Practical Partitioning

Rolf Tesmer

Data Solution Architect (DSA), Microsoft Australia

SQL Saturday #464, Melbourne

20th February 2016

Lego Box Analogy

Kids Lego Box Assessment, 2016

- Unorganised
- Too long to find pieces you need
- Gets messy again when played with

Me: But what if the kids want to build specific set?

Wife: Group it by colour

PARTITIONING

Wife: OK, then group it by set #

Me: *Ok done!* Uh, but how do the kids just find the Lego men?

SEARCHING

Wife: Mmmm, then they need to look at all the sets

Me: Aghhh! And if they play with 2 sets they'll all get mixed up again!

REORGANISE

Wife: Yeah, I guess you'll need to re-sort it again by the set #

Me: OK ok, maybe I'll just re-sort everything now by Lego shape?

REPARTITIONING

Wife: Well it sucks to be you!

Me: Hey yeah, it does! I think I'll just do nothing, its working anyway!

Housekeeping

Mobile Phones

please set to “stun” during sessions

Evaluations

complete online to be in the draw for fantastic prizes

SESSIONS	EVENT
http://www.sqlsaturday.com/464/sessions/sessionevaluation.aspx	http://www.sqlsaturday.com/464/eventeval.aspx

Wifi Details

SSID: eduroam // **Login**: ext-sqlsat // **Password**: sqlsaturd4y

Connect with the Community

speaker

PASS SQL saturday

#464 | MELBOURNE 2016

Mr. Fox SQL

@MrFoxSQL

Outfoxing SQL challenges by being rather damn sneaky...

Event staff, volunteers and speakers are here to help and answer questions.

Scan the QR code on the speaker badges to connect and network with them.

About me…

- **Solution Architect @ Microsoft** – Azure Data Services
- About **20 years** experience in all things SQL & Data
- Personal interest in **Azure PaaS**, **SQL spatial, big data / data lake, data warehousing (DW)**, **IoT, machine learning (ML)**

- **Am on LinkedIn *here*:** https://www.linkedin.com/in/**rolftesmer**
- **Plus I also blog *here*:** https://**mrfoxsql**.wordpress.com/

MY assumptions about **YOU**!

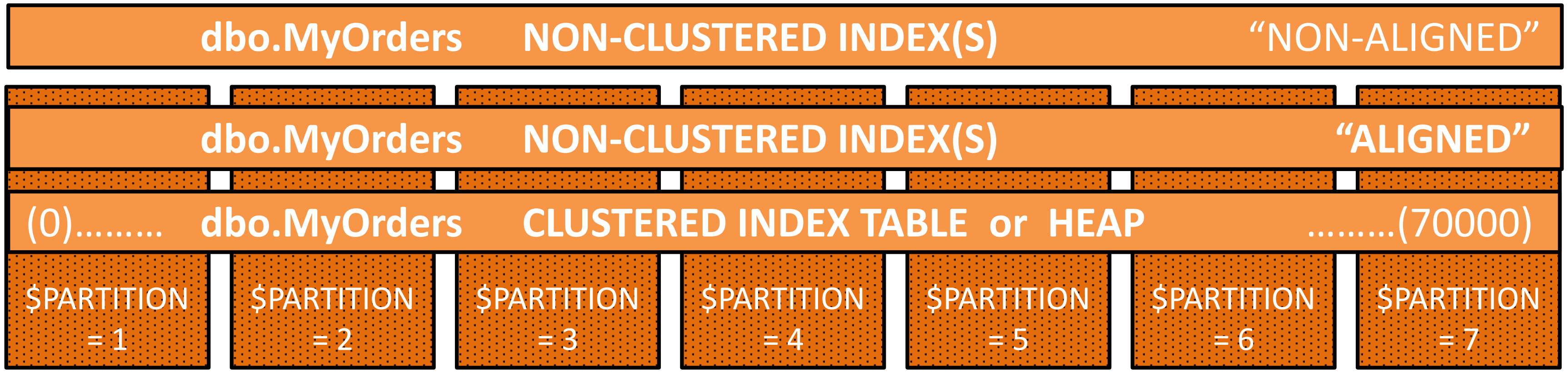
- Solid in **SQL Server**
- Solid in **Databases**
- Awareness of **Partitioning**

