Module 2 Day 4

Updating Data

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

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]
```

Insert

- INSERT INTO table_name (column1, column2, ...)
  VALUES (value1, value2, ...)
- Column-list is optional but highly recommended
- Same number of items in the column-list as the value-list
  - Values line up positionally
- IDENTITY columns must not be specified
- There are other forms of INSERT but this is the most common

Update

- UPDATE table_name
  SET column1 = value1,
  column2 = value2, …
  WHERE column = value
- Don’t forget the WHERE!!!!!
  - Unless it is your desire to update every row in the DB
- One or more column-value pairs can be listed
  - Other columns will remain untouched
- There are other forms of UPDATE but this is the most common

Delete

- DELETE FROM table_name
  WHERE column = value
- Don’t forget the WHERE!!!!!
  - Unless it is your desire to delete every row in the DB

Constraints

- Defined when the table is defined
- Constrain in some way the data that may be stored in the table
- NOT NULL – null values are not allowed in this column
- UNIQUE – the same value cannot exist in this column on multiple rows.
- Check – additional rule checking to ensure a column value is acceptable. E.g., and acceptable range of integer values
- Default – if an INSERT does not supply a value for the column, this value will be stored by default

Referential Integrity (RI) Constraints

- PRIMARY KEY – Defines the column(s) to be a PK.
  - Automatically enforces UNIQUE and NOT NULL constraints
- FOREIGN KEY – “Points” to a PK of some table
  - Constrains INSERT and UPDATE such that invalid references may not be stored
  - Constrains the DELETE on the PK table such that you cannot delete a PK value that is referenced elsewhere

Transactions

- **A**tomic – a single unit of work; entirely succeeds or entirely fails
- **C**onsistent – database is in a valid state before and after the transaction. No constraints are violated.
- **I**solated – transaction is independent from other transactions. Outsiders cannot see “partial” results.
- **D**urable – Once completed, the transaction is saved even if there is a crash.

Transactions

- BEGIN TRANSACTION
  -- Do some work here….
  INSERT …
  UPDATE …
  DELETE …
  yada yada…
  COMMIT TRANSACTION

- If ROLLBACK TRANSACTION is used instead of COMMIT, the entire transaction is abandoned

Schema Information

- ISO standard views for getting information about a db schema
  - SELECT * FROM INFORMATION_SCHEMA.TABLES
  - SELECT * FROM INFORMATION_SCHEMA. COLUMNS
  - SELECT * FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS
  - SELECT * FROM INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE
  - SELECT * FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS
- https://docs.microsoft.com/en-us/sql/relational-databases/system-information-schema-views/system-information-schema-views-transact-sql?view=sql-server-2016