Pivotal

A NEW PLATFORM FOR A NEW ERA

GPDB Storage Considerations

Pivotal® Greenplum Database

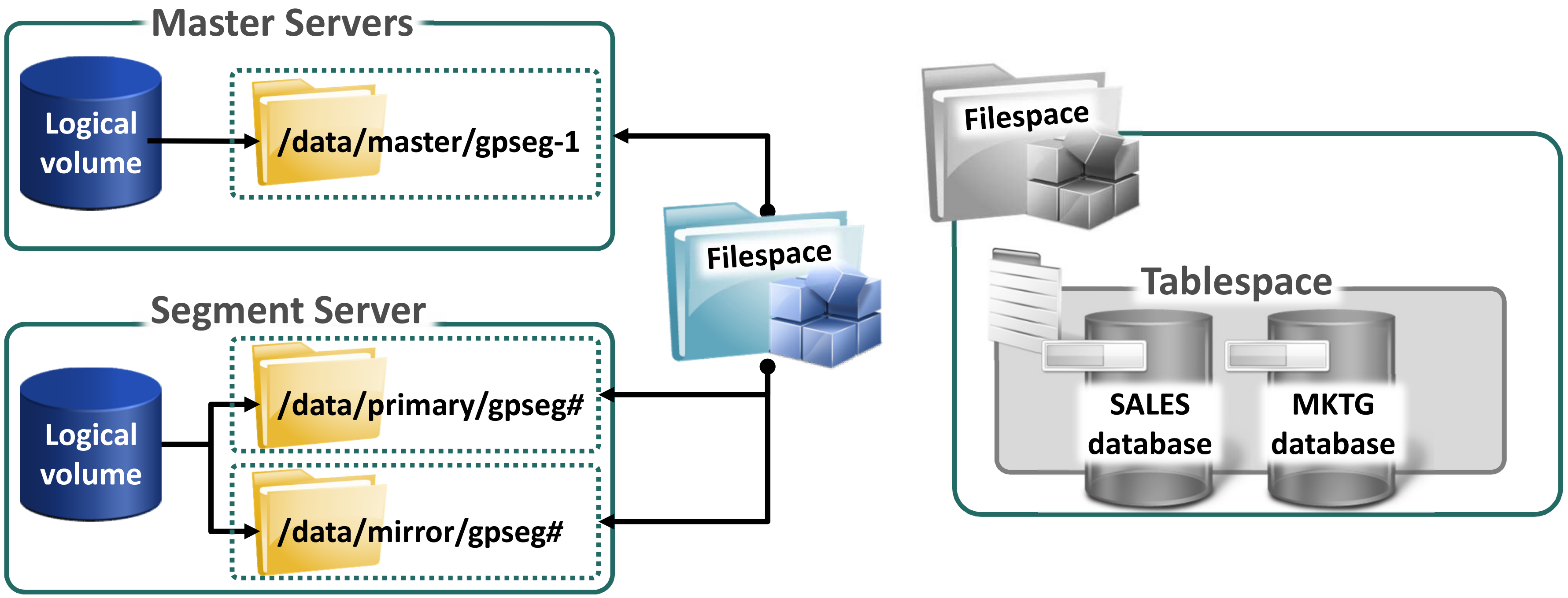
Pivotal™

Agenda

- Introduction
- Tablespaces and filespaces
- Additional table types (external, temp)
- Table storage models
- Compression options
- Test it out in the lab

© 2015 Pivotal Software, Inc. All rights reserved.

