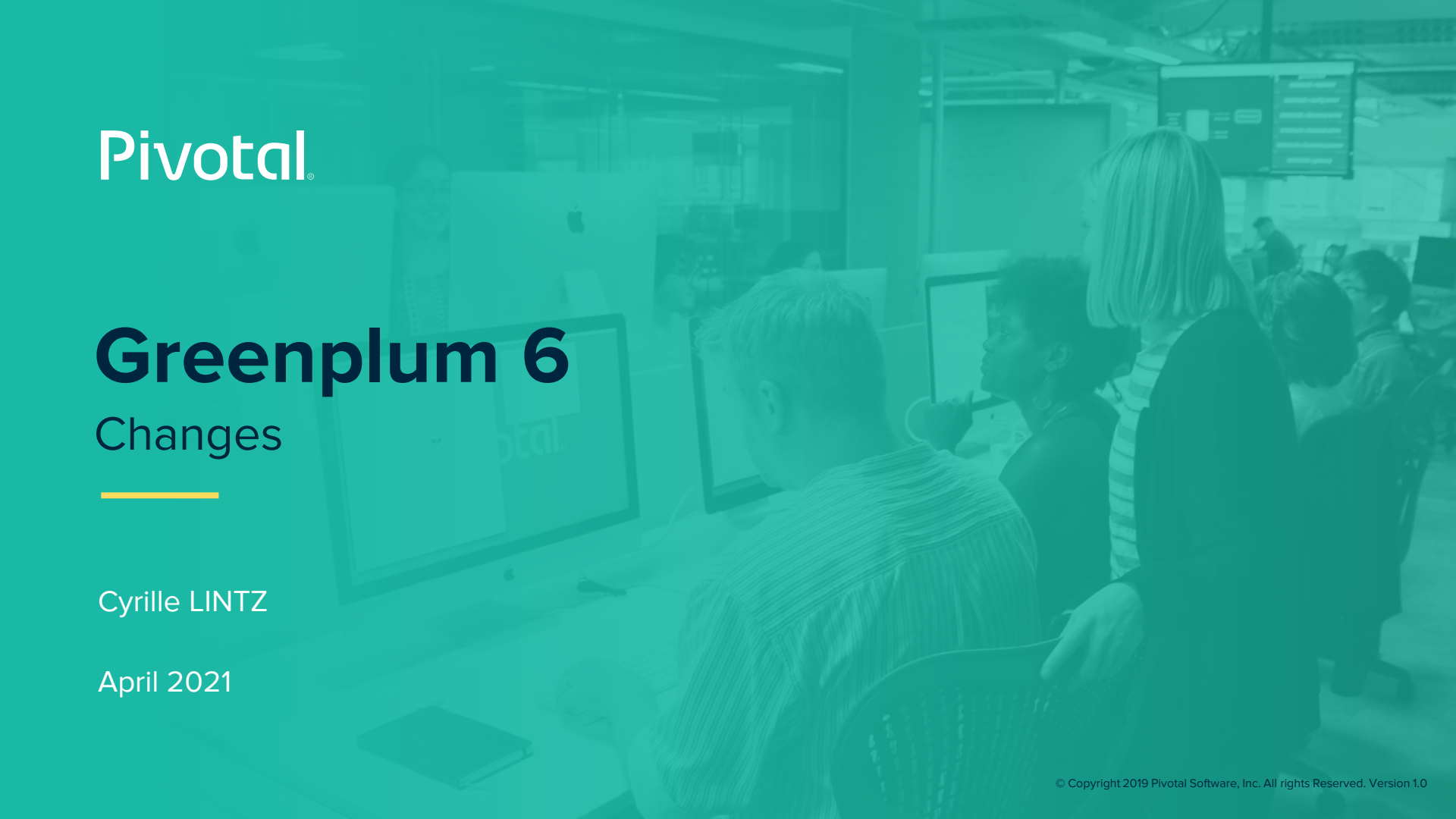


Pivotal

Greenplum 6

Changes

Cyrille LINTZ

April 2021

Agenda

- Querying
- Explain Analyze
- Data Types
- Storage / DDL
- Materialized Views
- Extensions
- Administration
- Backup / Restore
- Security
- Catalog & gp_toolkit
- Parameters
- Expansion
- Removed features
- Other items

Overview

- Merge with PostgreSQL 9.4 (so includes changes related to all releases between PG 8.4 and PG 9.4)

- Greenplum 6 provides a single client and loader tool package that you can download and install on a client system. Previous Greenplum releases provided separate client and loader packages.

Greenplum 6

Querying

Order by - Collate - 1/2

- Collation support to specify sort order and character classification behavior for data at the column level (PostgreSQL 9.1).
- Ability to define a COLLATE clause to a column (if collatable data type).
- If not specified, the column data type's default collation is used.
- GPORCA supports collation only when all columns in the query use the same collation. If columns in the query use different collations, then Greenplum uses the Postgres query planner.

Order by - Collate - 2/2

```
create table test_order
(f_default_collate text
,f_collate_c text collate "C"
,f_description text);
```

Order by field with no collate

```
select * from test_order order by f_default_collate;
f_default_collate | f_collate_c |  f_description
------------------+-------------+------------------
1                 | 1           | only the value 1
 2                |  2          | space + value 2
3                 | 3           | only the value 3
```

Order by field with collate

```
select * from test_order order by f_collate_c;
f_default_collate | f_collate_c |  f_description
------------------+-------------+------------------
 2                |  2          | space + value 2
1                 | 1           | only the value 1
3                 | 3           | only the value 3
```

Order by field with no collate, but adding collate

```
select * from test_order order by f_default_collate collate "C";
 f_default_collate | f_collate_c |  f_description
-------------------+-------------+------------------
 2                 |  2          | space + value 2
1                  | 1           | only the value 1
3                  | 3           | only the value 3
```

Pivotal

Recursive Query - 1/2

```
create table public.places
(place text,is_in text)
distributed by (place);



COPY public.places from stdin;
U.S.A       \N
California U.S.A
Florida     U.S.A
Massachusetts           U.S.A
San Francisco           California
Los Angeles             California
San Diego  California
Miami       Florida
Orlando     Florida
Tampa       Florida
United Kingdom          \N
London      United Kingdom
Leeds       United Kingdom
\.
```

Example 1 - Get Parent - Child

```
demo=# WITH RECURSIVE q AS
(SELECT place, is_in
              FROM places
              WHERE is_in is null
UNION ALL
              SELECT m.place, m.is_in
              FROM places m
              JOIN q ON q.place = m.is_in)
SELECT place, is_in FROM q;

     place       |     is_in
-----------------+----------------
 U.S.A           |
 Massachusetts   | U.S.A
 Florida         | U.S.A
 California      | U.S.A
 Orlando         | Florida
 Tampa           | Florida
 Miami           | Florida
 Los Angeles     | California
 San Diego       | California
 San Francisco   | California
 United Kingdom  |
 Leeds           | United Kingdom
 London          | United Kingdom
(13 rows)
```

Recursive Query - 2/2

Example 2 - Get level in the hierarchy + path

```
demo=# WITH RECURSIVE q (place, is_in, depth, path, cycle)
AS (   SELECT place, is_in, 1 as depth     ,ARRAY[place], false as cycle  FROM places
where is_in is null
UNION ALL
SELECT m.place, m.is_in, q.depth + 1 as depth ,path || m.place, m.place = ANY(path) FROM places m
JOIN q ON q.place = m.is_in
where not cycle)
SELECT place, is_in, depth, array_to_string(path,' ==> '), cycle FROM q;


     place      |     is_in      | depth |            array_to_string              | cycle
----------------+----------------+-------+-----------------------------------------+-------
 United Kingdom |                |     1 | United Kingdom                          | f
 Leeds          | United Kingdom |     2 | United Kingdom ==> Leeds                | f
 London         | United Kingdom |     2 | United Kingdom ==> London               | f
 U.S.A          |                |     1 | U.S.A                                   | f
 Massachusetts  | U.S.A          |     2 | U.S.A ==> Massachusetts                 | f
 Florida        | U.S.A          |     2 | U.S.A ==> Florida                       | f
 California     | U.S.A          |     2 | U.S.A ==> California                    | f
 Orlando        | Florida        |     3 | U.S.A ==> Florida ==> Orlando           | f
 Tampa          | Florida        |     3 | U.S.A ==> Florida ==> Tampa             | f
 Miami          | Florida        |     3 | U.S.A ==> Florida ==> Miami             | f
 Los Angeles    | California     |     3 | U.S.A ==> California ==> Los Angeles    | f
 San Diego      | California     |     3 | U.S.A ==> California ==> San Diego      | f
 San Francisco  | California     |     3 | U.S.A ==> California ==> San Francisco  | f
(13 rows)
```

Greenplum 6

Explain Analyze

Changes in Explain Analyze output

- New information in the Explain analyze output
  - Tag that indicates if a step was Never executed
  - Duration to build the query plan
  - Information about the number of loops in a Nested Loop
- Changes in the format of the output
  - Timing in the same line of the step
  - Spill information

