



Petabyte Scale Data Warehousing Greenplum

Data Loading

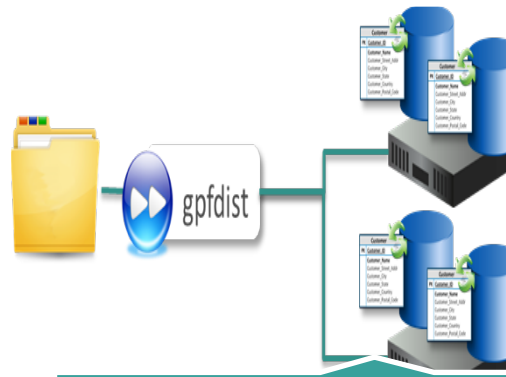
Postgres Conf 2018

Marshall Presser
Craig Sylvester
Andreas Scherbaum
17 April 2018

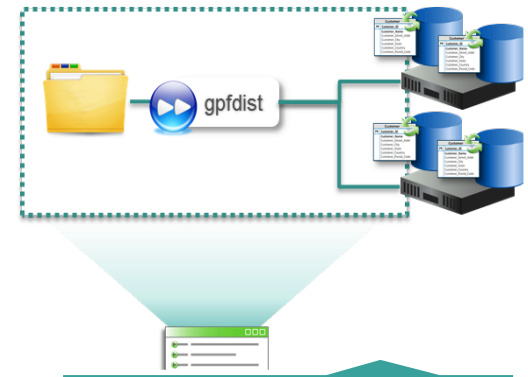
Data Loading Methods



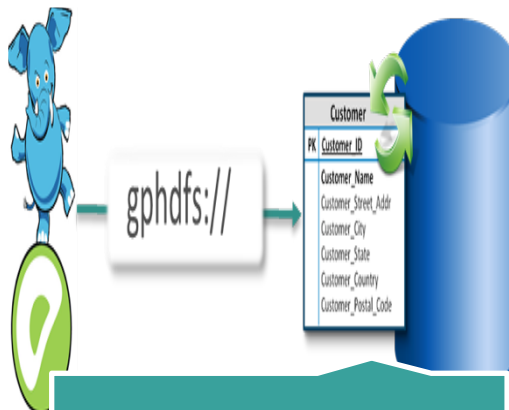
External tables



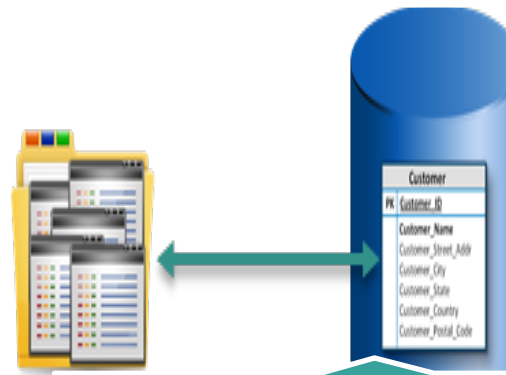
gpfdist



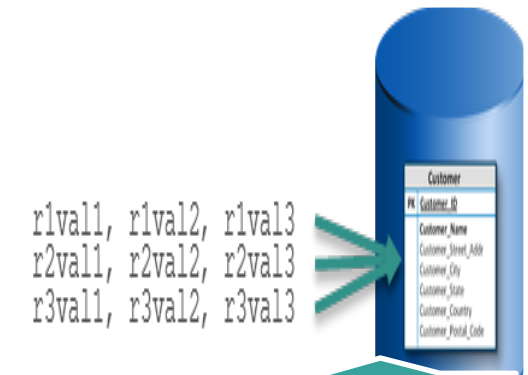
gpload



gphdfs



SQL COPY



SQL INSERT

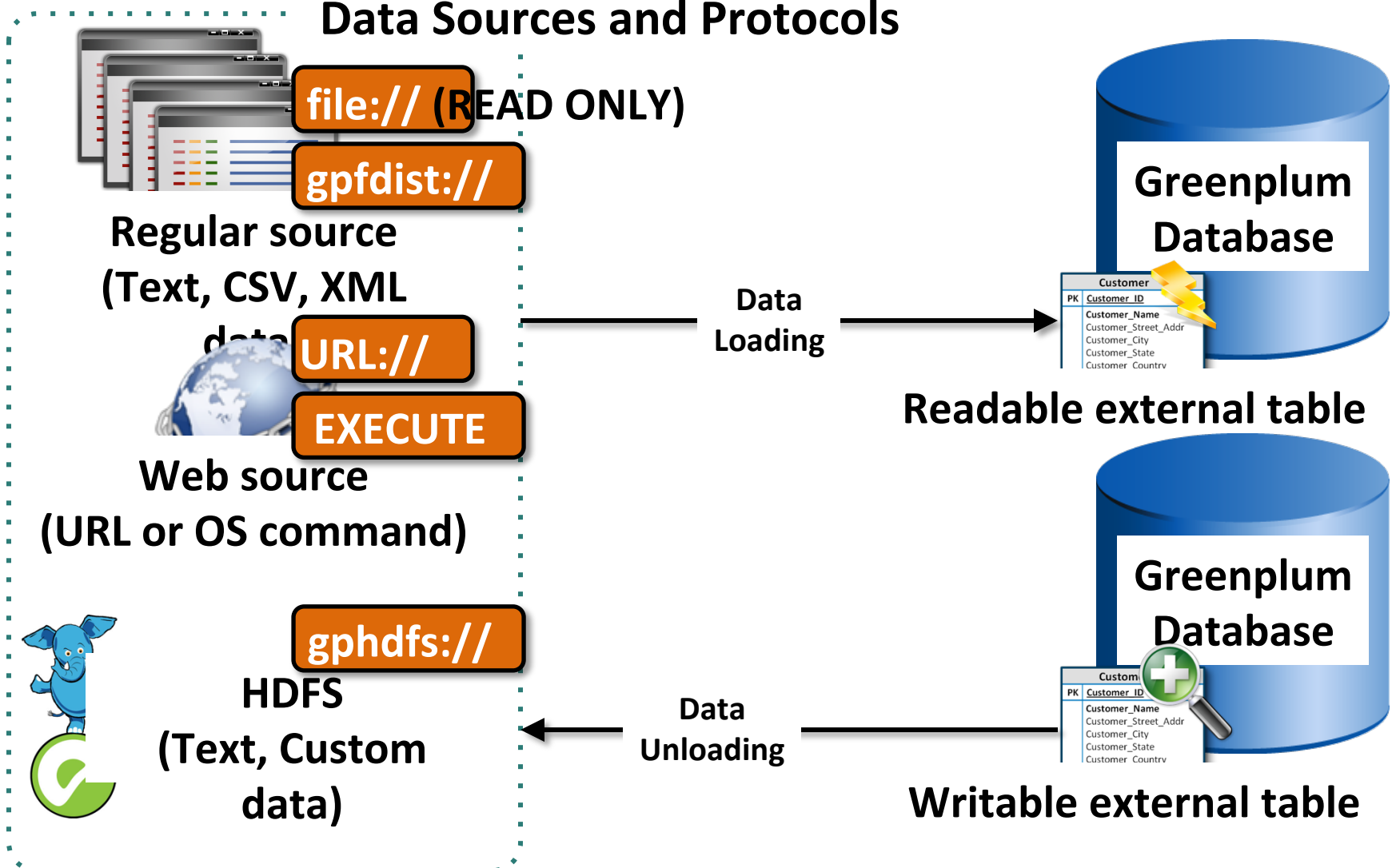
Loading with External Tables

Read-only external tables:

- Leverage parallel processing power of segments
- Can be accessed with `SELECT` statements
- Access data outside of the Greenplum Database
- Commonly used for ETL and data loading

External Table Types

Data Sources and Protocols



File-Based External Tables

When creating file-based external tables:

- Specify up to as many URIs as you have segments in the `LOCATION` clause
- Each URI points to an external data file or data source
- URIs do not need to exist prior to defining the external table
- The URI must exist when the data is queried

File-Based External Table Protocol and Format



Example: Create an external table with multiple URIs

```
CREATE EXTERNAL TABLE ext_expenses (name text, date date,  
amount float4, category text, description text)  
LOCATION (  
'file://seghost1/dbfast/external/expenses1.csv',  
'file://seghost1/dbfast/external/expenses2.csv',  
'file://seghost2/dbfast/external/expenses3.csv',  
'file://seghost2/dbfast/external/expenses4.csv',  
'file://seghost3/dbfast/external/expenses5.csv',  
'file://seghost3/dbfast/external/expenses6.csv',  
)  
FORMAT 'CSV' ( HEADER );
```

