

SQLskills Immersion Event

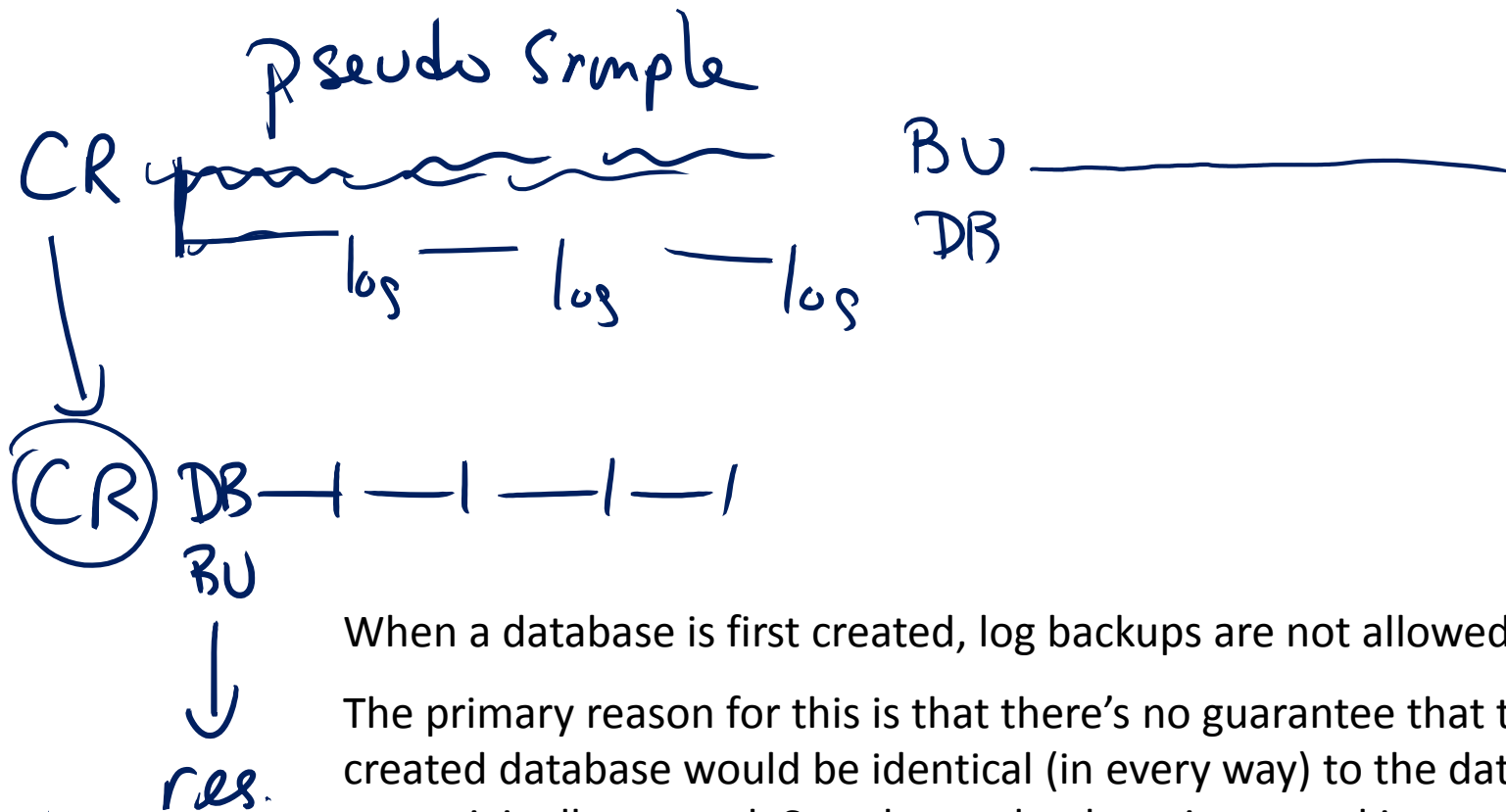
IEHADR: High Availability and Disaster Recovery

Module 4: Backup Strategy and Internals

Kimberly L. Tripp
Kimberly@SQLskills.com



NOTE: Includes updated to slides, a couple of new slides, some partitioning / VLDB slides/drawings plus whiteboard drawings from prior IEHADR deliveries (w/drawings done the week of May 8, 2017 in Chicago, Illinois – IEHADR Training)



When a database is first created, log backups are not allowed.

The primary reason for this is that there's no guarantee that the newly created database would be identical (in every way) to the database as it was originally created. So, when a database is created it runs in the pseudo-simple recovery model. This is where the log is cleared on checkpoint. There's no reason to keep it, it cannot be backed up.

However, once you do your first database backup, the full recovery model can be leveraged and therefore the log no longer clears...

This is the reason that some folks end up with a GIANT transaction log when they've only been performing full backups.

10GB DATA \ HUGG LOG (961GB)

In Full what to do?

if BU ITB?

Switch to simple (clears the log)

— [Full backup (only backs up data not free space)
Switch back Full?

Fix log size (shrinkfile)

BU 10GB \ 20GB log

What do you do if you have a transaction log that's grown wildly out of control?

Switch to simple to clear the log

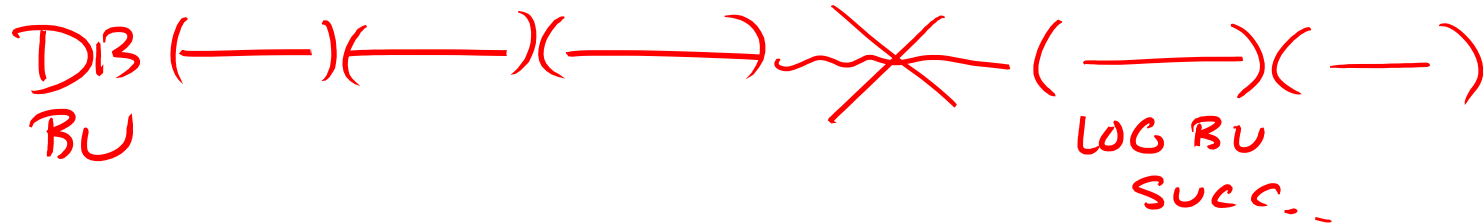
Perform a full database backup

Consider switching back to FULL (if you're going to perform log backups) OR stay in SIMPLE.

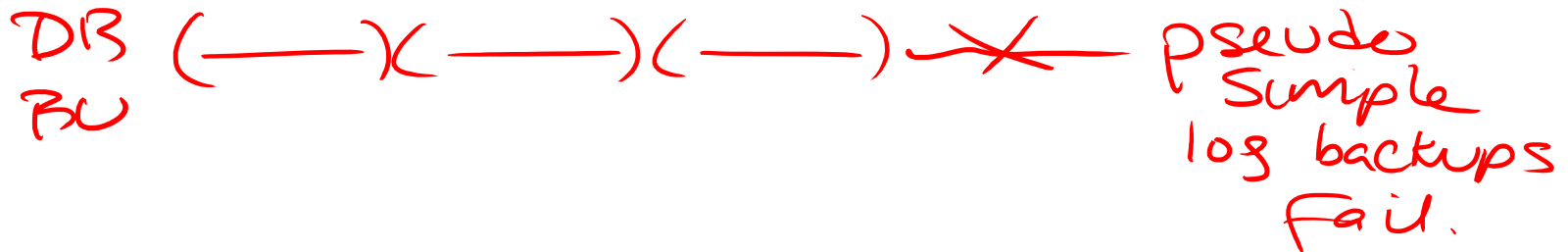
Fix the log size (DBCC SHRINKFILE)

Do another backup? Or, at least a log backup (if you're in FULL)

2000



2005



SQL Server 2000 allows the transaction log to be backed up even after the continuity of the log has been broken... Don't let this happen use trace flag 3231 to disable NO_LOG and TRUNCATE_ONLY for the log.

SQL Server 2005 sets the database back to a pseudo-simple recovery model after the continuity of the log has been broken. But, they still ALLOW the continuity of the log to be broken. Use trace flag 3231 OR 3031 to stop this from happening.

SQL Server 2008 does not allow manual clearing of the log. The NO_LOG and TRUNCATE_ONLY syntax has been deprecated.



Here we were discussing how log backups can occur concurrently with full database backups. They can even run numerous times during a single full. Each log backup will run but it will NOT clear the log. When the backup completes the data copy phase, the necessary log information will be backed up as well. This will allow the no-longer-needed log information to be cleared but it is not the full backup that caused the log to clear – it's the log backups that have already been done!

Also, if a full backup becomes damaged, SQL Server can use the logs as if the full backup had never occurred. The restore process would be:

- Use the most-recent (but working 😊) full database backup

- Use the last differential based on that full backup

- Use any logs from the last differential and until current (up-to-the-minute)

VLF Fragmentation

We briefly discussed VLFs (covered in IEPTO1), here are the relevant links:

- Kimberly’s blog post *“8 Steps to better Transaction Log throughput”*
 - <http://www.sqlskills.com/blogs/kimberly/post/8-Steps-to-better-Transaction-Log-throughput.aspx> (<http://bit.ly/67DFtV>)
- Beware of exactly 4GB growth/auto-growth bug
 - <http://www.sqlskills.com/BLOGS/PAUL/post/Bug-log-file-growth-broken-for-multiples-of-4GB.aspx> (<http://bit.ly/c3QQLr>)
 - Fixed in 2008 R2: <http://support.microsoft.com/kb/2633151>

2000

~~NORECOVERY~~

=

file

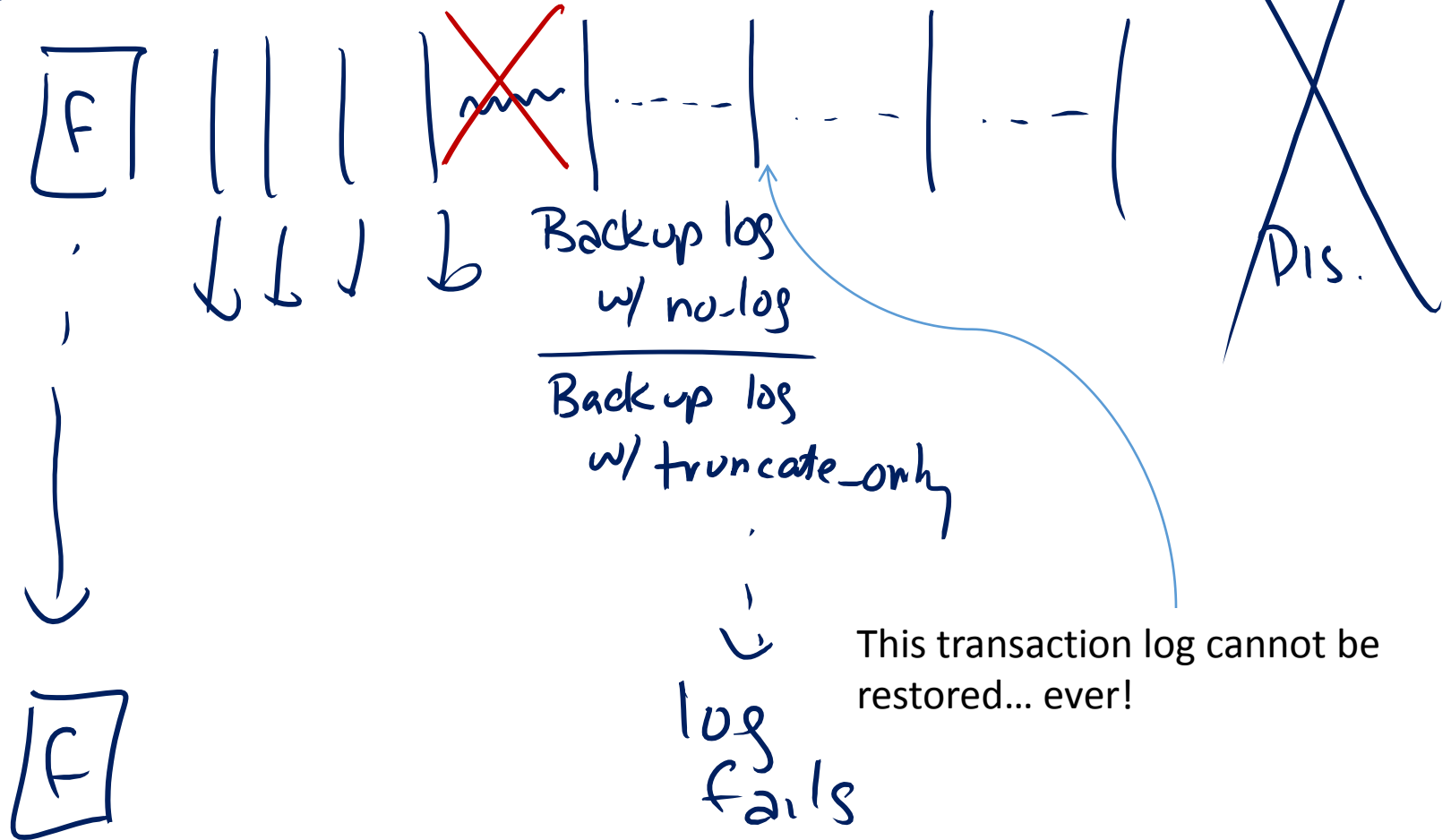


LSN

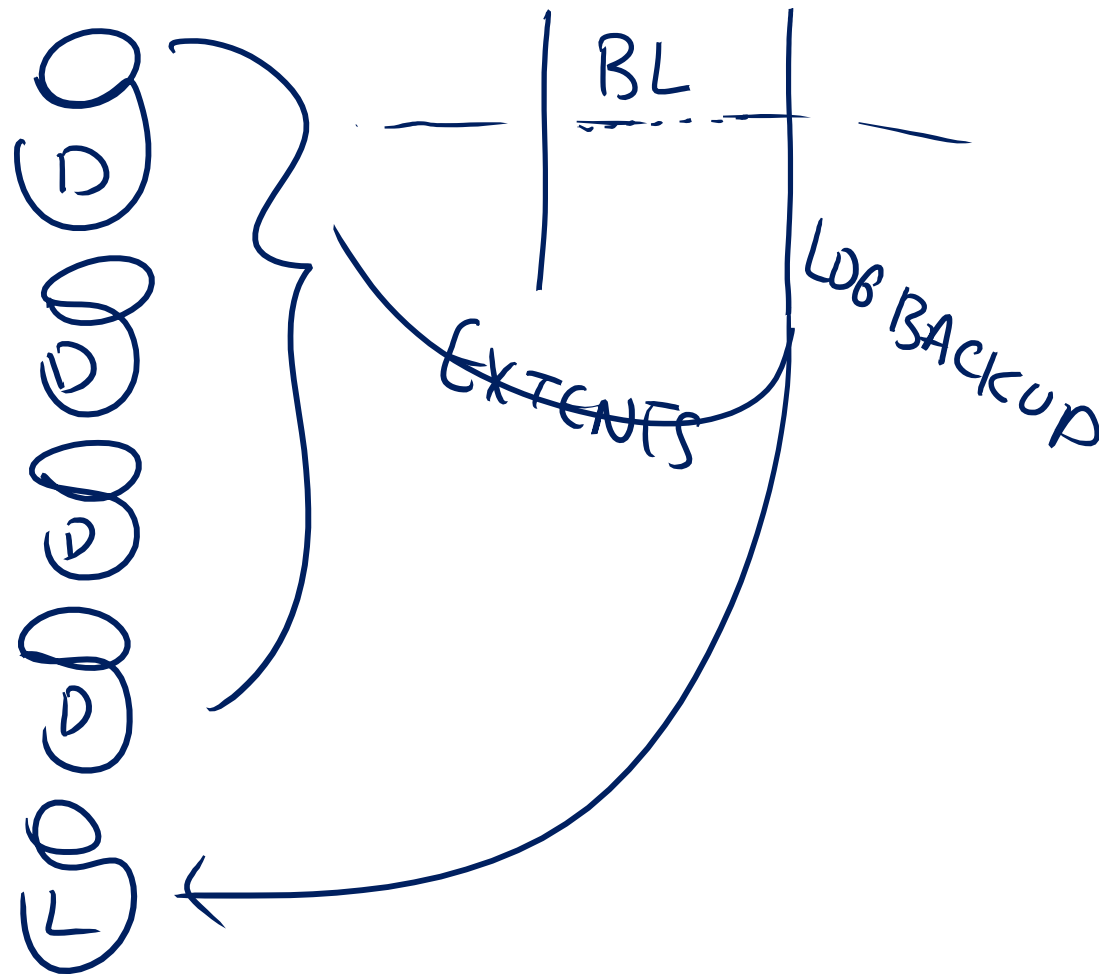
MIN

In SQL Server 2000, log backups could NOT occur concurrently with full database backups. However, log backups COULD occur concurrently with FILE or FILEGROUP backups. As a result, you could replace your full database backup strategy with FILEGROUP backups (and you have to backup ALL files/filegroups) and never have to stop your log backup job. This also meant that your secondary sites would NOT fall far behind (because log backups would NOT be blocked during FILE/FILEGROUP backups).

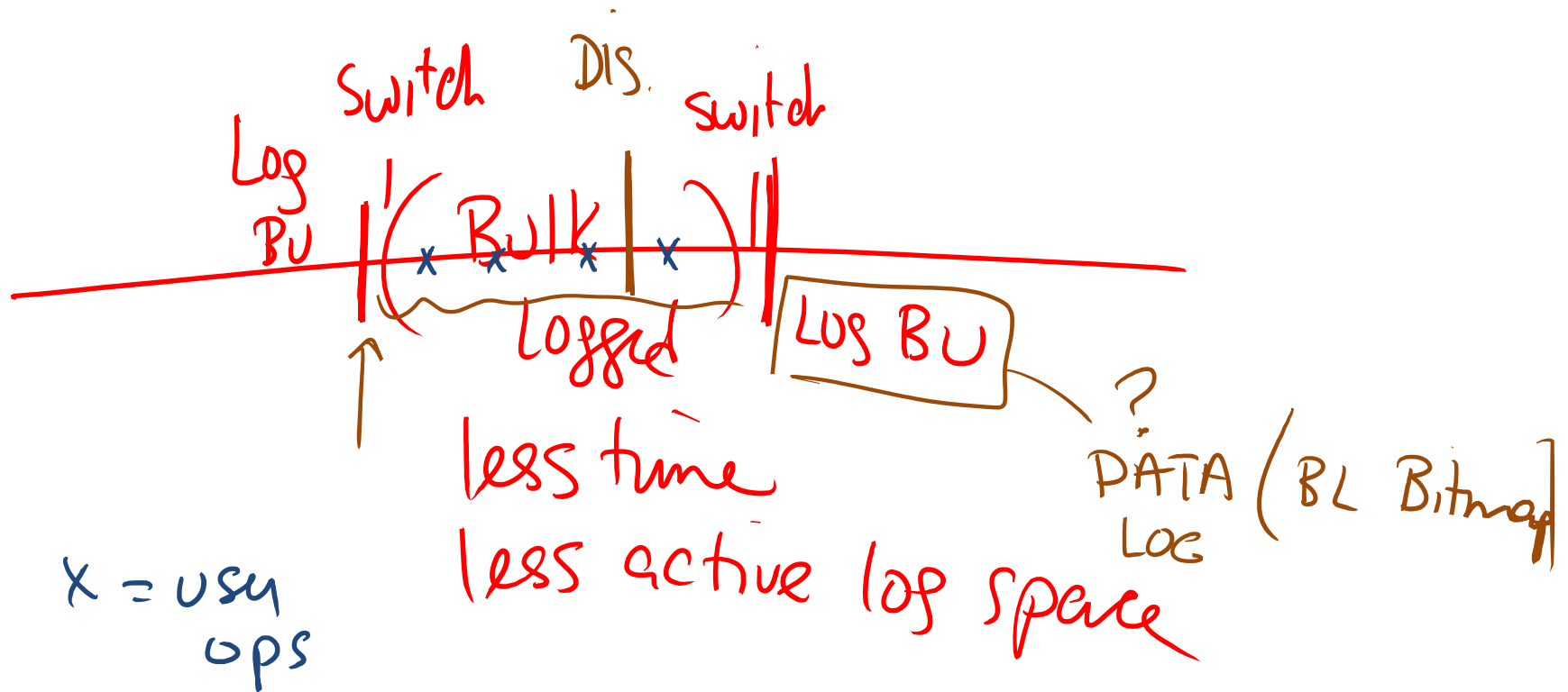
2000



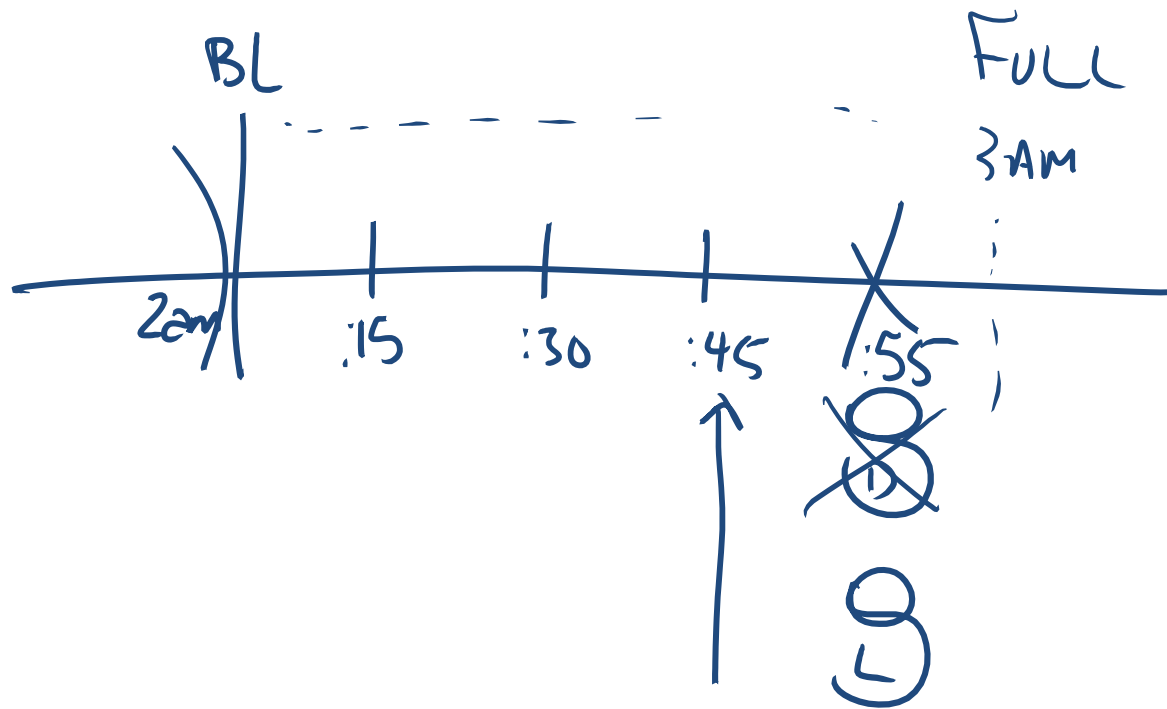
I haven't looked at this in a long time but there used to be a time (on SS200 – might still be true?) where clearing the log (and breaking the continuity of the log) “worked” but then SQL Server just allows additional log backups to succeed (without warning). Only on restore will you find out that you can't use those logs... yes, a differential or full backup will reduce your dependency on logs BUT use trace flag 3231 to disallow this from happening to begin with!



When you do a transaction log backup AFTER having performed bulk operations then the size of the backup will be about the same size it would have been had you been fully logged – why? Because SQL Server accesses the data portion combined with the log to get all of the needed information for recovery.

