



Module 9

Plan Cache & Index Analysis

SQL
skills

update stats

↳ plan invalidation?

auto update
stats
ON

OFF

This was just a small discussion about the fact that plan invalidation does not actually occur if you don't have auto update statistics (as a database option) turned off. Check out the demo script in the plan cache demos (I just added it); `DoNotTurnOffAutoUpdateStatistics.sql`

Also, be sure to review Erin Stellato's links about auto update stats and plan invalidation:

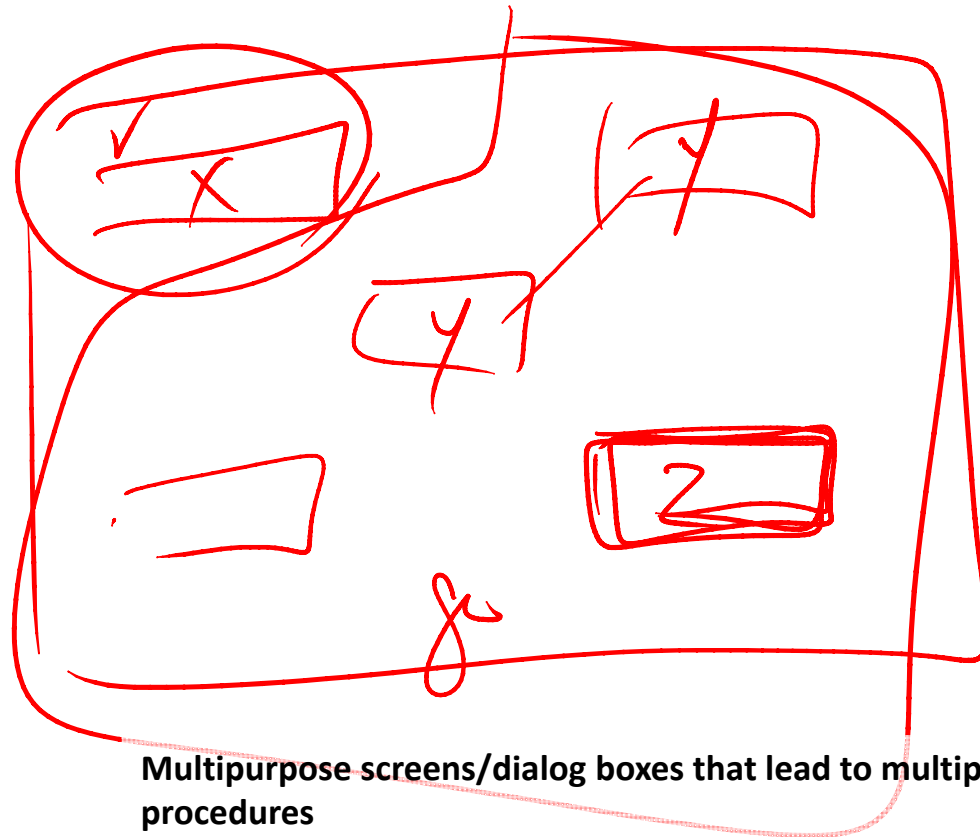
- Statistics and Recompilations: <http://erinstellato.com/2012/01/statistics-recompilations/>
- Statistics and Recompilations, Part II: <http://erinstellato.com/2012/02/statistics-recompilations-part-ii/>