Filespaces

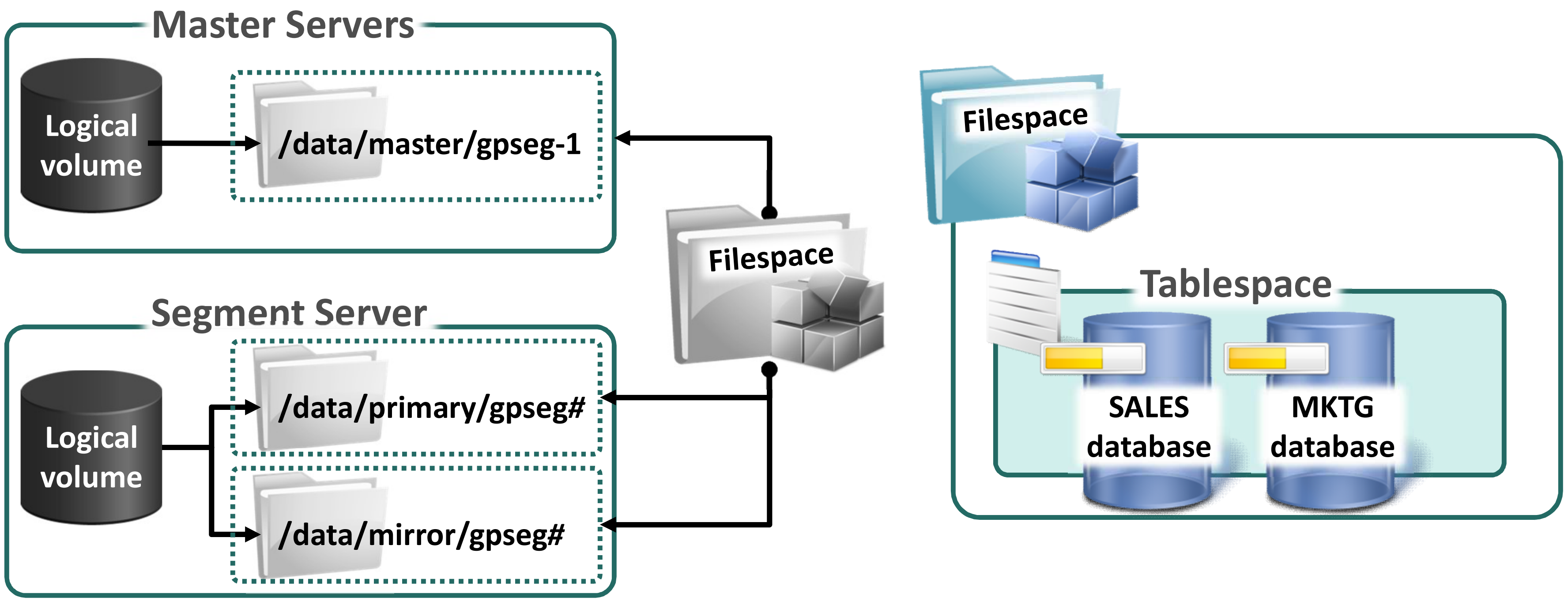


- By default, the system filespace, `pg_system`, is created on initialization
- All system relations are stored in the system filespace by default
- All user relations are also stored in the system filespace by default

Pivotal

© 2015 Pivotal Software, Inc. All rights reserved.

