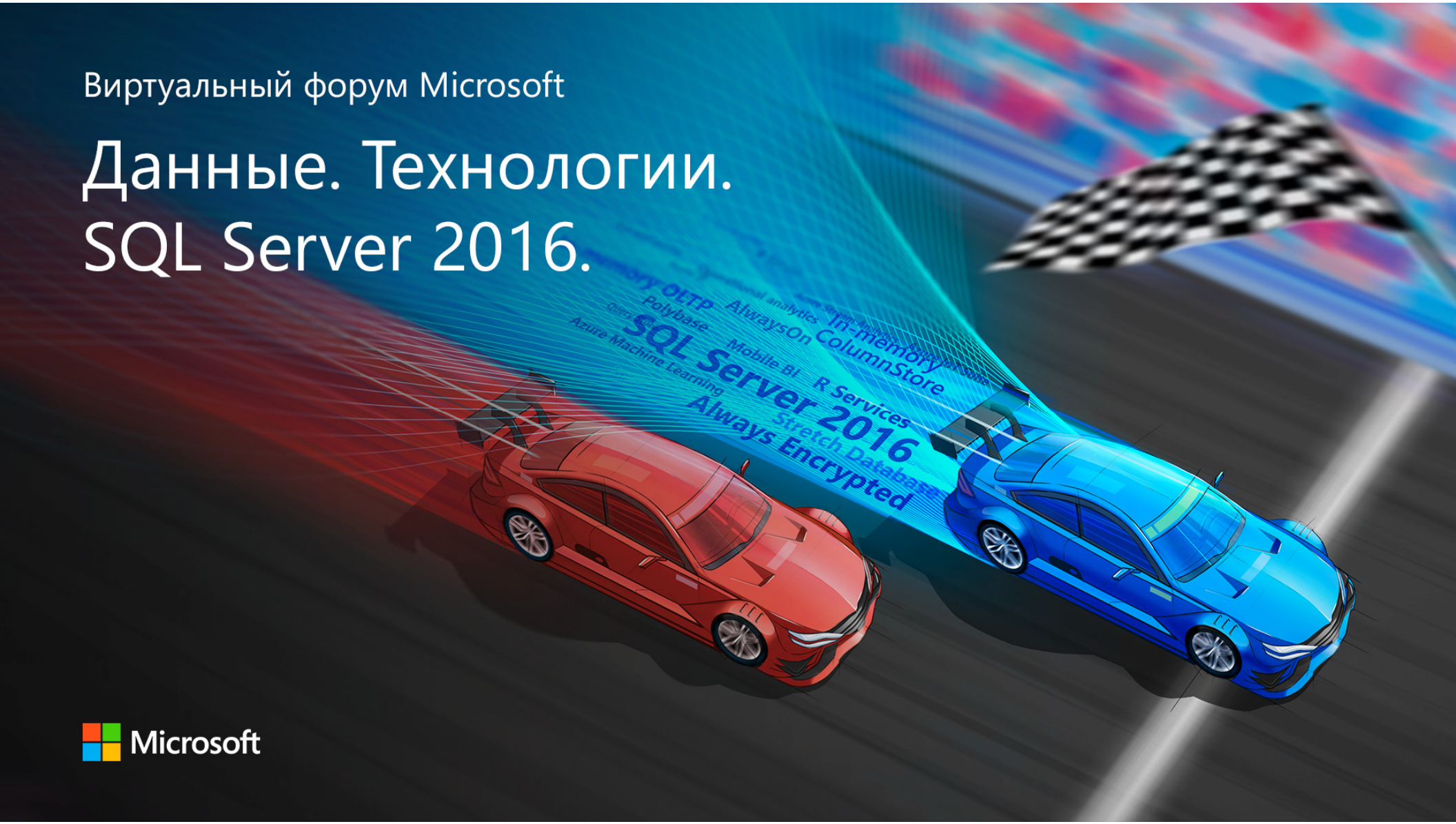
Виртуальный форум Microsoft

Данные. Технологии. SQL Server 2016.



Microsoft

Microsoft

In-Memory OLTP: Architecture, Implementation and SQL Server 2016 Enhancements

Dmitri Korotkevitch
http://aboutsqlserver.com

About me

20+ years in IT

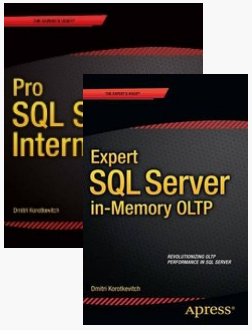
15+ years working with SQL Server

Microsoft Data Platform MVP

Microsoft Certified Master

Author:

- Expert SQL Server In-Memory OLTP
- Pro SQL Server Internals



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

Agenda

Why In-Memory OLTP?

In-Memory OLTP Internal Architecture and Implementation

SQL Server 2016 Enhancements

In-Memory OLTP in the systems with mixed workload

Current Hardware Trends

RAM prices:

Cost of 1GB RAM in 1990: $110,000
Cost of 1GB RAM in 2000: $1,200
Cost of 1GB RAM in 2010: $13
Currently: $6

CPUs have more cores without increase of the clock speed

“Classic Architecture”