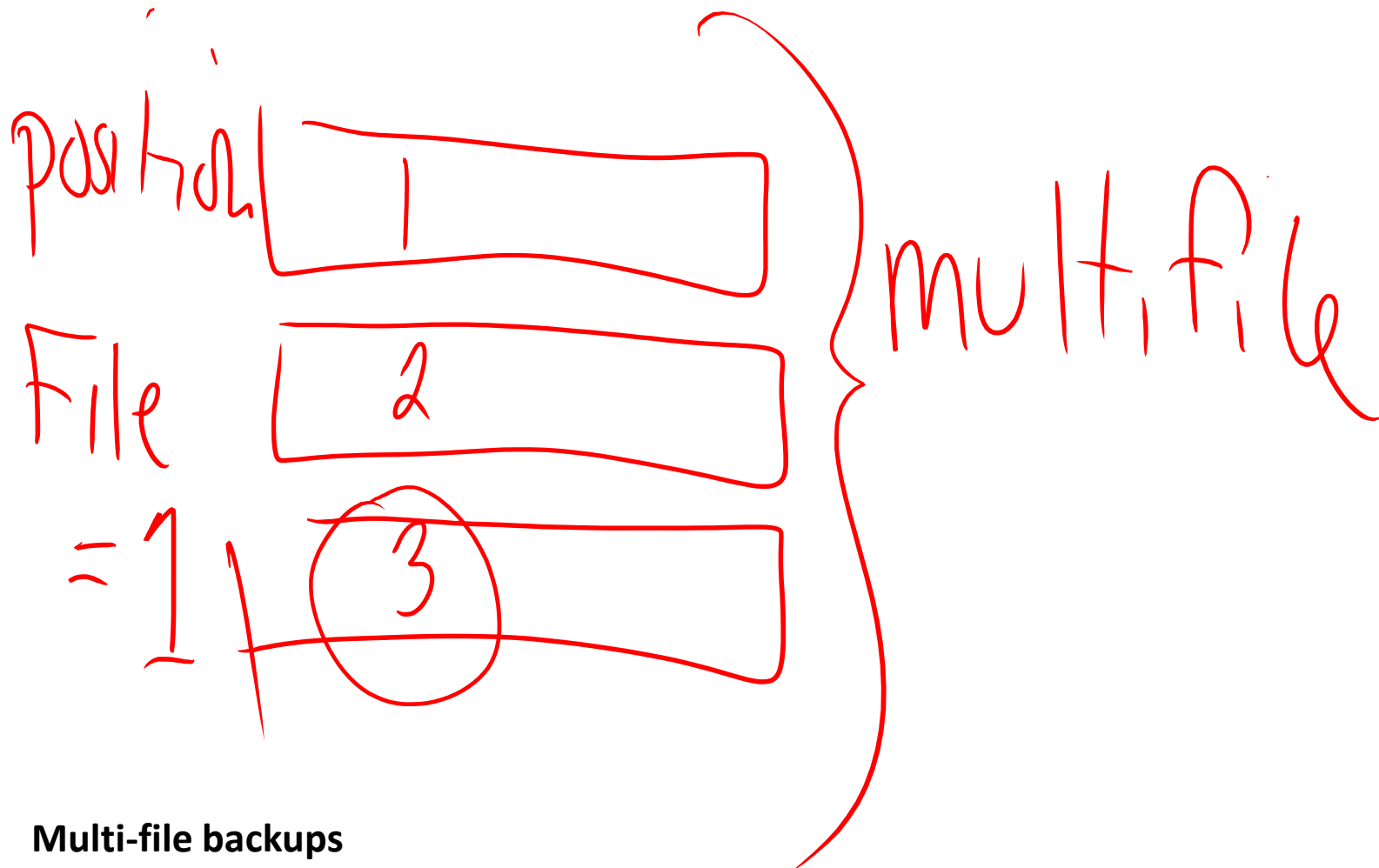


If you plan to use the BULK_LOGGED recovery model during certain batch/bulk operations, you should be sure to minimize the potential data loss by performing log backups before the change TO bulk_logged as well as AFTER the change back to the FULL recovery model. Remember, if a disaster were to occur while SQL Server is in BULK_LOGGED you cannot access the tail of the log (if any bulk operations have occurred). This is what you would lose if there were a disaster during the batch process.



Best Place for Bulk_Logged

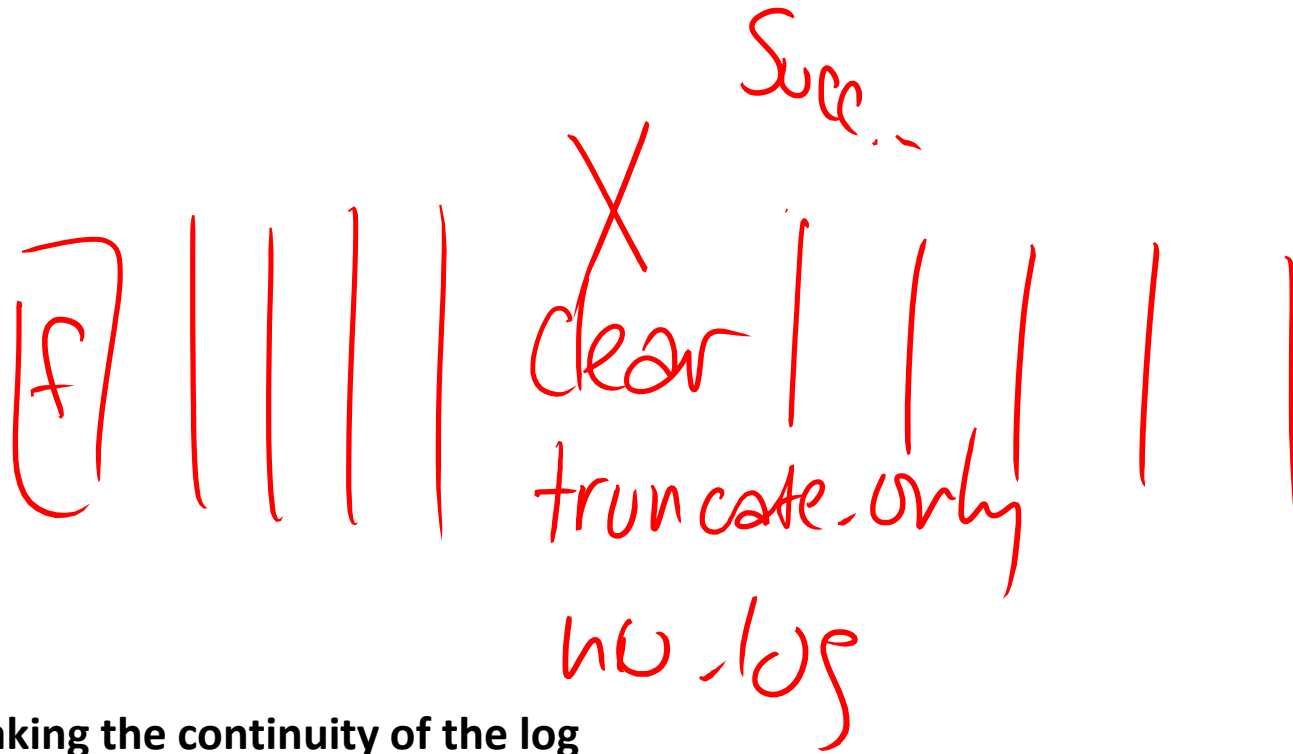
While you're in the bulk_logged recovery model (and IFF a bulk_logged operation as occurred) you can backup the log – as long as the data portion is available. If the data portion becomes damaged then up to the minute recovery will not be possible. It's very important to control when the db is in bulk_logged because even a table creator could put you into a minimally logged mode when you don't expect it (if you just always run in bulk_logged). See the prior slide for the best way to set/change recovery models from bull to bulk_logged and back.



Multi-file backups

Are multiple backups to one or more devices (known as a media set). Once a media set has been defined, all backups to that media set must match in number as well as compression setting. If you want to change compression or use a different number of backup devices – you must use the WITH FORMAT option. When specified the media set is overwritten and any remaining “fragments” are completely and totally unusable.

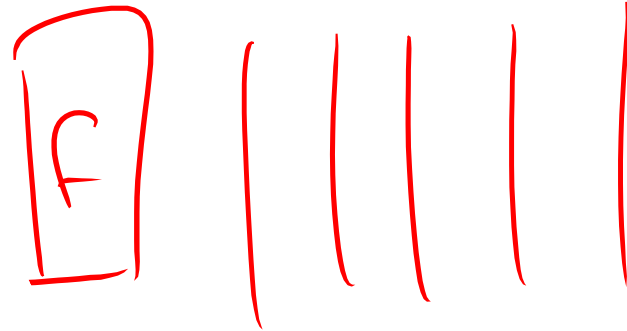
2000



Breaking the continuity of the log

In SQL Server 2000, you can continue to successfully perform log backups even after the continuity of the log has been broken (YIKES!). To stop this possibility of this happening when someone uses `BACKUP LOG WITH TRUNCATE_ONLY` or `NO_LOG`, use trace flag 3231 (see the slides for more details). These commands are turned into a NOOP (null operation) in SQL Server 2000 and 2005 with 3231. On SQL Server 2005, 3230 turns them into a CHECKPOINT (which has no [negative] effect on databases in the FULL or BULK_LOGGED recovery models).

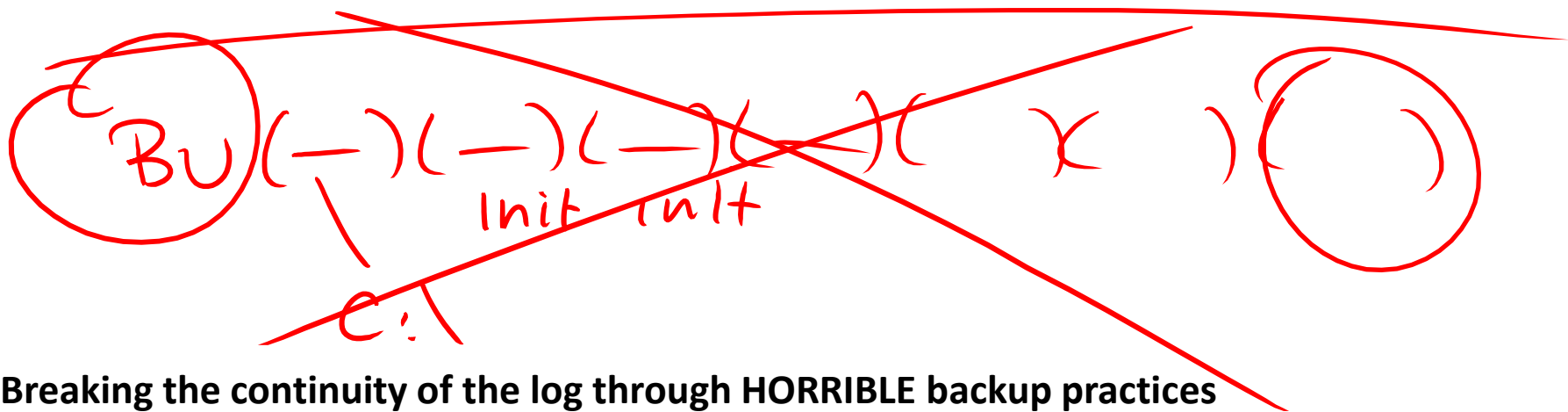
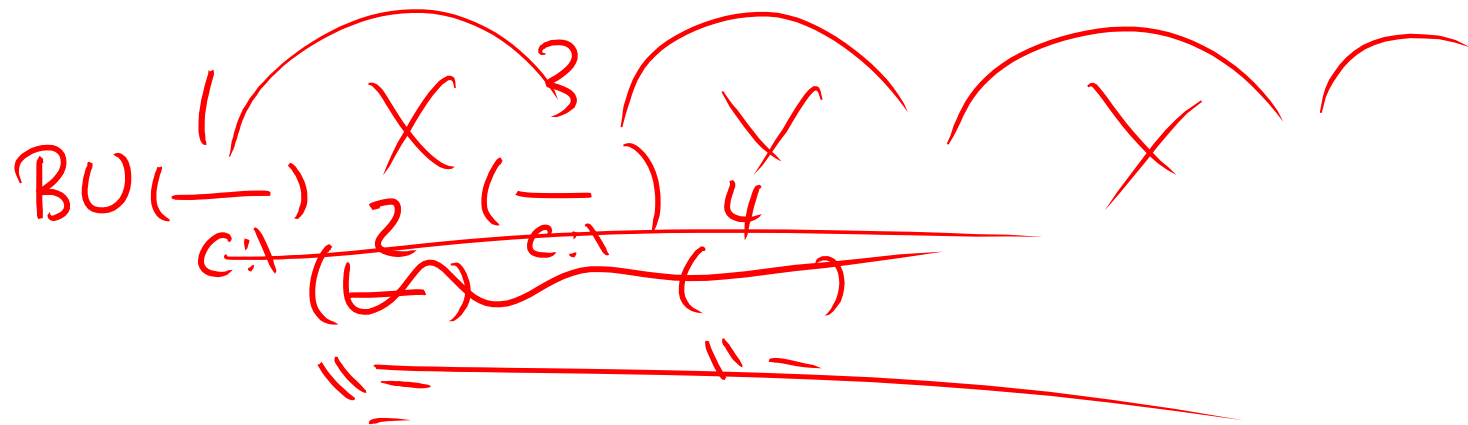
2005 +



Breaking the continuity of the log

In SQL Server 2005, breaking the log chain breaks the ability to backup the log without first doing some other “data” backup (full database, database differential, full filegroup, filegroup differential, full file or file differential). Your database is put into a pseudo SIMPLE recovery model where you log is cleared on checkpoint. Transaction log backups will fail.



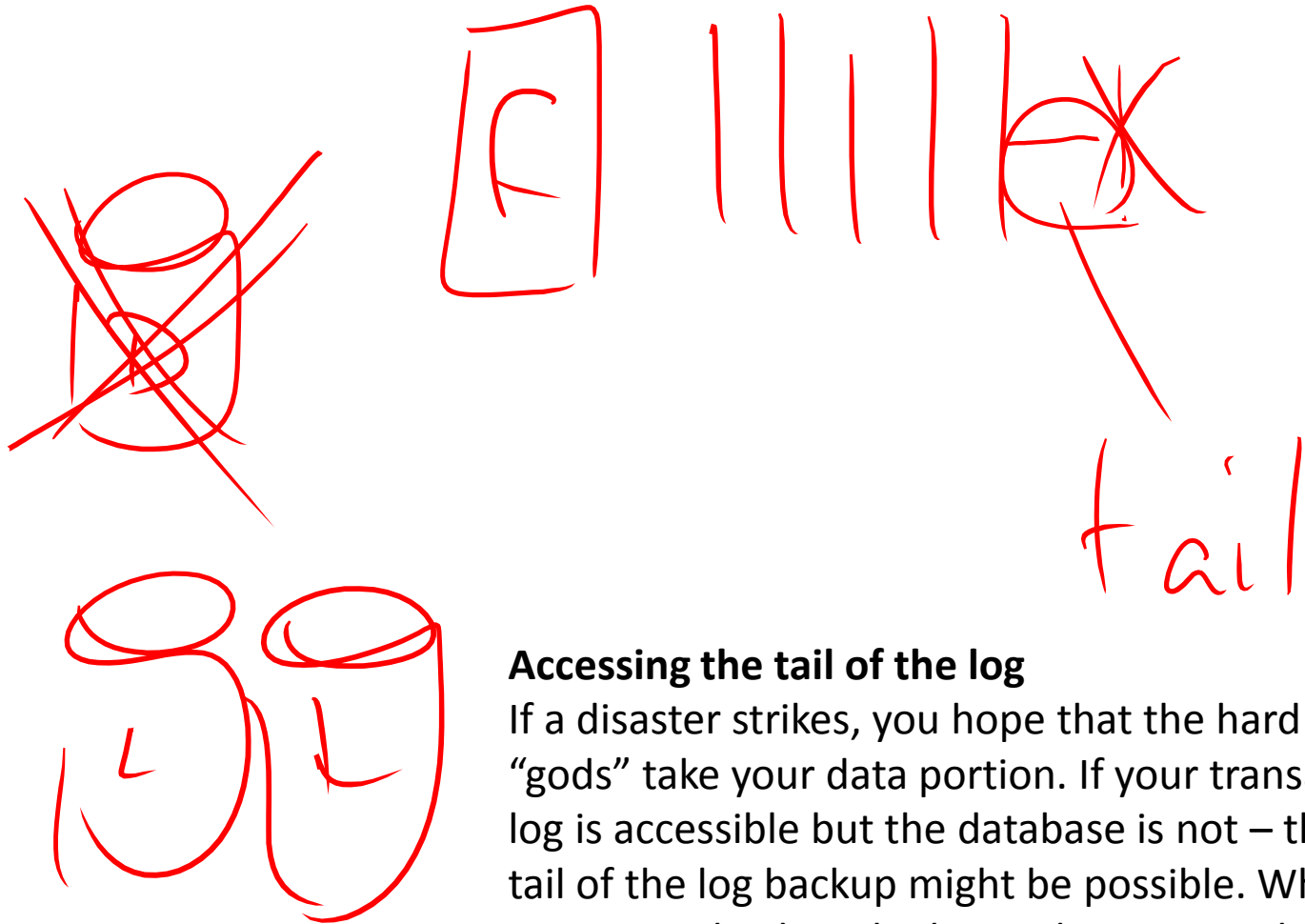


Breaking the continuity of the log through HORRIBLE backup practices

The first “strategy” was one that Edward had experienced. His customer had been backing up every OTHER backup to the network. So, local then network, then local, then network. This is a recipe for disaster!

The second “strategy” is one where the customer did NOT understand log backups AT ALL. They were backing up every log with INIT. If they had had a disaster they would have had the backup plus ONLY the last few transactions with NO WAY of getting the log to a useable point.

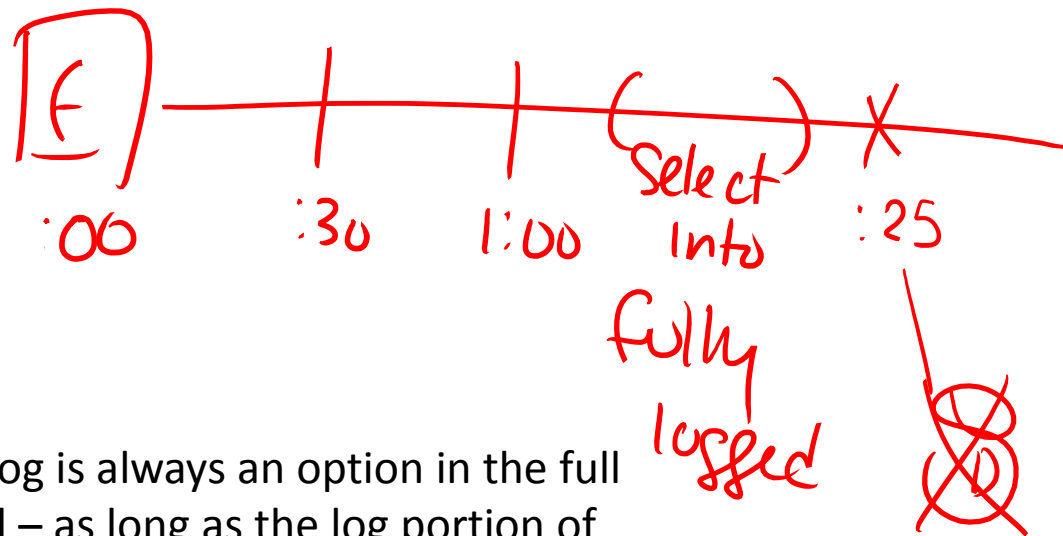
Key Point: TEST YOUR BACKUP/RESTORE STRATEGY



Accessing the tail of the log

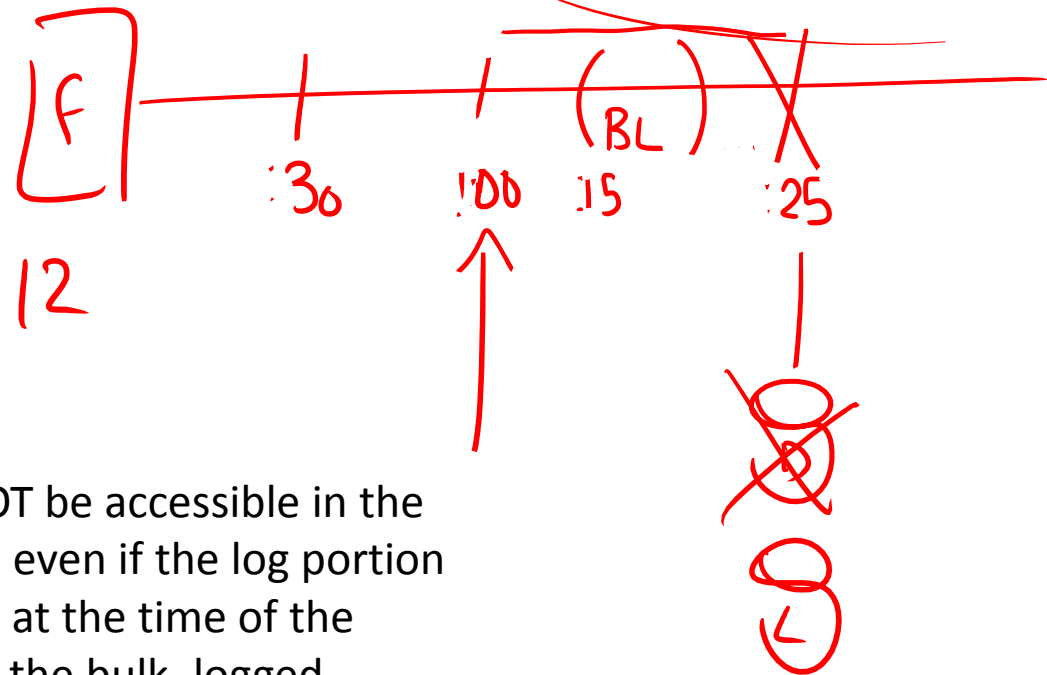
If a disaster strikes, you hope that the hard drive “gods” take your data portion. If your transaction log is accessible but the database is not – then a tail of the log backup might be possible. When it is, you can backup the log and get up-to-the-minute recovery of your database (with ZERO data loss). This is why redundancy for the log is the most important – even more so than for the data portion. However, you need redundancy there as well to minimize downtime.

Full RM



The tail of the log is always an option in the full recovery model – as long as the log portion of the database is accessible at the time of the disaster. If the database is in the FULL recovery model then ALL operations are fully logged (even those that have the ability to be minimally logged; they are ONLY minimally logged if the database recovery model is NOT FULL).

Ⓛ Tail is possible

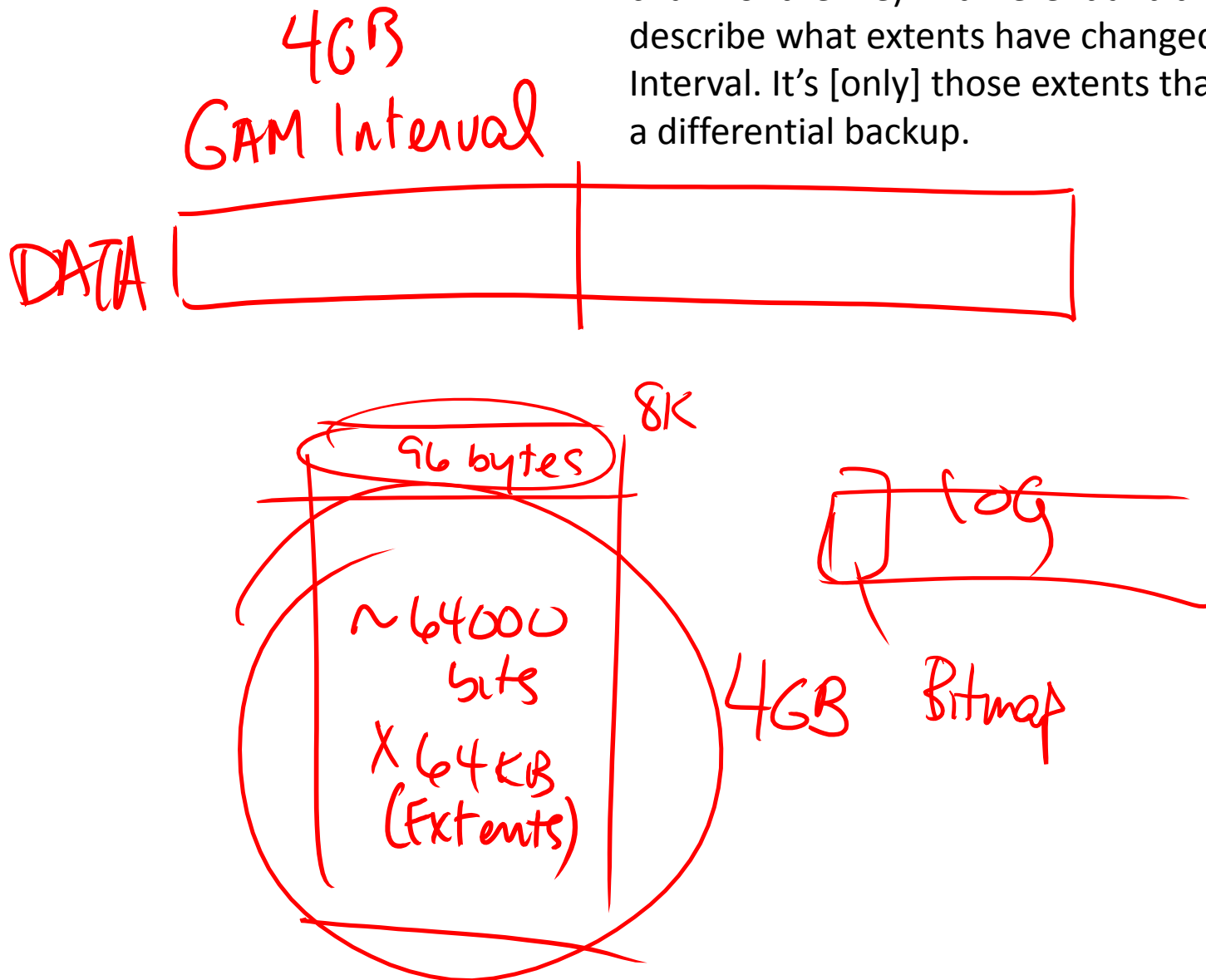


The tail of the log MIGHT NOT be accessible in the bulk_logged recovery model even if the log portion of the database is accessible at the time of the disaster. If the database is in the bulk_logged recovery model AND a bulk operation HAS occurred then the tail of the log cannot be backed up.

End result – be sure to protect the log by restricting the time that a database is in the bulk_logged recovery model and do not allow users into the database during this time.

IEPTO1 Internals Review

“Bitmap” pages in SQL Server use the 8K page as a map to describe *something* about a GAM Interval (4GB chunk of the file). A differential bitmap uses a page to describe what extents have changed within that GAM Interval. It’s [only] those extents that are backed up on a differential backup.



