CARLSON SCHOOL OF MANAGEMENT
UNIVERSITY OF MINNESOTA

Relational Data Analysis With Hive

MSBA 6330 Prof Liu

Slides credits: Cloudera Academic Partners Program

Relational Data Analysis With Hive

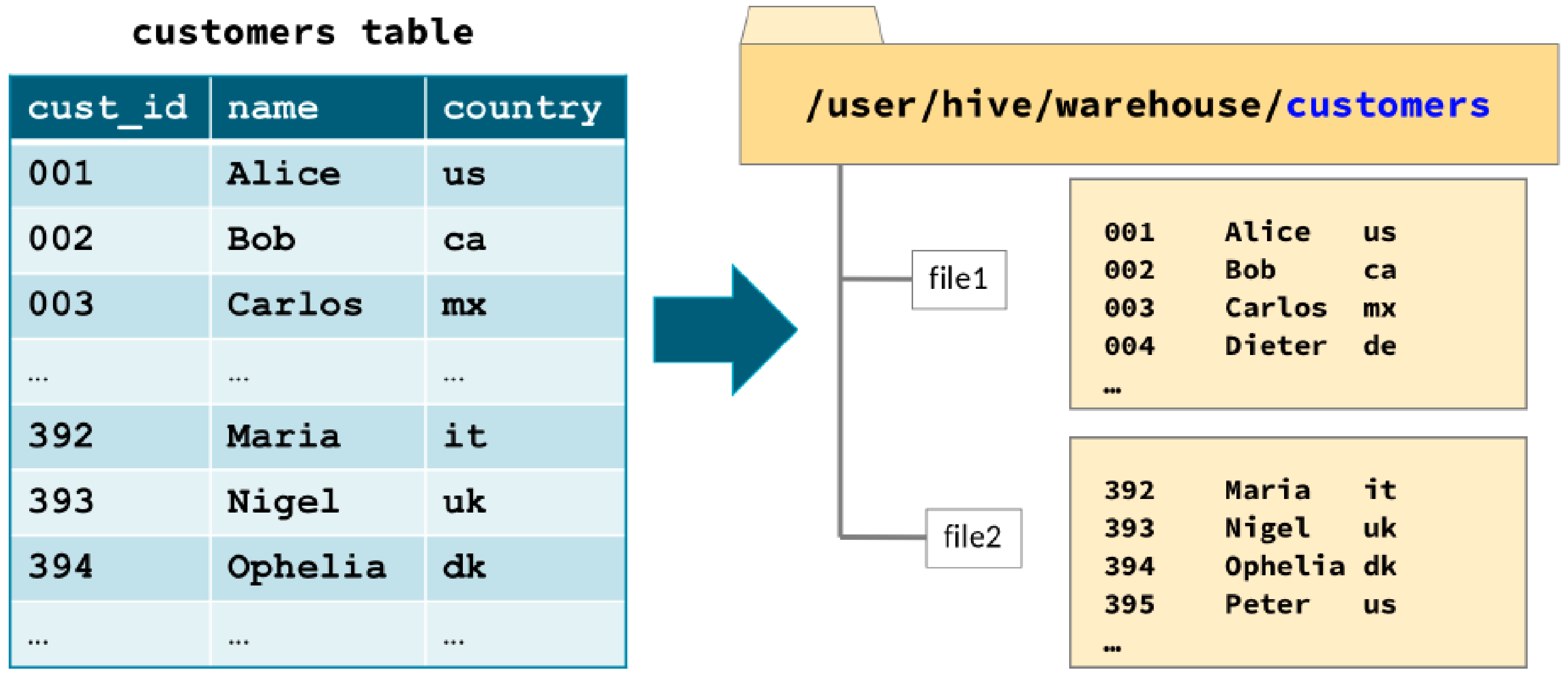
- In this chapter, you will learn
  - How to explore databases and tables in Hive
  - How HiveQL syntax compares to SQL
  - Which data types Hive supports
  - Which types of join operations Hive supports and how to use them

Relational Data Analysis With Hive

HIVE DATABASES AND TABLES

Hive Tables

- By default, Hive stores data for managed tables in the HDFS directory `/user/hive/warehouse`
  - Each table's data is stored in a subdirectory named after the table
  - A table's directory may contain multiple files
  - External tables can be stored else where (including in the cloud).



cust_id	name	country
001	Alice	us
002	Bob	ca
003	Carlos	mx
…	…	…
392	Maria	it
393	Nigel	uk
394	Ophelia	dk
…	…	…

```
001     Alice    us
002     Bob      ca
003     Carlos   mx
004     Dieter   de
…
```

```
392     Maria    it
393     Nigel    uk
394     Ophelia  dk
395     Peter    us
…
```

Hive Databases

- Each Hive table belongs to a specific database
  - If you don't specify a database, the table belongs to Hive's `default` database (not recommended, especially for large organizations)

- Please note that many small tables or lots of small partitions lead to small files in HDFS, which is not optimal.

