

Workbook for SQL SERVER 2005

A beginner's guide to effective programming

Why This Module

Microsoft SQL Server is a relational database management system (RDBMS) produced by Microsoft. Its primary query language is Transact-SQL, an implementation of the ANSI/ISO standard Structured Query Language (SQL). It is comprehensive, integrated data management and analysis software that enables organizations to reliably manage mission-critical information and confidently run today's increasingly complex business applications. SQL Server 2005 allows companies to gain greater insight from their business information and achieve faster results for a competitive advantage.

In today's computerized environment, there is usually a back-end application and a front-end application. The front-end application interacts with the backend application. A front-end application, developed using front-end tools, can be a stand-alone application residing on a desktop or it can be Web-based. End-users use the front-end applications to interact with backend applications. Data is typically stored at the back-end and usually managed by an RDBMS.

Some very popular RDBMSs are Microsoft SQL Server and Oracle server. This module covers the features of an RDBMS with specific reference to Microsoft SQL Server 2005. This module is necessary for any participant to appreciate the back end activities.

Contents

1. Overview Of SQL Server	7-20
1.1 Tools of SQL Server	7
1.2 SQL server profiler	11
1.3 SQL server database architecture	13
1.4 Managing databases	16
1.5 Crossword	20
2. Using Transact-SQL	21-35
2.1 Exploring Transact-SQL	22
2.2 Querying Data	27
2.3 Changing Data Through Queries	31
2.4 Crossword	35
3. Handling Complex Queries	36-57
3.1 Joining Multiple Tables	37
3.2 Create and alter table	42
3.3 Constraints	46
3.4 Implementing Sub Queries	51
3.5 Derived Tables	54
3.6 Crossword	57
4. Creating Tables	58-71
4.1 Defaults	59
4.2 Rules	64
4.3 Normalization	67
4.4 Relationships	69
4.5 Crossword	71
5. Introduction To Programming Objects	72-95
5.1 Creating and Managing Views	73
5.2 Indexes	76
5.3 Implementing stored procedure	80
5.4 Implementing triggers	84

5.5	Creating user defined functions	91
5.6	Crossword	95

6. Advanced T-SQL :Programmability	96-109
---	---------------

6.1	Batches	97
6.2	Scripts	100
6.3	Variables	100
6.4	Try and Catch	102
6.5	Cursor	105
6.6	Crossword	109

7. Managing Transactions and Locks	110-122
---	----------------








7.1	Introduction to Transactions	111
7.2	Managing transactions	111
7.3	SQL server Locking	116
7.4	Crossword	122






****Answers For Crosswords** **123**

****Contributors** **124**

Guidelines to Use this Workbook






Conventions Used

Convention	Description
 Topic	Indicates the Topic Statement being discussed.
Estimated Time	Gives an idea of estimated time needed to understand the Topic and complete the Practice session.
 Presentation	Gives a brief introduction about the Topic.
 Scenario	Gives a real time situation in which the Topic is used.
 Demonstration/Code Snippet	Gives an implementation of the Topic along with Screenshots and real time code.
<i>Code in Italic</i>	Represents a few lines of code in Italics which is generated by the System related to that particular event.
// OR	Represents a few lines of code (Code Snippet) from the complete program which describes the Topic.
 Context	Explains when this Topic can be used in a particular Application.
 Practice Session	Gives a practice example for the participant to implement the Topic, which gives him a brief idea how to develop an Application using the Topic.
 Check list	Lists the brief contents of the Topic.

 Common Errors	Lists the common errors that occur while developing the Application.
 Exceptions	Lists the exceptions which result from the execution of the Application.
 Lessons Learnt	Lists the lessons learnt from the article of the workbook.
 Best Practices	Lists the best ways for the efficient development of the Application.
 Notes	Gives important information related to the Topic in form of a note

1.0 Overview Of SQL Server

Topics

-  1.1 Tools of SQL Server
-  1.2 SQL Server profiler
-  1.3 SQL Server Database Architecture
-  1.4 Managing Databases
-  1.5 Crossword





Topic: Tools of SQL Server

Estimated Time – 40mins.



Objectives : At the end of the activity, the participant should understand

- SQL Server Management Studio
- SQL Server Integration Services(SSIS)
- SQL Server Analysis Service(SSAS)
- SQL Server Business Intelligence Development Studio



Presentation :

- **SQL Server Management Studio**
 - Is an integrated environment for accessing, configuring, managing, administering and developing all components of SQL Server.
 - Combines the features of Enterprise Manager, Query Analyzer, and Analysis Manager into a single environment.
 - Works with all components of SQL Server such as Reporting Services, Integration Services, SQL Server Compact Edition and Notification Services.
- **SQL Server Integration Services**
 - Is a platform for building high performance data integration solutions, including Extraction, Transformation and Load (ETL) packages for data warehousing.
 - Includes graphical tools and wizards for building and debugging packages.

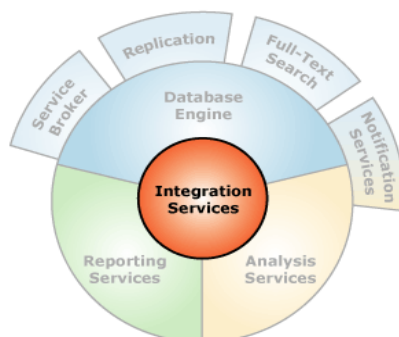


Figure 1.1-1: SQL Server Integration Services

- **SQL Server Analysis Services**

- Delivers Online Analytical Processing (OLAP) and data mining functionality for business intelligence applications.
- Supports OLAP by letting you design, create, and manage multi-dimensional structures that contain data aggregated from other data sources, such as relational databases.

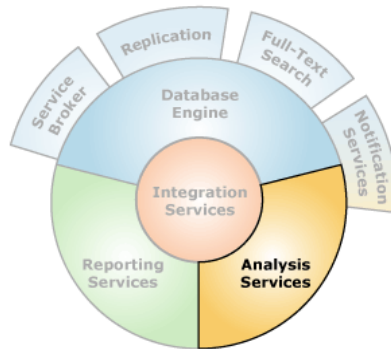


Figure 1.1-2: SQL Server Analysis Services

Business Intelligence Development Studio

- Is an integrated environment for developing business intelligence constructs, such as Cubes, Data sources, Reports, and Integration Services packages.
- Includes project templates that provide a context for developing specific constructs.
- Can aid in developing projects as part of a solution that is independent of any particular server.



Context :

- To understand the tools and services offered by SQL Server.
- To get superior Performance with right configuration.



Practice Session/s :

- Identify the Services that are a part of SSIS and SSAS.



Check List :

- Understanding of SQL Server Management Studio Tools.



Common Error/s :

- Unable to distinguish between different tools of SQL Server.



Exception/s :

- The tools provided in one version may not be supported by other versions.



Lesson/s Learnt :

- ☒ The approach of SQL Server aids in using different tools for diverse working environment.
- ☒ The method to achieve superior performance by using the suitable tool



Best Practice/s :

- Choose the appropriate tool according to the working environment.



Topic: SQL Server Profiler

Estimated Time – 45mins.



Objectives : At the end of the activity, the participant should understand-

- The significance of SQL Server Profiler.
- The function of SQL Server Profiler.



Presentation :

- **SQL Server Profiler**
 - Explains how queries are resolved internally.
 - Can be used in creating and observing a trace, as it runs and stores the results in a table.
 - Replay the trace results.
 - Large traces can be filtered as per the information provided.



Context :

- Analyzes and debugs SQL statements and stored procedures.
- Examines performance.
- Stress Analysis.
- Involved in common debugging and troubleshooting.
- Fine-tunes indexes.
- Audits and reviews security activity.



Practice Session/s :

- Open SQL Server Profiler and trace the security Audit in the database northwind.
- Using SQL Server Profiler traces the Performance of the T-SQL in the database northwind.



Check List :

- Significance of SQL Server Profiler.
- Techniques of using SQL Server Profiler to obtain the best result out of the method that is used.



Common Error/s :

- Attempting to access SQL Server Profiler without having the authorization to connect to a specific instance of SQL Server.



Exception/s :

- The use of SQL Server Profiler requires 10 MB of free disk space; if the available disk space is below 10 MB, it stops functioning.



Lesson/s Learnt :

- SQL Server Profiler is a tool that captures SQL Server events from the server and saves these events in a trace file.
- SQL Server Profiler is used to Monitor, Analyze, and Tune SQL performance.



Best Practice/s :

- To implement the tools of SQL Server Profiler, to ensure superior performance of the system.



Topic: SQL Server Database Architecture

Estimated Time – 25 mins.



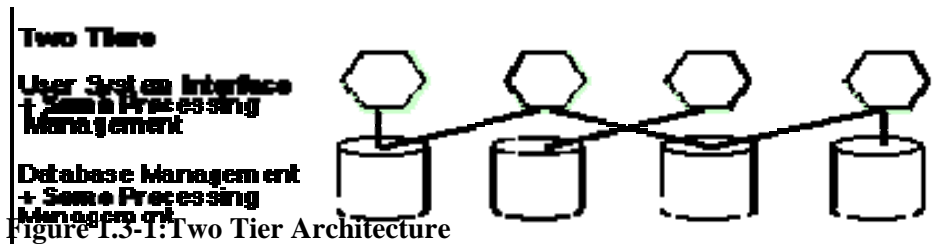
Objectives : On completion of the activity, the participant should be able to comprehend or know the

- Architectures of SQL Server Database.
- Two-tier and Three-tier architectures.
- Need to use different architectures.



Presentation :

- **Logical Architecture**
 - Data Mining is a process that involves the interaction of multiple components. Sources of data in SQL Server database or any other data source for use in training, testing or prediction can be accessed. On completion of solution it is deployed to an instance of Analysis Services.
 - To create a Data Mining project, the following must be defined - Data Sources, Data Mining Structures and Data Mining Models.
 - The Mining Structure is a Data Structure that defines the Data Domain from which Mining Models are built. A single mining structure can contain multiple mining models that share the same domain.
 - The Data Mining structure can also be divided into training and a test set by specifying a percentage or amount of data as a HOLDOUT.
- **Physical Architecture**
 - Two-Tier Architecture**
 - Consists of three components distributed between two layers: Client and Server.
 - User System Interface (such as session, text input, dialog, and display management services).
 - Processing Management (such as process development, process enactment, process monitoring, and process resource services).
 - Database Management (such as data and file services).



Three-Tier Architecture

It establishes a Server (or an "agent") between the Client and the Server.

- **The GUI Layer:** The 'Top Layer' that contains all things that is visible to the user, the 'outside' of the system (i.e. screen layout and navigation).
- **The Object Layer:** Core of the system, the linking pin between the other layers.
- **Database Layer:** Ensures Persistency. Object from the Object layer can write itself to one or more tables. It contains database, connection, table, SQL and result set.



Context :

- To understand the framework of different Architectures of the SQL Database.



Practice Session/s :

- Find out the necessity of using different Database Architecture in one application.
- Explain Active/Active and Active/Passive cluster configurations.
- Find out the various ways of moving Data/databases between Servers and Databases in SQL Server.
- Find out the importance of using ROLAP, MOLAP and OLAP.



Check List :

- Different layers of Database Architecture.
- Two-tier and three-tier Architecture.
- Need to use different Architecture.



Common Error/s :

- Executing an application, without linking all the layers (through reference or dll files) of N-tier Architecture may cause an error.



Exception/s :

- Two or more layers of the Database Architecture, when present on one system, may lead to System Malfunctioning.



Lesson/s Learnt :

- ☑ Importance of different Architectures of the Database.
- ☑ Different layers of Database Architecture.



Best Practice/s :

In N-tier Architecture, different layers should remain present on different Systems.



Topic: Managing Database

Estimated Time – 30 mins.



Objectives : On completion of the activity, the participant should be aware of

- Types of Databases.
- The ways to Manage Database.
- Significance of Database Models.



Presentation :

- **Managing database**
 - In an **Object Oriented Database**, information is represented in the form of **Objects**.
 - Database capabilities are combined with Object Oriented Programming Language capabilities to make Object Database Management System (ODBMS).
 - An ODBMS makes database objects appear as programming language objects in one or more object programming languages.
 - An ODBMS extends the programming language with transparent persistent data, concurrency control, data recovery, associative queries and other capabilities.
- **Database Models**
 - Flat model
 - Hierarchical model
 - Network model
 - Relational model
 - Dimensional model
 - Object database models
- **Types of Databases**

Master Database

- Records all the system level information and other Databases information present on SQL Server System.
- SQL Server will not work if the **Master** Database is unavailable.

Model Database

- It is used as the template for all databases created on an instance of SQL Server.
- **Tempdb** is created every time when SQL Server is started, hence **Model** Database must exist on a SQL Server System.

MSDB Database

- Used by SQL Server Agent for scheduling alerts, jobs and other features such as Service Broker and Database Mail.

TEMPDB database

- It is a System Database used by SQL Server to store temporary tables and procedures.

- **Files to store a database are:**

- **Primary files:** Contains startup information for the database. Every database has one primary file.
- **Secondary files:** Holds all the data that does not fit in the primary data file.
- **Transaction log:** Holds the log information used to recover the database. There must be at least one transaction log file for each database. The minimum size for a log file is 512 kilobytes (KB).



Scenario :

‘Barclays is one of the international banks. ‘Mr. Richard’ is the database administrator of the bank. A database has to be created for the account details to be stored.



Demonstration/Code Snippet :

Step 1: Create a database named ‘Accounting’ using the following code as shown below.

```
CREATE DATABASE Accounting
ON
    (NAME = 'Accounting',
    FILENAME = 'C:\Program Files\Microsoft SQL
    Server\MSSQL\data\AccountingData.mdf',
    SIZE = 10,
    MAXSIZE = 50,
    FILEGROWTH = 5)

LOG ON

    (NAME = 'AccountingLog',
    FILENAME = 'C:\Program Files\Microsoft SQL
    Server\MSSQL\data\AccountingLog.ldf',
    SIZE = 5MB,
    MAXSIZE = 25MB,
    FILEGROWTH = 5MB)

GO
```

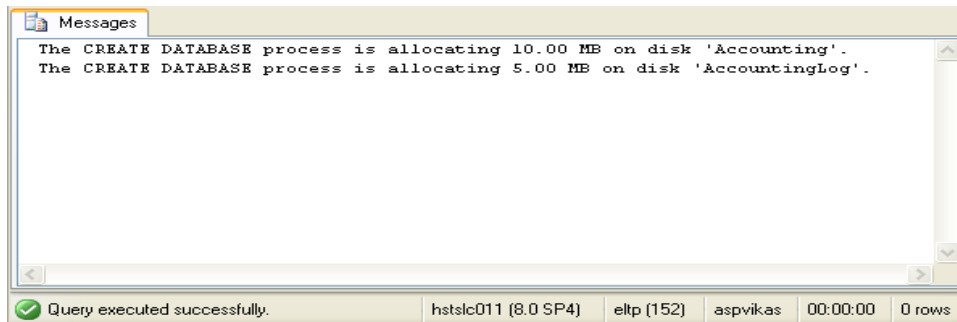


Figure 1.4-1: Result

Step 2: Once the database is created, the structure information can be viewed using the following stored procedure.

```
EXEC SP_HELPDB 'Accounting'
```

	name	db_size	owner	dbid	created	status	compatibility_level
1	Accounting	15.00 MB	ELTP	485	Jun 3 2008	Status=ONLINE, Updateability=READ_WRITE, Us...	80

	name	fileid	filename	filegroup	size	maxsize	growth	usage
1	Accounting	1	C:\Program Files\Microsoft SQL Server\MSSQ...	PRIMARY	10240 KB	51200 KB	5120 KB	data only
2	AccountingLog	2	C:\Program Files\Microsoft SQL Server\MSSQ...	NULL	5120 KB	25600 KB	5120 KB	log only

Figure 1.4-2:Scheme of the created table



Context :

- To understand the complete physical architecture of the SQL Database.
- To create a Database with given specifications.



Practice Session/s :

- Create a Database 'Example' with size 10MB, maximum size 35MB and file growth 5MB in the Database northwind.



Check List :

- Different types of Databases.
- Different types of Database Models.
- How to create a Database.
- 'Name' is the logical name that the server uses internally to refer the file.
- 'Filename' is the physical name on the disk that stores the actual data.



Common Error/s :

- After creation of a Database, only the database owner and the system Administrator can access it.
- Always include a value for Maximum size.



Exception/s :

- The 'Create' statement is used to create objects on the local server only.
- Database name should be unique.



Lesson/s Learnt :

- ☒ Understand the different types of Databases.
- ☒ How to create a Database.



Best Practice/s:

- Understand the different types of databases and which one is to be used while designing Database.

Crossword: Unit-1

Estimated Time: 10 mins.

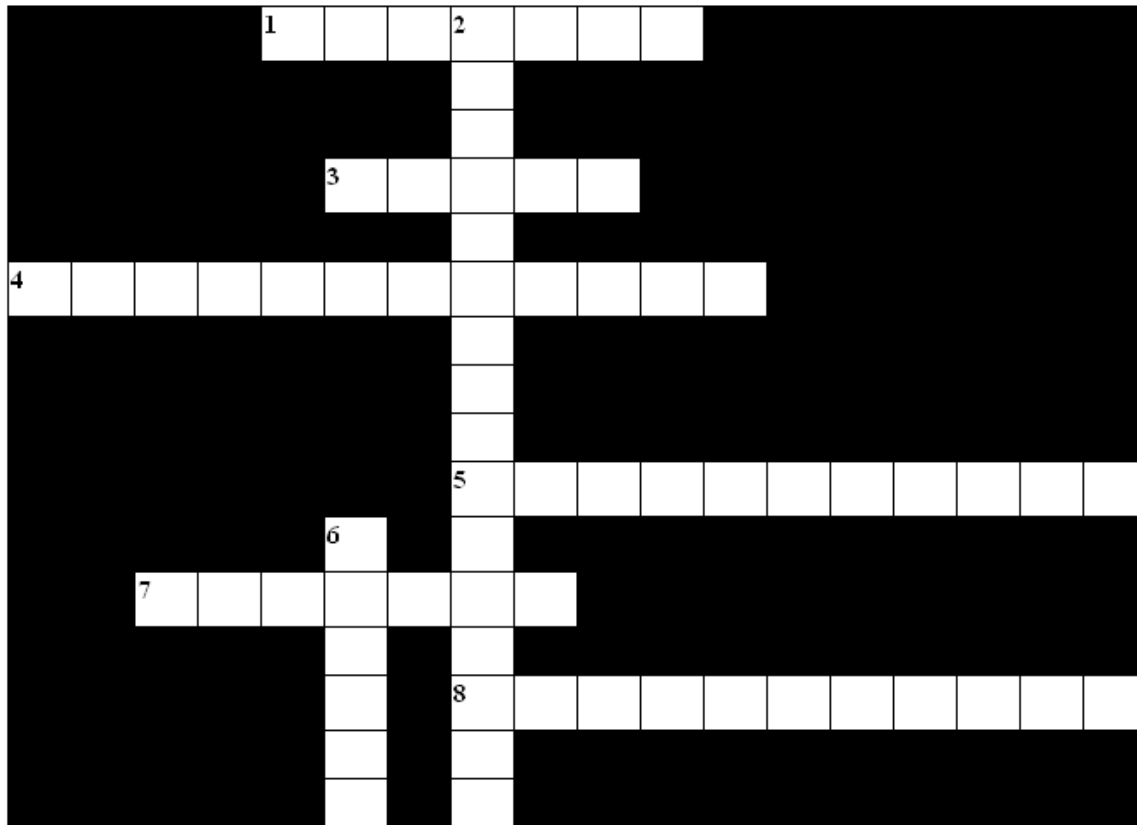


Fig.1-1

Across:





1. Every database has a compulsory file which contain the startup information for the database is called as(7)
3. What kind of result is created and watched by SQL Server Profiler.(5)
4. What are two layers distributed by three components in Two-Tier Architecture.(12)
5. During the course of normal operations, SQL Server utilizes a log to track all of the modifications performed within a database called as(11)
7. What is the form which represents information in an Object Oriented Database.(7)
8. Which Business Intelligence Studio can help to develop projects as part of a solution that is independent of any particular server.(11)

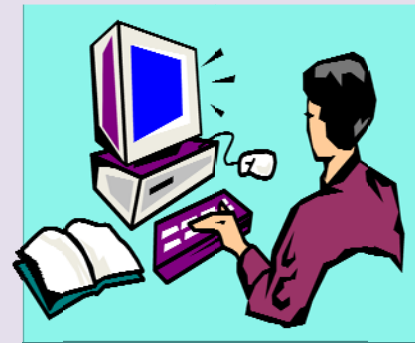
Down:

2. What is the integrated environment for accessing, configuring, managing, administering, and developing all components of SQL Server.(16)
6. What is the name of database used by SQL Server to store temporary tables and temporary stored procedures.(6)

2.0 Using Transact-SQL

Topics

-  2.1 Exploring Transact-SQL
-  2.2 Querying Data
-  2.3 Changing Data Through Queries
-  2.4 Crossword





Topic: Exploring Transact-SQL

Estimated Time – 25 mins.



