Chapter 10

**Indexes** are special data structures used to improve database performance.

* SQL Server and MySQL Server automatically create an index on all primary and foreign keys.
* Additional indexes may be assigned on other columns that are:
  + – Frequently used in WHERE clauses.
  + – Used for sorting data.
* Indexes are created to:
  + – Enforce uniqueness on columns
  + – Facilitate sorting
  + – Enable fast retrieval by column values
* Good candidates for indexes are columns that are frequently used with equal conditions in WHERE clause or in a join.
* Examples: CREATE INDEX CustNameIdx ON CUSTOMER(Name); CREATE UNIQUE INDEX WorkUniqueIndex ON WORK(Title, Copy, ArtistID);
* SQL Server supports two kinds of indexes:
  + – Clustered index: the data are stored in the bottom level of the index and in the same order as that index
  + – Nonclustered index: the bottom level of an index contains pointers to the data
* Clustered indexes are faster than non-clustered indexes for updating and retrieval.
* MySQL Server supports three kinds of indexes:
  + – B-Tree / Hash / R-Tree

The **IDENTITY** keyword implements a new constraint for surrogate keys.

* + – IDENTITY (m, n) creates a surrogate key with an Identity Seed (StartValue) of m and an Identity Increment of n:

CREATE TABLE CUSTOMER( CustomerID int NOT NULL IDENTITY (1000,1), Name char(25) NOT NULL, CONSTRAINT CustomerPK PRIMARY KEY (CustomerID), CONSTRAINT CustomerAK1 UNIQUE (Name) );

The **AUTO\_INCREMENT** keyword implements a new constraint for surrogate keys.

* The increment value is always one.
* The starting value is initially one, but can be modified using an ALTER TABLE statement: ALTER TABLE TRANS AUTO\_INCREMENT = 100;

The **CREATE VIEW** statement can be used to create views in Oracle.

* Unlike SQL-92, Oracle allows the ORDER BY clause in view definitions.
* Oracle supports the JOIN…ON syntax.
* Example:

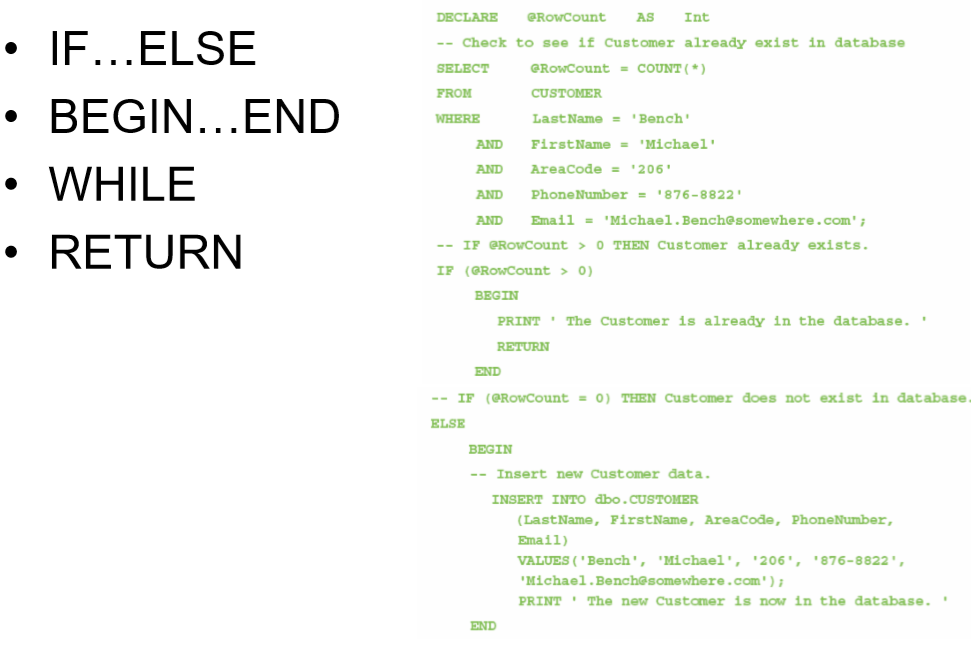
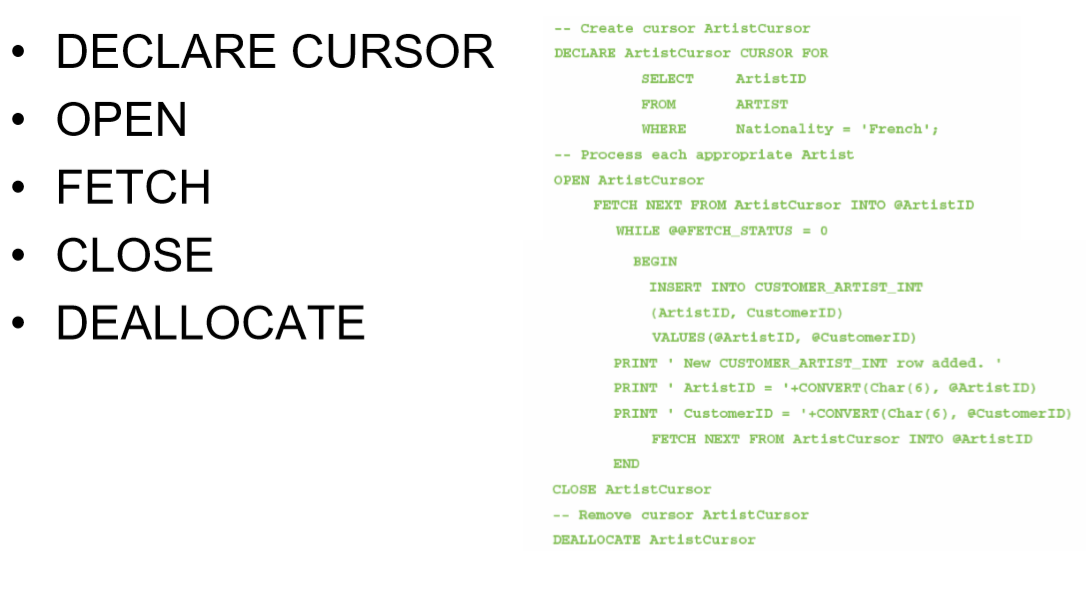
CREATE VIEW CustomerInterests AS SELECT C.LastName AS CustomerLastName, C.FirstName AS CustomerFirstName, A.LastName AS ArtistName FROM CUSTOMER C JOIN CUSTOMER\_ARTIST\_INT CAI ON C.CustomerID = CAI.CustomerID JOIN ARTIST A ON CAI.ArtistID = A.ArtistID;

