cloudera®

Data File Partitioning

Chapter 8

Course Chapters

1	Introduction	Course Introduction
2	Introduction to Hadoop and the Hadoop Ecosystem	Introduction to Hadoop
3	Hadoop Architecture and HDFS	
4	Importing Relational Data with Apache Sqoop	**Importing and Modeling Structured Data**
5	Introduction to Impala and Hive	
6	Modeling and Managing Data with Impala and Hive	
7	Data Formats	
8	**Data File Partitioning**	
9	Capturing Data with Apache Flume	Ingesting Streaming Data
10	Spark Basics	Distributed Data Processing with Spark
11	Working with RDDs in Spark	
12	Aggregating Data with Pair RDDs	
13	Writing and Deploying Spark Applications	
14	Parallel Processing in Spark	
15	Spark RDD Persistence	
16	Common Patterns in Spark Data Processing	
17	Spark SQL and DataFrames	
18	Conclusion	Course Conclusion

Data File Partitioning

In this chapter you will learn

- **How to improve query performance with data file partitioning**
- **How to create and populate partitioned tables in Impala and Hive**

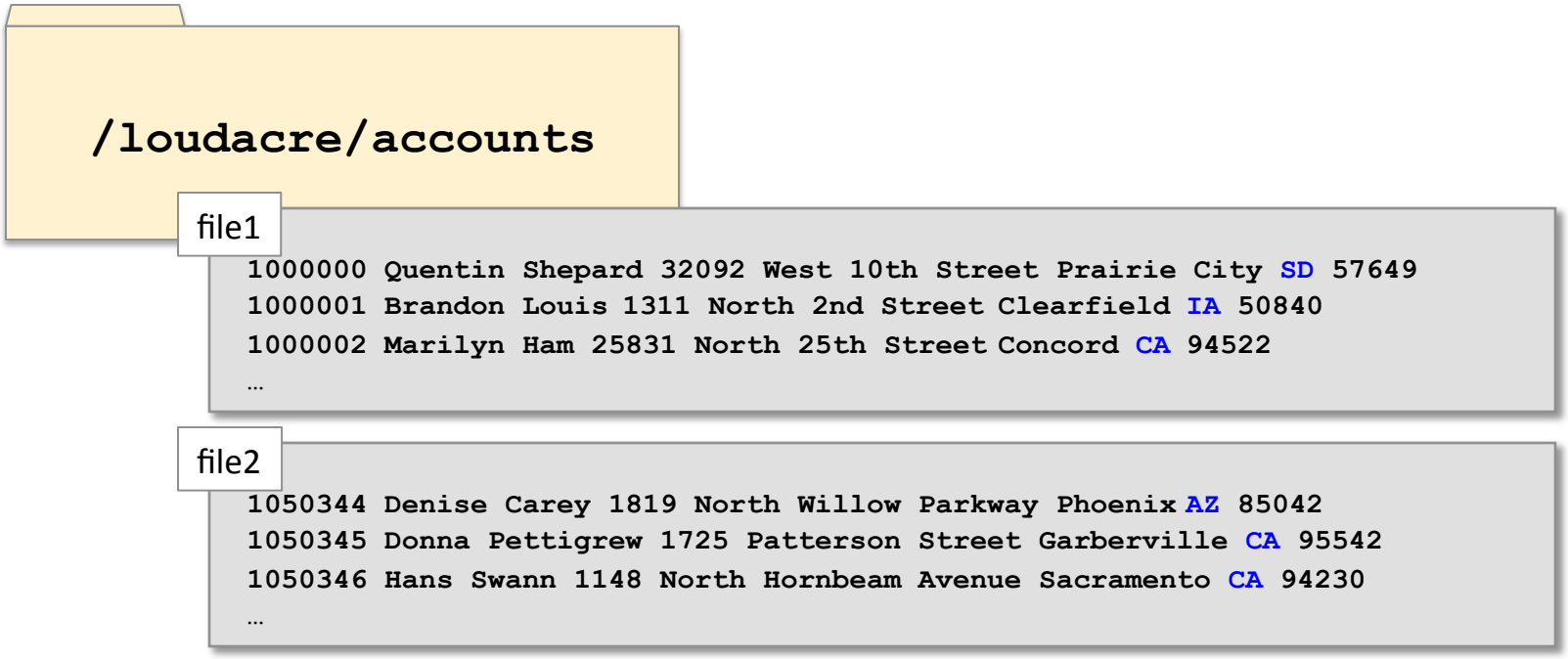
Chapter Topics

Data File Partitioning | **Importing and Modeling Structured Data**

- **Partitioning Overview**
- Partitioning in Impala and Hive
- Conclusion
- Homework: Partition Data in Impala or Hive

