On Rabbits and Elephants

(or Using RabbitMQ to stream data events across PostgreSQL Databases)

Don't try this at home.

(try it at work or at least on your laptop)

Use Cases

- Adding complexity to your environment
- Writing distributed map reduce algorithms in Erlang
- Ensuring data inconsistency between canonical data sources and data warehouses
- Impressing your hipster Ruby programmer friends
- Replace your Slony replication setup

What is RabbitMQ?

(don’t let the cloud based marketing deter you)

What is PostgreSQL?

(just kidding)

Doesn't speak elephant.

Doesn't speak rabbit.

Step-By-Step Instructions

1. Install your Erlang on the machine you into intend to run your RabbitMQ broker on. My preference is for Erlang R14b2, which almost makes me think they name Erlang releases after Star Wars characters, but they don't. Erlang is not as fun as Debian is. It doesn't need to be a beefy machine but it should have enough ram that if you don't have an active consumer it won't run out of memory.

2. Install RabbitMQ. This is a multistep process involving black magic. Depending on your distribution you can either download a package, or if your *nix impaired there is a windows installer too. Oh and don't forget to install the management plugin. You can download the management plugin from the RqbbitMQ site pretty easy. After you have downloaded it you put it in the RabbitMQ plugins directory. Kind of makes sense doesn't it?

3. Go to the store and buy RabbitMQ in Action and RabbitMQ in Detail. What do you mean you can't? No pre-order either? Well there are plenty of good blogs on the intarwebs that have good information such as *cough* mine. Seriously though if you are crazy enough to follow my advice and get this far out of actual interest in the subject, make sure you verse yourself in the operation of and the flexibility that is RabbitMQ.

4. Install one of the following: pl/Python, pl/Perl, pl/Ruby, pl/PHP, pl/PGSQL and pg_amqp, or write your own damn addon in c using librabbitmq and then distribute it using the very cool pgxn service. Buoy only need this on the databases your going to be writing triggers on. Are you still reading this? Seriously? I would have thought I had lost you by now. Either that or you are skipping ahead because I should have changed the slide by now. It could be that Selena is operating the slides, in which case I have no control over these slides which is a terrifying thought. You see there is very little content on the slides, a real lack of substance. If I stay on any one slide for too long then you'll realize I really don't know what I am talking about and then I'd not be asked back to give one of the most important talks of the conference.

5. It's almost time to write your publisher, but don't get too far ahead of yourself. You first need to make sure that you have figured out a routing paradigm to use. Don't you like that word? Paradigm. I hear it in my head as para dig'um. Anyway, back to routing paradigms. Depending on your use case you will have the ability to broadcast your events to a whole slew of databases that care about what you have to send, or just one. You can implement a 1-to-1 type of scenario where a "direct" exchange is setup and your data will be queued in a way that no message is duplicated to your client and there is transactional guarantees in RabbitMQ. You probably don't want that thou, as you're a fly -by-the-seat-of-your-pants type of person, aren't you? For that you'll want to turn on noAck mode in your consumers. Rabbit will just fling your messages to your consumers as fast as possible, just like an ape at a zoo and his favorite projectile.

6. This is where things get fun,. We are going to quite. Stored procedure in the language of your choice and have it publish to RabbitMQ using an exchange and a routing key, now at this point if you are still reading this, I have to question your sanity. This was a one off gag slide to make people at the conference think that I wax going to read all of this. Anyway, wirte your function that publishes and you are almost done. No seriously. Well not quite done because you still need the part that calls this function and then the consumer app that reads the data from RabbitMQ and then doe the actions in PostgreSQL that you want it to do.

7. Now, basically you're going to add after insert, update, delete triggers to the tables you care about. I'm. Not going to go into detail about how to do that since you are at a PostgreSQL. Conference and should know how to do that already. Instead I am going to assume you know what I am writing about and say that in these triggers you will want to call the function we wrote in step 6 from these trigger events,

8. Go to http://github.com/gmr/On-Rabbits-and-Elephants/ and download the sample code and use that instead of all of this nonsense.

