MILOŠ
-SQL-

NAJČEŠĆE GREŠKE

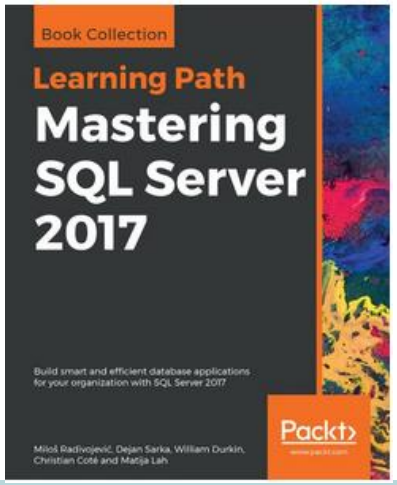
U RADU SA SQL SERVEROM

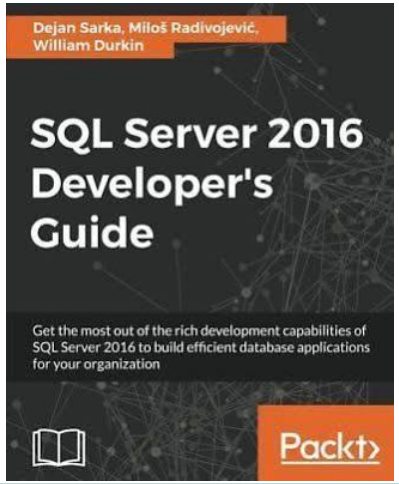
ABOUT ME

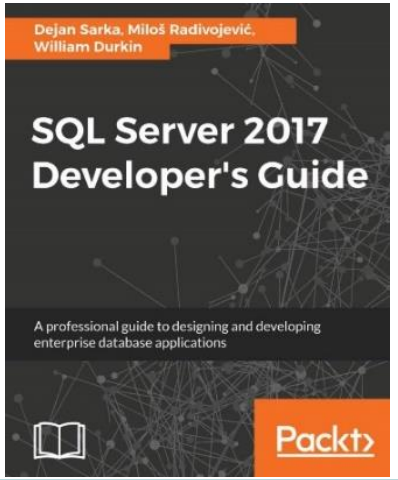
Microsoft® Most Valuable Professional

Entain

- 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

DATA SATURDAY #18

DATA SATURDAY #18 - CROATIA 2022

04 June 2022

DATA SATURDAY #18 – PRECON SEMINARS

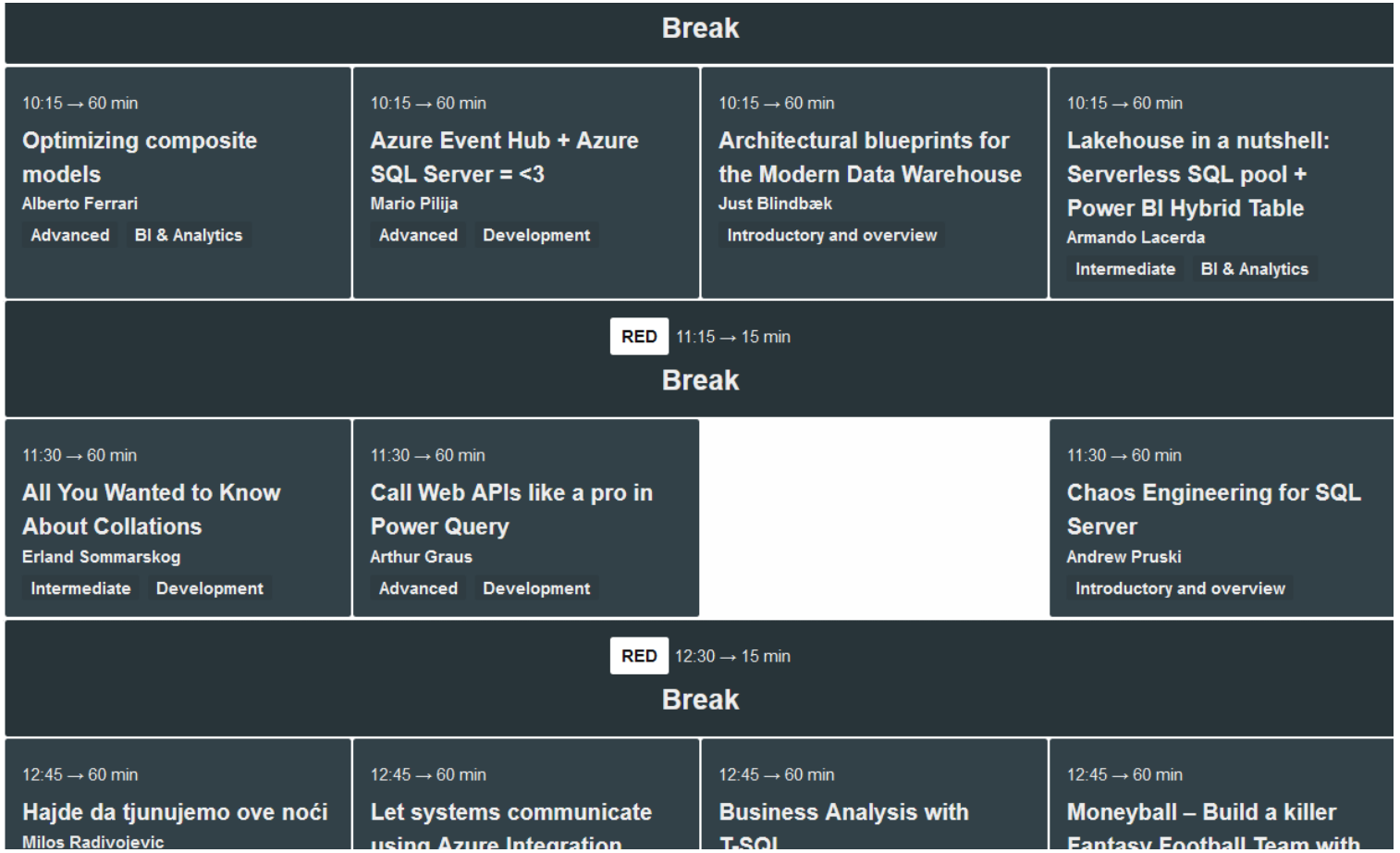
- **Dean Vitner and Torsten Strauß** - Query Engine and Execution Plans in Microsoft SQL Server
- **Miloš Radivojević** - Database Development Best Practices
- **Alberto Ferrari** - Improve Power BI performance by optimizing DAX

