
ORACLE®

GDPR: Fine Grained In-file Authorization with Apache Parquet

Jean-Pierre Dijcks
Big Data Product Management

@jpdijcks

ORACLE

Use Cases

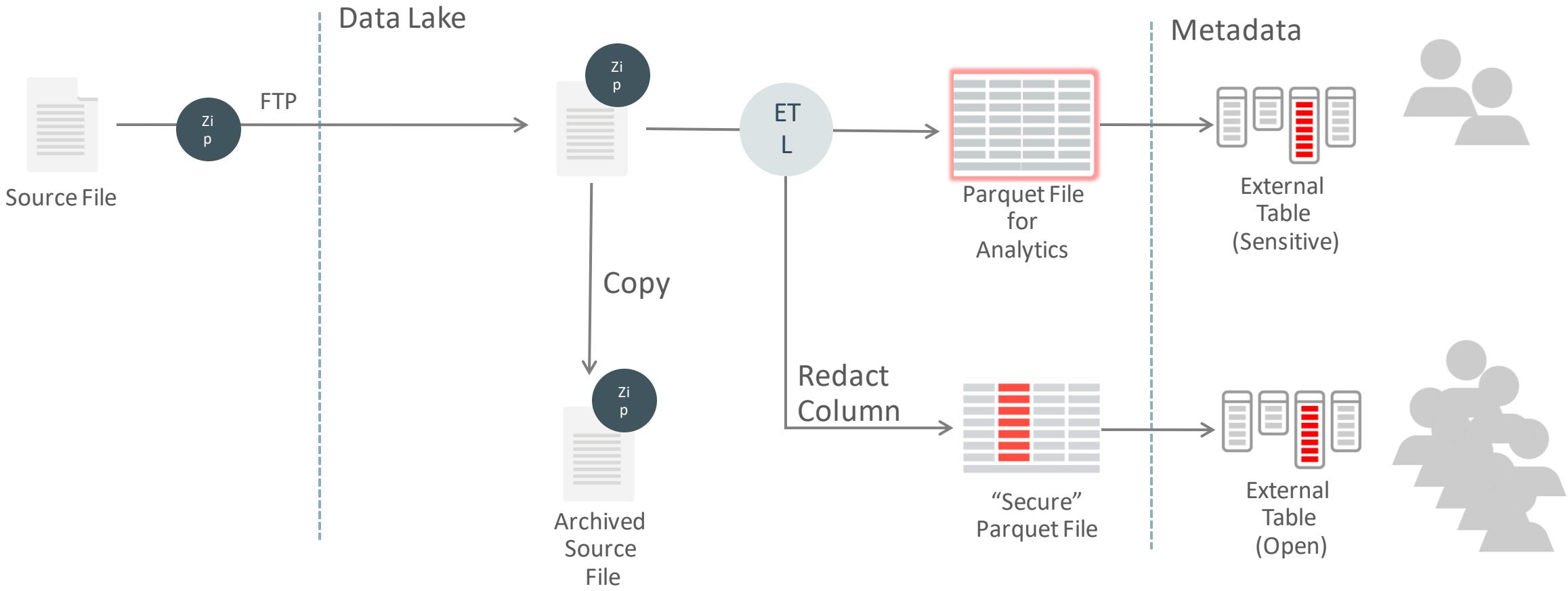
- The entire presentation focuses on Analytics
- The use cases involve file based data stores, like:
  - Hadoop Distributed File System (HDFS)
  - Object Stores
- As well as integration with Oracle Database

Files: Contradictions and Problems

- Files come in (many) multiples
- Container security limitations
  - Without knowing the file contents and structure, only course grained authorization is possible:
    - Protect an object store bucket / folder (All of nothing)
    - Protect a file (Access Control Lists)
- Moving data
  - Removes the security because the security is not tied to the file / data
- The metadata conundrum of schema-on-read:
  - Files are stored, and only defined upon reading that data. So, how to deal with:
    - Metadata, specifically Fine-Grained Access Control?
    - Performance?

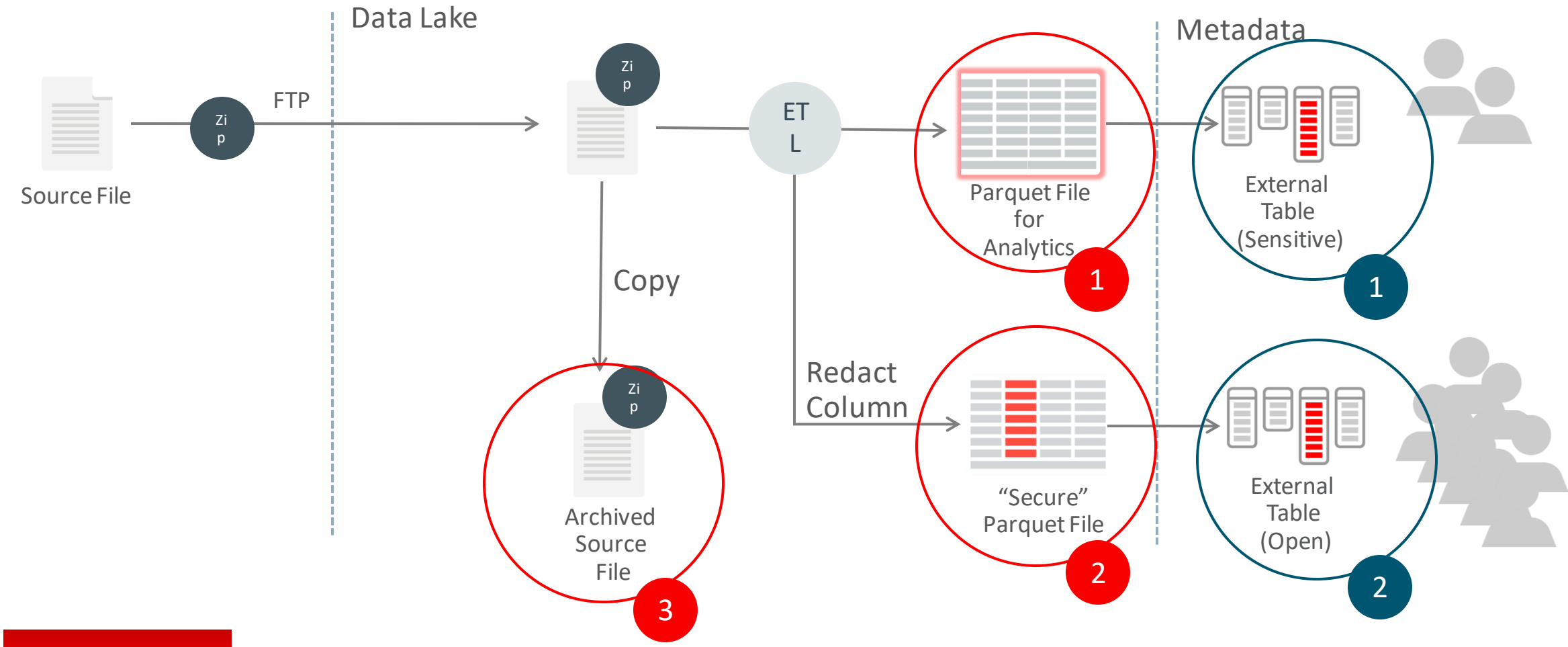
Data on Disk – Issues in Object Store (and HDFS)

Many file copies on disk, some are secure, some are not



Data on Disk – Issues in Object Store (and HDFS)

Many file copies on disk, some are secure, some are not

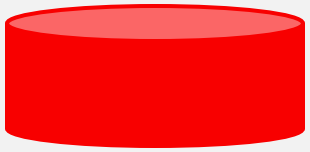


Data Access Fragmentation

Industry Leading Security, Compliance and Governance

How to provide equivalent security across other data stores and engines?

Oracle SQL



Data Warehouse

Object Store

Hadoop/HDFS

Data Lake

Possible Solutions to the Fragmentation Conundrum

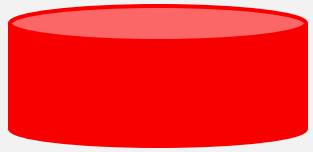
1. Create a single interface to all data
   - Oracle Solution: Leverage Oracle Big Data SQL to extend Oracle Database Security across other data sources
2. Create a broker as an intermediary to all Query Engines
3. Push-down security into the data assets
   - Oracle Solution: Deploy a net-new technology invention to embed fine-grained authorization INSIDE files across file based repositories

