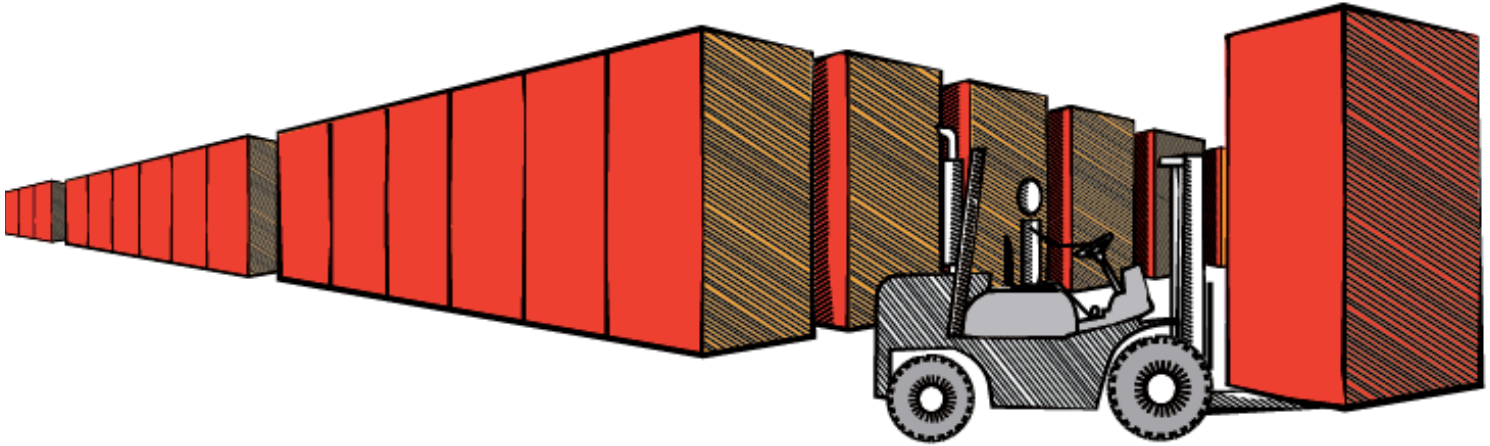


Amazon Redshift Data Loading

Petabyte-Scale Data Warehousing Service

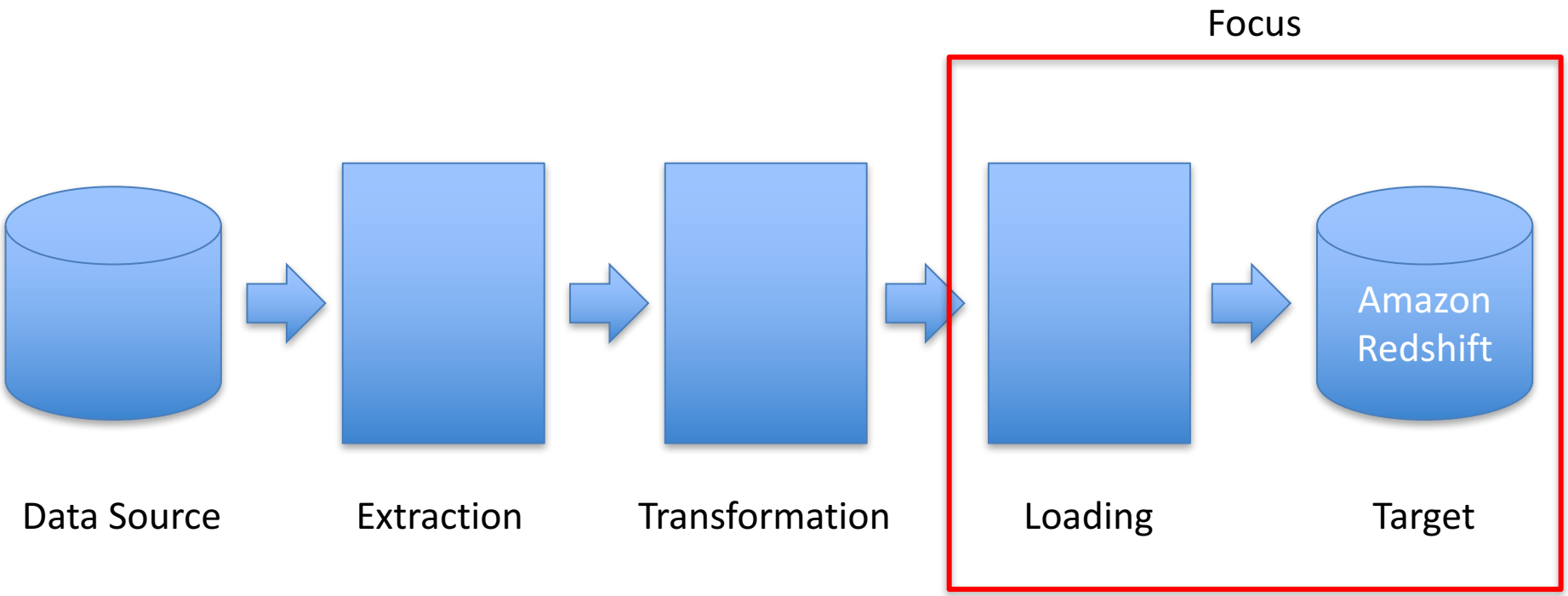
amazon web services

Agenda

- Data Loading Process in Amazon Redshift
- Loading Data Options
  - Amazon S3
    - Loading Data into S3
    - Preparing Data Files
    - Loading data into Redshift
  - AWS Database Migration Service
  - DynamoDB
  - Elastic MapReduce
  - Lambda
  - Remote Loading (SSH)
- Compression
- Default Data Values
- Analyzing Tables
- Vacuuming Tables
- Concurrent Write Operations
- Data Validation
- Troubleshooting and Best Practices for Loading Data