Agenda

- **Introduction** - What is Partitioning
- Partitioning **101** – Understand the Basics
- Partitioning **201** – Beyond the Basics
- Improving **Query** Performance
- Streamlining **Maintenance** Operations
- **Best Practices**
- **Questions & Answers** (Q&A)

Introduction – **What is Partitioning**

- **Physical horizontal** segregation of **table** and/or **index** into **multiple identical data structures**
- **Transparent** to applications
- Managed at **database level** by **SQL DBA / Developer**
- Mainly for VLDB **performance** and/or **maintenance**


dbo.MyOrders NON-CLUSTERED INDEX(S) "NON-ALIGNED"
dbo.MyOrders NON-CLUSTERED INDEX(S) "ALIGNED"
(0)......... dbo.MyOrders CLUSTERED INDEX TABLE or HEAP ....(70000)
$PARTITION = 1
$PARTITION = 2
$PARTITION = 3
$PARTITION = 4
$PARTITION = 5
$PARTITION = 6
$PARTITION = 7

Why Should I Use It

- **Why…**
  - Have **ownership** of **database schema**
  - Want **greater control** over **subsections** of data
  - Have **significant** amount of data *
  - Need to **increase** query **performance** *
  - Need to **reduce impact** of database **maintenance**
  - Want additional **database backup** / **recovery** options
  - Want to **archive data** using **sliding window** (SWITCH) *

Scalability!   Performance!   Availability!

Where Can I Use It

- **Where…**
  - On-Prem **SQL <u>Enterprise</u> Edition** (SQL 2005+)
  - On-Prem **APS** (aka **PDW**) (SQL MPP Appliance)
  - Azure (or 3rd party) IaaS **SQL <u>Enterprise</u> Edition** (SQL 2005+)
  - Azure **SQL Database** (v12 Basic/Standard/Premium) - PaaS
  - Azure **SQL Data Warehouse** (*currently in preview…*) - PaaS

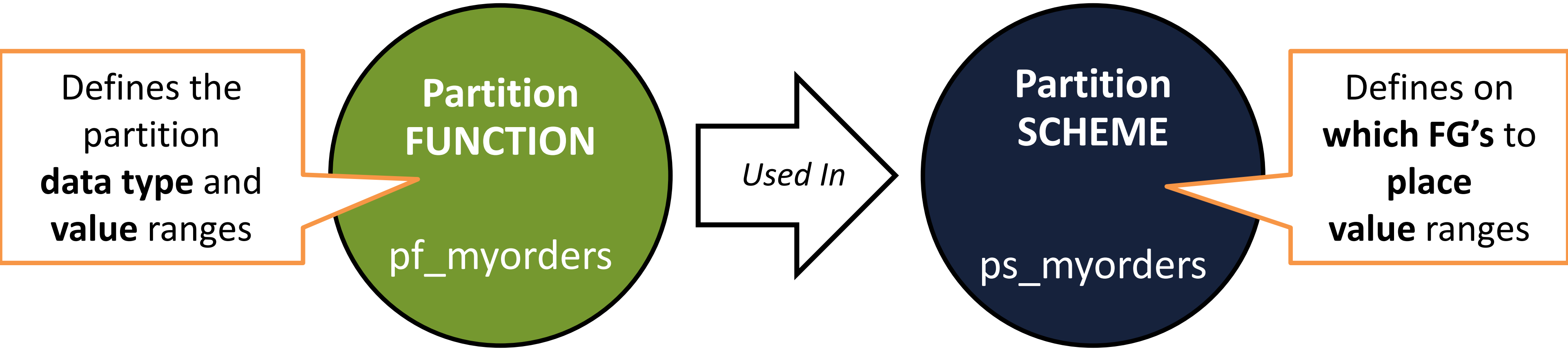
All Flavours of SQL Server!

