PERCONA

Databases run better with Percona

FOSDEM 2024 - PGDAY- BUILD YOUR OWN POSTGRESQL DBA OUT OF AVAILABLE MYSQL DBAS

Date: 2024-02-04

Time: 16:00–16:50

Room: PostgreSQL Devroom UD2.120 (Chavanne)

Historically it has been able to find MySQL DBAs than their PostgreSQL counterparts. The growth in PostgreSQL has increased the demand for new DBAs. So why not convert some MySQL DBAs into competent PostgreSQL DBAs?

This session covers where you need to concentrate on guiding this conversion. There are many similarities between the two RDMS but this session covers where the 'pinch points' are that will require guidance. Starting with setting up a basic instance for the conversion candidates with a familiar-ish training database, we will proceed into differences in MVCC, Indexing, TOAST, and other divergent areas.

You can very quickly have a new PostgreSQL DBA that you have built yourself.

PERCONA

Build Your Own PostgreSQL DBA Out Of Available MySQL DBAs

Dave Stokes
@Stoker
David.Stokes@Percona.com

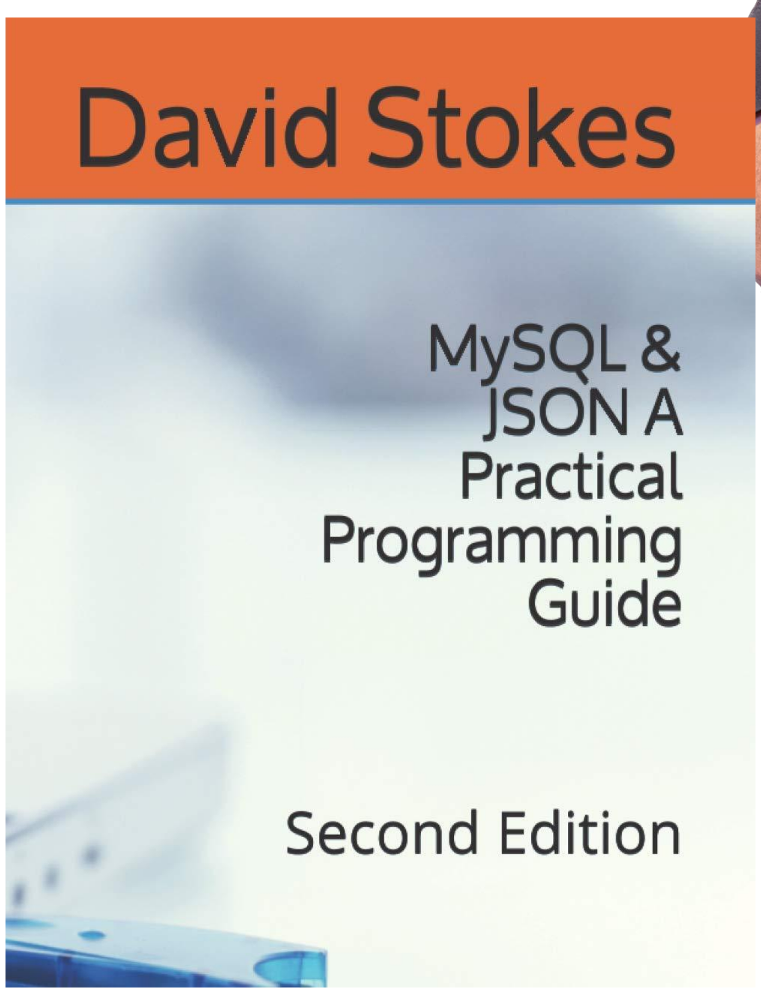
About Me!

Dave Stokes

Technology Evangelist

David.Stokes@Percona.com

@Stoker



Origin Story

2007 @ MySQL AB

Using Explain

Query tuning can be tough to learn

Hiring DBAs

Economics

Make versus Buy decision

https://www.investopedia.com/

A make-or-buy decision is an act of choosing between manufacturing a product in-house or purchasing it from an external supplier.