Exploring Hive Databases And Tables (1 Of 2)

- See which databases are available with the SHOW DATABASES command

```
SHOW DATABASES;
accounting
default
```

- Switch between databases with the USE command

```
SELECT * FROM customers;       -- customers in the default database
USE dualcore;                            -- Switch databases.
SELECT * FROM customers;       -- customers in dualcore
SELECt * FROM sales.customers; -- customers in dualcore
```

- See which tables the current database contains with the SHOW TABLES command

```
USE dualcore;
SHOW TABLES;
customers
Employees …
```

All Hive keywords are case-insensitive, including the names of Hive operators and functions.

Exploring Hive Databases And Tables (2 Of 2)

- See the basic structure for a table with the `DESCRIBE` command

```
> DESCRIBE orders;    -- Provide the fully qualified name
order_id
int cust_id
int order_date
timestamp
```

- `DESCRIBE FORMATTED` provides even more detailed information for those with advanced requirements

```
> DESCRIBE FORMATTED orders;
# col_name data_type            comment
order_id        int                        None
cust_id         int                        None
order_date timestamp            None
# Detailed Table Information ...
```

Relational Data Analysis With Hive

HIVEQL SYNTAX

An Introduction To HiveQL

- HiveQL is Hive's query language
  - Based on a subset of SQL-92, plus Hive-specific extensions
- Some limitations compared to 'standard' SQL
  - Some features are not supported
    - e.g. Updating or deleting individual records (not available before Hive v0.14)
  - Others are only partially implemented
    - Include joins on non-equality conditions

```
JOIN ON customers.id = orders.id     -- This is supported
JOIN ON customers.id > orders.id     -- This is not supported
JOIN ON customers.id <> orders.id    -- Nor is this
```

SQL support

- Semantics: Similar to MySQL
  - Select
  - Group by: Hive requires the group-by field to be among the selected fields.
  - Limit
  - Order by: Hive requires the order-by field to be among the selected fields
  - Where
  - UNION [ALL]
- Windowing /analytics functions (0.11+):
  - lead/lag/first_value/last_value
  - over/window/partition by/cube/rollup
  - rank(),row_number(),dense_rank(),cume_dist(), percent_rank(),ntile()

Hive functions

- Many functions are similar to MySQL (complete list)
  - keyword/function/identifier names are not case sensitive.

```
cast(<expr> as <type>):cast('1' as int)
length(s)
concat(s1,s2,s3,...)
concat_ws(separator,s1,s2,s3,...)
substr(s, start, length)
upper(s)/ucase(s),
trim(s), ltrim(s)
regexp_replace(s, regex, replacement)
repeat(s,n)
split(s, pattern)
instr(str, substr)
```

```
rlike(regex) – regular expression like.
to_date(s)
year(d)
month(d)
day(d)
from_unixtime(i)
size(Map or Array)
rand()
round(d)
floor(d)
ceil(d)
```

- But, string comparisons are case-sensitive

```
SELECT * FROM customers WHERE state
     IN ('CA', 'OR', 'WA', 'NV', 'AZ');
```

Subqueries In Hive

- It supports subqueries in the FROM and WHERE clauses
  - `SELECT ... FROM (subquery) name ...`
  - `SELECT ... FROM … WHERE EXISTS (subquery)`
  - `SELECT ... FROM … WHERE x  IN (subquery)`

```
SELECT prod_id, brand, name
      FROM (SELECT *
            FROM products
            WHERE (price - cost) / price > 0.65
            ORDER BY price DESC
            LIMIT 10) high_profits  -- Mandatory subquery name
      WHERE price > 1000            -- Price in cents
      ORDER BY brand, name;
```

- Support for correlated subqueries is limited.
  - E.g. cannot be used in aggregations or conditional statements.

