Database Design: Size DOES Matter!

Thomas LaRock

Technical Evangelist and Senior DBA

Confio Software

Why Are You Here?

- You have data
- It is growing larger
- Performance suffers
- You want to make things better
  – Or you have been *told* to make things better

CONFIO SOFTWARE

A little about me…

Microsoft CERTIFIED Master | SQL Server 2008

MVP Microsoft Most Valuable Professional

sqlrockstar@
thomaslarock.com

thomaslarock.com

Thomas
LaRock

@sqlrockstar

ignite8 CONFIO

igniteVM CONFIO

Agenda

- What's the problem?
- Why datatypes matter
- Solution options

THE PROBLEM

The Problem

- Your database has a design
  - Just like a ship, or a car, has a design

The Problem

- Your database has a design
  - Just like a ship, or a car, has a design

The Problem

- Your database has a design
  - Just like a ship, or a car, has a design
- Data goes in, rarely falls off

The Problem

- Your database has a design
  - Just like a ship, or a car, has a design
- Data goes in, rarely falls off
- Databases are mixed-use

The Problem

- Your database has a design
  - Just like a ship, or a car, has a design
- Data goes in, rarely falls off
- Databases are mixed-use
- Data professionals are often asked to convert a pickup truck into

The Problem

- Your database has a design
  - Just like a ship, or a car, has a design
- Data goes in, rarely falls off
- Databases are mixed-use
- Data professionals are often asked to convert a pickup truck into a Ferrari

The Problem

- Your database has a design
  - Just like a ship, or a car, has a design
- Data goes in, rarely falls off
- Databases are mixed-use
- Data professionals are often asked to convert a pickup truck into a Ferrari
- Great database performance starts with great database design

How Database Designs Fail

- Many designs do not account for size

How Database Designs Fail

- Many designs do not account for size
- Rowcounts *might* be considered

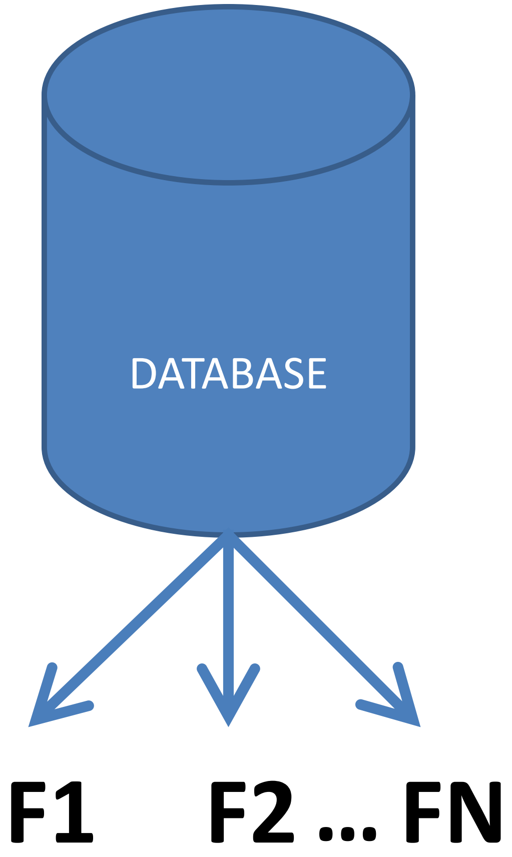
How Database Designs Fail

- Many designs do not account for size
- Rowcounts *might* be considered
- What about *inside* the rows?
  - Those are the columns
  - Their width is often overlooked

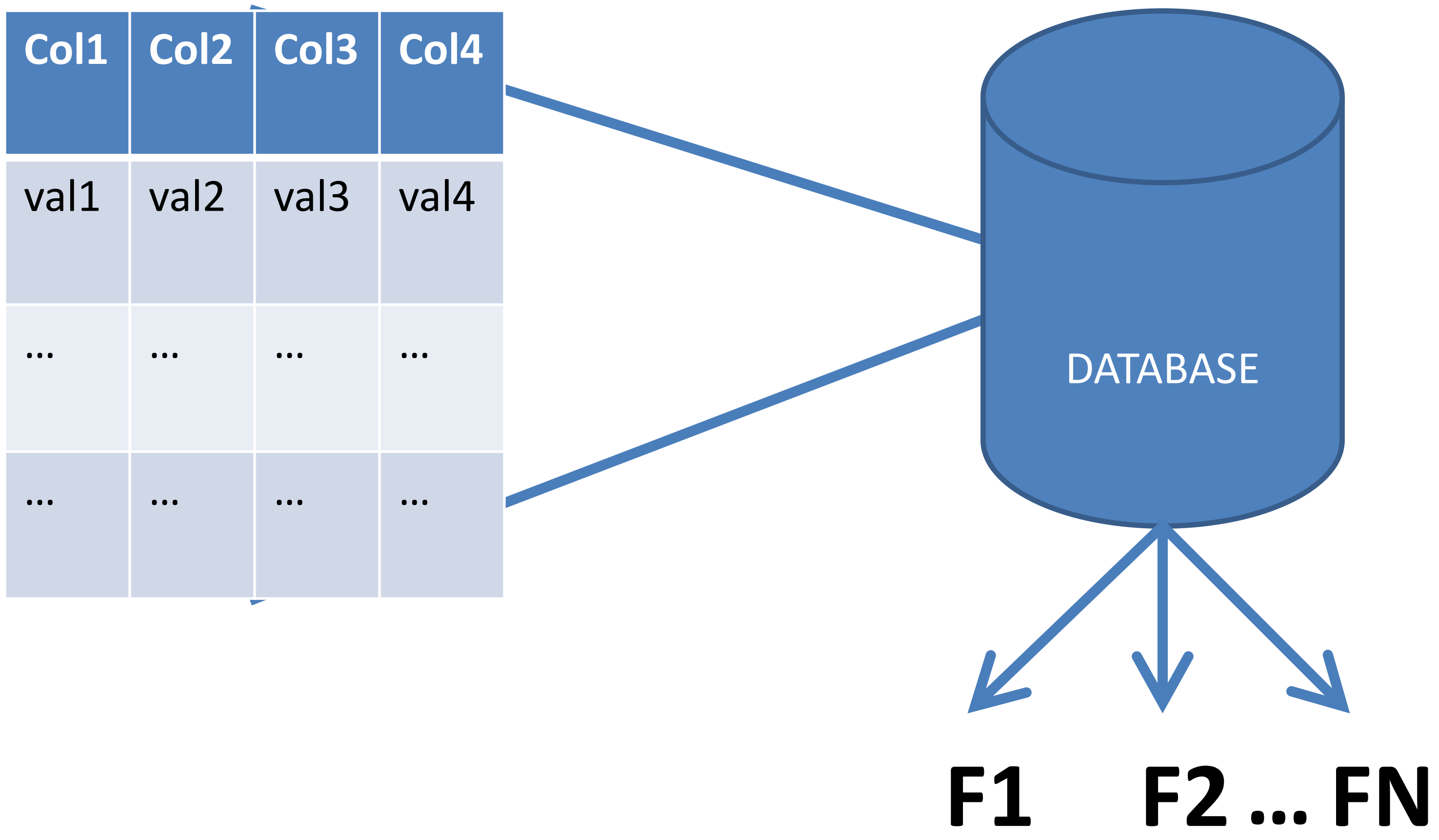
WHY DATATYPES MATTER

What Is A Database, Really?

What Is A Database, Really?

