PERFORMANCE TUNING

- We add indexes , primary keys.
- We create table partitions.
- We analyze query execution plan.
- Remove unnecessary large-table full-table scans
- Cache small-table full-table scans
- Verify optimal index usage
- Using hints to tune Oracle SQL
- Self-order the table joins

PERFORMANCE TUNING

NOTHING !!!!!

PERFORMANCE TUNING

- **How sensibly you use this tool.**
- **Which helps to reduce processing cost.**
- **Which helps to reduce storage cost.**

PERFORMANCE TUNING

- SELECT * FROM EMP;
- SELECT EMP_NAME, EMP_ADDR FROM EMP;
- Sharing virtual warehouse while dev activities.
- Order your data by filter columns during data loading process.
- Use multi cluster warehouse instead of spinning up existing cluster to bigger size.

PERFORMANCE TUNING

- There is no concept of index.
- No concept of primary key , foreign key constraints.
- No need of transaction management.
- There is no buffer pool.
- You will never encounter out of memory exception.

PERFORMANCE TUNING

- **But How ACID transactions are possible !!!!**

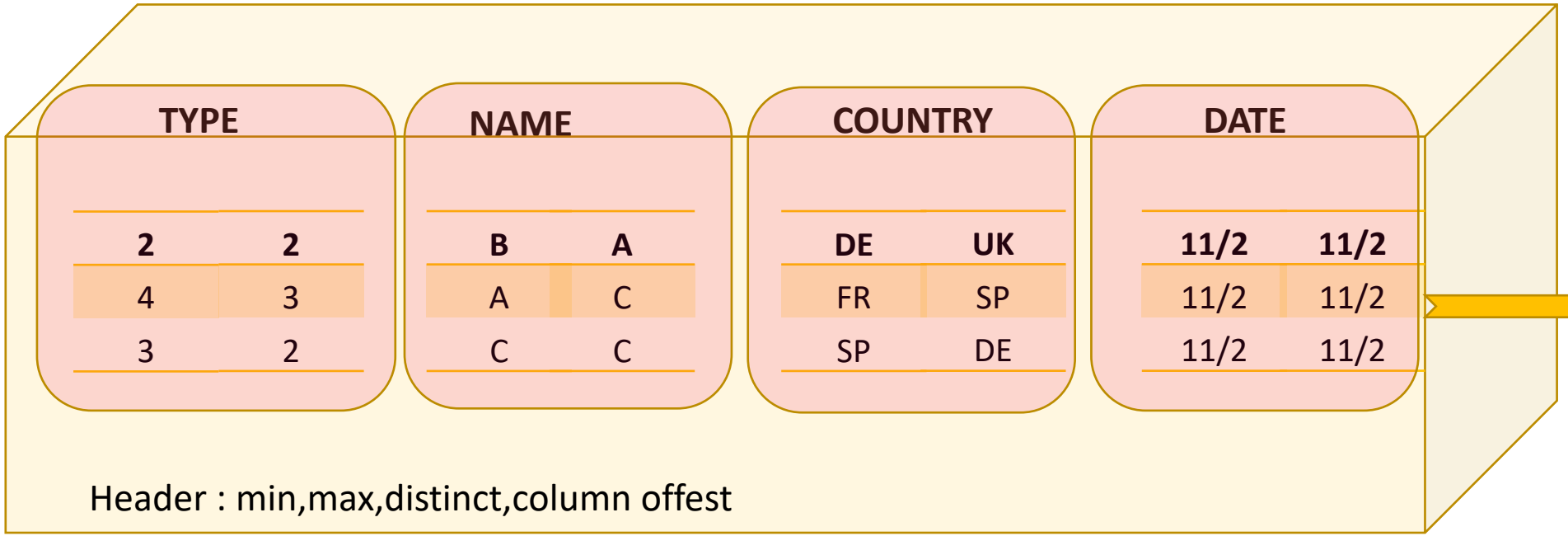
Query management and optimization

- **Parsing.**
- **Object resolution.**
- **Access control.**
- **Plan optimization.**
- **Snowflake will not use indexes.**

Query management and optimization

- Storage medium in snowflake is s3 and data format is compressed files.
- Maintaining indexes significantly increases volume of data and data loading time.
- User need to explicitly create indices. Which goes against snowflake philosophy of SAS.
- Maintaining indices can be complex , expensive and risky process.

Table File



Heavily compressed

Micro-partitioning is automatically **performed** on all Snowflake tables. Tables are transparently partitioned using the **ordering of the data as it is inserted/loaded**.