Use a Trigger

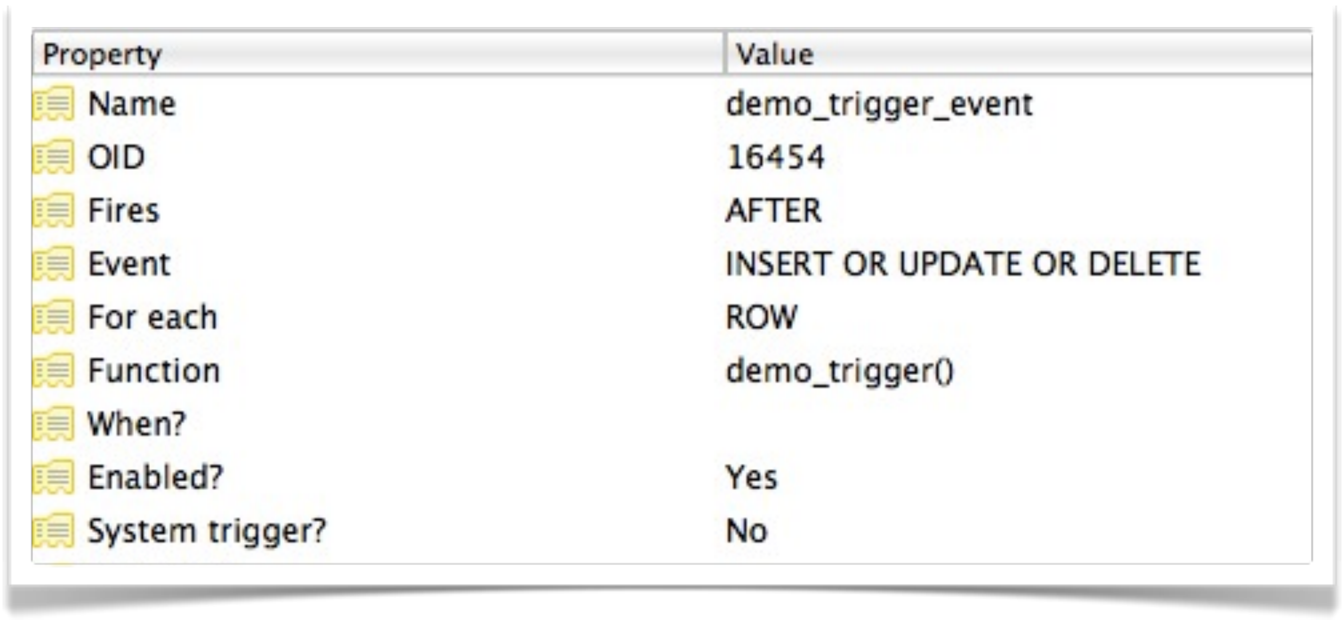
(benefit from the value of a trigger?)

Not that kind of trigger.

(who’s old enough to get the reference?)

Maybe this kind.

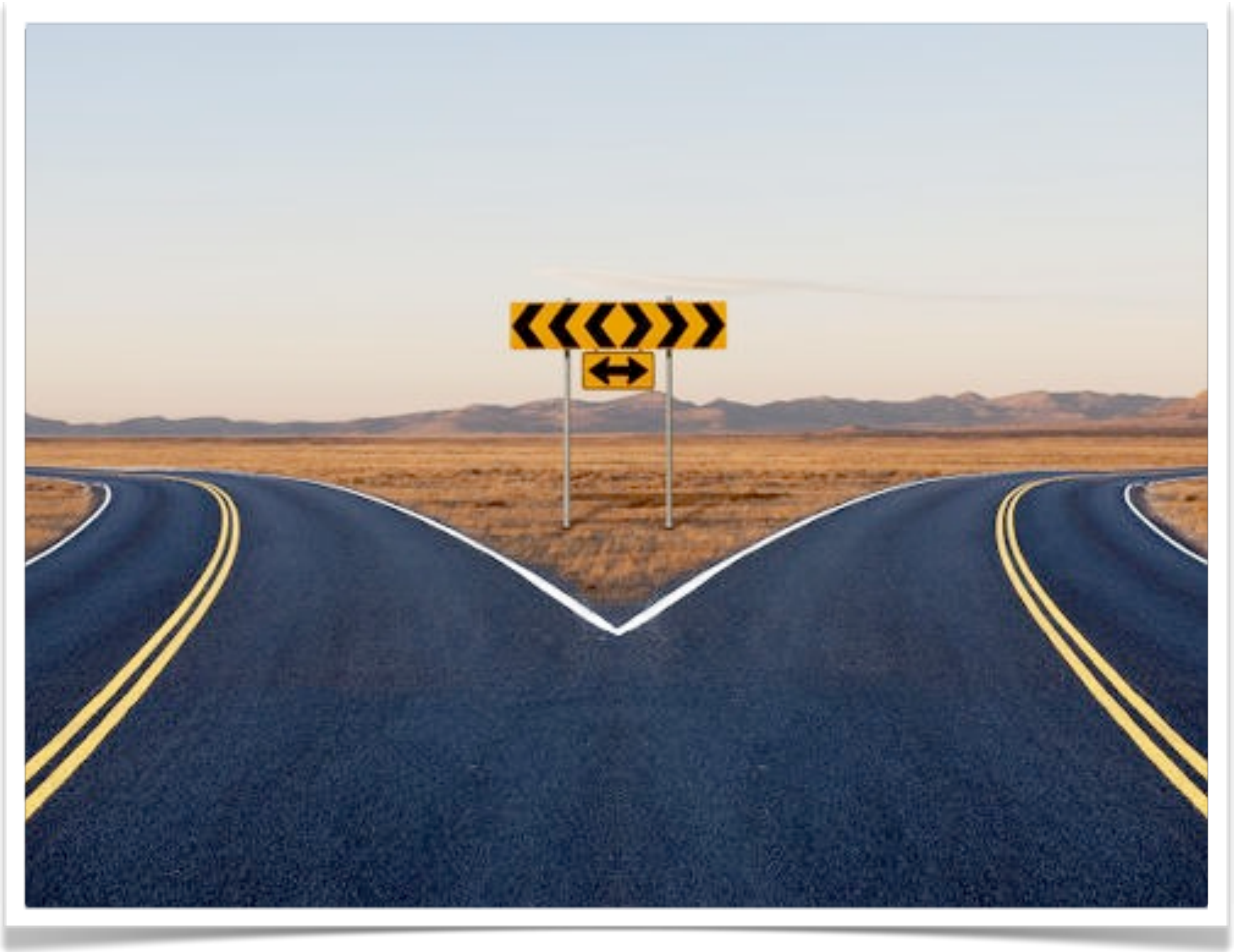
(btw: google image search for “footgun”)



