

Intelligent Data Management with SQL Server

Trainer Guide

For Aptech Centre Use Only

Intelligent Data Management with SQL Server

Trainer Guide

© 2021 Aptech Limited

All rights reserved.

No part of this book may be reproduced or copied in any form or by any means – graphic, electronic or mechanical, including photocopying, recording, taping, or storing in information retrieval system or sent or transferred without the prior written permission of copyright owner Aptech Limited.

All trademarks acknowledged.

APTECH LIMITED

Contact E-mail: ov-support@onlinevarsity.com

Edition I - 2021



The book **Intelligent Data Management with SQL Server Trainer Guide** begins with an introduction to RDBMS concepts and moves on to introduce SQL Server 2019. The book then covers various SQL Server topics such as data types, usage of Transact-SQL, and database objects such as indexes, stored procedures, functions, and so on. The book also introduces Azure SQL and cloud databases. The book describes transactions, programming elements with Transact-SQL, and finally troubleshooting errors with error handling techniques.

The book also explores SQL Server 2019 new features and enhancements. These include features such as Big Data clusters, PolyBase, Query Store, Stretch Database, and In-Memory enhancements. Besides these, you will also learn about the improved Performance Tools and Transact-SQL enhancements.

The faculty/trainer should teach the concepts in the theory class using the slides. This Trainer's Guide will provide guidance on the flow of the module and also provide tips and additional examples wherever necessary. The trainer can ask questions to make the session interactive and also to test the understanding of the students.

The knowledge and information in this book is the result of the concentrated effort of the Design Team, which is continuously striving to bring to you the latest, the best and the most relevant subject matter in Information Technology. As a part of Aptech's quality drive, this team does intensive research and curriculum enrichment to keep it in line with industry trends and learner requirements.

Contents

1. RDBMS Concepts
2. Entity-Relationship (E-R) Model and Normalization
3. Introduction to SQL Server 2019
4. Transact-SQL
5. Creating and Managing Databases
6. Creating Tables
7. Azure SQL
8. Accessing Data
9. Advanced Queries and Joins
10. Views, Stored Procedures, and Querying Metadata
11. Indexes
12. Triggers
13. Programming Transact-SQL
14. Transactions
15. Error Handling
16. Enhancements in SQL Server 2019
17. PolyBase, Query Store, and Stretch Database

Session 1: RDBMS Concepts

1.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. Prepare a question or two that will be a key point to relate the current session objectives.

1.1.1 Teaching Skills

To teach this session, you should be well versed with the concepts related to database, database management systems, RDBMS, entities, and tables.

You should teach the concepts in the theory class using the images provided. For teaching in the class, you are expected to use slides and LCD projectors.

In-Class Activities

Follow the order given here during In-Class activities.

Slide 2

Objectives

- Explain the concept of data and database
- Describe the approaches to data management
- Define a Database Management System (DBMS) and list its benefits
- Explain the different database models
- Define and explain RDBMS
- Describe entities and tables and list the characteristics of tables
- List the differences between a DBMS and an RDBMS

© Aptech Ltd. Session 1 / 2

Instructions to the Trainer(s):

Give students a brief overview of the current session through the session objectives listed in Slide 2.

1.2 In-Class Explanations

Slide 3

Introduction

- Organizations often maintain large amounts of data, generated as a result of day-to-day operations.
- A database:
 - is an organized form of such data.
 - may consist of one or more related data items called records.
 - is a data collection to which different questions can be asked.
- For example,
 - 'What are phone numbers and addresses of five nearest post offices?' or
 - 'Do we have any books in our library that deal with health food?'



© Aptech Ltd. Session 1/ 3

Instructions to the Trainer(s):

- Introduce the session. Using Slide 3, explain the necessity of database and its management.
- Organizations often maintain large amounts of data, which are generated as a result of day-to-day operations.
- A database is an organized form of such data. It may consist of one or more related data items called records.
- Think of a database as a data collection to which different questions can be asked. For example, 'What are the phone numbers and addresses of the five nearest post offices?' or 'Do we have any books in our library that deal with health food? If so, on which shelves are they located?' or 'Show me the personnel records and sales figures of five best-performing sales people for the current quarter, but their address details do not require to be shown'.

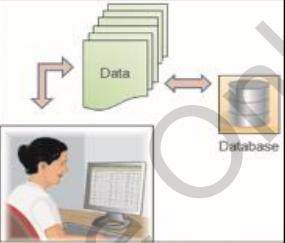
Data and Database

When data is gathered and analyzed, it yields information.
Intelligent interpretation of data yields information.

Information helps to foresee and plan events.

A database is an organized collection of data such that its contents can be easily accessed, managed, and updated.

Example:
A phone book is a database consisting of names, addresses, and telephone numbers.



© Aptech Ltd.

Session 1 / 4

Instructions to the Trainer(s):

- Using Slide 4, explain the data and database concept to students.
- Data means information and it is the most important component in any work that is done. In day-to-day activities, either existing data is used or more data is generated. When this data is gathered and analyzed, it yields information. It can be any information such as information about the vehicle, sports, airways, and so on. For example, a sport magazine journalist (who is a soccer enthusiast) gathers the score (data) of Germany's performance in 10 world cup matches. These scores constitute data. When this data is compared with the data of 10 world cup matches played by Brazil, the journalist can obtain information as to which country has a team that plays better soccer.
- Information helps to foresee and plan events. Intelligent interpretation of data yields information. In the world of business, to be able to predict an event and plan for it could save time and money. Consider an example, where a car manufacturing company is planning its annual purchase of certain parts of the car, which has to be imported since it is not locally available. If data of the purchase of these parts for the last five years is available, the company heads can actually compile information about the total amount of parts imported. Based on these findings, a production plan can be prepared. Therefore, information is a key-planning factor.
- A database is a collection of data. Some like to think of a database as an organized mechanism that has the capability of storing information. This information can be retrieved by the user in an effective and efficient manner.
- A phone book is a database. The data contained consists of individuals' names, addresses, and telephone numbers. These listings are in alphabetical order or indexed. This allows the user to reference a particular local resident with ease. Ultimately, this data is stored in a database somewhere on a computer. As people move to different cities or states, entries may have to

be added or removed from the phone book. Likewise, entries will have to be modified for people changing names, addresses, or telephone numbers, and so on.

In-Class Question:

Question: What is a database?

Answer: A database is a collection of data.

Slide 5

Data Management

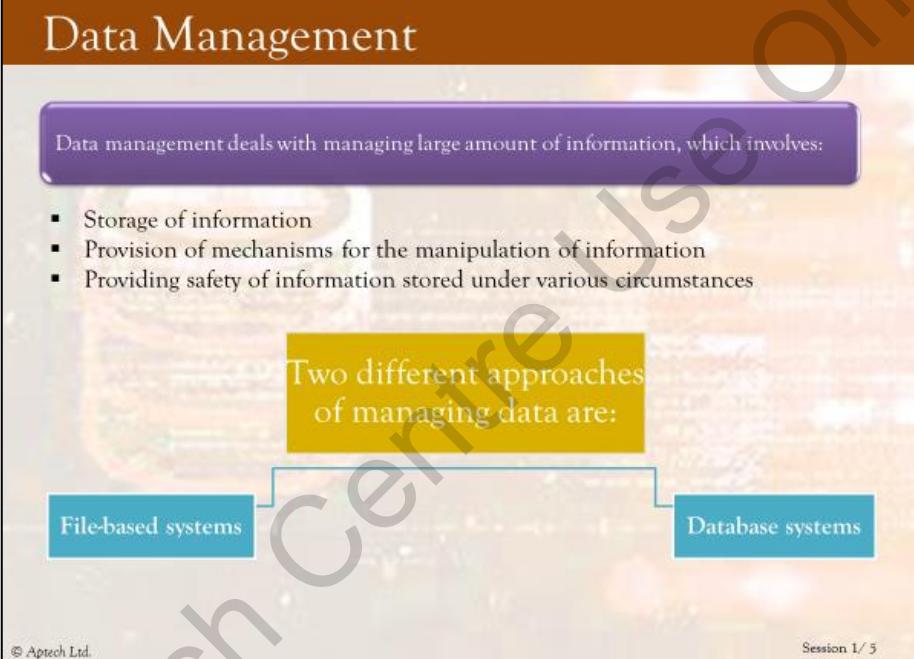
Data management deals with managing large amount of information, which involves:

- Storage of information
- Provision of mechanisms for the manipulation of information
- Providing safety of information stored under various circumstances

Two different approaches of managing data are:

File-based systems Database systems

© Aptech Ltd. Session 1 / 5



Instructions to the Trainer(s):

- Using Slide 5, explain data management.
- Data management deals with managing large amount of information, which involves both the storage of information and the provision of mechanisms for the manipulation of information.
- In addition, the system should also provide the safety of the information stored under various circumstances, such as multiple user access and so on.
- Two different approaches of managing data are as follows:
 - File-based systems
 - Database systems

File-based Systems

In a file-based system, data is stored in discrete files and a collection of such files is stored on a computer

Files of archived data were called tables

Rows in the table were called **records** and columns were called **fields**

Example:

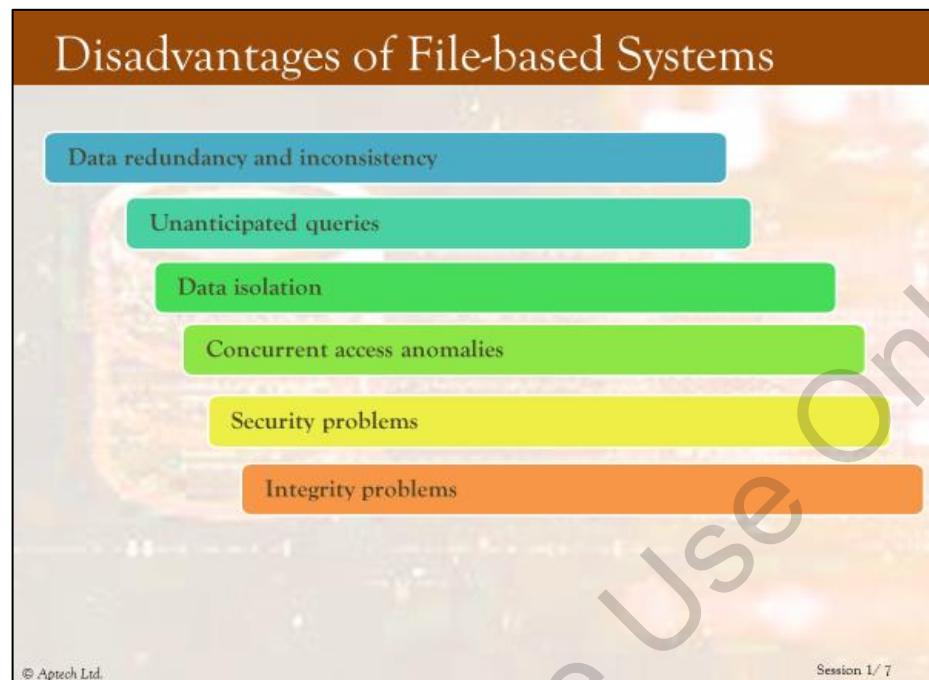
First Name	Last Name	Address	Phone
Eric	David	ericd@eff.org	2134560987
Selena	Sol	selena@eff.org	987-765-4321
Jordan	Lim	nadroj@otherdomain.com	2223456123

© Aptech Ltd.

Session 1/ 6

Instructions to the Trainer(s):

- Using Slide 6, explain File-based systems.
- Tell the students that storage of large amounts of data has always been a matter of huge concern. In early days, file-based systems were used. In this system, data was stored in discrete files and a collection of such files was stored on a computer. These could be accessed by a computer operator.
- Files of archived data were called tables because they looked like tables used in traditional file keeping. Rows in the table were called records and columns were called fields.
- Conventionally, before the database systems evolved, data in software systems was stored in flat files.

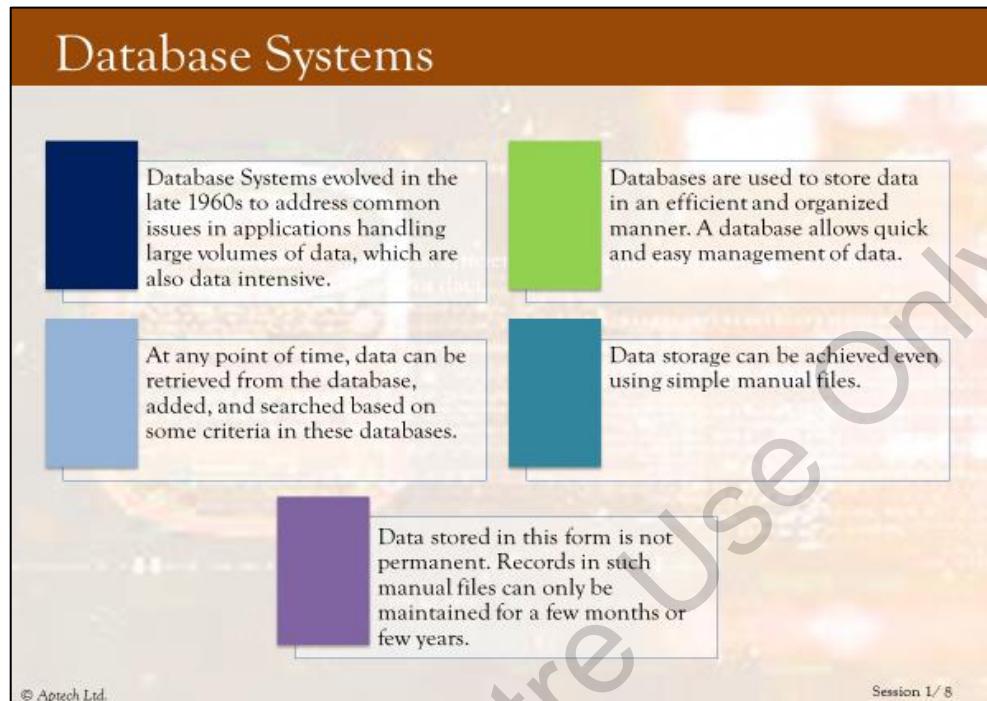


Instructions to the Trainer(s):

- Using Slide 7, explain the disadvantages of file-based systems in detail.
- In a file-based system, different programs in the same application may be interacting with different private data files. There is no system enforcing any standardized control on the organization and structure of these data files.
 - **Data redundancy and inconsistency**
 - Since data resides in different private data files, there are chances of redundancy and resulting inconsistency. For example, a customer can have a savings account as well as a mortgage loan. Here, the customer details may be duplicated since the programs for the two functions store their corresponding data in two different data files. This gives rise to redundancy in the customer's data. Since the same data is stored in two files, inconsistency arises if a change made in the data of one file is not reflected in the other.
 - **Unanticipated queries**
 - In a file-based system, handling sudden/ad-hoc queries can be difficult, since it requires changes in the existing programs. For example, the bank officer must generate a list of all the customers who have an account balance of \$20,000 or more. The bank officer has two choices: either obtain the list of all customers and have the required information extracted manually or hire a system programmer to design the necessary application program. Both alternatives are obviously unsatisfactory. Suppose that such a program is written and several days later, the officer must trim that list to include only those customers who have opened their account one year ago.

As the program to generate such a list does not exist, it leads to a difficulty in accessing the data.

- **Data isolation**
 - Data are scattered in various files and files may be in a different format. Though data used by different programs in the application may be related, they reside as isolated data files.
- **Concurrent access anomalies**
 - In large multi-user systems, the same file or record may must be accessed by multiple users simultaneously. Handling this in a file-based system is difficult.
- **Security problems**
 - In data-intensive applications, security of data is a major concern. Users should be given access only to required data and not to the whole database.
- **Integrity problems**
 - In any application, there will be certain data integrity rules, which must be maintained. These could be in the form of certain conditions/constraints on the elements of the data records. In the savings bank application, one such integrity rule could be 'Customer ID, which is the unique identifier for a customer record, should not be empty'. There can be several such integrity rules. In a file-based system, all these rules must be explicitly programmed in the application program.



The slide has a brown header bar with the title "Database Systems". Below the header, there are five colored callout boxes containing text. A large, faint watermark reading "For Sample Centre Use Only" is visible across the slide.

- Blue box:** Database Systems evolved in the late 1960s to address common issues in applications handling large volumes of data, which are also data intensive.
- Green box:** Databases are used to store data in an efficient and organized manner. A database allows quick and easy management of data.
- Blue box:** At any point of time, data can be retrieved from the database, added, and searched based on some criteria in these databases.
- Teal box:** Data storage can be achieved even using simple manual files.
- Purple box:** Data stored in this form is not permanent. Records in such manual files can only be maintained for a few months or few years.

© Aptech Ltd. Session 1/ 8

Instructions to the Trainer(s):

- Using Slide 8, explain the Database system.
- Database Systems evolved in the late 1960s to address common issues in applications handling large volumes of data, which are also data intensive.
- Databases are used to store data in an efficient and organized manner.
- A database allows quick and easy management of data. For example, a company may maintain details of its employees in various databases.
- Data storage can be achieved even using simple manual files. For instance, a college has to maintain information about teachers, students, subjects, and examinations.
- Details of the teachers can be maintained in a Staff Register and details of the students could be entered in a Student Register and so forth. However, data stored in this form is not permanent. Records in such manual files can only be maintained for a few months or few years. The registers or files are bulky, consume a lot of space and hence, cannot be kept for many years.
- Instead of this, if the same data was stored using database system, it could be more permanent and long-lasting.

Advantages of Database Systems

- Amount of redundancy in the stored data can be reduced
- No more inconsistencies in data
- Stored data can be shared
- Standards can be set and followed
- Data Integrity can be maintained
- Security of data can be implemented

© Aptech Ltd. Session 1 / 9

Instructions to the Trainer(s):

- Using Slide 9, explain the advantages of Database systems.
- Tell the students that information or data can be permanently stored in the form of computerized databases. A database system is advantageous because it provides a centralized control over the data.
- Some of the benefits of using such a centralized database system are as follows:
 - **The amount of redundancy in the stored data can be reduced**
 - In an organization, several departments often store the same data. Maintaining a centralized database helps the same data to be accessed by many departments. Thus, duplication of data or 'data redundancy' can be reduced.
 - **No more inconsistencies in data**
 - When data is duplicated across several departments, any modifications to the data have to be reflected across all departments. Sometimes, this can lead to inconsistency in the data. As there is a central database, it is possible for one person to take up the task of updating the data on a regular basis.
 - **The stored data can be shared**
 - A central database can be located on a server, which can be shared by several users. In this way, all users can access the common and updated information all the time.

- **Standards can be set and followed**
 - A central control ensures that a certain standard in the representation of data can be set and followed. For example, the name of an employee has to be represented as 'Mr. Larry Finner'. This representation can be broken down into following components:
 - A title (Mr.)
 - First name (Larry)
 - Last name (Finner)
- **Data Integrity can be maintained**
 - Data integrity refers to the accuracy of data in the database. For example, when an employee resigns and leaves the organization, consider that the Accounts department has updated its database and the HR department has not updated its records. The data in the company's records is therefore inaccurate. Centralized control of the database helps in avoiding these errors.
- **Security of data can be implemented**
 - In a central database system, the privilege of modifying the database is not given to everyone. This right is given only to one person who has full control over the database. This person is called as Database Administrator or DBA. The DBA can implement security by placing restrictions on the data. Based on the permissions granted to them, the users can add, modify, or query data.

Database Management System (DBMS) 1-3

A DBMS is a collection of related records and a set of programs that access and manipulate these records and enables the user to enter, store, and manage data.

In a centralized database system, the database is stored in the central location which everybody can have access from their machine.

A database is a collection of interrelated data and a DBMS is a set of programs used to add or modify this data.

Examples of database applications include:



© Aptech Ltd.

Session 1/ 10

Database Management System (DBMS) 2-3

A DBMS is a collection of related records and a set of programs that access and manipulate these records and enables the user to enter, store, and manage data.

Example

In a centralized database system, the database is stored in the central location which everybody can have access from their machine.

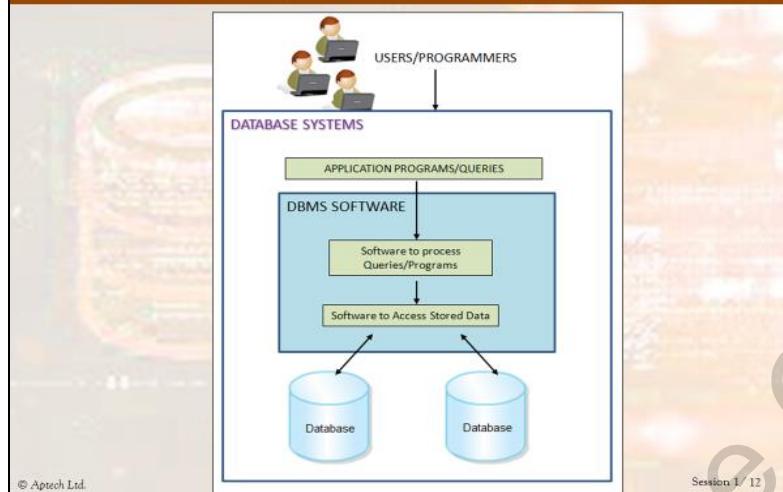


A database is a collection of interrelated data and a DBMS is a set of programs used to add or modify this data.

© Aptech Ltd.

Session 1/ 11

Database Management System (DBMS) 3-3



Instructions to the Trainer(s):

- Using Slides 10 to 12, explain the DBMS concept.
- A DBMS can be defined as a collection of related records and a set of programs that access and manipulate these records.
- A DBMS enables the user to enter, store, and manage data.
- The main problem with the earlier DBMS packages was that the data was stored in the flat file format. So, the information about different objects was maintained separately in different physical files.

In-Class Question:

Question: What is a DBMS?

Answer: A DBMS can be defined as a collection of related records and a set of programs that access and manipulate these records.

- Using Slide 11, explain that a DBMS provides an environment that is both convenient and efficient to use when there is a large volume of data and many transactions to be processed.
- Different categories of DBMS can be used, ranging from small systems that run on personal computers to huge systems that run on mainframes.
- Examples of database applications include the following:
 - Computerized library systems
 - Automated teller machines
 - Flight reservation systems
 - Computerized parts inventory systems

From a technical standpoint, DBMS products can differ widely. Different DBMS support different query languages, although there is a semi-standardized query language called

Structured Query Language (SQL). Sophisticated languages for managing database systems are called Fourth Generation Language (4GLs).

- Using Slide 12, tell students about the working of Database system.
- A DBMS enables the user to enter, store, and manage data.
- The main problem with the earlier DBMS packages was that the data was stored in the flat file format. So, the information about different objects was maintained separately in different physical files.
- Hence, the relations between these objects, if any, had to be maintained in a separate physical file.
- Thus, a single package would consist of too many files and vast functionalities to integrate them into a single system.

Slide 13

Benefits of DBMS 1-2

A DBMS is responsible for processing data and converting it into information.

A database for this purpose has to be manipulated, which includes querying the database to retrieve specific data, updating the database, and finally, generating reports.

These reports are the source of information, which is, processed data.

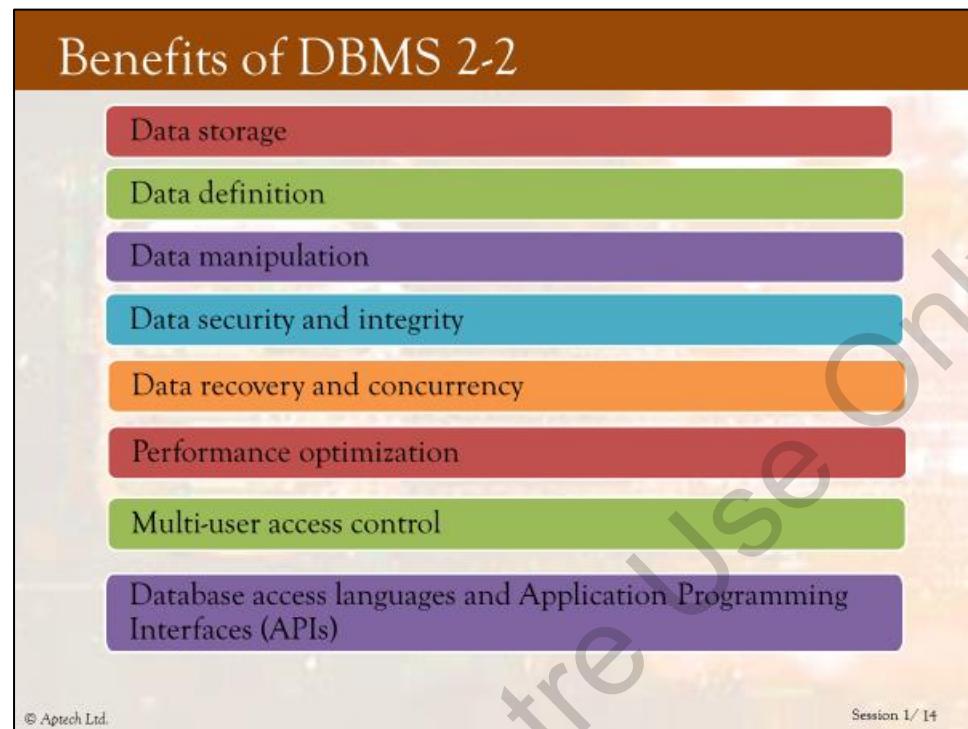
A DBMS is also responsible for data security and integrity.

© Aptech Ltd.

Session 1/ 13

Instructions to the Trainer(s):

- Using Slide 13, explain students the benefits of DBMS.
- A DBMS is responsible for processing data and converting it into information.
- For this purpose, the database has to be manipulated, which includes querying the database to retrieve specific data, updating the database, and finally, generating reports.
- These reports are the source of information, which is, processed data.
- A DBMS is also responsible for data security and integrity.



Instructions to the Trainer(s):

- Using Slide 14, tell students that they will understand different benefits of DBMS.
- The benefits of DBMS are explained as follows:
 - **Data storage:** The programs required for physically storing data, handled by a DBMS, is done by creating complex data structures, and the process is called data storage management.
 - **Data definition:** A DBMS provides functions to define the structure of the data in the application. These include defining and modifying the record structure, the type and size of fields, and various constraints/conditions to be satisfied by the data in each field.
 - **Data manipulation:** Once the data structure is defined, data must be inserted, modified, or deleted. The functions, which perform these operations, are also part of a DBMS. These functions can handle planned and unplanned data manipulation requirements.
 - **Data security and integrity:** Data security is of utmost importance when there are multiple users accessing the database. It is required for keeping a check over data access by users. Data in the database should contain as few errors as possible. For example, the employee number for adding a new employee should not be left blank. Telephone number should contain only numbers. Such checks are taken care of by a DBMS. Thus, the DBMS contains functions, which handle the security and integrity of data in the application. These can be easily invoked by the application and hence, the application programmer does not require to code these functions in the programs.

- **Data recovery and concurrency:** Recovery of data after a system failure and concurrent access of records by multiple users are also handled by a DBMS.
- **Performance:** Optimizing the performance of the queries is one of the important functions of a DBMS.
- **Multi-user access control:** At any point of time, more than one user can access the same data. A DBMS takes care of the sharing of data among multiple users, and maintains data integrity.
- **Database access languages and Application Programming Interfaces (APIs):** The query language of a DBMS implements data access. SQL is the most commonly used query language. A query language is a non-procedural language, where the user must request what is required and does not have to specify how it is to be done. Some procedural languages such as C, Visual Basic, Pascal, and others provide data access to programmers.

Slide 15

Database Models

Databases can be differentiated based on functions and model of the data.

A data model describes a container for storing data, and the process of storing and retrieving data from that container.

Analysis and design of data models has been the basis of the evolution of databases.

Each model has evolved from the previous one.
Commonly used Database Models are:

© Aptech Ltd. Session 1 / 15

Instructions to the Trainer(s):

- Using Slide 15, explain the Database Model.
- Databases can be differentiated based on functions and model of the data.
- A data model describes a container for storing data, and the process of storing and retrieving data from that container.
- The analysis and design of data models has been the basis of the evolution of databases. Each model has evolved from the previous one.
- Also, there are object-oriented model and object relational model.

Flat-file Data Model

In this model, the database consists of only one table or file

Is used for simple databases - for example, to store the roll numbers, names, subjects, and marks of a group of students

Cannot handle very complex data. It can cause redundancy when data is repeated more than once

➤ Following table depicts structure of a flat file database:

Roll Number	First Name	Last Name	Subject	Marks
45	Jones	Bill	Maths	84
45	Jones	Bill	Science	75
50	Mary	Mathew	Science	80

Instructions to the Trainer(s):

- Using Slide 16, explain the flat-file data model.
- In this model, the database consists of only one table or file. This model is used for simple databases - for example, to store the roll numbers, names, subjects, and marks of a group of students.
- The flat-file data model cannot handle very complex data. It can cause redundancy when data is repeated more than once.
- Tell the students that a flat-file database cannot contain multiple tables similar to a relational database.
- Most database programs such as Microsoft Access and FileMaker Pro can import flat file databases and use them in a larger relational database.

Hierarchical Data Model 1-3

In this model:

Different records are inter-related through hierarchical or tree-like structures.

Relationships are thought of in terms of children and parents.

A parent record can have several children, but a child can have only one parent.

To find data stored in this model, the user needs to know the structure of the tree.

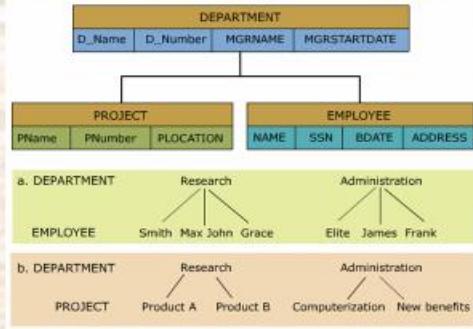
Windows Registry is an example of a hierarchical database storing configuration settings and options on Microsoft Windows operating systems.

Instructions to the Trainer(s):

- Using Slide 17, explain the Hierarchical Data Model.
- In the Hierarchical Model, different records are inter-related through hierarchical or tree-like structures.
- In this model, relationships are thought of in terms of children and parents.
- A parent record can have several children, but a child can have only one parent.
- To find data stored in this model, the user must know the structure of the tree.

Hierarchical Data Model 2-3

- Following figure illustrates an example of a hierarchical representation:



- Within the hierarchical model, Department is perceived as the parent of the segment.
- The tables, Project and Employee, are children.
- A path that traces the parent segments beginning from the left, defines the tree.
- This ordered sequencing of segments tracing the hierarchical structure is called the hierarchical path.

Instructions to the Trainer(s):

- Using Slide 18, explain the illustration of hierarchical representation.
- The Windows Registry is an example of a hierarchical database storing configuration settings and options on Microsoft Windows operating systems.
- Figure shown in slide 18 illustrates an example of a hierarchical representation.
- Within the hierarchical model, Department is perceived as the parent of the segment. The tables, Project and Employee, are children. A path that traces the parent segments beginning from the left, defines the tree.
- This ordered sequencing of segments tracing the hierarchical structure is called the hierarchical path. It is clear from the figure that in a single department, there can be many employees and a department can have many projects.

Hierarchical Data Model 3-3

- Advantages of a hierarchical model are as follows:

Data is held in a common database so data sharing becomes easier, and security is provided and enforced by a DBMS.

Data independence is provided by a DBMS, which reduces the effort and costs in maintaining the program.

This model is very efficient when a database contains a large volume of data.

- For example, a bank's customer account system fits the hierarchical model well because each customer's account is subject to a number of transactions.

Instructions to the Trainer(s):

- Using Slide 19, explain the advantages of hierarchical model.
- Advantages of a hierarchical model are as follows:
 - Data is held in a common database so data sharing becomes easier, and security is provided and enforced by a DBMS.
 - Data independence is provided by a DBMS, which reduces the effort and costs in maintaining the program.
- This model is very efficient when a database contains a large volume of data. For example, a bank's customer account system fits the hierarchical model well because each customer's account is subject to a number of transactions.

Network Data Model 1-4

This model is similar to the Hierarchical Data Model. It is actually a subset of the network model.

In the network data model, data is stored in sets, instead of the hierarchical tree format. This solves the problem of data redundancy.

For every database, a definition of the database name, record type for each record, and the components that make up those records is stored. This is called its network schema.

It allows application programs to access the required data from the database. Raima Database Manager (RDM) Server by Raima Inc. is an example of a Network DBMS.

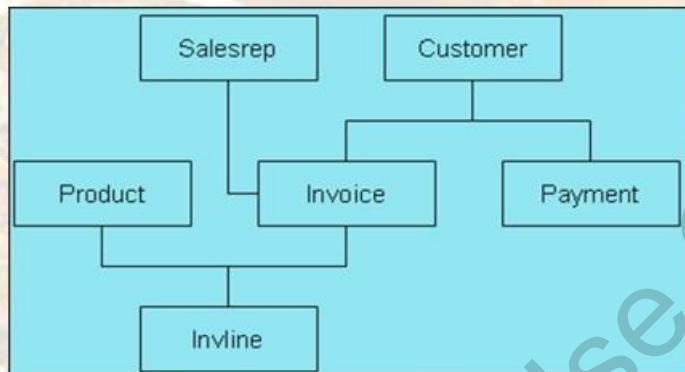
© Aptech Ltd. Session 1/ 20

Instructions to the Trainer(s):

- Using Slide 20, explain Network Data Model.
- Mention this model is similar to the Hierarchical Data Model. The hierarchical model is actually a subset of the network model.
- However, instead of using a single-parent tree hierarchy, the network model uses set theory to provide a tree-like hierarchy with the exception that child tables were allowed to have more than one parent.
- In the network model, data is stored in sets, instead of the hierarchical tree format. This solves the problem of data redundancy.
- The network model together with the hierarchical data model was a major data model for implementing numerous commercial DBMS. The network model structures and language constructs were defined by Conference on Data Systems Language (CODASYL).

Network Data Model 2-4

- Figure illustrates a series of one-to-many relationships:

**Instructions to the Trainer(s):**

- Slide 21 shows the illustration of Network Data Model. It illustrates and shows a series of one-to-many relationships as follows:
- A sales representative might have written many Invoice tickets, but each Invoice is written by a single Sales representative (Salesrep).
 - A Customer might have made purchases on different occasions. A Customer may have many Invoice tickets, but each Invoice belongs only to a single customer.
 - An Invoice ticket may have many Invoice lines (Invline), but each Invline is found on a single Invoice ticket.
 - A product may appear in several different Invline, but each Invline contains only a single Product.

Network Data Model 3-4

➤ Components of the language used with network models are:

- Data Definition Language (DDL)**
 - Used to create and remove databases and database objects. It enables the database administrator to define the schema components.
- Sub-schema DDL**
 - Enables the database administrator to define database components.
- Data Manipulation Language (DML)**
 - Used to insert, retrieve, and modify database information.
- Data Control Language (DCL)**
 - Used to administer permissions on the databases and database objects.

© Aptech Ltd. Session 1 / 22

Instructions to the Trainer(s):

- Using Slide 22, explain the components used by the network data model.
- Components of the language used with network models are as follows:
 - A Data Definition Language (DDL) that is used to create and remove databases and database objects. It enables the database administrator to define the schema components.
 - A sub-schema DDL that enables the database administrator to define the database components.
 - A Data Manipulation Language (DML), which is used to insert, retrieve, and modify database information. All database users use these commands during the routine operation of the database.
 - Data Control Language (DCL) is used to administer permissions on the databases and database objects.

Network Data Model 4-4

➤ Advantages are:

- Relationships are easier to implement in the network database model than in the hierarchical model.
- This model enforces database integrity.
- This model achieves sufficient data independence.

➤ Disadvantages are:

- The databases in this model are difficult to design.
- The programmer has to be familiar with the internal structures to access the database.
- The model provides a navigational data access environment.

➤ This model is difficult to implement and maintain.
➤ Computer programmers, rather than end users, utilize this model.

© Aptech Ltd. Session 1 / 23

Instructions to the Trainer(s):

- Using Slide 23, explain the advantages and disadvantages of network model.
- Advantages of such a structure are specified as follows:
 - The relationships are easier to implement in the network database model than in the hierarchical model.
 - This model enforces database integrity.
 - This model achieves sufficient data independence.
- Disadvantages are specified as follows:
 - The databases in this model are difficult to design.
 - The programmer has to be very familiar with the internal structures to access the database.
 - The model provides a navigational data access environment. Hence, to move from A to E in the sequence A-B-C-D-E, the user has to move through B, C, and D to get to E.

Relational Data Model 1-5

As the information needs grew and more sophisticated databases and applications were required, database design, management, and use became too cumbersome.

This led to the development of what came to be called the Relational Model database.

The term 'Relation' is derived from the set theory of mathematics. In the Relational Model, unlike the Hierarchical and Network models, there are no physical links.

All data is maintained in the form of tables consisting of rows and columns. Data in two tables is related through common columns and not physical links.

Operators are provided for operating on rows in tables. This model represents the database as a collection of relations.

Instructions to the Trainer(s):

- Using Slide 24, explain the Relational Data Model.
- The term 'Relation' is derived from the set theory of mathematics.
- As the information requirements grew and more sophisticated databases and applications were required, database design, management, and use became too cumbersome.
- The lack of query facility took a lot of time of the programmers to produce even the simplest reports.
- This led to the development of what came to be called the Relational Model database.

Relational Data Model 2-5

A row is called a tuple, a column, an attribute, and the table is called a relation.

The list of values applicable to a particular field is called domain.

Several attributes can belong to the same domain.

The number of attributes of a relation is called degree of the relation.

The number of tuples determines the cardinality of the relation.

Instructions to the Trainer(s):

- Using Slide 25, discuss the relational data model with the students.
- In the Relational Model, unlike the Hierarchical and Network models, there are no physical links.
- All data is maintained in the form of tables consisting of rows and columns.
- Data in two tables is related through common columns and not physical links.
- Operators are provided for operating on rows in tables.
- This model represents the database as a collection of relations. In this model's terminology, a row is called a tuple, a column is called an attribute, and table a relation.
- The list of values applicable to a particular field is called domain. It is possible for several attributes to have the same domain. The number of attributes of a relation is called degree of the relation.
- Some of the popular relational DBMSs are:
 - Oracle, Sybase, DB2, Microsoft SQL Server, and so on.

Relational Data Model 3-5

- In order to understand the relational model, consider **Students** and **Marks** tables:

Roll Number	Student Name
1	Sam Reiner
2	John Parkinson
3	Jenny Smith
4	Lisa Hayes
5	Penny Walker
6	Peter Jordan
7	Joe Wong

Students Table

Roll Number	Marks Obtained
1	34
2	87
3	45
4	90
5	36
6	65
7	89

Marks Table

- **Students** table displays the **Roll Number** and the **Student Name** and the **Marks** table displays the **Roll Number** and **Marks** obtained by the students.
- To locate students with marks above 40:
 - First, locate the roll numbers of those who have scored above 50 from the **Marks** table.
 - Second, their names have to be located in the **Students** table by matching the roll number.

Instructions to the Trainer(s):

- Using Slide 26, students can understand the concept of relational model in a better manner.
- In order to understand the relational model, consider tables on Slide 26.
- The Students table displays the Roll Number and the Student Name and the Marks table displays the Roll Number and Marks obtained by the students. Now, two steps must be carried out for students who have scored above 50. First, locate the roll numbers of those who have scored above 50 from the Marks table. Second, their names have to be located in the Students table by matching the roll number.

Relational Data Model 4-5

- The result is displayed as shown in the following table:

Roll Number	Student Name	Marks Obtained
2	John	87
4	Lisa	90
6	Peter	65
7	Joe	89

- It was possible to get this information because of two facts:

First, there is a column common to both the tables - Roll Number.

Second, based on this column, the records from two different tables could be matched and the required information could be obtained.

- In a relational model, data is stored in tables.
- A table in a database has a unique name that identifies its contents.
- Each table can be defined as an intersection of rows and columns.

Instructions to the Trainer(s):

- Using Slide 27, explain how formation of groups and grouping of data is done.
- It was possible to get this information because of two facts:
 - First, there is a column common to both the tables - Roll Number.
 - Second, based on this column, the records from the two different tables could be matched and the required information could be obtained.
- In a relational model, data is stored in tables.
- A table in a database has a unique name that identifies its contents. Each table can be defined as an intersection of rows and columns.

Relational Data Model 5-5

Advantages of the relational model

Gives programmer time to concentrate on logical view of the database rather than being bothered about the physical view.

Provides querying flexibility and hence, leads to popularity of relational databases.

Easy to handle model to the extent that even untrained people find it easy to generate handy reports and queries, without giving much thought to the need to design a proper database.

Disadvantages of the relational model

Hides all complexities of the system and hence, it tends to be slower than other database systems.

Instructions to the Trainer(s):

- Using Slide 28, explain advantages and disadvantages of relational model.

Advantages of the relational model

- The relational database model gives the programmer time to concentrate on the logical view of the database rather than being bothered about the physical view. One of the reasons for the popularity of the relational databases is the querying flexibility.
- Most of the relational databases use Structured Query Language (SQL). An RDBMS uses SQL to translate the user query into the technical code required to retrieve the requested data.
- Relational model is so easy to handle that even untrained people find it easy to generate handy reports and queries, without giving much thought to the requirement to design a proper database.

Disadvantages of the relational model

- Though the model hides all the complexities of the system, it tends to be slower than the other database systems.

As compared to all other models, the relational data model is the most popular and widely used.

Relational Database Management System (RDBMS) 1-3

Relational Model is an attempt to simplify database structures.

Represents all data in the database as simple row-column tables of data values.

An RDBMS is a software program that helps to create, maintain, and manipulate a relational database.

A relational database is a database divided into logical units called tables, where tables are related to one another within the database.

By having common keys, or fields, among relational database tables, data from multiple tables can be joined to form one large resultset.

© Aptech Ltd.

Session 1 / 29

Instructions to the Trainer(s):

- Using Slide 29, explain the RDBMS concept.
- The Relational Model is an attempt to simplify database structures. It represents all data in the database as simple row-column tables of data values.
- An RDBMS is a software program that helps to create, maintain, and manipulate a relational database.
- A relational database is a database divided into logical units called tables, where tables are related to one another within the database.
- Tables are related in a relational database, allowing adequate data to be retrieved in a single query (although the desired data may exist in more than one table).
- By having common keys or fields, among relational database tables, data from multiple tables can be joined to form one large resultset.

Relational Database Management System (RDBMS) 2-3

➤ Figure shows two tables related to one another through a common key (data value) in a relational database:

OrderID	Item
R001	Files
R002	Pens
R003	Pencils

OrderID	InvoiceNo	OrderDate
R001	1001	09/20/2012
R002	1002	09/12/2007
R003	1003	09/15/2009

Orders Table OrderDetails Table

➤ Thus, a relational database is a database structured on the relational model.
➤ Basic characteristic of a relational model is that in a relational model, data is stored in relations.

© Aptech Ltd. Session 1 / 30

Instructions to the Trainer(s):

- Using Slide 30, tell students about the relationships based on keys.
- The figure shows two tables related to one another through a common key (data value) in a relational database.
- Thus, a relational database is a database structured on the relational model. The basic characteristic of a relational model is that in a relational model, data is stored in relations.

Relational Database Management System (RDBMS) 3-3

➤ **Capitals** and **Currency** tables show a list of countries and their capitals and the countries and the local currencies used by them respectively:

Country	Capital
Greece	Athens
Italy	Rome
USA	Washington
China	Beijing
Japan	Tokyo
Australia	Sydney
France	Paris

Capitals Table

Country	Currency
Greece	Drachma
Italy	Lira
USA	Dollar
China	Renminbi (Yuan)
Japan	Yen
Australia	Australian Dollar
France	Francs

Currency Table

➤ Both tables have a common column, that is, the **Country** column.
 ➤ Now, to display the information about the currency used in Rome, first find the name of the country to which Rome belongs from table **Capitals**.
 ➤ Next, that country should be looked up in table **Currency** to find out the currency.
 ➤ It is possible to get this information because it is possible to establish a relation between the two tables through a common column called **Country**.

© Aptech Ltd.

Session 1 / 31

Instructions to the Trainer(s):

- Using Slide 31, capitals and currency tables are displayed for RDBMS.
- Tell the students that the Capitals table shown in the table displays a list of countries and their capitals, and the Currency table displays the countries and the local currencies used by them.
- Both the tables have a common column, that is, the Country column. Now, if the user wants to display the information about the currency used in Rome, first find the name of the country to which Rome belongs.
- This information can be retrieved from the table. Next, that country should be looked up in the table to find out the currency.
- It is possible to get this information because it is possible to establish a relation between the two tables through a common column called Country.

Terms Related to RDBMS 1-3

- Terms that are mostly used in an RDBMS are described as follows:

Data is presented as a collection of relations.

Each relation is depicted as a table.

Columns are attributes.

Rows ('tuples') represent entities.

Every table has a set of attributes that are taken together as a 'key' (technically, a 'superkey'), which uniquely identifies each entity.

Instructions to the Trainer(s):

- Using Slide 32, tell students they will learn the terms related to RDBMS.
- Following are terms related to RDBMS:
- Data is presented as a collection of relations
 - Each relation is depicted as a table
 - Columns are attributes
 - Rows represent entities
 - Every table has a set of attributes that are taken together as a 'key'

Terms Related to RDBMS 2-3

- Consider the scenario of a company maintaining customer and order information for products being sold and customer-order details for a specific month, say, August.
- Following tables are used to illustrate this scenario:

Cust_No	Cust_Name	Phone No
002	David Gordon	0231-5466356
003	Prince Fernandes	0221-5762382
003	Charles Yale	0321-8734723
002	Ryan Ford	0241-2343444
005	Bruce Smith	0241-8472198

Customer

Item_No	Description	Price
HW1	Power Supply	4000
HW2	Keyboard	2000
HW3	Mouse	800
SW1	Office Suite	15000
SW2	Payroll Software	8000

Items

Ord_No	Item_No	Qty
101	HW3	50
101	SW1	150
102	HW2	10
103	HW3	50
104	HW2	25
104	HW3	100
105	SW1	100

Order_Details

Ord_No	Ord_Date	Cust_No
101	02-08-12	002
102	11-08-12	003
103	21-08-12	003
104	28-08-12	002
105	30-08-12	005

Order_August

© Aptech Ltd.

Session 1 / 33

Instructions to the Trainer(s):

- Using Slide 33, tell students about tabular form for RDBMS.
- Consider the scenario of a company maintaining customer and order information for products being sold and customer-order details for a specific month, such as, August. The tables are used to illustrate this scenario.
- These tables depict tuples and attributes in the form of rows and columns. Various terms related to these tables are given in table.

Terms Related to RDBMS 3-3

- Following table lists terms related to tables:

Term	Meaning	Example from the Scenario
Relation	A table	Order_August, Order_Details, Customer and Items
Tuple	A row or a record in a relation	A row from Customer relation is a Customer tuple
Attribute	A field or a column in a relation	Ord_Date, Item_No, Cust_Name, and so on
Cardinality of a relation	The number of tuples in a relation	Cardinality of Order_Details relation is 7
Degree of a relation	The number of attributes in a relation	Degree of Customer relation is 3
Domain of an attribute	The set of all values that can be taken by the attribute	Domain of Qty in Order_Details is the set of all values which can represent quantity of an ordered item
Primary Key of a relation	An attribute or a combination of attributes that uniquely defines each tuple in a relation	Primary Key of Customer relation is Cust_No Ord_No and Item_No combination forms the primary key of Order_Details
Foreign Key	An attribute or a combination of attributes in one relation R1 that indicates the relationship of R1 with another relation R2. The foreign key attributes in R1 must contain values matching with those of the values in R2	Cust_No in Order_August relation is a foreign key creating reference from Order_August to Customer. This is required to indicate the relationship between orders in Order_August and Customer

© Aptech Ltd.

Session 1 / 34

Instructions to the Trainer(s):

- Using Slide 34, explain to students about how each term in RDBMS is explained with an example.
- The table shown in slide 34 explains each term in RDBMS.
- Different terms are mentioned, such as:
 - Relation
 - Tuple
 - Attribute
 - Foreign key
 - Primary key of a relation and so on

In-Class Question:

Question: What is a foreign key?

Answer: An attribute or a combination of attributes in one relation R1 that indicates the relationship of R1 with another relation R2 is termed as a foreign key.

RDBMS Users 1-2

➤ Many persons are involved in the design, use, and maintenance of a large database with a few hundred users.

Database Administrator (DBA)

- Collects information that will be stored in the database
- Responsible for authorizing access to database
- Coordinating and monitoring its use
- Acquiring software and hardware resources as needed
- Accountable for problems such as breach of security or poor system response time



Database Designer

- Responsible for identifying the data to be stored in the database
- Choosing appropriate structures to represent and store this data
- Communicate with all prospective database users, in order to understand their requirements
- To come up with a design that meets the requirements



© Aptech Ltd. Session 1 / 35

RDBMS Users 2-2

System Analysts and Application Programmers

- Determine the requirements of end users
- Develop specifications for pre-determined transactions that meet these requirements
- Implement these specifications as programs
- Test, debug, document, and maintain these pre-determined transactions
- Design, development, and operation of the DBMS software and system environment



DBMS Designers and Implementers

- Design and implement the DBMS modules and interfaces as a software package.



End User

- The end user invokes an application to interact with the system, or writes a query for easy retrieval, modification, or deletion of data.

© Aptech Ltd. Session 1 / 36

Instructions to the Trainer(s):

- Using Slides 35 and 36, explain the categories of RDBMS Users.
- The primary goal of a database system is to provide an environment for retrieving information from and storing new information into the database.

- For a small personal database, one person typically defines the constructs and manipulates the database. However, many persons are involved in the design, use, and maintenance of a large database with a few hundred users.
- For designing the database system, there are many users who perform different tasks.
- There are Database Administrators (DBAs) who collect information that is stored in the database, responsible for authorizing access, coordinating and monitoring, acquiring software and hardware resources, and so on.
- The Database designer is responsible for identifying the data stored in the database.
- The Database designer is also responsible to choose appropriate structures, store the data, communicate with the database users and understand the requirements.
- Using Slide 36, tell students that they will learn about different categories of RDBMS users.
- **System Analyst and Application programmers:**
 - Determine the requirements, develop specifications for pre-determined transactions test, debug, document, and maintain the pre-determined transactions
- **DBMS Designers and Implementers:**
 - Database Design is a collection of processes that facilitate the designing, development, implementation, and maintenance of DBMS.
- **End Users:**
 - They invoke the application to interact with the system or write a query for easy retrieval of data.

Entity

An entity is a person, place, thing, object, event, or even a concept, which can be distinctly identified.

Each entity has certain characteristics known as attributes.

A grouping of related entities becomes an entity set. Each entity set is given a name. The name of the entity set reflects the contents.

© Aptech Ltd.

Session 1 / 37

Instructions to the Trainer(s):

- Using Slide 37, explain to students that they will understand the concept of entity in RDBMS.
- An entity is a person, place, thing, object, event, or even a concept, which can be distinctly identified. For example, the entities in a university are students, faculty members, and courses.
- Each entity has certain characteristics known as attributes.
- For example, the student entity might include attributes such as student number, name, and grade. Each attribute should be named appropriately.
- A grouping of related entities becomes an entity set. Each entity set is given a name. The name of the entity set reflects the contents.
- Thus, the attributes of all the students of the university will be stored in an entity set called Student.

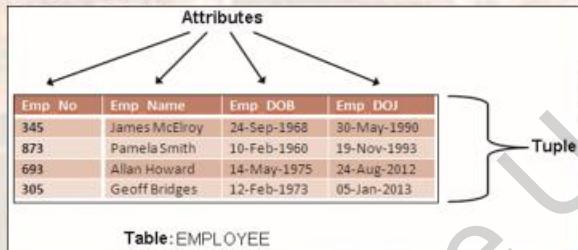
Tables and their Characteristics 1-2

Access and manipulation of data is facilitated by creation of data relationships based on a construct known as a table.

A table contains a group of related entities that is an entity set. The terms entity set and table are often used interchangeably.

A table is also called a relation. The rows are known as tuples. The columns are known as attributes.

- Following figure highlights characteristics of a table:



Tables and their Characteristics 2-2

- Characteristics of a table:

A two-dimensional structure composed of rows and columns is perceived as a table.

Each tuple represents a single entity within the entity set.

Each column has a distinct name.

Each row/column intersection represents a single data value.

Each table must have a key known as primary key that uniquely identifies each row.

All values in a column must conform to the same data format.

Each column has a specific range of values known as the attribute domain.

Each row carries information describing one entity occurrence.

The order of the rows and columns is immaterial in a DBMS.

Instructions to the Trainer(s):

- Using Slides 38 and 39, explain about tables and their characteristics.
- A table contains a group of related entities that is an entity set. The terms entity set and table are often used interchangeably.
- A table is also called a relation.

- The rows are known as tuples.
 - The columns are known as attributes.
- Using Slide 39, discuss the characteristics of table in RDBMS.
- Tell the students that the access and manipulation of data is facilitated by the creation of data relationships based on a construct known as a table.
- Characteristics of tables are:
- A two-dimensional structure composed of rows and columns is perceived in a table.
 - Each tuple represents a single entity within the entity set.
 - Each row/column intersection represents a single data value.
 - Order of rows and columns is immaterial.
 - Each column has a distinct name.
 - All values in a table must conform to the same data format.

Slide 40

Differences between a DBMS and an RDBMS	
DBMS	RDBMS
It does not need to have data in tabular structure nor does it enforce tabular relationships between data items.	In an RDBMS, tabular structure is a must and table relationships are enforced by the system. These relationships enable the user to apply and manage business rules with minimal coding.
Small amount of data can be stored and retrieved.	An RDBMS can store and retrieve large amount of data.
A DBMS is less secure than an RDBMS.	An RDBMS is more secure than a DBMS.
It is a single user system.	It is a multiuser system.
Most DBMSs do not support client/server architecture.	It supports client/server architecture.
Here, entities are given more importance and there is no relation established among these entities.	Here, a relation is given more importance. Thus, the tables in an RDBMS are dependent and the user can establish various integrity constraints on these tables so that the ultimate data used by the user remains correct.

© Aptech Ltd.

Session 1 / 40

Instructions to the Trainer(s):

- Using Slide 40, explain the difference between DBMS and RDBMS.
- Tell the students that in an RDBMS, a relation is given more importance. Thus, the tables in an RDBMS are dependent and the user can establish various integrity constraints on these tables so that the ultimate data used by the user remains correct.
- In case of a DBMS, entities are given more importance and there is no relation established among these entities.

Summary

- A database is a collection of related data stored in the form of a table.
- A data model describes a container for storing data and the process of storing and retrieving data from that container.
- A DBMS is a collection of programs that enables the user to store, modify, and extract information from a database.
- A Relational Database Management System (RDBMS) is a suite of software programs for creating, maintaining, modifying, and manipulating a relational database.
- A relational database is divided into logical units called tables. These logical units are interrelated to each other within the database.
- The main components of an RDBMS are entities and tables.
- In an RDBMS, a relation is given more importance, whereas, in case of a DBMS, entities are given more importance and there is no relation established among these entities.

Instructions to the Trainer(s):

- Show students Slide 41.
- Summarize the session by reading out each point on the slide.

Session 2: Entity-Relationships (E-R) Model and Normalization

2.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. Prepare a question or two that will be a key point to relate the current session objectives.

2.1.1 Teaching Skills

To teach this session, you should be well versed with concept about Data Modeling, Entity-Relationships (E-R) Model, ER Diagram, and Normalization.

You should teach the concepts in the theory class using the images provided. For teaching in the class, you are expected to use slides and LCD projectors.

In-Class Activities

Follow the order given here during In-Class activities.

Slide 2

Objectives

- Define and describe data modeling
- Identify and describe the components of the E-R model
- Identify relationships that can be formed between entities
- Explain E-R diagrams and their use
- Describe an E-R diagram, the symbols used for drawing, and show various relationships
- Describe various Normal Forms
- Outline uses of different Relational Operators

© Aptech Ltd. Session 2 / 2

Instructions to the Trainer(s):

Give students a brief overview of the current session through the session objectives listed in Slide 2.

2.2 In-Class Explanations

Slide 3

Introduction

② A data model is a group of conceptual tools that describes data, its relationships, and semantics.

② Some of the Data model examples are as follows:

- Entity-Relationship
- Relational
- Network
- Hierarchical models

Conceptual Data Modeling Elements

```
graph LR; BUM[Business Use-Case Model] --> BA[Business Actors]; UCM[Use-Case Model] --> SA[System Actors]; BAM[Business Analysis Model] --> BEC[Business Entity Classes]; BAM --> BWC[Business Worker Classes]; AM[Analysis Model] --> EC[Entity Classes]
```

© Aptech Ltd.

Session 2/3

Instructions to the Trainer(s):

- Using Slide 3, explain the necessity of data model.
- A data model is a group of conceptual tools that describes data, its relationships, and semantics. It also consists of the consistency constraints that the data adheres to.
- Examples of data models are as follows:
 - The Entity-Relationship
 - Relational
 - Network and
 - Hierarchical models
- The development of every database begins with the basic step of analyzing its data in order to determine the data model that would best represent it. Once this step is completed, the data model is applied to the data.

Data Modeling

The process of applying an appropriate data model to the data, in order to organize and structure it.

Its as essential as planning and designing are to any project.

Building a database without a data model is similar to developing a project without its plans and design.

Helps to define:

Relational tables, Primary and Foreign keys, Stored procedures and triggers

© Aptech Ltd.

Session 2/ 4

Instructions to the Trainer(s):

- Using Slide 4, explain the data modeling.
- The process of applying an appropriate data model to the data, in order to organize and structure it, is called data modeling.
- Data modeling is as essential to database development as are planning and designing to any project development. Building a database without a data model is similar to developing a project without its plans and design.
- Data models help database developers to define
 - The relational tables
 - Primary and foreign keys
 - Stored procedures
 - Triggers required in the database
- Data modeling can be broken down into following three broad steps:
 - **Conceptual Data Modeling:** The data modeler identifies the highest level of relationships in the data.
 - **Logical Data Modeling:** The data modeler describes the data and its relationships in detail. The data modeler creates a logical model of the database.
 - **Physical Data Modeling:** The data modeler specifies how the logical model is to be realized physically.

In-Class Question:

Question: What is data modeling?

Answer: The process of applying an appropriate data model to the data, in order to organize and structure it, is called data modeling.

Entity-Relationship (E-R) Model 1-5

Data models can be classified into three different groups:

Object-based logical models Record-based logical models Physical models

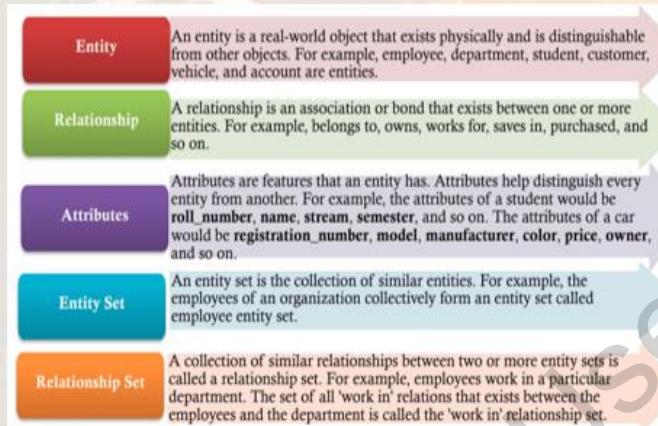
© Aptech Ltd. Session 2/5

Instructions to the Trainer(s):

- Using Slide 5, explain the E-R model.
- Entity-Relationship (E-R) model belongs to the first classification. Data can be perceived as real-world objects called entities and the relationships that exist between them.
- Data models can be classified into three different groups:
 - Object-based logical models
 - Record-based logical models
 - Physical models

Entity-Relationship (E-R) Model 2-5

□ Components of an E-R model:



© Aptech Ltd.

Session 2/ 6

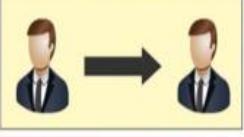
Instructions to the Trainer(s):

- Using Slide 6, explain the components of the E-R model.
- An E-R model consists of five basic components. They are as follows:
 - **Entity:** An entity is a real-world object that exists physically and is distinguishable from other objects. For example, employee, department, student, customer, vehicle, and account are entities.
 - **Relationship:** A relationship is an association or bond that exists between one or more entities. For example, belongs to, owns, works for, saves in, purchased, and so on.
 - **Attributes:** Attributes are features that an entity has. Attributes help distinguish every entity from another. For example, the attributes of a student would be roll_number, name, stream, semester, and so on. The attributes of a car would be registration_number, model, manufacturer, color, price, owner, and so on.
 - **Entity Set:** An entity set is the collection of similar entities. For example, the employees of an organization collectively form an entity set called employee entity set.
 - **Relationship Set:** A collection of similar relationships between two or more entity sets is called a relationship set. For example, employees work in a particular department. The set of all 'work in' relations that exists between the employees and the department is called the 'work in' relationship set.

Entity-Relationship (E-R) Model 3-5

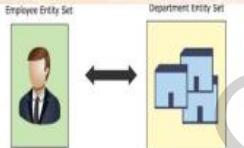
Self-relationships:

- Relationships between entities of the same entity set are called self-relationships.
- For example, a manager and his team member, both belong to the employee entity set.



Binary relationships:

- Relationships that exist between entities of two different entity sets are called binary relationships.
- For example, an employee belongs to a department.



© Aptech Ltd.

Session 2/7

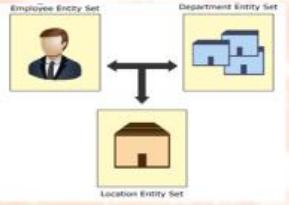
Instructions to the Trainer(s):

- Using Slide 7, explain the concept of relationship and its types.
- Relationships associate one or more entities and can be of three types. They are as follows:
 - **Self-relationships:** Relationships between entities of the same entity set are called self-relationships. For example, a manager and his team member, both belong to the employee entity set. The team member works for the manager. Thus, the relation, 'works for', exists between two different employee entities of the same employee entity set.
 - **Binary relationships:** Relationships that exist between entities of two different entity sets are called binary relationships. For example, an employee belongs to a department. The relation exists between two different entities, which belong to two different entity sets. The employee entity belongs to an employee entity set. The department entity belongs to a department entity set.

Entity-Relationship (E-R) Model 4-5

☰ Ternary relationships:

- Relationships that exist between three entities of different entity sets are called ternary relationships.
- For example, an employee works in the accounts department at the regional branch.



Relationships can also be mapped as:

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

© Aptech Ltd.

Session 2/8

Instructions to the Trainer(s):

- Using Slide 8, explain the concept of relationship and its types.
 - **Ternary relationships:** Relationships that exist between three entities of different entity sets are called ternary relationships. For example, an employee works in the accounts department at the regional branch. The relation, 'works' exists between all three, the employee, the department, and the location.
- Relationships can also be mapped as -
 - **One-to-One:** This kind of mapping exists when an entity of one entity set can be associated with only one entity of another set.
 - **One-to-Many:** This kind of mapping exists when an entity of one set can be associated with more than one entity of another entity set.
 - **Many-to-One:** This kind of mapping exists when many entities of one set is associated with an entity of another set.
 - **Many-to-Many:** This kind of mapping exists when any number of entities of one set can be associated with any number of entities of the other entity set.

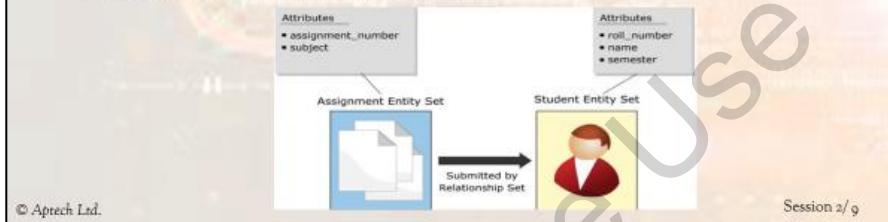
Entity-Relationship (E-R) Model 5-5

Some additional concepts in the E-R model are as follows:

- **Primary Keys:** A primary key is an attribute that can uniquely define an entity in an entity set.

Enrollment_Number	Name	Grade	Division
786	Ashley	Seven	B
957	Joseph	Five	A
1011	Kelly	One	A

- **Weak entity sets:** Entity sets that do not have enough attributes to establish a primary key are called weak entity sets.
- **Strong entity sets:** Entity sets that have enough attributes to establish a primary key are called strong entity sets.



Instructions to the Trainer(s):

- Using Slide 9, explain to the students the keys used in E-R model.
- Some additional concepts in the E-R model are as follows:
 - **Primary keys:** A primary key is an attribute that can uniquely define an entity in an entity set.
 - **Weak entity sets:** Entity sets that do not have enough attributes to establish a primary key are called weak entity sets.

Mention that weak entity set depends on some other entities primary key.

- **Strong entity sets:** Entity sets that have enough attributes to establish a primary key are called strong entity sets.

Entity-Relationship Diagrams 1-3

The E-R diagram is a graphical representation of the E-R model. The E-R diagram, with the help of various symbols, effectively represents various components of the E-R model.

Component	Symbol	Example
Entity	Entity	Student
Weak Entity	Weak Entity	Assignments
Attribute	Attribute	Roll_num
Relationship	Relationship	Saves in
Key Attribute	Attribute	Acct_num

E-R Diagram Symbols

© Aptech Ltd.

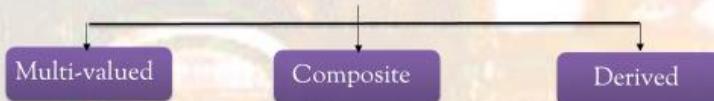
Session 2/10

Instructions to the Trainer(s):

- Using Slide 10, explain the Entity-Relationship Diagram.
- The E-R diagram is a graphical representation of the E-R model.
- The E-R diagram, with the help of various symbols, effectively represents various components of the E-R model.
- Different components are as follows:
 - Entity
 - Weak Entity
 - Attribute
 - Relationship
 - Key Attribute

Entity-Relationship Diagrams 2-3

Attributes in the E-R model can be further classified as follows:



Steps to construct an E-R diagram are as follows:

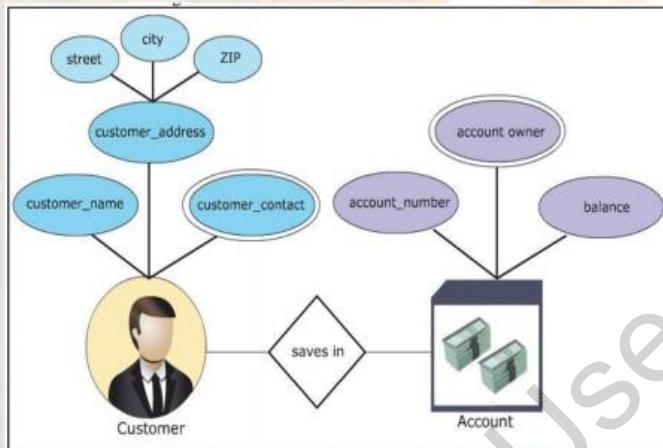
1. Gather all the data that must be modeled.
2. Identify data that can be modeled as real-world entities.
3. Identify the attributes for each entity.
4. Sort entity sets as weak or strong entity sets.
5. Sort entity attributes as key attributes, multi-valued attributes, composite attributes, derived attributes, and so on.
6. Identify the relations between the different entities.
7. Using different symbols draw the entities, their attributes, and their relationships. Use appropriate symbols while drawing attributes.

Instructions to the Trainer(s):

- Using Slide 11, explain the attributes in E-R Diagram.
- Attributes in the E-R model can be further classified as follows:
 - **Multi-valued:** A multi-valued attribute is illustrated with a double-line ellipse, which has more than one value for at least one instance of its entity. This attribute may have upper and lower bounds specified for any individual entity value. The telephone attribute of an individual may have one or more values, that is, an individual can have one or more telephone numbers. Hence, the telephone attribute is a multi-valued attribute.
 - **Composite:** A composite attribute may itself contain two or more attributes, which represent basic attributes having independent meanings of their own. The address attribute is usually a composite attribute, composed of attributes such as street, area, and so on.
 - **Derived:** Derived attributes are those whose value is entirely dependent on another attribute and are indicated by dashed ellipses. The age attribute of a person is the best example for derived attributes. For a particular person entity, the age of a person can be determined from the current date and the person's birth date.
- Steps of construction are:
 - Gather data
 - Identify entities
 - Identify the attributes
 - Sort entity sets
 - Sort attributes
 - Identify relations

Entity-Relationship Diagrams 3-3

An example of the E-R Diagram



© Aptech Ltd.

Session 2/12

Instructions to the Trainer(s):

- Using Slide 12, students can view the diagrammatic representation of E-R Diagram.
- Using different symbols draw the entities, their attributes, and their relationships. Use appropriate symbols while drawing attributes.
- Explain the scenario of a bank, with customers and accounts and tell the students to identify the entities, relationships, and attributes.

Normalization

Databases are characterized by large number of columns and records

Emp_No	Project_Id	Project_Name	Emp_Name	Grade	Salary
142	113, 124	BLUE STAR, MAGNUM	John	A	20,000
168	113	BLUE STAR	James	B	15,000
263	113	BLUE STAR	Andrew	C	10,000
109	124	MAGNUM	Bob	C	10,000

Department Employee Details

It consists of the following anomalies:

- Repetition anomaly
- Insertion anomaly
- Deletion anomaly
- Deletion anomaly

© Aptech Ltd.

Session 2/ 13

Instructions to the Trainer(s):

- Using Slide 13, explain the concept of Normalization.
- Normalization minimizes redundancy from a relation or set of relations.
- All databases are characterized by large number of columns and records. This approach has certain drawbacks.
- Normalization consists of following anomalies:
 - Repetition anomaly
 - Insertion anomaly
 - Deletion anomaly
- Various anomalies which lead to normalization are as under:
 - **Repetition anomaly:** The data such as Project_id, Project_name, Grade, and Salary repeat many times. This repetition hampers both, performance during retrieval of data and the storage capacity. This repetition of data is called the repetition anomaly. The repetition is shown in the table with the help of shaded cells.
 - **Insertion anomaly:** Suppose the department recruits a new employee named Ann. Now, consider that Ann has not been assigned any project. Insertion of her details in the table would leave columns Project_id and Project_name empty. Leaving columns blank could lead to problems later. Anomalies created by such insertions are called insertion anomalies. The anomaly can be seen in table.
 - **Deletion anomaly:** Suppose, Bob is relieved from the project MAGNUM. Deleting the record deletes Bob's Emp_no, Grade, and Salary details too. This loss of data is harmful as all of Bob's personal details are also lost as seen in the table. This kind of loss of data due to deletion is called deletion anomaly. The anomaly can be seen in the table.

- **Updating anomaly:** Suppose John was given a hike in Salary or John was demoted. The change in John's Salary or Grade must be reflected in all projects John works for. This problem in updating all the occurrences is called updating anomaly.
- Normalization is the process of removing unwanted redundancy and dependencies. Initially, Codd (1972) presented three normal forms (1NF, 2NF, and 3NF), all based on dependencies among the attributes of a relation. The fourth and fifth normal forms are based on multi-value and join dependencies and were proposed later.

➤

In-Class Question:

Question: What is normalization method?

Answer: Normalization methods allow the transformation of any element of an equivalence class of shapes under a group of geometric transforms into a specific one, fixed once for all in each class.

Slide 14

First Normal Form

The table has data related to projects and employees. The table must be split into two tables, that is, a Project Details table and an Employee Details table.

Project_Id	Project_Name
113	BLUE STAR
124	MAGNUM

Table 2.8: Project Details

Emp_No	Emp_Name	Grade	Salary
142	John	A	20,000
168	James	B	15,000
263	Andrew	C	10,000
109	Bob	C	10,000

© Aptech Ltd.

Session 2/ 14

Instructions to the Trainer(s):

- Using Slide 14, explain the First Normal Form.
- In order to achieve the first normal form, following steps must be performed:
 - Create separate tables for each group of related data
 - The table columns must have atomic values
 - All the key attributes must be identified

Second Normal Form

Tables are said to be in second normal form if:

- They meet the requirements of the first normal form
- There are no partial dependencies in the tables
- The tables are related through foreign keys

Project_Id	Project_Name
113	BLUE STAR
124	MAGNUM

Project Details After Conversion to Second Normal Form

Emp_No	Project_Id
142	113
142	124
168	113
263	113

Employee Project Details After Conversion to Second Normal Form

Emp_No	Emp_Name	Grade	Salary
142	John	A	20,000
168	James	B	15,000
263	Andrew	C	10,000

Employee Details After Conversion to Second Normal Form

© Aptech Ltd.

Session 2/15

Instructions to the Trainer(s):

- Using Slide 15, explain that students will learn the concept of Second Normal Form (2NF).
- Second Normal Form (2NF) is a normal form used in database normalization.
- The tables are said to be in second normal form if:
 - They meet the requirements of the first normal form
 - There are no partial dependencies in the tables
 - The tables are related through foreign keys

Third Normal Form

To achieve the third normal form:

- The tables should meet the requirements of the second normal form
- The tables should not have transitive dependencies in them

Project_Id	Project_Name
113	BLUE STAR
124	MAGNUM

Project Details After Conversion to Third Normal Form

Emp_No	Emp_Name	Grade
142	John	A
168	James	B
263	Andrew	C
109	Bob	C

Employee Details After Conversion to Third Normal Form

Emp_No	Project_Id
142	113
142	124
168	113
263	113
109	124

Employee Project Details After Conversion to Third Normal Form

Grade	Salary
A	20,000
B	15,000
C	10,000

Grade Salary Details After Conversion to Third Normal Form

© Aptech Ltd.