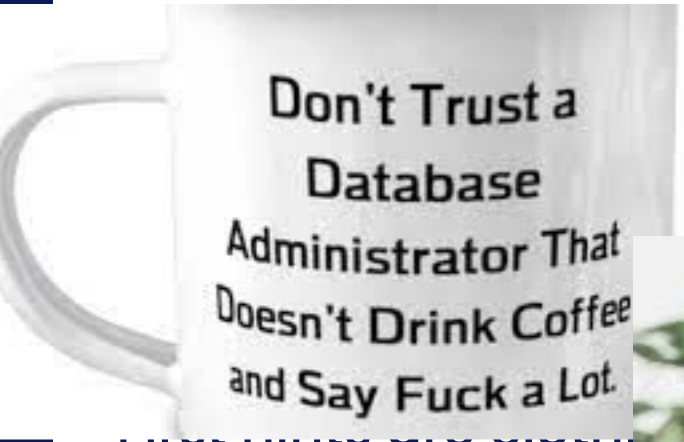
Make-or-buy decisions, like outsourcing decisions, speak to a comparison of the costs and advantages of producing in-house versus buying it elsewhere.

Why MySQL DBAs

- Available supply
- Basic knowledge of ‘DBA Skills’
- They are PG Curious
- Many similarities between the two
  - Make it easy to transition
  - Beware of pinch points
  - Show them the ‘goodies’
- ‘Cloud flight’

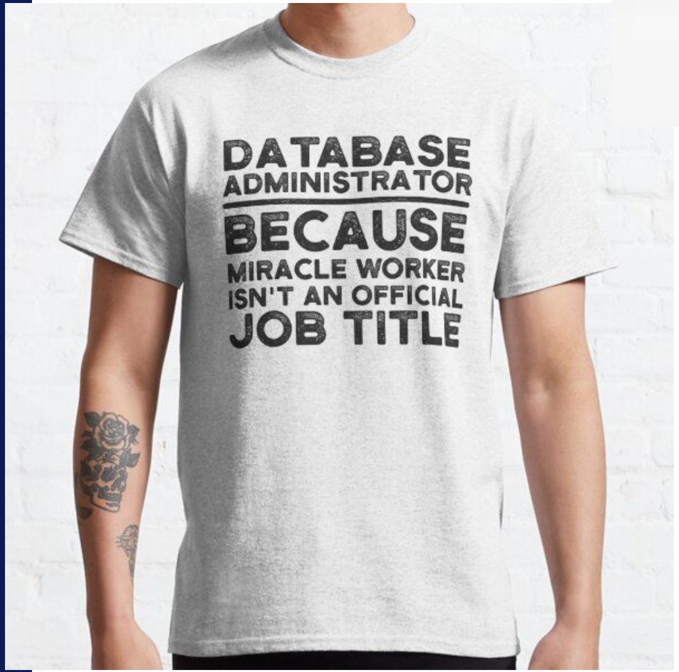
nize MySQL DBAs?

First hints are clothi


Don't Trust a Database Administrator That Doesn't Drink Coffee and Say Fuck a Lot.

BEING A DATABASE ADMINISTRATOR IS EASY... IT'S LIKE A WALK IN A PARK THAT LEADS TO A VOLCANO THAT SURROUNDS YOU IN MOLTEN LAVA

DATABASE ADMINISTRATOR
Someone who does precision guesswork based on unreliable data provided by those of questionable knowledge
See also wizard, magician


DATABASE ADMINISTRATOR BECAUSE MIRACLE WORKER ISN'T AN OFFICIAL JOB TITLE

PostgreSQL versus MySQL differences

Both:

Relational Database Management Systems
Open Source
Popular
Old enough to allowed to drink   (therefore seen as ‘not cool’ by some)

PostgreSQL:

Better SQL Standard Support
Governed by mailing list, consensus
Active community

MySQL:

‘Easier’
Governed (?) by Oracle
Active community

'The devil is in the details'

Ludwig Mies Van Der Rohe.