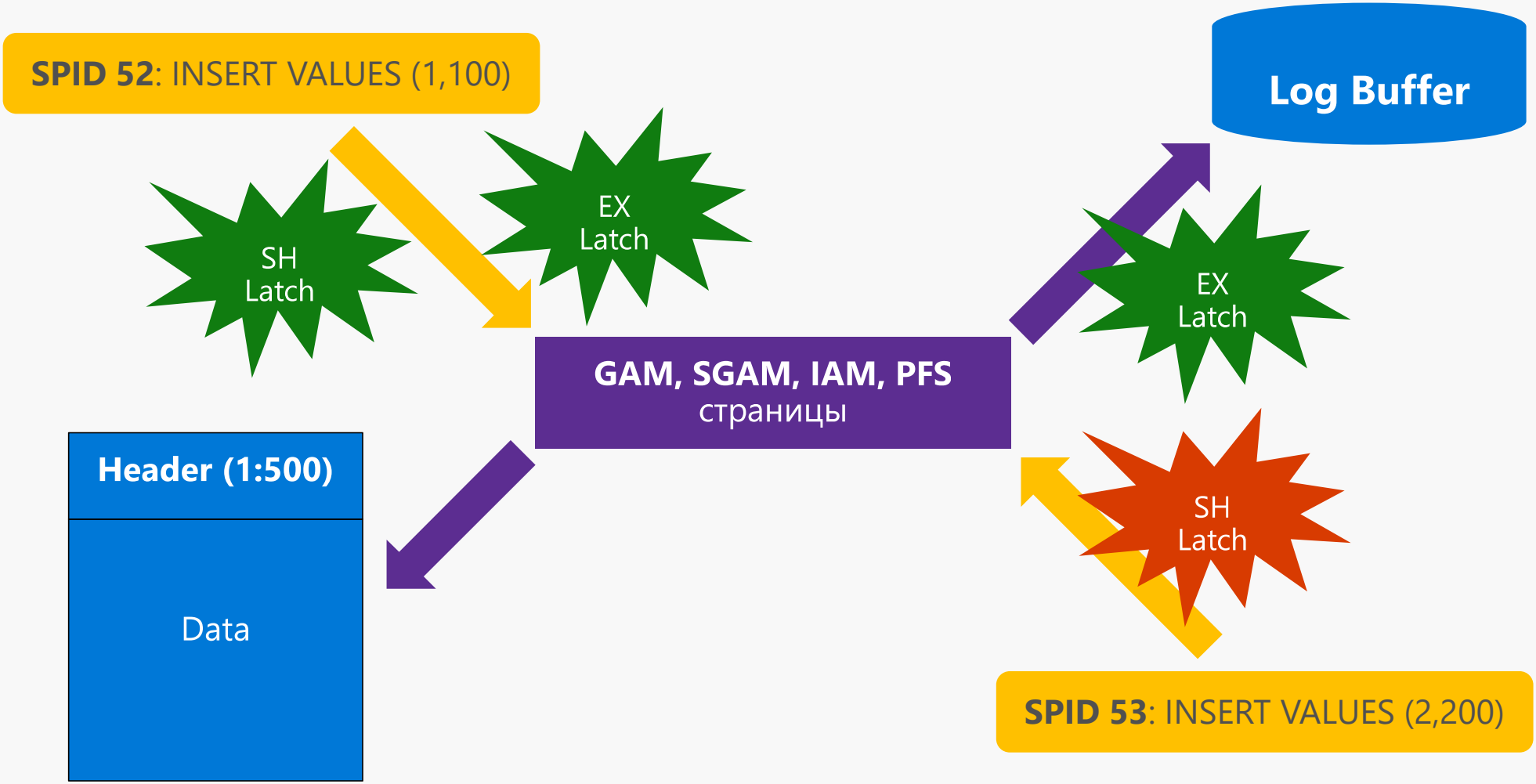
Locks

Transactional consistency

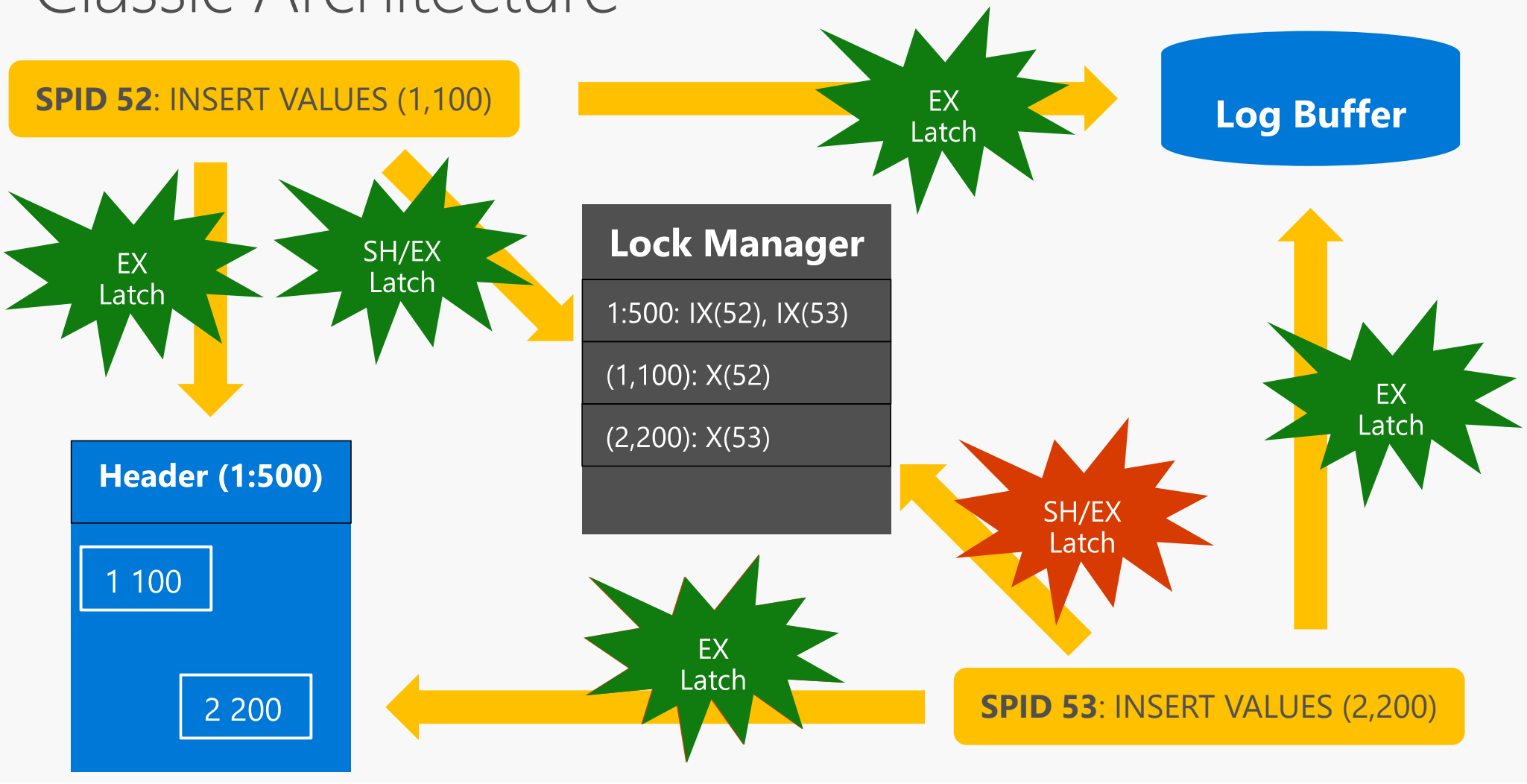
Latches

Physical consistency of in-memory data structures

“Classic Architecture”



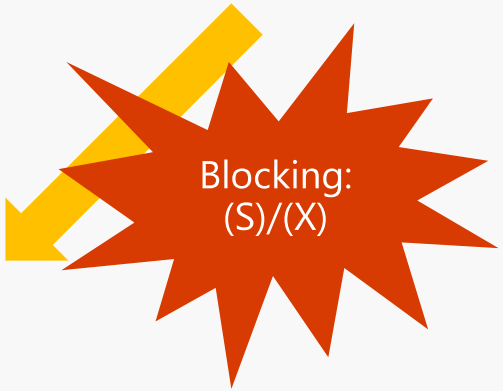
"Classic Architecture"



"Classic Architecture"

SPID 52: INSERT VALUES (1,100)

SPID 54: SELECT FROM T



Lock Manager

1:500: IX(52), IX(53), IS(54)

(1,100): X(52), **S(54)**

(2,200): X(53)

Header (1:500)

1 100

2 200

SPID 53: INSERT VALUES (2,200)

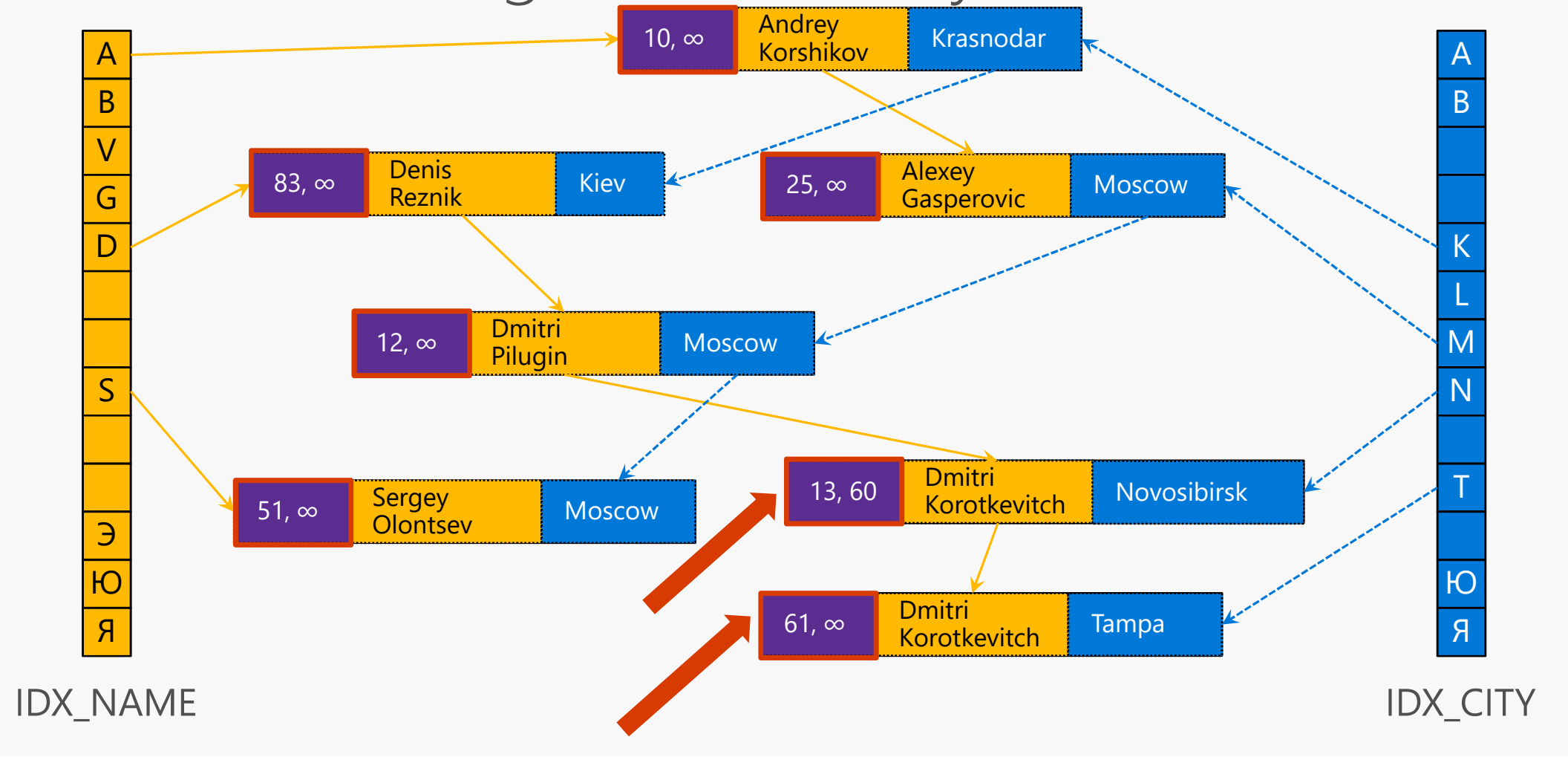
Demo

Latches

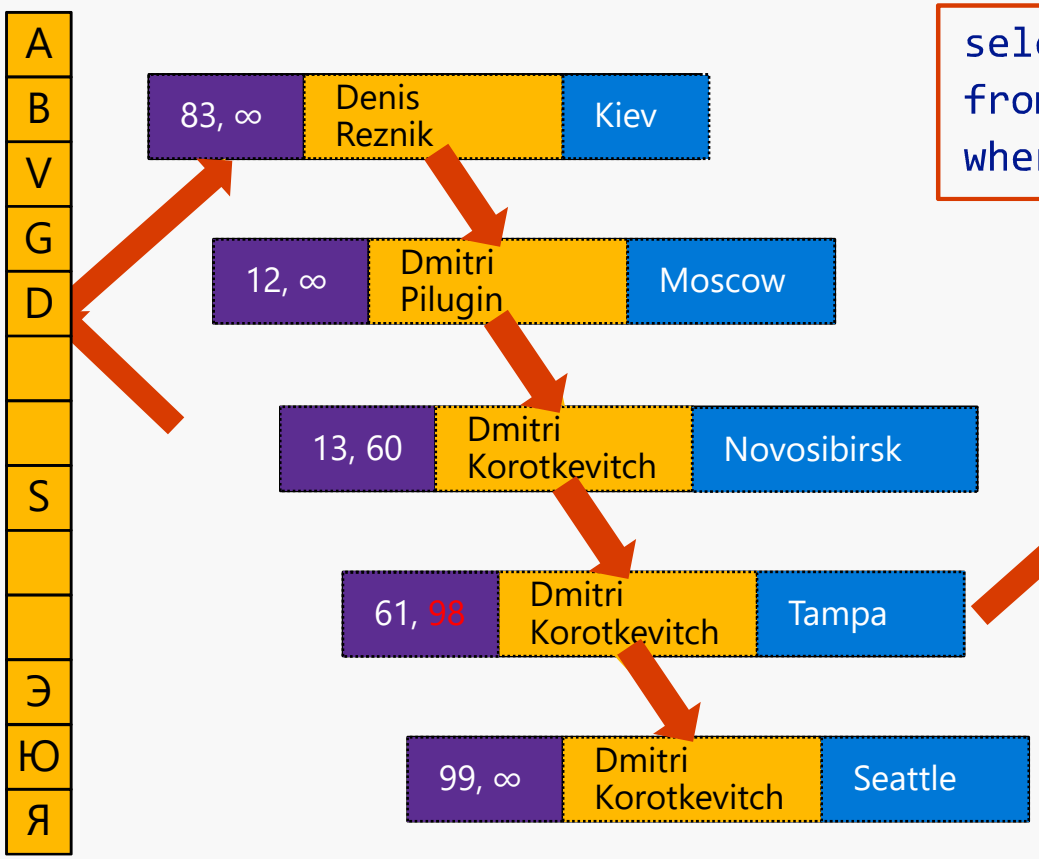
Data Processing in In-Memory OLTP

```
create table dbo.Community
(
      Name nvarchar(64) not null
            constraint IDX_NAME primary key nonclustered hash
            with (bucket_count=1024),
      City nvarchar(32) not null,
      FirstName nvarchar(64) not null,

      index IDX_CITY nonclustered hash(City)
      with (bucket_count=128),
)
with (memory_optimized=on, durability=schema_and_data);
```