Session 2/16

Instructions to the Trainer(s):

- Using Slide 16, explain the third normal form.
- To achieve the third normal form:
 - The tables should meet the requirements of the second normal form
 - The tables should not have transitive dependencies in them
- The Project Details, Employee Details, and Employee Project Details tables are in second normal form. If an attribute can be determined by another non-key attribute, it is called a transitive dependency. To make it simpler, every non-key attribute should be determined by the key attribute only. If a non-key attribute can be determined by another non-key attribute, it has to put it into another table.
- On observing different tables, it is seen that the Project Details and Employee Project Details tables do not exhibit any such transitive dependencies. The non-key attributes are totally determined by the key attributes. Project_name is only determined by Project_number. On further scrutinizing the Employee Details table, a certain inconsistency is seen. The attribute Salary is determined by the attribute Grade and not the key attribute Emp_no. Thus, this transitive dependency must be removed.
- Also, mention, Boyce-Codd normal form.
- A database table is in BCNF if and only if there are no non-trivial functional dependencies of attributes on anything other than a superset of a candidate key. At first glance it would seem that BCNF and 3NF is the same thing. However, in some rare cases it does happen that a 3NF table is not BCNF-compliant. This may happen in tables with two or more overlapping composite candidate keys.

Denormalization

By normalizing a database, redundancy is reduced

This, in turn, reduces the storage requirements for the database and ensures data integrity.

Complex join queries may have to be written often to combine the data in multiple tables.

Joins may practically involve more than three tables depending on the need for information.

© Aptech Ltd.

Session 2/17

Instructions to the Trainer(s):

- Using Slide 17, discuss the concept of denormalization.
- By normalizing a database, redundancy is reduced. This, in turn, reduces the storage requirements for the database and ensures data integrity. However, it has some drawbacks. They are as follows:
 - Complex join queries may have to be written often to combine the data in multiple tables.
 - Joins may practically involve more than three tables depending on the necessity for information.
- Tell the students that if such joins are used very often, the performance of the database will become very poor. The CPU time required to solve such queries will be very large too. In such cases, storing a few fields redundantly can be ignored to increase the performance of the database. The databases that possess such minor redundancies in order to increase performance are called denormalized databases and the process of doing so is called denormalization.

Relational Operators 1-6

- The relational model is based on the solid foundation of Relational Algebra.
- Relational Algebra consists of a collection of operators that operate on relations.
- Each operator takes one or two relations as its input and produces a new relation as its output.

Branch	Branch_Id	Reserve (Billion €)
London	BS-01	9.2
London	BS-02	10
Paris	BS-03	15
Los Angeles	BS-04	50
Washington	BS-05	30

Instructions to the Trainer(s):

- Using Slide 18, explain the relational operators.
- The relational model is based on the solid foundation of Relational Algebra.
- Relational Algebra consists of a collection of operators that operate on relations.
- Each operator takes one or two relations as its input and produces a new relation as its output.

Relational Operators 2-6

It consists of a set of operators

SELECT Operator: The SELECT operator is used to extract data that satisfies a given condition. The lowercase Greek letter sigma, ' σ ', is used to denote selection

Branch	Branch_Id	Reserve (Billion €)
London	BS-01	9.2
London	BS-02	10

PROJECT Operator: The PROJECT operator is used to project certain details of a relational table. The PROJECT operator only displays the required details leaving out certain columns. The PROJECT operator is denoted by the Greek letter pi.

Branch_Id	Reserve (Billion €)
BS-01	9.2
BS-02	10
BS-03	15
BS-04	50
BS-05	30

Instructions to the Trainer(s):

- Using Slide 19, explain the SELECT and PROJECT operators.
- The SELECT operator is used to extract data that satisfies a given condition. The lowercase Greek letter sigma, ' σ ', is used to denote selection.
- The PROJECT operator is used to project certain details of a relational table. The PROJECT operator only displays the required details leaving out certain columns. The PROJECT operator is denoted by the Greek letter pi, ' ϵ '.

In-Class Question:

Question: What is the use of SELECT operator?

Answer: The SELECT operator is used to extract data that satisfies a given condition.

Relational Operators 3-6

PRODUCT Operator: The PRODUCT operator, denoted by 'x' helps combine information from two relational tables.

Branch_Id	Loan Amount (Billion €)
BS-01	0.56
BS-02	0.84

The product operation on the Branch Reserve Details and Branch Loan Details tables would result in table:

Branch	Branch_Id	Reserve (Billion €)	Loan Amount (Billion €)
London	BS-01	9.2	0.56
London	BS-01	9.2	0.84
London	BS-02	10	0.56
London	BS-02	10	0.84
Paris	BS-03	15	0.56
Paris	BS-03	15	0.84
Los Angeles	BS-04	50	0.56
Los Angeles	BS-04	50	0.84

The product operation combines each record from the first table with all the records in the second table, somewhat generating all possible combinations between the table records.

Instructions to the Trainer(s):

- Using Slide 20, explain the PRODUCT operator.
- The PRODUCT operator, denoted by 'x' helps combine information from two relational tables.
- The product operation on the Branch Reserve Details and Branch Loan Details tables would result in table.
- The product operation combines each record from the first table with all the records in the second table, somewhat generating all possible combinations between the table records.

In-Class Question:

Question: What is Cartesian product operation in DBMS?

Answer: The Cartesian Product is also an operator which works on two sets. It is sometimes called the CROSS PRODUCT or CROSS JOIN. It combines the tuples of one relation with all the tuples of the other relation.

Relational Operators 4-6

UNION Operator: It collects data from different tables and presents a unified version of the complete data. The union operator is represented by the symbol, 'U'.

Branch	Branch_Id
London	BS-01
London	BS-02
Paris	BS-03

INTERSECT Operator: The INTERSECT operator generates data that holds true in all the tables it is applied on. It is based on the intersection set theory and is represented by the ' \cap ' symbol

Branch	Branch_Id
London	BS-01
London	BS-02

Instructions to the Trainer(s):

- Using Slide 21, explain the Union and Intersect Operator.
- The Union operator collects data from different tables and present a unified version of the complete data.
- The union operation is represented by the symbol, 'U'.
- The INTERSECT operator generates data that holds true in all the tables it is applied on.
- It is based on the intersection set theory and is represented by the ' \cap ' symbol.

Relational Operators 5-6

DIFFERENCE Operator: The DIFFERENCE operator, symbolized as ' $-$ ', generates data from different tables too, but it generates data that holds true in one table and not the other.

Branch	Branch_Id
Paris	BS-03

JOIN Operator: The JOIN operation is an enhancement to the product operation. It allows a selection to be performed on the product of tables.

Branch	Branch_Id	Reserve (Billion €)	Loan Amount (Billion €)
London	BS-01	9.2	0.56
London	BS-02	10	0.84

Instructions to the Trainer(s):

- Using Slide 22, explain to students about Difference and Join Operators.
- The DIFFERENCE operator, symbolized as ' $-$ ', generates data from different tables too, but it generates data that holds true in one table and not the other.
- The JOIN operation is an enhancement to the product operation. It allows a selection to be performed on the product of tables.

Relational Operators 6-6

DIVIDE Operator: Suppose an official wanted to see the branch names and reserves of all the branches that had loans. This process can be made very easy by using the DIVIDE operator.

Branch	Reserve (Billion €)
London	9.2
London	10

Instructions to the Trainer(s):

- Using Slide 23, discuss the Divide Operator.
- Suppose an official wanted to see the branch names and reserves of all the branches that had loans. This process can be made very easy by using the divide operator.
- All that the official has to do is divide the Branch Reserve Details table by the list of branches, that is, the Branch Id column of the Branch Loan Details table.

Summary

- Data modeling is the process of applying an appropriate data model to the data at hand.
- E-R model views the real-world as a set of basic objects and relationships among them.
- Entity, attributes, entity set, relationships, and relationship sets form the five basic components of E-R model.
- Mapping cardinalities express the number of entities that an entity is associated with.
- The process of removing redundant data from the tables of a relational database is called normalization.
- Relational Algebra consists of a collection of operators that help retrieve data from the relational databases.
- SELECT, PRODUCT, UNION, and DIVIDE are some of the relational algebra operators.

Instructions to the Trainer(s):

- Show students Slide 24.
- Summarize the session by reading out each point on the slide.

Session 3: Introduction to SQL Server 2019

3.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. Prepare a question or two that will be a key point to relate the current session objectives.

3.1.1 Teaching Skills

To teach this session, you should be well versed with SQL Server and basic architecture of SQL Server 2019. You should also be familiar with the editions of SQL Server 2019 and execution of Transact-SQL statements.

You should teach the concepts in the theory class using the images provided. For teaching in the class, you are expected to use slides and LCD projectors.

In-Class Activities

Follow the order given here during In-Class activities.

Slide 2

Objectives

- Describe an overview of SQL Server 2019
- Describe basic architecture and version history of SQL Server 2019
- Outline the process of connecting to SQL Server instances
- Define databases and list the key features of AdventureWorks2019 sample database
- Explain the components of SQL Server Management Studio GUI
- Explain script file creation and organization
- Explain the process to execute Transact-SQL queries

Instructions to the Trainer(s):

Give students a brief overview of the current session through the session objectives listed in Slide 2.

3.2 In-Class Explanations

Slide 3

Introduction to SQL Server 2019

SQL Server is an RDBMS developed by Microsoft.

- Provides an enterprise-level data management platform
- Includes numerous features and tools
- Targeted for large scale Online Transaction Processing (OLTP), data warehousing and e-commerce applications
- Provides support for Big data clusters in SQL server
- Helps to store and manage huge amount of information

The diagram illustrates the relationship between various Relational Database Management Systems (RDBMS). At the center is a red circle labeled 'RDBMS'. Surrounding it are four other circles, each representing a different RDBMS system: 'Oracle' (green, top), 'MySQL' (green, right), 'PostgreSQL' (blue, bottom), and 'SQL Server' (purple, left). Lines connect the central 'RDBMS' circle to each of the surrounding systems, indicating their interconnected nature.

© Aptech Ltd.

Session 3/3

Instructions to the Trainer(s):

- Introduce the session Using Slide 3 by explaining SQL Server 2019.
- SQL Server is an RDBMS developed by Microsoft. It provides an enterprise-level data management platform for an organization.
- SQL Server includes numerous features and tools that make it an outstanding database and data analysis platform. It is also targeted for large-scale Online Transactional Processing (OLTP), data warehousing, and e-commerce applications.
- SQL Server is the new version of SQL Server and was launched by Microsoft.
- One of the major features of this version of SQL Server is that it is available on the cloud platform.
- Using SQL Server not only helps an organization to store and manage huge amount of information, but also to protect and utilize this data at different locations as required.

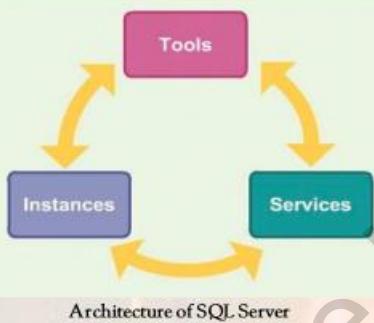
Features and Benefits of SQL Server			
Strong Security	Better Performance	Multiple Editions and Pricing Models	Simple and Easy Installation Process
<p>Policy-Based Management to detect security policies that are non-compliant. This feature allows only authorized personnel access to the database. Security audits and events can be written automatically to log files.</p>	<p>SQL Server has built-in transparent data compression feature along with encryption. SQL Server provides access control coupled with efficient permission management tools. It also offers an enhanced performance when it comes to data collection.</p>	<p>Microsoft has made available different editions of SQL Server for different kinds of users. These are also priced accordingly. Thus, from hobbyists to professional developers to enterprise users, there is an edition suitable for each one.</p>	<p>SQL Server is simple to install with a one-click installation procedure and readable GUI having easy instructions for the layman.</p>

Instructions to the Trainer(s):

- Using Slide 4, discuss with students the features and benefits of SQL Server 2019.
- Features and benefits are as follows:
 - **Strong Security:** This feature allows only authorized personnel access to the database. Security audits and events can be written to log files.
 - **Better Performance:** Provides access control coupled with efficient permission management tools. It offers enhanced performed when it comes to data collection.
 - **Multiple Editions and Pricing Models:** Microsoft has made available different editions SQL Server for different kinds of users. These are priced accordingly.
 - **Simple and Easy Installation process:** SQL Server is simple and easy to install having easy instructions for the layman.

Basic Architecture of SQL Server 2019 I-4

There are various components that form a part of SQL Server 2019. All the components come together to form the basic architecture of SQL Server 2019. These components can be represented under three major heads that are shown in figure:



Instructions to the Trainer(s):

- Using Slide 5, explain the basic architecture of SQL Server 2019.
- There are various components that form a part of SQL Server 2019.
- All the components come together to form the basic architecture of SQL Server 2019.

- These components can be represented under three major heads, namely:
 - Tools
 - Services
 - Instances

Basic Architecture of SQL Server 2019 2-4

Tools: There are a number of tools that are provided in SQL Server 2019 for development and query management of a database. The SQL Server Installation Center must be used to install SQL Server program features and tools. Features can also be modified or removed using SQL Server Installation Center.

Tool	Description
SQL Server Management Studio (SSMS)	One of the most important tools available in SQL Server 2019 is SSMS. SSMS is a GUI-based application provided with SQL Server 2019 that helps to create databases, database objects, query data, and manage the overall working of SQL Server.
SQLCMD	SQLCMD is a command-line tool that can be used in place of SSMS. It performs similar functions as SSMS, but in command format only.
SQL Server Installation Center	The SQL Server Installation Center tool can also be used to add, remove, and modify SQL Server programs.
SQL Server Configuration Manager	SQL Server Configuration Manager is used by database administrators to manage features of SQL software installed in client machines. This tool is not available to all users. It can be used to configure services, server protocols, client protocols, client aliases, and so on.
SQL Server Profiler	SQL Server Profiler is used to monitor an instance of the Database Engine or Analysis Services.
SQL Server Data Tools (SSDT)	SSDT is an Integrated Development Environment (IDE) used for Business Intelligence Components. It helps to design the database using a tool named Microsoft Visual Studio.
Connectivity Tools	Connectivity tools include DB-Library, Open Database Connectivity (ODBC), Object Linking and Embedding Database (OLE DB), and so on. These tools are used to communicate between the clients, servers, and network libraries.

© Aptech Ltd.

Session 3/6

Instructions to the Trainer(s):

- Using Slide 6, explain the tools used in SQL Server 2019.
- There are a number of tools that are provided in SQL Server 2019 for development and query management of a database.
- Different tools used are:
 - SQL Server Management Studio (SSMS)
 - SQL CMD
 - SQL Server Installation Center
 - SQL Server Configuration Manager
 - SQL Server Profiler
 - SQL Server Data Tools (SSDT)
 - Connectivity Tools

Basic Architecture of SQL Server 2019 3-4

Services: There are various services that are executed on a computer running SQL Server. These services run along with the other Windows services and can be viewed in the task manager.

SQL Server Database Engine	Database Engine is a core service that is used for storing, processing, and securing data. It is also used for replication, full-text search, and Data Quality Services (DQS). It contains tools for managing relational and eXtensible Markup Language (XML) data.
SQL Server Analysis Services (SSAS)	Analysis Services contain tools that help to create and manage Online Analytical Processing (OLAP). This is used for personal, team, and corporate business intelligence purposes. Analysis services are also used in data mining applications.
SQL Server Reporting Services (SSRS)	Reporting Services help to create, manage, publish, and deploy reports. These reports can be in tabular, matrix, graphical, or free-form format. Report applications can also be created using Reporting Services.
SQL Server Integration Services (SSIS)	Integration Services are used for moving, copying, and transforming data using different graphical tools and programmable objects. The DQS component is also included in Integration Services. Integration services help to build high-performance data integration solutions.
SQL Server Master Data Services	Master Data Services (MDS) are used for master data management. MDS is used for analysis, managing, and reporting information such as hierarchies, granular security, transactions, business rules, and so on.

Instructions to the Trainer(s):

- Using Slide 7, explain the services used in SQL Server 2019.
- There are various services that are executed on a computer running SQL Server.
- These services run along with the other Windows services and can be viewed in the task manager.
- Some of the SQL Server 2012 services are as follows:
 - **SQL Server Database Engine** - Database Engine is a core service that is used for storing, processing, and securing data. It is also used for replication, full-text search, and Data Quality Services (DQS). It contains tools for managing relational and eXtensible Markup Language (XML) data.
 - **SQL Server Analysis Services** - Analysis Services contain tools that help to create and manage Online Analytical Processing (OLAP). This is used for personal, team, and corporate business intelligence purposes. Analysis services are also used in data mining applications. These services also help to collaborate with PowerPivot, Excel, and even SharePoint Server Environment.
 - **SQL Server Reporting Services** - Reporting Services help to create, manage, publish, and deploy reports. These reports can be in tabular, matrix, graphical, or free-form format. Report applications can also be created using Reporting Services.
 - **SQL Server Integration Services** - Integration Services are used for moving, copying, and transforming data using different graphical tools and programmable objects. The DQS component is also included in Integration Services. Integration services help to build high-performance data integration solutions.

- **SQL Server Master Data Services** - Master Data Services (MDS) are used for master data management. MDS is used for analysis, managing, and reporting information such as hierarchies, granular security, transactions, business rules, and so on.

Slide 8

Basic Architecture of SQL Server 2019 4-4

Instances: All the programs and resource allocations are saved in an instance.

An instance includes the following:

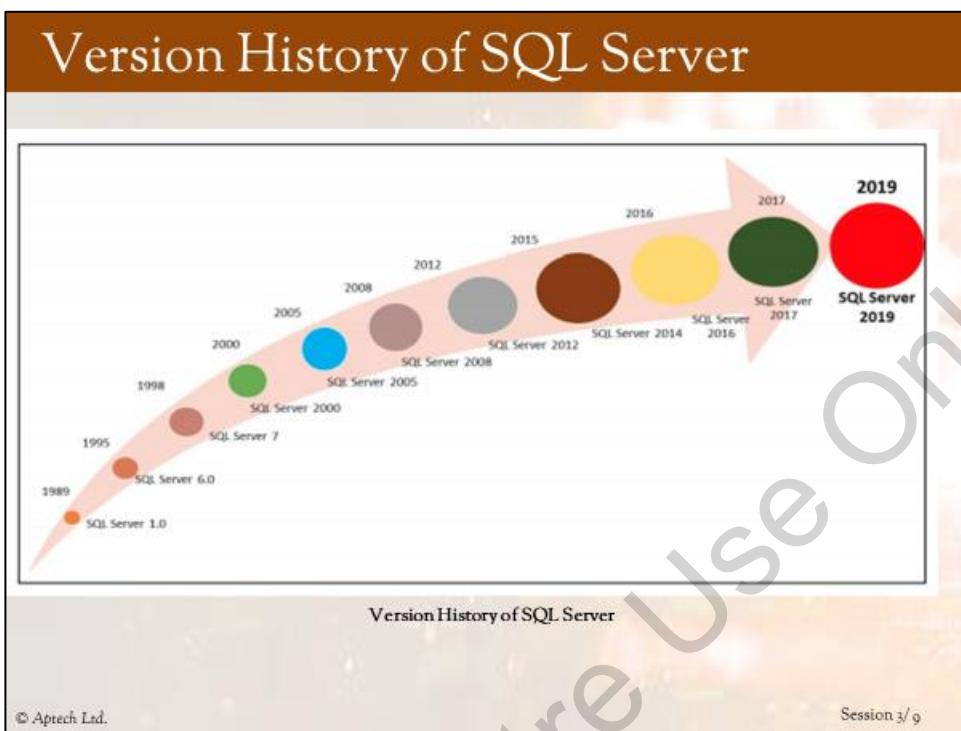
- Memory
- Configuration files
- CPU

➤ Each instance can be customized as per the requirement.
➤ Even permissions for each instance can be granted on individual basis.
➤ The resources can also be allocated to the instance accordingly, for example, the number of databases allowed.

© Aptech Ltd. Session 3/8

Instructions to the Trainer(s):

- Using Slide 8, explain the concept of instances to students.
- All the programs and resource allocations are saved in an instance.
- An instance can include:
 - Memory
 - Configuration files
 - CPU
- Multiple instances can be used for different users in SQL Server 2019. Each instance can be customized as per the requirement. Even permissions for each instance can be granted on individual basis.
- The resources can also be allocated to the instance accordingly, for example, the number of databases allowed.



Instructions to the Trainer(s):

- Using Slide 9, explain the version history of SQL Server.
- The first version of SQL Server was released in the year 1989.
- After this, there have been new versions released almost every year, with the latest one being SQL Server 2019.

Editions of SQL Server 1-2

- Based on database requirements, an organization can choose from any of these editions of SQL Server 2019 that have been released.
- Main editions of SQL Server 2019 are as follows:

Express/Web Edition	Standard/Web Edition	Enterprise/Web Edition
Key Features	Key Features	Key Features
<ul style="list-style-type: none"> Offers up to 16 cores of CPU for compute capacity Up to 64 GB of memory for buffer pool In-memory OLTP and Columnstore End-to-end encryption with secure enclaves Support for Linux and Windows containers UTF-8 character encoding Data classification and auditing 	<ul style="list-style-type: none"> Offers up to 24 cores of CPU for compute capacity Up to 128 GB of memory for buffer pool Supports automatic intelligent database tuning Azure Data Studio with notebook support Supports Big Data Clusters Supports Data virtualization by means of PolyBase Improved in-memory performance <p>In addition to Express/Web features</p>	<ul style="list-style-type: none"> Supports unlimited cores of CPU Provides unlimited memory for buffer pool Industry-leading performance with unmatched scalability Unlimited virtualization benefits Access to Power Business Intelligence (BI) Report Server <p>In addition to Standard and Express/Web features</p>

© Aptech Ltd. Session 3 / 10

Instructions to the Trainer(s):

- Using Slide 10, explain the editions of SQL Server.
- SQL Server is available in various editions, some of which include:
 - **Enterprise** – This is the edition that is recurrently released on most versions of SQL Server. This is the full edition of SQL Server which contains all the features of SQL Server 2019. The enterprise edition of SQL Server 2019 supports features such as Power View, xVelocity, Business Intelligence services, virtualization, and so on.
 - **Standard** – The standard edition is the basic edition of SQL Server that supports fundamental database and reporting and analytics functionality. However, it does not support critical application development, security, and data warehousing.
 - **Express** – Free to use and provides an easy level database for Web and Mobile apps.

For additional information, refer to:

<https://docs.microsoft.com/en-us/sql/sql-server/editions-and-components-of-sql-server-version-15?view=sql-server-ver15>

https://www.tutorialspoint.com/ms_sql_server/ms_sql_server_editions.htm

Editions of SQL Server 2-2

With each subsequent version, enhancements are made and new features are added

Features Added in SQL Server 2016

- Real-time operational analytics through PolyBase
- Support for R programming language
- SQL Server Machine Learning Services
- Dynamic data masking, Row level security, and Always encrypted
- Support for end to end mobile BI

Features Added in SQL Server 2017

- Support for Linux including Red Hat Enterprise Linux (RHEL), SUSE Linux Enterprise Server (SLES), and Ubuntu
- Support for Docker containers on Linux and Windows
- Python language support
- Automatic plan correction and adaptive query processing
- Cross platform availability groups
- Support for graph data
- Power BI reporting both on-premises and in the cloud
- Access to Power BI Report Server

Features Added in SQL Server 2019

- Big Data clusters with Apache Spark and HDFS
- Data virtualization to integrate external data sources
- Azure Machine Learning and Spark ML, Support for Kubernetes deployment, Free supported Java
- Native UTF-8 support Intelligent Query processing
- In-Memory Database: Persistent Memory support
- Accelerated database recovery, Free Data Recovery to Azure
- Always Encrypted with secure areas, Data classification, and auditing Vulnerability assessment
- Notebook support for T-SQL, Python, R, and Scala in Azure Data Studio
- SQL Server Analysis

© Aptech Ltd. Session 3/11

Instructions to the Trainer(s):

- Using Slide 11, explain the features added in major recent versions of SQL Server.
- Features added in SQL Server 2016 are:
 - Real time operational analytics
 - Support for R Programming language
 - SQL Server Machine learning services
 - Dynamic data masking, Row level security, always encrypted
- Features added in SQL Server 2017 are:
 - Support for Linux including Red Hat Enterprise Linux (RHEL)
 - Python language support
 - Support for graph data
 - Cross platform availability groups
- Features added in SQL Server 2019 are:
 - Big Data clusters with Apache Spark and HDFS
 - Azure Machine Learning and Spark ML
 - Accelerated database recovery
 - Network support for T-SQL, Python, R and so on

Connecting to SQL Server Instances 1-2

- SQL Server Management Studio (SSMS) is used to connect to SQL Server instances.
- SSMS is a tool used for
 - creating
 - querying
 - managing the databases

The steps for connecting to MySQL Server are as follows:

- Locate the Microsoft SQL Server Management Studio tool on the list of programs on Start menu and start the tool.
- In the Connect to Server dialog box, select the Server type as Database Engine.
- Type the Server name.
- Select either Windows Authentication or SQL Server Authentication, provide the required Login and Password, and click Connect.

Instructions to the Trainer(s):

- Using Slide 12, explain how to connect to SQL Server instances.
- SSMS is used to connect to SQL Server instances.
- SSMS is a tool used for creating, querying, and managing the databases.
- To open SSMS, connect to SQL Server 2019 by specifying the sever information and login credentials. The login credentials will include username and password. The detailed steps to connect to SQL Server instance are as follows:
 - Click **Start → All Programs → Microsoft SQL Server 2019 → SQL Server Management Studio.**
 - In the Connect to Server dialog box, select the Server type as Database Engine.
 - Type the Server name.



Instructions to the Trainer(s):

- Using Slide 13, explain students how to connect to SQL Server.
- Tell the students that SQL Server displays the Dialog box which consists of various credentials required.
- After the credentials are entered and authentication is done, students can view the SSMS Window.

In-Class Question:

Question: What is the use of SSMS?

Answer: The use of SSMS is to access, configure, manage, administer, and develop all components of SQL Server, Azure SQL Database, and Azure Synapse Analytics.

Introduction to Databases 1-2

A database is a collection of data stored in data files on a disk or some removable medium, which consists of data files to hold actual data

An SQL Server is made up of tables that stores sets of specific structured data

A table includes a set of rows (also called as records or tuples) and columns (also called as attributes)

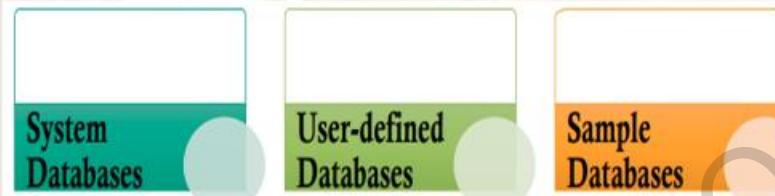
Each column in the table is intended to store a specific type of information, for example, dates, names, currency amounts, and numbers.

Instructions to the Trainer(s):

- Using Slide 14, introduce students to database.
- A database is an organized collection of data, generally stored and accessed electronically from a computer system. It supports the storage and manipulation of data.
- An SQL Server is made of tables that stores sets of specific structured data.
- A table includes a set of rows and columns.
- Each column in the table is intended to store a specific type of information.

Introduction to Databases 2-2

SQL Server 2019 supports three kinds of databases, which are as follows:



© Aptech Ltd.

Session 3 / 15

Instructions to the Trainer(s):

- Using Slide 15, explain different kinds of databases.
- SQL Server 2019 supports three kinds of database, they are as follows:
 - System database
 - User-defined database
 - Sample database

System Databases 1-2

SQL Server uses system databases to support different parts of the DBMS.

Each database supports the following:

- Has a specific role and stores job information that requires to be carried out by SQL Server
- Store data in tables, which contain the views, stored procedures, and other database objects
- Users can create their own databases, also called user-defined databases, and work with them

Instructions to the Trainer(s):

- Using Slide 16, explain the system database in SQL Server.
- SQL Server use system database to support different parts of DBMS.
- Each database supports the following:
 - Has a specific role and stores job information that requires to be carried out by SQL Server.
 - Store data in tables, which contain the views, stored procedures, and other database objects.
 - Users can create their own database, also called user-defined database and work with them.

System Databases 2-2	
Database	Description
master	The database records all system-level information of an instance of SQL Server.
msdb	The database is used by SQL Server Agent for scheduling database alerts and various jobs.
model	The database is used as a template for all databases to be created on the particular instance of SQL Server 2019.
resource	The database is a read-only database. It contains system objects included with SQL Server 2019.
tempdb	The database holds temporary objects or intermediate result sets.

System Databases

© Aptech Ltd. Session 3 / 17

Instructions to the Trainer(s):

- Using Slide 17, discuss the types of system database in SQL Server.
- Different types of system database are explained as follows:
 - **Master:** The database records all system-level information of an instance of SQL Server.
 - **MSDB:** Used for scheduling database alerts and various jobs.
 - **Model:** Template database for all user defined databases to be created on the particular instance of SQL Server 2019.
 - **Resource:** The database is a read-only database. It contains system objects included with SQL Server 2019.
 - **TempDB:** Temporary database to store temporary tables.

User-defined Databases

Using SQL Server 2019:

- Users can create their own databases, also called user-defined databases, and work with them.
- The purpose of these databases is to store user data.

© Aptech Ltd.

Session 3 / 18

Instructions to the Trainer(s):

- Using Slide 18, explain the user-defined database in SQL Server.
- Users can create their own database, also called user-defined databases, and work with them.
- The purpose is to store user data.

AdventureWorks2019 Sample Database 1-2

This database demonstrates use of new features introduced in SQL Server

The AdventureWorks2019 database schema covers many functional areas for a fictitious bicycle manufacturer. These areas include:

- Customer/sales force automation and analysis
- Human resources
- Purchasing/Vendor Electronic Data Interchange
- Manufacturing work flow

© Aptech Ltd. Session 3 / 19

Instructions to the Trainer(s):

- Using Slide 19, explain the adventure Works2019 Sample Database in SQL Server.
- AdventureWorks2019 database schema covers many functional areas for a fictitious bicycle manufacturer.
- These areas include:
 - Customer/sales force automation and analysis
 - Human resources
 - Purchasing/Vendor Electronic Data Interchange
 - Manufacturing work flow

AdventureWorks2019 Sample Database 2-2

The database comprises several features. Some of its key features are as follows:

A database engine that includes administration facilities, data access capabilities, Full-Text Search facility, Common Language Runtime (CLR) integration advantage, and more

A set of integrated samples for two multiple feature-based samples: HRResume and Storefront

Analysis Services and Integration Services

Notification Services

Replication Facilities

Reporting Services

The sample database consists of these parts:

- AdventureWorks2019: Sample OLTP database
- AdventureWorks2019DW: Sample Data warehouse
- AdventureWorks2019AS: Sample Analysis Services database

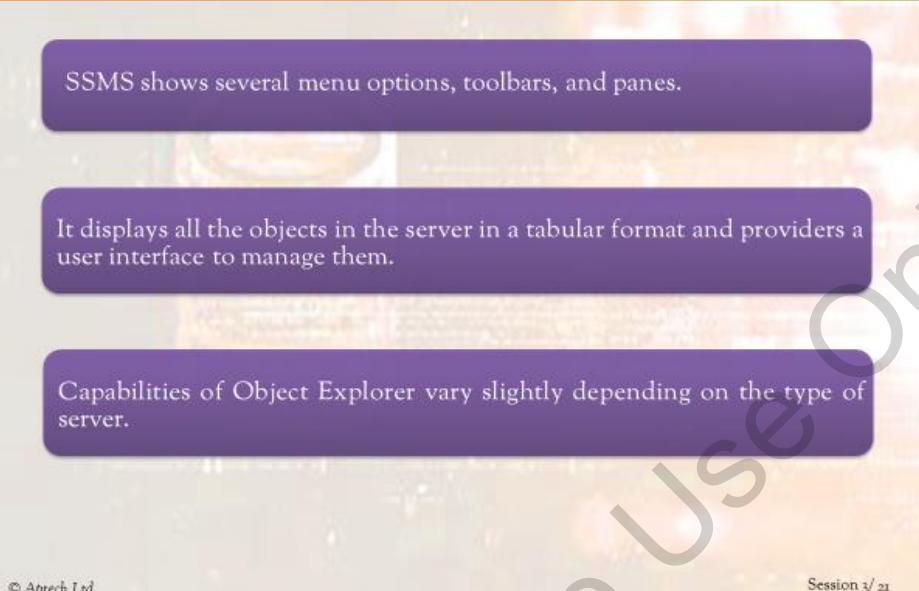
© Aptech Ltd.

Session 3 / 20

Instructions to the Trainer(s):

- Using Slide 20, explain the key features in sample database.
- The database comprises several features.
- Key features include:
 - A database engine includes administration facilities, data across capabilities, full text search facility, common language runtime, integration advantage, and more
 - A set of integrated samples for two-multiple feature-based samples: HRResume and Storefront
 - Analysis Services and Integration Services
 - Notification Services
 - Replication Facilities
 - Reporting Services

Understanding the SSMS User Interface



SSMS shows several menu options, toolbars, and panes.

It displays all the objects in the server in a tabular format and provides a user interface to manage them.

Capabilities of Object Explorer vary slightly depending on the type of server.

© Aptech Ltd. Session 3/ 21

Instructions to the Trainer(s):

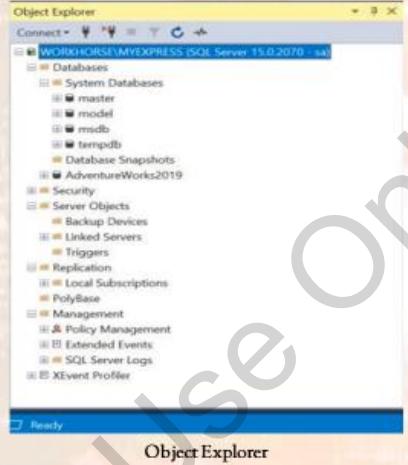
- Using Slide 21, explain the SSMS Interface.
- SSMS stands for SQL Server Management Studio.
- SSMS shows several menu options, toolbars, and panes.
- It displays all objects in the server in a tabular format and provides a user interface to manage them.
- SSMS provides tools to configure, monitor, and administer instances of SQL Server and databases.
- SSMS is very popular and widely used by the database developers because of following advantages:
 - Cost-free
 - Advanced user experience
 - Various add-in options
 - Easy installation

Role and Structure of Object Explorer in SQL Server 1-2

The structure includes

- Databases
- Security
- server objects
- Replications
- AlwaysOn High Availability
- Management
- Integration Services Catalogs

• Object Explorer can be accessed through SSMS by connecting to the database server.



The screenshot shows the SQL Server Object Explorer window. The left pane displays a tree view of database objects under the selected database 'WORKHORSE\MYEXPRESS'. The tree includes categories like System Databases, Security, Server Objects, Replication, and Management. Under 'System Databases', it lists master, model, msdb, tempdb, Database Snapshots, and AdventureWorks2019. Under 'Management', it lists Policy Management, Extended Events, SQL Server Logs, and XEvent Profiler. The right pane is labeled 'Object Explorer'.

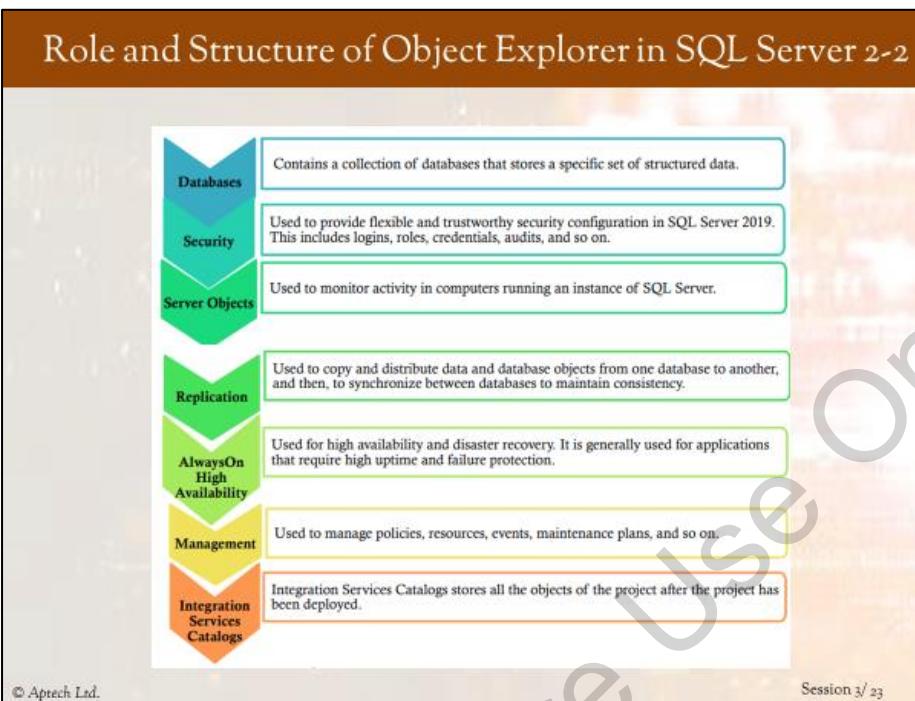
© Aptech Ltd.

Session 3 / 22

Instructions to the Trainer(s):

- Using Slide 22, students will learn the Difference and Join Operator.
- Object Explorer can be accessed through SSMS by connecting to the database server.
- The Object Explorer Details pane presents a tabular view of instance objects and the capability to search for specific objects.

- The structure includes:
 - Database
 - Security
 - Server objects
 - Replications
 - AlwaysOn High Availability
 - Management
 - Integration Services Catalogs



Instructions to the Trainer(s):

- Using Slide 23, discuss different structures in SQL server.
- Following are different structures explained as follows:
 - **Databases:** Contains a collection of databases that stores a specific set of structured data.
 - **Security:** Used to provide flexible and trustworthy security configuration in SQL Server 2019. This includes logins, roles, credentials, audits, and so on.
 - **Server Objects:** Used to monitor activity in computers running an instance of SQL Server.
 - **Replication:** Used to copy and distribute data and database objects from one database to another.
 - **AlwaysOn High Availability:** Used for high availability and disaster recovery.
 - **Management:** Used to manage policies, resources, events, maintenance plan, and so on.
 - **Integration Services Catalog:** Stores all objects of the project after it has been deployed.

Query Window

Query window is the area where you can type Transact-SQL (T-SQL) queries. Results of your queries also appear in this window.

The screenshot shows the SQL Server Management Studio (SSMS) interface. A query window titled 'SQLQuery1.sql - LA...orks2019 (sa (52))' contains the following T-SQL code:

```
1 USE Adventureworks2019
2 SELECT TOP 10 Name FROM HumanResources.Shift
```

The results pane below the query window displays the output of the query:

Name
1 Day
2 Evening
3 Night

At the bottom left of the slide, there is a small watermark-like text: '© Aptech Ltd.' and at the bottom right: 'Session 3 / 24'.

Instructions to the Trainer(s):

- Using Slide 24, discuss the query window in SQL Server.
- Query Window is the area where you can type Transact-SQL (T-SQL) queries.

Creating and Organizing Script Files 1-2

- Script files are files that contain a set of SQL commands.
- A script file can contain one or more SQL statements.
- The script files are stored in .sql format in SQL Server.

The diagram illustrates the conceptual layers of creating and organizing script files. It consists of three green rectangular boxes stacked vertically, each containing a white text label. A blue downward-pointing arrow is positioned between the first two boxes. The bottom box is labeled 'Conceptual Layers' at its base. The labels from top to bottom are 'Solution', 'Project', and 'Script'. The entire diagram is set against a background of a city skyline at night.

© Aptech Ltd. Session 3 / 25

Instructions to the Trainer(s):

- Using Slide 25, explain the students how to create and organize script files in SQL Server.
- Script files are files that contain a set of SQL commands.
- A script file can contain one or more SQL statements.
- The script files are stored in .sql format in SQL Server.
- The conceptual layers consist of:
 - Solution, Project and Script

Creating and Organizing Script Files 2-2

```
USE [AdventureWorksLT]
GO
INSERT INTO [Person].[Person]
([BusinessEntityID]
,[PersonType]
,[NameStyle]
,[Title]
,[FirstName]
,[MiddleName]
,[LastName]
,[EmailPromotion]
,[ModifiedDate])
VALUES(21987
,'EM'
,0
,'Mr.'
,'John'
,'Gareth'
,'Hopkins'
,0
,'2020-10-10')
GO
```

Code Snippet

© Aptech Ltd.

Session 3 / 26

Instructions to the Trainer(s):

- Using Slide 26, students can view the code snippet for Script files in SQL Server.

Transact-SQL Queries

Queries typed in Transact-SQL and saved as .sql files can be executed directly in the SSMS query window.

Steps to execute Transact-SQL queries are as follows:

1. In the query window, select the code to be executed.
2. On the SSMS toolbar, click Execute.
OR
On the Query menu, click Execute.
OR
Press F5 or Alt+X or Ctrl+E.

Query results can be displayed in three different formats. The three formats available are grid, text, and file view.

© Aptech Ltd. Session 3 / 27

Instructions to the Trainer(s):

- Using Slide 27, discuss the Transact-SQL Queries.
- The queries typed in Transact-SQL and saved as .sql files can be executed directly in the SSMS query window. The steps to execute Transact-SQL queries are as follows:
 - In the query window, select the code to be executed.
 - On the SSMS toolbar, click **Execute**. Alternatively, on Query menu, click **Execute** OR Press F5, Alt+X, or Ctrl+E.
- The query results can be displayed in three different formats. The three formats available are grid, text, and file view.
- The WITH RESULT SETS clause can also be used with a stored procedure, which returns multiple result sets and for each result set you can define the column name and data types for each column separately.

Summary

- Basic architecture of SQL Server 2019 includes tools, services, and instances.
- Three major editions of SQL Server are Express, Standard, and Enterprise.
- The structure of Object Explorer includes databases, security, server objects, replications, AlwaysOn High Availability, Management, Integration Services Catalogs, and so on.
- SSMS is used to connect to SQL Server instances. SSMS is a tool used for developing, querying, and managing the databases.
- Script files should be stored in .sql format in SQL Server 2019.
- Queries typed in Transact-SQL and saved as .sql files can be executed directly into the SSMS query window.

Instructions to the Trainer(s):

- Show students Slide 28.
- Summarize the session by reading out each point on the slide.

Session 4: Transact-SQL

4.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. Prepare a question or two that will be a key point to relate the current session objectives.

4.1.1 Teaching Skills

To teach this session, you should be well versed with SQL and Transact-SQL, their language elements, and operators.

You should teach the concepts in the theory class using the images provided. For teaching in the class, you are expected to use slides and LCD projectors.

In-Class Activities

Follow the order given here during In-Class activities.

Slide 2

Objectives

- Explain Transact-SQL
- List different categories of Transact-SQL statements
- Explain various data types supported by Transact-SQL
- Explain Transact-SQL language elements
- Explain sets and predicate logic
- Describe logical order of operators in the SELECT statement

© Aptech Ltd. Session 4 / 2

Instructions to the Trainer(s):

Give students a brief overview of the current session through the session objectives listed in Slide 2.

4.2 In-Class Explanations

Slide 3

Introduction

- ❑ SQL is the universal language used in the database world.
 - Most modern RDBMS products use some type of SQL dialect as their primary query language.
 - SQL can be used to create or destroy objects such as tables on the database server and to manipulate those objects, such as adding data into them or retrieving data from them.
- ❑ Transact-SQL is Microsoft's implementation of the standard SQL.
- ❑ Usually referred to as T-SQL, this language implements a standardized way to communicate to the database.
- ❑ It provides a comprehensive language that supports defining tables, inserting, deleting, updating, and accessing the data in the table.



© Aptech Ltd. Session 4/ 3

Instructions to the Trainer(s):

- Introduce the session Using Slide 3, explain the transact-SQL to the students.
- SQL is the universal language used in the database world. Most modern RDBMS products use some type of SQL dialect as their primary query language.
- SQL can be used to create or destroy objects, such as tables, on the database server and to do things with those objects, such as put data into them or query for data.
- Transact-SQL is Microsoft's implementation of the standard SQL. Usually referred to as T-SQL, this language implements a standardized way to communicate to the database.
- The Transact-SQL language is an enhancement to SQL, the American National Standards Institute (ANSI) standard relational database language.
- Transact-SQL provides a comprehensive language that supports
 - defining tables, inserting, deleting, updating, and accessing the data in the table.

Transact-SQL I-2

Transact-SQL is a powerful language offering features such as:

- data types
- temporary objects
- extended stored procedures
- Scrollable cursors
- conditional processing
- transaction control
- exception and error-handling

The Transact-SQL language also provides:

- improved performance
- increased functionality
- enhanced features.

Enhancements include scalar functions, paging, sequences, meta-data discovery, and better error handling support.

Instructions to the Trainer(s):

- Using Slide 4, explain the Transact-SQL.
- Transact-SQL is a powerful language offering features such as data types, temporary objects, and extended stored procedures.
- Scrollable cursors, conditional processing, transaction control, and exception and error-handling are also some of the features which are supported by Transact-SQL.
- The Transact-SQL language in SQL Server provides:
 - Improved Performance
 - Increased Functionality
 - Enhanced Features
- Enhancements include scalar functions, paging, sequences, meta-data discovery, and better error handling support.

Transact-SQL 2-2

Code Snippet 1:

```
USE AdventureWorks2019
SELECT LoginID FROM HumanResources.Employee
WHERE JobTitle = 'Design Engineer'
```

Results

LoginID
1 adventure-works\gail0
2 adventure-works\jossef0
3 adventure-works\sharon0

➤ Transact-SQL includes many syntax elements that are used by or that influence most statements.
➤ These elements include data types, predicates, functions, variables, expressions, control-of-flow, comments, and batch separators.

© Aptech Ltd. Session 4/5

Instructions to the Trainer(s):

- Using Slide 5, explain the elements in Transact-SQL.
- Transact-SQL includes many syntax elements that are used by or that influence most statements.
- These elements include data types, predicates, functions, variables, expressions, control-of-flow, comments, and batch separators.

Different Categories of Transact-SQL Statements

SQL Server supports three types of Transact-SQL statements, namely

- DDL
- DML
- DCL

Instructions to the Trainer(s):

- Using Slide 6, list and explain different categories of Transact-SQL.
- SQL Server supports three types of Transact-SQL statements, namely
 - **DDL:** Data Definition Language
 - **DML:** Data Manipulation Language
 - **DCL:** Data Control Language

Data Definition Language (DDL)

DDL is used to define and manage all attributes and properties of a database which includes:

- row layouts
- column definitions
- key columns
- file locations
- storage strategy

Most DDL statements take the following form, where object.name can be a table, view, trigger, stored procedure, and so on:

- CREATE object.name
- ALTER object.name
- DROP object.name

Instructions to the Trainer(s):

- Using Slide 7, explain the DDL commands in detail.
- DDL, which is usually part of a DBMS, is used to define and manage all attributes and properties of a database, including row layouts, column definitions, key columns, file locations, and storage strategy.
- DDL statements are used to build and modify the structure of tables and other objects such as views, triggers, stored procedures, and so on.
- For each object, there are usually CREATE, ALTER, and DROP statements (such as, CREATE TABLE, ALTER TABLE, and DROP TABLE).
- Most DDL statements take following forms:
 - CREATE object_name
 - ALTER object_name
 - DROP object_name
- In DDL statements, object_name can be a table, view, trigger, stored procedure, and so on.

Data Manipulation Language (DML)

DML is used to select, insert, update, or delete data in the objects defined with DDL

All database users can use these commands during the routine operations on a database

Different DML statements are as follows:

- SELECT statement
- INSERT statement
- UPDATE statement
- DELETE statement

Instructions to the Trainer(s):

- Using Slide 8, explain the concept of DML in detail.
- DML is used to select, insert, update, or delete data in the objects defined with DDL.
- All database users can use these commands during the routine operations on a database.
- Different DML statements are as follows:
 - SELECT statement
 - INSERT statement
 - UPDATE statement
 - DELETE statement

Data Control Language (DCL)

Data control language is used to control permissions on database objects

DCL statements are also used for securing the database

The three basic DCL statements are as follows:

- GRANT statement
- REVOKE statement
- DENY statement

Instructions to the Trainer(s):

- Using Slide 9, explain the DCL command in detail.
- Data is an important part of database, so proper steps should be taken to check that no invalid user accesses the data.
- Data control language is used to control permissions on database objects.
- Permissions are controlled by using the GRANT, REVOKE, and DENY statements.
- DCL statements are also used for securing the database. The three basic DCL statements are as follows:
 - GRANT statement
 - REVOKE statement
 - DENY statement

In-Class Question:

Question: What is the use of DCL?

Answer: Permissions are controlled by using the GRANT, REVOKE, and DENY statements which are DCL statements.

Data Types 1-3

- A data type is an attribute defining the type of data that an object can contain.
- Data types must be provided for columns, parameters, variables, and functions that return data values, and stored procedures that have a return code.

Following objects have data types:

- Columns present in tables and views
- Parameters in stored procedures
- Variables
- Transact-SQL functions that return one or more data values of a specific data type
- Stored procedures that have a return code belonging to the integer data type

Instructions to the Trainer(s):

- Using Slide 10, explain the data types in Transact-SQL.
- A data type is an attribute defining the type of data that an object can contain.
- Data types must be provided for columns, parameters, variables, and functions that return data values, and stored procedures that have a return code.
- Transact-SQL includes a number of base data types, such as varchar, text, and int.
- All data that is stored in SQL Server must be compatible with one of the base data types.
- Following objects have data types:
 - Columns present in tables and views
 - Parameters in stored procedures
 - Variables
 - Transact-SQL functions that return one or more data values of a specific data type
 - Stored procedures that have a return code belonging to the integer data type

System-defined data types		
Category	Data Type	A Column of This Type
Exact Numerics	int	Occupies four bytes of memory space. Is typically used to hold integer values. Can hold integer data from -2^31 (-2,147,483,648) to 2^31-1 (2,147,483,647).
	smallint	Occupies two bytes of memory space. Can hold integer data from -32,768 to 32,767.
	tinyint	Occupies one byte of memory space. Can hold integer data from 0 to 255.
	bigint	Occupies 8 bytes of memory space. Can hold data in the range -2^63 (-9,223,372,036,854,775,808) to 2^63-1 (9,223,372,036,854,775,807).
	numeric	Has fixed precision and scale.
Approximate Numerics	money	Occupies eight bytes of memory space. Represents monetary data values ranging from -922,337,203,685,477,5808 to 2^63-1 (922,337,203,685,477,5807).
	float	Occupies eight bytes of memory space. Represents floating point number ranging from -1.79E+308 through 1.79E+308.
	real	Occupies four bytes of memory space. Represents floating precision number ranging from -3.40E+38 through 3.40E+38.
Date and Time	datetime	Represents date and time. Stored as two 4-byte integers.
	smalldatetime	Represents date and time.
Character string	char	Stores character data that is fixed-length and non-Unicode.
	varchar	Stores character data that is variable-length and non-Unicode with a maximum of 8,000 characters.
	text	Stores character data that is variable-length and non-Unicode with a maximum length of 2^31 - 1 (2,147,483,647) characters.
Unicode types	nchar	Stores Unicode character data of fixed-length.
	nvarchar	Stores variable-length Unicode character data.
Other Data Types	timestamp	Occupies 8 bytes of memory space. Can hold automatically generated, unique binary numbers that are generated for a database.
	binary(n)	Stores fixed-length binary data with a maximum length of 8000 bytes.
	varbinary(n)	Stores variable-length binary data with a maximum length of 8000 bytes.
	image	Stores variable-length binary data with a maximum length of 2^30-1 (1,073,741,823) bytes.
	uniqueidentifier	Occupies 16 bytes of memory space. Also, stores a globally unique identifier (GUID).

© Aptech Ltd.

Session 4 / 11

Instructions to the Trainer(s):

- Using Slide 11, explain different types of data types.
- A data type constrains the values that an expression, such as a variable or a function, might take. This data type defines the operations that can be done on the data, the meaning of the data, and the way values of that type can be stored.
- SQL Server supports following data type's categories:
 - **Exact numeric:** bit, tinyint, smallint, int, bigint, decimal, numeric, money, and smallmoney
 - **Approximate numeric:** Read and float
 - **Date and time:** date, DateTime, datetime2, datetimeoffset, smalldatetime, and time
 - **Character strings:** char, varchar, and text
 - **Unicode character strings:** Nchar, Nvarchar, and Ntext
 - **Binary strings:** Binary, image, and varbinary
 - **Other data types:** Cursor, hierarchyid, sql_variant, table, rowversion, uniqueidentifier, XML, Spatial, and geography

Data Types 3-3

Alias data types

- These are based on the system-supplied data types.
- Alias data types are used when more than one table stores the same type of data in a column and has similar characteristics such as length, nullability.

```
CREATE TYPE [ schema_name. ] type_name { FROM base_type [ (precision [, scale]
    [ ] [ NULL | NOT NULL ] ) [ ; ] ] }
```

User-defined types

These are created using programming languages supported by the .NET Framework.

© Aptech Ltd.

Session 4 / 12

Instructions to the Trainer(s):

- Using Slide 12, explain the alias and user-defined types in Transact-SQL.
- **Alias data types:**
 - These are based on the system-supplied data types.
 - Alias data types are used when more than one table stores the same type of data in a column and has similar characteristics such as length, nullability, and type. In such cases, an alias data type can be created that can be used commonly by all these tables.
 - Alias data types can be created using the CREATE TYPE statement.
- **User-defined types:**
 - These are created using programming languages supported by the .NET Framework.
- Mention varchar(max), nvarchar(max), and varbinary(max) are large value data types while text, ntext, image, varchar(max), nvarchar(max), varbinary(max), and xml are large object data types.

Transact-SQL Language Elements

- The Transact-SQL language elements are used in SQL Server 2019 for working on the data that is entered in SQL Server database.

The Transact-SQL language elements includes:

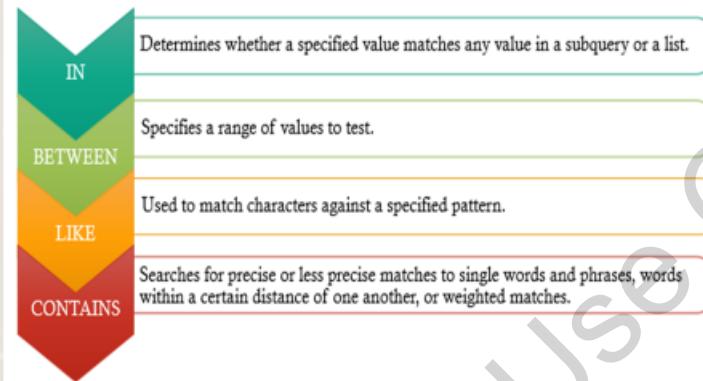
- Predicates
- Operators
- Functions
- Variables
- Expressions
- control-of-flow
- Errors and
- transactions, comments, and batch separators.

Instructions to the Trainer(s):

- Using Slide 13, explain students the elements in Transact-SQL.
- The Transact-SQL language elements are used for working on the data that is entered in SQL Server database.
- Tell the students that Transact-SQL language elements include:
 - Predicates
 - Operators
 - Functions
 - Variables
 - Expressions
 - Control-of-flow
 - Errors
 - Transactions, comments, and batch separators

Predicates and Operators 1-2

Predicates are used to evaluate whether an expression is TRUE, FALSE, or UNKNOWN.



© Aptech Ltd.

Session 4 / L4

Instructions to the Trainer(s):

- Using Slide 14, explain the predicates and operators in Transact-SQL.
- Predicates are used to evaluate whether an expression is TRUE, FALSE, or UNKNOWN.
- Some of the predicates available in Transact-SQL are as follows:
 - **IN** - Determines whether a specified value matches any value in a subquery or a list.
 - **BETWEEN** - Specifies a range of values to test.
 - **LIKE** - Used to match characters against a specified pattern.
 - **CONTAINS** - Searches for precise or less precise matches to single words and phrases, words within a certain distance of one another, or weighted matches.

Predicate	Example
IN	SELECT PersonType, Title, FirstName, LastName FROM AdventureWorks2019.Person.Person WHERE PersonType IN ('EM', 'SC')
BETWEEN	SELECT BusinessEntityID, NationalIDNumber, LoginID, JobTitle, HireDate FROM AdventureWorks2019.HumanResources.Employee WHERE HireDate BETWEEN '01-01-2010' AND '01-01-2013'
LIKE	SELECT DepartmentID, Name,GroupName,ModifiedDate FROM AdventureWorks2019.HumanResources.Department WHERE Name LIKE '%s'
CONTAINS	SELECT * FROM AdventureWorks2019.Person.Address WHERE CONTAINS(AddressLine1, 'Street')

Note that this statement will work only on a table with full text index.

Predicate Examples

Order	Operators
1	() Parentheses
2	*, /, %
3	+, -
4	=, <, >, >=, <=, !=, !=>
5	NOT
6	AND
7	BETWEEN, IN, CONTAINS, LIKE, OR
8	=

Precedence of Predicates and Operators

Instructions to the Trainer(s):

- Using Slide 15, explain the predicates and operators.
- A predicate is a condition expression that evaluates to a boolean value, either true or false.
- Predicates can be used as follows: In a SELECT statement's WHERE clause or HAVING clause to determine which rows are relevant to a particular query.
- Operators are used to perform arithmetic, comparison, concatenation, or assignment of values. For example, data can be tested to verify that the COUNTRY column for the customer data is populated (or has a NOT NULL value).
- The table in Slide 15 displays different types of operators.

Functions I-2

A function is a set of Transact-SQL statements that is used to perform some task. These functions can be useful when data is calculated or manipulated.

The four types of functions in SQL Server 2019 are as follows:

- Rowset functions**
In Transact-SQL, the rowset function is used to return an object that can be used in place of a table reference. For example, OPENDATASOURCE, OPENQUERY, OPENROWSET, and OPENXML are rowset functions.
- Aggregate functions**
Transact-SQL provides aggregate functions to assist with the summarization of large volumes of data. For example, SUM, MIN, MAX, AVG, COUNT, COUNTBIG, and so on are aggregate functions.
- Ranking functions**
Many tasks, such as creating arrays, generating sequential numbers, finding ranks, and so on can be implemented in an easier and faster way by using ranking functions. For example, RANK, DENSE_RANK, NTILE, and ROW_NUMBER are ranking functions.
- Scalar functions**
In scalar functions, the input is a single value and the output received is also a single value.

© Aptech Ltd. Session 4 / 16

Instructions to the Trainer(s):

- Using Slide 16, explain the functions in Transact-SQL.
- A function is a set of Transact-SQL statements that is used to perform some task.
- Transact-SQL includes a large number of functions. These functions can be useful when data is calculated or manipulated.
- In SQL, functions work on the data, or group of data, to return a required value. They can be used in a SELECT list, or anywhere in an expression.
- Four types of functions are as follows:
 - **Rowset functions** - In Transact-SQL, the rowset function is used to return an object that can be used in place of a table reference. For example, OPENDATASOURCE, OPENQUERY, OPENROWSET, and OPENXML are rowset functions.
 - **Aggregate functions** - Transact-SQL provides aggregate functions to assist with the summarization of large volumes of data. For example, SUM, MIN, MAX, AVG, COUNT, COUNTBIG, and so on are aggregate functions.
 - **Ranking functions** - Many tasks, such as creating arrays, generating sequential numbers, finding ranks, and so on can be implemented in an easier and faster way by using ranking functions. For example, RANK, DENSE_RANK, NTILE, and ROW_NUMBER are ranking functions.
 - **Scalar functions** - In scalar functions, the input is a single value and the output received is also a single value.

Functions 2-2

Function Type	Description	Example
Conversion function	The conversion function is used to transform a value of one data type to another. Additionally, it can be used to obtain a variety of special date formats.	CONVERT
Date and time function	Date and time functions are used to manipulate date and time values. They are useful to perform calculations based on time and dates.	GETDATE, SYSDATETIME, GETUTCDATE, DATEADD, DATEDIFF, YEAR, MONTH, DAY
Mathematical function	Mathematical functions perform algebraic operations on numeric values.	RAND, ROUND, POWER, ABS, CEILING, FLOOR
System function	SQL Server provides system functions for returning metadata or configuration settings.	HOST_ID, HOST_NAME, ISNULL
String function	String functions are used for string inputs such as char and varchar. The output can be a string or a numeric value.	SUBSTRING, LEFT, RIGHT, LEN, DATALENGTH, REPLACE, REPLICATE, UPPER, LOWER, RTRIM, LTRIM

Scalar Functions

Instructions to the Trainer(s):

- Using Slide 17, discuss the function type and its description in detail.
- Different types of functions are as follows:
 - **Conversion function:** Used to transform a value of one data type to another.
 - **Date and Time function:** Used to manipulate date and time values. They are useful to perform calculations based on time and dates.
 - **Mathematical functions:** They perform algebraic operations on numeric values.
 - **System function:** Returns metadata or configuration settings.
 - **String function:** Used for string inputs such as char and varchar.

Variables

A variable is an object that can hold a data value. In Transact-SQL, variables can be classified into local and global variables.

- In Transact-SQL, local variables are created and used for temporary storage while SQL statements are executed.
- Data can be passed to SQL statements using local variables.
- The name of a local variable must be prefixed with '@' sign.

For example,

```
DECLARE @SearchWord NVARCHAR(30)  
SET @SearchWord = N'performance'
```

Instructions to the Trainer(s):

- Using Slide 18, explain variables.
- A variable is an object that can hold a data value. In Transact-SQL, variables can be classified into local and global variables.
- In Transact-SQL, local variables are created and used for temporary storage while SQL statements are executed. Data can be passed to SQL statements using local variables. The name of a local variable must be prefixed with '@' sign.
- Global variables are in-built variables that are defined and maintained by the system.
- Global variables in SQL Server are prefixed with two '@' signs. The value of any of these variables can be retrieved with a simple SELECT query.

Expressions

An expression is a combination of identifiers, values, and operators that SQL Server can evaluate in order to obtain a result.

Expressions can be used in several different places when accessing or changing data.

SalesOrderID	CustomerID	SalesPersonID	TerritoryID	CurrentYear	NextYear
1 43659	29625	279	5	2011	2012
2 43660	29672	279	5	2011	2012
3 43661	29734	282	6	2011	2012
4 43662	29994	282	6	2011	2012
5 43663	29565	276	4	2011	2012
6 43664	29698	280	1	2011	2012
7 43665	29580	283	1	2011	2012
8 43666	30052	276	4	2011	2012
9 43667	29974	277	3	2011	2012
10 43668	29614	282	6	2011	2012
11 43669	29747	283	1	2011	2012

Expression Result

© Aptech Ltd.

Session 4 / 19

Instructions to the Trainer(s):

- Using Slide 19, explain expressions in Transact-SQL.
- An expression is a combination of identifiers, values, and operators that SQL Server can evaluate in order to obtain a result.
- Expressions can be used in several different places when accessing or changing data.

Control-of-Flow, Errors, and Transactions

Although Transact-SQL is primarily a data retrieval language, it supports control-of-flow statements for executing and finding errors.

Control-of-flow language determines the execution flow of

- Transact-SQL statements
- statement blocks
- user-defined functions
- and stored procedures

Control-of-Flow Statement	Description
IF. . . ELSE	Provides branching control based on a logical test.
WHILE	Repeats a statement or a block of statements as long as the condition is true.
BEGIN. . . END	Defines the scope of a block of Transact-SQL statements.
TRY. . . CATCH	Defines the structure for exception and error handling.
BEGIN TRANSACTION	Marks a block of statements as part of an explicit transaction.

Control-of-Flow Statements

Instructions to the Trainer(s):

- Using Slide 20, explain the control-of-flow, errors, and transactions.
- Although Transact-SQL is primarily a data retrieval language, it supports control-of-flow statements for executing and finding errors.
- Control-of-flow language determines the execution flow of Transact-SQL statements, statement blocks, user-defined functions, and stored procedures.

Comments

Comments are descriptive text strings, also known as remarks, in program code that will be ignored by the compiler.

Can be included inside the source code of a single statement, a batch, or a stored procedure.

SQL Server supports two types of commenting styles:

- (double hyphens)
- /* ... */ (forward slash-asterisk character pairs)

A complete line of code or part of a code can be marked as a comment, if two hyphens (- -) are placed at the beginning. The remainder of the line becomes a comment.

These comment characters can be used on the same line as code to be executed, on lines by themselves, or even within executable code. Everything between /* to */ is considered part of the comment. For a multiple-line comment, the open-comment character pair /* must begin the comment, and the close-comment character pair */ must end the comment.

© Aptech Ltd. Session 4 / 21

Instructions to the Trainer(s):

- Using Slide 21, explain comments in Transact-SQL.
- Comments are descriptive text strings, also known as remarks, in program code that will be ignored by the compiler.
- Comments can be included inside the source code of a single statement, a batch, or a stored procedure.
- Comments explain the purpose of the program, special execution conditions, and provide revision history information.
- Microsoft SQL Server supports two types of commenting styles:

- (double hyphens): A complete line of code or a part of a code can be marked as a comment, if two hyphens (- -) are placed at the beginning. The remainder of the line becomes a comment. Explain Code Snippet displays the use of this style of comment.

/* ... */ (forward slash-asterisk character pairs): These comment characters can be used on the same line as code to be executed, on lines by themselves, or even within executable code. Everything in the line beginning from the open comment pair /* to the close comment pair */ is considered part of the comment. For a multiple-line comment, the open-comment character pair /* must begin the comment, and the close-comment character pair */ must end the comment. Explain Code Snippet which displays the use of this style of comment.

Batch Separators

A batch is a collection of one or more Transact-SQL statements sent at one time from an application to SQL Server for execution.

The Transact-SQL statements in a batch are compiled into a single executable unit, called an execution plan.

The process wherein a set of commands are processed one at a time from a batch of commands is called batch processing.

A batch separator is handled by SQL Server client tools such as SSMS to execute commands.

Instructions to the Trainer(s):

- Using Slide 22, discuss on batch separators.
- A batch is a collection of one or more Transact-SQL statements sent at one time from an application to SQL Server for execution.
- The Transact-SQL statements in a batch are compiled into a single executable unit, called an execution plan.
- The statements in the execution plan are then executed one at a time.
- The process wherein a set of commands are processed one at a time from a batch of commands is called batch processing.
- A batch separator is handled by SQL Server client tools such as SSMS to execute commands. For example, you must specify GO as a batch separator in SSMS.

Sets and Predicate Logic

- Sets and Predicate Logic are the two mathematical fundamentals that are used in SQL Server 2019.
- Both these theories are used in querying of data in SQL Server 2019.

Instructions to the Trainer(s):

- Using Slide 23, explain the sets and predicate logic in Transact-SQL.
- Sets and Predicate Logic are the two mathematical fundamentals that are used in SQL Server. Both these theories are used in querying of data in SQL Server 2019.
- Set theory is a mathematical foundation used in relational database model.
- A set is a collection of distinct objects considered as a whole.
- Predicate logic is a mathematical framework that consists of logical tests that gives a result.

Set Theory

Set theory is a mathematical foundation used in relational database model. A set is a collection of distinct objects considered as a whole.

For example, all the employees under an Employee table can be considered as one set. Employees are different objects that form a part of the set in the Employee table.

Set Theory Applications	Application in SQL Server Queries
Act on the whole set at once.	Query the whole table at once.
Use declarative, set-based processing.	Use attributes in SQL Server to retrieve specific data.
Elements in the set must be unique.	Define unique keys in the table.
No sorting instructions.	The results of querying are not retrieved in any order.

Instructions to the Trainer(s):

- Using Slide 24, discuss the set theory in SQL Server 2019.
- Set theory is a mathematical foundation used in relational database model.
- A set is a collection of distinct objects considered as a whole.

- Set theory applications are:
 - Act on the whole set at once
 - Use declarative, set-based processing
 - Elements in the set must be unique
 - No sorting instructions

- Applications in SQL Server Queries:
 - Use attributes in SQL Server to retrieve specific data
 - Define unique keys in the table
 - Results of querying are not retrieved in any order

Predicate Logic

Predicate logic is a mathematical framework that consists of logical tests that gives a result. The results are always displayed as either true or false.

Some of the applications of predicate logic in Transact-SQL are as follows:

Enforcing data integrity using the CHECK constraint

Control-of-flow using the IF statement

Joining tables using the ON filter

Filtering data in queries using the WHERE and HAVING clause

Providing conditional logic to CASE expressions

Defining subqueries

Instructions to the Trainer(s):

- Using Slide 25, explain the predicate logic.
- Predicate logic is a mathematical framework that consists of logical tests that gives a result. The results are always displayed as either true or false.
- In Transact-SQL, expressions such as WHERE and CASE expressions are based on predicate logic.
- Predicate logic is also used in other situations in Transact-SQL.
- Some of the applications of predicate logic in Transact-SQL are:
 - Enforcing data integrity using the CHECK constraint
 - Control-of-flow using the IF statement
 - Joining tables using the ON filter
 - Filtering data in queries using the WHERE and HAVING clause
 - Providing conditional logic to CASE expressions
 - Defining subqueries

Logical Order of Operators in SELECT Statement 1-3

Along with syntax of different SQL Server elements, an SQL Server user must also know the process of how the entire query is executed.

This process is a logical process that breaks the query and executes the query according to a predefined sequence in SQL Server 2019.

Element	Description
<code>SELECT <select list></code>	Defines the columns to be returned
<code>FROM <table source></code>	Defines the table to be queried
<code>WHERE <search condition></code>	Filters the rows by using predicates
<code>GROUP BY <group by list></code>	Arranges the rows by groups
<code>HAVING <search condition></code>	Filters the groups using predicates
<code>ORDER BY <order by list></code>	Sorts the output

Elements of SELECT Statement

© Aptech Ltd. Session 4 / 26

Instructions to the Trainer(s):

- Using Slide 26, explain the logical order of operators in select statement.
- Along with the syntax of different SQL Server elements, an SQL Server user must also know the process of how the entire query is executed.
- This process is a logical process that breaks the query and executes the query according to a predefined sequence in SQL Server.
- The SELECT statement is a query that will be used to explain the logical process of query execution. Different elements are:
 - `SELECT <select list>`: Defines the columns to be retrieved
 - `FROM <table source>`: Defines the table to be queried
 - `WHERE <search condition>`: Filters the rows by using predicates
 - `GROUP BY <group by list>`: Arranges the rows by groups
 - `HAVING <search condition>`: Filters the groups using predicates
 - `ORDER BY <order by list>`: Sorts the output

Logical Order of Operators in SELECT Statement 2-3

```
USE AdventureWorks2019
SELECT SalesPersonID, YEAR(OrderDate) AS OrderYear FROM
Sales.SalesOrderHeader
WHERE CustomerID = 30084
GROUP BY SalesPersonID, YEAR(OrderDate)
HAVING COUNT(*) > 1
ORDER BY SalesPersonID, OrderYear;
```

In the example, the order in which SQL Server will execute the SELECT statement is as follows:

1. First, the FROM clause is evaluated to define the source table that will be queried.
2. Next, the WHERE clause is evaluated to filter the rows in the source table. This filtering is defined by the predicate mentioned in the WHERE clause.
3. After this, the GROUP BY clause is evaluated. This clause arranges the filtered values received from the WHERE clause.
4. Next, the HAVING clause is evaluated based on the predicate that is provided.
5. Next, the SELECT clause is executed to determine the columns that will appear in the query results.
6. Finally, the ORDER BY statement is evaluated to display the output.

© Aptech Ltd. Session 4 / 27

Instructions to the Trainer(s):

- Using Slide 27, explain the logical order of operators in select statement.
- In the example, the order in which SQL Server will execute the SELECT statement is as follows:
 - First, the FROM clause is evaluated to define the source table that will be queried.
 - Next, the WHERE clause is evaluated to filter the rows in the source table. This filtering is defined by the predicate mentioned in the WHERE clause.
 - After this, the GROUP BY clause is evaluated. This clause arranges the filtered values received from the WHERE clause.
 - Next, the HAVING clause is evaluated based on the predicate that is provided. Next, the SELECT clause is executed to determine the columns that will appear in the query results.
 - Finally, the ORDER BY statement is evaluated to display the output.

Logical Order of Operators in SELECT Statement 3-3

The order of execution for the SELECT statement would be as follows:

```
5. SELECT SalesPersonID, YEAR(OrderDate) AS OrderYear  
1. FROM SalesOrderHeader  
2. WHERE CustomerID = 30084  
3. GROUP BY SalesPersonID, YEAR(OrderDate)  
4. HAVING COUNT(*) > 1  
6. ORDER BY SalesPersonID, OrderYear;
```

Results

	SalesPersonID	OrderYear
1	279	2011
2	279	2013

SELECT Statement Result

© Aptech Ltd. Session 4 / 28

Instructions to the Trainer(s):

- Using Slide 28, explain the order in which SQL Server will execute the SELECT statement.
- The order of execution for the SELECT statement in Code Snippet would be as follows:

```
FROM SalesOrderHeader  
WHERE CustomerID = 30084  
GROUP BY SalesPersonID, YEAR(OrderDate)  
HAVING COUNT(*) > 1  
ORDER BY SalesPersonID, OrderYear;
```

Finally, the ORDER BY statement is evaluated to display the output.

Summary

- Transact-SQL is a powerful language which offers features such as data types, temporary objects, and extended stored procedures.
- SQL Server supports three types of Transact-SQL statements, namely, DDL, DML, and DCL.
- A data type is an attribute defining the type of data that an object can contain.
- The Transact-SQL language elements includes predicates, operators, functions, variables, expressions, control-of-flow, errors, and transactions, comments, and batch separators.
- Sets and Predicate Logic are the two mathematical fundamentals that are used in SQL Server 2019.
- Set theory is a mathematical foundation used in relational database model, where a set is a collection of distinct objects considered as a whole.
- Predicate logic is a mathematical framework that consists of logical tests that gives a result.

Instructions to the Trainer(s):

- Show students Slide 29.
- Summarize the session by reading out each point on the slide.