- **Datum:** Petak, 03.06.2022.
- **Prijave:** https://events.3nf.hr/

- **Website konferencije:** https://datasaturdays.com/2022-06-04-datasaturday0018/

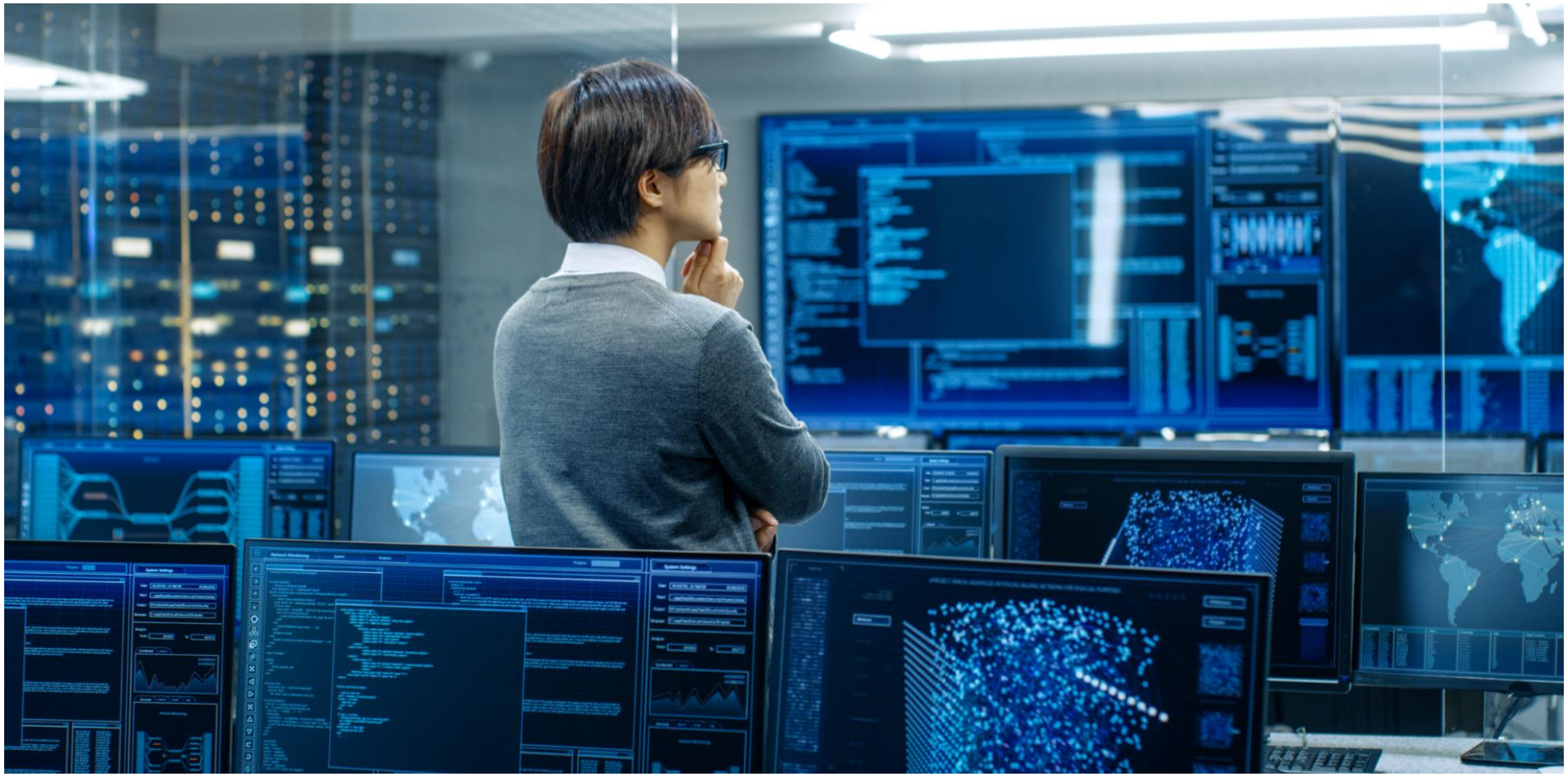
DATA SATURDAY #18 – CONFERENCE DAY

- **Raspored predavanja:** https://datasaturdays.com/2022-06-04-datasaturday0018/#schedule



MILOŠ
-SQL-

DATABASE ADMINISTRATION



TIPIČNE GREŠKE U ADMINISTRACIJI (1/2)

- RPO & RTO
- Lack of troubleshooting methodology
- Potcenjivanje kompleksnosti SQL Server apgrejda
- Preskakanje patching ciklusa
- Komunikacija (DBA/DEV or rest)
- Podešavanja na nivou servera
  - Remote Dedicated Admin Connection
  - Insant File Initialization
  - Power settings (windows server)
  - Cost threshold parallelism

TIPIČNE GREŠKE U ADMINISTRACIJI (2/2)

- Podešavanja na nivou baze
  - Veličina mdf i ndf fajlova (više terabajta)
  - Page Verify, Auto Close
- Previše prava za korisnike i servise
- Statistike
- Zapostavljeni indeksi

RPO & RTO

- Koliki su Vam dizvinete RPO i RTO?
- Kad ste poslednji put testirali ceo DR proces? Koliko je trajalo?

RPO & RTO

- Ko pokreće DR proces? Šta ako je ta osoba nedostupna?
- Je l može još neko to da uradi?
- Je l dokumentovano?
- Je l ima neophodna prava pristupa? Je l probo?

LACK OF TROUBLESHOOTING METHODOLOGY

- Spisak akcija i dijagnostičkih upita
- Treninzi dok ne postane rutina
- U slučaju velikih incidenata procesi, dokumentovani koraci i rutina spašavaju situaciju!

POTCENJIVANJE KOMPLEKSNOSTI SQL SERVER APGREJDA

- Zašto se apgrejduje?
- Kako izgleda server nakon apgrejda?

- Koraci:
  - Komunikacija
  - Priprema
  - Intenzivno testiranje
  - Knowledge sharing

PRESKAKANJE PATCHING CIKLUSA

- Ako radimo patching onda mora failover, a to je downtime, mi radije preskačemo patching...

