# Designing Schema for Hive

# Overview

Understand normalized storage in traditional databases

Understand why Hive stores data in denormalized form

# Hive Schema Design

Complex process

**Based on the type of dataset and kind of queries that are run**

Column structure and data types

Serialized forms in HDFS

Partitioning to improve query performance

Bucketing to tweak sampling and joins

# Normalized Storage in Traditional Databases

# Traditional Database Design

## Normalized data

**Data is stored in a granular form to minimize redundancy**

# Employee Information

**name**

**address**

**id**

**subordinates**

**department**

**grade**

id name grade department

id subordinates

id address

# Minimize Redundancy

**Employee Details**

**Employee Subordinates**

**Employee Address**

## Employee Details

| Id | Name | Department | Grade |
|---|---|---|---|
| 1 | Emily | Finance | 6 |

## Employee Subordinates

| Id | Subordinate Id |
|---|---|
| 1 | 2 |
| 1 | 3 |

## Employee Address

| Id | City | Zip Code |
|---|---|---|
| 1 | Palo Alto | 94305 |
| 2 | Seattle | 98101 |

## Employee Details

| Id | Name | Function | Grade |
|---|---|---|---|
| 1 | Emily | Finance | 6 |
| 2 | John | Finance | 3 |
| 3 | Ben | Finance | 4 |

# All employee details in one table

## Employee Subordinates

| Id | Subordinate Id |
|---|---|
| 1 | 2 |
| 1 | 3 |

Employees referenced only by ids everywhere else

## Employee Address

| Id | City | Zip Code |
|---|---|---|
| 1 | Palo Alto | 94305 |
| 2 | Seattle | 98101 |

**Data is made more granular by splitting it across multiple tables**

| Id | Name | Function | Grade |
|---|---|---|---|
| 1 | Emily | Finance | 6 |

| Id | Subordinate Id |
|---|---|
| 1 | 2 |
| 1 | 3 |

| Id | City | Zip Code |
|---|---|---|
| 1 | Palo Alto | 94305 |
| 2 | Seattle | 98101 |

# Normalization

| Id | Name | Function | Grade |
|---|---|---|---|
| 1 | Emily | Finance | 6 |

join

| Id | Subordinate Id |
|---|---|
| 1 | 2 |
| 1 | 3 |

# Query for Emily's department and her subordinates

# Normalization

**Minimizes redundancy, optimizes storage**

**Foreign keys to ensure valid joins**

**Updates in one location, no duplication of data**

# Denormalized Storage in Hive

# Denormalized Storage

## Denormalized data

**Data is compressed into one table to be read in a single operation**

# Denormalized Storage

**Disk space is very cheap**

**No foreign key constraints**

**Read operations, no data updates**

# Denormalized Storage

**Optimize the number of disk seeks**

**Store data for an entity in one location**

**Ignore redundancy, minimize joins**

# Denormalized Storage

| Id | Name | Function | Grade |
|---|---|---|---|
| 1 | Emily | Finance | 6 |
| 2 | John | Finance | 3 |
| 3 | Ben | Finance | 4 |

| Id | Subordinate Id |
|---|---|
| 1 | 2 |
| 1 | 3 |

# Denormalized Storage

| Id | Name | Function | Grade |
|---|---|---|---|
| 1 | Emily | Finance | 6 |
| 2 | John | Finance | 3 |
| 3 | Ben | Finance | 4 |

| Id | Subordinate Id |
|---|---|
| 1 | 2 |
| 1 | 3 |

| Id | Name | Function | Grade | Subordinates |
|---|---|---|---|---|
| 1 | Emily | Finance | 6 | **<ARRAY>** |
| 2 | John | Finance | 3 | |
| 3 | Ben | Finance | 4 | |

# Denormalized Storage

| Id | Name | Function | Grade |
|---|---|---|---|
| 1 | Emily | Finance | 6 |
| 2 | John | Finance | 3 |
| 3 | Ben | Finance | 4 |

| Id | Subordinate Id |
|---|---|
| 1 | 2 |
| 1 | 3 |

| Id | Name | Function | Grade | Subordinates |
|---|---|---|---|---|
| 1 | Emily | Finance | 6 | 2,3 |
| 2 | John | Finance | 3 | |
| 3 | Ben | Finance | 4 | |

# Denormalized Storage

| Id | Name | Function | Grade |
|---|---|---|---|
| 1 | Emily | Finance | 6 |
| 2 | John | Finance | 3 |
| 3 | Ben | Finance | 4 |

| Id | City | Zip Code |
|---|---|---|
| 1 | Palo Alto | 94305 |
| 2 | Seattle | 98101 |

| Id | Name | Function | Grade | Subordinates | Address |
|---|---|---|---|---|---|
| 1 | Emily | Finance | 6 | **2,3** | **<STRUCT>** |
| 2 | John | Finance | 3 | | |
| 3 | Ben | Finance | 4 | | |

# Denormalized Storage

| Id | Name | Function | Grade |
|---|---|---|---|
| 1 | Emily | Finance | 6 |
| 2 | John | Finance | 3 |
| 3 | Ben | Finance | 4 |

| Id | City | Zip Code |
|---|---|---|
| 1 | Palo Alto | 94305 |
| 2 | Seattle | 98101 |

| Id | Name | Function | Grade | Subordinates | Address |
|---|---|---|---|---|---|
| 1 | Emily | Finance | 6 | **2,3** | **Palo Alto, 94305** |
| 2 | John | Finance | 3 | | |
| 3 | Ben | Finance | 4 | | |

# Denormalized Storage

| Id | Name | Function | Grade | Subordinates | Address |
|---|---|---|---|---|---|
| 1 | Emily | Finance | 6 | **2,3** | **Palo Alto, 94305** |
| 2 | John | Finance | 3 | | |
| 3 | Ben | Finance | 4 | | |

**Store everything related to an employee in the same table**

# Denormalized Storage

| Id | Name | Function | Grade | Subordinates | Address |
|---|---|---|---|---|---|
| 1 | Emily | Finance | 6 | **2,3** | **Palo Alto, 94305** |
| 2 | John | Finance | 3 | | |
| 3 | Ben | Finance | 4 | | |

**Get all details about an employee in one read operation**

# Summary

**Understood the differences between normalized and denormalized storage**

**Understood why Hive stores data in denormalized form**