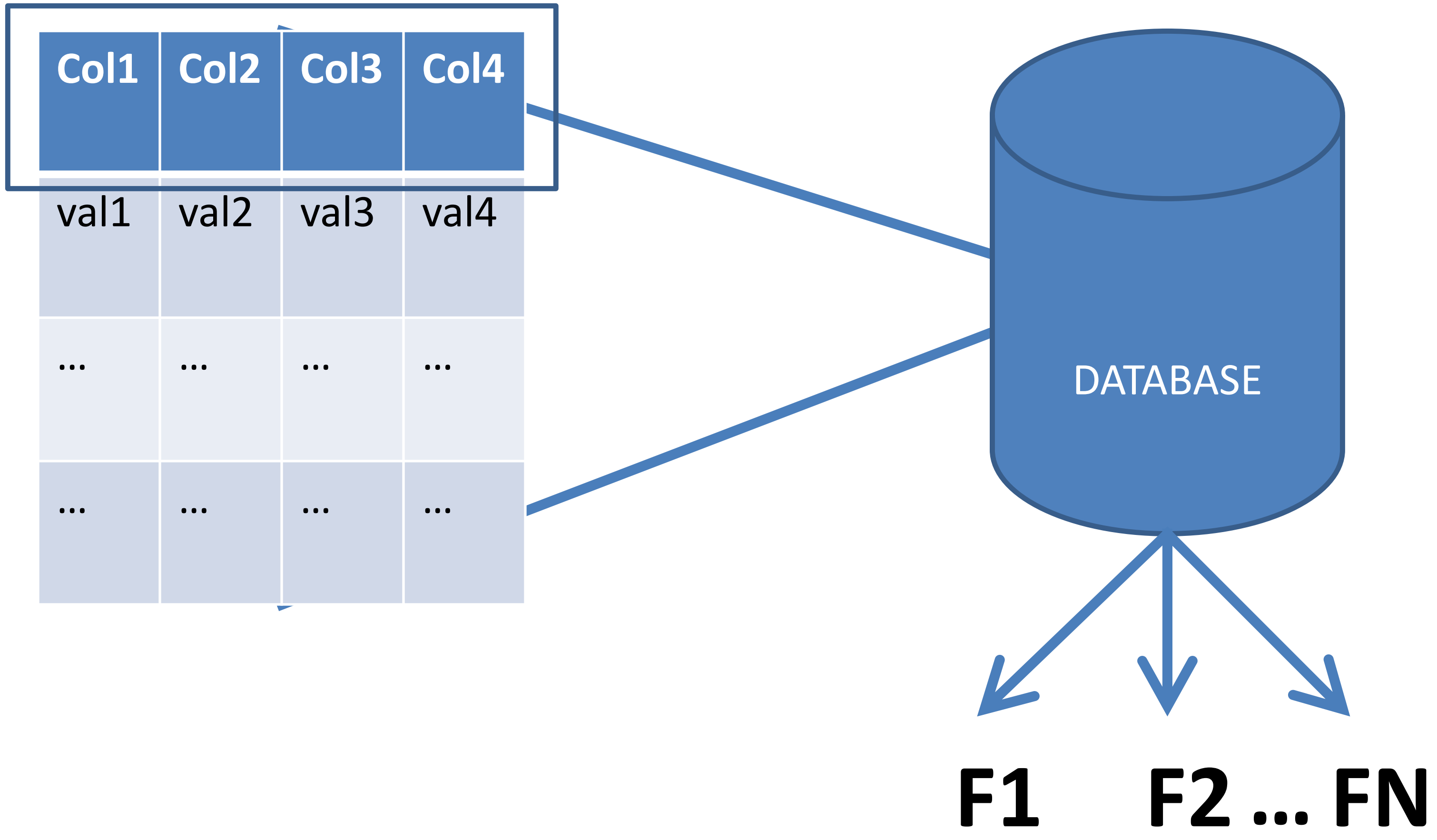


What Is A Database, Really?

