Birla Institute of Technology and Science, Pilani.
Database Systems
Lab No #6

Today's Topics:

- Transactions
- T-SQL
  - Overview
  - Control Constructs
  - T-SQL String Functions and Operators

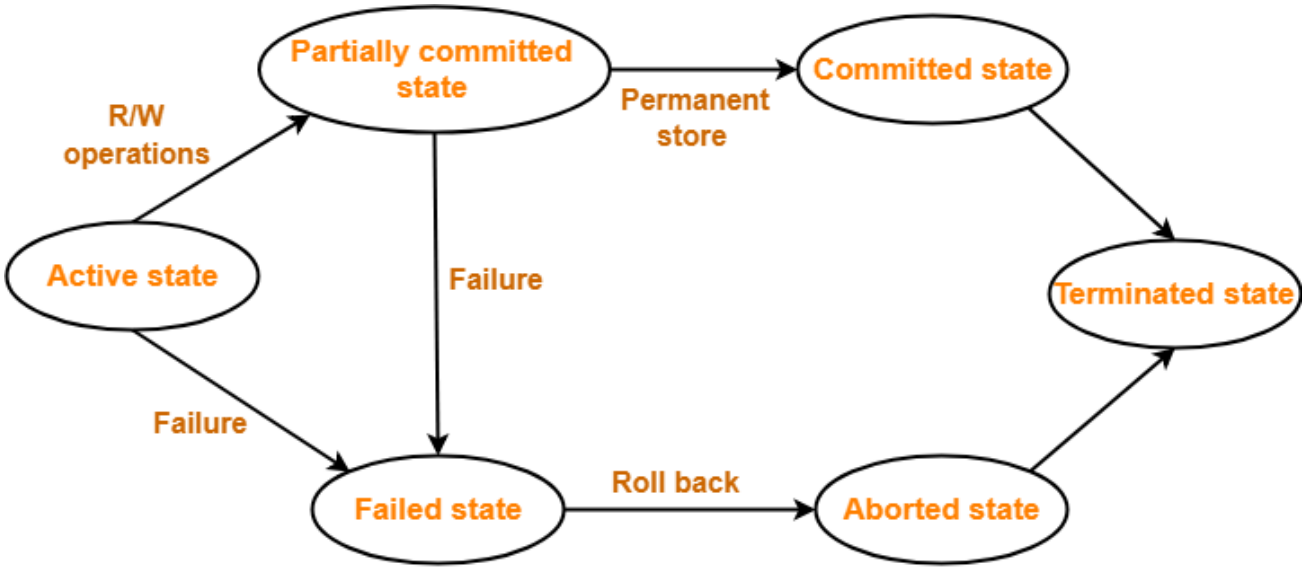
TRANSACTIONS

- Databases are all about sharing data, so it is common for multiple users to be accessing and even changing the same data at the same time. The simultaneous execution of operations is called concurrency. Sometimes concurrency can get us into trouble if our changes require multiple SQL statements. In general, if two or more users access the same data and one or more of the statements changes the data, we have a conflict. This is a classic problem in database systems; it is called the isolation or serializability problem. If the users perform multiple steps, conflicts can cause incorrect results to occur. To deal with this problem, databases allow the grouping of a sequence of SQL statements into an indivisible unit of work called a transaction.

  A transaction ends with either a commit or a rollback:
  - **Commit**—A commit permanently stores all of the changes performed by the transaction.
  - **Rollback**—A rollback removes all of the updates performed by the transaction, no matter how many rows have been changed. A rollback can be executed either by the DBMS to prevent incorrect actions or explicitly by the user.

- The DBMS provides the following guarantees for a transaction, called the ACID properties: Atomicity, consistency, Isolation, and durability. These properties will be covered in the course in detail.

- SQL starts a transaction automatically when a new statement is executed if there is no currently active transaction. This means that a new transaction begins automatically with the first statement after the end of the previous transaction or the beginning of the session.



Transaction States in DBMS

- A user may explicitly start a transaction using the BEGIN TRANSACTION statement.

```
BEGIN TRANSACTION <string>;
```

- Commit:

```
BEGIN TRANSACTION t1;
UPDATE vrs SET inventory = inventory + 10;
SELECT * FROM vrs;
COMMIT;
SELECT * FROM vrs;
```

- Rollback:

```
BEGIN TRANSACTION t2;
UPDATE vrs SET inventory = inventory -20;
SELECT * FROM vrs;
ROLLBACK;
SELECT * FROM vrs;
```

- Here note that without using begin transaction, if you execute the update query alone, the values are automatically committed. There is no way to revert back.