Views

Good plan

med } LOOK

BAD will



Multipurpose screens/dialog boxes that lead to multipurpose procedures

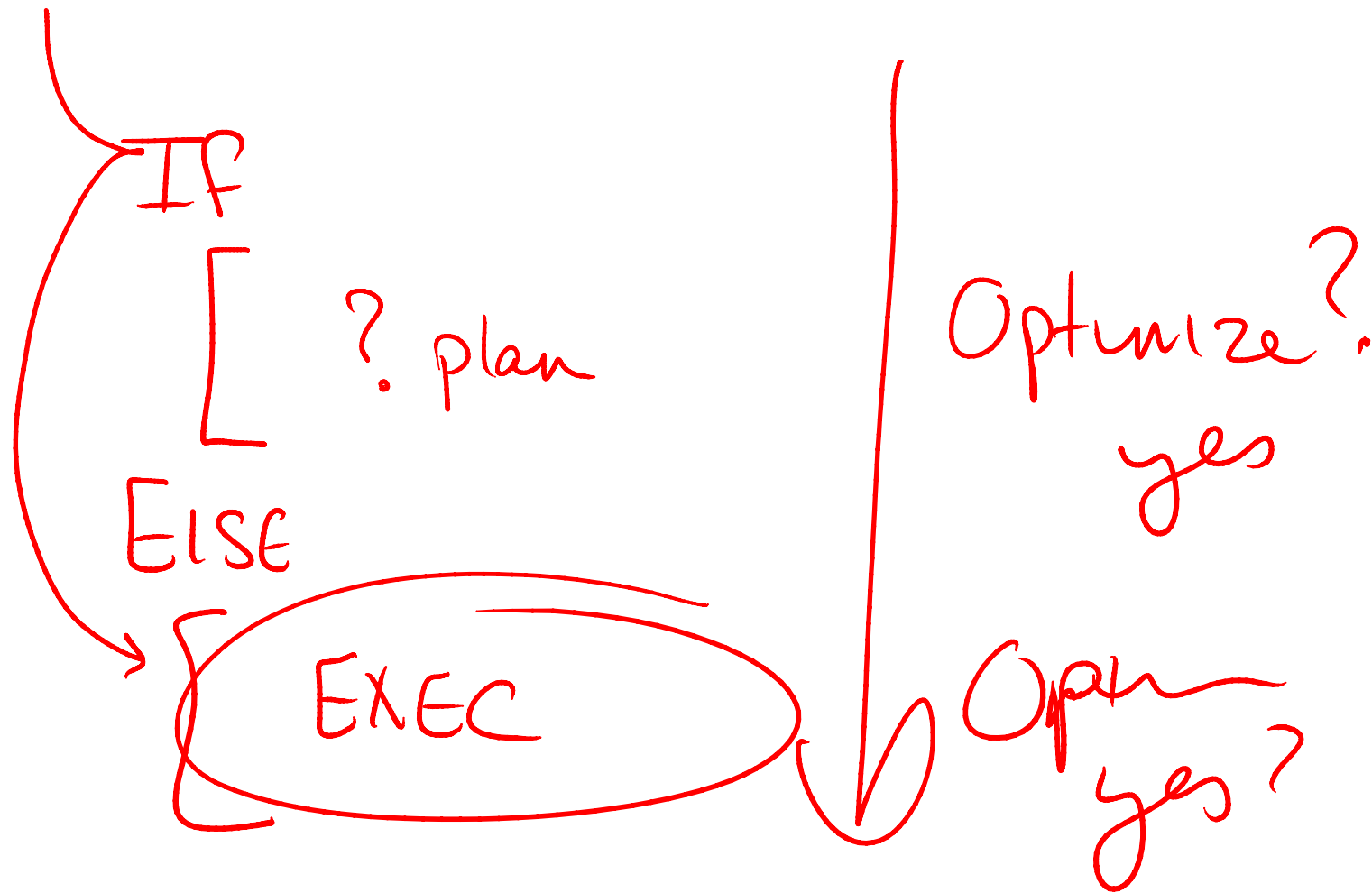
They looked/sounded good at the time?

But, they're often VERY hard to optimize with where clauses that look like:

```
WHERE (Column1 = @Value1 OR @Value1 IS NULL)
      AND (Column2 = @Value2 OR @Value2 IS NULL)
      AND (Column3 = @Value3 OR @Value3 IS NULL)
```

...

Check out the demo scripts in order to fully see how to better optimize this – using `sp_executesql` (for combinations that are stable) and `EXEC (@String)` for combinations that need to be recompiled. Or, use `sp_executesql` with strings that include `OPTION (RECOMPILE)` for the parameters that might not generate consistent plans.



Conditional logic and modularization

Creating branching logic for different types of parameters seems logical but because SQL Server optimizes the process of optimization (where they try to optimize anything that's optimizable), you can often end up with a plan that's not ideal.

