Data Chunking - Duplicates are not real! (mostly)

Short version: @Sakthi Pitchaiah + the data engineering team use chunking to make our deadlines possible. Otherwise it would take much longer to get the data into the warehouse.

This process splits the source file into multiple files and loads them individually. At the end of each load, a record id is generated. And because of that, there can be duplicates of record_id for every source file.

But that does not mean the data itself is duplicated.

In the instance where there is too much data to load and not enough memory for the cluster to process all at once, what the DEs do for providers like PF is to use the cluster's entire computing capacity by splitting the source files into multiple files and loading them individually, one by one.

At the end of each individual load, row_id is generated for every source file so you will see a large number of row_id but this is not because there is duplicated rows of data, it is because duplicates of record_id can be generated for every source file that was split prior to individual loading

---

In the context of data warehousing, "chunking" refers to a data partitioning technique that involves dividing large sets of data into smaller, more manageable pieces called "chunks" or "data chunks." **This process is also known as data segmentation or data partitioning.**

The main purpose of chunking data in data warehousing is to improve data processing efficiency, data retrieval speed, and overall system performance. When dealing with large volumes of data, processing the entire dataset at once can be resource-intensive and time-consuming. Chunking allows data to be split into smaller units, making it easier for systems to handle and process the data more efficiently.

Here's a high-level overview of how chunking data works in data warehousing:

1. **Dividing Data into Chunks:** In the context of data warehousing, the data is typically organized in tables or datasets. Chunking involves breaking down these large tables into smaller, more manageable chunks based on a specific criterion.
2. **Key-Based Chunking:** One common approach is key-based chunking, where data is partitioned based on specific column values, such as date ranges, alphabetical ranges, or numerical ranges. For example, a sales table could be partitioned into monthly chunks, where each chunk represents sales data for a particular month.
3. **Equal Size Chunks:** Another approach involves dividing the data into equal-sized chunks, irrespective of the values in the data. This method can be useful when there is no clear natural partitioning based on key values.
4. **Chunk Metadata:** Each chunk is typically associated with metadata that describes its characteristics, such as the range of key values it contains. This metadata is essential for efficiently querying and retrieving data from the chunks.
5. **Parallel Processing:** Chunking enables parallel processing, where multiple chunks can be processed simultaneously by different computing resources, such as servers or nodes in a distributed system. This parallelism significantly improves data processing performance.
6. **Data Pruning:** In some cases, chunks that are not needed for analysis or reporting can be pruned or excluded from processing, further optimizing system performance.
7. **Data Maintenance and Loading:** Chunking can also facilitate efficient data maintenance processes. For example, when new data is added to the data warehouse, it can be directly inserted into the appropriate chunks based on the key values.

Overall, chunking is an essential technique in data warehousing to optimize data processing and retrieval, especially when dealing with large volumes of data. By dividing the data into smaller, manageable units and leveraging parallel processing, data warehousing systems can deliver faster and more efficient analytical capabilities to users.

Chunking data in data warehousing can lead to the perception of duplicates or even create actual duplicates under certain circumstances. This is primarily due to the way data is divided and how it interacts with the chunking process. Examples of duplication:

1. **Boundary Overlap:** If data chunks are created based on key ranges, there may be overlapping boundaries between adjacent chunks. For example, if a sales table is partitioned by month, some transactions that fall on the border between two months could end up being duplicated in both chunks. This is because the same data record satisfies the criteria for both chunks.
2. **Parallel Processing Issues:** When parallel processing is involved, multiple chunks may be processed simultaneously by different computing resources. If a query or operation is not precisely designed to handle such parallelism, there is a possibility of processing the same data in multiple chunks, resulting in perceived duplicates.
3. **Data Loading Errors:** During the data loading process, if there are errors or issues, data might be mistakenly loaded into multiple chunks. For example, if a data loading operation is interrupted and restarted, the same data might be inserted again into different chunks.
4. **Merge and Split Operations:** In some cases, chunks might be merged or split to optimize data storage or performance. During these operations, data overlaps might occur, leading to perceived duplicates.
5. **Data Model Evolution:** As the data model evolves, changes in key values or partitioning criteria might be introduced. If not managed carefully, these changes can lead to data appearing in multiple chunks, causing confusion and perception of duplicates.
6. **Query Complexity:** In complex queries that involve joining or combining data from multiple chunks, improper handling of joins or aggregations can lead to results that appear as duplicates.

To mitigate the perception of duplicates and actual duplicates, data warehousing systems need to implement appropriate strategies:

- **Chunking Strategy Design:** Careful consideration should be given to the design of chunking strategies. Boundary overlaps should be minimized, and partitioning criteria must be chosen wisely to avoid data duplications.
- **Data Deduplication:** Implement data deduplication mechanisms to identify and eliminate actual duplicates caused by chunking or any other data integration process.
- **Metadata Management:** Accurate metadata about each chunk should be maintained, allowing queries and processing operations to avoid redundant data access.
- **Data Quality Assurance:** Thoroughly validate data loading processes and ensure error-free data insertion to prevent accidental duplicates.
- **Query Optimization:** Optimize queries to handle parallelism and chunk boundaries correctly, ensuring that results are accurate and do not include duplicates.
- **Regular Maintenance:** Regularly monitor and maintain data chunks to ensure data consistency and accuracy.

By carefully managing the chunking process and implementing proper controls, duplicates can be minimized, and the perception of duplicates can be eliminated in data warehouses.

References:

Data Warehousing Fundamentals