VERTICA

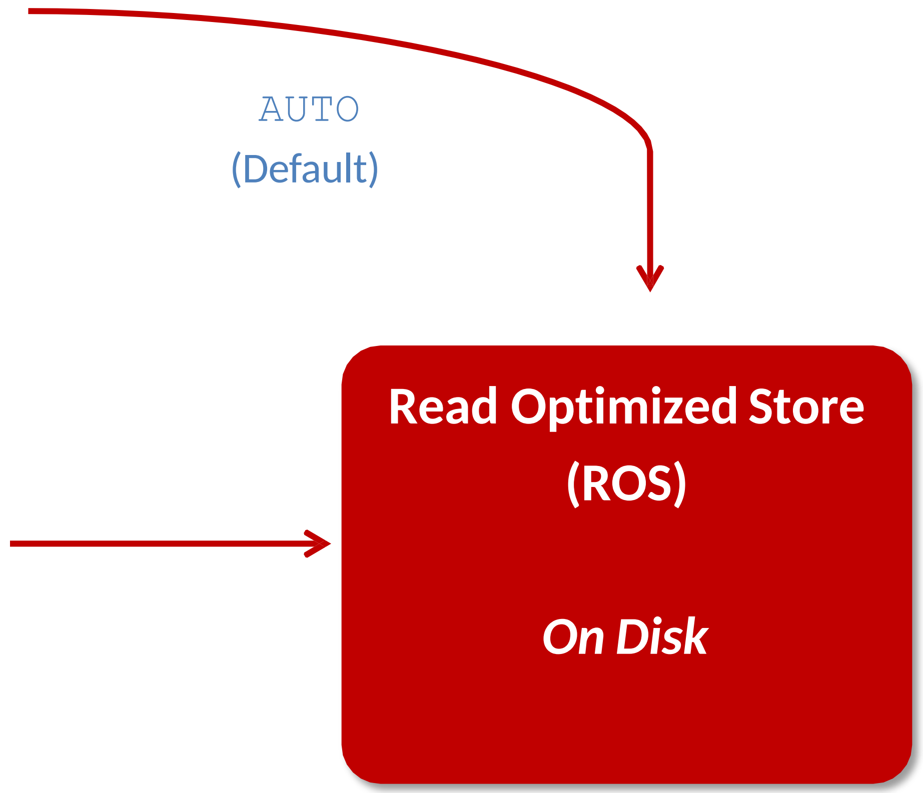
by opentext™

LOADING DATA INTO VERTICA

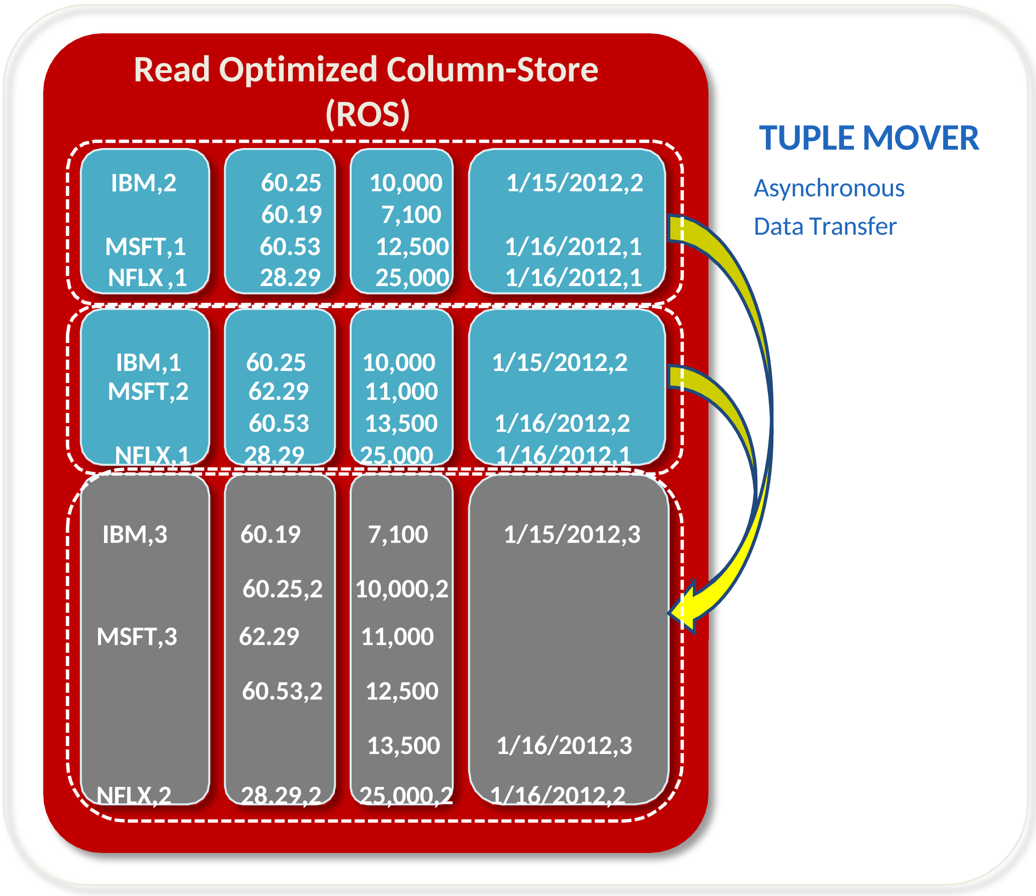
Loading to ROS: Trickle Loading and Bulk Loading

