Size Does Matter

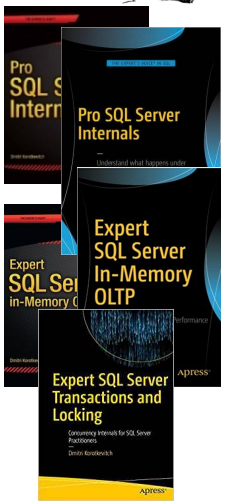
3-5 ways to reduce DB size and improve performance

Dmitri Korotkevitch
http://aboutsqlserver.com

Good morning!

20+ years of experience in IT
15+ years of experience working with SQL Server
Director, Database Services @ chewy.com
Microsoft Data Platform MVP
Microsoft Certified Master
Author

- Pro SQL Server Internals (v1-2)
- Expert SQL Server In-Memory OLTP (v1-2)
- Expert SQL Server Transactions and Locking



Blog: http://aboutsqlserver.com
Email: dk@aboutsqlserver.com

Why bother reducing DB Size?

Improves Performance

Simplifies SLA, RTO, RPO Compliance

Makes Administration and Maintenance Easier

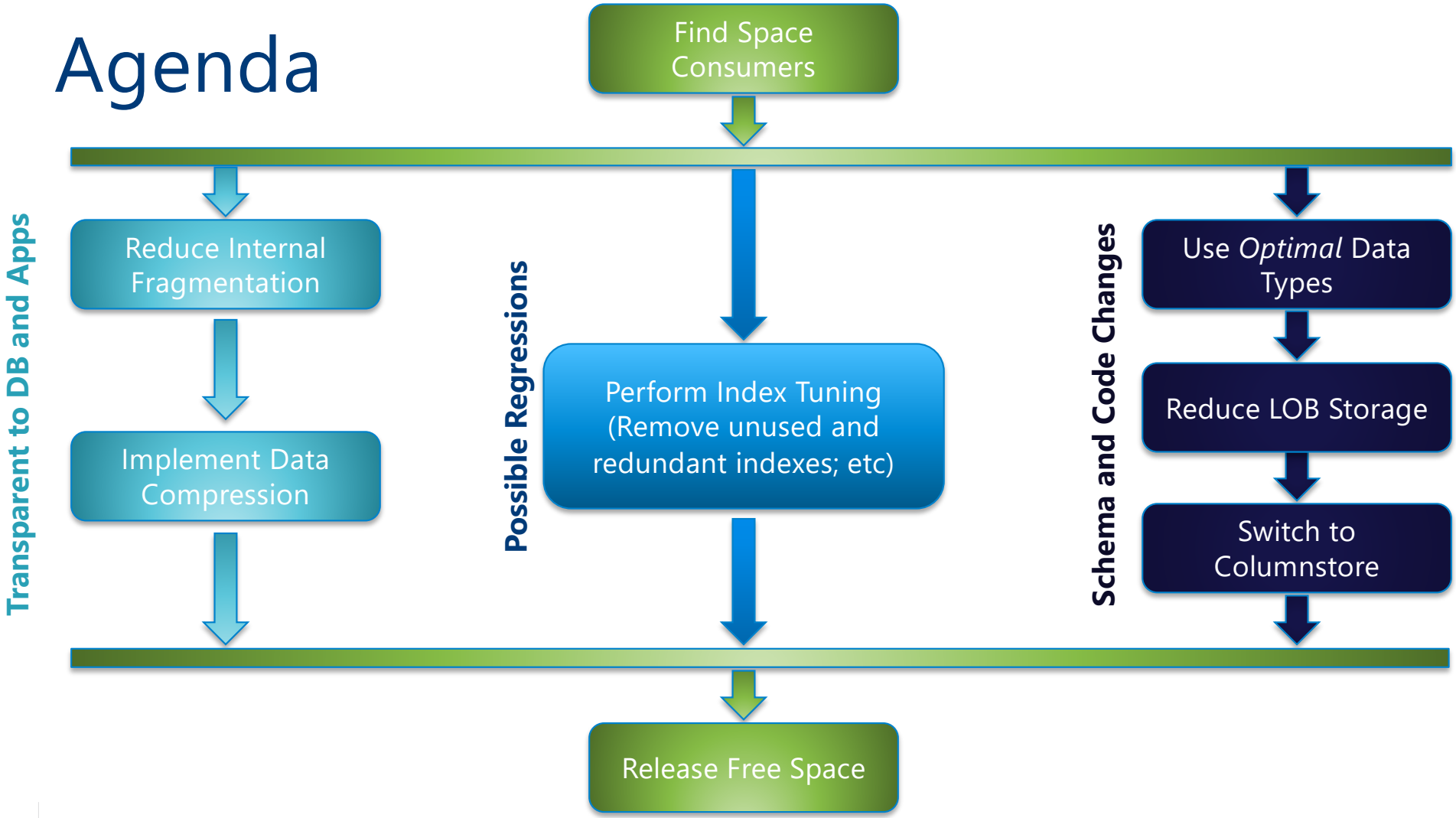
Reduces Hardware Cost

Picture from: https://openclipart.org/detail/172012/award-by-gnokii-172012

DBCC_SMARTSHRINK()

Demo

Agenda


Find Space Consumers
Transparent to DB and Apps
Reduce Internal Fragmentation
Implement Data Compression
Possible Regressions
Perform Index Tuning (Remove unused and redundant indexes; etc)
Schema and Code Changes
Use *Optimal* Data Types
Reduce LOB Storage
Switch to Columnstore
Release Free Space

Best Practices

Auto Shrink is OFF

- Greatly increases index fragmentation
- Contributes to I/O, CPU load and T-Log overhead
- Absolutely useless

