the cache at any time. This requires serialization of some parts of the compilation process, and this synchronization process is accomplished in part through the use of compile locks.

Read more on <http://support.microsoft.com/support/kb/articles/q263/8/89.asp>

SQL Server Magazine InstantDoc #23011

Q. What are the results of running this script?

```
CREATE PROCEDURE Test2
AS
    CREATE TABLE #t(x INT PRIMARY KEY)
    INSERT INTO #t VALUES (2)
    SELECT Test2Col = x FROM #t
GO

CREATE PROCEDURE Test1
AS
    CREATE TABLE #t(x INT PRIMARY KEY)
    INSERT INTO #t VALUES (1)
    SELECT Test1Col = x FROM #t
    EXEC Test2
GO

CREATE TABLE #t(x INT PRIMARY KEY)
    INSERT INTO #t VALUES (99)
GO

EXEC Test1
GO
```

A. 1

2

All references to the name within the stored procedure are resolved against the temporary table created in the procedure, not the version that existed before the procedure. Nested stored procedures can also create temporary tables with the same name as a temporary table created by the stored procedure that called it. All references to the table name in the nested stored procedure are resolved to the table created in the nested procedure.

Q. Which table keeps information about Stored Procedures?

A. The SysObjects table contains one row for each object (constraint, default, log, rule, stored procedure, and so on) created within a database. In tempdb, this table includes a row for each temporary object.

Q. What will be the value of @@FETCH_STATUS if a row that was a part of the cursor resultset has been deleted from the database after the time the stored procedure that opened the cursor was executed?

A -2.

Q. Why and when do stored procedure recompile?

A. Stored procedure will recompile if there is a sufficient number of rows in a table referenced by the stored procedure that have changed. SQL Server will recompile the stored procedure to be sure that the execution plan has the up-to-date statistics for the table. By default for temporary tables SQL Server determines that after 6 modifications any stored procedure referencing that table will need to be recompiled.

<http://support.microsoft.com/directory/article.asp?ID=KB;EN-US;Q195565&>

Here are few of the reasons why stored procedures will recompile:

- Dropping and recreating the stored procedure.
- Using WITH RECOMPILE clause in the CREATE PROCEDURE or the EXECUTE statement
- Changing the schema of any referenced objects
- Running the sp_recompile system stored procedure against a table referenced by the stored procedure
- Stored procedures plan dropping from the cache.
- Any reference to the temporary tables in the stored procedure.

The following SET options are ON by default in SQL Server, and changing the state of these options in your stored procedure will cause the stored procedure to recompile:

```
SET ANSI_DEFAULTS
SET ANSI_NULLS
SET ANSI_PADDING
SET ANSI_WARNINGS
SET CONCAT_NULL_YIELDS_NULL
```

A run-time recompilation of a stored procedure takes place in the following situation:

- Data in a table referenced by the routine has changed.
- The procedure contains a mixture of DDL and DML operations.
- Certain operations on temporary tables are performed.

Q. How can you find out which stored procedures are recompiling?

A. Use the SQL Profiler.

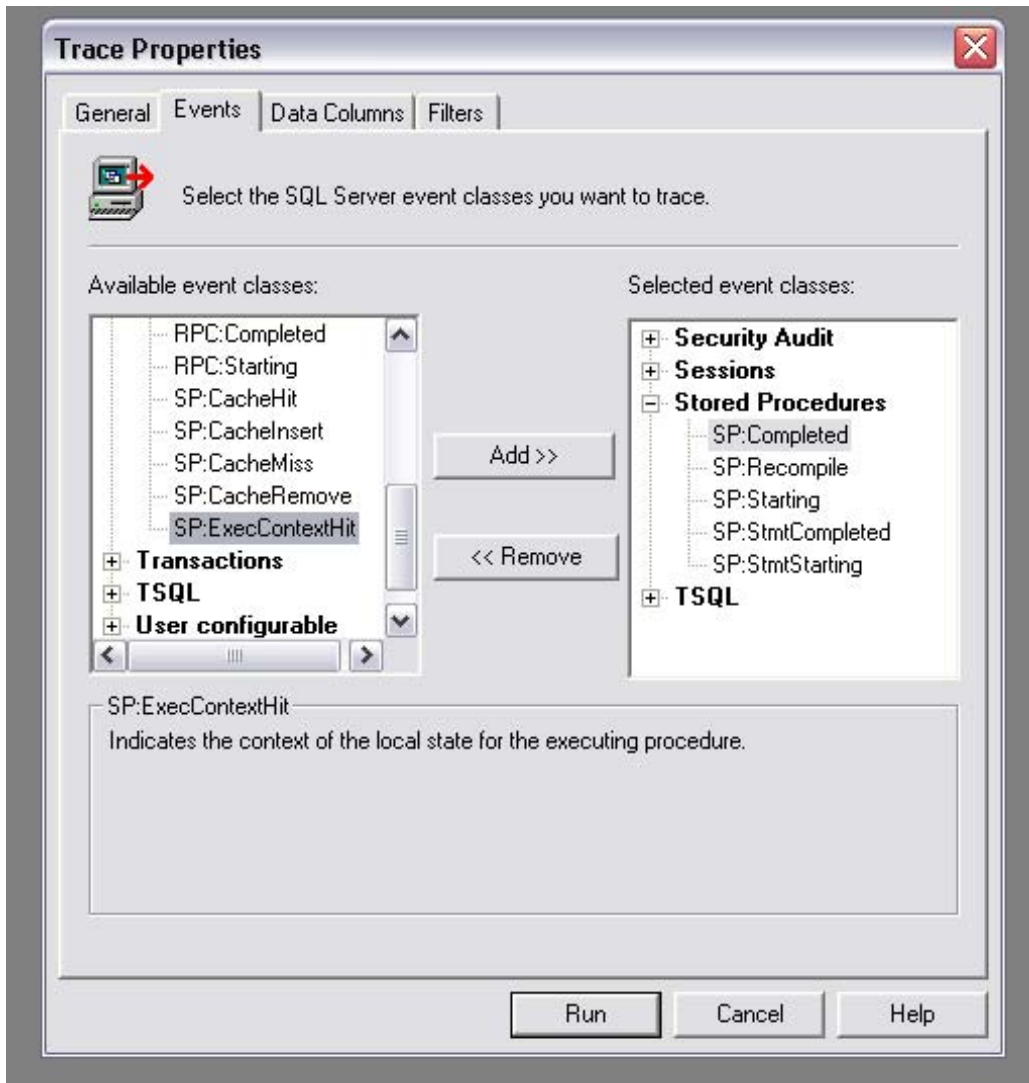
To determine if you have a problem with a specific stored procedure:

Start Profiler.

Start a new trace.

On the Events Tab, remove all default events and add SP:Recompile, SP:Starting, and SP:Completed under Stored Procedure events. If you want to determine the statement that causes the recompile also add SP:StmtStarting and SP:StmtCompleted to the selection.

Click on the filter tab, and add filter to reduce the clutter.



To determine the statement causing the recompile look at the statement immediately before and after the recompile if you included SP:StmtStarting and SP:StmtCompleted in your trace.

Q. How can you stop stored procedures from recompiling?

A. By using `sp_executesql` and using the `KEEPFIXED PLAN` query hint. `KEEPFIXED PLAN` Forces the query optimizer not to recompile a query due to changes in statistics or to the indexed column (update, delete, or insert). Specifying `KEEPFIXED PLAN` ensures that a query will be recompiled only if the schema of the underlying tables is changed or `sp_recompile` is executed against those tables.

```
SELECT COUNT(*) FROM #Temp OPTION (KEEPFIXED PLAN)
```

Recompiling takes place when temporary objects are created and referenced throughout the code. Temporary objects will not exist when the initial compilation of the code takes place, so SQL Server has to recompile the stored procedure during execution. This recompilation takes place after the temporary object is referenced for the first time.

By placing all of your temporary table creation statements together, SQL Server can create plans for those temporary tables when one of the tables is referenced for the first time. Recompile will still take place during the execution of the stored procedure, but it will happen only twice, not as many times as many temp tables you have. (Once for the stored procedure and once when the first reference to a temporary table is made).

SQL Server will also be able to reuse the execution plan for the stored procedure the next time the procedure is called there will be no recompiles. Like permanent objects, if you change the schema of a temporary table, the change will cause the stored procedure to recompile as well. Make all schema changes (such as index creation) right after your create table statements and before you reference any of the temporary tables.

Use SQL Profiler to determine if and when your procedures are being recompiled.

Cursors

Q. How many types of cursor type are there?

A. ODBC, ADO, and DB-Library define four cursor types supported by Microsoft® SQL Server™2000. The DECLARE CURSOR statement has been extended; thus you can specify the four cursor types for Transact-SQL cursors.

The four API server cursor types supported by SQL Server are:

Static cursors

Dynamic cursors

Forward-only cursors

Keyset-driven cursors

Static cursors detect few or no changes but consume relatively few resources while scrolling, although they store the entire cursor in tempdb. Dynamic cursors detect all changes but consume more resources while scrolling, although they make the lightest use of tempdb. Keyset-driven cursors lie in between, detecting most changes but at less expense than dynamic cursors.

Although the database API cursor models consider a forward-only cursor to be a distinct type of cursor, SQL Server does not. SQL Server considers both forward only and scroll as options that can be applied to static, keyset-driven, and dynamic cursors.

Q. What is the difference between insensitive and scroll cursor?

A. Insensitive cursors make a temporary copy of the data to be used by the cursor. All requests to the cursor are answered from this temporary table; modifications made to base tables will not be reflected in the data returned by fetches made to this cursor. This cursor does not allow modifications. In other words, this cursor remains *insensitive* to changes to underlying tables.

In a scroll cursor, committed deletes and updates made to the underlying tables (by any users) are reflected in subsequent fetches.

Q. If a table does not have a unique index, can a cursor be opened on it?

A. Yes, but only insensitive cursors can be opened on a table which does not have a unique index.

Q. Can a cursor be updated? If yes, how you can protect which columns are updated?

A. By default, cursors are updatable. To prevent updates from occurring to any row use READ ONLY options.

To specify the columns which can be updatable, use UPDATE OF option. For e.g.

```
DECLARE MyCursor CURSOR FOR
  Select * from Emp for UPDATE OF( FirstName ).
```

If OF column_list is supplied, only the columns listed will allow modifications. If no list is supplied, all columns can be updated unless the cursor has been defined as READ ONLY.

Q. While using a cursor, how can you differentiate between a deleted row and a row that has been inserted with NULL data values?

A. Use the global variable @@FETCH_STATUS after each FETCH is made against the cursor. This is covered in greater detail in the next question.

