The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**

The Curse and Blessings of Dynamic SQL

An SQL text by Erland Sommarskog, SQL Server MVP. Latest revision: 2015-04-14. An earlier version of this article is also available in German. Translations provided by SQL Server MVP Frank Kalis.

Introduction

If you follow the various newsgroups on Microsoft SQL Server, you often see people asking why they can't do:

```
SELECT * FROM @tablename
SELECT @colname FROM tbl
SELECT * FROM tbl WHERE x IN (@list)
```

For all three examples you can expect someone to answer *Use dynamic SQL* and give a quick example on how to do it. Unfortunately, for all three examples above, dynamic SQL is a poor solution. On the other hand, there are situations where dynamic SQL is the best or only way to go.

In this article I will discuss the use of dynamic SQL in stored procedures and to a minor extent from client languages. To set the scene, I start with a very quick overview on application architecture for data access. I then proceed to describe the feature dynamic SQL as such, with a quick introduction followed by the gory syntax details. Next, I continue with a discussion on SQL injection, a security issue that it is essential to have good understanding of when you work with dynamic SQL. This is followed by a section where I discuss why we use stored procedures, and how that is affected by the use of dynamic SQL. I carry on with a section on good practices and tips for writing dynamic SQL. I conclude by reviewing a number of situations where you could use dynamic SQL and whether it is a good or bad idea to do it.

The article covers all versions of SQL Server from SQL 6.5 to SQL 2008, with emphasis on SQL 2000 and later versions.

Note: many of the code samples in this text works against the **pubs** and **Northwind**