# BI for Hadoop

## How to make OLAP work for Hadoop

**Daria Hutchinson**

**AtScale Technical Publications**

The basic premise of Online Analytical Processing (OLAP) is to allow people to access data, ask questions, and get answers quickly. OLAP is the foundation of business intelligence (BI) – the broader set of tools and methodologies for gaining meaningful insights from raw transactional data. While OLAP is a concept that was first coined in 1993, the business need it addresses still exists today, perhaps even more so in this new era of Big Data. Although the underlying technologies have evolved, the goal of an OLAP engine is still the same - optimize the data so end-users don't have to wait to get the answers they need.

## Table of Contents

## The Trade-Offs of MOLAP, ROLAP and IMDB

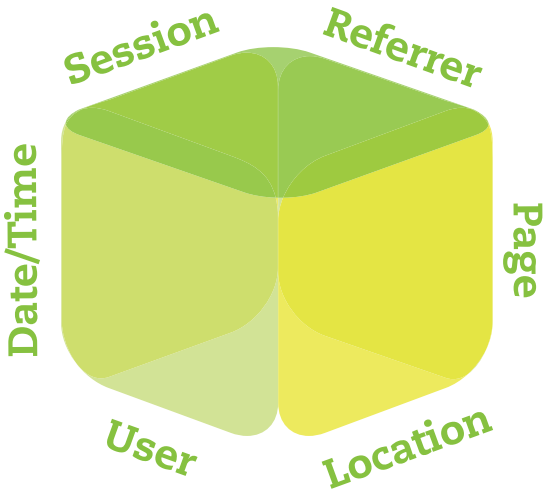
History has taught us that it is important to understand the past before planning for the future. Before tackling OLAP on Hadoop, it is important to understand some basic concepts of OLAP. When describing OLAP systems, there are two major approaches in how the data is optimized to support BI analysis - MOLAP (multi-dimensional OLAP) and ROLAP (relational OLAP). Some solutions may further optimize performance by loading the prepared data into an in-memory database (IMDB).

Historically, each approach has had its benefits and limitations. Going forward, these limitations are only amplified in the context of Big Data, and Big Data presents new challenges that were not a consideration before.

| | Benefits | Limitations |
|---|---|---|
| MOLAP | • Query performance is fast on pre-computed data<br>• Data modeling is intuitive to business users | • Does not scale to Big Data workloads<br>• Requires excessive storage<br>• Does not use standard query languages<br>• Operational overhead to manage separate cubes and data marts<br>• Performance gains diminish with high-cardinality dimensions<br>• Not elastic or flexible - small changes require rebuilding the entire cube |

| | | |
|---|---|---|
| ROLAP | • Scales to Big Data workloads<br>• Uses standard query languages<br>• Centralized data store provides a single 'source of truth' | • Slower query performance on non-aggregated data<br>• Requires complicated data modeling and ETL<br>• Operational overhead to manage aggregate tables |
| IMDB | • In-memory queries are substantially faster than disk reads | • Not cost-effective for Big Data<br>• Ephemeral storage does not persist through database restarts |

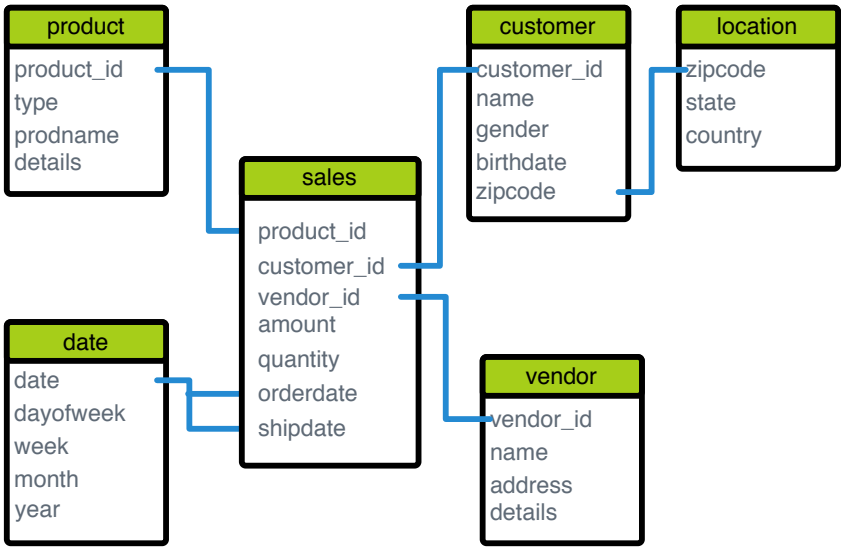
In the MOLAP approach, aggregate data is pre-calculated for every dimension combination. The result is a multi-dimensional cube, where each cell in the cube represents an intersection of *n* dimension values. For example, if you were building a cube to do analysis of sales transaction data, the number of cells in the cube is equal to the cross-product of all dimensions. So to count the number of customer purchases by gender (which has 2 distinct values - Male, Female) and by state (which has potentially 50 distinct values - AZ, CA, TX, and so on), the intersection of those two dimensions alone would require 100 pre-computed values in a MOLAP cube.