Explain analyze output in GPDB 6.x

```
                                                                QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------
 Insert  (cost=0.00..20487146.10 rows=102677647 width=24) (actual time=157.047..13418948.146 rows=109250681 loops=1)
   ->  Result  (cost=0.00..9256778.50 rows=102677647 width=28) (actual time=156.093..13282405.495 rows=109250681 loops=1)
         ->  Redistribute Motion 48:48  (slice2; segments: 48)  (cost=0.00..9253903.52 rows=102677647 width=24) (actual time=156.068..13263276.035 rows=109250681 loops=1)
               Hash Key: (((share0_ref2.id + 1) + (share0_ref3.id * 1048576)))
               ->  Sequence  (cost=0.00..9246190.38 rows=102677647 width=24) (actual time=10.267..12842420 [illegible] loops=1)
                     ->  Shared Scan (share slice:id 2:0)  (cost=0.00..433.07 rows=21846 width=1) (actual time=5.843..7.094 rows=22152 loops=1)
                           ->  Materialize  (cost=0.00..433.07 rows=21846 width=1) (never executed)
                                 ->  Seq Scan on one_m  (cost=0.00..431.74 rows=21846 width=32) (actual time=0.038..1.702 rows=22152 loops=1)
                     ->  Result  (cost=0.00..9243293.05 rows=102677647 width=24) (actual time=2.601..12833716.286 rows=110760000 loops=1)
                           ->  Nested Loop  (cost=0.00..9240828.78 rows=102677647 width=16) (actual time=0.191..31308.180 rows=110760000 loops=1)
                                 Join Filter: true
                                 ->  Broadcast Motion 48:48  (slice1; segments: 48)  (cost=0.00..432.23 rows=4701 width=8) (actual time=0.057..7.439 rows=5000 loops=1)
                                       ->  Result  (cost=0.00..432.14 rows=98 width=8) (actual time=0.096..2.783 rows=120 loops=1)
                                             Filter: (share0_ref3.id < 5000)
                                             ->  Shared Scan (share slice:id 1:0)  (cost=0.00..431.42 rows=21846 width=8) (actual time=0.069..1.583 rows=22152 loops=1)
                                 ->  Shared Scan (share slice:id 2:0)  (cost=0.00..431.42 rows=21846 width=8) (actual time=0.000..2.809 rows=22148 loops=5001)
 Planning time: 36.552 ms
   (slice0)    Executor memory: 316K bytes avg x 48 workers, 320K bytes max (seg34).
   (slice1)    Executor memory: 236K bytes avg x 48 workers, 236K bytes max (seg0).
 * (slice2)    Executor memory: 11049023685K bytes avg x 48 workers, 11213022006K bytes max (seg4).  Work_mem: 960K bytes max, 928K bytes wanted.
 Memory used:  147456kB
 Memory wanted:  6268kB
 Optimizer: Pivotal Optimizer (GPORCA) version 3.65.0
 Execution time: 13422870.304 ms
```

Step never executed

Duration to build the plan

loops for the nested loop

Explain analyze output in GPDB 6.x

```
 Gather Motion 2:1  (slice3; segments: 2)  (cost=0.00..2100.16 rows=500 width=10) (actual time=2403.886..2406.254 rows=500 loops=1)
   ->  HashAggregate  (cost=0.00..2100.14 rows=250 width=10) (actual time=2402.724..2402.750 rows=255 loops=1)
         Group Key: t2.e
         Extra Text: (seg0)   Hash chain length 4.0 avg, 8 max, using 63 of 64 buckets; total 1 expansions.

         ->  Redistribute Motion 2:2  (slice2; segments: 2)  (cost=0.00..1979.01 rows=1000000 width=10) (actual time=1687.076..2253.255 rows=1020000 loops=1)
               Hash Key: t2.e
               ->  HashAggregate  (cost=0.00..1947.71 rows=1000000 width=10) (actual time=1675.242..1884.061 rows=100091
                     Group Key: t2.e, t2.d
                     Extra Text: (seg1)   1000916 groups total in 32 batches; 1 overflows; 1000916 spill groups.
 (seg1)   Hash chain length 4.0 avg, 17 max, using 288932 of 294912 buckets; total 10 expansions.

                     ->  Redistribute Motion 2:2  (slice1; segments: 2)  (cost=0.00..1699.86 rows=1000000 width=10) (actual time=135.292..1218.033 rows=1000916 loops=1)
                           Hash Key: t2.e, t2.d
                           ->  HashAggregate  (cost=0.00..1668.56 rows=1000000 width=10) (actual time=144.686..941.668 rows=1001450 loops=1)
                                 Group Key: t2.e, t2.d
                                 Extra Text: (seg0)   Hash chain length 5.1 avg, 18 max, using 195414 of 196608 buckets; total 10 expansions.

                                 ->  Hash Left Join  (cost=0.00..1175.88 rows=1999146 width=10) (actual time=14.977..471.282 rows=100145
                                       Hash Cond: (t2.a = t1.a)
                                       Extra Text: (seg0)   Hash chain length 2.0 avg, 9 max, using 25666 of 32768 buckets.
                                       ->  Seq Scan on t2  (cost=0.00..455.20 rows=1000000 width=14) (actual time=0.011..103.769 rows=1001450 loops=1)
                                       ->  Hash  (cost=432.43..432.43 rows=50000 width=4) (actual time=14.816..14.816 rows=50082 loops=1)
                                             ->  Seq Scan on t1  (cost=0.00..432.43 rows=50000 width=4) (actual time=0.017..6.627 rows=50082 loops=1)
 Planning time: 19.214 ms
   (slice0)    Executor memory: 151K bytes.
   (slice1)    Executor memory: 12970K bytes avg x 2 workers, 12970K bytes max (seg0).  Work_mem: 1174K bytes max.
 * (slice2)    Executor memory: 9607K bytes avg x 2 workers, 9607K bytes max (seg0).  Work_mem: 8434K bytes max, 48577K bytes wanted.
   (slice3)    Executor memory: 144K bytes avg x 2 workers, 144K bytes max (seg0).
 Memory used:  25600kB
 Memory wanted:  194905kB
 Optimizer: Pivotal Optimizer (GPORCA) version 3.65.0
 Execution time: 2426.314 ms
(33 rows)
```

Information on spill files

Time to get the last row

Time to get the first row

Pivotal

Greenplum 6

Stored Procedures

Error Handling: custom Detail and Hint

DDL Function

```
CREATE FUNCTION sp_test_raise_specific_mess
(p_param text)
RETURNS text
AS $$
BEGIN
   if p_param = 'test'
   then
      return 'OK';
   else
      raise exception 'the parameter % is not the
expected one (e.g. test)', p_param
      using ERRCODE='22012';
   end if;

   EXCEPTION
      WHEN OTHERS THEN
         RAISE EXCEPTION '%,%', sqlerrm,sqlstate
            using DETAIL='detail custo',
                  HINT='hint custo';
END;
$$
LANGUAGE plpgsql;
```

Output when no error

```
demo=# select sp_test_raise_specific_mess('test');
 sp_test_raise_specific_mess
-----------------------------
 OK
(1 row)
```

Output when error

```
demo=# select sp_test_raise_specific_mess('test1');
ERROR:  the parameter test1 is not the expected one
(e.g. test),22012
DETAIL:  detail custo
HINT:  hint custo
```

Plpgsql: Change in variable substitution (1/2)

```
CREATE OR REPLACE FUNCTION f_test_var (p_var integer, out v_value text)
LANGUAGE plpgsql
AS $$ BEGIN
   SELECT 'value ' || p_var::text into v_value FROM(SELECT p_var) AS TABAL;
END; $$;
```

Variable used in SELECT part

Before GPDB6

```
select f_test_var(4);
 f_test_var
------------
 value 4
```

From GPDB6

```
select f_test_var (4);
ERROR:  column reference "p_var" is ambiguous
LINE 1: SELECT 'value ' || p_var::text  FROM(SELECT p_va...
                           ^
DETAIL:  It could refer to either a PL/pgSQL variable or a table
column.
QUERY:  SELECT 'value ' || p_var::text  FROM(SELECT p_var) AS TABAL
CONTEXT:  PL/pgSQL function f_test_var(integer) line 2 at SQL statement
```

Plpgsql: Change in variable substitution (2/2)

Solution 1: Add an alias

```
CREATE OR REPLACE FUNCTION f_test_var_fix1
(p_var integer, out v_value text)
LANGUAGE plpgsql
AS $$
BEGIN
   SELECT 'value ' || alias_p_var::text
   into v_value
   FROM(SELECT p_var as alias_p_var) AS TABAL;
END; $$;
```

```
select f_test_var_fix1(4);
 f_test_var_fix1
-----------------
 value 4
```

Solution 2: Add #variable_conflict use_variable

```
CREATE OR REPLACE FUNCTION f_test_var_fix2
(p_var integer, out v_value text)
LANGUAGE plpgsql
AS $$
#variable_conflict use_variable
BEGIN
   SELECT 'value ' || p_var::text
   into v_value
   FROM(SELECT p_var) AS TABAL;
END; $$;
```

```
select f_test_var_fix2(4);
 f_test_var_fix2
-----------------
 value 4
```

More information in postgresql documentation:

https://www.postgresql.org/docs/9.4/plpgsql-implementation.html

Greenplum 6

Data Types

JSONB (PostgreSQL 9.4)

- The data types json and jsonb are almost identical
  - json data is stored as an exact copy of the JSON input text
  - jsonb stores data in a decomposed binary form

Benefits

- more efficiency
- significantly faster to process
- supports indexing (GIN, btree, and hash)
- simpler schema designs (replacing entity-attribute-value tables)

Drawbacks

- slightly slower input (due to added conversion overhead),
- Possibly more disk space than plain json (due to a larger table footprint)
- certain queries (especially aggregate ones) may be slower (due to the lack of statistics)

JSONB (PostgreSQL 9.4) - Filter result

JSON

```
SELECT count(*) FROM books_json
WHERE data->'published' = 'false';

ERROR:  operator does not exist: json = unknown
LINE 2: WHERE data->'published' = 'false';
                               ^
HINT:  No operator matches the given name and
argument type(s). You might need to add explicit
type casts.
```

```
SELECT count(*) FROM books_json
WHERE data->>'published' = 'false';

 count
-------
     1
```

JSONB

```
SELECT count(*) FROM books_jsonb
WHERE data->'published' = 'false';

 count
-------
     1
```

