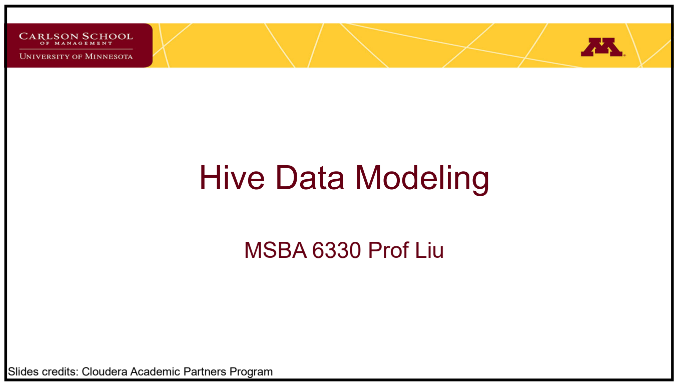
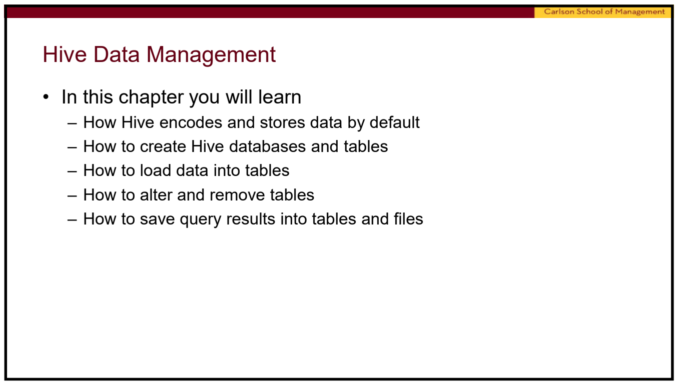


CARLSON SCHOOL OF MANAGEMENT
UNIVERSITY OF MINNESOTA

Hive Data Modeling

MSBA 6330 Prof Liu

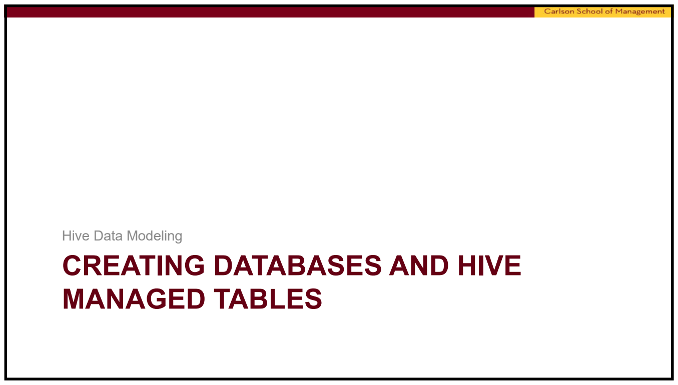
Slides credits: Cloudera Academic Partners Program



Carlson School of Management

Hive Data Management

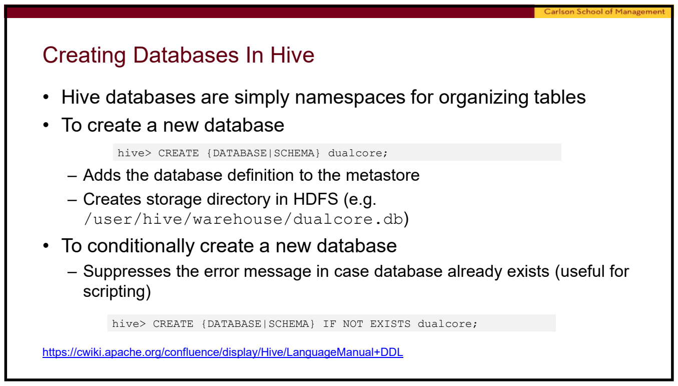
- In this chapter you will learn
  - How Hive encodes and stores data by default
  - How to create Hive databases and tables
  - How to load data into tables
  - How to alter and remove tables
  - How to save query results into tables and files