Objectives : At the end of the activity, the participant should be aware of

- Data types
- Operators used in the SQL Server



Presentation :

- In Microsoft SQL Server, each object (such as column, variable, or parameter) has a related data type, which is an attribute that specifies the type of data that the object can hold.
- SQL Server 2000 ships with 26 built-in (system) Data Types.
- Data Types and their categories which are used in the SQL Server are shown in the table below:

Class	Data Type Name	Size in Bytes
Integer	Bit	1
	Bigint	8
	Int	4
	smallInt	2
	TinyInt	1
Numeric	Decimal/Numeric	Varies
	Float (Approx. Numeric)	Varies
Money	Money	8
	SmallMoney	4
Date/Time	DateTime	8
	SmallDateTime	4
Special Numeric	Cursor	1
	Timestamp/rowversion	8
	UniqueIdentifier	16
Character	Char	Varies
	Varchar	Varies
	Text	Varies

	XML	Varies
Unicode	NChar	Varies
	NVarchar	Varies
	Ntext	Varies
Binary	Binary	Varies
	VarBinary	Varies
	Image	Varies
Other	Table	Special
	Sql_variant	Special

Figure 2.1-1: Table Of Operators

- An Operator is a symbol specifying an action that is performed on one or more expressions.
- Following are the list of Operators supported by the SQL Server:
 - Arithmetic Operator
 - Logical Operator
 - Assignment Operator
 - String Concatenation Operator
 - Bitwise Operator
 - Unary Operator
 - Comparison Operator



Scenario :

‘Fortune’ is a large hardware company dealing with the manufacturing and selling of the hardware parts of digital equipments. Mr. Brown is the manager of the sales department managing the employees in that department.

The manager wants to view the information of the employees fulfilling the following criteria:

- Title of the Employees can be either sales representatives or sales manager and
- Having salary greater than 5000 with
- Birth year between 1950 and 1970.
- The employees should be from the country United States or United Kingdom.



Demonstration / Code Snippet :

Step 1: Use the Employees table from the Northwind database. The data types of the attributes used in the table can be seen as shown below:

Use Northwind

Select * from Employees

	EmployeeID	LastName	FirstName	Title	TitleOfCo...	BirthDate	HireDate	Address
1	1	Hilton	Nancy	Sales Representative	Ms.	1948-12-0...	1992-05-01 ...	507 - 20th Ave. E. Apt. 2A
2	2	Witherspoon	Reese	Vice President, Sales	Dr.	1952-02-1...	1992-08-14 ...	908 W. Capital Way
3	3	Nightly	Janet	Sales Representative	Ms.	1963-08-3...	1992-04-01 ...	722 Moss Bay Blvd.
4	4	Maria	Margaret	Sales Representative	Mrs.	1937-09-1...	1993-05-03 ...	4110 Old Redmond Rd.
5	5	Testa	Steven	Sales Manager	Mr.	1955-03-0...	1993-10-17 ...	14 Garrett Hill
6	6	Fujiyama	Michael	Sales Representative	Mr.	1963-07-0...	1993-10-17 ...	Coventry House Miner Rd.

Figure 2.1-2: Result

Step 2: The data types of the attributes used in the table can be seen as shown below:

sp_help Employees \\To view the schema of the table.

	Column_name	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks	FixedLenNullInSource
1	EmployeeID	int	no	4	10	0	no	(n/a)	(n/a)
2	LastName	nvarchar	no	40			no	(n/a)	(n/a)
3	FirstName	nvarchar	no	20			no	(n/a)	(n/a)
4	Title	nvarchar	no	60			yes	(n/a)	(n/a)
5	TitleOfCourte...	nvarchar	no	50			yes	(n/a)	(n/a)
6	BirthDate	datetime	no	8			yes	(n/a)	(n/a)
7	HireDate	datetime	no	8			yes	(n/a)	(n/a)
8	Address	nvarchar	no	120			yes	(n/a)	(n/a)
9	City	nvarchar	no	30			yes	(n/a)	(n/a)
10	Region	nvarchar	no	30			yes	(n/a)	(n/a)
11	PostalCode	nvarchar	no	20			yes	(n/a)	(n/a)
12	Country	nvarchar	no	30			yes	(n/a)	(n/a)
13	HomePhone	nvarchar	no	48			yes	(n/a)	(n/a)
14	Extension	nvarchar	no	8			yes	(n/a)	(n/a)
15	Photo	image	no	16			yes	(n/a)	(n/a)
16	Notes	ntext	no	16			yes	(n/a)	(n/a)
17	ReportsTo	int	no	4	10	0	yes	(n/a)	(n/a)
18	PhotoPath	nvarchar	no	510			yes	(n/a)	(n/a)
19	salary	money	no	8	19	4	yes	(n/a)	(n/a)

Figure 2.1-3: Table Schema

Step 3: Execute the following query to get the suitable result according to the scenario

```
Select EmployeeID, Lastname, Firstname,
        Title, Titleofcourtesy, BirthDate,
        City, Country, Salary FROM Employees WHERE
        Title LIKE 'Sales%' and
        Country IN ('USA', 'UK') and
        Year(Birthdate) between '1950' and '1970' and
        Salary >= 5000
```

Note: Sales% indicates Title to be in {Sales Manager, Sales representative, Sales executive}

Year is a function that returns year part of a specified date.

	EmployeeID	Lastname	Firstname	Title	Titleofcourtesy	BirthDate	City	Country	Salary
1	5	Testa	Steven	Sales Manager	Mr.	1955-03-04 00:00:00.000	London	UK	5000.00
2	3	Nightly	Janet	Sales Representative	Ms.	1963-08-30 00:00:00.000	Kirkland	USA	5000.00

Figure 2.1-4: Result



Context :

- Data types are used to define variables appropriately.



Practice Session/s :

- Identify data types of the attributes for the following tables from the Northwind data base:
 - Orders
 - Customers
 - Suppliers
- Identify the situations in which the 'bigint' and 'smallint' data types are used.
- Identify the situations in which 'varchar' and 'Nvarchar' are used.



Check List :

- The different data types used in SQL Server.
- The use of the data types to store different values.
- The use of operators used across the queries executed in the SQL Server.



Common Error/s :

- Confusion amongst similar kinds of data types like 'char, varchar and nvarchar.'
- Assigning values to the variables according to their data types and sizes.



Exception/s :

- Attempting to store values to a variable against their defined Data Types.
- SQL Server has no concept of unsigned numeric Data Types; instead, refer to check constraint to be used.



Lesson/s Learnt :

- ☑ The various Data Types and their size in bytes.
- ☑ The categories into which the Data Types grouped.
- ☑ The concept of Operators and their use in different situations.



Best Practice/s :

- Apply appropriate Data Types.
- Distinguish the types of Operators before being applying.



Topic: Querying Data

Estimated Time – 20 mins.



Objectives : On completion of the activity, the participant should be able to:

- Explain the concept and structure of **Select** statement.
- Sort the details of the output using the **Order by** clause.
- Group the data according to requirement.
- Identify the different types of aggregate functions used in SQL Server.



Presentation :

- **Select** statement queries data from tables in the database. It has three clauses:
 - **Select** clause specifies the table column/s that are retrieved.
 - **From** clause specifies the table/s that are accessed.
 - **Where** is an optional clause which specifies the table rows retrieved.
- **Select** statement also supports following optional clauses:
 - **Order By** is used to order /sort the data sets retrieved from a SQL database.
 - **Group By** is used to collect data across multiple records and group the results by one or more columns.
 - **Having** is used to specify a search condition for a group or an aggregate.
- **Set Functions** are special summarizing functions used with Grouping and Aggregate Queries:
 - **Top** clause permits us to specify how many rows to return.
 - **Computes** clause generates the sum that appears as additional summary columns at the end of the result set.
 - **Compute By** clause allows us to see both, detail and summary of rows with one Select statement.
- The different aggregate functions used in SQL Server are as shown in the following:
 - **Sum** returns the sum of the values of a particular column.
 - **Avg** computes the average of the values of the specified column.
 - **Min / Max** functions returns the minimum or maximum value in the specified column
 - **Count (Expression|*)** is about counting the rows in a query.



Scenario :

Mc-Donald's is a major player in the market in the food beverage industry and registers huge sales on per day basis. Mr. Mickey working as a manager wants to send the list of number of employees from a particular country having same salary structure and salary greater than 4000 to the finance department, so that proper tax deduction can be made from their salary.



Demonstration / Code Snippet :

Step 1: Use the Employees table from the northwind database.

```
use northwind
```

```
select orderid,shipcity,freight from orders where shipcity='bern'
```

	orderid	shipcity	freight
1	10254	Bern	22.98
2	10370	Bern	1.17
3	10519	Bern	91.76
4	10731	Bern	96.65
5	10746	Bern	31.43
6	10966	Bern	27.19
7	11029	Bern	47.84
8	11041	Bern	48.22

Figure 2.2-1: Result

```
select shipcity,freight
from orders where shipcity='bern' group by freight,shipcity
```

	shipcity	freight
1	Bern	1.17
2	Bern	22.98
3	Bern	27.19
4	Bern	31.43
5	Bern	47.84
6	Bern	48.22
7	Bern	91.76
8	Bern	96.65

Figure 2.2-2: Result

```
select shipcity,freight
```

```
from orders where shipcity='bern' group by freight,shipcity
having freight>30
```

	shipcity	freight
1	Bern	31.43
2	Bern	47.84
3	Bern	48.22
4	Bern	91.76
5	Bern	96.65

Figure 2.2.3: Result

```
Select shipcity,shipcountry, Count(*) as 'No. of Employees' From
orders
group by shipcountry,shipcity having shipcity in('bern','lyon')
Order By shipcountry
```

//count (*) is used to count the no. of rows.

	shipcity	shipcountry	No. of Employees
1	Lyon	France	10
2	Bern	Switzerland	8

Figure 2.2-4: Result



Context :

- To retrieve data from table/s in the database using Select statement.
- To put condition/s on the data retrieval with WHERE clause.
- To retrieve data in either ascending or descending order.
- To collect data from multiple records; HAVING clause is used to put some check on the retrieved data.



Practice Session/s :

- Write a query to find the number of employees from 'employees' table present in the northwind database.
- Select all employees from 'employees' whose salary lies between Rs.4000 and Rs.5000.



Check List :

- Retrieval of data from the table/s of the database based on certain conditions.
- Use of different clauses, set and aggregate functions.



Common Error/s :

- Incorrect sequencing of clauses.
- “Select * from emp group by job” is invalid, because columns selected in **Select** clause must be there in **Group By** clause also.
- An aggregate function can't be used with **Where** clause
For e.g. Where sum (sal)>50000 //invalid
- Retrieving data from a table which is present in another Database.
- **Distinct** clause has to be written first in the column list of a select statement.
- The **Having** clause is used only if there is a **Group by** clause in the query.



Exception/s :

- Retrieval of a record from a column that is not found in the table.
- Count returns the number of rows present in a table including all null rows.



Lesson/s Learnt :

- ☒ Clauses used must be in correct sequence.
- ☒ Should use respective Database before working on the table.
- ☒ Be able to filter data using **where** and **having** clauses.



Best Practice/s :

- Using alias names for the columns with **Compute** clause.



Topic: Changing Data through Query

Estimated Time – 35 mins.



Objectives : At the end of the activity, the participant should know :

- The different ways of inserting data into a table.
- To update existing data in a table.
- To delete the data from a table.
- How to drop a table including its structure.
- The difference between drop, delete and truncate statements.



Presentation :

- **Insert** statement adds one or more records to a single table in a relational database.
- **Update** statement updates the data of one or more records in a table.
- **Delete** statement is used to delete an entire row from a table supplying different conditions to the WHERE clause.
- **Truncate** clears all rows of a table without deleting the table itself.
- **Drop** command deletes the structure of the table from the database.



Scenario :

‘Gold Gym’ is a well known body toning institute in the city of New York. It provides these services to the customers; Yoga, Aerobics, Body Building and Medicated therapy. Mr. James Kooky (Owner of the Gym) wants to offer a new service of Weight -loss to its customers.



Demonstration / Code Snippet :

Step 1: Use the existing table ‘gymdata’ from the Northwind database.

Use northwind

Select * from gymdata

	Services	Duration	fees	Timings
1	Aerobics	4 months	5000	4-6 AM
2	Body Building	6 months	10000	4-8 PM
3	Medicated Therapy	2 months	20000	5-9 PM
4	Yoga	3 months	3000	7-9 AM

Figure 2.3-1: Result

Step 2: Before inserting rows into a table, always view the structure of the table to identify data type mismatch with the column data.

SP_HELP gymdata

	Column_name	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks	FixedLenNullInSource
1	services	varchar	no	30			yes	no	no
2	duration	varchar	no	30			yes	no	no
3	fees	money	no	8	19	4	yes	(n/a)	(n/a)
4	timing	varchar	no	20			yes	no	no

Figure 2.3-2: Result

Step 3: Insert a row into the table in the database.

```
Insert into gymdata (Services,Duration,fees,Timings)
values ('weight loosing','4 months',4000,'7-8 AM')
Select * from gymdata
```

	Services	Duration	fees	Timings
1	Aerobics	4 months	5000	4-6 AM
2	Body Building	6 months	10000	4-8 PM
3	Medicated Therapy	2 months	20000	5-9 PM
4	weight loosing	4 months	4000	7-8 AM
5	Yoga	3 months	3000	7-9 AM

Figure 2.3-3: Result

Step 4: Update the Duration of the yoga course to 4 months which is a 3 month course

```
.Update gymdata Set Duration ='4 months' where services='yoga'
```

```
Select * from gymdata
```

	Services	Duration	fees	Timings
1	Aerobics	4 months	5000	4-6 AM
2	Body Building	6 months	10000	4-8 PM
3	Medicated Therapy	2 months	20000	5-9 PM
4	weight loosing	4 months	4000	7-8 AM
5	Yoga	4 months	3000	7-9 AM

Figure 2.3-4: Result Updated value

Step 5: Delete the details from the table with aerobics services.

```
Delete from gymdata where services ='aerobics'
```

The table appears as follows:

	Services	Duration	fees	Timings
1	Body Building	6 months	10000	4-8 PM
2	Medicated Therapy	2 months	20000	5-9 PM
3	weight loosing	4 months	4000	7-8 AM
4	Yoga	4 months	3000	7-9 AM

Figure 2.3-5: Result



Context :

- Insert statement populates a table with data.
- The existing records can be changed or altered using the Update statement.
- A single record or the entire table can be deleted from the database.



Practice Session/s :

- Insert the Employee ID, Title, and Hire Date of a new employee in the 'employee' table (Use Northwind).
- Update the salary of female associate to \$6000.
- Delete the records of Mr. Michael Fujiyama, as he has left the Organization.



Check List :

- Inserting data into table/s and updating.
- Delete row/s from the table.
- Delete the structure of the table.



Common Error/s :

- Unable to distinguish between DELETE and TRUNCATE command.
- UPDATE query must be followed by WHERE Clause, which is not followed.
- Using improper syntax while writing queries.



Exception/s :

- While inserting data in a table, if the number of values to be inserted doesn't match the number of columns, it causes an exception.
- Insertion of data into a table can be done by getting the values from another table from different database but in the same server.



Lesson/s Learnt :

- ☒ Insert command cannot be used to store more than one record at a time.
- ☒ Update command cannot be used on record which does not exist.



Best Practice/s :

- Use suitable Commands for different operations on the tables of the Database.

Crossword: Unit-2

Estimated Time: 10 mins.