- Patching znači bug fixing mnogih grešaka u operativnom sistemu i SQL Serveru
- Patching donosi security zakrpe
- Patching može da donese nove bagove kao posledicu bug fixinga
- Patching je praktično jedini test za failover!
- Restart servera = nova tempdb, počišćeni objekti, ispeglani eventualni problemi sa veličinom fajlova etc.

KOMUNIKACIJA DBA - DEVELOPERI

- Nema komunikacije
  - uglavnom prilikom incidenata
  - no knowledge sharing

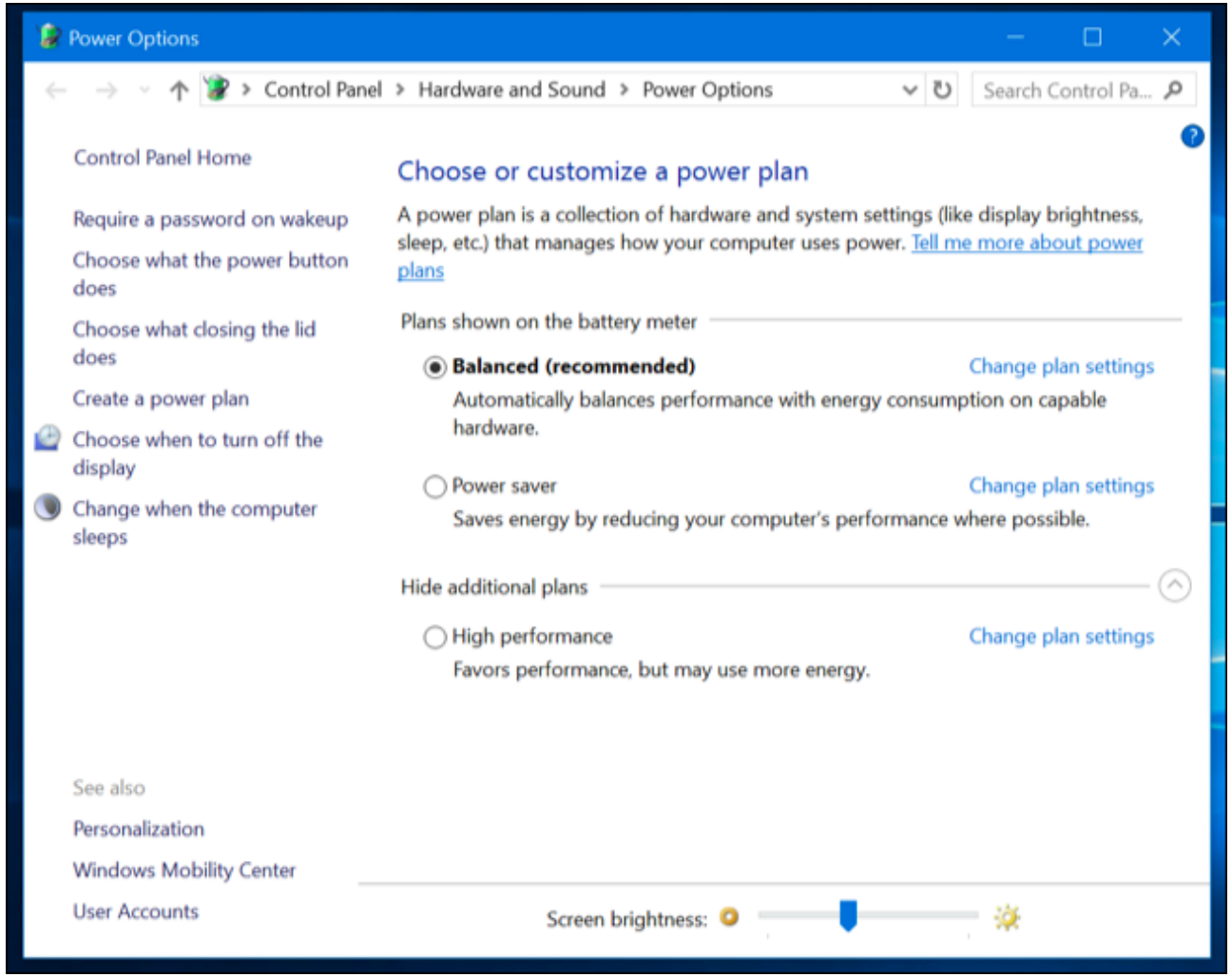
- Preporučuje se:
  - Kontinuirana komunikacija
  - Followups nakon većih incideneta
  - Knowledge sharing
  - Brifinzi u vezi sa planovima i predstojećim većim akcijama

PODEŠAVANJA NA NIVOU SERVERA

- Instant File Initialization
- Power settings (windows server)
- Cost Threshold for Parallelism
- Remote DAC connection

```
EXEC sp_configure 'remote admin connections', 1;
GO
RECONFIGURE
GO
```

- Više dejta fajlova (8) za tempdb



PODEŠAVANJA NA NIVOU BAZE

- Budite sigurni da su ove opcije izabrane na nivou baze:
  - AutoClose = FALSE
  - Page Verify = CHECKSUM