You found one!

So you find a likely MySQL DBA that you would like to convert. Congratulations!

You might mention that they will have:

- Better skills
- Cross training
- Enhanced job opportunities
- And the ability to now complain knowling about two databases!

So where do you start?

1. **Different approaches to same problems**
2. **New tools**
3. **The basics are still the basics**
   a. Backups/Restore
   b. Account administration
   c. Performance tuning
   d. Query tuning
4. **The really neat new stuff**
   a. Things like two JSON data types, MERGE, Indexes galore, ….
5. **The OMGHDWSHTPI2023* stuff**

*Oh My Goodness How Do We Still Have This Problem In 2023

First Steps

Build a simple PostgreSQL environment

First steps - Video Rental Database

Load whichever PG you want and get dvdrental.tar from https://www.postgresqltutorial.com/wp-content/uploads/2019/05/dvdrental.zip

```
$sudo su - postgres
$psql
postgresql=# CREATE DATABASE dvdrental;
postgresql=# exit;
$pgrestore -U postgres -d dvdrental dvdrental.tar
```

BLOCKBUSTER VIDEO
BLOCKBUSTER VIDEO
BLOCKBUSTER VIDEO
BLOCKBUSTER VIDEO
BLOCKBUSTER VIDEO
BLOCKBUSTER VIDEO
NEW PRICES AS ADVERTISED
THE END OF LATE FEES

First steps

Load whichever PG you want and get dvdrental.tar from https://www.postgresqltutorial.com/wp-content/uploads/2019/05/dvdrental.zip

```
$sudo su - postgres
$psql
postgresql=# CREATE DATABASE dvdrental;
postgresql=# exit;
$pgrestore -U postgres -d dvdrental dvdrental.tar
```

(still as user ‘postgres’)

```
$createuser –interactive -s <user>
```

The -s is for superuser

Yup this is dangerous as superuser bypasses some checks but remember you candidate is an experienced DBA (or should be)

PERCONA

Back in the <user> account

```
$psql -d dvdrental
dvdrental=#
```

What this provides

MySQL has used the Sakila database in documentation, training, blogs, and etcetera for decades.

Using the dvdrental database provides a familiar-ish database for learning

Easy to use, lots of things to join, and for relational basics

Ohhh, that is different

No SHOW TABLES?!?!?!?!

```
test=# SHOW TABLES;
ERROR:  unrecognized configuration parameter "tables"
test=#
```

\d commands

```
dvdrental=# \dt
             List of relations
 Schema |      Name      | Type  |  Owner
--------+----------------+-------+----------
 public | actor          | table | postgres
 public | address        | table | postgres
 public | category       | table | postgres
 public | city           | table | postgres
 public | country        | table | postgres
 public | customer       | table | postgres
 public | film           | table | postgres
 public | film_actor     | table | postgres
 public | film_category  | table | postgres
 public | inventory      | table | postgres
 public | language       | table | postgres
 public | payment        | table | postgres
 public | rental         | table | postgres
 public | staff          | table | postgres
 public | store          | table | postgres
(15 rows)
```

PERCONA

Cheat Sheets are okay

PERCONA

There is no `SHOW CREATE TABLE` either

```
dvdrental=# show create table actor;
ERROR:  syntax error at or near "create"
LINE 1: show create table actor;
             ^
dvdrental=# \d actor;
```

```
                                                Table "public.actor"
   Column    |            Type             | Collation | Nullable |                 Default
-------------+-----------------------------+-----------+----------+-----------------------------------------
 actor_id    | integer                     |           | not null | nextval('actor_actor_id_seq'::regclass)
 first_name  | character varying(45)       |           | not null |
 last_name   | character varying(45)       |           | not null |
 last_update | timestamp without time zone |           | not null | now()
Indexes:
    "actor_pkey" PRIMARY KEY, btree (actor_id)
    "idx_actor_last_name" btree (last_name)
Referenced by:
    TABLE "film_actor" CONSTRAINT "film_actor_actor_id_fkey" FOREIGN KEY (actor_id) REFERENCES actor(actor_id) ON UPDA
Triggers:
    last_updated BEFORE UPDATE ON actor FOR EACH ROW EXECUTE FUNCTION last_updated()
```