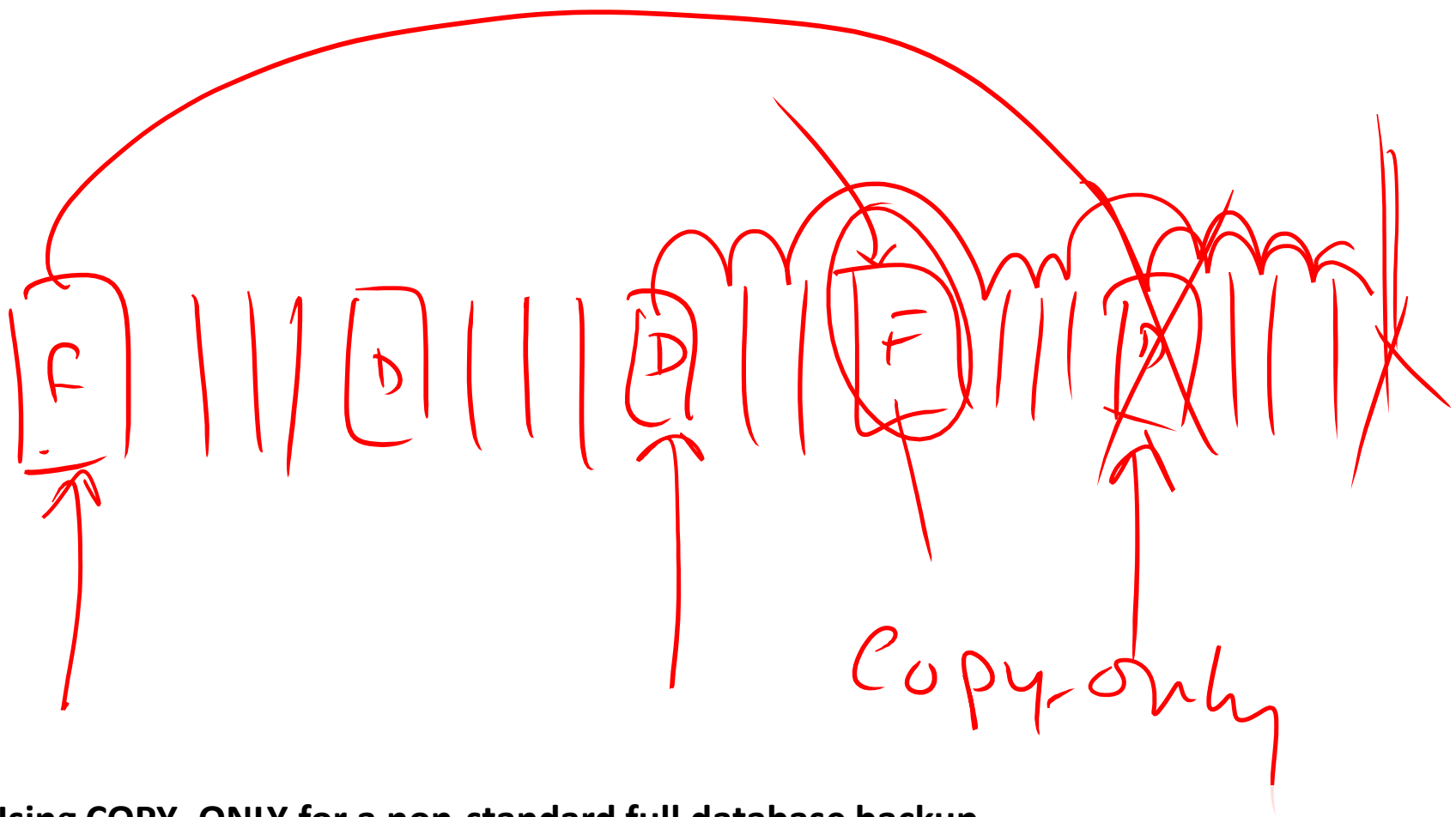
Damaged Partition: Summary

- Does not render the database unavailable as only the damaged filegroup is unavailable
- Does not render the partitioned table or partitioned view unavailable as only the damaged data is unavailable but the partitioned object can still be queried/accessed

NOTE: A partitioned view that has tables that are inaccessible generates an error when accessing the view; however, the tables on which the view are defined are still accessible. A partitioned table does NOT.

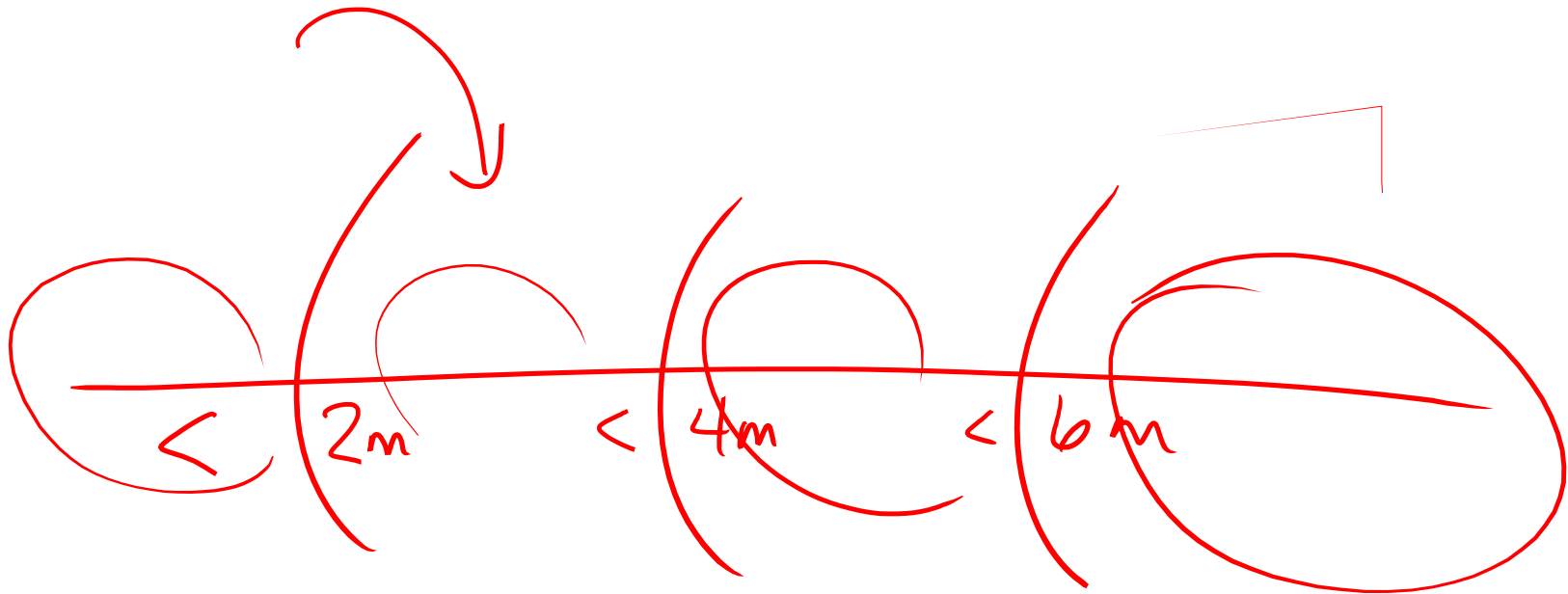
TIP: What you need to do with PVs is modify the view on EVERY restore (to remove / add tables to the view based on what's inaccessible / accessible after each restore).

- **Can proceed with recovery, also [almost] ONLINE!**
 - ❑ There is one hiccup here... (taking a file offline terminates ALL database connections)
 - ❑ Minimal impact to downtime as files are brought offline and/or when file is restored
- **Users can access the database during restore**



Using COPY_ONLY for a non-standard full database backup

If a developer/tester or another team needs a backup of a database, you do not just want to back it up and give it to them. Performing a full database backup resets the differential bitmap. All differentials would then be tied to that most recent backup and not the one that was performed during your regularly scheduled backups. As a result, your restore process might take quite a bit longer (and require quite a few more log backups to be applied). Use COPY_ONLY instead and the diff bitmap will not get reset!



Online operations – SalesDB partitioning

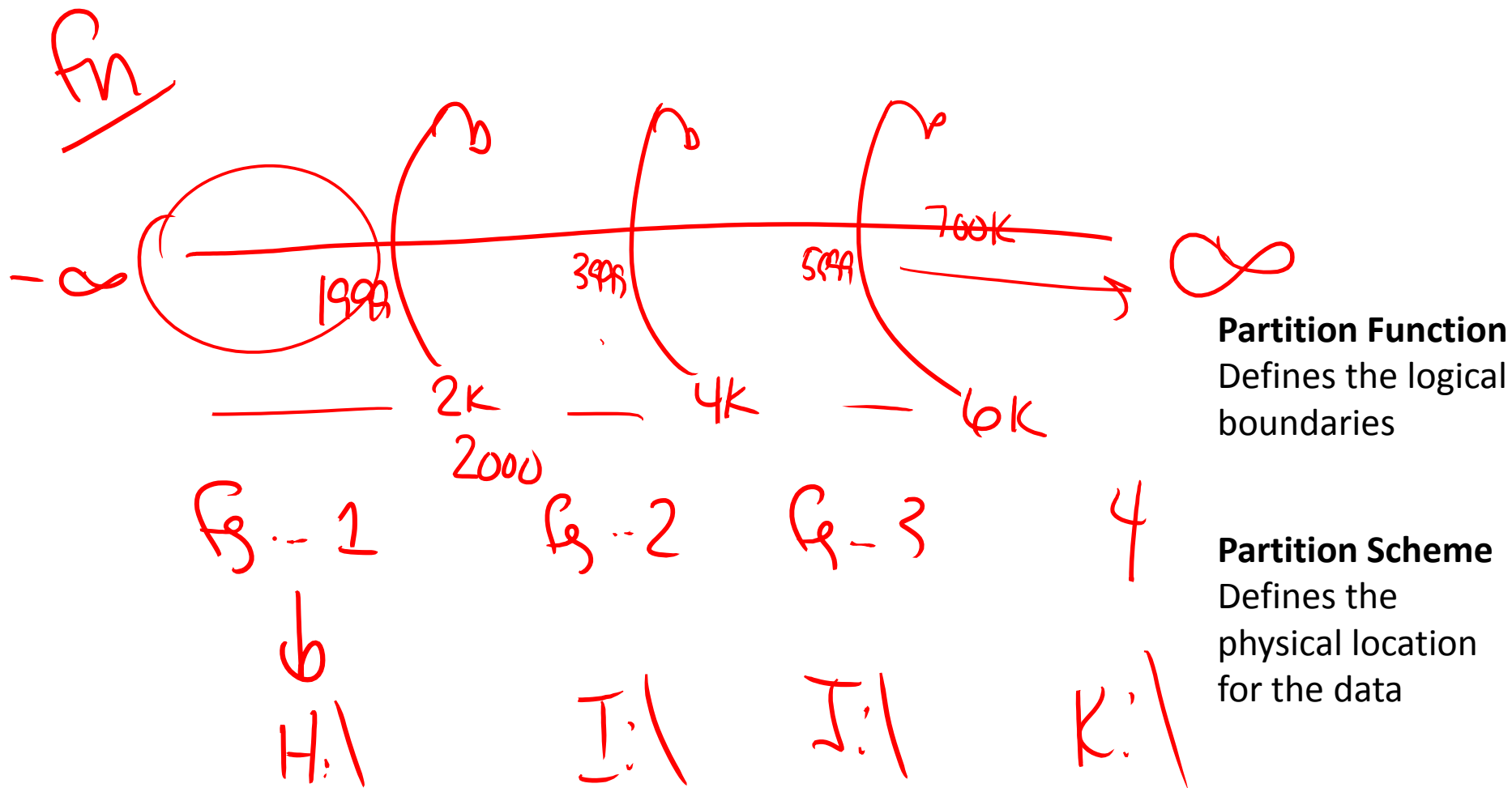
The sales table was partitioned with 3 boundary points: 2 million, 4 million and 6 million. It's a RIGHT-based partition function so the rows that match the boundary point will be to the RIGHT side. Specifically this translates into:

In partition 1: all rows less than 2 million

In partition 2: all rows equal to or greater than 2 million and less than 4 million

In partition 3: all rows equal to or greater than 4 million and less than 6 million

In partition 4: all rows equal to or greater than 6 million



Online operations – SalesDB partitioning

The sales table was partitioned with 3 boundary points: 2 million, 4 million and 6 million. It's a RIGHT-based partition function so the rows that match the boundary point will be to the RIGHT side. Specifically this translates into:

Drive H:\ (partition 1): all rows less than 2 million

Drive I:\ (partition 2): all rows equal to or greater than 2 million and less than 4 million

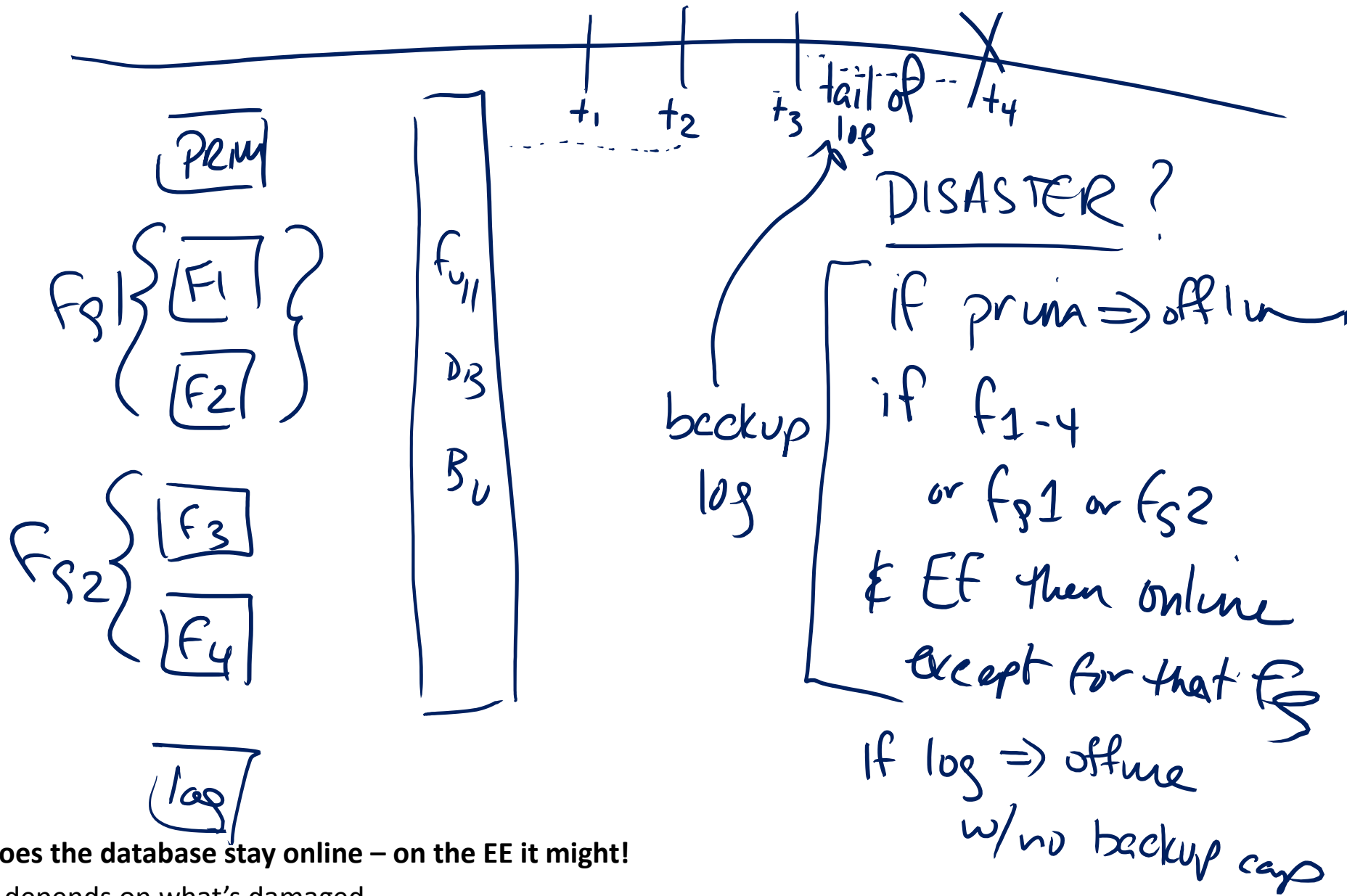
Drive J:\ (partition 3): all rows equal to or greater than 4 million and less than 6 million

Drive K:\ (partition 4): all rows equal to or greater than 6 million



Primary filegroup, Primary file?

The first file associated with a database is the MDF (main data file) which is also known as the PRIMARY DATA FILE (but Adobe beat us to PDF 😊). This file is critical for a database to be up and running. If you add files to a database without specifying a filegroup they are going to end up in the PRIMARY filegroup. Because the primary file is a member and because allocations can come from any file of the filegroup – ALL of the files of the primary filegroup end up being critical. Generally, it's important to isolate the primary filegroup (with just one mdf) and protect it (as much as you would the transaction log).



Does the database stay online – on the EE it might!

It depends on what's damaged....

If the primary (any file of the primary filegroup) or any log file are damaged – you're offline

If a user-defined filegroup has a damaged file then that filegroup will be offline.



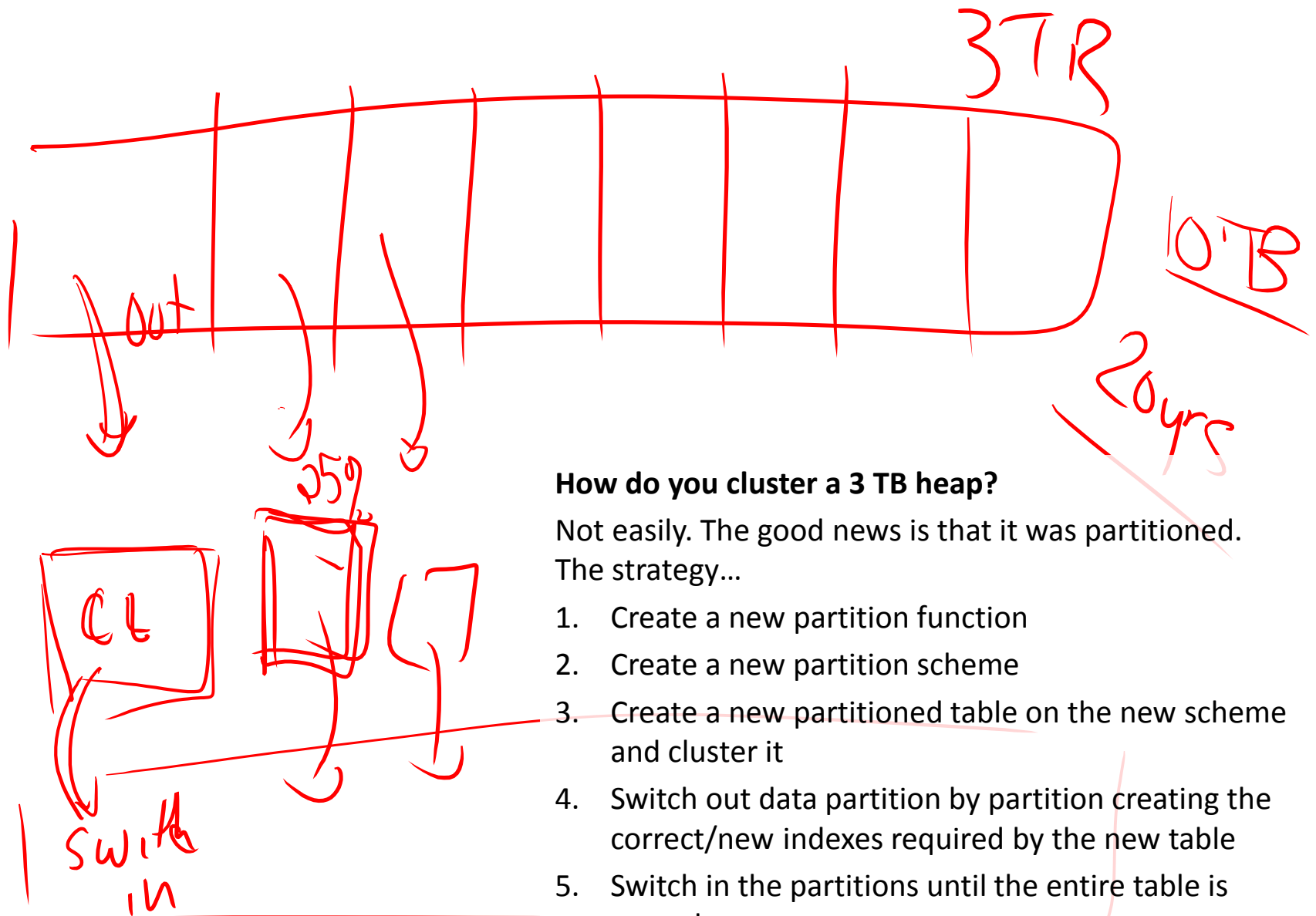
Sizing the PRIMARY filegroup

Sizing the primary is challenging but it's usually small. It's not directly related to data size but instead related to metadata complexity. Users/permissions as well as code (functions, procedures, assemblies) are what's stored in the "primary" portion of the database.

R0 2011 → R10
R0 2010 → R5
R0 2009 → R0

Isolating the primary filegroup

More specifically, isolate primary and separate this from your user-defined data. Create a filegroup for readwrite data that has at least 2 files and then read-only data can be in one or more filegroups (depending on availability requirements, backup/restore SLAs and hardware options/costs)



How do you cluster a 3 TB heap?

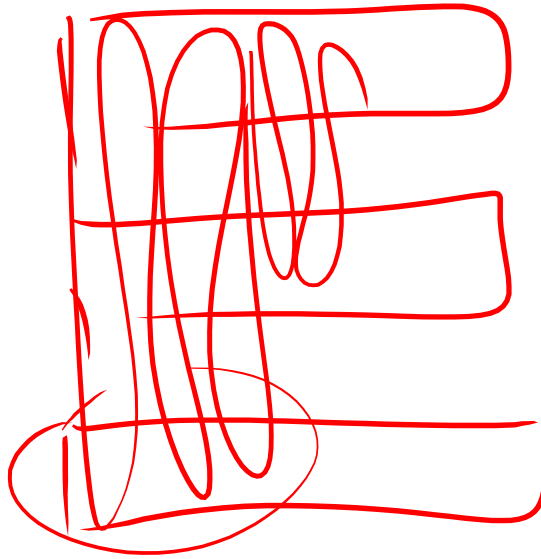
Not easily. The good news is that it was partitioned.
The strategy...

1. Create a new partition function
2. Create a new partition scheme
3. Create a new partitioned table on the new scheme and cluster it
4. Switch out data partition by partition creating the correct/new indexes required by the new table
5. Switch in the partitions until the entire table is moved over

Concerns:

- * primary/foreign keys
- * downtime

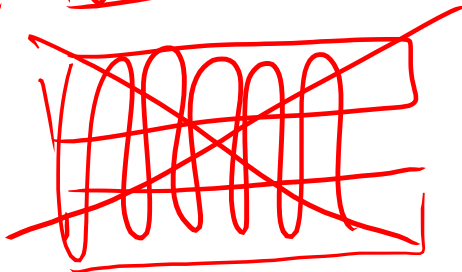
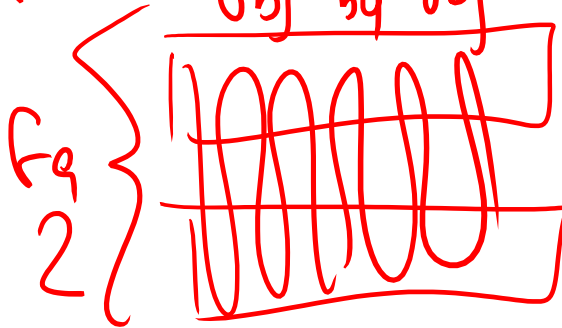
fg



Removing does
balance

Remove Empty file

obj by obj



not
ADD

New
fg
4

rebuild on



**Changing the number
of files in a filegroup**
See the next slide for
details.

Changing the number of files in a filegroup

Removing a file must be done through EMPTYFILE (which DOES rebalance the data among the remaining files in the filegroup). However, it is not object specific.

Adding a file to a filegroup does NOT rebalance. If you want to change a filegroup from two or three files to something like four or five, your best bet is to create a NEW filegroup and then rebuild all of your indexes into it.

Use CREATE with DROP_EXISTING to MOVE an index to another filegroup.

Now, as for moving LOB data... this is the most interesting and it DOES work if you use DROP_EXISTING and rebuild on a scheme or conversely from a scheme to just a single filegroup. However, going from one filegroup to another (even when using DROP_EXISTING) does NOT move the LOB data. There's a blog post in that information for sure. I might try and do this as a SQLServerPro post in my series on Partitioning (Partitioned Tables vs. Partitioned Views).