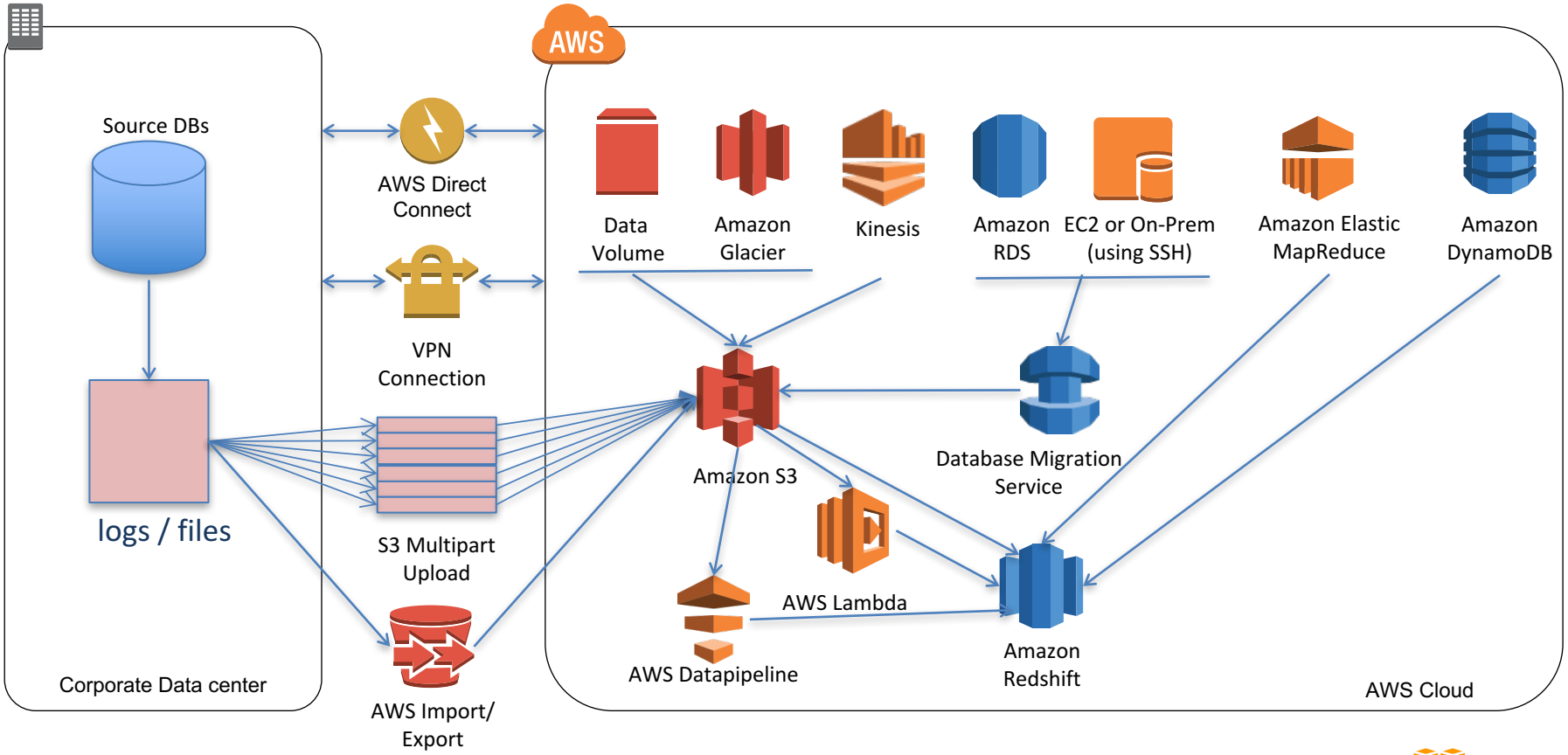
amazon web services

Data Loading Process



amazon web services

Amazon Redshift Loading Data Overview



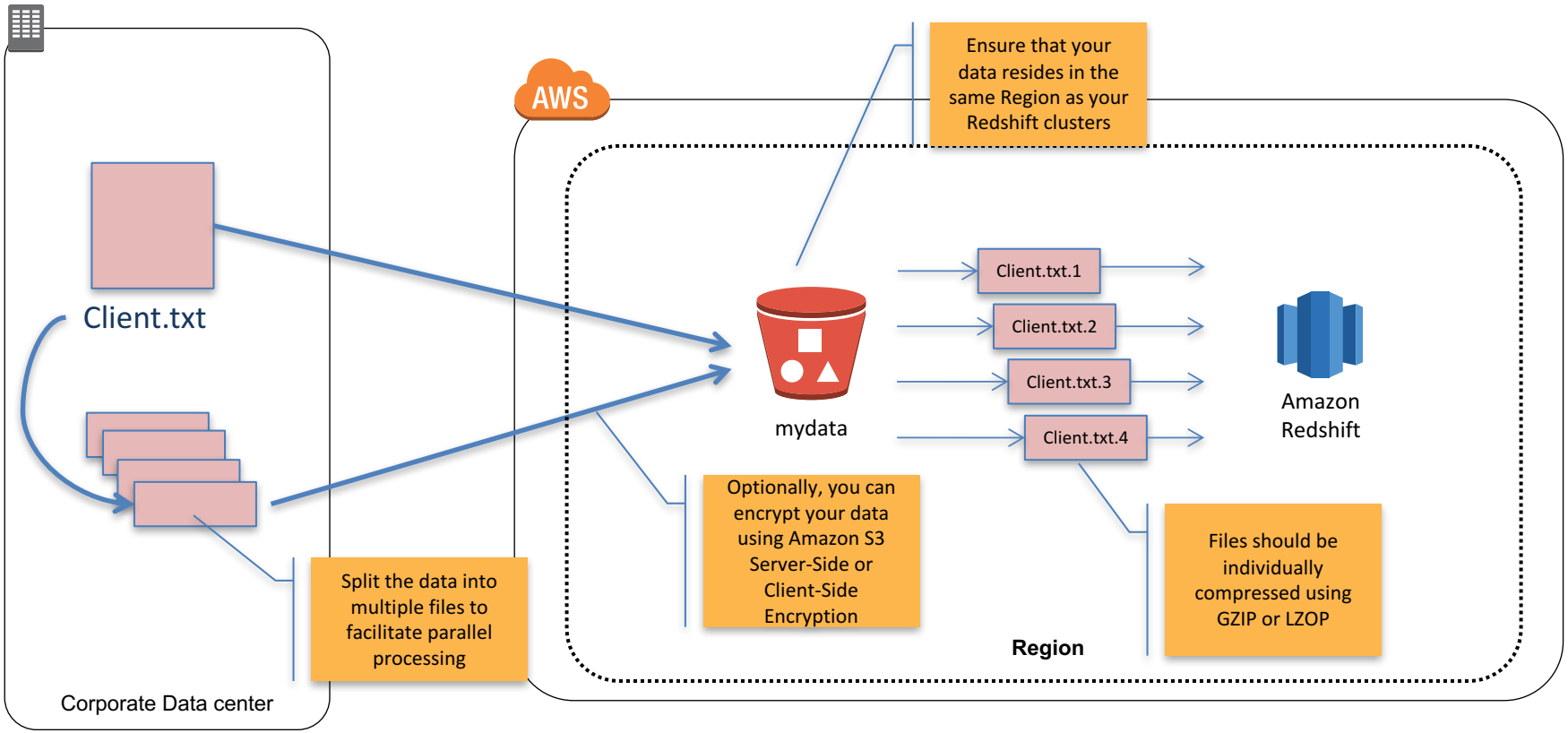
amazon web services

Loading Data Into Amazon S3

- RDS/EC2/On-premise Database - Database migration Service
- RDS MySQL using Data Pipeline http://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-template-copyrdstos3.html
- Data Volume on EBS – cp Command to copy to S3
- Glacier – Restore data back to S3 from archived backups
- Kinesis
  - Kinesis Streams – Kinesis Client Library and Kinesis Connector/ Lambda
  - Kinesis Firehose - load data into S3 bucket and then use Redshift COPY

amazon web services

Uploading Files to Amazon S3



amazon web services

Loading Data From Amazon S3

- Preparing Input Data Files
- Uploading files to Amazon S3
- Using COPY to load data from Amazon S3

Preparing Input Data Files Using Delimiters

Example of pipe ('|') delimited file

```
1|Customer#000000001|j5JsirBM9P|MOROCCO  0|MOROCCO|AFRICA|25-989-741-2988|BUILDING
2|Customer#000000002|487LW1dovn6Q4dMVym|JORDAN   1|JORDAN|MIDDLE EAST|23-768-687-3665|AUTOMOBILE
3|Customer#000000003|fkRGN8n|ARGENTINA7|ARGENTINA|AMERICA|11-719-748-3364|AUTOMOBILE
4|Customer#000000004|4u58h f|EGYPT    4|EGYPT|MIDDLE EAST|14-128-190-5944|MACHINERY
```

```
CREATE TABLE customer (
  c_custkey      integer       not null,
  c_name         varchar(25)   not null,
  c_address      varchar(25)   not null,
  c_city         varchar(10)   not null,
  c_nation       varchar(15)   not null,
  c_region       varchar(12)   not null,
  c_phone        varchar(15)   not null,
  c_mktsegment   varchar(10)   not null
);
```

amazon web services

Preparing Input Data Files using Fixed-width

```
CREATE TABLE customer (
  c_custkey     integer        not null,
  c_name        varchar(25)    not null,
  c_address     varchar(25)    not null,
  c_city        varchar(10)    not null,
  c_nation      varchar(15)    not null,
  c_region      varchar(12)    not null,
  c_phone       varchar(15)    not null,
  c_mktsegment  varchar(10)    not null
);
```

