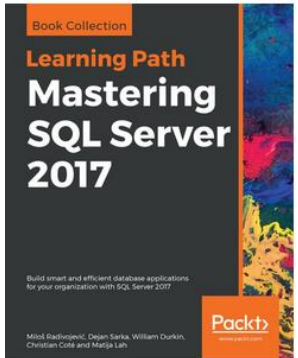
MILOŠ RADIVOJEVIĆ, MICROSOFT DATA PLATFORM MVP

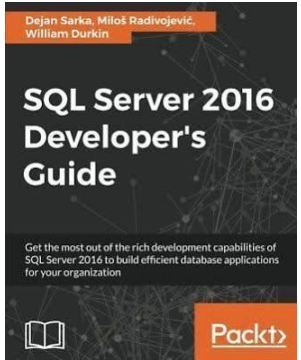
DATABASE DESIGN – COMMON MISTAKES

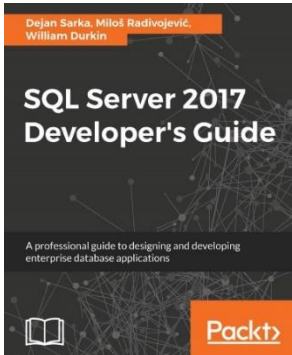
ABOUT ME

- Microsoft Data Platform MVP x8
- Head of Database Engineering at Entain, Vienna, Austria
- Co-Founder: SQL Pass Austria

- Contact:
  - E: milos@milossql.com
  - LinkedIn: milossql







WHY IS DATABASE DESIGN SO IMPORTANT?

- Database design is the foundation of the project
- Database usually outlives applications and services
- Changes are very expensive and practically impossible in the late stages of the project
- Good design saves money, time and reputation

DATABASE DESIGN - COMMON MISTAKES

- Database design is underestimated
  - Unattractive topic
  - No tools
- Not taking enough time
  - Agile
- Know-How
  - Who creates database tables?
- Database is a storage
  - “Business Logic belongs to the Business Layer”

DO NOT CREATE A TABLE IN TWO MINUTES!

DO NOT CREATE A TABLE IN TWO MINUTES!

- How does a programmer usually make a table?
  - finds CREATE TABLE some tables,
  - hits Copy
  - hits Paste
  - slightly changes the column names
  - selects the primary key
- Do not create table with copy / paste
- Nemojte ljudi, ko Boga vas molim!

DO NOT CREATE A TABLE IN TWO MINUTES!

- The implementation of business logic starts with the CREATE TABLE
- Take enough time to analyze requirements
- Changes and redesign are expensive and might be very painful

DO NOT CREATE A TABLE IN TWO MINUTES

UNLESS YOU ARE

NAMING CONVENTIONS

NAMING CONVENTIONS

Respect naming conventions

Colleagues will read your code more easily and you will reduce the possibility of different interpretations

- Renaming existing objects is very expensive
- Do not make exceptions - the exceptions end up becoming the rule

NAMING CONVENTIONS - AN EXAMPLE

Someone named a column **cDate**...

- Is it a start date, cut-off datem end dae or a transaction time stamp?
- What a question... of course, it is a start date!