- Trickle Load
  - `INSERT`
  - `UPDATE`
  - `DELETE`
  - `COPY`

- Bulk Load
  - `INSERT /*+DIRECT*/`
  - `UPDATE /*+DIRECT*/`
  - `DELETE /*+DIRECT*/`
  - `COPY DIRECT`



Tuple Mover - mergeout



- ROS created by COPY
- Defragments storage containers by merging them
- Purges deleted records
- Honors projection definition
- Maintains partition boundary on active partitions

Tuple Mover – mergeout parameters

- ActivePartitionCount
  - default 1, increase if simultaneously loading multiple partitions

- MergeOutInterval
  - default 10 minutes, periodic check for mergeout

opentext™

Vertica : Adding Rows

Adding rows: INSERT, COPY

- INSERT
  - Use infrequently, high overhead
  - If necessary `INSERT /*+DIRECT*/ … SELECT …`
- COPY
  - Faster approach to load a warehouse
  - Can load in parallel

COPY

- Bulk loads data from files to Vertica
  - COPY FROM STDIN pipes from STDIN
    - zcat table_a.gzip | vsql -c “COPY table_a from STDIN DIRECT;”
  - COPY FROM file loads from Vertica nodes
- Usually TEXT loading, NATIVE (via drivers)
- See SQL Reference manual for command options

COPY: Checking Data Format Before Loading

-
  ```
  $ file Date_Dimension.tbl
  Date_Dimension.tbl: ASCII text
  ```

-
  ```
  $ wc Date_Dimension.tbl
  1828   5484 221822 Date_Dimension.tbl
  ```

-
  ```
  $ file data*
   data1.txt: Little-endian UTF-16 Unicode text
    data2.txt: ISO-8859 text
  ```

COPY: Converting Files Before Loading Data

- iconv -f ISO88599 -t utf-8 data2.txt > data2-utf8.txt

COPY: Checking UTF-8 Compliance After Loading Data

- SELECT name FROM nametable WHERE ISUTF8(name) = FALSE;

COPY: Loading Data Interactively

- $ cat fact_table.tbl | vsql -c "COPY FACT_TABLE FROM STDIN DELIMITER '|' DIRECT";
- $ cat fact_table.tbl | vsql -c "COPY FACT_TABLE FROM LOCAL STDIN DELIMITER '|' DIRECT";

COPY LOCAL

- Enables copying of a file stored on a local machine
  - dbadmin access NOT required
  - Exceptions and rejections can be directed to the client machine
  - No need to set up permissions on Storage Locations
- `COPY <file> from LOCAL '/home/user/data/*’`
- `=> COPY store.store_dimension FROM LOCAL '/usr/files/my_data/input_file' GZIP;`
- `=>COPY simple_table FROM LOCAL 'input_file.bz' BZIP, 'input_file.bz' BZIP;`
- `=> COPY simple_table FROM LOCAL STDIN;`

