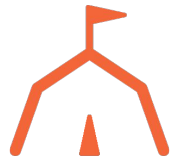
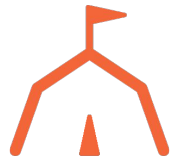
C#.NET BOOTCAMP

Databases

SQL Server

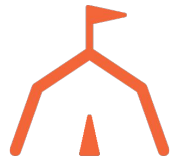


Intro To SQL Server



Topics

- Introduction to SQL server
- Creating a database
- Data manipulation

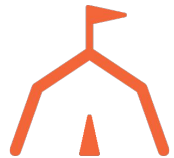


Relational Databases

- A relational database consists of one or more tables that consist of rows (records) and columns (fields).
- These table are related by keys.

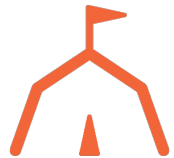
Relational Databases

- The primary key in a table is the one that uniquely identifies each of the rows in the table.
- A foreign key is used to relate the rows in one table to the rows in another table.



SQL Server Provides

- Support for SQL (Microsoft Transact SQL, or T-SQL)
- Support for multiple clients
- Connectivity
- Security
- Referential integrity
- Transaction processing

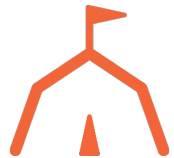


SQL Server Tools

- SQL Server - The SQL Server database server, which manages databases and tables, controls user access, and processes SQL queries.
- SQL Server Management Studio

Installing SQL Server

- You can download the Express edition
- Step by step installation



Data Types

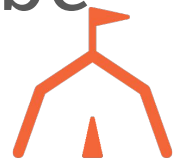
SQL Server Data Types

Selecting a data type for your fields

- Properly defining the fields in a table is important to the overall optimization of your database.

SQL Server Data Types

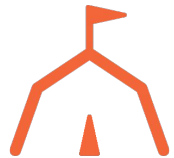
You should use only the type and size of field you really need to use; don't define a field as 10 characters wide if you know you're only going to use 2 characters. These types of fields (or columns) are also referred to as data types, after the type of data you will be storing in those fields.



SQL Server Data Types

SQL Server uses many different data types broken into three categories:

- Numeric.
- Date and time.
- String types.

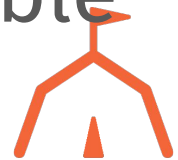


Numeric Data Types

- SQL Server uses all the standard ANSI SQL numeric data types
  - *INT (4 byte)*: A normal-sized integer that can be signed or unsigned. If signed, the allowable range is from -2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295.

Numeric Data Types

- SQL Server uses all the standard ANSI SQL numeric data types
  - *TINYINT (1 byte)*: A very small integer that can be signed or unsigned. If signed, the allowable range is from -128 to 127. If unsigned, the allowable range is from 0 to 255.

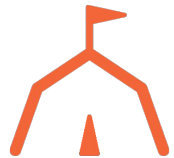


Numeric Data Types

- *SMALLINT (2 byte)* - A small integer that can be signed or unsigned. If signed, the allowable range is from -32768 to 32767. If unsigned, the allowable range is from 0 to 65535.

Numeric Data Types

- *BIGINT (8 bytes)* - A large integer that can be signed or unsigned. If signed, the allowable range is from -9223372036854775808 to 9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615.



Numeric Data Types

- *FLOAT* - A floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 10,2, where 2 is the number of decimals and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a FLOAT.

Numeric Data Types

- *REAL* - A double precision floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 places for a DOUBLE. REAL is a synonym for DOUBLE.

Date and Time Types

- *DATE* - A date in YYYY-MM-DD format, between 1000-01-01 and 9999-12-31. For example, December 30th, 1973 would be stored as 1973-12-30.

Date and Time Types

- *DATETIME* - A date and time combination in YYYY-MM-DD HH:MM:SS format, between 1000-01-01 00:00:00 and 9999-12-31 23:59:59. For example, 3:30 in the afternoon on December 30th, 1973 would be stored as 1973-12-30 15:30:00.