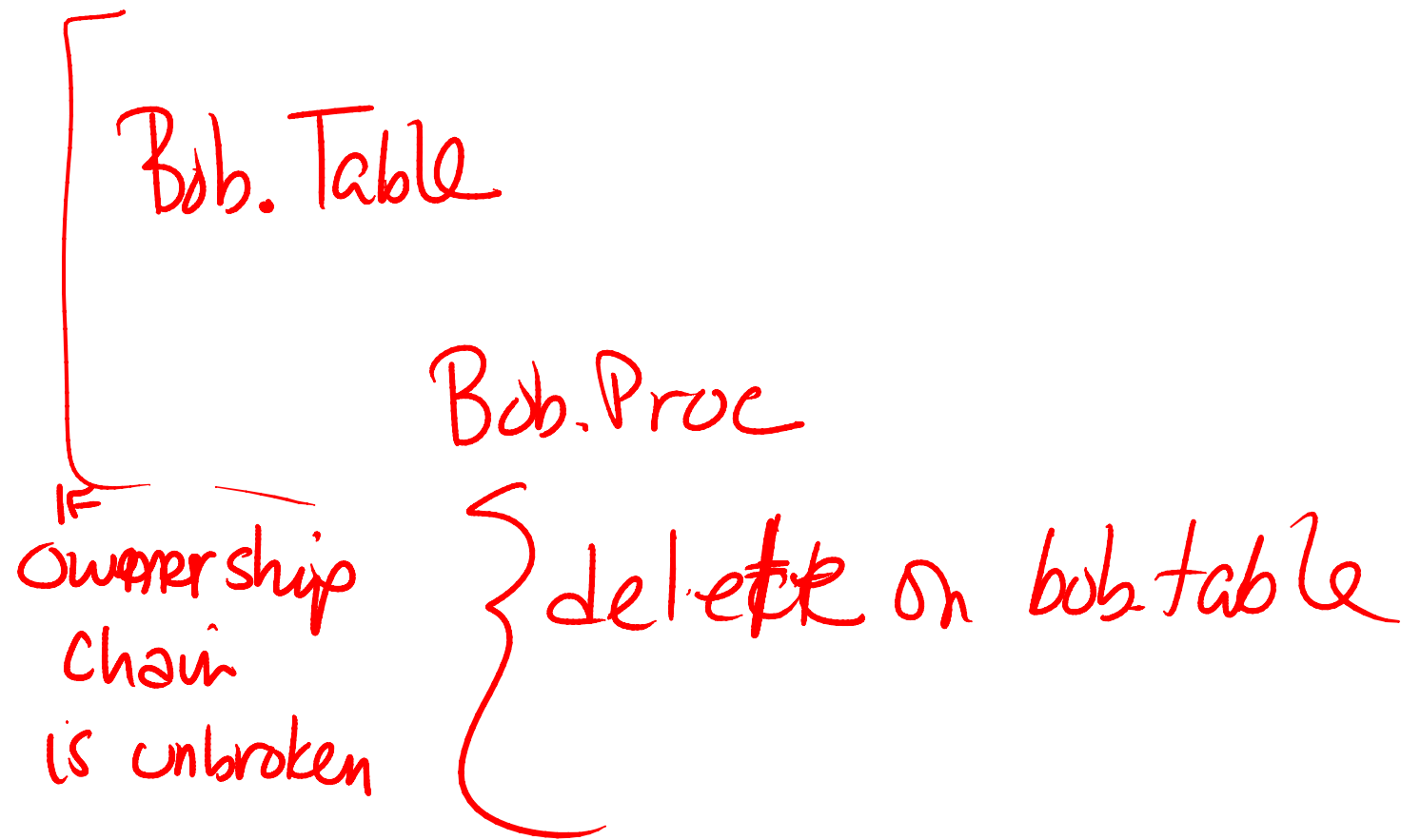
You are much better off breaking that code into smaller subprocedures – SQL Server will never step into a subprocedure unless it's executed and in this case it will only be executed with exactly the right parameters.

