Week two notes

Monday 4/29/19

* Disposable
* Using block
* Serialization, file I/O, XML, JSON
* Review async
* SoC, KISS, DRY, CML comments

Unmanaged resources

* Open file
* When we want control over when that resources is freed

IDisposable

* Dispose has to be called by programmer if this interface implemented
* Using statement/block calls dispose method
* “Disposable pattern” -look up best practice for

Using statement/block

Using(var httpClient=newHttpClient())

{

//open file

} //file is closed

Separation of concerns

* Loose coupling
* Maintainability
* Unit testability

DRY - Don’t Repeat Yourself

Documentation/documentation tools like DocFX(VS,VS Code parse these for hover-tooltips)

//regular comments

///<summary> adds two integers</summary>

///<remarks></remarks>

/// <param name=”a”>The First integer</param>

///<returns>The sum</returns>

///<throws name=” “>when </throws>

Public int Add(int a, int b)

{

}

Serialization

Writing data to a file?

* Requirements
* Pick a good enough format to reconstruct object from data you output to file
* Formats
  + XML
  + JSON
  + YAML
  + Roll your own text(avoid this when you can)
  + Rol your own binary(also avoid)
  + CSV
* XMLSerializer (old, deprecated, doesn’t support generics)
* JSON.NET aka Newtonsoft JSON (popular json for serialization using .net)

NuGet

.csproj

PackageReference

Under EACH project right click dependencies>manage NuGet packages

(have to install newtonsoft for each project/solution)

GAC (Global Assembly Cache)-global storage for all third party .dll assemblies on your computer

All methods that call async have to (async “infects” everything else)

Process for when writing asynchronous code (best when we do disk/network access,)

1. Look for method that ends in “Async” instead of the regular one
2. Await the returned task
3. This method needs “async” modifier
4. The return type is wrapped in a Task<…>
5. By convention, your async method should itself be named “….Async”

Using git in visual studio 2019

Tuesday 4/30/2019

SQL (structured query language)

* relational
* table, columns with data types, rows
* language for talking to data
* many variants: MySQL, SQLite, Transact-SQL(T-SQL, aka SQL Server), Oracle SQL, SAQL, PostgreSQL, we mainly stick to Transact-SQL(T-SQL, aka SQL Server)
* server software for SQL “SQL server” not Microsoft, refers to any servers that can host SQL databases

we created Azure sql database

named guerrero1904sql>resource group training-rg>location south central US>no elastic pool>basic>sample data(leave none for future)

sql commands

* --comments--
* --many statements in one file like this…
* The execute button F5 will run all the statements and show all the outputs
* When we don’t want to run the whole file, we highlight what we want to run and then hit F5
* SQL has many commands/statements
* The first category of them is called DML (Data Manipulation Language)
  + DML is short name of **Data Manipulation Language** which deals with data manipulation and includes most common SQL statements such SELECT, INSERT, UPDATE, DELETE, etc., and it is used to store, modify, retrieve, delete and update data in a database.
  + Most complicated/important DML statement is the SELECT statement
  + \* selects all rows/columns from a table

SQL data types

* Integer
  + TINYINT(1 byte)
  + SMALLIT(2 byte)
  + INT(4byte)(usually use this one)
  + FLOAT
  + REAL
  + DEECIMAL/NUMERIC(n,p)
    - (configurable, usually use this)
    - E.g. DECIMAL (5,3) is a number with enough precision for 5 decimal digits, with 3 of them after the deximal point. E.g. “xx.xxx”
    - DECIMAL(max)-maxprecision
* Currency
  + Money
  + CHARACTER(n)/CHAR(n) – fixed length string
    - CHAR(10) has space for 10 characters in every value
  + VARYING CHARACTER(n)/VARCHAR(n)
    - When we say ‘abc’, that is a VARCHAR literal
    - Doesn’t waste extra space
    - We can put (max)
  + NATIONAL CHARACTER(n)/NCHAR(n)
    - Allows any Unicode character
  + NVARCHAR(n) – always use this one in reality
    - Allows Unicode and doesn’t waste space
    - If I need a literal string in SQL with Unicode characters
    - I can use N’abc”
* Date/time
  + DATE for dates
  + TIME for times
  + DATETIME for timestamps
    - Don’t use this one, precision and domain are too small
  + DATETIME2(n)-has precision parameter
  + The max is 9?
  + DATETIMEOFFSET – for intervals of time
* Operators
  + Equals =
  + Not equals != or <>
  + And AND
  + OR or
  + < <= > >=
    - WHERE Year BETWEEN 2009 AND 2011 (includes both ends <= and >=)
  + Get parts of dates and times out of their types
    - EXTRACT(YEAR FROM OrderDate)