Query management and optimization

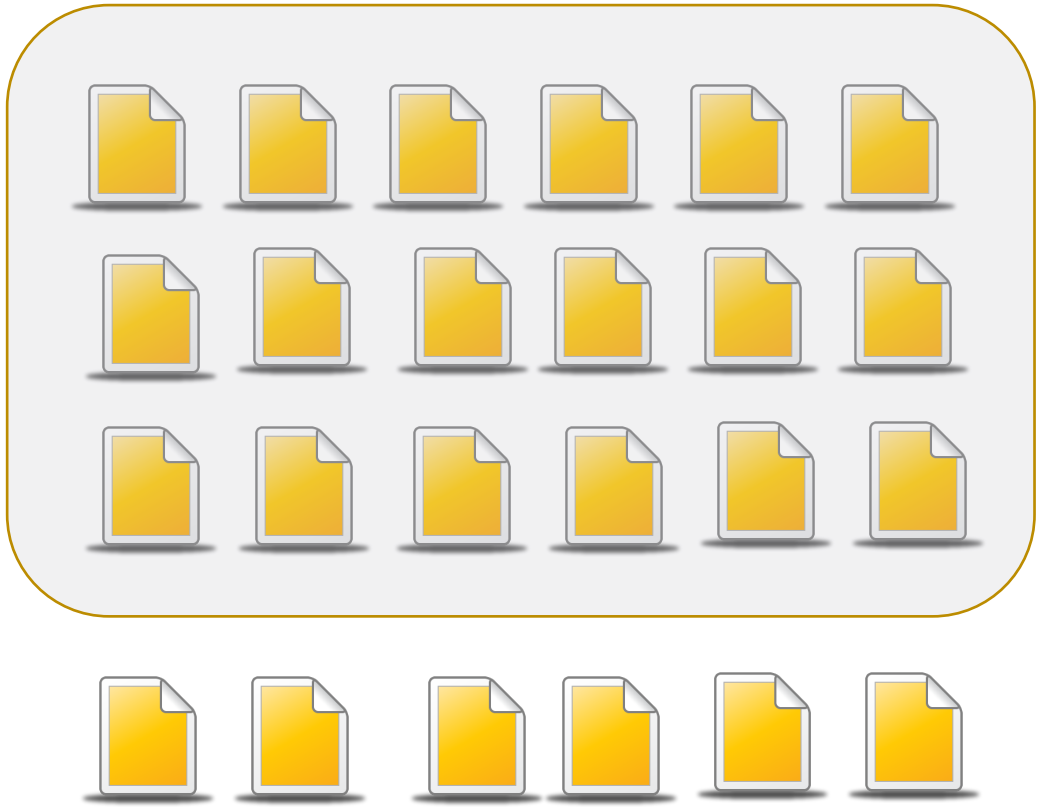
- **Pruning.**
- **Zone maps.**
- **Data skipping.**

Concurrency control

KEY	VALUE
TABLE1	S3:\\<some_location>\
TABLE1	S3:\\<some_location>\
TABLE1	S3:\\<some_location>\
TABLE1	S3:\\<some_location>\