Session 5: Creating and Managing Databases

5.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. Prepare a question or two that will be a key point to relate the current session objectives.

5.1.1 Teaching Skills

To teach this session, you should be well versed with creating and managing databases in SQL Server 2019.

You should teach the concepts in the theory class using the images provided. For teaching in the class, you are expected to use slides and LCD projectors.

In-Class Activities

Follow the order given here during In-Class activities.

Slide 2

Objectives

- Explain about modification of system data
- Describe adding of filegroups and transaction logs
- Outline the process to create a database
- Describe how to drop a database
- Explain database snapshots

Instructions to the Trainer(s):

Give students a brief overview of the current session through the session objectives listed in Slide 2.

5.2 In-Class Explanations

Slide 3

Modifying System Data

Users can avail administrative tools to fully administer system and manage all users and database objects:

SSMS Administration utilities: Is the core administrative console for SQL Server installations. It enables to perform high-level administrative functions, schedule routine maintenance tasks, and so forth.

SQL Server Management Objects (SQL-SMO) API: Includes complete functionality for administering SQL Server in applications.

Transact-SQL scripts and stored procedures: Use system stored procedures and Transact-SQL DDL statements.

Transact-SQL Query Window

AddressID	AddressLine1	AddressLine2	District	City	StateProvinceID	PostalCode	ModifiedDate
1	123 Main St.		Seattle	WA	10	98101	2008-05-12 12:00:00.000
2	456 Elm St.		Seattle	WA	10	98102	2008-05-12 12:00:00.000
3	789 Oak St.		Seattle	WA	10	98103	2008-05-12 12:00:00.000
4	1234 Pine St.		Seattle	WA	10	98104	2008-05-12 12:00:00.000
5	5678 Birch St.		Seattle	WA	10	98105	2008-05-12 12:00:00.000
6	9012 Fir St.		Seattle	WA	10	98106	2008-05-12 12:00:00.000
7	123 Main St.		Seattle	WA	10	98101	2008-05-12 12:00:00.000
8	456 Elm St.		Seattle	WA	10	98102	2008-05-12 12:00:00.000
9	789 Oak St.		Seattle	WA	10	98103	2008-05-12 12:00:00.000
10	1234 Pine St.		Seattle	WA	10	98104	2008-05-12 12:00:00.000
11	5678 Birch St.		Seattle	WA	10	98105	2008-05-12 12:00:00.000
12	9012 Fir St.		Seattle	WA	10	98106	2008-05-12 12:00:00.000
13	123 Main St.		Seattle	WA	10	98101	2008-05-12 12:00:00.000
14	456 Elm St.		Seattle	WA	10	98102	2008-05-12 12:00:00.000
15	789 Oak St.		Seattle	WA	10	98103	2008-05-12 12:00:00.000
16	1234 Pine St.		Seattle	WA	10	98104	2008-05-12 12:00:00.000
17	5678 Birch St.		Seattle	WA	10	98105	2008-05-12 12:00:00.000
18	9012 Fir St.		Seattle	WA	10	98106	2008-05-12 12:00:00.000

© Aptech Ltd.

Session 3/3

Instructions to the Trainer(s):

- Introduce the session using Slide 3, explain how to modify system data.
- Users are not allowed to directly update the information in system database objects, such as system tables, system stored procedures, and catalog views.
- However, users can avail a complete set of administrative tools allowing them to fully administer the system and manage all users and database objects. These are as follows:
 - **Administration utilities:** From SQL Server 2005 onwards, several SQL Server administrative utilities are integrated into SSMS. It is the core administrative console for SQL Server installations. It enables to perform high-level administrative functions, schedule routine maintenance tasks, and so forth. Figure shows the SQL Server 2012 Management Studio window.
 - **SQL Server Management Objects (SQL-SMO) API:** Includes complete functionality for administering SQL Server in applications.
 - **Transact-SQL scripts and stored procedures:** These use systems stored procedures and Transact-SQL DDL statements. Figure shows a Transact-SQL query window.

Viewing System Database Data

Database applications can determine catalog and system information by using any of these approaches:

System Catalog Views	Views displaying metadata for describing database objects in an SQL Server instance.
SQL-SMO	New managed code object model, providing a set of objects used for managing Microsoft SQL Server.
Catalog Functions, Methods, Attributes, or Properties of Data API	Used in ActiveX Data Objects (ADO), OLE DB, or ODBC applications.
Stored Procedures and Functions	Used in Transact-SQL as stored procedures and built-in functions.

© Aptech Ltd. Session 5/ 4

Instructions to the Trainer(s):

- Using Slide 4, explain methods to view system database data.
- Database applications can determine catalog and system information by using any of these approaches:
 - **System catalog views:** Views displaying metadata for describing database objects in an SQL Server instance.
 - **SQL-SMO:** New managed code object model, providing a set of objects used for managing Microsoft SQL Server.
 - **Catalog functions, methods, attributes, or properties of the data API:** Used in ActiveX Data Objects (ADO), OLE DB, or ODBC applications.
 - **Stored Procedures and Functions:** Used in Transact-SQL as stored procedures and built-in functions.

User-defined Databases

Using SQL Server 2019

- Users can create their own databases, also called user-defined databases, and work with them.
- The purpose of these databases is to store user data.

© Aptech Ltd. Session 5/5

Instructions to the Trainer(s):

- Using Slide 5, explain the User-defined Databases.
- Using SQL Server 2019:
 - Allows users to create their own databases, also called user-defined databases, and work with them.
 - The main purpose of this database is to store data.

Creating Databases Using Transact-SQL 1-3

To create a user-defined database, the information required is as follows:

- Name of the database
- Owner or creator of the database
- Size of the database
- Files and filegroups used to store it

Following is the syntax to create a user-defined database.

```
CREATE DATABASE DATABASE_NAME  
[ ON  
[ PRIMARY ] [ <filespec> [ ,...n ]  
[, <filegroup> [ ,...n ] ]  
[ LOGON { <filespec> [ ,...n ] } ]  
]  
[ COLLATE collation_name ]  
]  
[ ; ]
```

Instructions to the Trainer(s):

- Using Slide 6, discuss the concept of creating databases using Transact-SQL.
- To create a user-defined database, following information is required:
 - Name of the database
 - Owner or creator of database
 - Size of the database
 - Files and filegroups used to store it

Creating Databases Using Transact-SQL 2-3

SQL Server databases use two files - an .mdf file, known as the primary database file, containing the schema and data and a .ldf file, which contains the logs. A database may also use secondary database file, which normally uses an .ndf extension.

MDF stands for Master Database File.

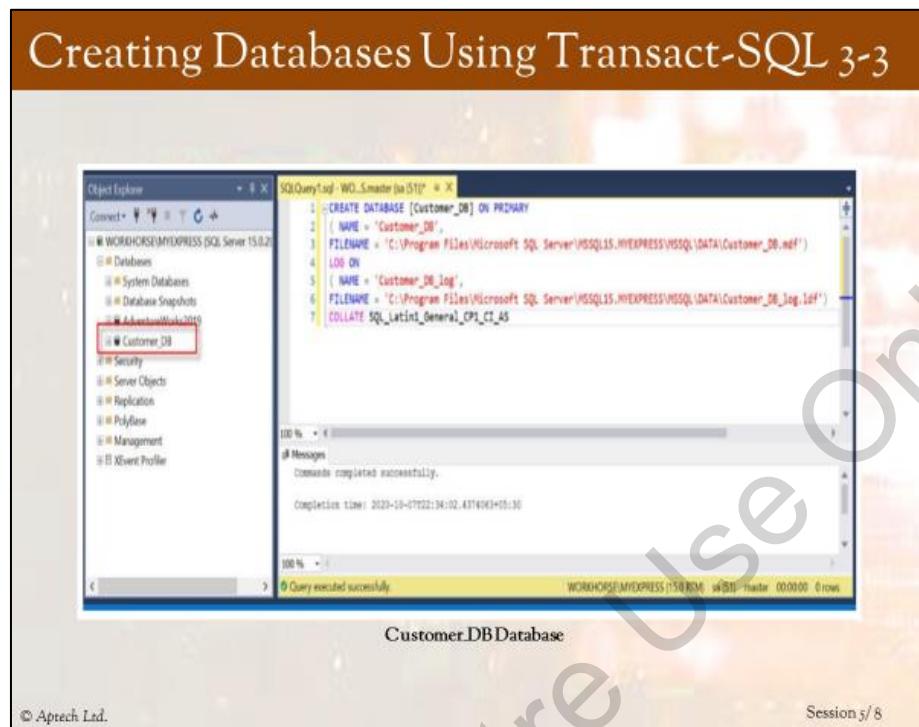
- It contains main information of a database that are part of the server.
- This extension also points to various other files.
- Plays a crucial role in information storage.

LDF stands for Log Database File.

- Stores information related to transaction logs for main data file.
- Keeps track of changes that have been made in the database.

Instructions to the Trainer(s):

- Using Slide 7, explain the files used in SQL Server database.
- SQL Server database stores two types of data files as Master Database Files and Log Database Files and abbreviated as MDF and LDF data files.
- A database may also use secondary database file, which normally uses an .ndf extension.
- MDF contains all the information of a database, this extension also points to various other files, plays a crucial role in information storage.
- LDF records all the transactions and changes to the database and keeps a track of the changes.



Instructions to the Trainer(s):

- Using Slide 8, explain how to create database.
- Explain Code Snippet which shows how to create a database with database file and transaction log file with collation name.
- After executing the code in Code Snippet, SQL Server 2019 displays the message 'Command(s) completed successfully'.
- The figure in slide 8 shows the database Customer_DB listed in the Object Explorer.

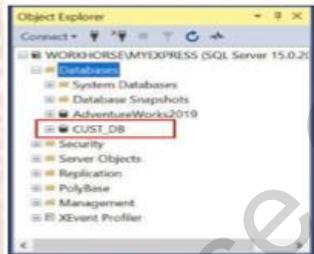
Modifying Databases

A user-defined database does the following:

- It grows or diminishes
- Its size expands or shrinks automatically or manually

```
ALTER DATABASE database_name
{
    <add_or_modify_files>
    | <add_or_modify_filegroups>
    | <set_database_options>
    | MODIFY NAME = new_database_name
    | COLLATE collation_name
}
[];
```

Syntax to modify a database



New Database Name CUST_DB

© Aptech Ltd. Session 5/ 9

Instructions to the Trainer(s):

- Using Slide 9, explain how to modify a database.
- As a user-defined database grows or diminishes, the database size will be expanded or be shrunk automatically or manually. Based on changing requirements from time to time, it may be found necessary to modify a database.
- Explain Code Snippet in Slide 9 which shows how to rename a database Customer_DB with a new database name, CUST_DB.
- Figure shows database Customer_DB is renamed with a new database name, CUST_DB.

In-Class Question:

Question: What keyword is used to modify a database?

Answer: Alter keyword is used to modify a database.

Ownership of Databases

In SQL Server 2019, ownership of a user-defined database can be changed.

- Ownership of system databases cannot be changed.
- The system procedure sp_changedbowner is used to change the ownership of a database.

```
sp_changedbowner [@loginname='login']
```

Instructions to the Trainer(s):

- Using Slide 10, explain ownership of database.
- In SQL Server 2019, the ownership of a user-defined database can be changed.
- Ownership of system databases cannot be changed.
- The system procedure sp_changedbowner is used to change the ownership of a database.
- After sp_changedbowner is executed, the new owner is known as the dbo user inside the selected database. The dbo receives permissions to perform all activities in the database. The owner of the master, model, or tempdb system databases cannot be changed.

For additional information, refer to:

<https://www.sqlshack.com/different-ways-to-change-database-owners-in-sql-server/>

Setting Database Options

Database-level options determine the characteristics of the database that can be set for each database

The options are unique to each database, so they do not affect other databases

Option Type	Description
Automatic options	Controls automatic behavior of database.
Cursor options	Controls cursor behavior.
Recovery options	Controls recovery models of database.
Miscellaneous options	Controls ANSI compliance.
State options	Controls state of database, such as online/offline and user connectivity.

Databases Options in SQL Server 2019

© Aptech Ltd. Session 5/ 11

Instructions to the Trainer(s):

- Using Slide 11, explain how to set database options.
- Database-level options determine the characteristics of the database that can be set for each database. These options are unique to each database, so they do not affect other databases. These database options are set to default values when a database is first created and can then, be changed by using the SET clause of the ALTER DATABASE statement.
- Different options are as follows:
 - **Automatic options:** Controls automatic behavior of database
 - **Cursor options:** Controls cursor behavior
 - **Recovery options:** Controls recovery models of database
 - **Miscellaneous options:** Controls ANSI compliance
 - **State options:** Controls state of database, such as online/offline and user connectivity

Filegroups 1-3

In SQL Server, data files are used to store database files

- For the sake of performance further subdivided into filegroups
- Used to group related files that together store a database object

Filegroup	Description
Primary	The filegroup that consists of the primary file. All system tables are placed inside the primary filegroup.
User-defined	Any filegroup that is created by the user at the time of creating or modifying databases.

Filegroups in SQL Server 2019

Instructions to the Trainer(s):

- Using Slide 12, explain the filegroups.
- In SQL Server, data files are used to store database files.
- The data files are further subdivided into filegroups for the sake of performance.
- Each filegroup is used to group related files that together store a database object. Every database has a primary filegroup by default. This filegroup contains the primary data file.
- The primary file group and data files are created automatically with default property values at the time of creation of the database.
- User-defined filegroups can then be created to group data files together for administrative, data allocation, and placement purposes.

Filegroups 2-3

Adding Filegroups to an existing database

- Filegroups can be created when the database is created for the first time or can be created later when more files are added to the database.
- However, files cannot be moved to a different filegroup after the files have been added to the database.

- A file cannot be a member of more than one filegroup at the same time.
- A maximum of 32,767 filegroups can be created for each database.
- Filegroups can contain only data files.
- Transaction log files cannot belong to a filegroup.

Name	Date modified
model_replicatedmaster.mdf	24-09-2019 15:09
modellog.ldf	07-10-2020 22:52
MS_AgentSigningCertificate.cer	28-09-2020 13:17
MSDBData.mdf	06-10-2020 22:29
MSDBLog.ldf	06-10-2020 22:31
SalesDB.mdf	07-10-2020 22:52
SalesDB_FG.ndf	07-10-2020 22:52
SalesDB_log.ldf	07-10-2020 22:52
tempdb.mdf	08-10-2020 13:44
templog.ldf	07-10-2020 22:52

Filegroup Added When Creating SalesDB Database

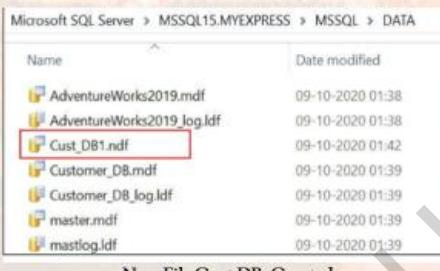
Instructions to the Trainer(s):

- Using Slide 13, explain how to add filegroups to an existing database.
- Filegroups can be created when the database is created for the first time or can be created later when more files are added to the database.
- However, files cannot be moved to a different filegroup after the files have been added to the database.
- A file cannot be a member of more than one filegroup at the same time. A maximum of 32,767 filegroups can be created for each database.
- Filegroups can contain only data files.
- Transaction log files cannot belong to a filegroup.

Filegroups 3-3

Default Filegroup

Objects are assigned to the default filegroup when they are created in the database. The PRIMARY filegroup is the default filegroup.



The screenshot shows the Microsoft SQL Server Management Studio interface. It is navigating through the path: Microsoft SQL Server > MSSQL15.MYEXPRESS > MSSQL > DATA. A list of files is displayed with the following details:

Name	Date modified
AdventureWorks2019.mdf	09-10-2020 01:38
AdventureWorks2019_log.ldf	09-10-2020 01:38
Cust_DB1.ndf	09-10-2020 01:42
Customer_DB.mdf	09-10-2020 01:39
Customer_DB_log.ldf	09-10-2020 01:39
master.mdf	09-10-2020 01:39
masterlog.ldf	09-10-2020 01:39

A message at the bottom of the list states "New File Cust.DB1 Created".

© Aptech Ltd. Session 5/ 14

Instructions to the Trainer(s):

- Using Slide 14, explain the default filegroup in SQL Server.
- Objects are assigned to the default filegroup when they are created in the database. The PRIMARY filegroup is the default filegroup.
- The default filegroup can be changed using the ALTER DATABASE statement.
- System objects and tables remain within the PRIMARY filegroup, but do not go into the new default filegroup.
- To make the FG_ReadOnly filegroup as default, it should contain at least one file inside it.

For more information on default filegroup, refer to:

<https://sqlstudies.com/2018/02/19/the-default-filegroup-and-why-you-should-care/>

Transaction Log 1-2

Recovery of individual transactions	An incomplete transaction is rolled back in case of an application issuing a ROLLBACK statement or the Database Engine detecting an error. The log records are used to roll back the modifications.
Recovery of all incomplete transactions when SQL Server is started	If a server that is running SQL Server fails, the databases may be left in an inconsistent state. When an instance of SQL Server is started, it runs a recovery of each database.
Rolling a restored database, file, filegroup, or page forward to the point of failure	The database can be restored to the point of failure after a hardware loss or disk failure affecting the database files.
Supporting transactional replication	The Log Reader Agent monitors the transaction log of each database configured for replications of transactions.
Supporting standby server solutions	The standby-server solutions, database mirroring, and log shipping depend on the transaction log.

© Aptech Ltd.

Session 5 / 15

Instructions to the Trainer(s):

- Using Slide 15, explain the concept of transaction log.
- A transaction log in SQL Server records all transactions and the database modifications made by each transaction.
- The transaction log is one of the critical components of the database. It can be the only source of recent data in case of system failure.
- The transaction logs support operations such as the following:
 - **Recovery of individual transactions:** An incomplete transaction is rolled back in case of an application issuing a ROLLBACK statement or the Database Engine detecting an error. The log records are used to roll back the modifications.
 - **Recovery of all incomplete transactions when SQL Server is started:** If a server that is running SQL Server fails, the databases may be left in an inconsistent state. When an instance of SQL Server is started, it runs a recovery of each database.
 - **Rolling a restored database, file, filegroup, or page forward to the point of failure:** The database can be restored to the point of failure after a hardware loss or disk failure affecting the database files.
 - **Supporting transactional replication:** The Log Reader Agent monitors the transaction log of each database configured for replications of transactions.
 - **Supporting standby server solutions:** The standby-server solutions, database mirroring, and log shipping depend on the transaction log.

Transaction Log 2-2

Working of Transaction Logs:

A database in SQL Server has at least one data file and one transaction log file. Data and transaction log information are kept separated, preferably on separate drives.

The rollback of each transaction is executed using following ways:

- A transaction is rolled forward when a transaction log is applied.
- A transaction is rolled back when an incomplete transaction is backed out.

Adding Log files to a database

```
ALTER DATABASE database_name
{
    ...
}
[+] <add_or_modify_files>::=
{
    ADD FILE <filespec> [....n]
    [ TO FILEGROUP {filegroup_name | DEFAULT} ]
    | ADD LOG FILE <filespec> [,...n]
    | REMOVE FILE logical_file_name
    | MODIFY FILE<filespec>
}
```

© Aptech Ltd.

Session 5/16

Instructions to the Trainer(s):

- Using Slide 16, explain the working of transaction logs and adding log files to a database.
- A database in SQL Server has at least one data file and one transaction log file.
- Data and transaction log information are kept separated, preferably on separate drives.
- The rollback of each transaction is executed using following ways:
 - A transaction is rolled forward when a transaction log is applied.
 - A transaction is rolled back when an incomplete transaction is backed out.

Creating Databases Using SSMS 1-2

When a database is created, data files should be as large as possible based on maximum amount of data, which is expected in the database.

New Database Option

New Database Window

© Aptech Ltd.

Session 5/17

Instructions to the Trainer(s):

- Using Slide 17, discuss the creation of database using SSMS.
- SQL Server Management Studio (SSMS) is an integrated environment which manages any SQL infrastructure.
- When a database is created, data files should be as large as possible based on maximum amount of data, which is expected in the database.
- Tell students that by using SSMS they can do the following:
 - Access, configure, manage, administer, and develop all components of SQL Server



Instructions to the Trainer(s):

- Using Slide 18, explain the recovery model how filegroup can be added in SSMS.
- A recovery model controls how transactions are logged and what type of operations can be performed.
- Filegroups can be defined as a process that are used for backup and management purposes.

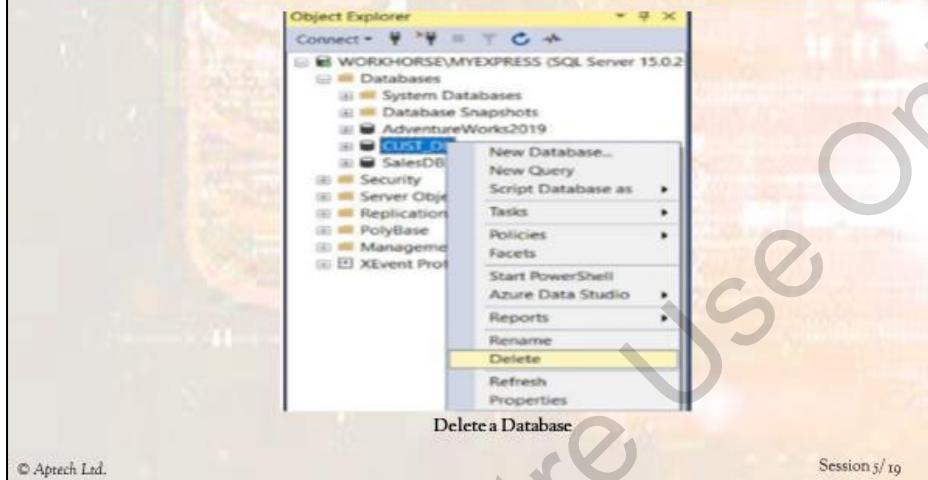
In-Class Question:

Question: What are the three types of recovery models?

Answer: SIMPLE, FULL, and BULK_LOGGED are the three types of recovery models.

Dropping a Database

A full backup of the database must be taken before dropping a database. A deleted database can be re-created only by restoring a backup.



Instructions to the Trainer(s):

- Using Slide 19, explain the procedure to drop a database.
- A full backup of the database must be taken before dropping a database.
- A deleted database can be re-created only by restoring a backup.
- The steps to delete or drop a database using SSMS are as follows:
 - In Object Explorer, connect to an instance of the SQL Server Database Engine, and then, expand that instance.
 - Expand Databases, right-click the database to delete, and then, click Delete, as shown in figure.
 - Confirm that the correct database is selected, and then, click OK.

Creating Database Snapshots

Database snapshot feature was first introduced in Microsoft SQL Server 2005

- It is a feature that provides a read-only, static view of a SQL Server database.
- If a user makes a mistake in a source database, the source database can be reverted to the previous state when the snapshot was created.

Advantages

- Provide a convenient, read-only copy of data.
- When queried, no deterioration of performance.
- Snapshot files are small and are very quick to create.

Disadvantages

- Snapshot backup cannot be created.
- Snapshot must exist on the same database server as that of the source database.
- A new user cannot be granted access to the data in a snapshot.

Instructions to the Trainer(s):

- Using Slide 20, explain how to create database snapshots.
- A database snapshot feature was introduced in Microsoft SQL Server 2005.
- While database snapshots provide a read-only, static view of SQL Server database.
- If a user makes a mistake in a source database, the source database can be reverted to the previous state when the snapshot was created.
- The advantage of using the database snapshots are:
 - Provide a convenient, read-only copy of data
 - When queried, no deterioration of performance
 - Snapshot files are small and very quick to create
- The disadvantages are:
 - Snapshot backups cannot be created
 - Exist on the same database server
 - New user is not granted the access to the data in a snapshot

Summary

- SQL Server uses system databases to support different components of the DBMS.
- The SQL Server data files are used to store database files, which are further subdivided into filegroups for the sake of performance.
- Databases can be created and dropped using SSMS or Transact-SQL.
- The CREATE DATABASE statement with various options can be used to create databases.
- ALTER DATABASE and DROP DATABASE are used to modify and delete a database respectively.
- A transaction log in SQL Server records all transactions and the database modifications made by each transaction.
- Objects are assigned to the default filegroup when they are created in the database. The PRIMARY filegroup is the default filegroup.
- A database snapshot is a read-only, static view of a SQL Server database.

Instructions to the Trainer(s):

- Show students Slide 21.
- Summarize the session by reading out each point on the slide.

Session 6: Creating Tables

6.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. Prepare a question or two that will be a key point to relate the current session objectives.

6.1.1 Teaching Skills

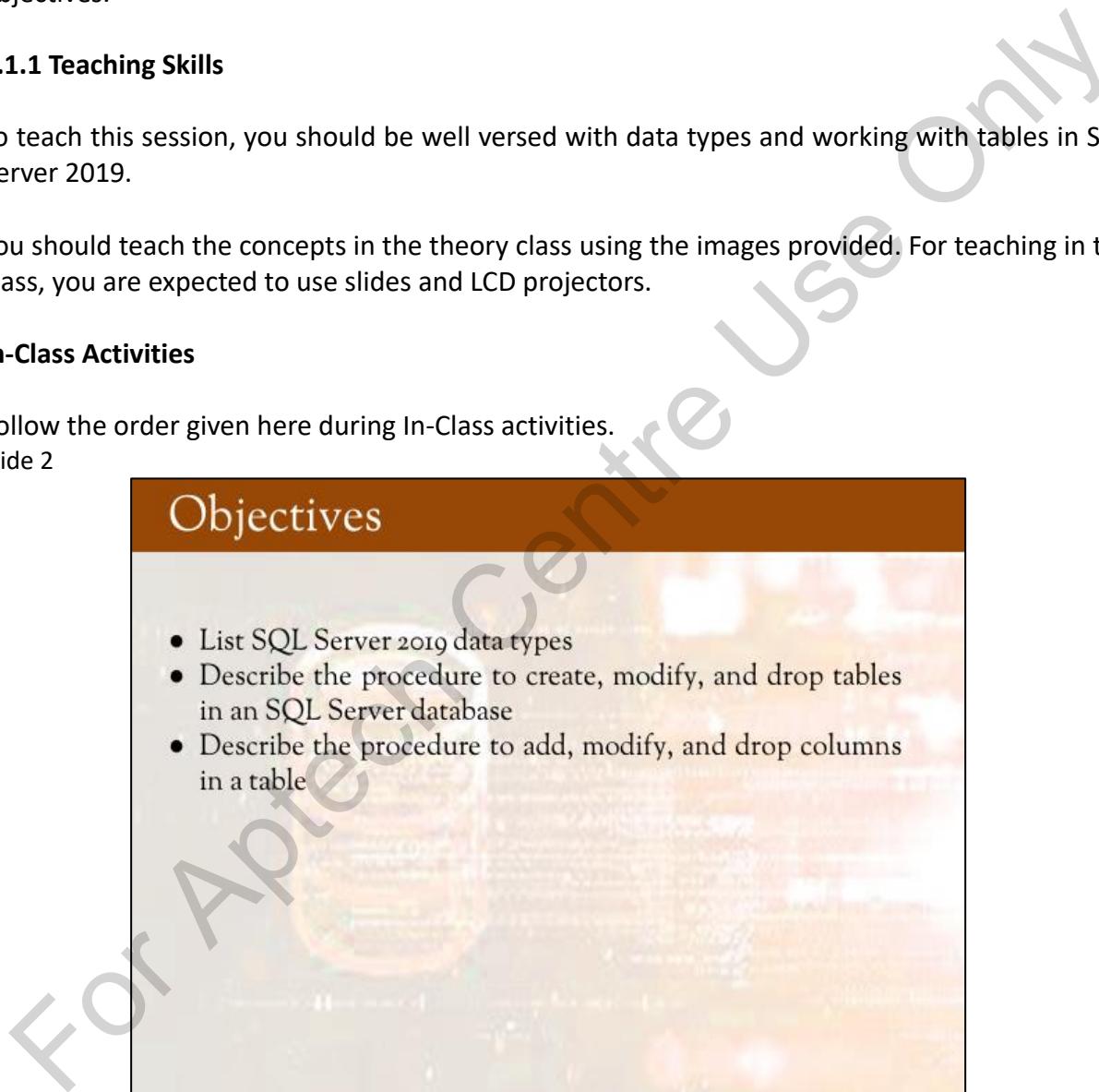
To teach this session, you should be well versed with data types and working with tables in SQL Server 2019.

You should teach the concepts in the theory class using the images provided. For teaching in the class, you are expected to use slides and LCD projectors.

In-Class Activities

Follow the order given here during In-Class activities.

Slide 2



Objectives

- List SQL Server 2019 data types
- Describe the procedure to create, modify, and drop tables in an SQL Server database
- Describe the procedure to add, modify, and drop columns in a table

© Aptech Ltd. Session 6/ 2

Instructions to the Trainer(s):

Give students a brief overview of the current session through the session objectives listed in Slide 2.

6.2 In-Class Explanations

Slide 3

Introduction

One of the most important types of database objects in SQL Server 2019 is a table.

Tables in SQL Server 2019 contain data in the form of rows and columns.

Each column may have data of a specific type and size.

© Aptech Ltd. Session 6 / 3

Instructions to the Trainer(s):

- Using Slide 3, explain what is a table.
- One of the most important types of database objects in SQL Server 2012 is a table.
- Tables in SQL Server 2012 contain data in the form of rows and columns.
- Each column may have data of a specific type and size.

For additional information, refer to:

<https://www.sqlshack.com/sql-server-table-structure-overview/>

<https://www.geeksforgeeks.org/introduction-of-ms-sql-server/>

Name	Description
hierarchyid	It is a system data type with variable length. You can use it to represent a position in a hierarchy.
geometry	It is a spatial data type, implemented as a Common Language Runtime (CLR) data type in SQL Server. It represents data in a Euclidean (flat) coordinate system. SQL Server supports a set of methods for this data type. The geometry type is predefined and available in each database. You can create table columns of type geometry and operate on geometry data similar to how you do on other CLR types.
geography	It is a spatial type for storing ellipsoidal (round-earth) data, such as GPS latitude and longitude coordinates. SQL Server supports a set of methods for the geography spatial data type. This type too is predefined and available in each database. You can create table columns of type geography.
xml	It is a special data type for storing XML data in SQL Server tables.
cursor	It is a data type for variables or stored procedure OUTPUT parameters that contain a reference to a cursor.
table	It is a special data type useful for storing result set temporarily in a table-valued function. You can use data from this for processing later. It can be used in functions, stored procedures, and batches
rowversion	It returns automatically generated, unique binary numbers within a database.

Advanced Data Types in SQL Server

© Aptech Ltd.

Session 6/ 4

Instructions to the Trainer(s):

- Using Slide 4, explain different advanced data types in SQL Server.
- A data type is an attribute that specifies the type of data an object can hold, such as numeric data, character data, monetary data, and so on.
- A data type also specifies the storage capacity of an object. Once a column has been defined to store data belonging to a particular data type, data of another type cannot be stored in it. In this manner, data types enforce data integrity.
- Different types of advanced data types in SQL Server are as follows:
 - hierarchyid: It is a system data type with variable length
 - geometry: Implemented as Common Language Runtime (CLR) data type in SQL Server
 - geography: Used for storing GPS Latitude and longitude coordinates
 - xml: used for storing XML data in SQL Server
 - cursor: Is a data type for variables or stored procedure
 - table: Used for storing result set temporarily inn a table-valued function

Creating, Modifying, and Dropping Tables

Most tables have a primary key, made up of one or more columns of the table

A primary key is always unique

The Database Engine will enforce the restriction that any primary key value cannot be repeated in the table

Thus, the primary key can be used to identify each record uniquely

© Aptech Ltd.

Session 6/5

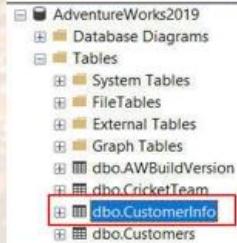
Instructions to the Trainer(s):

- Using Slide 5, explain the concept of creating, modifying, and dropping tables in a database.
- A Table is an object which stores data in Row & Column format.
- Most tables have a primary key, made up of one or more columns of the table.
- A primary key is always unique. The Database Engine will enforce the restriction that any primary key value cannot be repeated in the table. Thus, the primary key can be used to identify each record uniquely.

Creating Tables

The CREATE TABLE statement is used to create tables in SQL Server 2019. A simple basic syntax is as follows:

```
CREATE TABLE [database_name.] [schema_name.]table_name  
([<column_name>] [data_type] Null/Not Null,  
ON [filegroup | "default"]
```



Creating a Table

Instructions to the Trainer(s):

- Using Slide 6, discuss the concept of creating tables.
- In order to create a table, the first step to store data in the database. Post the creation of table, we can keep inserting the rows in the table.
- The CREATE TABLE statement is used to create tables in SQL Server 2019.

Modifying Tables

The ALTER TABLE statement is used to modify a table definition by altering, adding, or dropping columns and constraints, reassigning partitions, or disabling or enabling constraints and triggers.

Syntax:

```
ALTERTABLE [[database_name.] [schema_name].] schema_name.table_name  
    | ALTERCOLUMN ([<column_name>] [data_type] Null/NotNull,);  
    | ADD ([<column_name>] [data_type] Null/NotNull,);  
    | DROPCOLUMN ([<column_name>]);
```

Instructions to the Trainer(s):

- Using Slide 7, explain how tables can be modified.
- The ALTER TABLE statement is used to modify a table definition by altering, adding, or dropping columns and constraints, reassigning partitions, or disabling or enabling constraints and triggers.
- Before attempting to drop columns, however, it is important to ensure that the columns can be dropped. Under certain conditions, columns cannot be dropped, such as if they are used in a CHECK, FOREIGN KEY, UNIQUE, or PRIMARY KEY constraint, associated with a DEFAULT definition, and so forth.

Dropping Tables

The DROP TABLE statement removes a table definition, its data, and all associated objects such as indexes, triggers, constraints, and permission specifications for that table.

Syntax:

```
DROP TABLE <Table Name>
```

© Aptech Ltd.

Session 6/8

Instructions to the Trainer(s):

- Using Slide 8, explain how dropping a table can be done in a database.
- The DROP TABLE statement removes a table definition, its data, and all associated objects such as indexes, triggers, constraints, and permission specifications for that table.
- We delete the table when it is not required anymore.

Data Modification Statements 1-2

The statements used for modifying data are INSERT, UPDATE, and DELETE statements.

These are explained as follows:

- **INSERT Statement** - The INSERT statement adds a new row to a table.

```
INSERT [INTO] <Table Name> VALUES <values>
```

Syntax for INSERT Statement

© Aptech Ltd.

Session 6/9

Instructions to the Trainer(s):

- Using Slide 9, explain the concept of data modification statements.
- The statements you use to add, change, or delete data are called data modification statements.
- The statements used for modifying data are INSERT, UPDATE, and DELETE statements. These are explained as follows:

INSERT Statement – The INSERT statement adds a new row to a table. The outcome of this will be that one row with the given data is inserted into the table.

In-Class Question:

Question: What is data modification attack?

Answer: Modification Data Attacks (MDA) can be malicious and cause huge damages to a system. MDA happens when attackers interrupt, capture, modify, steal, or delete important information in the system via network access or direct access using executable codes.

Data Modification Statements 2-2

- **UPDATE Statement** - The UPDATE statement modifies the data in the table.

```
UPDATE <Table_Name>
SET <Column_Name = Value>
[WHERE <Search condition>]
```

Syntax for UPDATE Statement
- **DELETE Statement** - The DELETE statement removes rows from a table.

```
DELETE FROM <Table_Name> [WHERE <Search condition>]
```

Syntax for DELETE Statement

© Aptech Ltd. Session 6 / 10

Instructions to the Trainer(s):

- Using Slide 10, explain the Update and Delete Statement in Data Modification Statements.
- UPDATE Statement** - The UPDATE statement modifies the data in the table.
DELETE Statement - The DELETE statement removes rows from a table.

For additional information, refer to:

<https://swcarpentry.github.io/sql-novice-survey/09-create/index.html>

Column Nullability

The nullability feature of a column determines whether rows in the table can contain a null value for that column. In SQL Server, a null value is not same as zero, blank, or a zero length character string (such as '')

For example:

A null value in Color column of Production.Product table of AdventureWorks2019 database does not mean that the product has no color; it just means that color for the product is unknown or has not been set.

Nullability of a column can be defined either when creating a table or modifying a table. The NULL keyword is used to indicate that null values are allowed in the column and NOT NULL is used to indicate that null values are not allowed.

Instructions to the Trainer(s):

- Using Slide 11, explain the concept of column nullability.
- The nullability feature of a column determines whether rows in the table can contain a null value for that column. In SQL Server, a null value is not same as zero, blank, or a zero-length character string (such as '').
- For example, a null value in the Color column of the Production.Product table of the AdventureWorks2019 database does not mean that the product has no color; it just means that the color for the product is unknown or has not been set.
- Nullability of a column can be defined either when creating a table or modifying a table.
- For writable tables, any column that is going to store an SQL NULL value must have a null condition-name. If you have a table with multiple record types, it is possible to mark columns as nullable without an underlying null condition-name.
- The NULL keyword is used to indicate that null values are allowed in the column and the NOT NULL keywords are used to indicate that null values are not allowed.
- When inserting a row, if no value is given for a nullable column (that is, it allows null values), then, SQL Server automatically gives it a null value unless the column has been given a default definition. It is also possible to explicitly enter a null value into a column regardless of what data type it is or whether it has a default associated with it. Making a column non-nullable (that is, not permitting null values) enforces data integrity by ensuring that the column contains data in every row.

DEFAULT Definition I-2

A DEFAULT definition can be given for the column to assign it as a default value if no value is given at the time of creation.

For example:

It is common to specify zero as the default for numeric columns or 'N/A' or 'Unknown' as the default for string columns when no value is specified.

Demonstrating Use of DEFAULT			
	ProductID	Name	Price
1	111	Rivets	100.00

© Aptech Ltd.

Session 6 / 12

Instructions to the Trainer(s):

- Using Slide 12, explain the default definition concept.
- All values for the product details may not be known even at the time of data insertion. However, as per data consistency and integrity rules, the columns in a record should typically contain a value.
- Storing null values into such columns where the exact value of data is not known may not be desirable or practical.
- In such situations, a DEFAULT definition can be given for the column to assign it as a default value if no value is given at the time of creation.

For example, it is common to specify zero as the default for numeric columns or 'N/A' or 'Unknown' as the default for string columns when no value is specified.

DEFAULT Definition 2-2

Following cannot be created on columns with DEFAULT definitions:

A timestamp data type

An IDENTITY or ROWGUIDCOL property

An existing default definition or default object

Instructions to the Trainer(s):

- Using Slide 13, explain the default definition concept.
- A DEFAULT definition for a column can be created at the time of table creation or added at a later stage to an existing table.
- When a DEFAULT definition is added to an existing column in a table, SQL Server applies the new default values only to those rows of data, which have been newly added to the table.
- Following cannot be created on columns with DEFAULT definitions:
 - A timestamp data type
 - An IDENTITY or ROWGUIDCOL property
 - An existing default definition or default object

IDENTITY Property 1-3

- The IDENTITY property of SQL Server is used to create identifier columns that can contain auto-generated sequential values to uniquely identify each row within a table.

For example:

An identifier column could be created to generate unique student registration numbers automatically whenever new rows are inserted into the Students table.

- The identity number for the first row inserted into the table is called seed value.
- The increment, also called Identity Increment property, is added to the seed in order to generate further identity numbers in sequence.

Instructions to the Trainer(s):

- Using Slide 14, explain the IDENTITY Property concept.
- The IDENTITY property of SQL Server is used to create identifier columns that can contain auto-generated sequential values to uniquely identify each row within a table.
- For example, an identifier column could be created to generate unique student registration numbers automatically whenever new rows are inserted into the Students table.
- The identity number for the first row inserted into the table is called seed value and the increment, also called Identity Increment property, is added to the seed in order to generate further identity numbers in sequence.
- When a new row is inserted into a table with an identifier column, the next identity value is automatically generated by SQL Server by adding the increment to the seed.
- An identity column is often used for primary key values.

IDENTITY Property 2-3

A column having IDENTITY property must be defined using one of the following data types:

decimal, int, numeric, smallint, bigint, or tinyint

A column having IDENTITY property need not have a seed and increment value specified. If they are not specified, a default value of 1 will be used for both.

A table cannot have more than one column with IDENTITY property.

The identifier column in a table must not allow null values and must not contain a DEFAULT definition or object.

Columns defined with IDENTITY property cannot have their values updated.

The values can be explicitly inserted into the identity column of a table only if the IDENTITY_INSERT option is set ON. When IDENTITY_INSERT is ON, INSERT statements must supply a value.

Characteristics of the IDENTITY property

Instructions to the Trainer(s):

- Using Slide 15, explain the characteristics of identity property.
- The characteristics of the IDENTITY property are as follows:
 - A column having IDENTITY property must be defined using one of the following data types: decimal, int, numeric, smallint, bigint, or tinyint.
 - A column having IDENTITY property does not require to have a seed and increment value specified. If they are not specified, a default value of 1 will be used for both.
 - A table cannot have more than one column with IDENTITY property. The identifier column in a table must not allow null values and must not contain a DEFAULT definition or object.
 - Columns defined with IDENTITY property cannot have their values updated.
 - The values can be explicitly inserted into the identity column of a table only if the IDENTITY_INSERT option is set ON. When IDENTITY_INSERT is ON, INSERT statements must supply a value.

IDENTITY Property 3-3

The advantage of identifier columns is that SQL Server can automatically provide key values, thus reducing costs that would have been incurred for extra storage and improving performance.

	Person_ID	MobileNumber
1	500	983452201
2	501	993026654

IDENTITY Property Applied on Person.ID Column

© Aptech Ltd.

Session 6 / 16

Instructions to the Trainer(s):

- Using Slide 16, explain the advantages of identity property.
- The advantage of identifier columns is that SQL Server can automatically provide key values, thus reducing costs and improving performance.
- Once the IDENTITY property has been set, retrieving the value of the identifier column can be done by using the IDENTITYCOL keyword with the table name in a SELECT statement. To know if a table has an IDENTITY column, the OBJECTPROPERTY() function can be used. To retrieve the name of the IDENTITY column in a table, the COLUMNPROPERTY function is used.

Globally Unique Identifiers

In a networked environment, many tables may require to have a column consisting of a common globally unique value.

- Consider a scenario where data from multiple database systems such as banking databases must be consolidated at a single location.
- When the data from around the world is collated at the central site for consolidation and reporting, using globally unique values prevents customers in different countries from having the same bank account number or customer ID.

Results		Messages
	Person_ID	PersonName
1	362C4377-D194-4607-A466-7FFD2064EAFC	William Smith

Unique Identifier

Instructions to the Trainer(s):

- Using Slide 17, discuss the concept of globally unique identifiers.
- Often, in a networked environment, many tables may require to have a column consisting of a common globally unique value.
- Consider a scenario where data from multiple database systems such as banking databases must be consolidated at a single location. When the data from around the world is collated at the central site for consolidation and reporting, using globally unique values prevents customers in different countries from having the same bank account number or customer ID. To satisfy this requirement, SQL Server provides globally unique identifier columns. These can be created for each table containing values that are unique across all the computers in a network.
- Only one identifier column and one globally unique identifier column can be created for each table. To create and work with globally unique identifiers, a combination of ROWGUIDCOL, uniqueidentifier data type, and NEWID function are used.
- Values for a globally unique column are not automatically generated. One has to create a DEFAULT definition with a NEWID() function for a uniqueidentifier column to generate a globally unique value. The NEWID() function creates a unique identifier number which is a 16-byte binary string. The column can be referenced in a SELECT list by using the ROWGUIDCOL keyword.

Constraints 1-2

One of the important functions of SQL Server is to maintain and enforce data integrity.

- A constraint is a property assigned to a column or set of columns in a table to prevent certain types of inconsistent data values from being entered.
- Constraints are used to apply business logic rules and enforce data integrity.

Instructions to the Trainer(s):

- Using Slide 18, explain the concept of constraints.
- One of the important functions of SQL Server is to maintain and enforce data integrity. There are a number of means to achieve this but one of the commonly used and preferred methods is to use constraints.
- A constraint is a property assigned to a column or set of columns in a table to prevent certain types of inconsistent data values from being entered.
- Constraints are used to apply business logic rules and enforce data integrity.
- Constraints can be created when a table is created, as part of the table definition by using the CREATE TABLE statement or can be added at a later stage using the ALTER TABLE statement.

Constraints 2-2

Constraints can be categorized as column constraints and table constraints.

Column Constraint	Table Constraint
Is specified as part of a column definition and applies only to that column.	Can apply to more than one column in a table and is declared independently from a column definition. Table constraints must be used when more than one column is included in a constraint.

SQL Server supports the following types of constraints:



© Aptech Ltd.

Session 6/19

Instructions to the Trainer(s):

- Using Slide 19, explain the concept of Column and table constraint.
- Constraints can be categorized as
 - column constraints and table constraints
- A column constraint is specified as part of a column definition and applies only to that column.
- A table constraint can apply to more than one column in a table and is declared independently from a column definition. Table constraints must be used when more than one column is included in a constraint.
- SQL Server supports following types of constraints:
 - **NOT NULL** - Indicates that a column cannot store NULL value
 - **CHECK** - Ensures that the value in a column meets a specific condition
 - **UNIQUE** - Ensures that each row for a column must have a unique value
 - **FOREIGN KEY** - Ensures the referential integrity of the data in one table to match values in another table
 - **PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Ensures that a column (or combination of two or more columns) have a unique identity which helps to find a particular record in a table more easily and quickly

PRIMARY KEY

- A table typically has a primary key comprising a single column or combination of columns to uniquely identify each row within the table.
- The PRIMARY KEY constraint is used to create a primary key and enforce integrity of the entity in the table.

Messages
 (1 row affected)
 Msg 2627, Level 14, State 1, Line 1
 Violation of PRIMARY KEY constraint 'PK_EmpConta_7811348110185FA5'.
 Cannot insert duplicate key in object 'dbo.EmpContactPhone'.
 The duplicate key value is (101).
 The statement has been terminated.

Output Error Message for Duplicate EMP_ID

EMP_ID	MobileNumber	ServiceProvider	LandlineNumber
1	983345674	Verizon	NULL

Output of the Successfully Executed First INSERT Statement

Instructions to the Trainer(s):

- Using Slide 20, explain primary key concept.
- The PRIMARY KEY constraint is used to create a primary key and enforce integrity of the entity of the table. Only one primary key constraint can be created per table. Two rows in a table cannot have the same primary key value and a column that is a primary key cannot have NULL values.
- Hence, when a primary key constraint is added to existing columns of a table, SQL Server 2019 checks to see if the rules for primary key are complied with. If the existing data in the columns do not comply with the rules for primary key, then the constraint will not be added.

In-Class Question:

Question: What is the use of primary key?

Answer: A table typically has a primary key comprising a single column or combination of columns to uniquely identify each row within the table.

UNIQUE

A UNIQUE constraint is used to ensure that only unique values are entered in a column or set of columns. It allows developers to make sure that no duplicate values are entered.

- Primary keys are implicitly unique.
- Unique key constraints enforce entity integrity because once the constraints are applied; no two rows in the table can have the same value for the columns.
- UNIQUE constraints allow null values.
- A single table can have more than one UNIQUE constraint.

The screenshot shows a SQL Server Management Studio window. The 'Messages' tab displays an error message: '(1 row affected) Msg 2627, Level 14, State 1, Line 4 Violation of UNIQUE KEY constraint 'UQ__NewEmpCo__0A8D89724EBCC374'. Cannot insert duplicate key in object 'dbo.NewEmpContactPhone'. The duplicate key value is (NULL). The statement has been terminated.' Below it, the 'Output Error Message for Value Duplicate MobileNumber' tab shows a successful insert into the 'NewEmpContactPhone' table:

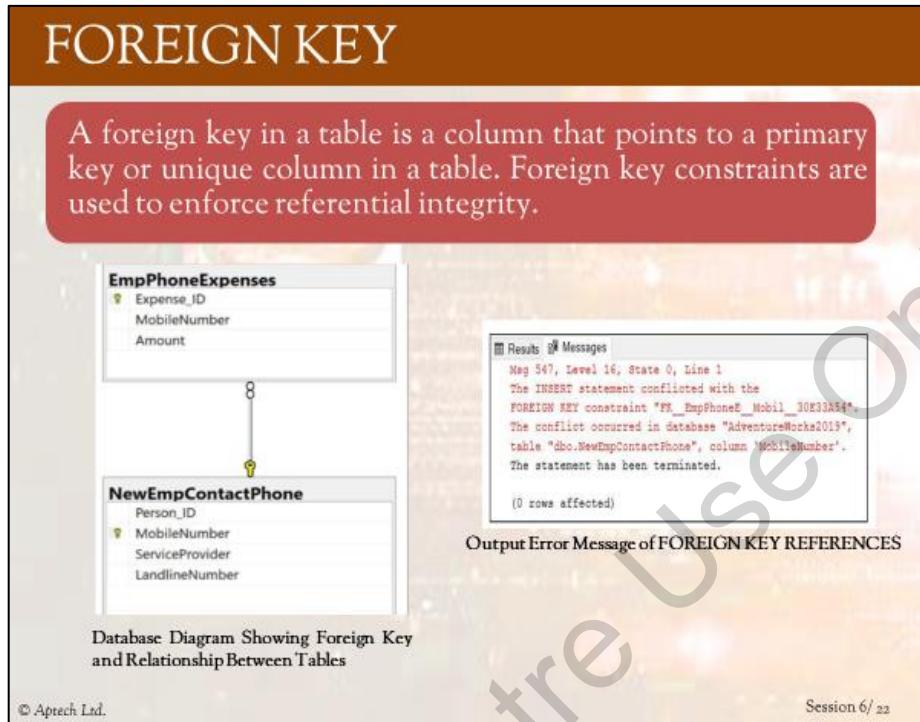
	Person_ID	MobileNumber	ServiceProvider	LandlineNumber
1	111	983345674	Verizon	NULL

Below the table, a message says 'Successfully Inserted Row'.

© Aptech Ltd. Session 6 / 21

Instructions to the Trainer(s):

- Using Slide 21, explain the unique constraints.
- A unique constraint is a single field or combination of fields that uniquely defines a record. Some of the fields can contain null values as long as the combination of values is unique.
- A UNIQUE constraint is used to ensure that only unique values are entered in a column or set of columns.
- It allows developers to make sure that no duplicate values are entered. Primary keys are implicitly unique.
- Unique key constraints enforce entity integrity because once the constraints are applied; no two rows in the table can have the same value for the columns.
- Though a value of NULL has been given for the LandlineNumber column, which are defined as UNIQUE, the command will execute successfully because UNIQUE constraints check only for the uniqueness of values but do not prevent null entries. The first statement shown in code snippet is executed successfully, but the next INSERT statement will fail even though the primary key value is different because the value for MobileNumber is a duplicate as shown in figure. This is because the column MobileNumber is defined to be unique and disallows duplicate values.



Instructions to the Trainer(s):

- Using Slide 22, explain they foreign key constraint.
- A foreign key in a table is a column that points to a primary key column in another table.
- Foreign key constraints are used to enforce referential integrity.
- A row is inserted into the table such that the mobile number is the same as one of the mobile numbers in EMP_ContactPhone. The command that will be written is shown in code snippet. The error message of code snippet is shown in figure.
- If there is no key in the referenced table having a value that is being inserted into the foreign key, the insertion will fail as shown in figure. It is, however, possible to add NULL value into a foreign key column.

CHECK

- A CHECK constraint limits the values that can be placed in a column.
- Check constraints enforce integrity of data.

- A CHECK constraint operates by specifying a search condition, which can evaluate to TRUE, FALSE, or unknown.
- Values that evaluate to FALSE are rejected.

Messages

```
Msg 547, Level 16, State 0, Line 1
The INSERT statement conflicted with the CHECK constraint
"CK_NewEMP_Ph_Amount_4BEFCEOF". The conflict occurred in
database "AdventureWorks2019",
table "dbo.NewEMP_PhoneExpenses", column 'Amount'.
The statement has been terminated.
```

Output Error Message of CHECK Constraint

© Aptech Ltd. Session 6/ 23

Instructions to the Trainer(s):

- Using Slide 23, explain the CHECK constraint.
- A CHECK constraint limits the values that can be placed in a column.
- Check constraints enforce integrity of data. For example, a CHECK constraint can be given to check if the value being entered into VoterAge is greater than or equal to 18. If the data being entered for the column does not satisfy the condition, then insertion will fail.
- A CHECK constraint operates by specifying a search condition, which can evaluate to TRUE, FALSE, or unknown.
- Values that evaluate to FALSE are rejected.
- Multiple CHECK constraints can be specified for a single column.
- A single CHECK constraint can also be applied to multiple columns by creating it at the table level.

NOT NULL

- A NOT NULL constraint enforces that the column will not accept null values.
- The NOT NULL constraints are used to enforce domain integrity, similar to CHECK constraints.

Instructions to the Trainer(s):

- Using Slide 24, explain the NOT NULL constraint.
- A NOT NULL constraint enforces that the column will not accept null values.
- The NOT NULL constraints are used to enforce domain integrity, similar to CHECK constraints.

Following code snippet is an example of creating not null constraint on columns:

```
CREATE TABLE [dbo].[Message](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [Message] [int] NOT NULL
)
```

The message value in the table cannot be null.

Summary

- Most tables have a primary key, made up of one or more columns of the table that identifies records uniquely.
- The nullability feature of a column determines whether rows in the table can contain a null value for that column.
- A DEFAULT definition for a column can be created at the time of table creation or added at a later stage to an existing table.
- The IDENTITY property of SQL Server is used to create identifier columns that can contain auto-generated sequential values to uniquely identify each row within a table.
- Constraints are used to apply business logic rules and enforce data integrity.
- A UNIQUE constraint is used to ensure that only unique values are entered in a column or set of columns.
- A foreign key in a table is a column that points to a primary key column in another table.
- A CHECK constraint limits the values that can be placed in a column.

Instructions to the Trainer(s):

- Show students Slide 25.
- Summarize the session by reading out each point on the slide.

Session 7: Azure SQL

7.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. Prepare a question or two that will be a key point to relate the current session objectives.

7.1.1 Teaching Skills

To teach this session, you should be well versed with basics of Azure SQL, steps to connect Azure SQL using SSMS and working with Azure SQL.

You should teach the concepts in the theory class using the images provided. For teaching in the class, you are expected to use slides and LCD projectors.

In-Class Activities

Follow the order given here during In-Class activities.

Slide 2

Objectives

- Explain Azure SQL
- List the features and benefits of Azure SQL
- State the differences between Azure SQL and on-premises SQL Server
- Explain steps to connect Azure SQL with SSMS

© Aptech Ltd. Session 7 / 2

Instructions to the Trainer(s):

Give students a brief overview of the current session through the session objectives listed in Slide 2.

7.2 In-Class Explanations

Slide 3

Introduction

- Cloud computing is a technology trend involving delivery of software, platforms, and infrastructure as services through Internet or networks.
- Microsoft Azure is a key offering in Microsoft's suite of cloud computing products and services.
- Database functions of Microsoft's cloud platform are provided by Microsoft Azure SQL Database, which is commonly known as **Azure SQL**.

- Data on Azure SQL does not have the constraint of being location-specific.
- This means that the data stored in SQL Azure can be viewed and edited from any location, as the entire data is stored on cloud storage platform.



© Aptech Ltd.

Session 7 / 3

Instructions to the Trainer(s):

- Introduce the session using Slide 3, explain the Azure SQL.
- Cloud computing is a technology trend, that involves the delivery of software, platforms, and infrastructure as services through the Internet or networks.
- Windows Azure is a key offering in Microsoft's suite of cloud computing products and services.
- The database functions of Microsoft's cloud platform are provided by Windows Azure SQL Database, which is commonly known as SQL Azure.
- SQL Azure can be used to store and manage data using queries and other functions that are similar to SQL Server.
- The data on SQL Azure does not have the constraint of being location-specific. This means that the data stored in SQL Azure can be viewed and edited from any location, as the entire data is stored on cloud storage platform.

Azure SQL 1-3



Cloud based relational database service that leverages existing SQL Server technologies

Extends functionality of Microsoft SQL Server for developing applications that are Web-based, scalable, and distributed

Is not just a single product but refers to a family of managed, intelligent, and secure products that use the SQL Server database engine in the Azure cloud

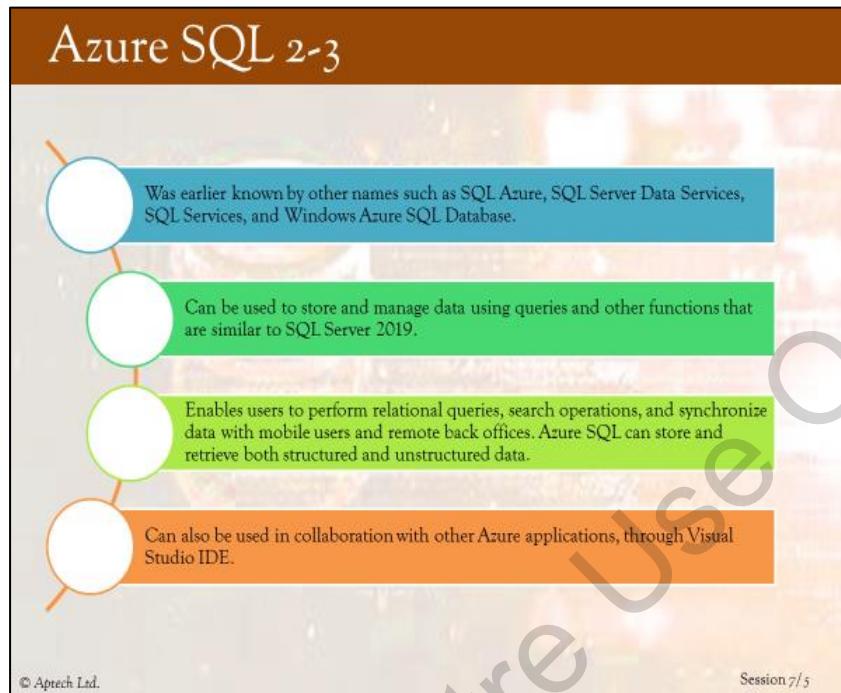
© Aptech Ltd.

Session 7/ 4



Instructions to the Trainer(s):

- Using Slide 4, explain on Azure SQL.
- SQL Azure is a cloud based relational database service that leverages existing SQL Server technologies.
- Microsoft SQL Azure extends the functionality of Microsoft SQL Server for developing applications that are Web-based, scalable, and distributed.
- SQL Azure is not just a single product but refers to a family of managed, intelligent, and secure products that use the SQL Server database.



Instructions to the Trainer(s):

- Using Slide 5, explain different applications and uses of Azure SQL to students.
- Can be used to store and manage data using queries and other functions that are similar to SQL Server 2019.
- Enables users to perform relational queries, search operations, and synchronize data with mobile users and remote back offices.
- Can store both structured and unstructured data.
- Manage and scale your entire application and data stack using Azure App Service.
- Can also be used in collaboration with other Azure applications through Visual Studio IDE.

Azure SQL 3-3

One of the competitors to Azure SQL is Amazon Web Services (AWS) and its Relational Database Services (RDS) product. Azure SQL is often compared to AWS RDS.

Azure SQL is:



- Both cloud based as well as on-premises applications can use the Azure SQL database.
- Applications retrieve data from Azure SQL through a protocol known as Tabular Data Stream (TDS).

© Aptech Ltd.

Session 7 / 6

Instructions to the Trainer(s):

- Using Slide 6, discuss the advantages of Azure SQL.
- Azure SQL is often compared to AWS RDS, Azure provides several different relational database services that are the equivalent of AWS' Relational Database Service (RDS).
- Azure SQL is 3.6 times faster and 86% less expensive than AWS.
- Both cloud-based as well as on-premises applications use the Azure SQL database.
- Applications retrieve data from Azure SQL through a protocol known as Tabular Data Stream (TDS).

Services and Products in the Azure SQL Family			
SQL Server on Azure Virtual Machines	Azure SQL Managed Instance	Azure SQL Database	Azure SQL Edge
Facilitates migration of existing apps or building new apps on the cloud for mission-critical SQL Server workloads.	Is the intelligent, scalable, cloud database service combining the broadcast SQL Server engine compatibility with benefits of a fully managed platform as a service.	Is the intelligent, scalable, relational database service built for the cloud. It is always up to date, with AI-powered and automated features that optimize performance.	Is a small-footprint, edge-optimized SQL database engine with built-in Artificial Intelligence (AI).

© Aptech Ltd.

Session 7/7

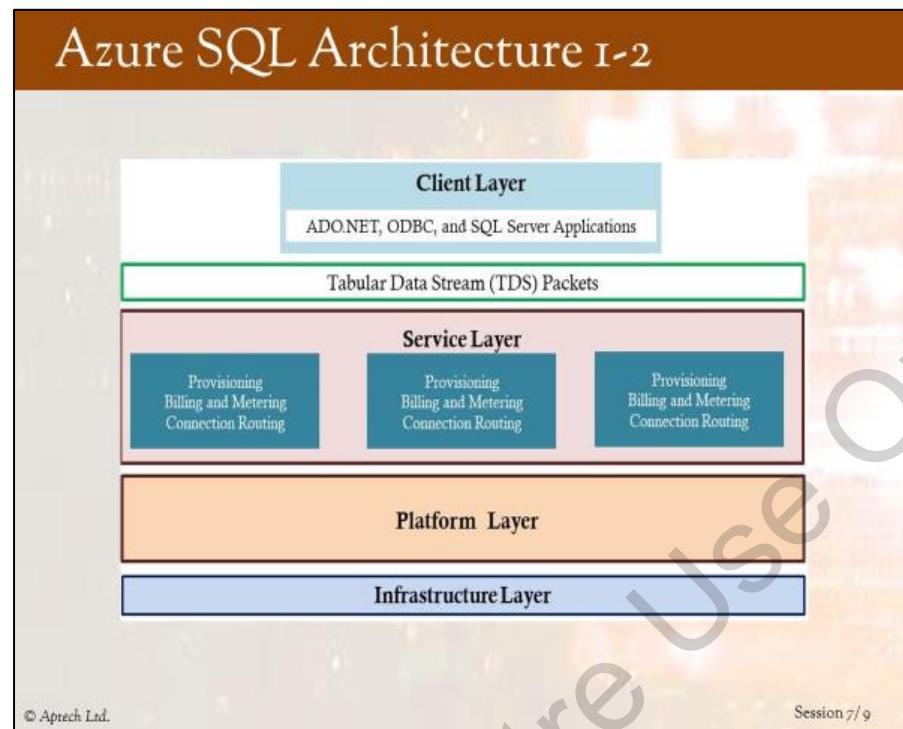
Instructions to the Trainer(s):

- Using Slide 7, explain the services and products in the Azure family.
- Azure SQL is a family of managed, secure, and intelligent products that use the SQL Server database engine in the Azure cloud.
- The services and products in the Azure family are as follows:
 - **SQL Server on Azure Virtual Machines:** Facilitates migration of existing apps or building apps on cloud.
 - **Azure SQL Managed Instance:** Is intelligent, scalable, cloud database service combining the broadcast SQL Server engine compatibility with benefits of fully managed platform as a service.
 - **Azure SQL Database:** Is intelligent, scalable, and relational database service provided for the cloud.
 - **Azure SQL Edge:** Is a small footprint edge-optimized SQL database engine with built-in AI.



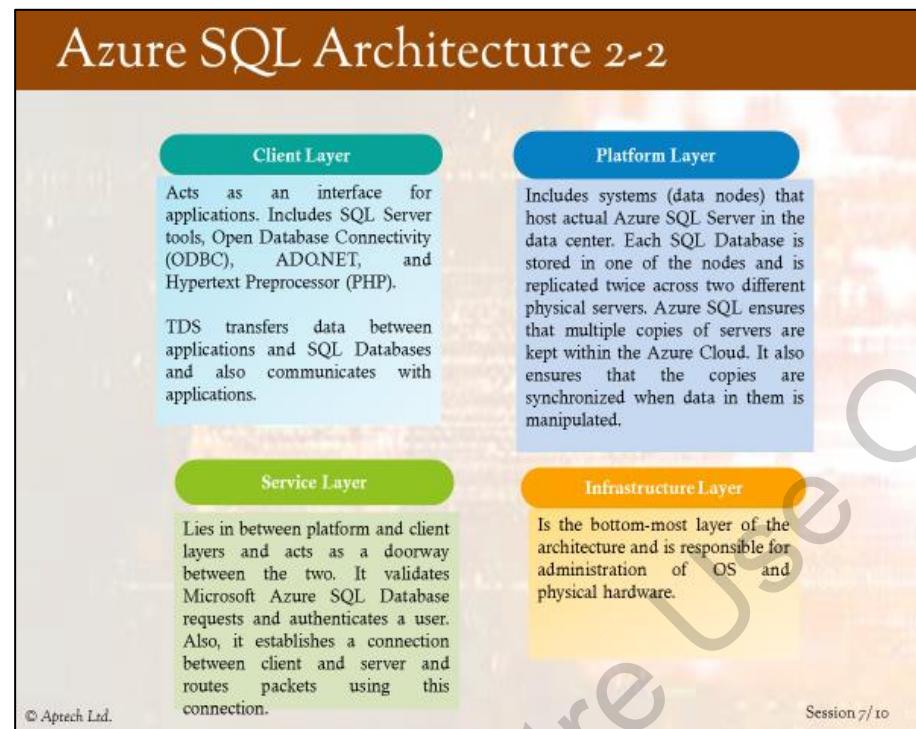
Instructions to the Trainer(s):

- Using Slide 8, explain the services offered by Azure SQL.
- Azure SQL comprises of the two important types of service, namely:
 - Infrastructure-as-a-Service (IaaS)
 - Platform-as-a-Service (PaaS)
- Azure SQL also consists of:
 - SQL Server on Azure Virtual Machines
 - Azure SQL Managed Instance
 - Azure SQL Database



Instructions to the Trainer(s):

- Using Slide 9, explain the students the architecture of Azure SQL.
- SQL Azure architecture contains four layers that works collectively to provide cloud based relational database. They are as follows:
 - Client layer
 - Service layer
 - Platform layer
 - Infrastructure layer
- These four layers help SQL Azure to work with third-party applications, open source, and many other technologies.



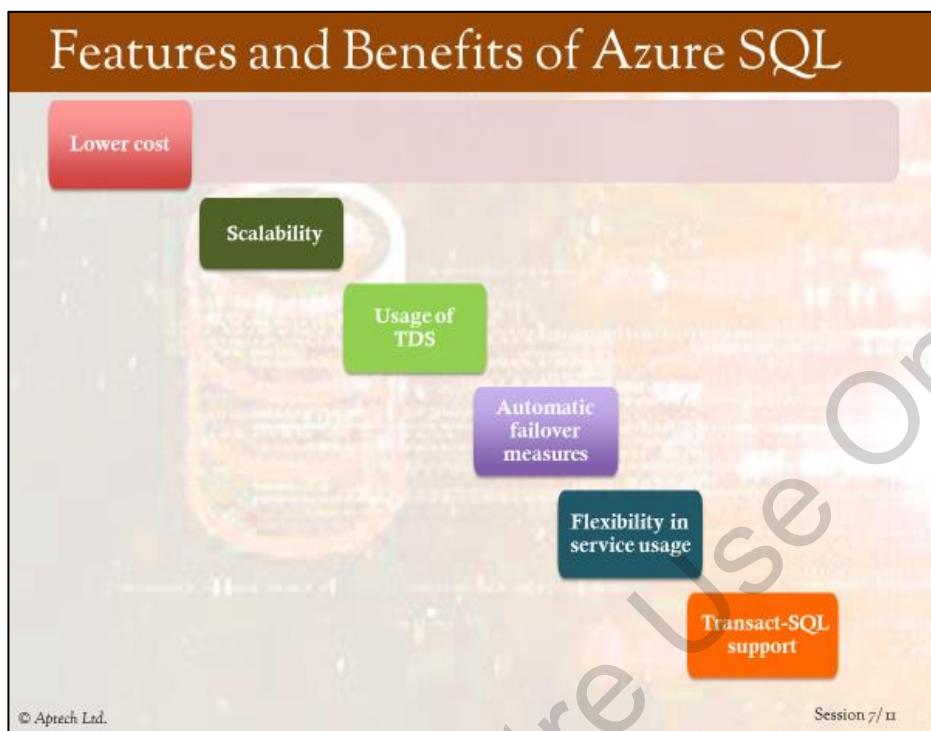
Instructions to the Trainer(s):

- Using Slide 10, explain in brief the four layers in Azure SQL architecture.
- **Client Layer**
 - Acts as an interface for applications.
 - Includes SQL Server tools, ODBC, ADO.NET, and PHP.
- **Services Layer**
 - Lies in between platform and client layers and acts as a doorway.
 - Validates Microsoft Azure SQL Database requests and authenticates a user.
 - Establishes connection between client and server.
- **Platform Layer**
 - Includes system that host actual Azure SQL Server in the data center.
 - Each SQL database server is stored in the form of nodes.
 - Ensures multiple copies of servers that are kept within the Azure Cloud.
- **Infrastructure Layer**
 - This layer represents the physical administration of hardware and OS that support Services layer.

In-Class Question:

Question: Is Azure SQL Server IaaS or PaaS?

Answer: Microsoft Azure SQL Database is a relational database-as-a-service, which falls into the industry category Platform as a Service (PaaS). Azure SQL Database is built on standardized hardware and software that is owned, hosted, and maintained by Microsoft.



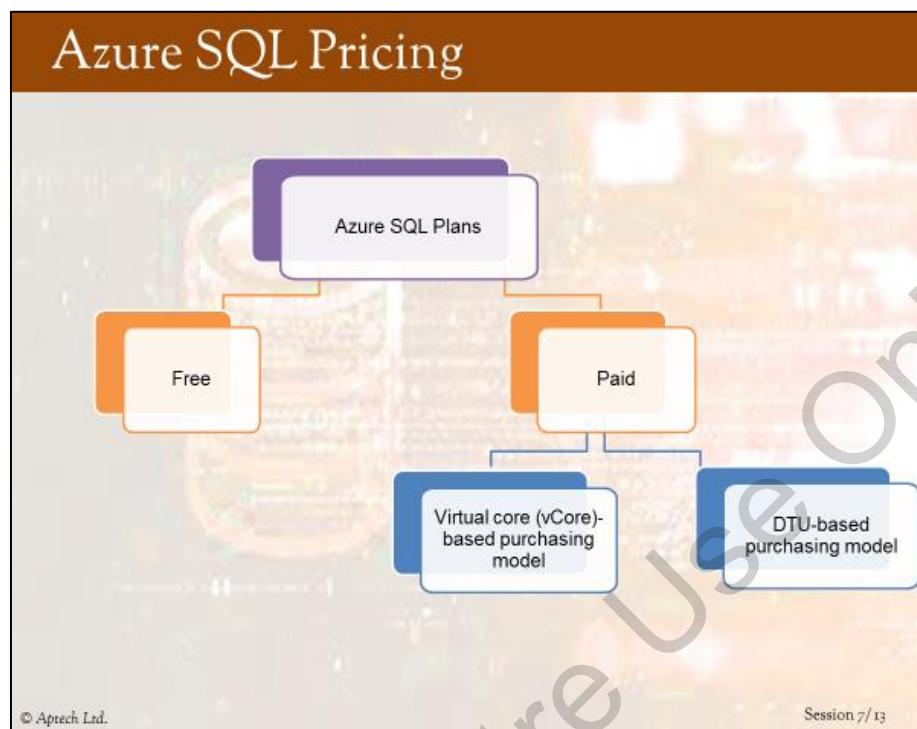
Instructions to the Trainer(s):

- Using Slide 11, explain the features and benefits of Azure SQL.
- Features and benefits of Azure SQL are:
 - Lower cost
 - Scalability
 - Usage of TDS
 - Automatic failover measures
 - Flexibility in service usage
 - Transact-SQL support

Difference between Azure SQL and On-Premises SQL Server	
Tools	On-premises SQL Server provides a number of tools for monitoring and management. All these tools may not be supported by Azure SQL, as there are a limited set of tools that are available in this version.
Backup	Backup and restore function must be supported in on-premises SQL Server for disaster recovery. For Azure SQL, as all the data is on the cloud platform, backup and restore is not required.
USE statement	USE statement is not supported by Azure SQL. Hence, the user cannot switch between databases in Azure SQL as compared to on-premises SQL Server.
Authentication	Azure SQL supports only SQL Server authentication and on-premises SQL Server supports both SQL Server authentication and Windows Authentication.
Transact-SQL support	Not all Transact-SQL functions are supported by Azure SQL.
Accounts and Logins	In Azure SQL, administrative accounts are created in the Azure management portal. Hence, there are no separate instance-level user logins.
Firewalls	Firewalls settings for allowed ports and IP addresses can be managed on physical servers for on-premises SQL Server. As an Azure SQL database is present on cloud, authentication through logins is the only method to verify the user.

Instructions to the Trainer(s):

- Using Slide 12, explain the difference between SQL Azure and On-Premises SQL server.
- On-premises software is a type of software delivery model that is installed and operated from a customer's in-house server and computing infrastructure.
- The vendor also provides after sales integration and support services but the security, availability, and overall management of it is the responsibility of customer.
- The difference is explained as follows:
 - **Tools:** On-premises SQL Server provides a number of tools for monitoring and management. All these tools may not be supported by Azure SQL, as there are a limited set of tools that are available in this version.
 - **Backup:** Backup and restore function must be supported in on-premises SQL Server for disaster recovery. For Azure SQL, all data is on cloud platform, backup, and restore is not required.
 - **USE Statement:** Not supported by Azure SQL. Hence, the user cannot switch between database in Azure SQL as compared to on-premises SQL Server.
 - **Authentication:** Azure SQL supports only Server authentication and on-premises SQL Server supports both SQL server authentication and Windows Authentication.
 - **Transact-SQL-Server:** Not all Transact-SQL functions are supported by Azure SQL.
 - **Accounts and Logins:** In Azure SQL, administrative accounts are created in the Azure management portal. Hence, there are no separate instance-level user logins.
 - **Firewalls:** For allowed ports and IP addresses can be managed on physical servers for on-premises SQL Server.