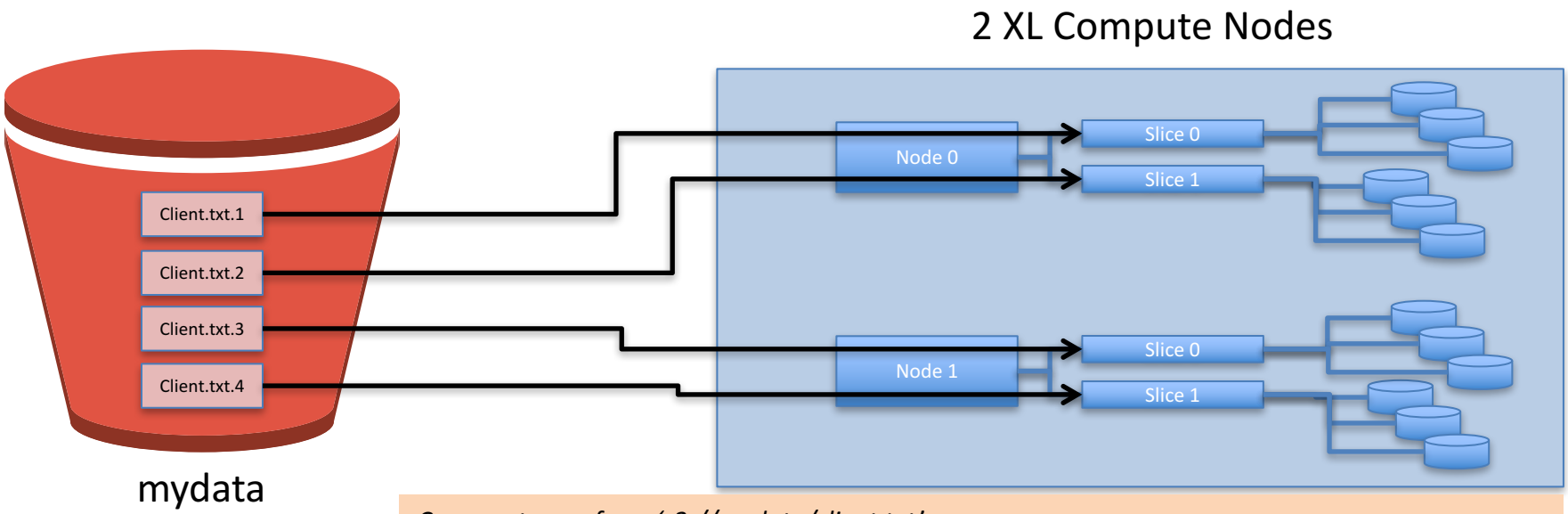
Copy customer from 's3://mydata/client.txt'
Credentials 'aws_access_key_id=<your-access-key>;
aws_secret_access_key=<your_secret_key>'
fixedwidth '0:3, 1:25, 2:25, 3:10, 4:15, 5:12, 6:15, 7:10';

1	RFK	900 Columbus	MOROCCO	MOROCCO	AFRICA	25-989-741-2988	BUILDING
2	JFK	800 Washington	JORDAN	JORDAN	MIDDLE EAST	23-768-687-3665	AUTOMOBILE
3	LBJ	700 Foxborough	ARGENTINA	ARGENTINA	AMERICA	11-719-748-3364	AUTOMOBILE
4	GWB	600 Kansas	EGYPT	EGYPT	MIDDLE EAST	14-128-190-5944	MACHINERY

Client.txt

amazon web services

Splitting Data Files

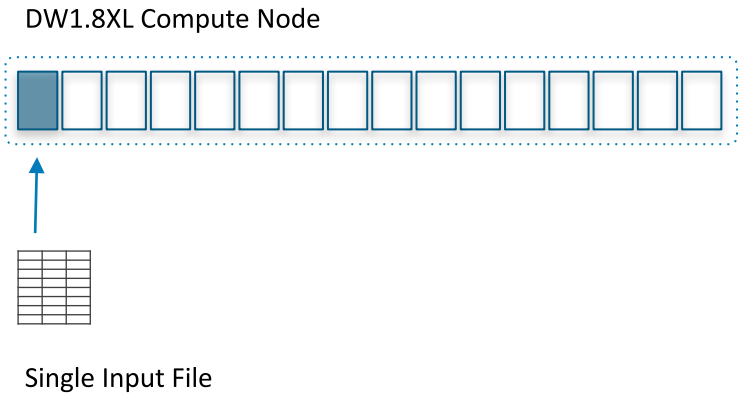


Copy customer from 's3://mydata/client.txt'
Credentials 'aws_access_key_id=<your-access-key>; aws_secret_access_key=<your_secret_key>'
Delimiter '|';

amazon web services

Loading – Use multiple input files to maximize throughput

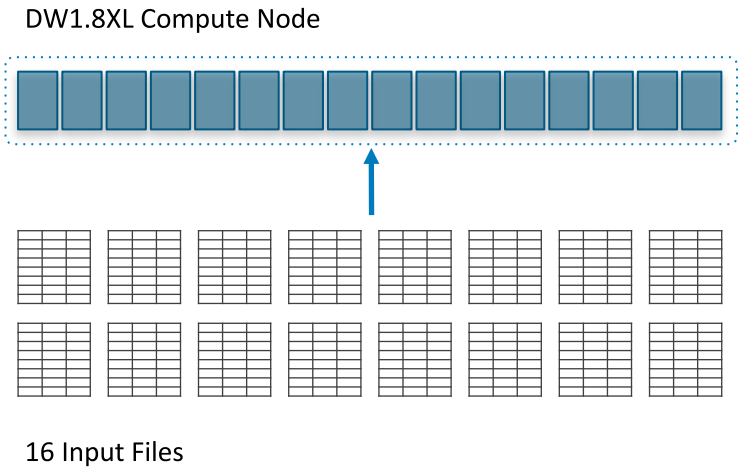
- Use the COPY command
- Each slice can load one file at a time
- A single input file means only one slice is ingesting data
- Instead of 100MB/s, you're only getting 6.25MB/s



amazon web services

Loading – Use multiple input files to maximize throughput

- Use the COPY command
- You need at least as many input files as you have slices
- With 16 input files, all slices are working so you maximize throughput
- Get 100MB/s per node; scale linearly as you add nodes



