The case of django-simple-history and 200M DB rows

Syed Muhammad Dawoud Sheraz Ali

Agenda

- Overview of django-simple-history
- Using django-simple-history in a write-heavy microservice
- Unintended issues and a long cleanup

whoami

- Software Engineer – Pakistan
- Python & Django enthusiast

Syed M. Dawoud Sheraz Ali

Medium

@dawoud.sheraz

SCAN ME

LinkedIn

syed-muhammad-dawoud-sheraz-ali

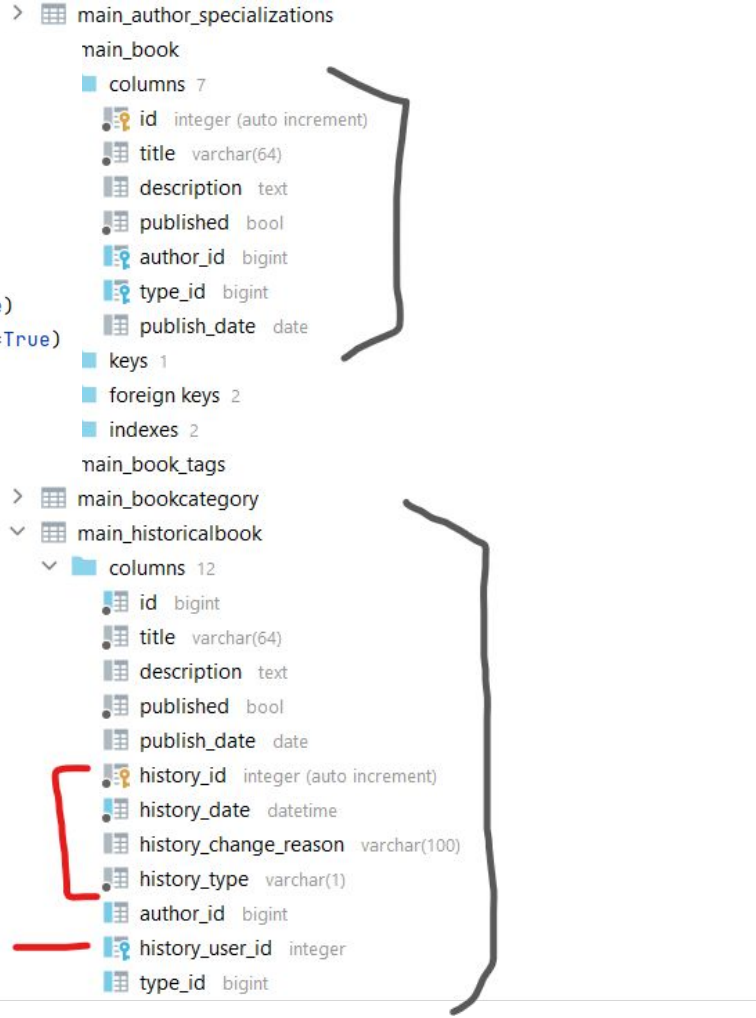
SCAN ME

django-simple-history

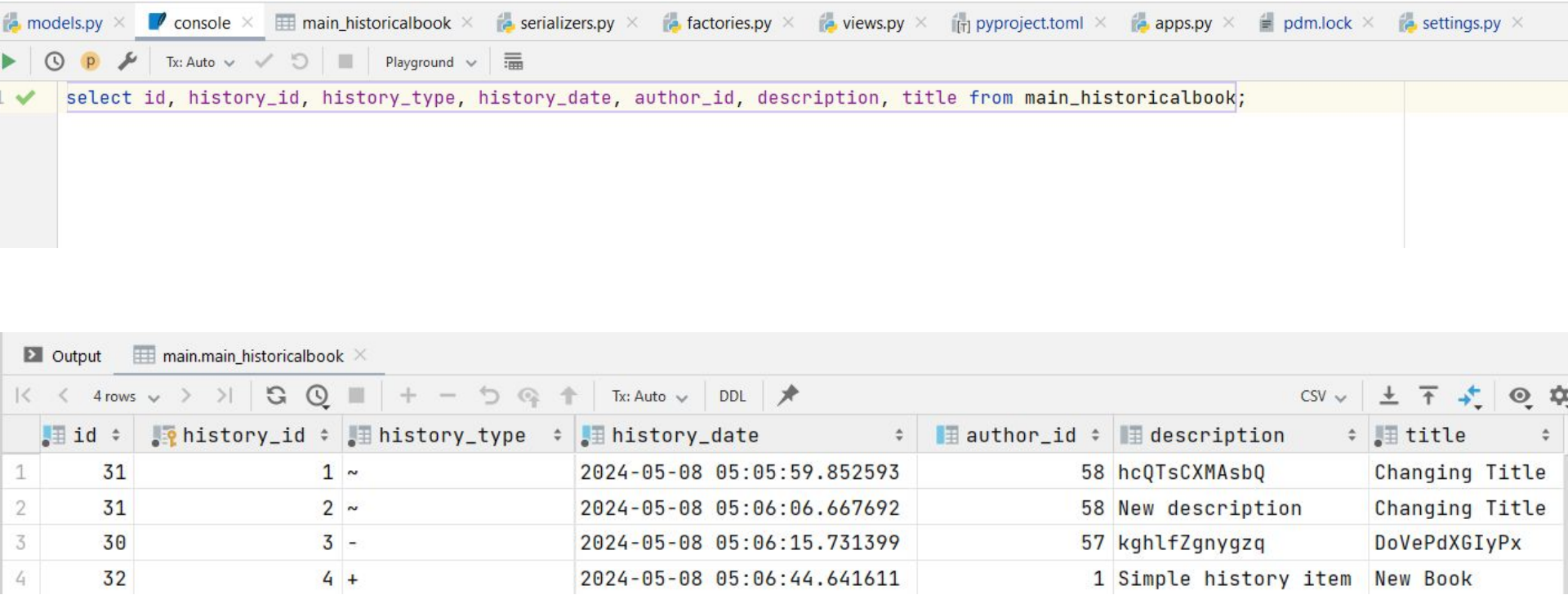
- Very “simple” – create snapshots of object on every change
- Automated persistence of a model object state on create, update, and delete
- Use-cases?
  - Maintain the history of individual model objects
  - Ability to revert back to a previous version of an object
  - Audit and Compliance
  - ...

```
9 usages
class Book(models.Model):
    title = models.CharField(max_length=64)
    description = models.TextField(null=True, blank=True)
    published = models.BooleanField(default=False)
    publish_date = models.DateField(auto_now=False, null=True)
    # Assume book can only have one author and is of a single type (horror, thriller)
    # this is only for simplicity
    author = models.ForeignKey(Author, related_name="books", on_delete=models.SET_NULL, null=True)
    type = models.ForeignKey(BookCategory, related_name="books", on_delete=models.SET_NULL, null=True)
    tags = models.ManyToManyField(Tags)

    history = HistoricalRecords()
```



- Create instance of HistoricalRecords() on model to start tracking history