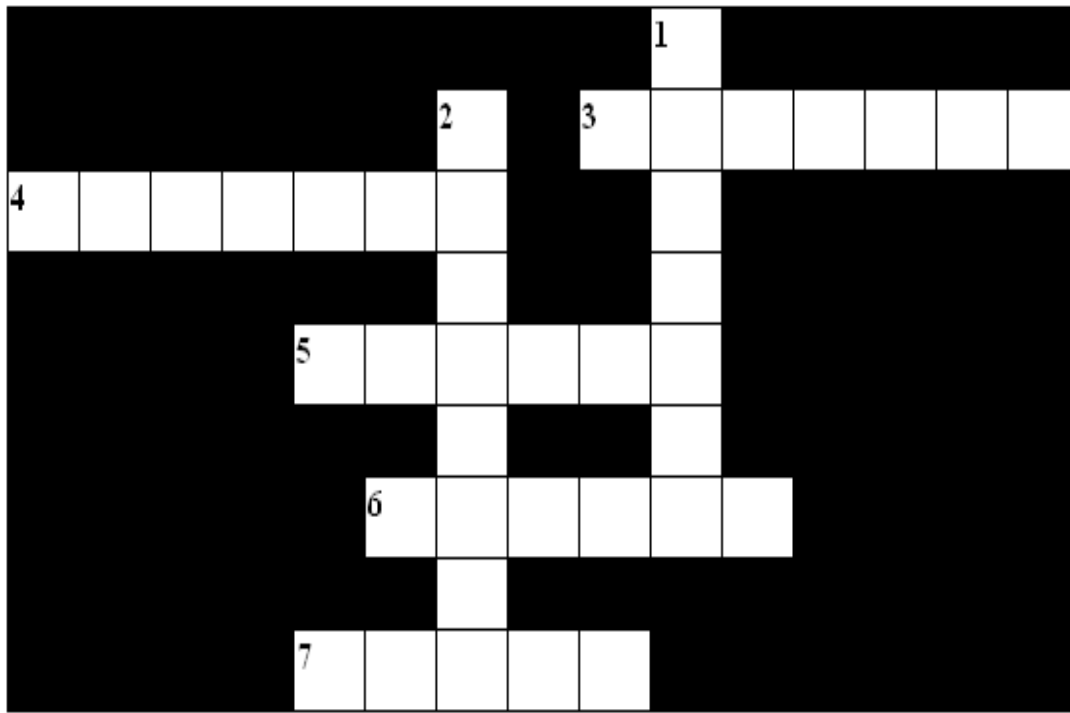


Fig.2-1

Across:







3. Which clause can be used in row aggregate functions? (7)
4. Which Keyword can be used for delays or to block execution until the set time? (7)
5. How many tables can be updated using INSERT statement in a single instance? (6)
6. Which clause can be used to specify a search condition for a group or aggregate? (6)
7. Which keyword is used within the while loop stop the execution of statements in T-SQL. (5)

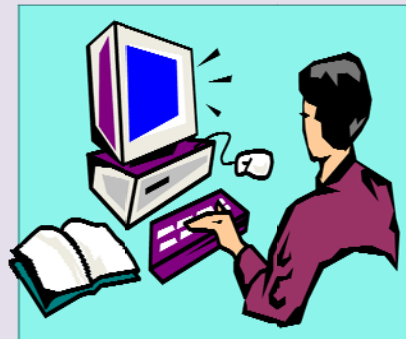
Down:

1. What type of result can be expected by IF statement in T-SQL. (7)
2. Which clause can be used to clear all rows of a table without deleting the table itself? (8)

3.0 Handling Complex Queries

Topics

-  3.1 Joining Multiple Tables
-  3.2 Create and Alter table
-  3.3 Constraints
-  3.4 Implementing Sub Queries
-  3.5 Derived Tables
-  3.6 Crossword





Topic: SQL Joins

Estimated Time – 35 mins.



Objectives : On completion of the activity, the participant should know

- The concept of Joins and its uses.
- The different types of Joins.



Presentation :

- The **Join** clause combines columns of one table to that of another to obtain a view of single table.
- **Types of Join**
 - INNER JOIN
 - OUTER JOIN (Left join & Right join)
 - FULL JOIN
 - CROSS JOIN
 - SELF JOIN (A Join to itself)
- **INNER JOIN** combines the records from two tables (based on the Join-predicate).
 - Inner Join can be classified further, as **Equi-Joins**: Uses only equality comparisons in the Join-predicate. **Non-Equijoin** links two or more tables based on a specified column value not equaling a specified column value in another table.
 - **Natural Join**: Predicates implicitly by comparing all columns in both tables that have the same column-name in the Joined tables.
- **OUTER JOIN** does not require each record in the two joined tables to have a matching record in the other table. The joined table retains each record—even if no other matching record exists.
- It subdivides further into :
 - **Left Outer Joins**: **Left Outer Join** for tables 'A' and 'B' always contains all records of the 'left' table (A), even if the join-condition does not find any matching record in the 'right' table (B).
 - **Right Outer Joins**: It closely resembles a Left Outer Join, except with the tables reversed.
- **Full Join** combines the results of both left and right outer joins and fills in NULLs for missing matches on either side.

- **Cross Join** returns the Cartesian product of the sets of records from the two joined tables.
- **Self-Join** can be used to simplify nested SQL queries, where the inner and outer queries reference the same table.



Scenario :

‘20th Century Productions’ is one of the largest movie production companies. The manager, Ms. Linda manages the details of actors respective to the films they have worked for. The manager has to produce different reports for the statistical agenda. So, the different reports can be produced using the various join types as shown in the demonstration.



Demonstration/Code Snippet :

Step 1: Consider the two tables ‘Films’ and ‘Actors’ from the database ‘Northwind’ respectively.

```
use Northwind
Select * From Films
Select * From Actors
```

	FilmID	FilmName	YearMade
1	1	My Fair Lady	1964
2	2	Unforgiven	1992

Figure 3.1-1:Result

	FilmID	FirstName	LastName
1	1	Rex	Harrison
2	1	Audrey	Hepburn
3	2	Clint	Eastwood
4	5	Humphrey	Bogart

Figure 3.1-2:Result

Step 2: To find the actors details who worked for the films in the Films table, one can use the inner join query as shown below.

-- ANSI SYNTAX

```
Select * from Films F INNER JOIN Actors A on
F.Filmid = A.Filmid
```

-- WHERE CLAUSE SYNTAX

```
Select * from Films F, Actors A where F.Filmid = A.Filmid
```

The output of the above query is shown in the below table:

	FilmID	FilmName	YearMade	FilmID	FirstName	LastName
1	1	My Fair Lady	1964	1	Rex	Harrison
2	1	My Fair Lady	1964	1	Audrey	Hepburn
3	2	Unforgiven	1992	2	Clint	Eastwood

Figure 3.1-3:Result

Step 3: To find all the details from the Films and Actors table, one can fire the full join query.

```
Select * from Films F FULL JOIN Actors A on
F.Filmid = A.Filmid
```

	FilmID	FilmName	YearMade	FilmID	FirstName	LastName
1	1	My Fair Lady	1964	1	Rex	Harrison
2	1	My Fair Lady	1964	1	Audrey	Hepburn
3	2	Unforgiven	1992	2	Clint	Eastwood
4	NULL	NULL	NULL	5	Humphrey	Bogart

Figure 3.1-4:Result

Step 4: To find all the details from the Films table and the actors working in those films, one can fire the left outer join query.

--ANSI SYNTAX

```
Select * from Films F LEFT OUTER JOIN Actors A on
F.Filmid = A.Filmid
```

-- WHERE CLAUSE SYNTAX

```
Select * from Films F, Actors A where F.Filmid != A.Filmid
```

	FilmID	FilmName	YearMade	FilmID	FirstName	LastName
1	1	My Fair Lady	1964	1	Rex	Harrison
2	1	My Fair Lady	1964	1	Audrey	Hepburn
3	2	Unforgiven	1992	2	Clint	Eastwood

Figure 3.1-5:Result

Similarly, to find all the details from the actors table and the films associated with those actors, one can fire the right outer join query.

--ANSI SYNTAX

```
Select * from Films F RIGHT OUTER JOIN Actors A on
```

```
F.Filmid = A.Filmid
-- WHERE CLAUSE SYNTAX
Select * from Films F, Actors A where F.Filmid =* A.Filmid
```

Step 5: To find all possible combinations from the actors and films, one can use the cross join query as shown below.

```
--ANSI SYNTAX
Select * from Films F CROSS JOIN Actors A

-- WHERE CLAUSE SYNTAX
Select * from Films F, Actors A
```

	FilmID	FilmName	YearMade	FilmID	FirstName	LastName
1	1	My Fair Lady	1964	1	Rex	Harrison
2	2	Unforgiven	1992	1	Rex	Harrison
3	1	My Fair Lady	1964	1	Audrey	Hepburn
4	2	Unforgiven	1992	1	Audrey	Hepburn
5	1	My Fair Lady	1964	2	Clint	Eastwood
6	2	Unforgiven	1992	2	Clint	Eastwood
7	1	My Fair Lady	1964	5	Humphrey	Bogart
8	2	Unforgiven	1992	5	Humphrey	Bogart

Figure 3.1-6:Result



Context :

- To obtain the details from multiple tables having at least one common attribute.



Practice Session/s :

- Obtain the last name of the employees who have given orders having order Id=10249 from the table 'employees' and 'orders' in 'Northwind' database.
- Select product name, company name, form products and suppliers where units in stock are more than 30.



Check List :

- Tables should have, at least one common attribute on which the Join is performed.
- Only records matching, the common attribute can be viewed.



Common Error/s :

- Common Attributes are generally not prefixed with aliases.



Exception/s :

- Joining tables with no matching columns throws an exception



Lesson/s Learnt :

- ☑ Tables to be joined should have at least one common attribute.
- ☑ Inner Joins are exclusive in nature while outer and full joins are inclusive.
- ☑ The joins can be written using both syntaxes, ANSI syntax and using **where** clause.



Best Practice/s :

- Use aliases to join the tables.
- Joins are not preferred because of the excessive memory wastage.



Topic: Create and Alter Table

Estimated Time – 20 mins.



Objectives : On completion of the activity, the participant should know how to:

- Create a table.
- Alter the existing columns in a table.
- Add columns to a table.



Presentation :

- Tables are the basic structure that store data in the database.
- **Create** command is used to create a table within a Database.
- **Alter** Command is used to alter the schema of an existing Table or Database.



Scenario :

FORTUNE is a big Software Training Institute which offers different courses. Mr. Donald is the administrator of the company maintaining the details of the trainees by creating table 'trainee'. The table holds the First Name, Last Name, Address, Age and Course of the trainee.



Demonstration/Code Snippet :

Step 1: Create a table named trainee having certain attributes like first name, last name, address, age and course having requisite data types:

Use Northwind

```
If Exists (select * from sysobjects where name = 'trainee')
begin
Drop table dbo.trainee
end
```

```
Go
create TABLE dbo.trainee (traineeid int identity(1001,1),
FirstName varchar(15),
LastName varchar(15),
Address varchar(15),
Age int,
course varchar(15))
```

Note: Identity column generates the column value automatically. Here, the value starts with integer 1001 and increases by 1 for each row inserted.

Step 2: The structure of the table trainee appears as follows:

`sp_help trainee` //used to see the structure of the table.

	Column_name	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks	FixedLenNullInSource	Collation
1	traineeid	int	no	4	10	0	no	(n/a)	(n/a)	NULL
2	FirstName	varchar	no	15			yes	no	no	SQL_Latin1_Ge
3	LastName	varchar	no	15			yes	no	no	SQL_Latin1_Ge
4	Address	varchar	no	15			yes	no	no	SQL_Latin1_Ge
5	Age	int	no	4	10	0	yes	(n/a)	(n/a)	NULL
6	course	varchar	no	15			yes	no	no	SQL_Latin1_Ge

Figure 3.2-1:Result

Mr. Donald wishes to keep the record of the Date of Birth of the trainees in the 'trainee' table. Add one more column 'dateofbirth' using Alter Command.

Step 3: As the rows are inserted, it is not required to enter the values for identity column that is 'traineeid'. The table appears as follows:

`Select * from trainee`

	traineeid	FirstName	LastName	Address	Age	course
1	1001	Tom	Alter	Chicago	21	C++
2	1002	John	Carter	New York	22	Java

Figure 3.2-2:Result

Step 4: Alter the table as now the management wants to add another attribute name 'dateofbirth'.

`Alter table trainee ADD dateofbirth datetime null`

`sp_help trainee`

	Column_name	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks	FixedLenNullInSource	Collation
1	traineeid	int	no	4	10	0	no	(n/a)	(n/a)	NULL
2	FirstName	varchar	no	15			yes	no	no	SQL_Latin1_G
3	LastName	varchar	no	15			yes	no	no	SQL_Latin1_G
4	Address	varchar	no	15			yes	no	no	SQL_Latin1_G
5	Age	int	no	4	10	0	yes	(n/a)	(n/a)	NULL
6	course	varchar	no	15			yes	no	no	SQL_Latin1_G
7	dateofbirth	datetime	no	8			yes	(n/a)	(n/a)	NULL

New Coloumn Added

Figure 3.2-3:Result

Step 5: After the alter command, the table appears as below:

	traineeid	FirstName	LastName	Address	Age	course	dateofbirth
1	1001	Tom	Alter	Chicago	21	C++	NULL
2	1002	John	Carter	New York	22	Java	NULL

Figure 3.2-4:Result



Context :

- To create a new table.
- Alter table is used to Insert or Delete any column from a table.



Practice Session/s :

- Write a query to drop the column 'dateofbirth' from the table 'trainee' and a query to add another column named ' phone no' having data type as 'varchar' in the table 'trainee' of northwind database.
- Differentiate between Drop, Delete and Truncate commands.



Check List :

- Creating a table
- Deleting or adding column/s in the table.



Common Error/s :

- Use of improper data types while creating a new table or altering an existing table.
- Not providing data according to the data types of the attributes



Exception/s :

- Altering a column of a table, having pre-defined constraints, results in an exception.



Lesson/s Learnt :

- ☒ To create a new table.
- ☒ To insert new column/s into an existing table.
- ☒ To alter the size of the existing column/s in a table.



Best Practice/s :

- Table should be created with name of administrator for better understanding.



Topic: Constraints

Estimated Time – 25 mins.



Objectives : At the end of the, the participant should understand:

- The importance data integrity and their categories.
- The concept of constraints.
- The types of different constraints available in SQL Server.
- Usefulness of the constraints.



Presentation :

- A constraint is a property assigned to a column or the set of columns in a table that prevents certain types of inconsistent data values from being placed in the column(s).
- The following categories of the data integrity exist:
 - ENTITY INTEGRITY ensures that there are no duplicate rows in a table.
 - DOMAIN INTEGRITY enforces valid entries for a given column by restricting the type, the format, or the range of possible values.
 - REFERENTIAL INTEGRITY ensures that rows cannot be deleted, which are used by other records (for example, corresponding data values between tables will be vital).
 - USER-DEFINED INTEGRITY enforces some specific business rules that do not fall into entity, domain, or referential integrity categories.
- Microsoft SQL Server supports the following constraints:
 - PRIMARY KEY constraint is a unique identifier for a row within a database table.
 - UNIQUE also enforces unique identifier. The only difference between primary and unique key is we can have more than one unique key in a table
 - FOREIGN KEY constraint prevents any actions that would destroy link between tables with the corresponding data values
 - CHECK constraint is used to limit the values that can be placed in a column
 - NOT NULL constraint enforces that the column will not accept null values
- DROPPING constraint drops constraints in an existing table by using the ALTER TABLE statement.
- A Constraint can be implemented in two ways.

- **TABLE LEVEL CONSTRAINTS:** - Specifies a limitation for a table that restricts the values that the table can store.

- ✓ **UNIQUE** (column_name {, column_name}) [WITH constraint_with_clause]
- ✓ **PRIMARY KEY** (column_name {, column_name}) [WITH constraint_with_clause]
- ✓ **FOREIGN KEY** (column_name {, column_name})
- ✓ **REFERENCES** [schema.]table_name [(column_name {, column_name })][WITH constraint_with_clause]

Example:

```
Create table yourtable (firstname char (20) not null,  
                        lastname char (20) not null,  
                        unique (firstname, lastname));
```

- **COLUMN LEVEL CONSTRAINTS:-**

- ✓ **UNIQUE** [WITH constraint_with_clause]
- ✓ **PRIMARY KEY** [WITH constraint_with_clause]
- ✓ **REFERENCES** [schema.]table_name[(column_name)] [WITH constraint_with_clause]

Example:

```
CREATE TABLE supplier_item  
(  
    Supp_no INTEGER NOT NULL PRIMARY KEY,  
    Item_no INTEGER NOT NULL,  
    qty INTEGER NOT NULL DEFAULT 0 ;  
)
```



Scenario :

SUVEN PHARMACY is a large company having several customers. The administrator of the company wants to display a list of all the customers details. The database contains a table named **dbo.customers**. The column values in the customers' table has to follow some constraints as mentioned below:

- The customerID has to be unique and null values are not allowed.
- The city column should display as 'unknown' by default if no data is provided.
- Either the phone or Fax should be entered.
- The company name has to be unique.



Demonstration/Code Snippet :

Step 1: The below code helps to add a default constraint to the column, City of the Customers table from Northwind database. The default value is 'Unknown'.

```
USE Northwind
ALTER TABLE dbo.Customers
ADD CONSTRAINT DF_City DEFAULT 'UNKNOWN' FOR City
```

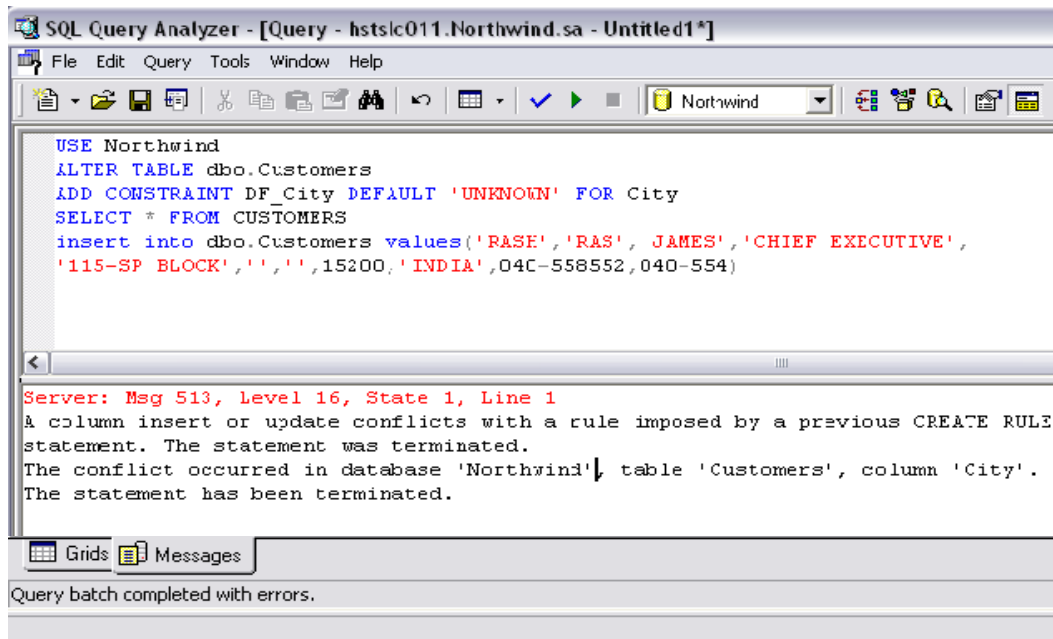


Figure 3.3-1:Output showing error has occurred.

Step 2: The below code adds a primary key constraint to the CustomerID column of the Customers table.

```
USE Northwind
ALTER TABLE dbo.Customers
ADD CONSTRAINT PK_Customers
PRIMARY KEY NONCLUSTERED (CustomerID)

Insert dbo.Customers
values ('ABCD', 'SATYAM', 'SIRI', 'Salesrepresentative', '120
Hanover Sq.', 'DEFAULT', 'NULL', '508207', 'INDIA',
040-32921690, '030-12345')
```

A row is then added to the table.