Concurrency control

Concurrency control



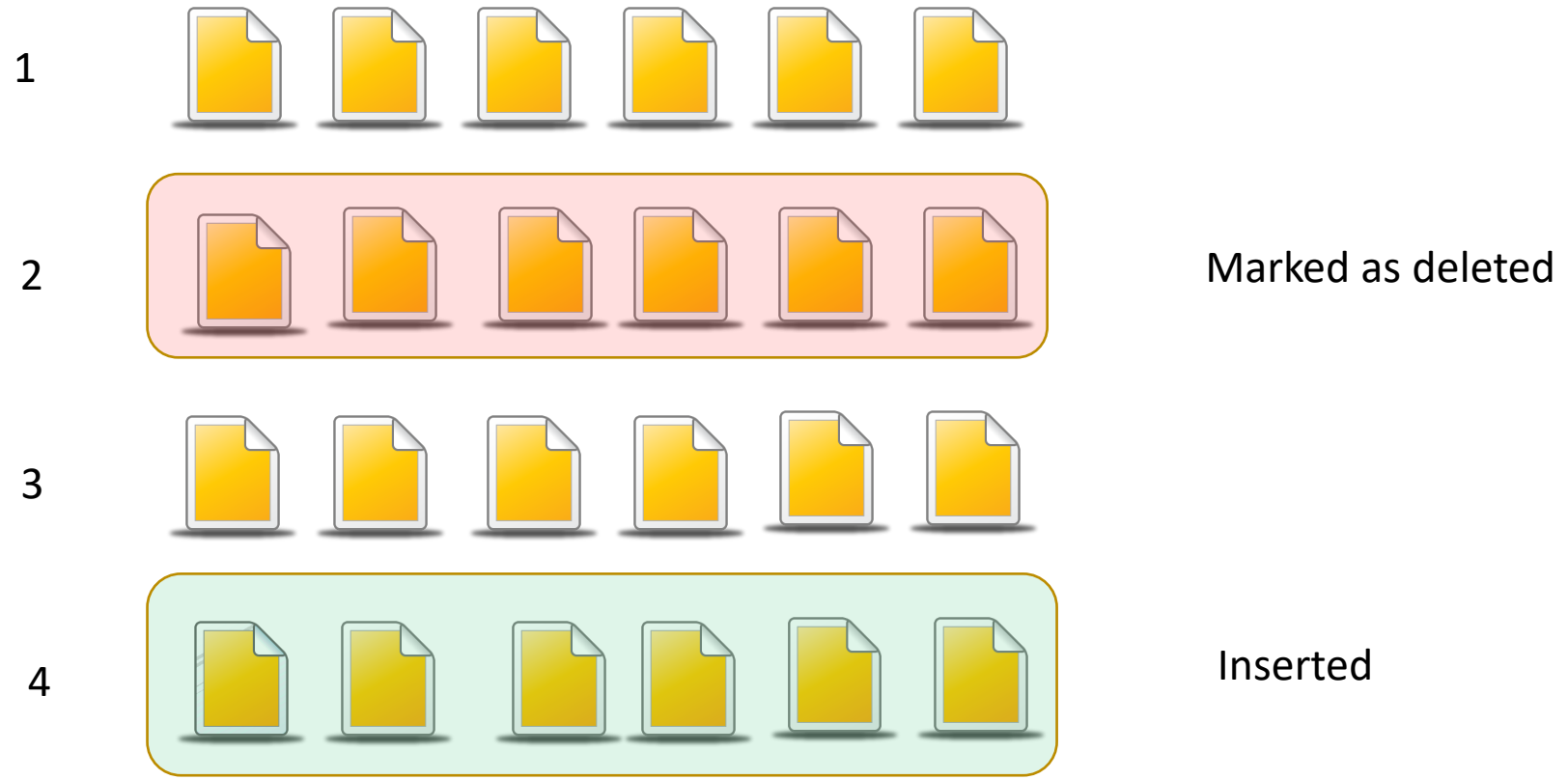
SNAPSHOT ISOLOATION

HOW UPDATES WORKS

- **DELETE**
- **INSERT**

HOW UPDATES WORKS

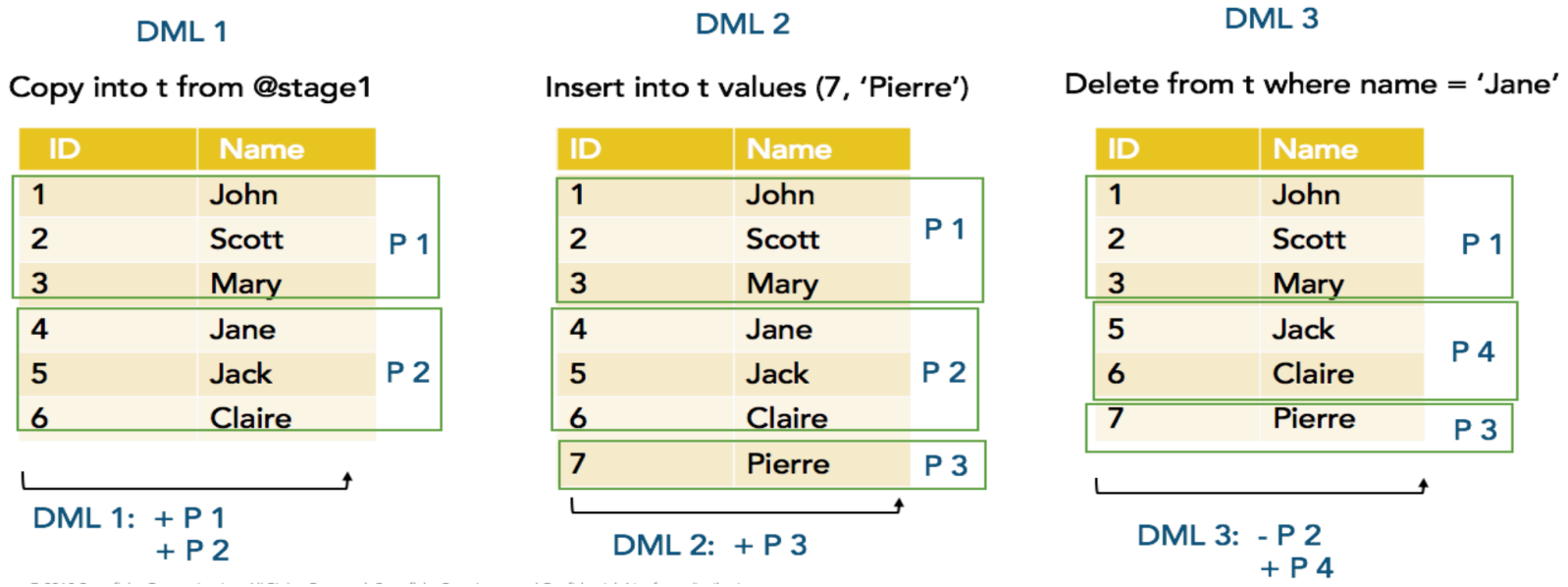
Key	Value	Active
EMP	1	Y
EMP	2	N
EMP	3	Y
EMP	4	Y



HOW UPDATES WORKS

Partitioned table and DML

- No update-in-places (AWS S3, Azure Blob are immutable file systems)
- DML **adds** or **deletes** partition files -> partition files are written in full



Why I have to remove files

- **Because we are using blob storage area like s3. Here you can only overwrite full files.**

LEASON LEARNED

- **Before executing update, statement check how many records will get impacted. If more than 80% of records are getting impacted, then instead of executing update you can consider recreating whole table.**
- **You can execute delete and insert statement separately.**
- **When you are trying to update or delete try considering numeric columns as its easy to scan micro- partitions.**