- Veličina .mdf odnosno .ndf fajlova ne bi trebalo da prelazi 2 TB

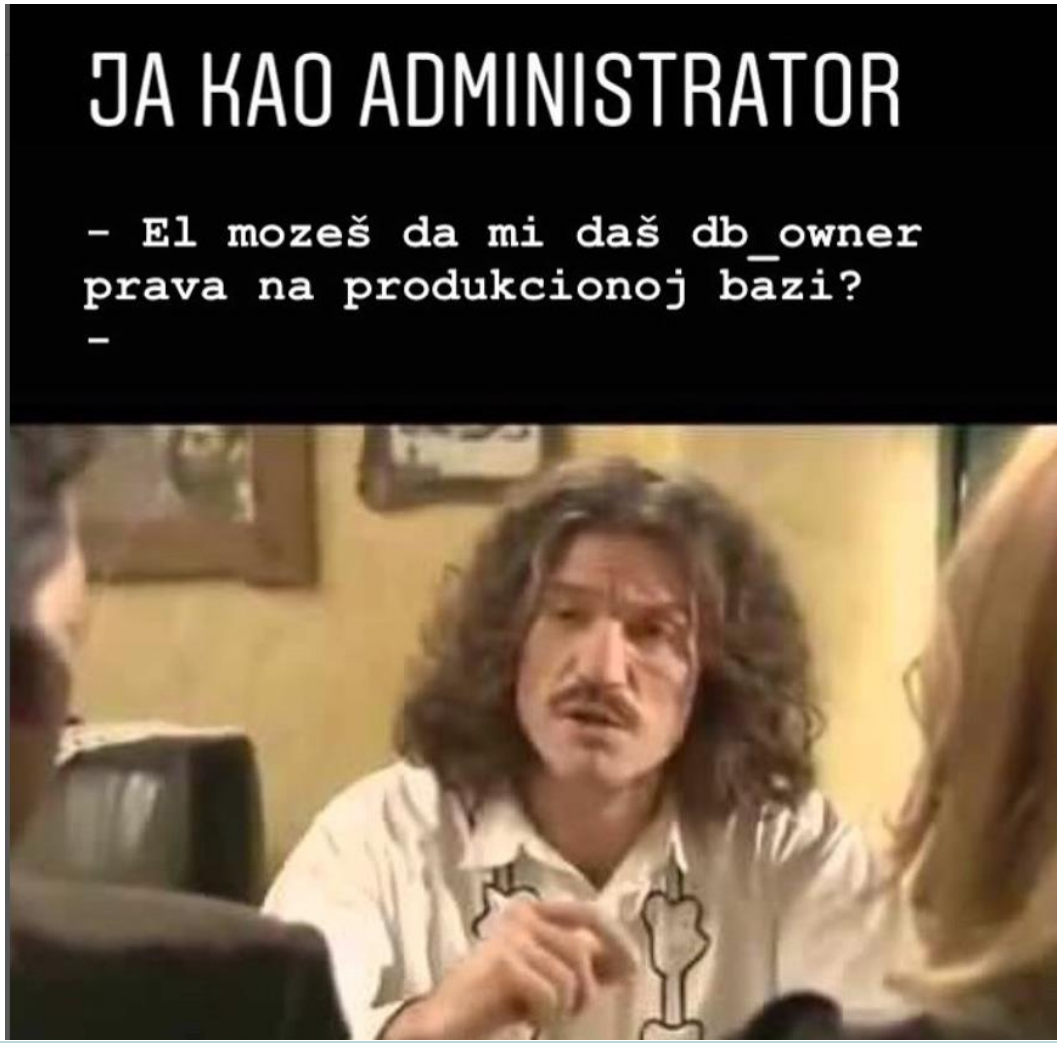
  u suprotnom

- ```
  Event ID: 823. The operating system returned error 665(The requested operation could not be completed due to a file system limitation) to SQL Server during a write at offset 0x0000248637e000 in file 'C:\MSSQL\DATA01\xxx'. Additional messages in the SQL Server error log and operating system error log may provide more detail. This is a severe system-level error condition that threatens database integrity and must be corrected immediately. Complete a full database consistency check (DBCC CHECKDB). This error can be caused by many factors; for more information, see SQL Server Books Online.
  ```
-
  ```

KORISNICI I PRISTUP SERVERIMA I BAZAMA

- Ko treba da ima pristup produkcionim serverima?
  - Servisi, aplikacije i monitoring alati
  - DBAs i, izuzetno, ostali performans trablšuteri

- Samo da odradimo rollout, posle ne mora!
- Mora developer nešto da proveri
- Nemamo admina, pa aj da svi imamo prava na serveru
- Više prava za korisnike je more agile, ovako moramo da čekamo, a to smanjuje produktivnost!
- Pa šta ako malo koristi dynamic SQL?!



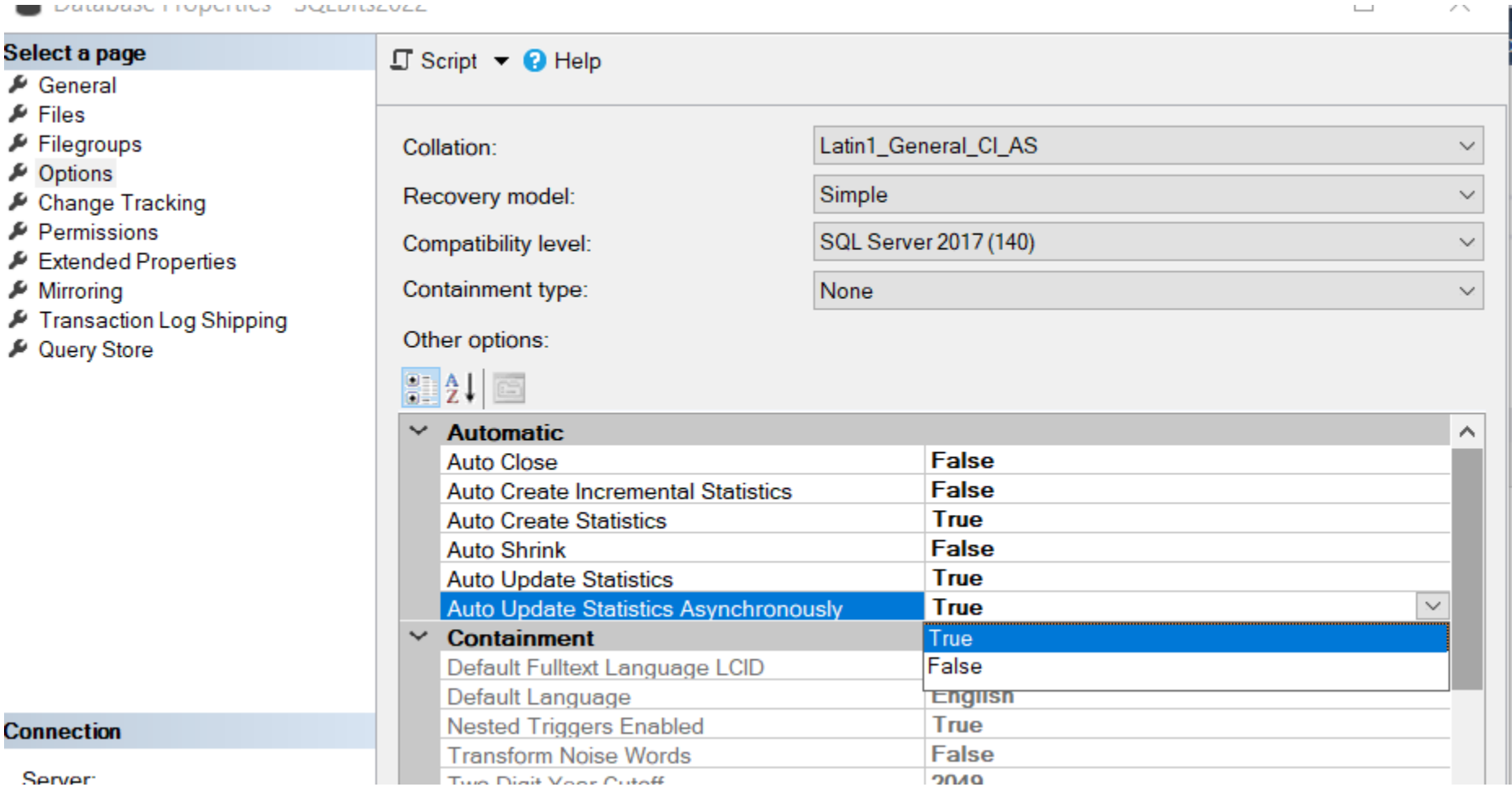
ODGOVORNOST ZA FUKCIONISANJE SERVERA I BAZA

STATISTIKA

- Opcija Auto Update Statistics Asynchronously
- Pogrešne statistike za velike tabele
- Troškovi automatskog ažuriranja statistika

AUTO UPDATE STATISTICS ASYNCHRONOUSLY

- Moja preporuka je da je postavi na TRUE



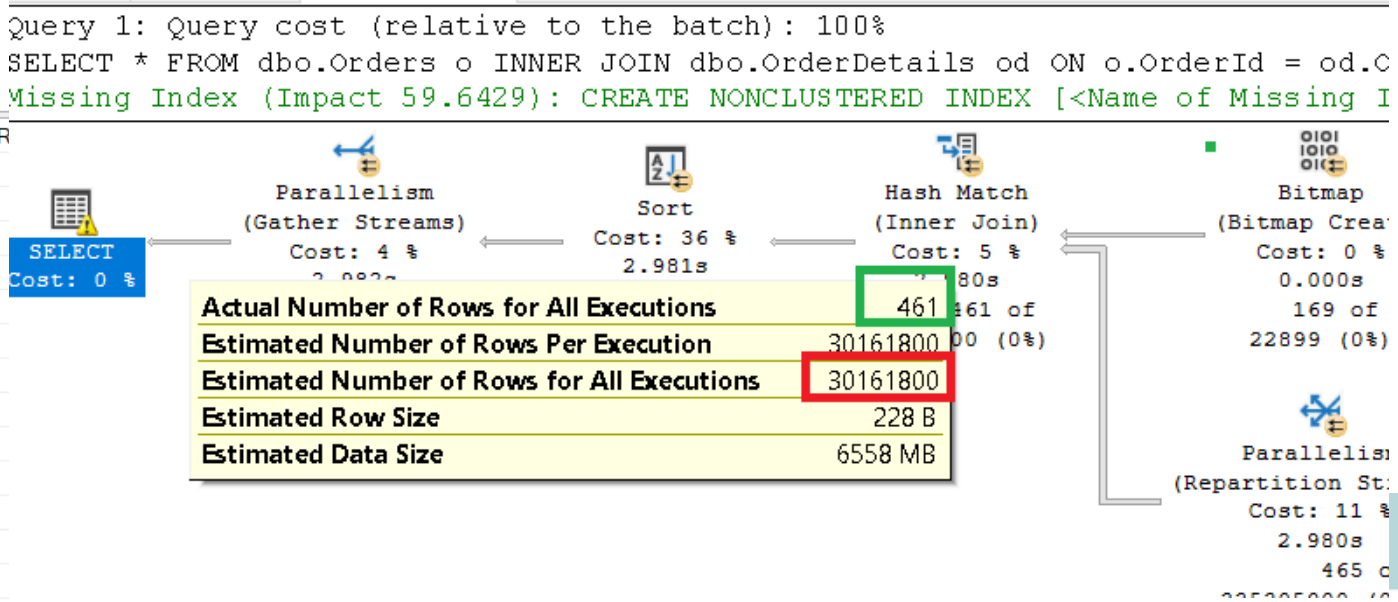
POGREŠNE STATISTIKE ZA VELIKE TABELE

- Auto update stats za velike tabele ide sa default sample rate
- To obično znači da imate ozbiljne overestimacije u planovima
- Rešenje je da naterate SQL Server da ih ažurira sa većim uzorkom (10, 20, 50%) ili da Vi preuzmete ažuriranje

	Name	Updated	Rows	Rows Sampled	Steps	Density	Average key
1	ix1	May 9 2022 1:42PM	100000000	452404	192	0,1711843	8

	All density	Average Length	Columns
1	1,309672E-05	4	CustomerId
2	1E-08	8	CustomerId, OrderId

	RANGE_HI_KEY	RANGE_ROWS	EQ_ROWS	DISTINCT_RANGE_ROWS	AVG_RANGE_R
25	1403733	368070,5	14972,36	283	1300,137
26	1508238	1312011	14972,36	860	1524,993