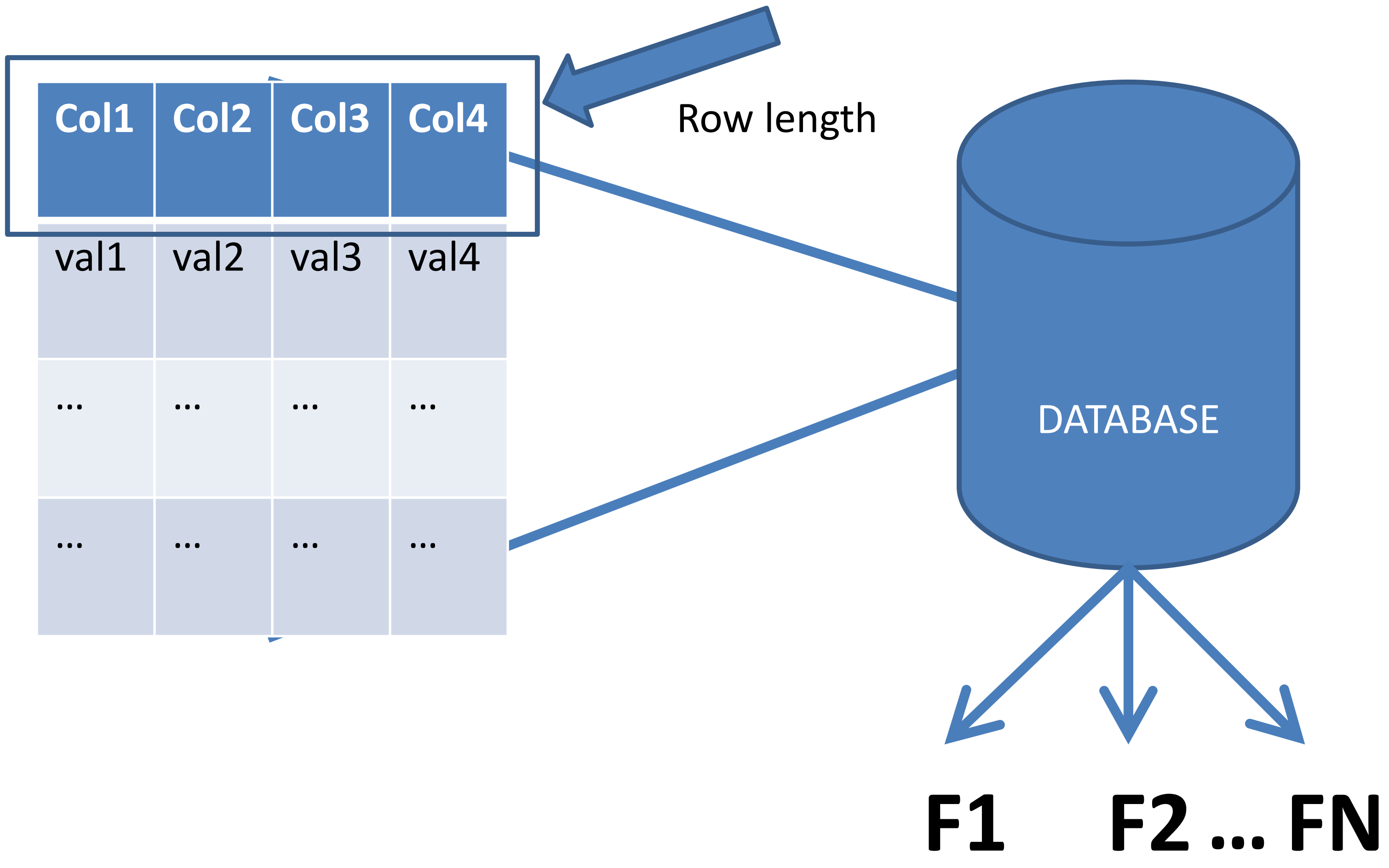


Col1	Col2	Col3	Col4
val1	val2	val3	val4
…	…	…	…
…	…	…	…

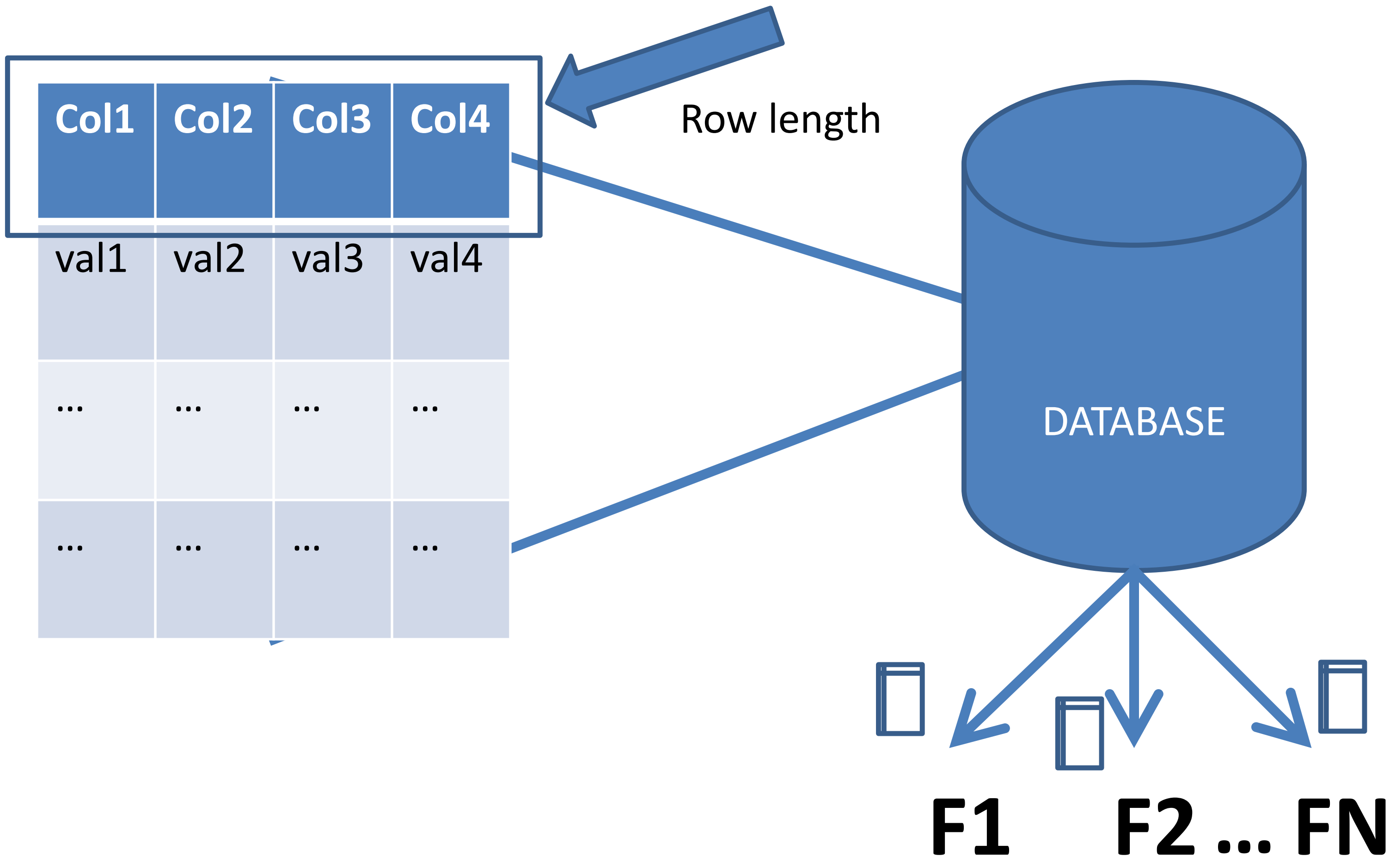
What Is A Database, Really?



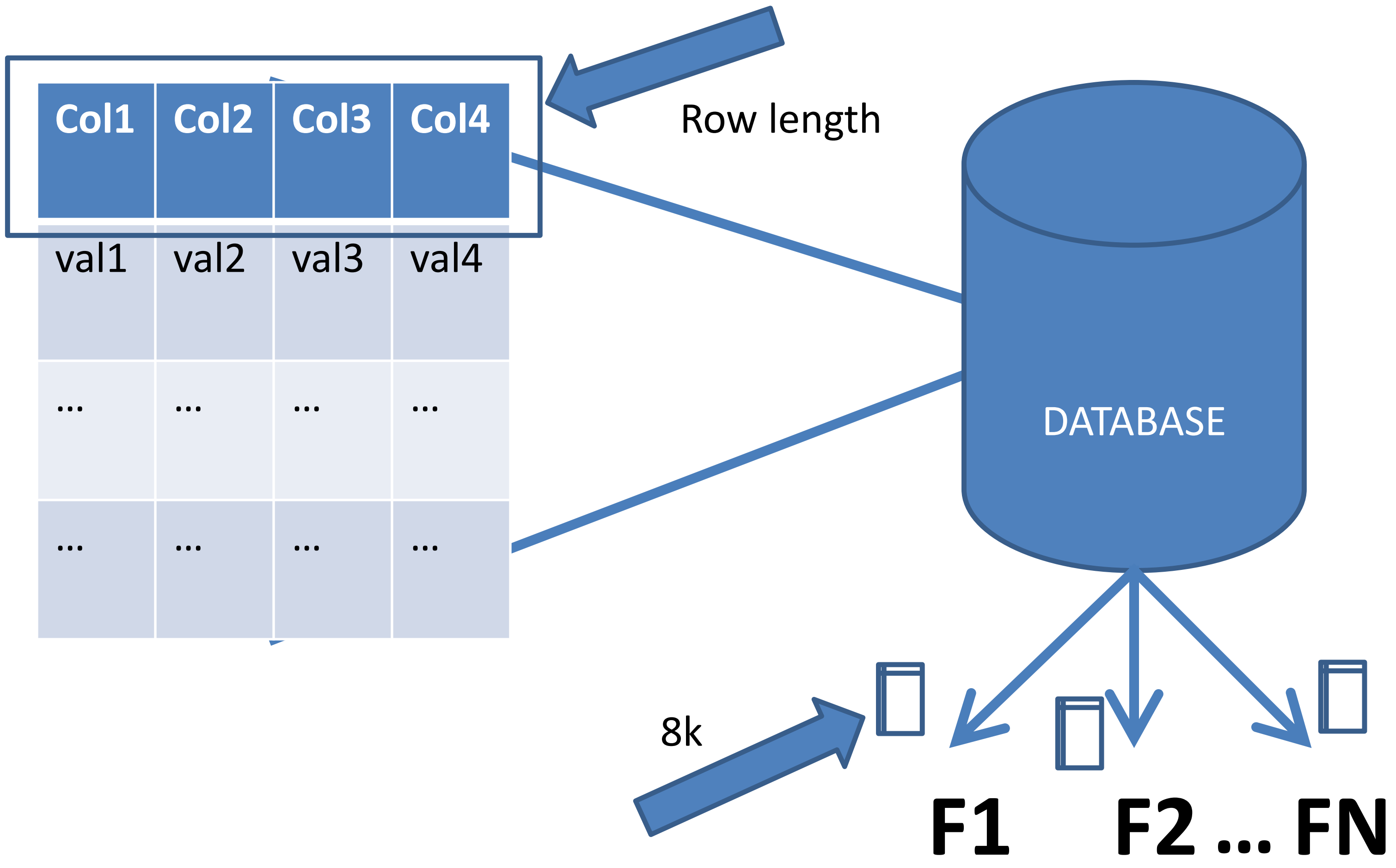
What Is A Database, Really?