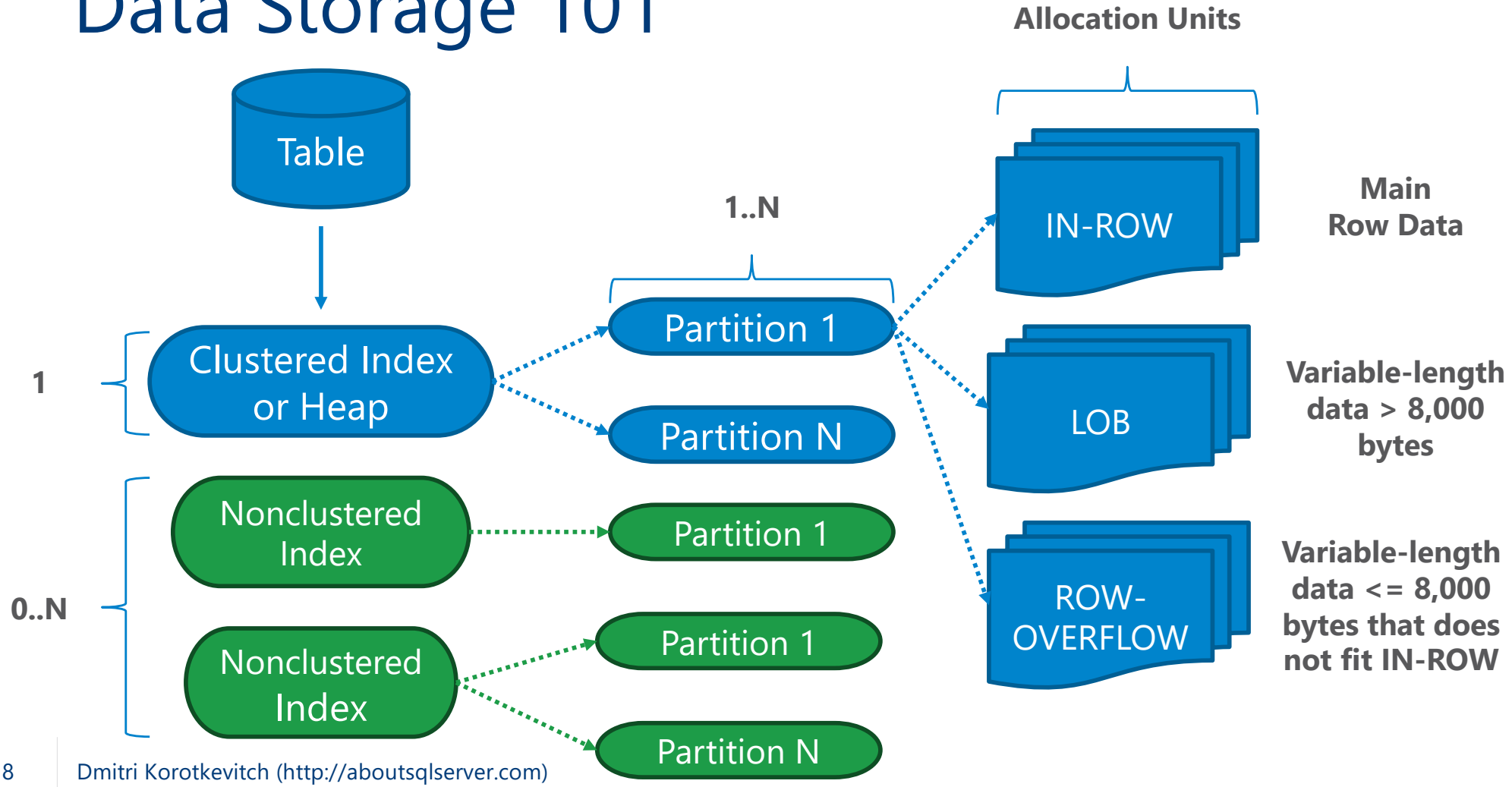
Instant File Initialization is ON

- Prevents zeroing-out of the newly allocated space in the data files
- Speed-up data file (auto)-growth and database restore
- Does not affect T-Log growth (it is always zeroing out)