Data Processing in In-Memory OLTP



Работа с Данными в In-Memory OLTP



IDX_NAME

Tx Time: 95

```
select Name, City
from dbo.Community
where Name = 'Dmitri Korotkevitch'
```

	Name	City
1	Дмитрий Короткевич	Тампа

Tx Time: 99

```
update dbo.Community
set City = 'Seattle'
where Name = 'Dmitri Korotkevitch'
```

Demo

Latch-Free Архитектура

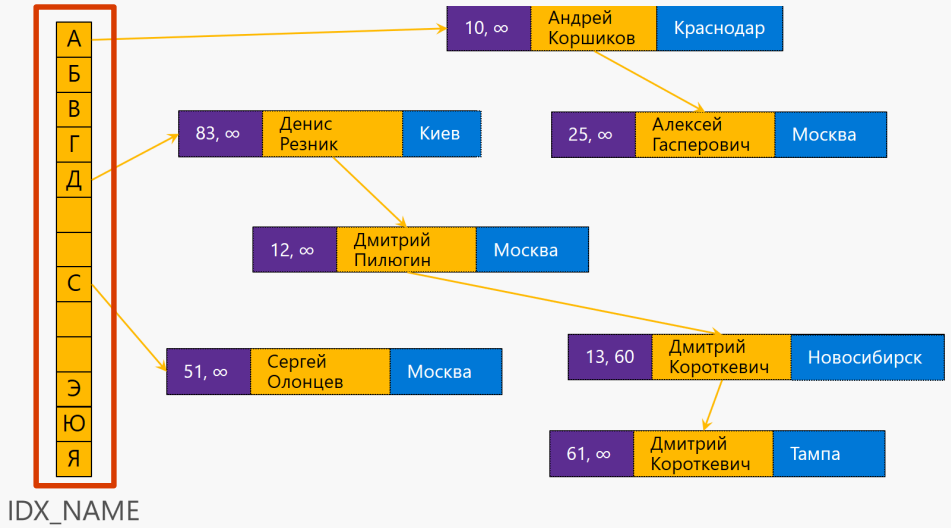
In-Memory OLTP Indexes- Hash

Optimized for point-lookup

where key1 = @param1 and key2 = @param2

Bucket_count sets the size of the hash table

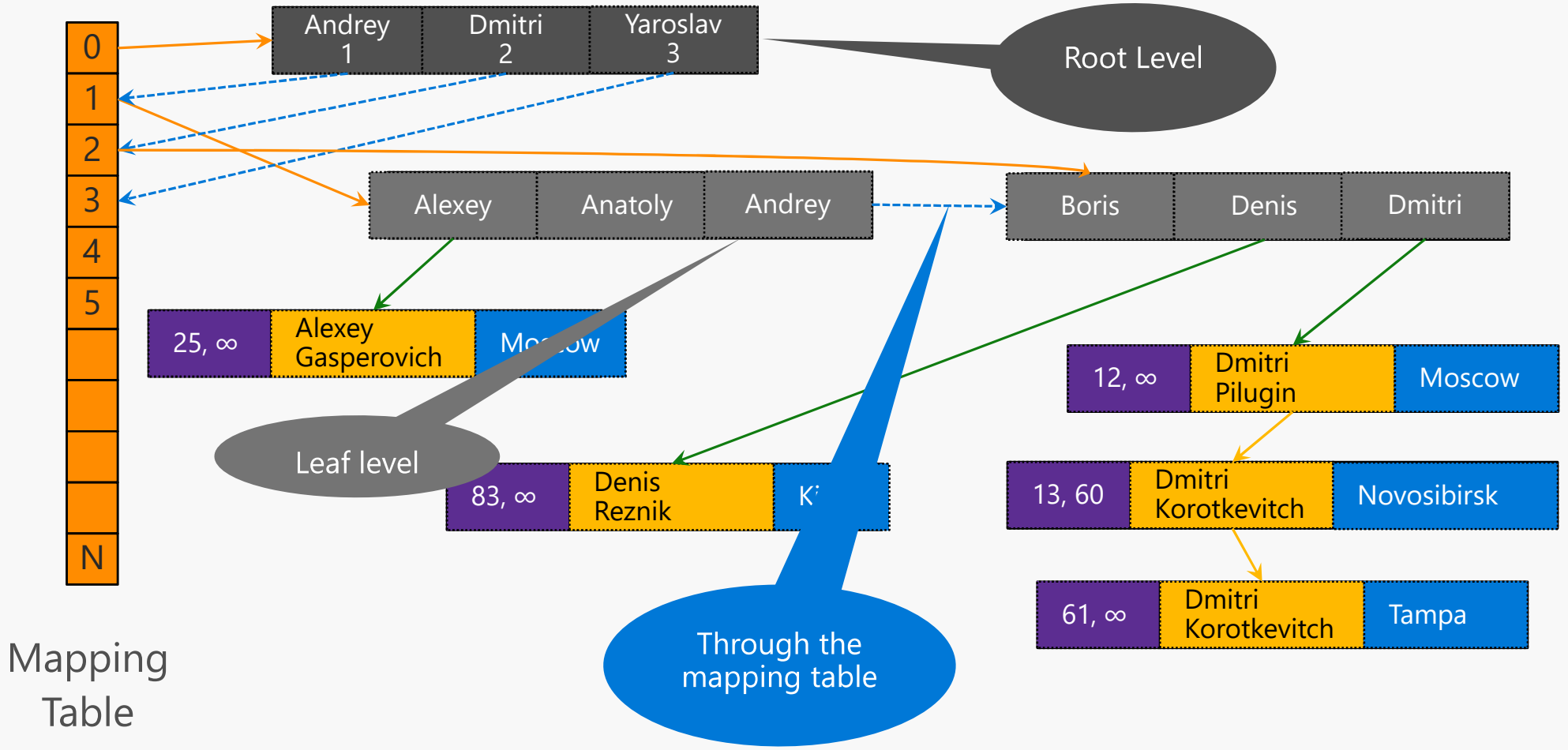
Should be about 1.5-2 times greater than index cardinality (# of unique keys in the index)



Nonclustered Indexes in In-Memory OLTP

```
alter table dbo.Community
    add index IdX_FIRSTNAME
    nonclustered(FirstName);
```