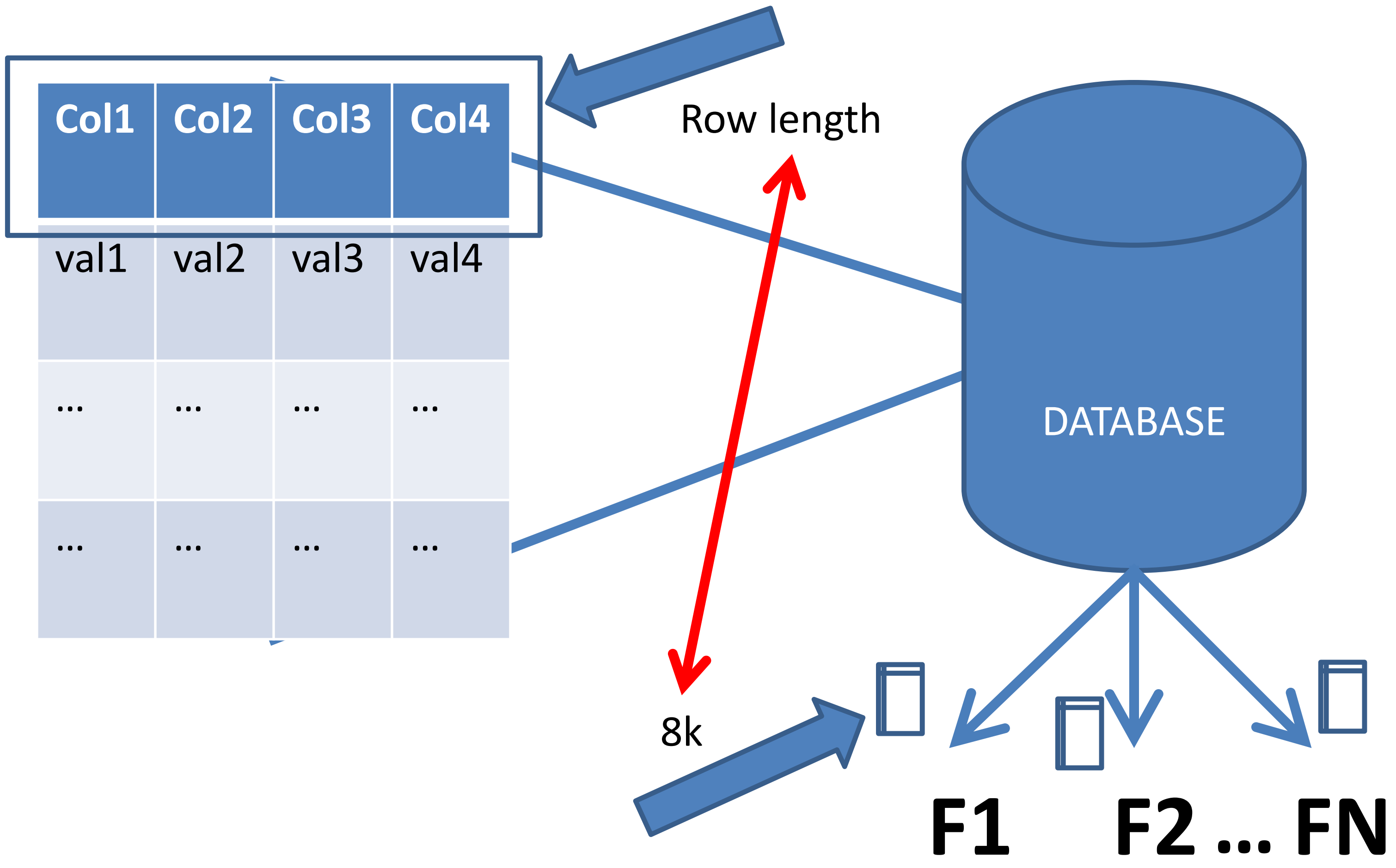
What Is A Database, Really?



What Is A Database, Really?



What Is A Database, Really?



What Is an Integer?

- Math major says: integers are numbers, positive and negative, including 0, without any fractions

CONFIO SOFTWARE

What Is an Integer?

- Math major says: integers are numbers, positive and negative, including 0, without any fractions
- Database designer: integers are the range of numbers from -2,147,483,648 to 2,147,483,647 and require 4 bytes of storage

What Is an Integer?

- Math major says: integers are numbers, positive and negative, including 0, without any fractions
- Database designer: integers are the range of numbers from -2,147,483,648 to 2,147,483,647 and require 4 bytes of storage
- Do you see a disconnect there?

Visual Studio Defaults

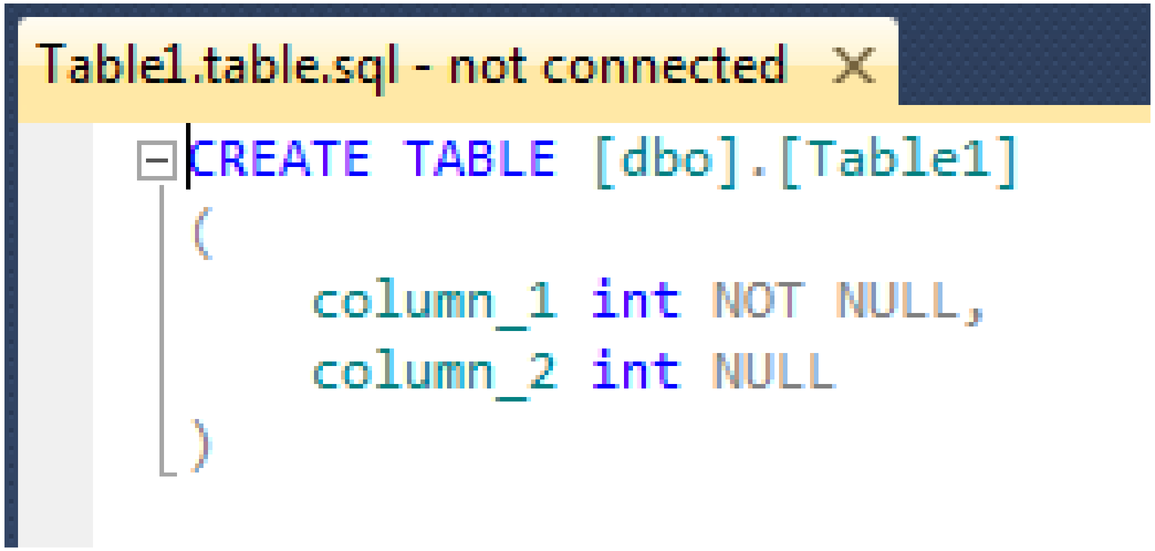


Table1.table.sql - not connected

```
CREATE TABLE [dbo].[Table1]
(
	column_1 int NOT NULL,
	column_2 int NULL
)
```

Visual Studio Defaults

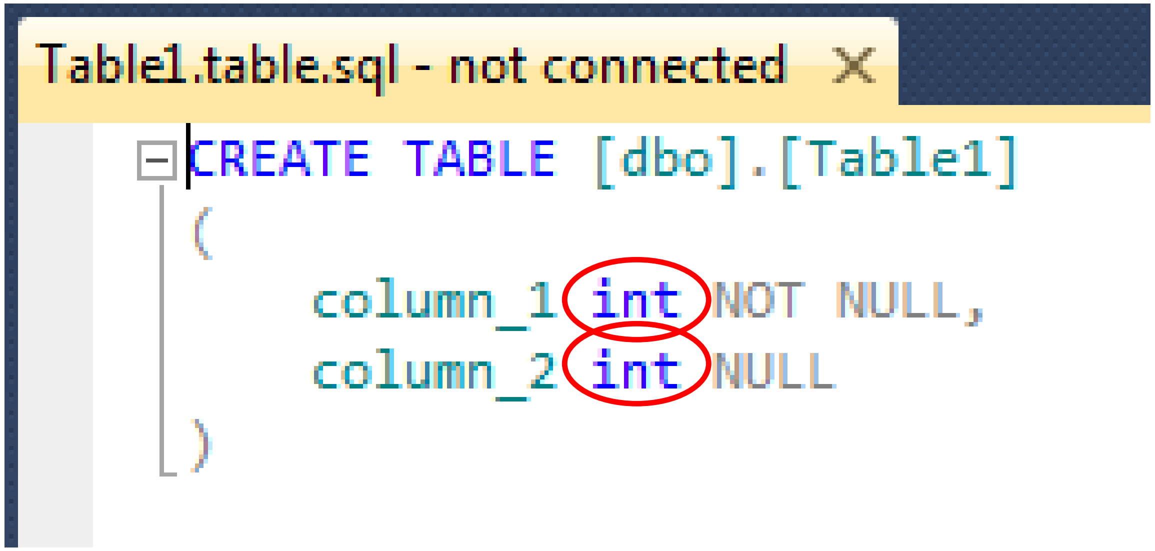


Table1.table.sql - not connected

```
CREATE TABLE [dbo].[Table1]
(
    column_1 int NOT NULL,
    column_2 int NULL
)
```