Protocol can be file,
gpfdist, gpfdists,
or gphdfs

Format can be
CSV, TEXT, XML,
or custom

You can define as
many URIs as you
have segments

Parallel File Distribution Program

The parallel file distribution program, `gpfdist`:

- Is a C program that uses HTTP
- Can be run on an external server
- Distributes data at 200 MB/s per `gpfdist`
- Provides full parallelism for best performance

The data load utility, `gpload`:

- Interfaces with and invokes `gpfdist`
- Creates an external table definition
- Executes `INSERT`, `UPDATE`, or `MERGE` to load data

Parallel File Distribution Program Example

```
gpfdist -d /var/load_files/expenses1 -p 8081 >> gpfdist.log  
2>&1 &  
gpfdist -d /var/load_files/expenses2 -p 8082 >> gpfdist.log  
2>&1 &
```



Example: Creating an external table using the gpfdist protocol

```
CREATE EXTERNAL TABLE ext_expenses  
  (name text, date date, amount float4, description text)  
  LOCATION (  
    'gpfdist://etlhost:8081/*',  
    'gpfdist://etlhost:8082/*')  
  FORMAT 'TEXT' (DELIMITER '|')  
  ENCODING 'UTF-8'  
  LOG ERRORS INTO ext_expenses_loadererrors  
  SEGMENT REJECT LIMIT 10000 ROWS ;
```


Parallel File Distribution Program Example (Cont'd.)



Example: Load data into a regular table

```
INSERT INTO expenses (SELECT * FROM ext_expenses);
```

Accessing Hadoop Data Using gphdfs

```
2 DROP EXTERNAL TABLE IF EXISTS wiki_pages_ext;
3 CREATE EXTERNAL TABLE wiki_pages_ext
4 (
5     LIKE wiki_pages
6 )
7 LOCATION ('gphdfs://hadoop-w-0:8020/user/gpadmin/word_count/out/part-m-*.')
8 FORMAT 'TEXT' (DELIMITER E'\t' NULL E'')
9 ENCODING 'UTF8'
10 LOG ERRORS INTO wiki_err SEGMENT REJECT LIMIT 1 PERCENT;
11 GRANT SELECT ON wiki_pages_ext TO demo;
12 /*
13 demo=# INSERT INTO wiki_pages
14 demo=# SELECT * FROM wiki_pages_ext;
15 NOTICE: Found 66 data formatting errors (66 or more input rows). Rejected 1
16 INSERT 0 15347677
17 Time: 9674.557 ms
18 demo=# select count(*) from wiki_pages;
19      count
20 -----
21 15347677
22 (1 row)
23 */
```

Accessing Hadoop Data Using gphdfs (Cont'd.)

One time setup for using gphdfs requires setting a couple of configuration parameters:

`gp_hadoop_target_version` -- corresponds with the version of Hadoop you're using

`gp_hadoop_home` -- points to your Hadoop client installation directory

gpload YAML Control File Example



Example: YAML Control File for Loading Data

```
VERSION: 1.0.0.1
DATABASE: faa2
USER: gpadmin
HOST: smdw
PORT: 5432
GLOAD:
  INPUT:
    - SOURCE:
        LOCAL_HOSTNAME:
          - mdw
        PORT: 8081
        FILE:
          - /rawdata/FAADData/On_Time_On_Time_Performance_*.csv
    - FORMAT: csv
    - DELIMITER: ','
    - ESCAPE: '"'
    - QUOTE: '"'
    - HEADER: true
    - ERROR_LIMIT: 100
    - ERROR_TABLE: faadata.faadataerr
  OUTPUT:
    - TABLE: faadata.factontimeperformance
    - MODE: insert
  PRELOAD:
    - TRUNCATE: true
    - REUSE_TABLES: false
  SQL:
    - BEFORE: "INSERT INTO audit VALUES('start', current_timestamp)"
    - AFTER: "INSERT INTO audit VALUES('end', current_timestamp)"
```