Instant File Initialization

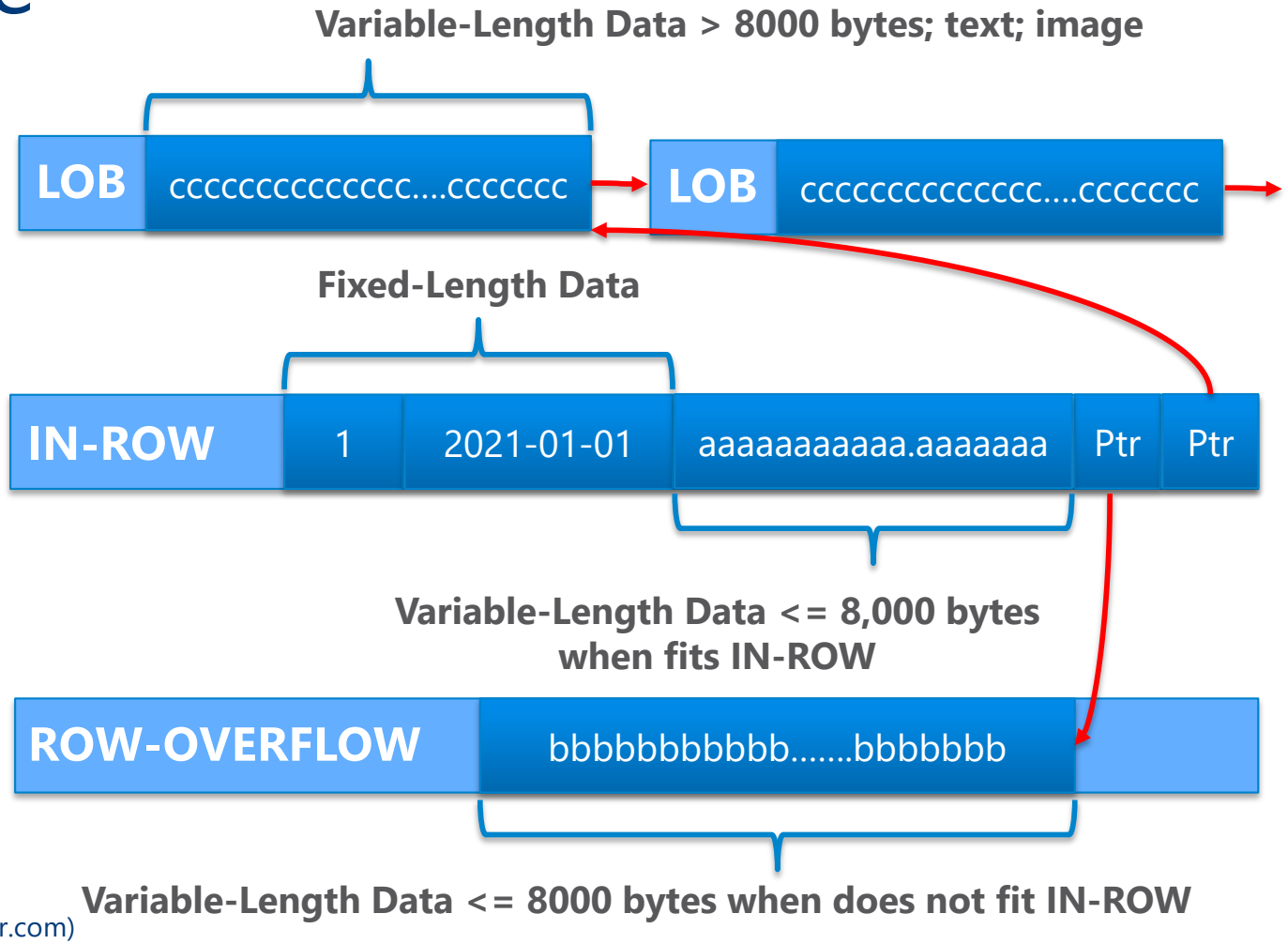
Demo

Data Storage 101



Data Storage

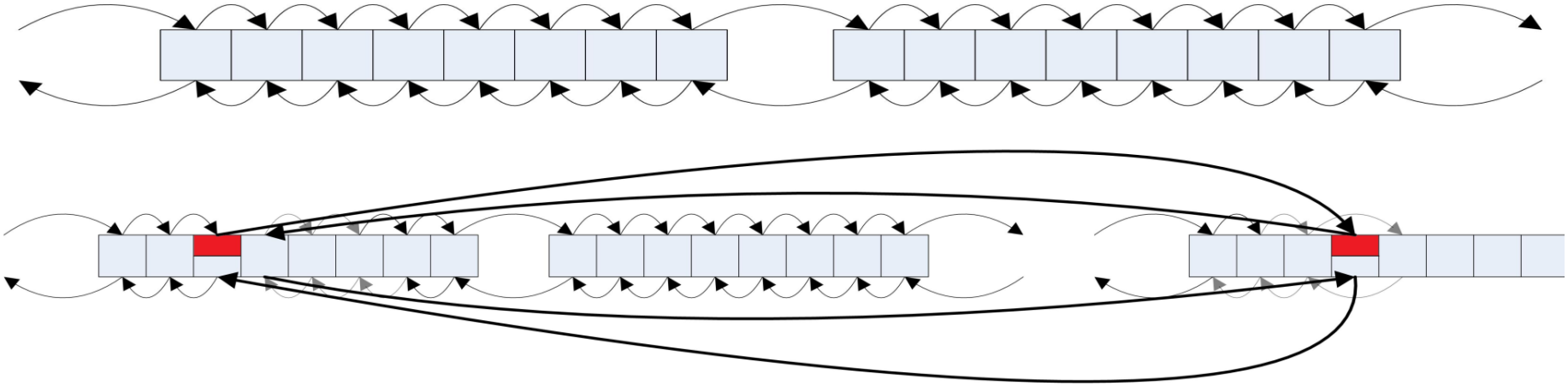
```
create table T
(
    ID int,
    DateCol datetime,
    VarCol1 varchar(max),
    VarCol2 varchar(5000),
    VarCol3 varchar(max)
);

insert into T values
(
    1
    ,'2021-01-01'
    ,REPLICATE('a',5000)
    ,REPLICATE('b',5000)
    ,REPLICATE('c',32000)
);
```



Find Space Consumers

Demo

Page Split & Fragmentation



Page Split moves ~50% of the data to another page

FILLFACTOR allows to reserve some space on the page

- Applies only at CREATE INDEX/ALTER INDEX REBUILD stages
- Decreases Page Splits / Fragmentation; however, increases index size

Internal Fragmentation

Demo

Data Compression

Uncompressed rows:

- Fixed-length columns: Storage size depends on the data type
- Variable-length columns: Storage size depends on the data value + 2 bytes (in most cases)

ROW compression:

- Reduces fixed-length columns storage overhead
- Extra 4-bit per column. Bad choice for fully-populated data types

PAGE compression:

- ROW compression + Prefix compression + Dictionary compression
- Done on data page scope