Here are the links to the current posts:

Q1: [Partitioned Tables v. Partitioned Views—Why are they even still around?](#)

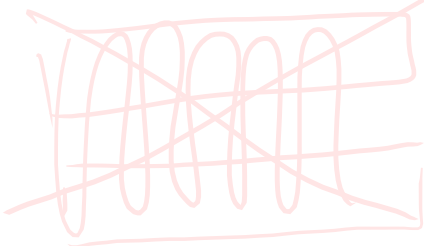
Q2: [Solutions to VLT concerns around statistics and maintenance!](#)

Q3: [Did SQL Server eliminate any partitions?](#)

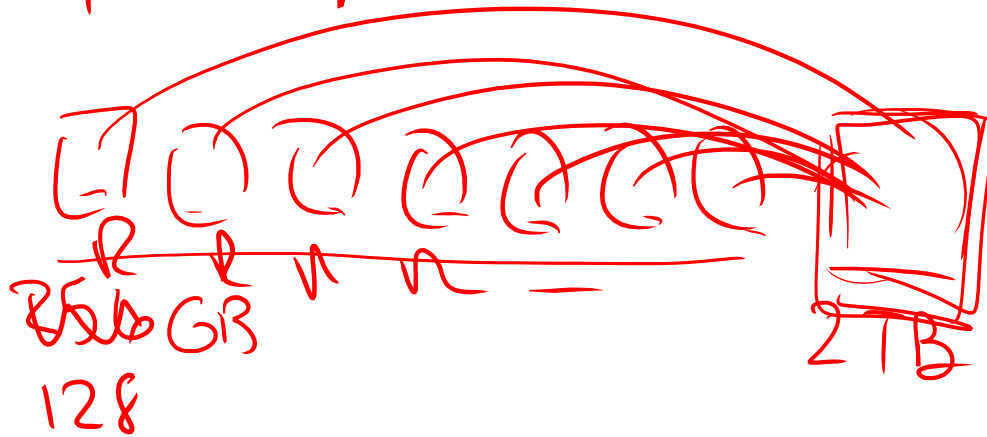
Q4: [How about Filtered Indexes instead of Partitioning?](#)

Q5: Is there a way to move LOB data? (or something like that)

The final consideration is availability. See the next slide for Brad's (@SQLPhilosopher's) trick!



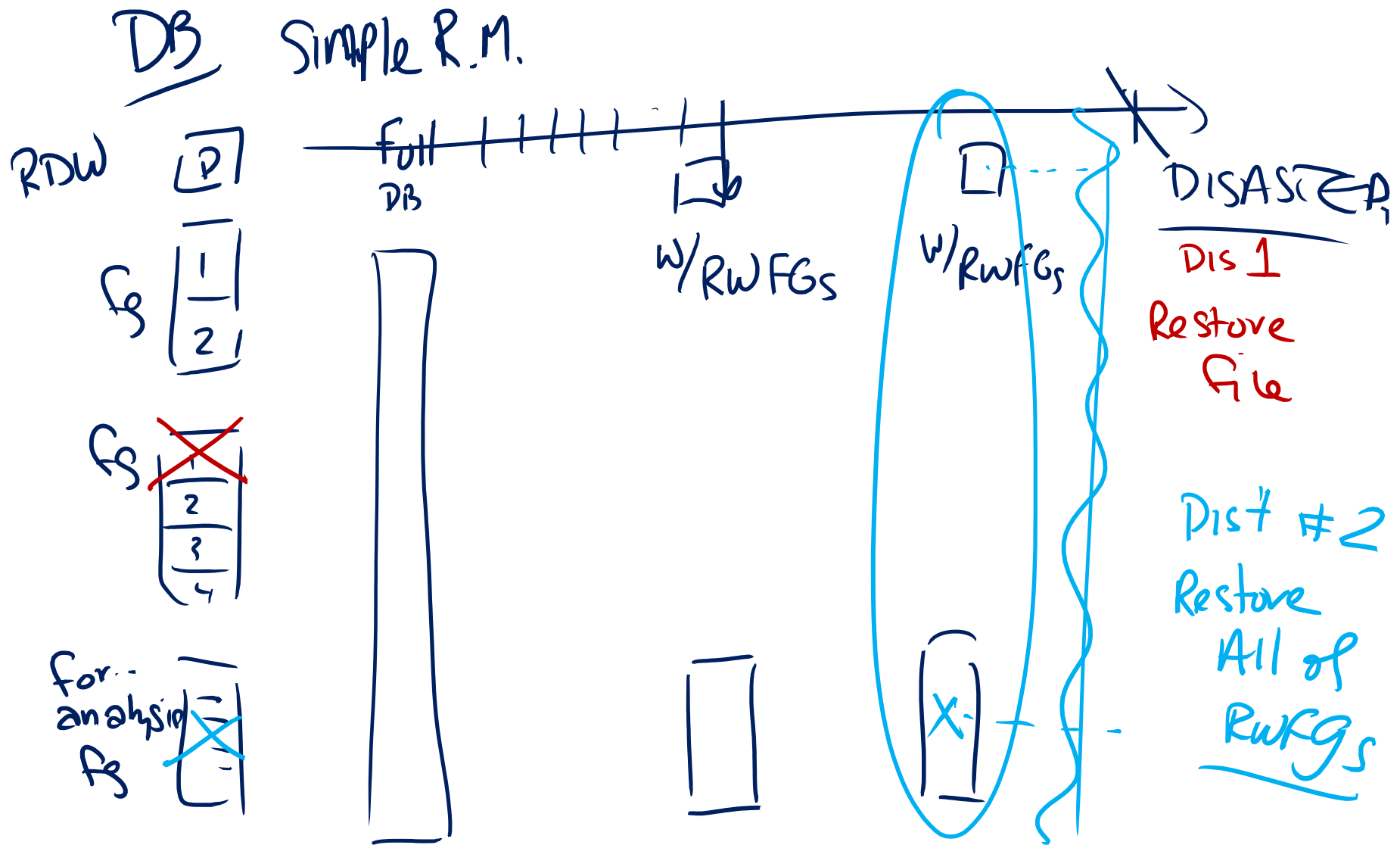
SQL philosopher . com



As an alternative, Brad (@SQLPhilosopher) came up with a way to move LOB data to a different number of files – and, do this as an ONLINE operation.

He described it fully here: <http://www.sqlphilosopher.com/wp/2012/02/moving-a-filegroup-to-a-new-disk-array-with-no-downtime/>.

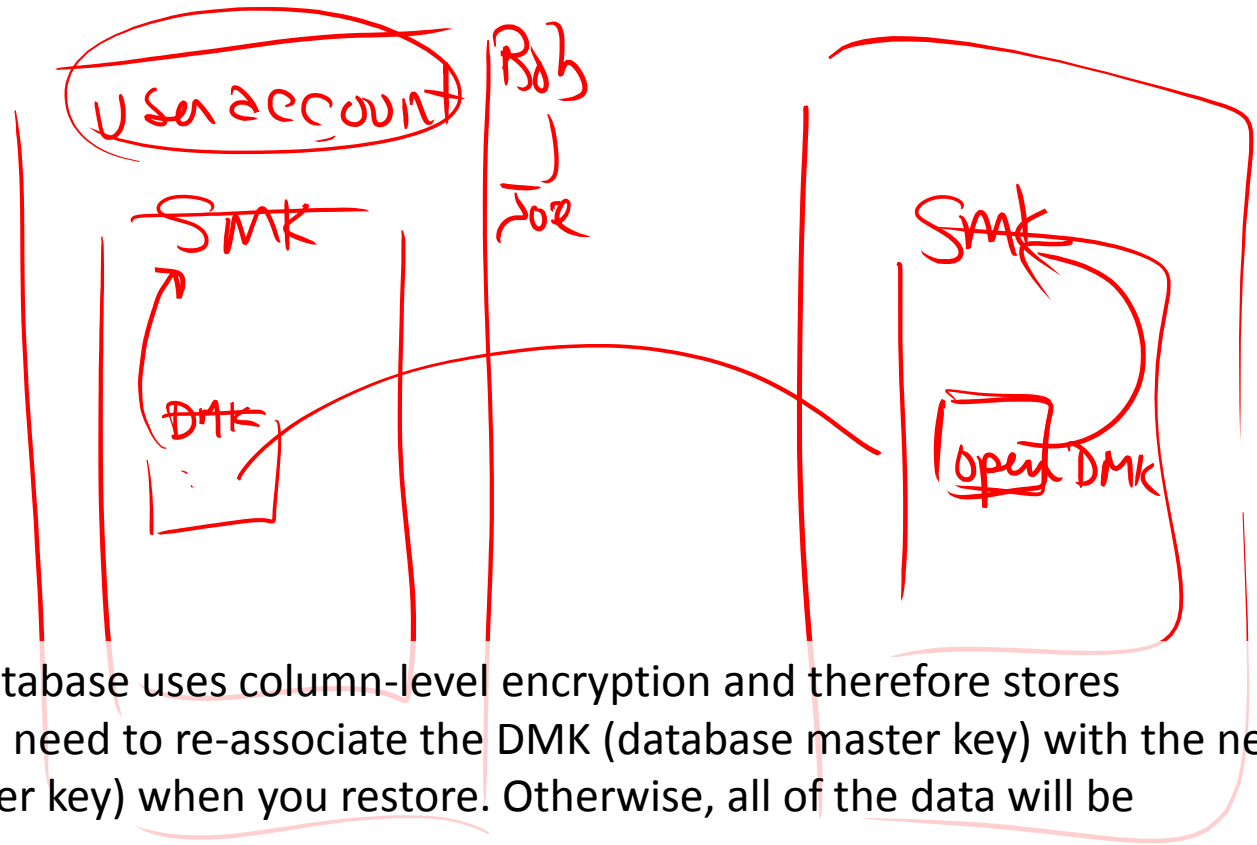
It's not a different filegroup but the end result (restructuring the filegroup) is the same!



Using WITH READ_WRITE_FILEGROUPS and the Simple recovery model:

For RDWs that are very large but largely RO, you can consider backing up the RW filegroups as a unit, backing up the RO filegroups after building the database. Then, in the event of a disaster you can restore ONLY the damaged RO file(s) OR ALL of the RWFGs (think of those as a unit – all must be at the same point in time because there's no log backup to restore to bring a subset UP to the same point in time....

DATA Encryption



The concept is this... if a database uses column-level encryption and therefore stores encrypted data, then you'll need to re-associate the DMK (database master key) with the new server's SMK (service master key) when you restore. Otherwise, all of the data will be inaccessible!

USE database;

go

-- Open the DMK with the SAME password used when created:

OPEN MASTER KEY DECRYPTION BY PASSWORD = 'original password';

go

-- Re-associate the DMK with the SMK of the new instance:

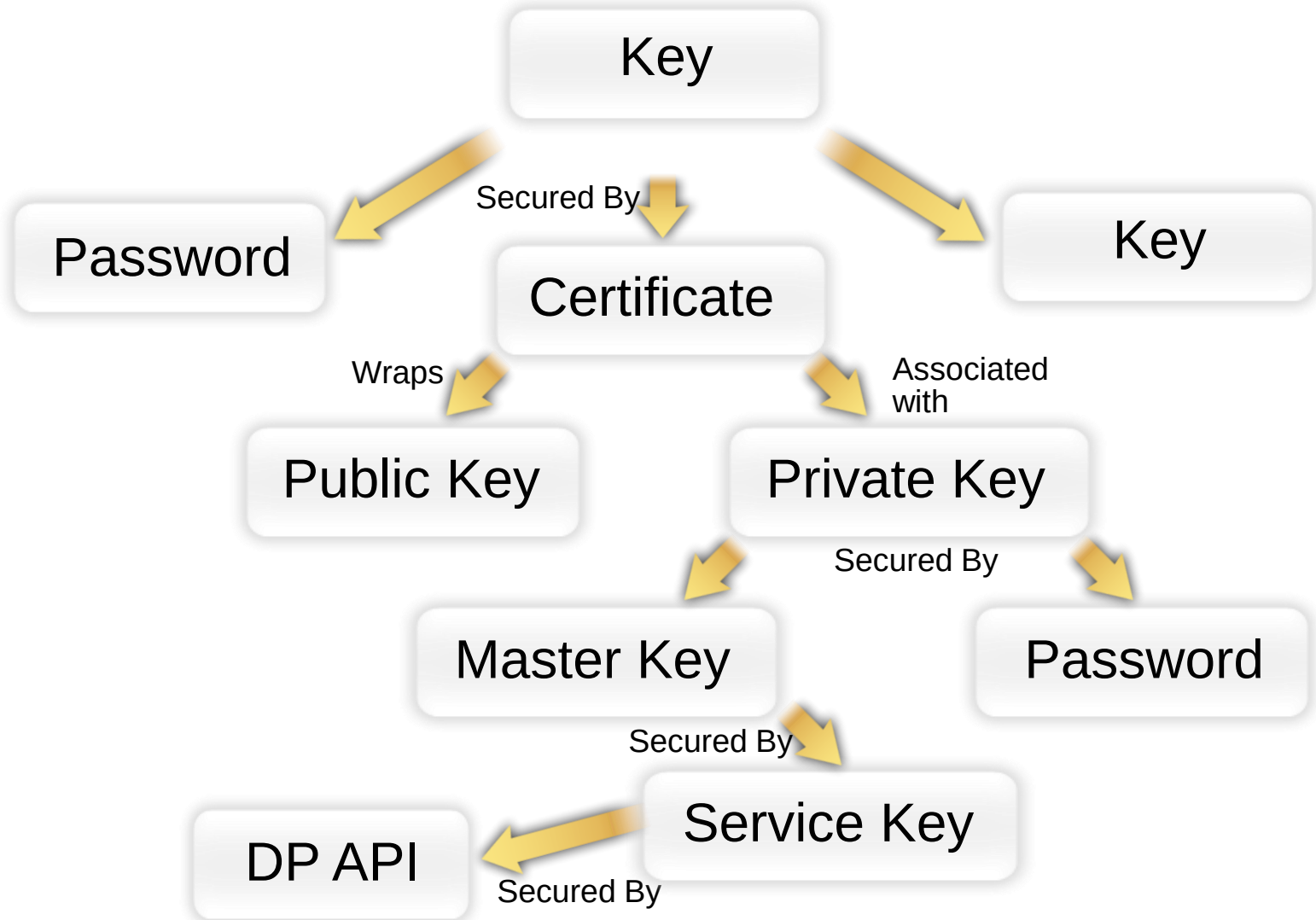
ALTER MASTER KEY ADD ENCRYPTION BY SERVICE MASTER KEY;

go

Encryption Hierarchy

- SQL Server 2005+ implements a framework to help protect encryption keys by using an encryption key hierarchy, as shown in the figure on this slide. In this hierarchy, each layer encrypts the layers that are below it.
- The Data Protection API (DPAPI) is at the top of this encryption key hierarchy. DPAPI is a pair of function calls that provide operating system–level data protection services to user and system processes. Because this API is part of the Microsoft Windows® operating system, applications can use DPAPI to encrypt data but do not have to use any cryptographic code other than the code that calls the DPAPI functions. DPAPI is a password-based data protection service. Therefore, DPAPI requires a password to encrypt the data.
- The SQL Server 2005 Service Master Key is a symmetric key that the **cryptGenKey** function automatically generates when an administrator installs SQL Server 2005. DPAPI uses the password of the account under which SQL Server 2005 runs to generate a key with which DPAPI encrypts the Service Master Key. Therefore, if an administrator changes the service account under which SQL Server 2005 runs, he or she must decrypt the Service Master Key by using the original credentials. Then, he or she must encrypt the Service Master Key by using the new credentials. We strongly recommend that administrators change the service account under which SQL Server 2005 runs only by using the SQL Server 2005 Computer Manager tool. This tool automatically performs the required steps to decrypt the Service Master Key, and to then re-encrypt the Service Master Key.
- The Service Master Key encrypts the Database Master Key. The Database Master Key is required in each database where an organization encrypts data. This key provides the same functionality as the Service Master Key. However, the functionality occurs at a database level instead of a SQL Server 2005 instance level.
- The Database Master Key is a symmetric key. It is not automatically created when a new SQL Server 2005 database is created. Therefore, the developer must explicitly create this key to implement encryption in a database.
- The Database Master Key encrypts the user keys that a developer creates. These user keys include certificates and asymmetric keys. In turn, certificates and asymmetric keys encrypt symmetric keys that the developer creates.
- **Note:** *A developer can also use certificates and asymmetric keys to encrypt data directly. However, certificates and asymmetric keys are best suited to encrypt only small amounts of data.*
- A developer can use symmetric keys to encrypt other symmetric keys that he or she creates or to encrypt data. This encryption key hierarchy is especially important to consider when the developer determines the type of key to use to encrypt newly created keys. The SQL Server 2005 encryption key framework gives a developer a large amount of flexibility to create a simple or complex encryption key hierarchy for each database that he or she creates. However, we recommend that a developer create as simple a key structure as his or her organization allows.

Encryption Hierarchy



SQLskills Immersion Event

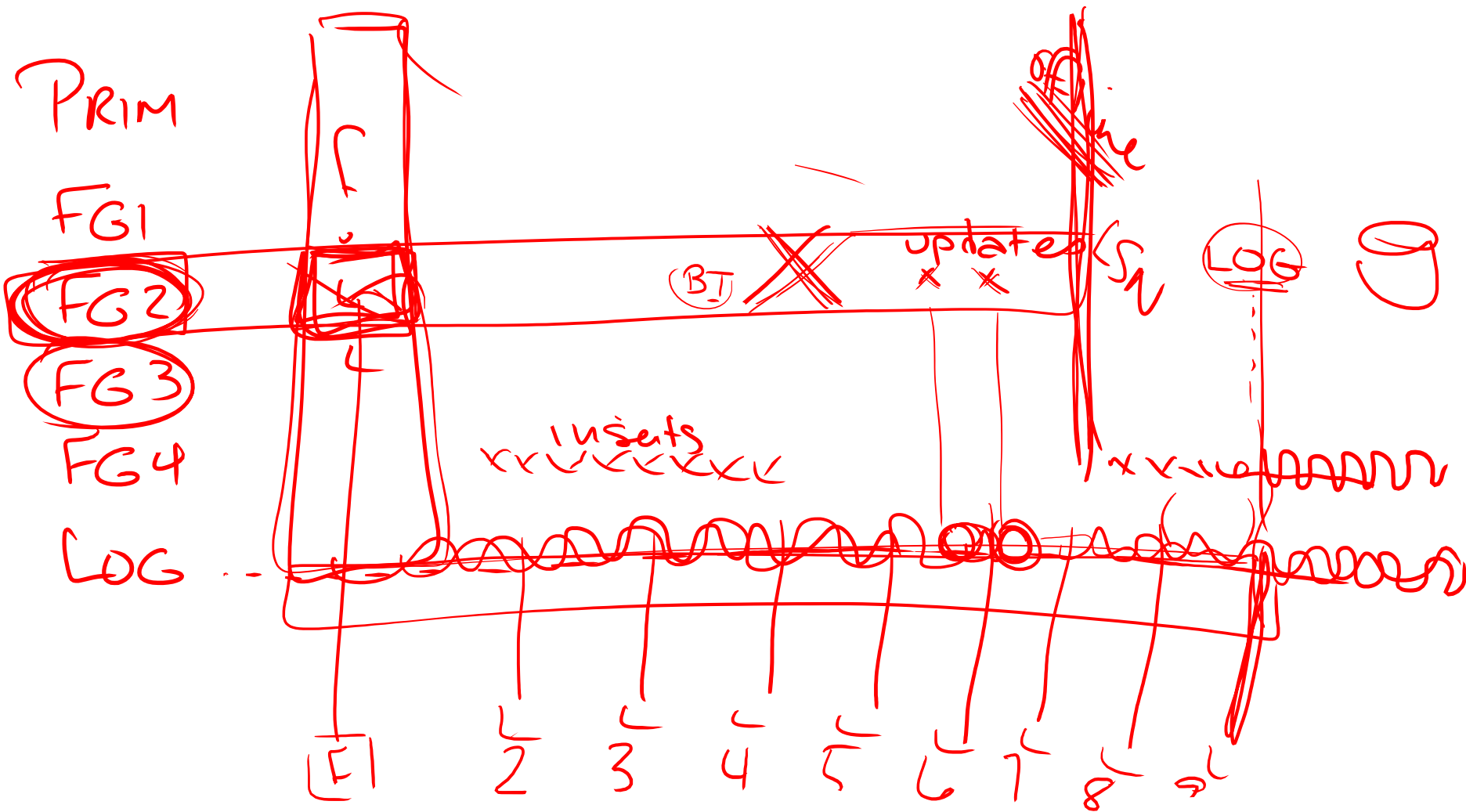
IEHADR: High Availability & Disaster Recovery

Module 5: Restore Internals and Procedures

Kimberly L. Tripp

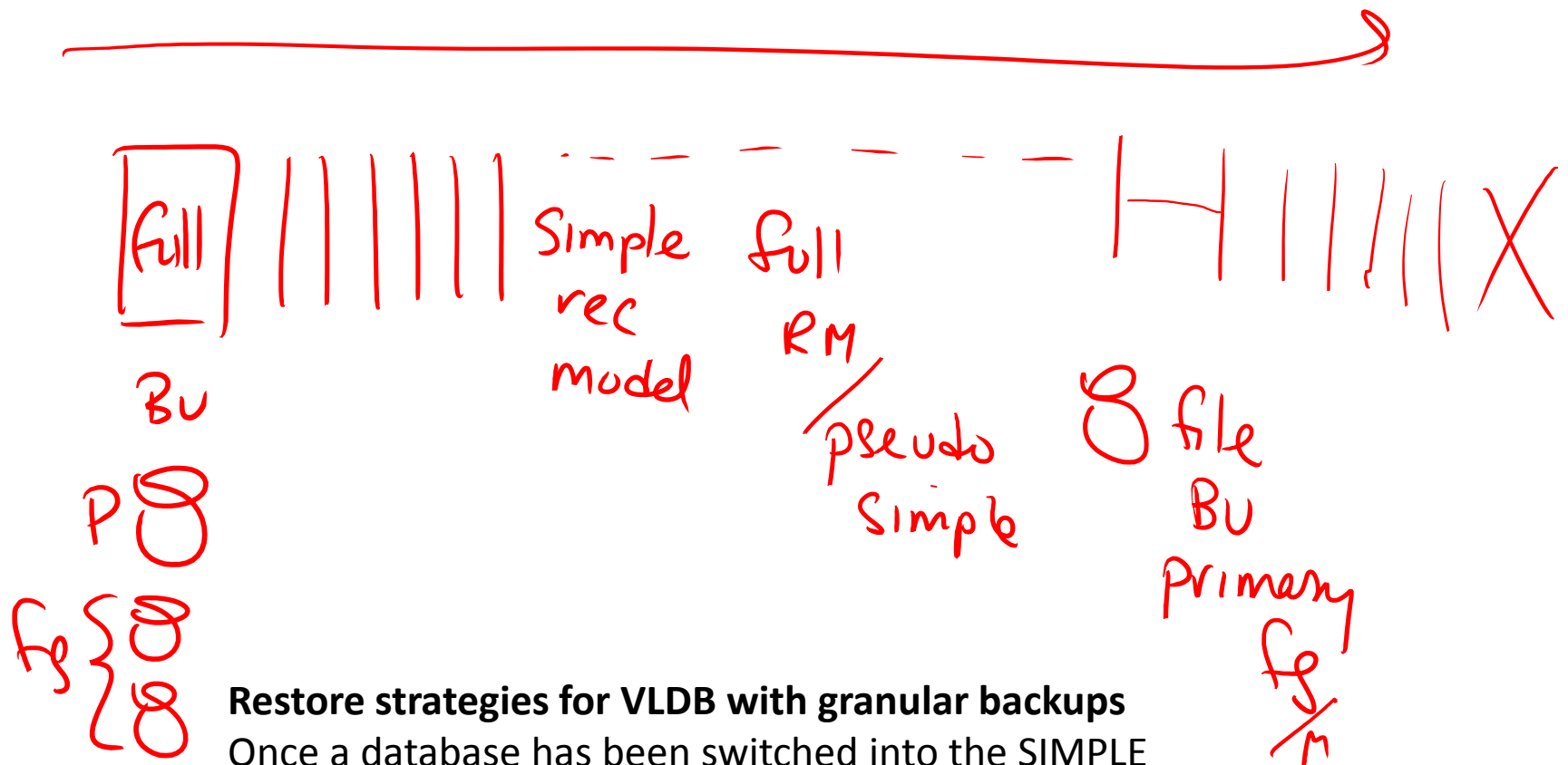
Kimberly@SQLskills.com





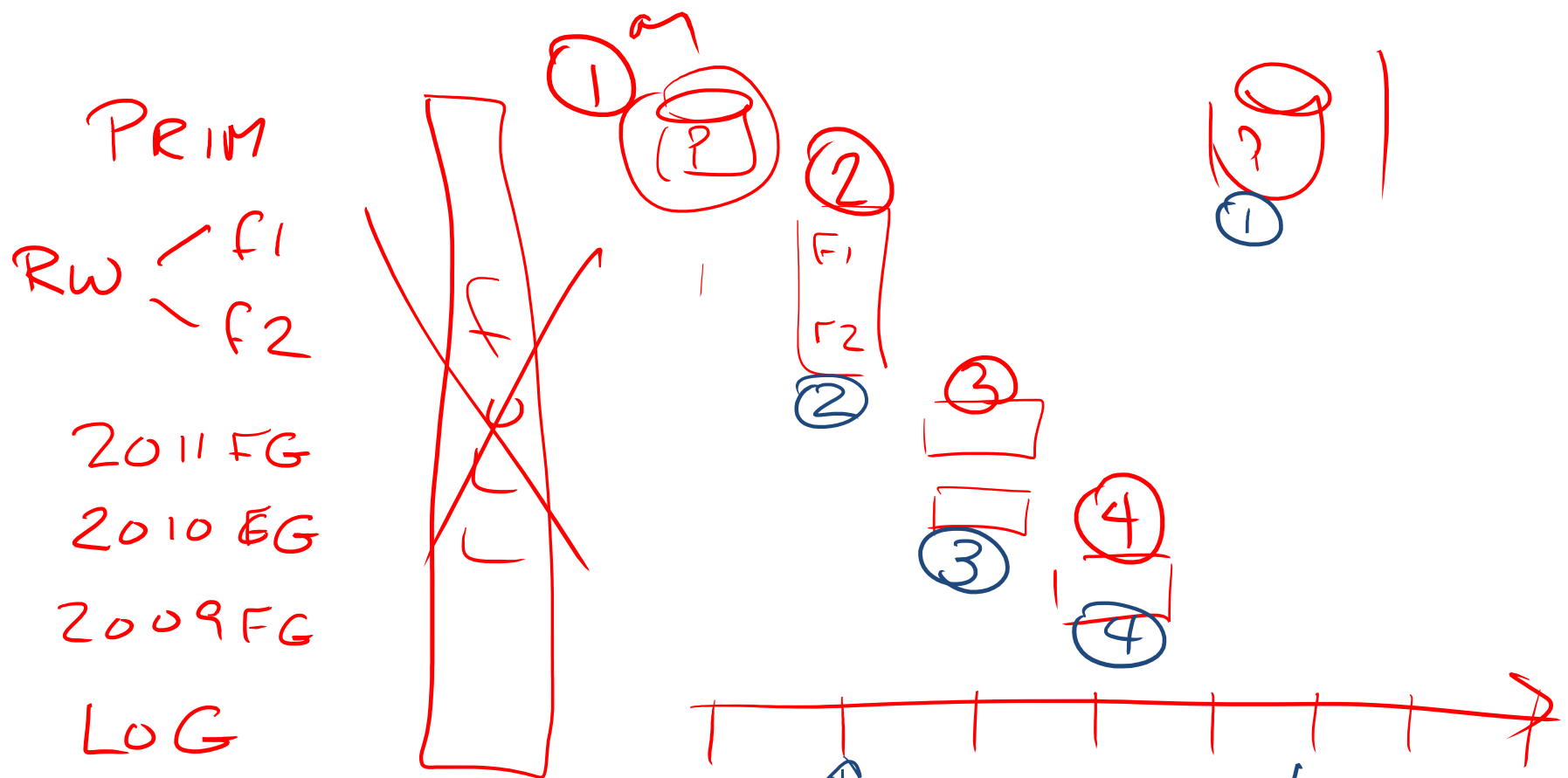
Taking a filegroup OFFLINE

This shows a multi-file database and a series of different types of backups. What was most important here is the understanding of what happens when a filegroup is taken OFFLINE. This is the point in time when SQL Server knows that no new transactions will occur. This is the point in time to which this filegroup must be restored (reconciled) in order for this filegroup to be brought online.



