***Review : SharedDB: Killing One Thousand Queries With One Stone***

***Problem:***

* Query at a time model optimizes performance in a best-effort way
* This model is not optimal for systems that need to process hundreds of concurrent queries

***why is it important:***

* Many modern applications need response time guarantees in high load situations.

***why existing solutions do not work:***

* query-at-time cannot meet the performance requirement of high and dynamic workload as it can suffer from resource contention and interference.
* Most of the systems start executing queries as soon as it arrives, which limits the opportunities to share computations across queries

***Core intuition:***

* Main idea of SharedDB is it batches queries and updates in order to share computation across these queries and updates.
* Adopts many ideas developed in the context of multi query optimization and data streaming.

***Design/*** ***Architecture :***

*Main components*

* **Data-Query Model:**
* Model to represent the intermediate query results.
* Additional column - query\_id - to keep track of the identifier of the queries
* Used to uniquely identify active queries.
* Column implemented as a set-valued attribute - this allows operator to be applied to the tuple only once - across multiple queries
* Main memory footprint also reduced significantly.
* Set-valued attribute implemented as a list data structure.

* **Global Query Plan:**
* Key innovation - is that way it processes joins, sorts and group-bys
* query\_id made a part of join predicate
* Support multiple join types - hash join, sorted join, cache-aware join
* Steps involved in hash join
  + Step 1 - Logical Query optimation - queries are parsed and compiled individually -thereby pushing down predicates.
  + Step 2 - The individual query plans are merged into a single global plan - which results in one big join instead of 3 small joins
  + Step 3 - join results routing carried out by using a grouping operator.
* Same idea applies to sorts and groupby
* **Shared Operators :**
* Designed to evaluate a number of queries concurrently by processing them in cycles
* Every cycle excutes set of active queries or subqueries
* Any additional queries that arrive after the cycle has started is queued

***Some Next steps:***

* Cost-based optimizer - to create global query in one pass
* Have a hybrid model – so that – we can make use of the query-at-time when the workload is not heavy and use the batching and shared computation model when works load is heavy.