Agenda

- **Introduction** - What is Partitioning
- Partitioning **101** – Understand the Basics
- Partitioning **201** – Beyond the Basics
- Improving **Query** Performance
- Streamlining **Maintenance** Operations
- **Best Practices**
- **Questions & Answers** (Q&A)

Partitioning 101 – Understand the Basics

- **Partitioning** requires **function** and **scheme**
  - Can **ONLY** be **single column** (**no composites**!)
  - Column **must** be part of **index** (can be **any** position)
  - Best performance if column is **primary method** of **access**
  - Lowest impacts if column value **does not change**


Defines the partition **data type** and **value** ranges
Partition FUNCTION
pf_myorders
Used In
Partition SCHEME
ps_myorders
Defines on **which FG's** to **place** **value** ranges

Partitioning 101 – Understand the Basics

INT datatype
4294967295

- **Partition Function**

Partition Function
pf_myorders

```
CREATE PARTITION FUNCTION pf_myorders (int)
AS RANGE RIGHT
FOR VALUES (0, 10000, 20000, 30000)
```

Better for Dates
No dealing with "…23:59:59.997"

```
AS RANGE RIGHT
```

null → (min..-1) (0..9999) (10000..19999) (20000..29999) (30000..max)

Don't run out of partition space!

```
AS RANGE LEFT
```

null → (min..0) (1..10000) (10001..20000) (20001..30000) (30001..max)

Partitioning 101 – Understand the Basics

- Partitions can be **different sizes / ranges**

```
CREATE PARTITION FUNCTION pf_myorders (int)
AS RANGE RIGHT
FOR VALUES (10, 1300, 1950, 201000)
```

INT datatype
4294967295

```
 (min..9) (10..1299) (1300..1949) (1950..200999) (201000..max)
```

2147483656

1289

649

199049

2147282647

Partitioning 101 – Understand the Basics

- **Partition Scheme**

Partition Scheme
ps_myorders

```
CREATE PARTITION SCHEME ps_myorders
AS PARTITION pf_myorders
TO ([PreFG], [FG_1], [FG_2], [FG_1], [FG_3])
RANGE  (min..-1)     (0..9999)  (10000..19999)(20000..29999) (30000..max)
```

Must match number of ranges in FUNCTION

…or can be positioned onto a single filegroup

```
ALL TO ([PRIMARY])
```

**** Note that Azure SQL DB and Azure SQL DW do not have filegroups!***

Partitioning 101 – Understand the Basics

- Can apply to **new** or **existing** tables / indexes

EXISTING

```
CREATE UNIQUE CLUSTERED INDEX
     pk_myorderid
     ON dbo.myorders (myorder_id)
     WITH (DROP_EXISTING = ON)
     ON ps_myorders(myorder_id)
GO
```

DROP_EXISTING = replace what's there

Will rebuild existing NC indexes *
(*but wont partition them!*)

NEW

```
CREATE TABLE dbo.myorders
(
      myorder_id          INT NOT NULL
    , CONSTRAINT pk_myorderid
      PRIMARY KEY CLUSTERED
      (myorder_id ASC)
      ON ps_myorders(myorder_id)
)
GO
```