SAVEPOINTS

- SQL allows you to create named placeholders, called savepoints, in the sequence of statements in a transaction. You can rollback to a savepoint instead of going to the beginning of the transaction. Only the changes made after the savepoint are undone. To set a savepoint, use the SAVE TRANSACTION command:

```
SAVE TRANSACTION <savepoint name>
```

- If we create a savepoint, we can rollback to that savepoint with the following:

```
ROLLBACK TRANSACTION <savepoint name>
```

- Executing ROLLBACK without designating a savepoint or executing a COMMIT deletes all savepoints back to the start of the transaction. A rollback to a particular savepoint deletes all intervening savepoints.

```
BEGIN transaction t3;

UPDATE vrs SET inventory = inventory + 25;
SELECT * FROM vrs;

SAVE TRANSACTION spoint1;

UPDATE vrs SET inventory = inventory - 15;
SELECT * FROM vrs;

SAVE TRANSACTION spoint2;

UPDATE vrs SET inventory = inventory + 30;

SELECT * FROM vrs;

SAVE TRANSACTION spoint3;

ROLLBACK TRANSACTION spoint1;

SELECT * FROM vrs;
```

EXERCISES ON EMPLOYEES DATABASE:

- Write a queries for each of the following based on employee database created in the previous labs.

1. Delete the Administration department and all of its sub departments without using transactions.
2. Using a transaction, delete the Administration department and all of its sub departments.

SQL is a standard that has been adopted by ANSI in 1986. It has been used as a base by many RDBMS (relational database management system) vendors. Both PL_SQL and T-SQL are relational database management system and are an extension to the Structured Query Language. The difference that separates PL-SQL from T-SQL is their proprietary. PL-SQL is a product of Oracle, whereas T-SQL is a Microsoft product.

Overview of PL/SQL

PL/SQL is a block-structured language. That is, the basic units (procedures, functions, and anonymous blocks) that make up a PL/SQL program are logical blocks, which can contain any

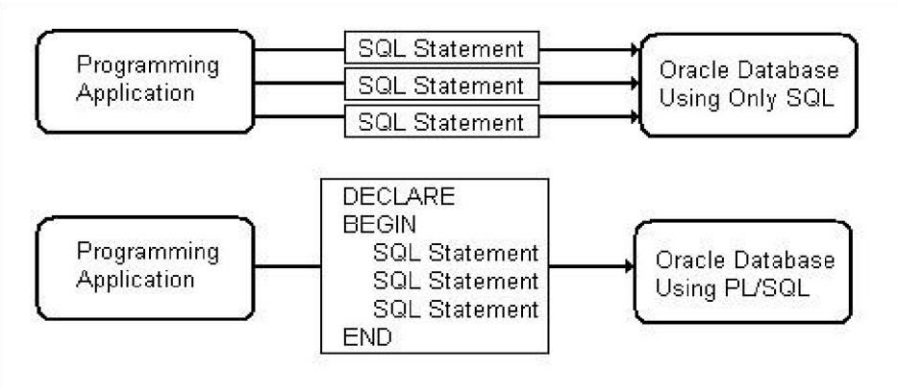
number of nested sub-blocks. Typically, each logical block corresponds to a problem or sub problem to be solved. PL/SQL is not case-sensitive.

PL/SQL provides additional capabilities that SQL lacks:

- Secure code through encryption and by storing code on a server instead of a client computer.
- Handle exceptions that arise due to data entry errors or programming errors. ○ Process record with iterative loop code that manipulates records one at a time. ○ Work with variables, records, arrays, objects, and other common programming language constructs.

These are the advantages of PL/SQL.

- **Better Application Performance:** PL/SQL provides the capability to define a "block" of programming statements, and can transmit this block to an Oracle database as a unit of work. When multiple SELECT statements are part of a PL/SQL block, there are still only two network trips. This improves system response time by decreasing the amount of network and performance overhead that is incurred with an approach where each SELECT statement processes individually.



Productivity, Portability, and Security: PL/SQL stored procedures run on a server instead of a client computer. This means that the procedures can be secured from tampering by unscrupulous hackers by restricting access through the use of standard Oracle database security. Consider a typical business requirement to update a customer order. Instead of granting access to the customer order table, we can grant access to a procedure that has been coded to update the table.

Overview of T-SQL

Transact-SQL is a product **Microsoft**, and it is an extension to SQL. T-SQL is entirely a database programming language which has variables, functions, data definition, data manipulation statements, loops conditional statements and procedures.

T-SQL also features function for string operation, mathematical operations, date and time processing, error checking. This kind of add-ons makes T-SQL turing complete, which means this computing language is universally accepted.