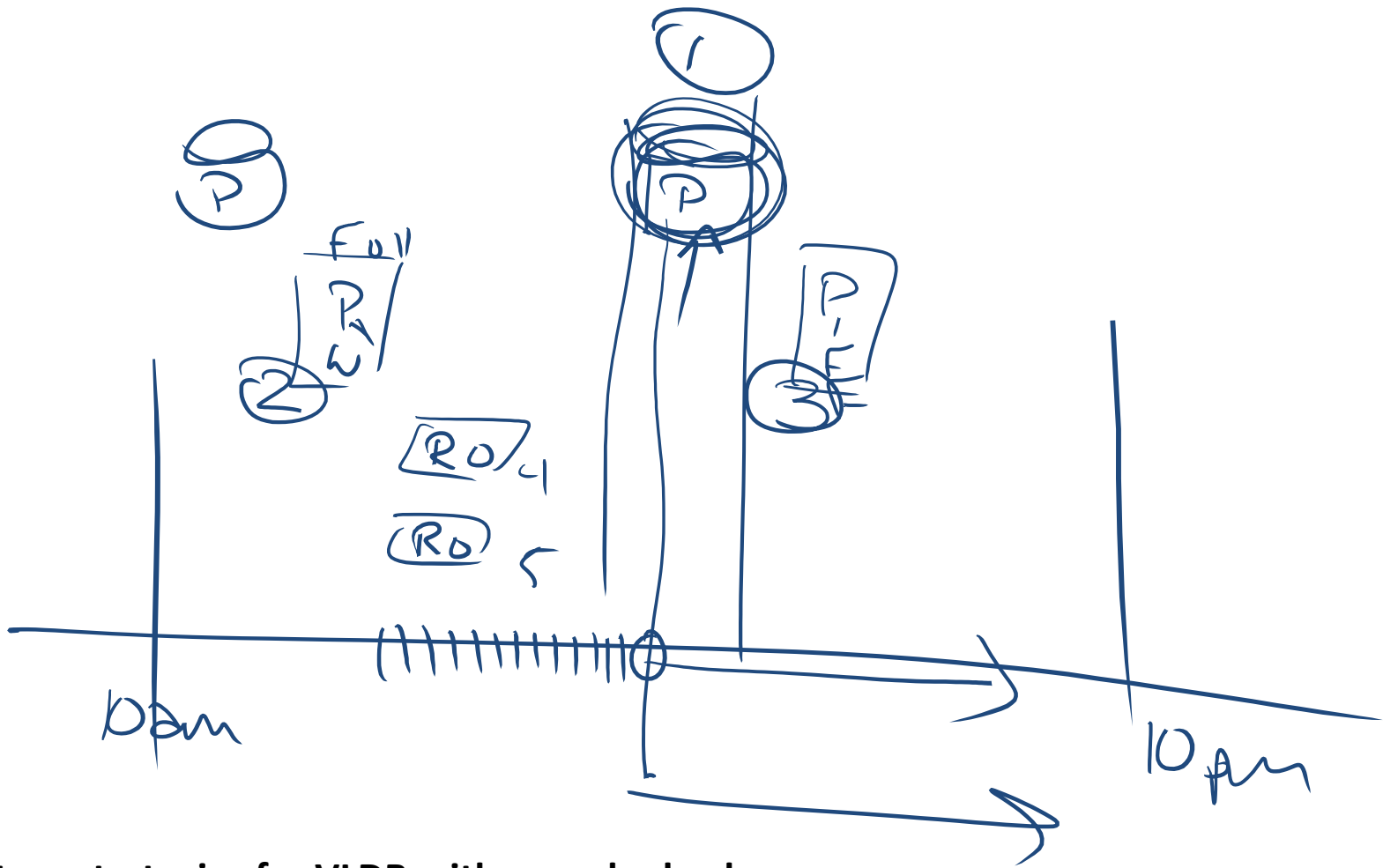
Restore strategies for VLDB with granular backups

Once a database has been switched into the SIMPLE recovery model then the continuity of the log has been broken. You do NOT need to do a full database backup to begin the process of log backups again. You only need to "begin" your backup strategy. If you perform any type of backup (full database, full filegroup, full file OR even a differential database, differential filegroup or differential file) then you have begun your strategy. However, you are still VERY vulnerable until ALL of your files have been backed up in some way. Always a good idea to do a database-level backup (full or differential) after breaking the continuity of the log.



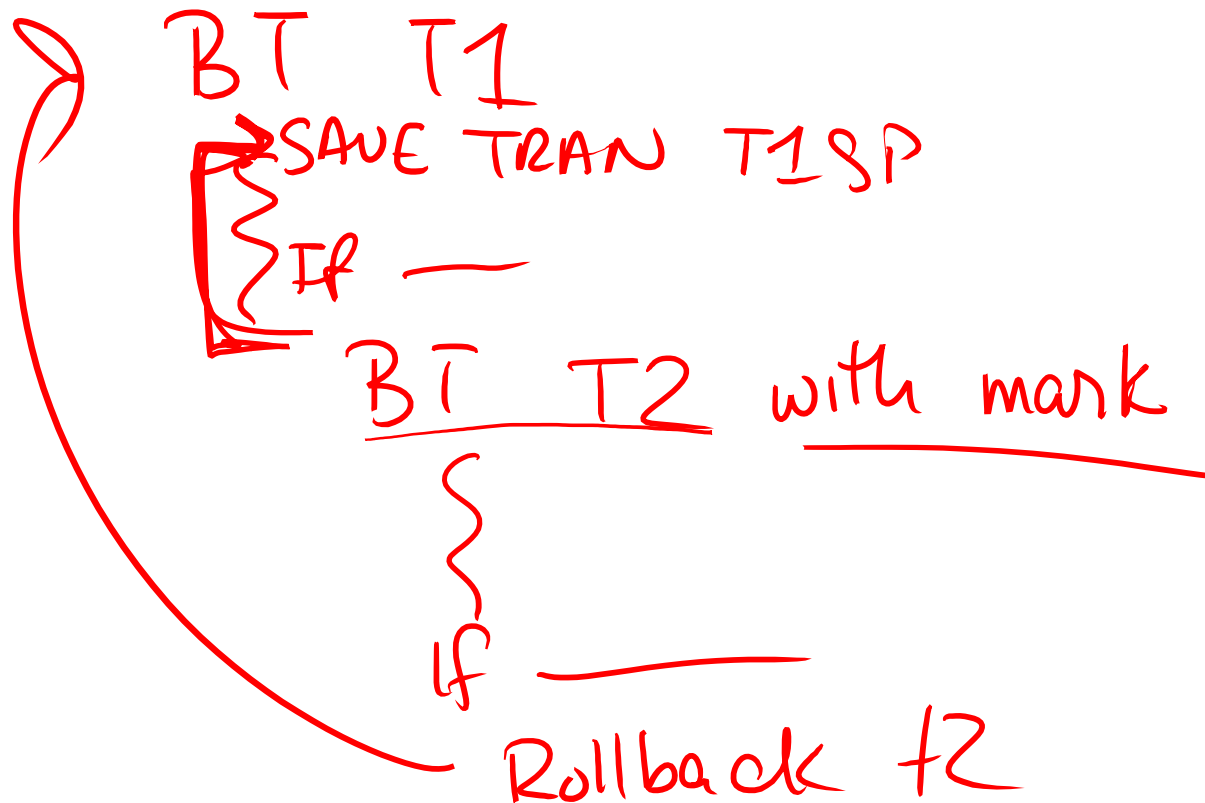
Restore strategies for VLDB with granular backups

To reconstruct a database from granular backups requires that ALL pieces are restored. However, since they were backed up at different times – a transaction log series that starts with the oldest filegroups minimum (or effective) LSN must be used. If you have a current primary and all of the filegroups (labeled in blue as 2, 3, 4) then you only need logs starting with the bottom blue arrow. If the blue backup #1 did not exist and you had to go back to the red #1 then you'd need all of the logs back to that point. The KEY point is that the database will be completely offline until the entire set of filegroups being restored are recovered to the SAME point in time. If you restore PARTIAL then you could bring it online more slowly/granularly.



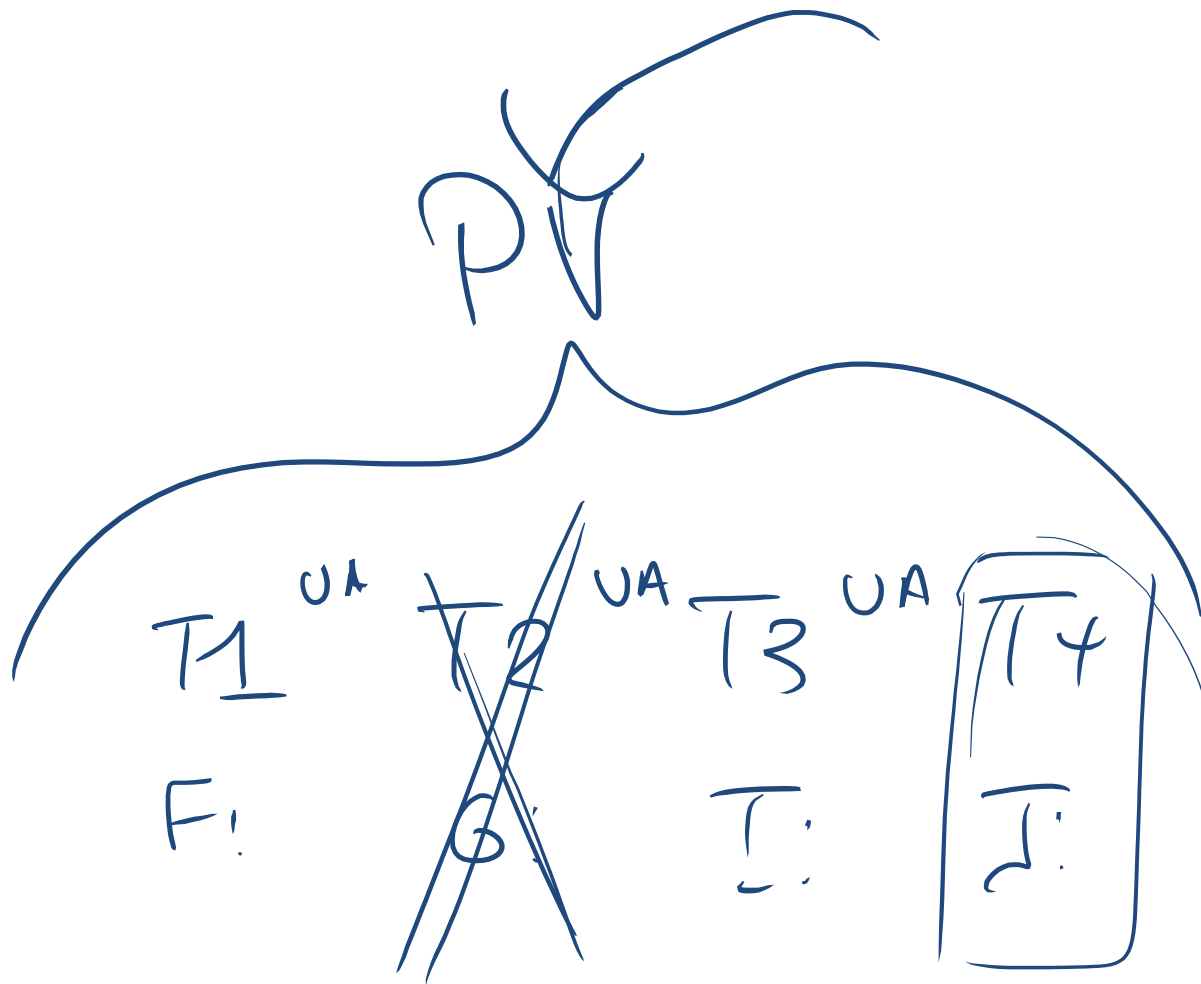
Restore strategies for VLDB with granular backups

This is a continuation of the last picture. If you have a primary filegroup backup (#1) then a RWFilegroup backup (#2) then you could restore that with logs to bring the entire database up to current (restoring the RO filegroups at virtually any time – possibly a lot later if you really just want to get the RW up to current). If you have a differential of the RWFilegroup then you could have restored it and then you only would have needed the logs from the backup time of the Primary filegroup.



Restore with STOPATMARK (or STOPBEFOREMARK)

A marked transaction has very few uses. You CANNOT rollback to a mark (remember, a transaction is always a SINGLE unit of work) and so what use do they serve? If they're nested – really none. However, if you do complete a marked transaction then you can use that as a restore point during recovery. Stop at mark – stops at and includes the completed/marked transaction and stop before mark – stops with all committed transactions prior to the mark.



Querying data when a filegroup is damaged

If you query a partitioned table when a filegroup is damaged then you'll only receive an error if you (try to) select from the damaged partition. This is also NOW true of a partitioned view. It was not always true (in 2005 a view that was damaged would fail). This was fixed in 2008+

SQLskills Immersion Event

IEHADR: High Availability & Disaster Recovery

Other Discussions

Kimberly L. Tripp

Kimberly@SQLskills.com



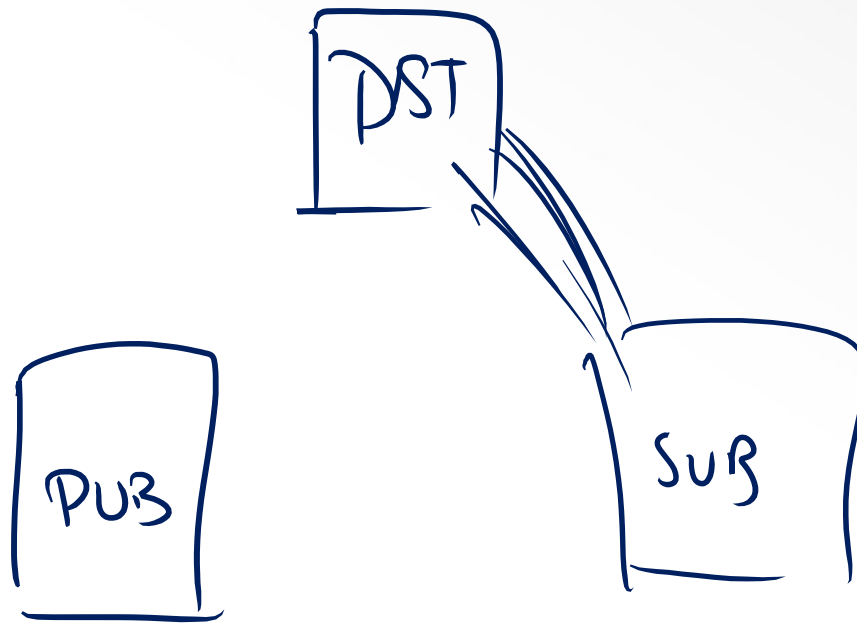
Plan Invalidation

■ Versions prior to 2012

- If database option: **auto_update_stats** is ON
 - Updating statistics causes plan invalidation
- If database option: **auto_update_stats** is OFF
 - Updating statistics does NOT cause plan invalidation
- If you manually update statistics and have set auto_update_statistics to OFF, add sp_recompile @tname to your stats scripts (remembering SCH_M problems)
- For more info: Erin Stellato's links about auto update stats/plan invalidation:
 - Statistics and Recompilations: <http://erinstellato.com/2012/01/statistics-recompilations/>
 - Statistics and Recompilations, Part II: <http://erinstellato.com/2012/02/statistics-recompilations-part-ii/>

■ SQL Server 2012+

- Plan invalidation is NOT affected by the setting of auto update stats
- Plan invalidation does NOT occur if data has not changed
 - Only for UPDATE STATISTICS
 - An index rebuild will update statistics as well as cause plan invalidation, even if no data has changed thus increasing the importance for *"only when data has changed"* logic in maintenance routines.



Read-committed, snapshot
versioning

This was from a side discussion on versioning and where it might be appropriate (great for the subscriber). I just wrote a blog post on this here:

Inconsistent analysis in read committed using locking

<http://www.sqlskills.com/blogs/kimberly/inconsistent-analysis-in-read-committed-using-locking/>

B/R

Online

non det. time of
restore

Only get "used"
Space

Rolling/online/upgrade

Detach

Offline

attach is exactly
when taken offline

potentially lots of
free space to move

Update stats?

update stats

TDE? neither will compress well

This was from a side discussion on backup/restore vs. detach/attach. The most important two items (IMO) for why B/R is better – you have a backup ☺ and you can do this without taking the database offline. Be careful using detach/attach as it might be a one way street.

This is a pretty good pros/cons list: <http://strictlysql.blogspot.com/2011/10/backuprestore-vs-detach-attach.html> and Aaron Bertrand also did an interesting post here: http://sqlblog.com/blogs/aaron_bertrand/archive/2011/01/21/downgrading-a-database-you-can-t-get-there-from-here.aspx.

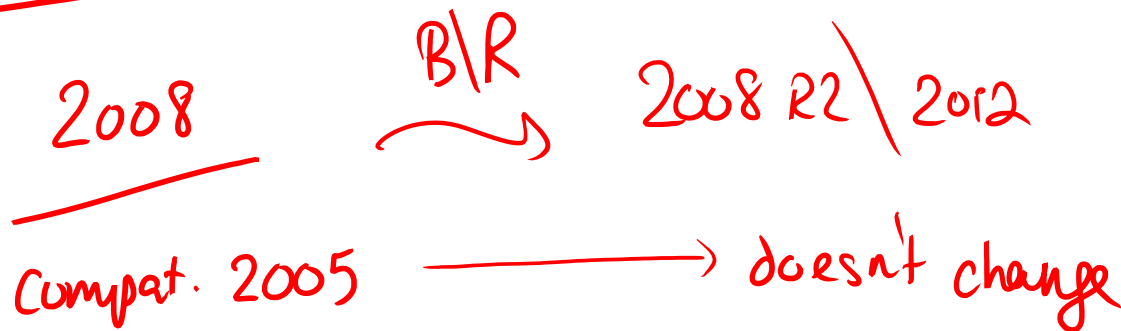
Backward Compatibility Levels

The following drawings discussed our tangent about backward compatibility levels.

The key points:

- The compat mode of the database when backed up is what will get restored. If the database compatibility mode is set to 6.5 (for example) on a 7.0 server and then backed up – the database's compat mode is 6.5. If this backup (a 7.0 database backup) is restored on SQL Server 2000 (8.0) then it will be restored and the database itself will be converted to an 8.0 database. However, the database compatibility mode will still be 6.5.
- Compat modes do not impact features, they impact parsing and keyword use. They are meant as a way to allow you to upgrade to the new release while having time to “fix” code – to remove keywords, etc.
- Setting compat modes can be done through the UI or the syntax. Often, the UI is limited in the modes available. For example, even on SQL Server 2005 (9.0) you can still set 6.5 and 6.5 – but not through the UI. Always check the syntax for setting compat modes:
 - Use `ALTER DATABASE` to set compatibility mode
 - For backward compatibility, you can still use `sp_dbcmtlevel`.

Compat



Upgrade advisor

Upgrade assistant

trace trace

=

Backup/restore is supported “up-level” only and only 2 versions. However, 2008 R2 is not counted in the two version restriction. For example, you can restore from:

- 2000 to 2005/2008/2008R2
- 2005 to 2008/2008R2/2012

But, not from 2000 to 2012.

The compat mode of the backed up database is the compat mode that’s restored (unless no longer supported).

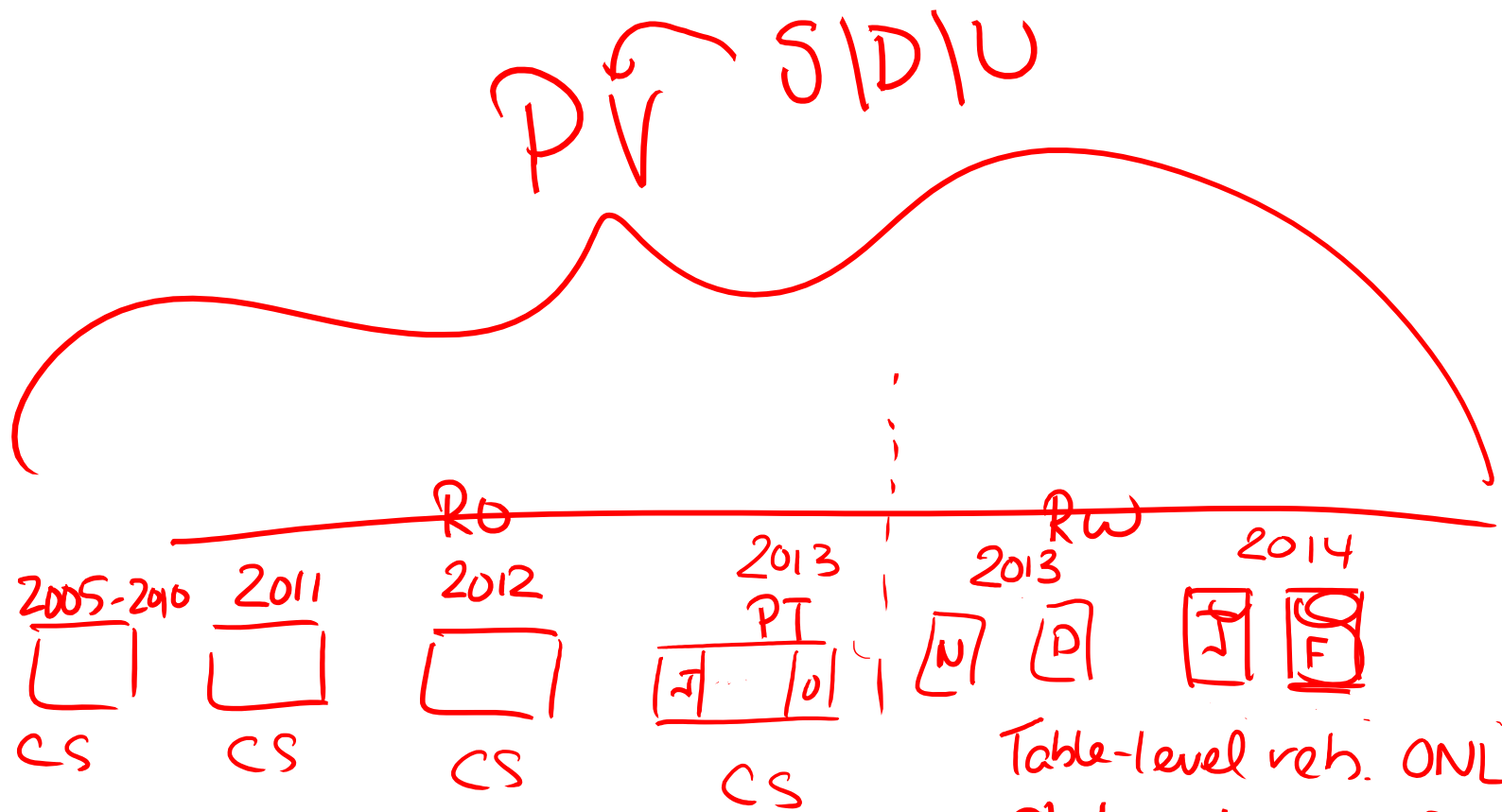
From SQL 2012 BOL: When a database is upgraded to SQL Server 2012 from any earlier version of SQL Server, the database retains its existing compatibility level **if it is at least 90**. Upgrading a database with a compatibility level below 90 sets the database to compatibility level 90. <http://technet.microsoft.com/en-us/library/bb510680.aspx>

Tools to help you upgrade: ?