Step 3: A 'check' constraint is added to two columns namely, Fax and Phone to be not null


```
USE Northwind
ALTER TABLE dbo.Customers
ADD CONSTRAINT CK_Postal CHECK (Fax IS NOT NULL OR PHONE IS NOT
NULL)
```

Step 4: The below code shows regarding creating a unique constraint to the Company Name of the Customers table.

```
USE Northwind
ALTER TABLE dbo.Customers
ADD CONSTRAINT U_CompanyName UNIQUE NONCLUSTERED (CompanyName)
```

Step 5: The below code is to create a foreign key on the table orders which refers to the CustomerID column of the customers table. Both the tables are present in the Northwind database.

```
USE Northwind
ALTER TABLE dbo.Orders
ADD CONSTRAINT FK_Orders_Customers
FOREIGN KEY (CustomerID)
REFERENCES dbo.Customers(CustomerID)
```

Step 6: The constraints in an existing table can be dropped by using the ‘Alter Table’ statement. The following example drops the pk_customer’s primary key constraint in the Customers table:

```
ALTER TABLE employee
DROP CONSTRAINT pk_customers
```

Step 7: The constraints in a table can be disabled by using the ‘Alter Table’ statement. The following example disables all the constraints in the Customers table:

```
ALTER TABLE customers NOCHECK CONSTRAINT ALL
```



Context :

- Used when data is to be entered in the table.
- To ensure accuracy and reliability of the data in the database.
- To ensure there are no duplicate rows in a table



Practice Session/s :

- Create a ‘primary key’ named pk_course for the course column in the trainee table in the northwind database.
- Create a ‘check’ constraint in the ‘age’ column, ensure that age is not greater than 65 or less than 15.



Check List :

- To know the various types of integrity constraints.
- To know the difference between unique and primary key.



Common Error/s :

- A foreign key constraint cannot be defined to a column if the corresponding column in the parent table doesn't exist as a primary key.



Exception/s :

- While altering the column of a table having pre-defined constraints, it will throw an exception. It is not possible to alter such a column.



Lesson/s Learnt :

- ☒ When and how to use constraints.
- ☒ Importance of the primary key in a table.



Best Practice/s :

- Define constraints to a table so that redundancy is less, business logic is clear and data remains consistent.



Topic: Sub Queries

Estimated Time – 20 mins.



Objective : On completion of the activity, the participant should be aware of:

- The concept of Sub queries and Derived tables.
- The need for a Sub query.
- The clauses used in Sub queries.



Presentation :

- A Sub query is a normal T-SQL query that is nested inside other query-using parentheses.

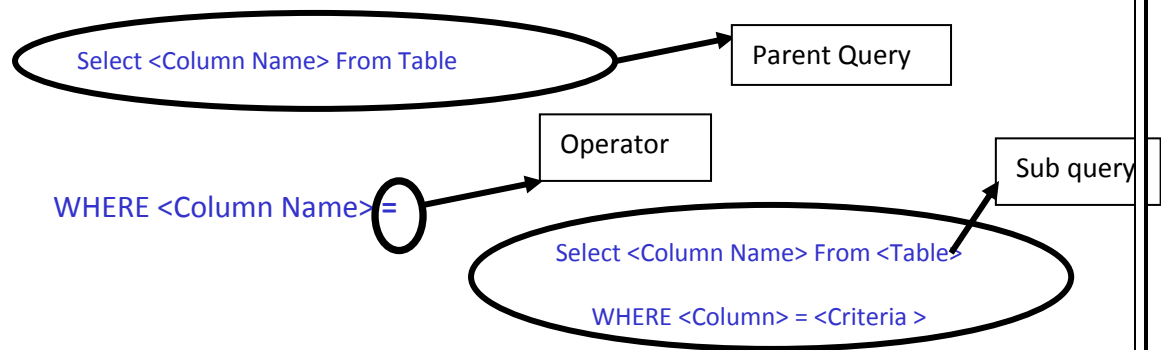


Figure 3.4-1 Structure of Sub query

- Sub query generally has to fill one of the number of needs:
 - To break a query into a series of logical steps.
 - To provide a listing to be the target of a WHERE clause together with[IN!EXISTS!ANY!ALL]
 - To provide a lookup driven by each individual record in a parent query.
- Sub query can return only one attribute.
- Different clauses in Sub queries are:
 - ANY and ALL.
 - IN and NOT IN.
 - EXISTS and DOES NOT EXIST.
- A Sub query can include one or more Sub queries. Any number of Sub queries can be nested in a statement and this is called **Multiple Levels of Nesting**.

- Many queries can be resolved by executing the Sub query once and substituting the resulting value or values into the WHERE clause of the outer query and this is known as **Correlated Sub query**.



Scenario :

E –Mall is a big shopping mall which is dealing with suppliers from different countries. The administrator of that mall always tries to update himself with the details of the different products that are imported from suppliers of different countries. In E-Mall database two major tables are Products and Suppliers. The administrator wants to know the details of the products supplied from UK. At this point, the ProductName, SupplierId, Quantity, UnitPrice, UnitsInStock of the products supplied from the UK can also be seen.



Demonstration / Code Snippet :

```
USE northwind

SELECT ProductID
       ,ProductName
       ,SupplierID
       ,QuantityPerUnit as Quantity
       ,UnitPrice
       ,UnitsInStock
FROM products AS p
WHERE EXISTS ( SELECT * FROM Suppliers AS s
               WHERE p.SupplierID = s.SupplierID
               AND s.Country = 'UK' )
```

	ProductID	ProductName	SupplierID	Quantity	UnitPrice	UnitsInStock
1	1	Chai	1	10 boxes x 20 bags	19.80	39
2	2	Chang	1	24 - 12 oz bottles	20.90	17
3	3	Syrup	1	12 - 550 ml bottles	11.00	13
4	68	Scottish Longbreads	8	10 boxes x 8 pieces	13.75	6
5	20	Sir Rodney's Marmalade	8	30 gift boxes	89.10	40
6	21	Sir Rodney's Scones	8	24 pkgs. x 4 pieces	11.00	3
7	19	Teatime Chocolate Biscuits	8	10 boxes x 12 pieces	10.12	25

Figure 3.4-2:Result



Context :

- To make complex queries simpler by breaking them into part.
- In situations of implementing self join.



Practice Session/s :

- Write a query to find the highest unit price where supplierID is the same in products and suppliers tables of Northwind database



Check List :

- To know the importance of various clauses in a Sub query.
- Understand the difference between a join and a Sub query.



Common Error/s :

- Inclusion of more than one attribute.
- Number of rows returned by the inner query needs an appropriate clause.



Exception/s :

- If the WHERE clause of an outer query includes a column name, it must be join-compatible with the column in the Sub query select list.
- The **ntext**, **text** and **image** data types are not allowed in the select list of Sub queries.



Lesson/s Learnt :

- ☒ To use Sub query.
- ☒ To use of **any**, **all**, **in**, **not in**, **exist**, **not exist** clauses.
- ☒ The use of nested Sub query.



Best Practice/s :

- If the query is too long having many parameters, terms or conditions, then it is advisable to use a sub query.



Topic: Derived Tables

Estimated Time – 25 Mins.



Objective : On completion of the activity, the participant should **know:**

- The use of Sub queries and derived tables



Presentation :

- A **Derived Table** is one that is created on-the-fly using the SELECT statement, and referenced just like a regular table or view.



Scenario :

LANDMARK is a very large book store which sells books of many authors. The administrator of the store wants to display a list of all the books with the name of their author/s with the price and the category. He has a database containing three major tables - authors, titles and titleauthor. Using derived table method he sorts out this data accurately and takes less time to do this as only those column which are required are selected. In authors table the details of authors is given. In titles table the information about the books is given. In titleauthor the relationship between author and titles using authorid and titleid only is given. The authors' first and last names with their booktitle, type and price is displayed by creating a derived table which matches the authors, titles and titleauthor tables by picking up the required columns.



Demonstration /Code Snippet :

USE pubs

```
SELECT      (a.au_fname+' '+a.au_lname) as Name
            ,a.title
            ,a.type
            ,a.price
FROM (SELECT o.au_fname
            ,o.au_lname
            ,t.title
            ,t.type
            ,t.price FROM authors as o
            join titleauthor as ta
            on o.au_id = ta.au_id
            join titles as t
            on t.title_id= ta.title_id )
AS a
```

	Name	title	type	price
1	Marjorie Green	The Busy Executive's Database Guide	business	-300.08
2	Abraham Bennet	The Busy Executive's Database Guide	business	-300.08
3	Michael O'Leary	Cooking with Computers: Surreptitious Balance Sh...	business	-364.40
4	Stearns MacFeather	Cooking with Computers: Surreptitious Balance Sh...	business	-364.40
5	Marjorie Green	You Can Combat Computer Stress!	business	-436.08
6	Dean Straight	Straight Talk About Computers	business	-300.08
7	Innes del Castillo	Silicon Valley Gastronomic Treats	mod_cook	159.92
8	Michel DeFrance	The Gourmet Microwave	mod_cook	23.92
9	Anne Ringer	The Gourmet Microwave	mod_cook	23.92
10	Cheryl Carson	But Is It User Friendly?	popular_comp	183.60
11	Ann Dull	Secrets of Silicon Valley	popular_comp	160.00
12	Sheryl Hunter	Secrets of Silicon Valley	popular_comp	160.00

Figure 3.5-1:



Context :

- Used to link Sub query to a table.
- Sub queries are used in situations where a self join or a view is used.



Practice Session/s :

- Make a query to find out the second highest value from table employees in the northwind database.



Check List :

- ☒ Importance of associating a Sub query to a table.
- ☒ Need to use any operator for Sub query.



Common Error/s :

- Non association of derived table to an alias.



Exception/s :

- Violation occurs with non updatable derived table in update statement.



Lesson/s Learnt :

- ☒ Use of derived tables in handling queries.



Best Practices/s :

- Competent in Derived tables

Crossword: Unit-3

Estimated Time: 10 mins.

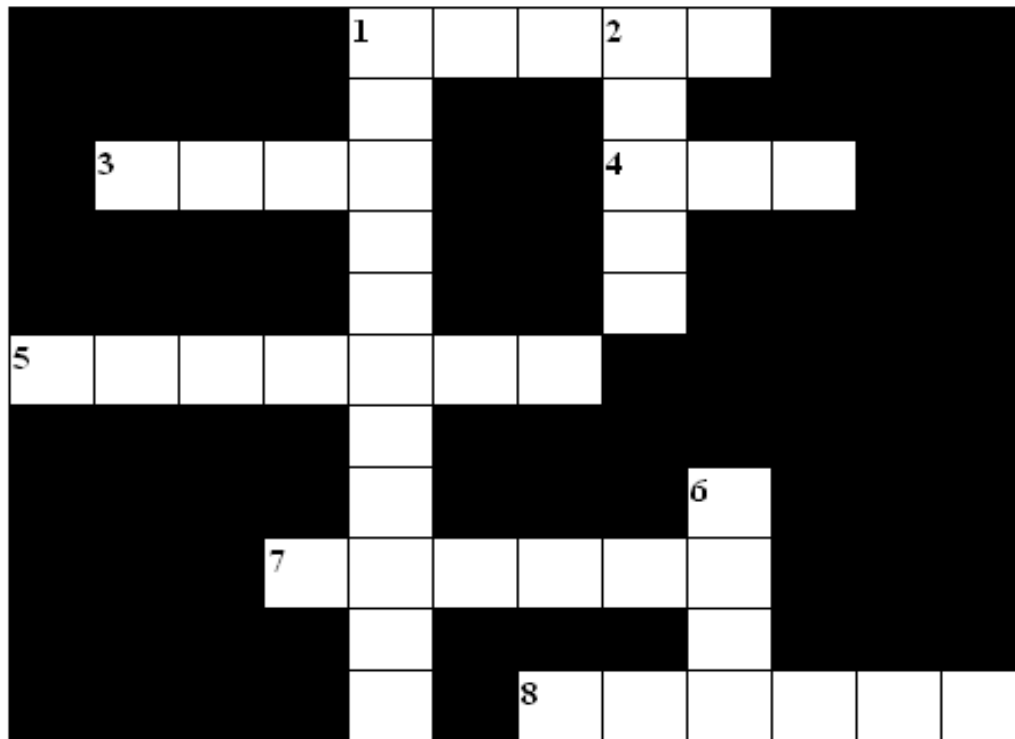


Fig.3-1

Across:


1. Which constraint is used to limit the values that can be placed in a column? (5)
3. Which clause combines columns of one table to that of another to obtain a view of single table? (4)
4. What is the maximum number of attributes that can be returned by Sub query. (3)
5. Which join is used predicate implicitly by comparing all columns in both tables that have the same column-name in the Joined tables? (7)
7. Which clause is used to enforce unique identifier and can map more than one column in a table? (6)
8. Which INTEGRITY ensures that there are no duplicate rows in a table? (6)

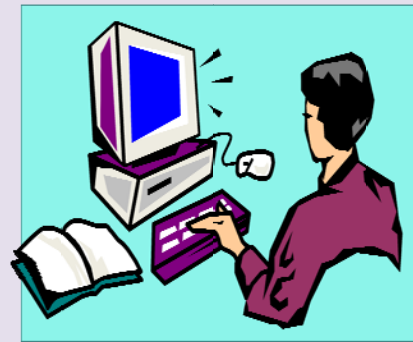
Down:

1. Which property is used to enforce data integrity that can create indexes for the table and its columns? (11)
2. Which join returns the Cartesian product of the sets of records from the two joined tables? (5)
6. Which join is used for the below statement? "Tables 'A' and 'B' always contain all records of the 'left' Table (A), even if the join-condition does not find any matching record in the 'right' Table (B)". (4)

4.0 Working With Tables

Topics

-  4.1. Defaults
-  4.2 Rules
-  4.3 Normalization
-  4.4 Relationships
-  4.5 Crossword





Topic: Rules and Defaults

Estimated Time – 25 mins.



Objectives : On completion of the activity, the participant should know

- The use of rules and defaults.
- The Difference of rules and defaults from the constraints.
- To Bind and unbind the rules and defaults to the tables.



Presentation :

- A rule allows you to specify what users can or cannot enter into a particular column or any column with a user-defined data type.
- To create a rule:
 - Use **create rule**.
 - Bind the rule to a column or user-defined data type using **sp_bindrule**.
 - Test the bound rule by inserting data.
 - Unbind the rule using **sp_unbindrule**.
- **Default** constraint creates an object called as default. When bound to a column or a user-defined data type, a default specifies a value to be inserted into the column to which the object is bound (or into all columns, in the case of a user-defined data type) when no value is explicitly supplied during an insert.
- After creating a default, use **sp_bindefault** to bind the default to a column or user-defined data type.
- Use “drop” to completely eliminate a rule or default from the database
- To determine the tables and data types that use a given rule or default, execute the stored procedure, **exec sp_depends <object name>**.



Scenario :

‘Oriental Solutions’ is a Data Warehousing company; Mr. Samuel is the database administrator. When any of the tables are populated with data, there should be some rules that have to be followed and if any of the data value is missing, there should be a default value for that.



Demonstration / Code Snippet:

Step 1: Use the table Emp_Salary from the Northwind database.

```
use Northwind
```

Step 2: Insert the data into the respective fields including a negative float value into the salary column.

```
Insert into Emp_Salary
(Account_no,Bank_name,salary,mnth,Working_days,Empid)
values(1001, 'ICICI', -121, 'feb', 21, 5)
```

The insertion query works successfully as shown in the output window.

	Account_no	Bank_Name	salary	mnth	Working_Days	Empid
1	1001	ICICI	-121	feb	21	5

Figure 4.1-1:Result

Step 3: Create a rule named 'SalaryRule' to join any column which results in a value greater than zero.

```
Use northwind
```

```
If Exists(select * from sysobjects where name = 'SalaryRule')
Begin
Drop Rule SalaryRule
End
GO
Create Rule SalaryRule as @salary > 0
```

The value of the column is verified and this value is used in the place of @salary.

Step 4: Bind the rule 'SalaryRule' to the salary column of the table Emp_Salary using the stored procedure sp_bindrule.

```
sp_bindrule 'SalaryRule', 'Emp_Salary.salary'
```

Step 5: Negative value cannot be used in the salary column of the Emp_Salary table, as the rule is bound to the salary column. It will show an error message.

```
Insert into Emp_Salary
(Account_no,Bank_name,salary,mnth,Working_days,Empid)
values(1002, 'ICICI', -121, 'feb', 21, 5)
```

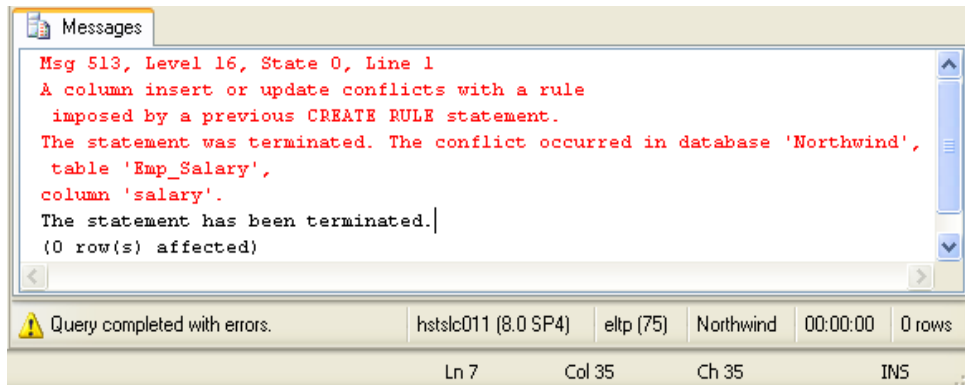


Figure 4.1-2: Output showing error has occurred.

Step 6: Unbind the rule from the column and drop the rule.

```
sp_unbindrule 'Emp_Salary.salary'
```

```
Drop Rule SalaryRule
```

Step 7: Create a default named 'SalaryDefault' to define a default of zero for salary.

```
Use northwind
```

```
If Exists(select * from sysobjects where name = 'SalaryDefault')
Begin
    Drop Default SalaryRule
End
```

```
GO
```

```
Create Default SalaryDefault as 0
```

The default is created successfully.

Step 8: Bind the default to the salary column of the Emp_Salary table.

```
sp_bindefault 'SalaryDefault', 'Emp_Salary.salary'
```

Step 9: Insert values in all the fields except in the salary column.

```
Insert into Emp_Salary
(Account_no,Bank_name,mnth,Working_days,Empid)
values (10056,'gfgf','jan',21,5)
```

After the query is executed, it is observed that the default value i.e. 0 is inserted in the salary column of the Emp_Salary table.

```
Select * from Emp_Salary
```

	Account_no	Bank_Name	salary	mnth	Working_Days	Empid
1	1001	ICICI	-121	feb	21	5
2	10056	gfgf	0	jan	21	5

Figure 4.1-3:Result



Context :

- Rules and defaults are defined independently and then bind to the tables.
- Rules and defaults are re-usable.



Practice Session/s :

Using the Northwind database, create a rule in the trainee table, with the condition that 'dateofbirth' should be greater than '03/01/1975'. Create a default name 'addressdefault' and assign value of it as 'Chicago' for the column 'address' in the 'trainee' table in the northwind database.



Check List :

- To differentiate default and default constraints.
- To identify the difference between rules and check constraint.
- Bind rules and defaults to the tables.



Common Error/s :

- Ensure the object to which the rule or default is bound is valid.
- The rules and defaults cannot be deleted or altered if they are bound to any of the data types.



Exception/s :

- Its execution is time consuming.
- Cannot be referenced across columns.
- Cannot reference other tables.



Lesson/s Learnt :

- ☑ Rules and defaults are the independent objects.
- ☑ Tables are re-usable.
- ☑ Tables are bound to data types.