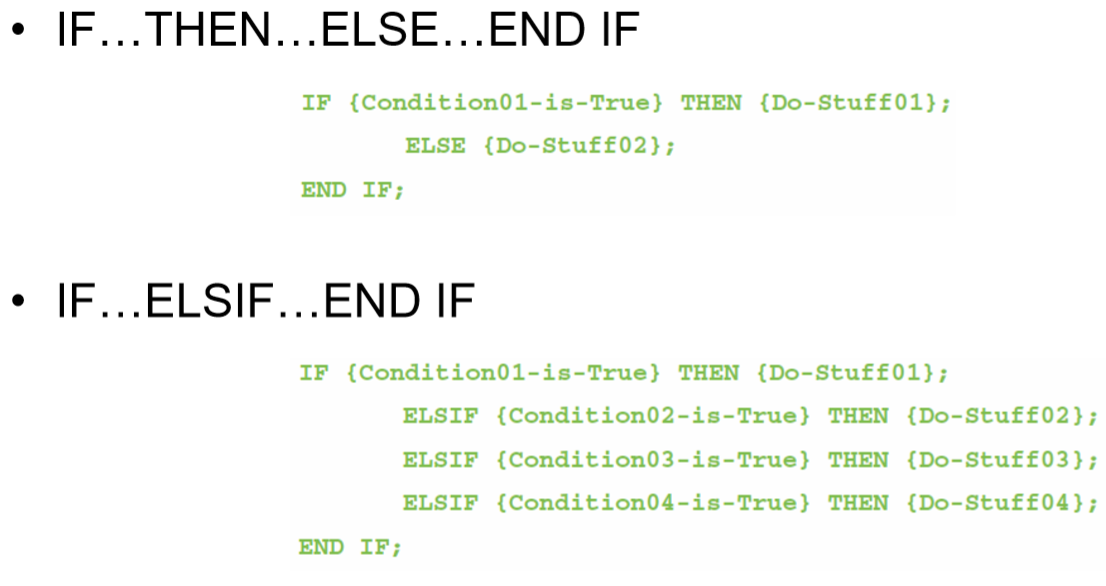
**SQL/Persistent Stored Modules (SQL/PSM)** is an ANSI/ISO standard for embedding procedural programming functionality into SQL.