String Types

- *NCHAR(M)* - A fixed-length string between 1 and 255 characters in length (for example NCHAR(5)), right-padded with spaces to the specified length when stored. Defining a length is not required, but the default is 1.

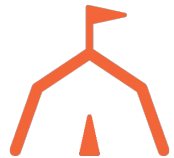
String Types

- *NVARCHAR(M)* - A variable-length string between 1 and 255* characters in length; for example NVARCHAR(25). You must define a length when creating a VARCHAR field.
- *old limit

String Types

- *Binary* or *TEXT* - A field with a maximum length of 65535 characters. BLOBs are "Binary Large Objects" and are used to store large amounts of binary data, such as images or other types of files. Fields defined as TEXT also hold large amounts of data; the difference between the two is that sorts and comparisons on stored data are case sensitive on BLOBs and are not case sensitive in TEXT fields. You do not specify a length with BLOB or TEXT.

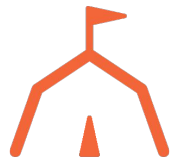
Creating Databases



Creating Databases

Use SQL server to create databases

Creating Tables



Creating Tables

```
CREATE TABLE table_name
(column_name column_type);
```

```
create table tutorials_tbl(
   tutorial_id INT NOT NULL ,
   tutorial_title NVARCHAR(100) NOT NULL,
   tutorial_author NVARCHAR(40) NOT NULL,
   submission_date DATE,
   PRIMARY KEY ( tutorial_id )
);
```

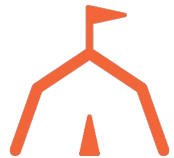
Create and Drop A Table

- We use the *CREATE TABLE* statement to create a table and the *DROP TABLE* statement to delete a table.
- We can use the *DROP TABLE* IF EXISTS statement to guard against an error resulting from attempting to delete a table that does not exist.

Select

- `SELECT` is to retrieve data from a table.
- This is achieved by using the `SELECT ... FROM` keywords along with some qualifying information.

```
SELECT column_name FROM table_name;
```



SQL Select Example

id	name	hero_name	primary_power
1	Bruce Banner	Hulk	Strength
2	Steve Rogers	Captain America	Tactician
3	Thor	Thor	Lightening
4	Tony Stark	Iron Man	Genius
5	Natasha Romanov	Black Widow	Marksman
6	Clint Barton	Hawkeye	Archer

```
SELECT hero_name FROM Avengers;
```

```
SELECT hero_name, primary_power FROM Avengers;
```

Wildcard

To return all columns, use the wildcard (*) operator.

```
SELECT * FROM Avengers;
```

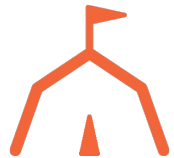
Where

Filter the data using the where clause

```
SELECT * FROM Avengers WHERE name='Bruce Banner';
```

Append AND/OR operators to WHERE statements

```
SELECT * FROM Avengers WHERE primary_power='Strength' OR name='Clint Barton';
```



Order By

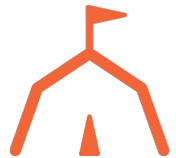
Order By will order the returned data in a specific way.

```
SELECT name, population FROM cities WHERE population > 1000000 ORDER BY population DESC;
```

Insert

INSERT INTO is the command used to put data into a table.

```
INSERT INTO table_name (column_name1, column_name2, ...) VALUES ('value1', 'value2', ... );
```

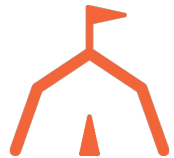


Insert

name	hero_name	primary_power
Bruce Banner	Hulk	Strength

What SQL statement would be used to add this row to the table:

name	hero_name	primary_power
Peter Parker	Spider-Man	Mouthing off



Insert

```
INSERT INTO Avengers(name, hero_name, primary_power)
VALUES('Peter Parker', 'Spider-Man', 'Mouthing off');
```

Update...Set

To modify an existing row in an existing table, use the UPDATE keyword.

```
UPDATE Avengers SET primary_power='Web Slinging' WHERE hero_name='Spider-Man';
```

The new row would now look like this:

name	hero_name	primary_power
Peter Parker	Spider-Man	Web Slinging

Delete From

To delete an existing row, use the DELETE FROM keywords.

```
DELETE FROM Avengers WHERE hero_name='Hawkeye';
```

Removes the entire record (all columns) from the database.