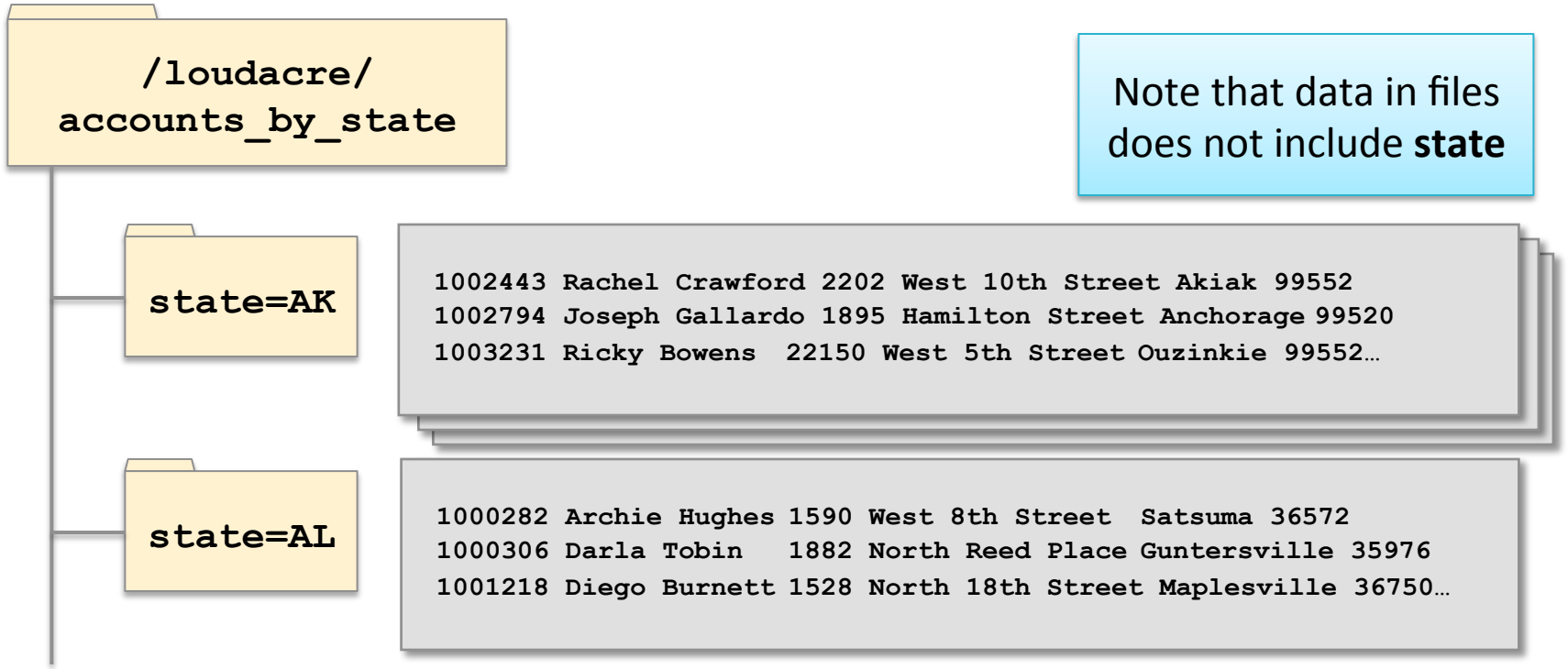
Data Storage Partitioning (1)

- **By default, all files in a data set are stored in a single HDFS directory**
  - *All* files in the directory are read during analysis or processing
  - "Full table scan"



Data Storage Partitioning (2)

- ***Partitioning* subdivides the data**
  - Analysis can be done on only the relevant subset of data
  - Potentially much faster!
- **Hadoop partitions using subdirectories**



Hadoop Partitioning

- **Partitioning is involved at two phases**
  - Storage – putting the data into correct partition (subdirectory)
  - Retrieval – getting the data out of the correct partition based on the query or analysis being done
- **Hadoop with built-in support for partitioning**
  - Hive and Impala (covered in next section)
  - Sqoop – When using the **`--hive-import`** option you can specify flags **`--hive-partition-key`** and **`--hive-partition-value`**
- **Other tools can be used to store partitioned data**
  - Spark and MapReduce
  - Flume (at ingestion)