Single Interface with Big Data SQL

Industry Leading Security,
Compliance and Governance

Oracle Big Data SQL

Enable Oracle Security
across diverse stores



Data Warehouse

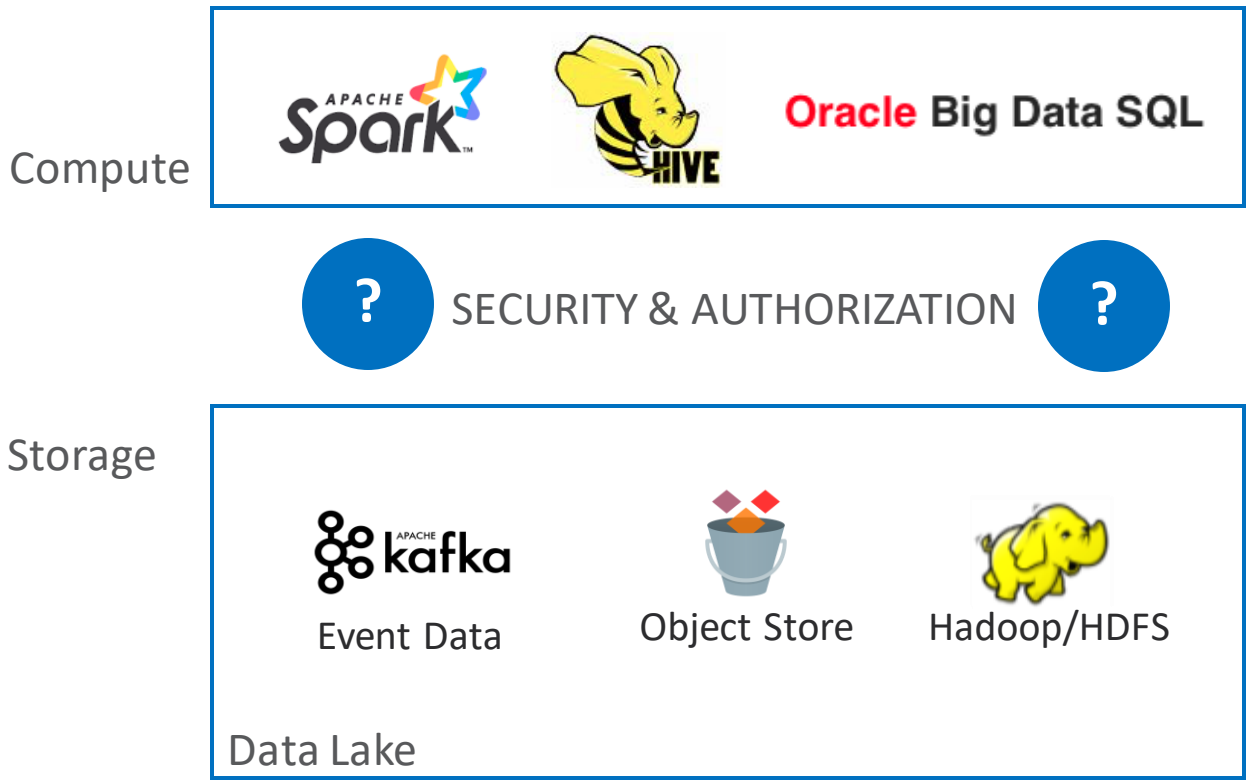
Event Data

Object Store

Hadoop/HDFS

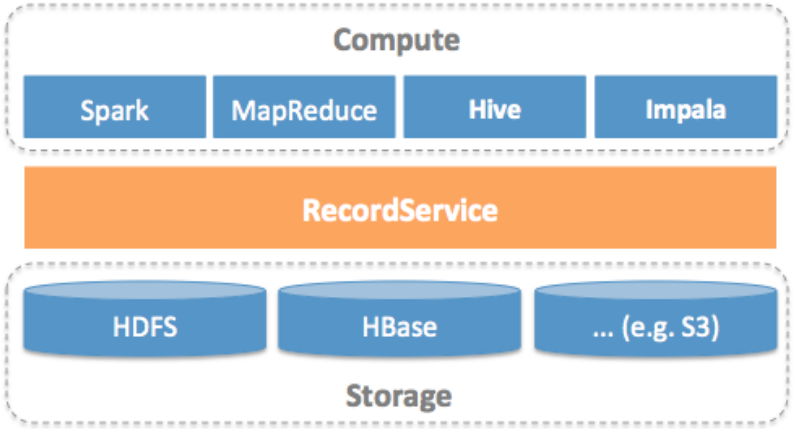
Data Lake

Securing File Based Repositories with Many APIs and Engines consuming Data



Solution Paths

- Security Broker “in the middle”
  - Bump in the wire causes Performance issues
  - Does not fundamentally alter the problem space (can be circumvented)
  - Has not succeeded in this environment!



Cloudera RecordService

- Embed “all of it” in the File/Data
  - How to do this in a schema-on-read paradigm?
  - What about performance?
  - How to parse data?
  - Adoption?
  - Etc.
- No one has solved this… Until now…

Solutions

Schema to the Rescue: Apache Parquet and ORC

- Does Solve:
  - Performance Issues by implementing Schema on Write:
    - Parsing data on write eliminates recurring query costs
    - Offering Columnar Structures ideally suited for analytics
    - IO avoidance both columnar and based on predicates and other smarts
  - Metadata issues
    - Self describing data with a schema (schema on write)

- Does NOT Solve:
  - In-file fine-grained access
    - Relies on external security metadata (Apache Sentry etc.) to ensure access controls
    - Redaction / Tokenization of sensitive columns (note project underway in community to address encryption)
    - File Sprawl, as customers often create a Parquet file for each new use case
  - Schema on Read Performance
    - Parquet / ORC are schema on write, not read
  - Auditing
    - Parquet does not contain all original data, so customers keep original files, often in HDFS and Object Store

Schema to the Rescue – Not really!

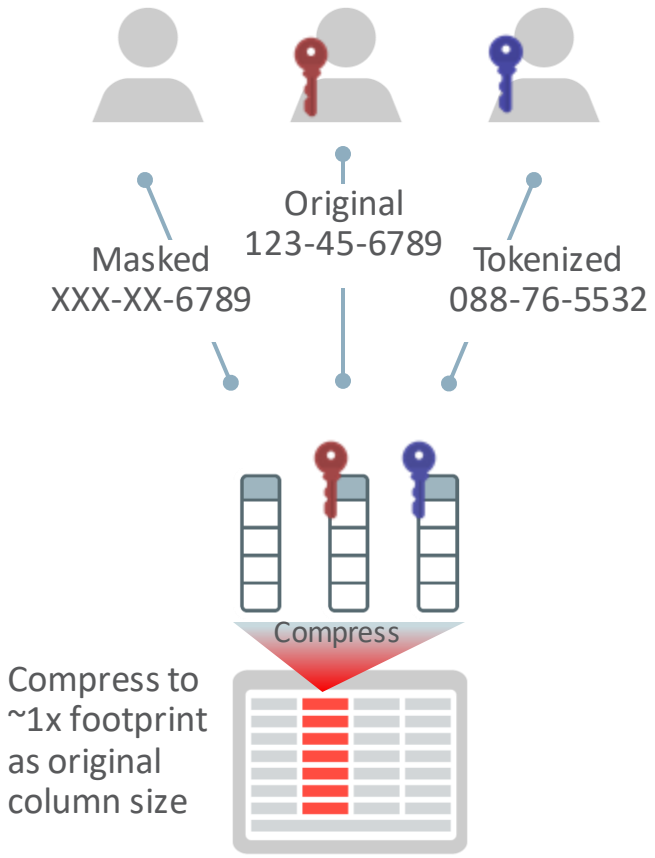
- We are now back to ETL is done first
  - Data must be well-understood
  - and we loose agility
- And we still have not solved the authorization issues
  - Must use that single entry point
  - Files, security rules and the engines are split
  - Copying files will remove security in many cases
  - Original data is stored (for auditing purposes) and (re-)used

Parquet with Enterprise Features

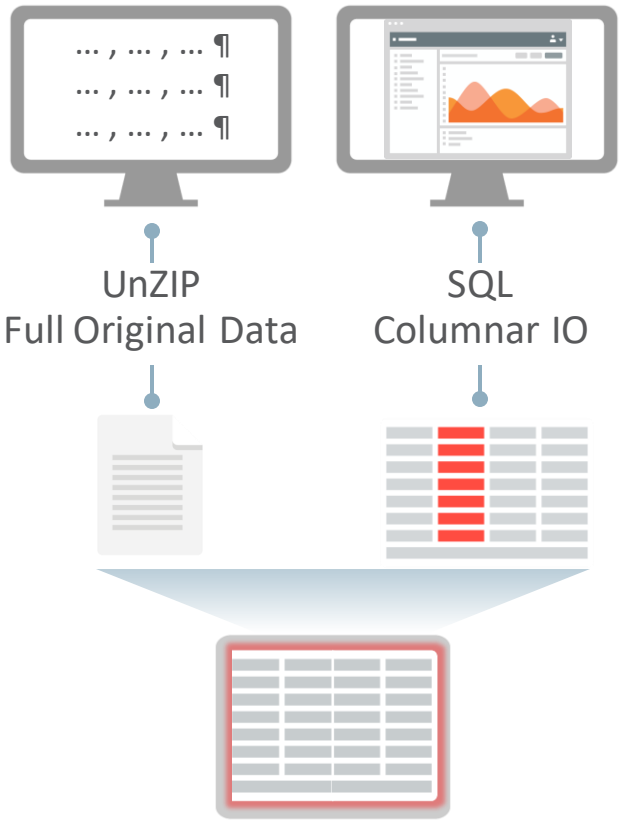
- Builds on Apache Parquet and is 100% Apache Parquet **Read Compatible**
- Enriched with Additional Enterprise Features (Oracle Intellectual Property) to deliver:
  - All benefits of Apache Parquet
    - Same Columnar IO and Performance as Apache Parquet
    - Any reader can read the proposed format as an Apache Parquet file
  - Security: Fine-grained access control **embedded** in the file, also enables redaction/tokenization etc. while keeping the storage footprint well below the original source file
  - Compliance: Full **archive** of all data, capable of complete unzip to original data with provenance
  - Agility: Enables **both** Schema on Write **and** Schema on Read on a **single** Enterprise Parquet file
- Does **not** require Hadoop to create, read or unzip files

