SQLintersection

SQL Server Indexing: Strategies for Performance

Discussions / drawings around VLTs

Kimberly L. Tripp
President / Founder, SQLskills.com
Kimberly@SQLskills.com
@KimberlyLTripp

SQL *intersection*

SQLskills
immerse yourself in sql server

Partitioning: PVs vs. PTs

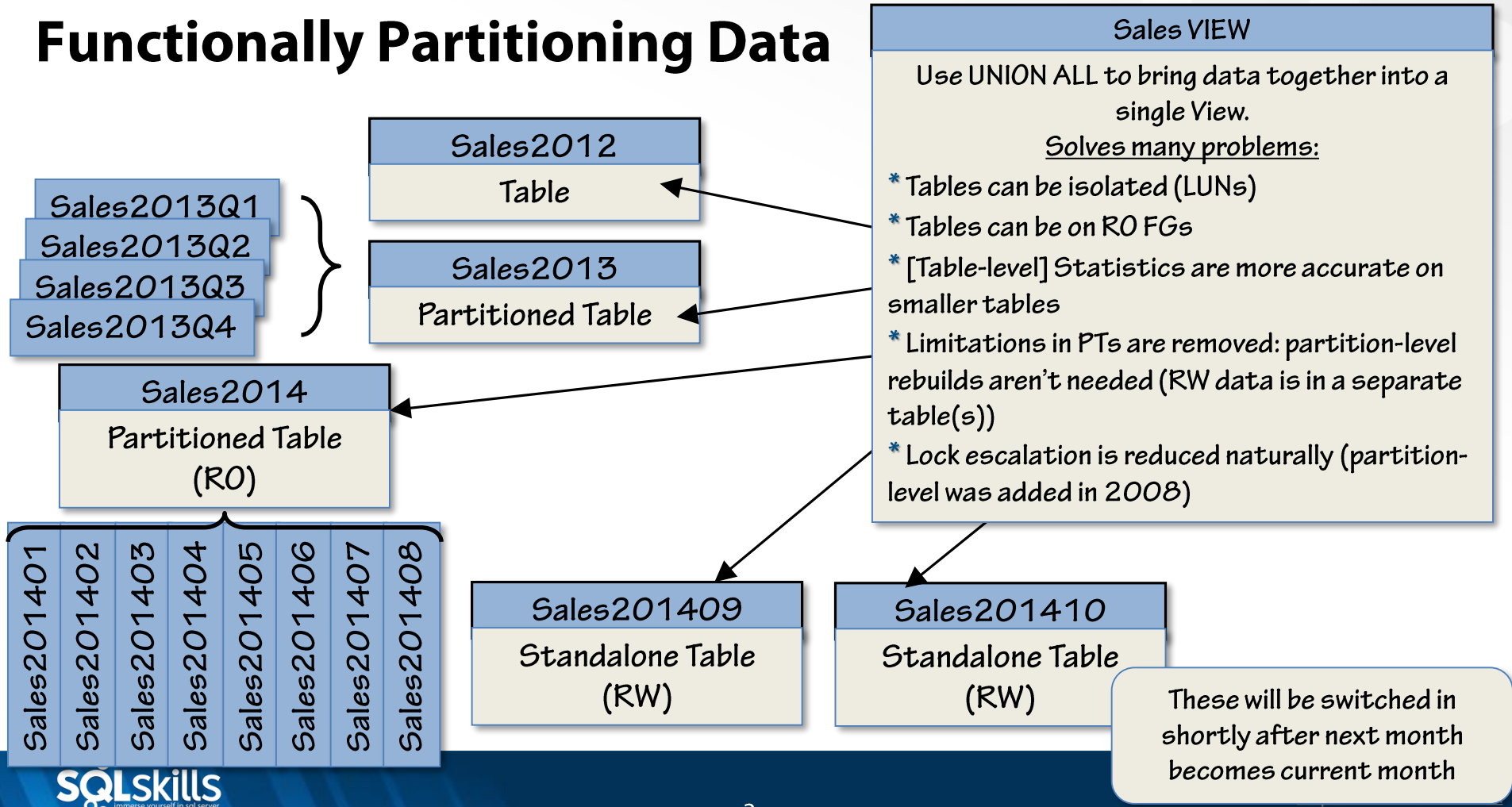
Partitioned views

- 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