Chapter Topics

- Partitioning Overview
- **Partitioning in Impala and Hive**
- Conclusion
- Homework: Partition Data in Impala or Hive

Example: Impala/Hive Partitioning Accounts By State (1)

- **Example: `accounts` is a non-partitioned table**

```
CREATE EXTERNAL TABLE accounts(
    cust_id INT,
    fname STRING,
    lname STRING,
    address STRING,
    city STRING,
    state STRING,
    zipcode STRING)
  ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ','
  LOCATION '/loudacre/accounts';
```

Example: Impala/Hive Partitioning Accounts By State (2)

- **What if most of Loudacre's analysis on the customer table was done by state? For example:**

```
SELECT fname, lname
  FROM accounts
  WHERE state='NY';
```

- **By default, all queries have to scan *all* files in the directory**
- **Use partitioning to store data in separate files by state**
  - State-based queries scan only the relevant files

Example: Impala/Hive Partitioning Accounts By State (3)

- **Create a partitioned table using PARTITIONED BY**

```
CREATE EXTERNAL TABLE accounts_by_state(
    cust_id INT,
    fname STRING,
    lname STRING,
    address STRING,
    city STRING,
    state STRING,
    zipcode STRING)
  PARTITIONED BY (state STRING)
  ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ','
  LOCATION '/loudacre/accounts_by_state';
```

Partition Columns

- **The partition column is displayed if you `DESCRIBE` the table**

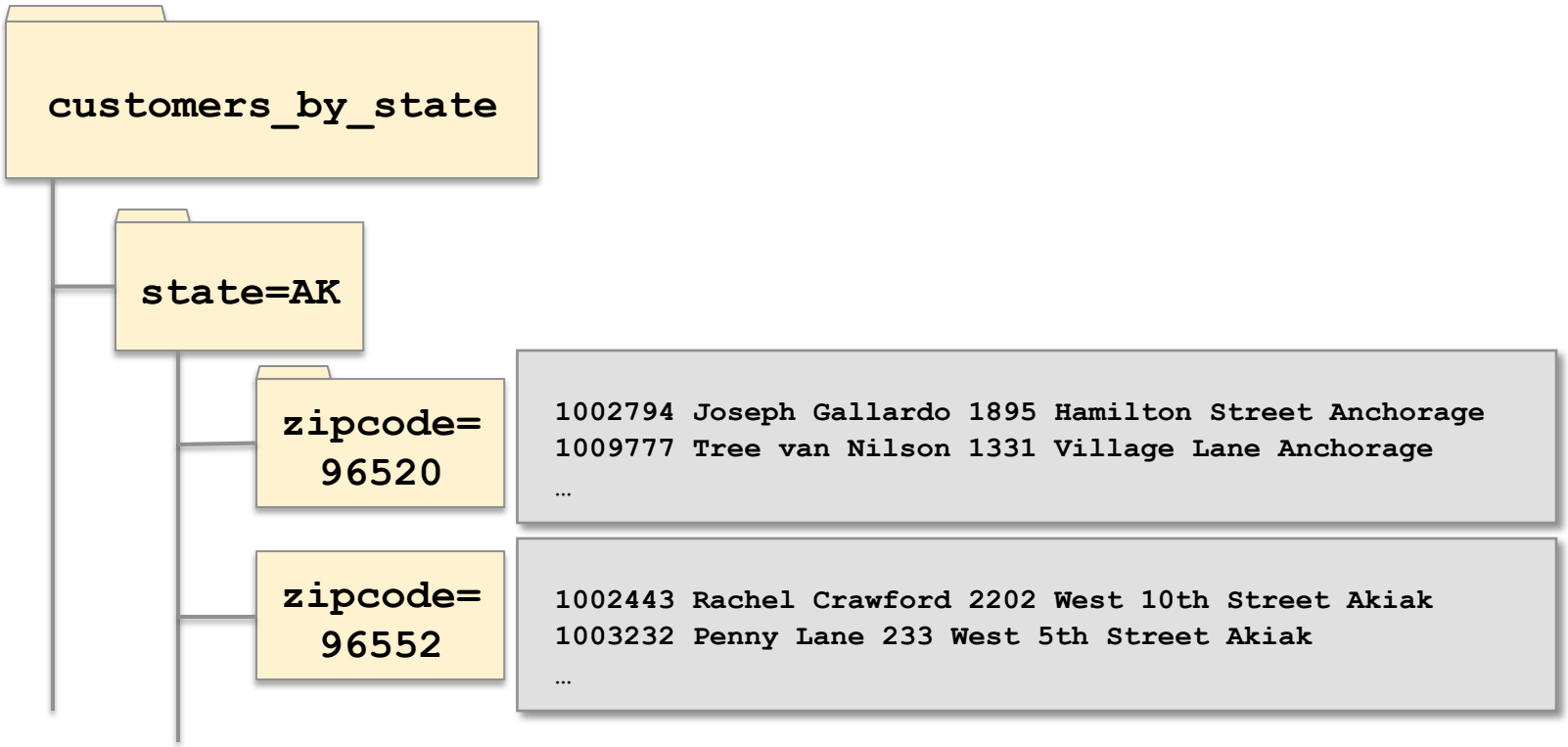
```
DESCRIBE accounts_by_state;
+---------+--------+---------+
| name    | type   | comment |
+---------+--------+---------+
| cust_id | int    |         |
| fname   | string |         |
| lname   | string |         |
| address | string |         |
| city    | string |         |
| zipcode | string |         |
| state   | string |         |
+---------+--------+---------+
```