Carlson School of Management

Hive Data Modeling

CREATING DATABASES AND HIVE MANAGED TABLES


Carlson School of Management

Creating Databases In Hive

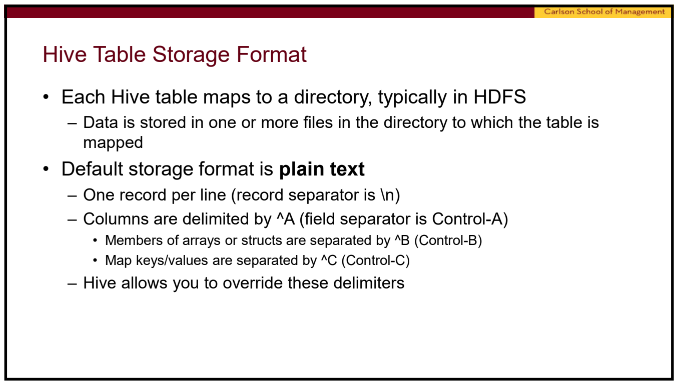
- Hive databases are simply namespaces for organizing tables
- To create a new database

```
hive> CREATE {DATABASE|SCHEMA} dualcore;
```

  - Adds the database definition to the metastore
  - Creates storage directory in HDFS (e.g. `/user/hive/warehouse/dualcore.db`)
- To conditionally create a new database
  - Suppresses the error message in case database already exists (useful for scripting)

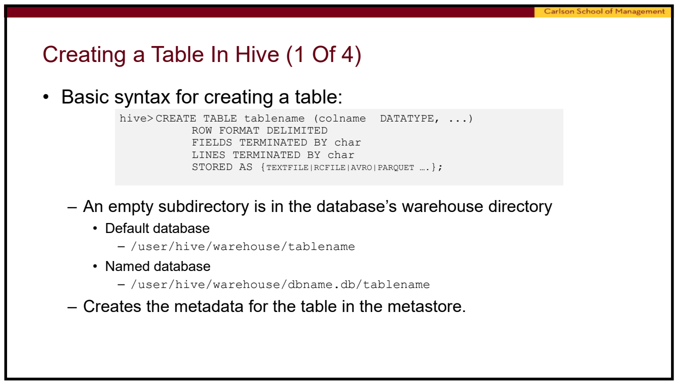
```
hive> CREATE {DATABASE|SCHEMA} IF NOT EXISTS dualcore;
```

https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL


Carlson School of Management

Hive Table Storage Format

- Each Hive table maps to a directory, typically in HDFS
  - Data is stored in one or more files in the directory to which the table is mapped
- Default storage format is **plain text**
  - One record per line (record separator is \n)
  - Columns are delimited by ^A (field separator is Control-A)
    - Members of arrays or structs are separated by ^B (Control-B)
    - Map keys/values are separated by ^C (Control-C)
  - Hive allows you to override these delimiters


Carlson School of Management

Creating a Table In Hive (1 Of 4)

- Basic syntax for creating a table:

```
hive>CREATE TABLE tablename (colname  DATATYPE, ...)
          ROW FORMAT DELIMITED
          FIELDS TERMINATED BY char
          LINES TERMINATED BY char
          STORED AS {TEXTFILE|RCFILE|AVRO|PARQUET ….};
```

  - An empty subdirectory is in the database's warehouse directory
    - Default database
      - `/user/hive/warehouse/tablename`
    - Named database
      - `/user/hive/warehouse/dbname.db/tablename`
  - Creates the metadata for the table in the metastore.