Property	Value
Name	demo_trigger_event
OID	16454
Fires	AFTER
Event	INSERT OR UPDATE OR DELETE
For each	ROW
Function	demo_trigger()
When?	
Enabled?	Yes
System trigger?	No

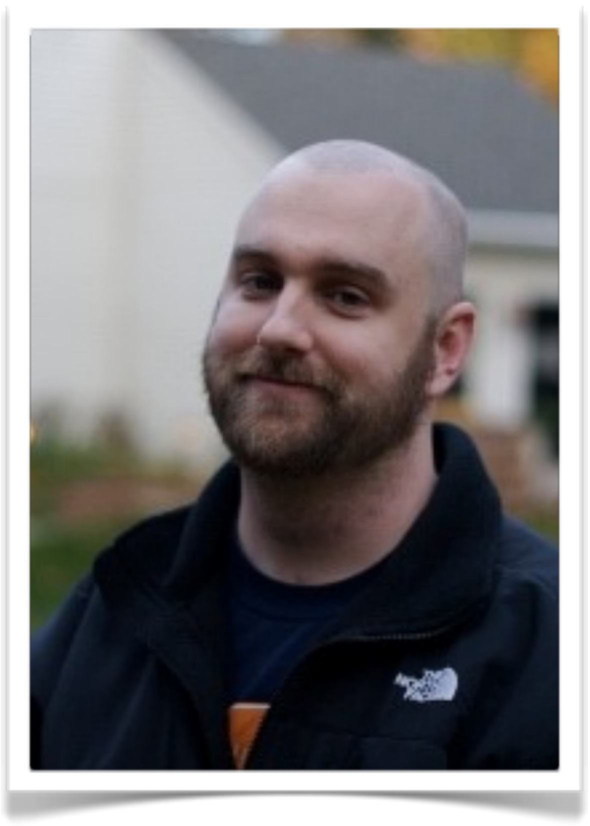
Really this kind.

(pretty boring, I know)



**“Two roads diverged in a yellow wood,
And sorry I could not travel both”**

(yes I just quoted Robert Frost in a lightning talk and not a high-school valedictorian speech)



Blame Him (for pg_amqp)

"But wait, there's more!"

Message Routing.

Examples

- Publish to a single consumer/destination
- Publish to every consumer listening to redundant_locations.*
- Publish to every consumer listening to *.my_awesome_table

Consumerism at its' finest.

“It’s Wafer Thin”

“Do I have to?”

(yes, go get the code and do it, it works, I promise. if it doesn’t I’ll refund the licensing fee for the code.)

Resources

- rabbitmq:
  http://www.rabbitmq.com

- pg_amqp:
  ```
  easy_install pgxnclient
  USE_PGXS=1 pgxn install pg_amqp
  ```
  http://pgxn.org/dist/pg_amqp/

- example code:
  https://github.com/gmr/On-Rabbits-and-Elephants

- follow me on twitter: @crad