Best Practice/s :

- Implement Rules and defaults to build consistency of the functionality across the database.



Topic: Normalization

Estimated Time – 35 mins.



Objectives : On completion of the activity, the participant should know

- Normalization and its types.
- How to normalize a table.
- How to create relationships.



Presentation :

- Normalization is the process of organizing data in a database. This includes creating tables and establishing relationships between those tables according to rules designed, both to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency.
- The database community has developed a series of guidelines to ensure databases are normalized. These are referred to as normal forms.
 - First Normal Form
 - Second Normal Form
 - Third Normal Form
 - Boyce-Codd Normal Form (BCNF)
 - Fourth Normal Form
 - Fifth Normal Form
 - Sixth Normal Form.
- **First Normal Form (1NF):**
 - A relation is said to be in First Normal Form (1NF) if and only if each attribute of the relation is atomic. More simply, to be in 1NF, each column must contain only a single value and each row must contain the same columns.
- **Second Normal Form (2NF):**
 - Table should meet all the requirements of the First Normal Form.
 - It further addresses the concept of removing duplicate data: each non key attribute in the relation must be functionally dependent upon the primary key.
- **Third Normal Form (3NF):**
 - Meet all the requirements of the Second Normal Form.
 - Remove columns that are not dependent upon the primary key.
- **BCNF(Boyce-Codd Normal Form):**

- BCNF tries to address situation where you have multiple overlapping candidate keys, this can happen if:
 - All candidate keys are composite keys.
 - There is more than one candidate key.
 - The candidate key should have one column that is in common with another candidate key.
- **Fourth Normal Form (4NF):**
 - A relation is in 4NF if it has no multi-valued dependencies.
- **Fifth Normal Form:**
 - Deals with non -loss and loss decompositions.
- **Sixth Normal Form:**
 - This is achieved only when you have eliminated any possibility of modification anomalies.
- **Some Facts:**
 - Redundant data wastes disk space and creates maintenance problems. If data that exists in more than one place must be changed, the data must be changed in exactly the same way in all locations. That is why redundancy should be eradicated.
 - While in inconsistency of data it is very difficult to access the data



Scenario :

Massachusetts institute of technology is a world renowned university training people in different skills. The Dean of the university Ms. Flower keeps track of the various mentors assigned to the students.

The Dean has with her a table where in the data is inconsistent and illogical, hence suffers from several problems while updating, deleting and inserting in the database. In order to eradicate this certain modifications need to be made.



Demonstration/Code Snippet :

Step 1: Following is the table available with the Dean:

```
use Northwind
```

```
Select * from univ_data
```

	studentID	Advisor	AdvRoom	Class1	class2	class3
1	1022	Jones	412	101-07	143-01	159-02
2	2322	Smith	220	201-01	211-02	214-01

Figure 4.3-1:Result

Since one student has several classes, these classes should be listed in a separate table. Fields class1, class2, and class3 in the above records are indications of design trouble.

Step 2: Create another table in first normal form by eliminating the repeating group (Class), as shown below.

/*.....Table after modifying in the 1st normal form becomes.....*/

```
Select * from univ_data1
```

	studentID	Advisor	AdvRoom	Class
1	1022	Jones	412	101-07
2	1022	Jones	412	143-01
3	1022	Jones	412	159-02
4	2322	Smith	220	201-01
5	2322	Smith	220	211-02
6	2322	Smith	220	214-01

Figure 4.3-2:Result

Step 3: Note the multiple Class values for each StudentID value in the above table. Class is not functionally dependent on StudentID (primary key), so this relationship is not in Second Normal Form. The following tables show the table now in 2NF.

```
Select* from univ_data2
```

```
Select* from univ_data3
```

	studentID	Advisor	AdvRoom
1	1022	Jones	412
2	2322	Smith	220

	studentID	Class
1	1022	101-07
2	1022	143-01
3	1022	159-02
4	2322	201-01
5	2322	211-02
6	2322	214-01
7	2322	201-01

Figure 4.3-3:Result

Step 4: In the student table above the AdvRoom column is not functionally dependent on advisor attribute and is not the primary key (studentID).

```
Select* from univ_data4
```

	studentID	Advisor
1	1022	Jones
2	2322	Smith

Figure 4.3-4:Result

```
Select* from univ_data5
```

	Advisor	AdvRoom
1	Jones	412
2	Smith	220

Figure 4.3-5:Result



Context :

- Normalization helps reduce the total amount of redundant data in the database. The less data there is, the less work SQL Server has to do, thus speeding up performance.
- Normalization helps to reduce the use of NULLS in the database. The use of NULLS in a database can reduce database performance, especially in WHERE clauses.
- Normalization reduces the number of columns in tables, which means that more rows can fit on a single data page, which boosts SQL Server read performance.



Practice Session/s :

- Design a scenario for a login module which includes a login table and a user role table. The users should have different roles and based on their role, they are given access. All the tables should be at least up to Third Normal Form. More tables can be added if required.



Check List :

- Possess knowledge of the use of normalization for enhancing the performance of SQL server.
- Be aware of the types of normalization.



Common Error/s :

- Handling the transitive dependencies in order to get the Third Normal Form.
- Not being able to distinguish between the different normal forms.



Exception/s :

- Without Normalization, there is data inconsistency in the tables and the relationships between them.



Lesson/s Learnt :

- ☒ Use of Normalization.



Best Practice/s :

- Normalize the tables created in the database to eradicate redundancy and inconsistency.



Topic: Relationships

Estimated Time – 30 mins.



Objectives : On completion of the activity, the participant should understand:

- The relationships between tables and their types.



Presentation :

- Databases maintain relationships between different data tables. This correlates data in numerous ways and ensures consistency of this data from table to table.
- **Types of Relationships**
 - One-to-One
 - One-to- Many
 - Many-to-Many
- **One-to-One relationships:**
 - This occurs when there is exactly one record in Table 'A' that corresponds to exactly one record in Table 'B'.
- **One-to-Many relationships**
 - This occurs when each record in Table 'A' has many linked records in Table 'B' but each record in Table 'B' may have only one corresponding record in Table 'A'.
- **Many-to-Many relationships**
 - This occurs when each record in Table 'A' may have many linked records in Table 'B' and vice-versa.



Scenario :

'Thomson Pvt. Limited' is a software company having a number of departments working in different domains. Each department has a manager. There are many employees working for each department. The networking and support unit work for all the departments.



Context :

- To know the role of both the tables A and B.
- To understand the boundaries and rules of the relationship, that the tables are in.



Practice Session/s :

A General Hospital consists of a number of specialized wards (such as Maternity, Pediatric, Oncology, etc). Each ward hosts a number of patients, who are admitted on the recommendation of their own GP and confirmed by a consultant employed by the Hospital. On admission, the personal details of every patient are recorded. A separate register is to be held to store the information of the tests undertaken and the results of a prescribed treatment. A number of tests are conducted for each patient. Each patient is assigned to one leading consultant but may be examined by another doctor, if required. Doctors are specialists in some branch of medicine and may be leading consultants for a number of patients, not necessarily from the same ward. Draw an E-R Model for the above scenario.



Check List :

- Understand the importance of relationships between the tables.
- Naming the relationships.
- Distinguish the types of relationship in the E-R diagram.



Common Error/s :

- Not identifying the appropriate relation type.
- Not naming the relationships.



Exception/s :

- Without relationships being defined appropriately the whole database design is difficult to understand.



Lesson/s Learnt :

- ☒ Importance of relationships.
- ☒ Define a relation between two tables.



Best Practice/s :

- In order to normalize a database, defining the relationships is a must.

Crossword: Unit-4

Estimated Time: 10 mins.

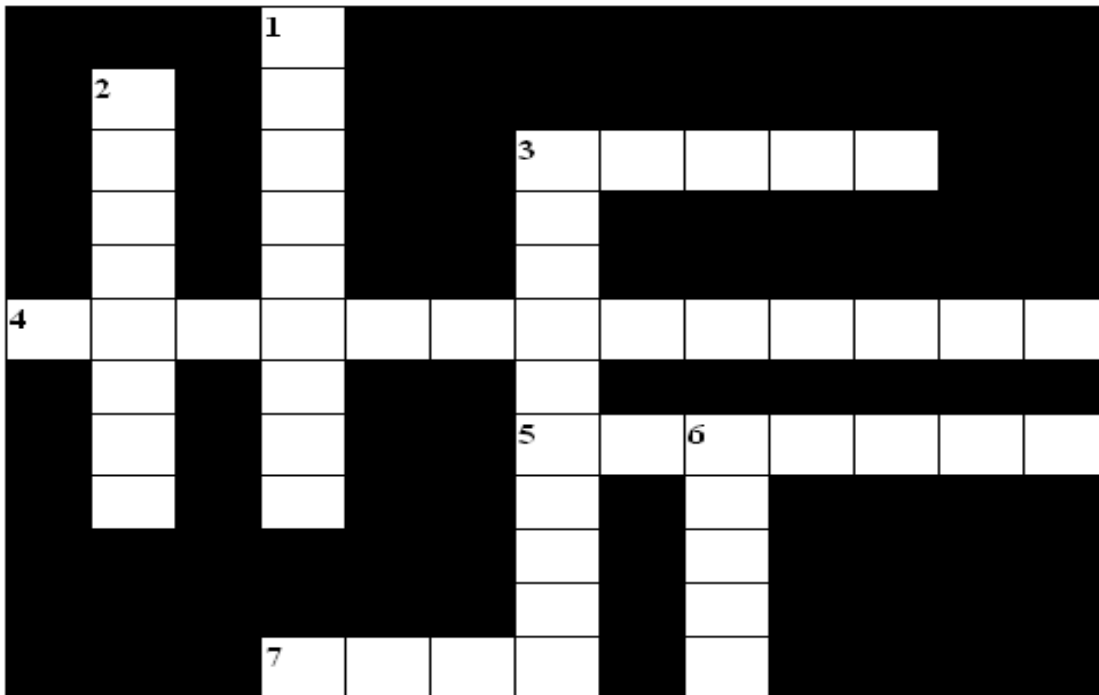


Fig.4-1

Across:







3. Which Normalization is achieved only when you have eliminated any possibility of modification anomalies? (5)
4. What is the process of organizing data in a database? (13)
5. What specifies a value to be inserted into the column to which the object is bound? (7)
7. What lets you specify; which users can or cannot enter into a particular column? (4)

Down:

1. In which Relationship each record in Table A may have many linked records in Table B but each record in Table B may have only one corresponding record in Table A. (9)
2. In which Relationship there is exactly one record in Table A that corresponds to exactly one record in Table B. (8)
3. How to bind the rule to a column or user-defined data type? (10)
6. Which Normalization Deals with non-loss and loss decompositions? (5)

5.0 Introduction To Programming Objects

Topics

-  5.1 Creating and Managing Views
-  5.2 Indexes
-  5.3 Implementing stored procedure
-  5.4 Implementing triggers
-  5.5 Creating user defined functions
-  5.6 Crossword





Topic: Views

Estimated Time – 30 mins.



Objectives : On completion of the session, the participant should know:

- What are views?
- How to implement views.



Presentation :

- A view is, in essence, a virtual table. It does not physically exist. Rather, it is created by a query, joining one or more tables.
- View can be **filtered** by using the where clause.
- Complex view can be obtained by performing joins.
- Alter view modifies a previously created view. It does not affect dependent stored procedures or triggers and does not change permissions.
- **Drop view** command removes one or more views from the current database.

Ex: This example creates and displays the result of the view.

```
USE northwind
--If exist (drop view)
--Else
CREATE VIEW prod_12
AS
SELECT *
FROM products
WHERE productID IN(1,2)

SELECT *
FROM prod_12
```



Scenario :

Dashes Industries is a large Firm and Mr. Jackson the CEO of the firm wants the employees of the firm who are also customers to be able to view only some of the details from the tables.



Demonstration/Code Snippet :

Step 1: Creating a view

```

If Exists (select * from sysobjects where name =
'customers_detail')
begin
Drop view customers_detail
End

else

create view customers_detail
as select orderid,c.customerid,employeeid,shipname,
shipcity,companyname,phone,country
from orders o inner join customers c on o.customerid=c.customerid
where employeeid=6 and orderid=11019

select * from customers_detail

```

	orderid	customerid	employeeid	shipname	shipcity	companyname	phone	country
1	11019	RANCH	6	Rancho grande	Buenos Aires	RANCH	(1) 123-5555	INDIA

Figure 5.1-1:Result



Context :

- Views are used to present specific column(s) from a table(s). It serves as a security mechanism.



Practice Session/s :

- Create a view "order_1" on the 'orders' table to include the rows with Freight > 35.



Check List :

- Possess knowledge on the importance of using view as a filter.



Common Error/s :

- Not providing alias names for the computed fields.
- Trying to manipulate the views.



Exception/s :

- Views cannot be created on a table which does not exist.
- Views cannot be modified, as it is the virtual representation of the table.
- Views cannot be combined with other SQL statements in a batch.

- The class of updatable views provided by SQL*server are
 - Instead of triggers.
 - Partitioned views.



Lesson/s Learnt :

- ☑ Uses of views.
- ☑ Uses of views in a query, a stored procedure, or from another view.



Best Practice/s :

- Views should be used when the same data (table) is required repeatedly.
- For viewing the data from different servers, distributed partitioned views should be used.



Topic: Indexes

Estimated Time – 40 mins.



Objectives : On completion of the session, the participant should be able to :

- Understand the B-Tree index structure.
- Appreciate the categories of indices.



Presentation :

- An index is a list arranged usually in an alphabetical order of some specified data.
- A "B-Tree" or "Balanced Tree" is the general structure that attempts to provide a consistent and relative low cost method of finding your way to a particular piece of information.

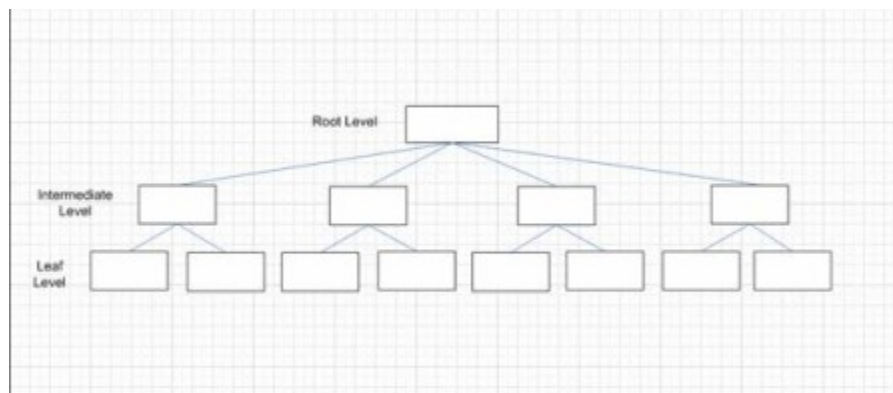


Figure 5.2-1 B-Tree

- **Page Split** means moving rows to a new page when a new row is added to a full index page
- **Fill Factor** reduces the potential for page splits by providing enough space for index expansion as data is added to the underlying table.
- A **Heap** is a table without a clustered index having a unique identifier or row ID that is created based on a combination of the extent, pages and a row offset for that row.
- **Types of Indexes**
 - Clustered index
 - Non-Clustered index

- A **Clustered Index** stores data rows in the table based on their key values. Each table can have only one clustered index. **Non-clustered** indexes have the same B-tree structure as Clustered indexes, with two significant differences:
 - The data rows are not stored in order.
 - The leaf level of a Non-Clustered index contains key values and references to the rows of data, instead of the data rows themselves.



Scenario :

The database administrator of a Software Company 'The Wizards' wants to maintain a record of all the employees, with each having a unique login Id to enter into the company website. The details of the login id and password are stored in a table. The data has to be retrieved faster as there are thousands of employees working in this organization.



Demonstration/Code Snippet :

Step 1: Use the Login table from the Northwind database. Create an index on the login_id column of the table.

Use Northwind

```
If Exists (select * from sysobjects where name = 'indxlogin_id ')
begin
Drop Index login.indxlogin_id
end
```

```
Create Index indxlogin_id ON login(login_id,login_password)
```

```
Select * from Login
```

	login_id	login_password	hint_question	hint_ans
1	bj72163	bharat	myname	name
2	hs71747	hemant123	Your School Name	B.H.S. Pilani
3	kp72087	kunal	wilson	name
4	rc72119	reenal	myname	name

Figure 5.2-2:Result

Step 2: Run the stored procedure SP_HELP to view the information of the indexes created on the table.

```
SP_HELP Login
```

	index_name	index_description	index_keys
1	indxlogin_id	nonclustered located on PRIMARY	login_id, login_password
2	PK__login__6442E2C9	clustered, unique, primary key located on PRIMARY	login_id

Figure 5.2-3:Result



Context :

- The use of indexes to search records from large databases
- To search for records with search key values within a range of values. A range query is one in which the query specifies any entry whose value is between two values.
- Searches for records in the table T1 that match, based on a join predicate, a row in another table T2 (an index nested loops join)



Practice Session/s :

- Create a Clustered or Non-Clustered index on any of the column of employees table.



Check List :

- Importance of using index for quicker retrieval (search) of the data on a huge database.



Common Error/s :

- Usage of wrong syntax.
- Run time error occurs if object is invalid.



Exception/s :

- Having many indexes may have negative impact on the performance because whenever there is a change in the table, immediately index has to reflect that change. This puts increases the load on the machine when more number of indexes are created.



Lesson/s Learnt :

- ☑ Index and the various types of indexes.
- ☑ Use of the Indexes.
- ☑ To specify the key(s) on which the index has to be created.



Best Practice/s :

- For searching data from the database it is advisable to use Indexes for quicker and logical retrieval.



Topic: Stored Procedures

Estimated Time – 35 mins.



Objectives : On completion of the session, the participant should know:

- The use of Stored Procedures.



Presentation :

- Stored Procedure is a saved collection of Transact-SQL statements that can take and return user-supplied parameters.
- **Alter Stored procedure** modifies a previously created Stored Procedure.
- **Drop Stored Procedures** deletes a Stored Procedure when no longer needed.
- Stored procedures can contain input and output parameters



Scenario :

The board member of Climbers Earthmovers wants to offer their employees an annual appraisal based on their department. The application has to be such that for a particular employee the salary has to be updated in the database depending on the department that the employee works in.



Demonstration/Code Snippet :

Step 1:

The table containing the employee details is contained in employee table which looks like:

```
Select * From Employee
```

	emp_code	emp_name	dept_co...	grade	a...	date_join	gender	salary	married
1	1	David BLain	GEN	M1	35	1995-01-01 00:00:00.000	M	16000	Y
2	2	Jack Nicolsion	SOFT	M3	27	1995-03-01 00:00:00.000	M	6000	N
3	3	Adam Sandler	SECY	S2	39	1996-07-24 00:00:00.000	M	4800	Y
4	4	Jack dawson	FIN	E2	39	1982-08-11 00:00:00.000	M	8000	Y
5	5	Heliray Faith	SOFT	E2	49	1982-03-03 00:00:00.000	F	8000	N
6	6	Andy Flower	SOFT	S6	26	1993-11-05 00:00:00.000	M	5500	N
7	7	Neilsia Taibu	SOFT	S5	30	1994-12-24 00:00:00.000	F	4600	Y
8	8	Orlando White	SOFT	S5	33	1990-01-12 00:00:00.000	M	5200	Y
9	9	Sudha Ganesan	MKTG	M2	32	1996-01-01 00:00:00.000	F	12500	Y

Figure 5.3-1:Result

Step 2:

In order to give appraisal the employee code or Id has to be entered and the salary will be updated based on the department to which the employee belongs.