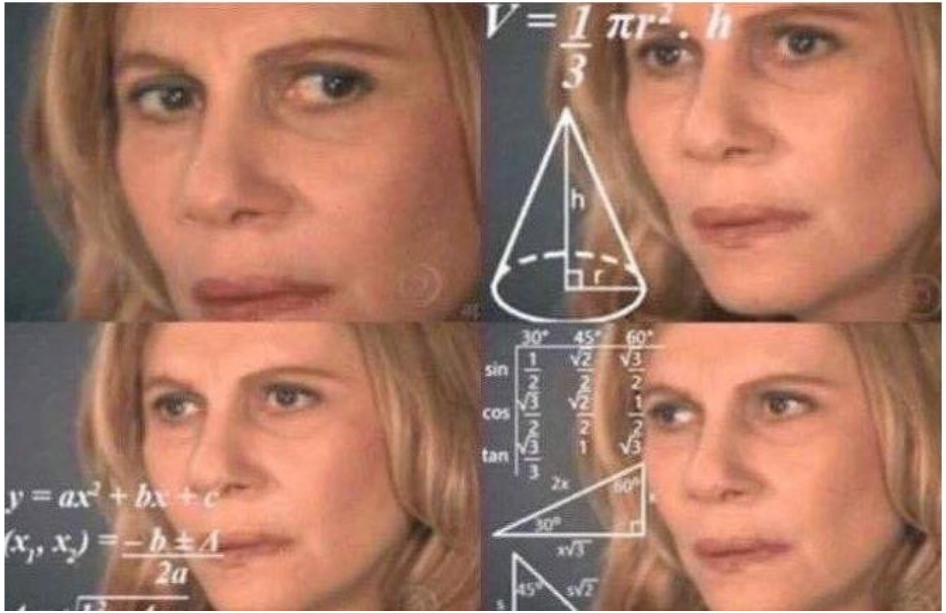
- The time is UTC, GMT or server time?
- It is UTC, brt, in the databases X, Y and Z we use UTC, in the other dbs server time

- WTF! Why then you did not use the name **cStartDateUTC** for it?

NAMING CONVENTIONS - WHAT TO AVOID?

- Abbreviations
- Spaces in object names
- Keywords
- Negations
  - isValid >>> isNotValid

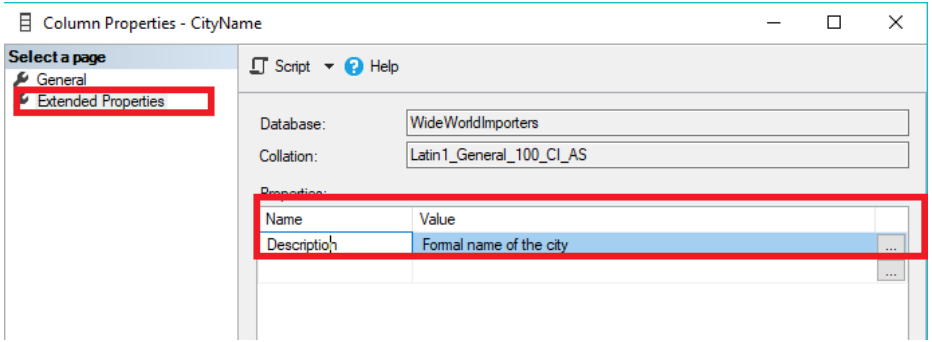
```
WHERE isNotValid NOT IN ('isNotValid','expired')
```

