
Pivotal
A NEW PLATFORM FOR A NEW ERA

# Data Definition Language (DDL) in GPDB

Pivotal® Greenplum Database

Pivotal

# Agenda

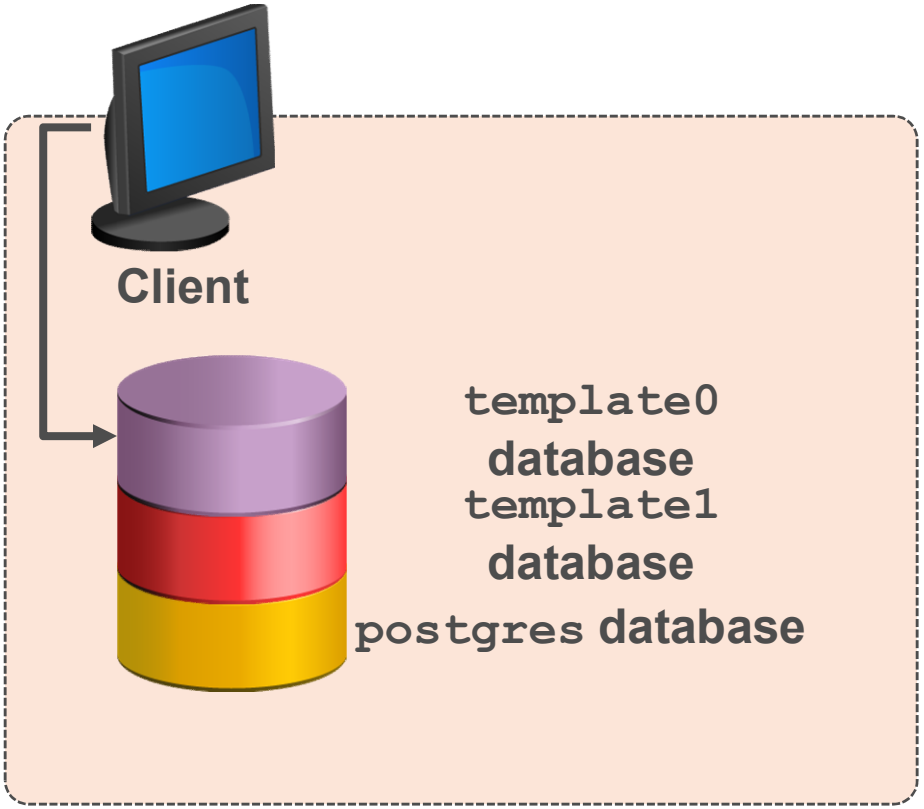
- Introduction
- Database
- Schema, table
- Constraints
- Data types
- View, sequence, index, tablespace, trigger

Pivotal

# Databases – Overview

Greenplum Database instance:

- Supports multiple databases
- Creates three databases by default:
  - `template0`
  - `template1`
  - `postgres`



# Database SQL Commands

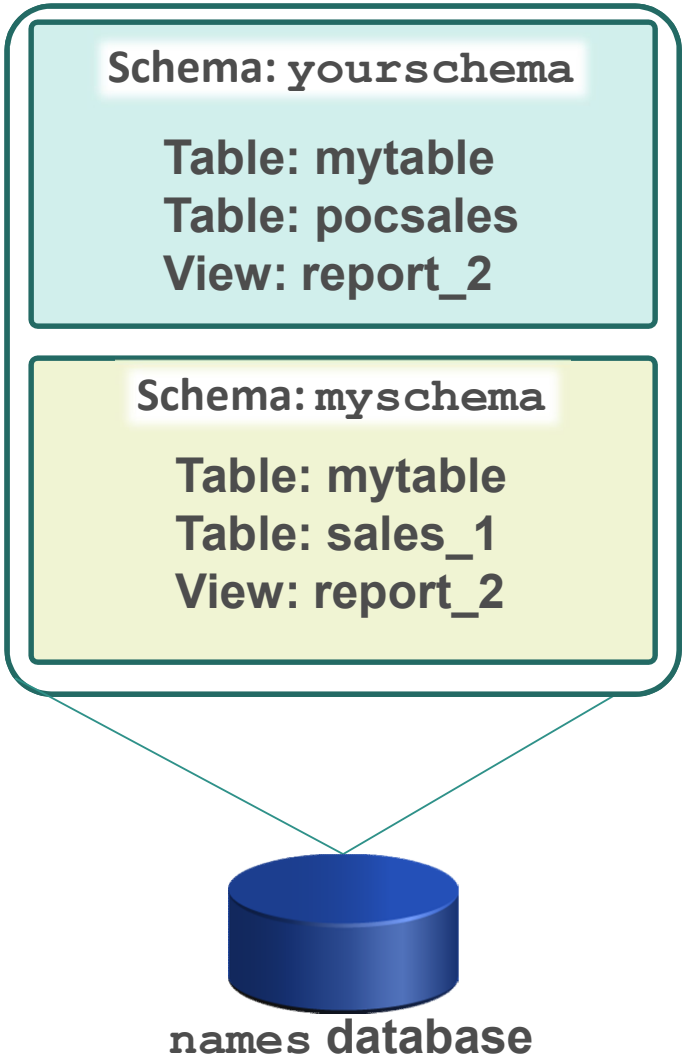
To manage databases, use the following SQL syntax or Greenplum application:

| Action | SQL Syntax | Greenplum Application |
|---|---|---|
| Create a database | `CREATE DATABASE` | `createdb` |
| Drop a database | `DROP DATABASE` | `dropdb` |
| Alter a database:<br>• Rename the database<br>• Assign a new owner<br>• Set configuration parameters | `ALTER DATABASE` | |

Pivotal

# Schema – Overview

- Logically organize objects
- A namespace
- Do not represent users
- Contain data objects
- Use *qualified* names to access objects
  EXAMPLE:
  `myschema.mytable`
- Can be added to the search path
  (`search_path`)



# Pre-existing Schemas

The following schemas exist in every database:

- `PUBLIC` schema
- System level schemas:

| Schema | Description |
|---|---|
| `pg_catalog` | Stores system catalog names, functions, and operators |
| `information_schema` | Contains views that describe objects in the database |
| `pg_toast` | Stores large objects that exceed page size |
| `pg_bitmapindex` | Stores bitmap index objects |
| `pg_aoseg` | Stores append-only objects |
| `gp_toolkit` | Administrative schema |

Pivotal

# Tables – Overview

## Tables in relational databases:

- Consist of columns and rows
- Have a fixed number and order of named columns
- Have a variable number of rows reflecting individual records
- No guaranteed row order

## Tables share the following characteristics:

- All tables in Greenplum Database are distributed based on a distribution key
- Some table features are not yet supported in Greenplum Database
  - Foreign key constraints
  - Limit on unique constraints

Pivotal

# Table Distribution Keys – Overview

- One column, or multiple columns, used to divide the data among the segments
- Should have high cardinality
- Specified with `CREATE TABLE` or `ALTER TABLE` using the `DISTRIBUTED BY` clause

```
CREATE TABLE sales
 (dt date, prc float, qty int, cust_id int,
  prod_id int, vend_id int)
DISTRIBUTED BY (dt, cust_id, prod_id);
```

- If table lacks a good candidate for distribution key, you can use `DISTRIBUTED RANDOMLY`
- If not explicitly declared, defaults to the table's PRIMARY KEY or the first column of the table

Pivotal

# Table SQL Commands – Modifying a Table

- Renaming columns
- Renaming tables
- Adding and removing columns

```
ALTER TABLE product ADD COLUMN msg_body TEXT;
```

- Adding and removing constraints

```
ALTER TABLE product ALTER COLUMN prod_no SET NOT NULL;
```

- Changing default values

```
ALTER TABLE product ALTER COLUMN prod_no SET DEFAULT -
999;
```

- Changing column data types

```
ALTER TABLE products ALTER COLUMN price TYPE
NUMERIC(10,2);
```

Pivotal

# Table Column Basics

- Each column has a data type
- Large set of data types
- User defined types supported
- Limit on the number of columns in a table: 1600

Pivotal

# Table and Column Constraints – Overview

Table and column constraints are supported with some limitations:

- `CHECK` table or column constraints
- `NOT NULL` column constraints
- `UNIQUE` column constraints
- `PRIMARY KEY` is used as the distribution key by default
- `FOREIGN KEY` constraints are supported, but *referential integrity* is **not** enforced

Pivotal

# Table and Column Constraints

**Example: Table and Column Constraints**

```
CREATE TABLE TR_CONSTRAINTS (
  transactionid int NOT NULL UNIQUE,
  price numeric CHECK (price > 0),
  on_sale char DEFAULT 'n'
);
```

Column cannot be NULL

Data in the column for that table is unique

If no value is entered, a default value of 'n' is entered for that field

Column cannot have a value equal to or less than zero

Pivotal

# External Table SQL Commands

To manage external tables, use the following SQL syntax:

| Action | SQL Syntax |
|---|---|
| Create a readable external table | `CREATE EXTERNAL [WEB] TABLE LOCATION (…) \| EXECUTE '…' FORMAT '…' (DELIMITER, '…');` |
| Create a writable external table | `CREATE WRITABLE EXTERNAL [WEB] TABLE LOCATION (…) \| EXECUTE '…' FORMAT '…' (DELIMITER, '…') DISTRIBUTED BY(…)\|RANDOMLY;` |
| Drop an external table | `DROP EXTERNAL [WEB] TABLE` |

Pivotal

# Data Types – Overview

Data types:

- Constrain a column to storing only a specific kind of data
- In Greenplum are the same as data types in PostgreSQL
- Supports types defined in the SQL standard
- You can also define arrays, single- or multi-dimensional
- Can be created with the `CREATE TYPE` SQL command

For Greenplum distribution key columns:

- No geometric data types are supported
- No user-defined data types are supported

Pivotal

# Greenplum Database Data Types

| Name | Size | Description |
|---|---|---|
| bigint | 8 bytes | Large range integer |
| bigserial | 8 bytes | Large autoincrementing integer |
| bit [(n)] | *n* bits | Fixed-length bit string |
| bit varying [(n)] | Actual # of bits | Variable-length bit string |
| boolean | 1 byte | Logical boolean (true/false) |
| box | 32 bytes | Rectangular box in the plane<br>(*not allowed in distribution key columns)* |
| bytea | 1 byte + binary | Variable-length binary string |
| character [(n)] | 1 byte + *n* | Fixed-length, blank padded |
| character varying [(n)] | 1 byte + *string size* | Variable-length with limit |
| cidr | 12 or 24 bytes | IPV4 and IPV6 networks |
| circle | 24 bytes | Circle in the plane<br>(*not allowed in distribution key columns*) |

Pivotal

# Greenplum Database Data Types (Cont'd.)

| Name | Size | Description |
|---|---|---|
| date | 4 bytes | Calendar date (year, month, day) |
| decimal [(p,s)] | variable | User-specified precision, exact |
| double precision | 8 bytes | Variable-precision, inexact |
| inet | 12 or 24 bytes | IPV4 and IPv6 hosts and networks |
| integer | 4 bytes | Usual choice for integer |
| interval [(p)] | 12 bytes | Time span |
| lseg | 32 bytes | Line segment in the plane<br>(*not allowed in distribution key columns)* |
| macaddr | 6 bytes | MAC addresses |
| money | 4 bytes | Currency amount |
| path | 16+16*n* bytes | Geometric path in the plane<br>(*not allowed in distribution key columns*) |
| point | 16 bytes | Geometric point in the plane<br>(*not allowed in distribution key columns*) |

Pivotal

# Greenplum Database Data Types (Cont'd.)

| Name | Size | Description |
|---|---|---|
| polygon | 40+16*n* bytes | Geometric path in the plane (*not allowed in distribution key columns*) |
| real | 4 bytes | Variable-precision, inexact |
| serial | 4 bytes | Autoincrementing integer |
| smallint | 2 bytes | Small range integer |
| text | 1 byte + *string size* | Variable unlimited length |
| time [(p)] [without time zone] | 8 bytes | Time of day only |
| time [(p)] [with time zone] | 12 bytes | Time of day only, with time zone |
| timestamp [(p)] [without timezone] | 8 bytes | Both date and time |
| timestamp [(p)] [with timezone] | 8 bytes | Both date and time, with time zone |

Pivotal

# Casting Data Types

A type cast:

- Is used to convert one data type to another
- Can be accomplished with the following syntax:

| Syntax | Example |
|---|---|
| `type 'string'` | `REAL '2.117902'` |
| `'string'::type` | `'some text'::TEXT` |
| `CAST ('string' AS type)` | `CAST ('2.117902' AS REAL)` |
| `value::type` | `(1.0 / 3)::NUMERIC(3, 2)` |

- Can be omitted if there is no ambiguity
- Accepts regular SQL notation or dollar quoting for string constants

Pivotal

# Additional Database Objects

Let us examine:

- Views
- Indexes
- Sequences
- Triggers
- Tablespaces

Pivotal

# Views – Overview

Views:

- Let you save frequently used or complex queries
- Can be accessed with `SELECT` statement
- Are not materialized on disk
- Are managed and accessed with the following commands:

| Action | Command |
|---|---|
| Create a view | `CREATE VIEW` |
| Drop or remove a view | `DROP VIEW` |
| List all views | `\dv` |
| See view definition | `\dv+ view_name` |

Pivotal

# View Example

```
CREATE VIEW iot_json AS
SELECT
    (CASE WHEN a[1] = '' THEN '0.0' ELSE a[1] END)::FLOAT absolute_throttle_pos_b
  , a[2]::FLOAT acceleration
  , (CASE WHEN a[3] = '' THEN '0.0' ELSE a[3] END)::FLOAT accelerator_throttle_pos_d
  , (CASE WHEN a[4] = '' THEN '0.0' ELSE a[4] END)::FLOAT accelerator_throttle_pos_e
  , a[5]::FLOAT barometric_pressure
  , a[7]::FLOAT catalyst_temp
  , (CASE WHEN a[8] = '' THEN '0.0' ELSE a[8] END)::FLOAT control_module_voltage
  , a[9]::FLOAT coolant_temp
  , a[10]::FLOAT distance_with_mil_on
  , TO_TIMESTAMP(SUBSTRING(a[27] FROM 1 FOR 10)::INT4)::TIMESTAMP date_time
FROM
(
 SELECT REGEXP_SPLIT_TO_ARRAY(csv_from_json(line), ',') a FROM iot_json_ext
) AS iot_row;
```

Pivotal

# Index – Overview

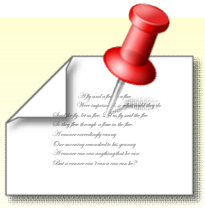
Indexes:

- Use random seek to find a rows in a relational database
- Should be used sparingly in Greenplum Database
- Only used for highly selective queries
- Are not always favored at query runtime
- Should be checked to ensure they improve performance
- Should be dropped if not used in queries
- Are created automatically if using `PRIMARY KEY`

# Sequences – Overview

A sequence:

- Is used to auto-increment unique ID columns
- Is generated by the Greenplum Master sequence generator process (`seqserver`)
- Has some function limitations:
  - `lastval` and `currval` not supported
  - `setval` cannot be used in queries that update data
  - `nextval` sometimes grabs a block of values for some queries. So values may sometimes be skipped in the sequence if all of the block turns out not to be needed at the segment level.
  - Sequences are based on bigint arithmetic



**Note:** bigserial and serial data types are auto incrementing integers.

Pivotal

# Triggers – Overview

- Triggers are considered to be a type of function
- They automatically execute a particular function when a specific type of operation is performed (like update, delete, and insert)
- Have limited support in Greenplum, in that:
  - The function must be `IMMUTABLE` to execute on segment instances
    - The immutable function cannot execute any SQL or modify the database.
  - Triggers are not supported on Append-Optimized tables.

Pivotal

# Tablespaces

- Provide an alternate location for tables, partitions, temp tables, or indexes
- Can facilitate placing certain data on faster disk drives, for example
- Require superuser privileges to create or move data objects to them
- Are created using the following syntax:
  `CREATE TABLESPACE` *`tablespace_name`*
  `FILESPACE` *`filespace_name`*

Pivotal

# Review

- Database
- Schema, table
- Constraints
- Data types
- View, sequence, index, tablespace, trigger

- Work the lab!

Pivotal


Pivotal
A NEW PLATFORM FOR A NEW ERA