Visual Studio Defaults

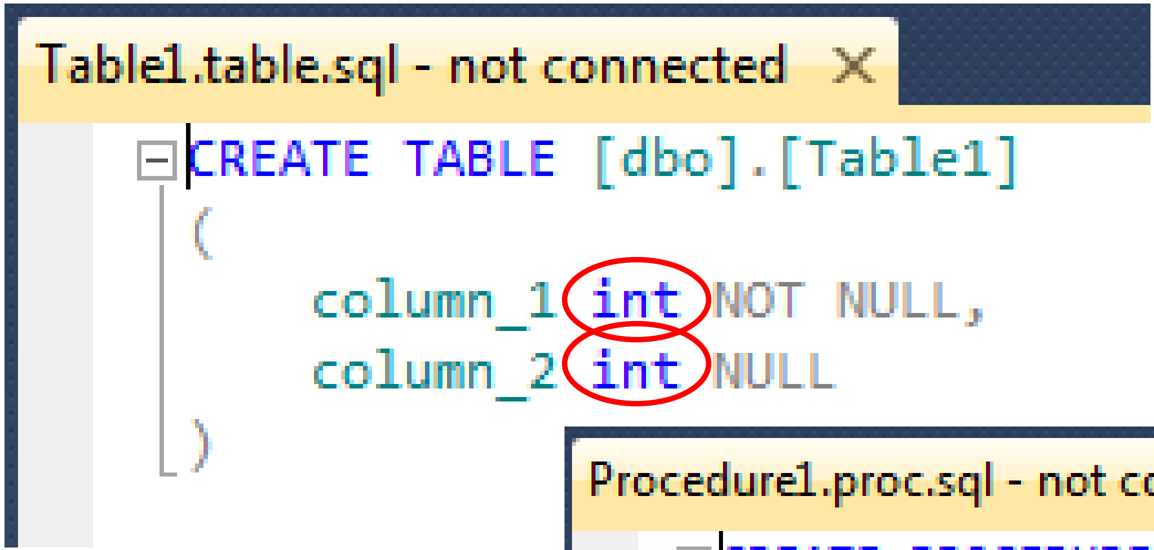
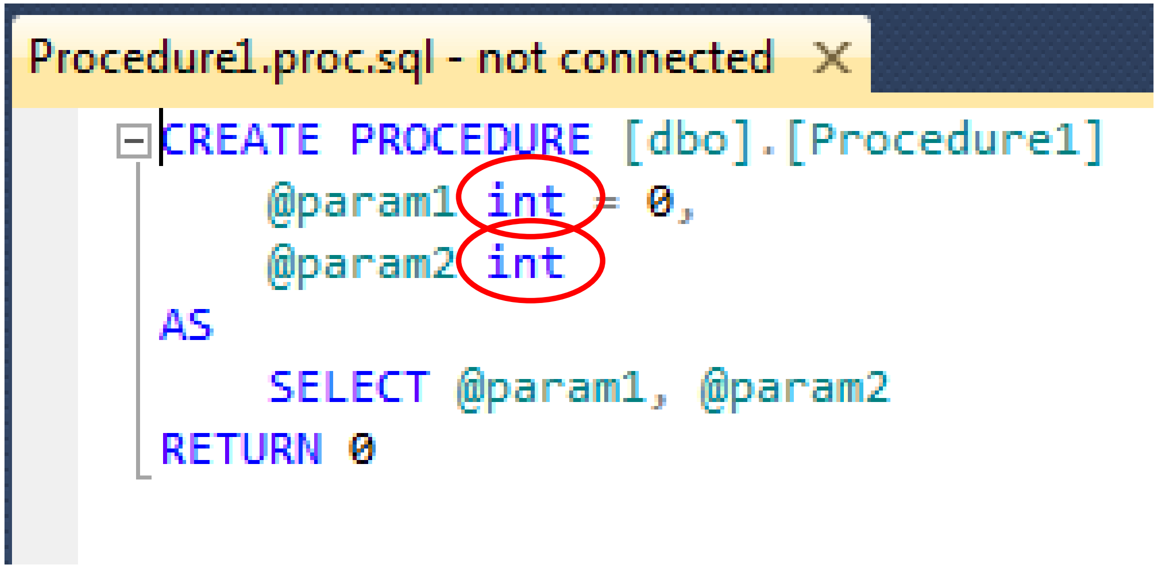


Table1.table.sql - not connected

```
CREATE TABLE [dbo].[Table1]
(
    column_1 int NOT NULL,
    column_2 int NULL
)
```



Procedure1.proc.sql - not connected

```
CREATE PROCEDURE [dbo].[Procedure1]
    @param1 int = 0,
    @param2 int
AS
    SELECT @param1, @param2
RETURN 0
```

Visual Studio Defaults

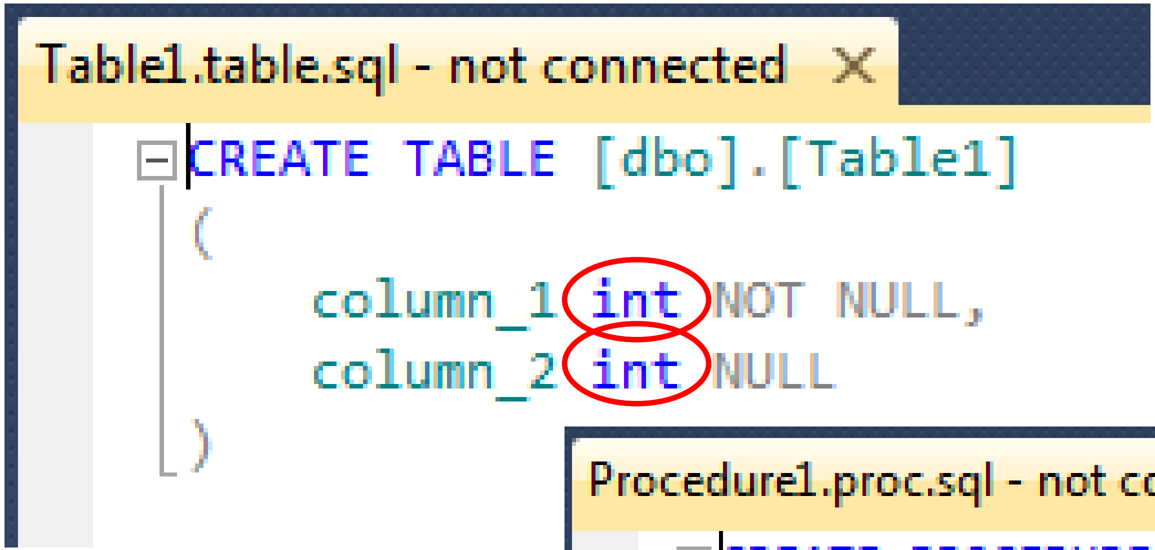
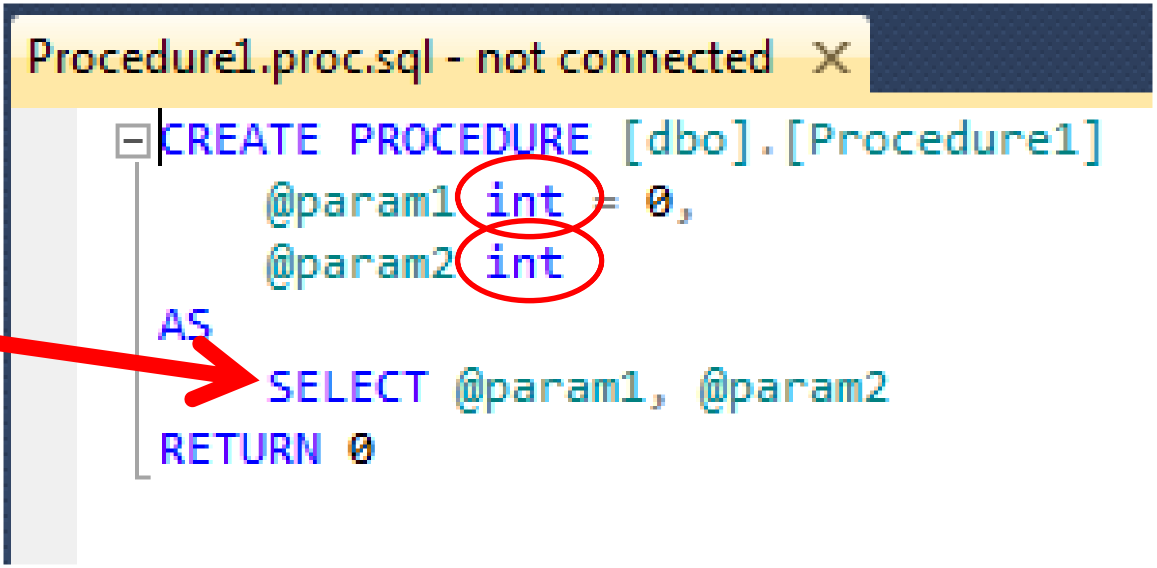


