# Building Your First Amazon Redshift Data Warehouse

## ANSWERING THE QUESTION, 'WHY AMAZON REDSHIFT'?

**Russ Thomas**
DATA ARCHITECT

@sqljudo www.sqljudo.com

## Course Assumptions

My best guess about your objectives.  Was I close?

Some background with relational and OLAP will be helpful

We will be moving along rapidly

## Overview

Redshift History

Redshift Architecture

Redshift Potential

Abacos Widgets

# Abacos Data Warehouse Needs

Massive size potential

Scale out capability

Compatibility with existing reporting and BI tools

## Abacos Data Warehouse Decision

**Amazon Redshift is our go-forward technology**

**This module will focus on why**

**Remaining modules will focus on how**

## What's in It for You

Experience building a fully functional DW on Redshift

Low to no cost trial

# History of Amazon Redshift

Data warehouses grow from MB to GB to TB and even PB

Scale out technologies such as Couchbase and Hadoop gain popularity as need for “big data” grows

Massively parallel relational databases such as Teradata or PDW continue to offer relational scale out architecture as expensive appliances

ParAccel develops a fork of Postgres 8.0.2 that scales out on commodity hardware

Amazon licenses ParAccel technologies and develops Amazon Redshift as an MPP DW running on cloud architecture

You begin taking a course on Redshift discovering a whole new world of possibilities

## Things to Remember About Amazon Redshift

Based on Postgres 8.0.2 but very much it’s own technology now

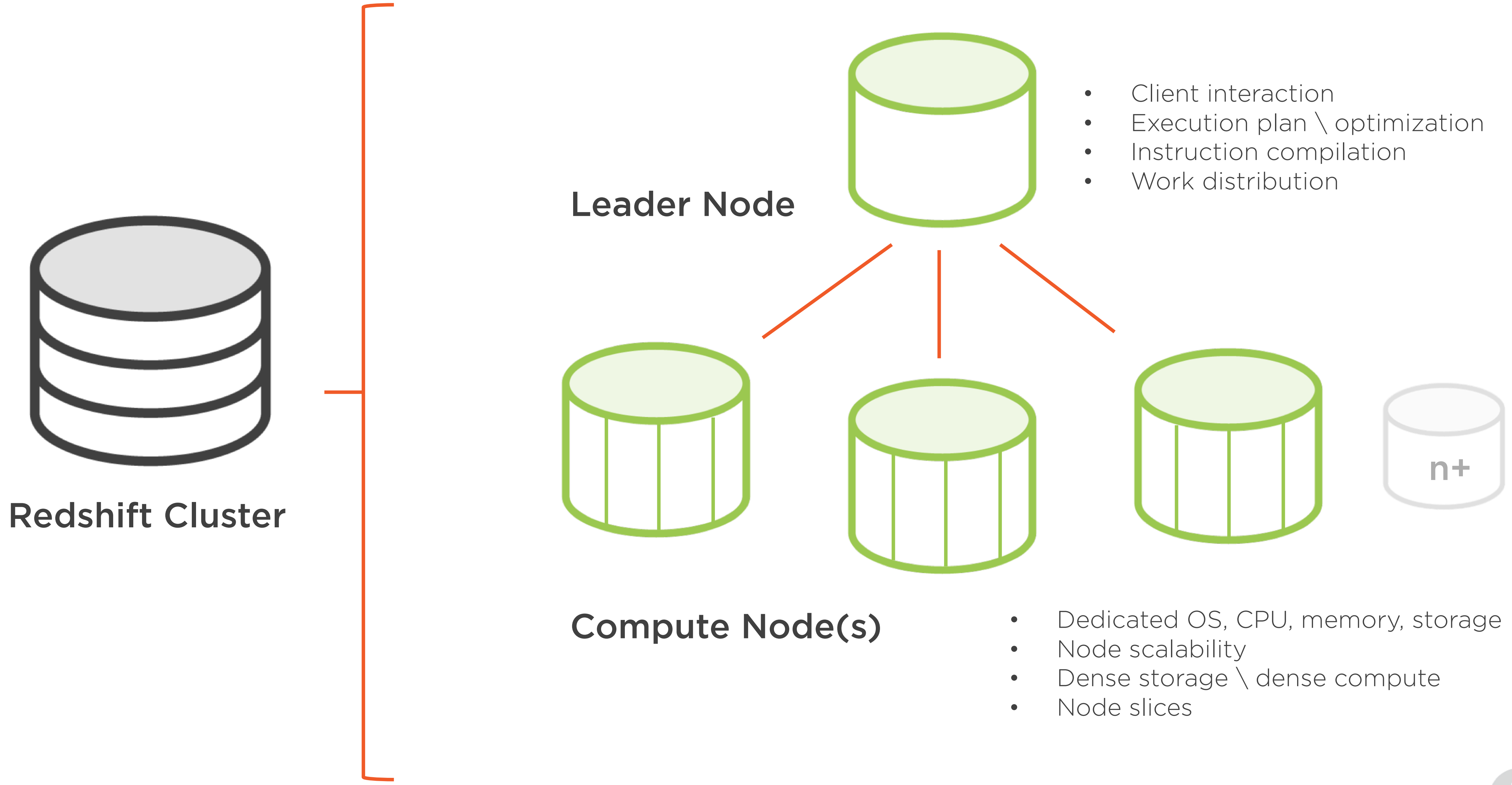
Not everything supported in Postgres is supported in Redshift

Nerd note about a “Redshift”

## Summary

**Amazon Redshift:**

- Massively parallel
- Relational (SQL) based
- Highly elastic cloud infrastructure



- Client interaction
- Execution plan \ optimization
- Instruction compilation
- Work distribution

- Dedicated OS, CPU, memory, storage
- Node scalability
- Dense storage \ dense compute
- Node slices