Parquet with Enterprise Features provides Complete Solution

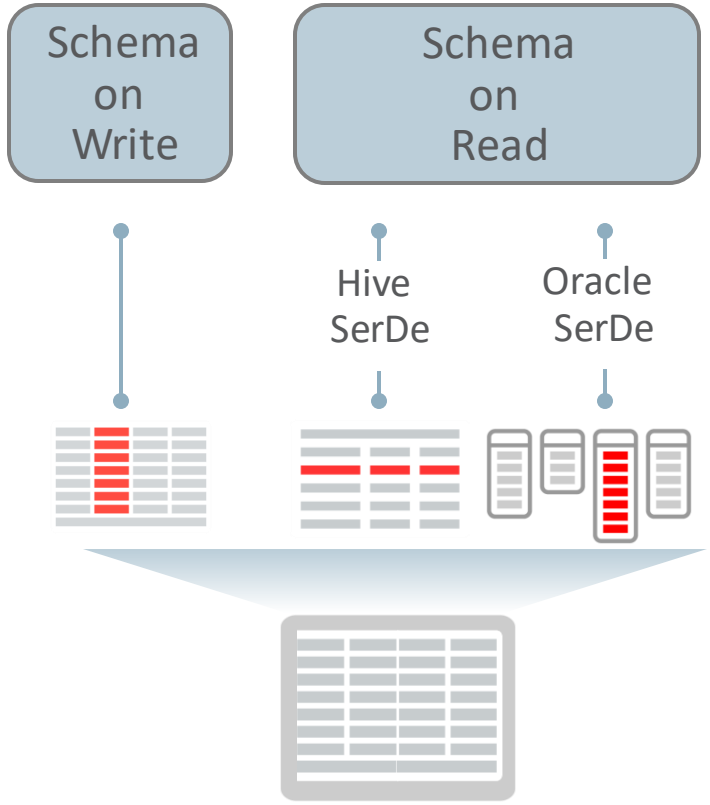
Fine grained, in-file access control



Compliance + Performance

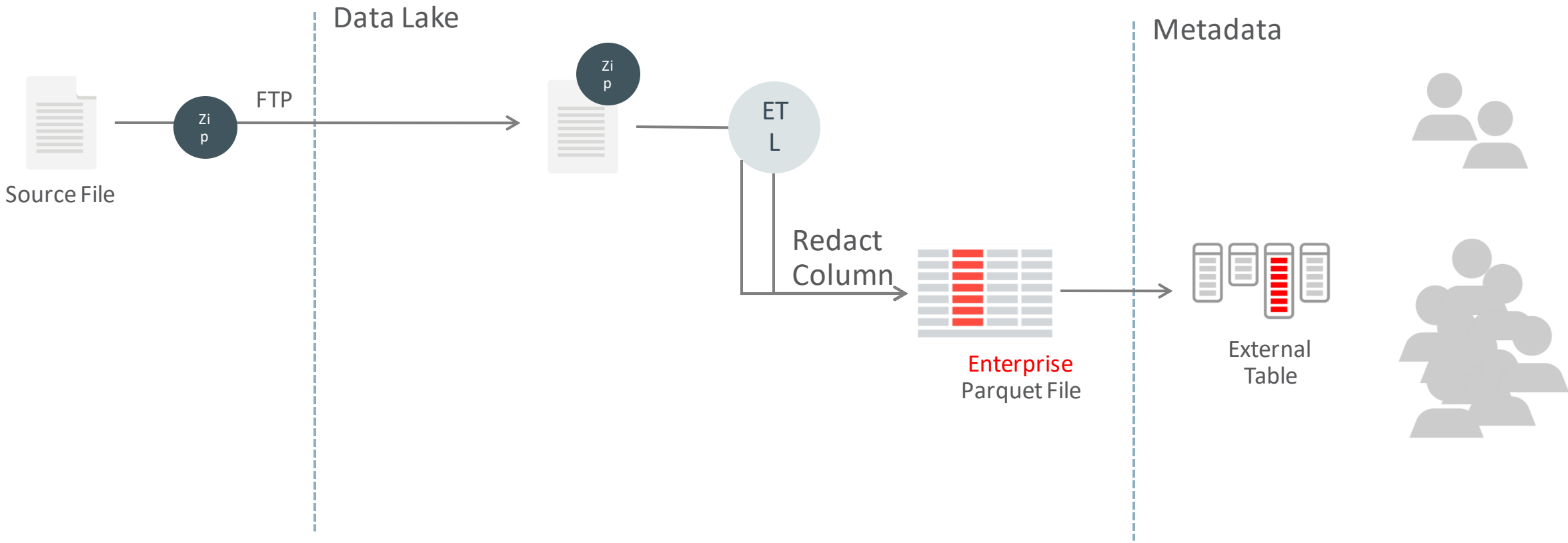


Agility: Fast Schema-on-Read



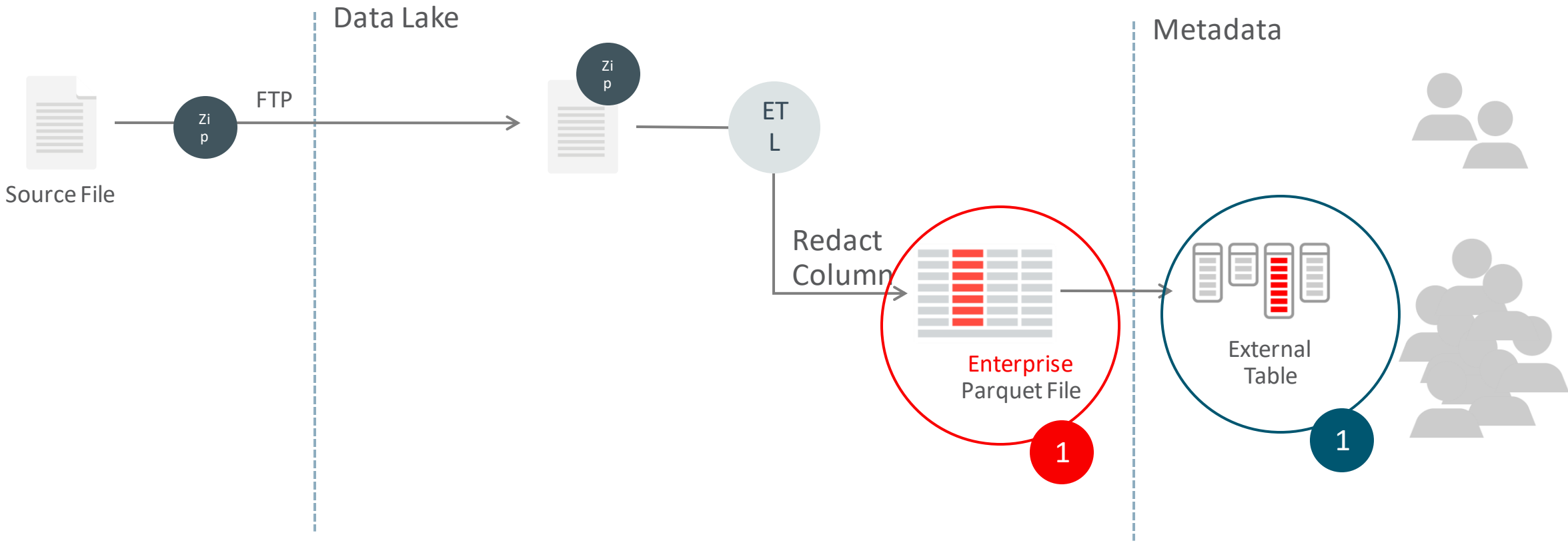
Data on Disk – Enterprise features for Apache Parquet