Некластерный Индекс в In-Memory OLTP

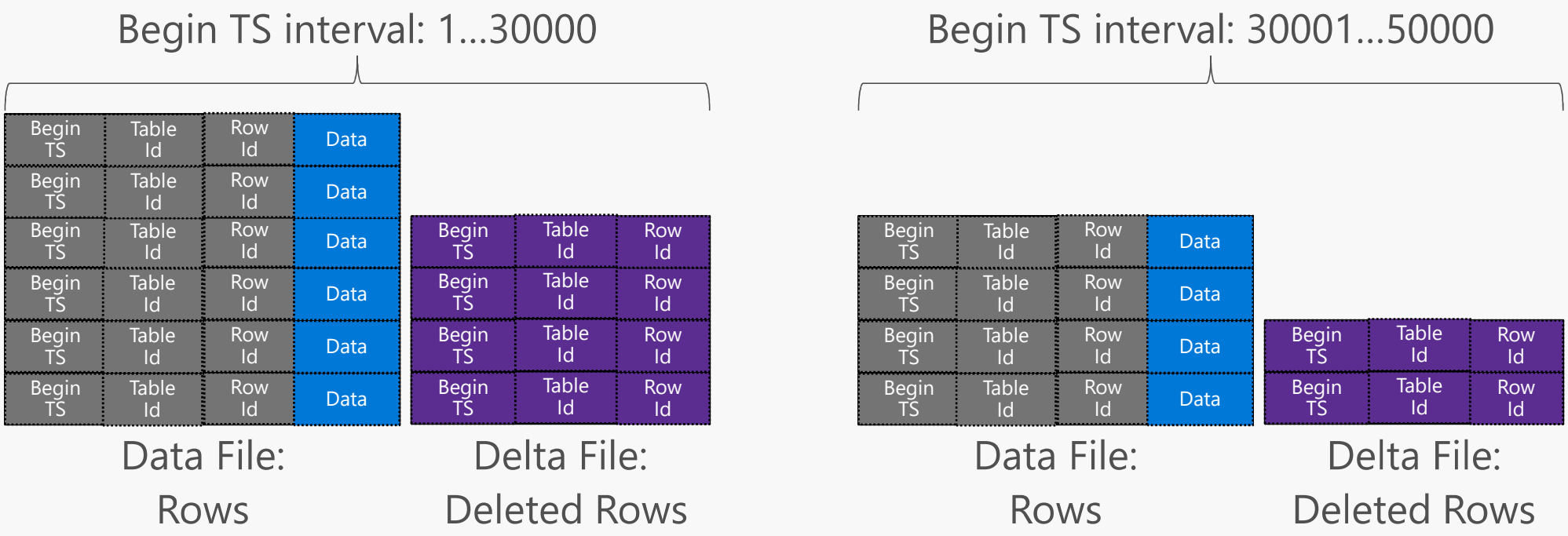


Data Storage

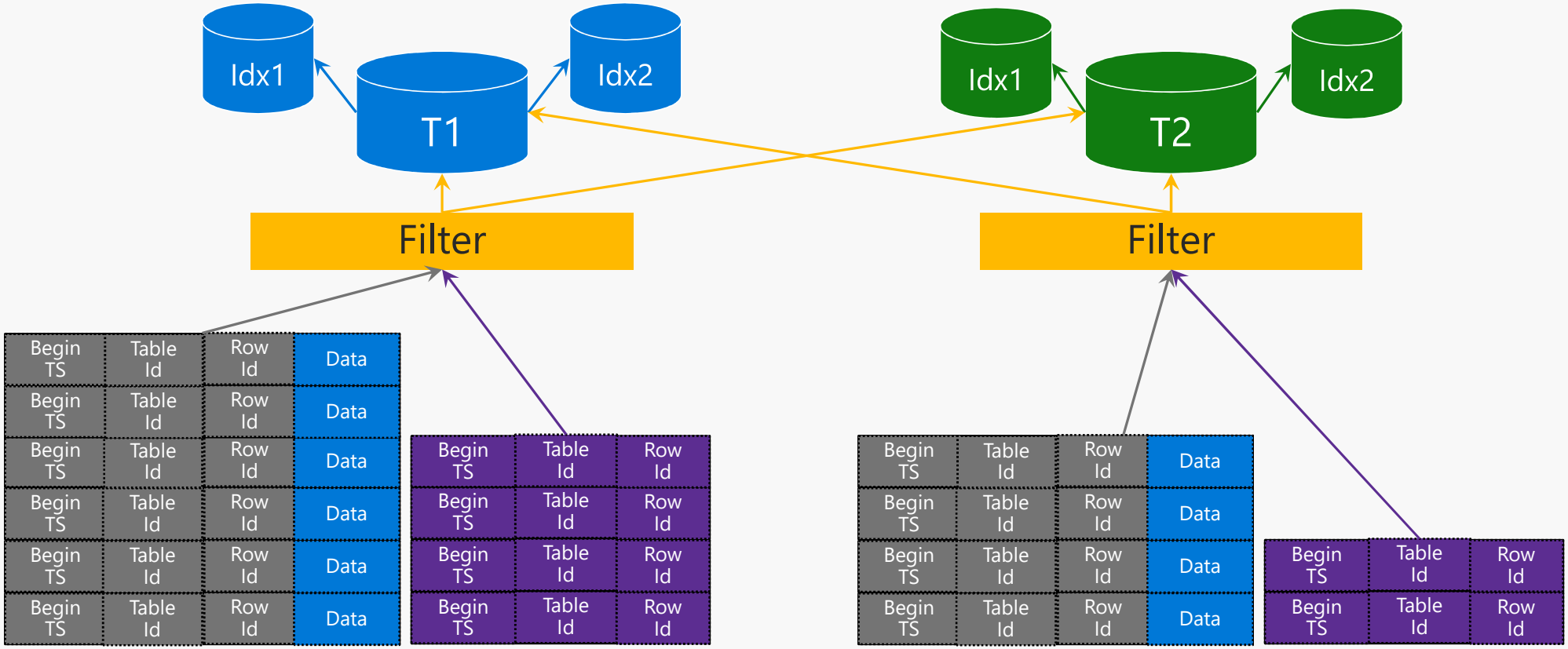
Data is stored in Checkpoint File Pairs (CFP)

Data file – stores row versions

Delta file – stores information about “deleted” rows



Reading Data into Memory



Transaction Logging

Logging is much more effective

- Log records are generated at COMMIT time (do not need UNDO portion)
- Every record covers multiple operations

CHECKPOINT threads constantly scan T-Log and populate Checkpoint File Pairs

SQL Server 2016 Enhancements

LOB Support (n)varchar(max) / varbinary(max)

Index Improvements

- BIN2 collation is no longer required
- Indexing NULL columns
- Unique indexes support
- Auto stats update

FK support

UNIQUE & CHECK support

Parallel plans (Interop table scans)

ALTER TABLE (offline)