amazon web services

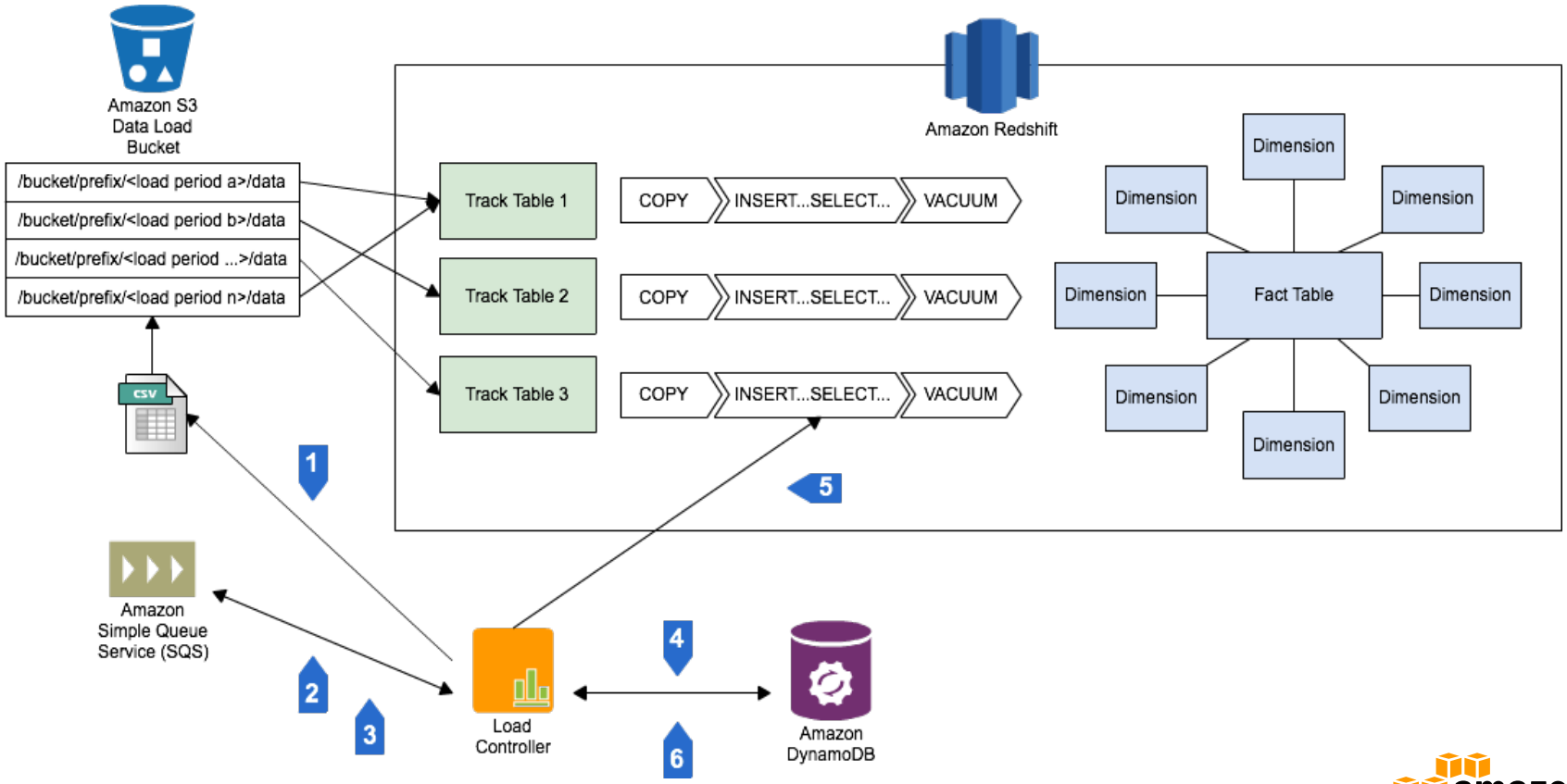
Loading Data with Manifest Files

- Use manifest to loads all required files
- Supply JSON-formatted text file that lists the files to be loaded
- Can load files from different buckets or wit different prefix

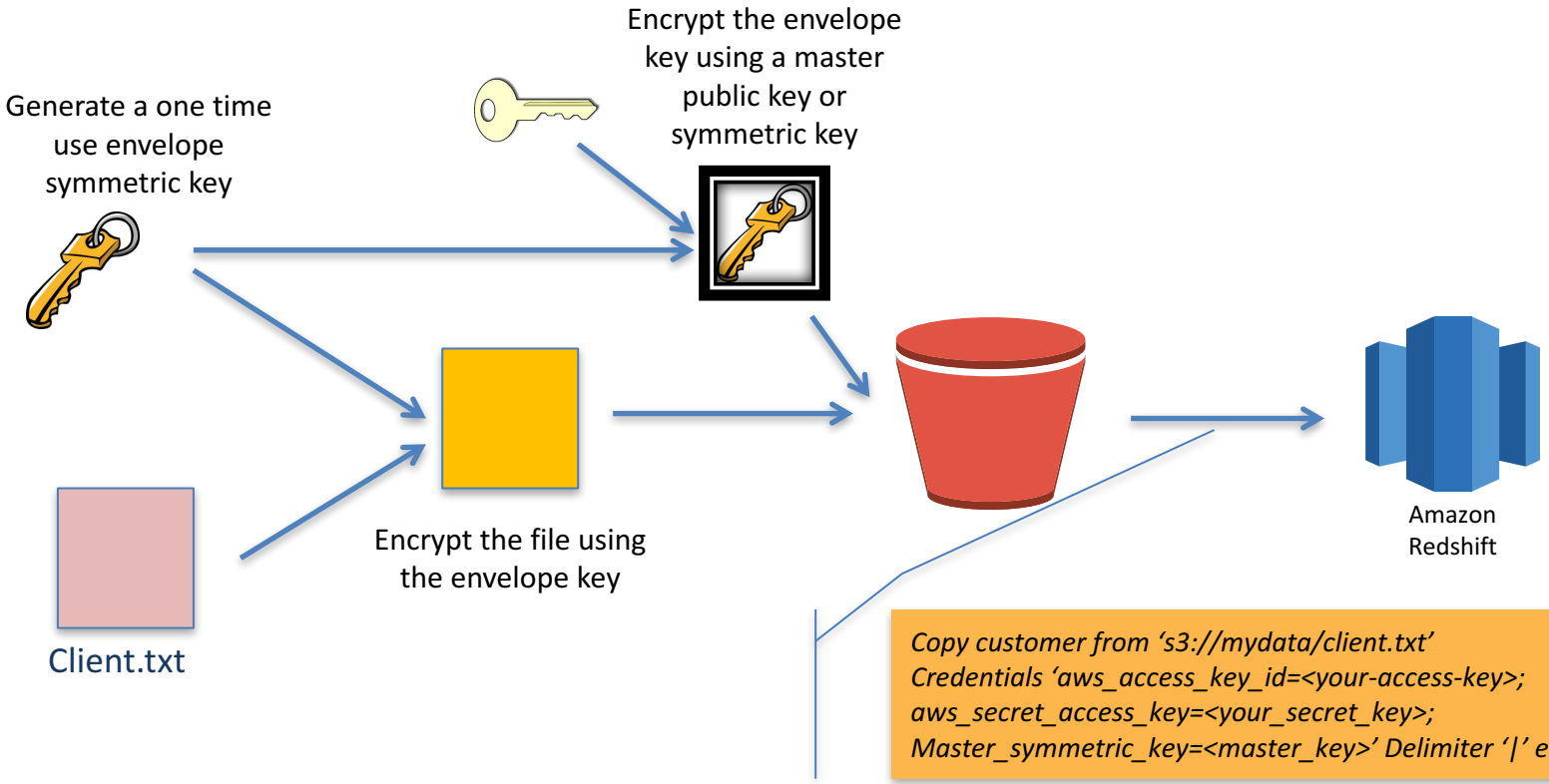
```
{
 "entries": [
    {"url":"s3://mybucket-alpha/2013-10-04-custdata", "mandatory":true},
    {"url":"s3://mybucket-alpha/2013-10-05-custdata", "mandatory":true},
    {"url":"s3://mybucket-beta/2013-10-04-custdata", "mandatory":true},
    {"url":"s3://mybucket-beta/2013-10-05-custdata", "mandatory":true}
 ]
}
```

amazon web services

Micro-Batch Loading



Loading Encrypted Data Files



Copy customer from 's3://mydata/client.txt'
Credentials 'aws_access_key_id=<your-access-key>;
aws_secret_access_key=<your_secret_key>;
Master_symmetric_key=<master_key>' Delimiter '|' encrypted;

amazon web services

Redshift COPY Command

- Loads data into a table from data files in S3 or from an Amazon DynamoDB table.
- The COPY command requires only three parameters:
  - Table name
  - Data Source
  - Credentials

    Copy *table_name* FROM *data_source* CREDENTIALS '*aws_access_credentials*'
- Optional Parameters include:
  - Column mapping options – mapping source to target
  - Data Format Parameters – FORMAT, CSV, DELIMITER, FIXEDWIDTH, AVRO, JSON, BZIP2, GZIP, LZOP
  - Data Conversion Parameters – Data type conversion between source and target
  - Data Load Operations –troubleshoot load times or reduce load times with parameters like COMROWS, COMPUPDATE, MAXERROR, NOLOAD, STATUPDATE

amazon web services

Loading JSON Data

- COPY uses a *jsonpaths* text file to parse JSON data
- JSONPath expressions specify the path to JSON name elements
- Each JSONPath expression corresponds to a column in the Amazon Redshift target table

Suppose you want to load the VENUE table with the following content

```
{ "id": 15, "name": "Gillette Stadium", "location": [ "Foxborough", "MA" ],
"seats": 68756 } { "id": 15, "name": "McAfee Coliseum", "location": [
"Oakland", "MA" ], "seats": 63026 }
```