CRUD

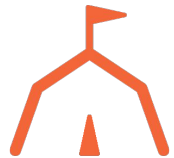
CRUD is a term that refers to the following operations:

- Create (Add a row)
- Read (Retrieve)
- Update (Modify a row)
- Delete (Delete a row)

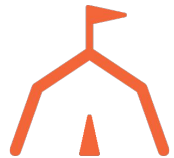
CRUD

These basic operations reflect the standard database operations (INSERT, SELECT, UPDATE, DELETE).

Almost every app will be, at its core, a CRUD app.

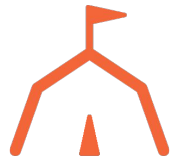


Pop Quiz
SQL



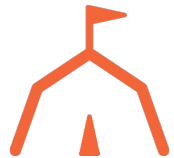
Pop Quiz

Get the name of the Avenger who's hero_name is Hawkeye.



Pop Quiz

List all the Avengers in alphabetical order by hero_name.



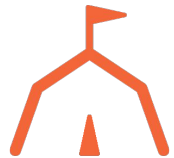
Pop Quiz

Add a new Avenger.

- name: Cluck Kent
- hero_name: Grant Chirpus
- primary_power: Finding Bugs

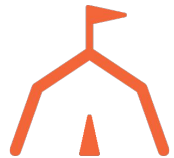
Pop Quiz

Change Thor's primary_power to Hammer.

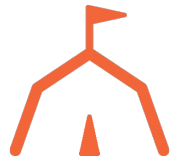


Pop Quiz

Remove Black Widow



Column Operators



Tickets

id	seat	price	num_sold
1	Box Level	105	4
2	Dress Circle	75	2
3	Main Floor	58	10
4	Mid Balcony	38	0
5	Upper Balcony	19	3

Count

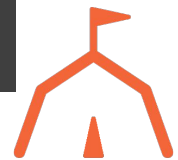
id	seat	price	num_sold
1	Box Level	105	4
2	Dress Circle	75	2
3	Main Floor	58	10
4	Mid Balcony	38	0
5	Upper Balcony	19	3

```
SELECT COUNT(*) FROM Tickets;
```

Count

id	seat	price	num_sold
1	Box Level	105	4
2	Dress Circle	75	2
3	Main Floor	58	10
4	Mid Balcony	38	0
5	Upper Balcony	19	3

```
SELECT COUNT(*) FROM Tickets WHERE num_sold <> 0;
```



Sum

id	seat	price	num_sold
1	Box Level	105	4
2	Dress Circle	75	2
3	Main Floor	58	10
4	Mid Balcony	38	0
5	Upper Balcony	19	3

```
SELECT SUM(num_sold) FROM Tickets;
```

Average

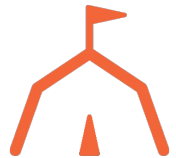
id	seat	price	num_sold
1	Box Level	105	4
2	Dress Circle	75	2
3	Main Floor	58	10
4	Mid Balcony	38	0
5	Upper Balcony	19	3

```
SELECT AVG(price) FROM Tickets;
```

Maximum

id	seat	price	num_sold
1	Box Level	105	4
2	Dress Circle	75	2
3	Main Floor	58	10
4	Mid Balcony	38	0
5	Upper Balcony	19	3

```
SELECT MAX(num_sold) FROM Tickets;
```