Creating A Table In Hive (2 Of 4)

- First, specify a name for the table, and column names and data types (later)
  - This is the only part of the syntax considered mandatory

```
CREATE TABLE tablename (colname  DATATYPE, ...)
    ROW FORMAT DELIMITED
    FIELDS TERMINATED BY char
    STORED AS {TEXTFILE|RCFILE|AVRO|PARQUET …};
```

  - If the rest of the definition is omitted, the file will be assumed to be:
    - A TEXTFILE
    - Whose fields are delimited using the Control-A character

Creating A Table In Hive (3 Of 4)

- The ROW FORMAT line instructs how each record in the table should be parsed.
- ROW FORMAT can be DELIMITED
  - Hive’s default delimiter is Control-A, but you may specify an alternate delimiter (see below)

```
CREATE TABLE tablename (colname  DATATYPE, ...)
    ROW FORMAT DELIMITED
    FIELDS TERMINATED BY '\t'
    STORED AS {TEXTFILE|RCFILE|AVRO|PARQUET…};
```

- ROW FORMAT can also be SERDE

```
ROW FORMAT SERDE serde_name WITH PROPERTIES (property specification)
```

  - There are SerDe's for text file formats e.g. JSON, XML, log, CSV
- ROW FORMAT is often omitted when a file format other than TEXTFILE is used
  - In this case, ROW FORMAT is a special SERDE corresponding to the file format (e.g. PARQUET).

Creating A Table In Hive (4 Of 4)

- Finally, you may declare the file format
  - Again, TEXTFILE is the default and does not need to be specified explicitly
  - Other popular formats include PARQUET and ORC

```
hive>    CREATE TABLE tablename (colname  DATATYPE, ...)
             ROW FORMAT DELIMITED
             FIELDS TERMINATED BY char
             STORED AS {TEXTFILE|RCFILE|AVRO|PARQUET|ORC...};
```

Example Table Definition

- The following example creates a new table named jobs
  – Data is stored as text with four comma-separated fields per line in the default database

```
CREATE TABLE jobs
     (id INT, title STRING, salary INT, posted TIMESTAMP)
     ROW FORMAT DELIMITED
     FIELDS TERMINATED BY ',';
```

The assumed file format is TEXTFILE

  – Example of corresponding record for the table above

```
1,Data Analyst,100000,2013-06-21 15:52:03
```

Create Externally Managed Tables

- Using EXTERNAL when creating the table that will not be managed by Hive
  – Dropping an external table removes only its metadata, not the actual data

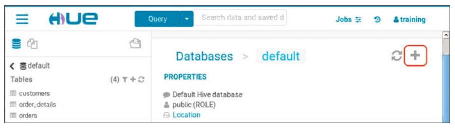
```
CREATE EXTERNAL TABLE adclicks (
     campaign_id  STRING,
     click_time TIMESTAMP,
     keyword STRING,
     site STRING,
     placement STRING,
     was_clicked  BOOLEAN,
     cost SMALLINT)
     LOCATION '/dualcore/ad_data';
```

The assumed file format is plain text with fields delimited by Ctrl+A

  – The location can be an Internet location.

Creating Tables Using the Table Browser

- **Hue's Table Browser provides a table creation wizard**
  – Supports most, but not all, table options



Hive Data Modeling

LOADING DATA INTO HIVE

Data Validation In Hive

- Hadoop and its ecosystem are '**schema on read**'
  - Unlike an RDBMS, Hive does not validate data on insert
    - Files are simply moved into place
  - Loading data into tables is therefore very fast
  - But errors in the file format will not be discovered until queries are performed
- Missing or invalid data in Hive will be represented as NULL

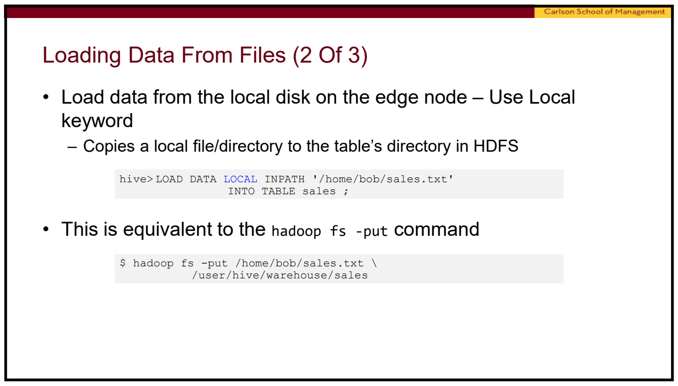
Loading Data From Files (1 Of 3)