## Redshift Query Optimization

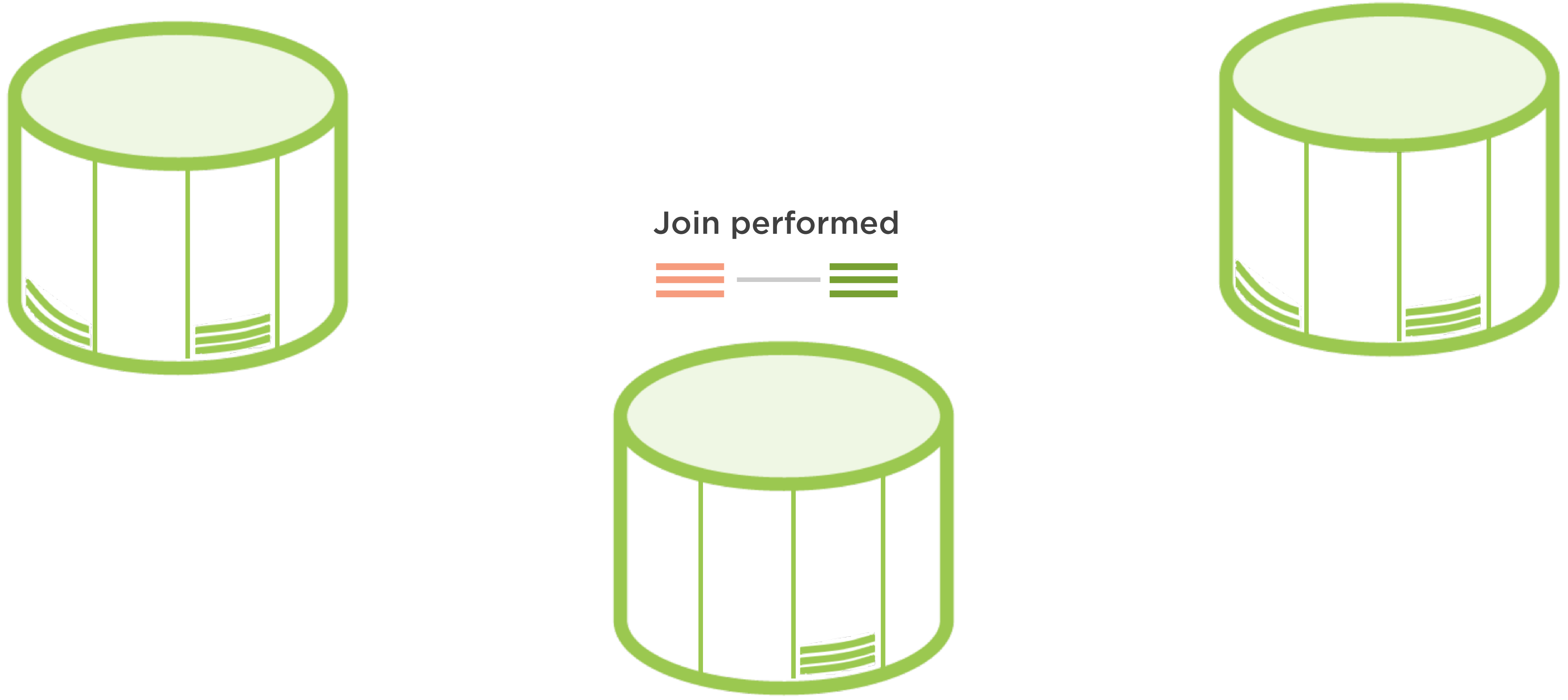
**Query optimization**

**Internal statistics and heuristics**

**ACID compliance**

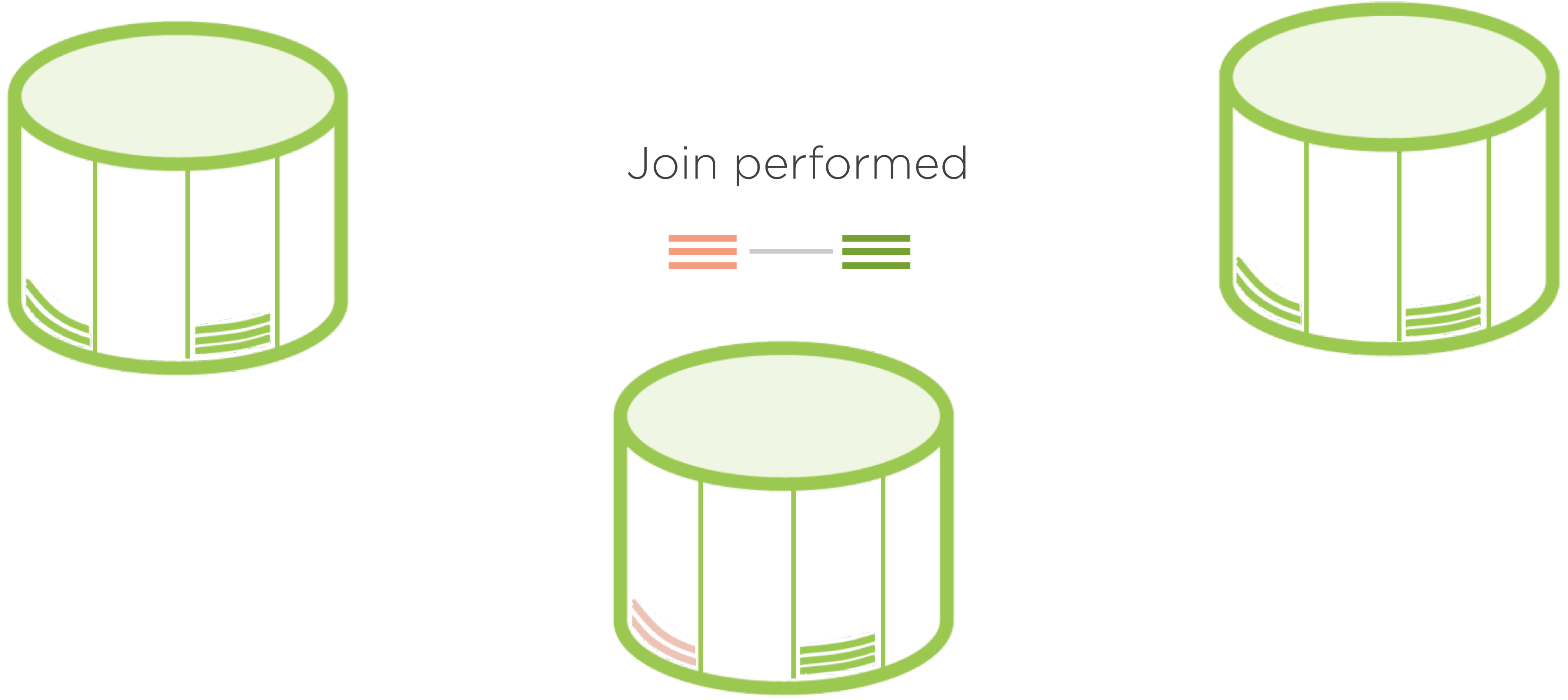
**Data distribution**

# Distribution Styles

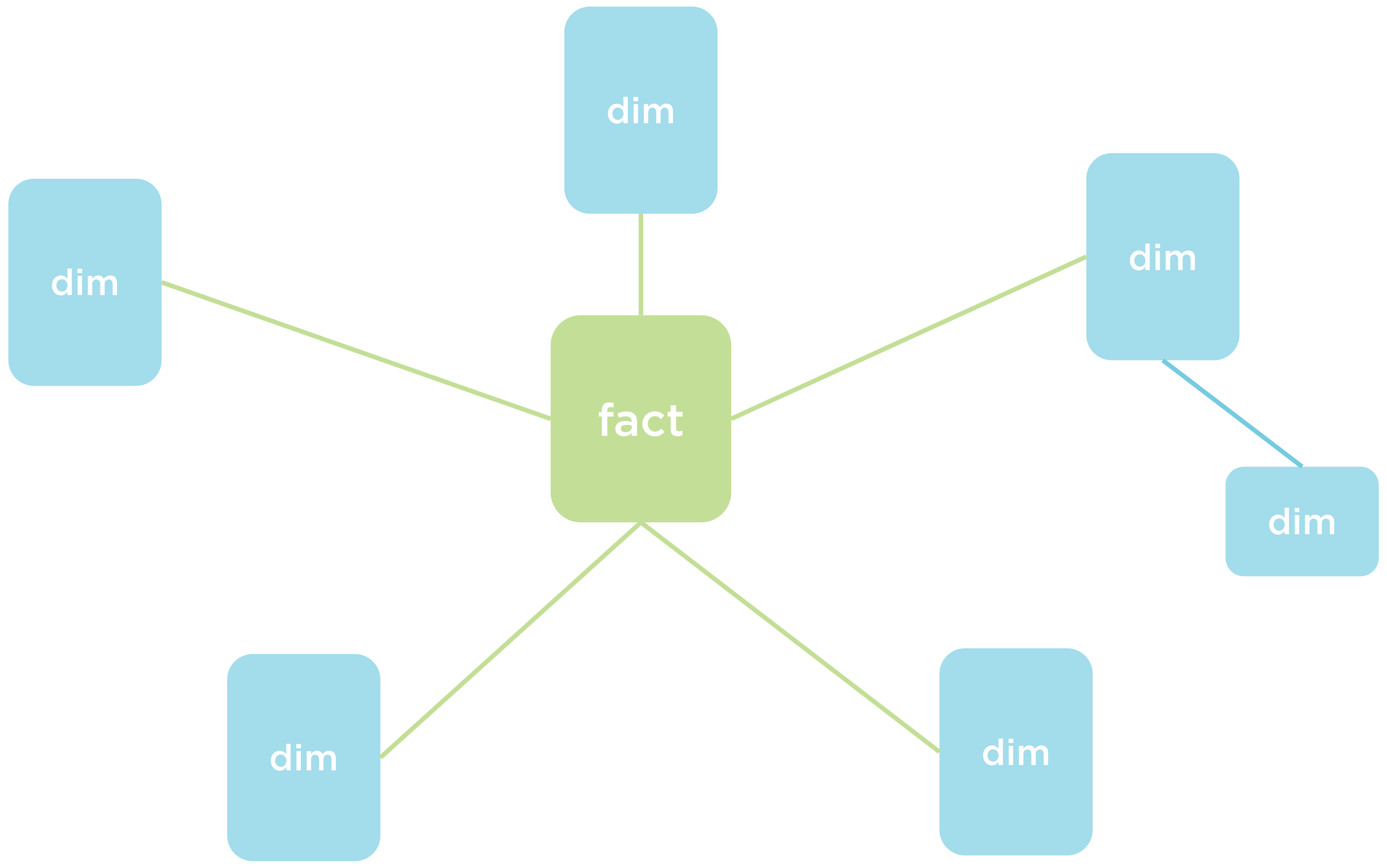


The goal of selecting a distribution style is minimizing impact of redistribution step