Minimum

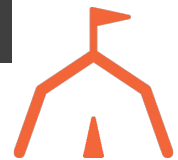
id	seat	price	num_sold
1	Box Level	105	4
2	Dress Circle	75	2
3	Main Floor	58	10
4	Mid Balcony	38	0
5	Upper Balcony	19	3

```
SELECT MIN(price) FROM Tickets;
```

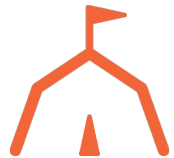
Combo

id	seat	price	num_sold
1	Box Level	105	4
2	Dress Circle	75	2
3	Main Floor	58	10
4	Mid Balcony	38	0
5	Upper Balcony	19	3

```
SELECT SUM(num_sold) AS `Total Sold`, SUM(price * num_sold) AS `Total Revenue`
FROM Tickets;
```



Relationships Between Tables



Tables Are Linked By ID

Student

id	name	class_id
1	G. Washington	1
2	M. Gandhi	1
3	N. Mandela	NULL
4	Q. Victoria	2

Class

id	title
1	.NET
2	Java
3	Front-End

Joins

id	name	class_id	id	title
1	G. Washington	1	1	.NET
2	M. Gandhi	1	1	.NET
4	Q. Victoria	2	2	Java

```
SELECT * FROM Student
JOIN Class ON Student.class_id = Class.id
```

Inner Join

The default join type. Where both tables match.

id	name	class_id	id	title
1	G. Washington	1	1	.NET
2	M. Gandhi	1	1	.NET
4	Q. Victoria	2	2	Java

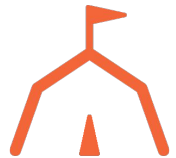
```
SELECT * FROM Student
INNER JOIN Class ON Student.class_id = Class.id
```

Left Join

Includes everything in the first table, even if it does not have a match.

id	name	class_id	id	title
1	G. Washington	1	1	.NET
2	M. Gandhi	1	1	.NET
3	N. Mandela	NULL	NULL	NULL
4	Q. Victoria	2	2	Java

```
SELECT * FROM Student
LEFT JOIN Class ON Student.class_id = Class.id
```

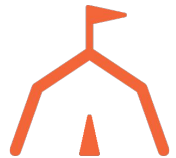


Right Join

Includes everything in the second table, even if it does not have a match.

id	name	class_id	id	title
1	G. Washington	1	1	.NET
2	M. Gandhi	1	1	.NET
NULL	NULL	NULL	3	Front-End
4	Q. Victoria	2	2	Java

```
SELECT * FROM Student
RIGHT JOIN Class ON Student.class_id = Class.id
```



Full Join

Includes everything in the both tables, even if it does not have a match.

id	name	class_id	id	title
1	G. Washington	1	1	.NET
2	M. Gandhi	1	1	.NET
3	N. Mandela	NULL	NULL	NULL
4	Q. Victoria	2	2	Java
NULL	NULL	NULL	3	Front-End

```
SELECT * FROM Student
FULL JOIN Class ON Student.class_id = Class.id
```

Cartesian Join

All the things.

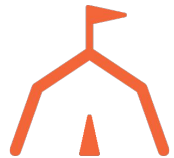
```
SELECT * FROM Student
CROSS JOIN Class
```

Join

Join can be combined with WHERE, ORDER BY and other SQL features.

```
SELECT Student.id, Student.name FROM Student
JOIN Class ON Student.class_id = Class.id
WHERE Class.title = '.NET' ORDER BY Student.name;
```

id	name
1	G. Washington
2	M. Gandhi

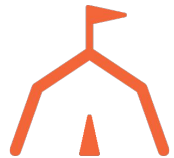


Stored Procedures

Stored Procedures

If you have a SQL query you're going to use repeatedly, you can store it as a stored procedure and call it.

Stored procedures can also have parameters, so you can pass in variable information.