Come back to this add from trainer code

Select Clauses

* + Select
  + From
  + Where

SQL – Week 2 topics

* RDBMS-table, row, column, PK, FK, referential integrity, anomalies, multiplicity
* Normalization-dependancy, candidate key, 1NF, 2NF, 3NF
* DML-select, insert, update, delete, truncate
* DDL-create, alter, drop
* Join-inner, left,right, full, cross, self
* Set operators-union, intersect, except, all,
* Subquery-CTE
* Aggregate functions-average, count, sum, min, max
* View-schema binding,
* Function-parameters, scalar, tabular
* Stored procedure-variable, if-else, try-catch, throw, raiseerror, print
* Trigger-before, after, instead of,
* Transaction-commit, rollback, savepoint
* ACID-atomic, consistent, isolated, durable
* Isolation-read\_uncommited, read\_commited, repeatable, serializable,
* Repository patern
* ADO.NET-connected,disconnected, DataReader, DataAdaptor, DataSet
* Entity Framework- ORM, DB-first, DBContext, SBSet, tracking, LINQ, deferred execution, testing

In SQL, indexes are 1 based

In t-sql, we can escape spaces in an identifier with double quotes or with brackets

Count (anything), when Group by is around, means the number of rows that were combined into the given result row

When we

Group by runs after where

If we want to filter again, after aggregation, we use HAVING

Logical order of execution of a select statement

1. From
2. Where
3. Group by
4. Having
5. Select
6. Order by

When we insert into a table, we need to provide a value for all columns that are not nullable and that have no default configured’’

DML-all the commands that work directly with rows

Select, Insert, update, delete, Truncate (debatable)

Truncate empties table but leaves behind column definitions, constraints, triggers, etc (harder to recover from using log files unless using savepoint)

JOINS

Cross join-“combines multiplies rows by columns”

Inner join (inner is implied if only type join)

Left join (left outer join)

Right join(right outer join)

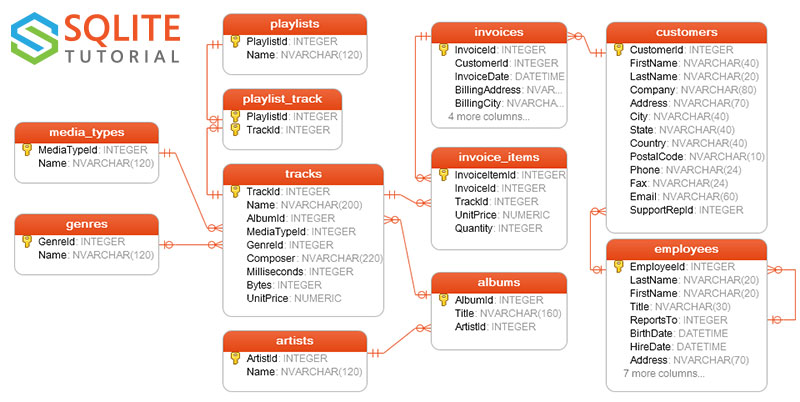
Full join

Database

* Schema
  + Table

<schema>.<table> this is the dbo default

Chinook database diagram

+++-

Check sql queries in trainer-code repository

--set operators

These operators are going to suppress duplicates and implement “set union” “set intersetion” and “set difference”

Union

We can switch off the removing of duplicates

Every set operator in some other versions of SQL has a regular (“distinct”) version, and an ALL version

In sql server only union has an ALL version

That is “set union” – like a big “or”

Set intersection is like a big AND

Set difference implemented by EXCEPT (in some other variants of SQL, Minus)

Subqueries

Every join query can be converted into a subquery-based query

We just use queries inside other queries…with the help of some operators

* In
* Not in
* Any
* All
* Exists

Similar to subquery is “common table expression” (CTE)

Thursday 5/2/19

Functional dependency- relationship between two sets of columns:

set of columns x and set of columns y

y is dependent of x

X-->Y

For each possible set of x values, there is exactly one possible set of y values

If two rows had all the same x, then their y must also be the same

Customer Id🡪FirstName

Candidate key(CK)-minimal set of columns which every other column in the table depends on

Can serve to uniquely identify the row

We choose one to do that and call it the primary key(PK)

Any key with >1 column we also call a “composite key”