Pivotal

JSONB (PostgreSQL 9.4) - Check containment & Existence

JSON

```
SELECT '["A", "B", "C"]'::json
@> '["A", "B"]'::json as check;

ERROR:  operator does not exist: json @> json
LINE 2: @> '["A", "B"]'::json as check;
        ^
HINT:  No operator matches the given name and
argument type(s). You might need to add explicit
type casts.
```

JSONB

```
SELECT '["A", "B", "C"]'::jsonb
@> '["A", "B"]'::jsonb as check;

 check
-------
 t
```

```
SELECT '["A", "B", "C"]'::json ? 'A' as check;

ERROR:  operator does not exist: json ? unknown
LINE 1: SELECT '["A", "B", "C"]'::json ? 'A' as
check;
                                       ^
HINT:  No operator matches the given name and
argument type(s). You might need to add explicit
type casts.
```

```
SELECT '["A", "B", "C"]'::jsonb ? 'A' as check;

 check
-------
 t
```

Pivotal

JSONB (PostgreSQL 9.4) - Indexes

JSON

```
CREATE TABLE table_json
(id integer
, data json
) distributed by (id);

CREATE INDEX idx_btree
ON table_json ((data->'published'));

ERROR:  data type json has no default operator class
for access method "btree"
HINT:  You must specify an operator class or define
a default operator class for the data type.

CREATE INDEX idx_gin
ON table_json using gin((data->'published'));

ERROR:  data type json has no default operator class
for access method "gin"
HINT:  You must specify an operator class or define
a default operator class for the data type.
```

JSONB

```
CREATE TABLE table_jsonb
(id integer
, data jsonb
) distributed by (id);

CREATE INDEX idx_btree
ON table_jsonb ((data->'published'));
CREATE INDEX



CREATE INDEX idx_gin
ON table_jsonb  using gin((data->'published'));
CREATE INDEX
```

Pivotal

Greenplum 6

Storage / DDL

CREATE Table - appendoptimized

- **appendonly** replaced by **appendoptimized** (both syntaxes work)

With appendonly

```
CREATE TABLE table_appendonly (a int)
with (appendonly=true) distributed randomly;

CREATE TABLE
```

With appendoptimized

```
CREATE TABLE table_appendoptimized (a int)
with (appendoptimized=true) distributed randomly;

CREATE TABLE
```

Content of pg_class

```
select relname,relkind,relstorage,reloptions from pg_class
where relname like 'table_append%';

        relname         | relkind | relstorage |    reloptions
------------------------+---------+------------+------------------
 table_appendoptimized | r       | a          | {appendonly=true}
 table_appendonly      | r       | a          | {appendonly=true}
```

Pivotal

Replicated Tables - 1/3

- The DISTRIBUTED REPLICATED clause replicates the entire table to all Greenplum Database segment instances.
- Useful when it is necessary to execute user-defined functions on segments when the functions require access to all rows in the table
- Useful to improve query performance by preventing broadcast and redistribution motions.
- Replicated tables can have both PRIMARY KEY and UNIQUE column constraints.

Replicated Tables - 2/3

With Non Replicated table

```
create table table_non_replicated (a int , b text)
distributed randomly;

insert into table_non_replicated
select id, 'val ' || id
from generate_series (1,10000) id;

select pg_relation_size('table_non_replicated');
pg_relation_size
------------------
          458752
```

```
select gp_segment_id, count(*) from
table_non_replicated group by 1;
 gp_segment_id | count
---------------+-------
             0 |  5011
             1 |  4989
```

With Replicated Tables

```
create table table_replicated (a int , b text)
distributed replicated;

insert into table_replicated
select id, 'val ' || id
from generate_series (1,10000) id;

select pg_relation_size('table_replicated');
pg_relation_size
------------------
         917504
```

Size is multiplied by the number of primaries

```
select gp_segment_id, count(*) from table_replicated
group by 1;
ERROR:  column "gp_segment_id" does not exist
LINE 1: select gp_segment_id, count(*) from ...
               ^
```

The field gp_segment_id doesn't exist in replicated tables

Pivotal

Replicated Tables - 3/3

With Non Replicated table

```
explain select count(*) from table_fact f inner join table_non_replicated d on f.a = d.a;
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=0.00..874.31 rows=1 width=8)
   ->  Gather Motion 2:1  (slice3; segments: 2)  (cost=0.00..874.31 rows=1 width=8)
         ->  Aggregate  (cost=0.00..874.31 rows=1 width=8)
               ->  Hash Join  (cost=0.00..874.31 rows=50000 width=1)
                     Hash Cond: (table_fact.a = table_non_replicated.a)
                     ->  Redistribute Motion 2:2  (slice1; segments: 2)  (cost=0.00..433.15 rows=50000 width=4)
                           Hash Key: table_fact.a
                           ->  Seq Scan on table_fact  (cost=0.00..432.15 rows=50000 width=4)
                     ->  Hash  (cost=431.22..431.22 rows=5000 width=4)
                           ->  Redistribute Motion 2:2  (slice2; segments: 2)  (cost=0.00..431.22 rows=5000 width=4)
                                 Hash Key: table_non_replicated.a
                                 ->  Seq Scan on table_non_replicated  (cost=0.00..431.12 rows=5000 width=4)
 Optimizer: PQO version 3.29.0
```

With Replicated table

```
explain select count(*) from table_fact f inner join table_replicated d on f.a = d.a;
                                          QUERY PLAN
-----------------------------------------------------------------------------------------------
 Aggregate  (cost=0.00..874.73 rows=1 width=8)
   ->  Gather Motion 2:1  (slice1; segments: 2)  (cost=0.00..874.73 rows=1 width=8)
         ->  Aggregate  (cost=0.00..874.73 rows=1 width=8)
               ->  Hash Join  (cost=0.00..874.73 rows=50000 width=1)
                     Hash Cond: (table_fact.a = table_replicated.a)
                     ->  Seq Scan on table_fact  (cost=0.00..432.15 rows=50000 width=4)
                     ->  Hash  (cost=431.23..431.23 rows=10000 width=4)
                           ->  Seq Scan on table_replicated  (cost=0.00..431.23 rows=10000 width=4)
 Optimizer: PQO version 3.29.0
```

1 slice vs 3 slices
No redistribution

Unlogged Tables

- Data written to unlogged tables is not written to the write-ahead (WAL) log, which makes them considerably faster than ordinary tables.
- However, the contents of an unlogged table are not replicated to mirror segment instances. Also an unlogged table is not crash-safe.
- Any indexes created on an unlogged table are automatically unlogged as well.
- For a unlogged table, the field **relpersistence** of pg_class has the value **'u'**

```
create unlogged table table_unlogged
(a int , b text)
distributed randomly;
```

Possibly useful for Working tables and staging tables (Truncate, Insert)

- **Temp tables are unlogged**

Zstandard Compression Algorithm

- New compression type: **Zstandard**

```
CREATE TABLE table_zstd5(id int4, t text)
WITH (appendoptimized=true, compresstype=zstd, compresslevel=5);
```

- ZStandard can provide for:
  - either good compression ratio or speed, depending on compression level
  - or a good compromise on both.

- The compression rate is good:

Compression type	Table size (B)	Ratio vs Heap Size
Heap	8,847,360	
Zlib - 3	1,215,304	7
Zstd - 1	132,608	67
Zstd - 19	91,976	96

CREATE TABLE - LIKE

- With LIKE syntax, ability to copy:
  - Default values
  - Indexes (New)
  - Constraints
  - Comments (New)
  - Storage (New)

Example

```
create table t_like
(a int ,
 b text default 'a',
 c text not null)
with (appendoptimized=true
, orientation=column
, compresstype=quicklz)
distributed by (a);

create index idx_t_like on  t_like(a);

comment on column t_like.c
is 'comment field c';
```

CREATE TABLE - LIKE (before GPDB 6)

Original table

```
demo=# \d+ t_like
                                        Append-Only Columnar Table "public.t_like"
 Column |  Type   |     Modifiers     | Storage  | Compression Type | Compression Level | Block Size |        Description
--------+---------+-------------------+----------+------------------+-------------------+------------+---------------------------
 a      | integer |                   | plain    | quicklz          | 1                 | 32768      |
 b      | text    | default 'a'::text | extended | quicklz          | 1                 | 32768      |
 c      | text    | not null          | extended | quicklz          | 1                 | 32768      | comment field c
Checksum: t
Indexes:
    "idx_t_like" btree (a)
Distributed by: (a)
Options: appendonly=true, orientation=column, compresstype=quicklz
```

```
create table t_like_copy
(like t_like including defaults including constraints);
```

Copied table

```
demo=# \d+ t_like_copy
                            Table "public.t_like_copy"
 Column |  Type   |     Modifiers     | Storage  | Description
--------+---------+-------------------+----------+-------------
 a      | integer |                   | plain    |
 b      | text    | default 'a'::text | extended |
 c      | text    | not null          | extended |
Distributed by: (a)
```

Default is copied

Constraint is copied

CREATE TABLE - LIKE (from GPDB 6)

Original table

```
demo=# \d+ t_like
                                                Append-Only Columnar Table "public.t_like"
 Column |  Type   |     Modifiers     | Storage  | Stats target | Compression Type | Compression Level | Block Size |
Description
--------+---------+-------------------+----------+--------------+------------------+-------------------+------------+-----------------
 a      | integer |                   | plain    |              | quicklz          | 1                 | 32768      |
 b      | text    | default 'a'::text | extended |              | quicklz          | 1                 | 32768      |
 c      | text    | not null          | extended |              | quicklz          | 1                 | 32768      | comment field c
Checksum: t
Indexes:
    "idx_t_like" btree (a)
Distributed by: (a)
Options: appendonly=true, orientation=column, compresstype=quicklz
```

```
create table t_like_copy (like t_like including all);
```