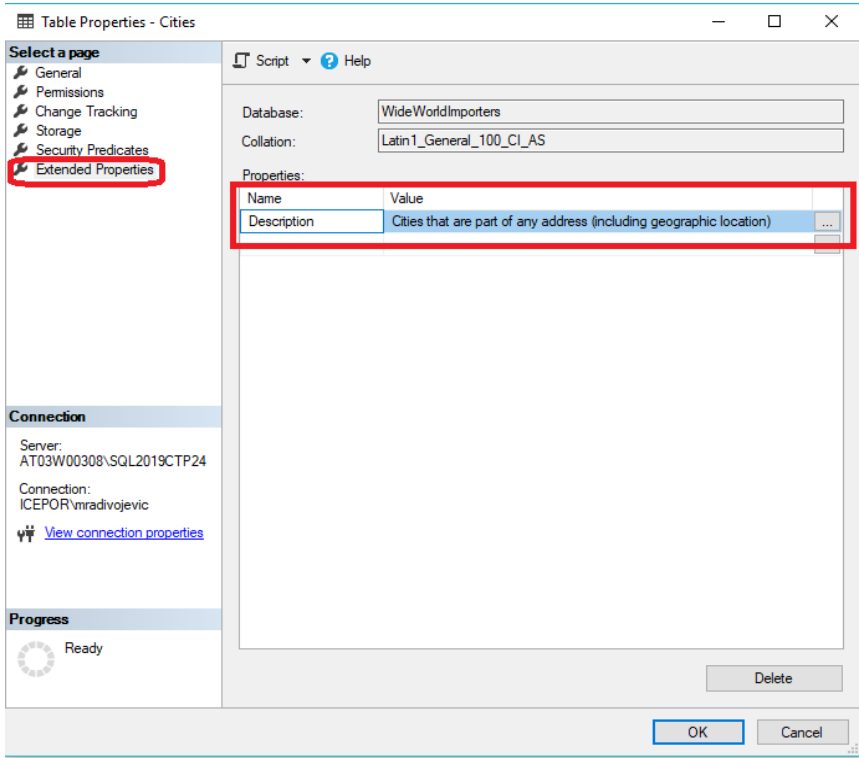


NAMING CONVENTION

- Document tables and columns
  - ExtendedProperties

- Better follow any convention than no convention





CHOOSE THE APPROPRIATE DATA TYPE

CHOOSE THE APPROPRIATE DATA TYPE

Which data type use developers most often for the Status field?

- INT

INT is a very flexible data type; it allows you to have many different statuses: 4 294 967 295

$$-2^{31} \leq INT \leq 2^{31} - 1$$

$$-2{,}147{,}483{,}648 \leq INT \leq 2{,}147{,}483{,}647$$

- Don‘t use INT for the Status, take TINYINT
- INT needs 4 bytes for storage and size does matter

DATA TYPE SIZE DOES MATTER...

```sql
CREATE TABLE dbo.StudentExams1(
    exam_number int NOT NULL,
    student_id int NOT NULL,
    exam_id int NOT NULL,
    exam_note int NULL,
    exam_date datetime  NULL,
 CONSTRAINT PK_StudentExams1 PRIMARY KEY CLUSTERED (exam_number ASC)
)
GO

CREATE TABLE dbo.StudentExams2(
    exam_number int NOT NULL,
    student_id int NOT NULL,
    exam_id int NOT NULL,
    exam_note tinyint NULL,
    exam_date date NULL,
 CONSTRAINT PK_StudentExams2 PRIMARY KEY CLUSTERED (exam_number ASC)
)
GO
```

DATA TYPE SIZE DOES MATTER...

```sql
DECLARE @date_from DATETIME = '20000101';
DECLARE @date_to DATETIME = '20171231';
DECLARE @number_of_rows INT = 10000000;
INSERT INTO dbo.StudentExams1
SELECT n AS exam_number,
    1 + ABS(CHECKSUM(NEWID())) % 50000 AS student_id,
    1 + ABS(CHECKSUM(NEWID())) % 40 AS exam_id,
    5 + ABS(CHECKSUM(NEWID())) % 6 AS exam_note,
(SELECT(@date_from +(ABS(CAST(CAST( NewID() AS BINARY(8) )AS INT))
% CAST((@date_to - @date_from)AS INT)))) exam_date
FROM dbo.GetNums(@number_of_rows)
GO
```

DATA TYPE SIZE DOES MATTER...

```
--Check table size
SELECT  OBJECT_NAME(s.object_id) AS table_name, CAST((s.used_page_count/128.0) AS int) ta
FROM sys.dm_db_partition_stats AS s
INNER JOIN sys.indexes AS i
    ON s.[object_id] = i.[object_id] AND s.index_id = i.index_id
INNER JOIN sys.tables AS t
    ON s.[object_id] = t.[object_id]
WHERE s.object_id IN ( OBJECT_ID('dbo.StudentExams1'), OBJECT_ID('dbo.StudentExams2')) AN
/*
table_name                                  table_size_MB
------------------------------------------- -------------
StudentExams1                               320
StudentExams2                               240
*/
```