Instructions to the Trainer(s):

- Using Slide 13, explain the Azure SQL Pricing.
- Azure SQL Plans are of two types:
 - Free
 - Paid
- The Paid Azure Plan comprises:
 - Virtual core (vCore) based purchasing model
 - DTU-based purchasing model

Connect to SQL Azure with SSMS 1-4

➤ Creating a database on the cloud:

- 1 • Type the address <http://portal.azure.com> in the Address bar of your browser and sign up/sign in.
- 2 • Click Start under Start with an Azure free trial. Fill in the required information for a new Azure SQL account. Verify your identity through phone and credit card.
- 3 • Click SQL databases under Azure services. The SQL databases page will be displayed.
- 4 • Click Create SQL database at the bottom of the page. You will be asked to fill up information such as database name and server name.
- 5 • When you finish creating the new server and specify that server name, click Review+create on Create SQL Database page. The database will be successfully created.

© Aptech Ltd. Session 7 / 14

Instructions to the Trainer(s):

- Using Slide 14, discuss on how to connect to SQL Azure with SSMS.
- Create a database on the cloud:
 - Type the address in the Address bar of your browser and sign-up/sign-in.
 - Click Start under Start with an Azure free trial. Fill in the required information for a new Azure SQL account. Verify your identity through phone and credit card.
 - Click SQL databases under Azure services. The SQL databases page will be displayed.
 - Click Create SQL database at the bottom of the page. You will be asked to fill up information such as database name and server name.
 - When you finish creating the new server and specify that server name, click Review + create on Create SQL Database page. Then database will be successfully created.

Connect to Azure SQL with SSMS 2-4

➤ The process of connecting SQL Azure with SSMS is as follows:

1. Sign in to your Microsoft Azure account online.
2. Open Microsoft SQL Server Management Studio.
3. In Connect to Server dialog box, specify name of the Azure SQL server.
4. In Authentication box, select SQL Server Authentication.

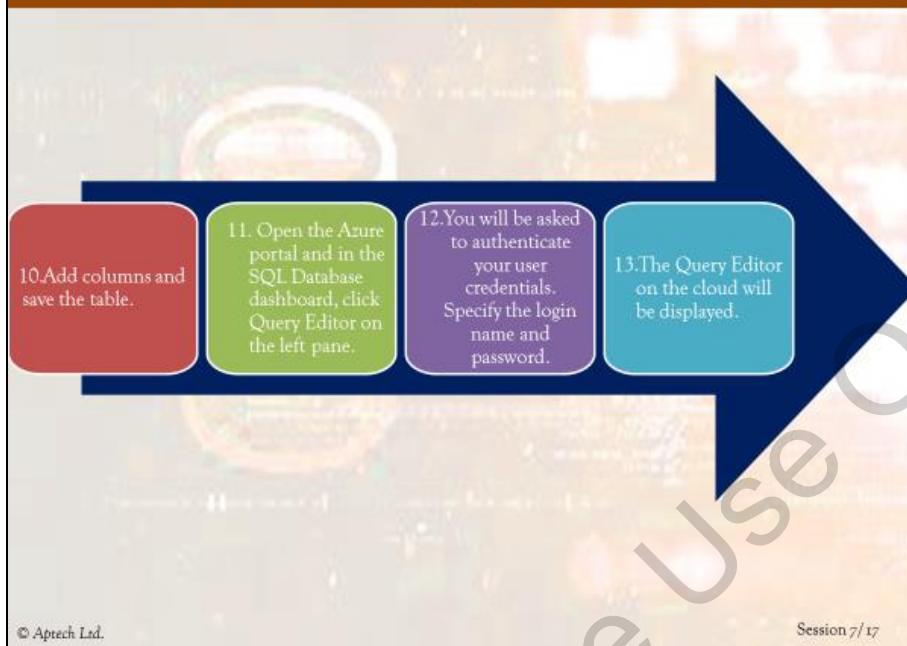
© Aptech Ltd. Session 7/15

Connect to Azure SQL with SSMS 3-4

5. In the Login box, type the name of the Azure SQL administrator account and the password.
6. Click Connect. You may then be prompted to create a new firewall rule.
7. Sign in to Azure by clicking Sign In. Your client IP address will be automatically populated in corresponding box.
8. Click Connect. Connection to database is successfully established.
9. Right-click the Tables node and click New→Table.

© Aptech Ltd. Session 7/16

Connect to Azure SQL with SSMS 4-4



Instructions to the Trainer(s):

- Using Slides 15 to 17, explain steps in the process of connecting SQL Azure with SSMS.
- The process of connecting SQL Azure with SSMS is as follows:
 - Sign in to your Microsoft Azure account online
 - Open Microsoft SQL-Server Management Studio
 - In Connect to Server dialog box, specify name of the Azure SQL server
 - In Authentication box, select SQL Server Authentication
 - In the Login box, type the name of Azure SQL administrator account and the password.
 - Click Connect. You may then be prompted to create a new firewall rule.
 - Sign in to Azure by clicking Sign In. Your client IP address will be automatically populated in corresponding box.
 - Click Connect. Connection to database is successfully established.
 - Right-click the Tables node and click New →Table.
 - Add columns and save the table.
 - Open the Azure portal and in the SQL Database dashboard, click Query Editor on the left pane.
 - You will be asked to authenticate your user credentials. Specify the login name and password.
 - The Query Editor on the cloud will be displayed.

Summary

- Most tables have a primary key, made up of one or more columns of the table that identifies records uniquely.
- The nullability feature of a column determines whether rows in the table can contain a null value for that column.
- A DEFAULT definition for a column can be created at the time of table creation or added at a later stage to an existing table.
- The IDENTITY property of SQL Server is used to create identifier columns that can contain auto-generated sequential values to uniquely identify each row within a table.
- Constraints are used to apply business logic rules and enforce data integrity.
- A UNIQUE constraint is used to ensure that only unique values are entered in a column or set of columns.
- A foreign key in a table is a column that points to a primary key column in another table.
- A CHECK constraint limits the values that can be placed in a column.

Instructions to the Trainer(s):

- Show students Slide 18.
- Summarize the session by reading out each point on the slide.

Session 8: Accessing Data

8.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. Prepare a question or two that will be a key point to relate the current session objectives.

8.1.1 Teaching Skills

To teach this session, you should be well versed with SELECT queries, various clauses, and working of XML, XML features, and so on.

You should teach the concepts in the theory class using the images provided. For teaching in the class, you are expected to use slides and LCD projectors.

In-Class Activities

Follow the order given here during In-Class activities.

Slide 2

Objectives

- Describe SELECT statement, its syntax, and use
- Explain various clauses used with SELECT
- State the use of ORDER BY clause
- Describe working with typed and untyped XML
- Explain the procedure to create, use, and view XML schemas

© Aptech Ltd. Session 8/ 2

Instructions to the Trainer(s):

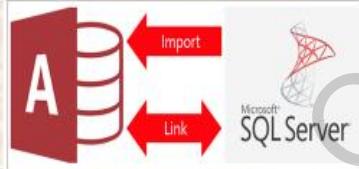
Give students a brief overview of the current session through the session objectives listed in Slide 2.

8.2 In-Class Explanations

Slide 3

Introduction

- ❑ The SELECT statement is a core command used to access data in SQL Server 2019.
- ❑ XML allows developers to develop their own set of tags and makes it possible for other programs to understand these tags.
- ❑ XML is the preferred means for developers to store, format, and manage data on the Web.



© Aptech Ltd. Session 8/3

Instructions to the Trainer(s):

- Using Slide 3, explain SELECT Statement.
- The SELECT statement is a core command used to access data in SQL Server 2019.
- XML allows developers to develop their own set of tags and makes it possible for other programs to understand these tags.
- XML is the preferred means for developers to store, format, and manage data on the Web.

SELECT Statement

Data in a table can be viewed using SELECT statement

This statement:

- Displays required information in a table
- Retrieves rows and columns from one or more tables
- Defines columns to be used for a query
- Consists of a series of expressions separated by commas
- Retrieves rows from database and enables selection of one or many rows or columns

Syntax:

```
SELECT <column_name1>...<column_nameN> FROM <table_name>
```

© Aptech Ltd.

Session 8/ 4

Instructions to the Trainer(s):

- Using Slide 4, explain the SELECT statement in details.
- A table with its data can be viewed using the SELECT statement.
- Following can be done using the SELECT Statement:
 - Displays the required information in a table
 - Retrieves rows and columns from one or more table
 - Defines columns to be used for a query
 - Consists of a series of expressions separated by commas
 - Retrieves rows from database and enables selectin of one or many rows or columns
- The syntax of SELECT statement can consist of a series of expressions separated by commas. Each expression in the statement is a column in the resultset.

In-Class Question:

Question: What is the use of SELECT statement?

Answer: The SELECT statement retrieves rows and columns from one or more tables.

SELECT Without FROM

- Many SQL versions use FROM in their query, but in all the versions from SQL Server 2005, including SQL Server 2019, one can use SELECT statements without using the FROM clause.
- Following code will display only first five characters from extreme left of the word 'International'.

```
SELECT LEFT('International',5)
```

The output is as follows:

The screenshot shows a SQL query results window. At the top, there are two tabs: 'Results' (which is selected) and 'Messages'. Below the tabs, the results are displayed in a table with one column labeled '(No column name)'. The table has two rows. The first row contains the number '1' and the string 'Inter'. The second row contains the number '2' and the string 'ntern'. Below the table, the text 'First Five Characters from the Extreme Left of the Word' is displayed. At the bottom left of the window, it says '© Aptech Ltd.' and at the bottom right, it says 'Session 8/5'.

First Five Characters from the Extreme Left of the Word

© Aptech Ltd. Session 8/5

Instructions to the Trainer(s):

- Using Slide 5, explain the SELECT statement without FROM.
- Many SQL versions use FROM in their query, but in all the versions from SQL Server 2005, including SQL Server 2012, one can use SELECT statements without using the FROM clause.
- The code will display only the first five characters from the extreme left of the word 'International'.

Displaying All Columns

- The asterisk (*) is used in the SELECT statement to retrieve all the columns from the table.
- It is used as a shorthand to list all the column names in the table named in the FROM clause.

Following is the syntax for selecting all columns

```
SELECT * FROM <table_name>
```

```
USE AdventureWorks2019
SELECT * FROM HumanResources.Employee
GO
```

	BusinessEntityID	NationalIDNumber	LoginID	OrganizationNode	OrganizationLevel	JobTitle
1	1	295847284	adventure-works\ken0	NULL	NULL	Chief Executive Officer
2	2	245797967	adventure-works\tem0	0x58	1	Vice President of Engineering
3	3	509647174	adventure-works\roberto0	0x5AC0	2	Engineering Manager
4	4	112457891	adventure-works\rob0	0x5AD8	3	Senior Tool Designer
5	5	695256908	adventure-works\igal0	0x5ADA	3	Design Engineer
6	6	998320692	adventure-works\josef0	0x5ADE	3	Design Engineer
7	7	134969118	adventure-works\dyer0	0x5AE1	3	Research and Development Manager
8	8	811994146	adventure-works\idane1	0x5AE158	4	Research and Development Engineer
9	9	658797903	adventure-works\gig0	0x5AE168	4	Research and Development Engineer
10	10	879342154	adventure-works\michael0	0x5AE178	4	Research and Development Manager

Displaying All Columns

Session 8/6

Instructions to the Trainer(s):

- Using Slide 6, explain how to display all columns.
- The asterisk (*) is used in the SELECT statement to retrieve all the columns from the table.
- It is used as a shorthand to list all the column names in the tables named in the FROM clause.

Displaying Selected Columns

- The SELECT statement displays or returns certain relevant columns that are chosen by the user or mentioned in the statement.
- To display specific columns, knowledge of the relevant column names in the table is required.

```
USE AdventureWorks2019  
SELECT LocationID, CostRate FROM  
Production.Location  
GO
```

	LocationID	CostRate
1	1	0.00
2	2	0.00
3	3	0.00
4	4	0.00
5	5	0.00
6	6	0.00
7	7	0.00
8	10	22.50
9	20	25.00
10	30	14.50
11	40	15.75
12	45	18.00
13	50	12.25
14	60	12.25

LocationID and CostRate Columns

Instructions to the Trainer(s):

- Using Slide 7, explain how to display selected rows.
- Mention that the SELECT statement displays or returns certain relevant columns that are chosen by the user or mentioned in the statement.
- To display specific columns, the knowledge of the relevant column names in the table is required.
- For example, to display the cost rates in various locations from Production.Location table in AdventureWorks2019 database, the SELECT statement is as shown in the code snippet.
- Figure shows LocationID and CostRate columns from AdventureWorks2019 database.

Different Expressions with SELECT

- SELECT statement allows users to specify different expressions in order to view the resultset in an ordered manner.
- These expressions assign different names to columns in the resultset, compute values, and eliminate duplicate values

Instructions to the Trainer(s):

- Using Slide 8, explain different expressions with SELECT.
- SELECT Statement allows users to specify different expressions in order to view the resultset in an ordered manner.
- These expressions assign different names to columns in the resultset, compute values, and eliminate duplicate values.

Using Constants in Result Sets

- Used when character columns are joined
- Help in proper formatting or readability
- Not specified as a separate column in the resultset
- More efficient for an application to build the constant values into the results

```
USE AdventureWorks2019
SELECT Name +':'+' CountryRegionCode +'->'+ Group FROM Sales.SalesTerritory
GO
```

Country Name, Country Region Code, and Corresponding Group

© Aptech Ltd. Session 8/9

Instructions to the Trainer(s):

- Using Slide 9, explain how to use constants in result sets.
- SELECT statement allows the users to specify different expressions in order to view the resultset in an ordered manner. These expressions assign different names to the columns in the resultset, compute values, and eliminate duplicate values.
- Character string constants are used when character columns are joined. They help in proper formatting or readability.
- These constants are not specified as a separate column in the resultset. It is usually more efficient for an application to build the constant values into the results when they are displayed, rather than making use of the server to incorporate the constant values.
- For example, to include ':' and '→' in the resultset so as to display the country name, country region code, and its corresponding group, the SELECT statement is shown in the code snippet.
- Figure displays the country name, country region code, and corresponding group from Sales.SalesTerritory of AdventureWorks2019 database.

Renaming ResultSet Column Names 1-2

- Columns displayed in resultsets of queries have corresponding headings specified in the table.

These headings can be:

- Changed
- Renamed
- Can be assigned a new name by using AS clause

- By customizing the headings, they become more understandable and meaningful.

Instructions to the Trainer(s):

- Using Slide 10, explain how to rename column name.
- When columns are displayed in the resultset they come with corresponding headings specified in the table.
- These headings can be:
 - Changed
 - Renamed
 - Assigned a new name by using AS clause
- Therefore, by customizing the headings, they become more understandable and meaningful.

In-Class Question:

Question: What is the use of RENAME command?

Answer: Changes the filename under which a file is stored.

Renaming ResultSet Column Names 2-2

The screenshot shows three separate result sets from SQL Server Management Studio (SSMS) illustrating how to rename columns.

- Result Set 1:** Shows a column named "NameRegionGroup" containing region mappings. The heading is "Column Heading Modified to NameRegionGroup".

	NameRegionGroup
1	Northwest:US->North America
2	Northeast:US->North America
3	Central US ->North America
4	Southwest:US->North America
5	Southeast:US->North America
6	Canada:CA->North America
7	France:FR->Europe
8	Germany:DE->Europe
9	Australia:AU->Pacific
10	United Kingdom:GB->Europe

- Result Set 2:** Shows a column named "ModifiedDate" containing dates. The heading is "Column Heading Modified to ChangedDate".

	ModifiedDate
1	2009-01-07 00:00:00.000
2	2008-01-24 00:00:00.000
3	2007-11-04 00:00:00.000
4	2007-11-28 00:00:00.000
5	2007-12-30 00:00:00.000
6	2013-12-16 00:00:00.000
7	2009-02-01 00:00:00.000
8	2008-12-22 00:00:00.000
9	2009-01-09 00:00:00.000
10	2009-04-26 00:00:00.000

- Result Set 3:** Shows a column named "ChangedDate" containing dates. The heading is "Column Heading Modified to ChangedDate".

	ChangedDate
1	2009-01-07 00:00:00.000
2	2008-01-24 00:00:00.000
3	2007-11-04 00:00:00.000
4	2007-11-28 00:00:00.000
5	2007-12-30 00:00:00.000
6	2013-12-16 00:00:00.000
7	2009-02-01 00:00:00.000
8	2008-12-22 00:00:00.000
9	2009-01-09 00:00:00.000
10	2009-04-26 00:00:00.000

Instructions to the Trainer(s):

- Using Slide 11, explain how to rename column name.
- The Rename Command changes the name of a column to a new column name. It is also used to change the table to a new table name.
- Slide 11 displays how column heading can be modified to NameRegionGroup and ChangedDate.

Computing Values in ResultSet

- A SELECT statement can contain mathematical expressions by applying operators to one or more columns.
- It allows a resultset to contain values that do not exist in the base table, but are calculated from the values stored in the base table.

	ProductID	StandardCost	Discount
1	707	12.0278	1.804170
2	707	13.8782	2.081730
3	707	13.0863	1.962945
4	708	12.0278	1.804170
5	708	13.8782	2.081730
6	708	13.0863	1.962945

Calculated Discount Amount

Instructions to the Trainer(s):

- Using Slide 12, explain how to compute values in result sets.
- A SELECT statement can contain mathematical expressions by applying operators to one or more columns. It allows a resultset to contain values that do not exist in the base table, but which are calculated from the values stored in the base table.
- For example, consider the table, Production.ProductCostHistory from AdventureWorks2019 database. Consider the example where the production people decide to give 15% discount on the standard cost of all the products.
- The discount amount does not exist but can be calculated by executing the SELECT statement shown in the code snippet. Figure shows the output where discount amount is calculated using SELECT statement.

Using DISTINCT

- The keyword DISTINCT prevents the retrieval of duplicate records. It eliminates rows that are repeating from the resultset of a SELECT statement.

For example,

- If the StandardCost column is selected without using the DISTINCT keyword, it will display all the standard costs present in the table.
- On using the DISTINCT keyword in the query, SQL Server will display every record of StandardCost only once.

Instructions to the Trainer(s):

- Using Slide 13, explain the DISTINCT keyword.
- The keyword DISTINCT prevents the retrieval of duplicate records. It eliminates rows that are repeating from the resultset of a SELECT statement.
- For example, if the StandardCost column is selected without using the DISTINCT keyword, it will display all the standard costs present in the table. Using the DISTINCT keyword in the query, SQL Server will display every record of StandardCost only once.

Using TOP and PERCENT

- The TOP keyword will display only first few set of rows as a resultset

- The TOP expression can also be used with other statements such as INSERT, UPDATE, and DELETE.

Syntax:

```
SELECT [ALL|DISTINCT] [TOPexpression [PERCENT] [WITHTIES]]
```

where,

expression: is the number or the percentage of rows to be returned as the result.

PERCENT: returns the number of rows limited by percentage.

WITH TIES: is the additional number of rows that is to be displayed.

© Aptech Ltd.

Session 8/ 14

Instructions to the Trainer(s):

- Using Slide 14, explain the TOP and PERCENT keywords.
- The TOP keyword will display only the first few set of rows as a resultset.
- The set of rows is either limited to a number or a percent of rows.
- The TOP expression can also be used with other statements such as
 - INSERT, UPDATE, and DELETE.
- The SELECT statement has various clauses associated with it. In this section, each clause is discussed in detail.

For example,

```
SELECT TOP 10 * FROM Production.ProductCostHistory
```

The query will give top 10 rows in the table.

SELECT with INTO

- The INTO clause creates a new table and inserts rows and columns listed in the SELECT statement into it.

INTO clause also inserts existing rows into the new table.

	ProductModelID	Name
1	122	All-Purpose Bike Stand
2	119	Bike Wash
3	115	Cable Lock
4	98	Chain
5	1	Classic Vest
6	2	Cycling Cap
7	121	Fender Set - Mountain
8	102	Front Brakes
9	103	Front Derailleur
10	3	Full-Finger Gloves
11	4	Half-Finger Gloves

New Table

© Aptech Ltd.

Session 8/15

Instructions to the Trainer(s):

- Using Slide 15, explain the INTO keyword.
- The INTO clause creates a new table and inserts rows and columns listed in the SELECT statement into it.
- INTO clause also inserts existing rows into the new table.
- In order to execute this clause with the SELECT statement, the user must have the permission to CREATE TABLE in the destination database.

In-Class Question:

Question: What is the use of INTO clause?

Answer: The INTO clause creates a new table and inserts rows and columns listed in the SELECT statement into it.

SELECT with WHERE 1-3

- The WHERE clause with SELECT statement is used to conditionally select or limit the records retrieved by the query.
- A WHERE clause specifies a Boolean expression to test the rows returned by the query.
- The row is returned if the expression is true and is discarded if it is false.

Operator	Description
=	Equal to
< >	Not equal to
>	Greater than
<	Less than
> =	Greater than or equal to
< =	Less than or equal to
!	Not

Operator	Description
BETWEEN	Between a range
LIKE	Search for an ordered pattern
IN	Within a range

Operators

Instructions to the Trainer(s):

- Using Slide 16, explain the WHERE clause.
- The WHERE clause with SELECT statement is used to conditionally select or limit the records retrieved by the query.
- A WHERE clause specifies a Boolean expression to test the rows returned by the query.
- The row is returned if the expression is true and is discarded if it is false.
- Operators in WHERE Clause are BETWEEN, LIKE, and IN
 - BETWEEN: Between a range
 - LIKE: Search for an ordered pattern
 - IN: Within a range

SELECT with WHERE 2-3

The screenshot shows two separate result sets from SSMS. The top result set, titled 'SELECT with WHERE clause', displays data from a table with columns ProductID, StartDate, EndDate, StandardCost, and ModifiedDate. The bottom result set, titled 'Output of Where Clause with < Operator', displays data from a table with columns DepartmentID, Name, GroupName, and ModifiedDate.

	ProductID	StartDate	EndDate	StandardCost	ModifiedDate
1	707	2012-05-30 00:00:00.000	2013-05-29 00:00:00.000	13.8782	2013-05-29 00:00:00.000
2	708	2012-05-30 00:00:00.000	2013-05-29 00:00:00.000	13.8782	2013-05-29 00:00:00.000
3	711	2012-05-30 00:00:00.000	2013-05-29 00:00:00.000	13.8782	2013-05-29 00:00:00.000
4	712	2012-05-30 00:00:00.000	2013-05-29 00:00:00.000	5.2297	2013-05-29 00:00:00.000
5	713	2012-05-30 00:00:00.000	2013-05-29 00:00:00.000	29.0807	2013-05-29 00:00:00.000

	DepartmentID	Name	GroupName	ModifiedDate
1	1	Engineering	Research and Development	2008-04-30 00:00:00.000
2	2	Tool Design	Research and Development	2008-04-30 00:00:00.000
3	3	Sales	Sales and Marketing	2008-04-30 00:00:00.600
4	4	Marketing	Sales and Marketing	2008-04-30 00:00:00.600
5	5	Purchasing	Inventory Management	2008-04-30 00:00:00.000
6	6	Research and Development	Research and Development	2008-04-30 00:00:00.000
7	7	Production	Manufacturing	2008-04-30 00:00:00.000
8	8	Production Control	Manufacturing	2008-04-30 00:00:00.000
9	9	Human Resources	Executive General and Administration	2008-04-30 00:00:00.000

Output of Where Clause with < Operator

© Aptech Ltd. Session 8/17

Instructions to the Trainer(s):

- Using Slide 17, discuss the SELECT with WHERE.
- Slide 17 displays the resultant data for SELECT with WHERE clause and the output of WHERE Clause with < Operator.
- WHERE Clause is used to filter the records from the table based on the specified condition.

SELECT with WHERE 3-3

Wildcard	Description	Example
-	It will display a single character	<code>SELECT * FROM Person.Contact WHERE Suffix LIKE 'Jr_'</code>
%	It will display a string of any length	<code>SELECT * FROM Person.Contact WHERE LastName LIKE 'B%'</code>
[]	It will display a single character within the range enclosed in the brackets	<code>SELECT * FROM Sales.CurrencyRate WHERE ToCurrencyCode LIKE '[C][AN][DY]'</code>
[^]	It will display any single character not within the range enclosed in the brackets	<code>SELECT * FROM Sales.CurrencyRate WHERE ToCurrencyCode LIKE '[A[^R][^S]]'</code>

Wildcard Characters

© Aptech Ltd.

Session 8/18

Instructions to the Trainer(s):

- Using Slide 18, explain the Wildcard characters concept.
- The WHERE clause can also be used with wildcard characters as shown in table.
- All wildcard characters are used along with LIKE keyword to make the query accurate and specific.
- For example, the wildcard characters discussed in Slide 18 are: -, %, [] and [^]. These are explained as follows:
 - - It will display a single character
 - % It will display a string of any length
 - [] It will display a single character within the range enclosed in the brackets
 - [^] It will display any single character not within the range enclosed in the brackets

In-Class Question:

Question: Is wildcard a command?

Answer: In software, a wildcard character is a kind of placeholder represented by a single character, such as an asterisk (*), which can be interpreted as a number of literal characters or an empty string. It is often used in file searches so the full name does not have to be typed.

GROUP BY Clause

- The GROUP BY clause partitions the resultset into one or more subsets.
- Each subset has values and expressions in common.
- If an aggregate function is used in the GROUP BY clause, the resultset produces single value per aggregate.

	WorkOrderID	(No column name)
1	13	17.6000
2	14	17.6000
3	15	4.0000
4	16	4.0000
5	17	4.0000
6	18	4.0000
7	19	4.0000

Output of GROUP BY Clause

Instructions to the Trainer(s):

- Using Slide 19, explain the GROUP BY Clause.
- The GROUP BY clause partitions the resultset into one or more subsets.
- Each subset has values and expressions in common. If an aggregate function is used in the GROUP BY clause, the resultset produces single value per aggregate.
- The GROUP BY keyword is followed by a list of columns, known as grouped columns.
- Every grouped column restricts the number of rows of the resultset.
- For every grouped column, there is only one row.
- The GROUP BY clause can have more than one grouped column.

ORDER BY Clause

- It specifies the order in which the columns should be sorted in a resultset.
- It sorts query results by one or more columns.

- A sort can be in either ascending (ASC) or descending (DESC) order.
By default, records are sorted in an ASC order.

	TerritoryID	Name	CountryRegionCode	Group	SalesYTD	SalesLastYear
1	8	Germany	DE	Europe	3805202.3478	1307949.7917
2	10	United Kingdom	GB	Europe	5012905.3658	1655623.3967
3	9	Australia	AU	Pacific	5077814.9154	2270548.9776
4	7	France	FR	Europe	4772598.3078	2306539.7601
5	3	Central	US	North America	3072175.118	3205014.0767
6	1	Northeast	US	North America	7087186.7882	3266044.4638
7	2	Northeast	US	North America	2402176.8478	3607148.9371
8	5	Southwest	US	North America	253867.2515	3925071.4318
9	4	Southwest	US	North America	10510853.8739	5366575.7098
10	6	Canada	CA	North America	6771829.1378	5893988.06

Output of ORDER BY Clause

Instructions to the Trainer(s):

- Using Slide 20, explain the ORDER BY clause.
- The ORDER BY Clause specifies the order in which the columns should be sorted in a resultset.
It sorts query results by one or more columns.
- A sort can be in either ascending (ASC) or descending (DESC) order.
- By default, records are sorted in an ASC order. To switch to the descending mode, use the optional keyword, DESC.
- When multiple fields are used, SQL Server considers the leftmost field as the primary level of sort and others as lower levels of sort.

Working with XML I-2

- Extensible Markup Language (XML) allows developers to develop their own set of tags and makes it possible for other programs to understand these tags.
- XML is the preferred means for developers to store, format, and manage data on the Web

Applications of today have a mix of technologies such as:

- ASP
- Microsoft .NET technologies
- XML
- SQL Server 2019 working in tandem

In such a scenario, it is better to store XML data within SQL Server 2019.

Instructions to the Trainer(s):

- Using Slide 21, explain the working with XML.
- Extensible Markup Language (XML) allows developers to develop their own set of tags and makes it possible for other programs to understand these tags.
- XML is the preferred means for developers to store, format, and manage data on the Web.
- Applications of today have a mix of technologies such as ASP, Microsoft .NET technologies, XML, and SQL Server 2019 working in tandem. In such a scenario, it is better to store XML data within SQL Server 2019.

Working with XML 2-2

Native XML databases in SQL Server 2019 have a number of advantages. Some of them are listed as follows:

Easy Data Search and Management - All the XML data is stored locally in one place, thus making it easier to search and manage.

Better Performance - Queries from a well-implemented XML database are faster than queries over documents stored in a file system. Also, the database essentially parses each document when storing it.

Easy data processing - Large documents can be processed easily.

Instructions to the Trainer(s):

- Using Slide 22, explain the working with XML.
- Native XML databases in SQL Server 2012 have a number of advantages. Some of them are listed as follows:
 - **Easy Data Search and Management** - All the XML data is stored locally in one place, thus making it easier to search and manage.
 - **Better Performance** - Queries from a well-implemented XML database are faster than queries over documents stored in a file system. Also, the database essentially parses each document when storing it.
 - **Easy data processing** - Large documents can be processed easily.
- SQL Server supports native storage of XML data by using the 'xml' data type.

XML Data Type

- The xml data type is used to store XML documents and fragments in an SQL Server database.
- An XML fragment is an XML instance with the top-level element missing from its structure.

The screenshot shows a SQL Server Management Studio window. The title bar says 'SQL Server Object Explorer'. In the center, there's a results grid with two columns: 'CellDetails' and 'Info'. The 'CellDetails' column contains a single row with the value '1'. The 'Info' column contains a single row with the XML fragment: '<Info><Call>Local</Call><Time>45 minutes</Time><...>'.

XML Data in Columns

© Aptech Ltd. Session 8/23

Instructions to the Trainer(s):

- Using Slide 23, explain the xml data.
- The `xml` data type is used to store XML documents and fragments in an SQL Server database.
- An XML fragment is an XML instance with the top-level element missing from its structure.
- A column of type `xml` can be added to a table at the time of creation or after its creation.
- The `xml` data type columns support `DEFAULT` values as well as the `NOT NULL` constraint.
- Data can be inserted into the `xml` column in the `Person.PhoneBilling` table as shown in code snippet in the Learner Guide. The output is shown in figure on Slide 23.
- The `DECLARE` statement is used to create variables of type `xml`.

Typed and Untyped XML 1-2

Two ways of storing XML documents

```
graph TD; A[Two ways of storing XML documents] --> B[Typed XML]; A --> C[Untyped XML]
```

TypedXMLinstance:

- An XML instance which has a schema associated
- It describes the structure and limits the contents of XML documents

© Aptech Ltd.

Session 8 / 24

Typed and Untyped XML 2-2

Untyped XML instance:

- Data can be created and stored in either table columns or variables depending upon the need and scope of data

Displaying XML Column with SELECT

TeamInfoXml

```
1 <MatchDetails>
2   <Team country="Australia" score="3" />
3   <Team country="Zimbabwe" score="2" />
4   <Team country="England" score="4" />
5 </MatchDetails>
```

Expanded XML Data in Column

© Aptech Ltd.

Session 8 / 25

Instructions to the Trainer(s):

- Using Slides 24 and 25, explain typed and untyped XML.
- There are two ways of storing XML documents in the `xml` data type columns namely,
 - Typed XML
 - Untyped XML
- **Typed XML instance:** An XML instance which has a schema associated with it is called typed XML instance. A schema is a header for an XML instance or document. It describes the

structure and limits the contents of XML documents by associating xml data types with XML element types and attributes.

- Using Slide 25, explain the Untyped XML instance.
- When we define an `xml` value as a variable, parameter, then it is called as Untyped XML.
- **Untyped XML instance:** Data can be created and stored in either table columns or variables depending upon the requirement and scope of data. Untyped XML type does not have schema.

Slide 26

Summary

- The SELECT statement retrieves rows and columns from tables.
- SELECT statement allows users to specify different expressions in order to view the resultset in an ordered manner.
- A SELECT statement can contain mathematical expressions by applying operators to one or more columns.
- The keyword DISTINCT prevents the retrieval of duplicate records.
- XML allows developers to develop their own set of tags and makes it possible for other programs to understand these tags.
- A typed XML instance is an XML instance which has a schema associated with it.
- XML data can be queried and retrieved using XQuery language.

© Aptech Ltd. Session 8 / 26

Instructions to the Trainer(s):

- Show students Slide 26.
- Summarize the session by reading out each point on the slide.

Session 9: Advanced Queries and Joins

9.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. Prepare a question or two that will be a key point to relate the current session objectives.

9.1.1 Teaching Skills

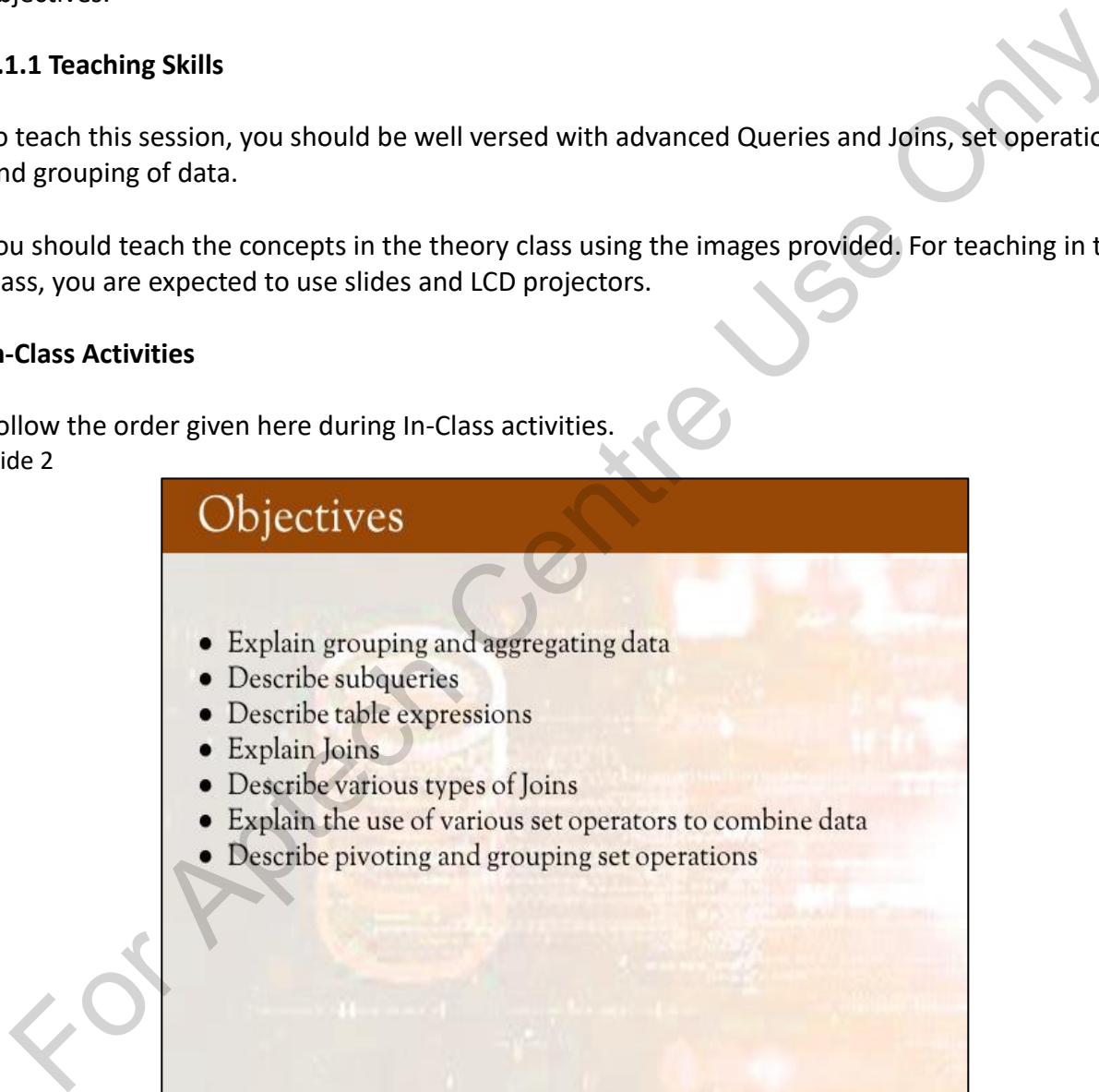
To teach this session, you should be well versed with advanced Queries and Joins, set operations and grouping of data.

You should teach the concepts in the theory class using the images provided. For teaching in the class, you are expected to use slides and LCD projectors.

In-Class Activities

Follow the order given here during In-Class activities.

Slide 2



Objectives

- Explain grouping and aggregating data
- Describe subqueries
- Describe table expressions
- Explain Joins
- Describe various types of Joins
- Explain the use of various set operators to combine data
- Describe pivoting and grouping set operations

© Aptech Ltd. Session 9 / 2

Instructions to the Trainer(s):

Give students a brief overview of the current session through the session objectives listed in Slide 2.

9.2 In-Class Explanations

Slide 3

Introduction

SQL Server 2019 includes several powerful query features that help to retrieve data efficiently and quickly.

- Data can be grouped and/or aggregated together in order to present summarized information.
- Joins help to combine column data from two or more tables based on a logical relationship between the tables.
- Set operators such as UNION and INTERSECT combines row data from two or more tables.
- PIVOT and UNPIVOT operators are used to transform the orientation of data from column-oriented to row-oriented and vice versa.
- GROUPING SET subclause of the GROUP BY clause helps to specify multiple groupings in a single query.

© Aptech Ltd.

Session 9/3

Instructions to the Trainer(s):

- Using Slide 3, explain that SQL Server 2019 includes several powerful query features that help you to retrieve data efficiently and quickly.
- Data can be grouped and/or aggregated together in order to present summarized information.
- Using the concept of subqueries, a resultset of a SELECT can be used as criteria for another SELECT statement or query. Joins help you to combine column data from two or more tables based on a logical relationship between the tables.
- On the other hand, set operators such as UNION and INTERSECT help you to combine row data from two or more tables.
- The PIVOT and UNPIVOT operators are used to transform the orientation of data from column-oriented to row-oriented and vice versa.
- The GROUPING SET subclause of the GROUP BY clause helps to specify multiple groupings in a single query.

Grouping Data 1-5

This clause partitions the resultset into one or more subsets and each subset has values and expressions in common.

- It is followed by a list of columns, known as grouped columns.
- It restricts the number of rows of the resultset.

WorkOrderID	TotalHoursPerWorkOrder
1	13
2	14
3	15
4	16
5	17
6	18
7	19
8	20
9	21
10	22

Using GROUP BY Clause

Instructions to the Trainer(s):

- Using Slide 4, explain grouping of data.
- The GROUP BY clause partitions the resultset into one or more subsets.
- The clause partitions the resultset into one or more subsets and each subset has values and expressions in common.
 - It is followed by a list of columns, known as grouped columns.
 - It restricts the number of rows of the resultset.
- Give an example of Production.WorkOrderRouting table in the AdventureWorks2019 database. The total resource hours per work order must be calculated. To achieve this, the records must be grouped by work order number, that is, WorkOrderID. Explain the code snippet which retrieves and displays the total resource hours per work order along with the work order number. In this query, a built-in function named SUM() is used to calculate the total. SUM() is an aggregate function. Aggregate functions will be covered in detail in a later section. Executing this query will return all the work order numbers along with the total number of resource hours per work order.

Grouping Data 2-5

GROUP BY with WHERE

- To restrict the rows for grouping.
- The rows satisfy the search condition are considered for grouping.
- The rows that do not meet the conditions in the WHERE clause are eliminated before any grouping is done.

WorkOrderID	TotalHoursPerWorkOrder
1	13
2	14
3	15
4	16
5	17
6	18
7	19
8	20

Output of GROUP BY with WHERE

© Aptech Ltd.

Session 9/5

Instructions to the Trainer(s):

- Using Slide 5, explain the GROUP BY with WHERE.
- The WHERE clause can also be used with GROUP BY clause to restrict the rows for grouping. The rows that satisfy the search condition are considered for grouping.
- The rows that do not meet the conditions in the WHERE clause are eliminated before any grouping is done.
- The GROUP BY with WHERE clause consists of following:
 - To restrict the rows for grouping
 - The rows satisfy the search condition are considered for grouping
 - The rows that do not meet the conditions in the WHERE clause are eliminated before any grouping is done

Grouping Data 3-5

GROUP BY with NULL

- If the grouping column contains a NULL value, that row becomes a separate group in the resultset.
- If the grouping column contains more than one NULL value, the NULL values are put into a single row.

	Class	AverageListPrice
1	NULL	16.314
2	H	1679.4964
3	L	370.6887
4	M	635.5816

Output of GROUP BY with NULL

© Aptech Ltd. Session 9/ 6

Instructions to the Trainer(s):

- Using Slide 6, explain the GROUP BY clause with NULL.
- Following can be considered for GROUP BY with NULL:
 - If the grouping column contains a NULL value, that row becomes a separate group in the resultset.
 - If the grouping column contains more than one NULL value, the NULL values are put into a single row.

For more information, refer to:

<https://learnsql.com/blog/null-values-group-clause/>

Grouping Data 4-5

GROUPBY with ALL

➤ The ALL keyword can also be used with the GROUP BY clause.
➤ It includes all the groups that the GROUP BY clause produces and even those which do not meet the search conditions.

Group	TotalSales
1 Europe	13590506.0212
2 North America	33182889.0168
3 Pacific	NULL

Output of GROUP BY with ALL

© Aptech Ltd. Session 9/7

Instructions to the Trainer(s):

- Using Slide 7, explain the GROUP BY with ALL.
- The ALL keyword can also be used with the GROUP BY clause.
- It is significant only when the SELECT has a WHERE clause.
- When ALL is used, it includes all the groups that the GROUP BY clause produces. It even includes those groups which do not meet the search conditions.
- Explain the syntax of using GROUP BY with ALL.
- Consider the Sales.SalesTerritory table. This table has a column named Group indicating the geographic area to which the sales territory belongs to. Code snippet shown in Learner Guide calculates and displays the total sales for each group. The output must display all the groups regardless of whether they had any sales or not. To achieve this, the code makes use of GROUP BY with ALL.
- Apart from the rows that are displayed in code snippet, it will also display the group 'Pacific' with null values as shown in figure on Slide 7. This is because the Pacific region did not have any sales.

Grouping Data 5-5

GROUP BY with HAVING

- HAVING clause is used only with SELECT statement to specify a search condition for a group.
- The HAVING clause acts as a WHERE clause in places where the WHERE clause cannot be used against aggregate functions such as SUM().

Syntax:

```
SELECT <column_name> FROM <table_name> GROUP BY <column_name> HAVING  
<search_condition>
```

© Aptech Ltd.

Session 9/8

Instructions to the Trainer(s):

- Using Slide 8, explain the GROUP BY with HAVING.
- HAVING clause is used only with SELECT statement to specify a search condition for a group.
- The HAVING clause acts as a WHERE clause in places where the WHERE clause cannot be used against aggregate functions such as SUM().
- Once you have created groups with a GROUP BY clause, you may wish to filter the results further.
- The HAVING clause acts as a filter on groups, similar to how the WHERE clause acts as a filter on rows returned by the FROM clause.
- Explain the code snippet in the Learner Guide which displays the row with the group 'Pacific' as it has total sales less than 6000000. The output of this is only one row, with Group name Pacific and total sales, 5977814.9154.

In-Class Question:



What is HAVING clause used for?

Answer:

The HAVING clause acts as a WHERE clause in places where the WHERE clause cannot be used against aggregate functions such as SUM().

Summarizing Data I-4

- GROUP BY clause also uses operators such as CUBE and ROLLUP to return summarized data.
- Number of columns in the GROUP BY clause determines number of summary rows in the resultset.

Instructions to the Trainer(s):

- Using Slide 9, explain how data can be summarized.
- GROUP BY clause uses operator such as CUBE and ROLLUP to return summarized data.
- Number of columns in the GROUP BY clause determines number of summary rows in the resultset.

Summarizing Data 2-4

CUBE:

CUBE is an aggregate operator that produces a super-aggregate row.

In addition to usual rows provided by the GROUP BY, it also provides the summary of rows that the GROUP BY clause generates.

Name	CountryRegionCode	TotalSales
1 Germany	DE	3805202.3478
2 NULL	DE	3805202.3478
3 France	FR	4772398.3078
4 NULL	FR	4772398.3078
5 United Kingdom	GB	5012905.3656
6 NULL	GB	5012905.3656
7 Central	US	3072175.118
8 Northeast	US	2402176.8476
9 Northwest	US	7887186.7882
10 Southeast	US	2538667.2515

Using GROUP BY with CUBE

Instructions to the Trainer(s):

- Using Slide 10, explain the CUBE operator.
- CUBE is an aggregate operator that produces a super-aggregate row.
- In addition to the usual rows provided by the GROUP BY, it also provides the summary of the rows that the GROUP BY clause generates.
- The summary row is displayed for every possible combination of groups in the resultset.
- The summary row displays NULL in the resultset, but at the same time returns all the values for those.
- Explain the syntax of CUBE.
- Explain the code snippet shown in Learner Guide which demonstrates the use of CUBE. It retrieves and displays the total sales of each country and also, the total of the sales of all the countries' regions. The output is shown in figure.

Summarizing Data 3-4

ROLLUP:

- It introduces summary rows into the resultset.
- Generates a resultset that shows groups arranged in a hierarchical order.
- It arranges the groups from the lowest to the highest.

- ROLLUP assumes a hierarchy among the dimension columns and only generates grouping sets based on this hierarchy.
- ROLLUP is often used to generate subtotals and totals for reporting purposes.
- ROLLUP is commonly used to calculate the aggregates of hierarchical data such as sales by year → quarter → month.

Instructions to the Trainer(s):

- Using Slide 11, explain the ROLLUP operator.
- In addition to the usual rows that are generated by the GROUP BY, it also introduces summary rows into the resultset. It is similar to CUBE operator but generates a resultset that shows groups arranged in a hierarchical order. It arranges the groups from the lowest to the highest.
- Group hierarchy in the result is dependent on the order in which the columns that are grouped are specified.
- Explain the code snippet in the Learner Guide which demonstrates the use of ROLLUP. It retrieves and displays the total sales of each country, the total of the sales of all the countries' regions and arranges them in order.

In-Class Question:

Question: What is used to add summary rows in the sorted result set?

Answer: ROLLUP operator is used to add summary rows in the sorted result set.

Summarizing Data 4-4

	Name	CountryRegionCode	TotalSales
1	Central	US	3072175.118
2	Central	NULL	3072175.118
3	France	FR	4772398.3078
4	France	NULL	4772398.3078
5	Germany	DE	3805202.3478
6	Germany	NULL	3805202.3478
7	Northeast	US	2402176.8476
8	Northeast	NULL	2402176.8476
9	Northwest	US	7887186.7882
10	Northwest	NULL	7887186.7882
11	Southeast	US	2538667.2515
12	Southeast	NULL	2538667.2515
13	Southwest	US	10510853.8739
14	Southwest	NULL	10510853.8739
15	United Kingdom	GB	5012905.3656
16	United Kingdom	NULL	5012905.3656
17	NULL	NULL	40001565.9004

Using GROUP BY with ROLLUP

© Aptech Ltd. Session 9/ 12

Instructions to the Trainer(s):

- Using Slide 12, explain GROUP BY with ROLLUP operator.
- The GROUP BY clause is a SQL command that is used to group rows that have the same values.
- The GROUP BY clause is used in the SELECT statement. Optionally it is used in conjunction with aggregate functions to produce summary reports from the database. That's what it does, summarizing data from the database.
- The queries that contain the GROUP BY clause are called grouped queries and only return a single row for every grouped item.
- GROUP BY ROLLUP is an extension of the GROUP BY clause that produces sub-total rows (in addition to the grouped rows). Sub-total rows are rows that further aggregate whose values are derived by computing the same aggregate functions that were used to produce the grouped rows.

Aggregate Functions 1-3

Developers require to perform analysis across rows, such as counting rows, meeting specific criteria, or summarizing total sales for all orders.

Aggregate functions enable to accomplish it.

Aggregate functions ignore NULLs, except when using COUNT(*) .

Aggregate functions in a SELECT list do not generate a column alias.

Aggregate functions in a SELECT clause operate on all rows passed to the SELECT phase.

Instructions to the Trainer(s):

- Using Slide 13, explain the aggregate functions.
- Occasionally, developers may also require to perform analysis across rows, such as counting rows meeting specific criteria or summarizing total sales for all orders. Aggregate functions enable to accomplish this.
- Since aggregate functions return a single value, they can be used in SELECT statements where a single expression is used, such as SELECT, HAVING, and ORDER BY clauses.
- Aggregate functions ignore NULLs, except when using COUNT(*) .
- Aggregate functions in a SELECT list do not generate a column alias.
- Aggregate functions in a SELECT clause operate on all rows passed to the SELECT phase. If there is no GROUP BY clause, all rows will be summarized.

Function Name	Syntax	Description
AVG	AVG(<expression>)	Calculates the average of all the non-NULL numeric values in a column.
COUNT or COUNT_BIG	COUNT(*) or COUNT(<expression>)	When (*) is used, this function counts all rows, including those with NULL. The function returns count of non-NULL rows for the column when a column is specified as <expression>. The return value of COUNT function is an int. The return value of COUNT_BIG is a big_int.
MAX	MAX(<expression>)	Returns the largest number, latest date/time, or last occurring string.
MIN	MIN(<expression>)	Returns the smallest number, earliest date/time, or first occurring string.
SUM	SUM(<expression>)	Calculates the sum of all the non-NULL numeric values in a column.

© Aptech Ltd.

Session 9/14

Instructions to the Trainer(s):

- Using Slide 14, explain different aggregate functions.
- SQL Server provides many built-in aggregate functions. Commonly used functions are included in table shown on slide 14.
- Six SQL Aggregate Functions are as follows:
 - AVG – calculates the average of a set of values.
 - COUNT – counts rows in a specified table or view.
 - MIN – gets the minimum value in a set of values.
 - MAX – gets the maximum value in a set of values.
 - SUM – calculates the sum of values.

In-Class Question:**Question:** What is an aggregate query?**Answer:** An aggregate query is a method of deriving group and subgroup data by analysis of a set of individual data entries.

Aggregate Functions 3-3

The screenshot shows two separate SQL query results in a window titled 'Using Aggregate Functions'.

Using Aggregate Functions:

	AvgUnitPrice	MinQty	MaxDiscount
1	465.0934	1	0.40

Using Aggregate Functions with Non-Numeric Data:

	Earliest	Latest
1	2011-05-31 00:00:00.000	2014-06-30 00:00:00.000

© Aptech Ltd.

Session 9 / 15

Instructions to the Trainer(s):

- Using Slide 15, explain how aggregate functions can be used.
- When using aggregates in a SELECT clause, all columns referenced in the SELECT list must be used as inputs for an aggregate function or must be referenced in a GROUP BY clause. Failing this, there will be an error.
- This returns an error stating that the column Sales.SalesOrderDetail.SalesOrderID is invalid in the SELECT list because it is not contained in either an aggregate function or the GROUP BY clause. As the query is not using a GROUP BY clause, all rows will be treated as a single group. All columns, therefore, must be used as inputs to aggregate functions. To correct or prevent the error, one must remove SalesOrderID from the query.
- Besides using numeric data, aggregate expressions can also include date, time, and character data for summarizing.

In-Class Question:

Question: Which function is used to count the number of rows in a table?

Answer:

count(*) function is used to count the number of rows in a table.

Spatial Aggregates 1-3

- SQL Server provides several methods that help to aggregate two individual items of geometry or geography data.

Method	Description
STUnion	Returns an object that represents the union of a geometry/geography instance with another geometry/geography instance.
STIntersection	Returns an object that represents the points where a geometry/geography instance intersects another geometry/geography instance.
STConvexHull	Returns an object representing the convex hull of a geometry/geography instance. A set of points is called convex if for any two points, the entire segment is contained in the set. The convex hull of a set of points is the smallest convex set containing the set. For any given set of points, there is only one convex hull.

Spatial Aggregate Methods

Instructions to the Trainer(s):

- Using Slide 16, explain the spatial aggregates.
- SQL Server provides several methods that help to aggregate two individual items of geometry or geography data.
- Following are the methods of spatial aggregates:
 - **STUnion:** Returns an object that represents the union of geometry/geography instance with another geometry/geography instance.
 - **Intersection:** Returns an object that represents the points where a geometry/geography instance interacts another geometry/geography instance.
 - **STConvexHull:** Returns an object representing the convex hull of a geometry/geography instance.

Spatial Aggregates 2-3

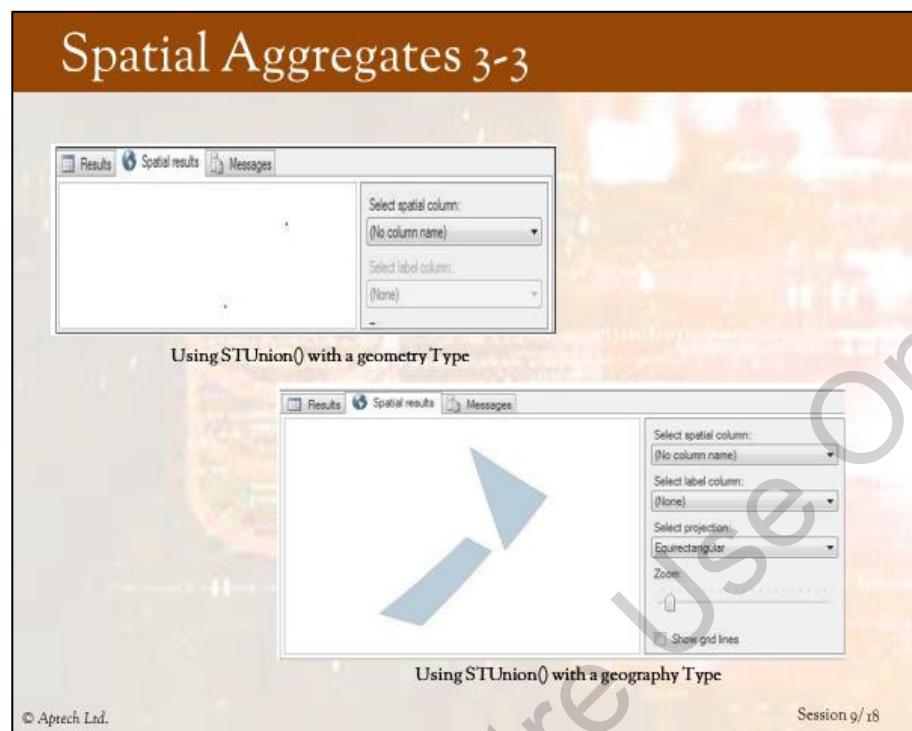
The diagram illustrates three spatial aggregation methods:

- STUnion()**: Shows two separate polygons being combined into a single merged polygon.
- STIntersection()**: Shows two overlapping polygons where their intersection area is highlighted.
- STConvexHull()**: Shows a set of points forming a convex hull, represented by a dashed line connecting the outermost points.

© Aptech Ltd. Session 9/17

Instructions to the Trainer(s):

- Using Slide 17, discuss different types of spatial aggregates.
- Slide 17 displays different types of spatial aggregates, such as STUnion, STIntersection, and STConvexHull.
- From the figure, students will be able to understand the working of each of these. Explain the code snippets which result in these outputs.
- The STUnion() combines two polygons using the STUnion() method resulting in a merged polygon.
- Using the STIntersection() with two polygons can lead to another polygon.
- A STConvexHull() can be formed from a set of points as displayed in Slide 17.



Instructions to the Trainer(s):

- Using Slide 18, explain STUnion() with a geometry and geography type.
- **STUnion (geometry Data Type):** This method always returns an object that represents the union of a geometry instance with another geometry instance.
- **STUnion (geography Data Type):** This method always returns null if the Spatial Reference Identifiers (SRIDs) of the geography instances do not match.

More Spatial Aggregates 1-5

➤ These aggregates are implemented as static methods, which work for either geography or geometry data types.

© Aptech Ltd. Session 9/19

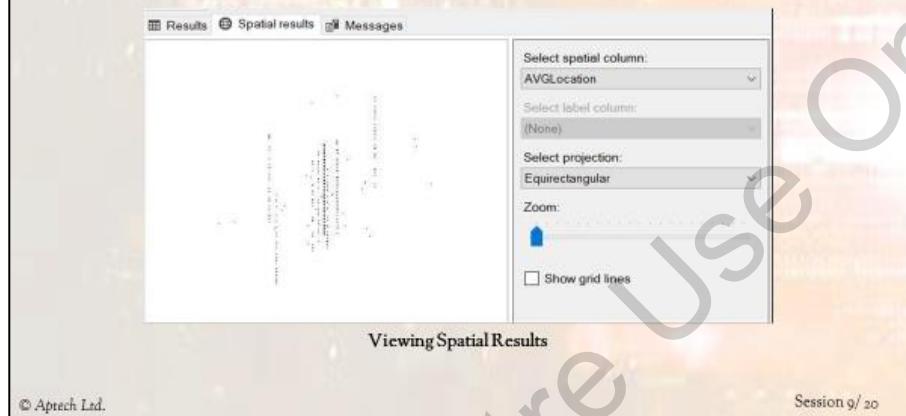
Instructions to the Trainer(s):

- Using Slide 19, explain the more spatial aggregates in SQL.
- SQL Server has introduced four new aggregates to the suite of spatial operators in SQL Server:
 - Union Aggregate
 - Envelope Aggregate
 - Collection Aggregate
 - Convex Hull Aggregate
- These aggregates are implemented as static methods, which work for either the geography or the geometry data types.
- Although aggregates are applicable to all classes of spatial data, they can be best described with polygons.

More Spatial Aggregates 2-5

Union Aggregate

- It performs a union operation on a set of geometry objects.
- It combines multiple spatial objects into a single spatial object, removing interior boundaries, where applicable.



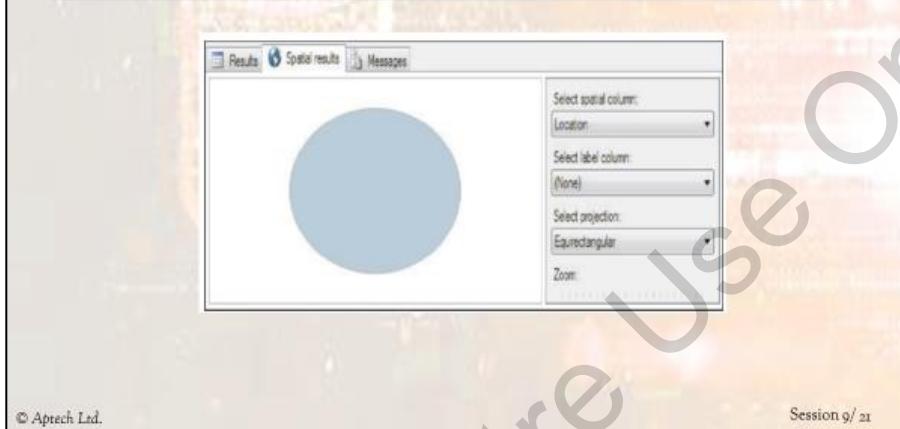
Instructions to the Trainer(s):

- Using Slide 20, explain the Union aggregate.
- It performs a union operation on a set of geometry objects. It combines multiple spatial objects into a single spatial object, removing interior boundaries, where applicable.
- The Union operator combines the results of two or more queries into a single result set that includes all the rows that belong to all queries in the Union.
- To view a visual representation of the spatial data, you can click the Spatial results tab in the output window.

More Spatial Aggregates 3-5

Envelope Aggregate

- The Envelope Aggregate returns a bounding area for a given set of geometry or geography objects.
- It exhibits different behaviors for geography and geometry types.



© Aptech Ltd.

Session 9/ 21

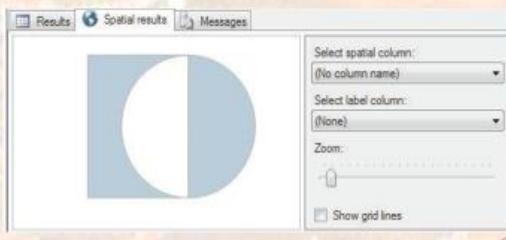
Instructions to the Trainer(s):

- Using Slide 21, explain the Envelope Aggregate.
- The envelope aggregate returns a bounding area for a given set of geometry or geography objects.
- The Envelope Aggregate exhibits different behaviors for geography and geometry types. Based on the type of object it is applied to, it returns different results.
- For the geometry type, the result is a 'traditional' rectangular polygon, which closely bounds the selected input objects. For the geography type, the result is a circular object, which loosely bounds the selected input objects.
- Furthermore, the circular object is defined using the new CurvePolygon feature.

More Spatial Aggregates 4-5

Collection Aggregate

- It returns a GeometryCollection/GeographyCollection instance with one geometry/geography part for each spatial object(s) in the selection set.



© Aptech Ltd.

Session 9/ 22

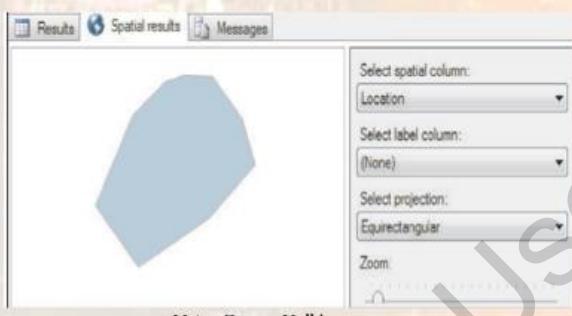
Instructions to the Trainer(s):

- Using Slide 22, explain the Collection Aggregate.
- It returns a GeometryCollection/GeographyCollection instance with one geometry/geography part for each spatial object(s) in the selection set.
- Slide 22 displays how it returns a GeometryCollection instance that contains a CurvePolygon and a Polygon.

More Spatial Aggregates 5-5

Convex Hull Aggregate

➤ It returns a convex hull polygon, which encloses one or more spatial objects for a given set of geometry/geography objects.



The screenshot shows a software window titled "Spatial results". On the left, there is a map view containing a blue convex hull polygon. On the right, there are several configuration options: "Select spatial column: Location", "Select label column: (None)", "Select projection: Equirectangular", and a "Zoom" button. Below the map, the text "Using ConvexHullAggregate" is displayed. At the bottom left is the copyright notice "© Aptech Ltd.", and at the bottom right is "Session 9/ 23".

Instructions to the Trainer(s):

- Using Slide 23, explain the ConvexHull aggregate.
- Tell the students that convex hull returns a convex hull polygon, which encloses one or more spatial objects for a given set of geometry/geography objects.

Subqueries I-2

- The outer query is called parent query and the inner query is called a subquery.
- The purpose of a subquery is to return results to the outer query.
- In other words, the inner query statement should return the column or columns used in the criteria of the outer query statement.

Using a Simple Subquery

	DueDate	ShipDate
1	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000
2	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000
3	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000
4	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000
5	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000
6	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000
7	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000
8	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000
9	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000
10	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000

© Aptech Ltd. Session 9/ 24

Instructions to the Trainer(s):

- Using Slide 24, explain the Subquery concept.
- Tell the students that SELECT statement or a query to return records that will be used as criteria for another SELECT statement or query.
- The outer query is called parent query and the inner query is called a subquery.
- The purpose of a subquery is to return results to the outer query.
- In other words, the inner query statement should return the column or columns used in the criteria of the outer query statement.
- The simplest form of a subquery is one that returns just one column. The parent query can use the results of this subquery using an = sign.

For more information on subqueries, refer to:

<https://docs.microsoft.com/en-us/sql/relational-databases/performance/subqueries?view=sql-server-ver15>

Subqueries 2-2

Based on results returned by inner query, a subquery can be classified as a **scalar subquery** or a **multi-valued subquery**:

Scalar subqueries return a single value. Here, the outer query must be written to process a single result.

Multi-valued subqueries return a result similar to a single-column table. Here, the outer query must be written to handle multiple possible results.

© Aptech Ltd.

Session 9/ 25

Instructions to the Trainer(s):

- Using Slide 25, explain the classification of subqueries.
- A subquery is also called an inner query or inner select, while the statement containing a subquery is also called an outer query or outer select.
- The inner query or subquery retrieves the most recent order date. This is then passed to the outer query, which displays due date and ship date for all the orders that were made on that particular date.
- Based on results returned by inner query, a subquery can be classified as:
 - Scalar subquery
 - Multi-valued subquery
- **Scalar Subqueries:** Scalar subqueries return a single value. Here, the outer query must be written to process a single result.
- **Multi-valued Subqueries:** They return result similar to single-column table. Here, the outer query must be written to handle multiple possible results.

Working with Multi-valued Queries

The ntext, text, and image data types cannot be used in the SELECT list of subqueries.

The SELECT list of a subquery introduced with a comparison operator can have only one expression or column name.

Subqueries that are introduced by a comparison operator not followed by the keyword ANY or ALL cannot include GROUP BY and HAVING clauses.

You cannot use DISTINCT keyword with subqueries that include GROUP BY.

You can specify ORDER BY only when TOP is also specified.

Instructions to the Trainer(s):

- Using Slide 26, discuss the working with multi-valued queries.
- Tell the students that they should remember following points when using subqueries:
 - The ntext, text, and image data types cannot be used in the SELECT list of subqueries.
 - The SELECT list of a subquery introduced with a comparison operator can have only one expression or column name.
 - Subqueries that are introduced by a comparison operator not followed by the keyword ANY or ALL cannot include GROUP BY and HAVING clauses.
 - You cannot use DISTINCT keyword with subqueries that include GROUP BY.
 - You can specify ORDER BY only when TOP is also specified.

Nested Subqueries

- A subquery that is defined inside another subquery is called a nested subquery.

For example:

Retrieve and display the names of persons from Canada

	Last Name	First Name
1	Vargas	Garrett
2	Saraiva	José

Output of Nested Subqueries

© Aptech Ltd.

Session 9 / 27

Instructions to the Trainer(s):

- Using Slide 27, explain the nested subqueries.
- A subquery that is defined inside another subquery is called a nested subquery.
- Consider that you wanted to retrieve and display the names of persons from Canada. There is no direct way to retrieve this information since the Sales.SalesTerritory table is not related to Person.Person table. Hence, a nested subquery is used.

Correlated Queries

- When a subquery takes parameters from its parent query, it is known as Correlated subquery.

For example:

Retrieve all the business entity ids of persons whose contact information was last modified not earlier than 2019.

BusinessEntityID
1 292
2 294
3 296
4 298
5 300
6 302
7 304

Output of Correlated Queries

Instructions to the Trainer(s):

- Using Slide 28, explain the correlated queries.
- In many queries containing subqueries, the subquery must be evaluated only once to provide the values required by the parent query. This is because in most of the queries, the subquery makes no reference to the parent query, so the value in the subquery remains constant.
- However, if the subquery refers to a parent query, the subquery must be reevaluated for every iteration in the parent query. This is because the search criterion in the subquery is dependent upon the value of a particular record in the parent query.
- When a subquery takes parameters from its parent query, it is known as Correlated subquery.

Joins

- Specifying the column from each table to be used for the join. A typical join specifies a foreign key from one table and its associated key in the other table.
- Specifying a logical operator such as =, <> to be used in comparing values from the columns.

Output of Join

	FirstName	LastName	JobTitle
1	Ken	Sánchez	Chief Executive Officer
2	Tem	Duffy	Vice President of Engineering
3	Roberto	Tamburro	Engineering Manager
4	Rob	Walters	Senior Tool Designer
5	Gail	Erickson	Design Engineer
6	Jossef	Goldberg	Design Engineer
7	Dylan	Miller	Research and Development Manager

Three types of joins:

© Aptech Ltd. Session 9/ 29

Instructions to the Trainer(s):

- Using Slide 29, explain the JOIN statement.
- Joins are used to retrieve data from two or more tables based on a logical relationship between tables.
- A join typically specifies foreign key relationship between the tables. It defines the manner in which two tables are related in a query by:
 - Specifying the column from each table to be used for the join. A typical join specifies a foreign key from one table and its associated key in the other table.
 - Specifying a logical operator such as =, <> to be used in comparing values from the columns.
 - Joins can be specified in either the FROM or WHERE clauses.
- There are three types of joins as follows:
 - Inner Joins
 - Outer Joins
 - Self-Joins

In-Class Question:

Question: What is use of joins?

Answer: Joins are used to retrieve data from two or more tables based on a logical relationship between tables.

Inner Join

- An inner join is formed when records from two tables are combined only if the rows from both the tables are matched based on a common column

Following is the syntax of an inner join:

```
SELECT<ColumnName1>,<ColumnName2>...<ColumnNameN>FROM Table_A  
AS Table_Alias_A  
INNER JOIN  
Table_B AS Table_Alias_B ON  
Table Alias A.<CommonColumn>=Table Alias B.<CommonColumn>
```

Instructions to the Trainer(s):

- Using Slide 30, explain the concept of Inner join.
- An inner join is formed when records from two tables are combined only if the rows from both the tables are matched based on a common column.
- In the code snippet shown in the Learner Guide, an inner join is constructed between Person.Person and HumanResources.Employee based on common business entity ids. Here again, the two tables are given aliases of A and B respectively. The output is the same as shown in figure.

Outer Join

- Outer joins are join statements that return all rows from at least one of the tables specified in the FROM clause, as long as those rows meet any WHERE or HAVING conditions of the SELECT statement.

Two types of commonly used outer joins

Left Outer
Join

Right Outer
Join

Instructions to the Trainer(s):

- Using Slide 31, explain the Outer join concept.
- Outer joins are join statements that return all rows from at least one of the tables specified in the FROM clause, as long as those rows meet any WHERE or HAVING conditions of the SELECT statement.
- Two types of commonly used outer joins are as follows:
 - Left Outer Join
 - Right Outer Join

Self-Join

- A self-join is used to find records in a table that are related to other records in the same table.
- A table is joined to itself in a self-join.

	ProductID	Color	Name	Name
1	317	Black	LL Crankarm	LL Crankarm
2	317	Black	LL Crankarm	ML Crankarm
3	317	Black	LL Crankarm	HL Crankarm
4	317	Black	LL Crankarm	Chainring
5	317	Black	LL Crankarm	HL Road Frame - Black, 58
6	317	Black	LL Crankarm	Sport-100 Helmet, Black
7	317	Black	LL Crankarm	Road-750 Black, 44
8	317	Black	LL Crankarm	Road-750 Black, 48
9	317	Black	LL Crankarm	Road-750 Black, 52
10	317	Black	LL Crankarm	Road-750 Black, 58

Self Join Example

Instructions to the Trainer(s):

- Using Slide 32, explain the self-join.
- A self-join is used to find records in a table that are related to other records in the same table. A table is joined to itself in a self-join.

MERGE Statement

The diagram illustrates the three actions of the MERGE Statement:

- Insert a new row from the source if the row is missing in the target table
- Update a target row if a record already exists in the source table
- Delete a target row if the row is missing in the source table

Below the diagram, a list of tasks to be performed by the MERGE statement is provided:

- Compare last and first names of customers from both source and target tables
- Update customer information in target table if the last and first names match
- Insert new records in target table if the last and first names in source table do not exist in target table
- Delete existing records in target table if the last and first names do not match with those of source table

© Aptech Ltd. Session 9/33

Instructions to the Trainer(s):

- Using Slide 33, explain the MERGE statement.
- The MERGE statement allows you to maintain a target table based on certain join conditions on a source table using a single statement. You can now perform following actions in one MERGE statement:
 - Insert a new row from the source if the row is missing in the target.
 - Update a target row if a record already exists in the source table.
 - Delete a target row if the row is missing in the source table.
- Consider that you want to:
 - Compare last and first names of customers from both source and target tables.
 - Update customer information in target table if the last and first names match.
 - Insert new records in target table if the last and first names in source table do not exist in target table.
 - Delete existing records in target table if the last and first names do not match with those of source table.
- The MERGE statement accomplishes the tasks in a single statement.
- MERGE also allows you to optionally display those records that were inserted, updated, or deleted by using an OUTPUT clause.

Common Table Expressions (CTEs)

➤ A CTE is similar to a temporary resultset defined within the execution scope of a single SELECT, INSERT, UPDATE, DELETE, or CREATE VIEW statement. A CTE is a named expression defined in a query.

➤ A CTE that include references to itself is called a **recursive CTE**.

CTEs are limited in scope to the execution of the outer query. Hence, when the outer query ends, the lifetime of the CTE will end.

You must define a name for a CTE and also, define unique names for each of the columns referenced in the SELECT clause of the CTE.

It is possible to use inline or external aliases for columns in CTEs.

A single CTE can be referenced multiple times in the same query with one definition.

Instructions to the Trainer(s):

- Using Slide 34, explain the Common Table Expressions (CTEs) to students.
- A Common Table Expression (CTE) is similar to a temporary resultset defined within the execution scope of a single SELECT, INSERT, UPDATE, DELETE, or CREATE VIEW statement.
- A CTE is a named expression defined in a query.
- A CTE is defined at the start of a query and can be referenced several times in the outer query.
- A CTE that include references to itself is called as a recursive CTE.
- Key advantages of CTEs are improved readability and ease in maintenance of complex queries.