DEFAULT
Proc

DSE — protects against SQLinj
execute under context
CALLER

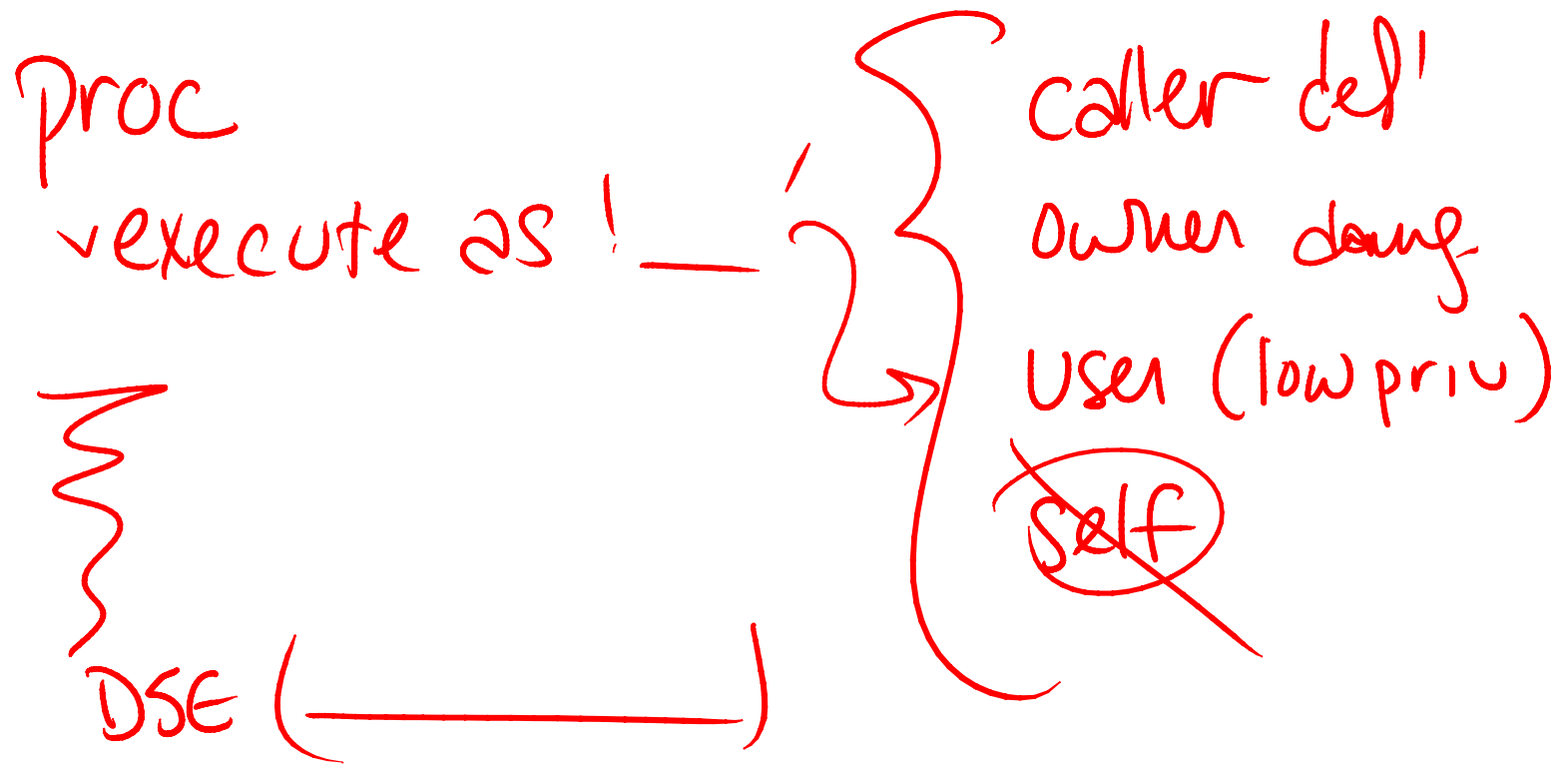
Dynamic String Execution and SQL Injection

TO prevent SQL Injection, SQL Server requires that strings execute under the context of the caller.



Without Dynamic String Execution permissions flow through the ownership chain

The idea is that when the ownership chain is unbroken then direct permissions (for example, to do a delete) are not necessary. As an owner you are giving rights to a process; this procedure can be protected with more business rules/logic and even reduced/isolated to specific rows. The end user does NOT need direct delete permissions to be able to execute the procedure as long as they've been given execute rights on the sproc.