COPY NO COMMIT

- COPY automatically commits by default
  - COPY NO COMMIT to prevent this
  - Enables review of error logs before deciding to commit
  - Combine Multiple COPY Statements in Same Transaction

```
COPY... NO COMMIT;
COPY... NO COMMIT;
COPY... NO COMMIT;
COPY X FROM LOCAL NO COMMIT;
COMMIT;
```

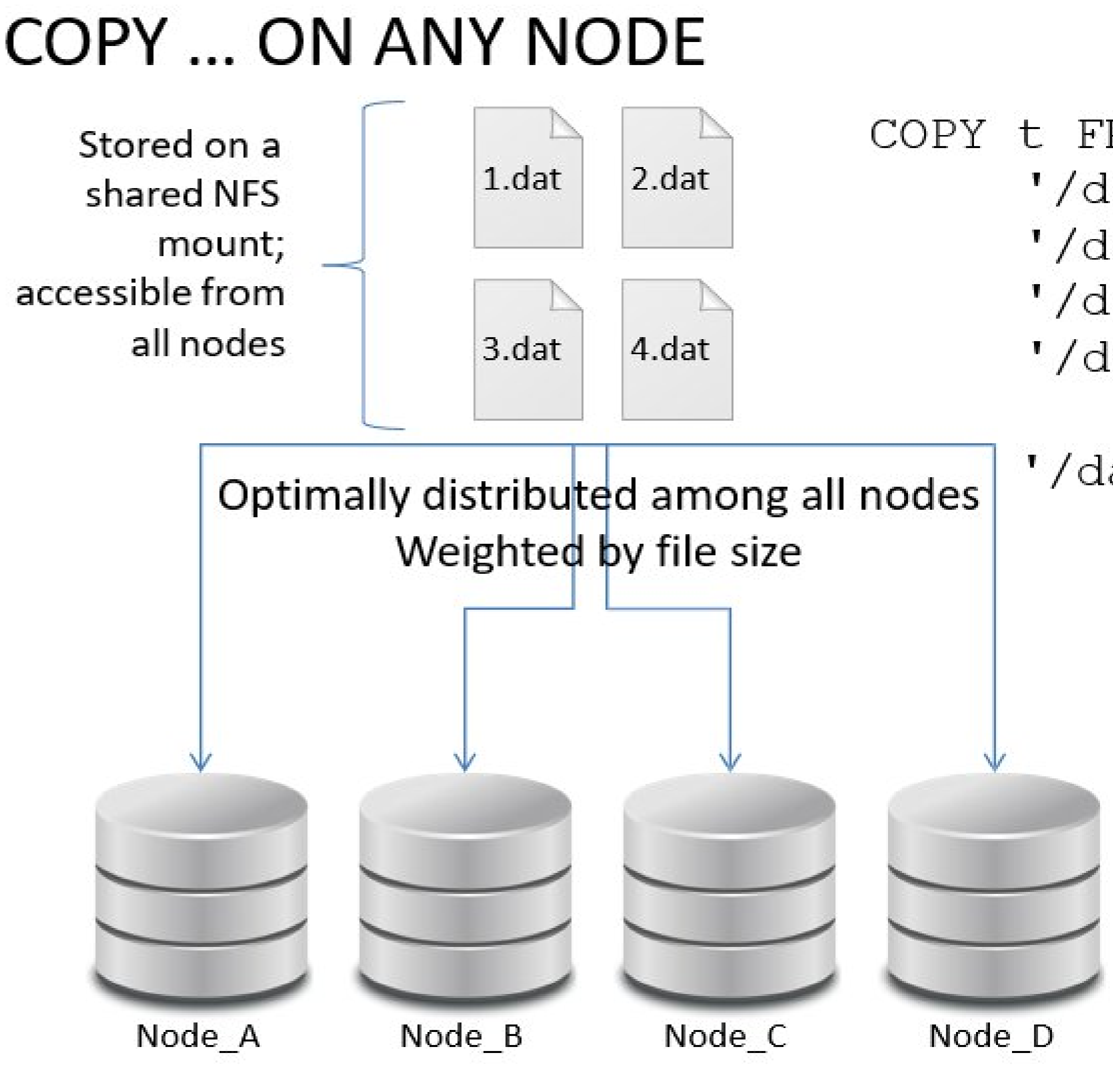
COPY Command Options

- Columns as expressions

  -
    ```
    COPY Retail.Dim (num, name, store, date as SYSDATE)
    FROM '/home/dbadmin/dim.txt' DELIMITER '|';
    ```
- Columns computed from other columns