Tablespaces

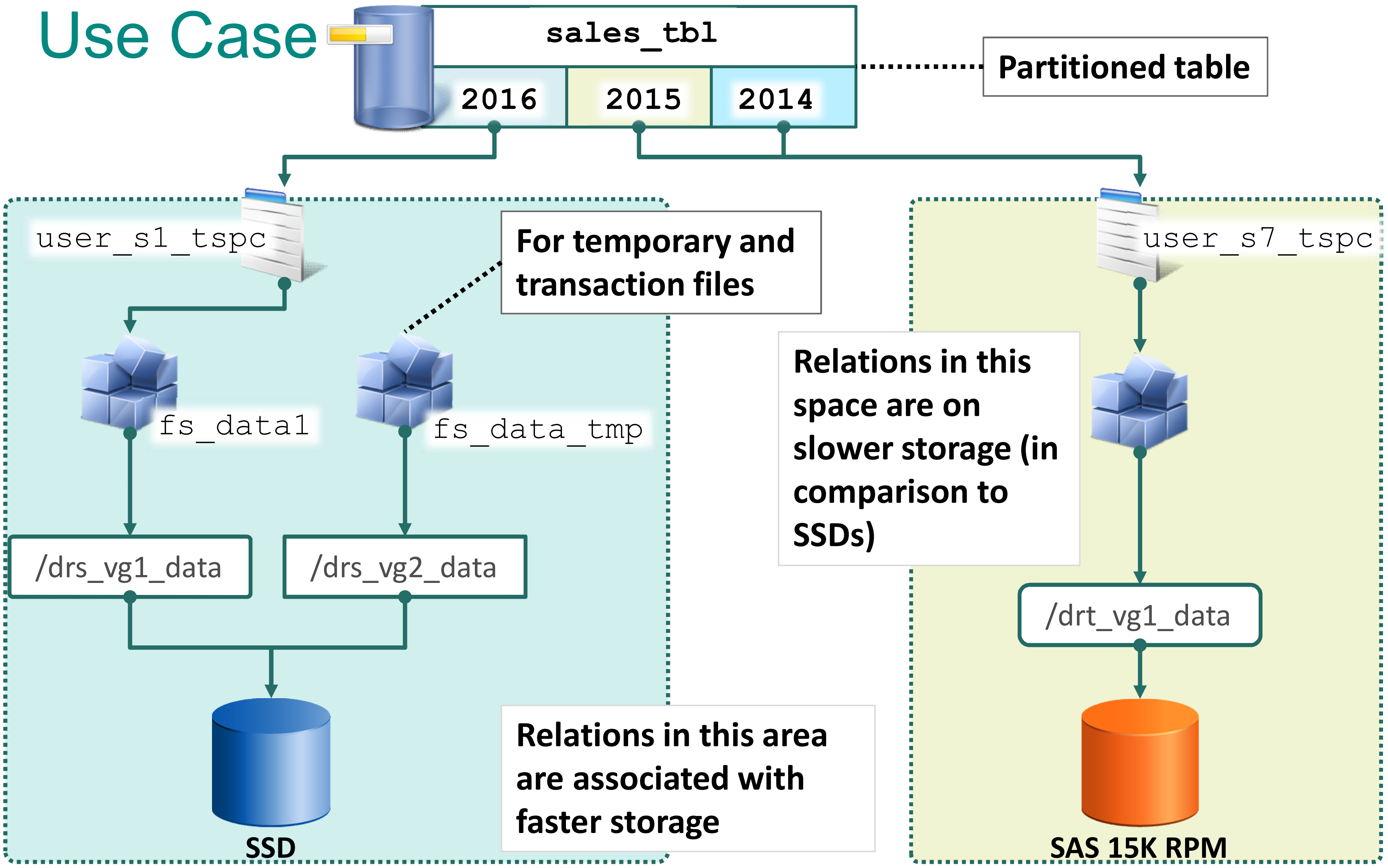


- Tablespaces *sit atop* filespaces interacting with the underlying filesystem
- A filespace can support multiple tablespaces
- Two tablespaces are created on initialization: `pg_default` and `pg_global`

Pivotal™

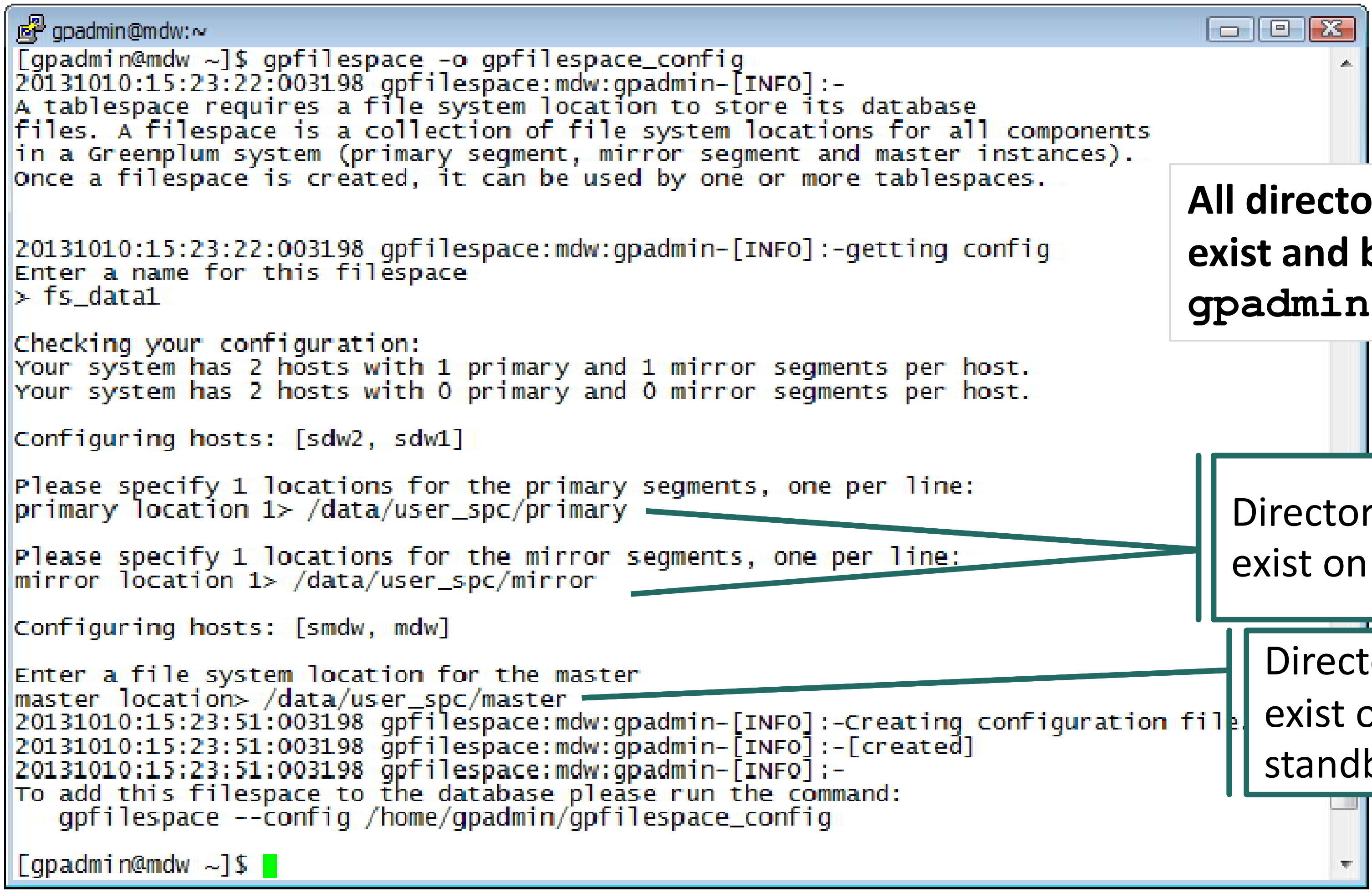
© 2015 Pivotal Software, Inc. All rights reserved.