Copied table

```
demo=# \d+ t_like_copy
                                                Append-Only Columnar Table "public.t_like_copy"
 Column |  Type   |     Modifiers     | Storage  | Stats target | Compression Type | Compression Level | Block Size
|        Description
--------+---------+-------------------+----------+--------------+------------------+-------------------+------------+-----------------
----------
 a      | integer |                   | plain    |              | quicklz          | 1                 | 32768      |
 b      | text    | default 'a'::text | extended |              | quicklz          | 1                 | 32768      |
 c      | text    | not null          | extended |              | quicklz          | 1                 | 32768      | comment on field
t_like.c
Checksum: t
Indexes:
    "t_like_copy_a_idx" btree (a)
Distributed by: (a)
Options: appendonly=true, orientation=column, checksum=true, compresslevel=1, compresstype=quicklz
```

Constraint is copied

Default is copied

Comment is copied

Index is copied

Storage is copied

Pivotal

CREATE TABLE - Unique constraint

Before GPDB 6

```
create table table_unique (a int, b int, c int
, constraint test_uniq unique (b,c))
distributed by (b,c);
NOTICE:  CREATE TABLE / UNIQUE will create implicit
index "table_unique_b_key" for table "table_unique"
CREATE TABLE

create table table_unique (a int, b int, c int
, constraint test_uniq unique (b,c))
distributed by (b);
NOTICE:  CREATE TABLE / UNIQUE will create implicit
index "table_unique_b_key" for table "table_unique"
CREATE TABLE

create table table_unique (a int, b int, c int
, constraint test_uniq unique (b,c))
distributed by (c);
ERROR:  UNIQUE constraint and DISTRIBUTED BY
definitions incompatible
HINT:  When there is both a UNIQUE constraint, and a
DISTRIBUTED BY clause, the DISTRIBUTED BY clause
must be equal to or a left-subset of the UNIQUE
columns
```

From GPDB 6

```
create table table_unique (a int, b int, c int
, constraint test_uniq unique (b,c))
distributed by (b,c);
CREATE TABLE

create table table_unique (a int, b int, c int
, constraint test_uniq unique (b,c))
distributed by (b);
CREATE TABLE

create table table_unique (a int, b int, c int
, constraint test_uniq unique (b,c))
distributed by (c);
CREATE TABLE
```

No more require
to be a left-subset

CREATE TABLE - Unique constraint - Alter DK

Before GPDB 6

```
create table table_unique (a int, b int, c int
, constraint test_uniq unique (b,c))
distributed by (b,c);
NOTICE:  CREATE TABLE / UNIQUE will create implicit
index "table_unique_b_key" for table "table_unique"
CREATE TABLE

alter table table_unique set distributed by (b);
ALTER TABLE

alter table table_unique set distributed by (c);
ALTER TABLE
```

From GPDB 6

```
create table table_unique (a int, b int, c int
, constraint test_uniq unique (b,c))
distributed by (b,c);
CREATE TABLE

alter table table_unique set distributed by (b);
ALTER TABLE

alter table table_unique set distributed by (c);
ALTER TABLE
```

Samer behavior

CREATE TABLE - Distribution – Custom Hash Algorithm

- Ability to specify the hash function used to distribute data across segment instances.
- See the example described https://gpdb.docs.pivotal.io/6-0/admin_guide/ddl/ddl-table.html#topic34

With OLD hash operator

```
demo=# EXPLAIN (COSTS OFF) SELECT a, b FROM atab_old_hash,
btab_old_hash WHERE a |=| b;
                              QUERY PLAN
-----------------------------------------------------------------------
 Gather Motion 48:1  (slice3; segments: 48)
   ->  Hash Join
         Hash Cond: (atab_old_hash.a |=| btab_old_hash.b)
         ->  Redistribute Motion 48:48  (slice1; segments: 48)
               Hash Key: atab_old_hash.a
               ->  Seq Scan on atab_old_hash
         ->  Hash
               ->  Redistribute Motion 48:48  (slice2; segments: 48)
                     Hash Key: btab_old_hash.b
                     ->  Seq Scan on btab_old_hash
```

2 redistributions

```
demo=# sselect gp_segment_id, a from atab_old_hash order by abs(a),a,1;
 gp_segment_id | a
---------------+----
            32 |  0
            42 | -1
             4 |  1
            37 | -2
            37 |  2
            27 | -3
            24 |  3
```

With NEW custom hash operator

```
demo=# EXPLAIN (COSTS OFF) SELECT a, b FROM atab_new_hash,
btab_new_hash WHERE a |=| b;
                        QUERY PLAN
----------------------------------------------------------
 Gather Motion 48:1  (slice1; segments: 48)
   ->  Hash Join
         Hash Cond: (atab_new_hash.a |=| btab_new_hash.b)
         ->  Seq Scan on atab_new_hash
         ->  Hash
               ->  Seq Scan on btab_new_hash
```

No redistribution
➔ co-located join

```
demo=# select gp_segment_id, a from atab_new_hash order by abs(a),a,1;
 gp_segment_id | a
---------------+----
            32 |  0
             4 | -1
             4 |  1
            37 | -2
            37 |  2
            24 | -3
            24 |  3
```

- val and val are co-located

Pivotal

Field - Reserved words

- The ROWS and RANGE SQL keywords have changed from reserved to unreserved, and may be used as table or column names without quoting.

Before GPDB 6

```
create table t_reserved_words
(rows int,
range int)
distributed randomly;
ERROR:  syntax error at or near "rows"
LINE 2: (rows int,
         ^
```

```
-- Workaround using ""
create table t_reserved_words
demo-# ("rows" int,
demo(# "range" int)
demo-# distributed randomly;
CREATE TABLE
```

From GPDB 6

```
create table t_reserved_words
(rows int
, range int)
distributed randomly;
CREATE TABLE
```

```
select rows,range from t_reserved_words ;
 rows | range
------+-------
(0 rows)
```

Greenplum 6

Materialized Views

Pivotal

Materialize Views

- Materialized views are similar to views. A materialized view enables you to save a frequently used or complex query, then access the query results in a SELECT statement as if they were a table. Materialized views persist the query results in a table-like form. While access to the data stored in a materialized view can be much faster than accessing the underlying tables directly or through a view, the data is not always current.

- Main commands are:
  - `CREATE MATERIALIZED VIEW` (ability to define the distribution policy and storage + collect statistics)
  - `REFRESH MATERIALIZED VIEW [with no data]`

Materialize Views vs Views

View

```
create view v_sales_cust
as
select c.marital_status, count(*)
from online_sales.sales_fact f
inner join cust_dim c
on c.cust_key = f.cust_key
group by 1;
```

```
explain select * from v_sales_cust;
                QUERY PLAN
-------------------------------------------
 Gather Motion 2:1  (slice3)
  -> HashAggregate
     Group Key: cust_dim.marital_status
     -> Redistribute Motion 2:2  (slice2)
        Hash Key: cust_dim.marital_status
        -> Result
           -> HashAggregate
              Group Key: cust_dim.marital_status
              -> Hash Join
                 Hash Cond:(sales_fact.cust_key=cust_dim.cust_key)
                 -> Seq Scan on sales_fact
                 -> Hash
                    -> Broadcast Motion 2:2  (slice1)
                       ->  Seq Scan on cust_dim
 Optimizer: Pivotal Optimizer (GPORCA) version 3.86.0
```

Materialized view

```
create materialized view vm_sales_cust
With (appendoptimized=true, compresstype=quicklz) as
select c.marital_status, count(*)
from online_sales.online_sales_fact f
inner join customer_dimension_test c
on c.customer_key = f.customer_key
group by 1
Distributed randomly;
```

```
explain select * from vm_sales_cust;
                QUERY PLAN
-------------------------------------------
 Gather Motion 2:1  (slice1)
   -> Seq Scan on vm_sales_cust
 Optimizer: Pivotal Optimizer (GPORCA) version 3.86.0
```

Only a seq scan
no join - no filter

Greenplum 6

Extensions

Oracle Extension

- The Oracle Compatibility Functions are now available in Greenplum Database as an extension, based on the PostgreSQL orafce project at https://github.com/orafce/orafce.

Before GPDB 6

```
cd $GPHOME/share/postgresql/contrib
psql -d demo -f orafunc.sql
```

⇒ **Create a single schema oracomp**

From GPDB 6

```
CREATE EXTENSION orafce;
```

⇒ **Create 13 schemas including oracle**

Impact on upgrade if the source DB uses oracompat.

Disk Quota

- 2 levels of disk quotas
  - **Schema**
  - **Role** (all tables owned by a user) inside 1 database: it works for superuser

- Set disk quota:

```
SELECT diskquota.set_schema_quota('schema1', '25MB');
SELECT diskquota.set_role_quota('user1', '50MB');
```

- Reset disk quota:

```
SELECT diskquota.set_schema_quota('schema1', '-1');
SELECT diskquota.set_role_quota('user1', '-1');
```

- Status of Disk quota:

```
SELECT * from diskquota.show_fast_schema_quota_view;
SELECT * from diskquota.show_fast_role_quota_view;
```

The query is cancelled only if before starting, the quota is already exceeded.