Simple queries work as expected

```
dvdrental=# SELECT *
              FROM actor
              ORDER BY last_name, first_name
              LIMIT 10;
 actor_id | first_name | last_name |      last_update
----------+------------+-----------+------------------------
       58 | Christian  | Akroyd    | 2013-05-26 14:47:57.62
      182 | Debbie     | Akroyd    | 2013-05-26 14:47:57.62
       92 | Kirsten    | Akroyd    | 2013-05-26 14:47:57.62
      118 | Cuba       | Allen     | 2013-05-26 14:47:57.62
      145 | Kim        | Allen     | 2013-05-26 14:47:57.62
      194 | Meryl      | Allen     | 2013-05-26 14:47:57.62
       76 | Angelina   | Astaire   | 2013-05-26 14:47:57.62
      112 | Russell    | Bacall    | 2013-05-26 14:47:57.62
      190 | Audrey     | Bailey    | 2013-05-26 14:47:57.62
       67 | Jessica    | Bailey    | 2013-05-26 14:47:57.62
(10 rows)
```

Simple backup

```
$ pg_dump dvdrental > backup.sql
```

- pg_dump is the name of the 'backup' program
- dvdrental is name of the database to be backed up
- Dumping the output to file backup.sql

Equivalent to mysqldump

Simple restore

```
$ sudo su - postgres
$ psql
(psql 14.3 (Ubuntu 2:14.3-3-focal))
Type “help” for help.
dvdrental=# CREATE DATABASE newdvd;
dvdrental=# \q
$ ^d
```

```
$ psql -d newdvd -f backup.sql
```

Cheat Sheet

\c dbname	Switch connection to a new database
\l	List available databases
\dt	List available tables
\d table_name	Describe a table such as a column, type, modifiers of columns, etc.
\dn	List all schemes of the currently connected database
\df	List available functions in the current database
\dv	List available views in the current database
\du	List all users and their assign roles
SELECT version();	Retrieve the current version of PostgreSQL server
\g	Execute the last command again
\s	Display command history
\s filename	Save the command history to a file
\i filename	Execute psql commands from a file
\?	Know all available psql commands
\h	Get helpEg:to get detailed information on ALTER TABLE statement use the \h ALTER TABLE
\e	Edit command in your own editor
\a	Switch from aligned to non-aligned column output
\H	Switch the output to HTML format
\q	Exit psql shell

Goodbye AUTO_INCREMENT, Hello SERIAL data type

Small Serial	2 bytes	1 to 32,767
Serial	4 bytes	1 to 2,147,483,647
Big Serial	8 bytes	1 to 9,223,372,036,854,775,807

Yup, MySQL has a SERIAL (`BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE)` but it is a) not widely used, b) will end up creating two indexes if also declared as the PRIMARY KEY.

PERCONA

We start sneaking in sequences!

```
dvdrental=# CREATE SCHEMA test;
CREATE SCHEMA
dvdrental=# \c test
You are now connected to database "test" as user "percona".
test=# CREATE TABLE x (x SERIAL, y CHAR(20), z CHAR(20));
CREATE TABLE
test=# \d x

                             Table "public.x"
 Column |     Type      | Collation | Nullable |           Default
--------+---------------+-----------+----------+------------------------------
 x      | integer       |           | not null | nextval('x_x_seq'::regclass)
 y      | character(20) |           |          |
 z      | character(20) |           |          |
```

Demo

```
test=# INSERT INTO x (y,z) VALUES (100,200),(300,450);
INSERT 0 2
test=# SELECT * FROM x;
 x |          y          |          z
---+---------------------+---------------------
 1 | 100                 | 200
 2 | 300                 | 450
(2 rows)
```