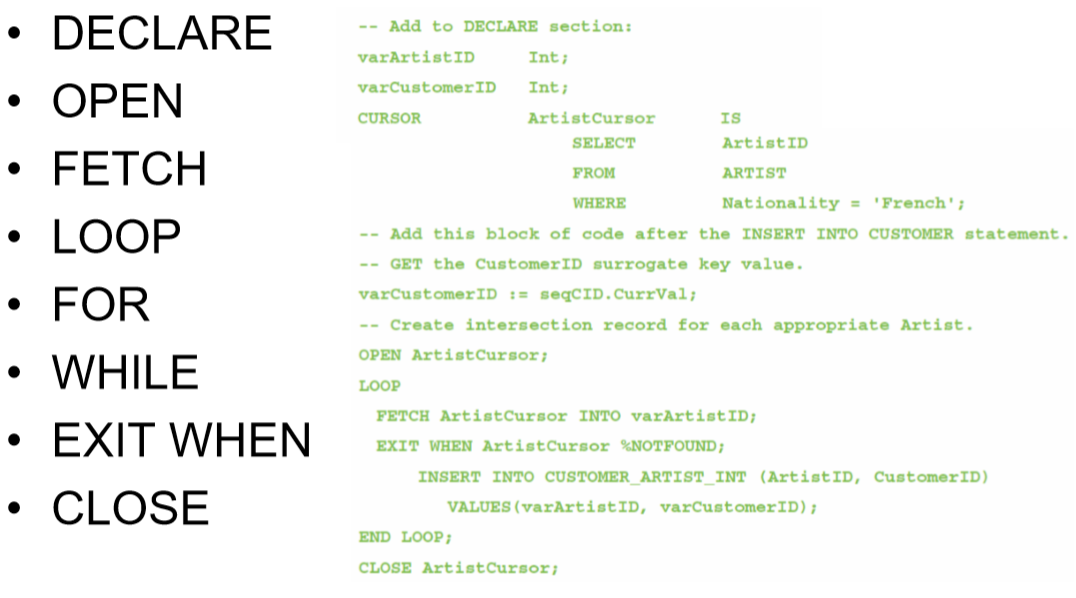
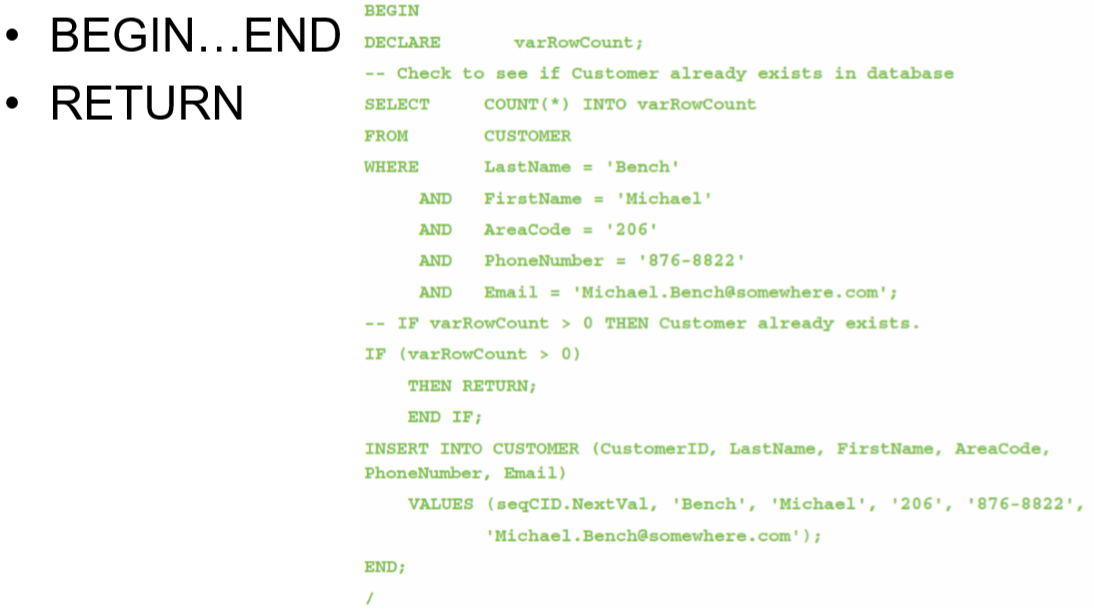
* Each DBMS product implements SQL/PSM in a different way, with some closer to the standard than others.
  + – Microsoft SQL Server 2012 calls its version Transact-SQL (TSQL).
  + – Oracle Database 11g Release 2 calls its variant Procedural Language/SQL (PL/SQL).
  + – MySQL 5.6 implements SQL/PSM, but has no special name for its variant of SQL.