Native Compilation

T-SQL code compiled into DLL

SQL Server loads DLL into SQL Server process memory

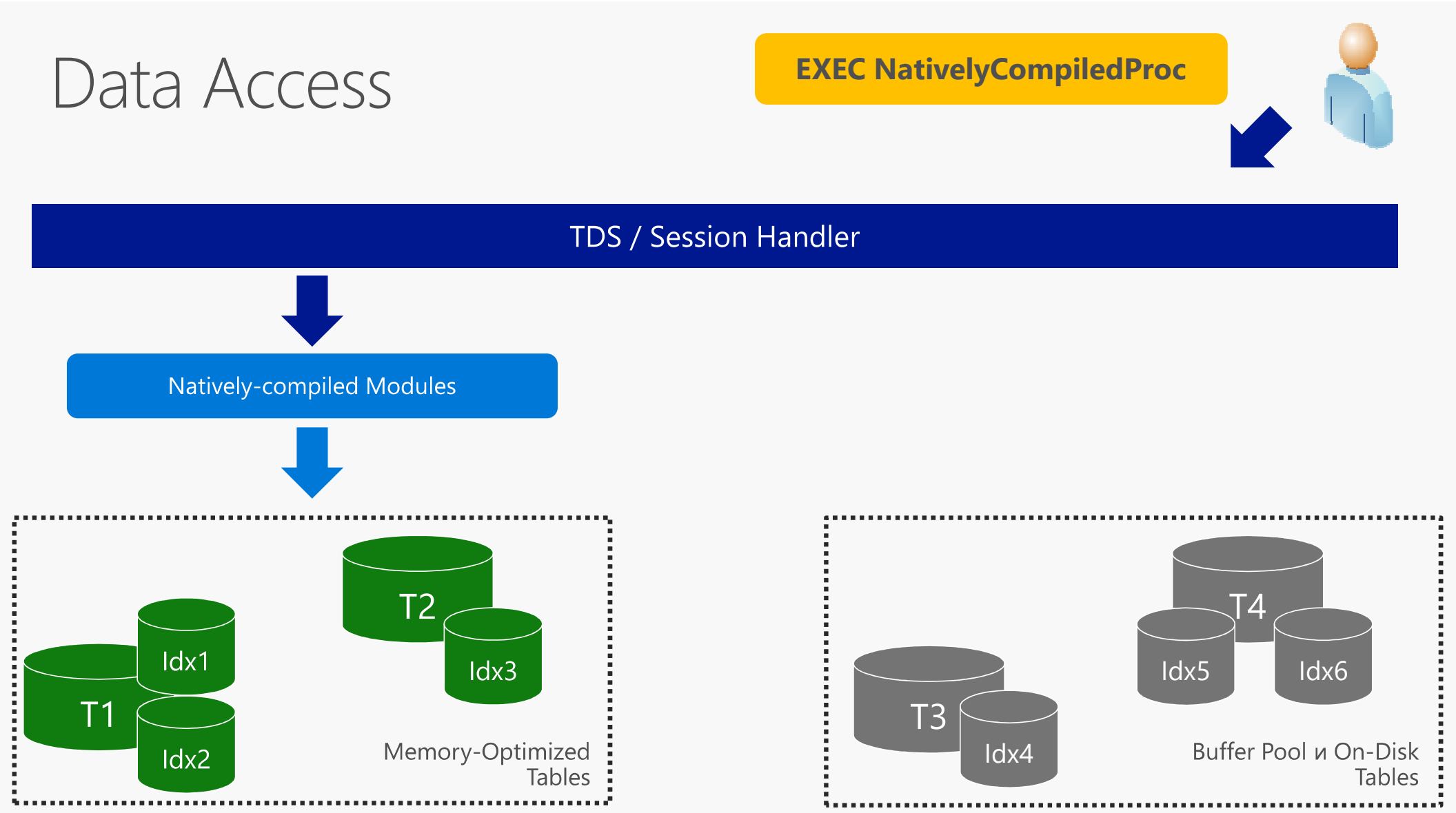
Optimization is done at compile time

10-40+ times performance increase

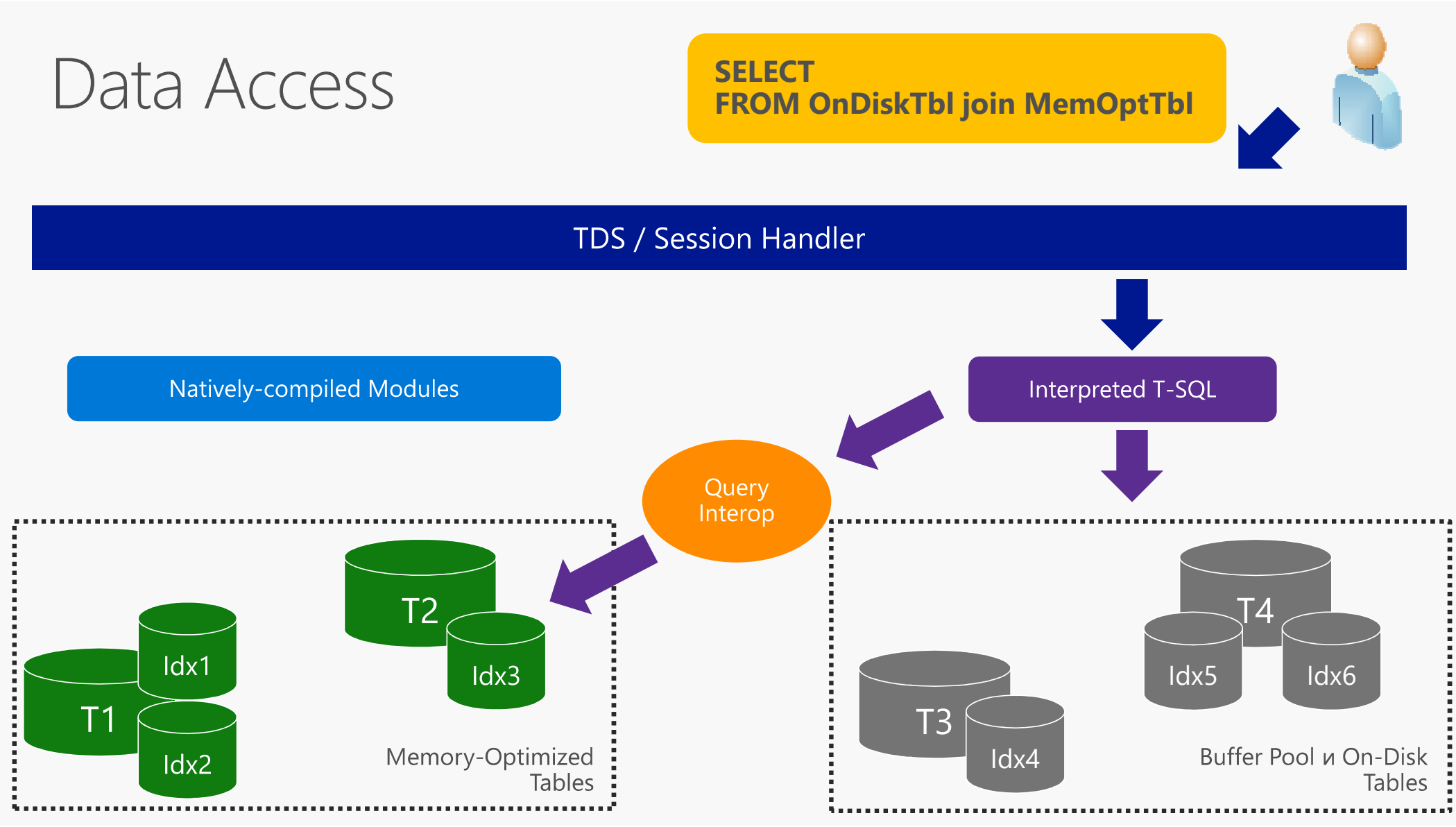
Demo

Native Compilation

Data Access


EXEC NativelyCompiledProc
TDS / Session Handler
Natively-compiled Modules
T1
Idx1
Idx2
T2
Idx3
Memory-Optimized Tables
T3
Idx4
T4
Idx5
Idx6
Buffer Pool и On-Disk Tables

Data Access


SELECT
FROM OnDiskTbl join MemOptTbl
TDS / Session Handler
Natively-compiled Modules
Interpreted T-SQL
Query
Interop
T1
Idx1
Idx2
T2
Idx3
Memory-Optimized
Tables
T3
Idx4
T4
Idx5
Idx6
Buffer Pool и On-Disk
Tables

SQL Server 2016 Enhancements

LOB support (n)varchar(max) / varbinary(max)

New natively-compiled modules:

Scalar functions, AFTER triggers, inline table-valued functions

Surface area improvements

OUTER JOIN
UNION, UNION ALL, SELECT DISTINCT
Subqueries (Exists, IN)
OR, NOT
OUTPUT in INSERT/UPDATE/DELETE statements

Nested calls in natively-compiled modules

Online ALTER

Additional Enhancements

2TB of durable data

Cross-features support

- Temporal Tables
- Query Store

TDE and RLS support

Optimizations

- Multi-threaded CHECKPOINT
- Multi-threaded RECOVERY
- CFP management improvements

In-Memory Columnstore Indexes

In-Memory OLTP and Mixed Workload

«Hot» data: current operational period

Data is highly volatile
OLTP workload

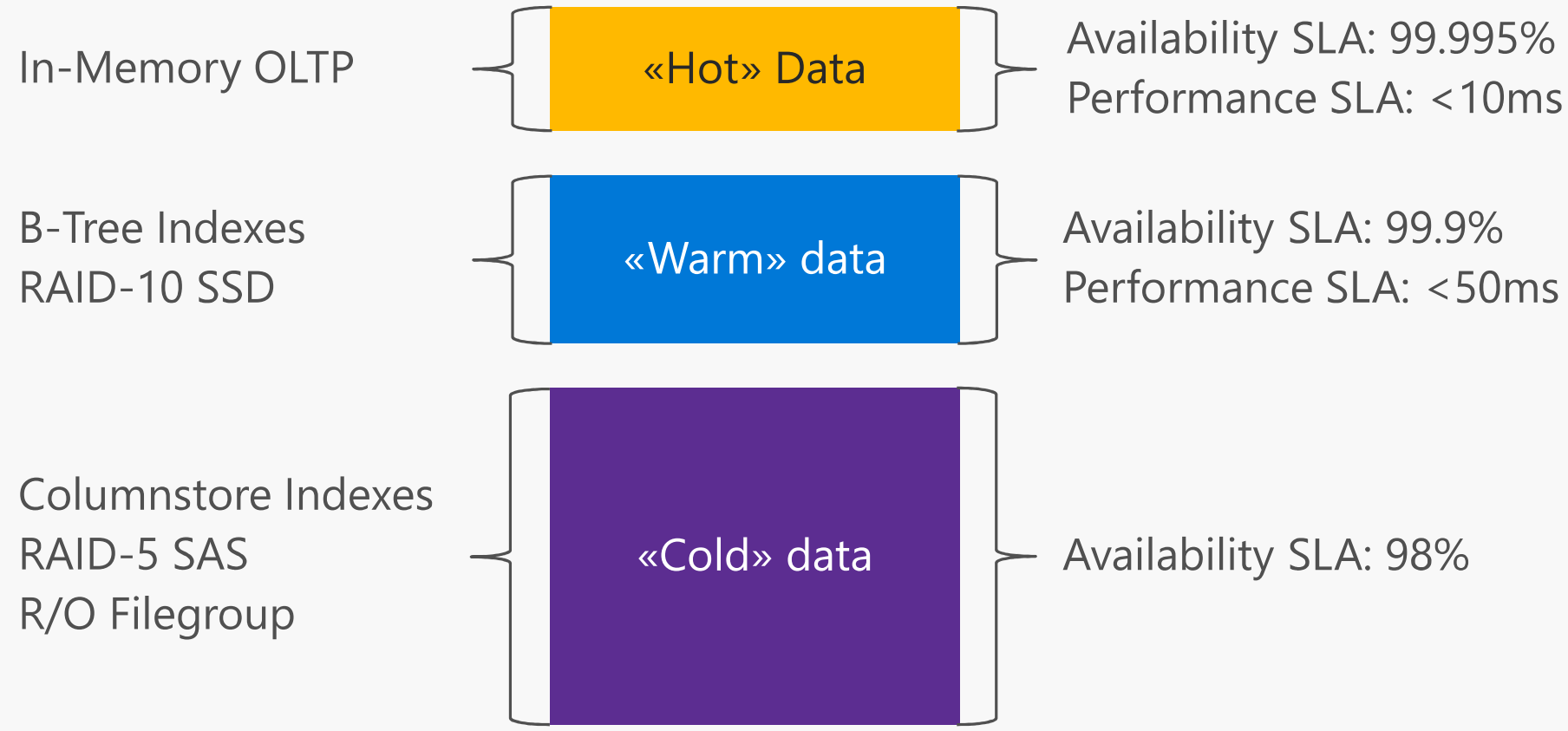
«Cold» data: old/archived data

Data is static, maybe read-only
DW workload (analysis, reporting)

«Warm» data: previous operational period

Data is static with some modifications
Mixed workload (OLTP + DW)