Microsoft Upgrade Advisor: (found within the feature pack) <http://www.microsoft.com/en-us/download/details.aspx?id=29065>

Upgrade Assistant Tool for SQL Server 2012 (en-US):

<http://social.technet.microsoft.com/wiki/contents/articles/2558.upgrade-assistant-tool-for-sql-server-2012-en-us.aspx>



A discussion around performance and availability came up wrt partitioning. This diagram was a refresh from IE1. Large tables don't perform well as a single table. For best performance, break large tables down into smaller pieces. Older data can be put on filegroups that are read-only. Newer (more critical) data can be put on faster, more reliable/redundant storage. Use the PV for selects, updates, and deletes but use SPs with application directed inserts (if you want to use identity columns).

Table-level reh. ONLINE &
Stats upd more freq
more accurate

SP

Inserts App Dir DSE
getdate
insert tnvon -

Backup/Restore Trace Flags

- **TF 3004 & 3014:** Show internal backup/restore steps (start/end times) for every phase. When combined with 3605 it will write them to the errorlog
 - DBCC TRACEON (3004);
 - DBCC TRACEON (3014);
 - DBCC TRACEON (3605);
- **TF 3213:** Expose buffer and transfer details
 - How It Works: SQL Server Backup Buffer Exchange (a VDI Focus)
<http://blogs.msdn.com/psssql/archive/2008/01/28/how-it-works-sql-server-backup-buffer-exchange-a-vdi-focus.aspx>
 - How It Works: How does SQL Server Backup and Restore select transfer sizes
<http://blogs.msdn.com/psssql/archive/2008/02/06/how-it-works-how-does-sql-server-backup-and-restore-select-transfer-sizes.aspx>
- **TF 3023:** How to enable the CHECKSUM option if backup utilities do not expose the option (<http://support.microsoft.com/kb/2656988>)
 - DBCC TRACEON (3023, -1)
 - The (-1) switches on the trace flag globally rather than just for your session

Backup/Restore Trace Flags

- **TF 3226:** Suppress successful backup messages from writing to the errorlog
 - [Fed up with BACKUP success messages bloating your error logs?](#) (Paul)
 - [When is too much success a bad thing?](#) (Kevin Farlee, SQL Server Dev Team)
- **TF 3231 and 3031:** Disable log clearing (NO_LOG and TRUNCATE_ONLY will not clear the log)
 - [Understanding backups and log-related Trace Flags in SQL Server 2000/2005 and 2008](#) (Kimberly)
- **Other useful posts around backup/restore**
 - [Search Engine Q&A #24: Why can't the transaction log use instant initialization?](#) (Paul)
 - [Manage the suspect_pages Table \(SQL Server\)](#) (msdn)

SQLskills Immersion Event

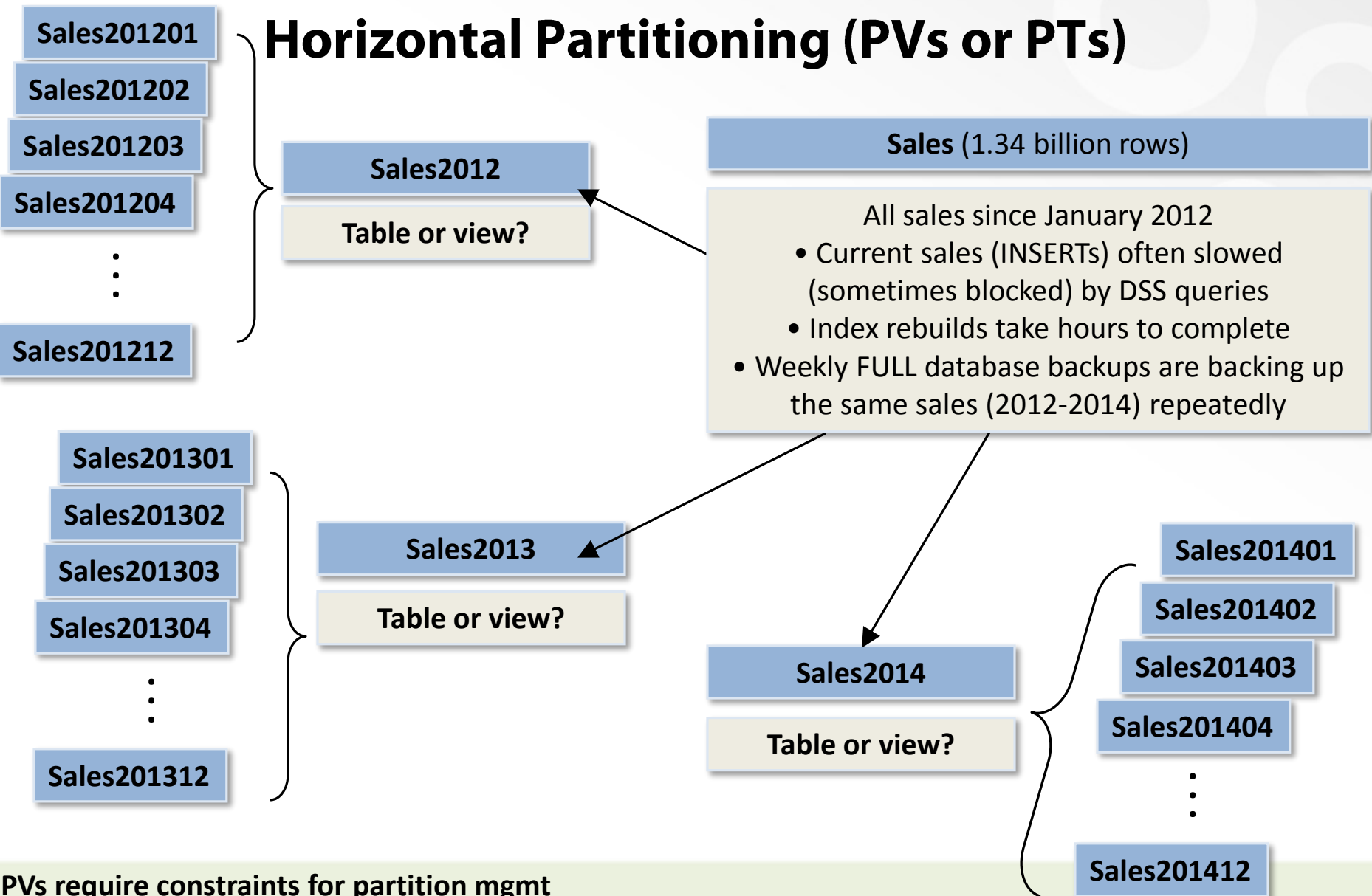
VLDBs: VLTs and Partitioning Strategies

**Discussions / drawings around VLDB
and “partitioning” large data sets (VLTs)**

Kimberly L. Tripp
Kimberly@SQLskills.com



Horizontal Partitioning (PVs or PTs)



PVs require constraints for partition mgmt
`CHECK ([SalesDate] >= '20140101'
AND [SalesDate] < '20140201')`

PTs require a partition function and a partition scheme for partition mgmt

Partitioning: PVs vs. PTs

Partitioned views

- Any edition
- Lots of tables to administer
 - Must create/drop indexes on all base tables
 - Can have different indexes
 - Harder for the optimizer to optimize with so many indexes
 - Must verify business logic so that there are no gaps or overlapping values
 - Each table has [potentially] better statistics as the tables are smaller
- Can rebuild any of the tables ONLINE (if using EE)
- Can support multiple constraints on one or more columns

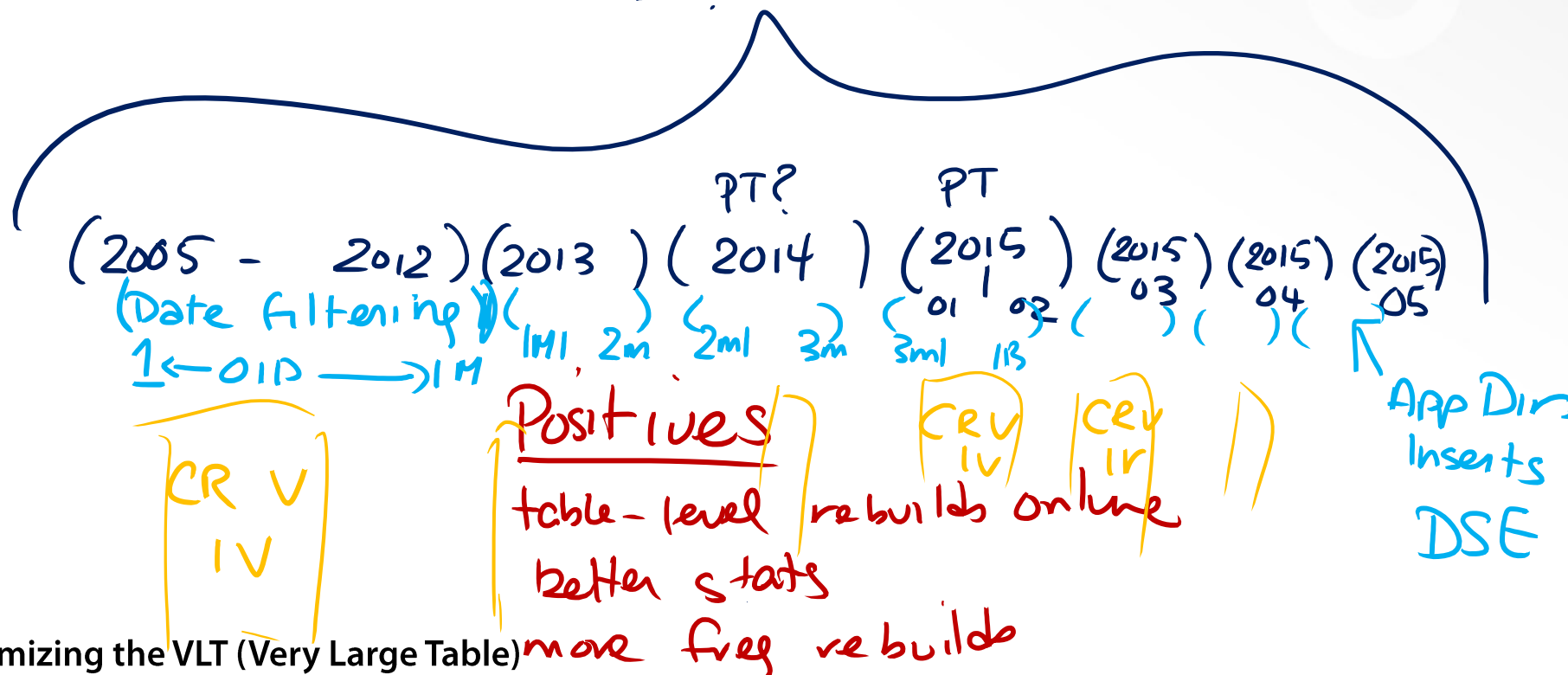
Partitioned tables

- Enterprise Edition only
- Only 1 table to administer
 - Only 1 table to create/drop indexes
 - All partitions have same indexes (which is easier for the optimizer to optimize)
 - Can create different indexes with filtered indexes
 - No possibility of errors (or gaps or overlapping values)
 - Table-level statistics can be less accurate for very large tables when there's a lot of skew to the data
 - "Incremental stats" do NOT fix this!
- Partition-level rebuilds are offline but can rebuild the ENTIRE table online (not desirable)
- Can only support partitioning over a single column

VLT

need part. col
as leading col of PK

U P V Great for S, D, U
=?

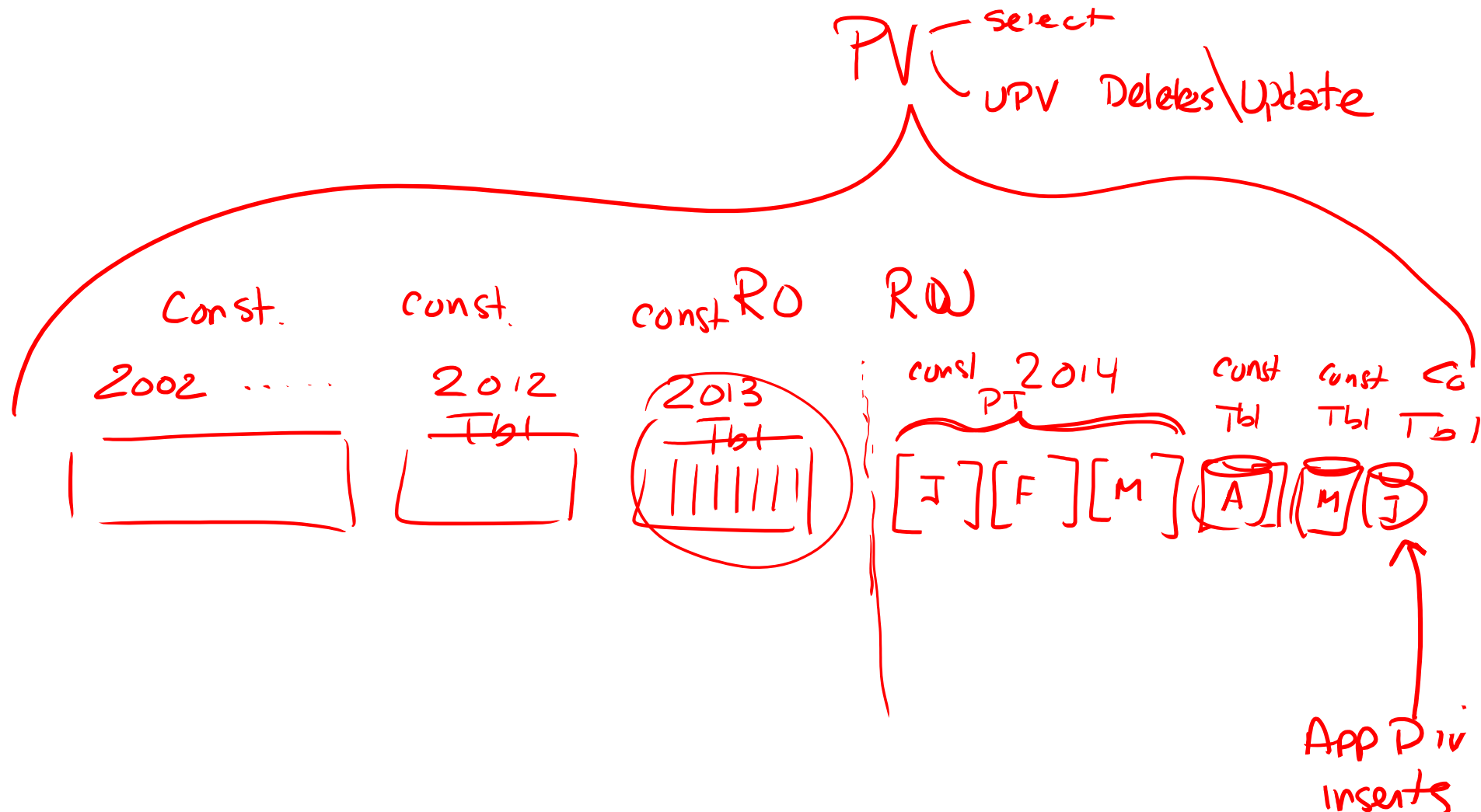


Optimizing the VLT (Very Large Table)

A very large table has many "problems"... to reduce those – don't have just ONE VLT – have smaller tables that are unioned (using UNION ALL) into a view. If you also have restrictive constraints (CHECK constraints) across all of the base tables this combination is called Partitioned Views. You get numerous benefits with this architecture including: better control / manageability, better statistics on each table, online operations, and the reduction of some operations all together (on historical data).

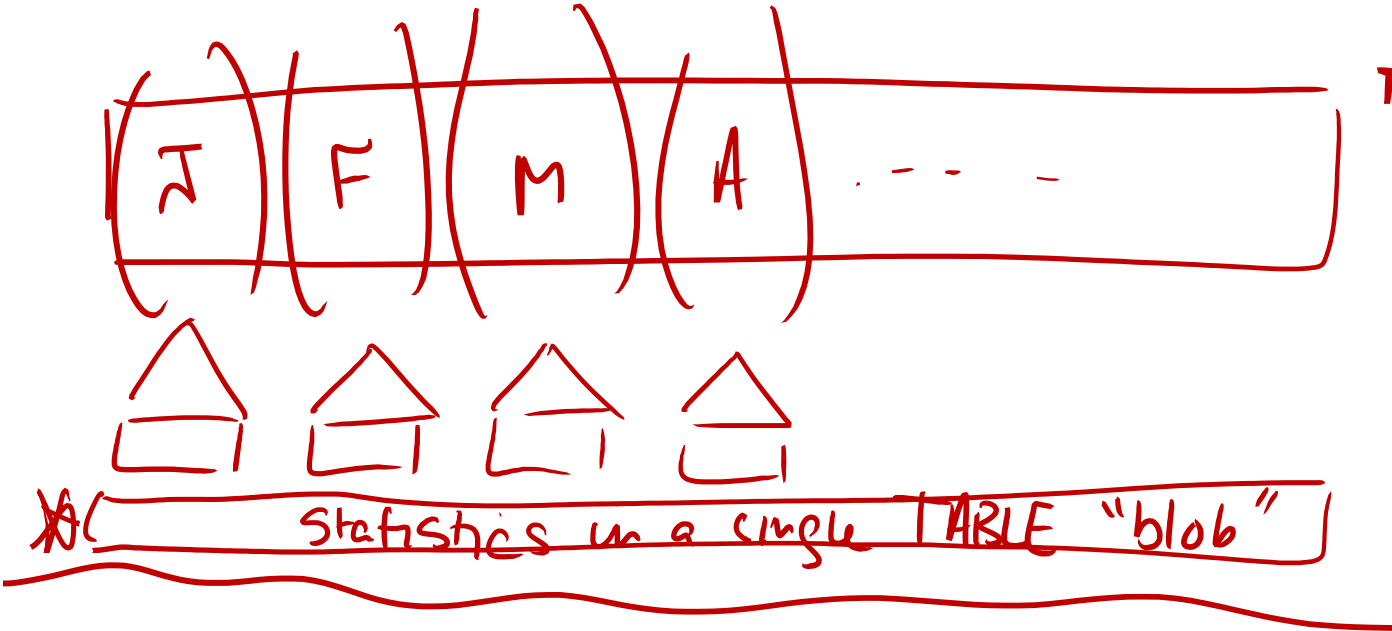
Architecting a “layered” approach to your table design

A better option is to separate the RW from the RO and put the RW in separate tables. Then, you can do online rebuilds at the table level. How do you deal with queries – put a partitioned view over them!

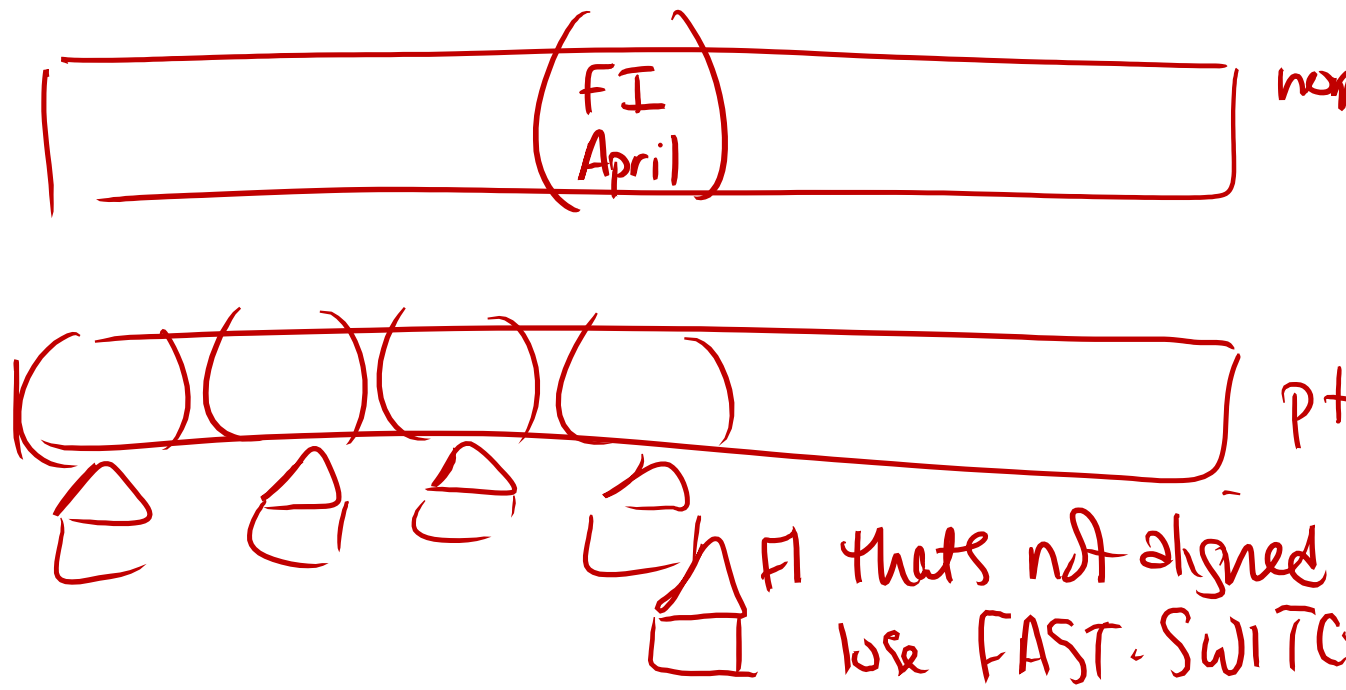


Filtering v. Partitioning

One of the frustrating things about a large table is that – even when partitioned, the statistics cover the entire table (leading to inefficiencies).



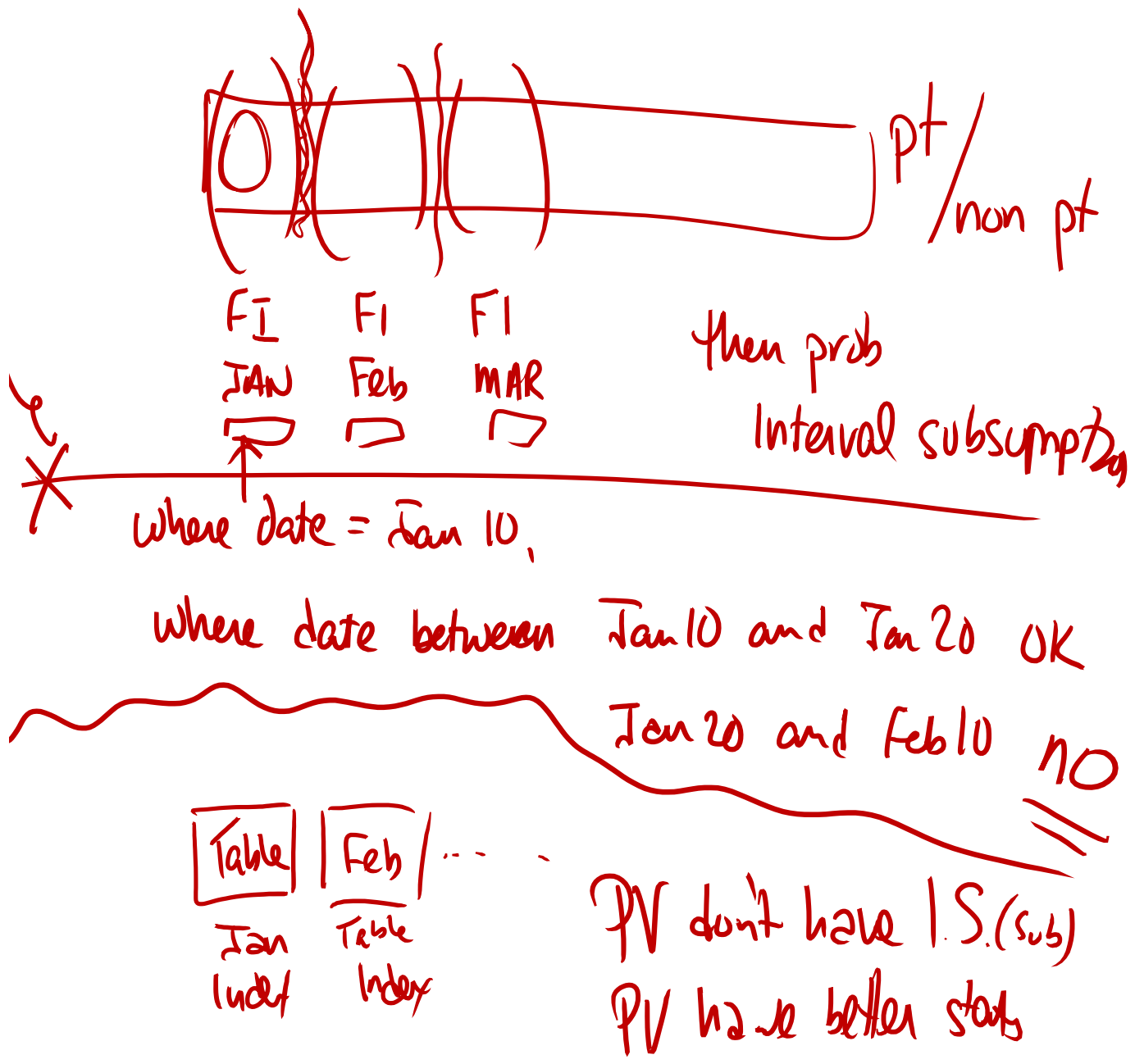
If you create a filtered index over just a single partition you get better statistics and an isolated index (just for the queries that need it) but then you lose the ability to do fast switching.