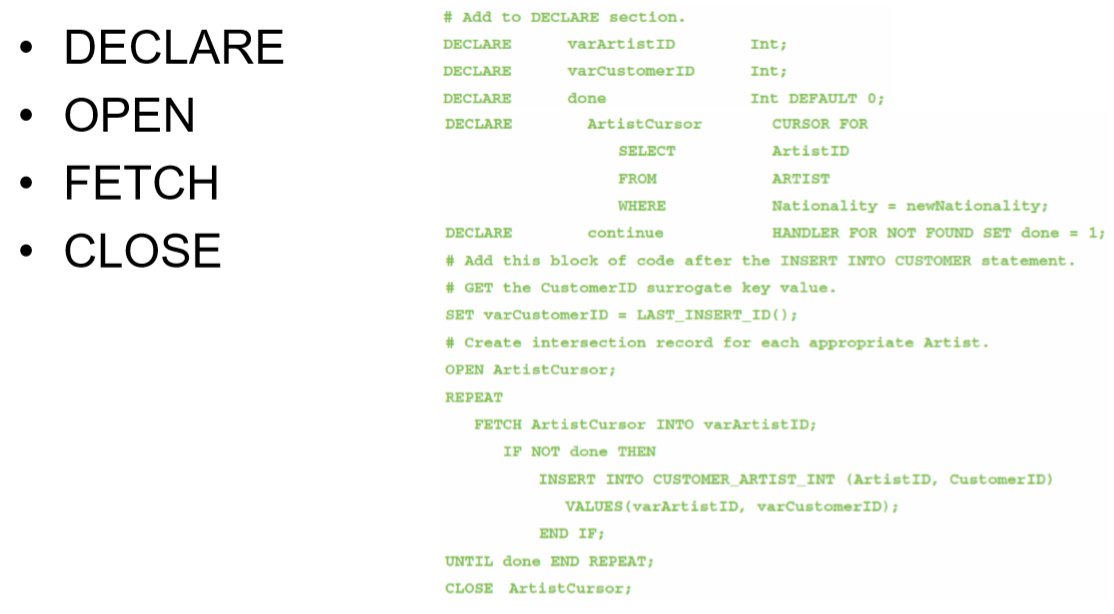
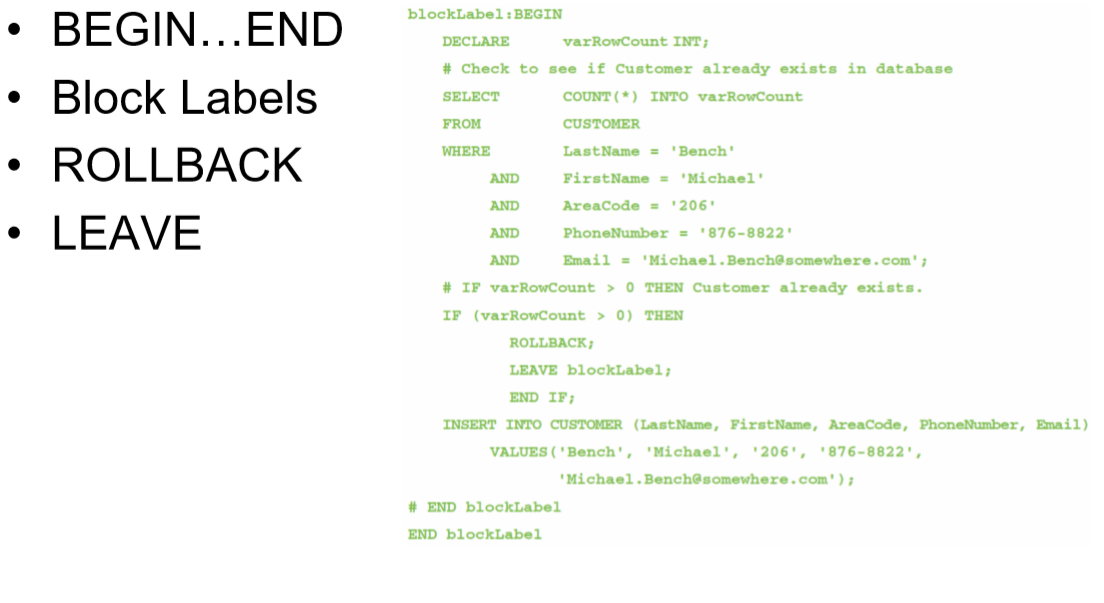
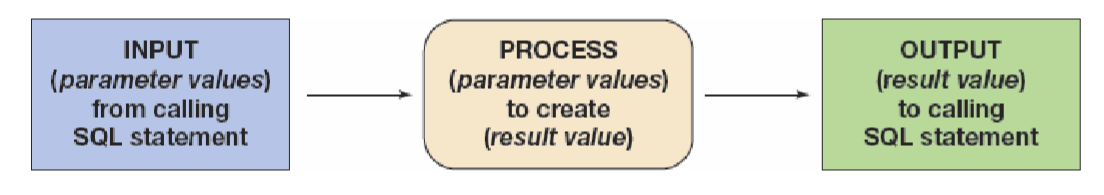
**Application Logic**