Draw arrow between supplier and phone-two way arrow

Arrow from name to supplier column-one way

Name-candidate key-draw arrow from name to supplier and from name to color

Color and supplier are dependant on name

Phone and supplier dependant on each other

Product

|  |  |  |  |
| --- | --- | --- | --- |
| Name (PK, FK) | supplier | Color | phone |
| Widget | Google | Red, black | 555-1234 |
| Thingybob | Amazon | Blue | 555-4321 |
| Knickknack | Google | Blue, red | 555-1234 |

A poor db design has anomalies:

* Update anomaly-duplicate data updated inconsistently
* Insert anomaly-no way to insert some data that unnecessarily requires other data
* Delete anomaly-deleting some data unnecessarily deletes other data

Normalization

3NF, 2nf,1nf

Non-CK columns must depend on…”the keys, the whole keys, and nothing but the keys”

First normal form:

* All rows must be unique, and, there is a PK
* Atomic values
* No repeating groups of columns

Supplier is dependant on name?

Supplier and phone depend on each other?

|  |  |  |  |
| --- | --- | --- | --- |
| Name (PK, FK) | supplier | Color | phone |
| Widget | Google | Red | 555-1234 |
| Thingybob | Amazon | Blue | 555-4321 |
| Knickknack | Google | Blue | 555-1234 |
| Widget | Google | Black | 1234 |
| Knicknack | Google | Red | 1234 |

Second normal form: (no partial dependencies)

* 1NF
* No non-CK column can be partially dependent on any CK
* Only can happen with composite CKs

Product Color

Both make up composite key

|  |  |
| --- | --- |
| Name | Color |
| W | R |
| W | Bla |
| T | Blu |
| K | Blu |
| K | R |

Product

|  |  |  |
| --- | --- | --- |
| Name | supplier | phone |
| Widget | Google | 555-1234 |
| Thingybob | Amazon | 555-4321 |
| Knickknack | Google | 555-1234 |

Third normal form

* 2NF
* No non-CK can depened on any other non-CK columns
* No transitive dependencies

Supplier

|  |  |
| --- | --- |
| Supplier(PK here) (FK in product) | Phone |
| G | 1234 |
| A | 4321 |

Product Color

Both make up composite key

|  |  |
| --- | --- |
| Name (FK in product) | Color |
| W | R |
| W | Bla |
| T | Blu |
| K | Blu |
| K | R |

Product

|  |  |
| --- | --- |
| Name (PK) | Supplier (PK in supplier) |
| Widget | Google |
| Thingybob | Amazon |
| Knickknack | Google |

Foreign key(FK)

* The columns of another table’s pk, but in this table
* Establishes references/relationships between rows in different tables

**Not necessary, but recommended next step (add IDs)**

Product

|  |  |  |
| --- | --- | --- |
| ID | Name (PK) | SupplierID |
| 1 | Widget | 1 |
| 2 | Thingybob | 2 |
| 3 | Knickknack | 1 |

Supplier

|  |  |  |
| --- | --- | --- |
| ID | Supplier | Phone |
| 1 | G | 1234 |
| 2 | A | 4321 |

Product Color

|  |  |
| --- | --- |
| Name (FK in product) | Color |
| W | R |
| W | Bla |
| T | Blu |
| K | Blu |
| K | R |

Referential integrity

**DDL**

**Data definition language**

* GO is for running code in different batches(maybe not DDL?)
* we can use alter table to add/delete columns, modify things
* constraints
* not null: null not allowed in the column
* Null: explicitly allowing null for documentation (Already there by default, maybe not really a constraint)
* Primary key-enforces uniqueness and NOT NULL, sets clustered index
* Unique: the column cannot have any duplicate values
  + (can be set on sets of columns, as well as just one)
* Default: provide a default value for this column
  + (either this or null is necessary when adding a new column to a table that already has data)
* Foreign Key-can set “cascade” behavior…ON DELETE CASCADE, ON DELETE SET NULL, ON UPDATE…
* CHECK: catch all, any Boolean expression you want to write to validate the values in a row. Is checked every time a row is updated or inserted
* IDENTITY(start=1, increment=1) :
  + useful for integer primary keys.
  + --prevents inserts or updates on the column, and gives automatic incrementing ID
  + It is possible to switch on IDENITY\_INSERT

Multiplicity

* One-to-one…both in same table
* 0/1..1: two tables, FK NOT NULLL, UNIQUE
* Many-to-one (1..N): two tables, FK NOT NULL
* Many-to-many(N..N) three tables, two FK

Check .sql files in trainer-code view for below

Computed column-a computed column for the name and date in this format: Name:(Year)

View functions- view is a virtual table based on the result-set of an SQL statement.