INSERT replies with the ***oid*** and the ***count***.
The ***`count`*** is the number of rows inserted or updated. ***`oid`*** is always 0

Values of ‘x’ generated by server

Table & Sequence created by create table

```
test=# \d
           List of relations
 Schema |  Name    |   Type     |  Owner
--------+----------+-----------+---------
 public | x        | table     | percona
 public | x_x_seq  | sequence  | percona
```

Create a table and load it with data?!?!?!

```
test=# create table test1 as (select generate_series(1,100) as id);
SELECT 100
test=# \d test1
               Table "public.test1"
 Column |  Type   | Collation | Nullable | Default
--------+---------+-----------+----------+---------
 id     | integer |           |          |


test=# select * from test1 limit 5;
 id
----
  1
  2
  3
  4
  5
(5 rows)
```

Fun with *wrapping* sequences

```
test=# create sequence wrap_seq as int minvalue 1 maxvalue 2 CYCLE;
CREATE SEQUENCE
test=# select NEXTVAL('wrap_seq');
 nextval
---------
       1
(1 row)

test=# select NEXTVAL('wrap_seq');
 nextval
---------
       2
(1 row)

test=# select NEXTVAL('wrap_seq');
 nextval
---------
       1
(1 row)

test=# select NEXTVAL('wrap_seq');
 nextval
---------
       2
(1 row)
```

Checking the details on sequences

```
test=# \d order_id;
                          Sequence "public.order_id"
  Type  | Start | Minimum |       Maximum       | Increment | Cycles? | Cache
--------+-------+---------+---------------------+-----------+---------+------
 bigint |  1001 |       1 | 9223372036854775807 |         1 | no      |

test=# \d wrap_seq;
                    Sequence "public.wrap_seq"
  Type   | Start | Minimum | Maximum | Increment | Cycles? | Cache
---------+-------+---------+---------+-----------+---------+-------
 integer |     1 |       1 |       2 |         1 | yes     |     1
```

Sticking Points

Where you need to guide converts

Explaining EXPLAIN - MySQL edition

```
SQL > EXPLAIN SELECT Name FROM City WHERE District='Texas' ORDER BY Name\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: City
   partitions: NULL
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 4188
     filtered: 10
        Extra: Using where; Using filesort
1 row in set, 1 warning (0.0011 sec)
Note (code 1003): /* select#1 */ select `world`.`city`.`Name` AS `Name` from `world`.`city
(`world`.`city`.`District` = 'Texas') order by `world`.`city`.`Name`
```

Learning new format

```
test=# EXPLAIN (ANALYZE) SELECT 1 FROM t2 WHERE ID=101;    #NO Index
                                          QUERY PLAN
-----------------------------------------------------------------------------------------------
 Seq Scan on t2  (cost=0.00..1693.00 rows=1 width=4) (actual time=0.019..5.641 rows=1 loops=1)
   Filter: (id = 101)
   Rows Removed by Filter: 99999
 Planning Time: 0.054 ms
 Execution Time: 5.658 ms
(5 rows)
test=# EXPLAIN (ANALYZE) SELECT 1 FROM t1 WHERE ID=101;    #YES Index
                                                   QUERY PLAN
----------------------------------------------------------------------------------------------------
-------------
 Index Only Scan using t1_pkey on t1  (cost=0.29..4.31 rows=1 width=4) (actual time=0.090..0.091
rows=1 loops=1)
   Index Cond: (id = 101)
   Heap Fetches: 0
 Planning Time: 0.469 ms
 Execution Time: 0.110 ms
```