Greenplum Database
connection information

gpload definition block

gpfdist
configuration
information

Pre-existing table
and data entry mode

Pre and post
SQL statements

Indentation
spaces, **not**
tabs,
determine
hierarchy

Preloading
data section

gpload Syntax

The gpload syntax is as follows:

```
gpload -f control_file [-l log_file] [-h  
hostname] [-p port] [-U username] [-d  
database] [-W] [-v | -V] [-q] [-D] gpload  
-? | --version
```

The following is an example of how it is used:



Example: Successful data load with gpload

```
$ gpload -f load_faadata.yaml  
2012-01-17 09:05:29|INFO|gpload session started 2012-01-17 09:05:29  
2012-01-17 09:05:29|INFO|started gpfdist -p 8081 -P 8082 -f "/rawdata/  
FAADOn_Time_Performance_*.csv" -t 30  
2012-01-17 09:11:23|INFO|running time: 353.36 seconds  
2012-01-17 09:11:23|INFO|rows Inserted = 20860045  
2012-01-17 09:11:23|INFO|rows Updated = 0  
2012-01-17 09:11:23|INFO|data formatting errors = 0  
2012-01-17 09:11:23|INFO|gpload succeeded  
$
```

Loading Data with gpload



Example: Load data with gpload

```
$ gpload -f load_faadata.yaml
```

gpadmin@mdw:~/course_samples

```
[gpadmin@mdw course_samples]$ gpload -f load_faadata.yaml
2012-01-19 06:10:57|INFO|gpload session started 2012-01-19 06:10:57
2012-01-19 06:10:57|INFO|started gpfdist -p 8081 -P 8085 -f "/rawdata/FAADData/On
_Time_On_Time_Performance_*.csv" -t 30
```

gpload starts gpfdist

Query Text

```
INSERT INTO "faadata"."factontimeperformance"
("year","quarterid","monthid","dayofmonth","dayid","flightdate","tailnum","flightnum","originairportid","origincityname","originstateabbreviation","originstatefipscode","originstatename","originwacid","destairportid","destcityname","deststateabbreviation","deststatefipscode","deststatename","destwacid","crsdeptime","deptime","depdelay","depdelay2","depdel15","depdelaygroupid","deptimeblkid","taxiout","wheelsoff","wheelson","taxiin","crsarrrtime","arrrtime","arrdelay","arrdelay2","arrdel15","arrdelaygroupid","arrtimeblkid","cancelled","cancellationid","diverted","crselapsedtime","actualelapsedtime","airtime","numflights","distance","distancegroupid","carrierdelay","weatherdelay","nasedelay","securitydelay","lateaircraftdelay","firstdeptime","totaladdgtime","longestaddgtime","divairportlandings","divertedreacheddest","divactualelapsedtime","divarrdelay","divdistance","div1airport","div1wheelson","div1totalgtime","div1longestgtime","div1wheelsoff","div1tailnum","div2airport","div2wheelson","div2totalgtime","div2longestgtime","div2wheelsoff","div2tailnum","div3airport","div3wheelson","div3totalgtime","div3longestgtime","div3wheelsoff","div3tailnum","div4airport","div4wheelson","div4totalgtime","div4longestgtime","div4wheelsoff","div4tailnum","div5airport","div5wheelson","div5totalgtime","div5longestgtime","div5wheelsoff","div5tailnum" FROM ext_gpload20120119_060258_15300
```

Data is loaded into target table

External Web Table Protocols and Format



Example: External WEB table with data loads from URLs

```
CREATE EXTERNAL WEB TABLE ext_expenses (name text, date
date, amount float4, description text)
    LOCATION ( 'http://intranet.company.com/expenses/
sales/expenses.csv', 'http://
intranet.company.com/expenses/finance/expenses.csv',
'http://intranet.company.com/expenses/ops/expenses.csv')
FORMAT 'CSV' ( HEADER );
```



Example: External WEB table with data loads from a script

```
CREATE EXTERNAL WEB TABLE log_output (linenum int, message
text)
EXECUTE '/var/load_scripts/get_log_data.sh'
ON HOST FORMAT 'TEXT' (DELIMITER '|');
```

External Web Table Protocols and Format (Cont'd.)