Dramatically simplified – Single File – Single Table – Always Secure

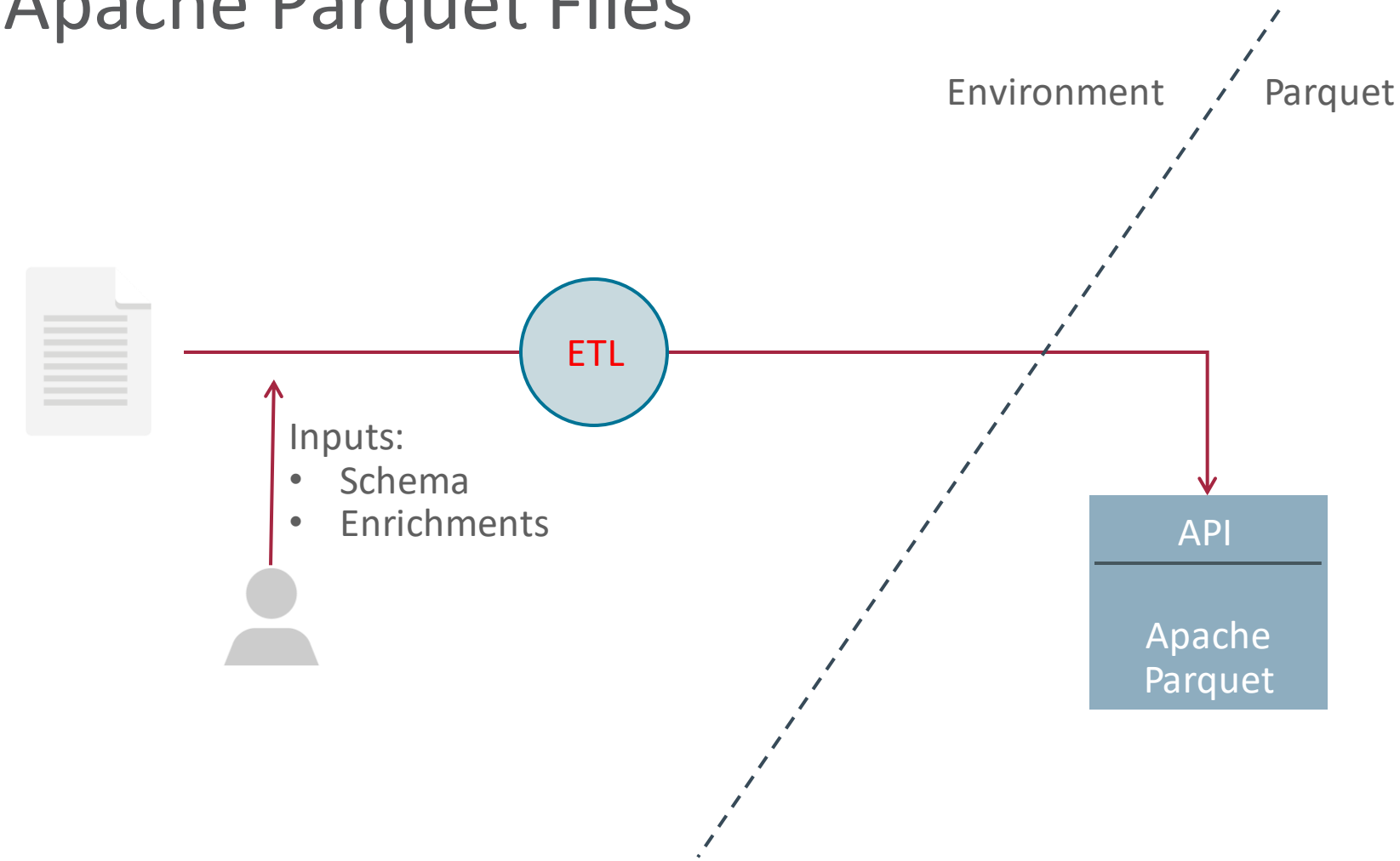


Data on Disk – Enterprise features for Apache Parquet

Dramatically simplified – Single File – Single Table – Always Secure

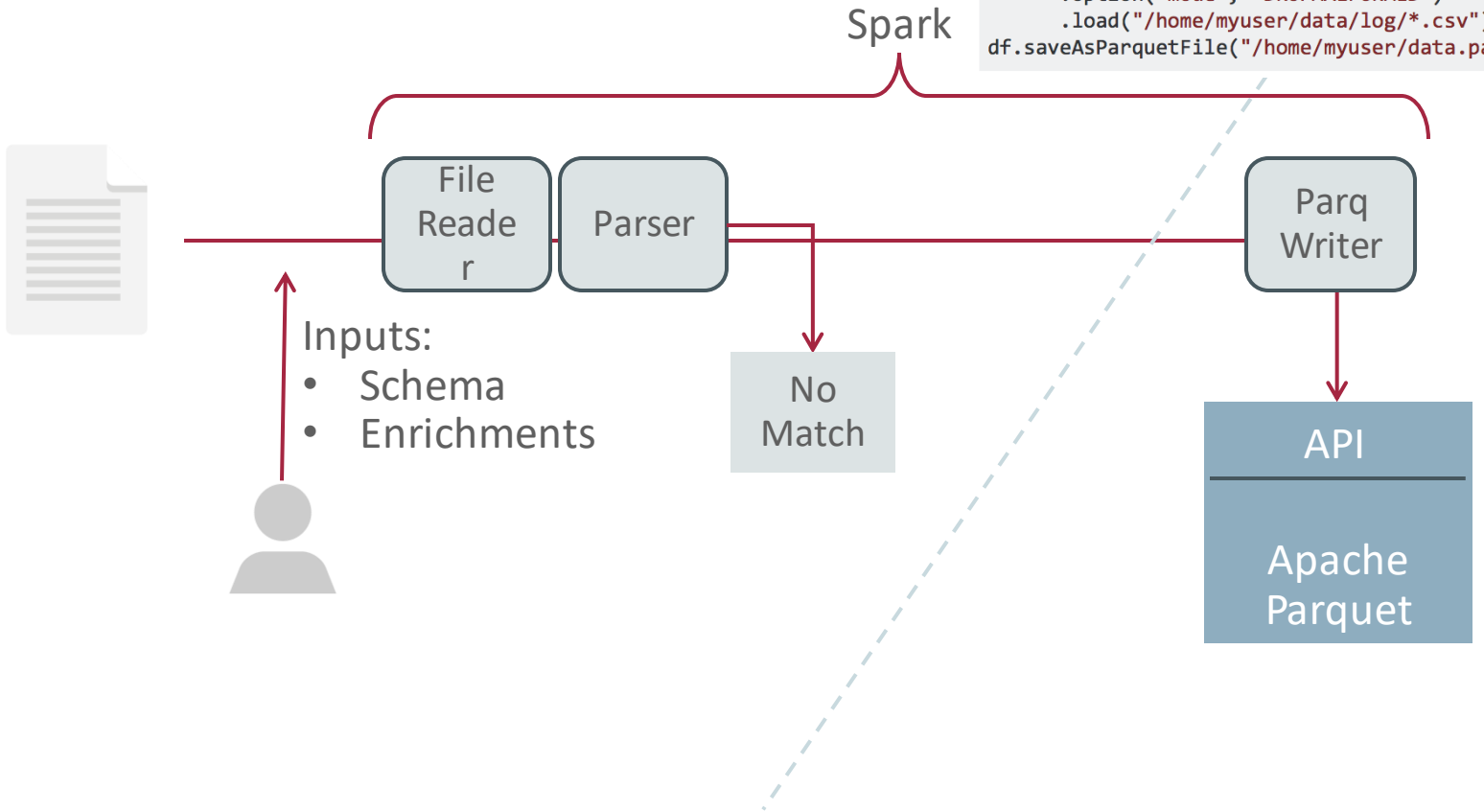


Writing Apache Parquet Files

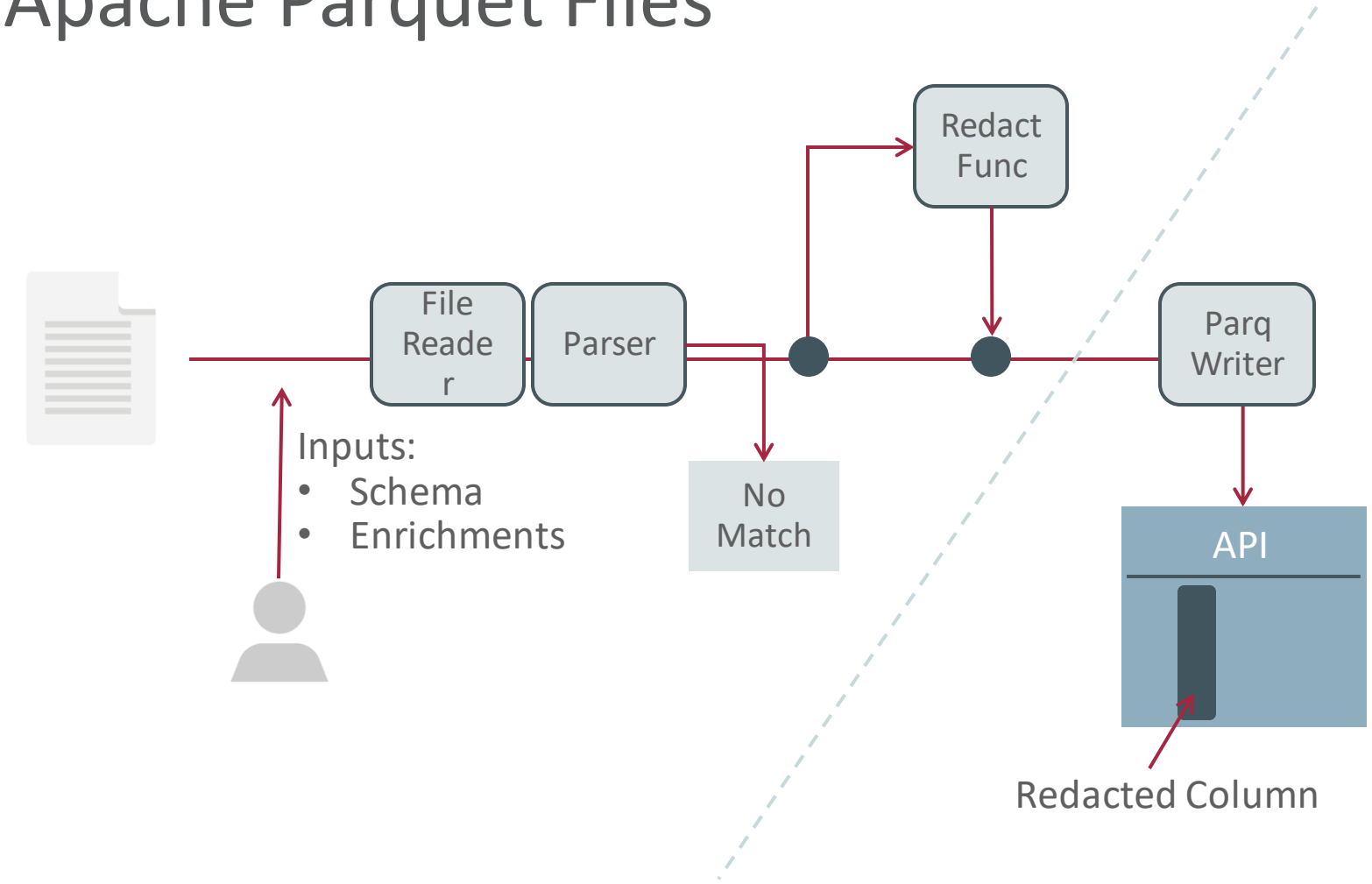


Writing Apache Parquet Files

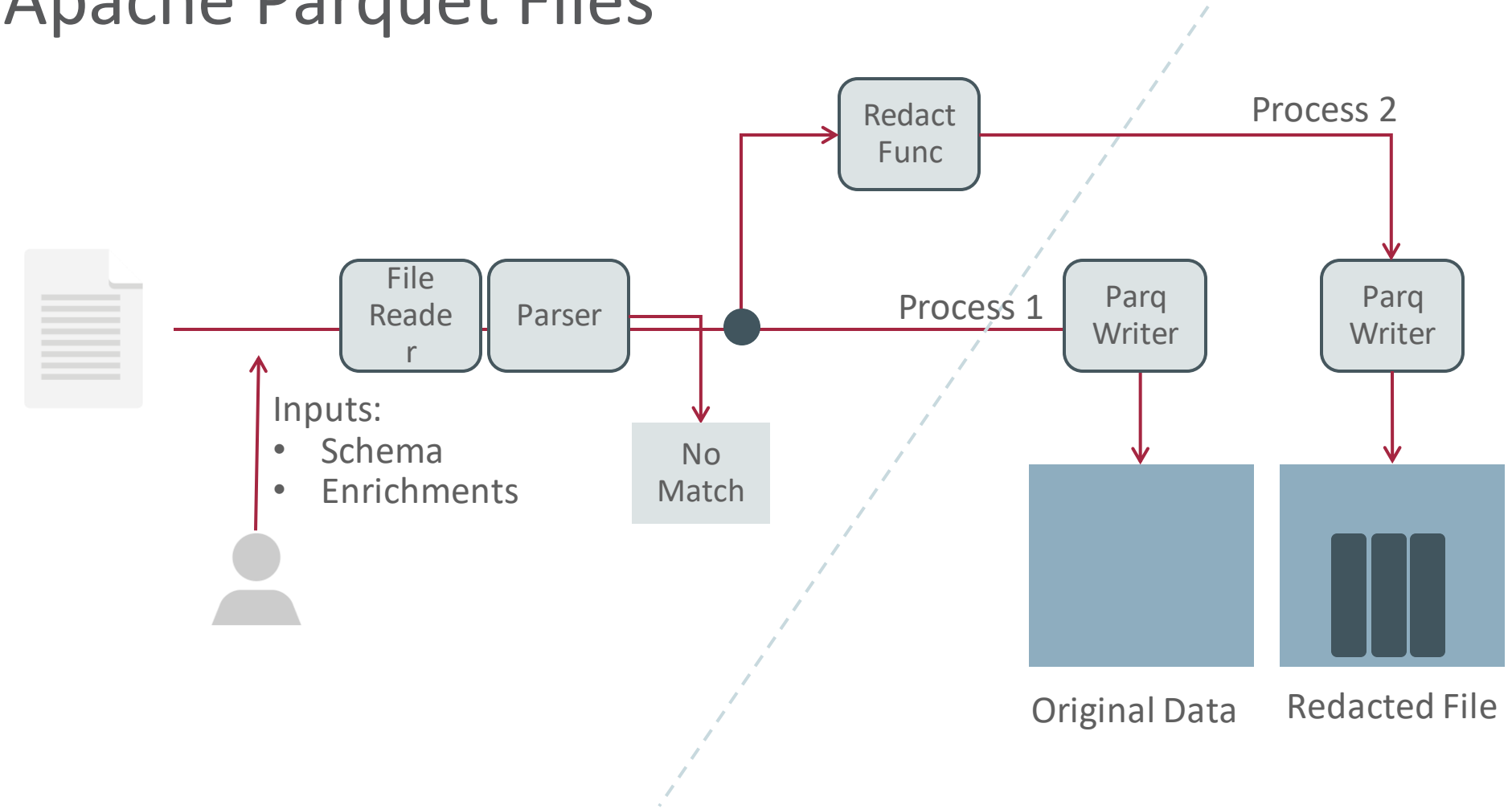
```
val df = sqlContext.read
      .format("com.databricks.spark.csv")
      .option("header", "true")
      .option("inferSchema", "true")
      .option("mode", "DROPMALFORMED")
      .load("/home/myuser/data/log/*.csv")
df.saveAsParquetFile("/home/myuser/data.parquet")
```