If a query exceeds the quota during its execution, the query will anyway succeed

Disk Quota: example for a schema

```
SELECT diskquota.set_schema_quota('schema1', '25MB');
```

```
Create table schema1.t1 (a bigint);
Insert into schema1.t1 select generate_series(1,100);
```

```
INSERT 0 100
```

```
SELECT schema_name, quota_in_MB,
pg_size_pretty(nspsize_in_bytes) as schema_size
FROM diskquota.show_fast_schema_quota_view;
```

```
 schema_name | quota_in_mb | schema_size
-------------+-------------+-------------
 schema1     |          25 |  64 kB
```

```
Insert into schema1.t1 select generate_series(1,1000000);
```

```
INSERT 0 1000000
```

```
SELECT schema_name, quota_in_MB,
pg_size_pretty(nspsize_in_bytes) as schema_size
FROM diskquota.show_fast_schema_quota_view;
```

```
 schema_name | quota_in_mb | schema_size
-------------+-------------+-------------
 schema1     |          25 |  35 MB
```

```
Insert into schema1.t1 select generate_series(1,10);
```

```
ERROR:  schema's disk space quota exceeded with
name:schema1
```

Auto_explain

- Automatically collect query plan for slowest queries in the Master Greenplum log

```
alter role user1 set session_preload_libraries='auto_explain';

alter role user1 set auto_explain.log_min_duration=60000;
```

- Ability to collect explain analyze output

```
alter role user1 set auto_explain.log_analyze=on;
```

Good tool to quickly diagnose long queries

Very interesting when a query used temp tables - no need to recreate the temp tables to get the explain plan

Greenplum 6

Administration

Pivotal

Reindex of shared catalog tables (like pg_shdepend)

Before GPDB 6

```
demo=# reindex table pg_shdepend ;
ERROR:  shared table "pg_shdepend" can only be
reindexed in stand-alone mode

demo=# reindex table pg_database ;
ERROR:  shared table "pg_database" can only be
reindexed in stand-alone mode
```

Workaround: Follow the KB https://pvtl.force.com/s/article/Reindex-fails-with-ERROR--shared-table-xxx-can-only-be-reindexed-in-stand-alone-mode

This method requires to stop Greenplum

From GPDB 6

```
demo=# reindex table pg_shdepend;
REINDEX


demo=# reindex table pg_database;
REINDEX
```

reindexdb -s takes now into account the shared tables

Pivotal

Changes on VACUUM and vacuumdb

- `VACUUM` can more easily skip pages it cannot lock. Now `VACUUM` skips a block it cannot lock and retries the block later.

  Reduce the frequency of a vacuum appearing to be "stuck," which occurs when VACUUM waits to lock a block for cleanup and another session has held a lock on the block for a long time

- `VACUUM` rechecks block visibility after it has removed dead tuples. If all remaining tuples in the block are visible to current and future transactions, the block is marked as all-visible.

- The partitions with no data are age-frozen. so that they do not have to be vacuumed separately and do not affect calculation of the number of remaining transaction IDs before wraparound occurs. These tables include the root and intermediate tables in the partition hierarchy and, if they are append-optimized, their associated meta-data tables.

  Reduce the number of required VACUUM and not increase the number of transactions

- `vacuumdb` utility has a new `-F` (`--freeze`) option to freeze row transaction information.

Greenplum 6

Backup / Restore

Pivotal

Parallel backup/restore tools

- gpcrondump & gpdbrestore are removed

  The only tools for parallel backup/restore are gpbackup / gprestore

- gpbackup & gprestore are separated from Greenplum DB installer.

  Separate binaries in Pivotal Network

Serial backup/restore tools

- pg_dump and pg_dumpall
  - New option **`--lock-wait-timeout=timeout`**
    When specified, instead of waiting indefinitely the dump fails if the utility cannot acquire shared table locks within the specified number of milliseconds.
  - The **`-d`** and **`-D`** options are removed.
    The corresponding long versions **`--inserts`** and **`--column-inserts`** are still supported.
  - New **`--binary-upgrade`** option, for use by in-place upgrade utilities.
  - New **`-w`** (**`--no-password`**) option

- pg_restore
  - New option **`-j`** (**`--number-of-jobs`**)
    This option can reduce time to restore a large database by running tasks such as loading data, creating indexes, and creating constraints concurrently.
  - New **`-w`** (**`--no-password`**) option

Greenplum 6

Security

Pivotal

GRANT on Schema

- Ability to all tables in a schema - one command line

```
demo=# create role user1 login password 'pivotal';
CREATE ROLE
demo=# create schema sch1;
CREATE SCHEMA
demo=# create table sch1.t1 (a int);
CREATE TABLE
demo=# create table sch1.t2 (a int);
CREATE TABLE
demo=# demo=# \dp+ sch1.*
                            Access privileges
 Schema | Name | Type  | Access privileges | Column access privileges
--------+------+-------+-------------------+--------------------------
 sch1   | t1   | table |                   |
 sch1   | t2   | table |                   |
(2 rows)

demo=# grant select on all tables in schema sch1 to user1;
GRANT

demo=# \dp+ sch1.*
                            Access privileges
 Schema | Name | Type  |    Access privileges    | Column access privileges
--------+------+-------+-------------------------+--------------------------
 sch1   | t1   | table | gpadmin=arwdDxt/gpadmin+|
        |      |       | user1=r/gpadmin         |
 sch1   | t2   | table | gpadmin=arwdDxt/gpadmin+|
        |      |       | user1=r/gpadmin         |
```

Column level privileges - 1/2

As gpadmin

```
create table t_grant_column (a int, b int, c int) distributed randomly;
insert into t_grant_column select id, id*2,id * 3 from generate_series(1,100) id;

\dp t_grant_column
                              Access privileges
 Schema |      Name      | Type  | Access privileges | Column access privileges
--------+----------------+-------+-------------------+--------------------------
 public | t_grant_column | table |                   |

grant select (b) on table t_grant_column to user1;

\dp t_grant_column
                              Access privileges
 Schema |      Name      | Type  | Access privileges | Column access privileges
--------+----------------+-------+-------------------+--------------------------
 public | t_grant_column | table |                   | b:                      +
        |                |       |                   |   user1=r/gpadmin
```

As user1

```
select * from t_grant_column limit 1;
ERROR:  permission denied for relation t_grant_column

select b from t_grant_column limit 1;
 b
----
 12
(1 rows)
```

Column level privileges - 2/2

As gpadmin

```
grant select on table t_grant_column to user2;

\dp t_grant_column
Access privileges
 Schema |      Name      | Type  |    Access privileges     | Column access privileges
--------+----------------+-------+--------------------------+--------------------------
 public | t_grant_column | table | gpadmin=arwdDxt/gpadmin+| b:                      +
        |                |       | user2=r/gpadmin          |   user1=r/gpadmin



revoke select (b) on table t_grant_column from user2;
NOTICE:  no privileges could be revoked
```

As user2

```
select * from t_grant_column limit 1;
 a | b  | c
---+----+----
 6 | 12 | 18
(1 row)
```

Unable to revoke a single column if a grant is set on the table level
Require to grant each wanted column individually

Pivotal

Greenplum 6

Catalog & gp_toolkit

pg_stat_activity: changes

Field	Description	Status	Comment
pid	Process ID of this backend	Renamed	Old name = procpid
client_hostname	Host name of the connected client.	New	This field will only be non-null for IP connections, and only when log_hostname is enabled.
query	Text of this backend's most recent query. .	renamed	Old name = current_query If state is active, it shows the current query. If other states, it shows the last executed query
state_change	Session defaults for server configuration parameters	New	Time when the state was last changed
state	Current overall state of this backend.	New	**active**: executing a query. **idle**: waiting for a new client command. **idle in transaction**: in a transaction, but is not currently executing a query. **idle in transaction (aborted)**: same as idle in transaction, except one of the statements in the transaction caused an error. **fastpath function call**: executing a fast-path function. **disabled**: reported if track_activities is disabled

pg_stat_activity: Example of changes

GPDB 4 or GPDB 5

```
SELECT  'terminate', sess_id
FROM pg_stat_activity
WHERE current_query = '<IDLE>'
AND usename = 'xv86888'
AND now() - query_start > INTERVAL ' 15 minutes';
```

```
select procpid, usename,query_start
,extract(epoch from (now() - query_start))::integer as elapse_sec
from pg_stat_activity
where current_query not like '<IDLE%'
and extract(epoch from (now() - query_start))/60::integer>120
```

```
copy (select -1,5432, procpid, sess_id, usename
, now()-query_start as elapse
,replace(substr(pg_stat_activity.current_query, 1, 50),' ',' ')
from pg_catalog.pg_stat_activity where sess_id<>1071118)
to stdout;
```

GPDB 6

```
SELECT  'terminate', sess_id
FROM pg_stat_activity
WHERE state = 'idle'
AND usename = 'xv86888'
AND now() - state_change > INTERVAL ' 15 minutes';
```

```
select pid, usename,query_start
,extract(epoch from (now() - query_start))::integer as elapse_sec
from pg_stat_activity
where state not like 'idle%'
and extract(epoch from (now() - query_start))/60::integer>120
```

```
copy (select -1,5432, pid, sess_id, usename
, now()-query_start as elapse
,replace(substr(pg_stat_activity.query, 1, 50),' ',' ')
from pg_catalog.pg_stat_activity where sess_id<>1071118)
to stdout;
```

pg_locks: changes (+ example)

Field	Description	Status	Comment
virtualxid	Virtual ID of a transaction, or NULL if the object is not a virtual transaction ID	New	Old name = procpid
virtualtransaction	Virtual ID of the transaction that is holding or awaiting this lock	Renamed + New type	Old name = transaction The new type is text: old type was xid
fastpath	True if lock was taken via fastpath, false if lock is taken via main lock table.	New	

GPDB 4 or GPDB 5

```
SELECT COUNT(*)
from (select  database,l.pid, usename
   , client_addr, application_name
   , case when granted then 'locking'
      else 'waiting' end status
   , c.relname, l.relation, locktype, l.mode
   , l.transactionid, l.transaction, a.current_query
   , sum(case when granted then 0 else 1 end)
       over (partition by database, c.relname)
       num_waiting_for_table
  from pg_locks l
  , pg_class c
  , pg_stat_activity a
  where l.relation=c.oid
  and l.pid=a.procpid
  ) v
where num_waiting_for_table > 0;
```

GPDB 6

```
SELECT COUNT(*)
from (select  database,l.pid, usename
   , client_addr, application_name
   , case when granted then 'locking'
      else 'waiting' end status
   , c.relname, l.relation, locktype, l.mode
   , l.transactionid, l.virtualtransaction, a.query
   , sum(case when granted then 0 else 1 end)
       over (partition by database, c.relname)
       num_waiting_for_table
  from pg_locks l
  , pg_class c
  , pg_stat_activity a
  where l.relation=c.oid
  and l.pid=a.pid
  ) v
where num_waiting_for_table > 0;
```

gp_distribution_policy: changes

- gp_distribution_policy now contains more information about Greenplum Database tables and the policy for distributing table data across the segments including the operator class of the distribution hash functions.