The procedure for the same is written below:

```
If Exists (increase_salary)
    Drop procedure increase_salary
Else

CREATE PROCEDURE increase_salary(@e_c INT)
AS
DECLARE @d_c VARCHAR(10)
DECLARE @tsalary float
SELECT @d_c=dept_code,@tsalary=salary
FROM employee WHERE emp_code=@e_c

IF @d_c='MKTG'
BEGIN
SET @tsalary = @tsalary * 1.5 ;
END

IF @d_c='FIN'
BEGIN
SET @tsalary = @tsalary*1.3 ;
END

IF @d_c='SOFT'
BEGIN
SET @tsalary = @tsalary*1.2 ;
END

ELSE
BEGIN
SET @tsalary = @tsalary*1.1 ;
END

UPDATE employee SET salary=@tsalary WHERE emp_code=@e_c
```

Step 3:

Inorder to give appraisal the emp_code has to be passed while executing the stored procedure. Salary of that corresponding employee will be updated.

EXECUTE increase_salary 3

The database table looks like:

	emp_code	emp_name	dept_co...	grade	a...	date_join	gender	salary	married
1	1	David BLain	GEN	M1	35	1995-01-01 00:00:00.000	M	16000	Y
2	2	Jack Nicolson	SOFT	M3	27	1995-03-01 00:00:00.000	M	6000	N
3	3	Adam Sandler	SECY	S2	39	1996-07-24 00:00:00.000	M	5280	Y
4	4	Jack dawson	FIN	E2	39	1982-08-11 00:00:00.000	M	8000	Y
5	5	Heliray Faith	SOFT	E2	49	1982-03-03 00:00:00.000	F	8000	N
6	6	Andy Flower	SOFT	S6	26	1993-11-05 00:00:00.000	M	5500	N
7	7	Neilsia Taibu	SOFT	S5	30	1994-12-24 00:00:00.000	F	4600	Y
8	8	Orlando White	SOFT	S5	33	1990-01-12 00:00:00.000	M	5200	Y
9	9	Sudha Ganesan	MKTG	M2	32	1996-01-01 00:00:00.000	F	12500	Y

updated (pointing to the salary value 5280 in row 3)

Figure 5.3-2:Result



Context :

- SQL queries expose database design in the code which may be changed.
- A Stored Procedure is a pre-compiled executable object that contains one or more SQL statements, hence they execute sooner at the database server.
- The time taken to pass the individual commands to the database server from the program is saved. The DB is issued just one command and it executes all queries. Hence, the overall interaction time with the DB server reduces a great deal.
- This decreases the network load.



Practice Session/s :

- Write a procedure to add a new employee in the employee table. The procedure should accept all the information present in the employee table. It should not add any employee having the employee id already given to some other employee.



Check List :

- Possess knowledge of concept of Stored Procedure and the procedure to create, alter and delete Stored Procedure.
- Use of Stored Procedures instead of set of SQL statements independently.



Common Error/s :

- Create Procedure May Cause Error 1203: This problem may occur if both of the following are true:
 1. The Stored Procedure contains a create table command.
 2. The Stored Procedure contains a query that violates ANSI rules on SELECT lists when DISTINCT and ORDER BY is used.
- There are two types of errors in SQL Server:
 1. Fatal.
 2. Non-fatal.

- Fatal error:
Fatal errors cause a procedure to abort processing and terminate the connection with the client application.
- Non fatal error:
Non-fatal errors do not abort processing a procedure or affect the connection with the client application. When a non-fatal error occurs within a procedure, processing continues on the line of code that follows the one that caused the error.



Exception/s :

- Parameters not found:
 - This exception is raised when the Stored Procedure has input parameter in the back-end and no parameter is passed from the front end.
- Parameter overflow:
 - This condition is raised when the value supplied is greater in range as data type present in the Stored Procedure.
- Create procedure statement cannot be combined with another SQL statement in a single batch.



Lesson/s Learnt :

- ☑ How to Implement Stored Procedures.
- ☑ To Alter and Delete a Procedure.
- ☑ Stored Procedure can be created in the current database.



Best Practice/s :

- Use Stored Procedure to reduce network load, reusability of code and enhance security.



Topic: Triggers

Estimated Time – 45 mins.



Objectives : On completion of the activity, the participant should

- Understand the importance of Triggers.
- Able to write Triggers.
- Identify the type of the Trigger to be implemented



Presentation :

- A **Database Trigger** is a procedural code that is automatically executed in response to certain events on a particular table in a database.
- There are two classes of Triggers; they are either "Row Triggers" or "Statement Triggers".
 - Row Triggers define an action for every row of a table
 - Statement Triggers occur only once per INSERT, UPDATE or DELETE statement.
- There are two types of Triggers.
 - Data Definition Language Triggers:
 - ✓ SQL has extended the Trigger functionality of Data Manipulation Language (DML) commands such as INSERT, UPDATE, and DELETE to incorporate Data Definition Language (DDL) commands like CREATE DATABASE, DROP TABLE, and ALTER TABLE.
 - Data Manipulation Language Triggers
 - ✓ DML Trigger statements use two special tables: the **deleted** table and the **inserted** tables.
 - ✓ The **deleted** table stores copies of the affected rows during DELETE and UPDATE statements.
 - ✓ The **inserted** table stores copies of the affected rows during INSERT and UPDATE statements
 - ✓ There are generally two types of DML Triggers.
 - ❖ After Trigger: - After Triggers are executed after the action of the INSERT, UPDATE, or DELETE statement is performed.

- ❖ **INSTEAD OF Triggers** are executed in place of the usual triggering action. **INSTEAD OF Triggers** can also be defined on views with one or more base tables, where they can extend the types of updates a view can support.



Scenario :

Global Software is a software company. The Program Manager, Mr. Robbins maintains the records of the associates according to batches during the training program. The details of the associates are entered at the time of the joining. These details cannot be changed through insertion, updating or deletion of the available data.



Demonstration/Code Snippet :

Step 1: Use the table associate details from the Northwind database

```
use Northwind
select * from associate_details
```

The output of the above query looks like:

	Uid	Name	Address	phone_no	Designation
1	1001	AAA	abcd	1234	eltp
2	1002	BBBB	abcd	7565	eltp
3	1003	CCC	abcd	9878	eltp
4	1004	DDD	abcd	4321	eltp

Figure 5.4-1:Result

Step 2: Create a Trigger named TR_AssociateDetailNotChanged on the table associate_details in the database using the code as follows:

```
Create TRIGGER TR_AssociateDetailNotChanged
ON associate_details
AFTER update, delete, insert
as
    if exists
        (
            select 'True'
            from associate_details where Uid is not null
        )
begin
    Raiserror('Details cannot be changed. Transaction
Failed.',16,1)
    Rollback Transaction --undo the action performed,if any.
end
```

With the above code the Trigger has been created.

Step 3: As the Trigger is created, it is not possible to insert, update or delete the details from the table `associate_details`. We can try it out with the following code:

```
update associate_details set address = 'MP' where Uid = 1001

delete from dbo.associate_details where Uid = 1001

insert into associate_details values
('1212','ee','df','232','fef')
```

As the above code is run, it gives the following error message.

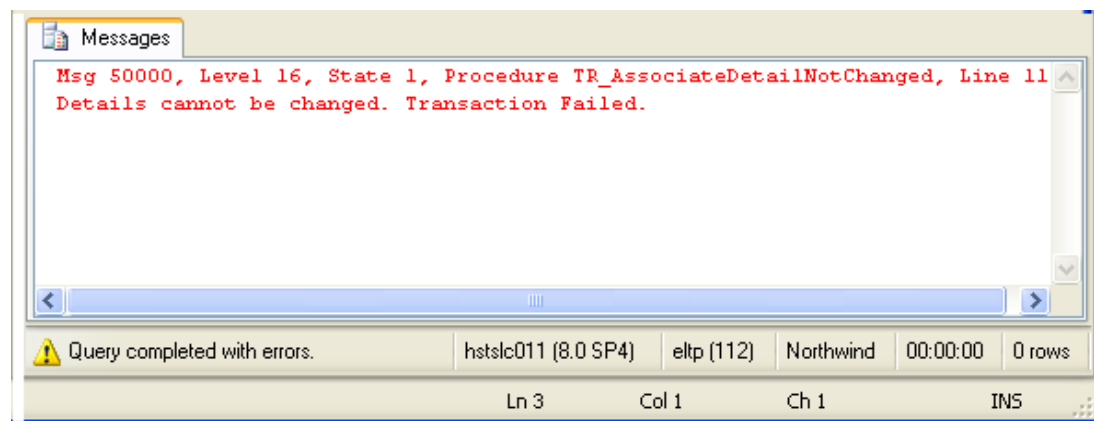


Figure 5.4-2:Output showing error has occurred.

The table `associate_details` retains its data.

```
select * from associate_details
```

	Uid	Name	Address	phone_no	Designation
1	1001	AAA	abcd	1234	eltp
2	1002	BBBB	abcd	7565	eltp
3	1003	CCC	abcd	9878	eltp
4	1004	DDD	abcd	4321	eltp

Figure 5.4-3:Result



Scenario :

The Zenith is a sales company. The administrator, Mr. John keeps record of the orders of the products ordered by their various customers in the Northwind database. Mr. John uses four tables for this purpose.



Demonstration/Code Snippet:

Step 1: Using Northwind database, create four tables namely, tbCustomers, tbOrders, tbProducts, tbOrderItems.

Use Northwind

```
Create table dbo.tbCustomers
(
    CustomerID varchar(5) NOT NULL PRIMARY KEY,
    Name varchar(40) NOT NULL
)
Create table dbo.tbOrders
(
    OrderID int IDENTITY NOT NULL PRIMARY KEY,
    CustomerID varchar(5) NOT NULL REFERENCES
dbo.tbCustomers(CustomerID),
    OrderDate datetime NOT NULL
)
Create table dbo.tbProducts
(
    ProductID int IDENTITY NOT NULL PRIMARY KEY,
    Name varchar(40) NOT NULL,
    UnitPrice money NOT NULL
)
Create table dbo.tbOrderItems
(
    OrderID int NOT NULL REFERENCES dbo.tbOrders(OrderID),
    ProductID int NOT NULL REFERENCES
dbo.tbProducts(ProductID),
    UnitPrice money NOT NULL,
    Quantity int NOT NULL CONSTRAINT PK_OrderItem PRIMARY KEY
    CLUSTERED(OrderID,ProductID)
)
```

Step 2: Insert some sample records into the table as shown below:

```
Insert into dbo.tbCustomers values('abcd', 'Good Garage')
Insert into dbo.tbOrders values('abcd', current_timestamp)
Insert into dbo.tbProducts values('widget', 5.00)
Insert into dbo.tbProducts values('bins', 8.01)
Insert into dbo.tbOrderitems values(1,1,5.00,3)
```

Step 3: Create a view named 'CustomersOrders_vw' that selects columns from one or more tables created above. The code to create the view is given in the following.

```
If exists(select * from sysobjects where name =
'CustomersOrders_vw')
drop view CustomersOrders_vw
```

```
print 'View dropped'

GO

Create View CustomersOrders_vw
AS
Select
o.OrderID,o.OrderDate,oi.ProductID,oi.UnitPrice,oi.Quantity
  From dbo.tbOrders as o,dbo.tbProducts as p,dbo.tbOrderItems
as oi
  where o.OrderID = oi.OrderID
```

Step 4: Insert a row into the view as shown below:

```
Insert into CustomersOrders_vw
(
  OrderID,OrderDate,ProductID,UnitPrice,Quantity
)
VALUES
(
  1, '2000-04-06', 2, 6.00, 10
)
```

The above code throws an exception as follows:

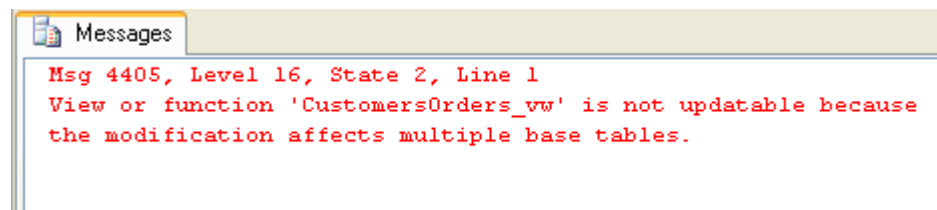


Figure 5.4-4:Output showing error has occurred.

Step 5: Create a Trigger named 'trCustomerOrderInsert' to solve the above problem.

```
If exists(select * from sysobjects where name =
'trCustomerOrderInsert')
drop trigger trCustomerOrderInsert
print 'trigger dropped'

GO

Create Trigger trCustomerOrderInsert ON CustomersOrders_vw
INSTEAD OF INSERT
AS
BEGIN
  If (Select count(*) From Inserted) > 0
  Begin
    Insert into dbo.tbOrderItems
    Select i.OrderID,i.ProductID,i.UnitPrice,i.Quantity
```



```

From Inserted as i, dbo.tbOrders as o
where i.OrderID = o.OrderID

If @@ROWCOUNT = 0 -- If no row has been in inserted...
                    -- Gives an error message
RAISERROR( 'No matching orders. Cannot perform insert' ,10,1)
End
END

```

Step 6: The Insert statement works fine after the Trigger has been created and the row gets inserted into the tbOrderItems table. The table tbOrderItems looks as below:

```
Select * From dbo.tbOrderItems
```

	OrderID	ProductID	UnitPrice	Quantity
1	1	1	5.00	3
2	1	2	6.00	10

Figure 5.4-5:Result



Context :

- Triggers are used only with Insert, update and Delete commands.
- Triggers are restricted to the current database, although you can reference an object outside the database.
- A trigger can reference a temporary table but can't modify one.
- A trigger can't reference a system table.



Practice Session/s :

- Create a Trigger in table 'employee' in Northwind to insert the details of a new employee providing all the details as required.
- Create a Trigger to update the country of those employees of 'employees' table who are in USA to UK as their base has been shifted.



Check List :

- Importance of using Triggers in SQL.



Common Error/s :

- Syntax is not followed properly.
- Any run time error can occur due to use of Invalid Object.



Exception/s :

- If a Trigger is fired and it is not closed, it throws an exception.
- If any of the DML statements contained in Trigger (like insert, update) don't find specified column then an exception is thrown.
- In the INSTEAD OF Triggers, there can be only one Trigger per table or view for each of the different Insert, Update, Delete.



Lesson/s Learnt :

- ☑ The triggering events of insert, update and delete can be associated with a single Trigger
- ☑ On the failure of Trigger, it's necessary to rollback the transaction to avoid inconsistencies
- ☑ Prefix the Trigger name appropriately
- ☑ Triggers are used for updating summary information.
- ☑ Triggers feed de-normalized tables for reporting.
- ☑ Triggers can set condition flags.
- ☑ Triggers can be nested.
- ☑ Triggers can be recursive.



Best Practice/s :

- Learn the use of Triggers and implement it successfully.



Topic: User Defined Functions

Estimated Time – 45 mins.



Objectives : On completion of the activity, the participant will know :

- What is UDF?
- How and When to use UDF?



Presentation :

- Functions are sub routines made up of one or more Transact-SQL statements that can be used to summarize the code for re-use.
- SQL Server 2000 supports three types of User Defined Functions:
 - **Scalar functions:** -Scalar functions return a single data value of the type defined in a RETURNS clause.
 - **Inline table-valued functions:** - Table-valued functions return a table. For an inline table-valued function, there is no function body; the table is the result set of a single SELECT statement.
 - **Multi statement table-valued functions:** -The function body, defined in a BEGIN...END block, contains the TRANSACT-SQL statements that build and insert rows into the table that will be returned.



Scenario :

‘Luminous Systems’ is one of the fortune 500 companies. Mr. Richardson is the HR personnel of the company. To view the employees information, the HR personnel has two options. One is to view the employee details according to their title of courtesy. Another way is to view all the details in a two dimensional format.



Demonstration/Code Snippet :

Step 1: Use the Employees table from the Northwind database.

```
Use Northwind
```

```
Select * From Employees
```

Step 2: Create a function that will give the details of the Employees from the table upon asking the Title of Courtesy.

```
Create Function dbo.fnTitleOfCourtesy(@toc nvarchar(50))
Returns nvarchar(50)
as Begin
Return @toc
End
```

Step 3: After the function has been created, it can be described by using the following query.

```
Select * from Employees where
dbo.fnTitleOfCourtesy(TitleOfCourtesy) =
dbo.fnTitleOfCourtesy('Mr.')
```

The output of the above query is shown below.

The function 'dbo.fnTitleOfCourtesy' returns a scalar value.

	EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate	Address
1	5	Testa	Steven	Sales M...	Mr.	1955-03-0...	1993-10-1...	14 Garrett H
2	6	Fujiyama	Michael	Sales R...	Mr.	1963-07-0...	1993-10-1...	Coventry Ho
3	7	Trump	Robert	Sales R...	Mr.	1960-05-2...	1994-01-0...	Edgeham H

Figure 5.5-1:Result

Step 4: Create another function to invoke all the Employees details from the Employees table.

```
Create Function dbo.fnGetEmployees()
Returns Table
as
Return (Select * From Employees)
GO
```

This function returns a table.

Step 4: To call the above function, the following query has to be run.

```
select * from dbo.fnGetEmployees()
```

The output of the above query is as shown below.

	EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate
1	1	Hilton	Nancy	Sales Representative	Ms.	1948-12-08 00:00:00.000
2	2	Witherspoon	Reese	Vice President, Sales	Dr.	1952-02-19 00:00:00.000
3	3	Nightly	Janet	Sales Representative	Ms.	1963-08-30 00:00:00.000
4	4	Maria	Margaret	Sales Representative	Mrs.	1937-09-19 00:00:00.000
5	5	Testa	Steven	Sales Manager	Mr.	1955-03-04 00:00:00.000
6	6	Fujiyama	Michael	Sales Representative	Mr.	1963-07-02 00:00:00.000
7	7	Trump	Robert	Sales Representative	Mr.	1960-05-29 00:00:00.000

Figure 5.5-2:Result



Context :

- To return a value even when we reference a select statement (unlike procedures).
- UDFs can be used to return tables.
- UDFs can also be used as Constraints or Default Values.
- UDFs can be used in Views to be returned as columns.



Practice Session/s :

- Design an UDF which will get the last day of the month when a date is passed to the same.
- Almost any OS data type can be returned from an UDF. Using this concept, Design an UDF that will return an UDDT (User Defined Data Type) from the UDF.



Check List :

- Have knowledge about the importance and use of UDF.
- Know how to use nested UDF.
- Be familiar with the process of how to use UDF to return tables.