The benefit of building MOLAP cubes is performance. Since every possible combination of dimensions is calculated ahead of time, slicing and dicing of multi-dimensional data is very fast. But this performance comes at a cost - the cost of redundant storage, proprietary query languages, and the administration

overhead that comes with managing silos of data. Plus, when dealing with the size of dimensions in big data (millions of IP addresses, hundreds of millions of users, billions of click events), MOLAP cubes cannot scale. Cube build times only increase with big data, and one failed cube build can disrupt the business processes that rely on that data.

In the ROLAP approach, data is not pre-computed ahead of time. Instead data is stored in relational tables that use a star (or snowflake) schema to model multi-dimensional data. A star schema consists of a central fact table that references a number of dimension tables. Tables are related to each other based on key fields that they have in common. For example, sales transaction data might be modeled around a central sales table that joins to various dimension tables to provide more context about a sale event.




Analysis tools then query these tables using standard query languages such as SQL, and results are calculated on-the-fly for the requested dimensions. The benefits of this approach is that it is more scalable in handling large data volumes, uses standard query languages, and does not require off-loading the data into another storage system for analysis. But with ROLAP, there are trade-offs as well. ROLAP engines are usually slower because the data is not aggregated ahead of time. To get around this performance hit, it is often necessary to build separate summary tables that contain pre-aggregated data in order to boost query performance. So what is gained in scalability and conformity, is lost in complicated ETL (extract, transform, load) processes to model the data and maintain the aggregate tables.

An in-memory database (IMDB) boosts performance of BI queries by loading an entire database or OLAP cube into memory. Processing data in-memory is always faster than reading data from disk. With Big Data, however, it is just not feasible or cost-effective to store hundreds of terabytes of data in RAM just waiting for a query to come along.

## OLAP in the Age of Hadoop

In the new world of Hadoop, consolidating and storing massive amounts of data is relatively easy. The Hadoop distributed file system (HDFS) can store virtually any data, structured and unstructured, without the requirement of defining schema ahead of time. This has allowed businesses to store more data in one place than was ever possible before. The potential of mining this 'data lake' for valuable insights is huge. However, the first generation of OLAP on Hadoop fell short of expectations. Most 'BI on Hadoop' solutions today are still relying too heavily on the solutions of the past, and fall into one of three camps:

- **Proprietary MOLAP Data Marts.** These solutions use Hadoop's batch processing capabilities to build MOLAP data marts, and then move the data off of Hadoop and into proprietary purpose-built analytics clusters or in-memory databases. This not only leaves the massive parallel processing capabilities of the Hadoop cluster under-utilized, but introduces silos of data, proprietary query engines, and another cluster of servers to manage.
- **HBase MOLAP Cubes.** These solutions keep the data on the Hadoop cluster, but use MapReduce to pre-aggregate the data and load it into MOLAP-like views and indexes in HBase (a NoSQL database that runs on top of HDFS). When utilizing this approach, separate cluster nodes are required for the HBase region servers. Given that HBase is known to be difficult to set up and maintain, this adds extra complexity to the overall solution. Additionally, OLAP workloads often involve scans over large ranges of data. HBase is not optimized for OLAP workloads, and can lead to a large number of queries to assemble the proper result set. Finally, with the high-cardinality dimensions that are common in Hadoop-scale data sets, dimensional explosion can lead to extremely long cube processing times and storage footprint explosion.
- **Hive ROLAP Tables.** Hive has become the standard relational data warehouse for Hadoop. It is a core component of the Hadoop ecosystem, and overlays structure on top of raw data in HDFS. Earlier generation Hive queries executed batch MapReduce jobs on the Hadoop cluster, which did not offer the interactive query performance required for BI. While it is now possible to use Hive with a more performant SQL-on-Hadoop engine instead of MapReduce, modeling the data into a star schema of relational Hive tables is an advanced skill that most data analysts don't have.

So what needs to happen to make OLAP directly on Hadoop possible? Hadoop demands a new OLAP approach that can handle its data volume and complexity. A modern OLAP solution should:

- **Take advantage of recent innovations in the Hadoop ecosystem.** The SQL-on-Hadoop engines such as Impala, SparkSQL, and Hive Tez now make it possible to use Hadoop's processing power for ad hoc query workloads, not just as a batch processing engine.
- **Be designed with the future in mind.** Too many of the existing BI and OLAP solutions available on Hadoop today were build using outdated MOLAP or ROLAP methodologies. They have inherited the limitations of those approaches and have not evolved to address the unique challenges of today's large, complex data sets.
- **Learn from the past, but leave the limitations behind.** A modern OLAP solution requires a new approach to address the new world of Big Data. The MOLAP model is easy to understand, but it doesn't scale. ROLAP scales, but requires complicated processes to model and optimize the data. A modern OLAP solution combines the scalability of ROLAP with the ease-of-use of MOLAP-like data modeling, making the full power of Hadoop accessible to more users and business processes.