Following guidelines must be remembered while defining CTEs:

- CTEs are limited in scope to the execution of the outer query. Hence, when the outer query ends, the lifetime of the CTE will end.
- You must define a name for a CTE and also, define unique names for each of the columns referenced in the SELECT clause of the CTE.
- It is possible to use inline or external aliases for columns in CTEs.
- A single CTE can be referenced multiple times in the same query with one definition.
- Multiple CTEs can also be defined in the same WITH clause.

Combining Data Using SET Operators

- SQL Server 2019 provides certain keywords, also called as operators, to combine data from multiple tables.

These operators are as follows:

- UNION
- INTERSECT
- EXCEPT

Instructions to the Trainer(s):

- Using Slide 35, explain the combining data set using SET Operators.
- SQL Server 2019 provides certain keywords, also called as operators, to combine data from multiple tables.
- These operators are as follows:
 - UNION
 - INTERSECT
 - EXCEPT

UNION Operator

- Results from two different query statements can be combined into a single resultset using UNION operator.
 - Query statements must have compatible column types and equal number of columns.
-
- The column names can be different in each statement, but the data types must be compatible.
 - By compatible data types, it means that it should be possible to convert the contents of one of the columns into another.

Following is the syntax of the UNION operator.

```
Query Statement1 UNION [ALL] Query Statement2
```

© Aptech Ltd.

Session 9/ 36

Instructions to the Trainer(s):

- Using Slide 36, explain the UNION Operator concept.
- The results from two different query statements can be combined into a single resultset using the UNION operator.
- The query statements must have compatible column types and equal number of columns. The column names can be different in each statement, but the data types must be compatible.
- By compatible data types, it means that it should be possible to convert the contents of one of the columns into another.
- For example, if one of the query statements has an int data type and the other query statement has a money data type, they are compatible and a union can take place between them because the int data can be converted into money data.

INTERSECT Operator

- The INTERSECT operator is used with two query statements to return a distinct set of rows that are common to both the query statements.

The basic rules for using INTERSECT are as follows:

- Number of columns and order in which they are given must be same in both queries.
- Data types of the columns being used must be compatible.

Instructions to the Trainer(s):

- Using Slide 37, explain the INTERSECT Operator.
- The INTERSECT Operator is used with two query statements to return a distinct set of rows that are common to both the query statements.
- The basic rules for using INTERSECT are as follows:
 - Number of columns and order in which they are given must be same in both the queries.
 - Data types of the column being used must be compatible.

EXCEPT Operator

- The EXCEPT operator returns all of the distinct rows from the query given on left of the EXCEPT operator and removes all rows from the resultset that match rows on right of the EXCEPT operator.

Instructions to the Trainer(s):

- Using Slide 38, explain the EXCEPT Operator.
- The EXCEPT operator returns all of the distinct rows from the query given on the left of the EXCEPT operator and removes all the rows from the resultset that match the rows on the right of the EXCEPT operator.
- The two rules that apply to INTERSECT operator are also applicable for EXCEPT operator.

Pivoting and Grouping Set Operations

➤ The process of transforming data from a row-based orientation to a column-based orientation is called pivoting.

➤ The PIVOT and UNPIVOT operators of SQL Server help to change the orientation of data from column-oriented to row-oriented and vice versa.

Instructions to the Trainer(s):

- Using Slide 39, explain the Pivoting and Grouping Set Operations.
- The process of transforming data from a row-based orientation to a column-based orientation is called pivoting.
- The PIVOT AND UNPIVOT operators of SQL Server help to change the orientation of data from column-oriented to row-oriented and vice versa.

PIVOT Operator

Grouping	Spreading	Aggregation
In the FROM clause, the input columns must be provided. The PIVOT operator uses those columns to determine which column(s) to use for grouping the data for aggregation.	Here, a comma-separated list of values that occur in the source data is provided that will be used as the column headings for the pivoted data.	An aggregation function, such as SUM, to be performed on the grouped rows.

Grouping without PIVOT

	TotalSalesYTD	Name
1	7887186.7882	Northwest
2	2402176.8476	Northeast
3	3072175.118	Central
4	10510853.8739	Southwest
5	2538667.2515	Southeast

© Aptech Ltd. Session 9/ 40

Instructions to the Trainer(s):

- Using Slide 40, explain the PIVOT Operator.
- Consider a scenario where data must be displayed in a different orientation than it is stored in, in terms of row and column layout.
- The process of transforming data from a row-based orientation to a column-based orientation is called pivoting.
- The PIVOT and UNPIVOT operators of SQL Server help to change the orientation of data from column-oriented to row-oriented and vice versa. This is accomplished by consolidating values present in a column to a list of distinct values and then projecting that list in the form of column headings.
- In simpler terms, to use the PIVOT operator, you must supply three elements to the operator:
 - **Grouping:** In the FROM clause, the input columns must be provided. The PIVOT operator uses those columns to determine which column(s) to use for grouping the data for aggregation.
 - **Spreading:** Here, a comma-separated list of values that occur in the source data is provided that will be used as the column headings for the pivoted data.
 - **Aggregation:** An aggregation function, such as SUM, to be performed on the grouped rows. Consider an example to understand the PIVOT operator. Code snippet is shown without the PIVOT operator and demonstrates a simple GROUP BY aggregation. As the number of records would be huge, the resultset is limited to 5 by specifying TOP 5.

UNPIVOT Operator

Source columns to be unpivoted

A name for the new column that will display the unpivoted values

A name for the column that will display the names of the unpivoted values

	SalesYear	TotalSales
1	2011	151706131.5475
2	2012	862786335.1754
3	2013	1190722789.0552
4	2014	391255200.8993

Output for UNPIVOT

© Aptech Ltd.

Session 9 / 41

Instructions to the Trainer(s):

- Using Slide 41, explain the UNPIVOT Operator.
- UNPIVOT performs almost the reverse operation of PIVOT, by rotating columns into rows.
- Unpivoting does not restore the original data. Detail-level data was lost during the aggregation process in the original pivot.
- UNPIVOT has no ability to allocate values to return to the original detail values. Instead of turning rows into columns, unpivoting results in columns being transformed into rows.
- SQL Server provides the UNPIVOT table operator to return a row-oriented tabular display from a pivoted data.
- When unpivoting data, one or more columns are defined as the source to be converted into rows.
- The data in those columns is spread, or split, into one or more new rows, depending on how many columns are being unpivoted.

- To use the UNPIVOT operator, you must provide three elements as follows:
 - Source columns to be unpivoted.
 - A name for the new column that will display the unpivoted values.
 - A name for the column that will display the names of the unpivoted values.

EXCEPT Operator

- The EXCEPT operator returns all the distinct rows from the query given on left of the EXCEPT operator and removes all rows from the resultset that match rows on right of the EXCEPT operator.

Instructions to the Trainer(s):

- Using Slide 42, explain the EXCEPT Operator.
- The EXCEPT operator returns all the distinct rows from the query given on left of the EXCEPT operator and removes all rows from the resultset that match rows on right of the EXCEPT operator.

Summary

- The GROUP BY clause and aggregate functions enable to group and/or aggregate data together in order to present summarized information.
- Spatial aggregate functions were first introduced in SQL Server 2012 and are supported in SQL Server 2019 as well.
- A subquery allows the resultset of one SELECT statement to be used as criteria for another SELECT statement.
- Joins help you to combine column data from two or more tables based on a logical relationship between the tables.
- Set operators such as UNION and INTERSECT help you to combine row data from two or more tables.
- The PIVOT and UNPIVOT operators help to change the orientation of data from column-oriented to row-oriented and vice versa.
- The GROUPING SET subclause of the GROUP BY clause helps to specify multiple groupings in a single query.

Instructions to the Trainer(s):

- Show students Slide 43.
- Summarize the session by reading out each point on the slide.

Session 10: Views, Stored Procedures, and Querying Metadata

10.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. Prepare a question or two that will be a key point to relate the current session objectives.

10.1.1 Teaching Skills

To teach this session, you should be well versed with views, stored procedures, and querying metadata.

You should teach the concepts in the theory class using the images provided. For teaching in the class, you are expected to use slides and LCD projectors.

In-Class Activities

Follow the order given here during In-Class activities.

Slide 2

Objectives

- Define views
- Describe the technique to create, alter, and drop views
- Define stored procedures and its types
- Describe the procedure to create, alter, and execute stored procedures
- Describe nested stored procedures
- Describe querying SQL Server metadata System Catalog views and functions

© Aptech Ltd. Session 10 / 2

Instructions to the Trainer(s):

Give students a brief overview of the current session through the session objectives listed in Slide 2.

10.2 In-Class Explanations

Slide 3

Introduction

- An SQL Server database has two main categories of objects:
 - Those that store data
 - Those that access, manipulate, or provide access to data
- Views and stored procedures belong to the latter category.

The diagram illustrates the creation of a view from two tables, Table 1 and Table 2. It shows a flow from the tables through a 'Create a View' step to a resulting 'View created from Table-1 and Table-2'. This view is then connected to a 'UI' (User Interface) and a 'Database'. The Database is represented by three cylinders labeled 'Data', 'Stored Procedure (Read, Insert, Update, Delete)', and 'SQL queries'. Arrows indicate the flow of data between the UI, the view, and the database components.

© Aptech Ltd. Session 10 / 3

Instructions to the Trainer(s):

- Introduce the session using Slide 3, explain the SQL Server database to students.
- An SQL Server database has two main categories of objects:
 - those that store data
 - those that access, manipulate, or provide access to data
- Views and stored procedures belong to this latter category.

For more information, refer to:

<https://www.sqlservertutorial.net/getting-started/what-is-sql-server/>

Views

The diagram features a central image of a bowl of soup, symbolizing a virtual table. Surrounding the bowl are four colored 3D blocks (blue, red, green, purple) representing base tables. Four callout points with text labels extend from the blocks towards the bowl:

- A view is a virtual table that is made up of selected columns from one or more tables.
- The tables from which the view is created are referred to as base tables.
- A view can also include columns from other views created in the same or a different database.
- A view can have a maximum of 1,024 columns.

© Aptech Ltd. Session 10/4

Instructions to the Trainer(s):

- Using Slide 4, explain the concept of views.
- A view is a virtual table that is made up of selected columns from one or more tables.
- The tables from which the view is created are referred to as base tables. These base tables can be from different databases.
- A view can also include columns from other views created in the same or a different database. A view can have a maximum of 1024 columns.
- The data inside the view comes from the base tables that are referenced in the view definition.
- The rows and columns of views are created dynamically when the view is referenced.

In-Class Question:

Question: What is a view?

Answer: A view is a virtual table that is made up of selected columns from one or more tables.

Creating Views

A user can create a view using columns from tables or other views only if the user has permission to access these tables and views.

A view is created using CREATE VIEW statement and it can be created only in the current database.

SQL Server verifies the existence of objects that are referenced in the view definition.

ProductID	ProductNumber	Name	SafetyStockLevel
1	AR-5381	Adjustable Race	1000
2	BA-8327	Bearing Ball	1000
3	BE-2349	BB Ball Bearing	800
4	BE-2908	Headset Ball Bearings	800
5	316	Blade	800
6	317	LL Crankarm	500
7	318	CA-6738	500
8	319	CA-7457	500
9	320	CB-2903	1000
10	321	Chaining Bolts	1000
		Chaining Nut	1000

Records from a View

© Aptech Ltd. Session 10 / 5

Instructions to the Trainer(s):

- A user can create a view using columns from tables or other views only if the user has permission to access these tables and views.
- A view is created using the CREATE VIEW statement and it can be created only in the current database.
- SQL Server verifies the existence of objects that are referenced in the view definition.
- While creating a view, test the SELECT statement that defines the view to make sure that SQL Server returns the expected result.
- The view can be created only after the SELECT statement is tested and the resultset has been verified.

For more information on creating views, refer to:

https://www.w3schools.com/sql/sql_view.asp

<https://www.sqlshack.com/create-view-sql-creating-views-in-sql-server/>

Creating Views Using JOIN Keyword 1-2

The JOIN keyword can also be used to create views.

The CREATE VIEW statement is used along with JOIN keyword to create a view using columns from multiple tables.

Syntax:

```
CREATEVIEW<view_name> AS  
SELECT* FROMtable_name1 JOIN  
table_name2  
ONtable_name1.column_name=table_name2.column_name
```

where,

view_name: specifies the name of the view.

table_name1: specifies the name of first table.

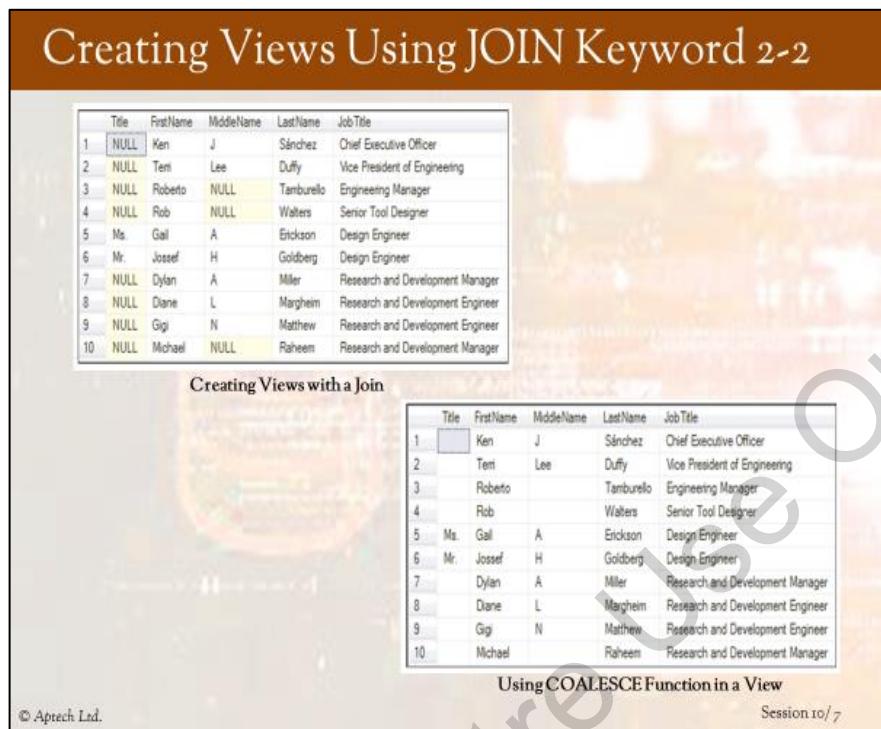
JOIN: specifies that two tables are joined using JOIN keyword.

table_name2: specifies the name of the second table.

Instructions to the Trainer(s):

- Using Slide 6, explain creating views using JOIN Keyword.
- The JOIN keyword can also be used to create views. The CREATE VIEW statement is used along with the JOIN keyword to create a view using columns from multiple tables.
- The Code Snippet in the Learner Guide illustrates how to create a view named vwPersonDetails with specific columns from the Person and Employee tables. The JOIN and ON keywords join the two tables based on BusinessEntityID column.

Creating Views Using JOIN Keyword 2-2



Creating Views with a Join

	Title	FirstName	MiddleName	LastName	JobTitle
1	NULL	Ken	J	Sánchez	Chief Executive Officer
2	NULL	Terri	Lee	Duffy	Vice President of Engineering
3	NULL	Roberto	NULL	Tamburello	Engineering Manager
4	NULL	Rob	NULL	Walters	Senior Tool Designer
5	Ms.	Gall	A	Erickson	Design Engineer
6	Mr.	Jossef	H	Goldberg	Design Engineer
7	NULL	Dylan	A	Miller	Research and Development Manager
8	NULL	Diane	L	Margheim	Research and Development Engineer
9	NULL	Gigi	N	Matthew	Research and Development Engineer
10	NULL	Michael	NULL	Raheem	Research and Development Manager

Using COALESCE Function in a View

© Aptech Ltd.

Session 10 / 7

Instructions to the Trainer(s):

- Using Slide 7, students will be able to understand the working of how views are created using JOIN.
- Slide 7 displays an example of how views are created using the JOIN keyword. This view will contain the columns Title, FirstName, MiddleName, and LastName from the Person table and JobTitle from the Employee table. Once the view is created, you can retrieve records from it, and manipulate and modify records as well.
- As shown in figure on Slide 7, all the rows may not have values for the Title or MiddleName columns - some may have NULL in them. A person seeing this output may not be able to comprehend the meaning of the NULL values. Hence, to replace all the NULL values in the output with a null string, the COALESCE () function can be used as shown in Code Snippet on Slide 7.

Guidelines and Restrictions on Views

- A view is created only in current database. The base tables and views from which the view is created can be from other databases or servers.
- View names must be unique and cannot be same as table names in the schema.
- A view cannot be created on temporary tables.
- A view cannot have a full-text index.
- A view cannot contain DEFAULT definition.
- The CREATE VIEW statement can include ORDER BY clause only if TOP keyword is used.
- Views cannot reference more than 1,024 columns.
- The CREATE VIEW statement cannot include INTO keyword.
- The CREATE VIEW statement cannot be combined with other Transact-SQL statements in a single batch.

© Aptech Ltd. Session 10 / 8

Instructions to the Trainer(s):

- Using Slide 8, explain the guidelines and restrictions on views.
- A view can be created using the CREATE VIEW command.
- Before creating a view, following guidelines and restrictions should be considered:
 - A view is created only in the current database. The base tables and views from which the view is created can be from other databases or servers.
 - View names must be unique and cannot be the same as the table names in the schema.
 - A view cannot be created on temporary tables.
 - A view cannot have a full-text index.
 - A view cannot contain the DEFAULT definition.
 - The CREATE VIEW statement can include the ORDER BY clause only if the TOP keyword is used.
 - Views cannot reference more than 1024 columns.
 - The CREATE VIEW statement cannot include the INTO keyword.
 - The CREATE VIEW statement cannot be combined with other Transact-SQL statements in a single batch.
- If a view contains columns that have values derived from an expression, such columns have to be given alias names. Also, if a view contains similarly-named columns from different tables, to distinguish these columns, alias names must be specified.

Modifying Data through Views

Views can be used to modify data in database tables.

Data can be inserted, modified, or deleted through a view using following statements:

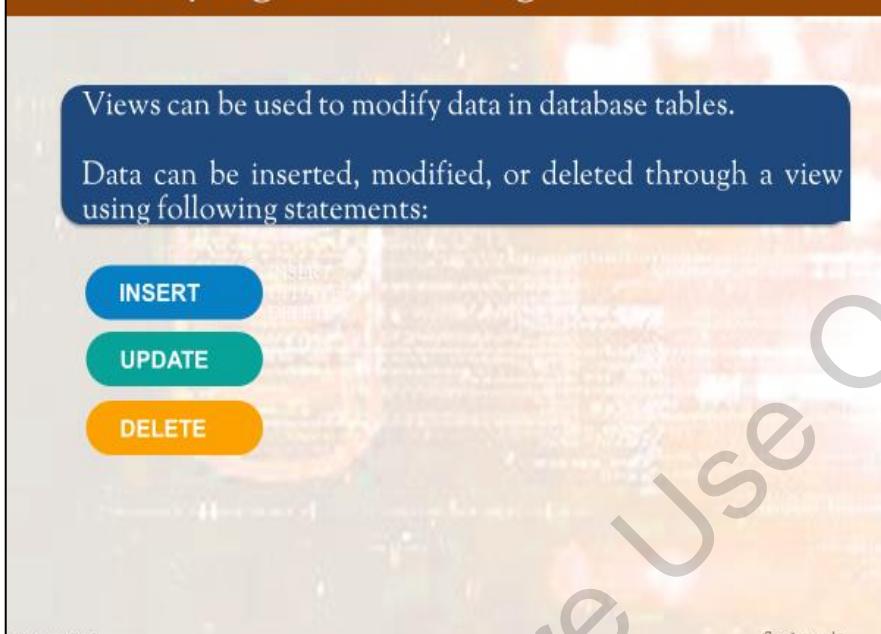
INSERT

UPDATE

DELETE

© Aptech Ltd.

Session 10/ 9



Instructions to the Trainer(s):

- Using Slide 9, explain how modification of data can be done through views.
- Views can be used to modify data in database tables.
- Data can be inserted, modified, or deleted through a view using following statements:
 - INSERT
 - UPDATE
 - DELETE

INSERT with Views

- The INSERT statement is used to add a new row to a table or a view.
- During the execution of the statement, if the value for a column is not provided, the SQL Server Database Engine must provide a value based on the definition of the column.
- The value for the column is provided automatically if the column:

Has an IDENTITY property

Has a default value specified

Has a timestamp data type

Takes null values

Is a computed column

© Aptech Ltd.

Session 10/10

Instructions to the Trainer(s):

- Using Slide 10, explain the INSERT with Views.
- The INSERT statement is used to add a new row to a table or a view.
- During the execution of the statement, if the value for a column is not provided, the SQL Server Database Engine must provide a value based on the definition of the column. If the Database Engine is unable to provide this value, then the new row will not be added.
- The value for the column is provided automatically if:
 - The column has an IDENTITY property.
 - The column has a default value specified.
 - The column has a timestamp data type.
 - The column takes null values.
 - The column is a computed column.
- While using the INSERT statement on a view, if any rules are violated, the record is not inserted.

UPDATE with Views

UPDATE statement can be used to change data in a view.

Updating a view also updates underlying table.

The value of a column with an IDENTITY property cannot be updated.

Records cannot be updated if the base table contains a TIMESTAMP column.

When there is a self-join with the same view or base table, the UPDATE statement does not work.

While updating a row, if a constraint or rule is violated, the statement is terminated, an error is returned, and no records are updated.

Instructions to the Trainer(s):

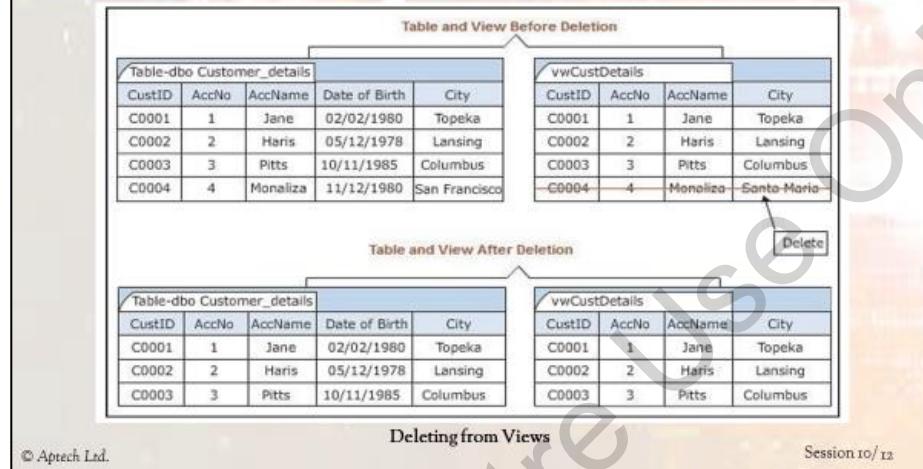
- Using Slide 11, explain how to update with views.
- The UPDATE statement can be used to change the data in a view. Updating a view also updates the underlying table.

Following rules and guidelines must be followed when using the UPDATE statement:

- The value of a column with an IDENTITY property cannot be updated.
- Records cannot be updated if the base table contains a TIMESTAMP column.
- When there is a self-join with the same view or base table, the UPDATE statement does not work.
- While updating a row, if a constraint or rule is violated, the statement is terminated, an error is returned, and no records are updated.

DELETE with Views

- SQL Server enables you to delete rows from a view. Rows can be deleted from the view using the DELETE statement.
- When rows are deleted from a view, corresponding rows are also deleted from the base table.



Instructions to the Trainer(s):

- Using Slide 12, explain how to delete with views.
- SQL Server enables you to delete rows from a view.
- Rows can be deleted from the view using the DELETE statement.
- When rows are deleted from a view, corresponding rows are deleted from the base table.
- For example, consider a view `vwCustDetails` that lists the account information of different customers. When a customer closes the account, the details of this customer must be deleted. This is done using the DELETE statement.

Altering Views

- A view can be modified or altered by dropping and recreating it or executing the ALTER VIEW statement.
- ALTER VIEW can be applied to indexed views; however, it unconditionally drops all indexes on the view.
- Views are often altered when a user requests for additional information or makes changes in the underlying table definition.

Instructions to the Trainer(s):

- Using Slide 13, explain how to alter a view.
- A view can be modified or altered by dropping and recreating it or executing the ALTER VIEW statement.
- The ALTER VIEW statement modifies an existing view without having to reorganize its permissions and other properties.
- ALTER VIEW can be applied to indexed views; however, it unconditionally drops all indexes on the view.
- Views are often altered when a user requests for additional information or makes changes in the underlying table definition.

Dropping Views

A view can be removed from the database if it is no longer required. This is done using the DROP VIEW statement.

The definition of the view and other information associated with the view is deleted from the system catalog.

© Aptech Ltd.

Session 10 / 14

Instructions to the Trainer(s):

- Using Slide 14, explain how to drop a view.
- A view can be removed from the database if it is no longer required. This is done using the DROP VIEW statement.
- When a view is dropped, the data in the base tables remains unaffected.
- The definition of the view and other information associated with the view is deleted from the system catalog.
- All permissions for the view are also deleted.
- If a user queries any view that references the dropped view, the user receives an error message.

In-Class Question:

Question: Does DROP TABLE drop views?

Answer: DROP TABLE always removes any indexes, rules, triggers, and constraints that exist for the target table.

Definition of a View

➤ The definition of a view helps to understand how its data is derived from the source tables.

Syntax:

```
sp_helptext<view_name>
```

Results

Text
1 CREATE VIEW vwEmployee_Personal_Details AS
2 SELECT e1.EmplID, FirstName, LastName, Designatio...
3 JOIN Employee_Salary_Details e2
4 ON e1.EmplID = e2.EmplID

Using sp_helptext to display View Definitions

© Aptech Ltd. Session 10 / 15

Instructions to the Trainer(s):

- Using Slide 15, explain the definition of view.
- The definition of a view helps to understand how its data is derived from the source tables. There are certain system stored procedures that help to retrieve view definitions.
- The `sp_helptext` stored procedure displays view related information when the name of the view is given as its parameter. Information about the definition of a view can be obtained if such information is not encrypted.

Creating a View Using Built-in Functions

Views can be created using built-in functions of SQL Server.

When functions are used, the derived column must include the column name in the CREATE VIEW statement.

	ProductName	AverageRate
1	Hard Disk Drive	3570.00
2	Portable Hard Drive	5580.00

Using Built-in Functions with Views

Instructions to the Trainer(s):

- Using Slide 16, explain how to create view using built-in functions.
- Views can be created using built-in functions of SQL Server.
- When functions are used, the derived column must include the column name in the CREATE VIEW statement.

CHECK OPTION

The CHECK OPTION is an option associated with the CREATE VIEW statement.

It is used to ensure that all the updates in the view satisfy the conditions mentioned in the view definition.

If the conditions are not satisfied, the database engine returns an error.

Syntax:

```
CREATE VIEW <view_name>
AS select_statement [WITHCHECKOPTION]
```

Instructions to the Trainer(s):

- Using Slide 17, explain the CHECK OPTION.
- The CHECK OPTION is an option associated with the CREATE VIEW statement. It is used to ensure that all the updates in the view satisfy the conditions mentioned in the view definition.
- If the conditions are not satisfied, the database engine returns an error.
- Thus, the CHECK OPTION is used to enforce domain integrity; it checks the definition of the view to see that the WHERE conditions in the SELECT statement is not violated.

In-Class Question:

Question: What is the use of CHECK OPTION?

Answer: The CHECK OPTION is used to ensure that all the updates in the view satisfy the conditions mentioned in the view definition.

SCHEMABINDING Option

- A view can be bound to the schema of the base table using the SCHEMABINDING option.
- This option can be used with CREATE VIEW or ALTER VIEW statements.

- The view definition must be first modified or deleted to remove dependencies on the table that is to be modified.

Instructions to the Trainer(s):

- Using Slide 18, explain the SCHEMABINDING option.
- A view can be bound to the schema of the base table using the SCHEMABINDING option. This option can be used with CREATE VIEW or ALTER VIEW statements.
- When SCHEMABINDING option is specified, the base table or tables cannot be modified that would affect the view definition.
- The view definition must be first modified or deleted to remove dependencies on the table that is to be modified.
- While using the SCHEMABINDING option in a view, you must specify the schema name along with the object name in the SELECT statement.

Using sp_refreshview

During the creation of a view, the SCHEMABINDING option is used to bind the view to the schema of the tables that are included in the view.

If changes are made to the underlying objects (tables or views) on which the view depends, the sp_refreshview stored procedure should be executed.

The sp_refreshview stored procedure updates the metadata for the view.

© Aptech Ltd.

Session 10/10

Instructions to the Trainer(s):

- Using Slide 19, explain how to use sp_refreshview.
- During the creation of a view, the SCHEMABINDING option is used to bind the view to the schema of the tables that are included in the view.
- However, a view can also be created without selecting the SCHEMABINDING option. In such a case, if changes are made to the underlying objects (tables or views) on which the view depends, the sp_refreshview stored procedure should be executed.
- The sp_refreshview stored procedure updates the metadata for the view.
- If the sp_refreshview procedure is not executed, the metadata of the view is not updated to reflect the changes in the base tables. This results in the generation of unexpected results when the view is queried.
- The sp_refreshview stored procedure returns code value zero if the execution is successful or returns a non-zero number in case the execution has failed.

Stored Procedures 1-2

- A stored procedure is a group of Transact-SQL statements that act as a single block of code that performs a specific task.
 - This block of code is identified by an assigned name and is stored in the database in a compiled form.
 - A stored procedure may also be a reference to a .NET Framework Common Language Runtime (CLR) method.
-
- Stored procedures are useful when repetitive tasks have to be performed. This eliminates the need for repetitively typing out multiple Transact-SQL statements and then repetitively compiling them.

Instructions to the Trainer(s):

- Using Slide 20, explain the stored procedures concept.
- A stored procedure is a group of Transact-SQL statements that act as a single block of code that performs a specific task. This block of code is identified by an assigned name and is stored in the database in a compiled form. A stored procedure may also be a reference to a .NET Framework Common Language Runtime (CLR) method.
- Stored procedures are useful when repetitive tasks have to be performed. This eliminates the necessity for repetitively typing out multiple Transact-SQL statements and then repetitively compiling them.
- Stored procedures can accept values in the form of input parameters and return output values as defined by the output parameters.

Stored Procedures 2-2

The diagram consists of four downward-pointing chevrons of increasing size from top to bottom, each pointing to a box containing a benefit of stored procedures. The first chevron is green and labeled 'Improved Security'. The second is also green and labeled 'Precompiled Execution'. The third is yellow and labeled 'Reduced Client/Server Traffic'. The fourth is orange and labeled 'Reuse of code'.

Improved Security: The database administrator can improve security by associating database privileges with stored procedures. Users can be given permission to execute a stored procedure even if user does not have permission to access tables or views.

Precompiled Execution: Stored procedures are compiled during the first execution. For every subsequent execution, SQL Server reuses this precompiled version. This reduces the time and resources required for compilation.

Reduced Client/Server Traffic: Stored procedures help in reducing network traffic. When Transact-SQL statements are executed individually, there is network usage separately for execution of each statement. When a stored procedure is executed, Transact-SQL statements are executed together as a single unit. This reduces network traffic.

Reuse of code: Stored procedures can be used multiple times. This eliminates need to repetitively type out hundreds of Transact-SQL statements every time a similar task is to be performed.

© Aptech Ltd. Session 10 / 21

Instructions to the Trainer(s):

- Using Slide 21, explain the Envelope Aggregate.
- Using stored procedures offers numerous advantages over using Transact-SQL statements. These are as follows:
 - **Improved Security:** The database administrator can improve the security by associating database privileges with stored procedures. Users can be given permission to execute a stored procedure even if the user does not have permission to access the tables or views.
 - **Precompiled Execution:** Stored procedures are compiled during the first execution. For every subsequent execution, SQL Server reuses this precompiled version. This reduces the time and resources required for compilation.
 - **Reduced Client/Server Traffic:** Stored procedures help in reducing network traffic. When Transact-SQL statements are executed individually, there is network usage separately for execution of each statement. When a stored procedure is executed, the Transact-SQL statements are executed together as a single unit. Network path is not used separately for execution of each individual statement. This reduces network traffic.
 - **Reuse of code:** Stored procedures can be used multiple times. This eliminates the necessity to repetitively type out hundreds of Transact-SQL statements every time a similar task is to be performed.

Types of Stored Procedures 1-3

➤ SQL Server supports various types of stored procedures such as:

- User-Defined Stored Procedures
- Extended Stored Procedures

User-Defined Stored Procedures

➤ Also known as custom stored procedures
➤ Used for reusing Transact-SQL statements for performing repetitive tasks

Two types of user-defined stored procedures

```
graph TD; A[Two types of user-defined stored procedures] --> B[Transact-SQL stored procedures]; A --> C[Common Language Runtime (CLR) stored procedures.]
```

© Aptech Ltd. Session 10 / 22

Instructions to the Trainer(s):

- Using Slide 22, explain the types of stored procedures.
- SQL Server supports various types of stored procedures. These are described as follows:
- **User-Defined Stored Procedures:** User-defined stored procedures are also known as custom stored procedures. These procedures are used for reusing Transact-SQL statements for performing repetitive tasks.
- There are two types of user-defined stored procedures:
 - Transact-SQL stored procedures
 - Common Language Runtime (CLR) stored procedures.
- Transact-SQL stored procedures consists of Transact-SQL statements whereas the CLR stored procedures are based on the .NET framework CLR methods.
- Both the stored procedures can take and return user-defined parameters.

Types of Stored Procedures 2-3

Extended Stored Procedures

- Helps SQL Server in interacting with the operating system
- Not resident objects of SQL Server
- They are procedures that are implemented as Dynamic-Link Libraries (DLL) executed outside the SQL Server environment
- Use the 'xp' prefix

Instructions to the Trainer(s):

- Using Slide 23, explain the extended stored procedure in detail.
- Extended stored procedures help SQL Server in interacting with the operating system.
- Extended stored procedures are not resident objects of SQL Server.
- They are procedures that are implemented as Dynamic-link Libraries (DLL) executed outside the SQL Server environment.
- The application interacting with SQL Server calls the DLL at run-time.
- The DLL is dynamically loaded and run by SQL Server. SQL Server allots space to run the extended stored procedures.
- Extended stored procedures use the 'xp' prefix.
- Tasks that are complicated or cannot be executed using Transact-SQL statements are performed using extended stored procedures.

Types of Stored Procedures 3-3

System Stored Procedures

- Used for interacting with system tables and performing administrative tasks such as updating system tables
- These procedures are located in the Resource database
- System stored procedures allow GRANT, DENY, and REVOKE permissions

A system stored procedure

- Is a set of pre-compiled Transact-SQL statements executed as a single unit.
- Is used in database administrative and informational activities.
- Provide easy access to the metadata information about database objects such as system tables, user-defined tables, views, and indexes.

Instructions to the Trainer(s):

- Using Slide 24, explain the system stored procedures.
- System stored procedures are commonly used for interacting with system tables and performing administrative tasks such as updating system tables.
- The system stored procedures are prefixed with 'sp_'. These procedures are located in the Resource database.
- These procedures can be seen in the sys schema of every system and user-defined database.
- System stored procedures allow GRANT, DENY, and REVOKE permissions.
- A system stored procedure is a set of pre-compiled Transact-SQL statements executed as a single unit.
- System procedures are used in database administrative and informational activities.
- These procedures provide easy access to the metadata information about database objects such as system tables, user-defined tables, views, and indexes.
- System stored procedures logically appear in the sys schema of system and user-defined databases. When referencing a system stored procedure, the sys schema identifier is used. The system stored procedures are stored physically in the hidden Resource database and have the sp_ prefix. System stored procedures are owned by the database administrator.
- Mention system tables are created by default at the time of creating a new database. These tables store the metadata information about user-defined objects such as tables and views. Users cannot access or update the system tables using system stored procedures except through permissions granted by a database administrator.

Classification of System Stored Procedures

Catalog Stored Procedures	All information about tables in the user database is stored in a set of tables called the system catalog. Information from the system catalog can be accessed using catalog procedures. For example, the sp_tables catalog stored procedure displays the list of all the tables in the current database.
Security Stored Procedures	Security stored procedures are used to manage the security of the database. For example, the sp_changedbowner security stored procedure is used to change the owner of the current database.
Cursor Stored Procedures	Cursor procedures are used to implement the functionality of a cursor. For example, the sp_cursor_list cursor stored procedure lists all the cursors opened by the connection and describes their attributes.
Distributed Query Stored Procedures	Distributed stored procedures are used in the management of distributed queries. For example, the sp_indexes distributed query stored procedure returns index information for the specified remote table.
Database Mail and SQL Mail Stored Procedures	Database Mail and SQL Mail stored procedures are used to perform e-mail operations from within the SQL Server. For example, the sp_send_dbmail database mail stored procedure sends e-mail messages to specified recipients. The message may include a query resultset or file attachments or both.

© Aptech Ltd. Session 10 / 25

Instructions to the Trainer(s):

- Using Slide 25, explain the classification of System Stored procedures.
- System stored procedures can be classified into different categories depending on the tasks they perform.
- Some of the important categories are as follows:
 - **Catalog Stored Procedures:** All information about tables in the user database is stored in a set of tables called the system catalog. Information from the system catalog can be accessed using catalog procedures. For example, the sp_tables catalog stored procedure displays the list of all the tables in the current database.
 - **Security Stored Procedures:** Security stored procedures are used to manage the security of the database. For example, the sp_changedbowner security stored procedure is used to change the owner of the current database.
 - **Cursor Stored Procedures:** Cursor procedures are used to implement the functionality of a cursor. For example, the sp_cursor_list cursor stored procedure lists all the cursors opened by the connection and describes their attributes.
 - **Distributed Query Stored Procedures:** Distributed stored procedures are used in the management of distributed queries. For example, the sp_indexes distributed query stored procedure returns index information for the specified remote table.
 - **Database Mail and SQL Mail Stored Procedures:** Database Mail and SQL Mail stored procedures are used to perform e-mail operations from within the SQL Server. For example, the sp_send_dbmail database mail stored procedure sends e-mail messages to specified recipients. The message may include a query resultset or file attachments or both.

Temporary Stored Procedures

Local Temporary Procedure	Global Temporary Procedure
Visible only to the user that created it	Visible to all users
Dropped at the end of the current session	Dropped at the end of the last session
Local Temporary Procedure	Global Temporary Procedure
Can only be used by its owner	Can be used by any user
Uses the # prefix before the procedure name	Uses the ## prefix before the procedure name

Instructions to the Trainer(s):

- Using Slide 26, explain the temporary stored procedures.
- Stored procedures created for temporary use within a session are called temporary stored procedures. These procedures are stored in the tempdb database.
- The tempdb system database is a global resource available to all users connected to an instance of SQL Server. It holds all temporary tables and temporary stored procedures.
- SQL Server supports two types of temporary stored procedures namely, local and global.

Remote Stored Procedures

Stored procedures that run on remote SQL Servers are known as remote stored procedures.

Remote stored procedures can be used only when the remote server allows remote access.

When a remote stored procedure is executed from a local instance of SQL Server to a client computer, a statement abort error might be encountered.

When such an error occurs, the statement that caused the error is terminated, but the remote procedure continues to be executed.

Instructions to the Trainer(s):

- Using Slide 27, explain the remote stored procedures.
- Stored procedures that run on remote SQL Servers are known as remote stored procedures.
- Remote stored procedures can be used only when the remote server allows remote access.
- When a remote stored procedure is executed from a local instance of SQL Server to a client computer, a statement abort error might be encountered.
- When such an error occurs, the statement that caused the error is terminated, but the remote procedure continues to be executed.

Extended Stored Procedures

- Extended stored procedures are used to perform tasks that are unable to be performed using standard Transact-SQL statements.
- Extended stored procedures use the 'xp_' prefix. These stored procedures are contained in the dbo schema of the master database.

Instructions to the Trainer(s):

- Using Slide 28, explain the remote stored procedures.
- Extended stored procedures are used to perform tasks that are unable to be performed using standard Transact-SQL statements.
- Extended stored procedures use the 'xp_' prefix.
- These stored procedures are contained in the dbo schema of the master database.
- Tell the students that when an extended stored procedure is executed, either in a batch or in a module, qualify the stored procedure name with master.dbo.

Custom or User-defined Stored Procedures

- In SQL Server, users are allowed to create customized stored procedures for performance of various tasks.
- Such stored procedures are referred to as user-defined or custom stored procedures.

CustomerID	TerritoryID	Name
1	15	Australia
2	33	Australia
3	51	Australia
4	69	Australia
5	87	Australia
6	105	Australia
7	123	Australia
8	141	Australia
9	159	Australia
10	177	Australia

Output of a Simple Stored Procedure

© Aptech Ltd.

Session 10 / 29

Instructions to the Trainer(s):

- Using Slide 29, explain the custom or user-defined stored procedures.
- In SQL Server, users are allowed to create customized stored procedures for performance of various tasks.
- Such stored procedures are referred to as user-defined or custom stored procedures.
- For example, consider a table `Customer_Details` that stores the details about all the customers. You would have to type out Transact-SQL statements every time you wished to view the details about the customers. Instead, you could create a custom stored procedure that would display these details whenever the procedure is executed.
- Creating a custom stored procedure requires `CREATE PROCEDURE` permission in the database and `ALTER` permission on the schema in which the procedure is being created.

Using Parameters

Parameters are of two types:

**Input
Parameters**

Allow the calling program to pass values to a stored procedure. These values are accepted into variables defined in the stored procedure.

**Output
Parameters**

Allow a stored procedure to pass values back to the calling program. These values are accepted into variables by the calling program.

© Aptech Ltd. Session 10 / 30

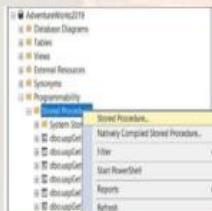
Instructions to the Trainer(s):

- Using Slide 30, explain how to use parameters.
- The real advantage of a stored procedure comes into picture only when one or more parameters are used with it.
- Data is passed between the stored procedure and the calling program when a call is made to a stored procedure.
- This data transfer is done using parameters. Parameters are of two types that are as follows:
 - **Input Parameters:** Input parameters allow the calling program to pass values to a stored procedure. These values are accepted into variables defined in the stored procedure.
 - **Output Parameters:** Output parameters allow a stored procedure to pass values back to the calling program. These values are accepted into variables by the calling program.

Using SSMS to Create Stored Procedures

➤ Steps to create a user-defined stored procedure using SSMS are:

1. Launch Object Explorer.
2. In Object Explorer, connect to an instance of Database Engine and after successfully connection, expand the instance.
3. Expand the Databases node and then, expand the AdventureWorks2019 database.



Creating a Stored Procedure

Parameter	Value
Author	Your name
Create Date	Today's date
Description	Returns year to sales data for a territory
Procedure_Name	uspGetTotals
@Param1	@territory
@Datatype_For_Param1	varchar(50)
Default_Value_For_Param1	NULL
@Param2	
@Datatype_For_Param2	
Default_Value_For_Param2	

Parameter Values

© Aptech Ltd. Session 10 / 31

Instructions to the Trainer(s):

- Using Slide 31, explain steps to create stored procedure using SSMS in details.
- Steps are as follows:
 - Launch Object Explorer.
 - In Object Explorer, connect to an instance of Database Engine and after successfully connection, expand the instance.
 - Expand the Database node and then, expand the Adventure Works2019 database.

Modifying and Dropping Stored Procedures

- Permissions associated with stored procedure are lost when a stored procedure is re-created. However, when a stored procedure is altered, permissions defined for the stored procedure remain same even though procedure definition is changed.

Dropping stored procedures

- Stored procedures can be dropped if they are no longer required. If another stored procedure calls a deleted procedure, an error message is displayed.

- If a new procedure is created using the same name as well as the same parameters as the dropped procedure, all calls to the dropped procedure will be executed successfully.

© Aptech Ltd.

Session 10 / 32

Instructions to the Trainer(s):

- Using Slide 32, explain how to modify and drop stored procedures.
- The permissions associated with the stored procedure are lost when a stored procedure is re-created.
- However, when a stored procedure is altered, the permissions defined for the stored procedure remains the same even though the procedure definition is changed.
- A procedure can be altered using the ALTER PROCEDURE statement.
- Dropping stored procedures are those that can be dropped if they are no longer required.

Nested Stored Procedures

- SQL Server 2019 enables stored procedures to be called inside other stored procedures.
- The called procedures can in turn call other procedures.
- This architecture of calling one procedure from another procedure is referred to as nested stored procedure architecture.
- When a stored procedure calls another stored procedure, the level of nesting is said to be increased by one.
- Similarly, when a called procedure completes its execution and passes control back to the calling procedure, the level of nesting is said to be decreased by one.
- The maximum level of nesting supported by SQL Server 2019 is 32.

Instructions to the Trainer(s):

- Using Slide 33, explain the nested stored procedures.
- SQL Server 2019 enables stored procedures to be called inside other stored procedures.
- The called procedures can in turn call other procedures.
- This architecture of calling one procedure from another procedure is referred to as nested stored procedure architecture.
- When a stored procedure calls another stored procedure, the level of nesting is said to be increased by one.
- Similarly, when a called procedure completes its execution and passes control back to the calling procedure, the level of nesting is said to be decreased by one.
- The maximum level of nesting supported by SQL Server 2019 is 32.

Querying System MetaData 1-3

The properties of an object such as a table or a view are stored in special system tables.

These properties are referred to as metadata. All SQL objects produce metadata.

This metadata can be viewed using system views, which are predefined views of SQL Server.

System Catalog Views

- These contain information about the catalog in a SQL Server system.
- A catalog is similar to an inventory of objects.
- These views contain a wide range of metadata.

© Aptech Ltd.

Session 10 / 34

Instructions to the Trainer(s):

- Using Slide 34, explain how to query system metadata.
- The properties of an object such as a table or a view are stored in special system tables. These properties are referred to as metadata.
- All SQL objects produce metadata. This metadata can be viewed using system views, which are predefined views of SQL Server.
- There are over 230 different system views and these are automatically inserted into the user created database. These views are grouped into several different schemas.
- **System Catalog Views:** These contain information about the catalog in a SQL Server system. A catalog is similar to an inventory of objects. These views contain a wide range of metadata.

Querying System MetaData 2-3

Information Schema Views

- Users can query information schema views to return system metadata.
- These views are useful to third-party tools that may not be specific for SQL Server.
- Information schema views provide an internal, system table independent view of the SQL Server metadata.

Information Schema Views	SQL Server System Views
They are stored in their own schema, INFORMATION_SCHEMA.	They appear in the sys schema.
They use standard terminology instead of SQL Server terms. For example, they use catalog instead of database and domain instead of user-defined data type.	They adhere to SQL Server terminology.
They may not expose all the metadata available to SQL Server's own catalog views. For example, sys.columns includes attributes for identity property and computed column property, while INFORMATION_SCHEMA.columns does not.	They can expose all the metadata available to SQL Server's catalog views.

Information Schema Views and SQL Server System Views

© Aptech Ltd.

Session 10 / 35

Instructions to the Trainer(s):

- Using Slide 35, explain the Information Schema Views.
- Users can query information schema views to return system metadata.
- These views are useful to third-party tools that may not be specific for SQL Server.
- Information schema views provide an internal, system table-independent view of the SQL Server metadata.
- Information schema views enable applications to work correctly although significant changes.
- The points given in table will help to decide whether one should query SQL Server-specific system views or information schema views.

Querying System MetaData 3-3

System Metadata Functions

- Built-in functions can return metadata to a query.
- These include scalar functions and table-valued functions.
- SQL Server metadata functions come in a variety of formats.

Function Name	Description	Example
OBJECT_ID(<object_name>)	Returns the object ID of a database object.	OBJECT_ID('Sales.Customer')
OBJECT_NAME(<object_id>)	Returns the name corresponding to an object ID.	OBJECT_NAME(197575742)
@@ERROR	Returns 0 if the last statement succeeded; otherwise returns the error number.	@@ERROR
SERVERPROPERTY(<property >)	Returns the value of the specified server property.	SERVERPROPERTY('Collation')

Common System Metadata Functions

Instructions to the Trainer(s):

- Using Slide 36, explain the System Metadata Functions.
- In addition to views, SQL Server provides a number of built-in functions that return metadata to a query.
- These include scalar functions and table-valued functions, which can return information about system settings, session options, and a wide range of objects.
- SQL Server metadata functions come in a variety of formats.
- The table on Slide 36 displays different functions and their description about the common system metadata functions.

In-Class Question:

Question: What is the use of @@error?

Answer: @@Error is used to detect the error in the execution of statements. It returns 0 if the last statement was executed successfully or else return the error number.

Querying Dynamic Management Objects

- Dynamic Management Views (DMVs) and Dynamic Management Functions (DMFs) are dynamic management objects that return server and database state information.
- DMVs and DMFs are collectively referred to as dynamic management objects.
- Used for examining the state of SQL Server instance, troubleshooting, and performance tuning.
- In order to query DMVs, it is required to have VIEW SERVER STATE or VIEW DATABASE STATE permission, depending on the scope of the DMV.

Instructions to the Trainer(s):

- Using Slide 37, explain how to query dynamic management objects.
- Dynamic Management Views (DMVs) and Dynamic Management Functions (DMFs) are dynamic management objects that return server and database state information.
- DMVs and DMFs are collectively referred to as dynamic management objects.
- They provide useful insight into the working of software and can be used for examining the state of SQL Server instance, troubleshooting, and performance tuning.
- In order to query DMVs, it is required to have VIEW SERVER STATE or VIEW DATABASE STATE permission, depending on the scope of the DMV.

Categorizing and Querying DMVs

Naming Pattern	Description
db	Database related
io	I/O statistics
Os	SQL Server Operating System Information
'tran'	Transaction-related
'exec'	Query execution-related metadata

Organizing DMVs by Function

Querying the sys.dm_exec_sessions DMV

© Aptech Ltd. Session 10/38

Instructions to the Trainer(s):

- Using Slide 38, explain how to categorize and query DMVs.
- Slide 38 displays different naming patterns. These are described as follows:
 - db: Database related
 - io: I/O statistics
 - Os: SQL Server Operating System Information
 - 'tran': Transaction-related
 - 'exec': Query execution-related metadata
- sys.dm_exec_sessions is a server-scoped DMV that displays information about all active user connections and internal tasks. This information includes login user, current session setting, client version, client program name, client login time, and more. The sys.dm_exec_sessions can be used to identify a specific session and find information about it.
- Here, is_user_process is a column in the view that determines if the session is a system session or not. A value of 1 indicates that it is not a system session, but rather a user session. The program_name column determines the name of client program that initiated the session. The login_time column establishes the time when the session began.

Summary

- A view is a virtual table that is made up of selected columns from one or more tables and is created using the CREATE VIEW command in SQL Server.
- Users can manipulate the data in views, such as inserting into views, modifying the data in views, and deleting from views.
- A stored procedure is a group of Transact-SQL statements that act as a single block of code that performs a specific task.
- SQL Server supports various types of stored procedures, such as User-Defined Stored Procedures, Extended Stored Procedures, and System Stored Procedures.
- System stored procedures can be classified into different categories such as Catalog Stored Procedures, Security Stored Procedures, and Cursor Stored Procedures.
- Input and output parameters can be used with stored procedures to pass and receive data from stored procedures.
- The properties of an object such as a table or a view are stored in special system tables and are referred to as metadata.
- DMVs and DMFs are dynamic management objects that return server and database state information. DMVs and DMFs are collectively referred to as dynamic management objects.

Instructions to the Trainer(s):

- Show students Slide 39.
- Summarize the session by reading out each point on the slide.

Session 11: Indexes

11.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. Prepare a question or two that will be a key point to relate the current session objectives.

11.1.1 Teaching Skills

To teach this session, you should be well versed with indexes and their performance.

You should teach the concepts in the theory class using the images provided. For teaching in the class, you are expected to use slides and LCD projectors.

In-Class Activities

Follow the order given here during In-Class activities.

Slide 2

Objectives

- Define and explain Indexes
- Explain Storage Structure
- Explain types of Indexes
- Understand Index Management

© Aptech Ltd. Session 11/ 2

Instructions to the Trainer(s):

Give students a brief overview of the current session through the session objectives listed in Slide 2.

11.2 In-Class Explanations

Slide 3

Introduction I-2

Indexes are special data structures associated with tables or views that help speed up the query.

Index Type	Description
Clustered	It sorts and stores the data rows of a table or view in order based on the clustered index key. Clustered index is implemented as a B-tree index structure that supports fast retrieval of the rows, based on their clustered index key values.
Nonclustered	Non-clustered index is defined on a table or view that has data in either a clustered structure or on a heap. Each index row in the non-clustered index contains nonclustered key value and a row locator. Locator points to data row in the clustered index or heap having key value. Rows in index are stored in order of the index key values, but the data rows are not guaranteed to be in any particular order unless a clustered index is created on the table.
Unique	Unique index ensures that index key contains no duplicate values and therefore, each row in the table or view is in some way unique.
Columnstore	Columnstore index stores and manages data by using column-based data storage and column-based query processing in in-memory.
Filtered	Columnstore indexes work well for data warehousing workloads that primarily perform bulk loads and read-only queries. Use the columnstore index to achieve up to 10x query performance gains over traditional row-oriented storage, and up to 7x data compression over the uncompressed data size.
Spatial	Optimized non-clustered index is suited to cover queries that select from a well-defined subset of data. It uses a filter predicate to index a portion of rows in a table. A well-designed filtered index can improve query performance, reduce index maintenance costs, and reduce index storage costs compared with full-table indexes.
XML	It provides the ability to perform certain operations more efficiently on spatial objects in a column of geometry data type.
XML	Due to large size of XML columns, queries that search within these columns can be slow. You can speed up these queries by creating an XML index on each column. An XML index can be a clustered or a nonclustered index.

© Aptech Ltd. Session 11 / 3

Instructions to the Trainer(s):

- Introduce the session using Slide 3, explain the concept of Indexes.
- Indexes are special data structures associated with tables or views that help speed up the query.
- SQL Server makes use of indexes to find data when a query is processed. The SQL Server engine uses an index in the similar way as a student uses a book index. For example, consider that you must find all references to INSERT statements in a SQL book. The immediate approach taken will be to scan each page of the book beginning from the starting page. You mark each time the word INSERT is found, until the end of the book is reached. This approach is time consuming and laborious. The second way is to use the index in the back of the book to find the page numbers for each occurrence of the INSERT statements. The second way produces the same results as the first, but by tremendously saving time.
- When SQL Server has not defined any index for searching, then the process is similar to the first way in the example; the SQL engine has to visit every row in a table. In database terminology, this behavior is called table scan or just scan.
- A table scan is not always troublesome, but it is sometimes unavoidable. However, as a table grows up to thousands and millions of rows and beyond, scans become slower and more expensive. In such cases, indexes are strongly recommended.
- Creating or removing indexes from a database schema will not affect an application's code. Indexes operate in the backend with support of the database engine. Moreover, creating an appropriate index can significantly increase the performance of an application.

- Different types of Indexes are:
- **Clustered:** Clustered indexes sort and store rows data in a table or view based on their fundamental values.
 - **Nonclustered:** Non-clustered index is defined on a table or view that has data in either a clustered or on a heap.
 - **Unique:** The unique index in SQL confirms that the index key does not contain any duplicate values and therefore, enables the users to analyze that every row in the table is unique in one or the other way.
 - **Columnstore:** A columnstore index is used for storing and querying large data warehousing fact tables. The Columnstore index works well for data warehousing work.
 - **Filtered:** A filtered index is used for improving query performance, reduce maintenance cost, and storage cost.
 - **Spatial:** Performs certain operations more efficiently.
 - **XML:** An XML index is used for speeding-up queries.

Introduction 2-2

- ❑ There are other types of indexes such as Hash, Memory optimize nonclustered, Index with included column, Index on computed columns, and Full text.
- ❑ A table scan is not always troublesome, but it is sometimes unavoidable.

Instructions to the Trainer(s):

- Using Slide 4, explain indexes.
- There are other type of indexes such as Hash, Memory optimize nonclustered, Index with included column, Index on computed columns, and Full text.
- When SQL Server has not defined any index for searching, then the process is similar to the first way in the example; the SQL engine must visit every row in a table. In database terminology, this behavior is called table scan or just scan.
- A table scan is not always troublesome, but it is sometimes unavoidable. However, as a table grows up to thousands and millions of rows and beyond, scans become slower and more expensive. In such cases, indexes are strongly recommended.

Overview of Data Storage

➤ SQL Server stores data in storage units known as data pages. These pages contain data in the form of rows.

➤ A page begins with a 96-byte header, which stores system information about the page.

This information includes the following:

- Page number
- Page type
- Amount of free space on the page
- Allocation unit ID of the object to which the page is allocated

© Aptech Ltd.

Session 11/3

Instructions to the Trainer(s):

- Using Slide 5, explain the overview of data storage.
- SQL Server stores data in storage units known as data pages. These pages contain data in the form of rows.
- A page begins with a 96-byte header, which stores system information about the page. This information includes the following:
 - Page number
 - Page type
 - Amount of free space on the page
 - Allocation unit ID of the object to which the page is allocated

Data Files 1-2

- All input and output operations in the database are performed at the page level.
- SQL Server stores data pages in files known as data files. The space allotted to a data file is divided into sequentially numbered data pages.

The diagram illustrates the structure of data files. It shows two separate boxes labeled "Data Files". The top box is titled "Primary Data File: File ID 01" and contains four page icons labeled "page 01:0000", "page 01:0001", "page 01:0002", and "page 01:0511". Below these are three ellipsis dots (...). The bottom box is titled "Secondary Data File: File ID 02" and contains four page icons labeled "page 02:0000", "page 02:0001", "page 02:0002", and "page 02:0127". Below these are three ellipsis dots (...). The entire diagram is set against a background of a city skyline at night.

© Aptech Ltd. Session 11/ 6

Data Files 2-2

There are three different types of data files:

- Primary**
 - A primary data file is automatically created at the time of creation of the database. This file has references to all other files in the database. The recommended file extension for primary data files is **.mdf**.
- Secondary**
 - These are optional user-defined data files. Data can be spread across multiple disks by putting each file on a different disk drive. Recommended file name extension for secondary data files is **.ndf**.
- Transaction Log**
 - Log files contain information about modifications carried out in the database. This information is useful in recovery of data in contingencies such as sudden power failure or the need to shift the database to a different server. There is at least one log file for each database. The recommended file extension for log files is **.ldf**.

© Aptech Ltd. Session 11/ 7

Instructions to the Trainer(s):

- Using Slide 6, explain data files.
- All input and output operations in the database are performed at the page level. This means that the database engine reads or writes data pages.

- A set of eight contiguous data pages is referred to as an extent.
- SQL Server stores data pages in files known as data files. The space allotted to a data file is divided into sequentially numbered data pages.
- Using Slide 7, explain different types of data files.
- There are three types of data files in SQL Server. These are as follows:
 - **Primary Data Files:** A primary data file is automatically created at the time of creation of the database. This file has references to all other files in the database. The recommended file extension for primary data files is .mdf.
 - **Secondary Data Files:** Secondary data files are optional in a database and can be created to segregate database objects such as tables, views, and procedures. The recommended file extension for secondary data files is .ndf.
 - **Transaction Log:** Log files contain information about modifications carried out in the database. This information is useful in recovery of data in contingencies such as sudden power failure or the necessity to shift the database to a different server. There is at least one log file for each database. The recommended file extension for log files is .ldf.

Slide 8

Requirement for Indexes

- To facilitate quick retrieval of data from a database, SQL Server provides the indexing feature.
- An index in SQL Server database contains information that allows to find specific data without scanning through the entire table.

Index			
A		C	
Adapter	1	Border	19
Aggregate	10	Bullet	58
Analysis	13		
Average	23	Consistency	20
		Connect	22
B			
Board	17	Communication	24
Brilliant	18	Character	30

Index in a Book

© Aptech Ltd.

Session n/8

Instructions to the Trainer(s):

- Using Slide 8, explain the requirements for Indexes.
- To facilitate quick retrieval of data from a database, SQL Server provides the indexing feature.
- Similar to an index in a book, an index in SQL Server database contains information that allows you to find specific data without scanning through the entire table.

Indexes

Records in a table are stored in the order in which they are inserted, this storage is unsorted.

When data is to be retrieved from such tables, the entire table requires to be scanned thus, slowing down the retrieval process.

To speed up data retrieval process, indexes are required.

Index	EmployeeID	EmployeeName	DepartmentID
CN00012	CN00016	John Keena	Purchase
CN00015	CN00015	Smith Jones	Accounts
CN00016	CN00020	Albert Walker	Sales
CN00020	CN00012	Rosa Stines	Administrator

Indexes

© Aptech Ltd. Session 11/9

Instructions to the Trainer(s):

- Using Slide 9, explain the Indexes.
- In a table, records are stored in the order in which they are entered. Their storage in the database is unsorted.
- When data is to be retrieved from such tables, the entire table must be scanned. This slows down the query retrieval process.
- To speed up query retrieval, indexes must be created.
- When an index is created on a table, the index creates an order for the data rows or records in the table. This assists in faster location and retrieval of data during searches.
- Indexes are automatically created when PRIMARY KEY and UNIQUE constraints are defined on a table. Indexes reduce disk I/O operations and consume fewer system resources.
- Consider following facts and guidelines about indexes:
 - Indexes increase the speed of queries that join tables or perform sorting operations.
 - Indexes implement the uniqueness of rows if defined when you create an index.
 - Indexes are created and maintained in ascending or descending order.

In-Class Question:

Question: What are the qualities of indexing?

Answer: Simplicity, Economical, Flexibility, Quick location, Suitability, Safety, and so on.

Scenario

- In a telephone directory, where large amount of data is stored and is frequently accessed, the storage is done in alphabetical order. If such data were unsorted, it would be nearly impossible to search for a specific telephone number.

- Similarly, in a database table having a large number of records that are frequently accessed, the data is to be sorted for fast retrieval.
- When an index is created on the table, the index either physically or logically sorts the records.

Instructions to the Trainer(s):

- Using Slide 10, explain the scenario.
- In a telephone directory, where a large amount of data is stored and is frequently accessed, the storage of data is done in an alphabetical order. If such data were unsorted, it would be nearly impossible to search for a specific telephone number.
- Similarly, in a database table having a large number of records that are frequently accessed, the data is to be sorted for fast retrieval.
- When an index is created on the table, the index either physically or logically sorts the records.
- Thus, searching for a specific record becomes faster and there is less strain on system resources.

Accessing Data Group-wise

➤ Indexes are useful when data is accessed group-wise.

For example, you make modifications to the conveyance allowance for all employees based on the department they work in.

Department Name	Employee Name
Marketing	Jenny Woods
Marketing	Merry Thomas
Marketing	John Updeke
Marketing	Robert Williamson
Sales	Smith Gordon
Sales	Albert Wang

Accessing Data Group-wise

© Aptech Ltd. Session n/ n

Instructions to the Trainer(s):

- Using Slide 11, explain how data can be accessed group-wise.
- Indexes are useful when data must be accessed group-wise.
- For example, you want to make modifications to the conveyance allowance for all employees based on the department they work in. Here, you wish to make the changes for all employees in one department before moving on to employees in another department. In this case, an index can be created as shown in the figure on the Department column before accessing the records.
- This index will create logical chunks of data rows based on the department. This again will limit the amount of data actually scanned during query retrieval.
- Hence, retrieval will be faster and there will be less strain on system resources.

Index Architecture

In SQL Server, data can be stored either in a sorted or random manner.

If it is stored in a sorted manner then, data is present in clustered structure.

If stored at random then, its present in a heap structure.

Employee_Details		
EmpID	EmpName	DeptID
CN00020	Rosa Stevens	BN0001
CN00018	John Updeek	BN0020
CN00019	Smith Gordon	BN0021
CN00012	Robert Tyson	BN0011

Heap Structure

Employee_Details		
EmpID	EmpName	DeptID
CN00012	Robert Tyson	BN0011
CN00018	John Updeek	BN0020
CN00019	Smith Gordon	BN0021
CN00020	Rosa Stevens	BN0001

Clustered Structure

Index Architecture

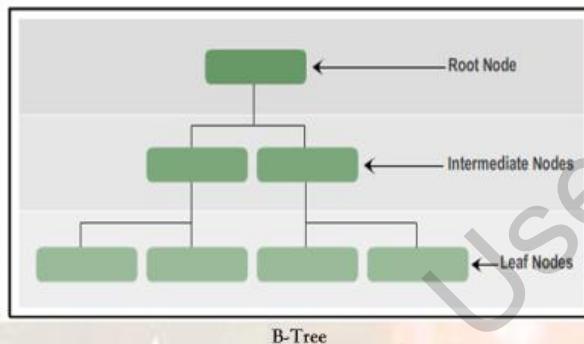
Instructions to the Trainer(s):

- Using Slide 12, explain the Index Architecture.
- In SQL Server, data in the database can be stored either in a sorted manner or at random.
- If data is stored in a sorted manner, the data is said to be present in a clustered structure.
- If it is stored at random, it is said to be present in a heap structure.

B-Tree

Each page in an index B-tree is called an index node.

Top node of the B-tree is called the root node, whereas the Bottom nodes are called the leaf nodes.



Instructions to the Trainer(s):

- Using Slide 13, discuss the B-Tree.
- In SQL Server, all indexes are structured in the form of B-Trees.
- A B-Tree structure can be visualized as an inverted tree with the root right at the top, splitting into branches and then, into leaves right at the bottom as shown in the figure.
- In a B-Tree structure, there is a single root node at the top. This node then branches out into the next level, known as the first intermediate level.
- The nodes at the first intermediate level can branch out further. This branching can continue into multiple intermediate levels and then, finally the leaf level.
- The nodes at the leaf level are known as the leaf nodes.

Index B-Tree Structure I-2

The diagram features four colored callouts on a background image of a city skyline at sunset:

- Teal Callout:** In the B-Tree structure of an index, the root node consists of an index page.
- Blue Callout:** There can be multiple intermediate levels in an index B-Tree.
- Green Callout:** This index page contains pointers present in the first intermediate level.
- Orange Callout:** The leaf nodes either data pages containing data rows or index pages containing index rows that point to data rows.

© Aptech Ltd. Session II/14

Instructions to the Trainer(s):

- Using Slide 14, explain the Index B-Tree Structure.
- In the B-Tree structure of an index, the root node consists of an index page.
- This index page contains pointers that point to the index pages present in the first intermediate level.
- These index pages in turn point to the index pages present in the next intermediate level. There can be multiple intermediate levels in an index B-Tree.
- The leaf nodes of the index B-Tree have either data pages containing data rows or index pages containing index rows that point to data rows.

In-Class Question:

Question: Why do we use B-tree?

Answer: A B-tree is a tree data structure that keeps data sorted and allows searches, insertions, and deletions in logarithmic amortized time.

Index B-Tree Structure 2-2

The diagram illustrates the Index B-Tree Structure with three distinct levels of nodes:

- Root Node:** The top level, represented by a green box labeled "Index Page".
- Intermediate Nodes:** The middle level, represented by two green boxes labeled "Index Page". These nodes have pointers pointing to the Leaf Nodes.
- Leaf Nodes:** The bottom level, represented by four green boxes labeled "Index/Data Page". These nodes represent the final data storage points.

Index B-Tree Structure

Different types of nodes are as follows:

- Root Node - Contains an index page with pointers pointing to index pages at the first intermediate level.
- Intermediate Nodes - Contain index pages with pointers pointing either to index pages at the next intermediate level or to index or data pages at the leaf level.
- Leaf Nodes - Contain either data pages or index pages that point to data pages.

© Aptech Ltd.

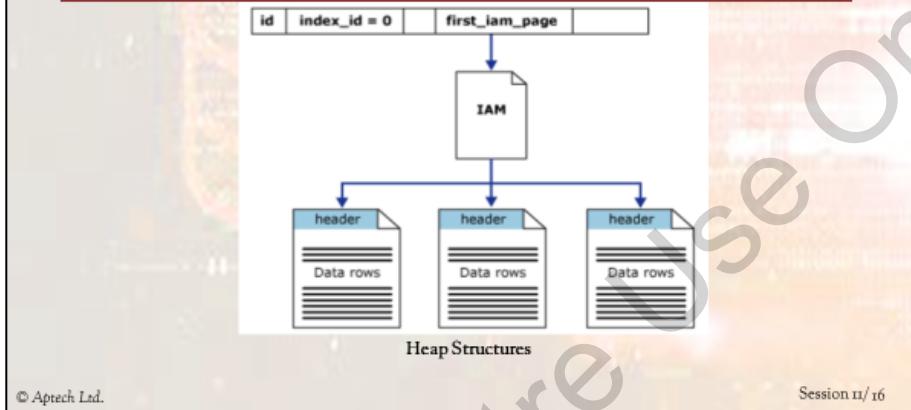
Session 11/15

Instructions to the Trainer(s):

- Using Slide 15, explain different types of nodes in Index B-Tree Structure.
- In a B-Tree structure, the top-most node is the Root Node and the bottom-most is the Leaf Node. The node in-between the Root and Leaf node is the Intermediate node.
- Different types of nodes are as follows:
 - **Root Node** - Contains an index page with pointers pointing to index pages at the first intermediate level.
 - **Intermediate Nodes** - Contain index pages with pointers pointing either to index pages at the next intermediate level or to index or data pages at the leaf level.
 - **Leaf Nodes** - Contain either data pages or index pages that point to data pages.

Heap Structures

- A heap is a table without a clustered index.
- In a heap structure, the data pages and records are not arranged in sorted order.
- The only connection is the information recorded in the Index Allocation Map (IAM) pages.



Instructions to the Trainer(s):

- Using Slide 16, explain the heap structures.
- In a heap structure, the data pages and records are not arranged in sorted order. The only connection between the data pages is the information recorded in the Index Allocation Map (IAM) pages.
- In SQL Server, IAM pages are used to scan through a heap structure.
- IAM pages map extents that are used by an allocation unit in a part of a database file.
- A heap can be read by scanning the IAM pages to look for the extents that contain the pages for that heap as shown in the figure.
- If an allocation unit contains extents from more than one file, there will be multiple IAM pages linked together in an IAM chain to map these extents.

Clustered Index Structures 1-2

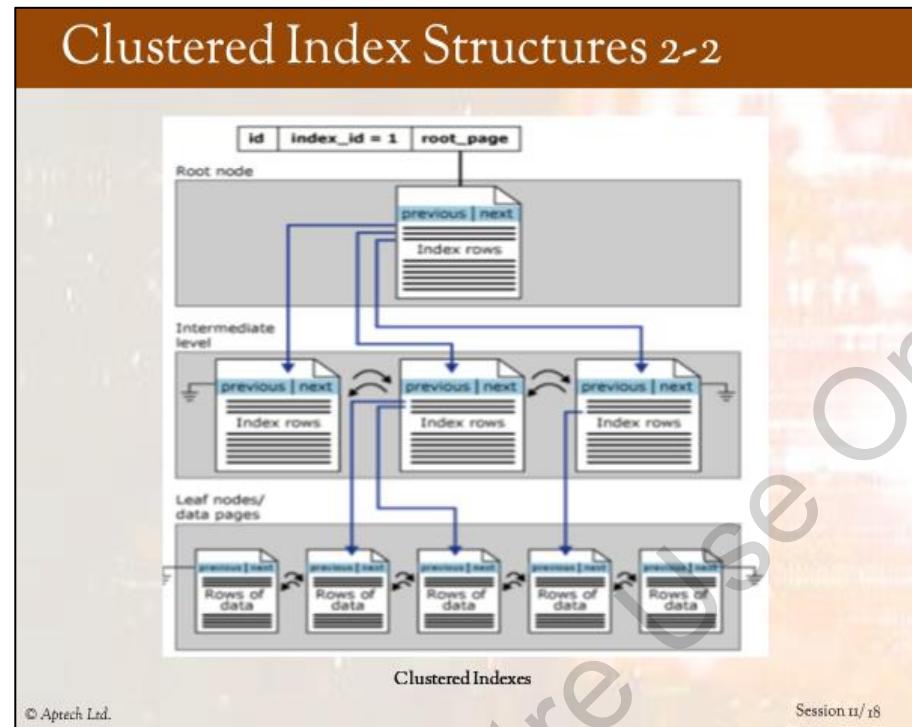
Clustered indexes are organized as B-Trees. Each page in an index B-tree is called an index node.

Clustered Index

- Leaf nodes contain data pages of the underlying table, root, and intermediate level nodes contain index pages holding index rows.
- Each index row contains a key value and a pointer to either an intermediate level page in the B-tree or a data row in the leaf level of the index.
- By default, a clustered index has a single partition. When a clustered index has multiple partitions, each partition has a B-tree structure that contains the data for that specific partition.
- The clustered index will also have one `LOB_DATA` allocation unit per partition if it contains large object (LOB) columns. It will also have one `ROW_OVERFLOW_DATA` allocation unit per partition if it contains variable length columns that exceed the 8,060 byte row size limit.

Instructions to the Trainer(s):

- Using Slide 17, explain the clustered index structures.
- A clustered index causes records to be physically stored in a sorted or sequential order.
- A clustered index determines the actual order in which data is stored in the database. Hence, you can create only one clustered index in a table.
- Uniqueness of a value in a clustered index is maintained explicitly using the `UNIQUE` keyword or implicitly using an internal unique identifier as shown in the figure.



Instructions to the Trainer(s):

- Using Slide 18, discuss the Clustered Indexes.
- From the figure displayed on Slide 18, students can view that the clustered index consists of Root node, Intermediate level and Leaf node/data pages.

Nonclustered Index Structures

- A nonclustered index is defined on a table that has data in either a clustered structure or a heap.
- Each index row in the nonclustered index contains a nonclustered key value and a row locator.