Table1.table.sql - not connected

```
CREATE TABLE [dbo].[Table1]
(
    column_1 int NOT NULL,
    column_2 int NULL
)
```

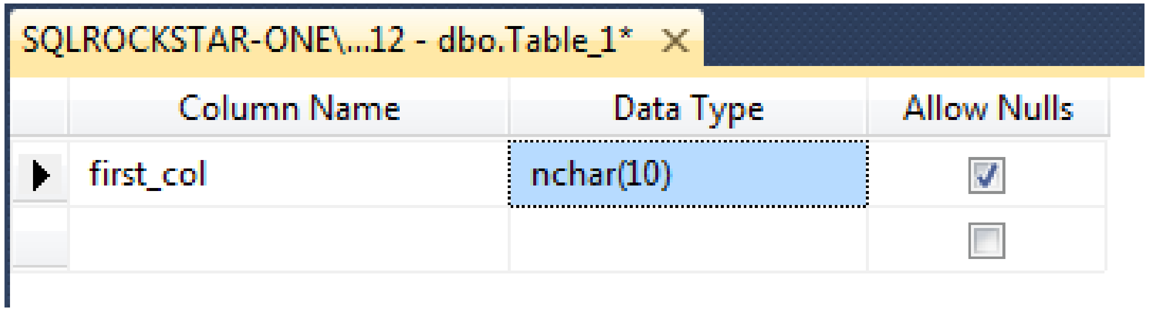


Procedure1.proc.sql - not connected

```
CREATE PROCEDURE [dbo].[Procedure1]
    @param1 int = 0,
    @param2 int
AS
    SELECT @param1, @param2
RETURN 0
```

Mismatch Likely!