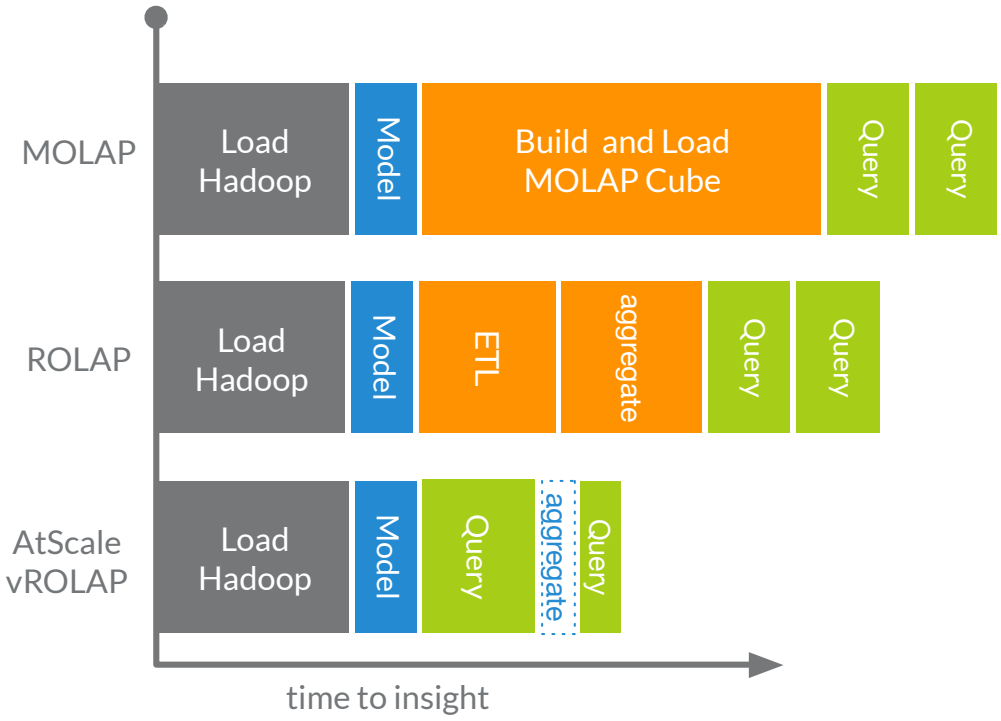
An OLAP solution that is purpose built for Hadoop needs a new approach - vROLAP (virtual relational OLAP). The basic goals of vROLAP are to support existing BI workloads using Hadoop as the sole platform for data storage, data discovery, data optimization, and query processing. A vROLAP solution has the following goals:

- **Model without Movement.** ROLAP is the right approach for big data. The exponential growth of data cannot scale with the old MOLAP approach. But the data that is landing in Hadoop does not lend itself well to a tidy multi-dimensional relational model. Instead of pre-processing this data into a ROLAP model up front, allow business users to define a vROLAP (virtual relational) model on top of the datasets stored in the Hadoop file system using multi-dimensional modeling concepts they already understand.
- **Smart Aggregates.** Maintaining aggregate tables is one of the biggest drawbacks of ROLAP, although to get adequate performance, it is a necessity. Instead of building

and maintaining summary tables up front, a vROLAP engine dynamically builds and maintains aggregates on-demand based on what BI users need.

- **Optimize the Queries not the Data.** OLAP engines of the past have focused on optimizing the data to support the queries submitted by BI tools. Instead of trying to wrangle big data into a form that works for BI queries, a vROLAP engine optimizes the queries to work with the data in its current form. It uses information about the data to get optimal performance from existing Hadoop resources.

vROLAP not only optimizes query performance, it shortens the entire 'time to insight' lifecycle. It removes the bottlenecks and complexity that have been a barrier to the widespread adoption of OLAP on Hadoop.



## About AtScale

AtScale is the first OLAP Engine for Hadoop. It allows non-technical business analysts to access data in HDFS and turn it into virtual, multi-dimensional ROLAP cubes ready for real-time analysis. Using standard ODBC/JDBC drivers, users can connect to the AtScale OLAP server from existing BI Tools, such as Tableau, Excel or Qlik. AtScale intercepts the queries issued from your BI tools, optimizes them, and executes them directly on the Hadoop cluster. It uses advanced machine-learning algorithms to optimize BI query workloads on-demand, and deliver the performance that users have come to expect from their legacy relational data warehouses and OLAP data marts.