Nonclustered indexes have a similar B-Tree structure as clustered indexes, but with the following differences:

- In a nonclustered index structure, the leaf level contains index rows.

The diagram illustrates the structure of a nonclustered index. It shows a B-Tree structure with a Root node, which contains fields for ID, index_id > 0, and root_page. The Root node points to Leaf nodes. Leaf nodes also contain index rows and previous/next pointers. Leaf nodes point to Data pages, which contain rows of data and previous/next pointers. A red bracket on the right indicates that the leaf level contains index rows, while the data pages are labeled as 'Heap or clustered index'.

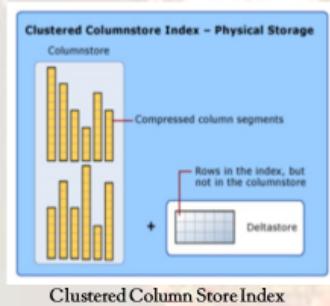
© Aptech Ltd. Session 11 / 19

Instructions to the Trainer(s):

- Using Slide 19, explain the Nonclustered index structures.
- A nonclustered index is defined on a table that has data in either a clustered structure or a heap.
- Nonclustered index will be the default type if an index is not defined on a table.
- Each index row in the nonclustered index contains a nonclustered key value and a row locator. This row locator points to the data row corresponding to the key value in the table.
- Nonclustered indexes have a similar B-Tree structure as clustered indexes but with following differences:
 - The data rows of the table are not physically stored in the order defined by their nonclustered keys.
 - In a nonclustered index structure, the leaf level contains index rows.
- Nonclustered indexes are useful when you require multiple ways to search data. Some facts and guidelines to be considered before creating a nonclustered index are as follows:
 - When a clustered index is re-created or the DROP_EXISTING option is used, SQL Server rebuilds the existing nonclustered indexes.
 - A table can have up to 999 nonclustered indexes.
 - Create clustered index before creating a nonclustered index.

Column Store Index

- A columnstore index is a feature in SQL Server for storing, retrieving, and managing data by using a columnar data format



Columnstore

It is logically organized data as a table with rows and columns and physically stored in a column-wise data format.

Rowstore

It is logically organized data as a table with rows and columns and then, physically stored in a row-wise data format.

Deltastore

It is a holding place for rows that are too few in number to be compressed into the columnstore. The deltastore stores the rows in rowstore format.

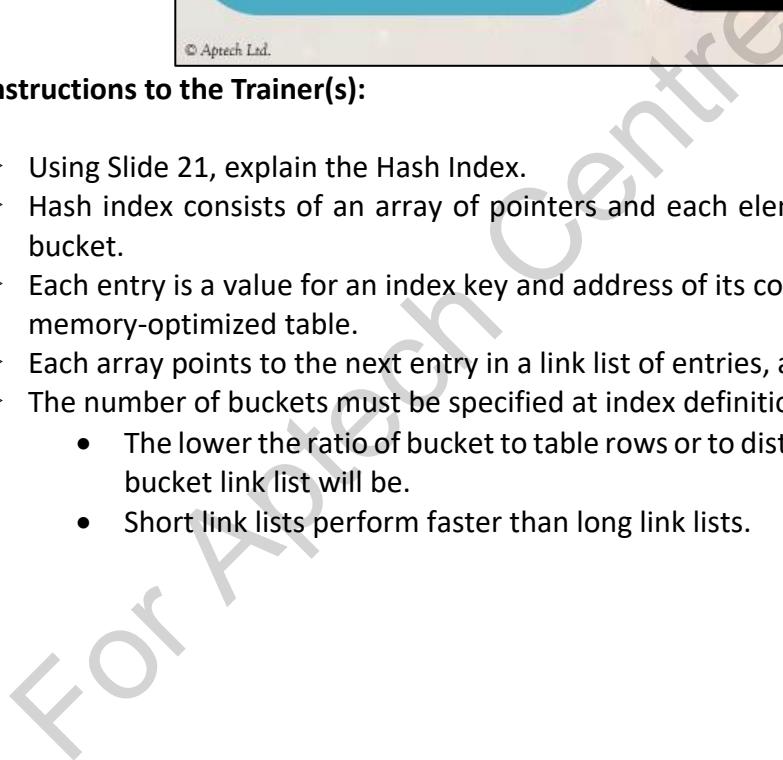
Columnstore indexes are mainly used for following reasons:

- To reduce storage costs
- Better performance

Instructions to the Trainer(s):

- Using Slide 20, explain the column store index.
- Column Store Index is a new feature in SQL Server. It enhances performance of data warehouse queries extensively.
- The regular indexes or heaps of older SQL Servers stored data in B-Tree structure row-wise, but the column store index in SQL Server stores data column-wise.
- Since the data transfer rate is slow in database servers, so column store index uses compression aggressively to reduce the disk I/O required to serve the query request.
- The B-Tree and heap stores data row-wise, which means data from all the columns of a row are stored together contiguously on the same page.
- When column store index is created, the data is stored column-wise, which means data of each individual column from each row is stored together on same page.

Hash Index 1-2



© Aptech Ltd.

The number of buckets must be specified at index definition time:

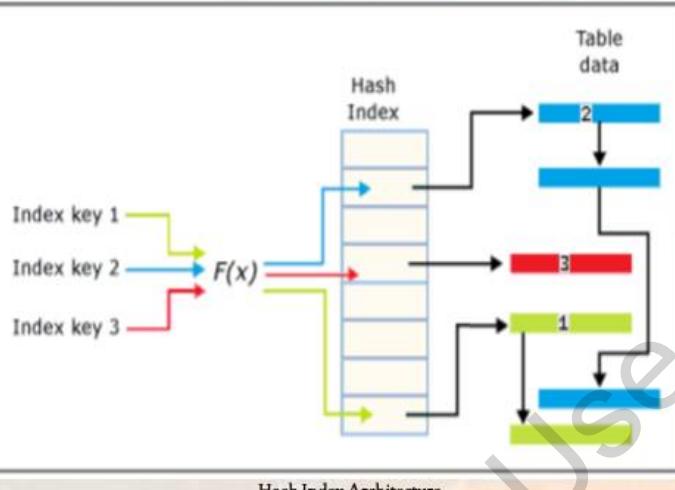
- Each entry is a value for an index key, and address of its corresponding row in the underlying memory-optimized table.
- Each entry points to the next entry in a link list of entries, all chained to the current bucket.

Session 11 / 21

Instructions to the Trainer(s):

- Using Slide 21, explain the Hash Index.
- Hash index consists of an array of pointers and each element of the array is called a hash bucket.
- Each entry is a value for an index key and address of its corresponding row in the underlying memory-optimized table.
- Each array points to the next entry in a link list of entries, all chained to the current bucket.
- The number of buckets must be specified at index definition time:
 - The lower the ratio of bucket to table rows or to distinct values, the longer the average bucket link list will be.
 - Short link lists perform faster than long link lists.

Hash Index 2-2



Instructions to the Trainer(s):

- Using Slide 22, explain the hash Index Architecture.
- Slide 22 displays the architectural flow for Hash Index.
- As observed from the figure, different Index keys 1,2,3... and so on are stored in the Hash Index, from where the data is then forwarded to the Table data.

XML Indexes

XML indexes can be created on xml data type columns. They index all tags, values, and paths over the XML instances in the column and benefit query performance.

Queries on XML columns are common in your workload. XML index maintenance cost during data modification must be considered.

```
graph TD; A([XML indexes]) --> B((Primary XML index)); A --> C((Secondary XML index))
```

© Aptech Ltd. Session 11 / 23

Instructions to the Trainer(s):

- Using Slide 23, explain the XML indexes.
- The xml data type is used to store XML documents and fragments as shown in the figure. An XML fragment is an XML instance that has a single top-level element missing.
- Due to the large size of XML columns, queries that search within these columns can be slow. You can speed up these queries by creating an XML index on each column.
- An XML index can be a clustered or a nonclustered index. Each table can have up to 249 XML indexes.
- XML Indexes are of two types:
 - **Primary XML Index:** The process of carrying out queries within an XML column can sometimes be slow. A primary XML index is created on each XML column to speed up these queries. It is a special index that shreds the XML data to store information.
 - **Secondary XML Index:** Secondary XML indexes are specialized XML indexes that help with specific XML queries. The features of secondary XML indexes are as follows:
 - Searching for values anywhere in the XML document.
 - Retrieving particular object properties from within an XML document.

Spatial Indexes

- In SQL Server, spatial indexes are built using B-trees, which means that the indexes must represent the 2dimensional spatial data in the linear order of B-trees.
- The index-creation process decomposes the space into a four-level grid hierarchy.
 - These levels are referred to as level 1 (the top level), level 2, level 3, and level 4.

The diagram shows a 4x4 grid of squares, representing a spatial index structure. It is divided into four levels of increasing resolution. Level 1 is the largest 4x4 square at the bottom left. Level 2 is a 2x2 grid of smaller squares above it. Level 3 is a 4x4 grid of even smaller squares above that. Level 4 is a single small square at the top right. A legend titled 'Key' indicates the color coding for each level: Level 1 is yellow, Level 2 is light blue, Level 3 is orange-red, and Level 4 is green. Below the grid, the text 'Four Levels of 4x4 Grids' is written.

© Aptech Ltd. Session 11 / 24

Instructions to the Trainer(s):

- Using Slide 24, discuss on Spatial Indexes.
- In SQL Server, spatial indexes are built using B-trees, which means that the indexes must represent 2dimensional spatial data in the linear order of B-trees.
- The index-creation process decomposes the space into a four-level grid hierarchy.
- These levels are referred to as level 1 (the top level), level 2, level 3, and level 4.

Full-Text Indexes

- Creating and maintaining a full-text index involves populating the index by using a process called a population also known as a crawl.

Types of population

A full-text index supports the following types of population:

- Full population
- Automatic or manual population based on change tracking
- Incremental population based on a timestamp

Instructions to the Trainer(s):

- Using Slide 25, explain the full-text indexes.
- Creating and maintaining a full-text index involves populating the index by using a process called Crawl.
- A full-text index supports following types of population:
 - Full population
 - Automatic or manual population based on change tracking
 - Incremental population based on a timestamp

Index Management

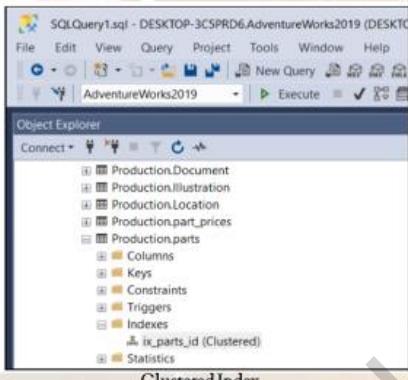
Index management allow user to manage indexes in terms various operation such as CREATE, ALTER, DROP, and so on.

Instructions to the Trainer(s):

- Using Slide 26, explain Index Management.
- Index management allow user to manage indexes in terms various operation such as CREATE, ALTER, DROP, and so on.

Create Clustered Index 1-5

➤ CREATE CLUSTERED index statement allow users to create CLUSTERED index on specified columns and table.



© Aptech Ltd. Session 11/ 27

Instructions to the Trainer(s):

- Using Slide 27, explain how to create index.
- CREATE CLUSTERED index statement allows users to create CLUSTERED index on specific tables and columns.

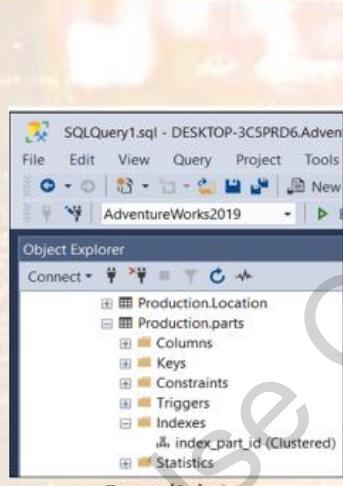
Create Clustered Index 2-5

RENAME INDEX

➤ The sp_rename is a system stored procedure that allows you to rename any user-created object in the current database including table, index, and column.

Syntax

```
EXEC sp_rename  
    index_name,  
    new_index.name,  
    N'INDEX';
```



© Aptech Ltd. Session 11 / 26

Instructions to the Trainer(s):

- Using Slide 28, explain creating clustered index.
- The sp_rename is a system stored procedure that allows users to rename any user-created object in the current database including table, index, and column.

Create Clustered Index 3-5

DISABLE INDEX

To disable an index, ALTER INDEX statement is used as follows:

Syntax

```
ALTER INDEX index_name ON  
table_name  
DISABLE;
```



```
SQLQuery2.sql - DE...D6[Lenovo pc (51)]* # X  
Use AdventureWorks2019;  
ALTER INDEX index_part_id  
ON production.parts  
DISABLE;  
100 % ▾  
Messages  
Commands completed successfully.  
Completion time: 2020-11-13T12:59:05.9584627+05:30
```

© Aptech Ltd.

Session 11 / 29

Instructions to the Trainer(s):

- Using Slide 29, explain the disable index concept.
- Disabling an index prevents user to get the access to the index.
- To disable an index, you can use the ALTER INDEX statement.

Create Clustered Index 4-5

ENABLEINDEX

This statement uses the ALTER INDEX statement to 'enable' or rebuild an index on a table.

Syntax

```
ALTER INDEX index_name  
ON table_name  
REBUILD;
```

The screenshot shows a SQL query window titled 'SQLQuery1.sql - DE...D6\Lenovo pc (52)*'. The query is:

```
ALTER INDEX index_part_id  
ON production.parts  
REBUILD;
```

The results pane shows the message: 'Commands completed successfully.' with a completion time of '2020-10-20T15:19:13.936457+05:30'. The window has a title bar 'EnableIndex'.

Instructions to the Trainer(s):

- Using Slide 30, explain how to enable index.
- Enabling an index is possible, after the update to the table is completed.
- To enable an index, use the ALTER INDEX statement, it enables or rebuilds an index on a table.

Create Clustered Index 5-5

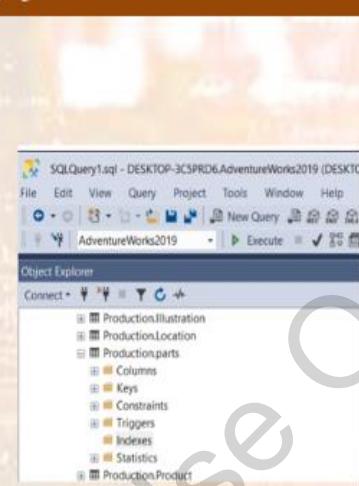
DROP INDEX

DROP INDEX statement removes one or more indexes from the current database.

Following is the syntax for DROP INDEX statement:

Syntax

```
DROP INDEX [IF EXISTS]
index.name
ON table.name;
```



© Aptech Ltd. Session 11 / 31

Instructions to the Trainer(s):

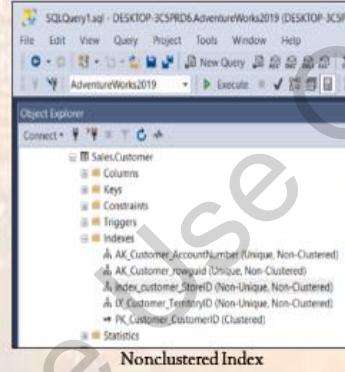
- Using Slide 31, explain how to drop an index.
- Whenever you want to delete some data from the table, delete columns or delete the entire table, you use the DROP INDEX statement.
- Any unwanted data that is occupying more space or storage can be deleted, so that the important data is stored and accessed whenever required.
- To Drop a data (whether table/column) use the DROP INDEX statement.

NonClustered Indexes

- It improves speed of data retrieval from tables. Unlike a clustered index, it sorts and stores data separately from the data rows in table.

Syntax

```
CREATE [NONCLUSTERED] INDEX index_name  
ON table_name(column_list);
```



© Aptech Ltd.

Session 11 / 32

Instructions to the Trainer(s):

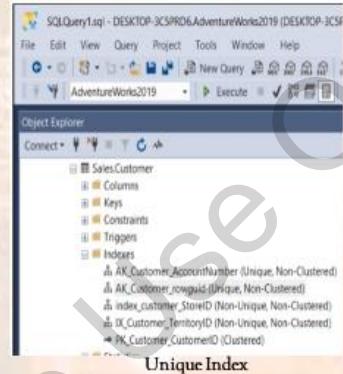
- Using Slide 32, explain the concept of NonClustered Indexes.
- A non-clustered index improves speed of data retrieval from the table.
- Unlike the clustered index, it sorts and stores data separately from the data rows in a table.

Unique Indexes

- Ensures that index key columns do not contain any duplicate values.
- It may consist of one or many columns.
- In case the unique index has multiple columns, the combination is unique.

Syntax

```
CREATE UNIQUE INDEX  
index_name  
ON table_name(column_list);
```



© Aptech Ltd.

Session 11 / 33

Instructions to the Trainer(s):

- Using Slide 33, explain unique indexes.
- A unique index guarantees that the index key contains no duplicate values and therefore, every row in the table is unique.
- Unique indexes consist of one or more columns.
- Suppose if there is some data or part of data that is common in the Unique index, the combination is then unique.

Filtered Indexes

➤ It is a nonclustered index with a predicate that allows to specify which rows should be added to the index.

Syntax

```
CREATE INDEX index_name  
ON table.name(column_list)  
WHERE predicate;
```

© Aptech Ltd. Session 11 / 34

Instructions to the Trainer(s):

- Using Slide 34, explain filtered indexes.
- A filtered index is an optimized nonclustered index especially suited to cover queries that select from a well-defined subset of data.
- A well-designed filtered index can improve query performance as well as reduce index maintenance and storage costs compared with full-table indexes.
- Filtered index is a nonclustered index with a predicate that allows to specify the rows that are added to the table.

Partitioned Table and Indexes

- SQL Server supports both table and index partitioning.
- Partitioning large tables or indexes can have the following manageability and performance benefits.

Benefits of Partitioning

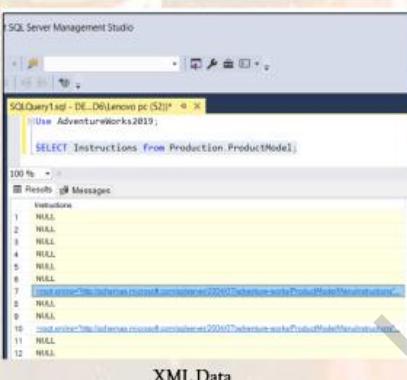
- Transfer or access subsets of data quickly and efficiently.
- Perform maintenance operations on one or more partitions. These operations are more efficient because they target only these data subsets, instead of the whole table.
- Improve query performance, based on the types of queries you frequently run and on your hardware configuration.

Instructions to the Trainer(s):

- Using Slide 35, explain the concept of partitioned table and indexes.
- Partitioning divides data into subsets. This makes large tables or indexes more manageable. Partitioning enables you to access data quickly and efficiently.
- Maintenance operations on subsets of data are performed more efficiently because they target only the required subset of data instead of the entire table.
- By default, a table or an index has only one partition that contains all the data or index pages.
- When a table or index uses multiple partitions, the data is partitioned horizontally into groups of rows.
- Benefits of Partitioning:
 - You can transfer or access subsets of data quickly and efficiently, while maintaining the integrity of a data collection.
 - You can perform maintenance operations on one or more partitions more quickly. The operations are more efficient because they target only these data subsets, instead of the whole table.
 - You may improve query performance, based on the types of queries you frequently run and on your hardware configuration.

XML Indexes 1-2

- XML data is stored in xml type columns as large binary objects (BLOBs).
- These XML instances can be large and the stored binary representation.



The screenshot shows a SQL Server Management Studio window titled 'SQLQuery1.sql - DE_D91Lenovo pc (SQL) - 4.x'. The query 'SELECT Instructions FROM Production.ProductModel' is run against the 'AdventureWorksLT' database. The results grid is labeled 'XML Data' at the bottom. It contains 12 rows, each showing a single value: 'NULL'. Row 7 contains a long XML string starting with '<root><Instructions>...</Instructions>'.

© Aptech Ltd. Session 11/36

Instructions to the Trainer(s):

- Using Slide 36, explain the XML Indexes.
- The xml data type is used to store XML documents and fragments as shown in the figure.
- An XML fragment is an XML instance that has a single top-level element missing.
- Due to the large size of XML columns, queries that search within these columns can be slow.
- You can speed up these queries by creating an XML index on each column.
- An XML index can be a clustered or a nonclustered index.

XML Indexes 2-2

➤ Primary XML index contains all the data in XML column.

Primary XML Index

➤ Secondary XML index creates a more specific index, based on the primary index.

Secondary XML Index

© Aptech Ltd. Session 11 / 37

Instructions to the Trainer(s):

- Using Slide 37, explain the types of XML Indexes.
- XML indexes can be created on a table only if there is a clustered index based on the primary key of the table. This primary key cannot exceed 15 columns.
- Different types of XML indexes are as follows:
 - **Primary XML Indexes** - The process of carrying out queries within an XML column can sometimes be slow. A primary XML index is created on each XML column to speed up these queries. It is a special index that shreds the XML data to store information.
 - **Secondary XML Indexes** - Secondary XML indexes are specialized XML indexes that help with specific XML queries. Features of secondary XML indexes are as follows:
 - Searching for values anywhere in the XML document.
 - Retrieving particular object properties from within an XML document.

Secondary XML indexes can only be created on columns that already have a primary XML index.

In-Class Question:

Question: What are different types of secondary indexes?

Answer: PATH Secondary index, VALUE Secondary index and PROPERTY Secondary index.

Columnstore Indexes

➤ Creating a column store index is done by using CREATE COLUMNSTORE INDEX command and has many of the same options as a regular index.

The screenshot shows the SQL Server Management Studio interface. In the Object Explorer, under the 'Sales.SalesOrderDetail' node, there are several indexes listed: AK_SalesOrderDetail_rowguid (Unique, Non-Clustered), IX_SalesOrderDetail_ProductID (Non-Unique, Non-Clustered), IX_SalesOrderDetail_OrderQty_Columnstore (Non-Unique, Columnstore), and PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID (Clustered). Below the Object Explorer, a large box contains the 'Estimated Execution Plan for Columnstore Index'. The plan shows a 'SELECT Cost: 0 %' query node connected to a 'Hash Match (Aggregate)' node, which is connected to a 'Columnstore Index Scan (NonClustered)' node for the '[SalesOrderDetail].[IX_SalesOrderDetail_OrderQty]' index. The cost for the scan is 78 %, and the cost for the aggregate is 22 %. The overall cost for the query is 0 %.

© Aptech Ltd. Session 11 / 38

Instructions to the Trainer(s):

- Using Slide 38, explain the Columnstore Indexes.
- Column Store Index is a new feature in SQL Server. It enhances performance of data warehouse queries extensively.
- The regular indexes or heaps of older SQL Servers stored data in B-Tree structure row-wise, but the column store index in SQL Server stores data column-wise.
- Since the data transfer rate is slow in database servers, so column store index uses compression aggressively to reduce the disk I/O required to serve the query request.
- The B-Tree and heap stores data row-wise, which means data from all the columns of a row are stored together contiguously on the same page.

Summary

- Indexes increase the speed of querying process by providing quick access to rows or columns in a data table.
- SQL Server stores data in storage units known as data pages.
- All input and output operations in a database are performed at the page level.
- A clustered index causes records to be physically stored in a sorted or sequential order.
- A nonclustered index is defined on a table that has data either in a clustered structure or a heap.
- XML indexes can speed up queries on tables that have XML data.
- Column Store Index enhances performance of data warehouse queries extensively.

Instructions to the Trainer(s):

- Show students Slide 39.
- Summarize the session by reading out each point on the slide.

Session 12: Triggers

12.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. Prepare a question or two that will be a key point to relate the current session objectives.

12.1.1 Teaching Skills

To teach this session, you should be well versed with the creation and alteration of triggers, performance implication of triggers, handling multiple rows in a session.

You should teach the concepts in the theory class using the images provided. For teaching in the class, you are expected to use slides and LCD projectors.

In-Class Activities

Follow the order given here during In-Class activities.

Slide 2

Objectives

- Explain triggers
- Explain the procedure to create and alter DML triggers
- Describe nested triggers
- Describe update functions
- Explain the handling of multiple rows in a session
- Explain the performance implication of triggers

© Aptech Ltd. Session 12 / 2

Instructions to the Trainer(s):

Give students a brief overview of the current session through the session objectives listed in Slide 2.

12.2 In-Class Explanations

Slide 3

Introduction

Trigger is a special type of stored procedure that is executed when an attempt is made to modify data in a table for which triggers are created.

Unlike standard system stored procedures, triggers cannot be executed directly, nor do they pass or receive parameters.

Triggers will get executed automatically when INSERT, UPDATE, or DELETE operations are performed.

In SQL Server, triggers are created using the CREATE TRIGGER statement.

The diagram shows a flow from an event to a trigger invocation. A green rounded rectangle labeled 'New record inserted' has a line pointing to a blue rounded rectangle labeled 'Trigger'. This 'Trigger' has an arrow pointing to a table labeled 'Employee_Details'. The table has columns 'EmployeeID' and 'EmployeeName', with four rows containing data: 1 Susan, 2 John, 3 Nancy, and 4 George. Below the table, the word 'Triggers' is written. The entire slide has a watermark reading 'For Classroom Use Only' diagonally across it.

© Aptech Ltd.

Session 12 / 3

Instructions to the Trainer(s):

- Using Slide 3, explain the concept of triggers.
- A trigger is a stored procedure that is executed when an attempt is made to modify data in a table protected by the trigger. Unlike standard system stored procedures, triggers cannot be executed directly, nor do they pass or receive parameters.
- Triggers are defined on specific tables and these tables are referred to as trigger tables.
- If a trigger is defined on the INSERT, UPDATE, or DELETE action on a table, it fires automatically when these actions are attempted.
- This automatic execution of the trigger cannot be circumvented. In SQL Server 2012, triggers are created using the CREATE TRIGGER statement.

In-Class Question:

What is a trigger?

Answer:

A trigger is a stored procedure that is executed when an attempt is made to modify data in a table protected by the trigger.

Use of Triggers I-4

- Triggers can contain complex processing logic and are generally used for maintaining low-level data integrity.
- The primary uses of triggers can be classified as follows:

Cascade changes in related tables

- Users can use a trigger to cascade changes through related tables.

For example,

Consider a table Sales.Customer in AdventureWorks2019 database having a foreign key, PersonID referencing the primary key, BusinessEntityID of the Person.Person table. If an update or a delete event occurs in the Person.Person table, an update or delete trigger can be defined to cascade these changes to the Sales.Customer table.

Instructions to the Trainer(s):

- Using Slide 4, explain the use of triggers.
- A trigger is a stored procedure that is executed when an attempt is made to modify data in a table.
- Triggers can contain complex processing logic and are generally used for maintaining low-level data integrity. The primary uses of triggers can be classified as follows:
- **Cascading changes in related tables:** Users can use a trigger to cascade changes through related tables. For example, consider a table Salary_Details having a FOREIGN KEY, EmpID referencing the PRIMARY KEY, and EmpID of the Employee_Details table. If an update or a delete event occurs in the Employee_Details table, an update, or delete trigger can be defined to cascade these changes to the Salary_Details table.

Use of Triggers 2-4

Enforce complex data integrity

DML Triggers	DDL Triggers	Logon Triggers
<ul style="list-style-type: none">DML triggers execute when data is inserted, modified, or deleted in a table or a view using the INSERT, UPDATE, or DELETE statements.	<ul style="list-style-type: none">DDL triggers execute when a table or a view is created, modified, or deleted using the CREATE, ALTER, or DROP statements.	<ul style="list-style-type: none">Logon triggers execute stored procedures when a session is established with a LOGON event. These triggers are invoked after the login authentication is complete and before the actual session is established. Logon triggers control server sessions by restricting invalid logins or limiting the number of sessions.

© Aptech Ltd.

Session 12/5

Instructions to the Trainer(s):

- Using Slide 5, explain how to enforce complex data integrity.
- Unlike CHECK constraints, triggers can reference the columns in other tables. This feature can be used to apply complex data integrity checks.
- Data integrity can be enforced by:
 - Checking constraints before cascading updates or deletes.
 - Creating multi-row triggers for actions executed on multiple rows.
- Enforcing referential integrity between databases.
- The enforce complex data integrity comprises:
 - **DML Triggers:** DML triggers execute when data is inserted, modified, or deleted. DML events include INSERT, UPDATE, or DELETE statements.
 - **DDL Triggers:** DDL triggers are used to restrict the DDL operations such as CREATE, ALTER, and DROP commands.
 - **Logon Triggers:** A logon trigger executes stored procedures when a session is established with a LOGON event.

Use of Triggers 3-4

Define custom error messages

- Used for providing more suitable or detailed explanations in certain error situations.
- Triggers can be used to invoke such predefined custom error messages when the relevant error conditions occur.

Maintain denormalized data

- Low-level data integrity can be maintained in denormalized database environments using triggers.
- It generally refers to redundant or derived data.

For example, if the value of the year is to be checked against complete dates, a trigger can be used to perform the check.

© Aptech Ltd. Session 12/ 6

Instructions to the Trainer(s):

- Using Slide 6, discuss custom error messages and maintain denormalized data concept.
- **Defining custom error messages:**
 - Custom error messages are used for providing more suitable or detailed explanations in certain error situations.
 - Triggers can be used to invoke such predefined custom error messages when relevant error conditions occur.
- **Maintaining denormalized data:**
 - Low-level data integrity can be maintained in denormalized database environments using triggers.
 - Denormalized data generally refers to redundant or derived data. Here, triggers are used for checks that do not require exact matches.
 - For example, if the value of the year is to be checked against complete dates, a trigger can be used to perform the check.

Use of Triggers 4-4

Compare before and after states of data

Triggers provide the option to reference changes that are made to data by INSERT, UPDATE, and DELETE statements.

This allows users to reference the affected rows when modifications are carried out through triggers.

© Aptech Ltd.

Session 12 / 7

Instructions to the Trainer(s):

- Using Slide 7, compare before and after states of data.
- Before triggers execute, the data is committed into the database and after triggers execute after the data has been inserted or updated in the database.
- Triggers provide the option to reference changes that are made to data by INSERT, UPDATE, and DELETE statements.
- This allows users to reference the affected rows when modifications are carried out through triggers.

Types of Triggers

Triggers are automatically executed when a language event occurs in a table or a view.

Language events can be classified as DML and DDL events.

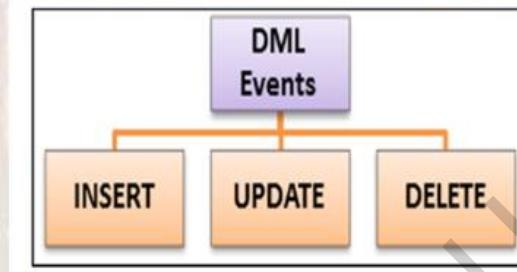
- Triggers associated with DML events are known as DML triggers, whereas triggers associated with DDL events are known as DDL triggers.

Instructions to the Trainer(s):

- Using Slide 8, explain the types of triggers.
- A trigger can be set to automatically execute an action when a language event occurs in a table or a view.
- Language events can be classified as DML events and DDL events.
- Triggers associated with DML events are known as DML triggers, whereas triggers associated with DDL events are known as DDL triggers.
- Triggers in SQL Server can be classified into three basic types:
 - **DML Triggers:** DML triggers execute when data is inserted, modified, or deleted in a table or a view using the INSERT, UPDATE, or DELETE statements.
 - **DDL Triggers:** DDL triggers execute when a table or a view is created, modified, or deleted using the CREATE, ALTER, or DROP statements.
 - **Logon Triggers:** Logon triggers execute stored procedures when a session is established with a LOGON event. These triggers are invoked after the login authentication is complete and before the actual session is established. Logon triggers control server sessions by restricting invalid logins or limiting the number of sessions.

Creating DML Triggers

- DML triggers are executed either on completion of the DML events or in place of the DML events.
- They enforce referential integrity by cascading changes to related tables when a row is modified.



© Aptech Ltd.

Session 12/9

Instructions to the Trainer(s):

- Using Slide 9, explain how to create DML Triggers.
- DML triggers are executed when DML events occur in tables or views. These DML events include the INSERT, UPDATE, and DELETE statements.
- DML triggers can execute either on completion of the DML events or in place of the DML events.
- DML triggers enforce referential integrity by cascading changes to related tables when a row is modified.
- DML triggers can perform multiple actions for each modification statement.
- DML events include:
 - INSERT, UPDATE, or DELETE statements

Introduction to Inserted and Deleted Tables

SQL statements in DML triggers use two special types of tables to modify data in the database.

When the data is inserted, updated, or deleted, SQL Server creates and manages these tables automatically.

Tables temporarily store the original as well as the modified data.

Inserted Table

- New rows are added to both the inserted table and the trigger table.
- The rows in the inserted table are copies of the new rows in the trigger table.

Deleted Table

- During the execution of a DELETE or UPDATE statement, rows are deleted from the trigger table and transferred to the deleted table.

Instructions to the Trainer(s):

- Using Slide 10, introduce Inserted and Deleted tables.
- The SQL statements in DML triggers use two special types of tables to modify data in the database.
- When the data is inserted, updated, or deleted, SQL Server creates and manages these tables automatically.
- The tables temporarily store the original as well as the modified data. These tables are as follows:
 - **Inserted Table:** The Inserted table contains copies of records that are modified with the INSERT and UPDATE operations on the trigger table. Trigger table is the table on which the trigger is defined. The INSERT and UPDATE operations insert new records into the Inserted and Trigger tables.
 - **Deleted Table:** The Deleted table contains copies of records that are modified with the DELETE and UPDATE operations on the trigger table. Trigger table is the table on which the trigger is defined. These operations delete the records from the trigger table and insert them in the Deleted table.

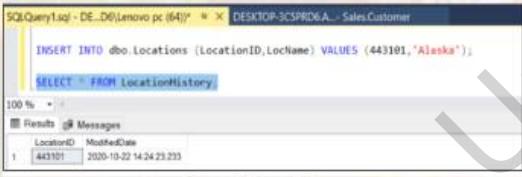
Insert Triggers

An INSERT trigger is executed when a new record is inserted in a table.

INSERT trigger ensures that the value being entered conforms to the constraints defined on that table.

When a user inserts a record in the table, INSERT trigger saves a copy of that record in inserted table.

It then checks whether the new value in the inserted table conforms to the specified constraints or not.



SQLQuery1.sql - DE...D0\lenovo pc (6A) | DESKTOP-5CSPRD6A... Sales.Customer

```
INSERT INTO dbo.locations (LocationID,LocName) VALUES (443101,'Alaska')
SELECT * FROM locationHistory
```

Results Messages

LocationID	ModifiedDate
443101	2020-10-22 14:24:23.233

Insert Trigger Test

© Aptech Ltd. Session 12/ 11

Instructions to the Trainer(s):

- Using Slide 11, explain the INSERT TRIGGER statement.
- An INSERT trigger is executed when a new record is inserted in a table. The INSERT trigger ensures that the value being entered conforms to the constraints defined on that table.
- When a user inserts a record in the table, the INSERT trigger saves a copy of that record in the Inserted table. It then checks whether the new value in the Inserted table conforms to the specified constraints.
- If the record is valid, the INSERT trigger inserts the row in the trigger table otherwise, it displays an error message.
- An INSERT trigger is created using the INSERT keyword in the CREATE TRIGGER and ALTER TRIGGER statements.

Update Triggers

- The UPDATE trigger copies the original record in the Deleted table and the new record into the Inserted table when a record is updated.

The screenshot shows a SQL query window titled "SQLQuery1.sql - DE...D6\Lenovo pc (64)*". The query is:

```
UPDATE dbo_LOCATIONS  
SET LocName='Atlanta'  
Where LocationID=443101;
```

Below the query, a "Results" tab displays the output of the SELECT statement:

LocationID	ModifiedDate
1	2020-10-22 14:24:23.233
2	2020-10-22 16:39:14.070

At the bottom of the window, it says "Update Trigger Test".

© Aptech Ltd.

Session 12 / 12

Instructions to the Trainer(s):

- Using Slide 12, explain the UPDATE TRIGGER statement.
- The UPDATE TRIGGER copies the original record in the Deleted table and the new record into the Inserted table when a record is updated. It then evaluates the new record to determine if the values conform to the constraints specified in the trigger table.
- If the new values are valid, the record from the Inserted table is copied to the trigger table.
- However, if the new values are invalid, an error message is displayed. Also, the original record is copied from the Deleted table back into the trigger table.
- An UPDATE trigger is created using the UPDATE keyword in the CREATE TRIGGER and ALTER TRIGGER statements.

Delete Triggers

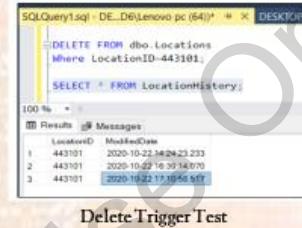
The DELETE trigger can be created to restrict a user from deleting a particular record in a table.

Following will happen if the user tries to delete the record:

The record is deleted from the trigger table and inserted in the Deleted table.

If there is a constraint on the record to prevent deletion, the DELETE trigger displays an error message.

The deleted record stored in the Deleted table is copied back to the trigger table.



```
SQLQuery1.sql - DE_D6\Lenovo pc (64) [ ] * X DESIYUR
DELETE FROM dbo.Locations
Where LocationID=443101;
SELECT * FROM LocationHistory;
100 % 100 % 100 %
Results Messages
LocationID ModifiedDate
1 443101 2020-10-22 16:39:48.233
2 443101 2020-10-22 16:39:48.070
3 443101 2020-10-22 17:10:45.517
```

Delete Trigger Test

Instructions to the Trainer(s):

- Using Slide 13, explain the DELETE TRIGGER statement.
- The DELETE trigger can be created to restrict a user from deleting a particular record in a table. Following will happen if the user tries to delete the record:
 - The record is deleted from the trigger table and inserted in the Deleted table.
 - It is checked for constraints against deletion.
 - If there is a constraint on the record to prevent deletion, the DELETE trigger displays an error message.
 - The deleted record stored in the Deleted table is copied back to the trigger table.
- A DELETE trigger is created using the DELETE keyword in the CREATE TRIGGER statement.

AFTER Triggers

- An AFTER trigger is executed on completion of INSERT, UPDATE, or DELETE operations.
- An AFTER trigger is executed when the constraint check in the table is completed.

AFTER Triggers

```

    graph TD
      A[INSERT] --> C[Process Completed]
      B[UPDATE] --> C
      D[DELETE] --> C
      C --> E[Execute]
      E --> F[AFTER Trigger]
  
```

SQL Query Test Results:

```

  SQLQuery1.sql - DE_D6\lenovo pc (64) [1] * DESKTOP-3CSPRD6A\SalesCustomer
  INSERT INTO dbo_LOCATIONS (LocationID,LocName) VALUES (443103,'SAN FRANCISCO');
  SELECT * FROM LocationHistory;
  100 %
  Results Messages
  LocationID ModifiedDate
  1 443101 2020-10-22 14:24:23.233
  2 443101 2020-10-22 16:39:14.070
  3 443101 2020-10-22 17:10:58.517
  4 443103 2020-10-22 18:45:24.730
  5 443103 2020-10-22 18:45:24.767
  After Insert Trigger Test
  
```

© Aptech Ltd. Session 12/ 14

Instructions to the Trainer(s):

- Using Slide 14, explain the AFTER trigger statement.
- An AFTER trigger is executed on completion of INSERT, UPDATE, or DELETE operations.
- AFTER triggers can be created only on tables. A table can have multiple AFTER triggers defined for each INSERT, UPDATE, and DELETE operation. If multiple AFTER triggers are created on the same table, the user must define the order in which the triggers must be executed.
- An AFTER trigger is executed when the constraint check in the table is completed. Also, the trigger is executed after the Inserted and Deleted tables are created.
- In the Code Snippet on Slide 14, If any employee record is deleted from the table, the AFTER DELETE trigger activates. The trigger displays the number of employee records deleted from the table.
- Tell the students that to specify the order for an AFTER trigger, use the sp_settriggerorder stored procedure. Following option can be used:
 - First
 - Last
 - None
- The first and last triggers must be two different DML triggers.

INSTEAD OF Triggers

- An INSTEAD OF trigger is executed in place of the INSERT, UPDATE, or DELETE operations.
- They are executed before constraint checks are performed on the table.
- These triggers are executed after the creation of the Inserted and Deleted tables.

```
SQLQuery1.sql - not connected* DESKTC
DELETE FROM dbo_LOCATIONS
Where LocationID=443101;
SELECT * FROM Locations;
```

100 %

Results Messages

Message

1 Sample Instead of trigger

Instead of Trigger Test

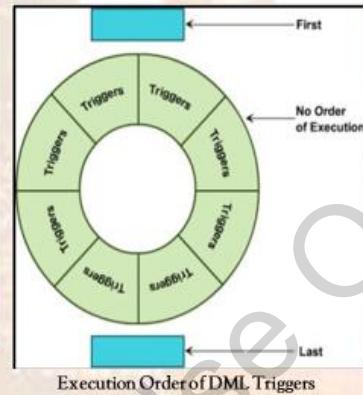
© Aptech Ltd. Session 12 / 15

Instructions to the Trainer(s):

- Using Slide 15, explain the Instead of triggers.
- An INSTEAD OF trigger is executed in place of the INSERT, UPDATE, or DELETE operations.
- INSTEAD OF triggers can be created on tables as well as views.
- A table or a view can have only one INSTEAD OF trigger defined for each INSERT, UPDATE, and DELETE operation.
- The INSTEAD OF triggers are executed before constraint checks are performed on the table.
- These triggers are executed after the creation of the Inserted and Deleted tables.
- The INSTEAD OF triggers increase the variety of types of updates that the user can perform against the view.

Execution Order of DML Triggers

- SQL Server allows users to specify which AFTER trigger is to be executed first and which is to be executed last.
- All the triggering actions have a first and last trigger defined for them.
- However, no two triggering actions on a table can have the same first and last triggers.

**Instructions to the Trainer(s):**

- Using Slide 16, explain the execution order of DML Triggers.
- SQL Server allows users to specify which AFTER trigger is to be executed first and last.
- All the triggering actions have a first and last trigger defined for them.
- However, no two triggering actions can have the same first and last triggers.

Viewing Definitions of DML Triggers

A trigger definition includes the trigger name, the table on which the trigger is created, the triggering actions, and the SQL statements that are executed.

SQL Server provides `sp_helptext` stored procedure to retrieve the trigger definitions.

Instructions to the Trainer(s):

- Using Slide 17, discuss how definitions can be viewed of DML Triggers.
- A trigger definition includes the trigger name, the table on which the trigger is created, the triggering actions, and the SQL statements that are executed.
- SQL Server provides `sp_helptext` stored procedure to retrieve the trigger definitions.

Modifying Definitions of DML Triggers

Trigger parameters are defined at the time of creating a trigger.

These parameters include the type of triggering action that invokes the trigger and the SQL statements that are executed.

```
graph TD; A[Modifying DML Triggers] --> B[DROP and RECREATE]; A --> C[ALTER TRIGGER statement];
```

SQLQuery1.sql - DE...D6\Lenovo pc (63)*

```
sp_helptext TRIGGER_UPDATE_Locations;
```

100 %

Messages

The text for object 'TRIGGER_UPDATE_Locations' is encrypted.

Completion time: 2020-10-23T09:33:43.0152002+05:30

Encrypted Trigger

© Aptech Ltd.

Session 12/18

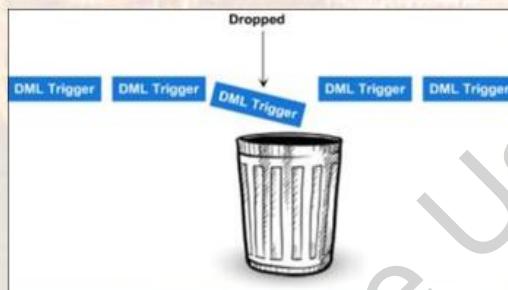
Instructions to the Trainer(s):

- Using Slide 18, explain how to modify definition of DML triggers.
- Trigger parameters are defined at the time of creating a trigger. These parameters include the type of triggering action that invokes the trigger and the SQL statements that are executed.
- If the user wants to modify any of these parameters for a DML trigger, a user can do so in any one of the two ways:
 - Drop and re-create the trigger.
 - ALTER TRIGGER statement for changing any data.

Dropping DML Triggers

SQL Server provides the option of dropping a DML trigger created on a table if the trigger is no longer required.

When a table is dropped, all the triggers defined on that table are also dropped.

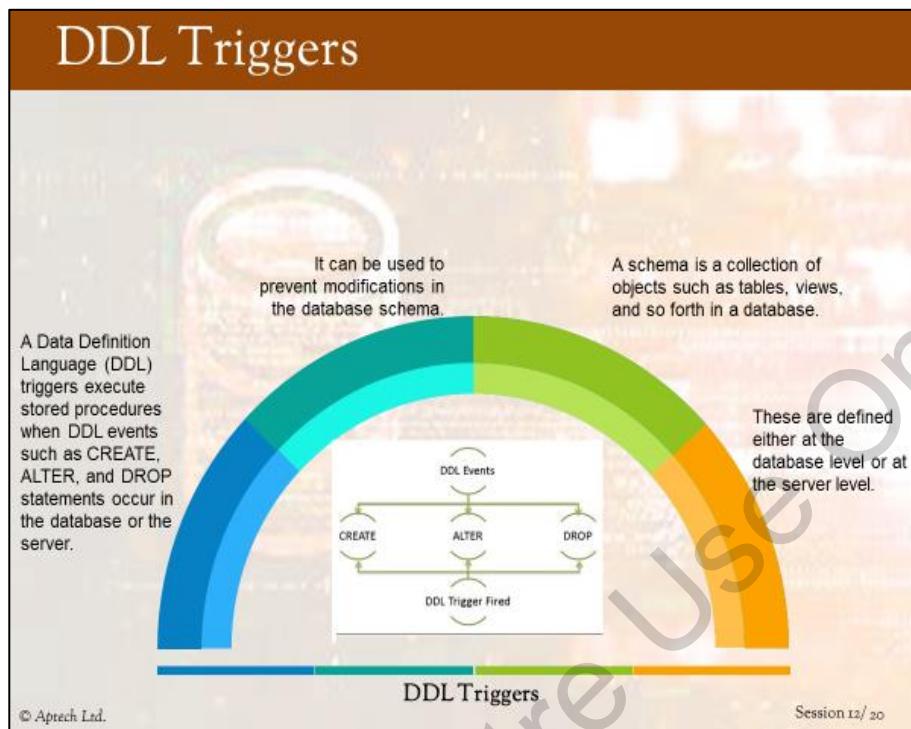


© Aptech Ltd.

Session 12 / 19

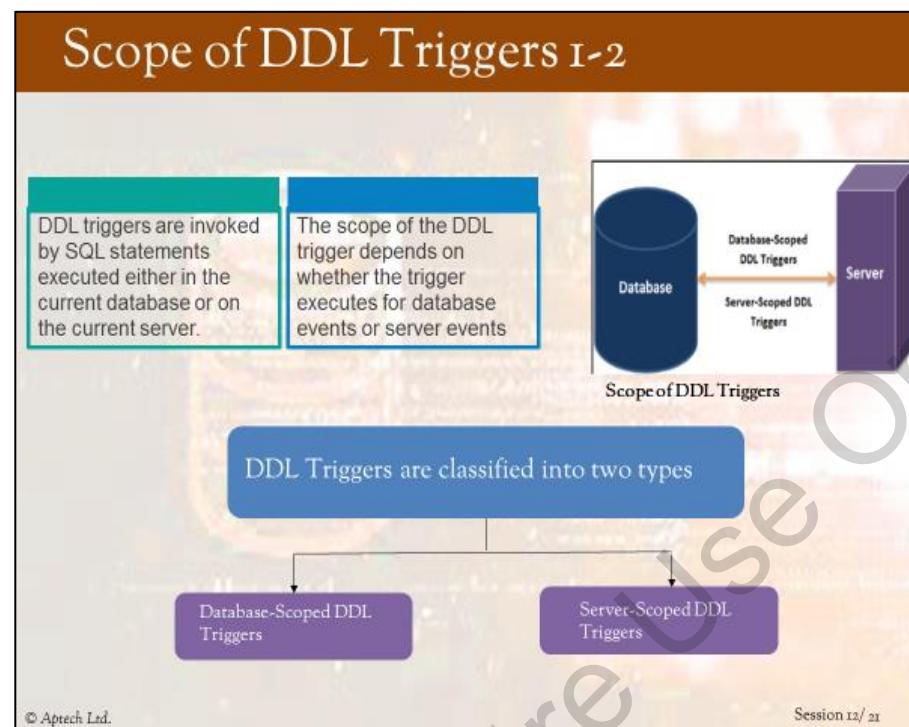
Instructions to the Trainer(s):

- Using Slide 19, explain how to drop DML triggers.
- SQL Server provides the option of dropping a DML trigger created on a table if the trigger is no longer required.
- The trigger can be dropped using the DROP TRIGGER statement.
- Multiple triggers can also be dropped using a single drop trigger statement.
- When a table is dropped, all the triggers defined on that table are also dropped.
- When the DML trigger is deleted from the table, the information about the trigger is also removed from the catalog views.



Instructions to the Trainer(s):

- Using Slide 20, explain the DDL triggers.
- A Data Definition Language (DDL) triggers execute stored procedures when DDL events such as CREATE, ALTER, and DROP statements occur in the database or the server.
- DDL triggers can operate only on completion of the DDL events.
- DDL triggers can be used to prevent modifications in the database schema. A schema is a collection of objects such as tables, views, and so forth in a database.
- DDL triggers can invoke an event or display a message based on the modifications attempted on the schema. DDL triggers are defined either at the database level or at the server level.



Instructions to the Trainer(s):

- Using Slide 21, explain the scope of DDL triggers.
- DDL triggers are invoked by SQL statements executed either in the current database or on the current server. For example, a DDL trigger created for a CREATE TABLE statement executes on the CREATE TABLE event in the database.
- A DDL trigger created for a CREATE LOGIN statement executes on the CREATE LOGIN event in the server.

In-Class Question:

Question: What are two scopes of DDL triggers?

Answer: Database-Specific DDL and Server-Specific DDL are two scopes of DDL triggers.

Scope of DDL Triggers 2-2

<ul style="list-style-type: none">• They are invoked by the events that modify the database schema.• Stored in the database and execute on DDL events, except those related to temporary tables.	<ul style="list-style-type: none">• Server-scoped DDL triggers are invoked by DDL events at the server level.• Stored in the master database.
Database-SScoped DDL Triggers	Server-SScoped DDL Triggers

© Aptech Ltd. Session 12 / 22

Instructions to the Trainer(s):

- Using Slide 22, explain the scope of DDL Triggers.
- The scope of the DDL trigger depends on whether the trigger executes for database events or server events.
- Accordingly, the DDL triggers are classified into two types, which are as follows:
 - **Database-SScoped DDL:** Triggers Database-scoped DDL triggers that are invoked by the events that modify the database schema. These triggers are stored in the database and execute on DDL events, except those related to temporary tables.
 - **Server-SScoped DDL:** Triggers Server-scoped DDL triggers are invoked by DDL events at the server level. These triggers are stored in the master database.

Nested Triggers

Both DDL and DML triggers are nested when a trigger implements an action that initiates another trigger.

DDL and DML triggers can be nested up to 32 levels.

If the nested triggers are allowed then, the triggers in the sequence start an infinite loop.

Nested triggers can be used to perform the functions such as storing the backup of the rows that are affected by the previous actions.

Instructions to the Trainer(s):

- Using Slide 23, explain the nested triggers.
- Both DDL and DML triggers are nested when a trigger implements an action that initiates another trigger.
- DDL and DML triggers can be nested up to 32 levels. Suppose if a trigger modifies a table on which there is another trigger, the second trigger is initiated, which then calls a third trigger, and so on.
- If the nested triggers are allowed, then the triggers in the sequence start an infinite loop. This will exceed the nesting level and the trigger will terminate.
- Nested triggers can be used to perform the functions such as storing the backup of the rows that are affected by previous actions.
- A Transact-SQL trigger executes the managed code through referencing a CLR routine, aggregate, or type, that references the counts as one level against the 32-level nesting limit.
- There are two types of recursion:
 - **Direct recursion:** When a trigger fires and performs an action that causes the same trigger to fire again.
 - **Indirect recursion:** When a trigger fires and performs an action that causes another trigger of the same type to fire.

UPDATE()

➤ UPDATE() function returns a Boolean value that specifies whether an UPDATE or INSERT action was performed on a specific view or column of a table.

➤ UPDATE() function can be used anywhere inside the body of a Transact-SQL UPDATE or INSERT trigger to test whether the trigger should execute some actions.

Instructions to the Trainer(s):

- Using Slide 24, discuss on UPDATE().
- UPDATE () function returns a Boolean value that specifies whether an UPDATE or INSERT action was performed on a specific view or column of a table.
- UPDATE () function can be used anywhere inside the body of a Transact-SQL UPDATE or INSERT trigger to test whether the trigger should execute some actions.

Handling of Multiple Rows in a Session

When a user writes the code for a DML trigger then, the statement that causes the trigger to fire will be single statement.

This single statement will affect multiple rows of data, instead of a single row.

This is a common behavior for DELETE and UPDATE triggers as these statements often affect multiple rows.

Instructions to the Trainer(s):

- Using Slide 25, explain how to handle multiple rows in a session.
- When a user writes the code for a DML trigger, then the statement that causes the trigger to fire will be a single statement.
- This single statement will affect multiple rows of data, instead of a single row. This is a common behavior for DELETE and UPDATE triggers as these statements often affect multiple rows.
- The behavior for INSERT triggers is less common as the basic INSERT statement adds only one row.
- When the functionality of a DML trigger involves automatically recalculating summary values of one table and storing the result in another table, then multirow considerations are important.

LOGON Triggers

- Logon triggers fire stored procedures in response to a LOGON event.
- This event is raised when a user session is established with an instance of SQL Server.

LOGON triggers are created at the server level and are useful in following cases:

- To audit login activity
- To control the login activity

Instructions to the Trainer(s):

- Using Slide 26, explain the LOGON Triggers.
- Logon triggers fire stored procedures in response to a LOGON event.
- This event is raised when a user session is established with an instance of SQL Server.
- LOGON triggers are created at the server level and are useful in following cases:
 - To audit login activity
 - To control the login activity

Performance Implication of Triggers

Triggers do not carry overheads, rather they are quite responsive.

Many performance issues can occur because of the logic present inside the trigger.

Similarly, consider that the trigger executes various SQL statements against other tables separate from the Inserted and Deleted tables.

This will again result in the slowdown of speed of SQL statements that are within the trigger.

© Aptech Ltd.

Session 12 / 27

Instructions to the Trainer(s):

- Using Slide 27, explain the performance implications of triggers.
- In reality, triggers do not carry overheads, rather they are quite responsive. However, many performance issues can occur because of the logic present inside the trigger. Suppose a trigger creates a cursor and loops through many rows, then there will be a slowdown in the process.
- Similarly, consider that the trigger executes various SQL statements against other tables separate from the Inserted and Deleted tables. This will again result in the slowdown of speed of SQL statements that are within the trigger.
- A good rule will be to keep the logic simple within the triggers and avoid using cursors while executing statements against another table and different tasks that cause performance slowdown.

Summary

- A trigger is a stored procedure that is executed when an attempt is made to insert, update, or delete data in a table that is protected by the trigger.
- Logon triggers execute stored procedures when a session is established with a LOGON event.
- DML triggers are executed when DML events occur in tables or views.
- The INSERT trigger is executed when a new record is inserted in a table.
- The UPDATE trigger copies the original record in the Deleted table and the new record into the Inserted table when a record is updated.
- The DELETE trigger can be created to restrict a user from deleting a particular record in a table.
- The AFTER trigger is executed on completion of INSERT, UPDATE, or DELETE operations.

Instructions to the Trainer(s):

- Show students Slide 28.
- Summarize the session by reading out each point on the slide.

Session 13: Programming Transact-SQL

13.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. Prepare a question or two that will be a key point to relate the current session objectives.

13.1.1 Teaching Skills

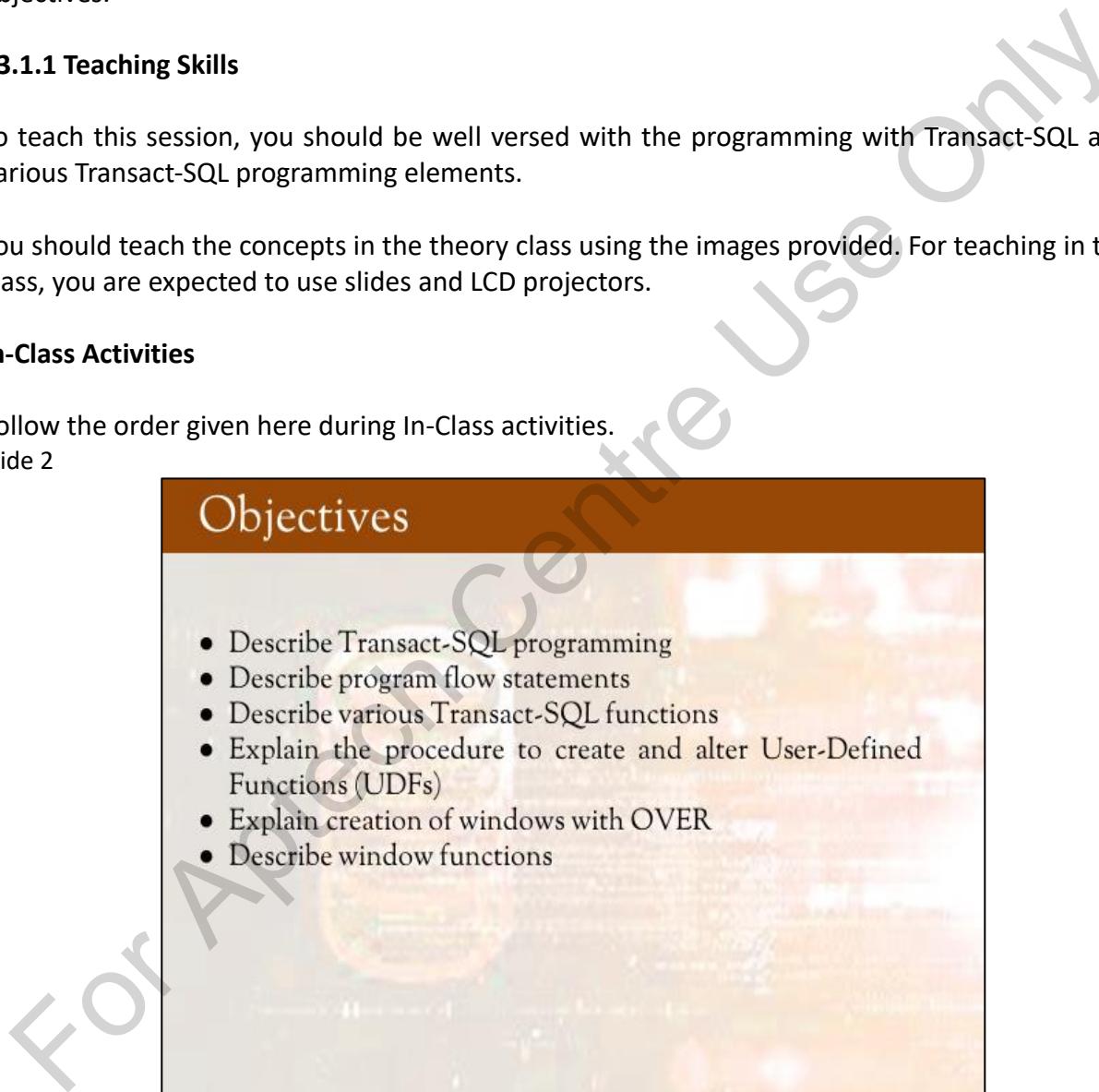
To teach this session, you should be well versed with the programming with Transact-SQL and various Transact-SQL programming elements.

You should teach the concepts in the theory class using the images provided. For teaching in the class, you are expected to use slides and LCD projectors.

In-Class Activities

Follow the order given here during In-Class activities.

Slide 2



Objectives

- Describe Transact-SQL programming
- Describe program flow statements
- Describe various Transact-SQL functions
- Explain the procedure to create and alter User-Defined Functions (UDFs)
- Explain creation of windows with OVER
- Describe window functions

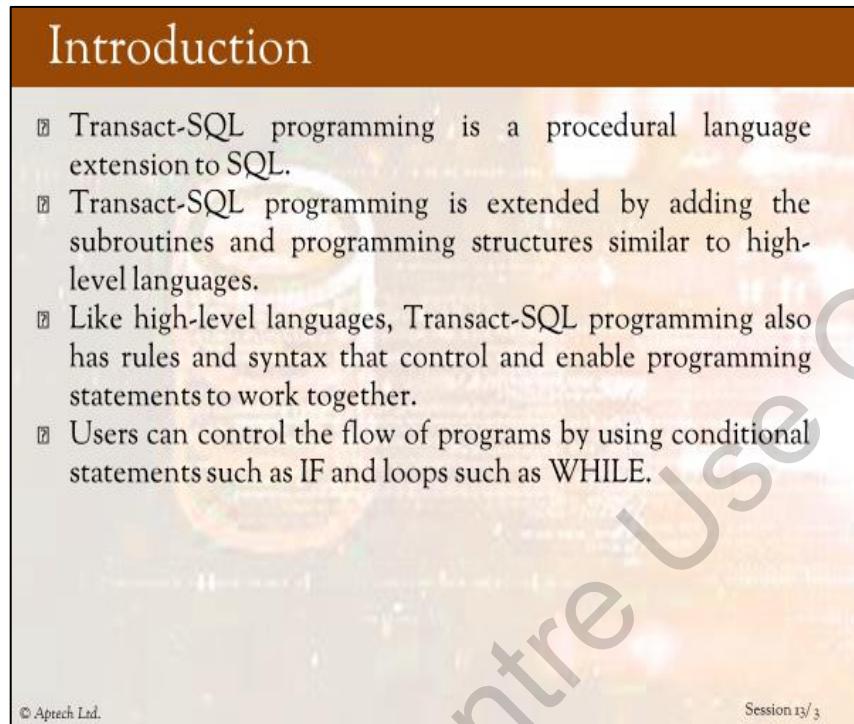
© Aptech Ltd. Session 13 / 2

Instructions to the Trainer(s):

Give students a brief overview of the current session through the session objectives listed in Slide 2.

13.2 In-Class Explanations

Slide 3



The slide has a brown header bar with the word "Introduction". The main content area contains a bulleted list of five points about Transact-SQL programming. At the bottom left is the copyright notice "© Aptech Ltd." and at the bottom right is "Session 13 / 3". A large watermark "For Author Use Only" is diagonally across the slide.

Introduction

- Transact-SQL programming is a procedural language extension to SQL.
- Transact-SQL programming is extended by adding the subroutines and programming structures similar to high-level languages.
- Like high-level languages, Transact-SQL programming also has rules and syntax that control and enable programming statements to work together.
- Users can control the flow of programs by using conditional statements such as IF and loops such as WHILE.

© Aptech Ltd. Session 13 / 3

Instructions to the Trainer(s):

- Introduce Transact-SQL using Slide 3.
- Transact-SQL programming is a procedural language extension to SQL.
- Transact-SQL programming is extended by adding the subroutines and programming structures similar to high-level languages.
- Similar to high-level languages, Transact-SQL programming also has rules and syntax that control and enable programming statements to work together.
- Users can control the flow of programs by using conditional statements such as IF and loops such as WHILE.

For more information, refer to:

<https://searchdatamanagement.techtarget.com/definition/T-SQL>

<https://www.dataquest.io/blog/sql-vs-t-sql/>

https://www.tutorialspoint.com/t_sql/index.htm

Transact-SQL Programming Elements 1-3

- Transact-SQL programming elements enable to perform various operations that cannot be done in a single statement.
- Users can group several Transact-SQL statements together by using one of the following ways:

Batches

- A batch is a collection of one or more Transact-SQL statements that are sent as one unit from an application to the server.

Stored Procedures

- A stored procedure is a collection of Transact-SQL statements that are precompiled and predefined on the server.

Instructions to the Trainer(s):

- Using Slide 4, explain Transact-SQL programming elements.
- Transact-SQL programming elements enable to perform various operations that cannot be done in a single statement. Users can group several Transact-SQL statements together by using one of the following ways:
 - **Batches:** In Transact-SQL, a batch is a set of SQL statements submitted together and executed.
 - **Stored Procedures:** Stored procedures are a collection of Transact-SQL statements stored within the database.

Transact-SQL Programming Elements 2-3

Triggers <ul style="list-style-type: none">A trigger is a special type of stored procedure that is executed when the user performs an event such as an INSERT, DELETE, or UPDATE operation on a table.	Scripts <ul style="list-style-type: none">A script is a chain of Transact-SQL statements stored in a file that is used as input to the SSMS code editor or sqlcmd utility.
---	---

© Aptech Ltd. Session 13 / 5

Instructions to the Trainer(s):

- Using Slide 5, explain triggers and scripts in Transact SQL.
- **Triggers:**
 - A trigger is a special type of stored procedure that automatically runs when an event occurs in the database server.
 - DML events are INSERT, UPDATE, or DELETE statements on a table or view.
- **Scripts:**
 - It is a set of T-SQL (Transact-SQL) statements stored in a file that is used as an input to SSMS code.

For more information, refer to:

<https://docs.microsoft.com/en-us/sql/t-sql/statements/create-trigger-transact-sql?view=sql-server-ver15>

<https://www.sqlshack.com/learn-sql-script/>

Transact-SQL Programming Elements 3-3

Following features enable users to work with Transact-SQL statements:

Variables

- A variable allows a user to store data that can be used as input in a Transact-SQL statement.

Control-of-flow

- Control-of-flow is used for including conditional constructs in Transact-SQL.

Error Handling

- Error handling is a mechanism that is used for handling errors and provides information to the users about the error occurred.

Instructions to the Trainer(s):

- Using Slide 6, discuss the concept of variables, control-of-flow, and error handling in T-SQL.
- **Variables:** A variable allows a user to store data that can be used as input in a T-SQL statement.
- **Control-of-flow:** Control-of-flow is used for including conditional constructs in Transact-SQL code execution.
- **Error handling:** Error handling in SQL Server gives us control over Transact-SQL code. It is a mechanism used for handling errors and provides information to its users.

Transact-SQL Batches 1-2

A Transact-SQL batch is a group of one or more Transact-SQL statements sent to the server as one unit from an application for execution.

In the execution plan, the SQL statements are executed one by one. It should be terminated with a semicolon.

A compile error such as syntax error restricts the compilation of the execution plan. So, if a compile-time error occurs, no statements in the batch are executed.

Instructions to the Trainer(s):

- Using Slide 7, explain Transact-SQL batches.
- A Transact-SQL batch is a group of one or more Transact-SQL statements sent to the server as one unit from an application for execution.
- SQL Server compiles the batch SQL statements into a single executable unit, also called as an execution plan. In the execution plan, the SQL statements are executed one by one.
- A Transact-SQL batch statement should be terminated with a semicolon. This condition is not mandatory, but the facility to end a statement without a semicolon is deprecated and may be removed in the new versions of SQL Server in the future. Hence, it is recommended to use semicolons to terminate batches.
- A compile error such as syntax error restricts the compilation of the execution plan.
- Hence, if a compile-time error occurs, no statements in the batch are executed.

Transact-SQL Batches 2-2

A run-time error such as a constraint violation or an arithmetic overflow has one of the following effects:

- Most of the run-time errors stop the current statement and the statements that follow in the batch.
- A specific run-time error such as a constraint violation stops only the existing statement and the remaining statements in the batch are executed.

© Aptech Ltd. Session 13/ 8

Instructions to the Trainer(s):

- Using Slide 8, explain the Transact-SQL batches.
- A run-time error such as a constraint violation or an arithmetic overflow has one of the following effects:
 - Most of the run-time errors stop the current statement and the statements that follow in the batch.
 - A specific run-time error such as a constraint violation stops only the existing statement and the remaining statements in the batch are executed.
- The SQL statements that execute before the run-time error is encountered are unaffected. The only exception is when the batch is in a transaction and the error results in the transaction being rolled back.

Transact-SQL Variables 1-3

- Variables allow users to store data for using as input in a Transact-SQL statement.

For example,

- Users can create a query that requires various types of data values specified in WHERE clause each time the query is executed.
- Here, the users can write logic to store variables with appropriate data.

Instructions to the Trainer(s):

- Using Slide 9, explain Transact-SQL variables.
- Variables allow users to store data to be used as input in a Transact-SQL statement.
- For example, users can create a query that requires various types of data values specified in the WHERE clause each time the query is executed. Here, the users can use variables in the WHERE clause and write the logic to store the variables with the appropriate data.

Transact-SQL Variables 2-3

DECLARE

- Variables are declared with the DECLARE statement in the body of a batch.
- These variables are assigned values by using the SELECT or SET statement.
- The variables are initialized with NULL values if the user has not provided a value at the time of the declaration.

The screenshot shows a SQL Server Management Studio window titled 'SQLQuery1.sql - DE-D9\lenovo pc (S2)'. The code in the query pane is:

```

GO
USE AdventureWorks2019;
GO
DECLARE @Find varchar(30) = 'Man%';
SELECT p.LastName, p.FirstName, ph.PhoneNumber FROM Person.Person AS p
JOIN Person.PersonPhone AS ph ON p.BusinessEntityID = ph.BusinessEntityID
WHERE LastName LIKE @Find;
  
```

The results pane displays a table with three rows of data:

	Lastname	FirstName	PhoneNumber
1	Manchepalli	Ajay	1 (11) 550 555-0174
2	Mane	Paul	1 (11) 550 555-0140
3	Mansurov	Tomas	1 (11) 550 555-0178

Below the results, a 'Contact Information' section is visible. The status bar at the bottom right indicates 'Session 13 / 10'.

© Aptech Ltd.

Session 13 / 10

Instructions to the Trainer(s):

- Using Slide 10, explain DECLARE statement in T-SQL.
- SQL Server provides DECLARE statement to set and declare local variables:
 - **DECLARE:**
 - Variables are declared with DECLARE statement in the body of a batch.
 - These variables are assigned values by using SELECT or SET statement.
 - The variables are initialized with NULL values if the user has not provided a value at the time of the declaration.

Transact-SQL Variables 3-3

SET

- The SET statement sets the local variable created by the DECLARE statement to the specified value.

SELECT

- The SELECT statement indicates that the specified local variable that was created using DECLARE should be set to the given expression.

The screenshot shows the SQL Server Management Studio interface. The Query Editor window contains the following T-SQL code:

```
USE AdventureWorks2019;
GO
DECLARE @var1 nvarchar(30);
SELECT @var1 = 'Unnamed Company';
SELECT @var1 = Name FROM Sales.Store
WHERE BusinessEntityID = 10;
SELECT @var1 AS 'Company Name';
```

The Results pane displays the output of the query:

Company Name
1 Unnamed Company

A large watermark reading "RE USE" is diagonally across the background.

Instructions to the Trainer(s):

- Using Slide 11, explain SET and SELECT statements in T-SQL.
 - Tell students that SET statement sets the local variable created by DECLARE statement to a specified value.
 - The SELECT statement indicates that specified local variable that was created using DECLARE should be set to given expression.
 - Differences between SET and SELECT statements:
 - It is possible to assign only one variable at a time using SET. However, using SELECT, you can make multiple assignments at once.
 - SET can only assign a scalar value while assigning from a query. It raises an error and does not work if the query returns multiple values/rows. However, SELECT assigns one of the returned values to the variable and the user will not even know that multiple values were returned.

Synonyms 1-5

Synonyms are database objects that serve following purposes:

They offer another name for a different database object, also called as the base object.

They present a layer of abstraction that guards a client application from the modifications made to the location and the name of the base object.

Database Objects

Extended stored procedure
SQL table-valued function
SQL stored procedure
Table (User-defined)
Replication-filter-procedure
SQL scalar function
SQL inline-table-valued function
View

© Aptech Ltd. Session 13 / 12

Instructions to the Trainer(s):

- Using Slide 12, explain synonyms in SQL Server 2019.
- Synonyms are database objects that serve following purposes:
 - They offer another name for a different database object, also called as the base object, which may exist on a remote or local server.
 - They present a layer of abstraction that guards a client application from the modifications made to the location and the name of the base object.

Synonyms 2-5

- Suppose users want to create a synonym and have a default schema that is not owned by them.
- In such a case, they can qualify the synonym name with the schema name that they actually own.

Synonyms and Schemas

For example,

- A user owns a schema Resources, but Materials is the user's default schema.
- If this user wants to create a synonym, he/she must prefix the name of the synonym with the schema Resources.

Instructions to the Trainer(s):

- Using Slide 13, continue explaining synonyms and also, explain about schemas in T-SQL.
- **Synonyms and Schemas:** Suppose users want to create a synonym and have a default schema that is not owned by them. In such a case, they can qualify the synonym name with the schema name that they actually own. Consider that a user owns a schema Resources, but Materials is the user's default schema. If this user wants to create a synonym, he/she must prefix the name of the synonym with the schema Resources.

Synonyms 3-5

Granting Permissions on Synonyms

Users can deny, grant, or revoke all or any of the permissions on a synonym.

Only members of the roles db_owner or db_ddladmin or synonym owners are allowed to grant permissions on a synonym.