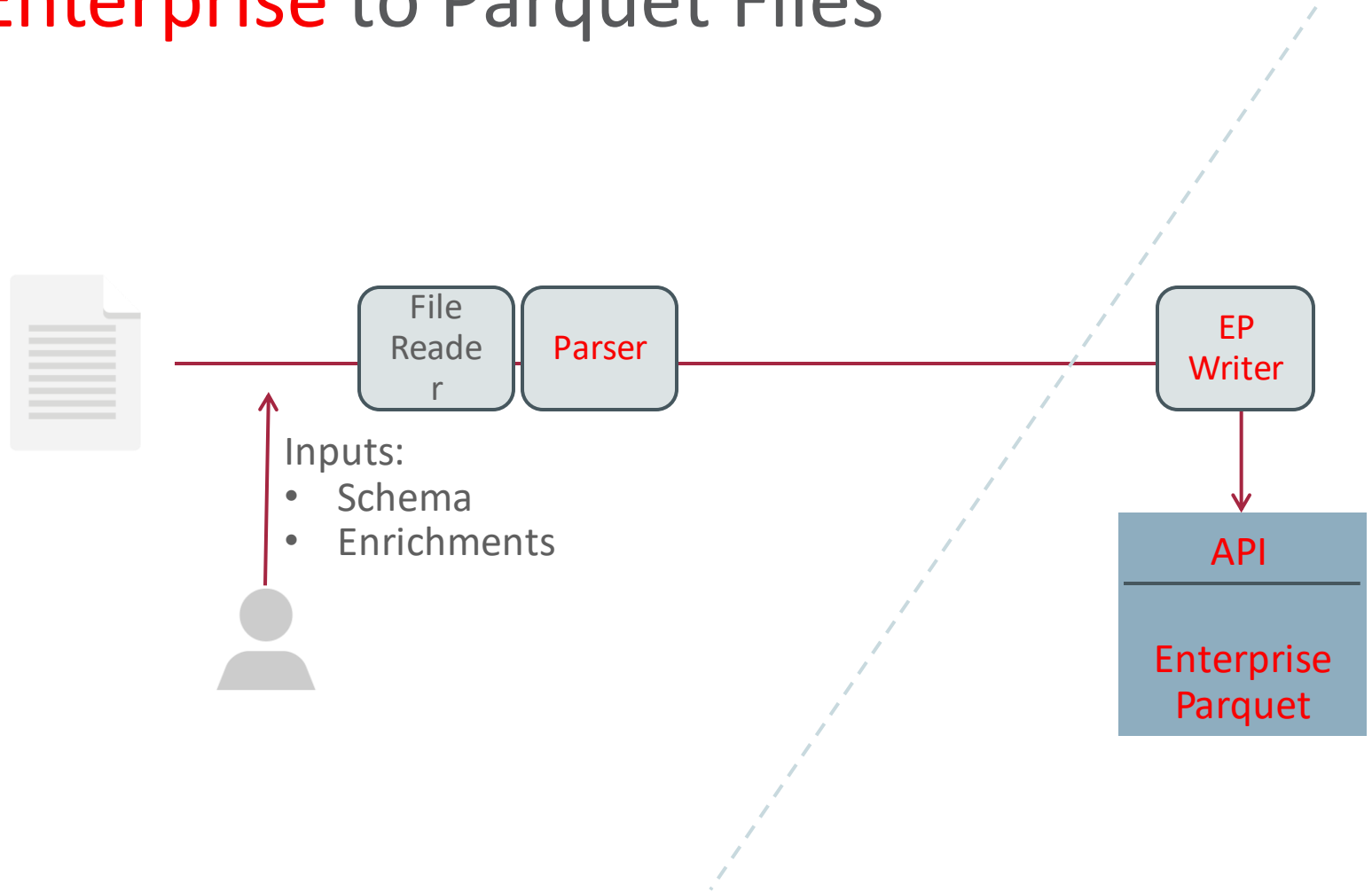
Writing Apache Parquet Files



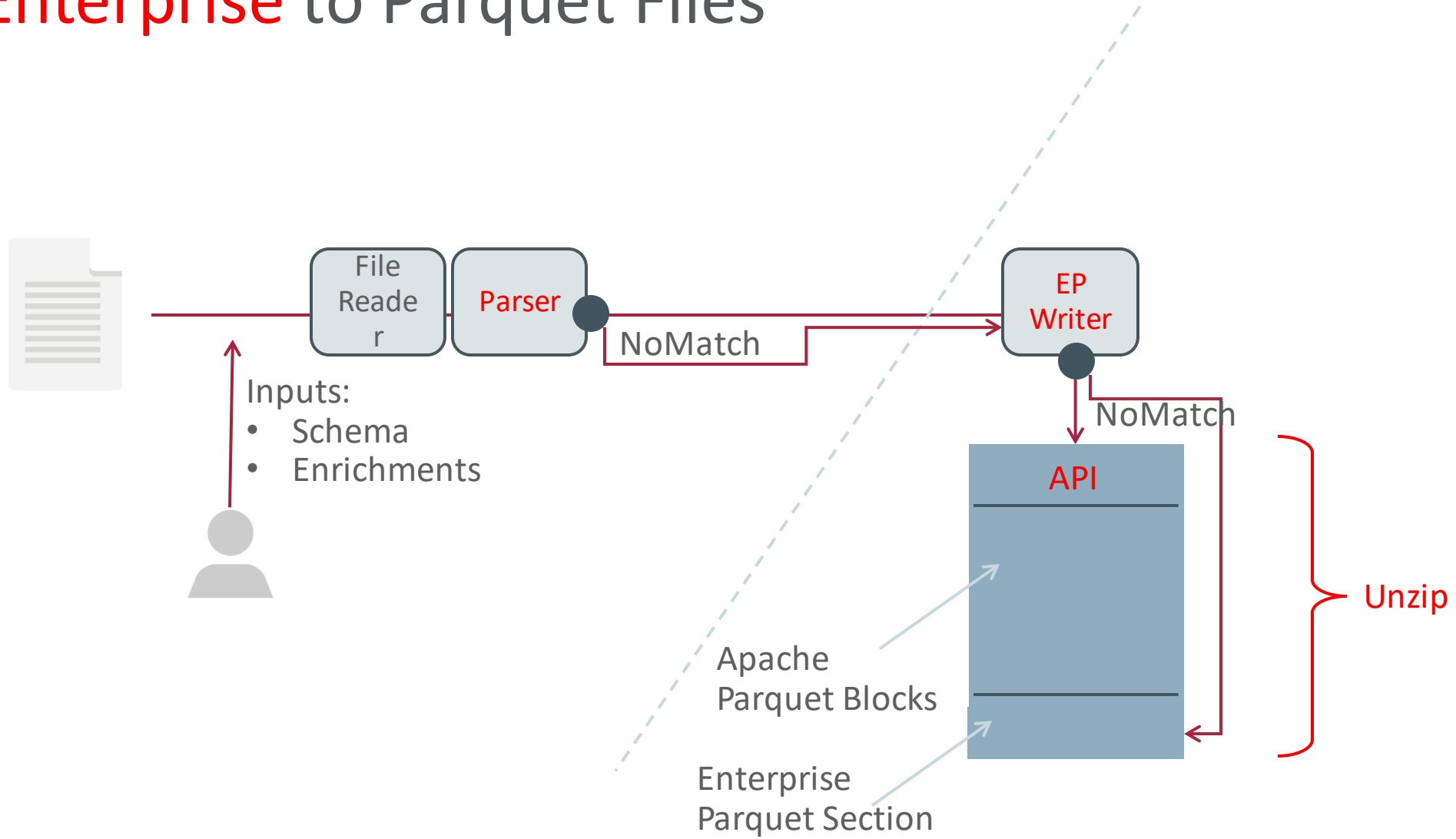
Writing Apache Parquet Files



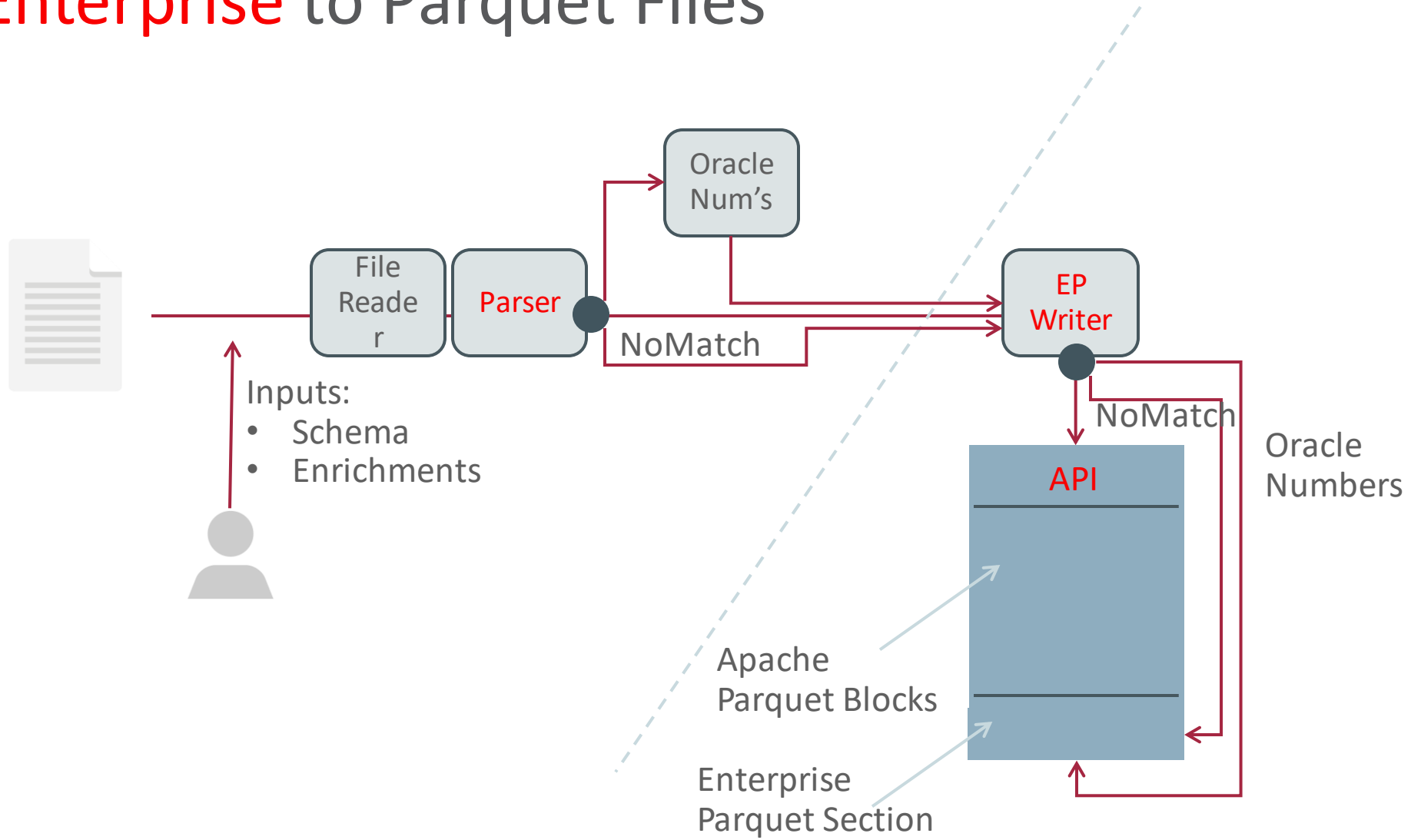
Adding Enterprise to Parquet Files



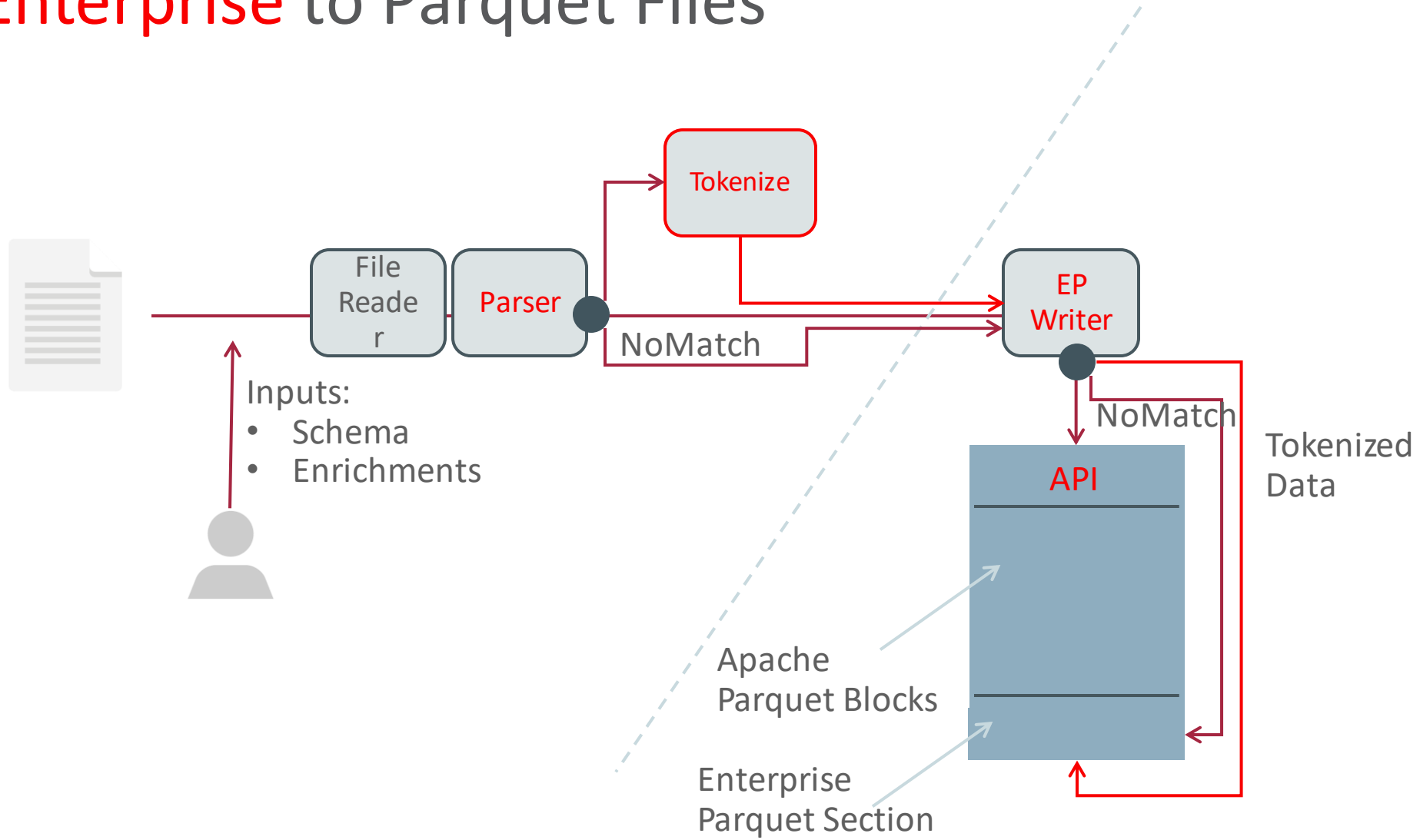