ROW and PAGE compression works only with IN-ROW data

Data Compression

Demo

Data Compression

My rule of thumb:

- Enable ROW compression in OLTP
- Enable PAGE compression for static data
  - Partition the data

PAGE compression may help even with the volatile data when disk is the bottleneck

LOB Compression (Pre SS2016)

```
[Microsoft.SqlServer.Server.SqlFunction
    (IsDeterministic = true, IsPrecise = true,
        DataAccess = DataAccessKind.None)]
public static SqlBytes BinaryCompress(SqlBytes input)
{
     if (input.IsNull)
         return SqlBytes.Null;

    using (MemoryStream result = new MemoryStream())
    {
        using (DeflateStream deflateStream = new DeflateStream(resul
            CompressionMode.Compress, true))
        {
            deflateStream.Write(input.Buffer, 0, input.Buffer.Length
            deflateStream.Flush();
            deflateStream.Close();
        }
        return new SqlBytes(result.ToArray());
    }
}
```

```
[Microsoft.SqlServer.Server.SqlFunction
    (IsDeterministic = true, IsPrecise = true,
        DataAccess = DataAccessKind.None)]
public static SqlBytes BinaryDecompress(SqlBytes input)
{
    if (input.IsNull)
        return SqlBytes.Null;

    int batchSize = 32768;
    byte[] buf = new byte[batchSize];

    using (MemoryStream result = new MemoryStream())
    {
        using (DeflateStream deflateStream =
                new DeflateStream(input.Stream,
                    CompressionMode.Decompress, true))
        {
            int bytesRead;
            while ((bytesRead = deflateStream.Read(buf, 0, batchSize)) > 0)
                result.Write(buf, 0, bytesRead);
        }
        return new SqlBytes(result.ToArray());
    }
}
```

LOB Compression (SS 2016+)

Use COMPRESS/DECOMPRESS functions

Consider to store LOB data outside of DB

LOB Compression

Demo

Use Appropriate Data Types

Obvious Mistakes:

- Boolean -> tinyint, smallint, int
- (n)char(N)

What kind of precision do you need?

- DateTime -> SmallDateTime or DateTime2
- Float -> Decimal

Do not use LOB data types unless you need them

Use Appropriate Data Types

1,000 rows - 15K
1,000,000 rows - 15M
365,000,000 rows - 5.4Gb

```
create table Positions
(
    ATime datetime not null, -- 8 bytes
    Latitude float not null, -- 8 bytes
    Longitude float not null, -- 8 bytes
    IsValid int not null, -- 4 bytes
    IsAssistanceUsed int not null, -- 4 bytes
    -- total: 32 bytes
)
```

```
create table Positions2
(
    ATime datetime2(0) not null, -- 6 bytes
    Latitude decimal(9,6) not null, -- 5 bytes
    Longitude decimal(9,6) not null, -- 5 bytes
    IsValid bit not null, -- 1 byte
    IsAssistanceUsed bit not null, -- 0 bytes
    -- total: 17 bytes
)
```

Think about future and do not be cheap

Rebuild indexes after table alteration

ALTER TABLE & Row Size

Demo

sys.dm_db_index_usage_stats

How often index appears in execution plans

- Seeks: Index Seek operation
- Scans: Index Scan operation
- Lookup: Key Lookup or RID Lookup operations
- Reads: Seeks + Scans + Lookups
- Updates: INSERT + UPDATE + DELETE

	Table	Index	Seeks	Scans	Lookups	Reads	Updates	Last Seek	Last Scan
1	[redacted]		74567916	0	755160	75323076	1510669	2016-11-18 07:32:15.437	NULL
2			57940278	0	0	57940278	1510669	2016-11-18 07:32:15.537	NULL
3			16724	0	0	16724	1510669	2016-11-18 07:00:20.827	NULL
4			1003195	1090508	0	2093703	1136428	2016-11-18 07:32:08.700	2016-11-1
5			544501	0	0	544501	1510669	2016-11-18 07:32:10.540	NULL
6			1	0	0	1	680021	2016-11-02 14:17:28.843	NULL
7			5210970	0	0	5210970	1510669	2016-11-18 07:32:14.423	NULL
8			0	0	0	0	1510669	NULL	NULL
9			7049584	0	0	7049584	2403273	2016-11-18 07:32:14.423	NULL
10			377572	0	0	377572	2403273	2016-11-18 07:32:08.750	NULL

sys.dm_db_index_operational_stats

Access methods, I/O, locking, latching activity

- Insert, Update, Delete counts (# of rows)
- singleton_lookup_count: Single-row Index Seek operations
- range_scan: Index Seeks on the range of rows + Index Scans
- LOB and ROW_OVERFLOW statistics
- Lock counts and waits on row- and page-levels
- Page latch count and waits
- Page IO latch count and waits
- And more..