# Efficient Distribution Join



# Data Warehouse Schema



# Distribution Styles

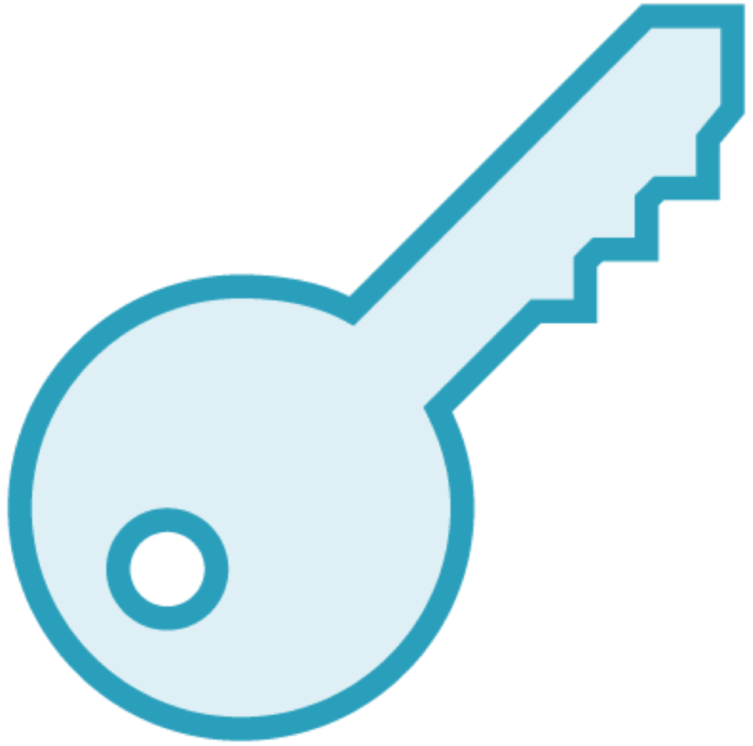
Even distribution

Key distribution

All distribution

## Even Distribution

- Distributes values evenly
- Default distribution style
- Good to average performance



