# Twitter Thread by Andy Pavlo

**Andy Pavlo**
@andy_pavlo

## If you're interested in DB internals, stop what you're doing and watch @CMUDB Quarantine Talk from Nico+Cesar about @SQLServer's Cascades query optimizer: https://t.co/FCdsbHHEaD

## Many talks this semester were good. This one is the best. My thread provides key takeaways

[6:50] Microsoft hired Goetz Graffe in the 1990s to help them rewrite original Sybase optimizer into Cascades. This framework is now used across all MSFT DB products (@SQLServer, @cosmosdb, @Azure_Synapse).

[14:43] The optimizer checks whether it has stats it will need before cost-based search. If not, it blocks planning until the DBMS generates them. This is different than other approaches we saw this semester where DBMS says "we'll do it live!" with whatever stats are available.

[21:05] Their Cascades' search starts small/simple and then they make decision on the fly whether to expand search based on the expected query runtime and performance benefit from more search.

[26:33] They explicitly have a property for Halloween Problem. Operators specify whether they protect from it and then optimizer ensures property is satisfied. This is mindblowing. I have never thought about using the optimizer for this but it makes sense. https://t.co/hjjoGCwyvl

[33:16] This is the menu of all the stats that they maintain for tables. Again, the latest research shows @SQLServer has the most accurate stats: https://t.co/d1btkxmsYf

[39:05] @SQLServer uses a general-purpose cardinality estimation framework. This allows them to programmatically select the best data structure to use per expression type. They rank choices by "quality of estimation". This needs further research.

[44:16] Question from @Lin_Ma_: Are you using ML for cardinality estimation?
Answer: @cosmosdb is using it. @SQLServer is more conservative and using a minor form of it.

[53:14] They use heuristics to pre-seed Cascades' memoization table with plans that they think will be good. This allows the search to start from a local optimum instead of a random location in search space.

[54:48] Optimizer uses logical timeouts (# of plans considered) instead of physical timeout (wallclock). This ensures that DBMS always produces plans with same quality under high load. Hand-tuned timeouts for different optimization stages.

[1:00:45] They also use pre-seeding to support DBA provided query plan hints! This is another genius idea that seems obvious once somebody shows it to you.

[1:03:32] This example shows the limitations of Cascades' tree-based plan search. For some optimizations, the DBMS must also consider hypergraphs. See Neumann SIGMOD'08: https://t.co/s825mXPMqK