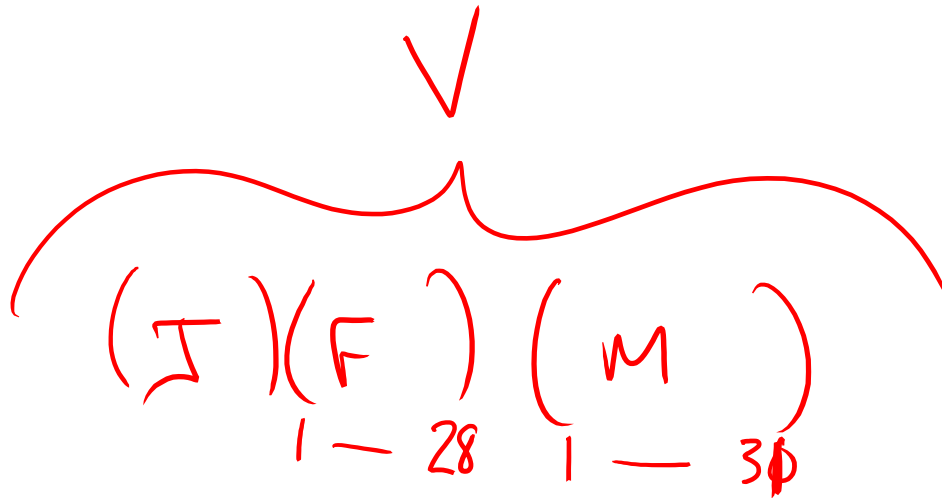


Filtering v. Partitioning

You might think that using JUST a filtered index approach would be better but then there's the interval subsumption problem.

Architecting the RIGHT solution and breaking down a VLT into smaller tables can be ideal. Partitioned Views (PVs) do NOT have interval subsumption problems. And, PVs have better statistics...



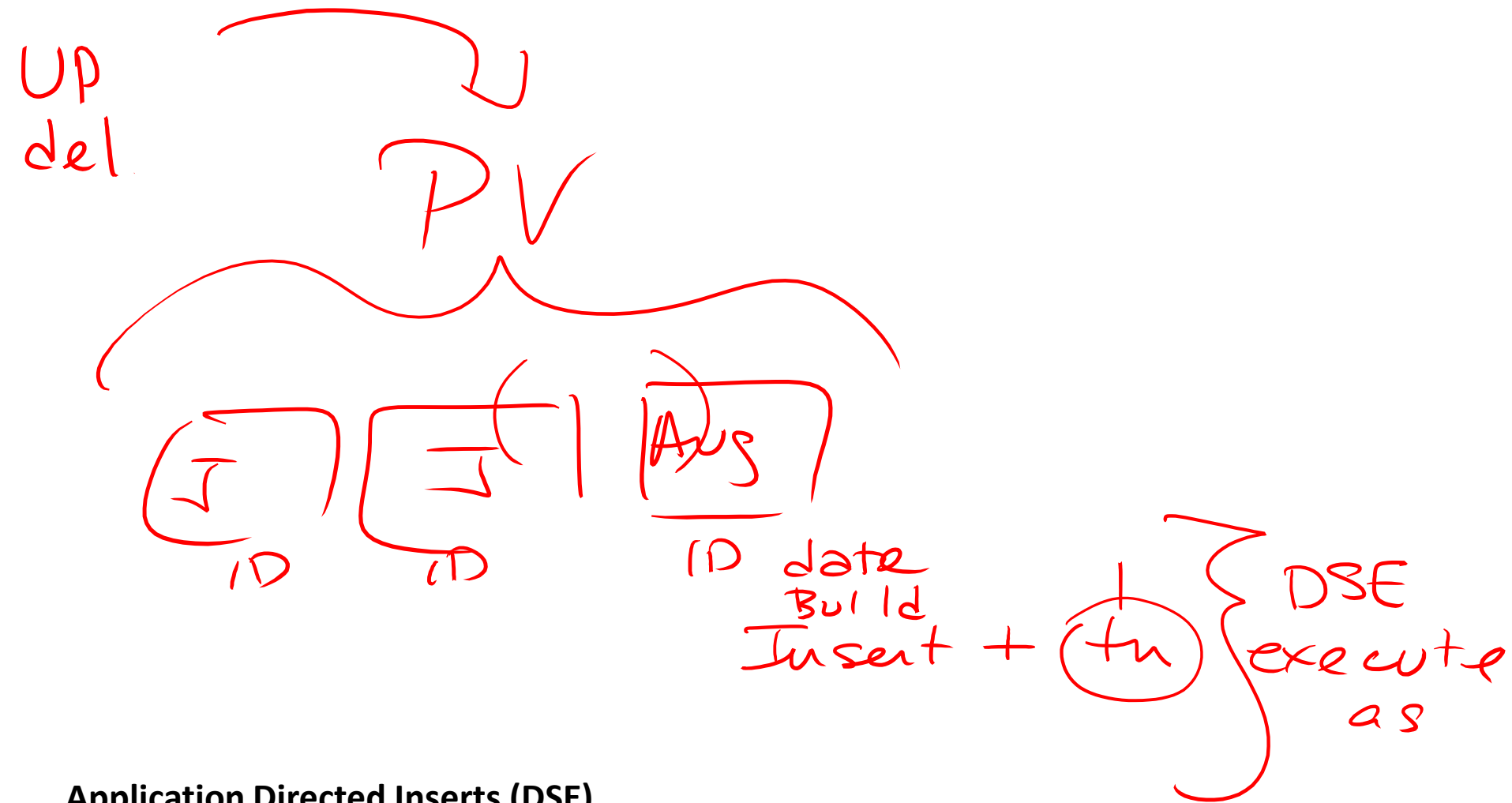


Partitioned Views and business logic

Be careful with partitioned views... there's nothing that's going to test your constraints such that there are no gaps... As a result, if you miss a day (for example Feb 29, 2008) then inserts into the view will fail because there's no table that exists to store that date.

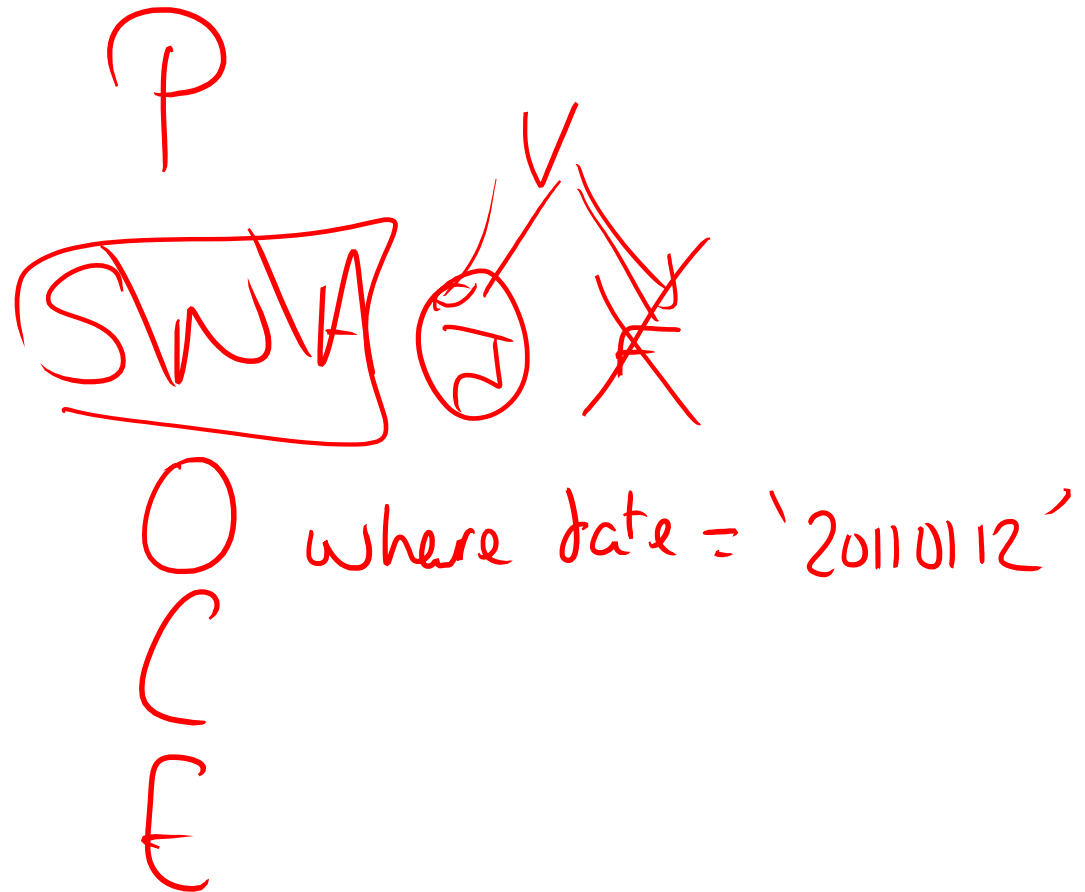
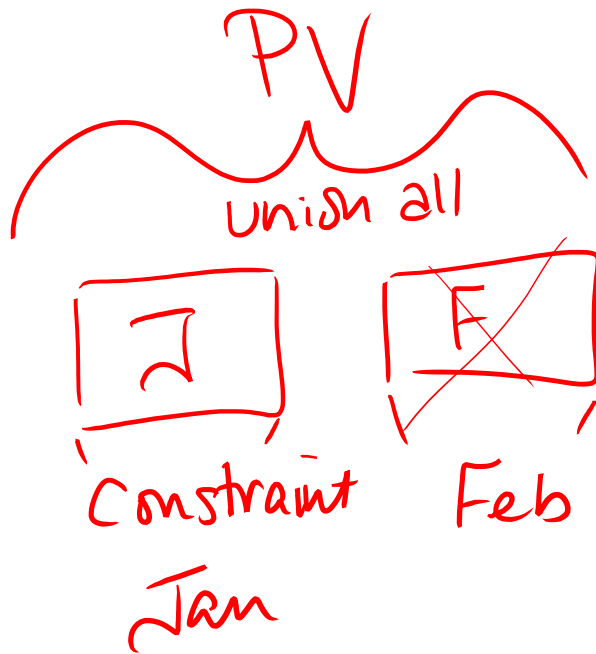
Alternatively, using partitioned tables – there's no way to have a gap or overlapping ranges.

But, there's a lot more to PVs and PTs (this isn't enough to choose one over the other) so this is just a bit more info to add to the list!



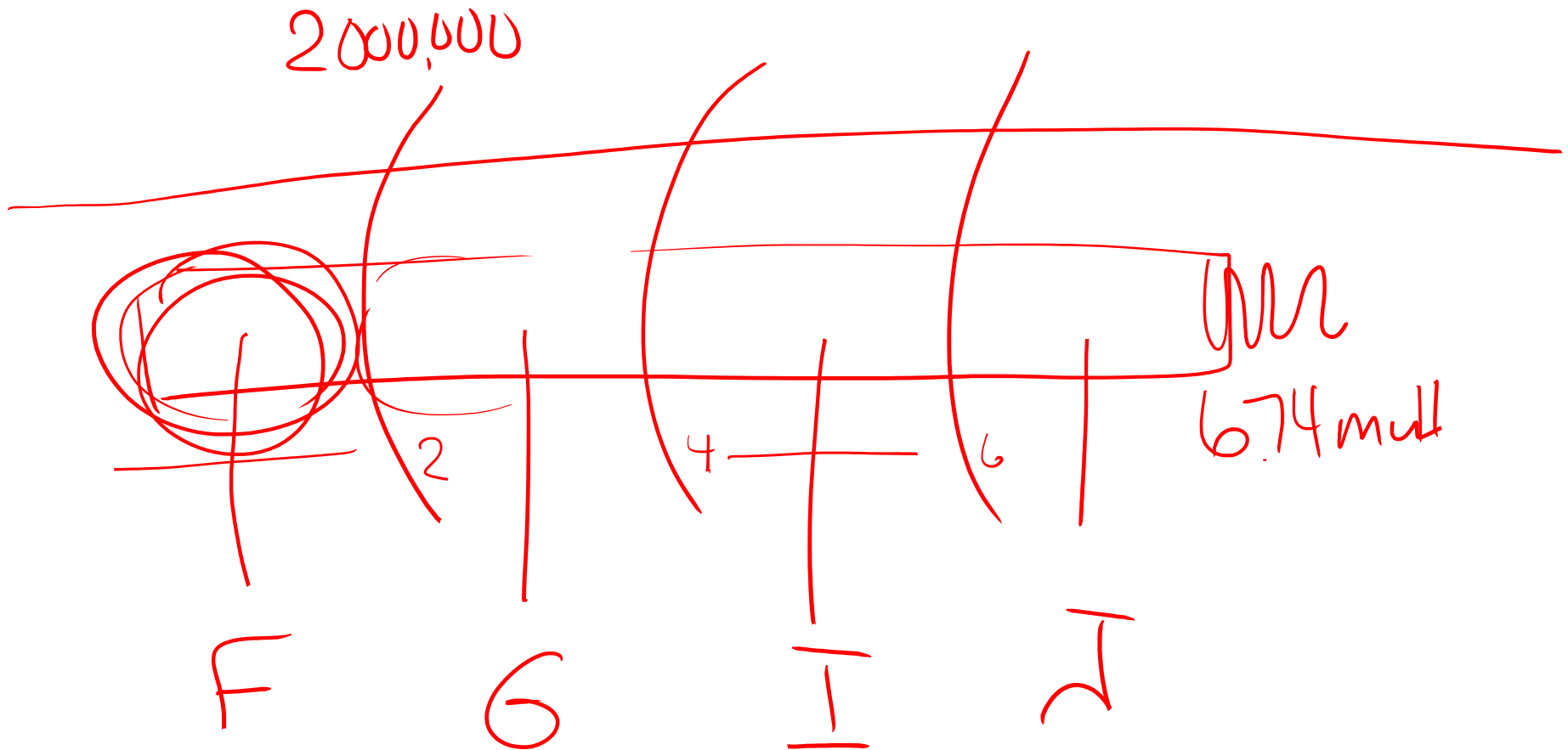
Application Directed Inserts (DSE)

Even with updateable partitioned views, I usually use application directed inserts. If you're concerned about dynamic string execution check out the Little Bobby Tables blog post: <http://www.sqlskills.com/BLOGS/KIMBERLY/post/Little-Bobby-Tables-SQL-Injection-and-EXECUTE-AS.aspx>



Partition Elimination

Constraints are validated during optimization. SQL Server is able – when querying through a view – to generate the query tree and then “prune” the tree. This partition elimination removes any of the redundant tables from access. All of this is as long as the constraint is trusted (or, should I say as long as it’s NOT untrusted. ☺)



Partitioning on integers (rather than date)

Another example of a partitioned table with 3 boundary points: 2million, 4 million and 6 million. It's a RIGHT-based partition function so the rows that match the boundary point will be to the RIGHT side. Specifically this translates into:

In partition 1: all rows less than 2 million

In partition 2: all rows equal to or greater than 2 million and less than 4 million

In partition 3: all rows equal to or greater than 4 million and less than 6 million

In partition 4: all rows equal to or greater than 6 million



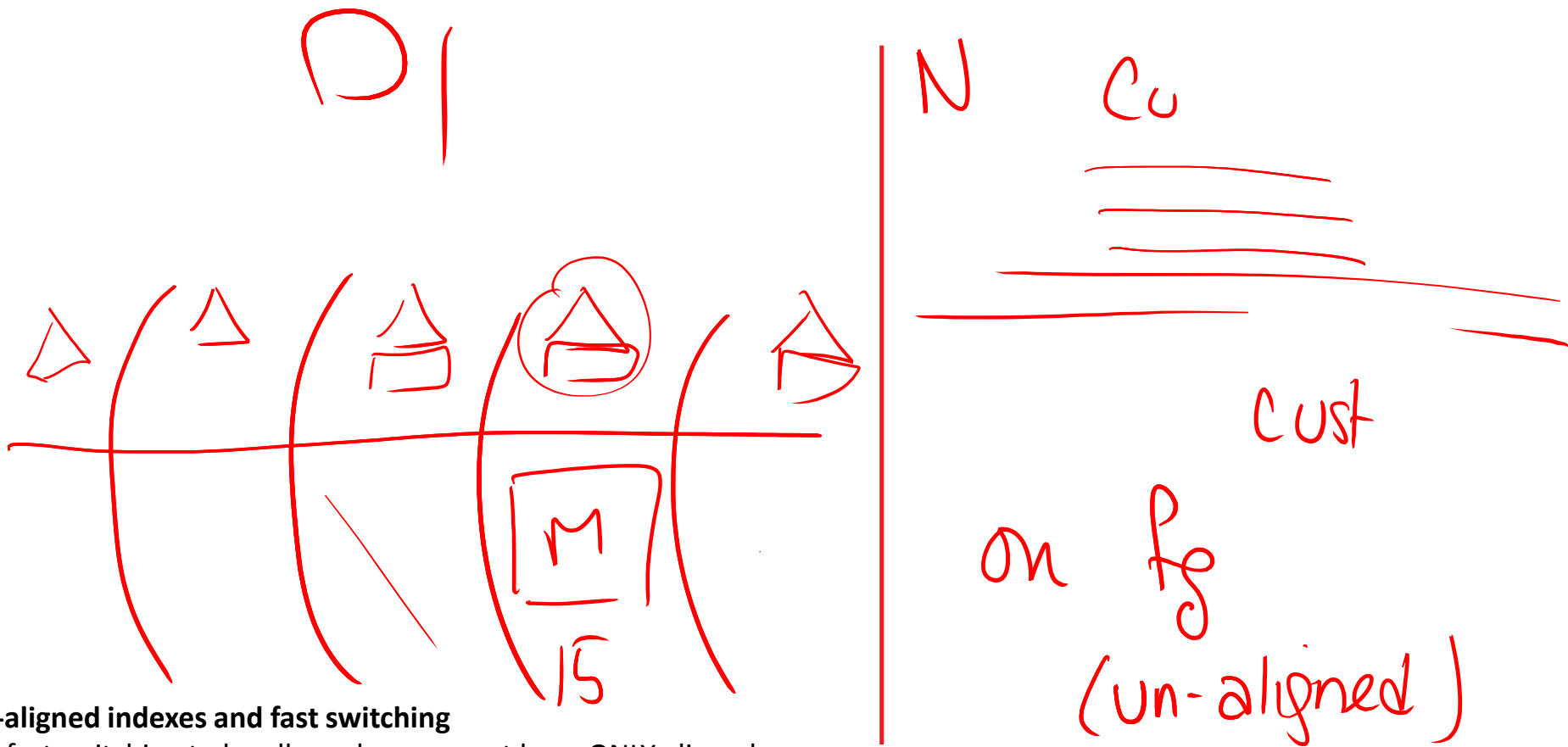
Aligned Indexes

Indexes can be aligned to the same partition scheme:

- Either by creating them ON SCHEME(col) or
- Accepting the default behavior during creation. Nonclustered indexes default to being created on the same scheme as the clustered index – unless they are an UNIQUE index. If the index is unique then the partitioning column must [explicitly] be part of the key.

Indexes can be unaligned

- The index has all of the nonclustered index data for the table in one leaf structure
- Fast-switching is NOT allowed if unaligned indexes exist



Un-aligned indexes and fast switching

For fast switching to be allowed – you must have ONLY aligned indexes. For an index to be aligned – there are rules. If the index is going to be unique (and aligned) then it MUST include the partitioning key. Indexes will default to being created on the same scheme as the table (they will default to being aligned – and therefore have all of these requirements).

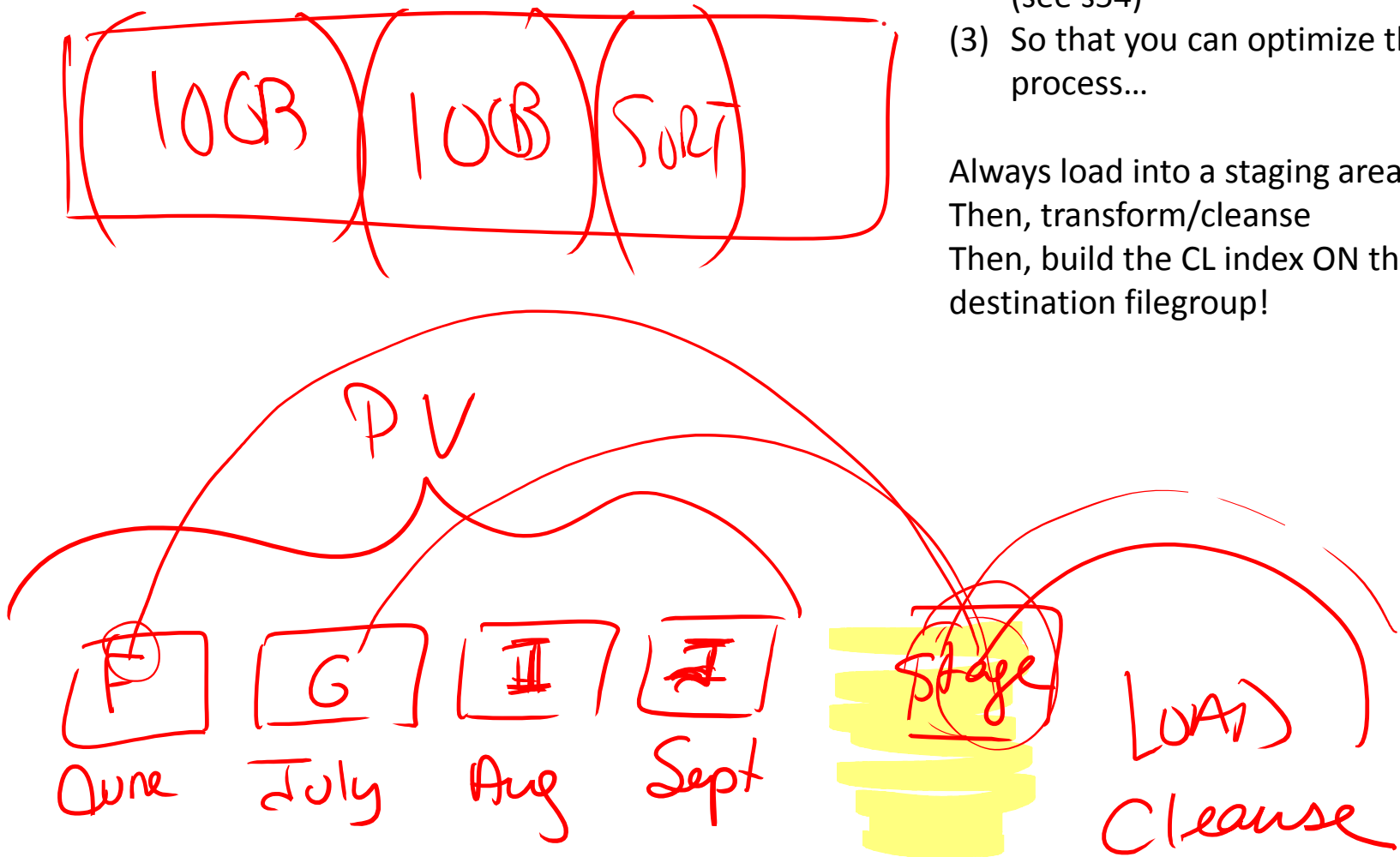
However, **if you're not interested in fast switching** – then you can have un-aligned indexes. In this case you create these indexes on a specific filegroup (or on a different scheme). These un-aligned indexes can be unique and do NOT have to include the partitioning column.

Staging Data

Why do you need a staging area?

- (1) So that you don't need to overallocate space within destination filegroups
- (2) So that you don't have to shrink (see s54)
- (3) So that you can optimize the process...