+25% STORAGE

DATA TYPE SIZE DOES MATTER...

```sql
--Check the data page number
SELECT OBJECT_NAME(p.object_id) AS table_name, data_pages
FROM sys.allocation_units AS a
INNER JOIN  sys.partitions  AS p
  ON a.container_id = p.partition_id
WHERE p.object_id IN ( OBJECT_ID('dbo.StudentExams1'), OBJECT_ID('dbo.StudentExams2'))
GO
/*
table_name                      data_pages
------------------------------- --------------------
StudentExams1                   40842
StudentExams2                   30960
*/
```

+30% DATA PAGES

DATA TYPE UNDERESTIMATION

- You can also underestimate data type

- INT vs. BIGINT
  - If you think you can have more than 200M rows, use BIGINT!
- SMALLINT vs. INT
  - If you think you can have more than 5K rows, use INT!

CHOOSE THE APPROPRIATE DATA TYPE

- Choose the smallest type of data that is sufficient to cover the required domain
- When you choose a robust data type
  - You need more disk space, backup and restore are slower
  - SQL Server must read more pages to produce the same result
- Choose the DATE data type, if you do not need precision in minutes or seconds
- Data type changes are very expensive, difficult to sell, and no one is interested in paying for them

CHOOSE THE APPROPRIATE DATA TYPE

- Changes that are necessary when changing the data type:
  - ALTER the data type of the column within the table
  - Update the parameters of stored procedures that use this column
  - ```
    IF OBJECT_ID('dbo.insertStudentExams','U') IS NULL
        DROP PROCEDURE dbo.insertStudentExams
    GO
    CREATE PROCEDURE dbo.insertStudentExams
    ```
  - ```
    //se.ExamNote = dr.GetInt32(2);
    se.ExamNote = (int) dr.GetByte(2);
    ```

    ```
    yint
    xam_id,@exam_note)
    ```

    ely invisible

USE DATABASE CONSTRAINTS

USING DATABASE CONSTRAINTS

- CHECK Constraints
  - Limit values allowed in a column
  - Everyone will interpret the values in the column in the same way
  - Increase data quality
- UNIQUE constraints
  - If something is unique, use the UNIQUE constant and share the uniqueness information with the RDBMS system!
- That's good for both integrity and performance!

USING DATABASE CONSTRAINTS

CHECK CONSTRAINTS

Business Logic belongs to Business Layer

USING DATABASE CONSTRAINTS

- Name your constraints!
- You cannot change the data type or rename a column if it has a constraint, you need to remove the constraint first

```
--change data type
ALTER TABLE k ALTER COLUMN c1 DATETIME;
GO
 /*
Msg 5074, Level 16, State 1, Line 121
The object 'DF__K__c1__239E4DCF' is dependent on column 'c1'.
Msg 4922, Level 16, State 9, Line 121
ALTER TABLE ALTER COLUMN c1 failed because one or more objects access this column.
*/
--you need to remove constraint first, but you need the name
ALTER TABLE k DROP CONSTRAINT DF__K__c1__239E4DCF;
ALTER TABLE k ALTER COLUMN c1 DATETIME;
ALTER TABLE k ADD CONSTRAINT DF_K_c1 DEFAULT GETDATE() FOR c1;
GO
```

USING DATABASE CONSTRAINTS

```sql
--this won't work on another server, you have to use dynamic SQL
DECLARE @sql NVARCHAR(300)
DECLARE @cname NVARCHAR(200) = (SELECT name FROM sys.default_constraints
WHERE parent_object_id = OBJECT_ID('dbo.K'))
SET @sql = CONCAT(N'ALTER TABLE k DROP CONSTRAINT ',@cname);
EXEC sp_executesql @sql;
ALTER TABLE k ALTER COLUMN c1 DATETIME;
ALTER TABLE k ADD CONSTRAINT DF_K_c1 DEFAULT GETDATE() FOR c1;
GO
```