Filespace and Tablespace Implementation – Use Case


sales_tbl
2016
2015
2014
Partitioned table
user_s1_tspc
For temporary and transaction files
user_s7_tspc
fs_data1
fs_data_tmp
Relations in this space are on slower storage (in comparison to SSDs)
/drs_vg1_data
/drs_vg2_data
/drt_vg1_data
Relations in this area are associated with faster storage
SSD
SAS 15K RPM

© 2015 Pivotal Software, Inc. All rights reserved.

Creating the Filespace Configuration File



```
[gpadmin@mdw ~]$ gpfilespace -o gpfilespace_config
20131010:15:23:22:003198 gpfilespace:mdw:gpadmin-[INFO]:-
A tablespace requires a file system location to store its database
files. A filespace is a collection of file system locations for all components
in a Greenplum system (primary segment, mirror segment and master instances).
Once a filespace is created, it can be used by one or more tablespaces.


20131010:15:23:22:003198 gpfilespace:mdw:gpadmin-[INFO]:-getting config
Enter a name for this filespace
> fs_data1

Checking your configuration:
Your system has 2 hosts with 1 primary and 1 mirror segments per host.
Your system has 2 hosts with 0 primary and 0 mirror segments per host.

Configuring hosts: [sdw2, sdw1]

Please specify 1 locations for the primary segments, one per line:
primary location 1> /data/user_spc/primary

Please specify 1 locations for the mirror segments, one per line:
mirror location 1> /data/user_spc/mirror

Configuring hosts: [smdw, mdw]

Enter a file system location for the master
master location> /data/user_spc/master
20131010:15:23:51:003198 gpfilespace:mdw:gpadmin-[INFO]:-Creating configuration file
20131010:15:23:51:003198 gpfilespace:mdw:gpadmin-[INFO]:-[created]
20131010:15:23:51:003198 gpfilespace:mdw:gpadmin-[INFO]:-
To add this filespace to the database please run the command:
   gpfilespace --config /home/gpadmin/gpfilespace_config

[gpadmin@mdw ~]$
```

All directories must exist and be owned by `gpadmin`

Directories must already exist on segment hosts

Directory must already exist on master and standby master hosts

Pivotal

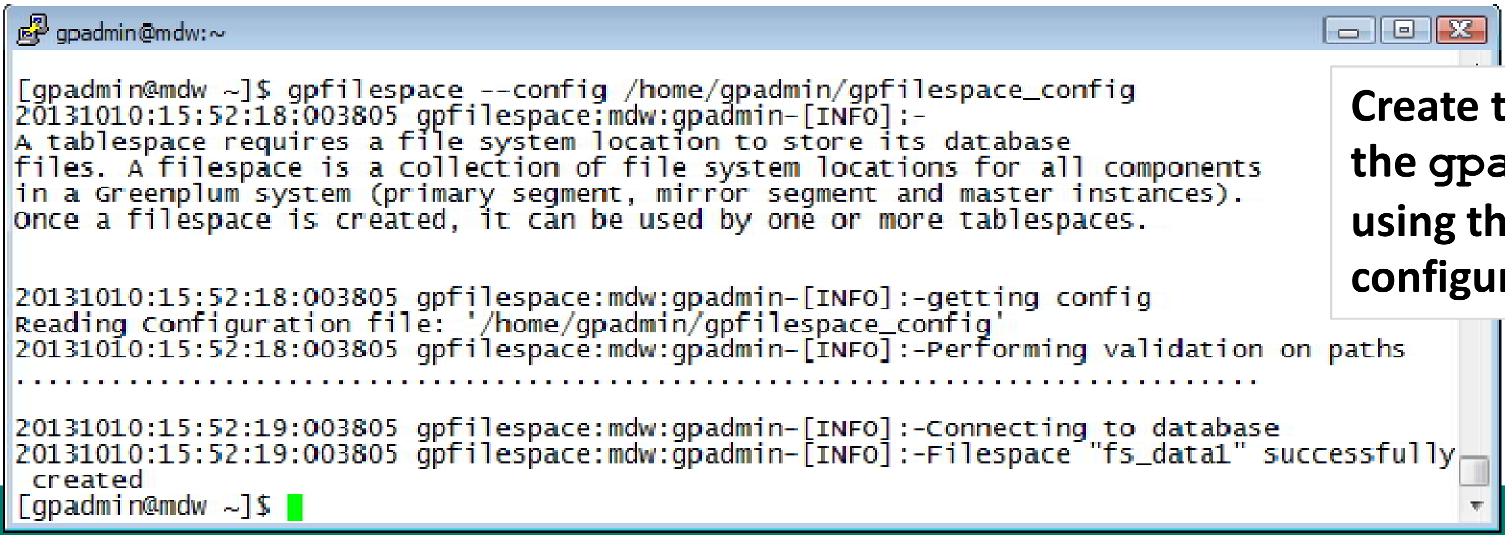
© 2015 Pivotal Software, Inc. All rights reserved.