Builds partitioned structure up front

Any NC indexes added later are partitioned

DEMONSTRATION 01

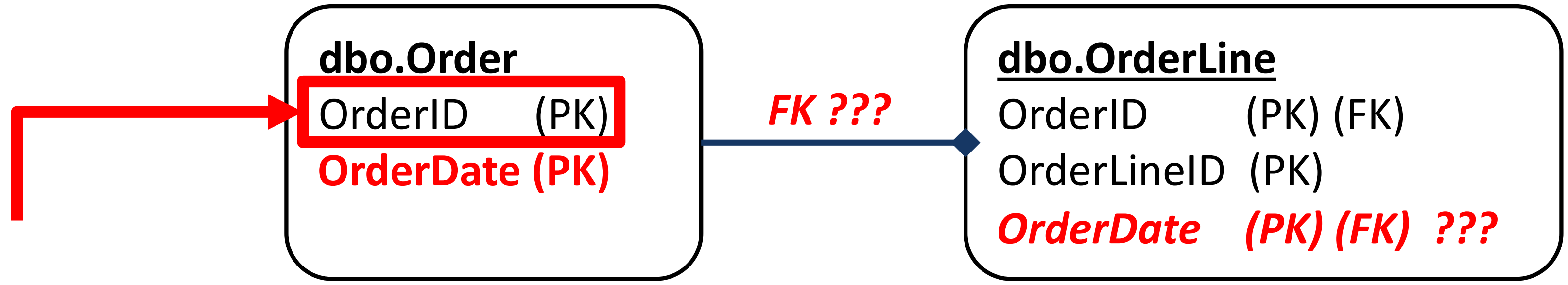
Creating partitioned tables & indexes

Agenda

- **Introduction** - What is Partitioning
- Partitioning **101** – Understand the Basics
- Partitioning **201** – Beyond the Basics
- Improving **Query** Performance
- Streamlining **Maintenance** Operations
- **Best Practices**
- **Questions & Answers** (Q&A)

Partitioning 201 – Beyond the Basics

- Can **impact logic** in **PK's, UK's** & **FK's**
  - Partition column **must** be part of **index keys**
  - Affects "**uniqueness**" of index, affects **related FK's**
  - Need to **enforce** "uniqueness" by **trigger** (*adds IO cost*)



t_iu_Order
(Insert/Update Trigger)
Check OrderID duplicates
IF EXISTS () ROLLBACK

OrderID	OrderDate
10	Feb2016
~~10~~	~~Mar2016~~

Partitioning 201 – Beyond the Basics

- Can **run out** of partitions = **data bundling**
  - Can **rebuild/change/extend** partitions – *but its **costly***!

```
...AS RANGE RIGHT FOR VALUES(10000, 20000, 30000)

RANGE: (min..9999) (10000..19999) (20000..29999) (30000..max)
ROWS:     (10000)        (10000)        (10000)       (90000)

ALTER PARTITION FUNCTION pf_myorders () (30000..39999) (40000..max)
SPLIT RANGE (40000);                       (10000)        (80000)
```

- Options…
  1. **SPLIT** last partition into **2** partitions (*deletes & reinserts*)
  2. Rebuild **clustered index** onto **new** partition function/scheme

DEMONSTRATION 02

Altering Partitioned Tables

Agenda

- **Introduction** - What is Partitioning
- Partitioning **101** – Understand the Basics
- Partitioning **201** – Beyond the Basics
- Improving **Query** Performance
- Streamlining **Maintenance** Operations
- **Best Practices**
- **Questions & Answers** (Q&A)

Partitioning for Performance

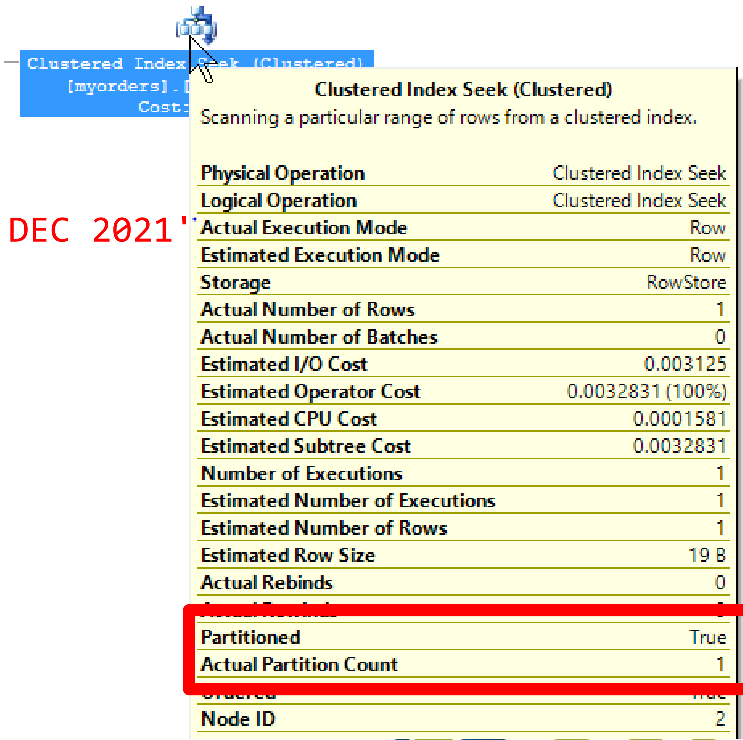
- How it can **help** your **queries**…
  - **Partition elimination** (as long as using partition key!)

```
SELECT  COUNT(*) as Records,
        MAX(myorder_id) as PrimaryOrder,
        SUM(myorder_amt) as OrderAmount
FROM    dbo.myorders
WHERE   myorder_date BETWEEN '1 JAN 2016' AND '31 DEC 2021'
AND     mysupplier_id = 55
```