Q. How can you know if the row fetched from cursor is still valid in underlying table?

A. A global variable, @@FETCH_STATUS, will be updated at every execution of FETCH. At a successful fetch, @@FETCH_STATUS will be set to 0. If no data was fetched because the requested cursor position exceeded the results set, -1 will be returned. If the row returned is no longer a member of the results set (if the row was deleted from the base table after the cursor was opened), @@FETCH_STATUS will return -2.

Always use this to determine the validity of the data returned from a cursor fetch prior to attempting any operation against that data.

Q. How can you find out how many rows returned in a cursor?

A. Use the @@CURSOR_ROWS function to determine how many rows have been returned.

Q. What does it mean if @@CURSOR_ROW returns a negative number?

A. Once a cursor has been opened; use the global variable @@CURSOR_ROWS to receive the number of qualifying rows in the last opened cursor. @@CURSOR_ROWS returns:

$-m$ If the cursor is being populated asynchronously. The value returned ($-m$) refers to the number of rows currently in the keyset.

n If the cursor is fully populated. The value returned (n) refers to the number of rows.

0 If no cursors have been opened or the last opened cursor has been closed or deallocated.

Q. How can you set the threshold at which SQL Server will generate keysets asynchronously?

A. Use the cursor threshold configuration option.

Q. What is the difference between Deallocate cursor and Close cursor?

A. Deallocate removes the cursor data structures. A closed cursor can be re-opened. DEALLOCATE releases all data structures associated with the cursor and remove the definition of the cursor.

Q. Define Cursor Locking.

A. The SELECT statement in a cursor definition obeys the transaction locking rules that apply to any other SELECT statement. In cursors, an additional set of scroll locks can be acquired based on the specification of a cursor concurrency level.

The transaction locks acquired by any SELECT statement are controlled by:

The transaction isolation level setting for the connection.

Any locking hints specified in the FROM clause.

These locks are held until the end of the current transaction for the SELECT statements. When SQL Server is in autocommit mode, each individual SQL statement is a transaction and the locks are freed when the statement finishes. If SQL Server is in explicit or implicit transaction mode, then the locks are held until the transaction is either committed or rolled back.

The locks generated by an independent SELECT or a cursor are always acquired when a row is retrieved. Cursors retrieve the rows at different times depending on the type of cursor:

Static cursors and Keyset-driven cursors retrieve the entire result set at the time the cursor is opened. This locks each row of the result set at open time.

Dynamic cursors (including forward-only cursors) do not retrieve rows until they are fetched. Locks are not acquired on the rows until they have been fetched.

Fast forward-only cursors vary in when they acquire their locks depending on the execution plan chosen by the query optimizer. If a dynamic plan is chosen, no locks are taken until the rows are fetched. If worktables are generated, then the rows are read into the worktable and locked at open time.

Datatypes

Q. Between Cast and Convert which function would you prefer and why?

A. Both the functions are used to convert one data type to another. CAST is based on the SQL-92 standard and is preferred over CONVERT. You may prefer to use convert, in situations where you want to convert a datetime value to a specific style.

Q. What are the new data types are introduced in SQL 2000?

A. SQL Server 2000 introduces three new data types. *bigint* is an 8-byte integer type. *sql_variant* is a type that allows the storage of data values of different data types. *table* is a type that allows applications to store results temporarily for later use. It is supported for variables, and as the return type for user-defined functions.

Q. Why it is recommended to avoid referencing a floating point column in the WHERE clause?

A. Approximate numeric data consists of numeric data preserved as accurately as the binary numbering system allows. Many floating-point values cannot be accurately represented in binary notation. For these numbers, it is necessary to store an approximation of the decimal number. Examples of these numbers are floating-point values ending in .3, .6, and .7. The IEEE 754 specification provides four rounding modes: round to nearest, round up, round down, and round toward zero. SQL Server uses round up. It is best to avoid referencing to floating-point columns in WHERE clauses. I.e. Float and Real.

[TOC](#)

Q. You have to store user responses of 'Yes' and 'No'. What kind of data type is best suited for this task?

A. The bit datatype is best suited to store Yes/No, True/False, and 1/0 values.

Q. What is the Exact Numeric data type in SQL?

A. Exact numeric data consists of numeric data with accuracy preserved to the least-significant digit. Exact numeric datatypes are different from the approximate numeric datatypes *float* and *real*, which cannot store all decimal numbers with complete accuracy. The *decimal* and *numeric* datatypes are identical in the SQL Server. Both are provided for ANSI compatibility.

These are the exact numeric datatypes:

decimal[(p[, s])] and *numeric*[(p[, s])]

Q. Explain timestamp datatype?

A. The timestamp is a datatype that is automatically updated every time a row containing a timestamp column is inserted or updated. Values in timestamp columns are not datetime data, but binary(8) varbinary(8) data, indicating the sequence of SQL Server activity on the row. A table can have only one timestamp column.

The timestamp datatype has no relation to the system time — it is simply a monotonically increasing counter whose values will always be unique within a database.

[TOC](#)**Q. How can a user-defined datatype be created?**

A. `sp_addtype` creates a user-defined datatype. Executing `sp_addtype` creates a user-defined datatype and adds it to the `systypes` system table. Once a user datatype is created, you can use it in the `CREATE TABLE` and `ALTER TABLE` statements, as well as bind defaults and rules to it.

```
sp_addtype typename, phystype [, nulltype]
```

Define each user datatype in terms of one of the physical (SQL Server – supplied) datatypes, preferably specifying `NULL` (allow null entries) or `NOT NULL` (disallow them). A user-defined datatype name must be unique in the database, but user-defined datatypes with different names can have the same definition.