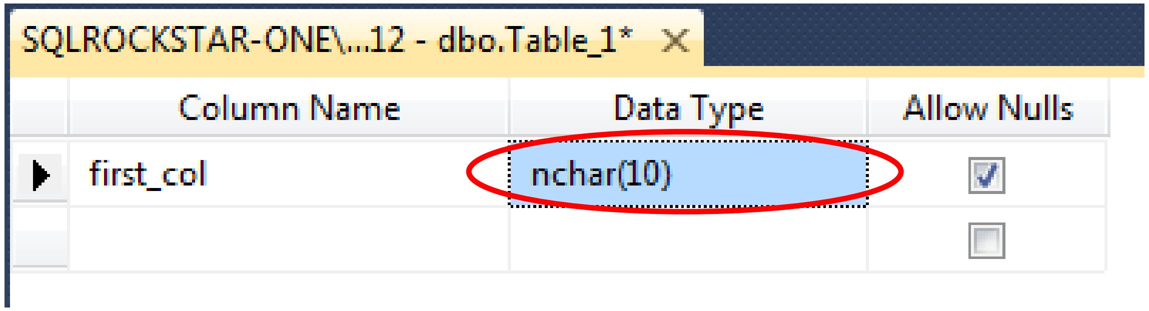
SSMS Defaults



SQLROCKSTAR-ONE\...12 - dbo.Table_1*

	Column Name	Data Type	Allow Nulls
▶	first_col	nchar(10)	☑
			☐

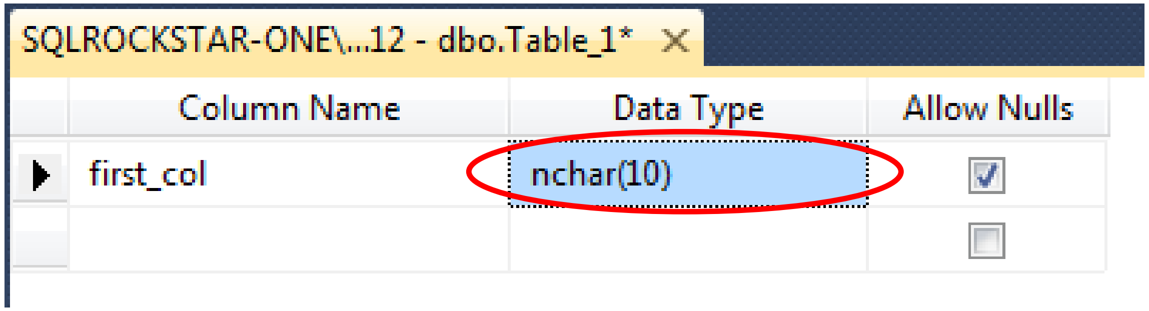
SSMS Defaults



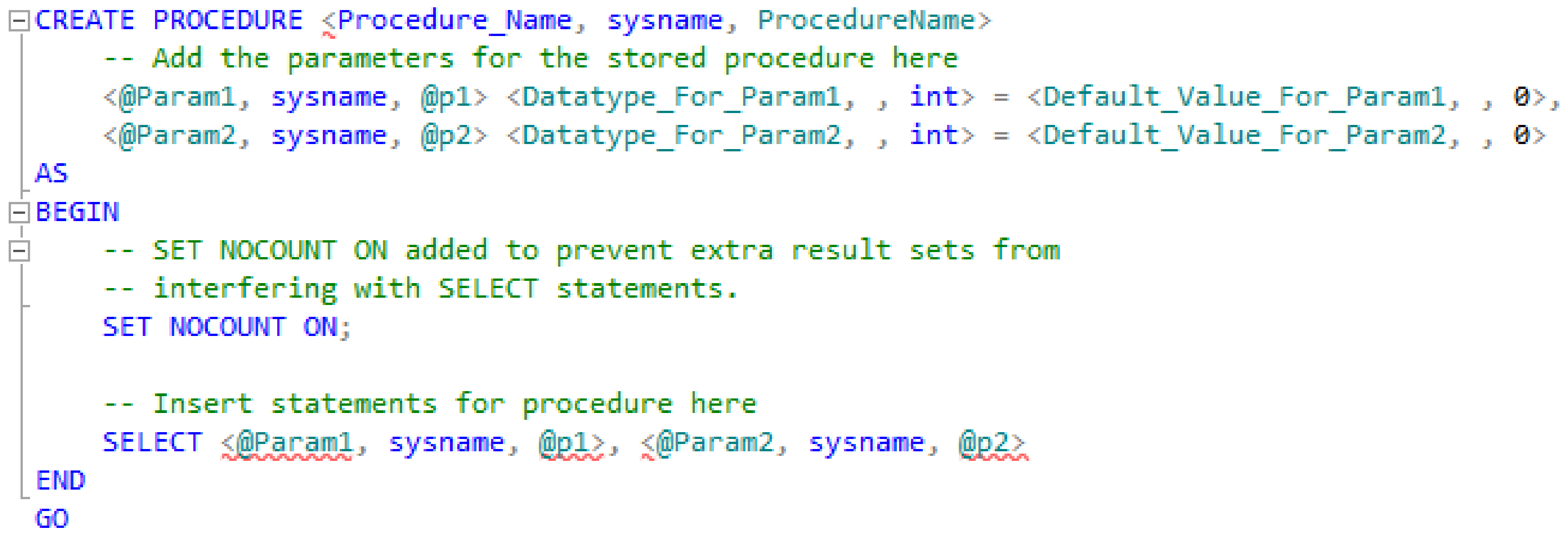
SQLROCKSTAR-ONE\...12 - dbo.Table_1*

	Column Name	Data Type	Allow Nulls
▶	first_col	nchar(10)	✓
			☐

SSMS Defaults

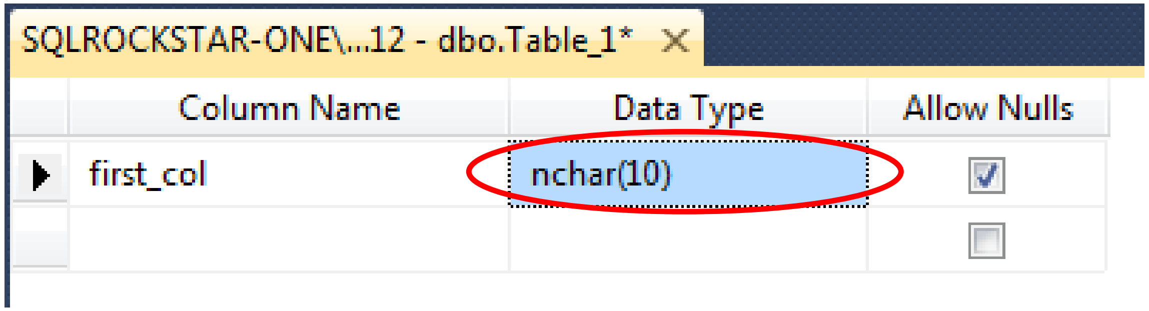

SQLROCKSTAR-ONE\...12 - dbo.Table_1*

Column Name	Data Type	Allow Nulls
first_col	nchar(10)	✓
		☐

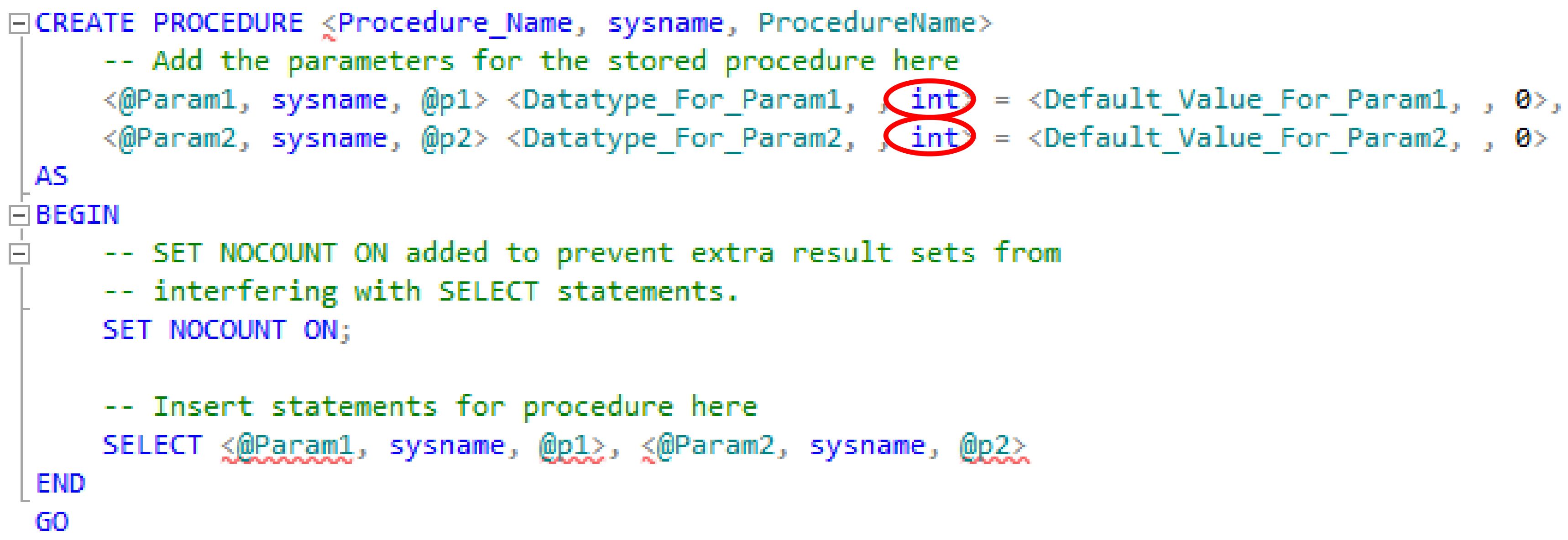

```
CREATE PROCEDURE <Procedure_Name, sysname, ProcedureName>
    -- Add the parameters for the stored procedure here
    <@Param1, sysname, @p1> <Datatype_For_Param1, , int> = <Default_Value_For_Param1, , 0>,
    <@Param2, sysname, @p2> <Datatype_For_Param2, , int> = <Default_Value_For_Param2, , 0>
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    SELECT <@Param1, sysname, @p1>, <@Param2, sysname, @p2>
END
GO
```

SSMS Defaults



Column Name	Data Type	Allow Nulls
first_col	nchar(10)	✓
		☐



```
CREATE PROCEDURE <Procedure_Name, sysname, ProcedureName>
    -- Add the parameters for the stored procedure here
    <@Param1, sysname, @p1> <Datatype_For_Param1, , int> = <Default_Value_For_Param1, , 0>,
    <@Param2, sysname, @p2> <Datatype_For_Param2, , int> = <Default_Value_For_Param2, , 0>
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    SELECT <@Param1, sysname, @p1>, <@Param2, sysname, @p2>
END
GO
```

Date Example

Datatype Name	Length (bytes)
Date	3
Smalldatetime	4
Time	5
Datetime2	6, 7, or 8
Datetime	8
Datetimeoffset	10

Where Will This Hurt?

- Disk space
- Memory space
- Performance

Datatype Pain #1: Disk Space

- Disk is cheap…sure it is…but there are hidden costs!
  - More storage = longer maintenance
  - More storage = longer database backups
  - More storage = longer tape backups

Datatype Pain #1: Disk Space

- Disk is cheap…sure it is…but there are hidden costs!
  - More storage = longer maintenance
  - More storage = longer database backups
  - More storage = longer tape backups
- Most are unaware how that tiny piece of data is stored multiple times
  - Wide clustering key is spread to EVERY NC index!

Datatype Pain #2: Memory Space

- Those extra bytes are read from disk into SQL Server buffer cache

Datatype Pain #2: Memory Space

- Those extra bytes are read from disk into SQL Server buffer cache
- Extra Logical I/O needed to return a query result

Datatype Pain #2: Memory Space

- Those extra bytes are read from disk into SQL Server buffer cache
- Extra Logical I/O needed to return a query result
- What about indexes?
  - Same thing, they drag around the extra I/O

DEMO

EXTRA LOGICAL I/O

Datatype Pain #3: Performance

- Datatype mismatch results in implicit conversions

Datatype Pain #3: Performance

- Datatype mismatch results in implicit conversions
- Optimizer knows this is bad, will display a warning for you

Datatype Pain #3: Performance

- Datatype mismatch results in implicit conversions
- Optimizer knows this is bad, will display a warning for you
- It is up to you to make a change, SQL Server won't change your code for you!

Datatype Pain #3: Performance

- Datatype mismatch results in implicit conversions
- Optimizer knows this is bad, will display a warning for you
- It is up to you to make a change, SQL Server won’t change your code for you!
- Issue with code generators (ADO.NET 3.5, EMF, nHibernate, LINQ)
  - Supposedly fixed?

DEMO

IMPLICIT CONVERSIONS

SOLUTION OPTIONS

First Step: Admit You Have a Problem!

- First sign: when OLAP queries interfere with OLTP processing

First Step: Admit You Have a Problem!

- First sign: when OLAP queries interfere with OLTP processing
- For SQL Server, that typically means high levels of locking waits and blocking

First Step: Admit You Have a Problem!