Creating the Filespace

Example: Filespace Configuration File

```
$ cat gpfilespace_config
filespace:fs_data1
mdw:1:/data/user_spc/master/gpseg-1
smdw:6:/data/user_spc/master/gpseg-1
sdw2:3:/data/user_spc/primary/gpseg1
sdw2:4:/data/user_spc/mirror/gpseg0
sdw1:2:/data/user_spc/primary/gpseg0
sdw1:5:/data/user_spc/mirror/gpseg1
```

Configuration file contains all directories needed by masters and segments


gpadmin@mdw:~

```
[gpadmin@mdw ~]$ gpfilespace --config /home/gpadmin/gpfilespace_config
20131010:15:52:18:003805 gpfilespace:mdw:gpadmin-[INFO]:-
A tablespace requires a file system location to store its database
files. A filespace is a collection of file system locations for all components
in a Greenplum system (primary segment, mirror segment and master instances).
Once a filespace is created, it can be used by one or more tablespaces.


20131010:15:52:18:003805 gpfilespace:mdw:gpadmin-[INFO]:-getting config
Reading Configuration file: '/home/gpadmin/gpfilespace_config'
20131010:15:52:18:003805 gpfilespace:mdw:gpadmin-[INFO]:-Performing validation on paths
..............................................................................

20131010:15:52:19:003805 gpfilespace:mdw:gpadmin-[INFO]:-Connecting to database
20131010:15:52:19:003805 gpfilespace:mdw:gpadmin-[INFO]:-Filespace "fs_data1" successfully
 created
[gpadmin@mdw ~]$
```

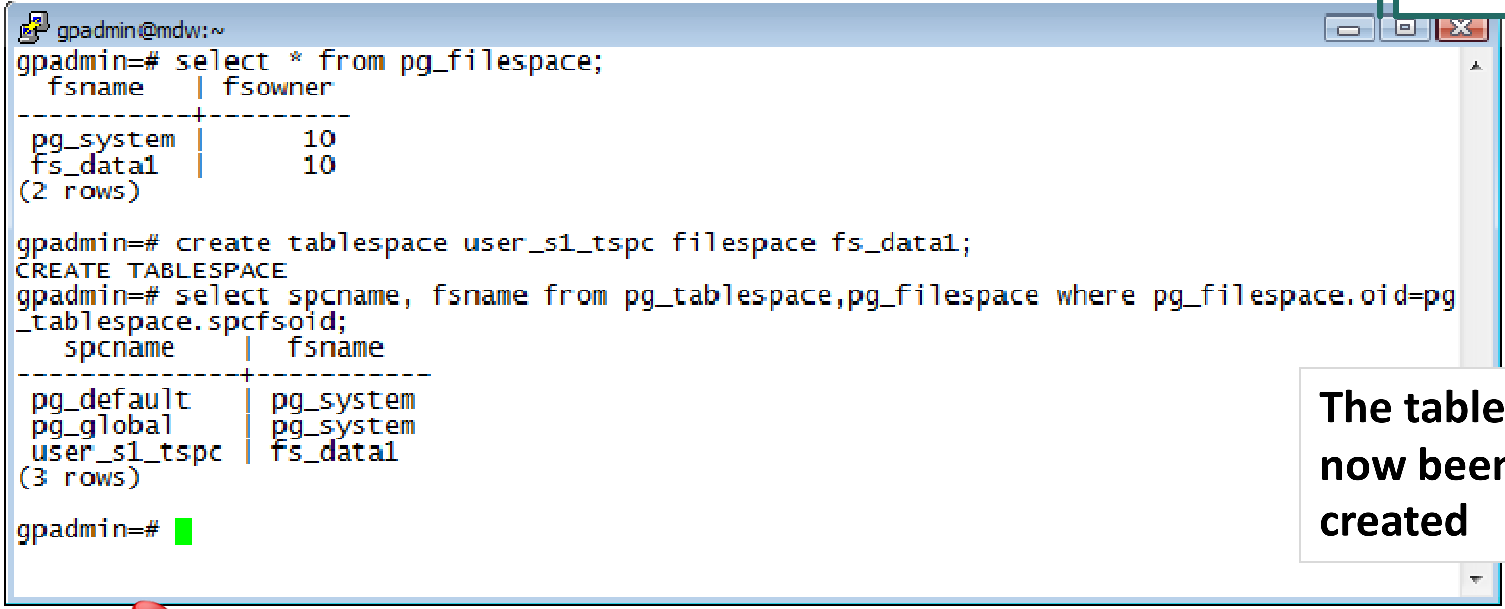
Create the filespace as the `gpadmin` user using the filespace configuration file