Examples**A. Datatype That Does Not Allow Null Values**

This example creates a user-defined datatype named `ssn` to be used for columns that hold social security numbers.

Notice that `varchar(11)` is enclosed in single quotation marks because it contains punctuation (parentheses).

```
sp_addtype ssn, 'VARCHAR(11)', 'NOT NULL'
```

Q. What are the approximate numeric data types?

A. Float and Real are approximate number data types for use with floating point numeric data. Floating point data is approximate; not all values in the data type range can be precisely represented.

Q. You are creating an application where Users are asked their gender. In the gender combo box you have three options: 'Male' , 'Female' and 'I choose not to disclose'. These options are stored in the table as 1, 0 or NULL. Which datatype should you use?

A. Specify the bit data type for the column.

Q. Which data types generate inaccurate results if used with an = or <> comparison in a WHERE clause of a SQL statement?

A. Float, Real. The float and real data types are known as approximate data types. The behavior of float and real follows the IEEE 754 specification on approximate numeric data types.

Approximate numeric data types do not store the exact values specified for many numbers; they store an extremely close approximation of the value. For many applications, the tiny difference between the specified value and the stored approximation is not noticeable. At times, though, the difference becomes noticeable. Because of this approximate nature of the float and real data types, do not use these data types when exact numeric behavior is required, such as in financial applications, in operations involving rounding, or in equality checks. Instead, use the integer, decimal, money, or smallmoney data types.

Avoid using float or real columns in WHERE clause search conditions, especially the = and <> operators. It is best to limit float and real columns to > or < comparisons.

Q. Beginning with SQL Server Version 7.0, a new enhanced data type nchar was added. What type of data is supported with this data type?

A. Unicode Data. Unicode data is stored using the nchar, nvarchar, and ntext data types in SQL Server. Use these data types for columns that store characters from more than one character set.

Q. What will happen if a column containing char type data is changed to the nchar data type?

A. The nchar data type takes double the amount of bytes to save the same amount of information. Unicode data can be saved in this column. Unicode data is stored using the nchar, nvarchar, and ntext data types in SQL Server. Use these data types for columns that store characters from more than one character set. Unicode characters store the character set and the character; as a result it requires twice the space as compared to char datatype.

Q. To automatically record the time on which the data was modified in a table, which data type should you choose for the column?

A. Use the datetime datatype. Date and time data types are used for representing date and time of day. Date and time data from January 1, 1753, to December 31, 9999, to an accuracy of one three-hundredth second, or 3.33 milliseconds. Values are rounded to increments of .000, .003, or .007 milliseconds, as shown in the table.

timestamp - A database-wide unique number. The storage size is 8 bytes.

The value of a timestamp column is unique within a database.

A table can have only one timestamp column. The value in the timestamp column is updated every time a row containing a timestamp column is inserted or updated. This property makes a timestamp column a poor candidate for keys, especially primary keys. Any update made to the row changes the timestamp value, thereby changing the key value. If the column is in a primary key, the old key value is no longer valid, and foreign keys referencing the old value are no longer valid.

XML in SQL

Q. What is XDR?

A. In Microsoft® SQL Server™ 2000, the XML-Data Reduced (XDR) language is used to create the schemas. The XDR is flexible and overcomes some of the limitations of the Document Type Definition (DTD), which also describes the document structure. Unlike DTDs, XDR schemas describe the structure of the document using the same syntax as the XML document. Additionally, in a DTD, all the data contents are character data. XDR language schemas allow you to specify the data type of an element or an attribute.

Q. What is the difference between FOR AUTO and FOR NESTED?

A. The NESTED mode of the client-side FOR XML is similar to the AUTO mode of the server-side FOR XML with the following exceptions:

When you query views by using AUTO mode on the server-side, the view name is returned as the element name in the resulting XML. For example, assume that the following view is created on the **Employees** table in the **Northwind** database:

```
CREATE VIEW EmpView AS (SELECT EmployeeID as EID,
                        FirstName as FName,
                        LastName as LName
                        FROM Employees)
```

The following template specifies a query against the **EmpView** view and also specifies server-side XML formatting:

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <sql: query client-side-xml="0">
    SELECT * FROM EmpView FOR XML AUTO
  </sql: query>
</ROOT>
```

When you execute the template, the following XML is returned. (Only partial results are shown.) Note that the element names are the names of the views against which the query is executed.

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <EmpView EID="1" FName="Nancy" LName="Smith" />
  <EmpView EID="2" FName="Andrew" LName="Fuller" />
  ...
</ROOT>
```

AUTO + CLIENT = Element name will be VIEW NAME

When you specify client-side XML formatting by using the corresponding NESTED mode, the base table name(s) are returned as the element name(s) in the resulting XML. For example, the following revised template executes the same SELECT statement, but the XML formatting is performed on the client-side (that is, **client-side-xml** is set to True in the template):

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <sql:query client-side-xml="1">
    SELECT * FROM EmpView FOR XML NESTED
  </sql:query>
</ROOT>
```

Executing this template produces the following XML. Note that the element name is the base table name in this case.

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <Employees EID="1" FName="Nancy" LName="Smith" />
  <Employees EID="2" FName="Andrew" LName="Fuller" />
  ...
</ROOT>
```

NESTED + CLIENT = Element name will be TableName

When you use AUTO mode of the server-side FOR XML, the table aliases that are specified in the query are returned as element names in the resulting XML. For example, consider this template:

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <sql:query client-side-xml="0">
    SELECT FirstName as fname,
           LastName as lname
    FROM   Employees E
    FOR XML AUTO
  </sql:query>
</ROOT>
```

Executing the template produces the following XML:

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <E fname="Nancy" lname="Smith" />
  <E fname="Andrew" lname="Fuller" />
  ...
</ROOT>
```

Hint: AUTO + SERVER = Table Alias

When you use the NESTED mode of the client-side FOR XML, the table names are returned as element names in the resulting XML. (Table aliases that are specified in the query are not used.) For example, consider this template:

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <sql:query client-side-xml="1">
    SELECT FirstName as fname,
           LastName as lname
    FROM   Employees E
    FOR XML NESTED
  </sql:query>
</ROOT>
```

Executing the template produces the following XML:

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <Employees fname="Nancy" lname="Smith" />
  <Employees fname="Andrew" lname="Fuller" />
  ...
</ROOT>
```

Hint: NESTED + SERVER = TableName

Q. What is the difference between FOR XML RAW and FOR XML AUTO?

A. RAW: Takes the query result and transforms each row in the result set into an XML element with a generic identifier <row /> as the element tag.

For e.g.

```
SELECT AppID, CommonName FROM Application for XML RAW
```

```
<row AppID="6" CommonName="Information services Intranet systems"/>
<row AppID="127" CommonName="Pulp Sales System"/>
<row AppID="153" CommonName="Pulp Dryer System"/>
<row AppID="154" CommonName="Chip Systems"/>
<row AppID="155" CommonName="Time Collection System"/>
```

AUTO: Returns query results in a simple, nested XML tree. Each table in the FROM clause is represented as an XML element. The columns listed in the SELECT clause are mapped to the appropriate element attributes.

```
<ApplicationAppID="6" CommonName="Information services Intranet
systems"/>
<Application AppID="127" CommonName="Pulp Sales System"/>
<Application AppID="153" CommonName="Pulp Dryer System"/>
<Application AppID="154" CommonName="Chip Systems"/>
<Application AppID="155" CommonName="Time Collection System"/>
```

Some more differences are apparent with few example.

```
SELECT Customers.CustomerID, Orders.OrderID, Customers.ContactName
FROM Customers, Orders
WHERE Customers.CustomerID = Orders.CustomerID
FOR XML RAW
```

```
<row CustomerID="ALFKI" OrderID="10643" ContactName="Maria Anders"/>
<row CustomerID="ALFKI" OrderID="10692" ContactName="Maria Anders"/>
<row CustomerID="ALFKI" OrderID="10702" ContactName="Maria Anders"/>
<row CustomerID="ALFKI" OrderID="10835" ContactName="Maria Anders"/>
<row CustomerID="ALFKI" OrderID="10952" ContactName="Maria Anders"/>
<row CustomerID="ALFKI" OrderID="11011" ContactName="Maria Anders"/>
```

```
SELECT Customers.CustomerID, Orders.OrderID, Customers.ContactName
FROM Customers, Orders
WHERE Customers.CustomerID = Orders.CustomerID
FOR XML AUTO
```

```
<Customers CustomerID="ALFKI" ContactName="Maria Anders">
  <Orders OrderID="10643"/>
  <Orders OrderID="10692"/>
  <Orders OrderID="10702"/>
  <Orders OrderID="10835"/>
```



```

    <Orders OrderID="10952"/>
</Customers>

```

The hierarchy in the result set is based on the order of tables identified by the columns specified in the SELECT clause; therefore, the order in which column names are specified in the SELECT clause is significant.

The tables are identified and nested in the order in which the column names are listed in the SELECT clause. The first, leftmost table identified forms the top element in the resulting XML document. The second leftmost table (identified by columns in the SELECT statement) forms a subelement within the top element, and so on.

If query specifies the ELEMENT option, then an element-centric document is returned.

```

SELECT Customers.CustomerID, Orders.OrderID, Customers.ContactName
FROM Customers, Orders
WHERE Customers.CustomerID = Orders.CustomerID
FOR XML AUTO, ELEMENTS

```

```

<Customers>
  <CustomerID>ALFKI</CustomerID>
  <ContactName>Maria Anders</ContactName>
    <Orders><OrderID>10643</OrderID></Orders>
    <Orders><OrderID>10692</OrderID></Orders>
    <Orders><OrderID>10702</OrderID></Orders>
    <Orders><OrderID>10835</OrderID></Orders>
    <Orders><OrderID>10952</OrderID></Orders>
    <Orders><OrderID>11011</OrderID></Orders>
</Customers>

```

When a column in the SELECT clause cannot be associated with any of the tables identified in the FROM clause (say a computed column), the column is added in the XML document in the deepest nesting level in place when it is encountered in the list. If such a column appears as the first column in the SELECT clause, the column is added to the top element.

```

SELECT top 2 Customers.CustomerID, Orders.OrderID,
Customers.ContactName, 23* 3 as ComputedColumn
FROM Customers, Orders
WHERE Customers.CustomerID = Orders.CustomerID
FOR XML AUTO, ELEMENTS

```

```

<Customers>
  <CustomerID>ALFKI</CustomerID>
  <ContactName>Maria Anders</ContactName>
  <Orders>
    <OrderID>10643</OrderID>
    <ComputedColumn>69</ComputedColumn>
  </Orders>
  <Orders>
    <OrderID>10692</OrderID>
    <ComputedColumn>69</ComputedColumn>
  </Orders>
</Customers>

```


Q.Explain FOR XML EXPLICIT Mode?