Adding Enterprise to Parquet Files



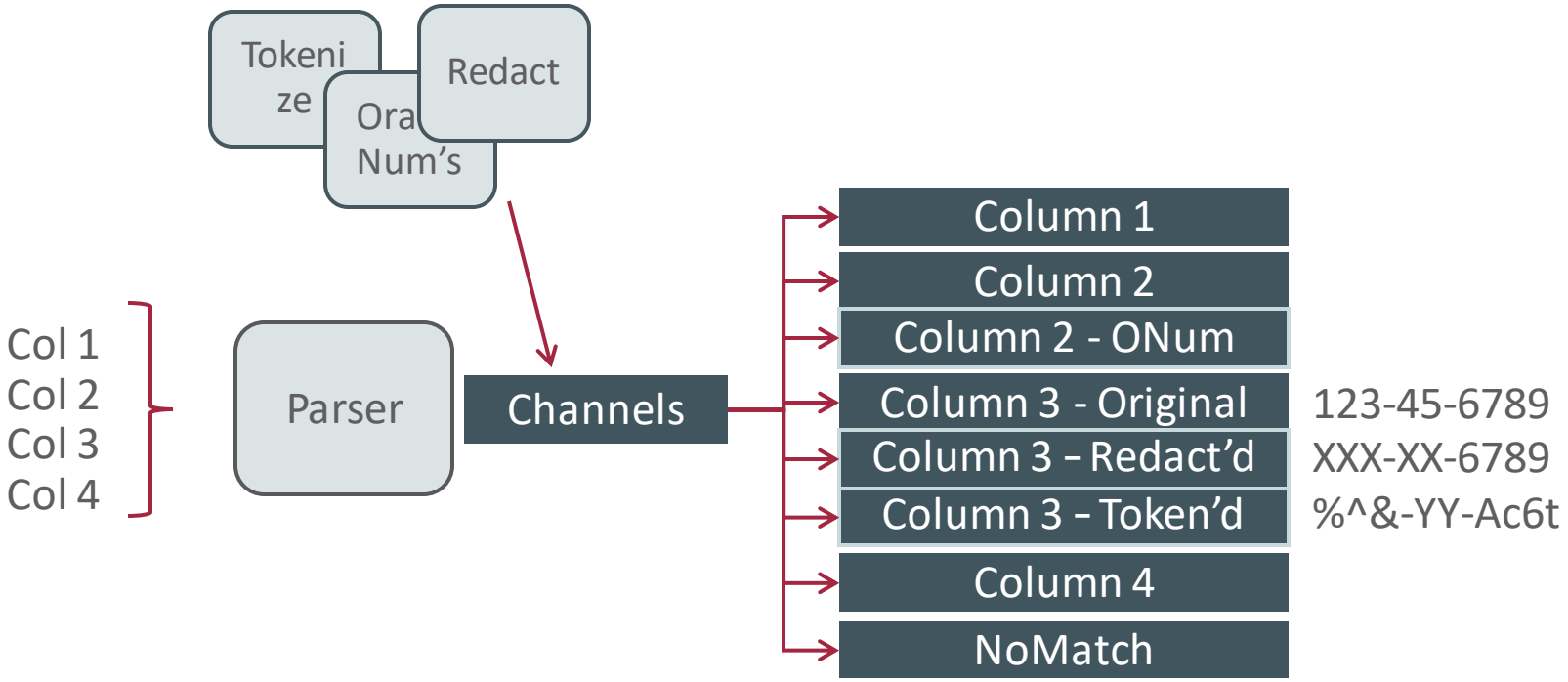
Adding Enterprise to Parquet Files



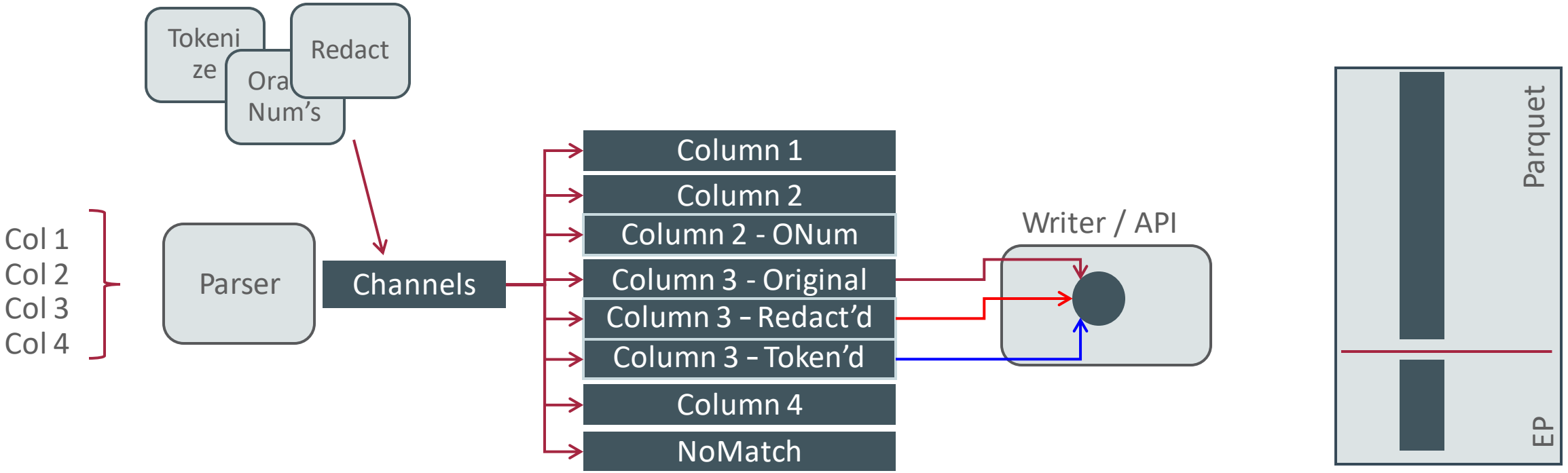
Adding Enterprise to Parquet Files



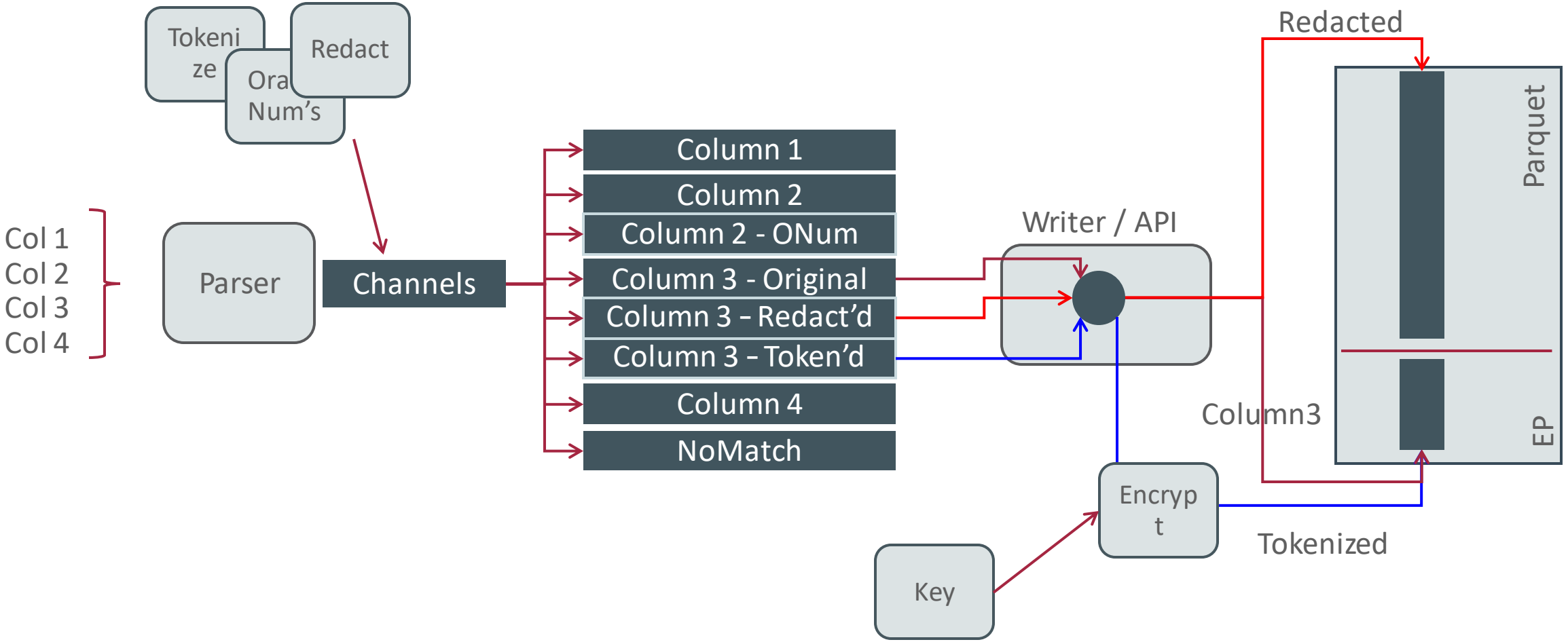