models.py
console
main_historicalbook
serializers.py
factories.py
views.py
pyproject.toml
apps.py
pdm.lock
settings.py
Tx: Auto
Playground
select id, history_id, history_type, history_date, author_id, description, title from main_historicalbook;
Output
main.main_historicalbook
4 rows
Tx: Auto
DDL
CSV
id history_id history_type history_date author_id description title
1 31 1 ~ 2024-05-08 05:05:59.852593 58 hcQTsCXMAsbQ Changing Title
2 31 2 ~ 2024-05-08 05:06:06.667692 58 New description Changing Title
3 30 3 - 2024-05-08 05:06:15.731399 57 kghlfZgnygzq DoVePdXGIyPx
4 32 4 + 2024-05-08 05:06:44.641611 1 Simple history item New Book

Django-simple-history: Under the hood

- New table starting with **historical_** is created
- On every save(), a new row is created in historical table
  - Simple-history utilize Django's post_save signal to do this
- Additional columns in historical table
  - history_user: the user performing the operation on object
  - history_date: the datetime of the operation
  - history_change_reason: the reason for the change, null by default
  - history_id: the primary key for the historical table, does not use original table's id for PK as there will be multiple entries for it
  - **history_type: + for create, ~ for update, and - for delete**

Features in simple-history

- Exclude certain fields from history
- Admin integration
- Querying history from main object
- Customize user tracking
- Calculate difference of two history instances
- Cleanup old and duplicate history
- .... And many others