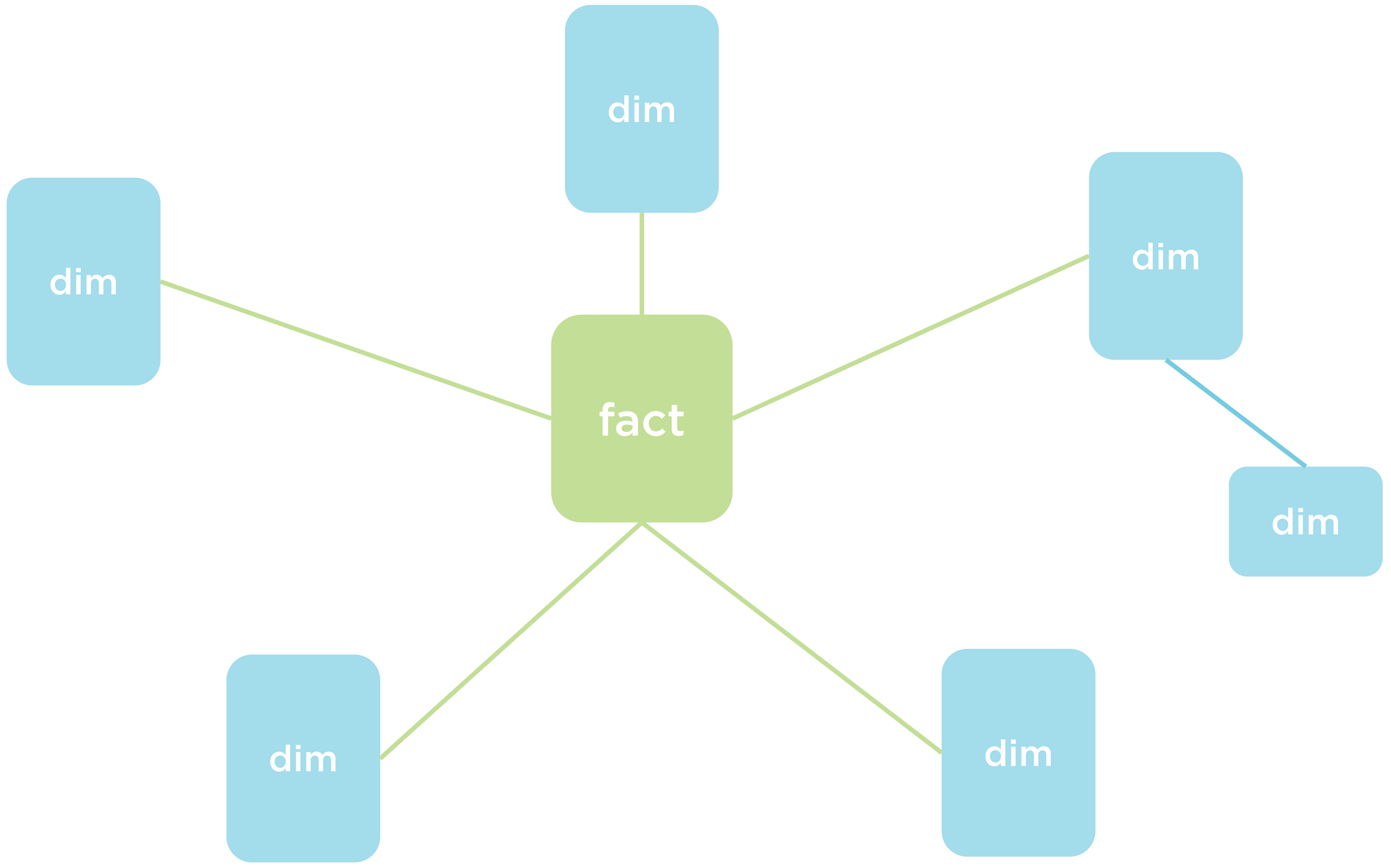
## Key Distribution

- Rows distributed by selected key
- Known relationships distributed across the same nodes
- Very effective for minimizing row redistribution costs

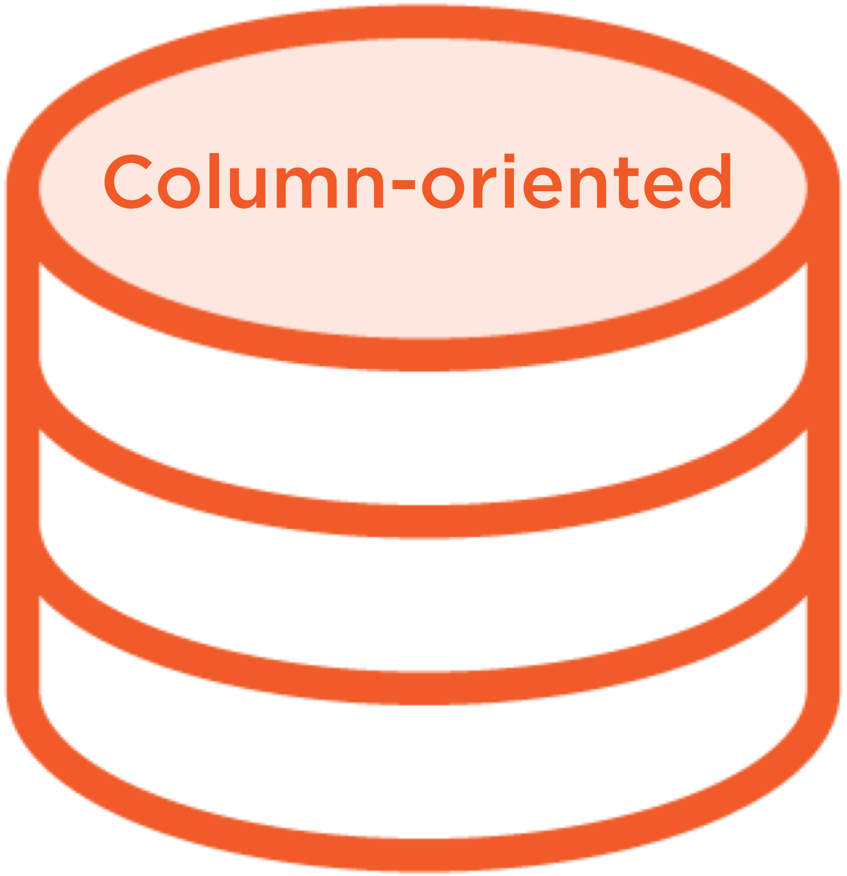
## All Distribution

- All rows are copied to all nodes
- Significant increase to storage and cost of insert, updates, and deletes
- Best for medium size tables that change rarely and actively joined