You would use the following jsonpaths file to parse the JSON data.

```
{ "jsonpaths": [ "$['id']", "$['name']", "$['location'][0]",
"$['location'][1]", "$['seats']" ] }
```

amazon web services

Loading Data in Avro Format

- Avro is a data serialization protocol. An Avro source file includes a schema that defines the structure of the data. The Avro schema type must be record.
- COPY uses a *avro_option* to parse Avro data. Valid values for avro_option are as follows:
  - 'auto' (default) - COPY automatically maps the data elements in the Avro source data to the columns in the target table by matching field names in the Avro schema to column names in the target table.
  - 's3://jsonpaths_file' - To explicitly map Avro data elements to columns, you can use an JSONPaths file.

```
Avro Schema
{
    "name": "person",
    "type": "record",
    "fields": [
        {"name": "id", "type": "int"},
        {"name": "guid", "type": "string"},
        {"name": "name", "type": "string"},
        {"name": "address", "type": "string"}
}
```

amazon web services

Supported Data Types

Data Type	Aliases	Description
SMALLINT	INT2	Signed two-byte integer
INTEGER	INT, INT4	Signed four-byte integer
BIGINT	INT8	Signed eight-byte integer
DECIMAL	NUMERIC	Exact numeric of selectable precision
REAL	FLOAT4	Single precision floating-point number
DOUBLE PRECISION	FLOAT8, FLOAT	Double precision floating-point number
BOOLEAN	BOOL	Logical Boolean (true/false)
CHAR	CHARACTER, NCHAR, BPCHAR	Fixed-length character string
VARCHAR	CHARACTER VARYING, NVARCHAR, TEXT	Variable-length character string with a user-defined limit
DATE		Calendar date (year, month, day)
TIMESTAMP	TIMESTAMP WITHOUT TIME ZONE	Date and time (without time zone)

The VARCHAR data type supports multi-byte characters up to a maximum of four bytes.

amazon web services

Other Considerations

- Do not include any special characters or syntax to indicate the last field in a record
- For NULL terminators, load these characters as NULLs into CHAR or VARCHAR by using NULL AS option
- Floating-point strings – use standard or exponential format
- Timestamp – must match DATEFORMAT (YYYY-MM-DD) or TIMEFORMAT(YYYY-MM-DD hh:mm:ss) strings
- CHAR and VARCHAR strings must not be longer than the length of the corresponding columns. VARCHAR strings are measured in bytes
- Use the ESCAPE option with the COPY command if your strings contain special characters (e.g., delimiters and embedded newlines)
- Specify 'auto' to automatically convert date/time format
- CSV import option instead of comma DELIMITER

amazon web services

AWS Database Migration Service (AWS DMS)

- AWS Database Migration Service supports both homogenous and heterogeneous data replication.
- Supported database sources include: (1) Oracle, (2) SQL Server, (3) MySQL, (4) Amazon Aurora, (5) PostgreSQL, and (6) ODBC. All sources are supported on-premises, in EC2, and RDS.
- Supported database targets include: (1) Amazon Aurora, (2) Oracle, (3) SQL Server, (4) MySQL, (5) PostgreSQL, and (6) **Amazon Redshift**. All Oracle, SQL Server, MySQL and Postgres targets are supported on-premises, in EC2 and RDS.
- Keep your apps running during the migration

amazon web services

Redshift endpoint for AWS DMS

The Amazon Redshift endpoint provides full automation for:

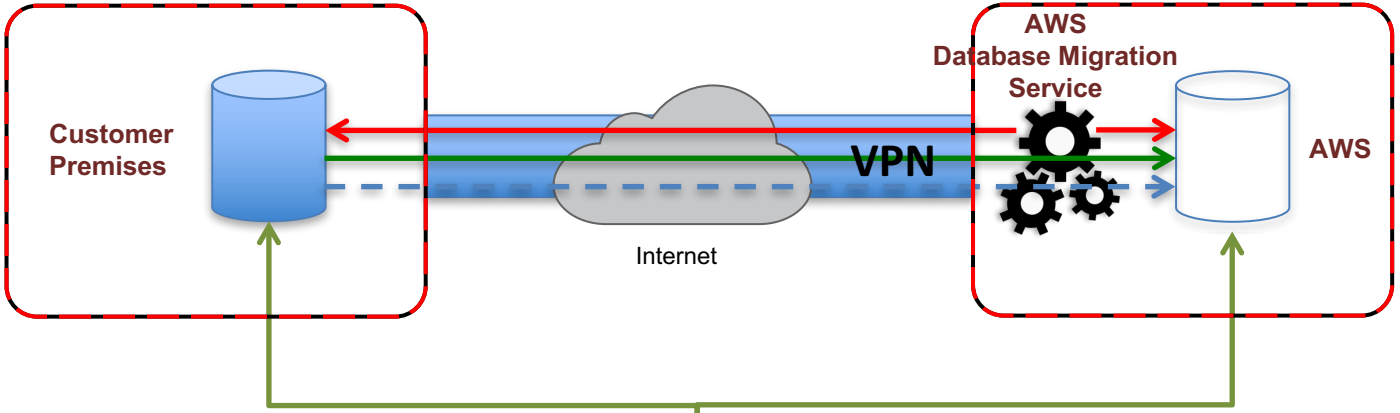
- Schema generation and data type mapping
- Full load of source database tables
- Incremental load of changes made to source tables
- Application of schema changes (DDL) made to the source tables.
- Synchronization between full load and CDC processes.

AWS DMS - Steps for Migration

- Reads the data from the source database and creates a series of CSV files on Amazon S3.
- Supports both full load and change processing operations.
  - Full-load operations:
    - Creates files for each table.
    - The table files are copied to a separate folder in Amazon S3 for each table.
    - When the files are uploaded to Amazon S3, a copy command is sent and the data in the files are copied into Amazon Redshift.
  - Change-processing operations:
    - Net changes are copied to the CSV files.
    - The net change files are uploaded to Amazon S3, then the correct data is copied to Amazon Redshift.

amazon web services

AWS DMS - Keep your apps running during the migration



- Start a replication instance
- Connect to source and target databases
- Select tables, schemas, or databases

Application Users

Let AWS Database Migration Service create tables, load data, and keep them in sync

Switch applications over to the target at your convenience

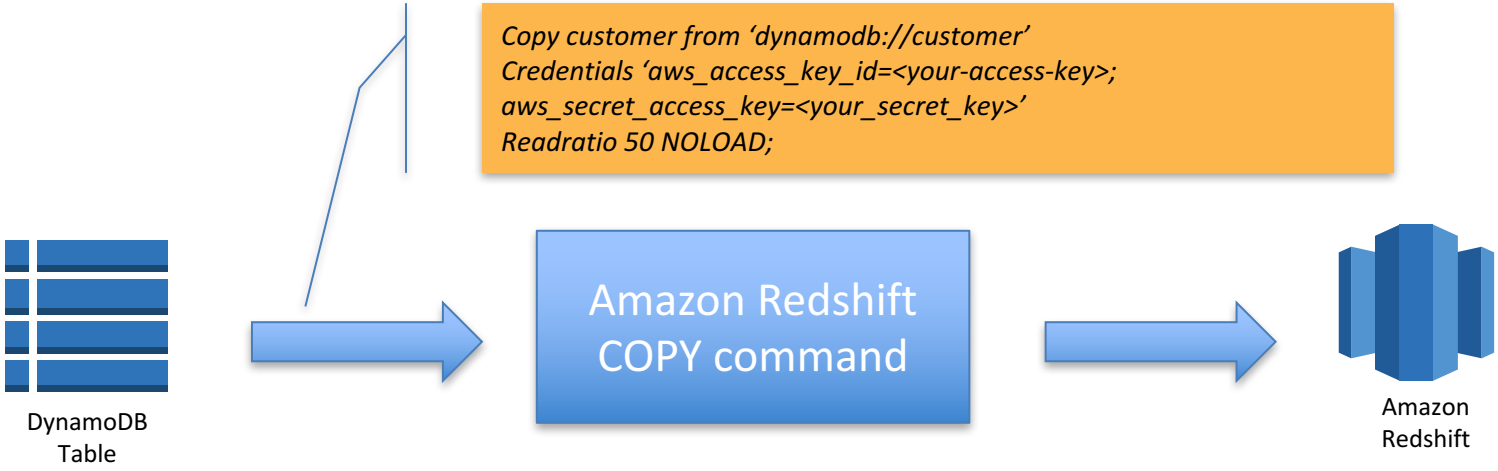
amazon web services

Loading Data from an Amazon DynamoDB Table

Differences	Amazon DynamoDB	Amazon Redshift
Table Names	• Up to 255 characters • May contain '.' (dot) and '-' (dash) characters • Case-sensitive	• Limited to 127 characters • Can't contain dots or dashes • Are NOT case-sensitive • Can't conflict with any Amazon Redshift reserved words
NULL	Does not support the SQL concept of NULL	Must specify how Amazon Redshift interprets empty or blank attribute values in Amazon DynamoDB
Data Types	STRING and NUMBER Data Types	Supported
	BINARY and SET Data Types	Not Supported