In-Memory OLTP and Mixed Workload

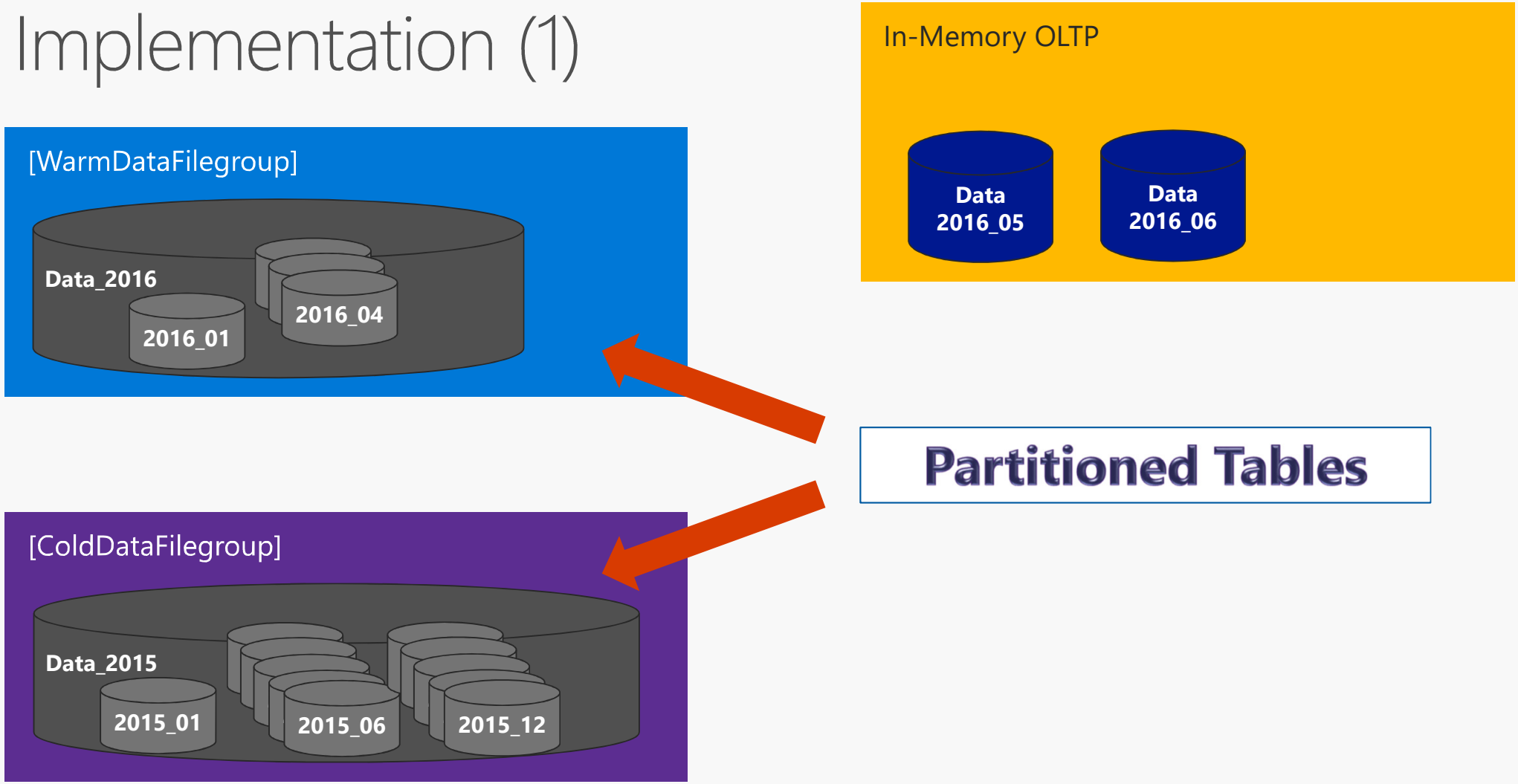


Data Partitioning

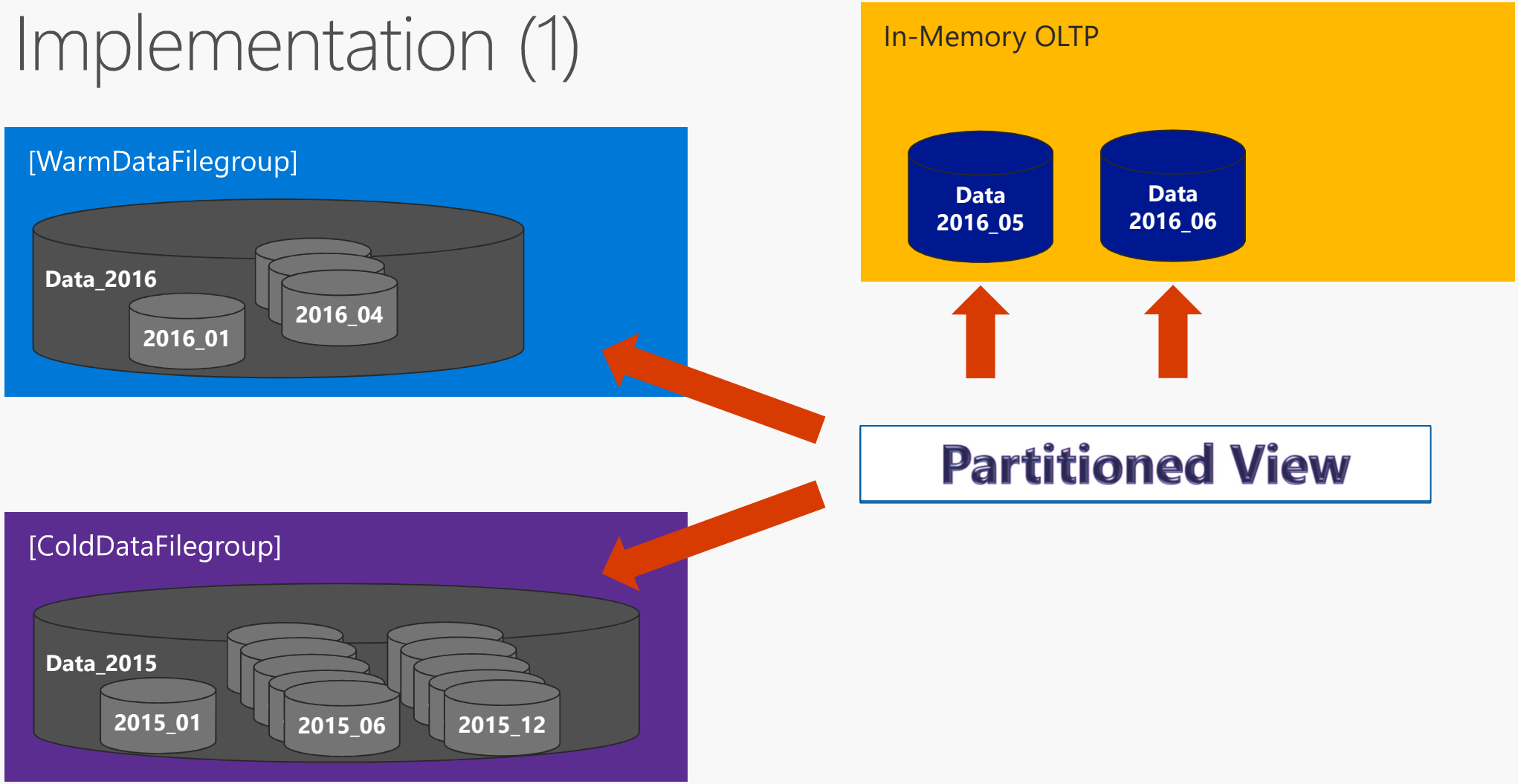
```
create table dbo.HotData
(
    TranDate datetime not null,
    TranId int not null primary key nonclustered hash,
    constraint CHK_HotData check (TranDate >= '2016-05-01')
)
with (memory_optimized=on);

create table dbo.WarmData
(
    TranDate datetime not null,
    TranId int not null,
    constraint CHK_WarmData
    check (TranDate >= '2016-01-01' and TranDate < '2016-05-01')
);
create unique clustered index IDX_WarmData
on dbo.WarmData(TranDate, TranId)
on psWarmData(TranDate);

create table dbo.ColdData
(
    TranDate datetime not null,
    TranId int not null,
    constraint CHK_WarmData check (TranDate < '2016-01-01')
);
create clustered columnstore index IDX_ColdData on dbo.ColdData
on psColdData(TranDate);
```

```
create view dbo.Data(TranDate, TranId)
as
    select TranDate, TranId
    from dbo.HotData
    where TranDate >= '2016-05-01'

    union all

    select TranDate, TranId
    from dbo.WarmData

    union all

    select TranDate, TranId
    from dbo.ColdData
```

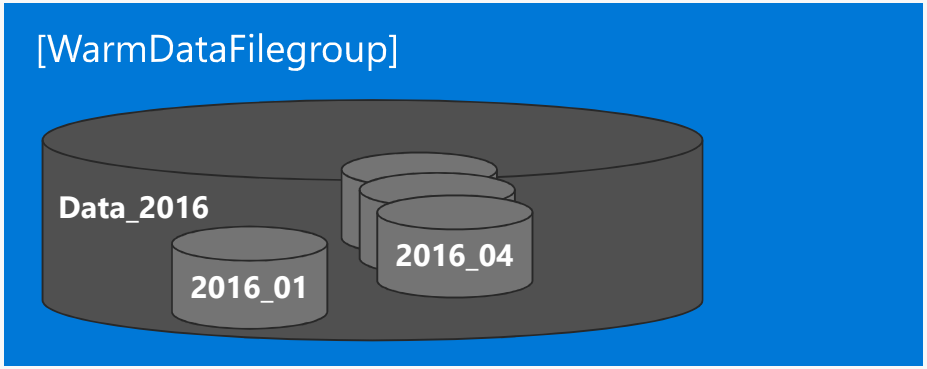
Implementation (1)

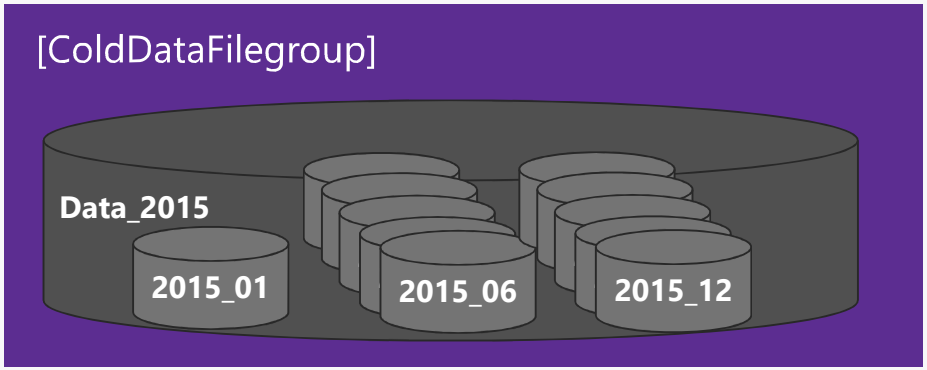


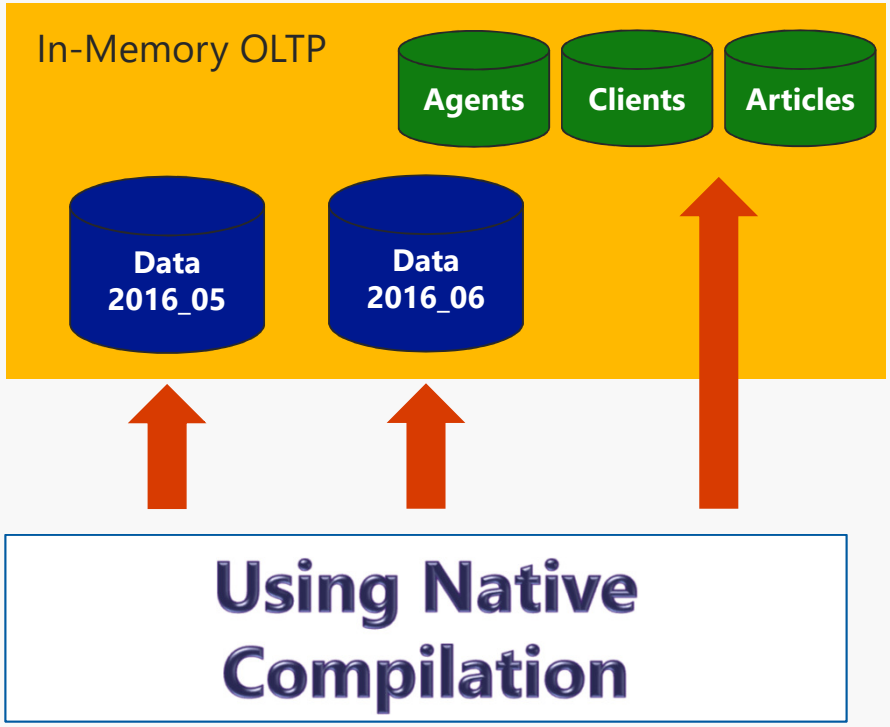
Implementation (1)