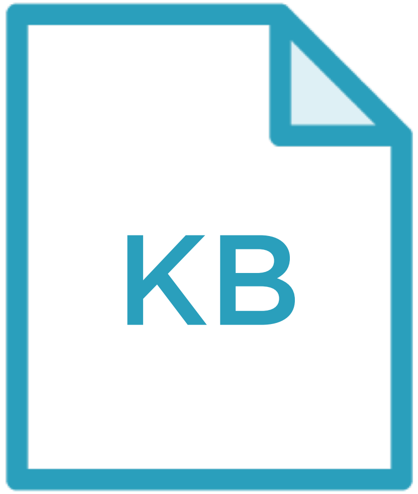
# Data Warehouse Schema



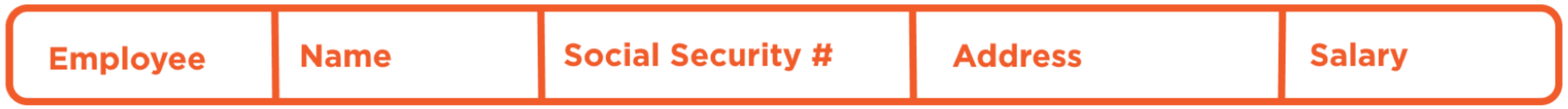
## Redshift Performance Features



## Row-based Architecture



Smaller storage chunks often called pages



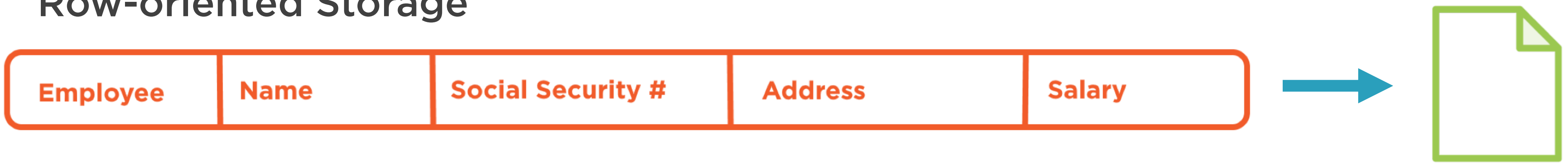
# Row-based Architecture

Smaller storage chunks often called pages

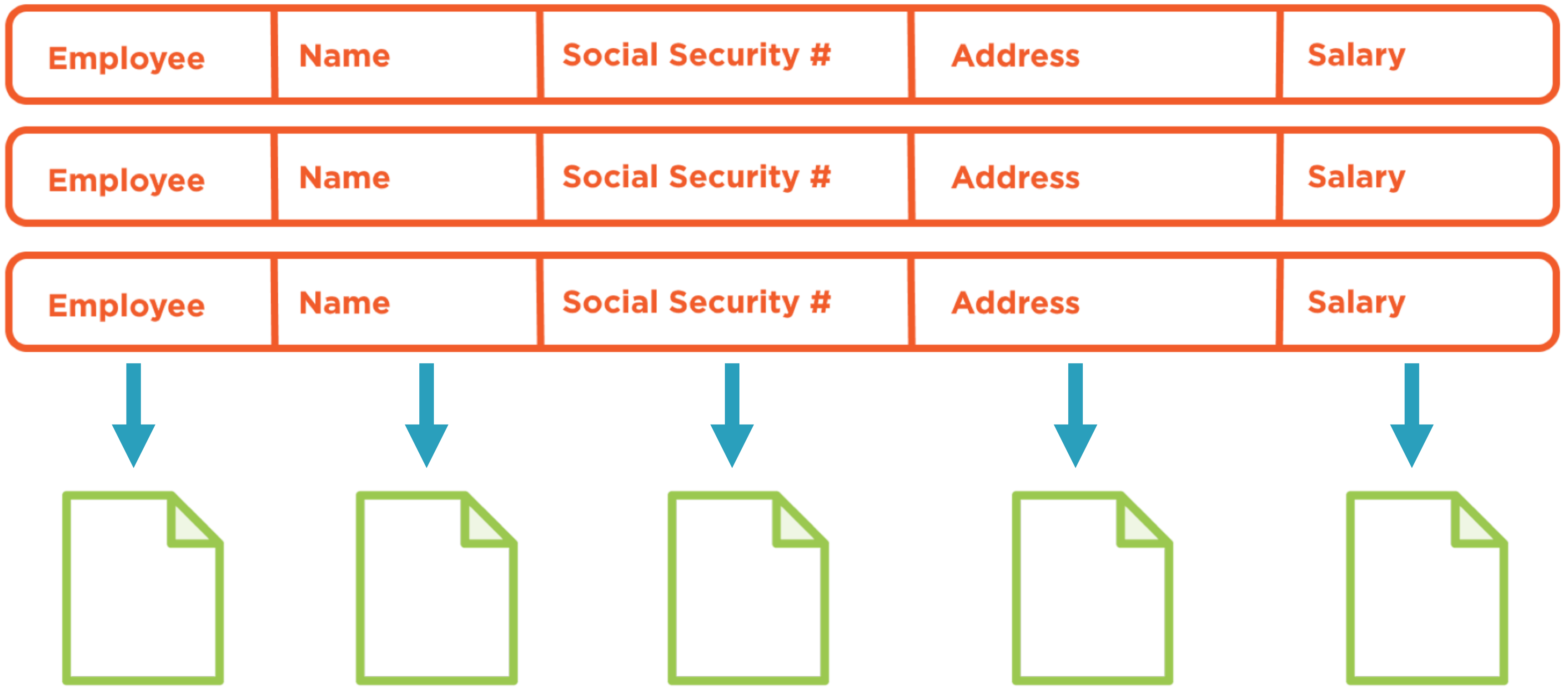
Pages contain all or most columns from each row