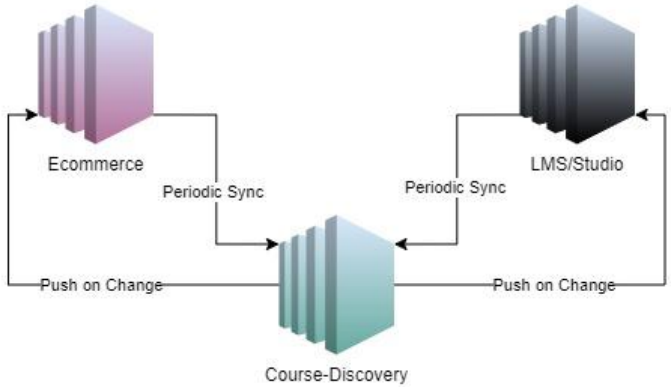

SIMPLE-HISTORY
LOOKS GOOD
WHAT'S WRONG THEN?
imgflip.com

Background

- Services based Company, client/customer of our team – edx.org/2U
- edx.org – uses Open edX, one of largest Open source EdTech platforms
- Various systems in Open edX
  - Edx-platform (LMS & Studio)
  - Discovery (Catalog Service)
  - Ecommerce
  - Credentials
  - ...
- Microservices Architecture with Data Redundancy

Discovery Catalog Service

- Managing marketing information
- Data Aggregator
  - Periodically pull data from Studio and Ecommerce
  - Push to Studio and Ecommerce on changes in Discovery
- Data additions and updates across all 3 backends is very frequent



Discovery Catalog Service

- Uses django-simple-history on core/important models
  - Course, CourseRun, Seats, Program
- Beneficial in debugging and locating data sync issues
  - Changes made in multiple systems at once leading to race conditions and data getting out of sync
  - Downstream learner impact

The problem we faced

- Deploying some changes to core models that had migrations
- Stage deployment went as expected (<15 mins)
- Prod deployment “stuck”, 1h+ on deploying
- Retrying the prod deployment ➜ migration already applied error on some core models

What Happened?

- Migrations had applied successfully on core models
- Migrations were being applied to Historical tables, taking way too much time
- Further Investigation ➜ 5 Historical tables aggregated to more than 350 Million DB rows 😱
    - 2 such tables were part of migration
- Why were there so many historical record rows?

Microservices and Data Sync

- Going back – Discovery service responsible for pulling and pushing information from Studio and E-commerce
- Pull happens periodically, push is on every operation in Discovery
- Pull pipeline
  - Fetch all records from Studio and Ecommerce and update values in Discovery
  - Some part of pipeline would execute save() on every pulled object even if there was no change
- Impact? – save() resulted in creating duplicate history row for each object every single day
  - For context, history was added back in 2019, we observed the issue in 2022

Analysis so far

- Pull pipeline running save() – duplicate history objects
- Duplicate history objects ➜ Increasing table size
- Increased table size ➜ migration take long time to execute

Next steps?

1. Clean Duplicates
2. Avoid Duplicates for Future

Clean Duplicates

- Clean_duplicate_history – Management command in simple-history to delete duplicate records
- Specify how many minutes of history to be deleted
- Specific models who duplicate history should be deleted
- Skip certain fields in history diff checks
    - Like skipping `modified` field if inheriting from TimeStamped model

Caveats?

- Cleanup is slow
- Impact on DB performance due to heavy reads
- For each object,
  - All history is fetched
  - Consecutive objects are diff checked
  - If case of no-diff, one object is removed and pointer moves forward

What it did for us?

- Ran it for 5 core models
- Some cleanup jobs were quick, some were not
- Cleanup (approximations)
  - 200 Million Rows
  - 45GB size freed

Avoiding Duplicates

- Utilize django-model-util’s FieldTracker
- On save(), if object does not have any changes, skip history creation

Closing Thoughts

- Simple-history – simple and effective package
- Using in a highly data changing microservices ecosystem ➜ unexpected DB size
- Caught us by surprise
- Fix was simple but needed some effort

References

1. https://django-simple-history.readthedocs.io/en/latest/
2. https://openedx.org/
3. https://github.com/openedx/course-discovery
4. https://django-simple-history.readthedocs.io/en/latest/utils.html#clean-duplicate-history
5. https://django-model-utils.readthedocs.io/en/latest/utilities.html#field-tracker

Medium

@dawoud.sheraz

SCAN ME

LinkedIn

syed-muhammad-dawoud-sheraz-ali

SCAN ME

DawoudSheraz

SCAN ME