- Load HDFS data into Hive using `hadoop fs -mv`
  - Can be done directly using `hadoop fs` commands
  - This example loads data from HDFS into Hive's sales table

```
$ hadoop fs -mv sales.txt /user/hive/warehouse/sales/
```

- Alternatively, use Hive's `LOAD DATA INPATH` command
  - Done from within the Hive shell (or a Hive script)
  - This moves data within HDFS, just like the command above
    - Destroying the original in the process
  - Source can be either a file or directory

```
hive> LOAD DATA INPATH 'sales.txt' INTO TABLE sales;
```



Carlson School of Management

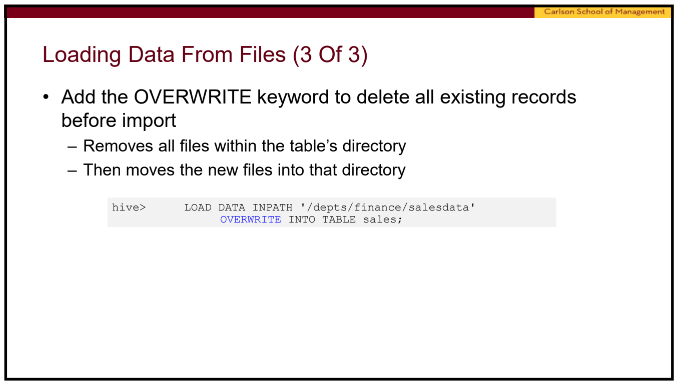
Loading Data From Files (2 Of 3)

- Load data from the local disk on the edge node – Use Local keyword
  - Copies a local file/directory to the table's directory in HDFS

```
hive> LOAD DATA LOCAL INPATH '/home/bob/sales.txt'
                INTO TABLE sales ;
```

- This is equivalent to the `hadoop fs -put` command

```
$ hadoop fs -put /home/bob/sales.txt \
           /user/hive/warehouse/sales
```

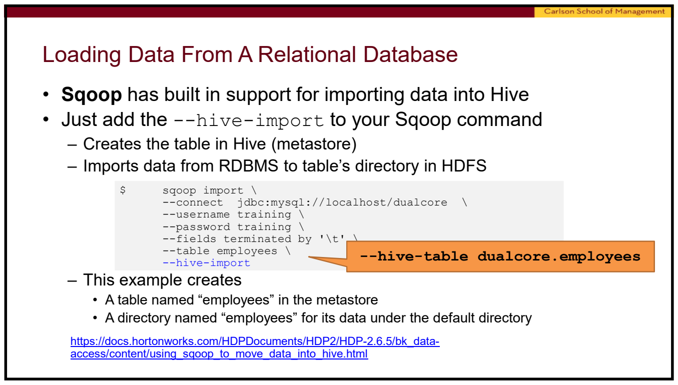


Carlson School of Management

Loading Data From Files (3 Of 3)

- Add the OVERWRITE keyword to delete all existing records before import
  - Removes all files within the table's directory
  - Then moves the new files into that directory

```
hive>     LOAD DATA INPATH '/depts/finance/salesdata'
               OVERWRITE INTO TABLE sales;
```



Carlson School of Management

Loading Data From A Relational Database

- **Sqoop** has built in support for importing data into Hive
- Just add the `--hive-import` to your Sqoop command
  - Creates the table in Hive (metastore)
  - Imports data from RDBMS to table's directory in HDFS

```
$      sqoop import \
       --connect  jdbc:mysql://localhost/dualcore  \
       --username training \
       --password training \
       --fields terminated by '\t' \
       --table employees \
       --hive-import
```