  -
    ```
    COPY Retail.Dim (num, name, store,
    joined filler date,
    joined_mth AS DATE_PART('month',joined),  joined_year AS
    DATE_PART('year',joined))  FROM '/home/dbadmin/dim3.txt'
    DELIMITER '|';
    ```
- Expressions CAN use most SQL functions, operators, constants, NULLs, and comments
- Expressions CANNOT use Vertica analytic or aggregate functions

Multi-file COPY


COPY ... ON ANY NODE
Stored on a shared NFS mount; accessible from all nodes
1.dat
2.dat
3.dat
4.dat
Optimally distributed among all nodes
Weighted by file size
Node_A
Node_B
Node_C
Node_D

```
COPY t FROM
     '/data/1.dat' ON ANY NODE,
     '/data/2.dat' ON ANY NODE,
     '/data/3.dat' ON ANY NODE,
     '/data/4.dat' ON ANY NODE;
                 OR
     '/data/*.dat' ON ANY NODE;
```

opentext™

Multi-file COPY

Þ COPY t FROM '/data/file1.dat' ON v_vmart_node0001, '/data/file2.dat' ON v_vmart_node0002;

Þ COPY t FROM '/data/*.dat' ON ANY NODE;

Þ COPY t FROM '/data/bigfile.dat' ON ANY NODE;

Þ => COPY t FROM '/data/big1.dat' ON (v_vmart_node0001, v_vmart_node0002, v_vmart_node0003),'/data/big2.dat' ON (v_vmart_node0004, v_vmart_node0005);

opentext™

Vertica : Updating Rows

Updating Rows: Update/Delete

- Delete
  - Check performance
  - Not performed in place
  - PURGE removes files
- Update
  - More efficient to split into delete and bulk load
  - Use delete best practices.

Updating Rows: Bulk Delete

- Extract all delete into a file
- => CREATE LOCAL TEMP TABLE data_to_delete (emp_id INT);
- => COPY data_to_delete FROM '/tmp/employee_to_delete.txt' ;
- => DELETE /*+ direct */ FROM store.store_orders_fact WHERE employee_key IN (SELECT * FROM data_to_delete);
- => DROP TABLE data_to_delete ;