1 row

465M rows

Partitioned by
mysupplier_id



Clustered Index Seek (Clustered)
[myorders].[
Cost:

Clustered Index Seek (Clustered)

Scanning a particular range of rows from a clustered index.

Physical Operation	Clustered Index Seek
Logical Operation	Clustered Index Seek
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Actual Number of Rows	1
Actual Number of Batches	0
Estimated I/O Cost	0.003125
Estimated Operator Cost	0.0032831 (100%)
Estimated CPU Cost	0.0001581
Estimated Subtree Cost	0.0032831
Number of Executions	1
Estimated Number of Executions	1
Estimated Number of Rows	1
Estimated Row Size	19 B
Actual Rebinds	0
Actual Rewinds	0
Partitioned	True
Actual Partition Count	1
Ordered	True
Node ID	2

Partitioning for Performance

- How it can **punish** your **queries**…
  - Not all queries benefit, some can go **dramatically** backwards
    - Can depend if your index is **aligned** or **not aligned**
  - In general **aligned indexes** are marginally **slower to access**
    - Sometimes **all** partitioned index **b-trees** accessed to **find key**
  - **SORT / TOP** queries can be **devastatingly** slow
    - When **Search key** not partitioned as **leading segment**

DEMONSTRATION 03

Partitioned Table Performance

Agenda

- **Introduction** - What is Partitioning
- Partitioning **101** – Understand the Basics
- Partitioning **201** – Beyond the Basics
- Improving **Query** Performance
- Streamlining **Maintenance** Operations
- **Best Practices**
- **Questions & Answers** (Q&A)

Partitioning for Maintenance

- Partition Index Optimisations
  - **Capture** fragmentation & **REBUILD** at **object partition level**
  - **SQL 2014** can **rebuild** partition **ONLINE** (*prior SQL = **OFFLINE***)

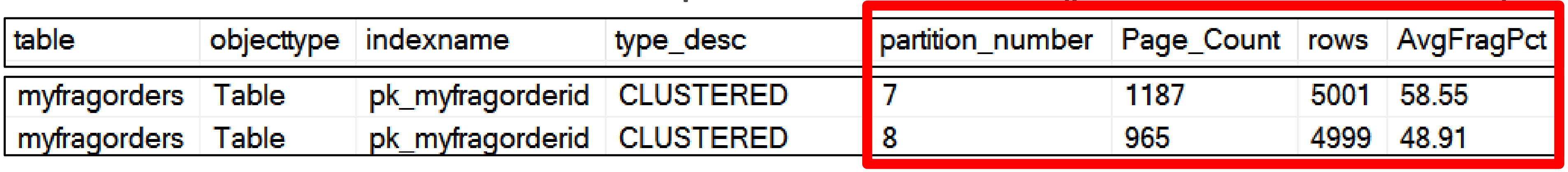


table	objecttype	indexname	type_desc	partition_number	Page_Count	rows	AvgFragPct
myfragorders	Table	pk_myfragorderid	CLUSTERED	7	1187	5001	58.55
myfragorders	Table	pk_myfragorderid	CLUSTERED	8	965	4999	48.91

```
ALTER INDEX uk_myfragorderid  ON dbo.myfragorders  rebuild partition = 8
WITH
(
     SORT_IN_TEMPDB = ON,
     ONLINE = ON, -- NOT POSSIBLE PRIOR TO SQL 2014
     DATA_COMPRESSION = NONE,
     MAXDOP = 0
)
GO
```

