Data File Partitioning

Prof. Hyuk-Yoon Kwon

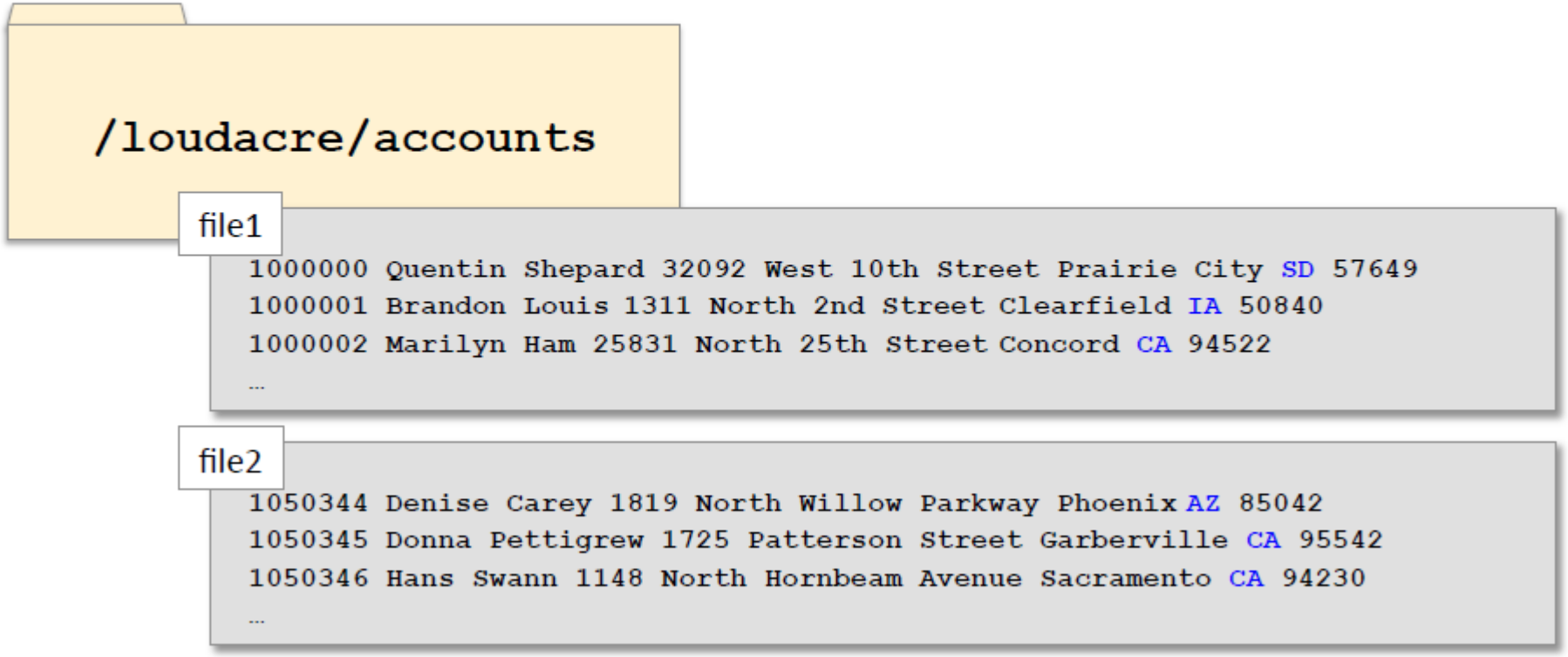
Data File Partitioning

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

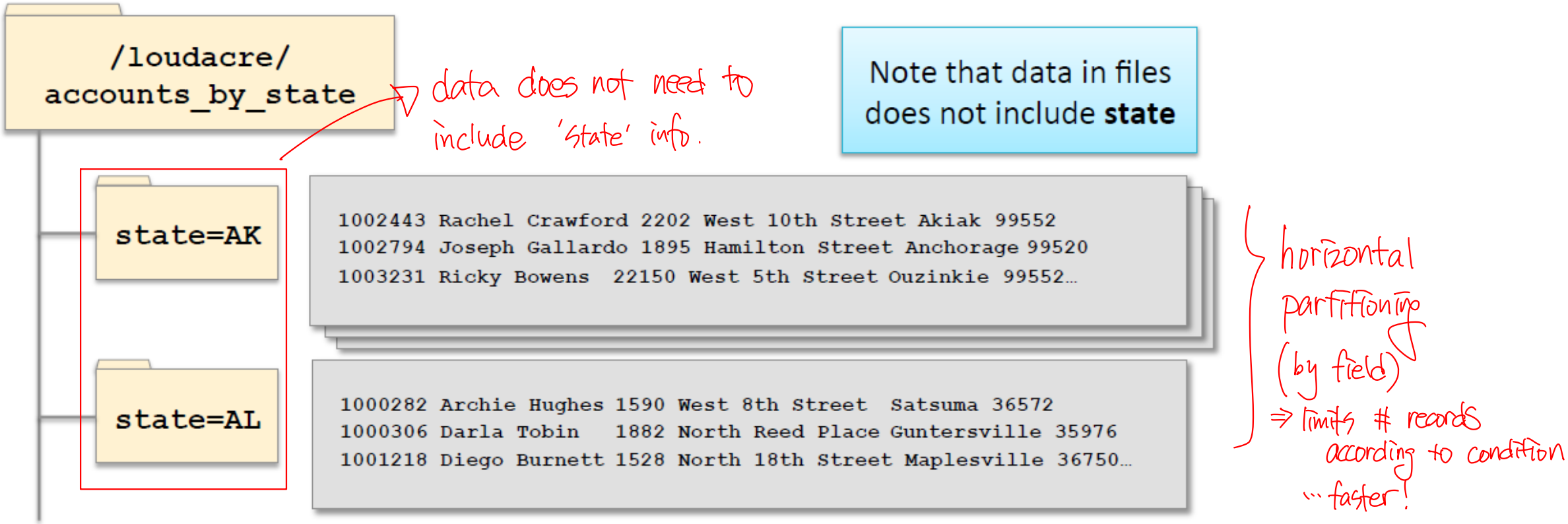
Data Storage Partitioning (1)

- **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" each file can be partitioned in HDFS



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) loading data
  - Retrieval – getting the data out of the correct partition based on the query or analysis being done querying data
- **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)