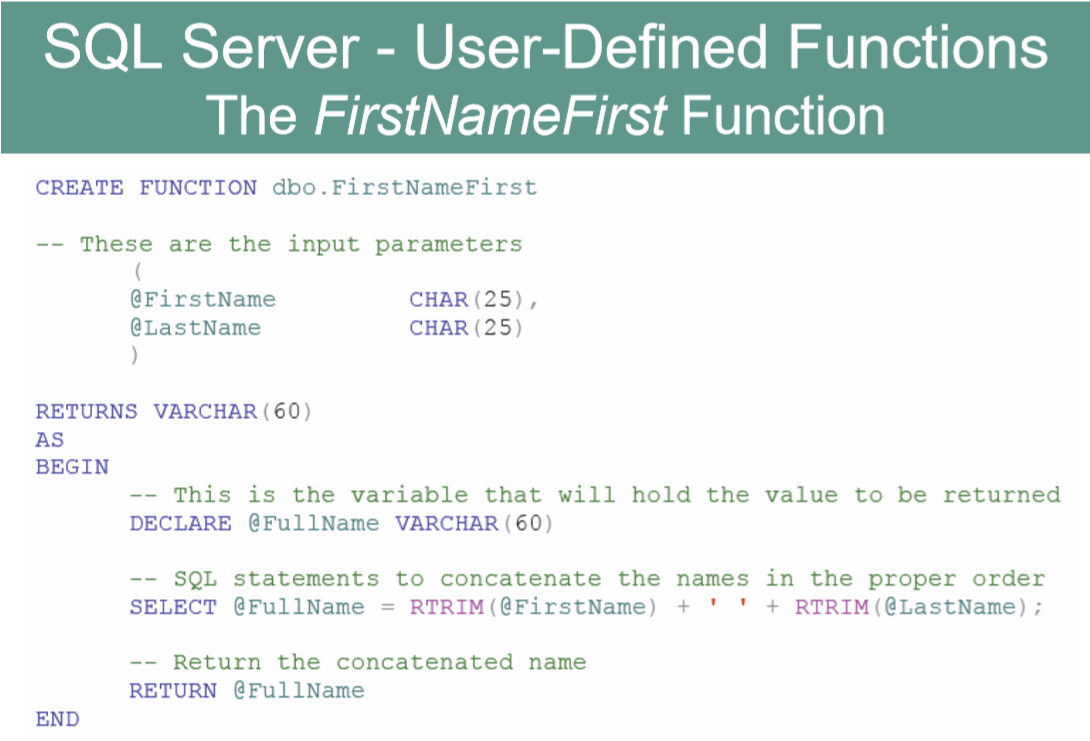
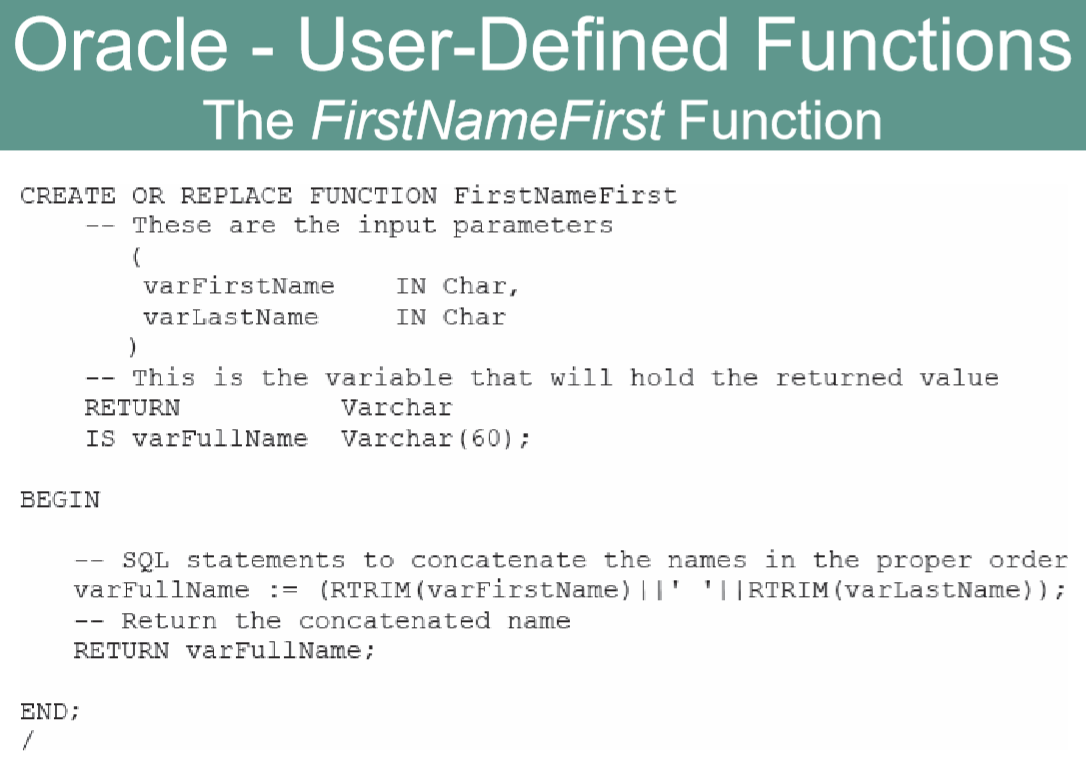
* SQL Server database applications can be processed using:
  + – A programming language, e.g., C# .NET, C++. NET, Visual Basic.NET, to invoke SQL Server DBMS commands
  + – PowerShell and the sqlps utility to invoke database commands stored in .sql files
  + – The Microsoft SQL Server Management Studio to invoke database commands stored in .sql files
  + – Stored procedures
  + – Triggers
* SQL Server implements a version of the standard SQL/Persistent Stored Modules (SQL/PSM)
* SQL Server’s variant is called Transact-SQL (TSQL)