	index_id	Table	Index	range_scan_count	singleton_lookup_count	row_lock_wait_in_ms	page_latch_wait_in_ms	page_io_latch_wait_in_ms
1	1	[redacted]	[redacted]	1411162	3638897399	0	71286	13634302
2	2	[redacted]	[redacted]	774	0	0	338	283589
3	3	[redacted]	[redacted]	760095	0	0	121	329284
4	4	[redacted]	[redacted]	32726	0	0	828	7878183
5	5	[redacted]	[redacted]	358	0	0	66	3138358
6	6	[redacted]	[redacted]	21	0	0	124	138602
7	8	[redacted]	[redacted]	0	0	0	519	68238
8	33	[redacted]	[redacted]	1	0	0	60	574234
9	34	[redacted]	[redacted]	11012	0	0	490	626016
10	35	[redacted]	[redacted]	0	0	0	126	23540

Usage and Operational Stats, and TableInfo

Demo

Index Tuning: Detecting Redundant Indexes

IDX1(A, B) & IDX2(A) -> IDX2 is redundant

IDX3(A) INCLUDE (B) & IDX4(A) INCLUDE (C) -> IDX5(A) INCLUDE(B,C)

IDX6(A,B) & IDX7(A,C) ->

- IDX8(A,B) INCLUDE (C)
- IDX9(A,C) INCLUDE (B)
- Or keep IDX6 and IDX7

Redundant Indexes

Demo

Releasing Empty Space

DBCC SHRINKFILE

- Introduces Fragmentation (consider to REORG indexes afterwards)
- Challenging in case of multiple files in the filegroup
- Extremely slow especially with LOB data

INSERT INTO NewTable SELECT FROM OldTable

- Offline Operation

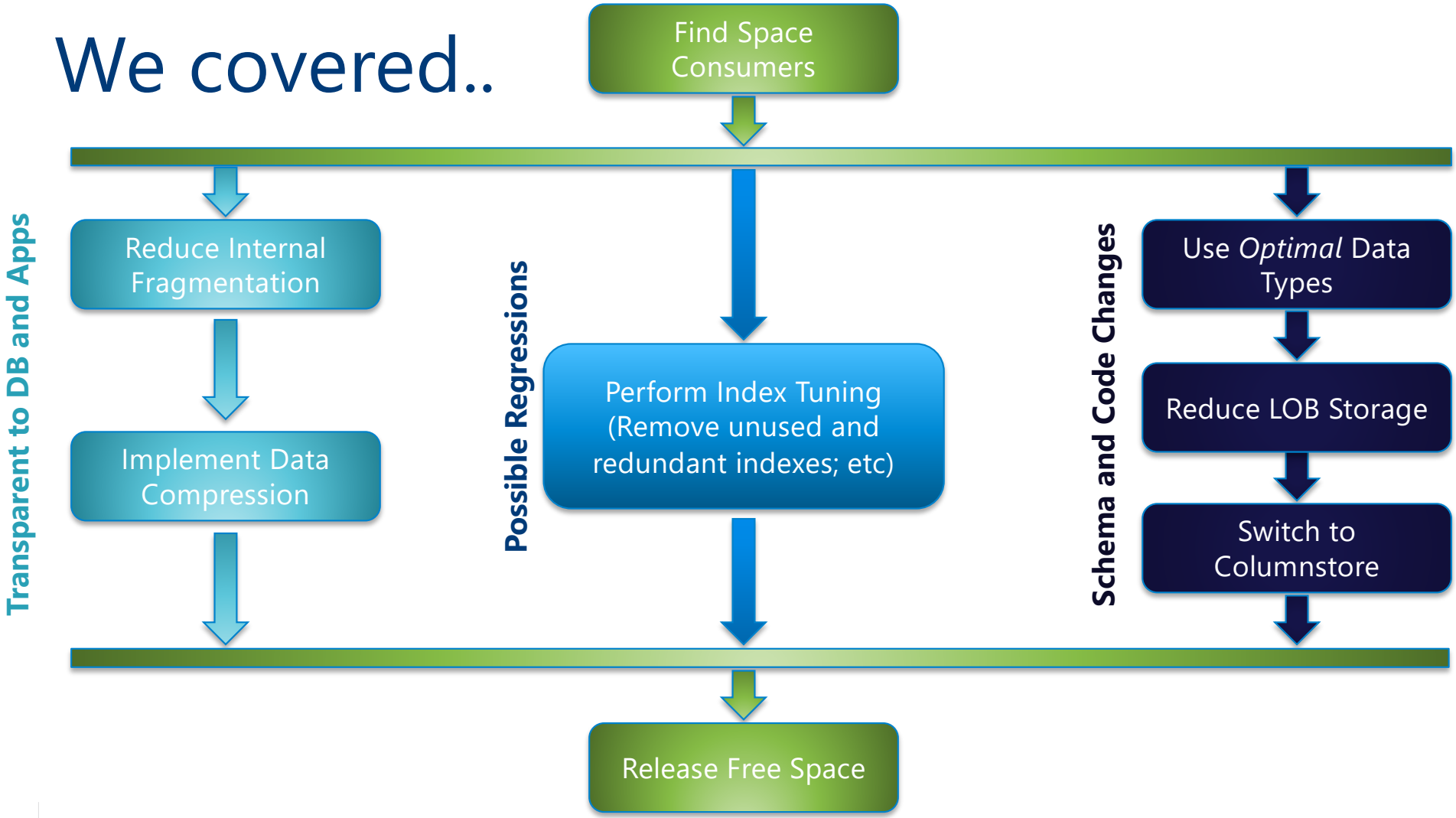
CREATE INDEX WITH (DROP_EXISTING=ON) ON [NewFileGroup]