Reading large number of column values is inefficient

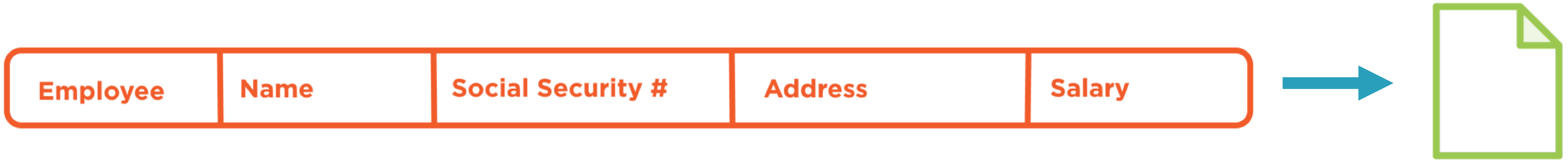
## Row-oriented Storage



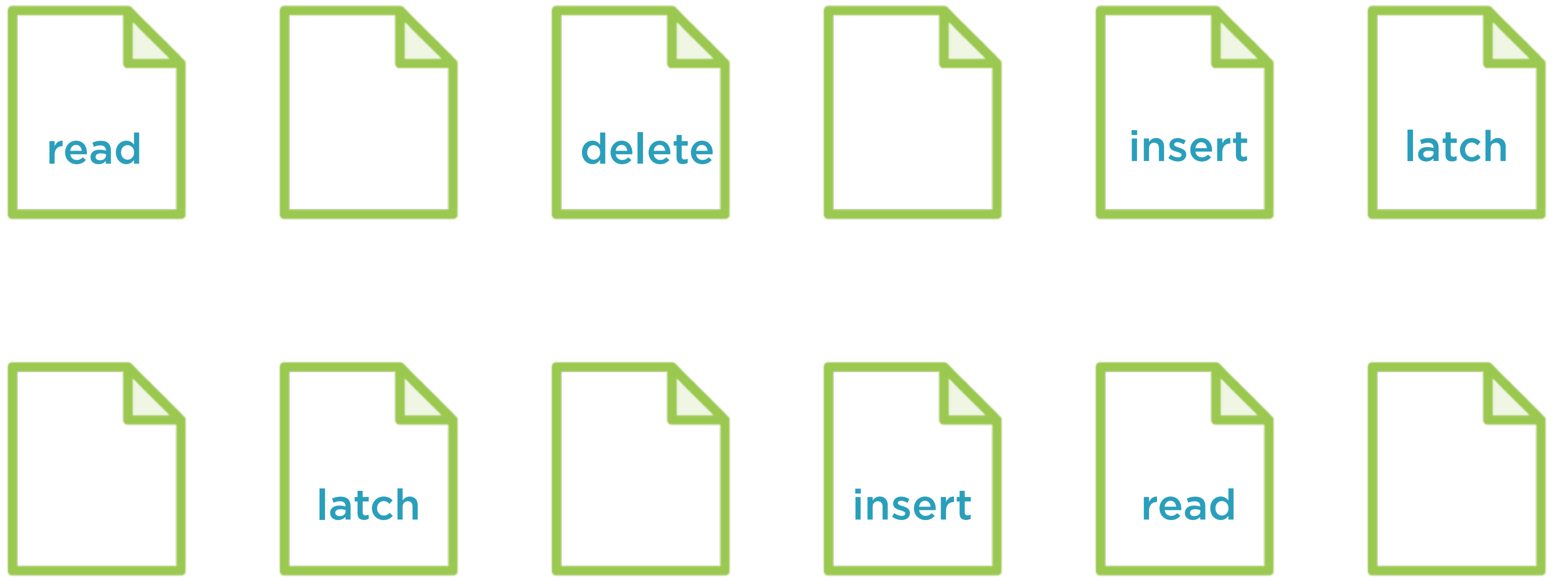
## Column-oriented Storage



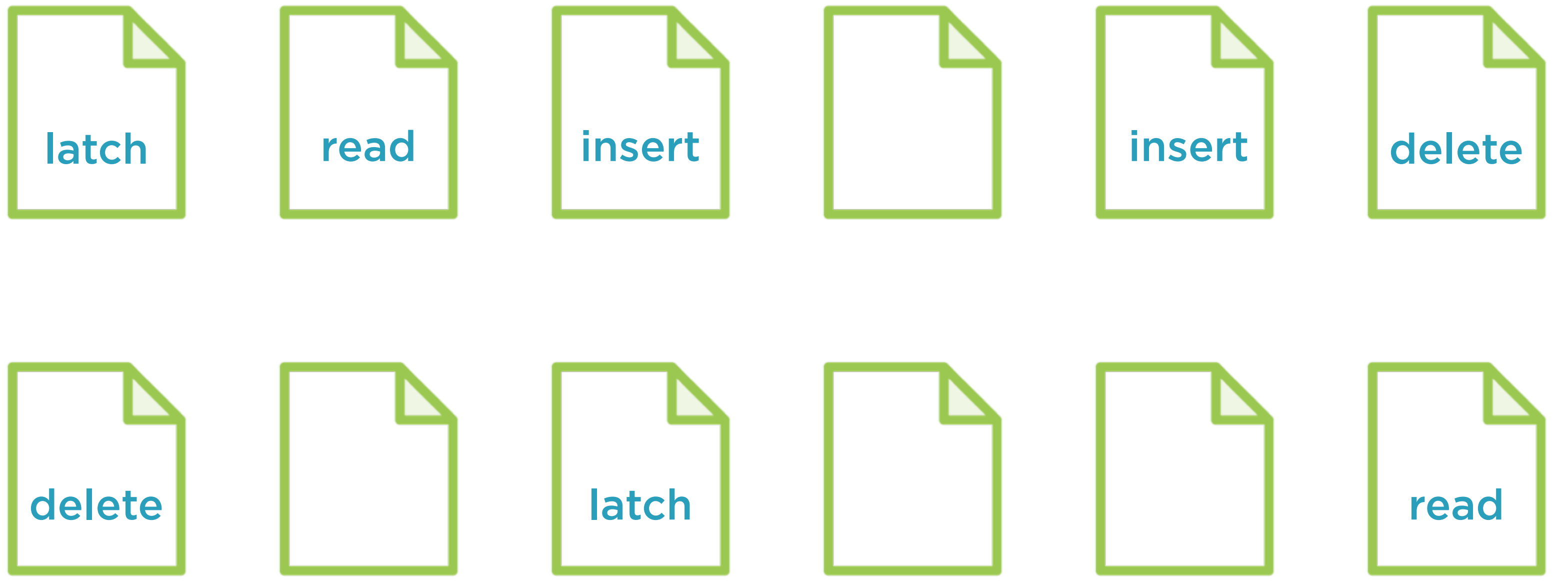
# Row-based Architecture



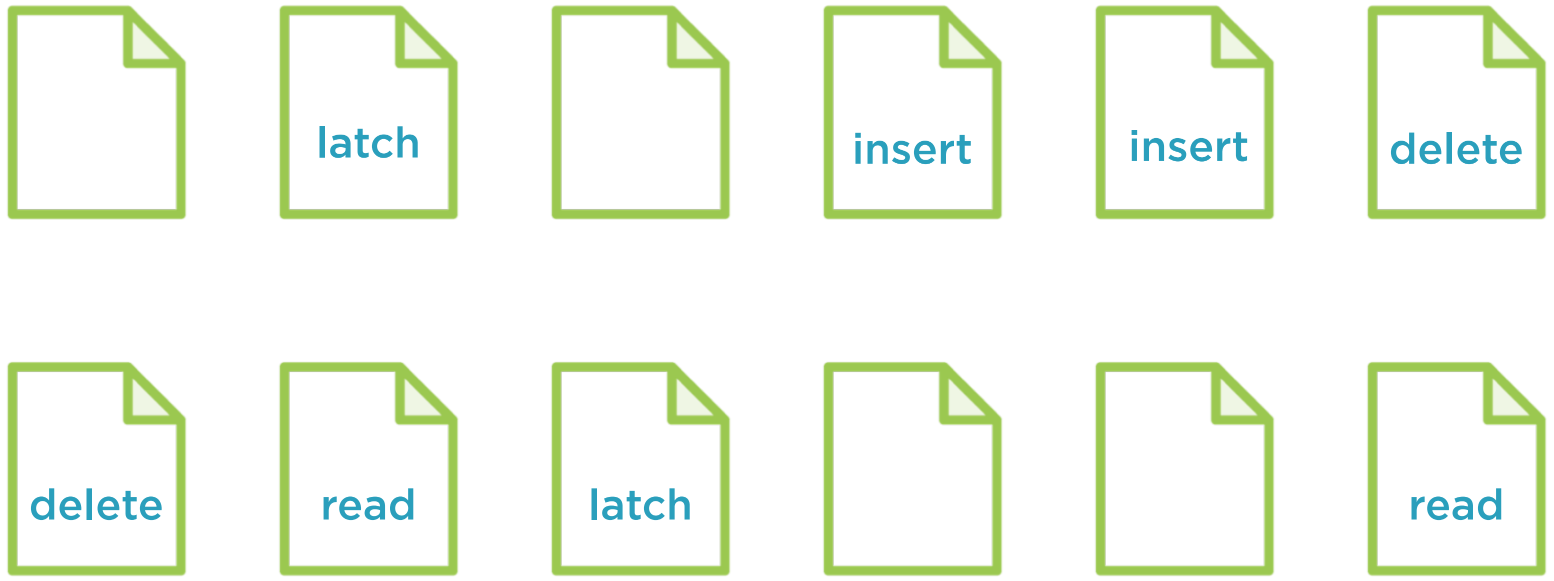
## Row-based Architecture



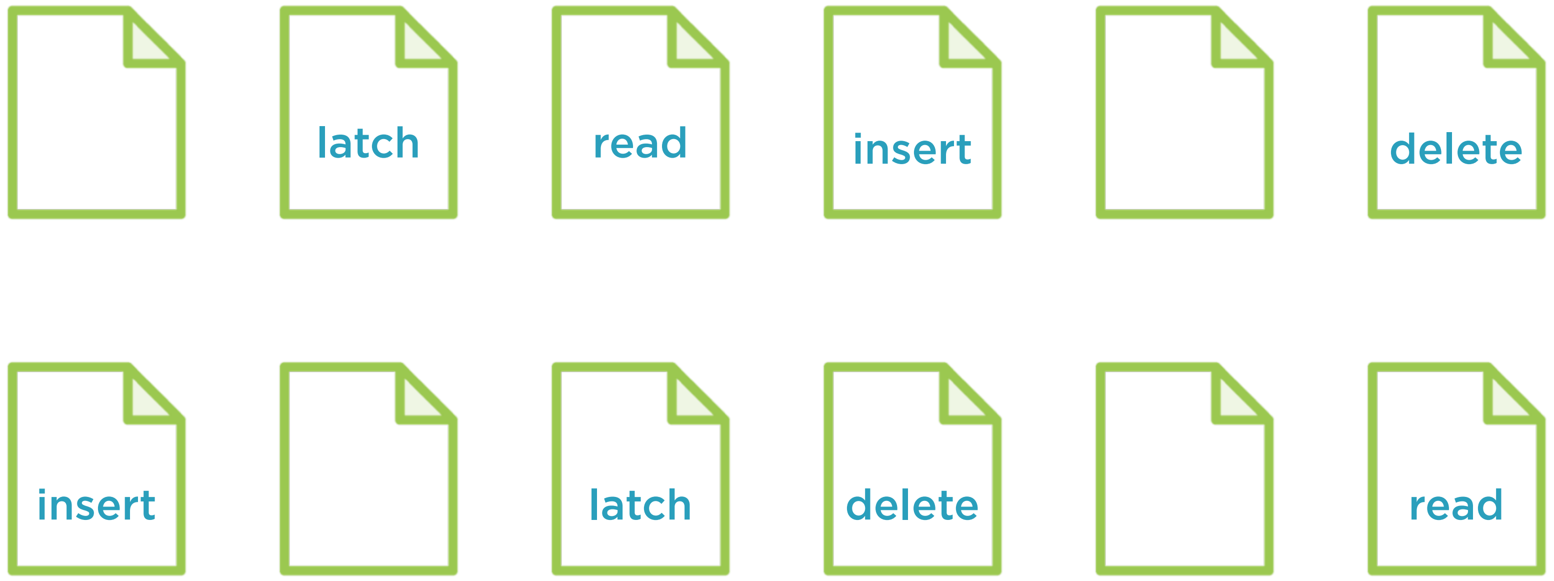
# Row-based Architecture



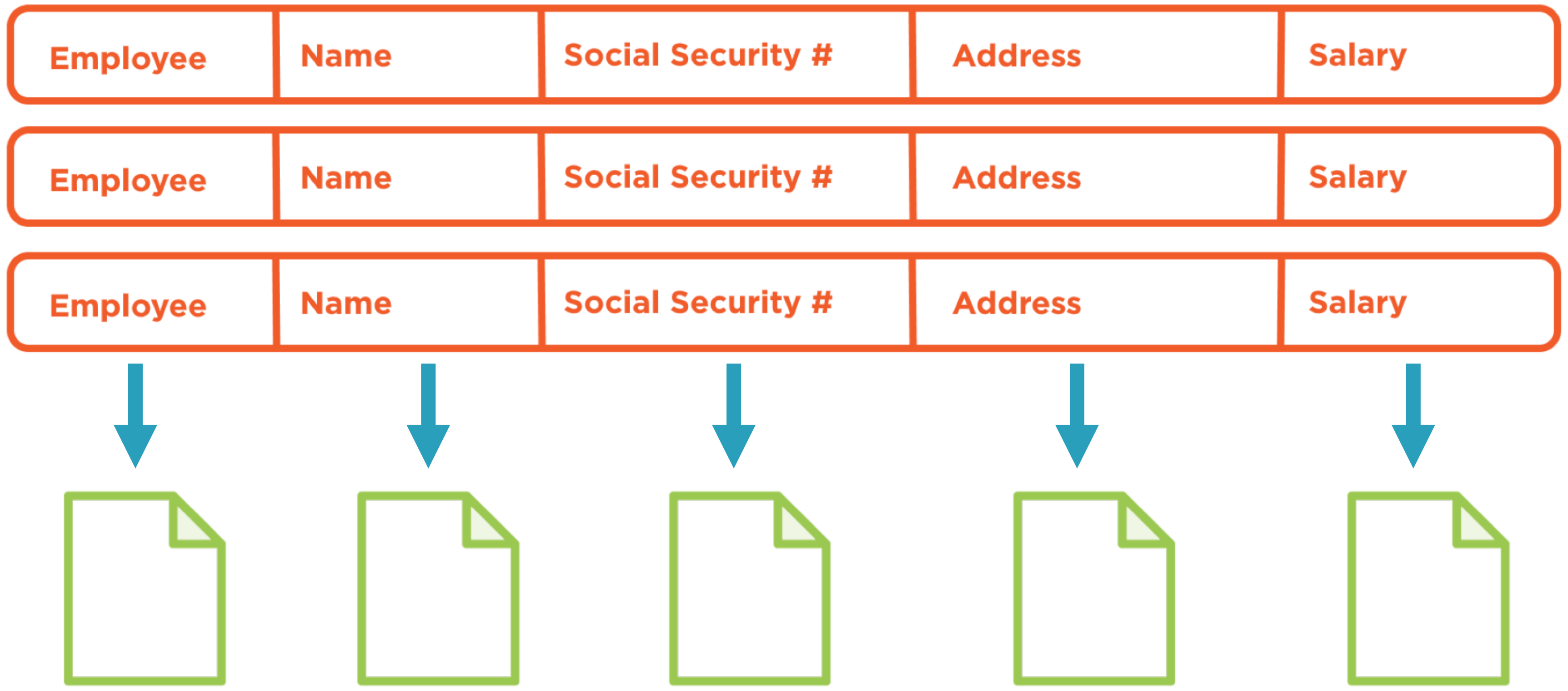