A. In an EXPLICIT mode, the query writer controls shape of the XML document returned by the execution of the query. The query must be written in a specific way so that the additional information about the expected nesting is explicitly specified as part of the query.

See SQL Server Books On-Line for a detail. You may be asked to write a query which uses XML EXPLICIT mode.

Q. Using the Customer, and Order table in Northwind database, Please write a query to produce following XML?

```
<Results>
  <Data>
    <row id="ALFKI" name="Maria Anders" orderid="10643"/>
    <row id="ALFKI" name="Maria Anders" orderid="10692"/>
    <row id="ALFKI" name="Maria Anders" orderid="10702"/>
    <row id="ALFKI" name="Maria Anders" orderid="10835"/>
    <row id="ALFKI" name="Maria Anders" orderid="10952"/>
  </Data>
</Results>
```

A.

```
SELECT '<Results><Data>'
```

```
SELECT 1 AS Tag,
       NULL AS Parent,
       dbo.Customers.CustomerID AS [row!1!id],
       dbo.Customers.ContactName AS [row!1!name],
       dbo.Orders.OrderID AS [row!1!orderid]
FROM   dbo.Customers, dbo.Orders
WHERE  dbo.Customers.CustomerID = dbo.Orders.CustomerID
FOR XML EXPLICIT
```

```
SELECT '</Data></Results>'
```

Q. What is XML Schema?

A. An XML schema describes the structure of an XML document and also describes the various constraints on the data in the document. When you specify XPath queries against the schema, the structure of the XML document returned is determined by the schema against which the XPath query is executed.

In an XSD schema, the `<xsd:schema>` element encloses the entire schema; all element declarations must be contained within the `<xsd:schema>` element. You can describe attributes that define the namespace in which the schema resides and the namespaces that are used in the schema as properties of the `<xsd:schema>` element.

The minimum XSD schema is:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:sql="urn:schemas-microsoft-com:mapping-
schema">
  ...
</xsd:schema>
```

Q. What is mapping schema?

A. A mapping schema is, in effect, an XML view of the relational data. These mappings can be used to retrieve relational data as an XML document. In the context of the relational database, it is useful to map the arbitrary XSD schema to a relational store. One way to achieve this is to annotate the XSD schema. An XSD schema with the annotations is referred to as a *mapping schema*, which provides information pertaining to how XML data is to be mapped to the relational store.

Q. You have a table 'customers' , which has three fields, Address, CustomerID and ContactName. You would like to retrieve rows as follows:

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <Customer CustomerID="ALFKI" ContactName="Maria Anders">
    <Address>Obere Str. 57</Address>
  </Customer>
</ROOT>
```

To do get the rows return in the above format you wrote a XDR Schema as follows:

```
<?xml version="1.0" ?>
<Schema xmlns="urn:schemas-microsoft-com:xml-data"
  xmlns:dt="urn:schemas-microsoft-com:datatypes"
  xmlns:sql="urn:schemas-microsoft-com:xml-sql">

  <ElementType name="Address" />
  <ElementType name="Customer" sql:relation="Customers" >
    <AttributeType name="CustomerID" />
    <AttributeType name="ContactName" />

    <attribute type="CustomerID" />
    <attribute type="ContactName" />
    <element type="Address" />
  </ElementType>
</Schema>
```

You did not get the desired result. What is wrong with the above XDR Schema?

A. The content attribute is not specified on the <Address> subelement. Without the content=textOnly attribute, the <Address> element does not map to the address column in the Customers table because, by default, elements map to a table and not to a field. So the correct schema will be:

```
<?xml version="1.0" ?>
<Schema xmlns="urn:schemas-microsoft-com:xml-data"
  xmlns:dt="urn:schemas-microsoft-com:datatypes"
  xmlns:sql="urn:schemas-microsoft-com:xml-sql">

  <ElementType name="Address" content="textOnly" />
  <ElementType name="Customer" sql:relation="Customers" >
    <AttributeType name="CustomerID" />
    <AttributeType name="ContactName" />

    <attribute type="CustomerID" />
    <attribute type="ContactName" />
    <element type="Address" />
  </ElementType>
</Schema>
```

As an alternative, instead of specifying content=textOnly attribute, you can specify sql:field annotation to map the <Address> subelement to the Address column:

```
<element type="Address" sql:field="Address" >
```

Q. What is XPath?

A. XPath (XML Path Language) is a graph navigation language. XPath is used to select a set of nodes from an XML document. Each XPath operator selects a node-set based on a node-set selected by a previous XPath operator.

XPath language is defined by the W3C as a standard navigation language. The XPath language specification, XML Path Language (XPath) version 1.0 W3C Proposed Recommendation 8 October 1999, can be found at the W3C Web site at <http://www.w3.org/TR/1999/PR-xpath-19991008.html>. A subset of this specification is implemented in SQL Server 2000.