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

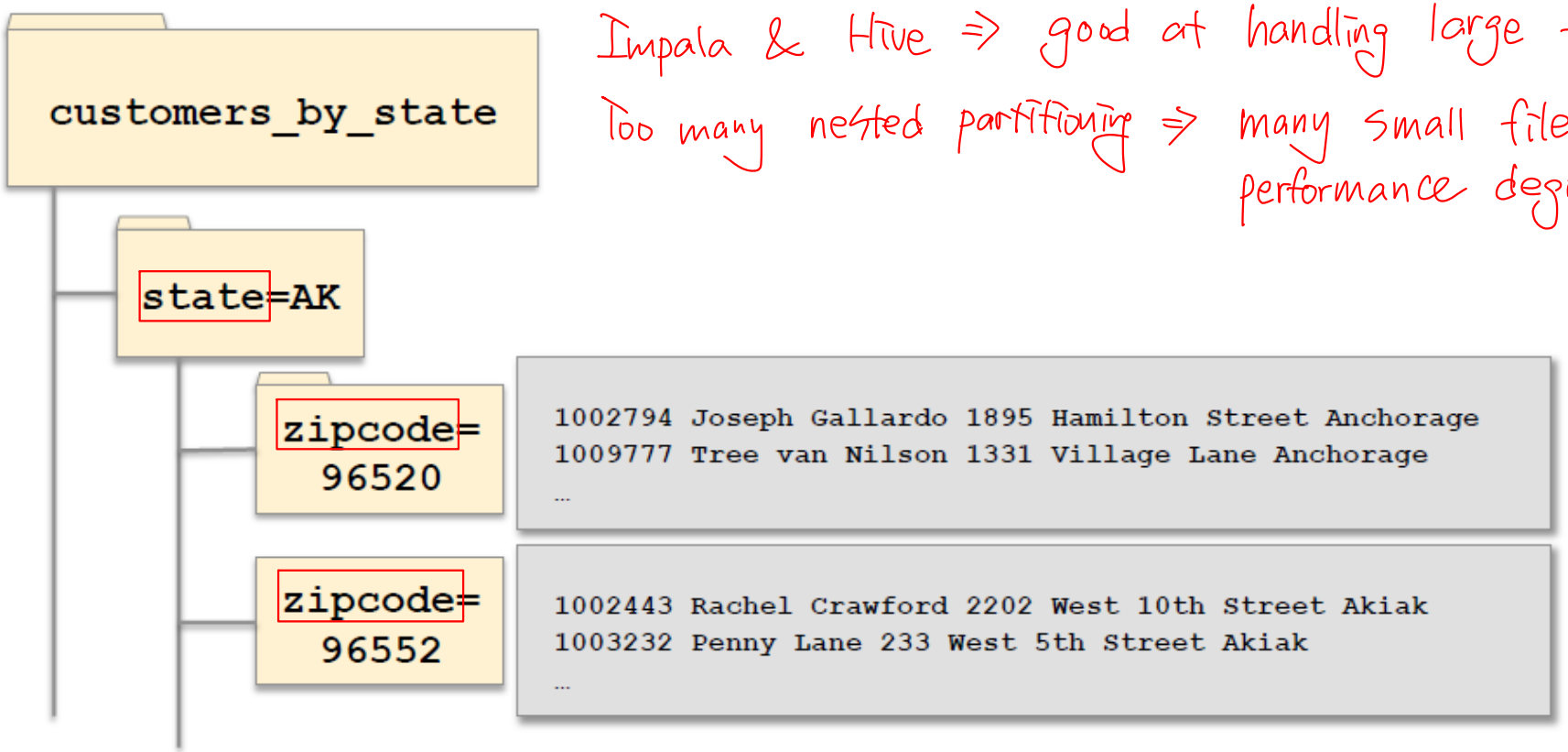
```
DESCRIBE accounts_by_state;
+---------+--------+---------+
| name    | type   | comment |
+---------+--------+---------+
| cust_id | int    |         |
| fname   | string |         |
| lname   | string |         |
| address | string |         |
| city    | string |         |
| zipcode | string |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

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

Nested Partitions

- You can also created **nested partitions** ··· no more than 2 level due to performance

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



Impala & Hive ⇒ good at handling large files

Too many nested partitioning ⇒ many small files
performance degraded

Loading Data Into a Partitioned Table

2 options depending on source & nature

- **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

Practice – Dynamic Partitioning

1. Import the **accounts** table from MySQL directly into the Hive
   - Hint1: use Sqoop command
   - Hint2: use the option “--hive-import”
2. Create a partitioned table by the **city** from **accounts** table
3. Insert data into the partitioned table from **accounts**
4. Create a partitioned table **accounts_areacode** by the **areacode** from **accounts** table
   - areacode could be defined by the first three digits of the phone number
5. Insert data into the partitioned table **accounts_areacode** from **accounts**
   - areacode could be obtained by substr(phone_number, 1, 3)
6. Create a nested partitioned table **accounts_nested** by (state, areacode) from accounts table
7. Insert data into the nested partitioned table **accounts_nested** from **accounts**

```
sqoop import --connect jdbc:mysql://localhost/loudacre --username training
--password training --fields-terminated-by '\t' --table accounts --hive-