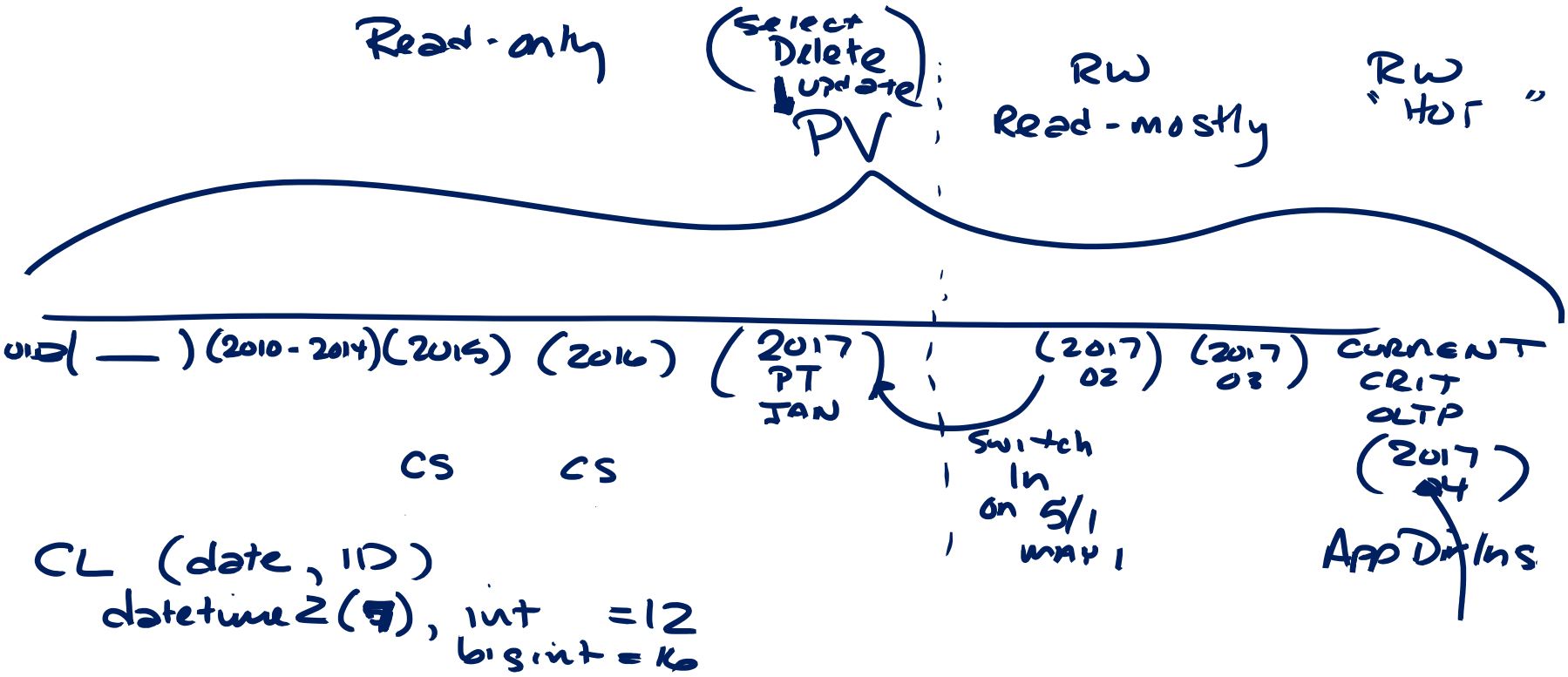
- Enterprise Edition only
- Only 1 table to administer
  - Only 1 table to create/drop indexes
    - All partitions have same indexes (which is easier for the optimizer to optimize)
    - Can create different indexes with filtered indexes
  - No possibility of errors (or gaps or overlapping values)
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