Permissions
DELETE
INSERT
TAKE OWNERSHIP
VIEW DEFINITION
CONTROL
EXECUTE
SELECT
UPDATE

© Aptech Ltd. Session 13/14

Instructions to the Trainer(s):

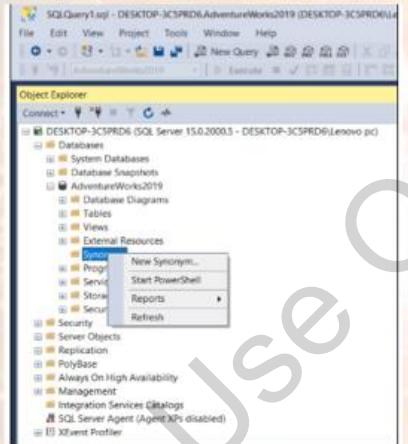
- Using Slide 14, explain the concept of granting permissions on Synonyms.
- **Granting Permissions on Synonyms:** Only members of the roles db_owner or db_ddladmin or synonym owners are allowed to grant permissions on a synonym. Users can deny, grant, or revoke all or any of the permissions on a synonym. Table displays the list of permissions that are applied on a synonym.
- Different permissions granted are as follows:
 - DELETE
 - INSERT
 - TAKE OWNERSHIP
 - VIEW DEFINITION
 - CONTROL
 - EXECUTE
 - SELECT
 - UPDATE

Synonyms 4-5

Working with Synonyms

To create a synonym using SSMS, perform the following steps:

- In Object Explorer, expand the database to create a new synonym.
- Select the Synonyms folder, right-click it and then, click New Synonym.



Creating a New Synonym

© Aptech Ltd. Session 13/15

Synonyms 5-5

Synonym name New name for the object. Here, Emp is the name.

Synonym schema Is the new name for the schema object. Here, HumanResources is used for the synonym and the object type.

Server name Name of the server to be connected. Here, the server name is specified as 10.2.110.140.

Database name Connects the object. Here, AdventureWorks2019 is the database name.

© Aptech Ltd. Session 13/16

Instructions to the Trainer(s):

- Using Slides 15 and 16, discuss how to create synonyms.
- To create a synonym in SQL Server 2019 using SSMS, one must perform following steps:
 - In Object Explorer, expand the database to create a new synonym.

- Select the Synonym folder, right-click it, and then, click New Synonym.
- Using Slide 16, explain the terminology concerning synonyms:
- **Synonym name:** Is the new name for the object.
 - **Synonym schema:** is the new name for the schema object.
 - **Server name:** Is the name of the server to be connected.
 - **Database name:** Is the database name to connect the object.
 - **Schema:** Is the schema that owns the object.

Program Flow Statements 1-2

Different types of program flow statements and functions supported by Transact-SQL are as follows:

➤ Transact-SQL Control-of-Flow language

- Control-of-flow language determines the execution flow of Transact-SQL statements, statement blocks, user-defined functions, and stored procedures.

Control-Of-Flow Language Keywords
RETURN
THROW
TRY...CATCH
WAITFOR
WHILE
BEGIN....END
BREAK
CONTINUE
GOTO label
IF...ELSE

Keywords

© Aptech Ltd.

Session 13 / 17

Program Flow Statements 2-2

➤ BEGIN....END

- The BEGIN...END statements with a series of Transact-SQL statements so that a group of Transact-SQL statements is executed.

➤ IF...ELSE

- The BEGIN...END statements with a series of Transact-SQL statements so that a group of Transact-SQL statements is executed.

➤ WHILE

- The WHILE statement specifies a condition for the repetitive execution of the statement block.

© Aptech Ltd.

Session 13 / 18

Instructions to the Trainer(s):

- Using Slides 17 and 18, explain program flow statements.
- There are different types of program flow statements and functions supported by Transact-SQL. Some of these are as follows:
 - **Transact-SQL Control-of-Flow language**
 - Control-of-flow language determines execution flow of Transact-SQL statements, statement blocks, user-defined functions, and stored procedures.
 - By default, Transact-SQL statements are executed sequentially, in the order they occur.

- Control-of-flow language elements allow statements to be executed in a particular order, to be related to each other, and made interdependent using constructs similar to programming languages.
- Using Slide 18, explain following statements:
 - **BEGIN....END**
 - The BEGIN...END statements is used to enclose a series of Transact-SQL statements as a group of Transact-SQL statements to be executed.
 - **IF...ELSE**
 - The IF...ELSE statement enforces a condition on the execution of a Transact-SQL statement.
 - The Transact-SQL statement is followed with IF keyword and the condition executes only if the condition is satisfied and returns TRUE.
 - The ELSE keyword is an optional Transact-SQL statement that executes only when the IF condition is not satisfied and returns FALSE.
 - **WHILE**
 - The WHILE statement specifies a condition for the repetitive execution of the statement block.
 - The statements are executed repetitively as long as specified condition is true.
 - The execution of statements in WHILE loop can be controlled by using BREAK and CONTINUE keywords.

Deterministic Built-in Functions	Non-Deterministic Built-in Functions
POWER	@@TOTAL_WRITE
ROUND	CURRENT_TIMESTAMP
RADIANS	GETDATE
EXP	GETUTCDATE
FLOOR	GET_TRANSMISSION_STATUS
SQUARE	NEWID
SQRT	NEWSEQUENTIALID
LOG	@@CONNECTIONS
YEAR	@@CPU_BUSY
ABS	@@DBTS
ASIN	@@IDLE
ACOS	@@IOBUSY
SIGN	@@PACK_RECEIVED
SIN	@@PACK_SENT

Deterministic and Non-deterministic Built-in Functions

© Aptech Ltd.

Session 13 / 19

Instructions to the Trainer(s):

- Using Slide 19, explain Transact-SQL functions.
- The Transact-SQL functions that are commonly used are as follows:
 - **Deterministic and non-deterministic functions**
 - User-defined functions possess properties that define the capability of the SQL Server Database Engine.
 - The Database engine is used to index the result of a function through either computed columns that the function calls or the indexed views that reference the functions.
 - One such property is the determinism of a function.
- Deterministic functions return the same result every time they are called with a definite set of input values and specify the same state of the database.
- Non-deterministic functions return different results every time they are called with specified set of input values even though the database that is accessed remains the same.

Transact-SQL Functions 2-3

Some functions are not always deterministic, but you can use them in indexed views if they are given in a deterministic manner.

Function	Description
CONVERT	Is deterministic only if one of these conditions exists: <ul style="list-style-type: none"> Has an sql_variant source type. Has an sql_variant target type and source type is non-deterministic. Has its source or target type as smalldatetime or datetime, has the other source or target type as a character string, and has a non-deterministic style specified. The style parameter must be a constant to be deterministic.
CAST	Is deterministic only if it is used with smalldatetime, sql_variant, or datetime.
ISDATE	Is deterministic unless used with the CONVERT function, the CONVERT style parameter is specified, and style is not equal to 0, 100, 9, or 109.
CHECKSUM	Is deterministic, with the exception of CHECKSUM(*)

Deterministic Functions

Instructions to the Trainer(s):

- Using Slide 20, explain different functions in Transact-SQL.
- Some functions are not always deterministic, but you can use them in indexed views if they are given in deterministic manner.
- The Deterministic functions are discussed as follows:
 - CONVERT
 - CAST
 - ISDATE
 - CHECKSUM

Transact-SQL Functions 3-3

Calling Extended Stored Procedures from Functions

- They are non-deterministic because the extended stored procedures may result in side effects on the database.

Scalar-valued Functions

- A Scalar-valued Function (SVF) always returns an int, bit, or string value.

Table-valued Functions

- Table-valued functions are user-defined functions that return a table.

© Aptech Ltd. Session 13/ 21

Instructions to the Trainer(s):

- Using Slide 21, explain the functions in T-SQL.
- **Calling Extended Stored Procedures from Functions:**
 - Functions calling extended stored procedures are non-deterministic because the extended stored procedures may result in side effects on the database.
 - Changes made to the global state of a database such as a change to an external resource, or updates to a table, file, or a network are called side effects.
 - For example, sending an e-mail or deleting a file can cause side effects. While executing an extended stored procedure from a user-defined function, the user cannot assure that it will return a consistent resultset.
 - Therefore, the user-defined functions that create side effects on the database are not recommended.
- **Scalar-Valued functions:**
 - A Scalar-Valued Function (SVF) always returns an int, bit, or string value.
 - The data type returned from and the input parameters of SVF can be of any data type except text, ntext, image, cursor, and timestamp.
 - An inline scalar function has a single statement and no function body.
 - A multi-statement scalar function encloses the function body in a BEGIN...END block.
- **Table-Valued functions:**
 - Table-valued functions are user-defined functions that return a table. Similar to an inline scalar function, an inline table-valued function has a single statement and no function body.

Altering User-defined Functions I-2

- Users can modify the user-defined functions in SQL Server 2019 by using the Transact-SQL or SSMS.
 - Changing the user-defined functions does not modify the functions' permissions, nor will it affect any stored procedures, triggers, or functions.

Limitations and Restrictions

The ALTER FUNCTION does not allow users to perform the following actions:

- Modify a scalar-valued function to a table-valued function.
- Modify an inline function to a multi-statement function.
- Modify a Transact-SQL to a CLR function.

Permissions

The ALTER permission is required on the schema or the function. If the function specifies a user-defined type, then it requires the EXECUTE permission on the type.

Instructions to the Trainer(s):

- Using Slide 22, explain how to alter user-defined functions.
- Users can modify the user-defined functions in SQL Server 2012 by using the Transact-SQL or SSMS.
- Changing the user-defined functions does not modify the functions permissions, nor will it affect any stored procedures, triggers, or functions.
- The ALTER FUNCTION does not allow the users to perform following actions:
 - Modify a scalar-valued function to a table-valued function.
 - Modify an inline function to a multi-statement function.
 - Modify a Transact-SQL to a CLR function.

Altering User-defined Functions 2-2

Modifying a User-defined function using SSMS

1. Click the plus (+) symbol beside the database that contains the function to be modified.
2. Click the plus (+) symbol next to the Programmability folder.
3. Click the plus (+) symbol next to the folder, which contains the function to be modified. There are four folder types as follows:
 - Table-valued Functions
 - Scalar-valued Functions
 - Aggregate Functions
 - System Functions
4. Right-click the function to be modified and then, select **Modify**. The code for the function appears in a query editor window.
5. In the query editor window, make the required changes to the `ALTER FUNCTION` statement body.
6. Click **Execute** on the toolbar to execute the `ALTER FUNCTION` statement.

Modifying a User-defined function using Transact-SQL

- o In the **Object Explorer**, connect to the Database Engine instance.
- o On the **Standard bar**, click **New Query**.
- o Type the `ALTER FUNCTION` code in the **Query Editor**.
- o Click **Execute** on the toolbar to execute the `ALTER FUNCTION` statement.

© Aptech Ltd. Session 13/ 23

Instructions to the Trainer(s):

- Using Slide 23, explain how to alter user-defined functions.
- ALTER permission is required on the schema or the function. If the function specifies a user-defined type, then it requires the EXECUTE permission on the type.
- **Modifying a User-defined function using SSMS:**
Click plus sign next to the folder that contains the function you wish to modify:
 - Table-valued Function.
 - Scalar-valued Function.
 - Aggregate Function.
- **Modifying a User-defined function using Transact-SQL:**
 - In the Object Explorer, connect to the Database Engine instance.
 - On the Standard bar, click New Query.
 - Type the `ALTER Function` code in the Query Editor.
 - Click Execute on the toolbar to execute the `ALTER FUNCTION` statement.

Creation of Windows with OVER

- In Transact-SQL, the OVER clause is used to define a window within a query resultset.
- Using windows and OVER clause with functions provides several advantages.

For instance,

They help to calculate aggregated values. They also enable row numbers in a resultset to be generated easily.

© Aptech Ltd.

Session 13 / 24

Instructions to the Trainer(s):

- Using Slide 24, explain how to create windows with OVER.
- A window function is a function that applies to a collection of rows. The word 'window' is used to refer to the collection of rows that the function works on.
- In Transact-SQL, the OVER clause is used to define a window within a query resultset.
- Using windows and the OVER clause with functions provides several advantages.
- For instance, they help to calculate aggregated values. They also enable row numbers in a resultset to be generated easily.

Windowing Components

- A window function is a function that applies to a collection of rows.
- In Transact-SQL, the OVER clause is used to define a window within a query resultset.
- Using windows and the OVER clause with functions provides several advantages.

Instructions to the Trainer(s):

- Using Slide 25, explain Windowing components.
- A window function is a function that applies to a collection of rows.
- The three core components of creating windows with the OVER clause, are as follows:
 - Partitioning
 - Ordering
 - Framing

Windowing Components 1-3

Three core components of creating windows with the OVER clause are as follows:

Partitioning

- Partitioning is a feature that limits the window of the recent calculation to only those rows from the resultset that contains the same values in the partition columns as in the existing row.
- It uses the PARTITION BY clause.

SalesOrderID	ProductID	OrderQty	Total	MaxOrderQty
1	43659	776	1	3
2	43659	773	2	3
3	43661	776	4	6
4	43661	773	2	6
5	43664	773	1	1
6	43665	773	1	2
7	43665	776	1	2
8	43667	773	1	1
9	43670	776	1	3
10	43670	773	2	3
11	43672	776	2	2
12	43676	776	2	2
13	43683	773	2	4
14	43683	776	2	4
15	43693	773	1	1
16	43694	776	3	5
17	43694	773	2	5

Partitioning with OVER Clause

© Aptech Ltd.

Session 13 / 26

Instructions to the Trainer(s):

- Using Slide 26, explain partitioning component.
- **Partitioning:**
 - Partitioning is a feature that limits the window of the recent calculation to only those rows from the resultset that contains the same values in the partition columns as in the existing row.
 - It uses the PARTITION BY clause.

Windowing Components 2-3

Ordering

- The ordering element defines the ordering for calculation in the partition.
- The ordering element has different meaning to some extent for different function categories. With ranking functions, ordering is spontaneous.

CustomerID	StoreID	Rnk_All	Rnk_Cust
701	844	813	1
700	1030	633	2
699	842	815	3
698	640	1009	4
697	1032	631	5
696	840	817	6
695	638	1011	7
694	1034	629	8
693	834	819	9
692	802	855	10
691	1036	627	11
690	836	821	12
689	1402	278	13
688	1038	625	14
687	834	823	15
686	1400	279	16
685	1040	623	17

✔ Query executed successfully.

TerritoryID	Name	SalesYTD	Rnk_One	Rnk_Two
1	Northwest	7087186.7882	2	1
2	Northeast	2402176.8476	10	1
3	Central	3072175.118	8	1
4	Southwest	10510853.8739	1	1
5	Southeast	2538667.2516	9	1
6	Canada	8771029.1376	3	1
7	France	4772398.3078	6	1
8	Germany	3089202.3478	7	1
9	Australia	5977814.9154	4	1
10	United Kingdom	5012905.3656	5	1

Partitioning and Ranking

© Aptech Ltd.

Session 13 / 27

Instructions to the Trainer(s):

- Using Slide 27, explain Ordering component.
- **Ordering:**
 - The ordering element defines the ordering for calculation in the partition. In a standard SQL ordering element, all functions are supported.
 - Earlier, SQL Server had no support for the ordering elements with aggregate functions as it only supported partitioning.
 - The ordering element has different meaning to some extent for different function categories. With ranking functions, ordering is spontaneous.

Windowing Components 3-3

Framing

The screenshot shows a database query results page. At the top, there is a title bar with the text "Windowing Components 3-3". Below the title bar, the word "Framing" is displayed. In the center of the page, there is a table with four columns: "ProductID", "Shelf", "Quantity", and "RunQty". The table contains 17 rows of data. A blue callout box is overlaid on the left side of the table, containing the following text:

- Framing is a feature that enables you to specify a further division of rows within a window partition.
- This is done by assigning upper and lower boundaries for the window frame that presents rows to the window function.

At the bottom of the table, a green message box displays the text "Query executed successfully." followed by a checkmark icon. The entire screenshot has a watermark reading "For Practice Only" diagonally across it.

ProductID	Shelf	Quantity	RunQty
1	A	408	408
2	B	324	732
3	A	353	1085
4	A	427	427
5	B	318	745
6	A	364	1109
7	A	585	585
8	B	443	1028
9	A	324	1352
10	A	512	512
11	B	422	934
12	A	388	1322
13	A	532	532
14	B	388	920
15	B	441	1361
16	C	283	283
17	A	158	441

Framing

© Aptech Ltd. Session 13 / 28

Instructions to the Trainer(s):

- Using Slide 28, explain framing component.
- **Framing:**
 - Framing is a feature that enables you to specify a further division of rows within a window partition.
 - This is done by assigning upper and lower boundaries for the window frame that presents rows to the window function.
 - In simple terms, a frame is similar to a moving window over the data that starts and ends at specified positions.
 - Window frames can be defined using the ROW or RANGE subclauses and providing starting and ending boundaries.

Window Functions 1-5

Some of the different types of window functions are as follows:

Ranking functions

- These functions return a rank value for each row in a partition.
- Based on the function that is used, many rows will return the same value as the other rows.
- Ranking functions are non-deterministic.

Ranking Functions	Description
NTILE	Spreads rows in an ordered partition into a given number of groups, beginning at 1. For each row, the function returns the number of the group to which the row belongs.
ROW NUMBER	Retrieves the sequential number of a row in a partition of a resultset, starting at 1 for the first row in each partition.
DENSE RANK	Returns the rank of rows within the partition of a resultset, without any gaps in the ranking. The rank of a row is one plus the number of distinct ranks that come before the row in question.

Ranking Functions

© Aptech Ltd.

Session 13 / 30

Window Functions 2-5

OFFSET functions

Different types of offset functions are as follows:

➤ SWITCHOFFSET

This function returns a DATETIMEOFFSET value that is modified from the stored time zone offset to a specific new time zone offset.

(No column name)	
1	1998-09-20 04:45:50.7134500 -08:00
ColDatetimeoffset	
1	1998-09-20 07:45:50.7134500 -05:00

Use of SWITCHOFFSET Function

© Aptech Ltd.

Session 13 / 30

Window Functions 3-5

➤ DATETIMEOFFSETFROMPARTS

This function returns a datetimeoffset value for the specified date and time with specified precision and offset.

Results		Messages
Result		
1	2010-12-31 14:23:23.0000000 +12:00	

Use of DATETIMEOFFSETFROMPARTS Function

➤ SYSDATETIMEOFFSET

These functions returns datetimeoffset(7) value which contains the date and time of the computer on which the instance of SQL Server is running.

SYSdatetime	SYSDATETIMEOFFSET	SYSUTCDATETIME
2020-10-27 21:37:55.1078981	2020-10-27 21:37:55.1078981+05:30	2020-10-27 16:07:55.1078981

Use of SYSDATETIMEOFFSET Function

© Aptech Ltd.

Session 13 / 30

Window Functions 4-5

Analytic Functions

SQL Server 2019 supports several analytic functions. These functions compute aggregate value based on a group of rows. Analytic functions compute running totals, moving averages, or top-N results within a group.

Function	Description
LEAD	Provides access to data from a subsequent row in the same resultset without using a self-join.
LAST_VALUE	Retrieves the last value in an ordered set of values.
LAG	Provides access to data from a previous row in the same resultset without using a self-join.
FIRST_VALUE	Retrieves the first value in an ordered set of values.
CUME_DIST	Computes the cumulative distribution of a value in a group of values.
PERCENTILE_CONT	Computes a percentile based on a continuous distribution of the column value in SQL..
PERCENTILE_DISC	Calculates a particular percentile for sorted values in an entire rowset or within distinct partitions of a rowset.

Analytic Functions

Window Functions 5-5

	Name	ListPrice	LessExpensive
1	Patch Kit/8 Patches	2.29	Patch Kit/8 Patches
2	Road Tire Tube	3.99	Patch Kit/8 Patches
3	Touring Tire Tube	4.99	Patch Kit/8 Patches
4	Mountain Tire Tube	4.99	Patch Kit/8 Patches
5	LL Road Tire	21.49	Patch Kit/8 Patches
6	ML Road Tire	24.99	Patch Kit/8 Patches
7	LL Mountain Tire	24.99	Patch Kit/8 Patches
8	Touring Tire	28.99	Patch Kit/8 Patches
9	ML Mountain Tire	29.99	Patch Kit/8 Patches
10	HL Road Tire	32.60	Patch Kit/8 Patches
11	HL Mountain Tire	35.00	Patch Kit/8 Patches

First Value() Function

Instructions to the Trainer(s):

- Using Slides 29 to 33, explain various window functions.
- Using Slide 29, introduce ranking functions. These functions return a rank value for each row in a partition. Based on the function that is used, many rows will return the same value as the other rows.
- Ranking functions are non-deterministic.
- Ranking functions comprise the following:
 - **NTILE:** Spreads rows in an ordered partition into a given number of groups.
 - **ROW NUMBER:** It assigns the sequential rank number to each unique record.
 - **DENSE RANK:** It assigns the rank number to each row in a partition. It does not skip the number for similar values.
- Using Slide 30, explain the OFFSET Functions.

- The SWITCHOFFSET() function returns a DATETIMEOFFSET changed from the stored time zone offset to a new time zone offset.
- Using Slide 31, explain following functions:
- **DATETIMEOFFSETFROMPARTS:**
 - DATETIMEOFFSETFROMPARTS function returns a datetimeoffset value for the specified date and time with specified precision and offset.
- **SYSDATETIMEOFFSET:**
 - These functions return datetimeoffset(7) value which contains the date and time of the computer on which the instance of SQL Server is running.
- Using Slide 32, explain analytic functions.
- SQL Server supports several analytic functions. These functions compute aggregate value based on a group of rows.
- Analytic functions compute running totals, moving averages, or top-N results within a group.
- SQL Server supports these analytic functions:
 - CUME_DIST (Transact-SQL): Computes the cumulative distribution of a value in a group of values.
 - FIRST_VALUE (Transact-SQL): Retrieves the first value in an ordered set of values.
 - LAG (Transact-SQL): Provides access to data from a previous row in the same resultset without using a self-join.
 - LAST_VALUE (Transact-SQL): Retrieves the last value in an ordered set of values.
 - LEAD (Transact-SQL): Provides access to data from a subsequent row in the same resultset without using a self-join.
 - PERCENTILE_CONT (Transact-SQL): Computes a percentile based on a continuous distribution of the column value in SQL.
 - PERCENTILE_DISC (Transact-SQL): Calculates a particular percentile for stored value in an entire rowset or within distinct partitions of rowset.
- Using Slide 33, explain the Windowing functions in T-SQL.
- Figure in Slide 33 displays the first value functions in windowing functions in T-SQL.
- The FIRST_VALUE() function is a window function that returns the first value in an ordered partition of a result set.
- Given an ordered set of rows, FIRST_VALUE returns the value of the specified expression with respect to the first row in the window frame.

Summary

- Transact-SQL provides basic programming elements such as variables, control-of-flow elements, conditional, and loop constructs.
- A batch is a collection of one or more Transact-SQL statements that are sent as one unit from an application to the server.
- Variables allow users to store data for using as input in other Transact-SQL statements.
- Synonyms provide a way to have an alias for a database object that may exist on a remote or local server.
- Deterministic functions always return same result each time when they are called with a definite set of input values.
- Non-deterministic functions return different results every time they are called with specified set of input values even though the database that is accessed remains the same.
- A window function is a function that applies to a collection of rows.

Instructions to the Trainer(s):

- Show students Slide 34.
- Summarize the session by reading out each point on the slide.

Session 14: Transactions

14.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. Prepare a question or two that will be a key point to relate the current session objectives.

14.1.1 Teaching Skills

To teach this session, you should be well versed with transactions, types of transactions, the procedure to implement these transactions, and how to manage transactions.

You should teach the concepts in the theory class using the images provided. For teaching in the class, you are expected to use slides and LCD projectors.

In-Class Activities

Follow the order given here during In-Class activities.

Slide 2

Objectives

- Define and describe transactions
- Explain the procedure to implement transactions
- Explain the steps to mark a transaction
- Distinguish between implicit and explicit transactions
- Explain isolation levels
- Explain the scope and different types of locks
- Explain transaction management

© Aptech Ltd. Session 14/2

Instructions to the Trainer(s):

Give students a brief overview of the current session through the session objectives listed in Slide 2.

14.2 In-Class Explanations

Slide 3

Introduction

- ② A transaction is a single unit of work.
- ② A transaction is successful only when all data modifications that are made in a transaction are committed and are saved in the database permanently.
- ② If the transaction is rolled back or cancelled, then it means that the transaction has encountered errors and there are no changes made to the contents of the database.
- ② Hence, a transaction can be either committed or rolled back.



© Aptech Ltd. Session 14 / 3

Instructions to the Trainer(s):

- Using Slide 3, explain the concept of transaction.
- A transaction is a single unit of work.
- A transaction is successful only when all data modifications that are made in a transaction are committed and are saved in the database permanently.
- If the transaction is rolled back or cancelled, then it means that the transaction has encountered errors and there are no changes made to the contents of the database.
- Hence, a transaction can be either committed or rolled back.

For more information, refer to:

<https://www.sqlshack.com/transactions-in-sql-server-for-beginners/>

<https://sqldatabasetutorials.com/sql-db/database-transactions/>

<https://simplesqltutorials.com/sql-server-transactions-for-beginners/>

<https://www.c-sharpcorner.com/UploadFile/84c85b/understanding-transactions-in-sql-server/>

Need for Transactions 1-4

There are many circumstances where users are required to make many changes to the data in more than one database tables.

In many cases, the data will be inconsistent that executes the individual commands.

Suppose if the first statement executes correctly, but other statements fail then the data remains in an incorrect state.

For example,

A good scenario will be the funds transfer activity in a banking system. The transfer of funds will need an INSERT and two UPDATE statements.

Instructions to the Trainer(s):

- Using Slide 4, explain the necessity for transactions.
- There are many circumstances where the users must make many changes to the data in more than one database tables.
- In many cases, the data will be inconsistent that executes the individual commands.
- Suppose if the first statement executes correctly but the other statements fail, then the data remains in an incorrect state.
- For example, a good scenario will be the funds transfer activity in a banking system. The transfer of funds will require an INSERT and two UPDATE statements. First, the user has to increase the balance of the destination account and then, decrease the balance of the source account.
- The user has to check that the transactions are committed and whether same changes are made to the source account and the destination account.

Need for Transactions 2-4

Defining Transactions

➤ A logical unit of work must exhibit four properties, called the Atomicity, Consistency, Isolation, and Durability (ACID) properties, to qualify as a transaction.

➤ Atomicity: If the transaction has many operations then, all should be committed. If any of the operation in the group fails then, it should be rolled back.
➤ Consistency: The sequence of operations must be consistent.
➤ Isolation: Operations that are performed must be isolated from other operations on the same server or on the same database.
➤ Durability: The operations that are performed on the database must be saved and stored in the database permanently.

© Aptech Ltd. Session 14/5

Instructions to the Trainer(s):

- Using Slide 5, define transactions.
- A logical unit of work must exhibit four properties, called the Atomicity, Consistency, Isolation, and Durability (ACID) properties, to qualify as a transaction.
 - **Atomicity:** If the transaction has many operations, then all should be committed. If any of the operation in the group fails, then it should be rolled back.
 - **Consistency:** The sequence of operations must be consistent.
 - **Isolation:** The operations that are performed must be isolated from the other operations on the same server or on the same database.
 - **Durability:** The operations that are performed on the database must be saved and stored in the database permanently.

Need for Transactions 3-4

Implementing Transactions

- Autocommit Transactions: In this mode, one does not need to write any specific statements to start and end the transactions. It is the default mode for SQL Server Database Engine.
- Explicit Transactions: Each transaction explicitly starts with the BEGIN TRANSACTION statement and ends with a ROLLBACK or COMMIT transaction.
- Implicit Transactions: A new transaction is automatically started when the earlier transaction completes and every transaction is explicitly completed.
- Batch-scoped Transactions: These transactions are related to Multiple Active Result Sets (MARS).
- Distributed Transactions: It spans two or more servers known as resource managers.

© Aptech Ltd. Session 14 / 6

Instructions to the Trainer(s):

- Using Slide 6, discuss how transactions are implemented.
- SQL Server supports transactions in several modes are shown on slide 6. Some of these modes are as follows:
 - **Autocommit Transactions:** Every single-line statement is automatically committed as soon as it completes. In this mode, one does not have to write any specific statements to start and end the transactions. It is the default mode for SQL Server Database Engine.
 - **Transactions:** Every transaction explicitly starts with the BEGIN TRANSACTION statement and ends with a ROLLBACK or COMMIT transaction.
 - **Implicit Transactions:** A new transaction is automatically started when the earlier transaction completes and every transaction is explicitly completed by using the ROLLBACK or COMMIT statement.
 - **Batch-scoped Transactions:** These transactions are related to Multiple Active Result Sets (MARS). Any implicit or explicit transaction that starts in a MARS session is a batch-scoped transaction. A batch-scoped transaction that is rolled back when a batch completes automatically is rolled back by SQL Server.
 - **Distributed Transactions:** It spans two or more servers known as resource managers.

Need for Transactions 4-4

Transactions Extending Batches

The transaction statements identify the block of code that should either fail or succeed and provide the facility where the database engine can undo or roll back the operations.

Errors that encounter during the execution of simple batch have the possibility of partial success, which is not a desired result.

This also led to inconsistencies in the tables and databases.

Users can add error-handling code to roll back the transaction in case of errors.

The error-handling code will undo the partial changes that were made before the error had occurred.

Instructions to the Trainer(s):

- Using Slide 7, explain the necessity of transactions.
- The transaction statements identify the block of code that should either fail or succeed and provide the facility where the database engine can undo or roll back the operations.
- The errors that encounter during the execution of simple batch have the possibility of partial success, which is not a desired result.
- This also led to inconsistencies in the tables and databases. To overcome this, users can add code to identify the batch as a transaction and place the batch between the BEGIN TRANSACTION and COMMIT TRANSACTION.
- Users can add error-handling code to roll back the transaction in case of errors.
- The error-handling code will undo the partial changes that were made before the error had occurred.
- This way, inconsistencies in the tables and databases can be prevented.

Controlling Transactions

- Transactions can be controlled through applications by specifying the beginning and ending of a transaction.
- Transactions are managed at the connection level, by default.
- When a transaction is started on a connection, all Transact-SQL statements are executed on the same connection and are a part of the connection until the transaction ends.

© Aptech Ltd. Session 14/ 8

Instructions to the Trainer(s):

- Using Slide 8, explain how to control a transaction.
- Transactions can be controlled through applications by specifying the beginning and ending of a transaction.
- This is done by using the database API functions or Transact-SQL statements.
- Transactions are managed at the connection level, by default.
- When a transaction is started on a connection, all Transact-SQL statements are executed on the same connection and are a part of the connection until the transaction ends.

Starting and Ending Transactions Using Transact-SQL 1-3

BEGIN TRANSACTION

- The BEGIN TRANSACTION statement marks the beginning point of an explicit or local transaction.

COMMIT TRANSACTION

- The COMMIT TRANSACTION statement marks an end of a successful implicit or explicit transaction.

COMMIT WORK

- The COMMIT WORK statement marks the end of a transaction.

© Aptech Ltd. Session 14 / 9

Instructions to the Trainer(s):

- Using Slide 9, explain how to start and end a session using Transact-SQL.
- One of the ways, users can start and end transactions by using Transact-SQL statements. Users can start a transaction in SQL Server in the implicit or explicit modes.
- Explicit transaction mode starts a transaction by using a BEGIN TRANSACTION statement.
- Users can end a transaction using the ROLLBACK or COMMIT statements.
- Following are different types of transactions:
 - **BEGIN TRANSACTION:** The BEGIN TRANSACTION statement marks the beginning point of an explicit or local transaction.
 - **COMMIT TRANSACTION:** The COMMIT TRANSACTION statement marks an end of a successful implicit or explicit transaction.
 - **COMMIT WORK:** The COMMIT WORK statement marks the end of a transaction.
- COMMIT TRANSACTION and COMMIT WORK are identical except for the fact that COMMIT TRANSACTION accepts a user-defined transaction name.

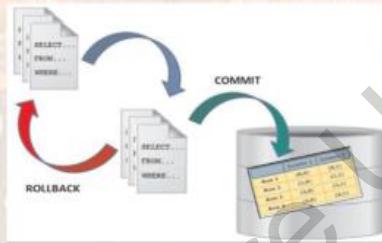
Starting and Ending Transactions Using Transact-SQL 2-3

ROLLBACK TRANSACTION

This transaction rolls back or cancels an implicit or explicit transaction to the starting point of the transaction or to a savepoint in a transaction.

It is used to delete all data modifications made from the beginning of the transaction or to a savepoint. It also releases the resources held by the transaction.

ROLLBACK WORK



© Aptech Ltd.

Session 14 / 10

Instructions to the Trainer(s):

- Using Slide 10, explain the ROLLBACK TRANSACTION and ROLLBACK WORK.
- **ROLLBACK TRANSACTION:** This transaction rolls back or cancels an implicit or explicit transaction to the starting point of the transaction or to a savepoint in a transaction. A savepoint is a mechanism to roll back some parts of transactions. The ROLLBACK TRANSACTION is used to delete all data modifications made from the beginning of the transaction or to a savepoint. It also releases the resources held by the transaction.
 - **ROLLBACK WORK:** This statement rolls back a user-specified transaction to the beginning of the transaction. The keyword WORK is optional and is rarely used.

Starting and Ending Transactions Using Transact-SQL 3-3

SAVE TRANSACTION

- The SAVE TRANSACTION statement sets a savepoint within a transaction.

© Aptech Ltd.

Session 14/ 11

Instructions to the Trainer(s):

- Using Slide 11, explain the Save transaction.
- The SAVE TRANSACTION statement sets a SAVEPOINT within a transaction.
- A SAVEPOINT transaction is created within a stored procedure. This will then be used to roll back only the changes made by the stored procedure if an active transaction has started before the stored procedure executes.

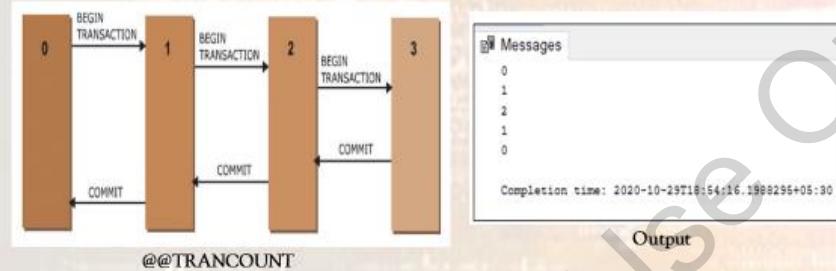
In-Class Question:

Question: Can a transaction be saved temporarily?

Answer: SAVEPOINT command is used to temporarily save a transaction so that you can rollback to that point whenever required.

The @@TRANCOUNT

- The @@TRANCOUNT system function returns a number of BEGIN TRANSACTION statements that occur in the current connection.



© Aptech Ltd.

Session 14 / 12

Instructions to the Trainer(s):

- Using Slide 12, explain @@TRANCOUNT.
- The @@TRANCOUNT system function returns a number of BEGIN TRANSACTION statements that occur in the current connection.
- The code snippet on Slide 12 which shows the effect that nested BEGIN and COMMIT statements have on @@TRANCOUNT variable.
- This code snippet displays the number of times the BEGIN TRAN and COMMIT statement execute in the current connection.

Marking a Transaction

Marking a transaction is useful only when the user is willing to lose recently committed transactions or is testing related databases.

Marking related transactions on a routine basis in every single related database creates a sequence of common recovery points in a database.

Concerns for Using Marked Transactions

- As the transaction mark consume log space, use them only for transactions that play an important role in the database recovery strategy.
- When the marked transaction is committed, then a row is inserted in the logmarkhistory table in msdb.
- If a marked transaction spans over multiple databases on different servers or on the same database server, the marks must be logged in the records of all affected databases.

Instructions to the Trainer(s):

- Using Slide 13, discuss how marking a transaction can be done.
- Users can use transaction marks to recover the related updates made to two or more related databases. Though this recovery loses every transaction that is committed after the mark was used as the recovery point.
- Marking a transaction is useful only when the user is willing to lose recently committed transactions or is testing related databases.
- Marking related transactions on a routine basis in every single related database creates a sequence of common recovery points in a database.
- The transaction marks are incorporated in log backups and are also recorded in the transaction log.
- In case of any disaster, user can restore each of the databases to the same transaction mark in order to recover them to a consistent point.
- Consider the following situations before inserting the named marks in the transaction:
 - As the transaction mark consume log space, use them only for transactions that play an important role in the database recovery strategy.
 - When the marked transaction is committed, then a row is inserted in the logmarkhistory table in msdb.
 - If a marked transaction spans over multiple databases on different servers or on the same database server, the marks must be logged in the records of all affected databases.

In-Class Question:

Question: What is a transaction statement in SQL?

Answer: A transaction is a logical unit of work that contains one or more SQL statements. A transaction begins with the first executable SQL statement. A transaction ends when it is committed or rolled back, either explicitly with a COMMIT or ROLLBACK statement or implicitly when a DDL statement is issued.

Slide 14

Create Marked Transactions

For creating a marked transaction, users can use the BEGIN TRANSACTION statement and the WITH MARK [description] clause.

The transaction log records the mark description, name, user, database, datetime information, and the Log Sequence Number (LSN).

Steps to create a marked transaction in a set of databases:

- Name the transaction in the BEGIN TRAN statement and use the WITH MARK clause.
- Execute an update against all of the databases in the set.

© Aptech Ltd. Session 14 / 14

Instructions to the Trainer(s):

- Using Slide 14, explain how marked transactions can be created.
- The mark for a specific transaction is inserted into transaction logs only on the server instance where the BEGIN TRANSACTION statement and the WITH MARK statement is executed.
- The transaction log records the mark description, name, user, database, datetime information, and the Log Sequence Number (LSN).
- Steps for creating marked transactions:
 - Name the transaction in the BEGIN TRAN statement and use WITH MARK clause.
 - Execute an update against all of the databases in the set.

Differences Between Implicit and Explicit Transactions	
Implicit	Explicit
These transactions are maintained by SQL Server for each and every DML and DDL statements.	These transactions are defined by programmers.
These DML and DDL statements execute under the implicit transactions.	DML statements are included to execute as a unit.
SQL Server will roll back the entire statement.	SELECT Statements are not included as they do not modify data.

Implicit Vs. Explicit Transactions

© Aptech Ltd.

Session 14 / 15

Instructions to the Trainer(s):

- Using Slide 15, explain the difference between implicit and explicit transactions.
- **Implicit Transaction:**
 - Implicit Transaction is the auto commit.
 - These transactions are maintained by SQL Server for each and every DML and DDL statements.
 - These DML and DDL statements execute under the implicit transactions.
- **Explicit Transaction:**
 - Explicit Transaction has the beginning, ending, and rollback of transactions.
 - These transactions are defined by the programmers.
 - DML statements are included to execute a unit.

Isolation Levels 1-2

- Transactions identify the isolation levels that define the degree to which one transaction must be isolated from the data modifications or resource that are made by other transactions.
- Isolation levels are defined in terms of which the concurrency side effects such as dirty reads are allowed.

Transaction isolation levels control the following:

- When data is read, are there any locks taken and what types of locks are requested?
- How much amount of time the read locks are held?
- If a read operation that is referencing a row modified by some other transaction is:
 - Blocking until the exclusive lock on the row is free.
 - Retrieving the committed version of the row that exists at the time when the transaction or statement started.
 - Reading the uncommitted data modification.

A transaction acquires an exclusive lock every time on each data that it modifies. Then, it holds that lock until the transaction is completed, irrespective of the isolation level that is set for that transaction.

Instructions to the Trainer(s):

- Using Slide 16, explain about isolation levels.
- Transactions identify the isolation levels that define the degree to which one transaction must be isolated from the data modifications or resource that are made by the other transactions.
- Isolation levels are defined in terms of which the concurrency side effects such as dirty reads are allowed.
- When one transaction changes a value and a second transaction reads the same value before the original change has been committed or rolled back, it is called as a dirty read.
- Transaction isolation levels control the following:
 - When data is read, are there any locks taken and what types of locks are requested?
 - How much amount of time are the read locks held?
 - If a read operation that is referencing a row modified by some other transaction is:
 - Blocking until the exclusive lock on the row is free.
 - Retrieving the committed version of the row that exists at the time when the transaction or statement started.
 - Reading the uncommitted data modification.

Isolation Level	Dirty Read	NonRepeatable Read
Read committed	No	Yes
Read uncommitted	Yes	No
Snapshot	No	No
Repeatable Read	No	No
Serializable	No	No

Instructions to the Trainer(s):

- Explain different levels in isolation listed on Slide 17.
- Transaction isolation levels mainly describe the protection levels from the special effects of changes made by other transactions for read operations.
- A lower isolation level increases the capability of several users to access data at the same time. However, it increases the number of concurrency effects such as dirty reads or lost updates that users might come across.
- On the other hand, a higher isolation level decreases the types of concurrency effects which user may encounter. This requires additional system resources and increases the chance of one transaction blocking another transaction.
- Different isolation levels are as follows:
 - Read committed
 - Read Uncommitted
 - Snapshot
 - Repeatable Read
 - Serializable

Scope and Different Types of Locks 1-5

The SQL Server Database Engine locks the resources that use different lock modes, which determine the resources that are accessible to concurrent transactions.

Lock Mode	Description
Update	Is used on resources that are to be updated.
Shared	Is used for read operations that do not change data such as <code>SELECT</code> statement.
Intent	Is used to establish a hierarchy of locks.
Exclusive	Is used for <code>INSERT</code> , <code>UPDATE</code> , or <code>DELETE</code> data-modification operations.
BULK UPDATE	Is used while copying bulk data into the table.
Schema	Is used when the operation is dependent on the table schema.

Lock Modes

Instructions to the Trainer(s):

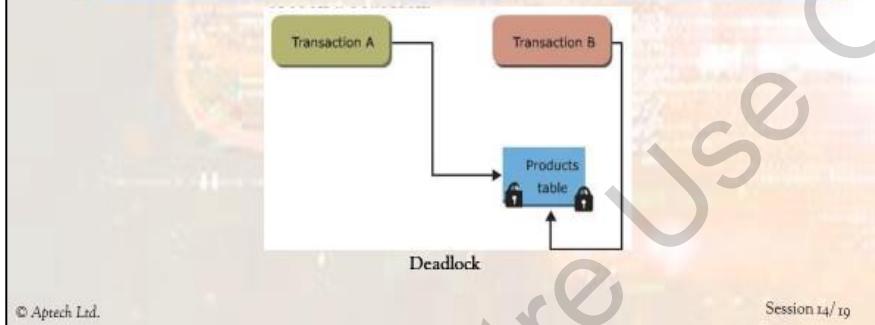
- Using Slide 18, explain the scope and different types of locks.
- The SQL Server Database Engine locks the resources that use different lock modes, which determine the resources that are accessible to concurrent transactions. Table on slide 18 lists the resource lock modes used by the database engine.
- Following are the types of locks:
 - **Update-** Used on resources that are to be updated.
 - **Shared-** Used for read operations that do not change data.
 - **Intent-** Used to establish hierarchy of locks.
 - **Exclusive-** Used for `INSERT`, `UPDATE`, or `DELETE` data.
 - **BULK UPDATE-** Used while copying bulk data in the table.
 - **Schema-** Used when the operation is dependent on the table schema.

Scope and Different Types of Locks 2-5

Different types of locks are as follows:

Update Locks

These locks avoid common forms of deadlock. In a serializable transaction, the transaction will read data, acquire a shared lock on the row or a page, and modify the data that requires lock conversion to an exclusive lock.



© Aptech Ltd.

Session 14 / 19

Instructions to the Trainer(s):

- Using Slide 19, explain the update lock.
- These locks avoid common forms of deadlock. In a serializable transaction, the transaction will read data, acquire a shared lock on the row or a page, and modify the data that requires lock conversion to an exclusive lock.
- When two transactions acquire a shared lock on a resource and try to update data simultaneously, the same transaction attempts the lock conversion to an exclusive lock.
- The shared mode to exclusive lock conversion should wait as the exclusive lock for one transaction is not compatible with shared mode lock of the other transaction a lock wait occurs.
- Similarly, the second transaction tries to acquire an exclusive lock for update.
- As both the transactions are converting to exclusive locks and each waits for the other transaction to release its shared lock mode, a deadlock occurs.

Scope and Different Types of Locks 3-5

Shared Locks

- These locks allow parallel transactions to read a resource under pessimistic concurrency control.
- Shared locks are released on a resource once the read operation is completed, except the isolation level is set to repeatable read or higher.

Exclusive Locks

- These locks prevent access to resources by concurrent transactions.
- By using an exclusive lock, no other transaction can change data and read operations take place only through the read uncommitted isolation level or NOLOCK hint.
- DML statements such as INSERT, DELETE, and UPDATE combine modification and read operations.

Instructions to the Trainer(s):

- Using Slide 20, explain the shared lock and exclusive locks.
 - **Shared Locks:** These locks allow parallel transactions to read a resource under pessimistic concurrency control. Transactions can change the data while shared locks exist on the resource. Shared locks are released on a resource once the read operation is completed, except the isolation level is set to repeatable read or higher.
 - **Exclusive Locks:** These locks prevent access to resources by concurrent transactions. By using an exclusive lock, no other transaction can change data and read operations take place only through the read uncommitted isolation level or NOLOCK hint. DML statements such as INSERT, DELETE, and UPDATE combine modification and read operations. These statements first perform a read operation to get data before modifying the statements. DML statements usually request both exclusive and shared locks. For example, if the user wants to use an UPDATE statement that modifies the row in one table that is dependent on a join with another table. Therefore, the update statements request shared lock on the rows that reads from the join table and request exclusive locks on the modified rows.

Scope and Different Types of Locks 4-5

Intent Locks

- To prevent other transactions from changing the higher-level resource in a way that will invalidate the lock at the lower-level.
- To improve the efficiency of the Database Engine for identifying the lock conflicts those are at the higher level of granularity.

Lock Mode	Description
Intent shared (IS)	Protects the requested shared lock on some resources that are lower in the hierarchy.
Intent exclusive (IX)	Protects the requested exclusive lock on some resources lower in the hierarchy. IX is a superset of IS, that protects requesting shared locks on lower level resources.
Shared with Intent Exclusive (SIX)	Protects the requested shared lock on all resources lower in the hierarchy and intent exclusive locks on some of the lower level resources. Concurrent IS locks are allowed at the top-level resource.
Intent Update (IU)	Protects the requested update locks on all resources lower in the hierarchy. IU locks are used only on page resources. IU locks are converted to IX locks if an update operation takes place.
Shared intent update (SIU)	Provides combination of S and IU locks, as a result of acquiring these locks separately and simultaneously holding both locks.
Update intent exclusive (UIX)	Provides combination of U and IX locks, as a result of acquiring these locks separately and simultaneously holding both locks.

Instructions to the Trainer(s):

- Using Slide 21, explain Intent Lock concept.
- The database engine uses intent locks for protecting and places an exclusive or shared lock on the resource that is at a lower level in the lock hierarchy.
- The name 'intent lock' is given because these locks are acquired before a lock at the low level and hence, indicate intent to place locks at low level.
- An intent lock is useful for two purposes:
 - To prevent other transactions from changing the higher-level resource in a way that will invalidate the lock at the lower-level.
 - To improve the efficiency of the database engine for identifying the lock conflicts those are at the higher level of granularity.
- For example, a shared intent locks are requested at the table level before requesting the shared locks on rows or pages within the table. Setting the intent lock at the table level protects other transaction from subsequently acquiring an exclusive lock on the table containing pages. Intent locks also contain Intent Exclusive (IX), Intent Shared (IS), and Shared with Intent Exclusive (SIX). Table shown on slide 21 lists the lock modes in Intent locks.

Scope and Different Types of Locks 5-5

Bulk Update locks

Bulk update locks are used by the database engine. These locks are used when a large amount of data is copied into a table. These locks allow multiple threads to load bulk data continuously in the same table.

Schema Locks

Schema modification locks are used by Database Engine while performing a table DDL operation such as dropping a table or a column. Schema stability locks are used by the database engine while compiling and executing the queries.

Key-Range Locks

These types of locks protect a collection of rows that are implicitly present in a recordset. Key-range locks prevent phantom reads.

Instructions to the Trainer(s):

- Using Slide 22, explain the bulk, schema, and key-range locks.
 - **Bulk Update locks:** Bulk update locks are used by the database engine. These locks are used when a large amount of data is copied into a table (bulk copy operations) and either the table lock on bulk load option is set or the TABLOCK hint is specified using the sp_table option. These locks allow multiple threads to load bulk data continuously in the same table however, preventing other processes that are not bulk loading data from accessing the table.
 - **Schema Locks:** Schema modification locks are used by database engine while performing a table DDL operation such as dropping a table or a column. Schema locks prevent concurrent access to the table, which means a schema lock blocks all external operations until the lock releases. Some DML operations such as truncating a table use the Schema lock to prevent access to affected tables by concurrent operations. Schema stability locks are used by the database engine while compiling and executing the queries. These stability locks do not block any of the transaction locks including the exclusive locks. Hence, the transactions that include X locks on the table continue to execute during the query compilation. Though, the concurrent DML and DDL operations that acquire the Schema modification locks do not perform on the tables.
 - **Key-Range Locks:** These types of locks protect a collection of rows that are implicitly present in a recordset which is being read by a Transact-SQL statement while using the serializable transaction isolation level. Key-range locks prevent phantom reads. By protecting the range of keys between rows, they also prevent the phantom deletions or insertions in the recordset that accesses a transaction.

Transaction Management

- SQL Server implements several transaction isolation levels that ensure the ACID properties of these transactions.
- In reality, it means that it uses locks to facilitate transactional access to shared database resources and also, prevent the interference between the transactions.

Instructions to the Trainer(s):

- Using Slide 23, explain the transaction management.
- Every single statement that is executed is, by default, transactional in SQL Server.
- If a single SQL statement is issued, then an implicit transaction is started. It means the statement will start and complete implicitly.
- When the users use explicit BEGIN TRAN/COMMIT TRAN commands, they can group them together as an explicit transaction. These statements will either succeed or fail.
- SQL Server implements several transaction isolation levels that ensure the ACID properties of these transactions.
- In reality, it means that it uses locks to facilitate transactional access to shared database resources and also, prevent the interference between the transactions.

Transaction Log 1-2

- Transaction log is a critical component of the database and if a system failure occurs, the transaction log will be required to bring the database to a consistent data.
- The transaction log should not be moved or deleted until users understand the consequences of doing it.

Operations Supported By Transaction Log

Individual transactions recovery	Incomplete transactions recovery when SQL Server starts	Transactional replication support	Disaster recovery solutions and high availability support	Roll back a file, restored database, filegroup, or page forward to the point of failure
----------------------------------	---	-----------------------------------	---	---

© Aptech Ltd. Session 14 / 24

Instructions to the Trainer(s):

- Using Slide 24, explain the transaction log.
- Each SQL Server database has a transaction log, which records all transactions and the database modifications made by every transaction.
- The transaction log should be truncated regularly to keep it from filling up.
- Monitoring log size is important as there may be some factors that delay the log truncation.
- Transaction log is a critical component of the database and if a system failure occurs, the transaction log will be required to bring the database to a consistent data.
- The transaction log should not be moved or deleted until users understand the consequences of doing it.
- The operations supported by the transaction log are as follows:
 - Individual transactions recovery
 - Incomplete transactions recovery when SQL Server starts
 - Transactional replication support
 - Disaster recovery solutions and high availability support
 - Roll back a file, restored database, filegroup, or page forward to the point of failure

Log_reuse_wait	Log_reuse_wait desc	Description
0	NOTHING	Specifies that at present, there are more than one reusable virtual log file.
1	CHECKPOINT	Specifies that there is no checkpoint occurred since the last log truncation or the head of the log has not moved beyond a virtual log file.
2	LOG_BACKUP	Specifies a log backup that is required before the transaction log truncates.
3	ACTIVE_BACKUP_OR_RESTORE	Specifies that the data backup or a restore is in progress.
4	ACTIVE_TRANSACTION	Specifies that a transaction is active.
5	DATABASE_MIRRORING	Specifies that the database mirroring is paused or under high-performance mode, the mirror database is significantly behind the principal database.

Catalog View Columns

© Aptech Ltd.

Session 14 / 25

Instructions to the Trainer(s):

- Using Slide 25, explain different transaction logs to the students.
- **NOTHING:** Specifies that at present there are more than one reusable virtual log file.
- **CHECKPOINT:** Specifies that no checkpoint occurred since the last log transaction.
- **LOG_BACKUP:** Specifies a log backup that is required before the transaction log truncates.
- **ACTIVE_BACKUP_OR_RESTORE:** Specifies that the data backup or restore is in progress.
- **ACTIVE_TRANSACTION:** Specifies that a transaction is active.
- **DATABASE_MIRRORING:** Specifies that the database mirroring is paused or under high performance mode.

Summary

- A transaction is a sequence of operations that works as a single unit.
- BEGIN TRANSACTION marks the beginning point of an explicit or local transaction.
- COMMIT TRANSACTION marks an end of a successful implicit or explicit transaction.
- ROLLBACK with an optional keyword WORK rolls back a user-specified transaction to the beginning of the transaction.
- @@TRANCOUNT is a system function that returns a number of BEGIN TRANSACTION statements that occur in the current connection.
- Isolation levels are provided by the transaction to describe the extent to which a single transaction must be isolated from changes made by other transactions.
- The SQL Server Database Engine locks the resources using different lock modes, which determine the resources that are accessible to concurrent transactions.

Instructions to the Trainer(s):

- Show students Slide 26.
- Summarize the session by reading out each point on the slide.

Session 15: Error Handling

15.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. Prepare a question or two that will be a key point to relate the current session objectives.

15.1.1 Teaching Skills

To teach this session, you should be well versed with error-handling techniques in SQL Server.

You should teach the concepts in the theory class using the images provided. For teaching in the class, you are expected to use slides and LCD projectors.

In-Class Activities

Follow the order given here during In-Class activities.

Slide 2

Objectives

- Explain error handling and its implementation
- Describe the TRY-CATCH block
- Explain the procedure to display error information
- Describe the @@ERROR and RAISERROR statements
- Explain the use of ERROR_STATE, ERROR_SEVERITY, and ERROR_PROCEDURE
- Explain the use of ERROR_NUMBER, ERROR_MESSAGE, and ERROR_LINE
- Describe the THROW statement

© Aptech Ltd. Session 15 / 2

Instructions to the Trainer(s):

Give students a brief overview of the current session through the session objectives listed in Slide 2.

15.2 In-Class Explanations

Slide 3

Introduction

- Error handling in SQL Server is now easier through different techniques.
- SQL Server has introduced options that can help you to handle errors efficiently.
- Often, it is not possible to capture errors that occur at the user's end.
- SQL Server provides the TRY...CATCH statement that helps to handle errors effectively at the back end.
- Several system functions can print error related information, which can help fix errors easily.

© Aptech Ltd. Session 15 / 3

Instructions to the Trainer(s):

- Using Slide 3, explain error handling in SQL Server.
- Error handling in SQL Server has become easy through a number of different techniques.
- SQL Server has introduced options that can help you to handle errors efficiently.
- Often, it is not possible to capture errors that occur at the user's end. SQL Server provides the TRY...CATCH statement that helps to handle errors effectively at the back end.
- There are a number of system functions that print error related information, which can help fix errors easily.

Types of Errors 1-5

- As a Transact-SQL programmer, one must be aware of various types of errors that can occur while working with SQL Server statements.
- The first step one can perform is to identify the type of the error and then, determine how to handle or overcome it.

Instructions to the Trainer(s):

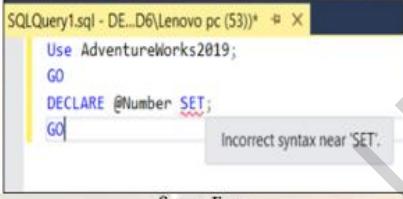
- Using Slide 4, explain the types of errors.
- As a Transact-SQL programmer, one must be aware of various types of errors that can occur while working with SQL Server statements.
- The first step one can perform is to identify the type of error and then, determine how to handle or overcome it.

Types of Errors 2-5

Some types of errors are as follows:

Syntax Errors

- Syntax errors are the errors that occur when code cannot be parsed by SQL Server. Such errors are detected by SQL Server before beginning the execution process of a Transact-SQL block or stored procedure.



The screenshot shows a SQL query window titled "SQLQuery1.sql - DE...D6\Lenovo pc (53)*". The code contains a syntax error: "DECLARE @Number SET; GO". A tooltip "Syntax Error" appears over the "SET" keyword, and the status bar at the bottom right says "Incorrect syntax near 'SET'".

© Aptech Ltd. Session 15/5

Instructions to the Trainer(s):

- Using Slide 5, explain syntax errors.
- The most common SQL error is a syntax error.
- Syntax errors are errors that occur when code cannot be parsed by SQL Server. Such errors are deleted by SQL Server before beginning the execution process of Transact-SQL block or stored procedure.

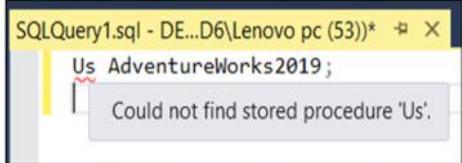
For more information, refer to:

<https://www.sqlshack.com/sql-syntax-errors/>

Slides 6 and 7

Types of Errors 3-5

If a user types a keyword or an operator wrongly because the user does not remember the valid usage, the code editor will appropriately indicate it.



SQLQuery1.sql - DE...D6\Lenovo pc (53)* Us AdventureWorks2019; Could not find stored procedure 'Us'.

Wrong Keyword Error

If the user forgets to type something that is required, the code editor will display an error once the user executes that statement.

- Syntax errors are easily identified as the code editor points them out. However, if users use a command-based application such as sqlcmd, the error is shown only after the code is executed.

© Aptech Ltd. Session 15/ 6

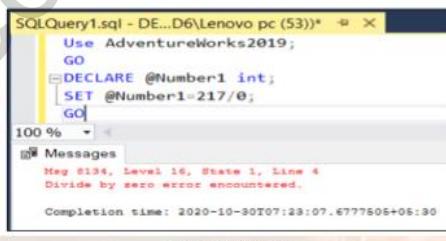
Types of Errors 4-5

Run-time Errors

- Run-time errors are errors that occur when the application tries to perform an action that is supported neither by SQL Server nor by the operating system.

Some instances where run-time errors can occur are as follows:

- Performing a calculation such as division by zero
- Trying to execute code that is not defined clearly



SQLQuery1.sql - DE...D6\Lenovo pc (53)* Use Adventureworks2019; GO DECLARE @Number1 int; SET @Number1=217/0; GO

100 %

Messages

Msg 8134, Level 16, State 1, Line 4 Divide by zero error encountered.

Completion time: 2020-10-30T07:23:07.6777505+05:30

Divide by Zero Error

© Aptech Ltd. Session 15/ 7

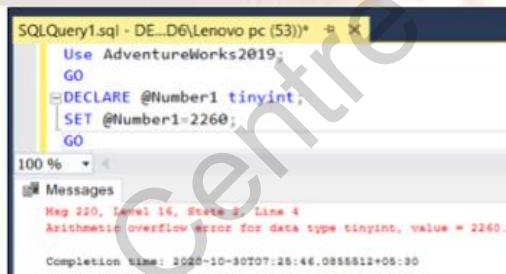
Instructions to the Trainer(s):

- Using Slides 6 and 7, discuss different types of errors.
- There are various scenarios in which errors may be caused. For example, user does not remember the valid usage and types a keyword or an operator wrongly, then an error is produced.
- Some types of errors are:
 - Syntax error
 - Runtime error

- **Syntax error:**
 - Are easily identified as the code editor points them out and thus, can be easily fixed.
- Using Slide 7, explain the Run-time errors.
- Run-time errors are the errors that occur when the application tries to perform an action that is supported neither by SQL Server nor by the operating system.
- Run-time errors are sometimes difficult to fix as they are not clearly identified or are external to the database.
- Some instances that occur in run-time errors are:
 - Performing a calculation such as division by zero.
 - Trying to execute code that is not defined clearly.

Slide 8

Types of Errors 5-5



The screenshot shows a SQL query window titled "SQLQuery1.sql - DE...D6\Lenovo pc (53)*". The query is:

```
Use AdventureWorks2019;
GO
DECLARE @Number1 tinyint,
        SET @Number1=2260;
GO
```

In the "Messages" pane, an error is displayed:

```
Msg 220, Level 14, State 1, Line 4
Arithmetic overflow error for data type tinyint, value = 2260.
```

Completion time: 10/20/2020 7:25:46.0855612+05:30

Following are some common situations when run-time error occurs:

- Using a stored procedure, or a function, or a trigger that is unavailable
- Trying to perform an action that an object or a variable cannot handle
- Attempting to access or use computer memory that is insufficient
- Trying to perform an action on incompatible types
- Using wrongly conditional statements

© Aptech Ltd. Session 15 / 8

Instructions to the Trainer(s):

- Using Slide 8, explain the common situations when run-time error occurs.
- Common situations when run-time error occurs are as follows:
 - Using a stored procedure, or a function, or a trigger that is unavailable
 - Trying to perform an action that an object or a variable cannot handle
 - Attempting to access or use computer memory that is insufficient
 - Trying to perform an action on incompatible types
 - Using wrongly conditional statements

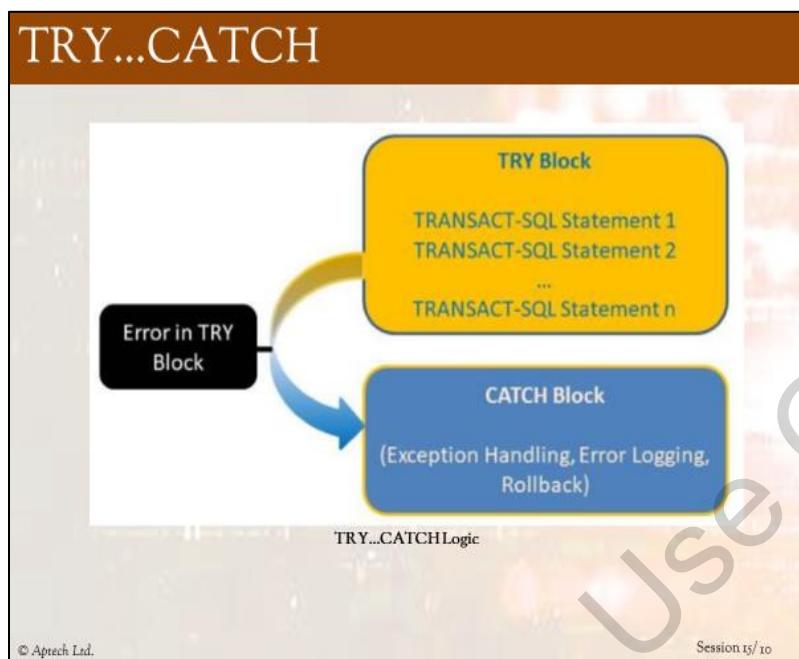
Implementing Error Handling

- While developing any application, one of the most important things that users need to take care of is error handling.
- In the same way, users also have to take care of handling exception and errors while designing the database.
- Various error handling mechanisms can be used.

- When executing some DML statements such as INSERT, DELETE, and UPDATE, users can handle errors to ensure correct output.
- When a transaction fails and the user must roll back the transaction, an appropriate error message can be displayed.
- When working with cursors in SQL Server, users can handle errors to ensure correct results.

Instructions to the Trainer(s):

- Using Slide 9, explain the process of implementing error handling.
- While developing any application, one of the most important things that users must take care of is error handling. In the same way, users also have to take care of handling exception and errors while designing the database. Various error handling mechanisms can be used.
- Some of them are as follows:
 - When executing some DML statements such as INSERT, DELETE, and UPDATE, users can handle errors to ensure correct output.
 - When a transaction fails and the user must roll back the transaction, an appropriate error message can be displayed.
 - When working with cursors in SQL Server, users can handle errors to ensure correct results.



Instructions to the Trainer(s):

- Using Slide 10, explain the TRY..CATCH statements.
- TRY...CATCH statements are used to implement exception handling in Transact-SQL. One or more Transact-SQL statements can be enclosed within a TRY block. If an error occurs in the TRY block, the control is passed to the CATCH block that may contain one or more statements.
- A TRY...CATCH construct will catch all run-time errors that have severity higher than 10 and that do not close the database connection. A TRY block is followed by a related CATCH block.
- A TRY...CATCH block cannot span multiple batches or multiple blocks of Transact-SQL statements.
- If there are no errors in the TRY block, after the last statement in the TRY block has executed, control is passed to the next statement following the END CATCH statement.
- If there is an error in the TRY block, control is passed to the first statement inside the CATCH block.
- If END CATCH is the last statement in a trigger or a stored procedure, control is passed back to the calling block.

For more information, refer to:

<https://docs.microsoft.com/en-us/sql/t-sql/language-elements/try-catch-transact-sql?view=sql-server-ver15>

In-Class Question:

Question: What is try catch in SQL?

Answer: The try catch block helps you to handle the errors in query effectively. If the SQL Server finds an error, then it exits from the TRY block and enters into the CATCH block and executes the statements inside the CATCH block.

Information 1-3

- It is a good practice to display error information along with the error, so that it can help to solve the error quickly and efficiently.
- To achieve this, system functions need to be used in the CATCH block to find information about the error that initiated the CATCH block to execute.

Using TRY...CATCH with error information

(No column name)				
	ErrorNumber	ErrorSeverity	ErrorLine	ErrorMessage
1	8134	16	2	Divide by zero error encountered.

Error Information

© Aptech Ltd.

Session 15/ 11

Information 2-3

Using TRY...CATCH with Transaction

The screenshot shows a SQL query window with the following code:

```
SQLQuery1.sql - DE...D6\Lenovo pc (51)*  X
--BEGIN TRANSACTION;
--BEGIN TRY
--    DELETE FROM Production.Product WHERE ProductID = 980;
--END TRY
--BEGIN CATCH
--    SELECT
--        ERROR_SEVERITY() AS ErrorSeverity,
--        ERROR_NUMBER() AS ErrorNumber,
--        ERROR_PROCEDURE() AS ErrorProcedure,
--        ERROR_STATE() AS ErrorState,
--        ERROR_MESSAGE() AS ErrorMessage
--    WHERE ERROR_LINE() AS ErrorLine; IF @TRANCOUNT > 0
--        ROLLBACK TRANSACTION;
--    END CATCH;
--IF @@TRANCOUNT > 0 COMMIT TRANSACTION;
GO
```

Below the code, the results pane shows a single row of data:

ErrorSeverity	ErrorNumber	ErrorProcedure	ErrorState	ErrorMessage	ErrorLine
16	547	NULL	0	The DELETE statement conflicted with the REFERENCE...	3

Constraint Violation Error

© Aptech Ltd.

Session 15/ 12

Information 3-3

Uncommittable Transactions

- If an error is generated in a TRY block, it causes the state of the current transaction to be invalid and the transaction is considered as an uncommitted transaction.
- An uncommittable transaction performs only ROLLBACK TRANSACTION or read operations.
- The transaction does not execute any Transact-SQL statement that performs a COMMIT TRANSACTION or a write operation.

© Aptech Ltd.

Session 13 / 13

Instructions to the Trainer(s):

- Using Slides 11 to 13, explain the concept of error information.
- It is a good practice to display error information along with the error, so that it can help to solve the error quickly and efficiently.
- To achieve this, system functions must be used in the CATCH block to find information about the error that initiated the CATCH block to execute.
- The functions return NULL when they are called outside the scope of the CATCH block.
- Using Slide 12, explain that TRY...CATCH block works with transactions. The statement inside the TRY block can generate a constraint violation error.
- Using Slide 13, explain about Uncommittable transactions.
- If an error is generated in a TRY block, it causes the state of the transaction to be invalid and the transaction is uncommitted transaction.
- An uncommittable transaction can only perform read operations or a ROLLBACK TRANSACTION.
- The transaction cannot execute any Transact-SQL statements that would generate a write operation or a COMMIT TRANSACTION.

@@ERROR

The @@ERROR function returns the error number for the last Transact-SQL statement executed.

```
USE AdventureWorks2019;
GO
BEGIN TRY
    UPDATE HumanResources.EmployeePayHistory SET PayFrequency = 4
    WHERE BusinessEntityID = 1;
END TRY
BEGIN CATCH
    IF @@ERROR = 547
        PRINT N'Check constraint violation has occurred.';
    END CATCH
```

It displays the following error message:
(0 rows affected)
Check constraint violation has occurred.

Instructions to the Trainer(s):

- Using Slide 14, explain the @@ERROR.
- The @@ERROR function returns the error number for the last Transact-SQL statement executed.
- The @@ERROR system function returns a value of the integer type.
- This function returns 0, if the previous Transact-SQL statement encountered no errors. It also returns an error number only if previous statements encounter an error.
- If an error is among the list of errors in the sys.messages catalog view including the value from the sys.messages.messages_id column for that error. Users can view the text associated with an @@ERROR error number in the sys.messages catalog view.

RAISERROR

- The RAISERROR statement starts error processing for a session and displays an error message.
- RAISERROR can reference a user-defined message stored in the sys.messages catalog view or build dynamic error messages at run-time.

Value	Description
LOG	Records the error in the error log and the application log for the instance of the Microsoft SQL Server Database Engine.
NOWAIT	Sends message directly to the client.
SETERROR	Sets the ERROR_NUMBER and @ERROR values to msg_id or 5000 irrespective of the severity level.

Type Specification Values

Following errors are returned back to the caller if RAISERROR executes:

- Out of scope of any TRY block
- Having severity of 10 or lower in TRY block
- Having severity of 20 or higher that terminates the database connection

Instructions to the Trainer(s):

- Using Slide 15, explain the RAISERROR statement.
- The RAISERROR statement starts the error processing for a session and displays an error message. RAISERROR can reference a user-defined message stored in the sys.messages catalog view or build dynamic error messages at run-time.
- The message is returned as a server error message to the calling application or to the associated CATCH block of a TRY...CATCH construct.
- The type specification values are as follows:
 - **LOG:** Records the error in the error log and the application log for the instance of the Microsoft SQL Server Database engine.
 - **NOWAIT:** Sends message directly to the client.
 - **SETERROR:** Sets the ERROR_NUMBER and @ERROR values to msg_id.

ERROR_STATE

➤ The ERROR_STATE system function returns the state number of the error that causes the CATCH block of a TRY...CATCH construct to execute.

Syntax:
ERROR_STATE()

- When called in a CATCH block, it returns the state number of the error message that caused the CATCH block to be run.

➤ ERROR_STATE is called from anywhere within the scope of a CATCH block. ERROR_STATE returns the error state regardless of how many times it is executed or whether it is executed within the scope of the CATCH block.

© Aptech Ltd. Session 15/ 16

Instructions to the Trainer(s):

- Using Slide 16, explain about ERROR_STATE.
- The ERROR_STATE system function returns the state number of the error that causes the CATCH block of a TRY...CATCH construct to execute.
- When called in a CATCH block, it returns the state number of the error message that caused the CATCH block to be run. This returns a NULL when it is called outside the scope of a CATCH block.
- ERROR_STATE is called from anywhere within the scope of a CATCH block.
- ERROR_STATE returns the error state regardless of how many times it is executed or whether it is executed within the scope of the CATCH block.
- This is in comparison with the functions such as @@ERROR that only returns the error number in the statement directly after the one that caused error or in the first statement of a CATCH block.

ERROR_SEVERITY

- The ERROR_SEVERITY function returns the severity of the error that causes the CATCH block of a TRY...CATCH construct to be executed.

Syntax:

ERROR_SEVERITY()

- It returns a NULL value if called outside the scope of the CATCH block. ERROR_SEVERITY can be called anywhere within the scope of a CATCH block.
- In nested CATCH blocks, ERROR_SEVERITY will return the error severity that is specific to the scope of the CATCH block where it is referenced. Users can use the ERROR_SEVERITY function in a CATCH block.

Instructions to the Trainer(s):

- Using Slide 17, explain the ERROR_SEVERITY.
- The ERROR_SEVERITY function returns the severity of the error that causes the CATCH block of a TRY...CATCH construct to be executed.
- It returns a NULL value if called outside the scope of the CATCH block. ERROR_SEVERITY can be called anywhere within the scope of a CATCH block.
- In nested CATCH blocks, ERROR_SEVERITY will return the error severity that is specific to the scope of the CATCH block where it is referenced.
- Users can use the ERROR_SEVERITY function in a CATCH block.

ERROR PROCEDURE

- The ERROR PROCEDURE function returns the trigger or a stored procedure name where the error has occurred that has caused the CATCH block of a TRY...CATCH construct to be executed.

Syntax:
ERROR PROCEDURE()

- It returns the nvarchar data type. When the function is called in a CATCH block, it will return the name of the stored procedure where the error occurred.
- ERROR PROCEDURE can be called from anywhere in the scope of a CATCH block.

Instructions to the Trainer(s):

- Using Slide 18, explain about ERROR _PROCEDURE.
- The ERROR_PROCEDURE function returns the trigger or a stored procedure name where the error has occurred that has caused the CATCH block of a TRY...CATCH construct to be executed.
- It returns the nvarchar data type. When the function is called in a CATCH block, it will return the name of the stored procedure where the error occurred. The function returns a NULL value if the error has not occurred within a trigger or a stored procedure.
- ERROR_PROCEDURE can be called from anywhere in the scope of a CATCH block.
- The function also returns NULL if this function is called outside the scope of a CATCH block.
- In nested CATCH blocks, the ERROR_PROCEDURE returns the trigger or stored procedure name specific to the scope of the CATCH block where it is referenced.

ERROR_NUMBER

- The ERROR_NUMBER system function when called in a CATCH block returns the error number of the error that causes the CATCH block of a TRY...CATCH construct to be executed.

Syntax:
ERROR_NUMBER()

- ERROR_NUMBER returns the error number irrespective of how many times it executes or whether it executes within the scope of a CATCH block.