Adding Enterprise to Parquet Files – Channels



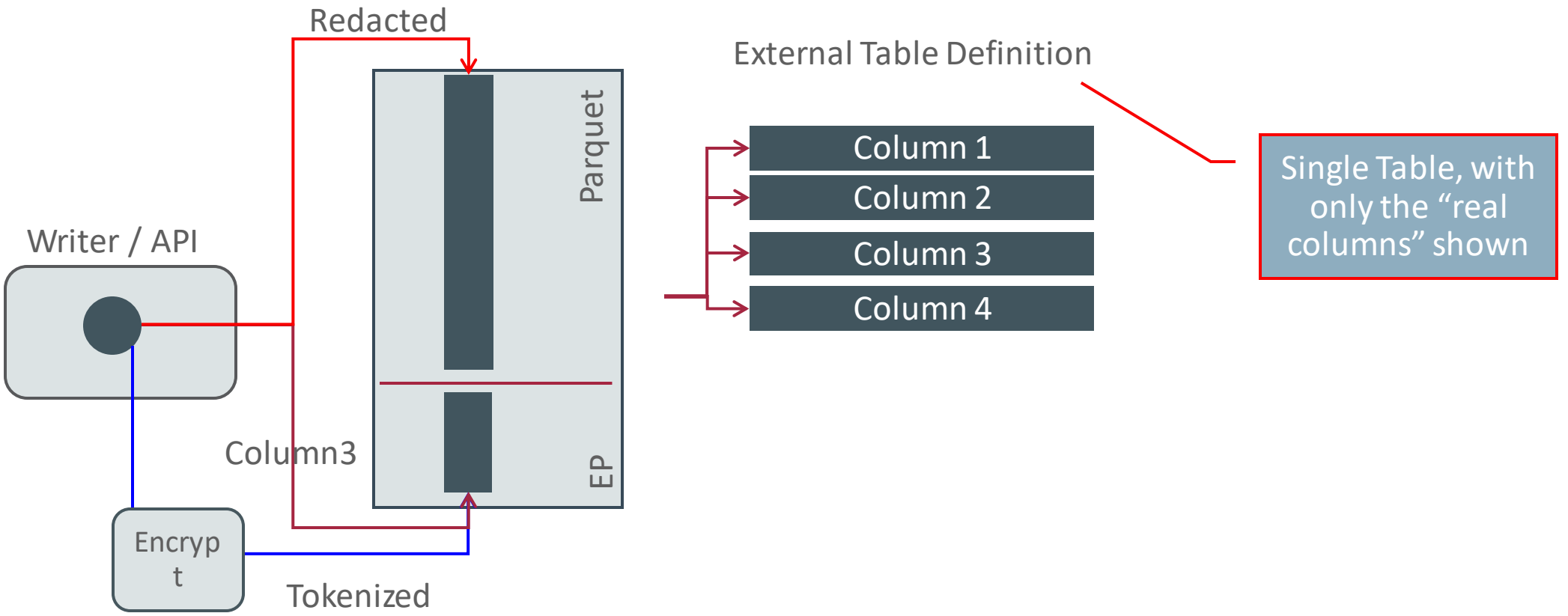
Adding Enterprise to Parquet Files – Feed to API/Writer



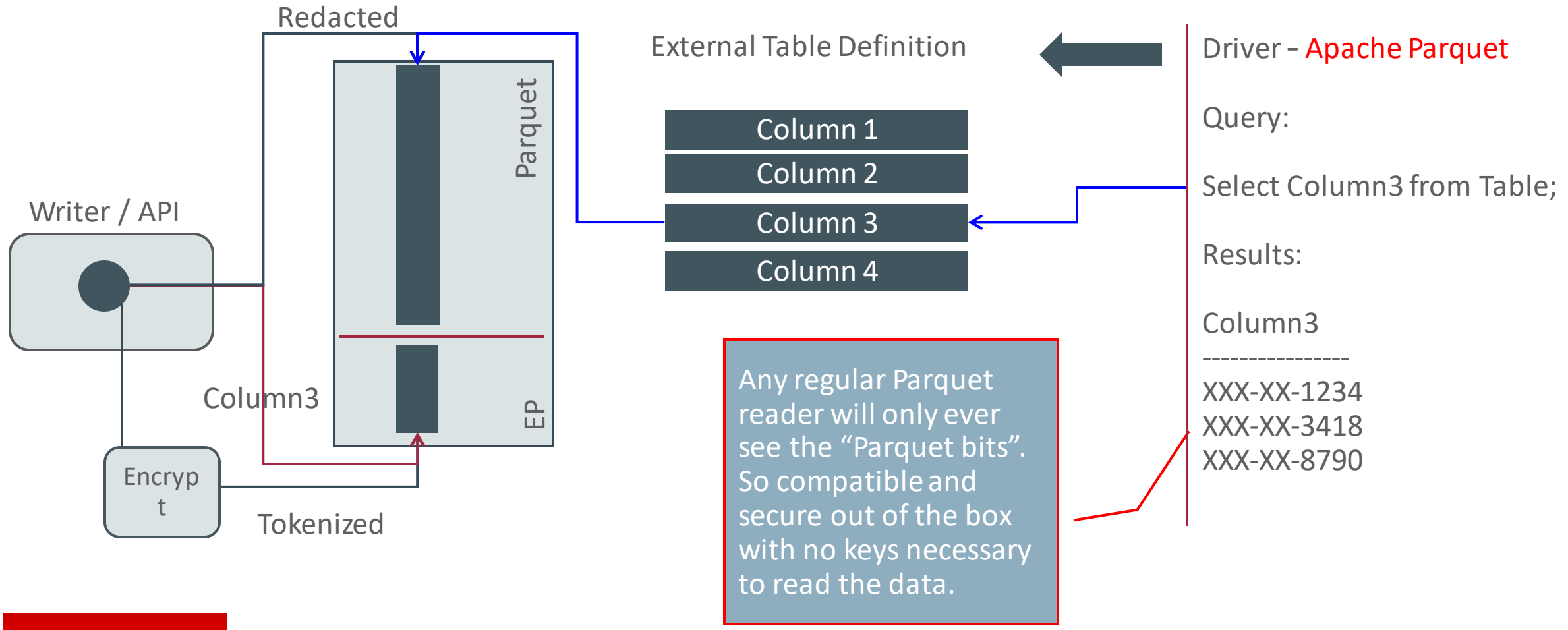
Adding Enterprise to Parquet Files – Key Fetching