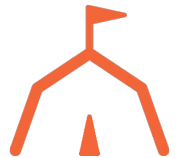


Example Stored Procedure

```
CREATE PROCEDURE CustomerSummary
AS
SELECT ContactName, ContactTitle, CompanyName, Country
FROM Customers
```

```
EXEC CustomerSummary
```

	ContactName	ContactTitle	CompanyName	Country
1	Maria Anders	Sales Representative	Alfreds Futterkiste	Germa...
2	Ana Trujillo	Owner	Ana Trujillo Emparedados y helad...	Mexico
3	Antonio Moreno	Owner	Antonio Moreno Taquería	Mexico
4	Thomas Hardy	Sales Representative	Around the Horn	UK
5	Christina Berglund	Order Administrator	Berglunds snabbköp	Sweden

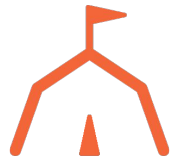


Example – Parameter

```
CREATE PROCEDURE CustSummaryByCountry @Country NVARCHAR(40)
AS
SELECT CustomerID, ContactName, ContactTitle, CompanyName,
Country FROM Customers WHERE Country = @Country
```

```
EXEC CustSummaryByCountry @Country = 'Germany'
```

	CustomerID	ContactName	ContactTitle	CompanyName	Country
1	ALFKI	Maria Anders	Sales Representative	Alfreds Futterkiste	Germa...
2	BLAUS	Hanna Moos	Sales Representative	Blauer See Delikatessen	Germa...
3	DRACD	Sven Ottlieb	Order Administrator	Drachenblut Delikatess...	Germa...
4	FRANK	Peter Franken	Marketing Manager	Frankenversand	Germa...
5	KOENE	Philip Cramer	Sales Associate	Königlich Essen	Germa...

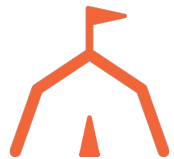


Example – Multiple Parameters

```
CREATE PROCEDURE BigLineItems @Units INT, @Qty SMALLINT
AS
SELECT * FROM [Order Details]
WHERE UnitPrice >= @Units AND Quantity >= @Qty
```

```
EXEC BigLineItems @Units=20, @Qty=10
```

	OrderID	ProductID	UnitPrice	Quantity	Discount
1	10249	51	42.40	40	0
2	10250	51	42.40	35	0.15
3	10252	20	64.80	40	0.05
4	10252	60	27.20	40	0
5	10255	59	44.00	30	0



Example - Join

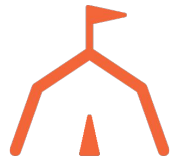
```
CREATE PROCEDURE ProductDetail @OrderId INT AS
SELECT o.OrderID, p.ProductName, o.Quantity, p.UnitPrice AS
  [CurrentPrice], o.UnitPrice AS [PriceAtOrder], o.Discount
FROM [Order Details] o
LEFT JOIN Products p ON o.ProductID = p.ProductID
WHERE o.OrderID = @OrderId
```

```
EXEC ProductDetail @OrderId = 10248
```

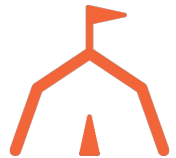
	OrderID	ProductName	Quantity	Current Price	Price at Order	Discount
1	10248	Queso Cabrales	12	21.00	14.00	0
2	10248	Singaporean Hokkien Fried M...	10	14.00	9.80	0
3	10248	Mozzarella di Giovanni	5	34.80	34.80	0

Altering Stored Procedures

```
ALTER PROCEDURE CustomerSummary
AS
SELECT CustomerId, ContactName, ContactTitle, CompanyName,
    Country
FROM Customers
```

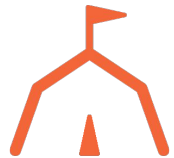


Recap



What You Should Know At This Point

- What are relational databases
- Know different relational DB products
- How SQL Server differs from other DB products
- What are the different SQL Server tools
- How to install and configure SQL Server



What You Should Know At This Point

- How to use SQL Server
- How to create tables
- How to create DBs (schema)
- Know SQL Server data types
- How to use SQL Server to query and modify data