Field	Description	Status	Comment
policytype	Table distribution policy	New	p - Partitioned policy. Table data is distributed among segment instances. r - Replicated policy. Table data is replicated on each segment instance.
numsegments	The number of segment instances on which the table data is distributed.	New	
attrnums	The column number(s) of the distribution column(s).	New	old name = attrnums The data type changed: from smallint[] to int2vector
distclass	The operator class identifier(s) of the distribution column(s).	New	

The new function **pg_get_table_distributedby()** was added to get the distribution policy for a table as a string.

pg_class

Field	Description	Status	Comment
reloftype	OID of an entry in pg_type for a composite type.	New	
relallvisible	Number of all-visible blocks	New	this value may not be up-to-date
relpersistence	The type of object persistence	New	p = heap or AO table u = unlogged temporary table t = temporary table
relispopulated		New	
relreplident		New	
relminmxid		New	
relrefs	Unused	Removed	
reltoastidxid	For a TOAST table, the OID of its index.	Removed	
relkind	The type of object	Changed (New values)	m = materialized view f = foreign table b = append-only block directory M = append-only visibility map

pg_attribute

Field	Description	Status	Comment
attcollation	The defined collation of the column, or zero if the is not of a collatable data type	New	
attacl	Column-level access privileges, if any have been granted specifically on this column	New	
attoptions	Attribute-level options, as "keyword=value" strings.	New	
attfdwoptions	Attribute-level foreign data wrapper options, as "keyword=value" strings.	New	

gp_segment_configuration

Field	Description	Status	Comment
datadir	Segment instance data directory.	New	No more need to join with the table pg_filespace_entry (removed)
replication_port	The TCP port the file block replication process is using to keep primary and mirror segments synchronized.	Removed	

The new field datadir is useful and simplify the ident of directory storing the segment

pg_authid

Field	Description	Status	Comment
rolreplication	Role is a replication role.	New	If Yes, this role can initiate streaming replication and set/unset the system backup mode using pg_start_backup and pg_stop_backup.
rolcreaterexthdfs	Privilege to create read external tables with the gphdfs protocol	Removed	
rolcreatewexthdfs	Privilege to create write external tables with the gphdfs protocol	Removed	
rolconfig	Session defaults for server configuration parameters	Removed	Role config is now stored in pg_db_role_setting

- **CREATE ROLE** and **ALTER ROLE** SQL commands have new parameters REPLICATION/NOREPLICATION

  These clauses determine whether a role is allowed to initiate streaming replication or put the system in and out of backup mode. A role having the REPLICATION attribute is a very highly privileged role, and should only be used on roles actually used for replication. If not specified, NOREPLICATION is the default .

pg_database

Field	Description	Status	Comment
datcollate	LC_COLLATE for this database.	New	
datctype	LC_CTYPE for this database.	New	
datminmxid	A Multixact ID is used to support row locking by multiple transactions. All multixact IDs before this one have been replaced with a transaction ID in this database.	New	This is used to track whether the database needs to be vacuumed in order to prevent multixact ID wraparound or to allow pg_multixact to be shrunk. It is the minimum of the per-table pg_class.relminmxid values.
datconfig	Session defaults for user-settable server configuration parameters	Removed	Database config is now stored in pg_db_role_setting

- **CREATE DATABASE** SQL command has new parameters LC_COLLATE and LC_CTYPE to specify the collation order and character classification for the new database.
- **createdb** command-line utility has new options -l (--locale), --lc-collate, and --lc-ctype to specify the locale and character classification for the database

pg_authid - pg_database: example of changes

- From GPDB6, **pg_db_role_setting** records the default values of server configuration settings for each role and database combination.

GPDB4 or GPDB5

```
select rolname, rolconfig
from pg_authid
Where rolname = 'gpadmin';


 rolname |                  rolconfig
---------+-----------------------------------------------
 gpadmin | {search_path=public,statement_timeout=0…}
```

GPDB6

```
select rolname, setconfig
from pg_authid
Left outer join pg_db_role_setting on oid = setrole
Where rolname = 'gpadmin';

 rolname |              setconfig
---------+-----------------------------------------------
 gpadmin | {search_path=public,statement_timeout=0…}
```

GPDB4 or GPDB5

```
select datname, datconfig
from pg_database
where datname = 'dssprod'


 datname |                   datconfig
---------+-----------------------------------------------
 dssprod | {"search_path=dba_adm, …",optimizer=on}
```

GPDB6

```
select datname, setconfig
from pg_database
Left outer join pg_db_role_setting on oid = setdatabase
Where datname = 'dsspoc';

 datname |             setconfig
---------+-----------------------------------------------
 dsspoc  | {"search_path=dba_adm,…",optimizer=on}
```

pg_proc

Field	Description	Status	Comment
protransform	Calls to this function can be simplified by this other function	New	
proiswindow	Function is a window function	Renamed	Old name = proiswin
proleakproof	The function has no side effects. No information about the arguments is conveyed except via the return value.	New	Any function that might throw an error depending on the values of its arguments is not leak-proof.
proexeclocation	Where the function executes when it is invoked	New	m - master only a - any segment instance s - all segment instances.

- **CREATE FUNCTION** SQL command has a new parameter EXECUTE ON . It specifies where (master or segment instance) a function executes when it is invoked during the query execution process.
  EXECUTE ON ANY (the default) indicates that the function can be executed on the master, or any segment instance, and it returns the same result regardless of where it is executed. Greenplum determines where the function executes.
  EXECUTE ON MASTER indicates that the function must execute only on the master instance.
  EXECUTE ON ALL SEGMENTS indicates that the function must execute on all primary segment instances, but not the master, for each invocation. The overall result of the function is the UNION ALL of the results from all segment instances.

NEW: gp_toolkit.gp_resgroup_status_per_host

- This view allows to see current memory (MBs) and CPU usage and allocation for each resource group on a per-host basis.

Field	Description	Comment
rsgname	The name of the resource group.	
groupid	OID of the resource group	
hostname		
cpu	Real-time CPU usage of the RG on the host.	
memory_used	Real-time memory usage of the RG on the host.	It includes RG fixed, RG shared memory and global shared memory used by the RG
memory_available	Unused fixed and shared memory for the RG available on the host.	not include available RG global shared memory
memory_quota_used	Real-time fixed memory usage for the RG on the host.	
memory_quota_available	Fixed memory available to the RG on the host.	
memory_quota_proposed	Total fixed memory allotted to the RG on the host.	
memory_shared_used	Group shared memory used by the RG on the host	
memory_shared_granted	Portion of group shared memory allotted to the RG on the host.	
memory_shared_proposed	Total amount of group shared memory requested by the RG on the host	

NEW: gp_toolkit.gp_resgroup_status_per_segment

- Same as gp_toolkit.gp_resgroup_status_per_host, but on a per-host and per-segment basis.

Field	Description	Comment
rsgname	The name of the resource group.	
groupid	OID of the resource group	
hostname		
segment_id		
cpu	Real-time CPU usage of the RG by the segment	
memory_used	Real-time memory usage of the RG by the segment	It includes RG fixed, RG shared memory and global shared memory used by the RG
memory_available	Unused fixed and shared memory for the RG available by the segment.	not include available RG global shared memory
memory_quota_used	Real-time fixed memory usage for the RG by the segment.	
memory_quota_available	Fixed memory available to the RG by the segment	
memory_quota_proposed	Total fixed memory allotted to the RG by the segment	
memory_shared_used	Group shared memory used by the RG by the segment	
memory_shared_granted	Portion of group shared memory allotted to the RG by the segment	
memory_shared_proposed	Total amount of group shared memory requested by the RG by the segment	

Greenplum 6

Parameters

New parameters - 1/5

Parameter	Default	Comment
autovacuum_multixact_freeze_max_age		Not documented
autovacuum_work_mem		Not documented
bonjour		Not documented
create_restartpoint_on_ckpt_record_replay		Not documented
default_transaction_deferrable		Read-only parameter
dynamic_shared_memory_type		Not documented
effective_io_concurrency		Not documented
enable_indexonlyscan		Not documented
enable_material		Not documented
event_source		Not documented
exit_on_error		Not documented
external_pid_file		Not documented
geqo_seed		Not documented
gin_fuzzy_search_limit		Not documented

New parameters - 2/5

Parameter	Default	Comment
gp_enable_global_deadlock_detector	**off**	If off (the default), Greenplum Database executes concurrent update and delete operations on a heap table serially. If on, concurrent updates are permitted and the Global Deadlock Detector determines when a deadlock exists, and breaks the deadlock by cancelling one or more backend processes associated with the youngest transaction(s) involved.
gp_enable_minmax_optimization		Not documented
gp_fts_mark_mirror_down_grace_period		Not documented
gp_fts_probe_retries	**5**	Specifies the number of times the fault detection process (ftsprobe) attempts to connect to a segment before reporting segment failure.
gp_global_deadlock_detector_period	**120 sec**	Specifies the executing interval (in seconds) of the global deadlock detector backend process.
hot_standby		Not documented
hot_standby_feedback		Not documented
huge_pages		Not documented
lo_compat_privileges		Not documented