DEMONSTRATION 04

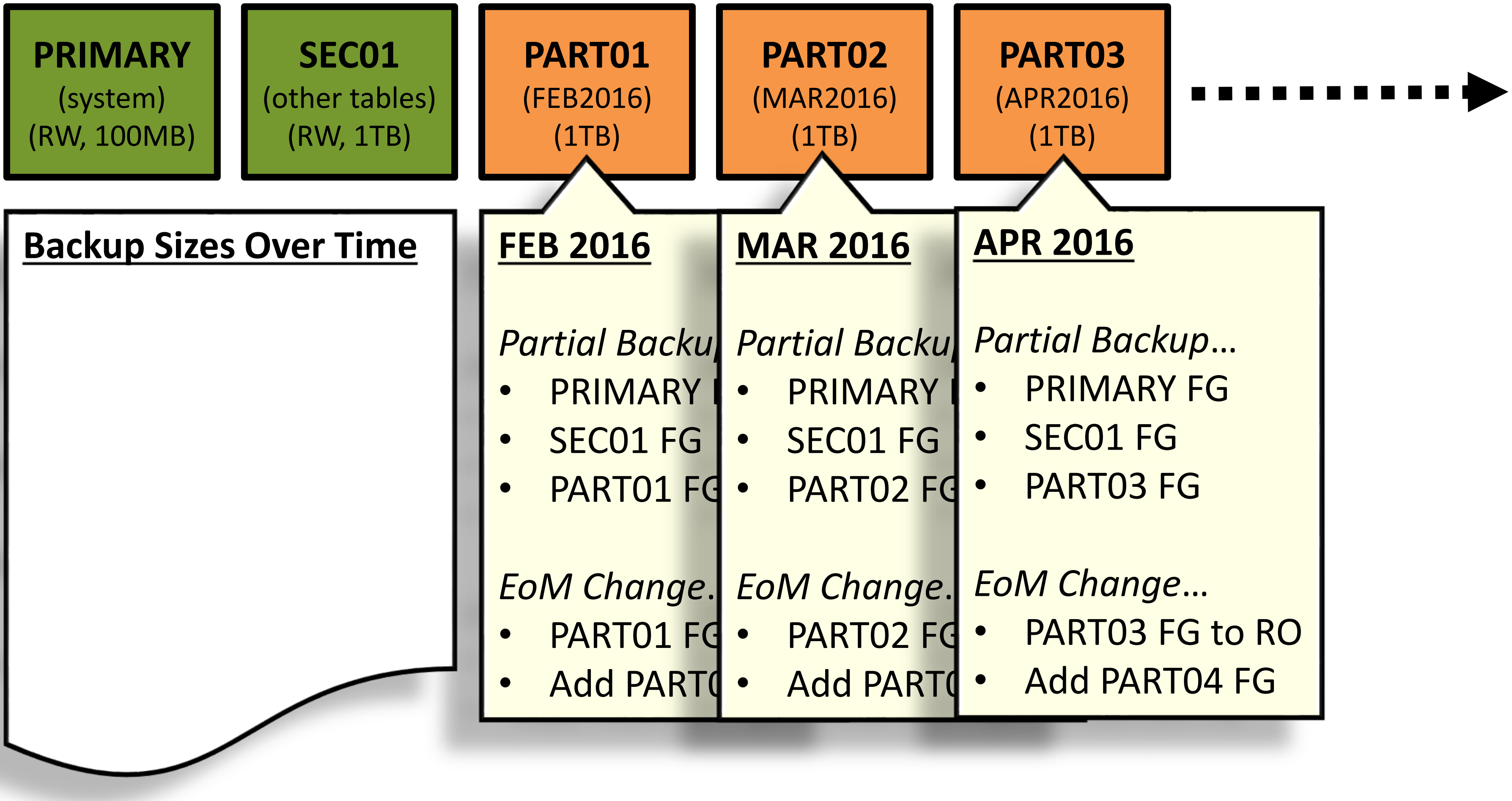
Partitioned Index Optimisation

Partitioning for Maintenance

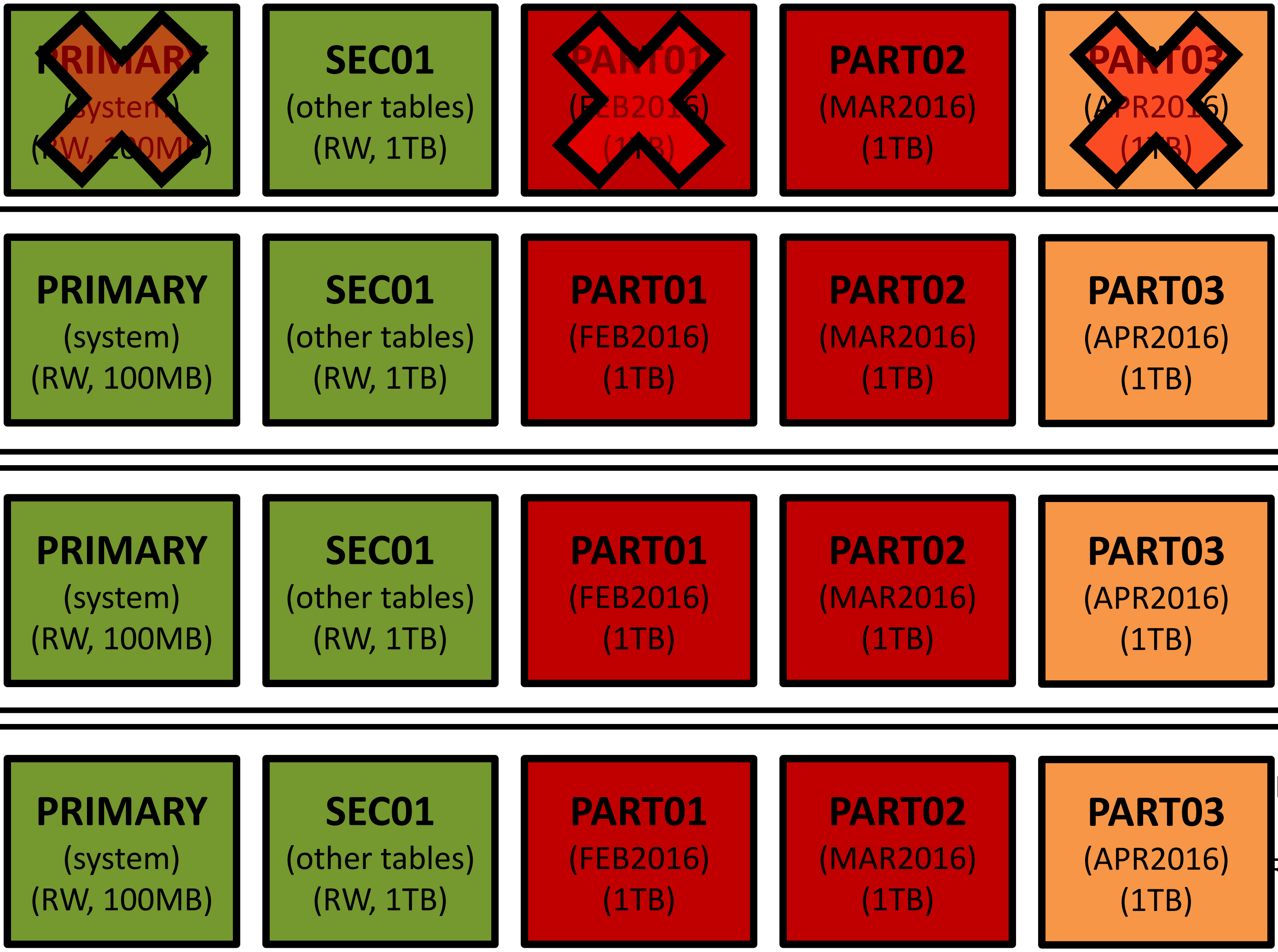
- **Partial Database Backups…**
  - If **parts** of DB **not updated** can set **FG's** to **READ_ONLY**
    - Is very nice if partitioned this way!
  - Can use **PARTIAL_BACKUP** to ONLY backup **RW FG's**

- **Partial Database Restores…**
  - Provide **quicker** DB **recovery** by <u>restoring</u> **RW FG's only**
  - <u>Recover</u> **RO FG's** in **descending** use order
  - DB **still accessible** even if FG's **not physically present**!