Dynamic String Execution, EXECUTE AS and SQL Injection

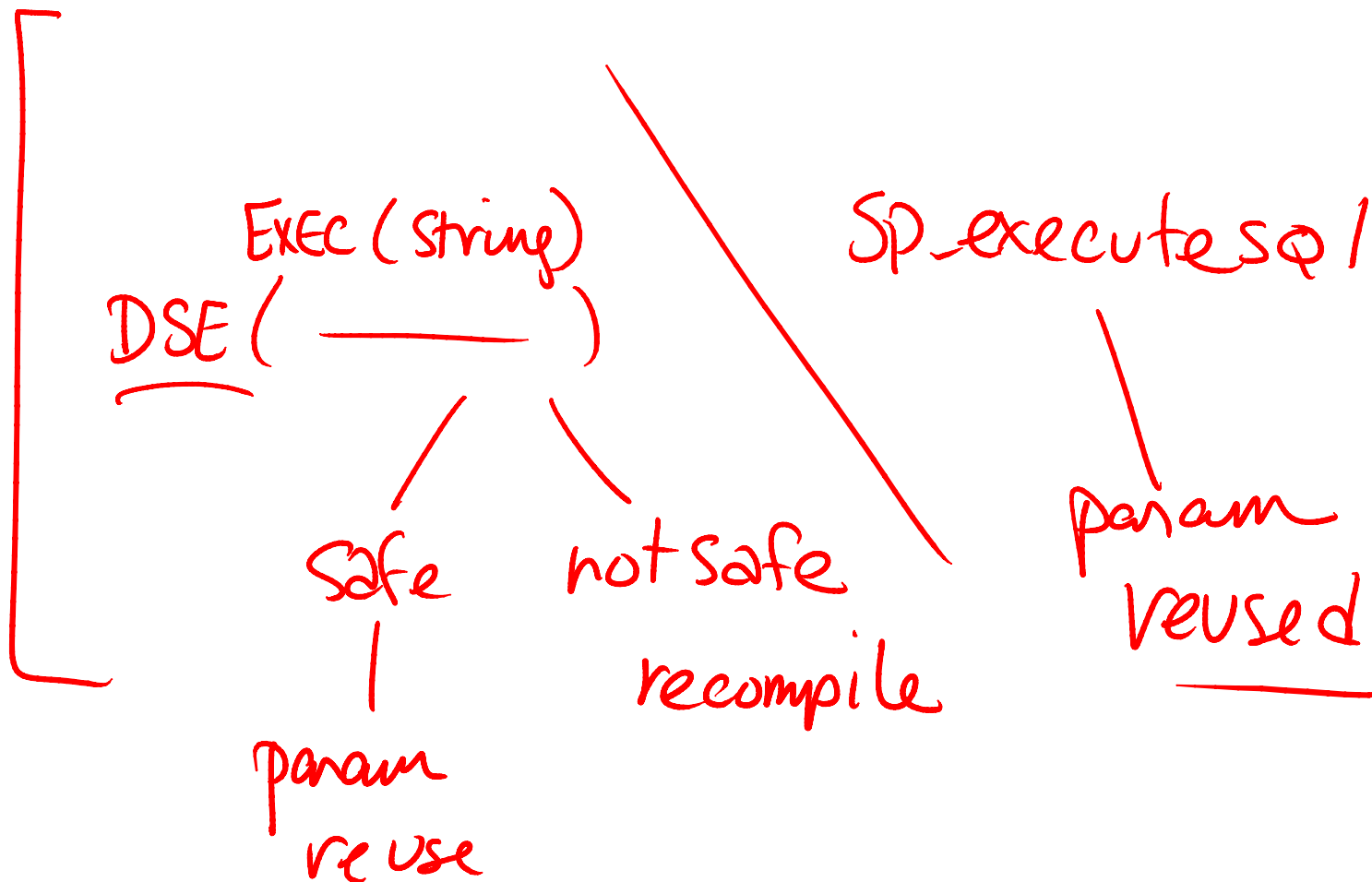
People complained that granting permissions directly to the user was a problem... so, they wanted an alternative. Enter: EXECUTE AS.

While it solves the problem of permissions, it adds more potential for SQLInjection.