New parameters - 3/5

Parameter	Default	Comment
lock_timeout	**0 msec**	Abort any statement that waits longer than the specified number of milliseconds while attempting to acquire a lock on a table, index, row, or other database object. The time limit applies separately to each lock acquisition attempt. The limit applies both to explicit locking requests (such as LOCK TABLE, or SELECT FOR UPDATE without NOWAIT) and to implicitly-acquired locks. If log_min_error_statement is set to ERROR or lower, Greenplum Database logs the statement that timed out. A value of zero (the default) turns off this lock wait monitoring.
log_file_mode		Not documented
max_pred_locks_per_transaction		Not documented
max_replication_slots		Not documented
max_standby_archive_delay		Not documented
max_standby_streaming_delay		Not documented
max_wal_senders		Not documented
max_worker_processes		Not documented
quote_all_identifiers		Not documented
restart_after_crash		Not documented
segment_size		Not documented

New parameters - 4/5

Parameter	Default	Comment
session_preload_libraries		Not documented
ssl_ca_file		Not documented
ssl_cert_file		Not documented
ssl_crl_file		Not documented
ssl_ecdh_curve		Not documented
ssl_key_file		Not documented
ssl_prefer_server_ciphers		Not documented
synchronous_standby_names		Not documented
temp_file_limit		Not documented
temp_tablespaces	***Empty***	temp_tablespaces specifies tablespaces in which to create temporary objects (temp tables and indexes on temp tables) when a CREATE command does not explicitly specify a tablespace.
trace_recovery_messages		Not documented
track_functions		Not documented
track_io_timing		Not documented
transaction_deferrable		Not documented

New parameters - 5/5

Parameter	Default	Comment
vacuum_defer_cleanup_age		Not documented
vacuum_freeze_table_age		Not documented
vacuum_multixact_freeze_min_age		Not documented
vacuum_multixact_freeze_table_age		Not documented
wal_block_size		Not documented
wal_keep_segments		Not documented
wal_level		Not documented
wal_log_hints		Not documented
wal_receiver_timeout		Not documented
wal_segment_size		Not documented
wal_sender_timeout		Not documented

Changed parameters - 1/2

Parameter	Status	Comment
bytea_output	**Default value**	Old default value: **escape** New default value: **hex**
debug_pretty_print	**Default value**	Old default value: **off** New default value: **on**
effective_cache_size	**Default value**	Old default value: **512MB** New default value: **16GB**
gp_recursive_cte	**Renamed**	Replacing gp_recursive_cte_prototype
gp_workfile_compress	**Renamed**	Replacing gp_workfile_compress_algorithm When workfile compression is enabled, Greenplum Database uses Zstandard compression.
log_rotation_size	**Default value**	Old default value: **0** New default value: **1GB**
temp_buffers	**Default value**	Old default value: **1024** New default value: **32MB**
track_activity_query_size	**Renamed**	Replacing pgstat_track_activity_query_size
unix_socket_directories	**Default value**	Old default value: New default value: **/tmp**
unix_socket_permissions	**Default value**	Old default value: **511** New default value: **0777**

Changed parameters - 2/2

Parameter	Status	Comment
optimizer_force_multistage_agg	**Default value**	Old default value: **on** New default value: **off**
wal_buffers	**Default value**	Old default value: **256kB** New default value: **4000kB**

Impact of optimizer_force_multistage_agg

optimizer_force_multistage_agg was "on" in GPDB 5, but was set to "off".
The context of this change is that forcing multi-stage aggregates affects OLTP workloads.

GPDB 5

```
demo=# explain select count(*) from t1;
                                      QUERY PLAN
---------------------------------------------------------------------------------------
 Aggregate  (cost=0.00..431.00 rows=1 width=8)
   ->  Gather Motion 2:1  (slice1; segments: 2)  (cost=0.00..431.00 rows=1 width=8)
         ->  Aggregate  (cost=0.00..431.00 rows=1 width=8)
               ->  Table Scan on t1  (cost=0.00..431.00 rows=1 width=1)
 Optimizer status: PQO version 3.32.0
```

1st Aggregate on segments

GPDB 6

```
demo=# explain select count(*) from t1;
                                      QUERY PLAN
---------------------------------------------------------------------------------------
 Aggregate  (cost=0.00..431.00 rows=1 width=8)
   ->  Gather Motion 2:1  (slice1; segments: 2)  (cost=0.00..431.00 rows=1 width=1)
         ->  Seq Scan on t1  (cost=0.00..431.00 rows=1 width=1)
 Optimizer: Pivotal Optimizer (GPORCA) version 3.65.0
```

All rows are moved to master and then agggregated

Workaround1: `Analyze the table`

Workaround2: `set optimizer_force_multistage_agg = on;`

Removed parameters - 1/2

Parameter	Status	Comment
add_missing_from	Removed	
custom_variable_classes	Removed	Gptext uses this parameter. But it is not anymore useful from GPDB6 http://docs-gptext-develop-staging.cfapps.io/320/topics/installing.html#gpconfig_gptext
explain_pretty_print	Removed	
gp_analyze_relative_error	Removed	
gp_backup_directIO*	Removed	
gp_connections_per_thread	Removed	
gp_email_*	Removed	Impact on customer sending emails from Greenplum (Vente Privée, SKY, …)
gp_enable_fallback_plan	Removed	
gp_enable_sequential_window_plans	Removed	Possible impact on upgrade at IVECO
gp_filerep_*	Removed	
gp_idf_deduplicate	Removed	
gp_hadoop_*	Removed	
gp_idf_deduplicate	Removed	
gp_max_csv_line_length	Removed	
gp_max_databases	Removed	

Removed parameters - 2/2

Parameter	Status	Comment
gp_max_filespaces	Removed	
gp_max_tablespaces	Removed	
gp_num_contents_in_cluster	Removed	
gp_workfile_checksumming	Removed	
gp_workfile_compress_algorithm	Renamed	Replaced by gp_workfile_compress_algorithm
krb_srvname	Removed	
log_count_recovered_files_batch	Removed	
max_fsm_*	Removed	
pgstat_track_activity_query_size	Renamed	Replaced by track_activity_query_size
regex_flavor	Removed	
replication_timeout	Removed	
ssl_renegotiation_limit	Removed	

Greenplum 6

Expansion

Expansion changes

- Greenplum 6 uses a new jump consistent hash algorithm to map hashed data values to Greenplum segments.

  The new algorithm ensures that, after new segments are added to the Greenplum 6 cluster, **only those rows that hash to the new segment need to be moved**.

  Greenplum 6 hashing has performance characteristics similar to earlier Greenplum releases, but should enable **faster database expansion**.

  Note that **the new algorithm is more CPU intensive** than the previous algorithm, so COPY performance may degrade somewhat on CPU-bound systems.

- The **`-D`** option is removed from the gpexpand utility. The expansion schema will be created in the postgres database.

- The **`gpstate`** utility has a new **`-x`** option, which displays details of an in-progress system expansion.

  `gpstate -s` and **`gpstate`** with no options specified also report if a system expansion is in progress.

Greenplum 6

Removed features

Removed features

Features	Type	Replaced by	Comment
gptransfer	Utility	**gpcopy**	Possible impact on upgrade
gpcrondump	Utility	**gpbackup**	Possible impact on upgrade
gpdbrestore	Utility	**gprestore**	Possible impact on upgrade
gpmfr	Utility	**gpbackup_manager**	
gphdfs	Ext table protocol	**pxf**	
gpseginstall	Utility	**N/A**	The binary has to be installed manually on each node (use of gpscp + gpssh) Cause: gpseginstall doesn't work with RPM
SNMP alerts or send email notifications		**???**	
Veritas NetBackup for backup		**N/A**	
use of **direct I/O** for backup		**N/A**	Direct/IO was previously used to bypass the buffering of memory within the file system cache
SSLv3	Security protocol	**N/A**	

Greenplum 6

Other items

Other changes inherited from PostgreSQL - 1

- The LOCK SQL command has an optional ONLY keyword (PostgreSQL 8.4). When specified, the table is locked without locking any tables that inherit from it.
- Support for user-defined I/O conversion casts. (PostgreSQL 8.4).

Other changes inherited from PostgreSQL - 2

- The SELECT and VALUES SQL commands support the SQL 2008 FETCH syntax (PostgreSQL 8.4). This clauss provides an alternative syntax for limiting the results returned by a query.

Before GPDB 6

```
[ WITH [ RECURSIVE1 ] with_query [, ...] ]
SELECT [ALL | DISTINCT [ON (expression [, ...])]]   * |
expression [[AS] output_name] [, ...]
[FROM from_item [, ...]]
[WHERE condition]
[GROUP BY grouping_element [, ...]]
[HAVING condition [, ...]]
[WINDOW window_name AS (window_specification)]
[{UNION | INTERSECT | EXCEPT} [ALL] select]
[ORDER BY expression [ASC | DESC | USING operator]
[NULLS {FIRST | LAST}] [, ...]]
[LIMIT {count | ALL}]
[OFFSET start]

[FOR {UPDATE | SHARE}
[OF table_name [, ...]]
[NOWAIT] [...]]
```

```
VALUES ( expression [, ...] ) [, ...]
[ORDER BY sort_expression [ASC | DESC | USING operator] [,
...]]
[LIMIT {count | ALL}]
[OFFSET start]
```

From GPDB 6

```
[ WITH [ RECURSIVE1 ] with_query [, ...] ]
SELECT [ALL | DISTINCT [ON (expression [, ...])]]   * |
expression [[AS] output_name] [, ...]
[FROM from_item [, ...]]
[WHERE condition]
[GROUP BY grouping_element [, ...]]
[HAVING condition [, ...]]
[WINDOW window_name AS (window_definition) [, ...] ]
[{UNION | INTERSECT | EXCEPT} [ALL | DISTINCT] select]
[ORDER BY expression [ASC | DESC | USING operator]
[NULLS {FIRST | LAST}] [, ...]]
[LIMIT {count | ALL}]
[OFFSET start [ ROW | ROWS ] ]
[FETCH { FIRST | NEXT } [ count ] { ROW | ROWS } ONLY]
[FOR {UPDATE | NO KEY UPDATE | SHARE | KEY SHARE}
[OF table_name [, ...]]
[NOWAIT] [...]]
```

```
VALUES ( expression [, ...] ) [, ...]
[ORDER BY sort_expression [ ASC | DESC | USING operator ] [,
...] ]
[LIMIT { count | ALL } ]
[OFFSET start [ ROW | ROWS ] ]
[FETCH { FIRST | NEXT } [count ] { ROW | ROWS } ONLY ]
```

Other changes inherited from PostgreSQL - 3

- INTERVAL Data Type Handling: PostgreSQL 8.4 improves the parsing of INTERVAL literals to align with SQL standards.