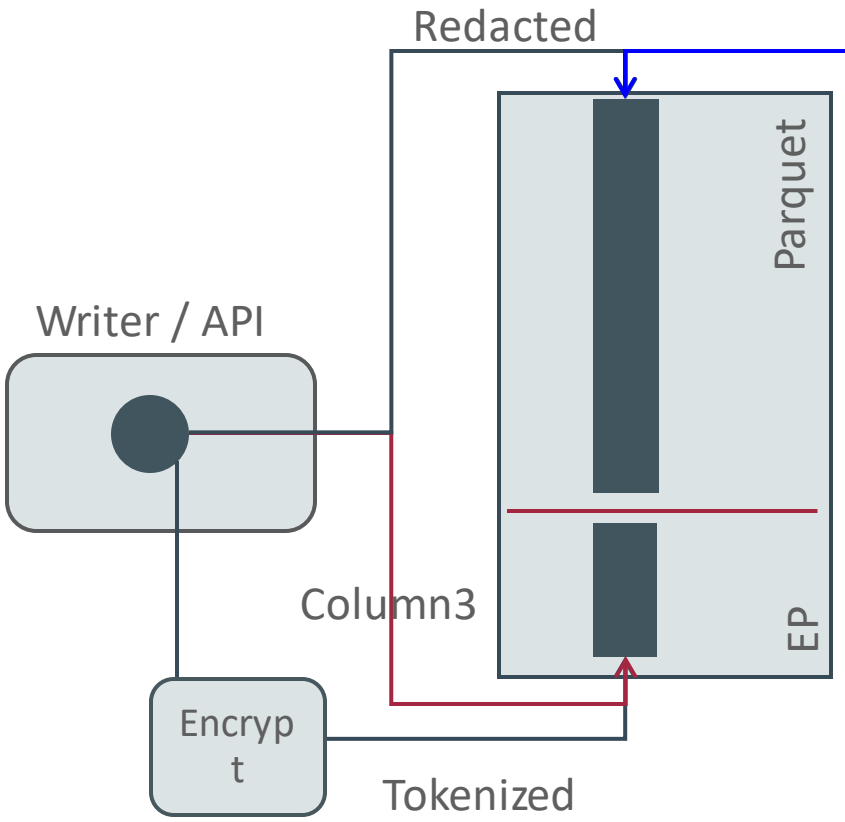
Adding Enterprise to Parquet Files – Metadata



Adding Enterprise to Parquet Files – Read Compatibility



Adding Enterprise to Parquet Files – Open Data Read



External Table Definition

- Column 1
- Column 2
- Column 3
- Column 4

Driver – Parquet w/ EntFeat
User - Non-privileged

Query:

Select Column3 from Table;

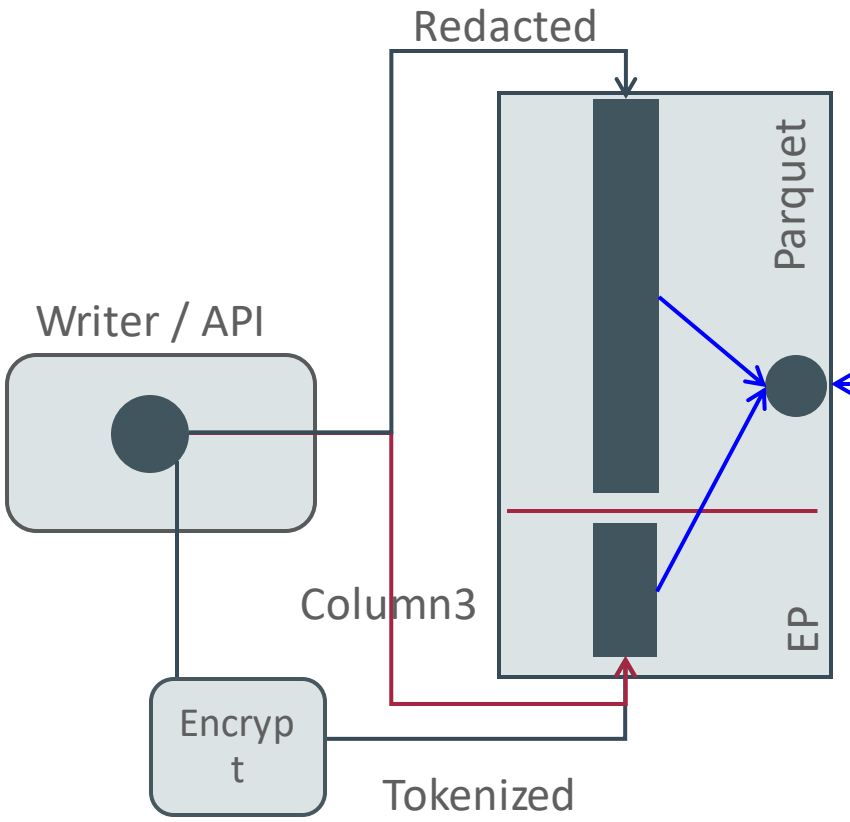
Results:

Column3

XXX-XX-1234
XXX-XX-3418
XXX-XX-8790

A non-privileged users sees the data in the open partition. In this case redacted and no need for keys. This would also what any standard parquet reader sees

Adding Enterprise to Parquet Files – Secure Data Read



External Table Definition

Column 1
Column 2
Column 3
Column 4

Driver – Parquet w/ EntFeat
User - Super Privileged

Query:

Select Column3 from Table;

Results:

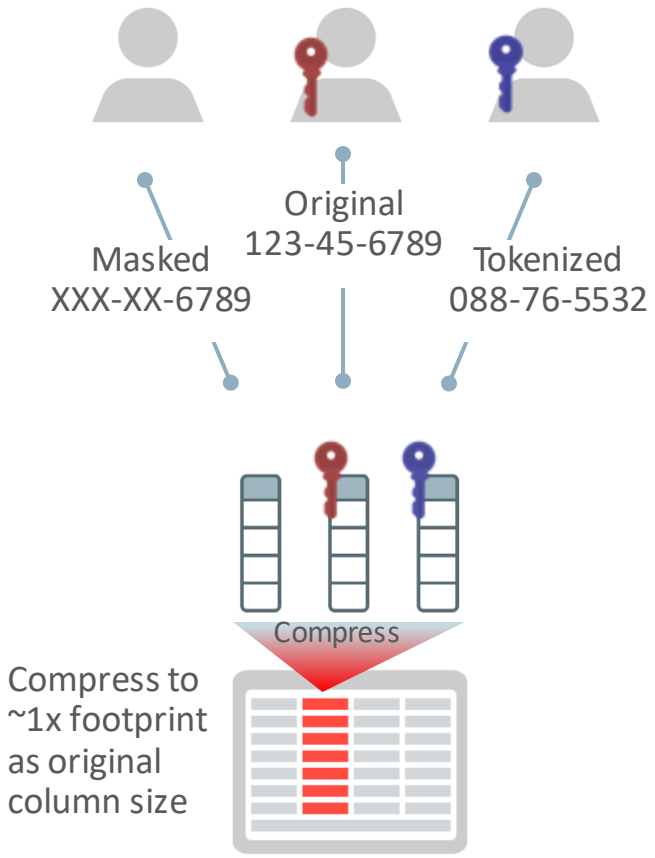
Column3

123-12-1234
659-11-3418
987-33-8790

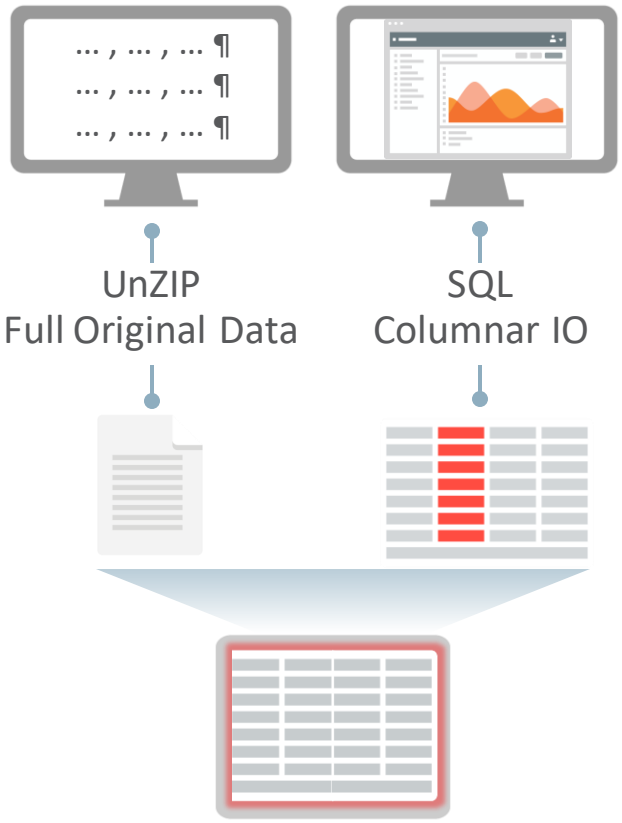
A privileged users sees the data created from both sections. A key is required with the query. This data is never visible without an EP Reader

Parquet with Enterprise Features provides Complete Solution

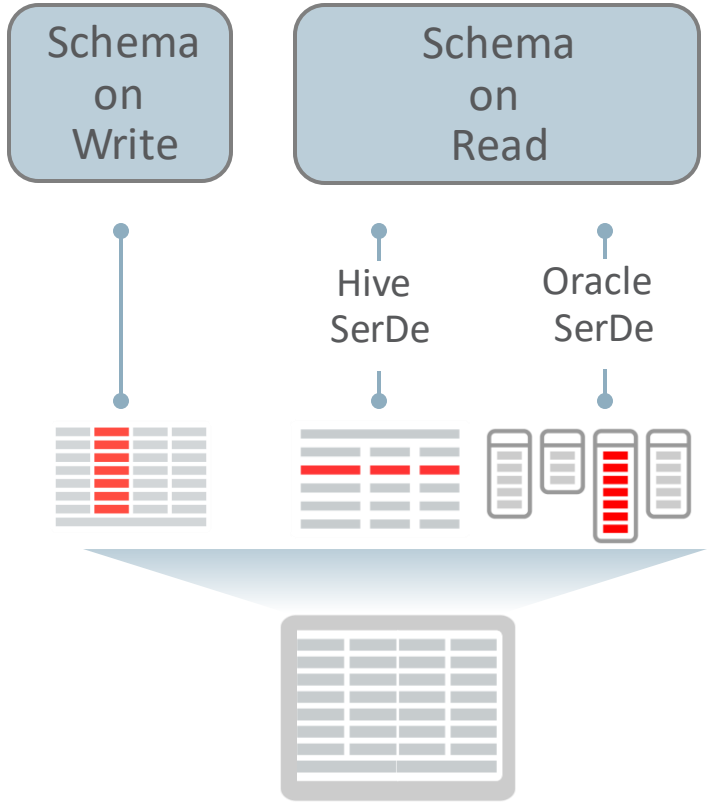
✓ Fine grained, in-file access control



✓ Compliance + Performance



✓ Agility: Fast Schema-on-Read



Integrated Cloud

Applications & Platform Services


ORACLE®