T-SQL performs best when used with Microsoft SQL servers as it is a Microsoft proprietary

Structure of a T-SQL program:

```
DECLARE
```

```
    Declaration statements
BEGIN
    Executable statements  END;
```

/* Every statement must be terminated by semicolon. Identifiers start with alphabets. Comments as shown. Reserved words cannot be used. Statements may span multiple lines*/

To enable desirable output

Anonymous block

```
DECLARE
     @num_age     Integer = 20;  -- assign value to variable BEGIN
       BEGIN
     SET @num_age = 20;
    print ('My age is: ' +str(@num_age)) END;
```

Declarations:

SQL Server data type is an attribute that specifies types of data of any object. Each column, variable and expression has related data type in SQL Server. You can choose a particular data type for a table column based on your requirement.

SQL Server offers seven categories including other category of data types for use.

Exact Numeric Types

Type	From	To
bigint	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
int	-2,147,483,648	2,147,483,647
smallint	-32,768	32,767
tinyint	0	255
bit	0	1
decimal	-10^38 +1	10^38 –1
numeric	-10^38 +1	10^38 –1
money	-922,337,203,685,477.5808	+922,337,203,685,477.5807
smallmoney	-214,748.3648	+214,748.3647

Numeric and decimal are Fixed precision and scale data types and are functionally equivalent.

Approximate Numeric Types

Type	From	To
Float	-1.79E + 308	1.79E + 308
Real	-3.40E + 38	3.40E + 38

Date and Time Types

Type	From	To
datetime(3.33 milliseconds accuracy)	Jan 1, 1753	Dec 31, 9999
smalldatetime(1 minute accuracy)	Jan 1, 1900	Jun 6, 2079
date(1 day accuracy. Introduced in SQL Server 2008)	Jan 1, 0001	Dec 31, 9999
datetimeoffset(100 nanoseconds accuracy. Introduced in SQL Server 2008)	Jan 1, 0001	Dec 31, 9999

datetime2(100 nanoseconds accuracy. Introduced in SQL Server 2008)	Jan 1, 0001	Dec 31, 9999
time(100 nanoseconds accuracy. Introduced in SQL Server 2008)	00:00:00.0000000	23:59:59.9999999

Character Strings

Sr.No	Type & Description
1	char Fixed-length non-Unicode character data with a maximum length of 8,000 characters.
2	varchar Variable-length non-Unicode data with a maximum of 8,000 characters.
3	Varchar (max) Variable-length non-Unicode data with a maximum length of 231 characters (Introduced in SQL Server 2005).
4	text Variable-length non-Unicode data with a maximum length of 2,147,483,647 characters

Unicode Character Strings

Sr.No	Type & Description
1	nchar Fixed-length Unicode data with a maximum length of 4,000 characters.
2	nvarchar Variable-length Unicode data with a maximum length of 4,000 characters.
3	Nvarchar (max) Variable-length Unicode data with a maximum length of 230characters (Introduced in SQL Server 2005).
4	ntext Variable-length Unicode data with a maximum length of 1,073,741,823 characters.

Binary Strings

Sr.No	Type & Description
1	binary Fixed-length binary data with a maximum length of 8,000 bytes.
2	varbinary Variable-length binary data with a maximum length of 8,000 bytes.
3	varbinary(max)
	Variable-length binary data with a maximum length of 231 bytes (Introduced in SQL Server 2005).
4	image Variable-length binary data with a maximum length of 2,147,483,647 bytes.

how to use data types

```
DECLARE
@wages integer,
@hours_worked integer = 40,
@hourly_salary float = 22.50,
@bonus integer = 150,
@country varchar(100),
@counter integer = 0,
@done bit,
```

```
@valid_id bit = 0 -- bit can take values 0/1/NULL BEGIN
      set @wages = (@hours_worked * @hourly_salary) + @bonus;
      set @country = 'France';      print (str(@valid_id));
      print (str(@wages) + ' ' + @country) END;
```

Error Handling: When a runtime error occurs, program control is passed to the exceptionhandling section of the block. The runtime error is then evaluated, and a specific exception is raised or executed.

SQL Server Exception Handling by TRY…CATCH

TRY..CATCH Syntax

```
 BEGIN TRY
--T-SQL statements
--or T-SQL statement blocks
END TRY
BEGIN CATCH
--T-SQL statements
--or T-SQL statement blocks END
CATCH
```

Error Functions used within CATCH block

- ERROR_NUMBER()
  This returns the error number and its value is same as for @@ERROR function.
- ERROR_LINE()
  This returns the line number of T-SQL statement that caused error.
- ERROR_SEVERITY()
  This returns the severity level of the error.
- ERROR_STATE()
  This returns the state number of the error.
- ERROR_PROCEDURE()
  This returns the name of the stored procedure or trigger where the error occurred.
- ERROR_MESSAGE()

  This returns the full text of error message. The text includes the values supplied for any substitutable parameters, such as lengths, object names, or times.