Partitioning for Maintenance - Backup



Partitioning for Maintenance – Restore



PART03 CORRUPT
DB offline? PART01+02 OK
Backup log tail
Restore PART03 from APR
Rollforward using logs

PART01 CORRUPT
DB offline? PART02+03 OK
Restore PART01 from FEB

PRIMARY CORRUPT
DB offline? PART01+02+03 OK
Backup log tail
Restore P+S+PART03 from APR
Rollforward using logs
Recover PART01+02

DEMONSTRATION 05

Partial Backup and Restore

Agenda

- **Introduction** - What is Partitioning
- Partitioning **101** – Understand the Basics
- Partitioning **201** – Beyond the Basics
- Improving **Query** Performance
- Streamlining **Maintenance** Operations
- **Best Practices**
- **Questions & Answers** (Q&A)

Wrapping Up – 10x Best Practice Tips

1. **Only** partition where **goals / benefits** clear
2. Choose suitable **partition column** used in **most queries**
3. **Always** leave **empty partition** at **start** / **end** of **ranges** (or pre-size it!)
4. Beware **impacts** to **PK**, **UK** and **FK** for partition **column**
5. Use **different FG's** for **different partitions** (for backup/restore)
6. Keep **regularly joined tables** in **same partition range** and **FG's**
7. **Don't share** partition function/scheme across objects (**SPLIT** impact)
8. If using **SWITCH** (for load/archive) then **partition align** all indexes
9. **Only rebuild fragmented** partitions, not full index
10. Use **MAXDOP**, **SORT_IN_TEMPDB**, **ONLINE** as **much** as possible

In partnership with PASS

Thank You… and Questions?

Please make sure you visit our fantastic sponsors:

Wardy it solutions™

RockSolid SQL

Microsoft

MELISSA DATA® Your Partner in Data Quality

JAMS ENTERPRISE JOB SCHEDULING

LobsterPot Solutions Pty Ltd

Profisee®

INFOREADY

sql agility

jade™

PLURALSIGHT

redgate ingeniously simple

PASS

How did we do?

Please complete an online Evaluation to be included the draw for a fantastic prize! There is a prize for each session timeslot and for the overall event survey – so the more feedback you provide the more chances you have.

Session Surveys

http://www.sqlsaturday.com/464/sessions/sessionevaluation.aspx

Post-Event Survey

http://www.sqlsaturday.com/464/eventeval.aspx

Appendix & References

- **Mr Fox SQL Partitioning Series**

  https://mrfoxsql.wordpress.com/2015/11/24/calculating-table-partition-sizes-in-advance/

  https://mrfoxsql.wordpress.com/2015/11/10/implementing-partition-aware-index-optimisation-procedures/

  https://mrfoxsql.wordpress.com/2015/07/07/implementing-partial-backups-and-restores/

  https://mrfoxsql.wordpress.com/2015/06/10/rebuilding-existing-partitioned-tables-to-a-new-partition-scheme/

  https://mrfoxsql.wordpress.com/2015/05/21/performance-impacts-of-partitioning-dml-triggers/

  https://mrfoxsql.wordpress.com/2015/05/13/deciding-whether-to-align-non-clustered-indexes/

  https://mrfoxsql.wordpress.com/2015/04/26/rebuild-a-standard-table-to-a-partitioned-table/

- **SQL Server Table Partitioning Resources**

  http://www.brentozar.com/sql/table-partitioning-resources/

- **Impacts of Partition SPLIT**

  http://blogs.msdn.com/b/sql_pfe_blog/archive/2013/08/13/oops-i-forgot-to-leave-an-empty-sql-table-partition-how-can-i-split-it-with-minimal-io-impact.aspx

- **Partial database backups**

  https://msdn.microsoft.com/en-us/library/ms191539(v=sql.120).aspx

- **Piecemeal restore of a database**

  https://msdn.microsoft.com/en-us/library/ms188671(v=sql.120).aspx

Appendix & References

- Some helpful building options…
  - ONLINE = **ON**
  - SORT_IN_TEMPDB = **ON**
  - MAXDOP = **0**

Will keep object online
Will take longer to execute
(*could use substantial tempdb*)

For high performance tempdb
(*Need approx. size of largest index*)

Can throttle command
or can open flood gates!