Common Error/s :

- Usage of wrong syntax.
- Run time error occurs if object is invalid.



Exception/s :

- Function must be defined, before it is called.
- Run time error occurs if the object, that the function calls, is invalid.



Lesson/s Learnt :

- When to use UDF and when to use Stored Procedures.
- UDF can be used to return tables as well.
- There is a limit to the nesting of UDFs



Best Practice/s :

- It is better to follow a common naming convention for the functions created.
- Create different functions for the operations that are carried frequently to enhance its re-usability.

Crossword: Unit-5

Estimated Time: 10 mins.

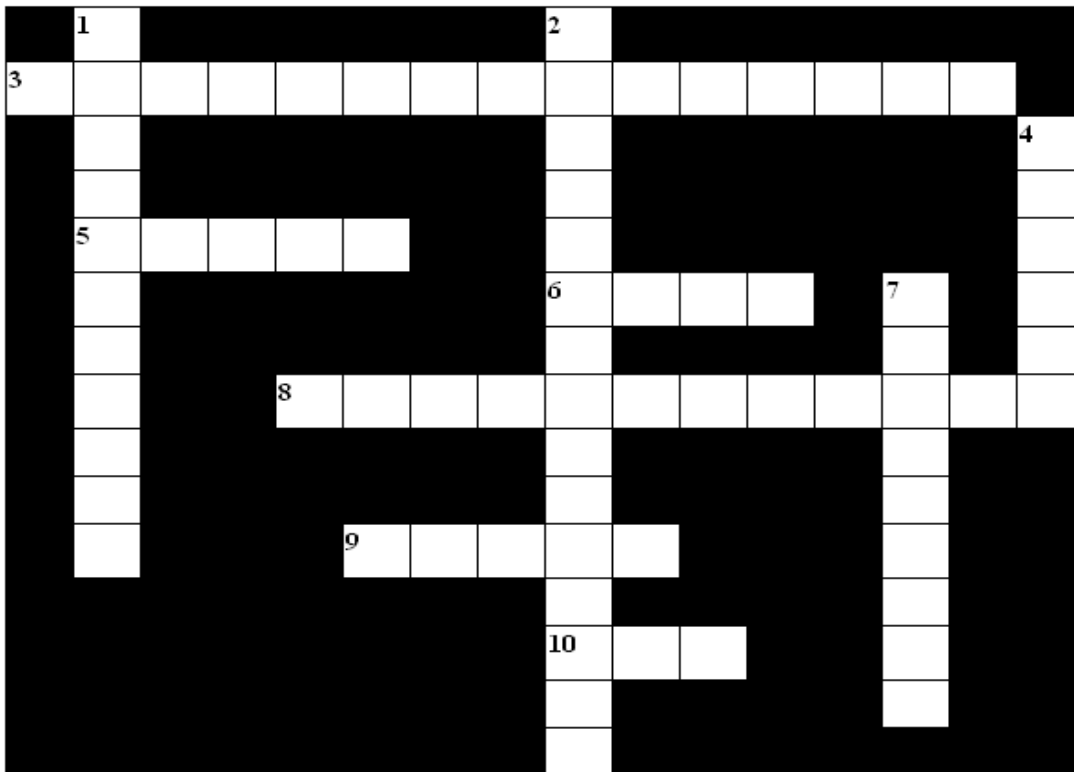


Fig.5-1

Across:







3. A procedural code that is automatically executed in response to certain events on a particular table in a database is called as? (15)
5. A list arranged usually in alphabetical order of some specified datum is called as? (5)
6. Which command is used to remove one or more views from the current database? (4)
8. Which Triggers executed after the action of the insert, update or delete statement is performed?
9. Which stored procedure modifies a previously created stored procedure? (5)
10. Subroutines made up of one or more T-SQL statements that can be used to encapsulate code for re-use is called (Acronym)? (3)

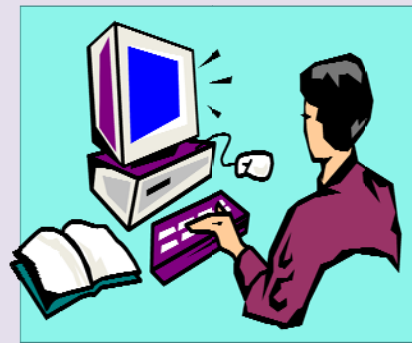
Down:

1. Which class is used for updatable views, provided by SQL * server? (11)
2. A saved collection of T-SQL statements that can take and return user-supplied parameters is called as? (15)
4. Which functions return a single data value of the type defined in a RETURNS clause? (6)
7. A method of moving rows to a new page when a new row is added to a full index page is called? (9)

6.0 Advanced T-SQL: Programmability

Topics

-  6.1. Batches
-  6.2 Scripts
-  6.3 Variables
-  6.4 control of flow statements
-  6.5 Cursor
-  6.6 Crossword





Topic: Batches

Estimated Time – 20 mins.



Objectives : On completion of the session the participant should be aware of:

- The concept for creating different batches.
- The rules that have to be followed in batches.
- The importance of 'GO' command.
- The errors encountered while creating batches and handling the errors.



Presentation :

- A **Batch** is a set of SQL statements submitted together and executed as a group, one after the other.
- The rules for **Batches** are as given below:
 - Any execution of a Stored Procedure after the first statement in a Batch must include the keyword 'EXECUTE'.
 - The scope of local (user-defined) variables is limited to a Batch, and cannot be referenced after a 'GO' command.
- 'GO' is not a Transact-SQL statement; it is a command recognized by SQL utilities and Teratrax Database Manager to signal the end of a batch of SQL statements.
- Errors in batches fall into two categories:
 - Syntax error: If the query parser finds a syntax error, processing of that batch is cancelled immediately as syntax check happens before the batch is compiled or executed
 - Run time error: Run time errors work differently. Any statement that is executed before the run time error is encountered is considered as completed, so anything that the statement does will remain intact unless it is part of an uncommitted transaction.



Scenario :

Mr. Anthony is an administrator of THE MEGA MART who wants to create a database to store all the records of the people working in the mart.

A table has to be created where the Emp_ID of the employees and their designation can be stored. At the same time, all the records are verified whether they have been inserted or not.



Demonstration/Code Snippet :

Step 1: Create a database named Anthony in which a table named Emp_Record is created. Insert some records into the table.

```
Create database Anthony
Create table Emp_Record
( Emp_ID numeric(9), Designation varchar(10))
Insert into Emp_Record values(73008, 'Manager')
Insert into Emp_Record values(73127, 'Executive')
Select * from Emp_Record
GO
```

Step 2: Select all and press F5. The output of the above Batch is as shown below.

	Emp_ID	Designation
1	73008	Manager
2	73127	Executive



Context :

- A Batch is used when a group of one or more Transact-SQL statements to be sent at one time from an application to Microsoft® SQL Server™ for execution.
- Batches are used when something has to happen either before or separately in a script.



Practice Session/s :

- Create a table for Mr. Anthony, which contains fields that store the personal information of employees such as Full Name, Address, Phone No and Date of Joining. Make all the queries 'GO' in just one event of a key.



Check List :

- Importance of batches in SQL Server.
- Know the usage of 'GO' command.



Common Error/s :

- Runtime Errors include errors such as an arithmetic overflow or a constraint violation.
- Syntax error, it prevents the compilation of the execution plan, so none of the statements in the Batch are executed.
- T-SQL statements cannot precede the 'GO' statement.
- Dependencies between Batches can be there if one Batch tries to perform a job that depends on the first Batch being complete.
- 'GO' indicates the end of a current Batch or time.
- Divide the Batch using a 'GO' statement if the execution of Defaults or Procedures or Rules or Triggers or Views is in the same Batch.



Exception/s :

- Execution of large Batch of SQL statements that gives many result sets may stop processing the Batch before all statements in the Batch are executed as SQL Server fills the output buffer until it hits an internal limit and cannot continue to process more result sets.
- Each Batch is sent to the server independently.
- An error in one Batch does not prevent another Batch from running.
- There are several commands that absolutely must be a part of their own Batch like :
 - Create Default
 - Create Procedure
 - Create Rule
 - Create Trigger
 - Create View



Lesson/s Learnt :

- ☒ To make a set of SQL commands run at one time, Batches should be used.
- ☒ There can be dependencies between Batches.



Best Practice/s :

- Understand the use of Batches and execute it successfully.
- Run the Drop statement in a separate and prior batch so that it will be complete when the batch with the Create statement executes.



Topic: Scripts and Variables

Estimated Time – 40 mins.



Objectives : On completion of the activity, the participant should know

- How to write scripts, declare variables in it and assign values to it.



Presentation :

- A script is treated as a unit that makes use of both system functions and local variables.
- The 'USE' statement sets the current database.
- The variables in a script are assigned values using SET.
- 'Select' is usually used to assign a variable when the source of information to be stored is in the form of a query.
- A Transact-SQL local variable is an object that can hold a single data value of a specific type.
- Variables in batches and scripts are typically used:
 - As a counter either to count the number of times a loop is performed or to control the no. of times the loop is performed.
 - To hold a data value to be tested by a control-of-flow statement.
 - To save a data value to be returned by a stored procedure return code or function return value.
- Variables are declared in the body of a batch or procedure with the DECLARE statement.



Scenario :

A Mega Mart Employee Karan wants to verify the name of a customer using his bill identity provided in the bill.



Demonstration/Code Snippet :

Step1: Below is a script written which uses the PUBS database. The script returns the buyers details by assigning a value of the BuyerID to a variable.

```
use pubs  
DECLARE @buyerID INT
```

```
SET @buyerID = 11
SELECT *
FROM buyers
WHERE buyer_id= @buyerID
GO
```

	buyer_name	buyer_id
1	John	11



Context :

- In a script, all the commands are usually built up to serve one overall purpose.
- Examples include scripts to build a database, system maintenance, backups etc.
- To hold a data value to be tested by a control-of-flow statement.



Practice Session/s :

- Write a simple program to print 'Climber' 10 times with help of a variable as a counter or loop.



Check List :

- Know how to declare a variable
- Know how to use a declared variable.



Common Error/s :

- Syntax error: The developer sometimes incorrectly defines the variable.



Exception/s :

- No exceptions so far. It is always advisable to use variables while coding.



Lesson/s Learnt :

- ☒ Variable names must begin with an '@' sign.
- ☒ Variable data types should be system-supplied or user-defined data type.
- ☒ A variable cannot be of **text**, **ntext**, or **image** data type.



Best Practice/s :

- Usage of the correct syntax for declaring a variable



Topic: Try and Catch

Estimated Time – 20 mins.



Objectives : At the end of the session, the participant should understand:

- The use of try and catch block.
- The types of errors handled in the catch block.
- The built-in error functions that can be used in a catch block.



Presentation :

- The conventional term referring to Try-Catch blocks is Exception Handling.
- The TRY/CATCH block consists of a block of code identified as a TRY block, followed by another block of code known as a CATCH block.
- If an error is encountered in the TRY block, then the CATCH block is executed to determine what actions should be taken to deal with the error encountered.
- TRY/CATCH blocks can be nested. In doing so, it helps in controlling error handling more efficiently.
- Errors that cannot be trapped by a TRY/CATCH block are:
 - Syntax error within a batch of T-SQL statements.
 - Deferred name resolution errors created by statement level recompilations.
 - If a process is terminated by a KILL command then a TRY/CATCH block does not capture this error
 - Client interrupt requests or broken client connections are not trapped by the TRY/CATCH block.



Scenario :

‘Global Solutions’ is a software company. Mr. Kevin is the database administrator in the company. The administrator wants to create a table named test_table if it is not present in the database.



Demonstration/Code Snippet :

Step 1: Use the Northwind database and create a table named test_table if it does not exist. It also gives the details of the errors that are encountered if the table exists. The errors are handled by the catch block.

```
Use Northwind

BEGIN TRY
-----Try and create our table
    CREATE TABLE test_table(
        coll int primary key
    )
END TRY

SELECT * FROM ERRORLOG

BEGIN CATCH
    DECLARE @ErrorNo int,
            @Severity tinyint,
            @State smallint,
            @LineNo int,
            @Message nvarchar(4000)

    SELECT
        @ErrorNo = ERROR_NUMBER(),
        @Severity = ERROR_SEVERITY,
        @State = ERROR_STATE(),
        @LineNo = ERROR_LINE(),
        @Message = ERROR_MESSAGE()

    IF @ErrorNo = 2714
        print 'WARNING: Skipping CREATE table already exists'
    ELSE
        RAISERROR(@Message,16,1)
END CATCH
```



Context :

- Ability to provide a technique to handle some or all possible errors that may occur in the given block of code, while still running the code.



Practice Session :

- Write a program to insert the details of an employee by writing the code in a try catch block.



Check List :

- Importance of a try catch block.
- Use of try catch block as an exception handling code.



Common Error/s :

- Connection error.
- Dataset fill error.



Exception/s :

- Is not compatible with the older version of SQL Server.
- Compatible only on SQL Server version of 2005 and above.



Lesson/s Learnt :

- ☑ How and when to use a Try-Catch block for exception handling.
- ☑ How to get all the details of error using Try and Catch block.



Best Practice/s :

- Always use a Try Catch block to show the type of error occurring while running a code, without exposing it.



Topic: Cursors

Estimated Time – 35 mins.



Objectives : On completion of the activity, the participant should know

- The use of Cursors in SQL Server.
- The way to declare a Cursor in SQL Server.
- The life span of a Cursor.
- The types of Cursor used in SQL Server.



Presentation :

- Cursor is a way of taking a set of data , capable of interacting with a single record at a time on that set.
- The result set that we place in a Cursor has several distinct features that sets it apart from a normal select statement:
 - Cursor can be declared separately without actually executing it.
 - The cursor and its result set, are named at declaration – it is then referred to by its name.
 - The result set in a cursor, once opened, continues to be open until it is closed.
 - Cursors have a special set of commands that are used to navigate the record set.
- The life span of a cursor is shown below:
 - Declare the cursor and fill it.
 - Open it.
 - Fetch from it.
 - Close cursor and reallocate / de-select it.
- SQL Server Compact Edition supports the following types of cursors:
 - **Base table:** Base table cursors are the lowest level of cursors available. Base table cursors can scroll forward or backward with minimal cost, and can be updated. It is fastest of all supported cursor types.
 - **Static:** A static cursor creates and stores a complete copy of the result set. The exception to this is long-value data that is retrieved when a user explicitly requests it. This result set is filled as and when needed.
 - **Forward-only:** The forward-only cursor is the fastest cursor that you can update, however it does not support scrolling. It supports only fetching the rows serially from the start to the end of the cursor. The rows are not retrieved from the database until they are fetched.

- **Forward-only/Read-only:** Forward-only/read-only cursors, referred to as forward-only cursors, are the fastest cursors, but cannot be updated.
- **Keyset-driven:** The keyset-driven cursor is a scrollable cursor that can be updated. A keyset-driven cursor is controlled by a set of physical identifiers known as the keyset. The keyset is based on all the rows that qualify for the SELECT statement at the time the cursor is opened. The keyset is built in a temporary table when the cursor is opened.



Scenario :

Barclays is a leading bank offering insurance and banking services to customers. The manager of a branch of the bank, Mr. Chris Churchill wants a banking application which successfully executes the banking transaction for their ATM's customers. A successful transaction means updating the cash credit and debit in their respective accounts and hence in the company database.



Demonstration/Code Snippet :

Step 1: Use the table bank_trans3 for bank transaction in the Northwind database.

	acc_no	acc_holder	tot_bal	trans_type	amt
1	100	prateek	5000	d	1000
2	101	ramesh	5000	d	1000

Step 2: The database to be updated may be either a single record or more, hence cursor is used in this case:

Use Northwind

```
If Exists (Select * from sysobjects where name = 'banktrans')
Begin
    DROP Procedure banktrans
End
```

GO

```
Create Procedure sp_banktrans
as
Begin
Begin Transaction
Declare @accno int,@trans_type CHAR,@amt
int,@tot_bal int,@total int
Declare CASH_cur cursor for select acc_no,tot_bal,trans_type,amt
from bank_trans3
```

```

Set @tot_bal =0

OPEN CASH_cur
FETCH NEXT FROM CASH_cur INTO @accno,@total,@trans_type,@amt

While @@fetch_status = 0
Begin

    If @trans_type = 'd'
    Begin
        SET @tot_bal = @total + @amt
    End
    Else
    Set @tot_bal = @total - @amt
    Update bank_trans3 SET tot_bal = @tot_bal
    WHERE acc_no = @accno
    Set @tot_bal =0
    FETCH NEXT FROM
    CASH_cur INTO @accno,@total,@trans_type,@amt
    Continue
End
CLOSE CASH_cur
Commit Transaction

-- Frees up the memory associated with the cursor
DEALLOCATE CASH_CUR
End

```

Step 3: After running the above cursor, the database is updated for various transactions and the table will look as follows:

	acc_no	acc_holder	tot_bal	trans_type	amt
1	100	prateek	6000	d	1000
2	101	ramesh	6000	d	1000

Step 4: Both the present account holders have made a transaction to deposit Rs.1000 each, hence the total balance has been updated after running the cursor.



Context :

- Cursor operations are used while dealing with the same data at the same time.



Practice Session/s :

- Use Fetch in a simple cursor.
- Use FETCH to store values in variables.

- Declare a SCROLL cursor and use the other FETCH options



Check List :

- Be aware of the importance of using cursor in SQL Server.
- Know the difference between forward_only and scrollability.



Common Error/s :

- Syntax is not followed properly.
- Run time error: Run time errors work differently. Any statement that has been executed before the run time error is encountered is considered as completed, so anything that the statement does will remain intact unless it is part of an uncommitted transaction.
- Opening a cursor that is already opened.
- Closing a cursor that is not opened.



Exception/s :

- Recommended to use in disconnected architecture
- The scope of a cursor is Global by default.
- Make use of **sp_describe_cursor** to list out all the currently active options of a cursor.



Lesson/s Learnt :

- ☒ To be able to implement cursor.
- ☒ To know how to fetch rows one at a time when multiple rows are returned.



Best Practice/s :

- Limit the use of cursors to only when absolutely necessary

Crossword: Unit-6

Estimated Time: 10 mins.