import --warehouse-dir=/user/hive/warehouse

create external table accounts_by_areacode (fname string, lname string,
address string, city string, zipcode string) partitioned by (areacode
string) row format delimited fields terminated by ',' location '/loudacre/
accounts_by_areacode';

insert overwrite table accounts_by_areacode partition (areacode) select
first_name, last_name, address, city, zipcode, substr(phone_number, 1, 3)
as areacode from accounts;

create external table accounts_by_nested (fname string, lname string,
address string, city string, zipcode string) partitioned by (state string,
areacode string) row format delimited fields terminated by ',' location '/
loudacre/accounts_by_nested';

insert overwrite table accounts_by_nested partition (state, areacode)
select first_name, last_name, address, city, zipcode, state,
substr(phone_number, 1, 3) as areacode from accounts;
```

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');
```

Practice – Static Partitioning

1. **Create the following partitioned table**

```
create external table temp(id int, name string, sal int)
partitioned by(city string)
location '/loudacre/test';
```

2. **Make static partitions for city='hyd' and city='sec' using alter table**

3. **Make two sample text files that contains the following contents in HDFS**

[test1.txt]

1 ravi 100 hyd

2 krishna 200 hyd

[test2.txt]

3 fff 300 sec

```
create external table temp (id int, name string, sal int) partitioned by (city
string) location '/loudacre/test';

alter table temp add partition (city='hyd');

alter table temp add partition (city='sec');

load data inpath '/loudacre/test1.txt' into table temp partition (city='hyd');

load data inpath '/loudacre/test2.txt' into table temp partition (city='sec');
```

4. **Load two sample files to the table for the created static partitions**