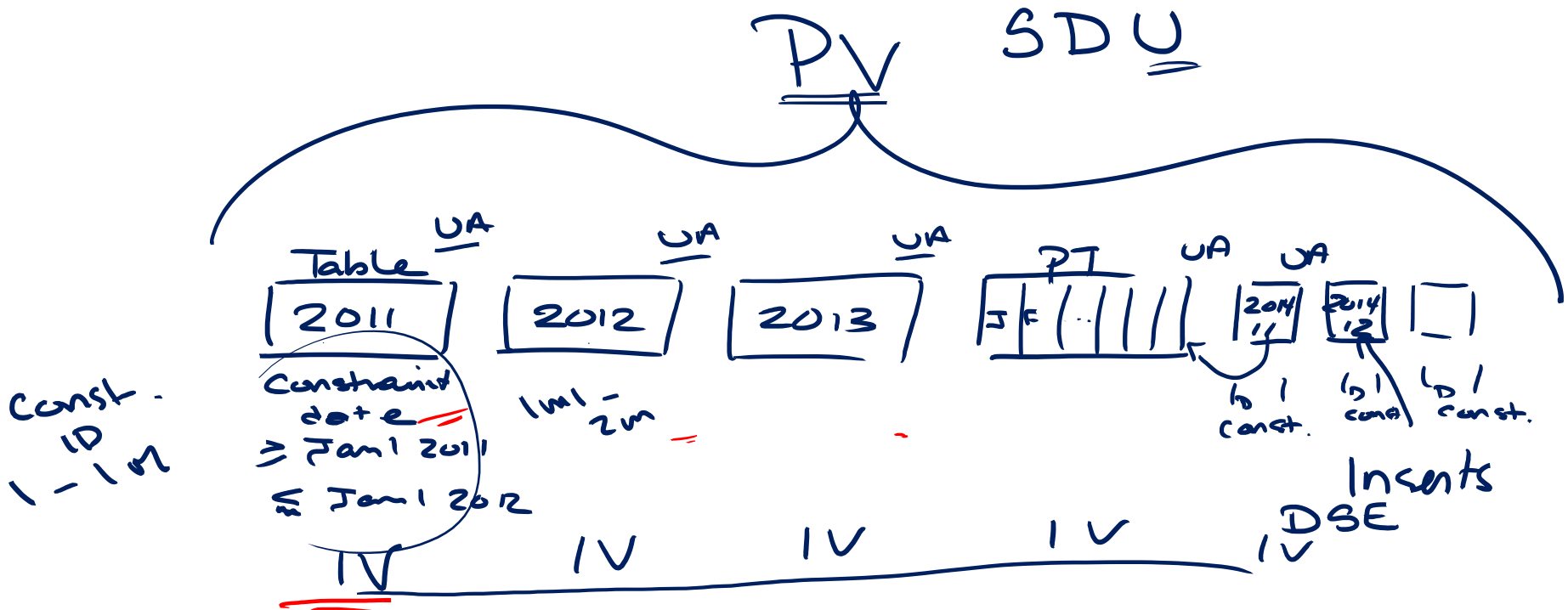
Functionally Partitioning Data



Architecting a "layered" approach to your table design

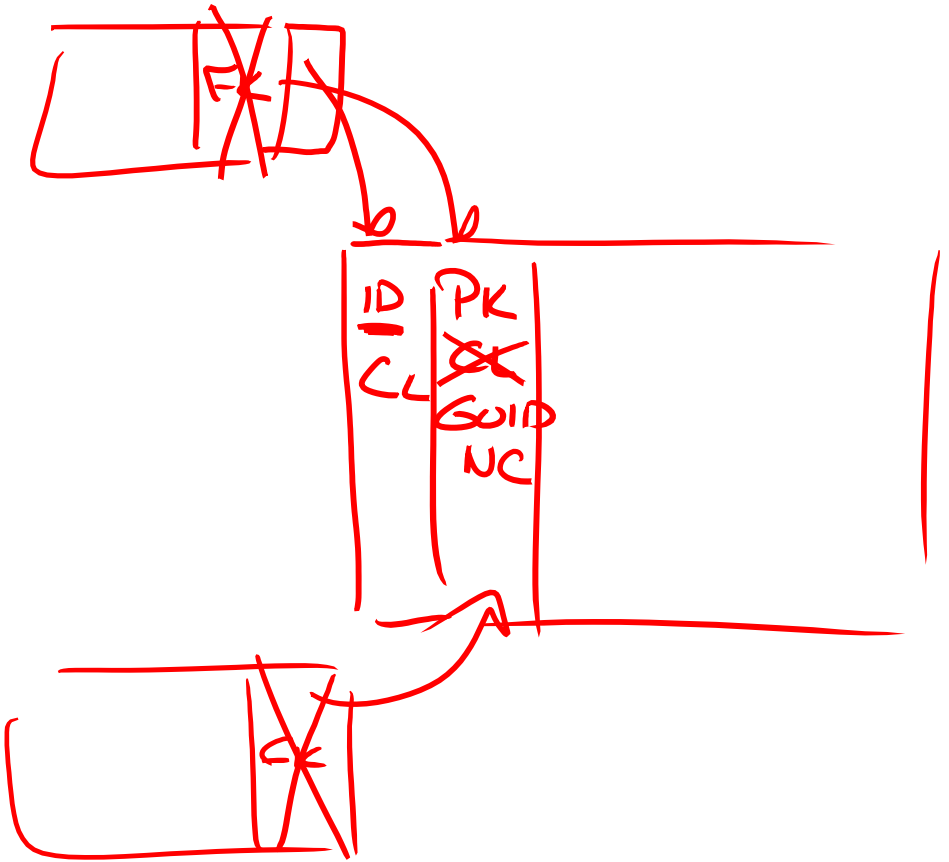
A better option is to separate the RW from the RO and put the RW in separate tables. Then, you can do online rebuilds at the table level. How do you deal with queries – put a partitioned view over them!