## Row-based Architecture



## Row-based Architecture



# Column-based Architecture



# Column-based Architecture

There is no clustered or non-clustered index

There is column-oriented, massively parallel, scale out architecture

Not effective for large amount of CRUD or small searches

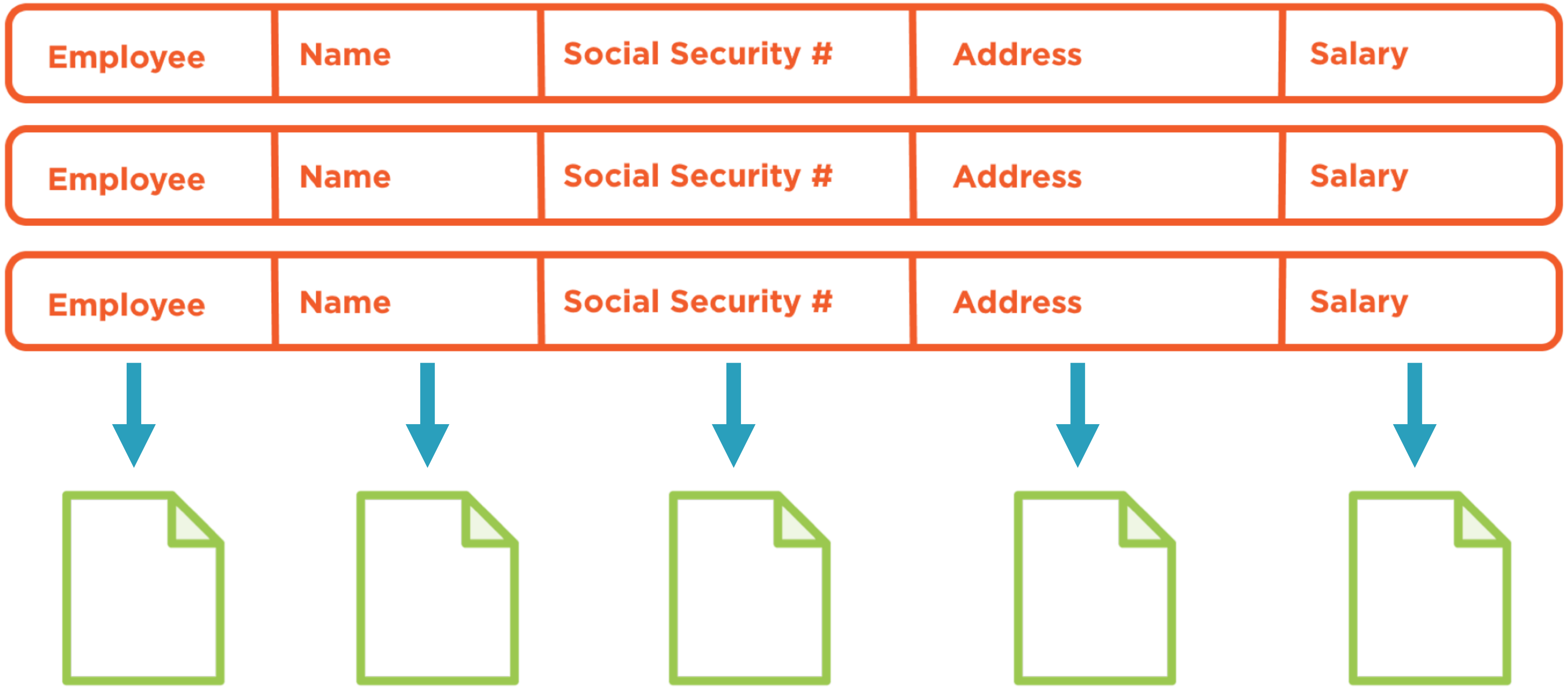
## AWS Database Offerings

- Relational database services
- NoSQL services
- Petabyte scale data warehousing

## Amazon Redshift Sorting

- A sortkey sorts data on disk
- Declared like distkey and primary key
- Part of create statement
- Only one sortkey permitted

# Column-based Architecture



## Effective Data Compression

Salary
Salary
Salary

- Optimized for single data type
- Speeds data reads
- Reduces overhead of redistribution
- Customizable compression

# Amazon Redshift Data Warehouse

Designed with 100% OLAP focus

Most limitations are from an OLTP perspective only

ABACOS WIDGETS

## Homework

- Review full schema creation scripts
- Identify different approaches
- Ensure sortkeys, distkeys make sense

**Next up: loading our data warehouse**