Example:

```
BEGIN TRY
      DECLARE @num INT;
      ---- Divide by zero to generate Error
      SET @num = 5/0;
      PRINT('This will not execute');
END TRY
BEGIN CATCH
      SELECT ERROR_NUMBER() AS ErrorNumber,
```

```
            ERROR_SEVERITY() AS ErrorSeverity,
            ERROR_STATE() AS ErrorState,
            ERROR_PROCEDURE() AS ErrorProcedure,
            ERROR_LINE() AS ErrorLine,
            ERROR_MESSAGE() AS ErrorMessage;
    END CATCH;
```

Control Constructs

IF SYNTAX

```
IF <condition> THEN <action> [ELSE <actions>];
```

WHILE LOOP SYNTAX:

```
WHILE Boolean_expression
    { sql_statement | statement_block | BREAK | CONTINUE }
```

Example of if

```
Declare
 @v_PurchaseAmount float = 1001,
 @v_DiscountAmount float;


BEGIN
      if not(@v_PurchaseAmount <= 1000)
        set @v_DiscountAmount = @v_PurchaseAmount * 0.05;
      else
        set @v_DiscountAmount = 0;
        print ('Discount' + str (@v_DiscountAmount));
end;
```

Example of WHILE LOOP:

```
declare
 @v_Counter integer = 1
begin
 while (@v_Counter < 5)
begin
 print ('count = ' + str(@v_Counter))
set @v_Counter = @v_Counter + 1
```

```
 end
end;
```

Taking Input from SQL

Select the following line and press Ctrl + Shift + M. Declare the variable before execution.

set @userinput = N'<schema_name, sysname, spt_values>'

EXERCISES:

I. Write a T-SQL code to insert data into the temp table. If the entered number is greater than 25 ignore it. If the entered number is less than 25 then insert one row for each number starting from one. Insert (1) in (Num_store), (2) in (Num_store) & so on for all the numbers except 5, 10, 15 & 20. As soon as the number becomes five enter (5, FIVE) in (Num_store, Char_store). If it becomes 10 enter (10, Ten) in (Num_store, Char_store). If it becomes 15 enter (15, FIFTEEN, SYSDATE) in (Num_store, Char_store, Date_store) & when it becomes 20 enter (20, TWENTY) in (Num_store, Char_store).

Table: Temp

Field name	Width
Num_store	number(2)
Char_store	Varchar(15)
Date_store	Date

T-SQL String Functions and Operators

T-SQL offers the concatenation operator (+) for joining two strings. The following table provides the string functions provided by T-SQL:

S.NO	Functions	Purpose
1.	**ASCII(x);**	Returns the ASCII value of the character x.
2.	**CHAR(x);**	Returns the character with the ASCII value of x.
3.	**CHARINDEX(findstring,x,[start_location])**	Searches for find_string in x and returns the position at which it occurs.
4.	**LEN(x);**	Returns the number of characters in x.
5.	**LOWER(x);**	Converts the letters in x to lowercase and returns that string.

6.	**LTRIM(x);**	Trims left spaces of x.
7.	**RTRIM(x);**	Trims right spaces of x.
8.	**REPLACE(x, search_string, replace_string);**	Searches x for search_string and replaces it with replace_string.
9.	**SOUNDEX(x) ;**	Returns a string containing the phonetic representation of x.
10.	**SUBSTRING(x, start, [length]);**	Returns a substring of x that begins at the position specified by start. An optional length for the substring may be supplied.
11.	**UPPER(x);**	Converts the letters in x to uppercase and returns that string.
12.	**LEFT(x,n);**	Left Part of the string x till the specified number of characters n
13.	**RIGHT(x,n);**	Right Part of the string x till the specified number of characters n
14.	**REPLICATE(x,n);**	Repeat string expression will come as output for a given string x with specified number of items n
15.	**REVERSE(x);**	Reverses string x.
16.	**UNICODE(x);**	Returns the Unicode value for the first character of given expression.

The following examples illustrate some of the above-mentioned functions and their use:

Example 1

```
DECLARE @greetings varchar(11) = 'hello world';
BEGIN
```

```
print (UPPER(@greetings)); print
(LOWER(@greetings));

/* retrieve the first character in the string */
print ( SUBSTRING (@greetings, 1, 1));
/* retrieve the last character in the string */

print( SUBSTRING (@greetings, len(@greetings), 1));
/* retrieve five characters,starting from the seventh position. */ print
SUBSTRING (@greetings, 7, 5);
/* retrieve the remainder of the string, starting from the second
position. */
print ( SUBSTRING (@greetings, 2, len(@greetings)));
/* find the location of the first "e" */ print
( CHARINDEX('e', @greetings));
END;
```

---------&----------