Q. What keyword you will use to get schema appended to the result set of a 'for XML' query?

A.

```
SELECT TOP 2 Customers.CustomerID
      FROM Customers
      FOR XML RAW
```

Query will produce following result

```
<row CustomerID="ALFKI" />
<row CustomerID="ANATR" />
```

To get the schema appended to the result set, write the query as follows:

```
SELECT TOP 2 Customers.CustomerID
      FROM Customers
      FOR XML RAW, XMLDATA
```

```
<Schema name="Schema5" xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <ElementType name="row" content="empty" model="closed">
    <AttributeType name="CustomerID" dt:type="string" />
    <attribute type="CustomerID" />
  </ElementType>
</Schema>
```

```
<row xmlns="x-schema:#Schema5" CustomerID="ALFKI" />
<row xmlns="x-schema:#Schema5" CustomerID="ANATR" />
```

Q. If I execute this query

```
SELECT Customers.CustomerID, Orders.OrderID, Customers.ContactName
FROM Customers, Orders
WHERE Customers.CustomerID = Orders.CustomerID
FOR XML AUTO
```

Following result set is returned:

```
<Customers CustomerID="ALFKI" ContactName="Maria Anders">
  <Orders OrderID="10643"/>
  <Orders OrderID="10692"/>
  <Orders OrderID="10702"/>
  <Orders OrderID="10835"/>
  <Orders OrderID="10952"/>
  <Orders OrderID="11011"/>
</Customers>
```

How can I rewrite this query to get all attributes as an element?

A. If ELEMENT key word is used, the query will return an element centric result set.

```
SELECT Customers.CustomerID, Orders.OrderID, Customers.ContactName
FROM Customers, Orders
WHERE Customers.CustomerID = Orders.CustomerID
FOR XML AUTO, ELEMENTS
```

```
<Customers>
  <CustomerID>ALFKI</CustomerID>
  <ContactName>Maria Anders</ContactName>
    <Orders><OrderID>10643</OrderID></Orders>
    <Orders><OrderID>10692</OrderID></Orders>
    <Orders><OrderID>10702</OrderID></Orders>
    <Orders><OrderID>10835</OrderID></Orders>
    <Orders><OrderID>10952</OrderID></Orders>
    <Orders><OrderID>11011</OrderID></Orders>
</Customers>
```


Clustering

Q. Explain Active/Passive and Active/Active cluster configurations?

A. Active/Passive cluster configuration is a combination of two nodes, one called a primary node and another called the secondary node. The primary node supports the clients and the secondary node acts as a failover backup node. An Active/Active cluster configuration is two simultaneous mirrored active/passive configurations.

[TOC](#)

SQL Server TID BITS

SQL server provide a feature, that support sorting by a column, which is not included in the select list.

E.g.

```
SELECT au_lname, au_fname FROM authors ORDER BY city
```

For outer joins, you should consider the *= operator obsolete and always use OUTER JOIN syntax. the *= operator might be dropped entirely in future releases of SQL Server.

- Always qualify the owner of an object when you reference it, it is always a good idea. You will see a slight performance gain if you fully qualify the stored procedure name.
- Clustered tables keep their data rows in order based on the clustered index key.
- Avoid using auto-grow on transaction logs because it is easy for a transaction log to grow uncontrolled from a bad application.
- On a PC or a laptop autogrow should be restrict to leave 100- 50 MB of the hard disk space.
- A file can be member of *only one* filegroup.
- Log files are not considered to be part of any file group because they are managed separately.
- A column that contains a limited number of unique values is a good candidate for a Clustered Index.
- No members are assigned to the Application role. The role is activated when the user runs the application. The user inherits the permission assigned to the application role.
- Neither system processes nor processes running extended stored procedures can be killed.
- Members of the fixed server role sysadmin and those who access an SQL server via the sa login account are mapped to the user account dbo in all databases. The dbo account is the owner of any objects created by these users.
- Members of the fixed database role db_owner can perform any tasks for the database in which they have been assigned to this role. However, any objects they create will be owned by their user account, not by the user account dbo. Members of the role db_owner can designate dbo as the owner of objects they create. This is a good practice because users can then refer to the objects without including the name of the owner in the object name.
- The default database for a new login ID created in SQL Enterprise Manager will be the master database if no default database is designated. The master database tracks information about SQL Server in general and about each user database. It should not be the default database for users, so it is a good practice to specify a default database for each user or group.
- When no explicit mappings are defined between an SQL server and a linked server, the security credentials of the current login are used. Thus, the current user's Login ID and password will be passed to the linked server.
- To access any SQL services on the linked server, the user must have a valid Login ID on that server or belong to a Windows NT group for which a Login ID has been defined.
- The Login account SQLAgentCmdExec is used to provide the security context for jobs that are owned by accounts that are not members of the sysadmin role.
- The smallest value for the FILEGROWTH parameter is one extent. One extent contains eight pages and each page is 8 KB in size.