Creating the Tablespace

Usage: Syntax to create a tablespace

```
gpadmin=# CREATE TABLESPACE tablespace_name [OWNER username]
FILESPACE filespace_name;
```

Must be unique



```
gpadmin=# select * from pg_filespace;
  fsname    | fsowner
-----------+---------
 pg_system |       10
 fs_data1  |       10
(2 rows)

gpadmin=# create tablespace user_s1_tspc filespace fs_data1;
CREATE TABLESPACE
gpadmin=# select spcname, fsname from pg_tablespace,pg_filespace where pg_filespace.oid=pg
_tablespace.spcfsoid;
   spcname    |   fsname
--------------+-----------
 pg_default   | pg_system
 pg_global    | pg_system
 user_s1_tspc | fs_data1
(3 rows)

gpadmin=# 
```

The tablespace has now been successfully created

Note: The maximum number of tablespaces and filespaces are represented as `gp_max_tablespaces` and `gp_max_filespaces` in the master `postgresql.conf` file.

Pivotal

Applying the Tablespace

Object	Example
Database	`CREATE DATABASE tt_db` **`TABLESPACE user_s1_tspc;`**
Table	`CREATE TABLE tt_rt (id int)` **`TABLESPACE user_s1_tspc;`**

Partitioned Table	`CREATE TABLE ttct2_part_rt (id int, id2 int)`
	`PARTITION BY LIST (id) (`
	`        PARTITION one VALUES (1),`
	`        PARTITION two VALUES (2)` **`TABLESPACE user_s1_tspc,`**
	`        PARTITION three VALUES(3)`
	`);`

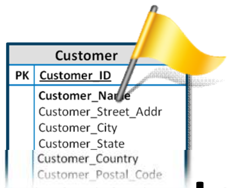
Index	`CREATE INDEX tt_idx ON tt_rt (id)` **`TABLESPACE user_s1_tspc;`**

Note: The default tablespace for the environment can be set by modifying the `default_tablespace` parameter in the master server's `postgresql.conf` file.

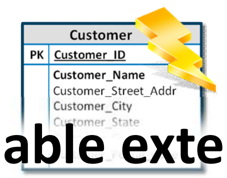
Pivotal™

© 2015 Pivotal Software, Inc. All rights reserved.

Additional Table Types



Temporary table



Readable external table

Writable external table

Temporary tables can be used for:

- Storing transient results needed for other session queries
- Perform transformations on data

External tables:

- Facilitate parallel data loading
- Stream data in from external sources
- Push data out of the database, in parallel

© 2015 Pivotal Software, Inc. All rights reserved.

Temporary Tables – Overview

- Session-specific
- Dropped at the end of the session
- Take precedence over permanent tables of the same name
- Created in a special schema created on connection to a session
- Are distributed
- Can be indexed and analyzed

© 2015 Pivotal Software, Inc. All rights reserved.

Creating a Temporary Table

Example: Creating a temporary table

```
gpadmin=# CREATE TEMP[ORARY] TABLE monthlytranssummary (
  storeid    INTEGER,
  customerid INTEGER,
  transmonth SMALLINT,
  salesamttot DECIMAL(10,2)
)
ON COMMIT PRESERVE ROWS
DISTRIBUTED BY (storeid, customerid);
```

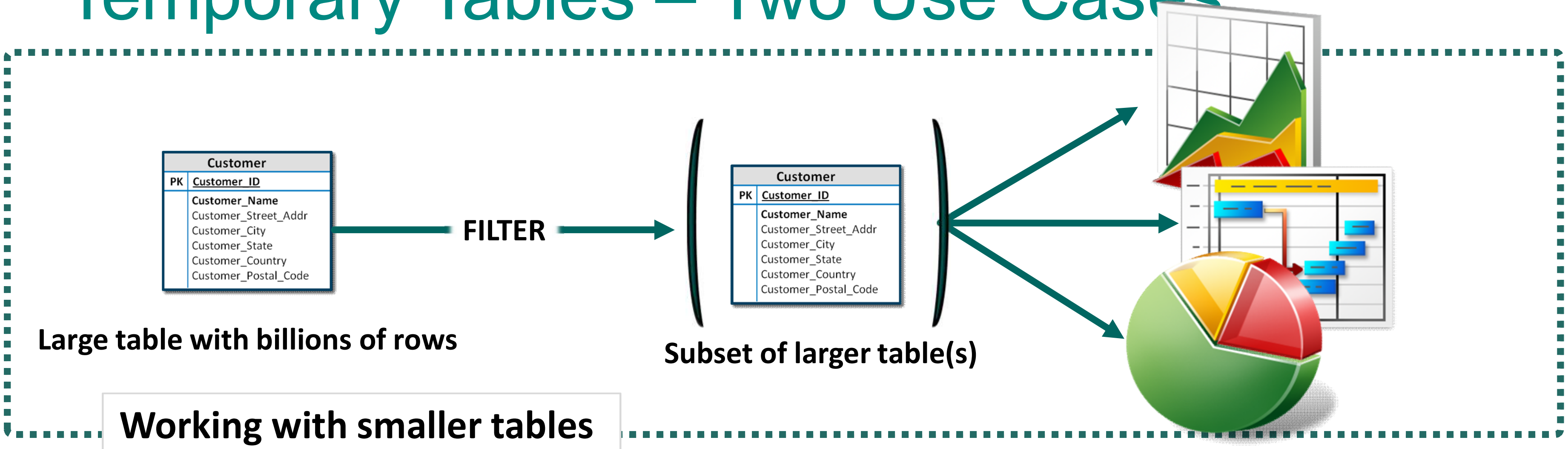
You can define how the temporary table will be handled *for transactions* with the `ON COMMIT` clause