amazon web services

Provisioned Throughput with Automatic Compression and NOLOAD Option



By default, the COPY command applies compression whenever you specify an empty target table with no compression encoding. The automatic compression analysis initially samples a large number of rows from the Amazon DynamoDB table. The sample size is based on the value of the COMPROWS parameter. The default is 100K rows per slice.

amazon web services

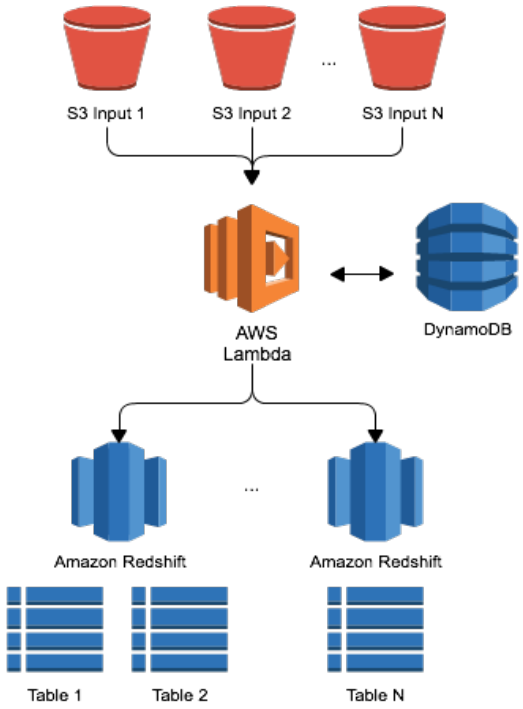
Loading Data from Amazon Elastic MapReduce

- Load data from Amazon EMR in parallel using COPY
- Specify Amazon EMR cluster ID and HDFS file path/name
- Amazon EMR must be running until COPY completes.

```
copy sales from 'emr:// j-1H7OUO3B52HI5/myoutput/part*'
credentials 'aws_access_key_id=<access-key id>;
aws_secret_access_key=<secret-access-key>';
```

amazon web services

Loading Data using Lambda



- AWS Lambda-based Amazon Redshift loader to offer you the ability to drop files into S3 and load them into any number of database tables in multiple Amazon Redshift clusters automatically, with no servers to maintain.
- Blog post https://blogs.aws.amazon.com/bigdata/post/Tx24VJ6XF1JVJAA/A-Zero-Administration-Amazon-Redshift-Database-Loader
- GitHub http://github.com/awslabs/aws-lambda-redshift-loader

amazon web services

Remote Loading using SSH

- Redshift COPY command can reach out to remote locations (EC2 and on premise) to load data using a secure shell script (SSH)

- To Remote Load follow this process:
  - Add cluster's public key to the remote host's authorized keys file
  - Configure remote host to accept connections from all cluster IP addresses
  - Create manifest file in JSON format and upload to S3 bucket
  - Issue a COPY command, including a reference to the manifest file

Copy from SSH Settings

Cluster Public Key:

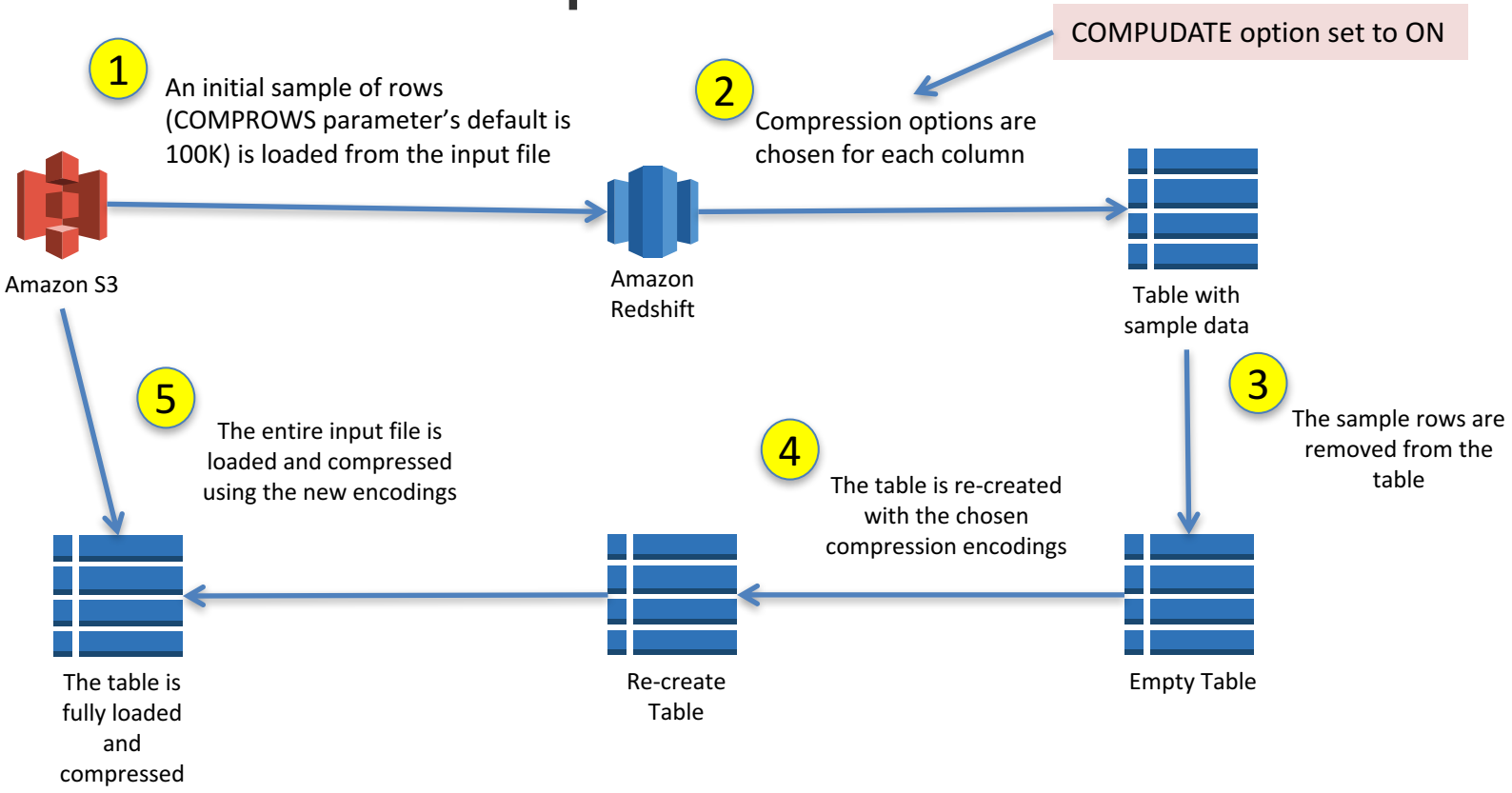
```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCTP3isxgGzVWoI
[illegible]
Vb21WRCX/v9JzDZZHNuA70BnMHCaMiRdmvsDOedZ
Amazon-Redshift
```

Node IP Addresses:

Node	Public IP	Private IP
Leader	54[illegible].207	10.[illegible].245
Compute-0	54.[illegible].198	10.[illegible]247
Compute-1	54.[illegible].182	10.1[illegible].173

amazon web services

Automatic Data Compression



amazon web services

Loading Default Data Values

```
CREATE TABLE customer (
  c_custkey      integer        not null,
  c_name         varchar(25)    not null,
  c_address      varchar(25)    not null,
  c_city         varchar(10)    not null,
  c_nation       varchar(15)    not null,
  c_region       varchar(12)    not null,
  c_phone        varchar(15)    not null,
  c_mktsegment   varchar(10)    not null default 'AWS',
);
```

- A DEFAULT expression containing RANDOM() will not include random data.
- DEFAULT expressions that contain CURRENT_DATE or SYSDATE are set to the timestamp of the current transaction.