--hive-table dualcore.employees

  - This example creates
    - A table named "employees" in the metastore
    - A directory named "employees" for its data under the default directory

https://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.6.5/bk_data-access/content/using_sqoop_to_move_data_into_hive.html

Carlson School of Management

Hive Data Modeling

ALTERING DATABASES AND TABLES

Carlson School of Management

Removing A Database

- Removing a database is similar to creating it
  - Just replace the CREATE keyword with DROP

```
hive> DROP DATABASE dualcore;
hive> DROP DATABASE IF EXISTS dualcore;
```

- These commands will fail if the database contains any tables
  - Add the CASCADE keyword to force removal
  - Caution: This command removes data and tables in HDFS!

```
hive> DROP DATABASE dualcore CASCADE;
```

Carlson School of Management

Removing A Table

- Syntax is similar to database removal

```
hive> DROP TABLE customers;
hive> DROP TABLE IF EXISTS customers;
```

- Caution: If the table is a Hive Managed table, these commands will remove data in HDFS!
  - Hive does not have a rollback or undo feature
  - Dropping an External table only removes the metadata, but not actual table files.

Hive Data Modeling

STORING QUERY RESULTS

Saving Query Output To A Table

- Use `INSERT OVERWRITE TABLE` to send the results of a SELECT statement to a Hive table instead
  - Destination table (with the same schema) must already exist
  - Existing contents will be deleted

```
hive> INSERT OVERWRITE TABLE ny_customers
                SELECT * FROM customers
                WHERE state = 'NY';
```

```
hive> INSERT INTO TABLE ny_customers
                SELECT * FROM customers
                WHERE state = 'NJ' OR state = 'CT';
```

- INSERT INTO TABLE adds records without first deleting the existing data
  - In other words, appends rather than overwrites

Creating Tables Based On Existing Data

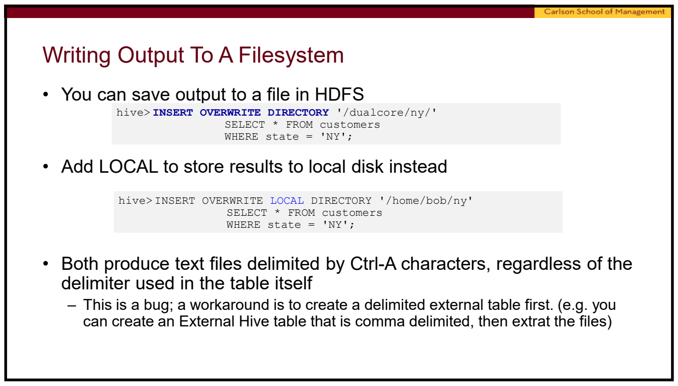
- Hive supports creating a table based on a SELECT statement

```
CREATE TABLE ny_customers AS
    SELECT cust_id, fname, lname FROM customers
    WHERE state = 'NY';
```

  - Column definitions and names are derived from the existing table
  - Use aliases in the SELECT statement to specify new names
- You can also create a new table using the definition of an existing table

```
CREATE TABLE jobs_archived LIKE jobs;
```

  - Column definitions and names are derived form the existing table
  - New table will contain no data



Writing Output To A Filesystem

- You can save output to a file in HDFS

```
hive> INSERT OVERWRITE DIRECTORY '/dualcore/ny/'
                SELECT * FROM customers
                WHERE state = 'NY';
```

- Add LOCAL to store results to local disk instead

```
hive> INSERT OVERWRITE LOCAL DIRECTORY '/home/bob/ny'
                SELECT * FROM customers
                WHERE state = 'NY';
```

- Both produce text files delimited by Ctrl-A characters, regardless of the delimiter used in the table itself
  - This is a bug; a workaround is to create a delimited external table first. (e.g. you can create an External Hive table that is comma delimited, then extrat the files)