27	1563090	447211,2	14972,36	275	1625,807
28	1601003	550447,8	14972,36	313	1757,394
29	1638003	345301	14972,36	234	1476,209
30	1682762	587586,4	14972,36	306	1919,02
31	1713920	450748,2	14972,36	321	1403,11
32	1764094	552216,3	14532	332	1661,84
33	1805344	576975,4	22898,91	323	1784,881
34	1829988	455169,5	14972,36	208	2190,321
35	1882523	673137,9	14972,36	441	1527,386
36	1903331	336458,4	14972,36	222	1516,508
37	1948376	643073,3	14972,36	244	2636,091



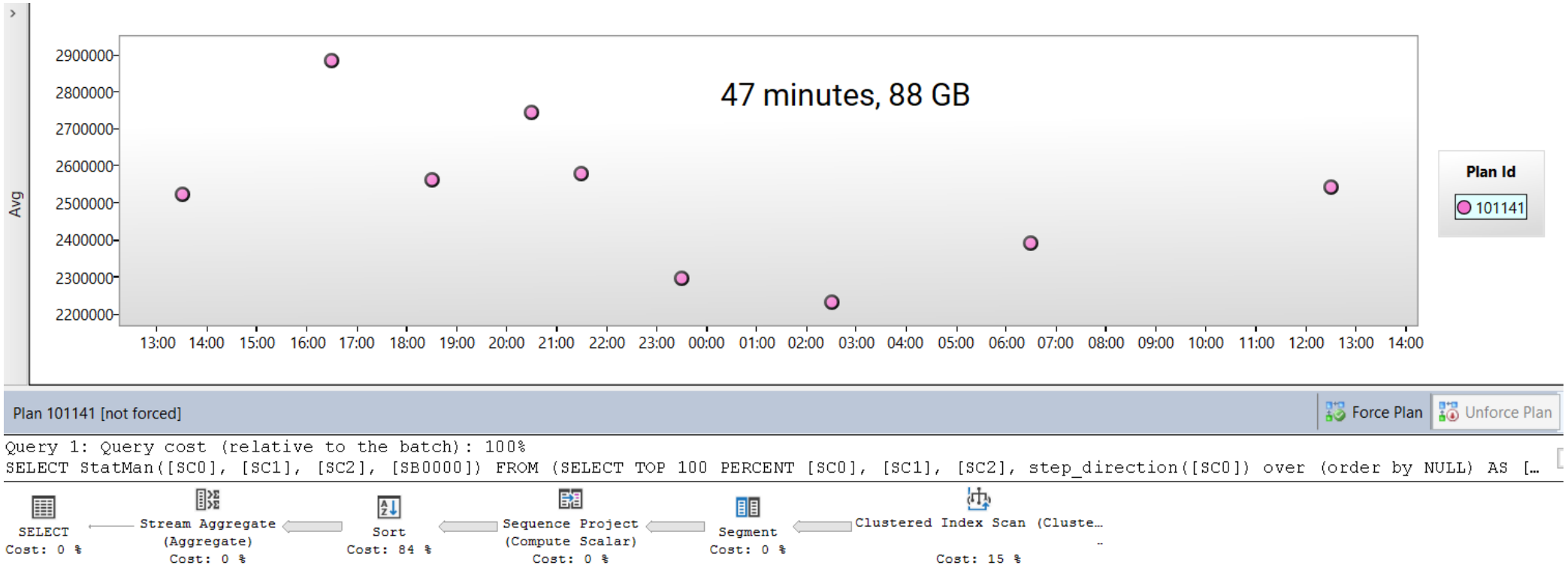
TROŠKOVI AUTOMATSKOG AŽURIRANJA STATISTIKA

- Automatsko ažuriranje se događa nakon SQRT(1000 * Table Cardinality)
- Za tabelu sa 10^9 redova, update se pali nakon 10^6 promena
  - Ne nakon promenjenih milion redova, već nakon milion promena
  - Hiljadu puta promenite hiljadu redova i eto automatskog update-a

- Automatsko ažuriranje za statistike na velikim tabelama je veoma skupo!

TROŠKOVI AUTOMATSKOG AŽURIRANJA STATISTIKA

- 9 puta dnevno, prosek 47 minuta, troši 88 GB
- Rešenje: preuzeti brigu o ažuriranju statistika za najveće i najznačajnije tabele!



PROBLEMI SA INDEKS MENADŽMENTOM

- Uglavnom nema indeks menadžmenta, to je što je
- Eventualno se radi automatski index rebuild, ko ima O.H. skript
- Čak i inicijalno dobro dizajnirani indeksi mogu da postanu neefikasni

- Treba raidti redovno servisiranje:
  - Identifikovati redundantne indekse
  - Konsolidacija indeksa
  - Uklanjanje indeks koji se ne koriste
  - Dodavanje novih indeksa koji bi poboljšali performanse
  - Unapređenje znanja o indeksima
  - Pametniji menadžment glede fragmentacije indeksa

MILOŠ
-SQL-

DATABASE DEVELOPMENT

TIPIČNE GREŠKE U DEVELOPMENTU (1/2)

- Potcenjenost značaj database dizajna
- Loši indeksi
- Nedostatak koncepta arhiviranja ili particionisanja
- Reporti na produkcionom serveru
- Neumerena upotreba hinta NOLOCK
- Nedovoljno znanje Transact-SQL-a
- Korišćenje ORM alata

TIPIČNE GREŠKE U DEVELOPMENTU (2/2)

- LOCK eskalacija prilikom brisanja ili apdejta na velikim tabelama
- Nepotrebna serijalizacija
- Redosled tabela kod JOIN operatora
- TOP 1 & DISTINCT
- Pogrešna očekivanja od grešaka i izuzetaka
- Operator BETWEEEN i datumi
- NOT IN & NULL

POTCENJIVANJE DATABASE DESIGNA

- Greške u dizajnu nisu odmah vidljive