1	RFK	900 Columbus	MOROCCO	MOROCCO	AFRICA	25-989-741-2988	AWS
2	JFK	800 Washington	JORDAN	JORDAN	MIDDLE EAST	23-768-687-3665	AUTOMOBILE
3	LBJ	700 Foxborough	ARGENTINA	ARGENTINA	AMERICA	11-719-748-3364	AUTOMOBILE
4	GWB	600 Kansas	EGYPT	EGYPT	MIDDLE EAST	14-128-190-5944	AWS

amazon web services

Updating Tables with DML Commands

- Redshift supports standard DML commands – INSERT, UPDATE, DELETE
- Redshift does not support single-command merge (upsert) statement
  - Load data into a staging table
  - Joining the staging table with the target table
    - UPDATE data where row exists
    - INSERT where no row exists

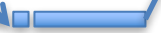
amazon web services

Analyzing Tables

ANALYZE command

The entire current database

A single Table



One or more specific columns in a single table

The ANALYZE command obtains a sample of rows, does some calculations, and saves resulting column statistics.

Statistics

You do not need to analyze all columns in all tables regularly

To maintain current statistics for tables:

- Run the ANALYZE command before running queries
- Run the ANALYZE command against the database routinely at the end of every regular load or update cycle
- Run the ANALYZE command against an new tables

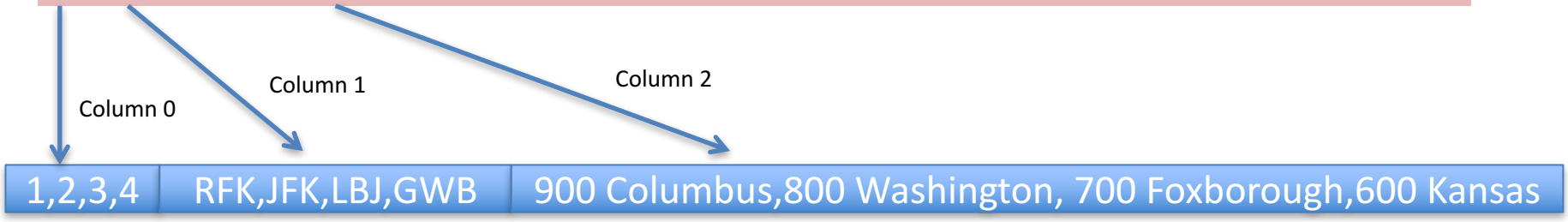
Analyze the columns that are frequently used in the following:

- Sorting and grouping operations
- Joins
- Query Predicates

amazon web services

Vacuuming Tables

1	RFK	900 Columbus	MOROCCO	MOROCCO	AFRICA	25-989-741-2988	BUILDING
2	JFK	800 Washington	JORDAN	JORDAN	MIDDLE EAST	23-768-687-3665	AUTOMOBILE
3	LBJ	700 Foxborough	ARGENTINA	ARGENTINA	AMERICA	11-719-748-3364	AUTOMOBILE
4	GWB	600 Kansas	EGYPT	EGYPT	MIDDLE EAST	14-128-190-5944	MACHINERY

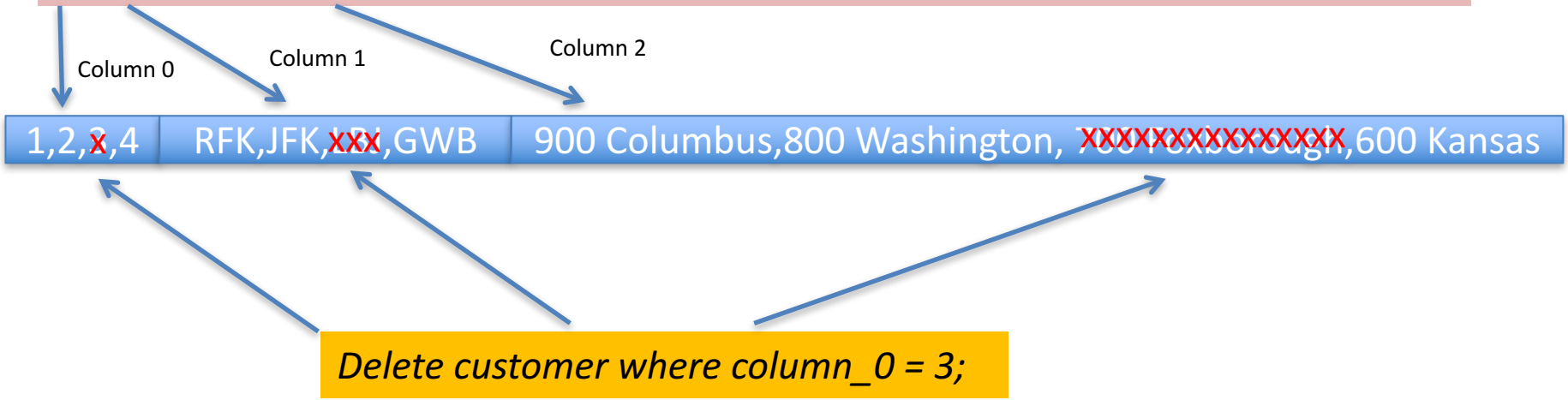


Amazon Redshift serializes all of the values of a column together.

amazon web services

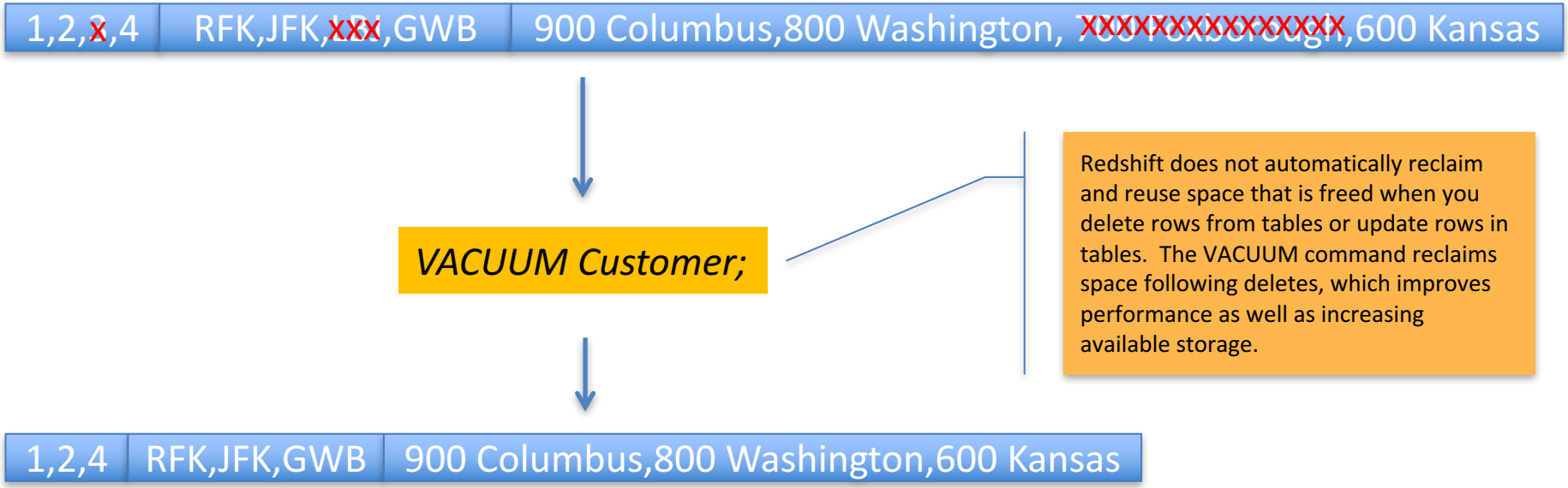
Vacuuming Tables

1	RFK	900 Columbus	MOROCCO	MOROCCO	AFRICA	25-989-741-2988	BUILDING
2	JFK	800 Washington	JORDAN	JORDAN	MIDDLE EAST	23-768-687-3665	AUTOMOBILE
3	LBJ	700 Foxborough	ARGENTINA	ARGENTINA	AMERICA	11-719-748-3364	AUTOMOBILE
4	GWB	600 Kansas	EGYPT	EGYPT	MIDDLE EAST	14-128-190-5944	MACHINERY