The following options to the `ON COMMIT` clause let you define how a temporary table is handled:

- `PRESERVE ROWS` – No action is taken on the table
- `DELETE ROWS` – The table is truncated
- `DROP` – The table is dropped

© 2015 Pivotal Software, Inc. All rights reserved.

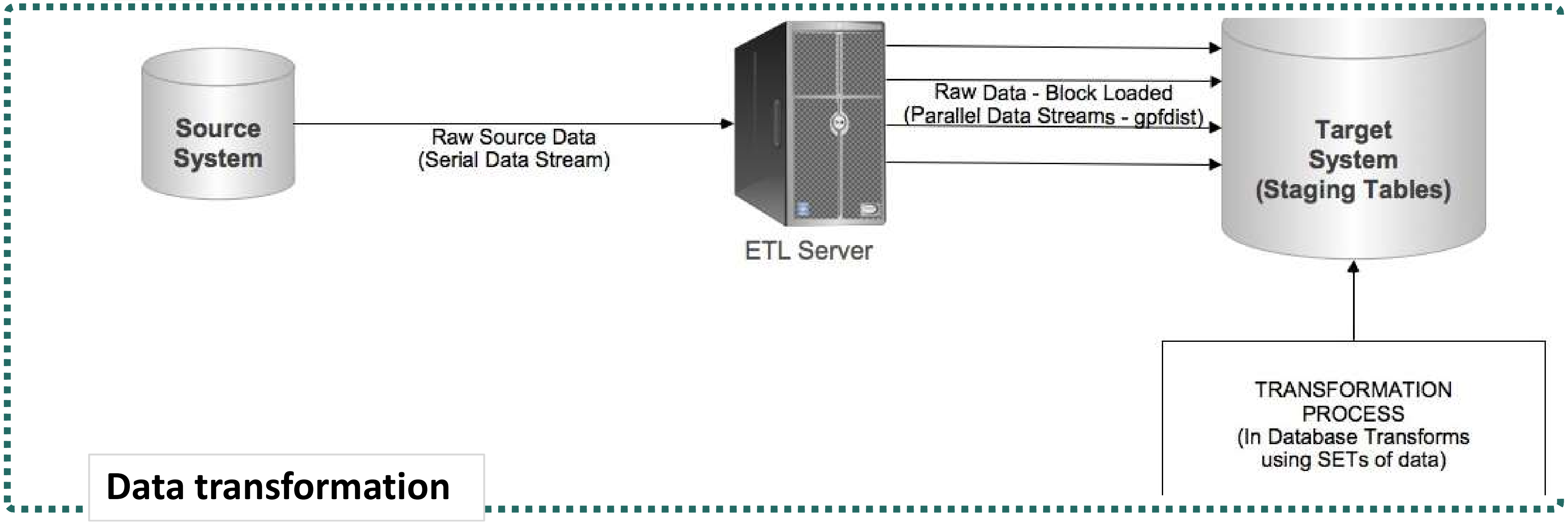
Temporary Tables – Two Use Cases



Large table with billions of rows

Subset of larger table(s)

Working with smaller tables

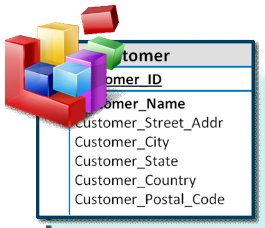


Data transformation

Pivotal

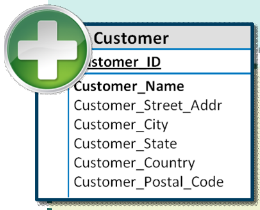
© 2015 Pivotal Software, Inc. All rights reserved.