Instructions to the Trainer(s):

- Using Slide 19, explain the ERROR_NUMBER.
- The ERROR_NUMBER system function when called in a CATCH block returns the error number of the error that causes the CATCH block of a TRY...CATCH construct to be executed.
- The function can be called from anywhere inside the scope of a CATCH block.
- The function will return NULL when it is called out of the scope of a CATCH block.
- ERROR_NUMBER returns the error number irrespective of how many times it executes or whether it executes within the scope of a CATCH block.
- This is different than the @@ERROR which only returns the error number in the statement immediately after the one that causes error or the first statement of the CATCH block.

ERROR_MESSAGE

- The ERROR_MESSAGE function returns the text message of the error that causes the CATCH block of a TRY...CATCH construct to execute.

Syntax:
ERROR_MESSAGE()

When the
ERROR_MESSAGE
function is called in the
CATCH block, it
returns the full text of
the error message that
causes the CATCH
block to execute.

The text includes the
values that are supplied
for any parameter that
can be substituted such
as object names, times, or
lengths.

© Aptech Ltd.

Session 15 / 20

Instructions to the Trainer(s):

- Using Slide 20, explain the ERROR_MESSAGE.
- The ERROR_MESSAGE function returns the text message of the error that causes the CATCH block of a TRY...CATCH construct to execute.
- When the ERROR_MESSAGE function is called in the CATCH block, it returns the full text of the error message that causes the CATCH block to execute.
- The text includes the values that are supplied for any parameter that can be substituted such as object names, times, or lengths. It also returns NULL if it is called outside the scope of a CATCH block.

ERROR_LINE

➤ The ERROR_LINE function returns the line number at which the error occurred in the TRY...CATCH block.

Syntax:
ERROR_LINE()

When this function is called in the CATCH block, it returns the line number where the error has occurred.

If the error has occurred within a trigger or a stored procedure, it returns the line number in that trigger or stored procedure.

© Aptech Ltd. Session 15/ 21

Instructions to the Trainer(s):

- Using Slide 21, explain the ERROR_LINE.
- The ERROR_LINE function returns the line number at which the error occurred in the TRY...CATCH block.
- When this function is called in the CATCH block, it returns the line number where the error has occurred.
- If the error has occurred within a trigger or a stored procedure, it returns the line number in that trigger or stored procedure.
- Similar to other functions, this function returns a NULL if it is called outside the scope of a CATCH block.

Slides 22 and 23

Errors Unaffected by the TRY...CATCH Construct 1-2

The TRY...CATCH construct does not trap the following conditions:

- Informational messages or warnings having a severity of 10 or lower
- An error that has a severity of 20 or higher that stops the SQL Server Database Engine task processing for the session
- If errors occur that have severity of 20 or higher and the database connection is not interrupted, the TRY...CATCH will handle the error
- Attentions such as broken client connection or client-interrupted requests
- When the session ends because of the KILL statements used by the system administrator

© Aptech Ltd. Session 15/ 23

Errors Unaffected by the TRY...CATCH Construct 2-2

Following types of errors are not handled by a CATCH block that occur at the same execution level as that of the TRY...CATCH construct:

- Compile errors such as syntax errors that restrict a batch from running
- Errors that arise in the statement-level recompilation such as object name resolution errors occurring after compiling due to deferred name resolution.

© Aptech Ltd. Session 15/ 23

Instructions to the Trainer(s):

- Using Slides 22 and 23, explain the errors unaffected by the TRY..CATCH construct.
- The TRY...CATCH construct does not trap the following conditions:
 - Informational messages or Warnings having a severity of 10 or lower.
 - An error that has a severity of 20 or higher that stops the SQL Server Database Engine task processing for the session. If errors occur that have severity of 20 or higher and the database connection is not interrupted, the TRY...CATCH will handle the error.
 - Attentions such as broken client connection or client-interrupted requests.
 - When the session ends because of the KILL statements used by the system administrator.
- Following types of errors are not handled by a CATCH block that occur at the same execution level as that of the TRY...CATCH construct:
 - Compile errors such as syntax errors that restrict a batch from running

- Errors that arise in the statement-level recompilation such as object name resolution errors occurring after compiling due to deferred name resolution

Slide 24

THROW

➤ The THROW statement raises an exception and transfers control of the execution to a CATCH block of a TRY...CATCH construct.

Following is the syntax of the THROW statement:

```
THROW [ [ error_number | @local_variable ],
      [ message | @local_variable ],
      [ state | @local_variable ] ]
[ ; ]
```

where,

- error_number: specifies a constant or variable that represents the error_number as int.
- message: specifies a variable or string that defines the exception message as nvarchar(2048).
- state: specifies a variable or a constant between 0 and 255 that specifies the state to associate with state of message as tinyint.

© Aptech Ltd. Session 15 / 24

Instructions to the Trainer(s):

- Using Slide 24, explain the THROW statement.
- The THROW statement raises an exception and transfers control of the execution to a CATCH block of a TRY...CATCH construct.
- In this code, the THROW statement is used to raise once again the exception that had last occurred.

In-Class Question:

Question: What is the use of THROW statement in SQL?

Answer: Throw is used to raise an exception and transfers execution to a CATCH block in SQL Server.

Summary

- Syntax errors are the errors that occur when code cannot be parsed by SQL Server.
- Run-time errors occur when the application tries to perform an action that is supported neither by Microsoft SQL Server nor by the operating system.
- TRY...CATCH statements are used to handle exceptions in Transact-SQL.
- TRY...CATCH constructs can also catch unhandled errors from triggers or stored procedures that execute through the code in a TRY block.
- GOTO statements can be used to jump to a label inside the same TRY...CATCH block or to leave a TRY...CATCH block.
- Various system functions are available in Transact-SQL to print error information about the error that occurred.
- The RAISERROR statement is used to start the error processing for a session and displays an error message.

Instructions to the Trainer(s):

- Show students Slide 25.
- Summarize the session by reading out each point on the slide.

Session 16: Enhancements in SQL Server 2019

16.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. Prepare a question or two that will be a key point to relate the current session objectives.

16.1.1 Teaching Skills

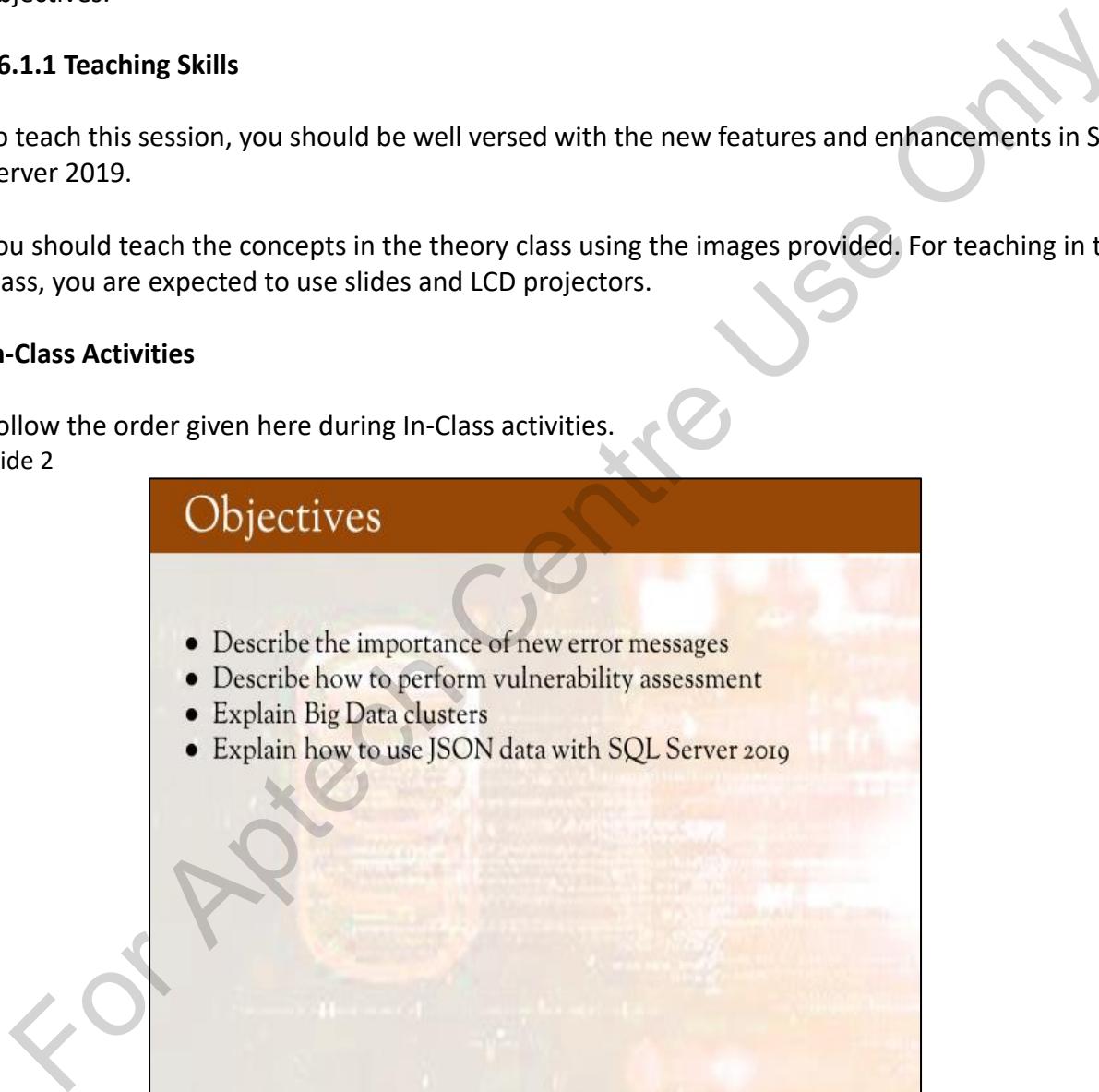
To teach this session, you should be well versed with the new features and enhancements in SQL Server 2019.

You should teach the concepts in the theory class using the images provided. For teaching in the class, you are expected to use slides and LCD projectors.

In-Class Activities

Follow the order given here during In-Class activities.

Slide 2



Objectives

- Describe the importance of new error messages
- Describe how to perform vulnerability assessment
- Explain Big Data clusters
- Explain how to use JSON data with SQL Server 2019

© Aptech Ltd. Session 16/2

Instructions to the Trainer(s):

Give students a brief overview of the current session through the session objectives listed in Slide 2.

16.2 In-Class Explanations

Slides 3 and 4

Verbose Truncation Warnings 1-2

The screenshot shows a SQL query window titled "SQLQuery2.sql - DE...D6[Lenovo pc (S)]". The query creates a table "tbl_Color" and inserts three rows: 'Red', 'Blue', and 'Green'. A truncation warning message is displayed at the bottom:

```
Msg 102, Level 14, State 10, Line 24
String or binary data would be truncated.
The statement has been terminated.
```

Completion time: 2020-11-17T15:18:34.8164999+05:30

General Error Message

© Aptech Ltd. Session 16/ 3

Verbose Truncation Warnings 2-2

The screenshot shows a SQL query window titled "SQLQuery1 - [DE...D6[Lenovo pc (S)]". The query creates a table "tbl_color" and inserts three rows: 'Red', 'Blue', and 'Green'. A truncation warning message is displayed at the bottom:

```
Msg 102, Level 14, State 1, Line 1
String or binary data would be truncated in column 'ColorName' in table 'tbl_color'. Column 'ColorName' is of type nvarchar(3).
The statement has been terminated.
```

Completion time: 2020-11-17T15:18:34.8164999+05:30

Detailed Error Message

© Aptech Ltd. Session 16/ 4

Instructions to the Trainer(s):

- Using Slides 3 and 4, explain about Verbose Truncation warnings to students.
- Verbose Truncation is one of the greatest features launched in SQL Server 2019.
- Helps to save time while performing the following:
 - importing, inserting, and updating huge amount of data.
- Using Slide 4, explain further.
- Verbose Truncation also allows users to set the database compatibility by:
 - Selecting Database Properties
 - Selecting Options
 - Setting Compatibility level

Vulnerability Assessment 1-7

- SQL Vulnerability Assessment is an easy-to-configure service that can discover, track, and help you reverse or reduce potential database vulnerabilities.
- You can use it to proactively improve your database security.

- Vulnerability Assessment is part of the Azure Defender for SQL offering, which is a unified package for advanced SQL security capabilities.
- Vulnerability Assessment can be accessed and managed via central Azure Defender for SQL portal.

© Aptech Ltd. Session 16 / 5

Instructions to the Trainer(s):

- Using Slide 5, elaborate regarding Vulnerability Assessment.
- SQL Vulnerability Assessment is an easy-to-configure service that helps users to:
 - Discover
 - Track
 - Help to reverse or reduce potential database vulnerabilities
- Tell the students that they can use to even improve their database security.
- Vulnerability Assessment is a part of Azure Defender which provides a unified package for advanced security capabilities.
- Through this, users can manage, secure, and access the required data.

For more information on Vulnerability assessment, refer to:

<https://searchsecurity.techtarget.com/definition/vulnerability-assessment-vulnerability-analysis>

<https://www.balbix.com/insights/vulnerability-assessments-drive-enhanced-security-and-cyber-resilience/>

<https://www.beyondtrust.com/resources/glossary/vulnerability-assessment>

Vulnerability Assessment 2-7

- SQL Vulnerability Assessment is a service that provides visibility into your security state.
- Vulnerability Assessment includes actionable steps to resolve security issues and enhance your database security.

- It can help you to:
 - Meet compliance requirements that require database scan reports
 - Meet data privacy standards
 - Monitor a dynamic database environment where changes are difficult to track

- The rules are based on Microsoft's best practices and focus on the security issues that present the biggest risks to your database and its valuable data.
- Results of the scan include actionable steps to resolve each issue and provide customized remediation scripts where applicable.

© Aptech Ltd. Session 16 / 6

Instructions to the Trainer(s):

- Using Slide 6, list and explain the types of errors.
- Vulnerability Assessment can be defined as a process in which it helps users identify various problems, risks in any system, hardware, software, computer, mobile devices, and so on.
- SQL Vulnerability Assessment provides security and steps that can be followed in order to enhance and secure the overall system. It can help to do the following:
 - Meet compliance requirements, meet data privacy standards, and monitor the dynamic database environment
- Rules are based on Microsoft's best practices and maintain the security issues that can prevent risks that can exploit the overall security.

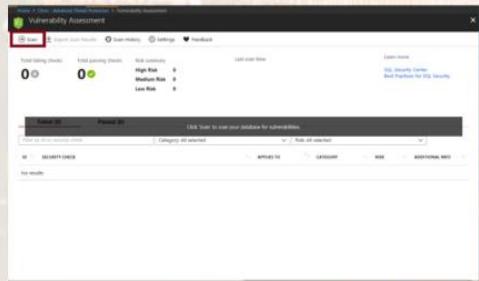
Vulnerability Assessment 3-7

You can customize an assessment report for your environment by setting an acceptable baseline for:

- ✓ Permission configurations
- ✓ Feature configurations
- ✓ Database settings

Following steps are used to implement vulnerability assessment:

Step 1: Run a scan.



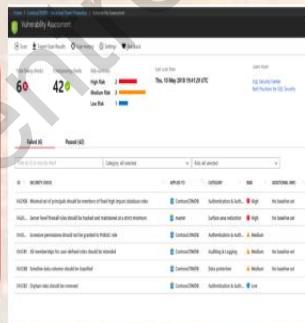
© Apteck Ltd. Session 16/7

Vulnerability Assessment 4-7

Step 2: View the report

When vulnerability scan is finished, scan report is automatically displayed in the Azure portal.

Results include warnings on deviations from best practices and a snapshot of your security-related settings, such as database principals and roles and their associated permissions.



Vulnerability Report

© Apteck Ltd. Session 16/8

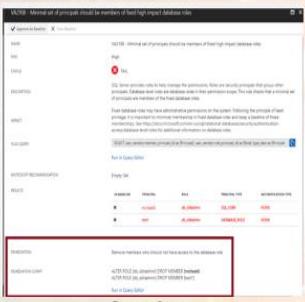
Vulnerability Assessment 5-7

Step 3: Analyze the results and resolve issues

Review your results and determine the findings in the report that are true security issues in your environment.

Drill down to each failed result to understand the impact of the finding and why each security check failed.

Use the actionable remediation information provided by the report to resolve the issue.



Security Issues

© Apteck Ltd. Session 16/9

Vulnerability Assessment 6-7

Step 4: Set your baseline

As you review your assessment results, you can mark specific results as being an acceptable baseline.

Results that match the baseline are considered as passing in subsequent scans.

After you have established your baseline security state, Vulnerability Assessment only reports on deviations from the baseline.

ID	Risk	Description
HAZ001	High	HAZ001 - Internal set of principals should be members of fixed high impact database roles
HAZ002	Medium	HAZ002 - Internal set of principals should be members of fixed high impact database roles

Approve as Baseline

© Aptech Ltd. Session 16/10

Vulnerability Assessment 7-7

Step 5: Run a new scan to see your customized tracking report

After you finish setting up your Rule Baselines, run a new scan to view the customized report.

Vulnerability Assessment now reports only the security issues that deviate from your approved baseline state.

Category	Action	Count
Security	Enforce security	30
Performance	Optimize performance	45
Compliance	Enforce compliance	10
Feature	Enable features	5

Customize Scan Report

© Aptech Ltd. Session 16/11

Instructions to the Trainer(s):

- Using Slides 7 to 11, explain the steps to implement vulnerability assessment.
- Tell the students that they can customize an assessment report for your environment by setting an acceptable baseline for:
 - Permission configurations
 - Feature configurations
 - Database settings
- Following are the steps to implement vulnerability assessment:
 - Step 1: Run a scan- Students can refer to the figure displayed on Slide 7.
 - Using Slide 8, explain the steps for vulnerability assessment.
 - Step 2: View the report
 - Once the scan process is completed, users can view the scanned report that is now displayed in the Azure portal.

- Viewing a report helps to identify the risks, monitor the activities, manage the data, and modify if required.
 - Viewing helps to identify if the scanned report consists of any errors that could be fixed.
- Using Slide 9, explain the steps for vulnerability assessment.
- Step 3: Analyze the results and resolve issues
- Once the report is scanned and viewed, users will now perform analysis. They will analyze the results that are generated and resolve the issues, if any.
 - Reviewing helps to determine the results and outputs generated in the report are true or not, this basically means taking concern of the security issues.
 - Drill down to each failed result to basically understand its impact and also to find out why the result failed or the failure occurred.
 - Take the actions required and resolve the problems.
- Using Slide 10, explain the steps for vulnerability assessment.
- Step 4: Set your baseline
- To perform on any system/project, it is important to create a baseline. Creating a baseline helps you to understand how the overall system will function and where exactly the error has occurred.
 - Once the reviewing is completed, you can mark the results that can be accepted on your baseline.
 - Failure takes place when the results are false, mismatched, and result in improper functioning of the system. Once you set the baseline and the results generated are true, this means that it has achieved Success.
 - After you have established your baseline security state, Vulnerability Assessment only reports on deviations from the baseline.
- Using Slide 11, explain the steps for vulnerability assessment.
- Step 5: Run a new scan to see your customized tracking report
- After you finish setting up your Rule Baselines, run a new scan to view the customized report.
 - Vulnerability Assessment now reports only the security issues that deviate from your approved baseline state.

In-Class Question:

Question: What is the importance of Vulnerability Assessment?

Answer: Vulnerability assessments allow security teams to apply a consistent, comprehensive, and clear approach to identifying and resolving security threats and risks.

Big Data Clusters 1-6

► In SQL Server 2019 Big Data Clusters allow to deploy scalable clusters of SQL Server, Spark, and Hadoop Distributed File System (HDFS) containers running on Kubernetes.

Popular uses of Big Data Clusters:

- Deploy scalable clusters of SQL Server, Spark, and HDFS containers running on Kubernetes
- Read, write, and process big data from Transact-SQL or Spark
- Easily combine and analyze high-value relational data with high-volume big data
- Query external data sources
- Store big data in HDFS managed by SQL Server
- Query data from multiple external data sources through the cluster
- Use the data for AI, machine learning, and other analysis tasks
- Deploy and run applications in Big Data Clusters • Virtualize data with PolyBase
- Query data from external SQL Server, Oracle, Teradata, MongoDB, and ODBC data sources with external tables
- Provide high availability for the SQL Server master instance and all databases by using Always On availability group technology

© Aptech Ltd. Session 16 / 12

Instructions to the Trainer(s):

- Using Slide 12, explain about Big Data Clusters.
- In SQL Server 2019, Big Data Clusters allow to deploy scalable clusters of SQL Server, Spark, and Hadoop Distributed File System (HDFS) containers running on Kubernetes.
- Popular uses of Big Data Clusters:
 - Deploy scalable clusters of SQL Server, Spark, and HDFS containers running on Kubernetes
 - Read, write, and process big data from Transact-SQL or Spark
 - Easily combine and analyze high-value relational data with high-volume big data
 - Query external data sources
 - Store big data in HDFS managed by SQL Server
 - Query data from multiple external data sources through the cluster
 - Use the data for AI, machine learning, and other analysis tasks
 - Deploy and run applications in Big Data Clusters
 - Virtualize data with PolyBase
 - Query data from external SQL Server, Oracle, Teradata, MongoDB, and ODBC data sources with external tables
 - Provide high availability for the SQL Server master instance and all databases by using Always On availability group technology

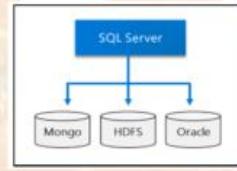
For more information on Big Data Clustering, refer to:

<https://www.bobpusateri.com/archive/2020/12/what-is-a-sql-server-big-data-cluster/>

Big Data Clusters 2-6

Data Virtualization

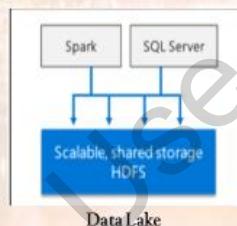
- SQL Server Big Data Clusters can query external data sources without moving or copying the data.



Data Virtualization

Data Lake

- Data Lake is a storage repository that holds a huge amount of raw data in its native format.
- It is a scalable HDFS storage pool.



Data Lake

© Aptech Ltd.

Session 16/ 13

Instructions to the Trainer(s):

- Using Slide 13, explain Data Virtualization and Data Lake.
- **Data Virtualization:** SQL Server Big Data Clusters can query external data sources without moving or copying the data.
- **Data Lake:** Data Lake is a storage repository that holds a huge amount of raw data in its native format. It is a scalable HDFS storage pool.

Big Data Clusters 3-6

Scale-out data mart

- Provides scale-out compute and storage to improve the performance of analyzing any data.

Integrated AI and Machine Learning

- Enables AI and machine learning tasks on the data stored in HDFS storage pools and the data pools.

© Aptech Ltd. Session 16 / L4

Instructions to the Trainer(s):

- Using Slide 14, explain Scale-out data mart and Integrated AI and ML.
- **Scale-out data mart:** Provides scale-out compute and storage to improve the performance of analyzing any data.
- **Integrated AI and ML:** Enables AI and machine learning tasks on the data stored in HDFS storage pools and the data pools.

Big Data Clusters 4-6

Kubernetes Terms

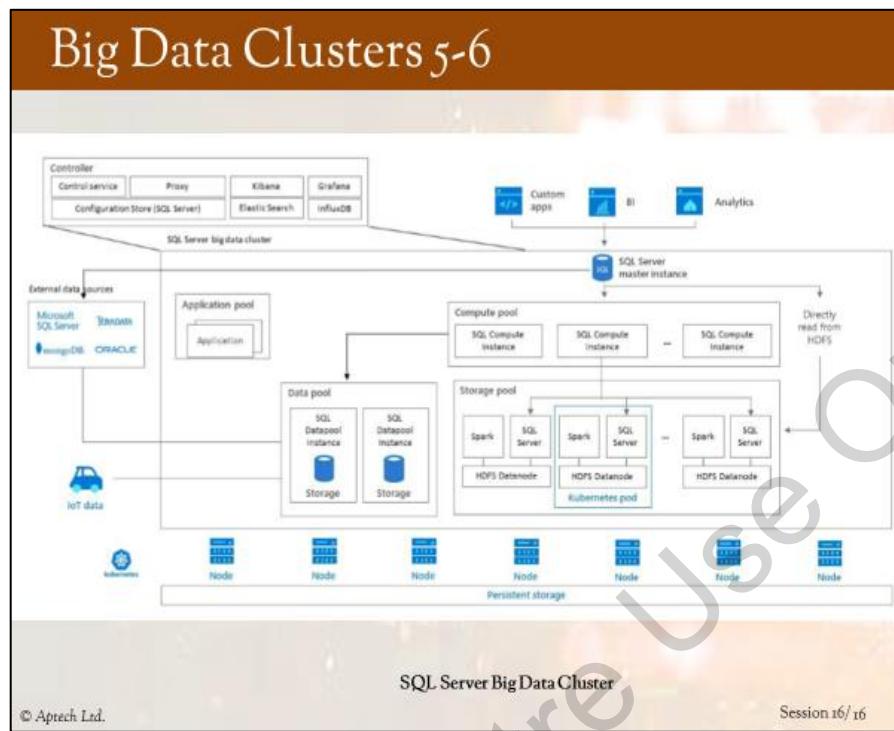
Term	Description
Cluster	A Kubernetes cluster is a set of machines, known as nodes. One node controls the cluster and is designated the master node; the remaining nodes are worker nodes. The Kubernetes master is responsible for distributing work between the workers and for monitoring the health of the cluster.
Node	A node runs containerized applications. It can be either a physical machine or a virtual machine. A Kubernetes cluster can contain a mixture of physical machine and virtual machine nodes.
Pod	A pod is the atomic deployment unit of Kubernetes. A pod is a logical group of one or more containers-and associated resources-required to run an application. Each pod runs on a node; a node can run one or more pods. The Kubernetes master automatically assigns pods to nodes in the cluster.

© Aptech Ltd.

Session 16/ 15

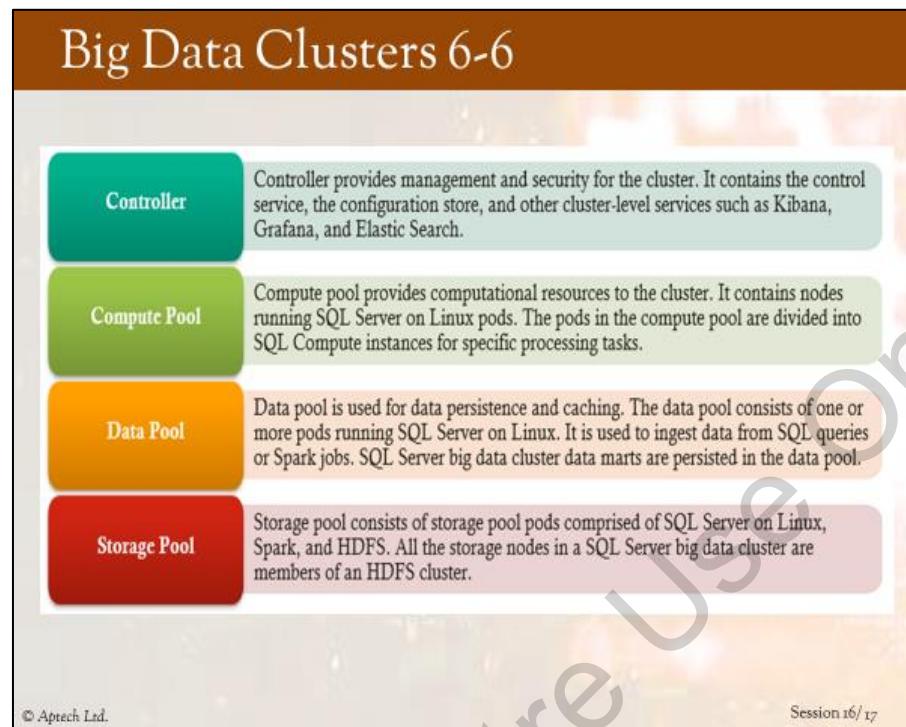
Instructions to the Trainer(s):

- Using Slide 15, explain Kubernetes terms in Big Data Clusters.
- Kubernetes terms in Big Data Clusters are described as follows:
 - **Cluster:** Set of machines known as nodes. These are basically responsible for controlling and distributing work between the workers.
 - **Node:** Run the applications. It can either be a physical or virtual machine.
 - **Pod:** Logical group of one or more containers and associated to run an application.



Instructions to the Trainer(s):

- Using Slide 16, tell students about the flow of SQL-Server Big Data Cluster.
- The Big Data Cluster consists of the Persistent storage, IoT data, External data resource, Controller, and SQL Server master instance.
- The controller consists of:
 - Control service, Proxy, Configuration Store (SQL Server), and so on.
- The compute pool comprises of the SQL Compute instance.
- Tell students that they can refer the figure displayed on Slide 16 for understanding the working of the system.



Instructions to the Trainer(s):

- Using Slide 17, explain components of Big Data Clusters.
- It consists of the following:
 - **Controller:** Manages and secures the cluster. It contains the control service, configuration store and other cluster-level services.
 - **Compute Pool:** Provides computational services. Contains nodes running SQL Server on Linux pods.
 - **Data Pool:** Used for data persistence and caching. It is used to ingest data from the SQL queries or Spark jobs.
 - **Storage Pool:** Responsible for storing SQL Server big data clusters that are members of an HDFS cluster.

In-Class Question:

Question: What are the applications of clustering?

Answer: Clustering analysis is broadly used in many applications such as market research, pattern recognition, data analysis, and image processing.

JSON data in SQL Server 1-4

- JSON is a textual data format that is used for exchanging data in modern Web and Mobile applications.
- JSON is also used to store unstructured data in log files or NoSQL databases such as Microsoft Azure Cosmos DB.
- Many REST Web services return results that are formatted as JSON text or accept data that is formatted as JSON.

Through SQL Server built-in functions and operators, you can:

- Parse JSON text and read or modify values
- Transform arrays of JSON objects into table format
- Run any Transact-SQL query on the converted JSON objects
- Format the results of Transact-SQL queries in JSON format

Instructions to the Trainer(s):

- Using Slide 18, explain about JSON data in SQL Server.
- JSON is a textual data format that is used for exchanging data in modern Web and Mobile applications.
- JSON is also used to store unstructured data in log files or NoSQL databases such as Microsoft Azure Cosmos DB.
- Many REST Web services return results that are formatted as JSON text or accept data that is formatted as JSON.
- Through SQL Server built-in functions and operators, you can:
 - Parse JSON text and read or modify values
 - Transform arrays of JSON objects into table format
 - Run any Transact-SQL query on the converted JSON objects
 - Format the results of Transact-SQL queries in JSON format

JSON data in SQL Server 2-4

Modify JSON Values

- To modify parts of JSON text, `JSON_MODIFY` (Transact-SQL) function is used to update the value of a property in a JSON string and return the updated JSON string.

```
{"info":{"address":[{"town":"Belgrade"}, {"town":"London"}, {"town":"Madrid"}]}
```

OPENJSON to Convert JSON to rowset

- Custom query language is not required to query JSON in SQL Server. To query JSON data, standard TSQL can be used.

JSON Variable to ROWSET

	id	firstName	lastName	age	dateOfBirth
1	2	John	Smith	25	NULL
2	5	Jane	Smith	NULL	2005-11-04 12:00:00.0000000

© Aptech Ltd. Session 16 / 19

Instructions to the Trainer(s):

- Using Slide 19, explain how to modify JSON and OPENJSON to convert JSON to rowset.
- **Modify JSON Values**
 - To modify parts of JSON text, `JSON_MODIFY` (Transact-SQL) function is used to update the value of a property in a JSON string and return the updated JSON string.
- **OPENJSON to Convert JSON to rowset**
 - Custom query language is not required to query JSON in SQL Server. To query JSON data, standard TSQL can be used.

JSON data in SQL Server 3-4

➤ JSON documents may have sub-elements and hierarchical data that cannot be directly mapped into the standard relational columns. In this case, you can flatten JSON hierarchy by joining parent entity with sub-arrays.

	id	firstName	lastName	age	dateOfBirth	skills	skill
1	2	John	Smith	25	NULL	NULL	NULL
2	5	Jane	Smith	NULL	2005-11-04 12:00:00.0000000	["SQL", "C#", "Azure"]	SQL
3	5	Jane	Smith	NULL	2005-11-04 12:00:00.0000000	["SQL", "C#", "Azure"]	C#
4	5	Jane	Smith	NULL	2005-11-04 12:00:00.0000000	["SQL", "C#", "Azure"]	Azure

Result of OPENJSON function calls

© Aptech Ltd. Session 16 / 20

Instructions to the Trainer(s):

- Using Slide 20, explain the JSON data in SQL Server.
- JSON documents may have sub-elements and hierarchical data that cannot be directly mapped into the standard relational columns.
- In this case, you can flatten JSON hierarchy by joining parent entity with sub-arrays.

JSON data in SQL Server 4-4

Export SQL Server Data to JSON

- You can format SQL Server data or results of SQL queries as JSON by adding the FOR JSON clause to a SELECT statement. The FOR JSON delegates the formatting of JSON output from your client applications to SQL Server.



The screenshot shows a SQL Server Management Studio window titled "SQLQuery1.sql - DE...DE (Lenovo pc (52))". It displays a JSON array of six objects, each representing a business entity with properties like BusinessEntityId, name, surname, and dob. The JSON output is as follows:

```

[{"BusinessEntityId":1,"Info":{"name":"Ken","surname":"Sanchez"}, "dob":"1989-01-07T00:00:00"}, {"BusinessEntityId":2,"Info":{"name":"Terri","surname":"Duffy"}, "dob":"2008-01-24T00:00:00"}, {"BusinessEntityId":3,"Info":{"name":"Roberto","surname":"Tamburello"}, "dob":"2007-11-04T00:00:00"}, {"BusinessEntityId":4,"Info":{"name":"Rita","surname":"Walters"}, "dob":"2007-11-28T00:00:00"}, {"BusinessEntityId":5,"Info":{"name":"Gail","surname":"Erickson"}, "dob":"2007-12-30T00:00:00"}, {"BusinessEntityId":6,"Info":{"name":"Jossie","surname":"Goldberg"}, "dob":"2013-12-16T00:00:00"}]
  
```

Exported JSON Data

© Aptech Ltd. Session 16 / 21

Instructions to the Trainer(s):

- Using Slide 21, explain how to export SQL Server Data to JSON.
- **Export SQL Server Data to JSON**
 - You can format SQL Server data or results of SQL queries as JSON by adding the FOR JSON clause to a SELECT statement. The FOR JSON delegates the formatting of JSON output from your client applications to SQL Server.

Summary

- Verbose Truncation Warnings makes troubleshooting easy by displaying additional and exact details in error messages.
- SQL Vulnerability Assessment is a service that provides visibility into your security state.
- Vulnerability report lists how many issues were found and their respective severities.
- SQL Server Big Data Clusters provide flexibility in interacting with Big Data.
- Kubernetes is an open source container orchestrator.
- Kubernetes is responsible for the state of the SQL Server Big Data Clusters.
- JSON is used to store unstructured data in log files or NoSQL databases such as Microsoft Azure Cosmos DB.

Instructions to the Trainer(s):

- Show students Slide 22.
- Summarize the session by reading out each point on the slide.

Session 17: PolyBase, Query Store, and Stretch Database

17.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. Prepare a question or two that will be a key point to relate the current session objectives.

17.1.1 Teaching Skills

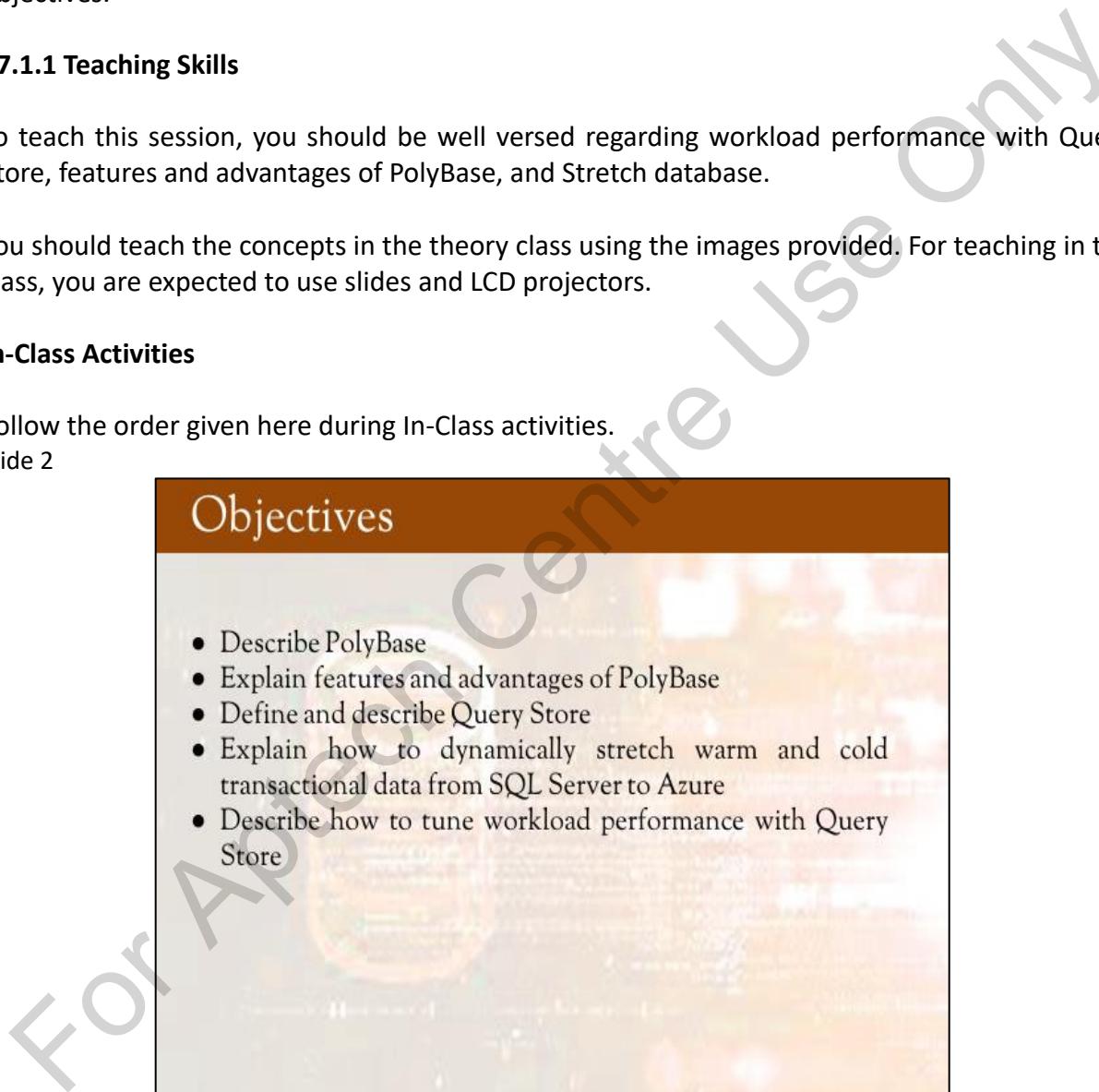
To teach this session, you should be well versed regarding workload performance with Query Store, features and advantages of PolyBase, and Stretch database.

You should teach the concepts in the theory class using the images provided. For teaching in the class, you are expected to use slides and LCD projectors.

In-Class Activities

Follow the order given here during In-Class activities.

Slide 2



Objectives

- Describe PolyBase
- Explain features and advantages of PolyBase
- Define and describe Query Store
- Explain how to dynamically stretch warm and cold transactional data from SQL Server to Azure
- Describe how to tune workload performance with Query Store

© Aptech Ltd.

Session 17 / 2

Instructions to the Trainer(s):

Give students a brief overview of the current session through the session objectives listed in Slide 2.

17.2 In-Class Explanations

Slide 3

The slide has a brown header bar with the title 'Understanding PolyBase 1-2'. Below the header is a green callout box containing two bullet points: '➤ PolyBase enables your SQL Server instance to process Transact-SQL queries that read data from external data sources.' and '➤ SQL Server 2016 and higher versions can access external data in Hadoop and Azure Blob Storage.'. Below the green box is a purple callout box containing two bullet points: '➤ Queries that access external data can also use to target relational tables in your SQL Server instance.' and '➤ This allows to combine data from external sources with high-value relational data in your database.'. At the bottom left is a small copyright notice '© Aptech Ltd.' and at the bottom right is 'Session 17 / 3'.

- PolyBase enables your SQL Server instance to process Transact-SQL queries that read data from external data sources.
- SQL Server 2016 and higher versions can access external data in Hadoop and Azure Blob Storage.

- Queries that access external data can also use to target relational tables in your SQL Server instance.
- This allows to combine data from external sources with high-value relational data in your database.

Instructions to the Trainer(s):

- Using Slide 3, tell students that they will understand the concept of PolyBase.
- Increase in storage of data in Hadoop or Azure creates the necessity for applications to connect with them and skills to work with the data.
- PolyBase enables SQL Server instance to process Transact-SQL queries that read data from external data sources.
- PolyBase is a built-in feature in SQL Server.
- SQL Server and higher versions can access external data in Hadoop and Azure Blob Storage.
- Queries that access external data can also use to target relational tables in your SQL Server instance.
- This allows to combine data from external sources with high-value relational data in your database.

For more information on PolyBase, refer to:

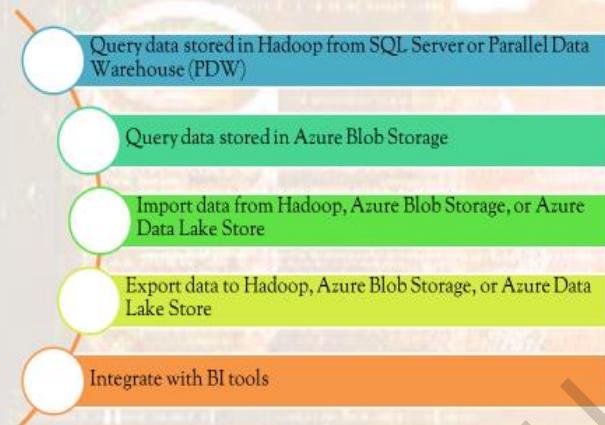
<https://www.red-gate.com/simple-talk/databases/sql-server/bi-sql-server/polybase-in-sql-server-2019-the-end-of-etl/>

<http://www.cs.put.poznan.pl/events/Microsoft-PolyBase.pdf>

<http://www.jamesserra.com/archive/2014/02/polybase-explained/>

Understanding PolyBase 2-2

PolyBase enables the following scenarios in SQL server:



© Aptech Ltd.

Session 17 / 4

Instructions to the Trainer(s):

- Using Slide 4, explain PolyBase in detail.
- PolyBase is useful in SQL Server during following scenarios:
 - Query data stored in Hadoop from SQL Server or Parallel Data Warehouse (PDW)
 - Query data stored in Azure Blob Storage
 - Import data from Hadoop, Azure Blob Storage, or Azure Data Lake Store
 - Export data to Hadoop, Azure Blob Storage, or Azure Data Lake Store
 - Integrate with BI tools

PolyBase Architecture I-2

Architecture of PolyBase has some similarities to Hadoop architecture.

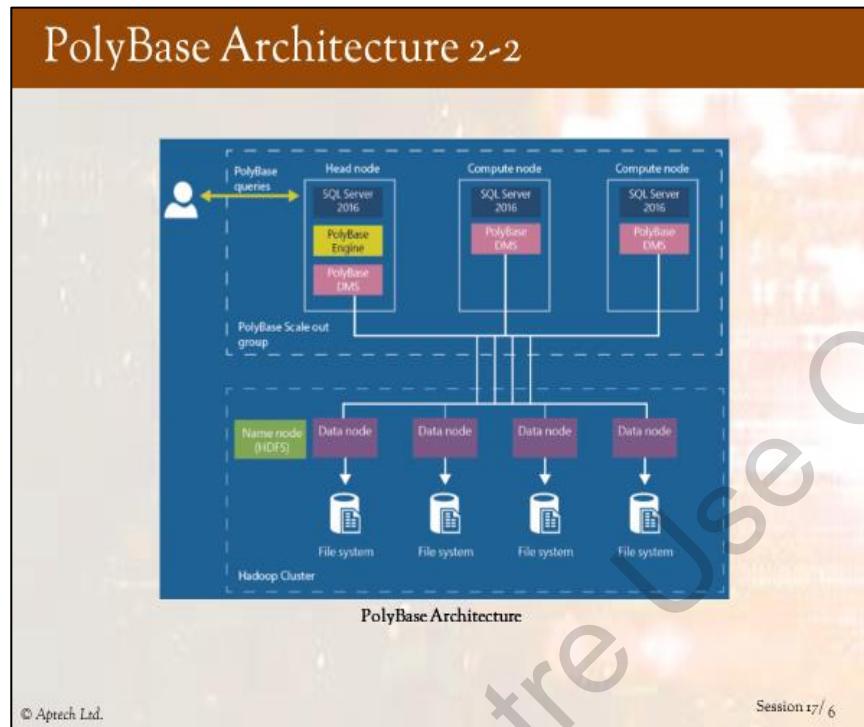
Head node	Compute node	Scale-out Reads
It contains SQL Server instance to which PolyBase queries are submitted. Each PolyBase group can have only one head node. A head node is a logical group of SQL Database Engine, PolyBase Engine, and PolyBase Data Movement Service on the SQL Server instance.	It contains SQL Server instance that assists with scale-out query processing on external data. A compute node is a logical group of SQL Server and the PolyBase data movement service on SQL Server instance. A PolyBase group can have multiple compute nodes. Both head node and compute nodes must run on the same version of SQL Server.	When querying external SQL Server, Oracle or Teradata instances, partitioned tables will benefit from scale-out reads. Each node in a PolyBase scale-out group can spin up to eight readers to read external data and each reader is assigned one partition to read in the external table.

© Aptech Ltd.

Session 17 / 5

Instructions to the Trainer(s):

- Using Slide 5, explain about PolyBase architecture.
- Architecture of PolyBase has some similarities to Hadoop architecture.
- It consists of the following:
 - Head node
 - Compute node
 - Scale-out Reads
- These are explained as follows:
 - **Head node:** It is a logical group of SQL Database Engine, PolyBase Engine, and PolyBase Data Movement Service on the SQL Server Instance.
 - **Compute node:** It contains SQL Server instance that assists with Scale-out query processing on external data.
 - **Scale-out Reads:** Each node can spin up to eight readers to external data and each reader is assigned one partition to read in the external table.



Instructions to the Trainer(s):

- Using Slide 6, students can view the diagrammatic representation of PolyBase Architecture.
- The PolyBase Architecture comprises:
 - PolyBase queries
 - PolyBase scale out group
 - Hadoop Cluster
- The Hadoop Cluster consists of Data nodes and File systems.
- The end user can access SQL Server PolyBase Engine, and PolyBase DMS

Additional Information:

PolyBase is the feature in SQL Server 2019 that is used to combine relational data in local database and non-relational data on Hadoop and generate BI Reports with ease.

Installing PolyBase in SQL Server 2019 I-4

Following are the pre-requisite to install PolyBase in SQL Server 2019:

64-bit SQL Server Evaluation edition.	Microsoft .NET Framework 4.5.	Minimum memory: 4 GB.	Minimum hard-disk space: 2 GB.	Recommended: Minimum of 16 GB RAM.	TCP/IP must be enabled for PolyBase to function correctly.
---------------------------------------	-------------------------------	-----------------------	--------------------------------	------------------------------------	--

© Aptech Ltd. Session 17 / 7

Instructions to the Trainer(s):

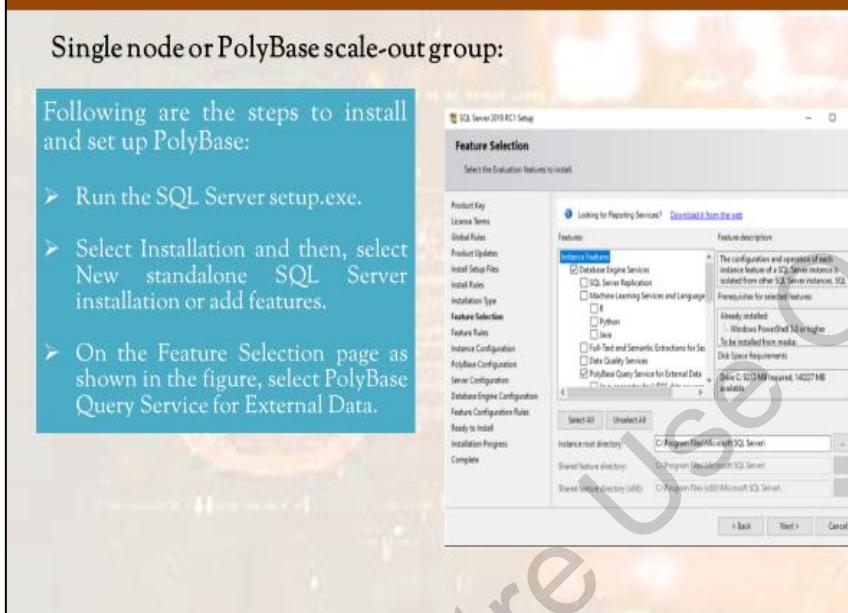
- Using Slide 7, explain the pre-requisites for installing PolyBase in SQL Server 2019.
- Following are the pre-requisites to install PolyBase in SQL Server 2019:
 - 64-bit SQL Server Evaluation edition
 - Microsoft .NET Framework 4.5
 - Minimum memory: 4 GB
 - Minimum hard-disk space: 2 GB
 - Recommended: Minimum of 16 GB RAM
 - TCP/IP must be enabled for PolyBase to function correctly

Installing PolyBase in SQL Server 2019 2-4

Single node or PolyBase scale-out group:

Following are the steps to install and set up PolyBase:

- Run the SQL Server setup.exe.
- Select Installation and then, select New standalone SQL Server installation or add features.
- On the Feature Selection page as shown in the figure, select PolyBase Query Service for External Data.



© Aptech Ltd.

Instructions to the Trainer(s):

- Using Slide 8, explain the Single node or PolyBase scale-out group.
- Following are the steps to install and set up PolyBase:
 - Run the SQL Server setup.exe.
 - Select Installation and then, select New standalone SQL Server installation or add features.
 - On the Feature Selection page as shown in the figure on slide 8, select PolyBase Query Service for External Data.

Installing PolyBase in SQL Server 2019 3-4

Enable PolyBase

- Once SQL Server installation is complete with the PolyBase feature, PolyBase must be configured to interact with external data sources.

Configure Hadoop Connectivity

- Run sp_configure with 'hadoop connectivity' and set an appropriate value for your provider.
- Restart SQL Server using services.msc.

SMS Agent Host	Provides ch...	Running	Automatic (D...	Local Syst...
SNMP Trap	Receives tra...	Manual	Local Servi...	
Software Protection	Enables the ...	Automatic (D...	Network S...	
Spot Verifier	Verifies pot...	Manual (Trig...	Local Syst...	
SQL Server (MSSQLSERVER)	Provides sto...	Running	Automatic	NT Service\...
SQL Server Agent (MSSQLSERVER)	Executes jo...	Manual	NT Service\...	
SQL Server Browser	Provides SQ...	Disabled	Local Servi...	
SQL Server PolyBase Data Movement Service. (MSSQLSERVER)	Manages co...	Running	Automatic	Network S...
SQL Server PolyBase Engine (MSSQLSERVER)	Creates, co...	Running	Automatic	Network S...

SQL Server Services

© Aptech Ltd.

Session 17 / 9

Instructions to the Trainer(s):

- Using Slide 9, explain how to enable PolyBase and configure Hadoop Connectivity.
- These are discussed as follows:
- **Enable PolyBase**
 - Once SQL Server installation is complete with the PolyBase feature, it must be configured to interact with external data sources.
- **Configure Hadoop Connectivity**
 - Run sp_configure with 'hadoop connectivity' and set an appropriate value for your provider.
 - Restart SQL Server using services.msc.

Installing PolyBase in SQL Server 2019 4-4

Configure an external table

To query the data in your Hadoop data source, you must define an external table to use in Transact-SQL queries. Following steps describe how to configure the external table:

- Create a master key on the database
- Create a database scoped credential for Kerberos-secured Hadoop clusters
- Create an external data source with CREATE EXTERNAL DATA SOURCE
- Create an external file format with CREATE EXTERNAL FILE FORMAT
- Create an external table pointing to data stored in Hadoop with CREATE EXTERNAL TABLE

Instructions to the Trainer(s):

- Using Slide 10, explain how to configure an external table.
- To query the data in your Hadoop data source, you must define an external table to use in Transact-SQL queries.
- Following steps describe how to configure the external table:
 - Create a master key on the database
 - Create a database scoped credential for Kerberos-secured Hadoop clusters
 - Create an external data source with CREATE EXTERNAL DATA SOURCE
 - Create an external file format with CREATE EXTERNAL FILE FORMAT
 - Create an external table pointing to data stored in Hadoop with CREATE EXTERNAL TABLE

Query Store

- SQL Server Query Store feature provides an insight on query plan choice and performance.
- It simplifies performance troubleshooting by helping quickly to find performance differences caused by query plan changes.
- Query Store automatically captures a history of queries, plans, and runtime statistics, and retains these for review.
- It separates data as per time frames so database usage patterns can be identified and query plan changes happened on the server are noted.

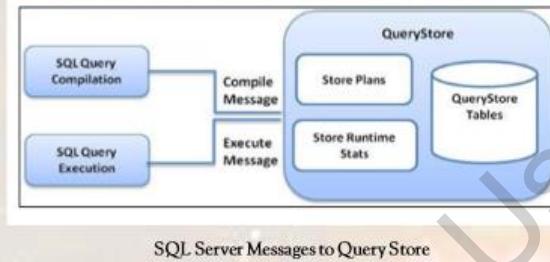
© Aptech Ltd. Session 17 / 11

Instructions to the Trainer(s):

- Using Slide 11, explain the concept of Query Store.
- SQL Server Query Store feature provides an insight on query plan choice and performance.
- It simplifies performance troubleshooting by helping quickly to find performance differences caused by query plan changes.
- Query Store automatically captures a history of queries, plans, and runtime statistics, and retains these for review.
- It separates data as per time frames so database usage patterns can be identified and query plan changes happened on the server are noted.
- Explain the query store architecture where the query information – execution plans and runtime stats are stored by Query Store.
- Explain that when a query gets submitted against a database that has the Query Store enabled, the compiled query Execution Plan is written to the Query Store Plan Store. Also, tell that the runtime information of the query is recorded in the Runtime Stats Store.
- Explain that initially the information is stored in cache.
- Tell the students that after a specific interval the information is written to disk.
- Explain that frequency is based on the specified parameters.
- Explain that information can be retrieved anytime, even after server restarts.

Query Store Architecture

- Each query compilation or execution by SQL Server sends a message to the Query store.
- The information about the compilation and execution is first stored in cache and then, stored to the disk.



© Aptech Ltd.

Session 17 / 12

Instructions to the Trainer(s):

- Using Slide 12, explain the Query Store Architecture.
- Each query compilation or execution by SQL Server sends a message to the Query store.
- The information about the compilation and execution is first stored in cache and then, stored to the disk.

Enabling Query Store

- Query Store is not enabled by default for new SQL Server and Azure Synapse Analytics (SQL DW) databases and is enabled by default for new Azure SQL Database databases.

Using SQL Server Management Studio

1. In Object Explorer, right-click a database and then, click Properties.
2. In the Database Properties dialog box, select the Query Store page.
3. In the Operation Mode (Requested) box, select Read Write.

Using Transact-SQL

Use the ALTER DATABASE statement to enable the query store for a given database shown as follows:

```
SET QUERY_STORE = ON (OPERATION_MODE = READ_WRITE);
```

© Aptech Ltd.

Session 17 / 13

Instructions to the Trainer(s):

- Using Slide 13, discuss how to enable query store.
- Query Store is not enabled by default for new SQL Server and Azure Synapse Analytics (SQL DW) databases and is enabled by default for new Azure SQL Database databases.
- **Using SQL Server Management Studio**
 - In Object Explorer, right-click a database and then, click Properties.
 - In the Database Properties dialog box, select the Query Store page.
 - In the Operation Mode (Requested) box, select Read Write.
- **Using Transact-SQL**

Use the ALTER DATABASE statement to enable the query store for a given database shown as follows:

```
SET QUERY_STORE = ON (OPERATION_MODE = READ_WRITE);
```

 - Explain about setting the query store enable to ‘True’ in SSMS or ‘ON’ with T-SQL.
 - List and explain the information about queries that can be viewed using the SSMS.
 - Give an instance or application in real life when a list of regressed queries or resource consumption details are required or encourage participants to share any such requirements in their experience.
 - Describe how to specify the values to configure Query store.

Configuring Query Store

Once the Query Store feature is enabled, other parameters for Monitoring and Query Store Retention can be configured.

The screenshot shows the 'Configure Query Store Using SSMS' dialog box with various configuration options like General, Operation Mode, Data Flush Interval, and Query Store Capture Policy. Below it, a SQL query window displays an ALTER DATABASE command setting up the Query Store with specific parameters such as operation mode, cleanup policy, data flush interval, and storage size.

Configure Query Store Using SSMS

SQLQuery1.sql - LENOVO-PC... (S21)*

```
ALTER DATABASE [DENO_1]
SET QUERY_STORE (OPERATION_MODE = READ ONLY,
CLEANUP_POLICY = (STALE_QUERY_THRESHOLD_DAYS = 367),
DATA_FLUSH_INTERVAL_SECONDS = 900,
INTERVAL_LENGTH_MINUTES = 60,
MAX_STORAGE_SIZE_MB = 100,
QUERY_CAPTURE_MODE = AUTO),
SIZE_BASED_CLEANUP_MODE = AUTO;
```

Configure Query Store Using Transact-SQL.

© Aptech Ltd. Session 17 / 14

Instructions to the Trainer(s):

- Using Slide 14, explain the configuration of Query store.
- Once the Query Store feature is enabled, other parameters for Monitoring and Query Store Retention can be configured.
- Slide 14, displays the window for configuration of query store using SSMS and Transact-SQL.

Additional Information:

Query Store feature and View Top Resource Consuming Queries option feature in SQL Server help to view a list of all queries that are consuming major resources.

INTERVAL_LENGTH_MINUTES parameter and DATA_FLUSH_INTERVAL_SECONDS parameters must be configured in the Query Store feature to specify the duration of keeping the information in the cache and the time when it is stored on the disk.

Additional Information:

For more information, refer to:

<http://slavasql.blogspot.in/2015/09/sql-server-2016-query-store.html>

<http://windowsitpro.com/sql-server-2016/what-sql-server-2016-query-data-store>

<http://www.infoq.com/news/2015/07/SQL-Server-Query-Store>

In-Class Question:

Question: Where are SQL queries stored?

Answer: The data in Query Store is stored in the database where the SQL Server Query Store is enabled.

Slide 15

Query Store System Objects

New system procedures and catalog views related to Query Store can be found in SQL Server.

name	type_desc
1 sp_query_store_force_plan	EXTENDED_STORED_PROCEDURE
2 sp_query_store_reset_exec_stats	EXTENDED_STORED_PROCEDURE
3 sp_query_store_remove_plan	EXTENDED_STORED_PROCEDURE
4 sp_query_store_flush_db	EXTENDED_STORED_PROCEDURE
5 database_query_store_options	VIEW
6 sp_query_store_remove_query	EXTENDED_STORED_PROCEDURE
7 sp_query_store_consistency_check	EXTENDED_STORED_PROCEDURE
8 sp_query_store_unforce_plan	EXTENDED_STORED_PROCEDURE
9 query_store_wait_stats	VIEW
10 query_store_runtime_stats_interval	VIEW
11 query_store_runtime_stats	VIEW
12 query_store_query_text	VIEW
13 query_store_query	VIEW
14 query_store_plan	VIEW
15 query_context_settings	VIEW

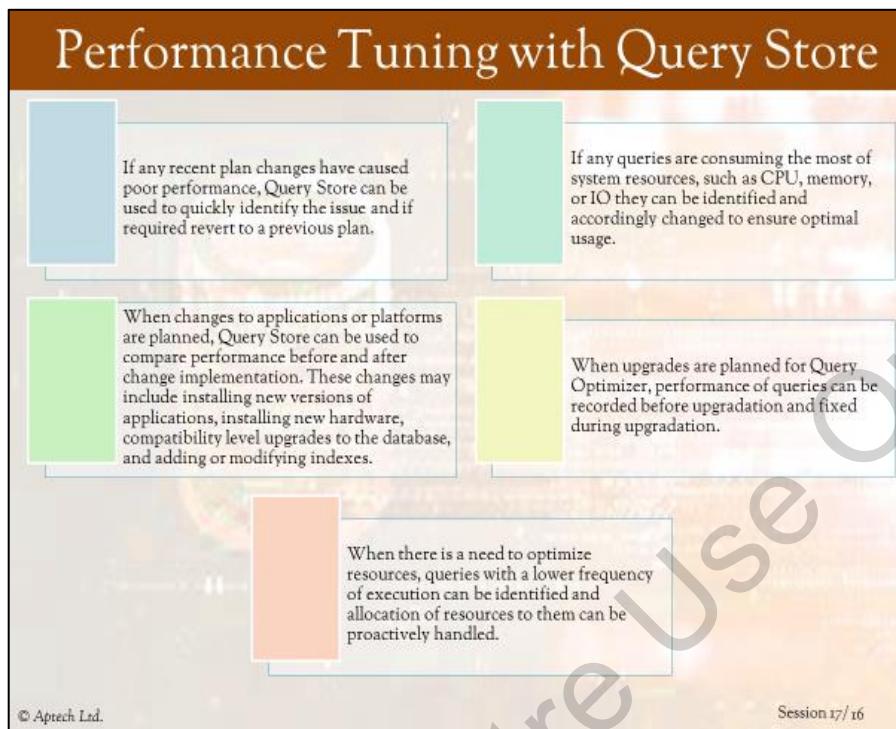
Query executed successfully.

System-Stored Procedures and Catalog Views

© Aptech Ltd. Session 17 / 15

Instructions to the Trainer(s):

- Using Slide 15, explain the concept of query store system objects.
- New system procedures and catalog views related to Query Store can be found in SQL Server.
- Explain how storing data for long periods on-premise is costly and how the facility to move it to the cloud and also query the data with the same ease as with on-premise data is useful.
- List the advantages of the Query store feature. Mention the uses of being able to access different versions of row data, such as analyzing trends, support business decisions, and revert to previous version of data in case of errors.

**Instructions to the Trainer(s):**

- Using Slide 16, explain about Performance Tuning with Query Store.
- If any recent plan changes have caused poor performance, Query Store can be used to quickly identify the issue and if required revert to a previous plan.
- If any queries are consuming the most of system resources, such as CPU, memory, or IO they can be identified and accordingly changed to ensure optimal usage.
- When changes to applications or platforms are planned, Query Store can be used to compare performance before and after change implementation. These changes may include installing new versions of applications, installing new hardware, compatibility level upgrades to the database, and adding or modifying indexes.
- When upgrades are planned for Query Optimizer, performance of queries can be recorded before upgradation and fixed during upgradation.
- When there is a requirement to optimize resources, queries with a lower frequency of execution can be identified and allocation of resources to them can be proactively handled.

Stretch Database

- Stretch Database is a new feature built into SQL Server that facilitates storage of a part of a database in the cloud.
- When there is a need to retain data for a longer time, Stretch Database feature enables secure migration of tables and data to the cloud.

Moreover, applications can still query the data in the same way as before. Enabling Stretch Database feature makes it possible to do the following:

- Move archived data as well as current data to the cloud securely by using the encryption features
- Access and query the data on the cloud at any time, without any changes to existing applications or queries
- Reduce storage requirements of on-premise data by using the vast storage capacity of the cloud
- Reduce processing burden on the on-premise data by running the processes on the cloud in a way that is transparent to the applications

Instructions to the Trainer(s):

- Using Slide 17, explain the Stretch Database concept.
- Stretch Database is a new feature built into SQL Server that facilitates storage of a part of a database in the cloud.
- When there is a requirement to retain data for a longer time, Stretch Database feature enables secure migration of tables and data to the cloud.
- Moreover, applications can still query the data in the same way as before. Enabling Stretch Database feature makes it possible to do the following:
 - Move archived data as well as current data to the cloud securely by using the encryption features
 - Access and query the data on the cloud at any time, without any changes to existing applications or queries
 - Reduce storage requirements of on-premise data by using the vast storage capacity of the cloud
 - Reduce processing burden on the on-premise data by running the processes on the cloud in a way that is transparent to the applications

In-Class Question:

Question: What are the applications of clustering?

Answer: Clustering analysis is broadly used in many applications such as market research, pattern recognition, data analysis, and image processing.

Question: Can memory-optimized tables be migrated to an Azure database?

Answer: No. Memory-optimized tables are not supported by Stretch database that is the feature enabling migration of data to Azure.

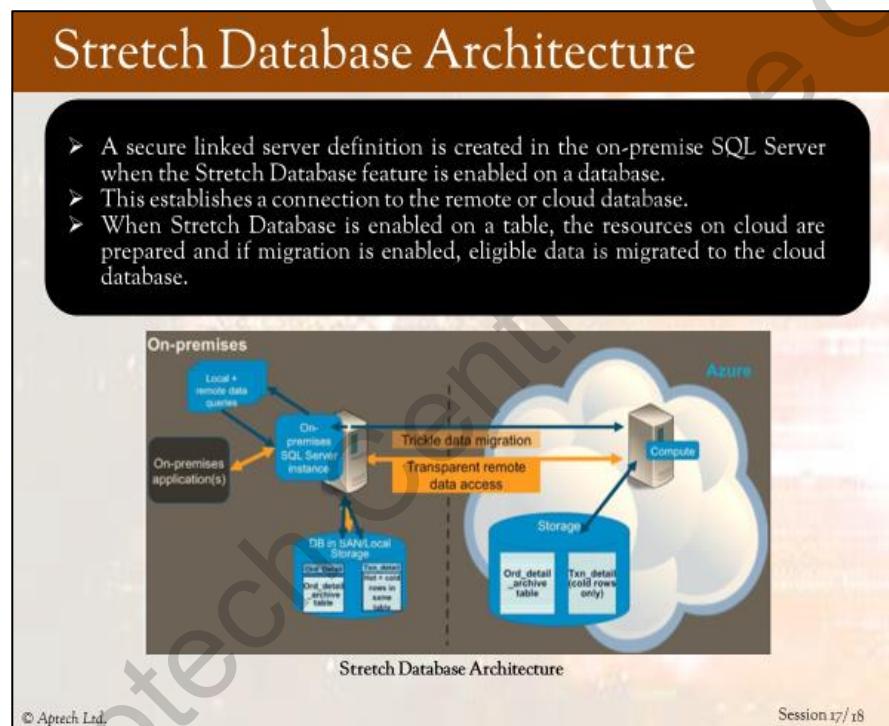
Additional Information:

For more information, refer to:

<http://www.thewhir.com/web-hosting-news/new-microsoft-hybrid-storage-capabilitiesstretch-databases-between-on-prem-and-azure-cloud>

<http://www.databasejournal.com/features/mssql/getting-started-with-stretch-databasefunctionality-in-sql-server-2016-part-1.html>

Slide 18



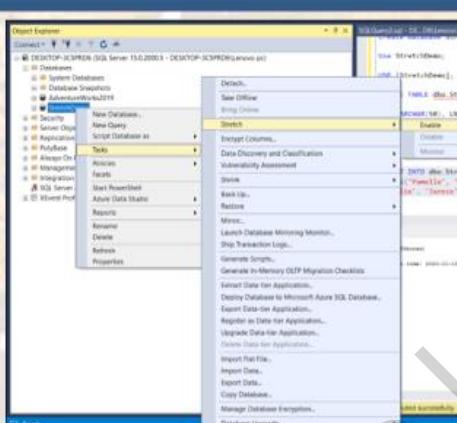
Instructions to the Trainer(s):

- Using Slide 18, explain the stretch database architecture.
- A secure linked server definition is created in the on-premise SQL Server when the Stretch Database feature is enabled on a database.
- This establishes a connection to the remote or cloud database.
- When Stretch Database is enabled on a table, the resources on cloud are prepared and if migration is enabled, eligible data is migrated to the cloud database.

Setting up Stretch Database 1-2

Prerequisite to Enable Stretch Database

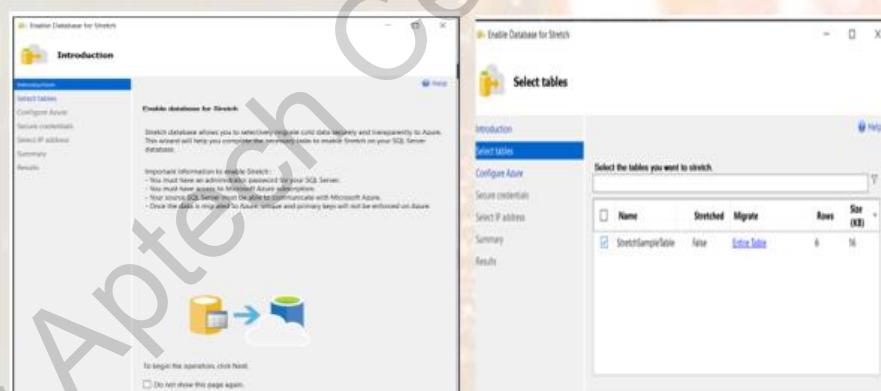
- Enable Database for Stretch wizard configures the server for Stretch.
- An administrator has to enable it manually by running sp_configure first then, run the wizard, or an administrator has to run the wizard.



Enabling Database for Stretch

© Aptech Ltd. Session 17 / 19

Setting up Stretch Database 2-2



Enable Database for Stretch

Select Tables

© Aptech Ltd. Session 17 / 20

Instructions to the Trainer(s):

- Using Slides 19 and 20, explain how to set up Stretch Database.
- Enable Database for Stretch wizard configures the server for Stretch.
- An administrator has to enable it manually by running sp_configure first then, run the wizard, or an administrator has to run the wizard.

Stretch Database Limitations 1-3

Limitations for Stretch Enable table are:

Constraints

- Uniqueness is not enforced for UNIQUE constraints and PRIMARY KEY constraints in the Azure table that contains the migrated data.

DML operations

- You are not able to UPDATE or DELETE rows that have been migrated or rows that are eligible for migration, in a Stretch-enabled table or in a view that includes Stretch-enabled tables.
- You are not able to INSERT rows into a Stretch-enabled table on a linked server.

Indexes

- You are not able to create an index for a view that includes Stretch-enabled tables.
- Filters on SQL Server indexes are not propagated to the remote table.

Stretch Database Limitations 2-3

Following items currently prevent you from enabling Stretch for a table:

Table properties

- Tables that have more than 1,023 columns or more than 998 indexes
- FileTables or tables that contain FILESTREAM data
- Tables that are replicated, or that are actively using Change Tracking or Change Data Capture
- Memory-optimized tables

Data types

- text, ntext, and image
- timestamp
- sql_variant
- XML
- CLR data types including geometry, geography, hierarchyid, and CLR user-defined types

Column types

- COLUMN_SET
- Computed columns

Stretch Database Limitations 3-3

Constraints

- Default constraints and check constraints
- Foreign key constraints that reference the table. In a parent-child relationship (for example, Order and Order_Detail), you can enable Stretch for the child table (Order_Detail) but not for the parent table (Order).

Indexes

- Full text indexes
- XML indexes
- Spatial indexes
- Indexed views that reference the table

© Aptech Ltd.

Session 17 / 23

Instructions to the Trainer(s):

- Using Slides 21 to 23, explain limitations for Stretch database.
- Limitations for Stretch Enable table are:

Constraints

- Uniqueness is not enforced for UNIQUE constraints and PRIMARY KEY constraints in the Azure table that contains the migrated data.

DML operations

- You are not able to UPDATE or DELETE rows that have been migrated or rows that are eligible for migration, in a Stretch-enabled table or in a view that includes Stretch-enabled tables.
- You are not able to INSERT rows into a Stretch-enabled table on a linked server.

Indexes

- You are not able to create an index for a view that includes Stretch-enabled tables.
- Filters on SQL Server indexes are not propagated to the remote table.

- Using Slide 22, explain the stretch database limitations.
- Following items currently prevent you from enabling Stretch for a table:

➤ Table properties

- Tables that have more than 1,023 columns or more than 998 indexes
- FileTables or tables that contain FILESTREAM data
- Tables that are replicated, or that are actively using Change Tracking or Change Data Capture
- Memory-optimized tables

- **Data types**
 - text, ntext, and image
 - timestamp
 - sql_variant
 - XML
 - CLR data types including geometry, geography, hierarchyid, and CLR user-defined types
- **Column types**
 - COLUMN_SET
 - Computed columns
- Using Slide 23, explain the stretch database limitations.
- **Constraints**
 - Default constraints and check constraints
 - Foreign key constraints that reference the table. In a parent-child relationship (for example, Order and Order_Detail), you can enable Stretch for the child table (Order_Detail) but not for the parent table (Order).
- **Indexes**
 - Full text indexes
 - XML indexes
 - Spatial indexes
 - Indexed views that reference the table

Summary

- The PolyBase feature provides seamless integration with external data sources, such as Hadoop or Azure Blob Storage.
- PolyBase eliminates the need to specialized skills on Hadoop internals by enabling query-runs on external data sources with simple Transact-SQL commands.
- Query Store is a built-in tool to improve performance by maintaining historical information of every query and execution plan.
- Query Store tracks the performance of queries and triggers alerts on poorly performing plans.
- Stretch Database in SQL Server 2019 enables stretching some part of a database to the Azure cloud, thereby, lowering long-term storage costs as well as maintenance efforts.
- Stretch Database is used to migrate archive and data in current transaction to the cloud securely.
- Stretch database is used to access cloud data at any time similar to the local data available on machines.

Instructions to the Trainer(s):

- Show students Slide 24.
- Summarize the session by reading out each point on the slide.