Example: External WEB table with data loads from a command

```
CREATE EXTERNAL WEB TABLE du_space (storage text)  
EXECUTE 'df -k' ON ALL FORMAT 'TEXT';
```

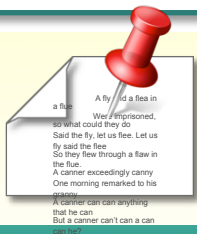

Environment Variables for Command-Based Web Tables

- Are not sourced at the segment host
- Can be set in the `EXECUTE` clause as follows:



Example: External WEB table with environment variables

```
CREATE EXTERNAL WEB TABLE TEST(segname text, abbrev text)
  EXECUTE 'export HNAME="$HOSTNAME"-SR; echo
"$HOSTNAME,$HNAME"' FORMAT 'TEXT' (DELIMITER ',');
SELECT * FROM test;
segname | abbrev
-----+-----
sdw2    | sdw2-SR
sdw1    | sdw1-SR
(2 rows)
```



Note: You can disable the use of the `EXECUTE` command in web table definitions by setting `gp_external_enable_exec` to `off`.

External Table Environment Variables

Variable	Description
<code>\$GP_CID</code>	Command count of the session executing the external table statement.
<code>\$GP_DATABASE</code>	The database that the external table definition resides in.
<code>\$GP_DATE</code>	The date the external table command was executed.
<code>\$GP_MASTER_HOST</code>	The host name of the Greenplum master host from which the external table statement was dispatched.
<code>\$GP_MASTER_PORT</code>	The port number of the Greenplum master instance from which the external table statement was dispatched.
<code>\$GP_SEG_DATADIR</code>	The location of the data directory of the segment instance executing the external table command.
<code>\$GP_SEG_PG_CONF</code>	The location of the <code>postgresql.conf</code> file of the segment instance executing the external command.

External Table Environment Variables (cont'd.)

Variable	Description
<code>\$GP_SEG_PORT</code>	The port number of the segment instance executing the external table command.
<code>\$GP_SEGMENT_COUNT</code>	The total number of primary segment instances in the Greenplum Database system.
<code>\$GP_SEGMENT_ID</code>	The ID number of the segment instance executing the external table command.
<code>\$GP_SESSION_ID</code>	The database session identifier number associated with the external table statement.
<code>\$GP_SN</code>	Serial number of the external table scan node in the query plan of the external table statement.
<code>\$GP_TIME</code>	The time the external table command was executed.
<code>\$GP_USER</code>	The database user executing the external table statement.
<code>\$GP_XID</code>	The transaction ID of the external table statement.

External Table Error Handling

When handling errors using external tables:

- Incorrectly formatted rows are rejected:
 - Rows with missing or extra attributes
 - Rows with columns of the wrong data type
 - Rows with invalid client encoding sequence
- CONSTRAINT errors (NOT NULL, CHECK, UNIQUE) are handled “all or nothing” -- not single row isolation
- Format of error handling clause:

```
[LOG ERRORS INTO error_table] SEGMENT  
REJECT LIMIT count [ROWS | PERCENT]
```

External Tables and Planner Statistics

Query planning of complex queries on external tables is not optimal because:

- Data resides outside the database
- No database statistics exist for external table data
- Data from external tables are not meant for frequent or ad-hoc access

COPY SQL Command

The COPY SQL command:

- Is a PostgreSQL command
- Loads all rows in one command and is not parallel
- Loads data from a file or from standard input
- Supports error handling similar to external tables

The following is an example of the command:



Example: Copy data from `/data/myfile.csv` into the table specified

```
COPY mytable FROM '/data/myfile.csv' WITH CSV HEADER;
```

Data Loading Performance Tips

- Drop indexes and recreate them after loading data
- Use gpfdist to load or unload data in Greenplum Database
- Spread the data evenly across as many ETL nodes as possible
- Split very large data files into equal parts and spread the data across as many file systems as possible
- Run two gpfdist instances per file system
- Run gpfdist on as many network interfaces as possible

Data Loading Performance Tips (Cont'd.)

- Use `gp_external_max_segs` to control the number of segments each `gpfdist` serves
- Always keep `gp_external_max_segs` and the number of `gpfdist` processes an even factor
- Always drop indexes before loading into existing tables and re-create the index after loading
- Always run `ANALYZE` on the table after loading it
- Disable automatic statistics collection during loading by setting `gp_autostats_mode` to `NONE`
- Run `VACUUM` after load errors to recover space