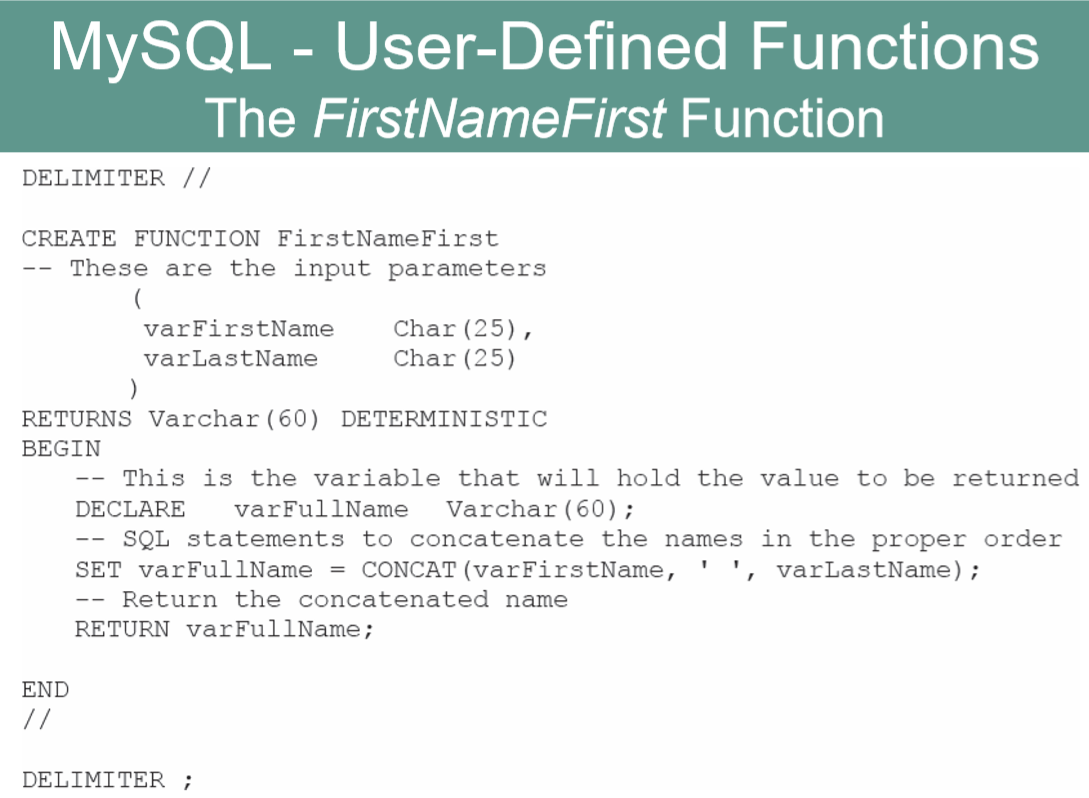


* Oracle Database implements the SQL/Persistent Stored Modules (SQL/PSM) standard in the Oracle Database Procedural Language/SQL (PL/SQL).
* Oracle Database also supports the use of the Java programming language for writing Oracle Database functions, stored procedures and triggers.
* Oracle database applications can be processed using:
  + – Programming languages to invoke DBMS commands
  + – The SQL\*Plus Command Line utility to invoke database commands stored in .sql files
  + – The SQL Developer to invoke database commands stored in .sql files
  + – Stored procedures
  + – Triggers
* A parameter is a value passed to a stored procedure.
* A variable is a value used within the stored procedure.
* Oracle blocks must end with a single slash (/) as a signal to compile and execute the code to create the procedure or trigger



* MySQL Server implements the **SQL/Persistent Stored Modules (SQL/PSM)** standard, but does have a special name for the MySQL variant of SQL—it is just called **SQL**!
* MySQL Server implements the SQL/Persistent Stored Modules (SQL/PSM) and is simply called SQL!
* MySQL Server database application can be processed using:
  + – Programming languages to invoke SQL Server DBMS commands
  + – The MySQL Command Line utility to invoke database commands stored in .sql files
  + – The MySQL Query Browser to invoke database commands stored in .sql files
  + – Stored procedures
  + – Triggers
* MySQL must use a delimiter other than the SQL semicolon(;) when creating stored procedures and triggers
* MySQL blocks must begin with BEGIN and end with END without semicolons.
  + BEGIN DECLARE {optional} {Executable statements {required} END
* A user-defined function (stored function) is a stored set of SQL statements that:
  + – is called by name from another SQL statement
  + – may have input parameters passed to it by the calling SQL statement, and
  + – returns an output value to the SQL statement that called the function.





* Using the FirstNameFirst function:

SELECT dbo.FristNameFirst(FirstName, LastName) AS CustomerName, Sreet, City, State, ZipPostalCode FROM CUSTOMER ORDER BY CustomerName;

A **stored procedure** is a compiled program stored within the database.