So..... Check out these posts:

[Little Bobby Tables, SQL Injection and EXECUTE AS](#)

["EXECUTE AS" and an important update your DDL Triggers \(for auditing or prevention\)](#)

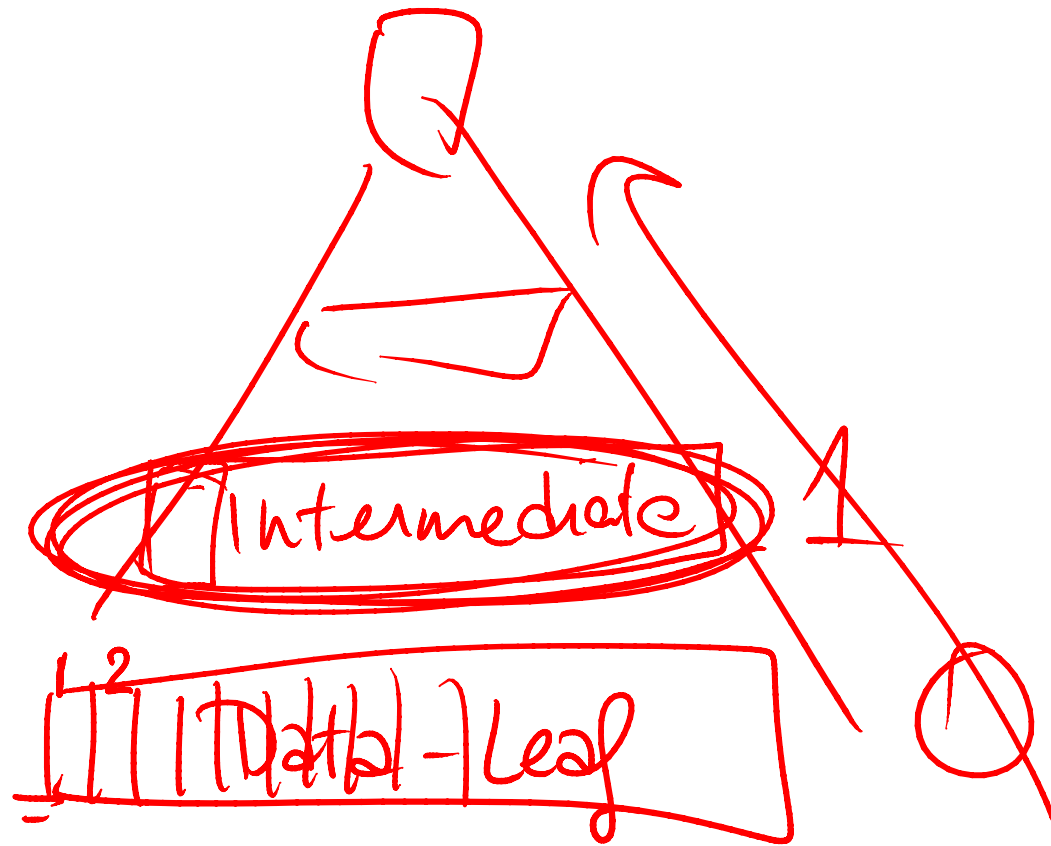


Using EXEC (string) [DSE] and/or sp_ExecuteSql [ES] inside of stored procedures

DSE and ES are not the same.

