Strategies to Edit Production Data

Julie Qiu

@jqiu25
Why We Edit
Production Data
Internal tools are not available.
Internal tools are not available.

Edge cases exist.
Internal tools are not available.

Edge cases exist.

Time-sensitive changes are needed.
I want to run this query

```
UPDATE products
SET name='julies-product'
WHERE id=1
```
What I actually ran

```
UPDATE products
SET name='julies-product'
```
MY GOD.
WHAT HAVE I DONE?
Let’s talk about strategies for safer editing.
Strategies for editing

Raw SQL
Local scripts
Existing server
Task runner
Script runner service
Things we’ll discuss

How the strategy works
What’s great
What’s not great
Strategy 1: Develop a Review Process for Manual Edits
I want to run this query

```
UPDATE products
SET name='julies-product'
WHERE id=1
```
Process for manual edits

1. Add record to a spreadsheet
# Manual Queries Spreadsheet

<table>
<thead>
<tr>
<th>Name</th>
<th>Julie Qiu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>1/19/2018</td>
</tr>
<tr>
<td>Description</td>
<td>Marketing team request</td>
</tr>
</tbody>
</table>
| Query        | UPDATE products
               SET name='juliies-product'
               WHERE id=1 |
| Reviewer Name| Codanda Appachu       |
| Reviewer Status | Approved / Changes Requested |
| Reviewer Comments | You spelled Julie wrong |

@jqiu25
Process for manual edits

1. Add record to a spreadsheet
2. Reviewer approves the query
Process for manual edits

1. Add record to a spreadsheet
2. Reviewer approves the query
3. Run the query
Run the query inside a transaction

\[
\text{BEGIN;}
\]

\[
\text{UPDATE} \ \text{products}
\]

\[
\text{SET} \ \text{name} = 'julies-product'
\]

\[
\text{WHERE} \ \text{id} = 1;
\]

\[
\text{END;}
\]
What’s great

• Easy to implement
What’s great

• Easy to implement
• Audit trail
What’s great

- Easy to implement
- Audit trail
- Promotes the right behaviors
<table>
<thead>
<tr>
<th>Benefits</th>
<th>Raw SQL</th>
<th>Local Scripts</th>
<th>Existing Server</th>
<th>Task Runner</th>
<th>Script Runner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Editing</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complex Logic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Long Scripts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code Review</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Audit Trail</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Interface</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Separate Environ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Centralize Configs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>... And More!</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What’s not great

- Easy to make mistakes
What’s not great

• Easy to make mistakes
• Audit trail is at will
What’s not great

- Easy to make mistakes
- Audit trail is at will
- Difficult to run long and complex logic
Varying this query

UPDATE products
SET name='julies-product'
WHERE id=1
A more complex version

UPDATE products
SET name='julies-product'
FROM brands
WHERE
  brands.name='julies-store'
  AND products.status='active'
  AND products.name IS NULL
Strategy 2: Run Scripts Locally
Run scripts locally

1. Write a script
Write the script in Python

```
UPDATE products
SET name='julies-product'
WHERE id=1
```
Run scripts locally

1. Write a script
2. Connect to remote database
Run scripts locally

1. Write a script
2. Connect to remote database
3. Run the script
Run the script

```python
python update_product_name.py
   --product_name='julies-product'
   --product_id=1
```
Run the script
(but do a dry-run!)

```python
python update_product_name.py
  --product_name="julies-product"
  --product_id=1
  --dry-run
```
What’s great

- Reusable
What’s great

- Reusable
- Easy to manipulate outputs
What’s great

- Reusable
- Easy to manipulate outputs
- Access to common code
<table>
<thead>
<tr>
<th>Benefits</th>
<th>Raw SQL</th>
<th>Local Scripts</th>
<th>Existing Server</th>
<th>Task Runner</th>
<th>Script Runner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Editing</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complex Logic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Long Scripts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code Review</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Audit Trail</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Interface</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Separate Environ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Centralize Configs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>... And More!</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What’s not great

• Easy to make mistakes
What’s not great

- Easy to make mistakes
- Logs are only available locally
What’s not great

- Easy to make mistakes
- Logs are only available locally
- Network disconnections
Varying this query

```
UPDATE products
SET name='julies-product'
WHERE id=1
```
Doing this 50 million times

UPDATE products
SET name='product-1'
WHERE id=1

UPDATE products
SET name='product-2'
WHERE id=2

UPDATE products
SET name='product-3'
WHERE id=3

UPDATE ...

