SQL With

Kennt Ihr SQL Abfragen wie
[crayon-6766d2c76295d900249458/]
jetzt könnte man hingehen und die case when Anweisung in die Where Klausel bringen. Dies würde aber den Code unnötig aufblähen und Fehler sind einprogrammiert. Jetzt bietet SQL die With Anweisung an. Diese kann man sich wie eine ausgelagerte View vorstellen, auf die man zugreifen kann.
[crayon-6766d2c762963500567213/]
Wir sehen , wir können auf die augelagerte Anweisung mit einem neuen Select zugreifen

---

Richtig casten

Ich will es mal kurz halten :

Casten ist nicht konvertieren, es ist eher eine Typzuweisung. Da in C# alles vom Typ Object erbt, können wir dem Object alles zuweisen was wir wollen. Um dann dem Compiler zu sagen, dieses Object ist aber vom Typ String und besitzt dementsprechend seine Eigenschaften, müssen wir es Casten. Dazu gibt es 2 Möglichkeiten : Einmal die (class) Object und die Object as class Möglichkeit.

Der Unterschied der beiden ist, dass im ersten sobald nicht gecastet werden kann eine Exception aufgerufen wird. Beim as casten wird das Object null.

Je nach Anwendungsfall ist beides Sinnvoll

Bulkupsert oder einfach Update, wenn bereits vorhanden ansonsten Insert in MSSQL

Lange habe ich nach einer vernünftigen Lösung gesucht, um riesige Datenmengen schnell in die Datenbank zu schreiben.

Als Quelle steht uns eine List<Class> mit der Klasse zur Verfügung, die der Datenbanktabelle gleicht und jede Menge Inhalte enthält. Versucht man diese Liste nun über das Entity Framework in die Datenbank zu jagen, merkt man schnell, dass das Entity Framework an seine Grenzen stößt. Alternativ haben wir das von der SQLCopy Klasse die BulkInsert() Methode, die aber nur Inserten kann. Findet das BulkInsert einen Index/Primärschlüssel, der bereits vorhanden ist, wird die ganze Show abgebrochen. Wie toll wäre es nun, wenn es nicht abbrechen würde, sondern diese Zeile einfach updated. Und das ganze Superschnell. Ich habe Testweise 10.000 Einträge in 400ms und 1.000.000 in 1 min in die Datenbank bekommen.

Methode Upsert:
[crayon-6766d2c762ebe691035646/]
Methode BulkInsert:
[crayon-6766d2c762ec3366493287/]
und die beiden Extension Methods:
[crayon-6766d2c762ec4922317046/]

Angewendet wird das ganze ganz einfach so:

1. Parameter beinhaltet die Liste mit allen Inhalten
2. Parameter gibt an, welche Spalten geprüft werden sollen, ob es zu einem Update kommen soll oder Insert. In meinem Fall, wenn PersNr und CardId bereits in der Datenbank vorhanden sind, so mache ein Update, sonst Insert
3. Parameter gibt an, was inserted werden soll.
4. Parameter gibt an, was geupdated werden soll, wenn Parameter 2 zutrifft.

Bitte auf die tatsächlichen Primärschlüssel, Indizes und NOT NULL Schlüssel/Attribute achten. Ansonsten kommt man schnell zu einem Fehler
[crayon-6766d2c762ec6716708923/]

String zusammensetzen Vergleich mit C#6

Wer kennt das nicht, da muss man z.B. einen SQL String basteln, der Werte aus Variablen zusammen setzen muss.

Hierfür haben wir die 3 Variablen, die in die Personen Tabelle hinzugefügt werden müssen:
[crayon-6766d2c7630cd203751907/]

1. Die meisten Entwickler arbeiten dann einfach ohne Parameter:
   [crayon-6766d2c7630d0139566436/]
2. Der nächste sagt, ich arbeite mit dem StringFormater:
   [crayon-6766d2c7630d2002005970/]
3. Ein weiterer sagt, ich nutze den StringBuilder:

[crayon-6766d2c7630d3506434963/]

4. Neue Art einen String zusammen zu setzen mit C# Version 6:
   [crayon-6766d2c7630d5343225766/]
   Meiner Meinung nach ist die neue Möglichkeit deutlich übersichtlicher. Außerdem habe ich als Entwickler immer die Möglichkeit in den geschweiften Klammern Änderungen an der Variable vorzunehmen. Ich kann z.B. hingehen und Vorname.ToUpper() machen.