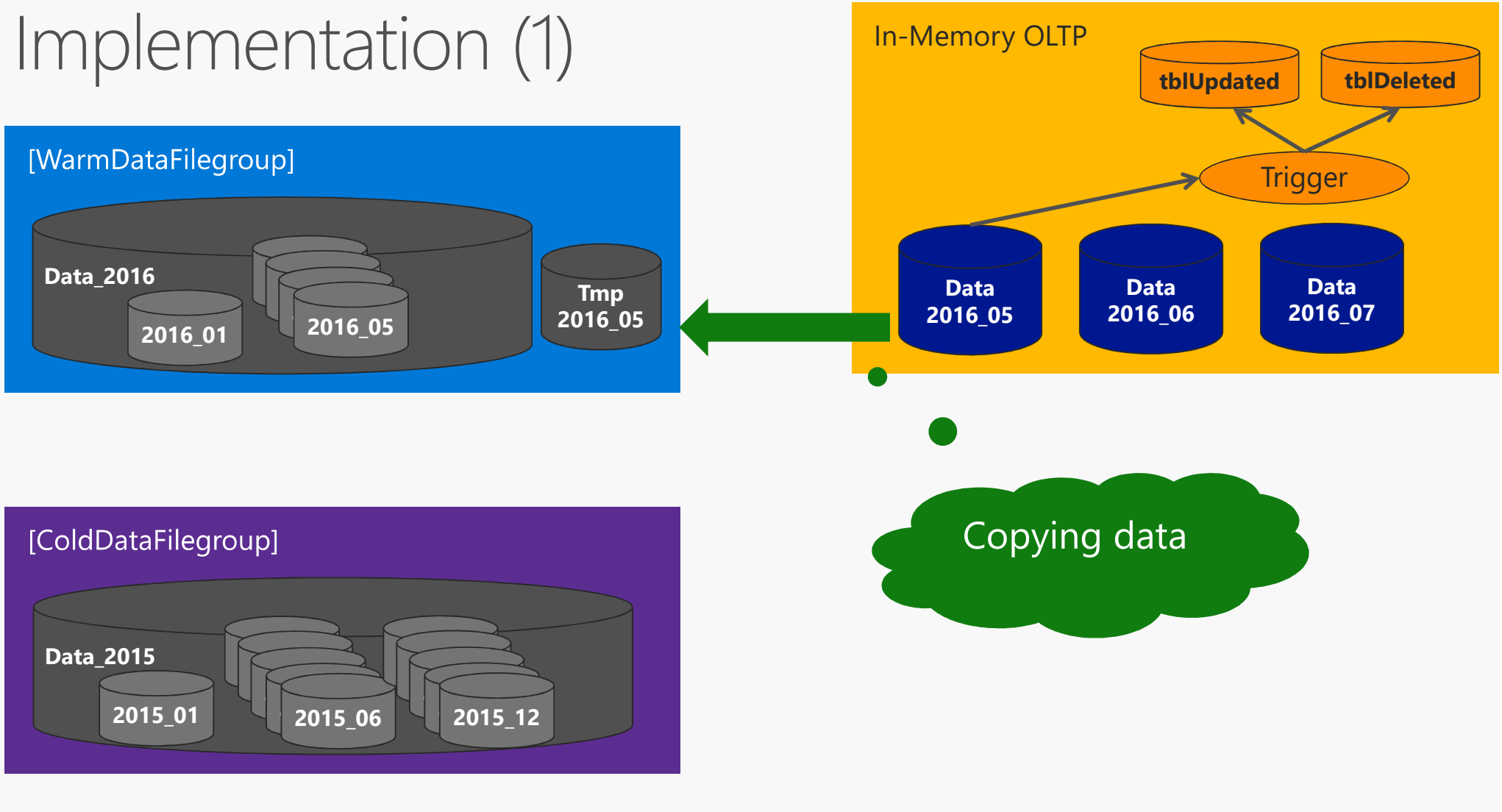
Implementation (1)



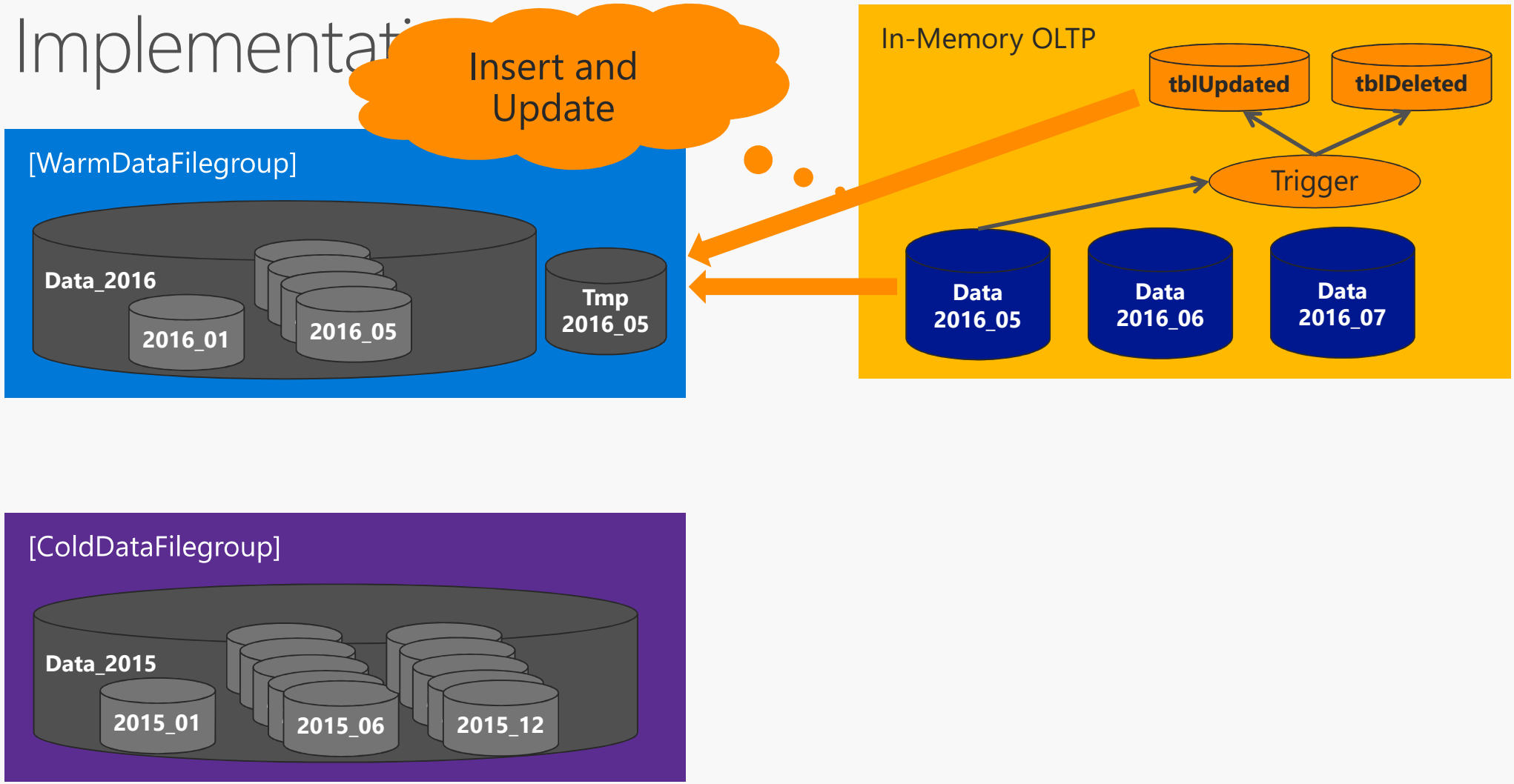




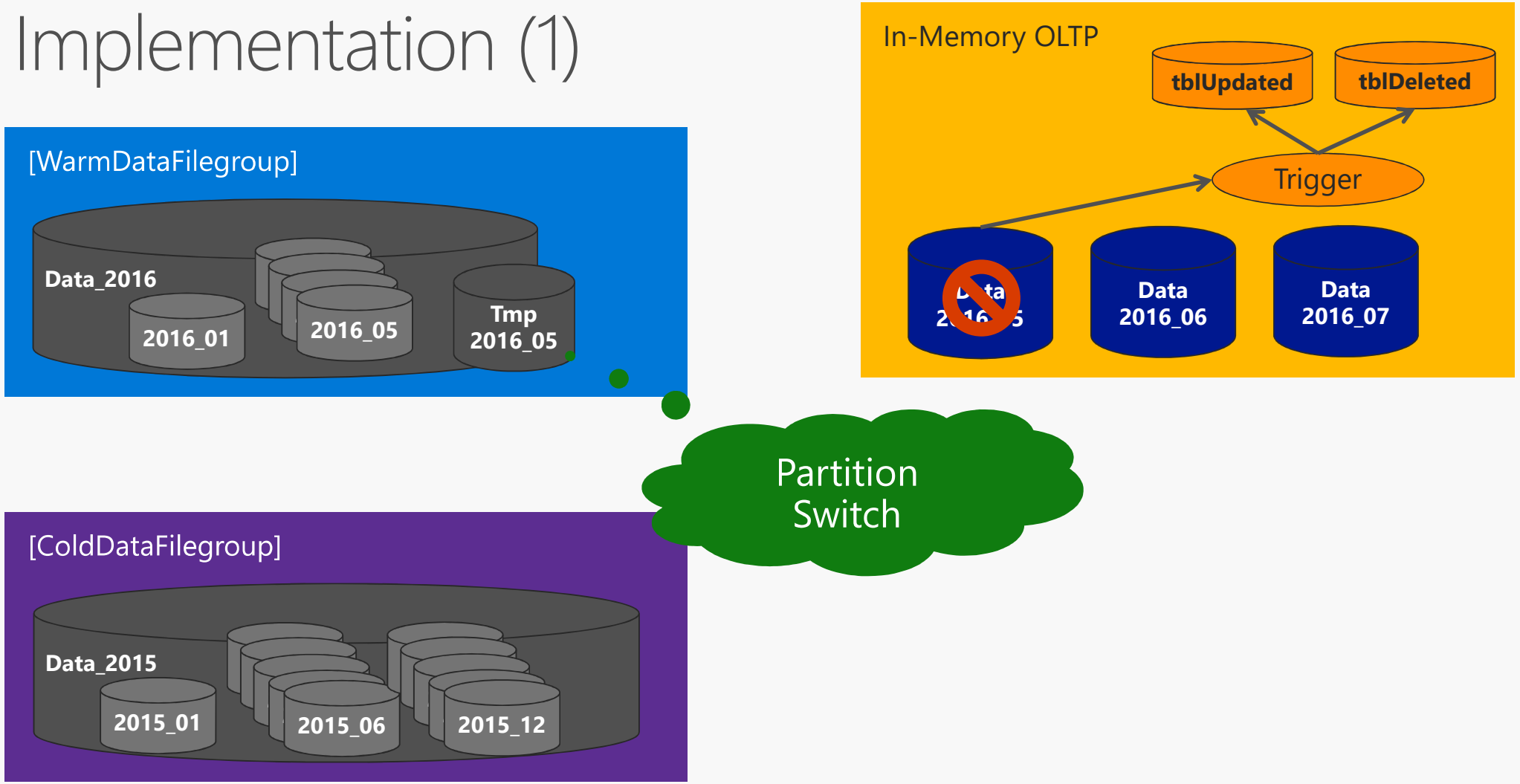
Implementation (1)