A partition column is a "virtual column"; data is not stored in the file

Nested Partitions

- **You can also created nested partitions**

```
… PARTITIONED BY (state STRING, zipcode STRING)
```



Loading Data Into a Partitioned Table

- **Dynamic partitioning**
  - Impala/Hive add new partitions automatically as needed at load time
  - Data is stored into the correct partition (subdirectory) based on column value
- **Static partitioning**
  - You define new partitions using `ADD PARTITION`
  - When loading data, you specify which partition to store data in

Dynamic Partitioning

- **We can create new partitions dynamically from existing data**

```
INSERT OVERWRITE TABLE accounts_by_state
  PARTITION(state)
  SELECT cust_id, fname, lname, address,
    city, zipcode, state FROM accounts;
```

- **Partitions are automatically created based on the value of the *last* column**
  - If the partition does not already exist, it will be created
  - If the partition does exist, it will be overwritten

Static Partitioning Example: Partition Calls by Day (1)

- **Loudacre's customer service phone system generates logs detailing calls received**
  - Analysts use this data to summarize previous days' calls
  - For example:

```
SELECT event_type, COUNT(event_type)
  FROM call_log
  WHERE call_date = '2014-10-01'
  GROUP BY event_type;
```

Static Partitioning Example: Partition Calls by Day (2)

- **Logs are generated daily, e.g.**

call-20141001.log

```
19:45:19,312-555-7834,CALL_RECEIVED
19:45:23,312-555-7834,OPTION_SELECTED,Shipping
19:46:23,312-555-7834,ON_HOLD
19:47:51,312-555-7834,AGENT_ANSWER,Agent ID N7501
19:48:37,312-555-7834,COMPLAINT,Item not received
19:48:41,312-555-7834,CALL_END,Duration: 3:22
…
```

call-20141002.log

```
03:45:01,505-555-2345,CALL_RECEIVED
03:45:09,505-555-2345,OPTION_SELECTED,Billing
03:56:21,505-555-2345,AGENT_ANSWER,Agent ID A1503
03:57:01,505-555-2345,QUESTION
…
```

Static Partitioning Example: Partition Calls by Day (3)

- **In the previous example, existing data was partitioned dynamically based on a column value**
- **This time we use static partitioning**
  - Because the data files do not include the partitioning data

Static Partitioning Example: Partition Calls by Day (4)

- **The partitioned table is defined the same way**

```
CREATE TABLE call_logs (
    call_time STRING,
    phone STRING,
    event_type STRING,
    details STRING)
 PARTITIONED BY (call_date STRING)
  ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ',';
```

Loading Data Into Static Partitions (1)

- **With static partitioning, you create new partitions as needed**
- **e.g. For each new day of call log data, add a partition:**

```
ALTER TABLE call_logs
  ADD PARTITION (call_date='2014-10-02');
```

- **This command**
  1. Adds the partition to the table's metadata
  2. Creates subdirectory
     `/user/hive/warehouse/call_logs/call_date=2014-10-02`

Loading Data Into Static Partitions (2)