REDUCE THE NUMBER OF NULLABLE COLUMNS

ISSUES WITH NULLS

- Three Valued Logic
- Different Interpretations
- Performance

p	NOT *p*
T	F
U	U
F	T

p	*q*	*p* AND *q*
T	T	T
T	U	U
T	F	F
U	T	U
U	U	U
U	F	F
F	T	F
F	U	F
F	F	F

p	*q*	*p* OR *q*
T	T	T
T	U	T
T	F	T
U	T	T
U	U	U
U	F	U
F	T	T
F	U	U
F	F	F

THREE VALUED LOGIC

```
CREATE TABLE dbo.Color (id INT NOT NULL, name VARCHAR(30) NOT NULL)
GO
INSERT INTO dbo.Color(id, name) VALUES(1, 'Black'),(2, 'White'), (3, 'Purple');
GO
SELECT * FROM dbo.Color;
/*Results
id          name
----------- ------------------------------
1           Black
2           White
3           Purple
*/
```

```
SELECT DISTINCT Color
FROM Production.Product;
```

109 %

Results | Messages

	Color
1	NULL
2	Black
3	Blue
4	Grey
5	Multi
6	Red
7	Silver
8	Silver/Black
9	White
10	Yellow

THREE VALUED LOGIC

```sql
SELECT * FROM dbo.Color
WHERE name IN (SELECT Color FROM AdventureWorks2019.Production.Product);
```

9 %

Results

```
id          name
----------- ------------------------------
1           Black
2           White
```

```sql
SELECT * FROM dbo.Color
WHERE name NOT IN (SELECT Color FROM AdventureWorks2019.Production.Product);
```

109 %

Results

```
id          name
----------- ------------------------------

(0 rows affected)
```

THREE VALUED LOGIC

```sql
SELECT * FROM dbo.Color
WHERE name NOT IN (SELECT Color FROM AdventureWorks2019.Production.Product)
UNION ALL
SELECT * FROM dbo.Color
WHERE name IN (SELECT Color FROM AdventureWorks2019.Production.Product);
```

09 %

Results

```
id          name
----------- ------------------------------
1           Black
2           White

(2 rows affected)
```

PURPLE IS MISSING!

DIFFERENT INTERPRETATIONS

- You should base your conclusions on NULLs

Id	FirstName	LastName	BirthDate	City	ZipCode	AddressLine
2	Vlado	Kreslin	1953-11-29	NULL	NULL	NULL

- But people make conclusions…

id	isConvicted
1	1
2	0
3	NULL
4	NULL
5	0

ADVANTAGES WITH NULLS

- NOT NULL

```sql
INSERT INTO dbo.Customers2 (Id, FirstName, LastName)
VALUES(5, N'Vlado', N'Kreslin');
```

109 %

Messages

```
Msg 515, Level 16, State 2, Line 37
Cannot insert the value NULL into column 'BirthDate', table 'DBDesign.dbo.Customers2'; column does not allow nulls. INSERT fails.
The statement has been terminated.
```

- NULL
  - INSERT/UPDATE will work!
  - But…

Id	FirstName	LastName	BirthDate	City
1	Zoran	Predin	1958-06-16	Maribor
2	Vlado	Kreslin	1953-11-29	NULL
3	asssd	effrefe	NULL	NULL
4	aaa	bbb	NULL	NULL

REDUCE THE NUMBER OF NULLABLE COLUMNS

- Reduce the number of columns that allow NULL
  - NULL brings the overhead
  - The application code must handle data sets that contain NULL
- Different actors usually have different interpretations of the rows in which the column has NULL
- Show that you care about the quality and consistency of the data
- Sometimes it's impossible to avoid NULL, but at least you should try
- If you can't answer quickly why a column should supports NULL, then change to NOT NULL

CONCLUSION

- The implementation of business logic starts with CREATE TABLE
- "Business logic belongs to the application; the database is just a storage" - one of the most dangerous myths in programming
- The database is more durable than the application
- Take enough time to design tables properly

CONCLUSION

- Carefully and consistently choose table and other objects names respecting the naming convention

- For each column in a table:
  - choose the smallest data type that covers the attribute domain
  - if the column can only accept certain values, use the CHECK constraint
  - reduce the number of columns that allow NULL

RECOMMENDED BOOKS