  - mogu mnogo da utiču na performanse
  - veoma teško se ispravljaju

- Tipične greške:
  - neadekvatan tip podataka
  - izbegavanje CHECK i UNIQUE constrainta
  - mnoštvo NULL kolona
  - nepoštovanje pravila normalizacije

LOŠI INDEKSI

- Nedovoljno znanja o dizajnu indeksa
- Nedovoljno vremena za testiranje
- Izostanak evaluacije indeksa nakon određenog vremena
- Slepo preuzimanje saveta iz SSMS-a ili sys.dm_db_missing_index_details

ARHIVIRANJE/RAZDVAJANJE

- Koncept „podaci ostaju u glavnoj tabeli do sudnjega dana“

- Ako ne postoji arhiviranje ili razdvajanje podataka
  - Teže je za management
  - Duži bekap i restore
  - Problemi sa upitima koji skaliraju linearno
  - Novi indeksi su preskupi

I OLTP TRANSAKCIJE I IZVEŠTAJI IZ ISTE BAZE

- Sve je u glavnoj bazi
  - i aktuelni podaci i arhiva
  - sve se pretražuje
- Svi izveštaji idu iz glavne baze
  - agregacije, skeniranja tabela, intezivno korišćenje CPU
  - Fleksibilni izveštaji i pretrage
  - from and to datumi, opcioni parametri
  - Najčešće nije neophodno, ali...
- Nemoguće skaliranje!

NOLOCK

- Zašto developeri koriste NOLOCK?
  - Zato što ima manje blokiranja
  - Zato što im je reko DBA (jer problemi koji time mogu da nastanu nisu DBA problemi)
- Problemi koje donosi NOLOCK
  - Pogrešan rezultat (preskočeni redovi, dupli redovi...)
  - Greška prilikom kompajliranja
  - Niti developeri niti admini ne nauče da propisno upravljaju izazovima konkurentnih procesa u SQL Serveru

LACK OF TRANSACT-SQL KNOWLEDGE

- Najvažnija veština za developera koji radi sa SQL Serverom – Transact-SQL
- Ko zna da piše dobar kod izbeći će mnoge probleme sa performansama
- Od TSQL znanja zavise dobre i loše odluke u smislu arhitekture projekta

- Tipični znaci da ne zna da piše dobar kod:
  - Koristi skalarne funkcije, nije ni čuo za inline-table valued funkcije
  - Koristi ORM tools, Entity Framework etc.
  - Ne koristi sa Window funkcije i APPLY operator
  - Moksuje temporary table i table varijable
  - Kako kveri uspori misli da je rešenje u NoSQL

LOCK ESCALATION

- Više od 5000 pojedinačnih lokova => SQL Server može da zaključe tabelu ili indeks
- Rešenje: UPDATE/DELETE u manjim blokovima (chunks)