Delete customer where column_0 = 3;

Vacuuming Tables



Redshift does not automatically reclaim and reuse space that is freed when you delete rows from tables or update rows in tables. The VACUUM command reclaims space following deletes, which improves performance as well as increasing available storage.

amazon web services

Vacuum – Deep Copy

- Is an alternate to VACUUM.
- Will remove deleted rows and also re-sort the table
- Is more efficient than VACUUM
- You can't make concurrent updates to the table

- Deep copy options:
  - Use original table DDL and run INSERT INTO…SELECT
    - Best option - Retains all table attributes
  - Use CREATE TABLE AS
    - New table does not inherit encoding, distkey, sortkey, primary keys, or foreign keys.
  - Use CREATE TABLE LIKE
    - New table inherits all attributes except primary and foreign keys
  - Use a TEMP table to COPY data out and back in again

amazon web services

ALTER TABLE APPEND

- Appends rows to a target table by moving data from an existing source table.
- Much faster than a similar CREATE TABLE AS or INSERT INTO operation.
- Data is moved, not duplicated.
- Cannot append an identity column.
- **Syntax:** ALTER TABLE *target_table_name* APPEND FROM *source_table_name* [IGNOREEXTRA | FILLTARGET]

amazon web services

Managing Concurrent Write Operations

Concurrent COPY/INSERT/DELETE/UPDATE Operations into the same table

Copy customer from 's3://mydata/client.txt'...;

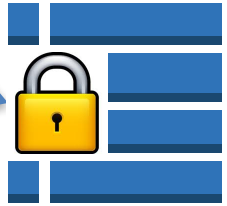
Session A

Transaction 1

Copy customer from 's3://mydata/client.txt'...;

Session B

Transaction 2



CUSTOMER

Transaction 1 puts on the write lock on the CUSTOMER table

Transaction 2 waits until transaction 1 releases the write lock

amazon web services

Managing Concurrent Write Operations

Concurrent COPY/INSERT/DELETE/UPDATE Operations into the same table

Begin;
Delete one row from CUSTOMERS;
Copy…;
Select count(*) from CUSTOMERS;
End;

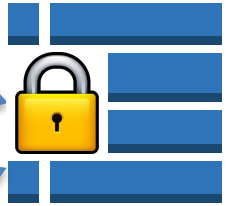
Session A

Begin;
Delete one row from CUSTOMERS;
Copy…;
Select count(*) from CUSTOMERS;
End;

Session B

Transaction 1

Transaction 2



CUSTOMER

Transaction 1 puts on the write lock on the CUSTOMER table

Transaction 2 waits until transaction 1 releases the write lock

amazon web services

Managing Current Write Operations

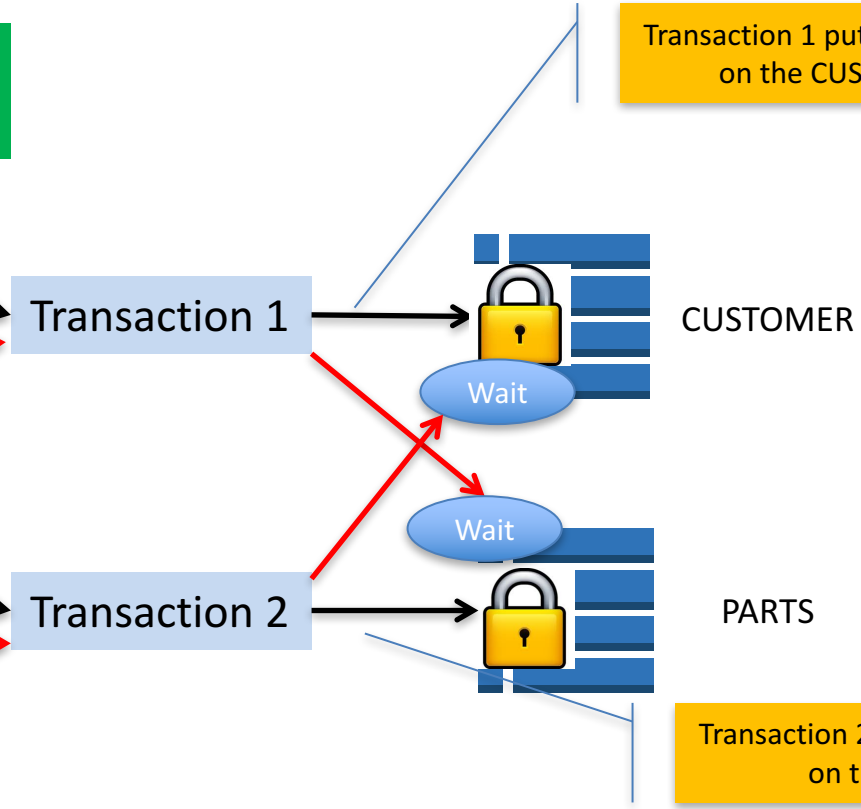
Potential deadlock situation for concurrent write transactions

Begin;
Delete 3000 rows from CUSTOMERS;
Copy…;
Delete 5000 rows from PARTS;
End;

Session A

Begin;
Delete 3000 rows from PARTS;
Copy…;
Delete 5000 rows from CUSTOMERS;
End;

Session B



Transaction 1 puts on the write lock on the CUSTOMER table

Transaction 2 puts on the write lock on the PARTS table

amazon web services

Data Validation and Troubleshooting Loads

- Two Amazon Redshift system tables can be helpful in troubleshooting data load issues:
  - STL_LOAD_ERRORS discovers the errors that occurred during specific loads.
  - STL_FILE_SCAN provides load times for specific files

amazon web services

Some Typical Load Errors

- Mismatch between data types in table and values in input data fields
- Mismatch between number of columns in table and number of fields in input data
- Mismatched quotes
  - Amazon Redshift supports both single and double quotes; however, these quotes must be balanced appropriately
- Incorrect format for date/time data in input files
- Out-of-range values in input files (for numeric columns)
- Number of distinct values for a column exceeds the limitation for its compression encoding

amazon web services

Best Practices for Loading Data

- Use a COPY Command to load data
- Use a single COPY command per table
- Split your data into multiple files
- Compress your data files with GZIP or LZOP
- Use multi-row inserts whenever possible
- Bulk insert operations (INSERT INTO…SELECT and CREATE TABLE AS) provide high performance data insertion
- Use Amazon Kinesis Firehose for Streaming Data direct load to S3 and/or Redshift

amazon web services

Best Practices for Loading Data Continued

- Load your data in sort key order to avoid needing to vacuum
- Organize your data as a sequence of time-series tables, where each table is identical but contains data for different time ranges
- Use staging tables to perform an upsert
- Run the VACUUM command whenever you add, delete, or modify a large number of rows, unless you load your data in sort key order
- Increase the memory available to a COPY or VACUUM by increasing wlm_query_slot_count
- Run the ANALYZE command whenever you've made a non-trivial number of changes to your data to ensure your table statistics are current

amazon web services