Module 1 Coding Assessment

- Validates your understanding of required concepts
- Prepares you for on-site assessments during interviews
- This is an ***Individual*** Assessment
  - No collaboration on this one
  - You will do *one* of the assessments included in the folder
- In-class, One hour
  - Add, Commit and Push your code when finished!
  - Make sure your code compiles!
- Project is in
  - {student-c}\module-1\Assessment

Module 2 Day 1

Introduction to Databases

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

File I/O

- Persistence: We were able to save and load program or object state
- Read in, update data, write out
- However:
  - We could not easily “share” the data among many users
  - We could not easily locate one small piece of the file and update only that
    - E.g., Just to complete one task, we had to write them all back out.
  - What if we had a lot of data?
    - What if I had tasks that I had completed over the past year? I’d have to load all that data just to find the current tasks
    - What if we stored tasks for thousands of users? How would we find mine?

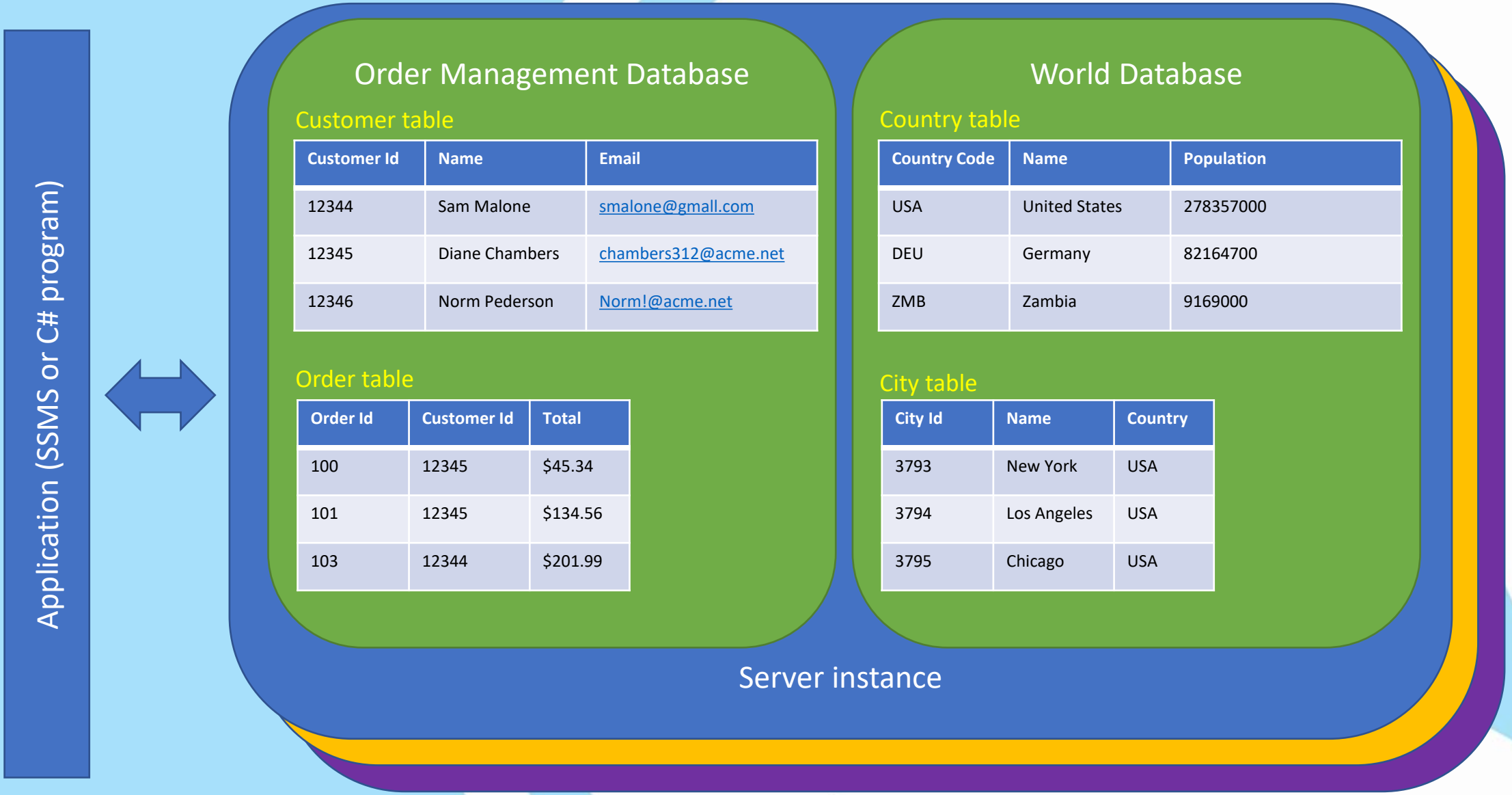
Database Management Systems - DBMS

- Special software specifically designed to **manage data**
- Handles very large amounts of data
- Shared access
- Quick retrieval and update
- Security
- Data Integrity – constraints and transactions
- Various types of DBMS
  - Relational, No-SQL, OO, Hierarchical, Analytical

Relational DBMS - RDBMS

- Microsoft SQL Server, PostgreSQL, Oracle, MySQL, DB2
- SQL – Structured Query Language
  - To define database structure
  - To Create, Read, Update and Delete data
  - To manage data access
- **Table** - Stores all the data for a specific type of entity (e.g., a Car)
- **Column** - Represents a data field (make, model, year)
- **Row** - represents a single entity (‘Honda’, ‘CRV’, 2005)
- Think spreadsheet

Relational DBMS Structure



SQL Server Column Data Types

- char, varchar, nchar, nvarchar
- int, decimal, bigint, money
- float
- date, datetime
- bit
- https://docs.microsoft.com/en-us/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-2016

SQL Server / C# Data Types

- https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/sql-server-data-type-mappings

SQL Server	C#		SQL Server	C#
bit	bool		char/nchar	string
date	DateTime		datetime	DateTime
decimal	decimal		float	double
int	int		money	decimal
ntext/text	string		nvarchar/ varchar	string
tinyint	byte			

Structured Query Language - SQL

- *Declarative* language (C# is *imperative*)
- DDL – Data Definition Language
  - Create Table, Create Index, Create Constraint, Alter Table
- DML – Data Manipulation Language
  - **C**reate, **R**ead, **U**pdate and **D**elete data
  - Create – Insert
  - Read – Select
  - Update – Update
  - Delete – Delete
- DCL – Data Control Language
  - Used for controlling access to data

SQL Server Management Studio

- Launching SSMS
- What you see:
  - Current database
  - Object Explorer
  - Query windows
  - Results window
- Creating a database from the UI
- Creating a database from a script
- Running one or many queries – Select / F5 / Execute
- Comments --

SELECT – Read Data

- SELECT column1, column 2… | *
  FROM table
  WHERE search_condition1 AND | OR search_condition2…
  ORDER BY column3, column4…
- WHERE search condition
  - =, <>, !=, >, >=, <, <=
  - IN (values / select), NOT IN (…)
  - BETWEEN value1 AND value2 (this is inclusive)
  - IS NULL, IS NOT NULL
  - LIKE 'search string' (see docs)
- AS 'Col-Name'
- DISTINCT, TOP nnn
- CAST / CONVERT (see docs)