C#/ SQL+ RDBMS

1. C#
   1. **Delegates** = a reference type that refers to a particular method with a specific return type and parameters
      1. Properties:
         1. Can be static or instantiable
         2. Can be assigned any method that matches the delegate type and is accessible
         3. Fully OO and encapsulate an object instance and method
            1. When instantiated, they’re invokable as if it were the method
         4. “Chainable” – allowing for multiple methods for a single event
      2. Str:
         1. Pass precise methods as arguments for other methods
            1. **Callback functions**
         2. Flexible via **covariance** and **contravariance**
      3. Lim:
      4. **Action delegates** vs **Function delegates**
      5. **Async/await =** keywordsthat enable asynchronous programming that allows concurrent tasks to be executed while other code is running
         1. Properties
            1. Non-parallel vs parallel asynchronous work:

Cooking breakfast. One person (thread) can complete all tasks to complete cooking breakfast by starting each task as soon as ready to progress **vs**

Requires multiple people (threads), where each person is responsible for 1 task that progresses when a different task is completed

* + - * 1. Ought to be employed whenever accessing the Web, a database, or working with files/images
      1. Str
         1. Important for tasks that are **potentially blocking**,
      2. Lim
    1. Threads
  1. XML
     1. Properties
     2. Str
     3. Lim
  2. Event Handlers
     1. Properties
     2. Str
     3. Lim
  3. Lambda Expressions = an anonymous function that takes input parameters from the L of the lambda operator and executes an expression/operation on the R of the operator. (Usually in place of a delegate)
     1. Properties
        1. Lambda operator: =>
           1. Intuitively: “goes to” or “becomes”
           2. Parameter types must be explicit (int x/string y) or implicit
     2. Statement Lambda
        1. Cannot be used for **expression trees**
     3. Expression Lambda
     4. Str
     5. Lim
  4. **Predicate**
  5. **Anonymous types**

1. SQL + Database Management
   1. **Normalization**
      1. Properties
      2. Str
      3. Lim
   2. **DML**
      1. Properties
      2. Str
      3. Lim
   3. .LINQ
      1. Properties
         1. Queries are *constructs,* like classes & objects
         2. Queries require an instantiated data source, a defined query expression, an executing via a **foreach** statement
         3. .LINQ allows **query variables**
         4. Query Syntax
            1. Basic SQL syntax – start w/ FROM and end with SELECT. Data type is not important, but the application always reads the query as Ienumerable<T>. **Whenever possible**
         5. Method-based Syntax (**Whenever necessary)**
      2. Str
         1. You can perform ordering, sorting, and filtering operations w/ minimal code
         2. Syntax and expressions are similar to SQL and other Db language, so you can easily translate knowledge
      3. Lim
   4. RDBMS
      1. Properties
      2. Str
      3. Lim
   5. ACID = a set of principles that guide database design so that database transaction occur, even in the event of errors or system failures.
      1. Properties
         1. Atomicity – a transaction executes on an “all-or-nothing” principle
         2. Consistency – the database transforms from 1 consistent state to another consistent state, after every transaction
         3. Isolation – Transactions should be independent and have no interaction/impact on each other
         4. Durability – Committed transactions will persist even in system failures, typically after a specialized logging procedure that to complete unfinished operations
      2. Str
      3. Lim
   6. Repo Pattern =
      1. Properties
      2. Str
      3. Lim
   7. Dependency Injection = A technique to reduce straight code dependency by achieving **inversion control** between classes & dependencies
      1. Properties
         1. Dependency: Anything an object requires in order to properly function
         2. Achieved by:
            1. Utilizes an interface or class to abstract the dependency implementation
            2. Dependency registered in service container .
            3. Service *injected* into class constructor where it’s used. Dependency is instantiated and discarded when no longer needed.
         3. Service Type Lifetimes:
            1. Transient -
            2. Scoped -
            3. Singleton -
      2. Str
      3. Lim
         1. Avoid static & stateful class members
   8. ORM
      1. Properties
      2. Str
      3. Lim
   9. EF
      1. Properties
      2. Str
      3. Lim
   10. Fluent API
       1. Properties
       2. Str
       3. Lim
   11. Scaffolding
       1. Properties
       2. Str
       3. Lim
   12. Input Validation
   13. View = SQL Table that is virtual and doesn’t allow you to actually change your data
       1. Computed Column = a column that is the result of some computation on another table. Not physically stored in a table unless its PERSISTED.
          1. Requires AS to make a column as a CC
       2. Computed Table
       3. Properties
          1. Compu
          2. Schema binding = prevents changes from happening to underlying tables
       4. Str
       5. Lim
   14. Stored Procedures =
       1. Scalar variables
       2. Properties
          1. 4 types:
             1. 1
             2. 2
             3. 3
             4. 4
       3. Str
          1. Can have SELECT & DML statements
          2. Input & Output parameters
             1. Exactly the same as in & out parameters in c#
          3. Can have Exception Handling
          4. Can use table variables and temporary variables
          5. Can call functions
       4. Lim
          1. Can’t be called from SELECT/WHERE
          2. Can’t be used in JOIN
   15. SQL Functions = Function stores in SQL that outputs a scalar or table as a result
       1. Properties
          1. Scalar Function: takes in 1 or more parameters, operates on a single value, and outputs a scalar{
             1. Accepts user parameters
             2. Performs an action
             3. Returns the result of the operation as a scalar or table}
             4. Can be used wherever a query is valid
          2. Aggregate functions
          3. Table-Valued Functions (https://bit.ly/3cz5I8j)
             1. Inline
             2. Multi-statement
       2. Str
          1. Instead of inputting a commonly-used complex function into every query, functions allow use to streamline code & be more efficient by invoking just the function
          2. Organized into DB under the programmability folder
          3. Can use IF and WHILE logic blocks
          4. Can call other functions
          5. Can be easily modified using ALTER
       3. Lim
          1. Have readonly access – cannot input or alter the database
          2. Cannot output multiple result sets
          3. Does not support error handling via “try..catch”, @error, or @raiseerror
          4. Cannot use SET statements
   16. Scalar = a single value of any datatype that resides in a table
   17. Table-valued parameter = a function parameter that is a table
   18. Creating a function:
       1. CREATE FUNCTION + name
       2. Parameter selection: Choose parameters/columns/factors to be used in operation
       3. RETURN datatype
       4. RETURN statement: operational content of the function
          1. Make sure the RETURN statement gives only **one** output by using parentheses correctly as needed
       5. End
   19. 