Before GPDB 6

```
select INTERVAL '1' year;
 interval
----------
 00:00:00

select INTERVAL '1' month;
 interval
----------
 00:00:00

select INTERVAL '1’ day;
 interval
----------
 00:00:00

select INTERVAL '1' week;
   week
----------
 00:00:01
```

From GPDB 6

```
select INTERVAL '1' year;
 interval
----------
 1 year

select INTERVAL '1' month;
 interval
----------
 1 mon

select INTERVAL '1’ day;
 interval
----------
 1 day

select INTERVAL '1' week;
   week
----------
 00:00:01
```

Other changes inherited from PostgreSQL – 4

- Change of default value for the GUC bytea_output ➔ impact the output of bytea fields

Before GPDB 6

```
show bytea_output;
 bytea_output
--------------
 escape
```

```
create temp table t1
As select encode('password','base64') as pwd;

select pwd,decode(pwd,'base64') from t1;
```

```
      pwd       |  decode
----------------+----------
 cGFzc3dvcmQ=   | password
```

From GPDB 6

```
show bytea_output;
 bytea_output
--------------
 hex
```

```
create temp table t1
As select encode('password','base64') as pwd;

select pwd,decode(pwd,'base64') from t1;
```

```
      pwd       |        decode
----------------+--------------------
 cGFzc3dvcmQ=   | \x70617373776f7264
```

```
set bytea_output = 'escape';
select pwd,decode(pwd,'base64') from t1;
```

```
      pwd       |  decode
----------------+----------
 cGFzc3dvcmQ=   | password
```

Pivotal

Other changes inherited from PostgreSQL – 5

- Change of behavior for casting from text to timestamp

Before GPDB 6

```
select '99991231235959'::TIMESTAMP;
```

```
      timestamp
---------------------
 9999-12-31 23:59:59
```

From GPDB 6

```
select '99991231235959'::TIMESTAMP;
```

```
ERROR:  date/time field value out of range:
"99991231235959"
LINE 1: select '99991231235959'::TIMESTAMP;
               ^
HINT:  Perhaps you need a different "datestyle" setting.
```

Add a space between Date and Time

```
select '99991231 235959'::TIMESTAMP;
```

```
      timestamp
--------------------
 9999-12-31 23:59:59
```

Pivotal

ERROR: non-MVCC snapshots are not supported in index-only scans

- A count to check the number of invisible rows on an indexed table can fail:

```
demo=# SET gp_select_invisible=true;
SET
demo=#SELECT count(*) from pg_stat_last_operation;
ERROR:  non-MVCC snapshots are not supported in index-only scans (nodeIndexonlyscan.c:136)
```

- Cause: The error is expected. The reason it fails is because you happened to get a plan that asked for an index only scan.

- Solution: `Set enable_indexonlyscan = off;` before the count.

```
SET gp_select_invisible=true;
set enable_indexonlyscan to off;
SELECT count(*) from pg_catalog.pg_stat_last_operation;
```

ERROR: column "ctid" of relation "*****" does not exist

- In GPDB 4, in order to mitigate the performance issue of analyze, some customers collected statistics only on the field **ctid**.

- From GPDB6, it is not anymore possible to analyze the field ctid

```
demo=# analyze public.test_analyze(ctid);
ERROR:  column "ctid" of relation "test_analyze" does not exist
```

- Solution: As the performance of analyze was highly improved from GPDB5, the workaround used in GPDB4 is not anymore useful. So analyze the complete table

To check and test

- PL/pgSQL
  - The RETURN QUERY EXECUTE statement, which specifies a query to execute dynamically (PostgreSQL 8.4). ==> **have a look here** https://www.postgresql.org/docs/8.4/plpgsql-control-structures.html
  - Conditional execution using the CASE statement (PostgreSQL 8.4).

- Support for GIN index method (PostgreSQL (8.3).

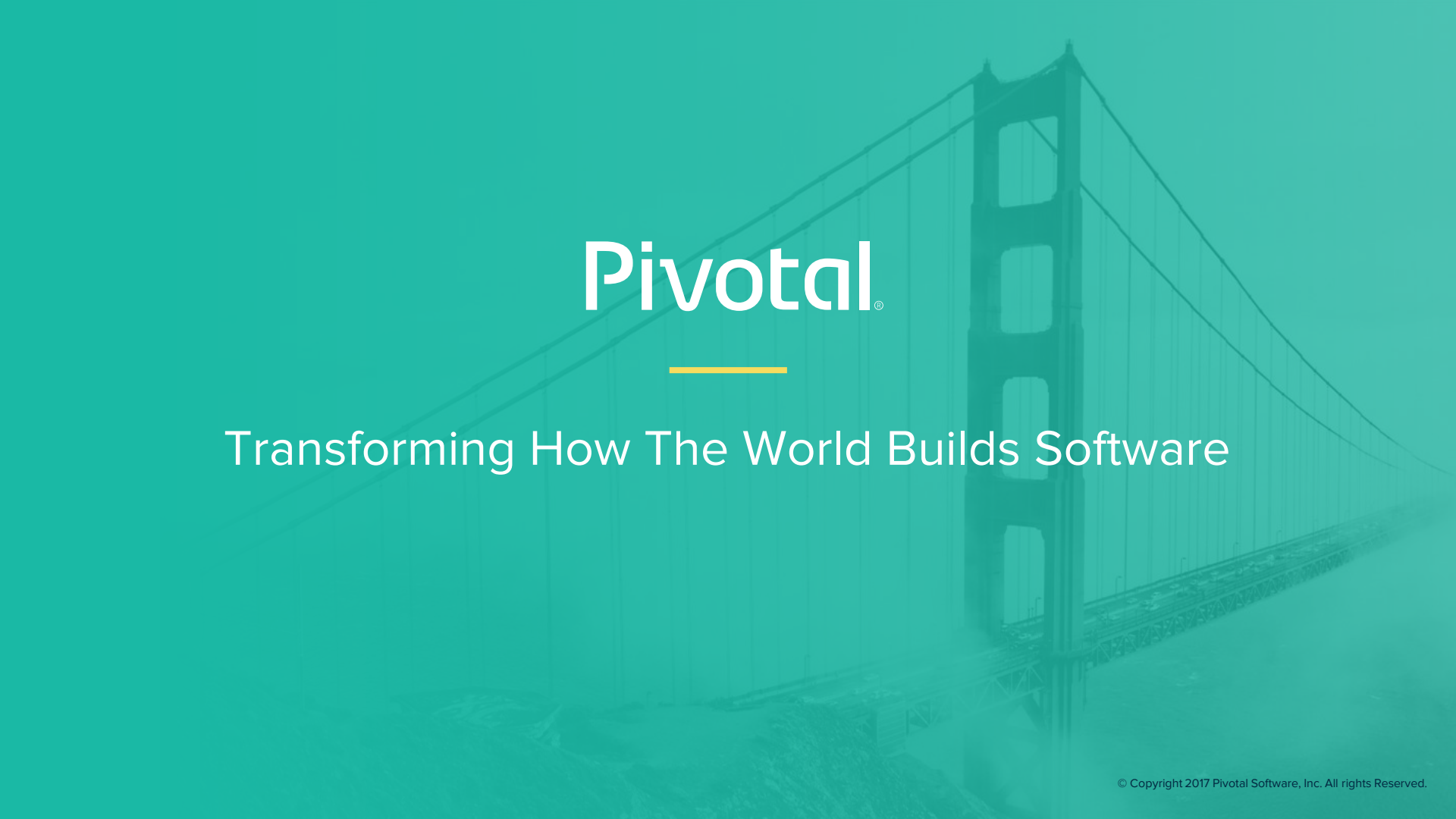
- DELETE, INSERT, and UPDATE supports the WITH clause, CTE (common table expression) (PostgreSQL 9.1).

To check and test

- Queries that use SELECT DISTINCT and UNION/INTERSECT/EXCEPT no longer necessarily return sorted output. Previously these queries always removed duplicate rows by using Sort/Unique processing. They now implement hashing to conform to behavior introduced in PostgreSQL 8.4; this method does not produce sorted output. If your application requires sorted output for these queries, alter the queries to use an explicit ORDER BY clause. Note that SELECT DISTINCT ON never uses hashing, so its behavior is unchanged from previous versions.

- Specifying the index name in the CREATE INDEX SQL command is now optional. Greenplum Database constructs a default index name from the table name and indexed columns.

- In Greenplum 6, a query on an external table with descendants will by default recurse into the descendant tables. This is a change from previous Greenplum Database versions, which never recursed into descendants. To get the previous behavior in Greenplum 6, you must include the ONLY keyword in the query to restrict the query to the parent table.



Transforming How The World Builds Software