Module 2 Day 5

Database Design

What makes an application?

- Program Data
  - ✓ Variables & .NET Data Types
  - ✓ Arrays
  - ✓ More Collections (list, dictionary, stack, queue)
  - ✓ Classes and objects (OOP)
- Program Logic
  - ✓ Statements and expressions
  - ✓ Conditional logic (if)
  - ✓ Repeating logic (for, foreach, do, while)
  - ✓ Methods (functions / procedures)
  - ✓ Classes and objects (OOP)
  - ❑ Frameworks (MVC)

- Input / Output
  - User
    - ✓ Console read / write
    - ❑ HTML / CSS
    - ❑ Front-end frameworks (HTML / CSS / JavaScript)
  - Storage
    - ✓ File I/O
    - ❖ Relational database
    - ❑ APIs

Design Exercise

- Take 15 minutes
- Discuss with the person next to you
- What should a DB that supports this look like?

Gallery Customer History Form

Customer Name

Jackson, Elizabeth
123 – 4th Avenue
Fonthill, ON
L3J 4S4

Phone (206) 284-6783

Purchases Made

Artist	Title	Purchase Date	Sales Price
03 - Carol Channing	Laugh with Teeth	09/17/2000	7000.00
15 - Dennis Frings	South toward Emerald Sea	05/11/2000	1800.00
03 - Carol Channing	At the Movies	02/14/2002	5550.00
15 - Dennis Frings	South toward Emerald Sea	07/15/2003	2200.00

The Gill Art Gallery wishes to maintain data on their customers, artists and paintings. They may have several paintings by each artist in the gallery at one time. Paintings may be bought and sold several times. In other words, the gallery may sell a painting, then buy it back at a later date and sell it to another customer.

Database Design Checklist

- ❑ Eliminate duplicative columns from the same table
  - This includes multiple values stuffed into a single column, or columns with names like phone1, phone2, etc.
  - Create a new table for this data and identify PK for it
- ❑ Make sure every column depends on the entire PK
  - Not just part of the PK (this is called a Partial Dependency)
  - Put that data into a separate table with PK, and add FK to the current table
- ❑ Make sure every column is not dependent on a non-key column
  - This is called a Transitive Dependency

Database Normalization

- A process used to organize a database into tables and columns
- A table should be about a *specific* topic and no more
- Minimizes data duplication (redundancy)
- Simplifies queries
- Avoids data modification anomalies
- https://www.essentialsql.com/get-ready-to-learn-sql-database-normalization-explained-in-simple-english/
- https://en.wikipedia.org/wiki/Database_normalization

Database Normalization

- First Normal Form (1NF) - The information is stored in a relational table with each column containing atomic values. There are no repeating groups of columns.
- Second Normal Form (2NF) - The table is in first normal form and all the columns depend on the table's primary key.
  - Single purpose
- Third Normal Form (3NF) - The table is in second normal form and all its columns are not transitively dependent on the primary key
  - Car Id – make – model

DDL – Data Definition Language

- Create and drop databases

CREATE DATABASE database_name

DROP DATABASE database_name

DDL – Create and Drop Tables

```
CREATE TABLE table_name (
        column_name1 int IDENTITY,
        column_name2 data_type(size) NOT NULL,
        column_name3 data_type(size),
CONSTRAINT pk_name_1 PRIMARY KEY (column_name1),
CONSTRAINT fk_name_1 FOREIGN KEY (column_name2) REFERENCES
table_name2(column_1)
)
```

```
DROP TABLE table_name
```

DDL – Alter Table

```
ALTER TABLE table_name ADD CONSTRAINT pk_constraint_name
        PRIMARY KEY (column_name(s))
```

```
ALTER TABLE table_name ADD CONSTRAINT fk_constraint_name
FOREIGN KEY (column_name) REFERENCES table(column_name)
```

```
ALTER TABLE table_name ADD CONSTRAINT chk_constraint_name
CHECK (column_name = 'value' OR column_name IN (values))
```