This is a good comparison of timings

Options in parens new to a MySQL DBA

And no YAML or XML output

Learning to read the output of EXPLAIN

```
dvdrental=# explain SELECT title, first_name, last_name
dvdrental-# FROM film f
dvdrental-# INNER JOIN film_actor fa ON f.film_id=fa.film_id
dvdrental-# INNER JOIN actor a ON fa.actor_id=a.actor_id;
                        QUERY PLAN
------------------------------------------------------------------------
 Hash Join  (cost=83.00..196.65 rows=5462 width=28)
   Hash Cond: (fa.actor_id = a.actor_id)
   ->  Hash Join  (cost=76.50..175.51 rows=5462 width=17)
         Hash Cond: (fa.film_id = f.film_id)
         ->  Seq Scan on film_actor fa  (cost=0.00..84.62 rows=5462 width=4)
         ->  Hash  (cost=64.00..64.00 rows=1000 width=19)
               ->  Seq Scan on film f  (cost=0.00..64.00 rows=1000 width=19)
   ->  Hash  (cost=4.00..4.00 rows=200 width=17)
         ->  Seq Scan on actor a  (cost=0.00..4.00 rows=200 width=17)
(9 rows)
```

PERCONA

Items to calmly discuss

Sequences

Materialized Views

EXPLAIN

Connecting to a process not a thread, the use of connection poolers

Vacuum (please let them know about autovacuum upfront)

Toast

Wrap around XIDs

There are lots of things to discover for a converting MySQL DBA

```
SELECT
  fa.actor_id,
  SUM(length) FILTER (WHERE rating = 'R'),
  SUM(length) FILTER (WHERE rating = 'PG')
FROM film_actor AS fa
LEFT JOIN film AS f
  ON f.film_id = fa.film_id
GROUP BY fa.actor_id
```

Some reading

https://www.youtube.com/watch?v=S7jEJ9o9o2o

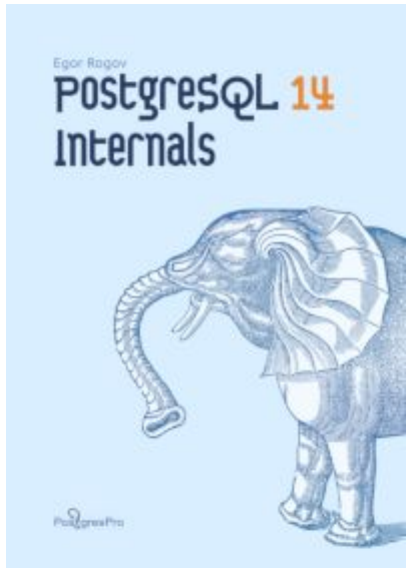
https://www.highgo.ca/2021/03/20/how-to-check-and-resolve-bloat-in-postgresql/

https://onesignal.com/blog/lessons-learned-from-5-years-of-scaling-postgresql/

https://www.postgresql.org/docs/

https://www.scalingpostgres.com/

https://psql-tips.org/psql_tips_all.html



Postgresql for MySQL DBAs videos

https://www.youtube.com/watch?v=S7jEJ9o9o2o&list=PLWhC0zeznqkmGAJDjVZu6zNsElQgYm6Rl

Long version of PostgreSQL For MySQL DBAs presentation

https://www.youtube.com/watch?v=S7jEJ9o9o2o&list=PLWhC0zeznqkmGAJDjVZu6zNsElQgYm6Rl

https://speakerdeck.com/stoker/percona-live-2023-postgresql-for-mysql-dbas

PERCONA

Innovate freely with highly available and reliable production PostgreSQL

Try Percona software:

- ➔ Percona Distribution for Postgres
- ➔ Percona Operator for PostgreSQL
- ➔ Percona Monitoring and Management (PMM)

We have a TDE solution looking for testers!

- ➔ **github.com/Percona-Lab/postgresql-tde**

Ask questions and leave your feedback:

- ➔ percona.community
- ➔ forums.percona.com
- ➔ github.com/percona

PERCONA
Distribution for PostgreSQL

PERCONA
Kubernetes Operators

PERCONA
Monitoring and Management

Thank You!

David.Stokes@Percona.com
@Stoker
Speakerdeck.com/Stoker