- A database extent consists of eight 8 KB pages. Different objects can now share an extent or an object can have its own extent. A table and index both have a minimum of two pages.
- The extended stored procedure `xp_trace_setqueryhistory` is used to enable the query history.
- Those who belong to the `db_ddladmin` role can manage database objects.
- The Transact-SQL statement `sp_changegroup` can be used to add a database User ID to a database group. Since a User ID may belong only to one group other than public, `sp_changegroup` also removes the user from any group other than public to which he or she is assigned.
- The Transact-SQL statement `sp_addgroup` allows a new group to be defined but cannot be used to add Database User IDs to the group.
- The Transact-SQL statement `sp_addlogin` can be used to create a Login ID and its associated password as well as define the default database and default language for the Login ID. It cannot be used to define the members of a database group.
- The Transact-SQL statement `sp_adduser` can be used to assign a database User ID to a Login ID and to indicate to which group the User ID should be assigned. This is used to define a new database User ID, not to reassign an existing User ID to another group.
- SQL 7.0 onward uses roles instead of groups, but the stored procedures for group management are provided for backward compatibility.
- Only the current database owner or a member of the `sysadmin` role can transfer ownership of a database to another user. The new owner must have a valid login account and must NOT have a user account for the database.
- A database may be taken offline with the stored procedure `sp_dboption`. The Options tab of the Database Properties dialog box in SQL Enterprise Manager does not provide this capability. This option is often used with databases that are on removable media.
- The database options “`dbo use only`”, “`select into / bulk copy`” and “`trunc. log on chkpt`” may all be set on the Options tab of the Database Properties dialog box in SQL Enterprise Manager.
- Setting a database to “`dbo use only`” allows active users to continue using the database, but no new users other than anyone aliased as the DBO are allowed.
- Setting the “`select into / bulk copy`” option turns off logging. This should be set when doing a fast bulk copy (bcp).
- Setting the “`trunc. log on chkpt`” option causes all committed transactions to be removed from the transaction log file whenever a checkpoint is done. This prevents the log from filling up, but also eliminates the capability of doing an incremental backup.
- Members of the fixed server role `securityadmin` can manage login accounts for an SQL server.
- Members of the fixed server role `serveradmin` can manage server configuration parameters.
- Members of the fixed database role `db_accessadmin` can manage users and groups for a database.
- The built-in account `sa` is a login account, not a role.
- The default password for the `sa` login account is blank (no password). Always change this to a secure password when SQL authentication is enabled.
- Two databases may contain tables with the same name. A single database may also contain tables with the ‘same name’ as long as the owner is different.

- With Windows ONLY NT Authentication, the user's Windows NT username is used as that individual's login ID for SQL. If the user's login ID has been defined as an sa account, the user will have sa privileges on the server. The sa login itself is not used if Windows NT Authentication ONLY is configured, so it is not necessary to assign a password to the sa account. A password could be assigned to the sa login ID, but it will be ignored with Windows NT Authentication.

[TOC](#)

- A**
- a parameterized view, 35
 - alternate, 15
 - application role, 39
 - approximate numeric data, 120
 - Automatic Recovery, 82
- B**
- Boyce/Codd normal form, 27
- C**
- candidate, 14
 - Cast, 118
 - clustered, 60
 - clustered index, 61
 - clustering, 64
 - Clustering, 133
 - COALESCE, 31
 - concurrency control, 51
 - Concurrency Control, 104
 - constraints, 55
 - Convert, 118
 - correlated subquery, 21
 - COUNT, 31
 - cross join, 54
 - Cube, 37
 - Cursor Locking, 117
 - CURSOR_ROW, 116
 - Cursors, 115
- D**
- Datatypes, 118
 - DBCC INDEXDEFRAG, 64
 - DBCC REINDEX, 64
 - deadlock, 50, 51
 - denormalization, 27
 - denormalize, 40
 - derived table, 31
 - derived Table, 74
 - DROPCLEANBUFFERS, 29
 - Dynamic cursors, 115
- E**
- Entity-Relationship Diagram, 19
 - ERD, 19
 - Exact Numeric data type, 118
 - extended property, 33
 - Extended property, 33
 - extended stored procedures, 33
- F**
- fill factor, 63, 86
 - First Normal form, 24
 - FOR AUTO, 123
 - FOR NESTED, 123
 - FOR XML AUTO, 125
 - FOR XML EXPLICIT, 127
 - FOR XML RAW, 125
 - Forward-only cursors, 115
 - Fourth Normal Form, 27
 - FREEPROCCACHE, 29
 - Full-text indexes, 65
- I**
- IDENTITY, 48
 - image, 34
 - image columns, 71
 - index, 60
 - Information Schema Views, 75
 - insensitive, 115
 - INSTEAD OF triggers, 108
 - Isolation levels, 104
- J**
- Joins, 52
- K**
- Keyset-driven cursors, 115
- L**
- linked server, 134
 - livelock, 50
 - Lock escalation, 106
 - Locks, 106
 - low fill factor, 64
- M**
- many-to-many, 19, 33
 - Many-to-Many Relationships**, 20
 - mapping schema, 128
 - master database, 134
 - Merge Replication, 101
 - mixed extent, 84
 - model database, 82
 - Multiple Nonclustered Indexes**, 61

- N**
- non-clustered, 60
 - Non-Clustered Index, 61
 - normalize, 23
- O**
- one-to-many, 19
 - One-to-Many Relationships**, 20
 - one-to-one, 19
 - One-to-One Relationships**, 20
 - Optimistic Concurrency Control, 104
 - outer join, 53
- P**
- page level lock, 106
 - parameterized views, 35
 - Pessimistic Concurrency Control, 104
 - PINTABLE, 49
 - primary, 16
 - primary key, 16
- R**
- RAID, 47
 - RAID5, 83
 - RAISEERROR, 35
 - READ COMMITTED, 104
 - READ UNCOMMITTED, 104
 - referential integrity, 18
 - REPEATABLE READ, 104
 - Replication, 100
 - Rollup, 36
 - row lock, 106
 - ROWGUIDCOL, 48
- S**
- scroll, 115
 - Second Normal Form, 25
 - Security, 102
 - SERIALIZABLE, 104
 - sp_changeobjectowner, 80
 - Static cursors, 115
 - statistics, 64, 87
- T**
- Third Normal Form, 26
 - timestamp, 119
 - Transactions, 104
 - Triggers, 107
- U**
- unique, 17
 - user-defined function, 34
- V**
- view, 102
 - View, 71
- W**
- What is a subquery, 21
 - WRITETEXT, 107
- X**
- XDR, 122
 - XML Schema, 128
 - XPath, 130