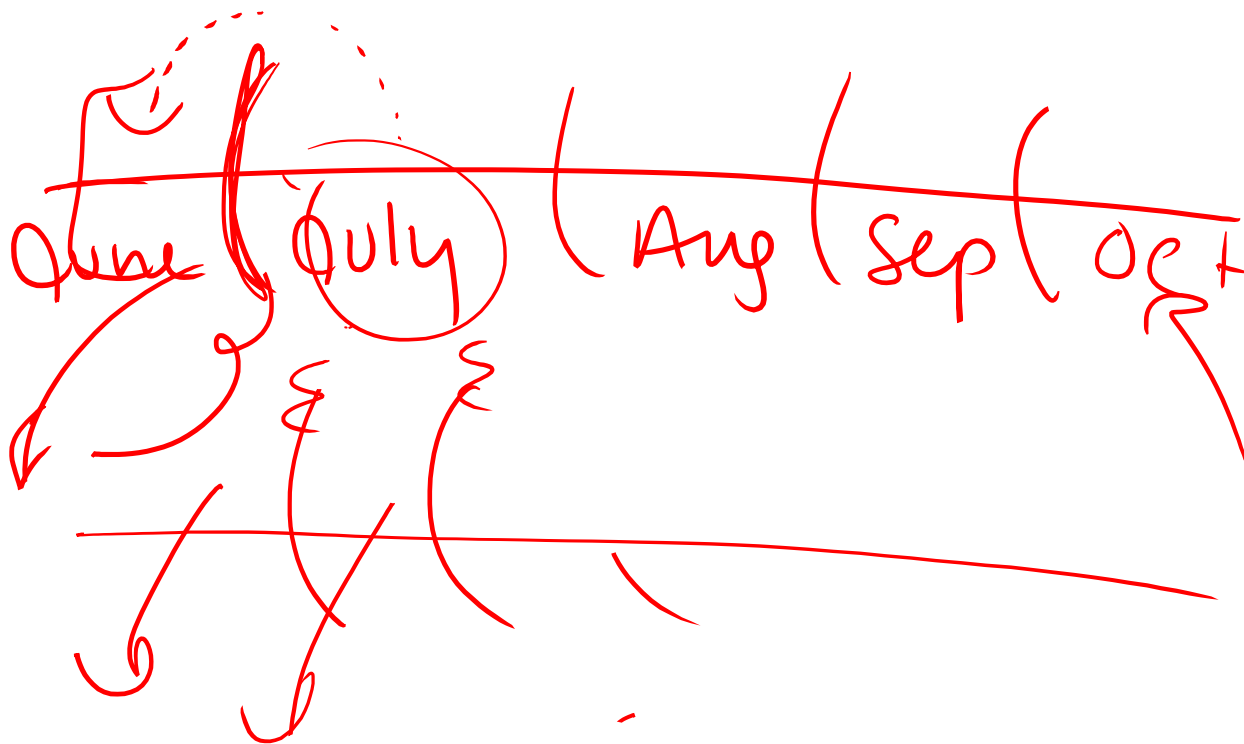
Always load into a staging area first.
Then, transform/cleanse
Then, build the CL index ON the destination filegroup!





Merging the right boundary

If the MERGE process is slow – you may have merged a boundary point that was NOT empty.



The most common case of this is when you start with a non-empty first partition using RIGHT-based partitioning. With RIGHT-based partitioning you should not MERGE until you have TWO empty partitions that surround the emptied boundary. Then, you can MERGE (top diagram).

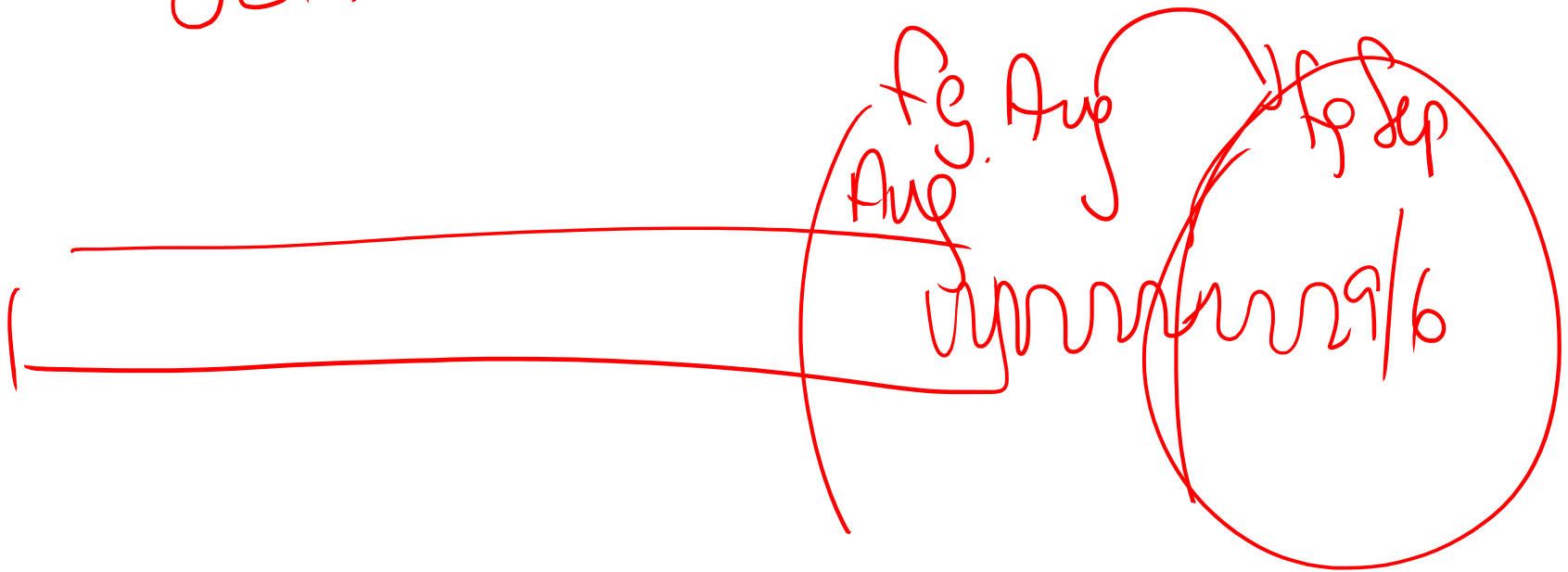
Prior to SQL Server 2014 – rebuilding the active partition – OFFLINE ☹️

With PTs – it's likely that your last partition (often, the current data) will be active. This is also where it's most likely that you'll have fragmentation. If you want to rebuild ONLY that last partition – you'll need take it offline to do it.

Or you need to rebuild the ENTIRE table online...

THIS IS FIXED IN SQL Server 2014

OLTP



Criteria

Always 1 yr.

What if Q3 04

merge
tiny

E

fs1

0701
2003
Q3

after switch

△

fs2

1001
2003
Q4

△

fs3

0101
2004
Q1

△

fs4

0401
2004
Q2

fsx

20040701
2004 Q3

Trust?

Constraint

IN

Heap

load Staging
Cleanse (CI?)

ready for prod

CR CL on fsx

CR NC on fsx

CR V/IV

CR CS

2008+ →

2012 →

OUT
CR 2003Q3 on fs1
CR CL
CR NC
CR V/IV
CR CS

drop table?

The prior slide showed the sliding window scenario – but we went through it slowly:

(1) Preparing the table into which we'll switch OUT our old partition

Review all of the slides for the requirements here but the key point is that this “staging” table MUST be on the same filegroup as the partition you are going to switch out.

(2) Preparing the table for our data load and what will become our new partition to switch IN

Again, be sure to review all of the slides for the requirements here but one thing you need to make sure of is that there's a TRUSTED constraint on this table before you switch in.

(3) Change the partitioned table to support the new filegroup and data range

Always set the NEXT USED filegroup with ALTER SCHEME

Once you have the correct filegroup specified then ALTER FUNCTION...SPLIT

- Now, you're ready

(4) and (5) can be in any order. SWITCH IN the new partition and SWITCH OUT the old.

(6) Clean up

Merge the boundary point but ONLY if it's empty (the next picture will remind you of what happens when you don't MERGE an empty boundary point)

Backup the filegroup where the partition resides that you just switched out. OR, drop the table.

① merge

③ Scheme next used fsg
split @ 20040701

E E

fsg1	fsg2	fsg3	fsg4	fsg5	fsg6
July 03	Oct 03	Jan 04	Apr 04		Switch IN

Switch
OUT

④

CR T 200307Q1

=====

on fsg2

①

CR CL

CR NC

2008 → CR V/IV

②

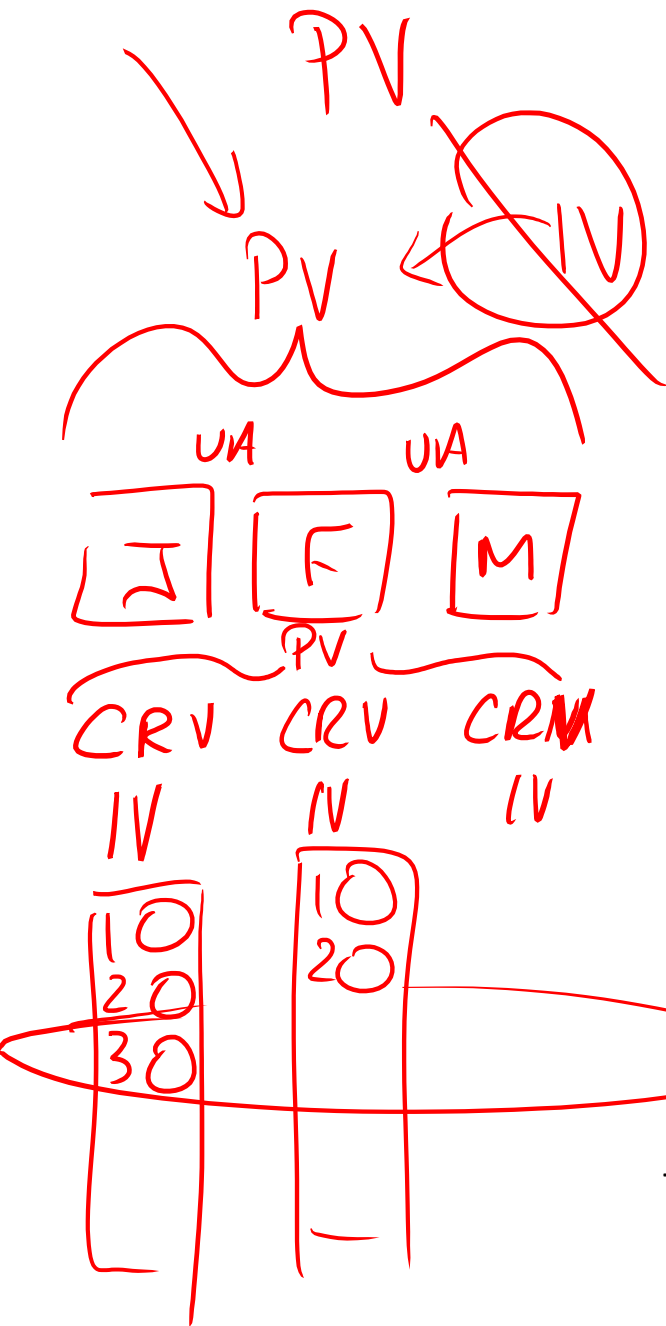
Staging
LOAD

cleanse
CR CL on fsg6
CR NC
CR V/IV

Const.

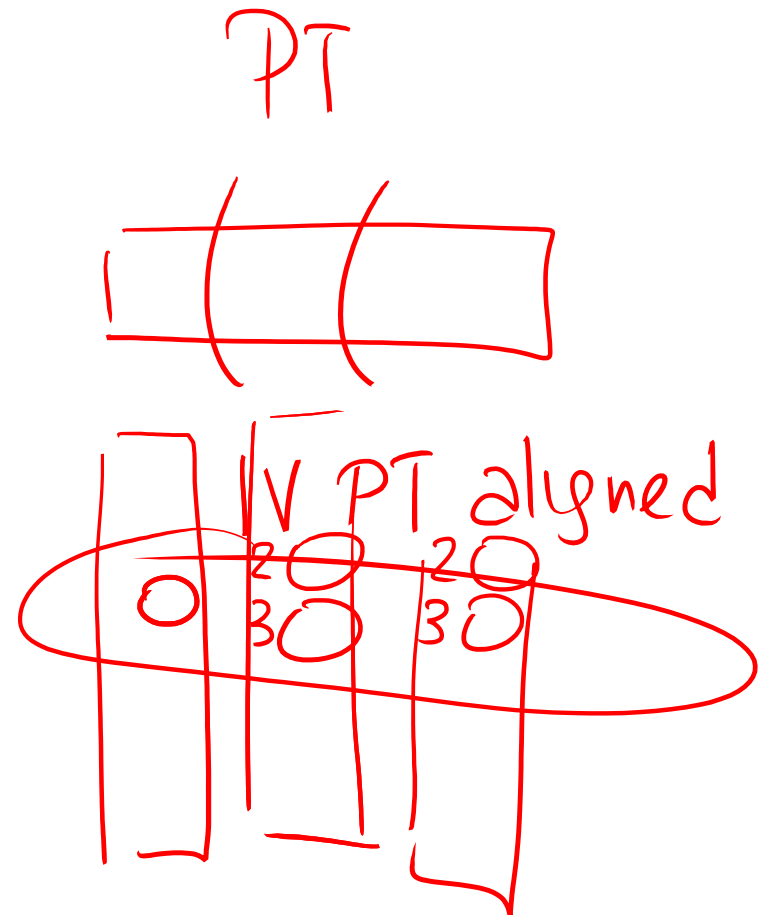
July 04

⑤



With PVs - If you want to create indexed views, you must create them individually per table (for the tables that underpin the PV). If you're not on EE then you'll also need to create a specific view to access them with the (WITH NOEXPAND) hint.

With PTs - If you want to create indexed views, you must create them as partition-aligned (for fast switching). This is available in SQL Server 2008+.



CL OD		Unique	
mdf	(SALES)	NC1	NC2
		CID	PID
		SalesID	NO PK
			<u>DATE, ID</u>

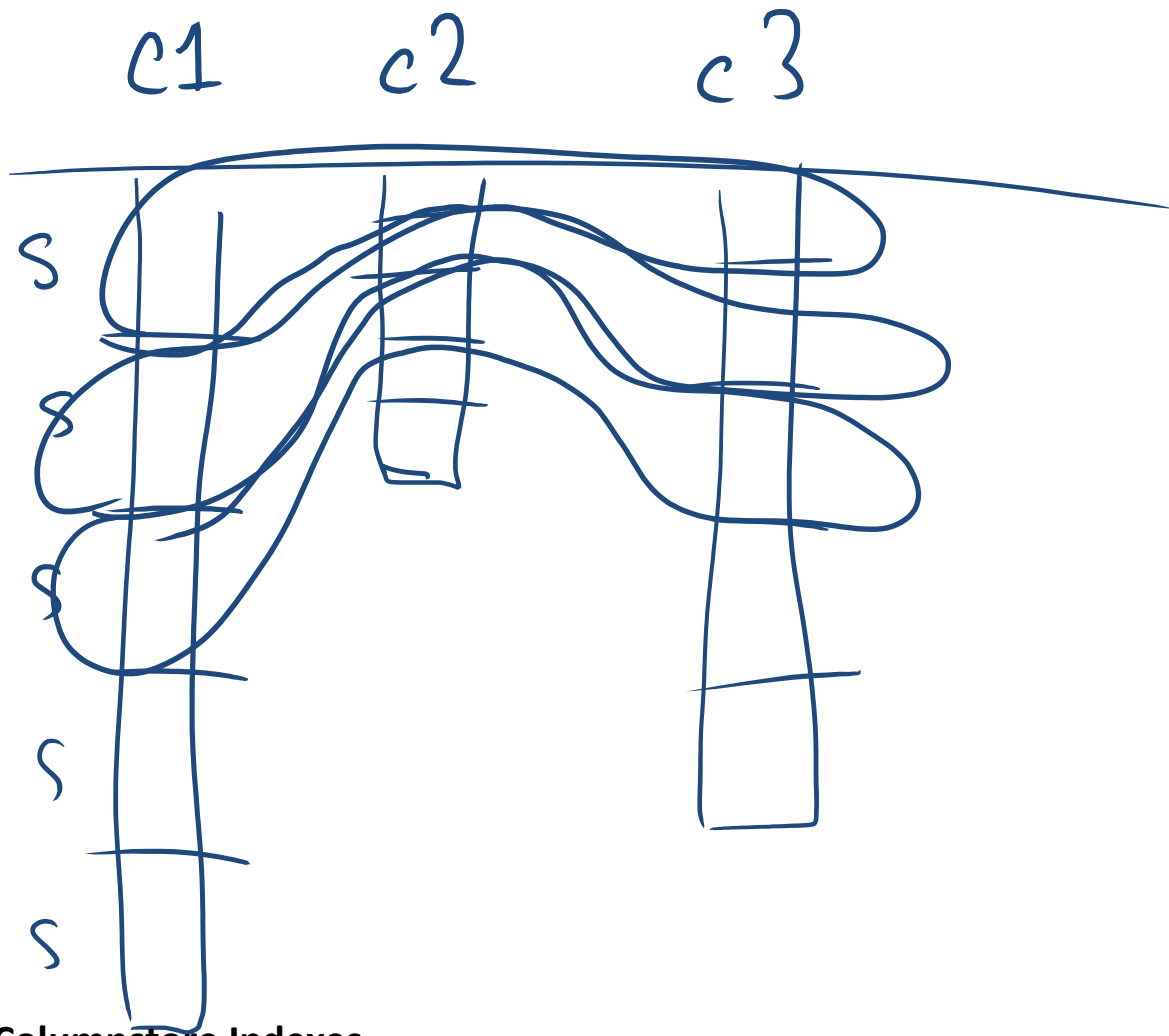


CRUCIAL INDEX
SalesPK on
Sales (SalesID)

Moving/partitioning the CL index

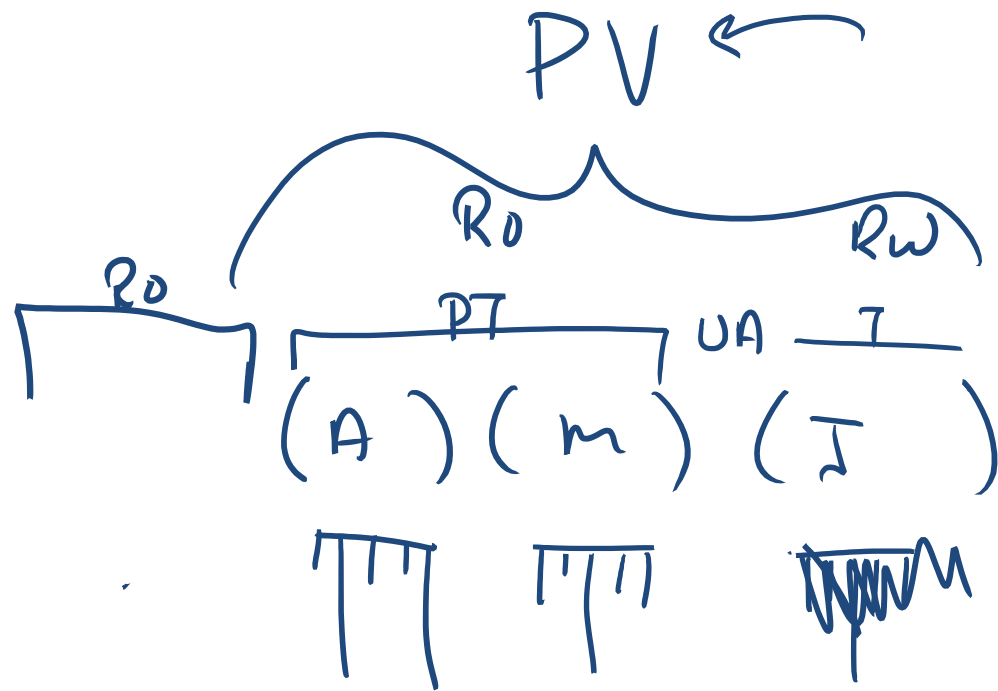
When you rebuild the CL index on a scheme the clustered index (the table) will be partitioned on the new scheme. However, nonclustered indexes will NOT be partitioned. Nonclustered indexes must be built separately/individually. And, they might need to be changed. All unique nonclustered indexes must have the partitioning column defined as part of the key.

If the CL table that you want to partition is clustered by date/id (and is the PK) then rebuilding that on the scheme is fairly easy. But, if it's not the PK and the PK is instead on SalesID then you'll change this PK to have the partitioning key as part of the PK. And, this means all FKs will need to change, etc. This can make the process of converting to a partitioned table become an offline process.



Columnstore Indexes

This was a drawing showing the possible compression of 3 columnstore-based indexes. Then, each columnstore index is broken down into segments. This is the base of batch-mode processing (another core part to the performance gains that can be recognized with columnstore indexes).



Might want to
have CS indexes
only on RO tables

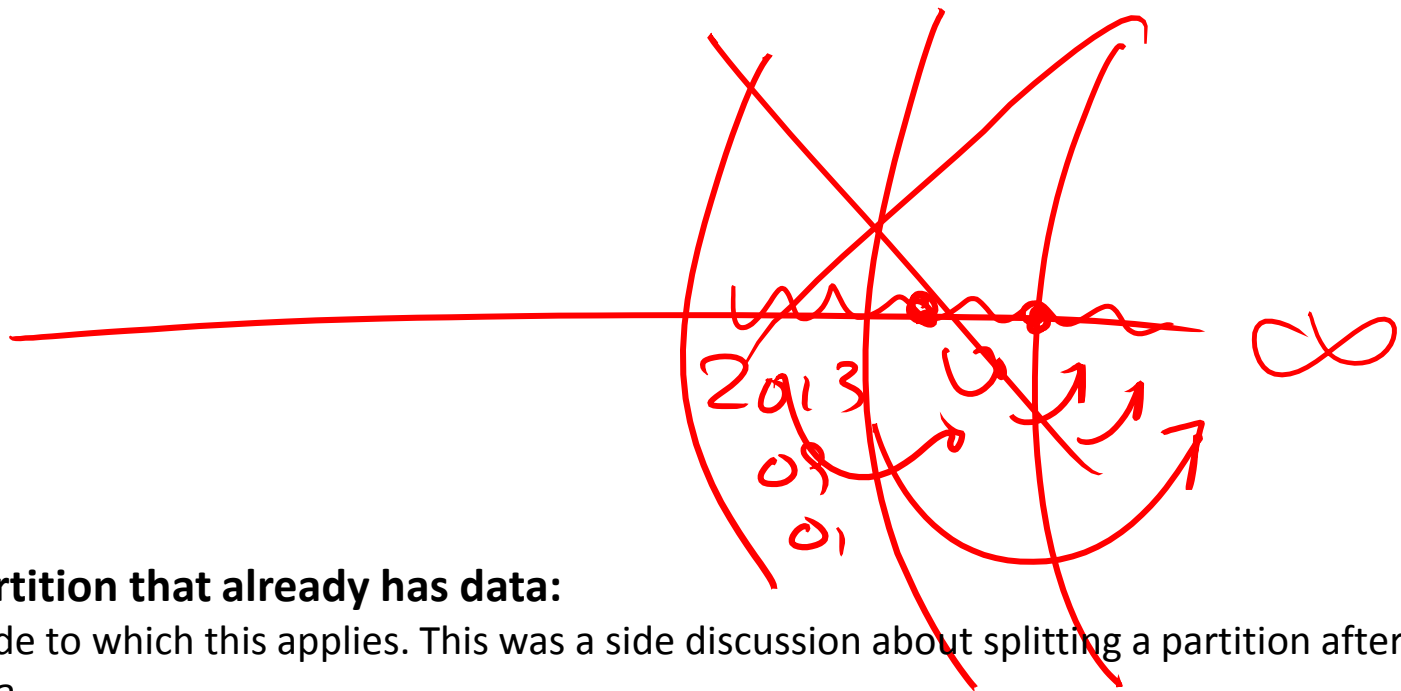
SP
(Q PT)
(Q T)
As
select:

Batch mode support

Since columnstore indexes make the table on which they're created read-only then you might want to use PVs to separate read-write and read-only data.

Having said that, one limitation of the current implementation of nonclustered columnstore indexes is that they do not support batch mode processing across views that include UNION ALL. The workaround is to use CTEs.

Check out the discussion: **Perform UNION ALL and Still Get the Benefit of Batch Processing** on the columnstore wiki here: <http://social.technet.microsoft.com/wiki/contents/articles/perform-union-all-and-still-get-the-benefit-of-batch-processing.aspx>



Splitting a partition that already has data:

There isn't a slide to which this applies. This was a side discussion about splitting a partition after it already has data.

If you forget to split for October and November and the last split was for Sept 1 – then, instead of splitting for October (which has to move BOTH October and November data) and then splitting for November (which has to move November's data again) – you should always split the last set (November) and then the earlier (October). Then, November's data only moves once and October's only once as well

However, if you're going to make significant changes to a partitioning scheme (like 4 partitions to 8) then instead of splitting 4 times – just create a new function and new scheme and then rebuild (possibly online) the object on the new scheme.