Variables-only have scope for current batch of commands, temporary

-table value variables-return temp table

User defined functions-must be readonly, no modification of table data permitted

Triggers-

Stored procedures

These allow modification to the data base

But, they can only be executed with EXECUTE statement

But they do not need to return anything

RaiseError, try catch, throw

While loops, maybe for loops

Procedures

Declare, execute, ?

Know what transactions are:

Transaction: ACID properties

* Atomicity: succeeds 100% or none at all, no partial application of operations
* Consistency: constraints of DB, referential integrity, etc. are preserved
* Isolation: if multiple transactions run at once, they must not interfere
* Durability: the transaction is not complete until its changes are made permanent(in non-volatile memory)

ex.

1. Check account balance
2. Decrement account
3. Increment other act

In SQL, statements are transactional and we can make our own

Isolation anomalies/violations

* “dirty read”- When one transaction sees the uncommited changes of another transaction
* Nonrepeatable read- when one transaction sees the uncommitted changes of another transaction
* “nonrepeatable read”-when one transaction reads the same row twice and sees different things (because of another started and completed transaction)
* Phantom read-when one transaction runs the same query twice and sees extra rows the second time(because another trans. Inserted them)

Isolation levels

Read\_uncommitted (allows dirty read)

Read\_committed (prevents dirty read) 🡨default

Repeatable (prevents nonrepeatable read)

Serializable (prevents phantom read)

+locks

+Overhead

Isolation levels allow for: +Possibility of deadlock

+Slower

+safer for data

Begin tran

… savepoint

…

… rollback

Commit tran

Know what RDBMS is

Relational db management system

Active directory objects?

ADO

Data adapter (DataReader)

Dataset

Connected architecture vs disconnected

Connected faster?

Object relational manager(ORM)

Entity framework

Dapper is another ORM (we don’t use this)

Entity framework

Code-first approach(“new database”) creates db based on your C# classes

Db-first approach(“existing database”) “reverse engineering”

Scaffold-DBContext

Hooking your project up with EntityFramework (also see EfEntries.App in trainer-code) using Db-first approach(need to have your db created already in azure/mssql)

Add secretconfiguration.cs to app project(still using hacky way)make sure to add it to gitignore in this project

Add Microsoft.entityframeworkcore.SqlServer nuget package to your startup project(consoleapp)

Microsoft.entityframeworkCore.Tools on DataAccessProject also add MicrosoftEntityFrameworkCore.design for command line

Maybe add these to each project to be safe

Open Package manager console (type it in search bar)

**Make sure you Switch to dataAccessproject in package manager console**

Console app has to reference data access: to do this: For consoleapp rightt click dependencies, add reference. Check box for dataAccess, ok

Input: (can run scaffold command many times when need to change db)

Scaffold-DBContext -Connection “connection string goes here” -Provider Microsoft.EntityFrameworkCore.SqlServer -StartupProject EfIntro.App(this is your appname, just example -OutputDir Entities -Force)

Scaffold-DbContext -Connection <connection-string-in-quotes> -Provider Microsoft.EntityFrameworkCore.SqlServer -StartupProject <name-of-startup-proj> -OutputDir Entities -Force

Scaffold-DBContext -Connection “Server=tcp:guerrero1904sql.database.windows.net,1433;Initial Catalog=MovieDb;Persist Security Info=False;User ID=francisco;Password=<password goes here>;MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;” -Provider Microsoft.EntityFrameworkCore.SqlServer -StartupProject EfIntro.App

Go into dbcontext file that’s auto generated(its in DataAccess project)

Deleted onconfig method completely (maybe gices compile error)

DBContext (database)

DbSet<Movie> (table)

Movie (row)

Go to program.cs

Add methods in program.cs to connect to database

To add entity framework logging

add package "Microsoft.Extensions.Logging.Console"

log levels

critical/fatal

error

warn

info

debug

trace

avoid putting anything in consolueUI you won’t want sticking around

customer repository in ours most likely

custRepo.GetAll()

{

foreach(var item in db.Customer

db.Customer)

//pull out objects here and set them to class values

}

Return;

Recommended code for recommended db layout

Do try catch around save changes

Implementing methods

Using

For entity framework

Repository classes example of some code that you could put

Var l=dbContext.Location

.Include(l => l.inventoryItem)

.ThenInclude(i=>i.Product)

First(l=>l.Id==locationId);

Mapping code example you could put in location?

Return new Lib.Locations

{

ProductStocks=…

dbLoc.InventoryItem

.Select(i=>i.Product)

}

Git checkout .

Removes all local changes or ignores them?