Relational Data Analysis With Hive

DATA TYPES

Hive Data Types

- Hive supports more than a dozen types
  - Most are similar to ones found in relational databases
  - Hive also supports three complex types
- Use the DESCRIBE command to see a table's column types

```
> DESCRIBE products;
prod_id           int
brand             string
name              string
price             int
cost              int
shipping_wt       int
```

Hive Integer Types

- Integer types are appropriate for whole (signed) numbers
  - Both positive and negative values allowed

Name	Size	Range	Example
TINYINT	1 Byte	-128 - 127	17
SMALLINT	2 Bytes	-32,768 - 32,767	5842
INT	4 Bytes	-2,147,483,648 - 2,147,483,647	84127213
BIGINT	8 Bytes	~-9.2 quintillion - ~ 9.2 quintillion	632197432180964

  - The default type for literal values is INT
- Best Practice:
  - Use the smallest type capable of doing the job

Hive Decimal Types

- Float/double for floating point numbers
  - Caution: Avoid using when exact values are required!
    - So a float value entered as 3.1 might actually be stored as 3.100000000000012
- Decimal for precise decimal numbers (e.g. money)

Name	Description	Example
FLOAT	Decimals	3.14159
DOUBLE	Very precise decimals	3.14159265358979323846
DECIMAL(p,s)	Controls scale/precision of a number	100.45 (p=5, s=2)

Other Simple (Scalar) Types In Hive

- Hive can also store several other types of information

Name	Description	Example
STRING	Character sequence	Betty F. Smith
CHAR(n)	Fixed-length character sequence	Hive␣␣(n=6)
VARCHAR(n)	Variable length character sequence	Hive (n=10)
BOOLEAN	True or False	TRUE
TIMESTAMP	Instant in time (UTC)	2013-06-14 16:51:05
BINARY	Raw bytes (Like VARBINARY in SQL)	N/A

Complex column types in Hive

- Hive also has a few complex data types
  - These are capable of holding multiple values

Name	Description & how to Define	Stored Data (suppose $ is the collection item delimitator)	Access members
ARRAY	Ordered list of values, all of the same type, e.g. `departments array<string>`	finance$marketing$hr	`departments[0]`
MAP	Key/value pairs, each of the same type e.g. `prices map<string,int>`	shoe#50$shirt#75	`prices['shirt']`
STRUCT	Named fields, of possibly mixed types e.g. `addr struct<city:string,state:string,zip:int>`	Minneapolis$MN$55455	`addr.city`

- Complex data types violate the "normal form", but offer fast data access
  - They are often desirable in Hadoop/Hive because they eliminate the need for big joins

Relational Data Analysis With Hive

JOINING DATASETS

Joins In Hive

- Hive supports several types of joins
  - Inner joins
  - Outer joins (Left, Right, and Full)
  - CROSS joins (supported in Hive 0.10 and later)
  - Left semi joins
- Only equality conditions are allowed in joins (equi-joins)
  - Valid: `customers.cust_id = orders.cust_id`
  - Invalid: `customers.cust_id <> orders.cust_id`
- For best performance, **list the largest table last in your query**
  - `Small_table JOIN big_table`

Join Syntax

- Hive requires the following syntax for joins

```
SELECT c.cust_id, name, total
     FROM customers c
     JOIN orders o ON (c.cust_id = o.cust_id);
```

- The above example is an inner join (the word "inner" is not required) which emits records only when the join key is found in both tables
- Implicit inner join syntax is not supported in Hive

```
SELECT c.cust_id, name, total
     FROM customers c, orders o
     WHERE (c.cust_id = o.cust_id);
```

X

Left Outer Join Example

- "OUTER" is required for outer joins in Hive
  - Customers Table

cust_id	name	country
a	Alice	us
b	Bob	ca
c	Carlos	mx
d	Dieter	dw

  - Orders Table

order_id	cust_id	total
1	a	1539
2	c	1871
3	a	6532
4	b	1456
5	z	2137

```
SELECT c.cust_id, name, total
      FROM customers c
      LEFT OUTER JOIN orders o
      ON (c.cust_id = o.cust_id);
```

cust_id	name	total
a	Alice	1539
a	Alice	6352
b	Bob	1456
c	Carlos	1871
d	Dieter	NULL

Full Outer Join Example

- Customers Table

cust_id	name	country
a	Alice	us
b	Bob	ca
c	Carlos	mx
d	Dieter	dw

- Orders Table

order_id	cust_id	total
1	a	1539
2	c	1871
3	a	6532
4	b	1456
5	z	2137

- Code

```
SELECT c.cust_id, name, total
     FROM customers c
     FULL OUTER JOIN orders o
     ON (c.cust_id = o.cust_id);
```

- Result

cust_id	name	total
a	Alice	1539
a	Alice	6352
b	Bob	1456
c	Carlos	1871
d	Dieter	NULL
NULL	NULL	2137

Essential Points

- Every Hive table belongs to exactly one database
  - The SHOW DATABASES command lists databases
  - The USE command switches the active database
  - The SHOW TABLES command lists all tables in a database
- Every column in a Hive table has an associated data type
  - Most simple column types are similar to SQL
  - Hive also supports a few complex types
- HiveQL syntax is familiar to those who know SQL
  - A subset of SQL-92, plus Hive-specific extensions
  - Supports inner, outer, and Left semi joins
  - Many SQL functions are built into Hive