- First sign: when OLAP queries interfere with OLTP processing
- For SQL Server, that typically means high levels of locking waits and blocking
- Another sign: plan cache bloat

Solutions

- **Find duplicate/unused/misused indexes**
- Compression
- Filtered indexes
- Archiving/partitioning
- Verify datatypes are correct
- Update code
- Update architecture

Index Maintenance

- Want to reduce the extra I/O
  - Remove duplicate indexes
  - Remove unused/misused indexes

Index Maintenance

- Want to reduce the extra I/O
  - Remove duplicate indexes
  - Remove unused/misused indexes
- Test thoroughly!
  - Examine plan cache to find if indexes are used
  - Plan cache resets on service restart (or manual)

Finding Mis-used Indexes

```
SELECT o.name, indexname=i.name, i.index_id
, reads=user_seeks + user_scans + user_lookups
, writes =  user_updates
, rows = (SELECT SUM(p.rows) FROM sys.partitions p WHERE p.index_id = s.index_id
AND s.object_id = p.object_id)
, CASE
         WHEN s.user_updates < 1 THEN 100
         ELSE 1.00 * (s.user_seeks + s.user_scans + s.user_lookups) /
s.user_updates
  END AS reads_per_write
, 'DROP INDEX ' + QUOTENAME(i.name)
+ ' ON ' + QUOTENAME(c.name) + '.' + QUOTENAME(OBJECT_NAME(s.object_id)) as 'drop
statement'
FROM sys.dm_db_index_usage_stats s
INNER JOIN sys.indexes i ON i.index_id = s.index_id AND s.object_id = i.object_id
INNER JOIN sys.objects o on s.object_id = o.object_id
INNER JOIN sys.schemas c on o.schema_id = c.schema_id
WHERE OBJECTPROPERTY(s.object_id,'IsUserTable') = 1
AND s.database_id = DB_ID() AND i.type_desc = 'nonclustered'
AND i.is_primary_key = 0 AND i.is_unique_constraint = 0
AND (SELECT SUM(p.rows) FROM sys.partitions p WHERE p.index_id = s.index_id AND
s.object_id = p.object_id) > 10000
ORDER BY reads
```

Solutions

- Find duplicate/unused/misused indexes
- **Compression**
- Filtered indexes
- Archiving/partitioning
- Verify datatypes are correct
- Update code
- Update architecture

Compression

- Native compression has two steps
  - Row compression
  - Page compression
  - Each table/index requires compression to be enabled

DEMO

ROW/PAGE COMPRESSION

Compression

- Native compression has two steps
  - Row compression
  - Page compression
  - Each table/index requires compression to be enabled
- ColumnStore
  - Dictionary compression
  - Segments and batches

Solutions

- Find duplicate/unused/misused indexes
- Compression
- **Filtered indexes**
- Archiving/partitioning
- Verify datatypes are correct
- Update code
- Update architecture

Filtered Indexes

- Look like regular indexes
  - Contain a WHERE clause
- Smaller footprint
- Less logical I/O

Solutions

- Find duplicate/unused/misused indexes
- Compression
- Filtered indexes
- **Archiving/partitioning**
- Verify datatypes are correct
- Update code
- Update architecture

Partitioning

- Can be viable alternative to proper archiving

Partitioning

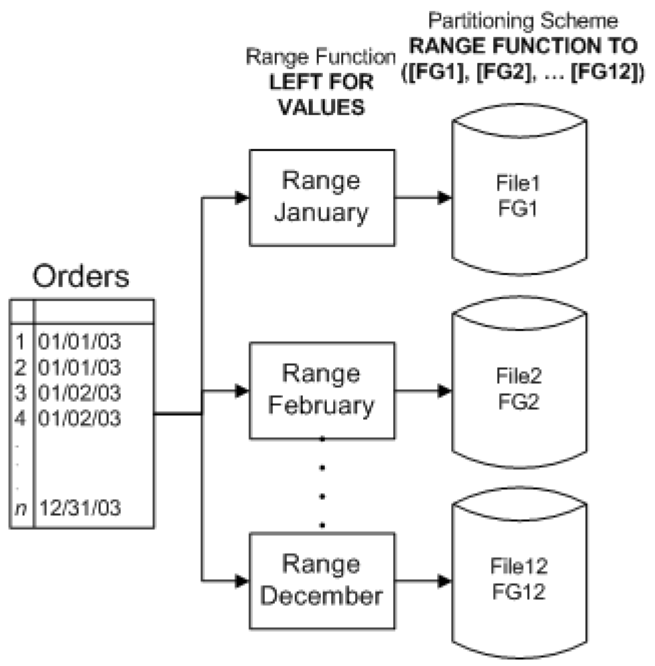
- Can be viable alternative to proper archiving
- Queries hit only the partitions necessary
  - Better than full table scans

Partitioning

- Can be viable alternative to proper archiving
- Queries hit only the partitions necessary
  - Better than full table scans
- Easier to maintain
  - Not necessarily
    easier to administer!

Partitioning

- Can be viable alternative to proper archiving
- Queries hit only the partitions necessary
  - Better than full table scans
- Easier to maintain
  - Not necessarily easier to administer!



Solutions

- Find duplicate/unused/misused indexes
- Compression
- Filtered indexes
- Archiving/partitioning
- **Verify datatypes are correct**
- Update code
- Update architecture

Verify Datatypes

- Compare datatype definition to actual data
  - i.e., BIGINT defined, but only SMALLINT used

Verify Datatypes

- Compare datatype definition to actual data
  - i.e., BIGINT defined, but only SMALLINT used
- You can estimate savings
  - Space as well as LIO

Verify Datatypes

- Compare datatype definition to actual data
  - i.e., BIGINT defined, but only SMALLINT used
- You can estimate savings
  - Space as well as LIO
- Can be intrusive to alter
  - Especially if PK/FK defined

Verify Datatypes

- Compare datatype definition to actual data
  - i.e., BIGINT defined, but only SMALLINT used
- You can estimate savings
  - Space as well as LIO
- Can be intrusive to alter
  - Especially if PK/FK defined
- Periodically check to make sure you are not running out!

DEMO

VERIFY DATATYPES

Solutions

- Find duplicate/unused/misused indexes
- Compression
- Filtered indexes
- Archiving/partitioning
- Verify datatypes are correct
- **Update code**
- **Update architecture**

Update Code/Architecture

- If you are seeing the signs, then you may consider updating code

Update Code/Architecture

- If you are seeing the signs, then you may consider updating code
- Some frameworks are not optimal
  - ADO.NET 3.5
  - LINQ to SQL
  - EMF

CONFIO SOFTWARE

Update Code/Architecture

- If you are seeing the signs, then you may consider updating code
- Some frameworks are not optimal
  - ADO.NET 3.5
  - LINQ to SQL
  - EMF
- Scale out architecture
  - Create real reporting solution
  - AlwaysOn

10 Steps For Right Sizing

1. Understand the storage requirements for every datatype you consider

10 Steps For Right Sizing

1. Understand the storage requirements for every datatype you consider
2. Review all design decisions based on the shape of the data – where it is now and where it is likely to be later.

10 Steps For Right Sizing

1. Understand the storage requirements for every datatype you consider
2. Review all design decisions based on the shape of the data – where it is now and where it is likely to be later.
3. Set datatypes based on business requirements, not tool defaults

10 Steps For Right Sizing

4. Measure and monitor fit of the data to its datatypes regularly

10 Steps For Right Sizing

4. Measure and monitor fit of the data to its datatypes regularly
5. Review each index creation request to see if it is duplicate

10 Steps For Right Sizing

4. Measure and monitor fit of the data to its datatypes regularly
5. Review each index creation request to see if it is duplicate
6. Measure for unused and duplicate indexes regularly

10 Steps For Right Sizing

7. Review new stored procedures to verify parameters are matching

10 Steps For Right Sizing

7. Review new stored procedures to verify parameters are matching
8. Find longest running and most often used queries

10 Steps For Right Sizing

7. Review new stored procedures to verify parameters are matching
8. Find longest running and most often used queries
9. Look for implicit conversions in your plan cache

10 Steps For Right Sizing

7. Review new stored procedures to verify parameters are matching
8. Find longest running and most often used queries
9. Look for implicit conversions in your plan cache
10. Remember that ***size matters***

Summary

- What’s the problem?
- Why datatypes matter
- Solution options

For More Information

- http://tinyurl.com/sql-datatypes
- http://tinyurl.com/imp-cols-in-plan-cache
- http://tinyurl.com/data-access-perf
- http://tinyurl.com/row-compression