@jqiu25
Strategy 3: Run Scripts on an Existing Server
Run on an existing server

1. Write a script
Run on an existing server

1. Write a script
2. Get the script onto a server
Run on an existing server

1. Write a script
2. Get the script onto a server
3. SSH and run inside a session
Maia McCormick @justin

heyyyyyy
hey
jenkins is maybe down?
returning 504s on browser, can't ssh in
Maia McCormick  3:41 PM
heyyyyyy @justin
heyyyyy
hey
jenkins is maybe down?
returning 504s on browser, can't ssh in

Julie Qiu  3:43 PM
same re: ssh ^
Sorry
Maia McCormick 3:55 PM
jenkins back up (rebooted the instance)
What’s great

• Ability to run long scripts
What’s great

- Ability to run long scripts
- Reliable network connectivity
What’s great

- Ability to run long scripts
- Reliable network connectivity
- Infrastructure already exists
<table>
<thead>
<tr>
<th>Benefits</th>
<th>Raw SQL</th>
<th>Local Scripts</th>
<th>Existing Server</th>
<th>Task Runner</th>
<th>Script Runner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Editing</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complex Logic</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Long Scripts</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code Review</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Audit Trail</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Interface</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Separate Environ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Centralize Configs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>... And More!</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What’s not great

- Scripts can affect resources on your server
What’s not great

• Scripts can affect resources on your server
• Not user friendly
What’s not great

- Scripts can affect resources on your server
- Not user friendly
- No persistent audit trail
Strategy 4: Use a Task Runner
Use a task runner

1. Write a script
Use a task runner

1. Write a script
2. Code review and run tests
Use a task runner

1. Write a script
2. Code review and run tests
3. Input arguments and run
Project Script Runner - Jenkins

This build requires parameters:

**BUILD_COMMIT**
master
commit to be built

**SCRIPT_AND_ARGS**
update_product_name.py --product_id=1 --product_name="julies-product"
name of the script to run along with the args

**EMAIL**
julie@shopspring.com
e-mail to send notifications to (optional)

**SQLALCHEMY_DATABASE_URI**
postgresql+psycopg2://script_runner:script_runnerpass@localhost:5432/branded
url of the database you want to connect to