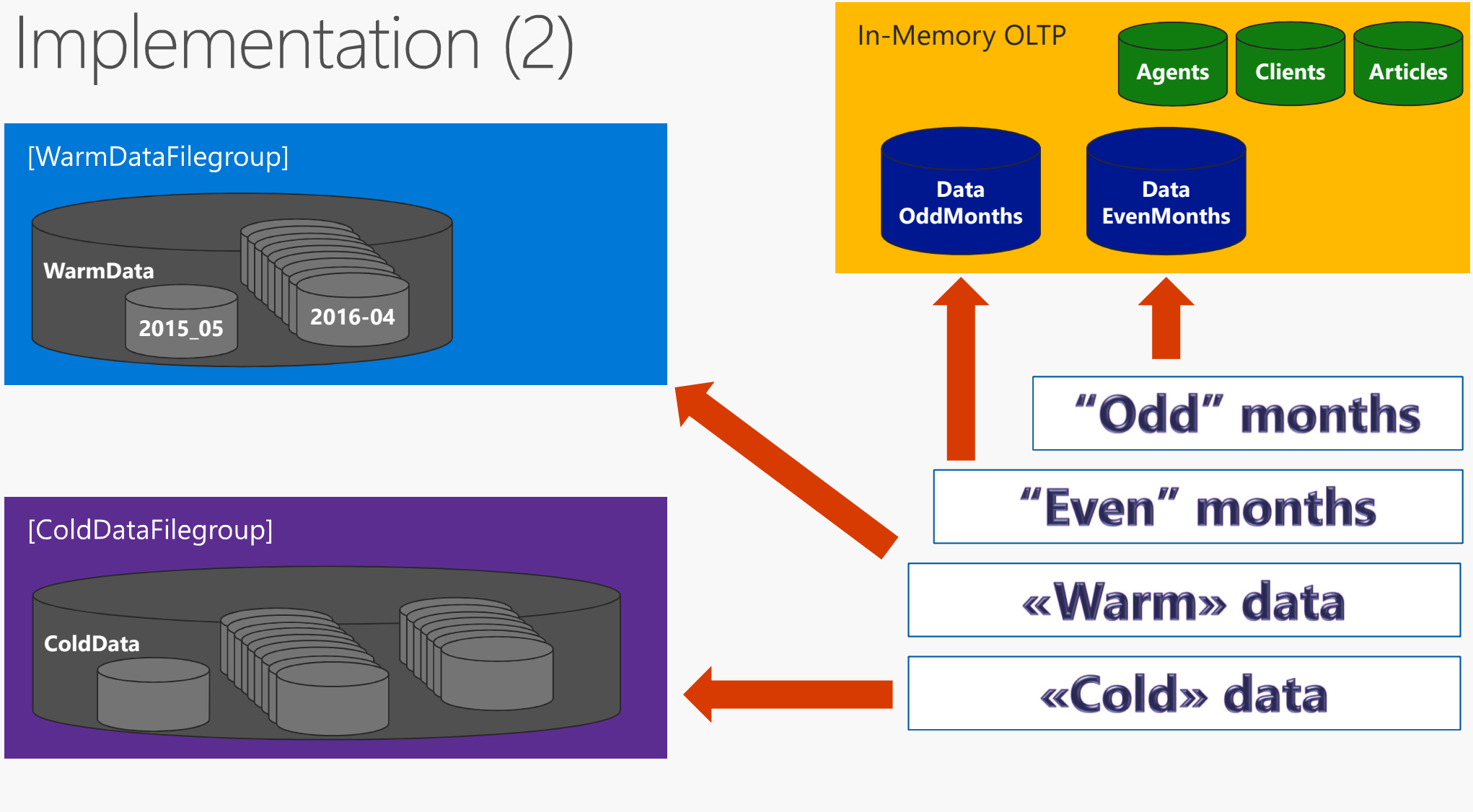
Implementati



Implementation (1)



Implementation (2)



Potential Issues

ALTER TABLE ADD CONSTRAINT CHECK scan index and hold Sch-M lock

Solution: Creating multiple CHECK constraints (January); (January..February); …

CREATE INDEX ON [NewFg] WITH (DROP_EXISTING=ON) does not move LOB data

Solution: CREATE INDEX ON NewPartitionScheme() WITH (DROP_EXISTING=ON)

Demo

In-Memory OLTP and mixed workload

We covered

Why In-Memory OLTP?

In-Memory OLTP Internal Architecture and Implementation

SQL Server 2016 Enhancements

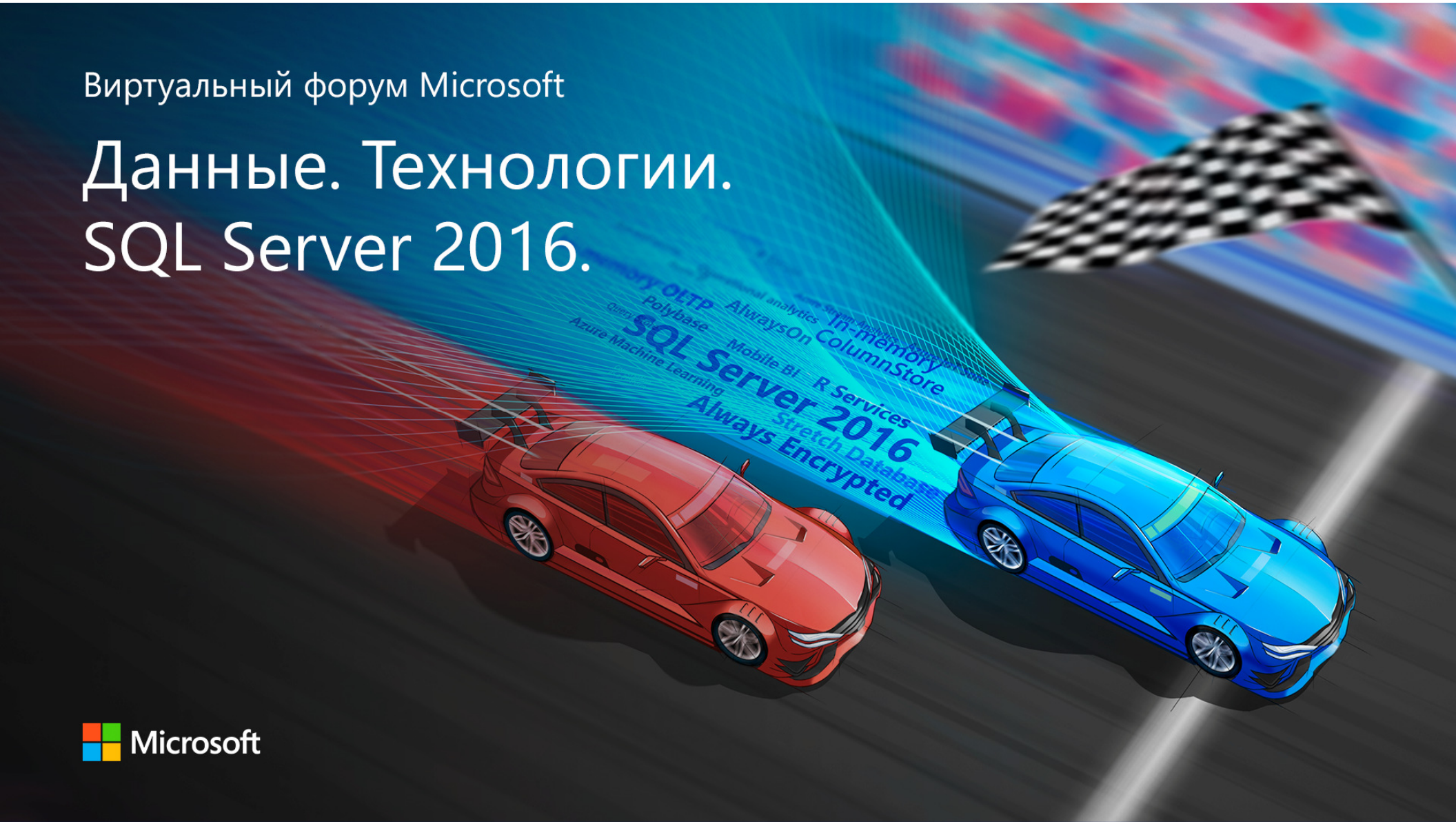
In-Memory OLTP in the systems with mixed workload

Demos: http://aboutsqlserver.com

Email: dk@aboutsqlserver.com

Виртуальный форум Microsoft

Данные. Технологии. SQL Server 2016.



Microsoft