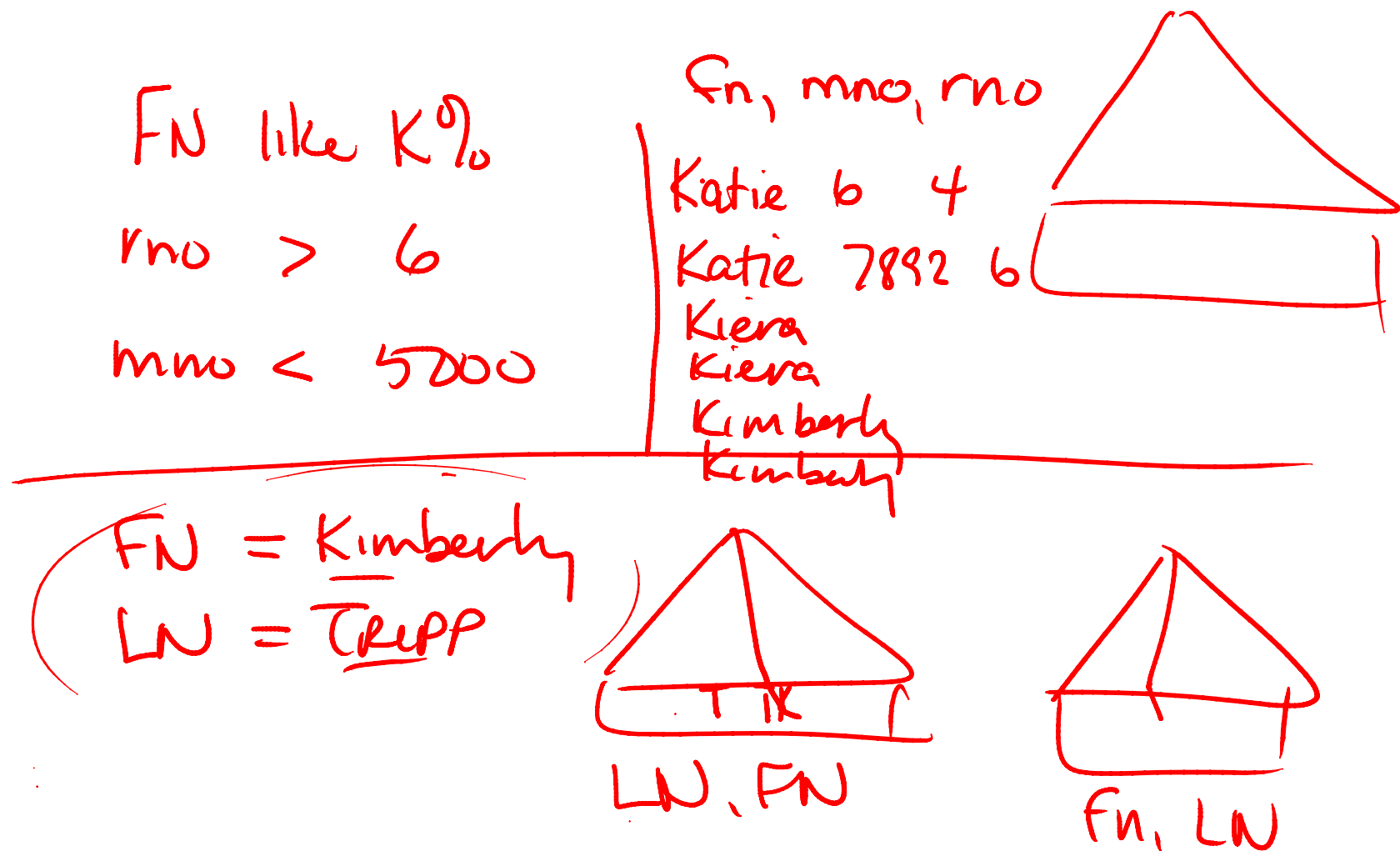
DSE is unknown until runtime. At that point the statement is treated as though it's adhoc. Because most statements are **not** going to be safe, the statement will end up being recompiled (and optimized) for each execution.

ES is forced parameterization. When a statement is stable you can use this to reduce compilation/CPU costs.



Index Health – Checking for fragmentation

Here I was drawing how `sys.dm_db_index_physical_stats` uses a scan of the first intermediate level (index level 1) to analyze for the value: `avg_fragmentation_in_percent`. If your analysis/rebuild scripts use this to determine fragmentation than you'd be better off just doing a limited scan. What `SAMPLED` and `DETAILED` offer are page-specific details (page density, forwarding pointers, etc.) and if you're NOT using those (most people aren't!) then you're wasting time with a `SAMPLED` or `DETAILED` scan during your maintenance window!



Adding more indexes

Here we were discussing the order of columns. If there are range-based predicates then it doesn't usually matter which column is second. You tend to want to put the most selective first (in terms of the PREDICATES supplied). If all of the predicates are equality-based then it doesn't really matter. You're best ordering them by the LEFT-based combination that's used most commonly.

KEY

Density_Vector

c1	c2	c3	c4

_____	_____		
_____	_____	_____	
_____	_____	_____	_____
c2	c4		
c2	c3		
c3	c4		
c2	c3	c4	

Multi-column, column-level statistics

Only DTA recommends multi-column, column-level statistics...

If you are about to trust (and create) a “green hint” recommendation (which comes from the Missing Index DMVs) then you might want to first run the query through DTA. If you end up creating the index that DTA recommends then you should also create the recommended stats (for that same table). These statistics can be helpful to the optimizer to better understand non-left-based subsets of the index for other possible uses.