Updating Rows: PURGE

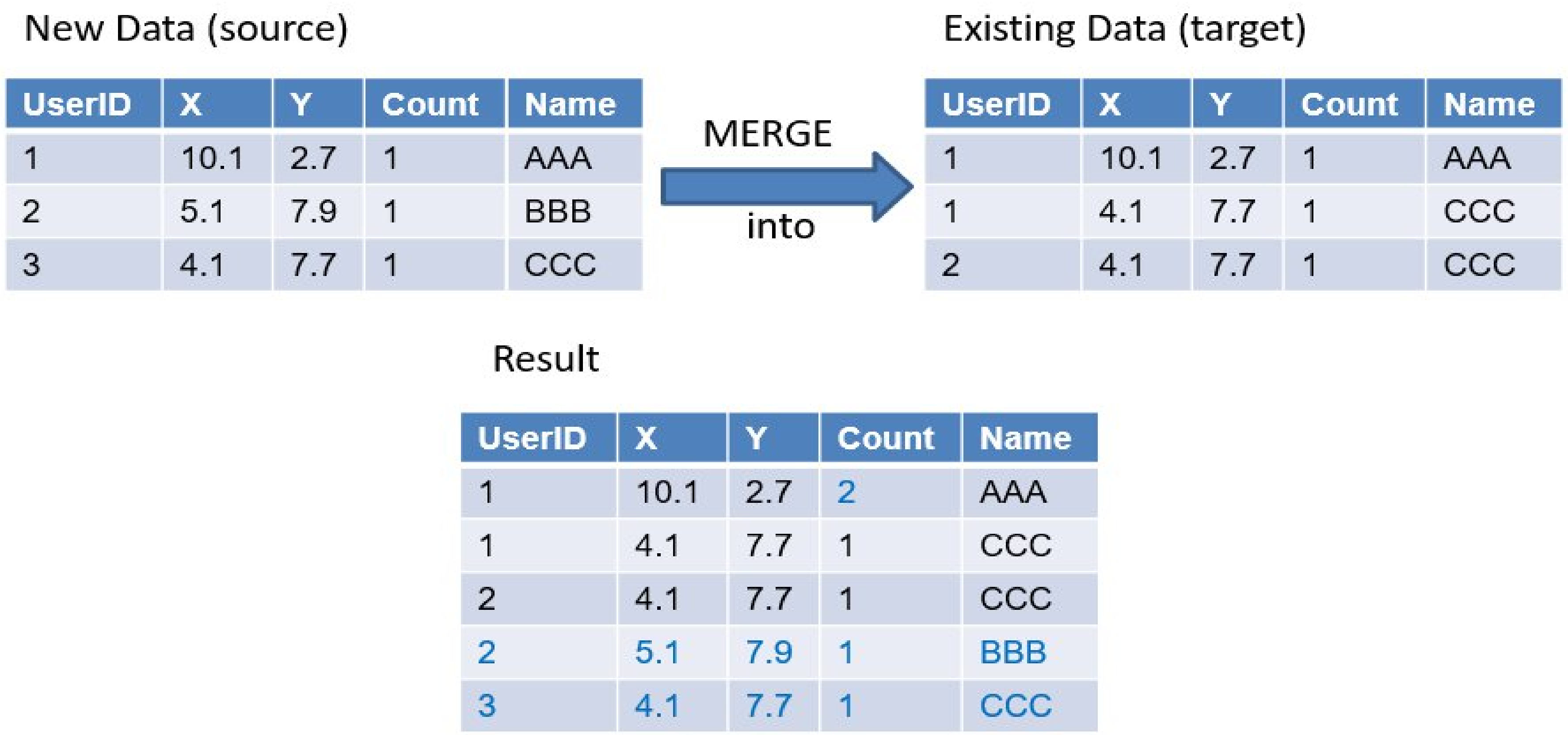
- Permanently removes delete vectors from ROS storage containers so disk space can be reused. PURGE removes all historical data up to and including the Ancient History Mark epoch.
- SELECT PURGE();
- SELECT PURGE_TABLE('store.store_sales_fact');

Bulk Update Recommendation

1. Store new batch by loading the new batch in staging table
2. Identify update records by joining batch with main table and extract keys for obsolete records
3. Store keys in separate table
4. Delete records through the support of a subquery
5. Bulk load the staging table
6. Clean up temporary tables

MERGE

- Data being loaded includes both new and existing data
- Inserts new records, updates existing records



New Data (source)

UserID	X	Y	Count	Name
1	10.1	2.7	1	AAA
2	5.1	7.9	1	BBB
3	4.1	7.7	1	CCC

Existing Data (target)

UserID	X	Y	Count	Name
1	10.1	2.7	1	AAA
1	4.1	7.7	1	CCC
2	4.1	7.7	1	CCC

Result

UserID	X	Y	Count	Name
1	10.1	2.7	2	AAA
1	4.1	7.7	1	CCC
2	4.1	7.7	1	CCC
2	5.1	7.9	1	BBB
3	4.1	7.7	1	CCC

MERGE Syntax

```
MERGE INTO location target  USING new location source
    ON source.userid = target.userid  AND source.x = target.x
        AND source.y = target.y  WHEN MATCHED THEN
    UPDATE SET count = target.count +  source.count
WHEN NOT MATCHED THEN
    INSERT VALUES (src.userid, src.x,  src.y, src.count, src.name);
```

opentext™

Vertica : Post Loading Tasks

Update Statistics

- Data statistics written to catalog for use by query Optimizer
- When to run?
  - After an initial data load
  - If data has changed > 50%
  - If query performance changes over time (stale statistics)

Update Statistics

- How to run?

```
SELECT analyze_statistics ('table-name');
```

  - Use ‘’ for all tables

- How to check statistics?

```
SELECT get_projection_status ('projection-name');
SELECT has_statistics FROM projections;
```

- Specific recommendations appear in Workload Analyzer in Management Console

opentext™

Thank you