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 the historical data). And, you can create IVs on the base tables for further gains.



Take offline
DROP FKs
DROP NCs
DROP CL\PK
Add new column
CR CL
CR PK NC
CR NCs
CR FKs
Bring Online

What does it take to change a CL PK?

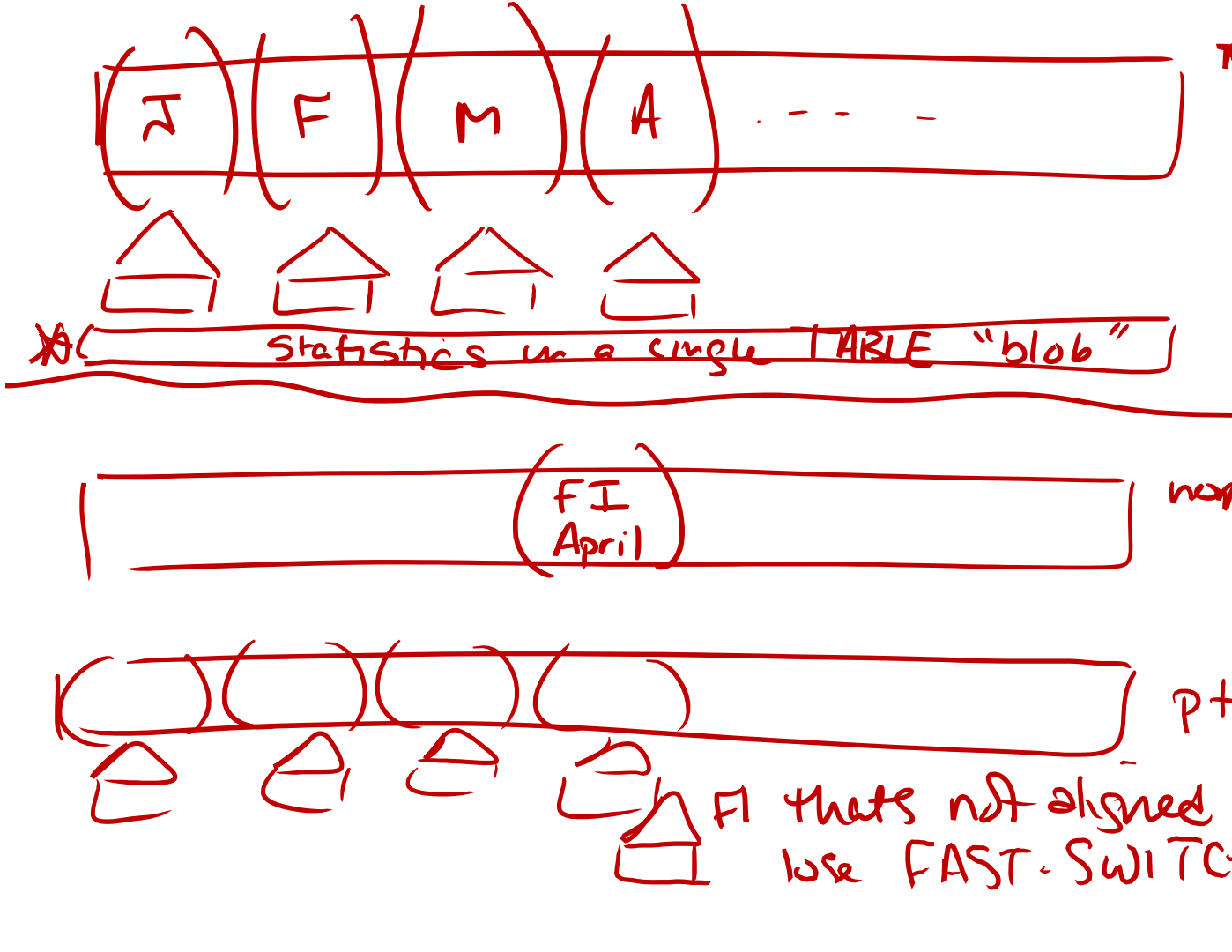
Unfortunately, A LOT. And, because of the lack of indexes and foreign keys – the entire process is OFFLINE.

This is why this is such an important decision to make early!

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…