Table Storage Models



Heap storage

- Default storage model
- Supports `INSERT`, `UPDATE`, `DELETE`
- Best for:
  - Data that is often modified
  - Smaller dimension tables
- Supports row-oriented tables
- Uses MVCC to support transactions



Append-optimized storage

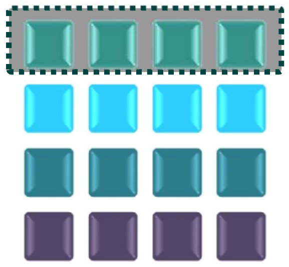
- Append-optimized storage model:
- Optimized for data warehouses
- Works best with denormalized data
- Supports `UPDATE` and `DELETE`
- Best for:
  - Older data
  - Large fact tables
- Supports row and column-oriented tables
- Supports in-database compression
- Uses a Visibility Map (visimap) to hide outdated rows

Pivotal™

© 2015 Pivotal Software, Inc. All rights reserved.

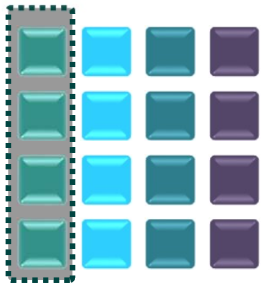
Row-Oriented and Column-Oriented Tables

Row-oriented storage



- Supports mixed workloads (INSERT, UPDATE, DELETE, SELECT)
- Is supported with on both heap and append-optimized storage

Column-oriented storage



- Works well with data warehouse workloads
- Works well for data where you aggregate over a small number of columns
- Efficient for data where you modify a single column
- Supported on append-optimized storage

© 2015 Pivotal Software, Inc. All rights reserved.

Creating Heap and Append-Optimized Tables

Action	Example
Creating a heap, row-oriented table	CREATE TABLE tc_heap (id int, descr text) DISTRIBUTED BY (id);
Creating an append-optimized, row-oriented table	CREATE TABLE tc_ao (id int, sales float) **WITH (appendonly=true)** DISTRIBUTED BY (id);
Creating an append-optimized, column-oriented table	CREATE TABLE tc_ao_c (id int, sales float) **WITH (appendonly=true, orientation=column)** DISTRIBUTED BY (id);

Note: You cannot modify the storage or orientation of a table once defined. You can create a new table with the desired options and migrate your data.

Pivotal

© 2015 Pivotal Software, Inc. All rights reserved.

Compressing Table Data

Compression Algorithm	Compression Levels	Description	Table-Level Compression	Row-Level Compression
ZLIB	1 - 9	Offers the most compact ratio with a potential impact to CPU performance	Supported	Supported
QUICKLZ	1	Offers faster, but lower, data compression	Supported	Supported
RLE_TYPE Delta Compression (specific data types)	1 - 4	Offers run-length encoding compression for columns based on repeated values	Unsupported	Supported

Question: What type of data do you think would work well with the different offerings of compression?

© 2015 Pivotal Software, Inc. All rights reserved.

Defining Append-Optimized Compression Tables

Action	Example
Creating a zlib compressed table with compression level 5	CREATE TABLE tc_ao_zlib5 (id int, sales float) **WITH (appendonly=true, compresstype=zlib, compresslevel=5)** DISTRIBUTED BY (id);
Creating a quicklz compressed table	CREATE TABLE tc_ao_quicklz (id int, sales float) **WITH (appendonly=true, compresstype=quicklz)** DISTRIBUTED BY (id);

Creating an AO table with an RLE compressed column and a zlib compressed column	CREATE TABLE tc_ao_rletype (
	    id int,
	    sales float **ENCODING (compresstype=zlib,**
	        **compresslevel=3)**,
	    salesdate date **ENCODING (compresstype=rle_type)**)
	WITH (appendonly=true, orientation=column)
	DISTRIBUTED BY (id);

© 2015 Pivotal Software, Inc. All rights reserved.

Defining Default Table Storage Options

`gp_default_storage_options`

Options	Level	Command

`APPENDONLY`
`BLOCKSIZE`
`CHECKSUM`
`COMPRESSTYPE`
`COMPRESSLEVEL`
`ORIENTATION`
	Object level	`CREATE TABLE … WITH (…)`

	Role level	`ALTER ROLE … SET …`
	Database level	`ALTER DATABASE … SET …`
	System level	`gpconfig …`

Highest priority

Lowest priority

Usage: Update default storage options at role level

```
names=> alter role student set
gp_default_storage_options='appendonly=true,compresstype=zlib';
Names=> set role student;
```

© 2015 Pivotal Software, Inc. All rights reserved.

Review

- Tablespaces and filespaces
- Additional table types (external, temp)
- Table storage models
- Compression options
- Test it out in the lab

© 2015 Pivotal Software, Inc. All rights reserved.

Pivotal

A NEW PLATFORM FOR A NEW ERA