**SQLALCHEMY_RO_DATABASE_URI**
url of the read only database you want to connect to

**LOOKUPD_HTTP_ADDR**
http address of the nsq lookupd that you want to use

**REDIS_DATABASE_HOST**
host name of the redis cluster to which you would like to connect

**DB_STATEMENT_TIMEOUT**
1000

**RUN_MODE**
dev

Build
15:27:21 Started by user Mihai_Pora
15:27:21 Cloning the remote Git repository
15:27:21 Using shallow clone
15:27:21 Avoid fetching tags
15:27:21 Cloning repository git@github.com:jellolabs/branded.git
15:27:21 > git init /var/lib/jenkins/jobs/Script Runner - Jenkins/workspace89 # timeout=10
15:27:21 Fetching upstream changes from git@github.com:jellolabs/branded.git
15:27:21 > git --version # timeout=10
15:27:21 using GIT_SSH to set credentials Branded Github Deploy Key
15:27:21 > git fetch --no-tags --progress git@github.com:jellolabs/branded.git +refs/heads/*:refs/remotes/origin/* --depth=1
15:28:27 > git config remote.origin.url git@github.com:jellolabs/branded.git # timeout=10
15:28:27 > git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
15:28:27 > git config remote.origin.url git@github.com:jellolabs/branded.git # timeout=10
15:28:27 Fetching upstream changes from git@github.com:jellolabs/branded.git
15:28:27 using GIT_SSH to set credentials Branded Github Deploy Key
15:28:27 > git fetch --no-tags --progress git@github.com:jellolabs/branded.git +refs/heads/*:refs/remotes/origin/* --depth=1
15:28:32 > git rev-parse origin/master"{commit}" # timeout=10
15:28:32 Checking out Revision 2053703e1fd94c7657ea829a2f449e0b4746e7e (origin/master)
15:28:32 > git config core.spearcheckout "{commit}" # timeout=10
15:28:32 > git checkout -f 2053703e1fd94c7657ea829a2f449e0b4746e7e
15:28:37 Commit message: "[OMS] prevent manual order updates if the order’s state has changed since last display"
15:28:37 > git rev-list --no-walk 2053703e1fd94c7657ea829a2f449e0b4746e7e # timeout=10
15:28:37 [workspace89] $ /bin/bash /tmp/jenkins121846489863985305.sh
15:28:37 SHOPIPPRING_INSTANCE_CONFIG: \
15:28:37 Running script
15:28:37 +++ dirname ./ops/jenkins/py3_script_runner.sh
15:28:37 ++ cd ./ops/jenkins
15:28:37 ++ git rev-parse --show-toplevel
15:28:37 + BRANDED="/data/jenkins/jobs/Script Runner - Jenkins/workspace89"
15:28:37 +++ dirname "/data/jenkins/jobs/Script Runner - Jenkins/workspace89/ops/jenkins/setup_py3.sh"
15:28:37 +++ cd "/data/jenkins/jobs/Script Runner - Jenkins/workspace89/ops/jenkins"
What’s great

- Persistent audit logs
What’s great

- Persistent audit logs
- Code review and automated tests
What’s great

- Persistent audit logs
- Code review and automated tests
- User interface
## Strategy

<table>
<thead>
<tr>
<th>Benefits</th>
<th>Raw SQL</th>
<th>Local Scripts</th>
<th>Existing Server</th>
<th>Task Runner</th>
<th>Script Runner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Editing</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Complex Logic</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Long Scripts</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Code Review</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Audit Trail</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>User Interface</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Separate Environ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Centralize Configs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>... And More!</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What’s not great

• Hard to manage credentials
What’s not great

- Hard to manage credentials
- Environments are not clearly separated
What’s not great

- Hard to manage credentials
- Environments are not clearly separated
- Inputs are not verified
Strategy 5: Build a Script Runner Service
Script runner service

1. Write a script
Script runner service

1. Write a script
2. Code review and run tests
Script runner service

1. Write a script
2. Code review and run tests
3. Choose environment and run
Project Script Runner V2 - Puppetized

This build requires parameters:

BUILD_COMMIT

master

commit to be built

SCRIPT_AND_ARGS

update_product_name.py --product_id=1 --product_name="julies-product"

name of the script to run along with the args

Enter the value for py3/shopspring/tools/(SCRIPT_ANDARGS)

For example, if you would run the script locally with:
py3/shopspring/tools/example/example.py dev -p 123

You would enter `example/example.py dev -p 123` in this field

ENVIRONMENT

dev

EMAIL

julie@shopspring.com

Build
What’s great

• Centralized configuration management
What’s great

- Centralized configuration management
- Separation of environments
What’s great

• Centralized configuration management
• Separation of environments
• User friendly interface
More things we can do

- Parallelize and scale
More things we can do

- Parallelize and scale
- Preview results
More things we can do

• Parallelize and scale
• Preview results
• ... up to you to customize!
Which strategy should I use?
## Strategy

<table>
<thead>
<tr>
<th>Benefits</th>
<th>Raw SQL</th>
<th>Local Scripts</th>
<th>Existing Server</th>
<th>Task Runner</th>
<th>Script Runner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Editing</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Complex Logic</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Long Scripts</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Testing</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Code Review</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Audit Trail</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>User Interface</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Separate Environ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Centralize Configs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>... And More!</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
Make something usable*

*because engineers are also people!
Invest the effort – it’s worth the cost.
Thanks!

Julie Qiu
@jqiu25