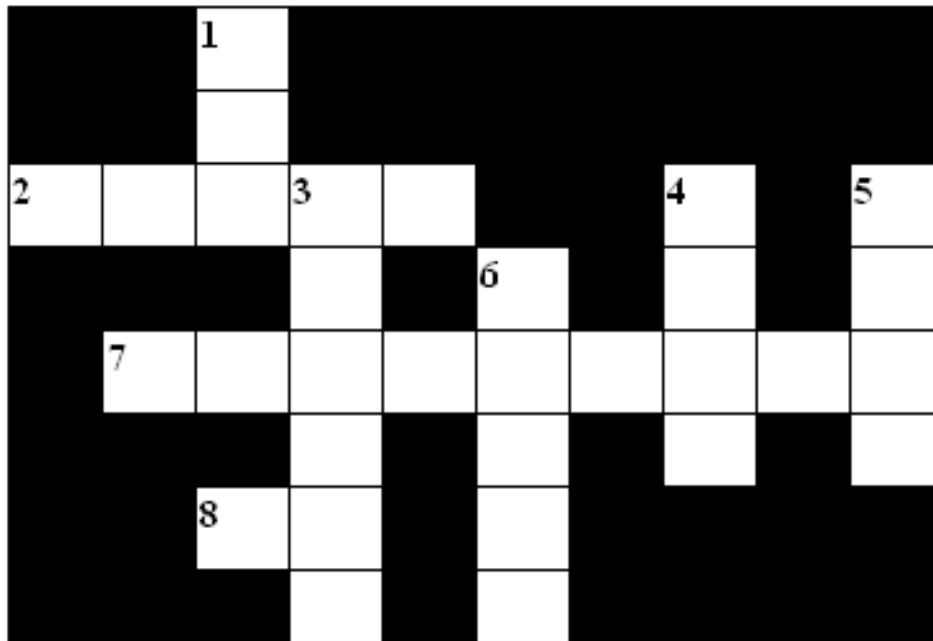


Fig.6-1

Across:







2. A set of SQL statements submitted together and executed as a group, one after the other is called as? (5)
7. What is used to hold a data value in batches and scripts? (9)
8. What is not a T-SQL statement; "Command that is recognized by SQL utilities and Teratrax Database Manager to signal the end of a batch of SQL statements". (2)

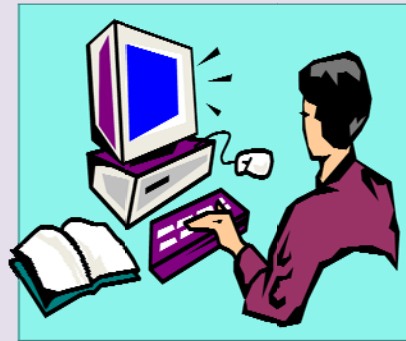
Down:

1. Which statement can be used to assign a value to a variable? (3)
3. A method of taking a set of data, and being able to interact with a single record at a time in that set is called as? (6)
4. Even TRY CATCH block cannot capture an error if a process is terminated by this command. What is that command? (4)
5. Which cursor is known as lowest level of cursor? (4)
6. If an error is encountered in TRY block, in which block it will be handled? (5)

7.0 Managing Transactions And Locks

Topics

-  7.1 Introduction
-  7.2 Managing transactions
-  7.3 SQL server Locking
-  7.4 Managing locks
-  7.5 Dead locks
-  7.6 Crossword





Topic: Transactions

Estimated Time – 25 mins.



Objectives : At the end of the session, the participant should understand the

- The importance of transactions management.
- The ACID properties.
- The transaction commands.



Presentation :

- Transactions are an inherent part of any application that collects or manipulates data.
- SQL Server has to ensure the data integrity. This means that no two users should modify the same piece of data at the same time. Nor should they read "dirty" data—modified but uncommitted rows.
- SQL Server writes the changes to the log file first; if all statements succeed, the transaction is then committed—the net result of changes is saved to the data file. If any of the statements fail, then the whole transaction is rolled back, and none of the changes are saved.
- Each transaction has to abide by the ACID properties, defined in the following table:

ACID Property	Meaning
Atomicity	Either all or no work is performed.
Consistency	A transaction must leave data in a consistent state.
Isolation	Each transaction is independent of all other transactions. That means each transaction will read data that was committed prior to beginning of the other transactions or after the end of the other transactions.
Durability	After transaction is committed, the data is in a persistent state, regardless of the circumstances. SQL Server records the transaction in the transaction log, and marks it as being committed. If the transaction is not committed, then SQL Server will roll all data changes back.

- Transaction has 2 types:

- By default, each INSERT, UPDATE, and DELETE statement runs within an implicit transaction.
- Explicit transactions must be specified by the programmer. Such transactions are included in BEGIN TRANSACTION ... COMMIT TRANSACTION block.



Scenario :

The HR Department of Citisoft Technologies wants an application that can delete the details of its employees on their resignation or their exit.

Their requirement is such that this information should be well preserved, secure and consistent, even when some unforeseen system failure occurs.



Demonstration/Code Snippet :

Step 1: Use the Employees table from the Northwind database:

```
Select * from Employees
```

	EmployeeID	LastName	FirstName	Title	TitleOfCourtesy
1	1	Hilton	Nancy	Sales Representative	Ms.
2	2	Witherspoon	Reese	Vice President, Sales	Dr.
3	3	Nightly	Janet	Sales Representative	Ms.
4	4	Maria	Margaret	Sales Representative	Mrs.
5	5	Testa	Steven	Sales Manager	Mr.
6	6	Fujiyama	Michael	Sales Representative	Mr.
7	7	Trump	Robert	Sales Representative	Mr.
8	8	Cullinan	Laura	Inside Sales Coordinator	Ms.

Figure 7.1-1: Output of the Query

Step 2: Now in order to delete employee details, the following procedure involving transaction is created:

```
Use northwind
```

```
If Exists
(Select * from sysobjects where name = 'Deleteemployee')
begin
DROP procedure Deleteemployee
end
```

```
GO
```



```

CREATE Procedure Deleteemployee @employeeID int
AS
-- This sproc performs deletion.It deletes all of the
-- info associated with supplied employeeId.

-- Start the transaction
Begin Transaction

-- : Issue the DELETE statements, checking @@ERROR after each
statement
Delete From Employees Where EmployeeID = @employeeID

-- Rollback the transaction if there were any errors
IF @@ERROR <> 0
    Begin
        -- Rollback the transaction
        ROLLBACK
        -- Raise an error and return
        RAISERROR ('Error in deleting employees in
deleteemployees.', 16, 1)
        RETURN
    End
Else --
If we reach this point, the commands completed successfully
-- Commit the transaction....
    COMMIT

```

Step 3: Execute the following procedure:

```
Exec proc Deleteemployee 8
```

	EmployeeID	LastName	FirstName	Title	TitleOfCourtesy
1	1	Hilton	Nancy	Sales Representative	Ms.
2	2	Witherspoon	Reese	Vice President, Sales	Dr.
3	3	Nightly	Janet	Sales Representative	Ms.
4	4	Maria	Margaret	Sales Representative	Mrs.
5	5	Testa	Steven	Sales Manager	Mr.
6	6	Fujiyama	Michael	Sales Representative	Mr.
7	7	Trump	Robert	Sales Representative	Mr.

Figure 7.1-2: Result of executed procedure

Notice that the 8th record got deleted from the database.



Context :

- Transactions are used when a batch of SQL statements is fired and the requirement is such that if all of them execute successfully then only the database needs to be updated or modified.
- To ensure the database remains and follows the ACID properties so that the data is secure even in case of system failure.



Practice Session/s :

- Write a stored procedure involving transaction where in you update a record from the table - Employees from Northwind database where input parameter is employeeId. If that employeeId is not present in the table then it should rollback the transaction and error message “employeeid does not exist” should be displayed.



Check List :

- Importance of ACID properties.
- The explicit and implicit transaction types abide to the ACID properties.



Common Error/s :

- Syntax errors in either writing the stored procedure or somewhere in the transaction clause.
- Rollback Transaction commands does not work if there is no corresponding Begin Transaction.
- Executing the stored procedure before it is created.



Exception/s :

The following exceptions can rise. The cause and action to be taken has been mentioned below

- **ERROR_INACTIVE_UOW**
Cause: No externally managed transaction is currently active for this thread.
Action: Ensure that the transaction is still active.
- **ERROR_BEGINNING_TRANSACTION**
Cause: Error binding the externally managed transaction.
Action: Examine the internal exception and take appropriate action.



Lesson/s Learnt :

- ☑ In order to make the database consistent and secure, transaction must be used either implicitly or explicitly.
- ☑ Implicit transactions do not require a BEGIN TRANSACTION statement but they need a COMMIT OR ROLLBACK.



Best Practice/s :

- Transactions should be used in order to ensure data integrity and security.
- Commit the transaction after necessary changes are done.



Topic: Locks

Estimated Time – 35 mins.



Objectives : At the end of the session, the participant should understand

- The concept of Locks in SQL Server.
- The kind of transactions held by the Locks.
- Different types of Locks supported by SQL Server.
- The meaning of deadlock.
- Ways of avoiding a deadlock situation during transactions.



Presentation :

- Microsoft's SQL Server use locks to prevent multiple users from making conflicting modifications to a set of data.
- When a set of data is locked by a user, no other users can modify that same set of data until the first user finishes modifying the data and relinquishes the lock.
- By default, SQL Server supports the READ COMMITTED isolation level.
- If the isolation level is changed due to some reasons, then one of the following occurrences might be experienced:
 - **Lost Updates** occur if two transactions modify the same data at the same time, and the transaction that completes first is lost.

Transaction1 Command	Transaction2 Command	What transaction 1 shows	What transaction 2 shows
Begin tran	Begin tran		
select creditlimit from emp where empid =101	select creditlimit, address from emp where emp_d =101	Creditlimit = 5000	Creditlimit = 5000
Update emp set creditlimit = 7500 where empid =101		Creditlimit = 7500	Creditlimit = 5000

End tran	Update emp address = xyz where empid = 101		Creditlimit = 5000
----------	--	--	-----------------------

Figure 7.3-1: Sequence of Transactions

Transaction 2 completes the update to the address. The credit limit of empid 101 is 5000 that is shown to transaction 2. The update has been lost that was made by transaction 1.

- **Non-Repeatable Reads** occur if a transaction is able to read the same row multiple times and gets a different value each time.

Transaction 1 Command	Transaction 2 Command	@Var	What transaction 1 thinks is in the table	Value in Table
Begin Tran		Null		125
Select @var = value from table1	Begin Tran	125	125	125
Set value = value-50	Update value,		75	
If @var >= 100	End Tran	125	125	75
Update value set value = value-100		125	125(waiting for lock to clear)	75
		125	75	Error

Figure 8.3-2: Non-Repeatable Reads

- **Dirty Reads** are a special case of non-repeatable read that occurs while transactions are modifying the data that one is reporting on.

Transaction1 Command	Transaction2 Command	Logical database value	Uncommitted database value	What transaction 2 shows
Begin Tran		3		

Update col = 5	Begin Tran	3	5	
Select anything	Select @var = col	3	5	5
Rollback	Update table1 set col2 = @var	3		5

Figure 7.3-3: Dirty Reads

- **Phantom Reads** occur due to a transaction being able to read a row on the first read, but not being able to modify the same row due to another transaction deleting rows from the same table.
- Below is an example of Phantom reads:

Use Northwind

```
Update Employees Set salary = 5000
where salary < 5000
```

```
Alter table Employees ADD ckSalary (salary > 5000)
```

If a select statement is run, the query will return atleast one row

Someone performs a insert statement at the very same time while update is running. Since it was an entirely new row, it did not have a lock on it and proceeded fine.

- **Types of locks:**

- Intent
- Shared
- Update
- Exclusive
- Schema
- Bulk Update

- **Lock Compatibility Table** below shows the compatibility of the resource lock modes (listed in increasing lock strength). Existing locks are shown by the columns, requested locks by the rows.

	IS	S	U	IX	SIX	X
Intent Shared(IS)	Yes	Yes	Yes	Yes	Yes	No
Shared(S)	Yes	Yes	Yes	No	No	No
Update(U)	Yes	Yes	No	No	No	No
Intent Exclusive(IX)	Yes	No	No	Yes	No	No
Shared with Intent Exclusive(SIX)	Yes	No	No	No	No	No
Exclusive(X)	No	No	No	No	No	No

Figure 7.3-4: Lock Compatibility

- Deadlock occurs when two users have locks on separate objects and each user wants a lock on the other's object.
- **Way to avoid deadlocks include the following:**
 - Ensure the database design is properly normalized.
 - Ensure that the application access server objects are in the same order each time.
 - During transactions, don't allow any user input. Collect it before the transaction begins.



Scenario :

'Wal Mart' is one of the departmental stores. The employees enter and update the details of the customer's deals along with the products. Two of the employees, named Jerry and Alley try to update the credit limit of purchase of the customers from a table.

Both the employees are working on the same table and updating records simultaneously.



Demonstration/Code Snippet :

Below is the deadlock condition (usually in a couple of seconds) that arises and raises an exception in one of the sessions.

Jerie's Session	Time Stamp	Aly's Session
UPDATE customers SET credit_limit = 1200 where	101	

customer_id = 754;		
	102	UPDATE customers SET account_mgr_id = 149 where customer_id = 843;
UPDATE customers SET credit_limit = 1200 where customer_id = 843; Waitng for Aly's session to complete.	103	
	104	UPDATE customers SET account_mgr_id = 149 where customer_id = 754; Waitng for Jerie's session to complete.
Error: deadlock detected while waiting for resource.		

Figure 7.3-5: Showing Deadlock



Context :

- Locks determine the technique in which the resources can be accessed by concurrent transactions.
- Lock method is used in concurrently execution of transactions in order to serialize access to a repository item.
- It is used to ensure transactional integrity and database consistency.
- Locking prevents users from reading data that is being changed by other users, and prevents multiple users from changing the same data at the same time.



Practice Session/s :

- An intranet is provided with only one printer. Two users want to use it for their specific tasks simultaneously. What is the transaction sequence you would suggest them to take, so that no deadlock appears in the system.



Check List :

- Understand the different types of locks and their uses.
- Know importance of using locks to avoid the case of deadlocks.



Common Error/s :

- Only compatible lock types can be placed on a resource that is already locked.
- Some run time error and syntax error can also occur.



Exception/s :

- The locking mechanism is handled implicitly by SQL Server.



Lesson/s Learnt :

- ☒ Importance of locking mechanism in SQL Server.
- ☒ How locks are implemented on different resources which has to be shared.



Best Practice/s :

- Usage of locks to ensure transactional integrity and database consistency.

Crossword: Unit-7

Estimated Time: 10 mins.

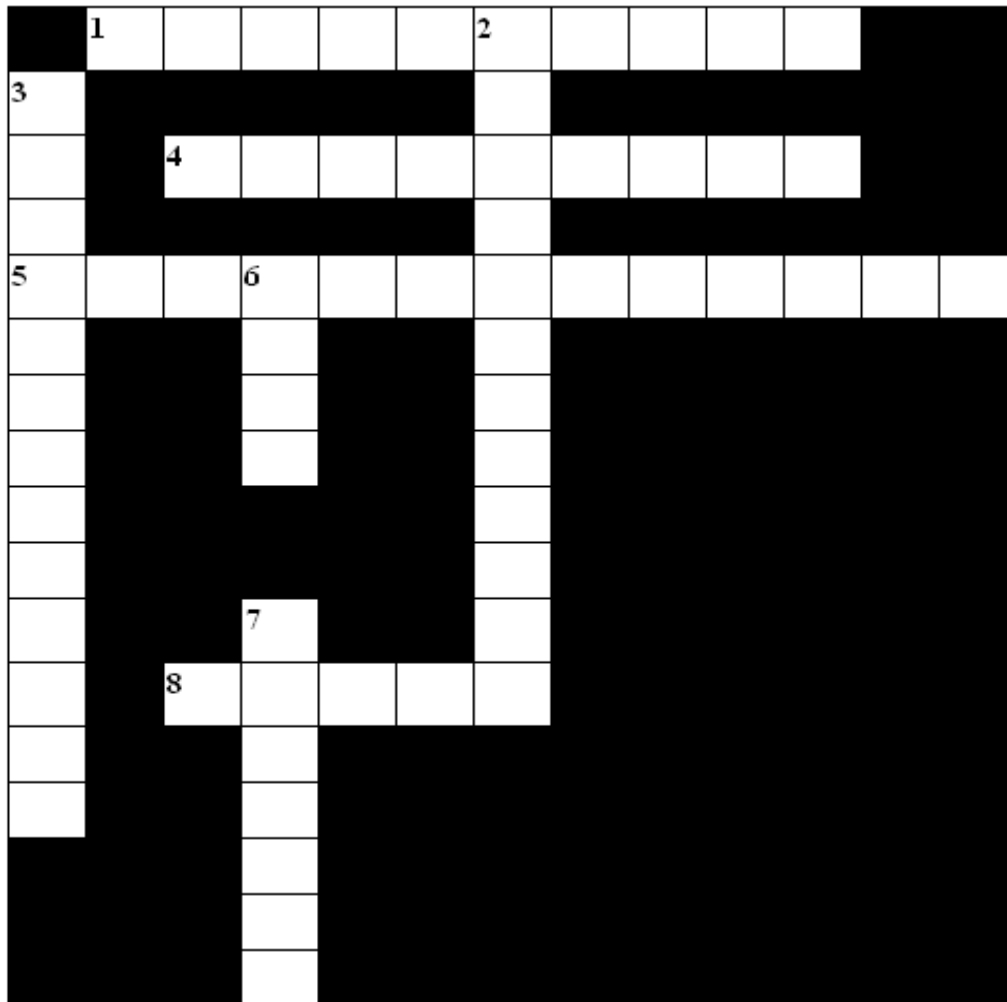


Fig.7-1

Across:

1. What happens if two transactions modify the same data at the same time, and the transaction that completes first is lost? (11)
4. In ACID property, what does 'I' stand for? (9)
5. A property in which no two users should modify the same piece of data at the same time. Nor should they read "dirty" data—modified but uncommitted rows. (13)
8. Which method is used by SQL Server to prevent multiple users from making conflicting modifications to a set of data? (5)

Down:

2. What happens if a transaction is able to read a row on the first read, but not is able to modify the same row due to another transaction deleting rows from the same table? (12)
3. Which isolation level supports default by SQL Server? (13)
6. Every transaction should abide by a group of properties, what is called as (Acronym)? (4)
7. The file in which SQL server write the changes first is called as? (7)

Answers For Crosswords

Unit-1

Across	1) Primary 3) Trace 4) Client Server 5) Transaction 7) Objects 8) Development
Down	2) Management Studio 6) TempDB

Unit-2

Across	3) Compute 4) Waitfor 5) Single 6) Having 7) Break
Down	1) Boolean 2) Truncate

Unit-3

Across	1) Compute 3) Check 4) One 5) Natural 7) Unique 8) Entity
Down	1) Constraints 2) Cross 6) Left

Unit-4

Across	3) Sixth 4) Normalization 5) Default 7) Rule
Down	1) One to Many 2) One to One 3) SP Bindrule 6) Fifth

Unit-5

Across	3) Database Trigger 5) Index 6) Drop 8) After Trigger 9) Alter 10) UDF
Down	1) Partitioned 2) Stored Procedure 4) Scalar 7) Page Split

Unit-6

Across	2) Batch 7) Variables 8) Go
Down	1) Set 3) Cursor 4) Kill 5) Base 6) Catch

Unit-7

Across	1) Lost updates 4) Isolation 5) Data Integrity 8) Locks
Down	2) Phantomreads 3) Read Committed 6) ACID 7) Log File

Team Members



Srikanth Nivarthi
47275



Sreenivas Ram
66223



Seshu Babu Barma
56150



Veerendra Kumar Ankem
77964



Teena Arora
74572

Contributors



Ranjeet Singh
66931



Nihar Vachharajani
72125



Niroja Panda
72108



Rashmi Priya
73008



Vikas Kumar Jha
70920



Business Transformation. Together.