- **Then load the day's data into the correct partition**

```
LOAD DATA INPATH '/mystaging/call-20141002.log'
  INTO TABLE call_logs
  PARTITION(call_date='2014-10-02');
```

  - This command moves the HDFS file **`call-20141002.log`** to the partition subdirectory

- **To overwrite all data in a partition**

```
LOAD DATA INPATH '/mystaging/call-20141002.log'
  INTO TABLE call_logs OVERWRITE
  PARTITION(call_date='2014-10-02');
```

Hive Only: Shortcut for Loading Data Into Partitions

- **Hive will create a new partition if the one specified doesn't exist**


```
LOAD DATA INPATH '/mystaging/call-20141002.log'
  INTO TABLE call_logs
  PARTITION(call_date='2014-10-02');
```

- **This command**
  1. Adds the partition to the table's metadata if it doesn't exist
  2. Creates subdirectory **`/user/hive/warehouse/call_logs/call_date=2014-10-02`** if it doesn't exist
  3. Moves the HDFS file **`call-20141002.log`** to the partition subdirectory

Viewing, Adding, and Removing Partitions

- **To view the current partitions in a table**

```
SHOW PARTITIONS call_logs;
```

- **Use `ALTER TABLE` to add or drop partitions**

```
ALTER TABLE call_logs
   ADD PARTITION (call_date='2013-06-05')
   LOCATION '/loudacre/call_logs/call_date=2013-06-05';
```

```
ALTER TABLE call_logs
   DROP PARTITION (call_date='2013-06-06');
```

Creating Partitions from Existing Partition Directories in HDFS

- **Partition directories in HDFS can be created and populated outside Hive or Impala**
  - For example, by a Spark or MapReduce application
- **In Hive, use the `MSCK REPAIR TABLE` command to create (or recreate) partitions for an existing table**


```
MSCK REPAIR TABLE call_logs;
```

When To Use Partitioning

- **Use partitioning for tables when**
  - Reading the entire data set takes too long
  - Queries almost always filter on the partition columns
  - There are a reasonable number of different values for partition columns
  - The data generation or ETL process segments data by file or directory names
    - Partition column values are not in the data itself

When *Not* To Use Partitioning

- **Avoid partitioning data into numerous small data files**
  - Don't partition on columns with too many unique values
- **Caution: This can happen easily when using dynamic partitioning!**
  - For example, partitioning customers by first name could produce thousands of partitions

Partitioning in Hive (1)

- **In older versions of Hive, dynamic partitioning is not enabled by default**
  - Enable it by setting these two properties

```
SET hive.exec.dynamic.partition=true;
SET hive.exec.dynamic.partition.mode=nonstrict;
```

- **Note: Hive variables set in Beeline are for the current session only**
  - Your system administrator can configure settings permanently

Partitioning in Hive (2)

- **Caution: if the partition column has many unique values, many partitions will be created**
- **Three Hive configuration properties exist to limit this**
  - **`hive.exec.max.dynamic.partitions.pernode`**
    - Maximum number of dynamic partitions that can be created by any given node involved in a query
    - Default 100
  - **`hive.exec.max.dynamic.partitions`**
    - Total number of dynamic partitions that can be created by one HiveQL statement
    - Default 1000
  - **`hive.exec.max.created.files`**
    - Maximum total files (on all nodes) created by a query
    - Default 100000

Chapter Topics

- Partitioning Overview
- Partitioning in Impala and Hive
- **Conclusion**
- Homework: Partition Data in Impala or Hive

Essential Points

- **Partitioning splits table storage by column values for improved query performance**
- **Partitions are HDFS directories**
  - Names follow the format ***`column=value`***
- **Partitions can be defined and loaded dynamically or statically**
- **Only partition on columns with a reasonable number of possible values**

Bibliography

The following offer more information on topics discussed in this chapter

- **Impala documentation on partitioning**
  - `http://tiny.cloudera.com/impalapart`
- **Improving Query Performance Using Partitioning in Apache Hive (Cloudera Engineering Blog)**
  - `http://tiny.cloudera.com/partblog`

Chapter Topics

- Partitioning Overview
- Partitioning in Impala and Hive
- Conclusion
- **Homework: Partition Data in Impala or Hive**

Homework: Partition Data in Impala or Hive

- **In this homework assignment you will**
  - Create a table for accounts that is partitioned by area code
- **Please refer to the Homework description**