```
DECLARE @current INT = 0;
DECLARE @batch_size INT = 5000;
DECLARE @cnt INT = 1;

WHILE (@cnt > 0)
BEGIN
    DELETE
    FROM dbo.MyTestTable
    WHERE SomeDate <'20210101'
    AND id > @current AND id <= @current + @batch_size


    SET @cnt = @@ROWCOUNT

    SET @current = @current + @batch_size
END
```

REDOSLED TABELA KOD JOIN OPERATORA

```
USE AdventureWorks2019;
SELECT c.CustomerID, c.AccountNumber, o.SalesOrderID, o.OrderDate
FROM Sales.Customer c
LEFT JOIN Sales.SalesOrderHeader o ON c.CustomerID = o.CustomerID
WHERE c.CustomerID IN (1, 11900);
```

Results | Messages

	CustomerID	AccountNumber	SalesOrderID	OrderDate
1	1	AW00000001	NULL	NULL
2	11900	AW00011900	45738	2012-02-21 00:00:00.000
3	11900	AW00011900	54097	2013-08-08 00:00:00.000
4	11900	AW00011900	61969	2013-12-11 00:00:00.000

REDOSLED TABELA KOD JOIN OPERATORA

```
SELECT c.CustomerID, c.AccountNumber, o.SalesOrderID, o.OrderDate, od.ProductID
FROM Sales.Customer c
LEFT JOIN Sales.SalesOrderHeader o ON c.CustomerID = o.CustomerID
INNER JOIN Sales.SalesOrderDetail od ON o.SalesOrderID = od.SalesOrderID
WHERE c.CustomerID IN (1, 11900);
```

109 %

Results | Messages

	CustomerID	AccountNumber	SalesOrderID	OrderDate	ProductID
1	11900	AW00011900	45738	2012-02-21 00:00:00.000	773
2	11900	AW00011900	54097	2013-08-08 00:00:00.000	779
3	11900	AW00011900	54097	2013-08-08 00:00:00.000	880
4	11900	AW00011900	61969	2013-12-11 00:00:00.000	966
5	11900	AW00011900	61969	2013-12-11 00:00:00.000	711
6	11900	AW00011900	61969	2013-12-11 00:00:00.000	882
7	11900	AW00011900	61969	2013-12-11 00:00:00.000	712

CustomerId = 1

REDOSLED TABELA KOD JOIN OPERATORA

```
SELECT c.CustomerID, c.AccountNumber, o.SalesOrderID, o.OrderDate, od.ProductID
FROM Sales.Customer c
LEFT JOIN
(
        Sales.SalesOrderHeader o INNER JOIN Sales.SalesOrderDetail od ON o.SalesOrderID =
        od.SalesOrderID
) ON c.CustomerID = o.CustomerID
WHERE c.CustomerID IN (1, 11900);
```

Results | Messages

	CustomerID	AccountNumber	SalesOrderID	OrderDate	ProductID
1	1	AW00000001	NULL	NULL	NULL
2	11900	AW00011900	45738	2012-02-21 00:00:00.000	773
3	11900	AW00011900	54097	2013-08-08 00:00:00.000	779
4	11900	AW00011900	54097	2013-08-08 00:00:00.000	880
5	11900	AW00011900	61969	2013-12-11 00:00:00.000	966
6	11900	AW00011900	61969	2013-12-11 00:00:00.000	711
7	11900	AW00011900	61969	2013-12-11 00:00:00.000	882
8	11900	AW00011900	61969	2013-12-11 00:00:00.000	712

TOP 1 & DISTINCT

```
SELECT c.id, c.custname,
(
    SELECT p.amount
    FROM dbo.NextPayment p WHERE p.custid = c.id
) x
FROM dbo.Customers c
```

```
Msg 512, Level 16, State 1, Line 46
Subquery returned more than 1 value. This is not permitted when the subquery follows =, !=,
<, <= , >, >= or when the subquery is used as an expression.
```

```
SELECT c.id, c.custname,
(
    SELECT TOP (1) p.amount
    FROM dbo.NextPayment p WHERE p.custid = c.id
) x
FROM dbo.Customers c
```

TOP 1 & DISTINCT

```sql
SELECT p.BusinessEntityId, FirstName, MiddleName, LastName, Title, Suffix, EmailPromotion
FROM Person.Person p
INNER JOIN Person.BusinessEntityAddress pa ON p.BusinessEntityID = pa.BusinessEntityID
INNER JOIN Person.Address a ON a.AddressID = pa.AddressID
WHERE p.BusinessEntityId BETWEEN 2994 AND 2996;
```

```sql
SELECT DISTINCT p.BusinessEntityId, FirstName, MiddleName, LastName, Title, Suffix,
EmailPromotion
FROM Person.Person p
INNER JOIN Person.BusinessEntityAddress pa ON p.BusinessEntityID = pa.BusinessEntityID
INNER JOIN Person.Address a ON a.AddressID = pa.AddressID
WHERE p.BusinessEntityId BETWEEN 2994 AND 2996;
```

MERGE PK VIOLATION

