Trimmer

This is the blockchain trimming logic which allows blockchains to be kept to a manageable size while maintaining data integrity (exact rates and ages still to be determined).

Configurable by node

- `TRIGGER_RATE` - for example every 10,000 blocks

Protocol

- `MAX_BLOCK_AGE` - 72 hours

Overview

- every `TRIGGER_RATE` number of blocks the validator will check to see if any older blocks need trimming (to keep blockchain a manageable size)
  - will auto-increment using redis cache
- calculate historical point in time using `stop_mark = (now - MAX_BLOCK_AGE)`
- any blocks younger than the `stop_mark` will remain untouched and any older ones trimmed off

Block hash	Date	*logic (not stored on model)*	status
08ae5	1:33pm UTC	younger than stop_mark	on_blockchain
20344	1:31pm UTC	younger than stop_mark	on_blockchain
9f383	1:26pm UTC	younger than stop_mark	on_blockchain
q3c42	1:25pm UTC	older than stop_mark	pending_removal
46bb9	1:23pm UTC	older than stop_mark	pending_removal
f8af5	1:22pm UTC	older than stop_mark	pending_removal
51c34	1:20pm UTC	older than stop_mark	removed

Logic

- can just keep this table in postgres
  - atomic operations
  - first step is to add in the new trim marker giving it a **`HEAD_BLOCK_HASH`**, time, and `on_blockchain`
- within a new **atomic transaction**
  - check if there are any `pending_removal` blocks
  - if so, **`return`**
    - this is because that indicates that a celery trimmer is already running
  - if no existing `pending_removal` are found, mark any necessary `on_blockchain` rows as `pending_removal`
  - what to mark as `pending_removal`
    - first check if there are at least 4 total existing rows first
    - check if there are **at least 3 unique block hashes left younger than the** `stop_mark`
    - this ensures that the head block and the latest seed block are never trimmer off
      - **`HEAD_BLOCK_HASH`**
      - some padding
      - **Seed Block** (needed for root account file / CV syncs)
    - mark all older `on_blockchain` rows as `pending_removal`
- if within that transaction *any* `on_blockchain` rows were converted to `pending_removal` then kick of the celery trimmer task
  - the atomic behavior (so no concurrent operations allowed on that table) along the `on_blockchain` to `pending_removal` change tracking prevents all possible race conditions

        - only 1 process can make changes to that table at a time
        - only the process that updated rows from `on_blockchain` to `pending_removal` can trigger a celery task
        - only that celery task can mark rows from `pending_removal` to `removed`
- celery trimmer task
    - start with oldest root account file and begin working through old blocks - oldest to newest
    - build and write the new root account file
    - trim off/delete the old blockchain
    - update all trimmed blocks from `pending_removal` to `removed`