- Issues with LOB Data

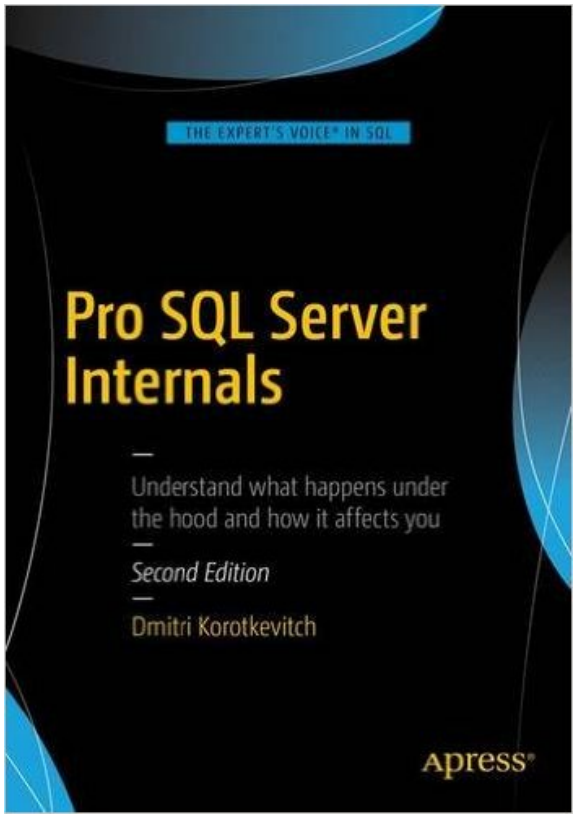
Moving Data

Demo

We covered..


Find Space Consumers
Transparent to DB and Apps
Reduce Internal Fragmentation
Implement Data Compression
Possible Regressions
Perform Index Tuning (Remove unused and redundant indexes; etc)
Schema and Code Changes
Use *Optimal* Data Types
Reduce LOB Storage
Switch to Columnstore
Release Free Space

Additional Resources



Email me anytime:
dk@aboutsqlserver.com

Blog: http://aboutsqlserver.com

- Scripts are available for download: http://aboutsqlserver.com/presentations
- Recent post: "Size Does Matter..": http://goo.gl/J644Zz

Thank You