Module 2 Day 3

Table Relationships – Keys, Cardinality and Joins

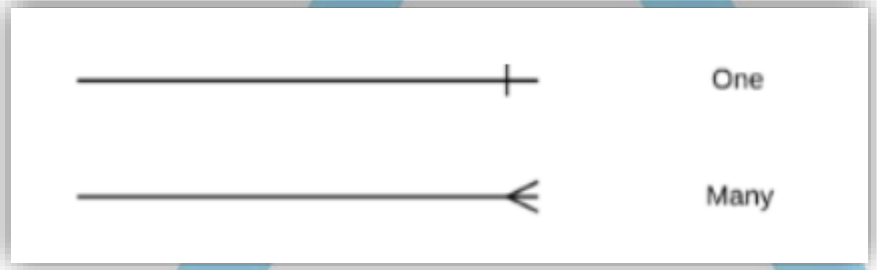
What makes an application?

- 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

Relationships and Cardinality

- A **Relationship** is an association between two tables using keys
- **Cardinality** is the number of occurrences in one entity which are associated to the number of occurrences in another.
  - 1 : 1
  - 1 : many
  - Many : many



- What are the relationship cardinalities in World?
- ERD – Entity Relationship Diagram

Keys – Important Points

- Used to create relationships between tables
- **Primary Key:** uniquely identifies each row within a table.
  - Cannot hold duplicate values across rows (must be unique)
  - Cannot be null
- **Foreign Key:** references a primary key in the source table.
  - May have multiple rows with the same value in this column
  - May be a nullable column
- What keys do we see in the World database?

Keys – Other Stuff

- Made up of one or more columns
- A **composite key** is a key made up of multiple columns
- Natural vs. Surrogate
  - **Natural Keys** are formed from values in the real world (e.g. SSN, ISBN)
  - **Surrogate Keys** are artificially created by the application to identify a unique record

Many-to-many Relationships

- Cannot be modelled directly between the two tables
- Association table is used “between” the two primary tables
  - Association table holds foreign keys to each primary table
  - Often these are the only columns in the table

Joins

- Connect two tables together using keys
- Matches row by row, finding all the keys that match
- Produces a “super-row” containing all the columns from both tables

```
SELECT table1.cols, table2.cols, ...
FROM table1
JOIN table2 ON table1.fkcolumn = table2.pkcolumn
```

- You may need to use table name to identify the correct column
  - Here is where you can use a table ALIAS

Inner vs. Outer Joins

- INNER JOIN is default (when only JOIN is specified)
  - Only rows that match both tables
  - Display capital cities and the country of which it is the capital

```
SELECT c.name AS capital, ctry.name AS country
FROM country ctry JOIN city c ON ctry.capital = c.id
```

- LEFT OUTER JOIN / RIGHT OUTER JOIN
  - All rows from the Left (First) table, even if there is no match on the Right (Second) table
  - Display all cities and the country of which it is the capital, or NULL if it is not

```
SELECT c.name AS city, ctry.name  country
FROM city c LEFT OUTER JOIN country ctry ON c.id = ctry.capital
```

- CROSS JOIN (RARE)
  - Relates every row in one table to every row in the other table (Cartesian product)

Union

- Combines two or more queries into a single result set
- The number and data type of columns must be identical
- Columns will be named based on the first query
- “Union” removes duplicate rows, “Union All” does not
- Order By is allowed on the last query only

```
SELECT expression1, expression2, ... expression_n
FROM tables
[WHERE conditions]
UNION
SELECT expression1, expression2, ... expression_n
FROM tables
[WHERE conditions]
```