* + – Transact-SQL surrounds basic SQL statements with programming constructs such as parameters, variables, and logic structures such as IF and WHILE.
* Stored procedures are programs that can:
  + – Have parameters.
  + – Invoke other procedures and functions.
  + – Return values.
  + – Raise exceptions.
* Creating stored procedures:
  + – Write a stored procedure in a text file and process the commands using the Microsoft SQL Server Management Studio.
* A stored procedure is a PL/SQL or Java program stored within the database.
* Stored procedures are programs that can:
  + – Have parameters.
  + – Invoke other procedures and functions.
  + – Return values.
  + – Raise exceptions.
* A stored procedure must be compiled and stored in the database.
* A stored procedure is a compiled program stored within the database.
* Stored procedures are programs that can:
  + – Have parameters.
  + – Invoke other procedures and functions.
  + – Return values.
  + – Raise exceptions.
* Creating stored procedures:
  + – Write a stored procedure in a text file and process the commands using the MySQL Query Browser
* An **SQL Server trigger** is a procedure that is invoked when a specified database activity occurs.
* Triggers can be used to:
  + – Enforce business rules.
  + – Set complex default values.
  + – Update views.
* – Implement referential integrity actions.
* SQL Server only supports INSTEAD OF and AFTER triggers.
  + – A table may have one or more AFTER triggers.
  + – AFTER triggers may not be assigned to views.
  + – A view or table may have only one INSTEAD OF trigger for each triggering action.
* Triggers can roll back the transactions that caused them to be fired.
* An Oracle trigger is a PL/SQL or Java procedure that is invoked when a specified database activity occurs.
* Triggers can be used to:
  + – Set default values.
  + – Enforce a Data Constraint.
  + – Update a view.
  + – Enforce referential integrity action.
  + – Handle exceptions.
* Trigger types:
  + – A command trigger will be fired once per SQL command.
  + – A row trigger will be fired once for every row involved in the processing of an SQL command.
    - There are three types of row triggers: BEFORE, AFTER, and INSTEAD OF.
    - BEFORE and AFTER triggers are placed on tables while INSTEAD OF triggers are placed on views.
    - Each trigger can be fired on INSERT, UPDATE, or DELETE commands.
* MySQL Server only supports BEFORE and AFTER triggers.
  + – Without the INSTEAD OF trigger, triggers on SQL views are not supported.
* MySQL trigger support is very limited, and triggers may not be used to:
  + – Make a change in the table that fired the trigger.
  + – Return an output value—triggers can use the LEAVE keyword to exit without returning a value.
  + – Make implicit or explicit ROLLBACKs of COMMITs.

**Concurrency control**

* Three factors determine the concurrency control behavior of SQL Server.
  + – Transaction isolation level
  + – Cursor concurrency setting
  + – Locking hints provided in the SELECT clause
* Locking behavior also changes, depending on whether actions occur in the context of transactions or cursors independently.
  + – Therefore, SQL Server places locks on behalf of the developer.
  + – Locks may be placed at many levels of granularity and may be promoted or demoted as work progresses.
* Oracle processes database changes by maintaining a System Change Number (SCN).
  + – SCN is a database-wide value that is incremented by Oracle when database changes are made.
* With SCN, SQL statements always read a consistent set of values; those that were committed at or before the time the statement was started.
* Oracle only reads committed changes; it will never read dirty data.
* One factor determines the concurrency control behavior of MySQL:
  + – Transaction isolation level
* Lock types include:
  + – Record locks—a lock on the index record
  + – Gap locks—generally, a lock on the unused index numbers between index records (see the MySQL documentation)
  + – Next-Key locks—a combined record lock on the index record together with a gap lock on the unused index numbers before the index record itself

A **sequence** is an object that generates a sequential series of unique numbers.

* + Create Sequence CustID Increment by 1 start with 1000;
* It is the best way to work with surrogate keys in Oracle.
* Two sequence methods:
  + – NextVal provides the next value in a sequence.
  + – CurrVal provides the current value in a sequence.
* Using sequences does not guarantee valid surrogate key values because it is possible to have missing, duplicate, or wrong sequence values in the table.
* Creating a sequence:
  + CREATE SEQUENCE seqCID INCREMENT BY 1 START WITH 1000;
* Entering data using a sequence:
  + INSERT INTO CUSTOMER (CustomerID, Name, AreaCode, PhoneNumber) VALUES( seqCID.NextVal, 'Mary Jones', '350', '555–1234');
* Retrieving the row just created:
  + SELECT \* FROM CUSTOMER WHERE CustomerID = seqCID.CurrVal;
* Oracle requires dates in a particular format.
* **TO\_DATE** function may be used to identify the format:
  + TO\_DATE('11/12/2020', 'MM/DD/YYYY')
    - 11/12/2020 is the date value.
    - MM/DD/YYYY is the pattern to be used when interpreting the date.
* The TO\_DATE function can be used with the INSERT and UPDATE statements to enter data:
  + INSERT INTO T1 VALUES( 100, TO\_DATE ('01/05/2020', 'DD/MM/YYYY'));