```sql
DROP TABLE IF EXISTS dbo.MergeExample;
CREATE TABLE dbo.MergeExample
(
   id INT NOT NULL,
   c1 NVARCHAR(50) NOT NULL,
   c2 NVARCHAR(30) NOT NULL,
   c3 NVARCHAR(30) NOT NULL,
   c4 INT NOT NULL
   CONSTRAINT PK_MergeExample PRIMARY KEY(id)
);
GO
```

MERGE PK VIOLATION

```
--session 1
BEGIN TRY
  WHILE 1 = 1
  BEGIN
    MERGE INTO dbo.MergeExample AS t
    USING (SELECT CHECKSUM(SYSDATETIME()), N'abc', N'test', 'blah',1)
          AS s(id,c1,c2,c3,c4)
        ON s.id = t.id
    WHEN MATCHED THEN UPDATE
```

145 %

Messages

```
(1 row affected)

(1 row affected)

(1 row affected)

(0 rows affected)
Msg 2627, Level 14, State 1, Line 4
Violation of PRIMARY KEY constraint 'PK_MergeExample'. Cannot insert duplicate key in object 'dbo.MergeExample'. The duplicate key value is (943271018).
```

MERGE PK VIOLATION

```
--session 2
BEGIN TRY
  WHILE 1 = 1
  BEGIN
    MERGE INTO dbo.MergeExample AS t
    USING (SELECT CHECKSUM(SYSDATETIME()), N'abc', N'test', 'blah',1)
          AS s(id,c1,c2,c3,c4)
       ON s.id = t.id
    WHEN MATCHED THEN UPDATE
```

%

Messages

```
(0 rows affected)
Msg 2627, Level 14, State 1, Line 4
Violation of PRIMARY KEY constraint 'PK_MergeExample'. Cannot insert duplicate key in object 'dbo.MergeExample'. The duplicate key value is (943322470).
```

MERGE PK VIOLATION

- Rešenje?

```
--session 1
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
BEGIN TRY
  WHILE 1 = 1
  BEGIN
    MERGE INTO dbo.MergeExample AS t
    USING (SELECT CHECKSUM(SYSDATETIME()), N'abc', N'test', 'blah',1)
          AS s(id,c1,c2,c3,c4)
```

45 %

Messages

```
(1 row affected)

(1 row affected)
```

MERGE PK VIOLATION

- Rešenje 1:
  - ignoriši

```
BEGIN CATCH
    IF ERROR_NUMBER() <> 2627
    THROW;
END CATCH;
```

- Rešenje 2:
  - Probaj ponovo

```
DECLARE @retries TINYINT = 2;
WHILE (@retries > 0)
BEGIN
  BEGIN TRY
                WHEN MATCHED  THEN UPDATE
---
    SET @retries = 0;
  END TRY

  BEGIN CATCH
IF ERROR_NUMBER() <> 2627
SET @retries -= 1;
    ELSE
      THROW;
END CATCH;
END
```

POGREŠNA OČEKIVANJA OD GREŠAKA

```
DROP TABLE IF EXISTS dbo.T1;
CREATE TABLE dbo.T1(id INT PRIMARY KEY);
GO

BEGIN TRAN
    INSERT INTO T1(id) VALUES(1);
    INSERT INTO T1(id) VALUES(1);
COMMIT

SELECT COUNT(*) FROM T1;
```

- Šta će da ispiše SELECT COUNT(*)?

- 0
- 1
- Izbaciće exception

OPERATOR BETWEEEN I DATUMI

```
USE AdventureWorks2019;

SELECT * FROM Sales.SalesOrderHeader
WHERE OrderDate BETWEEN '20130101 00:00:00.000' AND '20131231 23:59:59.999'
ORDER BY OrderDate DESC;
```

2 %

Results | Messages

SalesOrderID	RevisionNumber	OrderDate	DueDate	ShipDate	Status	OnlineOrderFlag	SalesOrderNumber	PurchaseOrderNumber	AccountNumber	CustomerID	SalesPersonID	TerritoryID	BillTo
63363	8	2014-01-01 00:00:00.000	2014-01-13 00:00:00.000	2014-01-08 00:00:00.000	5	1	SO63363	NULL	10-4030-014685	14685	NULL	9	2354
63364	8	2014-01-01 00:00:00.000	2014-01-13 00:00:00.000	2014-01-08 00:00:00.000	5	1	SO63364	NULL	10-4030-018299	18299	NULL	9	2255
63365	8	2014-01-01 00:00:00.000	2014-01-13 00:00:00.000	2014-01-08 00:00:00.000	5	1	SO63365	NULL	10-4030-021279	21279	NULL	9	2876
63366	8	2014-01-01 00:00:00.000	2014-01-13 00:00:00.000	2014-01-08 00:00:00.000	5	1	SO63366	NULL	10-4030-014518	14518	NULL	9	1630
63367	8	2014-01-01 00:00:00.000	2014-01-13 00:00:00.000	2014-01-08 00:00:00.000	5	1	SO63367	NULL	10-4030-013605	13605	NULL	7	2706
63368	8	2014-01-01 00:00:00.000	2014-01-13 00:00:00.000	2014-01-08 00:00:00.000	5	1	SO63368	NULL	10-4030-014170	14170	NULL	7	1894

```
SELECT * FROM Sales.SalesOrderHeader
WHERE OrderDate >= '20130101' AND OrderDate < '20140101'
ORDER BY OrderDate DESC;
```

MILOŠ
-SQL-

NOT IN I KOLONE KOJE PRIHVATAJU NULL

```
SELECT * FROM dbo.Color;
```

Results | Messages

id	name
1	Black
2	White
3	Purple

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

Results | Messages

id	name
1	Black
2	White

NOT EXISTS JE INTUITIVNIJI

```
SELECT * FROM dbo.Color c WHERE EXISTS(
SELECT 1 FROM AdventureWorks2019.Production.Product p WHERE p.Color = c.name)
UNION ALL
SELECT * FROM dbo.Color c WHERE NOT EXISTS(
SELECT 1 FROM AdventureWorks2019.Production.Product p WHERE p.Color = c.name);
```

20 %

Results | Messages

	id	name
1	1	Black
2	2	White
3	3	Purple